Oracle Database Migration
Pitfalls and Promise

An EnterpriseDB White Paper
for DBAs, Application Developers, and Enterprise Architects

July, 2009

http://www.enterprisedb.com
Executive Summary

Enterprises have a variety of reasons for wanting to migrate applications from one database platform to another including reducing license fee costs, simplifying heterogeneous architectures, or taking advantage of new technologies.

For technical reasons, migrations are difficult, costly, and risky because of the use of many vendor proprietary database features integrated into the once ‘standard’ platform of SQL. In addition, the re-training effort for people and subsequent difficulty in re-coding applications and testing the changes often don’t cost justify the benefits of the move. The result is vendor lock-in.

Previous vendor attempts at database compatibility have fallen far short of bridging the migration gap because of a server side only focus on easily translated features, poor success at finding analogs for proprietary features, a lack of application level compatibility (procedural languages and APIs), and a lack of compatible tools for DBAs and Developers.

Postgres Plus Advanced Server from EnterpriseDB solves the migration problem with compatibility in SQL Extensions, the Procedural Language, DBA and Developer tools, application APIs, and migration tools for Oracle database applications. This approach makes for significantly faster migrations, dramatically reduced re-coding efforts, shorter learning curves, significantly less risk, and ultimately a beneficial justification for making the move.

This paper contains an actual walk-through of migrating an Oracle sample schema and provides step by step instructions on how to get ready for a migration, select what to migrate, perform the migration, and setup the application to talk to the new database.

A free telephone consultation can be scheduled with an EnterpriseDB Oracle compatibility expert who can quickly help you to determine the quickest, safest path to a justifiable migration. If appropriate, a migration feasibility assessment can be performed on your Oracle databases.
The Roots of Vendor Lock-in

In order to reduce costs, simplify IT infrastructures, or to take advantage of new technology, companies regularly investigate or attempt to move from one database platform to another. This is not a new idea or movement as it has been happening in some fashion for many years. The difficulty that companies have faced with this process which in many cases prevented such a movement, is that migrations are not easy.

Although most commercial database products support some level of ANSI SQL, applications built for those products are far from portable. Database vendors have implemented unique data types and functions, proprietary APIs, stored procedure languages, command constructs and utilities that collectively eliminate the very portability ANSI SQL strives to achieve.

As application developers and DBAs learn to navigate those interfaces and deploy database applications that embed those proprietary constructs, vendors reach a significant level of technology lock-in. When customers try to migrate to a new database platform, they realize that there are substantial changes that need to be made to both the database and the application that results in a long and difficult process whose costs often don't justify the benefits of move.

History has proven that, once a vendor is locked in and is no longer motivated to compete for a seat at the IT table, bad things often happen - prices increase, support levels deteriorate and innovation stagnates.

The Struggle for Database Compatibility

A compatible database is one that claims to offer the same functionality as another database vendor. If the compatibility is adequate enough, then the problems of traditional migrations are avoided since the new database supports the same syntax, languages and functionality of the original database. Thus the cost of the migration (the term migration is not completely accurate since this is more like a port), is substantially reduced since the timeframe and effort involved is only a fraction of the original cost.

Database and third party vendors have made valiant attempts to produce migration tools to help you convert databases automatically in a somewhat simplistic and mechanical fashion. Migration tools are good at taking standard syntax from one language and converting it into the standard syntax of the new database.
The problem is that all database vendors offer different semantic functionality and thus one way of doing something may not translate into easily or completely into the new database. Thus manual intervention or recoding is necessary.

Migration tools also don't have human intelligence to decide that simply converting syntax to syntax may not be the best performing result either. So, even after the tool converts the code, manual intervention is necessary. These tools also can't guarantee the converted code will actually execute the same way as the original code and thus potentially return incorrect results!

In addition, these migration tools typically focus on the server / database side and completely ignore the application and any embedded SQL or calls to stored functions. This becomes the customer's responsibility to analyze and manually modify the application logic.

Other obstacles related to different database functionality include staff learning curves to understand the differences and learning the skills needed to operate different toolsets. Thus the database administrators and database developers need to be trained and educated on the capabilities / functionality of the new database. That takes time and money.

The time, energy and ultimately cost of the migrations become so unbearable that the migration project is typically shutdown even before it gets started.

**The Importance of a Migration Strategy**

Another impediment to database migrations is the lack of a well defined low risk low cost strategy for moving an enterprise from one database to another over time. EnterpriseDB, in working closely with customers over many years, has identified a working pattern of success for introducing new database technology into an existing legacy architecture.

The strategy is characterized by starting with low risk, easy to execute projects that demonstrate technical, financial and human feasibility in introducing an Oracle compatible technology into the organization.

