

THE BEST ORACLE DATABASE 11G NEW FEATURES

By Rich Niemic, Rolta TUSC

ABSTRACT

With Oracle 11g Oracle takes database technology to a new level! This paper will look at a few of the new features you can use with this new version. BUT, I warn you, there are so many great new features that I could never do the product justice with such a short paper. I will try to cover these new features with simple examples or screen shots, but you'll need to work with the new version yourself to get good at the new version. This only will serve as an introduction to several of the new features. See the Oracle Database 11g documentation for a detailed look at this product. Some of the new features that I will cover here include:

- Starting the new version
- The MEMORY_TARGET initialization parameter
- ADDM Enhancements
- SQL Plan Management and SQL Baselines
- SQL Repair Advisory
- SQL Performance Analyzer
- Real Application Testing
- Interval Partitioning
- Oracle Secure Files
- The Result Cache
- Invisible Indexes
- Nice Developer Feature – Sequences in PL/SQL Expressions
- DBMS_STATS & Optimizer Enhancements
- Multi-Column Statistics
- Grid Control
- Security Enhancements

STARTING THE NEW VERSION OF 11G – LIFE IS GOOD!

After installing Oracle 11g, we always want that happy message “Database Opened.” Here is the version of Oracle that I was using for my testing on Linux:

```
$ sqlplus ***/***
```

```
SQL*Plus: Release 11.1.0.6.0 - Production on Tue Oct 30 11:21:04 2007
```

```
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.
```

You could also check Enterprise Manager to check to see that things are up and running. The screenshot in Figure 1 shows the status of the 11g database (currently CPU bound after I put a load on the system for testing purposes):

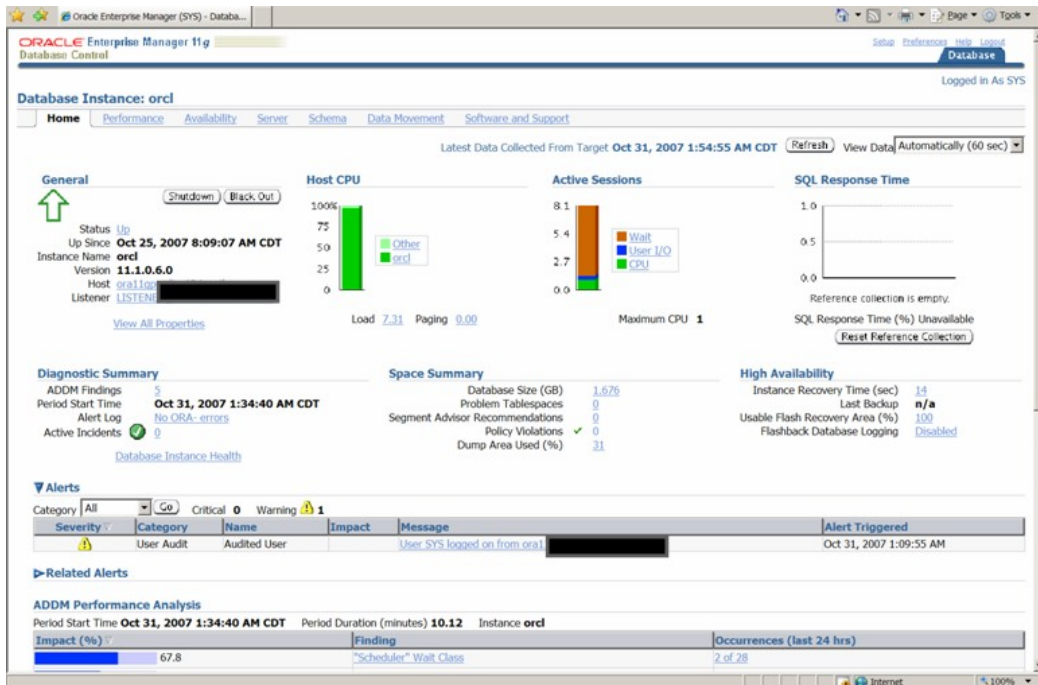


Figure 1: Viewing the 11g Database status via Enterprise Manager 11g

ORACLE NOW RUNS WITH A SINGLE INITIALIZATION PARAMETER!

DBA's of non-Oracle databases often complain that there is too many parameters to tune with Oracle (too much functionality for them). Nothing could be further from reality. Oracle does give you many options so that you can tailor initialization parameters EXACTLY to match what you're trying to accomplish, but most DBA's only set a very small percentage of those options. We could see that in 9i, Oracle started moving toward simpler settings with the SGA_MAX_SIZE parameter and changing using DB_BLOCK_BUFFERS to DB_CACHE_SIZE. In 10g, they created a new parameter called SGA_TARGET (a recommended parameter to set for Oracle Applications Best Practices). This allowed for a single setting for all of the SGA including the DB_CACHE_SIZE and SHARED_POOL_SIZE (although I would still set these – your setting is used as minimum used values for the parameters you set). In 11g, Oracle now includes combining the PGA and SGA into a single parameter: MEMORY_TARGET. You still should set minimums for DB_CACHE_SIZE (memory for data blocks), SHARED_POOL_SIZE (memory for statements) and PGA_AGGREGATE_TARGET (memory for sorting purposes and some joins). The SGA and PGA are now in Memory Target. The Oracle documentation gives a great diagram of this as shown in Figure 2.

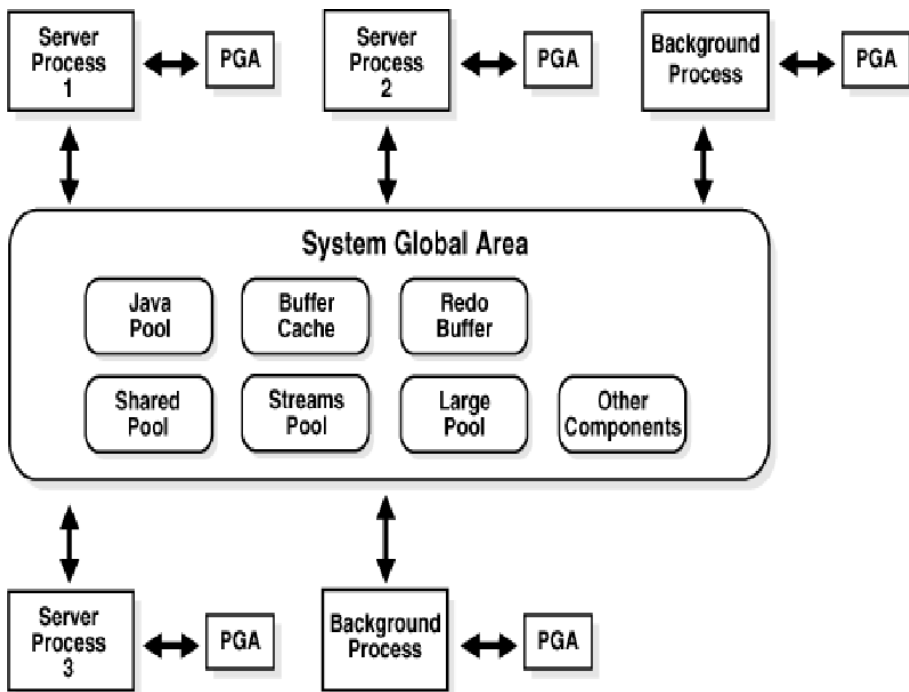


Figure 2: SGA & PGA combined in initialization parameter: MEMORY_TARGET

You can also view memory allocation in Enterprise Manager as displayed in Figure 3 & 4.

