Database Migration for Oracle™: From Alpha Server Tru64 UNIX to Integrity Server HP-UX 11i V2

Eric L. Speed
DB Systems/Software Engineer
BCS – Transition Engineering & Consulting (TEC)
Hewlett-Packard

Objectives

- Offer useful information regarding Oracle Database Migration Planning and Design Methodology for:
  - Source database server platform = AlphaServer Tru64 UNIX
  - Target database server platform = Integrity HP-UX 11i
- Provide important information to be shared with those HP customers anticipating an Oracle database migration due to platform transition
- Examine HP recommendations, tools, best practices, and cautionary notes
- Prompt questions and discussion on migration alternatives, issues and concerns…

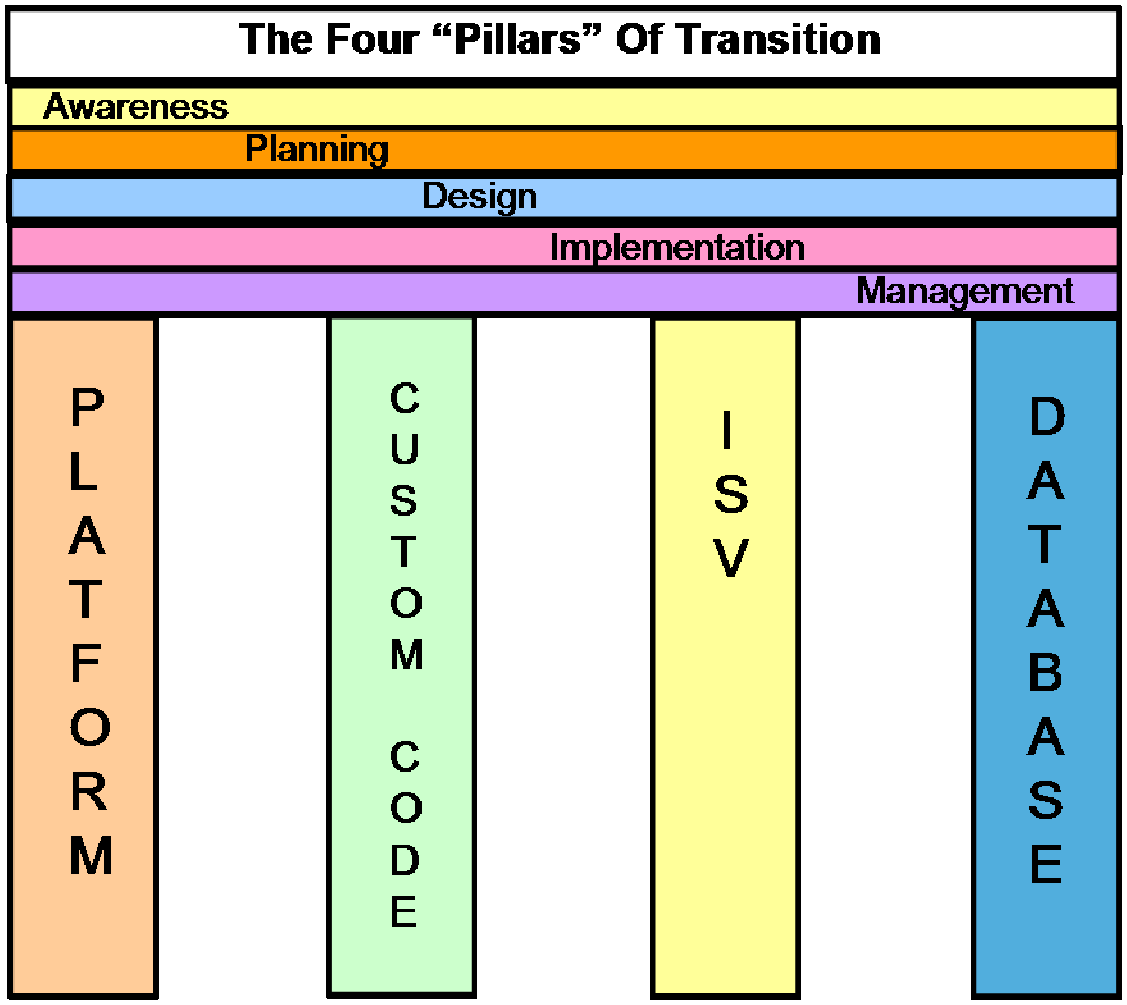
Oracle Database Migration – As a Component of a Platform Transition Project:

A High-Level View

A High-Level View

- Treat the platform transition, including database migration as a nothing less than a full-fledged **project**
- View database migration in synchronization with HP identified transition project phases:
  - **Awareness** – Identifying transition as a business and/or technology requirement
  - **Planning** – Defining the requirements, the team, the platform, the timeframe, the tools and the approach (Planning Assessment Documents)
  - **Design** – Create the actual migration procedure; drill down to detailed steps/tasks; prototyping/testing recommended
  - **Implementation** – Execute the prepared Design for the Planned timeframe
  - **Management** – Follow-up tasks; complete fine tuning, documentation, monitor and control the environment; re-adopt smooth production operations

A High-Level View



A High-Level View

- The Planning, Design and Implementation phases can be further broken down into:
  - **Pre-Migration**: Tasks that can be completed before the migration of the database table data content
  - **Data-Migration**: Tasks that can be completed during the migration of database table data and table data dependent objects (objects and data that must be migrated as part of the table data transfer)
  - **Post-Migration**: Tasks that can be completed after the table data migration step

Oracle Database Migration Planning

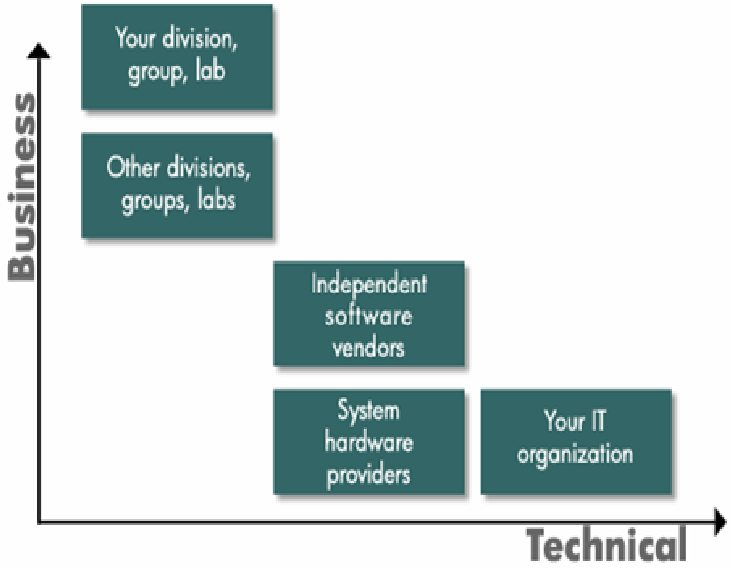
Business Considerations

Business Considerations

- Gathering Information
- Business Practice Transition
- Assessing Software Versions
- Choosing a Migration Time Frame
- Choosing an Oracle Upgrade vs. Migration Path

Gathering Information

- Periodic review of business requirements driving IT requirements
- Collect and analyze high level business and technical information that may have a bearing on the driving factors for transition and migration. Look for:
  - Changes in the nature, volatility and ownership of the data resource
  - Changes in access requirements and system demands
  - Changes in Service Level Agreements
  - Changes in the organizational structure
  - Changes in ISV relationships and software utilized
  - Other changes that may affect functional business groups, their relationship to IT and applications



Business Practice Transition

- Look for an intersection of business and technical needs that build a case for changes/enhancements to the IT architecture
- Review the information gathered and determine which potential changes in business practices may coincide with or provide an opportunity for database migration. Such an opportunity may result from a need to:
  - Change from a monolithic to a client/server or N-tier environment
  - Achieve more efficient workload balancing/management
  - Provide on-demand resource distribution through clustering or Grid computing
  - Increase degree of high availability and disaster recovery
  - Further distribute the data resource through replication
  - Reduce operational costs by consolidating application and database servers (large server partitioning)
  - Replace or retire existing database applications
  - Develop and implement new database applications

Assessing Software Versions

- Examine information gathered regarding current and future software required to run the business
- For all critical software look at the aspects of:
  - Availability and platform certification
  - Support and “maintainability”
  - Compatibility
  - Performance
  - Scalability
  - Interoperability
  - Licensing and method for determining cost

Choosing a Migration Timeframe

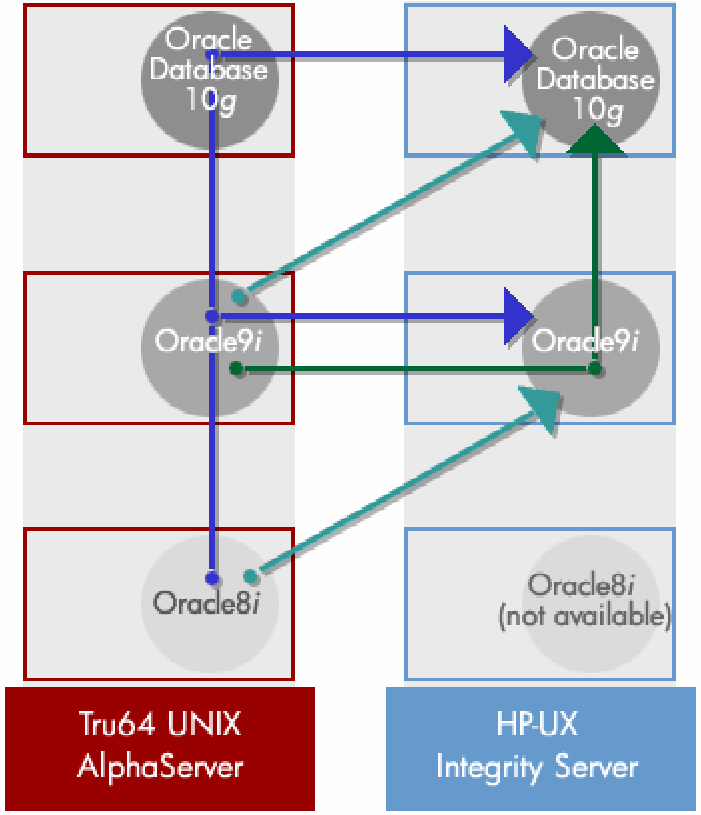
- Consider the following when determining a time frame for platform transition and database migration:
  - HP product availability and roadmaps -
    (ensure needed configuration and features are available to support the migration time frame desired)
  - ISV product availability and porting roadmaps –
    (ensure Independent Software Vendor supplied products are available in the versions and platform port required)
  - IT architecture and the possible phasing of transitions/migrations –
    (the structure of the IT environment may allow the transition of different Application and Database servers at different times)
  - Business intensity cycles and operational requirements -
    (seasonal lulls and other reduced processing periods)
  - Normal and extended planned downtime –
    (holidays, shutdowns, routine and special maintenance windows)
  - Service Level Agreements and required database availability
  - Estimates of database migration duration –
    (will the migration duration fit comfortably in a possible downtime window?)
  - Contingency timeframes – "The best laid plans of mice and men"…you know!

Choosing an Oracle Version Upgrade vs. a Migration Path – (1 of 2)

- "*Up and Over Approach*": Upgrade the Source server Oracle version to match the Target server Oracle version before commencing the migration effort (e.g. Oracle9i to Oracle9i)
  - Recommended by HP
  - Downside: more DBA time and resources to perform the upgrade in advance
  - Upside: less risk, less chance for errors, and reduced verification effort
- "*Diagonal Approach*": Source server DB uses older version of Oracle, and the Target server DB will use the newer version of Oracle – more or less an Export/Import driven upgrade migration across platforms and versions
  - Downside: more risk, with more intensive verification requirements
  - Upside: less time and less resource intensive
- "*Over and Up Approach*": Migrate, then upgrade on Target
  - Downside: not possible for AlphaServer Tru64 UNIX to Integrity HP-UX 11i migrations when Source database version is Oracle8i or and earlier - **not** supported on Integrity HP-UX 11i systems
  - Upside: will be possible for Oracle9i source to Oracle10g target scenario.

