Oracle Disclaimer: The following is intended to outline Oracle's 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.
Audience Knowledge / Versions

- Oracle7 Experience?
- Oracle8i Experience?
- Oracle9i Experience?
- Oracle10g Experience?
- Oracle Database 11g Experience?

Goals
- Present NEW features in an EASY way
- Focus on a few nice features of Oracle11g

Non-Goals
- Learn ALL aspects of Oracle11g
Overview

- Start Me Up – Using Memory Target
- The Result Cache
- Invisible Indexes & Online Index Rebuilds
- Nice Developer Features
- ADDM Enhancements
- SQL Plan Management and SQL Plan Baselines
- SQL Query Repair Advisor
- SQL Performance Analyzer
- Real Application Testing (Database Capture and Replay)
- Interval Partitioning & Partition Compression
- DBA Tools and DBMS_STATS Enhancements
- Grid Control & EM
- Security Enhancements & the Future Sizes
- Summary
Know the Oracle

"I admire risk takers. I like leaders – people who do things before they become fashionable or popular. I find that kind of integrity inspirational."

LAWRENCE J. ELLISON | Chairman & Chief Executive Officer, 2003
Oracle Firsts – Innovation!

1979 First commercial SQL relational database management system
1983 **First 32-bit** mode RDBMS
1984 First database with read consistency
1987 **First client-server** database
1994 First commercial and multilevel secure database evaluations
1995 **First 64-bit** mode RDBMS
1996 First to break the 30,000 TPC-C barrier
1997 **First Web** database
1998 First Database - Native **Java** Support; Breaks 100,000 TPC-C
1998 First Commercial RDBMS ported to **Linux**
2000 First database with **XML**
2001 First middle-tier database cache
2001 First RDBMS with **Real Application Clusters**
2004 First **True Grid Database**
2005 First **FREE Oracle Database** (10g Express Edition)
2006 First **Oracle Support for LINUX** Offering
2007 **Oracle 11g Released!**
2007: Version 11g Arrives

- The Focus has been Acquisitions and gaining Market Share
- Oracle 11g Database extends an already large lead
  - Easier to Manage the Database – Better Grid Control
  - Self Tuning through a variety of tools (Makes 1 person equal 10)
  - Better Security/Encryption & Recoverability via Flashback
  - Better Testing Tools (Real Application Testing)
- Andy Mendelsohn is still the database lead
- New releases of Siebel, PeopleSoft and Oracle12 Apps.
- New Oracle BI Suite & Acquisition of Hyperion
Testing the **Future** Version

Version 11.1.0.6.0 of the Database
# Oracle Database 11g Release 1: Upgrade Paths

## Direct Upgrade Path

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2.0.4.0 (or higher)</td>
<td>11.1.x</td>
</tr>
<tr>
<td>10.1.0.2.0 (or higher)</td>
<td>11.1.x</td>
</tr>
<tr>
<td>10.2.0.1.0 (or higher)</td>
<td>11.1.x</td>
</tr>
</tbody>
</table>

## In-Direct Upgrade Path

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Upgrade Path for Target Database</th>
<th>Target Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.3.0.0 (or lower)</td>
<td>7.3.4.x -- &gt; 9.2.0.8</td>
<td>11.1.x</td>
</tr>
<tr>
<td>8.0.5.0.0 (or lower)</td>
<td>8.0.6.x -- &gt; 9.2.0.8</td>
<td>11.1.x</td>
</tr>
<tr>
<td>8.1.7.0.0 (or lower)</td>
<td>8.1.7.4 -- &gt; 9.2.0.8</td>
<td>11.1.x</td>
</tr>
<tr>
<td>9.0.1.3.0 (or lower)</td>
<td>9.0.1.4 -- &gt; 9.2.0.8</td>
<td>11.1.x</td>
</tr>
</tbody>
</table>
Database Upgrade Assistant (DBUA)

- Command Line Option to Auto Extend System Files
- Express Edition Upgrade to others
- Integration with Oracle Database 11g Pre-upgrade Tool
- Moving Data Files into ASM, SAN, and Other File Systems
- Oracle Base and Diagnostic Destination Configuration
Database Upgrade Assistant (DBUA)

- DBUA checks before the upgrade:
  - Invalid user accounts or roles
  - Invalid data types or invalid objects
  - De-supported character sets
  - **Adequate resources** (rollback segments, tablespaces, and free disk space)
  - Missing SQL scripts needed for the upgrade
  - Listener running (if Oracle Enterprise Manager Database Control upgrade or configuration is requested)
  - Oracle Database software linked with Database Vault option. If Database Vault is enabled, Disable Database Vault before upgrade.
$ sqlplus ***/**

Copyright (c) 1982, 2007, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL > startup
ORACLE instance started.
Total System Global Area 422670336 bytes
Fixed Size 1300352 bytes
Variable Size 306186368 bytes
Database Buffers 109051904 bytes
Redo Buffers 6131712 bytes

Database mounted.
Database opened.
Database Information - UP!

Monitor Database (UP)

Users are Definitely Using it!
MEMORY_TARGET & Automatic Memory Management
• First there was some Automatic Memory Mgmt - 9i
  – SGA_MAX_SIZE introduced in 9i – Dynamic Memory
  – No more Buffers – DB_CACHE_SIZE
  – Granule sizes introduced - _ksm_granule_size
• Then came SGA_TARGET – 10g
  – Oracle Applications recommends setting this for SGA
  – Set minimums for key values (Data Cache / Shared Pool)
SGA & PGA will be MEMORY_TARGET
Moving from SGA_TARGET to: MEMORY_TARGET - EM
ADDM Enhancements
(Automatic Database Diagnostic Monitor)
ADDM enhancements

- Global ADDM so that Diagnostics are done across the entire cluster
- Emergency ADDM for use when database is hung
- On any granularity
  - Database Cluster
  - Database Instance
  - Specific Target (such as host, ASM...etc.)
- Over a specified time **NOT** tied to a pair of snapshots
ADDM Briefly

Specific Database Instance

We have 5 ADDM Findings

Check them Here
ADDM Briefly

Top ADDM Findings

Click a Single Timeframe

Let's Check the Hard Parse Issue
ADDM Briefly

Detailed Info & Findings

Add’l Info

Performance Finding Details: Hard Parse Due to Parse Errors

Finding: Hard parsing SQL statements that encountered parse errors was consuming significant database time.

Impact (%): 30

Period Start Time: Mar 22, 2007 10:00:23 PM CDT

Period Duration (minutes): 66.7

Filtered: No filters

Recommendations

Show All Details | Hide All Details

Details Category

- Application Analysis: Benefit (%) 30

Action: Investigate application logic to eliminate parse errors.

Findings Path

Expand All | Collapse All

Findings

- Hard parsing SQL statements that encountered parse errors was consuming significant database time.
- Hard parsing of SQL statements was consuming significant database time.
- Contention for latches related to the shared pool was consuming significant database time.
- Wait class "Sort cache" was consuming significant database time.

Additional Information

Waits for "library cache lock" amounted to 6% of database time.
A Big Problem Occurs

Run ADDM NOW!
ADDM - Run NOW!

