Best Practices for Deploying a Data Warehouse on Oracle Database 11g

Maria Colgan – Principal Product Manager
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

• Hardware
  • Building a balanced configuration
  • Disk / ASM configuration

• Logical model
  • Third Normal Form
  • Star Schema

• Physical model
  • Implementing Logical models
  • Data Loading

• System management
  • Parallel Query
  • Statistic management
  • Initialization parameters
  • Workload monitoring
Hardware

“The weakest link” defines the throughput
Balanced Configuration

“The weakest link” defines the throughput

CPU Quantity and Speed dictate number of HBAs
capacity of interconnect

HBA Quantity and Speed dictate number of Disk Controllers
Speed and quantity of switches

Controllers Quantity and Speed dictate number of Disks
Speed and quantity of switches

Disk Quantity and Speed
Disk Layout - S.A.M.E Architecture with RAID 1 and ASM
Data Warehouse hardware configuration best practices

• Build a balance hardware configuration
  • Total throughput = # cores X 100-200 MB (depends on chip set)
  • Total HBA throughput = Total core throughput
    • If total core throughput =1.6GB will need 4 4Gb HBAs
  • Use 1 disk controller per HBA Port (throughput capacity must be equal)
  • Switch must be same capacity as HBA and disk controllers
  • Max of 10 physical disks per controller (Use smaller drives 146 or 300 GB)
• Minimum of 4GB of Memory per core (8GB if using compression)
• Interconnect bandwidth should equal IO bandwidth (Infiniband)
• Use ASM with RAID 1 mirroring for redundancy
  • Create two ASM diskgroups (1 for data, 1 for flash recovery area)
  • Use ATTRIBUTE 'au_size' to increase allocation unit
Data Warehousing Hardware Solutions

**Custom**
- Complete Flexibility
- Any OS, any platform
- Easy fit into a company’s IT standards

**Reference Configurations**
- Documented best-practice configurations for data warehousing

**Optimized Warehouse**
- Scalable systems pre-installed and pre-configured: ready to run out-of-the-box

**HP Oracle Database Machine**
- Highest performance
- Pre-installed and pre-configured
- Sold by Oracle

Pre-configured, Pre-installed, Validated

Complete Flexibility
Logical Models

Visual representation of business concepts and relationships
Why Schema Modeling is important

• Model according to your business needs
  • Don't get lost in academia
  • Ignore the physical DB, hardware or end-user tools that will eventually be used
• Two main data warehouse models
  • Third Normal Form (3NF)
    • Minimizes data redundancy through normalization
    • Typically has a large number of tables
  • Star Schema
    • Simplest model
    • One fact table surrounded by multiple dimension tables
• Upon completion model should be
  • Easy to map to fact and dimension tables in physical database
  • Show clearly how information in operational systems will fit in data warehouse
Industry best practices on when to use each model

- 3rd Normal Form
  - Preserve a detailed record of each transaction without any data redundancy
  - Allows for rich encoding of attributes & all relationships between data elements
  - Users typically require a solid understanding of the data in order to navigate

- Star Schema
  - Dimensional approach that simplified the data model to facilitate access
  - Drill paths, hierarchy and query profile are embedded in the data model itself rather than the data
  - Easier for in-experienced users to navigate

- Forrester:
  - 3rd Normal Form is the self-less model (Neutral)
  - Star Schema is the selfish model (Subject oriented)
What does a 3\textsuperscript{rd} Normal Form schema look like?

- Large number of tables due to normalization
- Multiple fact tables
- Lots of large table joins
Optimizing 3\textsuperscript{rd} Normal Form
Requires 3 Ps - Power, Partitioning, Parallelism

- larger tables or fact tables should be partitioned
  - Use composite partitioning range-hash
  - Range to facility the data load and data elimination
  - Hash on join column to facility partition wise joins
  - Number of hash partitions should be power of 2 (#CPU X 2)

- Parallel Execution should be used
  - Instead of one process doing all the work multiple processes working concurrently on smaller units
  - Parallel degree should be power of 2

Goal is parallel partition wise joins
Partition Pruning

Q: What was the total sales for the weekend of May 20 - 22 2008?

