EDB Postgres™ Replication Server

with Multi-Master Support

User's Guide

EDB Postgres™ Replication Server 6.2

May 7, 2019

EDB Postgres™ Replication Server User's Guide
by EnterpriseDB® Corporation
Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

EnterpriseDB Corporation, 34 Crosby Drive, Suite 201, Bedford, MA 01730, USA
T +1 781 357 3390 **F** +1 978 467 1307 **E** info@enterprisedb.com **www**.enterprisedb.com

Table of Contents

1 Introduction 11
1.1 What's New 13
1.2 Typographical Conventions Used in this Guide 14
1.3 Other Conventions Used in this Guide 14
1.4 How to Use This Guide 15
2 Overview 16
2.1 Why Use Replication 16
2.1.1 Offloading Reporting and Business Intelligence Queries 16
2.1.2 Using Warm Standby Servers 16
2.1.3 Testing Systems in Parallel 16
2.1.4 Migrating Data 17
2.1.5 Write Availability 17
2.1.6 Write Scalability 17
2.1.7 Localized Data Access 17
2.2 Replication Concepts and Definitions 17
2.2.1 Comparison of Single-Master and Multi-Master Replication 18
2.2.2 Publications and Subscriptions 18
2.2.3 Single-Master (Master-to-Slave) Replication 22
2.2.4 Multi-Master Replication 23
2.2.5 Asynchronous 25
2.2.6 Snapshot and Synchronization Overview 25
2.2.7 Snapshot-Only Publications 27
2.2.8 Snapshot Replication 27
2.2.9 Synchronization Replication with the Trigger-Based Method 28
2.2.10 Synchronization Replication with the Log-Based Method 29
2.2.11 Multi-Master Parallel Replication 37
2.2.12 Table Filters 38
2.3 xDB Replication Server Components and Architecture 43
2.3.1 Physical Components 43
2.3.2 Logical Components 56
2.3.3 xDB Replication System Examples 60
2.4 Designing a Replication System 73

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.4.1 General Steps 73
2.4.2 Design Considerations 74
2.4.3 Restrictions on Replicated Database Objects 76
2.4.4 Performance Considerations 79
2.4.5 Distributed Replication 81
3 Installation and Uninstallation 85
3.1 Installing With Stack Builder or StackBuilder Plus 86
3.2 Installing from the Command Line 108
3.3 Installing the xDB RPM Package 113
3.4 Installing xDB on an SLES 12 Host 119
3.5 Post-Installation Host Environment 121
3.6 Uninstalling xDB Replication Server 122
3.7 Uninstalling the xDB RPM Package 128
4 Introduction to the xDB Replication Console 130
4.1 xDB Replication Console Tool Bar 131
4.1.1 Refresh 131
4.1.2 Create Publication 131
4.1.3 Publication Management 132
4.1.4 Create Subscription 133
4.1.5 Subscription Management 133
4.2 Saving Server Login Information 134
4.2.1 Server Login File 134
4.2.2 Security Risks of Saved Server Login Information 136
5 Single-Master Replication Operation 138
5.1 Prerequisite Steps 138
5.1.1 Setting Heap Memory Size for the Publication and Subscription Servers 138
5.1.2 Enabling Synchronization Replication with the Log-Based Method 139
5.1.3 Enabling Access to the Database Servers 140
5.1.4 Preparing the Publication Database 142
5.1.5 Preparing the Subscription Database 153
5.1.6 Verifying Host Accessibility 157
5.2 Creating a Publication 163
5.2.1 Registering a Publication Server 163
5.2.2 Adding a Publication Database 167

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.2.3 Adding a Publication 172
5.2.4 Control Schema Objects Created for a Publication 178
5.3 Creating a Subscription 191
5.3.1 Registering a Subscription Server 191
5.3.2 Adding a Subscription Database 194
5.3.3 Adding a Subscription 197
5.3.4 Subscription Metadata Object 202
5.4 On Demand Replication 203
5.4.1 Performing Snapshot Replication 204
5.4.2 Performing Synchronization Replication 207
5.5 Managing a Subscription 210
5.5.1 Updating a Subscription Server 211
5.5.2 Updating a Subscription Database 213
5.5.3 Updating a Subscription 217
5.5.4 Enabling/Disabling Table Filters on a Subscription 219
5.5.5 Removing a Subscription 224
5.5.6 Removing a Subscription Database 227
5.6 Performing Controlled Switchover 230
5.6.1 Controlled Switchover Overview 230
5.6.2 Controlled Switchover Steps 231
5.7 Performing Failover 235
5.8 Optimizing Performance 235
5.8.1 Optimizing Snapshot Replication 235
5.8.2 Optimizing Synchronization Replication 238
6 Multi-Master Replication Operation 247
6.1 Prerequisite Steps 247
6.1.1 Setting Heap Memory Size for the Publication Server 247
6.1.2 Enabling Synchronization Replication with the Log-Based Method 247
6.1.3 Preparing the Master Definition Node 248
6.1.4 Preparing Additional Master Nodes 249
6.1.5 Verifying Host Accessibility 251
6.2 Creating a Publication 253
6.2.1 Registering a Publication Server 253
6.2.2 Adding the Master Definition Node 254

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.2.3 Adding a Publication 257
6.3 Creating Additional Master Nodes 265
6.4 Control Schema Objects Created in Master Nodes 274
6.5 On Demand Replication 274
6.5.1 Performing Snapshot Replication 274
6.5.2 Performing Synchronization Replication 278
6.6 Conflict Resolution 282
6.6.1 Configuration Parameter and Table Setting Requirements 282
6.6.2 Conflict Types 283
6.6.3 Conflict Detection 285
6.6.4 Conflict Resolution Strategies 286
6.6.5 Conflict Prevention – Uniqueness Case 287
6.6.6 Conflict Prevention with an MMR-Ready Sequence 287
6.6.7 Automatic Conflict Resolution Example 302
6.6.8 Custom Conflict Handling 305
6.6.9 Manual Conflict Resolution for the Trigger-Based Method 317
6.6.10 Manual Conflict Resolution for the Log-Based Method 349
6.7 Viewing Conflict History 363
6.8 Updating the Conflict Resolution Options 366
6.9 Enabling/Disabling Table Filters on a Master Node 370
6.10 Switching the Master Definition Node 375
6.11 Ensuring High Availability 378
6.11.1 Significance of the Controller Database 379
6.11.2 Automatic Switchover of the Controller Database 380
6.11.3 Switching an Active Controller Database 380
6.11.4 Restarting with an Alternate Controller Database 380
6.12 Optimizing Performance 381
7 Common Operations 383
7.1 Selecting Tables with the Wildcard Selector 383
7.1.1 Wildcard Selector Patterns 383
7.1.2 Using the Wildcard Selector 387
7.2 Creating a Schedule 402
7.3 Managing a Schedule 410
7.3.1 Updating a Schedule 410

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.3.2 Removing a Schedule....414
7.4 Viewing Replication History....419
7.4.1 All Replication History....419
7.4.2 Hiding Synchronizations With Zero Transaction Counts....422
7.4.3 Shadow Table History....425
7.5 Managing History....427
7.5.1 Scheduling Shadow Table History Cleanup....428
7.5.2 Cleaning Up Shadow Table History....434
7.5.3 Cleaning Up Replication History....438
7.5.4 Cleaning Up Event History....443
7.6 Managing a Publication....443
7.6.1 Updating a Publication Server....444
7.6.2 Updating a Publication Database....450
7.6.3 Updating a Publication....455
7.6.4 Updating the Set of Available Table Filters in a Publication....468
7.6.5 Validating a Publication....472
7.6.6 Removing a Publication....479
7.6.7 Removing a Publication Database....483
7.7 Switching the Controller Database....487
7.8 Replicating DDL Changes....491
7.8.1 DDL Change Replication Process....496
7.8.2 DDL Change Replication Using the xDB Replication Console....497
7.9 Loading Tables From an External Data Source (Offline Snapshot)....502
7.9.1 Non-Batch Mode Synchronization....502
7.9.2 Offline Snapshot Configuration Options....504
7.9.3 Single-Master Replication Offline Snapshot....505
7.9.4 Multi-Master Replication Offline Snapshot....506
7.10 Replicating Postgres Partitioned Tables....508
7.10.1 Creating a Postgres 9.x Partitioned Table Publication....513
7.10.2 Creating a Postgres Version 10 or Later Partitioned Table Publication....516
7.11 Using Secure Sockets Layer (SSL) Connections....519
7.11.1 Configuring SSL on a Postgres Database Server....521
7.11.2 Configuring SSL for the Publication Server and Subscription Server....523
7.11.3 Requesting SSL Connection to the xDB Replication Server Databases....526

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.11.4 Summary of SSL Configuration Options....527
8 xDB Replication Server Command Line Interface....529
8.1 Prerequisite Steps....529
8.2 General Usage....530
8.2.1 Running xDB Replication Server CLI....530
8.2.2 Getting Help....531
8.2.3 Supplying the Publication or Subscription Server Login Information....532
8.2.4 Using Encrypted Passwords in Text Files....533
8.2.5 Running xDB Replication Server CLI Using a Parameter File....534
8.2.6 Testing the Command Exit Status....535
8.3 xDB Replication Server CLI Commands....536
8.3.1 Getting Help (help)....537
8.3.2 Printing the Version Number (version)....538
8.3.3 Printing the xDB Replication Server Version Number (repversion)....538
8.3.4 Encrypting Passwords (encrypt)....538
8.3.5 Printing the Time the Server Has Been Running (uptime)....539
8.3.6 Adding a Publication Database (addpubdb)....540
8.3.7 Printing Publication Database IDs (printpubdbids)....544
8.3.8 Printing Publication Database Details (printpubdbidsdetails)....544
8.3.9 Printing the Controller Database ID (printcontrollerdbid)....545
8.3.10 Printing the Master Definition Node Database ID (printmdndbid)....546
8.3.11 Updating a Publication Database (updatepubdb)....546
8.3.12 Removing a Publication Database (removepubdb)....548
8.3.13 Get Tables for a New Publication (gettablesfornewpub)....549
8.3.14 Creating a Publication (createpub)....550
8.3.15 Printing a List of Publications (printpublist)....554
8.3.16 Printing a List of Tables in a Publication (printpublishedtables)....555
8.3.17 Printing a List of Filters in a Publication (printpubfilterslist)....556
8.3.18 Adding Tables to a Publication (addtablesintopub)....556
8.3.19 Removing Tables from a Publication (removetablesfrompub)....560
8.3.20 Adding Table Filters to a Publication (addfilter)....561
8.3.21 Updating Table Filters in a Publication (updatefilter)....563
8.3.22 Removing a Table Filter from a Publication (removefilter)....564
8.3.23 Printing the Conflict Resolution Strategy (printconfresolutionstrategy)....565

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8.3.24 Updating the Conflict Resolution Strategy (updateconfresolutionstrategy) 566
8.3.25 Setting the Master Definition Node (setasmdn) 568
8.3.26 Setting the Controller (setascontroller) 569
8.3.27 Validating a Publication (validatepub) 570
8.3.28 Validating All Publications (validatepubs) 570
8.3.29 Removing a Publication (removepub) 571
8.3.30 Replicating DDL Changes (replicateddl) 572
8.3.31 Adding a Subscription Database (addsubdb) 573
8.3.32 Printing Subscription Database IDs (printsubdbids) 575
8.3.33 Printing Subscription Database Details (printsubdbidsdetails) 576
8.3.34 Updating a Subscription Database (updatesubdb) 577
8.3.35 Removing a Subscription Database (removesubdb) 579
8.3.36 Creating a Subscription (createsub) 580
8.3.37 Printing a Subscription List (printsublist) 581
8.3.38 Enabling Filters on a Subscription or Non-MDN Node (enablefilter) 582
8.3.39 Disabling Filters on a Subscription or Non-MDN Node (disablefilter) 584
8.3.40 Taking a Single-Master Snapshot (dosnapshot) 585
8.3.41 Take a Multi-Master Snapshot (dommrsnapshot) 587
8.3.42 Performing a Synchronization (dosynchronize) 589
8.3.43 Configuring a Single-Master Schedule (confschedule) 590
8.3.44 Configuring a Multi-Master Schedule (confschedulemmr) 592
8.3.45 Print Schedule (printschedule) 595
8.3.46 Updating a Subscription (updatesub) 596
8.3.47 Removing a Subscription (removesub) 598
8.3.48 Scheduling Shadow Table History Cleanup (confcleanupjob) 599
8.3.49 Cleaning Up Shadow Table History (cleanshadowhistforpub) 601
8.3.50 Cleaning Up Replication History (cleanrephistoryforpub) 602
8.3.51 Cleaning Up All Replication History (cleanrephistory) 603
9 Data Validator 604
9.1 Installation and Configuration 604
9.2 Performing Data Validation 607
10 Appendix 621
10.1 Certified and Supported Database Servers and Configurations 621
10.1.1 Certified and Supported Product Versions 621

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.1.2 A Note Regarding Oracle RAC and Oracle Exadata 621
10.1.3 Permitted Configurations and Combinations....622
10.2 Upgrading to xDB Replication Server 6.2 624
10.2.1 Upgrading with the Graphical User Interface Installer....625
10.2.2 Upgrading with the xDB Replication Server RPM Package....629
10.2.3 Updating the Publication and Subscription Server Ports 634
10.3 Resolving Problems....635
10.3.1 Error Messages 635
10.3.2 Where to Look for Errors....645
10.3.3 Common Problem Checklist....648
10.3.4 Troubleshooting Areas....650
10.4 Miscellaneous xDB Replication Server Processing Topics 663
10.4.1 Publication and Subscription Server Configuration Options....663
10.4.2 Encrypting the Password in the xDB Replication Configuration File 678
10.4.3 Writing a Cron Expression....679
10.4.4 Disabling Foreign Key Constraints for Snapshot Replications 681
10.4.5 Quoted Identifiers and Default Case Translation 683
10.4.6 Replicating the SQL Server SQL_VARIANT Data Type....684
10.5 Service Pack Maintenance....686

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

1 Introduction

This document describes the installation, configuration, architecture, and operation of the *EDB xDB Replication Server*. EDB xDB (cross database) Replication Server (referred to hereafter as *xDB Replication Server*) is an asynchronous replication system available for PostgreSQL® and for EDB Postgres™ Advanced Server. The latter will be referred to simply as *Advanced Server*.

xDB Replication Server can be used to implement replication systems based on either of two different replication models – *single-master* (master-to-slave) replication or *multi-master* replication.

Regardless of the chosen replication model, xDB Replication Server is extremely flexible and easy to use.

For single-master replication, PostgreSQL, Advanced Server, Oracle®, and Microsoft® SQL Server® are supported in an assortment of configurations (including cascading replication) allowing organizations to utilize it in multiple use cases with a variety of benefits.

The following are some combinations of cross database replications that xDB Replication Server supports for single-master replication:

- From Oracle to PostgreSQL
- From Oracle to Advanced Server
- From SQL Server to PostgreSQL
- From SQL Server to Advanced Server
- From Advanced Server to Oracle
- From PostgreSQL to SQL Server
- From Advanced Server to SQL Server
- Between PostgreSQL and Advanced Server

Note: Oracle Real Application Clusters (RAC) and Oracle Exadata are not supported by xDB Replication Server. These Oracle products have not been evaluated nor certified with xDB Replication Server. See Section 10.1 for the certified and supported database server products that may be used with xDB Replication Server.

For multi-master replication, xDB Replication Server supports the following configurations:

- Between PostgreSQL database servers
- Between PostgreSQL database servers and Advanced Servers in PostgreSQL compatible mode
- Between Advanced Servers in PostgreSQL compatible mode

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Between Advanced Servers in Oracle compatible mode

Note: See Section 10.1.3 for detailed information on supported source and target database server configurations.

The reader is assumed to have basic SQL knowledge and basic Oracle, SQL Server, or PostgreSQL database administration skills (whichever are applicable) so that databases, users, schemas, and tables can be created and database object privileges assigned.

This document is organized as follows:

- The remainder of Chapter 1 describes conventions used throughout this user's guide along with suggested sections to read based upon your purpose for using this guide.
- Chapter 2 provides an overview of xDB Replication Server including basic replication concepts and definitions, architecture and components of xDB Replication Server, and design guidelines for setting up a replication system.
- Chapter 3 gives instructions for installing and uninstalling xDB Replication Server.
- Chapter 4 provides an overview of the xDB Replication Console, the graphical user interface for using xDB Replication Server.
- Chapter 5 gives instructions for the configuration and operation of xDB Replication Server for single-master replication systems.
- Chapter 6 gives instructions for the configuration and operation of xDB Replication Server for multi-master replication systems.
- Chapter 7 describes operations that are common to both single-master and multi-master replication systems.
- Chapter 8 describes the xDB Replication Server Command Line Interface, an alternative to the graphical user interface for xDB Replication Server configuration and management.
- Chapter 9 gives instructions for configuration and usage of the Data Validator.
- Chapter 10 is an appendix containing troubleshooting tips, a list of error messages, their causes and resolutions, certified and supported database server products, permitted combinations of database servers in a replication system, xDB Replication Server product upgrade procedures, and other miscellaneous technical information.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

1.1 What's New

The following features have been added to xDB Replication Server version 6.1 to create xDB Replication Server version 6.2:

- Registering your xDB Replication Server product with an EnterpriseDB product license key is no longer required. Thus, all components related to registering the product have been removed. The following are the removed components: 1) the Product Registration dialog box accessed from the xDB Replication Console Help menu, 2) the `license_key` parameter located in the xDB Replication Configuration file, and 3) the xDB Replication Server CLI `registerkey` command.
- Partitioned tables created using the *declarative partitioning* feature of PostgreSQL and Advanced Server version 10 and later can now be replicated in a log-based single-master or multi-master replication system. For more information, see Section 7.10.
- In a single-master replication system, removal of a table from a publication that has one or more existing subscriptions is now permitted as long as the table to be removed is not the parent referenced in a foreign key constraint from a child table that is not being removed as well. Previously, no tables from a publication in a single-master replication system could be removed if there are existing subscriptions. For more information, see Section 7.6.3.2.
- Version 11 of PostgreSQL and Advanced Server are now supported. For more information, see Section 10.1.1.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

1.2 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed terms may be substituted. For example, `[ a | b ]` means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }` means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the preceding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

1.3 Other Conventions Used in this Guide

The following is a list of other conventions used throughout this document.

- This guide applies to both Linux and Windows systems. Directory paths are presented in the Linux format with forward slashes. When working on Windows systems, start the directory path with the drive letter followed by a colon and substitute back slashes for forward slashes.
- Much of the information in this document applies interchangeably to the PostgreSQL and EDB Postgres Advanced Server database systems. The term *Advanced Server* is used to refer to EDB Postgres Advanced Server. The term *Postgres* is used to generically refer to both PostgreSQL and Advanced Server. When a distinction needs to be made between these two database systems, the specific names, PostgreSQL or Advanced Server are used.
- The installation directory path of the PostgreSQL or Advanced Server products is referred to as *`POSTGRES_INSTALL_HOME`*. For PostgreSQL Linux installations,

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

this defaults to `/opt/PostgreSQL/x.x` for version 10 and earlier. For later versions, use the PostgreSQL community packages. For PostgreSQL Windows installations, this defaults to `C:\Program Files\PostgreSQL\x.x`. For Advanced Server Linux installations accomplished using the interactive installer for version 10 and earlier, this defaults to `/opt/PostgresPlus/x.xAS` or `/opt/edb/asx.x`. For Advanced Server Linux installations accomplished using an RPM package, this defaults to `/usr/ppas-x.x` or `/usr/edb/asx.x`. For Advanced Server Windows installations, this defaults to `C:\Program Files\PostgresPlus\x.xAS` or `C:\Program Files\edb\asx.x`. The product version number is represented by `x.x` or by `xx` for version 10 and later.

1.4 How to Use This Guide

The following is a list of suggested sections to read based upon your purpose for using this guide.

- If you are interested in a general overview of xDB Replication Server, read sections 2.1, 2.2, and 2.3.
- If you are planning on implementing a replication system using xDB Replication Server, read sections 2.1, 2.2, 2.3, and 2.4.
- If you are ready to install, configure, and begin using xDB Replication Server for single-master replication, read chapters 3, 4, 5, and 7.
- If you are ready to install, configure, and begin using xDB Replication Server for multi-master replication, read chapters 3, 4, 6, and 7.
- If you are interested in using the command line interface instead of the graphical user interface for operating xDB Replication Server, read Chapter 8.
- If you are interested in using the Data Validator, read Chapter 9.
- For certified and supported database product versions and configuration combinations, read Section 10.1.
- If you are upgrading from a previous version of xDB Replication Server, read Section 10.2.
- If you need help finding a solution to a problem, see Section 10.3.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2 Overview

This chapter defines basic replication terms and concepts, and presents an overview of the components and architecture of xDB Replication Server. The chapter concludes with design guidelines and directions for implementing a replication system using xDB Replication Server.

2.1 Why Use Replication

Replication of data can be employed in a variety of use cases in organizations where it is important to use the same data in multiple settings. This allows users to work with 'real' data that will yield 'real' results that are reliable in more than one setting. Support of both single-master and multi-master replication gives xDB Replication Server a broad range of supported use cases.

Some of the more popular uses of single-master replication include the following:

2.1.1 Offloading Reporting and Business Intelligence Queries

In this use case, users take all or just a subset of data from a production OLTP system and replicate it to another database whose sole purpose is to support reporting queries. This can have multiple benefits: a) reporting loads are removed from the OLTP system improving transaction processing performance, b) query performance improves as well without being subordinated to transactions on the system, and c) in Oracle installations, the reporting server duties can be handled by a product like Advanced Server reducing licensing costs for a reporting server.

2.1.2 Using Warm Standby Servers

When many organizations wish to improve the availability of their data, a cost effective solution is often the use of warm standby servers. These are database servers kept up to date with the online system through replication that can be brought online quickly in the event of a failure in the production system. Warm standby servers can also be used for regular maintenance by gracefully switching over to the standby server so that the production server can be brought offline for regular maintenance.

2.1.3 Testing Systems in Parallel

Often times, upgrading or moving to a new database system requires that the old and new systems be up and running in parallel to allow for testing and comparing results in real time. Replication can be employed in this use case and is frequently used in development and testing environments.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.1.4 Migrating Data

Similar to running in parallel, is the situation where data may be migrated from one system to another in a sort of 'seeding' operation. Replication can be very effective in this situation by quickly copying data.

Some reasons to consider multi-master replication include the following:

2.1.5 Write Availability

In single-master replication, only the master database is available for writes. The slave databases are read-only for applications. If the replicated target databases must be available for write access as well, multi-master replication can be employed for the same use cases as outlined for single-master replication, but with the additional advantage of write access to the slaves.

2.1.6 Write Scalability

In write-intensive applications, multi-master replication allows you to utilize multiple database servers on separate hosts to process write transactions independently of each other on their own master databases. Changes can then be reconciled across master databases according to your chosen schedule.

2.1.7 Localized Data Access

In a geographically dispersed application, local access to the database can be provided to regions of clients. Having the database servers physically close to clients can reduce latency with the database. Multi-master replication allows you to employ a WAN connected network of master databases that can be geographically close to groups of clients, yet maintain data consistency across master databases.

2.2 Replication Concepts and Definitions

xDB Replication Server is a software product that enables the implementation of a replication system. A *replication system* is software and hardware whose purpose is to make a copy of data from one location to another and to ensure the copied data is the same as the original over time.

xDB Replication Server applies the replication system concept to tables of Oracle, SQL Server, PostgreSQL, and Advanced Server database management systems.

The following sections present specific terms and concepts used when discussing xDB Replication Server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.2.1 Comparison of Single-Master and Multi-Master Replication

There are two models of replication systems supported by xDB Replication Server:

- **Single-Master Replication (SMR).** Changes (inserts, updates, and deletions) to table rows are allowed to occur in a designated master database. These changes are replicated to tables in one or more slave databases. The replicated tables in the slave databases are not permitted to accept any changes except from its designated master database. (This is also known as master-to-slave replication.)
- **Multi-Master Replication (MMR).** Two or more databases are designated in which tables with the same table definitions and initial row sets are created. Changes (inserts, updates, and deletions) to table rows are allowed to occur in any database. Changes to table rows in any given database are replicated to their counterpart tables in every other database.

For a single-master replication system, a variety of configurations are supported including:

- Replication between PostgreSQL and Advanced Server databases (between products in either direction)
- Replication *from* Oracle *to* PostgreSQL
- Replication *in either direction* between Oracle and Advanced Server
- Replication *in either direction* between SQL Server and PostgreSQL
- Replication *in either direction* between SQL Server and Advanced Server

For multi-master replication, the participating database servers in a given multi-master replication system must be of the same type:

- PostgreSQL database servers
- PostgreSQL database servers and Advanced Servers operating in PostgreSQL compatible mode
- Advanced Servers operating in PostgreSQL compatible mode
- Advanced Servers operating in Oracle compatible mode

Note: A given database cannot simultaneously participate in both a single-master replication system and a multi-master replication system.

2.2.2 Publications and Subscriptions

xDB Replication Server uses an architecture called *publish and subscribe*. The data to be made available for copying by a replication system is defined as a publication. To get a copy of that data, you must "subscribe" to that publication. The manner in which you subscribe is slightly different for single-master and multi-master replication systems.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In xDB Replication Server a *publication* is defined as a named set of tables and views within a database. The database that contains the publication is called the *publication database* of that publication.

In a single-master replication system, to get a copy of an xDB Replication Server publication, you must create a subscription. An xDB Replication Server *subscription* is a named association of a publication to a database to which the publication is to be copied. This database is called the *subscription database*.

Similar to a single-master replication system, when creating a multi-master replication system, you first define a publication in the publication database. You then add one or more additional databases that you want to participate in this multi-master replication system. As you add each database, it is associated with this replication system. You do not create an explicit, named subscription in a multi-master replication system.

In a single-master replication system, *replication* is said to occur when xDB Replication Server initiates and completes either of the following processes: 1) applies changes that have been made to rows in the publication since the last replication occurred, to rows in tables of the subscription database (called synchronization); or 2) copies rows of the publication to empty tables of the subscription database (called a snapshot). See Section 2.2.6 for further discussion on snapshots and synchronization.

The *subscription tables* are the tables in the subscription database created from corresponding tables or views in the publication.

Note: In a single-master replication system xDB Replication Server creates a table in the subscription database for each view contained in the publication.

In a multi-master replication system, the concept and definition of *replication* is nearly identical to a single-master replication system with the following modifications: 1) synchronization can occur between any pair of databases (referred to as master nodes) participating in the replication system; and 2) a snapshot can occur from the publication database designated as the master definition node to any of the other master nodes.

The following diagrams illustrate some basic single-master replication system examples.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

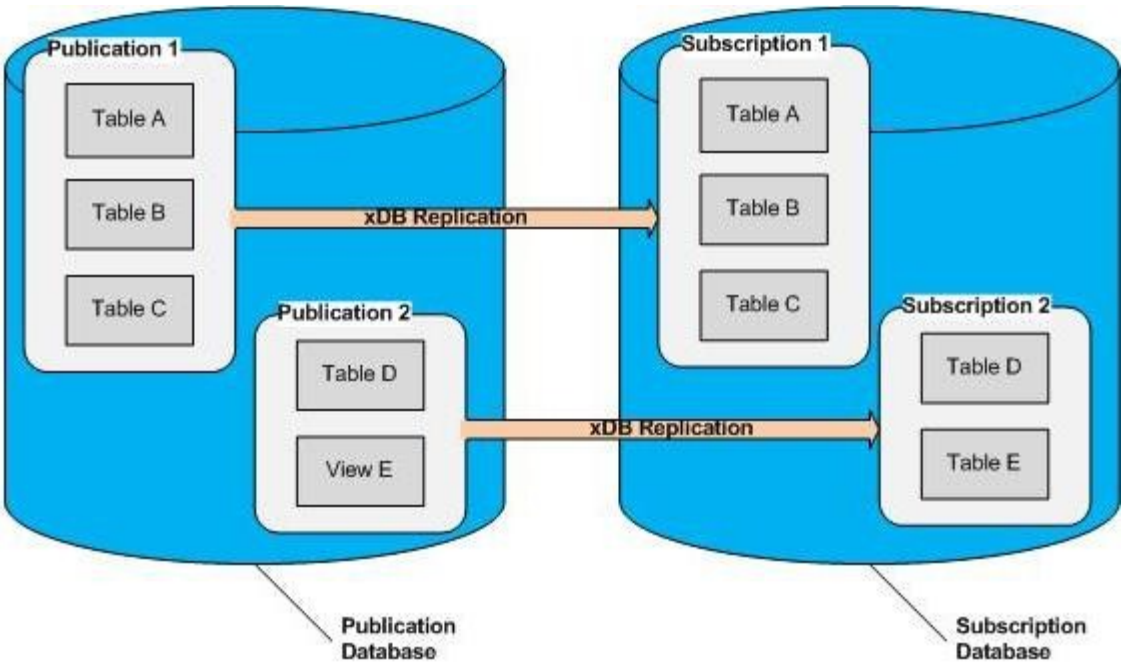


Figure 2-1 - Publications in one database replicating to subscriptions in another database

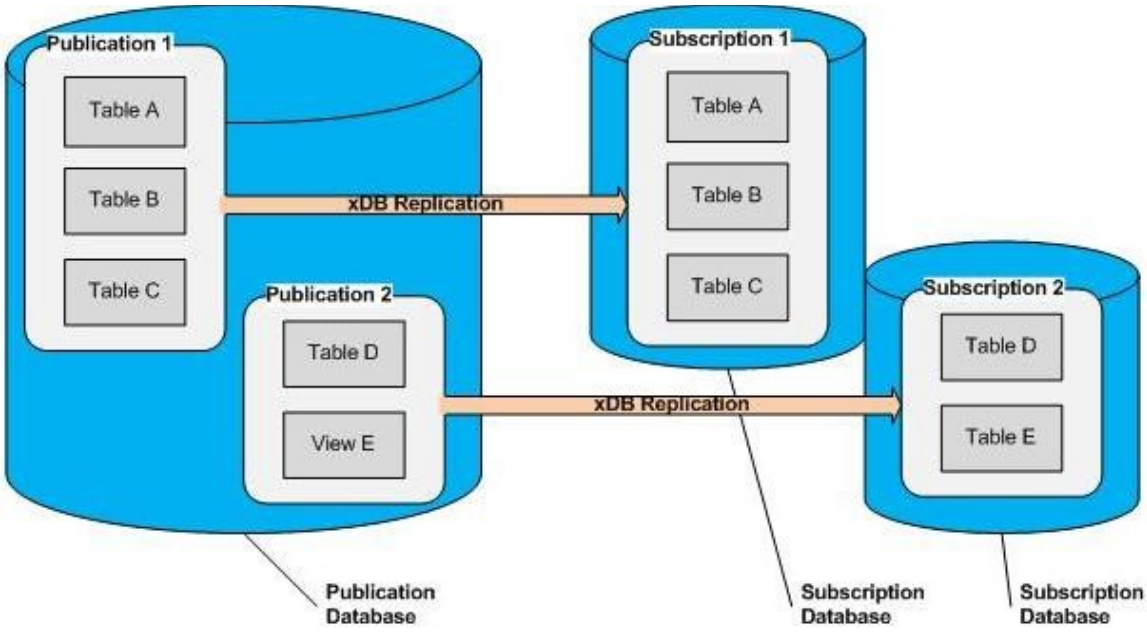


Figure 2-2 - Publications replicating to two subscription databases

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

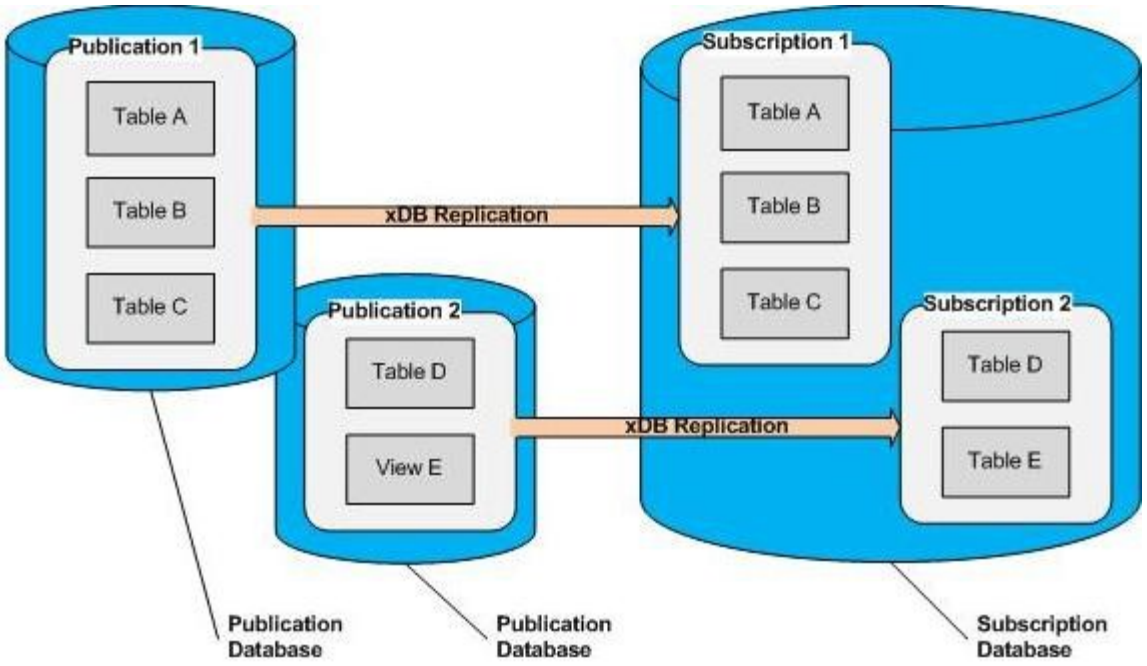


Figure 2-3 - Publications in two databases replicating to one subscription database

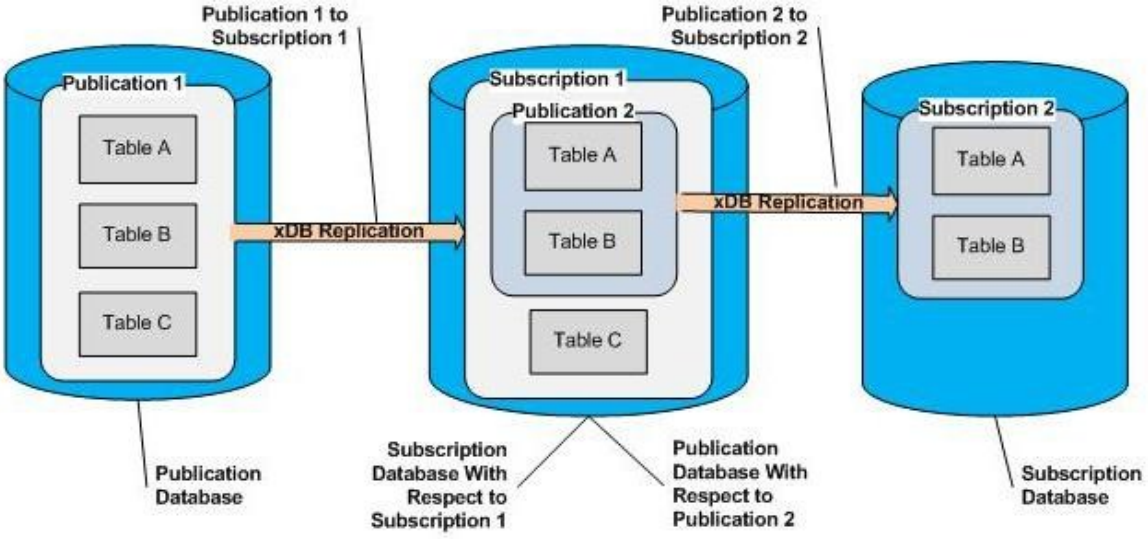


Figure 2-4 – Cascading Replication: Tables used in both a subscription and a publication

The preceding diagram illustrates that a table that has been created as a member of a subscription can be used in a publication replicating to another subscription. This scenario is called *cascading replication*.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following diagram illustrates a multi-master replication system with three master nodes.

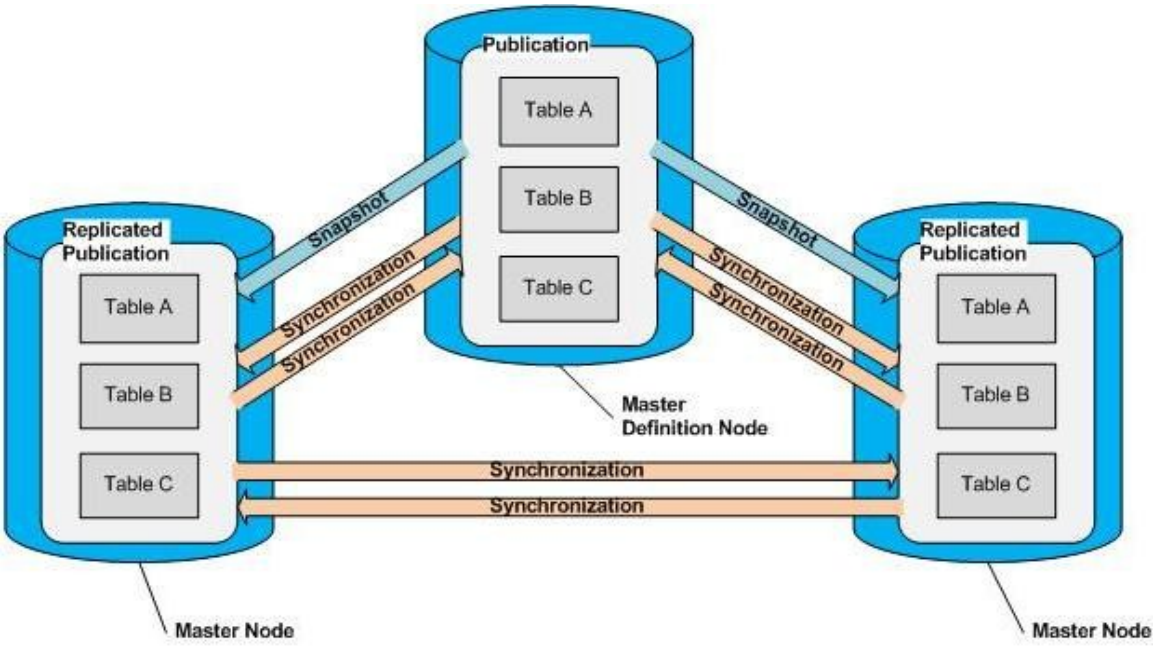


Figure 2-5 – Multi-master replication system

2.2.3 Single-Master (Master-to-Slave) Replication

xDB Replication Server performs *master-to-slave* replication when a single-master replication system is implemented. The publication is the master and the subscription is the slave. In a master-to-slave relationship, changes are propagated in one direction only, from the master to the slave.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

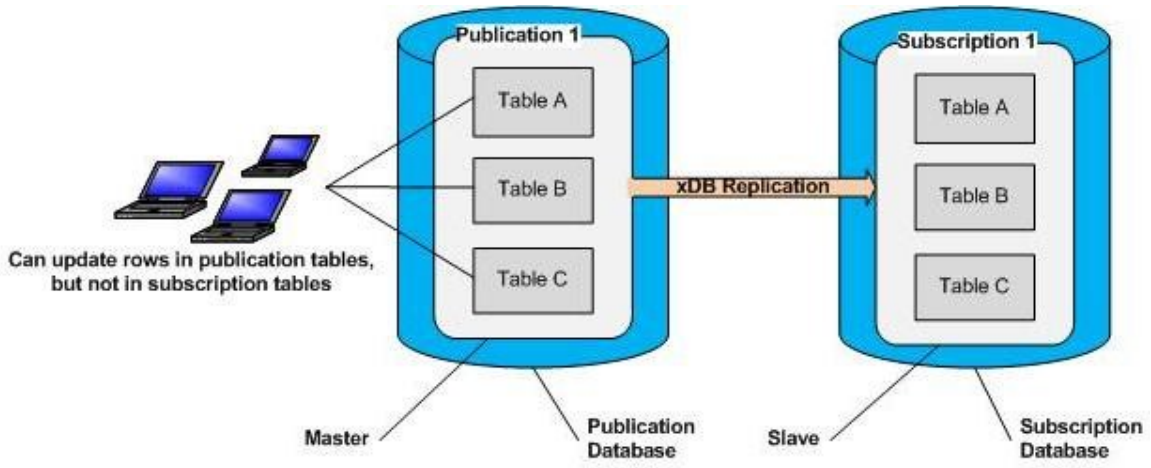


Figure 2-6 – Single-Master (Master-to-Slave) replication

Generally, changes must not be made to the definitions of the publication tables or the subscription tables. If such changes are made to the publication tables, they are not propagated to the subscription and vice versa unless the DDL change replication feature is used as described in Section 7.8. If changes are made to the table definitions without using the DDL change replication feature, there is a risk that future replication attempts may fail.

Changes must not be made to the rows of the subscription tables. If such changes are made, they are not propagated back to the publication. If changes are made to the subscription table rows, it is fairly likely that the rows will no longer match their publication counterparts. There is also a risk that future replication attempts may fail.

2.2.4 Multi-Master Replication

As an alternative to the single-master (master-to-slave) replication model, xDB Replication Server supports multi-master replication.

The following definitions are used when referring to multi-master replication systems.

A *master node* is a database participating in a multi-master replication system.

The database (master node) in which the publication is initially defined is specially designated as the *master definition node* (MDN). There can be only one master definition node at any given time, however, it is possible to change which master node is the master definition node. When it is important to make a distinction between the master definition node and all other master nodes that are not the master definition node, the latter are referred to as *non-MDN nodes*.

The master definition node has the following significance:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The publication is initially created in the master definition node, and the tables comprising the publication must exist in the database to be designated as the master definition node at the time the publication is defined.
- The publication can be initially replicated to other master nodes by means of a snapshot from the master definition node.
- Each subsequent master node added to the replication system must either: 1) contain no tables with the same schema-qualified names as the publication tables in the master definition node; or 2) contain all publication table definitions as they exist in the master definition node with the same schema-qualified names. In the first case, when you add the master node, you select the option to replicate the publication schema from the master definition node. In the second case, you do not select this option.
- The table rows in a master node can be reloaded from the master definition node. The master node tables are truncated and the rows reloaded by a snapshot from the master definition node.

Once the multi-master replication system is defined, changes (inserts, updates, and deletions) to rows of the publication tables on any master node are synchronized to all other master nodes on either an on demand or scheduled basis.

Generally, changes must not be made to the table definitions in any of the master nodes including the master definition node. If such changes are made, they are not propagated to other nodes in the multi-master replication system unless they are made using the DDL change replication feature described in Section 7.8. If changes are made to tables without using the DDL change replication feature, there is a risk that future replication attempts may fail.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

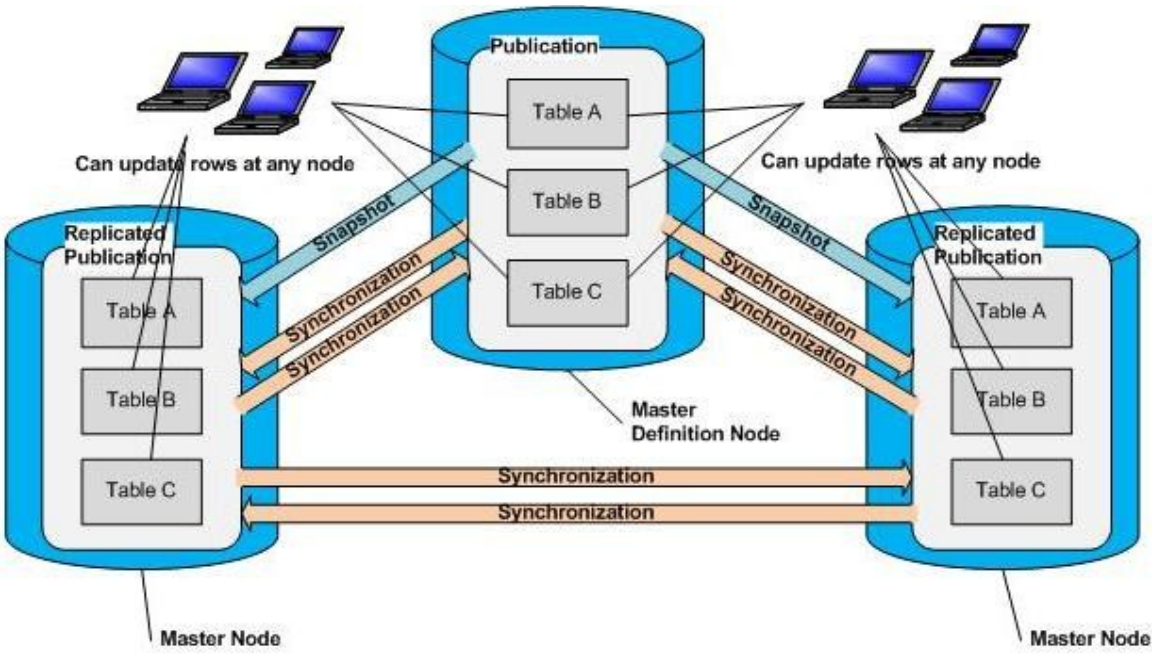


Figure 2-7 - In a multi-master replication system, table rows can be updated at any master node

2.2.5 Asynchronous

xDB Replication Server performs replications *asynchronously*. The systems hosting the databases do not always have to be running continuously in order for successful replication to occur. If one system goes offline, replication resumes when it comes back online if there is still pending data to replicate.

In addition you can create a schedule for your replication system. xDB Replication Server initiates and performs replications regularly according to the assigned schedule. This allows you to run the replication system unattended. See Section 7.2 for directions on creating a schedule.

2.2.6 Snapshot and Synchronization Overview

xDB Replication Server performs two different types of replications. These two main types are called snapshot replication and synchronization replication.

In either method, the *source* tables refer to the tables from which the replication data is originating (the publication in a single-master replication system, or the master node whose changes are being replicated to another master node in a multi-master replication system).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The *target* tables are the tables that are receiving the replication data from the source tables (the subscription tables in a single-master replication system, or the master node receiving changes from another master node in a multi-master replication system).

In *snapshot replication*, all existing rows in the target tables are deleted using the database system's `TRUNCATE` command. The tables are then completely reloaded from the source tables of the publication.

In *synchronization replication*, only the changes (inserts, updates, and deletions) to the rows in the source tables since the last replication are applied to the target tables.

Note: Deletion of all rows in a source table executed by the SQL `TRUNCATE` command results in replication to the target tables only if the log-based method of synchronization replication is used. If the trigger-based method of synchronization replication is used, execution of the `TRUNCATE` command on a source table does not replicate the effect to the target tables. You must perform a snapshot from the source table to the target tables if the trigger-based method is used. (The difference between the trigger-based method and the log-based method is discussed as follows.)

Synchronization replication is implemented using two different methods – the trigger-based method and the log-based method.

In the *trigger-based method* changes to rows in the source tables result in the firing of row-based triggers. These triggers record the changes in shadow tables. The changes recorded in the shadow tables are then periodically extracted from the shadow tables, converted to an in-memory data structure, and applied to the target tables by means of SQL statements executed using JDBC. See Section 2.2.9 for information on the trigger-based method.

In the *log-based method* changes to rows in the source tables are extracted from the Write-Ahead Log segments (WAL files) using asynchronous streaming replication implemented by the logical decoding feature available in Postgres database servers. The extracted changes are converted to an in-memory data structure and applied to the target tables by means of SQL statements executed using JDBC. See Section 2.2.10 for information on the log-based method.

In a multi-master replication system, the manner in which changes accumulated on all master nodes are replicated to all other master nodes is conceptually done in groups identified by the source master node with the changes to be replicated. See Section 2.2.11 for information on this process and the improvement for the log-based method with *parallel replication*.

In a single-master replication system, the very first replication to a newly created subscription must always be done by a snapshot. Subsequent replications can be done by snapshot or by synchronization provided that the publication is not defined as a snapshot-only publication as discussed in Section 2.2.7.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In a multi-master replication system, the very first replication from the master definition node to a newly added master node must always be done by a snapshot. Subsequent replications between master nodes occur by synchronization. However, it is possible to perform subsequent snapshots from the master definition node to any other master node.

2.2.7 Snapshot-Only Publications

When a publication is created in a single-master replication system, the publication can be defined as a *snapshot-only publication*. Replication from a snapshot-only publication can only be done using the snapshot replication method. Synchronization replication is not permitted on a snapshot-only publication.

A snapshot-only publication cannot be created in a multi-master replication system.

See Section 2.4.4 for a discussion of the advantages of using a snapshot-only publication.

2.2.8 Snapshot Replication

In snapshot replication, the target tables are completely reloaded from the source tables. The database system's truncate operation is used to delete all rows from the target tables.

For Oracle and SQL Server only: Oracle and SQL Server target tables are loaded using JDBC batches of `INSERT` statements.

For Postgres only: In general, Postgres target tables are loaded using the JDBC `COPY` command since using truncation and `COPY` is generally faster than if you were to execute an SQL `DELETE` statement against the entire table and then add the rows using JDBC batches of `INSERT` statements. If the `COPY` command fails, the publication server retries the snapshot using JDBC batches of `INSERT` statements.

If the target table (regardless of database type) contains a large object data type such as `BYTEA`, `BLOB`, or `CLOB` then rows are loaded one at a time per batch using an `INSERT` statement. This is to avoid a heap space error resulting from potentially large rows. Loading time can be decreased by allowing multiple inserts per batch, which is done by adjusting the configuration option `lobBatchSize` described in Section 5.8.1.

Note: Advanced Server supports a number of aliases for data types. Such aliases that translate to `BYTEA` are treated as large object data types. See the *Database Compatibility for Oracle Developers Reference Guide* for a listing of Advanced Server data types. (See the *Database Compatibility for Oracle Developer's Guide* for Advanced Server version 9.5 or earlier versions.)

Under certain circumstances, the corresponding Postgres target table created for certain types of Oracle partitioned tables is a set of inherited tables. In these cases, the SQL `DELETE` statement is used on the inherited child tables instead of truncation. See Section 10.4.1.4 for additional information on replicating Oracle partitioned tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

A server configuration option is available that forces the snapshot replication process to use the Oracle database link utility instead of JDBC `COPY` to populate the Postgres target tables from an Oracle publication. Oracle database link provides an additional performance improvement over JDBC `COPY`. See Section 5.8.1 for information on using the Oracle database link option.

See Section 5.8.1 for information on various configuration options to optimize snapshot replication.

2.2.9 Synchronization Replication with the Trigger-Based Method

If a publication in a single-master replication system is created that will be used in synchronization replications with the trigger-based method, the publication server installs an insert trigger, an update trigger, and a delete trigger on each publication table. In a multi-master replication system, each replicated table in each master node employing the trigger-based method has an insert trigger, an update trigger, and a delete trigger.

The publication server also creates a shadow table for each source table on which triggers have been created. A *shadow table* is a table used by xDB Replication Server to record the changes (inserts, updates, and deletions) made to a given source table. A shadow table records three types of record images: For each row inserted into the source table, the shadow table records the image of the inserted row. For each existing row that is updated in the source table, the shadow table records the after image of the updated row. For each row deleted from the source table, the shadow table records the primary key value of the deleted row.

Note: In a multi-master replication system, the before image of an updated row is also stored in the shadow table in order to perform update conflict detection. See Section 6.6 for information on conflict detection in a multi-master replication system.

After each change on the source table, one of the insert, update, or delete triggers is executed. These are row triggers, so for each row affected by the change, the trigger executes. Each execution of the trigger records a row of the appropriate type (insert, update, or deletion) in the shadow table of the corresponding source table.

Though changes made to the source tables since the last replication occurred are applied to the target tables using SQL `INSERT`, `UPDATE`, and `DELETE` statements, the actual SQL statements run against the target tables are not the same SQL statements that were run against the source tables.

When synchronization replication occurs, the publication server executes JDBC batches of SQL statements (also referred to as *transaction sets*) against the target tables. The batches contain an `INSERT` statement for each shadow table row recording an insert operation, an `UPDATE` statement for each shadow table row recording an update operation, and a `DELETE` statement for each shadow table row recording a delete operation. Each batch is executed in one transaction.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Shadow table rows that were applied to target tables can be viewed as shadow table history in the xDB Replication Console (see Section 7.4.3).

Note: A single SQL statement executed against a source table may result in many rows recorded in a shadow table, and therefore, many SQL statements executed against the target table. For example, if a single `UPDATE` statement affects 10 rows in the source table, 10 rows will be inserted into the shadow table – one for each row in the source table that was updated. When the publication server applies the changes to the target table, 10 `UPDATE` statements will be executed.

Note: For greater efficiency, when changes to the source tables consist of SQL statements that each affect a large number of rows, the publication server may employ the use of prepared SQL statements. See Section 5.8.2 for directions on how to control the usage of prepared SQL statements as well as information on various other configuration options to optimize synchronization replication.

2.2.10 Synchronization Replication with the Log-Based Method

In PostgreSQL 9.4 a feature has been introduced called *logical decoding* (also called *logical replication* or *changeset extraction*). This feature provides the capability to extract data manipulation language (DML) changes from the Write-Ahead Log segments (WAL files) in a readable format.

For information on logical decoding see the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/logicaldecoding.html

The key significance of this feature is the ability to capture data changes to the publication tables without impacting the online transaction processing rate against these tables that occurs when using the trigger-based method. The trigger-based method results in the firing of row-level triggers whenever data changes occur, then inserting these data changes into shadow tables for temporary storage before applying the changes to the target databases.

Thus, extracting data changes using logical decoding can be beneficial for improving database server throughput and replication latency.

However, note that the logical decoding interface streams changes for all tables in a given database, which may have a performance overhead associated with it. For example, if a database contains 100 tables, and the user is interested in replicating only a small subset of these tables, say only 20 tables in a single publication, the logical decoding protocol will stream changes for all 100 tables to the publication server. The publication server eventually filters out the changes for the irrelevant 80 tables. However, this results in network overhead caused by the additional changeset load that is not required by the replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Using logical decoding to extract changes from a publication database during xDB synchronization replication is referred to as the *log-based method*.

The following sections describe the basic requirements and concepts for the log-based method of synchronization replication.

2.2.10.1 Requirements and Restrictions

The following are the general requirements and restrictions when using the log-based method for any database of a single-master or multi-master replication system:

- The selection of either the trigger-based method or the log-based method is a characteristic applicable to only the publication database. The choice is made when defining the master database of a single-master replication system (see Section 5.2.2) or the master definition node of a multi-master replication system (see Section 6.2.2).
- The logical decoding feature, and hence the log-based method, is supported beginning with PostgreSQL version 9.4. Therefore, in order to use the log-based method for a publication database, that publication database must be running under PostgreSQL version 9.4 or later, or under Advanced Server version 9.4 or later.
- In a single-master replication system, whether the master database uses the trigger-based method or the log-based method has no additional impact on the rules for choosing the subscription database as described in Section 10.1. For example, even if the log-based method is chosen for the master database, the subscription database may be running on Postgres version 9.4 as well as any supported, earlier version of Postgres, as well as Oracle or SQL Server as described in Section 10.1.
- In a single-master replication system, the master database may contain one or more publications (that is, named sets of tables for replication). This is applicable to a master database using either the trigger-based method or the log-based method.
- It is permissible to have multiple, single-master replication systems running under a publication server where some master databases may use the trigger-based method while others use the log-based method.
- In a multi-master replication system, the selection of either the trigger-based method or the log-based method on the master definition node determines the method for all other master nodes. In other words, if the trigger-based method is chosen for the master definition node, then all other master nodes will use the trigger-based method. If the log-based method is chosen for the master definition node, then all other master nodes will use the log-based method.
- As a consequence of the restriction described in the preceding bullet point, in order to use the log-based method for a multi-master replication system, all of the master nodes of the system must be running under Postgres version 9.4 or later, and all such Postgres database clusters must be configured to use logical decoding for the log-based method.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Selection of the log-based method for any database impacts the configuration of the Postgres database cluster containing that database.

If you plan to use the log-based method with any publication database running under a Postgres database server, the following configuration parameter settings are required in the configuration file, `postgresql.conf`, of that Postgres database server:

- **wal_level.** Set to `logical`.
- **max_wal_senders.** Specifies the maximum number of concurrent connections (that is, the maximum number of simultaneously running WAL sender processes). Set at minimum, to the total number of master databases of single-master replication systems and master nodes of multi-master replication systems on this database server that will use the log-based method.
- **max_replication_slots.** Specifies the maximum number of replication slots. If the database server supports both single-master replication systems and multi-master replication systems, then `max_replication_slots` must be set at minimum to the sum of the requirements for both replication systems. For support of SMR systems, the minimum requirement is the total number of master databases of the single-master replication systems that will use the log-based method. For support of MMR systems, the minimum requirement is the total number of master nodes in the multi-master replication system multiplied by the number of master nodes residing on this database server. For information, see Section 2.2.10.4.
- **track_commit_timestamp.** Set to `on`. This configuration parameter applies only to Postgres database servers of version 9.5. See Section 6.6.1 for additional information.

Also see Section 5.1.2 for setting these parameters for a single-master replication system. See Section 6.1.2 for a multi-master replication system.

In addition, the `pg_hba.conf` configuration file of the Postgres database server must contain an entry permitting `REPLICATION` access for each database using the log-based method running on the database server. The access must be permitted to the publication database user specified when creating the publication database definition using the xDB Replication Console (see Section 5.2.2 for a single-master replication system or Section 6.2.2 for a multi-master replication system) or the xDB Replication Server Command Line Interface (CLI) (see Section 8.3.6).

See Section 5.1.6.3 for setting `REPLICATION` access for a single-master replication system. See Section 6.1.5 for a multi-master replication system.

For configuration options in the publication server configuration file that are specifically applicable to the log-based method see Section 10.4.1.15.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.2.10.2 Logical Replication Slots

When using the log-based method on a publication database, the underlying logical decoding framework exposes the data changes (the *changeset stream*) by means of a logical replication slot.

A *logical replication slot* represents a changeset stream and applies to a single database. The xDB Replication Server assigns a unique identifier, called the *slot name*, to each logical replication slot it creates in the form `xdb_dboid_pubid` where `dboid` is the publication database object identifier (OID) and `pubid` is the publication ID assigned by the xDB Replication Server. All slot names are unique within a Postgres database cluster.

Thus, for each single-master replication system using the log-based method, a replication slot is required for the publication database of each such system.

For a multi-master replication system using the log-based method, each master node requires a replication slot.

The maximum number of replication slots permitted for a database server is controlled by the `max_replication_slots` configuration parameter in the `postgresql.conf` file. Therefore this configuration parameter must be set to a large enough value to account for all publication databases defined with the log-based method of single-master replication systems running on the database server as well as all master nodes of a multi-master replication system defined with the log-based method running on the database server. Additional replication slots are required to support the usage of replication origin (see Section 2.2.10.4).See Section 5.1.2 for additional information on configuration parameters for single-master replication systems. See Section 6.1.2 for multi-master replication systems.

2.2.10.3 Streaming Replication with the WAL Sender Process

The changeset stream is accessible to the xDB publication server by the *WAL sender process* (*walsender*) using the streaming replication protocol.

The xDB publication server connects using the `walsender` interface through which changes are streamed on a continual basis. The continuous streaming eliminates the need for explicitly polling for changes.

The following are the basic synchronization steps using the log-based method:

1. A streaming replication connection to the database server is opened using `libpq` to establish a `walsender` communication channel.

2. A separate thread is used to monitor data changes streamed through the `walsender` interface.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3. As the data changes become available, they are transformed to populate an in-memory cache.

4. On the next scheduled interval, the in-memory cached data changes are applied to each of the target databases in JDBC batches of SQL statements (referred to as *transaction sets*) in the same manner as described in Section 2.2.9 for the trigger-based method. If one or more target database servers are not accessible, the data changes are saved in a local file on the host running the publication server. Section 2.2.10.5 for information on in-memory caching and data persistence.

5. The value of the WAL segment's log sequence number (LSN) identifying the last set of applied changes based on the last replicated transaction is updated. The update is confirmed to the database server.

6. The applied data changes are cleared from the in-memory cache.

7. Steps 3 through 6 are repeated.

Note: A single SQL statement executed against a source table may result in many rows modified and returned in the changeset stream, and therefore, many SQL statements executed against the target table. For example, if a single `UPDATE` statement affects 10 rows in the source table, 10 rows will be returned in the changeset stream – one for each row in the source table that was updated. When the publication server applies the changes to the target table, 10 `UPDATE` statements will be executed.

2.2.10.4 Replication Origin

Starting with Postgres version 9.5, a feature called *replication origin* has been introduced to the logical decoding framework. Replication origin allows an application to identify, label, and mark certain aspects of a logical decoding session.

For information on replication origin see the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/replication-origins.html

For the log-based method of synchronization replication, this provides performance improvement provided that the master nodes are running under Postgres version 9.5 or later.

As previously described, the log-based method uses the WAL files to obtain the changes applied to the publication tables. After the changes are retrieved through the `walsender` interface, the publication server applies the set of changes to the other master nodes using transaction sets consisting of JDBC batches of SQL statements. When these changes are applied to the tables in the other target master nodes, the same changes are also recorded in the WAL files of each database server hosting the target master nodes.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

These redundant or "replayed" changes are included in the changeset stream received by the publication server. These replayed changes must be ignored and not applied since they are duplicates of all changes that have already been applied to the target tables through the JDBC batches.

The replayed changes result in performance overhead as all such changes are transmitted over the network from the database server to the publication server, and then the publication server must discard such redundant changes.

With the replication origin feature, the publication server is able to set up the logical decoding sessions so that these replayed changes are not included in the changeset stream transmitted over the network to the publication server thus eliminating this performance overhead.

The following are the conditions under which replication origin is used:

- Replication origin applies to multi-master replication systems only, not to single-master replication systems.
- Replication origin eliminates streaming of replayed changes only from Postgres versions 9.5 or later. Replayed changes are still included in the changeset stream from Postgres version 9.4, but are discarded by the publication server. Thus multi-master replication systems consisting of both Postgres versions 9.4 and 9.5 utilize the replication origin advantage on the 9.5 database servers.
- The `max_replication_slots` configuration parameter must be set at a certain minimal level to ensure that the publication server can create the additional replication slots for replication origin.

For each master node database, in addition to the replication slot used for the changeset stream, an additional number of replication slots is required – one additional slot corresponding to every other master node to support the replication origin usage. Thus, for each master node, the total number of replication slots required is equal to the total number of master nodes in the entire MMR system.

Therefore, for a given database server (that is, a Postgres database cluster containing master node databases), the total number of replication slots required is equal to the total number of master nodes in the entire MMR system multiplied by the number of master node databases residing within the given database cluster.

For example, assume the usage of a 6-node multi-master replication system using three database clusters as follows:

- Database cluster #1 contains 3 master node databases.
- Database cluster #2 contains 2 master node databases.
- Database cluster #3 contains 1 master node database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The total number of master nodes is six. Multiply the number of master node databases in each database cluster by six to give the required minimum setting for `max_replication_slots` for that database cluster.

The following table shows the required, minimum settings for `max_replication_slots` as well as `max_wal_senders`.

Table 2-1 – Replication Origin Configuration Parameter Settings

Postgres Database Server	max_wal_senders	max_replication_slots
Cluster #1 (3 master nodes)	3	18
Cluster #2 (2 master nodes)	2	12
Cluster #3 (1 master node)	1	6

If the `max_replication_slots` parameter is not set to a high enough value, synchronization replication still succeeds, but without the replication origin performance advantage.

The publication server log file contains the following warning in such cases:

```
WARNING: Failed to setup replication origin xdb_mmrnode_c_emp_pub_6. Reason:
ERROR: could not find free replication state slot for replication origin with
OID 4
  Hint: Increase max_replication_slots and try again.
```

The following example shows some of the replication slot information for a 3-master node system running on a single database cluster.

The following shows the maximum allowable number of replication slots:

```
SHOW max_replication_slots;

max_replication_slots
----------------------
 9
(1 row)
```

The number should be sufficiently greater than the number of replication slots and replication origins currently allocated.

The following displays the replication slots:

```
SELECT slot_name, slot_type, database, active FROM pg_replication_slots ORDER
BY 1;

  slot_name  | slot_type | database  | active
-------------+-----------+-----------+--------
 xdb_47877_5 | logical   | mmrnode_a | t
 xdb_47878_5 | logical   | mmrnode_b | t
 xdb_47879_5 | logical   | mmrnode_c | t
(3 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows the replication origins.

```
SELECT * FROM pg_replication_origin ORDER BY 2;

 roident |          roname
---------+-------------------------
       5 | xdb_mmrnode_a_emp_pub_39
       2 | xdb_mmrnode_a_emp_pub_6
       1 | xdb_mmrnode_b_emp_pub_1
       6 | xdb_mmrnode_b_emp_pub_39
       3 | xdb_mmrnode_c_emp_pub_1
       4 | xdb_mmrnode_c_emp_pub_6
(6 rows)
```

The replication origin name is assigned in the format `xdb_srcdbname_pubname_remotedbid` where *srcdbname* is the source database name, *pubname* is the publication name, and *remotedbid* is the publication database ID of a remote database.

The replication slots are in the active state when the publication server is running. The replication slots are deactivated when the publication server is shut down.

The replication slots and replication origin sessions are deleted from the database cluster when their corresponding master nodes are removed from the multi-master replication system using the xDB Replication Console or the xDB Replication Server CLI.

Should some situation occur where the replication slots are not properly deleted when required, see Section 10.3.4.4 for instructions on manually deleting them.

2.2.10.5 In-Memory Caching and Persistence

The data changes are fetched and stored in memory buffers to optimize the data replication process. This avoids the overhead associated with repeatedly fetching the same set of changes from the database server when there are multiple target databases.

This approach is sufficient as long as all of the target databases are accessible during a replication event and the data fits within the available cache.

However, if one or more of the target databases is unavailable due to network connectivity problems, server down time, etc. the in-memory data changes must be persisted for later retrieval when the target databases becomes available for synchronization with the source database.

The xDB Replication Server architecture utilizes Java *object serialization* to persist the in-memory state of the data. Object serialization is the conversion of object data and other relevant information to a sequence of bytes that can then be stored in a file.

The following are examples that can result in the eviction of in-memory data to persistent storage:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Before the next replication event occurs, the in-memory cache is filled with the data changes and needs to be evicted to accommodate a new set of changes.
- In the replication system, there are multiple target databases. During a synchronization event, all of the changes available in the cache are applied successfully to some of the target databases. However one or more of the other target databases cannot be accessed. All of the applied changes held in memory must be persisted and retained so that these changes can be reloaded and applied when the inaccessible databases becomes available.

The cache size corresponds to the heap size configured for the publication server by the `-Xmx`*`nnnm`* setting of the `JAVA_HEAP_SIZE` parameter in the xDB Startup Configuration file. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

The persistence I/O overhead can be minimized by increasing the heap size value and defining a more frequent synchronization interval such as for every few seconds. See Section 7.2 for information on setting a replication schedule.

The data changes are persisted in a local file on the host running the publication server. The file is stored in the directory *`XDB_HOME`*`/xdata`.

Each time persistence occurs, a new file is created. After the files have been processed, they are periodically removed from disk.

2.2.11 Multi-Master Parallel Replication

For a multi-master replication system, transactions can be replicated from one master node to another by one of the synchronization methods described in the previous sections – either the trigger-based method (see Section 2.2.9) or the log-based method (see Section 2.2.10).

For a single replication event to be considered finished and complete, transactions that have occurred on all master nodes since the previous replication event must be successfully replicated to all other master nodes by the configured synchronization method.

This consists of a series of multiple replication sets, each identified by a master node acting as the source master node, which contains the transactions that needs to be replicated to all other master nodes acting as the target master nodes. So for a multi-master replication system consisting of *`n`* number of master nodes, there will be *`n`* such replication sets – each with a different master node acting as the source.

Since the initial support of multi-master replication systems in xDB Replication Server version 5.0, such a series of multiple replication sets were always initiated in a strictly serial manner. That is, the transaction replication from a source master node to all target master nodes must be completed before the start of the transaction replication from the next master node to all other target master nodes, and so on.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For example, consider a 3-master node system consisting of master node A, master node B, and master node C.

If applications have applied transactions to tables in all three master nodes and then a synchronization replication event is initiated either on demand by the xDB Replication Console, an xDB Replication Server CLI command, or by a scheduled replication, the transactions are replicated in the following manner:

1. Transactions that were made on master node A are replicated to master node B and master node C.

2. When Step 1 has been completed, transactions that were made on master node B are replicated to master node A and master node C.

3. When Step 2 has been completed, transactions that were made on master node C are replicated to master node A and master node B.

The time to complete the entire replication event, referred to as the *latency* time, is basically the sum of the replication times where each master node acts as the source (that is, the sum of the times for steps 1, 2, and 3).

For the log-based method, this latency time has been reduced by the implementation of *parallel replication* whereby each replication set from a given master node acting as the source, executes and runs simultaneously with all other replication sets where the other master nodes act as the source.

Thus, a replication set from a master node is not waiting for others to complete before it can start so steps 1, 2, and 3 all run simultaneously instead of one after the other.

Note that parallel replication applies only to the log-based method and not for the trigger-based method.

There is no required configuration setting to enable the use of parallel replication for the log-based MMR system.

Note: In addition to parallel replication, optimization of replicating from a given master node to all other master nodes (that is, within the context of a single replication set) has been implemented with the use of multiple threads. This is referred to as *parallel synchronization*. Parallel synchronization applies to both the trigger-based and log-based methods. See Section 5.8.2.2 for information on parallel synchronization.

2.2.12 Table Filters

Table filters specify the selection criteria for rows in publication tables or views that are to be included during replications to subscriptions from the publication database in a single-master replication system or between master nodes in a multi-master replication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

system. Rows that do not satisfy the selection criteria are excluded from replications to subscriptions or master nodes on which these table filters have been enabled.

2.2.12.1 Implementing Table Filters

Implementing table filters is a two-part process. First, a set of available table filters must be defined. This can be done during the process of creating the publication by defining specific, named rules applicable to selected publication tables or views expressed in the form of SQL `WHERE` clauses.

Once a set of available table filters have been defined, they must be enabled only on those subscription tables of a single-master replication system or master node tables of a multi-master replication system where filtering is to occur during replication to those particular target tables. No filtering occurs during replication to a target subscription table or master node table if no filters have been specifically enabled on that table in the subscription or master node.

It is strongly recommended that a snapshot replication be performed to the subscriptions or master nodes that contain tables on which the filtering criteria has changed either by the addition of filter rules, the removal of filter rules, or the modification of existing filter rules.

A snapshot ensures that the content of the subscription tables or master node tables is consistent with the updated filtering criteria.

Note (For MMR only): When using table filters in a multi-master replication system, the master definition node, which provides the source of the table content for a snapshot, should contain a superset of all the data contained in the other master nodes of the multi-master replication system. This ensures that the target of a snapshot receives all of the data that satisfies any filtering criteria enabled on the other master nodes.

On the contrary, if the master definition node contains only a subset of all the data contained in the other master nodes, then a snapshot to another master node may not result in the complete set of data that is required for that target master node.

2.2.12.2 Effects of Table Filtering

A filter enabled on a table only affects the results from snapshot or synchronization replications targeted to that table by the xDB Replication Server. Filtering has no effect on changes made directly on the target table by external user applications such as an SQL command line utility.

Filtering has the following effects on a targeted, filtered table.

Note: In the following discussion, a *result set* refers to the set of rows in a table satisfying the selection criteria of an `UPDATE` or `DELETE` statement executed on that table.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In a snapshot replication, a row from the source table of the snapshot is inserted into the target table if the row satisfies the filtering criteria. Otherwise the row is excluded from insertion into the target table.

When an `INSERT` statement is executed on a source table followed by a synchronization replication, the row is inserted into the target table of the synchronization if the row satisfies the filtering criteria. Otherwise the row is excluded from insertion into the target table.

When an `UPDATE` statement is executed on a source table followed by a synchronization replication, the `UPDATE` result set of the source table determines the action on the target table of the synchronization as follows.

- If a row in the result set has no corresponding row in the target table with the same primary key value, and the updated row in the result set satisfies the filtering criteria, then the row is inserted into the target table. (That is, a row that was previously non-existent in the target table is added because the updated row in the source table now satisfies the filtering criteria.)
- If a row in the result set has a corresponding row in the target table with the same primary key value, and the updated row in the result set satisfies the filtering criteria, then the row in the target table is updated accordingly. (That is, the update is applied to an existing, matching row in the target table that still satisfies the filtering criteria after the update.)
- If a row in the result set has a corresponding row in the target table with the same primary key value, and the updated row in the result set no longer satisfies the filtering criteria, then the corresponding row in the target table is deleted. (That is, an existing, matching row in the target table no longer satisfies the filtering criteria after the update, so the row is removed from the target table.)

When a `DELETE` statement is executed on a source table followed by a synchronization replication, the `DELETE` result set of the source table determines the action on the target table of the synchronization as follows.

- If a row in the result set has a corresponding row in the target table with the same primary key value, then the row with that primary key value is deleted from the target table. (That is, an existing, matching row in the target table is removed.)
- If a row in the result set has no corresponding row in the target table with the same primary key value, then no action is taken on the target table for that row. (That is, there is no existing, matching row in the target table, so there is no row to remove from the target table.)

Thus, regardless of whether the transaction on the source table is an `INSERT`, `UPDATE`, or `DELETE` statement, the goal of a table filter is to ensure that all rows in the target table satisfy the filter rule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.2.12.3 Table Settings and Restrictions for Table Filters

This section lists specific table settings and restrictions on the use of table filters.

REPLICA IDENTITY Setting for Filtering in a Log-Based Replication System

For replication systems using the log-based method of synchronization replication, a publication table on which a filter is to be defined must have the `REPLICA IDENTITY` option set to `FULL`.

Note: This `REPLICA IDENTITY FULL` setting is not required for tables in single-master, snapshot-only publications, See Section 2.2.7 for information on snapshot-only publications.

This setting is done with the `ALTER TABLE` command as shown by the following:

```
ALTER TABLE schema.table_name REPLICA IDENTITY FULL
```

For additional information see the `ALTER TABLE` SQL command in the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/sql-altertable.html

For example, for a publication table named `edb.dept`, use the following `ALTER TABLE` command:

```
ALTER TABLE edb.dept REPLICA IDENTITY FULL;
```

The `REPLICA IDENTITY` setting can be displayed by the PSQL utility using the `\d+` command:

```
edb=# \d+ edb.dept
                                  Table "edb.dept"
 Column |         Type          | Modifiers | Storage  | Stats target | Description
--------+-----------------------+-----------+----------+--------------+-------------
 deptno | numeric(2,0)          | not null  | main     |              |
 dname  | character varying(14) |           | extended |              |
 loc    | character varying(13) |           | extended |              |
Indexes:
    "dept_pk" PRIMARY KEY, btree (deptno)
    "dept_dname_uq" UNIQUE CONSTRAINT, btree (dname)
Referenced by:
    TABLE "emp" CONSTRAINT "emp_ref_dept_fk" FOREIGN KEY (deptno) REFERENCES
dept(deptno)
    TABLE "jobhist" CONSTRAINT "jobhist_ref_dept_fk" FOREIGN KEY (deptno) REFERENCES
dept(deptno) ON DELETE SET NULL
Replica Identity: FULL
```

The `REPLICA IDENTITY FULL` setting is required on tables in the following databases of a log-based replication system:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- In a single-master replication system, table filters are defined in the master database. Thus, the publication tables in the master database requiring filter definitions must be altered to a `REPLICA IDENTITY FULL` setting, but only if the publication is not a snapshot-only publication. See Section 2.2.7 for information on snapshot-only publications.
- In a multi-master replication system, table filters are defined in the master definition node. Thus, publication tables in the master definition node requiring filter definitions must be altered to a `REPLICA IDENTITY FULL` setting.
- In a multi-master replication system, non-MDN nodes should not have their tables' `REPLICA IDENTITY` option set to `FULL` unless transactions are expected to be targeted on those non-MDN nodes, and the transactions are to be filtered when they are replicated to the other master nodes.

The `REPLICA IDENTITY FULL` setting on a source table ensures that certain types of transactions on the source table result in the proper updates to the target tables on which filters have been enabled.

Note: In addition to table filtering requirements, the `REPLICA IDENTITY FULL` setting may be required on publication tables for other reasons in xDB Replication Server. See Section 6.6.1 for additional requirements.

Filtering Restrictions on Data Types

Table filters are not supported on binary data type columns. A binary data type is the Postgres data type `BYTEA`. In addition, table filters are not supported on Advanced Server columns with data types `BINARY`, `VARBINARY`, `BLOB`, `LONG RAW`, and `RAW` as these are alias names for the `BYTEA` data type.

2.2.12.4 Roadmap for Further Instructions

The specific details on implementing table filtering depend upon whether you are using a single-master replication system or a multi-master replication system. The following is a roadmap to the relevant sections for each type of replication system.

For using table filters in a single-master replication system see the following sections:

- Section 5.2.3 for information on defining the initial set of table filters that are to be available for selective enablement on subscriptions
- Section 5.3.3 for information on enabling available table filters on a newly created subscription
- Section 7.6.4 for information on adding, removing, or modifying rules comprising the set of available table filters
- Section 5.5.4 for information on changing which table filters have been enabled on an existing subscription

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For using table filters in a multi-master replication system see the following sections:

- Section 6.2.3 for information on defining the initial set of table filters that are to be available for selective enablement on master nodes
- Section 6.3 for information on enabling available table filters on a newly created master node
- Section 7.6.4 for information on adding, removing, or modifying rules comprising the set of available table filters
- Section 6.9 for information on changing which table filters have been enabled on an existing master node

2.3 xDB Replication Server Components and Architecture

This section describes the components and architecture of xDB Replication Server. Section 2.3.1 describes the executable programs, files, and databases that comprise xDB Replication Server. Section 2.3.2 defines the logical components of a replication system and how they correspond to the programs and databases. Section 2.3.3 illustrates some examples of replication systems.

2.3.1 Physical Components

xDB Replication Server is not a single, executable program, but rather a set of programs along with data stores containing configuration information and metadata that work together to form a replication system.

The following diagram illustrates the components of xDB Replication Server and how they are used to form a complete, basic, single-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

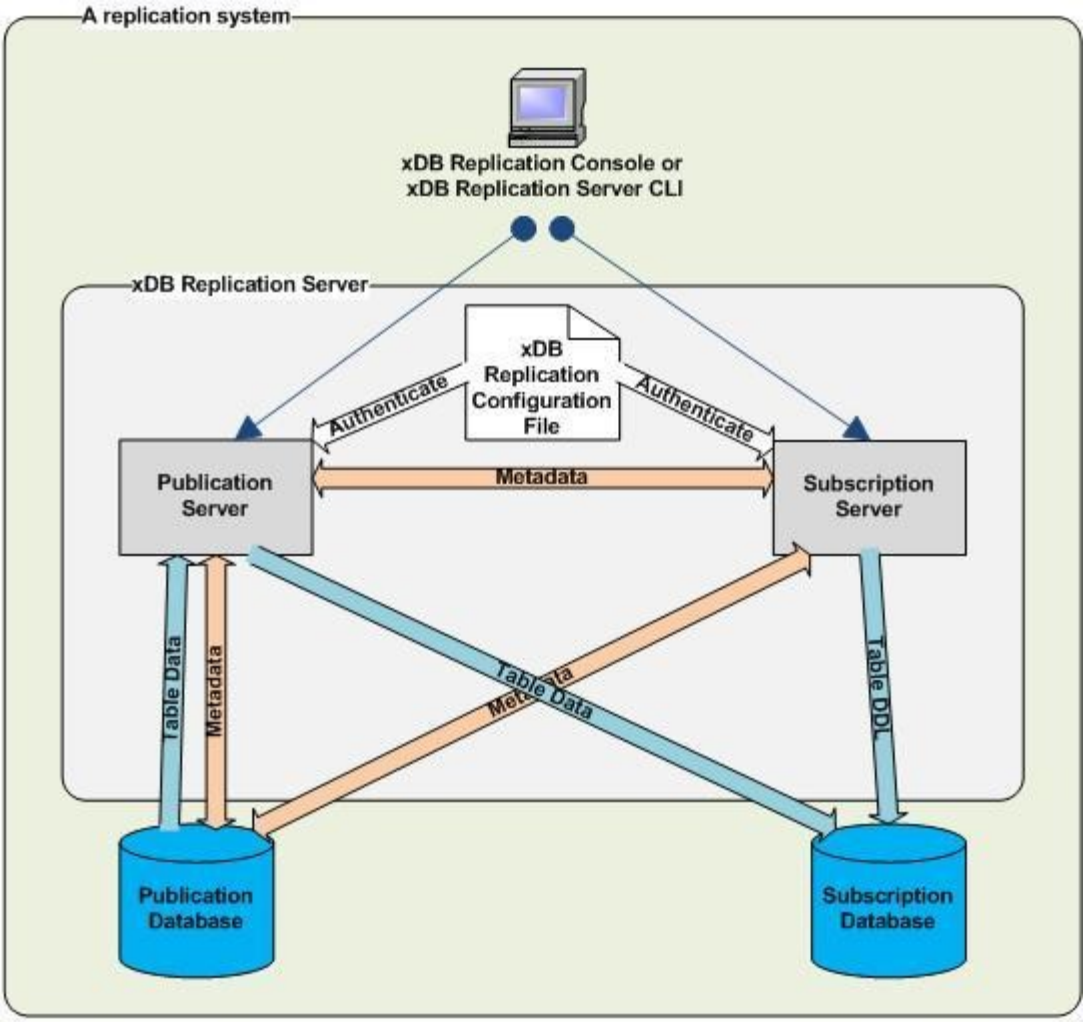


Figure 2-8 - xDB Replication Server - physical view (single-master replication system)

The following diagram illustrates the components of xDB Replication Server and how they are used to form a complete, basic, multi-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

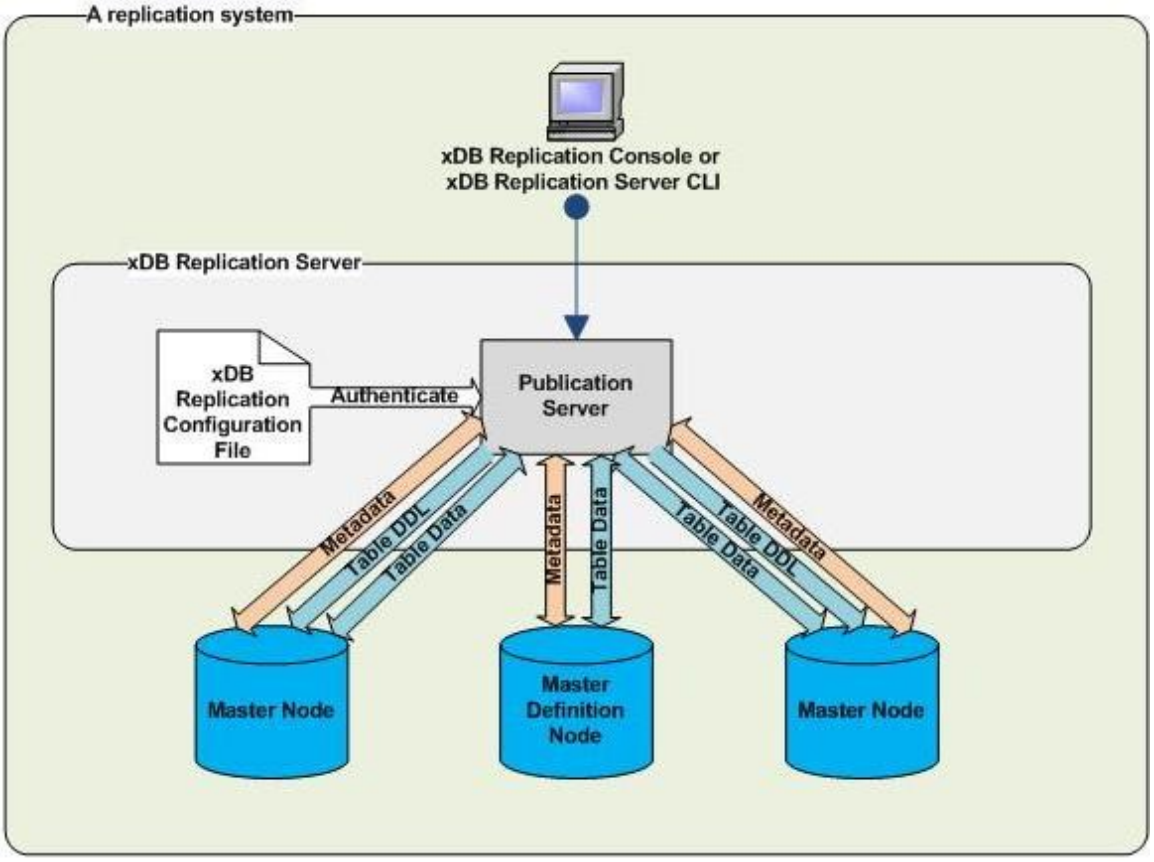


Figure 2-9 - xDB Replication Server - physical view (multi-master replication system)

The minimal configuration of xDB Replication Server for a basic replication system consists of the following software components:

- **Publication server.** The program that configures the publication database and master nodes for replication and performs replication.
- **Subscription server.** The program that configures the subscription database for replication and initiates replication. The subscription server is used only in single-master replication systems.
- **xDB Replication Configuration file.** Text file containing connection and authentication information used by the publication server and subscription server upon startup to connect to a publication database designated as the controller database. Also used to authenticate registration of the publication server and subscription server from the user interface when creating a replication system.
- **xDB Startup Configuration file.** Text file containing installation and configuration information used for the Java Runtime Environment when the publication server and subscription server are started.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The entire replication system is completed with the addition of the following components:

- User interfaces for configuring and maintaining the replication system
- One or more publication databases for a single-master replication system
- One or more subscription databases for a single-master replication system
- One master definition node for a multi-master replication system
- One or more additional master nodes for a multi-master replication system

The user interface, publication server, subscription server, publication database, subscription database, and master nodes can all run on the same host or on separate, networked hosts.

Any number of user interfaces can be used at any time to access any number of publication servers and subscription servers on the network as long as the network locations, user names, and passwords of the publication and subscription servers are known.

Any number of publication and subscription databases can participate in a single-master replication system.

Any number of master nodes can participate in a multi-master replication system.

The following sections describe each component in more detail.

2.3.1.1 Publication Server

The publication server creates and manages the metadata for publications. When a publication is created, the publication server creates database objects in the control schema of the publication database to record metadata about the publication.

Whenever a master node is added to a multi-master replication system, the publication server creates database objects in the control schema of the master node for recording metadata. For non-MDN nodes, the publication server also calls EnterpriseDB's Migration Toolkit to create the publication table definitions if so chosen at master node creation time.

Note: See Section 2.3.1.11 for information on the control schema.

The publication server is also responsible for performing a replication. For snapshot replications, the publication server calls EnterpriseDB's Migration Toolkit to perform the snapshot.

For single-master synchronization replications, the publication server uses the Java Database Connectivity (JDBC) interface to apply changes to the subscription table rows based on changes that have been recorded in either one of two ways: a) If the publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

database is running under Postgres version 9.4 or later and the logical decoding option has been chosen when creating the publication, changes are obtained from the Postgres WAL files using a logical replication slot, or b) In all other circumstances, changes are recorded in metadata tables (called shadow tables) in the publication database by row-based triggers that fire upon any insert, update, or deletion to the publication table rows.

For multi-master synchronization replications, the publication server performs the same process as for single-master synchronizations, but does so for each master node pair combination in the multi-master replication system.

The publication server may run on the same host as the other xDB Replication Server components, or it may run on a separate, networked host.

2.3.1.2 Subscription Server

Note: The subscription server is required only for single-master replication systems. The subscription server does not need to be running, nor even installed if only multi-master replication systems are in use.

The subscription server creates and manages the metadata for subscriptions. When a subscription is created, the subscription server creates database objects in the control schema of the publication database to record metadata about the subscription.

When a subscription is created, the subscription server calls EnterpriseDB's Migration Toolkit to create the subscription table definitions in the subscription database. The rows in the subscription tables are not populated until a replication occurs. Rows are populated by actions of the publication server.

The subscription server is also responsible for initiating a replication as a result of manual user action through the user interface, or a schedule created for the subscription. The subscription server initiates a call to the publication server that manages the associated publication. The publication server then performs the actual replication.

The subscription server may run on the same host as the other xDB Replication Server components, or it may run on a separate, networked host.

When the subscription server is started, it uses the information in the xDB Replication Configuration file found on its host to connect to the designated controller database.

2.3.1.3 xDB Replication Configuration File

The xDB Replication Configuration file contains the connection and authentication information used by any publication server or subscription server running on the host containing the file.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Specifically, the xDB Replication Configuration file is accessed in the following circumstances:

- When a publication server or subscription server is started on the host.
- When a publication server or subscription server is registered during the process of creating a replication system. Registration of a publication server or subscription server is done using the xDB Replication Console or the xDB Replication Server Command Line Interface.

The following table contains a brief description of the parameters in the xDB Replication Configuration file.

Table 2-2 - xDB Replication Configuration File

Parameter	Description
admin_user	xDB administrator user name (the admin user name) for registering a publication server or a subscription server on this host containing the xDB Replication Configuration file
admin_password	Encrypted password of the admin user
database	Database name of the controller database
user	Database user name of the controller database
password	Encrypted password of the controller database user
port	Port number on which the database server of the controller database listens for requests
host	IP address of the host running the database server of the controller database
type	Database type of the controller database such as oracle, enterprisedb, etc.

The xDB Replication Server product creates the content of this file as follows:

- The xDB Replication Configuration file and some of its initial content are created when you install a publication server or subscription server on a host during the xDB Replication Server installation process.
- Parameters admin_user and admin_password are determined during the xDB Replication Server installation process. See Chapter 3 for how the content of these parameters are determined.
- Parameters database, user, password, port, host, and type are set with the connection and authentication information of the first publication database definition you create with the xDB Replication Console or xDB Replication Server CLI. This database is designated as the controller database. See Section 2.3.1.12 for information on the controller database. See Section 5.2.2 for creating a publication database definition for a single-master replication system. See Section 6.2.2 for creating the publication database definition for a multi-master replication system.

The following is an example of the content of an xDB Replication Configuration file:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
#xDB Replication Server Configuration Properties
#Tue May 26 13:45:37 GMT-05:00 2015
port=1521
admin_password=ygJ9AxoJEX854elcVIJPTw\=\=
user=pubuser
admin_user=admin
type=oracle
password=ygJ9AxoJEX854elcVIJPTw\=\=
database=xe
host=192.168.2.23
```

Note: The passwords for the admin user name and the controller database user name are encrypted. Should you change either of these passwords, you must modify the corresponding password parameters in the xDB Replication Configuration file to contain the encrypted form of the new password. See Section 10.4.2 for directions on how to generate the encrypted form of a password.

See Section 3.5 for the file system location of the xDB Replication Configuration file.

2.3.1.4 xDB Startup Configuration File

The xDB Startup Configuration file contains installation and configuration information primarily used by the Java Runtime Environment (JRE) when any publication server or subscription server is started up on the host containing the file.

The content of the file is created by the xDB Replication Server installer when you install xDB Replication Server.

The following is an example of the content of an xDB Startup Configuration file:

```
#!/bin/sh

JAVA_EXECUTABLE_PATH="/usr/bin/java"
JAVA_MINIMUM_VERSION=1.7
JAVA_BITNESS_REQUIRED=64
JAVA_HEAP_SIZE="-Xms256m -Xmx1536m"
PUBPORT=9051
SUBPORT=9052
```

The following table contains a brief description of the parameters in the xDB Startup Configuration file.

Table 2-3 - xDB Startup Configuration File

Parameter	Description
JAVA_EXECUTABLE_PATH	Directory path to the Java runtime program used to start and run the publication and subscription servers
JAVA_MINIMUM_VERSION	The earliest JRE version that can be used by the publication and subscription servers
JAVA_BITNESS_REQUIRED	The bitness of the Java virtual machine required by the installed publication and subscription servers
JAVA_HEAP_SIZE	In -Xms*nnn*m *nnn* specifies the minimum Java heap size in

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Parameter	Description
	megabytes. In -Xmx*nnn*m *nnn* specifies the maximum Java heap size in megabytes
PUBPORT	Port number on which the publication server listens for requests
SUBPORT	Port number on which the subscription server listens for requests

The JAVA_EXECUTABLE_PATH parameter specifies the location of the Java runtime program as identified by the xDB Replication Server installer during the installation process. The setting of this parameter may be subsequently changed to a different JRE installation if so desired.

The JAVA_MINIMUM_VERSION parameter specifies the earliest version of the Java Runtime Environment that can be used with xDB Replication Server. This setting must not be changed.

The JAVA_BITNESS_REQUIRED parameter must not be altered. If the installed value is modified, or if it does not match the bitness of the Java virtual machine as identified by JAVA_EXECUTABLE_PATH, a number of errors may occur, which include failure of the publication and subscription servers to start and registration failure of the xDB Replication Server product.

See Section 5.1.1 for information on setting the JAVA_HEAP_SIZE parameter.

See Section 5.1.6.1 for information on the PUBPORT and SUBPORT parameters.

After making any modifications to the xDB Startup Configuration file, the publication server and subscription server must be restarted.

See Section 3.5 for the file system location of the xDB Startup Configuration file.

2.3.1.5 xDB Replication Console

The xDB Replication Console is the graphical user interface program you can use to create and control all aspects of a replication system.

Through a single xDB Replication Console, you can configure and operate a replication system running on the same host on which the xDB Replication Console is installed, or you can configure and operate replication systems where the xDB Replication Server components are distributed on different hosts in a networked environment.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

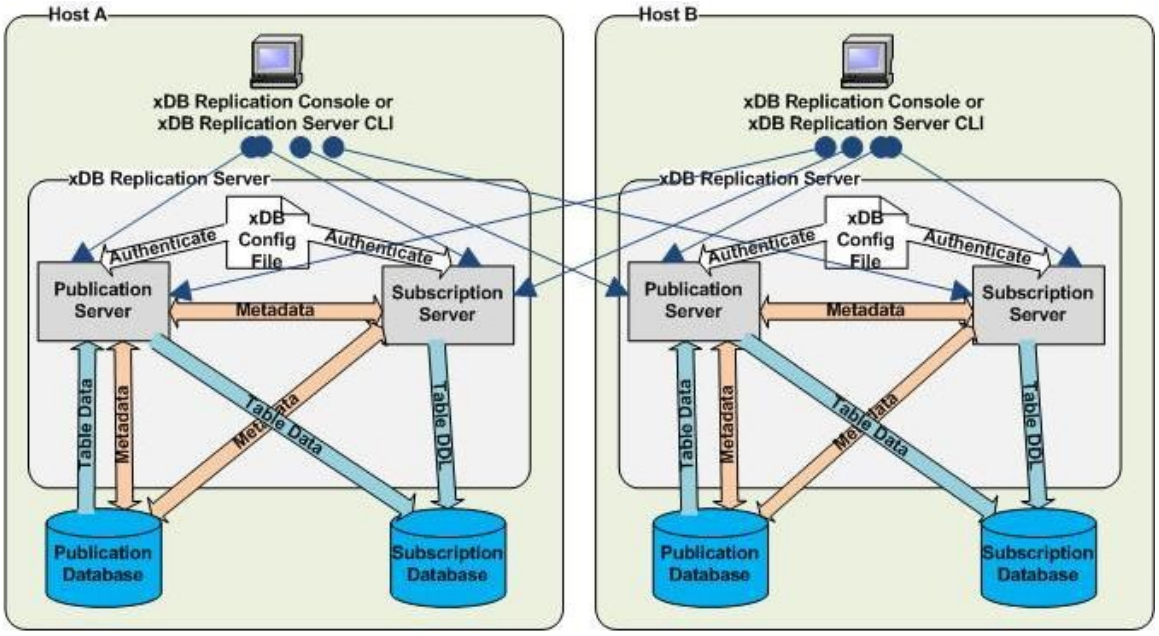


Figure 2-10 - xDB Replication Consoles accessing multiple hosts

In the preceding figure, there are two Postgres installations running on two networked hosts, each with its own xDB Replication Server installation. Each host is running a publication server and a subscription server.

The xDB Replication Console on each host can access and manage the replication systems on the other host if given the network IP address, port number, user name, and password with which the publication server and subscription server were installed with on the remote host.

See Chapter 4 for information on the user interface of the xDB Replication Console.

2.3.1.6 xDB Replication Server Command Line Interface

xDB Replication Server Command Line Interface (CLI) is a command line driven alternative to the xDB Replication Console graphical user interface, providing equivalent functionality for creating and controlling all aspects of a replication system.

Automation of replication system operations can be done by embedding xDB Replication Server CLI commands in scripts such as Bash for Linux.

xDB Replication Server CLI is installed whenever you choose to install the xDB Replication Console.

Chapter 8 provides directions for using xDB Replication Server CLI.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.3.1.7 Publication Database

The publication database contains the tables and views used in a publication. The publication database may be running on the same host or on a different host than where the publication server is running as long as the hosts are accessible to each other by a network.

Each publication database also contains a control schema, which is a collection of database objects containing metadata on all replication systems, both single-master and multi-master, controlled by the publication server connected to this publication database. See Section 2.3.1.11 for information on the control schema.

In a multi-master replication system, all master nodes are considered publication databases.

A database plays the roles of both a publication database and a subscription database if it contains publications and subscriptions.

2.3.1.8 Subscription Database

Note: The subscription database applies only to single-master replication systems.

The subscription database contains the tables created from a subscription. The subscription database may be running on the same host or on a different host than where the subscription server is running as long as the hosts are accessible to each other by a network.

A subscription database can also serve as a publication source for replicating to a third server if desired. This configuration is referred to as *cascading replication*.

A database plays the roles of both a publication database and a subscription database if it contains publications and subscriptions such as in the cascaded replication scenario.

2.3.1.9 Master Node

In a multi-master replication system, the databases containing the set of tables (the publication) for which row changes are to be replicated are called master nodes. The master nodes may be running on the same host or on different hosts than where the publication server is running as long as the hosts are accessible to each other by a network.

Each master node also contains a control schema, which is a collection of database objects containing metadata on all replication systems, both single-master and multi-master, controlled by the publication server connected to this master node. See Section 2.3.1.11 for information on the control schema.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The master nodes may be running under the same, or under multiple database server instances (Postgres database clusters).

2.3.1.10 Master Definition Node

The first node added to create a multi-master replication system is initially designated the master definition node. This node must contain the table definitions (and optionally, the initial set of rows) that are to be included in the publication.

As subsequent databases are added as master nodes to the replication system, the table definitions and initial row sets can optionally be propagated from the master definition node to the newly added master nodes.

After the multi-master replication system is defined, it is possible to reassign the role of the master definition node to another master node in the multi-master replication system. The significance of this reassignment is that snapshots can be taken from the newly appointed master definition node to other master nodes. This could be beneficial if the data in the old master definition node becomes corrupt or out-of-sync with the other master nodes and needs to be completely refreshed by a snapshot from another master node.

As with all master nodes, the master definition node contains a control schema, which is a collection of database objects containing metadata on all replication systems, both single-master and multi-master, controlled by the publication server connected to this master node. See Section 2.3.1.11 for information on the control schema.

2.3.1.11 Control Schema and Control Schema Objects

The *control schema* is a conceptual term referring to the collection of metadata database objects that define the logical and physical structure of, and enable the operation and maintenance of xDB Replication Server single-master and multi-master replication systems.

These metadata database objects, referred to as *control schema objects* consist of tables, sequences, functions, procedures, triggers, packages, etc.

The control schema objects store metadata such as type of replication system (single-master or multi-master), network location, database type, connection and authentication information for publication databases, subscription databases, and master nodes, names of publications and the tables and views they contain, names of subscriptions and the publications to which they are subscribed, replication transaction status, replication scheduling, replication history cleanup scheduling, replication history, etc.

Each publication database in a trigger-based, single-master replication system also contains control schema objects with the changes that have been made to rows in the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

publication and the status of whether or not those changes have been applied to the subscription tables.

Similarly, for a multi-master replication system, each trigger-based master node contains control schema objects with the changes that have been made to rows in the publication residing on that master node, and the statuses of whether or not those changes have been applied to the other master nodes in the multi-master replication system.

Note: For log-based single-master and multi-master replication systems, changes are extracted from the database server WAL files instead of being stored in control schema objects. See Section 2.2.10 for information on the log-based method.

The actual, physical database schemas implementing the control schema to which the control schema objects belong varies depending upon the database type (Oracle, SQL Server, or Postgres) and how the database was initially configured for use by xDB Replication Server.

The following points should be noted about the control schema:

- The control schema and its control schema objects are created in every publication database of both single-master and multi-master replication systems. That is, all master (publication) databases of single-master replication systems and all master nodes of multi-master replication systems.
- When a new master database is added for a single-master replication system or a new master node for a multi-master replication system, a snapshot operation is used to replicate the control schema to the newly added publication database assuming there is an existing controller database. See Section 2.3.1.12 for information regarding the controller database.
- Updates to the configuration of a single-master replication system or a multi-master replication system made by the xDB Replication Console or the xDB Replication Server Command Line Interface are synchronized between the control schemas on all publication databases to ensure that the metadata is consistent across all publication databases.
- The slave (subscription) database of single-master replication systems contains one, single table as its metadata database object. The term, *subscription metadata object*, is specifically used to refer to this database object in the subscription database. The general terms, control schema and control schema objects refer to the database objects in the publication databases.
- The control schema objects in all databases controlled by the same given publication server generally contain the same information. This allows any such database to provide the information needed by xDB Replication Server to control all single-master and multi-master replication systems running under that given publication server.
- Should a certain publication database of a replication system go offline due to database server problems, network connectivity issues, etc., the other replication systems running under the same publication server are still functional since the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

other publication databases can provide the control schema information required to run all replication systems.

2.3.1.12 Controller Database

In the xDB Replication Configuration file, the connection and authentication information for one publication database is included and as such, designated as the *controller database*.

As with all publication databases, the controller database contains the control schema with the replication system information for all single-master and multi-master replication systems run by the publication server that accesses that xDB Replication Configuration file.

The controller database serves as the primary provider of the replication system information to the publication server and the subscription server. Thus, upon initial startup, the publication server and subscription server attempt to connect to the designated controller database. This controller database then provides the metadata information for all replication systems.

Should the initial connection to the controller database fail for some reason, you can manually edit the xDB Replication Configuration file to provide the connection and authentication information for another publication database. Then upon startup of the publication server and subscription server, the control schema of this alternate publication database is used to provide the replication system information.

The initial controller database is determined by the first publication database definition created by the xDB Replication Console or the xDB Replication Server CLI either for a single-master or multi-master replication system. The publication server records the connection and authentication information in the xDB Replication Configuration file.

If you wish to delete the publication database definition of the current controller database, you must first designate another publication database, defined under the same publication server, as the controller using the xDB Replication Console. See Section 7.7 for directions on switching the controller to another publication database.

The following are some points regarding the controller database:

- The database server running the controller database must be running and accessible before starting the publication server and subscription server.
- For a single-master replication system, the publication server under which the publication database and publication are defined and the subscription server under which the subscription database and the subscription related to the publication are

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

defined, must both connect to the same the controller database. This gives the publication server and the subscription server access to the same control schema.

- When changes are made to the metadata maintained by the control schema in the controller database, these changes are replicated by the publication server to the control schemas of all other publication databases. This ensures that the metadata of all single-master and multi-master systems are complete and consistent in the control schemas of all publication databases. This allows you to switch the controller database at some later point in time. See Section 7.7 for information on switching the controller database.

Note: If the controller database is an Oracle or a SQL Server publication database, then a second Oracle or SQL Server publication database cannot be added to create a second single-master replication system. In order for xDB Replication Server to run more than one single-master replication systems consisting of Oracle or SQL Server publication databases, a Postgres publication database must be designated as the controller database.

Once you have multiple Oracle or SQL Server publication databases set up in single-master replication systems with a Postgres controller database, do not switch the controller database to an Oracle or SQL Server publication database.

2.3.2 Logical Components

This section discusses the logical components of a replication system, how they are related to each other, and how they correspond to the programs and databases in a replication system.

The logical components are created when you build a replication system using the xDB Replication Console or the xDB Replication Server CLI. The logical components are stored as part of the replication system metadata in the control schema of the publication databases.

Creating a replication system requires the following steps:

- Register a publication server
- Create a publication database definition
- Create a publication

For a single-master replication system, you then perform the following:

- Register a subscription server
- Create a subscription database definition
- Create a subscription

For a multi-master replication system, you create additional master nodes by creating additional publication database definitions.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Each of these steps creates a logical component that is represented by a node in the replication tree of the xDB Replication Console. See Chapter 4 for a description of the xDB Replication Console. A brief description of these components is given in the following sections.

2.3.2.1 Publication Server

The first step in creating a publication is to identify the publication server that is to be used to manage the publication. This process is called *registering the publication server*.

Using the xDB Replication Console or the xDB Replication Server CLI, a publication server is registered by giving the IP address and port number of the host on which the publication server is running, along with the admin user name and password stored in the xDB Replication Configuration file located on the host running the publication server. (This information is determined during the publication server installation process.)

When viewed in the xDB Replication Console, a registered publication server appears under the top level Replication Servers node in the replication tree. All publication related logical components are created subordinate to a registered publication server and appear underneath it in the replication tree.

Section 5.2.1 gives directions for registering a publication server for a single-master replication system. See Section 6.2.1 for a multi-master replication system.

2.3.2.2 Replication System Type (SMR/MMR)

Subordinate to a registered publication server, two nodes representing the replication system type appear. One is identified by the label `SMR` for single-master replication and the other has the label `MMR` for multi-master replication.

If you are creating a single-master replication system, you proceed to add logical components under the SMR type node.

If you are creating a multi-master replication system, you proceed to add logical components under the MMR type node.

2.3.2.3 Publication Database Definition

Subordinate to one of the Replication System Type nodes under a registered publication server, one or more publication database definitions can be created.

A publication database definition identifies a database whose tables and views are to be used in a publication. The identify information consists of the database server IP address, port number, a database user name and password, and the database identifier.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The publication server uses this information to connect to the publication database in order to create the replication system control schema in the publication database and perform the replications.

Though the process of creating a publication database definition is similar for single-master and multi-master replication systems, their usage within the replication system is somewhat different.

In a single-master replication system, a publication database definition identifies the storage area of one or more publications, each of which is eventually associated with its own subscription in a master-to-slave relationship.

In a multi-master replication system, each publication database definition subordinate to the MMR type node of a given publication server identifies a master node in a single, multi-master replication system.

Note: Currently, there can only be one multi-master replication system per publication server.

Section 5.2.2 discusses creating a publication database definition for a single-master replication system. See sections 6.2.2 and 6.3 for a multi-master replication system.

2.3.2.4 Publication

Subordinate to a publication database definition in a single-master replication system, one or more publications can be defined. A publication contains a list of tables and views that are to be replicated to a subscription database.

In a single-master replication system, the database user name specified in the publication database definition of the publication's parent, as viewed in the replication tree, must have the `SELECT` object privilege on any table or view that is to be included in the publication.

Subordinate to a publication database definition in a multi-master replication system, one and only one publication can be defined. The publication contains the list of tables that are to be replicated and kept synchronized in the master nodes of the multi-master replication system.

In a multi-master replication system, the database user name specified in the publication database definition of the publication's parent, as viewed in the replication tree, must have superuser privileges and be the owner of all tables to be included in the publication.

Section 5.2.3 discusses creating a publication for a single-master replication system. See Section 6.2.3 for a multi-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.3.2.5 Subscription Server

Note: The subscription server applies only to single-master replication systems. You do not register a subscription server when creating a multi-master replication system.

The first step in creating a subscription is to identify the subscription server that is to be used to manage the subscription. This process is called *registering the subscription server*.

Using the xDB Replication Console or the xDB Replication Server CLI, a subscription server is registered by giving the IP address and port number of the host on which the subscription server is running, along with the admin user name and password stored in the xDB Replication Configuration file located on the host running the subscription server. (This information is determined during the subscription server installation process.)

When viewed in the xDB Replication Console, a registered subscription server appears under the top level Replication Servers node in the replication tree. All subscription related logical components are created subordinate to a registered subscription server and appear underneath it in the replication tree.

Section 5.3.1 gives directions for registering a subscription server.

2.3.2.6 Subscription Database Definition

Note: The subscription database definition applies only to single-master replication systems. You do not create a subscription database definition when creating a multi-master replication system.

Subordinate to a registered subscription server, one or more subscription database definitions can be created.

A subscription database definition identifies a database to which a publication's tables and views are to be replicated. The identify information consists of the database server IP address, port number, a database user name and password, and the database identifier.

The subscription server uses this information to connect to the subscription database to create the table definitions.

The publication server also uses this information to connect to the subscription database when it performs replications.

Section 5.3.2 discusses creating a subscription database definition.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.3.2.7 Subscription

Note: The subscription applies only to single-master replication systems. You do not create a subscription when creating a multi-master replication system.

Subordinate to a subscription database definition, one or more subscriptions can be defined. A subscription associates a publication to a subscription database to which the publication's tables and views are to be replicated.

Each subscription can be associated with one and only one publication.

Section 5.3.3 discusses creating a subscription.

2.3.3 xDB Replication System Examples

This section contains examples of replication systems and how the logical components are used to define them.

In the accompanying diagrams, the logical components, represented by nodes in the replication tree of the xDB Replication Console, are superimposed on physical component diagrams. The logical components are shaded in yellow to aid in identifying them in the diagrams.

2.3.3.1 Oracle to PostgreSQL or Advanced Server Replication

The following is an illustration of a basic Oracle to PostgreSQL or Advanced Server single-master replication system. A single publication in Oracle contains tables from two schemas that are replicated to a database residing in either PostgreSQL or Advanced Server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

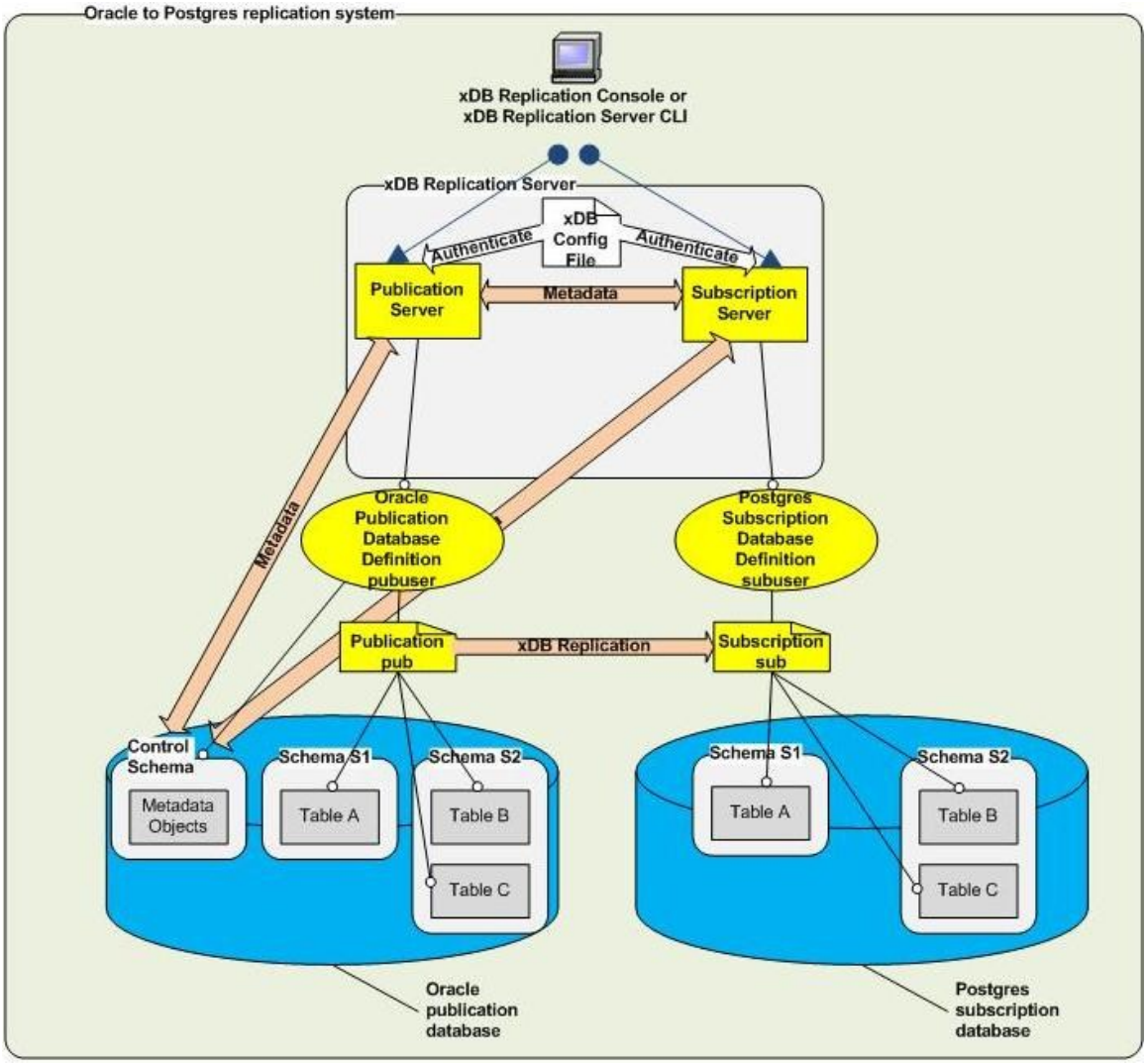


Figure 2-11 - Oracle to PostgreSQL or Advanced Server replication

The following describes the logical components in the preceding diagram:

- The publication server to be used is identified by registering its network location, user name, and password.
- A publication database definition is created subordinate to the SMR type node under the publication server. The Oracle database user name `pubuser` is specified in the definition along with the database network location and database identifier. When you create a user named `pubuser` in Oracle, a schema named `pubuser` is automatically created by Oracle at the same time. The publication server creates the control schema objects in the `pubuser` control schema for the replication system's metadata when you create the publication database definition.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- A publication named `pub` is created subordinate to the publication database definition. The publication consists of table `A` in schema `S1` and tables `B` and `C` in schema `S2`.
- The subscription server to be used is identified by registering its network location, user name, and password.
- A subscription database definition is created subordinate to the subscription server. The Postgres database user name `subuser` is specified in the definition along with the database network location and database identifier.
- A subscription named `sub` is created subordinate to the subscription database definition. When the subscription is created, the subscription server creates schemas named `S1` and `S2` in the subscription database. The table definitions for tables `A`, `B`, and `C` are also created at this time. When replication occurs, the publication server populates these tables with rows from the publication.

The following screen capture shows how the logical components of this replication system appear in the xDB Replication Console replication tree.

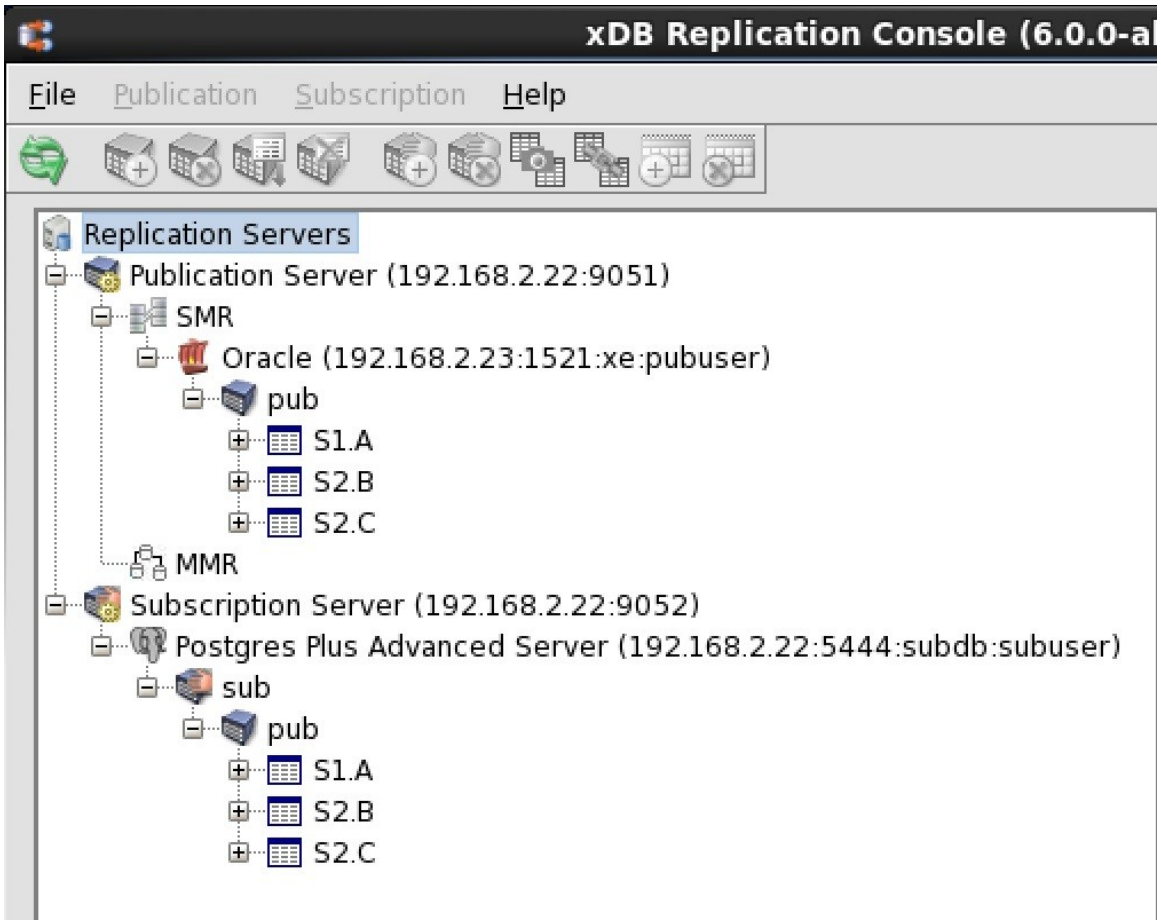


Figure 2-12 - Oracle to Postgres replication tree

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Chapter 4 for an introduction to the xDB Replication Console.

2.3.3.2 SQL Server to PostgreSQL or Advanced Server Replication

The following is an illustration of a basic SQL Server to PostgreSQL or Advanced Server single-master replication system. A single publication in SQL Server contains tables from two schemas that are replicated to a database residing in either PostgreSQL or Advanced Server.

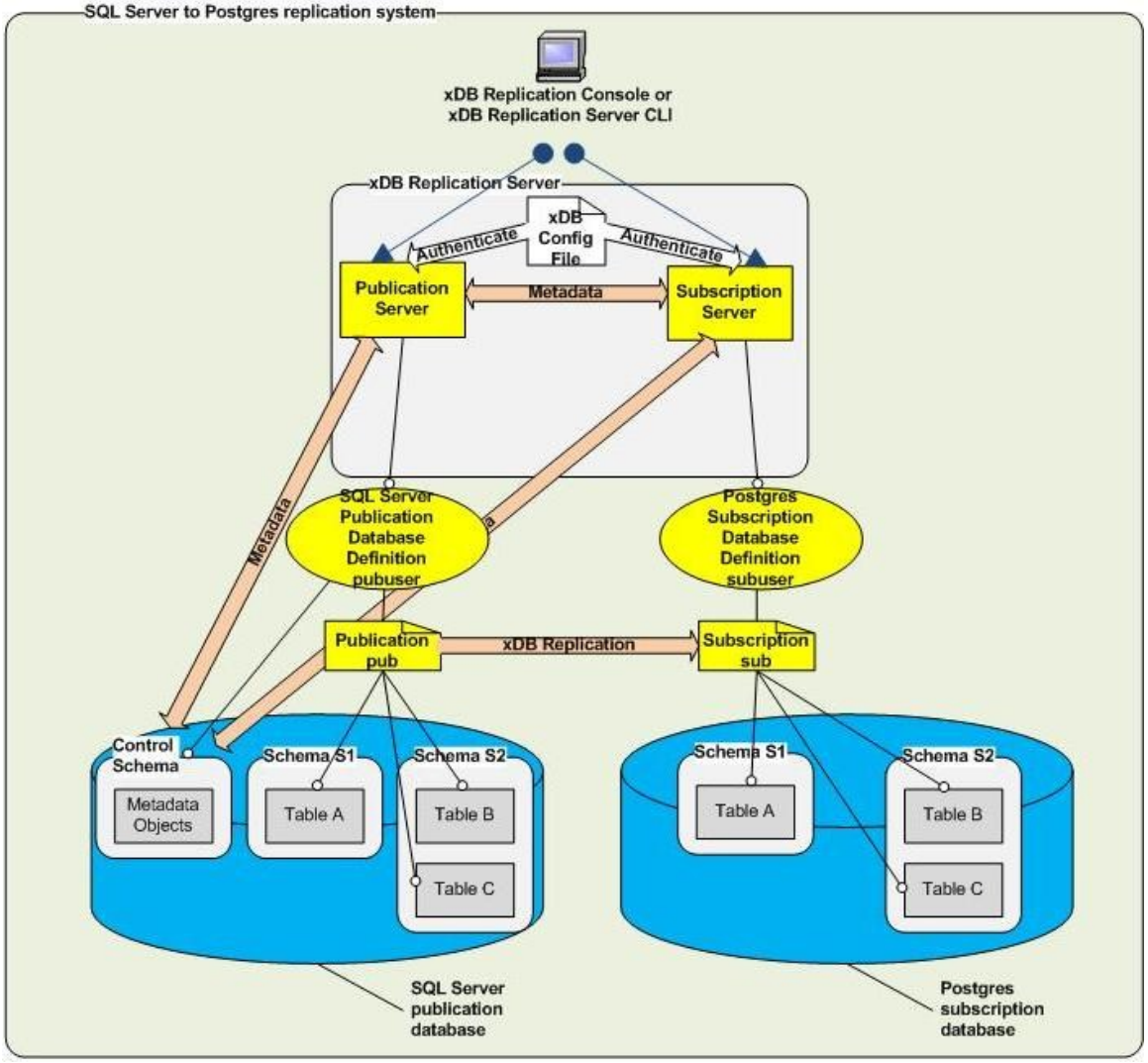


Figure 2-13 - SQL Server to PostgreSQL or Advanced Server replication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following describes the logical components in the preceding diagram:

- The publication server to be used is identified by registering its network location, user name, and password.
- A publication database definition is created subordinate to the SMR type node under the publication server. The SQL Server login `pubuser` is specified in the definition along with the database network location and database identifier. The schema `pubuser` was created during the publication database preparation step as described in Section 5.1.4.2. The `pubuser` schema along with the control schema consisting of three physical schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` are populated with the control schema objects for the replication system's metadata when you create the publication database definition.
- A publication named `pub` is created subordinate to the publication database definition. The publication consists of table `A` in schema `S1` and tables `B` and `C` in schema `S2`.
- The subscription server to be used is identified by registering its network location, user name, and password.
- A subscription database definition is created subordinate to the subscription server. The Postgres database user name `subuser` is specified in the definition along with the database network location and database identifier.
- A subscription named `sub` is created subordinate to the subscription database definition. When the subscription is created, the subscription server creates schemas named `S1` and `S2` in the subscription database. The table definitions for tables `A`, `B`, and `C` are also created at this time. When replication occurs, the publication server populates these tables with rows from the publication.

The following screen capture shows how the logical components of this replication system appear in the xDB Replication Console replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

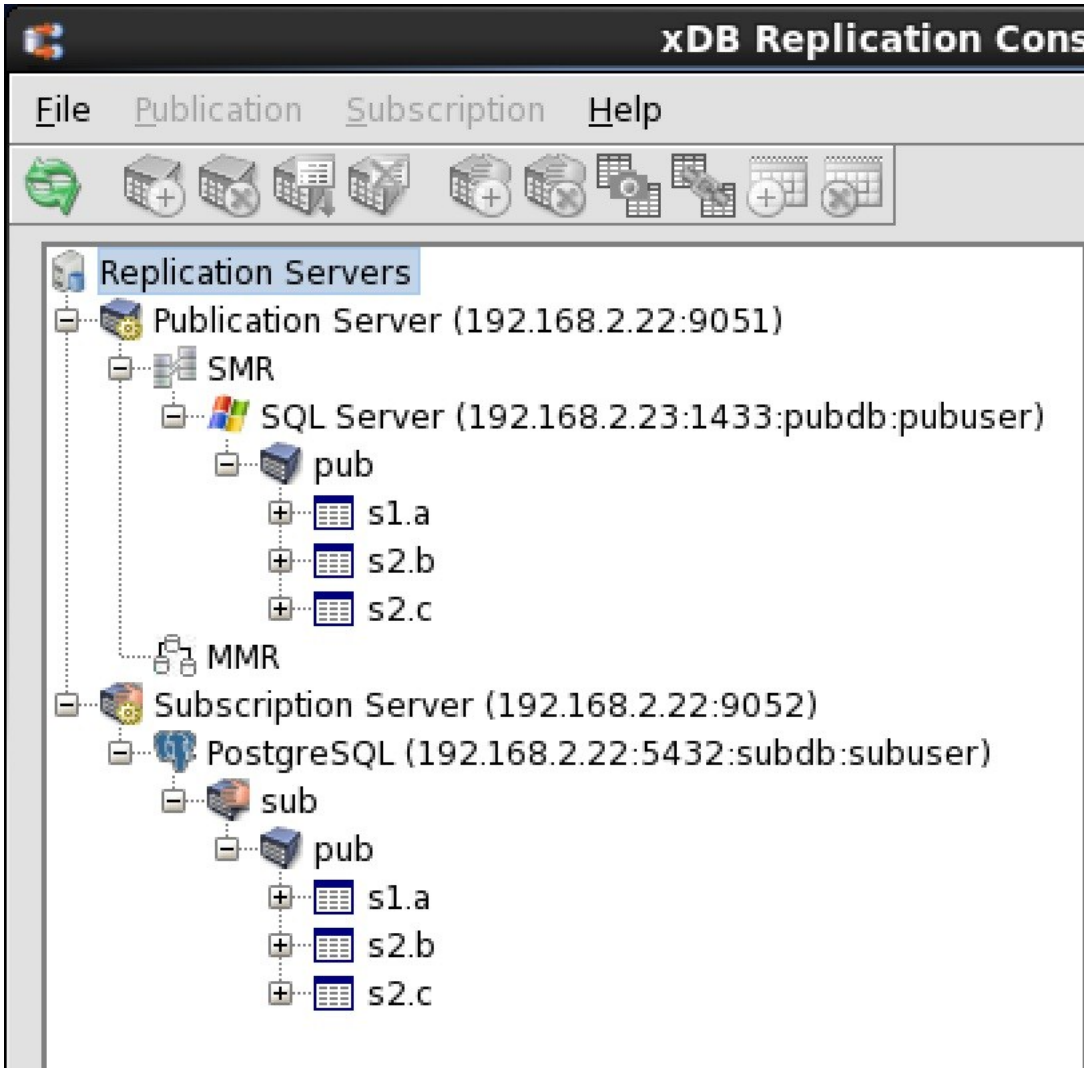


Figure 2-14 - SQL Server to Postgres replication tree

See Chapter 4 for an introduction to the xDB Replication Console.

2.3.3.3 Advanced Server to Oracle Replication

The following is an illustration of a basic Advanced Server to Oracle single-master replication system. A single publication in an Advanced Server database contains tables from two schemas that are replicated to an Oracle database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

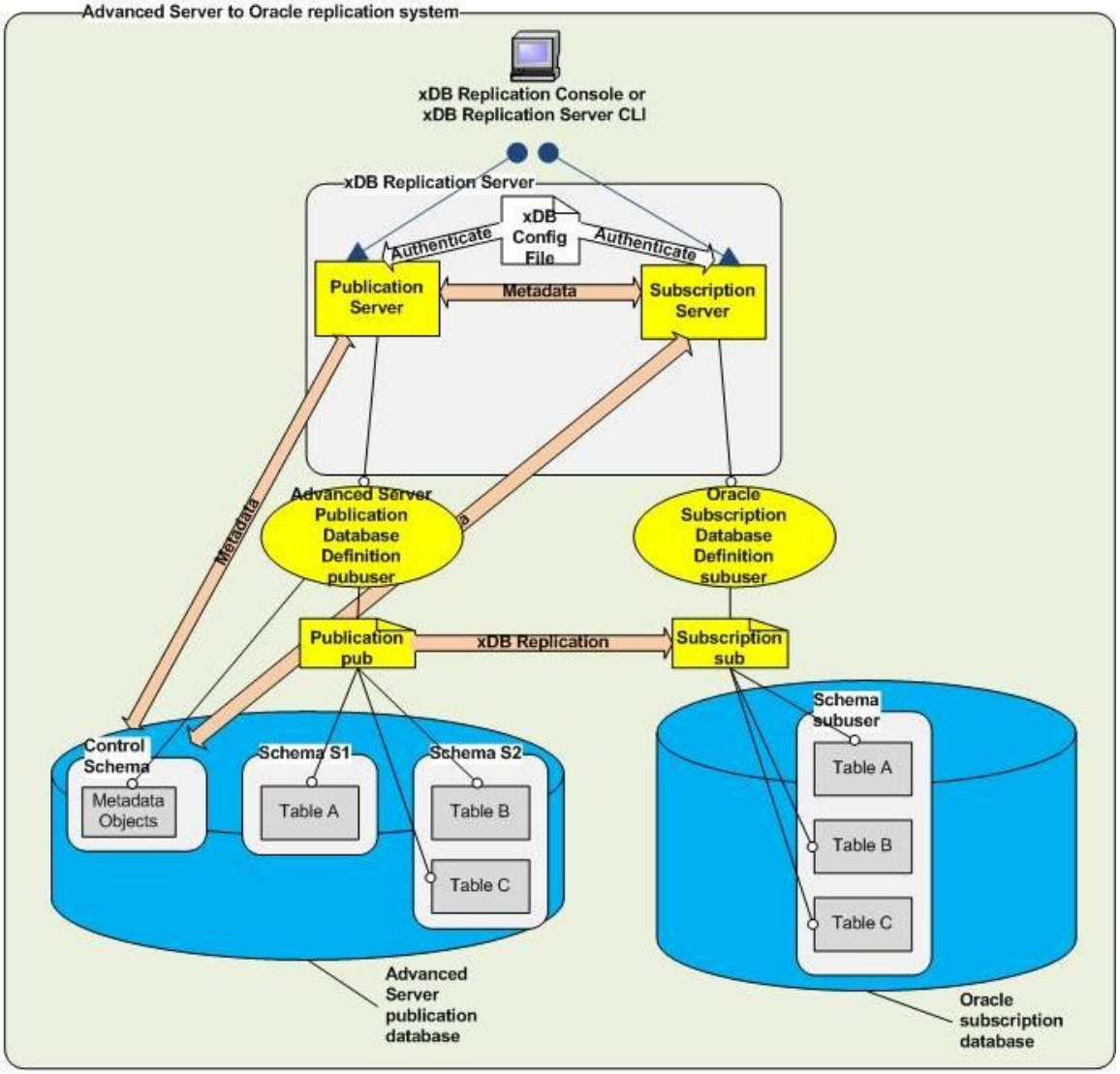


Figure 2-15 - Advanced Server to Oracle replication

The following describes the logical components in the preceding diagram:

- The publication server to be used is identified by registering its network location, user name, and password.
- A publication database definition is created subordinate to the SMR type node under the publication server. The Postgres database user name `pubuser` is specified in the definition along with the database network location and database identifier. The publication server creates the control schema consisting of three physical schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` and populates them with the control schema objects for the replication system's metadata when you create the publication database definition.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- A publication named `pub` is created subordinate to the publication database definition. The publication consists of table `A` in schema `S1` and tables `B` and `C` in schema `S2`.
- The subscription server to be used is identified by registering its network location, user name, and password.
- A subscription database definition is created subordinate to the subscription server. The Oracle database user name `subuser` is specified in the definition along with the database network location and database identifier.
- A subscription named `sub` is created subordinate to the subscription database definition. When you create a user named `subuser` in Oracle, a schema named `subuser` is automatically created by Oracle at the same time. The table definitions for tables `A`, `B`, and `C` are created in schema `subuser` when you create subscription `sub`. When replication occurs, the publication server populates these tables with rows from the publication.

The following screen capture shows how the logical components of this replication system appear in the xDB Replication Console replication tree.

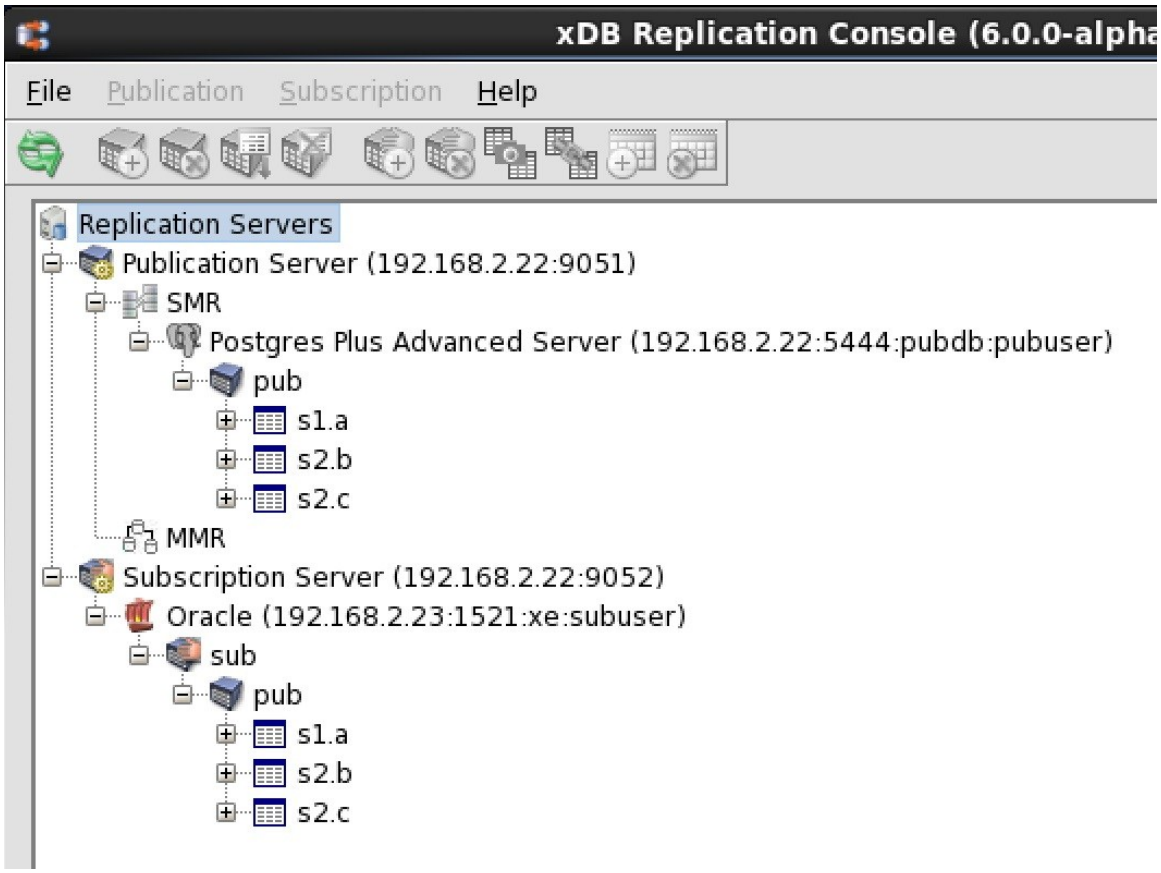


Figure 2-16 - Advanced Server to Oracle replication tree

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Chapter 4 for an introduction to the xDB Replication Console.

2.3.3.4 PostgreSQL or Advanced Server to SQL Server Replication

The following is an illustration of a basic PostgreSQL or Advanced Server to SQL Server single-master replication system. A single publication in a PostgreSQL or Advanced Server database contains tables from two schemas that are replicated to a SQL Server database.

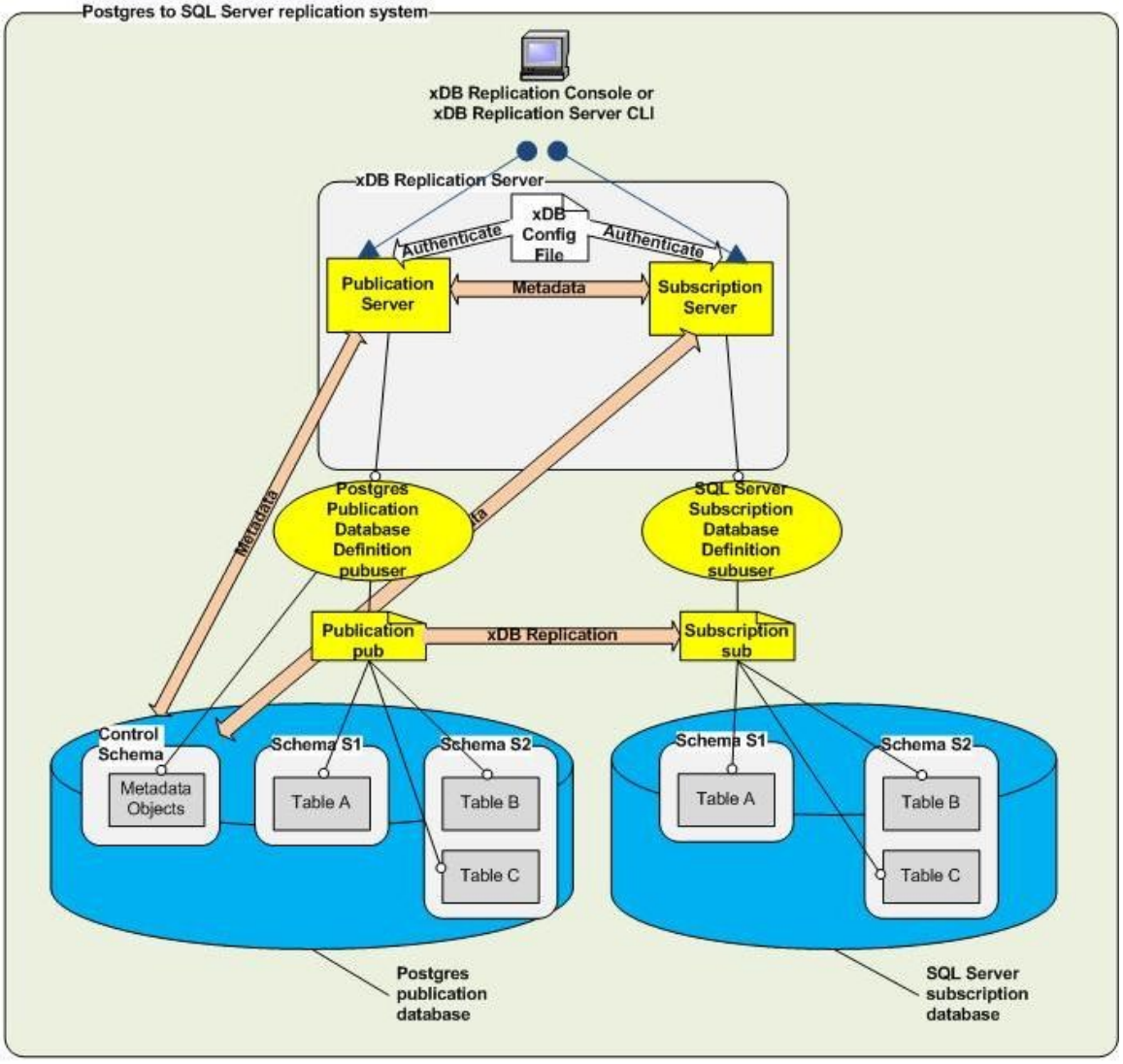


Figure 2-17 - PostgreSQL or Advanced Server to SQL Server replication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following describes the logical components in the preceding diagram:

- The publication server to be used is identified by registering its network location, user name, and password.
- A publication database definition is created subordinate to the SMR type node under the publication server. The Postgres database user name `pubuser` is specified in the definition along with the database network location and database identifier. The publication server creates the control schema consisting of three physical schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` and populates them with the control schema objects for the replication system's metadata when you create the publication database definition.
- A publication named `pub` is created subordinate to the publication database definition. The publication consists of table `A` in schema `S1` and tables `B` and `C` in schema `S2`.
- The subscription server to be used is identified by registering its network location, user name, and password.
- A subscription database definition is created subordinate to the subscription server. The SQL Server login `subuser` is specified in the definition along with the database network location and database identifier.
- A subscription named `sub` is created subordinate to the subscription database definition. When the subscription is created, the subscription server creates schemas named `S1` and `S2` in the subscription database. The table definitions for tables `A`, `B`, and `C` are also created at this time. When replication occurs, the publication server populates these tables with rows from the publication.

The following screen capture shows how the logical components of this replication system appear in the xDB Replication Console replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

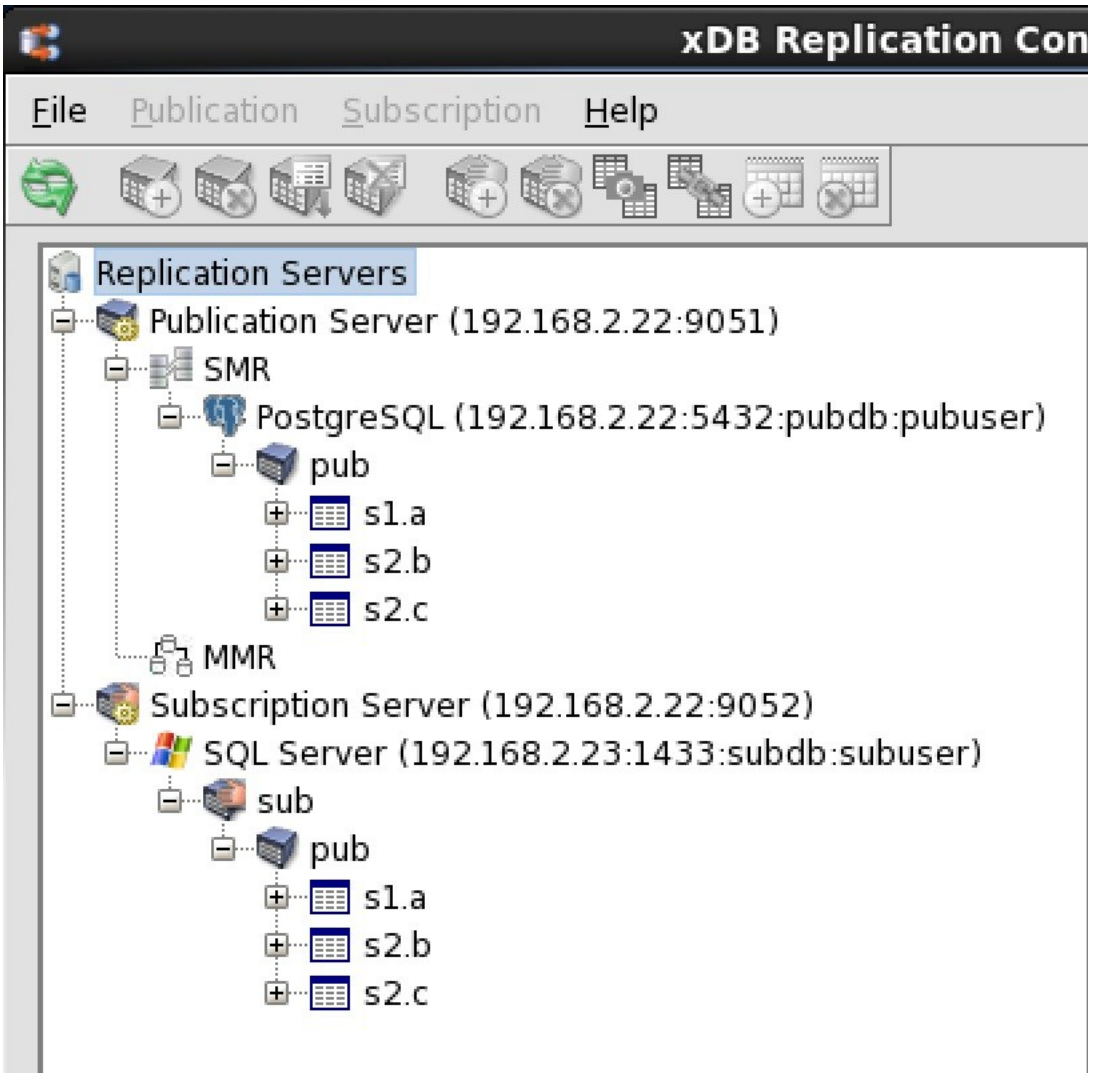


Figure 2-18 - Postgres to SQL Server replication tree

See Chapter 4 for an introduction to the xDB Replication Console.

2.3.3.5 Postgres Multi-Master Replication

The following is an illustration of a basic Postgres multi-master replication system. A publication in a Postgres master definition node contains tables from two schemas that are initially replicated to two other Postgres master nodes. The tables in all three master nodes can then be updated and synchronized with each other.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

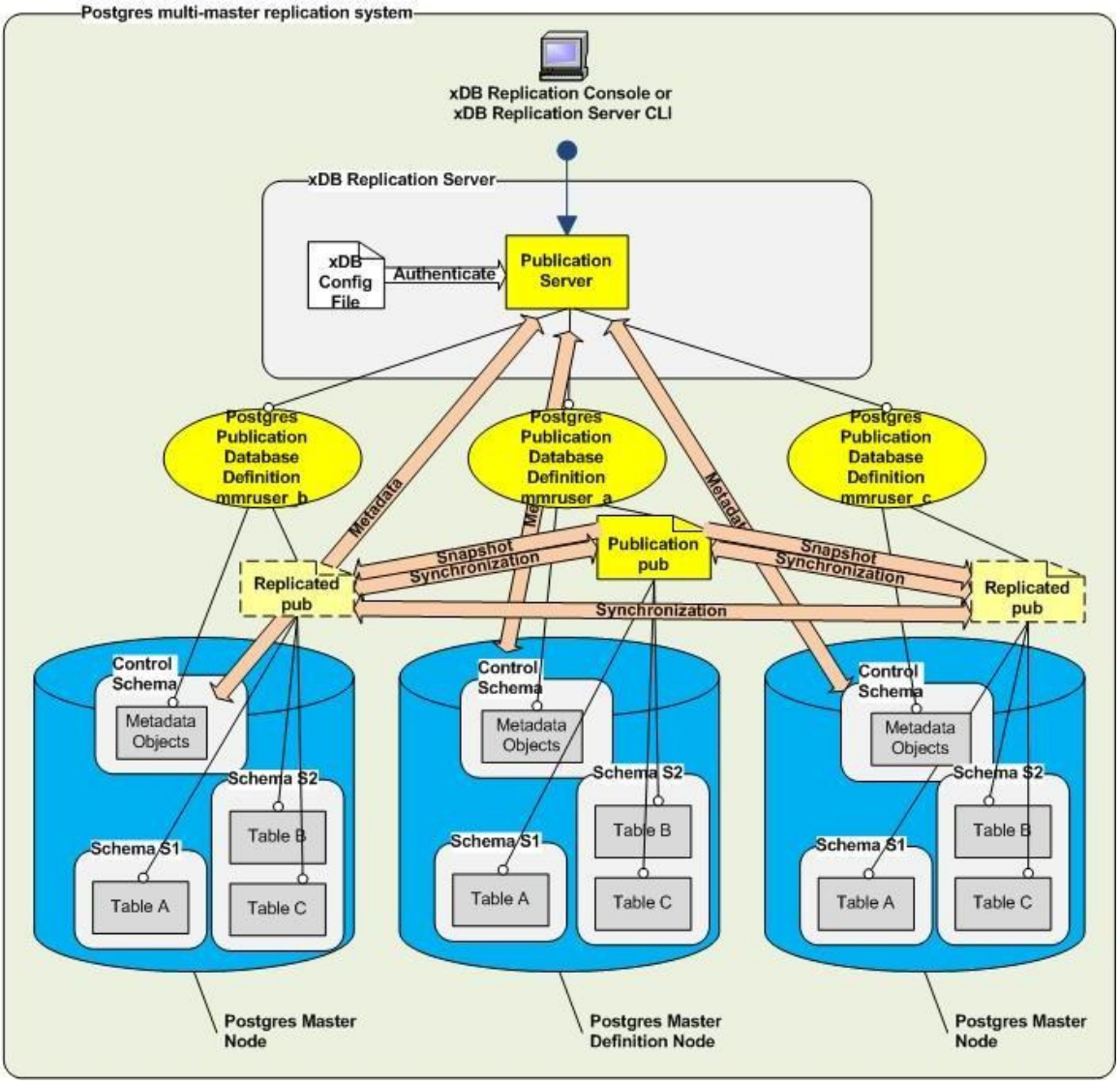


Figure 2-19 - Postgres multi-master replication system

The following describes the logical components in the preceding diagram:

- The publication server to be used is identified by registering its network location, user name, and password.
- A publication database definition is created subordinate to the MMR type node under the publication server. This first publication database definition identifies the master definition node. The Postgres database user name `mmruser_a` is specified in the definition along with the database network location and database identifier. The publication server creates the control schema consisting of three physical schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

_edb_scheduler and populates them with the control schema objects for the replication system's metadata when you create the publication database definition.

- A publication named pub is created subordinate to the publication database definition. The publication consists of table A in schema S1 and tables B and C in schema S2.
- A second master node is added by creating another publication database definition subordinate to the MMR type node of the publication server under which the master definition node resides. The Postgres database user name mmruser_b is specified in the definition along with the database network location and database identifier to create the second master node.
- When you add the second master node, you can choose to have the publication server create schemas S1 and S2 and the table definitions for A, B, and C for you, or you could have manually created the schemas and table definitions beforehand. The publication server creates the control schema consisting of three physical schemas _edb_replicator_pub, _edb_replicator_sub, and _edb_scheduler under which it creates the control schema objects to store the master node's metadata. When defining the master node, you can choose to have the publication server populate these tables with rows from the publication at this time, or you can defer table loading to a later point in time.
- A third master node is added in a similar manner using the Postgres database user name mmruser_c.

The following screen capture shows how the logical components of this replication system appear in the xDB Replication Console replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

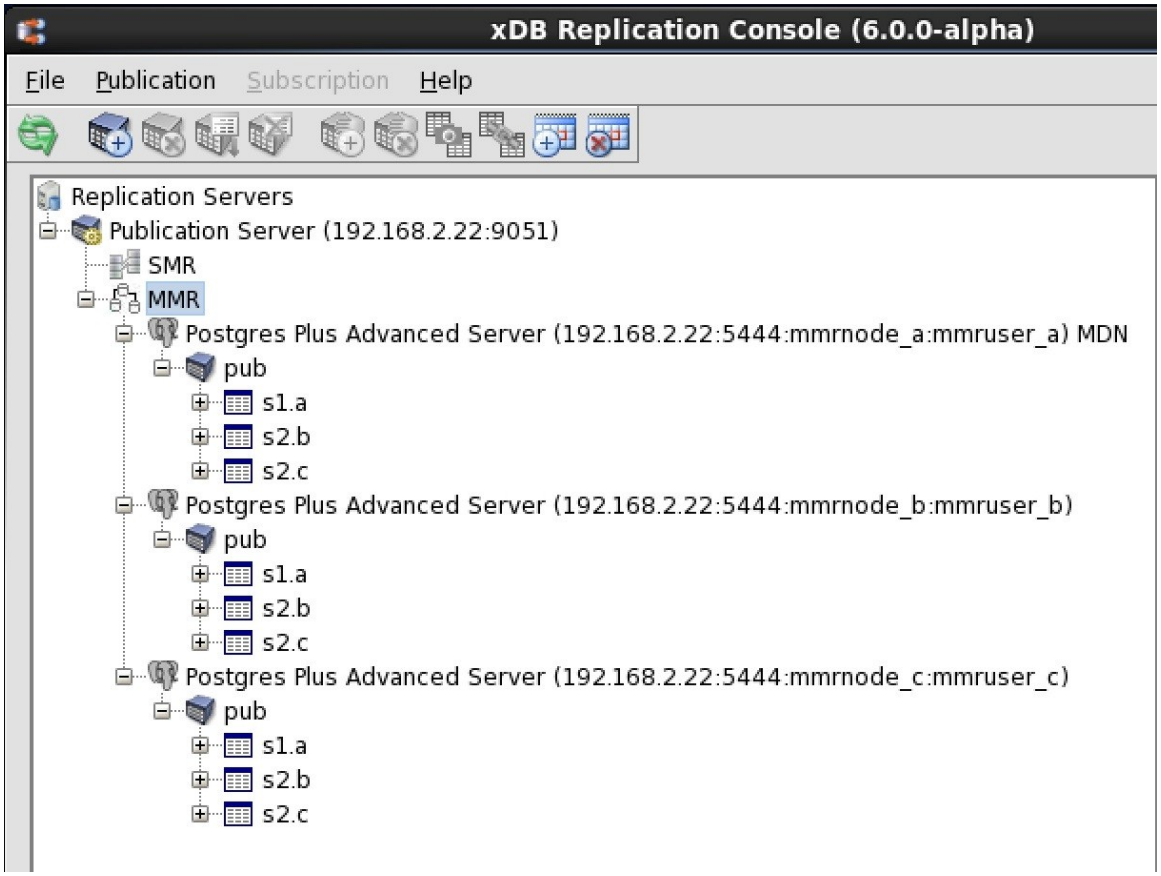


Figure 2-20 - Postgres multi-master replication tree

See Chapter 4 for an introduction to the xDB Replication Console.

2.4 Designing a Replication System

This section presents the general steps, design considerations, and best practices for designing a replication system before you begin the actual implementation.

2.4.1 General Steps

The following steps provide a general guideline for implementing a replication system.

Step 1: Determine if xDB Replication Server is the right solution for your requirements and you have chosen the best solution for your particular needs. xDB Replication Server can be used to implement single-master or multi-master replication systems. For single-master replication systems, the distinguishing characteristic of xDB Replication Server is its ability to replicate from an Oracle database to a PostgreSQL or Advanced Server database, from a SQL Server database to a PostgreSQL or Advanced Server database,

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

from an Advanced Server database to an Oracle database, or from a PostgreSQL or Advanced Server database to a SQL Server database.

Step 2: Plan the general strategy of how you will use xDB Replication Server. Will the single-master or multi-master model best suit your needs? (See Section 2.1 for use case examples of single-master and multi-master replication systems.) Will you be replicating from Oracle to Postgres, from SQL Server to Postgres, from Advanced Server to Oracle, or from Postgres to SQL Server? Will you be replicating between PostgreSQL and/or Advanced Server databases? How often will you need to replicate the data? Will replication be done on an ad hoc basis or does it need to occur regularly according to a schedule?

Step 3: Plan the logistics of your replication system. How many tables do you expect to replicate and what are their sizes in total number of bytes and number of rows? What percentage of rows do you expect to have been changed on each table between each replication? Are your database servers required to run on dedicated machines?

Step 4: Design your replication system. Determine whether your replication system will be distributed or will run on a single host. Determine the publications and subscriptions you will need and their tables and views. Make sure your publication tables meet the requirements for an xDB Replication Server publication. See sections 2.4.2 and 2.4.3 for details.

Step 5: Implement and test your replication system in a test environment. Try out your replication system on a subset of your publication data to ensure the replication process works as expected. Make sure the resulting replicated tables can be used as expected in your application. Establish preliminary metrics on how long the replication process will be expected to take in your full production environment.

Step 6: Implement and test your replication system in your production environment.

2.4.2 Design Considerations

Keep the following points in mind when designing a replication system:

- Multi-master replication is supported only on Postgres databases. In addition, Advanced Server databases must be running in the same compatibility mode – either all Oracle or all PostgreSQL.
- An Oracle table can be a member of at most one publication if all publications are subordinate to one publication database definition. However, an Oracle table can be a member of multiple publications if each publication is subordinate to a different publication database definition.
- A Postgres table can be a member of at most one publication.
- Each table used in a publication must have a primary key with the exception of tables in snapshot-only publications, which do not require a primary key.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Make sure table definitions are well established before creating publications. Unless the DDL change replication feature is used as described in Section 7.8, if a table definition is changed, any publication containing the table along with its associated subscription must be deleted and recreated, otherwise replication may fail. The same applies for the table definitions in a master definition node and its associated master nodes. Replication failures can be seen in the replication history.
- Views can be members of snapshot-only publications. In the subscription database, a view is replicated as a table.
- A publication may have multiple subscriptions.
- A subscription can be associated with at most one publication.
- A database can contain both publications and subscriptions.
- A given publication server can support only one multi-master replication system. All master nodes created subordinate to a given publication server are assumed to be part of the same multi-master replication system.
- A table that is created as a result of a subscription can be used in another publication. Thus, a publication can replicate data to a subscription which in turn, can be used in a publication to replicate to another subscription, thus creating a cascaded replication architecture.
- There are restrictions on the combinations and configurations of database servers that can be used for a publication and its subscription. See Section 10.1.3 for details on these restrictions.
- All replication system components must be running in order for replication to occur, or before performing any configuration, operation, or modification in the replication system. (The xDB Replication Console is used for the configuration and modification of a replication system. The xDB Replication Console does not need to be running in order for replication to occur.)
- In general, the order of creation of a replication system is as follows: 1) Create the required physical databases, database user names, tables, and views to be used in the replication system. 2) Define the replication system logical components using the xDB Replication Console or xDB Replication Server CLI. 3) Perform replication.
- In general, the order of removal of a single-master replication system is as follows: 1) Remove the replication system logical components using the xDB Replication Console or xDB Replication Server CLI starting with the subscriptions (Subscription nodes) and then their parent components (Subscription Database nodes). 2) Unregister the subscription server if you no longer have any need for it. 3) Repeat the same process for the publications. 4) After all replication system logical components have been removed (except for possibly the publication server and subscription server) you can drop any of the physical database objects in Oracle, SQL Server, or Postgres. **Do not drop the control schema objects manually, for example by using an SQL command line utility. Doing so may cause the xDB Replication Console and xDB Replication Server CLI to become inoperable.** (See Section 10.3.4.3 if this problem occurs.) Deleting the replication system logical components using the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

xDB Replication Console or xDB Replication Server CLI automatically drops the control schema objects from the physical database.

- The order of removal of a multi-master replication system is as follows: 1) Remove the replication system logical components using the xDB Replication Console or xDB Replication Server CLI starting with the publication database definitions of the non-MDN nodes. 2) Remove the publication from under the master definition node. 3) Remove the publication database definition of the master definition node. 4) After all replication system logical components have been removed (except for possibly the publication server) you can drop any of the physical database objects in Postgres. **Do not drop the control schema objects manually, for example by using an SQL command line utility. Doing so may cause the xDB Replication Console and xDB Replication Server CLI to become inoperable.**

2.4.3 Restrictions on Replicated Database Objects

When a subscription is created in a single-master replication system, the table definitions and most database objects and attributes associated with the publication tables are created in the subscription database by the subscription server.

If you so choose, the same process can automatically occur when a master node is added to a multi-master replication system. The table definitions and most database objects and attributes associated with the publication tables can be created in the newly added master node by the publication server.

The following is a list of database objects and table attributes that are replicated from the publication in either a single-master or multi-master replication system.

- Tables
- Views (for snapshot-only publications) created as tables in the subscription database
- Primary keys
- Not null constraints
- Unique constraints
- Check constraints
- Indexes

Note: Foreign key constraints are not replicated by the publication or subscription server in a single-master replication system. However, in a multi-master replication system, foreign key constraints are replicated from the master definition node to other master nodes.

Note: Sequences (database objects created by the `CREATE SEQUENCE` statement) are not replicated from the publication database to the subscription databases in a single-master replication system. Sequences are also not replicated from the master definition node to other master nodes in a multi-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

xDB Replication Server does have some restrictions on the types of tables it can replicate.

2.4.3.1 Restrictions on Oracle Database Objects

Certain types of Oracle partitioned tables can be replicated. See Section 10.4.1.4 for details.

Oracle global temporary tables cannot be replicated.

Oracle tables that include the following data types cannot be replicated:

- BFILE
- BINARY_DOUBLE
- BINARY_FLOAT
- MLSLABEL
- XMLTYPE

Oracle tables with the following data types can be used in snapshot-only publications, but cannot be used in synchronization replications:

- BLOB
- CLOB
- LONG
- LONG RAW
- NCLOB
- RAW

2.4.3.2 Restrictions on SQL Server Database Objects

SQL Server tables that include the following data types cannot be replicated:

- GEOGRAPHY
- GEOMETRY
- SQL_VARIANT

Note: See Section 10.4.6 for a method to replicate tables containing the SQL_VARIANT data type under certain conditions.

SQL Server tables with the following data types can be used in snapshot-only publications, but cannot be used in synchronization replications:

- BINARY
- IMAGE
- NTEXT

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- NVARCHAR(max)
- TEXT
- TIMESTAMP
- VARBINARY
- VARBINARY(max)

2.4.3.3 Restrictions on Postgres Database Objects

For replicating Postgres partitioned tables see Section 7.10 for details.

Postgres tables with the following data types in a column that is part of the primary key cannot be replicated:

- BLOB
- BYTEA
- RAW

Postgres tables that include OID based large objects cannot be replicated. For information on OID based large objects see `pg_largeobject` in the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/catalog-pg-largeobject.html

Postgres tables that include any *geometric* data types such as `POINT`, `POLYGON`, etc., cannot be replicated to an Oracle subscription database.

Postgres tables that include the following data types cannot be replicated to a SQL Server subscription database:

- ABSTIME
- ACLITEM
- CHKPASS
- CIRCLE
- CUBE
- JSON
- ROWID
- SEG
- Any `ARRAY` data type (that is, defined as *`data_type`*`[]`)
- Any user-defined data type (that is, defined as `CREATE TYPE` *`type_name`*)

Restrictions on Range Data Types

Postgres data types called *range types* were first supported in PostgreSQL version 9.2 and Advanced Server version 9.2. *Built-in range types* refer to the following built-in data types: `int4range`, `int8range`, `numrange`, `tsrange`, `tstzrange`, and `daterange`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Postgres tables containing the built-in range types can be included in the publication of a single-master or multi-master replication system.

However, this results in the following restrictions on the subscription databases of a single-master replication system or the additional master nodes of a multi-master replication system:

- If a publication table of a single-master replication system contains any built-in range types, then a database can be added as a subscription database only if the database server of the intended subscription database is Postgres version 9.2 or later.
- If a publication table of the master definition node in a multi-master replication system contains any built-in range types, then a database can be added as an additional master node only if the database server of this intended master node is Postgres version 9.2 or later.

Custom range types constructed with the `CREATE TYPE AS RANGE` command are not supported in xDB Replication Server.

2.4.4 Performance Considerations

This section discusses provides some general guidelines on performance considerations.

2.4.4.1 When to Use Snapshot or Synchronization

Generally, synchronization would be the quickest replication method since it only applies changes to the target tables since the last replication occurred.

However, if a large percentage of rows are changed between each replication, there may be a point where it would be faster to completely reload the target tables using a snapshot than to execute individual SQL statements on a large percentage of rows as would be done for synchronization replication. Experimentation may be necessary to determine if, and at what point a snapshot would be faster.

Snapshot replication may be an option for tables with the following characteristics:

- Tables are relatively small in size
- A large percentage of rows are changed between replications

Synchronization replication is the better option for tables with the following characteristics:

- Tables are large in size
- A small percentage of rows are changed between replications

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In a single-master replication system, if you find that one group of tables consistently replicates faster using snapshot replication, then these tables can be made part of a snapshot-only publication while the remaining tables can be members of a publication that uses synchronization replication.

2.4.4.2 When to Use On Demand Replication

The xDB Replication Console and xDB Replication Server CLI both give you the capability to immediately start a replication. This is called an *on demand* replication.

On demand replication can be performed at any time regardless of whether or not there is an existing schedule. An on demand replication does not change the date and time when the next replication is scheduled to occur according to an existing schedule.

If a publication is a snapshot-only publication, then the only type of on demand replication that can be performed on this publication is a snapshot.

If a publication is not a snapshot-only publication, you can perform an on demand replication using either the snapshot method or the synchronization method.

When you are in the development and testing phases of your replication system, you would typically use on demand replication so that you can immediately force the replication to occur and analyze the results.

When your replication system is ready for production, a schedule would typically be used so that replications can occur unattended at regular time intervals. See Section 7.2 for directions on creating a schedule.

There may be other situations where you would want to force a replication to take place ahead of its normal schedule. Reasons for performing an on demand replication may include the following:

- The number of changes to the source tables is growing at a faster rate than usual, and you do not want to wait for the regularly scheduled synchronization time to replicate all of the accumulated changes.
- You have setup your replication system to perform synchronizations, but on this occasion there have been an unusually large number of changes made to the source tables, and you would rather perform a snapshot of all source tables rather than execute a large number of SQL statements against the target tables.
- Changes have been made directly to the rows of the target tables so that they no longer have the same content as their source table counterparts. You can perform an on demand snapshot replication to reload all rows of the target tables from your current set of source tables.

Note: In a multi-master replication system, on demand snapshots can only be made from the master definition node to another master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Section 5.4 for directions on performing an on demand replication for a single-master replication system. See Section 6.5 for a multi-master replication system.

2.4.5 Distributed Replication

xDB Replication Server provides the flexibility of allowing you to run the replication system's components on separate machines on a network.

In fact xDB Replication Server is designed so that it is possible to set up replication systems where each of the components (publication server, subscription server, publication database, subscription database, and master nodes) may all run on the same host, each component may run on its own separate host, or any combination of components may run on any number of hosts.

However, for practical purposes, there are two basic scenarios. The simplest case is where all components are on the same host. The other case is where you have the Oracle or SQL Server database server running on a host separate from the rest of the replication system components.

This section discusses the advantages and disadvantages of each scenario.

2.4.5.1 Single Host

The simplest implementation of a replication system is when all replication components run on a single host. This means that the PostgreSQL or Advanced Server installation, the complete xDB Replication Server installation (publication server and subscription server), and the Oracle or SQL Server database server reside on the same machine.

The Postgres publication or subscription database as the case may be, can reside in the initial database cluster that is created when Postgres is installed on the host.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

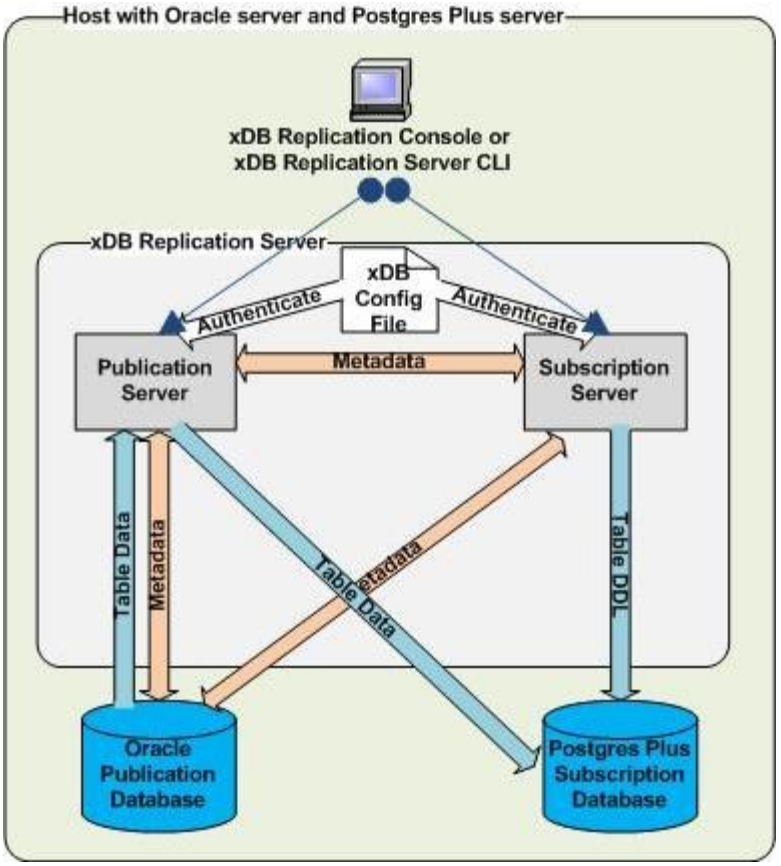


Figure 2-21 - Single host replication system

The advantages of a single host replication system are the following:

- There is a performance advantage since there is no network over which to push replication data, especially if large snapshots are involved.
- Configuration is much simpler. When creating the replication system logical components, the IP addresses of all components are the same.

The disadvantages of a single host replication system are the following:

- The replication system and the database servers all consume the resources of one machine, which can adversely affect database application performance.
- The publication and subscription databases may be in different geographic locations, thereby requiring multiple networked hosts.
- Your site may require the use of a dedicated host for the Oracle or SQL Server database server so xDB Replication Server could not reside on the same machine.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

2.4.5.2 Single-Master Replication Distributed Hosts

xDB Replication Server allows you to build a replication system with either or both of the publication database and the subscription database on separate hosts. This is illustrated in the following diagram:

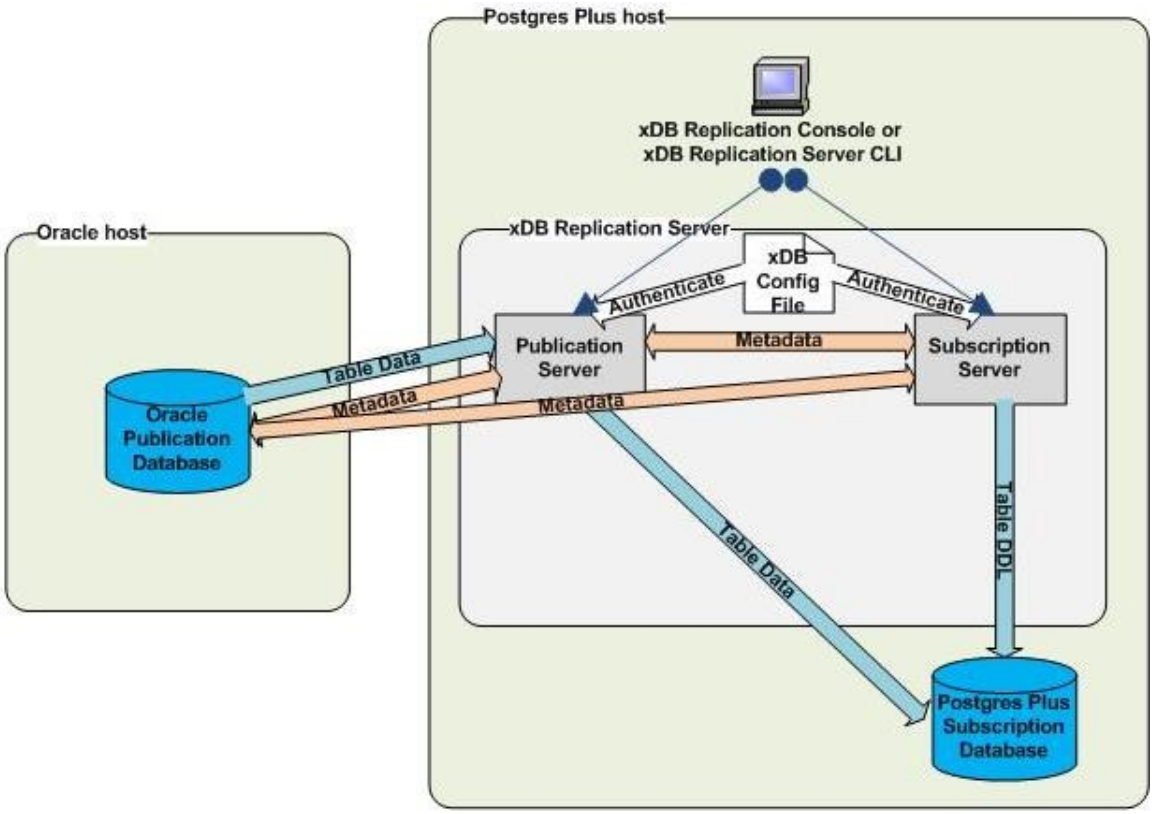


Figure 2-22 - Oracle database server on distributed host

The same remote distribution can be used for the subscription database instead of, or in addition to the publication database.

The advantages of a distributed host replication system are the following:

- The replication system and the database servers can each consume the resources of their own machines, which can be individually selected and tuned.
- The publication and subscription databases can be in different geographic locations.
- You can enforce stronger database security if only the database server is allowed to run on a host.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The disadvantages of a distributed host replication system are the following:

- There may be a performance disadvantage since there is a network over which to push replication data, especially if large snapshots are involved.
- Installation is more complex if the Postgres database must run on a different host than xDB Replication Server. This involves installing Postgres on two separate hosts.
- Configuration is more complex. The network and firewalls must be properly configured to allow the distributed components to communicate. When creating the replication system logical components, the correct IP addresses of all components must be used. In addition, the correct IP addresses must be kept up-to-date in the replication system control schema should they change in the networked environment.

2.4.5.3 Multi-Master Replication Distributed Hosts

In a multi-master replication system, the Postgres database servers running the master nodes can be running on a single or multiple hosts. The following example illustrates two master nodes running on database servers on separate hosts as well as a master node running on the same database server as the publication server.

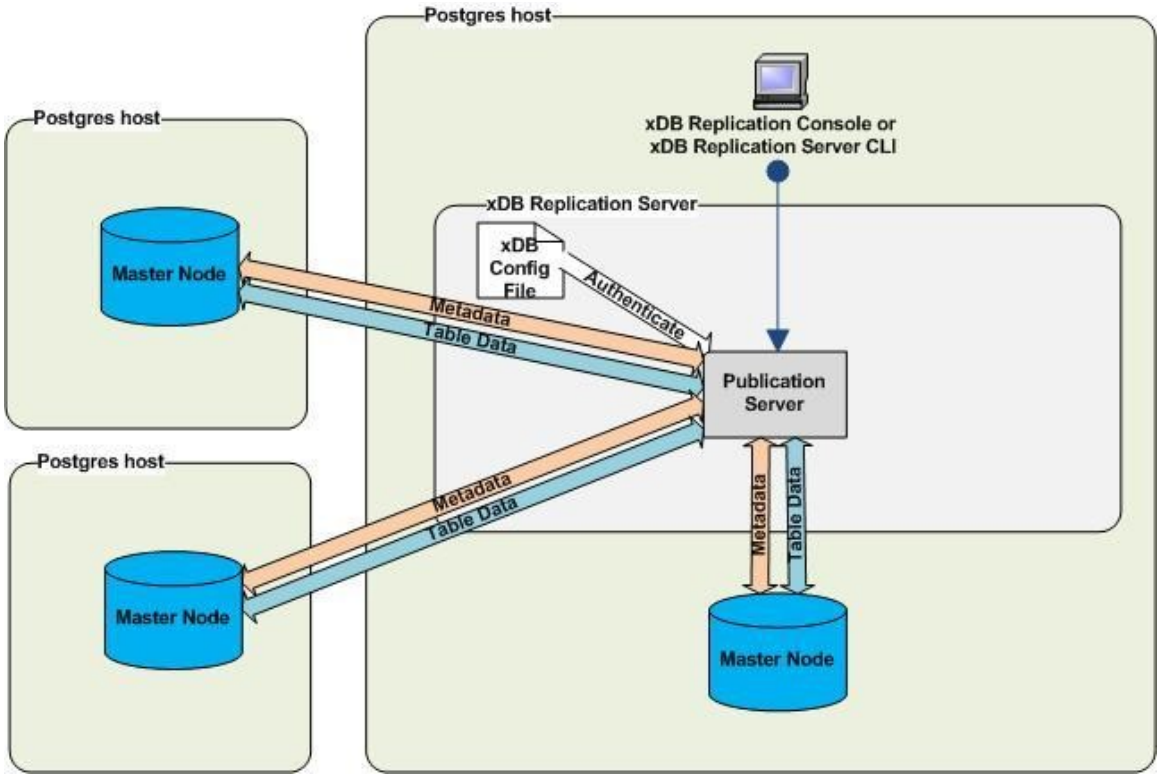


Figure 2-23 - Multi-master replication on distributed hosts

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3 Installation and Uninstallation

This chapter describes how to install and uninstall xDB Replication Server.

Installation of xDB Replication Server can be accomplished a number of different ways:

- Using a graphical user interface
- Running the xDB Replication Server installer program from the command line console in text or unattended mode
- Installing the xDB RPM package using the Yum package manager

The most common installation of xDB Replication Server is done with the graphical user interface invoked by Stack Builder or StackBuilder Plus depending upon whether you are using PostgreSQL or Advanced Server.

- **For PostgreSQL.** Install xDB Replication Server using Stack Builder after you have installed PostgreSQL.
- **For Advanced Server.** Install xDB Replication Server using StackBuilder Plus after you have installed Advanced Server.

For circumstances in which you do not wish to use the graphical user interface, the xDB Replication Server installer program can be downloaded from the EnterpriseDB website, and then invoked in text or unattended mode as well as the graphical user interface mode. See Section 3.2 for instructions on installing xDB Replication Server from the command line.

The xDB Replication Server product is also available as an RPM package in which case the Yum package manager is used for installation. See Section 3.3 for instructions on installing xDB Replication Server from the RPM package.

Section 3.1 describes the installation of xDB Replication Server through the graphical user interface of Stack Builder or StackBuilder Plus.

Note: If you have an older version of xDB Replication Server and existing replication systems, review Section 10.2 before installing xDB Replication Server.

If you later decide you wish to remove xDB Replication Server from your system see Section 3.6 for directions on uninstalling xDB Replication Server if you initially installed it with the graphical user interface or by invoking the installer program from the command line. See Section 3.7 for directions on uninstalling xDB Replication Server that was installed from the RPM package.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3.1 Installing With Stack Builder or StackBuilder Plus

Stack Builder and *StackBuilder Plus* are programs used to download and install add-on products and updates to PostgreSQL and Advanced Server. Stack Builder is used for PostgreSQL. StackBuilder Plus is used for Advanced Server.

Stack Builder and StackBuilder Plus are very similar in functionality and look-and-feel, differing primarily in the list of products offered.

This section demonstrates the installation of xDB Replication Server using StackBuilder Plus for Advanced Server. Steps are noted where the installation process differs for installation on PostgreSQL using Stack Builder.

Step 1: You must have Java Runtime Environment (JRE) version 1.7 or later installed on the hosts where you intend to install any xDB Replication Server component (xDB Replication Console, publication server, or subscription server). Any Java product such as Oracle Java or OpenJDK may be used.

Follow the directions for your host operating system to install Java runtime.

For Windows only: Be sure the system environment variable, `JAVA_HOME`, is set to the JRE installation directory of the JRE version and bitness (32-bit or 64-bit) you wish to use with the xDB Replication Server. The xDB Replication Server installer for a Windows platform contains both the 32-bit and 64-bit versions. The `JAVA_HOME` setting determines whether the 32-bit or the 64-bit version of xDB Replication Server is installed. (If `JAVA_HOME` is not set, then the first JRE version encountered in the `Path` system environment variable determines the xDB Replication Server version to be installed.)

Note: For Advanced Server versions prior to 9.3, a Java runtime is supplied and installed as part of the Advanced Server installation process, however, you must still have pre-installed a separate Java runtime system on your host. The xDB Replication Server installation process does not utilize the Java runtime supplied with Advanced Server.

Note: After installation of xDB Replication Server has completed, the path to your Java runtime program is stored in the xDB Startup Configuration file used by xDB Replication Server. Verify that the path to your Java runtime program set in the xDB Startup Configuration file is correct. See Section 3.5 for the location of this file.

Step 2: From the host's application menu, open the Postgres menu and choose Stack Builder or StackBuilder Plus.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

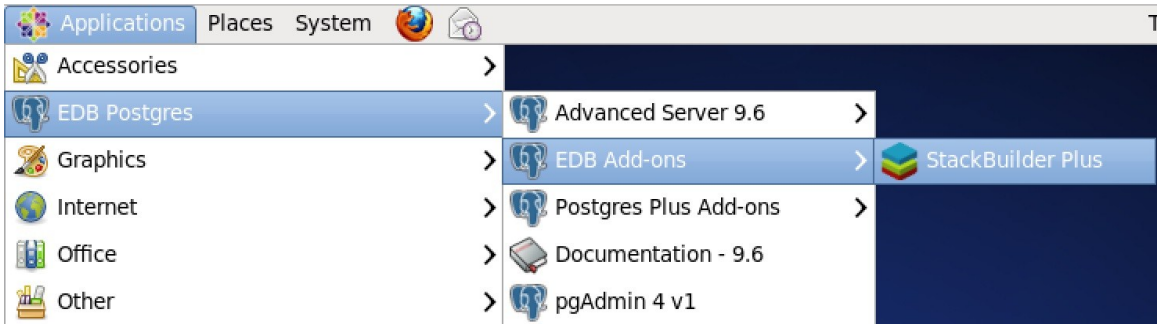


Figure 3-1 - Postgres application menu

Step 3 (For Linux only): Depending upon your Linux host, a dialog box or a prompt appears requesting the `root` account's password. Enter the `root` password and click the OK button.

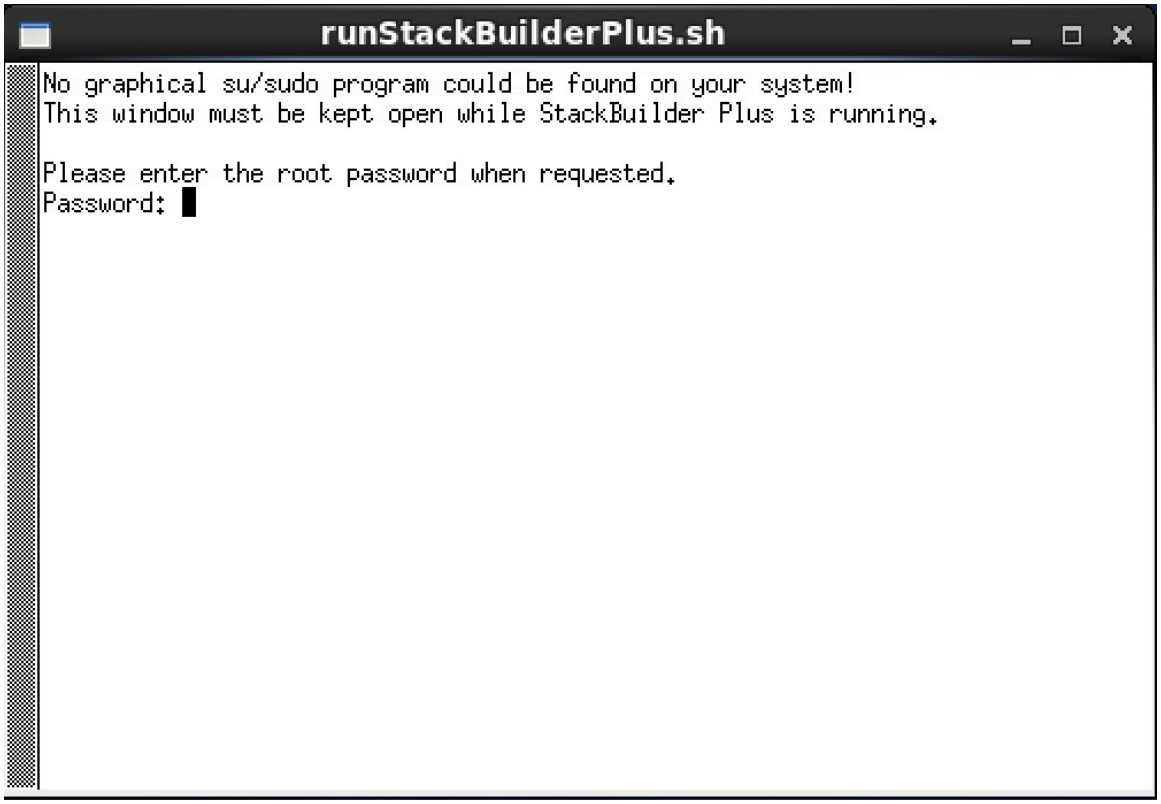


Figure 3-2 - Enter root account password

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4: The StackBuilder Plus welcome screen appears. Select your Postgres installation from the drop-down list and click the Next button.

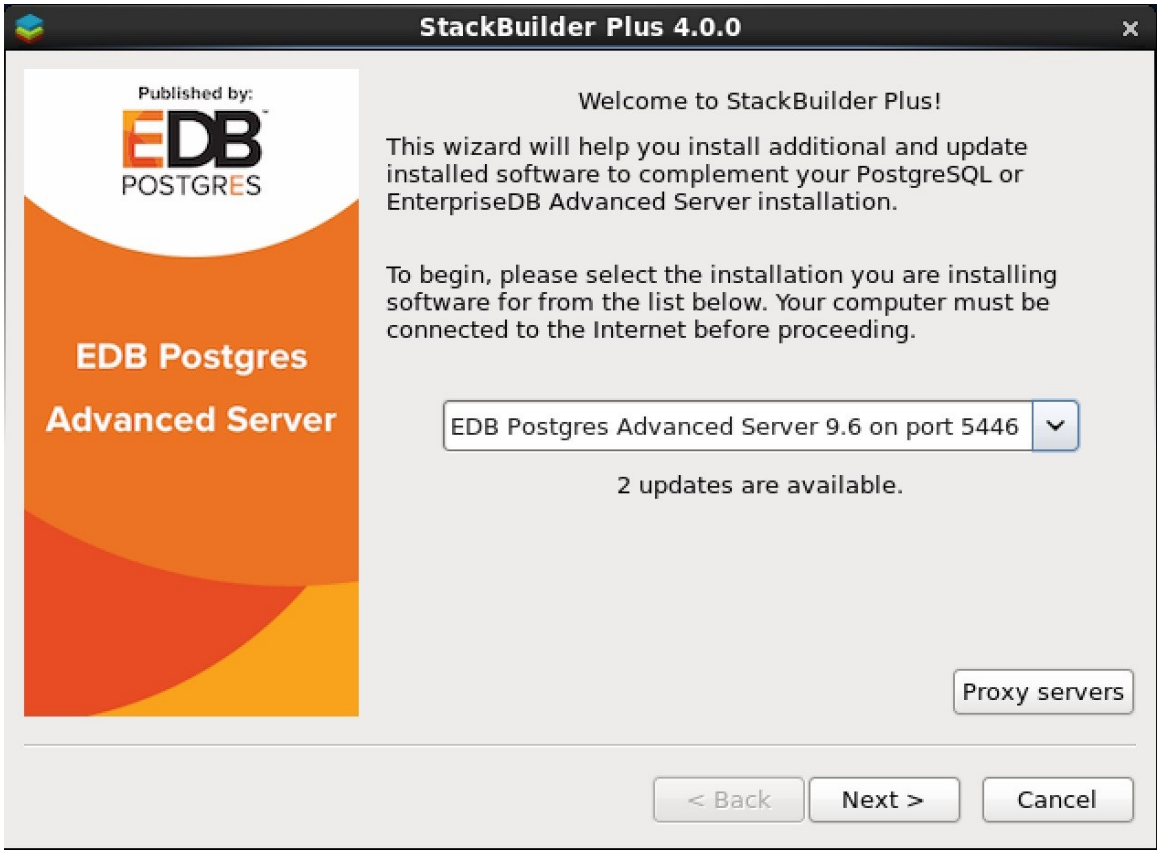


Figure 3-3 - StackBuilder Plus welcome screen

Step 5 (For Advanced Server): Expand the EnterpriseDB Tools node and check the box for Replication Server. Click the Next button.

Note: Though the following images show Replication Server v6.0, use the same process for Replication Server v6.2.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

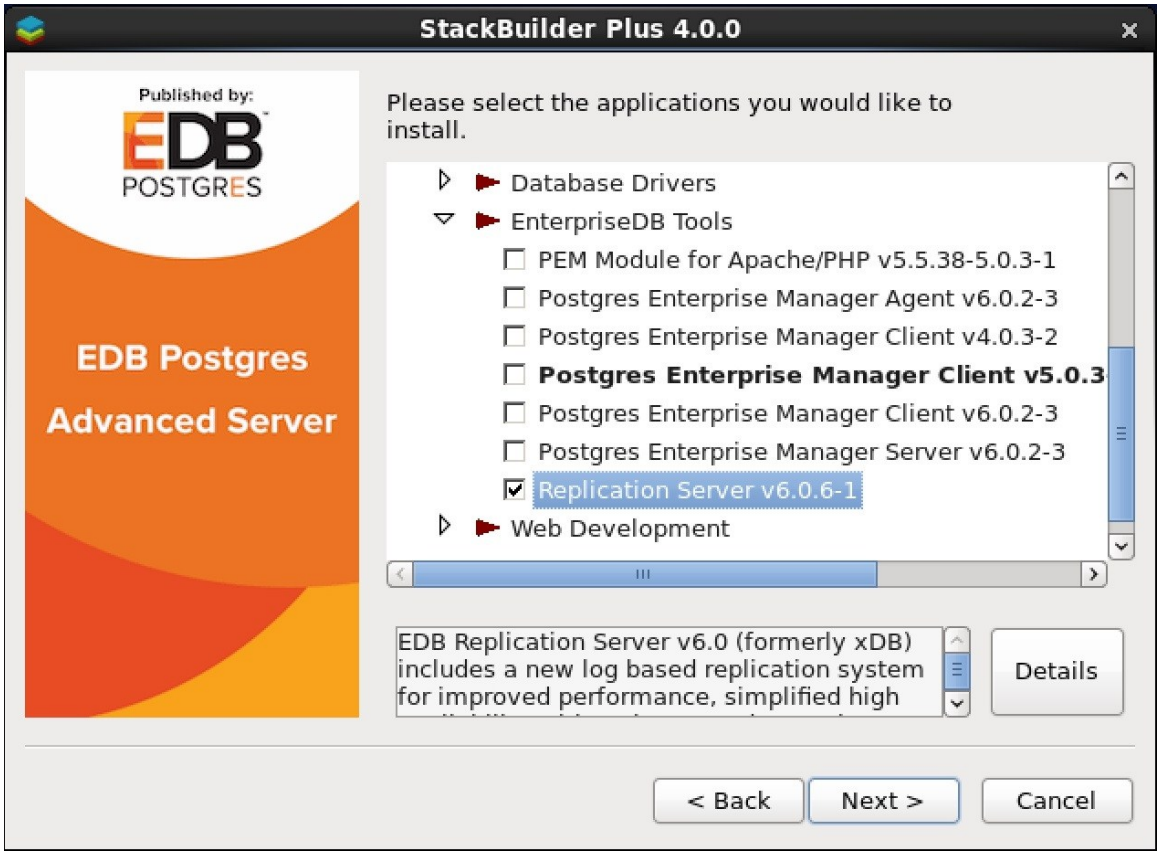


Figure 3-4 - StackBuilder Plus applications

Step 5 (For PostgreSQL): Expand the Registration-Required and Trial Products node, and then expand the EnterpriseDB Tools node. Check the box for Replication Server under the EnterpriseDB Tools list and click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

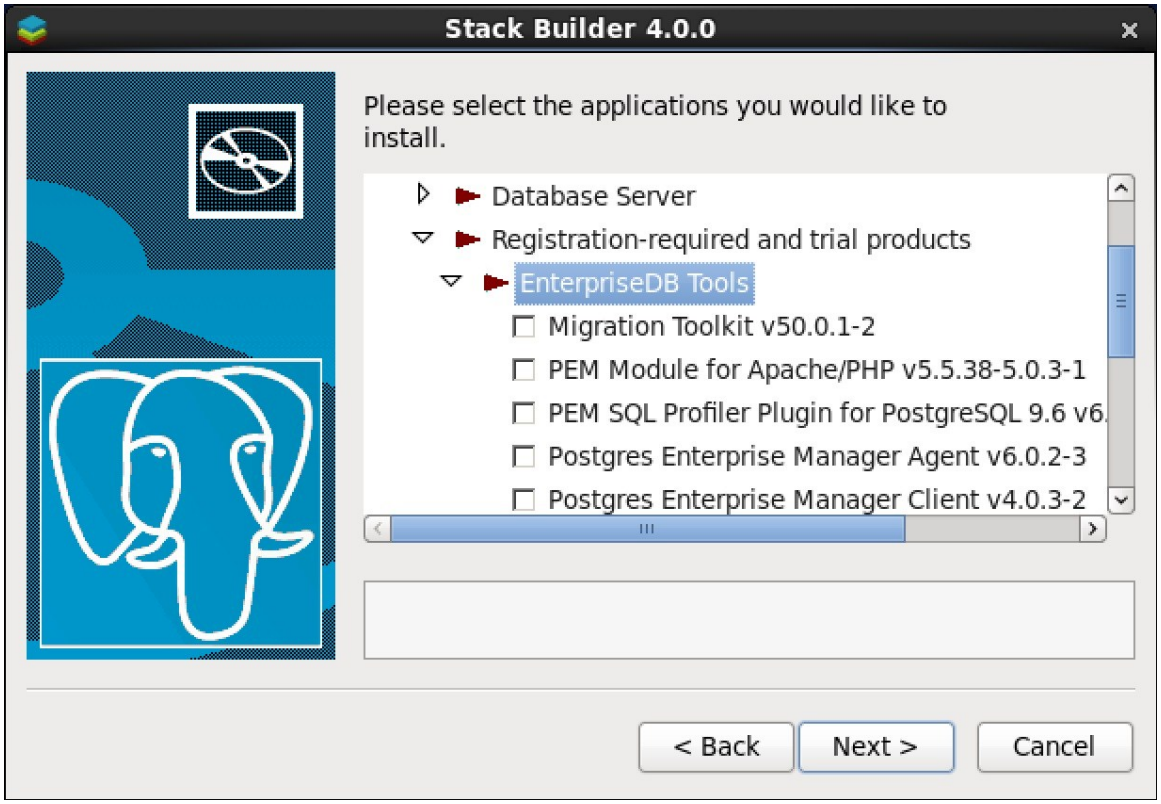


Figure 3-5 - Stack Builder applications

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

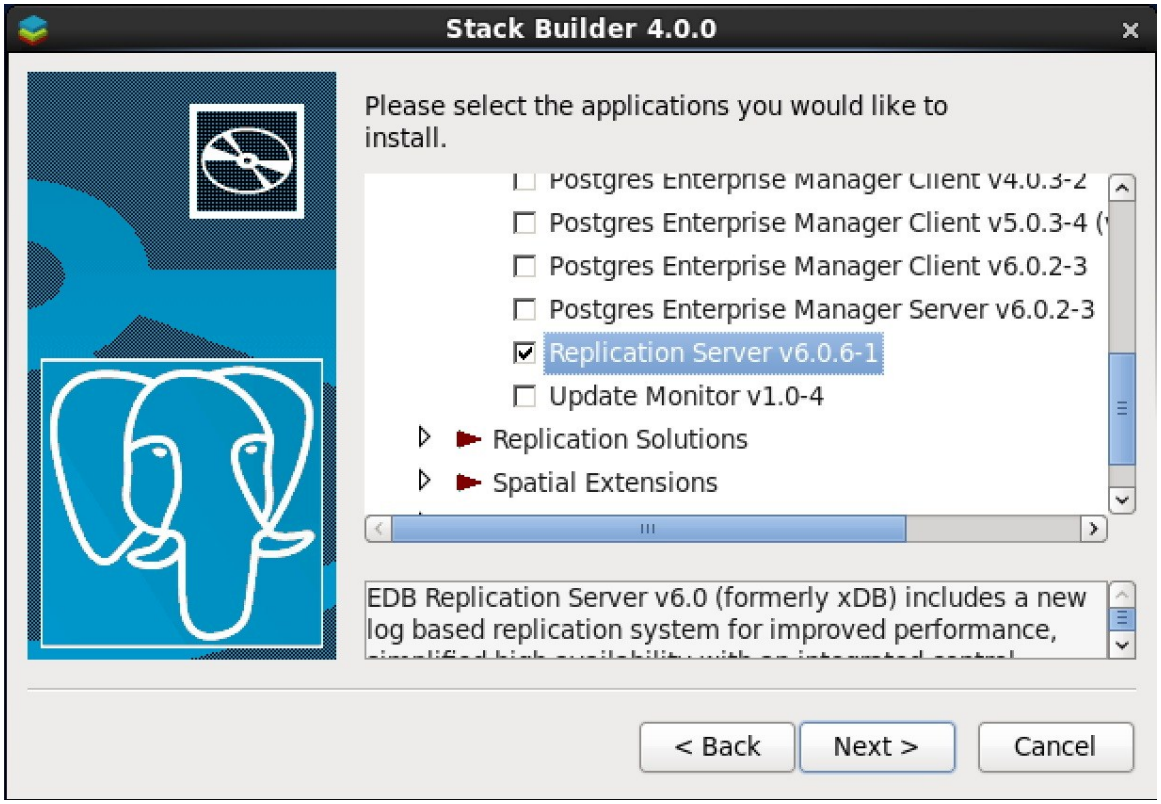


Figure 3-6 - EnterpriseDB Tools for PostgreSQL

Step 6 (For Advanced Server only): In the Account Registration screen, either enter your email address and password for your EnterpriseDB user account if you have one, or click the link in which case you will be directed to the registration page of the EnterpriseDB website where you can create an account. Click the Next button.