The table below summarizes a successful approach to introducing Postgres Plus Advanced Server into an enterprise with legacy technology and skill sets.
<table>
<thead>
<tr>
<th>Strategy</th>
<th>Benefits</th>
</tr>
</thead>
</table>
| Develop / Deploy New LOB Applications on Postgres Plus Advanced Server | • Significant cost savings for non mission critical systems  
• Leverages all existing Oracle skills  
• Very low risk |
Also, Postgres Plus Advanced Server has built in support for the OCI, assuring you that applications written in C or C++ will still run.

- **Migration Tools.** Functional compatibility is great but not if the conversion effort is cost prohibitive. Postgres Plus Advanced Server provides automated tools to move Oracle schema, data, packages, triggers, stored procedures, and functions to Postgres Plus Advanced Server in one step.

EnterpriseDB’s Postgres Plus Advanced Server products offers Oracle compatibility to a degree never seen in other databases. The migration toolset simply takes Oracle objects and recreates them automatically in the Postgres Plus Advanced Server database. The tool will also migrate your data so a minimal amount of manual work is necessary.

---

**An Oracle Migration Walk-Through**

This section presents the steps involved in an Oracle to Postgres Plus Advanced Server migration. In general a migration progresses through the following high level phases:

- Preparation
- Execution
- Validation
- Testing
- Tuning

**Migration Preparation**

In this first phase of the migration process, the servers and software are setup and configured. The setup requires the target Postgres Plus Advanced Server to be installed on a server with network connectivity to the source Oracle server.
After installing Advanced Server and starting the server, you need to copy the Oracle JDBC jar file from the Oracle installation directory to the jre/ext directory in the Postgres Plus Advanced Server home directory. Putting the jar file in this directory enables the Migration Studio tool to find the driver and use it to connect to Oracle. The following steps illustrate this process.

**Step 1: Prepare Connection to Oracle**

In this step, the Oracle JDBC driver is copied from the Oracle database home to the appropriate directory in the Advanced Server home directory.

Change into the Oracle home directory, in this case we are using Oracle XE version 10.2.0 and locate the JDBC directory. The driver is located in the lib subdirectory as shown below:

```
$ cd /usr/lib/oracle/x86_64/server/jdbc/lib
$ ls -a
```

```
total 28744
-rwxr-xr-x 1 oracle dba 4096 2008-09-08 17:27 ojdbc14.jar
-rwxr-xr-x 1 oracle dba 4096 2008-09-08 08:00 ...
```

Next, the ojdbcX.jar file is copied to the jre/lib/ext directory in the Postgres Plus Advanced Server home:
Once the JDBC jar is in place the Migration Studio tool is ready to establish a connection to the source Oracle database.

**Step 2: Establish Oracle Connections**

Start up the EnterpriseDB Migration Studio by selecting ‘Migration Studio’ from the Postgres Plus Advanced Server 8.3R2 Program group in the Start menu.

Once the Migration Studio is started, a connection to Advanced Server needs to be added if it is not already configured. Right Click on Servers and select ‘Add Server’ and then click on the New button in the Add Server dialog. Enter the Postgres Plus Advanced Server connection information for your database as illustrated below:
To ensure that the connection information entered is correct, the Test button is used which produces the following dialog when the connection is working:

Then the Oracle database connection needs to be setup. Right click on Servers again and click New.
In the Server Type pull down menu, select Oracle. Then enter the Oracle connection information:

![Oracle Server Connection](image)

Test this connection to make sure everything is working and you should see the success message again.

![Oracle Connection Test](image)

As a final task in the preparation phase, the objects to be migrated need to be determined. The Migration Studio has the capabilities to migrate entire schemas, groups of objects such as all tables or individual objects. The following illustrates how Migration Studio is used to browse an Oracle database and choose what objects to migrate.

**Step 3: Browse and Choose Oracle Objects to Migrate**
Using the object browser one can navigate from the Oracle database connection folder to the Databases folder and finally the schema folder. For this example, we will be using the sample HR schema that is included with all Oracle XE installations. This schema contains 7 tables, 2 procedures, 1 view, 2 triggers and 3 sequences.

If you continue to expand the HR schema folder, you will see folders for all the tables, views, packages, procedures, functions and sequences in the HR schema as show below.

By highlighting the HR folder, the entire HR schema can be migrated but you have the choice to migrate individual objects or groups of objects.

The decision to move only the schema / object definitions or move the data as part of the migration needs to be made. The Migration Studio offers the ability to move the data separately from the schema / objects. It is recommended that the first pass through the migration be done with the schema / objects only and then once that part is complete, return and migrate the data.