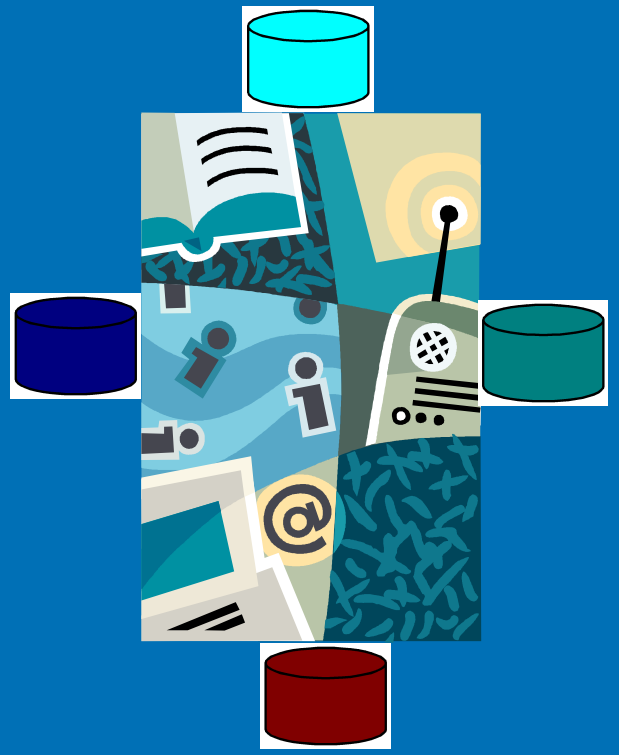
Choosing an Oracle Version Upgrade vs. a Migration Path – (2 of 2)

- How do I get there from here? A picture is worth a thousand words…
- Again, without Oracle8i version available on HP-UX 11i Integrity Server…cannot execute Oracle DBUA on the Target system



Oracle Database Migration Planning

Database Evaluation



Evaluating The Database

- Database Usage
- Planning Tools
- Reviewing Source Database Characteristics
- Database Workload & Resource Demands
- Database Physical Redesign
- Estimating Migration Duration
- Considering Database Access and Dependencies

Database Usage

- Types of Database Migration
  - Static:
    - The database is not being written to for the duration of the migration (downtime window); therefore, in-flight transactions are not a concern
    - Could be an OLTP database with an ample downtime window, or a DSS database
  - Dynamic:
    - The database must remain available for write transactions during the course of the migration
    - A minor downtime (<15 minutes) is permitted for network connection redirection
    - High-demand OLTP database
- Types of Database Migration Data
  - Static:
    - Unchanging data: table data that is not being updated during the migration window and perhaps could be migrated in advance (e.g. reference, code tables, history tables or partitions)
  - Dynamic:
    - Changing data: table data that could be modified during the migration window

Planning Tools

- HP **P**lanning **A**ssessment **D**ocuments (PAD) for Oracle
  - Database Migration for Oracle **PAD**s and other planning tools available for download now at:

    http://h30097.www3.hp.com/transition/modules.html
  - Much of the higher level PAD recommendations apply to any database migration; most of the technical PAD recommendations are for DB flavor = Oracle from AlphaServer Tru64 UNIX to Integrity HP-UX 11i
- HP DBMPA Tool (**D**ata**B**ase **M**igration **P**lanning **A**ssistant) is available for download from above URL:
  - Runs against all HP platforms (Java based)
  - Extracts database metadata
  - Can generate detailed reports about the subject database
  - Can generate DB object inventories for before and after "diff" validation
  - If a customer sends HP the extracted metadata (XML format) and system information, HP can create reports with estimated Export/Import migration times
  - Identifies the top 10 tables that take the longest to migrate
- Database Evaluation Matrix
  - Template MS Excel Spreadsheet to assist in collection of DB evaluation data
  - Can be used in tandem with or instead of DBMPA Tool

Reviewing Source Database Characteristics

- Oracle version and patches
- DB size (allocated vs. utilized)
- Relative types and numbers of objects
  - Number of Stored Procedures and Packages
  - Number and types of Triggers
- Types of tables; HEAP, CLUSTER, IOT
- Table size: row width and row count attributes
- Table column datatypes (any binary or user-defined datatypes?)
- Table partitioning (none, range, hash, sub-partitions)
- Number and type of User Indexes
- Constraints (primary keys for all tables?, other RFI etc.)
- Current DB tuning practices and init.ora parameters, such as db_block_size, db_file_multiblock_read_count, etc.

Database Workload & Resource Demands

- Source Database Workload Requirements
  - Changes/evolution in the workload (OLTP, DSS, OLAP, Mixed)
  - User population growth (allocated, connected, active)
  - Application expansion (functional) and extension (content)
  - Service level and availability changes
  - Changes here translate into changes to…
- Source Database Resource Demands Over Time
  - CPU utilization
  - Memory allocation
  - I/O capacity (space)
  - I/O bandwidth
  - Network bandwidth
  - Administration intensity (SysAdmin/DBA time and effort)
  - All of the above

Database Physical Redesign

- Can help improve future production performance in situations where changes in workload and resource demands have put a strain on the database server
- Does NOT include changes to the application or any logical database structures (e.g. splitting tables, etc.)
- Involves such application-transparent adjustments as:
  - Amount of database storage space allocated
  - A change in the database or tablespace block size
  - Changes to table partitioning status
  - Redistribution of table data across a different set of tablespaces
  - Redistribution of tablespaces across underlying datafiles
  - Relocation of an index to its own tablespace
  - Change residence of datafiles from raw to file system or vice versa
  - Re-mirroring and/or re-striping of database storage devices for improved I/O performance and availability
- Places some additional demand on Oracle DBA resources ☹
- Have a positive effect on DB migration performance ☺
- May require further customization of the DB migration procedure used

Estimating Migration Duration

- Identify the tasks that need to be performed while DB server access is limited (actual Data-Migration)
- No need to include tasks that may be performed in advance (Pre-Migration)
- Estimate or measure the task time required (see Prototyping) and include some “fudge factor” to accommodate the unexpected
- Include time for:
  - Source database verification
  - Source database backup
  - Export/Import table data (or other table data transfer tools)
  - Verification of data transfer success
  - Gather database optimizer statistics (if not pre-establishing)
  - Create user indexes, referential integrity constraints, stored procedures, triggers, etc.
  - Run through a final verification process (selective)
  - Synchronize source and target databases (if Dynamic migration)

Considering Database Access and Dependencies

1. Online user application client-server connections using SQL*NET
   a) Client application point query program
   b) Client application ad-hoc query report-writer queries
   c) Oracle Forms requests
2. Batch user application connections using SQL*NET
   a) Batch server-side complex, canned, scheduled report-writer requests
   b) Batch application routine update maintenance programs/scripts
   c) Batch application mass update programs triggered by data feeds from other servers
   d) Batch application data extraction programs providing data feeds to other servers
3. Database administration function connections
   a) Administrative GUI tools
   b) Batch server-side database monitoring/maintenance scripts (crontab scheduled)
   c) Backup and recovery utility jobs (scheduled and unscheduled)
4. Clustering and replication software connections
   a) Oracle Real Application Cluster (RAC) connections
   b) Oracle Advanced Replication Server or Oracle Streams connections
5. Other remote connections
   a) ODBC from PC users
   b) Stored procedures invoked on another database server (Oracle and non-Oracle)
   c) UNIX remote shell invocations (rsh/remsh)
   d) Remote procedure calls (RPC)
6. Oracle Client Libraries and versioning compatibility

Oracle Database Migration Planning

Risk Mitigation

Mitigating Risk

- All the Planning efforts discussed so far, plus:
- Benefits of Prototyping the Database Migration
- Prototyping Configuration Models
- Risk Reducing Measures

The Benefits of Prototyping the DB Migration - (1 of 2)

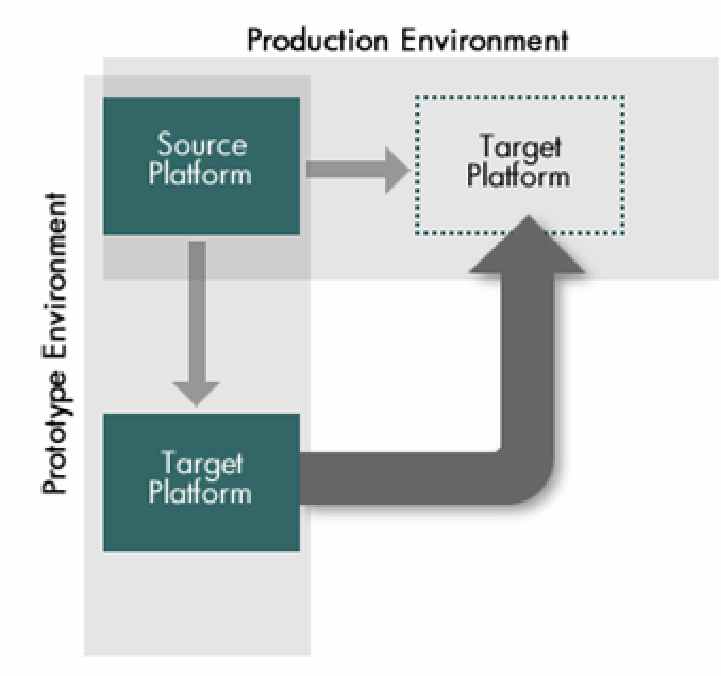
1. Trials the Target Server hardware and software environment
2. Aids in developing, testing and refining migration methods & procedures (tools, scripts, etc.)
3. Aids in testing the affects of cross-platform data transfer (binary datatypes, etc.)
4. Allows for advanced setup of the Target DB Server environment
5. Improves accuracy of final migration duration/downtime estimate
6. Helps determine affect of migration on Source system workload (intrusiveness during dynamic migration)
7. Assists in evaluating/testing DB physical redesign changes
8. Assists in evaluating/testing architecture changes (N-tier, etc.)

The Benefits of Prototyping the DB Migration - (2 of 2)

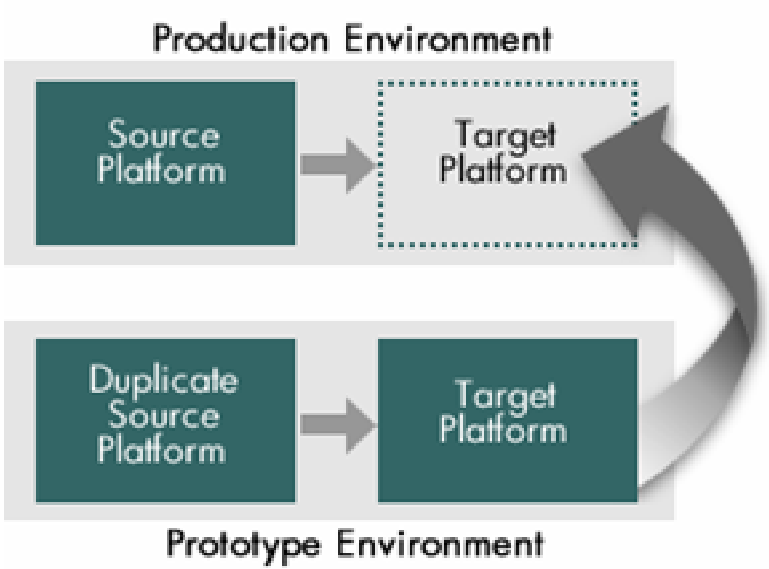
9. Provides for tuning experimentation for migration performance
10. Provides for tuning experimentation for production performance
11. Permits advance testing of DB application functionality
12. Permits advance testing of DBA infrastructure (scripts, tools, etc.)
13. Reveals any UNIX command, scripting or custom code porting issues
14. Proves performance, scalability and interoperability
15. Provides a new platform for training
16. Greatly reduces risk due to the unexpected!