Are you Sure?

Running
ADDM – Run NOW!

Done.

CPU Issue
ADDM – Run NOW!

Detail on CPU Issue?
Suggested Fixes
ADDMin – Run NOW!

View The Report

<table>
<thead>
<tr>
<th>Description</th>
<th>Active Sessions</th>
<th>Percent of Activity</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CPU Usage</td>
<td>3.52</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>2. Top SQL by DB Time</td>
<td>1.3</td>
<td>36.86</td>
<td>2</td>
</tr>
<tr>
<td>3. Hard Parse Due to Parse Errors</td>
<td>.9</td>
<td>25.56</td>
<td>1</td>
</tr>
<tr>
<td>4. &quot;User I/O&quot; wait Class</td>
<td>.81</td>
<td>22.69</td>
<td>0</td>
</tr>
<tr>
<td>5. PL/SQL Execution</td>
<td>.66</td>
<td>18.87</td>
<td>2</td>
</tr>
<tr>
<td>6. &quot;Scheduler&quot; Wait Class</td>
<td>.54</td>
<td>15.28</td>
<td>0</td>
</tr>
<tr>
<td>7. Hard Parse Due to Invalidations</td>
<td>.3</td>
<td>8.6</td>
<td>1</td>
</tr>
<tr>
<td>8. Top Segments by I/O</td>
<td>.26</td>
<td>7.44</td>
<td>1</td>
</tr>
<tr>
<td>9. Undersized instance memory</td>
<td>.10</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Findings and Recommendations
• Performance expert in a box
  - Now RAC specialist too!
• Identifies the most “Globally Significant” performance issues for the entire RAC database
• Database-wide and instance-level analysis
• Database-wide analysis of:
  - Global cache interconnect issues
  - Lock manager congestion issues
  - Global resource contention, e.g. IO bandwidth, hot blocks
  - Globally high-load SQL
  - Skew in instance response times
• Allows drill down to instances
• Runs proactively every hour when taking AWR Snapshots (default)
ADDM Considerations:

- **CPU Bottlenecks**
- **Undersized Memory Structures** – SGA / PGA
- **I/O Capacity Issues**
- **High Load SQL statements**
- **High Load PL/SQL**
- **RAC specific issues** – Global hot block/interconnect
- **Application issues** such as parsing, locks...etc.
- **Concurrency (buffer busy)** or **hot object issues**
- **Configuration issues** – Redo, Archive
SQL Tuning Advisors & SQL Plan Management (SPM)
SQL Plan Management

• SQL Plan Management is a mechanism that records/evaluates execution plan of SQL statements (good & bad) over time and builds SQL Plan baselines (replaces stored outlines) of existing plans known to be efficient.

• Events that cause the need for SQL Plan baselines:
  – New version of Oracle (New optimizer version – Use capture replay to test effect)
  – Changes to optimizer statistics or data changes
  – Schema, application or metadata changes (use SQL Advisor to get suggestions)
  – System settings changes (Use SQL Replay to find what works)
  – SQL Profile (statistics – data skews & correlated columns) creation

• Stored outlines are deprecated (discouraged) in Oracle Database 11g. Oracle highly recommends migrating existing stored outlines to SQL plan baselines. A SQL Profile contains additional STATISTICS for this SQL statement for the query optimizer to generate a better execution plan. An outline/baseline contains HINTS for this SQL statement for query optimizer to generate a
SQL Plan Management

- SQL Profile stores **STATISTICS** for a SQL statement for the query optimizer to generate a better execution plan.
- A Stored Outline/SQL Plan Baseline contains **HINTS** for this SQL statement for query optimizer to generate a better execution plan.
- A SQL Plan Baseline should evolve with changes in the system to analyze good/bad plans over time.
- View these in DBA_PLAN_BASELINES
- You can also export a SQL Tuning Set and import it to a new system. **Capture baselines for Tuning Set with DBMS_SPM** (see later slide on entire syntax). Can also use a pack/unpack function to pack/unpack all plans in a system for transporting.
SQL Plan Management
Create a SQL Tuning Set

Tuning Issue
Create a Tuning Set from Top 10 SQL
SQL Plan Management
Create a SQL Tuning Set

Tuning Set Name

Queries

```
SELECT de.OWNER || ''|| de.segment_name segment_name, de.segment_type segment_type, de.extent_id extent#, bh.dbablk - de.block_id + 1
FROM dual;
BEGIN EMOTIFICATION.QUEUE_READY(1, 2, 2); END;
SELECT X FROM DUAL;
BEGIN execute immediate 'alter session set NLS_NUMERIC_CHARACTERS = "-","."'; end;
SELECT event#, sql_id, sql_plan_hash_value, sql_opcode, session_id, session_serial#, module, action, client_id, DEC(WAIT_TIME, 0, 'W', 'C'), 1,
time_waited, service_hash, user_id, program, sample_f...
BEGIN EMOW fortunate(MGMT_NOB ENGINE. MODULE_NAME, 1); MGMT_NOB ENGINE.get_scheduled_steps(2, 3, 4, 5);
END;
BEGIN MGMT缾PAP,DEQUE_URL_REQUEST(p_node_id => 1, p_wait => 2, x_xml data => 3, x_request_id => 4, x_timestamp => 5,
x_return_status => 6); END;
BEGIN MGMT缾PAP,DEQUE_URL_REQUEST(p_node_id => 1, p_wait => 2, x_xml data => 3, x_request_id => 4, x_timestamp => 5,
x_return_status => 6); END;
BEGIN dbms_application_info.set_module(1, 2); dbms_application_info.set_client_info(3); dbms_session.set_identifier(4); end;
BEGIN EMOW fortunate(MGMT_NOB ENGINE. MODULE_NAME, 1); MGMT_NOB ENGINE.get_scheduled_steps(2, 3, 4, 5);
END;
```

Copyright © 1996, 2007, Oracle. All rights reserved.
Oracle, J D Edwards, PeopleSoft, and Red Hat are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager
SQL Plan Management
Viewing a SQL Tuning Set
SQL Plan Management
Create a SQL Tuning Set

Run the Tuning Advisor on this SQL Tuning Set (STS)
Run it NOW
SQL Plan Management
Create a SQL Tuning Set

Results
Select One query
And click View
SQL Plan Management
Click on any SQL ID

SQL Text

Waits & Statistics
SQL Plan Management
Create a SQL Tuning Set

SQL Profile Will Help 99%
SQL Plan Management
Create a SQL Tuning Set

Compare Before & After

---

### SQL Plan Management

**Create a SQL Tuning Set**

![Comparison of SQL Plans](image)

**Original Explain Plan (Annotated)**

- Indicates an adjustment from the original plan by the SQL Tuning Advisor
- Plan Hash Value: 2347322369