Note (For PostgreSQL only): Proceed to Step 7. If you are using PostgreSQL, account registration occurs later in the process.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

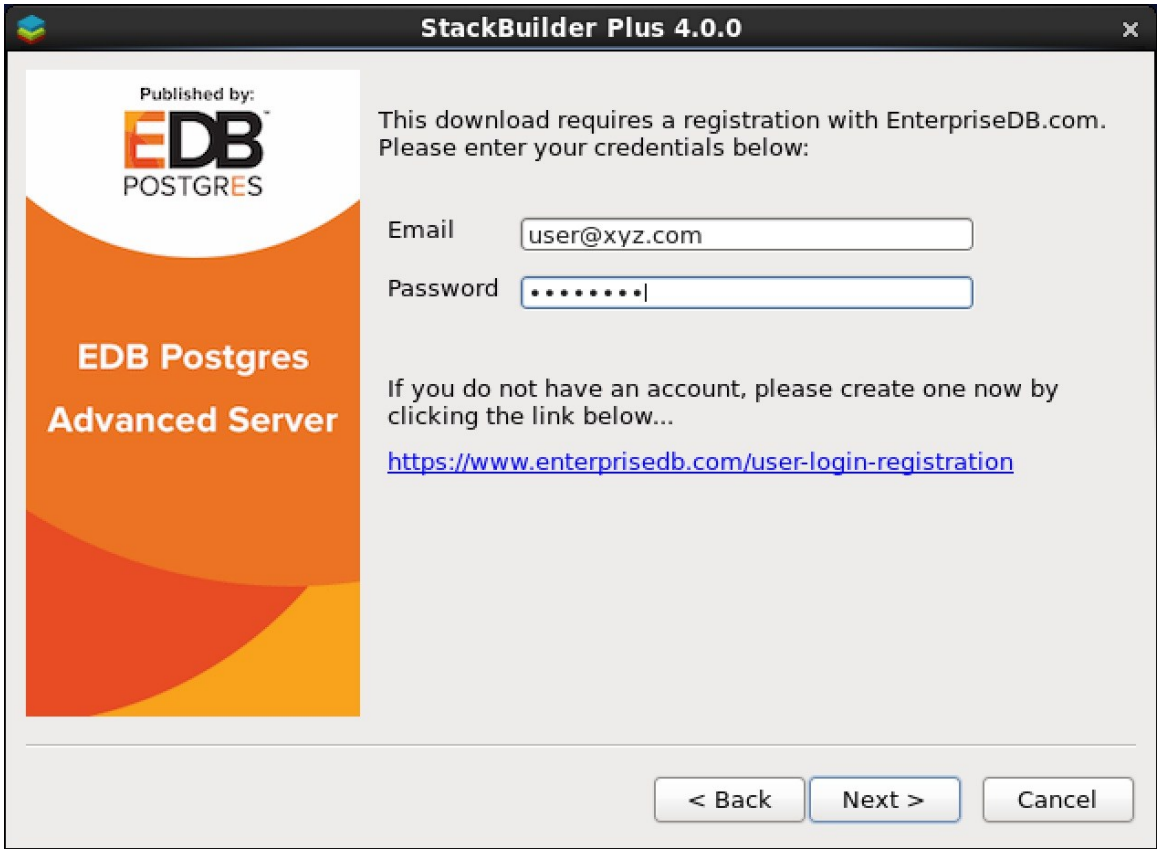


Figure 3-7 - EnterpriseDB account registration

Step 7: Verify that Replication Server appears in the list of selected packages. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

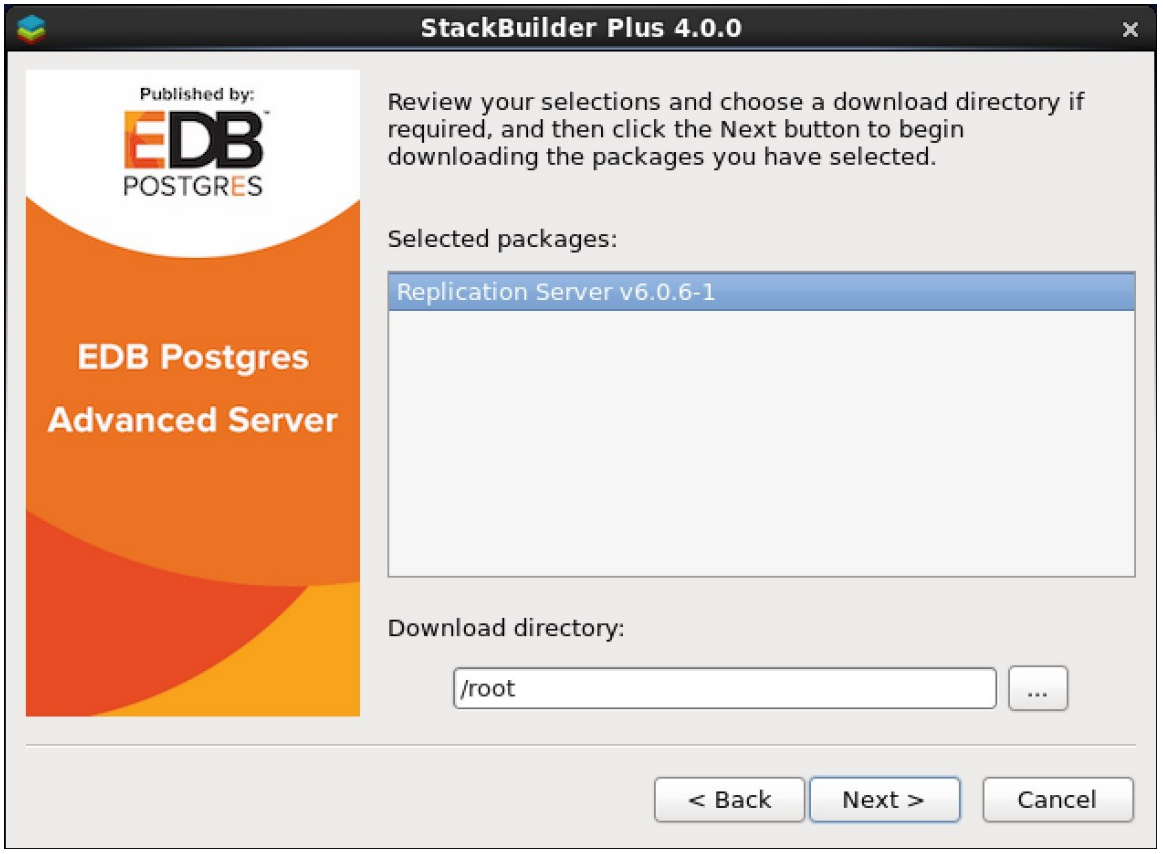


Figure 3-8 - Selected packages

An information box appears showing the download progress of the Replication Server package. This may take a few minutes.

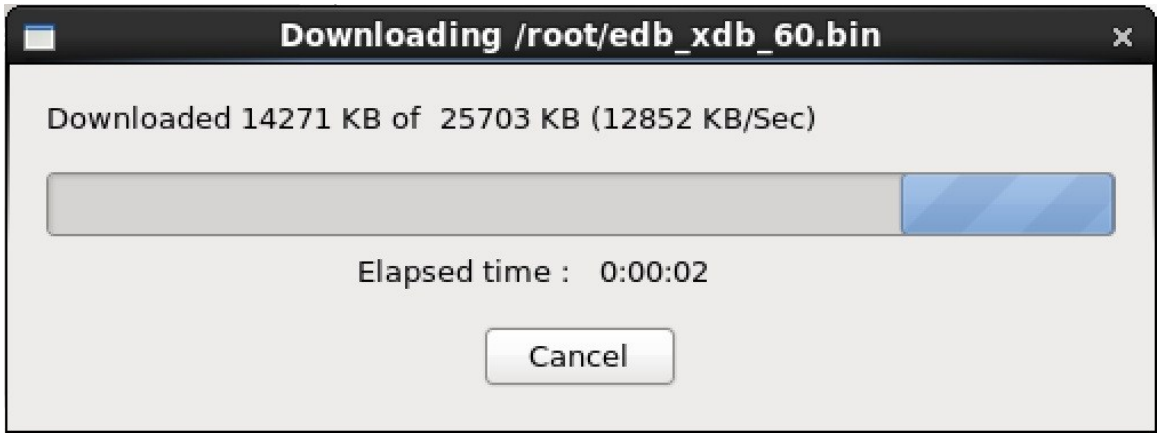


Figure 3-9 - Downloading progress

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 8: When downloading of the Replication Server package completes, the following screen appears that starts the installation of xDB Replication Server. Click the Next button.

Note: You can check the Skip Installation box if you wish to install xDB Replication Server some other time.

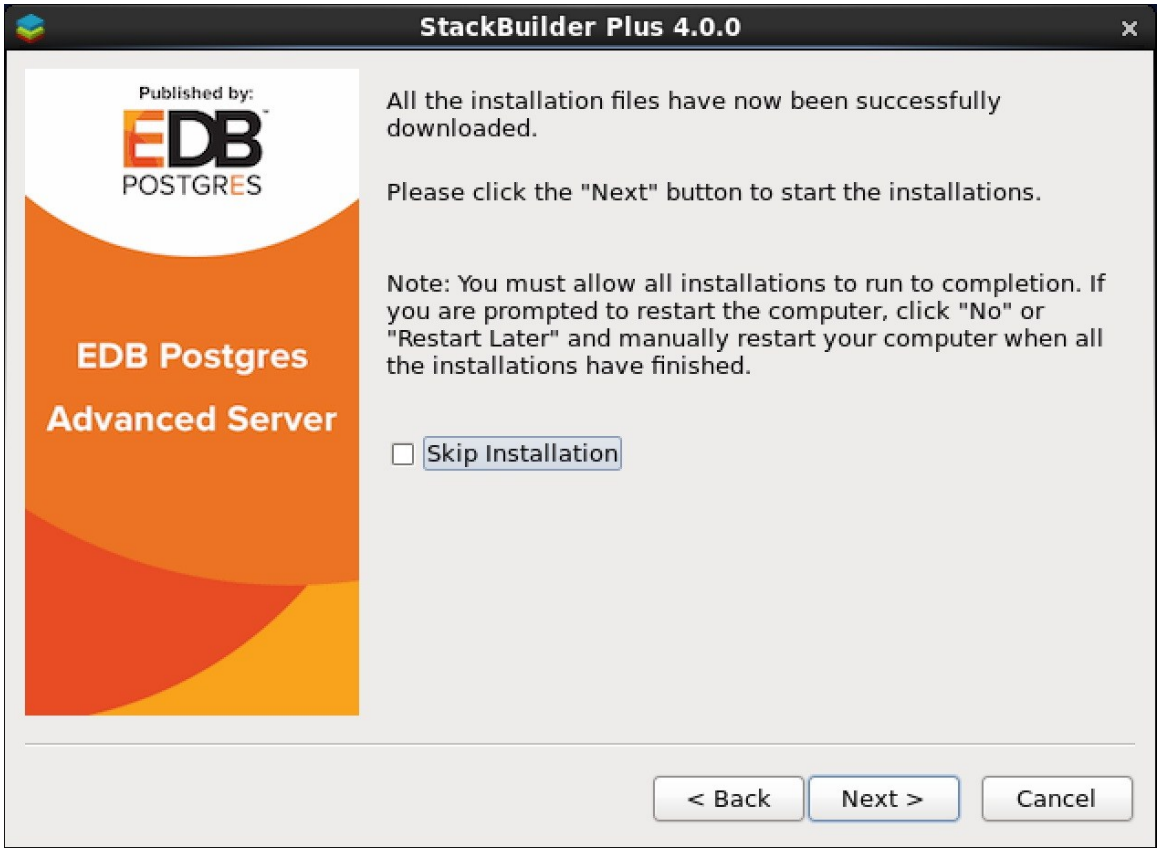


Figure 3-10 – Start installation

Step 9: Select the installation language and click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Figure 3-11 - Installation language

Step 10: In the Setup xDB Replication Server screen, click the Next button.

Figure 3-12 – Setup xDB Replication Server

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 11: Read the license agreement. If you accept the agreement, select the accept radio button and click the Next button.

Figure 3-13 - License agreement

Step 12: Browse to a directory where you want the xDB Replication Server components installed, or allow it to install the components in the default location shown. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

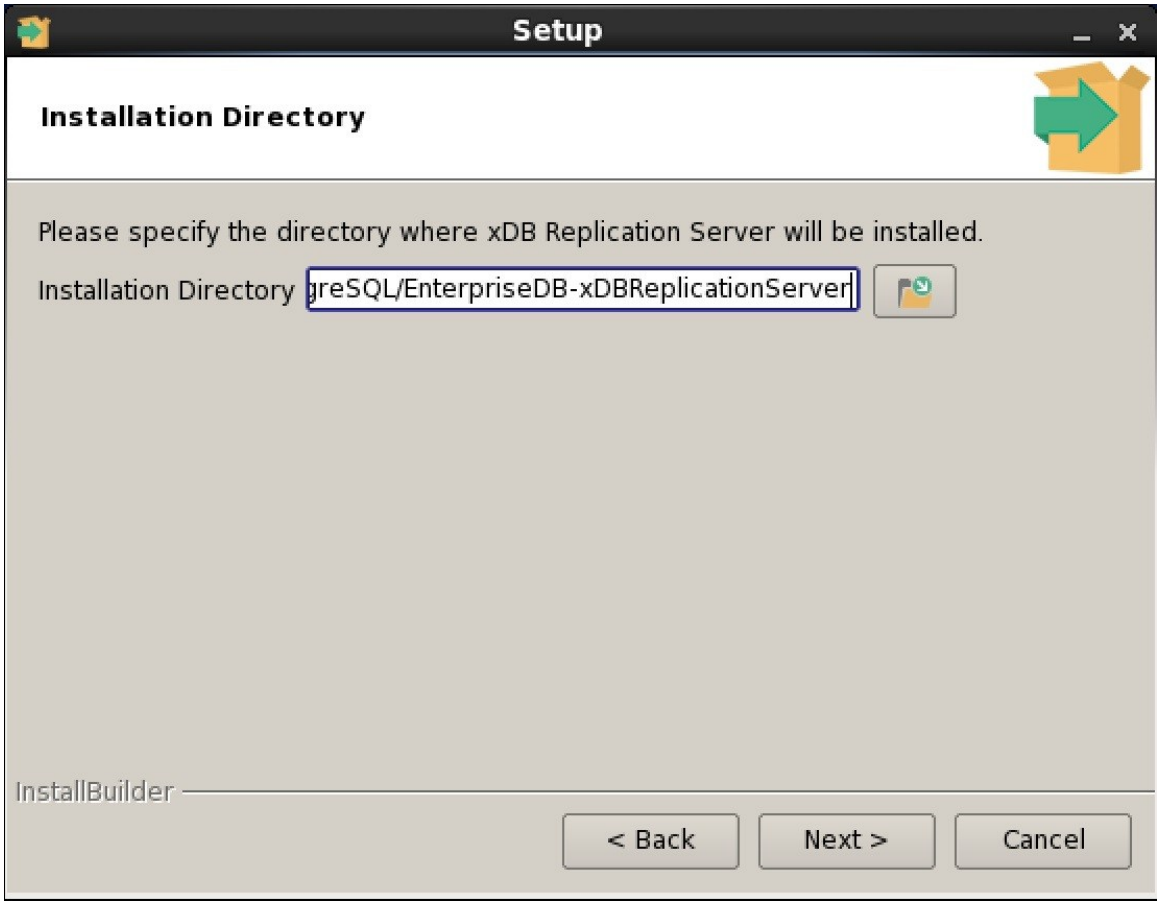


Figure 3-14 - Installation directory

Step 13: If you do not want a particular xDB Replication Server component installed on this particular host, uncheck the box next to the component name. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

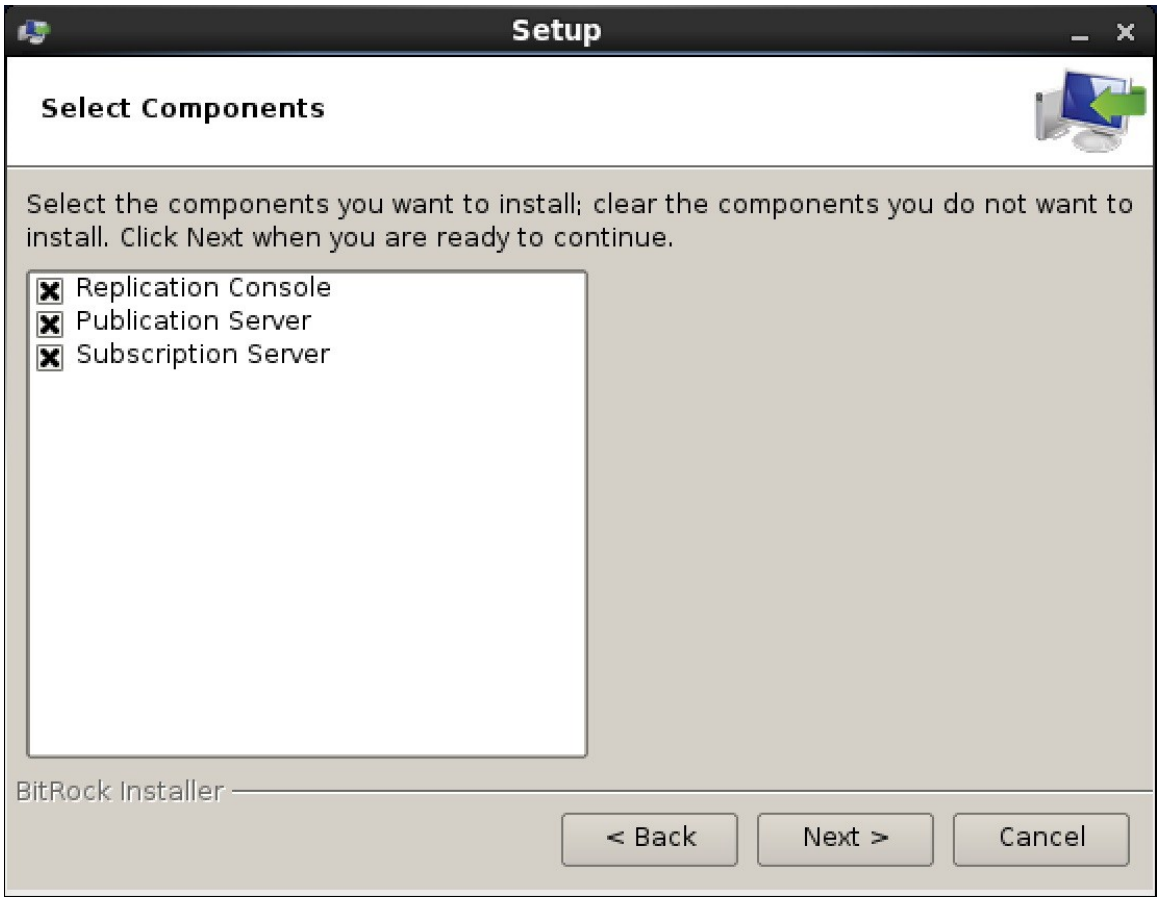


Figure 3-15 - Select components

Step 14: In the Account Registration screen select the radio button that applies to you. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

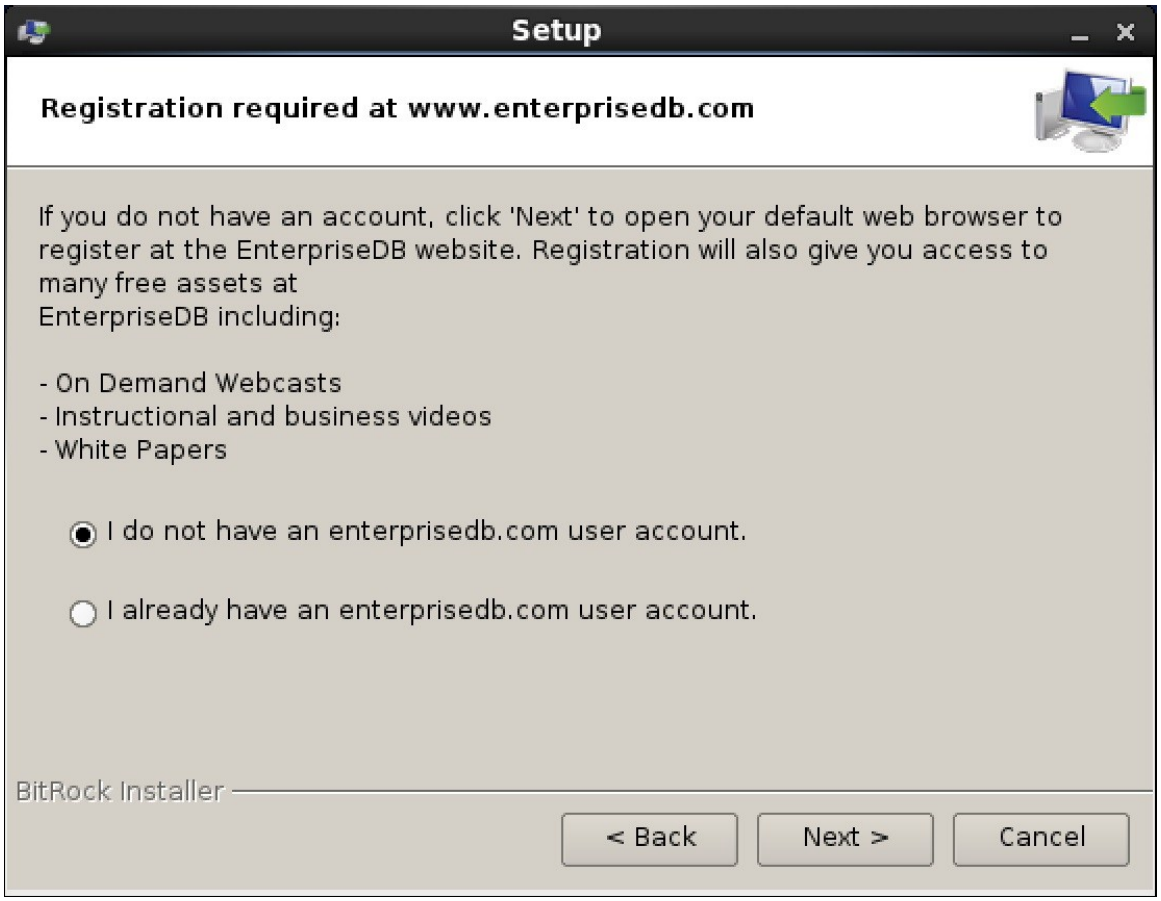


Figure 3-16 - Account registration

If you do not have an EnterpriseDB user account, you will be directed to the registration page of the EnterpriseDB website.

If you already have an EnterpriseDB user account, enter the email address and password for your EnterpriseDB user account as shown in the following screen. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

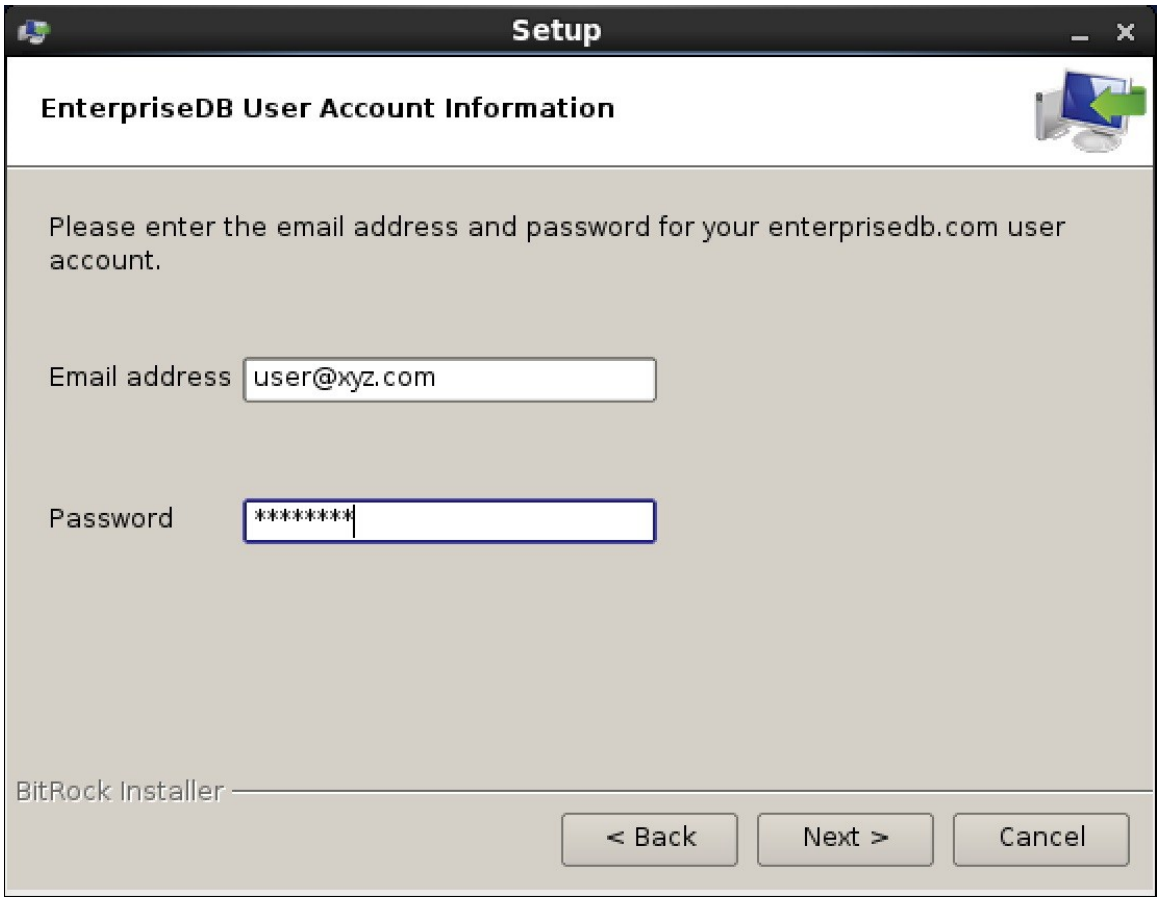


Figure 3-17 - My EnterpriseDB account

Step 15: Enter information for the xDB administrator.

Note: From this point on, it is suggested that you record the values you enter on these screens as they will be needed during the publication and subscription server registration process.

Enter values for the following fields:

- **Admin User.** The xDB administrator user name to authenticate certain usage of the xDB Replication Server such as registering a publication server or a subscription server running on this host. Any alphanumeric string may be entered for the admin user name. The default admin user name is `admin`.
- **Admin Password.** Password of your choice for the xDB administrator given in the Admin User field.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

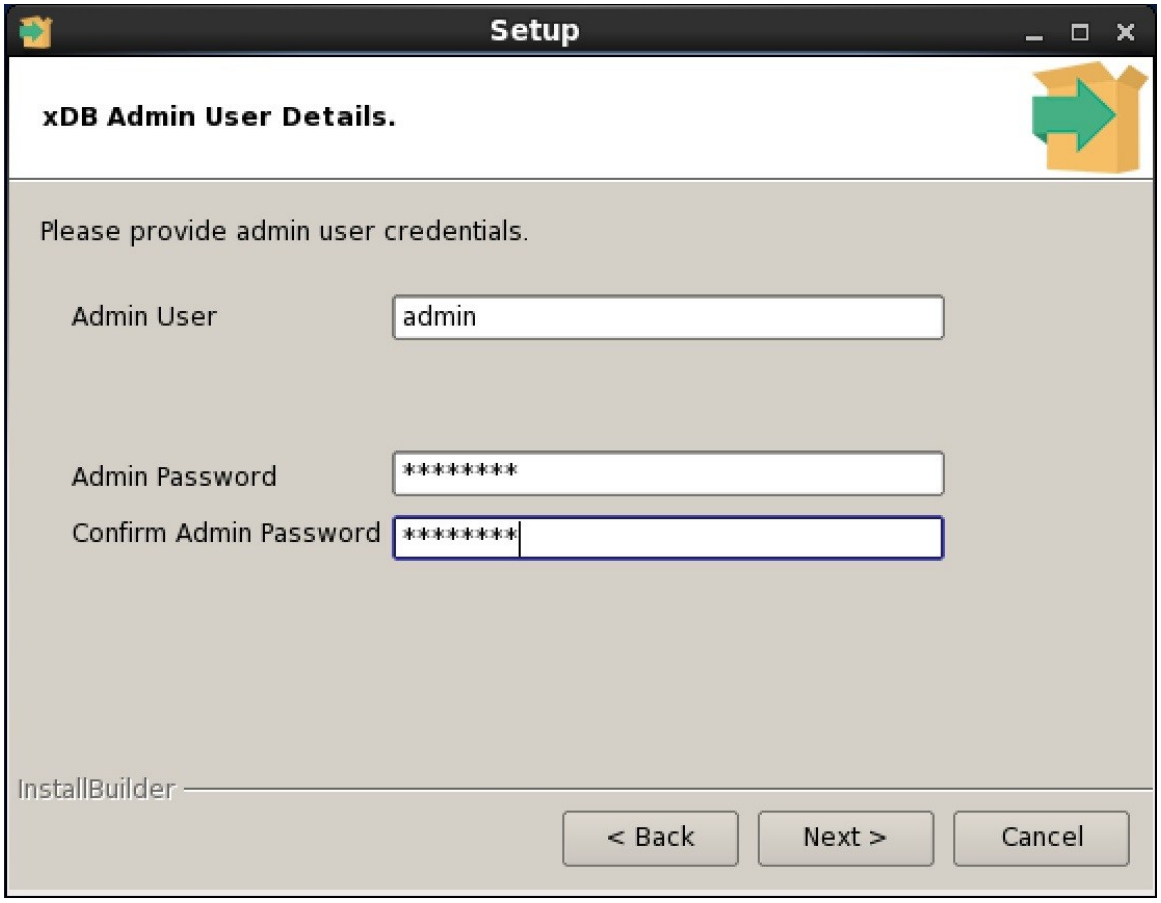


Figure 3-18 - xDB admin user information

The admin user and the admin password (in encrypted form) are saved to the xDB Replication Configuration file named `/etc/edb-repl.conf` (*`XDB_HOME`*`\etc\edb-repl.conf` on Windows hosts). Click the Next button.

Step 16 (Only if publication server is a selected component): Enter an available port on which the publication server will run. Default port number is `9051`. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

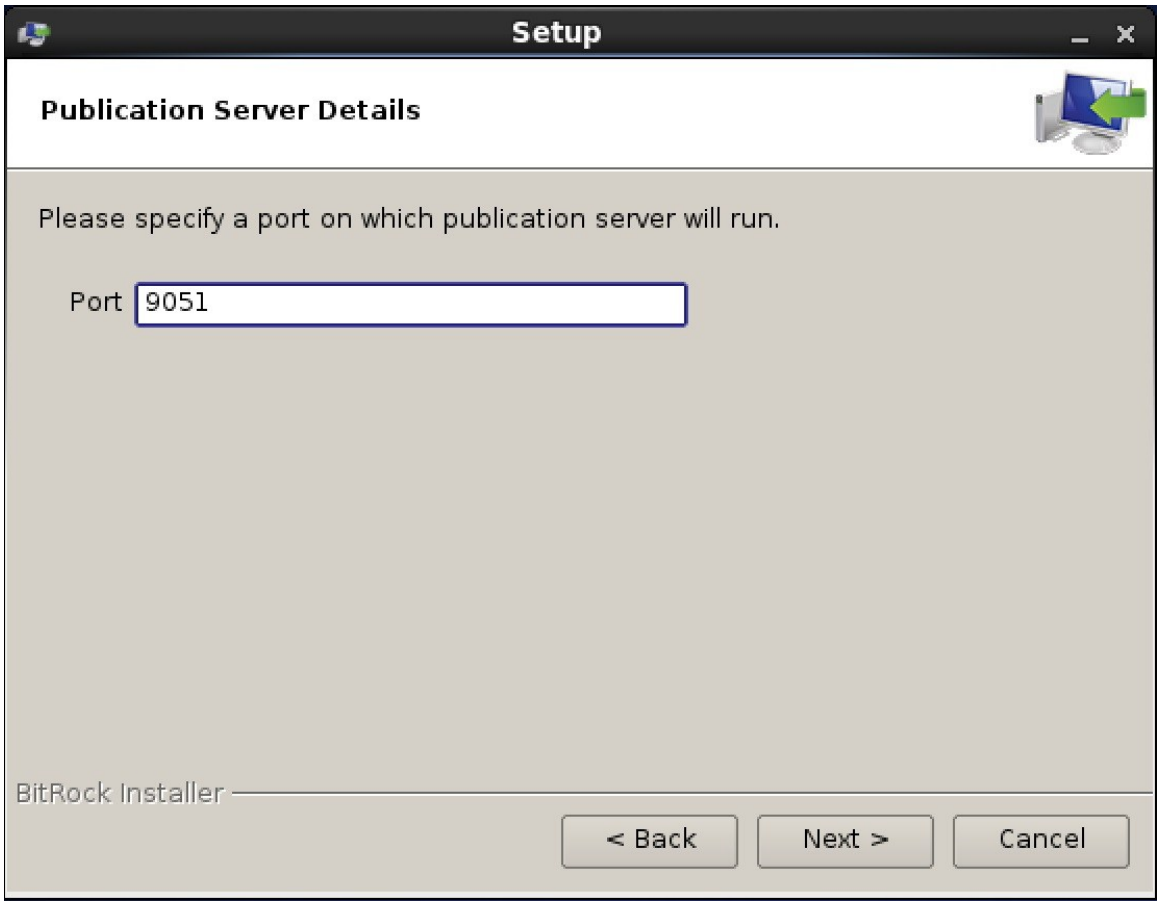


Figure 3-19 - Publication server details

Step 17 (Only if subscription server is a selected component): Enter an available port on which the subscription server will run. Default port number is `9052`. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

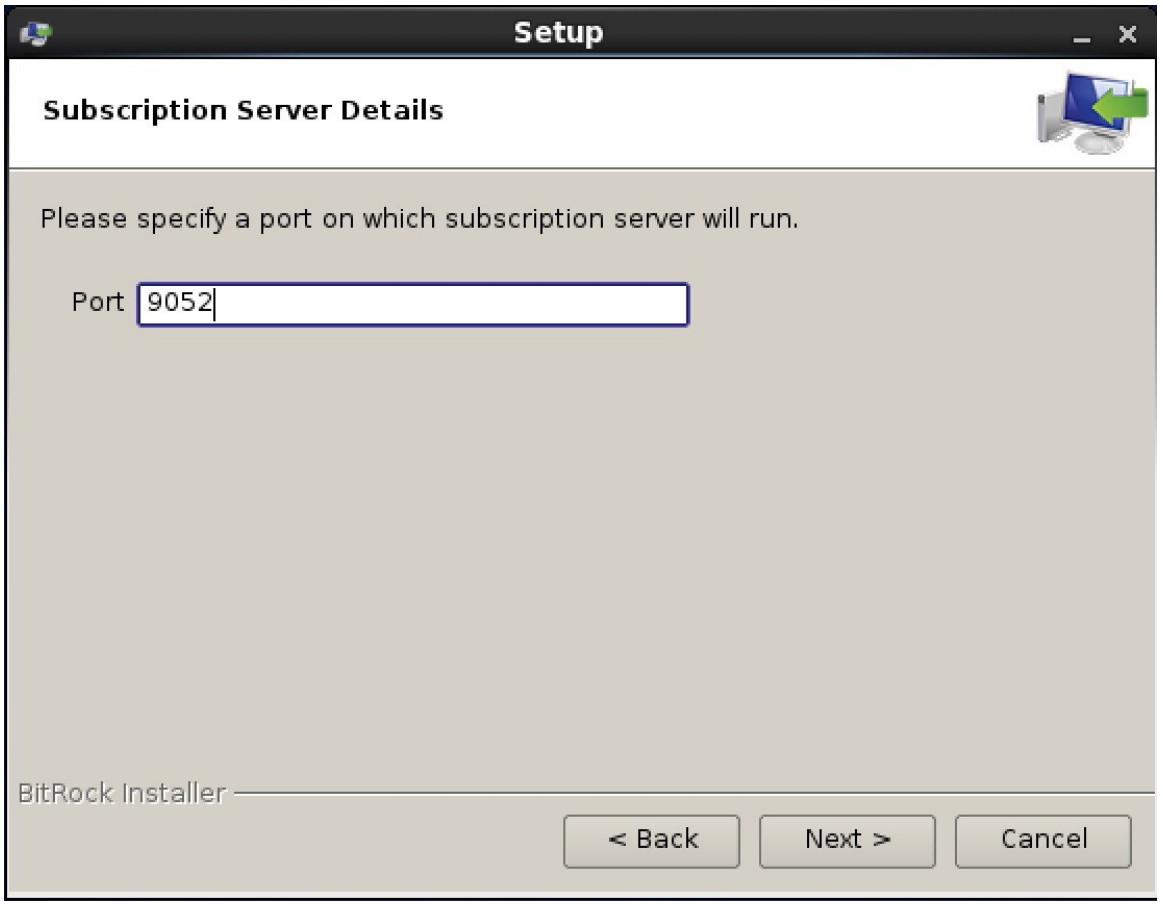


Figure 3-20 - Subscription server details

Step 18: For the operating system account under which the publication server or subscription server is to run, enter `postgres` (`enterprisedb` if you are using Advanced Server installed in Oracle compatible configuration mode).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

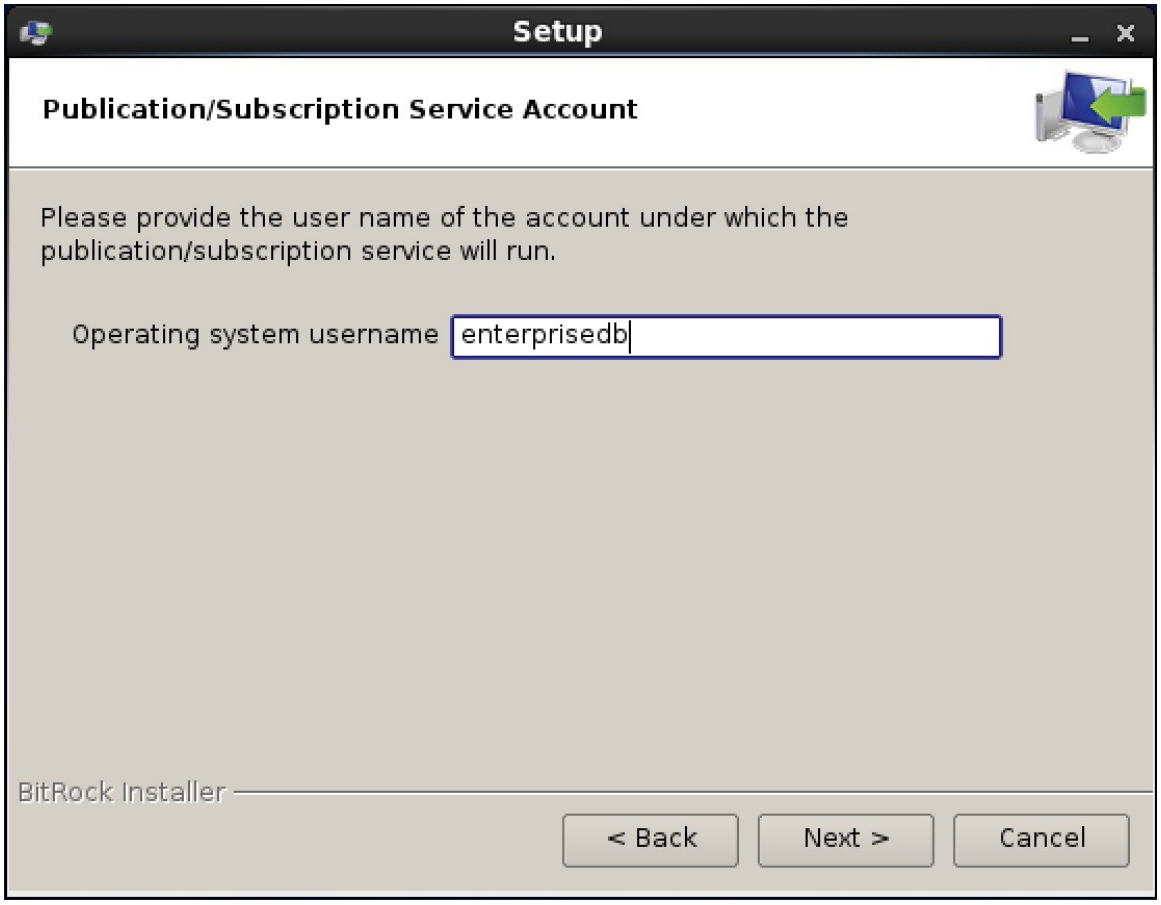


Figure 3-21 - Publication/subscription server operating system account

Step 19: On the Ready to Install screen, click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

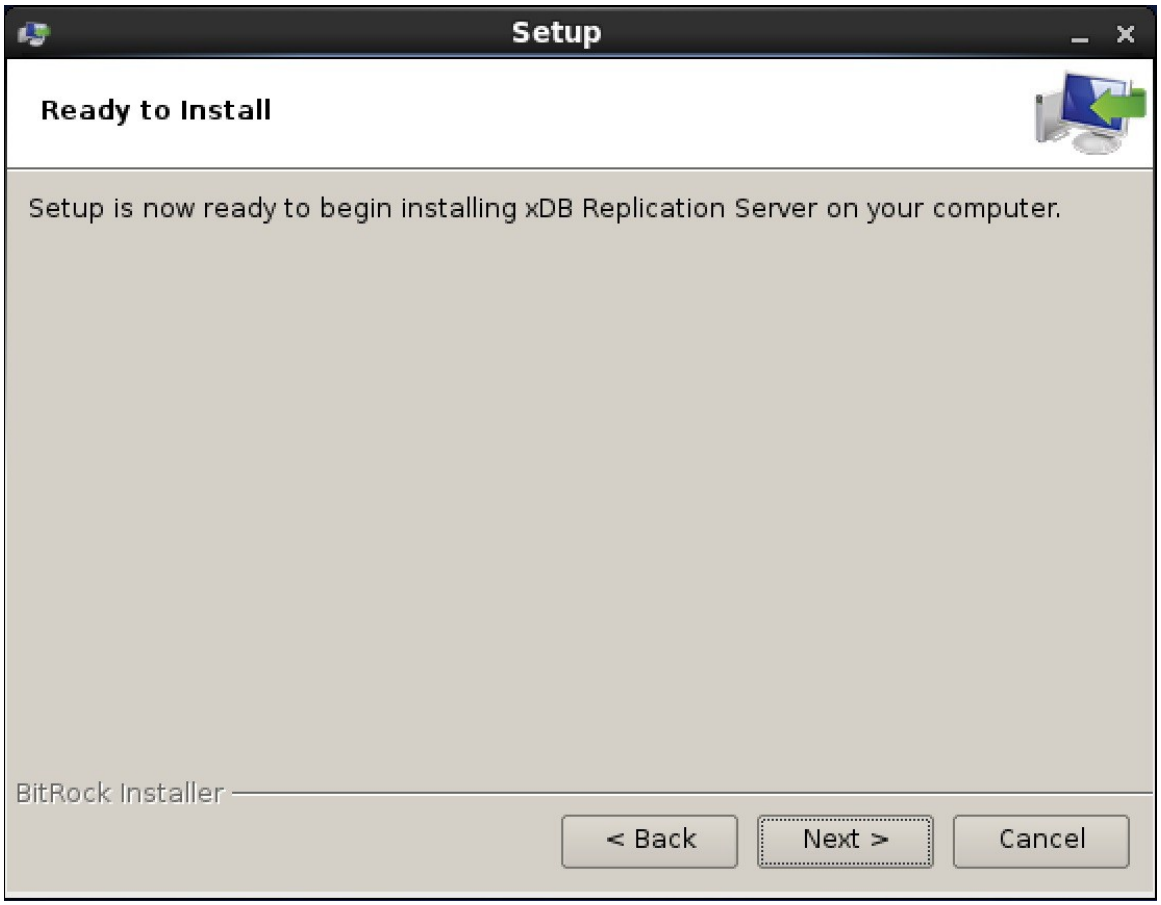


Figure 3-22 - Ready to install

An information box appears showing the installation progress of the xDB Replication Server selected components. This may take a few minutes.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

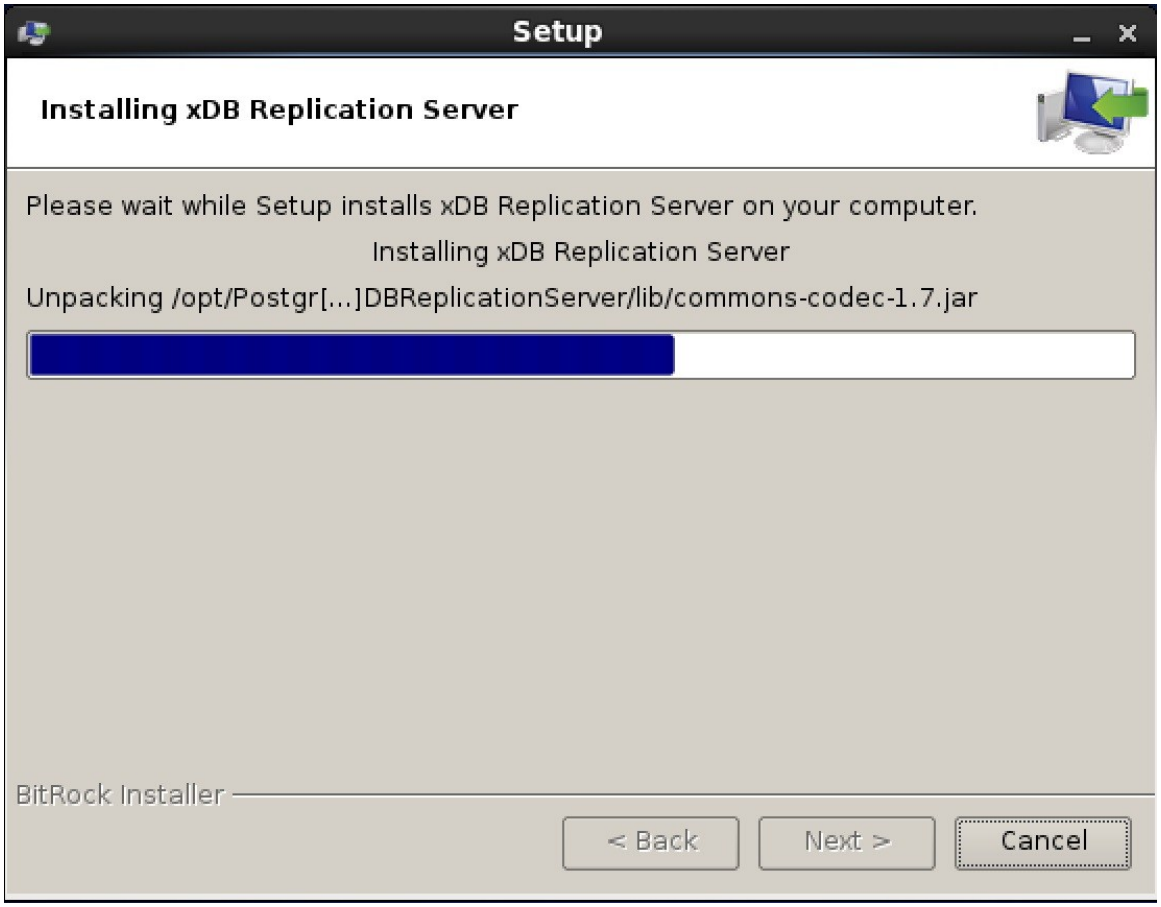


Figure 3-23 - Installation progress

Step 20: When installation has completed the following screen appears. Click the Finish button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Figure 3-24 – xDB Replication Server installation completion

Step 21: On the StackBuilder Plus Installation Complete screen, click the Finish button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

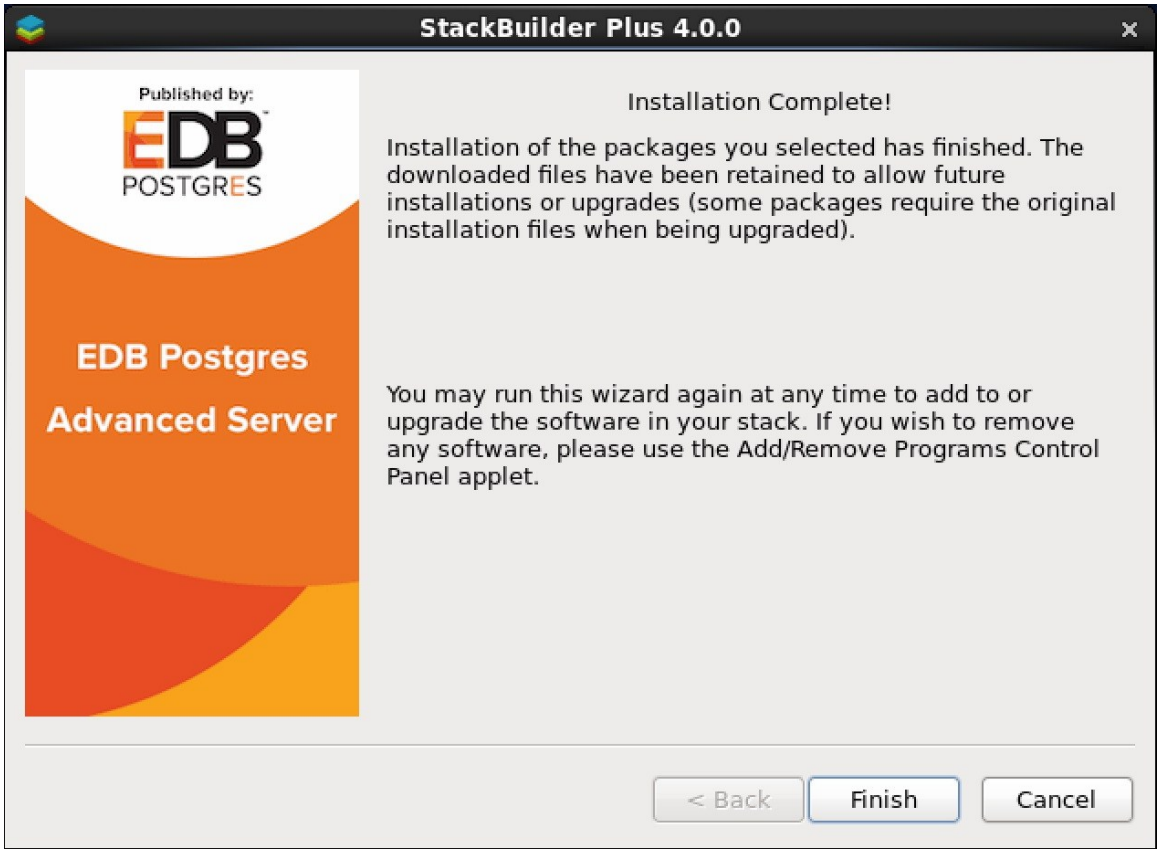


Figure 3-25 - StackBuilder Plus installation complete

Successful installation of xDB Replication Server results in the creation of directory structures and files in your host environment as described in Section 3.5.

3.2 Installing from the Command Line

The section provides directions for installing xDB Replication Server from the Linux or Windows command line console.

There are basically three ways of performing command line installation.

- **Text.** Include the `--mode text` parameter when invoking the installer to perform an installation from the command line during which you are prompted for user input.
- **Unattended.** Include the `--mode unattended` parameter when invoking the installer to perform an installation without user input. In this case, required parameters must be specified on the command line when invoking the installer or the `--optionfile` parameter must be used to specify a file containing the parameter settings.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- **Extract Only.** Invoke the installer with the `--extract-only` parameter to only extract the files when you do not hold the `root` privileges required to perform a complete installation.

The xDB Replication Server installer program can either be downloaded directly from the EnterpriseDB website or by using Stack Builder or StackBuilder Plus.

The installer program name may vary depending upon how you obtained it. The following are some examples illustrating command line installation.

Note: For additional detailed information on how to install EnterpriseDB products from the command line, see the *EDB Postgres Advanced Server Installation Guide* located at:

https://www.enterprisedb.com/resources/product-documentation

Note: You must have Java Runtime Environment (JRE) version 1.7 or later installed on the hosts where you intend to install any xDB Replication Server component (xDB Replication Console, publication server, or subscription server). Any Java product such as Oracle Java or OpenJDK may be used.

Follow the directions for your host operating system to install Java runtime.

Note: For Advanced Server versions prior to 9.3, a Java runtime is supplied and installed as part of the Advanced Server installation process, however, you must still have pre-installed a separate Java runtime system on your host. The xDB Replication Server installation process does not utilize the Java runtime supplied with Advanced Server.

The following example shows how to start the xDB Replication Server installation in text mode.

```
$ ./xdbreplicationserver-6.1.0-alpha-1-linux-x64.run --mode text
Language Selection

Please select the installation language
[1] English - English
    .
    .
    .
Please choose an option [1] :
------------------------------------------------------------------------------
Welcome to the Postgres Plus xDB Replication Server Setup Wizard.
    .
    .
    .
```

The following example shows how to start the installation in unattended mode with an options file.

```
$ su root
Password:
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ ./xdbreplicationserver-6.1.0-alpha-1-linux-x64.run --optionfile
/home/user/xdb_config
```

The following is the content of the options file, `xdb_config`.

```
mode=unattended
existing-user=user@xyz.com
existing-password=password
installer-language=en
prefix=/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer
enable-components=repconsole,pubserver,subserver
admin_user=admin
admin_password=password
pubport=9051
subport=9052
serviceaccount=enterprisedb
servicepassword=password
```

The following is a list of the parameters that may be specified when running the installer program. Most parameters have a default value if the parameter is not specified.

Parameters

`--help`

Display the list of valid options.

`--version`

Display the product version information.

`--extract-only { yes | no }`

Specify `yes` or `1` to extract the xDB Replication Server components and files without performing installation. Specify `no` or `0` to perform the installation of xDB Replication Server as well. The default is `no` or `0`.

`--unattendedmodeui { none | minimal | minimalWithDialogs }`

Specify the extent to which a user interface should be displayed during unattended installation. Specify `none` if no progress bars are to be displayed. Specify `minimal` if progress bars are to be displayed. Specify `minimalWithDialogs` if progress bars are to be displayed with dialog boxes if errors occur. The default is `minimal`.

`--optionfile` *`filename`*

Use the specified file containing installation configuration parameters in `parameter=`*`value`* format.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

`--mode { qt | gtk | xwindow | text | unattended }`

Specify the installation mode. Specify `qt` to use the Qt graphical toolkit. Specify `gtk` to use the Gtk graphical toolkit (for Linux only). Specify `xwindow` to use the X Windows graphical toolkit (for Linux only). Specify `text` for installation in a command line console (for Linux only). Specify `unattended` to perform installation without requesting user input. The default is `qt`.

`--debugtrace` *`debug_logfile`*

Specify this parameter to create a debug log file.

`--debuglevel { 0 | 1 | 2 | 3 | 4 }`

Specify the amount of detail to be written to the debug log file. Higher values provide more detail. The default level is `2`.

`--existing-user` *`edb_user_account`*

Specify your EnterpriseDB user account. (This is the email address used as your identifier when you created an account on the registration page of the EnterpriseDB website.)

`--existing-password` *`edb_user_password`*

Specify the password of your EnterpriseDB user account.

`--installer-language { en | zh_CN | zh_TW | ja | ko }`

Specify the installation language. Specify `en` for English. Specify `zh_CN` for Chinese Simplified. Specify `zh_TW` for Traditional Chinese. Specify `ja` for Japanese. Specify `ko` for Korean. The default is `en`.

`--prefix` *`installation_directory`*

The directory where the xDB Replication Server components are to be installed. The default is `/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer` for Linux systems. The default is `C:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer` for Windows systems.

`--enable-components {[ repconsole ] [,pubserver ] [,subserver ]}`

Specify the xDB Replication Server components to be installed. Specify `repconsole` for the xDB Replication Console and the xDB Replication Server Command Line Interface. Specify `pubserver` for the xDB publication server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Specify `subserver` for the xDB subscription server. At least one component must be included in this comma-separated list. The default is `repconsole,pubserver,subserver`.

`--disable-components {[ repconsole ] [,pubserver ] [,subserver ]}`

Specify the xDB Replication Server components to exclude from installation. The default is an empty list.

`--admin_user` *`admin_user`*

The xDB administrator user name to authenticate certain usage of the xDB Replication Server such as registering a publication server or a subscription server running on this host. Any alphanumeric string may be entered for the admin user name. The default admin user name is `admin`.

`--admin_password` *`admin_password`*

Password of your choice for the xDB administrator. There is no default for this parameter.

`--pubport` *`port`*

Port number for the publication server. The default is `9051`.

`--subport` *`port`*

Port number for the subscription server. The default is `9052`.

`--serviceaccount` *`account_name`*

The operating system account under which the publication server or subscription server is to run. The default is `postgres`.

`--servicepassword` *`account_password`*

The password for the operating system account. There is no default for this parameter.

Successful installation of xDB Replication Server results in the creation of directory structures and files in your host environment as described in Section 3.5.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3.3 Installing the xDB RPM Package

xDB Replication Server is also supplied as an RPM package available in the EDB Yum Repository. Use the Yum package manager to install xDB Replication Server from the RPM package.

For information about using Yum, see the Yum project website located at:

http://yum.baseurl.org/

To request credentials to the EDB Yum Repository, visit the following website:

https://www.enterprisedb.com/repository-access-request

For information about using the EDB Yum Repository see Chapter 3 of the *EDB Postgres Advanced Server Installation Guide* available from the EnterpriseDB website located at:

https://www.enterprisedb.com/resources/product-documentation

Note: Although the following primarily describes the installation of xDB Replication Server version 6.2, access to the RPM packages for prior xDB Replication Server versions are also described in order to differentiate the installation of these different versions.

Each xDB Replication Server component is available as an individual RPM package. Thus, you can install all xDB Replication Server components with a single `yum install` command, or you may choose to install selected, individual components by installing only those particular RPM packages.

The following table lists the RPM packages and the xDB Replication Server component it contains.

Table 3-1 - xDB Replication Server Component RPM Packages

Package Name	xDB Replication Server Component
`ppas-xdb`	All components
`ppas-xdb-console`	xDB Replication Console and the xDB Replication Server Command Line Interface
`ppas-xdb-publisher`	Publication server
`ppas-xdb-subscriber`	Subscription server
`ppas-xdb-libs`	Library files required by all components

The Advanced Server server libs package must be available for access by Yum when installing any xDB RPM package component. The `edb-as`*xx*`-server-libs` package is a component of the Advanced Server repository package for version 9.6 or later. The `ppas`*xx*`-server-libs` package is a component of the Advanced Server repository

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

package for version 9.5 or earlier. Step 3 shows how to enable access to the Advanced Server repository so Yum can access its server libs package.

To install any of the packages, invoke the following command as the `root` account:

```
yum install package_name
```

package_name is any of the packages listed under the Package Name column of the preceding table.

For example to install all xDB components, invoke the following:

```
yum install ppas-xdb
```

To install only the xDB Replication Console and xDB Replication Server Command Line Interface, invoke the following:

```
yum install ppas-xdb-console
```

To install only the publication server, invoke the following:

```
yum install ppas-xdb-publisher
```

Note: Though all xDB components are dependent upon and thus require installation of the server libs package, by using Yum, the dependency on the server libs is recognized when any xDB component is installed. Yum automatically installs the server libs package from the enabled Advanced Server repository along with your selected xDB RPM package.

The following are the steps to perform a complete xDB Replication Server installation with all xDB components.

Step 1: You must have Java Runtime Environment (JRE) version 1.7 or later installed on the hosts where you intend to install any xDB Replication Server component (xDB Replication Console, publication server, or subscription server). Any Java product such as Oracle Java or OpenJDK may be used.

Follow the directions for your host operating system to install Java runtime.

Note: For Advanced Server versions prior to 9.3, a Java runtime is supplied and installed as part of the Advanced Server installation process, however, you must still have pre-installed a separate Java runtime system on your host. The xDB Replication Server installation process does not utilize the Java runtime supplied with Advanced Server.

Step 2: From the EDB Yum Repository, click on the `edb-repo` link to download the repository RPM for all EnterpriseDB RPMs.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

As the `root` account, issue the following command to install this repository configuration package:

```
[root@localhost ~]# rpm -ivh edb-repo-latest.noarch.rpm
Preparing...                ########################################### [100%]
   1:edb-repo               ########################################### [100%]
```

Step 3: In directory `/etc/yum.repos.d`, the repository configuration file `edb.repo` is created, which is a text file containing a list of EnterpriseDB repositories, each denoted by an entry starting with the text `[repository_name]`.

Access to the packages in any of these repositories is accomplished by enabling the repository by editing the following in the repository entry:

- Using your requested credentials for the EDB Yum Repository, substitute the user name and password for the `<username>:<password>` placeholders of the `baseurl` parameter.
- Change the setting of the `enabled` parameter to `enabled=1`.

For example, to access the server libs package from the repository for Advanced Server version 9.6, enable the following entry:

```
[edbas96]
name=EnterpriseDB Advanced Server 9.6 $releasever - $basearch
baseurl=http://<username>:<password>@yum.enterprisedb.com/9.6/redhat/rhel-$releasever-
$basearch
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

To enable the appropriate repository for installing the desired version of xDB Replication Server, enable one of the following entries:

- To install version 6.2, enable the entry for `[enterprisedb-xdb60]`.
- To install a prior version, enable the entry for `[enterprisedb-tools]`.

Whichever version is chosen, be sure the other entries are disabled (that is, the parameter setting is `enabled=0` for the non-selected entries).

For example, to access xDB Replication Server version 6.2, enable the following entry:

```
[enterprisedb-xdb60]
name=EnterpriseDB XDB 6.0 $releasever - $basearch
baseurl=http://<username>:<password>@yum.enterprisedb.com/xdb60/redhat/rhel-
$releasever-$basearch
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Step 4: Install the xDB Replication Server RPM package.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following syntax installs the xDB RPM package files that have been previously downloaded to the current working directory. The local package files are installed by referencing the package name appended with an asterisk suffix (*):

```
yum install ppas-xdb*
```

To install the xDB RPM package directly from the EDB Yum Repository website, omit the asterisk suffix:

```
yum install ppas-xdb
```

The following is an example:

```
[root@localhost ~]# yum install ppas-xdb
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirrors.lga7.us.voxel.net
 * epel: mirror.steadfast.net
 * extras: mirror.vcu.edu
 * updates: mirror.vtti.vt.edu
edb-repos
| 2.4 kB     00:00
edbas10
| 2.5 kB     00:00
enterprisedb-dependencies
| 2.5 kB     00:00
enterprisedb-xdb60
| 2.4 kB     00:00
enterprisedb-xdb60/primary_db
| 4.2 kB     00:00
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package ppas-xdb.x86_64 0:6.2.0-1.rhel6 will be installed
--> Processing Dependency: ppas-xdb-subscriber for package: ppas-xdb-6.2.0-
1.rhel6.x86_64
--> Processing Dependency: ppas-xdb-publisher for package: ppas-xdb-6.2.0-
1.rhel6.x86_64
--> Processing Dependency: ppas-xdb-console for package: ppas-xdb-6.2.0-1.rhel6.x86_64
--> Running transaction check
---> Package ppas-xdb-console.x86_64 0:6.2.0-1.rhel6 will be installed
--> Processing Dependency: ppas-xdb-libs for package: ppas-xdb-console-6.2.0-
1.rhel6.x86_64
---> Package ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6 will be installed
--> Processing Dependency: ppas-libs for package: ppas-xdb-publisher-6.2.0-
1.rhel6.x86_64
--> Processing Dependency: libpq.so.5()(64bit) for package: ppas-xdb-publisher-6.2.0-
1.rhel6.x86_64
---> Package ppas-xdb-subscriber.x86_64 0:6.2.0-1.rhel6 will be installed
--> Running transaction check
---> Package edb-as10-server-libs.x86_64 0:10.2.7-2.rhel6 will be installed
---> Package ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
 Package                                 Arch                          Version
Repository                              Size
================================================================================
================================================================================
Installing:
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
 ppas-xdb                                   x86_64                         6.2.0-1.rhel6
enterprisedb-xdb60                         6.8 k
Installing for dependencies:
 edb-as10-server-libs                       x86_64                         10.2.7-2.rhel6
edbas10                                    536 k
 ppas-xdb-console                           x86_64                         6.2.0-1.rhel6
enterprisedb-xdb60                         3.0 M
 ppas-xdb-libs                              x86_64                         6.2.0-1.rhel6
enterprisedb-xdb60                          13 M
 ppas-xdb-publisher                         x86_64                         6.2.0-1.rhel6
enterprisedb-xdb60                          35 k
 ppas-xdb-subscriber                        x86_64                         6.2.0-1.rhel6
enterprisedb-xdb60                          12 k

Transaction Summary
================================================================================
================================================================================
Install       6 Package(s)

Total download size: 17 M
Installed size: 20 M
Is this ok [y/N]: y
Downloading Packages:
(1/6): edb-as10-server-libs-10.2.7-2.rhel6.x86_64.rpm
| 536 kB     00:09
(2/6): ppas-xdb-6.2.0-1.rhel6.x86_64.rpm
| 6.8 kB     00:00
(3/6): ppas-xdb-console-6.2.0-1.rhel6.x86_64.rpm
| 3.0 MB     00:01
(4/6): ppas-xdb-libs-6.2.0-1.rhel6.x86_64.rpm
|  13 MB     00:01
(5/6): ppas-xdb-publisher-6.2.0-1.rhel6.x86_64.rpm
|  35 kB     00:00
(6/6): ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64.rpm
|  12 kB     00:00
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total
1.2 MB/s |  17 MB     00:13
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
1/6
  Installing : ppas-xdb-console-6.2.0-1.rhel6.x86_64
2/6
  Installing : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
3/6
  Installing : edb-as10-server-libs-10.2.7-2.rhel6.x86_64
4/6
  Installing : ppas-xdb-publisher-6.2.0-1.rhel6.x86_64
5/6
  Installing : ppas-xdb-6.2.0-1.rhel6.x86_64
6/6
  Verifying  : ppas-xdb-6.2.0-1.rhel6.x86_64
1/6
  Verifying  : ppas-xdb-publisher-6.2.0-1.rhel6.x86 64
2/6
  Verifying  : ppas-xdb-console-6.2.0-1.rhel6.x86_64
3/6
  Verifying  : edb-as10-server-libs-10.2.7-2.rhel6.x86_64
4/6
  Verifying  : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
5/6
  Verifying  : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
6/6

Installed:
  ppas-xdb.x86_64 0:6.2.0-1.rhel6
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Dependency Installed:
  edb-as10-server-libs.x86_64 0:10.2.7-2.rhel6            ppas-xdb-console.x86_64
0:6.2.0-1.rhel6              ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6
  ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6              ppas-xdb-subscriber.x86_64
0:6.2.0-1.rhel6

Complete!
```

The xDB Replication Server is installed in directory location `/usr/ppas-xdb-x.x` where `x.x` is the xDB Replication Server version number as shown by the following:

```
[root@localhost ppas-xdb-6.2]# pwd
/usr/ppas-xdb-6.2
[root@localhost ppas-xdb-6.2]# ls -l
total 84
drwxr-xr-x 2 root         root          4096 Feb 23 16:05 bin
drwxr-xr-x 3 root         root          4096 Feb 23 16:05 etc
drwxr-xr-x 4 root         root          4096 Feb 23 16:05 lib
drwxr-xr-x 2 root         root          4096 Feb 23 16:05 share
drwx------ 2 enterprisedb enterprisedb  4096 Feb 20 22:17 xdata
-r--r--r-- 1 enterprisedb enterprisedb 64035 Feb 20 20:40 xdb_3rd_party_licenses.txt
```

Successful installation of xDB Replication Server results in the creation of directory structures and files in your host environment as described in Section 3.5.

Note: Neither the publication server nor the subscription server are running immediately following installation. If after reviewing the remaining steps, you wish to start the publication server, see Section 5.2.1. For starting the subscription server see Section 5.3.1.

Step 5 (For xDB Replication Server 6.2 or 6.1): In the xDB Replication Configuration file `/etc/edb-repl.conf`, you can either use the default password (`edb`) as the admin user password, or you can substitute a password of your choice. If you want to use your own password, see Section 10.4.2 on how to generate the encrypted form of the password. Place the encrypted password in the `admin_password` parameter of the xDB Replication Configuration file. The default admin user name is set to `admin` and can be changed as well. See Section 2.3.1.3 for information on the xDB Replication Configuration file.

Step 5 (For xDB Replication Server 5.1): In the xDB Replication Configuration file `/etc/edb-repl.conf`, verify that parameters `host`, `port`, `database`, `user`, and `password` are set to allow access to a Postgres database that you wish to use as the xDB Control database. If you wish to use a database other than the one identified by the current default settings, create the desired database and change the parameters to permit connection and authentication to this database to be used as the xDB Control database.

Step 6: The `JAVA_EXECUTABLE_PATH` parameter in the xDB Startup Configuration file should be set so that the Java runtime program can be accessed upon startup of the publication server and subscription server. If the publication server or subscription server startup fails due to inaccessibility to the Java program, be sure to set the path to your Java runtime program in the xDB Startup Configuration file. See Section 2.3.1.4 for

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

information on the xDB Startup Configuration file. See Section 3.5 for the location of this file.

3.4 Installing xDB on an SLES 12 Host

You can use the zypper package manager to install the xDB Replication Server on an SLES 12 host. zypper will attempt to satisfy package dependencies as it installs a package, but requires access to specific repositories that are not hosted at EnterpriseDB.

Before installing xDB, use the following commands to add EnterpriseDB repository configuration files to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/epas96-
sles.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-
tools.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-
dependencies.repo
```

Each command creates a repository configuration file in the `/etc/zypp/repos.d` directory. The files are named:

- `edbas96suse.repo`
- `edbasdependencies.repo`
- `edbastools.repo`

After creating the repository configuration files, use the `zypper refresh` command to refresh the metadata on your SLES host to include the EnterpriseDB repositories:

```
/etc/zypp/repos.d # zypper refresh
Repository 'SLES12-12-0' is up to date.
Repository 'SLES12-Pool' is up to date.
Repository 'SLES12-Updates' is up to date.
Retrieving repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' metadata ----------------------[\]

Authentication required for
'https://zypp.enterprisedb.com/9.6/suse/suse-12-x86_64'
User Name:
Password:

Retrieving repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' metadata.................................[done]
Building repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' cache.......................[done]
All repositories have been refreshed.
...
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When prompted for a `User Name` and `Password`, provide your connection credentials for the EnterpriseDB repository. If you need credentials, visit the following website:

https://www.enterprisedb.com/repository-access-request

Before installing EDB Postgres Advanced Server or supporting components, you must also add SUSEConnect and the SUSE Package Hub extension to the SLES host, and register the host with SUSE, allowing access to SUSE repositories. Use the commands:

```
zypper install SUSEConnect
SUSEConnect -p PackageHub/12/x86_64
SUSEConnect -p sle-sdk/12/x86_64
```

For detailed information about registering a SUSE host, visit:

https://www.suse.com/support/kb/doc/?id=7016626

Then add the Java repository and update the repository metadata:

Please note that only OpenJDK (version 1.8) is supported on SLES hosts of Java components. Before using an RPM installer to add Advanced Server or a supporting component that requires Java, use zypper to add supporting repository locations to your system.

Use the commands:

```
zypper addrepo
"http://download.opensuse.org/repositories/Java:/Factory/SLE_12_S
P2/Java:Factory.repo"
zypper addrepo
"http://download.opensuse.org/repositories/server:/Kolab:/3.3/SLE
_12/server:Kolab:3.3.repo"

zypper refresh
```

Then, you can use the zypper utility to install the xDB Replication Server components:

```
zypper install ppas-xdb*
```

Note: Before starting the publication server and subscription server, the `/etc/hosts` file must contain an entry for the host name that associates it to the host IP address as shown by the following example where 192.168.187.133 is the IP address and `linux-dm8s` is the host name:

```
192.168.187.133              linux-dm8s
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3.5 Post-Installation Host Environment

On Linux hosts where you installed xDB Replication Server with the graphical user interface or from the command line, you should now have a publication server daemon and a subscription server daemon running on your computer assuming you chose to install the publication server and subscription server components. If you installed the xDB RPM package, you must start the publication server and the subscription server based upon the instructions in Section 5.2.1 for the publication server and Section 5.3.1 for the subscription server. On Windows systems, the publication server and subscription server run as services named `Publication Service` and `Subscription Service`.

The Postgres application menu contains a new item for the xDB Replication Console.

Note: On some Linux systems, you may have to restart the server before you can see the xDB Replication Console choice in the application menu. If the xDB Replication Console choice is still unavailable in the application menu, it can be started by invoking the script *XDB_HOME*`/bin/runRepConsole.sh`.

Note: For xDB Replication Server installed from an xDB RPM package, the xDB Replication Console is started by invoking the script *XDB_HOME*`/bin/runRepConsole.sh`.

The following files are created that you may need during the configuration process.

Table 3-2 – Post-Installation Files

File Name	Location	Description
`edb-repl.conf` (Linux)	`/etc`	xDB Replication Configuration file
`edb-repl.conf` (Windows)	*XDB_HOME*`\etc`	xDB Replication Configuration file
`edb-xdbpubserver` (Linux)	`/etc/init.d`	Start, stop, or restart the publication server
`edb-xdbpubserver.service` (Linux)	`/usr/lib/systemd/system`	Start, stop, or restart the publication server (CentOS 7, RHEL 7)
`edb-xdbsubserver` (Linux)	`/etc/init.d`	Start, stop, or restart the subscription server
`edb-xdbsubserver.service` (Linux)	`/usr/lib/systemd/system`	Start, stop, or restart the subscription server (CentOS 7, RHEL 7)
`xdb_pubserver.conf`	*XDB_HOME*`/etc`	Publication server configuration file
`xdb_subserver.conf`	*XDB_HOME*`/etc`	Subscription server configuration file
`xdbReplicationServer-`*xx*`.config`	*XDB_HOME*`/etc/sysconfig`	xDB Startup Configuration file
`pubserver.log` (Linux)	`/var/log/xdb-`*x.x*	Publication server log file
`pubserver.log` (Windows)	*POSTGRES_HOME*`\.enterprisedb\xdb\`*x.x*	Publication server log file
`subserver.log` (Linux)	`/var/log/xdb-`*x.x*	Subscription server log file
`subserver.log` (Windows)	*POSTGRES_HOME*`\.enterprisedb\xdb\`*x.x*	Subscription server log file

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

File Name	Location	Description
edb-xdbpubserver.log (Linux)	/var/log/edb/xdbpubserver	Publication services startup log file
edb-xdbsubserver.log (Linux)	/var/log/edb/xdbsubserver	Subscription services startup log file
servers.xml	*USER_HOME*/.enterprisedb/xdb/*x.x*	Server login file

Note: *XDB_HOME* is the directory where xDB Replication Server is installed.

Note: *POSTGRES_HOME* is the home directory of the postgres operating system account (enterprisedb for Advanced Server installed in Oracle compatible configuration mode).

Note: The publication and subscription services startup log files (edb-xdbpubserver.log and edb-xdbsubserver.log) are not generated for Windows and Mac OS X operating systems.

Note: *USER_HOME* is the home directory of the operating system account in use.

Note: The xDB Replication Server version number is represented by *x.x* or by *xx* (for example 6.2 or 62).

3.6 Uninstalling xDB Replication Server

Uninstalling xDB Replication Server results in the removal of the publication server, the subscription server, the xDB Replication Console, the xDB Replication Server Command Line Interface, the xDB Replication Configuration file, the xDB Startup Configuration file, the publication server configuration file, and the subscription server configuration file.

Uninstalling xDB Replication Server does not remove any databases used as master nodes, publication databases, or subscription databases.

Use the xDB Replication Console or the xDB Replication Server Command Line Interface to delete any existing single-master or multi-master replication systems before you uninstall xDB Replication Server, otherwise the control schema objects created in the publication databases or master nodes will remain in those databases. These control schema objects must then be deleted manually such as by using an SQL command line utility.

If you installed xDB Replication Server using the xDB Replication Server installer program invoked from Stack Builder or StackBuilder Plus as described in Section 3.1 or you invoked the xDB Replication Server installer program from the command line as described in Section 3.2, uninstall xDB Replication Server by invoking the uninstall-xdbreplicationserver script as described in this section.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If you installed xDB Replication Server from the RPM package, uninstall it using the Yum package manager. See Section 3.7 for information.

For Linux only: The following steps are for uninstalling xDB Replication Server from a Linux host.

Step 1: As the `root` account, run the *`XDB_HOME`*`/uninstall-xdbreplicationserver` script from the directory where you installed xDB Replication Server.

```
$ su root
Password:
$ cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer
$ ./uninstall-xdbreplicationserver
```

Step 2: Click the Yes button to confirm uninstallation of xDB Replication Server.

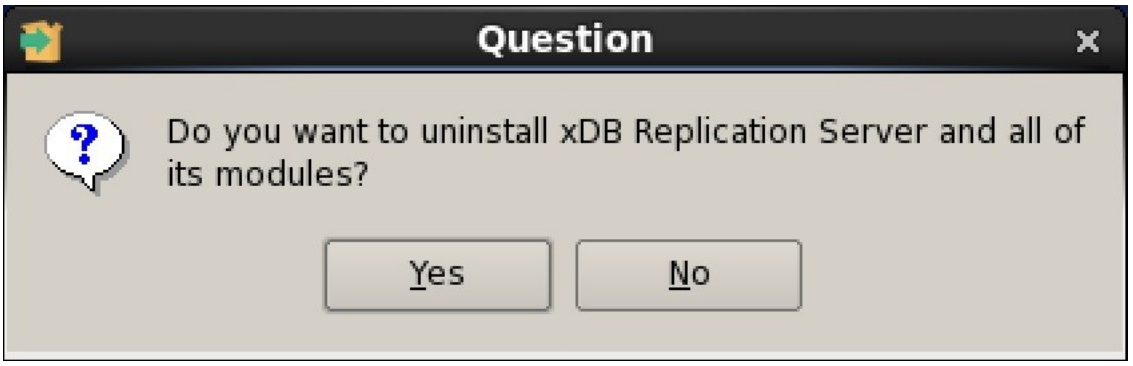


Figure 3-26 - Confirm xDB Replication Server uninstallation

Step 3: The Uninstallation Completed dialog box appears when the process has completed. Click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

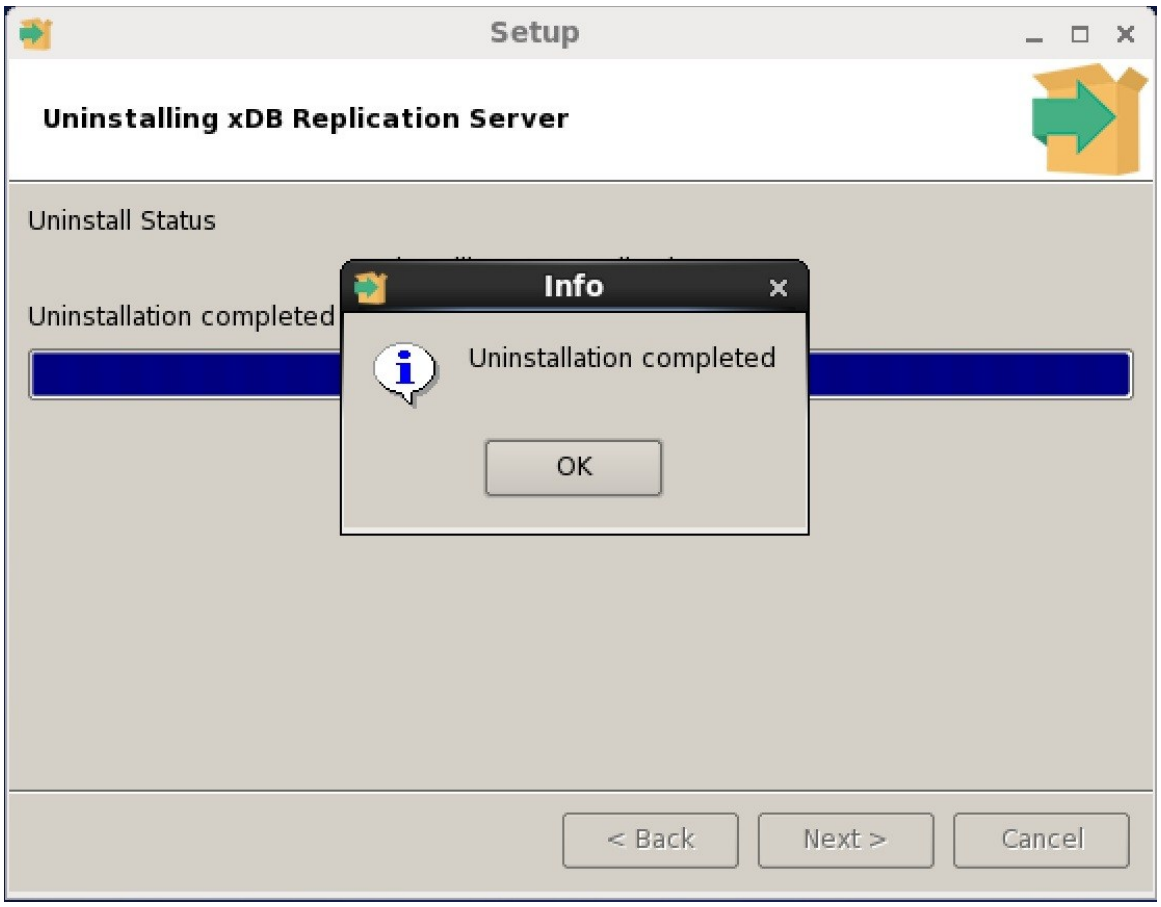


Figure 3-27 - Uninstallation completed

For Windows only: The following steps are for uninstalling xDB Replication Server from a Windows host.

Step 1: From the Windows Control Panel, select Uninstall a Program.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

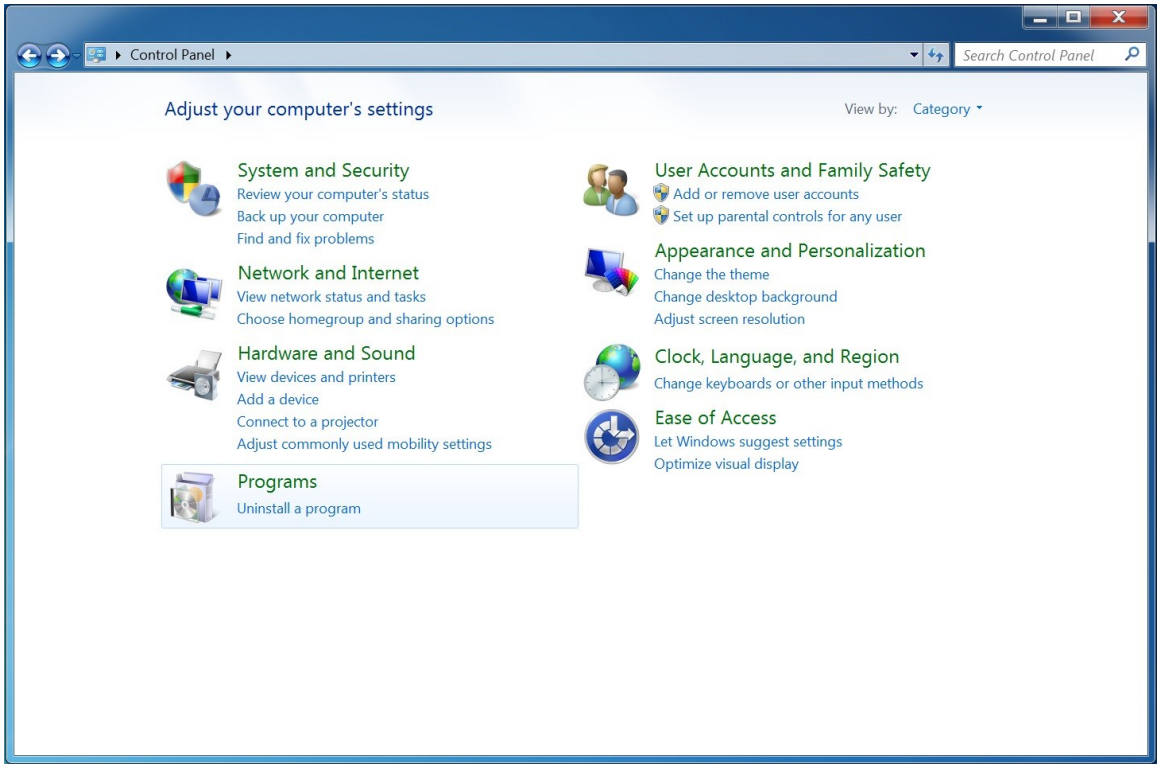


Figure 3-28 - Uninstall a program

Step 2: Select the xDB Replication Server product in the list of programs to uninstall or change. Click the Uninstall/Change button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

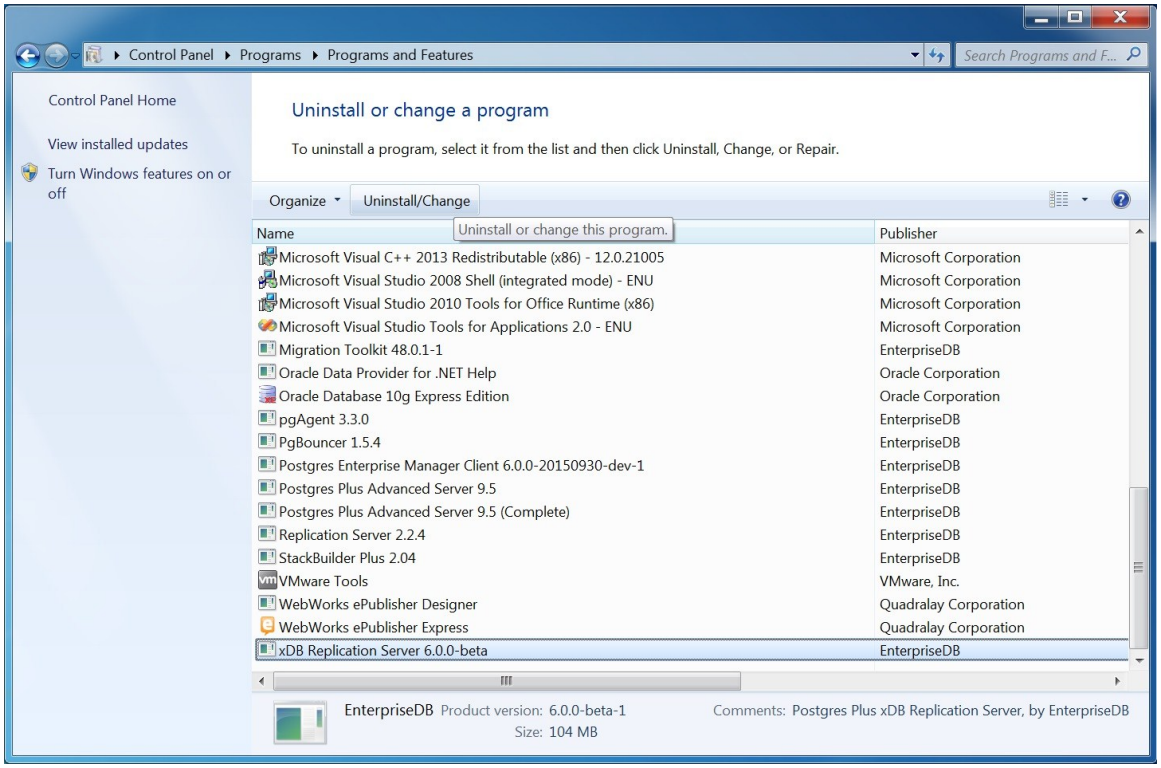


Figure 3-29 - Uninstall or change a program

Step 3: Click the Yes button to confirm uninstallation of xDB Replication Server.

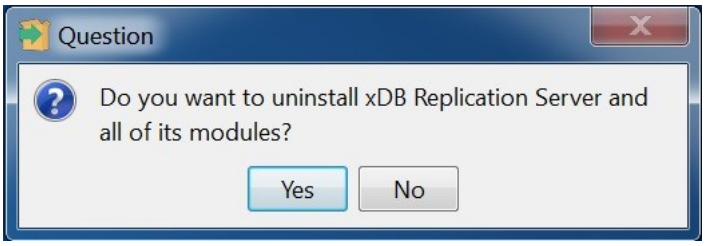


Figure 3-30 - Confirm xDB Replication Server uninstallation

Step 4: The Uninstallation Completed dialog box appears when the process has completed. Click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

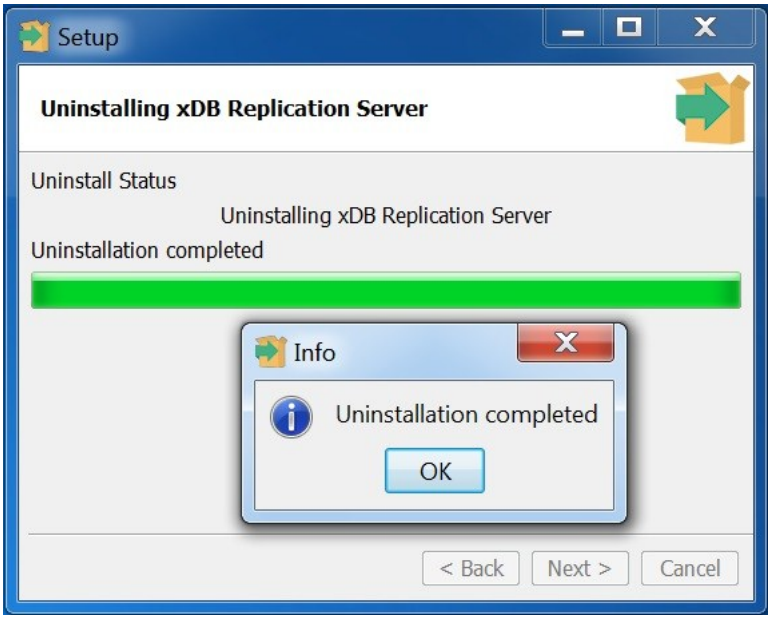


Figure 3-31 - Uninstallation completed

Uninstalling in Text or Unattended Mode

Uninstallation of xDB Replication Server can also be done without the use of the graphical user interface. This is illustrated by the following examples.

The following shows how to uninstall xDB Replication Server in text mode.

```
$ su root
Password:
$ ./uninstall-xdbreplicationserver --mode text
Do you want to uninstall xDB Replication Server and all of its modules?
[Y/n]: y

------------------------------------------------------------------------------
Uninstall Status

 Uninstalling xDB Replication Server
 0% _____________ 50% _____________ 100%
 ########################################

Info: Uninstallation completed
Press [Enter] to continue :
$
```

The following shows how to uninstall xDB Replication Server in unattended mode.

```
$ su root
Password:
$ ./uninstall-xdbreplicationserver --mode unattended
$
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3.7 Uninstalling the xDB RPM Package

If you installed xDB Replication Server from the RPM package, you can uninstall any xDB component by invoking the `yum remove package_name` command as the `root` account where *`package_name`* is any xDB Replication Server component RPM package as listed in the table in Section 3.3.

All xDB Replication Server components can be removed by the following command:

```
yum remove ppas-xdb*
```

An example is shown by the following:

```
[root@localhost ~]# yum remove ppas-xdb*
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package ppas-xdb.x86_64 0:6.2.0-1.rhel6 will be erased
---> Package ppas-xdb-console.x86_64 0:6.2.0-1.rhel6 will be erased
---> Package ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6 will be erased
---> Package ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6 will be erased
---> Package ppas-xdb-subscriber.x86_64 0:6.2.0-1.rhel6 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
 Package                                 Arch                          Version
Repository                               Size
================================================================================
================================================================================
Removing:
 ppas-xdb                                x86_64                        6.2.0-1.rhel6
@enterprisedb-xdb60                     0.0
 ppas-xdb-console                        x86_64                        6.2.0-1.rhel6
@enterprisedb-xdb60                     3.3 M
 ppas-xdb-libs                           x86_64                        6.2.0-1.rhel6
@enterprisedb-xdb60                      15 M
 ppas-xdb-publisher                      x86_64                        6.2.0-1.rhel6
@enterprisedb-xdb60                     126 k
 ppas-xdb-subscriber                     x86_64                        6.2.0-1.rhel6
@enterprisedb-xdb60                      19 k

Transaction Summary
================================================================================
================================================================================
Remove        5 Package(s)

Installed size: 18 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : ppas-xdb-6.2.0-1.rhel6.x86_64
1/5
  Erasing    : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
2/5
  Erasing    : ppas-xdb-console-6.2.0-1.rhel6.x86 64
3/5
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  Erasing    : ppas-xdb-publisher-6.2.0-1.rhel6.x86_64
4/5
  Erasing    : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
5/5
  Verifying  : ppas-xdb-6.2.0-1.rhel6.x86_64
1/5
  Verifying  : ppas-xdb-console-6.2.0-1.rhel6.x86_64
2/5
  Verifying  : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
3/5
  Verifying  : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
4/5
  Verifying  : ppas-xdb-publisher-6.2.0-1.rhel6.x86_64
5/5

Removed:
  ppas-xdb.x86_64 0:6.2.0-1.rhel6                        ppas-xdb-console.x86_64
0:6.2.0-1.rhel6                ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6
  ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6              ppas-xdb-subscriber.x86_64
0:6.2.0-1.rhel6

Complete!
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

4 Introduction to the xDB Replication Console

The xDB Replication Console is the graphical user interface that you use to configure and manage the replication system. The equivalent functionality can also be done using the xDB Replication Server CLI utility. See Chapter 8 for information on the xDB Replication Server CLI.

The xDB Replication Console window consists of the following main areas:

- **Menu Bar.** Menus for the replication system components
- **Tool Bar.** Icons for quick access to dialog boxes
- **Replication Tree.** Replication system components represented as nodes in an inverted tree
- **Information Window.** Tabbed window with information about a highlighted node in the replication tree

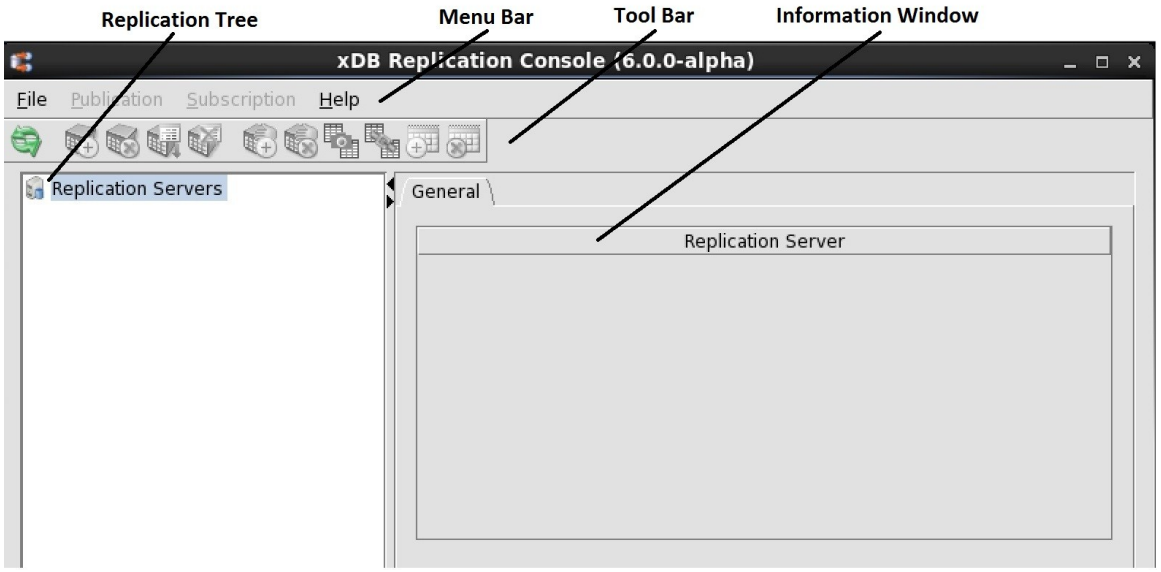


Figure 4-1 - xDB Replication Console window

The options that are available on the menu bar and tool bar are dependent upon the node highlighted in the replication tree. Only those options relevant to the highlighted node are available in the menu bar and tool bar.

The content of the information window applies to the highlighted node as well.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

4.1 xDB Replication Console Tool Bar

This section describes when the various tool bar icons are activated. The operations associated with the tool bar are described in sections 5.2 and 5.3 for single-master replication. For multi-master replication see Section 6.2.

Note: The publication server must be running in order to use tools relevant to publications. Similarly, the subscription server must be running in order to use tools relevant to subscriptions.

4.1.1 Refresh

The Refresh icon is always activated. Click the Refresh icon if the replication tree or information window does not appear to display the latest information after performing an operation. Clicking the Refresh icon ensures that the latest information is shown in the replication tree and in the information window.

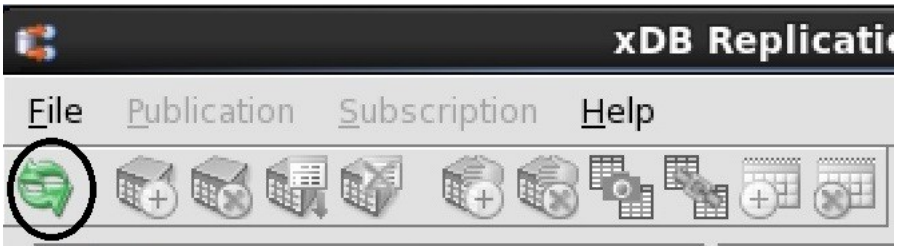


Figure 4-2 - Refresh icon

4.1.2 Create Publication

The Create Publication icon is activated when a Publication Database node is highlighted in the replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

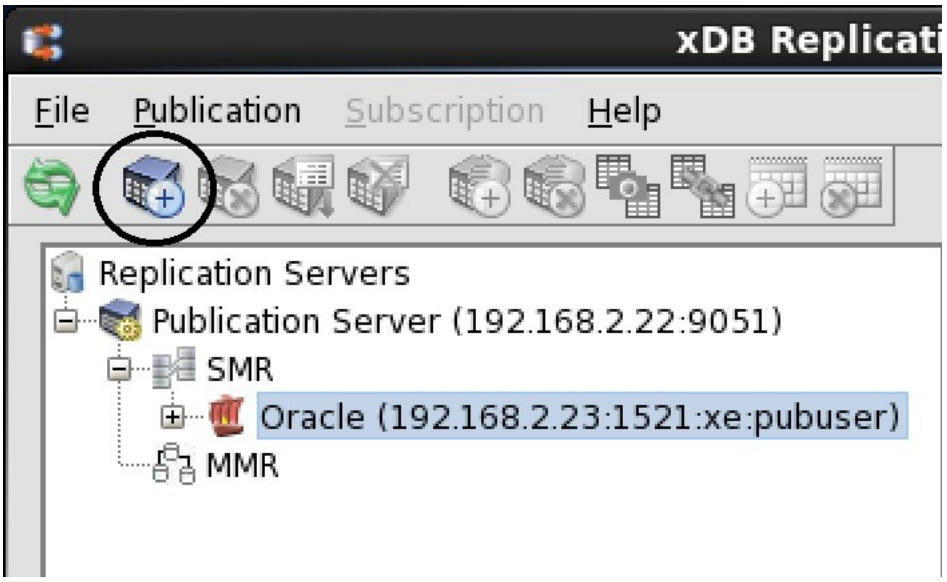


Figure 4-3 - Create Publication icon

4.1.3 Publication Management

The Remove Publication icon, Add Publication Tables icon, and Remove Publication Tables icon are activated when a Publication node is highlighted in the replication tree.

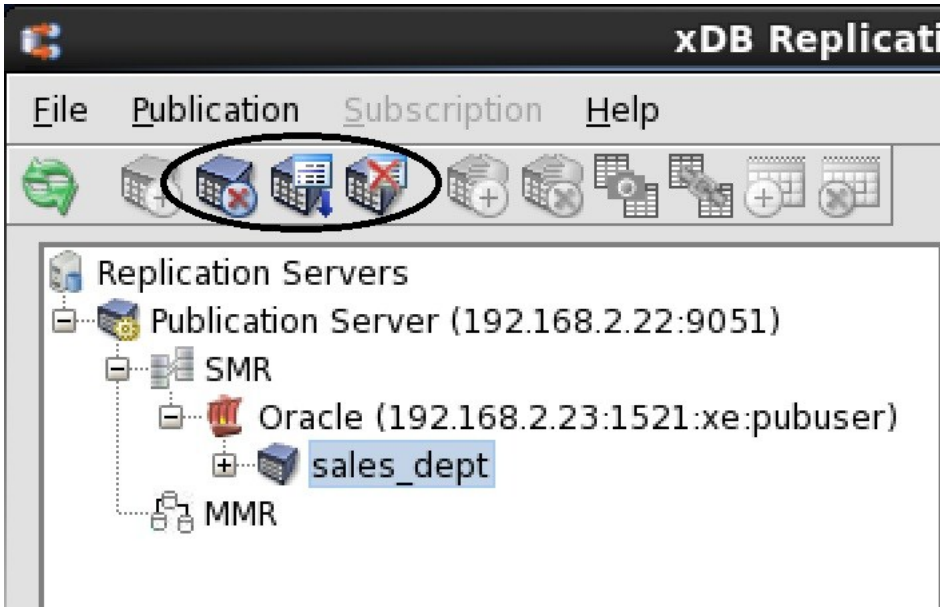


Figure 4-4 - Remove Publication, Add Publication Tables, and Remove Publication Tables icons

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

4.1.4 Create Subscription

The Create Subscription icon is activated when a Subscription Database node is highlighted in the replication tree.

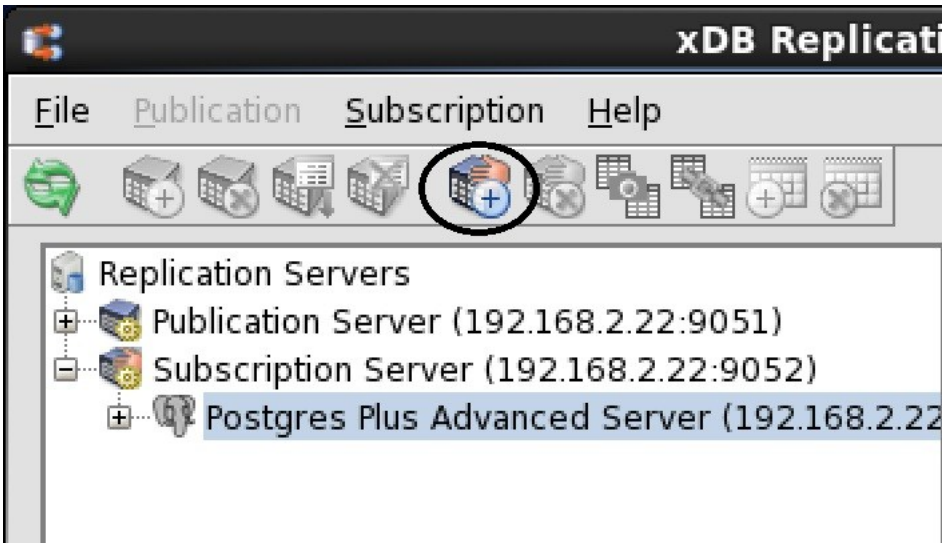


Figure 4-5 - Create Subscription icon

4.1.5 Subscription Management

The Remove Subscription icon, Snapshot icon, Synchronize icon, Configure Schedule icon, and Remove Schedule icon are activated when a Subscription node is highlighted in the replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

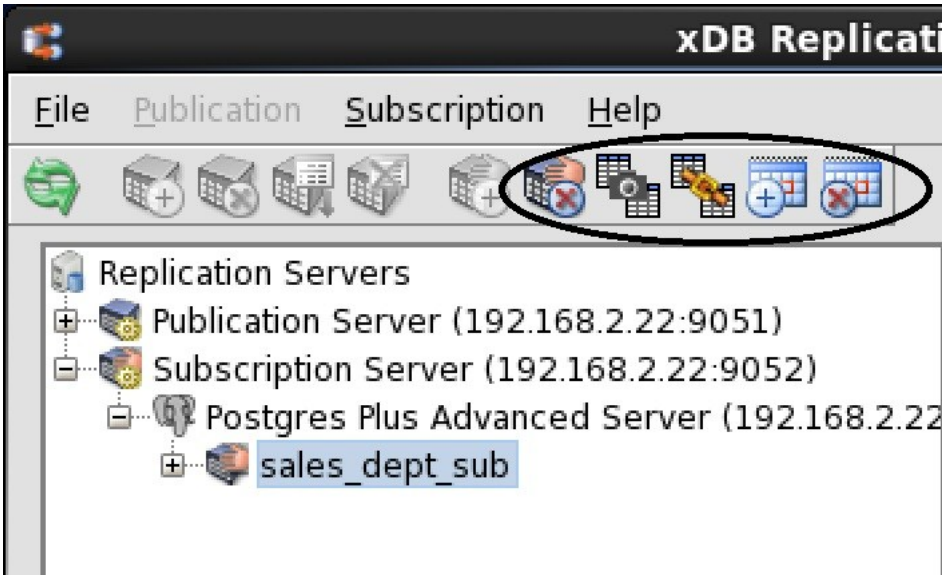


Figure 4-6 - Remove Subscription, Snapshot, Synchronize, Configure Schedule, and Remove Schedule icons

4.2 Saving Server Login Information

When you use the xDB Replication Console to create a replication system, you will need to register a publication server and a subscription server. During this process you are given the option to save the server's login information. This section describes what happens if you select this option.

The following discussion applies to both publication servers and subscription servers. These are generically referred to as "servers" in this discussion.

4.2.1 Server Login File

If you choose to save the login information, the server's network location (IP address and port number), admin user name, and password are stored in a *server login* file in a hidden location under the home directory of the operating system account with which you have opened the xDB Replication Console. See Section 3.5 for the location of this file.

The following shows the Register Publication Server dialog box where the option to save login information is presented as a check box. In this example `192.168.2.22` entered in the Host field, `9051` entered in the Port field, `admin` entered in the User Name field, and an encrypted form of the password entered in the Password field are saved in the server login file for this publication server if the admin user name and password validation are successful.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The values for User Name and Password that you enter are validated against the admin user name and password in the xDB Replication Configuration file residing on host `192.168.2.22`, in this case. The admin user name and password must successfully authenticate before registration of the publication server and saving of the publication server's login information in the server login file occur. See Section 2.3.1.3 for information on the xDB Replication Configuration file.

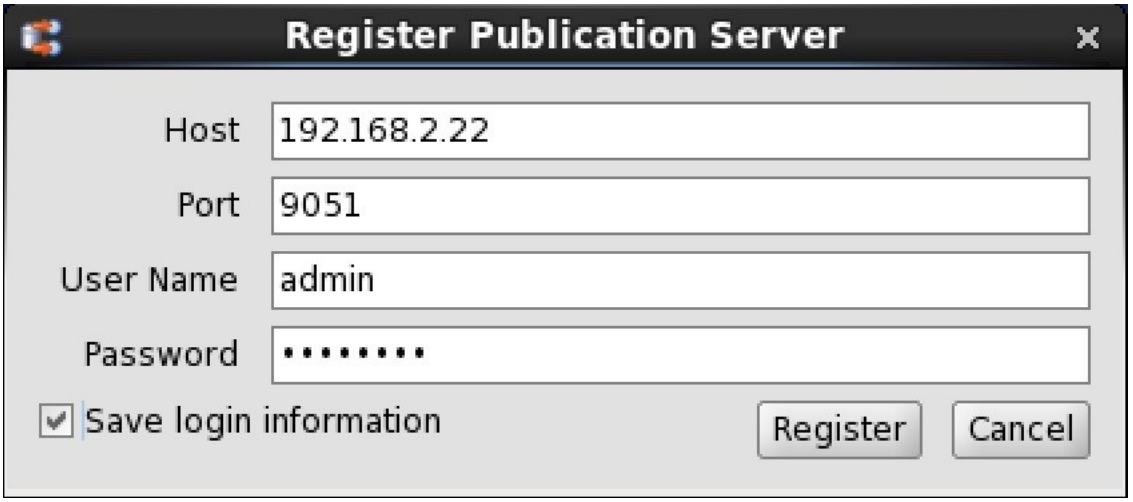


Figure 4-7 - Save login information option for a publication server

See Section 5.2.1 for more information on the purpose of these fields and the process of registering a publication server.

The following shows the Register Subscription Server dialog box. In this example `192.168.2.22` entered in the Host field, `9052` entered in the Port field, `admin` entered in the User Name field, and an encrypted form of the password entered in the Password field are saved in the server login file for this subscription server if the admin user name and password validation are successful.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Figure 4-8 - Save login information option for a subscription server

See Section 5.3.1 for more information on the purpose of these fields and the process of registering a subscription server.

Saving server login information gives you the convenience of immediate access to the publication server and any of its subordinate publications, or access to the subscription server and any of its subordinate subscriptions. That is, when you open the xDB Replication Console, the Publication Server nodes of saved publication servers immediately appear in the replication tree allowing you to perform administrative tasks on its subordinate publications.

Similarly, the Subscription Server nodes of saved subscription servers immediately appear in the replication tree allowing you to perform administrative tasks on its subordinate subscriptions.

If you did not save server login information, the server nodes would not be visible in the replication tree. You would have to re-enter the server's network location, admin user name, and password. In other words, you would have to register the server each time you open the xDB Replication Console.

Note: Each operating system account on a given host has its own server login file. Thus, the servers that are saved and appear in the xDB Replication Console when opened is independently determined for each operating system account.

4.2.2 Security Risks of Saved Server Login Information

The preceding section discussed the benefits of saving server login information.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The security risk associated with it is if unauthorized persons gain access to your operating system account, they could then potentially open the xDB Replication Console on your host using your operating system account.

If the login information of publication servers or subscription servers is saved, the corresponding Publication Server nodes or Subscription Server nodes immediately appear in the xDB Replication Console with no request for authentication information.

This allows an unauthorized person to perform any operation on the exposed publications and subscriptions including the potential to completely delete the replication system.

Note: The publication database and subscription database cannot be deleted, but unauthorized replications could be forced to occur.

Thus, it is important that operating system accounts are secure on hosts that have access to an xDB Replication Console and a replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5 Single-Master Replication Operation

This chapter describes how to configure and run xDB Replication Server for single-master replication systems.

For configuration and management of your replication system, the xDB Replication Console graphical user interface is used to illustrate the steps and examples in this chapter. The same steps can be performed from the operating system command line using the xDB Replication Server Command Line Interface (CLI). The commands of the xDB Replication Server CLI utility are described in Chapter 8.

5.1 Prerequisite Steps

Certain steps must be taken to prepare the host environments as well as the publication and subscription database servers before beginning the process of building a single-master replication system. This section describes these steps.

5.1.1 Setting Heap Memory Size for the Publication and Subscription Servers

The publication server and the subscription server are configured to run with a default set of heap size parameters. Either the default settings for 32-bit platforms or the default settings for 64-bit platforms are set by parameter `JAVA_HEAP_SIZE` when xDB Replication Server is installed.

This parameter is configured in the xDB Startup Configuration file. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

The following is an example of the xDB Startup Configuration file.

```
#!/bin/sh

JAVA_EXECUTABLE_PATH="/usr/bin/java"
JAVA_MINIMUM_VERSION=1.7
JAVA_BITNESS_REQUIRED=64
JAVA_HEAP_SIZE="-Xms256m -Xmx1536m"
PUBPORT=9051
SUBPORT=9052
```

On a 32-bit system, the initial heap size is set to 128 megabytes (`-Xms128m`) and the maximum limit is set to 512 megabytes (`-Xmx512m`). On a 64-bit system the initial heap size is 256 megabytes (`-Xms256m`) and the maximum limit is 1536 megabytes (`-Xmx1536m`).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Typically, these values can handle the average workloads. However, depending upon the average row size and pending backlog of replication updates, it may be beneficial to increase the default heap size settings.

The default values can be modified by changing the `JAVA_HEAP_SIZE` parameter setting in the xDB Startup Configuration file. Be sure to restart the publication server and the subscription server (see sections 5.2.1 and 5.3.1) after making such changes.

The heap size value should conform to the available RAM on the host running the publication server or subscription server. The basic guideline is that the maximum heap size should not exceed 25% of the total RAM size.

The following factors should also be considered.

If both the publication server and subscription server are running on the same host, then the minimum and recommended RAM capacity are shown by the following.

- **Minimum RAM Size.** For a 32-bit system, use 4 gigabytes; for a 64-bit system use 8 gigabytes.
- **Recommended RAM Size.** For a 32-bit system, use 8 gigabytes; for a 64-bit system use 16 gigabytes.

By default, both the publication server and subscription server are started and both are required for single-master replication systems. However if only multi-master replication systems are to be configured and used, then the subscription server is not needed. In such cases, the subscription server should be stopped to avoid redundant use of memory.

If both the publication server and the subscription server are running on the same host, then each server reserves its own heap buffer. Thus, the total heap size for both the publication and subscription servers, obtained by adding the maximum heap size for both servers, should comply with the available RAM on the host.

5.1.2 Enabling Synchronization Replication with the Log-Based Method

This section applies only to Postgres database servers of version 9.4 and later.

If you plan to use the log-based method of synchronization replication with any publication database running under the Postgres database server, the following configuration parameter settings are required in the configuration file, `postgresql.conf`, of the Postgres database server:

- **wal_level.** Set to `logical`.
- **max_wal_senders.** Specifies the maximum number of concurrent connections (that is, the maximum number of simultaneously running WAL sender processes). Set at minimum, to the number of SMR publication databases on this database

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

server that will use the log-based method. In addition, if MMR master nodes are to run on this database server, also add the number of MMR master nodes that will use the log-based method.

- **max_replication_slots.** Specifies the maximum number of replication slots. Set at minimum, to the number of SMR publication databases on this database server that will use the log-based method. In addition, if MMR master nodes are to run on this database server with the log-based method, see Section 2.2.10.4 for information on the additional number of replication slots required.

See Section 2.2.10 for information on the log-based method of synchronization replication.

The Postgres database server must be restarted after altering any of these configuration parameters.

In addition, the `pg_hba.conf` file requires an entry for each publication database user of publication databases that are to use the log-based method. Such database users must be included as a replication database user in the `pg_hba.conf` file. See Section 5.1.6.3 for additional information.

5.1.3 Enabling Access to the Database Servers

The following sections describe configuration steps required to use xDB Replication Server on various types of database servers.

The following section describes the steps to enable access to Oracle. See Section 5.1.3.2 for enabling access to SQL Server.

No special steps are required to enable access to a Postgres database server.

5.1.3.1 Enabling Access to Oracle

Note: The directions in this section apply only if Oracle will be used as the publication or subscription database.

An Oracle JDBC driver jar file such as, `ojdbc5.jar`, must be accessible to the Java virtual machine (JVM) on the host running the publication server and the subscription server. If the publication server and subscription server are running on separate hosts, the Oracle JDBC driver must be accessible to the JVM on each host. Oracle JDBC driver version `ojdbc5` or later must be used.

Step 1: Download the Oracle JDBC driver, for example, `ojdbc5.jar`, from the Oracle download site to the host that will be running the publication server.

Step 2: Copy file `ojdbc5.jar` to the directory *`XDB_HOME`*`/lib/jdbc`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ su root
Password:
$ cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/lib/jdbc
$ cp /home/user/Downloads/ojdbc5.jar .
$ ls -l
total 4032
-rw-rw-r-- 1 root root  355655 Jan 25 02:38 edb-jdbc14.jar
-rw-rw-r-- 1 root root  716209 Jan 25 02:38 edb-jdbc17.jar
-rw-rw-r-- 1 root root  317816 Jan 25 02:38 jtds-1.3.1.jar
-rw-r--r-- 1 root root 2091137 Jan 28 16:45 ojdbc5.jar
-rw-rw-r-- 1 root root  642809 Jan 25 02:38 postgresql-9.4-1201.jdbc4.jar
```

Note: You may also copy the `ojdbc5.jar` file to the `jre/lib/ext` subdirectory of the location where you installed your Java runtime environment.

Step 3: If the subscription server is running on a different host than the publication server, repeat steps 1 and 2 for the subscription server host.

5.1.3.2 Enabling Access to SQL Server

Note: The directions in this section apply only if SQL Server will be used as the publication or subscription database.

The jTDS JDBC driver jar file `jtds-1.3.1.jar` must be accessible to the Java virtual machine (JVM) on the host running the publication server and the subscription server. If the publication server and subscription server are running on separate hosts, the jTDS JDBC driver must be accessible to the JVM on each host.

When you install xDB Replication Server, the `jtds-1.3.1.jar` file is placed in the directory *`XDB_HOME`*`/lib/jdbc` so there is no manual configuration needed for this requirement.

Step 1: Be sure *SQL Server Authentication mode* is enabled on your SQL Server database engine. SQL Server Authentication mode allows the use of SQL Server logins such as the built-in system administrator login, `sa`.

Using the default settings for SQL Server installation, only *Windows Authentication mode* is enabled, which utilizes the accounts of the Windows operating system for authentication.

In order to permit SQL Server Authentication mode, you must change the authentication mode to *Mixed Mode Authentication*, which permits both Windows Authentication and SQL Server Authentication.

This can be done using *SQL Server Management Studio*. Refer to the appropriate SQL Server documentation for using SQL Server Management Studio.

Step 2: Be sure SQL Server is accepting TCP/IP connections. In the *SQL Server Configuration Manager*, under SQL Server Network Configuration, be sure the TCP/IP

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

protocol for the SQL Server instance is set to `Enabled`. The typical, default SQL Server instance names are `MSSQLSERVER` or `SQLEXPRESS`.

Step 3 (Required only for a SQL Server publication database): Be sure *SQL Server Agent* is enabled and running. SQL Server Agent is a Windows service that controls job scheduling and execution with SQL Server.

xDB Replication Server uses SQL Server Agent for certain operations such as for scheduled shadow table history cleanup (see Section 7.5.1).

SQL Server Agent can be started by using *SQL Server Configuration Manager*. Refer to the appropriate SQL Server documentation for using SQL Server Configuration Manager.

5.1.4 Preparing the Publication Database

This section discusses the preparation of a database that contains tables and views that will become members of publications.

The tables and views to be used for any given publication must all reside in the same database. This database becomes the publication database of that publication. A *publication database user name* must be created or already exist with the following characteristics:

- The publication database user can connect to the publication database.
- The publication database user has the privileges to create control schema objects to store metadata used for controlling and tracking the replication process.
- The publication database user can read the tables and views that are to become members of publications.
- For publications that will use synchronization replication with the trigger-based method, the publication database user can create triggers on the publication tables. (For Oracle, the publication database user must have trigger creation privilege even for snapshot-only publications, though triggers will only be created for publications using synchronization replication.)

The examples used throughout the rest of this user's guide are based on the following:

- The publication database user name is `pubuser`.
- The tables and view used in publications reside in a schema named `edb`.
- Three tables named `dept`, `emp`, and `jobhist` are members of schema `edb`.
- One view named `salesemp` is a member of schema `edb`. This view is a `SELECT` statement over the `emp` table.
- The Oracle system identifier (SID) of the publication database is `xe`. The SQL Server publication database name is `edb`. The Postgres publication database name is `edb`. (The cases of Oracle as the publication database, SQL Server as the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

publication database, and Postgres as the publication database are presented with examples in this section.)

For preparing an Oracle publication database, see the next section. For preparing a SQL Server publication database, see Section 5.1.4.2. For preparing a Postgres publication database, see Section 5.1.4.3.

5.1.4.1 Oracle Publication Database

Note (For Oracle 12c): The Oracle 12c *multitenant architecture* introduces the concept of the *container database* (CDB), which can contain multiple *pluggable databases* (PDBs). A pluggable database can be used as a publication database or a subscription database in a single-master replication system.

Oracle 12c still supports the usage of a single database referred to as a non-container database (non-CDB) that is compatible with Oracle versions prior to 12c. An Oracle 12c non-container database can also be used as a publication database or a subscription database in a single-master replication system.

The setup instructions for using an Oracle 12c publication database or subscription database are the same as for previous Oracle versions. Any special distinctions are indicated by a note within the instructions.

Step 1: Create a database user name for the publication database user. The publication database user name must have a password, and it must have the ability to create a database session. The publication database user becomes the owner of the control schema objects that will be created in the publication database to track, control, and record the replication process and history.

Note (For Oracle 12c Pluggable Database): The publication database user can be an Oracle *local user* or a *common user*. The local user exists within and has access to only a single, user-created *pluggable database* (PDB), which is to be used as the publication database. Common user names typically begin with `C##` or `c##` and can access multiple pluggable databases.

Note (For Oracle 12c Pluggable Database): Creation and granting of privileges for a local user must be done while connected to the pluggable database to be used as the publication database. Creation of a common user must be done within the Oracle 12c root container `CDB$ROOT`. Granting of privileges to the common user must be done while connected to the pluggable database to be used as the publication database.

Note (For Oracle 12c Non-Container Database): Creation and granting of privileges to the publication database user are performed in the same manner as for Oracle versions prior to 12c.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When creating the publication database definition, the publication database user name is entered in the Publication Service – Add Database dialog box (see Section 5.2.2).

```
CREATE USER pubuser IDENTIFIED BY password;
GRANT CONNECT TO pubuser;
```

Step 2: Grant the privileges needed to create the control schema objects.

The control schema objects are created in the schema owned by, and with the same name as the publication database user. That is, the publication database user's schema is the control schema for an Oracle publication database.

```
GRANT RESOURCE TO pubuser;
```

Step 3: Grant the privileges required to create triggers on the publication tables. The `CREATE ANY TRIGGER` privilege must be granted to the publication database user.

```
GRANT CREATE ANY TRIGGER TO pubuser;
```

Step 4: Grant the privileges required to lock publication tables when creating triggers. The `LOCK ANY TABLE` privilege must be granted to the publication database user.

```
GRANT LOCK ANY TABLE TO pubuser;
```

Step 5 (For Oracle 12c only): Grant the privileges required to access tablespaces. The `GRANT UNLIMITED TABLESPACE` privilege must be granted to the publication database user. This requirement applies to both a pluggable database and a non-container database.

```
GRANT UNLIMITED TABLESPACE TO pubuser;
```

Step 6: The publication database user must be able to read the tables and views that are to be included in publications.

```
GRANT SELECT ON edb.dept TO pubuser;
GRANT SELECT ON edb.emp TO pubuser;
GRANT SELECT ON edb.jobhist TO pubuser;
GRANT SELECT ON edb.salesemp TO pubuser;
```

Step 7 (Optional): Create one or more "group" roles containing the required privileges to access the tables and views of the publications that will be needed by application users.

Using roles is convenient if you wish to add new application users who need privileges to select, insert, update, or delete from any of the publication tables. A role containing the required privileges can then be granted to the new users instead of granting each privilege individually to each user.

The following example shows the creation of the role and the granting of the privileges on the publication tables to the role:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
CREATE ROLE appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.salesemp TO appgroup;
```

The following example shows the creation of a new user and the granting of the new role to the user:

```
CREATE USER appuser IDENTIFIED BY password;
GRANT CREATE SESSION TO appuser;
GRANT appgroup TO appuser;
```

5.1.4.2 SQL Server Publication Database

In SQL Server, an application gains access to the database server by supplying a *SQL Server login* and its associated password.

When an application connects to a particular database, the application assumes the identity and privileges of a *database user* that has been defined in that database. The database users in any given database are independent of database users in other databases with respect to their properties such as their role memberships and privileges. In fact, the same database user name can be defined in more than one database, each with its own distinct properties.

In each database, a database user can be *mapped* to a SQL Server login. When an application connects to a database using a SQL Server login to which a database user has been mapped, the application assumes the identity and privileges of that database user.

When using a SQL Server database as the publication database, a number of database users must be defined and mapped to a SQL Server login according to the following rules:

- A SQL Server login must exist that is to be used by the publication server to connect to SQL Server. The SQL Server login and password are specified when creating the publication database definition.
- In the publication database, a database user must exist that is to be the creator and owner of the control schema objects. This database user must be mapped to the SQL Server login used by the publication server.
- A schema must exist to contain certain control schema objects. The database user, described in the preceding bullet point, must either own this schema or have certain privileges on this schema so that the database user can create and update the control schema objects in this schema. This schema is one physical schema component of the overall control schema and must also be defined as the *default schema* of that database user. The other physical schemas comprising the overall control schema are always created by the publication server as `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The SQL Server database users that update the data in the application tables that are to be replicated must have certain privileges on the control schema objects. When an update on a replicated table occurs, a trigger fires that accesses and updates certain control schema objects. The appropriate privileges must be granted to SQL Server database users who require update access to the application tables.
- A database user must exist in the `msdb` database that is mapped to the SQL Server login used by the publication server. This database user must have certain privileges to execute jobs in the `dbo` schema of the `msdb` database. (The `msdb` database is used by *SQL Server Agent* to schedule alerts and jobs. SQL Server Agent runs as a Windows service.)

This example uses the following SQL Server login, database users, and mappings to comply with the aforementioned rules:

- The publication tables reside in database `edb`.
- The database user owning the schema containing the publication tables and the publication tables, themselves, is `edb`.
- The SQL Server login used by the publication server to connect to SQL Server is `pubuser`.
- The database user owning the control schema objects and mapped to SQL Server login `pubuser` in database `edb` is `pubuser`.
- The control schema used to contain certain control schema objects created by the publication server is `pubuser`. Other control schema objects are always created in `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`.
- The database user mapped to SQL Server login `pubuser` in database `msdb` is `pubuser_msdb`.

Note: The `sqlcmd` utility program is used to execute the SQL statements in these examples. The `USE` command establishes the database to which the subsequent statements are to apply. The `GO` command executes the preceding SQL statements as a batch. Placement of the `GO` command within a stream of SQL statements sometimes has significance depending upon the particular SQL statements.

Step 1: Create a SQL Server login for the xDB Replication Server publication database user. The login must have a password.

When creating the publication database definition, the SQL Server login is entered in the Publication Service – Add Database dialog box (see Section 5.2.2).

```
USE master;
GO
CREATE LOGIN pubuser WITH PASSWORD = 'password';
GO
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Create the database user and its required privileges for job scheduling in database `msdb`:

```
USE msdb;
GO
CREATE USER pubuser_msdb FOR LOGIN pubuser;
GO
GRANT EXECUTE ON SCHEMA :: dbo TO pubuser_msdb;
GRANT SELECT ON SCHEMA :: dbo TO pubuser_msdb;
GO
```

Step 3: Create the database user for the control schema object creation and ownership. The control schema objects are created in the publication database to track, control, and record the replication process and history. This example assumes some of the control schema objects are to be created in the schema named `pubuser`.

Note: The schema name you specify in the `WITH DEFAULT_SCHEMA` clause must be the schema you choose in Step 5. This schema does not have to exist before using it in the `CREATE USER FOR LOGIN WITH DEFAULT_SCHEMA` statement.

```
USE edb;
GO
CREATE USER pubuser FOR LOGIN pubuser WITH DEFAULT_SCHEMA = pubuser;
GO
```

Note: The remaining steps assume that the commands are given in the publication database (that is, the `USE edb` command has been previously given to establish the publication database `edb` as the current database.)

Step 4: Grant the database level privileges needed by the publication database user to create the control schema objects.

```
GRANT CREATE TABLE TO pubuser;
GRANT CREATE PROCEDURE TO pubuser;
GRANT CREATE FUNCTION TO pubuser;
GRANT CREATE SCHEMA TO pubuser;
GO
```

Step 5: Choose the control schema where some of the control schema objects are to reside.

To create the control schema objects in a new schema owned by the publication database user and created exclusively for this purpose (recommended approach) issue the following command:

```
CREATE SCHEMA pubuser AUTHORIZATION pubuser;
GO
```

Alternatively, to create the control schema objects in an existing schema such as in the same schema containing the publication tables (that is, schema `edb` in this example) use the following commands:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
GRANT ALTER   ON SCHEMA :: edb TO pubuser;
GRANT EXECUTE ON SCHEMA :: edb TO pubuser;
GRANT SELECT  ON SCHEMA :: edb TO pubuser;
GRANT INSERT  ON SCHEMA :: edb TO pubuser;
GRANT UPDATE  ON SCHEMA :: edb TO pubuser;
GRANT DELETE  ON SCHEMA :: edb TO pubuser;
GO
```

Step 6: Grant the privileges required to create triggers on the publication tables. The publication database user must have the `ALTER` privilege on the publication tables.

```
GRANT ALTER ON edb.dept TO pubuser;
GRANT ALTER ON edb.emp TO pubuser;
GRANT ALTER ON edb.jobhist TO pubuser;
GO
```

Step 7: The publication database user must be able to read the tables and views that are to be included in publications.

```
GRANT SELECT ON edb.dept TO pubuser;
GRANT SELECT ON edb.emp TO pubuser;
GRANT SELECT ON edb.jobhist TO pubuser;
GRANT SELECT ON edb.salesemp TO pubuser;
GO
```

Step 8 (Optional): Create one or more "group" roles containing the required privileges to access the tables and views of the publications that will be needed by application users.

Note: Creation of these roles can only be done after the SQL Server publication database definition has been created using the xDB Replication Console or xDB Replication Server CLI. (For example, see Section 5.2.2 for the xDB Replication Console usage.)

Using roles is convenient if you wish to add new application users who need privileges to select, insert, update, or delete from any of the publication tables. A role containing the required privileges can then be granted to the new users instead of granting each privilege individually to each user.

In addition to privileges on the publication tables, any user performing an insert, update, or delete operation on any of the publication tables requires privileges to certain control schema objects of the publication.

The following example shows the creation of the role `appgroup` and the granting of privileges on the publication tables to the role. The example assumes that in Step 5, schema `pubuser` was chosen as the control schema to store some of the control schema objects.

```
CREATE ROLE appgroup AUTHORIZATION edb;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.salesemp TO appgroup;
GRANT EXECUTE ON SCHEMA :: _edb_replicator_pub TO appgroup;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
GRANT SELECT  ON SCHEMA :: _edb_replicator_pub TO appgroup;
GRANT INSERT  ON SCHEMA :: _edb_replicator_pub TO appgroup;
GRANT UPDATE  ON SCHEMA :: _edb_replicator_pub TO appgroup;
GRANT INSERT  ON SCHEMA :: pubuser TO appgroup;
GO
```

The following example shows the creation of a new login and database user, and the addition of the database user as a member of the role in order to inherit its privileges:

```
CREATE LOGIN applogin WITH PASSWORD = 'password', DEFAULT_DATABASE = edb;
CREATE USER appuser FOR LOGIN applogin WITH DEFAULT_SCHEMA = edb;
EXEC sp_addrolemember @rolename = 'appgroup', @membername = 'appuser';
GO
```

Note (Granting privileges to individual users): As previously described, each application database user that is to modify the data in any of the publication tables must be granted certain privileges on the publication tables and the control schema objects. Using a group role for this purpose as described earlier in this step helps simplify this process.

Individual database users can be granted the privileges to access the publication tables and the controls schema objects in a similar fashion.

The following example shows the creation of a new login and database user, and the granting of the privileges on the publication tables and the control schema objects to the user:

```
CREATE LOGIN newlogin WITH PASSWORD = 'password', DEFAULT_DATABASE = edb;
CREATE USER newuser FOR LOGIN newlogin WITH DEFAULT_SCHEMA = edb;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.salesemp TO newuser;
GRANT EXECUTE ON SCHEMA :: _edb_replicator_pub TO newuser;
GRANT SELECT  ON SCHEMA :: _edb_replicator_pub TO newuser;
GRANT INSERT  ON SCHEMA :: _edb_replicator_pub TO newuser;
GRANT UPDATE  ON SCHEMA :: _edb_replicator_pub TO newuser;
GRANT INSERT  ON SCHEMA :: pubuser TO newuser;
GO
```

Note: Instead of using the preceding statements, which grant privileges at the schema level, a more granular level of privileges can be issued at the database object level using the following statements:

```
CREATE LOGIN newlogin WITH PASSWORD = 'password', DEFAULT_DATABASE = edb;
CREATE USER newuser FOR LOGIN newlogin WITH DEFAULT_SCHEMA = edb;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.salesemp TO newuser;
GRANT INSERT  ON pubuser.rrst_edb_dept TO newuser;
GRANT INSERT  ON pubuser.rrst_edb_emp TO newuser;
GRANT INSERT  ON pubuser.rrst_edb_jobhist TO newuser;
GO
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In addition, depending upon the version of SQL Server, grant the following additional privileges.

For SQL Server 2008: Grant the following privileges:

```
GRANT EXECUTE ON _edb_replicator_pub.nextval TO newuser;
GRANT SELECT  ON _edb_replicator_pub.rrep_tx_seq TO newuser;
GRANT INSERT  ON _edb_replicator_pub.rrep_tx_seq TO newuser;
GO
```

For SQL Server 2012, 2014: Grant the following privileges:

```
GRANT UPDATE ON _edb_replicator_pub.rrep_tx_seq TO newuser;
GRANT UPDATE ON _edb_replicator_pub.rrep_txset_seq TO newuser;
GRANT UPDATE ON _edb_replicator_pub.rrep_common_seq TO newuser;
GO
```

Using this approach, however, requires you to issue additional privileges for each application table that is later added to the publication.

5.1.4.3 Postgres Publication Database

When creating the publication database definition, a database user name must be specified that has the following characteristics:

- The database user can connect to the publication database.
- The database user has superuser privileges. Superuser privileges are required because the database configuration parameter `session_replication_role` is altered by the database user to `replica` for snapshot operations involving replication of the control schema from one publication database to another.
- The database user must have the ability to modify the system catalog tables in order to disable foreign key constraints on the control schema tables for snapshot operations involving replication of the control schema from one publication database to another. (See appendix Section 10.4.4 for more information on this requirement.)

Step 1: Create a database superuser for the publication database user. The publication database user name must have a password, and it must have the ability to create a database session. The publication database user becomes the owner of the control schema objects that will be created in the publication database to track, control, and record the replication process and history.

When creating the publication database definition, the publication database user name is entered in the Publication Service – Add Database dialog box (see Section 5.2.2).

```
CREATE ROLE pubuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2 (Optional): Create one or more "group" roles containing the required privileges to access the tables and views of the publications that will be needed by application users.

Note: The process described in this step is applicable to Postgres publications in both single-master and multi-master replication systems.

Using roles is convenient if you wish to add new application users who need privileges to select, insert, update, or delete from any of the publication tables. A role containing the required privileges can then be granted to the new users instead of granting each privilege individually to each user.

Any user performing an insert, update, or delete operation on any of the publication tables requires privileges on the publication tables and its schema as well as to certain control schema objects of the publication. These control schema objects reside under schema `_edb_replicator_pub`.

The following example shows the creation of the role `appgroup` and the granting of privileges on the publication tables to the role.

```
CREATE ROLE appgroup;
GRANT USAGE ON SCHEMA edb TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO appgroup;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.salesemp TO appgroup;
```

In addition, for the log-based method of synchronization replication, if the `TRUNCATE` command is to be permitted on the publication tables, grant the following additional privileges:

```
GRANT TRUNCATE ON edb.dept TO appgroup;
GRANT TRUNCATE ON edb.emp TO appgroup;
GRANT TRUNCATE ON edb.jobhist TO appgroup;
```

Also for the log-based method of synchronization replication for usage of the `TRUNCATE` command, grant the following privileges after creation of the publication database definition. (See Section 5.2.2 for information on creating the publication database definition for a single-master replication system. For a multi-master replication system, see Section 6.2.2.)

```
GRANT USAGE ON SCHEMA _edb_replicator_pub TO appgroup;
GRANT INSERT ON _edb_replicator_pub.rrep_wal_events_queue TO appgroup;
```

Finally, grant the group role to the desired database users. The following example shows the creation of a new user and the granting of the role to the user:

```
CREATE ROLE appuser WITH LOGIN PASSWORD 'password';
GRANT appgroup TO appuser;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note (Granting privileges to roles after publication creation): Roles for containing publication table privileges should be created before you create the publication. (See Section 5.2.3 for information on creating a publication for a single-master replication system. For a multi-master replication system, see Section 6.2.3.)

When you create the publication, the privileges that have been granted on the publication tables to roles that exist at the time, are applied to the control schema objects for those roles. So for the preceding example, the privileges required on the control schema objects for any publication created using `edb.dept`, `edb.emp`, `edb.jobhist`, or `edb.salesemp` are granted to role `appgroup` when you create that publication.

If, however, you create a role after the publication is created, you must explicitly grant the necessary privileges on the publication tables and control schema objects to the new role.

When using the trigger-based method of synchronization replication, a role must be granted the following privileges on the control schema objects:

- `USAGE` privilege on schema `_edb_replicator_pub`.
- `USAGE` privilege on sequence `rrep_tx_seq`.
- `INSERT` privileges on the shadow tables corresponding to publication tables in which the role will be inserting, updating, or deleting rows. Shadow tables follow the naming convention `rrst_schema_table`. Note that shadow tables exist only if the trigger-based method of synchronization is to be used.

The following example shows the creation of a new role and the granting of the privileges on the publication tables and the control schema objects to the role for the trigger-based method of synchronization replication:

```
CREATE ROLE newuser WITH LOGIN PASSWORD 'password';
GRANT USAGE ON SCHEMA edb TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO newuser;
GRANT USAGE ON SCHEMA _edb_replicator_pub TO newuser;
GRANT USAGE ON SEQUENCE _edb_replicator_pub.rrep_tx_seq TO newuser;
GRANT INSERT ON _edb_replicator_pub.rrst_edb_dept TO newuser;
GRANT INSERT ON _edb_replicator_pub.rrst_edb_emp TO newuser;
GRANT INSERT ON _edb_replicator_pub.rrst_edb_jobhist TO newuser;
```

When using the log-based method a role needs access to the publication tables and to certain control schema objects as well under certain circumstances.

When using the log-based method of synchronization replication, a role must be granted the following privileges on the control schema objects if the role is to be permitted to use the `TRUNCATE` command on the publication tables:

- `USAGE` privilege on schema `_edb_replicator_pub`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- `INSERT` privilege on table `_edb_replicator_pub.rrep_wal_events_queue`.

The following example shows the creation of a new role and the granting of the privileges on the publication tables to the role for the log-based method of synchronization replication:

```
CREATE ROLE newuser WITH LOGIN PASSWORD 'password';
GRANT USAGE ON SCHEMA edb TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.dept TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.emp TO newuser;
GRANT SELECT, INSERT, UPDATE, DELETE ON edb.jobhist TO newuser;
```

In addition, if the `TRUNCATE` command is to be permitted on the publication tables, grant the following additional privileges:

```
GRANT TRUNCATE ON edb.dept TO newuser;
GRANT TRUNCATE ON edb.emp TO newuser;
GRANT TRUNCATE ON edb.jobhist TO newuser;
GRANT USAGE ON SCHEMA _edb_replicator_pub TO newuser;
GRANT INSERT ON _edb_replicator_pub.rrep_wal_events_queue TO newuser;
```

5.1.5 Preparing the Subscription Database

This section discusses the preparation of a database that will be used as a subscription database.

The tables and views in a given publication must all be replicated to the same database. This database is called the subscription database. A *subscription database user name* must be created with the following characteristics:

- The subscription database user can connect to the subscription database.
- The subscription database user has the privileges to create database objects for the replicated tables and views from publications.
- The subscription database user has the privileges necessary to execute the `TRUNCATE` command on the replicated tables.

See Section 5.1.5.1 for preparation of a Postgres subscription database. See Section 5.1.5.2 for preparation of an Oracle subscription database. See Section 5.1.5.3 for preparation of a SQL Server subscription database.

5.1.5.1 Postgres Subscription Database

A database user name must be chosen or created to serve as the subscription database user. The user name must have a password. The subscription database user becomes the owner of the replicated database objects.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When creating the subscription database definition, the subscription database user name is entered in the Subscription Service – Add Database dialog box (see Section 5.3.2).

The subscription database user must also have the ability to run the `TRUNCATE` command on the subscription tables. This requires the following:

- The subscription database user must have superuser privileges.
- The subscription database user must have the ability to modify the system catalog tables in order to disable foreign key constraints on subscription tables. (See appendix Section 10.4.4 for more information on this requirement.)

You have the following two choices for choosing the subscription database user name:

- Use the Postgres user name `postgres` created upon installation of PostgreSQL (`enterprisedb` for Advanced Server installed in Oracle compatible configuration mode) for the subscription database user name. If you choose this option, skip Step 1 and proceed to Step 2.
- Create a new subscription database user name. For this option, proceed to Step 1.

Step 1: Create a superuser as the subscription database user.

```
CREATE ROLE subuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

Step 2: Create or choose the subscription database.

The names of the schemas containing the publication tables and views become the names of the Postgres schemas for the subscription tables. The subscription server creates these schemas in the subscription database when the subscription is created. If schemas with these names already exist in the subscription database, the existing schemas will be used to store the subscription tables.

For a SQL Server publication database: If the schema containing the publication tables and views in SQL Server is named `dbo`, then the subscription server creates a schema named `dbo_sql` in the Postgres subscription database for the subscription tables. (Schema `dbo` is a special reserved schema in Postgres.)

The existing schemas must not contain any tables or views with the same names as the publication tables and views. The subscription server returns an error if there are already identically named tables or views. You must delete or rename these tables and views before the subscription can be created.

A new subscription database owned by the subscription database user `subuser` can be created with the following:

```
CREATE DATABASE subdb OWNER subuser;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.1.5.2 Oracle Subscription Database

Step 1 (Optional): If you do not have an existing database that you want to use as your subscription database, create a new database. This step can be fairly complicated. Refer to the appropriate Oracle documentation for performing this task.

Step 2: Create a database user name for the subscription database user. The subscription database user name must have a password, and it must have the ability to create a database session. The subscription database user becomes the owner of the replicated database objects.

Note (For Oracle 12c Pluggable Database): The subscription database user can be an Oracle *local user* or a *common user*. The local user exists within and has access to only a single, user-created *pluggable database* (PDB), which is to be used as the subscription database. Common user names typically begin with `C##` or `c##` and can access multiple pluggable databases.

Note (For Oracle 12c Pluggable Database): Creation and granting of privileges for a local user must be done while connected to the pluggable database to be used as the subscription database. Creation of a common user must be done within the Oracle 12c root container `CDB$ROOT`. Granting of privileges to the common user must be done while connected to the pluggable database to be used as the subscription database.

Note (For Oracle 12c Non-Container Database): Creation and granting of privileges to the subscription database user are performed in the same manner as for Oracle versions prior to 12c.

When creating the subscription database definition, the subscription database user name is entered in the Subscription Service – Add Database dialog box (see Section 5.3.2).

```
CREATE USER subuser IDENTIFIED BY password;
GRANT CONNECT TO subuser;
```

Step 3: Grant the privileges needed to create the replicated database objects.

The replicated database objects are created in the schema owned by, and with the same name as the subscription database user.

```
GRANT RESOURCE TO subuser;
```

Step 4 (For Oracle 12c only): Grant the privileges required to access tablespaces. The `GRANT UNLIMITED TABLESPACE` privilege must be granted to the subscription database user. This requirement applies to both a pluggable database and a non-container database.

```
GRANT UNLIMITED TABLESPACE TO subuser;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.1.5.3 SQL Server Subscription Database

Step 1: Create or choose the subscription database.

The names of the schemas containing the publication tables and views become the names of the SQL Server schemas for the subscription tables. The subscription server creates these schemas in the subscription database when the subscription is created. If schemas with these names already exist in the subscription database, the existing schemas will be used to store the subscription tables.

Note: If the schema containing the publication tables and views is named `public`, then the subscription server creates a schema named `public_sql` in the SQL Server subscription database for the subscription tables.

The existing schemas must not contain any tables or views with the same names as the publication tables and views. The subscription server returns an error if there are already identically named tables or views. You must delete or rename these tables and views before the subscription can be created.

A new subscription database can be created as shown by the following:

```
USE master;
GO
CREATE DATABASE subdb;
GO
```

Step 2: Create a SQL Server login for the subscription database user. The login must have a password.

When creating the subscription database definition, the SQL Server login is entered in the Subscription Service – Add Database dialog box (see Section 5.3.2).

```
CREATE LOGIN subuser WITH PASSWORD = 'password';
GO
```

Step 3: In the subscription database, a database user must exist that is to be the creator and owner of the subscription tables. This database user must be mapped to the SQL Server login created in Step 2.

In this example, the database user is given the same name as the SQL Server login `subuser`.

```
USE subdb;
GO
CREATE USER subuser FOR LOGIN subuser;
GO
```

Step 4: Grant the database level privileges needed by the subscription database user to create the schema and tables for the subscription.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
GRANT CREATE SCHEMA TO subuser;
GRANT CREATE TABLE TO subuser;
GO
```

5.1.6 Verifying Host Accessibility

If more than one computer is used to host the components of the replication system, each computer must be able to communicate with the others on a network.

There are a number of different aspects of this topic as discussed in the following sections.

5.1.6.1 Firewalls and Access to Ports

Verify that the firewalls on the hosts allow access from the other hosts running replication system components. Refer to the directions for your host's operating system to enable accessibility.

In addition if you are running the xDB Replication Console or the xDB Replication Server CLI on a different host than where the publication server or subscription server are running, be sure the firewall on the host running these servers allows access to the ports used by the publication server and subscription server.

The xDB Replication Console and xDB Replication Server CLI access the publication server and the subscription server using Java Remote Method Invocation (RMI) through the designated ports.

The publication server uses the port number you specified on the Publication Server Details screen in Step 16 of Section 3.1 as well the port offset by a value of 2 greater than this specified port number. So for a default publication server installation, access is required for port numbers `9051` and `9053`.

The subscription server uses the port number you specified on the Subscription Server Details screen in Step 17 of Section 3.1 as well as the port offset by a value of 2 greater than this specified port number. So for a default subscription server installation, access is required for port numbers `9052` and `9054`.

When you install xDB Replication Server, the port numbers you specify for the publication server and the subscription server are stored in the xDB Startup Configuration file as shown by the following example. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

```
#!/bin/sh

JAVA_EXECUTABLE_PATH="/usr/bin/java"
JAVA_MINIMUM_VERSION=1.7
JAVA_BITNESS_REQUIRED=64
JAVA_HEAP_SIZE="-Xms256m -Xmx1536m"
PUBPORT=9051
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
SUBPORT=9052
```

If you want to use different port numbers, modify the `PUBPORT` and `SUBPORT` entries in the xDB Startup Configuration file and restart the publication server and subscription server.

Note: If you change the port numbers for the publication server or subscription server for which there are existing replication systems, there are additional updates you must perform upon these existing replication systems. See Section 7.6.1.2 for changes that must be made for the publication server metadata in the control schema if the port number used by the subscription server has been changed. See Section 5.5.3 for changes that must be made for the subscription metadata in the control schema if the port number used by the publication server has been changed.

5.1.6.2 Network IP Addresses

When configuring a replication system, you must supply the network location of various components such as the publication server, subscription server, publication database server, and subscription database server. This information, consisting of the component's IP address and port number, is stored in the control schema.

When one component needs to access another, it refers to the network location stored in the control schema.

During replication system configuration it is strongly suggested that you supply the actual network IP address of each component and avoid the usage of the loopback address, localhost or 127.x.x.x, even if all components are running on the same host.

You can obtain the network IP address using the following command:

For Linux only: Use the `/sbin/ifconfig` command.

For Windows only: Open a Command Prompt window and use the `ipconfig` command.

The loopback address works as long as the communicating components are on the same host, but if at some future point, you decide to move a component to a different host on the network, the loopback address stored as the component's network address in the control schema will no longer work for the component trying to make the connection.

For Linux only: You may need to modify the `/etc/hosts` file so that a host's network IP address is associated with the host's name.

Note: For an alternative to modifying the `/etc/hosts` file see Section 10.4.1.7.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The default configuration on Linux systems associates the host name with the loopback address in the `/etc/hosts` file as shown by the following example:

```
127.0.0.1               localhost.localdomain localhost
```

This is also verified by using the `hostname -i` command, which returns the IP address associated with the host name:

```
$ hostname -i
127.0.0.1
```

In these circumstances, certain xDB Replication Server components will have trouble locating its other components on the network as in the following cases:

- When the user interface attempts to connect to the publication server or subscription server
- When the subscription server attempts to connect to the publication server

If the loopback address `127.`*`x`*`.`*`x`*`.`*`x` is returned such as in the preceding example, edit the `/etc/hosts` file so that the network IP address is associated with the host name instead.

The following example shows the modified `/etc/hosts` file so that the host name `localhost` is now associated with the network IP address `192.168.2.22` instead of the loopback address `127.0.0.1`:

```
#127.0.0.1              localhost.localdomain localhost
192.168.2.22            localhost.localdomain localhost
::1                     localhost6.localdomain6 localhost6
```

On some Linux systems, you may need to restart the network service after you have modified the `/etc/hosts` file. This may be done a number of different ways depending upon the Linux system you are using as shown by the following variations:

```
service network restart
/etc/init.d/networking restart
sudo /etc/init.d/networking restart
```

The following example illustrates the `service network` command:

```
$ su root
Password:
$ service network restart
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
```

Use the following command for CentOS 7 or RHEL 7:

```
systemctl restart network
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The `hostname -i` command now returns the network IP address of the host:

```
$ hostname -i
192.168.2.22
```

5.1.6.3 Postgres Server Authentication

A Postgres database server uses the host-based authentication file, `pg_hba.conf`, to control access to the databases in the database server.

You need to modify the `pg_hba.conf` file in the following locations:

- On each Postgres database server that contains a Postgres publication database
- On each Postgres database server that contains a Postgres subscription database

In a default Postgres installation, this file is located in the directory *POSTGRES_INSTALL_HOME*`/data`.

The modifications needed to the `pg_hba.conf` file for each of the aforementioned cases are discussed in the following sections.

Postgres Publication Database

For a Postgres publication database, the following is needed to allow access to the publication database:

```
host pub_dbname    pub_dbuser    pub_ipaddr/32   md5
host pub_dbname    pub_dbuser    sub_ipaddr/32   md5
```

The value you substitute for *pub_dbname* is the name of the Postgres publication database you intend to use. The value you substitute for *pub_dbuser* is the publication database user name you created in Step 1 of Section 5.1.4.3.

For a Postgres publication database named `edb`, the resulting `pg_hba.conf` file appears as follows:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                                 md5
# IPv4 local connections:
host    edb         pubuser       192.168.2.22/32     md5
host    all         all           127.0.0.1/32        md5
# IPv6 local connections:
host    all         all           ::1/128             md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                      md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: The preceding example assumes the publication server and the subscription server are running on the same host, hence the single entry for database `edb`. If the publication server and subscription server are running on separate hosts, then the `pg_hba.conf` file on the publication database server would look like the following:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                             md5
# IPv4 local connections:
host    edb         pubuser       192.168.2.22/32   md5
host    edb         pubuser       192.168.2.24/32   md5
host    all         all           127.0.0.1/32      md5
# IPv6 local connections:
host    all         all           ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                    md5
#host    replication  enterprisedb  127.0.0.1/32      md5
#host    replication  enterprisedb  ::1/128           md5
```

In addition, the preceding examples assume publication database `edb` is using the trigger-based method of synchronization replication. If the log-based method is used, the `pg_hba.conf` file must contain an additional entry with the DATABASE field set to `replication` for *`pub_dbname`*, *`pub_dbuser`*, and *`pub_ipaddr`* to allow replication connections from the publication server on the host on which it is running.

The following shows a modification of the preceding example with this additional entry as the last line in the file:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                             md5
# IPv4 local connections:
host    edb         pubuser       192.168.2.22/32   md5
host    edb         pubuser       192.168.2.24/32   md5
host    all         all           127.0.0.1/32      md5
# IPv6 local connections:
host    all         all           ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                    md5
#host    replication  enterprisedb  127.0.0.1/32      md5
#host    replication  enterprisedb  ::1/128           md5
host    replication  pubuser       192.168.2.22/32   md5
```

See sections 2.2.10 and 5.1.2 for additional information on synchronization replication with the log-based method.

Reload the configuration file after making the modifications.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Choose Reload Configuration (Expert Configuration, then Reload Configuration on Advanced Server) from the Postgres application menu. This will put the modified `pg_hba.conf` file into effect.

Postgres Subscription Database

For a Postgres subscription database, the following entries are needed to allow access to the subscription database:

```
host sub_dbname    sub_dbuser    pub_ipaddr/32   md5
host sub_dbname    sub_dbuser    sub_ipaddr/32   md5
```

The values you substitute for *sub_dbuser* and *sub_dbname* are the subscription database user name and the subscription database name you created in steps 1 and 2 of Section 5.1.5.1.

For a Postgres subscription database named `subdb`, the resulting `pg_hba.conf` file appears as follows:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                               md5
# IPv4 local connections:
host    subdb       subuser       192.168.2.22/32     md5
host    all         all           127.0.0.1/32        md5
# IPv6 local connections:
host    all         all           ::1/128             md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                      md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
```

Note: The preceding example assumes that the publication server and the subscription server are running on the same host hence, only one entry is needed for database `subdb`. If the publication server and subscription server are running on separate hosts, then the `pg_hba.conf` file on the subscription database server looks like the following:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                               md5
# IPv4 local connections:
host    subdb       subuser       192.168.2.22/32     md5
host    subdb       subuser       192.168.2.24/32     md5
host    all         all           127.0.0.1/32        md5
# IPv6 local connections:
host    all         all            ::1/128            md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                      md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Reload the configuration file after making the modifications.

Choose Reload Configuration (Expert Configuration, then Reload Configuration on Advanced Server) from the Postgres application menu. This will put the modified `pg_hba.conf` file into effect.

5.2 Creating a Publication

Creating your first publication requires the following steps:

- Registering the publication server
- Adding the publication database
- Creating a publication by choosing the tables and views for the publication along with creating any optional filter clauses

Once the publication database has been added, as many publications can be created as there are available tables and views that are readable by the publication database user and that meet the criteria outlined in sections 2.4.2 and 2.4.3.

5.2.1 Registering a Publication Server

When you register a publication server, you are identifying the network location, admin user name, and password of a specific, running, publication server instance that you want to use to manage all aspects of the publications you will be creating subordinate to it.

It is important that you record the login information for the publication server as you must always use this same publication server instance to manage all publications created subordinate to it as represented in the xDB Replication Console replication tree.

Step 1: Start the publication server if it is not already running.

Note: If you are using Oracle publication or subscription databases, and the publication server has not been restarted since copying the Oracle JDBC driver to the `lib/jdbc` subdirectory of your xDB Replication Server installation, you must restart the publication server.

For Linux only: You can verify the publication server is running by using the `systemctl` command for CentOS 7 or RHEL 7, and the `service` command for previous Linux versions.

Use the following command for CentOS 7 or RHEL 7:

```
systemctl status edb-xdbpubserver
```

Use the following command for previous Linux versions:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
service edb-xdbpubserver status
```

If the publication server is running and you wish to restart it, use the `restart` option.

For CentOS 7 or RHEL 7:

```
systemctl restart edb-xdbpubserver
```

For previous Linux versions:

```
service edb-xdbpubserver restart
```

If the publication server is not running, use the `start` option.

For CentOS 7 or RHEL 7:

```
systemctl start edb-xdbpubserver
```

For previous Linux versions:

```
service edb-xdbpubserver start
```

Similarly, use the `stop` option to stop the publication server.

For Windows only: Open Control Panel, System and Security, Administrative Tools, and then Services. The publication server runs as a service named `Publication Service for xDB Replication Server.`

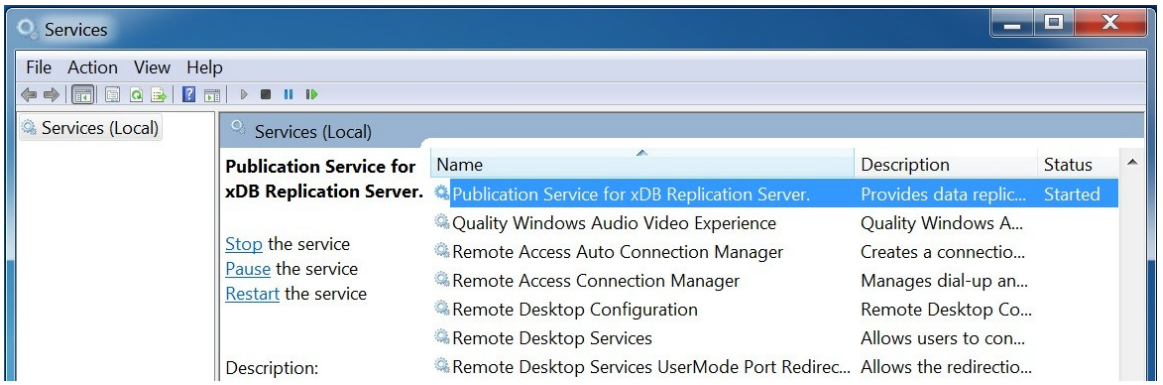


Figure 5-1 - Windows publication service

Use the Start or Restart link for the service.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If the publication server fails to start, see Section 10.3.2.4 for information.

Step 2: Register the publication server. Open the xDB Replication Console from the system's application menu. For xDB Replication Server installed from an xDB RPM package, the xDB Replication Console is started by invoking the script *XDB_HOME*/bin/runRepConsole.sh.

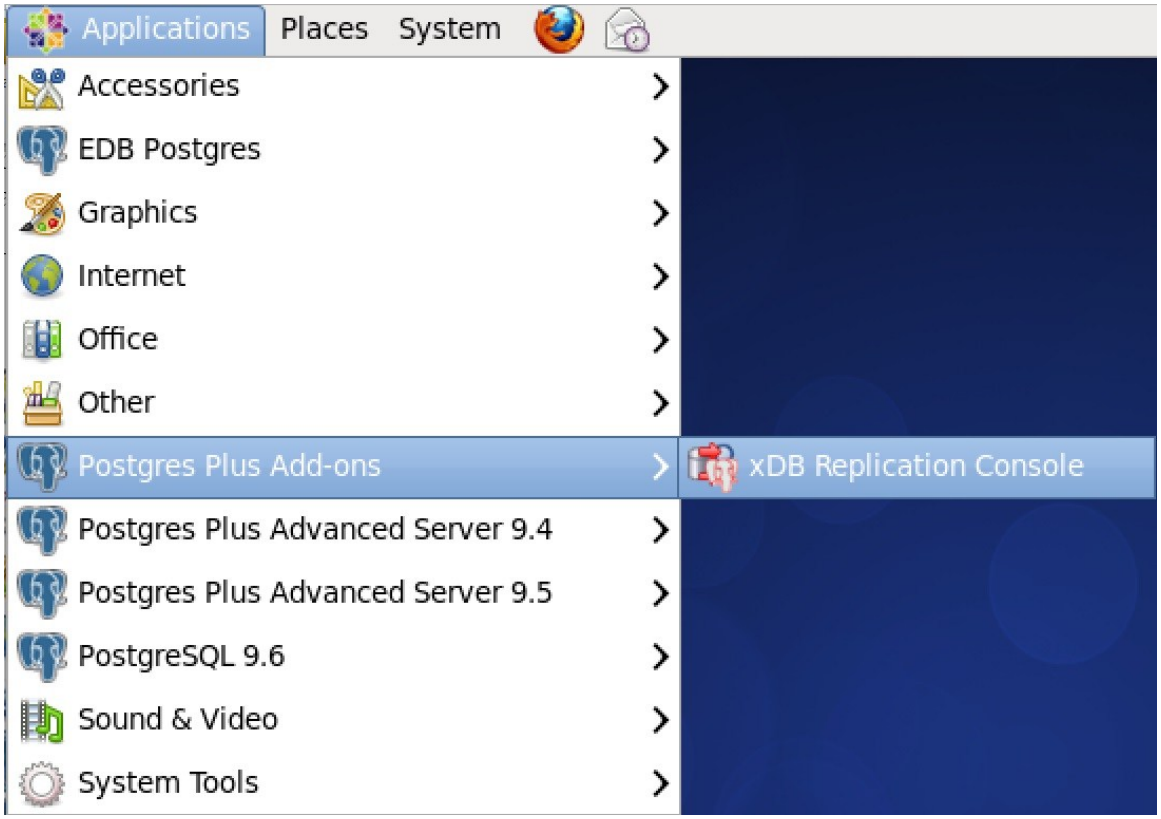


Figure 5-2 - xDB Replication Console menu option

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

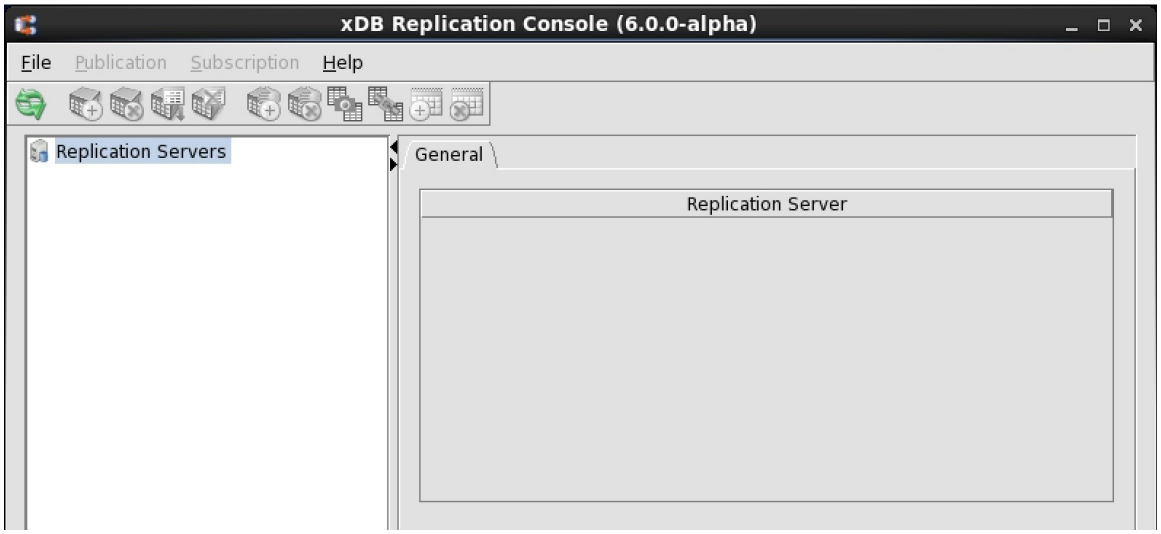


Figure 5-3 - xDB Replication Console

Step 3: Select the top level Replication Servers node. From the File menu, choose Publication Server, and then choose Register Server. Alternatively, click the secondary mouse button on the Replication Servers node and choose Register Publication Server. The Register Publication Server dialog box appears.

Enter the values you supplied during the installation of xDB Replication Server unless otherwise specified.

- **Host.** Network IP address of the host running the publication server. This is the network IP address used for *pub_ipaddr* in the `pg_hba.conf` file in Section 5.1.6.3. (Do not use `localhost` for this field.)
- **Port.** Port number the publication server is using. This is the port number you specified on the Publication Server Details screen in Step 16 of Section 3.1.
- **User Name.** Admin user name that is used to authenticate your usage of this publication server. This is the user name you specified on the xDB Admin User Details screen in Step 15 of Section 3.1.
- **Password.** Password of the admin user given in the User Name field.
- **Save login information.** Check this box if you do not want to re-register the publication server each time you open the xDB Replication Console. See Section 4.2 for additional information on the advantages and disadvantages of saving server login information.

Note: The user name and password combination you enter is authenticated against the admin user name and password in the xDB Replication Configuration file residing on the host with the IP address you enter in the Host field.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

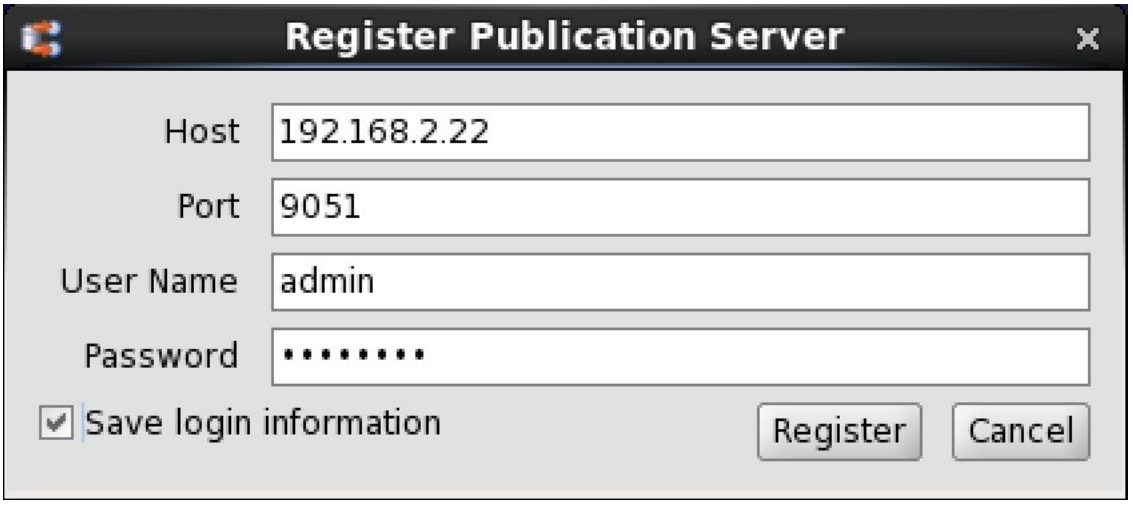


Figure 5-4 - Register Publication Server dialog box

Click the Register button after you have filled in the fields. A Publication Server node appears in the replication tree of the xDB Replication Console. Expand the Publication Server node to expose the SMR and MMR type nodes.

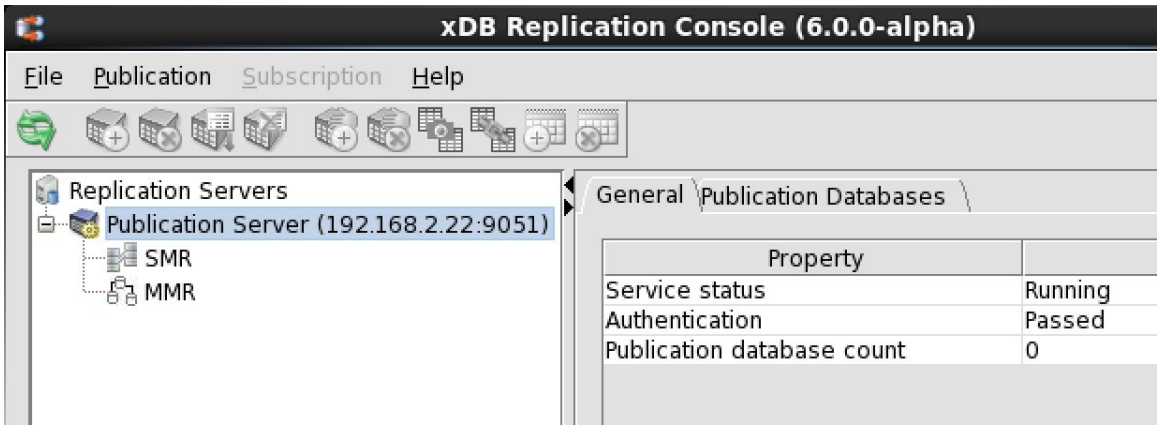


Figure 5-5 – Replication tree after registering a publication server

Continue to build the single-master replication system under the SMR type node.

5.2.2 Adding a Publication Database

The database in which publications are to reside must be identified to xDB Replication Server. This is done by creating a publication database definition.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

After the publication database definition is created, a Publication Database node representing that publication database definition appears in the replication tree of the xDB Replication Console. Publications that are to contain tables and views residing within this database can then be created under the Publication Database node.

You must enter database connection information such as the database server network address, database identifier, and database login user name and password when you create the publication database definition. The connection information is used by the publication server to access the publication tables and views when it performs replication.

Step 1: Make sure the database server in which the publication database resides is running and accepting client connections.

Step 2: Select the SMR type node under the Publication Server node. From the Publication menu, choose Publication Database, and then choose Add Database. Alternatively, click the secondary mouse button on the SMR type node and choose Add Database. The Publication Service – Add Database dialog box appears.

Step 3: Fill in the following fields:

- **Database Type.** Select Oracle, SQL Server, PostgreSQL, or Postgres Plus Advanced Server for the type of publication database. For an Advanced Server Oracle compatible installation, select the Postgres Plus Advanced Server option. For PostgreSQL or an Advanced Server PostgreSQL compatible installation, select the PostgreSQL option.
- **Host.** IP address of the host on which the publication database server is running.
- **Port.** Port on which the publication database server is listening for connections.
- **User.** The publication database user name created in Step 1 of Section 5.1.4.
- **Password.** Password of the database user.
- **Service ID (For Oracle).** Enter the Oracle System Identifier (SID) of the Oracle instance running the publication database if the SID radio button is selected. Enter the net service name of a connect descriptor as defined in the `TNSNAMES.ORA` file if the Service Name radio button is selected. **Note (For Oracle 12c Pluggable Database):** Use the service name.
- **Database (For Postgres or SQL Server).** Enter the Postgres or SQL Server database name.
- **URL Options (For SSL connectivity).** Enter the URL options to establish SSL connectivity to the publication database. See Section 7.11 for information on using SSL connections.
- **Changeset Logging (For Postgres).** Select Table Triggers to use the trigger-based method of synchronization replication. Select WAL Stream to use the log-based method of synchronization replication. See Section 2.2.9 for information on the trigger-based method. See Section 2.2.10 for information on the log-based method.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

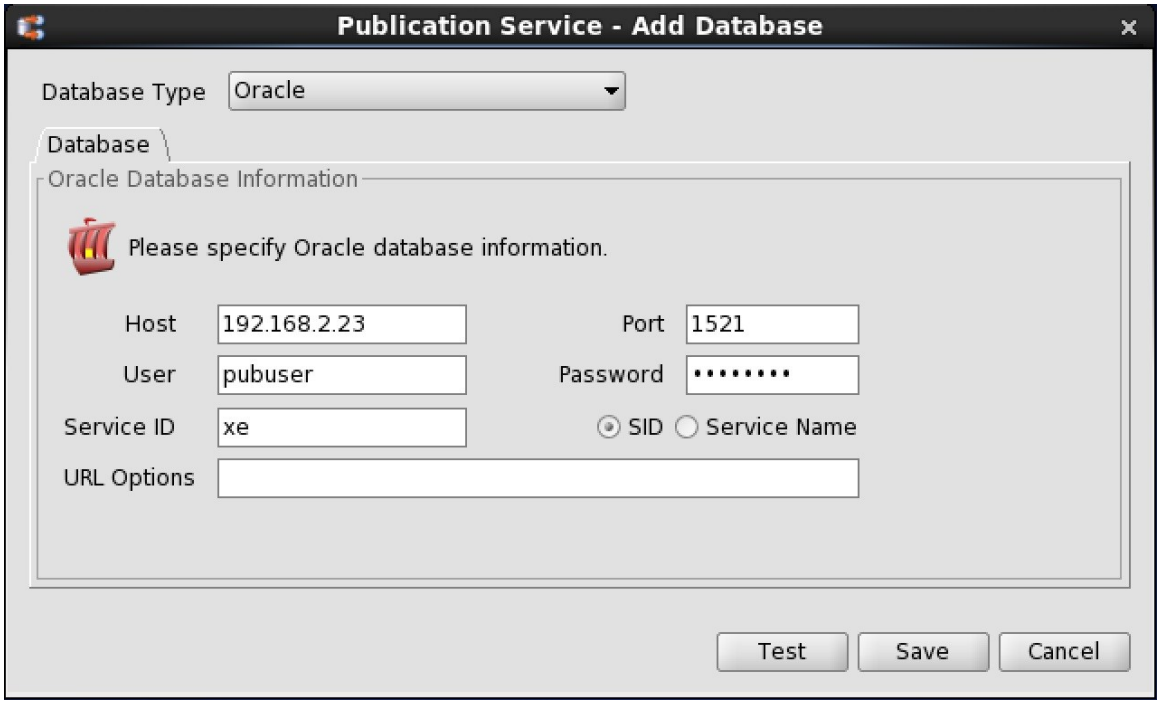


Figure 5-6 - Publication Service - Add Database dialog box

Note: If the controller database is an Oracle or a SQL Server publication database, then a second Oracle or SQL Server publication database cannot be added to create a second single-master replication system. In order for xDB Replication Server to run more than one single-master replication systems consisting of Oracle or SQL Server publication databases, a Postgres publication database must be designated as the controller database. See Section 2.3.1.12 for information on the controller database.

The following is the Publication Service – Add Database dialog box for a Postgres database that shows the Changeset Logging option for selecting either the trigger-based method or the log-based method of synchronization replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

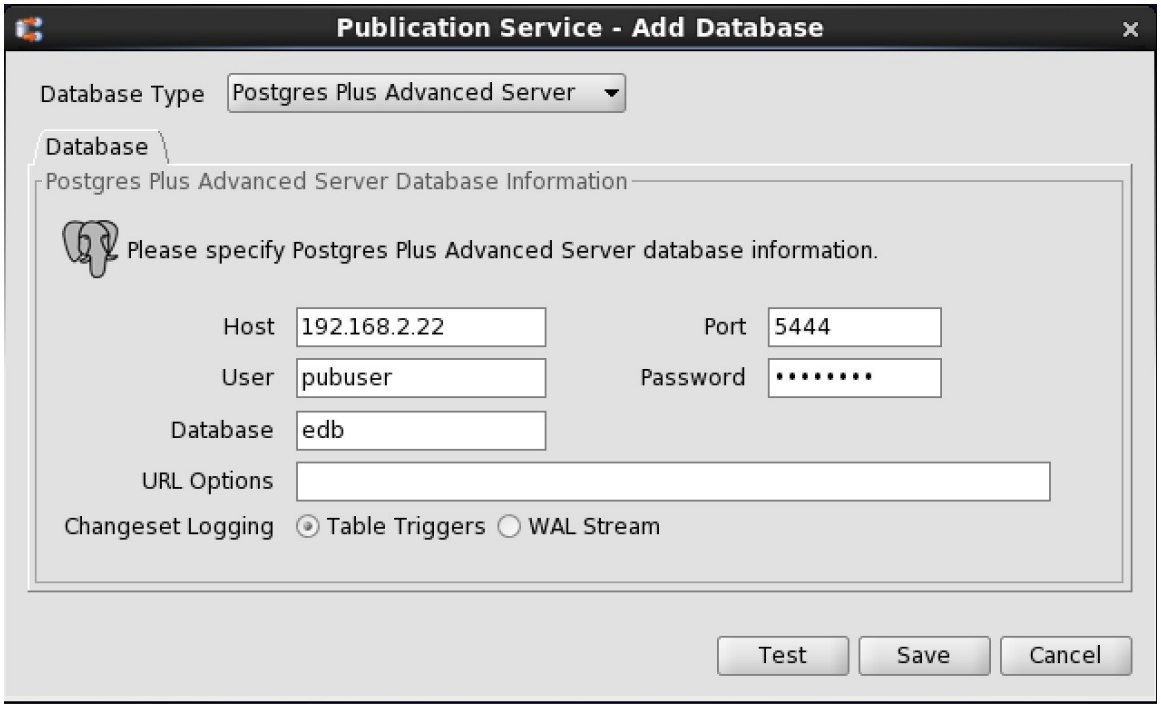


Figure 5-7 - Publication Service - Add Database dialog box for Postgres

Step 4: Click the Test button. If Test Result: Success appears, click the OK button, then click the Save button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

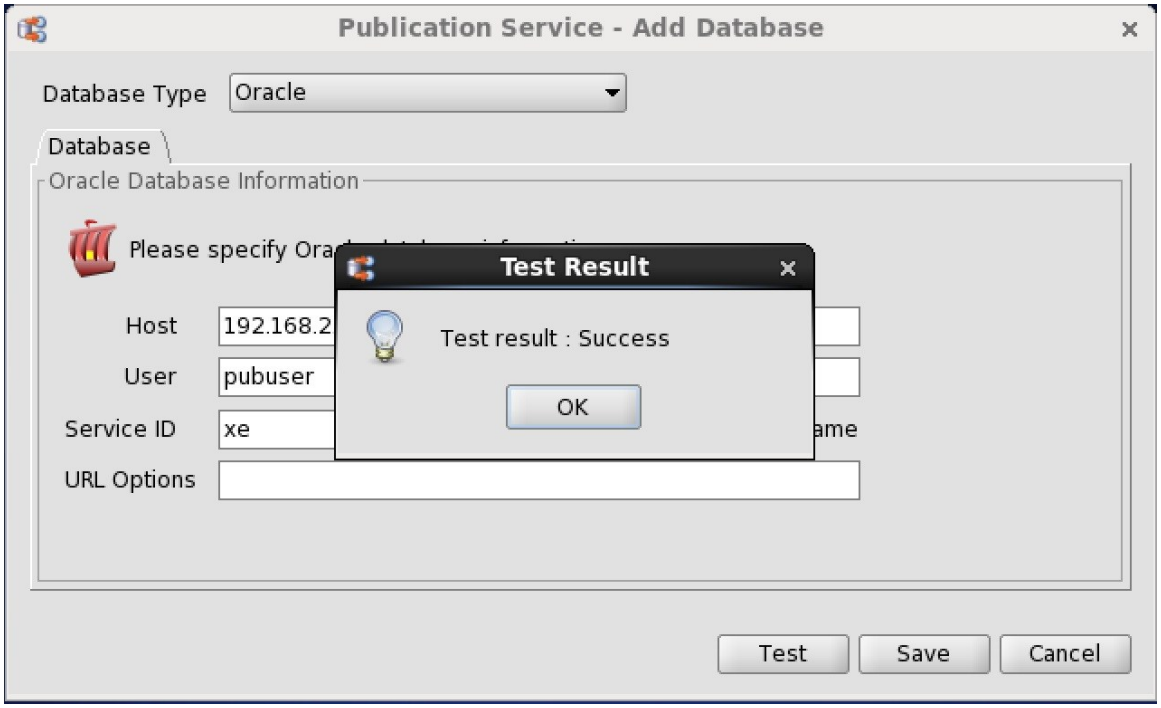


Figure 5-8 - Successful publication database test

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 4.

When the publication database definition is successfully saved, a Publication Database node is added to the replication tree under the Publication Server node.

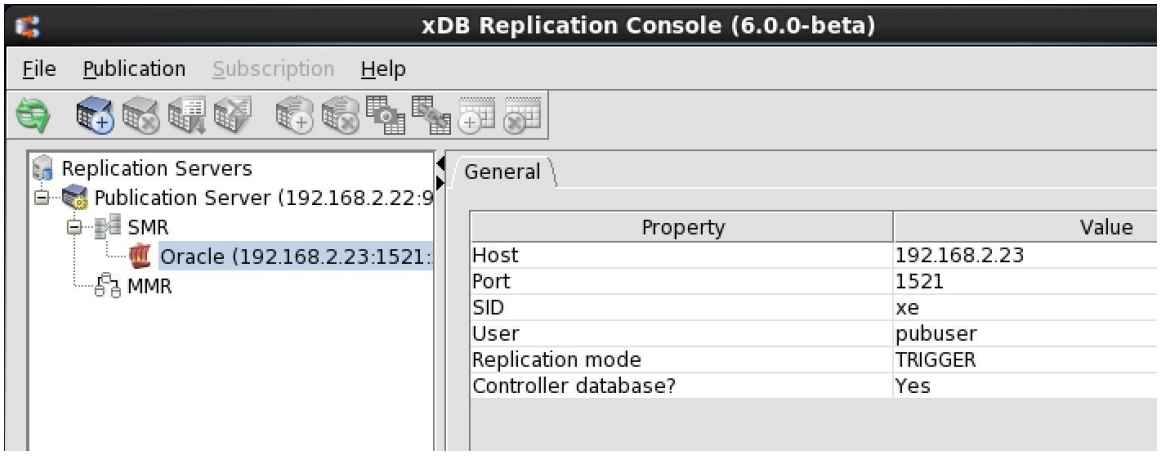


Figure 5-9 – Replication tree after adding a publication database

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For Oracle only: Multiple Oracle databases can be added as publication databases by completing the Add Database dialog box for each database. It is also permissible to add the same Oracle database as two or more distinct publication database definitions if you use different publication database user names for each publication database definition.

For Postgres or SQL Server: Multiple Postgres or SQL Server databases can be added as publication databases by completing the Add Database dialog box for each database. However, unlike Oracle, a given Postgres or SQL Server database can only be added once as a publication database definition.

5.2.3 Adding a Publication

Subordinate to a publication database definition, you create publications that contain tables and views of the database identified in the publication database definition.

Step 1: Select the Publication Database node. From the Publication menu, choose Create Publication. Alternatively, click the secondary mouse button on the Publication Database node and choose Create Publication. The Create Publication dialog box appears.

Step 2: Fill in the following fields under the Create Publication tab:

- **Publication Name.** Enter a name that is unique amongst all publications.
- **Snapshot-only replication.** Check the box if replication is to be done by snapshot only. Tables included in a snapshot-only publication do not require a primary key. Tables included in publications on which synchronization replication is to be used must have primary keys.
- **Publish.** Check the boxes next to the tables that are to be included in the publication. If the Snapshot-Only Replication box is checked, then views appear in the Publish list as well. Alternatively or in addition, click the Use Wildcard Selection button to use wildcard pattern matching for selecting publication tables.
- **Select All.** Check this box if you want to include all tables and views in the Available Tables list in the publication.
- **Use Wildcard Selection.** Click this button to use the wildcard selector to choose tables for the publication. See Section 7.1 for information on the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

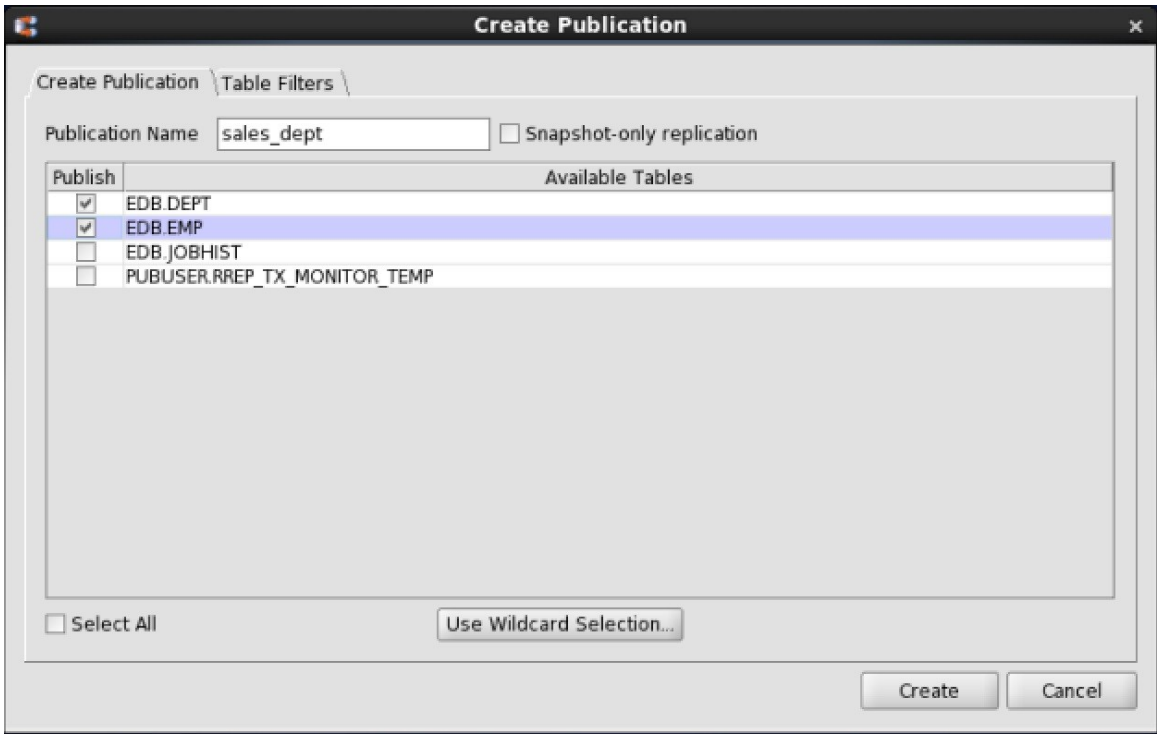


Figure 5-10 - Create Publication dialog box

If you wish to use table filters during replications from this publication, follow the directions in the next step to define the initial set of available table filters, otherwise go on to Step 4.

Step 3 (Optional): *Table filters* consist of a set of filter rules that control the selection criteria for rows replicated to the subscription tables during a snapshot or a synchronization replication.

Note: See Section 2.2.12.3 for table setup requirements for a log-based replication system as well as general restrictions on the use of table filters.

A *filter rule* consists of a *filter name* and a SQL `WHERE` clause (omitting the `WHERE` keyword) called the *filter clause*, which you specify for a table or view that defines the selection criteria for rows that are to be included during a replication.

Multiple filter rules may be defined for each table or view in the publication. If no filter rule is defined for a given table or view, then no filtering can be later enabled on a corresponding subscription table associated with that publication table.

After filter rules have been defined for a publication table or view, you can later choose whether or not to enable those filter rules on any subscription that you associate with that publication in accordance with the following rules.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- At most one filter rule can be enabled on a given table in a given subscription.
- The same filter rule may be enabled on the same given table in several, different subscriptions.
- Different filter rules may be enabled on the same given table but in different subscriptions.

If you want to define table filters on the publication tables or views, click the Table Filters tab. Select the table or view from the Table/View drop-down list for which you wish to add a filter rule. Click the Add Filter button.

In the Filter dialog box, enter a descriptive filter name and the filter clause to select the rows you want to replicate. The filter name and filter clause must meet the following conditions:

- For any given table or view, each filter rule must be assigned a unique filter name.
- For any given table or view, the filter clauses must have different syntaxes (that is, the filtering criteria must be different).

In the following example a filter rule is defined on the `DEPT` table so only rows where the `deptno` column contains `10`, `20`, or `30` are included in replications. All other rows are excluded from replication.

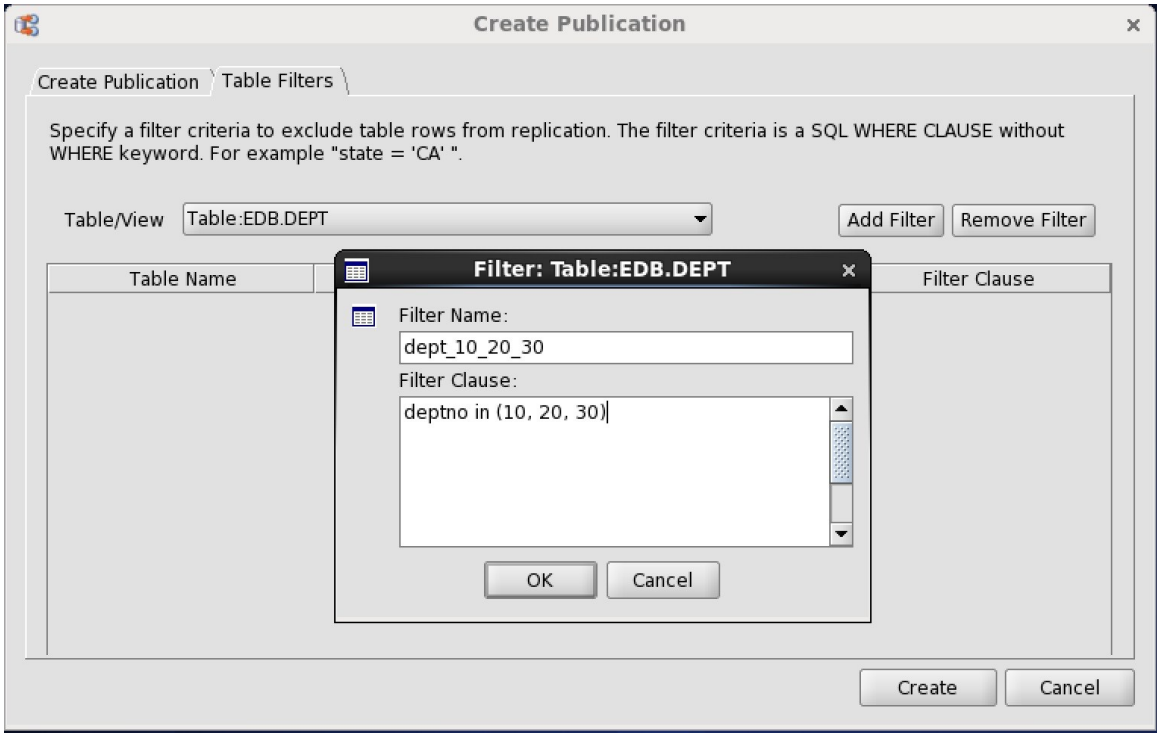


Figure 5-11 - Adding a filter rule for the DEPT table

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows a rule added to the EMP table by choosing EDB.EMP from the Table/View drop-down list and then entering the selection criteria for only rows with deptno containing 10 in the Filter dialog box.

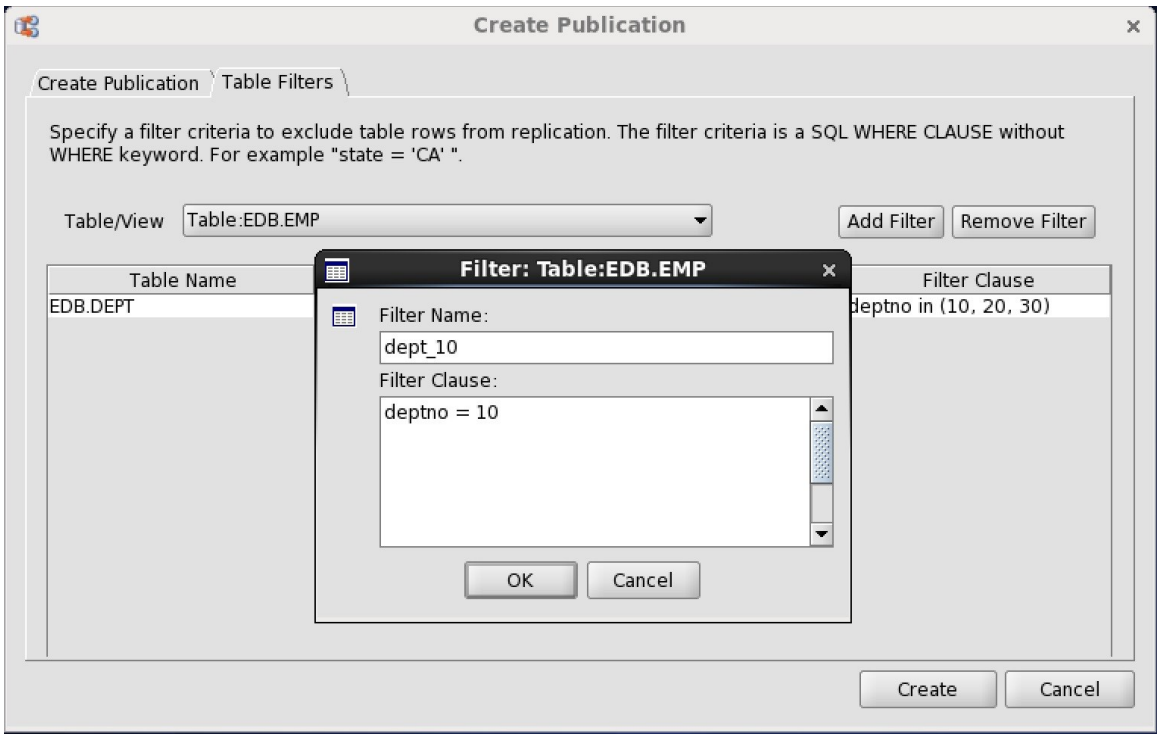


Figure 5-12 - Adding a filter rule for the EMP table

Repeating this process, additional filter rules can be added for the EMP table. The following shows the complete set of available filter rules defined for the DEPT and EMP tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

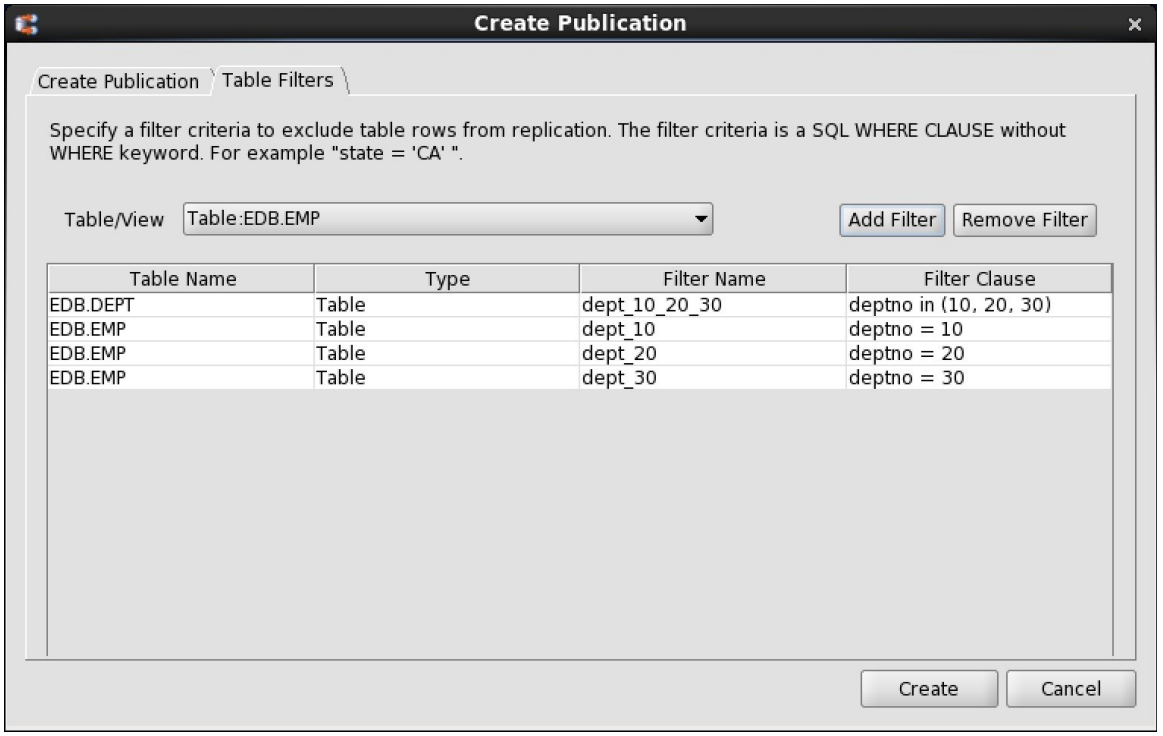


Figure 5-13 - Set of all available filter rules

To remove a filter rule, click the primary mouse button on the filter rule you wish to remove so the entry is highlighted and then click the Remove Filter button.

You may also modify the filter name or filter clause of a filter rule listed in the Table Filters tab by double-clicking on the cell of the filter name or filter clause you wish to change. When the cursor appears in the cell, enter the text for the desired change.

When creating a subscription, you may selectively enable these table filters on the corresponding subscription tables. See Section 5.3.3 for information on creating a subscription.

Step 4: Click the Create button. If Publication Created Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

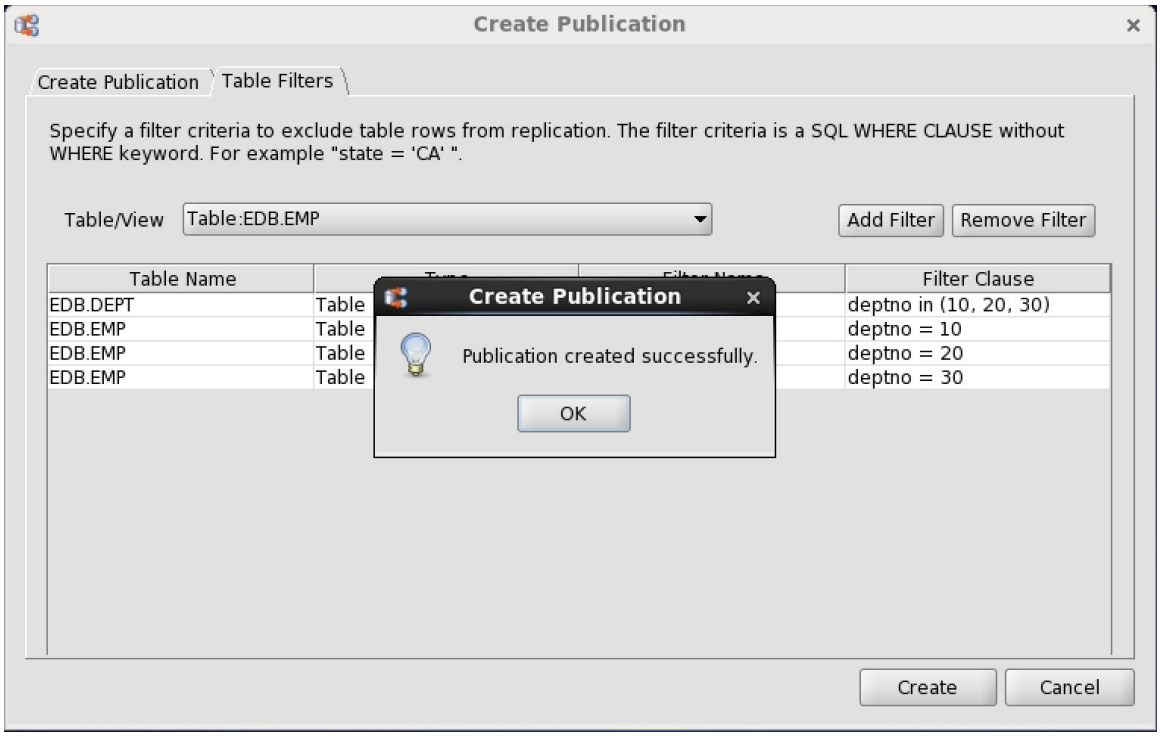


Figure 5-14 - Publication created successfully

Upon successful publication creation, a Publication node is added to the replication tree.

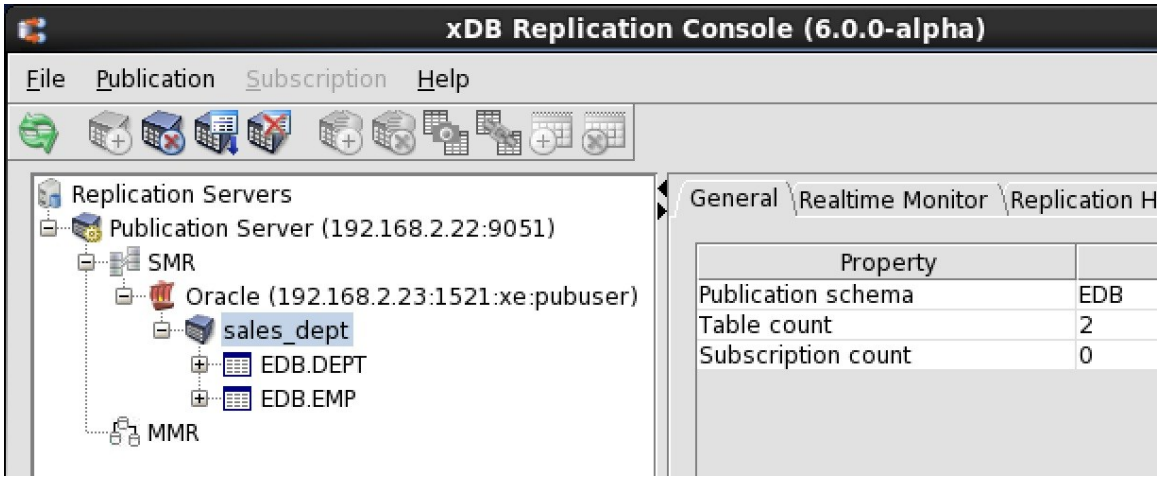


Figure 5-15 – Replication tree after adding a publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.2.4 Control Schema Objects Created for a Publication

After you have added a publication database definition and publications you will find the following control schema objects have been created in addition to your original publication tables and views:

- In the publication database, control schema objects are created to control and manage the xDB replication systems. How the control schema objects are organized under the actual, physical database schemas depends upon the publication database server type (that is, whether it is Oracle, SQL Server, or Postgres).
- If the publication is not a snapshot-only publication, that is synchronization replication is permitted, and synchronization replication is implemented using the trigger-based method, then three triggers and one shadow table are created for each publication table as part of the control schema.
- If the publication is using synchronization replication with the log-based method, then a single trigger is created for each publication table as part of the control schema.

The following sections list the control schema objects found in an Oracle, SQL Server, and Postgres publication database.

Do not manually delete any of these database objects as the replication system control schema will become corrupted.

When you remove publications and publication database definitions using the xDB Replication Console or xDB Replication Server CLI, the control schema objects are deleted during the removal process.

5.2.4.1 Oracle Control Schema Objects

The control schema objects created in the publication database user's schema (that is, the control schema) are shown in the following output:

```
SQL> CONNECT pubuser/password
Connected.
SQL> SET PAGESIZE 9999
SQL> SELECT table_name FROM user_tables ORDER BY table_name;

TABLE_NAME
------------------------------
RREP_LOCK
RREP_MMR_PUB_GROUP
RREP_MMR_TXSET
RREP_PROPERTIES
RREP_PUBLICATION_SUBSCRIPTIONS
RREP_PUBLICATION_TABLES
RREP_TABLES
RREP_TXSET
RREP_TXSET_HEALTH
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
RREP_TXSET_LOG
RREP_TX_MONITOR
RREP_TX_MONITOR_TEMP
RRST_EDB_DEPT
RRST_EDB_EMP
SCH_PUB_BLOB_TRIGGERS
SCH_PUB_CALENDARS
SCH_PUB_CRON_TRIGGERS
SCH_PUB_FIRED_TRIGGERS
SCH_PUB_JOB_DETAILS
SCH_PUB_JOB_LISTENERS
SCH_PUB_LOCKS
SCH_PUB_PAUSED_TRIGGER_GRPS
SCH_PUB_SCHEDULER_STATE
SCH_PUB_SIMPLE_TRIGGERS
SCH_PUB_TRIGGERS
SCH_PUB_TRIGGER_LISTENERS
SCH_SUB_BLOB_TRIGGERS
SCH_SUB_CALENDARS
SCH_SUB_CRON_TRIGGERS
SCH_SUB_FIRED_TRIGGERS
SCH_SUB_JOB_DETAILS
SCH_SUB_JOB_LISTENERS
SCH_SUB_LOCKS
SCH_SUB_PAUSED_TRIGGER_GRPS
SCH_SUB_SCHEDULER_STATE
SCH_SUB_SIMPLE_TRIGGERS
SCH_SUB_TRIGGERS
SCH_SUB_TRIGGER_LISTENERS
XDB_CLEANUP_CONF
XDB_CONFLICTS
XDB_CONFLICTS_OPTIONS
XDB_EVENTS
XDB_EVENTS_STATUS
XDB_MMR_PUB_GROUP
XDB_PUBLICATIONS
XDB_PUBLICATION_FILTER
XDB_PUBLICATION_FILTER_RULE
XDB_PUBLICATION_SUBSCRIPTIONS
XDB_PUBTABLES_IGNOREDCOLS
XDB_PUB_DATABASE
XDB_PUB_REPLOG
XDB_PUB_TABLE_REPLOG
XDB_SUBSCRIPTIONS
XDB_SUBSCRIPTION_TABLES
XDB_SUB_DATABASE
XDB_SUB_SERVERS
XDB_TABLES

57 rows selected.