Prototyping the DB Migration: Configuration Models

Single-source system prototype model:



Dual-source system prototype model:



Risk Reducing Measures - (1 of 2)

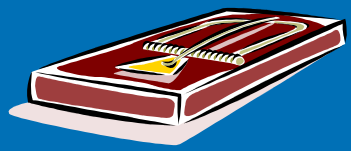
1. Have a detailed migration plan and procedures, including all tasks, task dependencies, and “task masters” (those assigned responsibility for each task)
2. Script/automate where it makes sense, ensuring that all scripts are “unbreakable” through testing
3. Ensure that all potential errors are trapped and displayed to monitored output logs
4. Freeze the application and database from changes well in advance of the migration…do not migrate with recent, un-prototyped modifications to source database and/or applications
5. If possible, perform a static DB migration as opposed to dynamic
6. Thoroughly prototype/test the migration procedure (more than once)
7. Thoroughly prototype/test ISV software and custom code applications in the new target environment

Risk Reducing Measures - (2 of 2)

8. Have source and target Oracle version be identical (Oracle9i to Oracle9i) if possible
9. Verify database integrity before beginning the migration
10. Backup the database before beginning the migration
11. Have a healthy DB post-migration verification/validation plan
12. Ensure that vendor support is available at the time of migration
13. Ensure that application development support is available
14. Implement a formal acceptance test plan with user signoff
15. Limit Production post-migration DB write transactions for a pre-determined period
16. Be able to fail-back to the Source production environment with ease

Oracle Database Migration Design

Tools, Tricks and Traps



Design Tools, Tricks and Traps

- The Database Migration Tool Set/Box
- DB Migration Tool Performance
- Tricks/Tips on Expediting the Migration process
- Potential Pitfalls

The Migration Tool Set – (1 of 2)

- Static Migration Tools
  - DDL Scripts (for create DB and some objects – maintained or generated)
  - Oracle Export/Import utilities
  - Oracle External Table Load Feature
  - HP DBFastTableCopy (TblCopy)
  - HP DBScriptSet Tool
  - HP File Mover Tool Suite (FMTS)
  - Third party tools (some GUI)
- Dynamic Migration Tools
  - Oracle Streams (9i to 9i) after Export/Import
  - Oracle Advanced Replication Server (8i to 9i)
  - Deferred Application Update Mechanisms (Custom)
    - Double Update – Stored Procedures
    - Transaction collection – batch apply
  - Third party tools ($$)

The Migration Tool Set – (2 of 2)

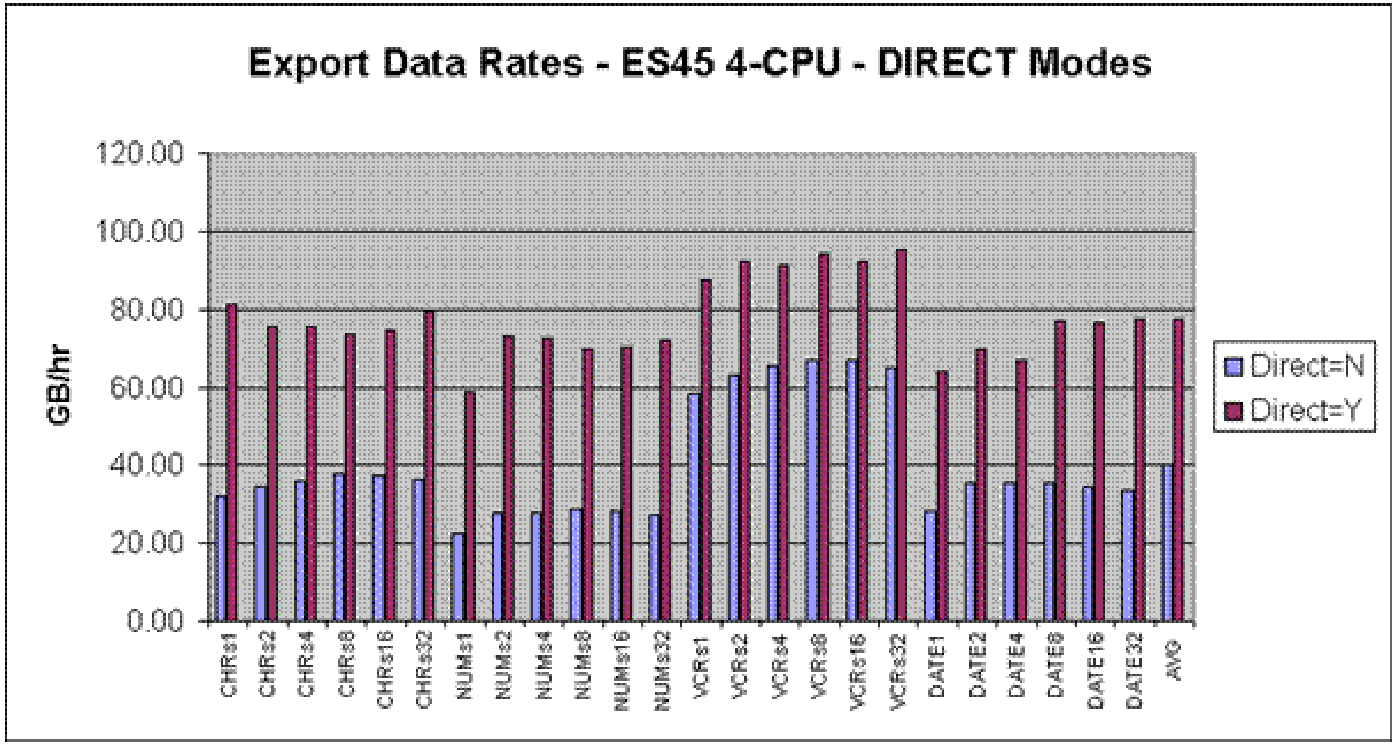
- Verification Tools
  - Error kick-outs/alerts from well-written migration script output logs
  - Export/Import output logs
  - Oracle DBVERIFY
  - Row Count Comparisons
  - Comparisons of “sampling query” results (much like benchmark ACID* tests)
  - HP DBCompare utility: Checksum and CRC against the table data Database object inventory “diff” reports (before vs. after)
  - Third party tools ($$)

Oracle Export/Import Utilities

- Fully supported by Oracle
- Well documented in the Oracle9i DB Utilities Guide
- Effective cross-platform, cross-endian
- May be used to Export from one Oracle version and Import to another (use respective version of the utility)
- May be used in several modes:
  - Full
  - Schema (user)
  - Table
  - Tablespace
- Use mode FULL=Y if the database is relatively small (simplifies things)
- May be used in conjunction with Oracle Streams
- May be executed in parallel to:
  - Migrate data from multiple tables or table partitions concurrently
  - Increase performance
  - Reduce migration time
- May be used with the Query option for selective table data migration
- HP Database Migration Engineering performance characterization testing

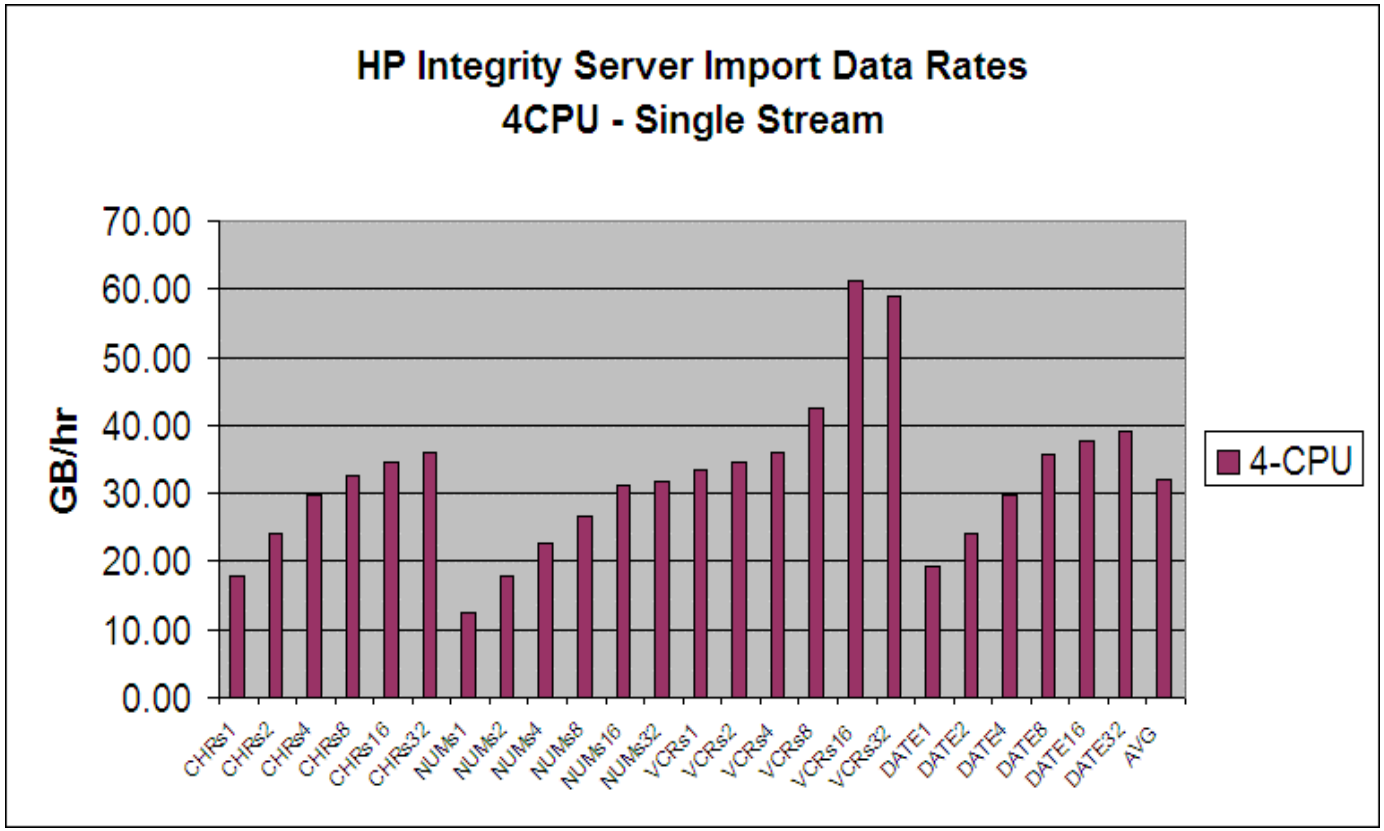
Export/Import Utilities: Tuning for Performance – (1 of 7)

- If tables contain common columnar datatypes (e.g. CHAR, VARCHAR2, NUMBER, DATE), use the DIRECT=Y option instead of DIRECT=N with the Export utility (50% to 100% faster)
- Refer to the Oracle9i Database Utilities Guide regarding the DIRECT option.
- Character datatype columns generally Export at a faster rate than numeric and date datatype columns (data conversion reasons)
- Exports conducted using an Alpha Server ES45 4-CPU (1.25 GHz) w/ (4) HBAs and (1) HSV110 (EVA) Storage Array housing 168 spindles



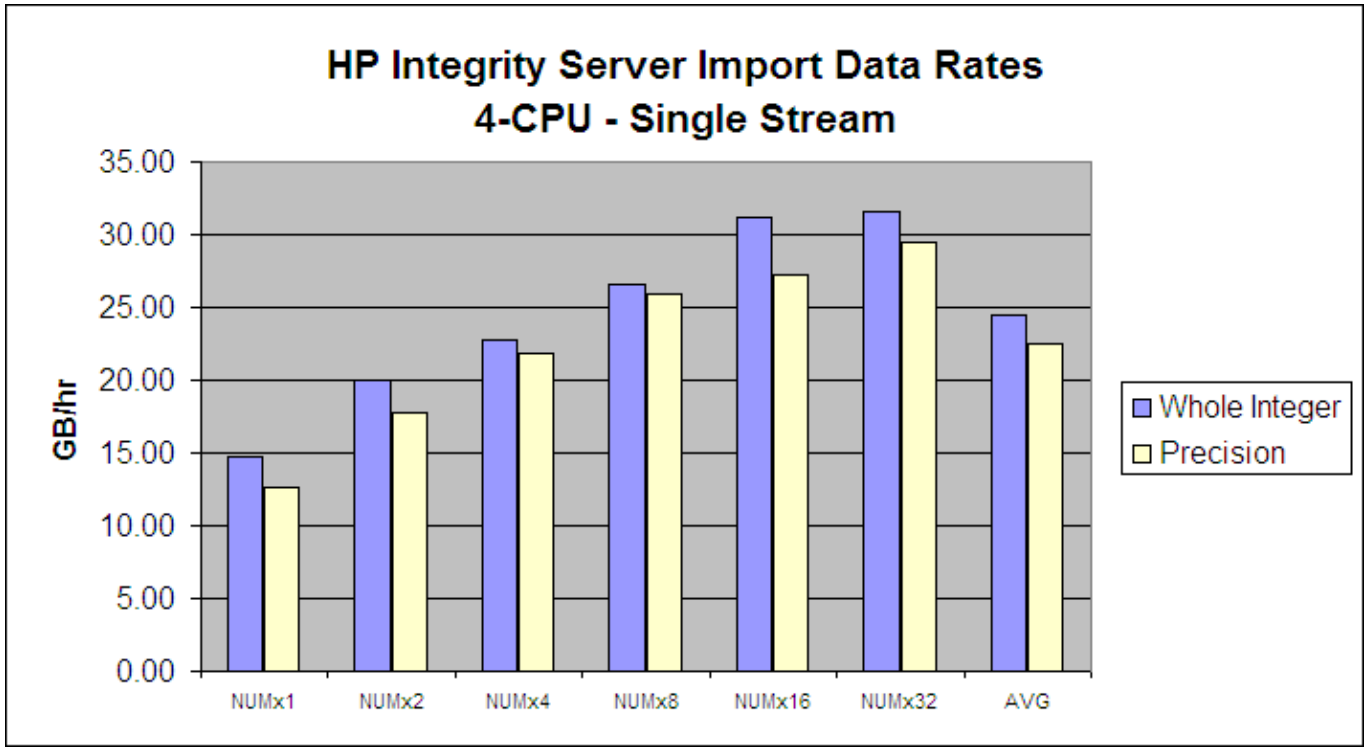
Export/Import Utilities: Tuning for Performance – (2 of 7)