Select sum(sales_amount)
From SALES
Where sales_date between to_date('05/20/2008','MM/DD/YYYY')
And to_date('05/23/2008','MM/DD/YYYY');

Only the 3 relevant partitions are accessed
**Partition Wise join**

Both tables have the same degree of parallelism and are partitioned the same way on the join column (cust_id)

A large join is divided into multiple smaller joins, each joins a pair of partitions in parallel

---

```sql
Select sum(sales_amount) 
From 
SALES s, CUSTOMER c 
Where s.cust_id = c.cust_id;
```
## Execution plan for partition-wise join

<table>
<thead>
<tr>
<th>ID</th>
<th>Operation</th>
<th>Name</th>
<th>Pstart</th>
<th>Pstop</th>
<th>TQ</th>
<th>PQ Distrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QC (RAND)</td>
</tr>
<tr>
<td>2</td>
<td>PX SEND QC (RANDOM)</td>
<td>:TQ10001</td>
<td></td>
<td></td>
<td>Q1,01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SORT GROUP BY</td>
<td></td>
<td></td>
<td></td>
<td>Q1,01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PX RECEIVE</td>
<td></td>
<td></td>
<td></td>
<td>Q1,01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PX SEND HASH</td>
<td>:TQ10000</td>
<td></td>
<td></td>
<td>Q1,00</td>
<td>HASH</td>
</tr>
<tr>
<td>6</td>
<td>SORT GROUP BY</td>
<td></td>
<td></td>
<td></td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PX PARTITION HASH ALL</td>
<td></td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TABLE ACCESS FULL</td>
<td>Customers</td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TABLE ACCESS FULL</td>
<td>Sales</td>
<td>1</td>
<td>128</td>
<td>Q1,00</td>
<td></td>
</tr>
</tbody>
</table>

Partition Hash All above the join & single PQ set indicate partition-wise join.
Partial Partition Wise join

Only the Sales table is hash partitioned on the cust_id column

Rows from customer are dynamically redistributed on the join key cust_id to enable partition-wise join

Select `sum(sales_amount)`
From
SALES s, CUSTOMER c
Where s.cust_id = c.cust_id;
What does a Star Schema look like?

- Called star schema because diagram resembles a star
- The center of the star consists of one or more fact tables
- The points of the star are the dimension tables
Select SUM(quantity_sold) total_umbrellas_sold_in_Boston

From Sales s, Customers c, Products p, Times t

Where s.cust_id = c.cust_id
And s.prod_id = p.prod_id
And s.time_id = t.time_id
And c.cust_city = 'BOSTON'
And p.product = 'UMBRELLA'
And t.month = 'MAY'
And t.year = 2008;

Q: What was the total number of umbrellas sold in Boston during the month of May 2008?
Optimizing Star schema

- Create bitmap index on foreign key columns in fact table
- Set STAR_TRANSFORMATION_ENABLED to TRUE

Goal is star transformation

- Powerful optimization technique that rewrites or transform SQL
- Executes the query in two phases
- The first phase retrieves necessary rows (row set) from the fact table
  - Bitmap joins between bitmap indexes on all of the foreign key columns
- The second phase joins this row set to the dimension tables
  - The join back to the dimension tables done using a hash join
Star Transformation in detail

Select SUM(quantity_sold) 
From Sales s, Customers c, Products p, Times t
Where s.cust_id = c.cust_id 
And s.prod_id = p.prod_id 
And s.time_id = t.time_id 
And c.cust_city = 'BOSTON' 
And p.product = 'UMBRELLA' 
And t.month = 'MAY' 
And t.year = 2008;

Step 1: Oracle rewrites / transforms the query to retrieve only the necessary rows from the fact table using bitmap indexes on foreign key columns

Step 2: Oracle joins the rows from fact table to the dimension tables

Select SUM(quantity_sold) 
From Sales s 
Where s.cust_id IN 
(Select c.cust_id From Customers c Where c.cust_city = 'BOSTON') 
And s.prod_id IN 
(Select p.prod_id From Products p where p.product = 'UMBRELLA') 
And s.time_id IN 
(Select t.time_id From Times t Where t.month = 'MAY' And t.year = 2008);
### Execution plan for Star Query