SQL> SELECT sequence_name FROM user_sequences ORDER BY sequence_name;

SEQUENCE_NAME
------------------------------
RREP_COMMON_SEQ
RREP_TXSET_SEQ
RREP_TX_SEQ

SQL> SELECT DISTINCT name FROM user_source WHERE type = 'PACKAGE';

NAME
------------------------------
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
RREP_PKG

SQL> SELECT trigger_name FROM user_triggers ORDER BY trigger_name;

TRIGGER_NAME
------------------------------
RRPD_EDB_DEPT
RRPD_EDB_EMP
RRPI_EDB_DEPT
RRPI_EDB_EMP
RRPU_EDB_DEPT
RRPU_EDB_EMP
SCH_PUB_BLOB_TRIGGERS_TRIGGER
SCH_PUB_CALENDARS_TRIGGER
SCH_PUB_CRON_TRIGGERS_TRIGGER
SCH_PUB_JOB_DETAILS_TRIGGER
SCH_PUB_JOB_LISTENERS_TRIGGER
SCH_PUB_SIMPLE_TRIGGERS_TRIG
SCH_PUB_TRIGGERS_TRIG
SCH_PUB_TRIGGER_LISTENERS_TRIG
SCH_SUB_BLOB_TRIGGERS_TRIGGER
SCH_SUB_CALENDARS_TRIGGER
SCH_SUB_CRON_TRIGGERS_TRIGGER
SCH_SUB_JOB_DETAILS_TRIGGER
SCH_SUB_JOB_LISTENERS_TRIGGER
SCH_SUB_SIMPLE_TRIGGERS_TRIG
SCH_SUB_TRIGGERS_TRIG
SCH_SUB_TRIGGER_LISTENERS_TRIG
XDB_CLEANUP_CONF_TRIGGER
XDB_CONFLICTS_OPTIONS_TRIGGER
XDB_CONFLICTS_TRIGGER
XDB_MMR_PUB_GROUP_TRIGGER
XDB_PUBLICATIONS_TRIGGER
XDB_PUBLICATION_FILTER_TRIGGER
XDB_PUBLICATION_SUBSCRIPT_TRIG
XDB_PUBLIC_FILTER_RULE_TRIGGER
XDB_PUBTABLES_IGNOREDCOLS_TRIG
XDB_PUB_DATABASE_TRIGGER
XDB_PUB_REPLOG_TRIGGER
XDB_PUB_TABLE_REPLOG_TRIGGER
XDB_SUBSCRIPTIONS_TRIGGER
XDB_SUBSCRIPTION_TABLES_TRIG
XDB_SUB_DATABASE_TRIGGER
XDB_SUB_SERVERS_TRIGGER
XDB_TABLES_TRIGGER

39 rows selected.

SQL> SELECT type_name, typecode FROM user_types;

TYPE_NAME                      TYPECODE
------------------------------ ------------------------------
RREP_SYNCID_ARRAY              COLLECTION
```

Note the following in the preceding output.

- The tables named according to the convention `RRST_schema_table` from the `SELECT` statement on `user_tables` are found only for synchronization publications. In this example, these tables are `RRST_EDB_DEPT` and `RRST_EDB_EMP`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The triggers named according to the convention `RRPD_schema_table`, `RRPI_schema_table`, and `RRPU_schema_table` from the `SELECT` statement on `user_triggers` are found only for synchronization publications. In this example, these triggers are `RRPU_EDB_DEPT`, `RRPI_EDB_DEPT`, `RRPD_EDB_DEPT`, `RRPI_EDB_EMP`, `RRPU_EDB_EMP`, and `RRPD_EDB_EMP`.

The following example shows what the same set of queries would look like if the publication was a snapshot-only publication:

```
SQL> CONNECT pubuser/password
Connected.
SQL> SET PAGESIZE 9999
SQL> SELECT table_name FROM user_tables ORDER BY table_name;

TABLE_NAME
------------------------------
RREP_LOCK
RREP_MMR_PUB_GROUP
RREP_MMR_TXSET
RREP_PROPERTIES
RREP_PUBLICATION_SUBSCRIPTIONS
RREP_PUBLICATION_TABLES
RREP_TABLES
RREP_TXSET
RREP_TXSET_HEALTH
RREP_TXSET_LOG
RREP_TX_MONITOR
RREP_TX_MONITOR_TEMP
SCH_PUB_BLOB_TRIGGERS
SCH_PUB_CALENDARS
SCH_PUB_CRON_TRIGGERS
SCH_PUB_FIRED_TRIGGERS
SCH_PUB_JOB_DETAILS
SCH_PUB_JOB_LISTENERS
SCH_PUB_LOCKS
SCH_PUB_PAUSED_TRIGGER_GRPS
SCH_PUB_SCHEDULER_STATE
SCH_PUB_SIMPLE_TRIGGERS
SCH_PUB_TRIGGERS
SCH_PUB_TRIGGER_LISTENERS
SCH_SUB_BLOB_TRIGGERS
SCH_SUB_CALENDARS
SCH_SUB_CRON_TRIGGERS
SCH_SUB_FIRED_TRIGGERS
SCH_SUB_JOB_DETAILS
SCH_SUB_JOB_LISTENERS
SCH_SUB_LOCKS
SCH_SUB_PAUSED_TRIGGER_GRPS
SCH_SUB_SCHEDULER_STATE
SCH_SUB_SIMPLE_TRIGGERS
SCH_SUB_TRIGGERS
SCH_SUB_TRIGGER_LISTENERS
XDB_CLEANUP_CONF
XDB_CONFLICTS
XDB_CONFLICTS_OPTIONS
XDB_EVENTS
XDB_EVENTS_STATUS
XDB_MMR_PUB_GROUP
XDB_PUBLICATIONS
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
XDB_PUBLICATION_FILTER
XDB_PUBLICATION_FILTER_RULE
XDB_PUBLICATION_SUBSCRIPTIONS
XDB_PUBTABLES_IGNOREDCOLS
XDB_PUB_DATABASE
XDB_PUB_REPLOG
XDB_PUB_TABLE_REPLOG
XDB_SUBSCRIPTIONS
XDB_SUBSCRIPTION_TABLES
XDB_SUB_DATABASE
XDB_SUB_SERVERS
XDB_TABLES

55 rows selected.

SQL> SELECT sequence_name FROM user_sequences ORDER BY sequence_name;

SEQUENCE_NAME
------------------------------
RREP_COMMON_SEQ
RREP_TXSET_SEQ
RREP_TX_SEQ

SQL> SELECT DISTINCT name FROM user_source WHERE type = 'PACKAGE';

NAME
------------------------------
RREP_PKG

SQL> SELECT trigger_name FROM user_triggers ORDER BY trigger_name;

TRIGGER_NAME
------------------------------
SCH_PUB_BLOB_TRIGGERS_TRIGGER
SCH_PUB_CALENDARS_TRIGGER
SCH_PUB_CRON_TRIGGERS_TRIGGER
SCH_PUB_JOB_DETAILS_TRIGGER
SCH_PUB_JOB_LISTENERS_TRIGGER
SCH_PUB_SIMPLE_TRIGGERS_TRIG
SCH_PUB_TRIGGERS_TRIG
SCH_PUB_TRIGGER_LISTENERS_TRIG
SCH_SUB_BLOB_TRIGGERS_TRIGGER
SCH_SUB_CALENDARS_TRIGGER
SCH_SUB_CRON_TRIGGERS_TRIGGER
SCH_SUB_JOB_DETAILS_TRIGGER
SCH_SUB_JOB_LISTENERS_TRIGGER
SCH_SUB_SIMPLE_TRIGGERS_TRIG
SCH_SUB_TRIGGERS_TRIG
SCH_SUB_TRIGGER_LISTENERS_TRIG
XDB_CLEANUP_CONF_TRIGGER
XDB_CONFLICTS_OPTIONS_TRIGGER
XDB_CONFLICTS_TRIGGER
XDB_MMR_PUB_GROUP_TRIGGER
XDB_PUBLICATIONS_TRIGGER
XDB_PUBLICATION_FILTER_TRIGGER
XDB_PUBLICATION_SUBSCRIPT_TRIG
XDB_PUBLIC_FILTER_RULE_TRIGGER
XDB_PUBTABLES_IGNOREDCOLS_TRIG
XDB_PUB_DATABASE_TRIGGER
XDB_PUB_REPLOG_TRIGGER
XDB_PUB_TABLE_REPLOG_TRIGGER
XDB_SUBSCRIPTIONS_TRIGGER
XDB_SUBSCRIPTION_TABLES_TRIG
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
XDB_SUB_DATABASE_TRIGGER
XDB_SUB_SERVERS_TRIGGER
XDB_TABLES_TRIGGER

33 rows selected.

SQL> SELECT type_name, typecode FROM user_types;

TYPE_NAME                      TYPECODE
------------------------------ ------------------------------
RREP_SYNCID_ARRAY              COLLECTION
```

Note: The `RREP_SYNCID_ARRAY` collection type is found only in an Oracle publication database.

5.2.4.2 SQL Server Control Schema Objects

Most of the control schema objects are created in schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`. Additional control schema objects are created in the schema you chose in Step 5 of Section 5.1.4.2. The following examples assume the schema of your choosing is `pubuser`. The publication tables are `dept` and `emp` located in the `edb` schema.

The following query lists the control schema objects located in the aforementioned schemas:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> SELECT s.name + '.' + o.name "Object Name", o.type_desc "Object Type"
2>   FROM sys.objects o,
3>        sys.schemas s
4>   WHERE s.name IN ('_edb_replicator_pub', '_edb_replicator_sub',
5>                    '_edb_scheduler', 'pubuser')
6>     AND o.type IN ('U','P','FN')
7>     AND o.schema_id = s.schema_id
8>   ORDER BY 1, 2;
9> GO
Object Name                                        Object Type
-------------------------------------------------- --------------------
_edb_replicator_pub.nextval                        SQL_STORED_PROCEDURE
_edb_replicator_pub.rrep_common_seq                USER_TABLE
_edb_replicator_pub.rrep_lock                      USER_TABLE
_edb_replicator_pub.rrep_mmr_pub_group             USER_TABLE
_edb_replicator_pub.rrep_mmr_txset                 USER_TABLE
_edb_replicator_pub.rrep_properties                USER_TABLE
_edb_replicator_pub.rrep_publication_subscriptions USER_TABLE
_edb_replicator_pub.rrep_publication_tables        USER_TABLE
_edb_replicator_pub.rrep_tables                    USER_TABLE
_edb_replicator_pub.rrep_tx_monitor                USER_TABLE
_edb_replicator_pub.rrep_tx_seq                    USER_TABLE
_edb_replicator_pub.rrep_txset                     USER_TABLE
_edb_replicator_pub.rrep_txset_health              USER_TABLE
_edb_replicator_pub.rrep_txset_log                 USER_TABLE
_edb_replicator_pub.rrep_txset_seq                 USER_TABLE
_edb_replicator_pub.sp_createsequence              SQL_STORED_PROCEDURE
_edb_replicator_pub.sp_dropsequence                SQL_STORED_PROCEDURE
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
_edb_replicator_pub.xdb_cleanup_conf                USER_TABLE
_edb_replicator_pub.xdb_conflicts                   USER_TABLE
_edb_replicator_pub.xdb_conflicts_options           USER_TABLE
_edb_replicator_pub.xdb_events                      USER_TABLE
_edb_replicator_pub.xdb_events_status               USER_TABLE
_edb_replicator_pub.xdb_mmr_pub_group               USER_TABLE
_edb_replicator_pub.xdb_pub_database                USER_TABLE
_edb_replicator_pub.xdb_pub_replog                  USER_TABLE
_edb_replicator_pub.xdb_pub_table_replog            USER_TABLE
_edb_replicator_pub.xdb_publication_filter          USER_TABLE
_edb_replicator_pub.xdb_publication_filter_rule     USER_TABLE
_edb_replicator_pub.xdb_publication_subscriptions   USER_TABLE
_edb_replicator_pub.xdb_publications                USER_TABLE
_edb_replicator_pub.xdb_pubtables_ignoredcols       USER_TABLE
_edb_replicator_pub.xdb_sub_servers                 USER_TABLE
_edb_replicator_sub.rrep_common_seq                 USER_TABLE
_edb_replicator_sub.xdb_sub_database                USER_TABLE
_edb_replicator_sub.xdb_subscription_tables         USER_TABLE
_edb_replicator_sub.xdb_subscriptions               USER_TABLE
_edb_replicator_sub.xdb_tables                      USER_TABLE
_edb_scheduler.sch_pub_BLOB_TRIGGERS                USER_TABLE
_edb_scheduler.sch_pub_CALENDARS                    USER_TABLE
_edb_scheduler.sch_pub_CRON_TRIGGERS                USER_TABLE
_edb_scheduler.sch_pub_FIRED_TRIGGERS               USER_TABLE
_edb_scheduler.sch_pub_JOB_DETAILS                  USER_TABLE
_edb_scheduler.sch_pub_JOB_LISTENERS                USER_TABLE
_edb_scheduler.sch_pub_LOCKS                        USER_TABLE
_edb_scheduler.sch_pub_PAUSED_TRIGGER_GRPS          USER_TABLE
_edb_scheduler.sch_pub_SCHEDULER_STATE              USER_TABLE
_edb_scheduler.sch_pub_SIMPLE_TRIGGERS              USER_TABLE
_edb_scheduler.sch_pub_TRIGGER_LISTENERS            USER_TABLE
_edb_scheduler.sch_pub_TRIGGERS                     USER_TABLE
_edb_scheduler.sch_sub_BLOB_TRIGGERS                USER_TABLE
_edb_scheduler.sch_sub_CALENDARS                    USER_TABLE
_edb_scheduler.sch_sub_CRON_TRIGGERS                USER_TABLE
_edb_scheduler.sch_sub_FIRED_TRIGGERS               USER_TABLE
_edb_scheduler.sch_sub_JOB_DETAILS                  USER_TABLE
_edb_scheduler.sch_sub_JOB_LISTENERS                USER_TABLE
_edb_scheduler.sch_sub_LOCKS                        USER_TABLE
_edb_scheduler.sch_sub_PAUSED_TRIGGER_GRPS          USER_TABLE
_edb_scheduler.sch_sub_SCHEDULER_STATE              USER_TABLE
_edb_scheduler.sch_sub_SIMPLE_TRIGGERS              USER_TABLE
_edb_scheduler.sch_sub_TRIGGER_LISTENERS            USER_TABLE
_edb_scheduler.sch_sub_TRIGGERS                     USER_TABLE
pubuser.CleanupShadowTables                         SQL_STORED_PROCEDURE
pubuser.ConfigureCleanUpJob                         SQL_STORED_PROCEDURE
pubuser.ConfigureCreateTxSetJob                     SQL_STORED_PROCEDURE
pubuser.CreateMultiTxSet                            SQL_STORED_PROCEDURE
pubuser.CreateTableLogTrigger                       SQL_STORED_PROCEDURE
pubuser.CreateTxSet                                 SQL_STORED_PROCEDURE
pubuser.CreateTxSet_old                             SQL_STORED_PROCEDURE
pubuser.CreateUniTxSet                              SQL_STORED_PROCEDURE
pubuser.GetNewTxsCount                              SQL_STORED_PROCEDURE
pubuser.getPackageVersionNumber                     SQL_SCALAR_FUNCTION
pubuser.JobCleanup                                  SQL_STORED_PROCEDURE
pubuser.JobCreateTxSet                              SQL_STORED_PROCEDURE
pubuser.LoadPubTableList                            SQL_STORED_PROCEDURE
pubuser.RemoveCleanupJob                            SQL_STORED_PROCEDURE
pubuser.RemoveCreateTxSetJob                        SQL_STORED_PROCEDURE
pubuser.rrst_edb_dept                               USER_TABLE
pubuser.rrst_edb_emp                                USER_TABLE

(78 rows affected)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note (For SQL Server 2012, 2014): The following database objects from the preceding list are no longer created as part of the control schema when the publication database is SQL Server 2012 or 2014:

```
Object Name                                        Object Type
-------------------------------------------------- --------------------
_edb_replicator_pub.nextval                        SQL_STORED_PROCEDURE
_edb_replicator_pub.rrep_common_seq                USER_TABLE
_edb_replicator_pub.rrep_tx_seq                    USER_TABLE
_edb_replicator_pub.rrep_txset_seq                 USER_TABLE
_edb_replicator_pub.sp_createsequence              SQL_STORED_PROCEDURE
_edb_replicator_pub.sp_dropsequence                SQL_STORED_PROCEDURE
_edb_replicator_sub.rrep_common_seq                USER_TABLE
```

SQL Server versions 2012 and 2014 support creation of sequence objects that can now perform the functionality previously provided by the preceding list of objects. The following are the sequence objects that are now used when the publication database is SQL Server 2012 or 2014:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> SELECT s.name + '.' + o.name "Object Name", o.type_desc "Object Type"
2>   FROM sys.objects o,
3>        sys.schemas s
4>   WHERE s.name IN ('_edb_replicator_pub', '_edb_replicator_sub',
5>                    '_edb_scheduler', 'pubuser')
6>     AND o.type IN ('SO')
7>     AND o.schema_id = s.schema_id
8>   ORDER BY 1, 2;
9> GO
Object Name                                        Object Type
-------------------------------------------------- --------------------
_edb_replicator_pub.rrep_common_seq                SEQUENCE_OBJECT
_edb_replicator_pub.rrep_tx_seq                    SEQUENCE_OBJECT
_edb_replicator_pub.rrep_txset_seq                 SEQUENCE_OBJECT

(3 rows affected)
```

The following is a continuation of the list of control schema objects for all SQL Server versions:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> SELECT s.name + '.' + o.name "Trigger Name", o.type_desc "Object Type"
2>   FROM sys.objects o,
3>        sys.schemas s
4>   WHERE s.name IN ('_edb_replicator_pub', '_edb_replicator_sub',
5>                    '_edb_scheduler', 'pubuser')
6>     AND o.type = 'TR'
7>     AND o.schema_id = s.schema_id
8>   ORDER BY 1;
9> GO
Trigger Name                                                Object Type
----------------------------------------------------------- -------------
_edb_replicator_pub.xdb_cleanup_conf_trigger                SQL_TRIGGER
_edb_replicator_pub.xdb_conflicts_options_trigger           SQL_TRIGGER
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
_edb_replicator_pub.xdb_conflicts_trigger                    SQL_TRIGGER
_edb_replicator_pub.xdb_mmr_pub_group_trigger                SQL_TRIGGER
_edb_replicator_pub.xdb_pub_database_trigger                 SQL_TRIGGER
_edb_replicator_pub.xdb_pub_replog_trigger                   SQL_TRIGGER
_edb_replicator_pub.xdb_pub_table_replog_trigger             SQL_TRIGGER
_edb_replicator_pub.xdb_public_filter_rule_trigger           SQL_TRIGGER
_edb_replicator_pub.xdb_publication_filter_trigger           SQL_TRIGGER
_edb_replicator_pub.xdb_publication_subscription_triggers    SQL_TRIGGER
_edb_replicator_pub.xdb_publications_trigger                 SQL_TRIGGER
_edb_replicator_pub.xdb_pubtables_ignoredcols_trig           SQL_TRIGGER
_edb_replicator_pub.xdb_sub_servers_trigger                  SQL_TRIGGER
_edb_replicator_sub.xdb_sub_database_trigger                 SQL_TRIGGER
_edb_replicator_sub.xdb_subscription_tables_trig             SQL_TRIGGER
_edb_replicator_sub.xdb_subscriptions_trigger                SQL_TRIGGER
_edb_replicator_sub.xdb_tables_trigger                       SQL_TRIGGER
_edb_scheduler.sch_pub_blob_triggers_trigger                 SQL_TRIGGER
_edb_scheduler.sch_pub_calendars_trigger                     SQL_TRIGGER
_edb_scheduler.sch_pub_cron_triggers_trigger                 SQL_TRIGGER
_edb_scheduler.sch_pub_job_details_trigger                   SQL_TRIGGER
_edb_scheduler.sch_pub_job_listeners_trigger                 SQL_TRIGGER
_edb_scheduler.sch_pub_simple_triggers_trigger               SQL_TRIGGER
_edb_scheduler.sch_pub_trigger_listeners_trigger             SQL_TRIGGER
_edb_scheduler.sch_pub_triggers_trigger                      SQL_TRIGGER
_edb_scheduler.sch_sub_blob_triggers_trigger                 SQL_TRIGGER
_edb_scheduler.sch_sub_calendars_trigger                     SQL_TRIGGER
_edb_scheduler.sch_sub_cron_triggers_trigger                 SQL_TRIGGER
_edb_scheduler.sch_sub_job_details_trigger                   SQL_TRIGGER
_edb_scheduler.sch_sub_job_listeners_trigger                 SQL_TRIGGER
_edb_scheduler.sch_sub_simple_triggers_trigger               SQL_TRIGGER
_edb_scheduler.sch_sub_trigger_listeners_trigger             SQL_TRIGGER
_edb_scheduler.sch_sub_triggers_trigger                      SQL_TRIGGER

(33 rows affected)
```

For non-snapshot only publication tables, triggers are created that reside in the schema containing the publication tables as shown by the following:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> SELECT s.name + '.' + o.name "Trigger Name"
2>   FROM sys.objects o,
3>        sys.schemas s
4>   WHERE s.name = 'edb'
5>     AND o.type = 'TR'
6>     AND o.name LIKE 'rr%'
7>     AND o.schema_id = s.schema_id
8>   ORDER BY 1;
9> GO
Trigger Name
------------------------------------
edb.rrpd_edb_dept
edb.rrpd_edb_emp
edb.rrpi_edb_dept
edb.rrpi_edb_emp
edb.rrpu_edb_dept
edb.rrpu_edb_emp

(6 rows affected)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Finally, some jobs are created in the `msdb` database after the subscription is created as shown by the following:

```
1> USE msdb;
2> GO
Changed database context to 'msdb'.
1> SELECT j.name "Job Name"
2>   FROM msdb.dbo.sysjobs j,
3>        master.dbo.syslogins l
4>   WHERE l.name = 'pubuser'
5>     AND j.name LIKE 'rrep%'
6>     AND j.owner_sid = l.sid
7>   ORDER BY 1;
8> GO
Job Name
-----------------------------------
rrep_cleanup_job_edb
rrep_txset_job_edb

(2 rows affected)
```

5.2.4.3 Postgres Control Schema Objects

The control schema objects are created in three schemas named `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`.

The control schema objects contained in `_edb_replicator_pub` are shown by the following:

```
edb=# SET search_path TO _edb_replicator_pub;
SET
edb=# \dt
                        List of relations
       Schema        |             Name              | Type  |  Owner
---------------------+-------------------------------+-------+---------
 _edb_replicator_pub | rrep_lock                     | table | pubuser
 _edb_replicator_pub | rrep_mmr_pub_group            | table | pubuser
 _edb_replicator_pub | rrep_mmr_txset                | table | pubuser
 _edb_replicator_pub | rrep_properties               | table | pubuser
 _edb_replicator_pub | rrep_publication_subscriptions | table | pubuser
 _edb_replicator_pub | rrep_publication_tables       | table | pubuser
 _edb_replicator_pub | rrep_tables                   | table | pubuser
 _edb_replicator_pub | rrep_tx_monitor               | table | pubuser
 _edb_replicator_pub | rrep_txset                    | table | pubuser
 _edb_replicator_pub | rrep_txset_health             | table | pubuser
 _edb_replicator_pub | rrep_txset_log                | table | pubuser
 _edb_replicator_pub | rrep_wal_events_queue         | table | pubuser
 _edb_replicator_pub | rrst_edb_dept                 | table | pubuser
 _edb_replicator_pub | rrst_edb_emp                  | table | pubuser
 _edb_replicator_pub | xdb_cleanup_conf              | table | pubuser
 _edb_replicator_pub | xdb_conflicts                 | table | pubuser
 _edb_replicator_pub | xdb_conflicts_options         | table | pubuser
 _edb_replicator_pub | xdb_events                    | table | pubuser
 _edb_replicator_pub | xdb_events_status             | table | pubuser
 _edb_replicator_pub | xdb_mmr_pub_group             | table | pubuser
 _edb_replicator_pub | xdb_pub_database              | table | pubuser
 _edb_replicator_pub | xdb_pub_replog                | table | pubuser
 _edb_replicator_pub | xdb_pub_table_replog          | table | pubuser
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
 _edb_replicator_pub | xdb_publication_filter         | table | pubuser
 _edb_replicator_pub | xdb_publication_filter_rule    | table | pubuser
 _edb_replicator_pub | xdb_publication_subscriptions  | table | pubuser
 _edb_replicator_pub | xdb_publications               | table | pubuser
 _edb_replicator_pub | xdb_pubtables_ignoredcols      | table | pubuser
 _edb_replicator_pub | xdb_sub_servers                | table | pubuser
(29 rows)

edb=# \ds
                    List of relations
       Schema        |      Name       |   Type   |  Owner
---------------------+-----------------+----------+---------
 _edb_replicator_pub | rrep_common_seq | sequence | pubuser
 _edb_replicator_pub | rrep_tx_seq     | sequence | pubuser
 _edb_replicator_pub | rrep_txset_seq  | sequence | pubuser
(3 rows)

edb=# SELECT nspname, pkgname FROM edb_package pk, pg_namespace ns
edb-#   WHERE nspname IN ('_edb_replicator_pub', '_edb_replicator_sub')
edb-#     AND pk.pkgnamespace = ns.oid;
       nspname       | pkgname
---------------------+----------
 _edb_replicator_pub | rrep_pkg
(1 row)

edb=# SELECT nspname, funname, typname FROM pg_function fn, pg_namespace ns,
edb-#     pg_type ty
edb-#   WHERE nspname = '_edb_replicator_pub'
edb-#     AND ns.oid = fn.funnamespace
edb-#     AND ty.oid = fn.funrettype
edb-#   ORDER BY typname, funname;
       nspname       |                   funname                    | typname
---------------------+----------------------------------------------+--------
-
 _edb_replicator_pub | capturetruncateevent                         | trigger
 _edb_replicator_pub | erep_filter_rule_delete_trigger_tgfunc       | trigger
 _edb_replicator_pub | erep_pub_database_trigger_tgfunc             | trigger
 _edb_replicator_pub | erep_publication_delete_trigger_tgfunc       | trigger
 _edb_replicator_pub | xdb_cleanup_conf_trigger_tgfunc              | trigger
 _edb_replicator_pub | xdb_conflicts_options_trigger_tgfunc         | trigger
 _edb_replicator_pub | xdb_conflicts_trigger_tgfunc                 | trigger
 _edb_replicator_pub | xdb_mmr_pub_group_trigger_tgfunc             | trigger
 _edb_replicator_pub | xdb_pub_database_trigger_tgfunc              | trigger
 _edb_replicator_pub | xdb_pub_replog_trigger_tgfunc                | trigger
 _edb_replicator_pub | xdb_pub_table_replog_trigger_tgfunc          | trigger
 _edb_replicator_pub | xdb_publication_filter_rule_trigger_tgfunc   | trigger
 _edb_replicator_pub | xdb_publication_filter_trigger_tgfunc        | trigger
 _edb_replicator_pub | xdb_publication_subscriptions_trigger_tgfunc | trigger
 _edb_replicator_pub | xdb_publications_trigger_tgfunc              | trigger
 _edb_replicator_pub | xdb_pubtables_ignoredcols_trigger_tgfunc     | trigger
 _edb_replicator_pub | xdb_sub_servers_trigger_tgfunc               | trigger
 _edb_replicator_pub | getpackageversionnumber                      | varchar
(18 rows)
```

The control schema objects contained in `_edb_replicator_sub` are shown by the following:

```
edb=# SET search_path TO _edb_replicator_sub;
SET
edb=# \dt
                      List of relations
       Schema        |          Name           | Type  |  Owner
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
--------------------+------------------------+-------+---------
 _edb_replicator_sub | xdb_sub_database       | table | pubuser
 _edb_replicator_sub | xdb_subscription_tables | table | pubuser
 _edb_replicator_sub | xdb_subscriptions      | table | pubuser
 _edb_replicator_sub | xdb_tables             | table | pubuser
(4 rows)

edb=# \ds
                   List of relations
       Schema        |      Name       |   Type   |  Owner
--------------------+-----------------+----------+---------
 _edb_replicator_sub | rrep_common_seq | sequence | pubuser
(1 row)

edb=# SELECT nspname, funname, typname FROM pg_function fn, pg_namespace ns,
edb-#     pg_type ty
edb-#   WHERE nspname = '_edb_replicator_sub'
edb-#     AND ns.oid = fn.funnamespace
edb-#     AND ty.oid = fn.funrettype
edb-#   ORDER BY typname, funname;
       nspname       |                funname                 | typname
--------------------+----------------------------------------+---------
 _edb_replicator_sub | xdb_sub_database_trigger_tgfunc        | trigger
 _edb_replicator_sub | xdb_subscription_tables_trigger_tgfunc | trigger
 _edb_replicator_sub | xdb_subscriptions_trigger_tgfunc       | trigger
 _edb_replicator_sub | xdb_tables_trigger_tgfunc              | trigger
(4 rows)
```

The control schema objects contained in `_edb_scheduler` are shown by the following:

```
edb=# SET search_path TO _edb_scheduler;
SET
edb=# \dt
                     List of relations
     Schema     |            Name             | Type  |  Owner
----------------+-----------------------------+-------+---------
 _edb_scheduler | sch_pub_blob_triggers       | table | pubuser
 _edb_scheduler | sch_pub_calendars           | table | pubuser
 _edb_scheduler | sch_pub_cron_triggers       | table | pubuser
 _edb_scheduler | sch_pub_fired_triggers      | table | pubuser
 _edb_scheduler | sch_pub_job_details         | table | pubuser
 _edb_scheduler | sch_pub_job_listeners       | table | pubuser
 _edb_scheduler | sch_pub_locks               | table | pubuser
 _edb_scheduler | sch_pub_paused_trigger_grps | table | pubuser
 _edb_scheduler | sch_pub_scheduler_state     | table | pubuser
 _edb_scheduler | sch_pub_simple_triggers     | table | pubuser
 _edb_scheduler | sch_pub_trigger_listeners   | table | pubuser
 _edb_scheduler | sch_pub_triggers            | table | pubuser
 _edb_scheduler | sch_sub_blob_triggers       | table | pubuser
 _edb_scheduler | sch_sub_calendars           | table | pubuser
 _edb_scheduler | sch_sub_cron_triggers       | table | pubuser
 _edb_scheduler | sch_sub_fired_triggers      | table | pubuser
 _edb_scheduler | sch_sub_job_details         | table | pubuser
 _edb_scheduler | sch_sub_job_listeners       | table | pubuser
 _edb_scheduler | sch_sub_locks               | table | pubuser
 _edb_scheduler | sch_sub_paused_trigger_grps | table | pubuser
 _edb_scheduler | sch_sub_scheduler_state     | table | pubuser
 _edb_scheduler | sch_sub_simple_triggers     | table | pubuser
 _edb_scheduler | sch_sub_trigger_listeners   | table | pubuser
 _edb_scheduler | sch_sub_triggers            | table | pubuser
(24 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
edb=# SELECT nspname, funname, typname FROM pg_function fn, pg_namespace ns,
edb-#     pg_type ty
edb-#   WHERE nspname = '_edb_scheduler'
edb-#     AND ns.oid = fn.funnamespace
edb-#     AND ty.oid = fn.funrettype
edb-#   ORDER BY typname, funname;
    nspname     |                 funname                  | typname
----------------+------------------------------------------+---------
 _edb_scheduler | sch_pub_blob_triggers_trigger_tgfunc     | trigger
 _edb_scheduler | sch_pub_calendars_trigger_tgfunc         | trigger
 _edb_scheduler | sch_pub_cron_triggers_trigger_tgfunc     | trigger
 _edb_scheduler | sch_pub_job_details_trigger_tgfunc       | trigger
 _edb_scheduler | sch_pub_job_listeners_trigger_tgfunc     | trigger
 _edb_scheduler | sch_pub_simple_triggers_trigger_tgfunc   | trigger
 _edb_scheduler | sch_pub_trigger_listeners_trigger_tgfunc | trigger
 _edb_scheduler | sch_pub_triggers_trigger_tgfunc          | trigger
 _edb_scheduler | sch_sub_blob_triggers_trigger_tgfunc     | trigger
 _edb_scheduler | sch_sub_calendars_trigger_tgfunc         | trigger
 _edb_scheduler | sch_sub_cron_triggers_trigger_tgfunc     | trigger
 _edb_scheduler | sch_sub_job_details_trigger_tgfunc       | trigger
 _edb_scheduler | sch_sub_job_listeners_trigger_tgfunc     | trigger
 _edb_scheduler | sch_sub_simple_triggers_trigger_tgfunc   | trigger
 _edb_scheduler | sch_sub_trigger_listeners_trigger_tgfunc | trigger
 _edb_scheduler | sch_sub_triggers_trigger_tgfunc          | trigger
(16 rows)
```

In addition, triggers and trigger functions are created in the schema containing the publication tables if the trigger-based method of synchronization replication is used.

```
edb=# SET search_path TO edb;
SET
edb=# \df rr*
                                List of functions
 Schema |         Name          | Result data type | Argument data types |  Type
--------+-----------------------+------------------+---------------------+---------
 edb    | rrpd_edb_dept_tgfunc  | trigger          |                     | trigger
 edb    | rrpd_edb_emp_tgfunc   | trigger          |                     | trigger
 edb    | rrpi_edb_dept_tgfunc  | trigger          |                     | trigger
 edb    | rrpi_edb_emp_tgfunc   | trigger          |                     | trigger
 edb    | rrpu_edb_dept_tgfunc  | trigger          |                     | trigger
 edb    | rrpu_edb_emp_tgfunc   | trigger          |                     | trigger
(6 rows)
```

If the log-based method of synchronization replication is used, the following triggers are created on the publication tables:

```
edb=# SELECT t.tgname AS "Trigger Name", c.relname AS "Table Name",
edb-#        f.funname AS "Trigger Function"
edb-#   FROM pg_trigger t, pg_function f, pg_class c
edb-#   WHERE tgname LIKE 'rrpt%'
edb-#     AND t.tgfoid = f.oid
edb-#     AND t.tgrelid = c.oid
edb-#   ORDER BY t.tgname;
 Trigger Name  | Table Name |   Trigger Function
---------------+------------+----------------------
 rrpt_edb_dept | dept       | capturetruncateevent
 rrpt_edb_emp  | emp        | capturetruncateevent
(2 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

These triggers are used to support synchronization replication of the `TRUNCATE` command when the log-based method is used.

5.3 Creating a Subscription

Creating your first subscription requires the following steps:

- Registering the subscription server
- Adding the subscription database
- Creating a subscription by choosing the publication to which to subscribe

Multiple subscriptions can be created in a subscription database. More than one subscription can also be created to subscribe against the same publication.

5.3.1 Registering a Subscription Server

When you register a subscription server, you are identifying the network location, admin user name, and password of a specific, running, subscription server instance that you want to use to manage all aspects of the subscriptions you will be creating subordinate to it.

It is important that you record the login information for the subscription server as you must always use this same subscription server instance to manage all subscriptions created subordinate to it as represented in the xDB Replication Console replication tree.

Step 1: Start the subscription server if it is not already running. Repeat the same process as in Step 1 of Section 5.2.1.

Note: If you are using Oracle publication or subscription databases, and the subscription server has not been restarted since copying the Oracle JDBC driver to the `lib/jdbc` subdirectory of your xDB Replication Server installation, you must restart the subscription server.

For Linux only: Use the `systemctl` command for CentOS 7 or RHEL 7, and the `service` command for previous Linux versions to start, stop, or restart `edb-xdbsubserver` for the subscription server. See Section 5.2.1 for information on how these commands are used.

For Windows only: Open Control Panel, System and Security, Administrative Tools, and then Services. Use the Start or Restart link for the service named `Subscription Service for xDB Replication Server`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

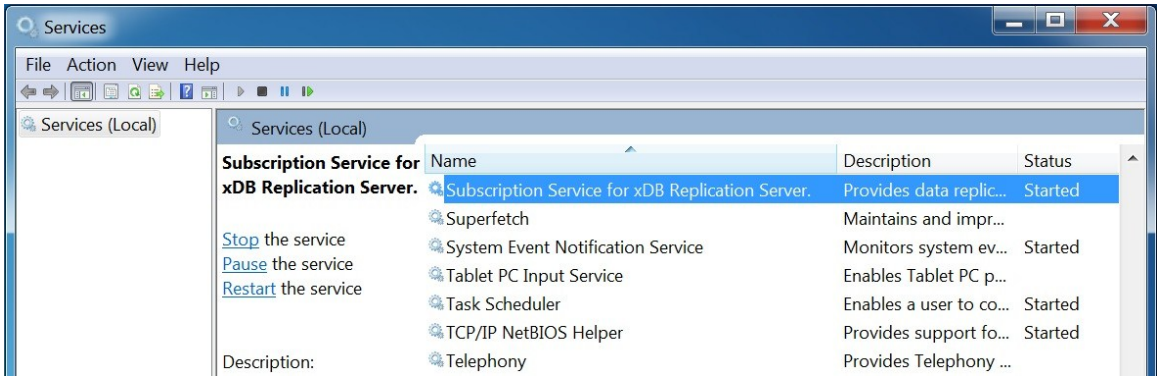


Figure 5-16 - Windows subscription service

If the subscription server fails to start, see Section 10.3.2.4 for information.

Step 2: Register the subscription server. Open the xDB Replication Console from the system's application menu. For xDB Replication Server installed from an xDB RPM package, the xDB Replication Console is started by invoking the script *XDB_HOME*`/bin/runRepConsole.sh`.

Step 3: Select the top level Replication Servers node. From the File menu, choose Subscription Server, and then choose Register Server. Alternatively, click the secondary mouse button on the Replication Servers node and choose Register Subscription Server. The Register Subscription Server dialog box appears.

Enter the values you supplied during the installation of xDB Replication Server unless otherwise specified.

- **Host.** Network IP address of the host running the subscription server. This is the network IP address used for *sub_ipaddr* in the `pg_hba.conf` file in Section 5.1.6.3. (Do not use `localhost` for this field.)
- **Port.** Port number the subscription server is using. This is the port number you specified on the Subscription Server Details screen in Step 17 of Section 3.1.
- **User Name.** Admin user name that is used to authenticate your usage of this subscription server. This is the user name you specified on the xDB Admin User Details screen in Step 15 of Section 3.1.
- **Password.** Password of the admin user given in the User Name field.
- **Save login information.** Check this box if you do not want to re-register the subscription server each time you open the xDB Replication Console. See Section 4.2 for additional information on the advantages and disadvantages of saving server login information.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: The user name and password combination you enter is authenticated against the admin user name and password in the xDB Replication Configuration file residing on the host with the IP address you enter in the Host field.

Figure 5-17 - Register Subscription Server dialog box

Click the Register button after you have filled in the fields. A Subscription Server node appears in the replication tree of the xDB Replication Console.

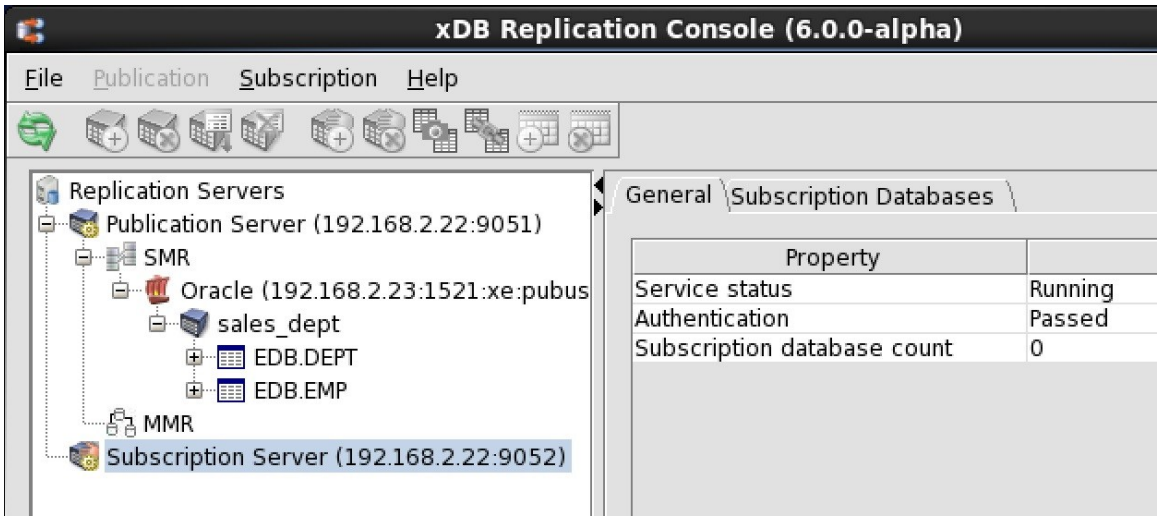


Figure 5-18 – Replication tree after registering a subscription server

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.3.2 Adding a Subscription Database

The database in which subscriptions are to reside must be identified to xDB Replication Server. This is done by creating a subscription database definition.

After the subscription database definition is created, a Subscription Database node representing that subscription database definition appears in the replication tree of the xDB Replication Console. Subscriptions created subordinate to this subscription database definition will have their publications replicated to the database identified by the subscription database definition.

You must enter database connection information such as the database server network address, database identifier, and database login user name and password when you create the subscription database definition. The connection information is used by the subscription server to create the subscription table definitions and by the publication server to perform replications.

Note the following restriction on the subscription database:

- **For Oracle only.** There must be no existing tables or views owned by the Oracle subscription database user that has the same name as a table or view in a publication that will be replicated to this database. For example, if the Oracle subscription database user name is `subuser`, and if a Postgres publication contains a table with the name `dept`, then the Oracle subscription database must not have an existing table or view with the schema-qualified name `subuser.dept` at the time you create the subscription.
- **For Postgres only.** There must be no existing tables or views with the same schema-qualified name as a table or view in a publication that will be replicated to this database. For example, if the publication contains a table with the schema-qualified name `edb.dept`, then the Postgres subscription database must not have an existing table or view with the schema-qualified name `edb.dept` at the time you create the subscription. **Note:** If the SQL Server publication schema name is `dbo`, the subscription tables are created under a schema named `dbo_sql` in Postgres.
- **For SQL Server only.** There must be no existing tables or views with the same schema-qualified name as a table or view in a publication that will be replicated to this database. For example, if the publication contains a table with the schema-qualified name `edb.dept`, then the SQL Server subscription database must not have an existing table or view with the schema-qualified name `edb.dept` at the time you create the subscription. **Note:** If the Postgres publication schema name is `public`, the subscription tables are created under a schema named `public_sql` in SQL Server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: A database that has been added as a publication database can also be used as a subscription database.

Step 1: Make sure the database server in which the subscription database resides is running and accepting client connections.

Step 2: Select the Subscription Server node. From the Subscription menu, choose Subscription Database, and then choose Add Database. Alternatively, click the secondary mouse button on the Subscription Server node and choose Add Database. The Subscription Service – Add Database dialog box appears.

Step 3: Fill in the following fields:

- **Database Type.** Select Oracle, SQL Server, PostgreSQL, or Postgres Plus Advanced Server for the type of subscription database. For an Advanced Server Oracle compatible installation, select the Postgres Plus Advanced Server option. For PostgreSQL or an Advanced Server PostgreSQL compatible installation, select the PostgreSQL option.
- **Host.** IP address of the host on which the subscription database server is running.
- **Port.** Port on which the subscription database server is listening for connections.
- **User.** The subscription database user name chosen in Section 5.1.5.1 for a Postgres subscription database or the database user name created in Step 2 of Section 5.1.5.2 for an Oracle subscription database or the database user name created in Step 2 of Section 5.1.5.3 for a SQL Server subscription database.
- **Password.** Password of the database user.
- **Service ID (For Oracle).** Enter the Oracle System Identifier (SID) of the Oracle instance running the subscription database if the SID radio button is selected. Enter the net service name of a connect descriptor as defined in the `TNSNAMES.ORA` file if the Service Name radio button is selected. **Note (For Oracle 12c Pluggable Database):** Use the service name.
- **Database (For Postgres or SQL Server).** Enter the Postgres or SQL Server database name.
- **URL Options (For SSL connectivity).** Enter the URL options to establish SSL connectivity to the subscription database. See Section 7.11 for information on using SSL connections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

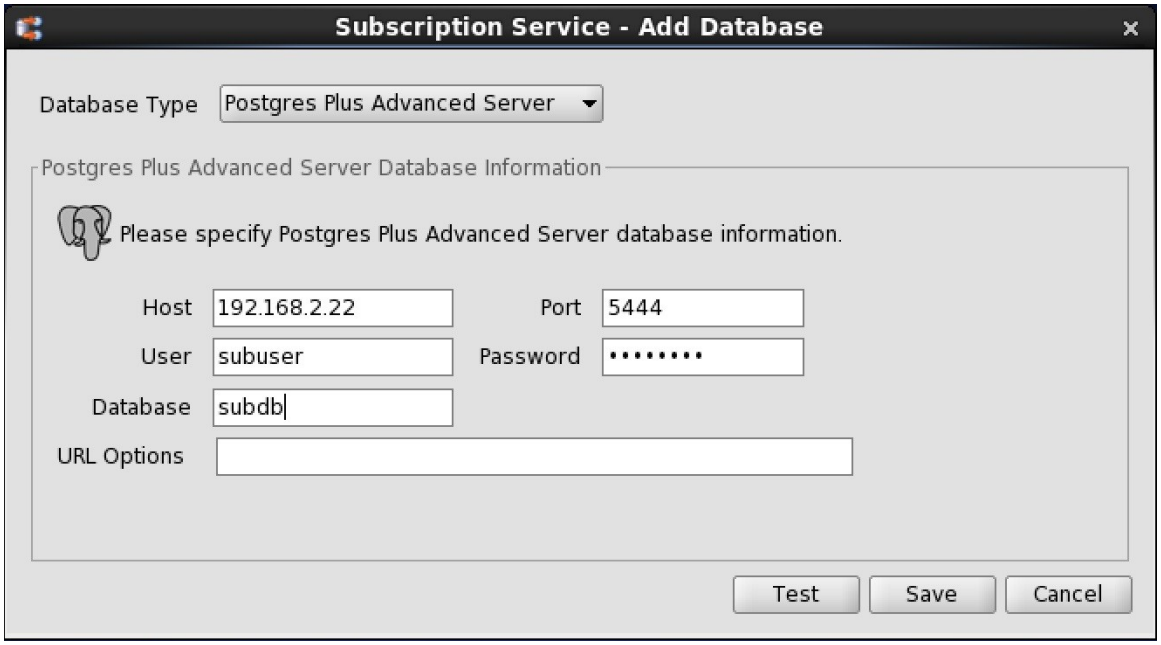


Figure 5-19 - Subscription Service - Add Database dialog box

Step 4: Click the Test button. If Test Result: Success appears, click the OK button, then click the Save button.

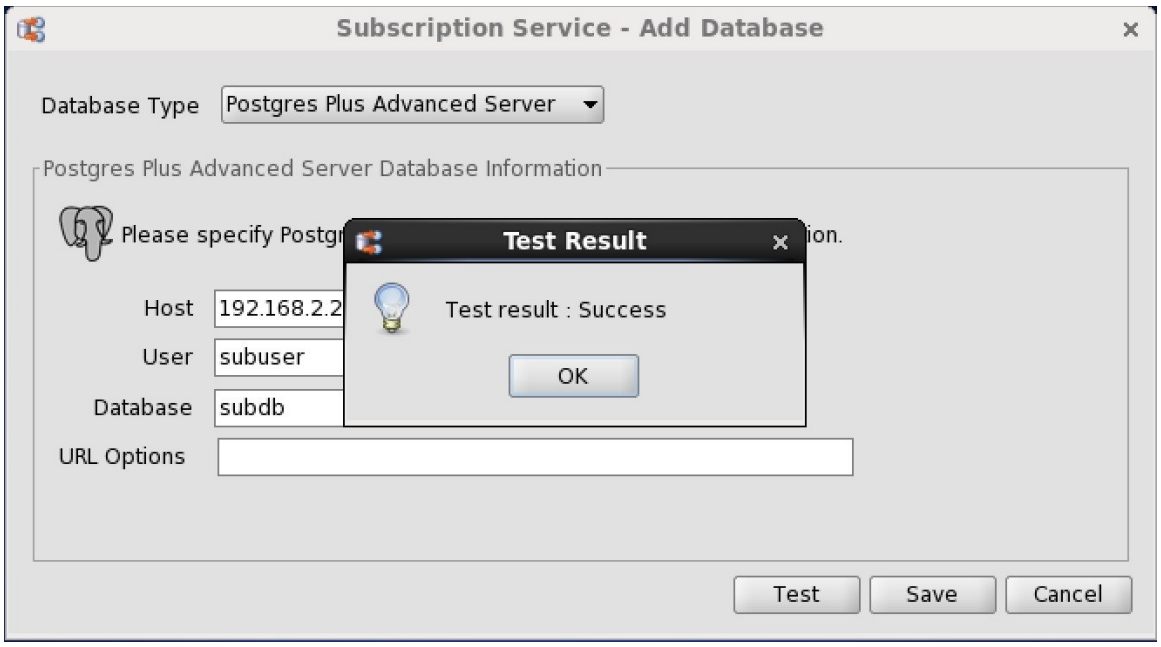


Figure 5-20 - Successful subscription database test

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 4.

When the subscription database definition is successfully saved, a Subscription Database node is added to the replication tree under the Subscription Server node.

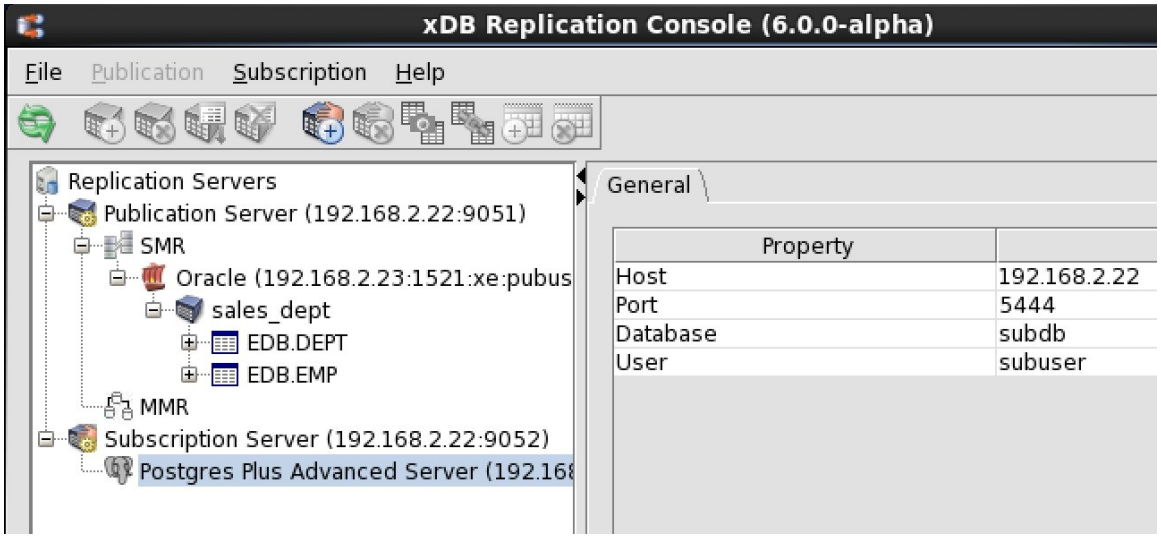


Figure 5-21 – Replication tree after adding a subscription database

5.3.3 Adding a Subscription

Subordinate to a subscription database definition, you create subscriptions. A subscription assigns the publication that is to be replicated to the database identified by the subscription database definition.

Step 1: Select the Subscription Database node. From the Subscription menu, choose Create Subscription. Alternatively, click the secondary mouse button on the Subscription Database node and choose Create Subscription. The Create Subscription dialog box appears.

Step 2: Fill in the following fields:

- **Subscription Name.** Enter a name for the subscription that is unique amongst all subscription names.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- **Host.** Network IP address of the publication server that is the parent node of the publication to be subscribed to. This is the same value entered in the Host field in Step 3 of Section 5.2.1.
- **Port.** Port used by the publication server. This is the same value entered in the Port field in Step 3 of Section 5.2.1.
- **User Name.** Admin user name of the publication server. This is the same value entered in the User Name field in Step 3 of Section 5.2.1.
- **Password.** Password of the admin user. This is the same value entered in the Password field in Step 3 of Section 5.2.1.
- **Publication Name.** Click the Load button to get a list of available publications. Select the publication to which to subscribe.

Figure 5-22 - Create Subscription dialog box

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3 (Optional): If you defined a set of available table filters for the publication, you have the option of enabling these filters on this subscription. See Section 5.2.3 for instructions on defining table filters. If you do not wish to filter the rows that are replicated to this subscription, go to Step 4.

Click the Filter Rules tab to enable one or more filter rules on the subscription. At most one filter rule may be enabled on any given subscription table.

In the following example the filter named `dept_10_20_30` is enabled on the `dept` table and the filter named `dept_30` is enabled on the `emp` table of this subscription.

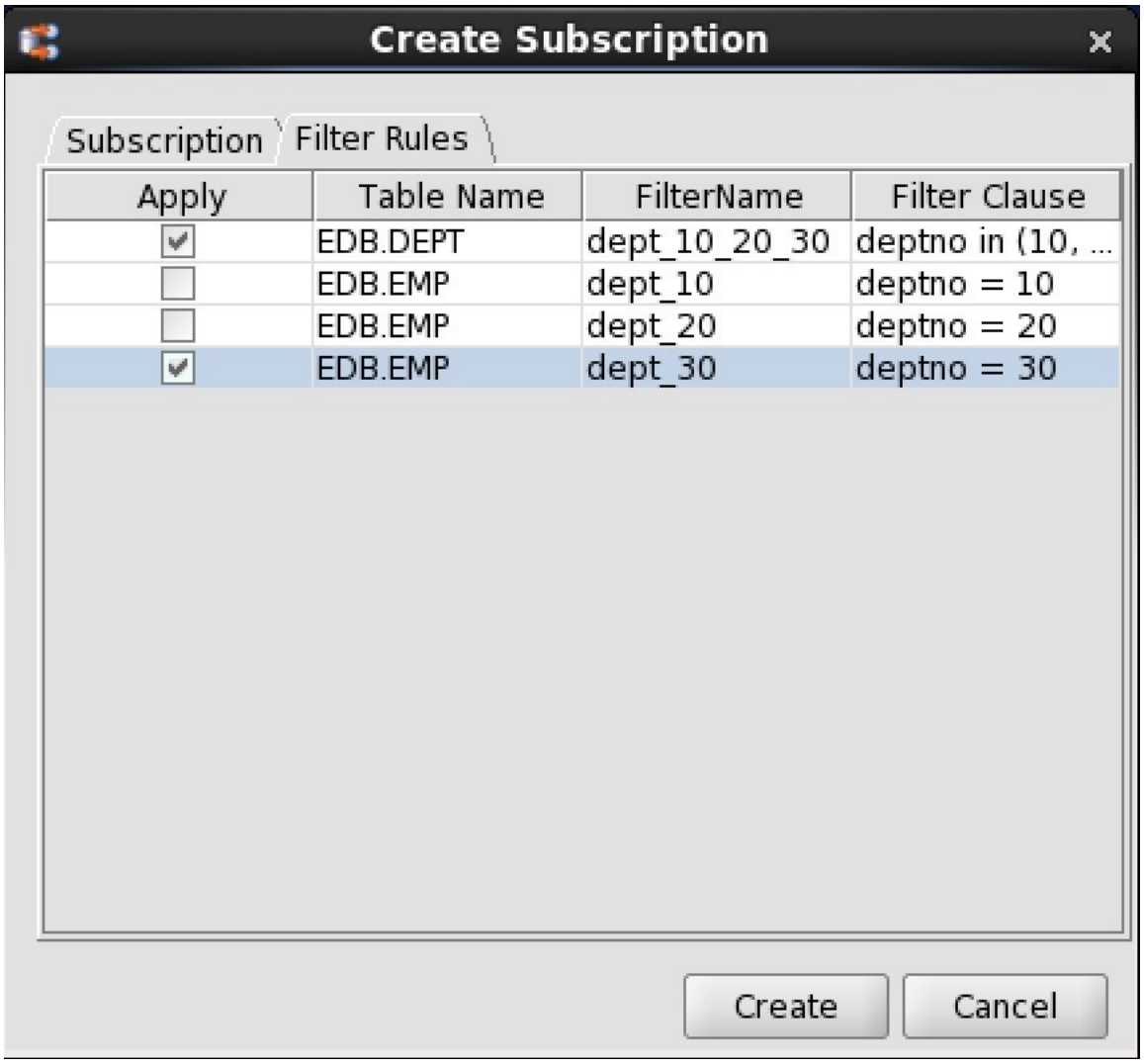


Figure 5-23 - Enabling filter rules on a subscription

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4: Click the Create button. If Subscription Created Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

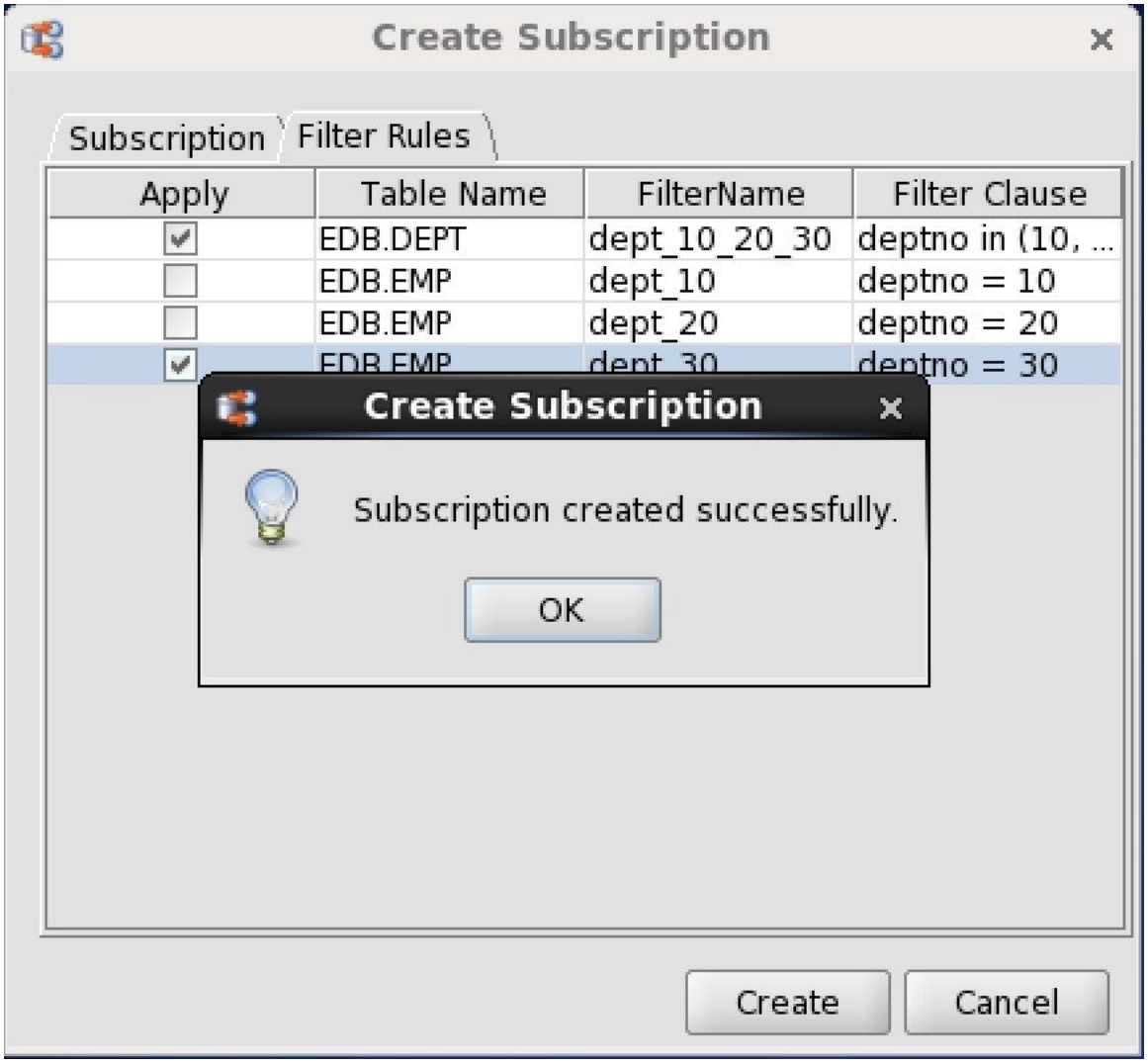


Figure 5-24 - Subscription created successfully

Upon successful subscription creation, a Subscription node is added to the replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

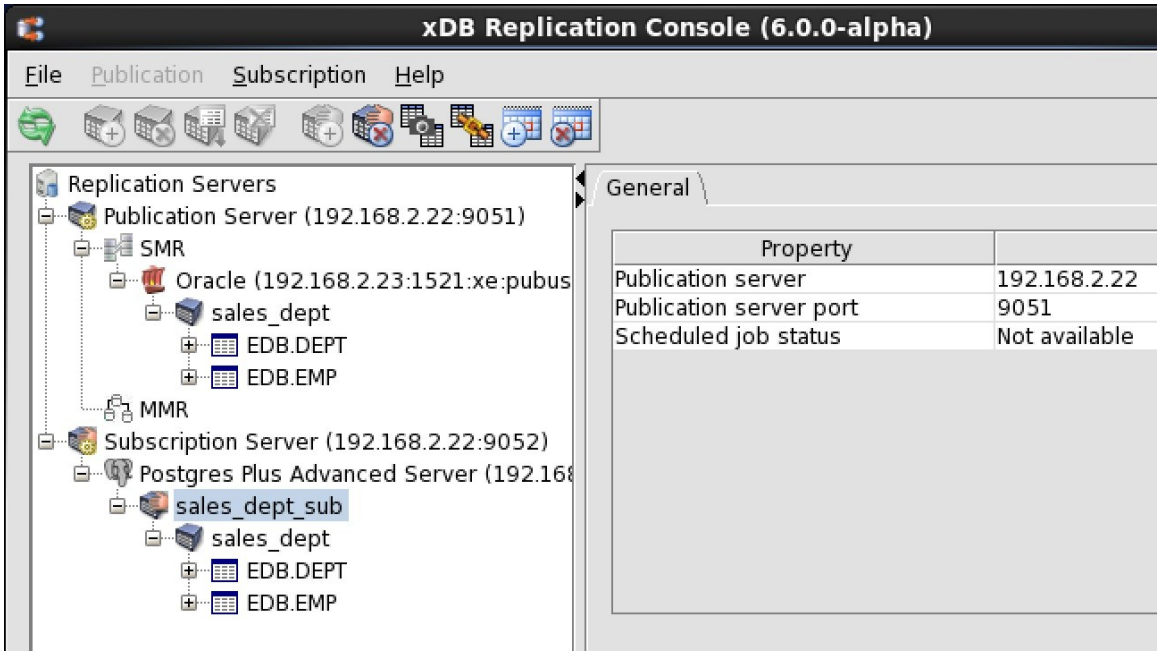


Figure 5-25 – Replication tree after adding a subscription

The tables and views from the publication are created in the subscription database, but without any rows. Rows are populated into the subscription tables when the first snapshot replication occurs.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

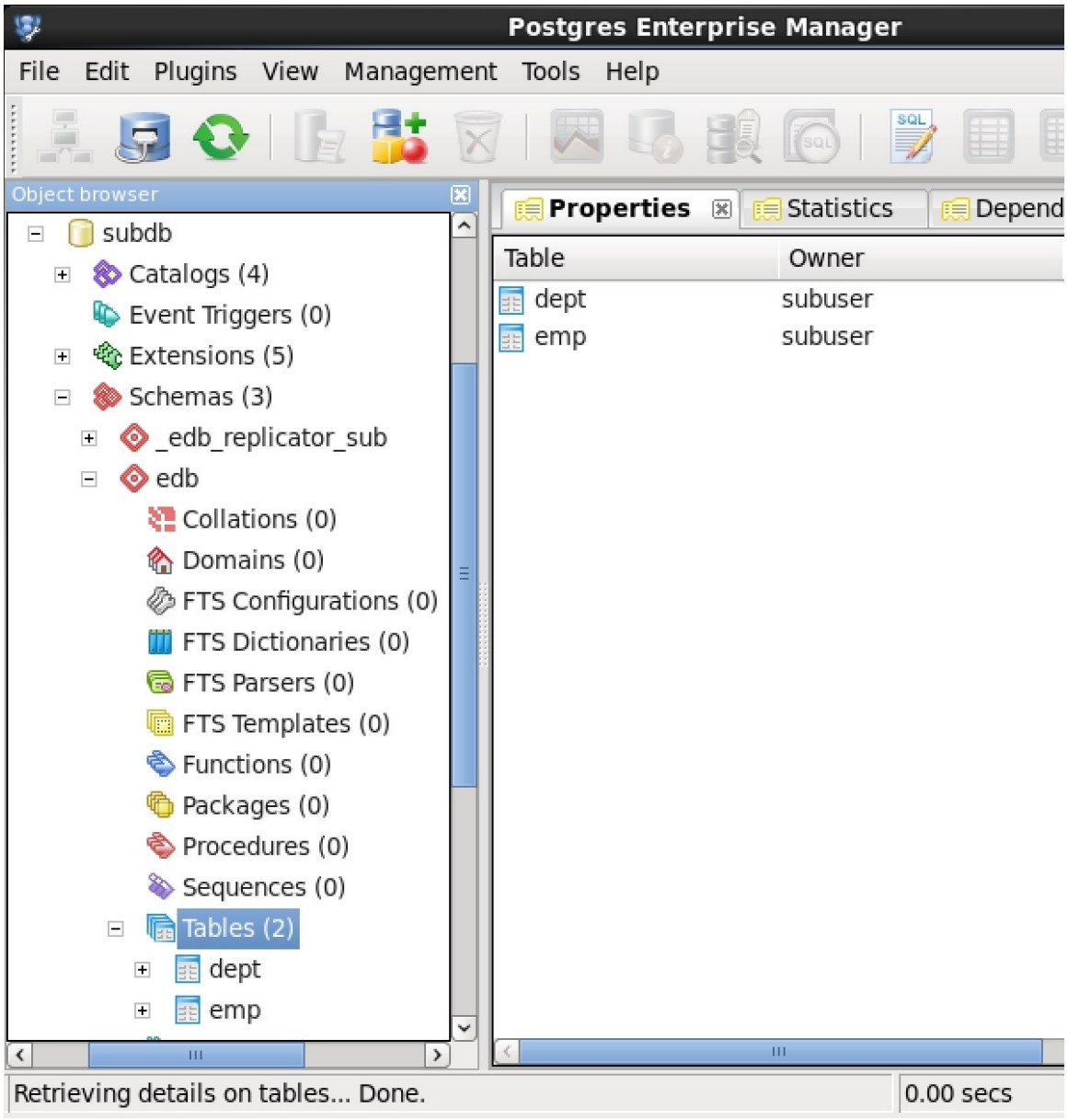


Figure 5-26 - Table definitions in the subscription database

5.3.4 Subscription Metadata Object

After you have added a subscription database definition you will find a single table named `rrep_txset_health` has been created as the subscription metadata object.

For Oracle only: The `RREP_TXSET_HEALTH` table is created in the subscription database user's schema as shown in the following output:

```
SQL> CONNECT subuser/password
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Connected.
SQL> SET PAGESIZE 9999
SQL> SELECT table_name FROM user_tables ORDER BY table_name;

TABLE_NAME
------------------------------
RREP_TXSET_HEALTH
```

For SQL Server only: The `rrep_txset_health` table is created in the schema named `_edb_replicator_sub`.

```
1> USE subdb;
2> GO
Changed database context to 'subdb'.
1> SELECT s.name + '.' + o.name "Object Name", o.type_desc "Object Type"
2>   FROM sys.objects o,
3>        sys.schemas s
4>   WHERE s.name <> 'edb'
5>     AND o.type IN ('U','P','FN')
6>     AND o.schema_id = s.schema_id
7>   ORDER BY 2, 1;
8> GO
Object Name                          Object Type
------------------------------------ ------------------------------------
_edb_replicator_sub.rrep_txset_health  USER_TABLE

(1 rows affected)
```

For Postgres only: The `rrep_txset_health` table is created in the schema named `_edb_replicator_sub`.

```
subdb=# SET search_path TO _edb_replicator_sub;
SET
subdb=# \dt
                    List of relations
       Schema        |       Name        | Type  |  Owner
---------------------+-------------------+-------+---------
 _edb_replicator_sub | rrep_txset_health | table | subuser
(1 row)
```

In all subscription database types (Oracle, SQL Server, and Postgres) when you remove the subscription database definitions using the xDB Replication Console or xDB Replication Server CLI, the subscription metadata object is deleted from the subscription database.

5.4 On Demand Replication

After a publication and subscription are created, there are a couple of choices for starting the replication process.

- Replication can be done immediately by taking an on demand snapshot.
- Replication can be scheduled to start at a later date and time by creating a schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

This section discusses the procedure for initiating a replication on demand. Section 7.2 discusses how to create a schedule.

5.4.1 Performing Snapshot Replication

The very first replication must be performed using snapshot replication. After the first snapshot replication, subsequent replications can be done using either the synchronization method (if the publication was not initially defined as a snapshot-only publication) or the snapshot method.

Step 1: Select the Subscription node of the subscription for which you wish to perform snapshot replication.

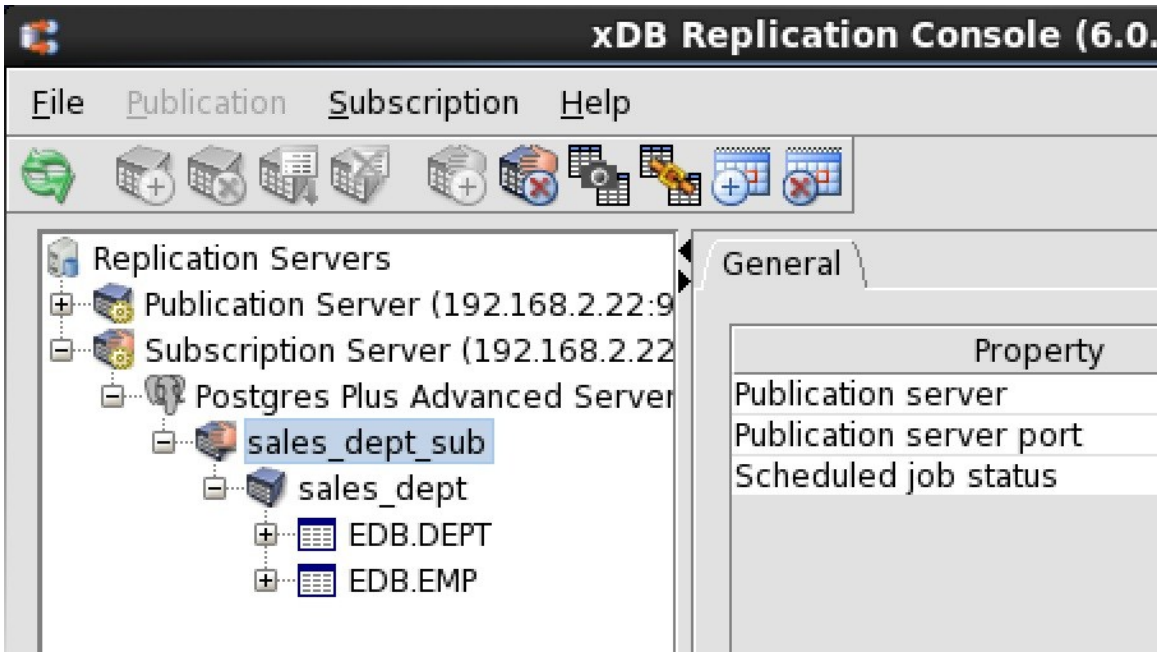


Figure 5-27 - Selecting a subscription for an on demand snapshot

Step 2: Open the Snapshot dialog box in any of the following ways:

- From the Subscription menu, choose Snapshot.
- Click the secondary mouse button on the Subscription node and choose Snapshot.
- Click the primary mouse button on the Snapshot icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

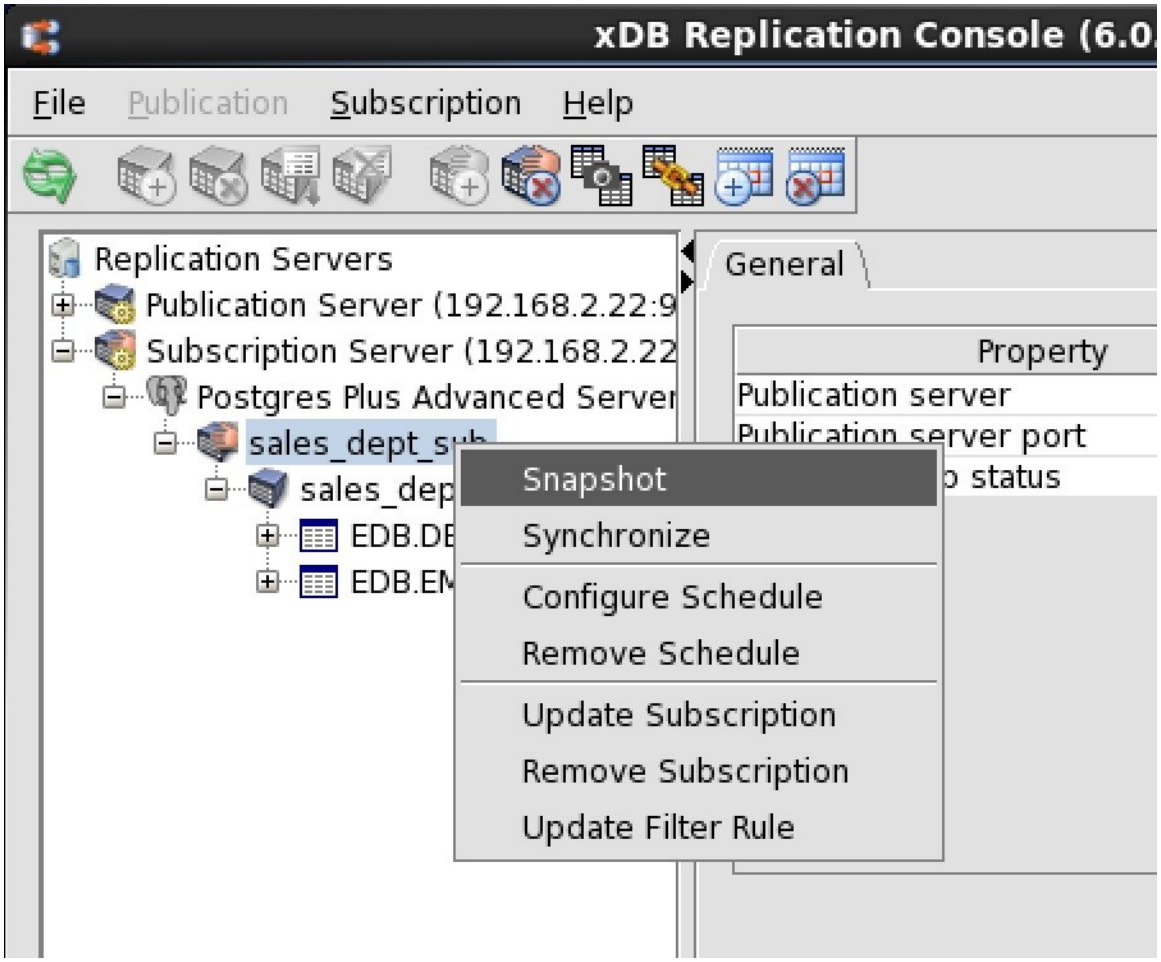


Figure 5-28 - Opening the Snapshot dialog box

Step 3: Select the Verbose Output check box only if you want to display the output from the snapshot in the dialog box. This option should be left unchecked in a network address translation (NAT) environment as a large amount of output from the snapshot may delay the response from the Snapshot dialog box. Click the Snapshot button to start snapshot replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

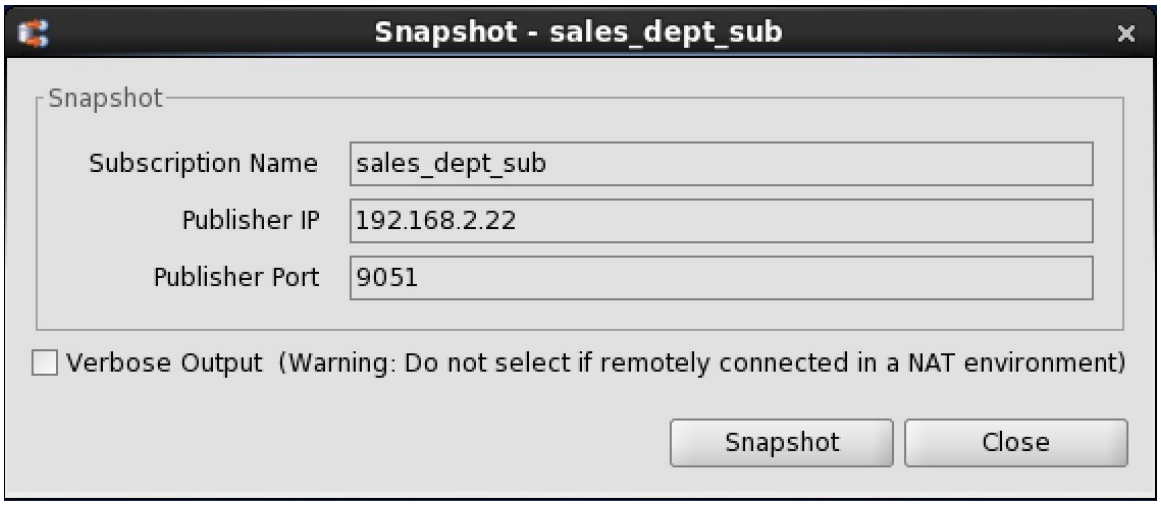


Figure 5-29 - Snapshot dialog box

Step 4: Snapshot Taken Successfully appears if the snapshot was successful. Click the OK button. If the snapshot was not successful, scroll through the messages in the Snapshot dialog box window if Verbose Output was selected or check the log files.

The status messages of each snapshot are saved in the *Migration Toolkit log* files named `mtk.log[.n]` (where `[.n]` is an optional history file count if log file rotation is enabled) in the following directories:

For Linux:

```
/var/log/xdb-x.x
```

For Windows:

```
POSTGRES_HOME\.enterprisedb\xdb\x.x
```

POSTGRES_HOME is the home directory of the Windows `postgres` account (`enterprisedb` account for Advanced Server installed in Oracle compatible configuration mode). The specific location of *POSTGRES_HOME* is dependent upon your version of Windows. The xDB Replication Server version number is represented by *x.x*.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

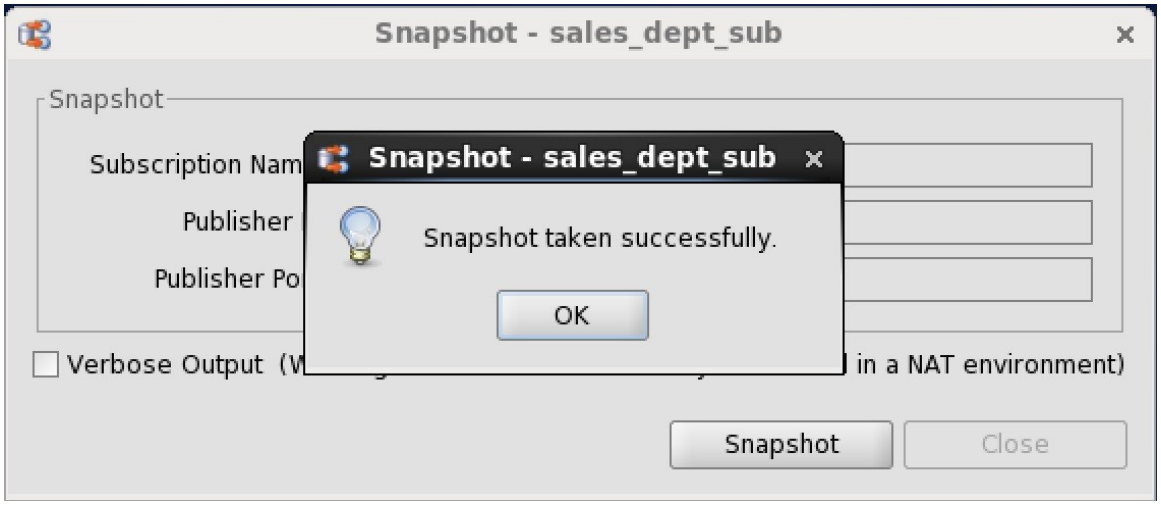


Figure 5-30 - Successful on demand snapshot

The publication has now been replicated to the subscription database. A record of the snapshot is maintained in the replication history. See Section 7.4 for information on how to view replication history.

5.4.2 Performing Synchronization Replication

After the first snapshot replication, subsequent replications can be performed using synchronization replication if the publication was not created as a snapshot-only publication.

Step 1: When the trigger-based method of synchronization replication is in use, select the Subscription node of the subscription for which you wish to perform synchronization replication.

When the log-based method of synchronization replication is in use, select the Subscription node of any subscription. For the log-based method, the synchronization replication will be performed on all subscriptions regardless of which one is selected.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

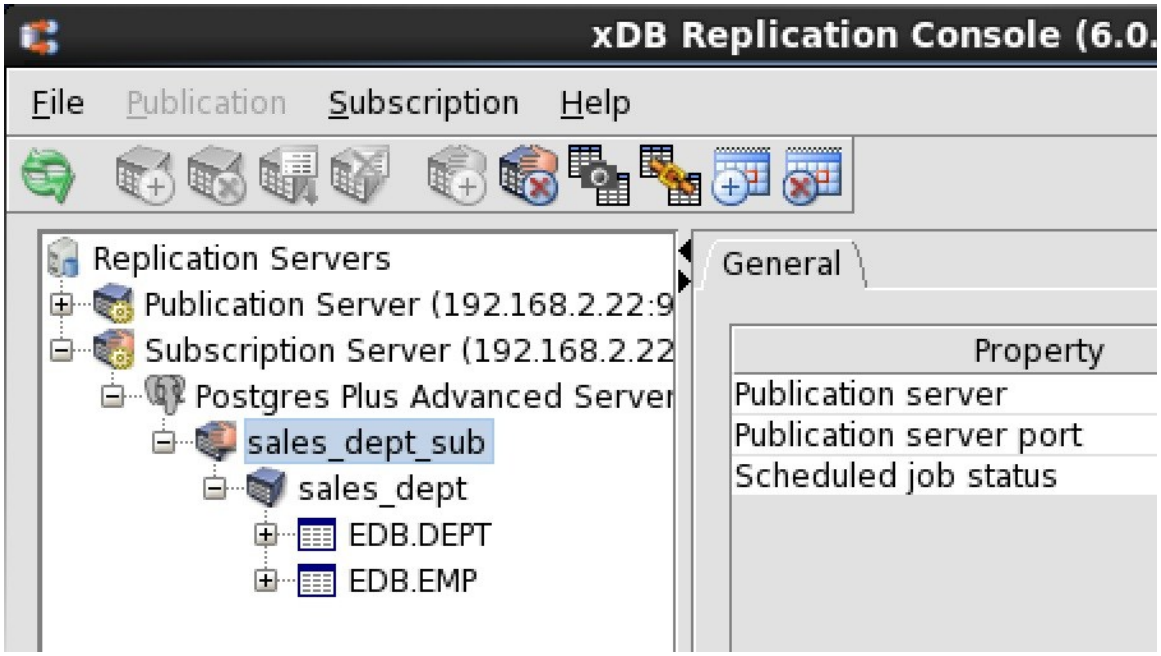


Figure 5-31 - Selecting a subscription for an on demand synchronization

Step 2: Open the Synchronize dialog box in any of the following ways:

- From the Subscription menu, choose Synchronize.
- Click the secondary mouse button on the Subscription node and choose Synchronize.
- Click the primary mouse button on the Synchronize icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

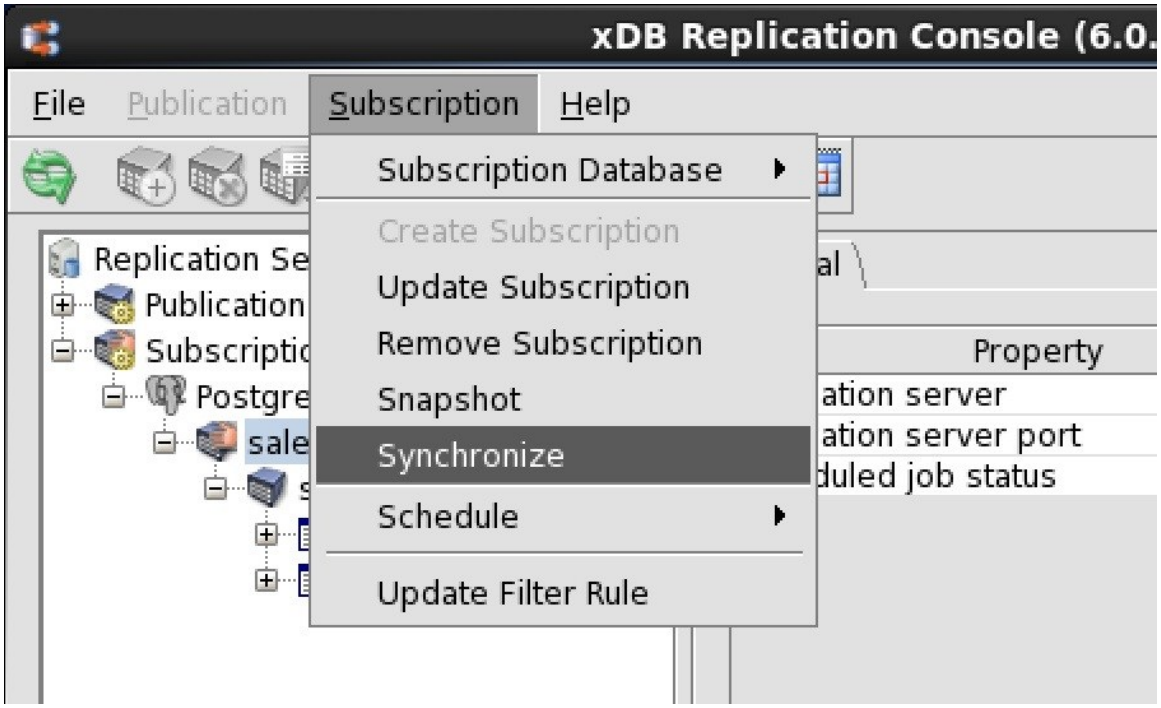


Figure 5-32 - Opening the Synchronize dialog box

Step 3: Click the Synchronize button to start synchronization replication.

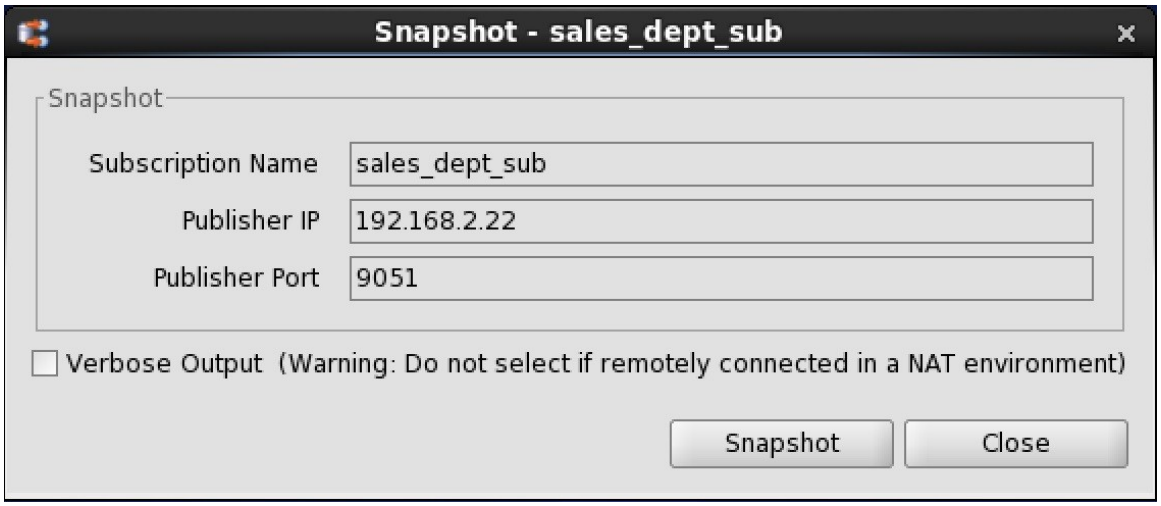


Figure 5-33 - Synchronize dialog box

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4: Subscription Synchronized Successfully appears if the synchronization was successful. Click the OK button. If the synchronization was not successful, scroll through the messages in the Synchronize dialog box window.

Figure 5-34 - Successful on demand synchronization

The operations that were applied to the subscription tables can be seen in the replication history. See Section 7.4 for information on how to view replication history.

5.5 Managing a Subscription

Note: This section discusses various aspects of managing a subscription of a replication system. For a similar discussion on managing a publication of a replication system, see Section 7.6.

After a subscription has been created, certain aspects of the underlying replication system environment might be subsequently altered for any number of reasons. Attributes that might change include the network location of the subscription database server, the network location of the host running the subscription server, database or operating system user names and passwords, and so forth.

The aforementioned information is saved in the replication system metadata when a subscription is created. Changes to these attributes result in inaccurate replication system metadata, which in turn may result in errors during subsequent replication attempts or replication system administration.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

This section describes how to update the metadata stored for the subscription server, the subscription database definition, and subscriptions in order to keep the information consistent with the actual replication system environment.

5.5.1 Updating a Subscription Server

When you register a subscription server in the xDB Replication Console, you may choose to save the subscription server's network location (IP address and port number), admin user name, and encrypted password in a server login file on the computer on which you are running the xDB Replication Console. See Section 4.2 for information on saving the login information.

The steps described in this section show you how to update the subscription server's login information in the server login file.

It is assumed that the xDB Replication Console is open on your computer and the subscription server whose login information you wish to alter in the server login file, appears as a Subscription Server node in the xDB Replication Console's replication tree.

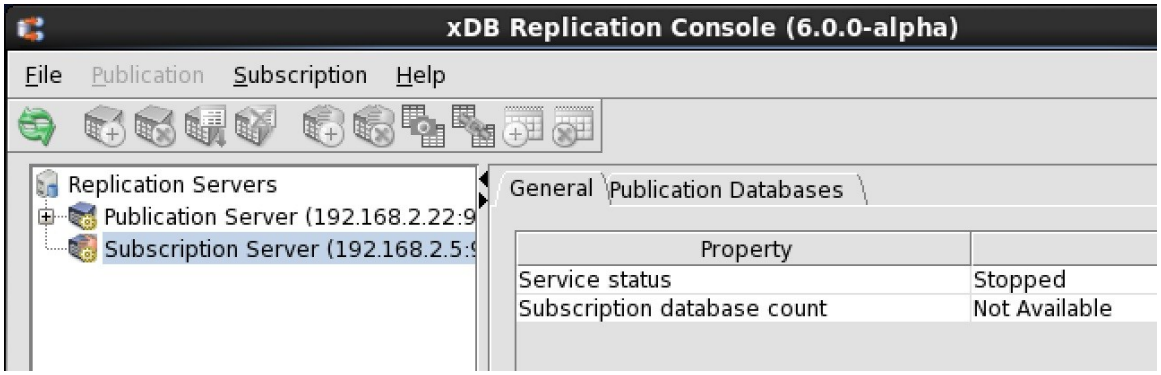


Figure 5-35 - Subscription Server node

You can perform the following actions on the server login file:

- Change the subscription server's login information (host IP address, port number, admin user name, and password) that you last saved in the server login file.
- Delete the subscription server's login information that is currently saved in the server login file. This is the default action, which will require you to register the subscription server again the next time you open the xDB Replication Console.
- Resave the subscription server's login information in the server login file. Each time you open the Update Subscription Server dialog box, you must choose to save the login information if you want it recorded in the server login file.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following steps change only the content of the server login file residing on the host under the current xDB Replication Console user's home directory. These changes do not alter any characteristic of the actual subscription server daemon (on Linux) or service (on Windows). These changes affect only how a subscription server is viewed through the xDB Replication Console on this host by this user.

Step 1: The subscription server whose login information you want to save, change, or delete in the server login file must be running before you can make any changes to the file. See Step 1 of Section 5.3.1 for directions on starting the subscription server.

Step 2: Click the secondary mouse button on the Subscription Server node and choose Update. The Update Subscription Server dialog box appears.

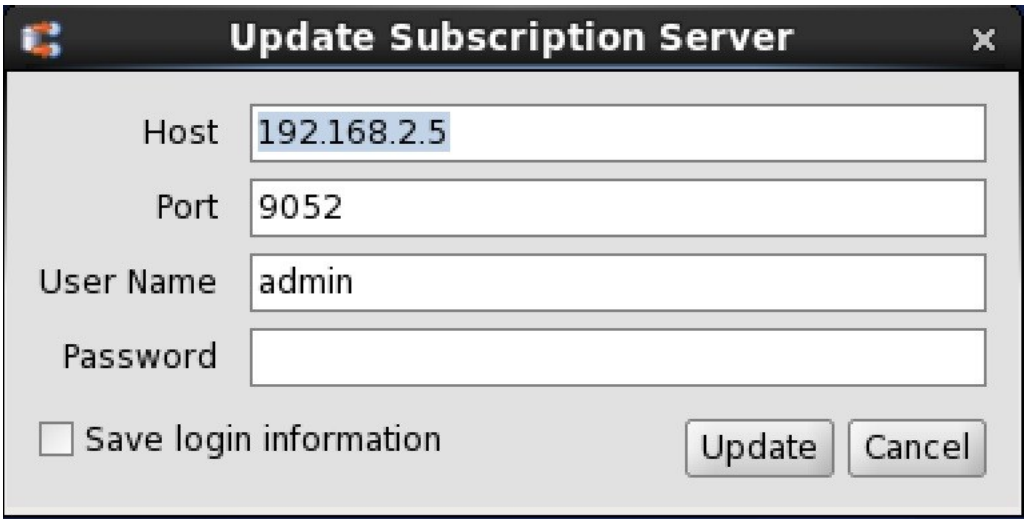


Figure 5-36 - Update Subscription Server dialog box

Step 3: Complete the fields in the dialog box according to your purpose for updating the server login file:

- If the subscription server now runs on a host with a different IP address or port number than what is shown in the dialog box, enter the correct information. You must also enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address you entered in the Host field. Check the Save Login Information box if you want the new login information saved in the server login file, otherwise leave the box unchecked in which case, access to the subscription server is available for the current session, but subsequent sessions will require you to register the subscription server again.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- If you want to delete previously saved login information, make sure the network location shown in the dialog box is still correct. Re-enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address in the Host field. Leave the Save Login Information box unchecked. Access to the subscription server is available for this session, but subsequent sessions will require you to register the subscription server again.
- If you want to save the current login information shown in the dialog box, make sure the network location shown in the dialog box is correct. Re-enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address in the Host field. Check the Save Login Information box.

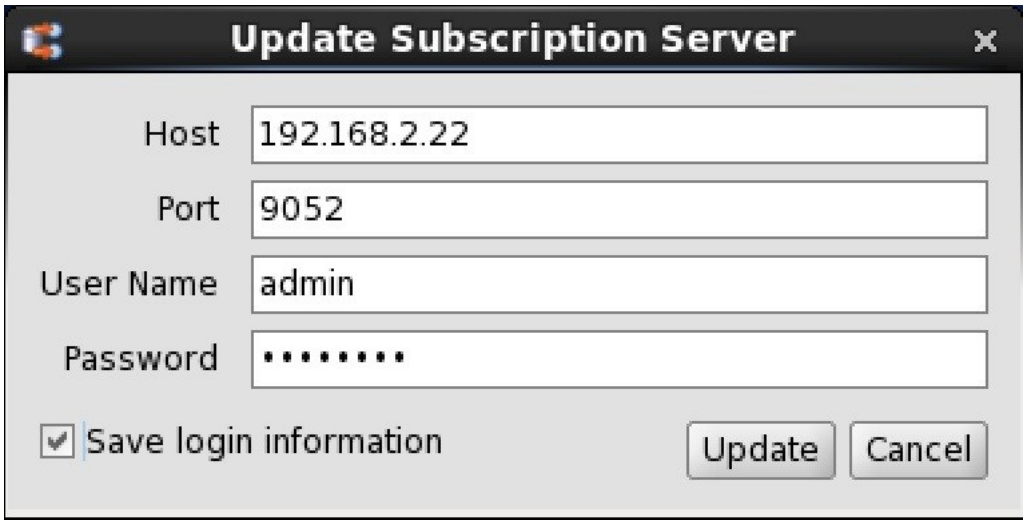


Figure 5-37 - Updated subscription server location

Step 4: Click the Update button. If the dialog box closes, then the update to the server login file was successful. Click the Refresh icon in the xDB Replication Console tool bar to show the updated Subscription Server node.

If an error message appears after clicking the Update button, the server login file is not modified. Investigate and correct the cause of the error. Repeat steps 1 through 4.

5.5.2 Updating a Subscription Database

When you create a subscription database definition, you save the subscription database server's network location (IP address and port number), the database identifier, a database login user name, and the user's password in the control schema accessed by the subscription server. This login information is used whenever a session needs to be

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

established with the subscription database. See Section 5.3.2 for information on creating a subscription database definition.

The steps described in this section show you how to update the subscription database login information stored in the control schema should any of these attributes of the actual, physical database change.

Note: Depending upon the database type (Oracle, SQL Server, or Postgres), certain attributes must not be changed. If you have already added subscriptions, you must not change any attribute that alters access to the schema where the subscription tables were created.

Attributes you must not change if there are existing subscriptions include the following:

- The Oracle login user name as the subscription tables already reside in this Oracle user's schema
- The database server network location if the new network location references a database server that does not access the database that already contains the subscription tables
- The database identifier if the new database identifier references a different physical database than where the subscription tables already reside

Attributes you may change include the following:

- The login user name's password to match a changed database user password
- The database server network location if the corresponding location change was made to the database server that accesses the subscription database
- The database identifier such as the Oracle service name, SQL Server database name, or Postgres database name if the corresponding name change was made on the database server
- All attributes may be changed if there are no existing subscriptions

Step 1: Make sure the database server that you ultimately wish to save as the subscription database definition is running and accepting client connections.

Step 2: Make sure the subscription server whose node is the parent of the subscription database definition you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 3: Select the Subscription Database node corresponding to the subscription database definition that you wish to update.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

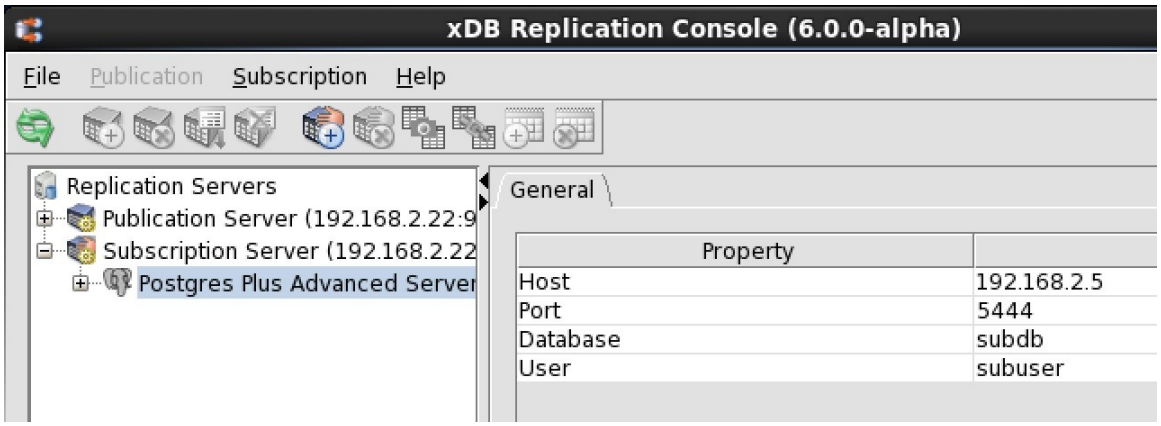


Figure 5-38 - Selecting a subscription database definition for update

Step 4: From the Subscription menu, choose Subscription Database, and then choose Update Database. Alternatively, click the secondary mouse button on the Subscription Database node and choose Update Database. The Update Database Source dialog box appears.

Step 5: Enter the desired changes. See Step 3 of Section 5.3.2 for the precise meanings of the fields.

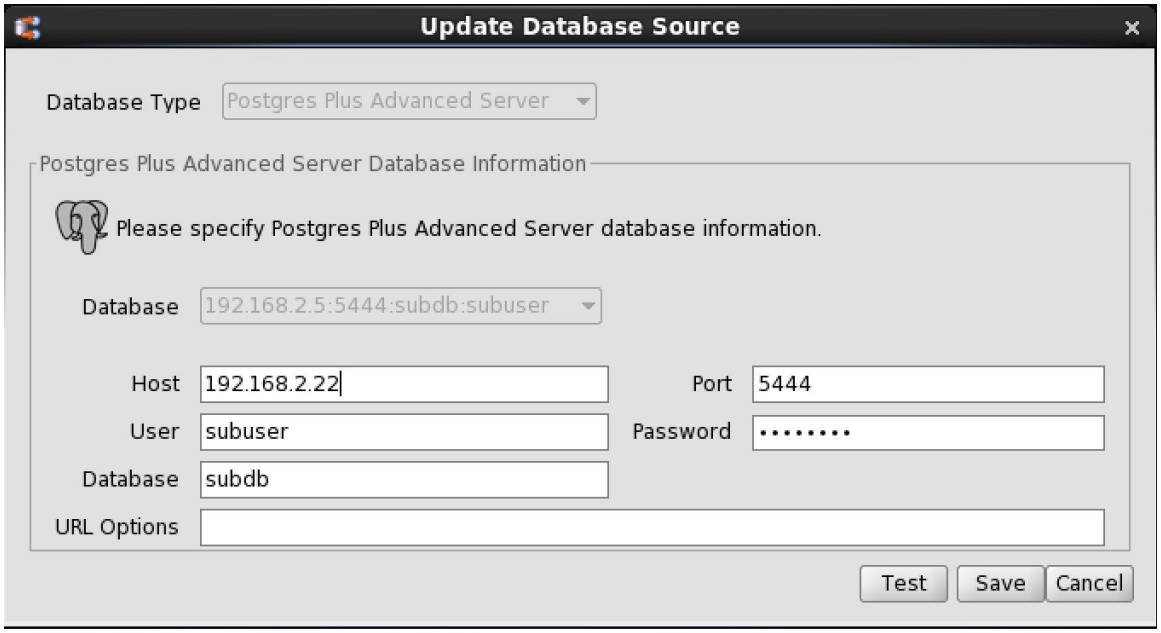


Figure 5-39 - Update Database Source dialog box

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 6: Click the Test button. If Test Result: Success appears, click the OK button, then click the Save button.

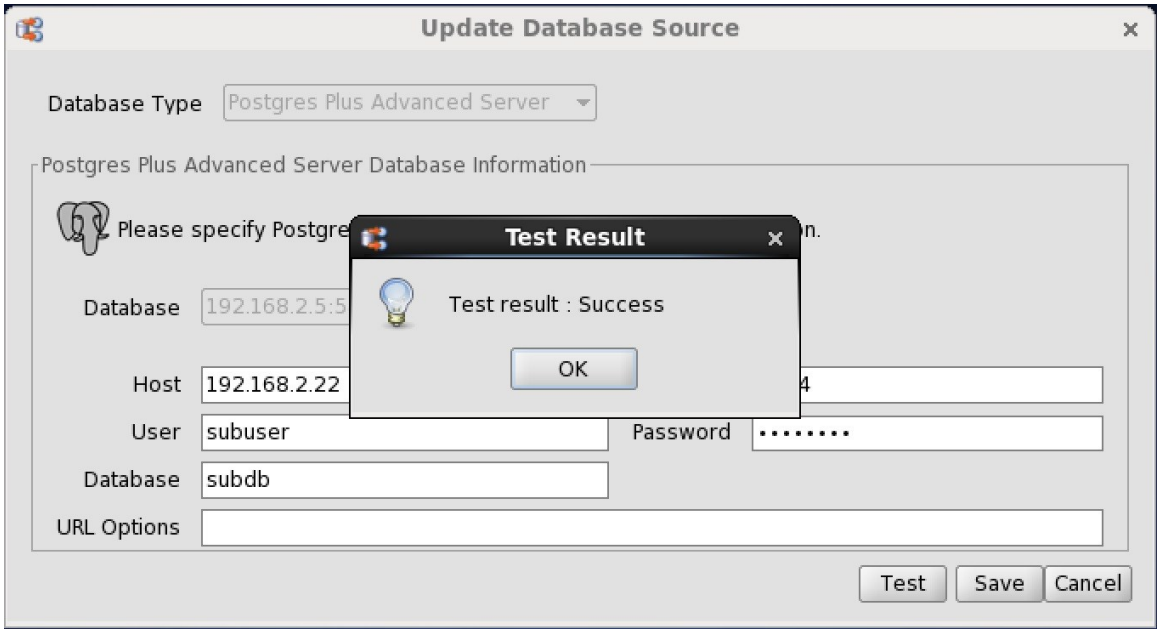


Figure 5-40 - Successful subscription database test

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 6.

Step 7: Click the Refresh icon in the xDB Replication Console tool bar to show the updated Subscription Database node and any of its subscriptions.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

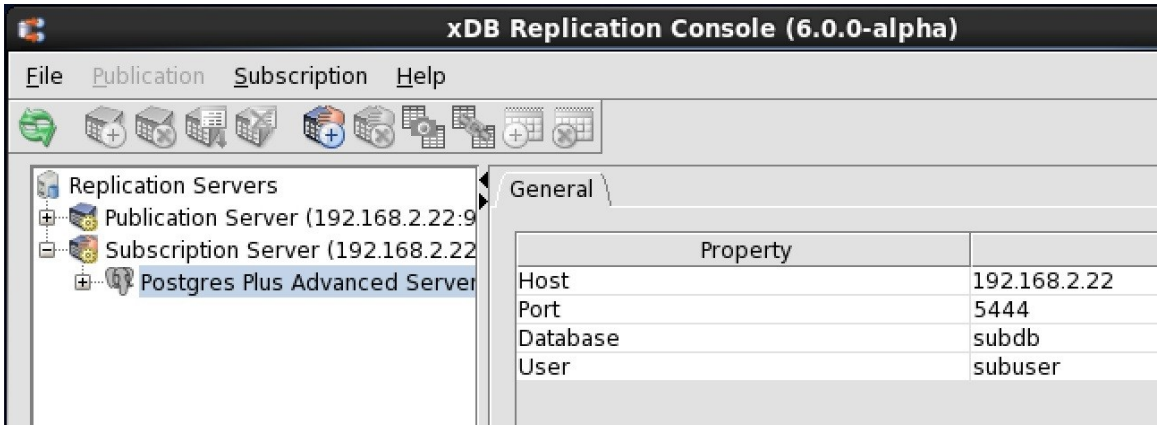


Figure 5-41 - Updated subscription database

5.5.3 Updating a Subscription

When a subscription is created, certain attributes of the subscribed publication are stored as part of the metadata for the subscription in the control schema. These include the following:

- The network IP address of the host running the publication server that is the parent of the subscribed publication
- The port number of the publication server

If the preceding attributes of the publication server change in the replication system environment, then the corresponding subscription metadata must also be changed so the subscription server can communicate with the correct publication server.

The following directions show how to update the publication server network IP address and port number within the subscription server's metadata.

Step 1: Make sure the subscription server whose node is the parent of the subscription you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 2: Select the Subscription node whose attributes you wish to update.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

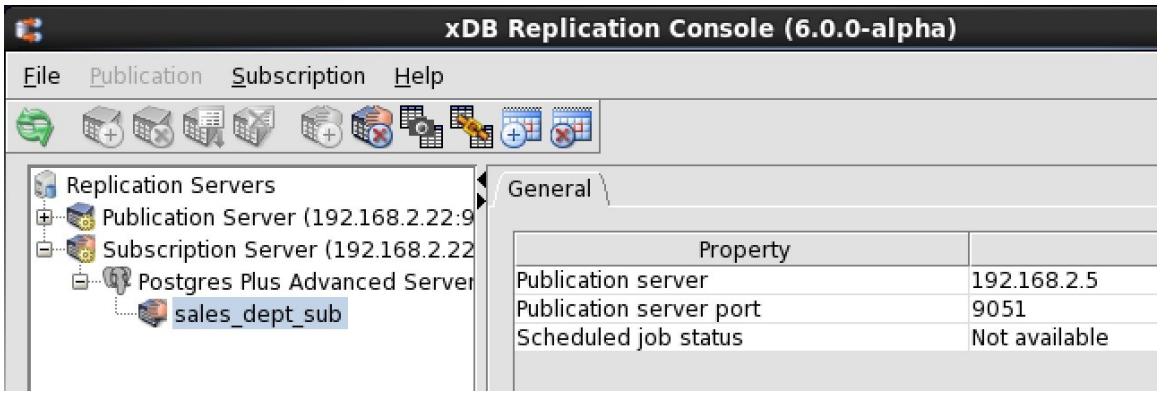


Figure 5-42 - Selecting a subscription to update

Step 3: From the Subscription menu, choose Update Subscription. Alternatively, click the secondary mouse button on the Subscription node and choose Update Subscription. The Update Subscription dialog box appears.

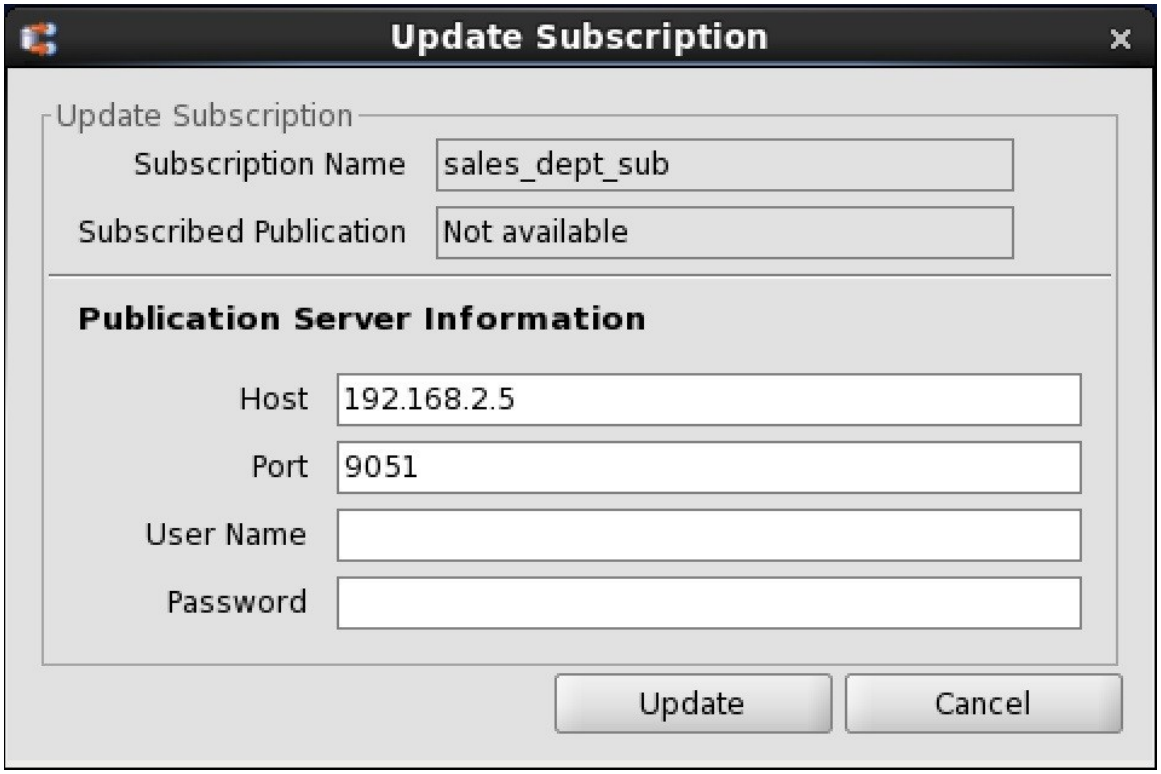


Figure 5-43 - Update Subscription dialog box

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4: If the publication server now runs on a host with a different IP address or port number than what is shown in the dialog box, enter the correct information. You must also enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host on which the publication server is running. Click the Update button.

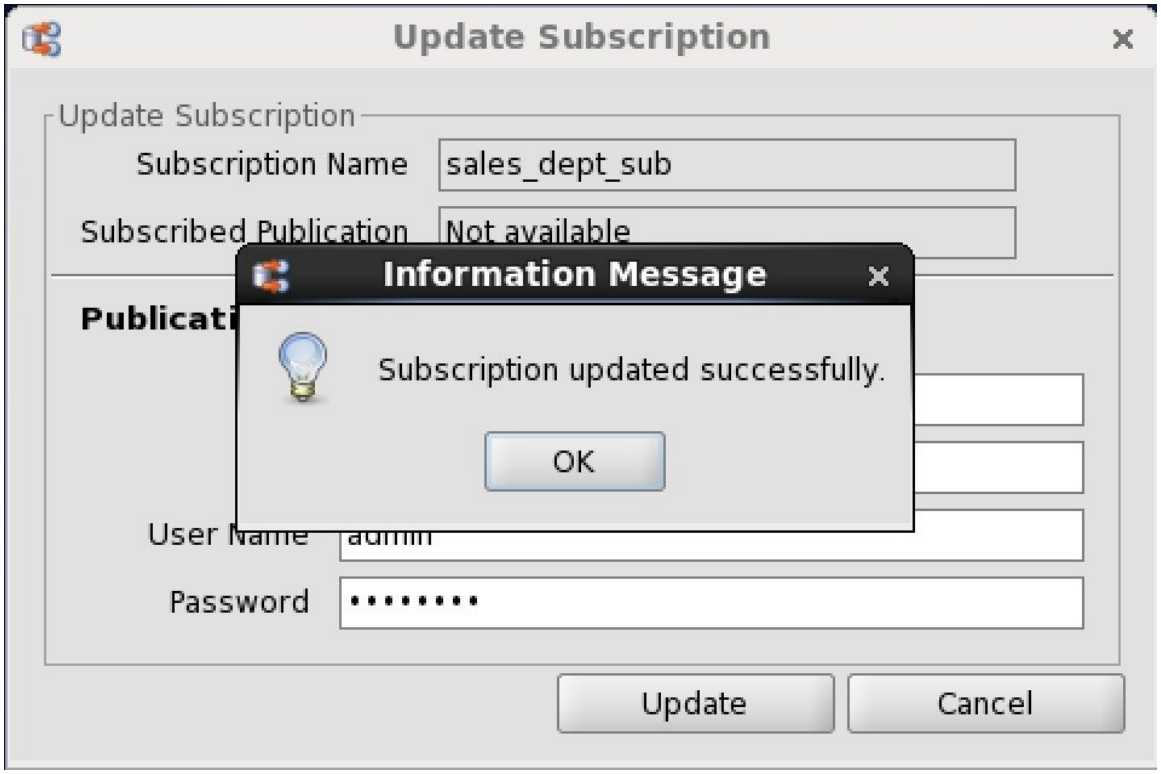


Figure 5-44 - Subscription successfully updated

Step 5: If Subscription Updated Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

Step 6: If the publication server with the new network location manages publications subscribed to by other subscriptions, repeat steps 1 through 5 for these other subscriptions.

5.5.4 Enabling/Disabling Table Filters on a Subscription

Table filters must first be defined in a set of available table filters in the publication before they can be enabled on a subscription. See Section 5.2.3 for information on defining table filters in a single-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following are the steps for enabling or disabling table filters on an existing subscription.

Step 1: Make sure the publication server whose node is the parent of the publication associated with the subscription you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server. Make sure the subscription server whose node is the parent of the subscription you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 2: Select the Subscription node of the subscription on which you wish to enable or disable individual filter rules.

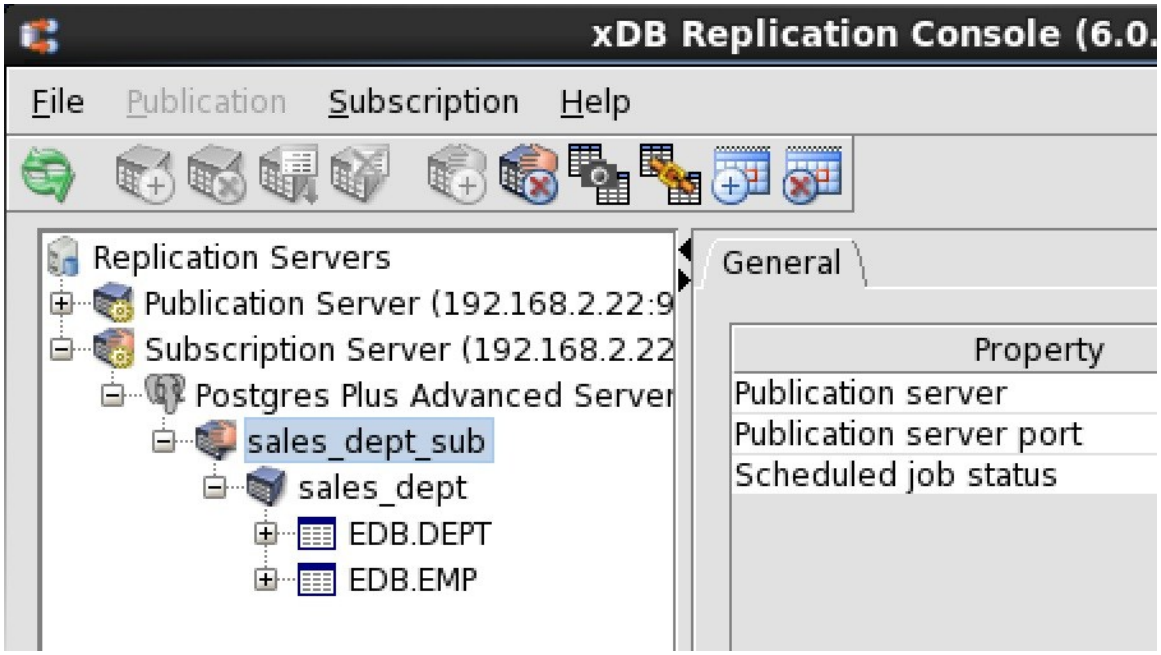


Figure 5-45 - Selecting a subscription on which to enable or disable filter rules

Step 3: Open the Filter Rules tab in any of the following ways:

- Choose Update Filter Rule from the Subscription menu.
- Click the secondary mouse button on the Subscription node and choose Update Filter Rule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

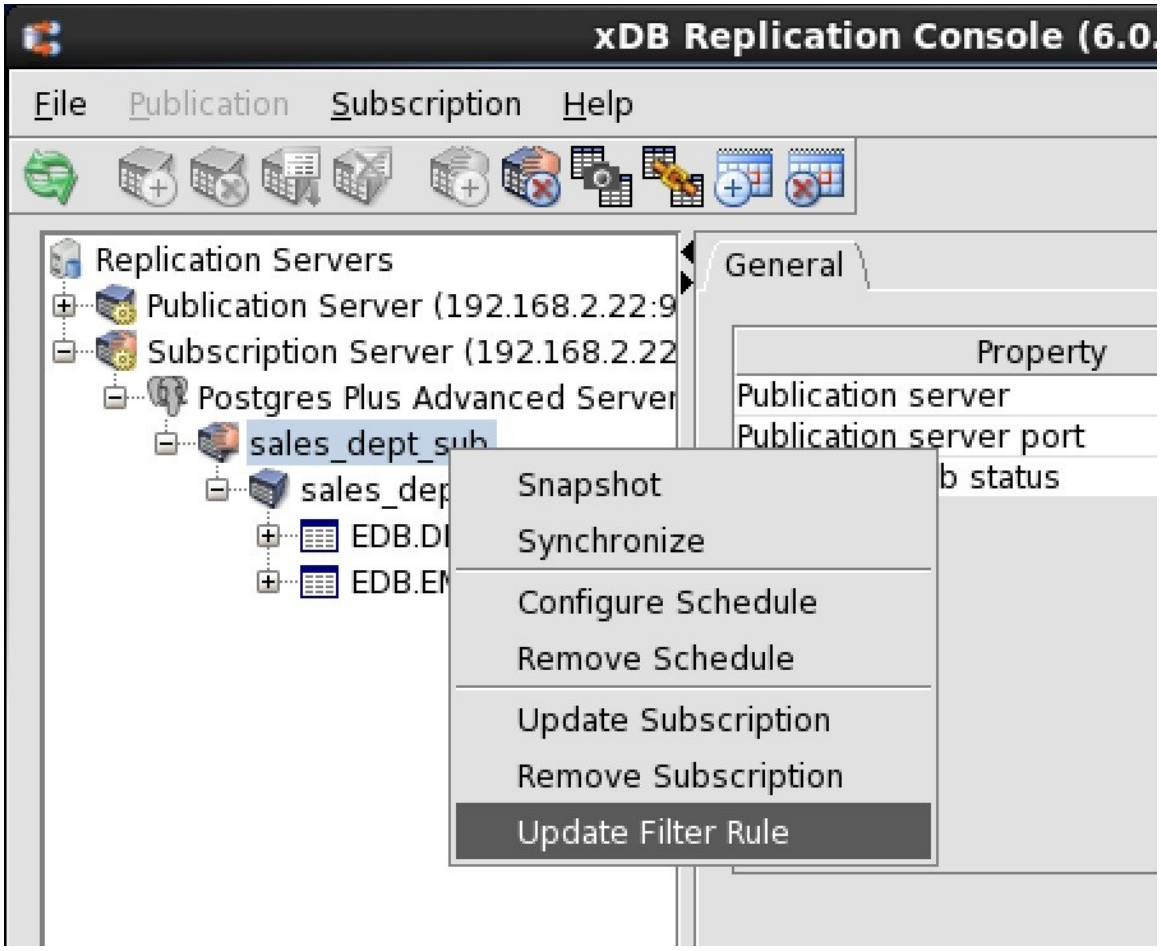


Figure 5-46 - Opening the Filter Rules tab on a subscription

Step 4: In the Filter Rules tab check or uncheck the boxes to specify the filter rules to enable or disable on the subscription. At most one filter rule may be enabled any given subscription table. Click the Update button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

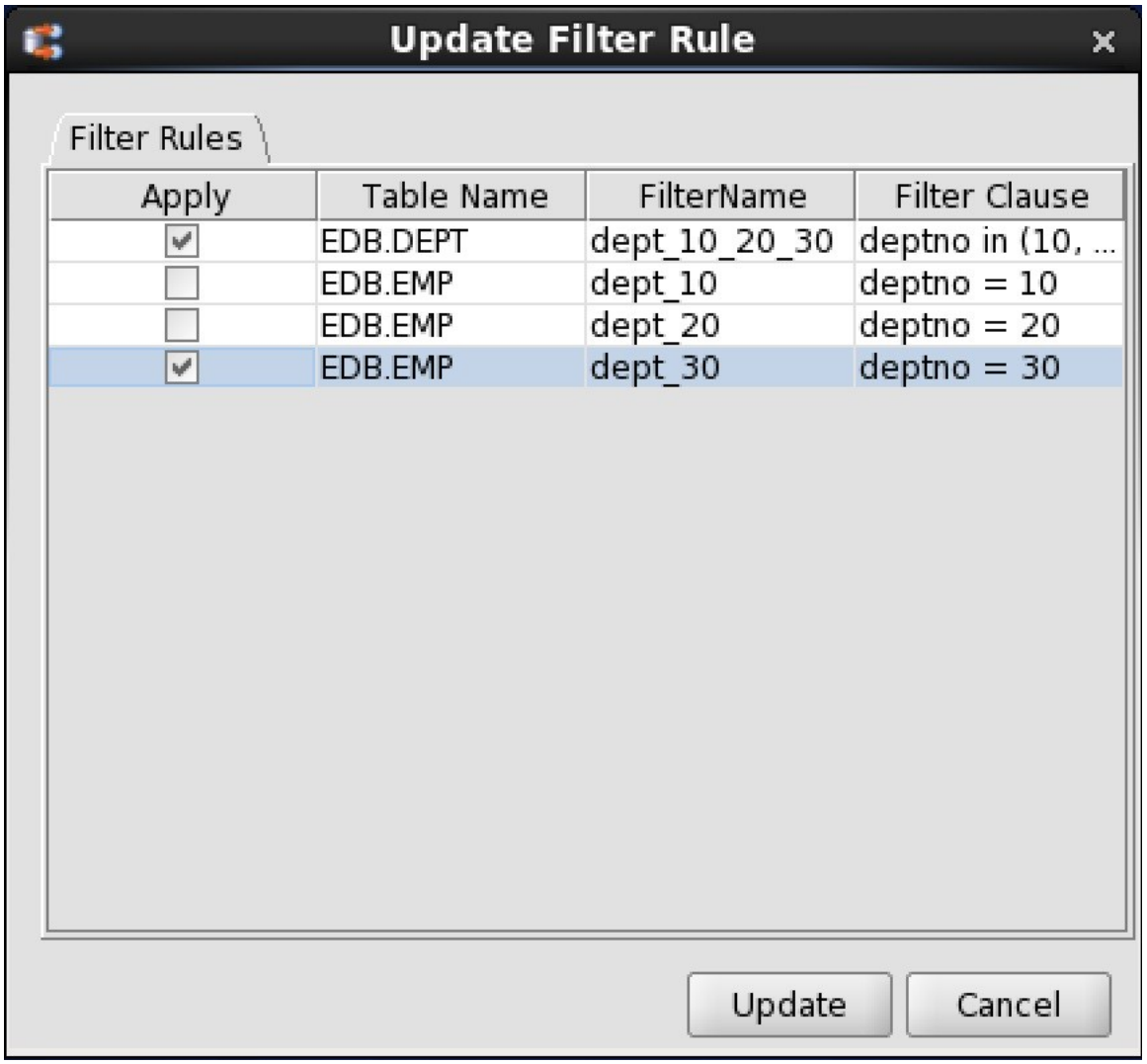


Figure 5-47 - Filter Rules tab

Step 5: A confirmation box appears presenting a warning message and a recommendation to perform a snapshot replication to any subscription on which you changed the filtering criteria.

Click the Ok button in the confirmation box to proceed with the update to the filter rule selections. Click the Cancel button to return to the Filter Rules tab if you wish to modify your filter rule selections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

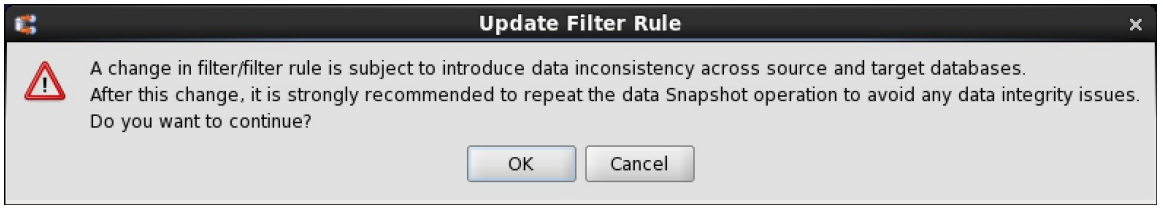


Figure 5-48 - Change filter rule confirmation

Step 6: If you clicked the Ok button in the preceding step, the Filter Rules updated successfully confirmation message appears if the update was successful.

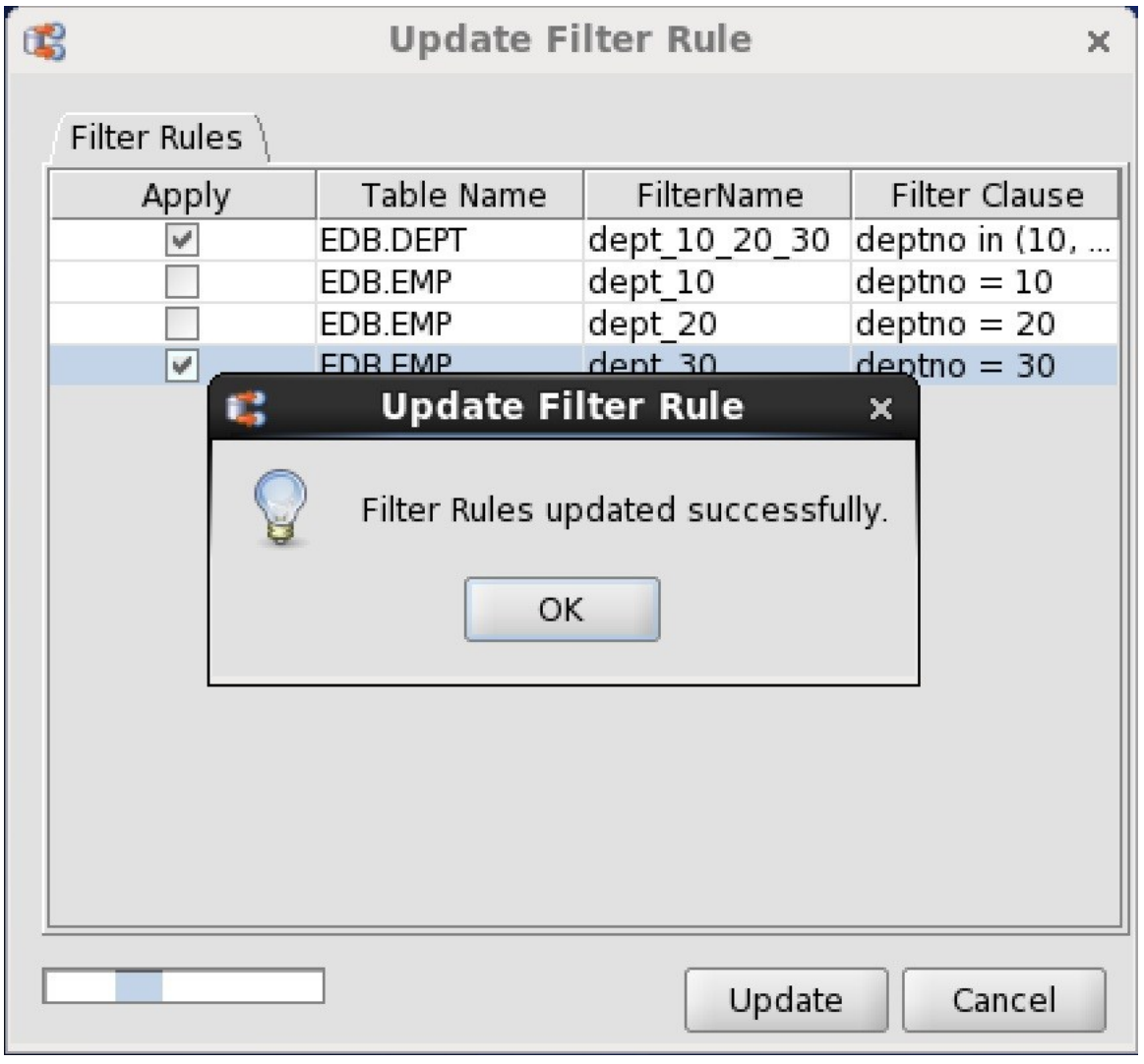


Figure 5-49 - Successful update of filter rules

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If you clicked the Cancel button in the preceding step, the Filter Rules tab reopens. You can modify your filter rule selections by repeating Step 4, or you can click the Cancel button in the Filter Rules tab to abort the filter rule updates on the subscription.

Step 7: It is strongly recommended that a snapshot replication be performed to the subscription that contains tables on which the filtering criteria has changed.

A snapshot ensures that the content of the subscription tables is consistent with the updated filtering criteria. See Section 5.4.1 for information on performing a snapshot replication.

5.5.5 Removing a Subscription

After a subscription is removed, replication can no longer occur for the publication that was associated with it until the publication is subscribed to with a new subscription.

Removing a subscription does not delete the subscription tables in the subscription database. It removes the identity and association of these tables to xDB Replication Server so the tables remain in the database until the DBA deletes them with `DROP TABLE` SQL statements.

Step 1: Make sure the subscription server whose node is the parent of the subscription you wish to remove is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 2: Select the Subscription node of the subscription that you wish to remove.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

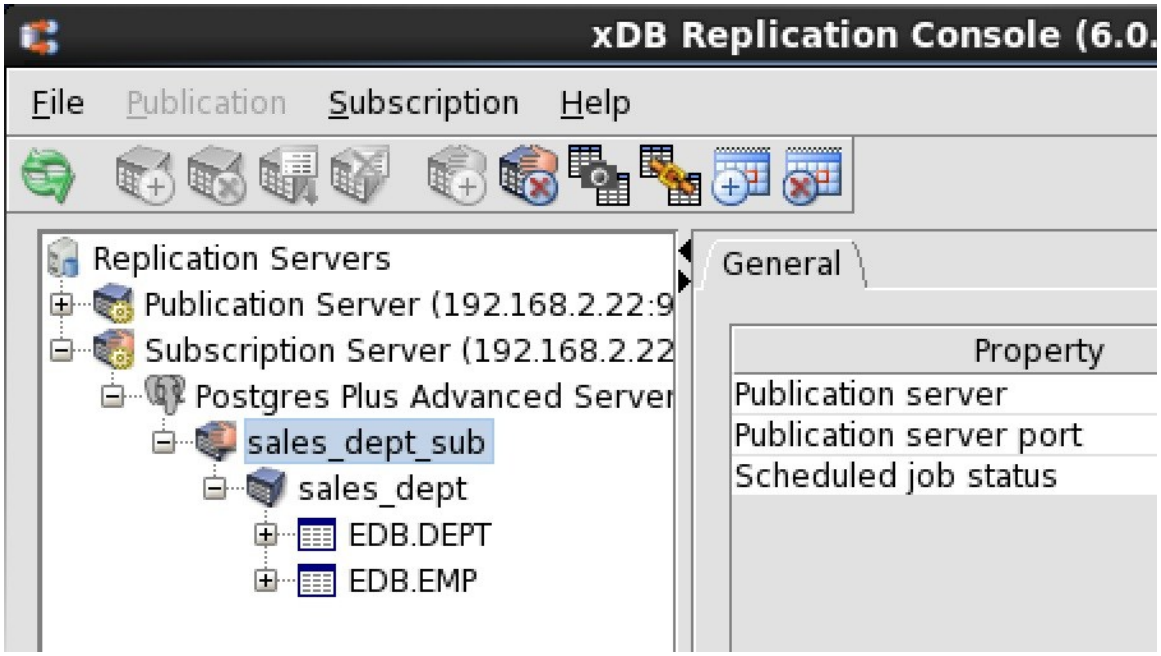


Figure 5-50 - Selecting a subscription to remove

Step 3: Remove the subscription in any of the following ways:

- Choose Remove Subscription from the Subscription menu.
- Click the secondary mouse button on the Subscription node and choose Remove Subscription.
- Click the primary mouse button on the Remove Subscription icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

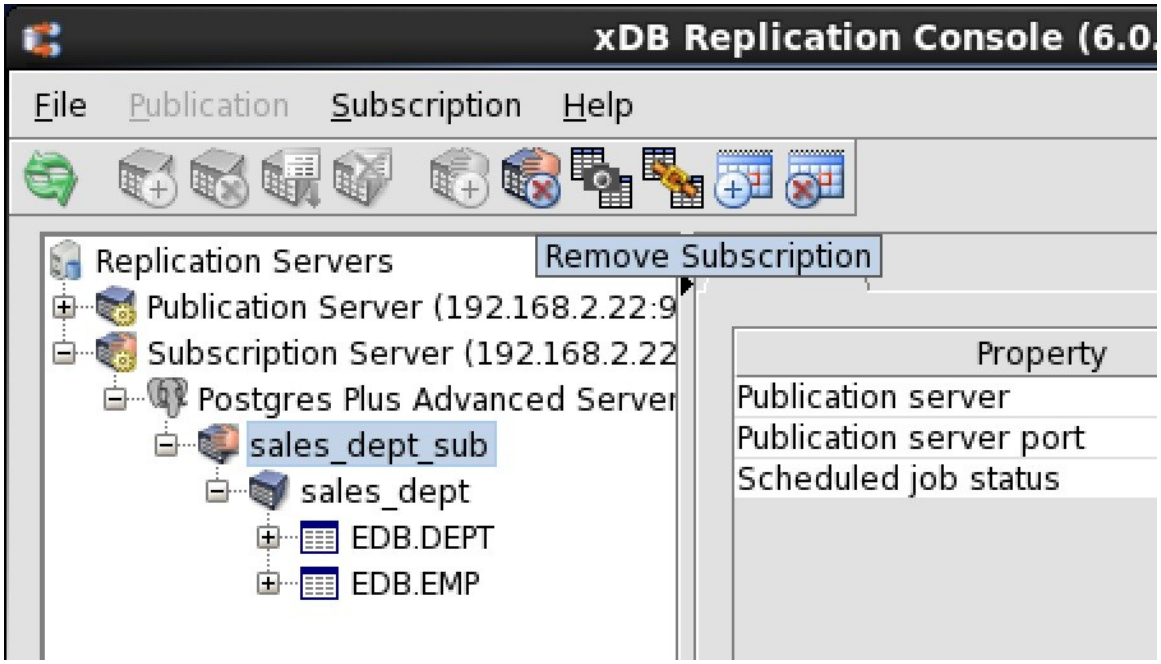


Figure 5-51 - Removing the subscription using the toolbar

Step 4: In the Remove Subscription confirmation box, click the Yes button.

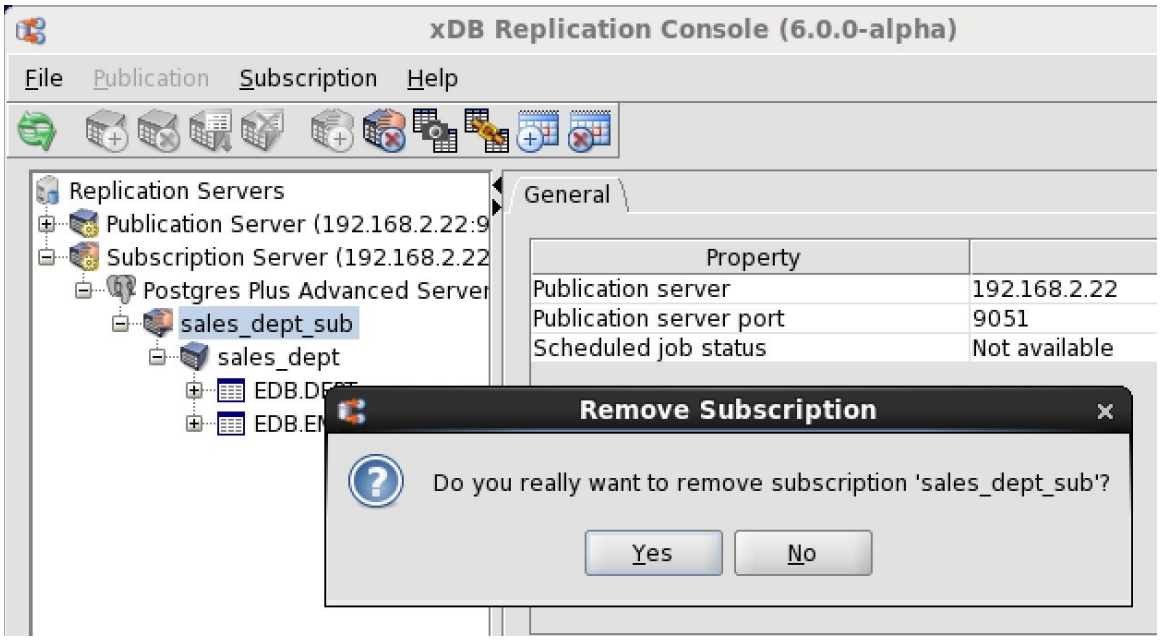


Figure 5-52 - Remove Subscription confirmation

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The Subscription node no longer appears under the Subscription Database node.

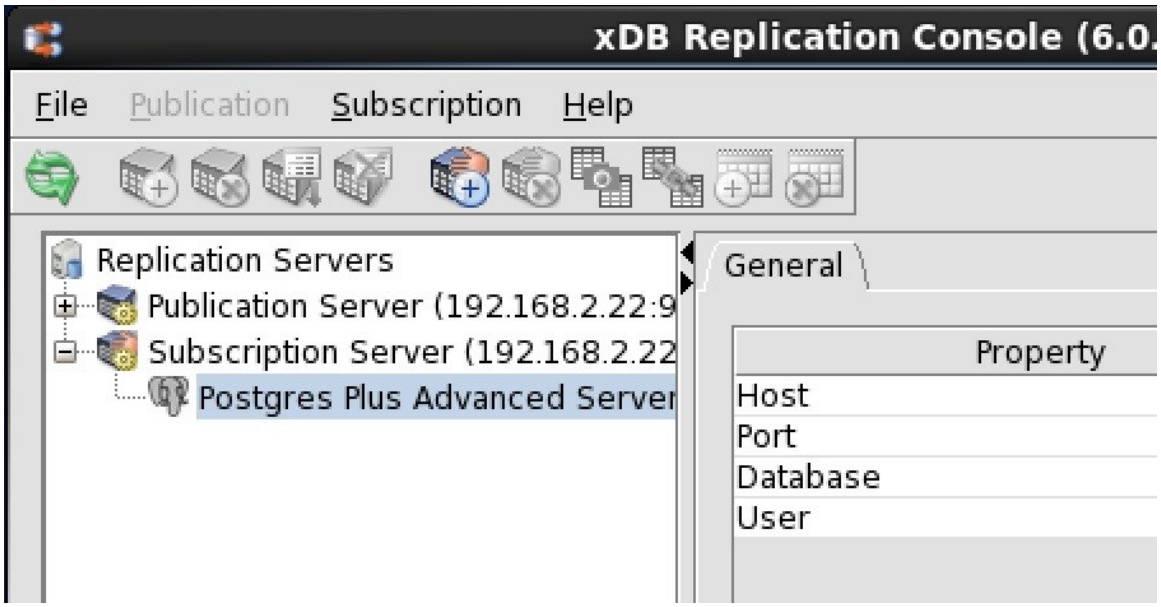


Figure 5-53 - Replication tree after removing a subscription

5.5.6 Removing a Subscription Database

Deleting a subscription database definition from xDB Replication Server is equivalent to removing its Subscription Database node. Before a Subscription Database node can be removed, all subscriptions under that Subscription Database node must be removed. See Section 5.5.5 for removing a subscription.

Removing a Subscription Database node does not delete the physical database from the database server. It removes the identity and association of the database to xDB Replication Server so no further replications can create or update tables in the database unless there are other subscription database definitions in xDB Replication Server with the same host and database identifier. The physical database can only be removed using the database management system's database removal procedures.

Step 1: Make sure the subscription server whose node is the parent of the subscription database definition you wish to remove is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Select the Subscription Database node that you wish to remove.

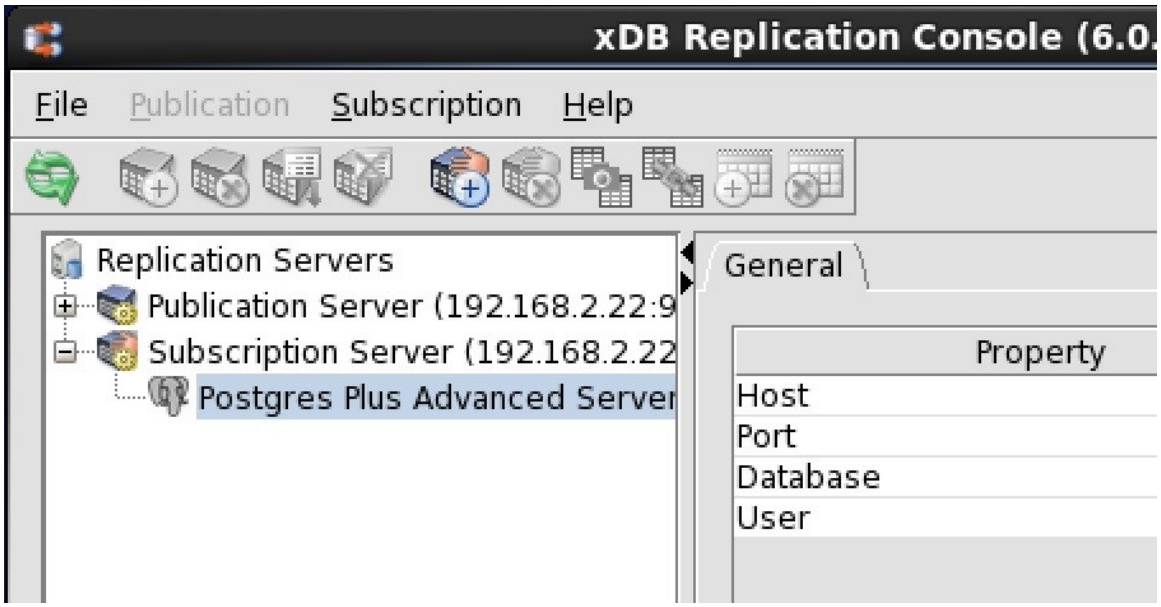


Figure 5-54 - Selecting a subscription database definition for removal

Step 3: From the Subscription menu, choose Subscription Database, then Remove Database. Alternatively, click the secondary mouse button on the Subscription Database node and choose Remove Subscription. The Remove Subscription Database confirmation box appears.

Step 4: In the Remove Subscription Database confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

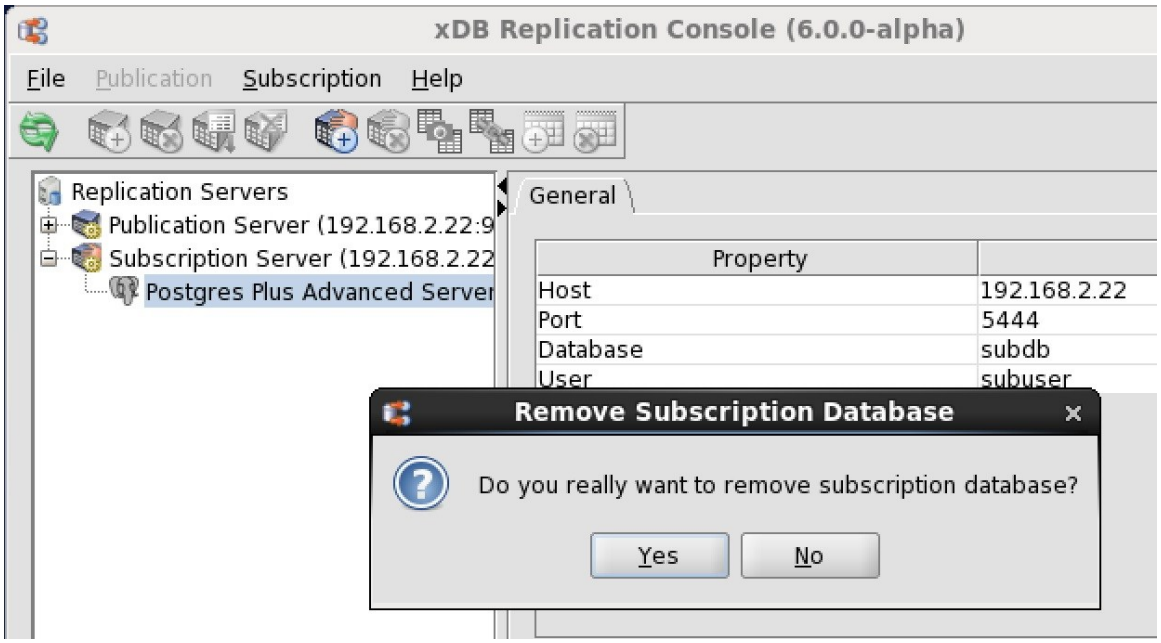


Figure 5-55 - Remove Subscription Database confirmation

The Subscription Database node no longer appears under the Subscription Server node.

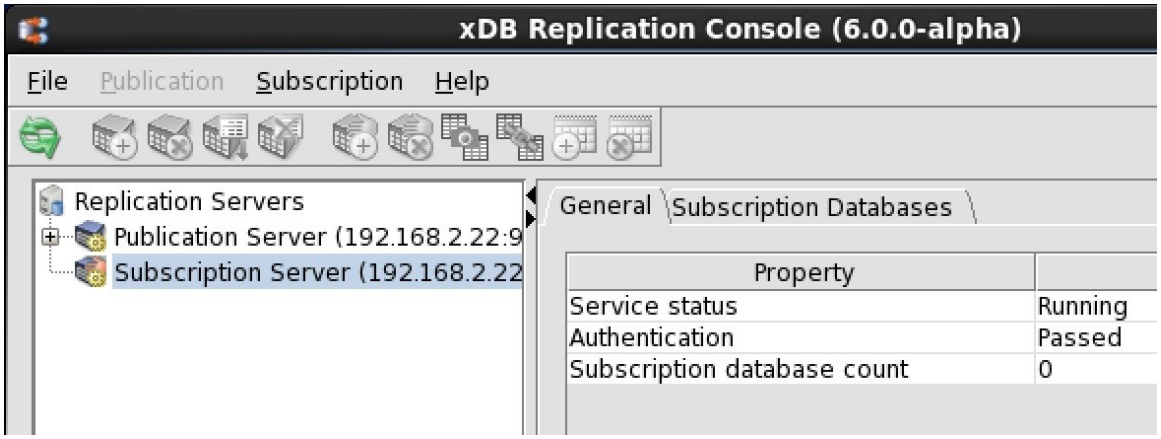


Figure 5-56 - Replication tree after removal of a subscription database

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.6 Performing Controlled Switchover

Controlled switchover is the exchanging of roles between a publication database and a subscription database. That is, the tables that were formerly publications become the subscription tables. The former subscription tables now become the publications.

Controlled switchover is useful for situations where the publication database must be taken offline such as for periodic maintenance. After the switchover, applications connect to the former subscription database to perform their queries and updates, while the former publication database is kept synchronized by replication.

Updates for replication are accumulated in shadow tables that are created on the former subscription tables during the controlled switchover procedure. When the former publication database is online, it is synchronized as the target of replication.

When you determine that you want to reverse the roles again so that the original publication database directly receives queries and updates from applications, and the original subscription database receives updates by replication, you perform the controlled switchover procedure once again, switching the roles back.

Note: This discussion assumes that the trigger-based method of synchronization replication is used by the publication database. If the publication database employs the log-based method, then it must be determined if the current subscription database meets the criteria for using the log-based method if that is so desired when it is switched to the role of the publication database. If the subscription database does not meet the criteria, then the trigger-based method must be implemented and used. See Section 2.2.10 for information on the log-based method and the necessary configuration steps that must be performed if the log-based method is to be used.

5.6.1 Controlled Switchover Overview

When you perform controlled switchover, you are modifying the replication system so that the database identified and referenced in the control schema as the publication database is the physical database that was originally defined as the subscription database.

Similarly, the database identified and referenced in the control schema as the subscription database is changed to the physical database on which the publication was originally defined.

You must also create the database objects on the former subscription database that xDB Replication Server uses to capture and store updates for replication.

In order to accomplish the controlled switchover, the following tasks must be performed:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Copy the control schema of the publication database (that is, all control schema objects, shadow tables, sequences, triggers, and a package) to the subscription database.
- Copy the control schema of the subscription database to the publication database.
- Update certain control schema tables so as to exchange the connection information for the publication database and subscription database. **These updates must be made in the control schema of all publication databases to ensure consistency of the control schema across all publication databases.**
- Modify the xDB Replication Configuration file to reference a new controller database if the former publication database was the designated controller database.

5.6.2 Controlled Switchover Steps

This section describes the steps for performing a controlled switchover.

The following assumptions are made about the replication system environment:

- "Node 1" is the server where the publication database originally resides. Its network IP address is `192.168.2.19`.
- "Node 2" is the server where the subscription database originally resides. Its network IP address is `192.168.2.20`.
- The publication and subscription databases have the same name.
- You use the publication database user for the role of the subscription database user and the subscription database user for the role of the publication database user in the switched environment.
- The publication server and subscription server are running on the same host (node 1).

Step 1: Stop all transaction processing against the publication database.

Step 2: Perform an on demand synchronization replication or a snapshot replication (for snapshot-only publications) in order to replicate any pending updates in the publication database shadow tables to the subscription database.

Step 3: Stop the publication server and the subscription server.

Step 4: Review the prerequisites in Section 5.1 to ensure that the subscription database and its host can be used in the role of a publication database, and the publication database and its host can be used in the role of a subscription database.

For practical purposes, the following items are the most likely to be affected:

- The publication database user must be a superuser with system catalog modification privileges to allow it to act as the new subscription database user.
- Additional entries may be needed in the `pg_hba.conf` files.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 5: Create a backup of schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` from the publication database on node 1.

Delete these schemas from the publication database on node 1 after the backup has been made.

Step 6: Create a backup of the replication triggers and their corresponding trigger functions on the publication tables on node 1. For the trigger-based method, these triggers are named with prefixes of `rrpd_`, `rrpi_` and `rrpu_`. The trigger functions are named with the same prefixes. For the log-based method, a trigger for each table is prefixed with `rrpt_`. The function is named `capturetruncateevent`.

Delete or disable these triggers on node 1.

Step 7: Create a backup of schema `_edb_replicator_sub` from the subscription database on node 2.

Delete this schema from the subscription database on node 2 after the backup has been made.

Step 8: Restore the backups of schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` created in Step 5 to the subscription database on node 2. Also restore the backup of the replication triggers and trigger functions created in Step 6 to the subscription database on node 2.

Step 9: Restore the backup of schema `_edb_replicator_sub` created in Step 7 to the publication database on node 1.

Step 10: Update the control schema objects so that the publication database definition references the new publication database (that is, the former subscription database) on node 2 and the subscription database definition references the new subscription database (that is, the former publication database) on node 1.

The connection information that may require updating includes the following:

- Host IP address
- Port number
- User name
- Password

These updates must be made in the control schema of all publication databases to ensure consistency of the control schema information should the controller database be switched at some later point in time.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For example, the following shows the update to the publication database definition so that its network IP address is now node 2 (`192.168.2.20`).

```
UPDATE _edb_replicator_pub.xdb_pub_database SET db_host = '192.168.2.20';
```

The following shows the update to the subscription database definition so that its network IP address is now node 1 (`192.168.2.19`).

```
UPDATE _edb_replicator_sub.xdb_sub_database SET db_host = '192.168.2.19';
```

Step 11: If you decide to use a publication server or subscription server on a new host, perform the following step, otherwise go to Step 12.

The following example assumes you decide to use the publication server and subscription server running on node 2.

Update the subscription metadata to the new location of the publication server managing its associated publication.

```
UPDATE _edb_replicator_sub.xdb_subscriptions SET pub_server_ip =
'192.168.2.20';
```

Update the publication metadata to the new location of the subscription server managing its associated subscription.

```
UPDATE _edb_replicator_pub.xdb_sub_servers SET sub_server_ip =
'192.168.2.20';
```

Step 12: Edit the xDB Replication Configuration file on the publication server and subscription server host so that it contains the controller database connection and authentication information for the new publication database now running on node 2.

The following is the modified xDB Replication Configuration file with the network location and authentication information of the new controller database now running on node 2.

```
#xDB Replication Server Configuration Properties
#Fri Jan 30 17:34:06 GMT-05:00 2015
port=5444
admin_password=ygJ9AxoJEX854elcVIJPTw\=\=
user=enterprisedb
admin_user=enterprisedb
type=enterprisedb
database=edb
password=ygJ9AxoJEX854elcVIJPTw\=\=
host=192.168.2.20
```

Step 13: Update the `pg_hba.conf` files of the database servers to allow access to the subscription database now on node 1 and the publication database now on node 2 in accordance with Section 5.1.6.3.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 14: When using the log-based method, create a replication slot on the database server that now contains the publication database.

Use the following query to obtain the slot name from the database server that was previously running the publication database, but is now the subscription database server:

```
SELECT slot_name FROM pg_replication_slots WHERE plugin = 'test_decoding';
  slot_name
-------------
 xdb_47919_5
(1 row)
```

Create a new replication slot on the database server that is now running the publication database, but was previously the subscription database server. The slot name from the previous query is used when creating the new replication slot.

```
SELECT pg_create_logical_replication_slot('xdb_47919_5', 'test_decoding');
pg_create_logical_replication_slot
-----------------------------------
 (xdb_47919_5,0/37A1270)
(1 row)
```

You may choose to keep the replication slot on the database server that now contains the subscription database, particularly if you plan to switch the publication and subscription databases back to their original roles at some future point. This eliminates the necessity for recreating the replication slot since it will still exist, but will be inactive until the publication is switched back to that database server.

Alternatively, you can delete the replication slot from the database server that now contains the subscription database. The replication slot is deleted with the following command:

```
SELECT pg_drop_replication_slot('xdb_47919_5');
```

See Section 10.3.4.4 for additional information on deleting the replication slot if the `pg_drop_replication_slot` function is not successful. If you switch back the databases to their original roles, you will just have to recreate the replication slot on the publication database server as previously described in this step.

Step 15: The controlled switchover is now complete. Start the publication server and the subscription server.

Step 16: After confirming that the publication tables are consistent with the subscription tables, the first replication operation must be a snapshot. After performing a snapshot, synchronization replications may be performed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

5.7 Performing Failover

Failover is the replacement of the publication database by the subscription database should a failure occur on the publication database or its host. Failover is considered an irreversible action so the subscription database permanently takes over the role of the publication database.

Generally, the same steps must be followed to perform a failover as was discussed for a controlled switchover in Section 5.6, however, the following points must also be taken into consideration:

- If the control schema objects on the publication database (that is, schemas `_edb_replicator_pub`, `_edb_replicator_sub`, `_edb_scheduler`, and their objects) cannot be salvaged or restored from a backup, then performing a failover may only be possible with the assistance of EnterpriseDB Technical Support Services.
- Pending updates not yet applied to the subscription may have been lost. The chances of this are greater if the interval between synchronizations is long.

If you determine that a failover is possible, follow the steps for a controlled switchover.

5.8 Optimizing Performance

Once you have become familiar with setting up and managing your replication system, you will often look for ways to optimize the performance of replications. This section discusses various publication server and subscription server configuration options available to improve the performance of snapshot and synchronization replications.

The publication server and subscription server configuration options are set in the publication server and subscription server configuration files, respectively. See Section 10.4.1 for a detailed explanation of how to set the configuration options in these files.

Note: Most of these configuration options are applicable to multi-master replication systems as well. Options applicable to multi-master replication systems are those that apply to the publication server and are not specific to a database product other than Postgres (such as an Oracle feature).

5.8.1 Optimizing Snapshot Replication

This section discusses configuration options for improving snapshot replication performance.

Note: The options described in this section apply to the publication server only and are set in the publication server configuration file unless otherwise specified.

copyViaDBLinkOra

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When the `copyViaDBLinkOra` option is set to `true`, the Oracle database link API, `dblink_ora`, is used instead of JDBC `COPY` to populate Advanced Server subscription tables from an Oracle publication during snapshot replication.

Oracle database link provides an additional performance improvement over JDBC `COPY`.

Note: The Oracle database link API feature is not available with PostgreSQL, therefore the `copyViaDBLinkOra` option is not applicable to PostgreSQL subscription tables.

Note: Prior to using `dblink_ora` with xDB Replication Server, there are a number of required configuration steps that must be performed in Advanced Server. For Advanced Server versions 9.3 or earlier, see the readme text file, `README-dblink_ora_setup.txt` located in the *`POSTGRES_INSTALL_HOME`*`/doc/contrib` directory for directions. For Advanced Server versions 9.4 or later, see Chapter dblink_ora in the *Database Compatibility for Oracle Developer's Guide* for directions.

```
copyViaDBLinkOra={true | false}
```

The default value is `false`.

useFastCopy

Set the `useFastCopy` option to `true` to skip Write-Ahead Log (WAL) logging during `COPY` operations in order to optimize data transfer speed.

The `archive_mode` configuration parameter in the `postgresql.conf` file of the target Postgres database server must be off (thereby disabling archiving of WAL data) in order to use the `useFastCopy` option.

```
useFastCopy={true | false}
```

The default value is `false`.

cpBatchSize

Use the `cpBatchSize` option to set the batch size (in Megabytes) that is used in the JDBC `COPY` operation during a snapshot. Increase the value of this option for large publication tables.

This option is influential when Postgres is the subscription database since the JDBC `COPY` operation is used to load Postgres subscription tables.

This option has no effect when Oracle or SQL Server is the subscription database. To tune loading of Oracle or SQL Server tables alter the `batchSize` option.

```
cpBatchSize=n
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The default value for *n* is 8.

batchSize

The batchSize option controls the number of INSERT statements in a JDBC batch.

This option is particularly significant when Oracle or SQL Server is the subscription database since tables of these database types are loaded using JDBC batches of INSERT statements.

For a Postgres subscription database, tables are loaded using JDBC COPY, however, if the COPY operation fails for some reason, then table loading is retried using JDBC batches of INSERT statements as in the case of Oracle and SQL Server.

```
batchSize=n
```

The default value for *n* is 100.

skipAnalyze

Set the skipAnalyze option to true if you want to skip execution of the ANALYZE command after loading Postgres subscription tables. The ANALYZE command gathers statistical information on the table contents. These statistics are used by the query planner.

```
skipAnalyze={true | false}
```

The default value is false.

snapshotParallelLoadCount

Note: To apply this option to a single-master replication system, it must be set for the subscription server within the subscription server configuration file. To apply this option to a multi-master replication system, it must be set for the publication server within the publication server configuration file.

The snapshotParallelLoadCount option controls the number of threads used to perform snapshot data replication in parallel mode. The default behavior is to use a single thread. However, if the target system architecture contains multi-CPUs/cores you can specify a value greater than 1, normally equal to the CPU/core count, to fully utilize the system resources.

```
snapshotParallelLoadCount=n
```

The default value is 1.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

lobBatchSize

If a table contains a column with a data type typically used for large objects such as `BYTEA`, `BLOB`, or `CLOB`, there is a greater possibility that a heap space error may occur because of a potentially large amount of data (hundreds of megabytes) brought into memory. In order to minimize the possibility of this error, a snapshot replication loads tables containing a large object data type, one row at a time using a single `INSERT` statement per batch.

If however, the large object data type column is known to contain relatively small amounts of data, you can increase the speed of a snapshot replication by increasing the value of the `lobBatchSize` option to allow a greater number of rows (specified by *n*) in each batch.

```
lobBatchSize=n
```

The default value is `1`.

5.8.2 Optimizing Synchronization Replication

This section discusses configuration options for improving synchronization replication performance.

In addition, for configuration options specifically applicable to publication databases configured with the log-based method of synchronization replication, see Section 10.4.1.15.

Note: The options described in this section apply to the publication server only and are set in the publication server configuration file.

5.8.2.1 Using Prepared SQL Statements

When synchronization replication occurs, the changes recorded in the shadow tables are applied to the subscription tables in JDBC batch updates. Within each batch, changes may be applied using either an individual SQL statement for each change; or a set of changes may be applied using a single, prepared SQL statement. A *prepared SQL* statement is parsed and compiled only once, but it can be executed multiple times using different values for certain components of the SQL statement in each execution. A SQL statement that is not prepared is parsed, compiled, and executed only once.

Prepared statements are useful only if the same type of SQL statement (`INSERT`, `UPDATE` or `DELETE`) is executed repeatedly and consecutively with the same target table, but with different values. If there is a sequence of consecutive changes that occur to the same table using the same operation such as inserting a set of rows into the same table populating the same columns, the publication server may apply these changes using a prepared statement. Otherwise, each change is applied with its own individual SQL statement.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

There are a number of server configuration options that control the characteristics of the JDBC batch along with if, when, and how often prepared statements are used. These are discussed in the following sections.

defaultBatchUpdateMode

The `defaultBatchUpdateMode` option controls whether the default mode is to use individual SQL statements in the JDBC batch update (this mode of operation is referred to as *BUS*) or to use prepared SQL statements in the JDBC batch update (this mode of operation is referred to as *BUP*).

```
defaultBatchUpdateMode={BUS | BUP}
```

The default value is `BUS`.

switchBatchUpdateMode

The `switchBatchUpdateMode` option controls whether or not the publication server dynamically switches between BUS mode and BUP mode during the replication process depending upon the type and sequence of updates it encounters in the shadow tables for the trigger-based method or the changeset stream for the log-based method.

```
switchBatchUpdateMode={true | false}
```

The default value is `true`.

This means using the default settings of `defaultBatchUpdateMode=BUS` and `switchBatchUpdateMode=true`, the publication server starts out by applying updates with individual SQL statements. When it encounters a stream of consecutive changes that can all be processed in a single prepared statement, it will switch to using prepared SQL statements.

Note: If you want a certain batch update mode used throughout all synchronization replications applied by a given publication server without switching update modes, set the `defaultBatchUpdateMode` option to the desired mode in combination with `switchBatchUpdateMode=false`. For example, if you only want prepared statements used, set the following options:

```
defaultBatchUpdateMode=BUP
switchBatchUpdateMode=false
```

Note: When Oracle is the subscription database, synchronization replication always occurs in BUP mode as if the preceding two options were always set. The reason for this is so large columns of `TEXT` data type from Postgres publications can successfully replicate to Oracle `CLOB` columns. In BUS mode an individual Oracle SQL statement has

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

a string literal maximum length of 4000 characters. This limitation does not occur for prepared SQL statements that are used in BUP mode.

busBatchThresholdCount

The `busBatchThresholdCount` option sets the number of consecutive updates of the same type that must be encountered in the shadow tables for the trigger-based method or the changeset stream for the log-based method before the publication server switches from BUS mode to BUP mode if dynamic switching is permitted (that is `switchBatchUpdateMode=true`).

```
busBatchThresholdCount=n
```

The default value for *n* is `5`.

The number of consecutive changes using the same table and SQL statement type must exceed the specified value *n* before a prepared statement is used.

Setting this threshold to a low value will encourage higher use of prepared statements while setting it to a high value will limit the use of prepared statements.

If changes to the publication were made using many SQL statements where each statement affected more than one row, then it may be beneficial to lower `busBatchThresholdCount` to encourage the use of prepared statements on the multiple shadow table rows resulting from each individual change on the publication.

bupBatchThresholdCount and bupBatchThresholdRepeatLimit

If BUP mode is employed, but the number of updates using the same prepared statement is low causing frequent switches to a new prepared statement, it may be more beneficial to use individual SQL statements (BUS mode).

For example, the following sequence of updates would be better processed in BUS mode:

```
INSERT INTO emp
INSERT INTO dept
INSERT INTO emp
INSERT INTO dept
DELETE FROM emp
UPDATE emp
UPDATE dept
INSERT INTO emp
INSERT INTO dept
DELETE FROM dept
INSERT INTO emp
DELETE FROM emp
INSERT INTO dept
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

However, in the following sequence, it is better to use BUP mode. Updates 1 thru 3 are batched in one prepared statement, 4 thru 7 in another prepared statement, 8 in its own prepared statement, and then 9 thru 15 in one prepared statement.

```
1. INSERT INTO emp
2. INSERT INTO emp
3. INSERT INTO emp
4. UPDATE dept
5. UPDATE dept
6. UPDATE dept
7. UPDATE dept
8. INSERT INTO emp
9. INSERT INTO dept
10. INSERT INTO dept
11. INSERT INTO dept
12. INSERT INTO dept
13. INSERT INTO dept
14. INSERT INTO dept
15. INSERT INTO dept
```

The `bupBatchThresholdCount` option is used in combination with the `bupBatchThresholdRepeatLimit` option to control the frequency of mode switches based on the volatility of expected update types to the publication.

```
bupBatchThresholdCount=m
```

The default value for *m* is `5`.

```
bupBatchThresholdRepeatLimit=n
```

The default value for *n* is `10`.

Each time the same prepared SQL statement is consecutively executed, an internal "batch" counter is incremented. If this batch count falls below `bupBatchThresholdCount` for the number of executions of a given prepared statement, then a second internal "repeat" counter is incremented by one. If the repeat counter eventually reaches `bupBatchThresholdRepeatLimit`, the update mode is switched from BUP to BUS.

Thus, if there are frequent, consecutive changes of prepared SQL statements (as measured against `bupBatchThresholdRepeatLimit`), each of which is executed a small number of times (as measured against `bupBatchThresholdCount`), then the mode of execution changes back to individual SQL statements instead of prepared statements.

Note: The publication server changes back to prepared statements when the threshold set by `busBatchThresholdCount` is met.

The following example illustrates the processing of updates when `bupBatchThresholdCount` is set to 3 and `bupBatchThresholdRepeatLimit` is set

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

to 4. A change to the "query domain" referred to in this example means a different statement type (`INSERT`, `UPDATE`, or `DELETE`) or a different target table are encountered in the next update, thus requiring the use of a different prepared SQL statement.

```
1. INSERT INTO emp
2. INSERT INTO emp
3. INSERT INTO dept
```

At this point the query domain is changed after the first two updates (change from table `emp` to `dept`) and the number of executions of the prior prepared statement (2) is less than `bupBatchThresholdCount`, so the repeat counter is set to 1.

```
4. INSERT INTO dept
5. INSERT INTO dept
6. INSERT INTO dept
7. INSERT INTO emp
```

The query domain is changed again (change from table `dept` to `emp`), but this time the number of executions (4) for the same query domain (updates 3 thru 6) exceeds `bupBatchThresholdCount` so the repeat counter is reset to 0.

```
8. INSERT INTO emp
9. UPDATE emp
```

The query domain is changed again (`INSERT` statement to `UPDATE` statement) and the number of executions (2) is less than `bupBatchThresholdCount`, so the repeat counter is incremented to 1.

```
10. UPDATE emp
11. INSERT INTO dept
12. DELETE FROM dept
13. INSERT INTO emp
```

The query domain is changed between updates 10 and 11, between updates 11 and 12, and between updates 12 and 13. At this point, the repeat counter has been incremented 3 more times to a value of 4. This now equals `bupBatchThresholdRepeatLimit`, so processing is changed from BUP mode to BUS mode.

5.8.2.2 Parallel Synchronization

Parallel synchronization takes advantage of multi-CPUs or cores in the system architecture by using multiple threads to apply transaction sets in parallel.

Parallel synchronization is applied in two ways:

- Multiple threads are used to load data for multiple tables in parallel from the source database. Each thread opens a separate connection therefore you will observe multiple connections with the source database. The pooling framework is used to cache the connections. After the threads are finished with the data load,

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

the idle connections are returned to the pool and remain there for a period of 3 minutes before being removed from the pool (as long as these are not reused).

- Changes are applied to multiple target databases in parallel. A transaction set from the source database is loaded only once. The target databases are updated in parallel from this loaded transaction set. When this transaction set has been applied to all targets (either successfully, or with failures on some targets), the next transaction set is loaded and applied in parallel. This aspect of parallel synchronization is particularly relevant to multi-master replication systems.

The following configuration options affect the usage of parallel synchronization.

syncLoadThreadLimit

The `syncLoadThreadLimit` option controls the maximum number of threads used to load data from source publication tables during parallel synchronization. The default count is 4. However, depending on the target system architecture (specifically, multi-CPUs/cores) you can choose to specify a custom count, normally equal to the CPU/core count, to fully utilize the system resources.

```
syncLoadThreadLimit=n
```

The default value is `4`.

dataSyncThreadCount

The `dataSyncThreadCount` option controls the maximum number of threads used to apply incremental changes during synchronization replication to the target slave databases (for single-master replication systems) or to the target master nodes (for multi-master replication systems) in parallel mode. The default behavior (when `dataSyncThreadCount` is set to 0) is to use as many threads as there are target nodes.

However, depending on the target system architecture (specifically, multi-CPUs/cores) you can choose to specify a custom count, normally equal to the CPU/core count, to fully utilize the system resources.

```
dataSyncThreadCount=n
```

The default value is `0`.

targetDBQueryTimeout

The `targetDBQueryTimeout` option controls the timeout interval (in milliseconds) before an attempt by the publication server to apply a transaction set on a target database is aborted by the database server (typically due to a lock acquired by another application on one or more of the target tables).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The `targetDBQueryTimeout` option sets the default lock timeout value to 10 minutes. Change the 10 minute default value to a higher value if you want to allow a longer wait time before the transaction is aborted. Change the value to 0 if you want to turn off usage of the `targetDBQueryTimeout` option in which case the timeout interval is controlled by the setting of the Postgres database server `statement_timeout` configuration parameter.

A higher value of `targetDBQueryTimeout` delays processing of subsequent transaction sets on other target databases because if a transaction set is blocked, the next transaction set cannot be loaded until: 1) the lock is released and the blocked transaction set can then be applied to completion, or 2) the `targetDBQueryTimeout` interval is exceeded.

If a timeout occurs, the waiting transaction set is marked as aborted for the particular blocked target database. The remaining pending transaction sets in this synchronization session are skipped for this target database, but are applied to all other target databases once the timeout interval has been exceeded. The aborted and skipped transaction sets are tried again when the next synchronization replication event occurs.

So for example, in a 3-node cluster with ten pending transaction sets, assume transaction set 1 is loaded and begins replicating to nodes 2 and node 3. Now, another application acquires a lock on one or more tables in node 2, putting the updates to these tables in a wait state. Replication of transaction set 1 can run to completion on node 3, but if the wait time exceeds the `targetDBQueryTimeout` interval, the database server cancels transaction set 1 on node 2. Replication of this transaction set to node 2 is marked as aborted in the xDB Replication Server metadata.

Transaction set 2 can now be loaded and run against node 3. Execution of transaction set 2 against node 2 is skipped since transaction sets must be applied in order and transaction set 1 was not successfully applied to node 2. Transaction sets 3 thru 10 are loaded and applied in order against node 3, but skipped for node 2.

In the next synchronization replication, transaction set 2 is tried again on node 2. If the lock has been released and the transaction set is applied successfully, the remaining transaction sets 3 thru 10 are applied to node 2.

Finally, synchronization replication continues with any new transaction sets.

```
targetDBQueryTimeout=n
```

The default value is `600000`.

5.8.2.3 Other Synchronization Configuration Options

The following are other configuration options affecting synchronization replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

syncBatchSize

The `syncBatchSize` option controls the number of statements in a synchronization replication JDBC batch.

```
syncBatchSize=n
```

The default value for *n* is `100`.

syncFetchSize

The `syncFetchSize` option controls how many rows are fetched from the publication database in one network round-trip. For example, if there are 1000 pending row changes, the default fetch size requires 5 database round-trips. Using a fetch size of 500 retrieves all changes in 2 round trips.

Fine tune the performance by using a fetch size that conforms to the average data volume consumed by rows fetched in one round trip.

```
syncFetchSize=n
```

The default value for *n* is `200`.

txSetMaxSize

The `txSetMaxSize` option defines the maximum number of transactional rows that can be grouped in a single transaction set. The publication server loads and processes the changes by fetching as many rows in memory as grouped in a single transaction set.

A higher value is expected to boost performance. However a very large value might result in an out of memory error. Increase/decrease the value in accordance with the average row size (low/high).

```
txSetMaxSize=n
```

The default value for *n* is `10000`.

enablePerformanceStats

Set `enablePerformanceStats` option to `true` only if you need to conduct performance testing and analyze the replication statistics. When enabled, the publication server creates additional triggers on the publication tables in each master node. The triggers produce transaction statistics that are recorded in the `mmr_transaction_history` table in the control schema. **This option should be disabled in a production environment to avoid performance overhead.**

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
enablePerformanceStats={true | false}
```

The default value is `false`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6 Multi-Master Replication Operation

This chapter describes how to configure and run xDB Replication Server for multi-master replication systems.

For configuration and management of your replication system, the xDB Replication Console graphical user interface is used to illustrate the steps and examples in this chapter. The same steps can be performed from the operating system command line using the xDB Replication Server Command Line Interface (CLI). The commands of the xDB Replication Server CLI utility are described in Chapter 8.

6.1 Prerequisite Steps

Certain steps must be taken to prepare the host environments as well as the database servers used as master nodes before beginning the process of building a multi-master replication system. This section describes these steps.

6.1.1 Setting Heap Memory Size for the Publication Server

Replication speed and efficiency can be affected by the heap memory size set for the publication server. The xDB Startup Configuration file sets a parameter controlling the minimum and maximum heap size allocated for the publication server. See Section 5.1.1 for guidelines and information on setting this parameter.

6.1.2 Enabling Synchronization Replication with the Log-Based Method

This section applies only to Postgres database servers of version 9.4 and later.

If you plan to use the log-based method of synchronization replication with any master node running under a given Postgres database server, the following configuration parameter settings are required in the configuration file, `postgresql.conf`, of each such Postgres database server:

- **wal_level.** Set to `logical`.
- **max_wal_senders.** Specifies the maximum number of concurrent connections (that is, the maximum number of simultaneously running WAL sender processes). Set at minimum, to the number of MMR master nodes on this database server that will use the log-based method. In addition, if SMR publication databases are to run on this database server, also add the number of SMR publication databases that will use the log-based method.
- **max_replication_slots.** Specifies the maximum number of replication slots. For support of MMR systems, the minimum is the total number of master nodes in the multi-master replication system multiplied by the number of master nodes residing on this database server. For information, see Section 2.2.10.4. In

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

addition, if SMR publication databases are to run on this database server, also add the number of SMR publication databases that will use the log-based method.

- **track_commit_timestamp.** Set to `on`. This configuration parameter applies only to Postgres database servers of version 9.5. See Section 6.6.1 for additional information.

See Section 2.2.10 for information on the log-based method of synchronization replication.

The Postgres database server must be restarted after altering any of these configuration parameters.

In addition, the `pg_hba.conf` file requires an entry for each publication database user of master nodes that are to use the log-based method. Such database users must be included as a replication database user in the `pg_hba.conf` file. See Section 6.1.5 for additional information.

6.1.3 Preparing the Master Definition Node

This section discusses the preparation of a database to be used as the master definition node.

When creating the publication database definition for the master definition node, a database user name must be specified that has the following characteristics:

- The database user can connect to the master definition node.
- The database user has superuser privileges. Superuser privileges are required because the database configuration parameter `session_replication_role` is altered by the database user when the master definition node receives updates from other master nodes during a synchronization replication. The database user temporarily changes `session_replication_role` to `replica` to prevent the triggers on the publication tables from firing. This session change also occurs for snapshot operations involving replication of the control schema from one publication database to another.
- The database user must have the ability to modify the system catalog tables in order to disable foreign key constraints on the publication tables for snapshots targeted to the publication, as well as for the control schema tables for snapshot operations involving the replication of the control schema from one publication database to another. (See appendix Section 10.4.4 for more information on this requirement.)

The examples used throughout the rest of this user's guide are based on the following master definition node:

- The database user name for the master definition node is `pubuser`.
- The tables used in the publication reside in a schema named `edb`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Three tables named `dept`, `emp`, and `jobhist` are members of schema `edb`.
- The database name of the master definition node is `edb`.

The following steps illustrate the preparation of the master definition node database user.

Step 1: Create a user name with login and superuser privileges for the master definition node. This user becomes the owner of xDB Replication Server metadata database objects that will be created in the master definition node to track, control, and record the replication process and history. The xDB Replication Server metadata database objects are created in a schema named `_edb_replicator_pub`.

When creating the publication database definition, the database user name is entered in the Publication Service – Add Database dialog box (see Section 6.2.2).

```
CREATE ROLE pubuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

Step 2 (Optional): If users are to access the data in the publication tables residing on this master node, it is convenient to have one or more "group" roles containing the required privileges to access these tables. Privileges must also be granted on the control schema objects to users who are to perform inserts, updates, or deletions on the publication tables.

When adding new users, granting these users membership in these roles gives them the privileges to access the publication tables. This eliminates the need to grant these privileges individually to each new user.

See Step 2 of Section 5.1.4.3 for information on creating such roles.

6.1.4 Preparing Additional Master Nodes

The following steps illustrate the creation of a database user and a database for an additional master node.

When creating the publication database definition for an additional master node, a database user name must be specified that has the following characteristics:

- The database user can connect to the master node.
- The database user has superuser privileges. Superuser privileges are required because the database configuration parameter `session_replication_role` is altered by the database user when the master node receives updates from other master nodes during a synchronization replication. The database user temporarily changes `session_replication_role` to `replica` to prevent the triggers on the publication tables from firing. This session change also occurs for snapshot operations involving replication of the control schema from one publication database to another.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The database user must have the ability to modify the system catalog tables in order to disable foreign key constraints on the publication tables for snapshots targeted to the publication, as well as for the control schema tables for snapshot operations involving the replication of the control schema from one publication database to another. (See appendix Section 10.4.4 for more information on this requirement.)
- If the additional master node is to reside on a different database server instance (that is, on a different host or port number) than the master definition node, then the same database user name should be used for this additional master node as used for the master definition node unless the publication server configuration option `skipTablePrivileges` is changed from its default value of `false` to `true`. See Section 10.4.1.13 for information on `skipTablePrivileges`.

There are also two possible options available with respect to how the publication tables are to be created in the master node:

- Allow the publication server to create the publication table definitions in the master node by copying the definitions from the master definition node at the time you add the publication database definition for the master node.
- Define the publication tables in the master node beforehand by running SQL DDL statements in the PSQL command line utility program or by using Postgres Enterprise Manager Client to create the tables.

If you create the table definitions "manually" as described in the second bullet point, be sure the publication tables are defined identically to the tables in the master definition node including schema names, table names, number of columns, column names, column data types, column lengths, primary key definitions, unique constraints, foreign key constraints, etc.

The examples used throughout the rest of this user's guide are based on the following:

- The database user name of the second master node is `mmruser`.
- The database name of the second master node is `mmrnode`.

Step 1: Create a database user name for the master node. This user becomes the owner of xDB Replication Server metadata database objects that will be created in the master node to track, control, and record the replication process and history. The xDB Replication Server metadata database objects are created in a schema named `_edb_replicator_pub`.

When creating the publication database definition for the master node, the database user name is entered in the Publication Service – Add Database dialog box (see Section 6.3).

```
CREATE ROLE mmruser WITH LOGIN SUPERUSER PASSWORD 'password';
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Create a database that will be used as the master node if such a database does not already exist.