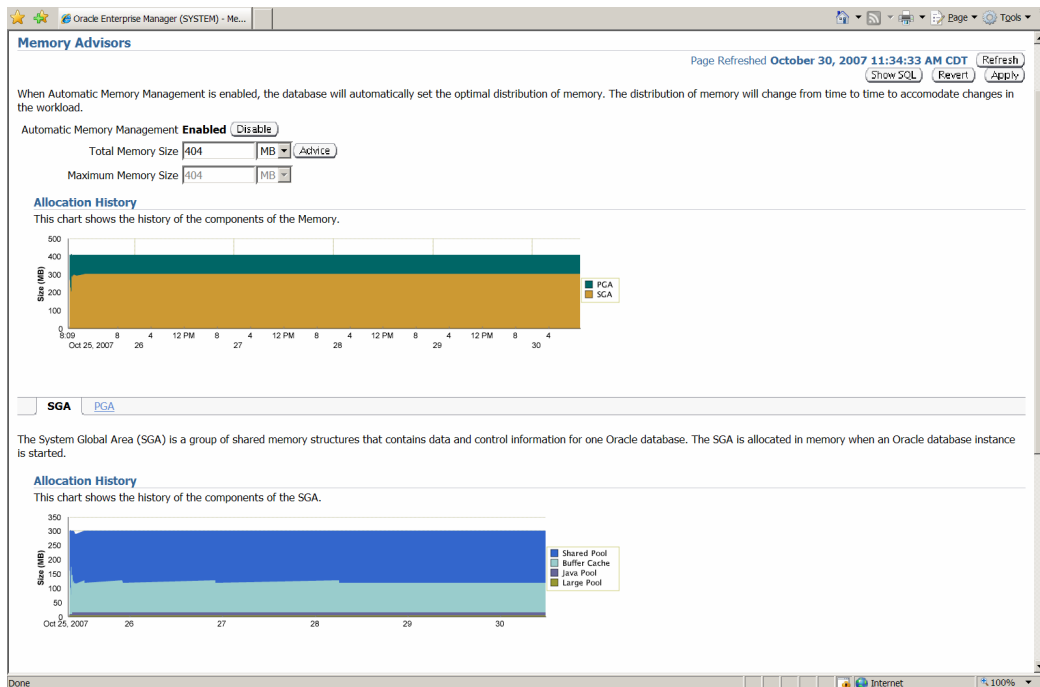


Figure 3: SGA & PGA view in Enterprise Manager

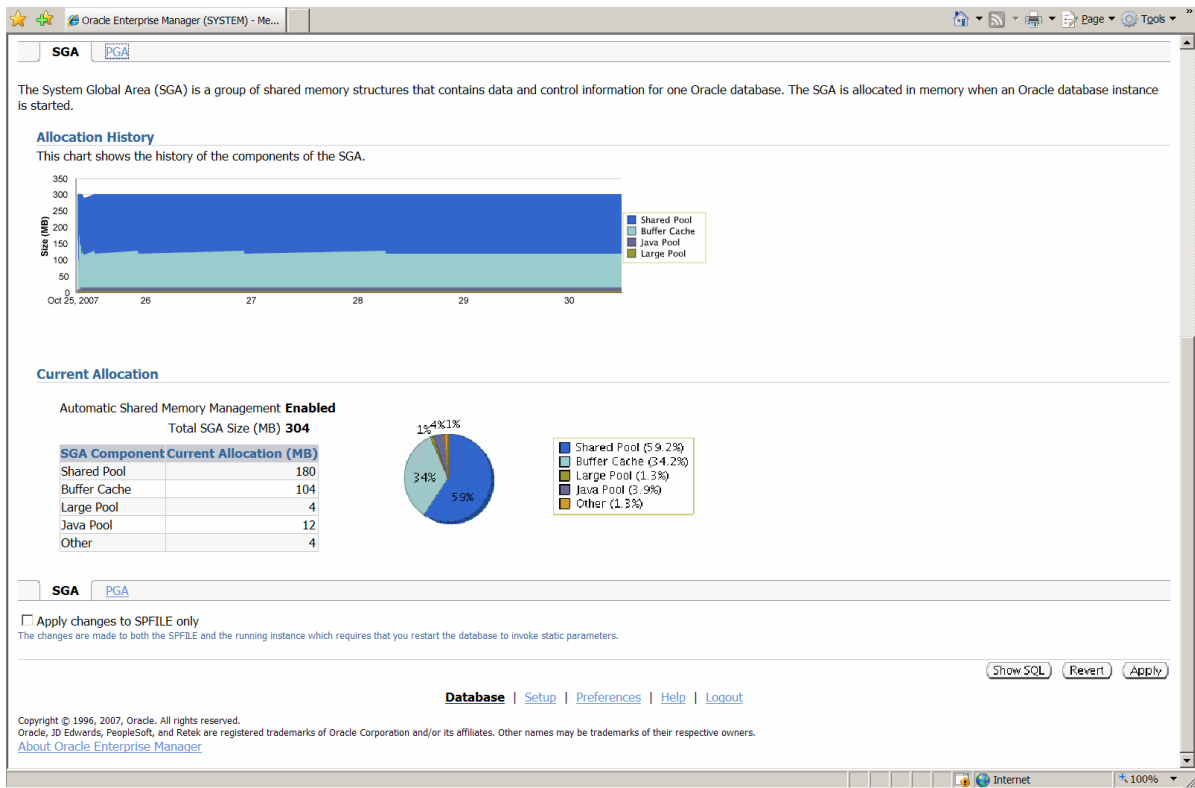


Figure 4: SGA & PGA view in Enterprise Manager (lower part of page)

ADDM IN 11G TAKES SYSTEM MANAGEMENT TO THE NEXT LEVEL

ADDM is Oracle's Automatic Database Diagnostic Monitor. If you've used it in Oracle 10g, it's even better in Oracle 11g. ADDM runs every hour (out of the box) after stats are collected to give you recommendations for your system. Some of the enhancements to ADDM in 11g include:

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

In Figures 5, you can see that the information gathered by ADDM is displayed in the main screen (bottom) of Enterprise Manager and Figure 6 shows recommendations.

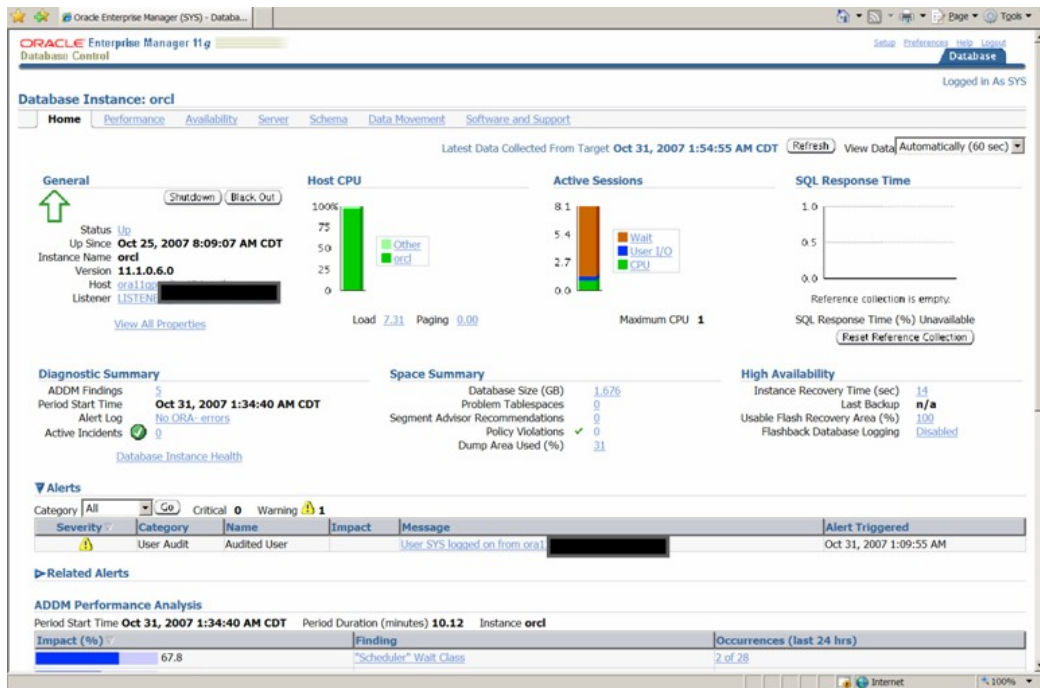


Figure 5: Note the ADDM Performance Analysis and recommendations at the bottom of Enterprise Manager main Database instance screen.

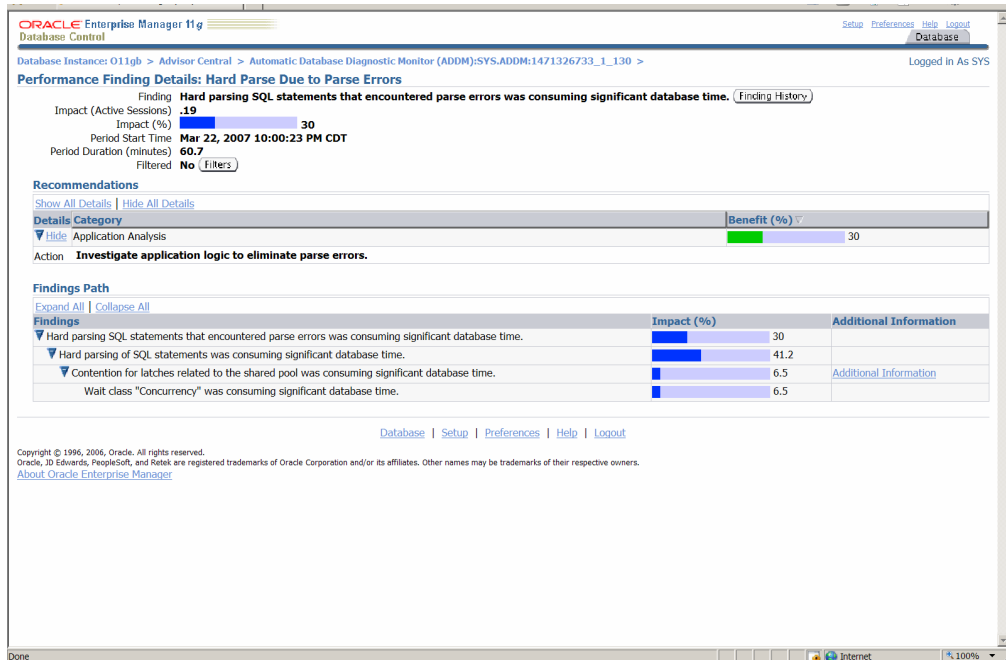


Figure 6: ADDM findings and solutions after clicking on a specific recommendation

Something new in 11g is the ability to run ADDM right now. Figure 7 shows a spike in performance when looking at the performance of the O11gb instance. You can see the button “Run ADDM Now” (button is just above the spike itself in this case) to immediately get recommendations for improving this performance spike. Figure 8 shows that the main cause of the spike was a CPU related.