<table>
<thead>
<tr>
<th>ID</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Pstart</th>
<th>Pstop</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT GROUP BY NO SORT</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCTS</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>TIMES</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PARTITION RANGE SUBQUERY</td>
<td></td>
<td>44144</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY LOCAL INDEX ROWID</td>
<td>SALES</td>
<td>44144</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>BITMAP CONVERSION TO ROWIDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>BITMAP AND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BITMAP MERGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BITMAP KEY ITERATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BUFFER SORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TABLE ACCESS FULL</td>
<td>TIMES</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>BITMAP INDEX RANGE SCAN</td>
<td>SALES_TIME_BIX</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>BITMAP MERGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>BITMAP KEY ITERATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>BUFFER SORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>TABLE ACCESS FULL</td>
<td>CUSTOMERS</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>BITMAP INDEX RANGE SCAN</td>
<td>SALES_TIME_BIX</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>BITMAP MERGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>BITMAP KEY ITERATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>BUFFER SORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCTS</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>BITMAP INDEX RANGE SCAN</td>
<td>SALES_PROD_BIX</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Phase 1**
- ID 3: TABLE ACCESS FULL
- ID 4: HASH JOIN
- ID 5: TABLE ACCESS FULL
- ID 6: PARTITION RANGE SUBQUERY
- ID 7: TABLE ACCESS BY LOCAL INDEX ROWID
- ID 8: BITMAP CONVERSION TO ROWIDS
- ID 9: BITMAP AND
- ID 10: BITMAP MERGE
- ID 11: BITMAP KEY ITERATION
- ID 12: BUFFER SORT
- ID 13: TABLE ACCESS FULL
- ID 14: BITMAP INDEX RANGE SCAN
- ID 15: BITMAP MERGE
- ID 16: BITMAP KEY ITERATION
- ID 18: TABLE ACCESS FULL
- ID 19: BITMAP INDEX RANGE SCAN
- ID 20: BITMAP MERGE
- ID 21: BITMAP KEY ITERATION
- ID 22: BUFFER SORT
- ID 23: TABLE ACCESS FULL

**Phase 2**
- ID 0: SELECT STATEMENT
- ID 1: SORT GROUP BY NO SORT
- ID 2: HASH JOIN
- ID 3: TABLE ACCESS FULL
- ID 4: HASH JOIN
- ID 5: TABLE ACCESS FULL
- ID 6: PARTITION RANGE SUBQUERY
- ID 7: TABLE ACCESS BY LOCAL INDEX ROWID
- ID 8: BITMAP CONVERSION TO ROWIDS
- ID 9: BITMAP AND
- ID 10: BITMAP MERGE
- ID 11: BITMAP KEY ITERATION
- ID 12: BUFFER SORT
- ID 13: TABLE ACCESS FULL
- ID 14: BITMAP INDEX RANGE SCAN
- ID 15: BITMAP MERGE
- ID 16: BITMAP KEY ITERATION
- ID 23: TABLE ACCESS FULL
- ID 24: BITMAP INDEX RANGE SCAN
Physical Models

Implementing the logical model
Can both logical models co-exist?
Physical implementation of Logical Model
Blueprint of physical layers in Data warehouse
Data Loading

Staging layer and beyond
Data Loading Best Practices

• **External Tables**
  • Allows flat file to be accessed via SQL PL/SQL as if it was a table
  • Enables complex data transformations and data cleansing to occur ‘on the fly’
  • Avoids space wastage

• **Direct Path in parallel**
  • Bypasses buffer cache and writes data directly to disk via multi-block async IO
  • Use parallel to speed up load
  • Remember to use `Alter session enable parallel DML`

• **Range Partitioning**
  • Enables partition exchange loads
Partition Exchange loading

1. Create external table for flat files
2. Use CTAS command to create non-partitioned table TMP_SALES
3. Create indexes
4. Gather Statistics
5. Alter table Sales exchange partition May_24_2008 with table tmp_sales

Sales table now has all the data
System management

Maximizing Resources and avoiding “tuning spiral of death”
System management
Keeping the lights on for your data warehouse