```
CREATE DATABASE mmrnode;
```

6.1.5 Verifying Host Accessibility

If more than one computer is used to host the components of the replication system, each computer must be able to communicate with the others on a network.

There are a number of different aspects to this topic.

- For a discussion of firewalls and access to ports see Section 5.1.6.1.
- For a discussion of network IP addresses see Section 5.1.6.2.

A Postgres database server uses the host-based authentication file, `pg_hba.conf`, to control access to the databases in the database server.

You need to modify the `pg_hba.conf` file on each Postgres database server that contains a master node.

In a default Postgres installation, this file is located in the directory *`POSTGRES_INSTALL_HOME`*`/data`.

The modification needed to the `pg_hba.conf` file is discussed in the following section.

Master Nodes

On each database server running a master node, the following is needed to allow access to the database:

```
host masternode_db   masternode_user   pub_ipaddr/32   md5
```

The value you substitute for *`masternode_db`* is the name of the database you intend to use as the master node. The value you substitute for *`masternode_user`* is the database user name you created in Step 1 of Section 6.1.3 or Step 1 of Section 6.1.4.

For two master nodes using databases named `edb` and `mmrnode` running on the same database server, the resulting `pg_hba.conf` file appears as follows:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                             md5
# IPv4 local connections:
host    edb         pubuser       192.168.2.22/32     md5
host    mmrnode     mmruser       192.168.2.22/32     md5
host    all         all           127.0.0.1/32        md5
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
# IPv6 local connections:
host    all         all            ::1/128            md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                      md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
```

If the master node using database `mmrnode` with database user name `mmruser` is running on a separate host than where database `edb` is running, the `pg_hba.conf` file on the database server with database `mmrnode` would look like the following:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                             md5
# IPv4 local connections:
host    mmrnode     mmruser       192.168.2.22/32     md5
host    all         all           127.0.0.1/32        md5
# IPv6 local connections:
host    all         all           ::1/128             md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                     md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
```

The preceding examples assume databases `edb` and `mmrnode` are using the trigger-based method of synchronization replication. If the log-based method is used, the `pg_hba.conf` file must contain additional entries with the DATABASE field set to `replication` for *`masternode_user`* and *`pub_ipaddr`* to allow replication connections from the publication server on the host on which it is running.

The following shows a modification of the first example with these additional entries as the last two lines in the file:

```
# TYPE  DATABASE    USER          ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local   all         all                             md5
# IPv4 local connections:
host    edb         pubuser       192.168.2.22/32     md5
host    mmrnode     mmruser       192.168.2.22/32     md5
host    all         all           127.0.0.1/32        md5
# IPv6 local connections:
host    all         all           ::1/128             md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication  enterprisedb                     md5
#host    replication  enterprisedb  127.0.0.1/32       md5
#host    replication  enterprisedb  ::1/128            md5
host    replication  pubuser       192.168.2.22/32     md5
host    replication  mmruser       192.168.2.22/32     md5
```

See sections 2.2.10 and 6.1.2 for additional information on synchronization replication with the log-based method.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Reload the configuration file after making the modifications.

Choose Reload Configuration (Expert Configuration, then Reload Configuration on Advanced Server) from the Postgres application menu. This will put the modified `pg_hba.conf` file into effect.

6.2 Creating a Publication

Creating a publication requires the following steps:

- Registering the publication server
- Adding the master definition node
- Creating a publication by choosing the tables for the publication along with the conflict resolution options
- Defining table filters to be enabled on any master nodes

6.2.1 Registering a Publication Server

Registering a publication server is done in a manner identical to single-master replication. See Section 5.2.1 for directions on registering a publication server.

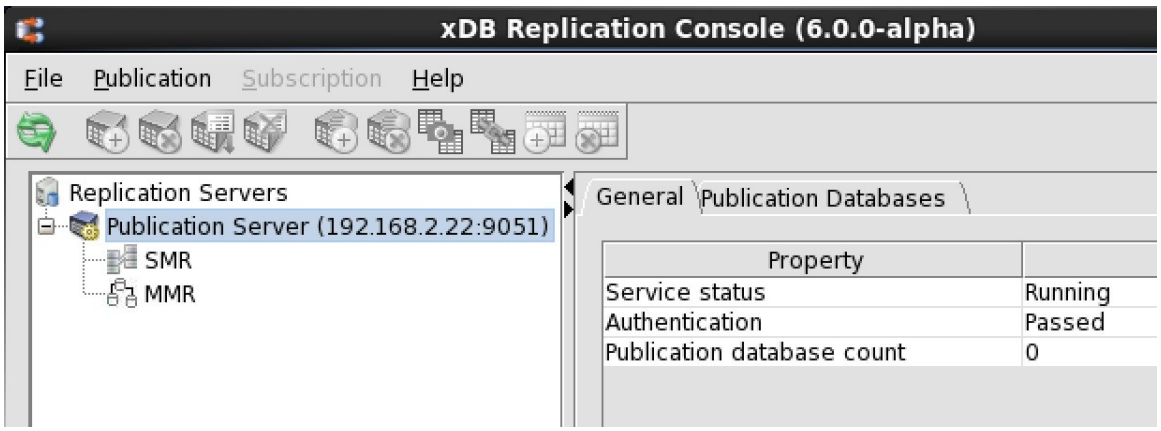


Figure 6-1 - Replication tree after registering a publication server

After you have successfully registered a publication server, the replication tree of the xDB Replication Console displays a Publication Server node under which are the SMR and MMR type nodes.

Continue to build the multi-master replication system under the MMR type node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.2.2 Adding the Master Definition Node

The first database to be identified to xDB Replication Server is the master definition node. This is done by creating a publication database definition subordinate to the MMR type node under the Publication Server node.

After the publication database definition is created, a Publication Database node representing the master definition node appears in the replication tree of the xDB Replication Console. A publication containing tables residing within this database can then be created under the Publication Database node.

You must enter database connection information such as the database server network address, database identifier, and database login user name and password when you create the publication database definition. The connection information is used by the publication server to access the publication tables when it performs replication.

Step 1: Make sure the database server for the master definition node is running and accepting client connections.

Step 2: Select the MMR type node under the Publication Server node. From the Publication menu, choose Publication Database, and then choose Add Database. Alternatively, click the secondary mouse button on the MMR type node and choose Add Database. The Publication Service – Add Database dialog box appears.

Step 3: Fill in the following fields:

- **Database Type.** Select PostgreSQL or Postgres Plus Advanced Server for the master definition node. For an Advanced Server Oracle compatible installation, select the Postgres Plus Advanced Server option. For PostgreSQL or an Advanced Server PostgreSQL compatible installation, select the PostgreSQL option.
- **Host.** IP address of the host on which the master definition node is running.
- **Port.** Port on which the master definition node is listening for connections.
- **User.** The database user name for the master definition node created in Step 1 of Section 6.1.3.
- **Password.** Password of the database user.
- **Database.** Enter the database name of the master definition node.
- **URL Options (For SSL connectivity).** Enter the URL options to establish SSL connectivity to the master definition node. See Section 7.11 for information on using SSL connections.
- **Changeset Logging (For Postgres).** Select Table Triggers to use the trigger-based method of synchronization replication. Select WAL Stream to use the log-based method of synchronization replication. See Section 2.2.9 for information on the trigger-based method. See Section 2.2.10 for information on the log-based method.
- **Node Priority Level.** An integer from 1 to 10, which is the priority level assigned to this master node for conflict resolution based on node priority. The highest

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

priority is 1 while the lowest is 10. See Section 6.6.4 for information on conflict resolution strategies. The default is 1 for the master definition node.

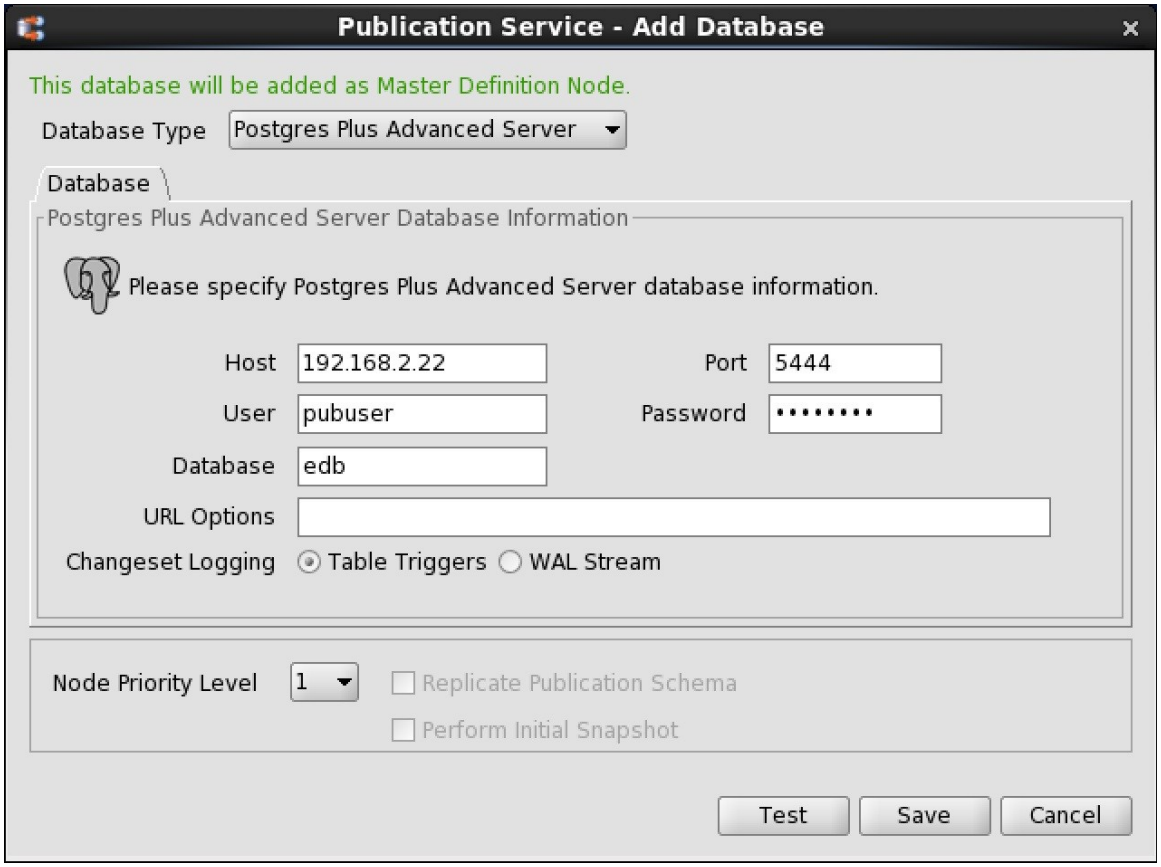


Figure 6-2 - Publication Service - Add Database dialog box for the master definition node

Step 4: Click the Test button. If Test Result: Success appears, click the OK button, then click the Save button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

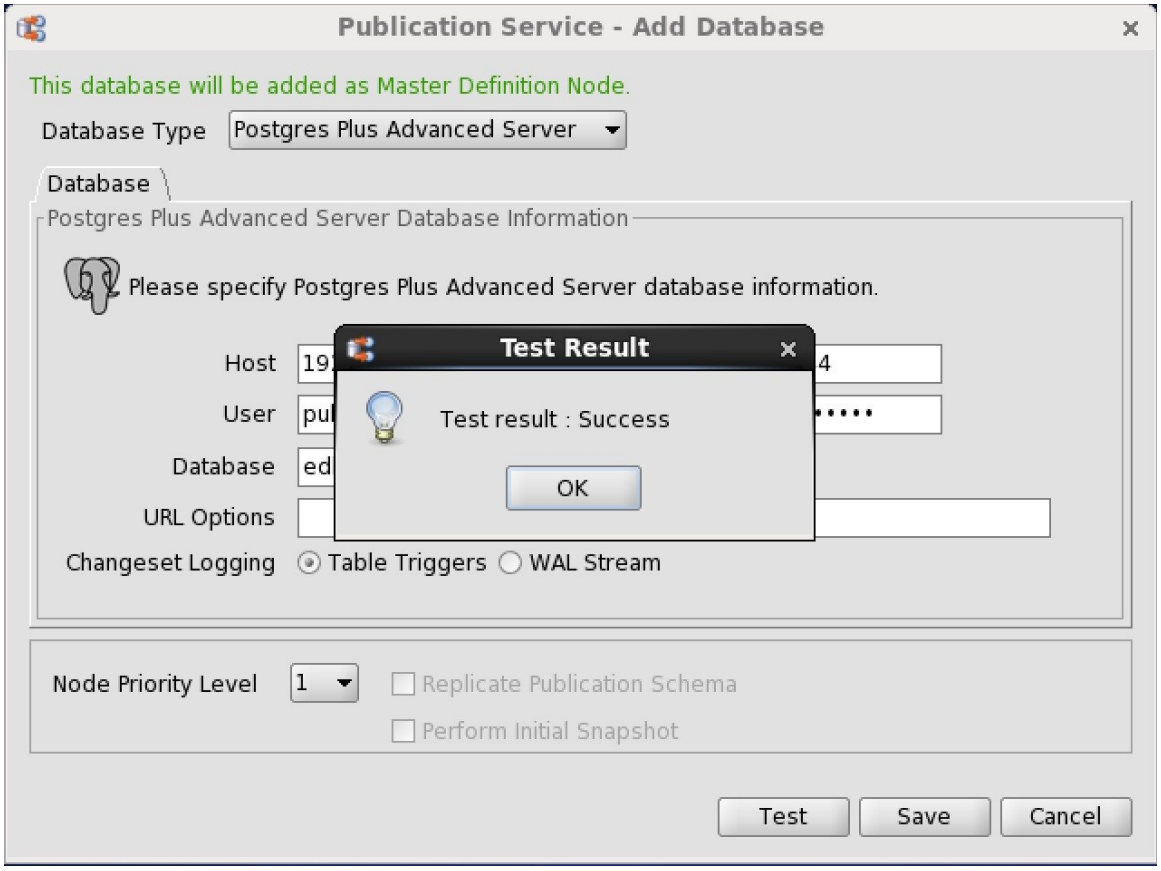


Figure 6-3 - Successful master definition node test

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 4.

When the publication database definition is successfully saved, a Publication Database node is added to the replication tree under the MMR type node of the Publication Server node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

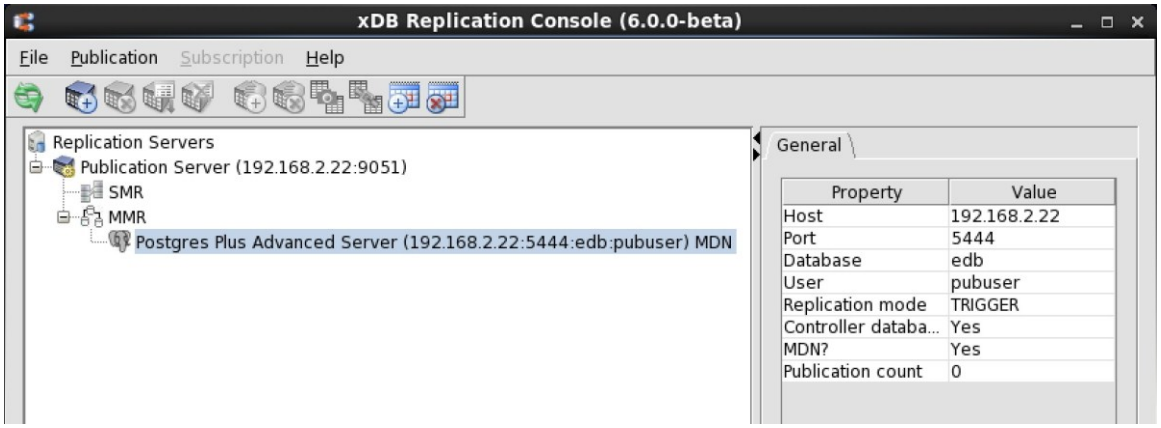


Figure 6-4 - Replication tree after adding the master definition node

The label `MDN` appears at the end of the node in the replication tree and in addition, the MDN field is set to `Yes` in the Property window to indicate this is the master definition node.

6.2.3 Adding a Publication

Subordinate to the master definition node, you create a publication that contains tables of the database.

Step 1: Select the Publication Database node. From the Publication menu, choose Create Publication. Alternatively, click the secondary mouse button on the Publication Database node and choose Create Publication. The Create Publication dialog box appears.

Step 2: Fill in the following fields under the Create Publication tab:

- **Publication Name.** Enter a name that is unique amongst all publications.
- **Publish.** Check the boxes next to the tables that are to be included in the publication. Alternatively or in addition, click the Use Wildcard Selection button to use wildcard pattern matching for selecting publication tables.
- **Select All.** Check this box if you want to include all tables in the Available Tables list in the publication.
- **Use Wildcard Selection.** Click this button to use the wildcard selector to choose tables for the publication. See Section 7.1 for information on the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

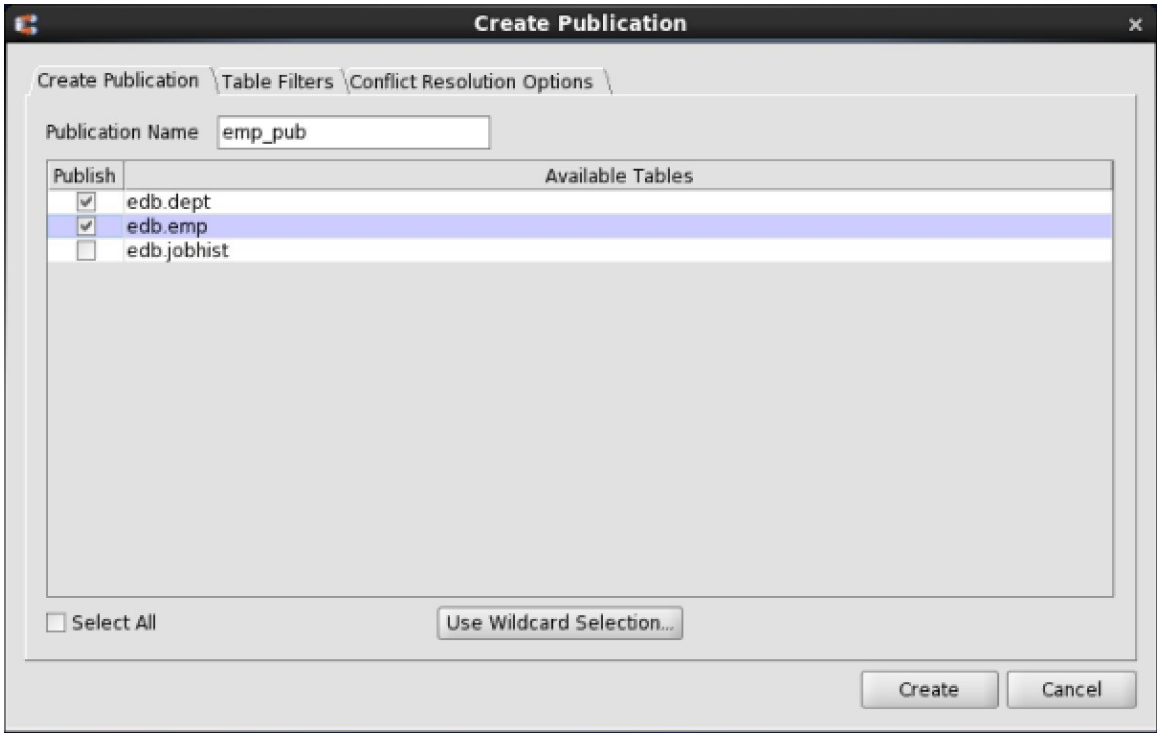


Figure 6-5 - Create Publication dialog box

If you wish to use table filters during replications between master nodes in this multi-master replication system, follow the directions in the next step to define the initial set of available table filters, otherwise go on to Step 4.

Step 3 (Optional): *Table filters* consist of a set of filter rules that control the selection criteria for rows replicated between master nodes during a snapshot or a synchronization replication.

Note: See Section 2.2.12.3 for table setup requirements for a log-based replication system as well as general restrictions on the use of table filters.

A *filter rule* consists of a *filter name* and a SQL `WHERE` clause (omitting the `WHERE` keyword) called the *filter clause*, which you specify for a table that defines the selection criteria for rows that are to be included during a replication.

Multiple filter rules may be defined for each table in the publication. If no filter rule is defined for a given table, then no filtering can be later enabled on that corresponding table in any master node of the multi-master replication system.

After filter rules have been defined for a publication table, you can later choose whether or not to enable those filter rules on any master node in the replication system in accordance with the following rules.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- At most one filter rule can be enabled on a given table in a given master node.
- The same filter rule may be enabled on the same given table in several, different master nodes.
- Different filter rules may be enabled on the same given table but in different master nodes.

If you want to define table filters on the publication tables, click the Table Filters tab. Select the table from the Table/View drop-down list for which you wish to add a filter rule. Click the Add Filter button.

In the Filter dialog box, enter a descriptive filter name and the filter clause to select the rows you want to replicate. The filter name and filter clause must meet the following conditions:

- For any given table, each filter rule must be assigned a unique filter name.
- For any given table, the filter clauses must have different syntaxes (that is, the filtering criteria must be different).

In the following example a filter rule is defined on the `dept` table so only rows where the `deptno` column contains `10`, `20`, or `30` are included in replications. All other rows are excluded from replication.

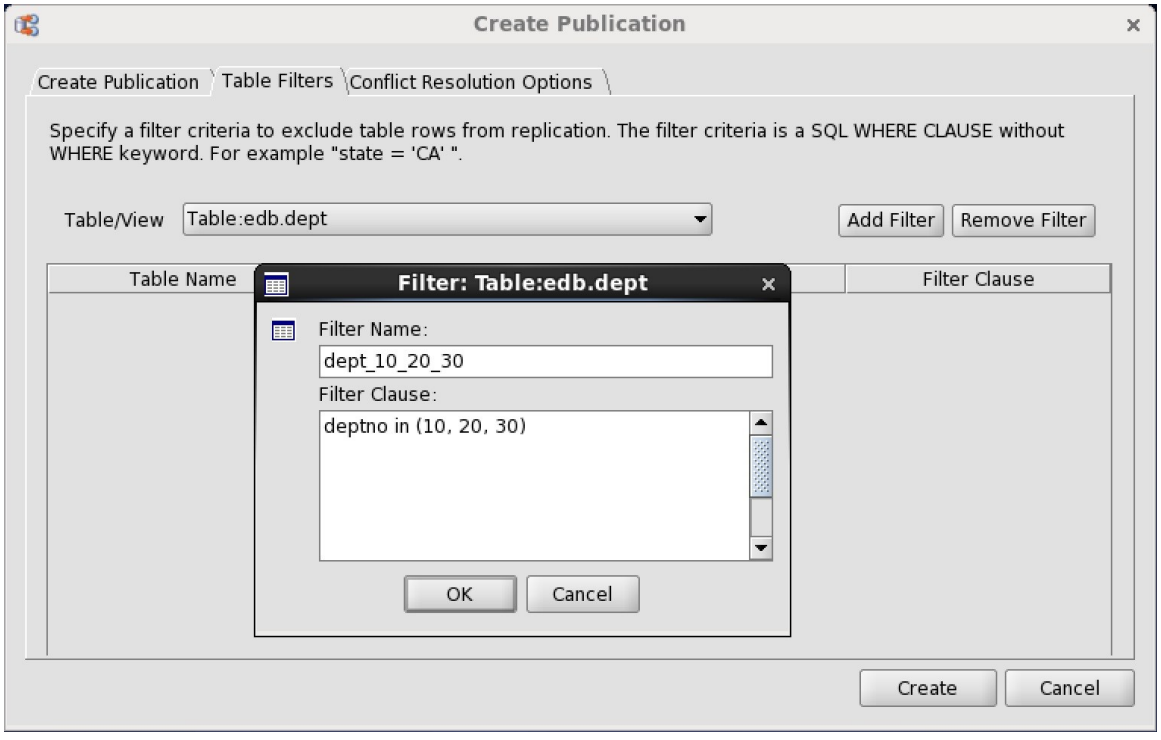


Figure 6-6 - Adding a filter rule for the dept table

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows a rule added to the `emp` table by choosing `edb.emp` from the Table/View drop-down list and then entering the selection criteria for only rows with `deptno` containing `10` in the Filter dialog box.

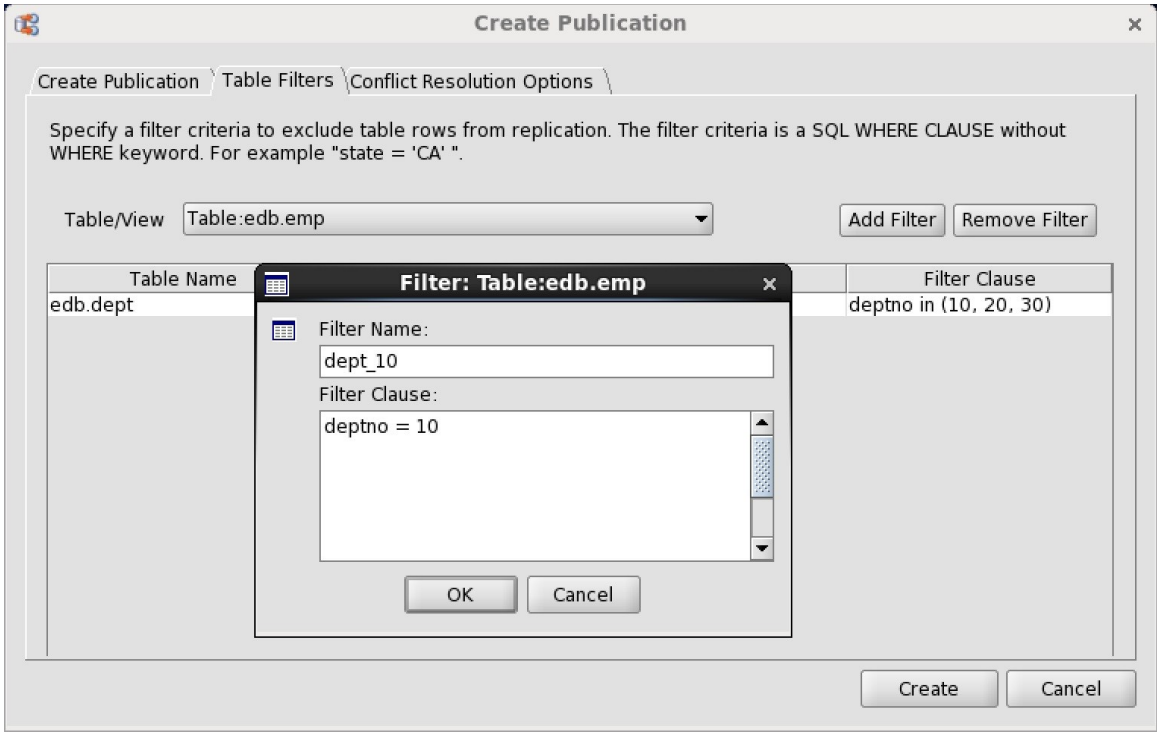


Figure 6-7 - Adding a filter rule for the emp table

Repeating this process, additional filter rules can be added for the `emp` table. The following shows the complete set of available filter rules defined for the `dept` and `emp` tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

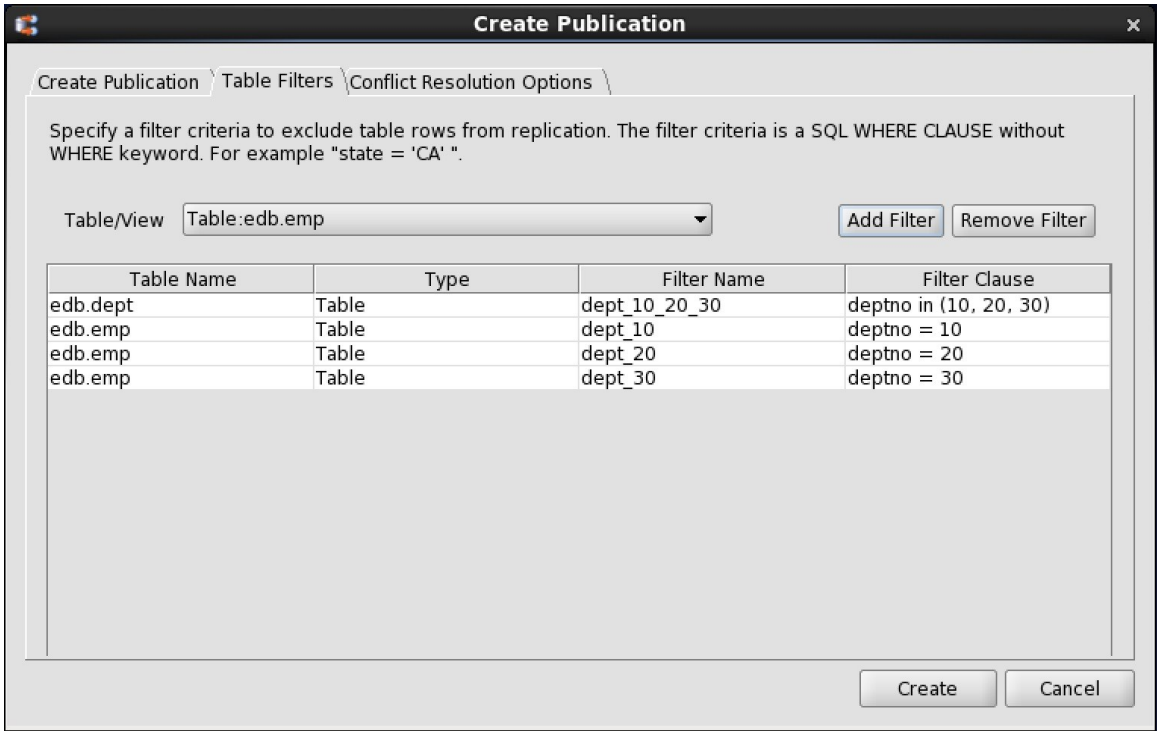


Figure 6-8 - Set of all available filter rules

To remove a filter rule, click the primary mouse button on the filter rule you wish to remove so the entry is highlighted and then click the Remove Filter button.

You may also modify the filter name or filter clause of a filter rule listed in the Table Filters tab by double-clicking on the cell of the filter name or filter clause you wish to change. When the cursor appears in the cell, enter the text for the desired change.

When creating additional master nodes, you may selectively enable these table filters on the corresponding tables in the additional master nodes. See Section 6.3 for information on creating additional master nodes.

Note: To enable table filters on the master definition node under which you are currently creating the publication, you must first switch the role of the master definition node to a different master node (see Section 6.10), and then follow the directions in Section 6.9 to enable the table filters.

This completes the process of defining table filters. The next step is changing conflict resolution options.

If you wish to change the conflict resolution options from their default settings, follow the directions in the next step, otherwise go on to Step 5.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4 (Optional): If you want to modify or see the current conflict resolution options, click the Conflict Resolution Options tab. For each table, you can select the primary conflict resolution strategy and a standby strategy by clicking the primary mouse button over the appropriate box to expose a drop-down list of choices.

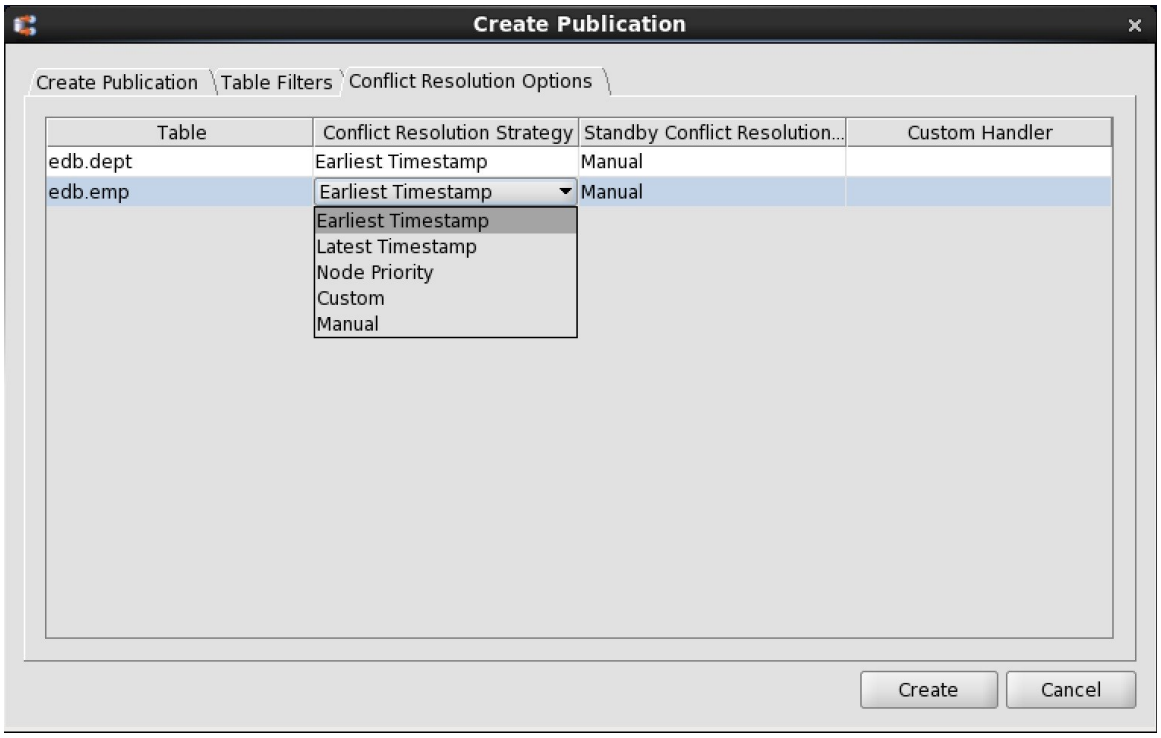


Figure 6-9 - Conflict Resolution Options tab

If during synchronization replication, conflicting changes are pending against the same row from different master nodes, the conflict resolution strategy determines which of the conflicting changes is accepted and replicated to all master nodes. The conflicting changes that are not accepted are discarded.

If the selection from the Conflict Resolution Strategy column does not resolve the conflict, the selection from the Standby Conflict Resolution Strategy column is applied. If neither strategy resolves the conflict, the event is marked as `Pending` in the Conflict History tab. See Section 6.7 for information on viewing conflict history.

An example of a conflict is when the same column of the same row is changed by transactions in two different master nodes. Depending upon the conflict resolution strategy in effect for the table, one of the transactions is accepted and replicated to all master nodes. The other transaction is discarded and not replicated to any master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following is a brief summary of each conflict resolution strategy:

- **Earliest Timestamp.** The conflicting change with the earliest timestamp is accepted and replicated to all other master nodes. All other conflicting changes are discarded.
- **Latest Timestamp.** The conflicting change with the latest timestamp is accepted and replicated to all other master nodes. All other conflicting changes are discarded.
- **Node Priority.** The conflicting change occurring on the master node with the highest priority level is accepted and replicated to all other master nodes. All other conflicting changes are discarded.
- **Custom.** Update/update conflicts are resolved with a PL/pgSQL custom conflict handling program.
- **Manual.** The conflict remains unresolved. Conflicting changes remain applied in each master node where they originated, but are not replicated to other master nodes. The proper adjustments must be manually applied in each master node.

See Section 6.6.4 for more information on conflict resolution strategies.

Step 5: If you expect update/update conflicts, then set the `REPLICA IDENTITY` option to `FULL` on those tables where the conflicts are expected to occur. See Section 6.6.1 for additional information.

Step 6: Click the Create button. If Publication Created Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

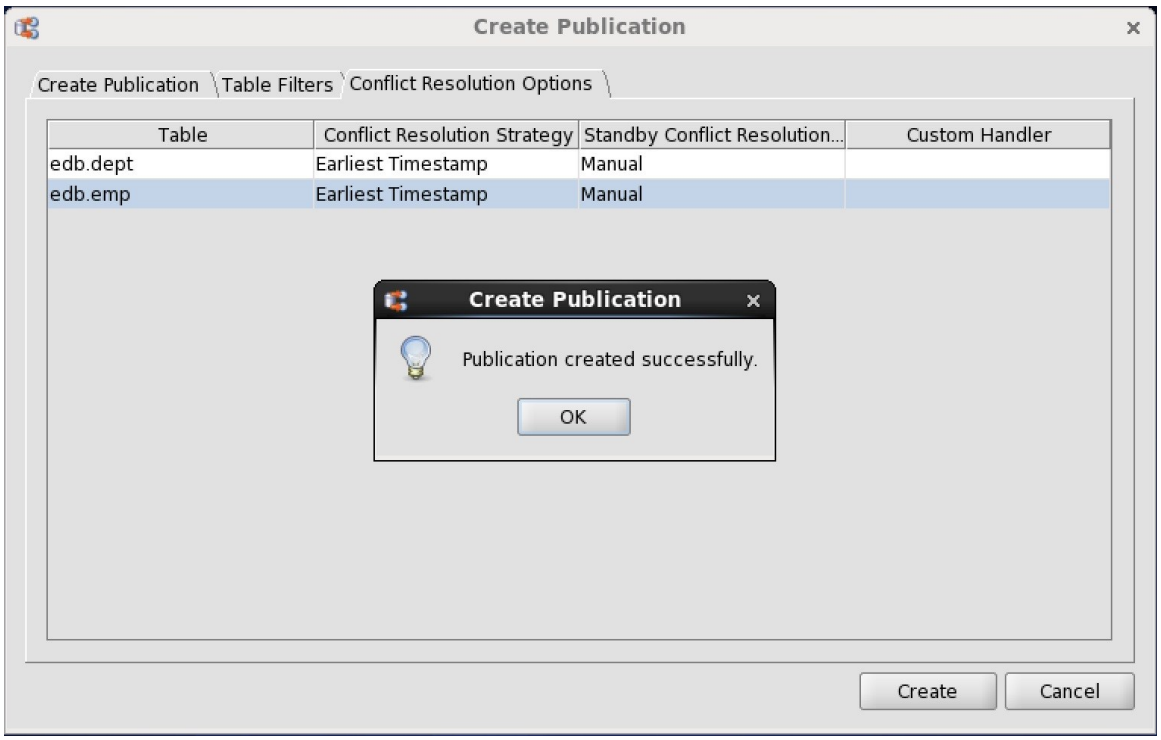


Figure 6-10 - Publication created successfully

Upon successful publication creation, a Publication node is added to the replication tree.

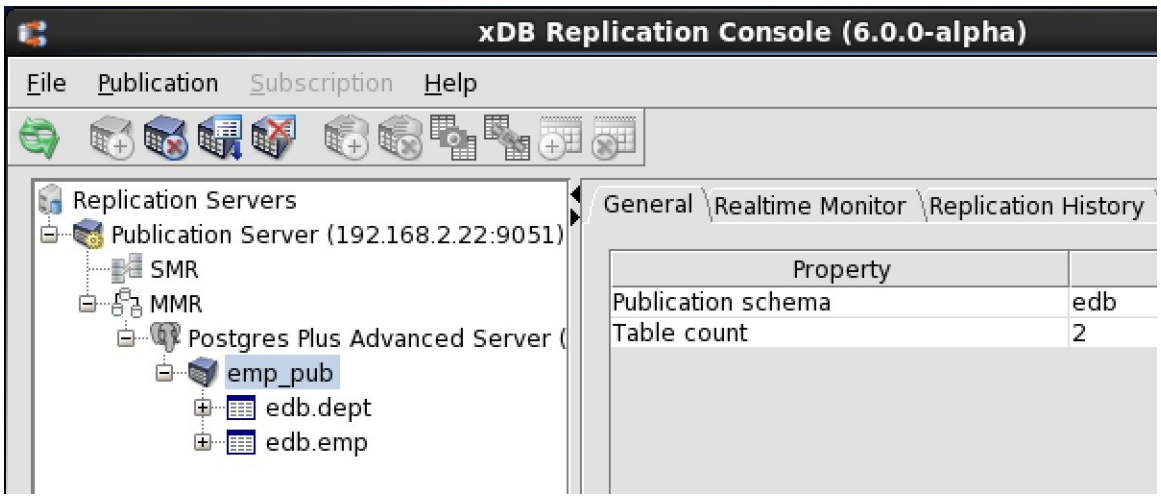


Figure 6-11 - Replication tree after adding a publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.3 Creating Additional Master Nodes

Once you have created the master definition node, you add additional databases to the multi-master replication system by defining additional master nodes.

This is done by creating additional publication database definitions subordinate to the MMR type node under the Publication Server node that contains the master definition node.

After the publication database definition is created, a Publication Database node representing the master node appears in the replication tree of the xDB Replication Console. The publication that was defined under the master definition node appears under the Publication Database node.

You must enter database connection information such as the database server network address, database identifier, and database login user name and password when you create the publication database definition. The connection information is used by the publication server to access the publication tables when it performs replication.

Step 1: Make sure the database server for the master definition node is running and accepting client connections.

Step 2: Select the MMR type node under the same Publication Server node that contains the master definition node. From the Publication menu, choose Publication Database, and then choose Add Database. Alternatively, click the secondary mouse button on the MMR type node and choose Add Database. The Publication Service – Add Database dialog box appears.

Step 3: Fill in the following fields:

- **Database Type.** Select PostgreSQL or Postgres Plus Advanced Server for the master node. For an Advanced Server Oracle compatible installation, select the Postgres Plus Advanced Server option. For PostgreSQL or an Advanced Server PostgreSQL compatible installation, select the PostgreSQL option.
- **Host.** IP address of the host on which the master node is running.
- **Port.** Port on which the master node is listening for connections.
- **User.** The database user name for the master node created in Step 1 of Section 6.1.4.
- **Password.** Password of the database user.
- **Database.** Enter the database name of the master node.
- **URL Options (For SSL connectivity).** Enter the URL options to establish SSL connectivity to the master node. See Section 7.11 for information on using SSL connections.
- **Changeset Logging (For Postgres).** This setting is predetermined by the selection on the master definition node (see Section 6.2.2). Table Triggers is for the trigger-based method of synchronization replication. WAL Stream is for the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

log-based method of synchronization replication. See Section 2.2.9 for information on the trigger-based method. See Section 2.2.10 for information on the log-based method.

- **Node Priority Level.** An integer from 1 to 10, which is the priority level assigned to this master node for conflict resolution based on node priority. The highest priority is 1 while the lowest is 10. See Section 6.6.4 for information on conflict resolution strategies. As each additional master node is added, the default priority level number increases assigning a lower priority level to each additional node.
- **Replicate Publication Schema.** Check this box if you want the publication server to create the publication table definitions in the new master node by copying the definitions from the master definition node. If you do not check this box, it is assumed that you have already created the table definitions in the master node. If you are using the offline snapshot technique to create this master node, do not check this box. See Section 7.9 for information on using an offline snapshot.
- **Perform Initial Snapshot.** Check this box if you want the publication server to perform a snapshot from the master definition node to this master node when you click the Save button. If you do not check this box, the tables on the master node will not be loaded until you perform a replication at some later time. If you are using the offline snapshot technique to create this master node, you should have already loaded the table rows. Therefore do not check this box unless you want to reload the data. See Section 7.9 for information on using an offline snapshot.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

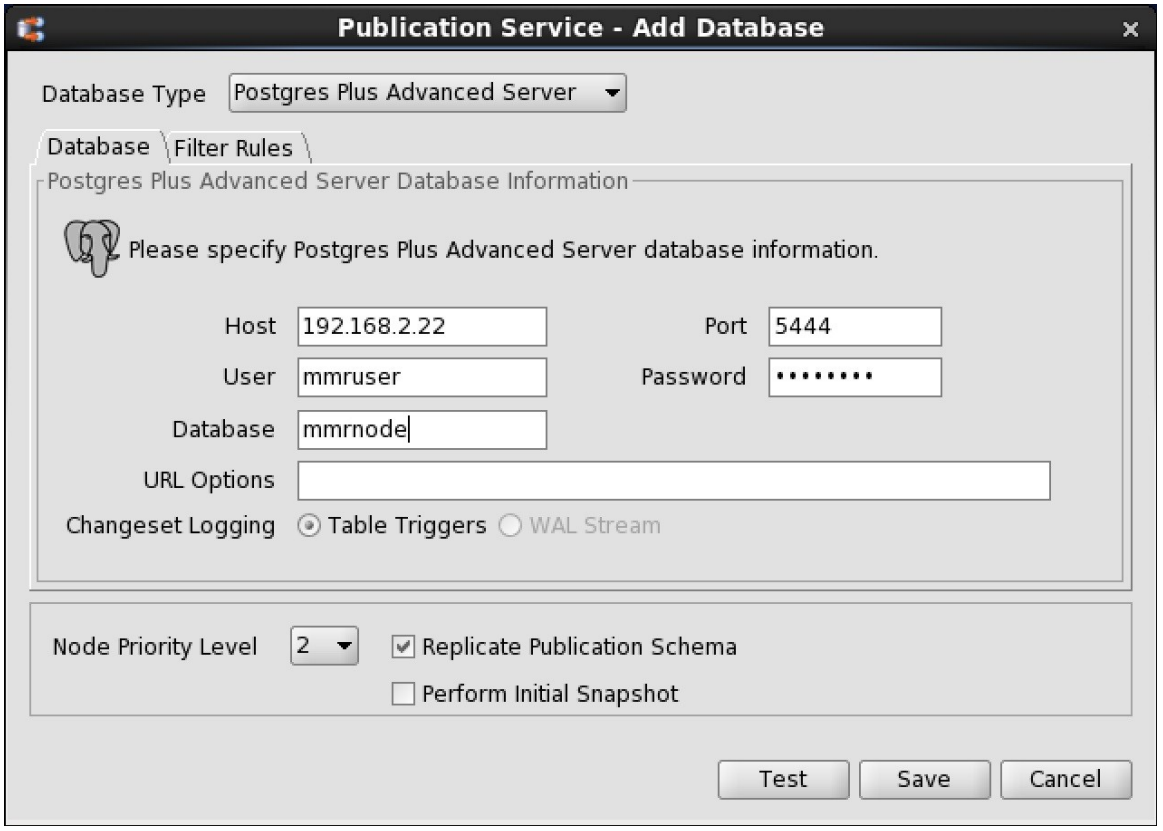


Figure 6-12 - Publication Service - Add Database dialog box for an additional master node

Note: Unless you intend to use the offline snapshot technique (see Section 7.9), it is suggested that you check the Perform Initial Snapshot box. An initial snapshot replication must be performed from the master definition node to every other master node before performing synchronization replications on demand (see Section 6.5.2) or by a schedule (see Section 7.2). If a newly added master node did not undergo an initial snapshot, any subsequent synchronization replication may fail to apply the transactions to that master node. The initial snapshot can also be taken by performing an on demand snapshot (see Section 6.5.1).

Step 4: Click the Test button. If Test Result: Success appears, click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

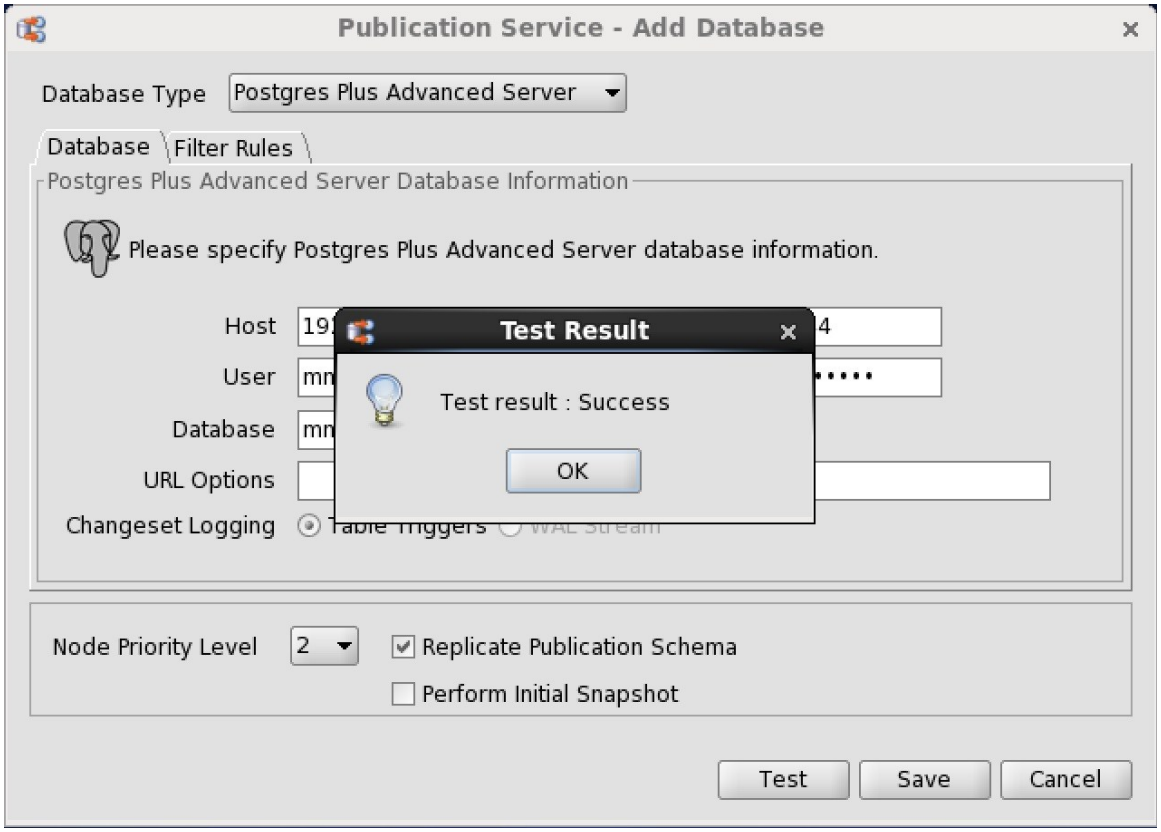


Figure 6-13 - Successful master node test

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 4.

Step 5 (Optional): If you defined a set of available table filters for the publication, you have the option of enabling these filters on this master node. See Section 6.2.3 for instructions on defining table filters. If you do not wish to filter the rows that are replicated to this master node, go to Step 6.

Note: See Section 2.2.12.3 for table setup requirements for a log-based replication system as well as general restrictions on the use of table filters.

Click the Filter Rules tab to apply one or more filter rules to the master node. At most one filter rule may be enabled on any given table in the master node.

In the following example the filter named `dept_10_20_30` is enabled on the `dept` table and the filter named `dept_30` is enabled on the `emp` table of this master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

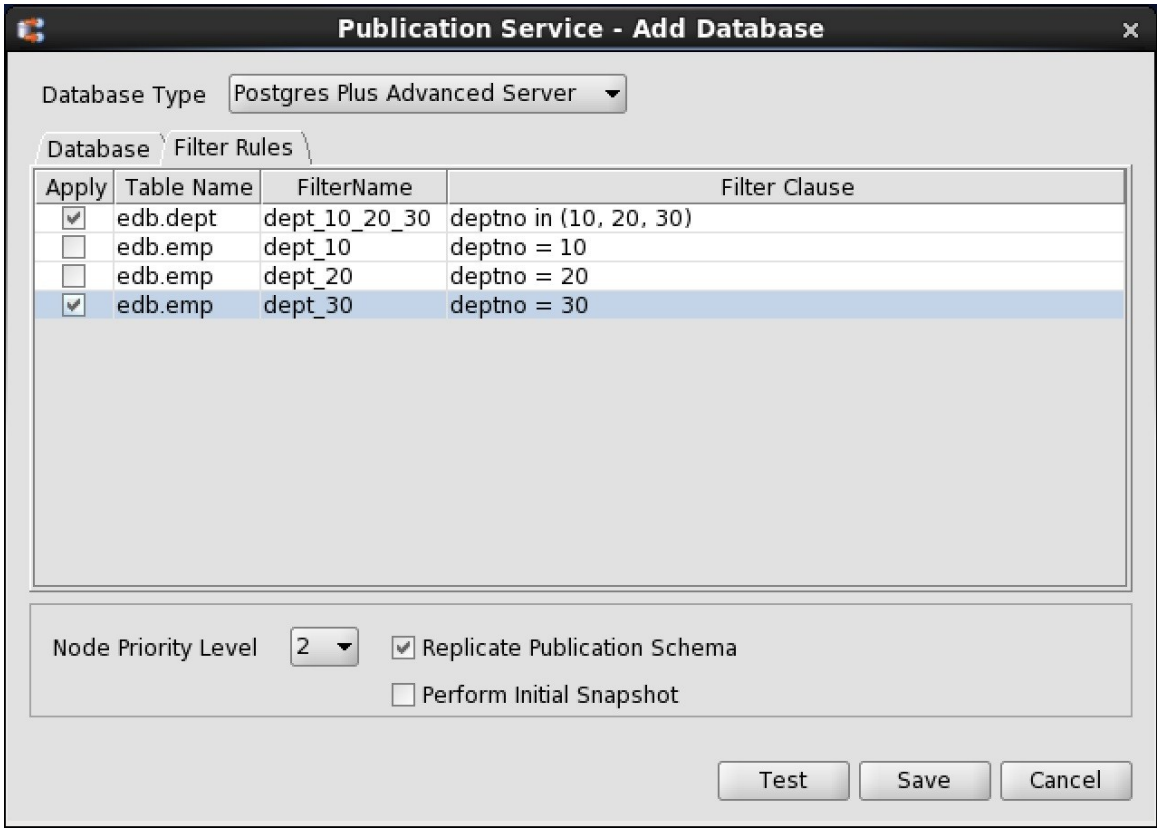


Figure 6-14 - Enabling filter rules on a master node

Step 6: Check the Perform Initial Snapshot box if you want the publication server to perform a snapshot from the master definition node to this master node when you click the Save button. If you do not check this box, the tables on the master node will not be loaded until you perform a replication at some later time.

If you are using the offline snapshot technique to create this master node, you should have already loaded the table rows. Therefore do not check this box unless you want to reload the data. See Section 7.9 for information on using an offline snapshot.

If you do check the Perform Initial Snapshot check box, the Verbose Output check box appears as shown by the following:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

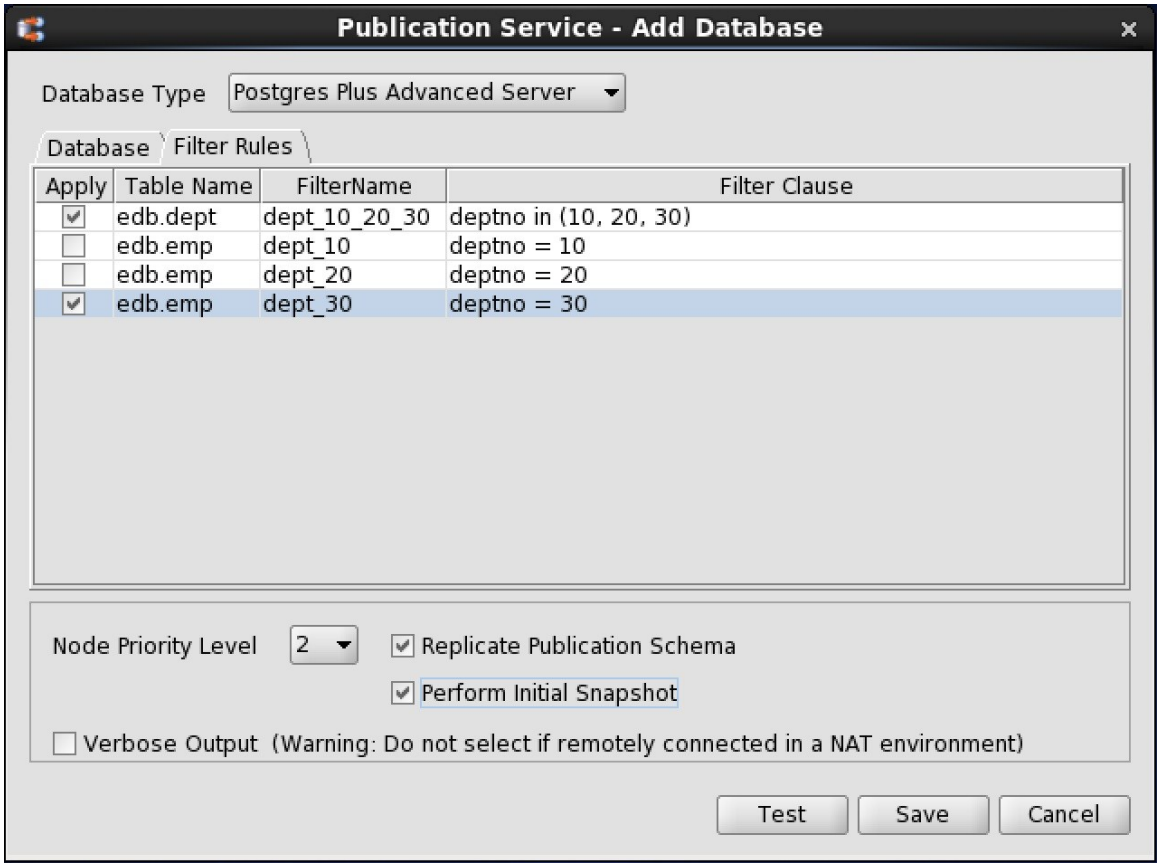


Figure 6-15 - Initial snapshot with verbose output option

If you skipped the enabling of table filters as described in Step 5, and you checked the Perform Initial Snapshot check box after Step 4, the Verbose Output check box is displayed as well:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

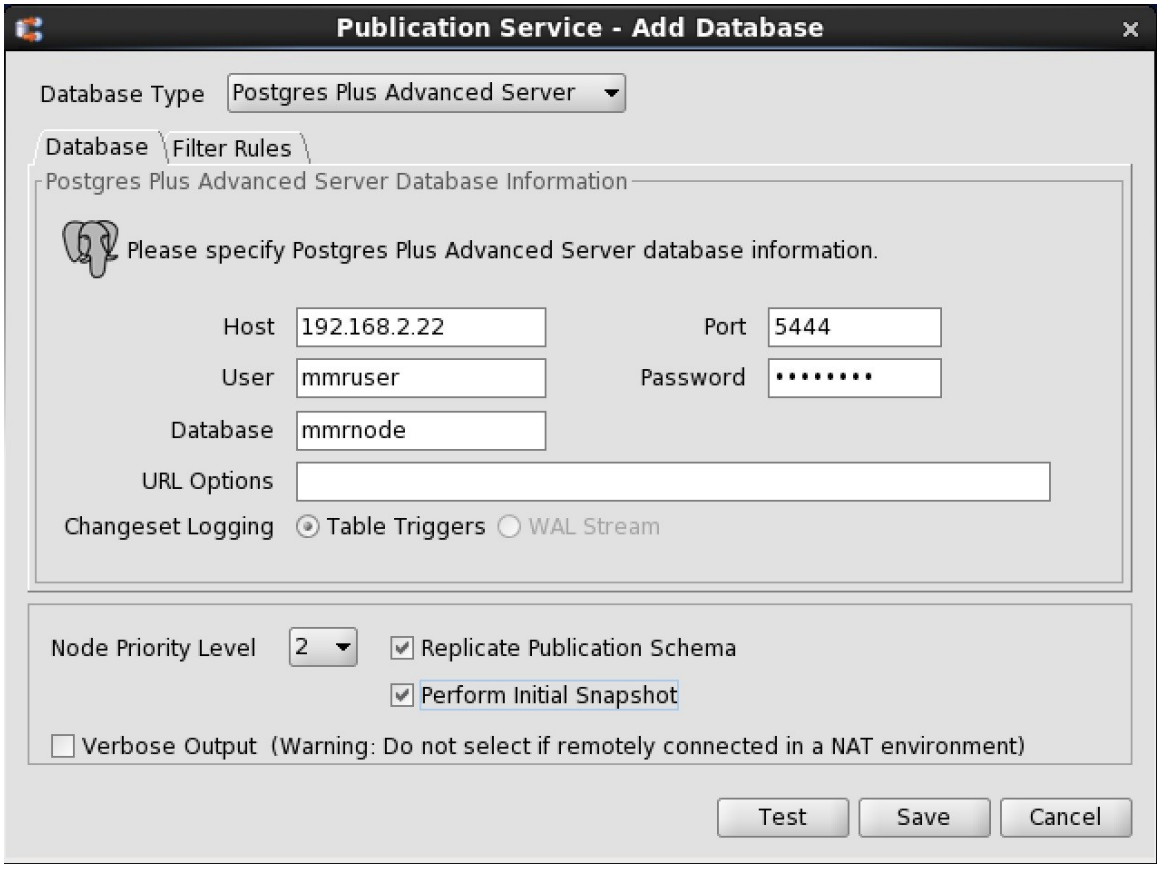


Figure 6-16 - Initial snapshot with verbose output option

Select the Verbose Output check box only if you want to display the output from the snapshot in the dialog box. This option should be left unchecked in a network address translation (NAT) environment as a large amount of output from the snapshot may delay the response from the Snapshot dialog box.

Click the Save button.

When the publication database definition is successfully saved, a Publication Database node is added to the replication tree under the MMR type node of the Publication Server node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

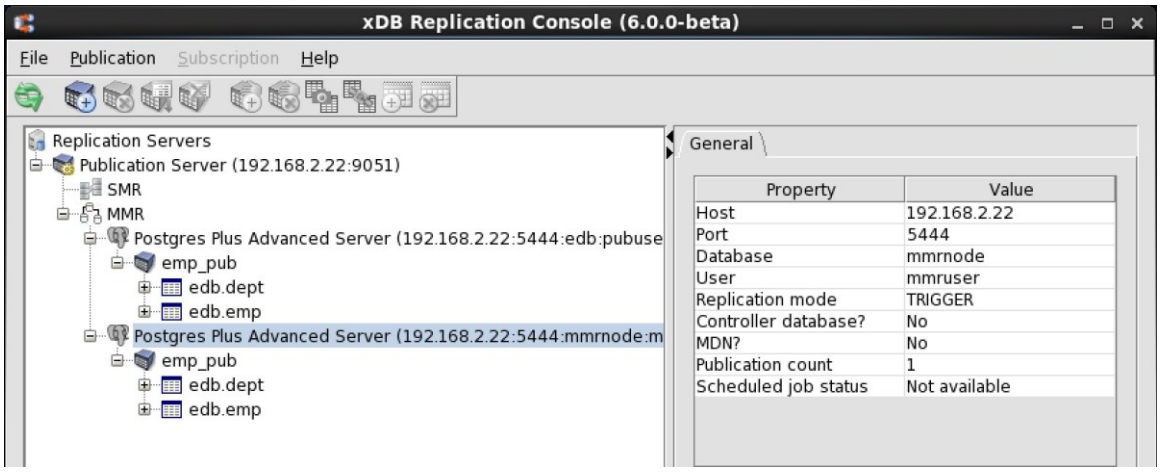


Figure 6-17 - Replication tree after adding an additional master node

Unlike the master definition node, the label MDN does not appear at the end of the node in the replication tree. The MDN field is set to NO in the Property window to indicate this is not the master definition node.

In addition, a Publication node appears under the newly added master node. This Publication node represents the publication in the master definition node, which is replicated to the master node.

If in Step 6, you checked the Perform Initial Snapshot check box, an initial snapshot replication is performed. If you checked the Verbose Output check box, the log of the snapshot is displayed as well.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

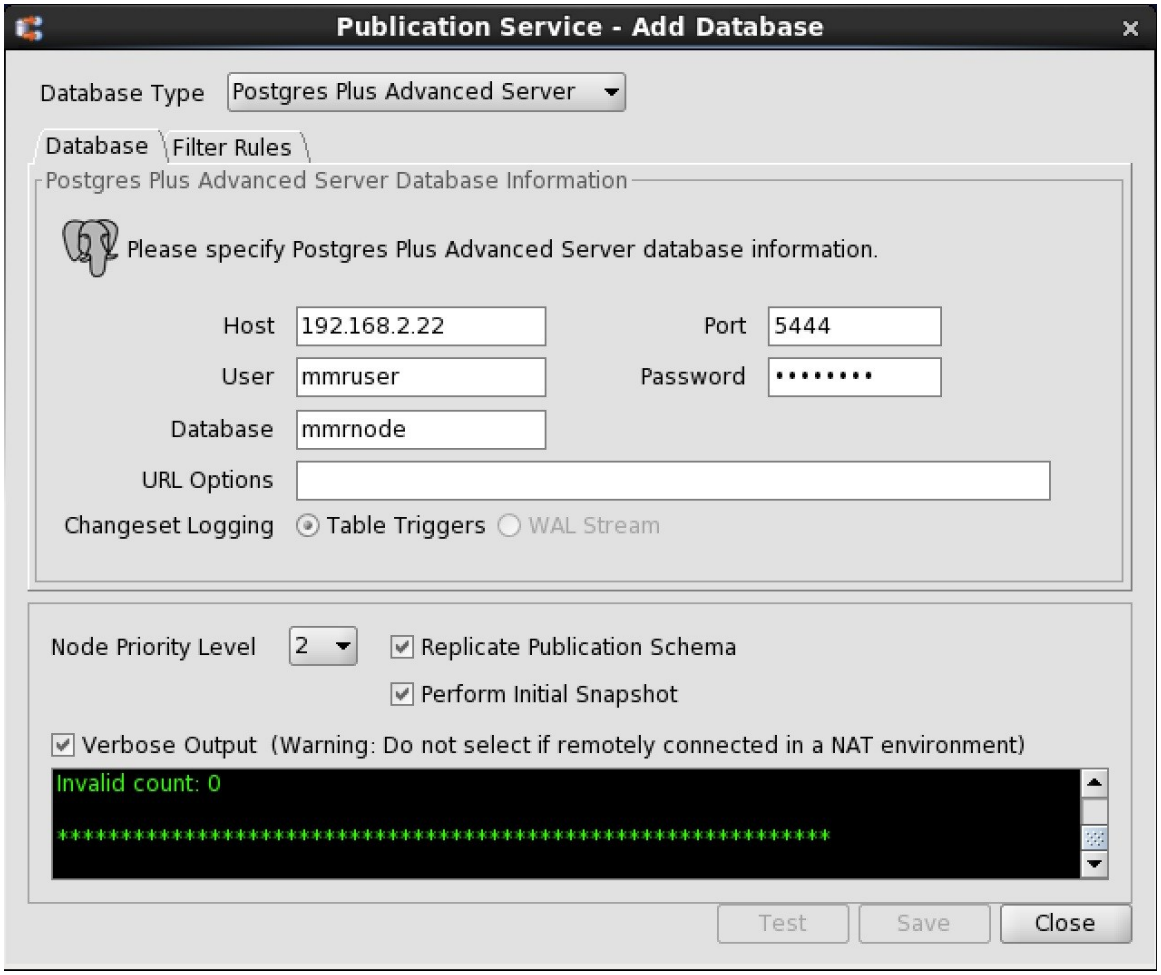


Figure 6-18 - Adding a master node with an initial snapshot

If the snapshot is successful, the replicated tables in the master node are loaded with the rows from the publication tables of the master definition node.

Step 7: If you expect update/update conflicts, then set the `REPLICA IDENTITY` option to `FULL` on those tables where the conflicts are expected to occur. See Section 6.6.1 for additional information.

Step 8 (Optional): If users are to access the data in the publication tables residing on this master node, it is convenient to have one or more "group" roles containing the required privileges to access these tables. For the trigger-based method, privileges must also be granted on the control schema objects to users who are to perform inserts, updates, or deletions on the publication tables. When using the log-based method a user needs access to the publication tables and to certain control schema objects as well under certain circumstances.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When adding new users, granting these users membership in these roles gives them the privileges to access the publication tables. This eliminates the need to grant these privileges individually to each new user.

After you perform the replication of the publication schema as shown in Step 3, you can grant the required privileges needed to access the publication tables and its control schema objects. See Step 2 of Section 5.1.4.3 for information on how this can be accomplished.

6.4 Control Schema Objects Created in Master Nodes

Creation of master nodes results in the creation of control schema objects in each master node database.

See Section 5.2.4 for the control schema objects created in each master node.

Do not delete any of these control schema objects as the replication system metadata will become corrupted.

When you remove a master node using the xDB Replication Console or xDB Replication Server CLI, all of its control schema objects are deleted from that master node database.

6.5 On Demand Replication

After a master definition node, its publication, and additional master nodes are created, there are a couple of choices for starting the replication process.

- Replication can be done immediately by performing an initial on demand snapshot after which synchronization replication may be performed.
- Replication can be scheduled to start at a later date and time by creating a schedule.

This section discusses the procedure for initiating a replication on demand. Section 7.2 discusses how to create a schedule.

6.5.1 Performing Snapshot Replication

A snapshot replication occurs from the master definition node to a selected master node.

When you create a master node for the first time, you have the option of performing an initial snapshot (see Step 3 of Section 6.3).

You can perform snapshots to this master node at any later point in time according to the following steps.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 1: Select the Publication node under the master node for which you wish to perform snapshot replication.

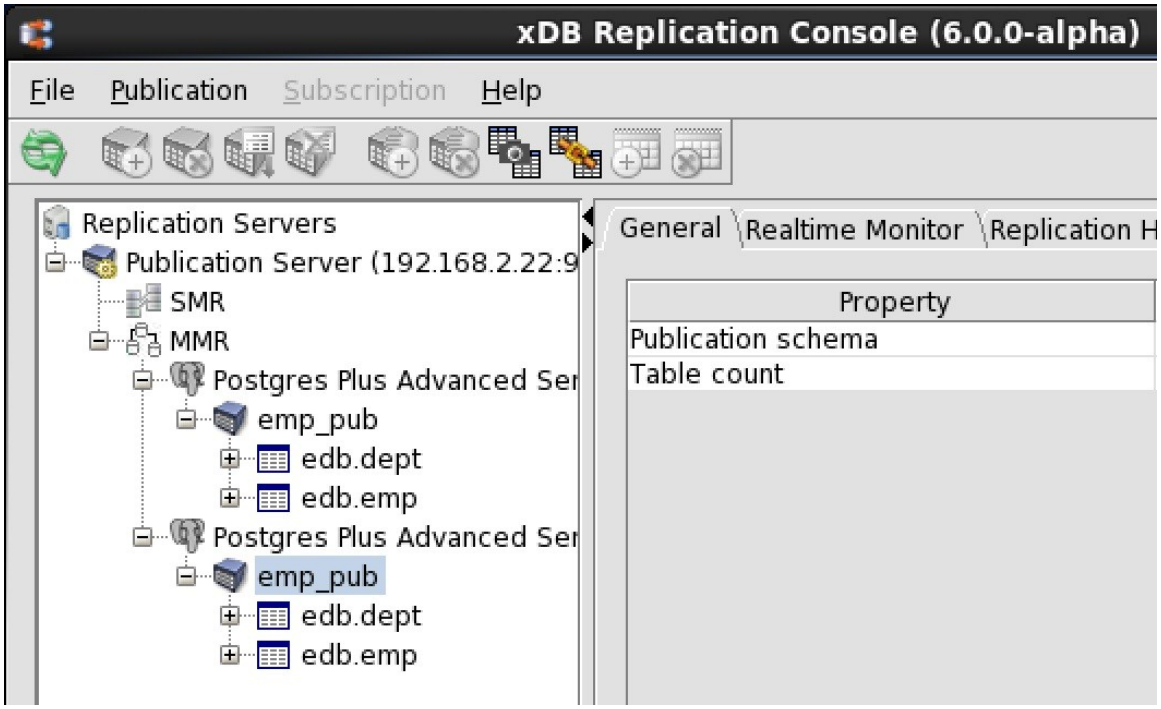


Figure 6-19 - Selecting a master node publication for an on demand snapshot

Step 2: Open the Snapshot dialog box in any of the following ways:

- Click the secondary mouse button on the Publication node and choose Snapshot.
- Click the primary mouse button on the Snapshot icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

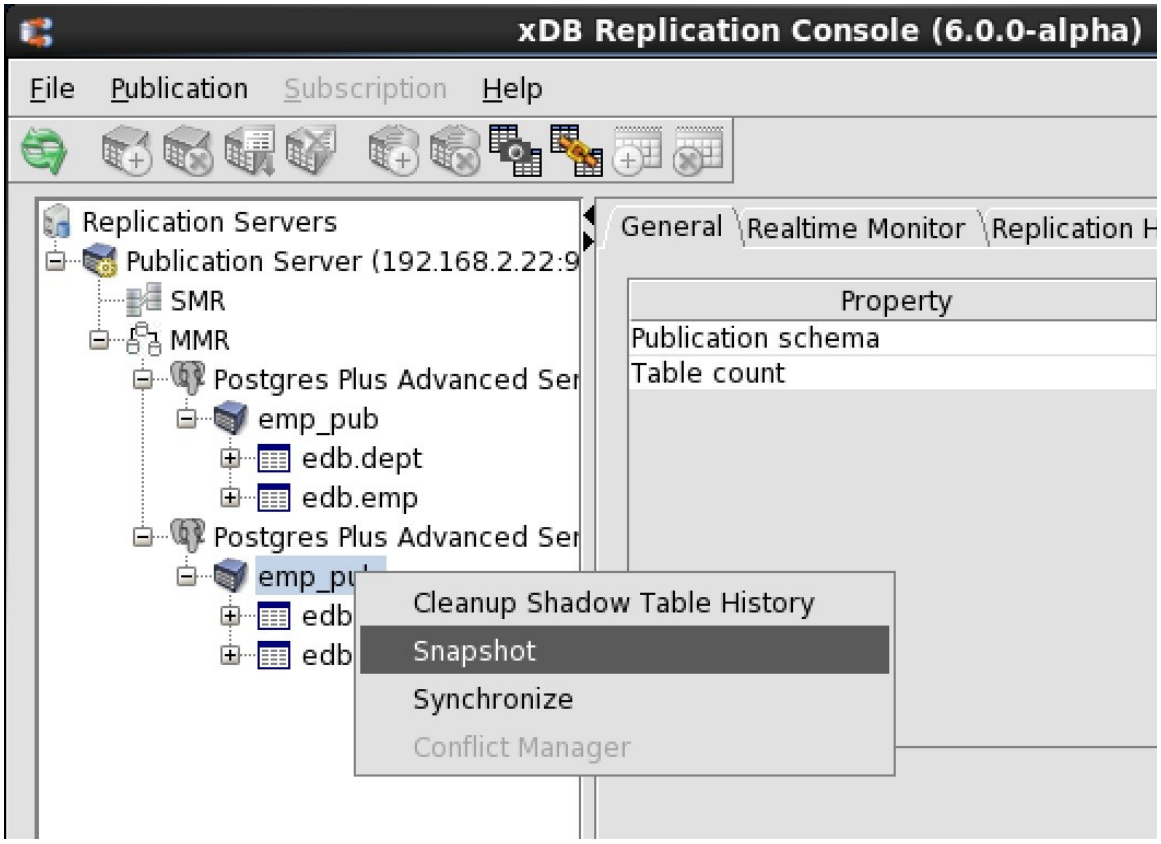


Figure 6-20 - Opening the Snapshot dialog box

Step 3: Select the Verbose Output check box only if you want to display the output from the snapshot in the dialog box. This option should be left unchecked in a network address translation (NAT) environment as a large amount of output from the snapshot may delay the response from the Snapshot dialog box. Click the Snapshot button to start snapshot replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Figure 6-21 - Snapshot dialog box

Step 4: Snapshot Taken Successfully appears if the snapshot was successful. Click the OK button. If the snapshot was not successful, scroll through the messages in the Snapshot dialog box window if Verbose Output was selected or check the log files.

The status messages of each snapshot are saved in the *Migration Toolkit log* files named `mtk.log[.n]` (where `[.n]` is an optional history file count if log file rotation is enabled) in the following directories:

For Linux:

```
/var/log/xdb-x.x
```

For Windows:

```
POSTGRES_HOME\.enterprisedb\xdb\x.x
```

POSTGRES_HOME is the home directory of the Windows `postgres` account (`enterprisedb` account for Advanced Server installed in Oracle compatible configuration mode). The specific location of *POSTGRES_HOME* is dependent upon your version of Windows. The xDB Replication Server version number is represented by *x.x*.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

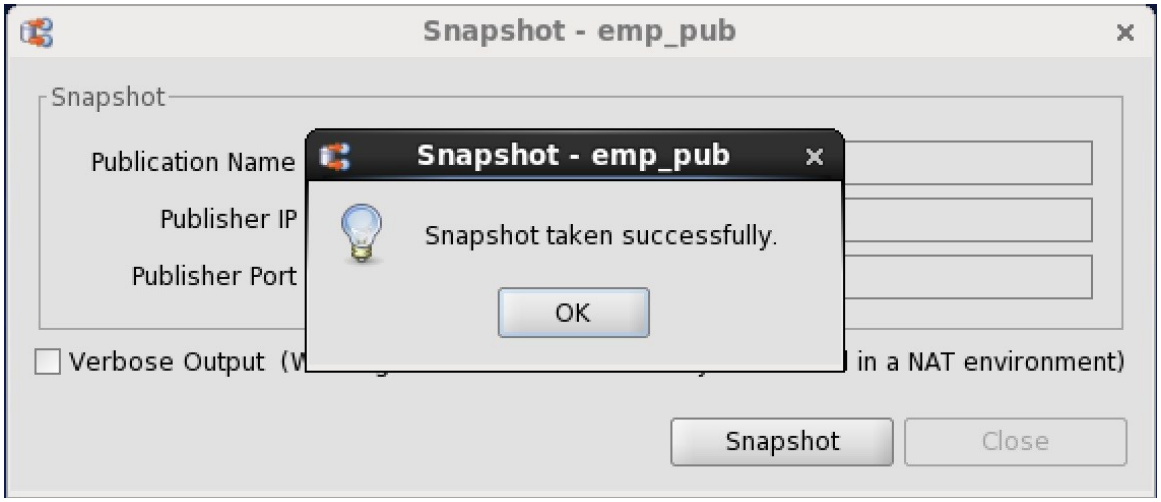


Figure 6-22 - Successful on demand snapshot

The publication has now been replicated from the master definition node to the selected master node. A record of the snapshot is maintained in the replication history. See Section 7.4 for information on how to view replication history.

6.5.2 Performing Synchronization Replication

Note: Be sure an initial snapshot replication has been performed from the master definition node to every other master node in the multi-master replication system. If a newly added master node did not undergo an initial snapshot, any subsequent synchronization replication may fail to apply the transactions to that master node. The initial snapshot could be taken when the master node is first added (see Section 6.3) or by performing an on demand snapshot (see Section 6.5.1).

When synchronization replication is performed in a multi-master replication system, a series of synchronization operations occur between every master node pair in the replication system.

For example, if a replication system consists of master nodes A, B, and C, synchronization is applied to the following node pairs whenever synchronization replication is initiated:

- Changes on node A are applied to node B.
- Changes on node A are applied to node C.
- Changes on node B are applied to node A.
- Changes on node B are applied to node C.
- Changes on node C are applied to node A.
- Changes on node C are applied to node B.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

There may be circumstances where changes made on different nodes result in conflicts. Section 6.6 discusses the types of conflicts that may occur and how they can be resolved.

The following steps describe how to initiate on demand synchronization replication.

Step 1: Select the Publication node under any master node. Regardless of the master node chosen, synchronization is applied to every master node pair in the replication system.

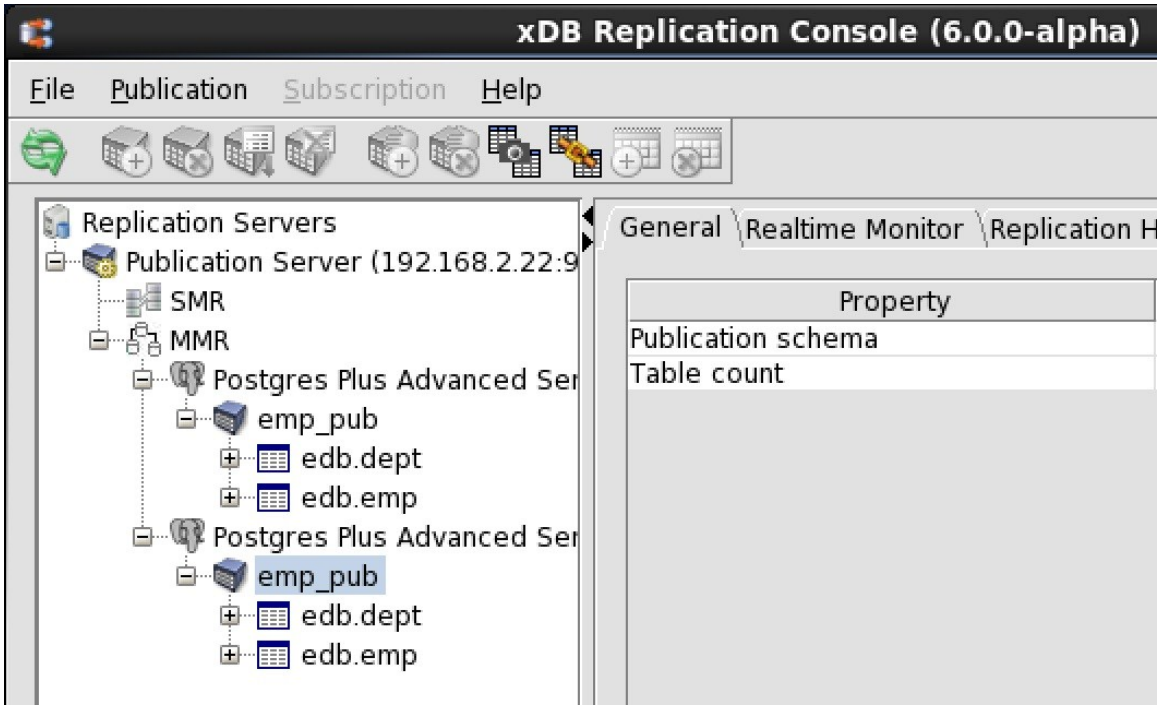


Figure 6-23 - Selecting a master node publication for an on demand synchronization

Step 2: Open the Synchronize dialog box in any of the following ways:

- Click the secondary mouse button on the Publication node and choose Synchronize.
- Click the primary mouse button on the Synchronize icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

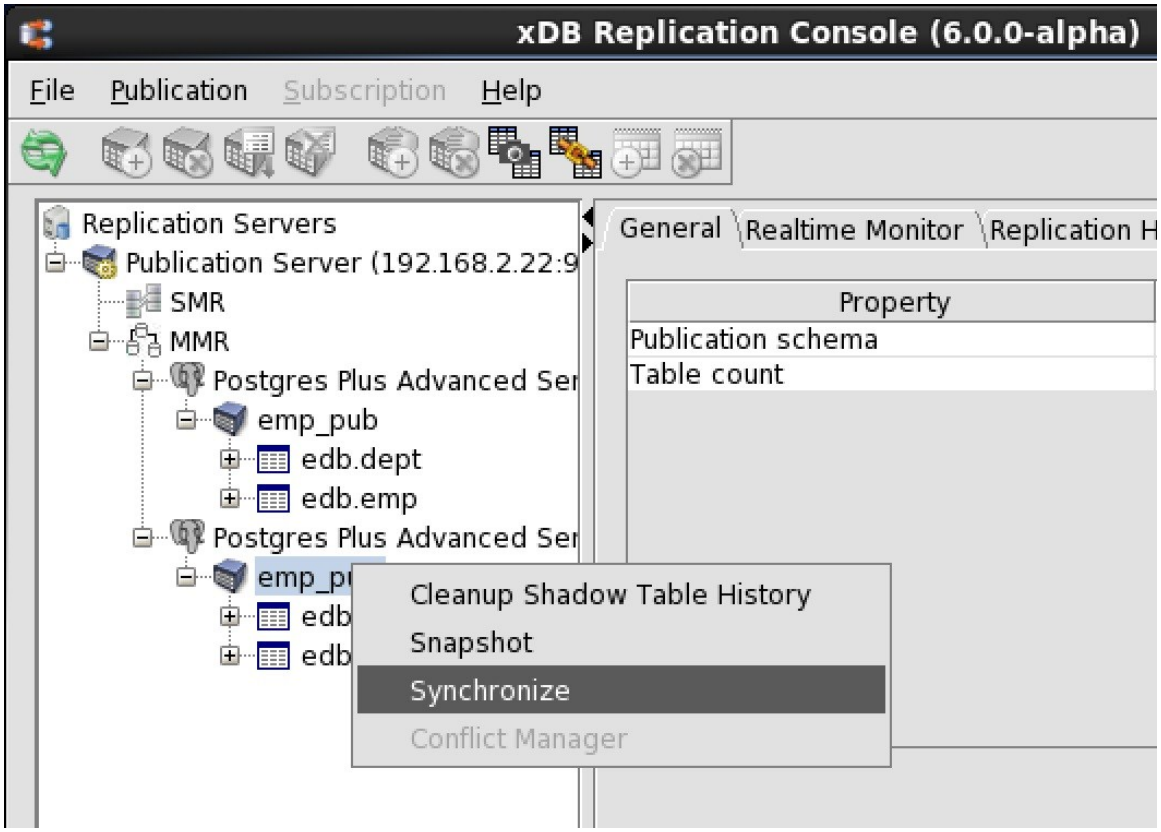


Figure 6-24 - Opening the Synchronize dialog box

Step 3: Click the Synchronize button to start synchronization replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

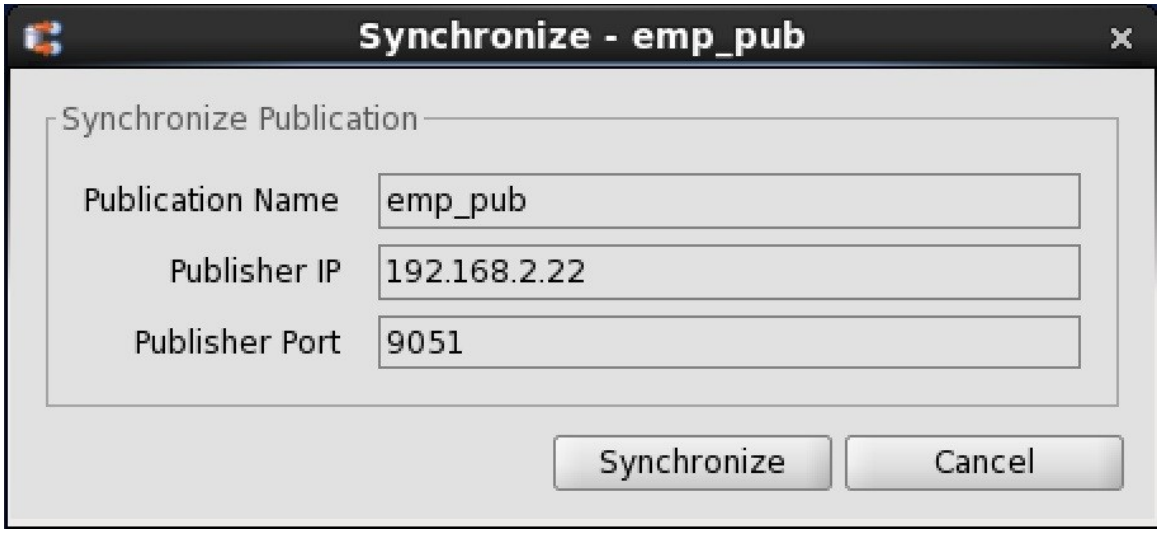


Figure 6-25 - Synchronize dialog box

Step 4: Publication Synchronized Successfully appears if the synchronization was successful. Click the OK button. If the synchronization was not successful, an error message is displayed.

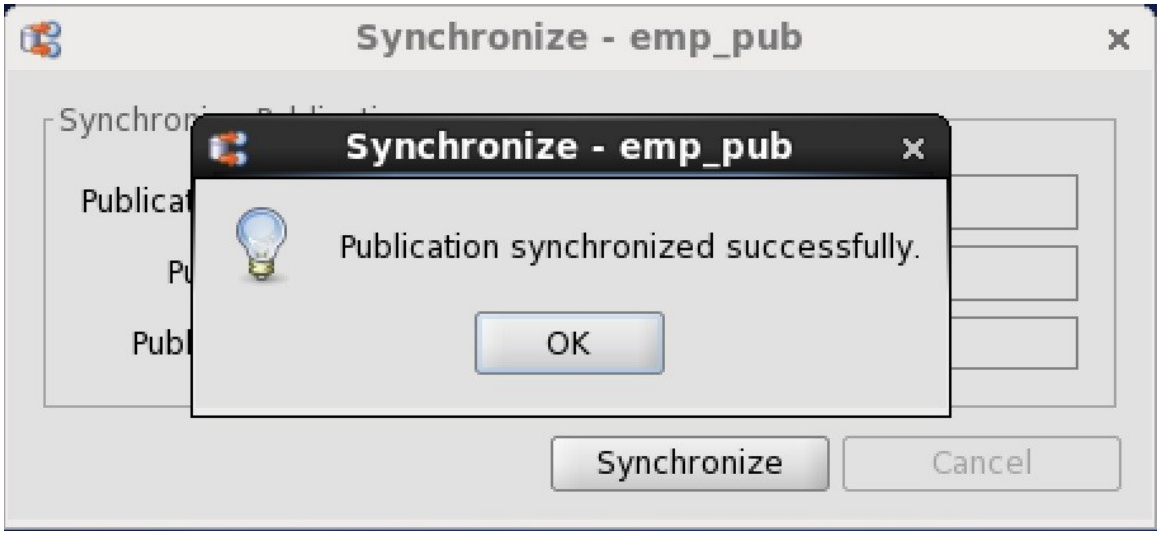


Figure 6-26 - Successful on demand synchronization

The operations that were applied to the publication tables can be seen in the replication history. See Section 7.4 for information on how to view replication history.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Conflicting changes that were encountered can be seen in the conflict history. See Section 6.7 for information on how to view conflict history.

6.6 Conflict Resolution

There are certain situations where synchronization replication may result in data conflicts arising from the row changes that took place on different master nodes.

Conflict resolution deals with the topic of the types of conflicts that might occur, the strategies for dealing with conflicts, and the options available for automatically resolving such conflicts.

6.6.1 Configuration Parameter and Table Setting Requirements

Depending upon the multi-master replication system environment, certain configuration settings may be required in order for the conflict resolution process to operate properly.

The following are required only for the log-based method. These do not apply to the trigger-based method.

- **track_commit_timestamp.** Any Postgres 9.5 database server containing a master node must have its `track_commit_timestamp` configuration parameter enabled. The `track_commit_timestamp` parameter is located in the `postgresql.conf` file. If `track_commit_timestamp` is not enabled, then update/update conflicts are not automatically resolved such as by using the earliest timestamp of the conflicting transactions. As a result, these conflicting transactions are left in a pending state. See Section 6.6.7 for an example of how update/update conflicts are automatically resolved.
- **REPLICA IDENTITY FULL.** If update/update conflicts are expected to occur on a given publication table, then the `REPLICA IDENTITY` setting for the table must be set to `FULL` on every master node. The case where update transactions occur on separate master nodes, but updating different columns in the same row, is not considered an update/update conflict. However, if `REPLICA IDENTITY` is not set to `FULL`, then this case will be recorded as an update/update conflict.

The `REPLICA IDENTITY` option is set to `FULL` using the `ALTER TABLE` command as shown by the following:

```
ALTER TABLE schema.table_name REPLICA IDENTITY FULL
```

The following is an example of the `ALTER TABLE` command:

```
ALTER TABLE edb.dept REPLICA IDENTITY FULL;
```

The `REPLICA IDENTITY` setting can be displayed by the PSQL utility using the `\d+` command:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
edb=# \d+ edb.dept
                                  Table "edb.dept"
 Column |         Type          | Modifiers | Storage  | Stats target | Description
--------+-----------------------+-----------+----------+--------------+-------------
 deptno | numeric(2,0)          | not null  | main     |              |
 dname  | character varying(14) |           | extended |              |
 loc    | character varying(13) |           | extended |              |
Indexes:
    "dept_pk" PRIMARY KEY, btree (deptno)
    "dept_dname_uq" UNIQUE CONSTRAINT, btree (dname)
Referenced by:
    TABLE "emp" CONSTRAINT "emp_ref_dept_fk" FOREIGN KEY (deptno) REFERENCES
dept(deptno)
    TABLE "jobhist" CONSTRAINT "jobhist_ref_dept_fk" FOREIGN KEY (deptno) REFERENCES
dept(deptno) ON DELETE SET NULL
Replica Identity: FULL
```

Note: In addition to conflict resolution requirements, the `REPLICA IDENTITY FULL` setting may be required on publication tables for other reasons in xDB Replication Server. See Section 2.2.12.3 for additional requirements.

6.6.2 Conflict Types

The types of conflicts can be summarized as follows:

- **Uniqueness Conflict.** A uniqueness conflict occurs when the same value is used for a primary key or unique column in an insert transaction on two or more master nodes. This is also referred to as an *insert/insert* conflict.
- **Update Conflict.** An update transaction modifies a column value in the same row on two or more master nodes. For example, an employee address column is updated on master node A, and another user updates the address column for the same employee on master node B. The timestamps of when the transactions occur on each node could be different, but both transactions occur in a time interval during which synchronization has not yet occurred. Thus when synchronization does take place, both conflicting transactions are to be applied. This is also referred to as an *update/update* conflict.
- **Delete Conflict.** The row corresponding to an update transaction on the source node is not found on the target node as the row has already been deleted on the target node. This is referred to as an *update/delete* conflict. Conversely, if there is a delete transaction on the source node and an update transaction for the same row on the target node, this case is referred to as a *delete/update* conflict. Finally, in the case where the row corresponding to a delete transaction on the source node is not found on the target node as the row has already been deleted on the target node is referred to as a *delete/delete* conflict.

The following table definition is used to illustrate conflict resolution examples:

```
CREATE TABLE addrbook (
    id              SERIAL PRIMARY KEY,
    name            VARCHAR(20),
    address         VARCHAR(50)
);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following table illustrates an example of a uniqueness conflict.

Table 6-1 - Uniqueness Conflict

Timestamp	Action	Master Node A	Master Node B
t1	**Node A:** `INSERT INTO addrbook (name, address) VALUES ('A', 'ADDR A');` **Node A:** `INSERT INTO addrbook (name, address) VALUES ('B', 'ADDR B');`	`id = 1, name = 'A', address = 'ADDR A'` `id = 2, name = 'B', address = 'ADDR B'`	
t2	**Node B:** `INSERT INTO addrbook (name, address) VALUES ('C', 'ADDR C');`	`id = 1, name = 'A', address = 'ADDR A'` `id = 2, name = 'B', address = 'ADDR B'`	`id = 1, name = 'C', address = 'ADDR C'`
t3	Synchronization pushes Node A changes to Node B		**Row change for INSERT tx id = 1 on Node A results in unique key conflict on Node B** `id = 1, name = 'C', address = 'ADDR C'` **`id = 1, name = 'A', address = 'ADDR A'`**

The following table illustrates an example of an update conflict.

Table 6-2 - Update Conflict

Timestamp	Action	Master Node A	Master Node B
t0		`id = 2, address = 'ADDR B'`	`id = 2, address = 'ADDR B'`
t1	**Node A:** `UPDATE addrbook SET address = 'ADDR B1' WHERE id = 2;`	`id = 2, address =` **`'ADDR B1'`**	`id = 2, address = 'ADDR B'`
t2	**Node B:** `UPDATE addrbook SET address = 'ADDR B2' WHERE id = 2;`	`id = 2, address = 'ADDR B1'`	`id = 2, address =` **`'ADDR B2'`**
t3	Synchronization pushes Node A changes to Node B		**Current value of address on Node B not equal old value on Node A (`'ADDR B2' <> 'ADDR B'`)**

The following table illustrates an example of a delete conflict.

Table 6-3 - Delete Conflict

Timestamp	Action	Master Node A	Master Node B
t0		`id = 2, address = 'ADDR B'`	`id = 2, address = 'ADDR B'`
t1	**Node A:** `UPDATE addrbook SET address = 'ADDR B1' WHERE id = 2;`	`id = 2, address =` **`'ADDR B1'`**	`id = 2, address = 'ADDR B'`
t2	**Node B:** `DELETE FROM addrbook WHERE id = 2;`	`id = 2, address = 'ADDR B1'`	**Row with `id = 2` deleted**

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Timestamp	Action	Master Node A	Master Node B
t3	Synchronization pushes Node A changes to Node B		**The row with `id = 2` is already deleted on target Node B, hence update from Node A fails.**

6.6.3 Conflict Detection

This section discusses the synchronization process and conflict detection.

When synchronization replication occurs, either on demand or on a scheduled basis, each of the master node changes is pushed to the other master nodes. See Section 2.2.11 for information on this process.

Using a 3-node example the following describes the conflict detection process.

- The replication server loads the first set of pending transactions from master node A. Transactions are processed on a transaction set basis. (The same process is used for single-master replication.) All pending transactions are grouped in one or more transaction sets to avoid loading a very large chunk of rows in memory that may result in an out of heap space issue.
- For an update transaction, the replication server queries the first target master node B to load the related row. If the old column value on the source master node A is different than the current column value on target master node B, the transaction is marked as an update/update conflict. If a related row is not found on the target master node, it is marked as an update/delete conflict.
- For a delete transaction, the replication server queries the target master node to load the related row. If a related row is not found on the target master node, the transaction is marked as a delete/delete conflict.
- When a conflict is detected, the conflict information such as the transaction ID, conflict type, and conflict detection timestamp are logged in the conflict table on the target master node.
- For a conflicting transaction, the replication server checks if any conflict resolution strategy has been selected for the specific table. If a strategy is found, it is applied accordingly and the conflict status is marked as *resolved*. If a strategy cannot be applied, the conflict status is marked as *unresolved* (also called *pending*).
- If no conflict is detected, the transactional change is replicated to the target master node and the transaction status for that target node is marked as *completed* in the source master node control schema. A transaction status mapping for each target master node is maintained on all master nodes. For example node A contains two mappings of status – one for node B and another for node C.
- All of these prior steps are repeated to process and replicate all pending transaction sets available on master node A to master node B.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Next, the publication server proceeds to replicate master node A pending transactional changes to the next target master node, C.
- Once the master node A changes are replicated to nodes B and C, the publication server replicates the pending changes available on master node B to nodes A and C.
- Finally, the master node C changes are replicated to nodes A and B.

6.6.4 Conflict Resolution Strategies

A number of built-in conflict resolution options are available to support automatic conflict resolution. The conflict resolution options are applicable to update/update and delete/delete conflicts.

Uniqueness (insert/insert), update/delete, and delete/update conflicts are marked unresolved and must be manually reconciled.

The following are the built-in, automatic conflict resolution options.

- **Earliest Timestamp.** When the earliest timestamp option is selected, the relevant rows involved in an update conflict from the source and target master nodes are compared based on the timestamp of when the update occurred on that particular node. The row change that occurred earliest is applied. The row changes with the later timestamps are discarded.
- **Latest Timestamp.** Same approach as earliest timestamp except the row change with the latest timestamp is accepted. The row changes with earlier timestamps are discarded.
- **Node Priority.** The row change of the master node with the highest node priority level is applied while the lower priority level master node changes are discarded. The node priority level is an integer in the range of 1 to 10, inclusive where 1 is the highest priority level and 10 is the lowest priority level.
- **Custom.** Custom conflict handling applies to update/update conflicts only. You must supply a PL/pgSQL program to resolve any conflicts that occur resulting from an update/update conflict. See Section 6.6.8 for information on using custom conflict handling.

The delete/delete conflict is always resolved implicitly regardless of the conflict resolution option in effect. The net impact of a delete/delete conflict is the removal of a given row, and the row in question has already been removed the from the source and target nodes.

For the earliest timestamp and latest timestamp conflict resolution strategies, the transaction timestamp is tracked in a column with data type `TIMESTAMP` in the shadow table.

Once selected, the conflict resolution strategy for a given table can later be changed to a different strategy (see Section 6.8).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.6.5 Conflict Prevention – Uniqueness Case

Since there is no automatic built-in resolution strategy for the uniqueness conflict, this section discusses strategies to avoid this problem that would be implemented by the DBA. This discussion is based on a realm of numeric values generated by a sequence such as for a unique primary key.

The following are possible strategies:

- **Node specific sequence range.** A sequence range is reserved for each master node. For example, master node A would have `MINVALUE = 1` and `MAXVALUE = 1000`, master node B would have `MINVALUE = 1001` and `MAXVALUE = 2000`, and so on for other nodes. This ensures that a unique ID is always generated across all master nodes.
- **Start value variation.** Each node is assigned a different start value. For example, master node A would have a `START` value of 1, node B would have 2, and node C would have 3. An increment greater than or equal to the number of nodes guarantees unique IDs as shown in Table 6-4.
- **Common sequence.** All nodes share a common sequence object, however this has the major disadvantage of slowing down transaction processing due to network round-trips associated with each ID generation.
- **MMR-ready sequence.** This is a technique that enhances the use of sequences and provides a more flexible, reliable approach for a distributed, multiple database architecture as is inherent in a multi-master replication system. This approach is recommended over the previously listed sequence techniques. See Section 6.6.6 for information on an MMR-ready sequence.

Table 6-4 - Sequence With Start Value Variation

Sequence Clause	Master Node A	Master Node B	Master Node C
`START WITH`	1	2	3
`INCREMENT BY`	5	5	5
Generated IDs	1, 6, 11, 16, …	2, 7, 12, 17, …	3, 8, 13, 18, …

6.6.6 Conflict Prevention with an MMR-Ready Sequence

To prevent uniqueness conflicts in a multi-master replication system, an *MMR-ready sequence* can be used to generate unique identifiers for each row of publication tables that do not have an inherent, unique identifier.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

An MMR-ready sequence incorporates a function and a sequence to return `BIGINT` data type, integer values. These values combine a user-assigned, unique database identifier for each master node with a sequence generated within that master node.

A publication table requiring an MMR-ready sequence can be altered to include a `BIGINT NOT NULL` column with a default value returned by the function.

An MMR-ready sequence satisfies the following characteristics:

- **Uniqueness.** The combination of the unique, database identifier with the sequence ensures that each row in a given table will have a unique value across all master nodes.
- **Clustered index support.** An MMR-ready sequence does not impair the usage of a clustered index to provide retrieval efficiency. MMR-ready sequence values are returned in a typical, ordered sequence – not as random values such as if the universally unique identifier (UUID) were used.
- **Effective migration support.** Tables already utilizing a sequence can be modified to use an MMR-ready sequence with minimal impact on existing primary keys and foreign keys.
- **Reliability and maintainability.** In summary, an MMR-ready sequence provides a reliable and maintainable method to avoid uniqueness conflicts.

The following sections provide the steps for creating an MMR-ready sequence followed by an example. The conversion process for existing sequences is described in Section 6.6.6.3.

6.6.6.1 Creating an MMR-Ready Sequence

The following are the steps to create an MMR-ready sequence in a database to participate as a master node in a multi-master replication system.

Begin these steps with the database to be used as the master definition node.

Step 1: Assign a unique, database identifier as an integer from 1 to 1024, inclusive. Thus, a maximum of 1024 databases can be uniquely identified in a multi-master replication system with an MMR-ready sequence.

Issue the following commands to create and set the database identifier:

```
ALTER DATABASE dbname SET cluster.unique_db_id TO db_id;
SET cluster.unique_db_id TO db_id;
```

Use a different *db_id* value for each database.

Step 2: Create a sequence to uniquely identify each table row within the database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
CREATE SEQUENCE seq_name START WITH 1 INCREMENT BY 1 NO
CYCLE;
```

Multiple sequences can be created if it is desired to use separate sequences for multiple tables within the publication. Be sure that the same sequence name is used across all databases for the same given table.

A publication table column that uses an MMR-ready sequence will include a `DEFAULT` clause referencing the sequence name in a function call. The publication table definition must be consistent across all master nodes by referencing the same sequence name in the function call.

Step 3: Create the following function that returns the next MMR-ready sequence value when a row is inserted into the table. This function is referenced by the `DEFAULT` clause of the publication table column.

```
CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;
```

The sequence name created in Step 2 is specified as the `seq_id` input argument when the function is added to the `DEFAULT` clause of the publication table column.

This function performs a bitwise shift left operation (`<< 52`) on the database identifier (`cluster.unique_db_id`), thus significantly increasing its numeric value. The next sequence value is then added to this number. Thus, all rows inserted in the table on a given database fall within a numeric range determined by the shifted, database identifier value.

Step 4 (Optional): Create the following function to obtain the current MMR-ready sequence value.

```
CREATE OR REPLACE FUNCTION mmr_sequence_currval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  currval($1::regclass);
$function$;
```

The `mmr_sequence_nextval` function must be invoked in the current session before calling the `mmr_sequence_currval` function.

Step 5: Add or modify the publication table column that is to use the MMR-ready sequence. The column data type must be `BIGINT`. The `mmr_sequence_nextval` function is specified in the `DEFAULT` clause as shown in the following example for column `id`.

```
CREATE TABLE table_name (
    id          BIGINT NOT NULL PRIMARY KEY
                DEFAULT mmr_sequence_nextval('seq_name'),
    field       VARCHAR2(20)
);
```

The column will also typically be the primary key.

Step 6: Repeat steps 1 through 4 for the other databases to be added as master nodes.

Note that Step 5 is omitted for the additional master nodes as the publication table definitions are replicated from the master definition node to the additional master nodes when they are created as described in Section 6.3.

Step 7: Create the complete, multi-master replication system as described in Chapter 6.

6.6.6.2 MMR-Ready Sequence Example

The following is an example of a 3-master node system using an MMR-ready sequence. The databases to be used as the master nodes are `mmrnode_a`, `mmrnode_b`, and `mmrnode_c`. A publication table named `mmr_seq_tbl` uses the MMR-ready sequence.

The following commands are invoked in database `mmrnode_a`, which will be the master definition node:

```
ALTER DATABASE mmrnode_a SET cluster.unique_db_id TO 1;
SET cluster.unique_db_id TO 1;

CREATE SEQUENCE mmr_seq START WITH 1 INCREMENT BY 1 NO CYCLE;

CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;

CREATE OR REPLACE FUNCTION mmr_sequence_currval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  currval($1::regclass);
$function$;

CREATE TABLE mmr_seq_tbl (
    id          BIGINT NOT NULL PRIMARY KEY
                DEFAULT mmr_sequence_nextval('mmr_seq'),
    field       VARCHAR2(20)
);
```

On `mmrnode_b` and `mmrnode_c`, the commands to create different settings for the configuration parameter `cluster.unique_db_id` are run as well as the commands to create the sequence and the functions.

On `mmrnode_b` the following commands are invoked. Note that `cluster.unique_db_id` is set to `2`.

```
ALTER DATABASE mmrnode_b SET cluster.unique_db_id TO 2;
SET cluster.unique_db_id TO 2;

CREATE SEQUENCE mmr_seq START WITH 1 INCREMENT BY 1 NO CYCLE;

CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;

CREATE OR REPLACE FUNCTION mmr_sequence_currval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  << 52)::bigint +
  currval($1::regclass);
$function$;
```

On `mmrnode_c` the following commands are invoked. Note that `cluster.unique_db_id` is set to `3`.

```
ALTER DATABASE mmrnode_c SET cluster.unique_db_id TO 3;
SET cluster.unique_db_id TO 3;

CREATE SEQUENCE mmr_seq START WITH 1 INCREMENT BY 1 NO CYCLE;

CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;

CREATE OR REPLACE FUNCTION mmr_sequence_currval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  currval($1::regclass);
$function$;
```

The multi-master replication system is created with the Replicate Publication Schema and the Perform Initial Snapshot options selected when creating the additional master nodes, `mmrnode_b` and `mmrnode_c`.

The resulting master nodes are shown in the xDB Replication Console. Note that the Default Value property of the `id` column uses the `mmr_sequence_nextval` function.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

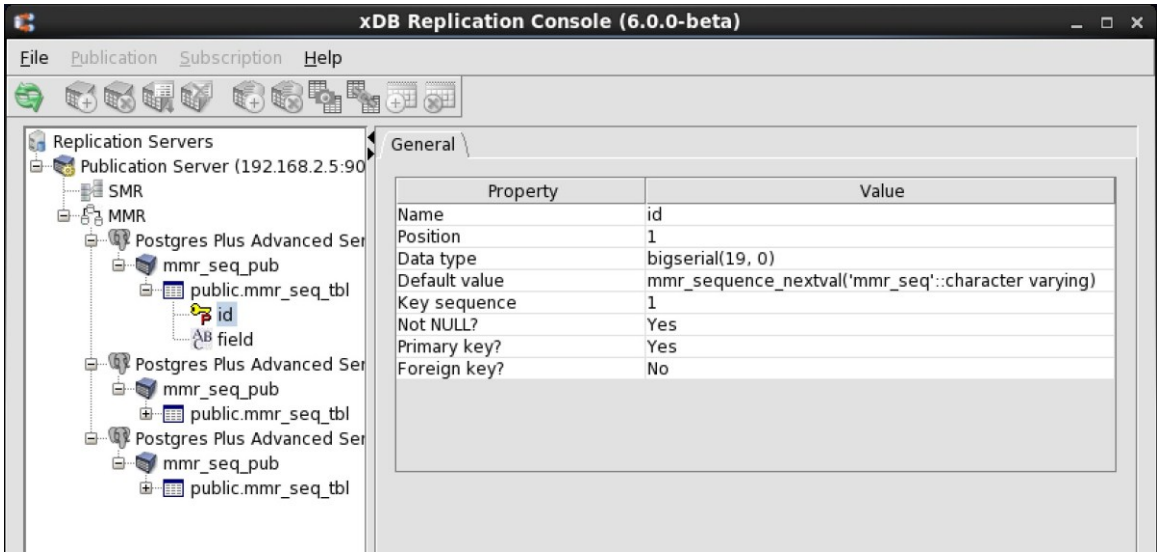


Figure 6-27 – Publication table with MMR-ready sequence

The following `INSERT` commands are invoked on `mmrnode_a`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 1');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 2');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 3');
```

The following `INSERT` commands are invoked on `mmrnode_b`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_b - Row 1');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_b - Row 2');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_b - Row 3');
```

The following `INSERT` commands are invoked on `mmrnode_c`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_c - Row 1');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_c - Row 2');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_c - Row 3');
```

A synchronization replication is performed.

No uniqueness conflicts occur as a unique value is generated for the `id` primary key column as shown by the following results on `mmrnode_a`:

```
mmrnode_a=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  9007199254740993 | mmrnode_b - Row 1
  9007199254740994 | mmrnode_b - Row 2
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  9007199254740995 | mmrnode_b - Row 3
 13510798882111489 | mmrnode_c - Row 1
 13510798882111490 | mmrnode_c - Row 2
 13510798882111491 | mmrnode_c - Row 3
(9 rows)
```

The same query on `mmrnode_b` shows the same set of rows:

```
mmrnode_b=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  9007199254740993 | mmrnode_b - Row 1
  9007199254740994 | mmrnode_b - Row 2
  9007199254740995 | mmrnode_b - Row 3
 13510798882111489 | mmrnode_c - Row 1
 13510798882111490 | mmrnode_c - Row 2
 13510798882111491 | mmrnode_c - Row 3
(9 rows)
```

The same results are present on `mmrnode_c`:

```
mmrnode_c=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  9007199254740993 | mmrnode_b - Row 1
  9007199254740994 | mmrnode_b - Row 2
  9007199254740995 | mmrnode_b - Row 3
 13510798882111489 | mmrnode_c - Row 1
 13510798882111490 | mmrnode_c - Row 2
 13510798882111491 | mmrnode_c - Row 3
(9 rows)
```

6.6.6.3 Converting a Standard Sequence to an MMR-Ready Sequence

If you have an existing application with tables that use a standard sequence such as with the `SERIAL` data type, these tables can be modified to use the MMR-ready sequence for incorporation into a multi-master replication system.

The basic conversion process is the following:

- Update the sequence values in the existing rows with MMR-ready sequence compatible values.
- Alter the column definition to be compatible with the MMR-ready sequence including modification or addition of the `DEFAULT` clause to use the MMR-ready sequence function to supply the default values for subsequent inserts.

To perform the conversion of existing sequence values, first, create the unique database identifier as described in Step 1 of Section 6.6.6.1.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Use the following function to update the existing primary key and foreign key values that are required to be converted.

```
CREATE OR REPLACE FUNCTION mmr_sequence_convert (
    old_seq_value   bigint
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint + $1;
$function$;
```

The function input argument is the old sequence value and the function returns the new MMR-ready sequence value.

The function input and return arguments are data type `BIGINT` so the existing sequence columns must be altered accordingly before using the function.

Finally, the sequence columns must include the clauses `BIGINT NOT NULL DEFAULT mmr_sequence_nextval('seq_name')` to supply MMR-ready sequence values for future inserts.

See Section 6.6.6.1 for information on creating the objects required for an MMR-ready sequence.

6.6.6.4 Conversion to an MMR-Ready Sequence Example

This section describes a basic example of how two tables with standard sequences used as primary keys as well as a parent-child relationship by a foreign key constraint can be converted to use the MMR-ready sequence, then employed in a multi-master replication system.

The tables are defined as follows:

```
CREATE TABLE mmr_seq_tbl (
    id          SERIAL PRIMARY KEY,
    field       VARCHAR2(20)
);

CREATE TABLE mmr_seq_child_tbl (
    id          SERIAL PRIMARY KEY,
    field       VARCHAR2(20),
    parent_id   INTEGER CONSTRAINT mmr_seq_tbl_fk
                REFERENCES mmr_seq_tbl(id)
);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note the foreign key constraint between columns `mmr_seq_child_tbl.parent_id` and `mmr_seq_tbl.id`.

The tables are populated with an initial set of rows:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 1');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 2');
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 3');

INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 1-1', 1);
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 1-2', 1);
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 2-1', 2);
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 2-2', 2);
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 3-1', 3);
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 3-2', 3);
```

The resulting table content is the following:

```
edb=# SELECT * FROM mmr_seq_tbl;
 id |       field
----+------------------
  1 | mmrnode_a - Row 1
  2 | mmrnode_a - Row 2
  3 | mmrnode_a - Row 3
(3 rows)

edb=# SELECT * FROM mmr_seq_child_tbl;
 id |        field        | parent_id
----+---------------------+-----------
  1 | mmrnode_a - Row 1-1 |         1
  2 | mmrnode_a - Row 1-2 |         1
  3 | mmrnode_a - Row 2-1 |         2
  4 | mmrnode_a - Row 2-2 |         2
  5 | mmrnode_a - Row 3-1 |         3
  6 | mmrnode_a - Row 3-2 |         3
(6 rows)
```

Prior to performing the conversion, obtain the current, maximum sequence value of the sequence to be converted to an MMR-ready sequence. In this example the value is 6 as seen in the `id` column of table `mmr_seq_child_tbl`.

This value is needed to set a newly created sequence that is to be used for the MMR-ready sequence, to a large enough starting value to avoid uniqueness conflict with the converted sequence values of the existing rows.

Converting Existing Standard Sequence Values

In order to convert the existing sequence values in columns `mmr_seq_tbl.id`, `mmr_seq_child_tbl.id`, and `mmr_seq_child_tbl.parent_id` the following steps are performed.

Permit deferred updates to the foreign key constraint.

```
ALTER TABLE mmr_seq_child_tbl ALTER CONSTRAINT mmr_seq_tbl_fk DEFERRABLE
INITIALLY DEFERRED;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Create the function to perform the sequence conversion.

```
CREATE OR REPLACE FUNCTION mmr_sequence_convert (
    old_seq_value   bigint
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint + $1;
$function$;
```

Change the sequence columns to data type `BIGINT` so they are large enough for the MMR-ready sequence.

```
ALTER TABLE mmr_seq_tbl ALTER COLUMN id SET DATA TYPE BIGINT;
ALTER TABLE mmr_seq_child_tbl ALTER COLUMN id SET DATA TYPE BIGINT;
ALTER TABLE mmr_seq_child_tbl ALTER COLUMN parent_id SET DATA TYPE BIGINT;
```

Set the unique database identifier to be used by the MMR-ready sequence.

```
ALTER DATABASE mmrnode_a SET cluster.unique_db_id TO 1;
SET cluster.unique_db_id TO 1;
```

Update the primary key and foreign key values with the `mmr_sequence_convert` function.

The updates affecting the foreign key constraint must be performed within the same transaction to avoid a foreign key violation error.

```
BEGIN TRANSACTION;
    UPDATE mmr_seq_tbl SET id = mmr_sequence_convert (id);
    UPDATE mmr_seq_child_tbl SET parent_id = mmr_sequence_convert (parent_id);
    UPDATE mmr_seq_child_tbl SET id = mmr_sequence_convert (id);
COMMIT;
```

Reset the foreign key constraint back to its original setting – for example:

```
ALTER TABLE mmr_seq_child_tbl ALTER CONSTRAINT mmr_seq_tbl_fk NOT DEFERRABLE;
```

After the conversion to the MMR-ready sequence, the table content is as follows:

```
edb=# SELECT * FROM mmr_seq_tbl;
        id         |       field
-------------------+-------------------
 4503599627370497 | mmrnode_a - Row 1
 4503599627370498 | mmrnode_a - Row 2
 4503599627370499 | mmrnode_a - Row 3
(3 rows)

edb=# SELECT * FROM mmr_seq_child_tbl;
        id         |        field        |    parent_id
-------------------+---------------------+-----------------
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
 4503599627370497 | mmrnode_a - Row 1-1 | 4503599627370497
 4503599627370498 | mmrnode_a - Row 1-2 | 4503599627370497
 4503599627370499 | mmrnode_a - Row 2-1 | 4503599627370498
 4503599627370500 | mmrnode_a - Row 2-2 | 4503599627370498
 4503599627370501 | mmrnode_a - Row 3-1 | 4503599627370499
 4503599627370502 | mmrnode_a - Row 3-2 | 4503599627370499
(6 rows)
```

The parent-child foreign key relationship between columns `mmr_seq_child_tbl.parent_id` and `mmr_seq_tbl.id` is maintained.

The primary key `id` values incorporate the old sequence values, but are increased by the addition of the 52-bit shifted, database identifier value.

Setting Up the MMR-Ready Sequence

The steps as described in Section 6.6.6.1 are now performed on the databases to be used as master nodes.

For database `mmrnode_a` that contains the converted tables, a new sequence is created with a starting value of 7 to avoid a primary key uniqueness conflict with the existing rows. In the original tables, the maximum used sequence value was 6.

When a sequence number is transformed to an MMR-ready sequence value, the same result is returned if the same database identifier is used with the same original number.

```
CREATE SEQUENCE mmr_seq START WITH 7 INCREMENT BY 1 NO CYCLE;
```

Create the function to return the MMR-ready sequence value.

```
CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;
```

Alter the primary key columns to use the function to return the default value.

```
ALTER TABLE mmr_seq_tbl ALTER COLUMN id SET DEFAULT mmr_sequence_nextval('mmr_seq');
ALTER TABLE mmr_seq_child_tbl ALTER COLUMN id SET DEFAULT
mmr_sequence_nextval('mmr_seq');
```

Repeat the MMR-ready sequence setup process for the other master nodes.

```
ALTER DATABASE mmrnode_b SET cluster.unique_db_id TO 2;
SET cluster.unique_db_id TO 2;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
CREATE SEQUENCE mmr_seq START WITH 1 INCREMENT BY 1 NO CYCLE;

CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;
```

Repeat the process for `mmrnode_c`.

```
ALTER DATABASE mmrnode_c SET cluster.unique_db_id TO 3;
SET cluster.unique_db_id TO 3;

CREATE SEQUENCE mmr_seq START WITH 1 INCREMENT BY 1 NO CYCLE;

CREATE OR REPLACE FUNCTION mmr_sequence_nextval (
    seq_id          VARCHAR
)
RETURNS bigint
LANGUAGE sql
AS
$function$
SELECT (
 (SELECT current_setting('cluster.unique_db_id'))::bigint
  << 52)::bigint +
  nextval($1::regclass);
$function$;
```

Tables After Initial Multi-Master Replication System Creation

The multi-master replication system is created using databases `mmrnode_a`, `mmrnode_b`, and `mmrnode_c` in a similar manner as described in Section 6.6.6.2.

After the system is created with the initial snapshot, `mmrnode_a`, `mmrnode_b`, and `mmrnode_c` all contain identical content. The following is the table content:

```
mmrnode_a=# SELECT * FROM mmr_seq_tbl;
        id        |       field
------------------+------------------
 4503599627370497 | mmrnode_a - Row 1
 4503599627370498 | mmrnode_a - Row 2
 4503599627370499 | mmrnode_a - Row 3
(3 rows)

mmrnode_a=# SELECT * FROM mmr_seq_child_tbl;
        id        |        field        |    parent_id
------------------+---------------------+------------------
 4503599627370497 | mmrnode_a - Row 1-1 | 4503599627370497
 4503599627370498 | mmrnode_a - Row 1-2 | 4503599627370497
 4503599627370499 | mmrnode_a - Row 2-1 | 4503599627370498
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
 4503599627370500 | mmrnode_a - Row 2-2 | 4503599627370498
 4503599627370501 | mmrnode_a - Row 3-1 | 4503599627370499
 4503599627370502 | mmrnode_a - Row 3-2 | 4503599627370499
(6 rows)
```

Subsequent Row Insertions and Synchronization

The following rows are inserted on `mmrnode_a`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_a - Row 4');
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_a - Row 4-
1', 4503599627370503);

mmrnode_a=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id        |       field
------------------+-------------------
 4503599627370497 | mmrnode_a - Row 1
 4503599627370498 | mmrnode_a - Row 2
 4503599627370499 | mmrnode_a - Row 3
 4503599627370503 | mmrnode_a - Row 4
(4 rows)

mmrnode_a=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id        |        field        |    parent_id
------------------+---------------------+------------------
 4503599627370497 | mmrnode_a - Row 1-1 | 4503599627370497
 4503599627370498 | mmrnode_a - Row 1-2 | 4503599627370497
 4503599627370499 | mmrnode_a - Row 2-1 | 4503599627370498
 4503599627370500 | mmrnode_a - Row 2-2 | 4503599627370498
 4503599627370501 | mmrnode_a - Row 3-1 | 4503599627370499
 4503599627370502 | mmrnode_a - Row 3-2 | 4503599627370499
 4503599627370504 | mmrnode_a - Row 4-1 | 4503599627370503
(7 rows)
```

The following rows are inserted on `mmrnode_b`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_b - Row 1');
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_b - Row 1-
1', 9007199254740993);

mmrnode_b=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id        |       field
------------------+-------------------
 4503599627370497 | mmrnode_a - Row 1
 4503599627370498 | mmrnode_a - Row 2
 4503599627370499 | mmrnode_a - Row 3
 9007199254740993 | mmrnode_b - Row 1
(4 rows)

mmrnode_b=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id        |        field        |    parent_id
------------------+---------------------+------------------
 4503599627370497 | mmrnode_a - Row 1-1 | 4503599627370497
 4503599627370498 | mmrnode_a - Row 1-2 | 4503599627370497
 4503599627370499 | mmrnode_a - Row 2-1 | 4503599627370498
 4503599627370500 | mmrnode_a - Row 2-2 | 4503599627370498
 4503599627370501 | mmrnode_a - Row 3-1 | 4503599627370499
 4503599627370502 | mmrnode_a - Row 3-2 | 4503599627370499
 9007199254740994 | mmrnode_b - Row 1-1 | 9007199254740993
(7 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following rows are inserted on `mmrnode_c`:

```
INSERT INTO mmr_seq_tbl (field) VALUES ('mmrnode_c - Row 1');
INSERT INTO mmr_seq_child_tbl (field, parent_id) VALUES ('mmrnode_c - Row 1-
1', 13510798882111489);

mmrnode_c=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
 13510798882111489 | mmrnode_c - Row 1
(4 rows)

mmrnode_c=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id         |        field        |     parent_id
-------------------+---------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1-1 |  4503599627370497
  4503599627370498 | mmrnode_a - Row 1-2 |  4503599627370497
  4503599627370499 | mmrnode_a - Row 2-1 |  4503599627370498
  4503599627370500 | mmrnode_a - Row 2-2 |  4503599627370498
  4503599627370501 | mmrnode_a - Row 3-1 |  4503599627370499
  4503599627370502 | mmrnode_a - Row 3-2 |  4503599627370499
 13510798882111490 | mmrnode_c - Row 1-1 | 13510798882111489
(7 rows)
```

After a synchronization replication is performed, there are no uniqueness conflicts. The following shows the synchronized, consistent tables in the master nodes:

Content of `mmrnode_a` after synchronization:

```
mmrnode_a=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  4503599627370503 | mmrnode_a - Row 4
  9007199254740993 | mmrnode_b - Row 1
 13510798882111489 | mmrnode_c - Row 1
(6 rows)

mmrnode_a=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id         |        field        |     parent_id
-------------------+---------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1-1 |  4503599627370497
  4503599627370498 | mmrnode_a - Row 1-2 |  4503599627370497
  4503599627370499 | mmrnode_a - Row 2-1 |  4503599627370498
  4503599627370500 | mmrnode_a - Row 2-2 |  4503599627370498
  4503599627370501 | mmrnode_a - Row 3-1 |  4503599627370499
  4503599627370502 | mmrnode_a - Row 3-2 |  4503599627370499
  4503599627370504 | mmrnode_a - Row 4-1 |  4503599627370503
  9007199254740994 | mmrnode_b - Row 1-1 |  9007199254740993
 13510798882111490 | mmrnode_c - Row 1-1 | 13510798882111489
(9 rows)
```

Content of `mmrnode_b` after synchronization:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
mmrnode_b=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  4503599627370503 | mmrnode_a - Row 4
  9007199254740993 | mmrnode_b - Row 1
 13510798882111489 | mmrnode_c - Row 1
(6 rows)

mmrnode_b=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id         |        field        |     parent_id
-------------------+---------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1-1 |  4503599627370497
  4503599627370498 | mmrnode_a - Row 1-2 |  4503599627370497
  4503599627370499 | mmrnode_a - Row 2-1 |  4503599627370498
  4503599627370500 | mmrnode_a - Row 2-2 |  4503599627370498
  4503599627370501 | mmrnode_a - Row 3-1 |  4503599627370499
  4503599627370502 | mmrnode_a - Row 3-2 |  4503599627370499
  4503599627370504 | mmrnode_a - Row 4-1 |  4503599627370503
  9007199254740994 | mmrnode_b - Row 1-1 |  9007199254740993
 13510798882111490 | mmrnode_c - Row 1-1 | 13510798882111489
(9 rows)
```

Content of `mmrnode_c` after synchronization:

```
mmrnode_c=# SELECT * FROM mmr_seq_tbl ORDER BY id;
        id         |       field
-------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1
  4503599627370498 | mmrnode_a - Row 2
  4503599627370499 | mmrnode_a - Row 3
  4503599627370503 | mmrnode_a - Row 4
  9007199254740993 | mmrnode_b - Row 1
 13510798882111489 | mmrnode_c - Row 1
(6 rows)

mmrnode_c=# SELECT * FROM mmr_seq_child_tbl ORDER BY id;
        id         |        field        |     parent_id
-------------------+---------------------+-------------------
  4503599627370497 | mmrnode_a - Row 1-1 |  4503599627370497
  4503599627370498 | mmrnode_a - Row 1-2 |  4503599627370497
  4503599627370499 | mmrnode_a - Row 2-1 |  4503599627370498
  4503599627370500 | mmrnode_a - Row 2-2 |  4503599627370498
  4503599627370501 | mmrnode_a - Row 3-1 |  4503599627370499
  4503599627370502 | mmrnode_a - Row 3-2 |  4503599627370499
  4503599627370504 | mmrnode_a - Row 4-1 |  4503599627370503
  9007199254740994 | mmrnode_b - Row 1-1 |  9007199254740993
 13510798882111490 | mmrnode_c - Row 1-1 | 13510798882111489
(9 rows)
```

6.6.7 Automatic Conflict Resolution Example

This example illustrates a scenario where a transaction change originating from the first master node is successfully applied to the second master node, but conflicts with the third master node. The conflict is resolved automatically.

The conflict resolution option is set to latest timestamp.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Table 6-5 – Automatic Conflict Resolution Example

Timestamp	Action	Master Node A	Master Node B	Master Node C
t0		id = 2, address = 'ADDR'	id = 2, address = 'ADDR'	id = 2, address = 'ADDR'
t1	**Node A:** UPDATE addrbook SET address = 'ADDR A' WHERE id = 2;	id = 2, address = **'ADDR A'**	id = 2, address = 'ADDR'	id = 2, address = 'ADDR'
t2	**Node C:** UPDATE addrbook SET address = 'ADDR C' WHERE id = 2;	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR'	id = 2, address = **'ADDR C'**
t3	Synchronization pushes Node A changes to Node B. Changes successfully applied.	id = 2, address = 'ADDR A'	id = 2, address = **'ADDR A'**	id = 2, address = 'ADDR C'
t4	Synchronization pushes Node A changes to Node C. Current address on Node C <> old value on Node A ('ADDR C' <> 'ADDR') hence conflict detected. Latest change on Node C accepted and Node A change discarded.	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR A'	id = 2, address = **'ADDR C'**
t5	No changes on Node B. Node C changes pushed to Node A that is successfully applied (Node A change already marked as discarded and hence is overwritten.)	id = 2, address = **'ADDR C'**	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR C'
t6	Node C changes pushed to Node B that is successfully applied. All nodes are in sync and have consistent state.	id = 2, address = 'ADDR C'	id = 2, address = **'ADDR C'**	id = 2, address = 'ADDR C'

There are a few situations where an update/update conflict may not be properly resolved according to the selected resolution options. These exceptions are described by the following.

Update/Update Conflict on Column Where New Value is Identical to Original Value

If there is an update to a publication table where the updated column value happens to be the same as the original column value, and then an update/update conflict occurs involving that column, there is the possibility that the final value of this column is not set

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

according to the chosen conflict resolution option in one of the conflicting master nodes. This is a known limitation.

For example, consider the following row in the `dept` table:

```
deptno |   dname    |  loc
--------+------------+--------
     40 | OPERATIONS | BOSTON
```

First, the following `UPDATE` statement is given in the master definition node:

```
edb=# UPDATE dept SET dname = 'OPERATIONS', loc = 'BEDFORD' WHERE deptno =
40;
UPDATE 1
edb=# SELECT * FROM dept WHERE deptno = 40;
 deptno |   dname    |   loc
--------+------------+---------
     40 | OPERATIONS | BEDFORD
(1 row)
```

Note that the original value, `OPERATIONS`, of column `dname` is the same as the value to which it is changed in the `UPDATE` statement.

The following `UPDATE` statement is then given in a second master node:

```
mmrnode=# UPDATE dept SET dname = 'LOGISTICS', loc = 'CAMBRIDGE' WHERE deptno
= 40;
UPDATE 1
mmrnode=# SELECT * FROM dept WHERE deptno = 40;
 deptno |   dname   |    loc
--------+-----------+-----------
     40 | LOGISTICS | CAMBRIDGE
(1 row)
```

After a synchronization replication using the earliest timestamp conflict resolution option, the row in the master definition node retains the update performed on it as expected since the update on the master definition node occurred first.

```
edb=# SELECT * FROM dept WHERE deptno = 40;
 deptno |   dname    |   loc
--------+------------+---------
     40 | OPERATIONS | BEDFORD
(1 row)
```

However the value of column `dname` in the second master node remains set to `LOGISTICS`. It was not reverted back to the value `OPERATIONS` from the master definition node as would normally be expected on a conflicting column. Note that as expected, the value in column `loc` is reverted from `CAMBRIDGE` back to the master definition node value of `BEDFORD`.

```
mmrnode=# SELECT * FROM dept WHERE deptno = 40;
 deptno |   dname   |   loc
--------+-----------+---------
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
     40 | LOGISTICS | BEDFORD
(1 row)
```

Update/Update Conflict on Primary Key Columns

An update/update conflict on the primary key column may not be consistently resolved according to the selected resolution option. That is, the column values may differ for the same row across multiple master nodes following the synchronization replication.

In addition, this conflict may not appear under the Conflict History tab in the xDB Replication Console. Even if a conflict resolution entry does appear under the Conflict History tab, the actual primary key values may not be consistent across the nodes as implied by the conflict resolution.

6.6.8 Custom Conflict Handling

For update/update conflicts, custom conflict handling utilizes a PL/pgSQL function to resolve the conflict. If you are using Advanced Server, a Stored Procedure Language (SPL) function can be used as well. When an update/update conflict is detected the function is called. How you set a certain parameter in the function determines the outcome of the conflict.

You must provide the function and add it to the master definition node using a utility such as PSQL or pgAdmin (Postgres Enterprise Manager Client in Advanced Server).

6.6.8.1 Custom Conflict Handling Function

An update/update conflict occurs if there is at least one conflicting column in the table.

A column is considered a *conflicting column* if it is updated on more than one master node in the same synchronization. Even if the new, updated value for the column is identical in the conflicting update transactions, the fact that the same column was updated on more than one master node makes it a conflicting column.

Each publication table must have its own custom conflict handling function to handle custom resolution for update/update conflicts on that particular publication table.

Custom conflict handling is designed to provide one of the following three outcomes based upon the setting of the `resolution_code` parameter, which is described later in this section.

- **Columns are to be set to the source node.** When the `resolution_code` parameter of the function is set to a value of `1`, the resultant setting of all columns in both conflicting nodes is obtained from the source node of the replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- **Columns are to be set to the target node.** When the *resolution_code* parameter of the function is set to a value of 2, the resultant setting of all columns in both conflicting nodes is obtained from the target node of the replication.
- **The function logic sets the column.** When the *resolution_code* parameter of the function is set to a value of 3, the resultant setting of the first conflicting column is obtained from the value returned in the *source* parameter coded within the function logic. The resultant setting of all other column values is obtained from the source node of the replication.

The following is an example of a custom conflict handling function where the conflicting columns are set to the target node.

```
CREATE OR REPLACE FUNCTION edb.custom_conflict_dept (
    INOUT   source              _edb_replicator_pub.rrst_edb_dept,
    IN      target              _edb_replicator_pub.rrst_edb_dept,
    IN      conflict_column     VARCHAR(255),
    OUT     resolution_message  VARCHAR(255),
    OUT     resolution_code     INTEGER
)
AS
$$
DECLARE
BEGIN
    resolution_code := 2;
    resolution_message := 'Custom conflict handling: Target node wins on
edb.dept ';
END;
$$
LANGUAGE plpgsql;
```

If the multi-master replication system is configured with the log-based method of synchronization replication the shadow tables of the `INOUT source` and `IN target` parameters are replaced with the actual publication tables as shown by the following:

```
CREATE OR REPLACE FUNCTION edb.custom_conflict_dept (
    INOUT   source              edb.dept,
    IN      target              edb.dept,
    IN      conflict_column     VARCHAR(255),
    OUT     resolution_message  VARCHAR(255),
    OUT     resolution_code     INTEGER
)
AS
$$
DECLARE
BEGIN
    resolution_code := 2;
    resolution_message := 'Custom conflict handling: Target node wins on
edb.dept ';
END;
$$
LANGUAGE plpgsql;
```

The following is an example of a custom conflict handling function where the function logic determines the value of the first conflicting column.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
CREATE OR REPLACE FUNCTION edb.custom_conflict_emp (
    INOUT   source              _edb_replicator_pub.rrst_edb_emp,
    IN      target              _edb_replicator_pub.rrst_edb_emp,
    IN      conflict_column     VARCHAR(255),
    OUT     resolution_message  VARCHAR(255),
    OUT     resolution_code     INTEGER
)
AS
$$
DECLARE
BEGIN
    resolution_code := 3;
    source.ename := 'Unknown';
    source.job := 'Unknown';
    source.mgr := 0;
    source.hiredate := '2013-01-01';
    source.sal := 0;
    source.comm := 0;
    resolution_message := 'Custom conflict handling: Defaults set on
edb.emp';
END;
$$
LANGUAGE plpgsql;
```

In this example, only the first conflicting column (based upon the column order within the table) is set to the value coded in the function. All other assignments to the *source* parameter are ignored. These other columns are set to the source node.

The following is a description of the function parameters.

Parameters

source

> INOUT parameter of the record type of the shadow table in schema _edb_replicator_pub of the master definition node on which conflicts are to be resolved. If the log-based method of synchronization replication is used, specify the actual publication table instead of the shadow table. The input values are the column values from the source node. When *resolution_code* is set to a value of 3, set the columns in this parameter to the values that are to be used for the final outcome.

target

> IN parameter of the record type of the shadow table in schema _edb_replicator_pub of the master definition node on which conflicts are to be resolved. If the log-based method of synchronization replication is used, specify the actual publication table instead of the shadow table. The input values are the column values from the target node.

conflict_column

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

IN parameter of type VARCHAR(255) containing the name of the column on which the update/update conflict has occurred. If more than one column is involved in the conflict, the name of the first conflicting column is returned.

resolution_message

OUT parameter of type VARCHAR(255) containing any informative message to be written to the publication server log file. The publication server configuration option logging.level must be set to at least the INFO level in order for the messages to appear in the publication server log file. See Section 3.5 for the location of the publication server log file.

resolution_code

OUT parameter of type INTEGER that you set to one of the following values to determine how to resolve the conflict: 1 to use the column values of the source node of the replication for the final outcome, 2 to use the column values of the target node of the replication for the final outcome, or 3 to use the value set for the *source* INOUT parameter of the first conflicting column as the final outcome for that column.

6.6.8.2 Adding a Custom Conflict Handling Function

The following are the steps to add a custom conflict handling function to the master definition node.

Step 1: The publication under the master definition node must exist before adding the function to the master definition node. See Section 6.2.3 for information on creating the publication.

Step 2: Add the function to the master definition node. The following example shows the addition of the function using PSQL.

```
edb=# \i /home/user/custom_conflict_dept.sql
CREATE FUNCTION
```

Step 3: Open the Conflict Resolution Options tab in any of the following ways:

- From the Publication menu, choose Update Publication, then Conflict Resolution Options.
- Click the secondary mouse button on the Publication node, choose Update Publication, and then choose Conflict Resolution Options.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

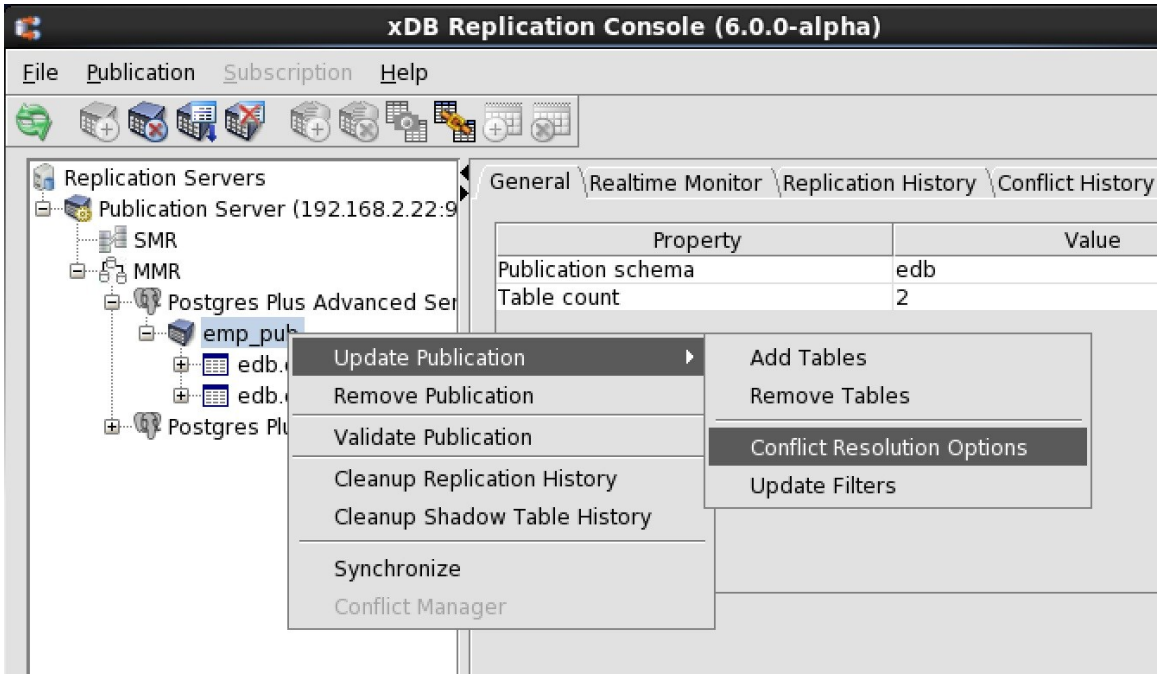


Figure 6-28 - Opening the Conflict Resolution Options tab

Step 4: For the table on which you want to use custom conflict handling, select Custom from the appropriate drop-down list. In the Custom Handler text box, enter the schema and function name used in the `CREATE FUNCTION` statement.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

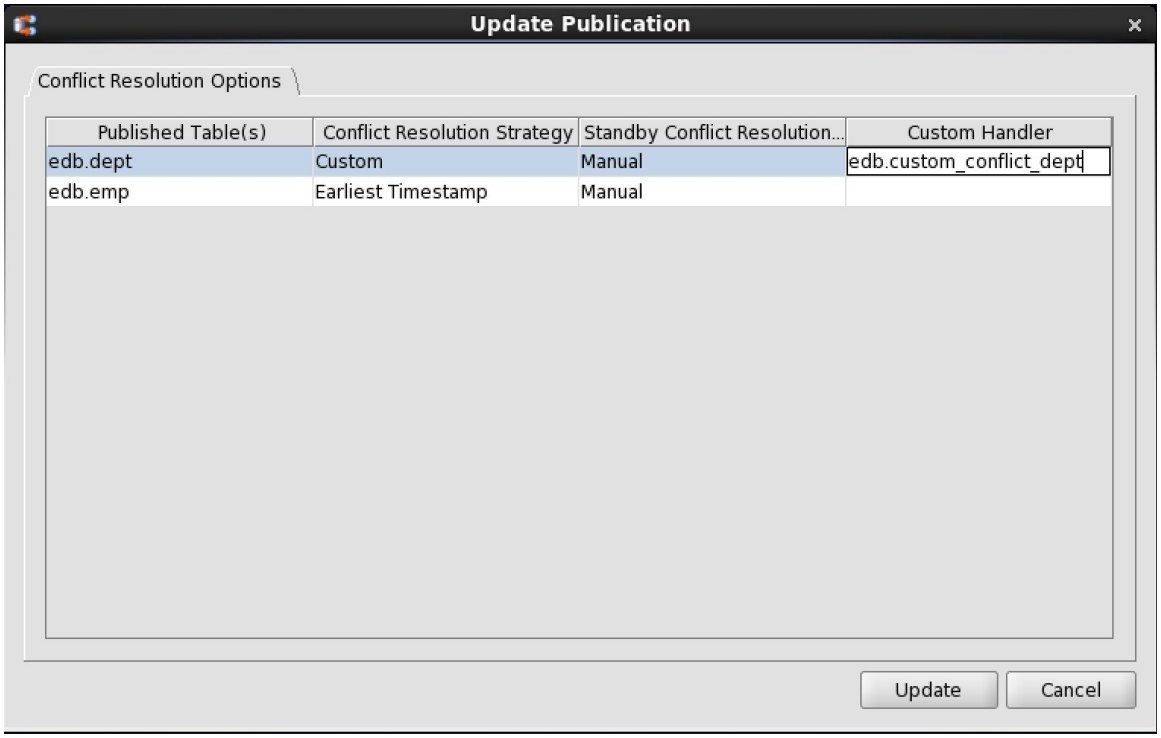


Figure 6-29 – Setting conflict resolution to custom conflict handling

Step 5: Click the Update button, and then click OK in response to the Conflict Resolution Options Updated Successfully confirmation message.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

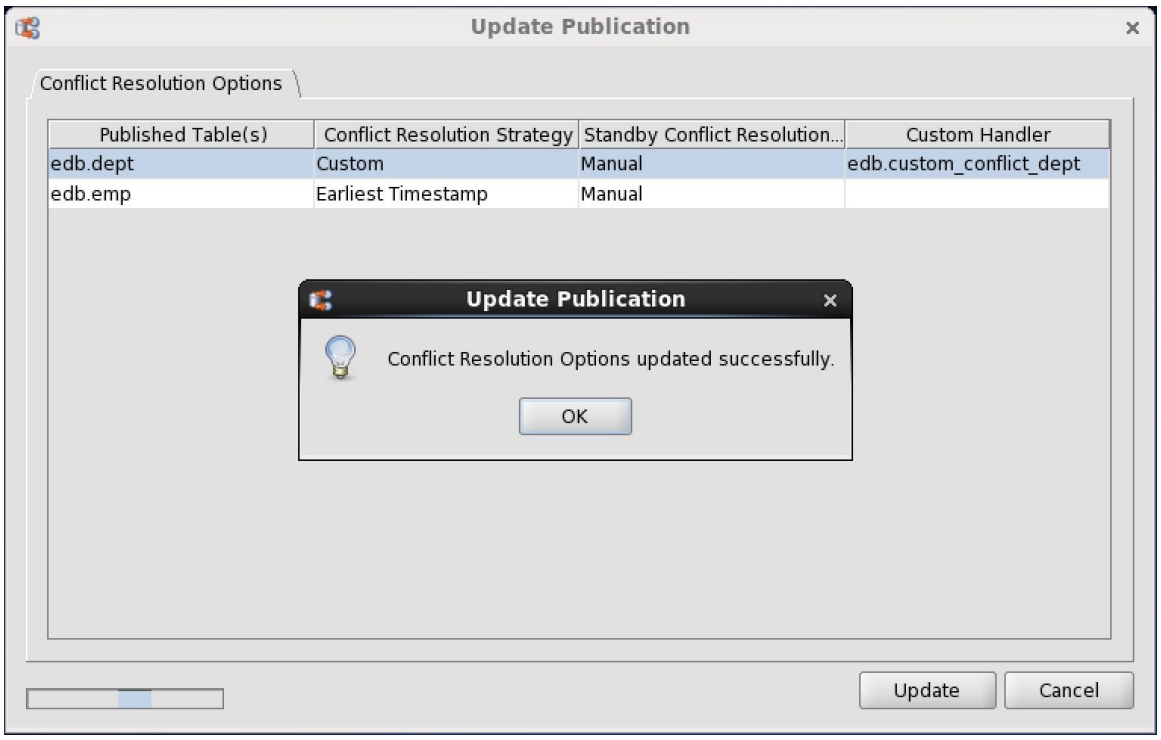


Figure 6-30 - Successfully updated conflict resolution options

Note: If the multi-master replication system uses custom conflict handling, and you subsequently switch the role of the master definition node to another master node, you must re-add the functions to the new master definition node. That is, you must repeat Step 2 on the new master definition node.

Note: If you wish to delete the multi-master replication system, before removing the publication you must drop all custom conflict handling functions from the master definition node.

The following example shows the deletion of a custom conflict function.

```
DROP FUNCTION
edb.custom_conflict_dept(_edb_replicator_pub.rrst_edb_dept,_edb_replicator_pu
b.rrst_edb_dept,varchar);
```

6.6.8.3 Custom Conflict Handling Examples

This section contains examples using custom conflict handling functions.

Setting Columns from the Source or Target

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following example shows the effect of custom conflict handling using the custom conflict handling function named `custom_conflict_dept` shown in Section 6.6.8.1. This function sets the target node as the winner of update/update conflicts on the `dept` table.

The following update is made on the master definition node, `edb`:

```
edb=# UPDATE dept SET loc = 'PORTLAND' WHERE deptno = 50;
UPDATE 1
edb=# SELECT * FROM dept;
 deptno |    dname     |   loc
--------+--------------+----------
     10 | ACCOUNTING   | NEW YORK
     20 | RESEARCH     | DALLAS
     30 | SALES        | CHICAGO
     40 | OPERATIONS   | BOSTON
     50 | ADVERTISING | PORTLAND
(5 rows)
```

The following update is made on a second master node, `mmrnode`:

```
mmrnode=# UPDATE dept SET loc = 'LOS ANGELES' WHERE deptno = 50;
UPDATE 1
mmrnode=# SELECT * FROM dept;
 deptno |    dname     |     loc
--------+--------------+-------------
     10 | ACCOUNTING   | NEW YORK
     20 | RESEARCH     | DALLAS
     30 | SALES        | CHICAGO
     40 | OPERATIONS   | BOSTON
     50 | ADVERTISING | LOS ANGELES
(5 rows)
```

After a synchronization replication, the update/update conflict is detected and resolved as shown in the Conflict History tab:

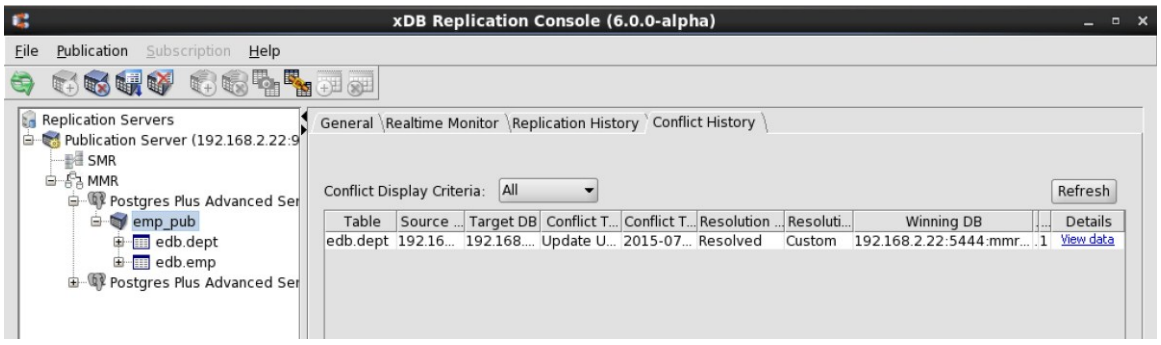


Figure 6-31 - Conflict resolved by custom conflict handling

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In the source master node the `loc` column of department `50` loses the value set in its `UPDATE` statement. The column is reset to the value from the target master node.

```
edb=# SELECT * FROM dept;
 deptno |    dname     |     loc
--------+--------------+-------------
     10 | ACCOUNTING   | NEW YORK
     20 | RESEARCH     | DALLAS
     30 | SALES        | CHICAGO
     40 | OPERATIONS   | BOSTON
     50 | ADVERTISING  | LOS ANGELES
(5 rows)
```

In the target master node the `loc` column of department `50` retains the value set from its `UPDATE` statement.

```
mmrnode=# SELECT * FROM dept;
 deptno |    dname     |     loc
--------+--------------+-------------
     10 | ACCOUNTING   | NEW YORK
     20 | RESEARCH     | DALLAS
     30 | SALES        | CHICAGO
     40 | OPERATIONS   | BOSTON
     50 | ADVERTISING  | LOS ANGELES
(5 rows)
```

The target node wins the conflict as determined by the setting of the *`resolution_code`* parameter to a value of `2` in the custom conflict handling function.

Setting Columns from the Function Logic

The following example shows the effect of custom conflict handling using the custom conflict handling function named `custom_conflict_emp` shown in Section 6.6.8.1. This function sets values coded in the function as the winner of update/update conflicts on the `emp` table.

The following is the row from the `emp` table prior to the update:

```
edb=# edb=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+-------+----------+------+---------------------+---------+---------+--------
  9001 | SMITH | SALESMAN | 7698 | 31-OCT-13 00:00:00 | 8000.00 | 4000.00 |     30
(1 row)
```

The following update is made in the master definition node, `edb`:

```
edb=# UPDATE emp SET ename = 'JONES', mgr = 7900, sal = 8500, comm = 5000 WHERE empno =
9001;
UPDATE 1
edb=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+-------+----------+------+---------------------+---------+---------+--------
  9001 | JONES | SALESMAN | 7900 | 31-OCT-13 00:00:00 | 8500.00 | 5000.00 |     30
(1 row)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following update is made in a second master node, `mmrnode`:

```
mmrnode=# UPDATE emp SET ename = 'ROGERS', mgr = 7788, sal = 9500, comm = 5000 WHERE
empno = 9001;
UPDATE 1
mmrnode=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename  |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+----------+------+---------------------+---------+---------+--------
  9001 | ROGERS | SALESMAN | 7788 | 31-OCT-13 00:00:00 | 9500.00 | 5000.00 |     30
(1 row)
```

After the synchronization replication the master node, `edb`, contains the following values for the conflicting row:

```
edb=# SELECT * FROM emp WHERE empno = 9001;
empno |  ename  |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+---------+----------+------+---------------------+---------+---------+--------
  9001 | Unknown | SALESMAN | 7900 | 31-OCT-31 00:00:00 | 8500.00 | 5000.00 |     30
(1 row)
```

After the synchronization replication the master node, `mmrnode`, contains the following values for the conflicting row:

```
mmrnode=# SELECT * FROM emp WHERE empno = 9001;
empno |  ename  |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+---------+----------+------+---------------------+---------+---------+--------
  9001 | Unknown | SALESMAN | 7900 | 31-OCT-31 00:00:00 | 8500.00 | 5000.00 |     30
(1 row)
```

The value of the first conflicting column is determined by the custom conflict handling function for both master nodes.

Setting Columns Using the Source and Target Shadow Tables

The following example shows how values from the source and target shadow tables can be used to set the final values in the conflicting column.

Note: As this custom conflict handling function uses a column (`rrep_old_quantity` in this example) that is a column of the shadow table and not of the actual publication table, this particular solution cannot be used for a publication using the log-based method of synchronization replication.

For this example, the following table is used, which contains product inventory.

```
CREATE TABLE inventory (
    item_id         NUMERIC PRIMARY KEY,
    name            VARCHAR(20),
    quantity        INTEGER
);
INSERT INTO inventory VALUES (1, 'LaserJet Printer 610', 50);
INSERT INTO inventory VALUES (2, 'Scanner 510', 10);
INSERT INTO inventory VALUES (3, 'LCD', 20);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When products are purchased at different locations, resulting in an inventory reduction on several master nodes, the remaining inventory must be properly updated on all master nodes to reflect the reduction in all locations. The custom conflict handling function is coded to properly record the remaining inventory if changes to the same item are made in several locations.

The following example uses master definition node, `edb`, and a second master node, `mmrnode`. Initially, the inventory table has the same contents on both master nodes.

```
edb=# SELECT * FROM inventory;
 item_id |          name          | quantity
---------+------------------------+----------
       1 | LaserJet Printer 610 |       50
       2 | Scanner 510          |       10
       3 | LCD                  |       20
(3 rows)
```

After creation of the master nodes, the following shows the resulting shadow table structures in the master definition node:

```
edb=# \d  edb_replicator_pub.rrst_edb_inventory;
                Table "_edb_replicator_pub.rrst_edb_inventory"
         Column          |            Type             |         Modifiers
-------------------------+-----------------------------+---------------------------
 rrep_sync_id            | numeric                     | not null
 rrep_common_id          | numeric                     |
 rrep_operation_type     | character(1)                |
 rrep_tx_timestamp       | timestamp without time zone | default current_timestamp
 item_id                 | numeric                     |
 name                    | character varying(20)       |
 quantity                | integer                     |
 rrep_old_item_id        | numeric                     |
 rrep_old_name           | character varying(20)       |
 rrep_old_quantity       | integer                     |
 rrep_tx_conflict_status | character(1)                |
Indexes:
    "rrst_edb_inventory_pkey" PRIMARY KEY, btree (rrep_sync_id)
```

Similarly, in the second master node the same shadow table is created.

```
mmrnode=# \d  edb_replicator_pub.rrst_edb_inventory
                Table "_edb_replicator_pub.rrst_edb_inventory"
         Column          |            Type             |         Modifiers
-------------------------+-----------------------------+---------------------------
 rrep_sync_id            | numeric                     | not null
 rrep_common_id          | numeric                     |
 rrep_operation_type     | character(1)                |
 rrep_tx_timestamp       | timestamp without time zone | default current_timestamp
 item_id                 | numeric                     |
 name                    | character varying(20)       |
 quantity                | integer                     |
 rrep_old_item_id        | numeric                     |
 rrep_old_name           | character varying(20)       |
 rrep_old_quantity       | integer                     |
 rrep_tx_conflict_status | character(1)                |
Indexes:
    "rrst_edb_inventory_pkey" PRIMARY KEY, btree (rrep_sync_id)
```

For an update transaction, the shadow table contains the column values before the update was made on the publication table (columns with names `rrep_old_`*`column_name`*) and

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

the values after the update was applied (columns named identically to the publication table column names).

The custom conflict handling function uses both the current and old values of the `quantity` columns from the source and target shadow tables as shown by the following.

```
CREATE OR REPLACE FUNCTION edb.custom_conflict_inventory (
    INOUT   source              _edb_replicator_pub.rrst_edb_inventory,
    IN      target              _edb_replicator_pub.rrst_edb_inventory,
    IN      conflict_column     VARCHAR(255),
    OUT     resolution_message  VARCHAR(255),
    OUT     resolution_code     INTEGER
)
AS
$$
DECLARE
BEGIN
    source.quantity := source.rrep_old_quantity
      - ((source.rrep_old_quantity - source.quantity)
       + (target.rrep_old_quantity - target.quantity));
    resolution_code := 3;
    resolution_message := 'Custom conflict handling: Quantity adjusted';
END;
$$
LANGUAGE plpgsql;
```

Assume two items with `item_id` of `1` are purchased on the master definition node:

```
edb=# UPDATE inventory SET quantity = quantity - 2 WHERE item_id = 1;
UPDATE 1
edb=# SELECT * FROM inventory WHERE item_id = 1;
 item_id |         name         | quantity
---------+----------------------+----------
       1 | LaserJet Printer 610 |       48
(1 row)
```

Also assume one item with `item_id` of `1` is purchased from the second master node:

```
mmrnode=# UPDATE inventory SET quantity = quantity - 1 WHERE item_id = 1;
UPDATE 1
mmrnode=# SELECT * FROM inventory WHERE item_id = 1;
 item_id |         name         | quantity
---------+----------------------+----------
       1 | LaserJet Printer 610 |       49
(1 row)
```

After the synchronization replication and invocation of the custom conflict handling function, the `quantity` column for `item_id 1` is correctly set to `47` in both master nodes:

```
edb=# SELECT * FROM inventory WHERE item_id = 1;
 item_id |         name         | quantity
---------+----------------------+----------
       1 | LaserJet Printer 610 |       47
(1 row)

edb=# \c mmrnode mmruser
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Password for user mmruser:
You are now connected to database "mmrnode" as user "mmruser".
mmrnode=# SET search_path TO edb;
SET
mmrnode=# SELECT * FROM inventory WHERE item_id = 1;
 item_id |         name         | quantity
---------+----------------------+----------
       1 | LaserJet Printer 610 |       47
(1 row)
```

6.6.9 Manual Conflict Resolution for the Trigger-Based Method

Note: The manual conflict resolution discussion in this section applies only to multi-master replication systems configured with the trigger-based method of synchronization replication. See Section 6.6.10 for information on manual conflict resolution for multi-master replication systems configured with the log-based method of synchronization replication.

As discussed in Section 6.6.5 there is no built-in, automatic conflict resolution strategy for the uniqueness (insert/insert) conflict. If a uniqueness conflict occurs, then you must modify rows in the publication tables containing the conflict as well as modify rows in the control schema tables in the master nodes to resolve the conflict.

Similarly, manual correction must be used for update/delete and delete/update conflicts. In addition, if the conflict resolution option is set to Manual (see Section 6.8) and a conflict occurs, this conflict must also be resolved in a manual fashion.

This section describes the updates you must make to the publication tables and the control schema tables in the master nodes.

This discussion is divided into the following topics:

- **Finding Conflicts.** Locating unresolved conflicts
- **Conflict Resolution Preparation.** Helpful setup steps to aid in the manual conflict resolution process
- **Overview of Correction Strategies.** Overview of the methods you can use to perform the corrections
- **Manual Publication Table Correction.** Manual correction of the publication tables
- **Correction Using New Transactions.** Using new transactions to bring all master nodes to a consistent state
- **Correction Using Shadow Table Transactions.** Using existing shadow table transactions to bring all master nodes to a consistent state

The following sections describe these topics in detail.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.6.9.1 Finding Conflicts

Conflicts can be found using the Conflict History tab as described in Section 6.7. The following is an example of the Conflict History tab. Click the Refresh button to reveal all of the latest conflicts.

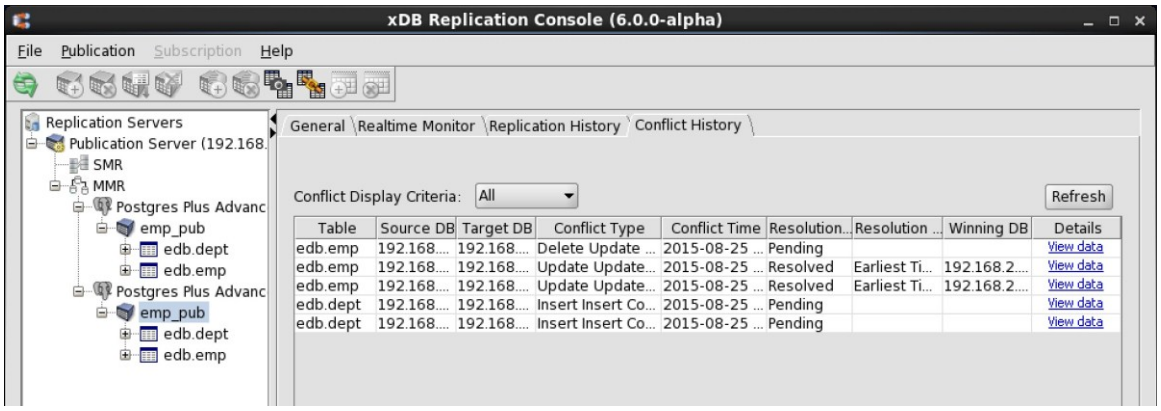


Figure 6-32 - Conflict History tab for trigger-based method

The Source DB and Target DB columns provide the IP address and database names of the source and target master nodes involved in the conflict.

Click the View Data link to show the details of the transactions involved in a particular conflict as shown by the following:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

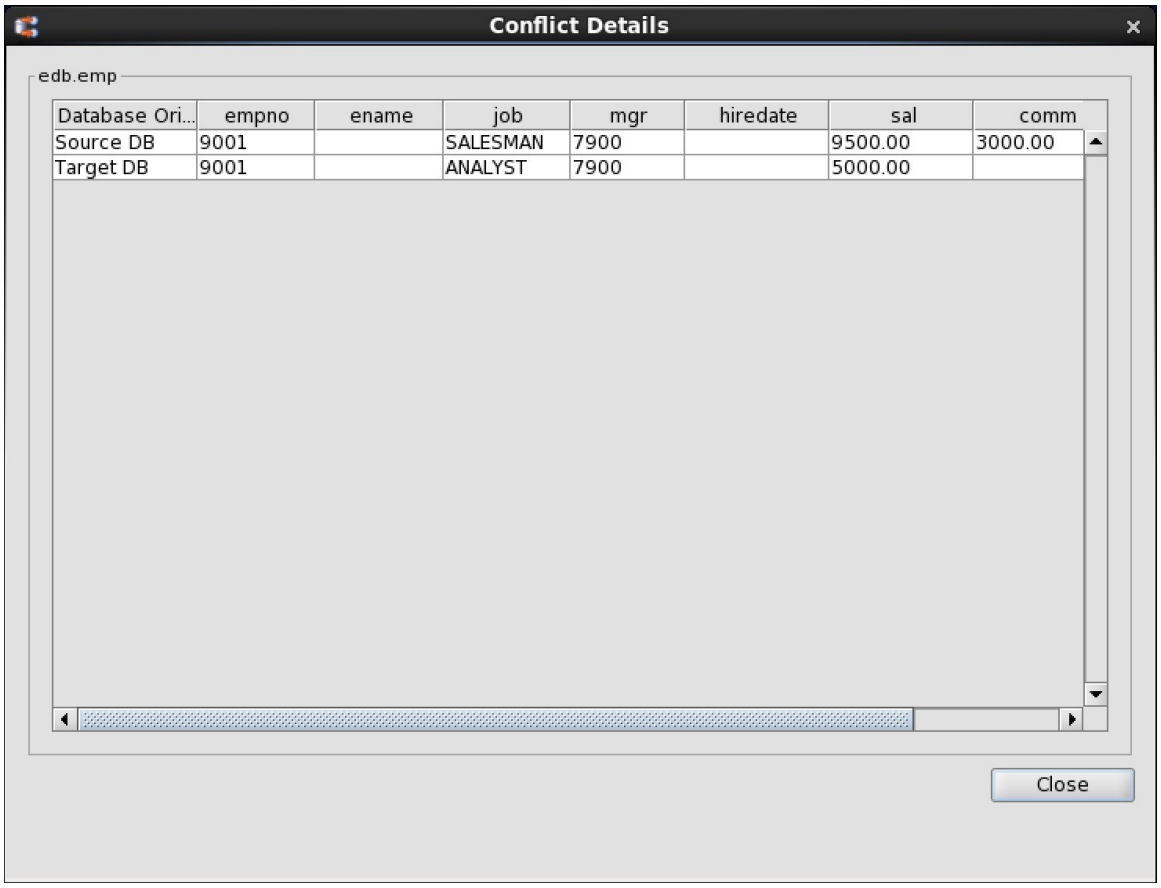


Figure 6-33 - Conflict Details window

You can also obtain this information from a SQL query rather than from the xDB Replication Console graphical user interface. The following query can be run from a master node to display information regarding pending (unresolved) conflicts:

```
SELECT DISTINCT
  conflict_type,
  table_name,
  pk_value,
  d1.db_host AS src_db_host,
  d1.db_port AS src_db_port,
  d1.db_name AS src_db_name,
  src_rrep_sync_id,
  d2.db_host AS target_db_host,
  d2.db_port AS target_db_port,
  d2.db_name AS target_db_name,
  target_rrep_sync_id,
  c.notes
FROM _edb_replicator_pub.xdb_conflicts c
  JOIN _edb_replicator_pub.xdb_pub_database d1 ON c.src_db_id = d1.pub_db_id
  JOIN _edb_replicator_pub.xdb_pub_database d2 ON c.target_db_id = d2.pub_db_id
  JOIN _edb_replicator_pub.rrep_tables t ON c.table_id = t.table_id
WHERE resolution_status = 'P'
ORDER BY t.table_name;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Example output from the query is shown by the following:

```
-[ RECORD 1 ]-------+-----------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | edb
src_rrep_sync_id    | 2
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
-[ RECORD 2 ]-------+-----------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode
src_rrep_sync_id    | 1
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | edb
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
-[ RECORD 3 ]-------+-----------------------------------------------------------
conflict_type       | DU
table_name          | emp
pk_value            |
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | edb
src_rrep_sync_id    | 5
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode
target_rrep_sync_id | 4
notes               |
```

6.6.9.2 Conflict Resolution Preparation

The following describes the setup of a database user with certain properties enabling you to modify the publication table rows for the purpose of manual conflict resolution.

Manual conflict resolution typically requires modification of rows in one or more publication tables to correct erroneous entries. Such changes can be done using a utility such as PSQL or pgAdmin (Postgres Enterprise Manager Client in Advanced Server).

Manual publication table corrections must usually be isolated – that is, these modifications must be limited to the publication tables you are directly changing and must not be replicated to the other master nodes as would normally occur in the multi-master replication system.

To prevent replication of changes to a publication table from occurring, the xDB Replication Server insert, update, and delete triggers on the publication tables must be

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

prevented from firing when you make these corrections to the publication table rows. If any of the insert, update, or delete triggers fire, an entry is added to the publication table's shadow table. This entry results in a transaction replicated to the other master nodes the next time synchronization replication occurs.

To prevent the triggers on the publication tables from firing, during the session in which you modify the publication table rows, the database server configuration parameter `session_replication_role` must be set to a value of `replica`. (The default setting of `session_replication_role` is `origin` in which case the triggers will fire.)

The suggested method to ensure the `replica` setting is in effect is to create a database user with a default session setting of `replica` for this parameter. Whenever you connect to a database with this database user, the `replica` setting will be in effect during this session.

Connect to a master node with this database user whenever you plan to make manual corrections to the publication tables in that particular node that are not to be replicated to the other master nodes.

In the following example database superuser `mmrmaint` is created and altered for this purpose:

```
mmrnode_a=# CREATE ROLE mmrmaint WITH LOGIN SUPERUSER PASSWORD 'password';
CREATE ROLE
mmrnode_a=# ALTER ROLE mmrmaint SET session_replication_role TO replica;
ALTER ROLE
```

When connected to a database with this database user, you can confirm this setting is in effect during the session by issuing the following command:

```
mmrnode_a=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)
```

6.6.9.3 Overview of Correction Strategies

Before you begin manual resolution correction, it is important to determine the extent of the inconsistencies that may have occurred in the publication tables across the master nodes of the replication system.

The Conflict History tab and the SQL query described in Section 6.6.9.1 can help determine the source of an initial conflict.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

However, once this conflict has occurred, your replication system may have processed and replicated additional transactions during that synchronization operation. Some of these subsequent replications may have succeeded as expected, but others may have failed or produced unexpected results as a consequence of the prior conflict.

If you have a replication schedule in effect, additional synchronization operations can occur, which may create additional conflicts.

Therefore, when you have discovered that a conflict has occurred, it is strongly recommended that you stop the publication server. Use the `stop` option of the Linux scripts or Windows services described in Step 1 of Section 5.2.1.

In this way, you can carefully analyze the content of the publication tables in question as well as any pending transactions in the shadow tables to determine the best course of action to take without the interference of continuing updates by a running replication system.

When analyzing your tables you must determine the following:

- Which publication tables contain inconsistent rows across master nodes (that is, missing rows on some master nodes, or rows with different column values for the same primary key on different master nodes).
- Which pending transactions in the shadow tables have not been applied to the publication tables across all master nodes. Pending transactions are denoted by a value of `P` in the `rrep_tx_conflict_status` column of the shadow table.
- Which transactions on the publication tables have occurred and are recorded in the shadow tables following the initial conflict, and whether or not these transactions have been applied completely and correctly to the publication tables across all master nodes. These transactions may not be marked as pending. Instead their `rrep_tx_conflict_status` column could be set to null meaning that no specific conflict was detected during replication, or the transaction has not yet been replicated. These transactions can be identified because they have a later `rrep_tx_timestamp` value than the transactions causing the initial conflict.

The general steps to resolving the problem following this analysis are the following:

Step 1: Make the necessary manual corrections to the rows in the publication tables across all master nodes to get them into an initial, consistent state so each publication table has the same set of identical rows across master nodes. This may be to a state before the conflicting transactions occurred, depending upon what you determine to be the easiest course of action for fully resolving the conflict.

Step 2: Apply or reapply transactions (either from your application or from the shadow tables) so that all publication tables across all master nodes are updated consistently according to the desired, expected result of what has been recorded in the shadow tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: In the shadow tables, update certain indicators for conflicting entries to show that these were resolved in Step 2.

Step 4: In the control schema, update certain indicators for the conflicting entries to show that these conflicts have been resolved. This update changes the Resolution Status of these entries to `Resolved` in the Conflict History tab. These entries will no longer appear in the SQL query described in Section 6.6.9.1.

Perform the Step 4 updates to the control schema of the controller database. The currently designated controller database can be determined from the content of the xDB Replication Configuration file (see Section 2.3.1.3). The publication server ensures that the control schema changes made on the controller database are replicated to the control schemas of all publication databases to maintain metadata consistency across all publication databases.

Step 5: Resume operation of your replication system. Start the publication server and recreate the replication schedule if you were using one.

For accomplishing steps 1 and 2, use some combination of the following methods. Which methods you use depend upon the state of your publication tables and the extent of pending transactions that need to be applied from the shadow tables.

- **Manual Publication Table Correction.** Use a utility such as PSQL or pgAdmin (Postgres Enterprise Manager Client in Advanced Server) to manually correct the rows in the publication tables across all master nodes without replicating these changes. Use the database user with `session_replication_role` set to `replica` for this purpose.
- **Correction Using New Transactions.** Rerun your application on one master node to create new transactions that you will allow to replicate to all other master nodes. Use this method after you have ensured that all publication tables are in a consistent state across all master nodes.
- **Correction Using Shadow Table Transactions.** Force the synchronization of transactions already recorded in the shadow tables. Use this method if there are many shadow table transactions that need to be applied, and it is simpler to force the synchronization of these transactions rather than reissuing the transactions from your application.

Each of these methods is described in more detail in the following sections.

For purposes of illustration, the following replication environment is used.

- A 3-node multi-master replication system has been established. The master node names are `mmrnode_a` (the master definition node and the controller database), `mmrnode_b`, and `mmrnode_c`.
- The publication is named `emp_pub` and uses the `dept` and `emp` tables that have been used as examples throughout this document.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The conflict used to illustrate the first two conflict resolution methods is a uniqueness conflict occurring on the `dept` table on primary key column `deptno` on value `50` resulting from the `INSERT` statements shown by the following:

On `mmrnode_a`, the following statement is run:

```
INSERT INTO dept VALUES (50, 'FINANCE', 'CHICAGO');
```

On `mmrnode_b`, the following statement is run:

```
INSERT INTO dept VALUES (50, 'MARKETING', 'LOS ANGELES');
```

A synchronization replication is then performed.

The following is the content of table `dept` on `mmrnode_a`:

```
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | FINANCE    | CHICAGO
(5 rows)
```

The following is the content of table `dept` on `mmrnode_b`:

```
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

The following is the content of table `dept` on `mmrnode_c`:

```
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

The Conflict History tab shows the following entry:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

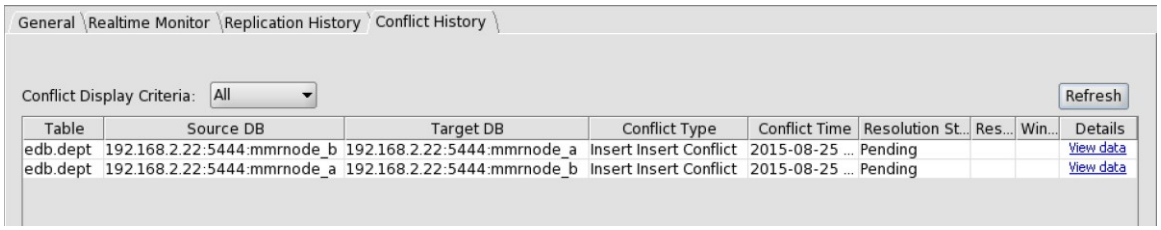


Figure 6-34 – Conflict History tab with a uniqueness conflict

The following is the output from the SQL query described in Section 6.6.9.1.

```
-[ RECORD 1 ]-------+------------------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode_a
src_rrep_sync_id    | 2
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode_b
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
-[ RECORD 2 ]-------+------------------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode_b
src_rrep_sync_id    | 1
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode_a
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
```

The following sections describe the application of different methods to resolve this conflict.

6.6.9.4 Manual Publication Table Correction

The first step required in all manual conflict resolutions is to ensure all publication tables are consistent across all master nodes – that is, all corresponding tables have the same rows with the same column values.

Once this state is achieved, you can then reapply transactions that may have failed to replicate successfully.

In the preceding example, the inconsistencies are the following:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Master nodes `mmrnode_a` and `mmrnode_b` each contain a row with primary key value `50`, but the other column values in the row are different.
- Master node `mmrnode_c` does not have a row with primary key value `50`.

Assuming that the correct state of the `dept` table should be the one in `mmrnode_b`, the following options are available to correct the state of all master nodes:

- Manually correct the `dept` table in `mmrnode_a` and `mmrnode_c`. That is, update the row in `mmrnode_a` so it has the correct values, and insert the missing row in `mmrnode_c`. The `dept` table on all nodes is now consistent and up-to-date.
- Manually delete the row with primary key value `50` from the table on both `mmrnode_a` and `mmrnode_b`. This brings the `dept` table on all master nodes back to a prior, consistent state. Then, with the multi-master replication system running, perform the insert transaction again using the correct column values on any one of the master nodes.
- Manually delete the incorrect row with primary key value `50` from the table on `mmrnode_a`. Leave the correct row in the table in `mmrnode_b`. This simulates the state where the correct transaction was run on `mmrnode_b`, is recorded in the shadow table, but has not yet been replicated, and the incorrect transaction was never run on `mmrnode_a`. Update the shadow table entry in `mmrnode_a` to indicate that it is discarded and to ensure it is not included in any future synchronizations. Update the metadata for the shadow table entry in `mmrnode_b` to force its inclusion in the next synchronization. Perform a synchronization replication so the accepted shadow table entry in `mmrnode_b` is replicated to `mmrnode_a` and `mmrnode_c`.

After the publication table rows are corrected, update the appropriate control schema table in the publication database currently designated as the controller database to indicate that the conflict has been resolved.

Each of the methods outlined in the preceding bullet points are described in more detail in the following sections. (The method described by the third bullet point is illustrated using a slightly more complex example on the `emp` table.)

The method outlined by the first bullet point is accomplished as follows.

Step 1: Manually correct the rows in the publication tables with `session_replication_role` set to `replica`.

On `mmrnode_a`, correct the erroneous row:

```
mmrnode_a=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | FINANCE    | CHICAGO
(5 rows)


mmrnode_a=# UPDATE dept SET dname = 'MARKETING', loc = 'LOS ANGELES' WHERE
deptno = 50;
UPDATE 1
mmrnode_a=# SELECT * FROM dept ORDER BY deptno;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_c`, insert the missing row:

```
mmrnode_c=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)


mmrnode_c=# INSERT INTO dept VALUES (50, 'MARKETING', 'LOS ANGELES');
INSERT 0 1
mmrnode_c=# SELECT * FROM dept ORDER BY deptno;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

The `dept` table on `mmrnode_a` and `mmrnode_c` now match the content of the table on `mmrnode_b`:

```
mmrnode_b=# SELECT * FROM dept ORDER BY deptno;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Update the shadow table entries for the conflicting transactions in the master nodes to indicate that the conflict has been resolved.

In each master node where a transaction occurred that is involved in the conflict, inspect the shadow table for the publication table in question.

Shadow tables are located in each master node in schema `_edb_replicator_pub`. Shadow tables follow the naming convention `rrst_schema_table` where *schema* is the name of the schema containing the publication table and *table* is the name of the publication table.

Note the following points regarding shadow tables:

- A row in a shadow table corresponds to an `INSERT`, `UPDATE`, or `DELETE` statement that is applied to the corresponding publication tables in the other master nodes. A shadow table row does not necessarily correspond to the SQL statement issued by the user application. For example, a SQL statement issued by a user application that includes a `WHERE` clause using a range such as greater than or less than, results in multiple, individual entries in the shadow table for each individual row in the result set of the application's SQL statement.
- The primary key of a shadow table is a program generated, positive integer in column `rrep_sync_id`. The `rrep_sync_id` values are unique amongst all shadow tables within a given master node. Therefore, the `rrep_sync_id` values for conflicting transactions may or may not have the same value across master nodes as this depends upon how many prior transactions were recorded in the shadow tables of each master node.
- A shadow table entry for a transaction involved in a conflict that has not yet been resolved contains a value of `P` (pending) in column `rrep_tx_conflict_status`. If a transaction is not involved in a conflict, this column is set to null. (The vast majority of shadow table entries should have null in this column.) If a transaction was involved in a conflict that was resolved automatically by the publication server, and this transaction was accepted as being correct, this column contains `C` (complete/accepted). If a transaction was involved in a conflict that was resolved automatically, and this transaction was deemed incorrect, this column contains `D` (discarded).

To find the shadow table entries involved in a conflict, use the Conflict History tab or SQL query as described in Section 6.6.9.1 and shown by the following output:

```
-[ RECORD 1 ]-------+-----------------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode_a
src_rrep_sync_id    | 2
target_db_host      | 192.168.2.22
target_db_port      | 5444
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
target_db_name      | mmrnode_b
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
-[ RECORD 2 ]-------+-----------------------------------------------------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode_b
src_rrep_sync_id    | 1
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode_a
target_rrep_sync_id | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
```

You can then query the shadow table in the desired master node on its `rrep_sync_id` value.

The following query is performed on the shadow table for the `dept` table in `mmrnode_a` on `rrep_sync_id` value `2` obtained from field `src_rrep_sync_id` of `RECORD 1` in the preceding output.

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.rrst_edb_dept WHERE
rrep_sync_id = 2;
-[ RECORD 1 ]----------+------------------------
rrep_sync_id           | 2
rrep_common_id         |
rrep_operation_type    | I
rrep_tx_timestamp      | 25-AUG-15 11:39:35.590648
deptno                 | 50
dname                  | FINANCE
loc                    | CHICAGO
rrep_old_deptno        |
rrep_old_dname         |
rrep_old_loc           |
rrep_tx_conflict_status | P
```

A similar query can locate the pending shadow table entry in `mmrnode_b` by querying on the key value obtained from field `src_rep_sync_id`: of `RECORD 2`:

```
mmrnode_b=# SELECT * FROM _edb_replicator_pub.rrst_edb_dept WHERE
rrep_sync_id = 1;
-[ RECORD 1 ]----------+------------------------
rrep_sync_id           | 1
rrep_common_id         |
rrep_operation_type    | I
rrep_tx_timestamp      | 25-AUG-15 11:39:57.980469
deptno                 | 50
dname                  | MARKETING
loc                    | LOS ANGELES
rrep_old_deptno        |
rrep_old_dname         |
rrep_old_loc           |
rrep_tx_conflict_status | P
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: To be certain no pending transactions are overlooked, you should examine the shadow tables in all master nodes that may have been involved in the conflict and search for entries where `rrep_tx_conflict_status` is set to `P`.

The following shows the `rrep_tx_conflict_status` column marked `P` (pending) in the Postgres Enterprise Manager Client.

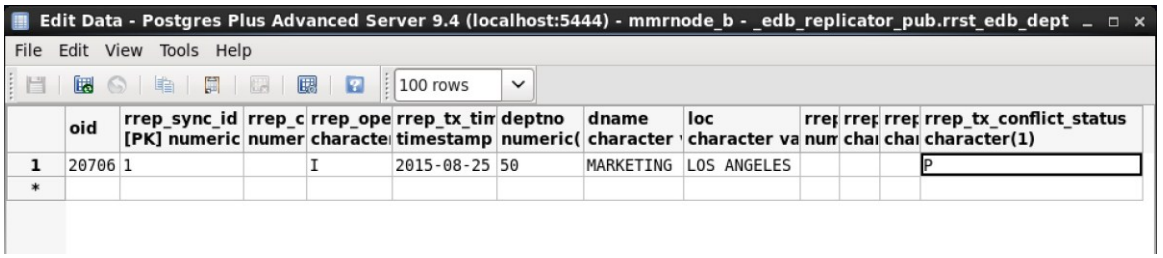


Figure 6-35 - Shadow table entry with pending conflict

Modify column `rrep_tx_conflict_status` by changing the value to `D` (discarded) to show that the pending conflict has been resolved. A value of `D` also ensures that the shadow table entry will not be replicated during any future synchronization replications.

Make this change to the shadow tables in both `mmrnode_a` and `mmrnode_b`.

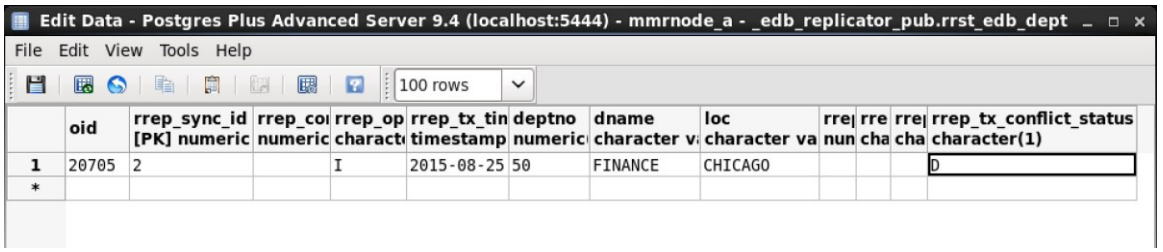


Figure 6-36 - Discarded shadow table entry

Be sure to qualify the row with the correct `rrep_sync_id` value if you perform the update using a SQL statement such as in the following:

```
UPDATE _edb_replicator_pub.rrst_edb_dept SET rrep_tx_conflict_status = 'D'
WHERE rrep_sync_id = 1;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

There is no shadow table entry in `mmrnode_c`, since an insert transaction was not performed in that master node by the application.

Step 3: In the control schema of the publication database currently designated as the controller database, modify the entries in the `xdb_conflicts` table to indicate the conflict has been resolved. Table `xdb_conflicts` is located in schema `_edb_replicator_pub`.

Note: The entries in table `xdb_conflicts` only affect the data that appears in the Conflict History tab and the SQL query described in Section 6.6.9.1. Changing entries in `xdb_conflicts` has no effect on future replication operations, but provides a way to keep a record of how past conflicts were resolved.

Note the following points regarding the `xdb_conflicts` table:

- A row in the `xdb_conflicts` table appears as an entry in the Conflict History tab.
- The primary key of the `xdb_conflicts` table is comprised of columns `src_db_id`, `target_db_id`, `src_rrep_sync_id`, and `target_rrep_sync_id`. Column `src_db_id` contains a unique identifier for the master node in which a transaction occurred that results in a conflict when replicated to the master node identified by `target_db_id`. `src_rrep_sync_id` is the shadow table identifier of the transaction on the source master node involved in the conflict while `target_rrep_sync_id` is the shadow table identifier of the transaction on the target master node that is involved in the conflict. **Note:** For uniqueness (insert/insert) conflicts, the `target_rrep_sync_id` value is always set to `0`. For a given uniqueness conflict, there are two entries in the `xdb_conflicts` table. The `src_rrep_sync_id` value in each of the two entries corresponds to the shadow table identifiers – one for the shadow table identifier associated with the source master node, the other for the shadow table identifier associated with the target master node.
- Table `xdb_pub_database` in the control schema associates the database identifiers `src_db_id` and `target_db_id` with the master node attributes such as the database name, IP address, and port.
- Column `table_id` is the identifier of the publication table on which the conflict occurred. Association of the `table_id` value with the publication table attributes such as its name, schema, and shadow table is found in each master node in `_edb_replicator_pub.rrep_tables`.
- For uniqueness (insert/insert) conflicts only, column `pk_value` contains text indicating the primary key value that resulted in the conflict. The text is formatted as *`column_name=value`*. If the primary key is composed of two or more columns, each column and value pair is separated by the keyword `AND` such as *`column_1=value_1`* `AND` *`column_2=value_2`*. This provides the primary key of the row in the publication table designated by `table_id` that resulted in the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

conflict. **Note:** Only uniqueness (insert/insert) conflicts contain the *column_name=value* text in the pk_value column. The pk_value column is null for all other conflict types (that is, update/update, delete/update, update/delete, and delete/delete conflicts).

- Column resolution_status indicates the status of the conflict. Possible values are P (pending) or C (completed – the conflict has been resolved). This status appears in the Resolution Status column of the Conflict History tab.
- Column win_db_id can be used to record the database identifier of the master node that contains the "winning" (accepted) transaction. This information appears in the Winning DB column of the Conflict History tab.
- Column win_rrep_sync_id can be used to record the shadow table identifier of the winning transaction.

The following shows the Conflict History tab prior to updating the xdb_conflicts table.

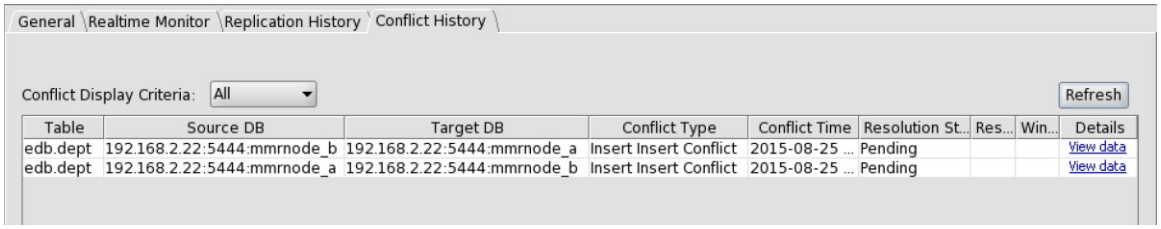


Figure 6-37 – Pending uniqueness conflict

The conflict entry for synchronization from mmrnode_a to mmrnode_b can be located in xdb_conflicts with the following query for this example:

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.xdb_conflicts
mmrnode_a-#   WHERE src_db_id = 1
mmrnode_a-#     AND target_db_id = 4
mmrnode_a-#     AND src_rrep_sync_id = 2
mmrnode_a-#     AND target_rrep_sync_id = 0;
-[ RECORD 1 ]-------+-----------------------------------------------------------------
src_db_id           | 1
target_db_id        | 4
src_rrep_sync_id    | 2
target_rrep_sync_id | 0
table_id            | 31
conflict_time       | 25-AUG-15 10:40:23.685738
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
pk_value            | deptno=50
```

The conflict entry for synchronization from `mmrnode_b` to `mmrnode_a` can be located in `xdb_conflicts` with the following query for this example:

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.xdb_conflicts
mmrnode_a-#   WHERE src_db_id = 4
mmrnode_a-#     AND target_db_id = 1
mmrnode_a-#     AND src_rrep_sync_id = 1
mmrnode_a-#     AND target_rrep_sync_id = 0;
-[ RECORD 1 ]-------+-----------------------------------------------------------------
src_db_id           | 4
target_db_id        | 1
src_rrep_sync_id    | 1
target_rrep_sync_id | 0
table_id            | 31
conflict_time       | 25-AUG-15 10:40:23.726889
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
pk_value            | deptno=50
```

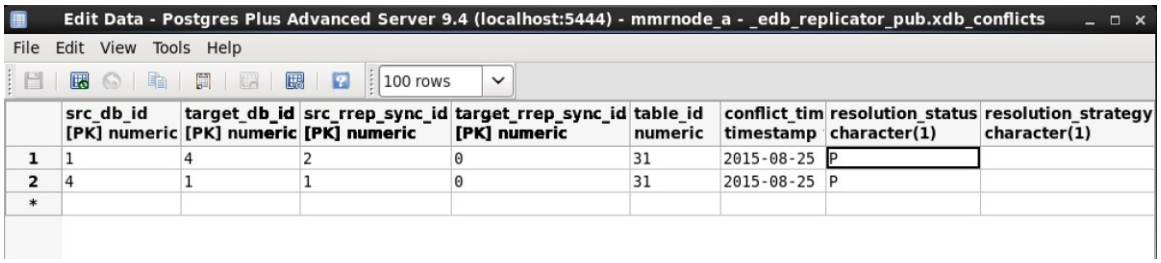
For uniqueness (insert/insert) conflicts only, the following query can be used to display both of the preceding entries:

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.xdb_conflicts
mmrnode_a-#   WHERE pk_value = 'deptno=50'
mmrnode_a-#     AND conflict_type = 'II'
mmrnode_a-#     AND resolution_status = 'P';
-[ RECORD 1 ]-------+-----------------------------------------------------------------
src_db_id           | 1
target_db_id        | 4
src_rrep_sync_id    | 2
target_rrep_sync_id | 0
table_id            | 31
conflict_time       | 25-AUG-15 10:40:23.685738
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
                    |   Detail: Key (deptno)=(50) already exists.
pk_value            | deptno=50
-[ RECORD 2 ]-------+-----------------------------------------------------------------
src_db_id           | 4
target_db_id        | 1
src_rrep_sync_id    | 1
target_rrep_sync_id | 0
table_id            | 31
conflict_time       | 25-AUG-15 10:40:23.726889
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "dept_pk"
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
                    |   Detail: Key (deptno)=(50) already exists.
pk_value            | deptno=50
```

These entries appear in the Postgres Enterprise Manager Client as shown by the following:



	src_db_id [PK] numeric	target_db_id [PK] numeric	src_rrep_sync_id [PK] numeric	target_rrep_sync_id [PK] numeric	table_id numeric	conflict_tim timestamp	resolution_status character(1)	resolution_strategy character(1)
1	1	4	2	0	31	2015-08-25	P	
2	4	1	1	0	31	2015-08-25	P	
*								

Figure 6-38 - Pending conflict in xdb_conflicts

Change the value in column `resolution_status` from `P` (pending) to `C` (completed) to indicate this conflict has been resolved. The value in `winning_db_id` is changed to `4` to indicate master node `mmrnode_b` contains the winning transaction. The value in `winning_rrep_sync_id` is changed to the value of `rrep_sync_id` for the shadow table entry of the transaction in `mmrnode_b` since this is the one deemed to be correct.

The SQL statement to perform this update for the `mmrnode_a` to the `mmrnode_b` synchronization conflict is the following:

```
UPDATE _edb_replicator_pub.xdb_conflicts SET
  resolution_status = 'C',
  win_db_id = 4,
  win_rrep_sync_id = 1
WHERE src_db_id = 1
  AND target_db_id = 4
  AND src_rrep_sync_id = 2
  AND target_rrep_sync_id = 0;
```

The SQL statement to perform this update for the `mmrnode_b` to the `mmrnode_a` synchronization conflict is the following:

```
UPDATE _edb_replicator_pub.xdb_conflicts SET
  resolution_status = 'C',
  win_db_id = 4,
  win_rrep_sync_id = 1
WHERE src_db_id = 4
  AND target_db_id = 1
  AND src_rrep_sync_id = 1
  AND target_rrep_sync_id = 0;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For uniqueness (insert/insert) conflicts only, the following SQL statement can be used to update both of the preceding entries simultaneously:

```
UPDATE _edb_replicator_pub.xdb_conflicts SET
  resolution_status = 'C',
  win_db_id = 4,
  win_rrep_sync_id = 1
WHERE pk_value = 'deptno=50'
  AND conflict_type = 'II'
  AND resolution_status = 'P';
```

The following are the updated `xdb_conflicts` entries:

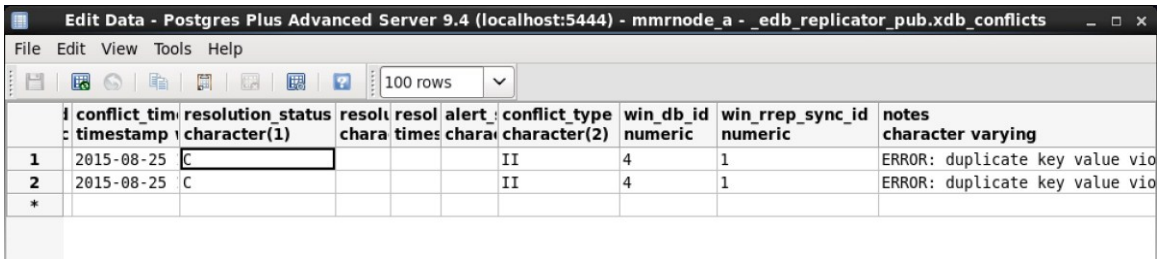


Figure 6-39 - Resolved conflict in xdb_conflicts

When viewed in the Conflict History tab, the entries now show `Resolved` instead of `Pending` in the Resolution Status column, and the Winning DB column shows the address of master node `mmrnode_b`.

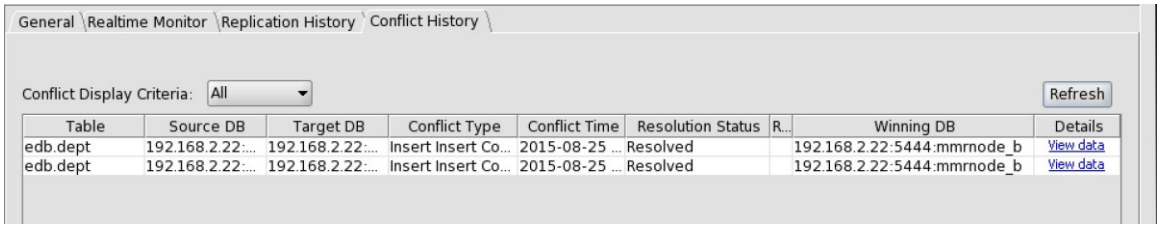


Figure 6-40 - Resolved uniqueness conflict

6.6.9.5 Correction Using New Transactions

Another method for bringing all the publication tables to a consistent state is by removing any changes caused by the conflicting transactions and then issuing new, corrected

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

transactions at one master node, which you allow the multi-master replication system to synchronize to all other master nodes.

Referring back to the uniqueness conflict on the `dept` table, instead of correcting the erroneous row and inserting the row into the master node where it is missing as described in Section 6.6.9.4, you can delete the conflicting rows from all master nodes, then insert the correct row in one master node and let the multi-master replication system synchronize the correct row to all master nodes.