<table>
<thead>
<tr>
<th>Operation</th>
<th>Line ID</th>
<th>Object</th>
<th>Object Type</th>
<th>Order Rows</th>
<th>Bytes</th>
<th>Cost Time</th>
<th>CPU Cost</th>
<th>I/O Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>0</td>
<td></td>
<td></td>
<td>121</td>
<td>0.270</td>
<td>983,655</td>
<td>12,350,714</td>
<td>168,630</td>
</tr>
<tr>
<td>SORT ORDER BY</td>
<td>1</td>
<td></td>
<td></td>
<td>120</td>
<td>0.270</td>
<td>983,655</td>
<td>12,350,714</td>
<td>168,630</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>2</td>
<td></td>
<td></td>
<td>119</td>
<td>0.270</td>
<td>983,654</td>
<td>12,350,714</td>
<td>168,630</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>3</td>
<td></td>
<td></td>
<td>7</td>
<td>1.708</td>
<td>8,647,788</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>0.176</td>
<td>710,600</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VIEW</td>
<td>5</td>
<td></td>
<td></td>
<td>3</td>
<td>0.013</td>
<td>355,300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SORT AGGREGATE</td>
<td>6</td>
<td></td>
<td>FIXED TABLE FULL</td>
<td>2</td>
<td>0.059</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>SYS.XSKSLTR_CHILDREN</td>
<td>TABLE (FIXED)</td>
<td>1</td>
<td>120.000</td>
<td>355,300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIXED TABLE FULL</td>
<td>8</td>
<td>SYS.XSKSLTR_CHILDREN</td>
<td>TABLE (FIXED)</td>
<td>4</td>
<td>7.670</td>
<td>355,300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIXED TABLE FULL</td>
<td>9</td>
<td>SYS.XBKH</td>
<td>TABLE (FIXED)</td>
<td>6</td>
<td>6.738</td>
<td>350,000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VIEW</td>
<td>10</td>
<td>SYS.DBA_EXTENTS</td>
<td>VIEW</td>
<td>118</td>
<td>0.114</td>
<td>89,423,1,074</td>
<td>1,122,790,014,976</td>
<td>15,330</td>
</tr>
</tbody>
</table>

**New Explain Plan With SQL Profile**

- Plan Hash Value: 2138758942

<table>
<thead>
<tr>
<th>Operation</th>
<th>Line ID</th>
<th>Object</th>
<th>Object Type</th>
<th>Order Rows</th>
<th>Bytes</th>
<th>Cost Time</th>
<th>CPU Cost</th>
<th>I/O Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>0</td>
<td></td>
<td></td>
<td>124</td>
<td>0.262</td>
<td>1,972</td>
<td>702,635</td>
<td>1,926</td>
</tr>
<tr>
<td>SORT ORDER BY</td>
<td>1</td>
<td></td>
<td></td>
<td>123</td>
<td>0.262</td>
<td>1,972</td>
<td>702,635</td>
<td>1,926</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>2</td>
<td></td>
<td></td>
<td>122</td>
<td>0.262</td>
<td>1,971</td>
<td>687,481</td>
<td>1,926</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>3</td>
<td></td>
<td></td>
<td>7</td>
<td>1.568</td>
<td>1</td>
<td>8,647,788</td>
<td>0</td>
</tr>
<tr>
<td>VIEW</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>0.176</td>
<td>0</td>
<td>710,600</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>3</td>
<td>0.013</td>
<td>0</td>
<td>355,300</td>
<td>0</td>
</tr>
<tr>
<td>SORT AGGREGATE</td>
<td>6</td>
<td></td>
<td>FIXED TABLE FULL</td>
<td>2</td>
<td>0.059</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>SYS.XSKSLTR_CHILDREN</td>
<td>TABLE (FIXED)</td>
<td>1</td>
<td>120.000</td>
<td>355,300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIXED TABLE FULL</td>
<td>8</td>
<td>SYS.XSKSLTR_CHILDREN</td>
<td>TABLE (FIXED)</td>
<td>4</td>
<td>7.670</td>
<td>355,300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIXED TABLE FULL</td>
<td>9</td>
<td>SYS.XBKH</td>
<td>TABLE (FIXED)</td>
<td>6</td>
<td>6.738</td>
<td>350,000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VIEW</td>
<td>10</td>
<td>SYS.DBA_EXTENTS</td>
<td>VIEW</td>
<td>121</td>
<td>0.222</td>
<td>235</td>
<td>14,023,343</td>
<td>234</td>
</tr>
<tr>
<td>UNION-ALL</td>
<td>11</td>
<td></td>
<td></td>
<td>120</td>
<td>0.222</td>
<td>235</td>
<td>14,023,343</td>
<td>234</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>12</td>
<td></td>
<td></td>
<td>72</td>
<td>0.222</td>
<td>235</td>
<td>14,023,343</td>
<td>234</td>
</tr>
</tbody>
</table>
SQL Plan Control

SQL Profiles stored in the system

SQL Profiles

SQL Plan Baselines
SQL Advisors

Tuning Advisors

Repair Advisor (next)
The SQL Repair Advisor

ORA-600
SQL Repair Advisor

- Used to Repair Problem SQL – Oracle Errors
- Reloads and recompiles SQL statements to gather diagnostics information to fix.
- Uses the diagnostic information to repair the problem SQL statement (DBMS_SQLDIAG)
- Will fix error going through compilation, execution and trying different routes (could be a slower route for now) to come up with a temporary SQL Patch without error until fixed.
SQL Repair Advisor – Go straight from Alerts

Go to the Database Instance

Click Alert (ORA-600) message text to see details
Click on View Problem Details to go to the Support Bench
Support Workbench - Details

Click on SQL Repair Advisor
Results from SQL Repair Advisor

Click on View to Get the Detail finding of the Advisor

Note a SQL Patch (FIX for the SQL) has been generated
Click on Implement To accept the SQL Patch

---

### SQL Repair Advisor

#### Recommendation / Confirmation

**Repair Recommendations for SQL ID: 9m7mvytcb4d14**

Select the desired recommendation and then click on the Implement button to apply the SQL patch, which is a special type of SQL Profile that will repair the SQL statement.

**SQL Text:**

```sql
delete from t1 where t1.a = 'a' and rowid <> (select max(rowid) from t2 where t1.a = t2.a and t1.b = t2.b and t1.d = t2.d)
```

**Findings and Recommendations**

---

**SQL Repair Results: SQL_DIAG_1174506262358**

**Confirmation**

The recommended SQL Patch was implemented successfully. Verify results by executing SQL in SQL Worksheet.

**Status**

- **COMPLETED**

**Time Limit (seconds)**

- **1800**

**SQL ID**

- **9m7mvytcb4d14**

**Running Time (seconds)**

- **18**

**Recommendations**

<table>
<thead>
<tr>
<th>Select SQL Text</th>
<th>Parsing Schema</th>
<th>SQL ID</th>
<th>SQL Patch</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>delete from t1 where t1.a = 'a' and rowid &lt;&gt; (select max(rowid) from t2 where t1.a = t2.a and t1.b = t2.b and t1.d = t2.d)</code></td>
<td></td>
<td>9m7mvytcb4d14</td>
<td>✔️</td>
</tr>
</tbody>
</table>
SQL Performance Analyzer
SQL Performance Analyzer

• Measure and report on performance before and after a change! DBMS_SQLTUNE package.