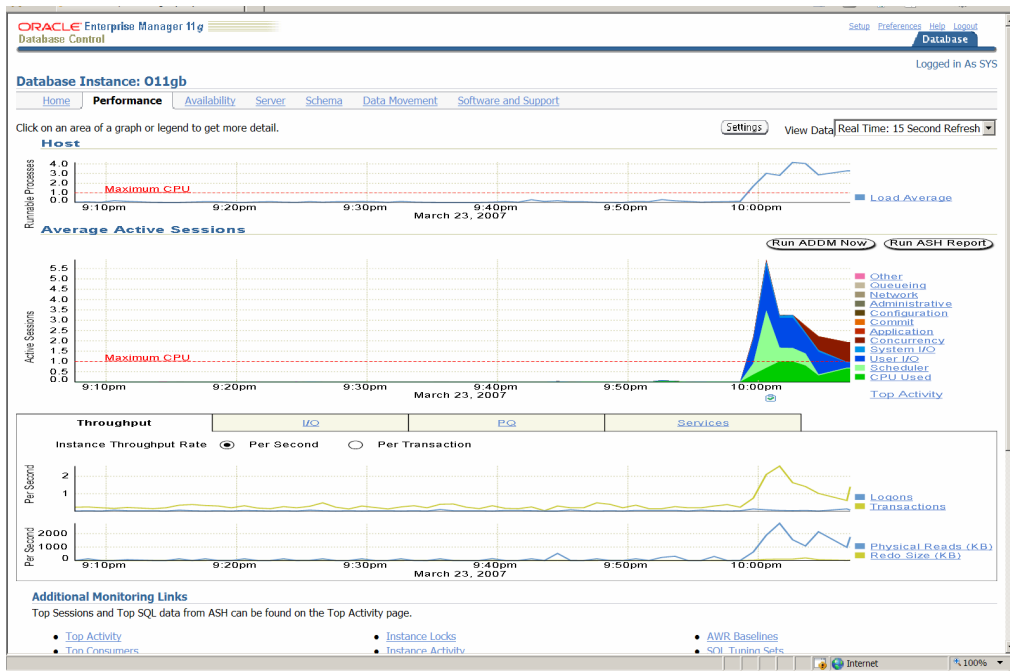


Figure 7: Run ADDM Now when a performance spike occurs

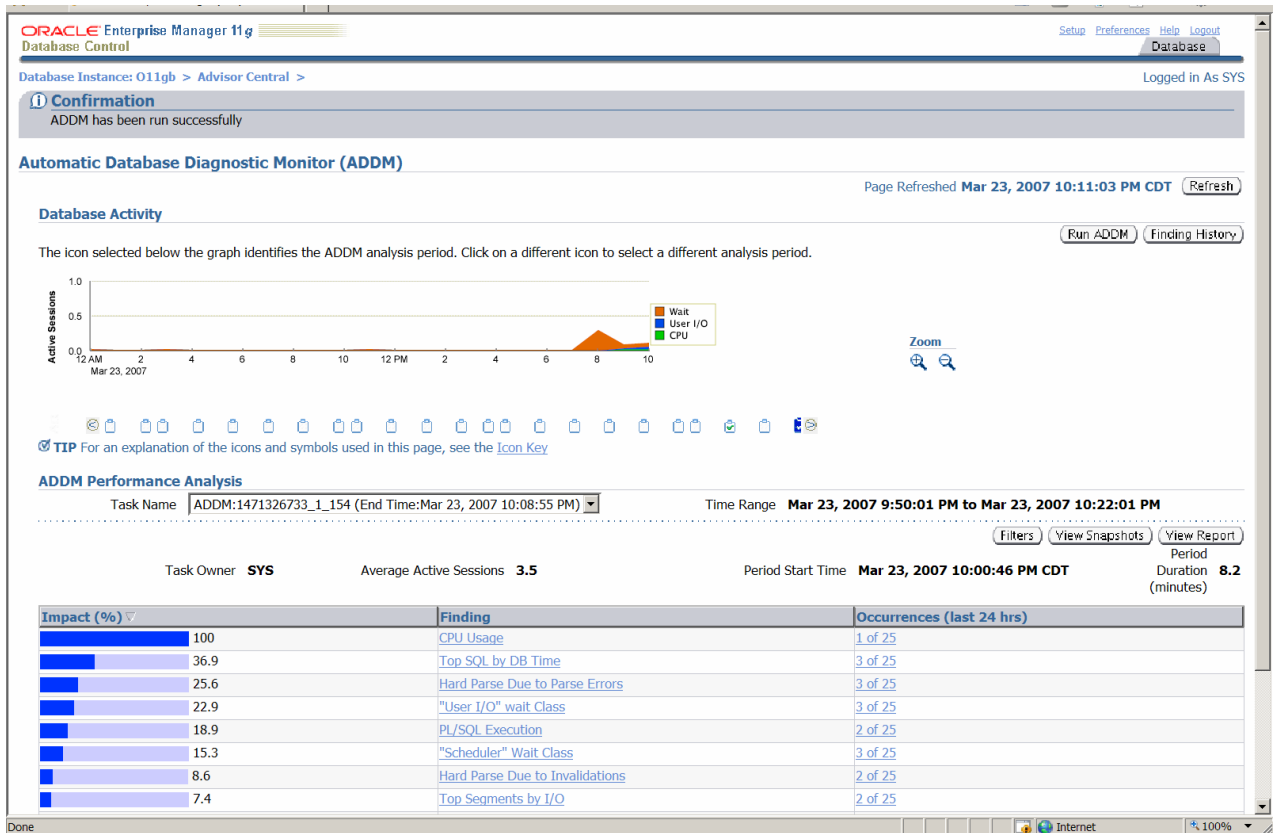


Figure 8: Result of “Run ADDM Now” is a CPU Usage Issue

Some of the ADDM considerations include:

- 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, Checkpoint.

ORACLE'S SQL PLAN MANAGEMENT (SPM)

Oracle's SQL Plan Management is another great tool! It 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 real application testing to test the 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

SQL Plan Management can recommend SQL Profiles or SQL Plan Baselines. 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 a SQL statement for the query optimizer to generate a better execution plan. **An outline/baseline contains HINTS** for a 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. You can view these in DBA_PLAN_BASELINES. You can also export a SQL Tuning Set and import it to new system. Capture baselines for Tuning Set with DBMS_SPM (see later example on entire syntax). You can also use a pack/unpack function to pack/unpack all plans in a system for transporting. Figures 10-14 show how to use Oracle's SPM creating a SQL tuning set through reviewing the results and other SQL Advisor Options.