Step 1: Manually delete the inserted row from the publication tables in all master nodes with `session_replication_role` set to `replica`.

On `mmrnode_a`, delete the erroneous row:

```
mmrnode_a=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)


mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | FINANCE    | CHICAGO
(5 rows)


mmrnode_a=# DELETE FROM dept WHERE deptno = 50;
DELETE 1
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

On `mmrnode_b`, delete the row even though the transaction created the correct result:

```
mmrnode_b=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)


mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)


mmrnode_b=# DELETE FROM dept WHERE deptno = 50;
DELETE 1
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

On `mmrnode_c`, no changes are required as the conflicting transaction did not insert a new row into the table on this node:

```
mmrnode_c=# SET search_path TO edb;
SET
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

Step 2: Rerun the transaction on one master node with the multi-master replication system running and with `session_replication_role` set to the default (`origin`).

For this example, the correct `INSERT` statement is executed on `mmrnode_a`:

On `mmrnode_a`:

```
mmrnode_a=# SHOW session_replication_role;
 session_replication_role
--------------------------
 origin
(1 row)


mmrnode_a=# INSERT INTO dept VALUES (50, 'MARKETING', 'LOS ANGELES');
INSERT 0 1
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

Step 3: Perform synchronization replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The same rows now appear in the publication table on all master nodes.

On `mmrnode_a`;

```
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_b`:

```
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_c`:

```
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

Step 4: Update the shadow table entries for the conflicting transactions in the master nodes to indicate that the conflict has been resolved as in Step 2 of Section 6.6.9.4.

Change the `rrep_tx_conflict_status` column from `P` (pending) to `D` (discarded) on all master nodes.

The resulting change for the shadow table on `mmrnode_a` is as follows.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

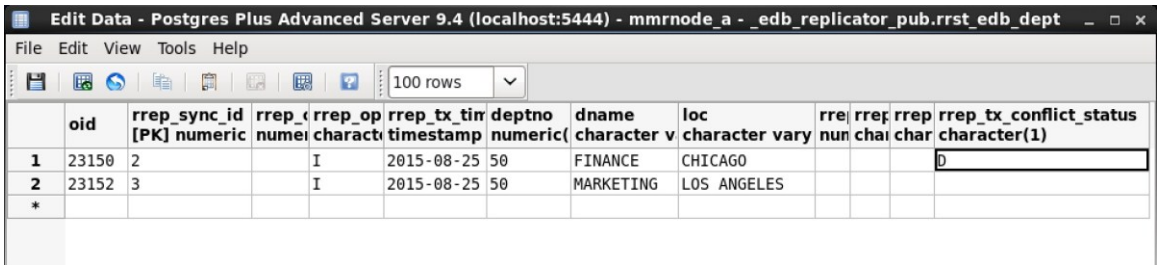


Figure 6-41 - Discarded shadow table entry

Note the second entry for the accepted transaction you ran in Step 2 where `rrep_tx_conflict_status` is set to null indicating there was no conflict.

The resulting change for the shadow table on `mmrnode_b` is as follows.

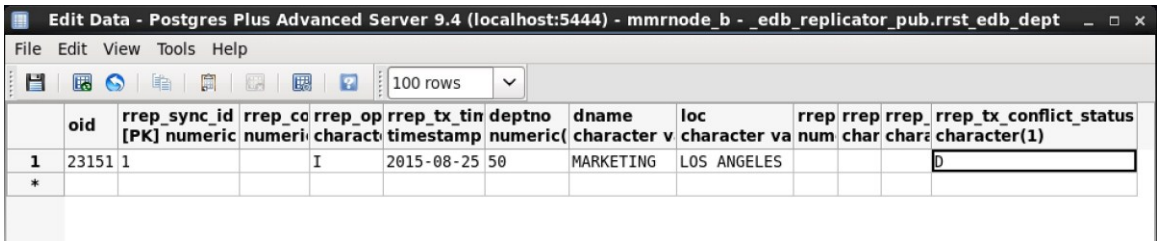


Figure 6-42 - Discarded shadow table entry

There is no shadow table entry in `mmrnode_c`, since an insert transaction was not performed in that master node by the application.

Step 5: In the control schema of the publication database currently designated as the controller database, modify the entries in the `xdb_conflicts` table to indicate the conflict has been resolved as in Step 3 of Section 6.6.9.4.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

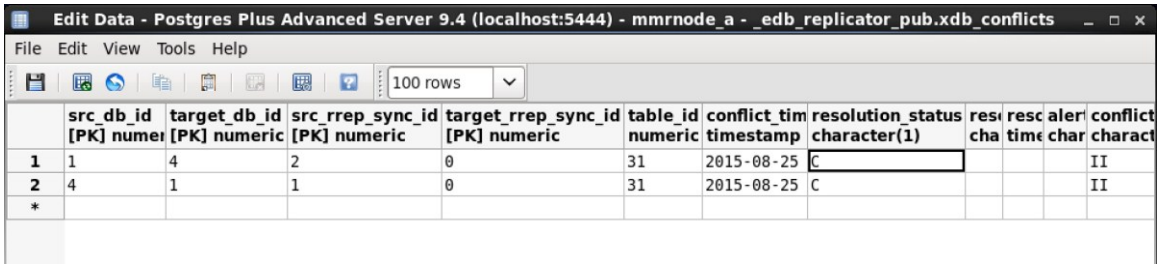


Figure 6-43 - Resolved conflict in xdb_conflicts

6.6.9.6 Correction Using Shadow Table Transactions

The final method for bringing all publication tables to a consistent state is by removing changes caused by the conflicting transactions and then modifying the publication table's metadata in such a way that the next synchronization results in the replication of transactions already stored in the shadow tables. Such transactions may not have been successfully replicated to all the other master nodes in a prior synchronization for various reasons.

The following is an example of such a case:

- Applications on two master nodes insert rows with the same primary key value. This will result in a uniqueness conflict when synchronization replication occurs.
- Following the insert on one master node, the application continues to apply updates to the newly inserted row. These updates are successfully applied to the row on this master node and are recorded in the shadow table on this node.
- Synchronization replication is performed.
- Since there is a uniqueness conflict, the rows with the conflicting primary key value are not replicated into the publication tables on the other master nodes.
- However, the conflicting row on the master node that was not directly updated will receive those update transactions by the replication, resulting in possibly inconsistent, updated rows on the two master nodes.

Instead of manually inserting the missing row into the other master nodes and manually changing the incorrect row; or instead of rerunning the application to reapply the correct insert and updates, the following option provides a way to reapply the transactions already recorded in the shadow table of the winning master node.

The example used to illustrate this method is based upon the following transactions on the `emp` table.

In `mmrnode_b`, the following row is inserted:

```
INSERT INTO emp (empno,ename,job,deptno) VALUES (9001,'SMITH','ANALYST',20);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In mmrnode_c, the following row is inserted with the same primary key value 9001 in the empno column:

```
INSERT INTO emp (empno,ename,job,deptno) VALUES (9001,'JONES','SALESMAN',30);
```

In mmrnode_c, this is followed by a series of updates to the newly inserted row:

```
UPDATE emp SET mgr = 7698 WHERE empno = 9001;
UPDATE emp SET sal = 9500 WHERE empno = 9001;
UPDATE emp SET comm = 5000 WHERE empno = 9001;
```

Synchronization replication is performed. The resulting content of the emp table is as follows:

On mmrnode_a the conflicting row has not been replicated:

```
mmrnode_a=# SELECT * FROM emp;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+-----------+------+---------------------+---------+---------+--------
  7369 | SMITH  | CLERK     | 7902 | 17-DEC-80 00:00:00 |  800.00 |         |     20
  7499 | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 00:00:00 | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN  | 7698 | 22-FEB-81 00:00:00 | 1250.00 |  500.00 |     30
  7566 | JONES  | MANAGER   | 7839 | 02-APR-81 00:00:00 | 2975.00 |         |     20
  7654 | MARTIN | SALESMAN  | 7698 | 28-SEP-81 00:00:00 | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER   | 7839 | 01-MAY-81 00:00:00 | 2850.00 |         |     30
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00 | 2450.00 |         |     10
  7788 | SCOTT  | ANALYST   | 7566 | 19-APR-87 00:00:00 | 3000.00 |         |     20
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00 | 5000.00 |         |     10
  7844 | TURNER | SALESMAN  | 7698 | 08-SEP-81 00:00:00 | 1500.00 |    0.00 |     30
  7876 | ADAMS  | CLERK     | 7788 | 23-MAY-87 00:00:00 | 1100.00 |         |     20
  7900 | JAMES  | CLERK     | 7698 | 03-DEC-81 00:00:00 |  950.00 |         |     30
  7902 | FORD   | ANALYST   | 7566 | 03-DEC-81 00:00:00 | 3000.00 |         |     20
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00 | 1300.00 |         |     10
(14 rows)
```

On mmrnode_b the conflicting row inserted on this node remains, but is updated with the transactions replicated from mmrnode_c:

```
mmrnode_b=# SELECT * FROM emp;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+-----------+------+---------------------+---------+---------+--------
  7369 | SMITH  | CLERK     | 7902 | 17-DEC-80 00:00:00 |  800.00 |         |     20
  7499 | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 00:00:00 | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN  | 7698 | 22-FEB-81 00:00:00 | 1250.00 |  500.00 |     30
  7566 | JONES  | MANAGER   | 7839 | 02-APR-81 00:00:00 | 2975.00 |         |     20
  7654 | MARTIN | SALESMAN  | 7698 | 28-SEP-81 00:00:00 | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER   | 7839 | 01-MAY-81 00:00:00 | 2850.00 |         |     30
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00 | 2450.00 |         |     10
  7788 | SCOTT  | ANALYST   | 7566 | 19-APR-87 00:00:00 | 3000.00 |         |     20
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00 | 5000.00 |         |     10
  7844 | TURNER | SALESMAN  | 7698 | 08-SEP-81 00:00:00 | 1500.00 |    0.00 |     30
  7876 | ADAMS  | CLERK     | 7788 | 23-MAY-87 00:00:00 | 1100.00 |         |     20
  7900 | JAMES  | CLERK     | 7698 | 03-DEC-81 00:00:00 |  950.00 |         |     30
  7902 | FORD   | ANALYST   | 7566 | 03-DEC-81 00:00:00 | 3000.00 |         |     20
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00 | 1300.00 |         |     10
  9001 | SMITH  | ANALYST   | 7698 |                     | 9500.00 | 5000.00 |     20
(15 rows)
```

On mmrnode_c the conflicting row inserted on this node remains along with the updates performed on this node:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
mmrnode_c=# SELECT * FROM emp;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+-----------+------+---------------------+---------+---------+--------
  7369 | SMITH  | CLERK     | 7902 | 17-DEC-80 00:00:00  |  800.00 |         |     20
  7499 | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 00:00:00  | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN  | 7698 | 22-FEB-81 00:00:00  | 1250.00 |  500.00 |     30
  7566 | JONES  | MANAGER   | 7839 | 02-APR-81 00:00:00  | 2975.00 |         |     20
  7654 | MARTIN | SALESMAN  | 7698 | 28-SEP-81 00:00:00  | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER   | 7839 | 01-MAY-81 00:00:00  | 2850.00 |         |     30
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00  | 2450.00 |         |     10
  7788 | SCOTT  | ANALYST   | 7566 | 19-APR-87 00:00:00  | 3000.00 |         |     20
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00  | 5000.00 |         |     10
  7844 | TURNER | SALESMAN  | 7698 | 08-SEP-81 00:00:00  | 1500.00 |    0.00 |     30
  7876 | ADAMS  | CLERK     | 7788 | 23-MAY-87 00:00:00  | 1100.00 |         |     20
  7900 | JAMES  | CLERK     | 7698 | 03-DEC-81 00:00:00  |  950.00 |         |     30
  7902 | FORD   | ANALYST   | 7566 | 03-DEC-81 00:00:00  | 3000.00 |         |     20
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00  | 1300.00 |         |     10
  9001 | JONES  | SALESMAN  | 7698 |                     | 9500.00 | 5000.00 |     30
(15 rows)
```

In this example, it is assumed that the desired, correct row is on `mmrnode_c`.

The following are the steps to reproduce the correct row, currently on `mmrnode_c`, to the other master nodes by synchronizing the shadow table entries that resulted from the original insert and updates to this row on `mmrnode_c`.

Step 1: Manually delete the inserted row from the publication tables on all master nodes except for `mmrnode_c`, which has the correct row. Be sure `session_replication_role` is set to `replica`.

On `mmrnode_a`, this row does not exist:

```
mmrnode_a=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename | job | mgr | hiredate | sal | comm | deptno
-------+-------+-----+-----+----------+-----+------+--------
(0 rows)
```

On `mmrnode_b`, delete the erroneous row:

```
mmrnode_a=# SHOW session_replication_role;
 session_replication_role
--------------------------
 replica
(1 row)

mmrnode_b=# DELETE FROM emp WHERE empno = 9001;
DELETE 1
```

On `mmrnode_c`, the correct, accepted row is left intact:

```
mmrnode_c=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  | hiredate |   sal   |  comm   | deptno
-------+-------+----------+------+----------+---------+---------+--------
  9001 | JONES | SALESMAN | 7698 |          | 9500.00 | 5000.00 |     30
(1 row)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: On the master nodes containing the conflicting row that is to be discarded, mark the shadow table entry for that row as discarded. This indicates the conflict on this row has been resolved and ensures this shadow table entry is not replicated in the future.

Change the `rrep_tx_conflict_status` column from `P` (pending) to `D` (discarded) on the losing node, `mmrnode_b` as shown by the following:

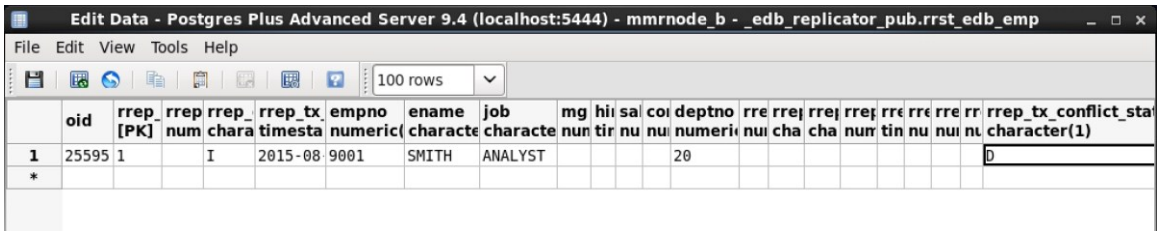


Figure 6-44 - Losing shadow table entry

Step 3: On winning node `mmrnode_c`, inspect the shadow table for the `emp` publication table.

The objective is to use the shadow table entries for the insert and three update transactions that were previously run on this node to replicate to the other master nodes during the next synchronization.

The leftmost columns of the shadow table appear as follows:

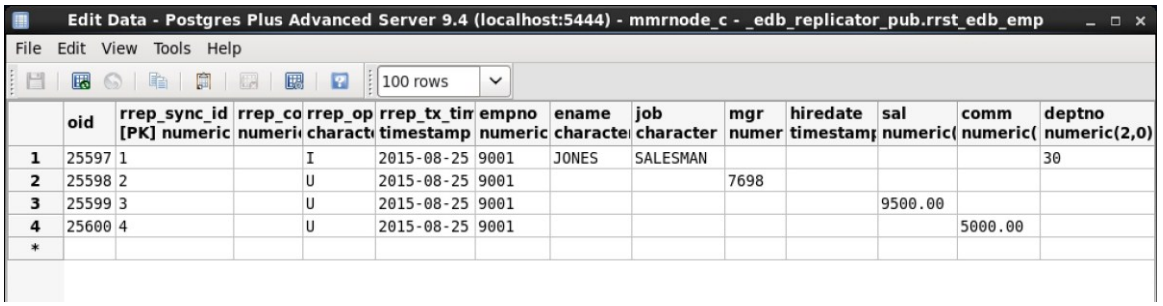


Figure 6-45 - Shadow table with multiple transactions

Make note of the `rrep_sync_id` values for these four entries, which are `1`, `2`, `3`, and `4` in this example.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows the rightmost columns of the shadow table from the prior figure. Note the contents of column `rrep_tx_conflict_status` furthest to the right.

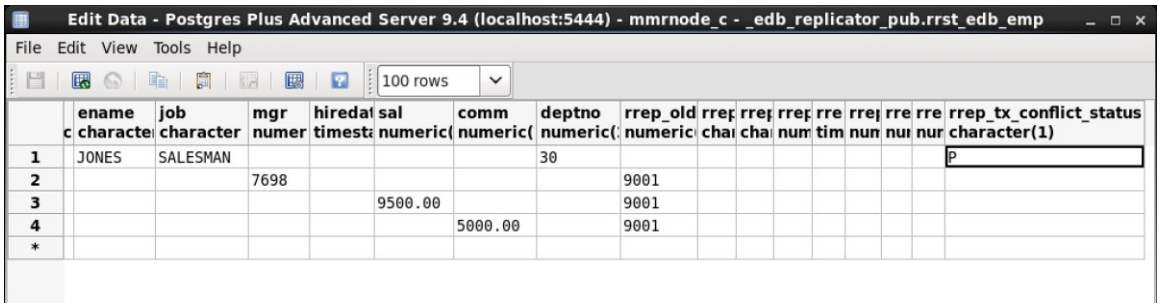


Figure 6-46 - Shadow table with multiple transactions (continued)

Make sure the `rrep_tx_conflict_status` column is null for these four entries. In this case, for the insert transaction, you will need to change the `P` (pending) value to null.

The resulting change for the `rrep_tx_conflict_status` column in the shadow table on `mmrnode_c` is shown by the following:

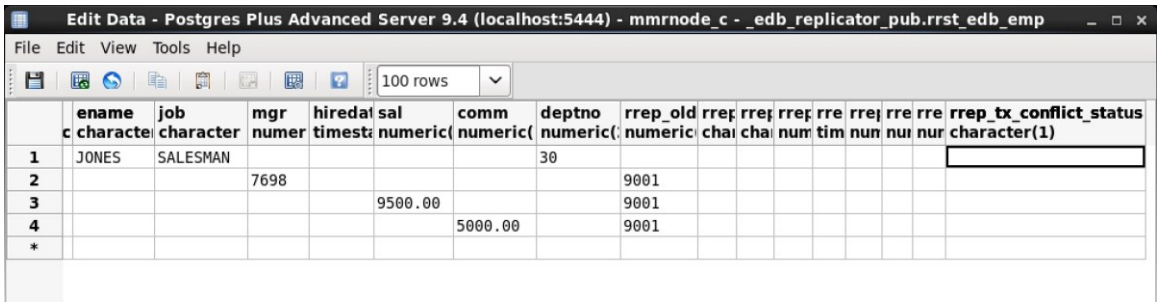


Figure 6-47 - Shadow table transactions set to replicate

Step 4: In order to replicate these four shadow table entries during the next synchronization, one or more entries must be added to the control schema table `_edb_replicator_pub.rrep_mmr_txset` on `mmrnode_c` to indicate pending status for synchronization to the target master nodes (`mmrnode_a` and `mmrnode_b`) of the four shadow table entries identified by the `rrep_sync_id` values of `1`, `2`, `3`, and `4` noted in Step 3.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

First, you must identify the `pub_id` and target `db_id` values that are to be associated with the pending transactions. To do so, invoke the following query substituting the `rrep_sync_id` values for *`sync_id`* in the query:

```
SELECT pub_id, db_id AS target_db_id
  FROM _edb_replicator_pub.rrep_mmr_txset
  WHERE start_rrep_sync_id <= sync_id
    AND end_rrep_sync_id >= sync_id;
```

In this example, there are four values to be substituted for *`sync_id`*, which are `1`, `2`, `3`, and `4`.

The results are the following:

```
mmrnode_c=# SELECT pub_id, db_id AS target_db_id
mmrnode_c-#   FROM _edb_replicator_pub.rrep_mmr_txset
mmrnode_c-#   WHERE start_rrep_sync_id <= 1 AND end_rrep_sync_id >= 1;
 pub_id | target_db_id
--------+--------------
      3 |            1
      3 |            4
(2 rows)

mmrnode_c=# SELECT pub_id, db_id AS target_db_id
mmrnode_c-#   FROM _edb_replicator_pub.rrep_mmr_txset
mmrnode_c-#   WHERE start_rrep_sync_id <= 2 AND end_rrep_sync_id >= 2;
 pub_id | target_db_id
--------+--------------
      3 |            1
      3 |            4
(2 rows)

mmrnode_c=# SELECT pub_id, db_id AS target_db_id
mmrnode_c-#   FROM _edb_replicator_pub.rrep_mmr_txset
mmrnode_c-#   WHERE start_rrep_sync_id <= 3 AND end_rrep_sync_id >= 3;
 pub_id | target_db_id
--------+--------------
      3 |            1
      3 |            4
(2 rows)

mmrnode_c=# SELECT pub_id, db_id AS target_db_id
mmrnode_c-#   FROM _edb_replicator_pub.rrep_mmr_txset
mmrnode_c-#   WHERE start_rrep_sync_id <= 4 AND end_rrep_sync_id >= 4;
 pub_id | target_db_id
--------+--------------
      3 |            1
      3 |            4
(2 rows)
```

The results indicate that the previously executed synchronization that attempted to apply the shadow table transactions identified by the `rrep_sync_id` values of `1`, `2`, `3`, and `4` were all for the publication identified by `pub_id` of `3`. The target master nodes were identified by `db_id` of `1` (for `mmrnode_a`) and `db_id` of `4` (for `mmrnode_b`).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Thus, at least two entries must be inserted into the control schema table `_edb_replicator_pub.rrep_mmr_txset` on `mmrnode_c`. At least one entry is required for the target `db_id` of `1` and at least one entry for the target `db_id` of `4`.

Since each entry in `_edb_replicator_pub.rrep_mmr_txset` consists of a range of `rrep_sync_id` values (identified by columns `start_rrep_sync_id` and `end_rrep_sync_id`) and the desired shadow table `rrep_sync_id` values happen to be contiguous (`1` thru `4`), a single entry can encompass the four `rrep_sync_id` values for a single target database.

Thus, in this example, a total of two entries can be added to `_edb_replicator_pub.rrep_mmr_txset` – one for each target database.

Note: If there were multiple, non-contiguous `rrep_sync_id` values required for synchronization (for example, `1`, `2`, `5`, and `6`), then multiple entries would be required for each target database. The entries would specify `rrep_sync_id` ranges to collectively cover all of the non-contiguous values, but omitting `rrep_sync_id` values that are not to be included in the synchronization (for example, one entry for `1` through `2` and a second entry for `5` through `6`).

Step 5: Insert the entries into the `_edb_replicator_pub.rrep_mmr_txset` control schema table as identified in the preceding step.

The two `INSERT` statements invoked on `mmrnode_c` are the following:

```
INSERT INTO _edb_replicator_pub.rrep_mmr_txset (set_id, pub_id, db_id,
status, start_rrep_sync_id, end_rrep_sync_id)
  values (nextval('_edb_replicator_pub.rrep_txset_seq'),3,1,'P',1,4);

INSERT INTO _edb_replicator_pub.rrep_mmr_txset (set_id, pub_id, db_id,
status, start_rrep_sync_id, end_rrep_sync_id)
  values (nextval('_edb_replicator_pub.rrep_txset_seq'),3,4,'P',1,4);
```

A query of the `_edb_replicator_pub.rrep_mmr_txset` metadata table displays the following:

```
mmrnode_c=# SELECT set_id, pub_id, db_id AS target_db_id, status,
mmrnode_c-#     start_rrep_sync_id, end_rrep_sync_id
mmrnode_c-#   FROM _edb_replicator_pub.rrep_mmr_txset;
 set_id | pub_id | target_db_id | status | start_rrep_sync_id | end_rrep_sync_id
--------+--------+--------------+--------+--------------------+------------------
      1 |      3 |            1 | C      |                  1 |                4
      1 |      3 |            4 | C      |                  1 |                4
      2 |      3 |            1 | P      |                  1 |                4
      3 |      3 |            4 | P      |                  1 |                4
(4 rows)
```

There are now two new entries with pending status (`P`), one for target `db_id` `1`, the other for target `db_id` `4`. Both entries cover the `rrep_sync_id` range of `1` through `4`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The two entries with completed status (C) are from the synchronization attempt that initially produced the conflict.

Step 6: Perform synchronization replication.

The insert and three update transactions recorded in the `rrst_edb_emp` shadow table on `mmrnode_c` are replicated to the other master nodes.

On `mmrnode_a`:

```
mmrnode_a=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  | hiredate |   sal   |  comm   | deptno
-------+-------+----------+------+----------+---------+---------+--------
  9001 | JONES | SALESMAN | 7698 |          | 9500.00 | 5000.00 |     30
(1 row)
```

On `mmrnode_b`:

```
mmrnode_b=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  | hiredate |   sal   |  comm   | deptno
-------+-------+----------+------+----------+---------+---------+--------
  9001 | JONES | SALESMAN | 7698 |          | 9500.00 | 5000.00 |     30
(1 row)
```

These rows now match the row created by the original transactions on `mmrnode_c`:

```
mmrnode_c=# SELECT * FROM emp WHERE empno = 9001;
 empno | ename |   job    | mgr  | hiredate |   sal   |  comm   | deptno
-------+-------+----------+------+----------+---------+---------+--------
  9001 | JONES | SALESMAN | 7698 |          | 9500.00 | 5000.00 |     30
(1 row)
```

Step 7: In the control schema of the publication database currently designated as the controller database, modify the entries in the `xdb_conflicts` table to indicate the conflict has been resolved as in Step 3 of Section 6.6.9.4.

For a uniqueness (insert/insert) conflict only, the following query on the `xdb_conflicts` table in the controller database can display the conflicts:

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.xdb_conflicts
mmrnode_a-#   WHERE pk_value = 'empno=9001'
mmrnode_a-#     AND conflict_type = 'II'
mmrnode_a-#     AND resolution_status = 'P';
-[ RECORD 1 ]-------+------------------------------------------------------------------
src_db_id           | 4
target_db_id        | 56
src_rrep_sync_id    | 1
target_rrep_sync_id | 0
table_id            | 32
conflict_time       | 25-AUG-15 15:27:20.928679
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "emp_pk"
                    |   Detail: Key (empno)=(9001) already exists.
pk_value            | empno=9001
-[ RECORD 2 ]-------+-----------------------------------------------------------
src_db_id           | 56
target_db_id        | 4
src_rrep_sync_id    | 1
target_rrep_sync_id | 0
table_id            | 32
conflict_time       | 25-AUG-15 15:27:20.970959
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               | ERROR: duplicate key value violates unique constraint "emp_pk"
                    |   Detail: Key (empno)=(9001) already exists.
pk_value            | empno=9001
```

The following SQL statement changes the value in column `resolution_status` from `P` (pending) to `C` (completed) to indicate this conflict has been resolved. The value in `winning_db_id` is changed to `56` to indicate master node `mmrnode_c` contains the winning transaction. The value in `winning_rrep_sync_id` is changed to the value of `rrep_sync_id` for the shadow table entry of the `INSERT` transaction in `mmrnode_c` since this is the one deemed to be correct.

```
UPDATE _edb_replicator_pub.xdb_conflicts SET
  resolution_status = 'C',
  win_db_id = 56,
  win_rrep_sync_id = 1
WHERE pk_value = 'empno=9001'
  AND conflict_type = 'II'
  AND resolution_status = 'P';
```

When viewed in the Conflict History tab, the entry now shows `Resolved` in the Resolution Status column, and the Winning DB column shows the address of master node `mmrnode_c`.

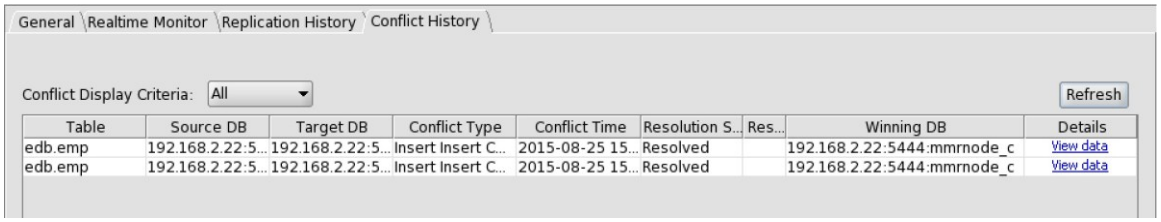


Figure 6-48 - Resolved uniqueness conflict

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

6.6.10 Manual Conflict Resolution for the Log-Based Method

Note: The manual conflict resolution discussion in this section applies only to multi-master replication systems configured with the log-based method of synchronization replication. See Section 6.6.9 for information on manual conflict resolution for multi-master replication systems configured with the trigger-based method of synchronization replication.

As discussed in Section 6.6.5 there is no built-in, automatic conflict resolution strategy for the uniqueness (insert/insert) conflict. If a uniqueness conflict occurs, then you must modify rows in the publication tables containing the conflict as well as modify rows in the control schema tables in the master nodes to resolve the conflict.

Similarly, manual correction must be used for update/delete and delete/update conflicts. In addition, if the conflict resolution option is set to Manual (see Section 6.8) and a conflict occurs, this conflict must also be resolved in a manual fashion.

This section describes the updates you must make to the publication tables and the control schema tables in the master nodes.

This discussion is divided into the following topics:

- **Finding Conflicts.** Locating unresolved conflicts
- **Conflict Resolution Concept for the Log-Based Method.** Basic concept on how to run transactions to apply corrections
- **Overview of Correction Strategies.** Overview of the methods you can use to perform the corrections
- **Manual Publication Table Correction.** Manual correction of the publication tables
- **Correction Using New Transactions.** Using new transactions to bring all master nodes to a consistent state

The following sections describe these topics in detail.

6.6.10.1 Finding Conflicts

Conflicts can be found using the Conflict History tab as described in Section 6.7. The following is an example of the Conflict History tab. Click the Refresh button to reveal all of the latest conflicts.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

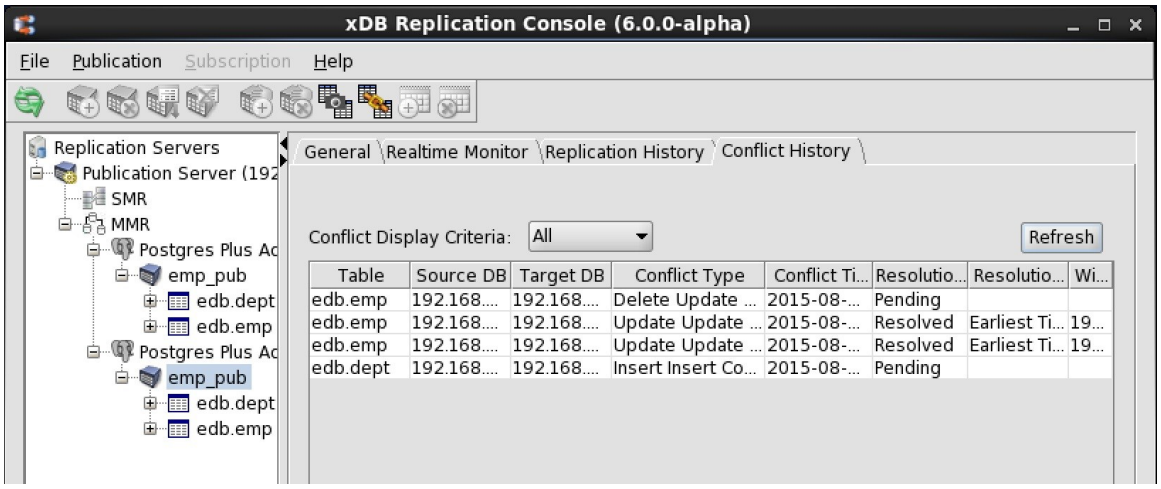


Figure 6-49 - Conflict History tab for log-based method

Note: The View Data link and Conflict Details window displayed for multi-master replication systems configured with the trigger-based method of synchronization replication are not available for multi-master replication systems configured with the log-based method of synchronization replication.

The Source DB and Target DB columns provide the IP address and database names of the source and target master nodes involved in the conflict.

You can also obtain this information from a SQL query rather than from the xDB Replication Console graphical user interface. The following query can be run from a master node to display information regarding pending (unresolved) conflicts:

```
SELECT DISTINCT
  conflict_type,
  table_name,
  pk_value,
  d1.db_host AS src_db_host,
  d1.db_port AS src_db_port,
  d1.db_name AS src_db_name,
  src_rrep_sync_id,
  d2.db_host AS target_db_host,
  d2.db_port AS target_db_port,
  d2.db_name AS target_db_name,
  target_rrep_sync_id,
  c.notes
FROM _edb_replicator_pub.xdb_conflicts c
  JOIN _edb_replicator_pub.xdb_pub_database d1 ON c.src_db_id = d1.pub_db_id
  JOIN _edb_replicator_pub.xdb_pub_database d2 ON c.target_db_id = d2.pub_db_id
  JOIN _edb_replicator_pub.rrep_tables t ON c.table_id = t.table_id
WHERE resolution_status = 'P'
ORDER BY t.table_name;
```

Example output from the query is shown by the following:

```
-[ RECORD 1 ]-------+-------------
conflict_type       | II
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | edb
src_rrep_sync_id    | 41939160
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode
target_rrep_sync_id | 42289824
notes               |
-[ RECORD 2 ]-------+-------------
conflict_type       | DU
table_name          | emp
pk_value            | empno=9003
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | edb
src_rrep_sync_id    | 41940704
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode
target_rrep_sync_id | 42292848
notes               |
```

6.6.10.2 Conflict Resolution Concept for the Log-Based Method

Manual conflict resolution typically requires modification of rows in one or more publication tables to correct erroneous entries. Such changes can be done using a utility such as PSQL or pgAdmin (Postgres Enterprise Manager Client in Advanced Server).

Manual publication table corrections must usually be isolated – that is, these modifications must be limited to the publication tables you are directly changing and must not be replicated to the other master nodes as would normally occur in the multi-master replication system.

To prevent the xDB Replication Server from replicating changes to one or more publication tables during a synchronization operation, the changes to the publication tables must be made within a transaction block that includes a reference to an xDB control schema table. This reference to the xDB control schema table causes the xDB Replication Server to skip the transaction block when performing a synchronization replication.

Note: Not every xDB control schema table prevents this replication of a transaction block. Use the SQL `UPDATE` statement as shown by the following.

The SQL `UPDATE` statement shown in the following transaction block is to be included to prevent replication of other publication table changes appearing within the same transaction block:

```
BEGIN;
UPDATE _edb_replicator_pub.rrep_properties SET value =
current_timestamp WHERE key = 'last_mcr_timestamp';

One or more SQL statements to correct publication tables
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
END;
```

When such a transaction block is executed within a master node, the insert, update, or deletion of rows to any publication table within the transaction block are not replicated to any other master node by the xDB Replication Server when the next synchronization replication occurs.

As many such transaction blocks can be run on any master node as necessary to change the publication table rows to resolve the conflicts. The resulting changes are isolated to the master node on which the transaction block is run, so each master node can be independently corrected.

6.6.10.3 Overview of Correction Strategies

Before you begin manual resolution correction, it is important to determine the extent of the inconsistencies that may have occurred in the publication tables across the master nodes of the replication system.

The Conflict History tab and the SQL query described in Section 6.6.10.1 can help determine the source of an initial conflict.

However, once this conflict has occurred, your replication system may have processed and replicated additional transactions during that synchronization operation. Some of these subsequent replications may have succeeded as expected, but others may have failed or produced unexpected results as a consequence of the prior conflict.

If you have a replication schedule in effect, additional synchronization operations can occur, which may create additional conflicts.

Therefore, when you have discovered that a conflict has occurred, it is strongly recommended that you stop the publication server. Use the `stop` option of the Linux scripts or Windows services described in Step 1 of Section 5.2.1.

In this way, you can carefully analyze the content of the publication tables in question to determine the best course of action to take without the interference of continuing updates by a running replication system.

When analyzing your tables you must determine which publication tables contain inconsistent rows across master nodes (that is, missing rows on some master nodes, or rows with different column values for the same primary key on different master nodes).

The general steps to resolving the problem following this analysis are the following:

Step 1: Make the necessary manual corrections to the rows in the publication tables across all master nodes to get them into an initial, consistent state so each publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

table has the same set of identical rows across master nodes. This may be to a state before the conflicting transactions occurred, depending upon what you determine to be the easiest course of action for fully resolving the conflict.

Step 2: Apply transactions (either from your application or from transaction blocks as defined in Section 6.6.10.2) so that all publication tables across all master nodes are updated consistently according to the desired, expected result.

Step 3: In the control schema, update certain indicators for the conflicting entries to show that these conflicts have been resolved. This update changes the Resolution Status of these entries to `Resolved` in the Conflict History tab. These entries will no longer appear in the SQL query described in Section 6.6.10.1.

Perform the Step 3 updates to the control schema of the controller database. The currently designated controller database can be determined from the content of the xDB Replication Configuration file (see Section 2.3.1.3). The publication server ensures that the control schema changes made on the controller database are replicated to the control schemas of all publication databases to maintain metadata consistency across all publication databases.

Step 4: Resume operation of your replication system. Start the publication server and recreate the replication schedule if you were using one.

For accomplishing steps 1 and 2, use some combination of the following methods. Which methods you use depends upon the state of your publication tables.

- **Manual Publication Table Correction.** Use a utility such as PSQL or pgAdmin (Postgres Enterprise Manager Client in Advanced Server) to manually correct the rows in the publication tables across all master nodes without replicating these changes. Apply these manual corrections within the transaction block described in Section 6.6.10.2.
- **Correction Using New Transactions.** Rerun your application on one master node to create new transactions that you will allow to replicate to all other master nodes. Use this method after you have ensured that all publication tables are in a consistent state across all master nodes.

Each of these methods is described in more detail in the following sections.

For purposes of illustration, the following replication environment is used.

- A 3-node multi-master replication system has been established. The master node names are `mmrnode_a` (the master definition node and the controller database), `mmrnode_b`, and `mmrnode_c`.
- The publication is named `emp_pub` and uses the `dept` and `emp` tables that have been used as examples throughout this document.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The conflict used to illustrate the conflict resolution methods is a uniqueness conflict occurring on the `dept` table on primary key column `deptno` on value `50` resulting from the `INSERT` statements shown by the following:

On `mmrnode_a`, the following statement is run:

```
INSERT INTO dept VALUES (50, 'FINANCE', 'CHICAGO');
```

On `mmrnode_b`, the following statement is run:

```
INSERT INTO dept VALUES (50, 'MARKETING', 'LOS ANGELES');
```

A synchronization replication is then performed.

The following is the content of table `dept` on `mmrnode_a`:

```
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | FINANCE    | CHICAGO
(5 rows)
```

The following is the content of table `dept` on `mmrnode_b`:

```
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

The following is the content of table `dept` on `mmrnode_c`:

```
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

The Conflict History tab shows the following entry:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

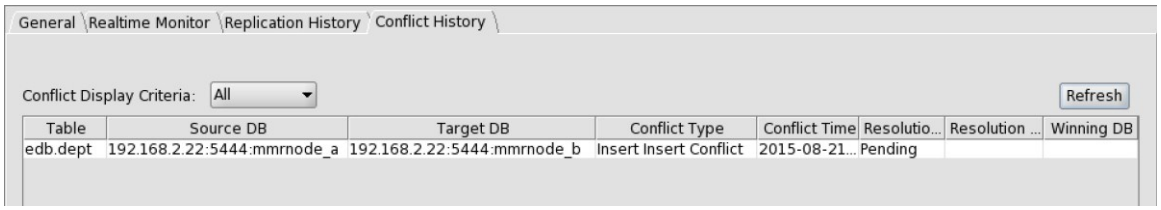


Figure 6-50 - Conflict History tab with a uniqueness conflict

The following is the output from the SQL query described in Section 6.6.10.1.

```
-[ RECORD 1 ]-------+-------------
conflict_type       | II
table_name          | dept
pk_value            | deptno=50
src_db_host         | 192.168.2.22
src_db_port         | 5444
src_db_name         | mmrnode_a
src_rrep_sync_id    | 44713808
target_db_host      | 192.168.2.22
target_db_port      | 5444
target_db_name      | mmrnode_b
target_rrep_sync_id | 44718040
notes               |
```

The following sections describe the application of different methods to resolve this conflict.

6.6.10.4 Manual Publication Table Correction

The first step required in all manual conflict resolutions is to ensure all publication tables are consistent across all master nodes – that is, all corresponding tables have the same rows with the same column values.

Once this state is achieved, you can then reapply transactions that may have failed to replicate successfully.

In the preceding example, the inconsistencies are the following:

- Master nodes `mmrnode_a` and `mmrnode_b` each contain a row with primary key value `50`, but the other column values in the row are different.
- Master node `mmrnode_c` does not have a row with primary key value `50`.

Assuming that the correct state of the `dept` table should be the one in `mmrnode_b`, the following options are available to correct the state of all master nodes:

- Manually correct the `dept` table in `mmrnode_a` and `mmrnode_c`. That is, update the row in `mmrnode_a` so it has the correct values, and insert the missing row in `mmrnode_c`. The `dept` table on all nodes is now consistent and up-to-date.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Manually delete the row with primary key value `50` from the table on both `mmrnode_a` and `mmrnode_b`. This brings the `dept` table on all master nodes back to a prior, consistent state. Then, with the multi-master replication system running, perform the insert transaction again using the correct column values on any one of the master nodes.

After the publication table rows are corrected, update the appropriate control schema table in the publication database currently designated as the controller database to indicate that the conflict has been resolved.

Each of the methods outlined in the preceding bullet points are described in more detail in the following sections.

The method outlined by the first bullet point is accomplished as follows.

Step 1: Manually correct the rows in the publication tables with SQL statements incorporated within a transaction block as described in Section 6.6.10.2.

On `mmrnode_a`, correct the erroneous row by running the following transaction block:

```
BEGIN;
UPDATE _edb_replicator_pub.rrep_properties SET value = current_timestamp
  WHERE key = 'last_mcr_timestamp';
UPDATE edb.dept SET dname = 'MARKETING', loc = 'LOS ANGELES'
  WHERE deptno = 50;
COMMIT;
```

This is shown by the following:

```
mmrnode_a=# BEGIN;
BEGIN
mmrnode_a=# UPDATE  edb_replicator_pub.rrep_properties SET value = current_timestamp
mmrnode_a-#   WHERE key = 'last_mcr_timestamp';
UPDATE 1
mmrnode_a=# UPDATE edb.dept SET dname = 'MARKETING', loc = 'LOS ANGELES'
mmrnode_a-#   WHERE deptno = 50;
UPDATE 1
mmrnode_a=# COMMIT;
COMMIT
mmrnode_a=# SELECT * FROM edb.dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_c`, insert the missing row with the following transaction block:

```
BEGIN;
UPDATE _edb_replicator_pub.rrep_properties SET value = current_timestamp
  WHERE key = 'last_mcr_timestamp';
INSERT INTO edb.dept VALUES (50,'MARKETING','LOS ANGELES');
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
COMMIT;
```

This is shown by the following:

```
mmrnode_c=# BEGIN;
BEGIN
mmrnode_c=# UPDATE _edb_replicator_pub.rrep_properties SET value = current_timestamp
mmrnode_c-#   WHERE key = 'last_mcr_timestamp';
UPDATE 1
mmrnode_c=# INSERT INTO edb.dept VALUES (50,'MARKETING','LOS ANGELES');
INSERT 0 1
mmrnode_c=# COMMIT;
COMMIT
mmrnode_c=# SELECT * FROM edb.dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

The `dept` table on `mmrnode_a` and `mmrnode_c` now match the content of the table on `mmrnode_b`:

Step 2: In the control schema of the publication database currently designated as the controller database, modify the entry in the `xdb_conflicts` table to indicate the conflict has been resolved. Table `xdb_conflicts` is located in schema `_edb_replicator_pub`.

Note: The entries in table `xdb_conflicts` only affect the data that appears in the Conflict History tab and the SQL query described in Section 6.6.10.1. Changing entries in `xdb_conflicts` has no effect on future replication operations, but provides a way to keep a record of how past conflicts were resolved.

Note the following points regarding the `xdb_conflicts` table:

- A row in the `xdb_conflicts` table appears as an entry in the Conflict History tab.
- The primary key of the `xdb_conflicts` table is comprised of columns `src_db_id`, `target_db_id`, `src_rrep_sync_id`, and `target_rrep_sync_id`. Column `src_db_id` contains a unique identifier for the master node in which a transaction occurred that results in a conflict when replicated to the master node identified by `target_db_id`. `src_rrep_sync_id` is the identifier of the transaction on the source master node involved in the conflict while `target_rrep_sync_id` is the identifier of the transaction on the target master node that is involved in the conflict. **Note:** The `src_rrep_sync_id` and `target_rrep_sync_id` values are used internally by xDB Replication Server and are not needed for the manual conflict resolution process.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Table `xdb_pub_database` in the control schema associates the database identifiers `src_db_id` and `target_db_id` with the master node attributes such as the database name, IP address, and port.
- Column `table_id` is the identifier of the publication table on which the conflict occurred. Association of the `table_id` value with the publication table attributes such as its name and schema is found in each master node in `_edb_replicator_pub.rrep_tables`.
- Column `pk_value` contains text indicating the primary key value that resulted in the conflict. The text is formatted as *`column_name=value`*. If the primary key is composed of two or more columns, each column and value pair is separated by the keyword `AND` such as *`column_1=value_1`* `AND` *`column_2=value_2`*. This provides the primary key of the row in the publication table designated by `table_id` that resulted in the conflict.
- Column `resolution_status` indicates the status of the conflict. Possible values are `P` (pending) or `C` (completed – the conflict has been resolved). This status appears in the Resolution Status column of the Conflict History tab.
- Column `win_db_id` can be used to record the database identifier of the master node that contains the "winning" (accepted) transaction. This information appears in the Winning DB column of the Conflict History tab.

The following shows the Conflict History tab prior to updating the `xdb_conflicts` table.

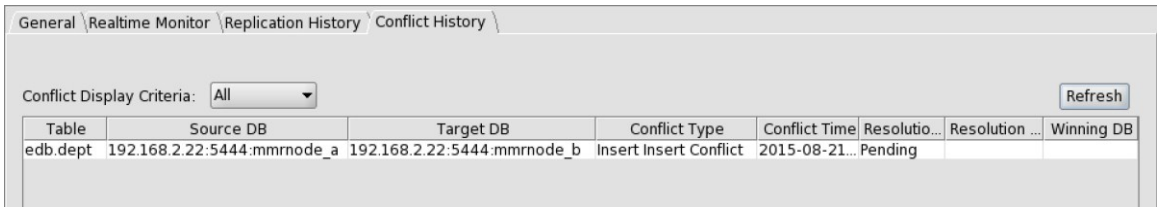


Figure 6-51 - Pending uniqueness conflict

The entry for the pending insert/insert conflict on the `deptno` primary key value of `50` can be located in `xdb_conflicts` with the following query for this example:

```
mmrnode_a=# SELECT * FROM _edb_replicator_pub.xdb_conflicts
mmrnode_a-#   WHERE pk_value = 'deptno=50'
mmrnode_a-#     AND conflict_type = 'II'
mmrnode_a-#     AND resolution_status = 'P';
-[ RECORD 1 ]-------+------------------------
src_db_id           | 1
target_db_id        | 22
src_rrep_sync_id    | 44713808
target_rrep_sync_id | 44718040
table_id            | 31
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
conflict_time       | 21-AUG-15 15:34:55.134171
resolution_status   | P
resolution_strategy |
resolution_time     |
alert_status        |
conflict_type       | II
win_db_id           | 0
win_rrep_sync_id    | 0
notes               |
pk_value            | deptno=50
```

This entry appears in the Postgres Enterprise Manager Client as shown by the following:

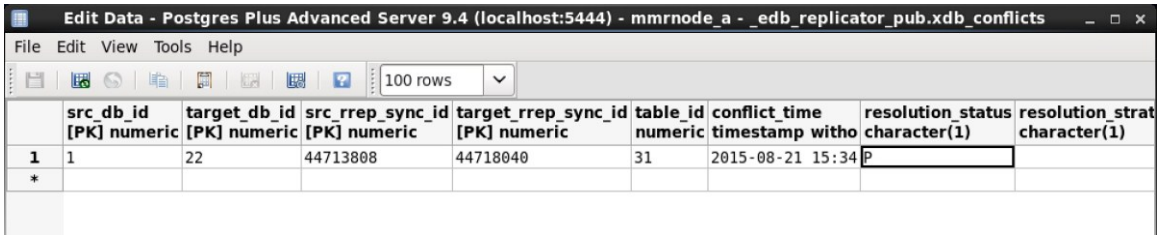


Figure 6-52 - Pending conflict in xdb_conflicts

Change the value in column `resolution_status` from `P` (pending) to `C` (completed) to indicate this conflict has been resolved. The value in `winning_db_id` is changed to `22` to indicate master node `mmrnode_b` contains the winning transaction.

The SQL statement to perform this update for the `mmrnode_a` to the `mmrnode_b` synchronization conflict is the following:

```
UPDATE _edb_replicator_pub.xdb_conflicts SET
  resolution_status = 'C',
  win_db_id = 22
WHERE pk_value = 'deptno=50'
    AND conflict_type = 'II'
    AND resolution_status = 'P';
```

The following is the updated `xdb_conflicts` entry:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

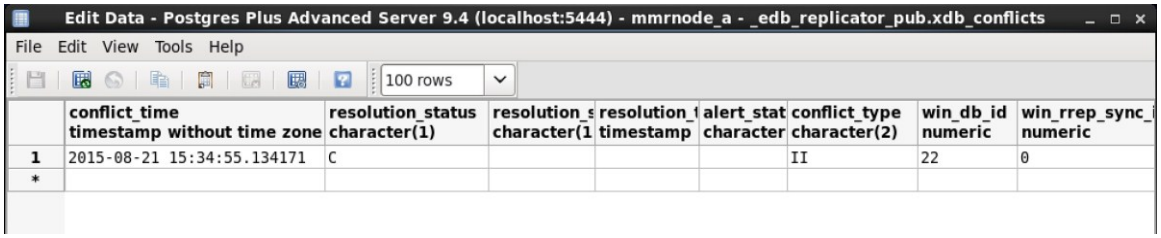


Figure 6-53 - Resolved conflict in xdb_conflicts

When viewed in the Conflict History tab, the entry now shows `Resolved` instead of `Pending` in the Resolution Status column, and the Winning DB column shows the address of master node `mmrnode_b`.

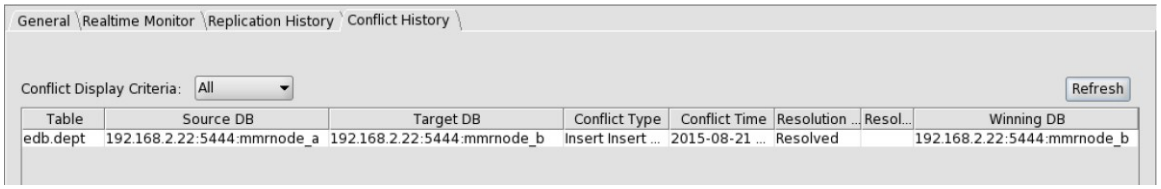


Figure 6-54 - Resolved uniqueness conflict

6.6.10.5 Correction Using New Transactions

Another method for bringing all the publication tables to a consistent state is by removing any changes caused by the conflicting transactions and then issuing new, corrected transactions at one master node, which you allow the multi-master replication system to synchronize to all other master nodes.

Referring back to the uniqueness conflict on the `dept` table, instead of correcting the erroneous row and inserting the row into the master node where it is missing as described in Section 6.6.10.4, you can delete the conflicting rows from all master nodes, then insert the correct row in one master node and let the multi-master replication system synchronize the correct row to all master nodes.

Step 1: Manually delete the inserted row from the publication tables in all master nodes using the transaction block described in Section 6.6.10.2.

On `mmrnode_a`, delete the erroneous row with the following transaction block:

```
BEGIN;
UPDATE _edb_replicator_pub.rrep_properties SET value = current_timestamp
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  WHERE key = 'last_mcr_timestamp';
DELETE FROM edb.dept WHERE deptno = 50;
COMMIT;
```

This is shown by the following:

```
mmrnode_a=# BEGIN;
BEGIN
mmrnode_a=# UPDATE _edb_replicator_pub.rrep_properties SET value =
current_timestamp
mmrnode_a-#   WHERE key = 'last_mcr_timestamp';
UPDATE 1
mmrnode_a=# DELETE FROM edb.dept WHERE deptno = 50;
DELETE 1
mmrnode_a=# COMMIT;
COMMIT
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

On `mmrnode_b`, delete the row even though the transaction created the correct result:

```
mmrnode_b=# BEGIN;
BEGIN
mmrnode_b=# UPDATE _edb_replicator_pub.rrep_properties SET value =
current_timestamp
mmrnode_b-#   WHERE key = 'last_mcr_timestamp';
UPDATE 1
mmrnode_b=# DELETE FROM edb.dept WHERE deptno = 50;
DELETE 1
mmrnode_b=# COMMIT;
COMMIT
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

On `mmrnode_c`, no changes are required as the conflicting transaction did not insert a new row into the table on this node:

```
mmrnode_c=# SET search_path TO edb;
SET
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Rerun the correct transaction on one master node with the multi-master replication system running. Do not run this within the transaction block described in Section 6.6.10.2 as the objective is to synchronize it to all master nodes.

For this example, the correct `INSERT` statement is executed on `mmrnode_a`:

On `mmrnode_a`:

```
mmrnode_a=# INSERT INTO dept VALUES (50, 'MARKETING', 'LOS ANGELES');
INSERT 0 1
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

Step 3: Perform synchronization replication.

The same rows now appear in the publication table on all master nodes.

On `mmrnode_a`;

```
mmrnode_a=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_b`;

```
mmrnode_b=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

On `mmrnode_c`;

```
mmrnode_c=# SELECT * FROM dept;
 deptno |   dname    |     loc
--------+------------+-------------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | MARKETING  | LOS ANGELES
(5 rows)
```

Step 4: In the control schema of the publication database currently designated as the controller database, modify the entry in the `xdb_conflicts` table to indicate the conflict has been resolved as in Step 2 of Section 6.6.10.4.

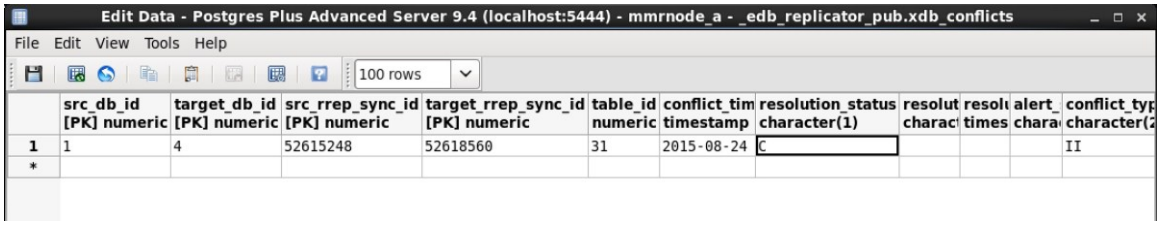


Figure 6-55 - Resolved conflict in xdb_conflicts

6.7 Viewing Conflict History

Conflict history shows the following types of events that occurred during synchronization replication:

- Uniqueness conflicts where two or more master nodes attempted to insert a row with the same primary key value or unique column value.
- Update/update conflicts where two or more master nodes attempted to update the same column of the same row
- Update/delete and delete/update conflicts where one master node attempted to update a row that was deleted by another master node.

See Section 6.6 for more information on conflict resolution.

Note: The conflict history can be viewed from the Publication node under any master node in the multi-master replication system. The history shows conflicts on all publication tables of all master nodes that occurred during synchronization, and hence, the history appears the same regardless of the master node under which it is viewed.

Note: For uniqueness (insert/insert) conflicts the number of entries appearing under the Conflict History tab differs when the trigger-based method of synchronization replication is used as compared to the log-based method. If the trigger-based method is used, a single insert/insert conflict appears as two entries in the conflict history. Each entry differs in that the source and target database fields for the two conflicting master nodes are

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

interchanged. If the same conflict occurs when the log-based method is used, only one entry appears in the conflict history.

The following steps describe how to view the conflict history.

Step 1: Select any Publication node under a Database node representing a master node. Tabs labeled General, Realtime Monitor, Replication History, and Conflict History appear.

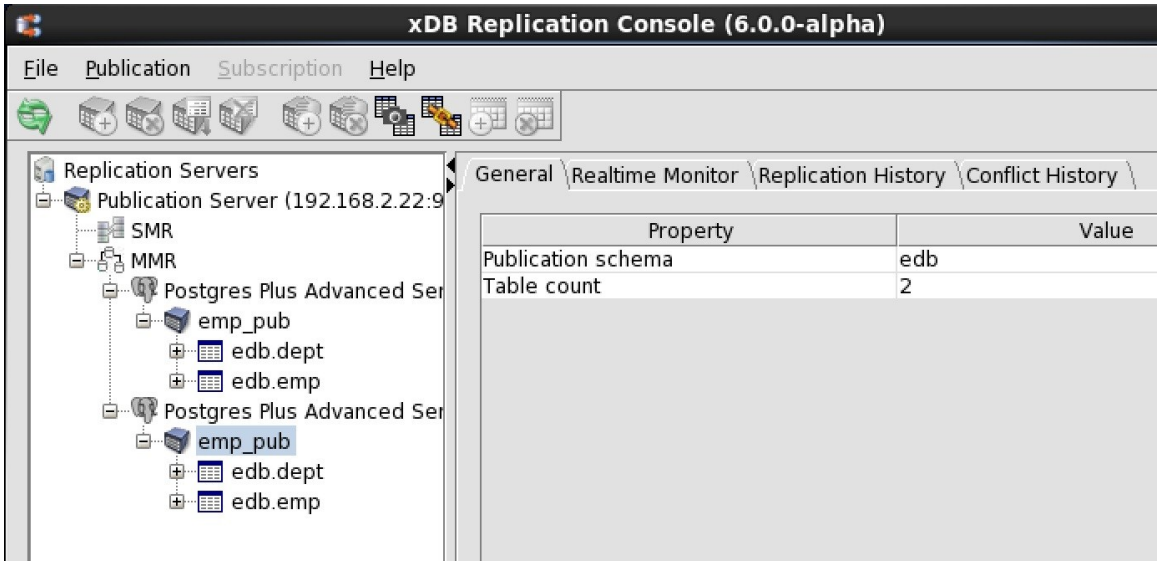


Figure 6-56 - Selecting a publication on which to view conflict history

Step 2: Click the Conflict History tab to show conflict history. Click the Refresh button to ensure all conflicts are listed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

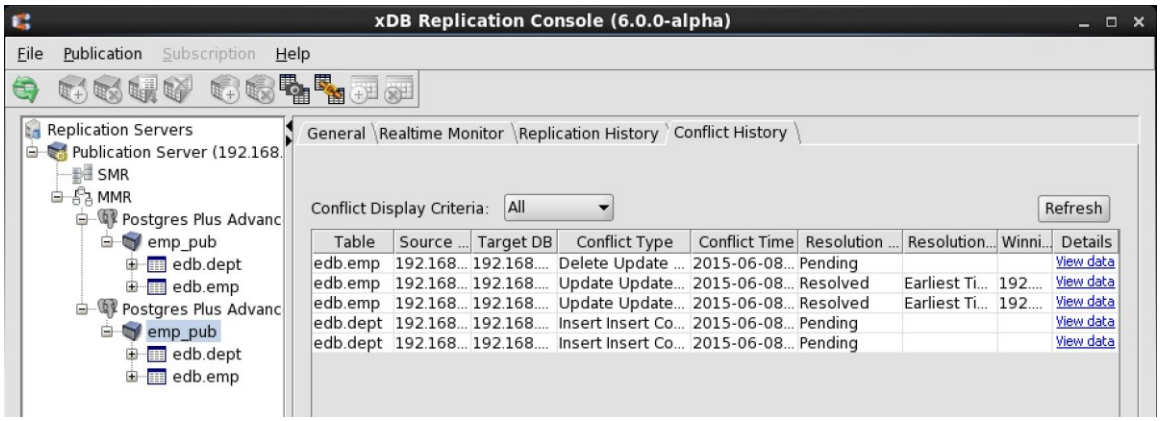


Figure 6-57 - Conflict History tab

Step 3: Use the Conflict Display Criteria drop-down list to display only conflicts of the chosen status.

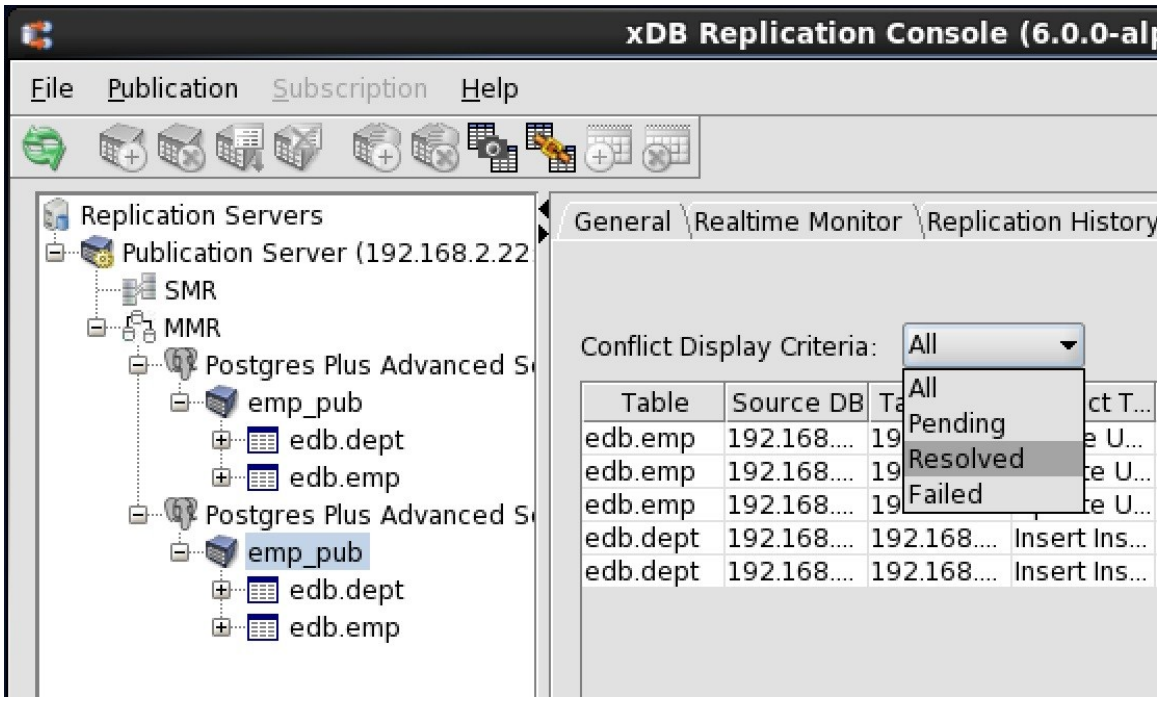


Figure 6-58 - Selecting conflict history by status

Step 4: Click the View Data link to show the details of a particular conflict.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: The View Data link and Conflict Details window are available only for multi-master replication systems configured with the trigger-based method of synchronization replication. There is no View Data link or Conflict Details window for multi-master replication systems configured with the log-based method of synchronization replication.

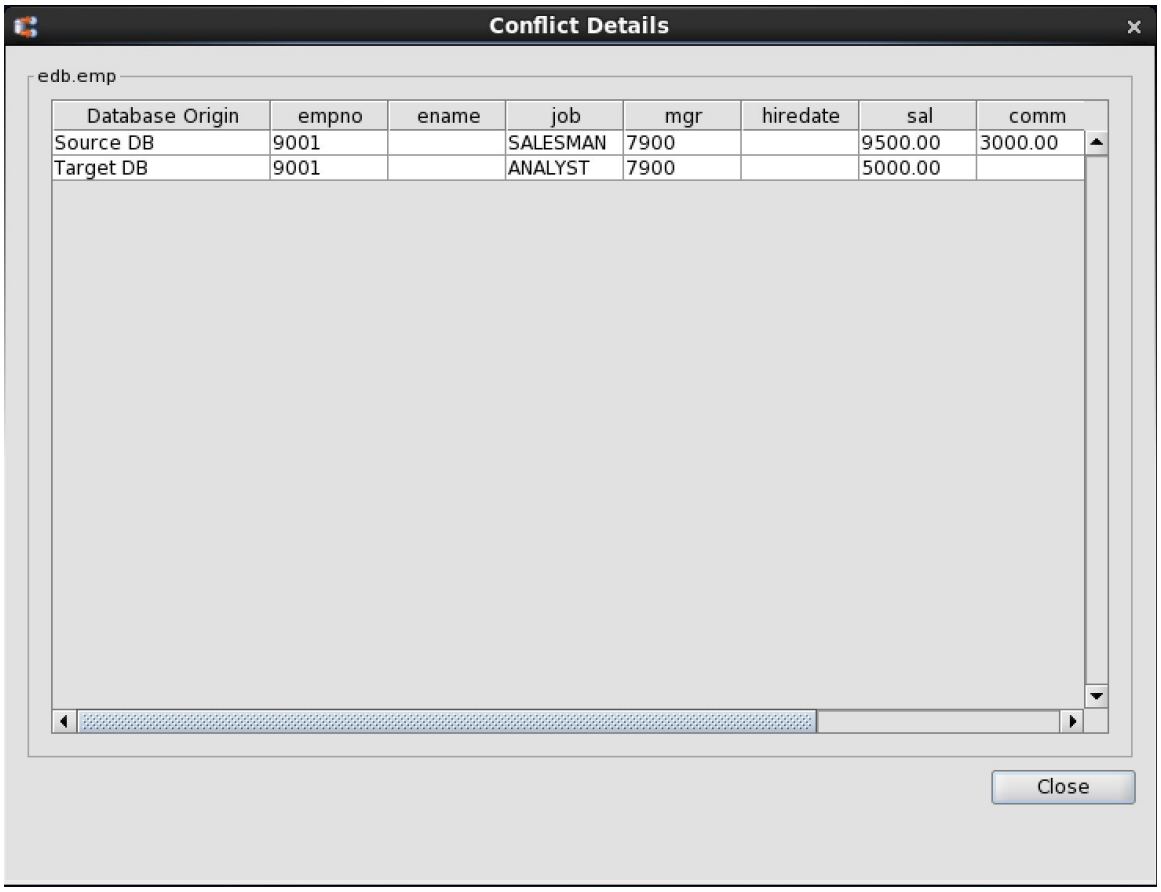


Figure 6-59 - Conflict Details window

6.8 Updating the Conflict Resolution Options

A current conflict resolution option on a publication table can be changed. See Section 6.6 for information on conflict resolution.

Step 1: Make sure the publication server whose node is the parent of the publication you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: Select the Publication node under the Publication Database node representing the master definition node.

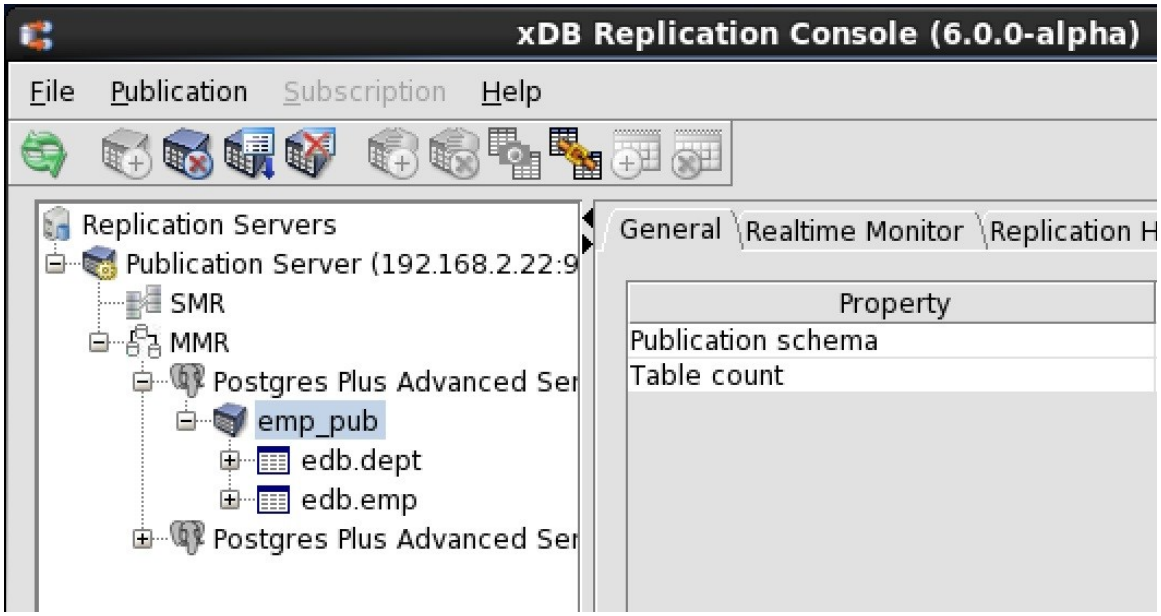


Figure 6-60 - Selecting a publication in which to update conflict resolution options

Step 3: Open the Conflict Resolution Options dialog box in any of the following ways:

- From the Publication menu, choose Update Publication, then Conflict Resolution Options.
- Click the secondary mouse button on the Publication node, choose Update Publication, and then choose Conflict Resolution Options.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

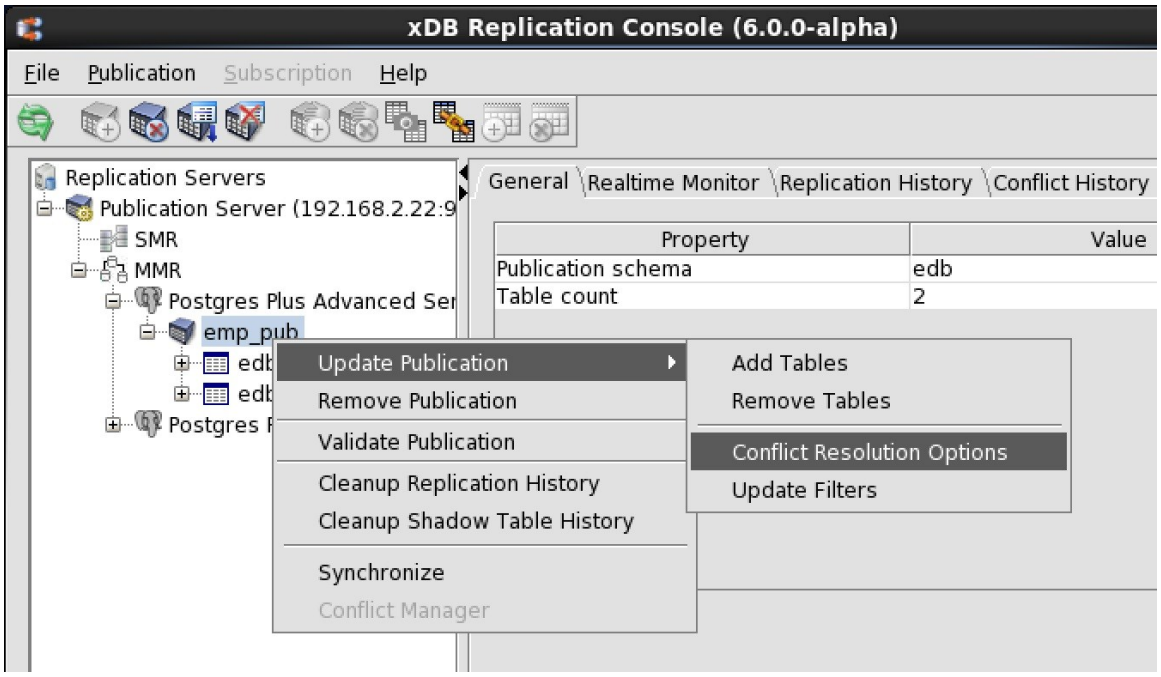


Figure 6-61 - Opening the Conflict Resolution Options dialog box

Step 4: For each table, you can select the primary conflict resolution strategy and a standby strategy by clicking the primary mouse button over the appropriate box to expose a drop-down list of choices.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

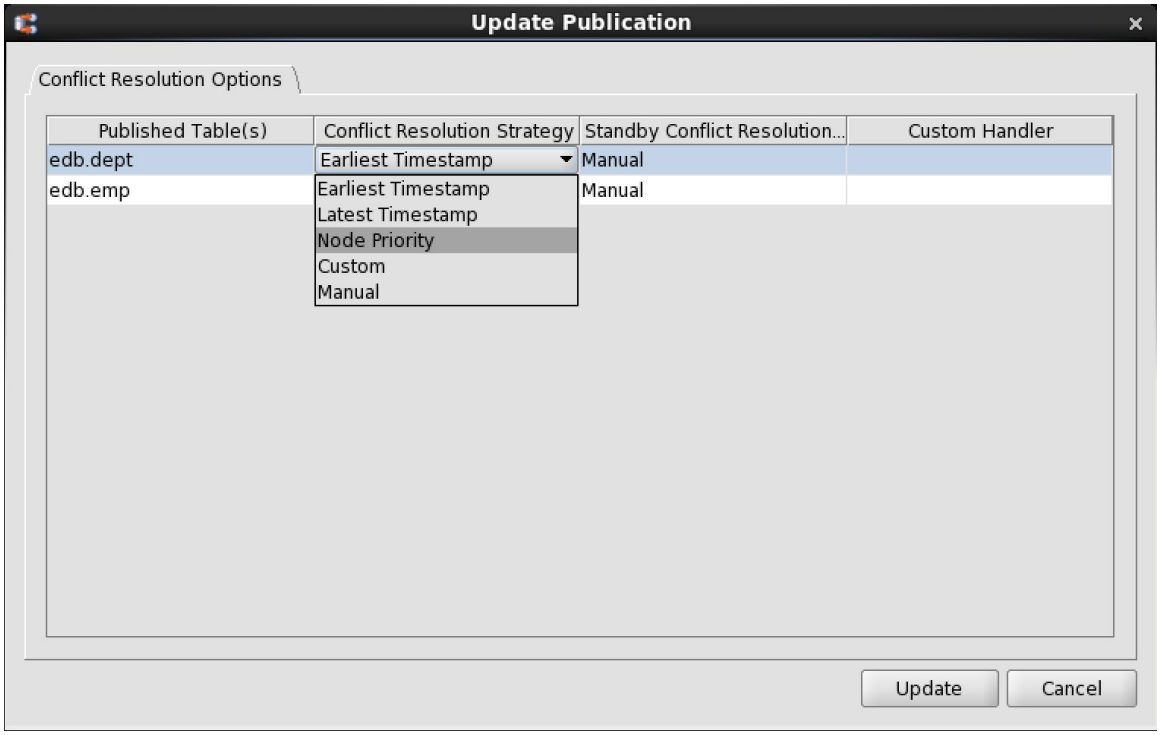


Figure 6-62 - Updating conflict resolution strategies

Step 5: Click the Update button, and then click OK in response to Conflict Resolution Options Updated Successfully.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

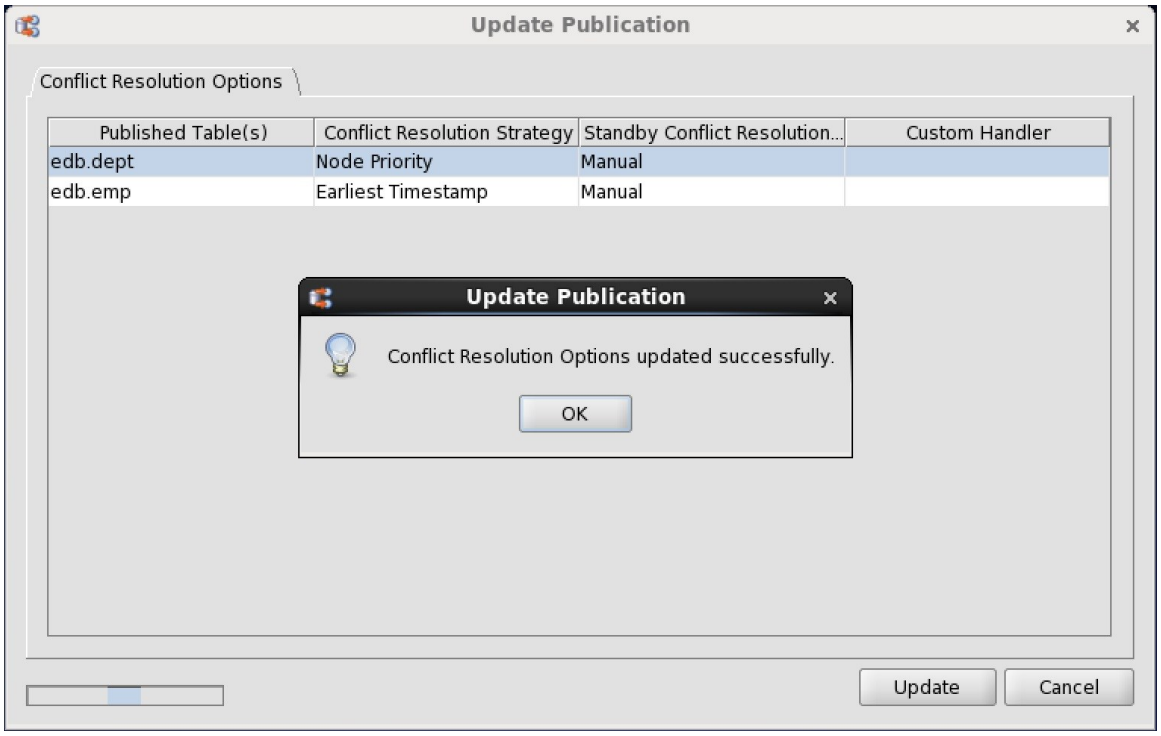


Figure 6-63 - Successfully updated conflict resolution options

6.9 Enabling/Disabling Table Filters on a Master Node

Table filters must first be defined in a set of available table filters in the publication before they can be enabled on a master node. See Section 6.2.3 for information on defining table filters in a multi-master replication system.

Note: See Section 2.2.12.3 for table setup requirements for a log-based replication system as well as general restrictions on the use of table filters.

The following are the steps for enabling or disabling table filters on an existing master node.

Step 1: Make sure the publication server whose node is the parent of the master nodes of the replication system is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication Database node corresponding to the master node on which you wish to enable or disable individual filter rules.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

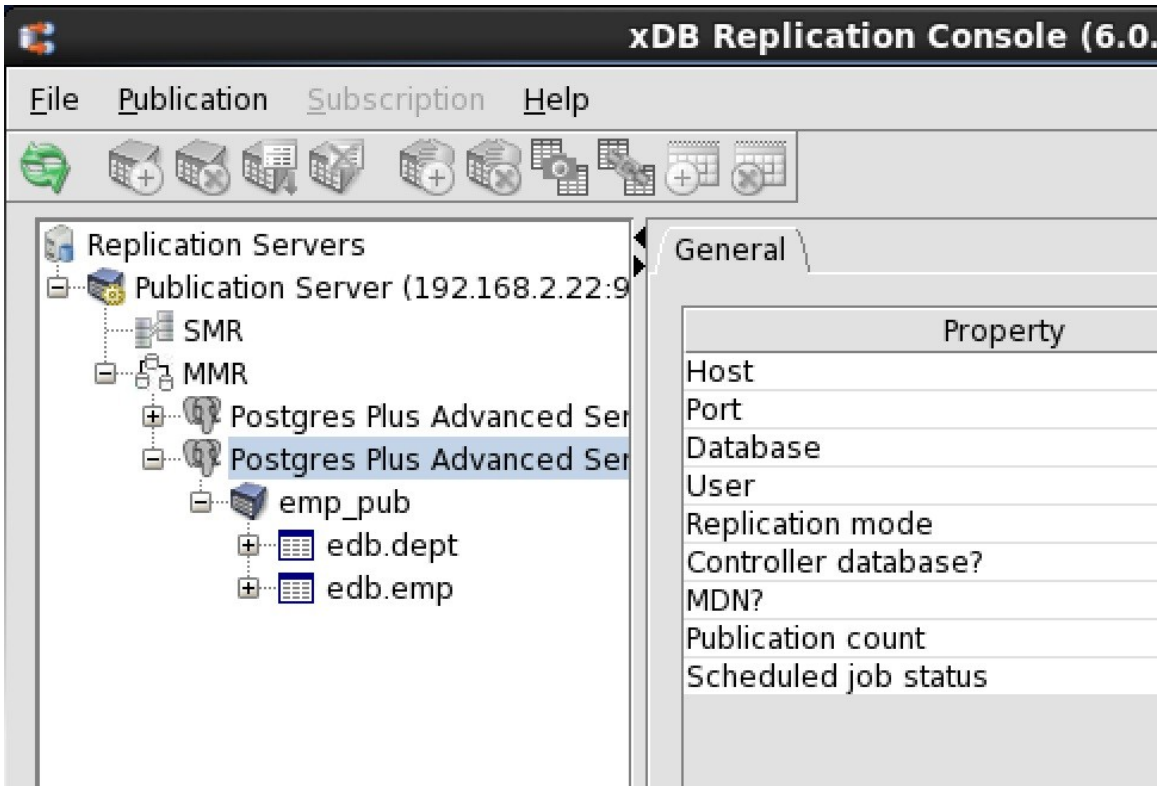


Figure 6-64 - Selecting a master node on which to enable or disable filter rules

Step 3: Click the secondary mouse button on the Publication Database node and choose Update Filter Rule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

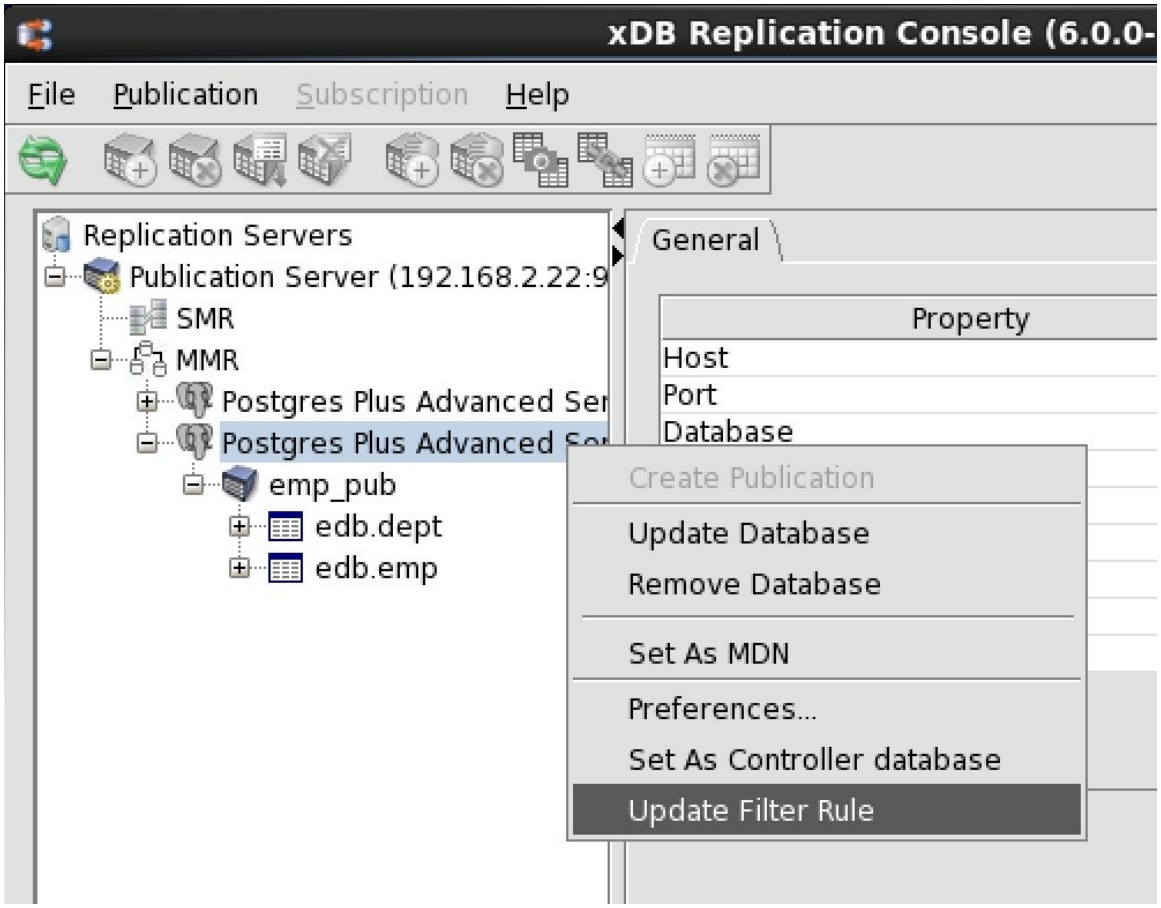


Figure 6-65 - Opening the Filter Rules tab on a master node

Note: If you wish to enable or disable filter rules on the current master definition node, you must first switch the role of the master definition node to another master node in order to expose the Update Filter Rule option in the master node context menu. See Section 6.10 for directions on switching the master definition node.

The master node you choose as the new master definition node should contain a superset of, or at least an equivalent set of data as the current master definition node. The reason for this is to ensure that the former master definition node contains the complete set of data satisfying the filtering criteria after you take a snapshot from the new master definition node to the former master definition node on which you just enabled the table filters.

Step 4: In the Filter Rules tab check or uncheck the boxes to specify the filter rules to enable or disable on the master node. At most one filter rule may be enabled on any given table. Click the Save button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

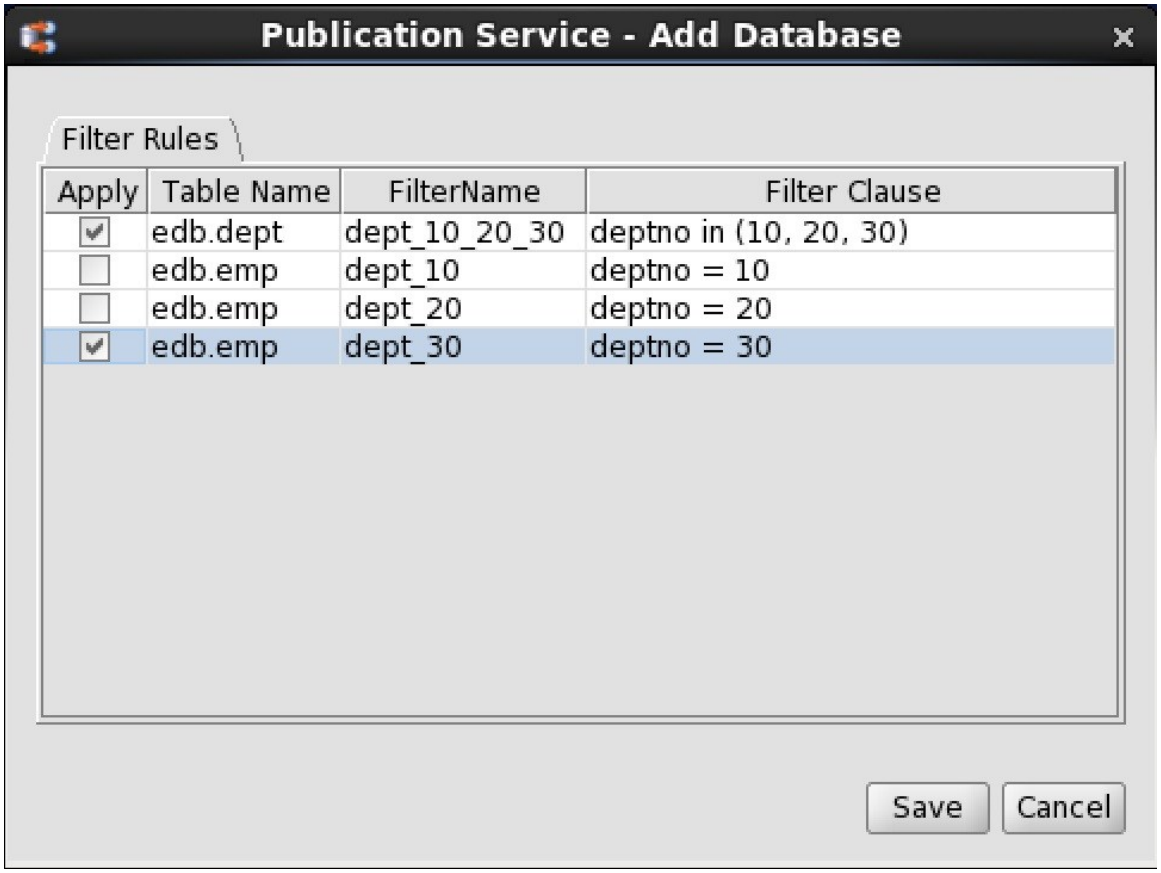


Figure 6-66 - Filter Rules tab

Step 5: A confirmation box appears presenting a warning message and a recommendation to perform a snapshot replication to any master node on which you changed the filtering criteria.

Click the Ok button in the confirmation box to proceed with the update to the filter rule selections. Click the Cancel button to return to the Filter Rules tab if you wish to modify your filter rule selections.

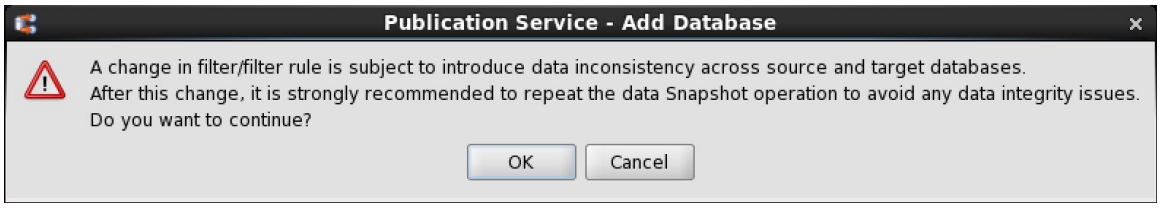


Figure 6-67 - Change filter rule confirmation

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 6: If you clicked the Ok button in the preceding step, the Filter Rules updated successfully confirmation message appears if the update was successful.

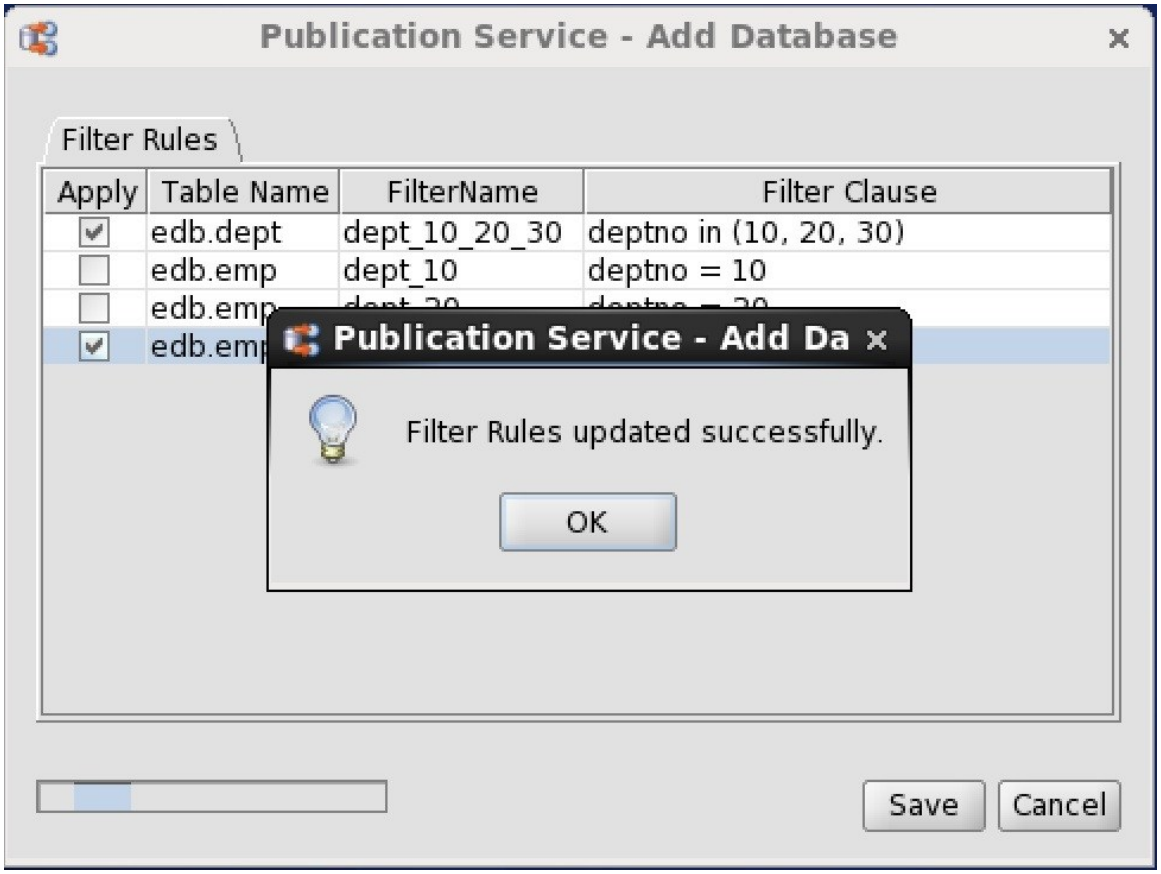


Figure 6-68 - Successful update of filter rules

If you clicked the Cancel button in the preceding step, the Filter Rules tab reopens. You can modify your filter rule selections by repeating Step 4, or you can click the Cancel button in the Filter Rules tab to abort the filter rule updates on the master node.

Step 7: It is strongly recommended that a snapshot replication be performed to the master node that contains tables on which the filtering criteria has changed.

A snapshot ensures that the content of the master node tables is consistent with the updated filtering criteria. See Section 6.5.1 for information on performing a snapshot replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: The master definition node, which provides the source of the table content for a snapshot, should contain a superset of all the data contained in the other master nodes of the multi-master replication system. This ensures that the target of the snapshot receives all of the data that satisfies the updated filtering criteria.

On the contrary, if the master definition node contains only a subset of all the data contained in the other master nodes, then a snapshot to another master node may not result in the complete set of data that is required for that target master node.

6.10 Switching the Master Definition Node

After the multi-master replication system is created, you can switch the role of the master definition node with another master node.

Step 1: Make sure the publication server whose node is the parent of the master nodes of the replication system is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication Database node corresponding to the master node that you wish to set as the master definition node.

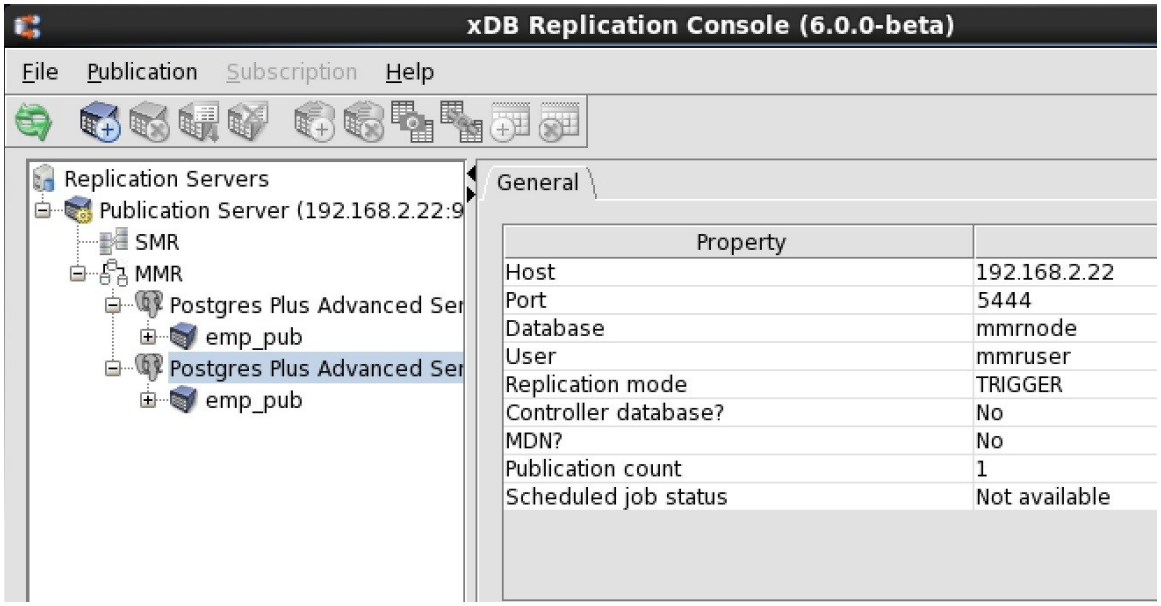


Figure 6-69 - Selecting the master node to set as the master definition node

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: Click the secondary mouse button on the Publication Database node and choose Set as MDN.

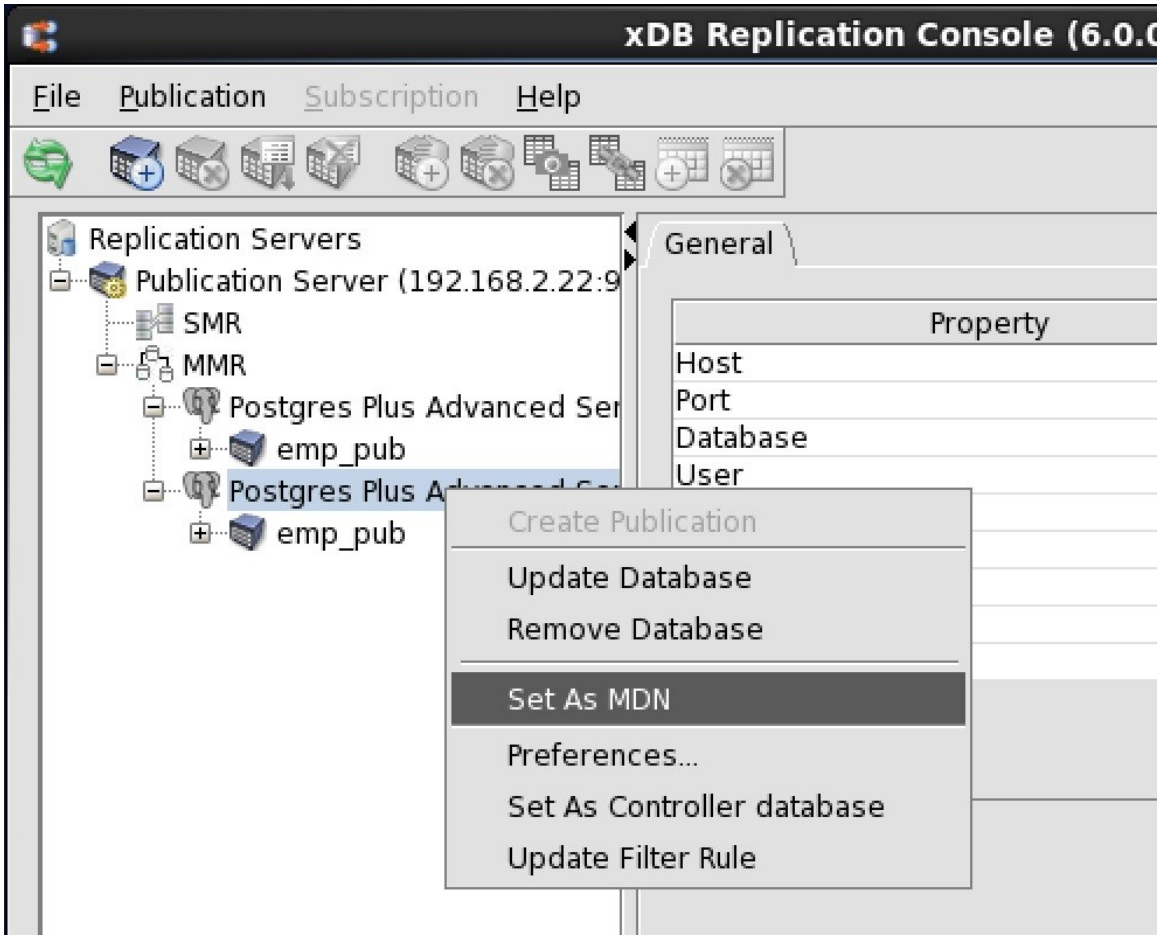


Figure 6-70 - Setting the master definition node

Step 4: In the Set as MDN confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

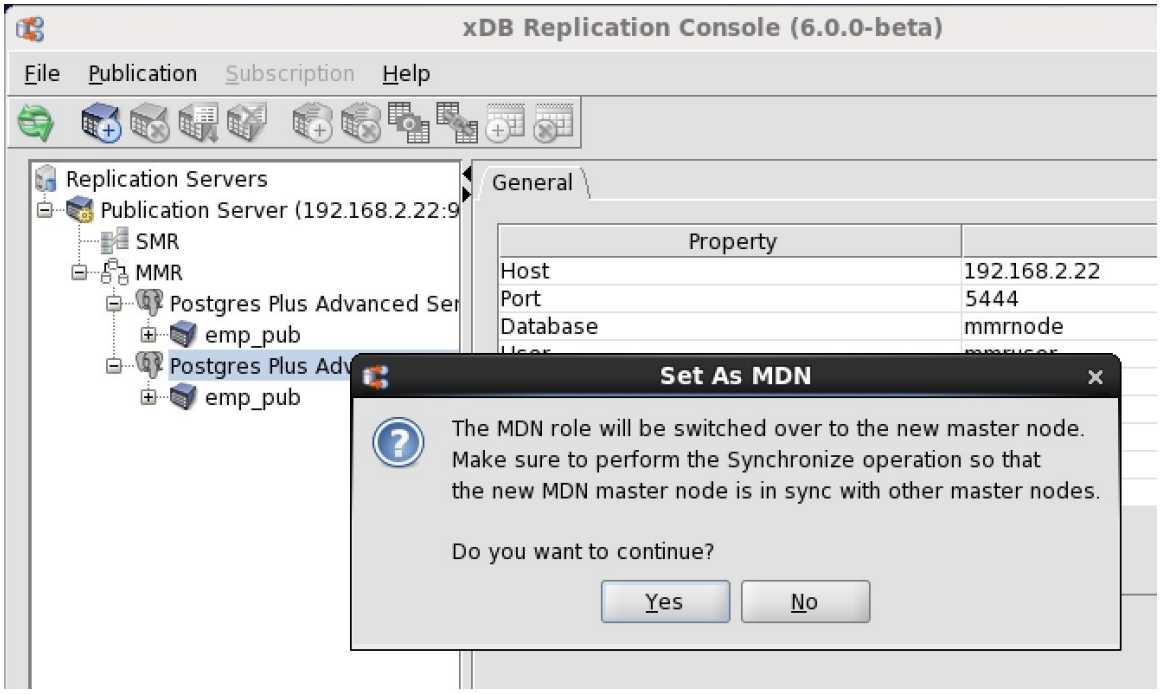


Figure 6-71 - Set as MDN confirmation

Step 5: The selected master node is now the master definition node.

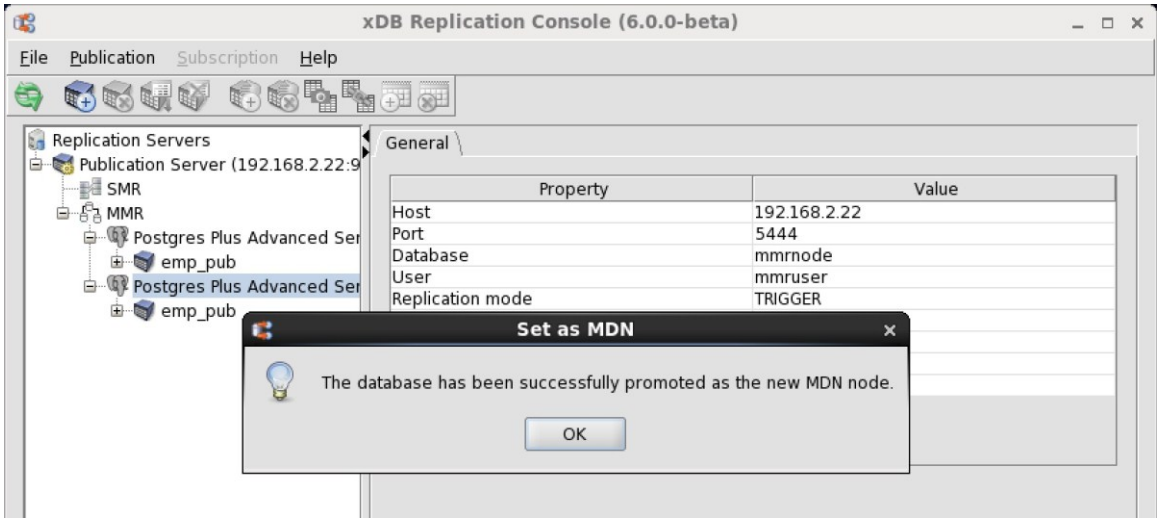


Figure 6-72 - Database promoted to master definition node

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 6: The value `Yes` in the MDN field of the Property window indicates this database is the master definition node.

Note: The new master definition node is moved to the top of the replication tree in the xDB Replication Console.

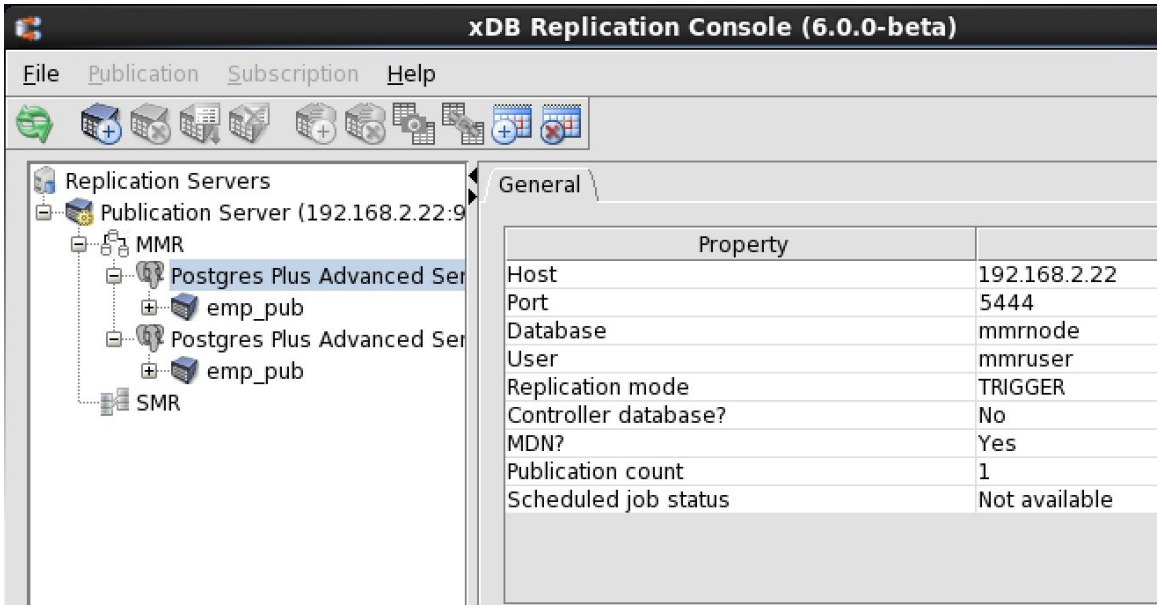


Figure 6-73 - Master definition node (MDN) indicated by 'Yes' in the Property window

Note: You should now perform a synchronization replication to ensure that the new master definition node is synchronized with the other master nodes. See Section 6.5.2 for directions on performing a synchronization replication.

6.11 Ensuring High Availability

In a multi-master replication system, the master nodes participating in replication can reside on separate physical hosts. If any master node goes offline, the master nodes on the other hosts continue to synchronize transactions amongst themselves thereby ensuring consistency of the publication tables on the remaining active master nodes. When an offline master node is brought back online, pending transactions involving that master node are synchronized with the other master nodes of the replication system. No transaction data is lost between the master nodes.

Thus, an inherent characteristic of multi-master replication systems is that each master node serves as a backup for the other nodes, and any such node can provide consistent publication data to applications.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Similarly, the complete, multi-master replication system configuration information (that is, the control schema and its control schema objects) is stored in each publication database (that is, every master node) of the multi-master replication system.

Thus, should any master node go offline, the configuration information stored in the control schema is always available to the publication server in order to continue operation of the replication system.

Though every publication database contains a copy of the control schema, the publication database designated as the controller database has special significance to the operation of the replication system.

The significance of the controller database and its proper usage to ensure high availability of the replication system are discussed in the following sections.

6.11.1 Significance of the Controller Database

At any given point in time during operation of the replication system, one of the publication databases of the master nodes is designated as the controller database.

The controller database can be identified in either of the following ways:

- In the xDB Replication Console, when a master node is selected, the Controller database field in the Property window is set to `Yes` if this master node is the current controller database.
- In the xDB Replication Configuration file, the authentication and connection parameters are set to the controller database. See Section 2.3.1.3 for information on the xDB Replication Configuration file.

When a replication system is in use, the xDB Replication Server, and particularly, the publication server component, accesses the currently designated controller database for any configuration information.

Any changes that you make to the replication system configuration using the xDB Replication Console or the xDB Replication Server CLI are first updated in the control schema of the controller database, and then replicated by the xDB Replication Server to the other publication databases to keep such information consistent.

Note: Replication history may take a longer period of time to replicate from the controller database to the other publication databases, therefore it is possible that some replication history may be lost if access to the controller database fails, and a switchover is made to another publication database to act as the controller database. See Section 7.4 for information on replication history.

Therefore it is important that the controller database be accessible whenever the replication system is use.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

There may be circumstances where access to the controller database cannot be maintained such as scheduled maintenance that must be performed on the database server hosting the controller database or an unexpected network or system problem.

Such circumstances are addressed in the following sections.

6.11.2 Automatic Switchover of the Controller Database

If the publication server is currently running with its connection to the controller database, and that database suddenly becomes inaccessible such as with a network or system problem, the xDB Replication Server automatically performs a connection to another online publication database to act as the controller database.

Thus, there is no apparent disruption in the operation of the xDB Replication Server.

The controller database authentication and connection information is modified accordingly in the xDB Replication Configuration file (see Section 2.3.1.3). Thus, any subsequent startups of the publication and subscription servers use this newly designated controller database.

The controller database can be subsequently changed to use another publication database as described in sections 6.11.3 and 6.11.4.

6.11.3 Switching an Active Controller Database

If at some point, the database server hosting the controller database must be taken offline for maintenance or some other reasons you can switch the role of the controller database to another publication database.

If the publication server is currently running, this switch can be made using the xDB Replication Console (see Section 7.7) or the xDB Replication Server CLI (see Section 8.3.26).

After the switch has occurred, you can take the former controller database offline. Any pending transactions involving the former controller database will be applied after it is brought back online.

If the publication server is not running, you can still change the controller database so that the publication server connects to a newly designated controller database when the publication server is started. See Section 6.11.4 for information on this method.

6.11.4 Restarting with an Alternate Controller Database

If for some reason the currently designated controller database cannot be accessed by the publication server, certain symptoms may occur such as the following:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The xDB Replication Console is unresponsive or the xDB Replication Server CLI commands fail unpredictably.
- The publication server is not running and it cannot be successfully started.

If it is determined that the controller database is inaccessible, then you can switch the controller database role to another publication database by editing the xDB Replication Configuration file so it contains the connection information of another master node in the replication system. See Section 2.3.1.3 for information on the xDB Replication Configuration file.

After the xDB Replication Configuration file has been modified, restart the publication server along with the subscription server if you are using that as well. See Section 5.2.1 for instructions on starting the publication server. See Section 5.3.1 for instructions on starting the subscription server.

6.12 Optimizing Performance

Various publication server configuration options are available to optimize the performance of multi-master replication systems.

Almost all publication server performance related configuration options for single-master replication systems are equally applicable to multi-master replication systems (except when they are database produce specific, such as for Oracle). See Section 5.8 for a discussion of these options.

The publication server configuration options are set in the publication server configuration file. See Section 10.4.1 for a detailed explanation of how to set the configuration options in this file.

In addition, for configuration options specifically applicable to publication databases configured with the log-based method of synchronization replication, see Section 10.4.1.15.

The following are some additional configuration options applicable to multi-master replication systems only.

uniquenessConflictDetection

The `uniquenessConflictDetection` option determines if uniqueness conflict needs to be detected at data load time or should be deferred to when data is applied against a target master node. Possible values are `EAGER` and `LAZY`. Set it to `EAGER` if there is a high probability of duplicate inserts across master nodes.

When the number of master nodes is equal to two, then the conflict detection is performed in the default `LAZY` mode.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When the number of master nodes is greater than two, then the conflict detection is always performed in `EAGER` mode. (A `LAZY` mode setting is ignored.) This is primarily required to avoid removing the already replicated conflicted changes from a target node, which otherwise is an expensive option.

```
uniquenessConflictDetection={EAGER | LAZY}
```

The default value is `LAZY` when the number of master nodes is two.

skipConflictDetection

The `skipConflictDetection` option controls whether or not to skip conflict detection during synchronization replication. The default is `false` and should be changed only when the probability of data conflict across master nodes is zero. For example if each master node operates on an independent set of data then turning on this option improves the replication time.

```
skipConflictDetection={true | false}
```

The default value is `false`.

deadlockRetryCount

In a multi-master replication system, if a deadlock is detected on a target master node, the `deadlockRetryCount` option controls the number of times the publication server attempts to retry application of the changes in the current replication cycle after waiting for the number of milliseconds specified by `deadlockWaitTime`. Set `deadlockRetryCount` to 0 to turn off this option in which case the failed changes are attempted in the next replication cycle.

```
deadlockRetryCount=n
```

The default value for *n* is `1`.

deadlockWaitTime

The `deadlockWaitTime` option is used with the `deadlockRetryCount` option to set the wait time in milliseconds before the publication server attempts to retry application of the changes on the target master node.

```
deadlockRetryCount=n
```

The default value for *n* is `1000`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7 Common Operations

This chapter describes configuration and maintenance operations of xDB Replication Server that are common to both single-master and multi-master replication systems.

For configuration and management of your replication system, the xDB Replication Console graphical user interface is used to illustrate the steps and examples in this chapter. The same steps can be performed from the operating system command line using the xDB Replication Server Command Line Interface (CLI). The commands of the xDB Replication Server CLI utility are described in Chapter 8.

Note: Though most steps described in this chapter apply to both single-master and multi-master replication systems, those steps that apply only to single-master replication systems are noted with **For SMR only**. Those steps that apply only to multi-master replication systems are noted with **For MMR only**.

7.1 Selecting Tables with the Wildcard Selector

When selecting tables for creating a publication for a single-master replication system (see Section 5.2.3) or a multi-master replication system (see Section 6.2.3), there may be cases where the number of available tables for selection is so large that simply choosing them from a checklist becomes a difficult and time-consuming process.

This difficulty can also be encountered when adding tables to an existing publication (see Section 7.6.3.1) or deleting tables from an existing publication (see Section 7.6.3.2).

In such cases, the wildcard selector provides the capability to choose a set of tables by using pattern matching similar to the technique used by the SQL statement `LIKE` clause.

7.1.1 Wildcard Selector Patterns

Pattern matching as performed by the wildcard selector is the process in which the eligible tables for an operation are returned in a filtered list if their schema and table name combination match a character string called a *pattern*.

Matching a pattern means that the schema and table name combined in a string formatted as `schema_name.table_name` matches the pattern, character by character, according to the rules designated for the characters appearing in the pattern.

If the `schema_name.table_name` string matches the pattern, then the schema and table are displayed in the *filtered list* for that pattern, which is the Available Tables field of the Wildcard Selector dialog box. You can then selectively choose the tables from the filtered list to be added to a local list, which contains the potential, candidate tables for the operation for which you are using the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Similarly, you can remove tables from the local list that had been previously selected if you decide that you do not want these particular tables applied for the operation.

With the exception of characters called *wildcards*, characters appearing in a pattern require that the character in the corresponding position in the *schema_name.table_name* string must match the pattern character in a case insensitive manner (that is, the letters A or a, match both A and a).

The pattern characters called *wildcard characters* or simply *wildcards* are interpreted in a special manner when compared to the corresponding character position of the *schema_name.table_name* character string.

Interpretation of pattern characters is described by the following:

- ? – Single-character wildcard specifies that any single character may exist in its position of the pattern. (The SQL LIKE clause uses the underscore character (_) for this purpose.)
- % - Multi-character wildcard specifies that any combination of multiple characters, including the absence of any character, may exist in its position of the pattern.
- [*abc...*] – List wildcard specifies that any one of the characters listed within the brackets may exist in its position of the pattern.
- [*a-d*] – Range wildcard specifies that any one character that is greater than or equal to the character preceding the hyphen (-) and less than or equal to the character following the hyphen may exist in its position of the pattern.
- [*abcd-f...*] – List and range combination wildcard specifies that any character that matches any of the list or range wildcard descriptions as described in the previous two bullet points may exist in its position of the pattern.
- Any character specified in the pattern other than ?, %, [,], and the characters enclosed within the square brackets of a list or range wildcard must exist in its position of the pattern. Pattern matching of such characters is case insensitive (for example, a pattern of edb.dept matches a schema and table with the name EDB.Dept).
- NOT *pattern*, !*pattern*, ! *pattern* – Exclusive pattern specifies that tables that match the pattern string indicated by *pattern* are omitted from the filtered list. Tables that do not match *pattern* are included in the filtered list. The keyword NOT may be in uppercase, lowercase, or mixed case, but must be followed by a single space character preceding *pattern*. !*pattern* specifies that *pattern* immediately follows the exclamation point (!) with no intervening space character. ! *pattern* specifies that a single space character exists between *pattern* and the exclamation point (!).
- *pattern** - Specify the asterisk (*) immediately following the pattern with no intervening space character if you want to include tables in the filtered list that match *pattern* and have been previously selected (that is, the local list tables) along with tables that have not been selected. In the filtered list, each previously

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

selected, local list table is displayed with a check mark in its check box. Each filtered list table that was not previously selected has no check mark in its check box. By default when the asterisk is omitted, only tables that have not been previously selected are returned in the filtered list. Using the asterisk is useful for removing currently selected tables from the local list.

The wildcard pattern definitions and examples can be seen from a help screen displayed by clicking the secondary mouse button on the Filter Pattern text field of the Wildcard Selector dialog box:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

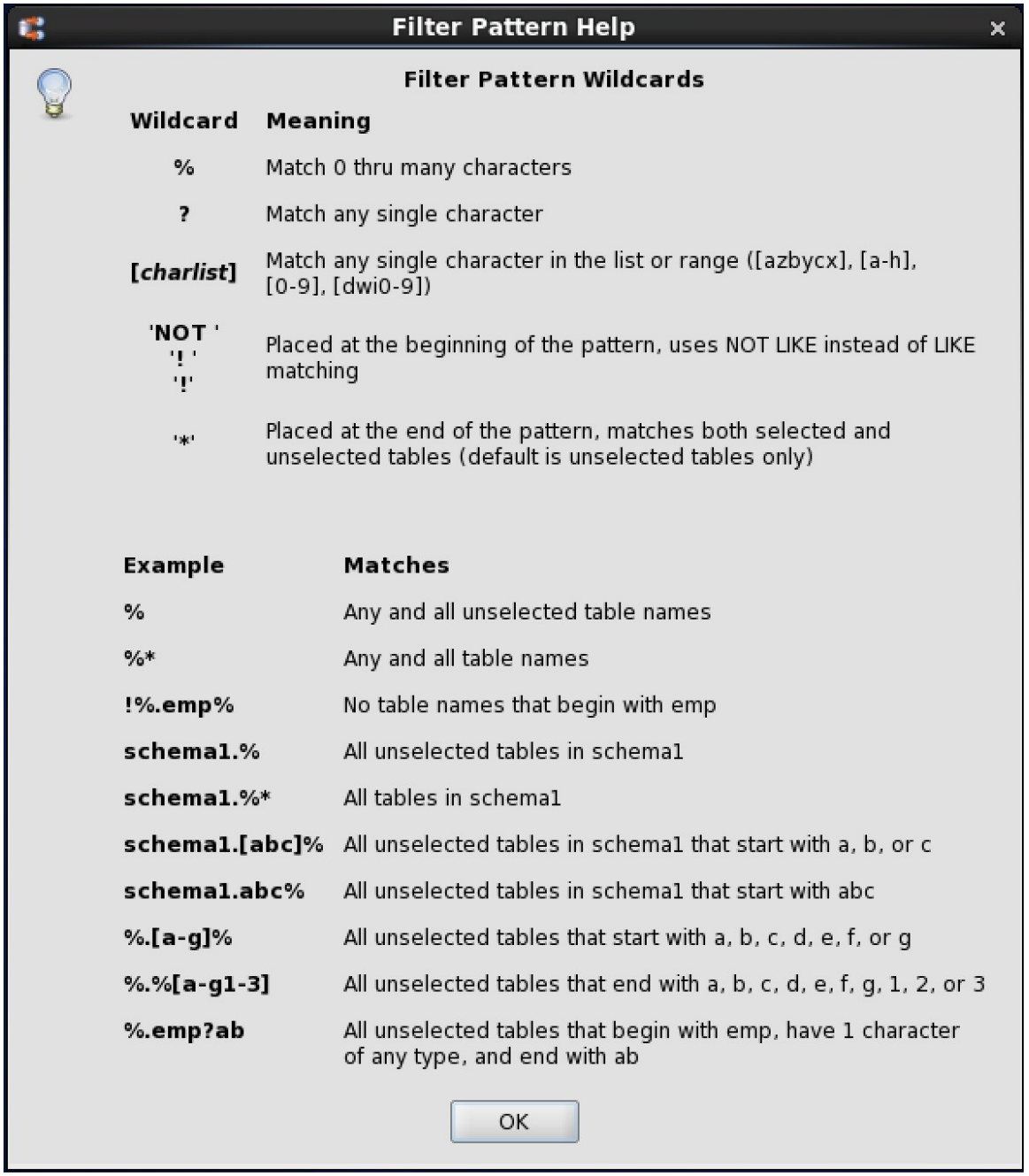


Figure 7-1 - Filter Pattern Help screen

The following section describes the basic steps for using the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.1.2 Using the Wildcard Selector

This section describes the basic process of using the wildcard selector. The following terms are used in the Wildcard Selector dialog box and the description of the wildcard selector feature:

- **Calling Dialog Box.** This is the dialog box of the operation from which you invoke the Wildcard Selector dialog box. The final set of tables from the wildcard selector is applied to the operation managed by the calling dialog box. Possible calling dialog boxes are the Create Publication dialog box (see Section 5.2.3 for a single-master replication system or Section 6.2.3 for a multi-master replication system), the Add Tables dialog box (see Section 7.6.3.1), and the Remove Tables dialog box (see Section 7.6.3.2).
- **Table List.** This is the list of currently selected tables displayed in the calling dialog box. Each selected table has a check mark in its check box.
- **Local List.** This is a temporary, internal copy of the table list managed by the wildcard selector. The wildcard selector allows you to add tables to the local list and to remove tables from the local list. When you click the Done button of the Wildcard Selector dialog box, the local list becomes the table list. In other words, the local list tables appear as the selected tables of the calling dialog box.
- **Unselected Tables.** These are the tables eligible for, but have not been selected for the operation with which you are using the wildcard selector. When you click the Filter List button, the unselected tables that match the filter pattern are listed in the Available Tables field of the Wildcard Selector dialog box. To list all unselected tables, use the percent sign (%) for the filter pattern.
- **Selected Tables.** These are the tables you have selected for the operation with which you are using the wildcard selector. That is, these are the tables comprising the local list. To display selected tables that match a filter pattern, add the asterisk character (*) immediately after the filter pattern. Each selected table has a check mark in its check box.

The following describes the steps for using the wildcard selector.

Step 1: Prior to opening the Wildcard Selector dialog box, you may start selecting tables from the list of available tables of the calling dialog box by adding a check mark to the check box of each such table.

From the calling dialog box, click the Use Wildcard Selection button to open the Wildcard Selector dialog box.

The tables that you have preselected are included in the local list used by the wildcard selector to manage the addition or removal of tables.

For example, the following is the Create Publication dialog box from which the wildcard selector can be used:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

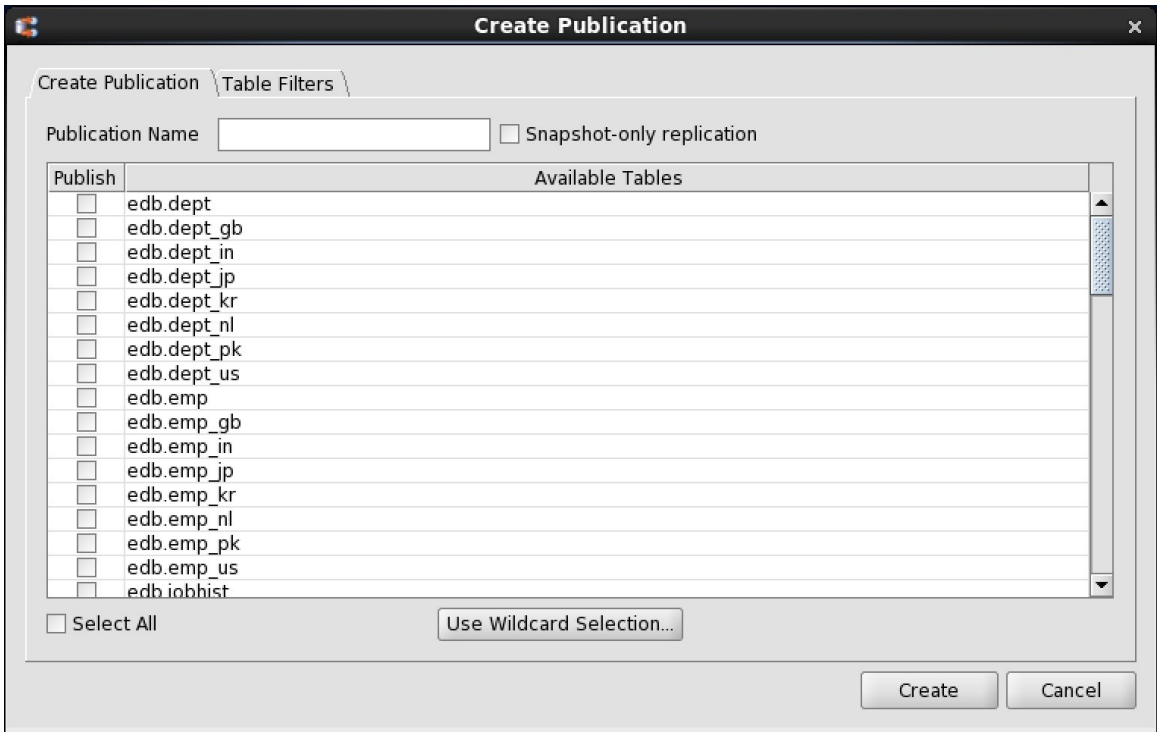


Figure 7-2 – Invoking the wildcard selector from a calling dialog box

Step 2: The Available Tables field displays the filtered list matching the pattern used in the Filter Pattern text field.

When the Wildcard Selector dialog box is initially opened, the default filter pattern is the percent sign (%), which returns all eligible, unselected tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

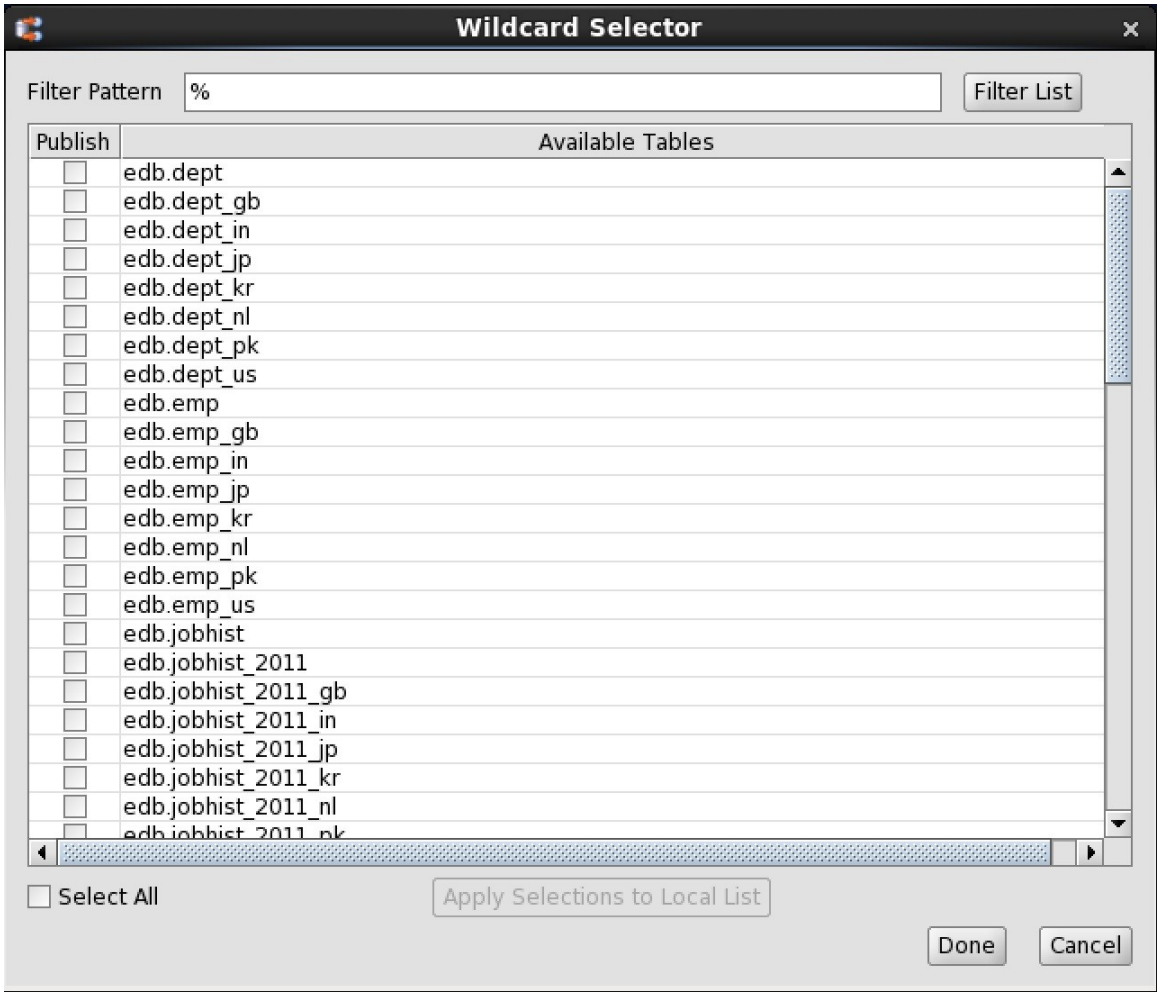


Figure 7-3 - Wildcard Selector dialog box

Step 3: Enter a pattern in the Filter Pattern text field to narrow down your desired table selection. Click the Filter List button to display the tables that match the pattern.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

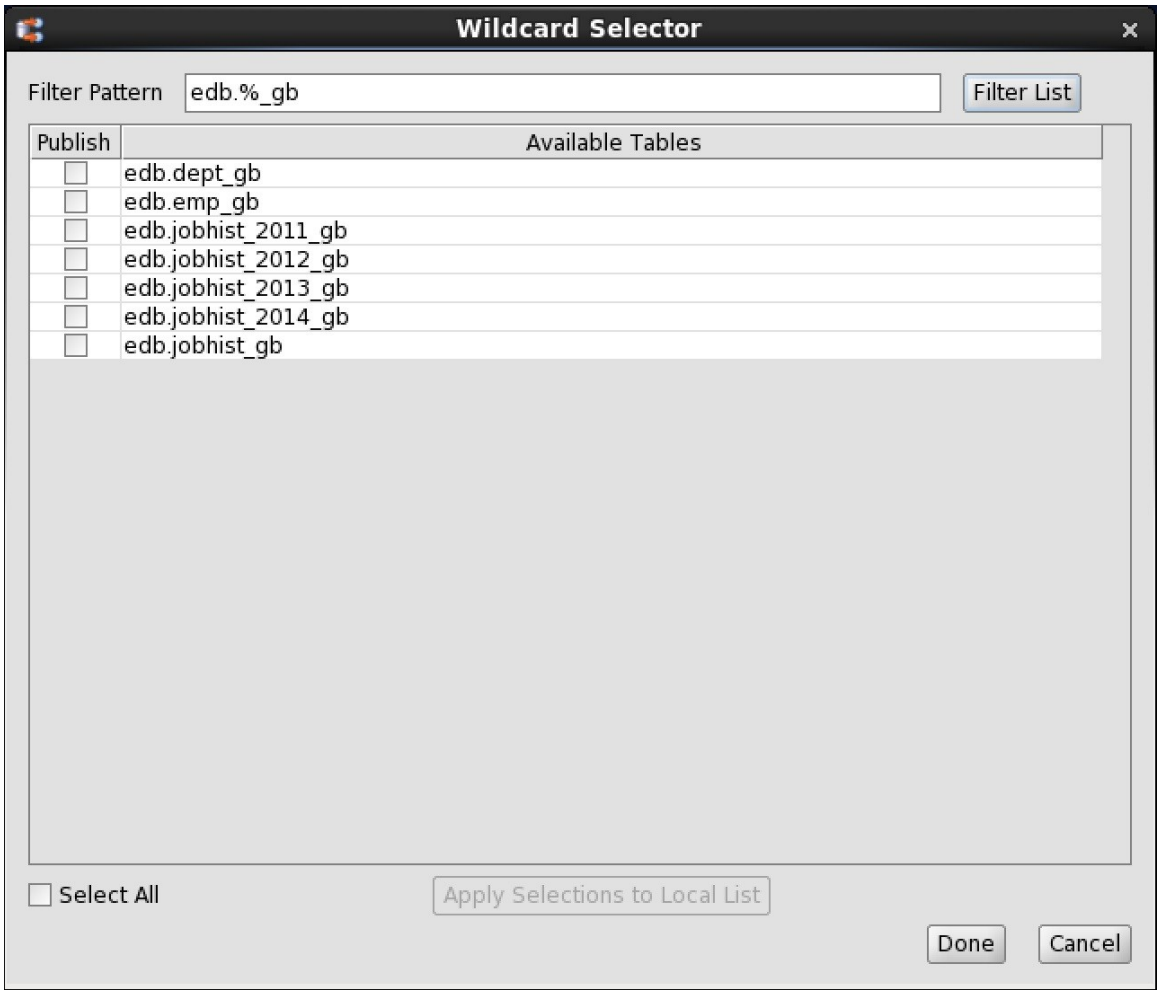


Figure 7-4 - Tables matching a filter pattern

Step 4: Select tables from the Available Tables list that you want to add to the local list by placing a check mark in each such table's check box. You can also click the Select All check box to select all tables and then individually deselect certain tables by removing its check mark.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

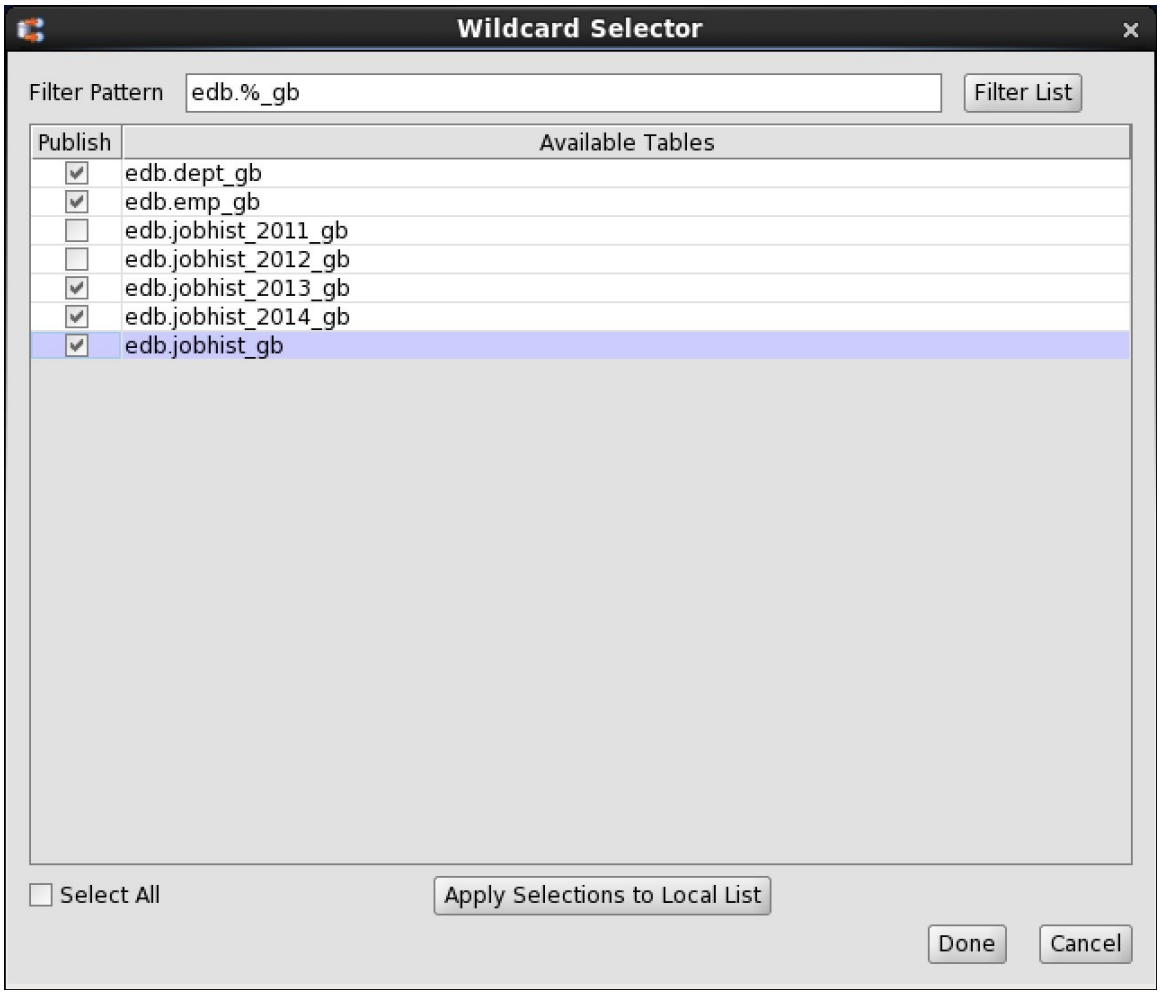


Figure 7-5 - Tables selected for the local list

Step 5: Click the Apply Selections to Local List button to add the selected tables to the local list.

The following example shows that the selected tables have been removed from the Available Tables list after the Apply Selections to Local List button was clicked since they are no longer unselected.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

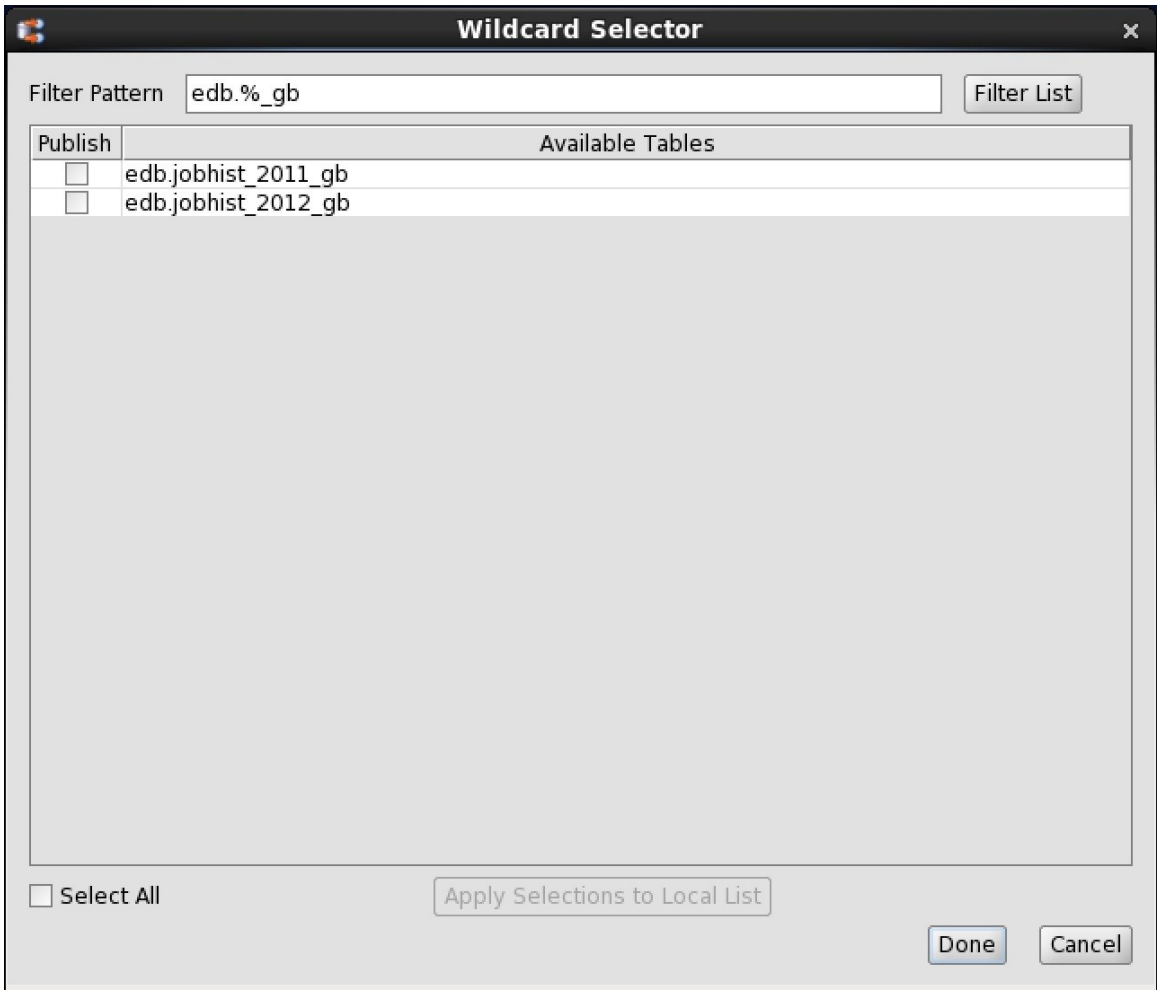


Figure 7-6 - Selected tables added to the local list

Note: You can click the Cancel button at any time to terminate the wildcard selector without applying the local list changes to the table list of the calling dialog box.

Step 6: As many times as desired, repeat steps 3 through 5 using the filter patterns needed to add all of your desired tables to the local list.

The following example shows a second filter pattern and the returned filter list.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

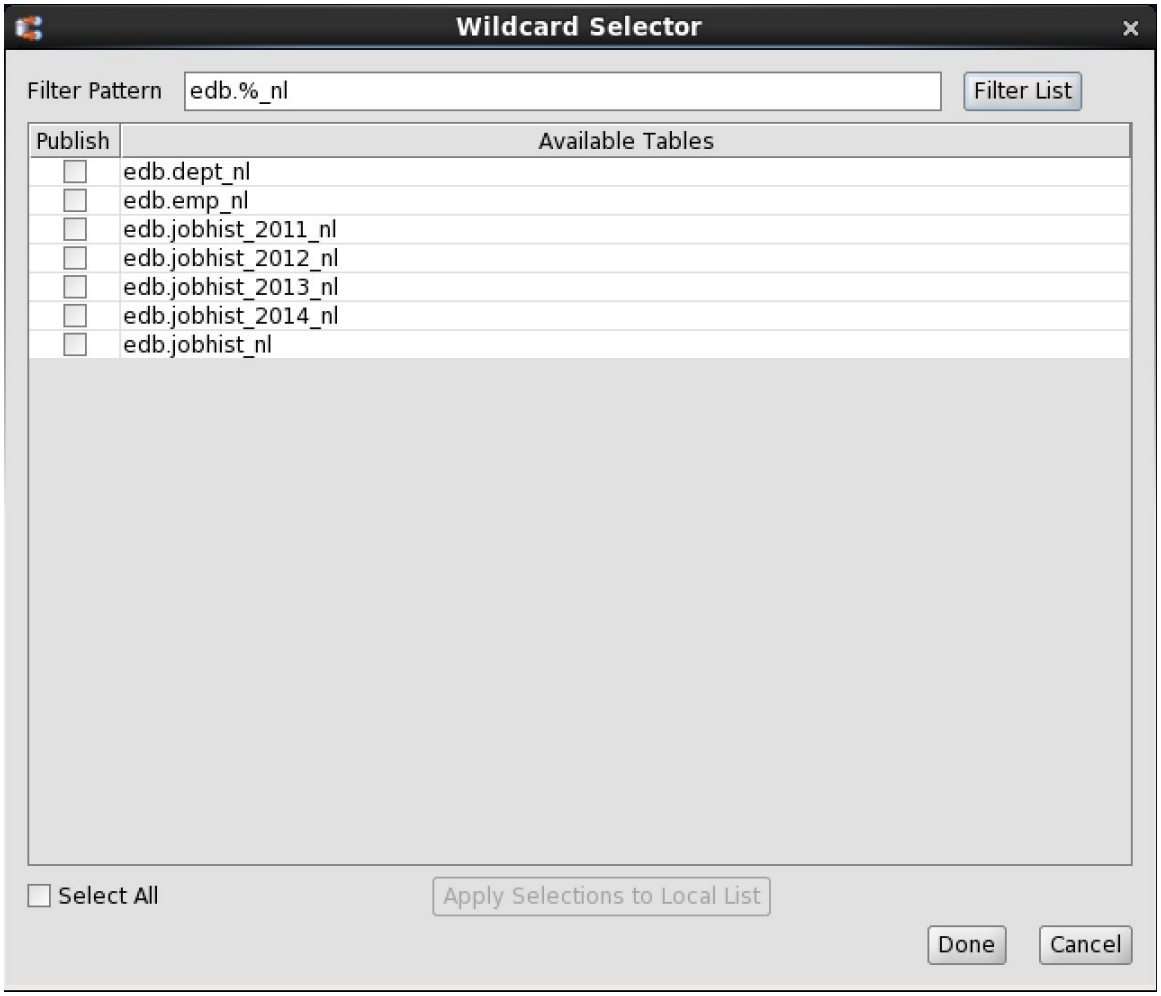


Figure 7-7 - Tables matching a second filter pattern

All tables are then selected from this filtered list by clicking the Select All check box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

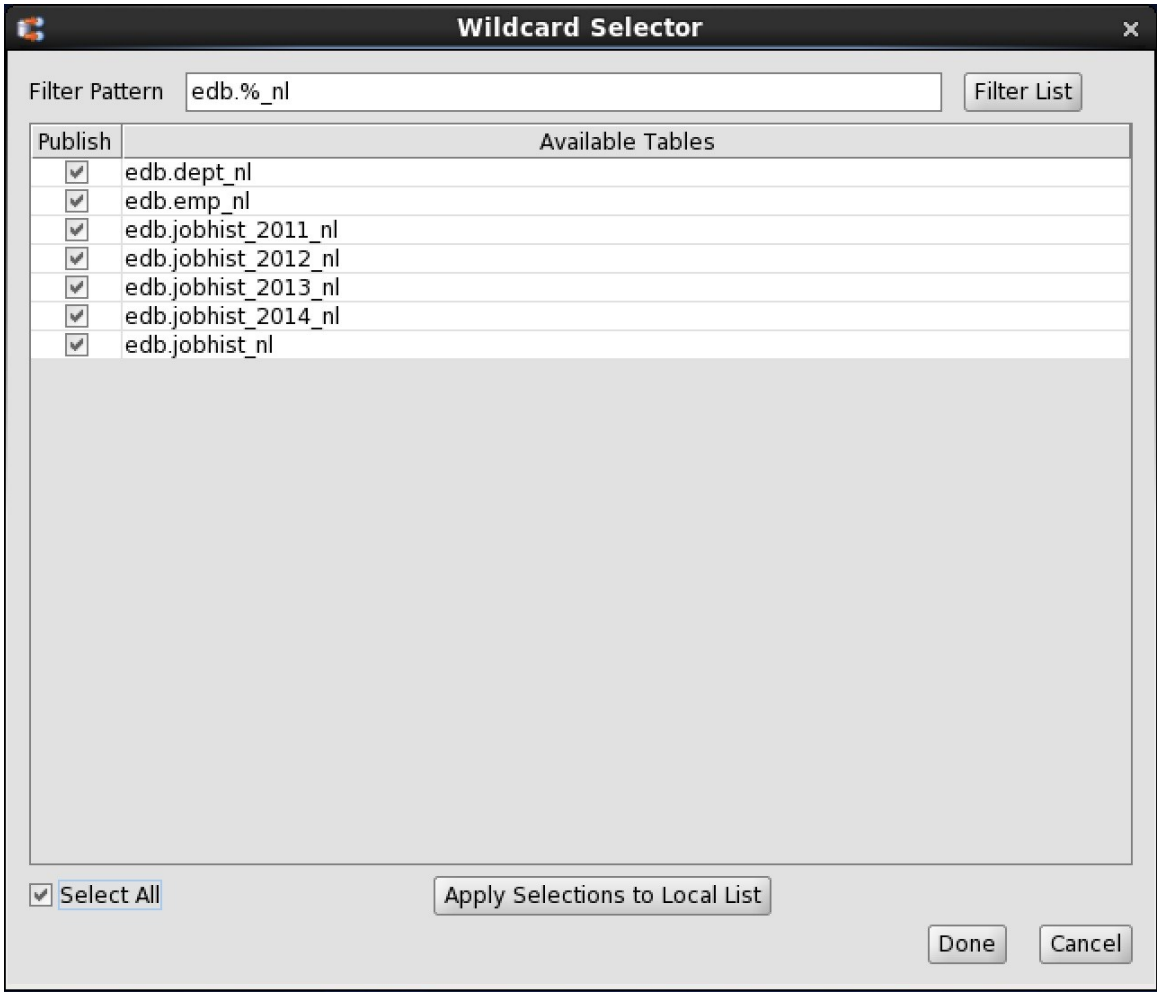


Figure 7-8 - Select all tables

The Apply Selections to Local List button is clicked to add all tables to the local list. After applying the selections, there are no unselected tables remaining that match the filter pattern.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Figure 7-9 – All filter list tables added to the local list

By using the asterisk after the pattern, you can display previously selected tables comprising the local list. Each selected table has a check mark its check box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

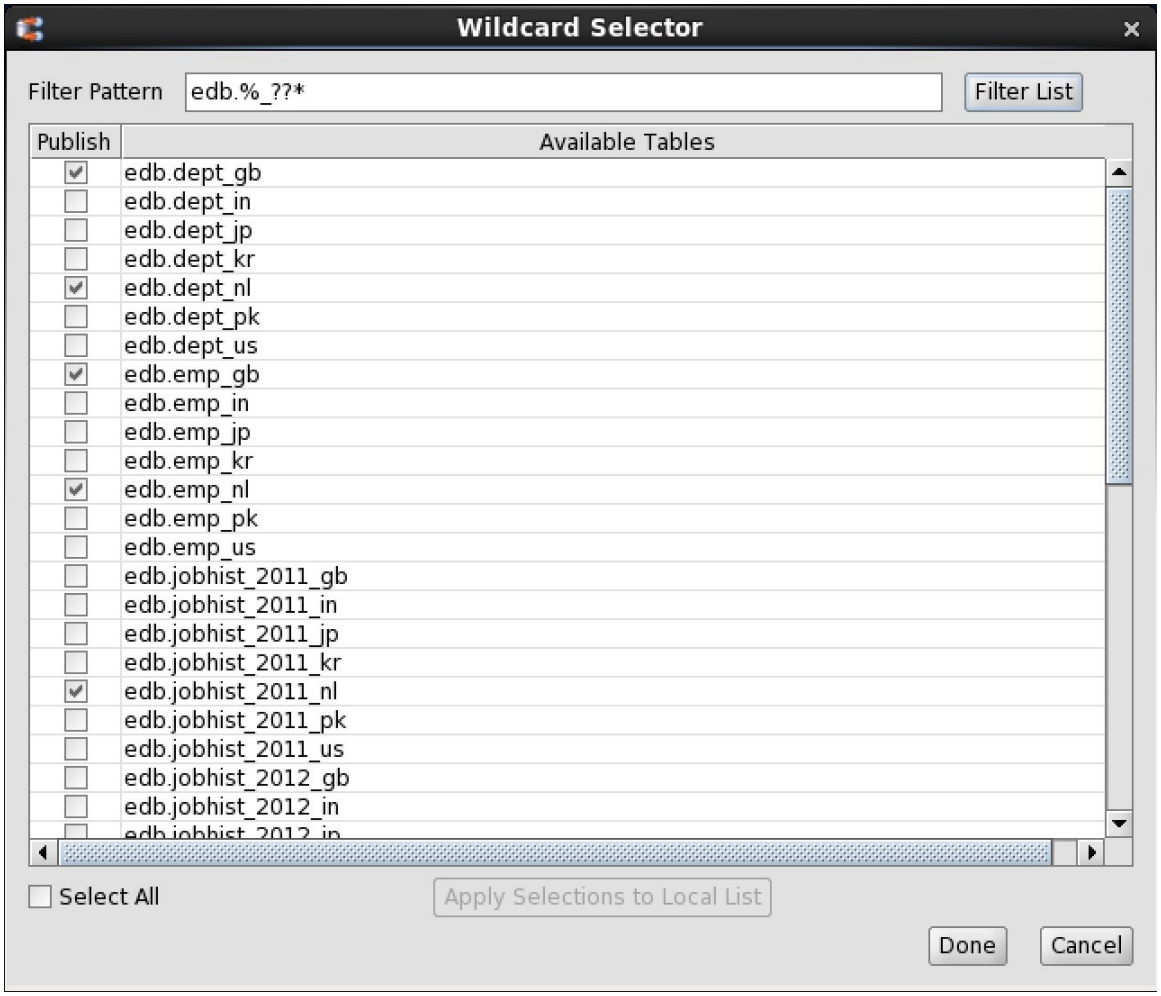


Figure 7-10 - Display selected and unselected tables matching the filter pattern

You can remove selected tables from the local list by clicking on each such table's check box to remove the check mark.

The following filter pattern includes the tables to be removed from the local list.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

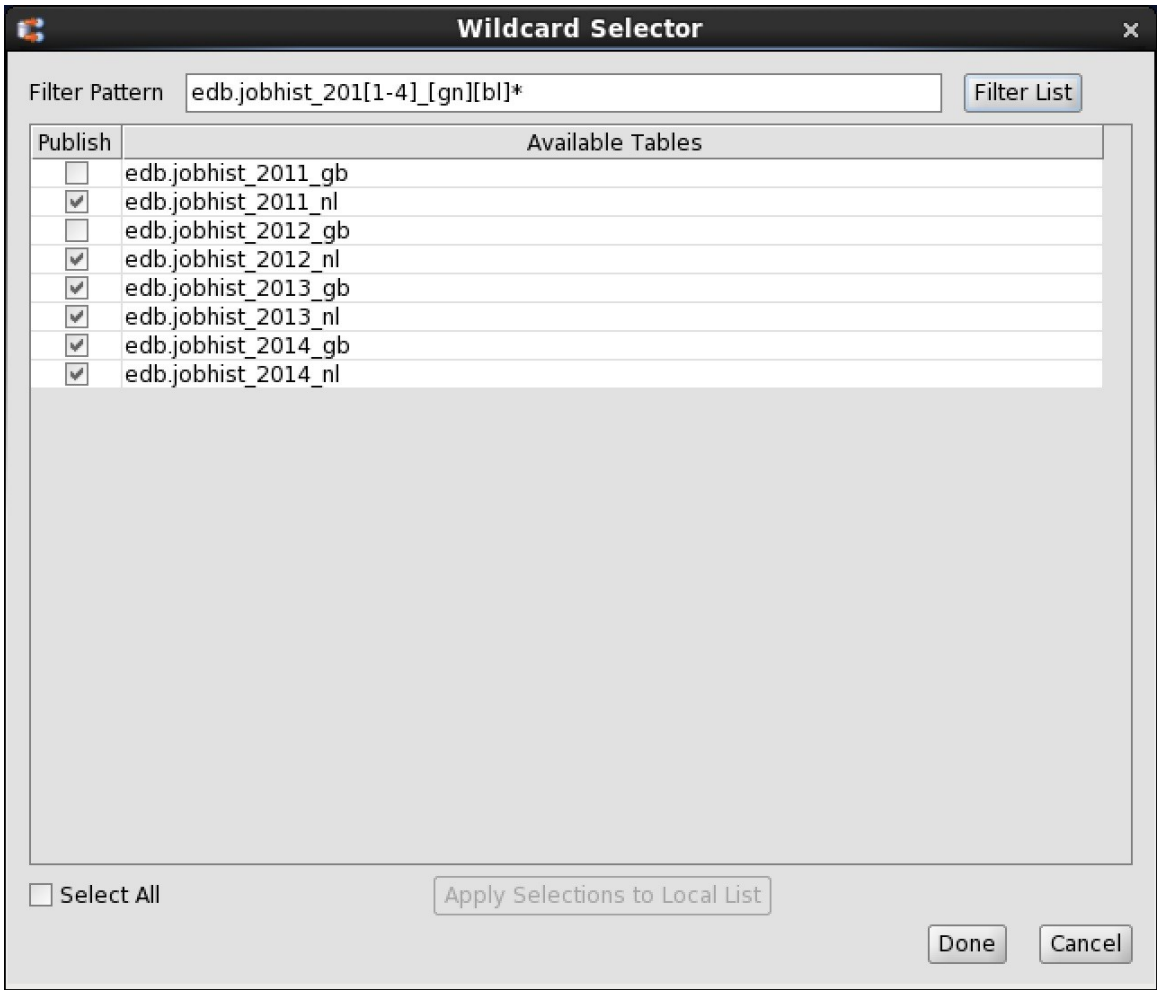


Figure 7-11 - List selected tables to be removed

The check marks are removed from the selected tables to be removed from the local list.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

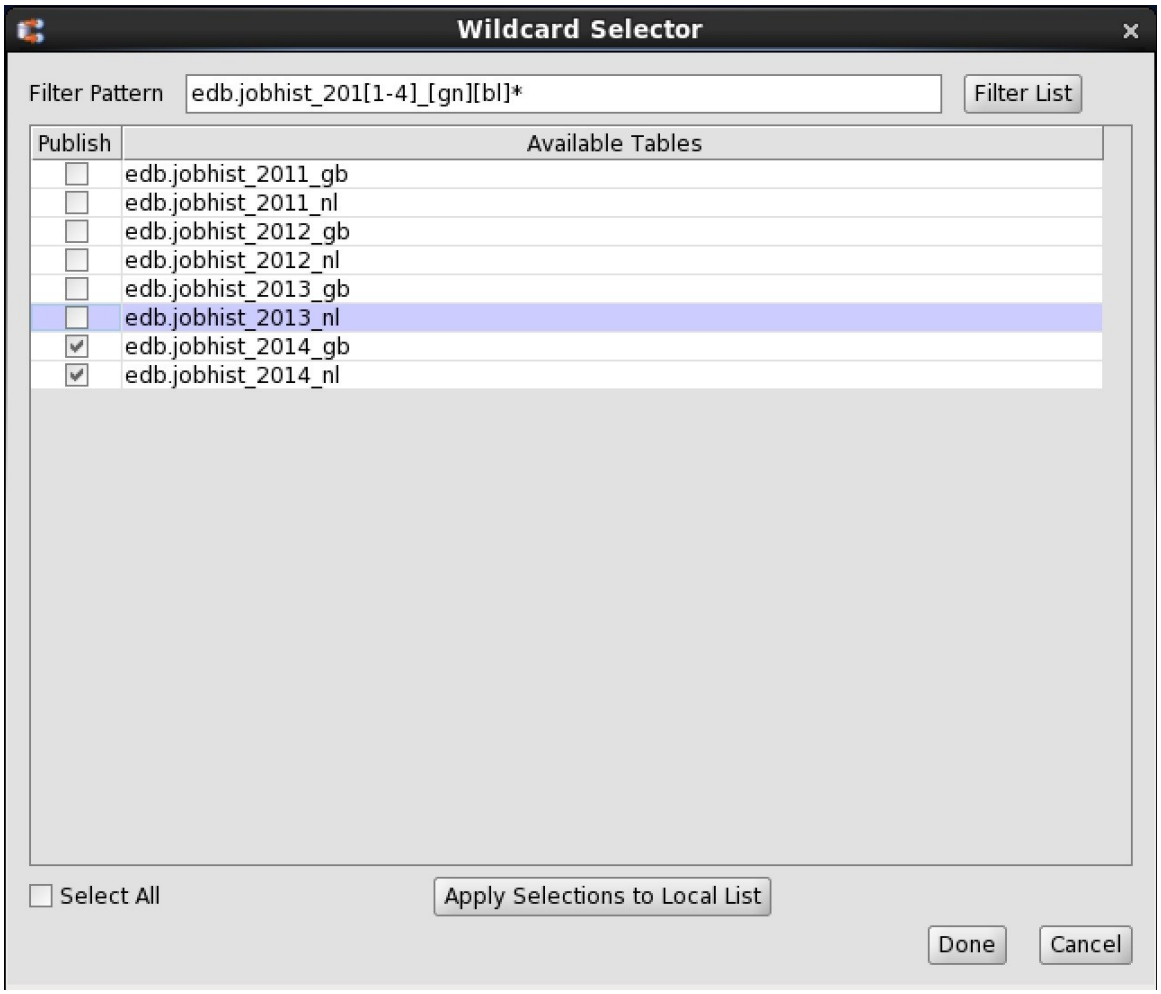


Figure 7-12 - Deselect tables to be removed from the local list

The removal of the deselected tables from the local list occurs along with the addition of any newly selected tables when you click the Apply Selections to Local List button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

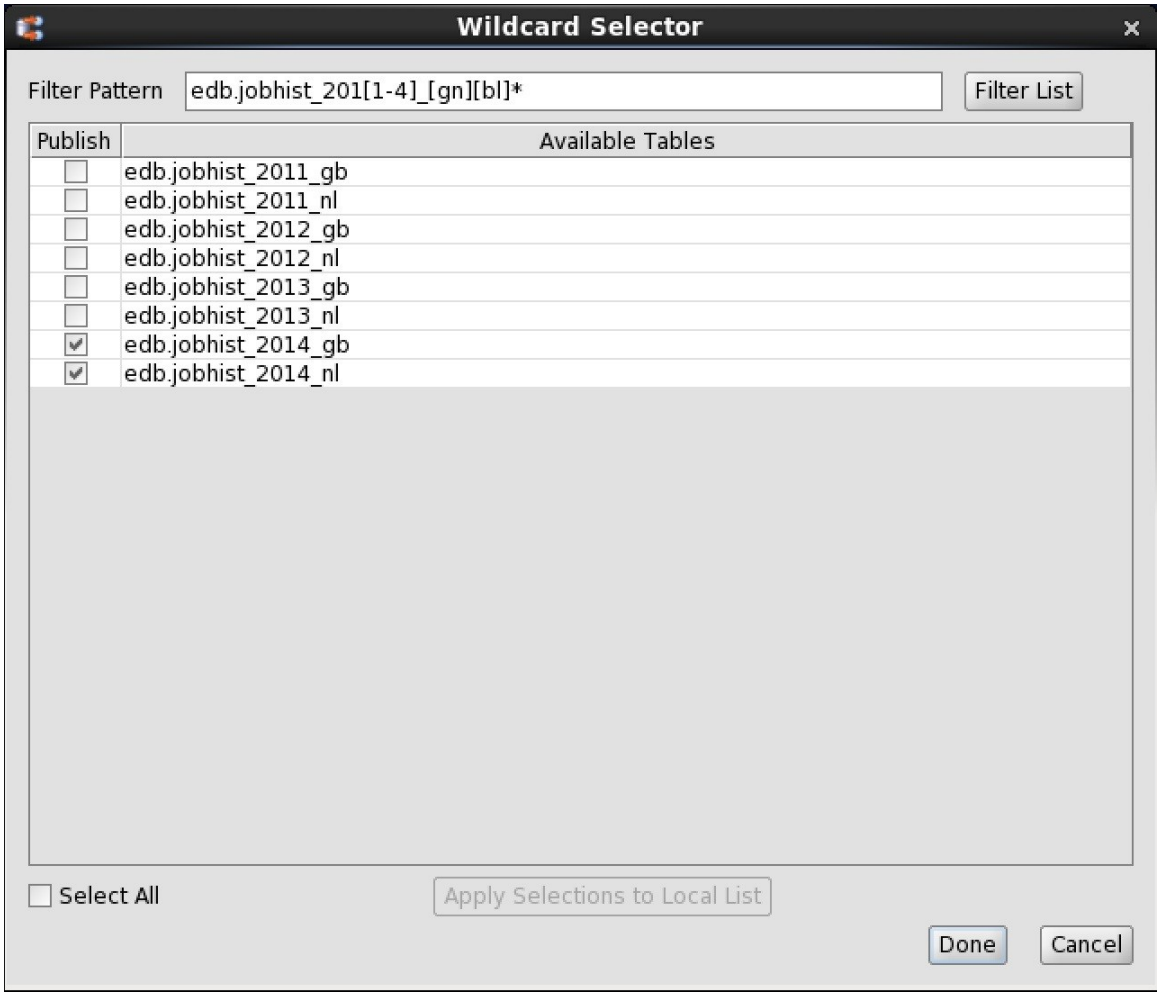


Figure 7-13 - Deselected tables removed from the local list now shown as unselected

The deselected tables still appear in the Available Tables list since they still match the pattern, but as unselected tables (that is, with no check mark in each such table's check box).

Step 7: When the local list contains all of your desired, selected tables, click the Done button. The Wildcard Selector dialog box closes, and the local list becomes the list of selected tables displayed by the calling dialog box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

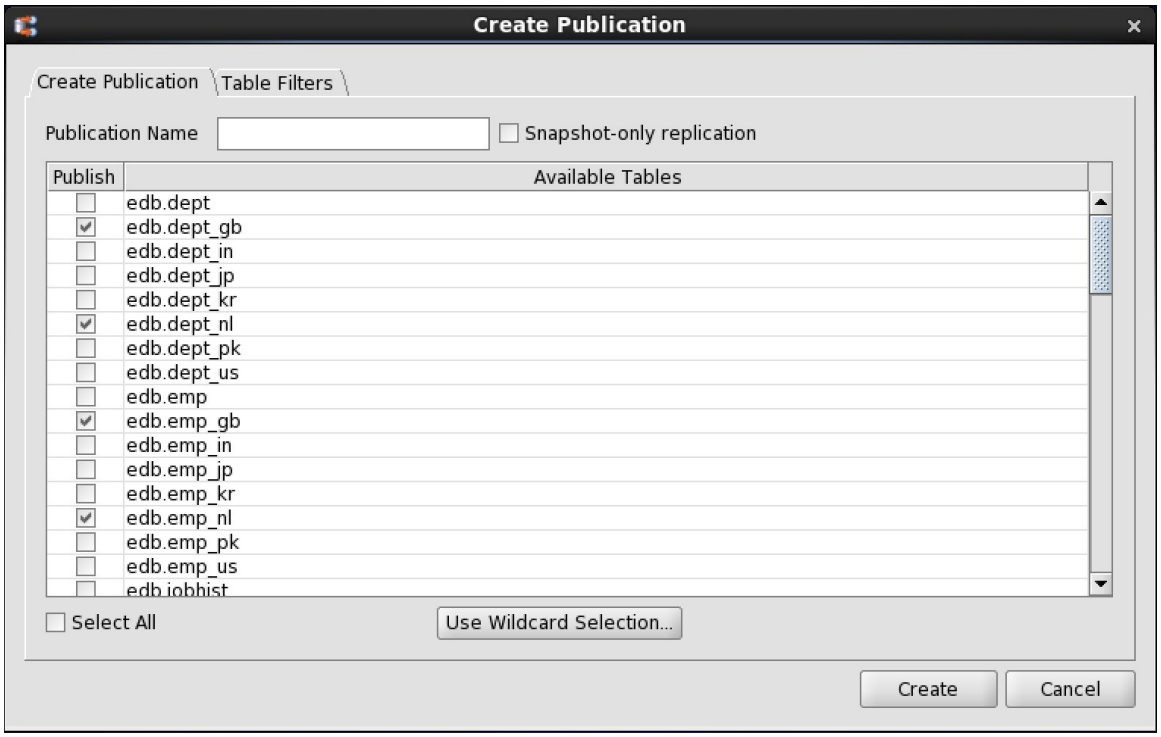


Figure 7-14 - Create Publication calling dialog box with applied local list

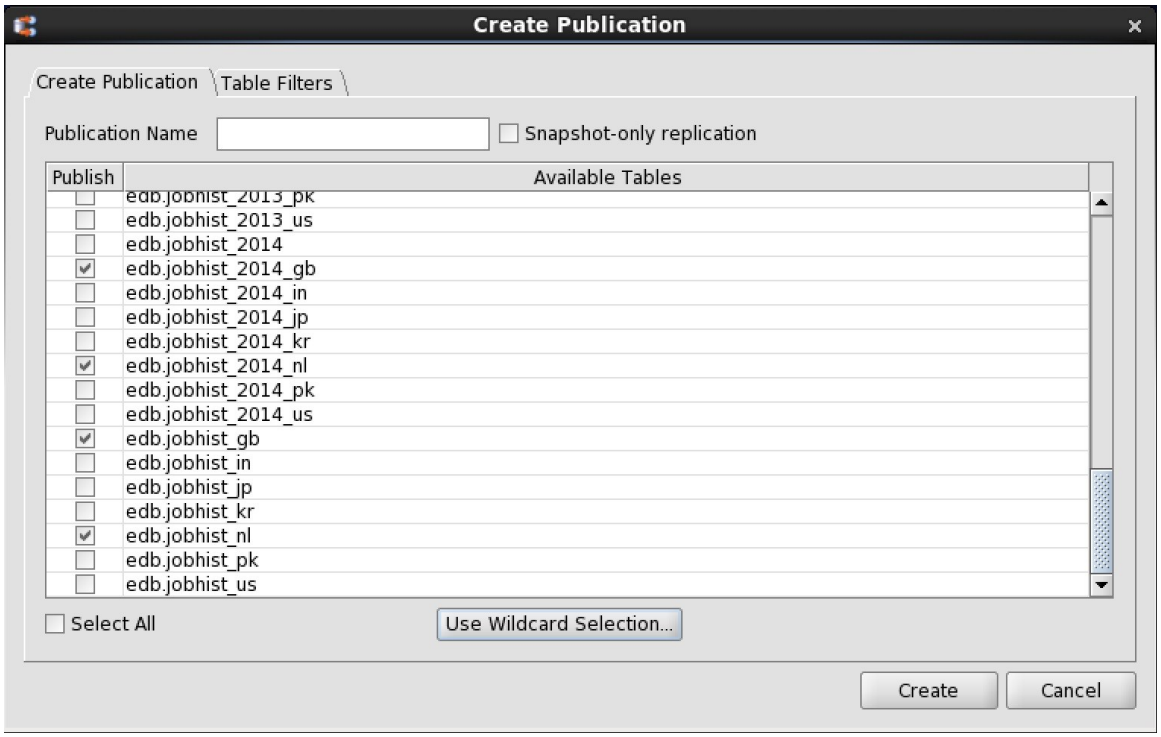


Figure 7-15 - Create Publication calling dialog box with applied local list (continued)

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Alternatively, if you decide that you do not wish to apply the local list, click the Cancel button. The local list changes are discarded and the table list of the calling dialog box remains unchanged.

Step 8: You can invoke the wildcard selector again and repeat the process to add tables to, or remove tables from the table list by beginning with Step 1.

The following example verifies that if you were to invoke the wildcard selector a second time, the local list includes the table list created from the prior closure of the wildcard selector.

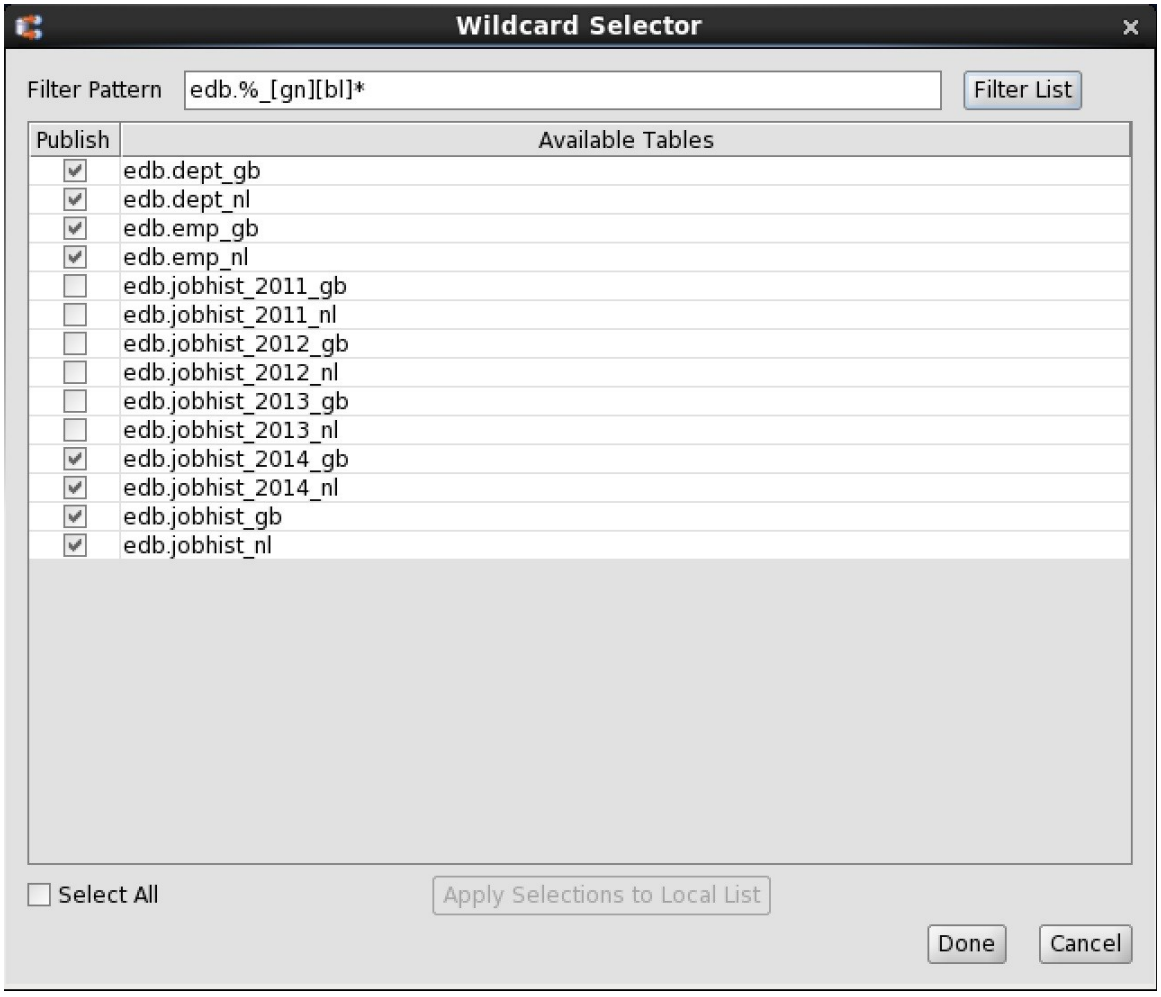


Figure 7-16 - Local list includes selected tables from the calling dialog box table list

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 9: When the calling dialog box contains the complete list of your desired tables, click the appropriate button of the calling dialog box to complete the operation with the selected tables.

The following shows the publication created from the selected tables.

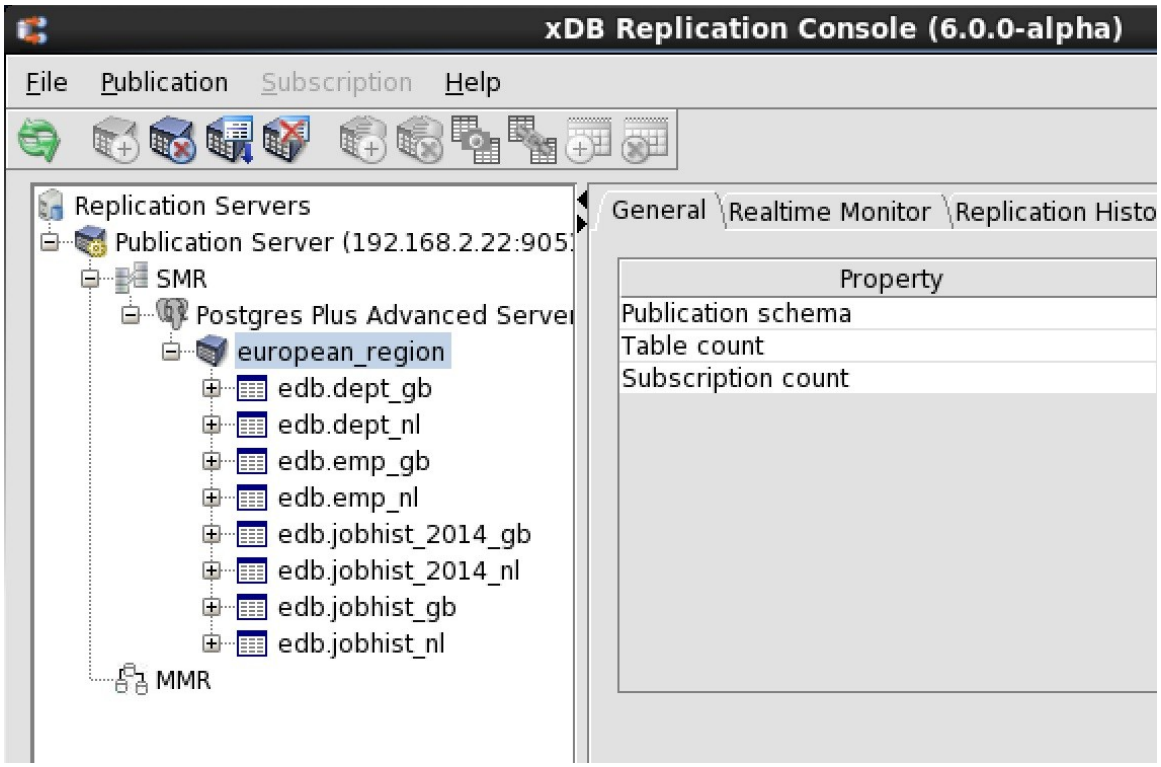


Figure 7-17 - Publication created with the selected tables

7.2 Creating a Schedule

A *schedule* establishes recurring points in time when replication is to occur.

Note (For MMR only): Be sure an initial snapshot replication has been performed from the master definition node to every other master node in the multi-master replication system. If a newly added master node did not undergo an initial snapshot, any subsequent synchronization replication initiated by a schedule may fail to apply the transactions to that master node. The initial snapshot could be taken when the master node is first added (see Section 6.3) or by performing an on demand snapshot (see Section 6.5.1).

In a single-master replication system, once a schedule is created the subscription server initiates replications according to the schedule until either the schedule is changed or

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

removed. In a multi-master replication system, the publication server handles this process.

See Section 7.3 for changing or removing a schedule.

When a scheduled replication is to take place, all components of the replication system must be running:

- Publication database server
- Subscription database server (applies only to single-master replication systems)
- Publication server
- Subscription server (applies only to single-master replication systems)

If any of the preceding components are not running at the time of a scheduled replication, then replication does not occur at that point in time. The replication occurs at the next scheduled replication time when all applicable replication system components are running.

For synchronization replications with the trigger-based method, changes that have occurred on the source tables that were not replicated due to a skipped, scheduled replication are maintained as pending transactions in the shadow tables of the source database.

For synchronization replications with the log-based method, changes that have been extracted from the WAL files to in-memory structures, but have not been applied are persisted using Java object serialization to files on the host running the publication server.

All changes since the last successful replication are applied whenever the next scheduled replication occurs. Thus, accumulated changes are never lost due to a missed replication.

For snapshot replications, skipped, scheduled replications present no problems since a snapshot replication replaces all of the data in the target tables with the current source data.

Step 1 (For SMR only): Select the Subscription node of the subscription for which you wish to create a schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

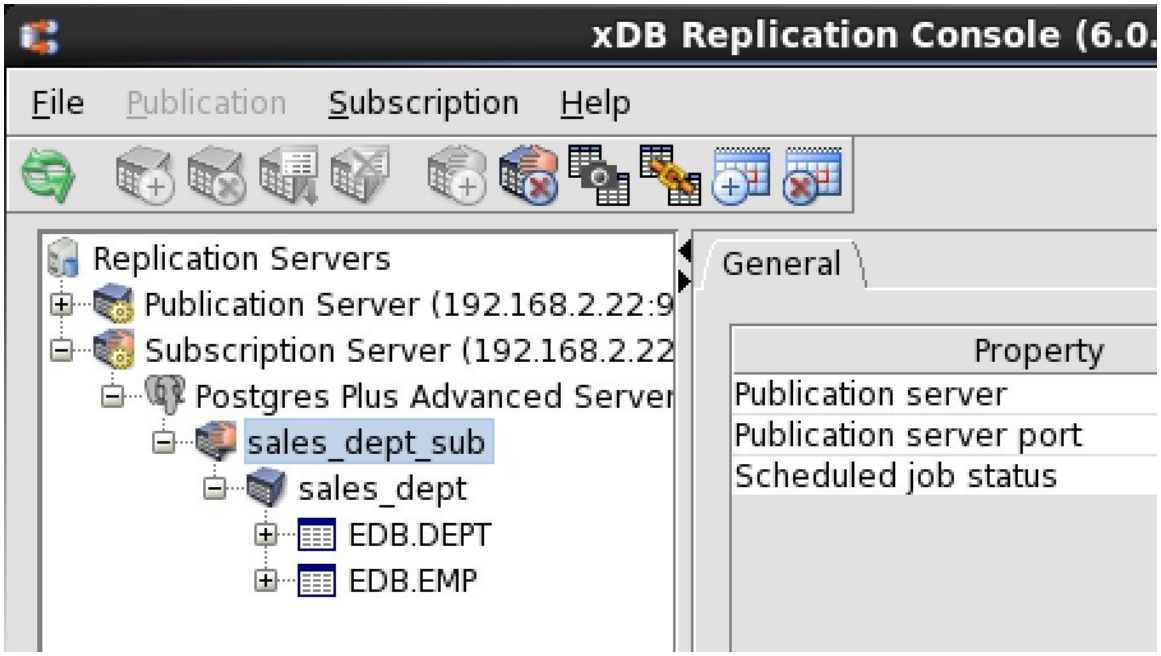


Figure 7-18 - Selecting a subscription on which to set a schedule

Step 1 (For MMR only): Select the Publication Database node designated as the controller database. (The Controller database field in the Property window is set to `Yes` for the controller database.)

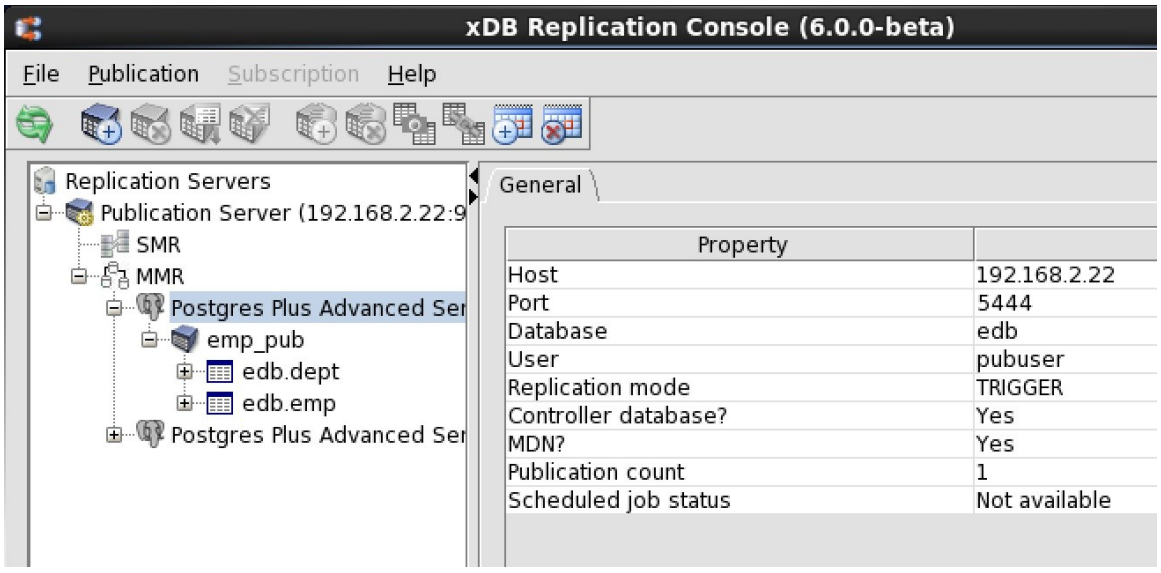


Figure 7-19 - Selecting the controller database on which to set a schedule

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2 (For SMR only): Open the Scheduled Task Wizard dialog box in any of the following ways:

- From the Subscription menu, choose Schedule, then Configure Schedule.
- Click the secondary mouse button on the Subscription node and choose Configure Schedule.
- Click the primary mouse button on the Configure Schedule icon.

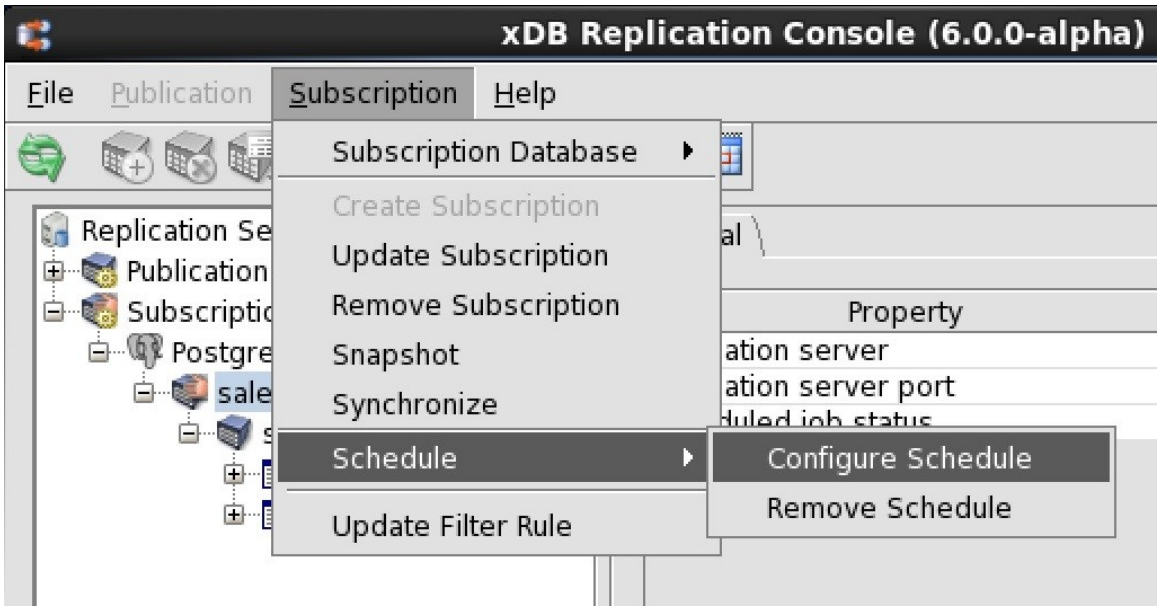


Figure 7-20 - Opening the Scheduled Task Wizard dialog box on a subscription

Step 2 (For MMR only): Open the Scheduled Task Wizard dialog box in any of the following ways:

- Click the secondary mouse button on the Publication Database node and choose Configure Schedule.
- Click the primary mouse button on the Configure Schedule icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

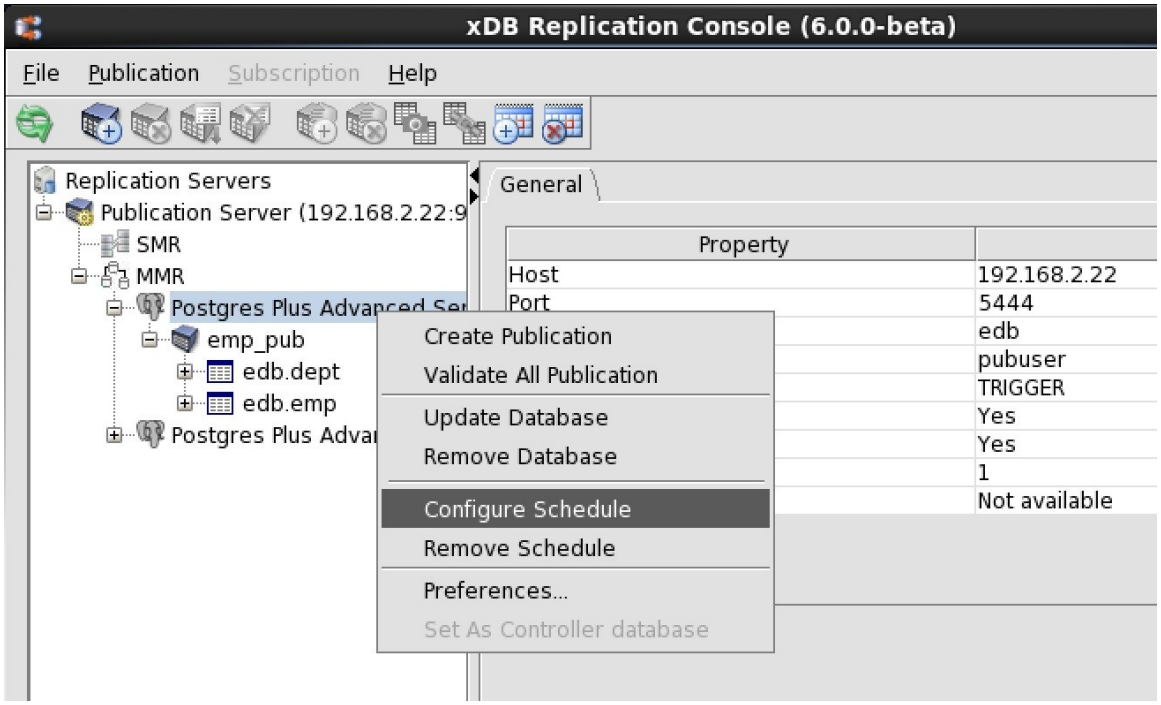


Figure 7-21 - Opening the Scheduled Task Wizard dialog box on the controller database

Step 3: In the Scheduled Task Wizard dialog box, select the radio button for either synchronization replication or snapshot replication.

Note: If the publication associated with this subscription is a snapshot-only publication, then only Snapshot may be chosen.

Note: In a multi-master replication system, only Synchronize may be chosen.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

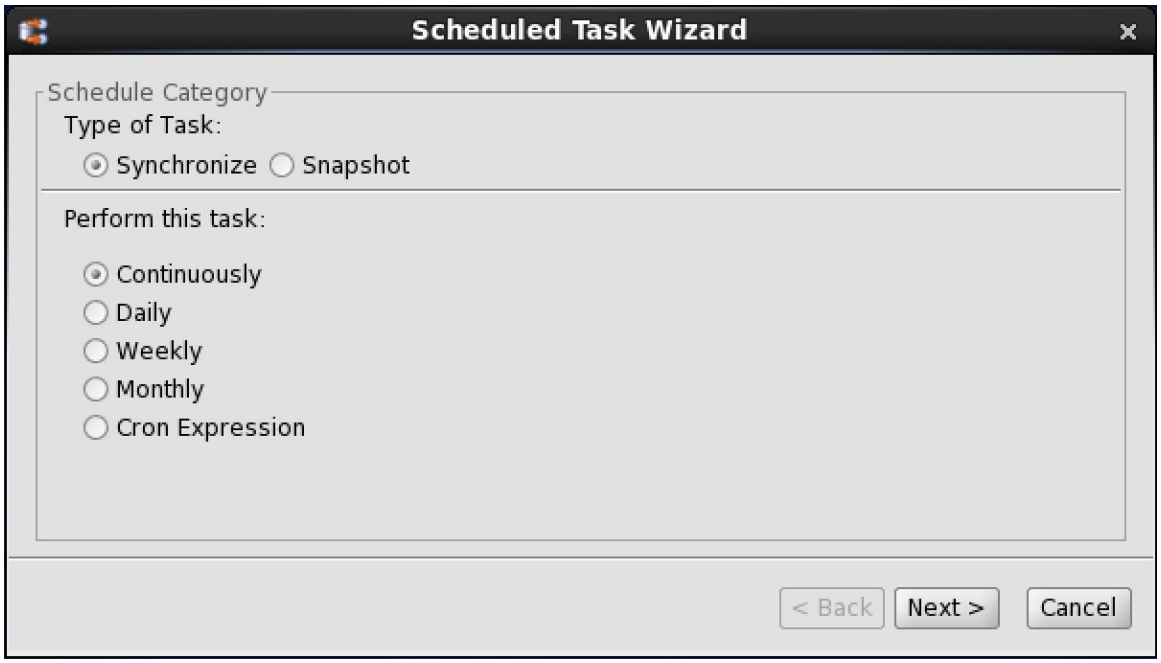


Figure 7-22 - Scheduled Task Wizard dialog box

Step 4: Select the radio button for the scheduled replication frequency, or select Cron Expression to write your own cron expression. The frequency choices have the following meanings:

- **Continuously.** Schedules replication to run continuously at an interval in seconds that you specify. Select this option if the source tables change frequently during the day and the target tables must be kept up-to-date throughout the course of the day.
- **Daily.** Schedules replication to run once a day at the time you choose. Select this option if the target tables need to be refreshed daily.
- **Weekly.** Schedules replication to run once a day at the time you choose, but only on the specific days of the week you choose. Select this option if you need more flexibility than a daily schedule, and the target tables do not have to be refreshed every day.
- **Monthly.** Schedules replication to run one day per month on the day of the month and time you choose, but only on the specific months you choose. Select this option if updates to the source tables are not very frequent, and the target tables can be out-of-date by a month or more. The Monthly option allows you to schedule replication for as frequently as once a month or infrequently as once a year.
- **Cron Expression.** Provides additional flexibility for specifying a schedule beyond the four preceding radio button choices. See appendix Section 10.4.3 for directions on writing a cron expression.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following example shows the selection of a weekly schedule.

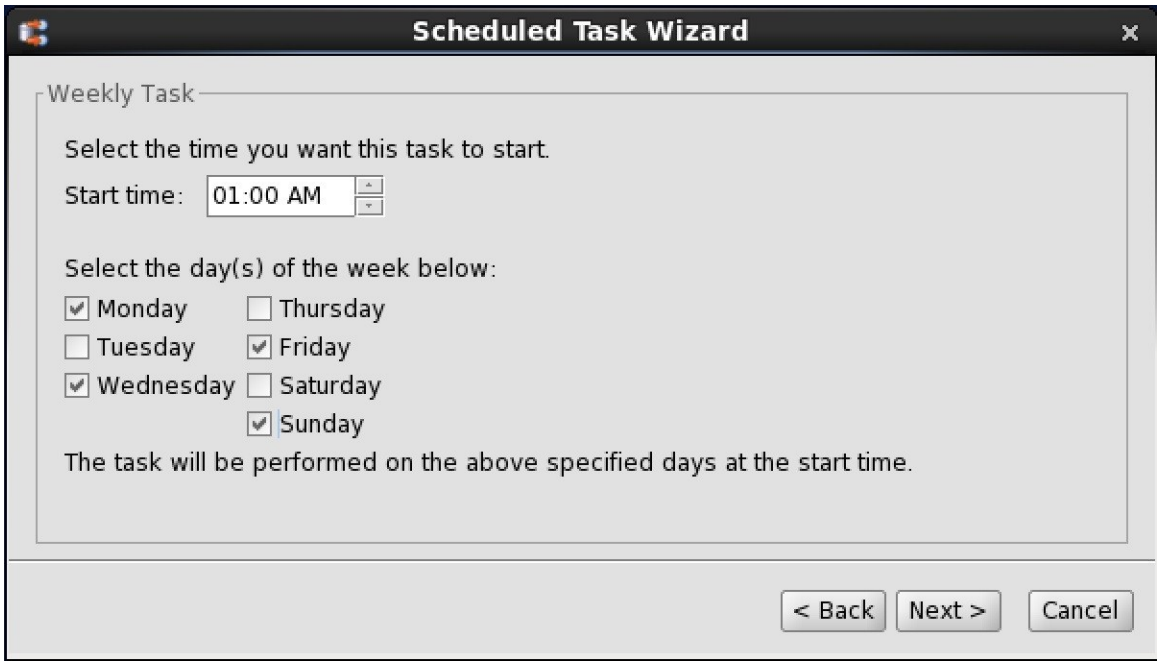


Figure 7-23 - Selecting a weekly schedule

Step 5: After completing the Scheduled Task Wizard dialog box, click the Next button.

Step 6: Your selected schedule will appear. Click the Finish button to accept the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

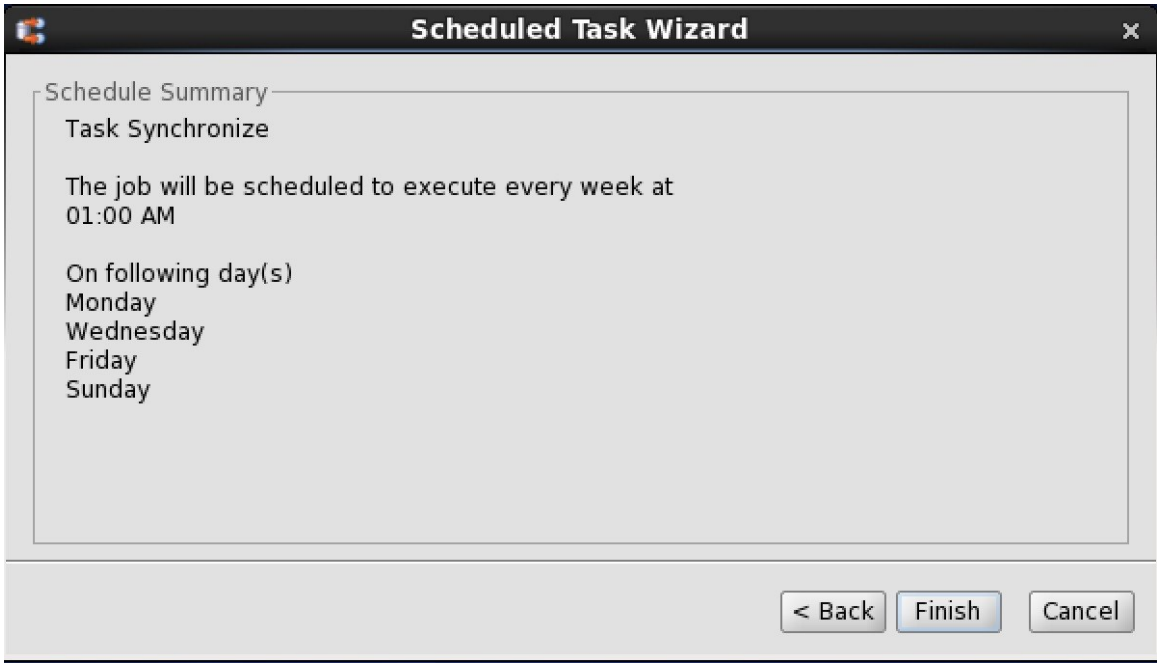


Figure 7-24 - Scheduled Task Wizard summary

If you click the Refresh icon, you will see the schedule properties in the General tab.

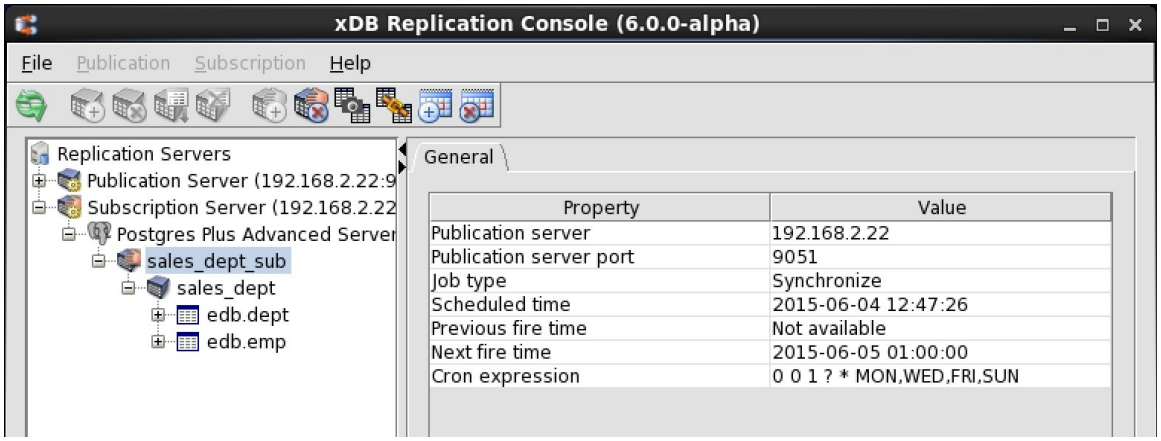


Figure 7-25 - Information window with the replication schedule

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.3 Managing a Schedule

Once a schedule has been created, xDB Replication Server performs replications according to the schedule until the schedule is updated or removed.

The updating or removal of a schedule has no effect on a replication that has already been started. If a replication is in progress when the schedule is updated or removed, the in progress replication continues until completion.

7.3.1 Updating a Schedule

The following steps illustrate how to change an existing schedule.

Step 1 (For SMR only): Make sure the subscription server whose node is the parent of the subscription you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 1 (For MMR only): Make sure the publication server whose node is the parent of the controller database you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Subscription node of the subscription for which you wish to update the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

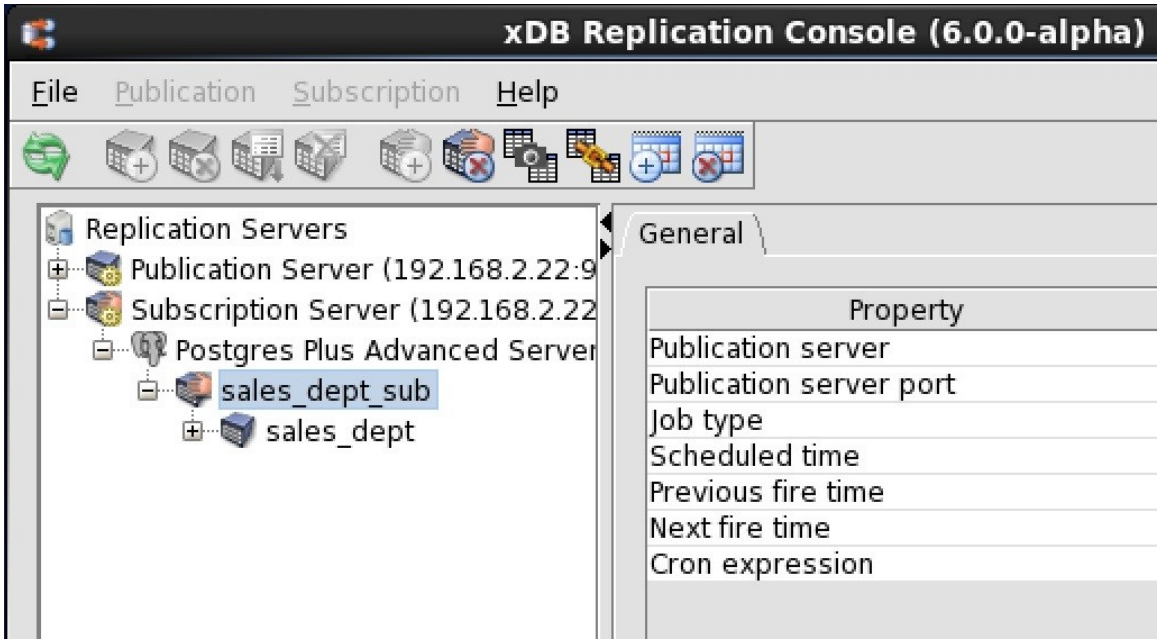


Figure 7-26 - Selecting a subscription whose schedule is to be updated

Step 2 (For MMR only): Select the Publication Database node designated as the controller database for which you wish to update the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

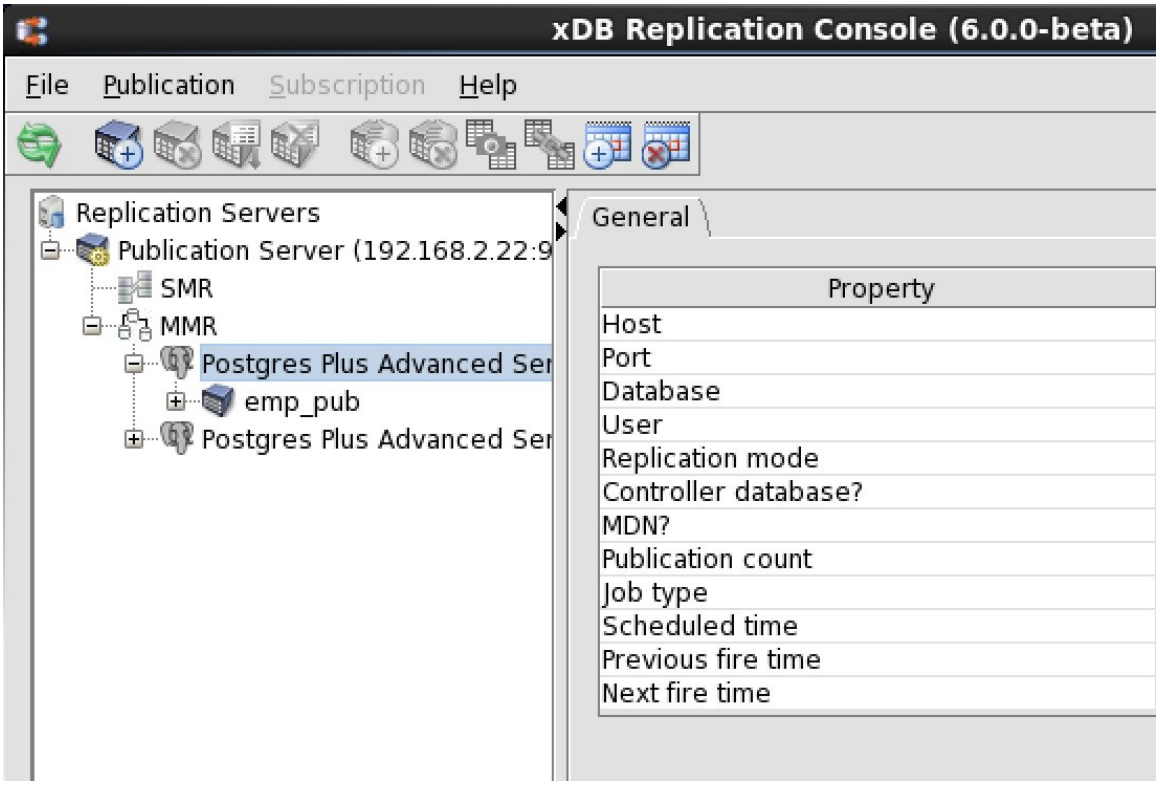


Figure 7-27 - Selecting the controller database whose schedule is to be updated

Step 3 (For SMR only): Open the Scheduled Task Wizard dialog box in any of the following ways:

- From the subscription menu, choose Schedule, then Configure Schedule.
- Click the secondary mouse button on the Subscription node and choose Configure Schedule.
- Click the primary mouse button on the Configure Schedule icon.

Step 3 (For MMR only): Open the Scheduled Task Wizard dialog box in any of the following ways:

- Click the secondary mouse button on the Publication Database node and choose Configure Schedule.
- Click the primary mouse button on the Configure Schedule icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

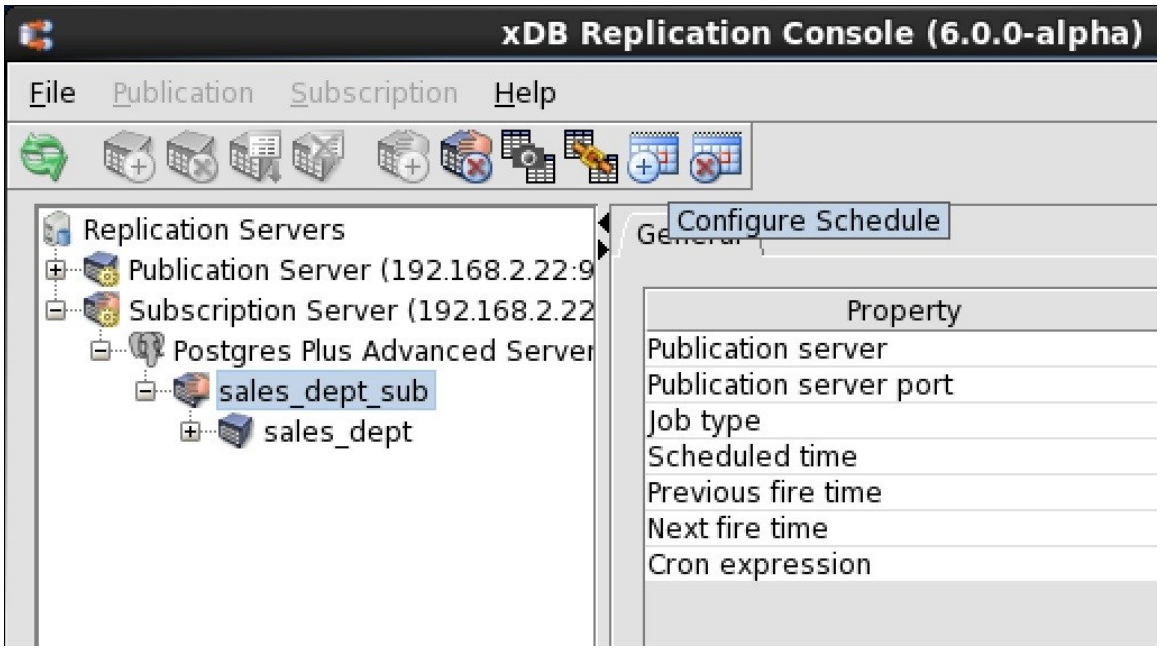


Figure 7-28 - Opening the Scheduled Task Wizard dialog box from the tool bar

Step 4: The Configure Scheduler confirmation box appears. Click the Yes button.

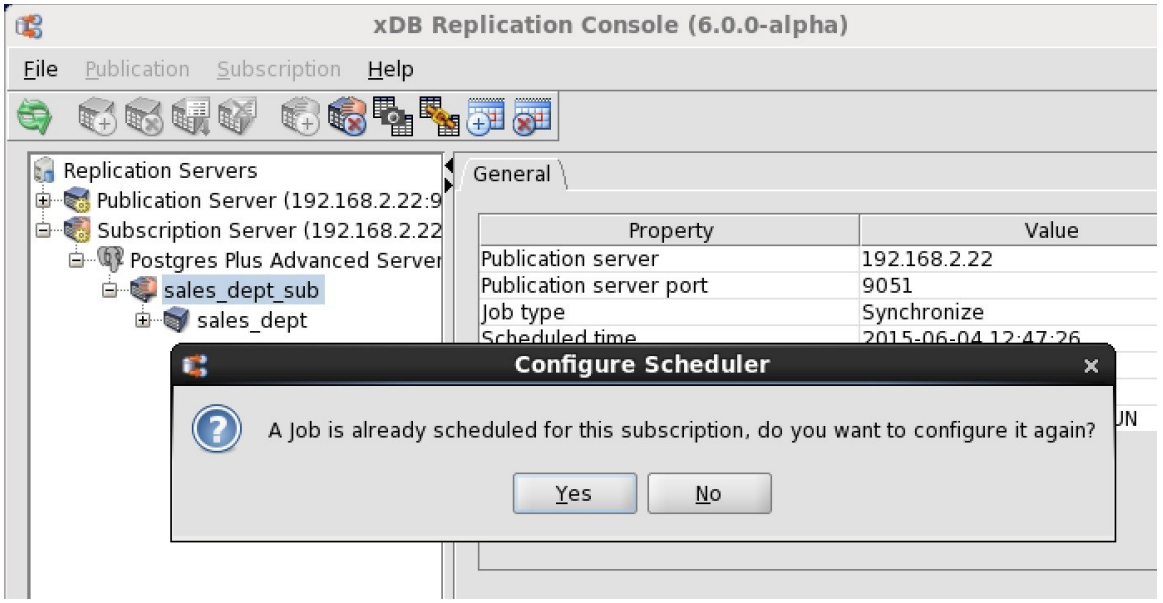


Figure 7-29 - Configure Scheduler confirmation

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 5: In the Scheduled Task Wizard dialog box, create the new schedule. See Step 3 of Section 7.2 for details on how to create a new schedule.

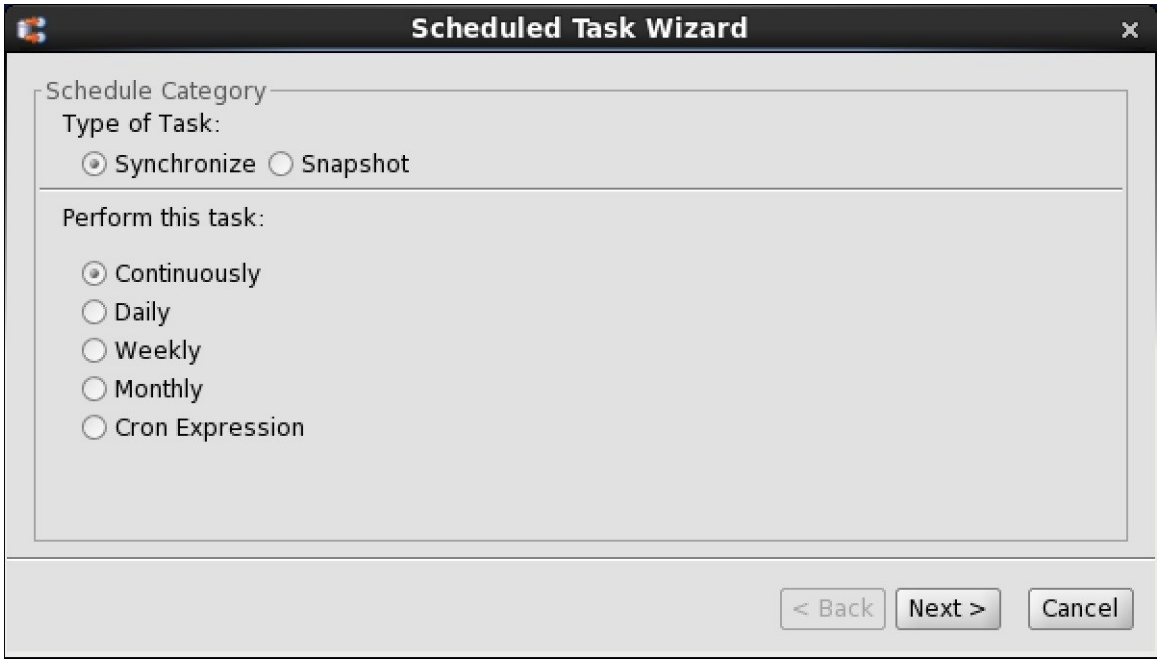


Figure 7-30 - Scheduled Task Wizard dialog box

7.3.2 Removing a Schedule

If you no longer wish replication to take place automatically, you must remove the schedule. You can always re-add a schedule or perform on demand replication.

Step 1 (For SMR only): Make sure the subscription server whose node is the parent of the subscription you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.3.1 for directions on starting and registering a subscription server.

Step 1 (For MMR only): Make sure the publication server whose node is the parent of the controller database you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Subscription node of the subscription for which you wish to remove the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

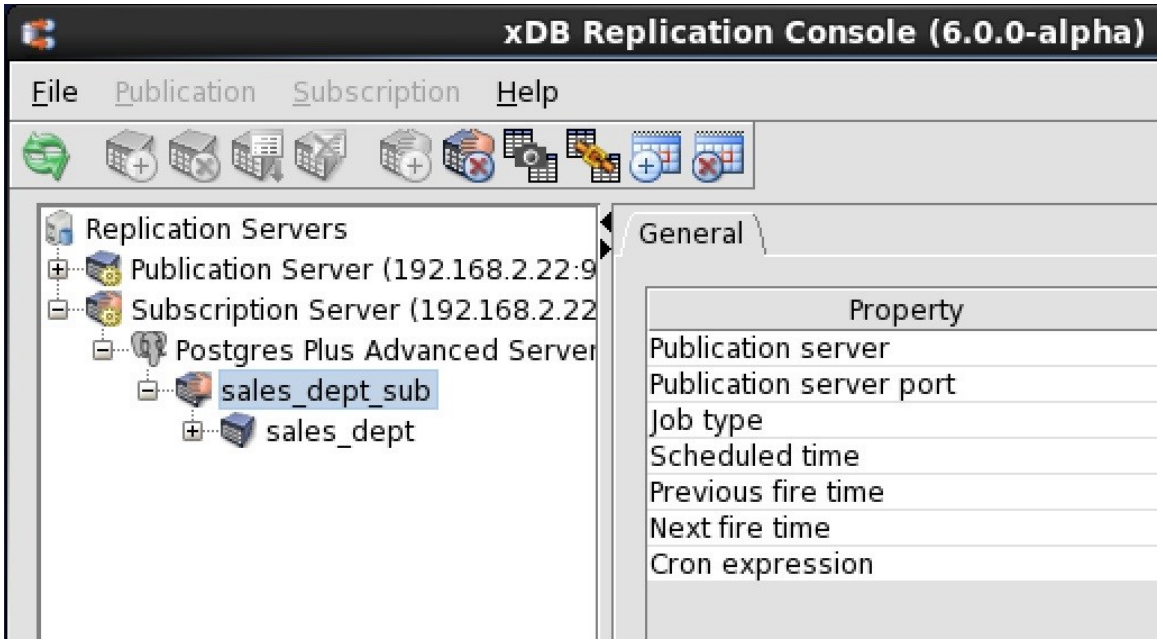


Figure 7-31 - Selecting a subscription for removal of a schedule

Step 2 (For MMR only): Select the Publication Database node designated as the controller database for which you wish to remove the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

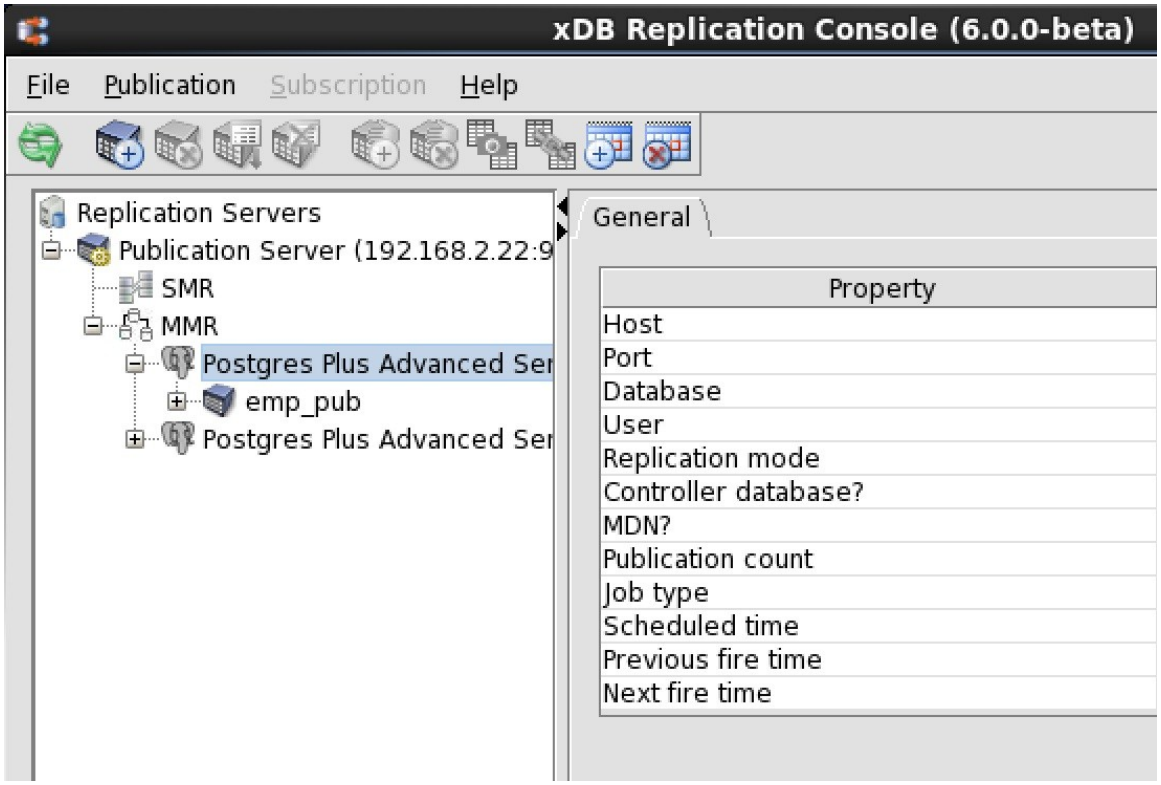


Figure 7-32 - Selecting the controller database for removal of a schedule

Step 3 (For SMR only): Remove the schedule in any of the following ways:

- From the Subscription menu, choose Schedule, then Remove Schedule.
- Click the secondary mouse button on the Subscription node and choose Remove Schedule.
- Click the primary mouse button on the Remove Schedule icon.

Step 3 (For MMR only): Remove the schedule in any of the following ways:

- Click the secondary mouse button on the Publication Database node and choose Remove Schedule.
- Click the primary mouse button on the Remove Schedule icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

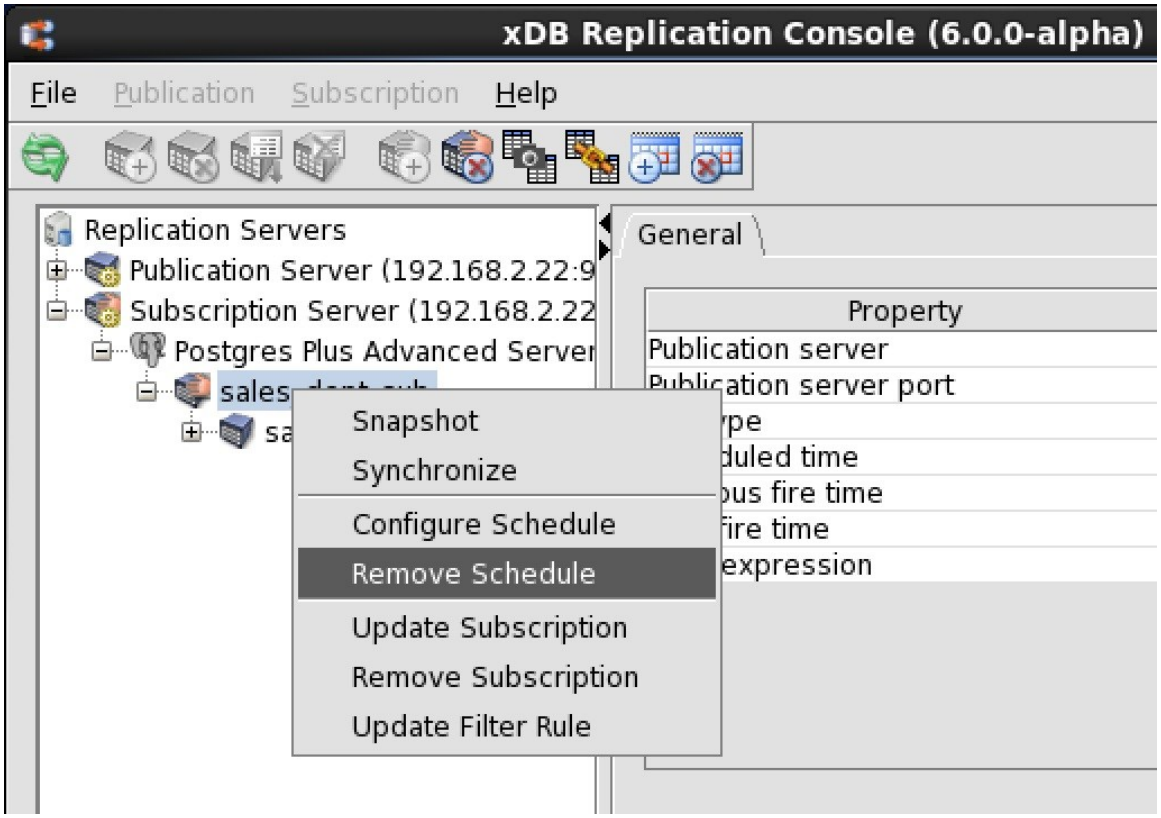


Figure 7-33 - Removing the schedule

Step 4: In the Removing Schedule confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

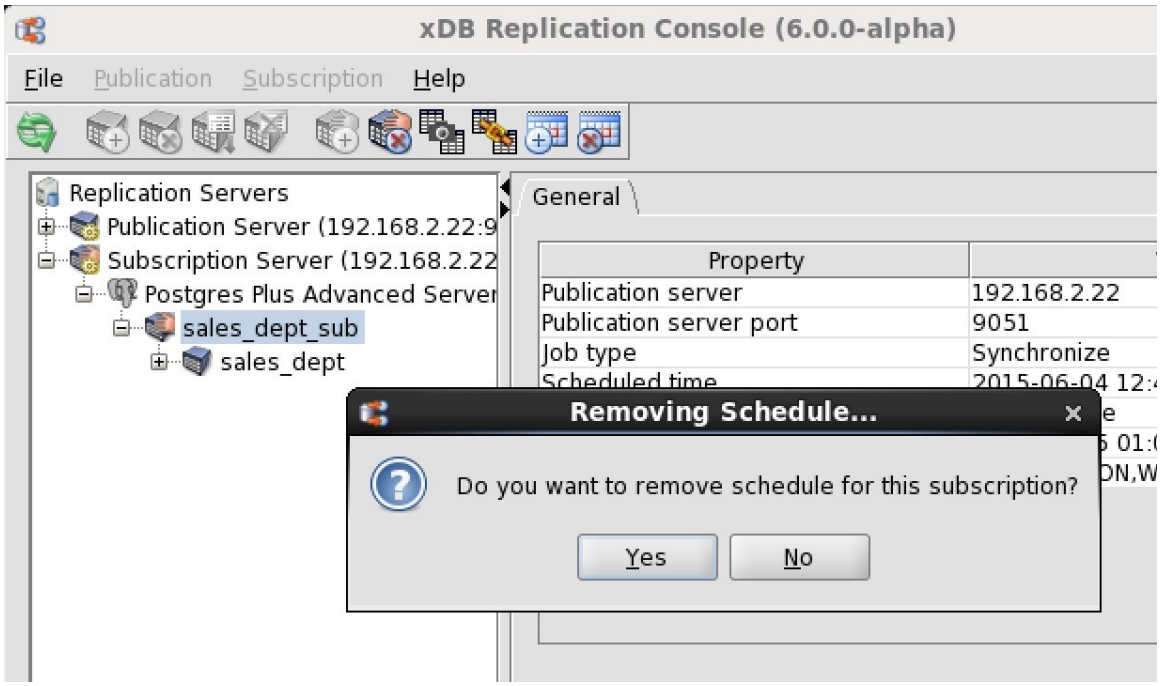


Figure 7-34 - Removing Schedule confirmation

If you click the Refresh icon in the tool bar, you will notice that schedule information no longer appears in the information window.

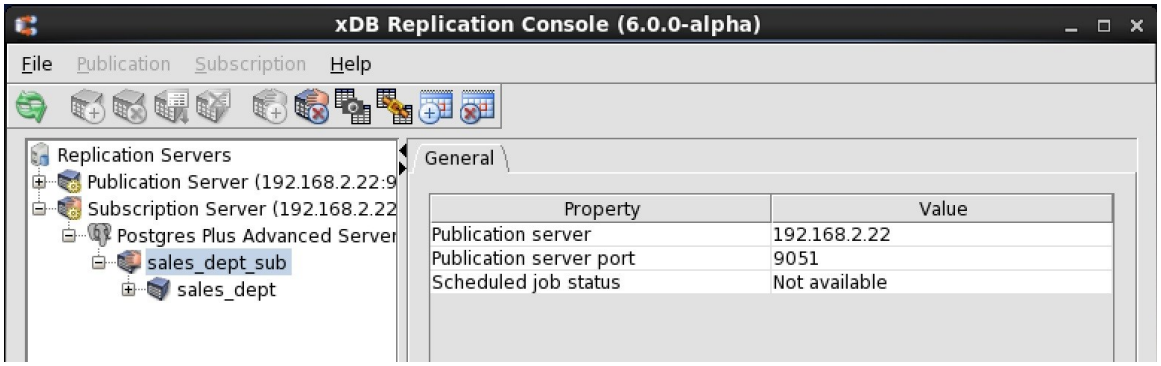


Figure 7-35 - Information window after schedule removal

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.4 Viewing Replication History

A summary of replications performed on each subscription or master node can be viewed in the xDB Replication Console. A detailed replication history showing each insert, update, and deletion made against each target table can be viewed as well. See Section 2.2.9 for a discussion on how changes are applied to target tables for the target-based method of synchronization replication. See Section 2.2.10 for information on the log-based method of synchronization replication.

Note (For SMR Only): The replication history can be viewed from the Publication node as well as from the Subscription node. The history shown for a Publication node is actually the exact same set of inserts, updates, and deletions made on the subscription tables by the publication server during synchronization. The history shown for a Publication node does not show the actual SQL statements processed on the publication tables that originated from user applications.

Note (For MMR only): The replication history can be viewed from the Publication node under any master node in the multi-master replication system. The history shown includes inserts, updates, and deletions made on all publication tables of all master nodes by the publication server during synchronization, and hence, the history appears the same regardless of the master node under which the history is viewed.

7.4.1 All Replication History

Replication history shows the following types of events that occur on a given subscription or master node:

- Snapshot replications
- Synchronization replications where at least one change (insert, update, or deletion) was applied to a target table
- Synchronization replications where no updates were applied to any of the target tables since the last restart of the publication server

The following steps describe how to view the replication history of the events in the preceding list.

Step 1 (For SMR only): Select the node beneath the Subscription node. Tabs labeled General, Realtime Monitor, and Replication History appear.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

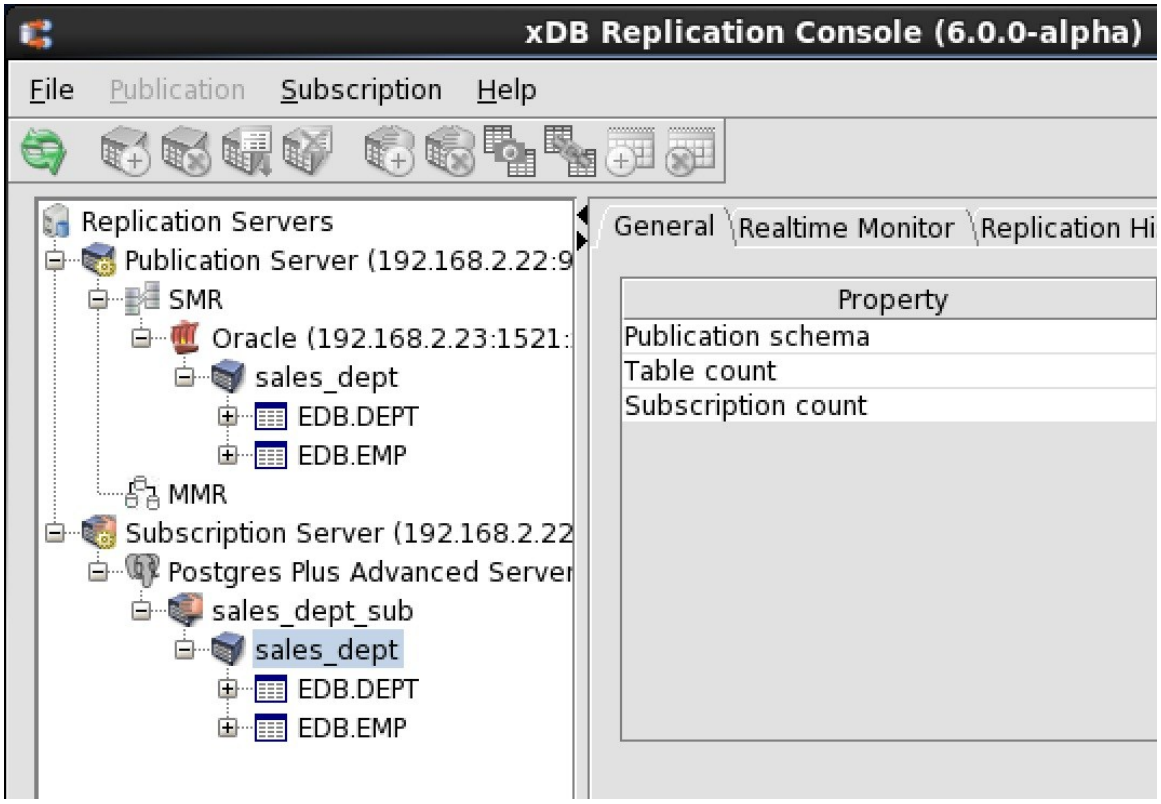


Figure 7-36 - Selecting a subscription on which to view replication history

Step 1 (For MMR only): Select any Publication node under a Database node representing a master node. Tabs labeled General, Realtime Monitor, Replication History, and Conflict History appear.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

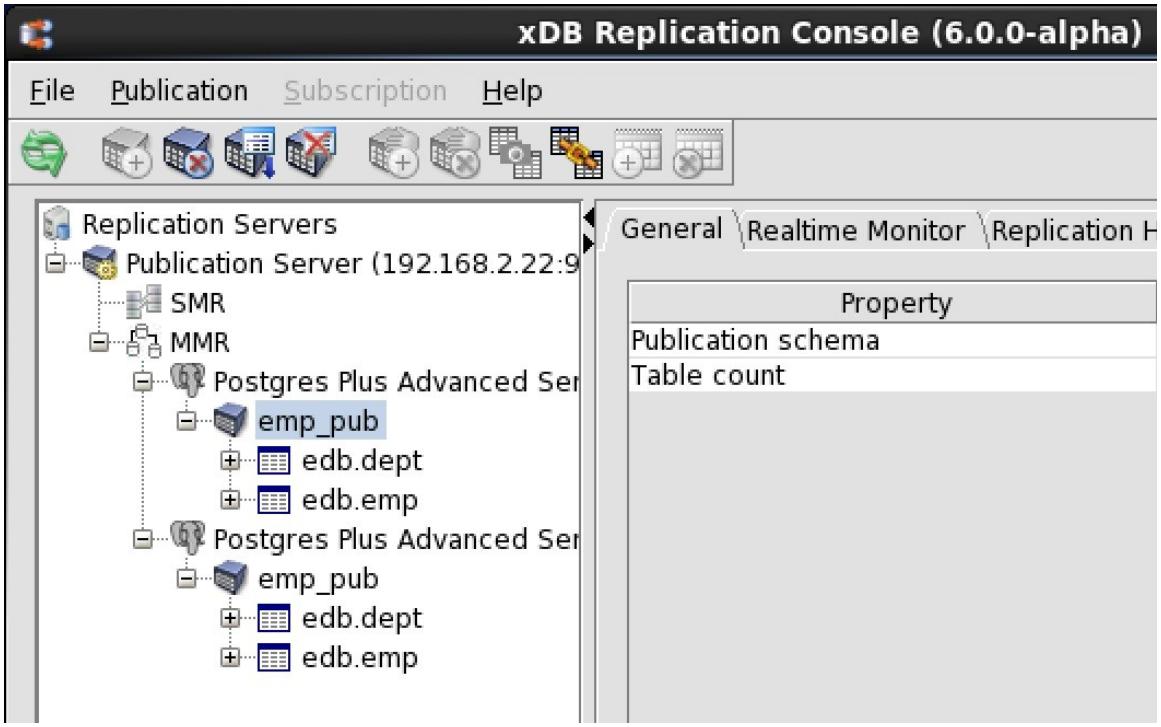


Figure 7-37 - Selecting a publication on which to view replication history

Step 2: Click the Replication History tab to show a history of replications.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

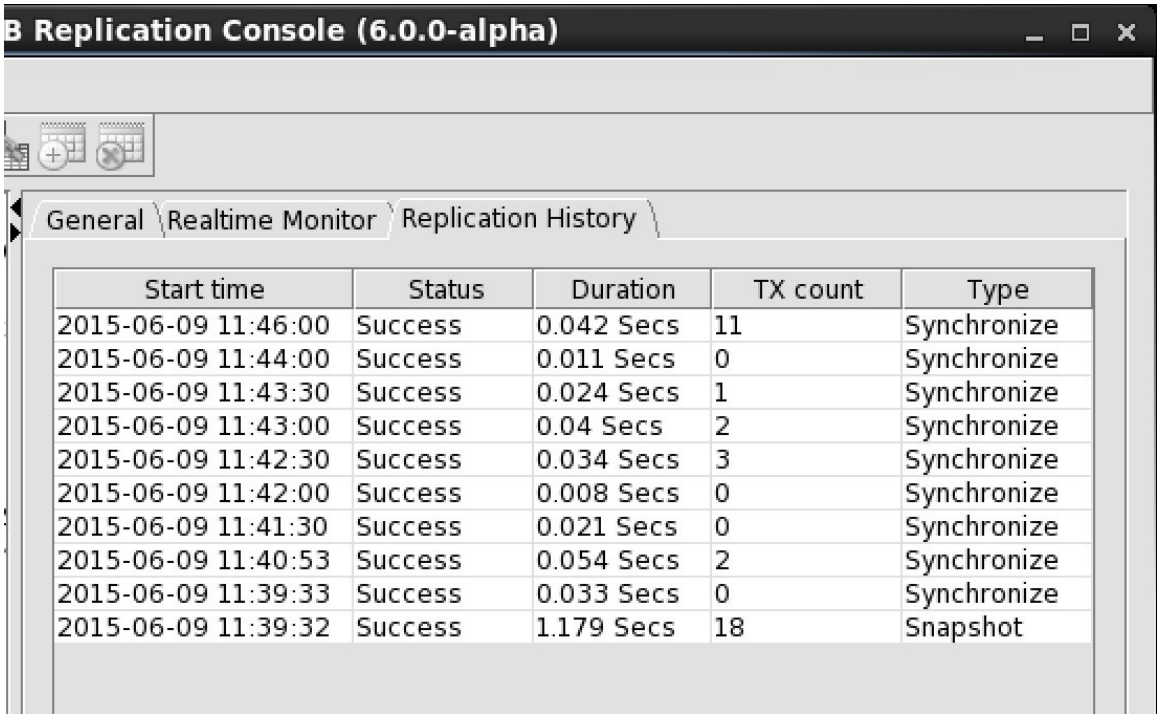


Figure 7-38 - Replication History tab

Note: Every snapshot replication and each synchronization replication with at least one update produces a history record that is maintained in replication history tables in the control schema. Over time the size of the replication history tables will grow. Replication history records can be periodically deleted. See Section 7.5.3 for information on cleaning up replication history.

7.4.2 Hiding Synchronizations With Zero Transaction Counts

You may notice synchronization replications with transaction counts of zero. These records indicate that there were no changes to synchronize at the time the replication occurred. For scheduled replications that occur frequently, this may result in a large number of lines in the Replication History tab, thus obscuring the more meaningful replications with non-zero transaction counts as shown below.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

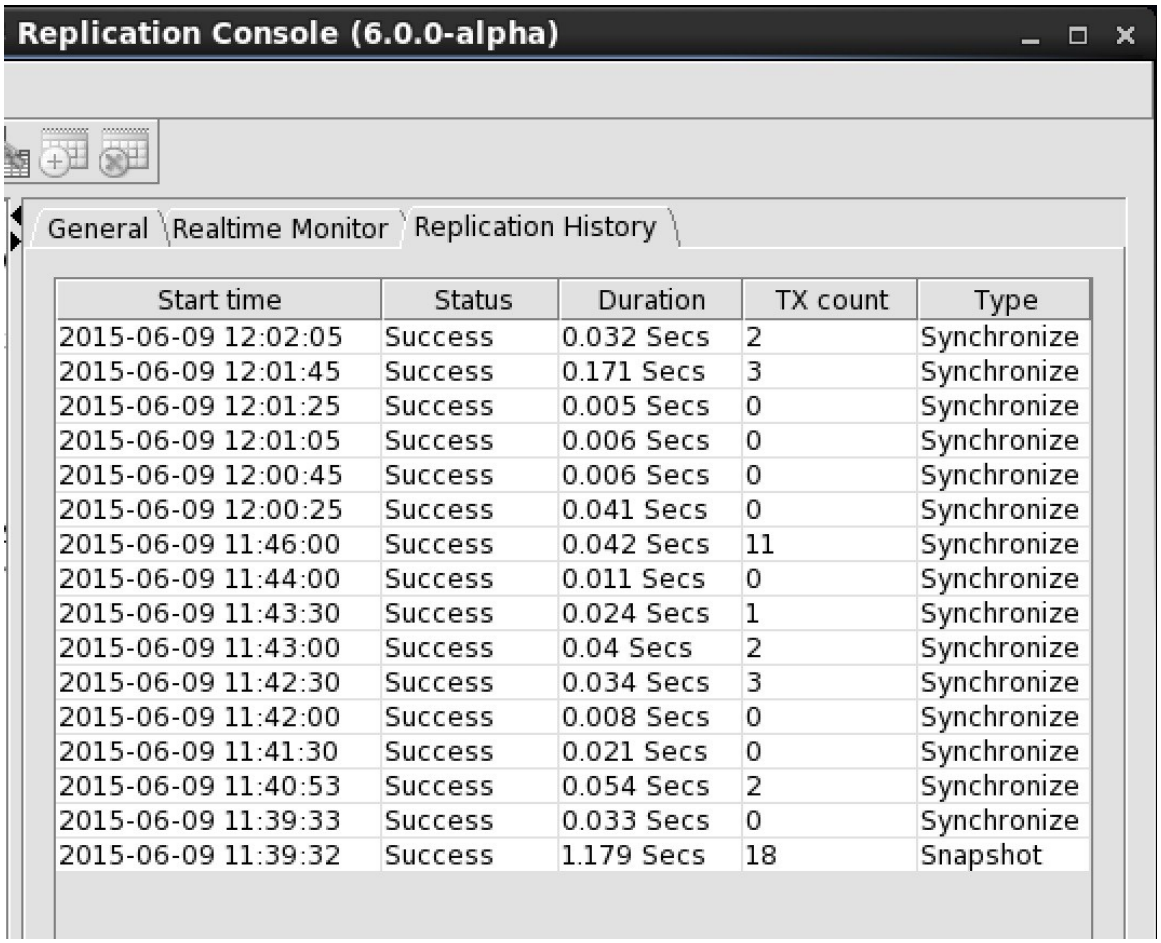


Figure 7-39 - Replication history with zero transaction counts

While viewing the Replication History tab, you can hide the records with zero transaction counts as follows:

Step 1: Check the Show History With Transactions Count > 0 check box located at the bottom of the Replication History tab.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

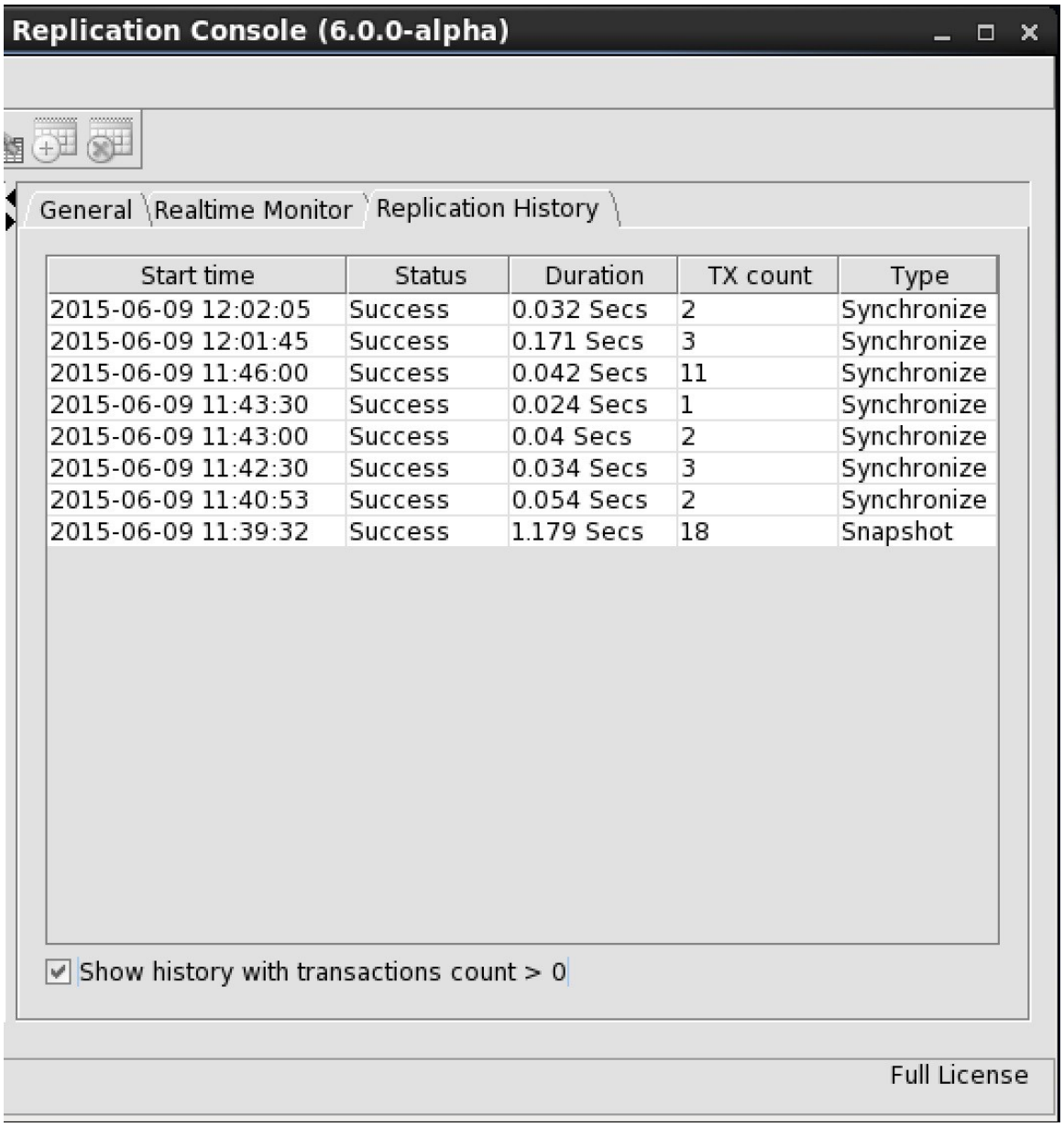


Figure 7-40 - Setting replication history to hide zero transaction count records

Step 2: The next time the Replication History tab refreshes, only the replications with non-zero transaction counts appear in the Replication History.

Note: Zero transaction count replication records are maintained in the publication server memory. By default, they are not permanently stored on disk. Therefore when the publication server is shut down, the in-memory zero transaction count replication records are no longer available.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When the publication server starts running again, zero transaction count replication records will appear on the Replication History tab as zero transaction count replications occur.

If you wish to permanently store zero transaction count replication records to disk, set the publication server configuration option `persistZeroTxRepEvent` to `true`. See Section 10.4.1.12 for further information.

7.4.3 Shadow Table History

Expanding the nodes under the Subscription node of a single-master replication system, or the Publication node of a multi-master replication system provides more information about the subscription or publication.

Step 1: Select a table to reveal tabs that contain general information about the table and the replication history of the table. Expand a Table node to reveal the columns in the table.

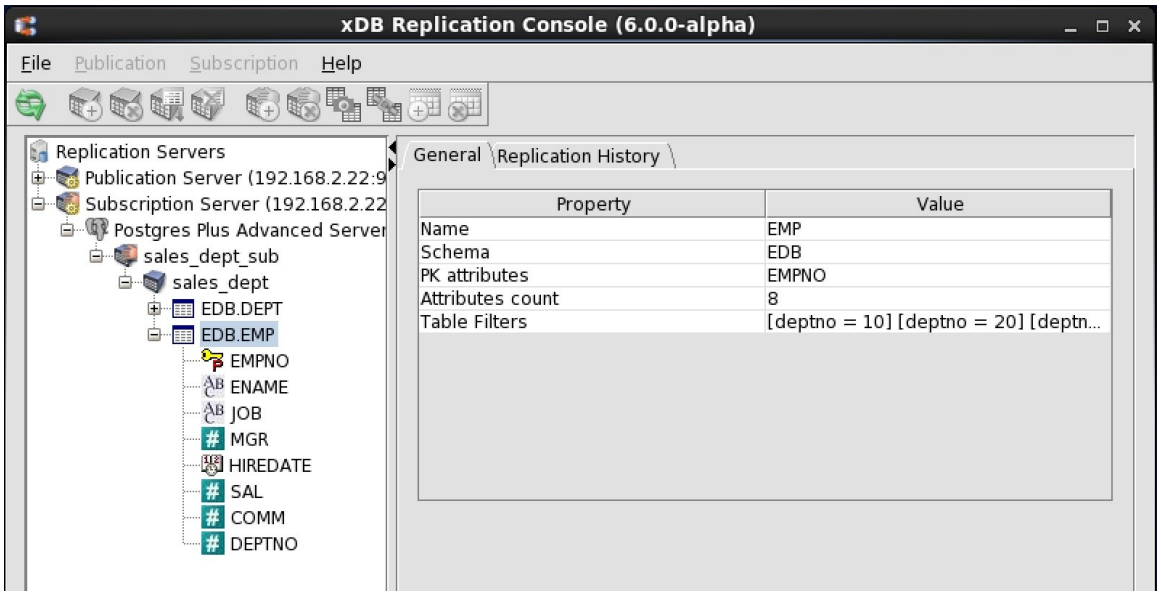


Figure 7-41 - Table properties and columns

Step 2: Click the Replication History tab to show a history of replications for this table.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

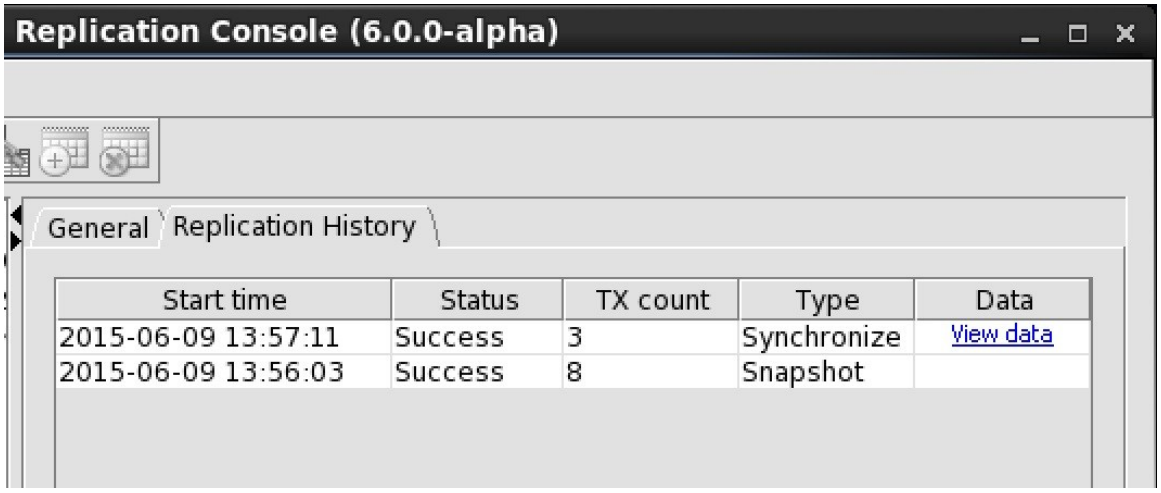


Figure 7-42 - Table replication history tab

Step 3: Click the View Data link to show a list of each change made to the table during the synchronization replication. The Synchronize History window shows two update operations followed by one insert operation against the `emp` target table that correspond to the following set of SQL statements executed on the `emp` source table:

```
UPDATE emp SET hiredate = TO_DATE('07-JUN-15'), mgr = 7698 WHERE empno IN
(9001, 9002);
INSERT INTO emp (empno, ename, job, mgr, deptno) VALUES (9003, 'JOHNSON',
'SALESMAN', 7698, 30);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

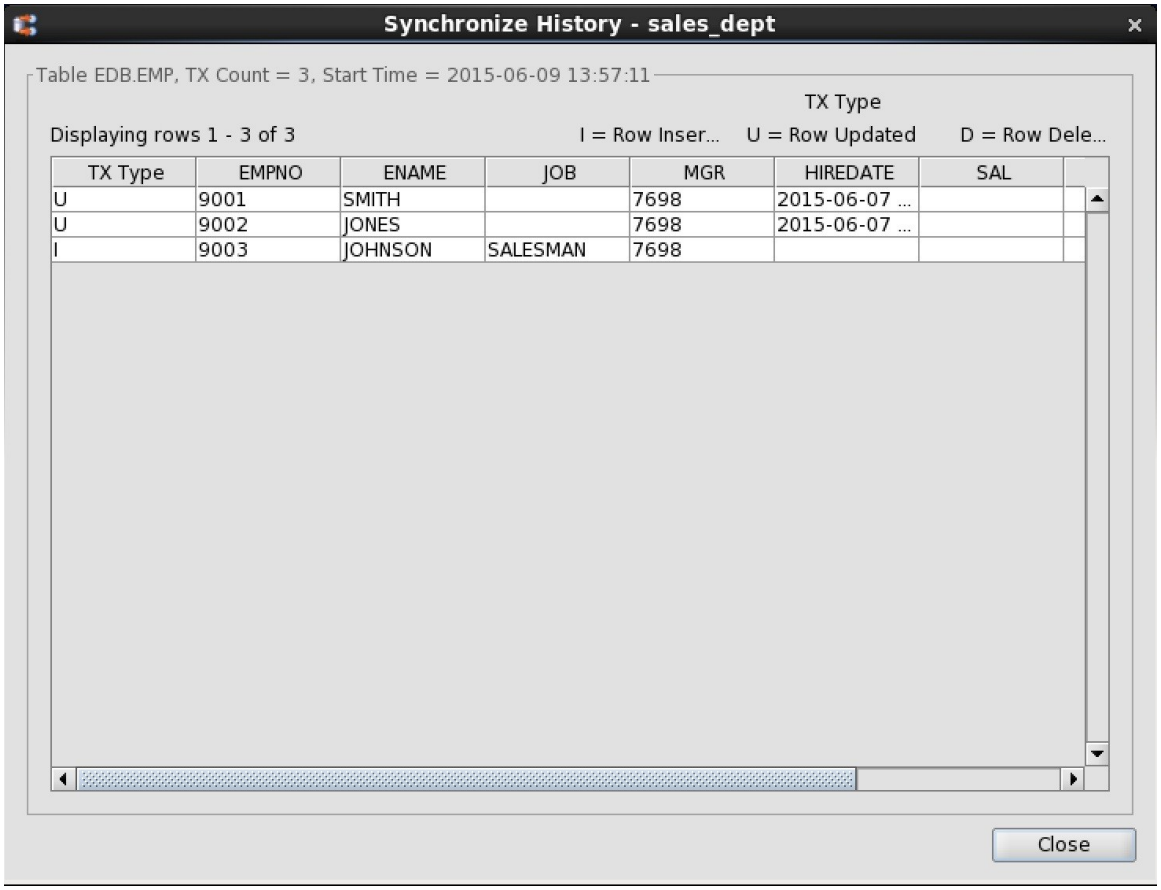


Figure 7-43 - Synchronize History window

Note: Since all insert, update, and delete operations on all source tables are recorded in shadow tables, the size of the shadow tables may grow considerably over time for volatile source tables. The rows shown in the Synchronize History window are obtained from these shadow tables. Rows in the shadow tables can be periodically deleted. See Section 7.5.2 for information on cleaning up the shadow tables.

7.5 Managing History

xDB Replication Server maintains three types of history:

- **Shadow Table History.** Records of each change (insert, update, or delete) that was applied to each target table during synchronization replications using the trigger-based method. There is no shadow table history for synchronization replications using the log-based method.
- **Replication History.** Summary records of each replication.
- **Event History.** Records of each change that was applied to various control schema tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The size of the control schema tables that store these history records grows over time, and thus there are a number of methods referred to as *cleanup* to delete such history records.

Shadow table history cleanup information is described in sections 7.5.1 and 7.5.2.

For replication history cleanup, see Section 7.5.3.

For event history cleanup, see Section 7.5.4.

7.5.1 Scheduling Shadow Table History Cleanup

A preference can be set for each publication database definition to determine if and when shadow table history cleanup should be scheduled for all publications appearing under its corresponding Publication Database node.

Shadow table history cleanup has no benefit for snapshot-only publications so if all of your publications under a Publication Database node are snapshot-only publications, then scheduled shadow table history cleanup should be disabled following steps 1 through 4.

Replication history is not deleted by scheduling shadow table history cleanup.

Whenever a new publication database definition is created, there is a scheduled default setting of every Sunday at 12:00 AM midnight for shadow table history cleanup.

Note: A configuration option is available to force shadow table history cleanup after every synchronization replication. See Section 10.4.1.9 for information on this option.

Note: The cleanup of certain processed rows in the shadow tables may be delayed beyond the next scheduled cleanup, but will eventually be removed in subsequent cleanup events.

For Oracle only: For scheduling of shadow table history cleanup on an Oracle publication database, the Oracle `DBMS_JOB` package on the Oracle database server is used. The time you specify in the schedule for cleanup is passed and stored in `DBMS_JOB` without time zone translation.

For example, assume the publication server is running on a host in New York and the Oracle publication database is on a server in California, which has a 3-hour time difference. If you set shadow table history cleanup to run at 12:00 AM midnight according to the New York based publication server, the cleanup job on the California based Oracle database will start at 12:00 AM midnight Pacific Time (in California), which would be 3:00 AM Eastern Time (in New York).

For SQL Server only: For scheduling of shadow table history cleanup on a SQL Server publication database, SQL Server Agent is used on the host running SQL Server. The

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

time you specify in the schedule for cleanup is passed to SQL Server Agent without time zone translation. The effect is the same as described for Oracle in the preceding example.

For Postgres only: For scheduling of shadow table history cleanup on a Postgres publication database, the Quartz scheduler is used on the host running the publication server based on the location of the controller database.

For example, assume the publication server is running on a host in New York and the Postgres publication database on which cleanup is to be scheduled is also the controller database and is on a host in California. If you set shadow table history cleanup to run at 12:00 AM midnight according to the New York based publication server, the cleanup job on the California based Postgres database will start at 12:00 AM midnight Pacific Time (in California), which would be 3:00 AM Eastern Time (in New York).

By contrast, assume the publication server is running on a host in New York along with the controller database, and the Postgres publication database on which cleanup is to be scheduled is on a host in California. If you set shadow table history cleanup to run at 12:00 AM midnight according to the New York based publication server and controller database, the cleanup job on the California based Postgres database will start at 12:00 AM midnight Eastern Time (in New York), which would be 9:00 PM Pacific Time (in California).

For Oracle only: The cleanup job on an Oracle publication database runs independently of the publication server, so the cleanup job will run regardless of whether or not the publication server is running.

For Postgres only: The publication server must be running in order for the cleanup job to run on a Postgres publication database.

Note: An alternative to using the Quartz scheduler when Postgres is the publication database, is to use pgAgent job scheduling instead. See Section 10.4.1.8 for information on how to use pgAgent job scheduling and the advantages, thereof.

The following steps show how to alter the default setting.

Step 1: Make sure the publication server whose node is the parent of the publication database definition whose cleanup scheduling preference you want to set is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication Database node for which you want to set the cleanup scheduling preference.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

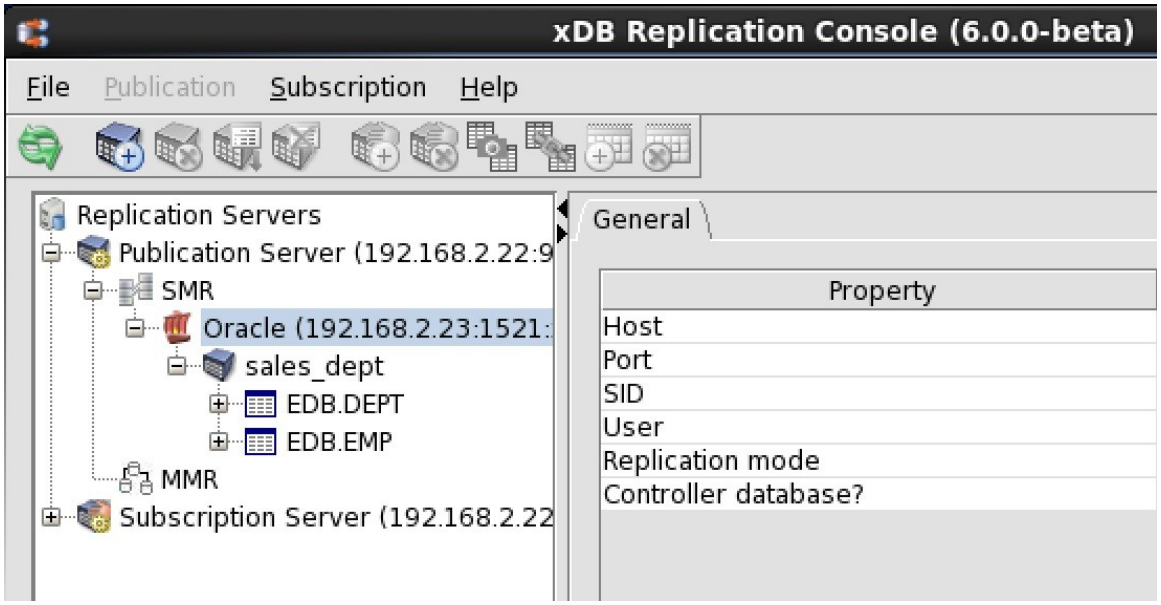


Figure 7-44 - Selecting the publication database for cleanup scheduling

Step 3: From the Publication menu, choose Preferences. Alternatively, click the secondary mouse button on the Publication Database node and choose Preferences. The Publication Server Preferences dialog box appears.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

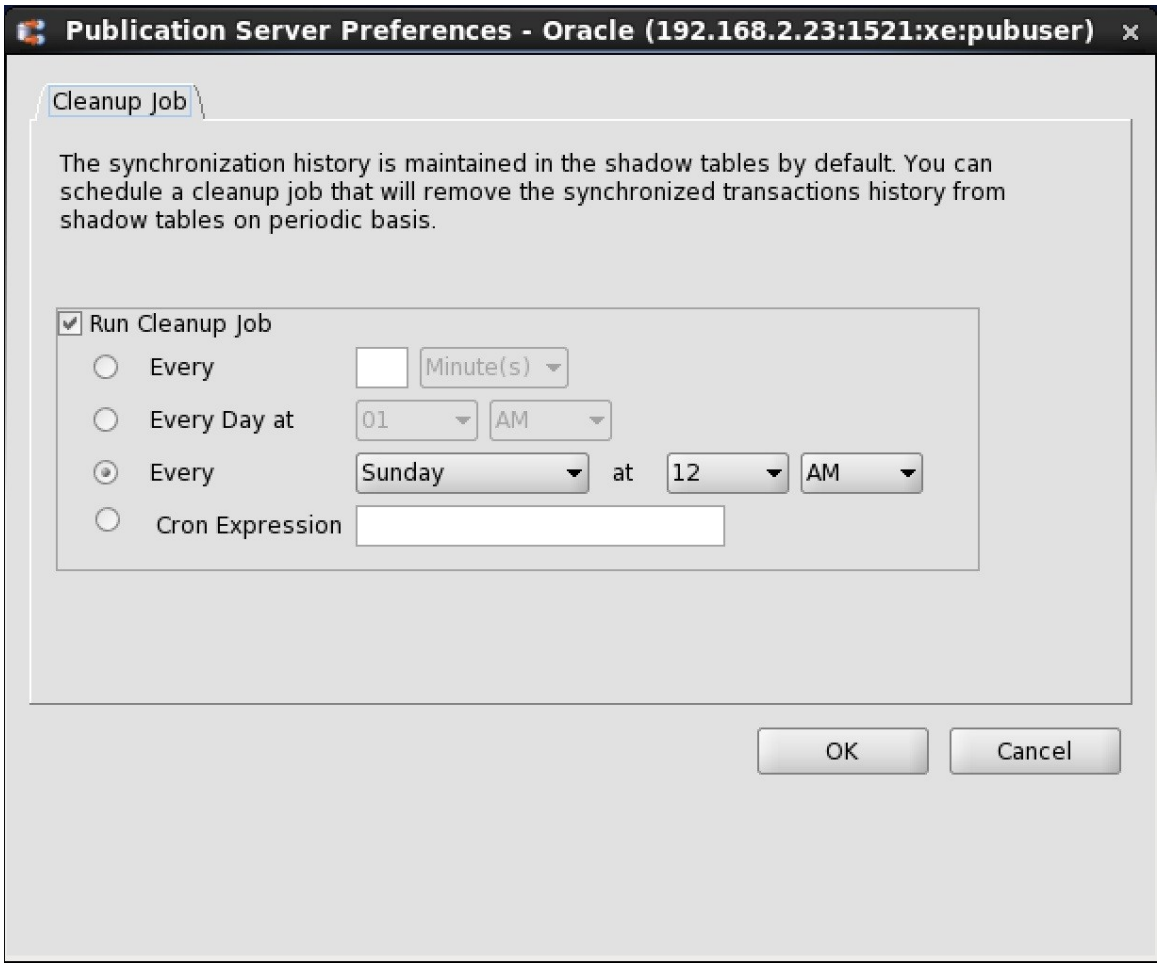


Figure 7-45 - Publication Server Preferences dialog box

Step 4: In the Publication Server Preferences dialog box, uncheck the box if you do not want to run a scheduled shadow table history cleanup job. Click the OK button and skip the remaining steps.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

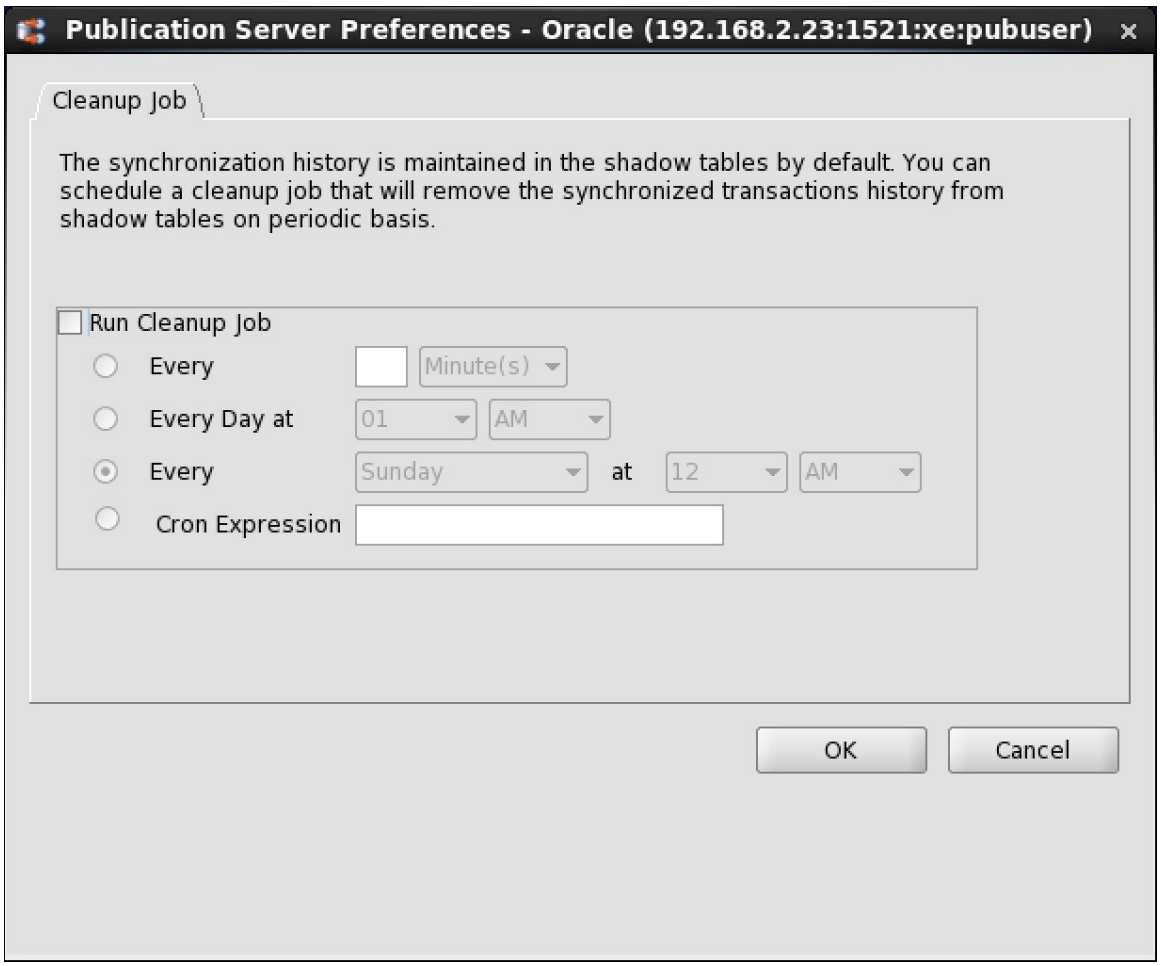


Figure 7-46 - Disabled schedule for shadow table history cleanup

Step 5: If you want to schedule shadow table history cleanup, make sure the Run Cleanup Job check box is selected. Select the radio button for the cleanup frequency. The frequency choices have the following meanings:

- **Every number of minutes/hours.** Schedules shadow table history cleanup to run continuously at an interval in either minutes or hours that you specify. Select this option if there are huge volumes of updates to the publication tables during the course of the day, every day.
- **Every Day at hour of day.** Schedules shadow table history cleanup to run once a day on the hour you choose. Select this option if updates to the publication tables are frequent enough to require more than once a week cleanup, but not needed more than once a day.
- **Every selected day of week at hour of day.** Schedules shadow table history cleanup to run once a week on the day and at the hour you choose. Select this

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

option if updates to the publication tables are infrequent and you do not want to run cleanup manually.

- **Cron Expression.** Provides additional flexibility for specifying a schedule beyond the three preceding radio button choices. See appendix Section 10.4.3 for directions on writing a cron expression.

Note: A configuration option is available to force shadow table history cleanup after every synchronization replication. See Section 10.4.1.9 for information on this option.

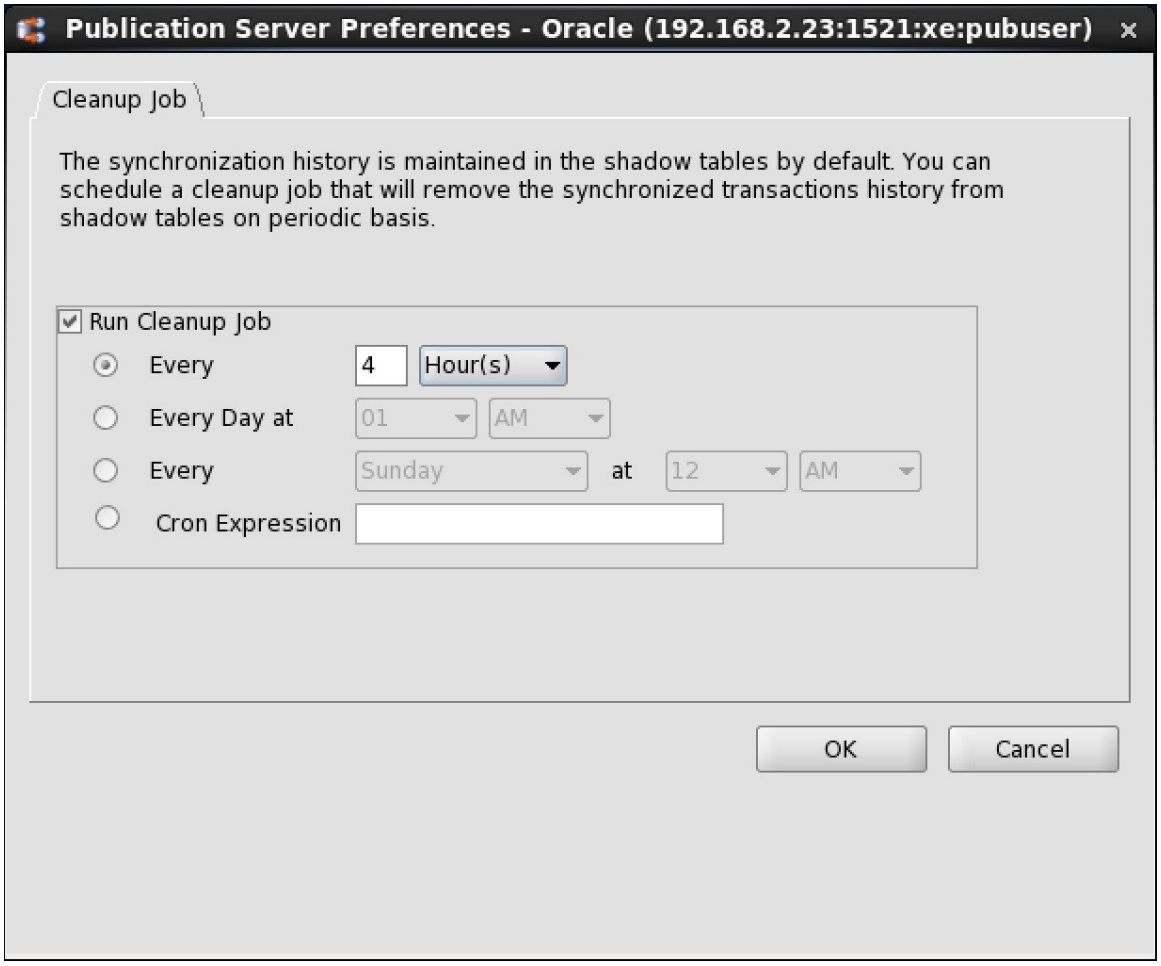


Figure 7-47 - Cleanup Job dialog box

Step 6: Click the OK button to accept the schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.5.2 Cleaning Up Shadow Table History

Non snapshot-only publications (that is, publications on which synchronization replications occur) whose tables experience frequent changes should have their shadow table history cleaned up periodically, otherwise the amount of disk space consumed by the shadow tables in the publication database may grow too rapidly.

When shadow table history is cleaned up, the rows in the following xDB Replication Server metadata tables are deleted:

- `RREP_TXSET`
- `RREP_TXSET_LOG`
- `RRST_schema_table`

For Oracle only: When Oracle is the publication database, these tables are located in the publication database in the schema of the publication database user.

For SQL Server only: When SQL Server is the publication database, these tables are located in the publication database in the schema you chose during Step 5 of Section 5.1.4.2.

For Postgres only: When Postgres is the publication database, these tables are located in the publication database in schema `_edb_replicator_pub`.

Shadow table history cleanup can be scheduled to run periodically (see Section 7.5.1) or it can be run on demand.

Note: The cleanup of certain processed rows in the shadow tables may not occur during an on demand cleanup or may be delayed beyond the next scheduled cleanup, but will eventually be removed in subsequent cleanup events.

The following are the steps to run shadow table history cleanup on demand for a chosen publication.

Step 1: Make sure the publication server whose node is the parent of the publication whose shadow table history you wish to clean up is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication node of the publication for which you want to clean up the shadow table history.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

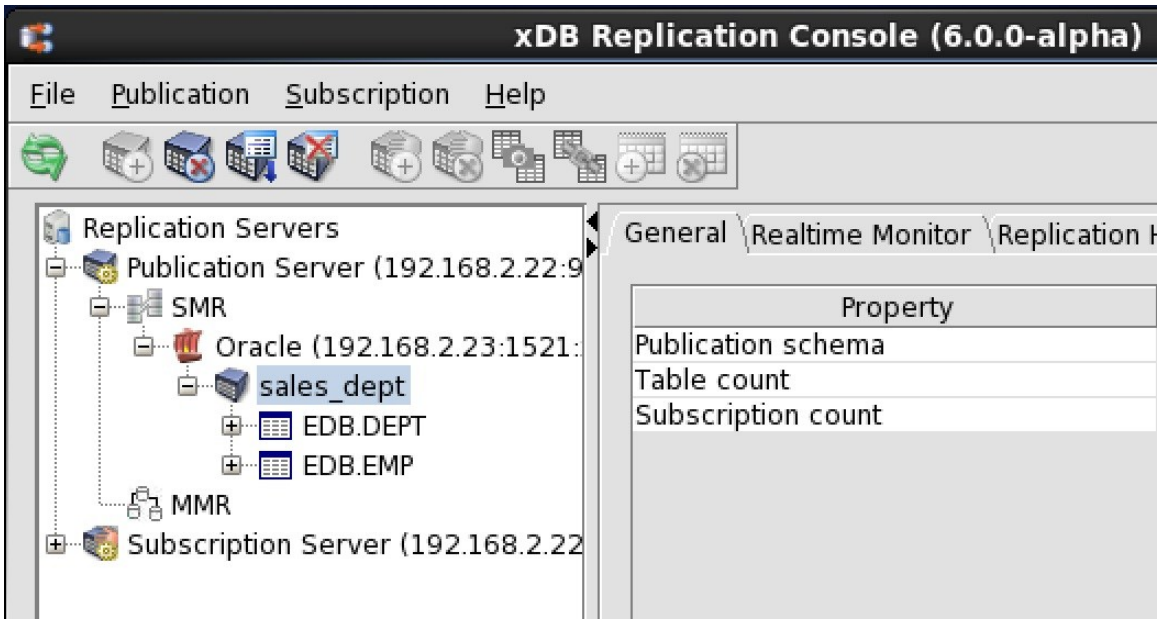


Figure 7-48 - Selecting a publication for shadow table history cleanup

Step 3: From the Publication menu, choose Cleanup Shadow Table History. Alternatively, click the secondary mouse button on the Publication node and choose Cleanup Shadow Table History. The Cleanup Synchronization History confirmation box appears.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

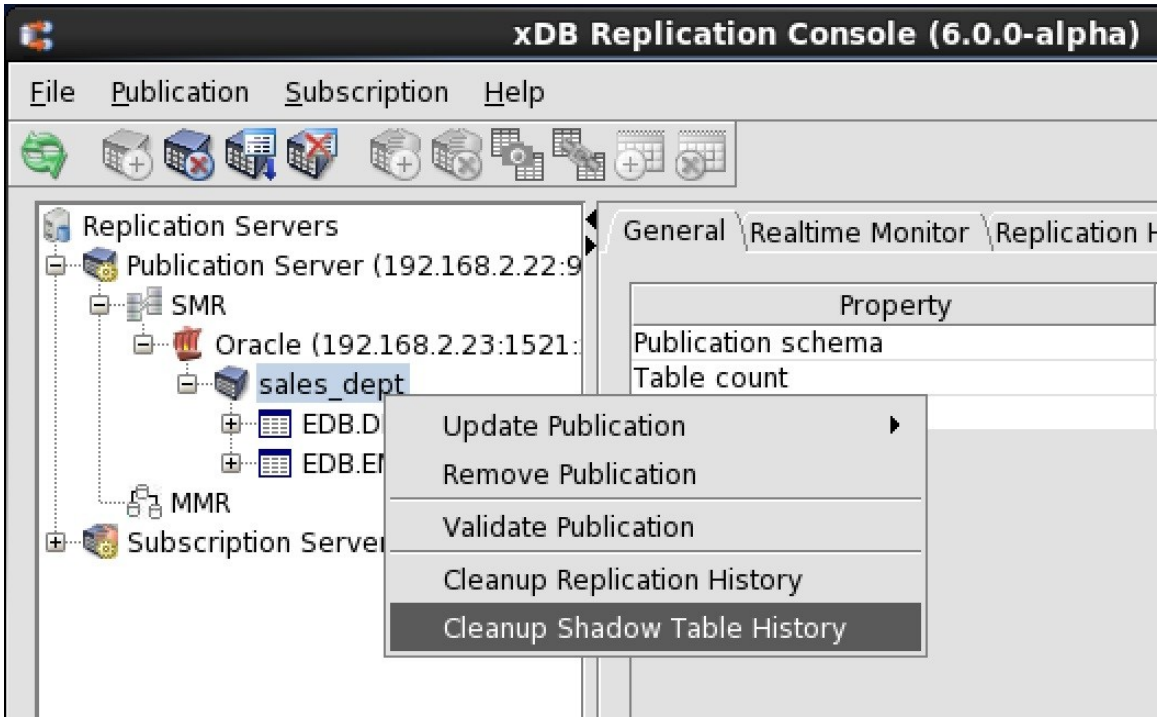


Figure 7-49 - Cleaning up shadow table history

Step 4: Click the Yes button in the Cleanup Synchronization History confirmation box.

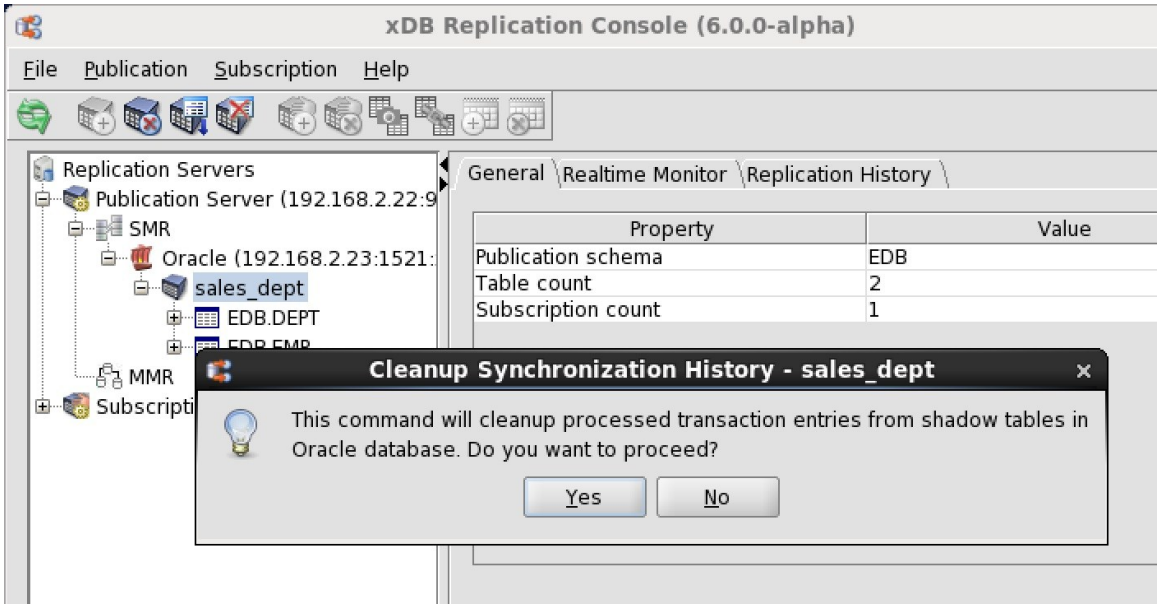


Figure 7-50 - Cleanup Synchronization History confirmation

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 5: Click the Yes button in response to Shadow Table's Transaction History Removed Successfully.

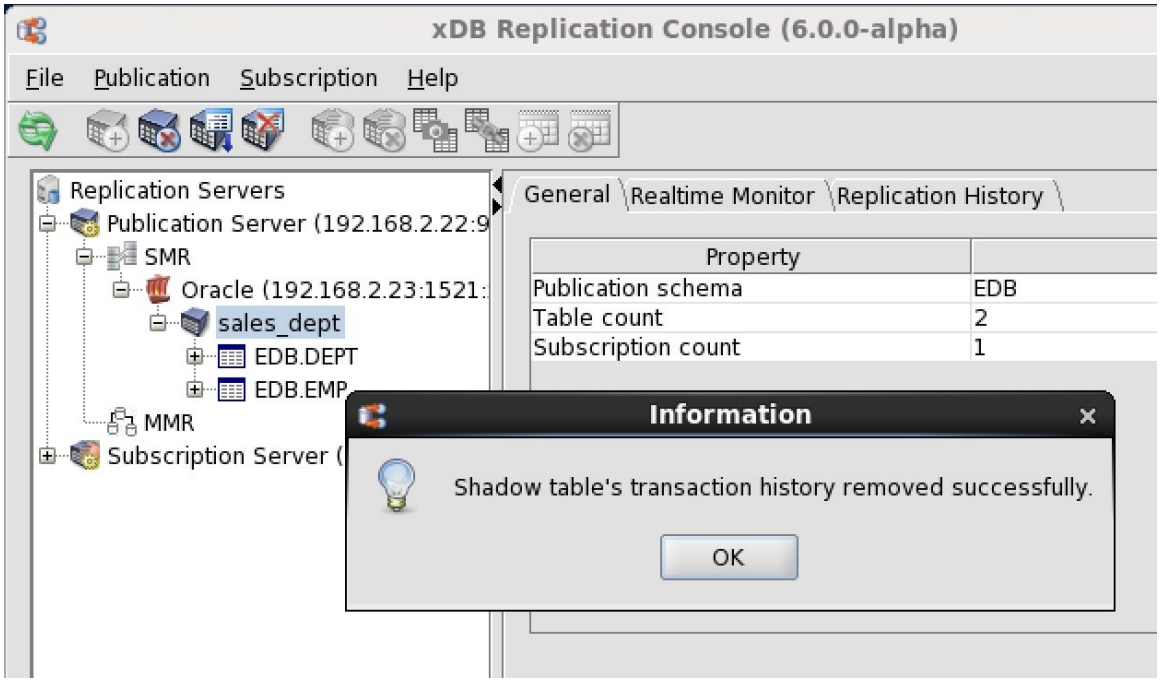


Figure 7-51 - Successful cleanup of shadow table history

After shadow table history cleanup, if you click the View Data link of the Replication History tab, an information message appears stating that there is no synchronization history to view.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

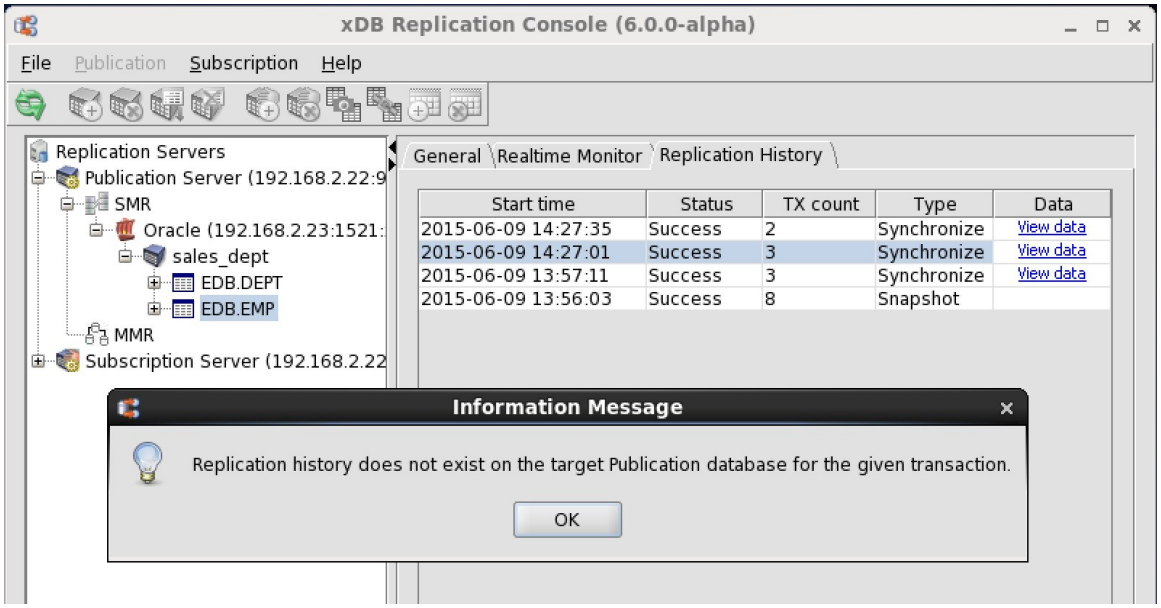


Figure 7-52 - View Data link after shadow table history cleanup

7.5.3 Cleaning Up Replication History

Cleaning up replication history deletes rows from the following tables in the control schema:

- `xdb_pub_replog`
- `xdb_pub_table_replog`

The following are the steps to run replication history cleanup for a chosen publication.

Step 1: Make sure the publication server whose node is the parent of the publication whose replication history you wish to cleanup is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication node of the publication for which you want to clean up replication history.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

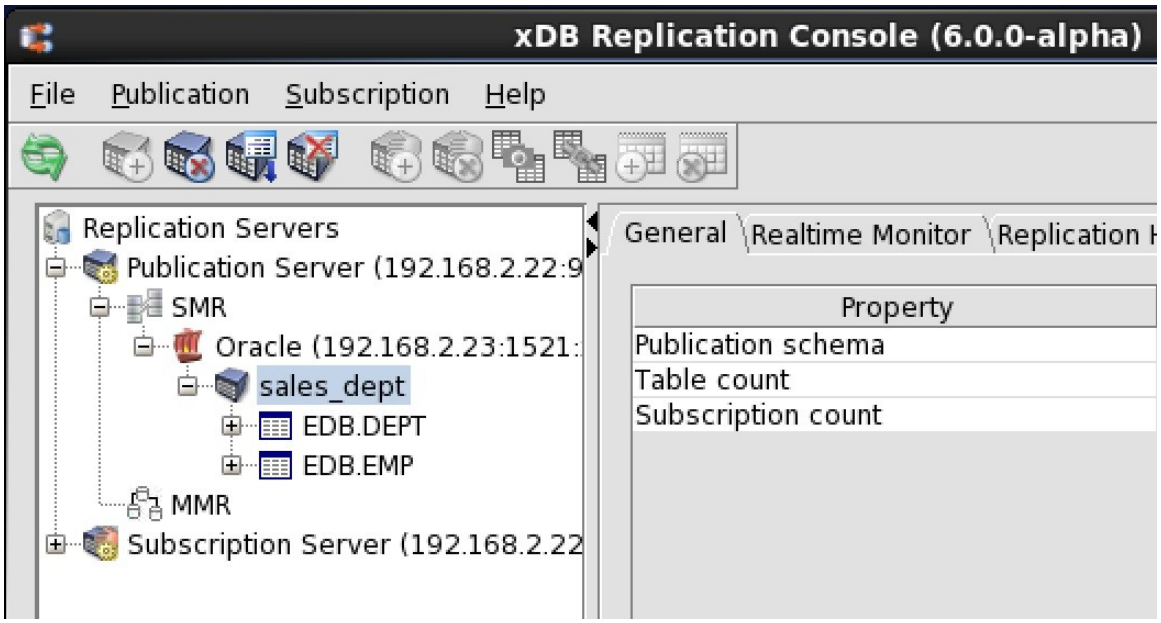


Figure 7-53 - Selecting a publication for replication history cleanup

Step 3: From the Publication menu, choose Cleanup Replication History. Alternatively, click the secondary mouse button on the Publication node and choose Cleanup Replication History. The Cleanup Replication History confirmation box appears.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

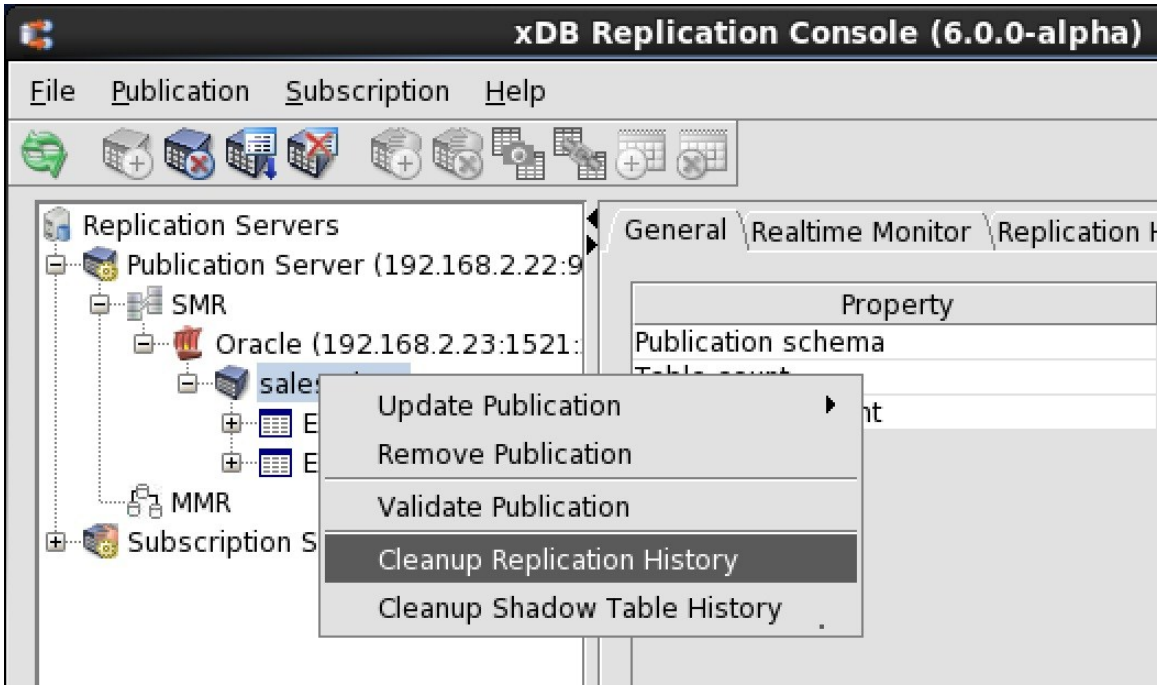


Figure 7-54 - Cleaning up replication history

Step 4: Click the Yes button in the Cleanup Replication History confirmation box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

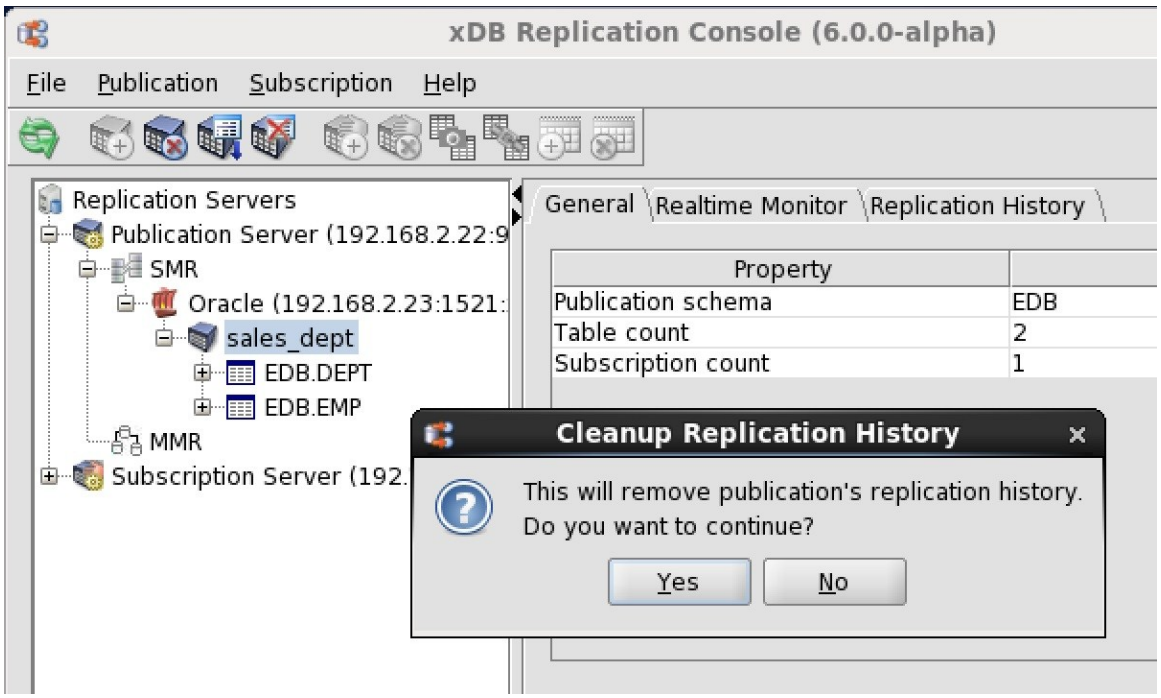


Figure 7-55 - Cleanup Replication History confirmation

Step 5: Click the Yes button in response to Replication History Has Been Removed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

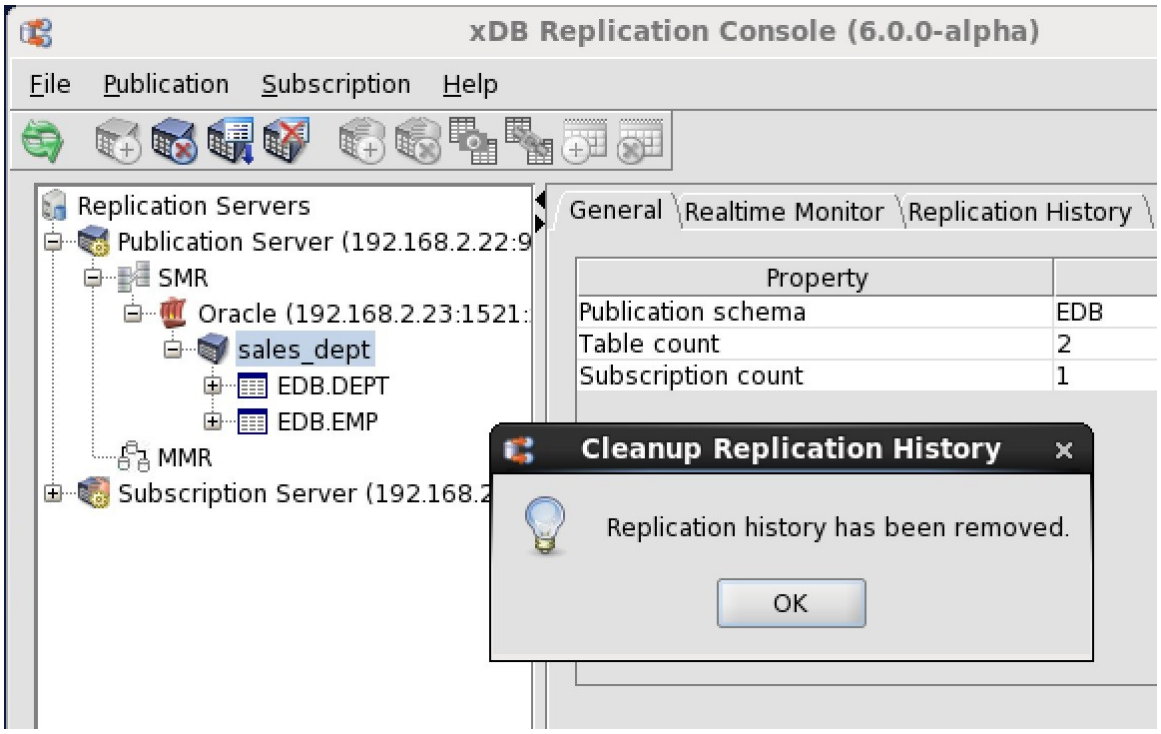


Figure 7-56 - Successful cleanup of replication history

After replication history cleanup, if you click the Replication History tab, no history records appear.

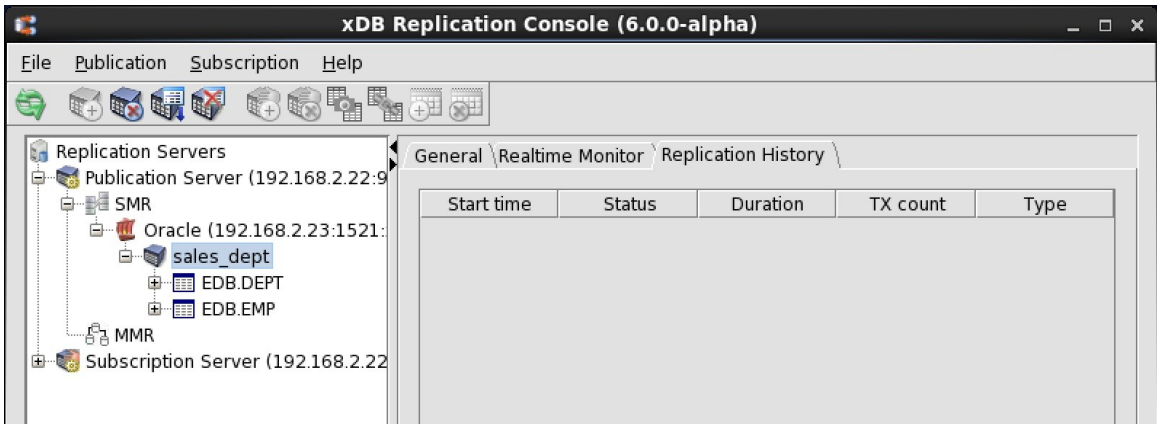


Figure 7-57 - Replication History tab after replication history cleanup

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.5.4 Cleaning Up Event History

Unlike shadow table history (Section 7.5.2) and replication history (Section 7.5.3), event history is neither viewable nor removable using the xDB Replication Console.

Event history is a recording of various updates to the control schema tables that occur during xDB Replication Server processing. Hence, the event history content grows significantly over time.

The tables containing event history, and thus the tables to be cleaned up are the following:

- `xdb_events`
- `xdb_events_status`

In addition, the replication history tables are cleaned up as well. These tables can also be manually cleaned up as described in Section 7.5.3.

- `xdb_pub_replog`
- `xdb_pub_table_replog`

For Oracle, these tables are located in the schema of the publication database user.

For SQL Server and Postgres, these tables are located in schema `_edb_replicator_pub`.

The event history and replication history data in the control schema are deleted on a scheduled, daily basis at 12 AM, thus reducing the number of rows in tables `xdb_events`, `xdb_events_status`, `xdb_pub_replog`, and `xdb_pub_table_replog`.

Publication server configuration option `historyCleanupDaysThreshold` provides the capability to designate how old the completed data must reach before its removal. The default setting is that completed data must be older than seven days before it is deleted during the daily 12 AM cleanup process.

In order to cleanup all completed event and replication history regardless of its age, set `historyCleanupDaysThreshold` to a value of `0`, then restart the publication server. The cleanup occurs during the next scheduled 12 AM cleanup process.

See Section 10.4.1.10 for the `historyCleanupDaysThreshold` option.

7.6 Managing a Publication

After a publication has been created, certain aspects of the underlying replication system environment might be subsequently altered for any number of reasons. Attributes that

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

might change include the network location of the publication database server, the network location of the host running the publication server, database or operating system user names and passwords, and so forth.

The aforementioned information is saved in the replication system metadata when a publication is created. Changes to these attributes result in inaccurate replication system metadata, which in turn may result in errors during subsequent replication attempts or replication system administration.

This section describes how to update the metadata stored for the publication server, the publication database definition, and publications in order to keep the information consistent with the actual replication system environment.

7.6.1 Updating a Publication Server

There are two aspects of metadata related to the publication server that may need to be updated depending upon the change in the host environment:

- If the network location (IP address or port number), admin user name, or password of the publication server changes, and if you have saved this information in the server login file, you need to update the server login file with the new information.
- If the network location (IP address or port number) of a subscription server changes, then for each publication server managing a publication associated with a subscription of the changed subscription server, you must update the publication server's metadata with the subscription server's new network location. This type of update applies only to single-master replication systems.

The first type of update is discussed in the following section. The second type of update is discussed in Section 7.6.1.2.

7.6.1.1 Publication Server Login File

When you register a publication server in the xDB Replication Console, you may choose to save the publication server's network location, admin user name, and encrypted password in a server login file on the computer on which you are running the xDB Replication Console. See Section 4.2 for information on saving the login information.

The steps described in this section show you how to update the publication server's login information in the server login file.

It is assumed that the xDB Replication Console is open on your computer and the publication server whose login information you wish to alter in the server login file, appears as a Publication Server node in the xDB Replication Console's replication tree.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

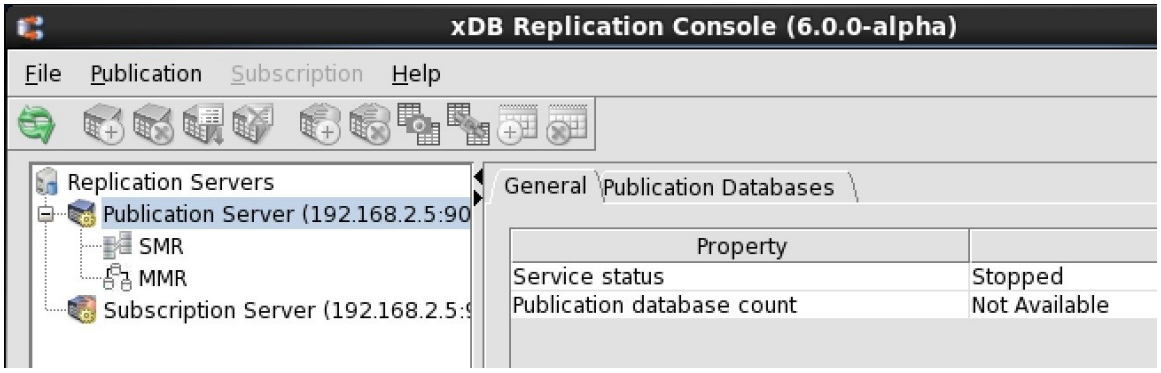


Figure 7-58 - Publication Server node

You can perform the following actions on the server login file:

- Change the publication server's login information (host IP address, port number, admin user name, and password) that you last saved in the server login file.
- Delete the publication server's login information that is currently saved in the server login file. This is the default action, which will require you to register the publication server again the next time you open the xDB Replication Console.
- Resave the publication server's login information in the server login file. Each time you open the Update Publication Server dialog box, you must choose to save the login information if you want it recorded in the server login file.

The following steps change only the content of the server login file residing on the host under the current xDB Replication Console user's home directory. These changes do not alter any characteristic of the actual publication server daemon (on Linux) or service (on Windows). These changes affect only how a publication server is viewed through the xDB Replication Console on this host by this user.

Step 1: The publication server whose login information you want to save, change, or delete in the server login file must be running before you can make any changes to the file. See Step 1 of Section 5.2.1 for directions on starting the publication server.

Step 2: Click the secondary mouse button on the Publication Server node and choose Update. The Update Publication Server dialog box appears.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

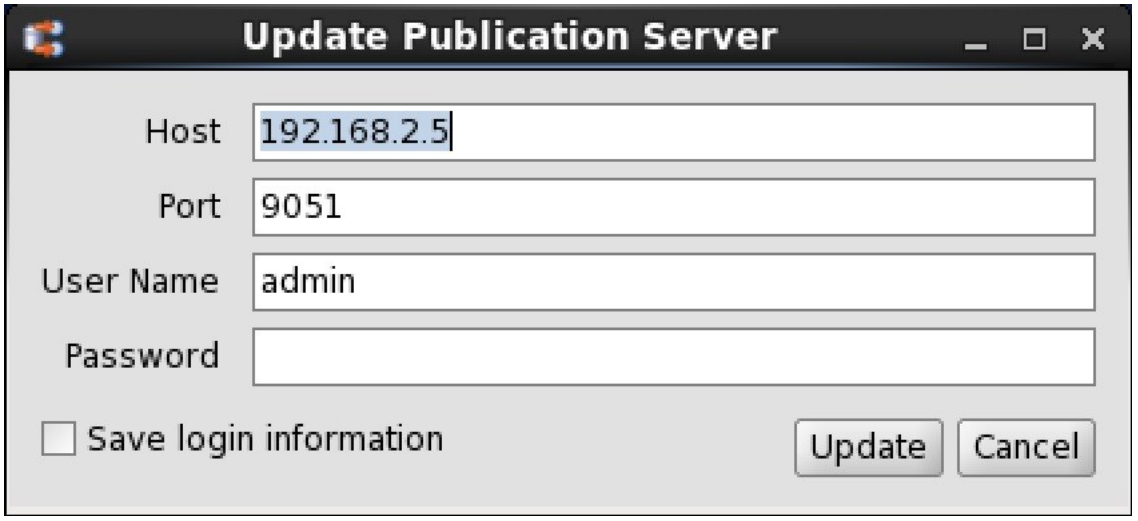


Figure 7-59 - Update Publication Server dialog box

Step 3: Complete the fields in the dialog box according to your purpose for updating the server login file:

- If the publication server now runs on a host with a different IP address or port number than what is shown in the dialog box, enter the correct information. You must also enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address you entered in the Host field. Check the Save Login Information box if you want the new login information saved in the server login file, otherwise leave the box unchecked in which case, access to the publication server is available for the current session, but subsequent sessions will require you to register the publication server again.
- If you want to delete previously saved login information, make sure the network location shown in the dialog box is still correct. Re-enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address in the Host field. Leave the Save Login Information box unchecked. Access to the publication server is available for this session, but subsequent sessions will require you to register the publication server again.
- If you want to save the current login information shown in the dialog box, make sure the network location shown in the dialog box is correct. Re-enter the admin user name and password saved in the xDB Replication Configuration file that resides on the host identified by the IP address in the Host field. Check the Save Login Information box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

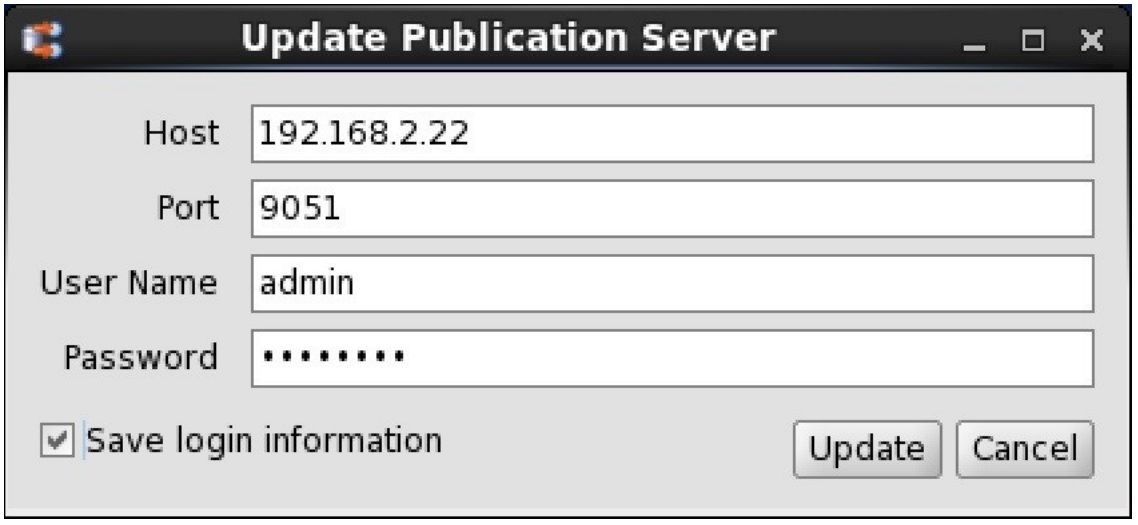


Figure 7-60 - Updated publication server location

Step 4: Click the Update button. If the dialog box closes, then the update to the server login file was successful. Click the Refresh icon in the xDB Replication Console tool bar to show the updated Publication Server node.

If an error message appears after clicking the Update button, the server login file is not modified. Investigate and correct the cause of the error. Repeat steps 1 through 4.

7.6.1.2 Subscription Server Network Location

Note: This section applies only to single-master replication systems.

Part of the metadata stored for each publication server is the network location of subscription servers that manage subscriptions associated with the publication server's publications.

This network information enables the publication server to communicate with the subscription server.

If the network location of a subscription server changes after subscriptions have been created, the publication server metadata must be updated with the new network location, otherwise a communication failure occurs between the publication server and the subscription server that is made apparent by the following message that appears when you open the xDB Replication Console.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

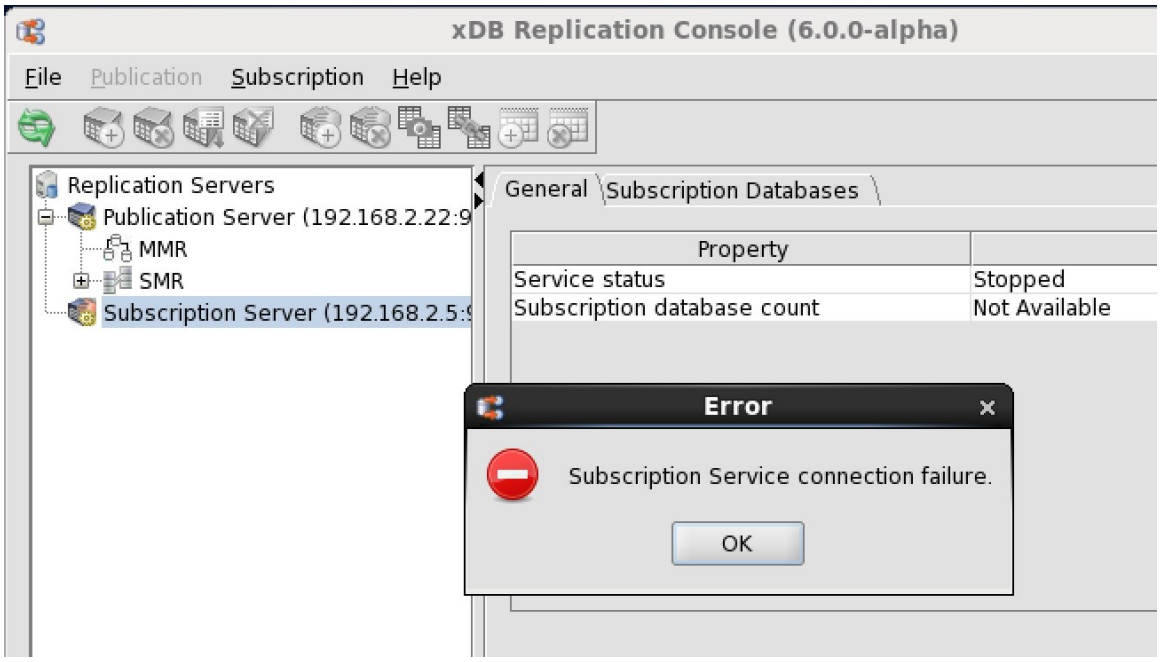


Figure 7-61 - Subscription server connection failure

The following are the steps to update the subscription server network location within a publication server's metadata.

Step 1: The publication server whose metadata you want to change must be running. See Step 1 of Section 5.2.1 for directions on starting the publication server.

Step 2: Click the secondary mouse button on the Publication Server node and choose Update Subscription Servers. The Update Subscription Servers dialog box appears.

Note: If the error message box reappears, click the OK button and repeat Step 2.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

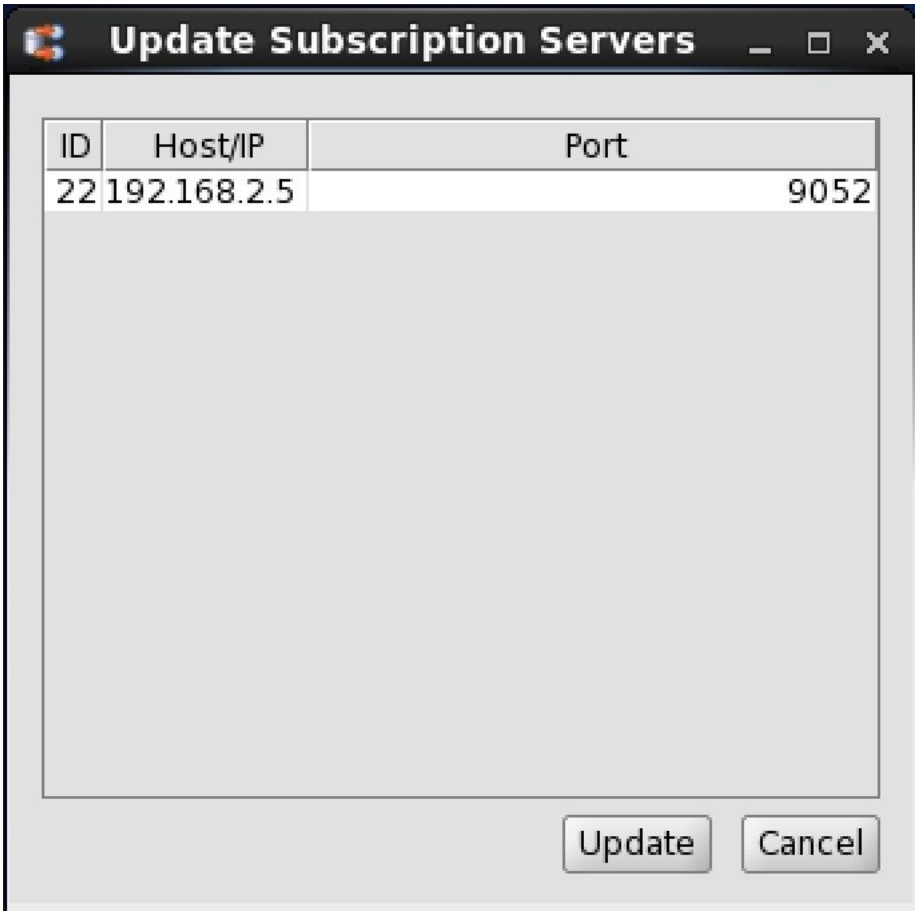


Figure 7-62 - Update Subscription Servers dialog box

Step 3: Enter the new network location for each subscription server in the list whose network location has changed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

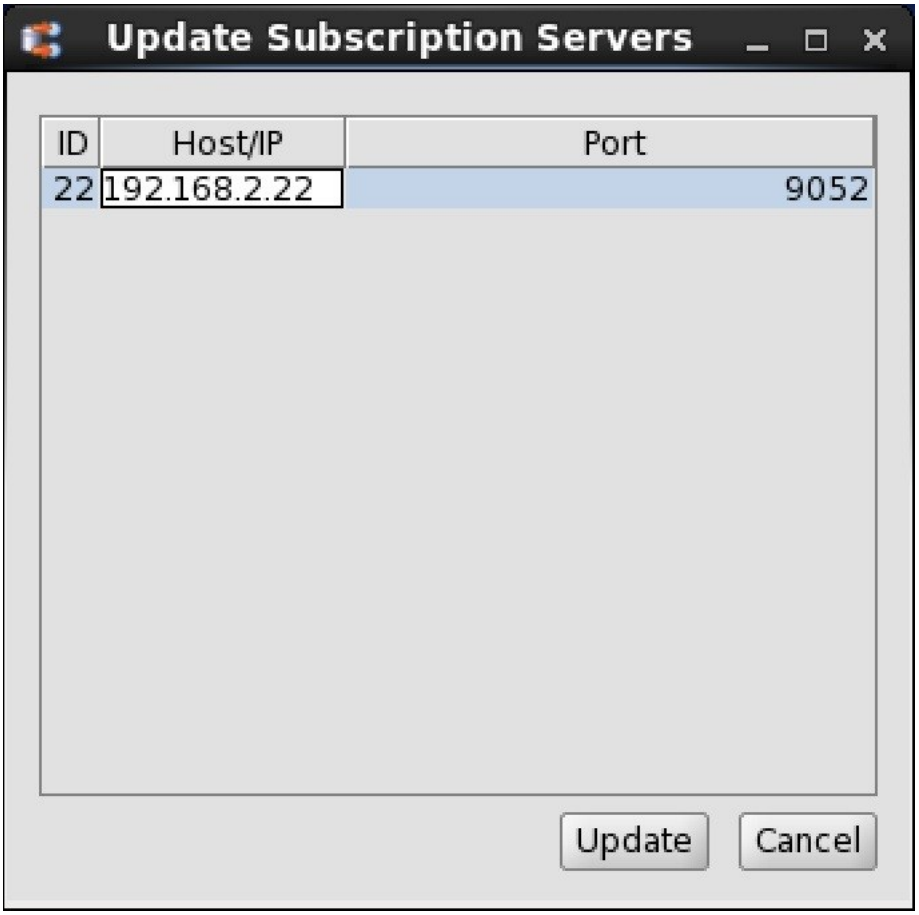


Figure 7-63 - Updated subscription server location

Step 4: Click the Update button. If the dialog box closes, then the update to the publication server's metadata was successful.

If an error message appears, investigate and correct the cause of the error. Repeat steps 1 through 4.

Step 5: If the subscription server with the new network location manages subscriptions associated with publications in other publication servers, repeat steps 1 through 4 for these other publication servers.

7.6.2 Updating a Publication Database

When you create a publication database definition, you save the publication database server's network location (IP address and port number), the database identifier, a database login user name, and the user's password in the control schema accessed by the publication server. This login information is used whenever a session needs to be established with the publication database. See Section 5.2.2 for information on creating a

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

publication database definition for a single-master replication system. See sections 6.2.2 and 6.3 for a multi-master replication system.

The steps described in this section show you how to update the publication database login information stored in the control schema should any of these attributes of the actual, physical database change.

Note: Depending upon the database type (Oracle, SQL Server, or Postgres), certain attributes must not be changed. You must not change any attribute that alters access to the schema where the control schema objects were created when you originally added this publication database definition. See Section 5.2.4 for the location of the control schema objects.

Attributes you must not change include the following:

- The Oracle login user name of an Oracle publication database as the control schema objects already reside in this Oracle user's schema
- The database server network location if the new network location references a database server that does not access the publication database where the control schema objects are stored
- The database identifier if the new database identifier references a different physical database than where the control schema objects are stored

Attributes you may change include the following:

- The login user name's password to match a changed database user password
- The database server network location if the corresponding location change was made to the database server that accesses the publication database
- The database identifier such as the Oracle service name, SQL Server database name, or Postgres database name if the corresponding name change was made on the database server

Step 1: Make sure the database server that you ultimately wish to save as the publication database definition is running and accepting client connections.

Step 2: Make sure the publication server whose node is the parent of the publication database definition you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 3: Select the Publication Database node corresponding to the publication database definition that you wish to update.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

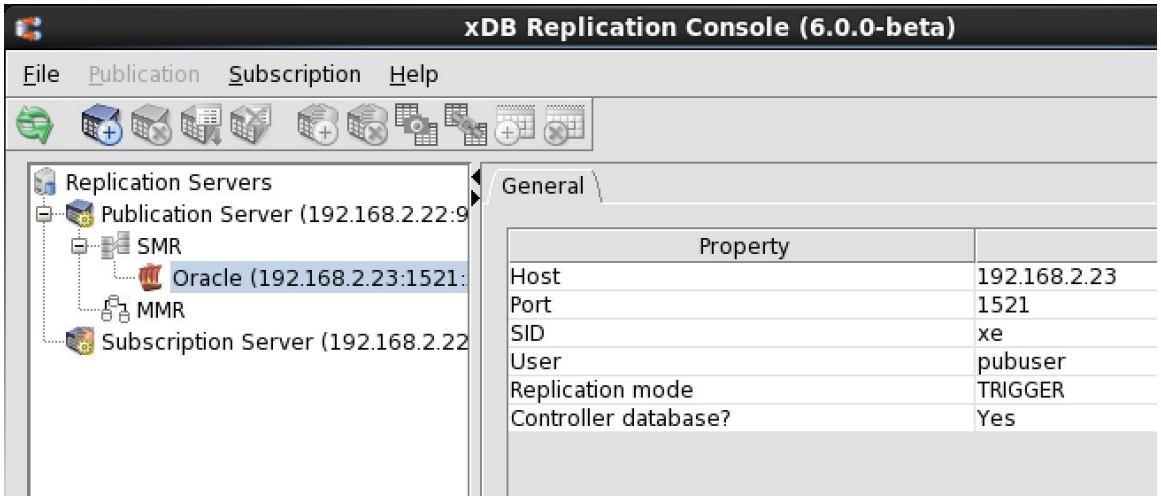


Figure 7-64 - Selecting a publication database definition for update

Step 4: From the Publication menu, choose Publication Database, and then choose Update Database. Alternatively, click the secondary mouse button on the Publication Database node and choose Update Database. The Update Database Source dialog box appears.

Step 5: Enter the desired changes. See Step 3 of Section 5.2.2 for the precise meanings of the fields for a single-master replication system. See sections 6.2.2 and 6.3 for a multi-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

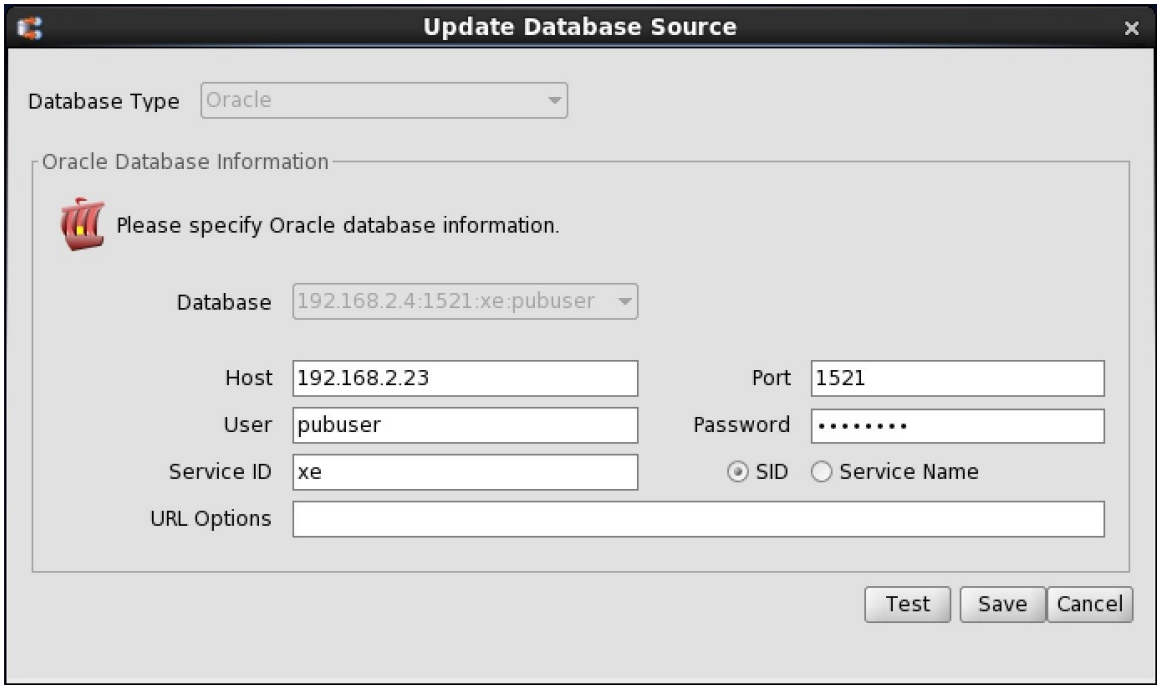


Figure 7-65 - Update Database Source dialog box for a single-master replication system

Step 6: Click the Test button. If Test Result: Success appears, click the OK button, then click the Save button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

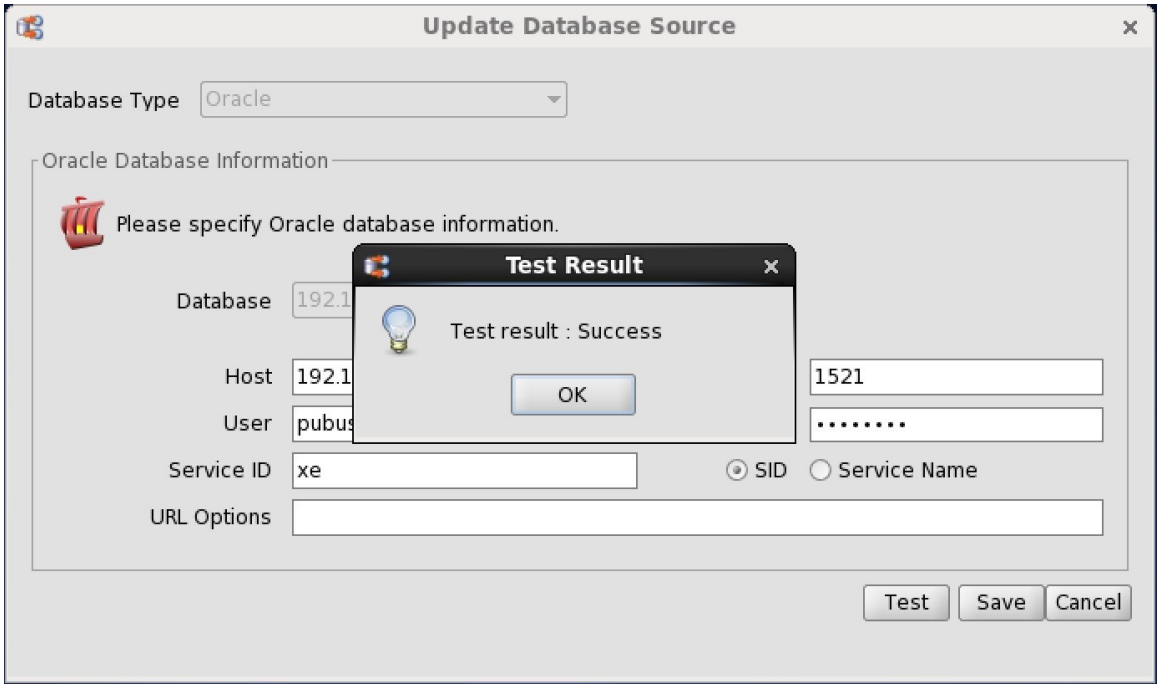


Figure 7-66 - Successful publication database test

If an error message appears investigate the cause of the error, correct the problem, and repeat steps 1 through 6.

Step 7: Restart the publication server. See Section 5.2.1 for directions on restarting the publication server.

Step 8: Click the Refresh icon in the xDB Replication Console tool bar to show the updated Publication Database node and any of its publications.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

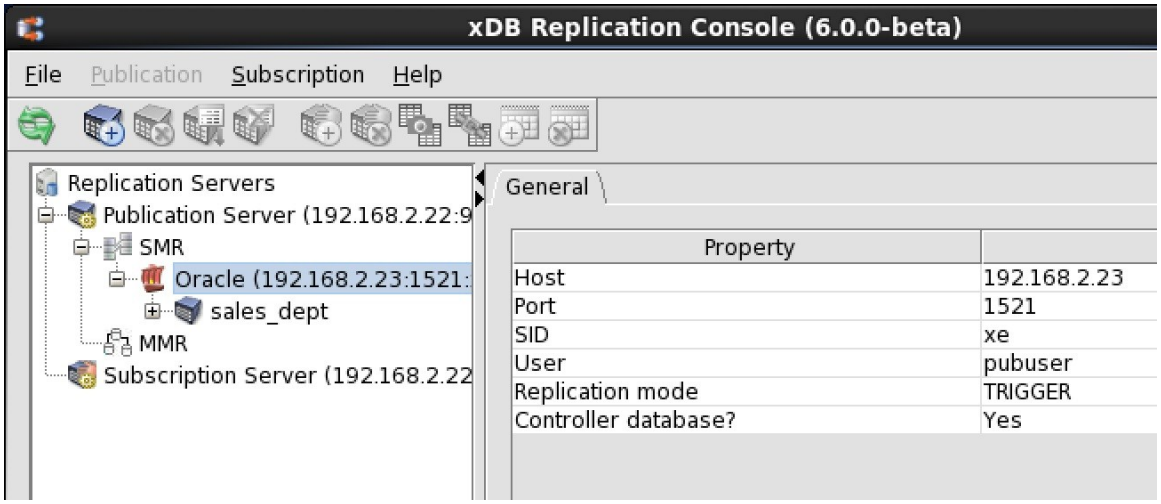


Figure 7-67 - Updated publication database

7.6.3 Updating a Publication

Existing publications can be updated in the following ways:

- Tables can be added to the publication
- Tables can be removed from the publication
- Filter rules can be updated on publication tables

7.6.3.1 Adding Tables to a Publication

For a single-master replication system, you can add tables to a publication, even while there are existing subscriptions associated with the publication.

Similarly for a multi-master replication system, you can add tables to a publication while there are additional master nodes in the replication system.

The following are the steps to add tables to an existing publication.

Step 1: Make sure the publication server whose node is the parent of the publication you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Publication node of the publication to which you wish to add tables.

Step 2 (For MMR only): Select the Publication node under the Publication Database node representing the master definition node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

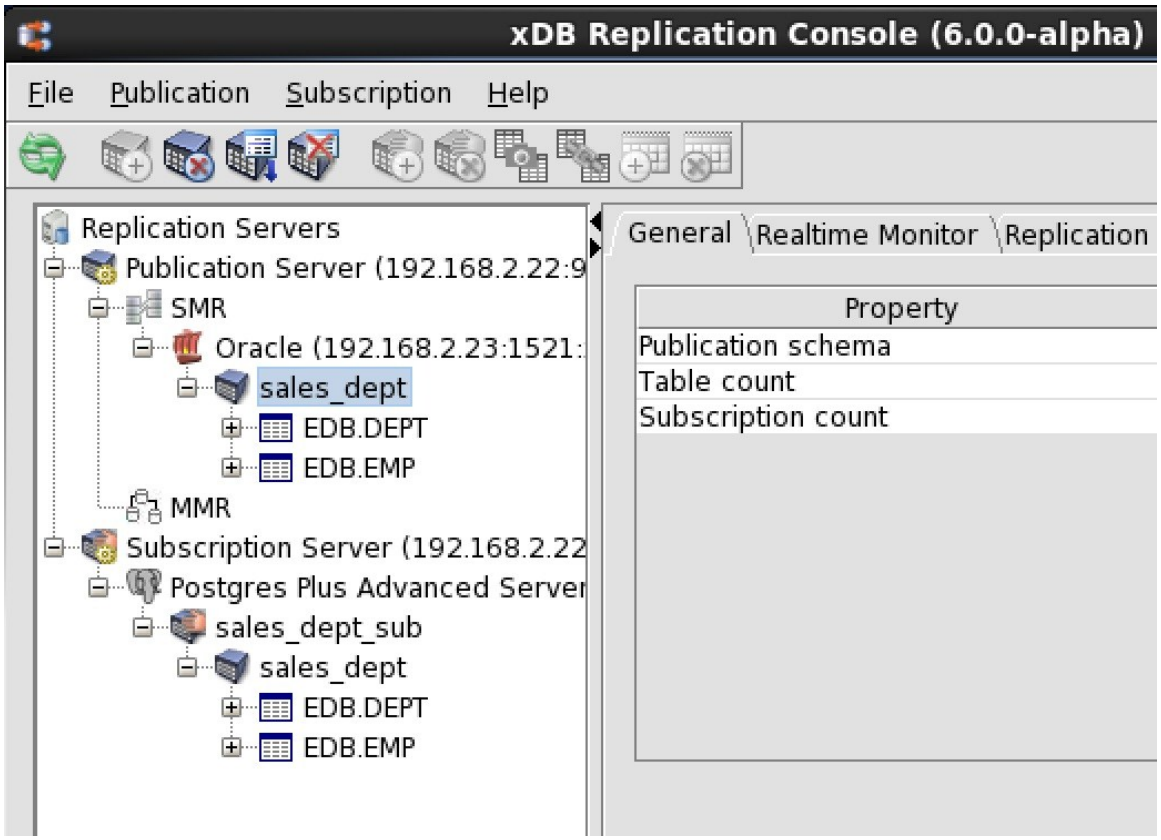


Figure 7-68 - Selecting a publication to which to add tables

Step 3: Open the Add Tables dialog box in any of the following ways:

- From the Publication menu, choose Update Publication, then Add Tables.
- Click the secondary mouse button on the Publication node, choose Update Publication, and then choose Add Tables.
- Click the primary mouse button on the Add Publication Tables icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

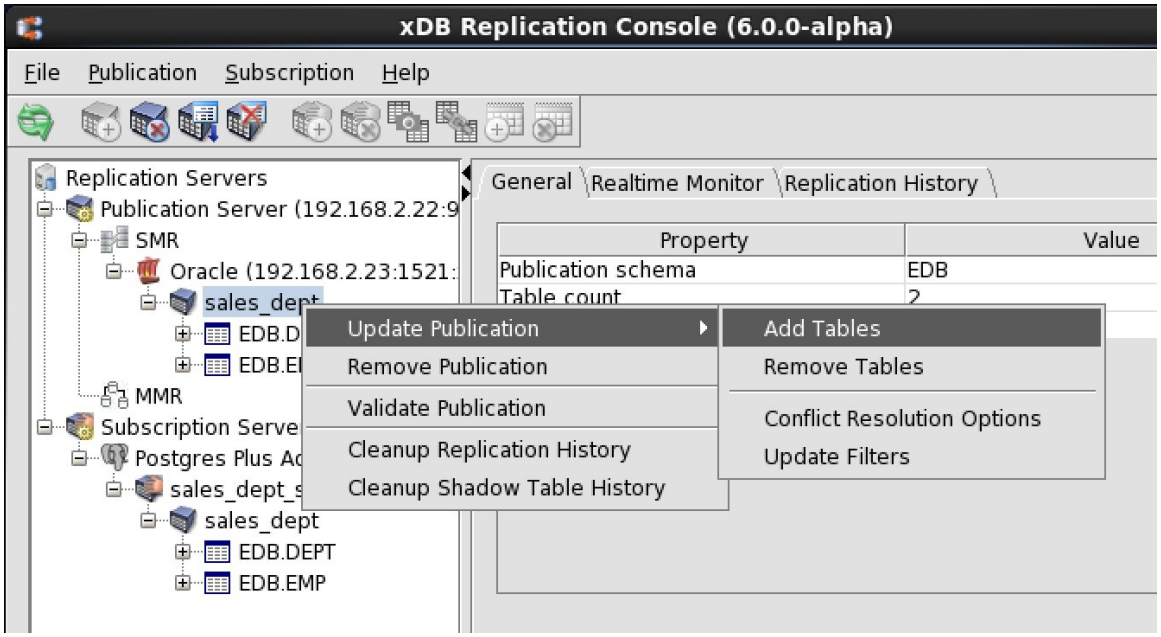


Figure 7-69 - Opening the Add Tables dialog box

Step 4: Fill in the following fields in the Add Tables tab of the Add Tables dialog box:

- **Add.** Check the boxes next to the table names from the Available Tables list that are to be added to the publication. If the publication is a snapshot-only publication, then views would appear in the Available Tables list as well. The Available Tables list contains only tables and views that are not already members of other publications under the same Publication Database node. Alternatively or in addition, click the Use Wildcard Selection button to use wildcard pattern matching for selecting tables to be added to the publication.
- **Select All.** Check this box if you want to include all tables and views in the Available Tables list in the publication.
- **Use Wildcard Selection.** Click this button to use the wildcard selector to choose tables for the publication. See Section 7.1 for information on the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

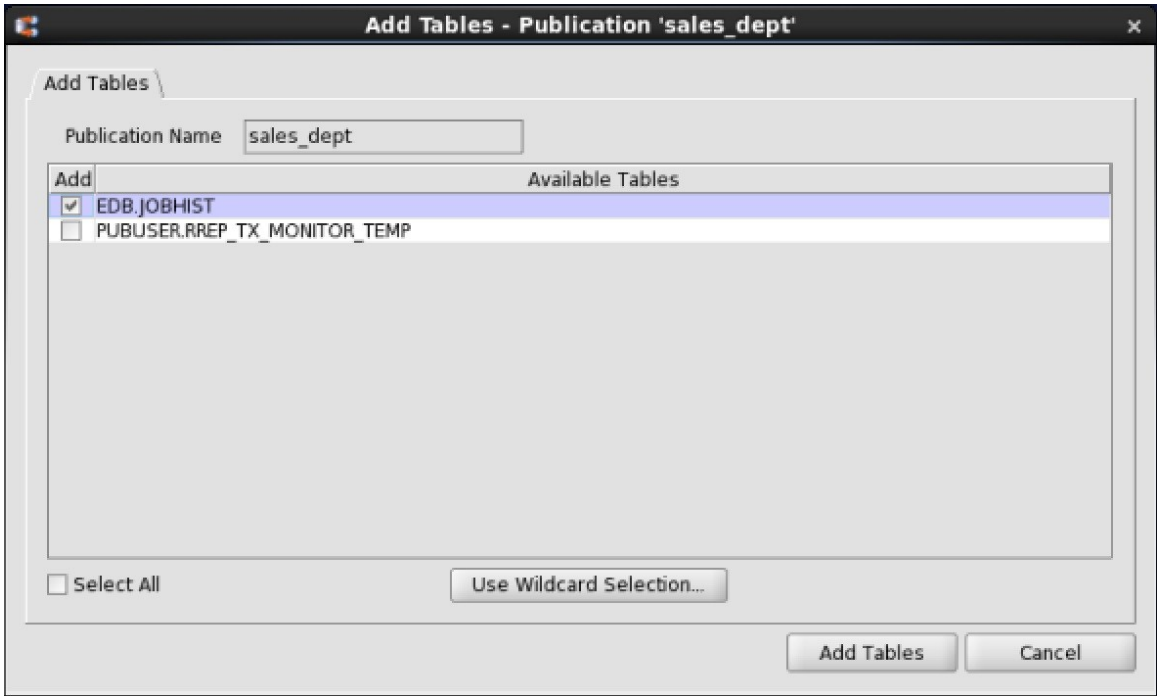


Figure 7-70 - Add Tables dialog box

If you wish to omit certain rows of the publication tables or views from being replicated follow the directions in the next step to create a filter, otherwise go on to Step 6.

Step 5 (Optional): If you want to filter the rows of the publication tables or views, click the Table Filters tab. Define filter rules by entering a unique, descriptive filter name and an appropriate SQL `WHERE` clause in the Filter dialog box to select the rows you want to replicate.

For a single-master replication system, see Section 5.2.3 for information on defining table filters on a publication table.

For a multi-master replication system, see Section 6.2.3.

Step 6 (For SMR only): Click the Add Tables button. If Publication Updated Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

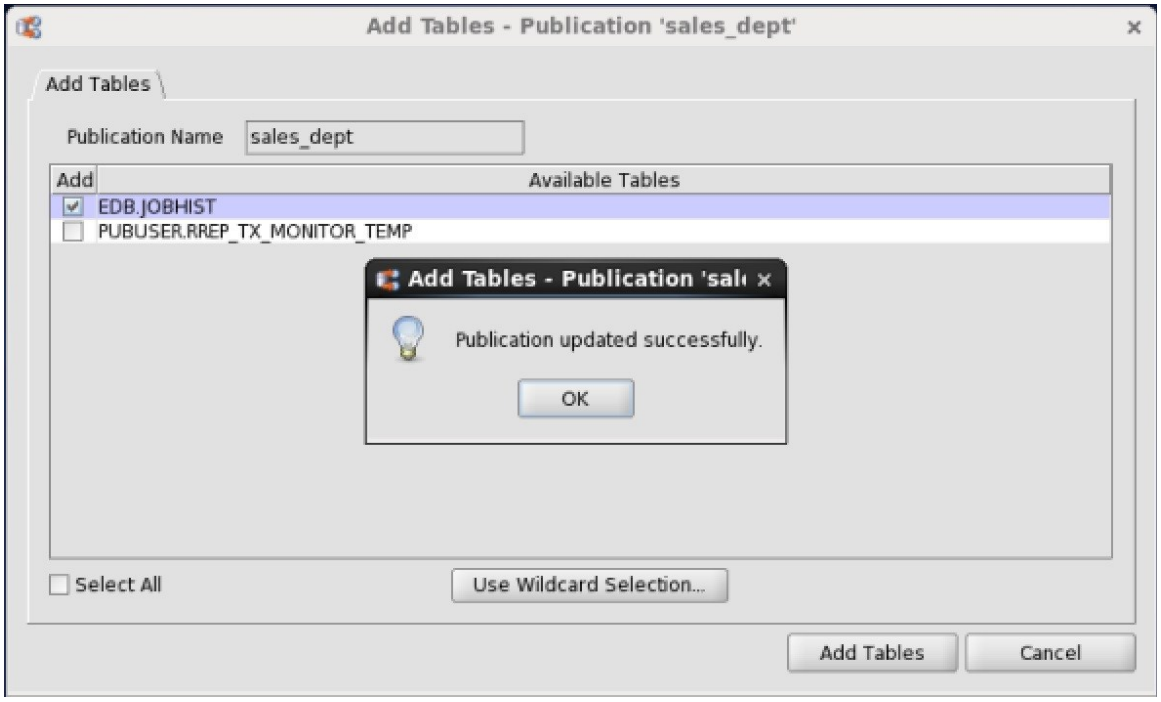


Figure 7-71 - Successfully added tables to publication

Step 6 (For MMR only): Click the Add Tables button. The Data Sync Check dialog box appears warning you that synchronization replication is performed before the table is added.

If you wish to perform synchronization at some later point in time then add the table, click the No button.

If you wish to proceed, click the Yes button. If Publication Updated Successfully appears, click the OK button, otherwise investigate the error and make the necessary corrections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

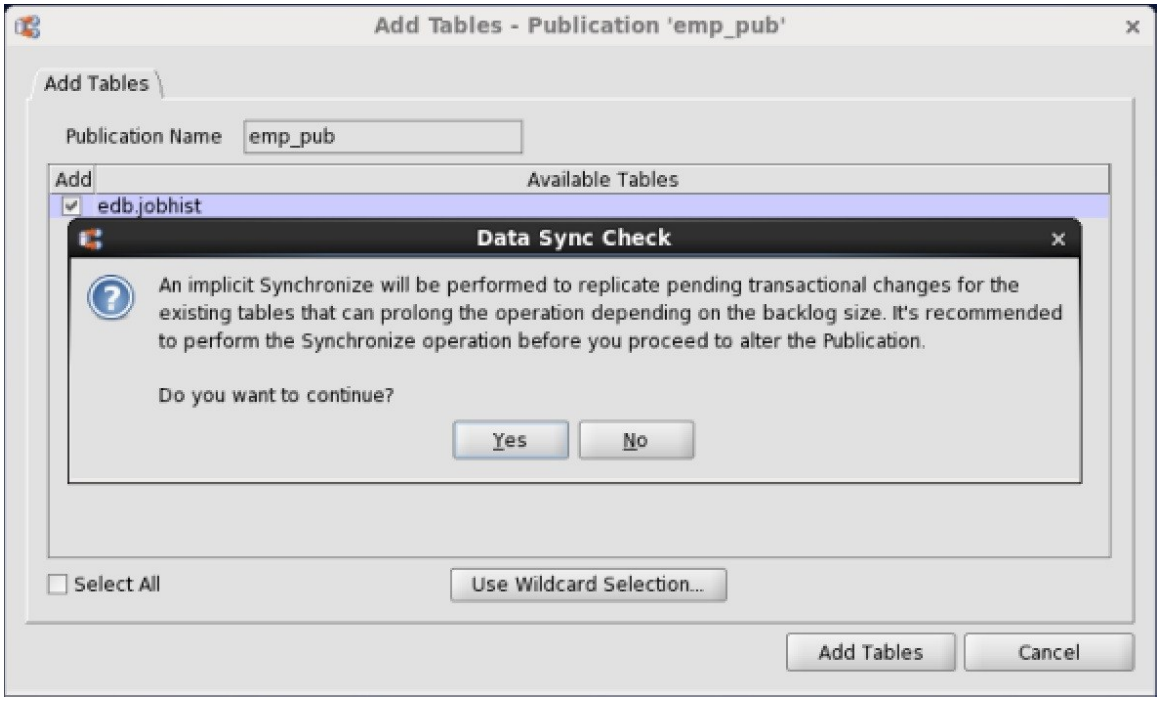


Figure 7-72 - Data Sync Check dialog box

Step 7: The replication tree appears as follows with the newly added table under the Publication node. Click the Refresh icon. The newly added table appears under the Subscription nodes of a single-master replication system or the additional master nodes of a multi-master replication system.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

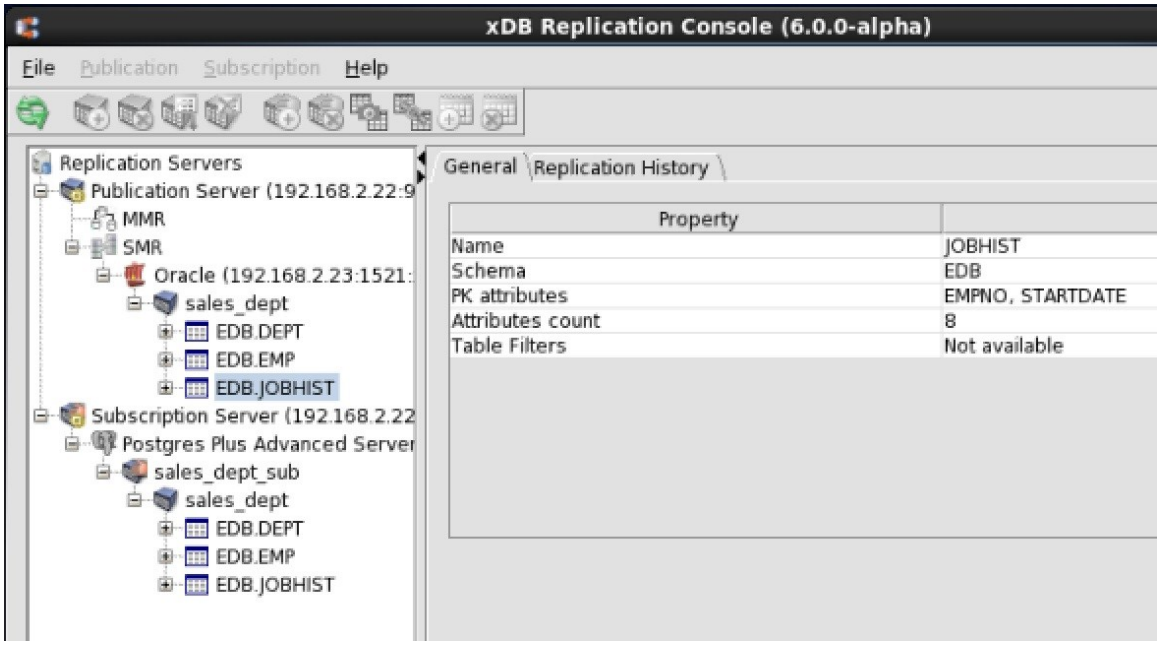


Figure 7-73 - Publication and subscription with added table

Step 8 (For MMR only): If you want to modify or see the default conflict resolution options assigned to the newly added table, follow the directions in Section 6.8.

Step 9 (Optional): If you defined table filters on the newly added table, and you wish to use these filters on any subscriptions or master nodes, you must enable the filters on the table within the desired subscriptions or master nodes.

For a single-master replication system, see Section 5.5.4 for directions on enabling table filters on a subscription.

For a multi-master replication system, see Section 6.9 for directions on enabling table filters on a master node.

7.6.3.2 Removing Tables from a Publication

You can remove one or more tables from a publication, but only if the following condition is true:

- The tables to be removed are not parent tables referenced by foreign key constraints of child tables that are not selected for removal as well.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

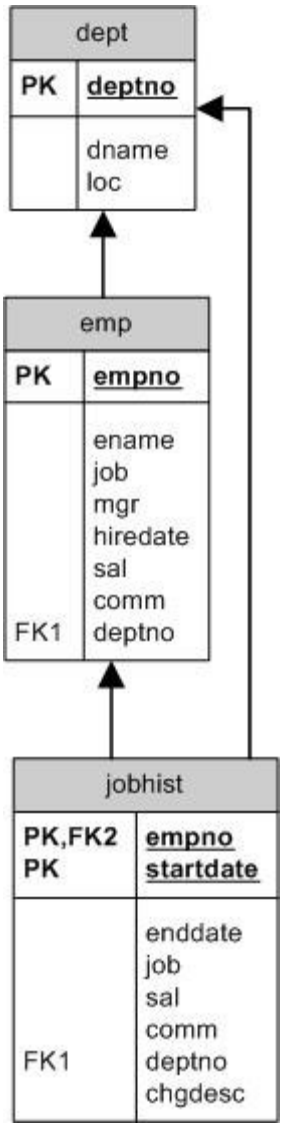


Figure 7-74 - Entity relationship diagram of tables with foreign key constraints

In the preceding entity relationship diagram, the `emp` table has a foreign key constraint referencing the `dept` table, and the `jobhist` table has two foreign key constraints. One constraint references the `emp` table and the other references the `dept` table.

If all three tables are in the publication, then you can remove the following combinations of tables:

- Remove the `jobhist` table only.
- Remove both the `jobhist` table and the `emp` table.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 1: Make sure the publication server whose node is the parent of the publication you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Publication node of the publication from which you wish to remove tables.

Step 2 (For MMR only): Select the Publication node under the Publication Database node representing the master definition node.

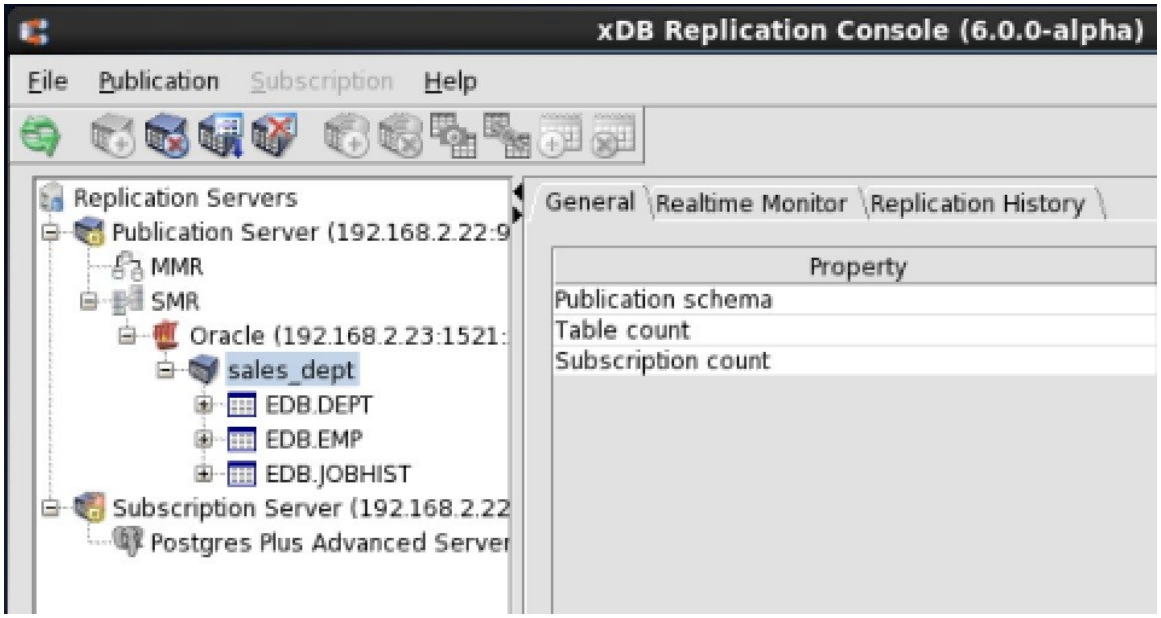


Figure 7-75 - Selecting a publication from which to remove tables

Step 3: Open the Remove Tables dialog box in any of the following ways:

- From the Publication menu, choose Update Publication, then Remove Tables.
- Click the secondary mouse button on the Publication node, choose Update Publication, and then choose Remove Tables.
- Click the primary mouse button on the Remove Publication Tables icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

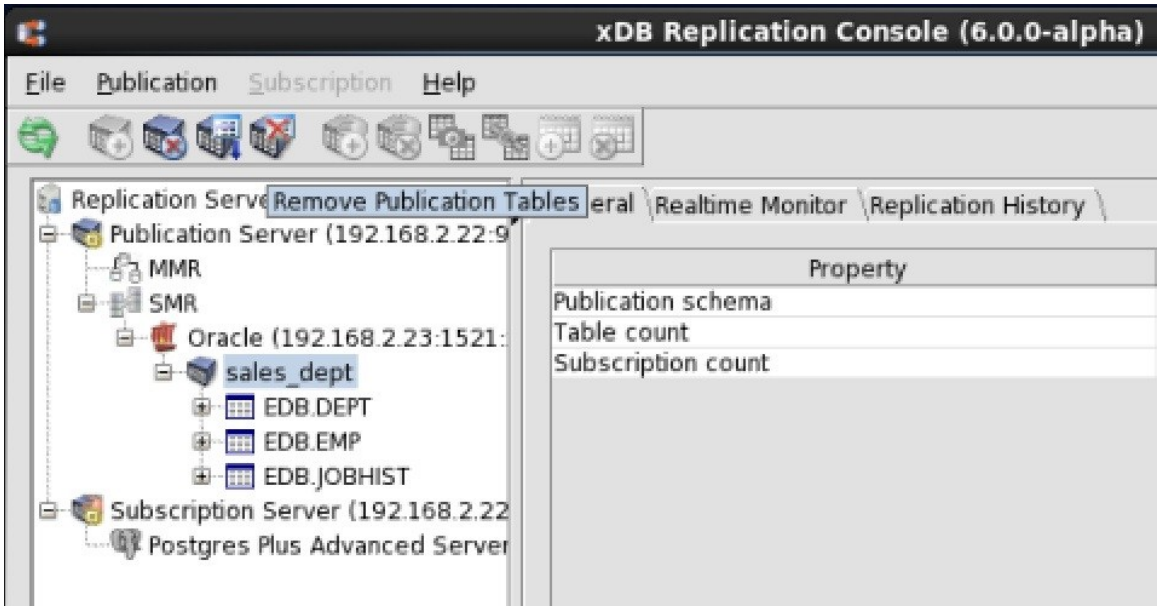


Figure 7-76 - Opening the Remove Tables dialog box by clicking the toolbar icon

Step 4: Use the Remove Tables dialog box as follows:

- **Remove.** Check the boxes next to the table names from the Available Tables list that are to be removed from the publication. If the publication is a snapshot-only publication, then views would appear in the Available Tables list as well. Alternatively or in addition, click the Use Wildcard Selection button to use wildcard pattern matching for selecting tables to be removed from the publication.
- **Use Wildcard Selection.** Click this button to use the wildcard selector to choose tables to remove from the publication. See Section 7.1 for information on the wildcard selector.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

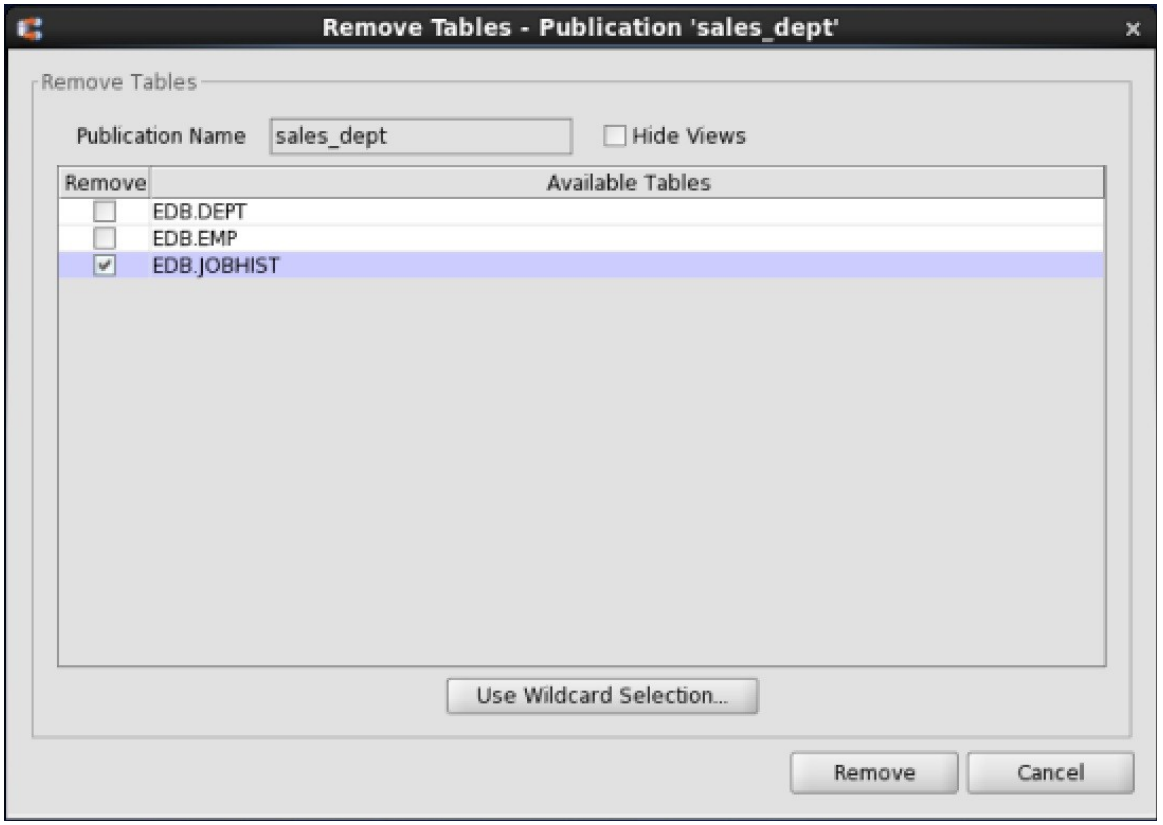


Figure 7-77 - Remove Tables dialog box

Step 5: Click the Remove button, then click the Yes button of the confirmation box.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

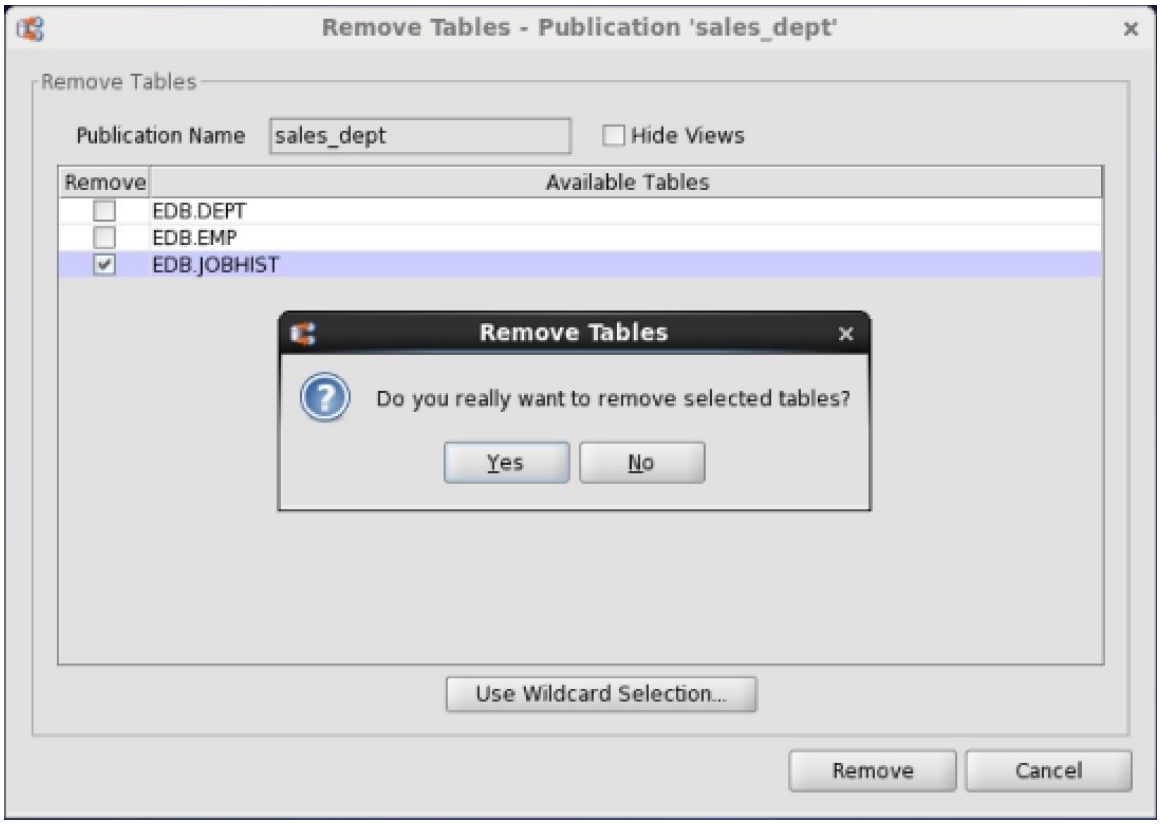


Figure 7-78 - Remove Tables confirmation

Step 6: Click the OK button in response to Tables Removed Successfully.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

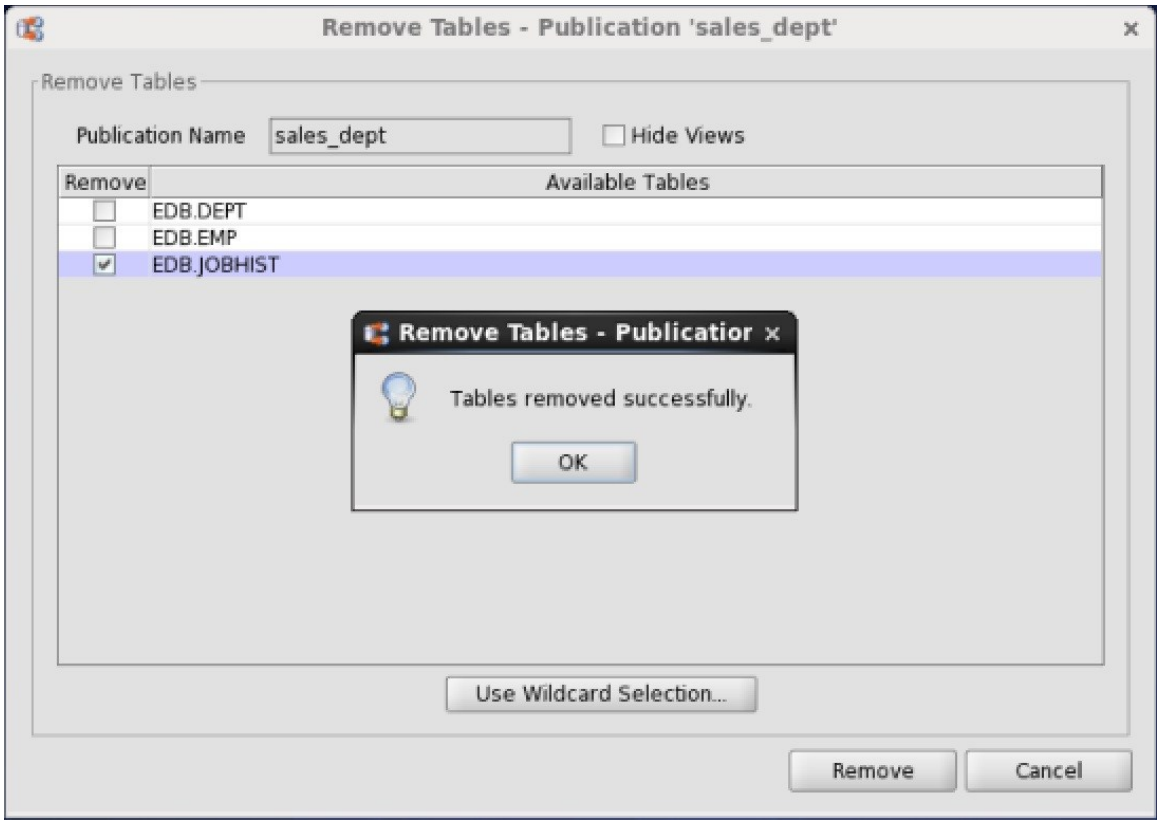


Figure 7-79 - Successfully removed tables from publication

The replication tree appears as follows without the removed table under the Publication node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

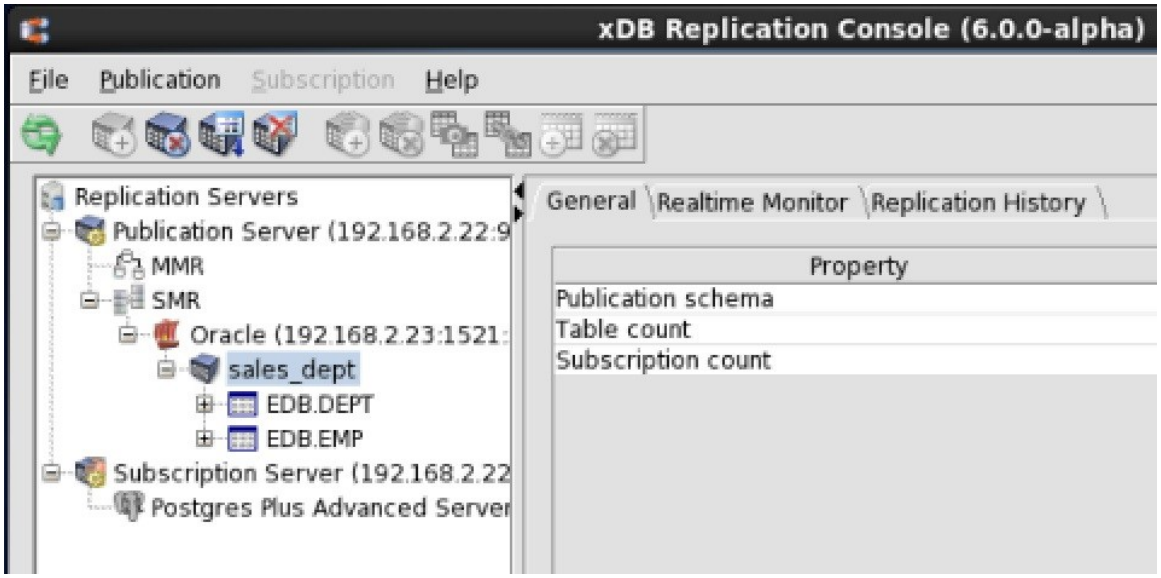


Figure 7-80 - Publication minus removed table

7.6.4 Updating the Set of Available Table Filters in a Publication

Once a set of available table filters has been defined in the publication of a single-master replication system or a multi-master replication system, the set can subsequently be updated by adding new filter rules, removing existing filter rules, or modifying existing filter rules.

Note: See Section 2.2.12.3 for table setup requirements for a log-based replication system as well as general restrictions on the use of table filters.

See Section 5.2.3 for information on using table filters in a single-master replication system and Section 6.2.3 for a multi-master replication system.

Updating the set of available table filters in a publication has the following implications:

- After you have added new filter rules to a publication, you must then enable these newly added filter rules on the subscriptions or master nodes on which you want these filter rules to have an effect. See Section 5.5.4 for directions on enabling filter rules on a subscription. See Section 6.9 for enabling filter rules on a master node.
- After you have removed existing filter rules from a publication, the removed filter rules are automatically deleted from any associated subscription or master node on which they had been enabled. That is, you do not need to modify the subscriptions or master nodes to disable the filter rules once they have been removed from the publication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- After you have modified existing filter rules (that is, changed the filter name or filter clause), the modifications are automatically applied to any subscriptions or master nodes on which the filter rules had been enabled. That is, you do not need to make any changes in the associated subscriptions or master nodes.

After your updates to the set of available table filters in the publication have been completed, and the filter rules have been enabled or disabled on the target subscriptions or master nodes, a snapshot replication should be performed on any subscription or master node affected by an updated filter rule to ensure that the content of the targeted subscription tables or master node tables is consistent with the current set of filter rules enabled on those tables.

The following are the steps to update the set of available filter rules in a publication.

Step 1: Make sure the publication server whose node is the parent of the publication you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Publication node of the publication in which you wish to update the set of available table filters.

Step 2 (For MMR only): Select the Publication node under the Publication Database node representing the master definition node.

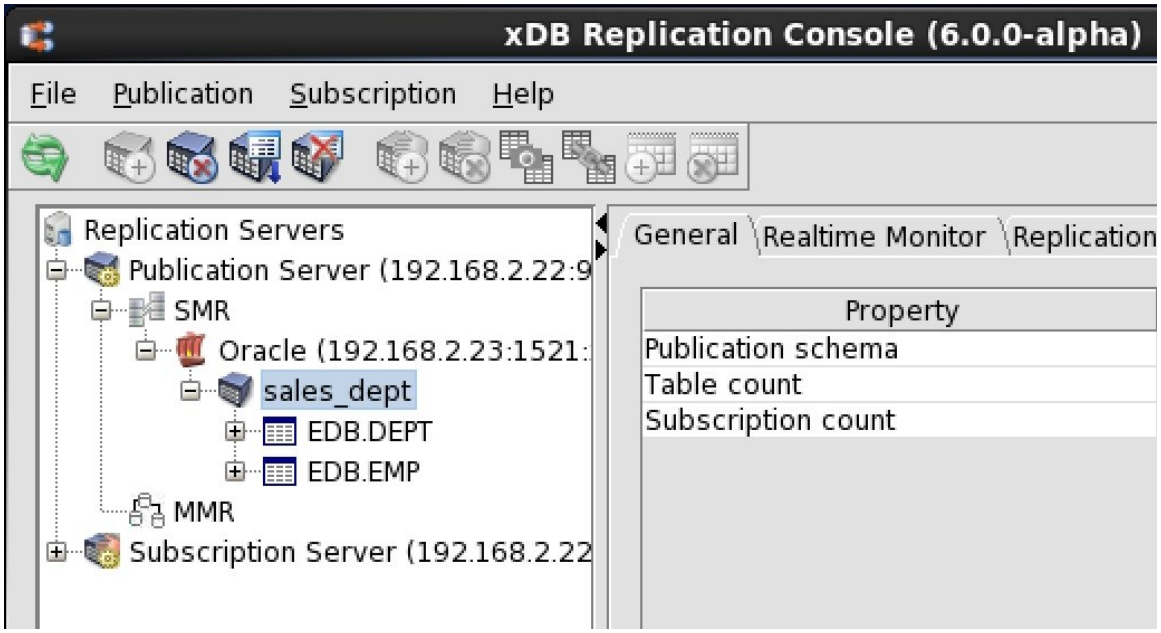


Figure 7-81 - Selecting a publication in which to update the set of available table filters

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: Open the Update Filters dialog box in any of the following ways:

- From the Publication menu, choose Update Publication, then Update Filters.
- Click the secondary mouse button on the Publication node, choose Update Publication, and then choose Update Filters.

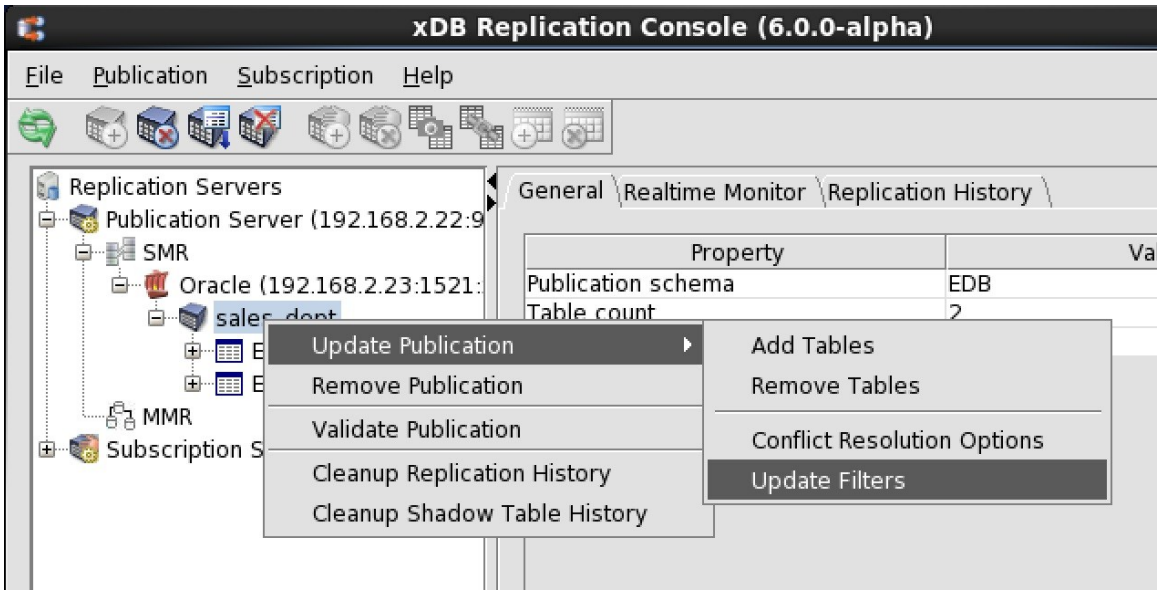


Figure 7-82 - Opening the Update Filters dialog box

Step 4: The set of all available filter rules defined in the publication are listed under the Table Filters tab.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

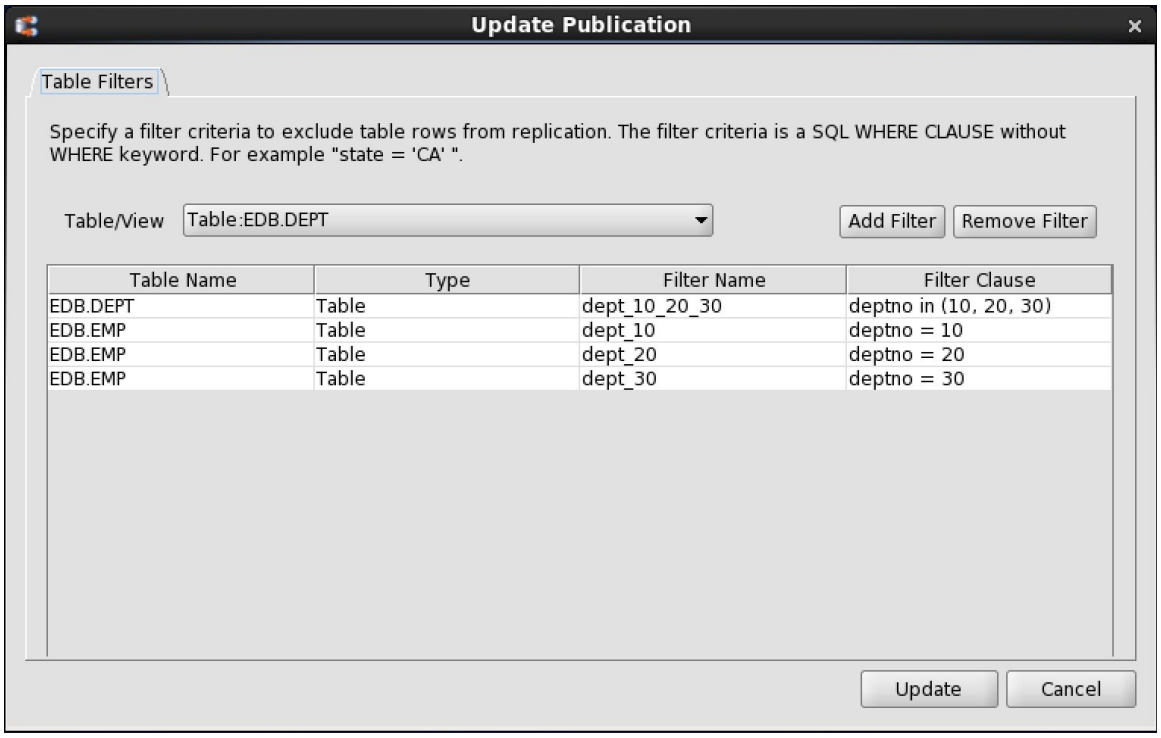


Figure 7-83 - Set of all available filter rules

To add a new filter rule, from the Table/View drop-down list select the table or view for which you wish to add a filter and click the Add Filter button. Fill in the information in the dialog box that appears. (See Section 5.2.3 for more details on adding individual filter rules in a single-master replication system. See Section 6.2.3 for a multi-master replication system.)

To remove a filter rule, click the primary mouse button on the filter rule you wish to remove so the entry is highlighted and click the Remove Filter button.

You may also modify the filter name or filter clause of a filter rule listed in the Table Filters tab by double-clicking on the cell of the filter name or filter clause you wish to change. When the cursor appears in the cell, enter the text for the desired change.

When you are satisfied with the updated set of available table filters, click the Update button.

Step 5: A confirmation box appears presenting a warning message and a recommendation to perform a snapshot replication to any subscription or master node on which you intend to enable the change in filtering criteria.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Click the Ok button in the confirmation box to proceed with the update to the filter rules. Click the Cancel button to return to the Filter Rules tab if you wish to modify your filter rule updates.

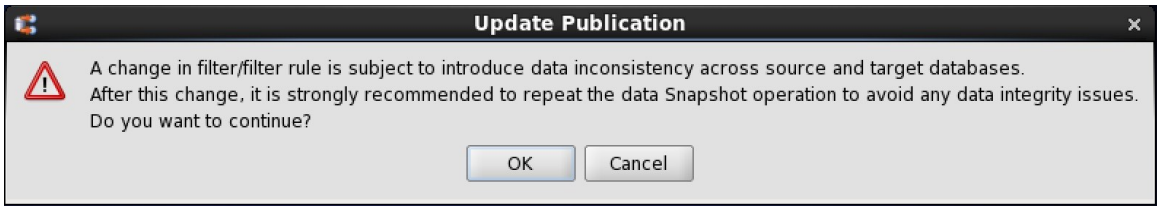


Figure 7-84 - Change filter rule confirmation

Step 6: You may selectively enable any new filter rules to the corresponding tables of the associated subscriptions or master nodes. See Section 5.5.4 for information on enabling table filters on a subscription. See Section 6.9 for enabling table filters on a master node.

7.6.5 Validating a Publication

Once a publication is created, **do not directly change the definitions of the tables belonging to the publication**. Doing so may cause a failure during the replication process. Examples of table definitions that must not be altered include:

- Adding or removing columns to a table
- Renaming columns
- Changing the data types of columns
- Changing the lengths of columns
- Changing a not nullable column to nullable or a nullable column to not nullable
- Adding or removing uniqueness constraints
- Adding or removing check constraints

In a single-master replication system, xDB Replication Server does not propagate table definition changes to the subscription tables once the subscription tables are created. Rows that may be allowed in an altered publication table may be illegal in the unaltered subscription table and will cause an error during replication.

Similarly, in a multi-master replication system, table definition changes are not propagated from one master node to another except when a new master node is added, and you choose to replicate the schema definition from the master definition node.

In addition, for synchronization replication with the trigger-based method, triggers are generated on the publication tables that use certain attributes of these tables. If the table definition is changed, the trigger may no longer function properly.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: Do not change the triggers generated by xDB Replication Server. If it becomes necessary to regenerate the triggers, you must remove the associated publication and then recreate the publication.

Note: Certain table definition changes can be made and propagated by xDB Replication Server by using the DDL change replication feature. See Section 7.8 for information on the DDL change replication feature.

If you do not use the DDL change replication feature, then the following general steps must be taken if table definition changes are made.

In a single-master replication system, if changes were made to the definitions of one or more publication tables, the resolution to the problem must be handled on a case by case basis as it depends upon the type of changes that were made. In the worst case scenario, the subscription and publication must be removed and recreated as follows:

- Remove the subscription that is associated with the publication. See Section 5.5.5 for directions to remove a subscription.
- Remove the subscription tables from the subscription database. This is done with SQL `DROP TABLE` statements in the database system.
- Remove the publication. See Section 7.6.6 for directions to remove a publication.
- Re-add the publication. See Section 5.2.3 for directions to add a publication.
- Re-add the subscription. See Section 5.3.3 for directions to add a subscription.

In a multi-master replication system, if changes were made to the definitions of one or more publication tables on one or more master nodes, the resolution to the problem involves:

- Making sure the table definitions are updated on all master nodes so that they are identical, or updating the table definition on the master definition node so it can be replicated to the other master nodes.
- Recreating the publication database definitions of the master nodes.

The general steps are the following:

- Remove the publication database definitions of all master nodes except for the master definition node. See Section 7.6.7 for directions to remove a publication database definition.
- Remove the publication. See Section 7.6.6 for directions to remove a publication.
- Remove the publication database definition of the master definition node. See Section 7.6.7 for directions to remove a publication database definition.
- At this point all of the triggers, shadow tables, and metadata have been removed from the master nodes.
- With respect to the publication table definitions, you can either: 1) update the table definitions on all master nodes so that they are identical, or 2) assume the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

table definitions on the master definition node are up-to-date, and delete the out-of-date table definitions on all other master nodes.

- Re-add the publication database definition for the master definition node. See Section 6.2.2 for directions to add the master definition node.
- Re-add the publication. See Section 6.2.3 for directions to add a publication.
- Re-add additional master nodes. See Section 6.3 for directions to add an additional master node. When creating a master node, uncheck the Replicate Publication Schema check box if you have already created the table definitions on all master nodes. Check the Replicate Publication Schema check box if you want to propagate the table definitions from the master definition node to all other master nodes. A snapshot reloads the master node tables from the master definition node.

7.6.5.1 Validating a Single Publication

xDB Replication Server provides a way to verify that certain characteristics of publication tables have not been altered since the publication was created.

Note: This validation feature is only available for publications using the trigger-based method of synchronization replication. This validation feature is not available for publications using the log-based method of synchronization replication.

The validation operation described here and in Section 7.6.5.2 can check for the following types of table modifications:

- Addition of columns to a table
- Removal of columns from a table
- Renaming of columns

Note: In a multi-master replication system, publication tables in only the master definition node are validated. The validation operation does not check if table definitions have changed in other master nodes.

The following steps show how to validate a single publication:

Step 1: Make sure the publication server whose node is the parent of the publication you wish to validate is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Publication node of the publication you want to validate.

Step 2 (For MMR only): Select the Publication node under the Publication Database node representing the master definition node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: From the Publication menu, choose Validate Publication. Alternatively, click the secondary mouse button on the Publication node and choose Validate Publication.

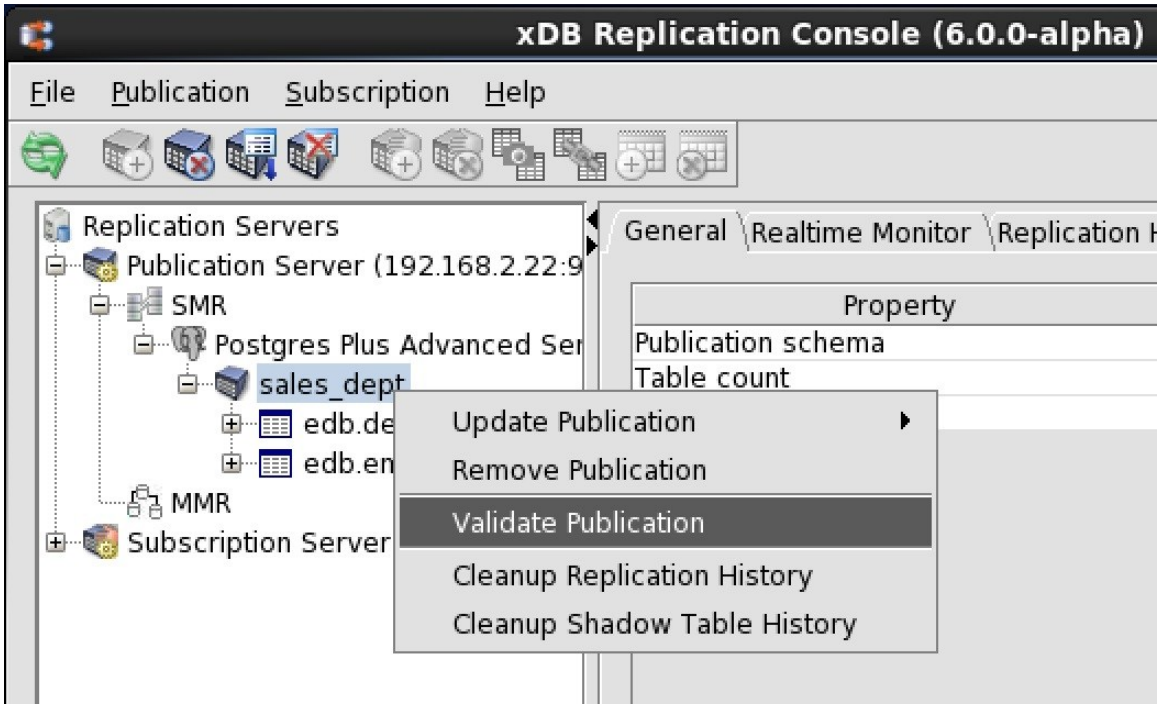


Figure 7-85 - Validating a selected publication

Step 4: If All Schema of Published Tables in Publication '*publication_name*' Are Up-To-Date appears, click the OK button. If an error appears, determine which tables were changed and what changes were made to the table definitions. These issues need to be resolved on a case by case basis as discussed earlier in this section.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

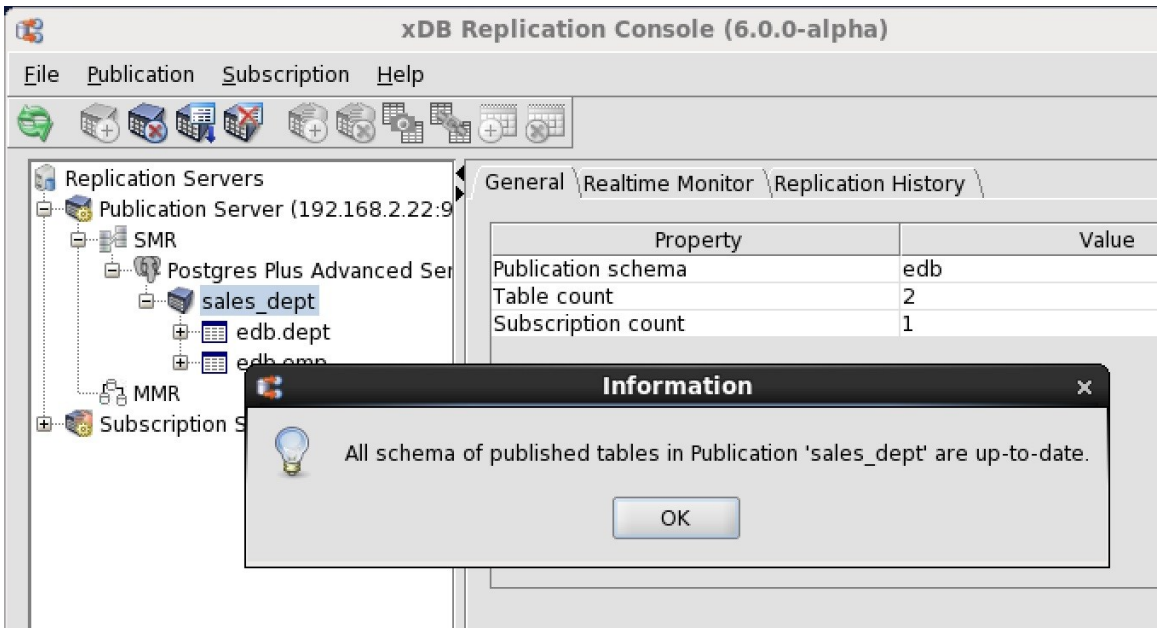


Figure 7-86 - Successful validation of all tables in the selected publication

7.6.5.2 Validating All Publications

All publications under a single Publication Database node can be validated in one operation.