- Tables with a greater avg_row_len generally Import at a faster rate (to a point)
- Import tests conducted on an HP Rx5670 4-CPU (1.0 GHz) Integrity Server with (4) HBAs and (1) HSV110 (EVA) Storage Array housing168 spindles



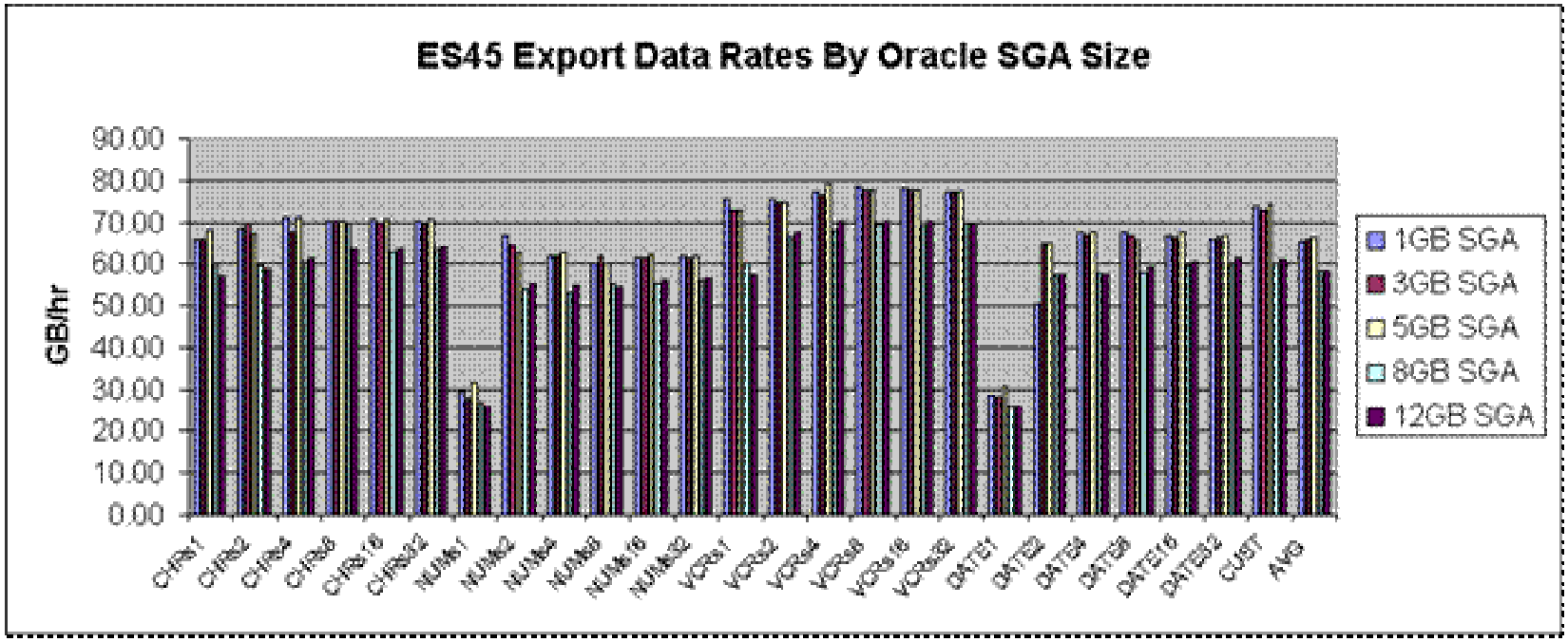
Export/Import Utilities: Tuning for Performance – (3 of 7)

- NUMBER data type columns containing whole integer values appear to Import (and Export) at a slightly faster data rate (about 9% on average) than those containing precision data values (numeric data values containing a decimal point).
- Trend becomes more apparent as the number of columns containing precision data in a table is increased



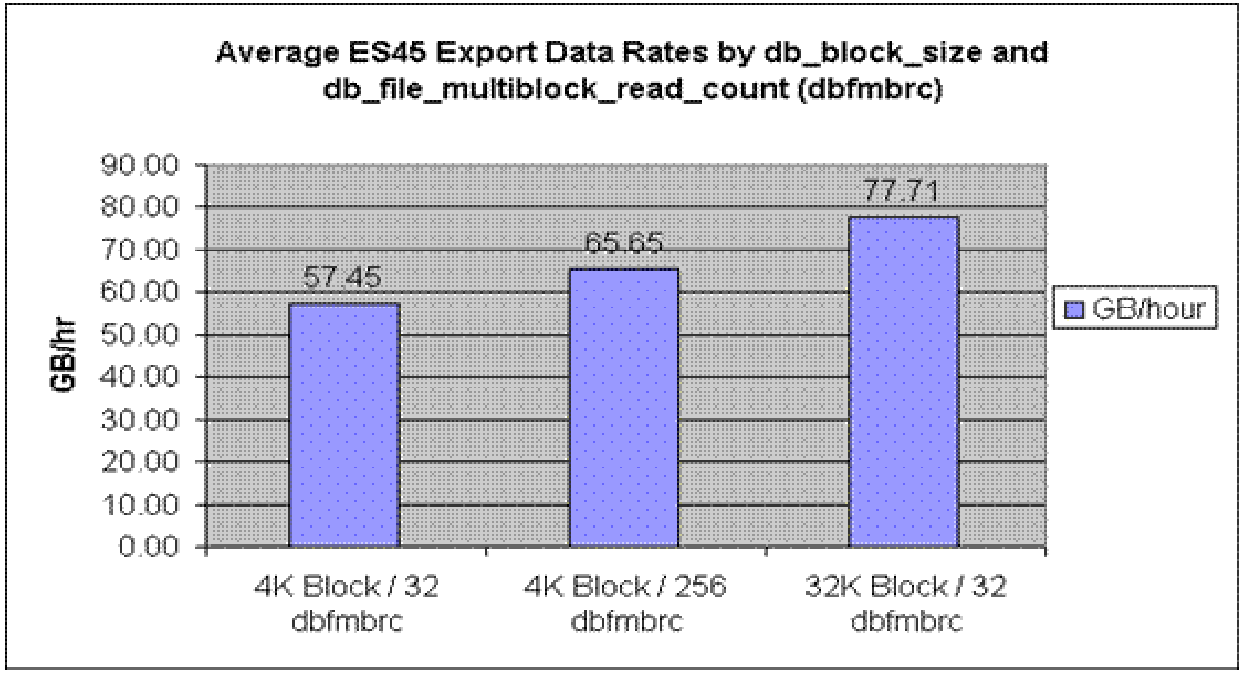
Export/Import Utilities: Tuning for Performance – (4 of 7)

- Export (and Import) testing with different Oracle SGA sizes
- Export (and Import) testing was conducted with Oracle instances having 1GB, 3GB, 5GB, 8GB, and 12GB SGA sizes
- With 8GB and 12GB, the db cache was large enough to retain the subject table
- Identical table data Exported at **slightly** different data rates depending on the Oracle SGA size (10% in some cases)



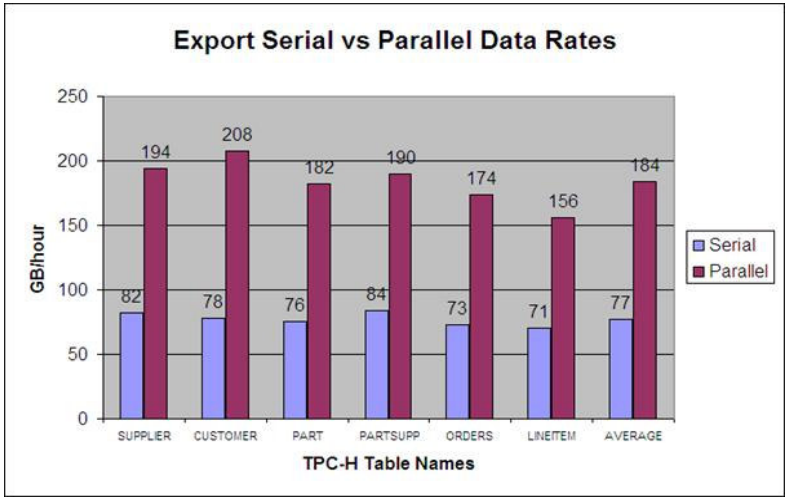
Export/Import Utilities: Tuning for Performance – (5 of 7)

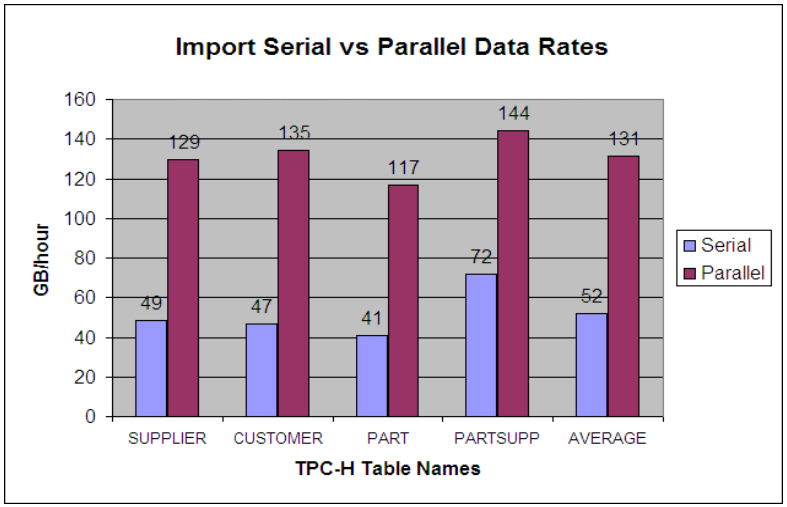
- Effects of db_block_size and db_file_multiblock_read_count:
  - In general, table data stored using a larger db_block_size (8K, 16K) will take less time to Export than table data stored using a smaller db_block_size (4K)
  - Reduce the adverse performance effects of a smaller block size by increasing the Oracle db_file_multiblock_read_count (dbfmbrc) parameter value (init.ora)
  - Increases the size of the Oracle I/O request (db_block_size X db_file_multiblock_read_count)
  - Maximum Oracle I/O request size for Oracle8i and Oracle9i = 1MB



Export/Import Utilities: Tuning for Performance – (6 of 7)

- Multiple concurrent parallel execution:
  - Divide Export/Import operations into multiple parallel execution streams
  - Export from or Import into a separate tablespaces, tables or table partitions concurrently
  - Enabling parallel concurrent executions of Export and Import will reduce overall table data transfer time
  - Maximum number of concurrent parallel executions limited by system resources available
  - Test ES45 and Rx5670 systems handled up to (4) Export/Imports per CPU
  - Note: Used table definitions from the TPC-H benchmark kit provided by the Transaction Processing Performance Council (TPC)



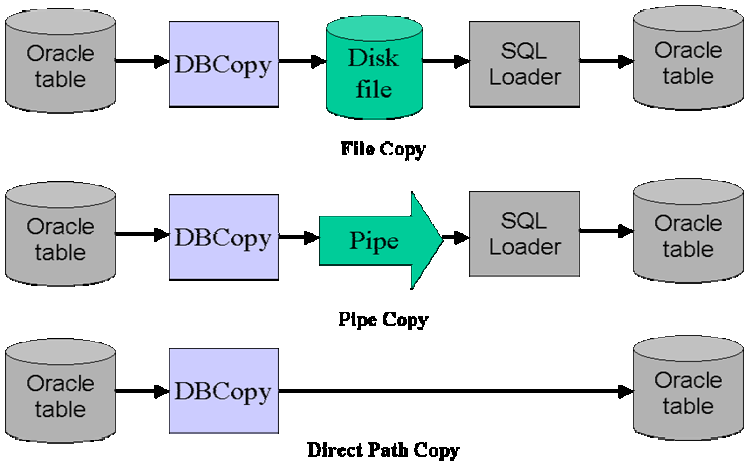


Export/Import Utilities: Tuning for Performance – (7 of 7)

- Miscellaneous Points & Hints:
  - Gather Optimizer statistics (DBMS_STATS.GATHER*) before Export
  - Improved performance if Oracle datafiles and Export dump file system reside on separate devices and/or I/O paths
  - If a table has primary and sub-partitions, Exporting by primary partition is faster than exporting by sub-partition (regardless of parallelism employed)
  - Table *hash* partitions Export faster than table *range* partitions
  - Index-organized tables (IOT) appear to export about 50% slower and import about 40% slower than traditional heap tables
  - Embedding indexes in the export and import (INDEXES=Y) operations took longer to execute in total than exporting and importing the table data (INDEXES=N), and then recreating the indexes manually using a DDL script
  - When using Export, ensure that the BUFFER= parameter (DIRECT=N) or RECORDLENGTH= parameter (DIRECT=Y) is set to a value appropriate for the subject table. Refer to the *Oracle 9i Database Utilities Guide* for instructions on determining the optimal value for this parameter
  - Export performance is better if executed on source system (locally) than from the target using SQL*NET
  - A fast network connection between Source and Target servers is needed!