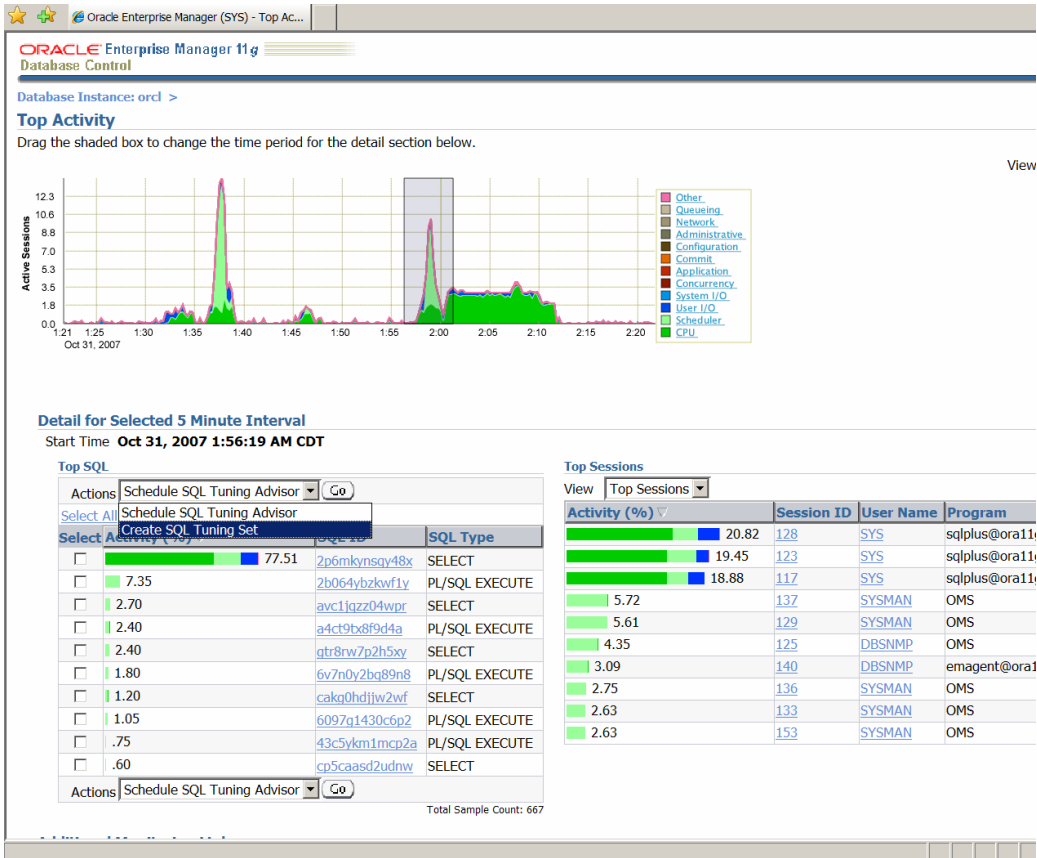


Figure 9: SPM: Start by creating a SQL Tuning Set for a performance problem

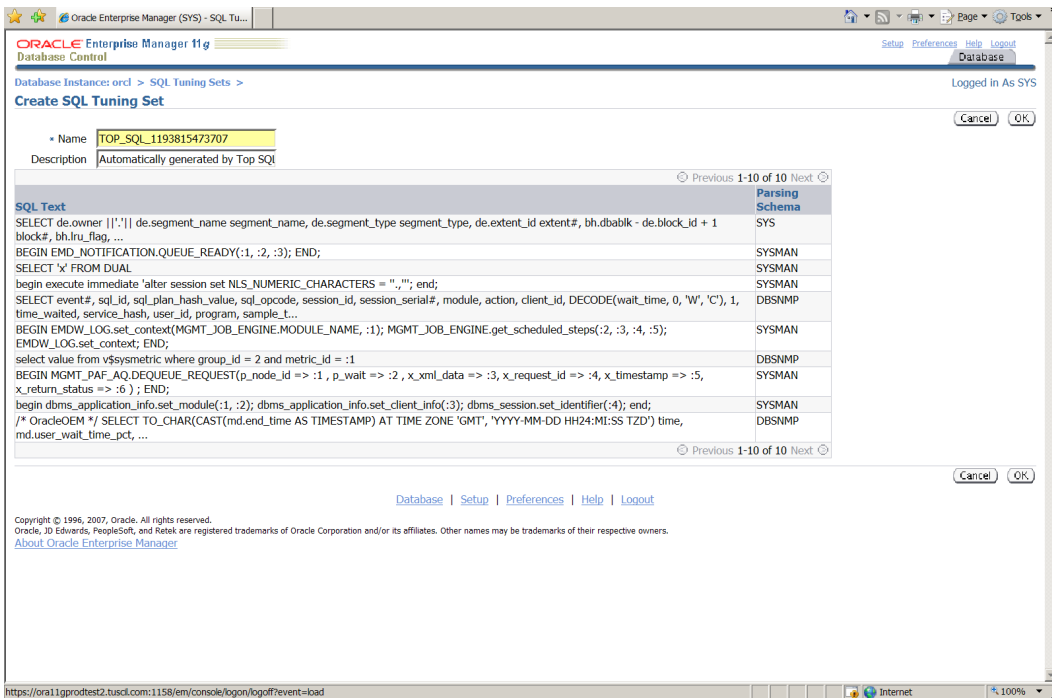


Figure 10: SQL Tuning Set for a performance problem

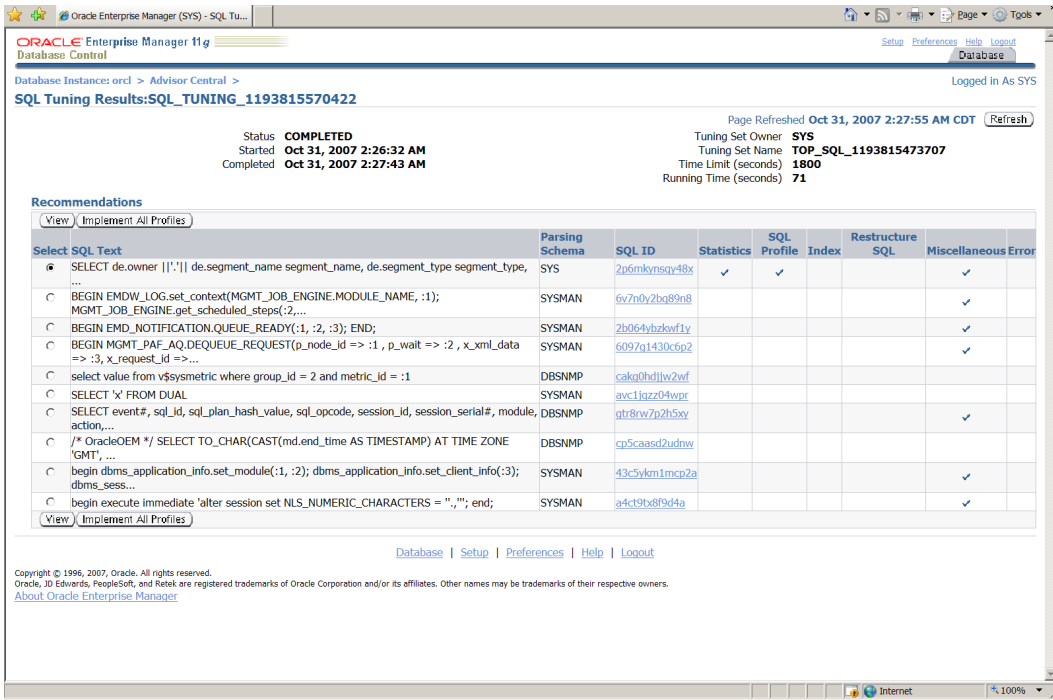


Figure 11: SQL Tuning Set results with suggestions

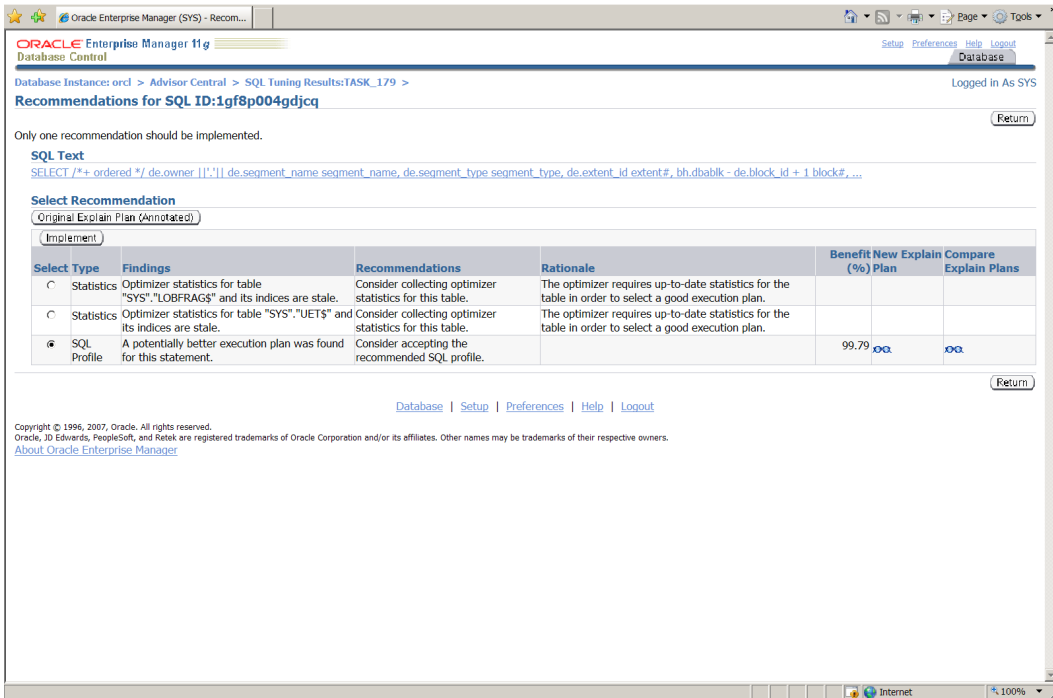


Figure 12: Recommendations for a specific statement

Oracle Enterprise Manager (SYS) - Explain ...

Database Instance: orcl > Advisor Central > SQL Tuning Results:TASK_179 > Recommendations for SQL ID:1gf8p004gdjqc

Logged in As SYS

Compare Explain Plans

Original Explain Plan (Annotated)
 Indicates an adjustment from the original plan by the SQL Tuning Advisor
 Plan Hash Value **2347322369**

Operation	Line ID	Object	Object Type	Order	Rows	Bytes	Cost Time	CPU Cost	I/O Cost
SELECT STATEMENT	0				121	0.270	983,655 11,804	12,350,714,281,984	168,630
SORT ORDER BY	1				120	0.270	983,655 11,804	12,350,714,281,984	168,630
NESTED LOOPS	2				119	0.270	983,654 11,804	12,350,698,553,344	168,630
HASH JOIN	3				7	1.708	0 1 1	8,647,788	0
NESTED LOOPS	4				5	0.176	0 1 1	710,600	0
VIEW	5				3	0.013	0 1 1	355,300	0
SORT AGGREGATE	6				2	0.059			
FIXED TABLE FULL	7	SYS.X\$KSLLTR_CHILDREN	TABLE (FIXED)	1	120,000	0 1 1	355,300	0	0
FIXED TABLE FULL	8	SYS.X\$KSLLTR_CHILDREN	TABLE (FIXED)	4	7,670	0 1 1	355,300	0	0
FIXED TABLE FULL	9	SYS.X\$SBH	TABLE (FIXED)	6	6,738	0 1 1	350,000	0	0
VIEW	10	SYS.DBA_EXTENTS	VIEW	118	0.114	89,423 1,074	1,122,790,014,976	15,330	

New Explain Plan With SQL Profile
 Plan Hash Value **2138758942**

Operation	Line ID	Object	Object Type	Order	Rows	Bytes	Cost Time	CPU Cost	I/O Cost
SELECT STATEMENT	0				124	0.262	1,972 24	702,635,712	1,926
SORT ORDER BY	1				123	0.262	1,972 24	702,635,712	1,926
HASH JOIN	2				122	0.262	1,971 24	687,481,920	1,926
HASH JOIN	3				7	1.568	1 1 1	8,647,788	0
NESTED LOOPS	4				5	0.176	0 1 1	710,600	0
VIEW	5				3	0.013	0 1 1	355,300	0
SORT AGGREGATE	6				2	0.059			
FIXED TABLE FULL	7	SYS.X\$KSLLTR_CHILDREN	TABLE (FIXED)	1	120,000	0 1 1	355,300	0	0
FIXED TABLE FULL	8	SYS.X\$KSLLTR_CHILDREN	TABLE (FIXED)	4	7,670	0 1 1	355,300	0	0
FIXED TABLE FULL	9	SYS.X\$SBH	TABLE (FIXED)	6	5,469	0 1 1	350,000	0	0
VIEW	10	SYS.DBA_EXTENTS	VIEW	121	18.229	1,970 24	671,240,320	1,926	
UNION-ALL	11				120				
NESTED LOOPS	12				72	0.222	235 3	14,023,343	234

Figure 13: Compare explain plans for before/after recommendations

Oracle Enterprise Manager (SYS) - SQL Ad...

ORACLE Enterprise Manager 11g
 Database Control

Database Instance: orcl > Advisor Central >

Logged in As SYS

SQL Advisors

The SQL Advisors address several important use cases having to do with SQL: identify physical structures optimizing a SQL workload, tune individual statements with heavy execution plans, identify and correct result set divergence, build test cases for failed SQL.

SQL Access Advisor
[SQL Access Advisor](#) Evaluate an entire workload of SQL and recommend indexes, partitioning, materialized views that will improve the collective performance of the SQL workload.

SQL Tuning Advisor
[SQL Tuning Advisor](#) Analyze individual SQL statements, and recommend SQL profiles, statistics, indexes, and restructured SQL to SQL performance.
[Automatic SQL Tuning Results](#) View the results of automated execution of SQL Tuning Advisor on observed high-load SQL.

SQL Repair Advisor
 The SQL Repair Advisor can analyze and potentially patch failing SQL statements.