Note: This validation feature is only available for publications using the trigger-based method of synchronization replication. This validation feature is not available for publications using the log-based method of synchronization replication.

Note: In a multi-master replication system, publication tables in only the master definition node are validated. The validation operation does not check if table definitions have changed in other master nodes.

Step 1: Make sure the publication server whose node is the parent of the publications you wish to validate is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2 (For SMR only): Select the Publication Database node under which you want to validate all publications.

Step 2 (For MMR only): Select the Publication Database node representing the master definition node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: From the Publication menu, choose Validate All Publications. Alternatively, click the secondary mouse button on the Publication Database node and choose Validate All Publications.

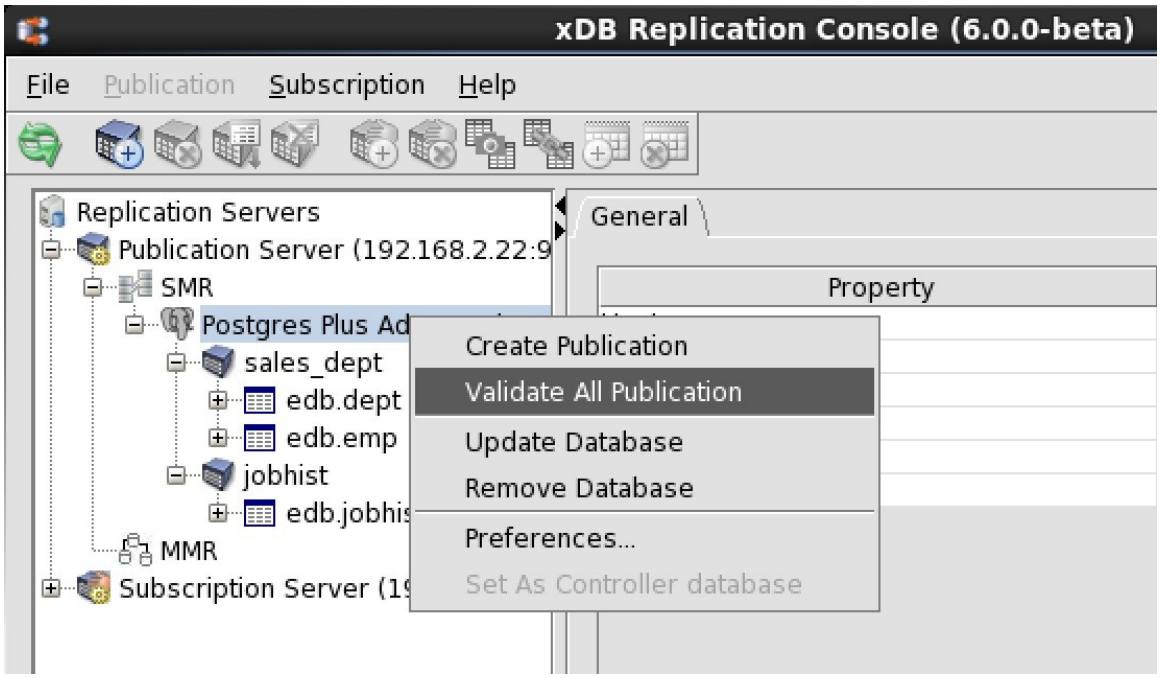


Figure 7-87 - Validating all publications subordinate to a selected publication database

Step 4: If there were no modified tables, click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

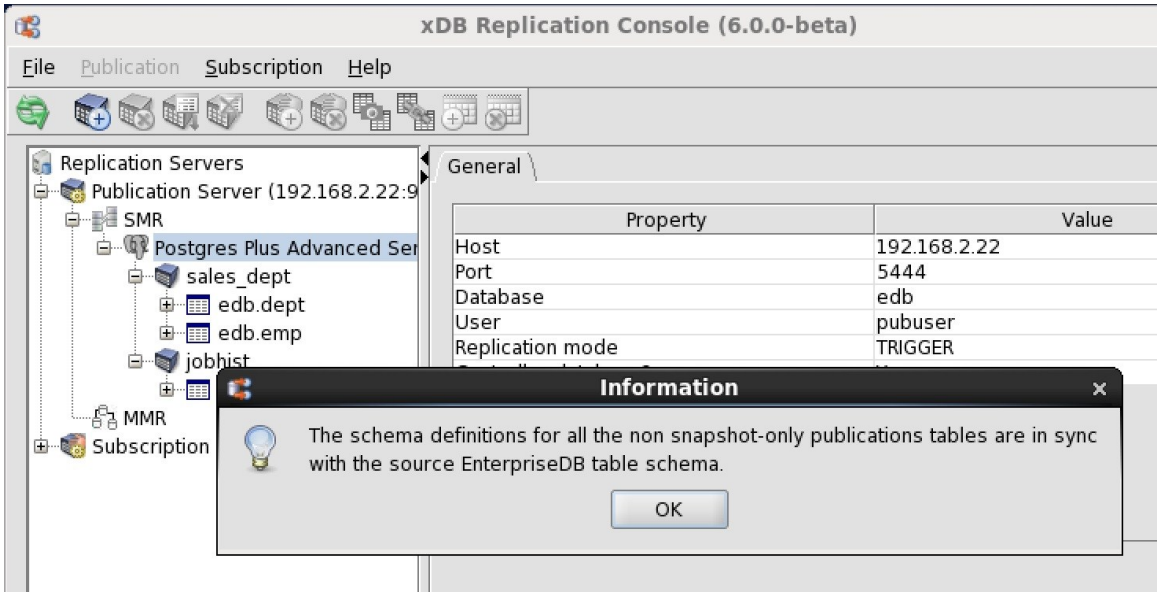


Figure 7-88 - Successful validation of all tables in all publications subordinate to a selected publication database

If there were modified tables, a list of publications that contain the modified tables is displayed. Determine which tables were changed and what changes were made to the table definitions. These issues need to be resolved on a case by case basis as discussed earlier in this section.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

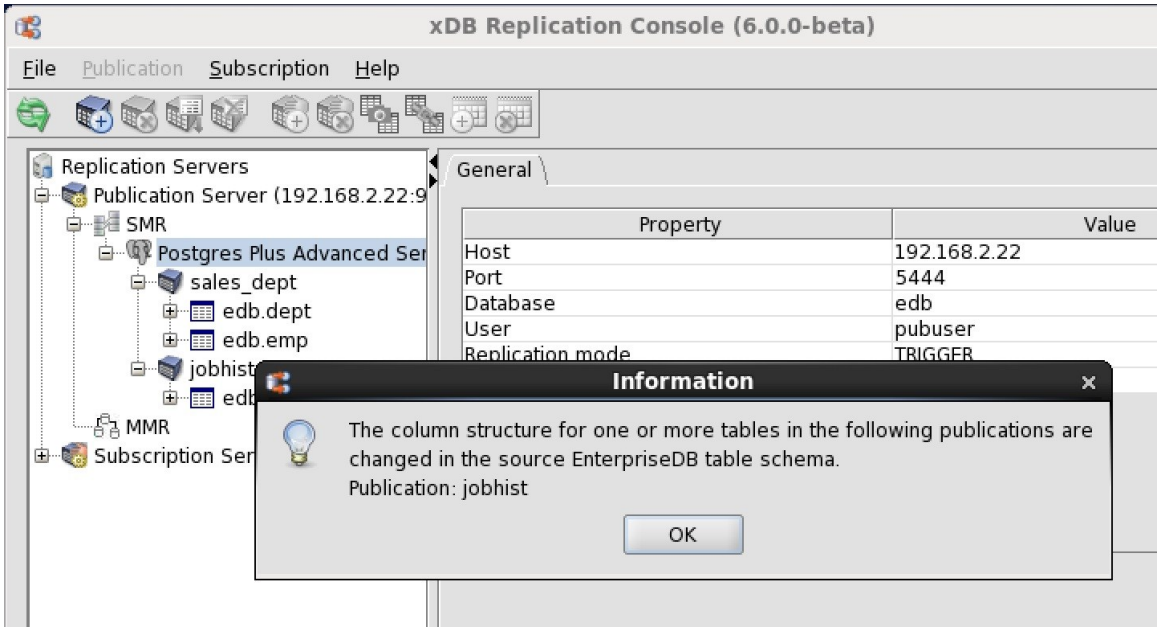


Figure 7-89 - List of publications with modified tables

7.6.6 Removing a Publication

In a single-master replication system, a publication can be removed before its associated subscriptions are removed. See Section 5.5.5 for directions to remove a subscription.

In a multi-master replication system, the publication is removed from under the Publication Database node representing the master definition node. Before a publication can be removed, all non-MDN nodes must be removed. See Section 7.6.7 for directions to remove a publication database definition of a master node.

Removing a publication does not delete the publication tables in the publication database. It removes the identity and association of these tables to xDB Replication Server so the tables remain in the database until the DBA deletes them with the `DROP TABLE` SQL statement.

The publication database user name is also left intact along with some of the xDB Replication Server metadata database objects. Shadow tables and triggers associated with the publication tables that were created by the publication server are deleted when the publication is removed. The remaining metadata database objects are deleted when the publication database definition is removed.

Step 1: Make sure the publication server whose node is the parent of the publication you wish to remove is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2 (For SMR only): Select the Publication node of the publication that you wish to remove.

Step 2 (For MMR only): Select the Publication node under the Publication Database node representing the master definition node.

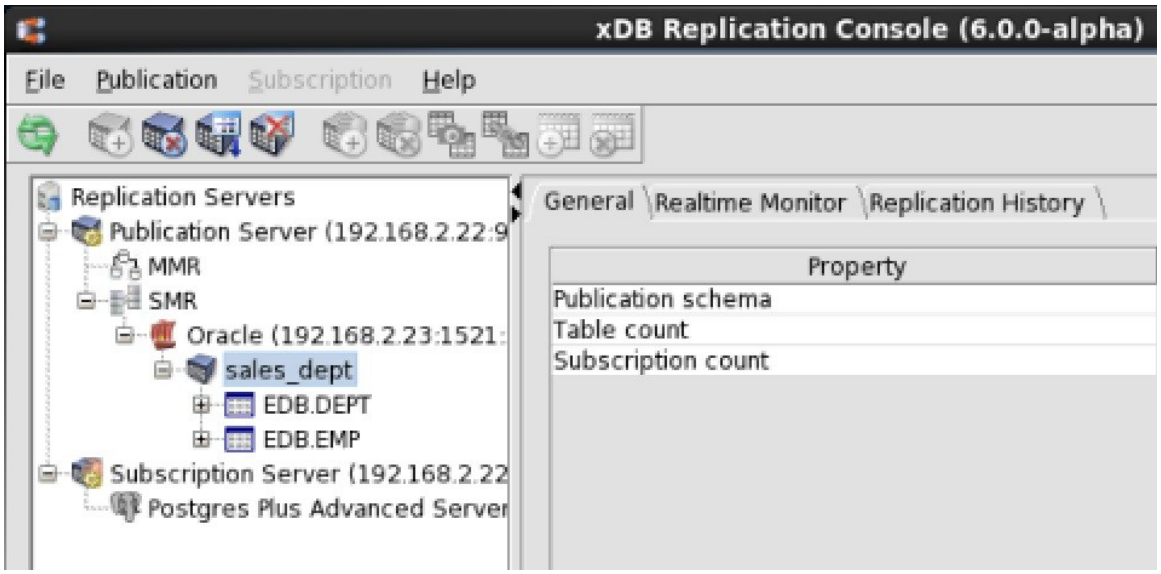


Figure 7-90 - Selecting a publication to remove

Step 3: Remove the publication in any of the following ways:

- Choose Remove Publication from the Publication menu.
- Click the secondary mouse button on the Publication node and choose Remove Publication.
- Click the primary mouse button on the Remove Publication icon.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

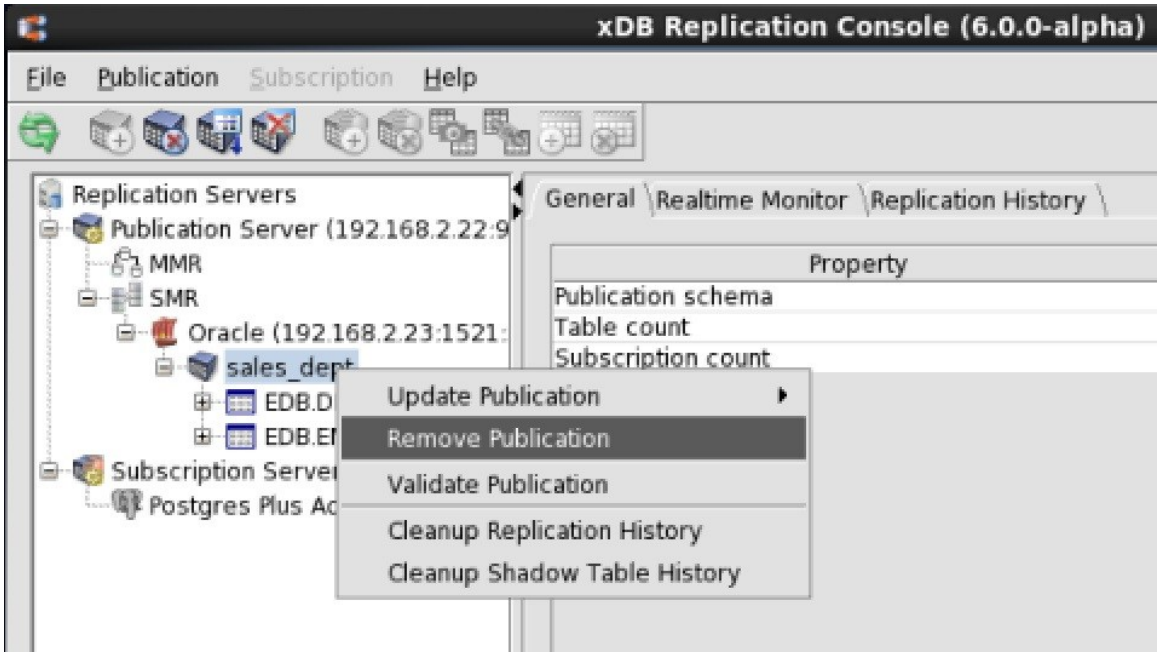


Figure 7-91 - Removing the publication using the menu bar

Step 4: In the Remove Publication confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

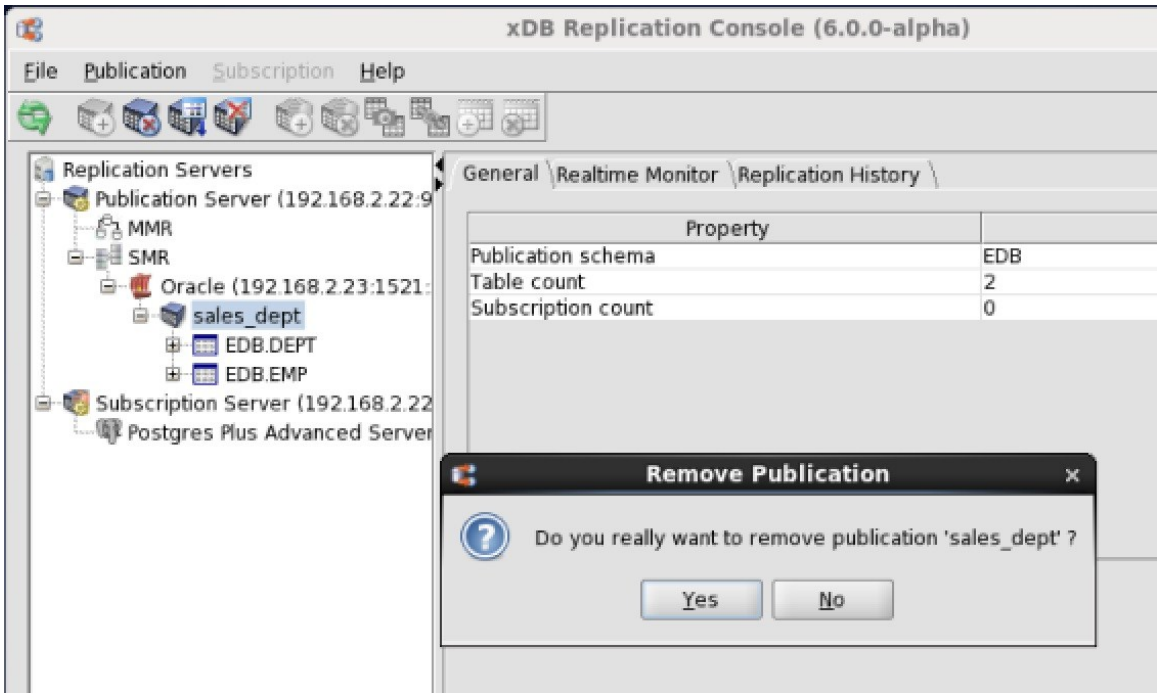


Figure 7-92 - Remove Publication confirmation

The Publication node no longer appears under the Publication Database node.

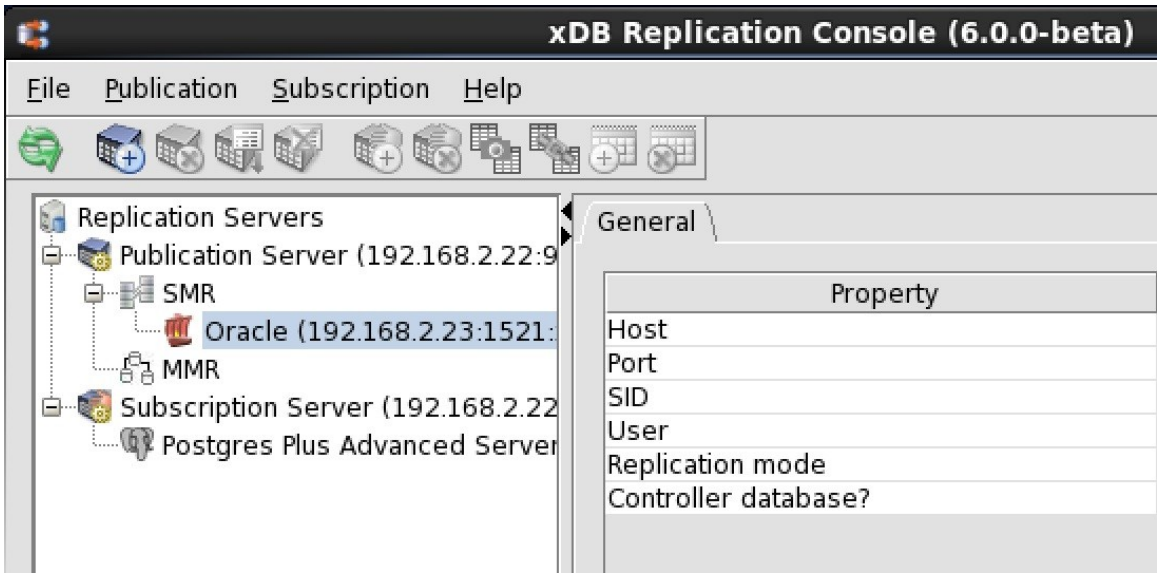


Figure 7-93 - Replication tree after removing a publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.6.7 Removing a Publication Database

Deleting a publication database definition from xDB Replication Server is equivalent to removing its Publication Database node.

If the Publication Database node to be removed is currently designated as the controller database, and there are additional publication databases in other single-master or multi-master replication systems, then you must first switch the controller database role to another publication database. See Section 7.7 for information on switching the controller database.

If the Publication Database node to be removed is the only remaining publication database (that is, there are no other single-master or multi-master replication systems), then this database can remain as the controller database since there is no other publication database available to be designated as the controller database. However, any existing subscription database definition must be removed before removing the last Publication Database node.

In a single-master replication system, before a Publication Database node can be removed, all publications under that Publication Database node must be removed. See Section 7.6.6 for directions on removing a publication.

In a multi-master replication system, removing a Publication Database node representing a master node (other than the master definition node), eliminates that node's future participation in the replication system. Synchronization replications no longer involve tables in the removed master node.

In a multi-master replication system, removing the Publication Database node representing the master definition node removes the remaining metadata database objects of that particular multi-master replication system, effectively removing the multi-master replication system (except for the database objects comprising the publication tables).

Removing the Publication Database node representing the master definition node entails the following steps:

- If the multi-master replication system is the only xDB replication system (that is, there are no single-master replication systems), then switch the controller database to the master definition node if the designated controller database is not currently the same database as the master definition node.
- If there are one or more single-master replication systems in addition to the multi-master replication system, switch the controller database to a Postgres publication database of a single-master replication system. If none of the single-master publication databases is of type Postgres, and there are more than one Oracle or SQL Server publication databases, then you must create a Postgres publication database for a single-master replication system just for the purpose of designating it as the controller database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- All Publication Database nodes representing non-MDN nodes must be removed. Repeat the process described in this section for each such master node.
- The publication under the Publication Database node representing the master definition node must be removed. See Section 7.6.6 for directions on removing a publication.
- Remove the Publication Database node representing the master definition node using the process described in this section.

Removing a Publication Database node does not delete the physical database from the database server. It removes the identity and association of the database to xDB Replication Server so no further replications can originate from tables in the database unless there are other publication database definitions in xDB Replication Server with the same host and database identifier. The physical database can only be removed using the database management system's database removal procedures.

The publication database user name is also left intact.

All xDB Replication Server metadata database objects created for that publication database definition are deleted.

For Oracle and SQL Server: All metadata database objects under the publication database user's schema are deleted.

For Postgres only: The schema `_edb_replicator_pub` and all of its database objects are deleted from the publication database.

The following are the steps to remove the Publication Database node and equivalently, the publication database definition:

Step 1: Make sure the publication server whose node is the parent of the publication database definition you wish to remove is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication Database node that you wish to remove.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

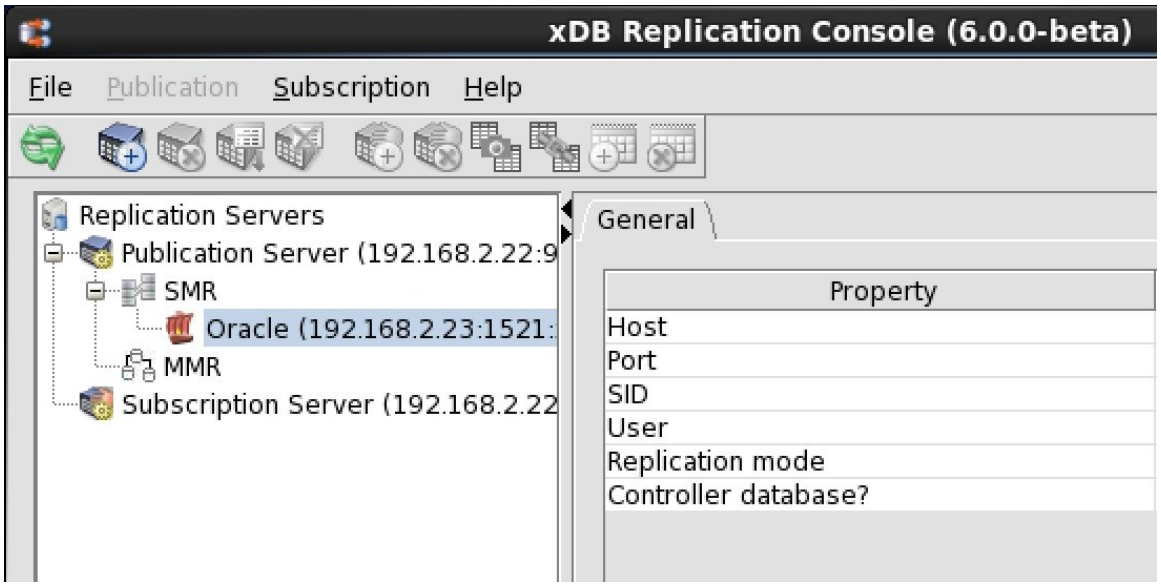


Figure 7-94 - Selecting a publication database definition for removal

Step 3: From the Publication menu, choose Publication Database, then Remove Database. Alternatively, click the secondary mouse button on the Publication Database node and choose Remove Database. The Remove Publication Database confirmation box appears.

Step 4: In the Remove Publication Database confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

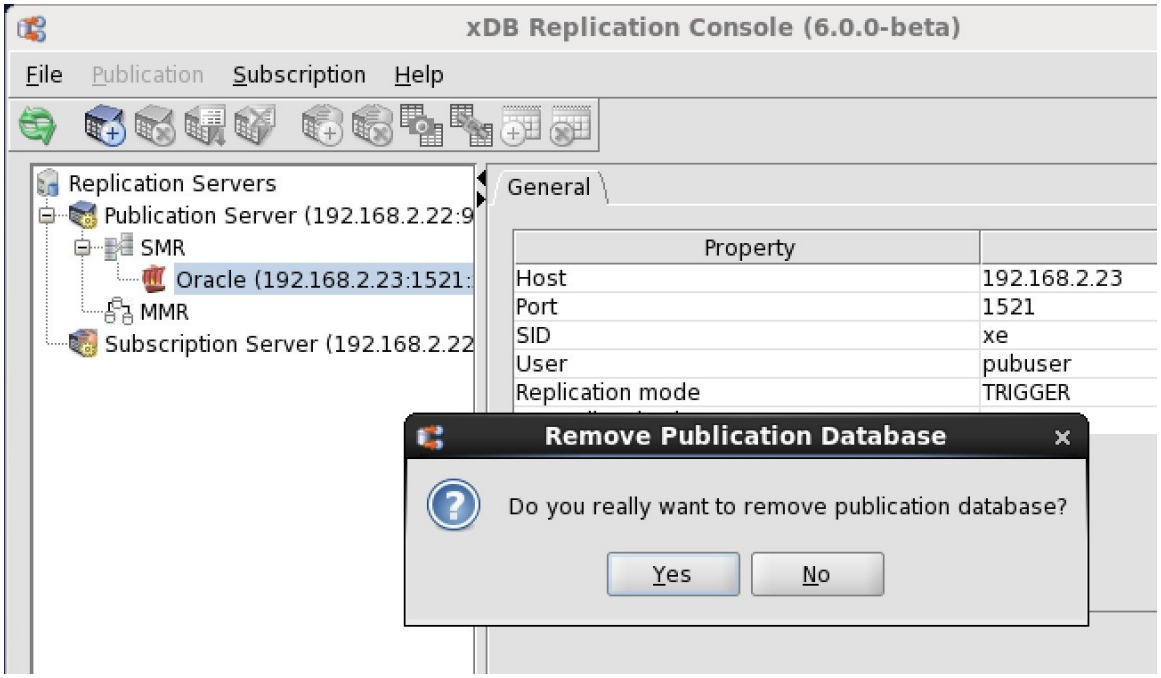


Figure 7-95 - Remove Publication Database confirmation

The Publication Database node no longer appears under the Publication Server node.

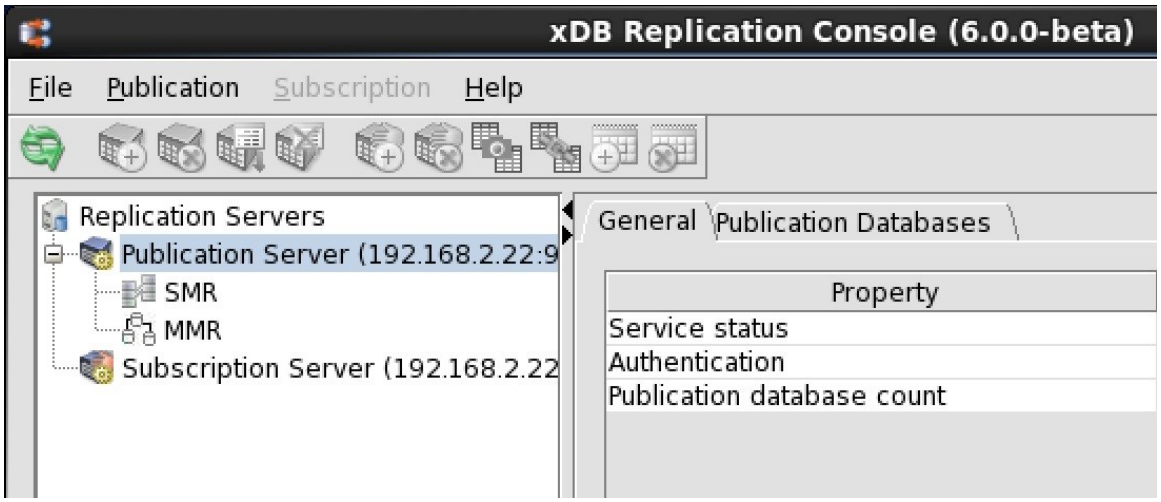


Figure 7-96 - Replication tree after removing a publication database

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.7 Switching the Controller Database

The controller database is designated in the xDB Replication Configuration file and determines the publication database to which the publication server and subscription server initially connect upon startup. See Section 2.3.1.12 for information on the controller database. See Section 2.3.1.3 for information on the xDB Replication Configuration file.

The current controller database cannot be removed from a replication system unless it is the last publication database remaining subordinate to the publication server. (That is, there are no other publication databases managed by the publication server.)

If there are more than one publication databases, and you wish to remove the publication database currently designated as the controller database, you must first set another publication database as the controller, and then you can remove the publication database previously designated as the controller.

The publication database used as the controller can be the master database of any single-master replication system or any master node of a multi-master replication system. Any database type (Oracle, SQL Server, or Postgres) is acceptable as the controller database.

Note: If the controller database is an Oracle or a SQL Server publication database, then a second Oracle or SQL Server publication database cannot be added to create a second single-master replication system. In order for xDB Replication Server to run more than one single-master replication systems consisting of Oracle or SQL Server publication databases, a Postgres publication database must be designated as the controller database.

Once you have multiple Oracle or SQL Server publication databases set up in single-master replication systems with a Postgres controller database, do not switch the controller database to an Oracle or SQL Server publication database.

Upon switching the controller database, the publication server updates the xDB Replication Configuration file so the parameters `user`, `password`, `host`, `port`, `database`, and `type` are set to the connection and authentication settings for the selected publication database.

The following are the steps to set another publication database as the controller database.

Step 1: Make sure the publication server whose node is the parent of the publication databases is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 2: Select the Publication Database node corresponding to the publication database that you wish to set as the controller database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

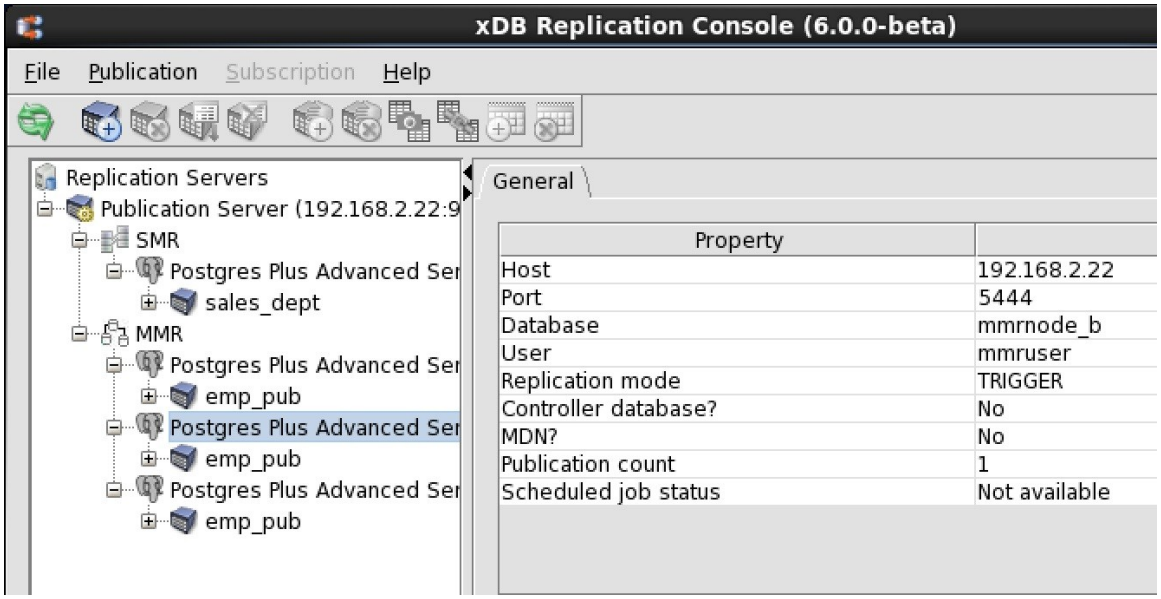


Figure 7-97 - Selecting the publication database to set as the controller database

Step 3: Click the secondary mouse button on the Publication Database node and choose Set as Controller database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

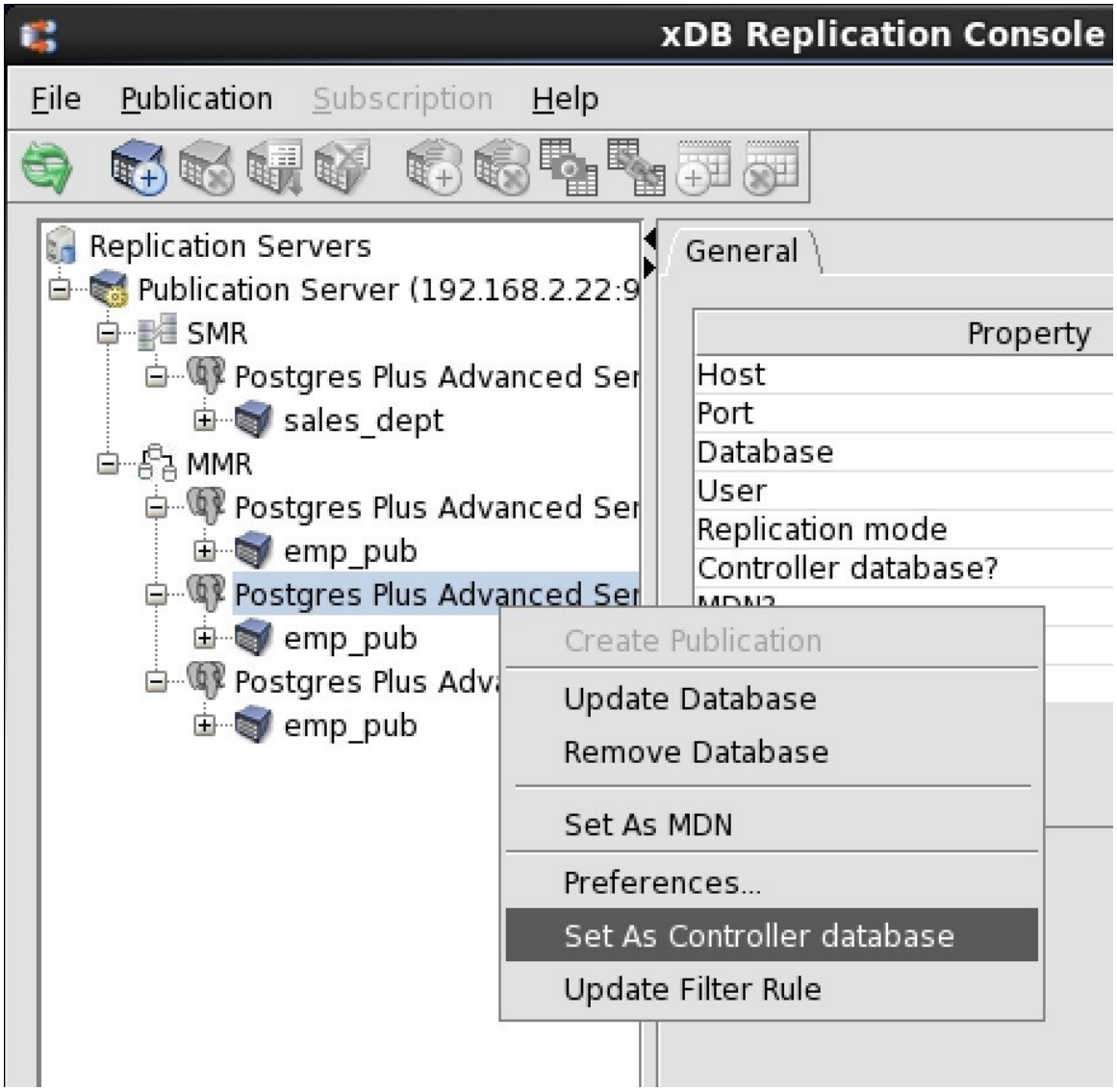


Figure 7-98 - Setting the controller database

Step 4: In the Set as Controller database confirmation box, click the Yes button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

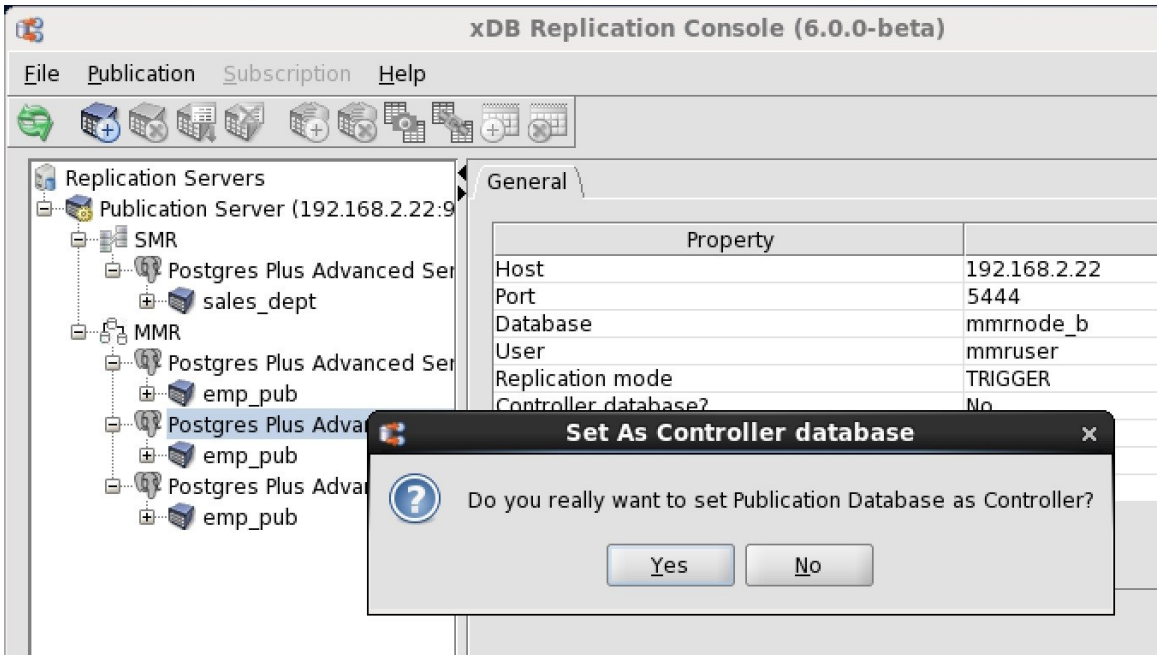


Figure 7-99 - Set as Controller database confirmation

Step 5: The selected publication database has now been set as the controller database.

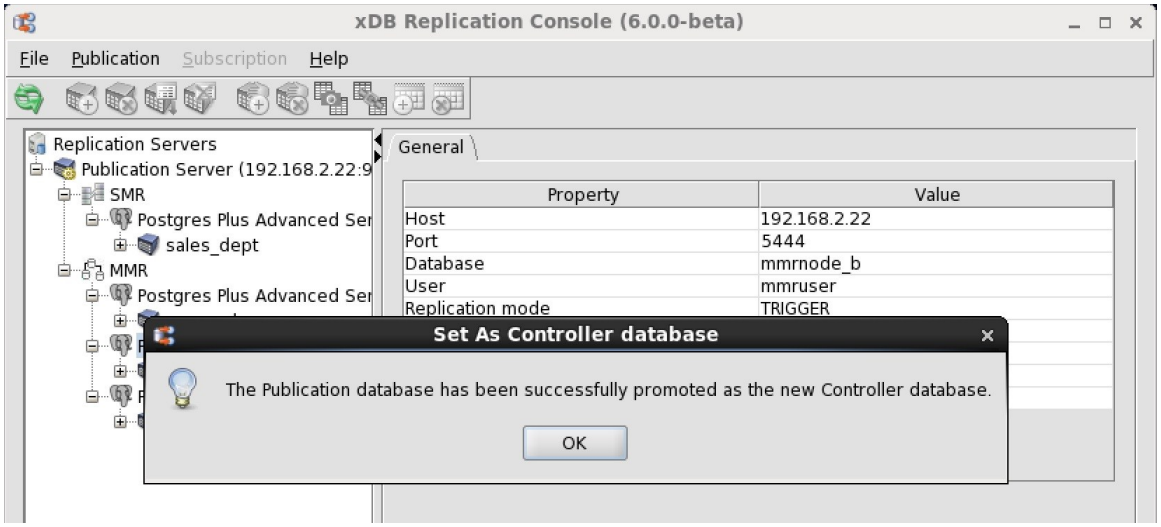


Figure 7-100 - Publication database promoted to controller database

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 6: The value `Yes` in the Controller database field of the Property window indicates this database is the controller database.

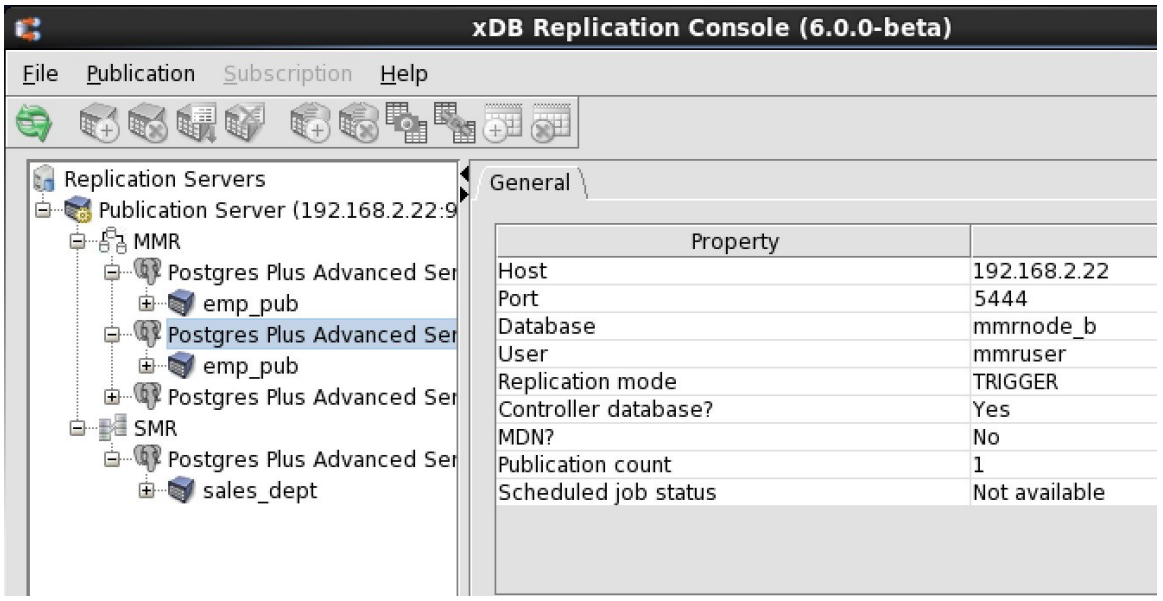


Figure 7-101 - Controller database indicated by 'Yes' in the Property window

The following shows the xDB Replication Configuration file after the controller database has been switched to the master node database `mmrnode_b`.

```
#xDB Replication Server Configuration Properties
#Thu Oct 15 14:42:35 GMT-05:00 2015
port=5444
admin_password=ygJ9AxoJEX854elcVIJPTw\=\=
user=mmruser
admin_user=admin
type=enterprisedb
database=mmrnode_b
password=ygJ9AxoJEX854elcVIJPTw\=\=
host=192.168.2.22
```

7.8 Replicating DDL Changes

Once a replication system has been created and is in operation, there may be occasions where it is necessary to make changes to the publication table definitions. These data definition language (DDL) changes may include the following:

- Adding new columns to a table

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Renaming existing columns
- Modifying a column data type
- Modifying a column constraint
- Removing columns

Note: See Section 7.6.5 for information on dealing with other types of table definition changes.

Table definition changes are generally implemented using the SQL `ALTER TABLE` statement, which is issued in an SQL command line utility program such as PSQL.

The *DDL change replication* feature accepts one or more `ALTER TABLE` statements. The statements may be provided by means of a text file or by entering them directly into the Alter Publication Table dialog box. The latter can be done by copying and pasting the statements into the dialog box, or by directly typing in the statements. The DDL change replication feature then performs the following actions:

- Applies the `ALTER TABLE` statements to the appropriate target table in the publication and subscription databases of a single-master replication system, or in all master nodes (including the master definition node) of a multi-master replication system.
- For the trigger-based method of synchronization replication, modifies the insert/update/delete triggers that add data into the shadow table whenever a transaction occurs on the target table.
- For the trigger-based method of synchronization replication, modifies the shadow table to properly accommodate the target table changes.

The DDL change replication feature is supported for Oracle and SQL Server subscription databases as well as Postgres subscription databases. However, the publication database must always be a Postgres database.

The syntax of the `ALTER TABLE` statement accepted by the DDL change replication features is as follows:

```
ALTER TABLE schema.table_name action
```

where *`action`* can be any of the following:

Rename an existing column:

```
RENAME [ COLUMN ] column_name TO  new_column_name
```

Add a column to the table:

```
ADD [ COLUMN ] column_name data_type
[ DEFAULT dflt_expr ]
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
[ column_constraint_1 [ column_constraint_2 ] ...]
```

Drop a column from the table:

```
DROP [ COLUMN ] column_name [ RESTRICT ]
```

Change the data type of a column:

```
ALTER [ COLUMN ] column_name [ SET DATA ] TYPE data_type
[ COLLATE "collation" ]
[ USING data_type_expr ]
```

Set the `DEFAULT` value of a column:

```
ALTER [ COLUMN ] column_name SET DEFAULT dflt_expr
```

Note: The `SET DEFAULT` clause is not supported when Oracle or SQL Server is the subscription database.

Drop the `DEFAULT` value of a column:

```
ALTER [ COLUMN ] column_name DROP DEFAULT
```

Note: The `DROP DEFAULT` clause is not supported when Oracle or SQL Server is the subscription database.

Set the column to reject null values:

```
ALTER [ COLUMN ] column_name SET NOT NULL
```

Note: The `SET NOT NULL` clause is not supported when SQL Server is the subscription database.

Allow the column to accept null values:

```
ALTER [ COLUMN ] column_name DROP NOT NULL
```

Note: The `DROP NOT NULL` clause is not supported when SQL Server is the subscription database.

The following restrictions apply to the manner in which the `ALTER TABLE` statements are specified whether it is in a text file or entered directly into the dialog box:

- Each `ALTER TABLE` statement must be terminated by a semicolon and begin on a separate line.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Although the Postgres `ALTER TABLE` statement allows multiple actions per statement, the xDB DDL change replication feature permits only one *`action`* per `ALTER TABLE` statement.
- The target table of all `ALTER TABLE` statements must be the same.
- The `DROP COLUMN` action cannot be specified for a column that comprises part of the table's primary key.

Parameters

`schema`

The name of the schema containing *`table_name`*. This value is case-sensitive.

`table_name`

The name of the table containing the column to be added, modified, or dropped. This value is case-sensitive.

`column_name`

The name of the column to be added, modified, or dropped.

`new_column_name`

The new name of the column specified in the `RENAME COLUMN` clause.

`data_type`

The data type of the column.

`dflt_expr`

An expression for the default value of the column.

`column_constraint_n`

A column constraint such as a `UNIQUE` or `CHECK` constraint. For additional information on column constraints see the `CREATE TABLE` SQL command in the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/sql-createtable.html

`RESTRICT`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In the DROP COLUMN clause, do not drop the column if there are objects dependent upon it. This is the default. **Note:** You cannot specify the CASCADE option as it is not supported by the DDL change replication feature.

collation

Collation assigned to the column. If omitted, the column data type's default collation is used. Examples of collation are default, C, POSIX, en_US, en_GB, or de_DE.

data_type_expr

An expression specifying how the column value with the new data type is to be converted from the column value with the old data type. This expression may reference other columns in the same table. If omitted, the default conversion is an assignment cast from the old data type to the new data type.

Examples

The following are examples of ALTER TABLE statements that can be used by the DDL change replication feature.

The following set of ALTER TABLE statements, either specified by a text file or entered directly into the dialog box, adds columns to the edb.emp table.

```
ALTER TABLE edb.emp ADD COLUMN gender CHAR(1) CHECK(gender IN ('M','F'));
ALTER TABLE edb.emp ADD COLUMN gradelevel VARCHAR2(4);
ALTER TABLE edb.emp ADD COLUMN title VARCHAR2(10);
```

The following ALTER TABLE statement changes the data type length of the title column and sets its values with the USING *data_type_expr* clause.

```
ALTER TABLE edb.emp
  ALTER COLUMN title SET DATA TYPE VARCHAR(25) USING
    CASE job
      WHEN 'CLERK' THEN 'ADMINISTRATIVE ASSISTANT'
      WHEN 'ANALYST' THEN 'R & D SPECIALIST'
      WHEN 'SALESMAN' THEN 'MARKETING REPRESENTATIVE'
      WHEN 'MANAGER' THEN 'SUPERVISOR'
      WHEN 'PRESIDENT' THEN 'CHIEF EXECUTIVE OFFICER'
    END;
```

The following query shows the values assigned to the title column after the DDL change replication feature applies the preceding ALTER TABLE statement to the edb.emp table. This change to the title column and assignment of values occurs in all the subscription databases of a single-master replication system or in all the master nodes of a multi-master replication system.

```
edb=# SELECT empno, ename, job, title FROM emp;
 empno | ename  |    job    |          title
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
-------+--------+----------+-------------------------
  7369 | SMITH  | CLERK    | ADMINISTRATIVE ASSISTANT
  7499 | ALLEN  | SALESMAN | MARKETING REPRESENTATIVE
  7521 | WARD   | SALESMAN | MARKETING REPRESENTATIVE
  7566 | JONES  | MANAGER  | SUPERVISOR
  7654 | MARTIN | SALESMAN | MARKETING REPRESENTATIVE
  7698 | BLAKE  | MANAGER  | SUPERVISOR
  7782 | CLARK  | MANAGER  | SUPERVISOR
  7788 | SCOTT  | ANALYST  | R & D SPECIALIST
  7839 | KING   | PRESIDENT | CHIEF EXECUTIVE OFFICER
  7844 | TURNER | SALESMAN | MARKETING REPRESENTATIVE
  7876 | ADAMS  | CLERK    | ADMINISTRATIVE ASSISTANT
  7900 | JAMES  | CLERK    | ADMINISTRATIVE ASSISTANT
  7902 | FORD   | ANALYST  | R & D SPECIALIST
  7934 | MILLER | CLERK    | ADMINISTRATIVE ASSISTANT
(14 rows)
```

The following set of `ALTER TABLE` statements drops the columns that were added in the first example.

```
ALTER TABLE edb.emp DROP COLUMN gender;
ALTER TABLE edb.emp DROP COLUMN gradelevel;
ALTER TABLE edb.emp DROP COLUMN title;
```

The DDL change replication feature can be invoked from either the xDB Replication Console (see Section 7.8.2) or the xDB Replication Server CLI (see Section 8.3.30).

The next section describes the process that occurs during DDL change replication.

7.8.1 DDL Change Replication Process

The DDL statement is executed in a controlled manner such that the target table is exclusively locked (by the default setting of configuration option `ddlChangeTableLock`) during the course of the operation. This is done to avoid loss of any transactions while the replication triggers and shadow table are modified by the DDL change replication process. Only one target table is locked at a time while DDL change replication takes place on that table, its triggers, and shadow table.

If there is a backlog of pending transactions, it is recommended to perform an explicit synchronization replication before performing DDL change replication to avoid prolonging the DDL change replication process for a longer period of time.

DDL change replication should be performed when the OLTP rate is very low (near zero).

Note: Exclusive acquisition of each target table during the DDL change replication process can be turned off by setting `ddlChangeTableLock` to `false`. However, this should be done only when there are no write transactions taking place against the target table, otherwise transactions may not be recorded by the replication system. See Section 10.4.1.11 for additional information on the `ddlChangeTableLock` configuration option.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following is the series of steps that occur during the DDL change replication process.

- The publication server performs a health check across all databases in the replication system to ensure they can be accessed. If any database is not available the DDL change replication process is aborted with a notification to the user.
- If the publication server configuration option `ddlChangeTableLock` is set to its default value of `true`, an exclusive table lock is requested on the table to which the DDL change is to be applied. If another application already has a lock on the table, there is a wait time of 2 minutes after which the DDL change replication process is aborted if the lock is not released before then. If `ddlChangeTableLock` is set to `false`, an exclusive table lock is not requested.
- The DDL statement is executed against the target table. The replication triggers and shadow table are modified accordingly. If an error occurs, the user is informed and the operation is aborted. If `ddlChangeTableLock` is set to `true`, the exclusive lock is released.
- The preceding two bullet points are repeated on the target table for each database in the replication system.
- The in-memory table metadata definition is refreshed to reflect the DDL change. The user is informed of successful completion of the operation.
- If an error occurs during the prior steps, any changes up to that point are rolled back so that the publication table, replication triggers, and shadow table are reverted back to their original state prior to the start of this operation. If one or more databases goes down before completion of the operation, the publication is marked as dirty to avoid further replication events.

The following section describes how to initiate DDL change replication using the xDB Replication Console.

7.8.2 DDL Change Replication Using the xDB Replication Console

DDL change replication can be applied using the xDB Replication Console as follows.

Step 1: If you plan to use a file to supply the `ALTER TABLE` statements to a publication table, prepare the text file. Make sure this text file is accessible by the operating system account with which you will open the xDB Replication Console.

Alternatively, you can copy and paste, or directly type in the `ALTER TABLE` statements into the Alter Publication Table dialog box without having to save the statements in a file.

Step 2: Make sure the publication server whose node is the parent of the publication containing the table you wish to change is running and has been registered in the xDB Replication Console you are using. See Section 5.2.1 for directions on starting and registering a publication server.

Step 3: Under the publication database of a single-master replication system, or under the master definition node of a multi-master replication system, open the Alter Publication

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Table dialog box by clicking the secondary mouse button on the Table node of the table to be modified and choose Alter Table.

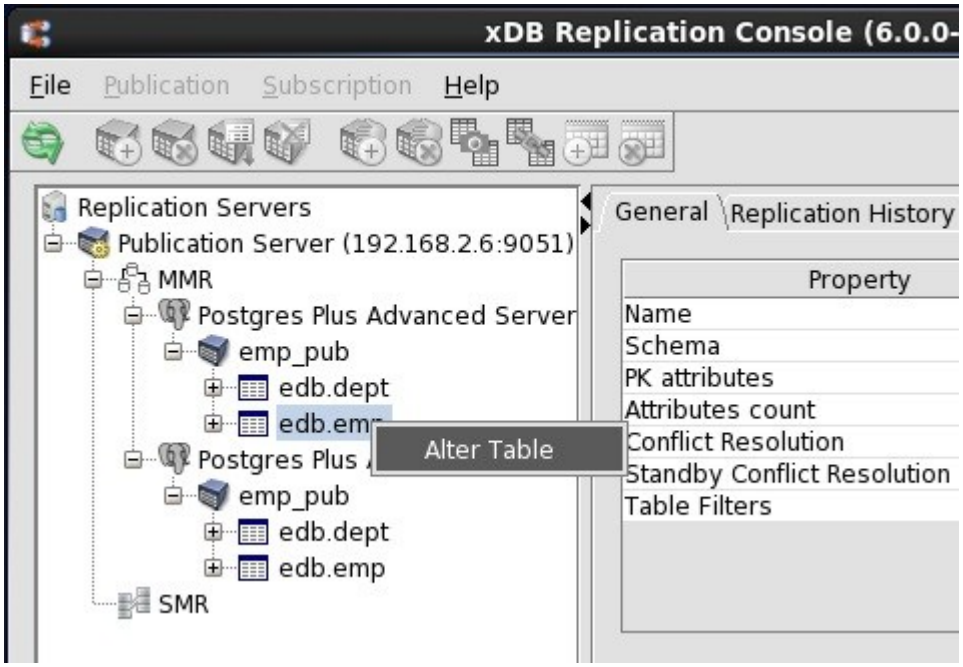


Figure 7-102 - Selecting a table to alter

Step 4: In the Alter Publication Table dialog box, if you saved the `ALTER TABLE` statements in a text file, make sure the DDL Script File option is selected, browse for this file, and click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

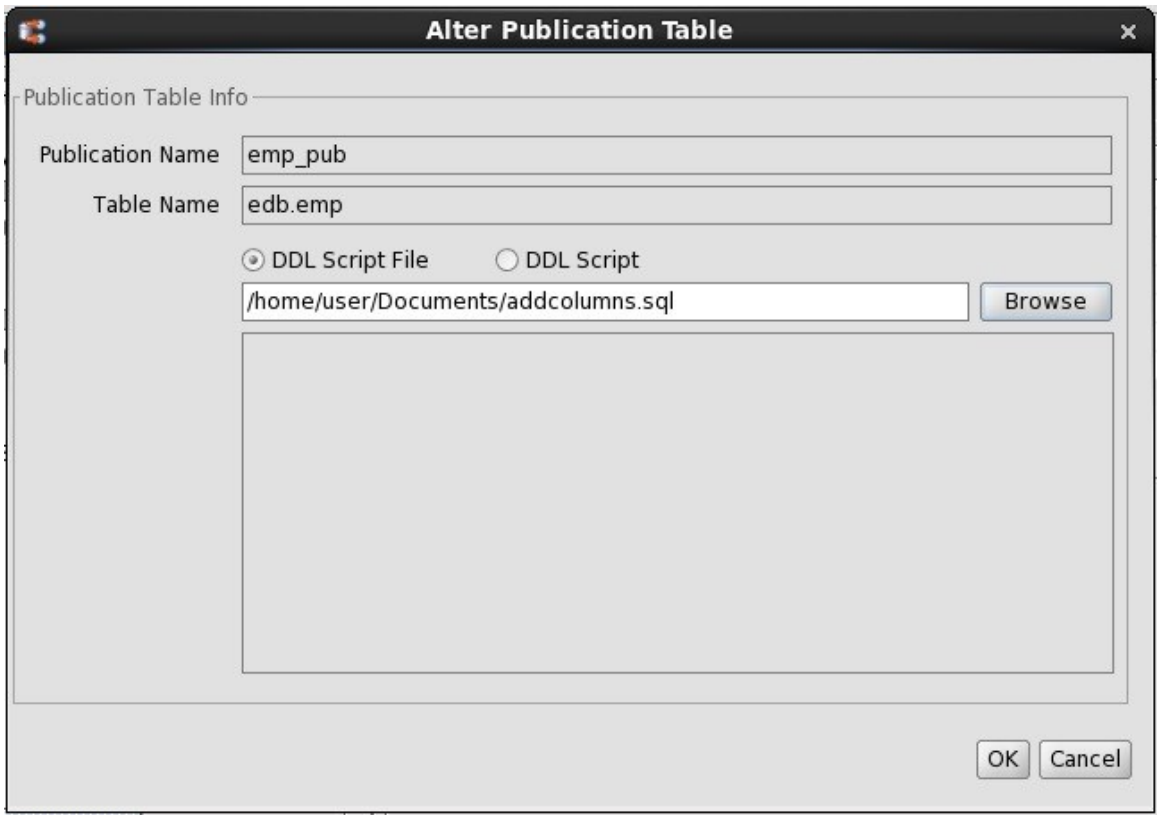


Figure 7-103 - Alter Publication Table dialog box with DDL script file

Alternatively, if you are directly entering the `ALTER TABLE` statements, select the DDL Script option instead of the DDL Script File option. Directly type in, or copy and paste the `ALTER TABLE` statements from your source into the text box. Click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

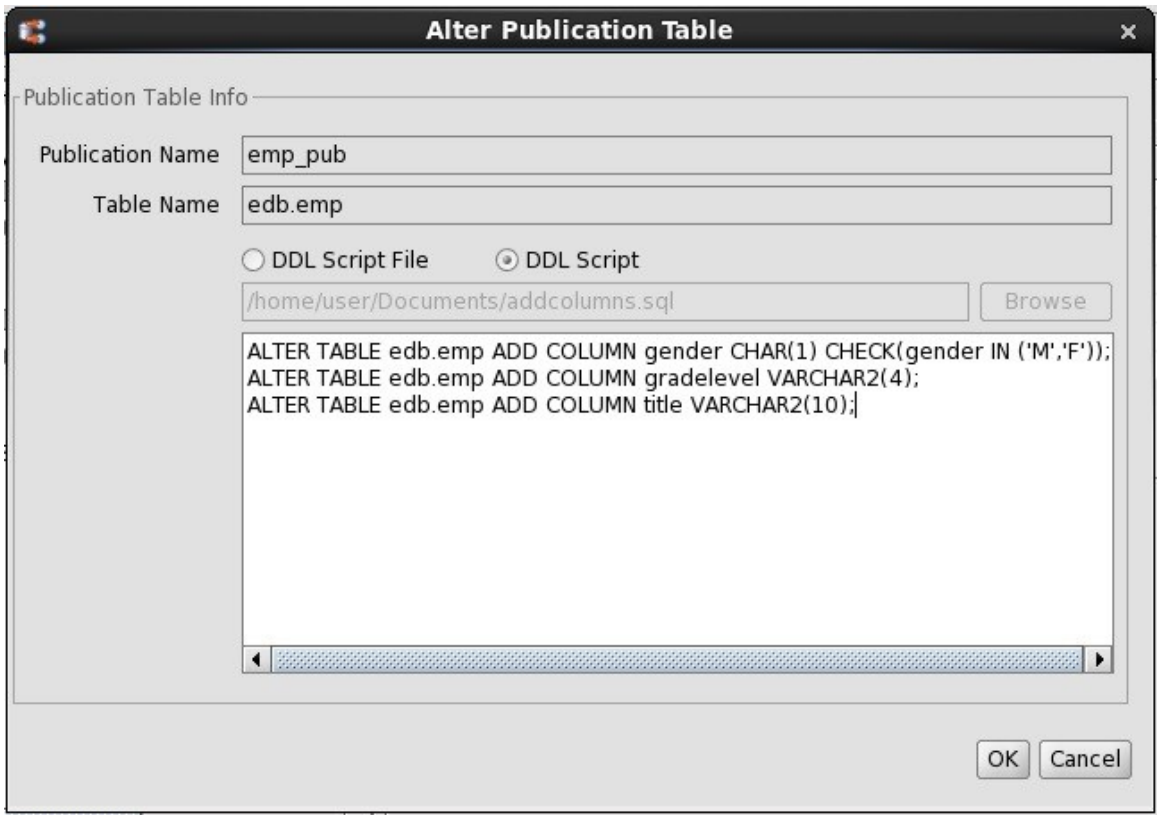


Figure 7-104 - Alter Publication Table dialog box with copy and paste

Step 5: If the DDL replicated successfully message box appears, the DDL change was successful across all databases. Click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

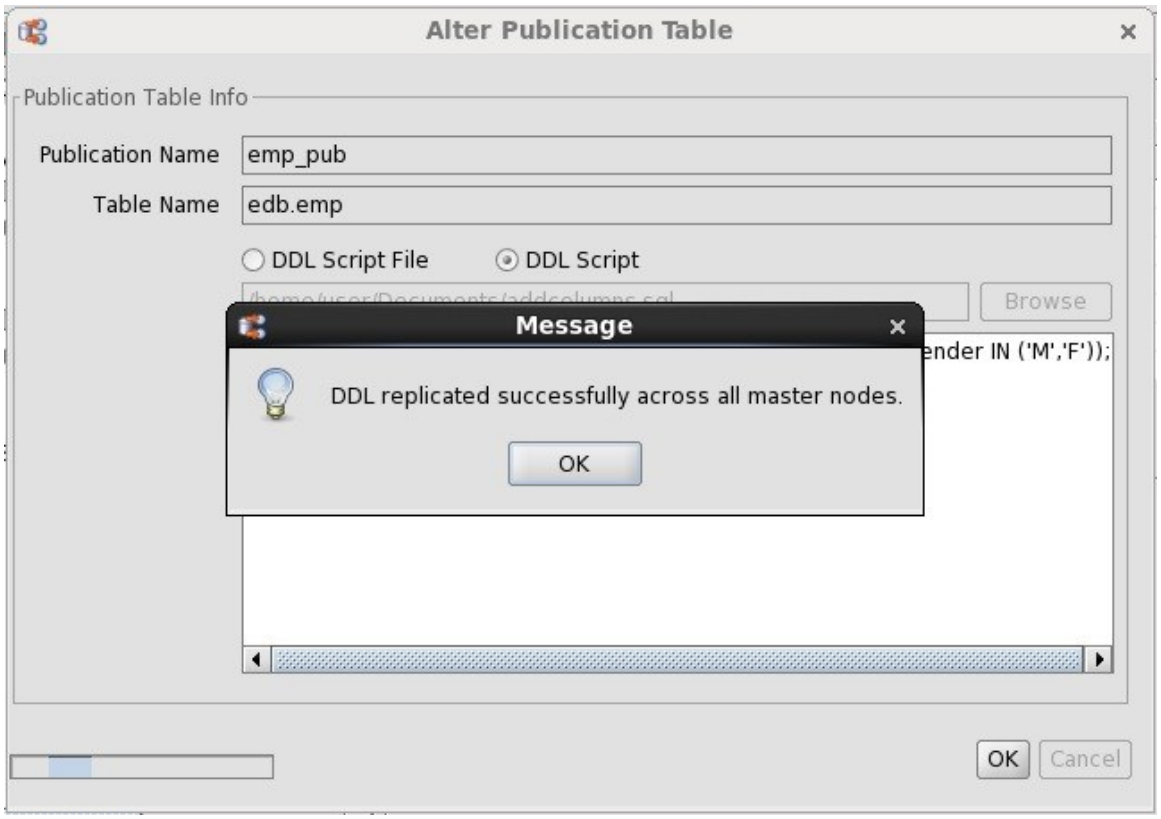


Figure 7-105 - DDL replicated successfully

If DDL replication was not successful, the problem must be investigated and resolved on a case by case basis. Factors to look for include the following:

- Were the modifications in the `ALTER TABLE` statements successfully applied to the target table in each database of the replication system?
- For the trigger-based method, were the replication triggers on the target table modified to account for the `ALTER TABLE` statements?
- For the trigger-based method, was the shadow table `RRST_`*`schema_table`* located in the `_edb_replicator_pub` schema in each database of the replication system modified to account for the `ALTER TABLE` statements?

If it is apparent that the replication system is not in a consistent state regarding the table definitions, see the beginning of Section 7.6.5 for guidance on how to deal with such issues.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.9 Loading Tables From an External Data Source (Offline Snapshot)

There may be circumstances when you want to initially load your target tables (subscription tables of a single-master replication system, or non-MDN nodes of a multi-master replication system) using a method other than the snapshot replication functionality of xDB Replication Server. This is referred to as using an *offline snapshot*.

For example, you might initially load the tables by running the Migration Toolkit from the command line or by using a backup from an external data source.

When you load the target tables using an offline snapshot, special preparations must be taken to account for the following deviations from the default target table creation and loading process:

- In the typical, default scenario xDB Replication Server creates the target table definitions when you define the subscription in a single-master replication system, or add an additional master node in a multi-master replication system. When using an offline snapshot, creation of the target table definitions is expected to be your responsibility. You must therefore prevent xDB Replication Server from creating the target table definitions.
- In the typical, default scenario xDB Replication Server performs synchronization replication using batches of SQL statements. If any statement in a batch results in an error, all statements in the batch are rolled back. When using an offline snapshot, if there is the possibility that the external data source used to load the target tables already has transactions applied to it that are also recorded in the shadow tables of the source tables, then you must perform the first synchronization replication in non-batch mode. This is because the batch of synchronization transactions may include SQL statements that have already been applied to the target, which may result in a statement failure in certain cases.

This section discusses how to deal with the preceding two points for both a single-master and a multi-master replication system.

First, non-batch mode synchronization and why it must be used in certain cases is explained in more detail in the following section.

7.9.1 Non-Batch Mode Synchronization

Synchronization replications are done in batches of updates, each batch committed in a separate transaction. Therefore if any single update in a batch fails, all the updates in the batch are rolled back.

This process has the following implications.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Prior to and during the time when the offline snapshot is in progress, there may be updates to the source tables, which are recorded in the source tables' shadow tables. After the offline snapshot completes, there may be additional updates to the source tables that are also recorded in the shadow tables.

Since xDB Replication Server has no knowledge of the external data source used to load the target tables, it is unknown to xDB Replication Server whether or not any of the updates made to the source tables during or after the offline snapshot, have already been included in the data used to load the target tables.

As a result, the shadow tables may include a mixture of duplicate updates that have already been applied to the target tables, as well as new updates that have not been applied to the target tables.

If you then perform synchronization replication, the publication server attempts to apply all updates recorded in the shadow tables in batches.

If one of the updates had been an insertion of a new row, and this new row is already in the target table loaded from the offline snapshot, a duplicate key error results when the publication server attempts to apply the batch containing the `INSERT` statement for this row. The duplicate key error forces the rollback of the entire batch. This causes the exclusion of updates in the batch that may not yet have been carried over to the target tables. The source tables and target tables are now inconsistent since there were updates to the source tables that have not been applied to the target tables.

Note: The effects of applying `UPDATE` and `DELETE` statements in the batch to a target table that already has been changed by these updates does not cause the same problem as repeated application of `INSERT` statements. The `UPDATE` statement would just change the row to the same values a second time. When a `DELETE` statement affects no rows, this is not considered an error by the database server, and therefore, no rollback of the batch occurs.

The solution to the potential rollback of a batch is to apply the shadow table updates in non-batch mode. That is, each SQL statement is individually committed. In that way, if an insertion of a row fails due to a duplicate key error, that statement alone is rolled back. The error does not affect the other shadow table updates that must be applied since all updates are enclosed within their own, individual transactions.

The `batchInitialSync` configuration option controls whether the first synchronization replication occurs in batch or non-batch mode.

If you are using an offline snapshot in an active replication system where updates are occurring to the source tables and transactions are thus accumulating in the shadow tables for the trigger-based method, it is advisable to set `batchInitialSync` to `false` to perform the first synchronization replication in non-batch mode.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: An offline snapshot cannot be used to add a subscription or a master node to an active replication system that uses the log-based method. For the log-based method, offline snapshots can only be used to initially configure the system, and not to update it with additional nodes after the publication database or master node is actively receiving transactions.

If you are using offline snapshots to initially create the entire replication system that has yet to be activated, and the content of the offline snapshots are all assumed to be consistent for the source and target tables, then `batchInitialSync` can be left with its default setting of `true` since it is assumed that the first synchronization replication will not apply any duplicate updates.

7.9.2 Offline Snapshot Configuration Options

The following are the configuration options that you need to modify when using an offline snapshot.

Note: These options apply to the publication server only.

offlineSnapshot

The `offlineSnapshot` option must be set to `true` before creating the subscription for a single-master replication system, or before adding the master node for a multi-master replication system.

```
offlineSnapshot={true | false}
```

The default value is `false`.

When set to `true`, the `offlineSnapshot` option prevents the usual creation of the subscription schema and table definitions when the subscription is defined in a single-master replication system since it is assumed that you are creating the subscription table definitions and loading them from an external source other than the publication.

When adding the master node in a multi-master replication system, leave the Replicate Publication Schema and Perform Initial Snapshot boxes unchecked (see Section 6.3).

When `offlineSnapshot` is set to `true`, this has the direct effect within the control schema by setting column `has_initial_snapshot` to a value of `0` indicating an offline snapshot is used for the target subscription or master node represented by the row. Column `has_initial_snapshot` is set in table `xdb_publication_subscriptions` for a single-master replication system and in table `xdb_mmr_pub_group` for a multi-master replication system.

The setting of `has_initial_snapshot` influences the behavior of the `batchInitialSync` option as explained in the following section.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

After the first replication completes to the target subscription or master node, `has_initial_snapshot` is changed to `Y` by xDB Replication Server.

batchInitialSync

The `batchInitialSync` option is used to control whether the first synchronization after loading the target tables from an offline snapshot is done in batch mode (the default) or non-batch mode.

Set the `batchInitialSync` option to `false` to perform synchronization replication in non-batch mode.

The `offlineSnapshot` configuration option must have first been set to `true` prior to creating the subscription or adding the additional master node. A non-batch mode synchronization occurs only if `batchInitialSync` is `false` and the `has_initial_snapshot` column in the control schema is set to a value of `O` as described for the `offlineSnapshot` option.

```
batchInitialSync={true | false}
```

The default value is `true`.

7.9.3 Single-Master Replication Offline Snapshot

An offline snapshot can be used to initially load the subscription tables of a single-master replication system. For a publication that is intended to have multiple subscriptions, it is possible to create some of the subscriptions using the default xDB Replication Server snapshot replication process as described in Section 5.4.1, while other subscriptions can be created from an offline snapshot.

The following steps describe how to create a subscription from an offline snapshot.

Step 1: Register the publication server, add the publication database definition, and create the publication as described in Section 5.2.

Step 2: Register the subscription server and add the subscription database definition as described in sections 5.3.1 and 5.3.2, respectively.

Note: Steps 3 and 4 must be performed before creating the subscription. Steps 3 through 9 can be repeated each time you wish to create an additional subscription from an offline snapshot.

Step 3: Modify the publication server configuration file if these options are not already set as described by the following:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Change the `offlineSnapshot` option to `true`. When the publication server is restarted, `offlineSnapshot` set to `true` has the effect that: 1) creating a subscription does not create the schema and subscription table definitions in the subscription database as is done with the default setting, and 2) creating a subscription sets a column in the control schema indicating an offline snapshot is used to load this subscription.
- Set the `batchInitialSync` option to the appropriate setting for your particular situation as discussed at the end of Section 7.9.1.

Step 4: Restart the publication server if the publication server configuration file was modified in Step 3. See Section 5.2.1 for directions on restarting a publication server.

Step 5: In the subscription database, create the schema, the subscription table definitions, and load the subscription tables from your offline data source. The subscription database user name used in Section 5.3.2 must have full privileges over the database objects created in this step. Also review the beginning of Section 5.3.2 regarding the rules as to how xDB Replication Server creates the subscription definitions from the publication for each database type as you must follow these same conventions when you create the target definitions manually.

Step 6: Add the subscription as described in Section 5.3.3.

Step 7: Perform an on demand synchronization replication. See Section 5.4.2 for directions on performing an on demand synchronization replication.

Step 8: If you are not planning to load any other subscriptions using an offline snapshot at this time, change the `offlineSnapshot` option back to `false` and the `batchInitialSync` option to `true` in the publication server configuration file.

Step 9: Restart the publication server if you modified the publication server configuration file in Step 8.

7.9.4 Multi-Master Replication Offline Snapshot

An offline snapshot can be used to initially load the master nodes of a multi-master replication system. It is possible to load some of the master nodes using the xDB Replication Server snapshot replication functionality when defining the master node as described in Section 6.3 or by using an on demand snapshot as described in Section 6.5.1, while other master nodes can be loaded from an offline snapshot.

Note: Offline snapshots are not supported for a multi-master replication system that is actively in use. Any changes on an active master node will be lost during the offline snapshot process of dumping or restoring the data of another node.

The following steps describe how to create a master node from an offline snapshot.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 1: Register the publication server, add the master definition node, and create the publication as described in Section 6.2.

Note: The following steps must be performed before adding a master node that is to be loaded by an offline snapshot. Steps 2 through 10 can be repeated each time you wish to create an additional master node from an offline snapshot.

Step 2: Be sure there is no schedule defined on the replication system, otherwise remove the schedule for the duration of the following steps. See Section 7.3.2 for directions on removing a schedule.

Step 3: Modify the publication server configuration file if these options are not already set as described by the following:

- Change the `offlineSnapshot` option to `true`. When the publication server is restarted, `offlineSnapshot` set to `true` has the effect that adding a master node sets a column in the control schema indicating an offline snapshot is used to load this master node.
- Set the `batchInitialSync` option to the appropriate setting for your particular situation as discussed at the end of Section 7.9.1.

Step 4: Restart the publication server if the publication server configuration file was modified in Step 3. See Section 5.2.1 for directions on restarting a publication server.

Step 5: In the database to be used as the new master node, create the schema, the table definitions, and load the tables from your offline data source.

Step 6: Add the master node as described in Section 6.3 with options Replicate Publication Schema and Perform Initial Snapshot unchecked.

Step 7: Perform an initial on demand synchronization. See Section 6.5.2 for directions on performing an on demand synchronization.

Step 8: If you are not planning to load any other master nodes using an offline snapshot at this time, change the `offlineSnapshot` option back to `false` and the `batchInitialSync` option to `true` in the publication server configuration file.

Step 9: Restart the publication server if you modified the publication server configuration file in Step 8.

Step 10: Re-add the schedule if one had been removed in Step 2. See Section 7.2 for directions on creating a schedule.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

7.10 Replicating Postgres Partitioned Tables

Both PostgreSQL and Advanced Server support partitioned tables, which can be replicated with xDB Replication Server in either a single-master or multi-master replication system.

The following are the various partitioning techniques:

- Advanced Server partitioning compatible with Oracle databases
- Postgres declarative partitioning (applies to both PostgreSQL and Advanced Server version 10 and later)
- Postgres table inheritance (applies to both PostgreSQL and Advanced Server)

If you are using Advanced Server, partitioned tables can be created using the `CREATE TABLE` statement with partitioning syntax compatible with Oracle databases. For information on partitioning compatible with Oracle databases, see Chapter 10 "Table Partitioning" in the *EDB Postgres Advanced Server 10.0 Database Compatibility for Oracle Developers Guide* available from the EnterpriseDB website located at:

https://www.enterprisedb.com/resources/product-documentation

If you are using version 10 or later of PostgreSQL or Advanced Server, *declarative partitioning* can be used to create partitioned tables. The `CREATE TABLE` syntax for creating a declarative partitioned table is similar to the partitioning compatible with Oracle databases, but the individual partitions of the declarative partitioned table must be separately created with their own `CREATE TABLE` statements.

If you are using native PostgreSQL version 9.6 or earlier, you must use a technique called *table inheritance* where you first create a parent table from which you then create one or more child tables that inherit the columns of the parent. Each child is an independent table in its own right except that it includes the column definitions of its parent. You then define a trigger on the parent table to direct which child table an inserted row is to be stored. Table inheritance can be used on Advanced Server as well.

For information on declarative partitioning and table inheritance, see the PostgreSQL core documentation available at:

https://www.postgresql.org/docs/current/static/ddl-partitioning.html

Regardless of the partitioning method, the resulting partitioned table is comprised of a parent table with a set of child tables.

Replication of these Postgres partitioned tables in a single-master or multi-master replication system is accomplished in the same manner.

Note the following general restrictions when the publication contains a partitioned table:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- SQL Server cannot be used as a subscription database.
- When using table inheritance, the subscription databases must be Postgres – they cannot be Oracle or SQL Server.

All three partitioning techniques are illustrated on the `emp` table used as an example throughout this document. The partitioned table is then used in a publication of a multi-master replication system in the following sections:

- For creating a publication in Postgres 9.x, see Section 7.10.1.
- For creating a publication in Postgres 10 or later, see Section 7.10.2.

The following creates the partitioned table in Advanced Server using partitioning compatible with Oracle databases:

```
CREATE TABLE emp (
    empno           NUMERIC(4) PRIMARY KEY,
    ename           VARCHAR(10),
    job             VARCHAR(9),
    mgr             NUMERIC(4),
    hiredate        DATE,
    sal             NUMERIC(7,2),
    comm            NUMERIC(7,2),
    deptno          NUMERIC(2)
)
PARTITION BY LIST(deptno)
(
    PARTITION dept_10 VALUES (10),
    PARTITION dept_20 VALUES (20),
    PARTITION dept_30 VALUES (30)
);
--
--  Load the 'emp' table
--
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,NULL,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-
81',1600,300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'02-APR-
81',2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-
81',1250,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-
81',2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-
81',2450,NULL,10);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-
87',3000,NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-
81',5000,NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-81',1500,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-87',1100,NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'03-DEC-81',950,NULL,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'03-DEC-81',3000,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'23-JAN-82',1300,NULL,10);
```

The following creates the partitioned table in PostgreSQL or Advanced Server 10 or later using declarative partitioning:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: When creating a declarative partitioned table that is to be replicated using xDB Replication Server, the `PRIMARY KEY` constraint must be included in the `CREATE TABLE` statements of the individual partitions, not in the `CREATE TABLE` statement of the parent table to be partitioned.

```
CREATE TABLE emp (
    empno           NUMERIC(4),
    ename           VARCHAR(10),
    job             VARCHAR(9),
    mgr             NUMERIC(4),
    hiredate        DATE,
    sal             NUMERIC(7,2),
    comm            NUMERIC(7,2),
    deptno          NUMERIC(2)
)
PARTITION BY LIST(deptno);
--
--  Create the partitions
--
--  The partitions must contain the PRIMARY KEY constraint
--
CREATE TABLE emp_dept_10 PARTITION OF emp (empno PRIMARY KEY)
    FOR VALUES IN (10);
CREATE TABLE emp_dept_20 PARTITION OF emp (empno PRIMARY KEY)
    FOR VALUES IN (20);
CREATE TABLE emp_dept_30 PARTITION OF emp (empno PRIMARY KEY)
    FOR VALUES IN (30);
--
--  Load the 'emp' table
--
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,NULL,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-
81',1600,300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'02-APR-
81',2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-
81',1250,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-
81',2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-
81',2450,NULL,10);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-
87',3000,NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-
81',5000,NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-81',1500,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-87',1100,NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'03-DEC-81',950,NULL,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'03-DEC-81',3000,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'23-JAN-82',1300,NULL,10);
```

The following creates the partitioned table in PostgreSQL or Advanced Server using table inheritance:

```
--
--  Create the parent table
--
CREATE TABLE emp (
    empno           NUMERIC(4) NOT NULL CONSTRAINT emp_pk PRIMARY KEY,
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
    ename           VARCHAR(10),
    job             VARCHAR(9),
    mgr             NUMERIC(4),
    hiredate        DATE,
    sal             NUMERIC(7,2),
    comm            NUMERIC(7,2),
    deptno          NUMERIC(2)
);
--
--  Create the child tables
--
CREATE TABLE emp_dept_10 (
    CHECK (deptno = 10)
) INHERITS (emp);
CREATE TABLE emp_dept_20 (
    CHECK (deptno = 20)
) INHERITS (emp);
CREATE TABLE emp_dept_30 (
    CHECK (deptno = 30)
) INHERITS (emp);

ALTER TABLE emp_dept_10 ADD CONSTRAINT emp_dept_10_pk PRIMARY KEY (empno);
ALTER TABLE emp_dept_20 ADD CONSTRAINT emp_dept_20_pk PRIMARY KEY (empno);
ALTER TABLE emp_dept_30 ADD CONSTRAINT emp_dept_30_pk PRIMARY KEY (empno);
--
--  Create the trigger function to insert into the proper child by deptno
--
CREATE OR REPLACE FUNCTION emp_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.deptno = 10 THEN
        INSERT INTO emp_dept_10 VALUES (NEW.*);
    ELSIF NEW.deptno = 20 THEN
        INSERT INTO emp_dept_20 VALUES (NEW.*);
    ELSIF NEW.deptno = 30 THEN
        INSERT INTO emp_dept_30 VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Department # out of range.';
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;
--
--  Create the trigger
--
CREATE TRIGGER insert_emp_trigger
    BEFORE INSERT ON emp
    FOR EACH ROW EXECUTE PROCEDURE emp_insert_trigger();
--
--  Load the 'emp' table
--
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,NULL,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-
81',1600,300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'02-APR-
81',2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-
81',1250,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-
81',2850,NULL,30);
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-
81',2450,NULL,10);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-
87',3000,NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-
81',5000,NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-81',1500,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-87',1100,NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'03-DEC-81',950,NULL,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'03-DEC-81',3000,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'23-JAN-82',1300,NULL,10);
```

The following illustrates the types of SQL queries that can be made on the parent and child tables to show which tables actually contain the rows.

Querying the parent table, `emp`, with the asterisk appended to the table name in the `SELECT` statement, shows the rows in the parent and child tables. This is the default behavior if the asterisk is omitted.

```
edb=# SELECT * FROM emp*;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+-----------+------+---------------------+---------+---------+--------
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00 | 2450.00 |         |     10
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00 | 5000.00 |         |     10
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00 | 1300.00 |         |     10
  7369 | SMITH  | CLERK     | 7902 | 17-DEC-80 00:00:00 |  800.00 |         |     20
  7566 | JONES  | MANAGER   | 7839 | 02-APR-81 00:00:00 | 2975.00 |         |     20
  7788 | SCOTT  | ANALYST   | 7566 | 19-APR-87 00:00:00 | 3000.00 |         |     20
  7876 | ADAMS  | CLERK     | 7788 | 23-MAY-87 00:00:00 | 1100.00 |         |     20
  7902 | FORD   | ANALYST   | 7566 | 03-DEC-81 00:00:00 | 3000.00 |         |     20
  7499 | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 00:00:00 | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN  | 7698 | 22-FEB-81 00:00:00 | 1250.00 |  500.00 |     30
  7654 | MARTIN | SALESMAN  | 7698 | 28-SEP-81 00:00:00 | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER   | 7839 | 01-MAY-81 00:00:00 | 2850.00 |         |     30
  7844 | TURNER | SALESMAN  | 7698 | 08-SEP-81 00:00:00 | 1500.00 |    0.00 |     30
  7900 | JAMES  | CLERK     | 7698 | 03-DEC-81 00:00:00 |  950.00 |         |     30
(14 rows)
```

The following queries show how the rows are physically divided amongst the child tables. The use of the `ONLY` keyword results in rows only in the specified table of the `SELECT` statement, and not from any of its children.

```
edb=# SELECT * FROM ONLY emp;
 empno | ename | job | mgr | hiredate | sal | comm | deptno
-------+-------+-----+-----+----------+-----+------+--------
(0 rows)


edb=# SELECT * FROM ONLY emp_dept_10;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   | comm | deptno
-------+--------+-----------+------+---------------------+---------+------+--------
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00 | 2450.00 |      |     10
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00 | 5000.00 |      |     10
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00 | 1300.00 |      |     10
(3 rows)


edb=# SELECT * FROM ONLY emp_dept_20;
 empno | ename |   job   | mgr  |      hiredate       |   sal   | comm | deptno
-------+-------+---------+------+---------------------+---------+------+--------
  7369 | SMITH | CLERK   | 7902 | 17-DEC-80 00:00:00 |  800.00 |      |     20
  7566 | JONES | MANAGER | 7839 | 02-APR-81 00:00:00 | 2975.00 |      |     20
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  7788 | SCOTT | ANALYST | 7566 | 19-APR-87 00:00:00 | 3000.00 |      |     20
  7876 | ADAMS | CLERK   | 7788 | 23-MAY-87 00:00:00 | 1100.00 |      |     20
  7902 | FORD  | ANALYST | 7566 | 03-DEC-81 00:00:00 | 3000.00 |      |     20
(5 rows)


edb=# SELECT * FROM ONLY emp_dept_30;
 empno | ename  |   job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+----------+------+---------------------+---------+---------+--------
  7499 | ALLEN  | SALESMAN | 7698 | 20-FEB-81 00:00:00 | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN | 7698 | 22-FEB-81 00:00:00 | 1250.00 |  500.00 |     30
  7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 00:00:00 | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER  | 7839 | 01-MAY-81 00:00:00 | 2850.00 |         |     30
  7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 00:00:00 | 1500.00 |    0.00 |     30
  7900 | JAMES  | CLERK    | 7698 | 03-DEC-81 00:00:00 |  950.00 |         |     30
(6 rows)
```

The following section shows creation of the publication when using Postgres 9.6 or an earlier version. This also applies if the partitioned table is created with table inheritance.

Section 7.10.2 shows creation of the publication when using partitioning compatible with Oracle databases or declarative partitioning on a Postgres 10 or later database server.

7.10.1 Creating a Postgres 9.x Partitioned Table Publication

The following describes creating the publication on a Postgres database server of version 9.6 or earlier.

Follow the directions in Section 6.2 to create a master definition node along with a publication containing the partitioned table. (For a single-master replication system, create the publication database along with the publication according to the directions in Section 5.2.)

When creating the publication you must select the parent table along with the child tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

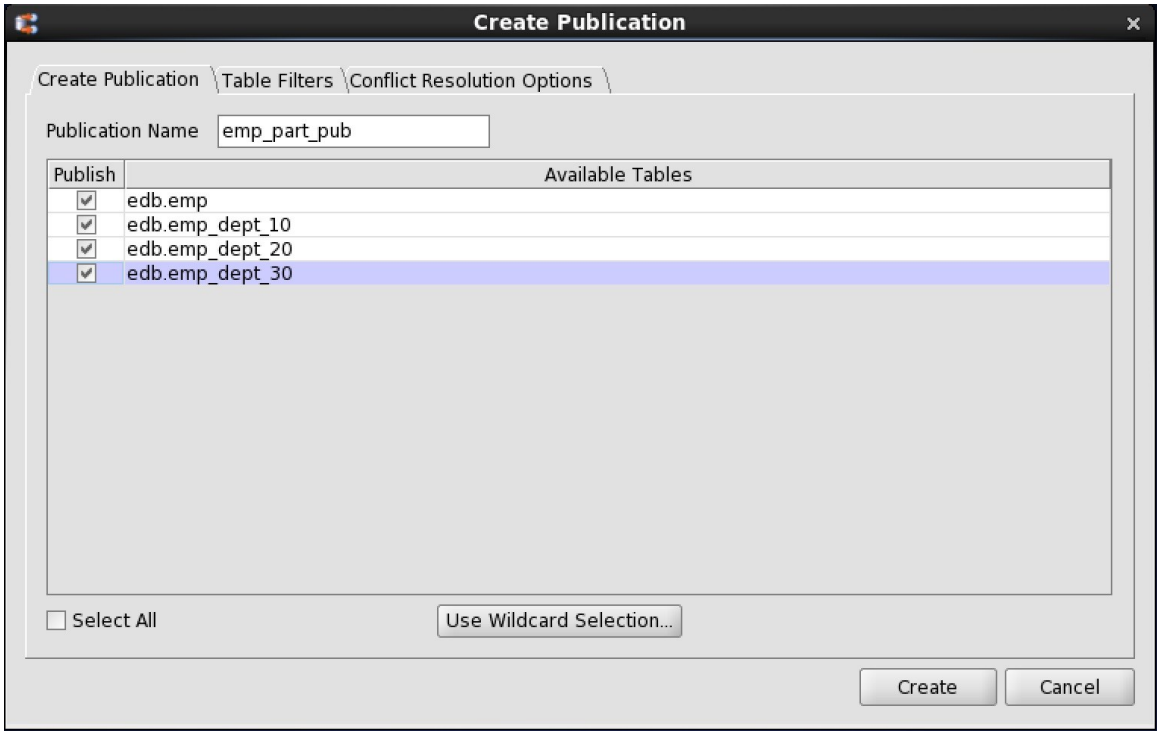


Figure 7-106 - Creating a publication for a Postgres 9.x partitioned table

The following shows the resulting replication tree for the partitioned table in the master definition node:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

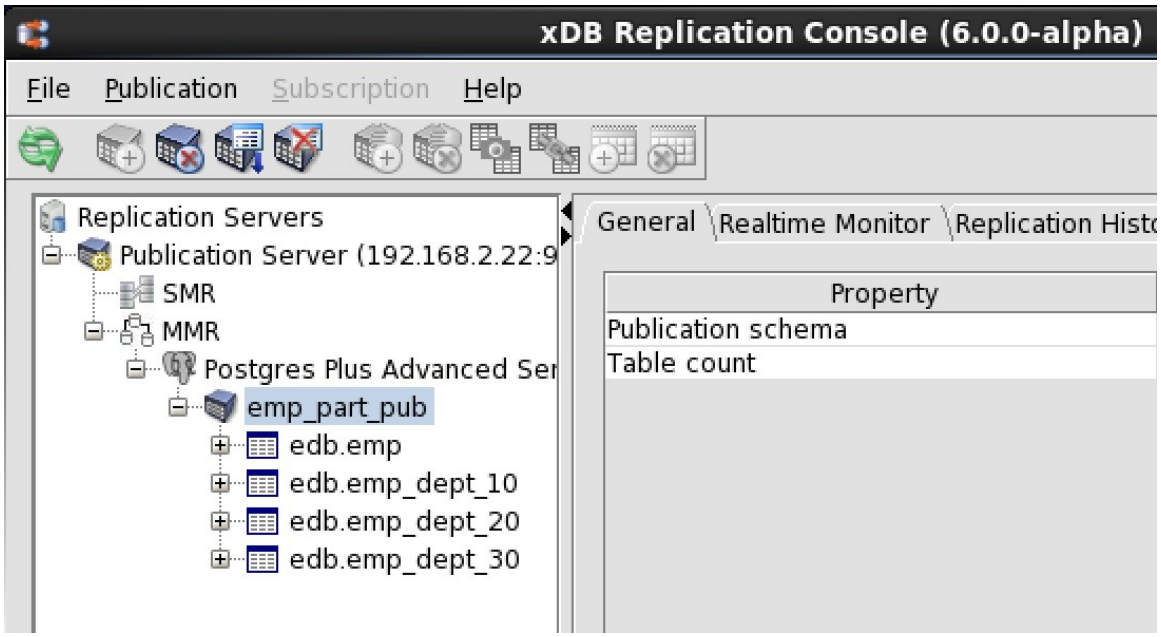


Figure 7-107 – Publication containing a Postgres 9.x partitioned table

Create additional master nodes as described in Section 6.3. (For a single-master replication system, create the subscription database and subscription according to the directions in Section 5.3.)

The following shows the resulting multi-master replication system after you have added an additional master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

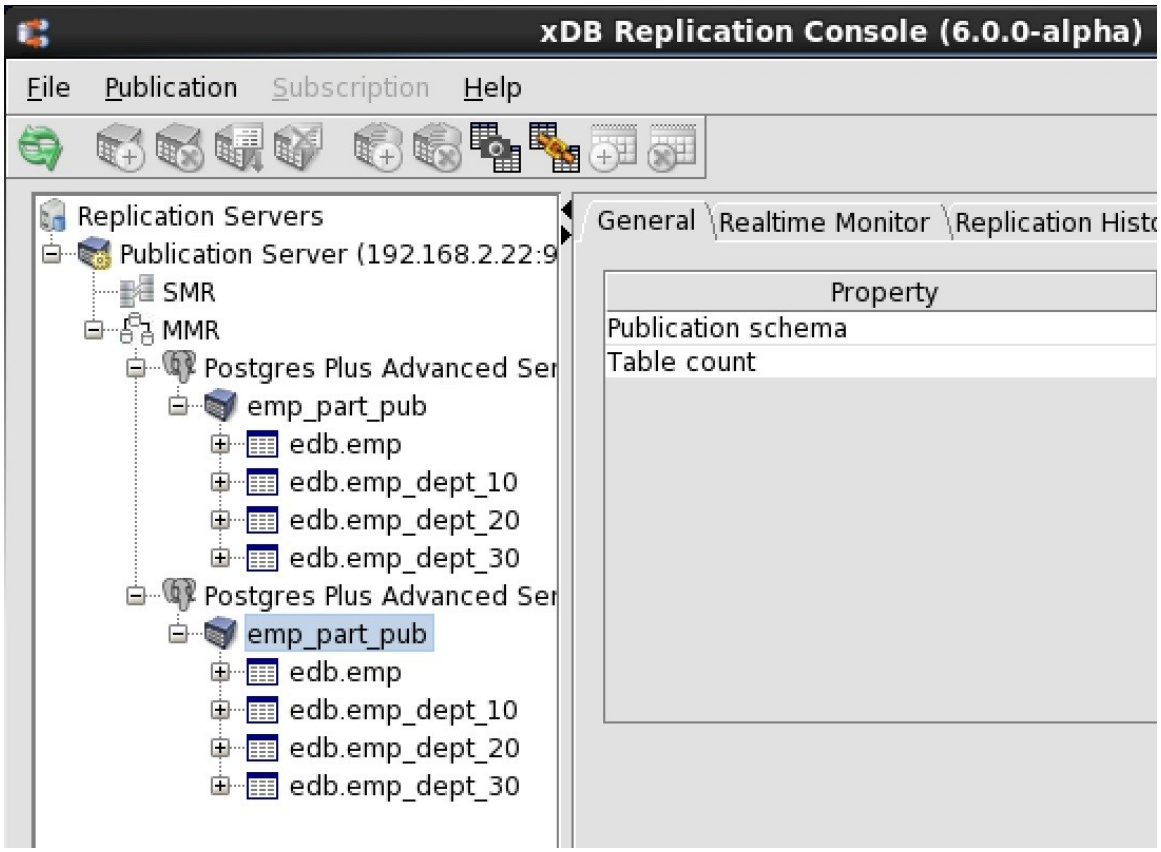


Figure 7-108 - MMR system with a Postgres 9.x partitioned table

The partitioned table can now be kept synchronized on the master nodes of the multi-master replication system.

7.10.2 Creating a Postgres Version 10 or Later Partitioned Table Publication

The following describes creating the publication using either partitioning compatible with Oracle databases or Postgres declarative partitioning on a Postgres 10 or later database server.

Note: If you are using table inheritance, you must still use the process described in Section 7.10.1 even when creating the publication on a Postgres 10 or later database server.

The following restrictions apply when the publication contains a table with partitioning compatible with Oracle databases or declarative partitioning:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- The log-based method of synchronization replication must be selected for the publication database. The trigger-based method cannot be used.
- In a single-master replication system, the subscription databases must be Postgres version 10 or later. Oracle and SQL Server cannot be used as a subscription database.
- In a multi-master replication system, all master nodes must be Postgres version 10 or later with the same compatibility mode as the master definition node (that is, either compatible with native PostgreSQL or compatible with Oracle databases). For more information on the multi-master replication system compatibility modes, see Section 10.1.3.3.

Follow the directions in Section 6.2 to create a master definition node along with a publication containing the partitioned table. (For a single-master replication system, create the publication database along with the publication according to the directions in Section 5.2.)

When creating the publication, only the parent table appears and is selected.

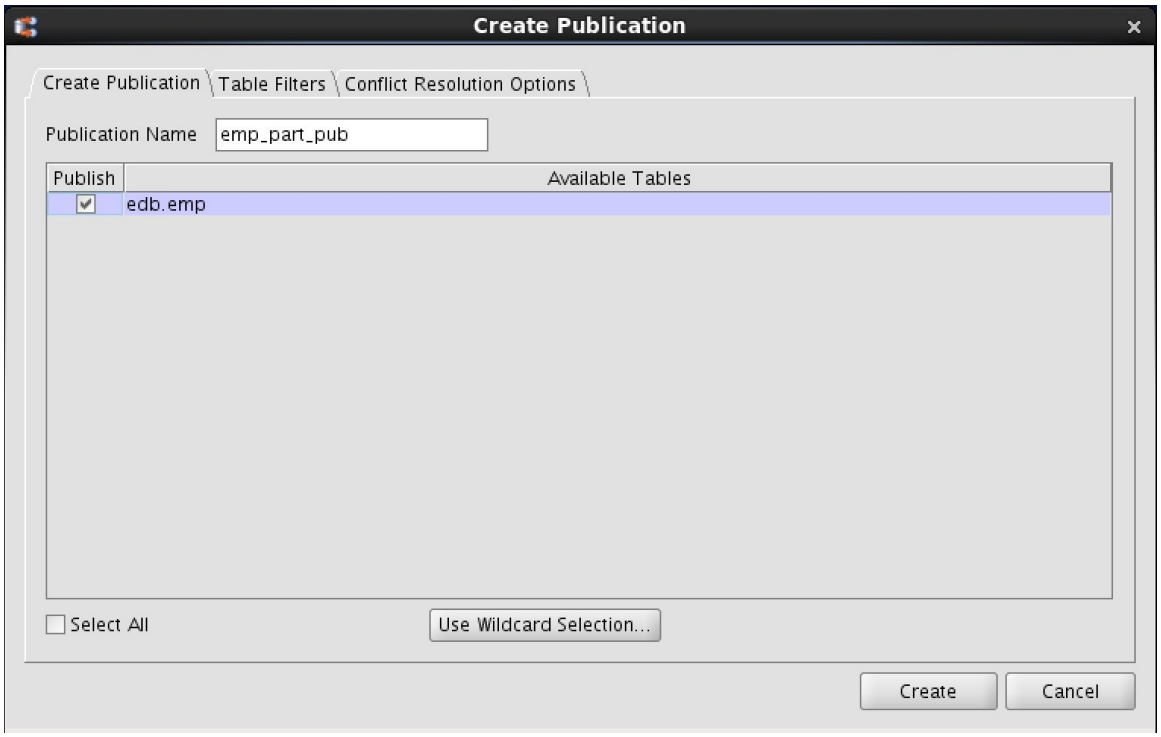


Figure 7-109 - Creating a publication for a Postgres 10 or later partitioned table

The following shows the resulting replication tree for the partitioned table in the master definition node:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

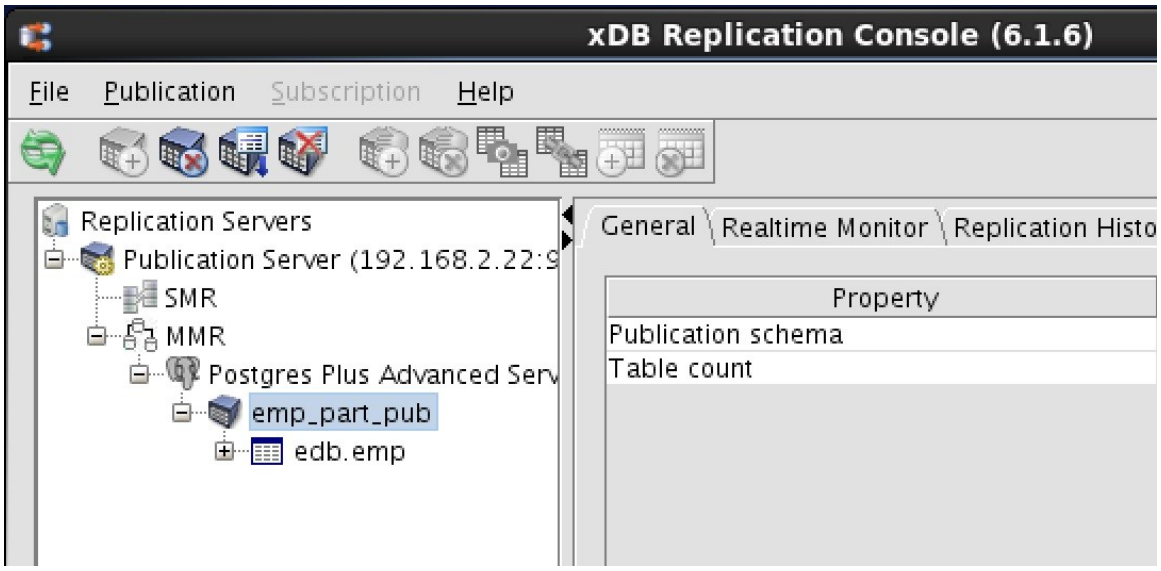


Figure 7-110 - Publication containing a Postgres 10 or later partitioned table

Create additional master nodes as described in Section 6.3. (For a single-master replication system, create the subscription database and subscription according to the directions in Section 5.3.)

The following shows the resulting multi-master replication system after you have added an additional master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

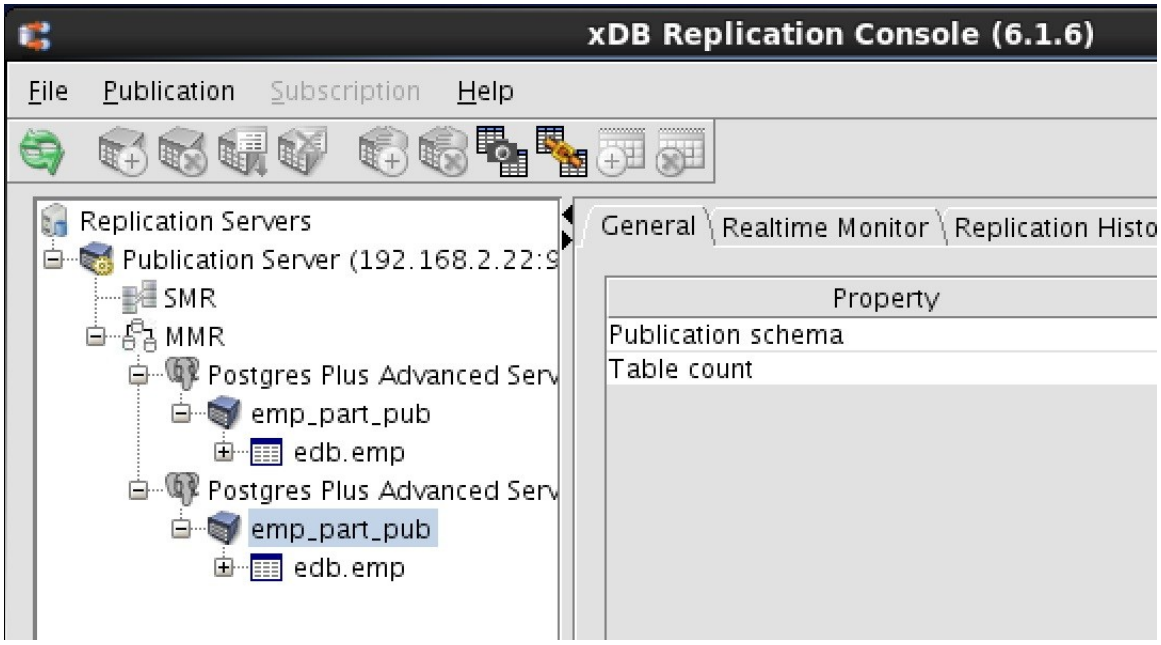


Figure 7-111 - MMR system with a Postgres 10 or later partitioned table

The partitioned table can now be kept synchronized on the master nodes of the multi-master replication system.

7.11 Using Secure Sockets Layer (SSL) Connections

Publication server and subscription server connections to Postgres publication databases, Postgres subscription databases, and Postgres master nodes can be accomplished using secure sockets layer (SSL) connectivity.

xDB Replication Server does not support SSL connections to Oracle and SQL Server databases used within any xDB replication system.

For a single-master replication system, the following connections can be made to Postgres databases enabled with SSL:

- Publication server connection to the publication database and to the subscription databases.
- Subscription server connection to the subscription databases.
- Migration Toolkit connection to the publication and subscription databases.

For a multi-master replication system, the following connections can be made to Postgres databases enabled with SSL:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Publication server connection to the master definition node and the non-MDN nodes.
- Migration Toolkit connection to the master definition node and the non-MDN nodes.

Note: SSL connections are not used from the xDB Replication Console or the xDB Replication Server Command Line Interface. The xDB user interfaces communicate with the publication server and subscription server, which in turn connect to the publication/subscription databases or master nodes.

Note: The Migration Toolkit connection using SSL occurs within the context of the publication server and subscription server SSL connections. Therefore, there are no separate steps that you need to perform for the Migration Toolkit SSL connection.

Using SSL requires various prerequisite configuration steps performed on the database servers involved with the SSL connections as well as on the Java truststore and keystore on the hosts running the publication server and subscription server.

The Java *truststore* is the file containing the Certificate Authority (CA) certificates with which the Java client (the publication server and subscription server) uses to verify the authenticity of the server to which it is initiating an SSL connection.

The Java *keystore* is the file containing private and public keys and their corresponding certificates. The keystore is required for client authentication to the server, which is used for xDB Replication Server SSL connections.

The following is material to which you can refer to for guidance in setting up the SSL connections:

- See the section on secure TCP connections with SSL in Chapter 17 "Server Setup and Operation" in the *PostgreSQL Core Documentation* located at:

  https://www.postgresql.org/docs/current/static/ssl-tcp.html

  for information on setting up SSL connectivity to Postgres database servers.
- For information on JDBC client connectivity using SSL see the section on configuring the client in Chapter 4 "Using SSL" in the The PostgreSQL JDBC Interface (https://jdbc.postgresql.org/documentation/94/ssl.html).

The following sections provide additional information for the configuration steps of using SSL with the xDB Replication Server.

- Configuring SSL on a Postgres database server (Section 7.11.1)
- Configuring SSL on a JDBC client for the publication and subscription servers (Section 7.11.2)

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Requesting SSL connection to the xDB Replication Server databases (Section 7.11.3)

7.11.1 Configuring SSL on a Postgres Database Server

This section provides an example of configuring SSL on a Postgres database server to demonstrate the use of SSL with xDB Replication Server. A self-signed certificate is used for this purpose.

Step 1: Create the certificate signing request (CSR).

In the following example the generated certificate signing request file is `server.csr`. The private key is generated as file `server.key`.

```
$ openssl req -new -text -nodes -subj
'/C=US/ST=Massachusetts/L=Bedford/O=EnterpriseDB/OU=XDB/emailAddress=support@
enterprisedb.com/CN=enterprisedb' -keyout server.key -out server.csr
Generating a 1024 bit RSA private key
..............................................++++++
.++++++
writing new private key to 'server.key'
-----
```

Note: When creating the certificate, the value specified for the common name field (designated as `CN=enterprisedb` in this example) must be the database user name that is specified in the User field of the Add Database or Update Database dialog box used when defining the publication database (see Section 5.2.2), subscription database (see Section 5.3.2), or master nodes (see sections 6.2.2 and 6.3).

Alternatively, user name maps can be used as defined in the `pg_ident.conf` file to permit more flexibility for the common name and database user name. Steps 8 and 9 describe the use of user name maps.

Step 2: Generate the self-signed certificate.

The following generates a self-signed certificate to file `server.crt` using the certificate signing request file, `server.csr`, and the private key, `server.key`, as input.

```
$ openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt
Signature ok
subject=/C=US/ST=Massachusetts/L=Bedford/O=EnterpriseDB/OU=XDB/emailAddress=s
upport@enterprisedb.com/CN=enterprisedb
Getting Private key
```

Step 3: Make a copy of the server certificate (`server.crt`) to be used as the root Certificate Authority (CA) file (`root.crt`).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ cp server.crt root.crt
```

Step 4: Delete the now redundant certificate signing request (`server.csr`).

```
$ rm server.csr
```

Step 5: Move or copy the certificate and private key files to the Postgres database server data directory, *`POSTGRES_INSTALL_HOME`*`/data`.

```
$ mv root.crt /opt/PostgresPlus/9.4AS/data
$ mv server.crt /opt/PostgresPlus/9.4AS/data
$ mv server.key /opt/PostgresPlus/9.4AS/data
```

Step 6: Set the file ownership and permissions on the certificate files and private key file.

Set the ownership to the operating system account that owns the `data` subdirectory of the Postgres database server, which is either `enterprisedb` or `postgres` depending upon the chosen installation mode (Oracle compatible or PostgreSQL compatible) when you installed your Postgres database server.

```
$ chown enterprisedb root.crt server.crt server.key
$ chgrp enterprisedb root.crt server.crt server.key
$ chmod 600 root.crt server.crt server.key
$ ls -l
total 140
        .
        .
        .
-rw------- 1 enterprisedb enterprisedb  1346 Mar 15 09:31 root.crt
-rw------- 1 enterprisedb enterprisedb  1346 Mar 15 09:30 server.crt
-rw------- 1 enterprisedb enterprisedb  1704 Mar 15 09:28 server.key
```

Step 7: In the `postgresql.conf` file, make the following modifications.

```
ssl = on                                # (change requires restart)
ssl_cert_file = 'server.crt'            # (change requires restart)
ssl_key_file = 'server.key'             # (change requires restart)
ssl_ca_file = 'root.crt'                # (change requires restart)
```

Step 8: Modify the `pg_hba.conf` file to enable SSL usage on the desired publication, subscription, or master node databases.

In the `pg_hba.conf` file, the `hostssl` type indicates the entry is used to validate SSL connection attempts from the client (the publication server and the subscription server).

The authentication method is set to `cert` with the option `clientcert=1` in order to require an SSL certificate from the client against which authentication is performed using the common name of the certificate (`enterprisedb` in this example).

The `map=sslusers` option specifies that a mapping named `sslusers` defined in the `pg_ident.conf` file is to be used for authentication. This mapping allows a connection

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

to the database if the common name from the certificate and the database user name attempting the connection match the `SYSTEM-USERNAME/PG-USERNAME` pair listed in the `pg_ident.conf` file.

The following is an example of the settings in the `pg_hba.conf` file if the publication and subscription databases (`edb` and `subnode`) must use SSL connections.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
hostssl   edb,subnode   all  192.168.2.0/24   cert clientcert=1 map=sslusers
```

Step 9: The following shows the user name maps in the `pg_ident.conf` file related to the `pg_hba.conf` file by the `map=sslusers` option. These user name maps permit you to specify database user names `pubuser`, `subuser`, `mmruser`, or `enterprisedb` in the User field of the Add Database or Update Database dialog box when adding the publication, subscription, or master node databases in the xDB Replication Console.

In other words, these are the permitted set of database user names that can be used by the publication server and subscription server to connect to the publication, subscription, or master node databases.

```
# MAPNAME       SYSTEM-USERNAME         PG-USERNAME
sslusers        enterprisedb            pubuser
sslusers        enterprisedb            subuser
sslusers        enterprisedb            mmruser
sslusers        enterprisedb            enterprisedb
```

Step 10: Restart the Postgres database server after you have made the changes to the Postgres configuration files.

7.11.2 Configuring SSL for the Publication Server and Subscription Server

After you have configured SSL on the Postgres database server, the following steps provide an example of generating a certificate and keystore file for the publication server and subscription server (the JDBC clients).

Step 1: Using files `server.crt` and `server.key` located under the Postgres database server `data` subdirectory, create copies of these files and move them to the host where the publication server and subscription server are running.

For this example, assume file `xdb.crt` is a copy of `server.crt` and `xdb.key` is a copy of `server.key`.

Step 2: Create a copy of `xdb.crt`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ cp xdb.crt xdb_root.crt
$ ls -l
total 12
-rw-r--r-- 1 user user 1346 Mar 15 09:58 xdb.crt
-rw-r--r-- 1 user user 1704 Mar 15 09:58 xdb.key
-rw-r--r-- 1 user user 1346 Mar 15 10:00 xdb_root.crt
```

Step 3: Create a Distinguished Encoding Rules (DER) format of file `xdb_root.crt`. The generated DER format of this file is `xdb_root.crt.der`. The DER format of the file is required for the `keytool` program in the next step.

```
$ openssl x509 -in xdb_root.crt -out xdb_root.crt.der -outform der
$ ls -l
total 16
-rw-r--r-- 1 user user 1346 Mar 15 09:58 xdb.crt
-rw-r--r-- 1 user user 1704 Mar 15 09:58 xdb.key
-rw-r--r-- 1 user user 1346 Mar 15 10:00 xdb_root.crt
-rw-rw-r-- 1 user user  954 Mar 15 10:05 xdb_root.crt.der
```

Step 4: Use the `keytool` program to create a keystore file (`xdb.keystore`) using `xdb_root.crt.der` as the input. This process adds the certificate of the Postgres database server to the keystore file.

The `keytool` program can be found under the `bin` subdirectory of the Java Runtime Environment installation.

You will be prompted for a new password. Save this password for the next step.

```
$ /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre/bin/keytool -keystore
xdb.keystore -alias xdbstore -import -file xdb_root.crt.der
Enter keystore password:
Re-enter new password:
Owner: CN=enterprisedb, EMAILADDRESS=support@enterprisedb.com, OU=XDB,
O=EnterpriseDB, L=Bedford, ST=Massachusetts, C=US
Issuer: CN=enterprisedb, EMAILADDRESS=support@enterprisedb.com, OU=XDB,
O=EnterpriseDB, L=Bedford, ST=Massachusetts, C=US
Serial number: d7e9966b48e91523
Valid from: Tue Mar 15 08:30:37 GMT-05:00 2016 until: Wed Mar 15 08:30:37
GMT-05:00 2017
Certificate fingerprints:
     MD5:  5D:32:AB:47:A2:44:48:84:0B:CA:EC:9E:C9:28:CE:64
     SHA1: 31:14:C4:0A:E6:93:AA:2C:3E:4B:09:77:AB:94:DB:71:CB:58:99:D9
     SHA256:
2B:EA:59:35:E6:5B:07:07:30:96:D4:80:B0:E1:13:5B:5E:45:97:2E:D0:5C:4F:D8:2F:A6
:23:DA:F8:30:D6:17
     Signature algorithm name: SHA1withRSA
     Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore
$ ls -l
total 20
-rw-r--r-- 1 user user 1346 Mar 15 09:58 xdb.crt
-rw-r--r-- 1 user user 1704 Mar 15 09:58 xdb.key
-rw-rw-r-- 1 user user 1019 Mar 15 10:18 xdb.keystore
-rw-r--r-- 1 user user 1346 Mar 15 10:00 xdb_root.crt
-rw-rw-r-- 1 user user  954 Mar 15 10:05 xdb_root.crt.der
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 5: Generate the encrypted form of the new password specified in the preceding step.

The encrypted password must be specified with the `sslTrustStorePassword` configuration option of the publication server configuration file for publication server SSL connections and the subscription server configuration file for subscription server SSL connections. (See Section 10.4.1 for information on the publication server and subscription server configuration files.)

Encrypt the password using the xDB Replication Server CLI `encrypt` command. The following example shows this process encrypting the password contained in file `infile`.

```
$ export PATH=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre/bin:$PATH
$ cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
$ java -jar edb-repcli.jar -encrypt -input ~/infile -output ~/pwdfile
$ cat ~/pwdfile
LGn6+AagiXqumxVHlOKk3w==
```

Step 6: Create a PKCS #12 format of the keystore file (`xdb_pkcs.p12`) using files `xdb.crt` and `xdb.key` as input.

You will be prompted for a new password. Save this password for the next step.

```
$ openssl pkcs12 -export -in xdb.crt -inkey xdb.key -out xdb_pkcs.p12
Enter Export Password:
Verifying - Enter Export Password:
$ ls -l
total 24
-rw-r--r-- 1 user user 1346 Mar 15 09:58 xdb.crt
-rw-r--r-- 1 user user 1704 Mar 15 09:58 xdb.key
-rw-rw-r-- 1 user user 1019 Mar 15 10:18 xdb.keystore
-rw-rw-r-- 1 user user 2557 Mar 15 10:34 xdb_pkcs.p12
-rw-r--r-- 1 user user 1346 Mar 15 10:00 xdb_root.crt
-rw-rw-r-- 1 user user  954 Mar 15 10:05 xdb_root.crt.der
```

Step 7: Generate the encrypted form of the new password specified in the preceding step,

The encrypted password must be specified with the `sslKeyStorePassword` configuration option of the publication server configuration file for publication server SSL connections and the subscription server configuration file for subscription server SSL connections.

Encrypt the password using the xDB Replication Server CLI `encrypt` command.

Step 8: Copy files `xdb.keystore` and `xdb_pkcs.p12` to a directory location where they are to be accessed by the publication server and subscription server.

Step 9: In the publication server and subscription server configuration files, set the location of file `xdb.keystore` with the `sslTrustStore` option and the location of file `xdb_pkcs.p12` with the `sslKeyStore` option.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows the SSL configuration options set for the files generated in this example.

```
sslTrustStore=/tmp/sslclient/xdb.keystore
sslTrustStorePassword=LGn6+AagiXqumxVHlOKk3w==
sslKeyStore=/tmp/sslclient/xdb_pkcs.p12
sslKeyStorePassword= ygJ9AxoJEX854elcVIJPTw==
```

The encrypted `sslTrustStorePassword` is obtained from Step 5 after being specified for the `keytool` program in Step 4.

The encrypted `sslKeyStorePassword` is obtained from Step 7 after being specified for the `openssl pkcs12` program in Step 6.

Section 7.11.4 contains a summary of the publication server and subscription server configuration options for SSL connections.

Step 10: Restart the publication and subscription servers.

7.11.3 Requesting SSL Connection to the xDB Replication Server Databases

Once SSL connectivity has been configured, a URL option must be supplied when configuring a single-master or multi-master replication system for those databases to which an SSL connection is intended to be used.

The SSL URL option informs Java to use SSL when the publication server or subscription server attempts to connect to an xDB Replication Server database (publication, subscription, or master node database) on which the SSL URL option has been set to true.

The configuration steps where these options are specified are as follows:

- For using SSL connections in a single-master replication system, the URL options must be specified as shown in Section 5.2.2 for the publication database and in Section 5.3.2 for the subscription databases.
- For using SSL connections in a multi-master replication system, the URL options must be specified as shown in Section 6.2.2 for the master definition node and in Section 6.3 for the non-MDN nodes.

For publication, subscription, and master node databases, in the URL Options field of the Add Database or Update Database dialog box, enter the following:

```
ssl=true
```

The following is an example of the Add Database dialog box with the `ssl=true` URL option.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

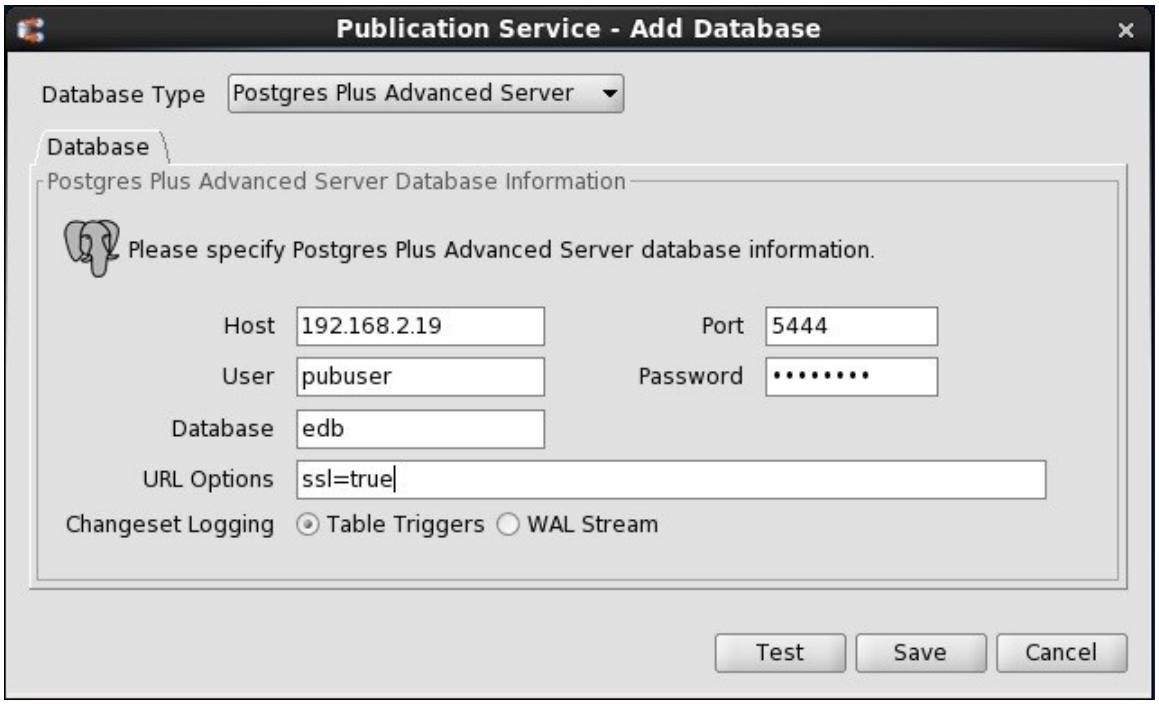


Figure 7-112 - Add Database with SSL URL option

Note: If you no longer wish to use an SSL connection to an xDB Replication Server database, you must completely delete the `ssl=true` text from the URL Options field of the Add Database or Update Database dialog box. Simply changing `true` to `false` does not have the effect of disabling the SSL option.

7.11.4 Summary of SSL Configuration Options

The following is a summary of the publication server and subscription server configuration options that are applicable to SSL connections.

sslTrustStoreType

The `sslTrustStoreType` option specifies the truststore format. Set this option to the Java truststore format of the client.

```
sslTrustStoreType=truststore_format
```

The default value for *`truststore_format`* is `jks` for the JKS truststore file format.

sslTrustStore

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The xDB Replication Server uses the default Java truststore for SSL connectivity.

The typical default location of the truststore is in directory *JAVA_HOME*/jre/lib/security or *JAVA_HOME*/lib/security in a file named cacerts. (*JAVA_HOME* is the Java installation directory.)

Specify the full directory path to the truststore file with this option.

```
sslTrustStore=truststore_file
```

sslTrustStorePassword

Encrypt the password for the Java system truststore using the xDB Replication Server CLI encrypt command (see Section 8.3.4) and specify the encrypted password with the sslTrustStorePassword option.

```
sslTrustStorePassword=encrypted_password
```

sslKeyStoreType

The sslKeyStoreType option specifies the keystore format. Set this option to the Java keystore format of the client.

```
sslKeyStoreType=keystore_format
```

The default value for *keystore_format* is pkcs12 for the PKCS #12 keystore file format.

sslKeyStore

Specify the full directory path to the keystore file with this option.

```
sslKeyStore=keystore_file
```

sslKeyStorePassword

Encrypt the password for the Java system keystore using the xDB Replication Server CLI encrypt command (see Section 8.3.4) and specify the encrypted password with the sslKeyStorePassword option.

```
sslKeyStorePassword=encrypted_password
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8 xDB Replication Server Command Line Interface

This chapter discusses the syntax and usage of the xDB Replication Server Command Line Interface (CLI). This utility program is a command line driven alternative to the xDB Replication Console.

The steps for creating a replication system using the xDB Replication Server CLI are no different than those required when using the xDB Replication Console. The logical components of the replication system must be created in the same order, with the same sets of attributes as when creating the replication system with the xDB Replication Console.

You should understand the concepts and steps presented in chapters 2, and 5 (for single-master replication) or 6 (for multi-master replication) before building a replication system using the xDB Replication Server CLI.

There are no restrictions on using both the xDB Replication Console and the xDB Replication Server CLI to build and manage the same replication system.

In Section 8.3, the syntax and examples are given for each xDB Replication Server CLI command run individually. Where applicable, the discussion of a command contains a reference back to its xDB Replication Console counterpart where a detailed description of the affected component and its attributes can be found.

8.1 Prerequisite Steps

This section describes the installation and setup required prior to using the xDB Replication Server CLI.

The xDB Replication Server CLI is included if the xDB Replication Console component is chosen when installing xDB Replication Server. The xDB Replication Server CLI is a Java application found in directory *XDB_HOME*/bin.

Step 1: Follow the installation steps given in Chapter 3 to install xDB Replication Server.

Step 2: Follow the prerequisite steps given in Section 5.1 for single-master replication systems or Section 6.1 for multi-master replication systems.

Step 3: Set the Java Runtime Environment as described by the following discussion.

On the host from which you intend to run the xDB Replication Server CLI, the Java Runtime Environment (JRE) must be present and the Java runtime bin directory must be

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

included in the path of the operating system user name that will be used to run xDB Replication Server CLI.

The xDB Startup Configuration file, `xdbReplicationServer-xx.config`, contains the path of the JRE runtime program that was detected during the installation of xDB Replication Server. The following is an example of the xDB Startup Configuration file (see Section 3.5 for the location of this file.)

```
#!/bin/sh

JAVA_EXECUTABLE_PATH="/usr/bin/java"
JAVA_MINIMUM_VERSION=1.7
JAVA_BITNESS_REQUIRED=64
JAVA_HEAP_SIZE="-Xms256m -Xmx1536m"
PUBPORT=9051
SUBPORT=9052
```

For example, using the JRE path shown in the preceding configuration file, enter the following on the command line or add it to your profile:

```
export PATH=/usr/bin:$PATH
```

On Windows systems, open the Properties dialog box of My Computer, choose Advanced System Settings, and then click on Environment Variables. Edit the `Path` system environment variable to include the Java Runtime Environment `bin` directory. Alternatively, you can set the path for just the current session when you open the Command Prompt window as in the following example:

```
SET Path=C:\Program Files\Java\jre1.8.0_45\bin;%Path%
```

8.2 General Usage

This section explains the general usage rules for the xDB Replication Server CLI.

8.2.1 Running xDB Replication Server CLI

You can run the xDB Replication Server CLI from any host on which you can run the xDB Replication Console. The xDB Replication Server CLI is run by executing the `java` runtime program and specifying the following arguments to the `java` program:

- The path to the xDB Replication Server CLI jar file `edb-repcli.jar`
- An xDB Replication Server CLI command
- One or more publication names or subscription names if the command acts on a publication or subscription
- Parameters and their values that are applicable to the command

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The Java jar file `edb-repcli.jar` is located in directory *`XDB_HOME`*`/bin`.

Each xDB Replication Server CLI command has the following general syntax:

```
-command [ { pubname | subname } ...]
         [ -parameter [ value ] ...] ...
```

In the preceding syntax diagram, *`command`* is the name of an xDB Replication Server CLI command. The command name must be prefixed by a hyphen character (-). If the command acts on a publication, the name of the publication represented by *`pubname`* is specified. If the command acts on a subscription, the subscription name represented by *`subname`* is specified. Certain commands may allow the specification of more than one publication name or more than one subscription name.

Depending upon the command, one or more parameters may follow. Each parameter name must have a hyphen prefix. You may need to specify one or more values for each given parameter.

If a command takes more than one parameter, the order in which the parameters are specified makes no difference. Each parameter must be followed only by the values that pertain to it.

Command names and parameter names are all case sensitive and must be given as shown in Section 8.3.

The general, complete, execution syntax that you enter at the command line prompt has the following format:

```
java -jar XDB_HOME/bin/edb-repcli.jar
  -command [ { pubname | subname } ...]
           [ -parameter [ value ] ...] ...
```

The preceding syntax must be entered as one logical line on the command line. It is broken up into multiple lines in the preceding syntax diagram for the purpose of clarity.

Note: You can continue a command onto the next physical line if you enter the operating system's continuation character (for example, the backslash character (\) in Linux or the caret character (^) in Windows) before pressing the Enter key.

8.2.2 Getting Help

If you execute the xDB Replication Server CLI with the `help` command, xDB Replication Server CLI will list a syntax summary of all commands.

See Section 8.3.1 for details on the `help` command.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8.2.3 Supplying the Publication or Subscription Server Login Information

This section discusses the syntax and usage of an xDB Replication Server CLI parameter, required by many commands, named `repsvrfile`. Using parameter `repsvrfile` is the xDB Replication Server CLI equivalent for the process of registering the publication server or the subscription server in the xDB Replication Console.

Section 5.2.1 discusses how the first step in building a replication system is to register the publication server. In the xDB Replication Console, the registered publication server appears as a node in the replication tree. The Publication Server node provides a context to which you can add other logical components of the replication system.

When using the xDB Replication Server CLI, there is no replication tree image available with which to relate the other logical components of the replication system. Instead, whenever you execute an xDB Replication Server CLI command that requires the context of a publication server or subscription server, you must specify the publication server's login information or the subscription server's login information by means of the `repsvrfile` parameter.

The `repsvrfile` parameter takes as its value, the path to a text file that contains the login information of either the publication server instance or the subscription server instance that you want to use. The general xDB Replication Server CLI command syntax including the `repsvrfile` parameter is shown in the following diagram:

```
-command [ { pubname | subname } ...]
         [ -parameter [ value ] ...] ...
         [ -repsvrfile repsvrfile ]
         [ -parameter [ value ] ...] ...
```

The xDB Replication Server CLI command to be executed is represented by *`command`*. If required, publication names represented by *`pubname`* or subscription names represented by *`subname`* are specified next. The path to the text file containing either the publication server or subscription server login information is represented by *`repsvrfile`*. The parameters and their values that are used with *`command`* are denoted by *`parameter`* and *`value`*.

The order on the command line in which `-repsvrfile` *`repsvrfile`* and *`-parameter`* and its values are given does not matter. For example, `-repsvrfile` *`repsvrfile`* can be given as the first parameter on the command line, the last parameter on the command line, or somewhere in between other parameters.

The following is an example of *`repsvrfile`* for a publication server:

```
host=localhost
port=9051
user=admin
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
# Password is in encrypted form.
password=ygJ9AxoJEX854elcVIJPTw==
```

The following is an example of *repsvrfile* for a subscription server:

```
host=localhost
port=9052
user=admin
# Password is in encrypted form.
password=ygJ9AxoJEX854elcVIJPTw==
```

These files can be located in any directory as long as they can be read by the operating system user running the xDB Replication Server CLI.

In your file, be sure to replace the values of the following fields with the values for your publication server or subscription server:

- Host
- Port
- User
- Password

This is the same information with which you would need to register the publication server or subscription server if you were using the xDB Replication Console. See Section 5.2.1 for additional information on registering the publication server. See Section 5.3.1 for information on registering the subscription server.

The following example illustrates how the `repsvrfile` parameter is used along with the `printpublist` command.

```
$ java -jar edb-repcli.jar -printpublist -repsvrfile ~/pubsvrfile.prop
Printing publications ...
analysts_managers
dept_emp
emp_pub
```

8.2.4 Using Encrypted Passwords in Text Files

When using the xDB Replication Server CLI, text files are used to store certain information, which may include user names and passwords. An example is the files containing publication server and subscription server login information used with the `repsvrfile` parameter.

In the file specified with parameter `repsvrfile`, the `password` field must be set to a password in encrypted form. Using an encrypted password prevents unauthorized personnel from accessing the publication server or subscription server using the values of `user` and `password` if the file was somehow compromised. (The encrypted password cannot be used to access the publication server or subscription server from its dialog box in the xDB Replication Console.)

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Section 8.3.4 for directions on generating an encrypted password using the `encrypt` command.

8.2.5 Running xDB Replication Server CLI Using a Parameter File

The `paramfile` command allows you to run an xDB Replication Server CLI command and its parameters that have been coded into a text file. This technique is useful if you want to save the command and its parameters for repeated executions.

The syntax for executing `paramfile` is shown by the following:

```
java -jar XDB_HOME/bin/edb-repcli.jar
  -paramfile cmdparamfile
```

The syntax of the xDB Replication Server CLI command and its parameters coded into text file *`cmdparamfile`* is the same as if given at the command line prompt as shown by the following:

```
-command [ { pubname | subname } ...]
         [ -parameter [ value ] ...] ...
         [ -repsvrfile repsvrfile ]
         [ -parameter [ value ] ...] ...
```

Using the `paramfile` command has the following restrictions:

- Only one xDB Replication Server CLI command can be coded into the parameter file *`cmdparamfile`*.
- The parameters to be used with the xDB Replication Server CLI command must all be included in *`cmdparamfile`*. You cannot code some of the parameters into *`cmdparamfile`* and give other parameters on the command line.

The following example creates an Advanced Server publication database definition using a parameter file named `addpubdb_advsvr`.

The following is the content of parameter file `addpubdb_advsvr`:

```
-addpubdb
  -repsvrfile /home/user/pubsvrfile.prop
  -dbtype enterprisedb
  -dbhost 192.168.2.4
  -dbport 5444
  -dbuser pubuser
  -dbpassword ygJ9AxoJEX854elcVIJPTw==
  -database edb
  -repgrouptype s
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For Windows only: The `-repsvrfile` directory path can be specified with either the forward slash or backslash character. Enclose the entire directory path in double quotation marks if a directory name contains space characters:

```
-addpubdb
  -repsvrfile "C:\Users\User Name\repcli\pubsvrfile.prop"
  -dbtype enterprisedb
  -dbhost 192.168.2.23
  -dbport 5444
  -dbuser pubuser
  -dbpassword ygJ9AxoJEX854elcVIJPTw==
  -database edb
  -repgrouptype s
```

Note: Unlike entering the xDB Replication Server CLI command and its parameters directly at the command line prompt, when coded into a text file, no continuation characters are needed to continue onto the following lines.

The following shows the execution of the `paramfile` command:

```
$ java -jar edb-repcli.jar -paramfile ~/addpubdb_advsvr
Adding Publication Database...
Publication database added successfully. Publication Database id:1
```

8.2.6 Testing the Command Exit Status

After executing an xDB Replication Server CLI command, you can test the exit status to determine if the command execution was successful.

An exit status of `0` indicates successful execution. A non-zero exit status indicates a failure has occurred.

Table 8-1 - Replication Server CLI Exit Status Codes

Exit Status	Description
0	Success
201	Invalid command
202	I/O error
203	Decryption failed
204	Authentication failed
205	Publication service failure
206	Remote exception
207	Subscription service failure

For Linux only: The environment variable, `$?`, contains the exit status.

The following example shows the `0` exit status upon the successful execution of the `addpubdb` command contained in the `addpubdb_advsvr` parameter file described in Section 8.2.5.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ java -jar edb-repcli.jar -paramfile ~/addpubdb_advsvr
Adding publication database...
Publication database added successfully. Publication database id:1
$ echo $?
0
```

On the other hand, the following example shows a non-zero exit status when the command failed with an error.

```
$ java -jar edb-repcli.jar -paramfile ~/addpubdb_advsvr
Adding publication database...
Error:The connection attempt failed.
$ echo $?
200
```

For Windows only: The environment variable, `%ERRORLEVEL%`, contains the exit status.

The following shows the exit status upon successful command execution on a Windows system.

```
C:\Users>java -jar C:\\"Program Files"\PostgreSQL\EnterpriseDB-
xDBReplicationServer\bin\edb-repcli.jar -paramfile addpubdb_advsvr
Adding publication database...
Publication database added successfully. Publication database id:1

C:\Users>ECHO %ERRORLEVEL%
0
```

The following shows the exit status upon unsuccessful command execution on a Windows system.

```
C:\Users>java -jar C:\\"Program Files"\PostgreSQL\EnterpriseDB-
xDBReplicationServer\bin\edb-
repcli.jar -paramfile addpubdb advsvr
Adding publication database...
Error:FATAL: password authentication failed for user "myuser"

C:\Users>ECHO %ERRORLEVEL%
200
```

8.3 xDB Replication Server CLI Commands

This section provides a description, syntax diagram, and examples of each xDB Replication Server CLI command.

Commands are presented in the order in which they will typically be used, following the order in which xDB Replication Console operations are performed.

Note: Though most commands described in this section apply to both single-master and multi-master replication systems, those commands that apply only to single-master replication systems are noted with **For SMR only**. Those commands that apply only to multi-master replication systems are noted with **For MMR only**. The same notation is

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

used for command parameters that may apply only to single-master replication systems or multi-master replication systems.

For the examples used in this section, it is assumed that the xDB Replication Server CLI commands are executed after you have made *XDB_HOME*/bin your current working directory, thereby eliminating the need to specify the full path of *XDB_HOME*/bin for each execution of the edb-repcli.jar file. For example, assuming xDB Replication Server is installed in the default installation directory you have issued the following command in Linux:

```
cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
```

In Windows, the equivalent is the following:

```
cd C:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer\bin
```

Whenever the repsvrfile parameter appears in the examples, file ~/pubsvrfile contains the publication server login information and is located in the user's home directory while ~/subsvrfile contains the subscription server login information. For Windows, the equivalent usage is %HOMEPATH%\pubsvrfile and %HOMEPATH%\subsvrfile.

The examples in this section were run on Linux so you will see use of the Linux continuation character, which is a backslash (\), to show how an xDB Replication Server CLI command can be continued onto the next line if you do not want to wrap the text in your terminal window. For Windows, use the Windows continuation character, which is a caret (^).

8.3.1 Getting Help (help)

The help command provides a syntax summary of all xDB Replication Server CLI commands.

Synopsis

```
-help
```

Examples

```
$ java -jar edb-repcli.jar -help
Usage: java -jar edb-repcli.jar [OPTIONS]

Where OPTIONS include:
-help   Prints out Replication CLI command-line usage
-version        Prints out Replication CLI version
-encrypt -input <file> -output <file>   Encrypts input file to output file
-repversion -repsvrfile <file>  Prints Replication Server version
                        .
                        .
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
.
```

8.3.2 Printing the Version Number (version)

The `version` command provides the xDB Replication Server CLI's version number.

Synopsis

```
-version
```

Examples

```
$ java -jar edb-repcli.jar -version
Version: 6.1.0-alpha
```

8.3.3 Printing the xDB Replication Server Version Number (repversion)

The `repversion` command provides the xDB Replication Server's version number.

Synopsis

```
-repversion -repsvrfile pubsvrfile
```

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

```
$ java -jar edb-repcli.jar -repversion -repsvrfile ~/pubsvrfile.prop
6.1.0-alpha
```

8.3.4 Encrypting Passwords (encrypt)

The `encrypt` command encrypts the text supplied in an input file and writes the encrypted result to a specified output file. Use the `encrypt` command to generate an encrypted password that can be copied into a text file that will be referenced by an xDB Replication Server CLI command that requires a user name and the user's password.

Synopsis

```
-encrypt -input infile -output pwdfile
```

The text in *infile* is processed using the MD5 encryption algorithm, and the encrypted text is written to file *pwdfile*. Make sure that *infile* contains only the text that you

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

want to encrypt and that there are no extraneous characters or empty lines before the text or after the text that you want to encrypt.

Parameters

infile

The file containing the text to be encrypted.

pwdfile

The file containing the encrypted form of the text from *infile*.

Examples

The file `infile` contains the word "password".

```
password
```

The `encrypt` command is then executed producing a file named `pwdfile`.

```
$ java -jar edb-repcli.jar -encrypt -input ~/infile -output ~/pwdfile
```

The content of file `pwdfile` contains the encrypted form of "password".

```
ygJ9AxoJEX854elcVIJPTw==
```

8.3.5 Printing the Time the Server Has Been Running (uptime)

The `uptime` command prints the time interval since the publication server has been up and running.

Synopsis

```
-uptime -repsvrfile pubsvrfile
```

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

```
$ java -jar edb-repcli.jar -uptime -repsvrfile ~/pubsvrfile.prop
0 days 0 hours 4 minutes
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8.3.6 Adding a Publication Database (addpubdb)

The `addpubdb` command adds a publication database definition.

Synopsis

```
-addpubdb
  -repsvrfile pubsvrfile
  -dbtype { oracle | enterprisedb | postgresql | sqlserver }
  -dbhost host
  -dbport port
  -dbuser user
{ -dbpassword encrypted_pwd | -dbpassfile pwdfile }
[ -oraconnectiontype { sid | servicename } ]
  -database dbname
[ -urloptions jdbc_url_parameters ]
[ -filterrule filterid_1[,filterid_2 ] ...]
[ -repgrouptype { m | s } ]
[ -replicatepubschema { true | false } ]
[ -initialsnapshot
  [ -verboseSnapshotOutput { true | false } ] ]
[ -nodepriority priority_level ]
[ -changesetlogmode { T | W } ]
```

The `addpubdb` command creates a new publication database definition. The `addpubdb` command displays a unique publication database ID that is assigned to the newly created publication database definition. The publication database ID is used to identify the publication database definition on which to operate when running other xDB Replication Server CLI commands.

See Section 5.2.2 for details on the database connection information that must be supplied when adding a publication database definition for a single-master replication system. See sections 6.2.2 and 6.3 for a multi-master replication system.

Parameters

pubsvrfile

The file containing the publication server login information.

-dbtype

Specify `oracle` if the database is an Oracle database. Specify `enterprisedb` if the database is an Advanced Server database in Oracle compatible configuration mode. Specify `postgresql` if the database is a PostgreSQL database or an Advanced Server database in PostgreSQL compatible configuration mode. Specify `sqlserver` if the database is a Microsoft SQL Server database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

host

The IP address of the host on which the publication database server is running.

port

The port number on which the database server is listening for connections.

user

The publication database user name.

encrypted_pwd

The encrypted password of the publication database user. See Section 8.3.4 for directions on using the `encrypt` command to generate an encrypted password.

pwdfile

The file containing the encrypted password of the publication database user.

`-oraconnectiontype`

Specify `sid` if the Oracle system ID (SID) is used to identify the publication database in the `database` parameter. Specify `servicename` if the Oracle service name is used to identify the publication database in the `database` parameter. **Note:** For Oracle 12c, use the service name.

dbname

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the publication database.

jdbc_url_parameters

Extended usage of JDBC URL parameters such as for support of SSL connectivity. (See Section 7.11 for information on SSL connectivity to the publication database.)

filterid_n

For MMR only: Applies to non-MDN nodes. Comma-separated list of filter IDs identifying the filter rules from the set of available table filters to enable on the corresponding tables in the new master node. Use the `printpubfilterslist` command to obtain the filter IDs for the available filter rules in the publication (see Section 8.3.17). **Note:** There must be no white space between the comma and filter IDs.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

`-repgrouptype`

Specify `s` if this command applies to a single-master replication system. Specify `m` if this command applies to a multi-master replication system. If omitted, the default is `s`.

`-replicatepubschema`

For MMR only: Applies to non-MDN nodes. Set this option to `true` if you want the publication table definitions replicated from the master definition node when creating a new master node. Set this option to `false` if you have already created the table definitions in the new master node. If omitted, the default is `true`. Do not specify this parameter when creating the master definition node.

`-initialsnapshot`

For MMR only: Applies to non-MDN nodes. Specify this option if you want an initial snapshot replication to be performed when creating the master node. Omit this option if you do not want an initial snapshot replication to be performed when creating the master node.

Note (For MMR only): Unless you intend to use the offline snapshot technique (see Section 7.9), it is suggested that you specify this option. An initial snapshot replication must be performed from the master definition node to every other master node before performing synchronization replications on demand (see Section 8.3.42) or by a schedule (see Section 8.3.44). If a newly added master node did not undergo an initial snapshot, any subsequent synchronization replication may fail to apply the transactions to that master node. The initial snapshot can also be taken by performing an on demand snapshot (see Section 8.3.41).

`-verboseSnapshotOutput`

Set this option to `true` if you want the output from the snapshot to be displayed. Set this option to `false` if you do not want the snapshot output displayed. If omitted, the default is `true`.

Note: This option may be given only directly following the specification of the `-initialsnapshot` option.

`priority_level`

For MMR only: Integer value from 1 through 10 assigning the priority level to a master node with 1 having the highest priority and 10 having the lowest priority.

`-changesetlogmode`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Specify `T` to use the trigger-based method of synchronization replication for this publication database. Specify `W` to use the log-based (WAL) method of synchronization replication for this publication database. If omitted, the default is `T`.

Examples

The following example adds a publication database definition for an Oracle database. The encrypted password is given on the command line with the `dbpassword` parameter. A publication database ID of `1` is assigned to the database by the publication service.

```
$ java -jar edb-repcli.jar -addpubdb -repsvrfile ~/pubsvrfile.prop \
>   -dbtype oracle -dbhost 192.168.2.6 -dbport 1521 \
>   -dbuser pubuser -dbpassword ygJ9AxoJEX854elcVIJPTw== \
>   -oraconnectiontype sid \
>   -database xe \
>   -repgrouptype s
Adding publication database...
Publication database added successfully. Publication database id:1
```

The following example adds a publication database definition for an Advanced Server database. The encrypted password is read from a file named `pwdfile` with the `dbpassfile` parameter. A publication database ID of `2` is assigned to the database by the publication service.

```
$ java -jar edb-repcli.jar -addpubdb -repsvrfile ~/pubsvrfile.prop \
>   -dbtype enterprisedb -dbhost 192.168.2.7 -dbport 5444 \
>   -dbuser pubuser -dbpassfile ~/pwdfile \
>   -database edb \
>   -repgrouptype s
Adding publication database...
Publication database added successfully. Publication database id:2
```

The following example adds a publication database definition for a master definition node in a multi-master replication system.

```
$ java -jar edb-repcli.jar -addpubdb -repsvrfile ~/pubsvrfile.prop \
>   -dbtype enterprisedb -dbhost 192.168.2.6 -dbport 5444 \
>   -dbuser pubuser -dbpassword ygJ9AxoJEX854elcVIJPTw== \
>   -database edb \
>   -repgrouptype m \
>   -nodepriority 1
Adding publication database...
Publication database added successfully. Publication database id:3
```

The following example adds a publication database definition for a master node (other than the master definition node) in a multi-master replication system where an initial snapshot is not invoked (the `initialsnapshot` parameter is omitted). Filter rules with filter IDs `8` and `16` are applied to this master node. A node priority level of 3 is assigned to the master node.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: A publication must be created in the master definition node before creating additional master nodes. See Section 8.3.14 for the command to create a publication.

```
$ java -jar edb-repcli.jar -addpubdb -repsvrfile ~/pubsvrfile.prop \
>   -dbtype enterprisedb -dbhost 192.168.2.7 -dbport 5444 \
>   -dbuser mmruser -dbpassword ygJ9AxoJEX854elcVIJPTw== \
>   -database mmrnode \
>   -filterrule 8,16 \
>   -repgrouptype m \
>   -nodepriority 3
Adding publication database...
Replicating publication schema...
Publication database added successfully. Publication database id:24
```

8.3.7 Printing Publication Database IDs (printpubdbids)

The `printpubdbids` command prints the publication database IDs of the publication database definitions.

Synopsis

```
-printpubdbids -repsvrfile pubsvrfile
```

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

The following example lists the publication database IDs of the publication database definitions.

```
$ java -jar edb-repcli.jar -printpubdbids -repsvrfile ~/pubsvrfile.prop
Printing publication database ids...
2
1
24
3
```

8.3.8 Printing Publication Database Details (printpubdbidsdetails)

The `printpubdbidsdetails` command prints the connection information for each publication database definition.

Synopsis

```
-printpubdbidsdetails –repsvrfile pubsvrfile
```

The output has the following components:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

dbid:*host*:*port*:*dbname*:*user*

Note: The database user's password is not displayed.

Parameters

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID assigned to the publication database definition.

host

The IP address of the host on which the publication database server is running.

port

The port number on which the database server is listening for connections.

dbname

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the publication database.

user

The publication database user name.

Examples

There are four publication database definitions subordinate to the publication server identified by the content of file `pubsvrfile.prop`.

```
$ java -jar edb-repcli.jar -printpubdbidsdetails \
>   -repsvrfile ~/pubsvrfile.prop
Printing publication database ids with details...
id:host:port:database|sid:user
2:192.168.2.7:5444:edb:pubuser
1:192.168.2.6:1521:xe:pubuser
24:192.168.2.7:5444:mmrnode:mmruser
3:192.168.2.6:5444:edb:pubuser
```

8.3.9 Printing the Controller Database ID (printcontrollerdbid)

The `printcontrollerdbid` command prints the publication database ID of the controller database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Synopsis

```
-printcontrollerdbid -repsvrfile pubsvrfile
```

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

The following example prints the publication database ID of the controller database.

```
$ java -jar edb-repcli.jar -printcontrollerdbid -repsvrfile ~/pubsvrfile.prop
Printing Controller database id...
1
```

8.3.10 Printing the Master Definition Node Database ID (printmdndbid)

For MMR only: The `printmdndbid` command prints the publication database ID of the master definition node.

Synopsis

```
-printmdndbid -repsvrfile pubsvrfile
```

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

The following example prints the publication database ID of the master definition node.

```
$ java -jar edb-repcli.jar -printmdndbid -repsvrfile ~/pubsvrfile.prop
Printing MDN Publication database id...
3
```

8.3.11 Updating a Publication Database (updatepubdb)

The `updatepubdb` command provides the ability to change the connection information for an existing publication database definition identified by its publication database ID.

Synopsis

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
-updatepubdb
  -repsvrfile pubsvrfile
  -pubdbid dbid
  -dbhost host
  -dbport port
  -dbuser user
{ -dbpassword encrypted_pwd | -dbpassfile pwdfile }
[ -oraconnectiontype { sid | servicename } ]
  -database dbname
[ -urloptions jdbc_url_parameters ]
[ -nodepriority priority_level ]
```

The publication database definition to be updated is identified by the `pubdbid` parameter.

See Section 5.2.2 for details on the database connection information that must be supplied for a publication database definition for a single-master replication system. See sections 6.2.2 and 6.3 for a multi-master replication system.

Parameters

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the publication database definition to be updated.

host

The IP address of the host on which the publication database server is running.

port

The port number on which the database server is listening for connections.

user

The publication database user name.

encrypted_pwd

The password of the database user in encrypted form. See Section 8.3.4 for directions on using the `encrypt` command to generate an encrypted password.

pwdfile

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The file containing the password of the database user in encrypted form.

`-oraconnectiontype`

Specify `sid` if the Oracle system ID (SID) is used to identify the publication database in the `database` parameter. Specify `servicename` if the Oracle service name is used to identify the publication database in the `database` parameter. **Note:** For Oracle 12c, use the service name.

`dbname`

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the publication database.

`jdbc_url_parameters`

Extended usage of JDBC URL parameters such as for support of SSL connectivity. (See Section 7.11 for information on SSL connectivity to the publication database.) Specification of the `urloptions` parameter completely replaces any existing JDBC URL parameters that may have previously been specified with this database. Omission of the `urloptions` parameter deletes any existing JDBC URL parameters that may have previously been specified with this database.

`priority_level`

For MMR only: Integer value from 1 through 10 assigning the priority level to a master node with 1 having the highest priority and 10 having the lowest priority.

Examples

In the following example, an existing publication database definition with publication database ID `1` is updated.

```
$ java -jar edb-repcli.jar -updatepubdb -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 1 \
>   -dbhost 192.168.2.6 -dbport 1521 \
>   -dbuser pubuser -dbpassfile ~/pwdfile \
>   -oraconnectiontype sid \
>   -database xe
Updating publication database ...
Publication database with ID 1 is updated successfully.
```

8.3.12 Removing a Publication Database (removepubdb)

The `removepubdb` command removes a publication database definition.

Synopsis

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
-removepubdb
  -repsvrfile pubsvrfile
  -pubdbid dbid
```

The publication database definition to be removed is identified by the `pubdbid` parameter.

See Section 7.6.7 for additional information on removing a publication database.

Parameters

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the publication database definition to be removed.

Examples

The publication database definition identified by publication database ID `1` is removed.

```
$ java -jar edb-repcli.jar -removepubdb -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 1
Removing Publication Database...
Publication Database removed.
```

8.3.13 Get Tables for a New Publication (gettablesfornewpub)

The `gettablesfornewpub` command lists the tables and views that are available for inclusion in a new publication from a given publication database definition.

Synopsis

```
-gettablesfornewpub -repsvrfile repsvrfile -pubdbid dbid
```

Parameters

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the publication database definition whose available tables and views are to be listed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Examples

For the publication database definition identified by publication database ID `1`, the tables available for inclusion in a publication are `EDB.DEPT`, `EDB.EMP`, and `EDB.JOBHIST`. The view available for inclusion in a publication is `EDB.SALESEMP`.

```
$ java -jar edb-repcli.jar -gettablesfornewpub \
>   -repsvrfile ~/pubsvrfile.prop -pubdbid 1
Fetching tables/views list ...
[[EDB.DEPT, TABLE], [EDB.EMP, TABLE], [EDB.JOBHIST, TABLE], [EDB.SALESEMP,
VIEW]]
```

8.3.14 Creating a Publication (createpub)

The `createpub` command creates a new publication.

Synopsis

```
-createpub pubname
  -repsvrfile pubsvrfile
  -pubdbid dbid
  -reptype { s | t }
  -tables schema_t1.table_1 [ schema_t2.table_2 ] ...
[ -views schema_v1.view_1 [ schema_v2.view_2 ] ...]
[ -tablesfilterclause
    "ordinal_t1:filtername_t1:filterclause_t1"
  [ "ordinal_t2:filtername_t2:filterclause_t2" ] ...]
[ -viewsfilterclause
    "ordinal_v1:filtername_v1:filterclause_v1"
  [ "ordinal_v2:filtername_v2:filterclause_v2" ] ...]
[ -conflictresolution
    ordinal_t1:{ E | L | N | M | C:customhandler_t1 }
  [ ordinal_t2:{ E | L } N | M | C:customhandler_t2 } ] ...]
[ -standbyconflictresolution
    ordinal_t1:{ E | L | N | M | C:customhandler_t1 }
  [ ordinal_t2:{ E | L } N | M | C:customhandler_t2 } ] ...]
[ -repgrouptype { m | s } ]
```

The `createpub` command adds a new publication subordinate to the publication database definition with the publication database ID given by parameter `pubdbid`. If the publication is designated as snapshot-only by setting parameter `reptype` to `s`, then any views listed after the `views` parameter are ignored.

See Section 5.2.3 for additional information on creating a publication for a single-master replication system. See Section 6.2.3 for a multi-master replication system.

Note: The schema names, table names, and view names that you supply as values for the `tables` and `views` parameters are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example `edb.dept`). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The publication name to be given to the new publication.

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the publication database definition sub ordinate to which the new publication is to be added.

`-reptype`

Specify `s` if the publication is to be a snapshot-only publication. Specify `t` if the publication is to allow synchronization replications.

schema_tn

The name of the schema containing the *n*th table of the `tables` parameter list. This value is case-sensitive.

table_n

The table name of the *n*th table in the `tables` parameter list. This value is case-sensitive.

schema_vn

For SMR only: The name of the schema containing the *n*th view of the `views` parameter list. This value is case-sensitive.

view_n

For SMR only: View name of the *n*th view in the `views` parameter list. This value is case-sensitive.

ordinal_tn

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The ordinal number (that is, the position in the list counting from left to right starting with 1) of a table in the `tables` parameter list to which an attribute is to be applied.

`filtername_tn`

The filter name to be assigned to the filter rule on the table.

`filterclause_tn`

The filter clause to be applied to the table in the `tables` parameter list at the position indicated by *`ordinal_tn`*.

`ordinal_vn`

For SMR only: The ordinal number (that is, the position in the list counting from left to right starting with 1) of a view in the `views` parameter list to which an attribute is to be applied.

`filtername_vn`

The filter name to be assigned to the filter rule on the view.

`filterclause_vn`

For SMR only: The filter clause to be applied to the view in the `views` parameter list at the position indicated by *`ordinal_vn`*.

`-conflictresolution`

For MMR only: For the conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling. The specified conflict resolution applies to the table in the position given by *`ordinal_tn`* counting from left to right in the `tables` parameter list. If omitted the default is `E`.

`-standbyconflictresolution`

For MMR only: For the standby conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling. The specified conflict resolution applies to the table in the position given by *`ordinal_tn`* counting from left to right in the `tables` parameter list. If omitted the default is `M`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

customhandler_tn

> **For MMR only:** For the conflict resolution option or the standby conflict resolution option, specify *customhandler_tn* as the function name with an optional schema prefix (that is, formatted as *schema*.*function_name*) as given in the CREATE FUNCTION command for the custom conflict handling function created for the table in the tables parameter list at the position indicated by *ordinal_tn*. The custom conflict handling function must be added to the master definition node. See Section 6.6.8.2 for an example of adding the custom conflict handling function using PSQL. The custom handler name option must be specified if and only if the conflict resolution option or the standby conflict resolution option is set for custom conflict handling with the c value.

-repgrouptype

> Specify s if this command applies to a single-master replication system. Specify m if this command applies to a multi-master replication system. If omitted, the default is s.

Examples

In the following example, a publication named dept_emp is created that contains the EDB.DEPT and EDB.EMP tables of an Oracle database. The replication method is synchronization.

```
$ java -jar edb-repcli.jar -createpub dept_emp \
>   -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 1 \
>   -reptype t \
>   -tables EDB.DEPT EDB.EMP
Creating publication...
Tables:[[EDB.DEPT, TABLE], [EDB.EMP, TABLE]]
Filter clause:[]
Publication created.
```

In the following example, a publication named salesemp is created that contains the EDB.SALESEMP view of an Oracle database. The replication method is snapshot-only.

```
$ java -jar edb-repcli.jar -createpub salesemp \
>   -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 1 \
>   -reptype s \
>   -views EDB.SALESEMP
Creating publication...
Tables:[[EDB.SALESEMP, VIEW]]
Filter clause:[]
Publication created.
```

In the following example, a publication named analysts_managers is created that contains the edb.dept table and employees from the edb.emp table who are analysts or

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

managers. The tables are in an Advanced Server database. The replication method is snapshot-only.

```
$ java -jar edb-repcli.jar -createpub analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 2 \
>   -reptype s \
>   -tables edb.dept edb.emp \
>   -tablesfilterclause "2:jobgrade_11:job IN ('ANALYST', 'MANAGER')"
Creating publication...
Tables:[[edb.dept, TABLE], [edb.emp, TABLE]]
Filter clause:[FilterName:jobgrade_11   FilterClause:job IN ('ANALYST',
'MANAGER')      ]
Publication created.
```

The following example creates a publication for a multi-master replication system. One table filter is defined on table `edb.dept` and three table filters are defined on table `edb.emp`. Table `edb.dept` is assigned node priority conflict resolution and latest timestamp as the standby conflict resolution strategy. Table `edb.emp` is assigned earliest timestamp conflict resolution and manual resolution (the default) as its standby strategy.

```
$ java -jar edb-repcli.jar -createpub emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 3 \
>   -reptype t \
>   -tables edb.dept edb.emp \
>   -tablesfilterclause "1:dept_10_20_30:deptno in (10, 20, 30)" \
>     "2:dept_10:deptno = 10" \
>     "2:dept_20:deptno = 20" \
>     "2:dept_30:deptno = 30" \
>   -conflictresolution 1:N 2:E \
>   -standbyconflictresolution 1:L 2:M \
>   -repgrouptype m
Creating publication...
Tables:[[edb.dept, TABLE], [edb.emp, TABLE]]
Filter clause:[FilterName:dept_10_20_30 FilterClause:deptno in (10, 20, 30)
, FilterName:dept_10   FilterClause:deptno = 10 , FilterName:dept_20
FilterClause:deptno = 20        , FilterName:dept_30   FilterClause:deptno =
30 ]
Conflict Resolution Option:[ Node Priority, Earliest Timestamp ]
Standby Conflict Resolution Option:[ Latest Timestamp, Manual ]
Publication created.
```

8.3.15 Printing a List of Publications (printpublist)

The `printpublist` command prints a list of publication names.

Synopsis

```
-printpublist -repsvrfile pubsvrfile
 [ -pubdbid dbid ]
 [ -printpubid ]
```

Parameters

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

pubsvrfile

The file containing the publication server login information.

dbid

If the `pubdbid` parameter is specified, only the publication names subordinate to the publication database definition specified by *dbid* are printed. If the `pubdbid` parameter is omitted, all publication names subordinate to the publication server are printed.

`-printpubid`

Specify this parameter to print the publication IDs as well as the publication names.

Examples

```
$ java -jar edb-repcli.jar -printpublist -repsvrfile ~/pubsvrfile.prop
Printing publications ...
analysts_managers
dept_emp
emp_pub
salesemp
```

8.3.16 Printing a List of Tables in a Publication (printpublishedtables)

The `printpublishedtables` command prints a list of tables and views that belong to the given publication.

Synopsis

```
-printpublishedtables pubname -repsvrfile pubsvrfile
```

Parameters

pubname

The name of the publication whose tables and views are to be printed.

pubsvrfile

The file containing the publication server login information.

Examples

The tables belonging to publication `dept_emp` are printed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ java -jar edb-repcli.jar -printpublishedtables dept_emp \
>   -repsvrfile ~/pubsvrfile.prop
Printing tables under publication: dept_emp

EDB.DEPT
EDB.EMP
```

8.3.17 Printing a List of Filters in a Publication (printpubfilterslist)

The `printpubfilterslist` command prints a list of table filters that are defined in the given publication.

Synopsis

```
-printpubfilterslist pubname -repsvrfile pubsvrfile
```

Parameters

pubname

The name of the publication whose table filters are to be printed.

pubsvrfile

The file containing the publication server login information.

Examples

The table filters in publication `analysts_managers` are printed.

```
$ java -jar edb-repcli.jar -printpubfilterslist analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop
Printing publications ...
FilterID:47      FilterName:jobgrade_11  FilterClause:job IN ('ANALYST',
'MANAGER')
```

The table filters defined in publication `emp_pub` are printed.

```
$ java -jar edb-repcli.jar -printpubfilterslist emp_pub \
>   -repsvrfile ~/pubsvrfile.prop
Printing publications ...
FilterID:8       FilterName:dept_10_20_30        FilterClause:deptno in (10,
20, 30)
FilterID:9       FilterName:dept_10      FilterClause:deptno = 10
FilterID:10      FilterName:dept_20      FilterClause:deptno = 20
FilterID:16      FilterName:dept_30      FilterClause:deptno = 30
```

8.3.18 Adding Tables to a Publication (addtablesintopub)

The `addtablesintopub` command adds tables or views into an existing publication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Synopsis

```
-addtablesintopub pubname
  -repsvrfile pubsvrfile
[ -tables schema_t1.table_1 [ schema_t2.table_2 ] ...]
[ -views schema_v1.view_1 [ schema_v2.view_2 ] ...]
[ -tablesfilterclause
    "ordinal_t1:filtername_t1:filterclause_t1"
  [ "ordinal_t2:filtername_t2:filterclause_t2" ] ...]
[ -viewsfilterclause
    "ordinal_v1:filtername_v1:filterclause_v1"
  [ "ordinal_v2:filtername_v2:filterclause_v2" ] ...]
[ -conflictresolution
    ordinal_t1:{ E | L | N | M | C:customhandler_t1 }
  [ ordinal_t2:{ E | L } N | M | C:customhandler_t2 } ] ...]
[ -standbyconflictresolution
    ordinal_t1:{ E | L | N | M | C:customhandler_t1 }
  [ ordinal_t2:{ E | L } N | M | C:customhandler_t2 } ] ...]
[ -repgrouptype { m | s } ]
```

The `addtablesintopub` command updates an existing publication identified by *pubname*. If the publication is snapshot-only, then any views listed after the `views` parameter are ignored.

See Section 7.6.3.1 for additional information on adding tables to a publication.

Note: The schema names, table names, and view names that you supply as values for the `tables` and `views` parameters are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example `edb.dept`). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The name of the publication to which tables or views are to be added.

pubsvrfile

The file containing the publication server login information.

schema_tn

The name of the schema containing the *n*th table of the `tables` parameter list. This value is case-sensitive.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

table_n

The name of the *n*th table in the `tables` parameter list. This value is case-sensitive.

schema_vn

For SMR only: The name of the schema containing the *n*th view of the `views` parameter list. This value is case-sensitive.

view_n

For SMR only: The name of the *n*th view in the `views` parameter list. This value is case-sensitive.

ordinal_tn

The ordinal number (that is, the position in the list counting from left to right starting with 1) of a table in the `tables` parameter list to which an attribute is to be applied.

filtername_tn

The filter name to be assigned to the filter rule on the table.

filterclause_tn

The filter clause to be applied to the table in the `tables` parameter list at the position indicated by *ordinal_tn*.

ordinal_vn

For SMR only: The ordinal number (that is, the position in the list counting from left to right starting with 1) of a view in the `views` parameter list to which an attribute is to be applied.

filtername_vn

The filter name to be assigned to the filter rule on the view.

filterclause_vn

For SMR only: The filter clause to be applied to the view in the `views` parameter list at the position indicated by *ordinal_vn*.

`-conflictresolution`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For MMR only: For the conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling. The specified conflict resolution applies to the table in the position given by *`ordinal_tn`* counting from left to right in the `tables` parameter list. If omitted the default is `E`.

`-standbyconflictresolution`

For MMR only: For the standby conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling. The specified conflict resolution applies to the table in the position given by *`ordinal_tn`* counting from left to right in the `tables` parameter list. If omitted the default is `M`.

`customhandler_tn`

For MMR only: For the conflict resolution option or the standby conflict resolution option, specify *`customhandler_tn`* as the function name with an optional schema prefix (that is, formatted as *`schema`*`.`*`function_name`*) as given in the `CREATE FUNCTION` command for the custom conflict handling function created for the table in the `tables` parameter list at the position indicated by *`ordinal_tn`*. The custom conflict handling function must be added to the master definition node. See Section 6.6.8.2 for an example of adding the custom conflict handling function using PSQL. The custom handler name option must be specified if and only if the conflict resolution option or the standby conflict resolution option is set for custom conflict handling with the `C` value.

`-repgrouptype`

Specify `s` if this command applies to a single-master replication system. Specify `m` if this command applies to a multi-master replication system. **Note:** This parameter is not required and may be completely omitted. It is present to provide support for its usage in previous xDB Replication Server CLI versions.

Examples

In the following example, table `edb.jobhist` and view `edb.salesemp` are added to an existing publication named `analysts_managers`.

```
$ java -jar edb-repcli.jar -addtablesintopub analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -tables edb.jobhist \
>   -views edb.salesemp
Adding tables to publication analysts_managers ...
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Tables:[[edb.jobhist, TABLE], [edb.salesemp, VIEW]]
Filter clause:[null, null]
Publication updated successfully
```

8.3.19 Removing Tables from a Publication (removetablesfrompub)

The `removetablesfrompub` command removes tables from a publication.

Synopsis

```
-removetablesfrompub pubname
  -repsvrfile pubsvrfile
[ -tables schema_t1.table_1 [ schema_t2.table_2 ] ...]
[ -views schema_v1.view_1 [ schema_v2.view_2 ] ...]
```

See Section 7.6.3.2 for additional information on removing tables from a publication.

Note: The schema names, table names, and view names that you supply as values for the `tables` and `views` parameters are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example `edb.dept`). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The name of the publication from which tables or views are to be removed.

pubsvrfile

The file containing the publication server login information.

schema_tn

The name of the schema containing the *n*th table of the `tables` parameter list. This value is case-sensitive.

table_n

The name of the *n*th table in the `tables` parameter list. This value is case-sensitive.

schema_vn

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The name of the schema containing the *n*th view of the `views` parameter list. This value is case-sensitive.

view_n

The name of the *n*th view in the `views` parameter list. This value is case-sensitive.

Examples

In the following example, table `edb.jobhist` and view `edb.salesemp` are removed from the `analysts_managers` publication.

```
$ java -jar edb-repcli.jar -removetablesfrompub analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -tables edb.jobhist \
>   -views edb.salesemp
Removing tables and views from publication analysts_managers ...

Tables and views removed successfully
```

8.3.20 Adding Table Filters to a Publication (addfilter)

The `addfilter` command adds the definition of table filter rules to the specified publication.

This makes the filter rules available for subsequent enablement on target subscriptions or non-MDN nodes.

Enabling a filter rule on a specified, target subscription or non-MDN node results in the filtering of data during replication from the source table to the target table.

If the filter rule is not enabled on a target subscription or non-MDN node, then it has no impact during replication on such subscription or non-MDN node.

See Section 8.3.38 for information on enabling table filter rules.

Synopsis

```
-addfilter pubname
  -repsvrfile pubsvrfile
[ -tables schema_t1.table_1 [ schema_t2.table_2 ] ...]
[ -views schema_v1.view_1 [ schema_v2.view_2 ] ...]
[ -tablesfilterclause
    "ordinal_t1:filtername_t1:filterclause_t1"
  [ "ordinal_t2:filtername_t2:filterclause_t2" ] ...]
[ -viewsfilterclause
    "ordinal_v1:filtername_v1:filterclause_v1"
  [ "ordinal_v2:filtername_v2:filterclause_v2" ] ...]
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Section 2.2.12 for additional information on table filters.

Note: The schema names and table or view names that you supply as values for the `tables` or `views` parameters are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example `edb.dept`). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The name of the publication in which table filters are to be added.

pubsvrfile

The file containing the publication server login information.

schema_tn

The name of the schema containing the *n*th table of the `tables` parameter list. This value is case-sensitive.

table_n

The name of the *n*th table in the `tables` parameter list. This value is case-sensitive.

schema_vn

For SMR only: The name of the schema containing the *n*th view of the `views` parameter list. This value is case-sensitive.

view_n

For SMR only: The name of the *n*th view in the `views` parameter list. This value is case-sensitive.

ordinal_tn

The ordinal number (that is, the position in the list counting from left to right starting with 1) of a table in the `tables` parameter list to which an attribute is to be applied.

filtername_tn

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The filter name to be assigned to the filter rule on the table.

filterclause_tn

The filter clause to be applied to the table in the `tables` parameter list at the position indicated by *ordinal_tn*.

ordinal_vn

For SMR only: The ordinal number (that is, the position in the list counting from left to right starting with 1) of a view in the `views` parameter list to which an attribute is to be applied.

filtername_vn

The filter name to be assigned to the filter rule on the view.

filterclause_vn

For SMR only: The filter clause to be applied to the view in the `views` parameter list at the position indicated by *ordinal_vn*.

Examples

In the following example, a table filter is added to table `edb.emp` in publication `analysts_managers`.

```
$ java -jar edb-repcli.jar -addfilter analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -tables edb.emp \
>   -tablesfilterclause "1:jobgrade_9:job = 'SALESMAN'"
Creating Filter(s)
Tables:[[edb.emp, TABLE]]
Filter clause:[FilterName:jobgrade_9    FilterClause:job = 'SALESMAN'   ]
Filter(s) created successfully.
```

8.3.21 Updating Table Filters in a Publication (updatefilter)

The `updatefilter` command changes the filter clauses of the specified tables or views.

Synopsis

```
-updatefilter pubname
  -repsvrfile pubsvrfile
  -tablesfilterclause
    "filterid_1:filterclause_1"
  [ "filterid_2:filterclause_2" ] ...
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The next, subsequent replication to any target subscriptions or non-MDN nodes on which these filter rules had been enabled reflects the changes to the filter clauses.

See Section 2.2.12 for additional information on table filters.

Parameters

pubname

The name of the publication in which the filter clauses are to be updated.

pubsvrfile

The file containing the publication server login information.

filterid_n

Filter ID identifying the filter rule for which the filter clause is to be changed. Use the `printpubfilterslist` command to obtain the filter IDs for the available filter rules in the publication (see Section 8.3.17).

filterclause_n

The new filter clause to be used.

Examples

The filter clause with filter ID `26` in publication `analysts_managers` is modified.

```
$ java -jar edb-repcli.jar -updatefilter analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -tablesfilterclause "26:job = 'CLERK'"
Updating Filter(s)
Filter clause:[26:job = 'CLERK']
Filter(s) updated successfully.
```

8.3.22 Removing a Table Filter from a Publication (removefilter)

The `removefilter` command deletes the table filter from the specified publication.

Synopsis

```
-removefilter pubname
  -repsvrfile pubsvrfile
  -filterid filterid
```

The removed filter rule no longer applies to any target subscriptions or non-MDN nodes on which the filter rule had been enabled.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Section 2.2.12 for additional information on table filters.

Parameters

pubname

The name of the publication containing the filter rule to be removed.

pubsvrfile

The file containing the publication server login information.

filterid

Filter ID identifying the filter rule to be deleted. Use the `printpubfilterslist` command to obtain the filter IDs for the filter rules in the publication (see Section 8.3.17).

Examples

In the following example, the filter rule with filter ID `26` is removed from publication `analysts_managers`.

```
$ java -jar edb-repcli.jar -removefilter analysts_managers \
>   -repsvrfile ~/pubsvrfile.prop \
>   -filterid 26
Removing Filter
Filter removed successfully
```

8.3.23 Printing the Conflict Resolution Strategy (printconfresolutionstrategy)

For MMR only: The `printconfresolutionstrategy` command prints the conflict resolution strategy and the standby conflict resolution strategy of the specified table.

Synopsis

```
-printconfresolutionstrategy pubname
  -repsvrfile pubsvrfile
  -table schema_t.table_name
```

See Section 6.6 for additional information on conflict resolution.

Note: The schema name and table or view name that you supply as values for the `table` parameter are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

edb.dept). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The name of the publication containing the table whose conflict resolution strategy is to be printed.

pubsvrfile

The file containing the publication server login information.

schema_t

The name of the schema containing *table_name*. This value is case-sensitive.

table_name

The name of the table whose conflict resolution strategy is to be printed. This value is case-sensitive.

Examples

In the following example, the conflict resolution strategy on Advanced Server table edb.emp in publication emp_pub is printed.

```
$ java -jar edb-repcli.jar -printconfresolutionstrategy emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -table edb.emp
Primary/Standby Conflict Resolution Strategy...
Conflict Resolution Option:[ Earliest Timestamp ]
Standby Conflict Resolution Option:[ Manual ]
```

8.3.24 Updating the Conflict Resolution Strategy (updateconfresolutionstrategy)

For MMR only: The updateconfresolutionstrategy command changes the conflict resolution strategy or standby conflict resolution strategy of the specified table.

Synopsis

```
-updateconfresolutionstrategy pubname
  -repsvrfile pubsvrfile
  -table schema_t.table_name
  -conflictresolution { E | L | N | M | C }
  -standbyconflictresolution { E | L | N | M | C }
[ -customhandlername customhandler ]
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

See Section 6.8 for additional information on updating the conflict resolution strategy.

Note: The schema name and table or view name that you supply as values for the `table` parameter are case-sensitive. Unless quoted identifiers were used to build the database objects, Oracle names must be entered using uppercase letters (for example, `EDB.DEPT`), and Advanced Server names must be entered in lowercase letters (for example `edb.dept`). See Section 10.4.5 for additional information on quoted identifiers and case translation.

Parameters

pubname

The name of the publication containing the table whose conflict resolution strategy is to be updated.

pubsvrfile

The file containing the publication server login information.

schema_t

The name of the schema containing *table_name*. This value is case-sensitive.

table_name

The name of the table whose conflict resolution strategy is to be updated. This value is case-sensitive.

`-conflictresolution`

For the conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling.

`-standbyconflictresolution`

For the standby conflict resolution option, specify `E` for earliest timestamp conflict resolution, `L` for latest timestamp conflict resolution, `N` for node priority conflict resolution, `M` for manual conflict resolution, or `C` for custom conflict handling.

customhandler

For the custom handler name option, specify *customhandler* as the function name with an optional schema prefix (that is, formatted as

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

schema.function_name) as given in the `CREATE FUNCTION` command for the custom conflict handling function. The custom conflict handling function must be added to the master definition node. See Section 6.6.8.2 for an example of adding the custom conflict handling function using PSQL. The custom handler name option must be specified if and only if the conflict resolution option or the standby conflict resolution option is set for custom conflict handling with the `C` value.

Examples

The conflict resolution strategy on Advanced Server table `edb.emp` in publication `emp_pub` is modified to use latest timestamp conflict resolution with a standby strategy of node priority conflict resolution.

```
$ java -jar edb-repcli.jar -updateconfresolutionstrategy emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -table edb.emp \
>   -conflictresolution L \
>   -standbyconflictresolution N
Updating Primary/Standby Conflict Resolution Strategy...
The Primary/Standby conflict resolution strategies were updated successfully.
```

Custom conflict handling is set for the `edb.dept` table along with the custom conflict handling function `edb.custom_conflict_dept`.

```
$ java -jar edb-repcli.jar -updateconfresolutionstrategy emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -table edb.dept \
>   -conflictresolution C \
>   -standbyconflictresolution N \
>   -customhandlername edb.custom_conflict_dept
Updating Primary/Standby Conflict Resolution Strategy...
The Primary/Standby conflict resolution strategies were updated successfully.
```

8.3.25 Setting the Master Definition Node (setasmdn)

For MMR only: The `setasmdn` command sets a master node to the role of master definition node.

Synopsis

```
-setasmdn pubdbid
  -repsvrfile pubsvrfile
```

See Section 6.10 for additional information on setting the master definition node.

Parameters

pubdbid

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The publication database ID of the master node to be given the role of master definition node.

pubsvrfile

The file containing the publication server login information.

Examples

The following example sets the master node with publication database ID 9 as the master definition node.

```
$ java -jar edb-repcli.jar -setasmdn 9 -repsvrfile ~/pubsvrfile.prop
Updating the database node to be promoted as the new MDN node.
The database has been successfully promoted as the new MDN node.
```

8.3.26 Setting the Controller (setascontroller)

The `setascontroller` command sets a publication database to be designated as the controller database. The publication database may be the master database of a single-master replication system or a master node of a multi-master replication system.

Synopsis

```
-setascontroller pubdbid
  -repsvrfile pubsvrfile
```

See Section 7.7 for additional information on setting the controller database.

Parameters

pubdbid

The publication database ID of the publication database to be designated as the controller database.

pubsvrfile

The file containing the publication server login information.

Examples

The following example sets the publication database with publication database ID 4 as the controller database.

```
$ java -jar edb-repcli.jar -setascontroller 4 -repsvrfile ~/pubsvrfile.prop
Updating the Publication database to be promoted as the new Controller
database...
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
The Publication database has been successfully promoted as the new Controller
database.
```

8.3.27 Validating a Publication (validatepub)

The `validatepub` command checks if any of the definitions of the tables in the given publication have changed since the publication was created.

Synopsis

```
-validatepub pubname
  -repsvrfile pubsvrfile
  -repgrouptype { m | s }
```

See Section 7.6.5 for additional information on validating publications.

Parameters

pubname

The name of the publication whose tables are to be validated.

pubsvrfile

The file containing the publication server login information.

`-repgrouptype`

Specify `s` if this command applies to a single-master replication system. Specify `m` if this command applies to a multi-master replication system.

Examples

In the following example, publication `dept_emp` is validated.

```
$ java -jar edb-repcli.jar -validatepub dept_emp \
>   -repsvrfile ~/pubsvrfile.prop -repgrouptype s
Validating publication dept_emp ...
All schema of published tables in Publication {0} are up-to-date
```

8.3.28 Validating All Publications (validatepubs)

The `validatepubs` command checks if any of the definitions of the tables subordinate to the given publication database definition have changed since the publication was created.

Synopsis

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
-validatepubs
  -repsvrfile pubsvrfile
  -pubdbid dbid
  -repgrouptype { m | s }
```

See Section 7.6.5 for additional information on validating publications.

Parameters

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the publication database definition under which all publication tables are to be validated.

-repgrouptype

Specify `s` if this command applies to a single-master replication system. Specify `m` if this command applies to a multi-master replication system.

Examples

In the following example, the Oracle publication database definition identified by publication database ID `1` is validated.

```
$ java -jar edb-repcli.jar -validatepubs \
>   -repsvrfile ~/pubsvrfile.prop \
>   -pubdbid 1 -repgrouptype s
Validating all available publications ...
The schema definitions for all the non snapshot-only publications tables are
in sync
with the source.
The "validatepubs" feature is not available for the following snapshot-only
publications:
-  salesemp
```

8.3.29 Removing a Publication (removepub)

The `removepub` command removes one or more publications.

Synopsis

```
-removepub pubname_1 [ pubname_2 ] ...
  -repsvrfile pubsvrfile
  -repgrouptype { m | s }
```

See Section 7.6.6 for additional information on removing a publication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Parameters

pubname_n

The name of a publication to be removed.

pubsvrfile

The file containing the publication server login information.

-repgrouptype

Specify s if this command applies to a single-master replication system. Specify m if this command applies to a multi-master replication system.

Examples

A publication named dept_emp is removed from a single-master replication system.

```
$ java -jar edb-repcli.jar -removepub dept_emp \
>   -repsvrfile ~/pubsvrfile.prop -repgrouptype s
Removing publication...
Publication dept_emp unpublished successfully.
```

8.3.30 Replicating DDL Changes (replicateddl)

The replicateddl command applies an ALTER TABLE statement to a publication table in all databases of a replication system as well as updates the xDB Replication Server insert/update/delete triggers and shadow table associated with that publication table.

Synopsis

```
-replicateddl pubname
  -repsvrfile pubsvrfile
  -table schema_t.table_name
  -ddlscriptfile script_file
```

See Section 7.8 for additional information on DDL change replication.

Parameters

pubname

The name of the publication containing the table to which the ALTER TABLE statement is to be applied.

pubsvrfile

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The file containing the publication server login information.

schema_t

The name of the schema containing *table_name*. This value is case-sensitive.

table_name

The name of the table in the ALTER TABLE statement whose definition is to be modified. This value is case-sensitive.

script_file

Path to the file containing the ALTER TABLE statements.

Examples

The following example shows the addition of a column named title to table edb.emp. The ALTER TABLE statement is in the text file addcolumn.sql as shown by the following:

```
ALTER TABLE edb.emp ADD COLUMN title VARCHAR(20);
```

The replicateddl command is executed using the addcolumn.sql file to update the triggers and shadow tables on the master nodes:

```
$ java -jar edb-repcli.jar -replicateddl emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -table edb.emp \
>   -ddlscriptfile ~/addcolumn.sql
DDL changes successfully replicated to all master nodes.
```

8.3.31 Adding a Subscription Database (addsubdb)

For SMR only: The addsubdb command adds a subscription database definition.

Synopsis

```
-addsubdb
  -repsvrfile subsvrfile
  -dbtype { oracle | enterprisedb | postgresql | sqlserver }
  -dbhost host
  -dbport port
  -dbuser user
{ -dbpassword encrypted_pwd | -dbpassfile pwdfile }
[ -oraconnectiontype { sid | servicename } ]
  -database dbname
[ -urloptions jdbc_url_parameters ]
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The `addsubdb` command creates a new subscription database definition. The `addsubdb` command displays a unique subscription database ID that is assigned to the newly created subscription database definition. The subscription database ID is used to identify the subscription database definition on which to operate when running other xDB Replication Server CLI commands.

See Section 5.3.2 for details on the database connection information that must be supplied when adding a subscription database definition.

Parameters

subsvrfile

The file containing the subscription server login information.

-dbtype

Specify `oracle` if the database is an Oracle database. Specify `enterprisedb` if the database is an Advanced Server database in Oracle compatible configuration mode. Specify `postgresql` if the database is a PostgreSQL database or an Advanced Server database in PostgreSQL compatible configuration mode. Specify `sqlserver` if the database is a Microsoft SQL Server database.

host

The IP address of the host on which the subscription database server is running.

port

The port number on which the database server is listening for connections.

user

The subscription database user name.

encrypted_pwd

The encrypted password of the subscription database user. See Section 8.3.4 for directions on using the `encrypt` command to generate an encrypted password.

pwdfile

The file containing the encrypted password of the subscription database user.

-oraconnectiontype

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Specify `sid` if the Oracle system ID (SID) is used to identify the subscription database in the `database` parameter. Specify `servicename` if the Oracle service name is used to identify the subscription database in the `database` parameter. **Note:** For Oracle 12c, use the service name.

dbname

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the subscription database.

jdbc_url_parameters

Extended usage of JDBC URL parameters such as for support of SSL connectivity. (See Section 7.11 for information on SSL connectivity to the subscription database.)

Examples

The following example adds a subscription database definition for an Oracle database. The encrypted password is given on the command line with the `dbpassword` parameter. A subscription database ID of `1` is assigned to the database by the subscription server.

```
$ java -jar edb-repcli.jar -addsubdb -repsvrfile ~/subsvrfile.prop \
>   -dbtype oracle -dbhost 192.168.2.6 -dbport 1521 \
>   -dbuser subuser -dbpassword ygJ9AxoJEX854elcVIJPTw== \
>   -oraconnectiontype sid \
>   -database xe
Adding Subscription Database...
Subscription database added successfully. Subscription Database id:1
```

The following example adds a subscription database definition for an Advanced Server database. The encrypted password is read from a file named `pwdfile` with the `dbpassfile` parameter. A subscription database ID of `2` is assigned to the database by the subscription server.

```
$ java -jar edb-repcli.jar -addsubdb -repsvrfile ~/subsvrfile.prop \
>   -dbtype enterprisedb -dbhost 192.168.2.7 -dbport 5444 \
>   -dbuser subuser -dbpassfile ~/pwdfile \
>   -database subdb
Adding Subscription Database...
Subscription database added successfully. Subscription Database id:2
```

8.3.32 Printing Subscription Database IDs (printsubdbids)

For SMR only: The `printsubdbids` command prints the subscription database IDs of the subscription database definitions.

Synopsis

```
-printsubdbids -repsvrfile subsvrfile
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Parameters

subsvrfile

The file containing the subscription server login information.

Examples

The following example lists the subscription database IDs of the subscription database definitions.

```
$ java -jar edb-repcli.jar -printsubdbids -repsvrfile ~/subsvrfile.prop
Printing subscription database ids...
1
2
```

8.3.33 Printing Subscription Database Details (printsubdbidsdetails)

For SMR only: The `printsubdbidsdetails` command prints the connection information for each subscription database definition.

Synopsis

```
-printsubdbidsdetails -repsvrfile subsvrfile
```

The output has the following components:

```
dbid:host:port:dbname:user
```

Note: The database user's password is not displayed.

Parameters

subsvrfile

The file containing the subscription server login information.

dbid

The subscription database ID assigned to the subscription database definition.

host

The IP address of the host on which the subscription database server is running.

port

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The port number on which the database server is listening for connections.

dbname

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the subscription database.

user

The subscription database user name.

Examples

The following are the subscription database definitions subordinate to the subscription server identified by the content of file `subsvrfile.prop`.

```
$ java -jar edb-repcli.jar -printsubdbidsdetails \
>   -repsvrfile ~/subsvrfile.prop
Printing subscription database ids with details...
id:host:port:database|sid:user
1:192.168.2.6:1521:xe:subuser
2:192.168.2.7:5444:subdb:subuser
```

8.3.34 Updating a Subscription Database (updatesubdb)

For SMR only: The `updatesubdb` command provides the ability to change the connection information for an existing subscription database definition identified by its subscription database ID.

Synopsis

```
-updatesubdb
  -repsvrfile subsvrfile
  -subdbid dbid
  -dbhost host
  -dbport port
  -dbuser user
{ -dbpassword encrypted_pwd | -dbpassfile pwdfile }
[ -oraconnectiontype { sid | servicename } ]
  -database dbname
[ -urloptions jdbc_url_parameters ]
```

The subscription database definition to be updated is identified by the `subdbid` parameter.

See Section 5.3.2 for details on the database connection information that must be supplied for a subscription database definition.

Parameters

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

subsvrfile

The file containing the subscription server login information.

dbid

The subscription database ID of the subscription database definition to be updated.

host

The IP address of the host on which the subscription database server is running.

port

The port number on which the database server is listening for connections.

user

The subscription database user name.

encrypted_pwd

The password of the database user in encrypted form. See Section 8.3.4 for directions on using the `encrypt` command to generate an encrypted password.

pwdfile

The file containing the password of the database user in encrypted form.

`-oraconnectiontype`

Specify `sid` if the Oracle system ID (SID) is used to identify the subscription database in the `database` parameter. Specify `servicename` if the Oracle service name is used to identify the subscription database in the `database` parameter. **Note:** For Oracle 12c, use the service name.

dbname

The Postgres or SQL Server database name, the Oracle SID, or the Oracle service name of the subscription database.

jdbc_url_parameters

Extended usage of JDBC URL parameters such as for support of SSL connectivity. (See Section 7.11 for information on SSL connectivity to the subscription database.) Specification of the `urloptions` parameter completely

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

replaces any existing JDBC URL parameters that may have previously been specified with this database. Omission of the `urloptions` parameter deletes any existing JDBC URL parameters that may have previously been specified with this database.

Examples

In the following example, an existing subscription database definition with subscription database ID `2` is updated.

```
$ java -jar edb-repcli.jar -updatesubdb -repsvrfile ~/subsvrfile.prop \
>   -subdbid 2 \
>   -dbhost 192.168.2.7 -dbport 5444 \
>   -dbuser subuser -dbpassfile ~/pwdfile \
>   -database subdb
Updating subscription database ...
Subscription database with ID 2 is updated successfully.
```

8.3.35 Removing a Subscription Database (removesubdb)

For SMR only: The `removesubdb` command removes a subscription database definition.

Synopsis

```
-removesubdb -repsvrfile subsvrfile -subdbid dbid
```

The subscription database definition to be removed is identified by the `subdbid` parameter.

See Section 5.5.6 for additional information on removing a subscription database.

Parameters

`subsvrfile`

The file containing the subscription server login information.

`dbid`

The subscription database ID of the subscription database definition to be removed.

Examples

The subscription database definition identified by subscription database ID `2` is removed.

```
$ java -jar edb-repcli.jar -removesubdb -repsvrfile ~/subsvrfile.prop \
>   -subdbid 2
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Removing Subscription Database...
Subscription Database removed.
```

8.3.36 Creating a Subscription (createsub)

For SMR only: The `createsub` command creates a new subscription.

Synopsis

```
-createsub subname
  -subsvrfile subsvrfile
  -subdbid dbid
  -pubsvrfile pubsvrfile
  -pubname pubname
[ -filterrule filterid_1[,filterid_2 ] ...]
```

The `createsub` command adds a new subscription subordinate to the subscription database definition with the subscription database ID given by parameter `subdbid`.

See Section 5.3.3 for additional information on creating a subscription.

Parameters

subname

The subscription name to be given to the new subscription.

subsvrfile

The file containing the subscription server login information of the subscription server under which the new subscription is subordinate.

dbid

The subscription database ID of the subscription database definition subordinate to which the new subscription is to be added.

pubsvrfile

The file containing the publication server login information of the publication server under which the publication is subordinate to which the new subscription is to be associated.

pubname

The publication to which the new subscription is to be associated.

filterid_n

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Comma-separated list of filter IDs identifying the filter rules from the set of available table filters to enable on the corresponding tables in the new subscription. Use the `printpubfilterslist` command to obtain the filter IDs for the available filter rules in the publication (see Section 8.3.17). **Note:** There must be no white space between the comma and filter IDs.

Examples

In the following example, a subscription named `dept_emp_sub` is created in the Advanced Server subscription database identified by subscription database ID `2`. The subscription is associated with a publication named `dept_emp`.