HP DBFastTableCopy Tool – (1 of 2)

- Executable = “TblCopy”
- Developed in-house by HP Software Engineers
- Supports four modes of execution:
  - File Copy (extracts to file; load component = SQL*Loader or ETLF)
  - Pipe Copy (extracts to FIFO pipe; load component = SQL*Loader or ETLF)
  - Direct Path Copy (performs both extract and load)
  - Information Only
- Utilizes an in-memory buffering approach
- Direct Path mode streams data from source DB to target DB with no use of intermediate files
- Direct Path mode can be up to 3X faster than Export/Import

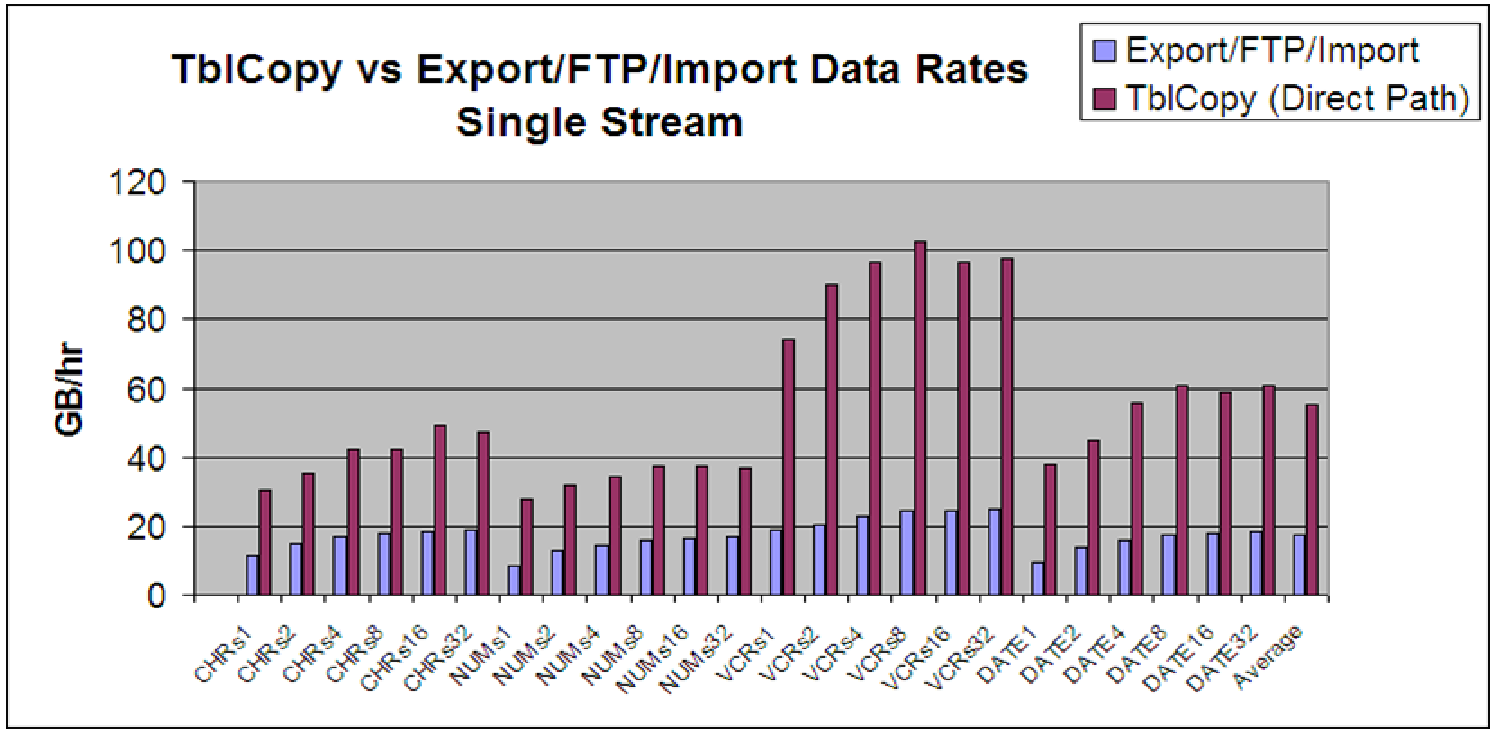


HP DBFastTableCopy Tool – (2 of 2)

- Direct Path mode can be up to 3X faster than Export/Import
- Supports multiple read and write threads for improved performance
- Supports parallelism by partition or by datafile ROWID range
- Copies the data content of one table per “TblCopy” execution
- Can tune relative performance varying:
  - Number of concurrent read threads (-r)
  - Number of concurrent write threads (-w)
  - Rows-per-buffer parameter (-C)
- Empty table object must be created in the target DB first
- No interaction/integration with Oracle Streams for a dynamic migration; primarily a static database migration tool
- May not support all user-defined or binary datatypes
- Available for release in calendar Q4 of 2004 with User Guide
- Free to customers migrating Oracle databases to HP Integrity Server

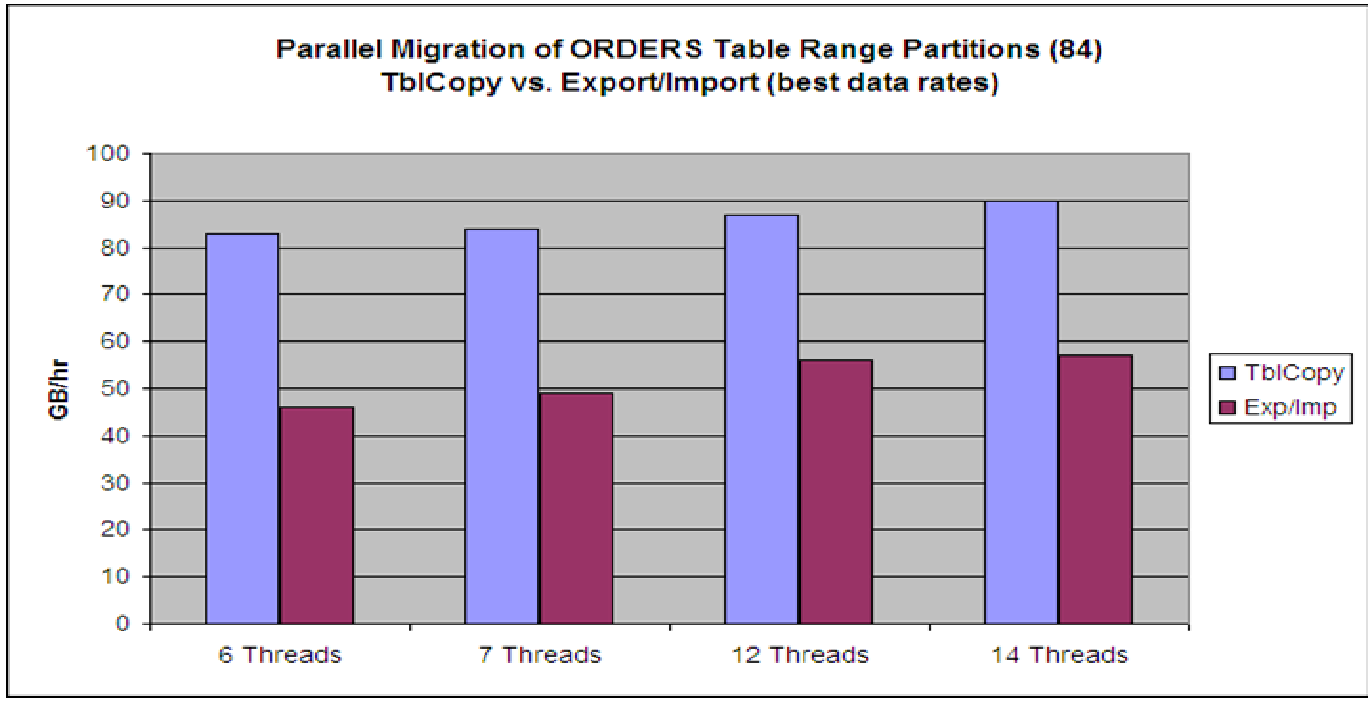
HP DBFastTableCopy Tool: Relative Performance – (1 of 3)

- TblCopy and complete exp/ftp/imp process performance compared
- Test used the datatype characterization tables (45 million rows each)
- Single-Stream test: no parallelism used



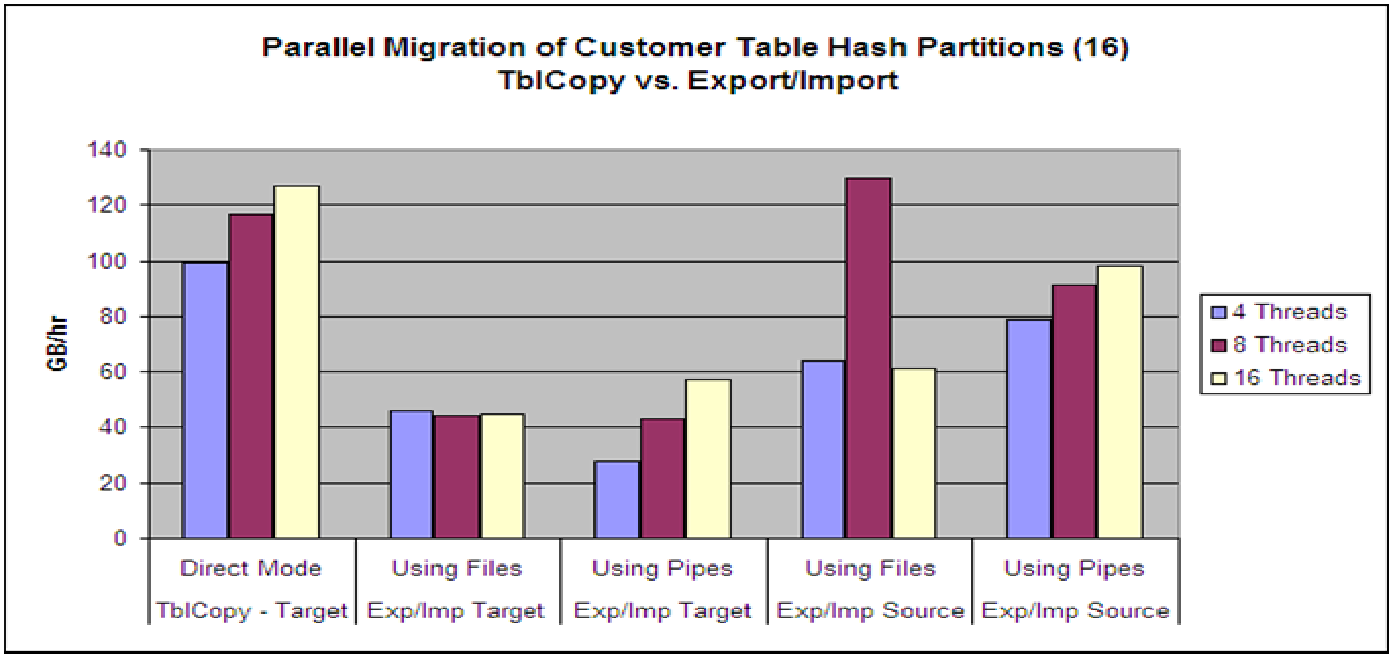
HP DBFastTableCopy Tool: Relative Performance – (2 of 3)

- TblCopy and complete exp/ftp/imp process performance compared
- Test used TPC-H ORDERS table:
  - 45 million rows
  - 84 date range primary partitions X 16 hash sub-partitions
- Process (exp/imp) and Thread (TblCopy) parallelism used



HP DBFastTableCopy Tool: Relative Performance – (3 of 3)

- TblCopy and complete exp/imp process performance compared
- Test used TPC-H CUSTOMER table:
  - 45 million rows
  - 16 hash partitions on c_custkey (NUMBER) column
- Process and Thread parallelism used