The reason for this 2 step process is to allow you time to resolve any issues with the schema / object creation that might affect the data load. For instance, one possible issue is if a column name from the Oracle system is a reserved word in Postgres Plus Advanced Server. If this were to happen, the table creation would fail and thus the data load would fail as well.
Migration Execution

The process of performing the migration is very straightforward. At this point, the Migration Wizard, which is part of the Migration Studio would be utilized. The following steps illustrate a two step migration where the schema/object definitions are migrated first and then, if successful, the data is moved.

Step 1 Open the Migration Wizard

Once the schema / object to be migrated is found in the browser, right click on the object name or folder name and choose 'Online Migration'. This will display the Migration Wizard dialog.

The Migration Wizard contains choices for:

- The target Postgres Plus Advanced Server
- The destination database in that server
- The source schema to be migrated (the same schema will be created in the destination database)
- The delimiter used to separate columns of data during the migration
- Migrating schema and data or just the schema

Typically, the only changes you will make here are:

- Changing the delimiter character if you know the default delimiter occurs in your data, or
- Selecting Definition Only if you plan to migrate the schema first and return later for the data.

The following figure shows that the source HR schema is about to be migrated to the Postgres Plus Advanced Server running on the localhost on port 5444. The migrated HR schema will be created in the edb database. If you wish to migrate to a different database in this cluster, you can choose the database from the drop down menu.

The default delimiter is acceptable in this walk-through since it does not exist in our data. Since we are illustrating a two step migration, the 'Definition Only' check box has been checked.
Once you are ready to start the migration, click Run. The migration will be performed and a running log of the tools activity will be displayed. Scrolling down to the bottom of the log allows you to see a message indicating the success of the migration.
After the Migration Wizard completes, the migration log should be reviewed to check for errors that might have occurred.

Errors during the migration typically fall into the following categories:

- Reserved word conflicts
- Feature implementation differences
- Features not currently supported
Reserved word conflicts occur when a column of a table is named with a reserved word used in Postgres Plus Advanced Server. Common cases of this include the words *time* and *date*. The workaround is to either change the name of the column, (i.e. use time_c) or put the column name in double quotes ("date").

Feature implementation differences refer to features that both databases support but with different syntax. For example, both databases support range and list partitioning but the syntax and steps to create a partitioned table are different in each database. In this case, the Migration Wizard migrates the table as a Postgres partitioned table using the correct syntax and functionality available in Postgres Plus Advanced Server.

Other feature implementation differences include full text search or spatial data support. The Postgres database has had support for these features for quite some time but the implementation is quite different than Oracle's and thus the Migration Wizard is not able to migrate the functionality over. It does migrate the data(the text data) over but you may need to change the data types to take more optimal advantage of the Postgres functionality.

Then there are the features that Postgres Plus Advanced Server simply doesn't support yet. Features in this category include things like Automatic Storage Management (ASM), Flashback database, database resource manager, table compression, Advanced Queuing, external tables, materialized views and analytic functions.

Automatic Storage Management (ASM) is the ability of the database to act as it's own volume management system. Postgres doesn't have this and relies on the volume management system of the hardware for this functionality.

Table compression can be accomplished by putting those tables in a tablespace that resides on a compressed filesystem. This is supported by Postgres.

Advanced Queuing, although not built into the database can be and has been worked around using external messaging systems such as ActiveMQ, TIBCO® or MQ Series so those are options for you if you use AQ.

External tables will need to be loaded into staging tables in the database. Using EnterpriseDB's EDB*Loader tool to load the data quickly is the recommended approach.

Materialized views exist in the form of summary tables that can be maintained by triggers, similar to the Oracle implementation but the setup
is manual and not Oracle compatible. Automatic query rewrite is not currently supported so your application will need to be made aware of the summary table’s existence.

Analytic Functions like lag, lead and dense_rank are not supported in the 8.3x releases of Postgres Plus Advanced Server. These have been introduced in the PostgreSQL 8.4 release so they will be available in the 8.4 release of Postgres Plus Advanced Server.

If errors are encountered during migration, EnterpriseDB can be quickly notified by emailing us at migrations@enterprisedb.com. We are continuously enhancing our compatibility and input from customers is invaluable in helping us prioritize our development plans and helping customers troubleshoot new issues.

After any errors are resolved and the schema / objects are successfully migrated, it is time to migrate the data from the Oracle database to Postgres Plus Advanced Server. This process is similar to the earlier migration step only this time you will navigate to the Table folder in the browser and right click on that Table folder.
After choosing *Online Migration* you will once again be entering the Migration Wizard. As you will see, there is only a slight change in its look. You will want to check the box next to 'Data Only' as shown below.
Once this is done, select Run and the data migration will begin. Again, check the log for any errors that may have occurred during the data migration.

Note: Make sure table(s) exist on target node.
The amount of time that the migration takes will be dependent on the amount of data to be moved and the speed of the connection between the two databases. For larger databases, another method of moving the data can be used that uses Oracle's native OCI connectivity instead of JDBC. This can speed up the migration substantially.