```
$ java -jar edb-repcli.jar -createsub dept_emp_sub \
>   -subsvrfile ~/subsvrfile.prop \
>   -subdbid 2 \
>   -pubsvrfile ~/pubsvrfile.prop \
>   -pubname dept_emp
Creating subscription...
Subscription created successfully
```

8.3.37 Printing a Subscription List (printsublist)

For SMR only: The `printsublist` command prints a list of subscription names.

Synopsis

```
-printsublist -repsvrfile subsvrfile -subdbid dbid
```

Parameters

subsvrfile

The file containing the subscription server login information.

dbid

The subscription names subordinate to the subscription database definition specified by *dbid* are printed.

Examples

```
$ java -jar edb-repcli.jar -printsublist -repsvrfile ~/subsvrfile.prop \
>   -subdbid 2
Printing subscriptions ...
dept_emp_sub
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8.3.38 Enabling Filters on a Subscription or Non-MDN Node (enablefilter)

The `enablefilter` command enables one or more filter rules on a single-master replication system subscription or on a multi-master replication system master node other than the master definition node.

The `enablefilter` command is used when it is desired to apply a filter rule to a subscription or a non-MDN node, but the filter rule did not yet exist or it was neglected to be included with the subscription or the non-MDN node when these components were initially created.

Synopsis

```
-enablefilter
  -repsvrfile pubsvrfile
{ -subname subname | -dbid dbid }
  -filterids filterid_1 [ filterid_2 ] ...
```

Enabling a filter rule applies it to the specified, target subscription or non-MDN node, and thus filters the data during replication from the source table to the target table.

See Section 2.2.12 for additional information on table filters.

Before enabling a filter rule, it must have been defined in the source publication in one of several possible ways:

For SMR:

- The table filter was defined in the publication of the master database when it was initially created either by the `createpub` command (see Section 8.3.14) or by the xDB Replication Console (see Section 5.2.3).
- The table filter was added to an existing publication using the `addfilter` command (see Section 8.3.20) or by the xDB Replication Console (see Section 7.6.4).

For MMR:

- The table filter was defined in the publication of the master definition node when it was initially created either by the `createpub` command (see Section 8.3.14) or by the xDB Replication Console (see Section 6.2.3).
- The table filter was added to an existing publication using the `addfilter` command (see Section 8.3.20) or by the xDB Replication Console (see Section 7.6.4).

Enable the filter rule as follows:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For SMR: Use the `enablefilter` command or the xDB Replication Console (see Section 5.5.4).

For MMR: Use the `enablefilter` command or the xDB Replication Console (see Section 6.9).

Once a filter rule has been enabled, it filters the data during replication from the source table to the target table.

A filter rule can subsequently be disabled so that it no longer filters the data during replication to the target table (see Section 8.3.39).

Parameters

pubsvrfile

The file containing the publication server login information.

subname

For SMR only: The name of the subscription containing the tables on which the filter rules are to be enabled.

dbid

For MMR only: The publication database ID of the non-MDN node containing the tables on which the filter rules are to be enabled.

filterid_n

One or more filter IDs separated by space characters identifying the filter rules from the set of available table filters to enable on the corresponding tables in the SMR subscription specified by *subname* or in the MMR non-MDN node specified by *dbid*. Use the `printpubfilterslist` command to obtain the filter IDs for the available filter rules in the publication (see Section 8.3.17).

Examples

In the following example, a filter rule is enabled on a subscription of a single-master replication system.

```
$ java -jar edb-repcli.jar -enablefilter -repsvrfile ~/pubsvrfile.prop \
>   -subname analysts_managers_sub \
>   -filterids 47
Enabling filters...
Filter rule(s) updated successfully.
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In the following example, multiple filter rules are enabled on a master node that is not the master definition node of a multi-master replication system.

```
$ java -jar edb-repcli.jar -enablefilter -repsvrfile ~/pubsvrfile.prop \
>   -dbid 139 \
>   -filterids 8 16
Enabling filters...
Filter rule(s) updated successfully.
```

8.3.39 Disabling Filters on a Subscription or Non-MDN Node (disablefilter)

The `disablefilter` command disables one or more filter rules on a single-master replication system subscription or on a multi-master replication system master node other than the master definition node.

Synopsis

```
-disablefilter
  -repsvrfile pubsvrfile
{ -subname subname | -dbid dbid }
  -filterids filterid_1 [ filterid_2 ] ...
```

Disabling a filter rule prevents it from being applied to the specified, target subscription or non-MDN node, and thus does not filter the data during replication from the source table to the target table.

See Section 2.2.12 for additional information on table filters.

Disable the filter rule as follows:

For SMR: Use the `disablefilter` command or the xDB Replication Console (see Section 5.5.4).

For MMR: Use the `disablefilter` command or the xDB Replication Console. (see Section 6.9).

Disabling a filter rule does not remove its definition from the publication. Thus, the filter rule still exists and can still be enabled on target subscriptions or non-MDN nodes.

To remove a filter rule so that it no longer exists, perform the following:

For either SMR or MMR: Use the `removefilter` command (see Section 8.3.22) or the xDB Replication Console (see Section 7.6.4).

Parameters

pubsvrfile

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The file containing the publication server login information.

subname

For SMR only: The name of the subscription containing the tables on which the filter rules are to be disabled.

dbid

For MMR only: The publication database ID of the non-MDN node containing the tables on which the filter rules are to be disabled.

filterid_n

One or more filter IDs separated by space characters identifying the currently enabled table filters that are to be disabled in the SMR subscription specified by *subname* or in the MMR non-MDN node specified by *dbid*.

Examples

In the following example, a filter rule is disabled on a subscription of a single-master replication system.

```
$ java -jar edb-repcli.jar -disablefilter -repsvrfile ~/pubsvrfile.prop \
>   -subname analysts_managers_sub \
>   -filterids 47
Disabling filters...
Filter rule(s) updated successfully.
```

In the following example, multiple filter rules are disabled on a master node that is not the master definition node of a multi-master replication system.

```
$ java -jar edb-repcli.jar -disablefilter -repsvrfile ~/pubsvrfile.prop \
>   -dbid 139 \
>   -filterids 8 16
Disabling filters...
Filter rule(s) updated successfully.
```

8.3.40 Taking a Single-Master Snapshot (dosnapshot)

For SMR only: The `dosnapshot` command performs snapshot synchronization on the specified subscription in a single-master replication system.

Synopsis

```
-dosnapshot subname -repsvrfile subsvrfile
[ -verboseSnapshotOutput { true | false } ]
```

See Section 5.4.1 for additional information on performing snapshot replication.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Parameters

subname

The name of the subscription for which the snapshot is to be taken.

subsvrfile

The file containing the subscription server login information.

-verboseSnapshotOutput

Set this option to `true` if you want the output from the snapshot to be displayed. Set this option to `false` if you do not want the snapshot output displayed. If omitted, the default is `true`.

Examples

In the following example snapshot replication is performed on subscription `dept_emp_sub`.

```
$ java -jar edb-repcli.jar -dosnapshot dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop
Performing snapshot...
Source database connectivity info...
conn =jdbc:oracle:thin:@192.168.2.6:1521:xe
user =pubuser
password=******
Target database connectivity info...
conn =jdbc:edb://192.168.2.7:5444/subdb
user =subuser
password=******
Connecting with source Oracle database server...
Connecting with target EnterpriseDB database server...
Importing redwood schema EDB...
Table List: 'DEPT','EMP'
Loading Table Data in 8 MB batches...
Disabling FK constraints & triggers on edb.dept before truncate...
Truncating table DEPT before data load...
Disabling indexes on edb.dept before data load...
Loading Table: DEPT ...
[DEPT] Migrated 4 rows.
[DEPT] Table Data Load Summary: Total Time(s): 0.182 Total Rows: 4
Disabling FK constraints & triggers on edb.emp before truncate...
Truncating table EMP before data load...
Disabling indexes on edb.emp before data load...
Loading Table: EMP ...
[EMP] Migrated 14 rows.
[EMP] Table Data Load Summary: Total Time(s): 0.178 Total Rows: 14
Enabling FK constraints & triggers on edb.dept...
Enabling indexes on edb.dept after data load...
Enabling FK constraints & triggers on edb.emp...
Enabling indexes on edb.emp after data load...
Performing ANALYZE on EnterpriseDB database...
Data Load Summary: Total Time (sec): 1.866 Total Rows: 18 Total Size(MB): 0.0
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Schema EDB imported successfully.


Migration process completed successfully.

Migration logs have been saved to /var/log/xdb-rep/build571

******************** Migration Summary ********************
Tables: 2 out of 2
Constraints: 4 out of 4

Total objects: 6
Successful count: 6
Failure count: 0

************************************************************
Snapshot taken successfully.
```

8.3.41 Take a Multi-Master Snapshot (dommrsnapshot)

For MMR only: The `dommrsnapshot` command performs snapshot synchronization on the specified master node in a multi-master replication system.

```
-dommrsnapshot pubname
  -repsvrfile pubsvrfile
  -pubhostdbid dbid
[ -verboseSnapshotOutput { true | false } ]
```

Parameters

pubname

The name of the publication for which the snapshot is to be taken.

pubsvrfile

The file containing the publication server login information.

dbid

The publication database ID of the target master node for the snapshot replication.

`-verboseSnapshotOutput`

Set this option to `true` if you want the output from the snapshot to be displayed. Set this option to `false` if you do not want the snapshot output displayed. If omitted, the default is `true`.

Examples

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In the following example snapshot replication is performed on publication `emp_pub` to the target master node identified by publication database ID `9`.

```
$ java -jar edb-repcli.jar -dommrsnapshot emp_pub \
>   -pubhostdbid 9 \
>   -repsvrfile ~/pubsvrfile.prop
Performing snapshot...
Source database connectivity info...
conn =jdbc:edb://192.168.2.6:5444/edb
user =pubuser
password=******
Target database connectivity info...
conn =jdbc:edb://192.168.2.7:5444/mmrnode
user =mmruser
password=******
Connecting with source EnterpriseDB database server...
Connecting with target EnterpriseDB database server...
Importing enterprisedb schema edb...
Table List: 'dept','emp'
Loading Table Data in 8 MB batches...
Disabling FK constraints & triggers on edb.dept before truncate...
Truncating table dept before data load...
Disabling indexes on edb.dept before data load...
Loading Table: dept ...
[dept] Migrated 5 rows.
[dept] Table Data Load Summary: Total Time(s): 0.247 Total Rows: 5
Disabling FK constraints & triggers on edb.emp before truncate...
Truncating table emp before data load...
Disabling indexes on edb.emp before data load...
Loading Table: emp ...
[emp] Migrated 14 rows.
[emp] Table Data Load Summary: Total Time(s): 0.163 Total Rows: 14
Enabling FK constraints & triggers on edb.dept...
Enabling indexes on edb.dept after data load...
Enabling FK constraints & triggers on edb.emp...
Enabling indexes on edb.emp after data load...
Performing ANALYZE on EnterpriseDB database...
Data Load Summary: Total Time (sec): 0.8 Total Rows: 19 Total Size(MB): 0.0

Schema edb imported successfully.


Migration process completed successfully.

Migration logs have been saved to /var/log/xdb-rep/build571

******************** Migration Summary ********************
Tables: 2 out of 2
Constraints: 4 out of 4

Total objects: 6
Successful count: 6
Failure count: 0

************************************************************
Snapshot taken successfully.
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

8.3.42 Performing a Synchronization (dosynchronize)

The `dosynchronize` command performs synchronization replication on the specified subscription for a single-master replication system, or for an entire multi-master replication system.

Synopsis

```
-dosynchronize { subname | pubname }
  -repsvrfile { subsvrfile | pubsvrfile }
[ -repgrouptype { s | m } ]
```

For a single-master replication system use:

```
-dosynchronize subname -repsvrfile subsvrfile
```

For a multi-master replication system use:

```
-dosynchronize pubname -repsvrfile pubsvrfile -repgrouptype m
```

Note (For SMR only): The `dosynchronize` command can be used on a subscription without first having to perform a snapshot using the `dosnapshot` command. The `dosynchronize` command automatically performs the first required snapshot.

Note (For MMR only): Be sure an initial snapshot replication has been performed from the master definition node to every other master node in the multi-master replication system. If a newly added master node did not undergo an initial snapshot, any subsequent synchronization replication may fail to apply the transactions to that master node. The initial snapshot could be taken when the master node is first added (see Section 6.3 or Section 8.3.6) or by performing an on demand snapshot (see Section 6.5.1 or Section 8.3.41).

See Section 5.4.2 for additional information on performing synchronization replication for a single-master replication system. See Section 6.5.2 for a multi-master replication system.

Parameters

subname

> **For SMR only:** The name of the subscription for which synchronization replication is to be performed.

pubname

> **For MMR only:** The name of the publication for which synchronization replication is to be performed.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

subsvrfile

For SMR only: The file containing the subscription server login information.

pubsvrfile

For MMR only: The file containing the publication server login information.

-repgrouptype

Specify s if this command applies to a single-master replication system. Specify m if this command applies to a multi-master replication system. If omitted, the default is s.

Examples

In the following example, synchronization replication is performed on subscription dept_emp_sub of a single-master replication system.

```
$ java -jar edb-repcli.jar -dosynchronize dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop
Performing synchronize...
Synchronize done successfully.
```

In the following example, synchronization replication is performed on publication emp_pub of a multi-master replication system. Note that the -repgrouptype m parameter is required in this case.

```
$ java -jar edb-repcli.jar -dosynchronize emp_pub \
>   -repsvrfile ~/pubsvrfile.prop -repgrouptype m
Performing synchronize...
Publication synchronized successfully.
```

8.3.43 Configuring a Single-Master Schedule (confschedule)

For SMR only: The confschedule command creates a schedule as to when recurring replications are to be initiated for a single-master replication system.

Synopsis

```
-confschedule subname -repsvrfile subsvrfile
{ -remove | -jobtype { s | t }
  { -realtime no_of_sec |
    -daily hour minute |
    -weekly day_of_week hour minute |
    -monthly month day_of_month hour minute |
    -cronexpr "cron_expression"
  }
}
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If the `remove` parameter is specified, then the schedule is deleted from the subscription. No other parameters other than *`subname`* and `repsvrfile` can be specified in this case.

If the `remove` parameter is omitted, then the `jobtype` parameter and one of parameters `realtime`, `daily`, `weekly`, `monthly`, or `cronexpr` must be specified along with the *`subname`* and `repsvrfile` parameters. If there is an existing schedule for subscription *`subname`*, it will be replaced by the new schedule.

See Section 7.2 for additional information on creating a schedule.

Parameters

`subname`

The name of the subscription for which a replication schedule is to be created.

`subsvrfile`

The file containing the subscription server login information.

`-remove`

If the `remove` parameter is specified, then any existing schedule is removed from the subscription. If the `remove` parameter is not specified, then a schedule is created for the subscription.

`-jobtype`

Specify `s` if the scheduled replication is to be done by snapshot. Specify `t` if the scheduled replication is to be done by synchronization. If the associated publication is a snapshot-only publication, then `-jobtype s` must be used.

`no_of_sec`

The number of seconds between scheduled replications. This can be any integer greater than `0`.

`hour`

The hour of the day based on a 24-hour clock. This can be any integer from `0` to `23`.

`minute`

The minute of the hour. This can be any integer from `0` to `59`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

day_of_week

The day of the week. This can be any of the following values: SUN, MON, TUE, WED, THU, FRI, or SAT. This value is case insensitive so Sun and sun will work as well as SUN.

month

The month of the year. This can be any of the following values: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC. This value is case insensitive so Jan and jan will work as well as JAN.

day_of_month

The day of the month. This can be any integer greater than or equal to 1, and less than or equal to the number of days in *month*.

cron_expression

A cron expression. See appendix Section 10.4.3 for information on writing a cron expression.

Examples

In the following example, a schedule is created to perform synchronization replication on subscription dept_emp_sub once every 5 minutes.

```
$ java -jar edb-repcli.jar -confschedule dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop \
>   -jobtype t \
>   -realtime 300
Configuring scheduler ...
Job is successfully scheduled.
```

In the following example, the schedule is removed from subscription dept_emp_sub.

```
$ java -jar edb-repcli.jar -confschedule dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop \
>   -remove
Configuring scheduler ...
Scheduled job is removed.
```

8.3.44 Configuring a Multi-Master Schedule (confschedulemmr)

For MMR only: The confschedulemmr command creates a schedule as to when recurring replications are to be initiated for a multi-master replication system.

Note: Be sure an initial snapshot replication has been performed from the master definition node to every other master node in the multi-master replication system. If a

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

newly added master node did not undergo an initial snapshot, any subsequent synchronization replication initiated by a schedule may fail to apply the transactions to that master node. The initial snapshot could be taken when the master node is first added (see Section 6.3 or Section 8.3.6) or by performing an on demand snapshot (see Section 6.5.1 or Section 8.3.41).

Synopsis

```
-confschedulemmr pubdbid -pubname pubname
  -repsvrfile pubsvrfile
{ -remove |
  { -realtime no_of_sec |
    -daily hour minute |
    -weekly day_of_week hour minute |
    -monthly month day_of_month hour minute |
    -cronexpr "cron_expression"
  }
}
```

If the `remove` parameter is specified, then the schedule is deleted from the publication. No other parameters other than *`pubdbid`*, `pubname`, and `repsvrfile` can be specified in this case.

If the `remove` parameter is omitted, then one of parameters `realtime`, `daily`, `weekly`, `monthly`, or `cronexpr` must be specified along with the *`pubdbid`*, `pubname`, and `repsvrfile` parameters. If there is an existing schedule for publication *`pubname`*, it will be replaced by the new schedule.

See Section 7.2 for additional information on creating a schedule.

Parameters

`pubdbid`

The publication database ID of the publication database definition representing the master definition node on which to configure the schedule.

`pubname`

The name of the publication for which a replication schedule is to be created.

`pubsvrfile`

The file containing the publication server login information.

`-remove`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If the `remove` parameter is specified, then any existing schedule is removed from the publication. If the `remove` parameter is not specified, then a schedule is created for the publication.

no_of_sec

The number of seconds between scheduled replications. This can be any integer greater than `0`.

hour

The hour of the day based on a 24-hour clock. This can be any integer from `0` to `23`.

minute

The minute of the hour. This can be any integer from `0` to `59`.

day_of_week

The day of the week. This can be any of the following values: `SUN`, `MON`, `TUE`, `WED`, `THU`, `FRI`, or `SAT`. This value is case insensitive so `Sun` and `sun` will work as well as `SUN`.

month

The month of the year. This can be any of the following values: `JAN`, `FEB`, `MAR`, `APR`, `MAY`, `JUN`, `JUL`, `AUG`, `SEP`, `OCT`, `NOV`, or `DEC`. This value is case insensitive so `Jan` and `jan` will work as well as `JAN`.

day_of_month

The day of the month. This can be any integer greater than or equal to `1`, and less than or equal to the number of days in *month*.

cron_expression

A cron expression. See appendix Section 10.4.3 for information on writing a cron expression.

Examples

In the following example, a schedule is created to perform synchronization replication on publication `emp_pub` subordinate to the master definition node whose publication database ID is `6`. Replication is to occur daily at 8:00 AM.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
$ java -jar edb-repcli.jar -confschedulemmr 6 -pubname emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -daily 8 00
Configuring scheduler ...
Job is successfully scheduled.
```

In the following example, the schedule is removed from publication `emp_pub`.

```
$ java -jar edb-repcli.jar -confschedulemmr 6 -pubname emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -remove
Configuring scheduler ...
Scheduled job is removed.
```

8.3.45 Print Schedule (printschedule)

The `printschedule` command prints a recurring replication schedule.

Synopsis

```
-printschedule { subname | pubname }
  -repsvrfile { subsvrfile | pubsvrfile }
[ -repgrouptype { s | m } ]
```

For a single-master replication system use:

```
-printschedule subname -repsvrfile subsvrfile
```

For a multi-master replication system use:

```
-printschedule pubname -repsvrfile pubsvrfile -repgrouptype m
```

Parameters

subname

> **For SMR only:** The name of the subscription for which the schedule is to be printed.

pubname

> **For MMR only:** The name of the publication for which the schedule is to be printed.

subsvrfile

> **For SMR only:** The file containing the subscription server login information.

pubsvrfile

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For MMR only: The file containing the publication server login information.

`-repgrouptype`

Specify `s` if this command applies to a single-master replication system. Specify `m` if this command applies to a multi-master replication system. If omitted, the default is `s`.

Examples

In the following example the schedule is printed for a subscription in a single-master replication system.

```
$ java -jar edb-repcli.jar -printschedule dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop
Printing subscription schedule ...

Job type                Synchronize

Scheduled time          2012-06-19 13:27:20

Previous fire time      2012-06-19 13:27:20

Next fire time          2012-06-19 13:32:20
```

In the following example the schedule is printed for a publication in a multi-master replication system. Note that the `-repgrouptype m` parameter is required in this case.

```
$ java -jar edb-repcli.jar -printschedule emp_pub \
>   -repsvrfile ~/pubsvrfile.prop \
>   -repgrouptype m
Printing subscription schedule ...

Job type                Synchronize

Scheduled time          2012-06-19 13:27:55

Previous fire time      Not available

Next fire time          2012-06-20 08:00:00

Cron expression         0 0 8 * * ?
```

8.3.46 Updating a Subscription (updatesub)

For SMR only: The `updatesub` command allows you to update certain metadata of a given subscription. This metadata allows the subscription server to find the host running the publication server that manages the publication associated with the subscription.

Synopsis

```
-updatesub subname
  -subsvrfile subsvrfile
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
-pubsvrfile pubsvrfile
-host newpubsvr_ipaddress
-port newpubsvr_port
```

The `updatesub` command allows you to update the subscription metadata consisting of the IP address and port number identifying the publication server that is the parent of the publication associated with the subscription.

This metadata is essential to the proper operation of the replication system since it is the means by which subscription server locates the publication server whenever a replication needs to be performed on a given subscription. The replication process is always carried out by the publication server that manages the publication associated with the subscription.

You would use the `updatesub` command in the scenario where you have built your replication system using IP addresses that are valid at that point in time. At some later point, the IP address assigned to the host running the publication server has changed.

You use the `host` and `port` parameters of the `updatesub` command to supply the new network address identifying the publication server.

See Section 5.5.3 for additional information on updating a subscription.

Parameters

subname

The name of the subscription whose metadata is to be updated.

subsvrfile

The file containing the subscription server login information for the subscription server in which subscription *subname* was created.

pubsvrfile

The file containing publication server login information for the publication server that manages the publication associated with subscription *subname*. Note that the values that you supply for *newpubsvr_ipaddress* and *newpubsvc_port* must be the same as the values set in fields `host` and `port` in file *pubsvrfile*.

newpubsvr_ipaddress

The new IP address for the publication server that manages the publication associated with subscription *subname*. This value must be the same as the IP address specified for field `host` in file *pubsvrfile*.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

newpubsvr_port

The new port number for the publication server that manages the publication associated with subscription *subname*. This value must be the same as the port number specified for field `port` in file *pubsvrfile*.

Examples

If the publication server host IP address has been changed to `192.168.2.7`, then make sure the publication server login information in file `pubsvrfile.prop` contains the new IP address as shown by the following:

```
host=192.168.2.7
port=9051
user=enterprisedb
# Password is in encrypted form.
password=ygJ9AxoJEX854elcVIJPTw==
```

To update the metadata for subscription `dept_emp_sub` so that its subscription server can find the new publication server host, run the following command:

```
$ java -jar edb-repcli.jar -updatesub dept_emp_sub \
>   -subsvrfile ~/subsvrfile.prop \
>   -pubsvrfile ~/pubsvrfile.prop \
>   -host 192.168.2.7 \
>   -port 9051
Updating subscription dept_emp_sub...

Subscription is updated successfully
```

8.3.47 Removing a Subscription (removesub)

For SMR only: The `removesub` command removes a subscription.

Synopsis

```
-removesub subname -repsvrfile subsvrfile
```

See Section 5.5.5 for additional information on removing a subscription.

Parameters

subname

The name of the subscription to be removed.

subsvrfile

The file containing the subscription server login information.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Examples

A subscription named `dept_emp_sub` is removed.

```
$ java -jar edb-repcli.jar -removesub dept_emp_sub \
>   -repsvrfile ~/subsvrfile.prop
Removing subscription...
Subscription removed successfully.
```

8.3.48 Scheduling Shadow Table History Cleanup (confcleanupjob)

The `confcleanupjob` command creates a schedule as to when shadow table history is to be deleted.

Synopsis

```
-confcleanupjob pubdbid -repsvrfile pubsvrfile
{ -disable | -enable
  { -minutely no_of_minutes |
    -hourly no_of_hours |
    -daily hour |
    -weekly day_of_week hour |
    -cronexpr "cron_expression"
  }
}
```

If the `disable` parameter is specified, then the schedule is deleted. No other parameters other than *`pubdbid`* and `pubsvrfile` can be specified in this case.

If the `disable` parameter is omitted, then the `enable` parameter and one of parameters `minutely`, `hourly`, `daily`, `weekly`, or `cronexpr` must be specified along with the *`pubdbid`* and `pubsvrfile` parameters.

See Section 7.5.1 for additional information on creating a schedule for shadow table history cleanup.

Parameters

`pubdbid`

Publication database ID of the publication database definition for which a schedule is to be enabled or disabled for deleting shadow table history.

`pubsvrfile`

The file containing the publication server login information.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

`-disable`

If the `disable` parameter is specified, then any existing shadow table history cleanup schedule is removed from the publication database definition. If the `disable` parameter is not specified, then `enable` must be specified.

`-enable`

Establish a schedule for shadow table history cleanup.

`no_of_minutes`

The number of minutes between scheduled shadow table history cleanup jobs. This can be any integer between `1` and `59` inclusive.

`no_of_hours`

The number of hours between scheduled shadow table history cleanup jobs. This can be any integer between `1` and `12` inclusive.

`hour`

The hour of the day based on a 24-hour clock. This can be any integer from `0` to `23`.

`day_of_week`

The day of the week. This can be any of the following values: `SUNDAY`, `MONDAY`, `TUESDAY`, `WEDNESDAY`, `THURSDAY`, `FRIDAY`, or `SATURDAY`. This value is case insensitive so `Sunday` and `sunday` will work as well as `SUNDAY`.

`cron_expression`

A cron expression. See appendix Section 10.4.3 for information on writing a cron expression.

Examples

In the following example shadow table history cleanup is scheduled to run once every 3 hours on the shadow tables created within the publication database definition identified by publication database ID `1`.

```
$ java -jar edb-repcli.jar -confcleanupjob 1 \
>   -repsvrfile ~/pubsvrfile.prop \
>   -enable -hourly 3
Configuring cleanup job ...
Cleanup job configured.
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In the following example shadow table history cleanup is scheduled to run once a day at 6:00 PM on the shadow tables created within the publication database definition identified by publication database ID `1`.

```
$ java -jar edb-repcli.jar -confcleanupjob 1 \
>   -repsvrfile ~/pubsvrfile.prop \
>   -enable -daily 18
Configuring cleanup job ...
Cleanup job configured.
```

In the following example shadow table history cleanup is scheduled to run every Wednesday at 8:00 AM on the shadow tables created within the publication database definition identified by publication database ID `1`.

```
$ java -jar edb-repcli.jar -confcleanupjob 1 \
>   -repsvrfile ~/pubsvrfile.prop \
>   -enable -weekly WEDNESDAY 8
Configuring cleanup job ...
Cleanup job configured.
```

In the following example the shadow table history cleanup job is disabled on the publication database definition identified by publication database ID `1`.

```
$ java -jar edb-repcli.jar -confcleanupjob 1 \
>   -repsvrfile ~/pubsvrfile.prop -disable
Configuring cleanup job ...
Cleanup job removed.
```

8.3.49 Cleaning Up Shadow Table History (cleanshadowhistforpub)

The `cleanshadowhistforpub` command deletes the shadow table history for the specified publication.

Synopsis

```
-cleanshadowhistforpub pubname
  -repsvrfile pubsvrfile
[ -mmrdbid dbid_1[,dbid_2 ] ...]
```

See Section 7.5.2 for additional information on cleaning up shadow table history.

Parameters

pubname

The name of the publication for which the shadow table history is to be deleted.

pubsvrfile

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The file containing the publication server login information.

dbid_n

For MMR only: The publication database ID of the master node for which the shadow table history is to be deleted. This parameter is required for a multi-master replication system specifying one or more comma-separated, publication database IDs. **Note:** There must be no white space between the comma and publication database IDs.

Examples

In the following example shadow table history is deleted for publication `dept_emp`.

```
$ java -jar edb-repcli.jar -cleanshadowhistforpub dept_emp \
>   -repsvrfile ~/pubsvrfile.prop
Removing shadow table's transaction history ...

Shadow table's transaction history removed successfully.
```

8.3.50 Cleaning Up Replication History (cleanrephistoryforpub)

The `cleanrephistoryforpub` command deletes the replication history for the specified publication.

Synopsis

```
-cleanrephistoryforpub pubname -repsvrfile pubsvrfile
```

See Section 7.5.3 for additional information on cleaning up replication history.

Parameters

pubname

The name of the publication for which replication history is to be deleted.

pubsvrfile

The file containing the publication server login information.

Examples

In the following example replication history is deleted for publication `dept_emp`.

```
$ java -jar edb-repcli.jar -cleanrephistoryforpub dept_emp \
>   -repsvrfile ~/pubsvrfile.prop
Removing publication's replication history ...
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Replication history has been removed.
```

8.3.51 Cleaning Up All Replication History (cleanrephistory)

The `cleanrephistory` command deletes the replication history for all publications in the specified publication server.

Synopsis

```
-cleanrephistory –repsvrfile pubsvrfile
```

See Section 7.5.3 for additional information on cleaning up replication history.

Parameters

pubsvrfile

The file containing the publication server login information.

Examples

In the following example, replication history is deleted for all publications in the publication server identified by the content of file `pubsvrfile.prop`.

```
$ java -jar edb-repcli.jar -cleanrephistory -repsvrfile ~/pubsvrfile.prop
Removing all publication's replication history ...

Replication history has been removed.
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

9 Data Validator

The Data Validator is a utility that compares the rows of one or more tables within a schema of a database against the rows of the tables with the same names within a schema of another database. The Data Validator generates a summary of the comparison noting the number of rows whose column values differ. A file containing detailed information regarding any differences is also generated.

The two databases being compared are referred to as the *source database* and the *target database*. The source database can be of type Oracle, EnterpriseDB, SQL Server, Sybase®, or MySQL®. The target database must be either Oracle or EnterpriseDB.

An EnterpriseDB database type means either an Advanced Server database or a PostgreSQL database.

The tables available for comparison are those found in the schema of the source database. Tables in the target database that do not exist in the source database schema are ignored.

Note: The Data Validator does not validate columns having the following data types. Tables containing one or more columns of these types will only be partially validated.

- `BFILE`
- `STRUCT`
- `REF`
- `ARRAY`
- `BLOB`
- `CLOB`
- `RAW`
- `LONG RAW`

Note: Regarding the usage of the Data Validator with tables in an xDB Replication Server single-master or multi-master replication system, be sure all synchronization replication between the source and target xDB Replication Server tables has been completed before using the Data Validator. If synchronization replication is still in progress, it is probable that the Data Validator will report differences in table content.

9.1 Installation and Configuration

Step 1: When you install the xDB Replication Server product, the components for the Data Validator are installed as well. See Chapter 3 for information on installing the xDB Replication Server product.

Also, when you uninstall the xDB Replication Server product, the Data Validator components are uninstalled as well.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following components that you use to run the Data Validator are installed when you install the xDB Replication Server product.

Table 9-1 - Data Validator Files

File Name	Location	Description
`datavalidator.properties`	*XDB_HOME*`/etc`	Data Validator Properties file
`runValidation.sh` (Linux)	*XDB_HOME*`/bin`	Data Validator execution script
`runValidation.bat` (Windows)	*XDB_HOME*`\bin`	Data Validator execution script

Note: *XDB_HOME* is the directory where xDB Replication Server is installed. This may or may not be the same as the Postgres home directory depending upon how xDB Replication Server is installed.

Step 2: If you plan to use an Oracle database as the source or target database, download the Oracle JDBC driver and place it in the *JAVA_HOME*`/jre/lib/ext` directory.

Step 3: Edit the `datavalidator.properties` file located in the *XDB_HOME*`/etc` directory and specify the connection information for the source and target databases you want to compare.

Any of these parameters can be overridden by an option when you invoke the Data Validator script. See Section 9.2 for additional information on invoking the Data Validator.

The following are the parameters in the `datavalidator.properties` file.

Table 9-2 - Data Validator Properties File Parameters

Parameter	Description
`source_dbms`	Type of the source database. Values may be `enterprisedb`, `oracle`, `sqlserver`, `sybase`, or `mysql`.
`source_host`	IP address or server name of the host running the database server of the source database
`source_port`	Port number on which the database server of the source database listens for requests
`source_database`	Database name of the source database
`source_user`	Database user name of the source database
`source_password`	Unencrypted password of the source database user
`target_dbms`	Type of the target database. Values may be `enterprisedb` or `oracle`.
`target_host`	IP address or server name of the host running the database server of the target database
`target_port`	Port number on which the database server of the target database listens for requests
`target_database`	Database name of the target database
`target_user`	Database user name of the target database
`target_password`	Unencrypted password of the target database user

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following is the initial content of the `datavalidator.properties` file after installation:

```
############################################################
#               Source database connection                 #
############################################################

#
#source_dbms=(enterprisedb | oracle | sqlserver | sybase | mysql)
#

source_dbms=oracle
source_host=localhost
source_port=1521
source_database=xe
source_user=hr
source_password=hr


#source_dbms=mysql
#source_host=localhost
#source_port=3306
#source_database=test
#source_user=root
#source_password=

#source_dbms=sqlserver
#source_host=localhost
#source_port=1433
#source_database=pubs
#source_user=sa
#source_password=

#source_dbms=sybase
#source_host=localhost
#source_port=5004
#source_database=test
#source_user=sa
#source_password=

############################################################
#               Target database connection                 #
############################################################

#
#target_dbms=(enterprisedb | oracle)
#
target_dbms=enterprisedb
target_host=localhost
target_port=5444
target_database=edb
target_user=enterprisedb
target_password=edb
```

Step 4: Determine the location for the Data Validator logs directory.

Before invoking the Data Validator for the first time, be sure you have determined where the Data Validator logs directory is to be located.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The Data Validator generates a log file with a name formatted as `datavalidator_yymmdd-hhmiss.log` in the logs directory for each run.

If there are row differences between the source and target tables, a file with a name formatted as `datavalidator_yymmdd-hhmiss.diff` is also generated that contains output of the errors in *diff* format. Use a graphical diff tool like Kompare to view this file to highlight the specific differences.

The Data Validator attempts to create a subdirectory named `logs` within the *XDB_HOME*`/bin` directory the first time you invoke the Data Validator without the `-ld` option. If you do not invoke the Data Validator as the `root` account, it is likely that the run will fail as it attempts to create subdirectory `logs` in the *XDB_HOME*`/bin` directory where typically only the `root` account has this privilege.

The same situation also exists on Windows hosts as the account you are using must have the permission to create a subdirectory in the *XDB_HOME*`\bin` location.

Choices for determining and setting the Data Validator directory for the log and diff files are the following:

- Run the Data Validator as the `root` account. This enables the Data Validator to create the `logs` subdirectory within the *XDB_HOME*`/bin` directory, and then to create the log and diff files in the `logs` subdirectory.
- Create the *XDB_HOME*`/bin/logs` directory structure before running the Data Validator. Modify the permissions on directory *XDB_HOME*`/bin/logs` so the operating system account you use to run the Data Validator has the privilege to create files in the directory.
- Use the `-ld` *log_directory_path* option to allow the Data Validator to create the log and diff files in the specified directory location *log_directory_path*. Be sure the operating system account you use to run the Data Validator has the proper privileges to either create the lowest level subdirectory specified by *log_directory_path* if it does not already exist, or to create files within the specified directory if the full directory path already does exist.

Once you have determined and verified that your operating system account you plan to use to run the Data Validator can create files in the log directory, you can proceed with performing data validation.

9.2 Performing Data Validation

The current working directory from which you invoke the Data Validator script `runValidation.sh` (`runValidation.bat` for Windows) must be the `bin` subdirectory containing the script (that is, *XDB_HOME*`/bin`).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For example, if the xDB Replication Server is installed into its default directory location, then issue the following command before invoking the Data Validator:

```
cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
```

Similarly for Windows hosts, issue the following:

```
cd C:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer\bin
```

The general command format for invoking the Data Validator is the following:

```
./runValidation.sh { -ss | --source-schema } schema_name
  [ option ] ...
```

schema_name is the name of the schema in the source database containing the tables to be validated. The choices for *option* are listed later in this section within the Options subsection.

For Windows hosts, the command format is the following:

```
runValidation { -ss | --source-schema } schema_name
  [ option ] ...
```

The following option displays the Data Validator version:

```
./runValidation.sh { -v | --version }
```

On Linux the version is displayed as follows:

```
$ ./runValidation.sh --version
EnterpriseDB DataValidator Build 3
```

On Windows the version is displayed as follows:

```
C:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer\bin>runValidation -v
EnterpriseDB DataValidator Build 3
```

The following option displays the help information.

```
./runValidation.sh { -h | --help }
```

This is shown by the following:

```
$ ./runValidation.sh --help
Usage:
     runValidation.sh (-v  | --version) | (-h | --help)
     runValidation.sh (-ss | --source-schema) SOURCE_SCHEMA [OPTIONS]
CONNECTION_INFO_FILE

OPTIONS:
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
(-ts | --target-schema)   target-schema-name
(-it | --include-tables)  comma-seperated-tables-name
(-et | --exclude-tables)  comma-seperated-tables-name

(-ld | --logging-dir)     logging-dir-path
(-ds  | --display-summary) (true|false)
(-srs | --skip-rowsonlyin-source) (true|false)
(-srt | --skip-rowsonlyin-target) (true|false)
(-srb | --skip-rowsin-both) (true|false)
(-fs  | --fetch-size) row count
(-bs  | --batch-size) row count

(-sdbms | --source-dbms)     source database type
(-sh | --source-host) source database server name/IP
(-sp | --source-port) source database server port
(-sdb | --source-database)     source database name
(-su | --source-user) source database user id
(-spw | --source-password) source database user password
(-tdbms | --target-dbms)     target database type
(-th | --target-host) target database server name/IP
(-tp | --target-port) target database server port
(-tdb | --target-database)     target database name
(-tu | --target-user) target database user id
(-tpw | --target-password) target database user password
```

The general syntax for all options except for `--version` and `--help` is shown by the following:

```
./runValidation.sh -ss schema
  [ -ts schema ]
  [ -it table_1 [,table_2 ] ... ]
  [ -et table_1 [,table_2 ] ... ]
  [ -srs { true | false } ]
  [ -srt { true | false } ]
  [ -srb { true | false } ]
  [ -ld log_directory_path ]
  [ -ds { true | false } ]
  [ -sdbms database_type ]
  [ -sh host ]
  [ -sp port ]
  [ -sdb dbname ]
  [ -su user ]
  [ -spw password ]
  [ -tdbms database_type ]
  [ -th host ]
  [ -tp port ]
  [ -tdb dbname ]
  [ -tu user ]
  [ -tpw password ]
  [ -bs row_count ]
  [ -fs row_count ]
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For clarity, the preceding syntax diagram shows only the single-character form of the option. The **Options** subsection lists both the single-character and multi-character forms of the options.

Specification of any database connection option (`-sdbms` through `-tpw` listed in the preceding syntax diagram) overrides the corresponding parameter in the `datavalidator.properties` file. See Section 9.1 for information on the `datavalidator.properties` file.

Options

`-ss, --source-schema` *schema*

The schema of the source database containing the tables to be compared against the target database.

`-ts, --target-schema` *schema*

The schema of the target database containing the tables to be compared against the source database. If omitted, the schema of the target database is the same schema as specified for the source database with the `-ss` option.

`-it, --include-tables` *table_1* `[,`*table_2* `] ...`

The tables within the source schema that are to be included for comparison. If omitted, all tables within the source schema are compared against tables in the target schema with the exception of those tables excluded from comparison using the `-et` option. **Note:** There must be no white space between the comma and table names.

`-et, --exclude-tables` *table_1* `[,`*table_2* `] ...`

The tables within the source schema that are to be excluded from comparison. If omitted, only those tables specified with the `-it` option are included for comparison. If both the `-it` and `-et` options are omitted, all source schema tables are included for comparison. **Note:** There must be no white space between the comma and table names.

`-srs, --skip-rowsonlyin-source { true | false }`

When `true` is specified, the logging of differences for rows that exist only in the source database table are skipped. The default is `false`.

`-srt, --skip-rowsonlyin-target { true | false }`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When `true` is specified, the logging of differences for rows that exist only in the target database table are skipped. The default is `false`.

`-srb, --skip-rowsin-both { true | false }`

When `true` is specified, the logging of differences for rows that exist both in the source and target database tables with the same primary key, but with different non-primary key values are skipped. The default is `false`.

`-ld, --logging-dir` *`log_directory_path`*

Directory path to where the Data Validator log and diff files are to be created and stored. If *`log_directory_path`* does not exist, Data Validator attempts to create it. If a full directory path is not specified *`log_directory_path`* is created or assumed to be located relative to the *`XDB_HOME`*`/bin` subdirectory where the `runValidation.sh` script is invoked. (That is, the logs directory is *`XDB_HOME`*`/bin/`*`log_directory_path`*.) Be sure the operating system account used to invoke the `runValidation.sh` script has the privileges to create the directory if it does not already exist, or to create files in the specified directory if it does already exist. If omitted, the default is the *`XDB_HOME`*`/bin/logs` directory.

`-ds, --display-summary { true | false }`

Specify `true` to display only the Data Validator summary. This omits the source and target database connection information as well as the detailed breakdown of the results by source database table. Specify `false` to display all of the Data Validator results. The type and amount of information that is displayed at the command line console when the Data Validator is invoked is the same information that is also stored in the log file for that run. If omitted, the default is `false` (that is, all of the Data Validator results is displayed).

`-sdbms, --source-dbms` *`database_type`*

The type of the source database server. Supported types are `oracle`, `enterprisedb`, `sqlserver`, `sybase`, and `mysql`.

`-sh, --source-host` *`host`*

The IP address or server name of the host on which the source database server is running.

`-sp, --source-port` *`port`*

The port number on which the source database server is listening for connections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

`-sdb, --source-database` *dbname*

The database name of the source database.

`-su, --source-user` *user*

The database user name for connecting to the source database.

`-spw, --source-password` *password*

The password of the source database user in unencrypted form.

`-tdbms, --target-dbms` *database_type*

The type of the target database server. Supported types are `enterprisedb` and `oracle`.

`-th, --target-host` *host*

The IP address or server name of the host on which the target database server is running.

`-tp, --target-port` *port*

The port number on which the target database server is listening for connections.

`-tdb, --target-database` *dbname*

The database name of the target database.

`-tu, --target-user` *user*

The database user name for connecting to the target database.

`-tpw, --target-password` *password*

The password of the target database user in unencrypted form.

`-bs, --batch-size` *row_count*

The `-bs` option specifies the number of rows to group in a batch to be used for comparison across the source and target database tables. For example, if a table contains 1000 rows, then a `-bs` setting of 100 requires 10 batch iterations to complete the comparison across the source and target databases. The Data Validator reads 100 rows, both from the source and target tables, and adds them in

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

source and target buffers. The validation thread then reads the 100 rows from the source and target buffers and performs the comparison. It will then move to read and prepare the next 100 rows for comparison and so on. Note that the actual database round trips required to bring in 100 rows from the database depends on the `-fs` option for the fetch size. For example, an `-fs` setting of 100 needs just one round trip whereas an `-fs` setting of 10 requires 10 database round trips.

`-fs, --fetch-size` *`row_count`*

Performing data validation for tables that are quite large in size may cause the Data Validator to terminate with an out of heap space error when using the default fetch size of 5000 rows. Use the `-fs` option to specify a smaller fetch size to help avoid the out of heap space issue. The result set iteration will bring in as many rows as represented by the *`row_count`* value in a single database round trip.

Examples

The following examples use an Oracle source database and an Advanced Server target database to compare the tables in schema `EDB` on Oracle against the tables in schema `public` in Advanced Server.

The following lists the tables in schema `EDB` along with the content of tables `DEPT` and `EMP` in the Oracle source database:

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
------------------------------
ORATAB
DEPT
EMP
JOBHIST

SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 FINANCE        CHICAGO

SQL> SELECT * FROM emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      9001 SMITH      ANALYST         7566                 8500                    20
      9002 ROGERS     SALESMAN        7698                 8000       4000         30

16 rows selected.
```

The following lists the tables in schema `public` along with the content of tables `dept` and `emp` in the Advanced Server `edb` database:

```
edb=# \dt
            List of relations
 Schema |  Name   | Type  |    Owner
--------+---------+-------+-------------
 public | dept    | table | enterprisedb
 public | emp     | table | enterprisedb
 public | jobhist | table | enterprisedb
(3 rows)

edb=# SELECT * FROM dept;
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
(4 rows)

edb=# SELECT * FROM emp;
 empno | ename  |    job    | mgr  |      hiredate       |   sal   |  comm   | deptno
-------+--------+-----------+------+---------------------+---------+---------+--------
  7369 | SMITH  | CLERK     | 7902 | 17-DEC-80 00:00:00 |  800.00 |         |     20
  7499 | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 00:00:00 | 1600.00 |  300.00 |     30
  7521 | WARD   | SALESMAN  | 7698 | 22-FEB-81 00:00:00 | 1250.00 |  500.00 |     30
  7566 | JONES  | MANAGER   | 7839 | 02-APR-81 00:00:00 | 2975.00 |         |     20
  7654 | MARTIN | SALESMAN  | 7698 | 28-SEP-81 00:00:00 | 1250.00 | 1400.00 |     30
  7698 | BLAKE  | MANAGER   | 7839 | 01-MAY-81 00:00:00 | 2850.00 |         |     30
  7782 | CLARK  | MANAGER   | 7839 | 09-JUN-81 00:00:00 | 2450.00 |         |     10
  7788 | SCOTT  | ANALYST   | 7566 | 19-APR-87 00:00:00 | 3000.00 |         |     20
  7839 | KING   | PRESIDENT |      | 17-NOV-81 00:00:00 | 5000.00 |         |     10
  7844 | TURNER | SALESMAN  | 7698 | 08-SEP-81 00:00:00 | 1500.00 |    0.00 |     30
  7876 | ADAMS  | CLERK     | 7788 | 23-MAY-87 00:00:00 | 1100.00 |         |     20
  7900 | JAMES  | CLERK     | 7698 | 03-DEC-81 00:00:00 |  950.00 |         |     30
  7902 | FORD   | ANALYST   | 7566 | 03-DEC-81 00:00:00 | 3000.00 |         |     20
  7934 | MILLER | CLERK     | 7782 | 23-JAN-82 00:00:00 | 1300.00 |         |     10
  9001 | SMITH  | SALESMAN  | 7698 |                    | 8000.00 | 4000.00 |     30
  9002 | ROGERS | SALESMAN  | 7698 |                    | 9500.00 | 4000.00 |     30
(16 rows)
```

Note the following differences:

- The Oracle `EDB` schema contains one additional table named `ORATAB` that does not exist in the Advanced Server `public` schema.
- The Oracle `DEPT` table contains one extra row with `DEPTNO 50` that does not exist in the Advanced Server `dept` table.
- The rows in the `EMP` table with `EMPNO` values `9001` and `9002` have column values that differ between the Oracle and Advanced Server tables
- In this example, the `JOBHIST` table contains identical rows for both the Oracle and Advanced Server tables.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The content of the `datavalidator.properties` file is set as follows:

```
#################################################################
#               Source database connection                      #
#################################################################

#
#source_dbms=(enterprisedb | oracle | sqlserver | sybase | mysql)
#

source_dbms=oracle
source_host=192.168.2.23
source_port=1521
source_database=xe
source_user=edb
source_password=password

#################################################################
#               Target database connection                      #
#################################################################

#
#target_dbms=(enterprisedb | oracle)
#

target_dbms=enterprisedb
target_host=localhost
target_port=5444
target_database=edb
target_user=enterprisedb
target_password=password
```

The following example compares all tables in the Oracle `EDB` schema against the Advanced Server `public` schema.

The Data Validator log files are created in directory `/home/user/datavalidator_logs` as specified with the `-ld` option. The operating system account used to invoke the `runValidation.sh` script has write access to the `/home/user` directory so the Data Validator can create the `datavalidator_logs` subdirectory.

```
$ cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
$ pwd
/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
$ ./runValidation.sh -ss edb -ts public -ld /home/user/datavalidator_logs

EnterpriseDB DataValidator Build 3
--------------------------------------------------------------------------------
Source and target databases connection information
--------------------------------------------------------------------------------

Source database:

     DBMS:     ORACLE
     Host:     192.168.2.23
     Port:     1521
     Database: xe
     User:     edb

Target database:
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
     DBMS:     ENTERPRISEDB
     Host:     localhost
     Port:     5444
     Database: edb
     User:     enterprisedb

--------------------------------------------------------------------------------
Databases data validation process started...
--------------------------------------------------------------------------------

Validating Table DEPT
     Rows validated: 5
     Finished validating table DEPT with 1 errors.
     Logging errors details in the diff file...

Validating Table EMP
     Rows validated: 16
     Finished validating table EMP with 2 errors.
     Logging errors details in the diff file...

Validating Table JOBHIST
     Rows validated: 17
     Finished validating table JOBHIST with 0 errors.

Validating Table ORATAB
     Table not validated as it does not exist on the target database.


DataValidator found 3 errors across source and target databases.
For detailed error report see datavalidator_20150713-144417.diff file.

--------------------------------------------------------------------------------
Data validation process has completed.
--------------------------------------------------------------------------------

********************************************************************************
                            DataValidator Summary
********************************************************************************

              All tables count: 4

              Validated tables count: 3
              Rows count: 38
              Errors count: 3

              Missing tables on the target database count: 1
              Tables list:
                  - EDB.ORATAB

              Tables having only unsupported datatypes count: 0

              Tables having primary key limitation count: 0

              Total time(s): 0.678
              Rows per second: 56

********************************************************************************
```

The Data Validator output indicates the following:

- There is one error in the `DEPT` table (the missing row).
- There are two errors in the `EMP` table (the two rows with mismatching column values)
- The `JOBHIST` table contains no errors.
- The `ORATAB` table does not exist on the target database.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following shows the files created in the Data Validator logs directory:

```
$ pwd
/home/user/datavalidator_logs
$ ls -l
total 24
-rw-rw-r-- 1 user user 18999 Aug 13 15:44 datavalidator_20150713-144417.diff
-rw-rw-r-- 1 user user  2133 Aug 13 15:44 datavalidator_20150713-144417.log
```

The log file contains the same content as displayed when the Data Validator is invoked. The diff file compares the differences where errors were detected.

The following is the diff file as displayed in a text editor:

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

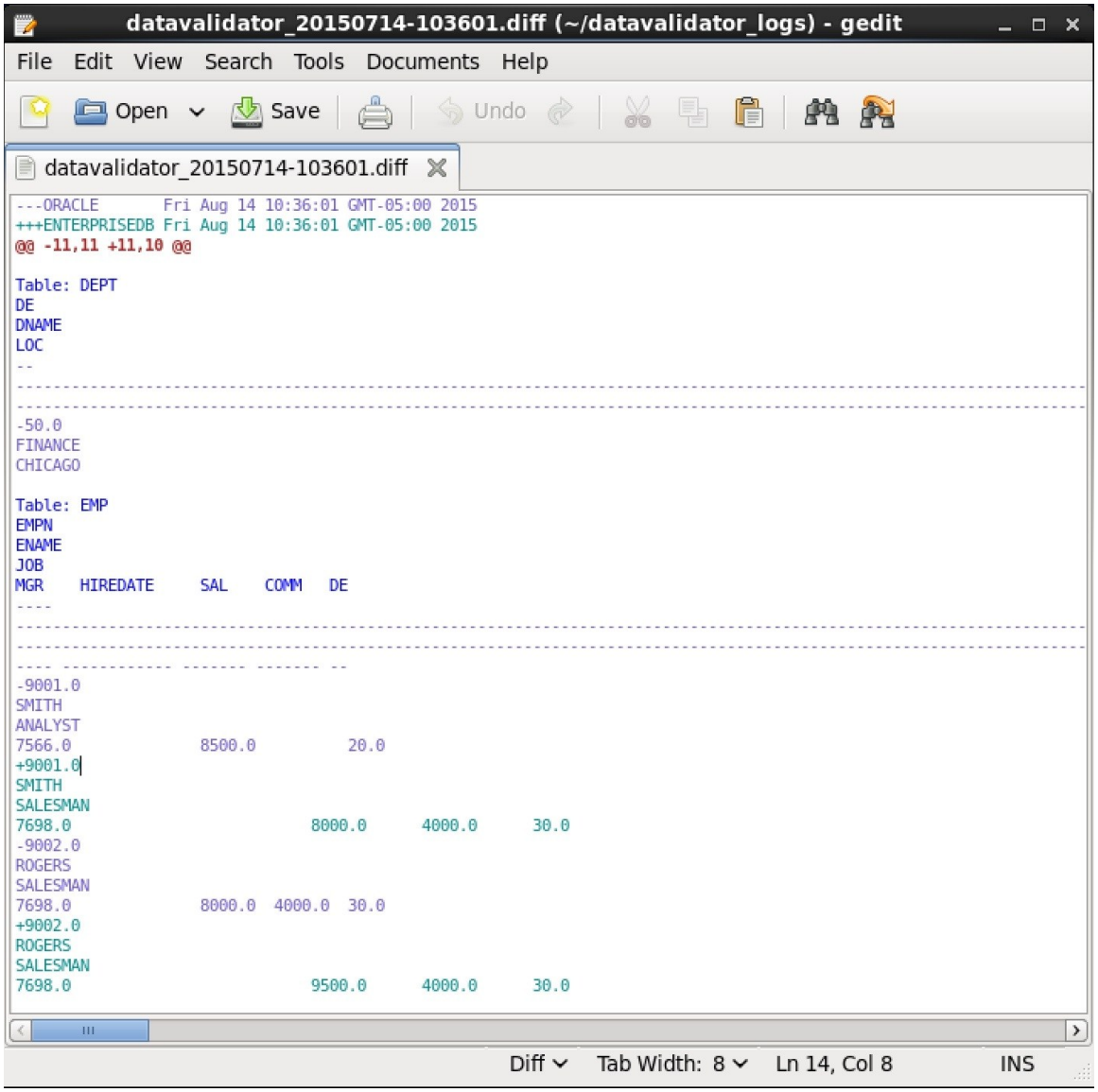


Figure 9-1 - Data Validator diff file

The following example includes only tables `dept` and `emp` with the `-it` option when comparing the Oracle `EDB` schema against the Advanced Server `public` schema.

```
$ cd /opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
$ pwd
/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin
$ ./runValidation.sh -ss edb -ts public -ld /home/user/datavalidator_logs -it
dept,emp

EnterpriseDB DataValidator Build 3
-------------------------------------------------------------------------
Source and target databases connection information
-------------------------------------------------------------------------
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Source database:

     DBMS:     ORACLE
     Host:     192.168.2.23
     Port:     1521
     Database: xe
     User:     edb

Target database:

     DBMS:     ENTERPRISEDB
     Host:     localhost
     Port:     5444
     Database: edb
     User:     enterprisedb

--------------------------------------------------------------------------------
Databases data validation process started...
--------------------------------------------------------------------------------

Validating Table DEPT
     Rows validated: 5
     Finished validating table DEPT with 1 errors.
     Logging errors details in the diff file...

Validating Table EMP
     Rows validated: 16
     Finished validating table EMP with 2 errors.
     Logging errors details in the diff file...


DataValidator found 3 errors across source and target databases.
For detailed error report see datavalidator_20150714-123353.diff file.

--------------------------------------------------------------------------------
Data validation process has completed.
--------------------------------------------------------------------------------

********************************************************************************
                              DataValidator Summary
********************************************************************************

            All tables count: 2

            Validated tables count: 2
            Rows count: 21
            Errors count: 3

            Missing tables on the target database count: 0

            Tables having only unsupported datatypes count: 0

            Tables having primary key limitation count: 0

            Total time(s): 0.539
            Rows per second: 39

********************************************************************************
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following example excludes tables `ORATAB` and `jobhist` with the `-et` option when comparing the Oracle `EDB` schema against the Advanced Server `public` schema. The `-ds true` option results in the display of only the Data Validator summary.

```
$ ./runValidation.sh -ss edb -ts public -ld /home/user/datavalidator_logs -et
ORATAB,jobhist -ds true
Databases data validation process started...

*************************************************************************
                            DataValidator Summary
*************************************************************************

            All tables count: 2

            Validated tables count: 2
            Rows count: 21
            Errors count: 3

            Missing tables on the target database count: 0

            Tables having only unsupported datatypes count: 0

            Tables having primary key limitation count: 0

            Total time(s): 0.535
            Rows per second: 39

*************************************************************************
```

For this run, the corresponding log file contains only the Data Validator summary, omitting the source and target database connection information along with the error breakdown by table.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10 Appendix

This chapter discusses various miscellaneous topics.

10.1 Certified and Supported Database Servers and Configurations

This section lists the certified and supported versions of database server products that may be used with xDB Replication Server along with the permitted combinations of such products within an xDB replication system.

10.1.1 Certified and Supported Product Versions

The following are the database product versions that may be used with xDB Replication Server.

- **PostgreSQL versions 9.2, 9.3, 9.4, 9.5, 9.6, 10, and 11**
- **Advanced Server versions 9.2, 9.3, 9.4, 9.5, 9.6, 10, and 11**
- **Oracle 10g Release 2 version 10.2.0.1.0** has been explicitly certified. Newer minor versions in the 10.2 line are supported as well.
- **Oracle 11g Release 2 version 11.2.0.2.0** has been explicitly certified. Newer minor versions in the 11.2 line are supported as well.
- **Oracle 12c version 12.1.0.2.0** has been explicitly certified. Newer minor versions in the 12.1 line are supported as well.
- **SQL Server 2008 version 10.50.1617.0** has been explicitly certified. Newer minor versions in the 10.50 line are supported as well.
- **SQL Server 2012 version 11.0.6020.0** has been explicitly certified. Newer minor versions in the 11.0 line are supported as well.
- **SQL Server 2014 version 12.0.5000.0** has been explicitly certified. Newer minor versions in the 12.0 line are supported as well.

Note: For Advanced Server version 11, certification has been initially performed against version 11 beta, but no issue is expected with the version 11 GA release.

Please contact your EnterpriseDB Account Manager or sales@enterprisedb.com if you require support for other platforms.

10.1.2 A Note Regarding Oracle RAC and Oracle Exadata

Oracle Real Application Clusters (RAC) and Oracle Exadata are not supported by xDB Replication Server. These Oracle products have not been evaluated nor certified with xDB Replication Server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.1.3 Permitted Configurations and Combinations

Depending upon the database products you are using with xDB Replication Server (Oracle, SQL Server, PostgreSQL, or Advanced Server) along with the *compatibility configuration mode* if you are using Advanced Server, certain combinations of a source database server and a target database server are not permitted for a publication and its associated subscription in a single-master replication system.

Similarly, only certain combinations of database products and Advanced Server compatibility configuration modes can be used together in a multi-master replication system.

For a single-master replication system, the source refers to the database server of the publication database. The target refers to the database server of the subscription database.

For a multi-master replication system, all of the participating database servers act as both a source and a target for all other participating database servers, so the restrictions pertain to the combinations of database servers and compatibility configuration modes that can be used together in the same multi-master replication system.

This section presents the specific combinations of database server configurations that can be used for a publication and its associated subscription in a single-master replication system, and the combinations of database server configurations that can be used in a multi-master replication system.

10.1.3.1 Advanced Server Compatibility Configuration Modes

Advanced Server supports two *compatibility configuration modes* of operation, which are the following:

- **Oracle compatible configuration mode.** Operations are performed using Oracle syntax and semantics for data types, functions, database object creation, and so forth. This mode is useful when your applications are migrated from Oracle, or you want your applications built in an Oracle compatible fashion.
- **PostgreSQL compatible configuration mode.** Operations are performed using native PostgreSQL syntax and semantics. This mode is useful when your applications are migrated from PostgreSQL, or you want your applications built in a PostgreSQL compatible fashion.

For more information on features supported in Oracle compatible configuration mode, see the *Database Compatibility for Oracle Developer's Guide* located at:

https://www.enterprisedb.com/resources/product-documentation

The compatibility configuration mode is selected at the time you install Advanced Server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.1.3.2 Permitted SMR Source and Target Configurations

The following table shows the combinations of source and target database server products and Advanced Server compatibility configuration modes permitted by xDB Replication Server for single-master replication systems:

Table 10-1 – Permitted Source and Target Configurations

Source \ Target	Oracle	Microsoft SQL Server	PostgreSQL	Advanced Server (Oracle compatible)	Advanced Server (PostgreSQL compatible)
Oracle	No	No	Yes	Yes	Yes
Microsoft SQL Server	No	No	Yes	Yes	Yes
PostgreSQL	No	Yes	Yes	Yes	Yes
Advanced Server (Oracle compatible)	Yes	Yes	No	Yes	No
Advanced Server (PostgreSQL compatible)	No	Yes	Yes	Yes	Yes

In the preceding table, the left hand column lists the possible source database server products including the possible Advanced Server compatibility configuration modes. The top row lists the same set of possible target database server products and Advanced Server compatibility configuration modes.

'Yes' at the intersection of a source and target indicates that xDB Replication Server permits replication using that combination of database server configurations for a publication and its associated subscription. 'No' indicates replication is not permitted for that combination.

10.1.3.3 Permitted MMR Database Server Configurations

For multi-master replication systems, each master node acts as both a source for all master nodes and a target for all master nodes. Thus, the permitted database servers comprising a particular multi-master replication system or *cluster* is determined by the overall composition of the cluster, which is initially established when selecting the database type of the master definition node (see Step 3 in Section 6.2.2).

There are two basic cluster types that can be characterized as follows:

- **PostgreSQL compatible cluster.** All master nodes must consist of PostgreSQL database servers or Advanced Servers installed in PostgreSQL compatible configuration mode.
- **Advanced Server Oracle compatible cluster.** All master nodes must consist of Advanced Servers installed in Oracle compatible configuration mode.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following table summarizes the permitted database server configurations allowed in the two cluster types.

Table 10-2 – Permitted Database Server Configurations by Cluster Type

Database Server \ Cluster Type	PostgreSQL Compatible Cluster	Advanced Server Oracle Compatible Cluster
PostgreSQL	Yes	No
Advanced Server (PostgreSQL compatible)	Yes	No
Advanced Server (Oracle compatible)	No	Yes

In the preceding table, the left hand column lists the possible database server products including the possible Advanced Server compatibility configuration modes. The top row lists the supported cluster types.

'Yes' at the intersection of a database server and cluster type indicates that xDB Replication Server permits the database server and the specified configuration mode in the cluster type. 'No' indicates the database server and the specified configuration mode cannot be used in the cluster type.

10.2 Upgrading to xDB Replication Server 6.2

This section describes the process of installing xDB Replication Server 6.2 when you have existing single-master or multi-master replication systems that are running under xDB Replication Server version 6.1.x or 6.0.x.

It is assumed that you will be installing xDB Replication Server 6.2 on the same host machine that is currently running xDB Replication Server 6.1.x or 6.0.x, and that you will then manage the existing replication systems using xDB Replication Server 6.2.

A direct upgrade is supported only from xDB Replication Server versions 6.1.x or 6.0.x.

If you have xDB Replication Server 5.1.x, you must first upgrade this version to xDB Replication Server 6.0. See Section 10.2 "Upgrading to xDB Replication Server 6.0" in the *EDB Replication Server 6.0 User's Guide* located at:

https://www.enterprisedb.com/resources/product-documentation

After upgrading to version 6.0, you can then upgrade to 6.2.

The following sections illustrate the upgrade process from xDB Replication Server 6.1. The same steps apply for upgrading from xDB Replication Server 6.0, but with different version numbers in the file and directory names of the older product.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.2.1 Upgrading with the Graphical User Interface Installer

Perform the following steps to upgrade to xDB Replication Server 6.2 using the graphical user interface installer.

Step 1: Any pending backlog of transactions on the publication tables must be replicated before starting the upgrade process.

Step 2: After all pending transactions have been replicated to their target databases, stop the xDB Replication Server 6.1.x publication server and subscription server (see sections 5.2.1 and 5.3.1).

Step 3: Install xDB Replication Server 6.2. See Chapter 3 for instructions on installing xDB Replication Server, but note the differences described in the following steps.

Step 4: Following the acceptance of the license agreement in Step 11 of Section 3.1, the Select Components screen appears, but with the entries grayed out. The old xDB Replication Server components are replaced by the new ones in the old xDB Replication Server's directory location. Click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

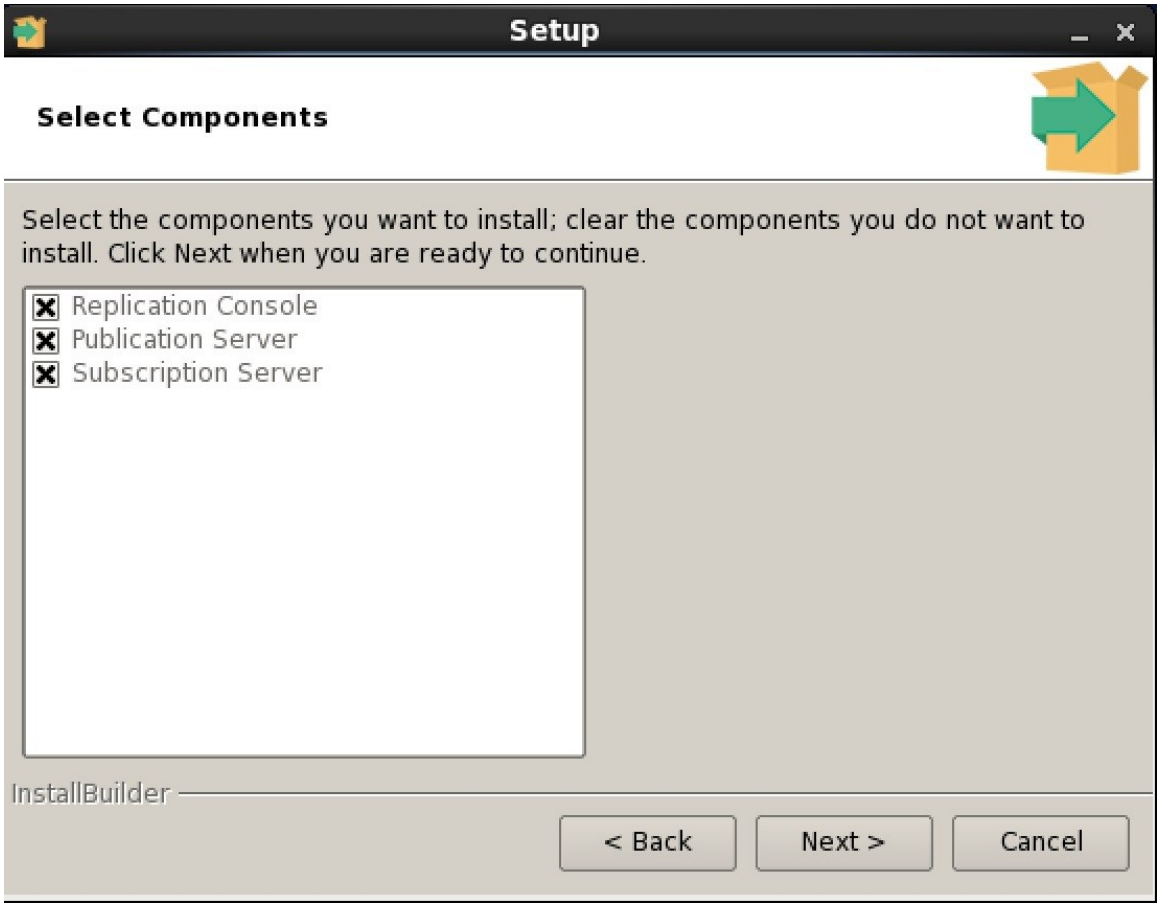


Figure 10-1 - Select components

Step 5: The Existing Installation screen confirms that an existing xDB Replication Server installation was found. Click the Next button to proceed with the upgrade.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

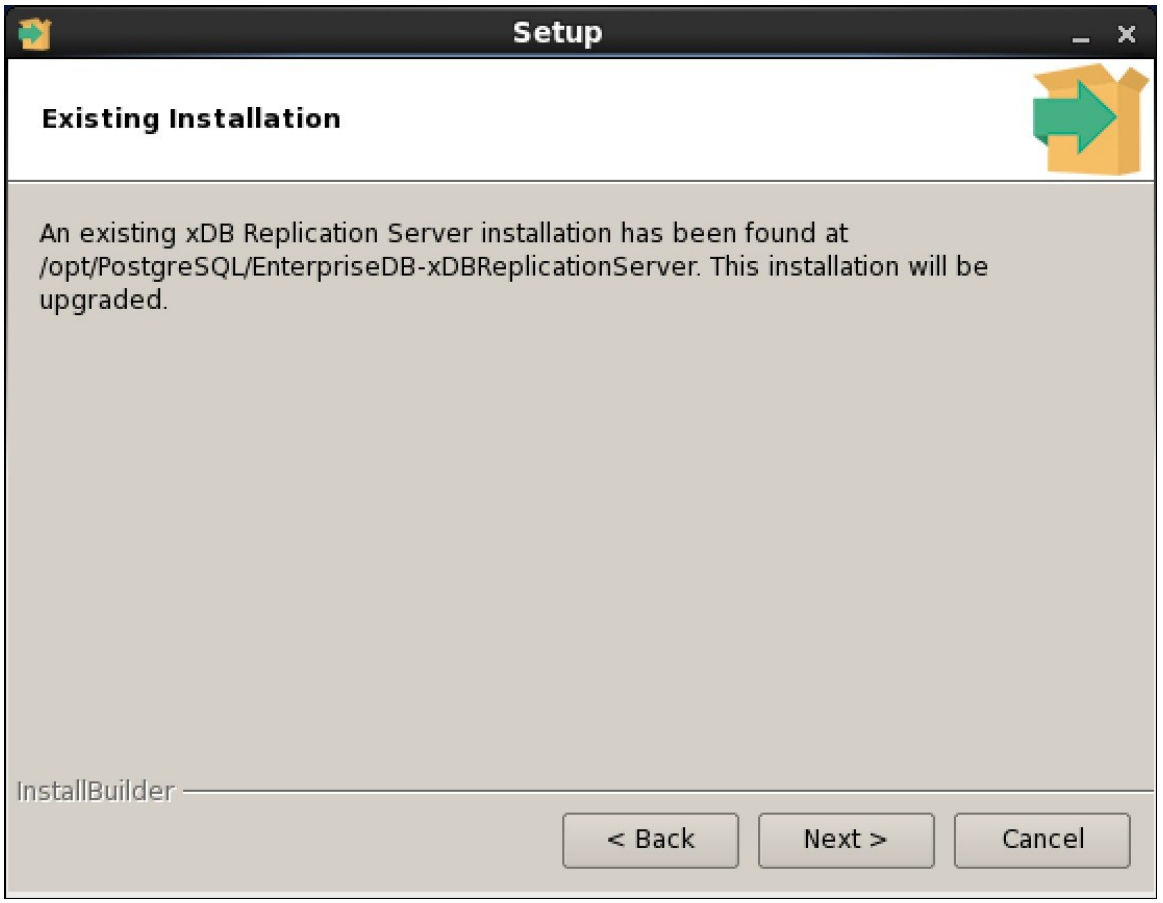


Figure 10-2 - Existing installation

Step 6: On the Ready to Install screen, click the Next button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

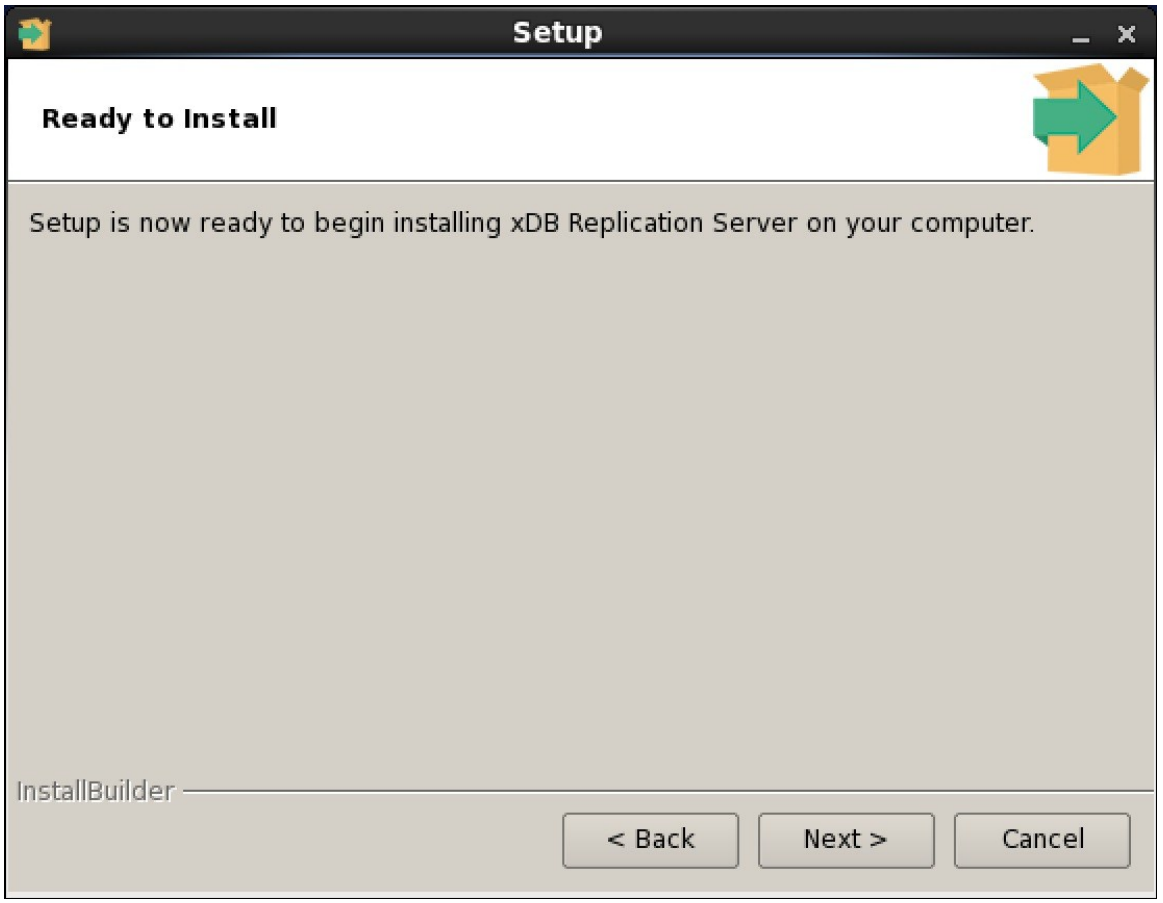


Figure 10-3 - Ready to install

Step 7: The remaining screens that appear confirm completion of the installation process and allow you to exit from Stack Builder or StackBuilder Plus.

Step 8: After installation completes, the publication server of the new xDB Replication Server product should be running, connected to the controller database used by xDB Replication Server 6.1. The subscription server may or may not be running at this point, however, that is an expected outcome of this process.

Step 9: Complete the publication server and subscription server configuration file setup.

In the *XDB_HOME*/etc directory, a new set of configuration files for xDB Replication Server version 6.2 are created. These files are named xdb_pubserver.conf.new and xdb_subserver.conf.new. The new configuration files contain any new configuration options added for xDB Replication Server 6.2.

The old configuration files used by xDB Replication Server version 6.1.x remain unchanged as xdb_pubserver.conf and xdb_subserver.conf.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Merge the old and new configuration files so that the resulting, active configuration files contain any new xDB Replication Server 6.2 configuration options as well as any non-default settings you used with xDB Replication Server 6.1.x and wish to continue to use with xDB Replication Server 6.2.

The final set of active configuration files must be named `xdb_pubserver.conf` and `xdb_subserver.conf`.

In the *`XDB_HOME`*`/etc/sysconfig` directory, make sure the xDB Startup Configuration file `xdbReplicationServer-62.config` contains the parameter settings you wish to use with xDB Replication Server 6.2. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

Step 10: Restart the publication server and the subscription server (see sections 5.2.1 and 5.3.1).

Step 11: Check the publication server and subscription server log files to verify that no errors have occurred (see Section 10.3.2.4).

Step 12: Adjust the publication server and subscription server port numbers if necessary.

The xDB Replication Server 6.2 publication and subscription servers are installed to use the default port numbers `9051` and `9052`, respectively. If the xDB Replication Server 6.1.x replication systems used port numbers other than `9051` and `9052`, then perform the modifications to correct this inconsistency as described in Section 10.2.3.

If no such adjustment to the port numbers is needed, register the publication server and subscription server with the xDB Replication Console as described in sections 5.2.1 and 5.3.1. The existing replication systems should appear in the replication tree of the xDB Replication Console.

Step 13: You are now ready to use xDB Replication Server 6.2 to create new replication systems and manage existing ones.

10.2.2 Upgrading with the xDB Replication Server RPM Package

If you are using xDB Replication Server 6.1.x that was installed using the xDB RPM package, upgrading to xDB Replication Server 6.2 from an RPM package is accomplished as described in this section.

Note: Be sure the repository configuration file `edb.repo` for xDB Replication Server 6.2 is set up in the `/etc/yum.repos.d` directory. See Section 3.3 for information.

Step 1: Any pending backlog of transactions on the publication tables must be replicated before starting the upgrade process.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 2: After all pending transactions have been replicated to their target databases, stop the xDB Replication Server 6.1.x publication server and subscription server (see sections 5.2.1 and 5.3.1).

Step 3: Save a copy of the following configuration files:

- `/etc/edb-repl.conf`
- `/usr/ppas-xdb-6.1/etc/xdb_pubserver.conf`
- `/usr/ppas-xdb-6.1/etc/xdb_subserver.conf`
- `/usr/ppas-xdb-6.1/etc/sysconfig/xdbReplicationServer-61.config`

Copies of these files are typically saved by the upgrade process if the files had been modified since their original installation. However, it is safest to save copies in case the upgrade process fails to do so. Use the saved files as your xDB Replication Server 6.1.x configuration files for the updates described in Step 7.

Step 4: If any Oracle publication or subscription databases are used in existing single-master replication systems, make sure a copy of the Oracle JDBC driver, version `ojdbc5` or later, will be accessible by the publication server and subscription server where xDB Replication Server 6.2 will be installed. See Section 5.1.3.1 for information.

Note: There are two options available: Option 1) Copy the Oracle JDBC driver to the `jre/lib/ext` subdirectory of your Java runtime environment. Option 2) Copy the Oracle JDBC driver to the `lib/jdbc` subdirectory of the xDB Replication Server installation directory.

It is suggested that you perform option 1 (copy the Oracle JDBC driver to the `jre/lib/ext` subdirectory of your Java runtime environment).

If on the other hand you perform option 2, you must copy the Oracle JDBC driver to the `/usr/ppas-xdb-6.2/lib/jdbc` directory after you have installed xDB Replication Server 6.2.

Step 5: It is best to ensure that the controller database is up and running. The other publication and subscription databases of existing SMR and MMR systems do not need to be up and running.

Step 6: As the `root` account invoke the `yum update` command to begin the upgrade from xDB Replication Server 6.1.x to xDB Replication Server 6.2 as shown by the following:

```
yum update ppas-xdb*
```

Be sure to include the asterisk character (`*`) following `ppas-xdb` in order to update all xDB Replication Server components.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following is an example:

```
[root@localhost ~]# yum update ppas-xdb*
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: mirrors.lga7.us.voxel.net
 * epel: mirror.steadfast.net
 * extras: mirror.vcu.edu
 * updates: mirror.vtti.vt.edu
edb-repos
| 2.4 kB     00:00
enterprisedb-xdb60
| 2.4 kB     00:00
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package ppas-xdb.x86_64 0:6.1.3-1.rhel6 will be updated
---> Package ppas-xdb.x86_64 0:6.2.0-1.rhel6 will be an update
---> Package ppas-xdb-console.x86_64 0:6.1.3-1.rhel6 will be updated
---> Package ppas-xdb-console.x86_64 0:6.2.0-1.rhel6 will be an update
---> Package ppas-xdb-libs.x86_64 0:6.1.3-1.rhel6 will be updated
---> Package ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6 will be an update
---> Package ppas-xdb-publisher.x86_64 0:6.1.3-1.rhel6 will be updated
---> Package ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6 will be an update
---> Package ppas-xdb-subscriber.x86_64 0:6.1.3-1.rhel6 will be updated
---> Package ppas-xdb-subscriber.x86_64 0:6.2.0-1.rhel6 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
 Package                                  Arch                              Version
Repository                               Size
================================================================================
================================================================================
Updating:
 ppas-xdb                                 x86_64                            6.2.0-
1.rhel6                          enterprisedb-xdb60                          6.8 k
 ppas-xdb-console                         x86_64                            6.2.0-
1.rhel6                          enterprisedb-xdb60                          3.0 M
 ppas-xdb-libs                            x86_64                            6.2.0-
1.rhel6                          enterprisedb-xdb60                           13 M
 ppas-xdb-publisher                       x86_64                            6.2.0-
1.rhel6                          enterprisedb-xdb60                           35 k
 ppas-xdb-subscriber                      x86_64                            6.2.0-
1.rhel6                          enterprisedb-xdb60                           12 k

Transaction Summary
================================================================================
================================================================================
Upgrade       5 Package(s)

Total download size: 16 M
Is this ok [y/N]: y
Downloading Packages:
(1/5): ppas-xdb-6.2.0-1.rhel6.x86_64.rpm
| 6.8 kB     00:00
(2/5): ppas-xdb-console-6.2.0-1.rhel6.x86_64.rpm
| 3.0 MB     00:01
(3/5): ppas-xdb-libs-6.2.0-1.rhel6.x86_64.rpm
|  13 MB     00:01
(4/5): ppas-xdb-publisher-6.2.0-1.rhel6.x86_64.rpm
|  35 kB     00:00
(5/5): ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64.rpm
|  12 kB     00:00
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total
3.5 MB/s |  16 MB     00:04
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating   : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
1/10
  Updating   : ppas-xdb-publisher-6.2.0-1.rhel6.x86_64
2/10
  Updating   : ppas-xdb-console-6.2.0-1.rhel6.x86_64
3/10
  Updating   : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
4/10
  Updating   : ppas-xdb-6.2.0-1.rhel6.x86_64
5/10
  Cleanup    : ppas-xdb-6.1.3-1.rhel6.x86_64
6/10
  Cleanup    : ppas-xdb-subscriber-6.1.3-1.rhel6.x86_64
7/10
  Cleanup    : ppas-xdb-console-6.1.3-1.rhel6.x86_64
8/10
  Cleanup    : ppas-xdb-publisher-6.1.3-1.rhel6.x86_64
9/10
  Cleanup    : ppas-xdb-libs-6.1.3-1.rhel6.x86_64
10/10
warning: /usr/ppas-xdb-6.1/etc/sysconfig/xdbReplicationServer-61.config saved as
/usr/ppas-xdb-6.1/etc/sysconfig/xdbReplicationServer-61.config.rpmsave
  Verifying  : ppas-xdb-6.2.0-1.rhel6.x86_64
1/10
  Verifying  : ppas-xdb-publisher-6.2.0-1.rhel6.x86_64
2/10
  Verifying  : ppas-xdb-console-6.2.0-1.rhel6.x86_64
3/10
  Verifying  : ppas-xdb-libs-6.2.0-1.rhel6.x86_64
4/10
  Verifying  : ppas-xdb-subscriber-6.2.0-1.rhel6.x86_64
5/10
  Verifying  : ppas-xdb-publisher-6.1.3-1.rhel6.x86_64
6/10
  Verifying  : ppas-xdb-subscriber-6.1.3-1.rhel6.x86_64
7/10
  Verifying  : ppas-xdb-6.1.3-1.rhel6.x86_64
8/10
  Verifying  : ppas-xdb-console-6.1.3-1.rhel6.x86_64
9/10
  Verifying  : ppas-xdb-libs-6.1.3-1.rhel6.x86_64
10/10

Updated:
  ppas-xdb.x86_64 0:6.2.0-1.rhel6             ppas-xdb-console.x86_64 0:6.2.0-1.rhel6
ppas-xdb-libs.x86_64 0:6.2.0-1.rhel6  ppas-xdb-publisher.x86_64 0:6.2.0-1.rhel6
  ppas-xdb-subscriber.x86_64 0:6.2.0-1.rhel6

Complete!
```

At this point the publication server and the subscription server for xDB Replication Server 6.2 are not running.

The directories now contain the following:

- xDB Replication Server 6.2 is installed in directory location `/usr/ppas-xdb-6.2`.
- xDB Replication Server 6.1.x remains in directory location `/usr/ppas-xdb-6.1`, but with the files removed from the subdirectories such as `bin` and `lib`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- In the `etc` subdirectory, there may be the configuration files renamed as `xdb_pubserver.conf.rpmsave` and `xdb_subserver.conf.rpmsave`.
- In the `etc/sysconfig` subdirectory, there may be the configuration file renamed as `xdbReplicationServer-61.config.rpmsave`.
- In the `/etc` directory, there may be either one or two xDB Replication Configuration files named `edb-repl.conf` and possibly `edb-repl.conf.rpmsave`. The file `edb-repl.conf` should contain the connection and authentication information for the controller database used by the xDB 6.1.x publication server. The file `edb-repl.conf.rpmsave` contains only the new administrator user parameters `admin_user` and `admin_password`. Before starting the publication server and subscription server, be sure the controller database is up and running, and the `edb-repl.conf` file contains the controller database connection and authentication parameters.

Step 7: Complete the publication server and subscription server configuration file setup.

In the `/usr/ppas-xdb-6.2/etc` directory, a new set of configuration files for xDB Replication Server version 6.2 are created. These files are named `xdb_pubserver.conf` and `xdb_subserver.conf`. The new configuration files contain any new configuration options added for xDB Replication Server 6.2.

The old configuration files used by xDB Replication Server version 6.1.x might be found in the `/usr/ppas-xdb-6.1/etc` directory renamed as `xdb_pubserver.conf.rpmsave` and `xdb_subserver.conf.rpmsave`.

Note: If these files do not exist, use the ones you saved in Step 3.

Merge the old and new configuration files so that the resulting, active configuration files contain any new xDB Replication Server 6.2 configuration options as well as any non-default settings you used with xDB Replication Server 6.1.x and wish to continue to use with xDB Replication Server 6.2.

The final set of active configuration files must be contained in directory `/usr/ppas-xdb-6.2/etc` named `xdb_pubserver.conf` and `xdb_subserver.conf`.

In the `/usr/ppas-xdb-6.2/etc/sysconfig` directory, make sure the xDB Startup Configuration file `xdbReplicationServer-62.config` contains the parameter settings you wish to use with xDB Replication Server 6.2. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

Step 8: Restart the publication server and the subscription server (see sections 5.2.1 and 5.3.1).

Step 9: Check the publication server and subscription server log files to verify that no errors have occurred (see Section 10.3.2.4).

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 10: Adjust the publication server and subscription server port numbers if necessary.

The xDB Replication Server 6.2 publication and subscription servers are installed to use the default port numbers `9051` and `9052`, respectively. If the xDB Replication Server 6.1.x replication systems used port numbers other than `9051` and `9052` for the publication and subscription servers, then perform the modifications to correct this inconsistency as described in Section 10.2.3.

If no such adjustment to the port numbers is needed, register the publication server and subscription server with the xDB Replication Console as described in sections 5.2.1 and 5.3.1. The existing replication systems should appear in the replication tree of the xDB Replication Console.

Step 11: You are now ready to use xDB Replication Server 6.2 to create new replication systems and manage existing ones.

10.2.3 Updating the Publication and Subscription Server Ports

The newly installed publication server and subscription server of xDB Replication Server 6.2 are configured to use the default port numbers `9051` and `9052`, respectively. These port numbers are set in the xDB Startup Configuration file as described in Section 2.3.1.4.

If your xDB Replication Server 6.1.x replication systems were running under port numbers other than `9051` and `9052`, some of your settings in xDB Replication Server 6.2 must be adjusted to continue to use these existing replication systems.

Note: The following changes regarding port `9052` and the subscription server are only needed if you are running a single-master replication system. If you are using only a multi-master replication system, then only the changes involving port `9051` and the publication server are needed.

There are two methods to correct this as summarized by the following two points:

- To continue to use the old port numbers (other than `9051` and `9052`) that were in use for xDB Replication Server 6.1.x, stop the publication and subscription servers. Change the settings of the `PUBPORT` and `SUBPORT` parameters in the xDB Startup Configuration file from `9051` and `9052` to the old port numbers used by xDB Replication Server 6.1.x. Restart the publication and subscription servers. Register the publication server and the subscription server with the old xDB Replication Server 6.1.x port numbers along with the admin user and password as described in sections 5.2.1 and 5.3.1.
- To use the default port numbers `9051` and `9052` with the xDB Replication Server 6.1.x replication systems, you must replace the old port numbers with the default port numbers `9051` and `9052`. Register the publication server and the subscription server with port numbers `9051` and `9052` along with the admin user and password

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

as described in sections 5.2.1 and 5.3.1. For single-master replication systems only, you then need to change the port numbers stored in the control schema from the old port numbers to `9051` and `9052`. First, perform the procedure described in Section 7.6.1.2, and then perform the procedure described in Section 5.5.3.

After making the changes as previously described, click the Refresh button of the xDB Replication Console. The replication tree of the xDB Replication Console should display the complete set of nodes for the replication systems.

10.3 Resolving Problems

This section contains tips for locating and correcting various problems that may occur.

10.3.1 Error Messages

The following table is a list of certain error messages that can appear from the xDB Replication Console. The messages are listed in alphabetical order based on the first word in the message following "a", "an", or "the".

When an error message is displayed by the xDB Replication Console, it may be followed by a specific reason as denoted by `Reason: reason_for_failure` as in the following example:

```
Authentication failed. Reason: Invalid user name/password.
```

The various specific reasons are not listed for all messages in the table.

This table also lists only the messages that typically involve initial configuration operations requiring additional information for resolving the problem. Messages related to less complicated corrections for simpler operations are not listed in this table.

Table 10-3 - xDB Replication Server Error Messages

Error Messages and Resolutions
`Authentication failed. Reason: Invalid user name/password.` *Resolution:* Occurs when registering a publication server or subscription server. Verify the user name and password you enter matches the admin user name and password in the xDB Replication Configuration file on the host you are running the publication server or subscription server. See Section 2.3.1.3.
`Cannot register database because it is already registered by a publication service.` *Resolution:* Only one publication database definition can be created for any given database. (Oracle is the exception whereby more than one publication database definition can be created for the same Oracle database if different Oracle user names are specified in each publication database definition.)
`The connection could not be established with the server. Verify that the server is running and accepting connections. Reason: Connection refused to host: xxx.xxx.xx.xxx; nested exception is: java.net.ConnectException: Connection refused`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
Resolution: Occurs whenever a Java RMI connection cannot be made to the publication server, the subscription server, or a database server. Can occur when registering a publication or subscription server, adding a publication database or a subscription database, or identifying the publication server for a new subscription. Verify you have entered the correct host IP address and port number of the server. Verify the server is running (see Section 10.3.4.2). If the server is running on Linux, verify that in the `/etc/hosts` file, the host name is mapped to the correct network IP address, which matches the IP address returned by the Linux `/sbin/ifconfig` command, and also matches the IP address you entered in the Host field of the dialog box. Alternatively, instead of modifying the `/etc/hosts` file, set configuration option `java.rmi.server.hostname` to the IP address of the publication or subscription server (see Section 10.4.1.7). Do not use the loopback address `127.x.x.x` for this entry.
`Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.` *Resolution:* Occurs when attempting to save a publication database definition. The publication server cannot connect to the database server network location given in the Add Database dialog box. Verify that the correct IP address and port for the database server are given. Verify that the database server is running and is accessible from the host running the publication server.
`Could not connect to the database server. Reason: FATAL: number of requested standby connections exceeds max_wal_senders (currently n)` *Resolution:* Occurs when attempting a snapshot replication from a publication database configured with the log-based method of synchronization replication (that is, WAL based logical replication), and the additional concurrent connection for logical replication exceeds the current setting, *n*, of the `max_wal_senders` configuration parameter in the `postgresql.conf` file. Increase the value of `max_wal_senders` in the `postgresql.conf` file of the database server running the publication database. Restart the database server containing the publication database. See Section 2.2.10.
`Currently no publication exists on the publication server. Please create at least one publication on the server and then retry.` *Resolution:* Occurs when attempting to create a subscription. If there are no publications in the specified publication server, then this error message is displayed.
`The database cannot be registered because a partial schema already exists. A manual cleanup is required to proceed. For help with manual cleanup please check out our product documentation.` *Resolution:* The metadata database objects from a prior publication already exist in the schema under which the publication server is attempting to create new metadata database objects. Perform the operation described in Section 10.3.4.3.
`Database cannot be removed. Reason: Publication database connection cannot be removed as one or more publications are defined against it.` *Resolution:* Make sure all publications subordinate to the publication database definition have been removed. If no publications appear under the Publication Database node in the xDB Replication Console replication tree and the error persists, there may be a problem with the control schema objects. Perform the operation described in Section 10.3.4.3.
`Database cannot be removed. Reason: Publication service failed to clean up replication control schema tables.` *Resolution:* The control schema objects under the Oracle publication database user schema or under the Postgres or SQL Server schemas `_edb_replicator_pub`, `_edb_replicator_sub`, or `_edb_scheduler` cannot be deleted by the publication server. The control schema objects or schemas may have already been deleted. The publication database definition cannot be removed using the xDB Replication Console. Perform the operation described in Section 10.3.4.3.
`Database cannot be removed. Reason: The target publication database is`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
`currently set as the Controller database and is being referenced by one or more dependent nodes.` *Resolution:* Occurs when attempting to remove the publication database currently set as the controller database. Select another publication database to be used as the controller database. Use the Set As Controller option in the publication databases' context menu to set this database as the controller database. You can then remove the original publication database. See Section 7.7.
`Database cannot be set as controller. Reason: Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.` *Resolution:* Occurs when attempting to set a publication database as the controller database and the database is not accessible by the publication server. Verify that the correct IP address and port has been defined in the publication database definition. Verify that the database server is running and is accessible from the host running the publication server.
`Database connection cannot be added. Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.` *Resolution:* Occurs when attempting to save a subscription database definition. The subscription server cannot connect to the database server network location given in the Add Database dialog box. Verify that the correct IP address and port for the database server are given. Verify that the database server is running and is accessible from the host running the subscription server.
`Database connection cannot be added. FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`", user "`*`user_name`*`", database "`*`db_name`*`", SSL off` *Resolution:* Occurs when attempting to save a subscription database definition. The subscription server is not permitted to connect to the database at the network location given in the Add Database dialog box. Verify that the database host IP address, port number, database user name, password, and database identifier are correct. Verify there is an entry in the `pg_hba.conf` file permitting access to the database by the given user name originating from the IP address where the subscription server is running.
`Database connection cannot be added. Controller database is not initialized yet.` *Resolution:* Occurs when attempting to add a subscription database. Verify that the xDB Replication Configuration file on the host running the subscription server contains an entry for a valid controller database. Verify that a publication database has been defined under the publication server as the controller database and its connection information is recorded in the xDB Replication Configuration file. See Section 2.3.1.3.
`The database type for the selected database is different than that of the MDN database. Each database should be of the same type in a MMR cluster.` *Resolution:* All database servers in a multi-master replication system must be of the same type – either all PostgreSQL (or Advanced Server installed in PostgreSQL compatible configuration mode); or all Advanced Server installed in Oracle compatible configuration mode. This error message is displayed when attempting to add a master node and the database server type differs from the database server type of the master definition node. See Section 10.1.3.3.
`An error occurred while removing tables from other Master node(s). Please refer to the user manual for instructions on how to remove shadow tables and triggers from Master node(s).` *Resolution:* When a master node of a multi-master replication system is deleted using the xDB Replication Console or the xDB Replication Server CLI, the control schema objects that were created in the master node are also dropped. These include schemas `_edb_replicator_pub`,

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
_edb_replicator_sub, and _edb_scheduler. For the log-based method of synchronization replication there are shadow tables and triggers on the publication tables as well. If any of these control schema objects fail to be dropped, this error message is displayed. See Section 10.3.4.3 for directions on how to remove these control schema objects.
FATAL: no pg_hba.conf entry for host "*xxx.xxx.xx.xxx*", user "*user_name*", database "*db_name*", SSL off *Resolution:* Occurs when attempting to save a publication database definition. The publication server is not permitted to connect to the database at the network location given in the Add Database dialog box. Verify that the database host IP address, port number, database user name, password, and database identifier are correct. Verify there is an entry in the pg_hba.conf file permitting access to the database by the given user name originating from the IP address where the publication server is running.
Filter cannot be defined for Binary data type column(s)e.g. BYTEA, BLOB, RAW. *Resolution:* Occurs when attempting to define a filter rule on a column with a binary data type in a publication table. Filter rules are not permitted on such columns. See Section 2.2.12.3.
Filter with same name/clause already exist on table/view: *schema.table_name* *Resolution:* When adding a filter rule on a publication table, the same filter name or the same filter clause (WHERE clause) cannot be used more than once on a given table. Modify the duplicate filter name or filter clause so it is unique for the table.
The initial snapshot is not performed for this subscription. Please take the snapshot first and then proceed with the synchronize operation. *Resolution:* A snapshot replication must be performed before the first synchronization replication. Perform an on demand snapshot replication.
It is recommended to use a network IP address, the loopback address may result in connectivity issues. *Resolution:* This warning is given when localhost or 127.0.0.1 is specified as the host address of a replication system component. If is strongly recommended that all replication system components are identified by their specific IP address on the network.
The log triggers creation failed for one or more publication tables. Make sure the database is in valid state and user is granted the required privileges. *Resolution:* Either the user does not have the trigger creation privilege or there is a database server problem. The database server message is displayed as part of the error.
The MMR mode is currently not supported for *database_type* database. *Resolution:* A database server of type *database_type* cannot be used in a multi-master replication system. Only Advanced Server or PostgreSQL database servers may be used as master nodes in a multi-master replication system.
Multiple filters of same table are not allowed. *Resolution:* When creating a subscription in a single-master replication system or creating a master node other than the master definition node in a multi-master replication system, only one filter may be selected for a given table. Uncheck the additional boxes in the Apply column under the Filter Rules tab if more than one box is selected.
No JDBC Client driver is configured for the Oracle data source. *Resolution:* Occurs when creating an Oracle publication or subscription database definition. Copy the Oracle JDBC driver file ojdbc*x*.jar to subdirectory lib/jdbc of where the publication server or subscription server is installed on the host running the publication server or subscription server. Restart the

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
publication server or subscription server.
`No Publication found on MDN node, additional Master node cannot join MMR cluster.` *Resolution:* Occurs when attempting to add a second master node to a multi-master replication system, but no publication has been defined under the master definition node. Create a publication under the master definition node, then add the additional master nodes. See Section 6.2.3.
`None of the target master/subscription databases is accessible, hence the replication process failed to complete.` *Resolution:* Synchronization replication failed due to the unavailability of a target database. See the publication server log file for details. See Section 10.3.2.
`One or more master database node(s) are defined against this publication. Removing the publication will invalidate the MDN.` *Resolution:* Master nodes are still defined in a multi-master replication system in which an attempt is being made to delete the publication from the master definition node. All master nodes (other than the master definition node) must be deleted first before deleting the publication from the master definition node. Perform this deletion process with the xDB Replication Console or xDB Replication Server CLI.
`One or more subscriptions are defined against this publication. Removing the publication will invalidate the subscription. Do you want to continue?` *Resolution:* Warning issued when you attempt to remove a publication with subscriptions associated with it. You can remove the publication, but the subscriptions are no longer usable and should be removed as well.
`Multiple Publications creation is currently not supported.` *Resolution:* Only one publication is supported in a multi-master replication system and only one such multi-master replication system can exist for an xDB Replication Server installation.
`Only subscription which has subscribed against a publication with transactional replication type, can be synchronized.` *Resolution:* You cannot perform synchronization replication on a snapshot-only publication. Perform snapshot replication instead.
`The Oracle/MS SQL Server cannot be registered if the active Controller database is a non-PG/PPAS database.` *Resolution:* Occurs when creating an Oracle or SQL Server publication database definition and the current controller database is not a Postgres database (that is, the controller database is an Oracle or SQL Server database). In order to create an Oracle or SQL Server publication database, create and designate a Postgres publication database as the controller database. See Section 7.7.
`Parent table` *`table_name`* `is not selected when its child tables are part of the publication list.` *Resolution:* Table selected for a publication has a foreign key referencing a parent table that has not been chosen for the publication. This is only a warning that the parent table will not be part of the subscription.
`Problem occurred in publish process. Reason: Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.` *Resolution:* Occurs when attempting synchronization replication and the controller database is not accessible by the publication server. Verify that the correct IP address and port has been defined in the publication database definition of the controller database. Verify that the database server is running and is

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
accessible from the host running the publication server.
`Problem occurred in publish process. Reason: ERROR: permission denied for schema _edb_replicator_pub` *Resolution:* For a Postgres publication, verify that the publication database user has `CREATE ON DATABASE` privilege on the publication database, or the database user is a superuser.
`Publication cannot be created. One or more tables have no attributes defined and cannot be published. Unselect the specific tables and retry.` *Resolution:* In Postgres, it is possible to create a table with no columns. A publication is not allowed to include a Postgres table with no columns since the corresponding subscription table cannot be created in Oracle.
`Publication cannot be created. Publication` *publication_name* `already exists on the publisher server. Please choose a different name and then proceed.` *Resolution:* Publication names must be unique within a publication server. Enter a different publication name.
`Publication cannot be created. Table` *schema.table_name* `replica identity is set to` *replica_identity_setting*`. To define a Filter, the table replica identity should be set to FULL.` *Resolution:* Occurs when a table filter is attempted to be defined on a publication table used in a log-based replication system. Use the `ALTER TABLE` statement to change `REPLICA IDENTITY` to `FULL`. See Section 2.2.12.3.
`Publication cannot be created. Table` *table_name* `does not contain a primary key. Transactional replication is not supported for a non-pk table.` *Resolution:* All tables used for synchronization replication must have primary keys. Create a primary key on the table or add the table to a snapshot-only publication.
`Publication cannot be created. The publication creation process timed out as one or more tables may be locked by another session. Please retry later.` *Resolution:* For a Postgres publication that is not for snapshot-only, the publication database user must be able to create triggers on the publication tables. In order to do this, the publication database user must have the privilege to execute the `ALTER TABLE` statement on the publication tables and the publication database user must have `CREATE` and `USAGE` privileges on the schema containing the publication tables. Verify that one of the following is true: 1) All the tables in the publication are owned by the publication database user and the user has `CREATE` and `USAGE` privileges on the publication tables' schemas, or 2) the publication database user is a superuser.
`Publication cannot be removed. Reason: Publication` *publication_name* `cannot be removed. Reason: Error: cannot drop table _edb_replicator_pub.rrst_`*schema_table_name* `because other objects depend on it.` *Resolution:* PL/pgSQL custom conflict handler functions may exist in the master definition node that are dependent upon the publication's shadow tables. Drop the custom conflict handler functions before deleting the publication.
`Publication cannot be updated. Reason: The parent table` *schema.table_name* `is selected for removal while it has one or more child tables in the publication list. Make sure that parent-child dependency holds in the publication tables.` *Resolution:* Choose the child tables for removal as well as the parent table.
`Publication defined in MMR cluster cannot be subscribed in SMR cluster.` *Resolution:* A given publication cannot be used in both a multi-master replication system and a single-

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
master replication system.
`Publication does not exist on the publication server. It might have been removed.` *Resolution:* The publication does not exist for a given subscription. The subscription is no longer usable and must be removed.
`Publication having subscription against it, cannot be updated by removing tables from it.` *Resolution:* Remove the subscription, remove tables from the publication, then add the subscription.
`The publication schema cannot be created. Reason: ERROR: Permission denied for database` *db_name*. *Resolution:* Occurs when attempting to create the publication database definition and the specified publication database user does not have the privilege to create a schema in database *db_name*. Grant the `CREATE` privilege on the database to the publication database user
`Publication Service connection failure.` *Resolution:* Verify that the publication server is running. See Section 10.3.4.2. Verify that the database server hosting the controller database specified in the xDB Replication Configuration file is running and the publication server is connected to it. See Section 2.3.1.3.
`The replication process could not be completed due to a database failure. Check the database state and retry.` *Resolution:* May be caused by characters in the publication data that are illegal for the character set of the subscription database. Check the snapshot replication failure log file or the database server log file. See Section 10.4.1.2.
`Replication server does not support Oracle to Oracle replication.` *Resolution:* See Section 10.1.3 for supported database server configurations. Use Oracle products for Oracle to Oracle replication.
`A replication slot is not available on the target database server. Please configure the max_replication_slots GUC on the database server.` *Resolution:* Occurs when attempting to add a publication database definition with the log-based method of synchronization replication, and the `max_replication_slots` configuration parameter in the `postgresql.conf` file is not set to a large enough value to accommodate the additional database. Increase the value of the `max_replication_slots` parameter and restart the database server. See Section 2.2.10 for additional information.
`Subscription` *subscription_name* `already exists on the subscriber server. Please choose a different name and then proceed.` *Resolution:* Subscription names must be unique within a subscription server. Enter a different subscription name.
`Subscription` *subscription_name* `cannot be removed. Reason: Publication does not exist on the publication server.` *Resolution:* Warning issued if the subscription you are attempting to remove does not have an associated publication. You can still remove the subscription.
`Subscription database connection cannot be removed as one or more subscriptions are defined against it.` *Resolution:* You cannot remove a subscription database definition if there are subordinate subscriptions.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
Remove the subscriptions first.
`Subscription does not exist on the subscription service. It might have been removed by some other user.` *Resolution:* The Subscription node you are trying to select no longer represents an existing subscription. The subscription may have been removed by a concurrent xDB Replication Console or xDB Replication Server CLI session. Click the Refresh icon in the xDB Replication Console toolbar to display the current replication tree.
`Subscription Service connection failure.` *Resolution:* Verify that the subscription server is running. See Section 10.3.4.2
`'Synchronize Publication' process failed for one or more master nodes. Please see logs for more details.` *Resolution:* Synchronization replication failed to complete for all target databases in the multi-master replication system due to the unavailability of some target database. See the publication server log file for details. See Section 10.3.2.
`A table with large object type PK attribute cannot be published for (synchronize) incremental replication.` *Resolution:* Oracle doesn't log changes for a large object column. Such a column cannot be referenced in the triggers that log changes to the shadow tables. Use snapshot-only replication instead.

`Test result: Failure`
`Database connection information test failed. Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.`
 	
Resolution: Occurs when testing the connection of a publication or subscription database definition. The publication or subscription server cannot connect to the database server network location given in the Add Database dialog box. Verify that the correct IP address and port for the database server are given. Verify that the database server is running and is accessible from the host running the publication or subscription server.

`Test result: Failure`
`Database connection information test failed. FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`", user "`*`user_name`*`", database "`*`db_name`*`", SSL off`
 	
Resolution: Occurs when testing the connection of a publication or subscription database definition. The publication or subscription server is not permitted to connect to the database at the network location given in the Add Database dialog box. Verify that the database host IP address, port number, database user name, password, and database identifier are correct. Verify there is an entry in the `pg_hba.conf` file permitting access to the database by the given user name originating from the IP address where the publication or subscription server is running.

`Test result: Failure`
`Database connection information test failed. IO exception: The Network Adapter could not establish the connection.`
 	
Resolution: Verify that the database server is running. For Oracle, verify that the Oracle listener program `lsnrctl` is running.

`Test result: Failure`
`The target database server cannot be registered for WAL based logical replication. Reason: The database server is not configured for logical replication. Reason: FATAL: must be superuser or replication role to start walsender.`
 	
Resolution: Occurs when attempting to add a publication database definition with the log-based method

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
of synchronization replication (that is, WAL based logical replication), and the publication database user is not a superuser or does not have `REPLICATION` privilege. Grant the publication database user the appropriate privilege or specify a different database user who has the appropriate privilege for logical replication as the publication database user. See Section 2.2.10.

`Test result: Failure`
`The target database server cannot be registered for WAL based logical replication. Reason: The database server is not configured for logical replication. Reason: FATAL: no pg_hba.conf entry for replication connection from host "xxx.xxx.xx.xxx", user "user_name", SSL off`
 	
Resolution: Occurs when attempting to add a publication database definition with the log-based method of synchronization replication (that is, WAL based logical replication), and there is no entry in the `pg_hba.conf` file where the DATABASE field is set to `replication` for *user_name*. The `pg_hba.conf` file of the target database server must contain a replication entry for the publication database user name specified when creating the publication database definition. See Section 2.2.10.

`Test result: Failure`
`The target database server cannot be registered for WAL based logical replication. Reason: The database server is not configured for logical replication. Reason: FATAL: number of requested standby connections exceeds max_wal_senders (currently n)`
 	
Resolution: Occurs when attempting to add a publication database definition with the log-based method of synchronization replication (that is, WAL based logical replication), and the additional concurrent connection for logical replication exceeds the current setting, *n*, of the `max_wal_senders` configuration parameter in the `postgresql.conf` file. Increase the value of `max_wal_senders` in the `postgresql.conf` file of the database server running the publication database. Restart the database server containing the publication database. See Section 2.2.10.

`Test result: Failure`
`The target database server cannot be registered for WAL based logical replication. Reason: The target database server version x.x does not support WAL logical decoding.`
 	
Resolution: Occurs when attempting to create a publication database definition with the log-based method of synchronization replication (that is, WAL based logical replication), and the Postgres database server is not version 9.4 or later. Only Postgres database servers of version 9.4 or later support the log-based method of synchronization replication. See Section 2.2.10.

`Unable to apply DDL changes.` *Resolution:* The DDL statements in the text file specified for the DDL change replication feature contain syntax errors or are not supported by the DDL change replication feature. See Section 7.8.
`Unable to communicate with remote server.` *Resolution:* Occurs when attempting an operation such as performing synchronization replication or creating a schedule on a publication or subscription database that cannot be accessed by the xDB Replication Console. Verify that the publication and/or subscription servers are running. Verify that the database servers of the publication and/or subscription databases are running.

`Unable to create schema tables in target master database.`
`Unable to create publication shadow tables.`
`Unable to create subscription schema tables.`
`DB-42501: com.edb.util.PSQLException: ERROR: permission denied for relation pg_class.`
 	
Resolution: Occurs when attempting to create an MMR publication database definition and the publication server is unable to create the control schema objects in the new publication database. This typically results when creating a second publication database definition and the publication server is unable

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
to copy by snapshot the control schema objects from the controller database to the new publication database. The publication database user of the new publication database must be a superuser. In addition, in system catalog table `pg_catalog.pg_authid`, column `rolcatupdate` must be set to `true` for this superuser. See Section 10.4.4.
`Unable to create Subscription` *`subscription_name`*`. Reason: Connection rejected: FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`" user "`*`user_name`*`", database "`*`db_name`*`", SSL off` *Resolution:* Occurs when creating a subscription. The subscription server running on host *xxx.xxx.xx.xxx* could not access the controller database. Verify that the `pg_hba.conf` file on the controller database server permits access from the subscription server host
`Unable to create subscription schema tables. Org.postgresql.util.PSQLException: FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`" user "`*`user_name`*`", database "`*`db_name`*`", SSL off` *Resolution:* Occurs when creating a subscription. The subscription server running on host *xxx.xxx.xx.xxx* could not access the publication database. Verify that the `pg_hba.conf` file on the publication database server permits access from the subscription server host.
`Unable to create subscription schema tables. The database type is not supported.` *Resolution:* The subscription database type is not supported for the intended publication database type. See Section 10.1.3.2 for a list of permitted source and target database server configurations.
`Unable to create subscription schema tables. The target database schema already contains one or more tables with the same name as the table(s) in the source database.` *Resolution:* The subscription server was unable to create a subscription table definition in the intended target schema. Typically, the reason is that a table with the same name already exists in the target schema of the subscription database. This can occur if you create a subscription, then remove it, but fail to drop the table definitions created under the target schema, then try to create the subscription a second time.

`Unable to create subscription schema tables.`
`Unable to create publication shadow tables.`
`Unable to create subscription schema tables.`
`DB-42501: com.edb.util.PSQLException: ERROR: permission denied for relation pg_class.`
 	
Resolution: Occurs when attempting to create an SMR publication database definition and the publication server is unable to create the control schema objects in the new publication database. This typically results when creating a second publication database definition and the publication server is unable to copy by snapshot the control schema objects from the controller database to the new publication database. The publication database user of the new publication database must be a superuser. In addition, in system catalog table `pg_catalog.pg_authid`, column `rolcatupdate` must be set to `true` for this superuser. See Section 10.4.4.

`Unable to perform snapshot for subscription` *`subscription_name`*`. Reason: DB-42501: com.edb.util.PSQLException: ERROR: permission denied for relation pg_class.` *Resolution:* Occurs when attempting a snapshot replication. The database user of the database receiving the snapshot must be a superuser. In addition, in system catalog table `pg_catalog.pg_authid`, column `rolcatupdate` must be set to `true` for this superuser. See Section 10.4.4.
`Unable to perform snapshot for subscription` *`subscription_name`*`. Reason: org.postgresql.util.PSQLException: FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`", user "`*`user_name`*`", database "`*`db_name`*`", SSL off`

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Error Messages and Resolutions
Resolution: Occurs when attempting a snapshot replication. The publication server running on host *xxx.xxx.xx.xxx* could not access the subscription database. Verify that the `pg_hba.conf` file on the subscription database server permits access from the publication server host.
`Unable to synchronize. Reason: FATAL: no pg_hba.conf entry for host "`*`xxx.xxx.xx.xxx`*`", user "`*`user_name`*`", database "`*`db_name`*`", SSL off` *Reason:* Occurs during an implicit synchronization following snapshot replication. The publication server running on host *xxx.xxx.xx.xxx* could not access the subscription server's controller database. Verify that the `pg_hba.conf` file on the subscription server permits access from the publication server host using network address *xxx.xxx.xx.xxx*.
`Unable to update publication database information. Reason: Publication control schema does not exist on target database.` *Resolution:* The control schema objects in the publication database may have been deleted or corrupted. For an Oracle publication database the control schema objects are located in the publication database user's schema. For a Postgres or SQL Server publication database the metadata database objects are located in schemas `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`. See Section 10.3.4.3.
`The user has insufficient privileges to manage publications. Grant required privileges as listed below and then proceed with operation.` *Resolution:* An Oracle publication database user must have `CONNECT`, `RESOURCE`, and `CREATE ANY TRIGGER` privileges.

10.3.2 Where to Look for Errors

There are a number of places to look to find more detailed information about a replication error that may have occurred. This section provides a guide as to where to look for various types of errors.

10.3.2.1 General Replication Status

In the xDB Replication Console, view the replication history. See Section 7.4.

10.3.2.2 Snapshot Replication Failures

View the log file found in the following path:

For Linux:

```
/var/log/xdb-x.x/mtk.log
```

For Windows:

```
POSTGRES_HOME\.enterprisedb\xdb\x.x\mtk.log
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

POSTGRES_HOME is the home directory of the Windows `postgres` account (`enterprisedb` account for Advanced Server installed in Oracle compatible configuration mode). The specific location of *POSTGRES_HOME* is dependent upon your version of Windows. The xDB Replication Server version number is represented by *x.x*.

See Section 10.4.1.1 for more information on setting log file options.

10.3.2.3 Synchronization Replication Failures

Check the database server log file.

The typical default location of these files is:

```
POSTGRES_INSTALL_HOME/data/pg_log
```

10.3.2.4 Publication and Subscription Server Startup Failures

View the publication server and subscription server log files `pubserver.log[.n]` and `subserver.log[.n]` in the following directory:

For Linux:

```
/var/log/xdb-x.x
```

For Windows:

```
POSTGRES_HOME\.enterprisedb\xdb\x.x
```

`[.n]` is an optional, integer suffix whose presence depends upon the `logging.file.count` configuration option described in Section 10.4.1.1.

POSTGRES_HOME is the home directory of the Windows `postgres` account (`enterprisedb` account for Advanced Server installed in Oracle compatible configuration mode). The specific location of *POSTGRES_HOME* is dependent upon your version of Windows. The xDB Replication Server version number is represented by *x.x*.

Note: The severity level of messages logged in these files can be controlled by a configuration option. See Section 10.4.1.1.

For Linux only: View the publication service and subscription service startup log files `edb-xdbpubserver.log` and `edb-xdbsubserver.log` as well as the service script log files `edb-xdbpubserver_script.log` and `edb-xdbsubserver_script.log` in directories `/var/log/edb/xdbpubserver` and `/var/log/edb/xdbsubserver`. These log files contain the output from the scripts used to start the publication server and subscription server, and can typically be used to confirm the port number on which the publication and subscription servers were started.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Note: The publication service and subscription service startup log files are not generated for Windows and Mac OS X operating systems.

If there is an entry for a controller database in the xDB Replication Configuration file, verify that this controller database is accessible with the designated connection information. The controller database parameters are `host`, `port`, `type`, `user`, and `password`.

The following is an example of the content of an xDB Replication Configuration file with an Oracle database as the controller database:

```
#xDB Replication Server Configuration Properties
#Tue May 26 13:45:37 GMT-05:00 2015
port=1521
admin_password=ygJ9AxoJEX854elcVIJPTw\=\=
user=pubuser
admin_user=admin
type=oracle
password=ygJ9AxoJEX854elcVIJPTw\=\=
database=xe
host=192.168.2.23
```

See Section 2.3.1.3 for information on the xDB Replication Configuration file.

Also check the database server log file of the controller database.

10.3.2.5 Database Server Errors

Check the database server log file.

The typical default location of these files is:

POSTGRES_INSTALL_HOME/data/pg_log

10.3.2.6 Oracle Errors

For problems in Oracle, first find the directory locations of the log files by issuing the following commands in SQL*Plus:

```
SQL> SHOW PARAMETER USER_DUMP_DEST;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
user_dump_dest                       string      /usr/lib/oracle/xe/app/oracle/
                                                 admin/XE/udump
```

The directory given by parameter `USER_DUMP_DEST` contains errors given by user processes.

```
SQL> CONNECT system/password
Connected.
SQL> SHOW PARAMETER BACKGROUND_DUMP_DEST;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
background_dump_dest                 string      /usr/lib/oracle/xe/app/oracle/
                                                 admin/XE/bdump
```

The directory given by parameter `BACKGROUND_DUMP_DEST` contains errors given by the Oracle background processes.

Find the latest log file in the preceding directories to investigate the problem.

10.3.3 Common Problem Checklist

Use the following checklist to verify that the proper configuration steps have been followed. Omission of one or more of these steps is a common source of errors.

Step 1: Verify that the database server of the publication database, the database server of the subscription database (for single-master replication systems), and the database servers of the master nodes (for multi-master replication systems) are all running.

Step 2: When viewing information in the xDB Replication Console, click the Refresh icon in the toolbar to ensure you are viewing the most current information, especially after making a configuration change to your replication system.

Step 3: Verify that the publication server and the subscription server (for single-master replication systems) are running. If they are not running and cannot be started see Section 10.3.4.2.

Step 4: If you are using an Oracle publication or subscription database, verify that the Oracle JDBC driver file has been copied to the *`XDB_HOME`*`/lib/jdbc` directory. *`XDB_HOME`* is the location where you installed xDB Replication Server.

See Section 5.1.3.1.

Step 5: Verify that the necessary privileges have been granted to the publication database user.

For an Oracle publication database, verify that the publication database user has `CONNECT`, `RESOURCE`, and `CREATE ANY TRIGGER` privileges.

See Section 5.1.4.1.

For a SQL Server publication database, verify the following:

- In the `msdb` database, verify that the database user mapped to the SQL Server login given in the publication database definition has `EXECUTE` and `SELECT` privileges on schema `dbo`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- In the publication database, verify that the database user mapped to the SQL Server login given in the publication database definition has its default schema set to the schema containing the xDB Replication Server metadata database objects.
- For the same database user discussed in the prior paragraph, verify that this database user is either the owner of the schema containing the xDB Replication Server metadata database objects, or has the following privileges on this schema: `ALTER`, `EXECUTE`, `SELECT`, `INSERT`, `UPDATE`, and `DELETE`.
- For the same database user discussed in the prior paragraph, verify that this database user has `CREATE TABLE` and `CREATE PROCEDURE` privileges.
- For the same database user discussed in the prior paragraph, verify that this database user has `ALTER` privilege on the publication tables.
- For any database user that will be updating the publication tables, verify that these database users have `EXECUTE`, `SELECT`, and `INSERT` privileges on the schema containing the xDB Replication Server metadata database objects.

See Section 5.1.4.2.

For a Postgres publication database in a single-master replication system, verify that the publication database user is a superuser and has the privilege to modify `pg_catalog` tables.

See Section 5.1.4.3.

For the master definition node in a multi-master replication system, verify that the publication database user is a superuser and has the privilege to modify `pg_catalog` tables.

See Section 6.1.3.

For a master node other than the master definition node in a multi-master replication system, verify that the master node database user is a superuser and has the privilege to modify `pg_catalog` tables.

See Section 6.1.4.

Step 6: Verify that the necessary privileges have been granted to the subscription database user.

For an Oracle subscription database, verify that the subscription database user has `CONNECT` and `RESOURCE` privileges.

For a Postgres subscription database, verify that the subscription database user is a superuser and has the privilege to modify `pg_catalog` tables.

See Section 5.1.5.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 7 (For Linux only): Verify that the network IP address returned by the `/sbin/ifconfig` command either matches the IP address associated with the host name in the `/etc/hosts` file (see Section 5.1.6.2), or matches the IP address specified with the `java.rmi.server.hostname` configuration option in the publication and subscription server configuration files (see Section 10.4.1.7).

10.3.4 Troubleshooting Areas

The following topics provide information on specific problem areas you may encounter.

10.3.4.1 Java Runtime Errors

If errors are encountered regarding the Java Runtime Environment such as the Java program cannot be found or Java heap space errors, check the parameters set in the xDB Startup Configuration file `xdbReplicationServer-xx.config`. See Section 2.3.1.4 for information on the xDB Startup Configuration file.

The following is an example of the content of the xDB Startup Configuration file:

```
#!/bin/sh

JAVA_EXECUTABLE_PATH="/usr/bin/java"
JAVA_MINIMUM_VERSION=1.7
JAVA_BITNESS_REQUIRED=64
JAVA_HEAP_SIZE="-Xms256m -Xmx1536m"
PUBPORT=9051
SUBPORT=9052
```

If you make any changes to the parameters in the xDB Startup Configuration file, be sure to restart the publication server and subscription server after making the modifications.

10.3.4.2 Starting the Publication Server or Subscription Server

Note: The subscription server only applies to single-master replication systems.

If you cannot start the publication server or the subscription server perform the following steps:

Step 1: Check the `pubserver.log` and `subserver.log` files for errors.

Step 2: Check the log file of the database server running the controller database for errors.

Step 3: Verify that the user name and password in the xDB Replication Configuration file on the hosts running the publication server and subscription server match a database user name and password in the database server running the controller database that the publication server and subscription server are attempting to access.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 4: If the controller database is a Postgres database, verify that the `pg_hba.conf` file of its Postgres database server has entries that allow access to the controller database from the IP addresses of the hosts running the publication server and subscription server by the user name in the xDB Replication Configuration file.

10.3.4.3 Deleting the Control Schema and Control Schema Objects

The control schema completely describes the replication system. The control schema and its control schema objects must be complete and correct in order for replication to occur properly. In addition, the configuration and maintenance operations performed through the xDB Replication Console or the xDB Replication Server CLI cannot be accomplished properly unless the control schema is complete and correct.

There may be occasions where the control schema becomes corrupted. Either one or more control schema tables containing metadata are inadvertently deleted, or the data within the control schema tables becomes corrupted. Typically, corruption occurs in the form of the first case – one or more control schema tables were deleted, or the entire control schema and its contents were deleted manually using an SQL utility rather than through the operation of the xDB Replication Console or xDB Replication Server CLI.

In these situations, there may be no other choice than to remove all of the remaining control schema objects using the database management system's deletion functions, which effectively deletes all replication systems managed by the control schema.

The same control schema deletion procedure must be performed in all publication databases that share the same control schema information as the current controller database given in the xDB Replication Configuration file.

From the viewpoint of the xDB Replication Console replication tree, a publication server that connects to the controller database has subordinate to it, the publication databases sharing the same control schema information.

In the following example, the SMR publication database `edb` as well as the three MMR master node databases `mdnnode`, `mmrnode_a`, and `mmrnode_b` are all managed by the same publication server, which connects to the controller database designated in the xDB Replication Configuration file. Thus, all publication databases `edb`, `mdnnode`, `mmrnode_a`, and `mmrnode_b` contain what should be the same control schema information.

The control schema must be removed from all four publication databases if it is determined that the control schema is corrupted in any of the four publication databases.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

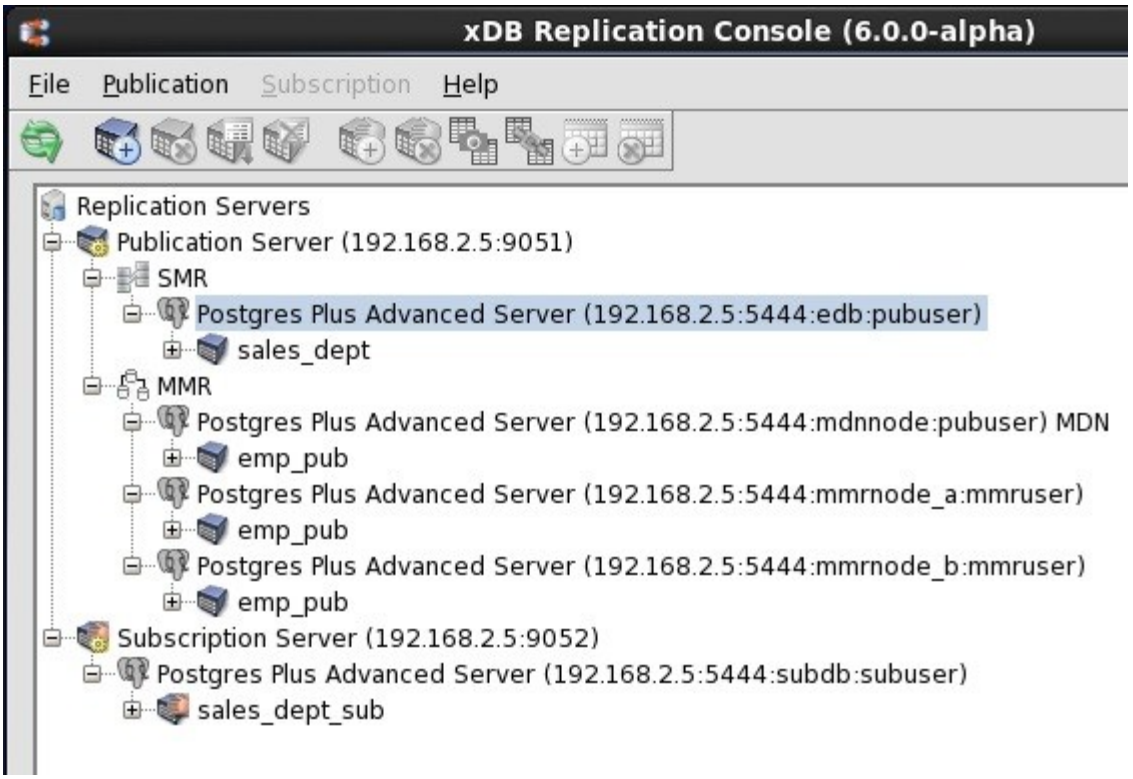


Figure 10-4 - Publication databases with shared controller database

Finally, the subscription databases of SMR systems contain a control schema object, which must be deleted as well.

In the preceding example, subscription database `subdb` contains a control schema object that may have to be deleted if control schema deletion is performed on the publication database.

The instructions in this section describe how to completely remove all control schema objects created by the xDB Replication Server product leaving just your original publication tables and any replicated subscription tables or publication tables of multi-master system nodes. Hence, the definition and framework for all existing single-master and multi-master replication systems are deleted. In effect, this simulates the situation when you have installed the xDB Replication Server product for the first time.

After you have performed this deletion process, single-master replication systems must then be recreated following the directions in sections 5.2 onward. A multi-master replication system must be recreated following the directions in sections 6.2 onward.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Warning: Do not attempt this if any replication systems are running in production. All replication systems will become inoperable.

This section describes what to look for in order to tell if the control schema is not complete, and if so, what must be deleted to completely remove the replication system. This section does not discuss the internal contents of the control schema objects. If all of the control schema objects are present, then review the checklist in Section 10.3.3 before proceeding with deletion of the control schema as it is fairly unlikely that the content of a control schema table becomes corrupted.

If you decide that you must delete all of the control schema objects, follow the steps as discussed in the following:

Step 1: Stop the publication server.

Step 2: Stop the subscription server.

Step 3: Look for the control schema objects contained within a publication database. In the example used in this section, `pubuser` is the publication database user name. The publication consists of two tables – `dept` and `emp`.

For Oracle only: See Section 5.2.4.1 for a list of Oracle control schema objects.

For SQL Server only: See Section 5.2.4.2 for a list of SQL Server control schema objects.

For Postgres only: See Section 5.2.4.3 for a list of Postgres control schema objects.

Step 4: If the schema that is supposed to contain the control schema objects (the publication database user name for Oracle, or the control schema you created or selected when configuring a SQL Server publication database along with `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`, or `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler` for Postgres) is missing, or there are missing database objects under the control schema, then you may need to complete the process of removing all remaining control schema objects.

If you decide to undergo this procedure, you must remove the control schema objects from all publication databases. You must also remove all subscription metadata objects from the subscription databases. Proceed with Step 7 and repeat Step 7 for all publication databases. Then proceed with Step 8 and repeat Step 8 for all subscription databases.

If the control schema objects look intact, repeat Step 3 for all other publication databases. If the control schema objects of all publication databases appear intact, then proceed with Step 5.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 5: For single-master replication systems, the subscription database contains a single control schema object in the form of a table named `rrep_txset_health`. See Section 5.3.4 for a listing of this control schema object for each type of subscription database.

For each subscription database, verify the presence of this subscription metadata object.

Step 6: If at this point, all control schemas and control schema objects appear intact in all publication databases and all subscription databases, then chances are that the problem lies elsewhere. Do not go proceed with any further steps in this section. Instead, recheck the checklist in Section 10.3.3.

If it was determined that incomplete control schema objects exist, and you decide to go ahead with the deletion process, proceed with Step 7.

Step 7: Repeat this step for every publication database to delete its control schema and control schema objects.

For Oracle only: If the publication user name still exists, then log onto SQL*Plus or any other Oracle database administration utility and drop all control schema objects owned by the publication user. Alternatively, you can drop the publication database user along with its database objects using the cascade option, but the publication database user must be recreated and privileges reassigned if you intend to rebuild your replication systems. See Section 5.1.4 for directions on creating the publication database user. The following example illustrates use of the cascade option:

```
SQL> CONNECT system/password
Connected.
SQL> DROP USER pubuser CASCADE;

User dropped.
```

For SQL Server only: If any of the control schema objects listed in Step 3 still exist, then log onto the SQL Server command line program, `sqlcmd`, or SQL Server Management Studio and drop these objects. The following example assumes some of the control schema objects were created under schema `pubuser`. The other control schema objects are created under `_edb_replicator_pub`, `_edb_replicator_sub`, and `_edb_scheduler`. The publication tables are `dept` and `emp` located in schema `edb`.

The following example shows how to delete the jobs in the `msdb` database:

```
1> USE msdb;
2> GO
Changed database context to 'msdb'.
1> EXEC sp_delete_job @job_name = 'rrep_cleanup_job_edb';
2> GO
1> EXEC sp_delete_job @job_name = 'rrep_txset_job_edb';
2> GO
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The next example shows the deletion of the triggers on the non-snapshot only publication tables:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP TRIGGER edb.rrpd_edb_dept;
2> DROP TRIGGER edb.rrpi_edb_dept;
3> DROP TRIGGER edb.rrpu_edb_dept;
4> DROP TRIGGER edb.rrpd_edb_emp;
5> DROP TRIGGER edb.rrpi_edb_emp;
6> DROP TRIGGER edb.rrpu_edb_emp;
7> GO
```

The control schema objects under the `_edb_replicator_pub` schema are dropped as shown by the following:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP TABLE _edb_replicator_pub.rrep_lock;
2> DROP TABLE _edb_replicator_pub.rrep_mmr_pub_group;
3> DROP TABLE _edb_replicator_pub.rrep_mmr_txset;
4> DROP TABLE _edb_replicator_pub.rrep_properties;
5> DROP TABLE _edb_replicator_pub.rrep_publication_subscriptions;
6> DROP TABLE _edb_replicator_pub.rrep_publication_tables;
7> DROP TABLE _edb_replicator_pub.rrep_tables;
8> DROP TABLE _edb_replicator_pub.rrep_tx_monitor;
9> DROP TABLE _edb_replicator_pub.rrep_txset;
10> DROP TABLE _edb_replicator_pub.rrep_txset_health;
11> DROP TABLE _edb_replicator_pub.rrep_txset_log;
12> DROP TABLE _edb_replicator_pub.xdb_cleanup_conf;
13> DROP TABLE _edb_replicator_pub.xdb_conflicts;
14> DROP TABLE _edb_replicator_pub.xdb_conflicts_options;
15> DROP TABLE _edb_replicator_pub.xdb_events;
16> DROP TABLE _edb_replicator_pub.xdb_events_status;
17> DROP TABLE _edb_replicator_pub.xdb_mmr_pub_group;
18> DROP TABLE _edb_replicator_pub.xdb_pub_database;
19> DROP TABLE _edb_replicator_pub.xdb_pub_table_replog;
20> DROP TABLE _edb_replicator_pub.xdb_pub_replog;
21> DROP TABLE _edb_replicator_pub.xdb_publication_filter;
22> DROP TABLE _edb_replicator_pub.xdb_publication_filter_rule;
23> DROP TABLE _edb_replicator_pub.xdb_publication_subscriptions;
24> DROP TABLE _edb_replicator_pub.xdb_publications;
25> DROP TABLE _edb_replicator_pub.xdb_pubtables_ignoredcols;
26> DROP TABLE _edb_replicator_pub.xdb_sub_servers;
27> GO
```

For SQL Server 2008 only: Drop the following control schema objects when the publication database is SQL Server 2008:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP PROCEDURE _edb_replicator_pub.nextval;
2> DROP PROCEDURE _edb_replicator_pub.sp_createsequence;
3> DROP PROCEDURE _edb_replicator_pub.sp_dropsequence;
4> DROP TABLE _edb_replicator_pub.rrep_common_seq;
5> DROP TABLE _edb_replicator_pub.rrep_tx_seq;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
6> DROP TABLE _edb_replicator_pub.rrep_txset_seq;
7> GO
```

For SQL Server 2012, 2014 only: Drop the following control schema objects when the publication database is SQL Server 2012 or 2014:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP SEQUENCE _edb_replicator_pub.rrep_tx_seq;
2> DROP SEQUENCE _edb_replicator_pub.rrep_txset_seq;
3> DROP SEQUENCE _edb_replicator_pub.rrep_common_seq;
4> GO
```

Drop the `_edb_replicator_pub` control schema:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP SCHEMA _edb_replicator_pub;
2> GO
```

The control schema objects under the `_edb_replicator_sub` schema as well as the schema itself are dropped as shown by the following.

Note (For SQL Server 2012, 2014): When the publication database is SQL Server 2012 or 2014, the first table in the following list, `rrep_common_seq`, does not exist. Therefore do not issue the first `DROP TABLE _edb_replicator_sub.rrep_common_seq` command.

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP TABLE _edb_replicator_sub.rrep_common_seq;
2> DROP TABLE _edb_replicator_sub.xdb_sub_database;
3> DROP TABLE _edb_replicator_sub.xdb_subscription_tables;
4> DROP TABLE _edb_replicator_sub.xdb_subscriptions;
5> DROP TABLE _edb_replicator_sub.xdb_tables;
6> DROP SCHEMA _edb_replicator_sub;
7> GO
```

The control schema objects under the `_edb_scheduler` schema as well as the schema itself are dropped as shown by the following:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP TABLE _edb_scheduler.sch_pub_BLOB_TRIGGERS;
2> DROP TABLE _edb_scheduler.sch_pub_CALENDARS;
3> DROP TABLE _edb_scheduler.sch_pub_CRON_TRIGGERS;
4> DROP TABLE _edb_scheduler.sch_pub_SIMPLE_TRIGGERS;
5> DROP TABLE _edb_scheduler.sch_pub_TRIGGER_LISTENERS;
6> DROP TABLE _edb_scheduler.sch_pub_FIRED_TRIGGERS;
7> DROP TABLE _edb_scheduler.sch_pub_TRIGGERS;
8> DROP TABLE _edb_scheduler.sch_pub_JOB_LISTENERS;
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
9> DROP TABLE _edb_scheduler.sch_pub_JOB_DETAILS;
10> DROP TABLE _edb_scheduler.sch_pub_LOCKS;
11> DROP TABLE _edb_scheduler.sch_pub_PAUSED_TRIGGER_GRPS;
12> DROP TABLE _edb_scheduler.sch_pub_SCHEDULER_STATE;
13> DROP TABLE _edb_scheduler.sch_sub_BLOB_TRIGGERS;
14> DROP TABLE _edb_scheduler.sch_sub_CALENDARS;
15> DROP TABLE _edb_scheduler.sch_sub_CRON_TRIGGERS;
16> DROP TABLE _edb_scheduler.sch_sub_SIMPLE_TRIGGERS;
17> DROP TABLE _edb_scheduler.sch_sub_TRIGGER_LISTENERS;
18> DROP TABLE _edb_scheduler.sch_sub_FIRED_TRIGGERS;
19> DROP TABLE _edb_scheduler.sch_sub_TRIGGERS;
20> DROP TABLE _edb_scheduler.sch_sub_JOB_LISTENERS;
21> DROP TABLE _edb_scheduler.sch_sub_JOB_DETAILS;
22> DROP TABLE _edb_scheduler.sch_sub_LOCKS;
23> DROP TABLE _edb_scheduler.sch_sub_PAUSED_TRIGGER_GRPS;
24> DROP TABLE _edb_scheduler.sch_sub_SCHEDULER_STATE;
25> DROP SCHEMA _edb_scheduler;
26> GO
```

The control schema objects under the `pubuser` schema are dropped as shown by the following:

```
1> USE edb;
2> GO
Changed database context to 'edb'.
1> DROP FUNCTION pubuser.getPackageVersionNumber;
2> DROP PROCEDURE pubuser.CleanupShadowTables;
3> DROP PROCEDURE pubuser.ConfigureCleanUpJob;
4> DROP PROCEDURE pubuser.ConfigureCreateTxSetJob;
5> DROP PROCEDURE pubuser.CreateMultiTxSet;
6> DROP PROCEDURE pubuser.CreateTableLogTrigger;
7> DROP PROCEDURE pubuser.CreateTxSet;
8> DROP PROCEDURE pubuser.CreateTxSet_old;
9> DROP PROCEDURE pubuser.CreateUniTxSet;
10> DROP PROCEDURE pubuser.GetNewTxsCount;
11> DROP PROCEDURE pubuser.JobCleanup;
12> DROP PROCEDURE pubuser.JobCreateTxSet;
13> DROP PROCEDURE pubuser.LoadPubTableList;
14> DROP PROCEDURE pubuser.RemoveCleanupJob;
15> DROP PROCEDURE pubuser.RemoveCreateTxSetJob;
16> DROP TABLE pubuser.rrst_edb_dept;
17> DROP TABLE pubuser.rrst_edb_emp;
18> GO
```

For Postgres only: If any of the schemas `_edb_replicator_pub`, `_edb_replicator_sub`, or `_edb_scheduler` still exist in the publication database, drop the schema and all of its database objects. The following example shows a connection established in `psql` to the publication database `edb`. The `DROP SCHEMA CASCADE` statement is then used to drop the schemas.

```
edb=# \c edb enterprisedb
You are now connected to database "edb" as user "enterprisedb".

edb=# DROP SCHEMA _edb_replicator_pub CASCADE;
NOTICE:  drop cascades to 51 other objects
DETAIL:  drop cascades to sequence _edb_replicator_pub.rrep_common_seq
drop cascades to sequence _edb_replicator_pub.rrep_tx_seq
            .
            .
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
            .
DROP SCHEMA

edb=# DROP SCHEMA _edb_replicator_sub CASCADE;
NOTICE:  drop cascades to 9 other objects
DETAIL:  drop cascades to sequence _edb_replicator_sub.rrep_common_seq
drop cascades to table _edb_replicator_sub.xdb_sub_database
            .
            .
            .
DROP SCHEMA

edb=# DROP SCHEMA _edb_scheduler CASCADE;
NOTICE:  drop cascades to 40 other objects
DETAIL:  drop cascades to table _edb_scheduler.sch_pub_job_details
drop cascades to table _edb_scheduler.sch_pub_job_listeners
            .
            .
            .
DROP SCHEMA
```

For synchronization replication with the trigger-based method, in the schema containing the publication tables, drop the triggers and trigger functions associated with the publication tables:

```
edb=# SET search_path TO edb;
SET
edb=# DROP FUNCTION rrpd_edb_dept_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpd_edb_dept on table dept
DROP FUNCTION
edb=# DROP FUNCTION rrpi_edb_dept_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpi_edb_dept on table dept
DROP FUNCTION
edb=# DROP FUNCTION rrpu_edb_dept_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpu_edb_dept on table dept
DROP FUNCTION
edb=# DROP FUNCTION rrpd_edb_emp_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpd_edb_emp on table emp
DROP FUNCTION
edb=# DROP FUNCTION rrpi_edb_emp_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpi_edb_emp on table emp
DROP FUNCTION
edb=# DROP FUNCTION rrpu_edb_emp_tgfunc() CASCADE;
NOTICE:  drop cascades to trigger rrpu_edb_emp on table emp
DROP FUNCTION
```

Step 8: Repeat this step for every subscription database to delete its control schema and control schema object.

For single-master replication systems, the subscription database contains a single control schema object in the form of a table named `rrep_txset_health`.

Delete this table in all subscription databases. For SQL Server and Postgres subscription databases, delete the parent schema `_edb_replicator_sub` as well. For Oracle subscription databases, the parent schema is not generated by xDB Replication Server, so it your decision as to whether to keep or delete the parent schema.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For Oracle only: The `RREP_TXSET_HEALTH` table is created in the subscription database user's schema. Drop this table.

```
SQL> CONNECT subuser/password
Connected.
SQL> DROP TABLE rrep_txset_health;

Table dropped.
```

For SQL Server only: The `rrep_txset_health` table is created in the schema named `_edb_replicator_sub`. Drop this table and schema.

```
1> USE subdb;
2> GO
Changed database context to 'subdb'.
1> DROP TABLE _edb_replicator_sub.rrep_txset_health;
2> GO
1> DROP SCHEMA _edb_replicator_sub;
2> GO
```

For Postgres only: The `rrep_txset_health` table is created in the schema named `_edb_replicator_sub`. Drop this table and schema.

```
edb=# \c subdb enterprisedb
You are now connected to database "subdb" as user "enterprisedb".
subdb=# DROP SCHEMA _edb_replicator_sub CASCADE;
NOTICE:  drop cascades to table _edb_replicator_sub.rrep_txset_health
DROP SCHEMA
```

Step 9: In the xDB Replication Configuration file, delete the lines containing the following parameters: `user`, `password`, `host`, `port`, `database`, and `type`.

Keep the lines with the following parameters: `admin_user`, `admin_password`, and `license_key` (if it exists).

See Section 2.3.1.3 for information on the xDB Replication Configuration file. See Section 3.5 for the file system location of the xDB Replication Configuration file.

The absence of these parameters prevents the publication server and subscription server from attempting to connect to this database upon publication and subscription server startup.

The xDB Replication Configuration file should appear as follows without the controller database connection and authentication information:

```
#xDB Replication Server Configuration Properties
#Fri Jan 30 17:34:06 GMT-05:00 2015
admin_password=ygJ9AxoJEX854elcVIJPTw\=\=
admin_user=enterprisedb
```

Step 10: Start the publication server.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 11: Start the subscription server.

Step 12: In the replication tree you should see the following:

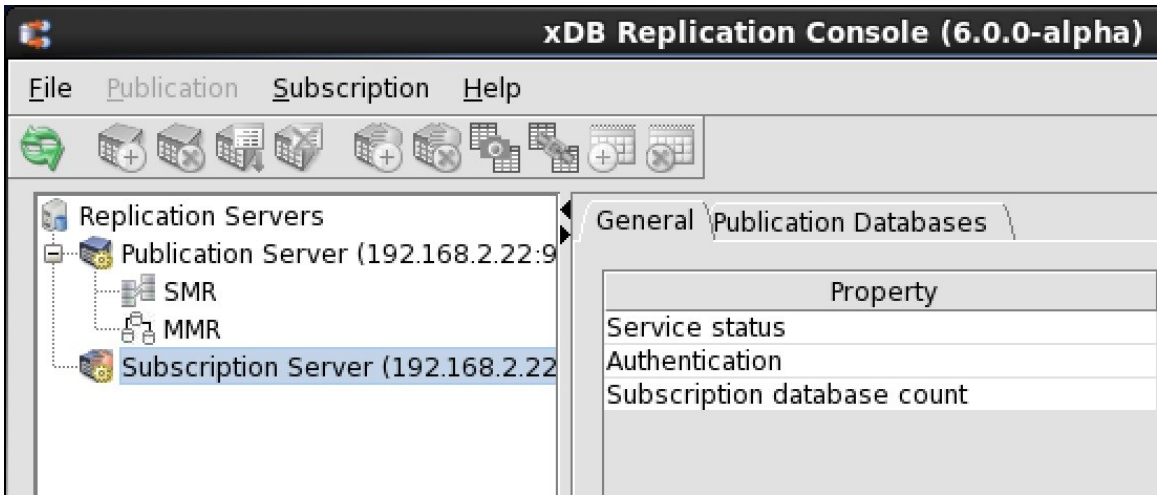


Figure 10-5 - Replication tree after removal of all control schema objects

All the nodes under the SMR and MMR type nodes beneath the Publication Server node, and under the Subscription Server node no longer appear.

Step 13: You will need to recreate the replication system as described in sections 5.2 onward for a single-master replication system. See sections 6.2 onward for a multi-master replication system.

10.3.4.4 Dropping Replication Slots for Log-Based Synchronization Replication

As described in Section 2.2.10.2 logical replication slots are used for the log-based method of synchronization replication. While a log-based replication system is in use, these replication slots remain connected to the Postgres databases. When the replication system is removed, these replication slots are also deleted.

There may be circumstances where it is desired to drop a Postgres database used in a replication system, but the replication system could not be removed according to the normal procedure of using the xDB Replication Console or the xDB Replication Server CLI.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

In such cases, it is assumed that the replication system has somehow become corrupted, and it is simply desired to delete the replication system components including some of the databases used in the replication system.

When the log-based method is used, certain additional procedures may be required to remove the replication slots before dropping the databases.

Postgres does not permit a database to be dropped if a replication slot is connected to it. The following describes how the replication slots can be removed in order to drop a database.

Warning: Do not attempt this if any replication systems are running in production. All replication systems will become inoperable.

Replication slots can be displayed by the following query on the database server containing the databases to be dropped:

```
edb=# SELECT slot_name, slot_type, database, active, active_pid FROM
pg_replication_slots;
  slot_name  | slot_type | database | active | active_pid
-------------+-----------+----------+--------+------------
 xdb_14793_5 | logical   | edb      | t      |       5288
 xdb_79910_5 | logical   | mmrnode  | t      |       5327
(2 rows)
```

The `active` column indicates whether or not the replication slot is active.

To deactivate an active replication slot, first stop the publication server. If the `active` column of the replication slot now displays `f` for false then you can remove the replication slot.

If the replication slot is still active, then you can deactivate it by terminating the process shown in the `active_pid` column with the following command:

```
edb=# SELECT pg_terminate_backend(5327);
 pg_terminate_backend
----------------------
 t
(1 row)
```

The following now shows that replication slot `xdb_79910_5` for database `mmrnode` has been deactivated:

```
edb=# SELECT slot_name, slot_type, database, active, active_pid FROM
pg_replication_slots;
  slot_name  | slot_type | database | active | active_pid
-------------+-----------+----------+--------+------------
 xdb_14793_5 | logical   | edb      | t      |       5288
 xdb_79910_5 | logical   | mmrnode  | f      |
(2 rows)
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Drop the replication slot with the following command by specifying the slot name:

```
edb=# SELECT pg_drop_replication_slot('xdb_79910_5');
 pg_drop_replication_slot
--------------------------

(1 row)
```

Now, the dropped replication slot does not appear when the `pg_replication_slots` directory is queried:

```
edb=# SELECT slot_name, slot_type, database, active, active_pid FROM
pg_replication_slots;
  slot_name   | slot_type | database | active | active_pid
--------------+-----------+----------+--------+------------
 xdb_14793_5 | logical   | edb      | t      |       5288
(1 row)
```

The database can now be successfully dropped:

```
edb=# DROP DATABASE mmrnode;
DROP DATABASE
```

In addition, replication origins can be displayed with the following command:

```
edb=# SELECT * FROM pg_replication_origin;
 roident |        roname
---------+----------------------
       1 | xdb_mmrnode_emp_pub_1
       2 | xdb_edb_emp_pub_6
(2 rows)
```

The following command can be used to remove a replication origin:

```
edb=# SELECT pg_replication_origin_drop('xdb_mmrnode_emp_pub_1');
 pg_replication_origin_drop
----------------------------

(1 row)
```

The following shows this replication origin has been removed:

```
edb=# SELECT * FROM pg_replication_origin;
 roident |        roname
---------+----------------------
       2 | xdb_edb_emp_pub_6
(1 row)
```

For additional information on logical decoding functions see Section 9.26.6 "Replication Functions" under Section 9.26 "System Administration Functions" in the *PostgreSQL Core Documentation* located at:

https://www.postgresql.org/docs/current/static/functions-admin.html

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

After performing this process, it is unlikely that removal of the entire replication system can be done with the xDB Replication Console or the xDB Replication Server CLI. Complete removal of the remaining replication system components must be done manually. Part of this process is removing the control schema and control schema objects from the publication databases. See Section 10.3.4.3 for information on this procedure.

10.4 Miscellaneous xDB Replication Server Processing Topics

This section contains various topics covering the following:

- Handling special characters in replication data
- Replicating Oracle partitioned tables
- Performing an offline snapshot and subsequent synchronization
- Generating an encrypted password
- Writing cron expressions

10.4.1 Publication and Subscription Server Configuration Options

The publication server and the subscription server support various configuration options for purposes such as the following:

- Optimize synchronization performance based on the types of transactions affecting the publication. (See Section 5.8.2 for details on these particular options.)
- Utilize alternate loading methods in snapshot replications. (See Section 5.8.1 for details on these particular options.)
- Special configuration options for multi-master replication. (See Section 6.12 for details on these particular options.)
- Adjust memory usage and transaction size for replications.
- Replicate certain Oracle partitioned table types.
- Replicate special characters found in publication data.
- Special configuration options for the log-based method of synchronization replication. (See Section 10.4.1.15 for details on these particular options.)

Most options apply to the publication server only, although a few are used by the subscription server.

The configuration options for the publication server are set and passed in a text file called the *publication server configuration file* with file name `xdb_pubserver.conf`.

The configuration options for the subscription server are set and passed in a text file called the *subscription server configuration file* with file name `xdb_subserver.conf`.

See Section 3.5 for the directory locations of these files.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Modified publication server configuration options take effect after the publication server is restarted. Similarly, modified subscription server configuration options take effect after the subscription server is restarted.

The configuration options that have been explicitly put into effect by overriding their defaults in the configuration files are logged in the publication server log file and the subscription server log file. Section 3.5 contains the directory locations of these log files.

The following is a description of how to set the configuration options. This is followed by sections describing the purpose of each option.

Step 1: The publication and subscription server configuration files are created during xDB Replication Server installation and already contain all of the configuration options as comments with their default settings.

To change the setting of a configuration option, edit the publication server or subscription server configuration file by removing the comment symbol (#) from the option and substituting the desired value in place of the currently coded value.

The following example shows a portion of the publication server configuration file where replacement of null characters in the publication data has been activated and the replacement character has been set to the question mark character.

```
replaceNullChar = true

#Null Replacement Character
nullReplacementChar = ?
```

Step 2: Restart the publication or subscription server.

Use the following command for CentOS 7 or RHEL 7:

```
systemctl restart edb-xdbpubserver
```

Use the following command for previous Linux versions:

```
service edb-xdbpubserver restart
```

The following sections provide additional detail on the server configuration options.

10.4.1.1 Controlling Logging Level, Log File Sizes, and Rotation Count

Note: The options described in this section apply to the publication server and the subscription server unless otherwise specified.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The following options control various aspects of message logging in the publication server log file, the subscription server log file, and the Migration Toolkit log file.

See Section 10.3.2.4 for additional information on the publication and subscription server log files.

See Section 10.3.2.2 for additional information on the Migration Toolkit log file.

logging.level

Set the `logging.level` option to control the severity of messages written to the publication server log file and the subscription server log file.

```
logging.level={OFF | SEVERE | WARNING | INFO | CONFIG | FINE
| FINER | FINEST | ALL}
```

The default value is `WARNING`.

logging.file.size

Set the `logging.file.size` option to control the maximum file size (in megabytes) of the publication server log file and the subscription server log file.

Note: If `logging.file.count` is set to `0`, the setting of `logging.file.size` is ignored. The log file is allowed to grow without limit.

```
logging.file.size=n
```

The default value is `50` megabytes.

logging.file.count

Set the `logging.file.count` option to control the number of files in the log file rotation history of the publication server log file and the subscription server log file.

```
logging.file.count=n
```

The default value for *n* is `20`.

A non-zero value of *n* specifies the maximum number of log files that are to be created.

Note: In the remaining discussion the publication server log file named `pubserver.log` is used as an example. For the subscription server, the log file is named `subserver.log`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

- Specify a value of `0` to disable log file rotation and create a single, unlimited size log file named `pubserver.log`. This log file will grow to an unlimited size ignoring any setting of `logging.file.size`.
- Specify a value of `1` to disable log file rotation and create a single, limited size log file named `pubserver.log`. The log file is deleted and a new one is created each time the log file reaches the size limit set by `logging.file.size`.
- Specify a value of `2` or greater to enable log file rotation. All log file names have an integer suffix (for example, `pubserver.log.0`, `pubserver.log.1`, `pubserver.log.2`).

When log file rotation is enabled, the log file with the greatest integer suffix contains the oldest messages. When there are enough messages to generate every file in the history rotation, the oldest messages are in `pubserver.log.`*`n`*`-1` where *n* is the setting of `logging.file.count`. Log file `pubserver.log.0` is the current, active log file containing the most recent messages.

When log file rotation is enabled and the current, active log file (`pubserver.log.0`) reaches the size specified by `logging.file.size`, then the following events occur:

- The log file containing the oldest messages (`pubserver.log.`*`n`*`-1`) is deleted.
- Each remaining log file is renamed with the next greater integer suffix (`pubserver.log.`*`m`* is renamed to `pubserver.log.`*`m`*`+1` with *m* varying from `0` to *n*`-2`).
- A new, active log file is created (`pubserver.log.0`).

mtk.logging.file.size

Note: This option applies to the publication server only.

Set the `mtk.logging.file.size` option to control the maximum file size (in megabytes) of the Migration Toolkit log file.

```
mtk.logging.file.size=n
```

The default value is `50` megabytes.

mtk.logging.file.count

Note: This option applies to the publication server only.

Set the `mtk.logging.file.count` option to control the number of files in the log file rotation history of the Migration Toolkit log file.

```
mtk.logging.file.count=n
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The default value for *n* is 20.

A non-zero value of *n* specifies the maximum number of history log files that are to be created.

- Specify a value of 0 to disable log file rotation and create a single, limited size log file named mtk.log. The log file is deleted and a new one is created each time the log file reaches the size limit set by mtk.logging.file.size.
- Specify a value of 1 or greater to enable log file rotation. All log file names have an integer suffix (for example, mtk.log.1, mtk.log.2).

When log file rotation is enabled, the log file with the greatest integer suffix contains the oldest messages. When there are enough messages to generate every file in the history rotation, the oldest messages are in mtk.log.*n* where *n* is the setting of mtk.logging.file.count.

Log file mtk.log is the current, active log file containing the most recent messages.

When the current, active log file (mtk.log) reaches the size specified by mtk.logging.file.size, then the following events occur:

- The log file containing the oldest messages (mtk.log.*n*) is deleted.
- Each remaining log file with a suffix is renamed with the next greater integer suffix (mtk.log.*m* is renamed to mtk.log.*m*+1 with *m* varying from 1 to *n*-1).
- Log file mtk.log is renamed to mtk.log.1.
- A new, active log file is created (mtk.log).

10.4.1.2 Replacing Null Characters

Note: The options described in this section apply to the publication server only.

A character consisting of binary zeros (also called the null character string) and represented as 000 in octal or 0x00 in hexadecimal can result in errors when attempting to load such data into a Postgres character column.

You may get the following error in the Migration Toolkit log file when performing a snapshot replication of an Oracle table that contains the null character string:

```
Loading Table Data in 8 MB batches...
Disabling FK constraints & triggers on edb.null_test before truncate...
Truncating table NULL_TEST before data load...
Disabling indexes on edb.null_test before data load...
Loading Table: NULL_TEST ...
Error Loading Data into Table: NULL_TEST: ERROR: invalid byte sequence for
encoding "UTF8": 0x00
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
  Where: COPY null_test, line 2
```

The same circumstance may also produce the following error in the Migration Toolkit log file:

```
Loading Table Data in 8 MB batches...
Disabling FK constraints & triggers on edb.null_clob before truncate...
Disabling indexes on edb.null_clob before data load...
Loading Large Objects into table: NULL_CLOB ...
[NULL_CLOB] Migrated 1 rows.
com.edb.util.PSQLException: Zero bytes may not occur in string parameters.,
Skipping Batch
```

If any of these errors occur, you can set an option that will convert each null character encountered in character columns of the source tables to a space character, or to any other character of your choice, before loading the target tables.

Note: This option does not alter null characters encountered in columns with binary data types such as Oracle `RAW` and `BLOB` data types.

Set the following option:

```
replaceNullChar=true
```

This option results in the substitution of a space character for each null character encountered in the source character data.

If you want to use a character other than a space character to replace each null character, use the following option in addition to `replaceNullChar=true`.

```
nullReplacementChar=char
```

`char` is a single character you want to substitute for the null character. For example, the following combination will replace each null character with the hash symbol #.

```
replaceNullChar=true
nullReplacementChar=#
```

10.4.1.3 Schema Migration Options

Note: The option described in this section applies to the subscription server only.

The option in this section controls how certain aspects of the publication database schema are migrated to the subscription database.

skipCheckConst

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

By default, column `CHECK` constraints from publication tables are migrated to the subscription table definitions when the subscription is created. Set this option to `true` if you do not want `CHECK` constraints as part of the subscription table definitions.

Setting this option to `true` is useful if the `CHECK` constraint is based on a built-in function supported by the publication database server, and this built-in function does not exist in the subscription database server.

```
skipCheckConst={true | false}
```

The default value is `false`.

10.4.1.4 Replicating Oracle Partitioned Tables

Note: The option described in this section must be set to the same value for both the publication server and the subscription server.

Note: This feature applies only for subscriptions in an Advanced Server database. It does not apply to subscriptions in a PostgreSQL database.

In Oracle, table partitioning provides the capability to store table rows in different physical locations (tablespaces) according to a rule defined on the table.

The most common types of Oracle table partitioning are the following:

- **Range Partitioning.** Ranges of values defined on a column determine which tablespace a row is stored.
- **List Partitioning.** A list of values defined on a column determines which tablespace a row is stored.
- **Hash Partitioning.** An algorithm on a column generates a hash key, which determines which tablespace a row is stored.

Note: If you are using Advanced Server, table partitioning using Oracle compatible table partitioning syntax is an available feature. See the section on table partitioning in the *Database Compatibility for Oracle Developer's Guide* for information. See Section 7.10 for information on including Postgres partitioned tables in a replication system. The `importPartitionAsTable` option described in this section applies only to table partitioning in an Oracle database.

The `importPartitionAsTable` option controls what happens when an Oracle partitioned table is part of the publication.

```
importPartitionAsTable={true | false}
```

The default value is `false`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Depending upon the Oracle partitioned table type and the setting of the `importPartitionAsTable` option one of the following may occur:

- A set of inherited tables is created in Advanced Server to which the Oracle partitioned table is replicated. The rows can be stored in different physical locations.
- A plain, single table with no inheritance is created in Advanced Server to which the Oracle partitioned table is replicated. All rows are stored in one physical location.
- No table is created in Advanced Server for the Oracle partitioned table, and an error message is written to the Migration Toolkit log file.

When `importPartitionAsTable=false` (the default setting), the following occurs:

- A list partitioned table is replicated as a set of inherited Advanced Server tables.
- A range partitioned table is replicated as a set of inherited Advanced Server tables.
- A hash partitioned table is not replicated to Advanced Server, and an error message is written to the Migration Toolkit log file.

Note: If there are subscription tables created as sets of Advanced Server inherited tables, then you must also set the `enableConstBeforeDataLoad` option in the publication server configuration file to `true`. See Section 10.4.1.6 for information on the `enableConstBeforeDataLoad` option.

When `importPartitionAsTable=true`, the following occurs:

- A list partitioned table is replicated as a single Advanced Server table with no inheritance.
- A range partitioned table is replicated as a single Advanced Server table with no inheritance.
- A hash partitioned table is replicated as a single Advanced Server table with no inheritance.

Setting the `importPartitionAsTable` option to `true` allows you to replicate a broader range of Oracle partitioned table types, but as normal Advanced Server tables without simulating partitions by using inheritance.

10.4.1.5 Specifying a Custom URL for an Oracle JDBC Connection

Note: The option described in this section applies to the publication server only.

By default the xDB Replication Server supports the basic thin client URL pattern for an Oracle JDBC connection. If there is a requirement to specify custom connectivity credentials, specify the advanced URL using the following option.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

```
oraJDBCCustomURL=customURL_string
```

The following is an example of custom connectivity to an Oracle database.

```
oraJDBCCustomURL=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=
$HOST)(PORT=$PORT))(CONNECT_DATA=(SERVICE_NAME
=$SERVICE_NAME)(SERVER=DEDICATED)))
```

The parameters prefixed with a dollar sign (`$`) are dynamically replaced based on the actual connection values specified when adding the Oracle publication database (see Section 5.2.2). Alternatively, the parameters prefixed with a dollar sign can be replaced by hardcoded values in the URL string in which case these hardcoded values override what is specified when adding the publication database.

10.4.1.6 Snapshot Replication Options

Note: The options described in this section apply to the publication server only unless otherwise specified.

The server configuration options discussed in this section apply to snapshot replications.

escapeTabDelimiter

When JDBC `COPY` is used in snapshot replication, the data delimiter between column values is an escaped tab character (`\t`). Set this option to `false` if you do not want to escape the tab delimiter character.

```
escapeTabDelimiter={true | false}
```

The default value is `true`.

mtkCopyDelimiter

When JDBC `COPY` is used in snapshot replication, the data delimiter between column values is an escaped tab character (`\t`). Set this option to change the data delimiter character.

```
mtkCopyDelimiter=c
```

c denotes the single replacement character for the data delimiter.

The default value is `\t`.

enableConstBeforeDataLoad

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The `enableConstBeforeDataLoad` option controls whether or not table constraints, including triggers, are re-enabled before loading data into target tables. The default process is that the tables are loaded first, and then the constraints are enabled afterwards.

Activate this option if there are triggers that affect how data is loaded into the target tables.

If there are target tables created as sets of Postgres inherited tables resulting from partitioned Oracle source tables, then this option must be enabled.

```
enableConstBeforeDataLoad={true | false}
```

The default value is `false`.

10.4.1.7 Assigning an IP Address for Remote Method Invocation

Note: The option described in this section applies to the publication server and the subscription server.

For Linux only:

An alternative method to modifying the `/etc/hosts` file so that the hostname is associated with a non-loopback IP address as discussed in Section 5.1.6.2 is to specify the network IP address using the `java.rmi.server.hostname` option.

In the publication server configuration file, set this option to the network IP address of the host running the publication server.

In the subscription server configuration file, set this option to the network IP address of the host running the subscription server.

```
java.rmi.server.hostname=xxx.xxx.xx.xxx
```

For example, instead of modifying the `/etc/hosts` file to look like the following for a publication or subscription server running on host `192.168.2.19`:

```
#127.0.0.1              localhost.localdomain localhost
192.168.2.19             localhost.localdomain localhost
```

You can set the IP address in the server configuration file as shown by the following:

```
#On Linux machines, the localhost to real IP may not give correct results. Hence
#users are advised to override the following property with server IP address
java.rmi.server.hostname=192.168.2.19
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.4.1.8 Using pgAgent Job Scheduling

Note: The option described in this section applies to the publication server only.

Note: Using pgAgent job scheduling has significance only if Postgres is the publication database.

Note: You must have pgAgent installed and running on the host where the publication database resides.

When the `pgdbschedule` option is set to `true`, xDB Replication Server uses the *pgAgent* job scheduler instead of the default Quartz job scheduler.

When activated, pgAgent takes over the following scheduling tasks from Quartz:

- Scheduling shadow table history cleanup in the publication database. See Section 7.5.1 for information on scheduling shadow table history cleanup.
- Scheduling transaction set creation. A transaction set creation job is scheduled to run every hour to create transaction sets from the updates on the source tables. Transaction sets are applied to the target tables.

Unlike the Quartz scheduler, pgAgent can still run and perform its tasks even if the publication server is not running.

```
pgdbschedule={true | false}
```

The default value is `false`.

10.4.1.9 Forcing Immediate Shadow Table Cleanup

Note: The option described in this section applies to the publication server only.

A cleanup job is provided that can be run on demand or on a schedule to remove dead (processed) tuples from the shadow tables (see Section 7.5).

However, to perform even quicker cleanup scheduling, turn on this option to force the cleanup of shadow tables after every synchronization replication.

```
postSyncShadowTableCleanup={true | false}
```

The default value is `false`.

10.4.1.10 Setting Event History Cleanup Threshold

The event history cleanup job is scheduled to run every day at 12 AM to remove completed, historical, event and replication history data from the control schema

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

`xdb_events`, `xdb_events_status`, `xdb_pub_replog`, and `xdb_pub_table_replog` tables that are older than *n* days. By default the history data older than seven days is removed.

Specify a value of `0` to cleanup all, completed, event history and replication history data, regardless of its age.

See Section 7.5.4 for information on cleaning up event and replication history.

```
historyCleanupDaysThreshold=n
```

The default value is `7` days.

10.4.1.11 DDL Change Replication Table Locking

Note: The option described in this section applies to the publication server only.

When the DDL change replication process is invoked, each affected table in the replication system is acquired in turn with an exclusive lock before the DDL change is applied to the table.

Set `ddlChangeTableLock` to `false` if you do not want an exclusive lock placed on the table before applying the DDL change. This option should be set to `false` only if there are no write transactions expected on the target table. If write transactions do occur, they may not be recorded by the replication system.

See Section 7.8 for information on DDL change replication.

```
ddlChangeTableLock={true | false}
```

The default value is `true`.

10.4.1.12 Persisting Zero Transaction Count Replication History

Note: The option described in this section applies to the publication server only.

If you wish to maintain zero transaction count records in the replication history after the publication server is restarted, set `persistZeroTxRepEvent` to `true`. Otherwise, zero transaction count records are no longer available once the publication server is restarted.

See Section 7.4 for information on viewing replication history.

```
persistZeroTxRepEvent={true | false}
```

The default value is `false`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.4.1.13 Skipping Grants of Table Level User Privileges on MMR Target Tables

Note: This option applies to the publication server only.

When creating non-MDN nodes in a multi-master replication system, the publication server creates the publication tables and their corresponding shadow tables in the non-MDN node database.

When `skipTablePrivileges` is set to `false`, which is the default value, the database user privileges on the publication tables in the master definition node are granted to the same database users on the publication tables in the newly created non-MDN node.

The required privileges are also granted to these database users on the corresponding shadow tables in the non-MDN node so these database users can perform updates on the publication tables and the changes are recorded in the corresponding shadow tables. This enables proper synchronization replication of any such changes.

This granting of privileges occurs only for database users with privileges on the master definition node publication tables at the time the non-MDN node is defined using the xDB Replication Console or the xDB Replication Server CLI.

If you do not want the publication server to grant these database user privileges to the non-MDN publication tables and shadow tables when defining the non-MDN node, set `skipTablePrivileges` to `true`. In this case, you must explicitly grant the privileges on the publication tables and corresponding shadow tables in the non-MDN node for any database user that you wish to provide update access to on these tables. See Step 2 of Section 5.1.4.3 for information regarding the required privileges.

This usage would typically be for the case where different database users are to access the non-MDN node publication tables than the database users who access the master definition node publication tables.

```
skipTablePrivileges={true | false}
```

The default value is `false`.

10.4.1.14 Applying Grants of Table Level User Privileges on SMR Target Tables

Note: This option applies to the subscription server only.

Note: This option applies only when both the publication database and the subscription database are Postgres databases.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

When creating a subscription in a single-master replication system, the subscription server creates the subscription tables in the subscription database.

When `skipTablePrivileges` is set to `true`, which is the default value, no database user privileges are granted on these subscription tables to any database user. By default the subscription database user specified when the subscription database definition is created (see Section 5.3.2) is the owner of the subscription tables.

This is the typical, expected scenario since the data in subscription tables should not be updated by user applications other than the xDB Replication Server.

Database users that require access to the subscription tables must be explicitly granted such privileges.

If however, you do want the subscription server to grant database user privileges to the subscription tables for the same database users that already have access privileges to the publication tables, set `skipTablePrivileges` to `false` in the subscription server configuration file. (The setting of `skipTablePrivileges` in the publication server configuration file is ignored for this process in a single-master replication system.)

In this case, the same access privileges are granted on the subscription tables to database users with privileges on the publication tables at the time the subscription is defined using the xDB Replication Console or the xDB Replication Server CLI.

```
skipTablePrivileges={true | false}
```

The default value is `true`.

10.4.1.15 Log-Based Method of Synchronization Options

Note: This option applies to the publication server only.

walTxSetCreationInterval

When using the log-based method of synchronization replication the `walTxSetCreationInterval` option controls the time interval between creations of the transaction sets, which affects the size of the transaction set (that is, the batch size). The default setting results in the creation of a transaction set every 5,000 milliseconds (5 seconds) assuming changes to the publication tables to be replicated are available.

This value should be adjusted based on the workload (that is, the transaction per minute (TPM) rate) on the target publication tables.

If the TPM rate is on a higher end, the `walTxSetCreationInterval` option should be set to a relatively low value.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

If the TPM rate is on a lower end, the option should be set to a higher value to ensure that a transaction set is large enough to allow an average batch size in the range of 100 to 500 transactions.

```
walTxSetCreationInterval=n
```

The default value is `5000` milliseconds.

walStreamQueueLimit

The `walStreamQueueLimit` option defines the upper limit for the number of WAL entries that can be held in the queue pending for processing at a point in time. Once the queue becomes full, the WAL stream receiver blocks additions until space becomes available in the queue as transaction entries are popped out of the queue for processing.

A value of 0 indicates there will be no upper limit. Note that too high a setting may result in Java heap space out of memory errors. See Section 5.1.1 for information on adjusting the Java heap memory size.

```
walStreamQueueLimit=n
```

The default value is `10000`.

pendingTxSetThreshold

The `pendingTxSetThreshold` option defines the upper threshold limit for the number of pending transaction sets that when reached, causes the extraction of transaction data from the WAL stream and its parsing to be put on hold until the pending transactions are processed.

This is to avoid a situation where the data is continuously pushed over the WAL stream channel, but is not being processed and applied due to some failure or lack of scheduling of the synchronization process. This may result in a Java heap space out of heap memory error. See Section 5.1.1 for information on adjusting the Java heap memory size.

```
pendingTxSetThreshold=n
```

The default value is `10`.

10.4.1.16 Setting the Apache DBCP Connection Validation Query Timeout

Note: This option applies to the publication server only.

The Apache Commons Database Connection Pooling (DBCP) component is the Apache pooling framework used by the publication server for establishing JDBC connections.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The `jdbc.pool.validationQueryTimeout` option controls the timeout setting when a validation query is executed at the time of allocating a connection from the pool. This is the amount of time in seconds before an exception is returned if the connection validation query does not succeed.

The default timeout value is 30 seconds. In situations where network connections are not reliable, the timeout value can be increased accordingly to allow more time for the connection attempt. Specify a value of 0 if no timeout is desired.

```
jdbc.pool.validationQueryTimeout=n
```

The default value is `30`.

10.4.2 Encrypting the Password in the xDB Replication Configuration File

If you need to change the password in the xDB Replication Configuration file, you must first encrypt the password. Use the `encrypt` command of the xDB Replication Server CLI to generate the encrypted form of the password from its plain text form given in an input file.

Step 1: Create a text file with the password you wish to encrypt. Do not leave any white space before or after the password.

The following example shows the text `newpassword` in the input file `passfile`:

```
$ cat passfile
newpassword
$
```

Step 2: Use the `edb-repcli.jar` file to execute the xDB Replication Server CLI with the `encrypt` command by first including the Java `bin` directory in your `PATH` environment variable and making *`XDB_HOME`*`/bin` your current working directory.

For example, assuming `/usr/bin` contains the `java` executable program and xDB Replication Server is installed into the *`POSTGRES_INSTALL_HOME`* directory, perform the following:

```
$ export PATH=/usr/bin:$PATH
$ cd /opt/PostgresPlus/9.4AS/bin
$ java -jar edb-repcli.jar -encrypt -input ~/passfile -output ~/encrypted
```

The following shows the encrypted form of the password in the output file `encrypted`:

```
$ cat ~/encrypted
4mKq/4jQQoV2IypCSmPpTQ==
$
```

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Step 3: Copy and paste the encrypted password into the xDB Replication Configuration file.

```
#xDB Replication Server Configuration Properties
#Thu Sep 03 11:13:27 GMT-05:00 2015
admin_password=4mKq/4jQQoV2IypCSmPpTQ==
admin_user=admin
```

10.4.3 Writing a Cron Expression

A *cron expression* is a text string used to express a schedule of dates and times. The Linux `cron` tool uses a cron expression to schedule the execution of a job. xDB Replication Server uses the Quartz job scheduling system for scheduling replications.

When creating a schedule for an xDB Replication Server replication system, a cron expression can be specified. There are a number of formats for cron expressions. You must use the cron expression format supported by Quartz.

The remainder of this section provides an overview of most of the types of cron expressions that can be used for an xDB Replication Server schedule. For a more comprehensive treatment of cron expressions, refer to the Quartz documentation.

A Quartz cron expression consists of six mandatory fields, followed by one optional field. Each field is separated from its neighbors by one or more consecutive space characters. The fields are order dependent and are listed as they must appear below:

```
ss mi hr dd mm dow [ yyyy ]
```

Table 10-4 - Cron Expression Fields

Field	Values	Description
ss	0 - 59	Second of the minute
mi	0 - 59	Minute of the hour
hr	0 - 23	Hour of the day
dd	1 - 31 or ?	Day of the month – if *dow* is given, then *dd* must be specified as ?
mm	1 - 12 or JAN - DEC	Month of the year (3-letter month abbreviations are not case sensitive)
dow	1 – 7 or SUN – SAT or ?	Day of the week – if *dd* is given, then *dow* must be specified as ? (3-letter day of the week abbreviations are not case sensitive)
yyyy	1970 - 2099	Year – if omitted, then any year applies

There are a number of characters that have special meaning that can be utilized in all fields unless noted.

Table 10-5 - Cron Expression Special Characters

Character	Meaning	Example
,	Separates a list of values	MON,WED,FRI – Every Monday, Wednesday, and Friday
-	Separates the low and high end of a range of values	MON-FRI – Every Monday through Friday

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

Character	Meaning	Example
*	Allows all legal values for the field	0 10 14 * * ? – Every day of every month at 2:10 PM
x/i	Specifies an increment, i, starting with x	0 0/10 * * * ? – Every 10 minutes starting on the hour for every day of every month (e.g., 8:00:00, 8:10:00, 8:20:00)
L	When used in the day of the month (dd) field, means the last day of the month	0 30 15 L 8 ? – Every August 31st at 3:30 PM
L	When used by itself in the day of the week field (dow), means Saturday	30 0 12 ? AUG L – The next Saturday in August at 30 seconds past 12:00 noon
xxxL	When used in the day of the week field (dow) following a day of the week, means the last xxx day of the month	30 0 12 ? AUG 6L – The last Friday in August at 30 seconds past 12:00 noon
xW	Used in the day of the month field (dd) following a day of the month, x, to specify the weekday closest to x without going over into the next or previous month.	1W – The weekday closest to the 1st of the month. If the 1st is a Wednesday, the result is Wednesday the 1st. If the 1st is a Sunday, the result is Monday the 2nd. If the 1st is a Saturday, the result is Monday the 3rd because the result does not go into the previous or following month.
xxx#n	Used in the day of the week field (dow) to specify the nth xxx day of the month	2#3 – The third Monday of the month (2 = Monday, 3 = third occurrence)

The following illustrates some examples of cron expressions.

Table 10-6 - Cron Expression Examples

Cron Expression	Meaning
0 0 12 20 AUG ? 2009	12:00:00 noon on August 20, 2009
0 15 13 ? AUG WED	1:15:00 PM every Wednesday in August
30 30 8 ? * MON,WED,FRI	8:30:30 AM every Monday, Wednesday, and Friday of every month
0 0 8 ? * 2-6	8:00:00 AM Monday thru Friday of every month
0 0/30 8,9,10 15,L * ?	8:00:00 AM, 8:30:00 AM, 9:00:00 AM, 9:30:00 AM, 10:00:00 AM, 10:30:00 AM on the 15th and the last day of the month of every month
0 0 9 ? 9 L	9:00:00 AM each Saturday in September
0 0 1 ? * MonL	1:00:00 AM on the last Monday of the month of every month
0 30 16 15W sep ?	4:30:00 PM on the weekday of September closest to the 15th
0 30 16 ? * WED#2	4:30:00 PM on the second Wednesday of every month

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

10.4.4 Disabling Foreign Key Constraints for Snapshot Replications

In a snapshot replication, the publication server calls EnterpriseDB's Migration Toolkit, which disables foreign key constraints on tables so it can truncate the target tables before loading rows. In Postgres, foreign key constraints are implemented using triggers, so in actuality, the Migration Toolkit disables triggers on the target tables by setting column `relhastriggers` of `pg_catalog.pg_class` to `false` for each target table.

No user (not even a superuser) is allowed to directly modify the data in a Postgres system catalog table unless the following conditions are satisfied:

- The database user attempting to modify the rows of a system catalog table is a superuser.
- In the Postgres system catalog table `pg_catalog.pg_authid`, the column `rolcatupdate` is set to `true` for the row identifying the superuser attempting to update a system catalog table. This requirement applies only to Postgres version 9.4 or earlier. The column `rolcatupdate` no longer exists in Postgres 9.5 or later.

To verify that a user has the privilege to update the system catalog tables, select the user name under the Login Roles node in pgAdmin (Postgres Enterprise Manager Client in Advanced Server). The Update Catalogs property should be set to `Yes`.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

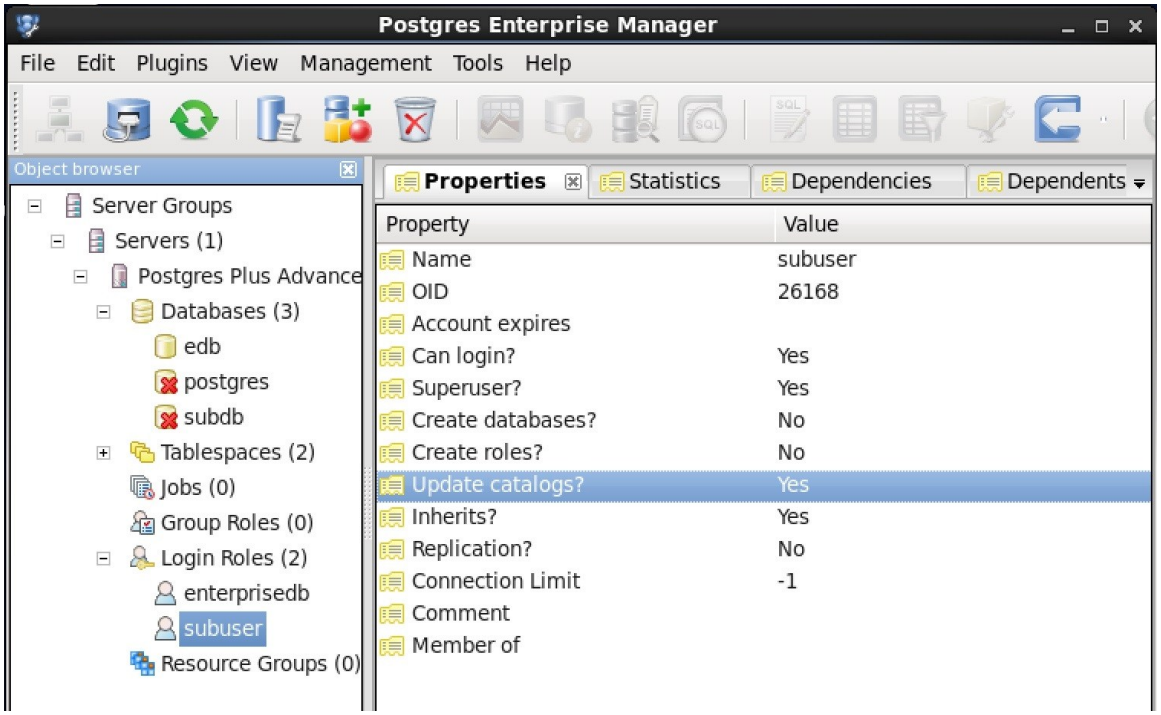


Figure 10-6 - User with privilege to update system catalogs

If the Update Catalogs property is set to `No`, click the secondary mouse button on the user name in the Object Browser and choose Properties from the menu. Select the Role Privileges tab, check the Can Modify Catalog Directly box, and click the OK button.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

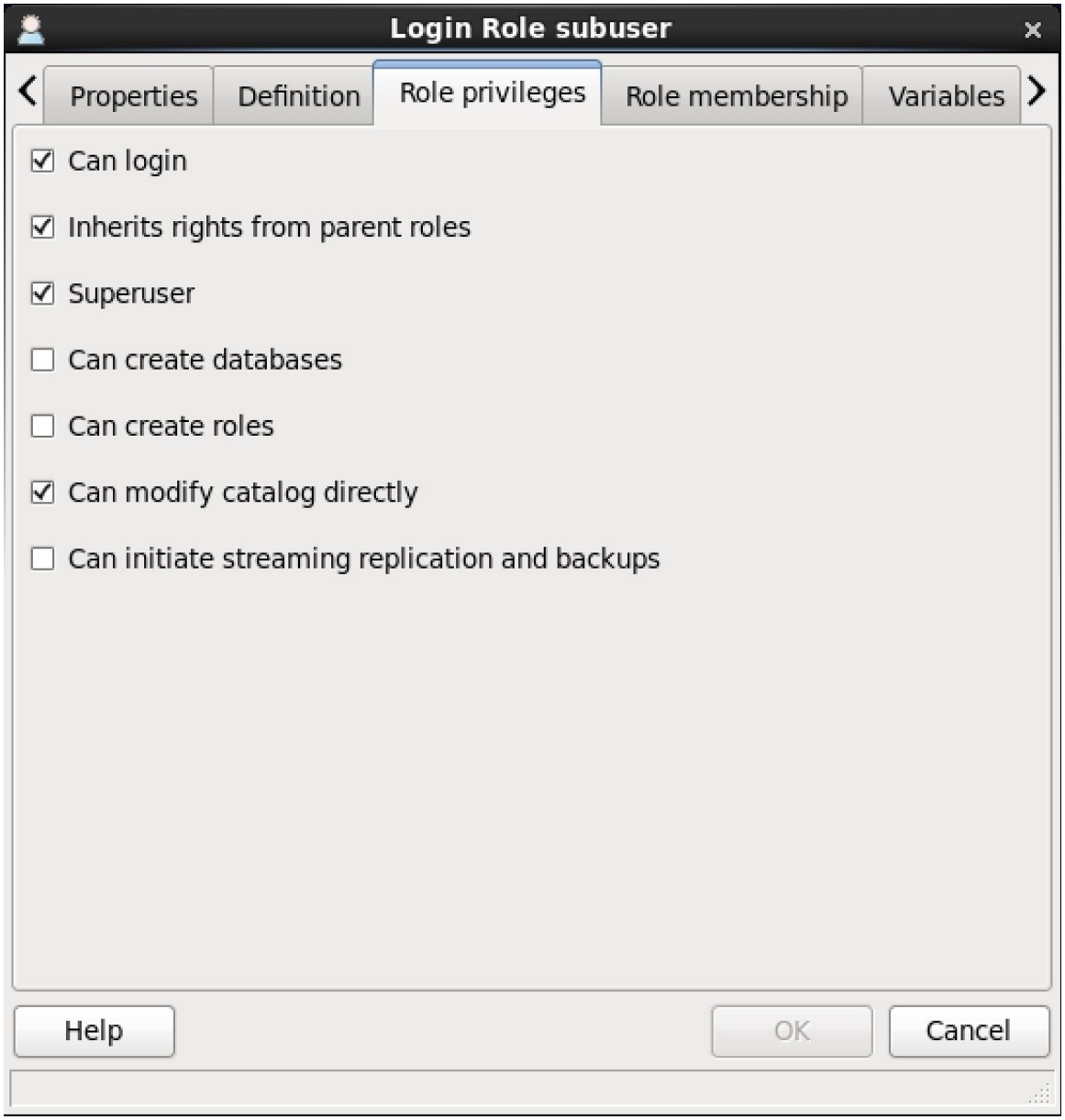


Figure 10-7 - Granting system catalog update privilege

10.4.5 Quoted Identifiers and Default Case Translation

A *quoted identifier* is an identifier created with its name enclosed within double quote characters ("). The text enclosed within double quotes is stored as the object identifier name exactly as given with no default case translation of alphabetic characters. Quoted identifiers occur in both Oracle and Postgres.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

For example, `CREATE TABLE "MyTable" ...` produces a table name that is stored in the database system's data dictionary as `MyTable`. References to this table must be made using an uppercase `M`, an uppercase `T`, and lowercase letters for the rest of the name.

If a database object is created without the double quotes surrounding its identifier name, default case translation of alphabetic characters occurs.

In Oracle, the default case translation is to uppercase. For example, `CREATE TABLE MyTable ...` would result in an object identifier name of `MYTABLE`.

In Postgres, the default case translation is to lowercase. For example, `CREATE TABLE MyTable ...` would result in an object identifier name of `mytable`.

10.4.6 Replicating the SQL Server SQL_VARIANT Data Type

This section discusses how to replicate a table containing the SQL Server `SQL_VARIANT` data type.

The `SQL_VARIANT` data type defines a column so that the individual values in that column may be of different data types. For example, the same `SQL_VARIANT` column can store values that have been explicitly cast as character, integer, numeric, and date/time.

However, if a table containing a `SQL_VARIANT` column is to be replicated to a Postgres database, the usage of the column in Postgres is restricted to a single data type to which all the values in the `SQL_VARIANT` column are implicitly convertible (that is, without the use of explicit casting). For example, an integer value is implicitly convertible to a `FLOAT` data type, but a floating point value is not implicitly convertible to an `INTEGER` data type.

The following restrictions apply in order to use replication on tables with the `SQL_VARIANT` data type:

- The values stored within the `SQL_VARIANT` columns of the table to be replicated must be implicitly convertible to the same data type in Postgres.
- If there is more than one table with `SQL_VARIANT` columns to be replicated to the same Postgres database, then all such `SQL_VARIANT` columns must contain values that are implicitly convertible to the same data type in Postgres.

In the Postgres subscription database, you define a domain named `sql_variant` that maps to an underlying data type to which all values in the `SQL_VARIANT` columns are implicitly convertible.

The following example shows how to set up replication for a table containing a `SQL_VARIANT` data type used to store numeric values, but of different data types.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

The SQL Server table definition is the following:

```
CREATE TABLE variant_tbl (
    f1              INTEGER PRIMARY KEY,
    f2              SQL_VARIANT
);

INSERT INTO variant_tbl VALUES (1, CAST(1423.23 AS NUMERIC(6,2)));
INSERT INTO variant_tbl VALUES (2, CAST(8001 AS INTEGER));
INSERT INTO variant_tbl VALUES (3, CAST('4321' AS CHAR(4)));
GO
```

The following query uses a function named `SQL_VARIANT_PROPERTY` to show the values stored in column `f2` and their data types.

```
1> SELECT *,
2>     SQL_VARIANT_PROPERTY(f2,'BaseType') AS basetype,
3>     SQL_VARIANT_PROPERTY(f2,'Precision') AS precision,
4>     SQL_VARIANT_PROPERTY(f2,'Scale') AS scale
5> FROM variant_tbl;
6> GO
f1          f2         basetype   precision  scale
----------- ---------- ---------- ---------- ----------
          1 1423.23    numeric    6          2
          2 8001       int        10         0
          3 4321       char       0          0

(3 rows affected)
```

In the Postgres subscription database, create a domain named `sql_variant` with an underlying data type that is compatible with the values that are stored in the SQL Server `SQL_VARIANT` column:

```
CREATE DOMAIN sql_variant AS NUMERIC(6, 2);
```

After replication occurs, the subscription table is created using the `sql_variant` domain in place of the `SQL_VARIANT` data type of the publication table.

At the bottom of the following Object Browser window, note the presence of the `sql_variant` domain under the Domains node of the `public` schema.

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

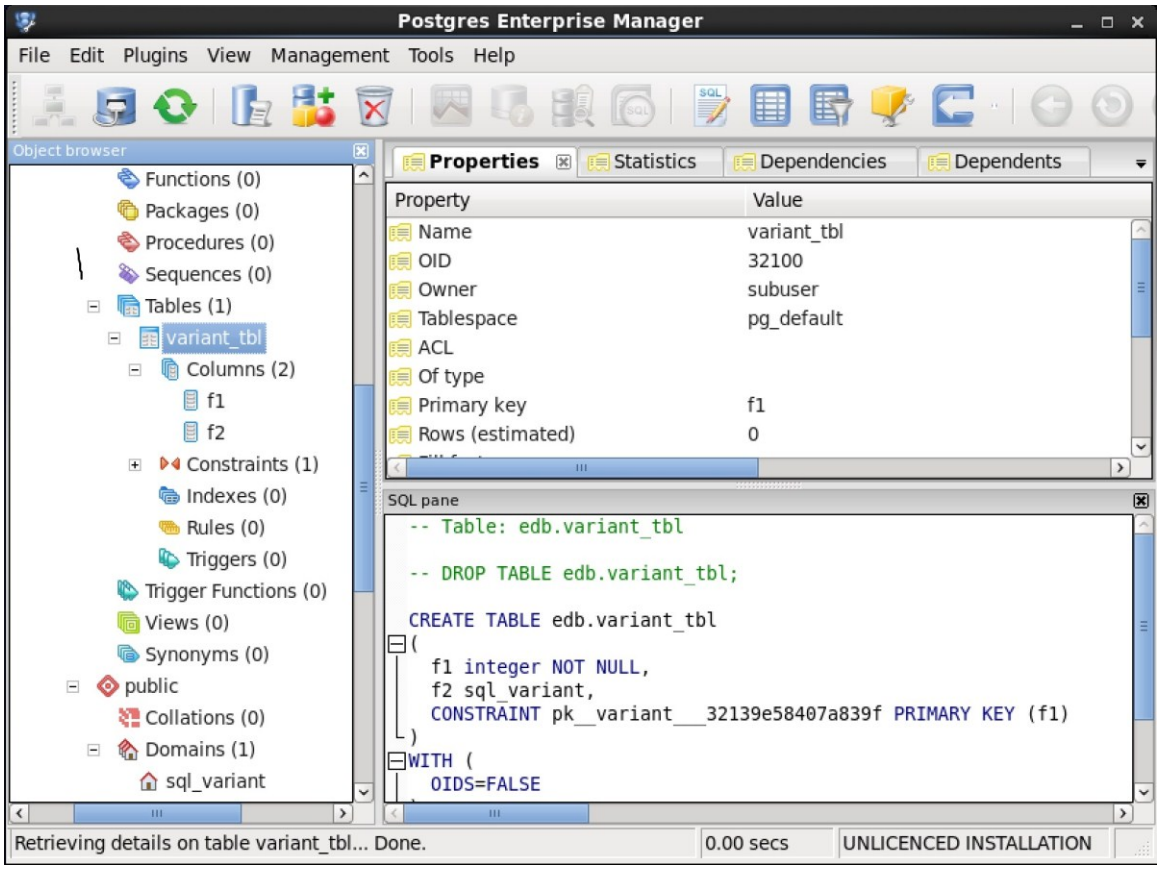


Figure 10-8 - Subscription table with sql_variant column

10.5 Service Pack Maintenance

Maintenance items (bug fixes and enhancements) that have been added to this version of xDB Replication Server are listed below.

1. Registering your xDB Replication Server product with an EnterpriseDB product license key is no longer required. Thus, all components related to registering the product have been removed. The following are the removed components: 1) the Product Registration dialog box accessed from the xDB Replication Console Help menu, 2) the license_key parameter located in the xDB Replication Configuration file, and 3) the xDB Replication Server CLI registerkey command. (43230)

2. Partitioned tables created using the declarative partitioning feature of PostgreSQL and Advanced Server version 10 can now be replicated in a log-based single-master or multi-master replication system. (43134)

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.

3. In an SMR system, removal of a table from a publication that has one or more existing subscriptions is now permitted. Previously, no tables from a publication in an SMR system could be removed if there are existing subscriptions. (43110)

Copyright © 2010 - 2019 EnterpriseDB Corporation. All rights reserved.