Great for:
• Database Upgrades
• Application Upgrades
• Hardware Changes
• Database or Schema Changes
• Best for SQL Tuning – Especially Batches
Easy to run – SQL Focus (Test SGA settings):

- Capture SQL
- Transport SQL
- Create a Replay Task
- Set up the environment to Test
- Make any changes to Test (such as SGA/Optimizer)
- Compare before and after performance
- Tune the problems!
SQL Performance Analyzer

Click Here
SQL Performance Analyzer
Guided Workflow
SQL Performance Analyzer
Optimizer Upgrade (10g vs. 11g)
Real Application Testing!
Database workload capture and replay
Database workload capture and replay

- Used to capture **database workload** on one system and replay later on a different system. Useful to compare two different systems.
- Could rival LoadRunner in the future (may be more precise!)

**Brief Steps:**

- **Capture** workload on a database even from 10gR2
- **Restore** the database on a test system to the SCN when capture begins
- Perform **upgrade** and make changes to the test system as needed
- Preprocess the captured workload if it is not preprocessed
- Configure the test system for replay (I don’t do this here)
- **Replay workload** on the restored database (I don’t have this in this presentation, but will show some of the screens to do it)
- Great to test upgrade to 11g (Capture 10gR2 then test against...
11g Changes Supported
- Database Upgrades, Patches
- Schema, Parameters
- RAC nodes, Interconnect
- OS Platforms, OS Upgrades
- CPU, Memory
- Storage
- Etc.
Pre-Change (could be 10gR2) Production System

Post-Change Test System

Can use Snapshot Standby as test system
Database Replay
FYI Only – Download to view in detail

Real App Testing:
Database Replay
Capture Workload – FYI Only
Capture Workload – FYI Only

Database Instance: orcl

Confirmation

Job 'CAPTURE-ORCL-20071031041652' to capture the workload has been created successfully.

View Log

Once the capture is complete you will need to do the following prior to replaying the workload on a different system:
1. Optionally export the AWR data.
2. Restore the replay database on a test system to match the capture database at the start of the workload capture.
3. Make changes (such as perform an upgrade) to the test system as needed.
4. Copy the workload to the test system.
5. Preprocess the captured workload.

Database Replay

The Database Replay feature allows database workload to be captured on one system and replayed later on a different system. Replay a captured workload can be useful to compare two different systems.

<table>
<thead>
<tr>
<th>Task</th>
<th>Task Name</th>
<th>Description</th>
<th>Go to Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Capture Workload</td>
<td>Choose this option to capture workload on this database.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Preprocess Captured Workload</td>
<td>Preprocessing will prepare a captured workload for replay. This must be done once for every captured workload.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Replay Workload</td>
<td>Choose this option to replay a preprocessed workload on this database.</td>
<td></td>
</tr>
</tbody>
</table>

Active Capture and Replay

<table>
<thead>
<tr>
<th>Select Name</th>
<th>Type</th>
<th>Directory Object</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overview

The following are the typical steps to perform Database Replay:
1. Capture the workload on a database. (Task 1)
2. Optionally export the AWR data. (Task 1)
3. Restore the replay database on a test system to match the capture database at the start of the workload capture.
4. Make changes (such as perform an upgrade) to the test system as needed.
5. Copy the workload to the test system.
6. Preprocess the captured workload. (Task 2)
7. Configure the test system for the replay.
8. Replay the workload on the restored database. (Task 3)

Database | Setup | Preferences | Help | Logout

Copyright © 1996, 2007, Oracle. All rights reserved.
Oracle, J.D. Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager

61
Database workload – Preprocess
FYI Only
Database workload capture and replay
On NEW system (shortened) – FYI Only
Replay Options...

• Synchronized Replay
  – Exact Concurrency, commits & data divergence minimal

• Unsynchronized Replay
  – Not the same concurrency or commits
  – Data divergence can be large depending on load test performed

• Creates Report
  – Data Divergence
  – Error Divergence
  – Performance Divergence

![Elapsed Time Comparison Chart](chart.png)
Partitioning: (Briefly Only)

- Tables can be split into many pieces (10g).
- Only a subset of the data is queried
- All of the data COULD be queried
- Leads to enhanced performance of large tables
- Re-orgs & backups can be done on a partition level
- 4 quick examples follow (many many rules for each)
- WHAT’S NEW IN ORACLE 11G
CREATE TABLE DEPT
(DEPTNO NUMBER(2),
DEPT_NAME VARCHAR2(30))
PARTITION BY RANGE(DEPTNO)
(PARTITION D1 VALUES LESS THAN (10) TABLESPACE DEPT1,
PARTITION D2 VALUES LESS THAN (20) TABLESPACE DEPT2,
PARTITION D3 VALUES LESS THAN (MAXVALUE) TABLESPACE DEPT3);

INSERT INTO DEPT VALUES (1, 'DEPT 1');
INSERT INTO DEPT VALUES (7, 'DEPT 7');
INSERT INTO DEPT VALUES (10, 'DEPT 10');
INSERT INTO DEPT VALUES (15, 'DEPT 15');
INSERT INTO DEPT VALUES (22, 'DEPT 22');
create table cust_sales (
acct_no number(5),
cust_name char(30),
sale_day integer not null,
sale_mth integer not null,
sale_yr integer not null)
partition by range (sale_yr, sale_mth, sale_day)
(partition cust_sales_q1 values less than (1998, 04, 01) tablespace users1,
partition cust_sales_q2 values less than (1998, 07, 01) tablespace users2,
partition cust_sales_q3 values less than (1998, 10, 01) tablespace users3,
partition cust_sales_q4 values less than (1999, 01, 01) tablespace users4,
partition cust_sales_qx values less than (maxvalue, maxvalue, maxvalue) tablespace users4);
create table cust_sales_hash ( 
acct_no number(5),
cust_name char(30),
sale_day integer not null,
sale_mth integer not null,
sale_yr integer not null)
partition by hash (acct_no)
partitions 4
store in (users1, users2, users3, users4);
CREATE TABLE test5 (data_item INTEGER, length_of_item INTEGER,
    storage_type VARCHAR(30), owning_dept NUMBER,
    storage_date DATE) PARTITION BY RANGE (storage_date) SUBPARTITION BY HASH(data_item) SUBPARTITIONS 4
STORE IN (data_tbs1, data_tbs2, data_tbs3, data_tbs4)
    (PARTITION q1_1999 VALUES LESS THAN (TO_DATE('01-apr-1999', 'dd-mon-yyyy')),
     PARTITION q2_1999 VALUES LESS THAN (TO_DATE('01-jul-1999', 'dd-mon-yyyy')),
     PARTITION q3_1999 VALUES LESS THAN (TO_DATE('01-oct-1999', 'dd-mon-yyyy'))
     (SUBPARTITION q3_1999_s1 TABLESPACE data_tbs1,
      SUBPARTITION q3_1999_s2 TABLESPACE data_tbs2),
     PARTITION q4_1999 VALUES LESS THAN (TO_DATE('01-jan-2000', 'dd-mon-yyyy'))
     SUBPARTITIONS 8
STORE IN (q4_tbs1, q4_tbs2, q4_tbs3, q4_tbs4,
    q4_tbs5, q4_tbs6, q4_tbs7, q4_tbs8),
 PARTITION q1_2000 VALUES LESS THAN (TO_DATE('01-apr-2000', 'dd-mon-yyyy')));
create table dept_part
(deptno number(2),
dname varchar2(14),
loc varchar2(13))
partition by list (dname)
(partition d1_east values ('BOSTON', 'NEW YORK'),
partition d2_west values ('SAN FRANCISCO', 'LOS ANGELES'),
partition d3_south values ('ATLANTA', 'DALLAS'),
partition d4_north values ('CHICAGO', 'DETROIT'));

Table created.
Interval Partitioning – 11g

- This is a helpful addition to range partitioning where Oracle automatically creates a partition when the inserted value exceeds all other partition ranges. 11g also has Ref & Virtual Column Partitioning (not covered here).

