

Oracle Database 11g Production Case Study: Porting to Oracle Database 11g Tips, Techniques, and Experiences

Marcel Kratochvil
marcel@piction.com
CTO Piction

Migrating to Oracle Database 11g is not difficult, but there are some points to consider when doing the migration, especially if that migration is from older releases of Oracle including Oracle8.

This paper will cover issues when migrating, migrating techniques, and how to migrate to 11g to take advantage of the new LOB features and performance improvements in PL/SQL. The paper will cover lessons learnt when migrating customers from Windows and Unix installs, to the latest Oracle release of the database.

This paper has been based on experiences gained since using Oracle 11g in its first beta testing release until now. It is also based on performing a number of live database upgrades on customer sites. These were customers using Piction, which now has customers in Australia, New Zealand, the US and the UK. Upgrades were done on Windows and Unix, and from Oracle8 to Oracle10.

Piction is a web based application used for the management of digital image assets. The company has been running since 2000, and now has customers using the product in Museums, Photo Labs, Galleries, Print Media, Marketing and Universities. Piction was ported to Oracle 11g during its first beta release and has been enhanced to take advantage of key Oracle 11g features. With a recent port of the application to Oracle XE, Piction now supports most multimedia types including Photos, Documents, Video and Audio. Customer use it as an Image Warehouse to an electronic shopfront allowing for the download, processing, management and selling of images. The application is predominantly written in PL/SQL but also uses Java.

Customer installs vary in size and complexity. Not all Piction customers have upgraded to Oracle 11g or will upgrade to Oracle 11g. For example one current customer is very happy on Oracle8 running on an older version of Piction, whilst another customer has over 3 million images on a Windows 2000 Workstation and has been running non-stop for 6 years. Some new customers are now insisting on using Oracle 11g with a new install. What determines the upgrade is based on the skill set of the DBAs and the complexity of the database environment they are in. Piction customers range from very small to very large, so some sites do not have DBAs. In this case Piction has been designed to run without DBAs and remote DBA management is done to monitor the database. A number of government departments using Piction will only upgrade when all applications in the department can be upgraded to the same database version.

What's stopping you from upgrading?

Based on listening to members of Oracle User Groups and customers using Oracle, the biggest impediment to upgrading to Oracle 11g is the complex list of upgrade dependencies. This includes having the right hardware, operating system version and support from software vendors. Gone are the days when all usage of the database was done by a team of local developers. In today's environment, a lot of software is purchased from vendors. In a number of cases, the software released by vendors is dependent on software from other vendors. Only when all the vendors have confirmed the software is supported on Oracle 11g, can it be upgraded. Even then it gets more complex, especially if there are conditions attached to the upgrade. It might require patches to be installed, or maybe an upgrade to a complete new version. This might then have a cascade effect of requiring new hardware, an upgrade to the operating system, or even an upgrade to the clients. The result is that upgrading is not that simple anymore.

To add to the complexity is if multiple applications are located in the one database. Upgrading the database means all applications will need to be upgraded at the same time. All the application owners will need to be in agreement that an upgrade is needed and all software needs to be Oracle 11g compliant.

Also, the adage if its not broke, why fix it? Applies to Oracle 11g. Most developers using Oracle 9 and Oracle 10 are in agreement that these versions are mature and provide sufficient functionality for their programming environment. Trying to justify to users to go through the pain, cost and testing, just to upgrade to the latest version of Oracle can be very difficult, especially if the only reasons are that it might run faster and there are some cool new features in it.

So the end result is that if a successful upgrade path is found to move to Oracle 11g, there will likely be resistance from the users who will not see any real benefit and only cost in upgrading. This means that the requirement to upgrade will need to be sold to the users.

Finally one of the biggest impediments to upgrading is the view in the user community of not upgrading until Oracle has released the second release in a version. From Oracle 8 to Oracle10, the feeling is that stable releases only appear in the second release. This means that the view from the Oracle user community is that they are waiting for 11gR2 before seriously thinking about upgrading.