- Parallel Execution
  - Use common sense to apply parallelism only where it will help performance and not hinder it
- Resource Manager
  - If your data warehouse is CPU bound protects critical tasks from interference from non-critical tasks
- Always have accurate Optimizer statistics
  - Use INCREMENTAL statistic maintenance or copy_stats to keep large partitioned fact - table up to date
- Set only the initialization parameters that you need to
  - Avoid tuning spiral of death by not tuning
- Workload Monitoring
  - Take hourly AWR or statspack report
  - Use EM to do real-time system monitoring
    - New Parallel Execution and SQL Monitoring screen in 11g
SQL Parallel Execution

QC is the user session that initiates the parallel SQL statement & it will distribute the work to parallel servers.

Parallel servers communicate among themselves & the QC using messages that are passed via memory buffers in the shared pool.

Parallel servers - individual sessions that perform work in parallel. They are allocated from a pool of globally available parallel server processes and assigned to a given operation.

Messages  ↔  Parallel server connection  ↔  QC connection
SELECT c.cust_name, s.purchase_date, s.amount
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;

Parallel Servers do majority of the work
Controlling PQ on RAC
Using services

Create two services

Srvctl add service –d database_name
   -s ETL
   -r sid1, sid2

Srvctl add service –d database_name
   -s AHOC
   -r sid3, sid4

Note: Prior to 11g use init.ora parameters instance_groups and parallel_instance_group to control PQ on RAC
Use Parallel Execution with common sense

- Parallel execution provides performance boost but requires more resources
- General rules of thumb for determining the appropriate DOP
  - objects smaller than 200 MB should not use any parallelism
  - objects between 200 MB and 5GB should use a DOP of 4
  - objects beyond 5GB use a DOP of 32

Settings may vary on your system- either in size range or DOP - and highly depend on your target workload & hardware configuration
Efficiency Statistics Management

• How do I gather accurate Statistics
  • “.. Compute statistics gives accurate results but takes too long ..”
  • “.. Sampling is fast but not always accurate ..”
  • “.. AUTO SAMPLE SIZE does not always work with data skew ..”
• New groundbreaking implementation for AUTO SAMPLE SIZE
  • Faster than sampling
  • Accuracy comparable to compute statistics

• Gathering statistics on one partition (e.g. after a bulk load) causes a full scan of all partitions to gather global table statistics Extremely time and resource intensive
• Use incremental statistics
  • Gather statistics for touched partition(s) ONLY
  • Table (global) statistics are built from partition statistics
Incremental Global Statistics

1. Partition level stats are gathered & synopsis created

Sales Table
- May 18th 2008
- May 19th 2008
- May 20th 2008
- May 21st 2008
- May 22nd 2008
- May 23rd 2008

S1
S2
S3
S4
S5
S6

2. Global stats generated by aggregating partition synopsis

Global Statistic

Sysaux Tablespace
3. A new partition is added to the table & Data is Loaded

4. Gather partition statistics for new partition

5. Retrieve synopsis for each of the other partitions from Sysaux

6. Global stats generated by aggregating the original partition synopsis with the new one
Managing your workload
Oracle Database Resource Manager

- If your data warehouse is CPU bound, consider using CPU Resource Management
  - Protects critical tasks from interference from non-critical tasks
  - Allows CPU to be utilized according to a specific ratio
  - Prevents thrashing and system instability that can occur with excessive CPU loads
To use Resource Manager:

1. Define Consumer Groups for each type of workload
   - Priority DSS consumer group
   - Normal DSS consumer group
   - Maintenance consumer group

2. Create rules to dynamically map sessions to consumer groups, based on session attributes

   **Mapping Rules**
   - service = ‘PRIORITY’
   - Oracle username = ‘LARRY’
   - Oracle username = ‘Maria’
   - client program name = ‘ETL’
   - function = ‘BACKUP’
   - query has been running > 1 hour