There are the following restrictions:
- You can only specify one partitioning key column, and it must be of NUMBER or DATE type.
- Interval partitioning is NOT supported for index-organized tables.
- You can NOT create a domain index on an interval-partitioned table.
CREATE TABLE DEPT_new
(DEPTNO NUMBER(2),
DEPT_NAME VARCHAR2(30))
PARTITION BY RANGE(DEPTNO)
(PARTITION D1 VALUES LESS THAN (10),
PARTITION D2 VALUES LESS THAN (20),
PARTITION D3 VALUES LESS THAN (30));

Table created.

SQL> insert into dept_new values(40, 'test2');
insert into dept_new values(40, 'test2')
*
ERROR at line 1:
ORA-14400: inserted partition key does not map to any partition
### Interval Partitioning – 11g

```sql
select segment_name, partition_name
from dba_segments
where segment_name = 'DEPT_NEW';
```

<table>
<thead>
<tr>
<th>SEGMENT_NAME</th>
<th>PARTITION_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT_NEW</td>
<td>D1</td>
</tr>
<tr>
<td>DEPT_NEW</td>
<td>D2</td>
</tr>
<tr>
<td>DEPT_NEW</td>
<td>D3</td>
</tr>
</tbody>
</table>
CREATE TABLE DEPT_NEW2
(DEPTNO NUMBER(2),
DEPT_NAME VARCHAR2(30))
PARTITION BY RANGE(DEPTNO)
INTERVAL(10)

(PARTITION D1 VALUES   LESS THAN (10),
PARTITION D2 VALUES     LESS THAN (20),
PARTITION D3 VALUES     LESS THAN (30))

Table created.

SQL > insert into dept_new2 values(40, 'test2');
1 row created.
```sql
insert into dept_new2 values(40,null);
insert into dept_new2 values(50,null);
insert into dept_new2 values(99,null);

select segment_name, partition_name
from dba_segments
where segment_name = 'DEPT_NEW2'
```

<table>
<thead>
<tr>
<th>SEGMENT_NAME</th>
<th>PARTITION_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT_NEW2</td>
<td>D1</td>
</tr>
<tr>
<td>DEPT_NEW2</td>
<td>D2</td>
</tr>
<tr>
<td>DEPT_NEW2</td>
<td>D3</td>
</tr>
<tr>
<td>DEPT_NEW2</td>
<td>SYS_P41</td>
</tr>
<tr>
<td>DEPT_NEW2</td>
<td>SYS_P42</td>
</tr>
<tr>
<td>DEPT_NEW2</td>
<td>SYS_P43</td>
</tr>
</tbody>
</table>
Partition Compression

• You can now COMPRESS individual partitions
• Compression as high as 3.5 to 1 is possible
• Compressed Tables now support
  – DML Statements
  – Add and Drop Column
  – Partition level COMPRESS or NOCOMPRESS
• ALTER TABLE... COMPRESS (old compress)
• ALTER TABLE... NOCOMPRESS
• Table compression now supported for OLTP
• New Advanced Compression Option (chargeable):
  – CREATE TABLE t1 COMPRESS FOR ALL OPERATIONS
CREATE TABLE DEPT_new3
  (DEPTNO NUMBER(2),
   DEPT_NAME VARCHAR2(30))
  COMPRESS
PARTITION BY RANGE(DEPTNO)
interval(10)
  (PARTITION D1 VALUES LESS THAN (10),
   PARTITION D2 VALUES LESS THAN (20)
   NOCOMPRESS,
   PARTITION D3 VALUES LESS THAN (30))

Table created.
Large-Scale Data Warehouses

Feature Usage

* Oracle Survey
Nice DBA Feature

Oracle Secure Files
FYI Only
Oracle SecureFiles
High-Performance Large Objects

• High-performance transactional access to large object data
  – RFID, DICOM medical, CAD, images, 3D spacial
  – low-latency, high throughput, concurrent access
  – space-optimized storage

• Protect your valuable data ... Keep large objects in the database!
  – transactions
  – transparent encryption
  – compression and de-duplication
  – database-quality security, reliability, and scalability

• Better security, single view and management of data
• Superset of LOB interfaces – easy migration
Oracle Secure Files
Better Performance than LOBs...

Read Performance

Write Performance

Adding Files using New Disk Space – 2x faster than LOBs
Adding Files using Deleted Space – 22x faster than LOBs
PL/SQL Reads – 6x Faster than LOBs

Your mileage will vary....
Nice Developer Tools/Improvements

Result Cache
Invisible Indexes
PL/SQL Expressions
The Result Cache

• **Function Results** of queries and query fragments can be cached in memory for **future executions**.
  - Choose calculations that frequently run
  - Choose data that does NOT frequently change

• **RESULT_CACHE & RELIES_ON** clauses

• **Takes its memory from the Shared Pool**
  - Set with **RESULT_CACHE_SIZE**
  - **RESULT_CACHE_MODE=force** (manual) – no auto

• **DBMS_RESULT_CACHE.FLUSH** to clear

• Is **NOT** passed between RAC/Grid nodes

• Check the docs for other Restrictions & Rules!!
Result Cache Performance
Example Query (1M Row Test)

```sql
select *
from (select *
      from (select t.country_name, t.city_name,
              sum(t.salary) a_sum, max(t.salary) a_max
      from emps t
      group by t.country_name, t.city_name)
      order by a_max desc)
where rownum < 2;
```
Step 1 - In Session 1
Executed query with the RESULT_CACHE hint and it returned an elapsed time of 3.18 seconds (cache it).