In order to use the OCI connectivity, you will need to setup one of the forms of database links that Postgres Plus Advanced Server supports, i.e. dblink_ora or Oracle style database links. The decision as to which type of link to use will depend on which version of Postgres Plus Advanced Server you are using as the Oracle database links are only supported in version 8.3. Setup of these database link methods require additional steps that are outside of the scope of this paper. If you would like to learn more about these methods, please contact an EnterpriseDB representative at migrations@enterprisedb.com.

There are rarely any errors during the data migration. The most common error encountered at this stage is when the chosen delimiter occurs in the data resulting in an error for each offending row (usually stating that extra characters were found at the end of the record).

Care needs to be taken to choose the proper delimiter. The default delimiter is a pipe '|' and that works in the majority of the cases. If there is a failure due to this delimiter choice and you need to try another delimiter, simply go back into the Migration Studio, browse to the offending table, right click on the table name, choose Migration Wizard, change the delimiter and run the migration process again.

**Migration Validation**

You can now browse the Advanced Server database and see the migrated objects.
You can select a table name, for example employees, and right click on it and choose 'View Data' from the popup menu. This brings up the built in SQL Interactive tool and allows you to run queries and view the data in the migrated table.
An additional task that should be performed to validate the success of the migration from a database point of view is to run SQL statements in both Oracle and Postgres Plus Advanced Server, gather the record counts of each table. Assuming the data in Oracle hasn't changed, these record counts should be equal.

**Migration Testing**

Validating the migration from a database point of view is only one part of the necessary testing. A vital part of proving that the migration was successful is to test the new system using the same application that was used against the source Oracle system. This step obviously involves the porting of the application so that it can connect to Postgres Plus Advanced Server. For Java applications, that simply involves changing the JDBC driver and the connection URL. For ODBC applications, create an EnterpriseDB data source and tell the application to use it instead of Oracle's datasource.

One common concern or question asked by many potential migration customers is 'What about the embedded SQL statements in my application?'. Advanced Server contains a compatibility parser that understands the Oracle flavor of SQL and the majority of the extensions that Oracle has added to its SQL support. Thus, those embedded statements don't need to modified!
Other applications such as those written in C would require additional steps depending on which Oracle API it was coded to utilize. For more information on this topic, please contact an EnterpriseDB engineer at migrations@enterprisedb.com.

**Tuning the Migrated System**

The last step in any migration is to make adjustments to the new system in order to get the best possible performance. Most of the tuning steps will revolve around changing database configuration parameters to resemble those of the original Oracle system. One may wish to add additional tablespaces to the Advanced Server database as a way to move tables around and spread out the I/O of the system.

Other tuning steps might include using a different type of index, such as partial indexes or using partitioning where it wasn't used before since that option was not purchased for the Oracle system. Partitioning is a core feature of Postgres Plus Advanced Server. By using partitioning, you get the performance benefit of partition elimination which allows the database to scan smaller chunks of data to retrieve your information resulting in much faster response times.

**Conclusion**

Traditional migrations present many obstacles. When a translation from one database to another is done, there are many syntactical issues that arise that make the migration difficult, time consuming and results in suboptimal code.

EnterpriseDB’s Oracle compatible Postgres Plus Advanced Server eliminates these obstacles since a translation is not being performed. Instead, the same syntax is being used to create the database objects in the new database system and thus the errors and required time that cause most migration projects to be halted or never even get started, are avoided.

This paper illustrated the steps in the migration process and demonstrated a sample migration. You now have the knowledge necessary to begin investigating you own migration. EnterpriseDB has the expertise to assist you during your project. If you would like to discuss your project in more details with an EnterpriseDB engineer, please contact us at migrations@enterprisedb.com.
For a complete TCO and Technical analysis of migrating your enterprise from Oracle to Postgres Plus Advanced Server, please contact EnterpriseDB at: https://www.enterprisedb.com/about/contact_us.do or contact the Sales department at: sales-us@enterprisedb.com (US), sales-intl@enterprisedb.com (International), or call +1-732-331-1315, 1-877-377-4352.

About EnterpriseDB

EnterpriseDB is the leading provider of enterprise class products and services based on PostgreSQL, the world's most advanced open source database. The company's Postgres Plus products are ideally suited for transaction-intensive applications requiring superior performance, massive scalability, and compatibility with proprietary database products. Postgres Plus also provides an economical open source alternative or complement to proprietary databases without sacrificing features or quality. EnterpriseDB has offices in North America, Europe, and Asia. The company was founded in 2004 and is headquartered in Edison, N.J. For more information, please call +1-732-331-1300 or visit http://www.enterprisedb.com.