[SQL Incident Analysis](#) SQL incident analysis is initiated from the Support Workbench for SQL failures that are generating Support Workbench incidents.
[Click here to go to Support Workbench.](#)

[SQL Failure Analysis](#) SQL failure analysis is used for non-incident SQL failures and can be accessed through either SQL Details or SQL Worksheet.
[Click here to go to SQL Worksheet.](#)

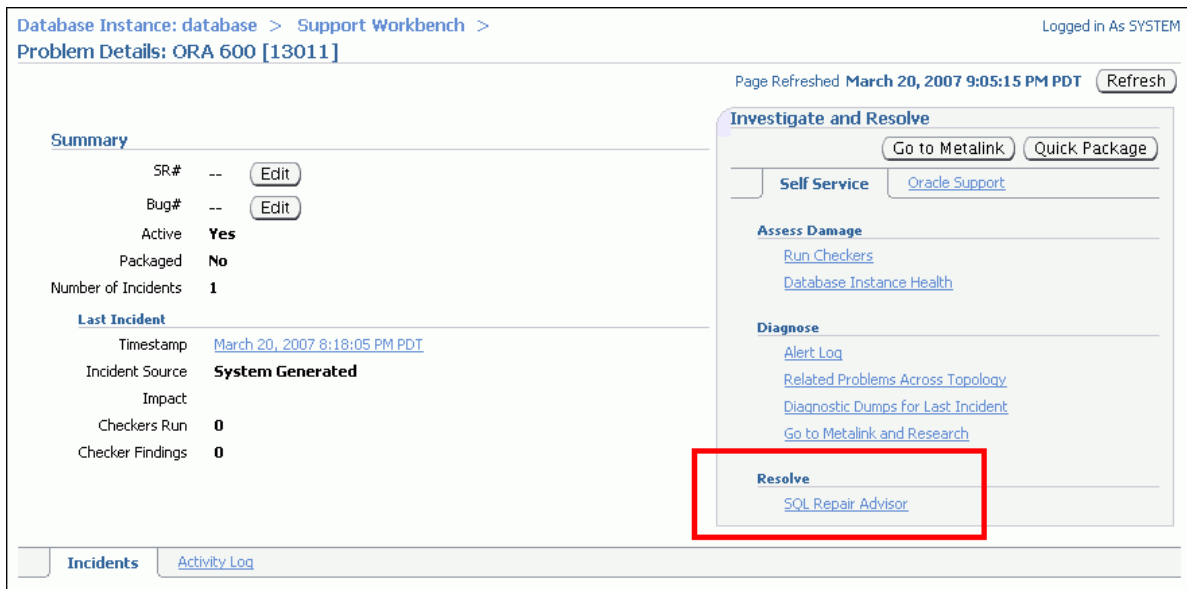
[Database](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

Copyright © 1996, 2007, Oracle. All rights reserved.
 Oracle, JD Edwards, PeopleSoft, and Retak are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
[About Oracle Enterprise Manager](#)

Figure 14: There are other SQL Advisors

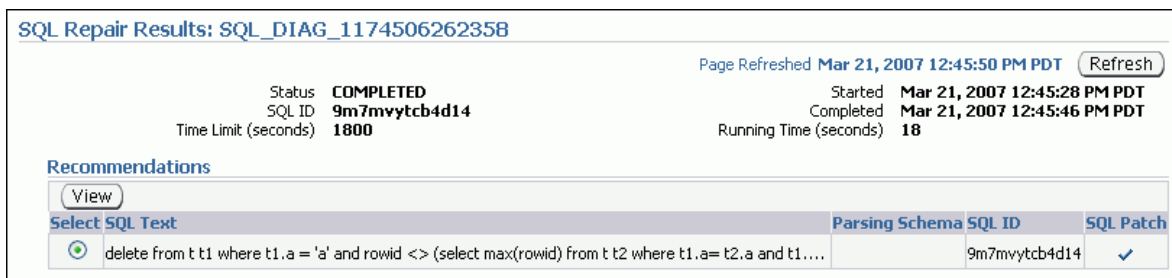
SQL REPAIR ADVISOR

Oracle's new SQL Repair Advisor helps you to get by Oracle errors until they are patched. It is used to repair problem SQL that comes up with an Oracle error. It reloads and recompiles SQL statements to gather diagnostics information to fix the statement. It then uses the diagnostic information to repair the problem SQL statement (DBMS_SQLDIAG). It will try to fix the 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. It will tell you if it's going to be a slower route, which is nice. You still want to log the issue with Metalink to ensure a permanent fix to the issue. You can access the SQL Repair Advisor straight from the Alerts. Figures 15 & 16 are screenshots from Oracle that show the use of the SQL Repair Advisor.



The screenshot shows the Oracle Support Workbench interface. The top navigation bar includes "Database Instance: database > Support Workbench >" and "Problem Details: ORA 600 [13011]". The user is logged in as SYSTEM. The page was refreshed on March 20, 2007 at 9:05:15 PM PDT. The main content area is divided into two sections: "Summary" and "Investigate and Resolve". The "Summary" section shows details for a problem, including SR#, Bug#, Active status (Yes), Packaged status (No), and Number of Incidents (1). The "Last Incident" section shows a timestamp of March 20, 2007 at 8:18:05 PM PDT, Incident Source (System Generated), Impact, Checkers Run (0), and Checker Findings (0). The "Investigate and Resolve" section has a "Resolve" sub-section where the "SQL Repair Advisor" link is highlighted with a red box. Other links in the "Investigate and Resolve" section include "Go to Metalink", "Quick Package", "Self Service", "Oracle Support", "Assess Damage", "Run Checkers", "Database Instance Health", "Diagnose", "Alert Log", "Related Problems Across Topology", "Diagnostic Dumps for Last Incident", and "Go to Metalink and Research".

Figure 15: After clicking on an alert for errors, the SQL Repair Advisor is available



The screenshot shows the "SQL Repair Results" page for SQL_DIAG_1174506262358. The page was refreshed on Mar 21, 2007 at 12:45:50 PM PDT. The repair status is "COMPLETED". The SQL ID is 9m7mvytcb4d14. The Time Limit (seconds) is 1800. The repair started on Mar 21, 2007 at 12:45:28 PM PDT and completed on Mar 21, 2007 at 12:45:46 PM PDT. The Running Time (seconds) is 18. The "Recommendations" section shows a table with one row of results:

Select SQL Text	Parsing Schema	SQL ID	SQL Patch
View delete from t t1 where t1.a = 'a' and rowid <> (select max(rowid) from t t2 where t1.a= t2.a and t1...		9m7mvytcb4d14	✓

Figure 16: The SQL Repair Advisor has found a SQL Patch to fix my error

SQL PERFORMANCE ANALYZER

The SQL Performance Analyzer is great for measuring and reporting on performance before and after a change. It uses the DBMS_SQLTUNE package. It is great to use for Database Upgrades, Application Upgrades, Hardware Changes, Database or Schema Changes, and it's very good for SQL Tuning, especially batches where you can create a SQL Tuning Set to test the before and after. It's also easy to run between two systems since you can Capture the SQL on one system, Transport the SQL to the new system, Make any Changes to test (such as SGA/Optimizer) and then Compare before and after performance. Tune the problems! Re-test it!

Within SQL Performance Analyzer Workflows, there are three choices that allow you to test before and after experiments. The first is an Optimizer Upgrade Simulation, which is the ability to test the effects of changing the optimizer version for a SQL Tuning Set. The second is a Parameter Change where you can test and compare initialization parameter changes on a SQL Tuning Set. The third is a Guided Workflow, which is the ability to create a SQL Performance Analyzer Task and execute customer experiments using manually created replay trials using your own changed parameters. Figure 17 shows the main screen where these three choices are available. Figure 18 shows comparative results of the before and after results testing a parameter change. For each of the three options the result is this fantastic comparison illustrated in Figure 18 that shows graphically the before and after performance of all of the statements.