That following are some of the selling points that can be used to justify the upgrade to Oracle 11g:

When selling to DBAs, Developers and System Administrators:

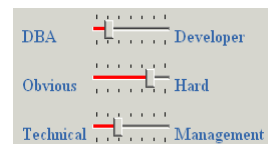
1. Its faster than Oracle 10g
2. It will work with the latest releases of Windows, Solaris and Linux
3. Its architecture is better suited for virtualization
4. There are more built-in tools to assist in debugging and managing the database, PL/SQL and Java
5. Its more secure
- 6.

When selling to Managers and Users:

1. Oracle has desupported Oracle9 and early releases of Oracle10. Its more cost effective to migrate now in a careful, controlled structure, rather than later when skill sets and knowledge in using the older release is lost.
2. Its more cost effective to support a later release than an older one.
3. A new feature allows stateful HTML applications to be built, meaning friendlier and more powerful HTML applications.
4. New applications being release by vendors will only work on Oracle11g.
- 7.
- 8.
- 9.
10. The following section now lists the different features of the database and offers guidance and experience when upgrading.

Character Set

Customer lesson learnt:



If you are using Oracle 10, I recommend that the character set used by the database is one that supports UTF-8.

The default offered by Oracle in the older releases should not be used.

Why UTF-8? Mainly so that the database inherently supports XML. Experience has shown that most applications are moving to, or have support for web services. To support this data set correctly requires having support for the UTF-8 character set. Problem was, in older versions of Oracle a number of Oracle features did not work correctly with the UTF-8 character set, so the DBAs would install the database with the default character set.

With Oracle 11g its easier to install the database with UTF-8 support. The experience seen is that this will cause problems. In two upgrades done (one from Oracle8), both had to be redone because only after the upgrade did some parts of the application start to behave differently. The problem areas were ones that dealt with the storage of clob's and the data in them.

So the lesson is that if you are migrating, keep to the character set in the old database. If you are creating a new application in a fresh new database, use a UTF-8 compliant character set.

New PL/SQL Packages

Sometimes in each new release, Oracle introduces a new feature and they don't detail what it can do, but that feature is exceptional in what it can accomplish. In Oracle 11g the dbms_xa package has been made available to PL/SQL. Its a feature that has been around in various forms since Oracle7. Now that it can be accessed from PL/SQL it opens up how we can access the internet. The internet goes from stateless to stateful.

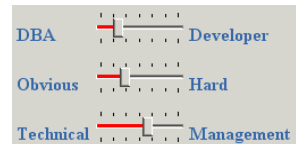
For those who have no idea what this means. Simply, a transaction can now span multiple html pages. Mod PL/SQL (and subsequently Java and PHP) can now offer transactional locking (and timeouts). Previous features required to manage row locking used hashing rows or checking for row timestamps. They are now redundant. You can now wrap transactions in the dbms_xa package calls

(very easy to do), and have true transactional management.

This feature is so powerful, it will change the nature of how we use the internet with Oracle. A bold statement, but it means that true transactional systems can now be ported to HTML. All they need is some friendly interfaces to assist with high speed data entry and the requirement for using client tools is now redundant. No need for a middle tier, no need to use form based tools, you can now build any type of transactional application in HTML (with mod PL/SQL) and be guaranteed true transactional support.

PL/SQL Compiler

Customer lesson learnt:



Two lessons learnt if you want PL/SQL compiled down to whats called native. Native means taking the PL/SQL, converting it to C and compiling the C code. Its the PL/SQL equivalent of the Java Just in Time Compiler (JIT).

i. Make sure when you create the database, you specify the init.ora parameter PLSQL_CODE_TYPE=native, before you create the database. Setting it after is annoying as you have to reinstall PL/SQL to ensure all the PL/SQL packages are compiled natively.

ii. All the PL/SQL is stored compiled in the SYSTEM tablespace. Its not stored externally in directories like it was in previous versions. The result is that on database creation, create SYSTEM with at least 1Gb of storage, preferred is 1.5Gb.

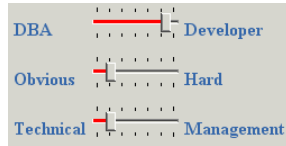
But is PL/SQL compiled actually faster?

Initially testing showed that compiled PL/SQL when used in mod PL/SQL was not noticeably faster. What one has to realise is that PL/SQL already ran very fast in its pseudo code compiled mode, so to see the benefit of compilation, involves doing tests that compiling to native would highlight. These tests involve mathematical functions (e.g. rotate an image). In these tests, the

performance difference of compiling to native was noticeably better.

