



# Cross Platform Migration of a Large Production Database

By Piers Cherryl

***This article describes how The Wet Seal, Inc. migrated a 300GB production database from HP-UX to IBM AIX: the issues involved and how we addressed them. The goal is to describe a real world business problem, the options available to address it, and an outline of the solution with a couple of the key techniques used. Be aware, however, that this approach requires tuning and testing for any given situation, and 'your mileage may vary' applies.***

In early 2002 The Wet Seal, Inc. decided to undertake a major overhaul of its MIS infrastructure that included moving key production systems to a new, more secure data center located several miles from its main offices and connected by a Gigabit Ethernet Metropolitan Area Network. It was decided that this would also be a good opportunity to upgrade to new servers as these could be installed at the new facility and tested with a minimum of disruption to the business. The challenge posed to the DBA team was — were they locked into staying with their existing vendor, HP, or could they strengthen their negotiating position by demonstrating their willingness and ability to switch to a new vendor? Over the course of ten weeks, with the help of a loaned pSeries server from IBM, they planned and implemented a migration that demonstrated the risk and disruption involved in moving a large production database to a new vendor platform could be reasonably managed. By their reckoning the subsequent competition for their business saved Wet Seal hundreds of thousands of dollars in server hardware costs while significantly boosting capacity. In the end they did indeed switch to IBM and the production migration went as smoothly as the trial had predicted.

So what is the big deal about changing vendor platform? The problem is that Oracle datafiles are not guaranteed binary compatible with different operating systems and hardware. This means that one cannot simply copy Oracle datafiles from a machine with PA RISC CPUs to a machine with POWER CPUs and expect it to work. Even if by coincidence it seems to work Oracle Support Services would not be required to help resolve any subsequent issues. According to Oracle's Migration manual this leaves two options: using Oracle's export/import utilities or data copying using Net8 and database links.

Oracle's export utility is normally used to create logical backups of databases, schemas or individual tables into dump files on disk. Unlike Oracle datafiles these dump files are guaranteed portable across different hardware architectures. This allows exported data to be copied from, say a Windows NT server to a Solaris server, where they can then be imported into another database already existing on the new platform. However, by Oracle's own admission using export/import is extremely slow except for very small databases. This approach may also require large amounts of disk space for the dump files.

Oracle also suggests data copying using Net8 and database links but this is also slow and does not address the issue of migrating stored procedures, triggers, grants and constraints, etc.

Aside from Oracle's recommendations there are third party tools that can assist with database migrations. One tool in particular that we looked at seriously was Quest Software's Shareplex (<http://www.quest.com/shareplex/>). This is a sophisticated package that supports efficient data replication in real-time across different operating systems and versions of Oracle. After careful consideration, however, we decided that we could afford an outage of a few hours and thus achieve our objective with a cheaper, more cost effective approach.

The approach we eventually did adopt and refine to migrate large databases was to harness the Oracle export/import utilities with Unix scripts and a number of manual steps. This works well for a data warehouse where all updates are performed by nightly batch programs, and with care it can be applied to a large OLTP system as well. The main issues with this approach are how to minimize the production system downtime during the migration and how to migrate the data rapidly without using large amounts of disk space or tape. The steps involved are as follows:

- (1) Create the target database with relevant system, rollback and temporary tablespaces. Leave the database in 'no archive log' mode until after it has been populated.
- (2) Add empty tablespaces to the target database to match the source.

*continued on page 24*

- (3) Copy user account information from the source database to the target by using `tfscuser.sql` (Metalink Doc.Id.1019554.6)
- (4) Copy across database links and public synonyms.
- (5) Export/import schemas using the Oracle `exp` and `imp` utilities. Where the amount of data is large, say more than 100MB, export with `ROWS=N` to skip copying of table rows and import with `INDEXES=N` and `CONSTRAINTS=N`.
- (6) Export/import row data for schemas with large amounts of data using scripts, see below. Make sure that triggers are disabled before this step, and reenabled afterwards.
- (7) Build the indexes and constraints for the schemas in the previous step.
- (8) Run any appropriate analyze jobs and quality assurance tests.
- (9) Switch the database into logging mode, if appropriate, and perform a full backup of the new database.
- (10) Place any tablespaces into read-only mode as appropriate.

Of these steps (1) to (4) can be completed while the source database is still in use, assuming of course that it is relatively straight-forward to enforce a freeze of tablespace usage, user accounts and synonyms. It may also be possible to export/import some schemas without table row data while the source database is in use but beware that sequences are updated frequently and may have to be updated or regenerated after the migration. Large read-only tables or partitions in the source database may also be good candidates to migrate prior to the cutover.

At some point, however, updates to the source database must be halted, and the main bulk of the table row data migrated. The source database must remain open during the migration but most accounts should be locked and disconnected, or the database restarted in restricted mode. To minimize downtime we performed the table row migration by using Unix pipes to export and import data directly from one database to another across a high-speed network. Here is an example script fragment:

```
# This code runs on the target host.
# First rexec to the source host, create a pipe and export to it...
#
rexec $source_host "/usr/sbin/mknod $exp_pipe p"

rexec $source_host \
"export ORACLE_HOME=/opt/oracle/product/8.1.7; \
/opt/oracle/product/8.1.7/bin/exp \
system/$system_password@$source_SID tables=$source_schema.$table \
file=$exp_pipe log=$exp_pipe.log \
consistent='N' compress='N' direct='Y' recordlength=65535" &

# Create a local pipe and prepare to import from it.
#
mknod $imp_pipe p

imp system/$system_password fromuser=$source_schema \
touser=$target_schema tables=$table \
file=$imp_pipe log=$imp_pipe.log \
buffer=131072 commit='Y' ignore='Y' indexes='N' \
recordlength=65535 \
feedback=1000 skip_unusable_indexes=TRUE grants=N constraints=N &

# Connect the two pipes across the network
#
rexec $source_host "dd if=$exp_pipe" | dd "of=$imp_pipe"
wait
```

An introduction to using export/import with Unix pipes can be found in Metalink Doc.Id.30528.1. This approach also sidesteps issues with 2GB file

limits for Oracle dump files on some operating systems. The log files created by `exp` and `imp` should be grepped to verify the operation was successful.

Multiple tables or partitions can be copied in parallel by using a script that iterates through a list of all tables to be copied - ideally in decreasing order of size - and starts subshells to copy each one, pausing while a predetermined number of subshells are already running, for example:

```
export_import_tables() {
  while (( $# > 0 )); do
    table=$1
    shift 1

    while let subshells="$(ps -f|awk '{print$3}'|grep $$|wc -l)";
      (( $subshells > $max_concurrent_jobs ))
    do
      print "Pausing ($# remaining)"
      sleep $polling_interval
    done
    export_import_table $table &
  done
  wait
}
```

We chose to use a two-phase approach, migrating all the table rows before building indexes so that we could confirm as early as possible that all the data had been copied across successfully. Alternately, if the disk subsystem is properly configured it may be more efficient overall to leave `INDEXES=Y` and have data copying and index building going on for different tables simultaneously.

The SQL to build the indexes can be obtained using the `imp INDEXFILE` option. One can either obtain a single script for all the indexes in a schema, or separate scripts for each table. As we wanted to optimize performance we piped the SQL across for separate tables from the source database in a similar way to the data and built multiple indexes simultaneously. Performance can be further improved by building the indexes using the `NOLOGGING` and `PARALLEL` options. Beware that older versions of `exp` have bugs so thorough testing is essential and it may be necessary to find workarounds.

Once indexes have been built constraints can be created or reenabled. Again the SQL is available using the `imp INDEXFILE` option but needs to be massaged with `awk` or `sed`. Unique and check constraints for a table can be processed immediately after the table's indexes are complete but referential constraints must wait until all indexes are complete so we broke them out into their own processing phase.

Our experience was that building the indexes and constraints took almost as long as copying the table rows. We were able to migrate a 300GB database across a gigabit ethernet network with between six and seven hours downtime using a Symmetrix disk array and fully loaded IBM pSeries 6M1. The preparation was considerable and the cutover was certainly not simultaneous but the return on effort invested was well worth it.



### About the Author

**Piers Cherryl** is the manager of UNIX and database administration at The Wet Seal, Inc. an apparel retailer with 600 stores across the United States. He has 17 years IT experience and is certified to administer Oracle, DB2, AIX and HP-UX.