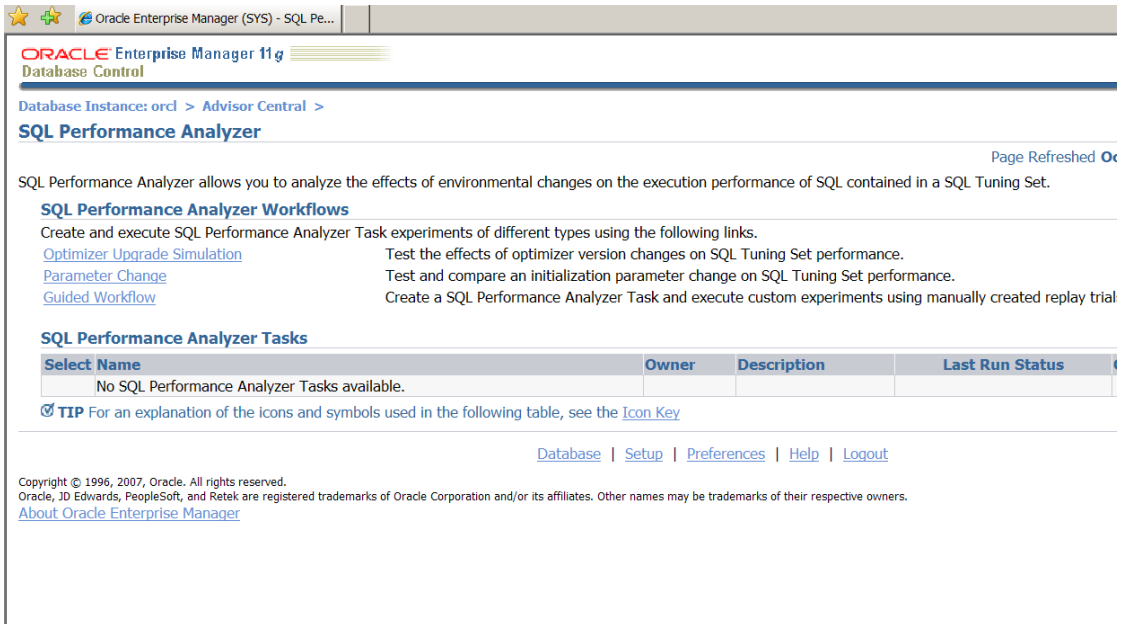


Figure 17: The SQL Performance Analyzer offers multiple Choices for Tuning

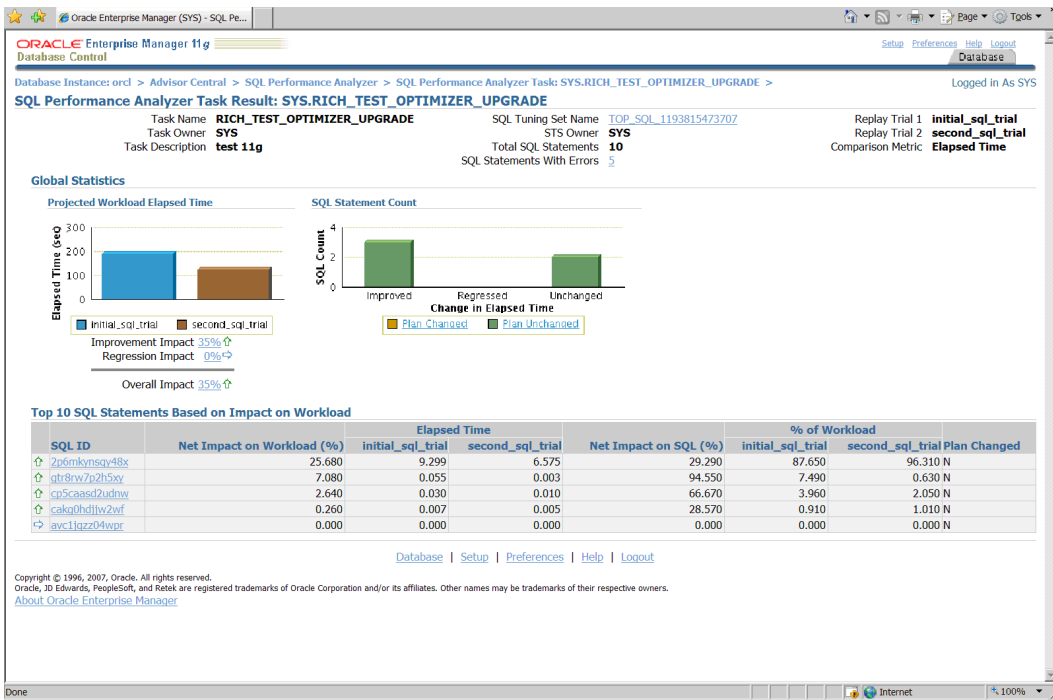


Figure 18: The SQL Performance Analyzer final screen shows before/after results

REAL APPLICATION TESTING (CAPTURE AND REPLAY)

This is an OUTSTANDING new feature within Oracle that allows you to capture a **database workload** on one system and replay later on a different system. This is useful to compare two physically different systems with the exact same hardware. Perhaps this will rival other testing tools in the future (may be more precise!). It even allows a capture of a workload on a database from 10gR2 and a restore of the database and testing the workload on a new Oracle 11g system.

You use this by restoring a copy of the database on a test system to the SCN when capture begins. You then perform an upgrade and make changes to the test system as needed. There are some additional preprocessing steps and you need to configure the test system for replay. You then replay the replay workload on the restored database. It is more complex than explained here, so please refer to the Oracle documentation for detailed information and the exact steps.

INTERVAL PARTITIONING

With the advent of 11g, Oracle also added a very nice feature for partitioning called interval partitioning. Previously, you always had to specify a MAXVALUE where any record not falling into a specified partition would fall. If you failed to specify a MAXVALUE, you would get an error if a value fell outside the range of specified values as detailed in the example below:

```
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

In 11g, I can now use interval partitioning to help in range partitioning where Oracle automatically creates a partition when the inserted value exceeds all other partition ranges. There are some restrictions such as:

- You can only specify one partitioning key column that must be 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.

In 11g, I can now use interval partitioning to eliminate this potential user error. Consider the following example and note the INTERVAL keyword:

```
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.

```
insert into dept_new2 values(40,null);
```

1 row created.

```
insert into dept_new2 values(50,null);
```

1 row created.

```
insert into dept_new2 values(99,null);
```

1 row created.

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

SEGMENT_NAME	PARTITION_NAME
DEPT_NEW2	D1
DEPT_NEW2	D2
DEPT_NEW2	D3
DEPT_NEW2	SYS_P41
DEPT_NEW2	SYS_P42
DEPT_NEW2	SYS_P43

You can see from the example above that I no longer get an error inserting the rows that are not in the pre-created partitions. You can also see that three new partitions were created for the three new intervals based on the new inserted values.

PARTITION COMPRESSION

Another 11g enhancement is the ability to COMPRESS individual partitions. A compression as high as 3.5 to 1 is possible. Compressed tables now support: DML Statements, Add and Drop Column, and Partition level COMPRESS or NOCOMPRESS.

IMPORTANT NOTE: There is a New Advanced Compression Option which is chargeable for some compression options. Please check licensing to see which ones.

The example below shows an example of compressing the entire table, BUT, leaving a single partition (D2) as not compressed.

```
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.

An Oracle study showed the following features/options used by Large-Scale Data Warehouses. You can see from the graph that partitioning and compression are very important:

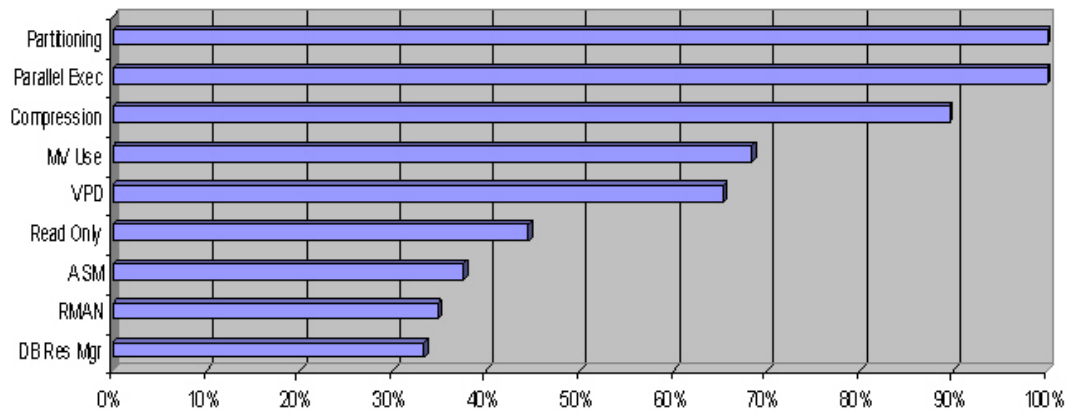


Figure 19: Features used by Large-Scale Data Warehouses (Oracle study)

ORACLE SECURE FILES

Oracle has increased the performance of large objects by introducing Oracle Secure Files. These provide high-performance transactional access to large object data such as:

- Documents
- Medical
- CAD
- Imaging

It provides low-latency, high throughput, concurrent access by using space-optimized storage. In Oracle testing, when compared to LOBs, Secure files were 22x faster when adding files using deleted space, 6x faster when using PL/SQL reads, and 2x faster adding files using new disk space.

It is crucial to protect and secure your valuable large object data by keeping it in the database. Storing objects in the database ensures:

- Database security, reliability, and scalability
- Transparent data encryption when accessed
- Compression and de-duplication
- Single view and management of data
- Superset of LOB interfaces – easy migration

THE RESULT CACHE

With the result cache you can cache function results of queries and query fragments in memory for **future executions**. Oracle even does this internally to improve performance for all SQL. You would generally want to choose calculations that are frequently run & also choose data that does NOT frequently change. The **RESULT_CACHE & RELIES_ON** clauses are used. You set the size with the **RESULT_CACHE_SIZE** initialization parameter and it takes its memory from the Shared Pool. You can also set the parameter **RESULT_CACHE_MODE=force (manual)** to force its use. Similar to how you flush the buffer cache or shared pool, you can use **DBMS_RESULT_CACHE.FLUSH** to clear the contents. Note that results are NOT Passed between RAC/Grid nodes. Please check the Oracle docs for other Restrictions & Rules! We did a test to see how fast it was with a 1M row table. The query below is the one that we used:

```
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;
```

Then we ran the query two different times in two different sessions:

Step 1 - In Session 1 we executed the query **with the RESULT_CACHE hint** and it returned an elapsed time of 3.18 seconds (cache it).

Step 2 - In Session 2 we executed query without the **RESULT_CACHE** hint, but with **RESULT_CACHE_MODE=force** and it returned an elapsed time of 0.86 seconds (cached!!).

If you use autotrace (in SQL*Plus: Alter session set autot on;) you will see the result cache in the explain plan output as displayed in the code below:

```
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			
2	SORT AGGREGATE		1		
3	INDEX FULL SCAN	PK_EMP	14	1 (0)	00:00:01

THE INVISIBLE INDEX

It's hard to decide which columns to index. The primary key is automatically indexed, the foreign keys SHOULD also be indexed, then what? Even harder yet is deciding which index to remove that might be a bad index. Every time a record is inserted, all of the indexes have to be updated. If the column of an index is updated, the index has to be updated. How do decide which index to drop without causing a slew of full table scans or cartesian joins for subsequent user queries. The answer just might be the invisible index. It allows you to turn off the use of the index, but continue to maintain the index (for any DML – INSERT/UPDATE/DELETE) in case you need to quickly turn it back on. You do this by making it visible or invisible:

- ALTER INDEX idx1 INVISIBLE;
- ALTER INDEX idx1 VISIBLE;
- CREATE INDEX... INVISIBLE;

SEQUENCES IN PL/SQL EXPRESSIONS

In previous versions of Oracle you needed to retrieve the value of a sequence (CURRVAL / NEXTVAL) by invoking a cursor (explicit or implicit). **In 11g, no cursor is needed**, therefore the code is much more efficient. For big jobs – you save MANY cursors! Consider the following two examples showing the pre-11g and post-11g ways of doing things:

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;
```

OPTIMIZER ADVANCES

(Special Thanks: Maria Colgan, Penny Avril & Debbie Migliore for contributions to this section)

The Oracle optimizer is also going to a new level of sophistication. There is an improved SPEED and Quality to gathering stats using AUTO-SAMPLING. When you manually gather stats, it's usually impossible to find a sample size that works for ALL tables, so you often need to COMPUTE which is slower. It is especially hard to find a good sample size when the data distribution is heavily skewed. Oracle's NEW auto-sampling "discovers" the best sample size for every table in your system for you. It is pitched to give you the Quality of a COMPUTE with SPEED of a SAMPLE. Oracle's goal is to OBSOLETE the need and use of sampling. The Accuracy is pitched to be comparable to COMPUTE.

Another issue with gathering stats is with regard to partitions. 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 with is extremely time consuming. In 10g, you have to manual copy statistics to new partitions.

In 11g, it gathers stats for TOUCHED PARTITIONS only! Table stats are refreshed WITHOUT scanning the un-touched partitions.

Another issue with gathering stats is that 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, you can now gather stats and save as PENDING. You can then 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! Below is a VERY simple example showing some of the commands:

Gather Stats but make them PENDING:

```
select dbms_stats.get_prefs('PUBLISH', 'SH', 'CUST') publish from dual;
```

```
PUBLISH
```

```
-----
```

```
TRUE
```

(currently the stats are published for the CUST table owned by SH)

```
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
```

```
-----
```

```
FALSE
```

(NOW the stats are NOT published for the CUST table owned by SH)

Check to see if Stats are out there for the CUST table:

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

TABLE_NAME	ANALYZE_T	NUM_ROWS	BLOCKS	AVG_ROW_LEN
-----	-----	-----	-----	-----
CUST				

(NO stats for the CUST table)

```
execute dbms_stats.gather_table_stats('SH', 'CUST');
PL/SQL procedure successfully completed.
```

(Gather stats for the CUST table)

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

TABLE_NAME	ANALYZE_T	NUM_ROWS	BLOCKS	AVG_ROW_LEN
-----	-----	-----	-----	-----
CUST				

(STILL NO stats for the CUST table – they need to be PUBLISHED)

PUBLISH Stats after Testing Complete:

```
alter session set optimizer_use_pending_statistics = true;
(Then run your query – If ready/better – publish the new stats)
```

```
exec dbms_stats.publish_pending_stats('SH', 'CUST');
PL/SQL procedure successfully completed.
```

(PUBLISH stats for the CUST table)

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

TABLE_NAME	ANALYZE_T	NUM_ROWS	BLOCKS	AVG_ROW_LEN
CUST	13-OCT-07	55500	1485	180

(NOW there are stats for the CUST table – they have been PUBLISHED)

```
exec dbms_stats.delete_table_stats('SH', 'CUST'); <to delete>
```

MULTI-COLUMN STATISTICS (EXTENDED OPTIMIZER 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 Optimizer has to estimate the correct cardinality. Will the additional column (as in those examples in bulleted list above) be a condition that reduces the result set or not? Should that additional column be used? Oracle calculates correlated statistics (between the columns) so the optimizer will make great decisions. Single column statistics and histograms are not enough!

Oracle also provides a way to you to collect stats on a group of columns. It is fully integrated into the existing statistics framework. It is automatically maintained with column statistics and has immediate benefits for any application by computing the accurate cardinalities for inter-related columns that you deem are going to be used together. It ensures that multiple predicates on the same table are estimated correctly. Consider a simple example using the CUSTOMERS tables.

First, consider the output of column level statistics on the table:

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

```
COLUMN_NAME                NUM_DISTINCT HISTOGRAM
-----
CUST_VALID                  2 NONE
COUNTRY_ID                  19 FREQUENCY
CUST_STATE_PROVINCE        145 NONE
CUST_CITY_ID                620 HEIGHT BALANCED
CUST_CITY                   620 NONE
CUST_LAST_NAME              908 NONE
CUST_FIRST_NAME             1300 NONE
CUST_ID                     55500 NONE
...
23 rows selected.
```

Now lets create the extended statistics group combining multiple columns & re-gather statistics on the CUSTOMER table (query user_tab_col_statistics to see new column):