SQL Statements

Customer lesson learnt:



Any select statement that references an object needs an alias to reference it, and an alias name to make it unique.

So if your code was:

```
select myimage.source.localdata from myimage;
```

should be changed to

```
select u.myimage.source.localdata blb from myimage u;
```

In 8i to 10G when Oracle retrieved data from an index it kept it sorted. In fact, it was a useful tip to remove ordering and retrieve data just from the index.

From Oracle 11g this cannot be guaranteed. The optimizer now has new algorithms for doing queries and it might not mean the data comes back sorted.

The solution is to put an order by on the statements.

Workaround? Put an order by on SQL Statements. Yep, its painful.

At this point of time I am not aware of any init.ora parameter or lingusitic setting that forces a sort.

The SQL Standard does say that to ensure ordering using an Order by. This doesn't help if you have 3rd party apps or if you can track down the source for your app.

I don't think Oracle realise that this one cosmetic issue will cause most sites to not upgrade, as its this cosmetic issue that causes users the most heartache.

This issue has impacted upgrades I have done and made me rethink coding strategies when trying to build an app that works across multiple Oracle releases.

```
---- Oracle 8i
SQL>create table test (coll varchar2(100));
SQL>insert into test values ('BBB');
```

```
SQL>insert into test values ('AAA');
SQL>insert into test values ('KKK');
SQL>insert into test values ('CCC');
SQL>commit;
```

```
SQL>select distinct coll from test;
```

```
COL1
-----
AAA
BBB
CCC
KKK
```

4 rows selected.

```
SQL>show parameter opt
```

```
NAME TYPE VALUE
-----
object_cache_optimal_size integer 102400
optimizer_features_enable string 8.1.7
optimizer_index_caching integer 0
optimizer_index_cost_adj integer 100
optimizer_max_permutations integer 80000
optimizer_mode string FIRST_ROWS
optimizer_percent_parallel integer 0
```

---- Oracle 11g

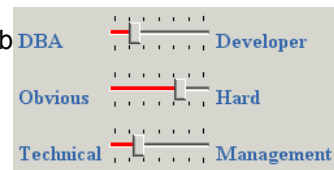
```
SQL> select distinct coll from test;
```

```
COL1
-----
BBB
AAA
CCC
KKK
```

4 rows selected.

Lobs: Securefiles

Securefiles are the new faster lob management architecture which comes bundled in with Oracle 11g.



From the developer, there is no difference between Securefiles and the Basicfiles (older 10g method) except that securefiles are faster. For the DBAs, Securefiles offer built in security and compression, so it is a good idea to migrate to use them.

There is no migration tool to migrate the lobs.

A table alter command is required:

e.g.

```

alter table umo move
  LOB (image_thumbnail.source.localdata)
STORE AS SECUREFILE 1 image_thumbnail
  (TABLESPACE PICTURE_IMG_1 enable
storage in row
  RETENTION AUTO
  NOCOMPRESS
  KEEP_DUPLICATES
  STORAGE (MAXEXTENTS UNLIMITED
PCTINCREASE 0)
  CACHE LOGGING);

```

This command will migrate over to use the new Securefile storage.

MultiMedia

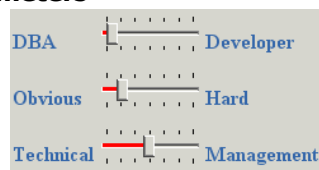
With Oracle Multimedia, the upgrade challenge is to convert the lob to securefile. This ensures Oracle Multimedia performs to its optimal.

With Oracle 11g, Oracle Multimedia has greater support for large images, especially JPEGs which can now be processed if they are more than 2mb in size.

With the Java Just in time Compiler now built into the database, its best when a database is created, to run some dummy loads that used Oracle Multimedia, to force the Java JIT to compile all relevant Oracle Multimedia routines.

Database Parameters

Customer
lesson learnt.



With Oracle 11g, the choice when using the cost based optimizer comes down to a simple two: *first_rows* and *all_rows*. With *first_rows* you can define an approximate number of rows that are expected to be returned.