Step 2 - In Session 2
Executed query without the RESULT_CACHE hint, but with RESULT_CACHE_MODE=force and it returned an elapsed time of 0.86 seconds (cached!!)
The Result Cache – Autotrace

select count(*) from emp; (note the result cache)

COUNT(*)
-----------
14

Execution Plan
-----------------------------------------------
Plan hash value: 2937609675

| Id  | Operation       | Name                          | Rows | Cost (%CPU) | Time     |
-----------------------------------------------
| 0   | SELECT STATEMENT|                               |     1|     1  (0) | 00:00:01 |
| 1   | RESULT CACHE    | 4ntcq5q3m4ayb26wqthu7pbn17    |     1|            |          |
| 2   | SORT AGGREGATE  |                               |     1|            |          |
| 3   | INDEX FULL SCAN | PK_EMP                        |    14|     1  (0) | 00:00:01 |
The Invisible Index

- Set an index to VISIBLE or INVISIBLE
  - ALTER INDEX idx INVISIBLE;
  - ALTER INDEX idx VISIBLE;
  - CREATE INDEX... INVISIBLE;
- Great to turn off indexes for a while when you think they’re not being used, but BEFORE you drop them.
- Can use NO_INDEX (to override visibility).
- The index IS MAINTAINED during DML
- Great for testing!
Allow Sequences in PL/SQL Expressions

• In Previous Versions needed to retrieve the value of a sequence (CURRVAL / NEXTVAL) by invoking a cursor (explicit or implicit).

In 11g:
• **No cursor is needed** so the code is more efficient.
• For big jobs – Saves MANY cursors
Allow Sequences in PL/SQL Expressions

**OLD Way**

DECLARE
    V_NEW_VAL  NUMBER;
BEGIN
    SELECT MY_SEQ.NEXTVAL INTO V_NEW_VAL
    FROM DUAL;
END;

**NEW Way**

DECLARE
    V_NEW_VAL  NUMBER;
BEGIN
    V_NEW_VAL := MY_SEQ.NEXTVAL;
END;
Create & Rebuild Index Online
You can create/rebuild indexes even when doing DML on the base table, but it’s better to do during low DML activity.

Prior to Oracle 11g, this required an exclusive lock at the beginning and end of the rebuild. This lock could cause DML delays and performance spike. This lock is no longer required for this operation.

Rebuild is faster than a DROP and CREATE

Basic Syntax:

CREATE INDEX index_name ON table (col1,...) ONLINE;
Index created.

ALTER INDEX index_name REBUILD ONLINE;
Index altered.
Rebuild Index or Coalesce (FYI)
Coalesce Example from Oracle Doc.

Before ALTER INDEX vmoore COALESCE;

After ALTER INDEX vmoore COALESCE;
Rebuild Index or Coalesce

**Rebuild:**
- Quickly move index to another tablespace
- Requires more disk space
- Creates new index tree and shrinks heights
- Change storage.tblspc w/o dropping

**Coalesce**
- Can’t move to another tablespace
- Requires much less space than rebuild
- Coalesces leaf blocks that are in the same branch
- Quickly frees index leaf blocks for use
Optimizer Statistics & Other Optimizer Advances

Special Thanks: Maria Colgan, Penny Avril & Debbie Migliore
Improved SPEED and Quality
Gathering Stats – AUTO-SAMPLING

• Manually gather stats: Impossible to find sample size that works for ALL tables - need COMPUTE

• Especially hard to find a good sample size when the data distribution is very skewed.

• NEW Auto-sampling: “Discovers” the best sample size for every table in your system for you.
  
  – Get the **Quality of a COMPUTE** with **SPEED of a SAMPLE**
  
  – Oracle’ goal is to **OBSOLETE** the need and use of
• In 10g, if you gather stats on one partition after a bulk load it causes a full scan of all partitions to gather global table statistics which is extremely time consuming.

• In 10g, you have to manually copy statistics to new partition.

• In 11g Gather stats for TOUCHED PARTITIONS only!

• Table stats are refreshed WITHOUT scanning the un-touched partitions.
Manage New Statistics
Gather Stats but make PENDING

- Currently DBAs are scared to gather stats on a table that is changing for fear of unpredictable execution plans.
- You have to ‘FREEZE’ critical plans or stats.
- In 11g, gather stats and save as PENDING.
- Verify the new stats won’t adversely affect things by checking them with a single user using an alter session or try them out on a different system.
- When everything looks good – then, PUBLISH them for all to use!
Gather Stats but make them PENDING

```
select dbms_stats.get_prefs('PUBLISH', 'SH', 'CUST') publish from dual;
PUBLISH
-------------------
TRUE
exec dbms_stats.set_table_prefs('SH', 'CUST', 'PUBLISH', 'false');
PL/SQL procedure successfully completed.

select dbms_stats.get_prefs('PUBLISH', 'SH', 'CUST') publish from dual;
PUBLISH
```
Gather Stats but make them PENDING

```sql
select table_name, last_analyzed analyze_time, num_rows, blocks, avg_row_len
from user_tables
where table_name = 'CUST';

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>ANALYZE_T</th>
<th>NUM_ROWS</th>
<th>BLOCKS</th>
<th>AVG_ROW_LEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

execute dbms_stats.gather_table_stats('SH', 'CUST');

PL/SQL procedure successfully completed.

```sql
select table_name, last_analyzed analyze_time, num_rows, blocks, avg_row_len
from user_tables
where table_name = 'CUST';

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>ANALYZE_T</th>
<th>NUM_ROWS</th>
<th>BLOCKS</th>
<th>AVG_ROW_LEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
alter session set optimizer_use_pending_statistics = true;
(Then run your query – If ready/better – publish the new stats)

dbmstats.publish_pending_stats('SH', 'CUST');
PL/SQL procedure successfully completed.

select table_name, last_analyzed, analyze_time, num_rows, blocks,
    avg_row_len
from user_tables
where table_name = 'CUST';

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>ANALYZE_T</th>
<th>NUM_ROWS</th>
<th>BLOCKS</th>
<th>AVG_ROW_LEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST</td>
<td>13-OCT-07</td>
<td>55500</td>
<td>1485</td>
<td>180</td>
</tr>
</tbody>
</table>

dbmstats.delete_table_stats('SH', 'CUST'); < to
Extended Optimizer Statistics: New Multi-Column Statistics

- Corporate data often has correlations between different columns of a table. For example:
  - A job title is correlated to the salary.
  - The season affects the sold amounts of items such as swim suits sell more in the summer and snow shoes sell more in the winter.
  - The make of a car and color are often used together but are not really correlated well so the filter doesn’t reduce the result set.

- Optimizer has to estimate the correct cardinality
  - Will the additional column condition reduce the result set or not? Should it be used.

- Oracle calculates correlated statistics so the optimizer will make great decisions. Single column statistics and histograms are not enough!
Example

SELECT make, price, color
FROM cars_dot_com
WHERE make = 'CORVETTE';

<table>
<thead>
<tr>
<th>Make</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORVETTE</td>
<td>40,000</td>
<td>RED</td>
</tr>
<tr>
<td>CORVETTE</td>
<td>60,000</td>
<td>BLACK</td>
</tr>
<tr>
<td>CORVETTE</td>
<td>50,000</td>
<td>SILVER</td>
</tr>
<tr>
<td>CADILLAC</td>
<td>90,000</td>
<td>RED</td>
</tr>
<tr>
<td>JEEP</td>
<td>35,000</td>
<td>BLACK</td>
</tr>
<tr>
<td>JEEP</td>
<td>45,000</td>
<td>SLIVER</td>
</tr>
</tbody>
</table>

- Three records selected.
- Single column statistics are accurate
Example, cont.