   **Consumer Groups**
   - Priority DSS
   - Normal DSS
   - Maintenance

3. Define Resource Plan: policy for managing workloads
Oracle Database Resource Manager

- Resource Manager can manage Degree Of Parallelism (DOP)
  - Specify maximum DOP for each consumer group
    - “OLTP” consumer group: max DOP = 0 (serial queries only)
    - “Low-Priority” consumer group: max DOP = 4
    - “Critical Reports” consumer group: max DOP = unlimited
  - Actual DOP determined by
    - Hints and parallel execution parameters -> proposed DOP
    - MIN(proposed DOP, consumer group’s max DOP) -> actual DOP
Initialization parameters

Only set what you really need to

- **Db_block_size**
  - 8, 16 or 32 (Larger may help with compression ratio)

- **Db_file_multiblock_read_count**
  - 1024/db_block_size

- **Memory_target / SGA_target & PGA_Aggregate_target**
  - Shared_pool needs to be size to accommodate PQ message buffers

- **Parallel_min_servers**
  - DOP X Avg # Concurrent Queries

- **Parallel_max_servers**
  - DOP X Max # Concurrent Queries

- **Parallel_execution_message_size**
  - 16K

- **Parallel_adaptive_multi_user**
  - False

- **Star_transformation_enabled**
  - True (if have star schema)
Workload monitoring

- Take hourly AWR or statspack snapshots
  - Establish baseline performance
- Use V$ performance views for command-line real-time monitoring
  - V$_session, (G)V$_PQ, (G)V$_PX, (G)V$PX_PROCESS
  - New view (G)V$SQL_MONITOR
    - Enables real-time monitoring of long-running or parallel SQL statements
- Or use Oracle Enterprise Manager Database Control 11g for
  - New Parallel Execution screens
    - Identifies all parallel execution activity
    - Displays how and where parallel resources are being used
  - SQL Monitoring screens
    - Visually identifies which parts of an execution plan are expensive relative to overall cost of statement
    - Provides information about the parallel server sets & work distribution
For More Information

http://search.oracle.com

OR

If you have more questions later, feel free to ask.
### Campground Demos

<table>
<thead>
<tr>
<th>Demo</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Warehousing with Oracle Database 11g – booth L28</td>
<td>Moscone West Exhibit Hall</td>
</tr>
<tr>
<td>Oracle Optimized Warehouses - booth L24</td>
<td>Moscone West Exhibit Hall</td>
</tr>
<tr>
<td>Oracle Partitioning – booth L34</td>
<td>Moscone West Exhibit Hall</td>
</tr>
<tr>
<td>Oracle Data Mining – booth L21</td>
<td>Moscone West Exhibit Hall</td>
</tr>
<tr>
<td>Oracle OLAP – booth L27</td>
<td>Moscone West Exhibit Hall</td>
</tr>
</tbody>
</table>
# Recommended Sessions

<table>
<thead>
<tr>
<th>Session Title</th>
<th>Date</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Warehouse Builder Road Map</td>
<td>Tuesday, Sept 23</td>
<td>9:00 a.m.</td>
<td>Moscone South: 102</td>
</tr>
<tr>
<td>Oracle Database 11g: Stories from a Data Warehouse Implementation</td>
<td>Tuesday, Sept 23</td>
<td>1:00 p.m.</td>
<td>Moscone South: 304</td>
</tr>
<tr>
<td>Get the Best Out of Oracle Partitioning: Things You Always Wanted to Know</td>
<td>Wednesday, Sept. 24</td>
<td>1:00 p.m.</td>
<td>Moscone South: 104</td>
</tr>
<tr>
<td>Oracle's New Database Accelerator: Query Processing Revolutionized</td>
<td>Thursday, Sept. 25</td>
<td>9:00 a.m.</td>
<td>Moscone South: 103</td>
</tr>
<tr>
<td>Oracle Database 11g for Data Warehousing</td>
<td>Thursday, Sept. 25</td>
<td>10:30 a.m.</td>
<td>Moscone South: 103</td>
</tr>
<tr>
<td>Oracle Real Application Clusters and QLogic InfiniBand: Yahoo! Large-Scale Data Warehouse</td>
<td>Thursday, Sept. 25</td>
<td>3:00 p.m.</td>
<td>Moscone South: 306</td>
</tr>
</tbody>
</table>