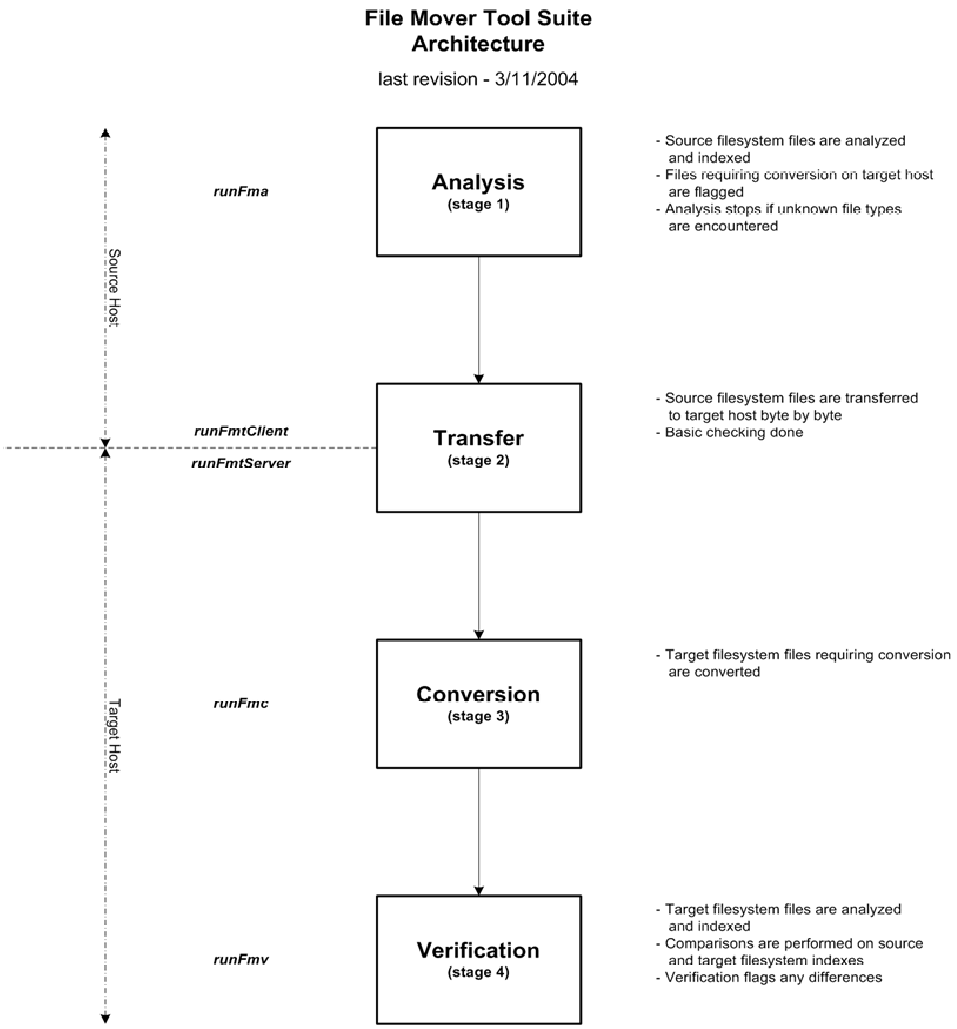
HP DBScriptSet Tool

- A collection of PERL and shell scripts to assist in the process of migrating Oracle databases from Alpha Tru64 UNIX ® to Integrity Server HP-UX 11i V2
- Intermediate generated scripts can be user-tailored
- Introduces parallelism where appropriate (exp/imp)
- Consists of three script-driven phases
- ***DBuilder*** – "Builds" the framework of scripts to facilitate the creation of a target database, based on the physical & logical structure of the source database
- ***DLoader –*** A mechanism to implement scheduling and bulk data transfer using Export and Import
- ***DCompleter*** – A utility to generate scripts which control the scheduling and execution of post data load operations
- Each phase is divided into "gen" (generate) and "run" (execute) components
- Available for release in calendar Q4 of 2004 with User Guide
- Free to customers migrating Oracle databases to HP Integrity Server

HP File Mover Tool - (1 of 2)

- Designed to assist customers with file system data transition
- Moves directory contents from source Tru64 UNIX host to target HP-UX hosts
- Not designed for moving Oracle database datafiles
- FMT is a suite of tools that operate in four sequential stages:
  - Stage 1 - Analysis
  - Stage 2 - Transfer
  - Stage 3 - Conversion (as necessary)
  - Stage 4 – Verification
- Comprehensive testing and refinement of the tool suite continues
- Test environments/scenarios will be expanded as products evolve (other platform pairs)

HP File Mover Tool - (2 of 2)



Dynamic Database Migration – Oracle Streams

- Use Export/Import utilities to migrate bulk of table data
  - Use CONSISTENT=Y or OBJECT_CONSISTENT=Y Export parameter
  - Use STREAMS_INSTANTIATION=Y Import parameter (sets SCN)
  - Use STREAMS_CONFIGURATION=Y for FULL database Import (Streams metadata)
  - Disallow any DDL changes to objects during an Export for Streams
- Source and target databases synchronized using Oracle Streams must be running Oracle 9.2.0.3 or later
- Should have PRIMARY KEY defined for all tables synchronized using Streams for performance reasons (avoid table scans)
- Application SQL may not contain BUFFER hints, which have the side effect of disabling supplemental logging required by Oracle Streams
- Database must be running in ARCHIVELOG mode
- Requires init.ora TRANSACTION_AUDITING = TRUE
- Requires init.ora SHARED_POOL_SIZE >=100MB
- No capture for datatypes: NCLOB, LONG, LONG RAW, BFILE, ROWID, UROWID
- No capture for user-defined types (including object types, REFS, varrays, nested tables)
- Refer to the HP TM PAD for Oracle Streams
- Refer to *Oracle9i Streams Release 2* documentation

HP DBCompare Utility - Verification

- Standalone utility – compares data content of a table between source and target databases (one table pair per execution)
- Uses multiple processing threads and buffers per execution for enhanced performance
- Command line execution and parameters
- Designed to execute from UNIX shell script
- Errors flagged to *stdout*
- Uses a combination of methods:
  - 32-bit Checksum (both additive and multiplicative)
  - 64-bit Cyclic Redundancy Check (CRC)
  - Each column is check summed
  - A CRC is calculated for each row
  - The set of row CRCs is check summed
- The Checksum method reliability is on the order of 1-1/232 or 0.999999997672
- The CRC method reliability is on the order of 1- 1/264- or 0.99999999999999999458
- For limitations, refer to the HP TM PAD for the DBCompare utility (e.g. 1024 columns)
- Available for release in calendar Q4 of 2004 with User Guide
- Free to customers migrating Oracle databases to HP Integrity Server

Expediting a Database Migration

- Thorough advanced planning
- Perform the migration in phases if possible
- Correct choice of tools/utilities
- Detailed, fully tested procedures and scripts
- Improve step/task efficiency through iterative prototyping efforts
- A target system and Oracle instance tuned for migration
- Pre-migration – perform as many tasks in advance as possible
- Pre-population of objects and optimizer statistics
- Parallelism in migrating table data where it makes design sense
- Concise validation/verification steps

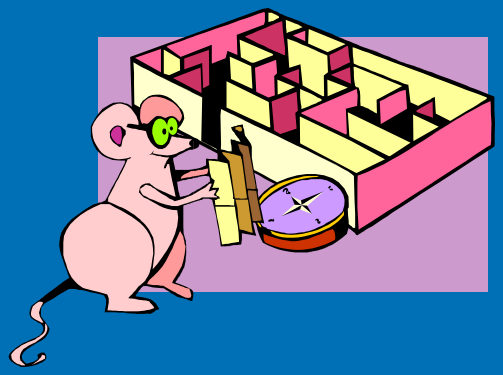
Potential Pitfalls / Traps

- Poor / incomplete planning process
- Considering only technical factors (ignoring business factors)
- Attempting to migrate prematurely
- Underestimating the total effort / resources needed
  - Time
  - People
  - Hardware
  - Software
  - Budget
- Forcing a migration to fit an unrealistic time frame / schedule
- Not developing a detailed step-by-step migration procedure with assigned responsibilities
- Incomplete prototyping / testing of migration procedures
- Migrating new application and/or database changes
- Migrating Binary datatype column data not prototyped

Considering Binary Data Types

- Binary datatypes (LONG, LONG RAW, CLOB, BLOB, etc.) across Endian
- Potential Pitfall
- FLOAT embedded in LONG RAW, etc.
- Test migration results of these in advance (prototyping)
- If problems/issues, can convert datatypes before migrating:
  - LONG to CLOB
  - LONG RAW to BLOB
  - Caution: May affect the application, so test all related functionality
- Oracle Streams does not support capture of NCLOB, LONG, LONG RAW binary datatypes

Oracle Database Migration Testing & Implementation



Database Migration Procedure: Pre-Migration Steps – (1 of 2)

1. Establish Migration as a True Project
2. Begin Migration Project Planning Process
3. Determine first-pass schedule
4. Complete Business Requirements Definition
5. Complete Database Information Gathering and Analysis Tasks (source systems)
   - Database and server dependencies
   - All database and server interfaces
   - Oracle version and patch data
   - DB and object information (user-defined datatypes, stored procedures, triggers, RI constraints, etc.)
   - Retain copies of the init.ora file(s)
   - Generate/collect database and object create DDL scripts as backup
   - Database sizing requirements data (current and projected growth)
6. Complete Target Platform Solution Architecture
7. Order Target Platform Hardware/Software (including Oracle distribution, third-party software required, etc.)
8. Complete Database Migration Design (scripts, tools, tasks, team)
9. Based on the above, determine the appropriate dates to migrate (usually during your favorite holiday)
10. Install/Configure Target Server Hardware

Database Migration Procedure: Pre-Migration Steps – (2 of 2)

11. Configure high-speed network connectivity (Gigabit Ethernet)
12. Design and configure Physical server storage arrays (I/O sub-system devices)
13. Install HP-UX 11i V2 and associated products on Target including necessary patches (extremely important) for Oracle
14. Configure HP-UX kernel in support of Oracle (Refer to Oracle Installation Guide for HP-UX for all details)
15. Configure LVM volume groups, logical volumes and VxFS file systems needed
16. Create planned directory structures including $ORACLE_HOME
17. Create HP-UX users:groups including “oracle:dba”
18. Install/Configure Oracle9i + on Target system from software distribution
19. Configure Oracle as needed (init.ora, listener.ora, tnsnames.ora files)
20. Install ancillary (third-party) software as needed (DBA tools and/or applications)
21. Address custom code considerations for any server-side app programs
22. Establish DBA support infrastructure (cronjobs, utility scripts, monitoring tools, backup/recovery mechanism, etc.)
23. Create Oracle database instance on Target system
24. Load Oracle system objects (e.g. catalog.sql, catproc.sql, etc.) on Target
25. Define user tablespaces on Target

Database Migration Procedure: Data-Migration Steps

26. Update source DB/table optimizer statistics (DBMS_STATS.GATHER*)
27. DBVerify the Source database
28. Backup the Source database
29. Initiate DB object and data transfer procedure using tools selected (DDL scripts, Export/Import, TblCopy, etc.)
30. Update target DB/table optimizer statistics (DBMS_STATS.GATHER*)
31. Create appropriate Indexes
32. Create referential integrity constraints
33. Create Stored Procedures, Triggers, etc.
34. Verify/validate database migration
35. Attempt all this as a **prototype** effort first!

Database Migration Procedure: Post-Migration Steps

34. Verify database integrity
35. Backup database on new platform before use
36. Application functionality testing with finite/identified updates
37. Re-point network identification (DNS, etc.) for Production connectivity switchover
38. Re-test network/client connectivity
39. Re-establish replication strategy (if applicable)
40. Resume Production operations schedule
41. Adjust tuning for performance if necessary
42. Monitor for the required amount of time…then celebrate!

Oracle Database Migration Supplemental Materials and Sample Scripts

Export / Import Utility Design Decisions

Design Point (1):

"***Design Preparedness***"

Begin addressing Design Points for using the Oracle Export and Import utilities to migrate database metadata and user table data from the Source database to the Target database.

Design Point Assumptions:

- ✓ You have read the Export / Import Utility Planning Assessment Document (PAD).
- ✓ You have formulated a migration plan which includes a knowledge of which databases will be migrated and when (phase of your plan).
- ✓ You have accumulated all the information about the database(s), either using the DBMPA Tool, the Database Evaluation Matrix Spreadsheet, both, or an alternate method of your choosing.
- ✓ You have available, and at the ready, the Oracle documentation for the version(s) of Export and Import you will be using.
- ✓ You have an understanding of what the available downtime window for the database(s) to be migrated will be.
- ✓ You have full knowledge of any prerequisite steps that should be accomplished (i.e. target database create, catexp.sql or catalog.sql)

Design Point Options:

(Choose **ALL** options listed)

a) Read the Oracle9i Database Utilities Guide, Export and Import chapters.

b) Read the Oracle Export and Import documentation for the Source DB version, if not Oracle9i.

c) Review all the information collected about the databases (and tables) to be migrated.

d) Note and record Export and Import scenarios that may apply to your data.

e) Prepare / review the general Export/Import plan that will now be extended to incorporate detailed Migration design procedures.

f) Ensure that all prerequisite steps to Export/Import have been considered.