SELECT make, price, color
FROM cars_dot_com
WHERE make = 'CORVETTE'
AND COLOR = 'RED';

<table>
<thead>
<tr>
<th>Make</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORVETTE</td>
<td>40,000</td>
<td>RED</td>
</tr>
<tr>
<td>CORVETTE</td>
<td>60,000</td>
<td>BLACK</td>
</tr>
<tr>
<td>CORVETTE</td>
<td>50,000</td>
<td>SILVER</td>
</tr>
<tr>
<td>CADILLAC</td>
<td>90,000</td>
<td>RED</td>
</tr>
<tr>
<td>JEEP</td>
<td>35,000</td>
<td>BLACK</td>
</tr>
<tr>
<td>JEEP</td>
<td>45,000</td>
<td>SILVER</td>
</tr>
</tbody>
</table>

- One record selected.
- No correlated columns
- Additional predicate reduces result set
- Single column statistics are STILL sufficient
Example, cont.

- Three records selected.
  - Correlated columns
  - Additional predicate **has no effect**
  - Single column statistics are **NOT sufficient**
  - Must use ‘=‘ and not < or >

```
SELECT make, price, color
FROM   cars_dot_com
WHERE  make = 'CORVETTE'
AND    PRICE = 50000;
```
Manage New Statistics – FYI Only
EXTENDED Statistic Group

- Provides a way to **collect stats on a group of columns**
- **Full integration into existing statistics framework**
  - Automatically maintained with column statistics
  - Instantaneous and transparent benefit for any application
- **Accurate cardinalities for inter-related columns**
  - Multiple predicates on the same table are estimated correctly
select column_name, num_distinct, histogram
from user_tab_col_statistics where table_name = 'CUSTOMERS';

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>NUM_DISTINCT</th>
<th>HISTOGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_VALID</td>
<td>2</td>
<td>NONE</td>
</tr>
<tr>
<td>COUNTRY_ID</td>
<td>19</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>CUST_STATE_PROVINCE</td>
<td>145</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_CITY_ID</td>
<td>620</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_CITY</td>
<td>620</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_LAST_NAME</td>
<td>908</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_FIRST_NAME</td>
<td>1300</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_ID</td>
<td>55500</td>
<td>NONE</td>
</tr>
</tbody>
</table>

... 23 rows selected.
Now let's create the extended statistics group & re-gather statistics on the CUSTOMER table (query user_tab_col_statistics to see new column):

```sql
select dbms_stats.create_extended_stats('SH','CUSTOMERS', '(country_id, cust_state_province)') from dual;
```

```
DBMS_STATS.CREATE_EXTENDED_STATS('SH','CUSTOMERS','(CO-----------------------------
SYS_STUJGVLRVH5USVDU$XNV4_IR#4
exec dbms_stats.gather_table_stats('SH','CUSTOMERS', method_opt => 'for all columns size skewonly');
PL/SQL procedure successfully completed.
```
Manage New Statistics – FYI Only
Now there are Extended Statistics

```sql
select column_name, num_distinct, histogram
from user_tab_col_statistics where table_name = 'CUSTOMERS';
```

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>NUM_DISTINCT</th>
<th>HISTOGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_STUJGVLRVH5USVDU$XNV4_IR#4</td>
<td>145</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>CUST_VALID</td>
<td>2</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>COUNTRY_ID</td>
<td>19</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>CUST_STATE_PROVINCE</td>
<td>145</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>CUST_CITY_ID</td>
<td>620</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_CITY</td>
<td>620</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_LAST_NAME</td>
<td>908</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_FIRST_NAME</td>
<td>1300</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_ID</td>
<td>55500</td>
<td>HEIGHT BALANCED</td>
</tr>
</tbody>
</table>

... 24 rows selected.
Manage New Statistics – FYI Only

DROP Extended Statistics

exec dbms_stats.drop_extended_stats('SH', 'CUSTOMERS', '(country_id, cust_state_province)');

PL/SQL procedure successfully completed.

select column_name, num_distinct, histogram
from user_tab_col_statistics where table_name = 'CUSTOMERS';

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>NUM_DISTINCT</th>
<th>HISTOGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_VALID</td>
<td>2</td>
<td>NONE</td>
</tr>
<tr>
<td>COUNTRY_ID</td>
<td>19</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>CUST_STATE_PROVINCE</td>
<td>145</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_CITY_ID</td>
<td>620</td>
<td>HEIGHT BALANCED</td>
</tr>
<tr>
<td>CUST_CITY</td>
<td>620</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_LAST_NAME</td>
<td>908</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_FIRST_NAME</td>
<td>1300</td>
<td>NONE</td>
</tr>
<tr>
<td>CUST_ID</td>
<td>55500</td>
<td>NONE</td>
</tr>
</tbody>
</table>

... 23 rows selected.
Enterprise Manager for the Grid
Enterprise Manager: Back in Time!
Performance Manager: Back in Time!
113

Grid Control – 10gR2;

Many more Options!
Enterprise Manager 11g
Just a bit of changes...

Specific Database Instance
We have 5 ADDM Findings
Check them Here
Security Enhancements
Security Enhancements

• **11g is more restrictive**
  - Password lock time (1), password grace time (7) and password life time (180) all more restrictive; Failed login attempts stays the same (10).
  - Passwords will be case sensitive now! (on by default)
  - Enhanced hashing algorithm for passwords / DES still available.
  - **Strong passwords** (set via password complexity verification in EM or SQL):
    • Minimum 8 characters
    • At least one letter and one digit
    • Not servername or servername(1-100)
    • Not a common password (i.e. welcome1)
    • Must differ from previous password by 3 characters minimum
Security Enhancements

AUDIT_TRAIL = DB (default)

- Audit Trail is ON by default (was off in 10g),
- AUDIT_TRAIL = DB is now the default.
- Things that will be audited by default include:
  - CREATE USER, CREATE SESSION, CREATE ANY TABLE, CREATE ANY PROCEDURE, CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE ANY LIBRARY, CREATE PUBLIC DB LINK
  - ALTER USER, ALTER ANY TABLE, ALTER ANY PROCEDURE, ALTER PROFILE, ALTER DATABASE, ALTER SYSTEM, AUDIT SYSTEM
  - DROP USER, DROP ANY TABLE, DROP ANY PROCEDURE, DROP PROFILE
  - GRANT ANY PRIVILEGE, GRANT ANY OBJECT PRIVILEGE
  - EXEMPT ACCESS POLICY
  - AUDIT SYSTEM
- Cost of Auditing improved to be 1-2% cost on TPCC benchmark.
Oracle Database Security
Built over MANY years...

Oracle Audit Vault
Oracle Database Vault
DB Security Evaluation #19
Transparent Data Encryption
EM Configuration Scanning
Fine Grained Auditing (9i)
Secure application roles
Client Identifier / Identity propagation
Proxy authentication
Enterprise User Security
Global roles
Virtual Private Database (8i)
Database Encryption API
Strong authentication (PKI, Kerberos, RADIUS)
Native Network Encryption (Oracle7)

1977 Government customer

2007
The Future: 8 Exabytes
Look what fits in one 10g Database!

2K – A typewritten page
5M – The complete works of Shakespeare
10M – One minute of high fidelity sound
2T – Information generated on YouTube in one day
10T – 530,000,000 miles of bookshelves at the Library of Congress
20P – All hard-disk drives in 1995 (or your database in 2010)
700P – Data of 700,000 companies with Revenues less than $200M
1E – Combined Fortune 1000 company databases (average 1P each)
1E – Next 9000 world company databases (average 100T each)
8E – Capacity of ONE Oracle10g Database (CURRENT)
12E to 16E – Info generated before 1999 (memory resident in 64-bit)
16E – Addressable memory with 64-bit (CURRENT)
161E – New information in 2006 (mostly images not stored in DB)
1Z – 1000E (Zettabyte - Grains of sand on beaches -125 Oracle DBs)
100TY – 100T-Yottabytes – Addressable memory 128-bit (FUTURE)
8 Exabytes:
Look what fits in one 10g Database!

- All databases of the largest 1,000,000 companies in the world (3E).
  or
- All Information generated in the world in 1999 (2E)
  or
- All Information generated in the world in 2003 (5E)
  or
- All Email generated in the world in 2006 (6E)
  or
- 1 Mount Everest filled with Documents (approx.)
Compelling Technology Statistics!

Radio: 10 years to reach 50M users
TV: 15 years to reach 50M users
Cable: 20 years to reach 50M users
Internet: 25 years to reach 50M users
Wireless: 30 years to reach 50M users

Years to Reach 50M Users

Radio
TV
Cable
Internet
Wireless

Years
0 5 10 15 20 25 30 35 40
Friedman’s 6 Dimensions of Understanding Globalization*

- Politics (Merging)
- Culture (Still disparate)
- Technology (Merging/Merged)
- Finance (Merging/Merged)
- National security (Disparate)
- Ecology (Merging)

* Sited from Mark Hasson, PSU, Global Pricing and International Marketing.
<table>
<thead>
<tr>
<th>Version</th>
<th>V$ Views</th>
<th>X$ Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>23</td>
<td>? (35)</td>
</tr>
<tr>
<td>7</td>
<td>72</td>
<td>126</td>
</tr>
<tr>
<td>8.0</td>
<td>132</td>
<td>200</td>
</tr>
<tr>
<td>8.1</td>
<td>185</td>
<td>271</td>
</tr>
<tr>
<td>9.0</td>
<td>227</td>
<td>352</td>
</tr>
<tr>
<td>9.2</td>
<td>259</td>
<td>394</td>
</tr>
<tr>
<td>10.1.0.2</td>
<td>340 (+31%)</td>
<td>543 (+38%)</td>
</tr>
<tr>
<td>10.2.0.1</td>
<td>396</td>
<td>613</td>
</tr>
<tr>
<td>11.1.0.6</td>
<td>484 (+22%)</td>
<td>798 (+30%)</td>
</tr>
</tbody>
</table>
Summary

- Start Me Up – Using Memory Target
- The Result Cache
- Invisible Indexes & Online Index Rebuilds
- Other Nice Developer Tools
- ADDM Enhancements
- SQL Plan Management & SQL Plan Baselines
- SQL Query Repair Advisor
- SQL Performance Analyzer
- Real Application Testing (Capture & Replay)
- Interval Partitioning & Partition Compression
- DBA Tools and DBMS_STATS Enhancements
- Grid Control & EM
- Security Enhancements & the Future Sizes
- Summary
For More Information

- www.tusc.com
- *Oracle9i Performance Tuning Tips & Techniques; Richard J. Niemiec; Oracle Press* (May 2003)
- *Oracle 10g Tuning (June 11, 2007)*

“If you are going through hell, keep going” - Churchill
更多信息

- www.tusc.com
- Oracle9i Performance Tuning Tips & Techniques; Richard J. Niemiec; Oracle Press (May 2003)
- Oracle 10g Tuning (June 11, 2007)

“成功只访问那些没空追求它的人。”

- Henry David Thoreau
“Life is not measured by the number of breaths we take, but by the moments that take our breath away.”

-Dr. Moorehead
Copyright Information

- Neither TUSC, Rolta, Oracle nor the author guarantee this document to be error-free. Please provide comments/questions to rich@tusc.com.
- TUSC © 2009. This document cannot be reproduced without expressed written consent from an officer of TUSC, but the audience of this presentation may reproduce or copy this for conference & website use.

Contact Information

Rich Niemiec: rich@tusc.com

www.tusc.com
Save the Date!

May 3-7, 2009

Orange County Convention Center West; Orlando, Florida
“Success usually comes to those that are too busy to be looking for it.”

- Henry David Thoreau
References

- *Oracle10g Performance Tuning Tips & Techniques; Richard J. Niemiec; Oracle Press*
- Database Secure Configuration Initiative: Enhancements with Oracle Database 11g, www.oracle.com
- All Oracle11g Documentation from Oracle Beta Site
- Introduction to Oracle Database 11g, Ken Jacobs
- Oracle Database 11g New Features, Linda Smith
- New Optimizer Features in 11g, Maria Colgan
- Thanks Dan M., Bob T., Brad, Joe, Heidi, Mike K., Debbie, Maria, Linda

All companies and product names are trademarks or registered trademarks of the respective owners.

Dedicated to the memory of Stan Yellott, Mark Beaton, Ray Mansfield, Lex De Haan, Elaine DeMeo and Jim Gray.
TUSC Services

• Oracle Technical Solutions
  – Full-Life Cycle Development Projects
  – Enterprise Architecture
  – Database Services

• Oracle Application Solutions
  – Oracle Applications & EMP Implementations/Upgrades
  – Oracle Applications & Hyperion Tuning

• Managed Services
  – 24x7x365 Remote Monitoring & Management
  – Functional & Technical Support

• Training & Mentoring

• Oracle Authorized Reseller
“The Oracle Experts” since 1988
- Editors Choice – Consultant of the Year 2002 & 2004
- Authorship, User Groups and Various Awards
- One of the first 6 Oracle Masters in the World
- Certified Advantage Partner
Rich’s Overview
(rich@tusc.com)

- President of TUSC – A Rolta Company:
  - Inc. 500 Company (Fastest Growing 500 Private Companies)
  - 7 Offices in the United States (U.S.); Based in Chicago
  - Oracle Advantage Partner in Tech & Applications

- Author (3 Oracle Best Sellers):
  - Oracle Performing Tips & Techniques (Covers Oracle7 & 8i)
  - Oracle9i Performance Tips & Techniques
  - Oracle Database 10g Performance Tips & Techniques

- Former President of the International Oracle Users Group
- Current President of the Midwest Oracle Users Group
- Chicago Entrepreneur Hall of Fame - 1998
- Entrepreneur of the Year & National Hall of Fame - 2001
- MOUG Top Speaker Twelve Times
- National Trio Achiever award - 2006
- Oracle Certified Master & Oracle Ace Director
- Purdue Outstanding Electrical & Computer and Engineer - 2007