For legacy applications not designed to work with the cost based optimizer, now that you are forced to use it, its best to choose the *first_rows* option, as this will give you the closest proximity to behavior that was available with rule. For those who built their apps to use *choose*, then its best to move to *all_rows*.

There can be a lot of debate about which option is ultimately the best to use, but the simple rule I follow is that I used *first_rows* when the application is OLTP, and *all_rows* when the application is OLAP or similar to a datawarehouse. For databases that are a mixture of the two, the alter session command can be used in a pre-login trigger to set the optimizer mode based on the type of user. If this still isn't enough to differentiate, set the database to *first_rows* and modify all SQL statements to *all_rows* that will return a large number of rows. If its not possible to access the source code to change the statements, then clever use of triggers might allow you to switch between the various optimizer settings. By the time you get to this point, you will be in the need to do some serious tuning and look at other configuration options.

There are now a lot more background processes when the database starts up. With Oracle8 there were 4 main ones: DBWR, LGWR, SMON and PMON. On Oracle 11g there are over 20 now that can start up depending on configuration settings. Any external programs/scripts that the DBAs wrote to monitor the health of the database by looking at these processes will need to be enhanced to take into account these new processes.

A couple of new parameters:

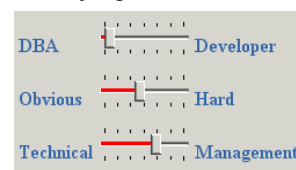
DDL_LOCK_TIMEOUT

Should be configured to a non 0 value. Its used to enable timeout on any DDL statement. So rather than hanging, the statement will fail after the period of seconds specified. For DBAs this means that doing upgrades of applications will make it more obvious when there is a locking failure.

JAVA_JIT_ENABLED

This new parameter enables automatic compilation of commonly used Java routines. By default its enabled. This means that when the database is first used, the JIT might fire and compile one or more Java programs. This might be disconcerting if the DBA is focused on initial tuning and it is trying to track down unknown CPU spikes.

Embedded
Gateway



To use Apache or use the embedded gateway?

For those moving to a web based environment (and nearly everyone is), this question will need to be answered. The embedded gateway came in with 10gR2 as a lightweight webserver, built into the database, designed to run PL/SQL (yes it came in with XML, but in reality it was there for Mod PL/SQL apps). It doesn't have anywhere near the features Apache has, but it is simple to configure, is configured in the database, and removes the need for a middle tier. Using the gateway means there is no need to use/configure Apache (great for sites that are Microsoft) and manage password files.

The downside is configuration. There is no way to perform URL redirects and access the file system to access static icons (in regards to this, there are a number of workarounds, from building one yourself to using XML DB). There is also no PHP, Perl or LDAP access.

So, if you only have a Mod PL/SQL application (or using Oracle Express – Apex), the gateway is a great way to go. If not, then you will need to look at using Apache.

Apache 2

With the release of Oracle 11g, Oracle also released support for Apache 2.0. Users of Apache have been waiting a long time for support of this release, and with it Oracle has re-architected the directory structure of the install to be more logical and consistent.

The install comes from a separate download (Oracle HTTP Server (Apache 2.0) (10.1.3.3.0) for Microsoft Windows (32-bit))

On install, the Mod PL/SQL and Apache files are stored in \$ORACLE_HOME/ohs e.g. with the Apache 1.3 install the httpd.conf file was found in:

\$ORACLE_HOME/apache/xxx
where in Apache 2.0 the same file is found in:

\$ORACLE_HOME/ohs/conf

The PL/SQL Dad configuration file is found in \$ORACLE_HOME/ohs/modplsql/conf

The only issue found was that the notes for the DAD configuration file are for 1.3 and not 2.0. (see dadTool.README). For Windows the location of the Perl program for encrypting the passwords is referenced as:

```
set ORACLE_HOME=new_ORACLE_HOME_path
set PATH=%ORACLE_HOME
%\Apache\modplsql\conf;%PATH%
set PATH=%ORACLE_HOME
%\perl\5.6.1\bin\MSWin32-x86;%PATH%
```

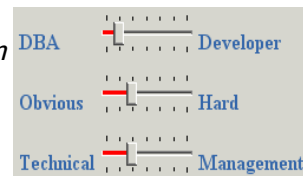
when it should be:

```
set ORACLE_HOME=new_ORACLE_HOME_path
set
PATH=P:\oracle\apache2\ohs\modplsql\conf;
%PATH%
set PATH=%ORACLE_HOME
%\perl\5.8.3\bin\MSWin32-x86;%PATH%
```

The latest Apache release is 2.2, and there is no official planned release date for support of this release.

Alert Logs

Customer lesson learnt:



The two big mistake made with the alert logs has been not to configure the default location to something familiar and not to specify a max dumpfile size.

With Oracle 11g, the location of the alert logs has changed to the value of the init.ora value:

diagnostic_dest

e.g.

diagnostic_dest = /u01 or
diagnostic_dest = p:\oracle

Oracle will create a series of directories underneath this that is good for diagnosing issues, but is daunting for first time users. On windows, to find the alert.log, it is found in:

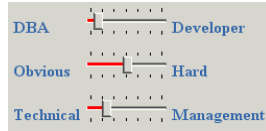
P:\oracle\diag\rdbms\vnla\vnla\trace

it is also found in :

P:\oracle\diag\rdbms\vnla\vnla>alert with the file log.xml. This isn't a valid XML file, it contains the alert messages marked up with XML tags.

Installation

The installation of the Oracle software is very similar to Oracle10. The only difference is that there is new screen which will allow you to link directly to metalink. This is a new feature which has been discussed by Oracle before. The important lesson, is that if you want to use this feature ensure you have your metalink customer details handy.



The creation of the Oracle database now requires more memory. A minimum of 1Gb is needed, but 2Gb is recommended. Actually memory required is dependent on features used. If Java is used (and note that PL/SQL supplied packages are mostly written in Java), then the recommendation is 2Gb minimum to ensure the database runs properly.

For those Oracle users still on a 32bit system, this is getting uncomfortably close to the magical 3Gb memory limit. Some operating systems / hardware can extend this beyond 3Gb, but with Oracle 11g now is the time to start thinking about 64 bit operating systems.

The install of Oracle 11g requires more disk space (allow 2Gb), and the bare minimum size for the database should be 10Gb. But storage is cheap and now that 1Tb sized disks are out 10Gb is tiny in comparison, so don't get miserly on disk storage.

When using Oracle Multimedia, its best to create the database with the following minimum values:

SYSTEM – 1Gb
SYSAUX - 1Gb
UNDO – 20Gb
TEMPORARY – 30Gb
Redo Logs – 10 x 100Mb

32bit vs 64bit

When looking at Linux vs Windows (and leaving Solaris out of it, as it has had 64bit support for a long time), there is now an

attraction to go to a 64 bit platform, especially if a low cost solution is required. As 64bit supports more than 3Gb of memory, there is more flexibility with database management if the Oracle database can grab a large amount of memory.

For under \$2000 you can buy a chassis, motherboard, 64bit CPU with 8Gb of memory and a couple of 500Gb disks. The experience gained from using Oracle Enterprise Linux 64bit is that there are still a number of programs that have not been ported from 32bit to 64bit, especially video drivers, network drivers and disk drivers. This means that on install, you will have to compile them in yourself or track them down on the internet. For a novice Linux user, the cost of doing this would exceed the cost of the hardware. So the recommendation is to ensure the hardware platform already has support for the drivers on purchase, which will drive the price up.

The other downside is a number of linux routines do not yet run in 64bit. When I did my last EL5 install, Samba was not available making it very hard to work in the o/s. Of course this will change over time, but the key message was that the linux migration was not mature for low cost off the shelf implementations (now if you are a seasoned Linux user and live by it, this isn't an issue).

For Windows, the 64bit version of XP came with most drivers and had a good level of support for 32bit routines. Over the course of 3 months, Microsoft upgrades added more core routines that had been migrated to 64bit. The only serious issue was with VNC causing catastrophic and random machine failure. The workaround was to use remote desktop. The Windows XP 64bit version with Oracle 64bit, works very well and is easy to use and is reliable.

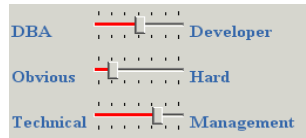
Enterprise Manager

There is no requirement to run with Oracle Enterprise Manager. It does come nicely bundled with the database and installs easily, but if you are running a site that uses web based applications, you will need to ensure it is tightly reigned in at the firewall so only authorised users can access it.