Export / Import Utility Design Decisions

Design Point (2):

"***Determining Export/Import Mode***"

The Export and Import utilities may be executed in one of four different modes:

- Full
- User (Owner)- sometimes called Schema
- Table
- Tablespace

Design Point Assumptions:

- ✓ You have read the Export Modes and Import Modes sections of the Oracle Database Utilities Guide chapters on Export and Import. Specific attention has been given to the object types allowed for each mode.
- ✓ Tablespace mode may not use "Direct Path" mode (see Design Point 4).
- ✓ Users having the EXP_FULL_DATABASE role may Export in any of the four modes. Users must have this role to use FULL mode.
- ✓ Users having the IMP_FULL_DATABASE role may Import in any of the four modes. Users must have this role to use FULL mode.
- ✓ In general, FULL mode is used for smaller databases (less than 10GB in size), as it offers less opportunity for parallelism, but a less complex approach.

Design Point Options:

(Choose an option, depending on the migration requirements):

a) **FULL** – Entire database; all objects and data (if ROWS=Y).

b) **USER** – Most objects and data owned by the user (synonymous with "schema").

c) **TABLE** – The table object, some related objects and data (if ROWS=Y). Associated triggers are included (if TRIGGERS=Y), but procedural objects are not. Table partitions may be specified.

d) **TABLESPACE** – Tablespace definitions and most objects and data contained within the tablespace. The only mode to allow tablespace definitions.

Export / Import Utility Design Decisions

Design Point (3):

"***Gathering Source DB Statistics***"

The gathering of accurate database/table optimizer statistics must be performed against the source database before Export is used.

Design Point Assumptions:

- ✓ The **Analyze** command is typically used for databases using a version of Oracle prior to Oracle8i.
- ✓ Oracle recommends using the **DBMS_STATS** command package for Oracle8i and Oracle9i databases for more accurate statistics gathering.

Design Point Options:

(Choose one option, depending on database version).

a) Use the Analyze command to gather table statistics (Choose this option for pre-Oracle8i databases).

b) Use the DBMS_STATS.GATHER* command set to gather accurate table, schema or database statistics (Choose this option for Oracle8i or Oracle9i).

Export / Import Utility Design Decisions

Design Point (4):

"***Determining Export Path Mode***"

The Export Utility is able to execute in "conventional path" mode or "direct path" mode. The choice of path mode may have a significant impact on Export performance. The execution mode does not affect the subsequent Import operation.

Design Point Assumptions:

- ✓ Tables containing objects and columns defined as certain binary datatypes (LOBs) must use "conventional path" mode (DIRECT=N) prior to Oracle version 8.1.5.
- ✓ Oracle version 8.1.5 and higher allows the use of "direct path" mode (DIRECT=Y) universally. Embedded objects and LOB columns will automatically default to "conventional path" (noted in the Export .dmp file), while the more commonly defined datatype columns will use "direct path".
- ✓ "Direct path" mode Export executions may offer a 50-100% increase in Export speed/performance over "conventional path" mode.

Design Point Options:

(Choose the appropriate option for each database/table):

a) DIRECT=N

Use this Export parameter to execute in "conventional path" mode. It will support all Oracle datatypes.

Use the **"BUFFER="** Export parameter to specify buffer size.

b) DIRECT=Y

Use this Export parameter to execute in "direct path" mode. It will support common datatypes and allow Export to run faster. Use the **"RECORDLENGTH="** Export parameter to specify buffer size in this mode. The value should be a multiple of system I/O request size or db_block_size.

This path mode does not support Exporting or Importing in Tablespace mode.

Export / Import Utility Design Decisions

Design Point (5):

"***Advanced Table Object Creation***"

When dealing with a large and/or complex database migration, it is often advantageous to create empty table objects and pre-load them in advance of the final Export/Import of table data.

Design Point Assumptions:

- ✓ Empty table objects may be created in advance on the target database. They may be created in tablespaces named as they originally were on the source database, or in differently tablespaces.
- ✓ Table objects defined in advance of a final data migration may be test loaded and then truncated any number of times in order to help trial or prototype the migration design.
- ✓ The use of the **IGNORE=Y** Import parameter allows the Import utility to bypass the table object create step and just load the data. In this case, if the Import encounters an existing table object, it ignores the fact that the table already exists, and does not generate an error.

Design Point Options:

(Consider this option for selected database tables):

- ✓ If underlying tablespace and/or datafile organization is to be **different** in the target server database for user tables than it was in the source database server, use new SQL scripts to create the new empty tables. Otherwise, use Export/Import to create the target database tables.
- ✓ Use Export/Import to load the target database tables with Production data in trial / prototyping runs.
- ✓ Use DBMS_STATS.GATHER_TABLE_STATS to gather optimizer statistics before the final data migration.
- ✓ Compile/create other objects that reference these tables in advance. Be careful of TRIGGERS and CONSTRAINTS…these should be disabled before the final table data migration if they are created in advance.
- ✓ **Truncate** the target database tables before final Export/Import of the production source table data. Associated objects and statistics remain intact.
- ✓ Re-Export/Import the production table data using the **IGNORE=Y** Import parameter.

Export / Import Utility Design Decisions

Design Point (6):

"***Pre-loading Static Table Data***"

The concept of "**static**" or unchanging table data was discussed in the Database Migration Planning Assessment Documents. It is advantageous to pre-load table data that is not subject to change in advance of the final data migration. This reduces the duration required for the final data migration window.

Design Point Assumptions:

- ✓ Static table data may include relatively constant data that resides in code or reference tables.
- ✓ Static table data my include all the data within a table or part of the data within a table.
- ✓ Static table data may be isolated as data residing in specific table partitions (e.g. date range partitions), or as data identified as having a specific set of column values.

Design Point Options:

(Consider one the following options for each database table subject to migration):

a) The table contains code or reference data that will not be subject to change, or is subject to change only outside the window of the migration project. Consider migrating such tables in advance of the final data migration.

b) The table is updated infrequently, the update volume is low, and the updates can be easily identified and/or deferred to applied later. Consider migrating such tables in advance of the final data migration, while collecting further updates to be applied during the final data migration effort.

c) The table is partitioned, such that updates are made to one or a few identifiable partitions. Consider migrating the table partitions not subject to changes in advance of the final data migration. Migrate the active or "dynamic" partitions during the final data migration effort.

d) The table is not partitioned in such a way as to isolate the changing data, but data rows subject to updates can be identified using selected criteria. Consider using the Export "**QUERY=**" parameter to separate the migration of static data and dynamic data with SQL select criteria.

e) The table is subject to random updates that are not easily identifiable or predictable. Data must be migrated during the final data migration effort. After final Export/Import operations, the use of Oracle Streams may be required to synchronize updates between the source and target databases.

Export / Import Utility Design Decisions

Design Point (7):

"***Determining Export Parallelism***"

The Export Utility is able to execute against individual tables and/or table partitions. Multiple Export operations may run concurrently against different tables and/or table partitions.

In general, HP has been determined that given enough memory, two Exports may be successfully executed per available Alpha Server host CPU.

Design Point Assumptions:

- ✓ Export parallelism may not be used when the FULL=Y (full database Export) is set.
- ✓ Exporting table partitions requires the table name and the partition name be specified in the "TABLES=" Export parameter

  (e.g. "TABLES=CUST:SYSP4145").
- ✓ It is more efficient to Export partitioned tables using primary partition names instead of sub-partition names.

Design Point Options:

(Consider the Option examples):

Your Database Evaluation spreadsheet data yields the following:

Source DB Host: AS/DS20 2-CPU

Maximum recommended concurrent Exports: (4)

a) Concurrent Exports of multiple non-partitioned tables:

Concurrent Export (of 4)	**Export Time Line Duration**
1	CUSTOMER.....PART.....
2	PARTSUPP.....SUPPLIER.....
3	NEWORD.....ORDHIST.....
4	ORDRLINE.....

b) Concurrent Exports of Table Partitions (CUST table partitioned 8 ways):

Concurrent Export (of 4)	**Export Time Line Duration**
1	CUST:SYS_P4145.....CUST:SYS_P4149.....
2	CUST:SYS_P4146.....CUST:SYS_P4150.....
3	CUST:SYS_P4147.....CUST:SYS_P4151.....
4	CUST:SYS_P4148.....CUST:SYS_P4152.....

Export / Import Utility Design Decisions

Design Point (8):

"***Determining Import Parallelism***"

The Import Utility is able to execute using as input, individual table and/or table partition Export dump files. Multiple Import operations may run concurrently to load different tables and/or table partitions.

In general, HP has been determined that given enough memory, two Imports may be successfully executed per available Itanium2 Integrity Server host CPU.

Design Point Assumptions:

- ✓ Import parallelism will not be used when the FULL=Y (full database Import) is set.
- ✓ Importing table partitions requires the table name and the partition name be specified in the "TABLES=" Import parameter

  (e.g. "TABLES=CUST:SYSP4145").
- ✓ It is more efficient to Import partitioned tables using primary partition names instead of sub-partition names.
- ✓ Parallel Imports should be designed not to overload areas of the I/O subsystem, creating "hot spots".

Design Point Options:

(Consider the Option examples):

Your Database Evaluation spreadsheet data yields the following:

Target DB Host: Rx2600 2-CPU Integrity Server

Maximum recommended concurrent Imports: (4)

a) Concurrent Imports of multiple non-partitioned tables:

Concurrent Import (of 4)	**Import Time Line Duration**
1	CUSTOMER.....PART.....
2	PARTSUPP.....SUPPLIER.....
3	NEWORD.....ORDHIST.....
4	ORDRLINE.....

b) Concurrent Imports of Table Partitions:

(CUST table partitioned 8 ways)

Concurrent Import (of 4)	**Import Time Line Duration**
1	CUST:SYS_P4145.....CUST:SYS_P4149.....
2	CUST:SYS_P4146.....CUST:SYS_P4150.....
3	CUST:SYS_P4147.....CUST:SYS_P4151.....
4	CUST:SYS_P4148.....CUST:SYS_P4152.....

Export / Import Utility Design Decisions

Design Point (9):

"***Parallel Table Export/Import Execution Methods (9i to 9i)***"

As shown in previous examples, multiple Export and Import operations may be executed concurrently to load different tables and/or table partitions.

To extend this concept, a script may be run from either the Source DB server or the Target DB server that permits multiple Exports to extract from the Source database while multiple Imports load the Target database.

Design Point Assumptions:

- ✓ The same version of Export/Import is used (source and target DB = Oracle9i).
- ✓ If the Export/Import script is executed on the Source server, the Export will run locally against the Source Oracle instance, and the Import will run remotely (SQL*NET) against the Target Oracle instance.
- ✓ If the Export/Import script is executed on the Target Server, the Export will run remotely (SQL*NET) against the Source Oracle instance, and the Import will run locally against the Target Oracle instance.
- ✓ HP migration engineering has found that using .dmp **files** instead of .dmp **pipes** provides better overall Export/Import performance, but uses more disk space.
- ✓ Export .dmp files should be located on **separate** file system(s) / device(s) than the Oracle database datafiles.