```
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.
```

Now consider the output of column level statistics on the table and the new multi-column statistics are visible:

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

COLUMN_NAME	NUM_DISTINCT	HISTOGRAM
SYS_STUJGVLRVH5USVDU\$XNV4_IR#4	145	FREQUENCY
CUST_VALID	2	FREQUENCY
COUNTRY_ID	19	FREQUENCY
CUST_STATE_PROVINCE	145	FREQUENCY
CUST_CITY_ID	620	HEIGHT BALANCED
CUST_CITY	620	HEIGHT BALANCED
CUST_LAST_NAME	908	HEIGHT BALANCED
CUST_FIRST_NAME	1300	HEIGHT BALANCED
CUST_ID	55500	HEIGHT BALANCED
...		

24 rows selected.

To DROP Extended Statistics:

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

PL/SQL procedure successfully completed.

MANAGING THE GRID

Using Enterprise Manager to manage the database is fast and one of the most productive ways to manage many systems at once. The Database Control screen in 11g is displayed in Figure 20. One of the best screens to manage the grid is displayed in Figure 21 (Note that 11g Grid Control was not out yet, so I'm using 10g Grid Control to show this powerful tool). It's the screen to click on a cluster and see whether the nodes are up or down as well as see the individual nodes. Here is the cluster "ioug" showing six nodes that are all up. To get to this screen, I just went to the Targets tab and clicked on the ioug cluster.

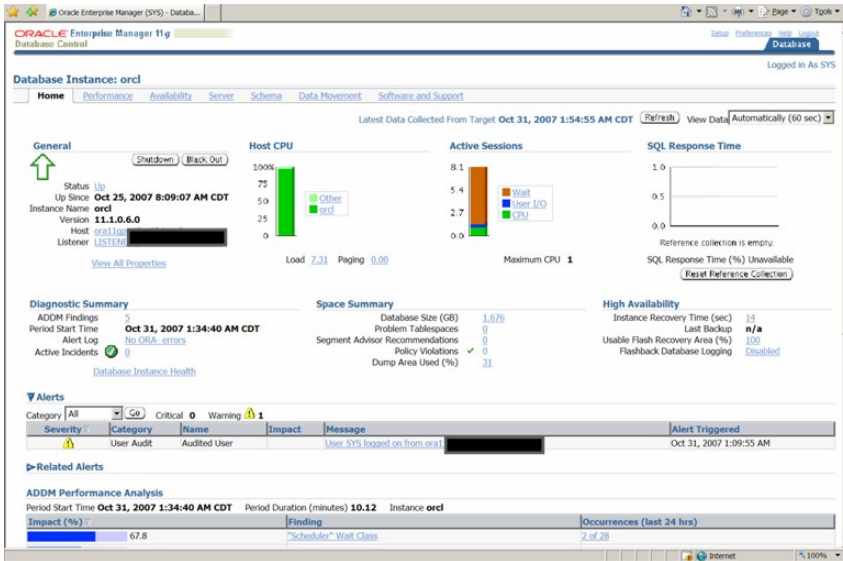


Figure 20: The 11g Enterprise Manager - Database Control

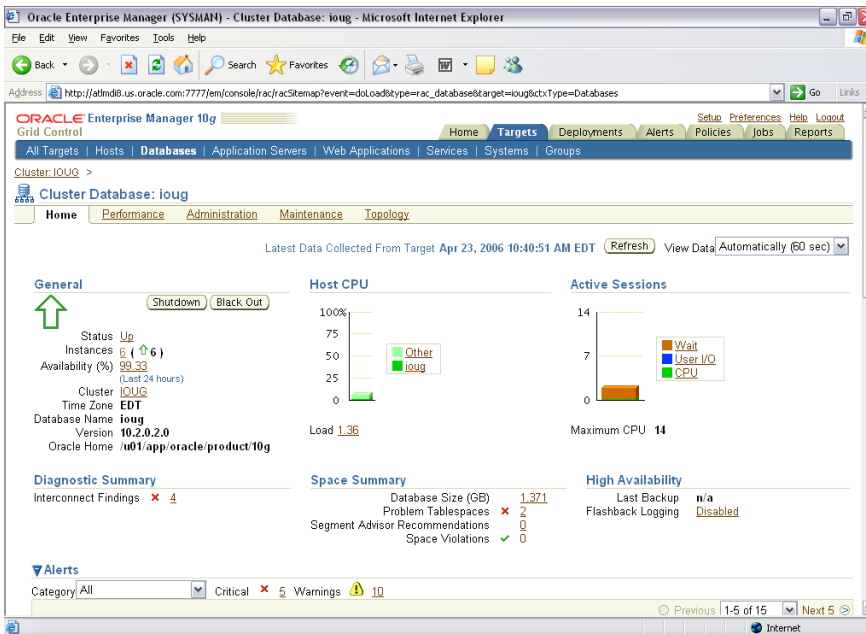


Figure 21: Looking at the IOUG Cluster Database under Targets/Databases

If you move down the page a bit, you can see the instances (all using ASM) that are associated with this cluster as shown in Figure 22.

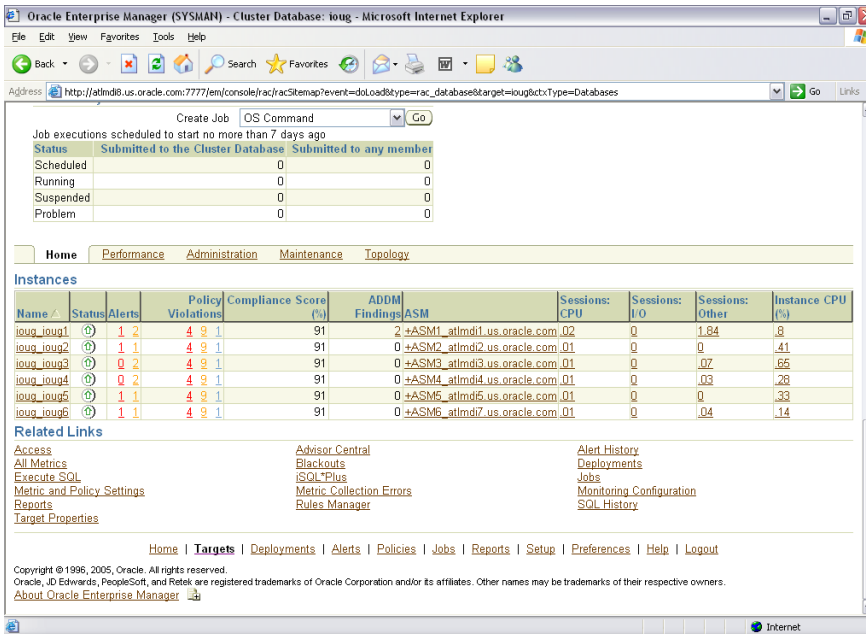


Figure 22: Further down the page of Figure 1 we see the individual Nodes 1-6

If I click on the topology tab (see Figure 23), we can see the topology for all six instances (each instance is on a separate node, so there are also six separate nodes. Notice that my mouse if over one of the instances and additional information about the instance is provided.

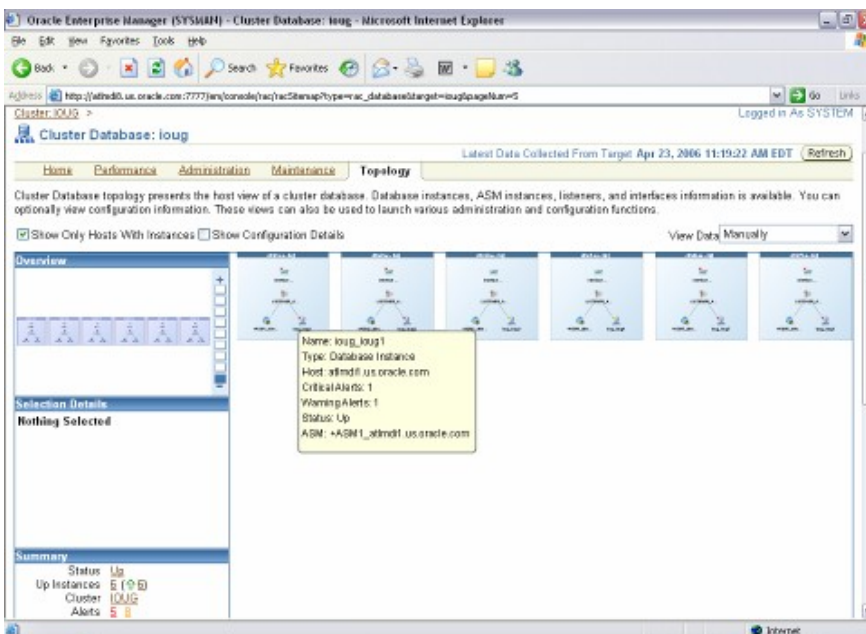


Figure 23: Looking at the Topology of the 6 Nodes in the IOUG Cluster

If I click on the Performance tab and then click onto the CPU Used chart (see Figure 24), I can see performance all nodes in the “ioug” cluster, each in a different color.

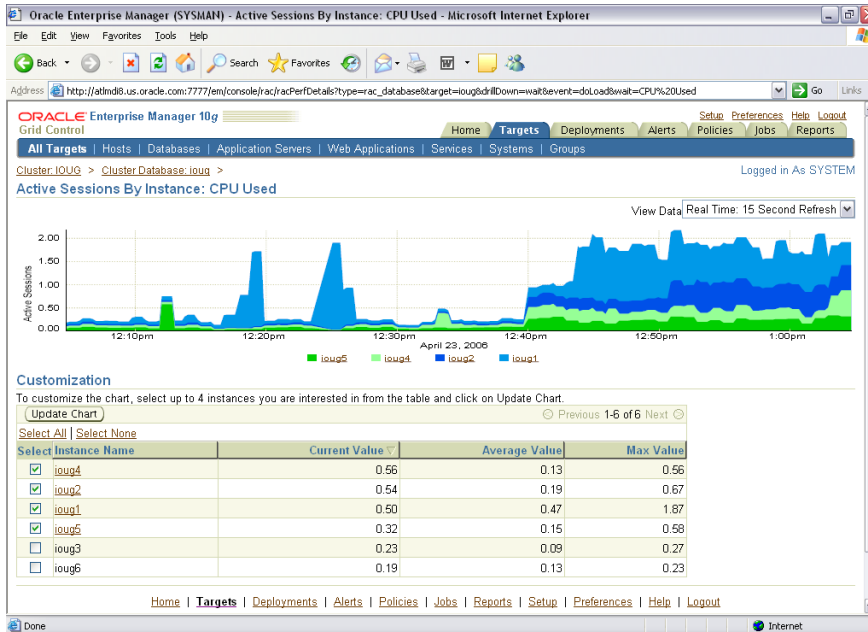


Figure 24: Looking at CPU for 4 of the selected nodes in the IOUG Cluster

Figure 25 shows an extremely helpful screen, the Hang Analysis screen of Enterprise manager, which allows you to view blocking sessions and resolve them. It is a very fast screen even when the system is overwhelmed. I usually kill the session in SQL*Plus after finding them here.



Figure 25: Hang Analysis Screen

SECURITY ENHANCEMENTS

Here are a few new things to consider in the world of security. Please see the Oracle documentation for an exhaustive list. **A BIG CHANGE in 11g is that Passwords will be case sensitive now!** Overall, 11g is more restrictive “out of the box” than 10g was as far as security goes. The Password lock time (1), password grace time (7) and password life time (180) are all more restrictive. The Failed login attempts stays the same (10). There is also an enhanced hashing algorithm for passwords / DES still available. Strong passwords (set via password complexity verification in Enterprise Manager 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

Another BIG CHANGE is that the Audit Trail is ON by default (was off in 10g) and AUDIT_TRAIL=DB is now the default.

Things that will be audited by default (AUDIT_TRAIL=DB) 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

The cost of auditing is said to have improved to be 1-2% cost on the TPC-C benchmark.

THE V\$ VIEWS OVER THE YEARS AND IN 11G

The ability that Oracle gives you to manage the database when you need it is unsurpassed. As stated earlier in this article, you can set a single initialization parameter for the Oracle database and yet the amount of tools that Oracle provides for you to modify, analyze or tweak your database based on your specific business rules, parameters or operating issues is absolutely unbelievable. Here is a listing of the V\$ views over the version of Oracle:

<u>Version</u>	<u>V\$ Views</u>	<u>X\$ Tables</u>
6	23	(35?)
7	72	126
8.0	132	200
8.1	185	271
9.0	227	352
9.2	259	394
10.1.0.2	340 (+31%)	543 (+38%)
10.2.0.1	396	613
11.1.0.6	484 (+22%)	798 (+30%)

SUMMARY

We've covered a lot of topics in this paper. It was quite painful to put together due to the length required to cover this amount of new features, but I wanted to give a broad coverage of some of the 11g features that I thought every manager should be aware of when looking into the next version of Oracle. Often, people look at the pain of converting to the next version instead of looking to the benefits in the next version. Hopefully, this paper has given you some reasons to investigate whether it's time for you to start testing 11g in your business.

REFERENCES:

Oracle10g Performance Tuning Tips and Techniques; Niemiec, 2007

New Optimizer Features in 11g, Maria Colgan

All Oracle11g Documentation from Oracle Beta Site

Introduction to Oracle Database 11g, Ken Jacobs

Oracle Database 11g New Features, Linda Smith

Oracle9i Performance Tuning Tips and Techniques, Niemiec, 2003

Database Secure Configuration Initiative: Enhancements with 11g, www.oracle.com

www.tusc.com, www.oracle.com, www.ioug.org

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

Special thanks Dan M., Bob T., Brad, Joe, Heidi, Mike K., Debbie, Maria, Linda for testing!

AUTHOR BIOGRAPHY

Rich Niemiec, 48, Rolta's President of International EICT (Enterprise Information & Communications Technology) and President of Rolta TUSC – A Rolta Company. Rolta TUSC is a Chicago-based systems integrator of Oracle-based business solutions since 1988; Rolta TUSC was the Oracle Partner of the Year in 2002, 2004, 2007 & 2008. Rolta is an international market leader in IT-based geospatial solutions, and caters to industries as diverse as infrastructure, telecom, electric, airports, defense, homeland security, urban development, town planning and environmental protection. Rich is the past President of the International Oracle Users Group (IOUG) and the current President of the Midwest Oracle Users Group (MOUG). Rich is one of six originally honored worldwide Oracle Certified Masters. In 2007, he authored the Oracle bestseller "Oracle10g Performance Tuning Tips & Techniques," an update of his previous 2 Oracle best sellers on Oracle8i and Oracle9i Performance Tuning. Rich was inducted into the Entrepreneurship Hall of Fame in 1998.

Rolta TUSC is an expert level consultancy that helps companies optimize their investment in Oracle technology. We provide integrated functional and technical solutions since 1988 in the areas of Oracle's E-Business Suite, Business Intelligence/Data Warehousing, Custom Development, Managed Services/Remote DBA, Database Services, Training & Mentoring and Oracle Licensing. Rolta also has software that works with the Oracle database in the areas of GIS, SOA Services and Business Intelligence. Please report errors in this article to rich@tusc.com. Neither TUSC nor the author warrants that this document is error-free. TUSC © 2010. This document cannot be reproduced without expressed written consent from an officer of TUSC except Collaborate 2008 may make copies and make this paper available as needed for the conference and proceedings.

Thanks Oracle: Ken Jacobs, Debbie Migliore, Penny Avril, Maria Colgan. and Linda Smith.

Oracle Disclaimer: This paper 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.