For DBAs managing a large number of databases at the one site, Enterprise Manager offers a great solution for management. It is not designed for novice DBAs and the interface can be initially daunting. If you are an experienced DBA, the interface is more intuitive to use. With all the new features in Oracle 11g, experienced DBAs who are new to 11g might want to use Enterprise Manager to ease them through the transition process of getting acclimatised to the version.

The Little Thing

SQL*Plus on Windows. There is no sqlplusw.exe,



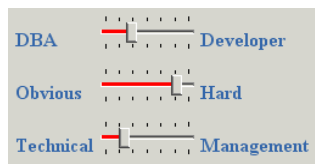
which was the gui-ish version for windows.

The standard DOS sqlplus.exe is the only one available. This one is easier to use, as you can use DOS command line editing and DOS retrieval of previous commands. All desktops that had previously referred to sqlplusw.exe need to be changed to sqlplus.exe.

The other minor issue is that there is no display icon for the DOS version on the desktop, and displaying a standard DOS icon is boring. But that's just a little thing.

Security

1. Customer lesson learnt:



A new privilege has come in Oracle 11g that might bite you if you use utl_tcp (utl_smtp, utl_mail, utl_http and other tcp/ip based tools). The new dbms_network_acl_admin package allows fine grained control over access to the network. On upgrade to Oracle 11g its not obvious that you need this privilege. On two separate upgrades this problem appeared. Testing had been done except on sending emails because the test server didn't have SMTP support. The assumption was that it worked, which is a dangerous assumption to make.

On production upgrade, emails were not sent out, users complained and chaos ensued. A quick review discovered the issue and the ACL commands (see below) were run to give access to the network.

If you just want to migrate and get things working, the following will give a schema full network access from within the database:

```
BEGIN
-- required for Oracle11 to access network
DBMS_NETWORK_ACL_ADMIN.CREATE_ACL(
acl => 'myapp.xml',
description => 'Network permissions for *',
principal => 'SCOTT',
is_grant => TRUE,
privilege => 'connect');
END;
/
commit;

BEGIN
DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL(
acl => 'myapp.xml',
host => '*');
END;
/
commit;

exec
DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE('picti
on.xml','SCOTT', TRUE, 'resolve');
commit;

SELECT
DECODE( DBMS_NETWORK_ACL_ADMIN.CHECK_PRIVIL
EGE( 'myapp.xml', 'SCOTT', 'resolve'), 1,
'GRANTED', 0, 'DENIED', NULL) PRIVILEGE
FROM DUAL;
```

2.For all Oracle accounts the default profile has been changed to make it more secure. This means that Password_Lock_Time, Password_Grace_Time, Password_Life_Time and Failed_Login_Attempts now have values assigned to them.

This might cause issues for DBAs who are not expecting to suddenly hear from users who complain about not being able to login.

More importantly, if an application (e.g. Forms) is not written to handle errors from logging in, then it might start failing on login without correctly notifying the user.

Summary

So the question everyone wants an answer to, is should you upgrade to Oracle 11g, wait for 11gR2 or just not upgrade at all?

Not upgrading is cost effective in the short term and ensures short term stability. Experience has shown that if you do not upgrade on a regular basis (e.g. every 2 years), then the cost to upgrade when you are

eventually forced to upgrade will be greater than upgrading every two years.

This cost comes in paying for vendor support to do a large upgrade, cost with having to get the users to do a lot of testing, the cost with having to likely do application changes and the cost in downtime and added complexity. The debating question is, is it easier to do one large upgrade every 10 years, or a series of smaller upgrades every 2 years?

With my DBA hat on, I would recommend doing upgrades on a regular basis. This way everyone gets used to them happening, budget for it, and have the mindset in place to

do them and be happy to do them. There is nothing worse than having to force the users of an application in to doing an upgrade, as it will likely end in failure.

So sell to the users in your organization the benefits of upgrading to Oracle 11g and look to migrate to it sooner rather than later.

Marcel Kratochvil
CTO Piction
marcel@piction.com
<http://www.piction.com>
<http://www.eternal-donut.com>