Design Point Options:

(Consider the Option example):

Sample K-shell script performing concurrent Export/Import of a Table partitioned 16 ways executing (4) iterations of (4) table partitions (4 X 4) from the **Source** Oracle database server (uses intermediate .dmp **files** on the Source server):

```
#!/bin/ksh
# Perform Oracle9i Export of  database table
# Use for Testing Partitioned exports...4X4 (4 iterations X 4 partitions)
TABLENAME=$1
echo "Oracle export/import started at " `date`
echo "Oracle export/import for TABLE = " $TABLENAME
i=0
PMAX=4
while [ $i -le 3 ]
do
      CNT=1
      while [ $CNT -le $PMAX ]
      do
      PNUM=$((CNT + $i * $PMAX))
      exp tpcd/tpcd parfile=$MIGR_DIR/parms/exp_${TABLENAME}_p${PNUM}.par &
      CNT=`expr $CNT + 1`
      done
      wait
      CNT=1
      while [ $CNT -le $PMAX ]
      do
      PNUM=$((CNT + $i * $PMAX))
      imp tpcd/tpcd@TPCH-tgt  parfile=$MIGR_DIR/parms/imp_${TABLENAME}_p${PNUM}.par
            &
      CNT=`expr $CNT + 1`
      done
wait
i=`expr $i + 1`
done
echo "Oracle export/import completed at " `date`
```

Export / Import Utility Design Decisions

- **Design Point (9) (continued):**

  "***Parallel Table Export/Import Execution Methods (9i to 9i)***"

Sample **Export** Parameter File:

```
BUFFER=65535
COMPRESS=N
CONSISTENT=N
DIRECT=Y
FEEDBACK=0
FILE=/exports/exp_customer_p1.dmp
FULL=N
GRANTS=N
HELP=N
INDEXES=N
LOG=/Oracle9i/app/product/9202/tpch/SF_300GB/migration/logs/exp_custome
        r_p1.log
POINT_IN_TIME_RECOVER=N
RECORDLENGTH=65535
RECORD=N
ROWS=Y
STATISTICS=NONE
TABLES=(customer:SYS_P4145)
```

- **Design Point Options:**

  (Consider the Option example):

  Sample K-shell script performing concurrent Export/Import of a Table partitioned 16 ways executing (4) iterations of (4) table partitions

  (4 X 4) from the **Target** Oracle database server (uses intermediate .dmp **files** on the Target server):

```
#!/bin/ksh
# Perform Oracle9i Export of  database table
# Use for Testing Partitioned exports...4X4 (4 iterations X 4 partitions)
TABLENAME=$1
echo "Oracle export/import started at " `date`
echo "Oracle export/import for TABLE = " $TABLENAME
i=0
PMAX=4
while [ $i -le 3 ]
do
        CNT=1
        while [ $CNT -le $PMAX ]
        do
        PNUM=$((CNT + $i * $PMAX))
        exp tpcd/tpcd@TPCH-src  parfile=$MIGR_DIR/parms/exp_${TABLENAME}_p${PNUM}.par
             &
        CNT=`expr $CNT + 1`
        done
        wait
        CNT=1
        while [ $CNT -le $PMAX ]
        do
        PNUM=$((CNT + $i * $PMAX))
        imp tpcd/tpcd parfile=$MIGR_DIR/parms/imp_${TABLENAME}_p${PNUM}.par &
        CNT=`expr $CNT + 1`
        done
wait
i=`expr $i + 1`
done
echo "Oracle export/import completed at " `date`
```

Export / Import Utility Design Decisions

Design Point (9) (continued):

"***Parallel Table Export/Import Execution Methods (9i to 9i)***"

Sample **Import** Parameter File:

```
ANALYZE=N
BUFFER=65535
COMMIT=N
DESTROY=N
FEEDBACK=0
FILE=/exports/exp_customer_p1.dmp
FULL=N
GRANTS=N
HELP=N
IGNORE=Y
INDEXES=N
LOG=/oracle/ora920/tpch/SF_300GB/migration/logs/imp_customer_p1.log
POINT_IN_TIME_RECOVER=N
RECORDLENGTH=65535
ROWS=Y
TABLES=(customer:SYS_P4145)
```

Design Point Options:

(Consider the Option example):

Sample shell script performing concurrent Export/Import of a Table partitioned 16 ways executing (4) iterations of (4) table partitions (4 X 4) from the **Target** Oracle database server (uses intermediate .dmp **pipes** on the Target server):

```
#!/bin/ksh
# Perform Oracle9i Export of tpch database
# Use for Testing Partitioned exports...with Pipes
# Executed From/On the Target Database Server
# ITER Iterations X PMAX Partitions = Total Table Primary Partitions
TABLENAME=$1
ITER=$2
PMAX=$3
echo "Oracle tpch export/import started at " `date`
echo "Oracle tpch export/import for TABLE = " $TABLENAME
i=1
PNUM=1
while [ $i -le $ITER ]
do
CNT=1
      while [ $CNT -le $PMAX ]
      do
      exp tpcd/tpcd@TPCH-src parfile=$MIGR_DIR/parms/exp_${TABLENAME}_p${PNUM}.par &
      sleep 2
      imp tpcd/tpcd parfile=$MIGR_DIR/parms/imp_${TABLENAME}_p${PNUM}.par &
      CNT=`expr $CNT + 1`
      PNUM=`expr $PNUM + 1`
      done
wait
i=`expr $i + 1`
done
echo "Oracle tpch export/import completed at " `date`
```

Note: Refer to the UNIX "**mknod**" command for information on creating special PIPE files.

Export / Import Utility Design Decisions

Design Point (10)

"***Using Export/Import for Index Creation***"

After user table data has been successfully loaded, and ideally, optimizer table statistics gathered, any appropriate user indexes need to be created. Several methods for creating user indexes on the target database tables are optioned here.

Design Point Assumptions:

- ✓ The data for the table to be indexed has been loaded in its entirety without error.
- ✓ Optimizer statistics have been successfully gathered / established for the table to be indexed.
- ✓ The Export parameter, **INDEXES=Y**, enables the collection of user index information during the Export operation for the tables specified explicitly or implicitly by the Export mode parameters.
- ✓ Subject/target Index Degree of Parallelism (DOP) has been adjusted to achieve desired level of CPU saturation to improve Index create performance.

Design Point Options:

(Choose an Index Create Option):

A) Use Export/Import with the FULL=Y option:

- User indexes are Exported and created by the Import in the proper sequence as part of an entire database.
- Being the simplest approach, use this for smaller databases.
- **INDEXES=Y** is the default.

B) Use Export/Import to create user indexes in other modes:

- Export using the **INDEXES=Y** parameter to capture user-generated index metadata.
- Import using the **INDEXES=Y** parameter. Import will create the related user indexes as a step after the base tables are created/loaded. No optimizer statistics are gathered during index creation.
- Use if gathering database optimizer statistics afterwards.

C) Use Export/Import to generate user index create syntax:

- Export using the **INDEXES=Y** parameter to capture user-generated index metadata.
- Import using **INDEXES=Y** and **INDEXFILE=<file_name>** to capture user index create syntax to a file. Note: If **CONSTRAINTS=Y** is used, constraint syntax will also be captured to the file.
- Ensure the syntax file contains correct password and connect strings.
- No database objects are Imported. Re-run the Import with INDEXES=N to create the other database objects.
- Edit the user index create SQL file to include a "**compute statistics**" clause for each index create statement.
- Use the file of user index create SQL to create the indexes **after** table optimizer statistics have been gathered.
- Use this flexible approach with large, complex migrations to create user indexes separately after table data migration.

D) Use DBA maintained or third-party tool generated index create scripts:

- Export and Import using the **INDEXES=N** parameter.
- Ensure the "**compute statistics**" clause is included in each index create statement.
- Use external user index create SQL script(s) to create the indexes **after** table optimizer statistics have been gathered.
- Generally the fastest, most flexible method for large databases.

Export / Import Utility Design Decisions

- **Design Point (11): "*Sample Script Generation of Export parameter files for partitioned tables*"**

```
#!/bin/ksh
# Generate Parameter Files Oracle9i for Export of a 16-way hash partitioned tables
# PNUM input parameter supplies the starting partition number for the table
TABLENAME=$1
PNUM=$2
echo "Oracle export parameter generation started at " `date`
echo "Oracle export of TABLE = " $TABLENAME
PCNT=1
PMAX=16
while [ $PCNT -le $PMAX ]
do
echo "BUFFER=65535"       > $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "COMPRESS=N"      >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "CONSISTENT=N"    >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "DIRECT=Y"           >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "FEEDBACK=0"        >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo FILE=/exports/exp_${TABLENAME}_p${PCNT}.dmp >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "FULL=N"             >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "GRANTS=N"          >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "HELP=N"             >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "INDEXES=N"         >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo LOG=$MIGR_DIR/logs/exp_${TABLENAME}_p${PCNT}.log >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "POINT_IN_TIME_RECOVER=N" >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "RECORDLENGTH=65535" >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "RECORD=N"          >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "ROWS=Y"            >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "STATISTICS=NONE" >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
echo "TABLES=("${TABLENAME}":SYS_P"${PNUM}")" >> $MIGR_DIR/parms/exp_${TABLENAME}_p${PCNT}.par
PCNT=`expr $PCNT + 1`
PNUM=`expr $PNUM + 1`
done
echo "Oracle export parameter generation completed at " `date`
```

Export / Import Utility Design Decisions

- **Design Point (12): "*Sample Script Generation of Import parameter files for partitioned tables*"**

```
#!/bin/ksh
# Generate Parameter Files Oracle9i for Import of 16-way hash partitioned table
TABLENAME=$1
PNUM=$2
echo "Oracle import parameter generation started at " `date`
echo "Oracle import of TABLE = " $TABLENAME
PCNT=1
PMAX=16
while [ $PCNT -le $PMAX ]
do
echo "ANALYZE=N"       > $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "BUFFER=65535"   >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "COMMIT=N"       >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "DESTROY=N"      >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "FEEDBACK=0"     >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo FILE=/flatfiles/exports/exp_${TABLENAME}_p${PCNT}.dmp >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "FULL=N"         >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "GRANTS=N"       >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "HELP=N"         >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "IGNORE=Y"       >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "INDEXES=N"      >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo LOG=$MIGR_DIR/logs/imp_${TABLENAME}_p${PCNT}.log >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "POINT_IN_TIME_RECOVER=N" >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "RECORDLENGTH=65535" >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "ROWS=Y"         >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
echo "TABLES=("${TABLENAME}":SYS_P"${PNUM}")" >> $MIGR_DIR/parms/imp_${TABLENAME}_p${PCNT}.par
PCNT=`expr $PCNT + 1`
PNUM=`expr $PNUM + 1`
done
echo "Oracle import parameter generation completed at " `date`
```

HP DBFastTableCopy Tool Sample Run Script

```
#!/usr/bin/ksh -x
echo "TblCopy for Customer table Started at: " `date`
# Truncate existing customer table in target database
sqlplus tpch/tpch <<EOF
truncate table customer;
exit;
EOF
# Run TblCopy on HP-UX Integrity Server target system
export ROW_BUFFERS=$1
export READ_THREADS=$2
export WRITE_THREADS=$3
date
# Build the command line for TblCopy
# -O = <type of output> F=file, P=pipe, D=database I=info only
# -S = <source Oracle db service>
# -D = <destination Oracle db service
# -T = <name of Oracle table> to copy
# -u = <username/password>
# -P = <read partitioning scheme> P=partition, R=rowid, N=none
# -L = <path> for the destination file (number + .dat added)
# -F = <number> of files (hence load streams) to use
# -N = <total records> to copy (approximate - nearest multiple of buffer)
# -C = <number> of records to copy in a block (buffer)
# -X = no WRITE to either file or pipe, no load start
# -p = <list of partitions to copy> (otherwise all)
# -f = <prefetch row count>
# -q = prevent load processing from starting
# -r = number of read threads, if possible
# -Z = <trace level> for debugging - 0, 1, 2, 3 are valid
cmd = " "
cmd="${cmd} -O D"
cmd="${cmd} -S TPCH_src"
cmd="${cmd} -D TPCH_tgt"
cmd="${cmd} -L /flatfiles/exports/dbcopy"
cmd="${cmd} -F ${WRITE_THREADS}"
cmd="${cmd} -r ${READ_THREADS}"
cmd="${cmd} -C ${ROW_BUFFERS}"
cmd="${cmd} -N 45000000"
cmd="${cmd} -u tpch/tpch"
cmd="${cmd} -T customer"
cmd="${cmd} -P P"
#cmd="${cmd} -q"
cmd="${cmd} -Z 2"

./TblCopy ${cmd} > tblcopy_customer_C${ROW_BUFFERS}_r${READ_THREADS}_w${WRITE_THREADS}.log 2>&1

echo "TblCopy for Customer table Completed at: " `date`
```

HP DBCompare Utility Sample Run Script

```
#!/usr/bin/ksh -x
# -x

echo "DBCompare Utility Run Started at: " `date`

# Sample Run script for HP DBCompare Utility
#
#
export ORACLE_SID=tpch
echo $ORACLE_SID
date
DATABASE_A=$1
DATABASE_B=$2
DB_USERPASS=$4
COMP_TABLE=$3

# Build the command line for DBCompare
# Mandatory options

# -A = <Source Oracle db service>
# -B = <Target Oracle db service>
# -C = <number> of records to compare in each block (buffer)
# -T = <name of Oracle table> to compare
# -u = <username/password>

# optional options

# -P = <partition name> of partition to be compared
# -Z = <trace level> for debugging - 0, 1, 2, 3 are valid

cmd=" "

# EDIT these lines - for mandatory options

cmd="${cmd} -A ${TPCH_src}"
cmd="${cmd} -B ${TPCH_tgt}"
cmd="${cmd} -C 65530"
cmd="${cmd} -T ${ORDERS}"
cmd="${cmd} -u ${tpch/tpch}"

# EDIT OR COMMENT these lines - for optional options

#cmd="${cmd} -P ORDR_1"
cmd="${cmd} -Z 2"

./DBCompare ${cmd}   2>&1

echo "DBCompare Utility Run Completed at: " `date`
```


hp
invent

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional