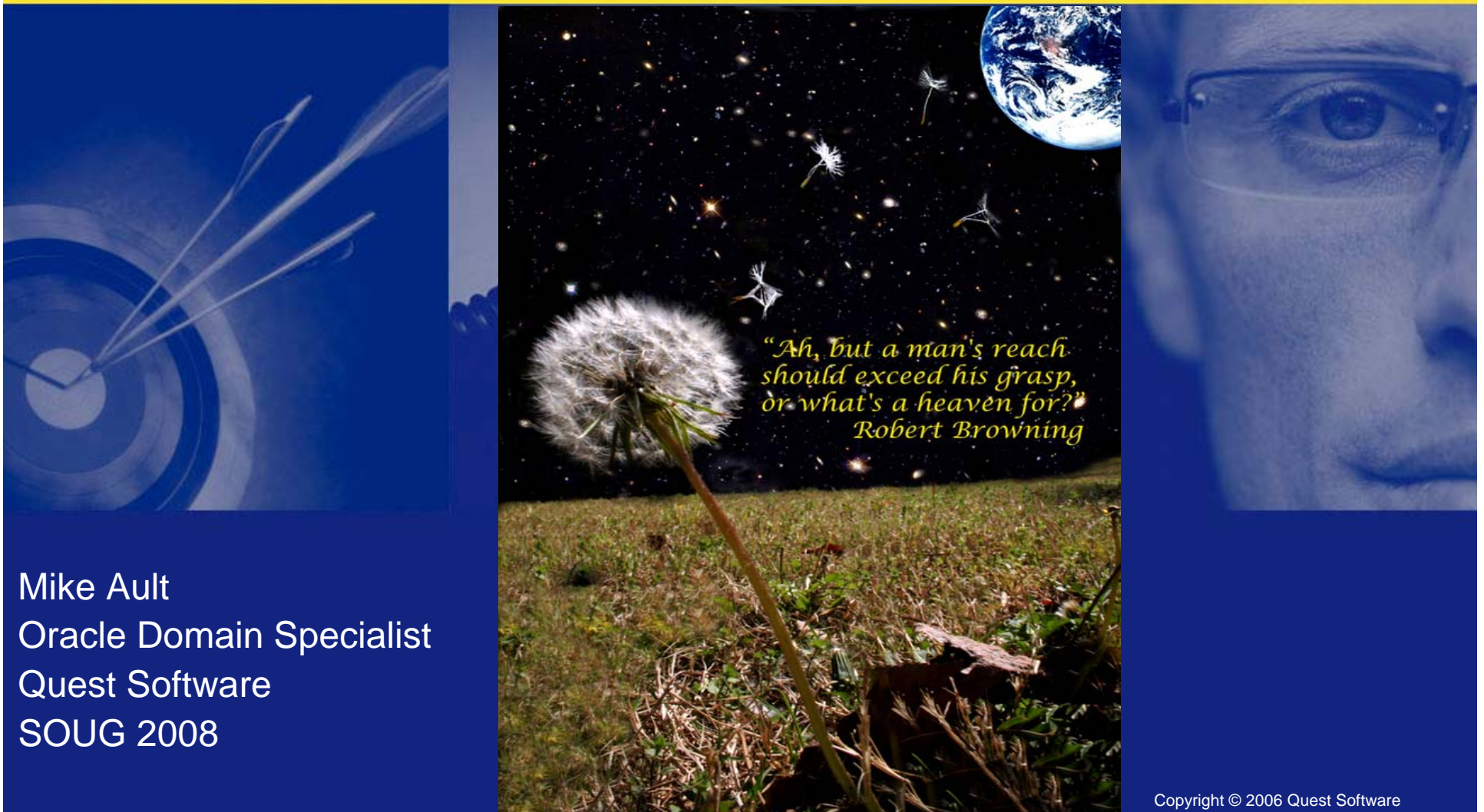


# The New Tuning Universe of Oracle11g

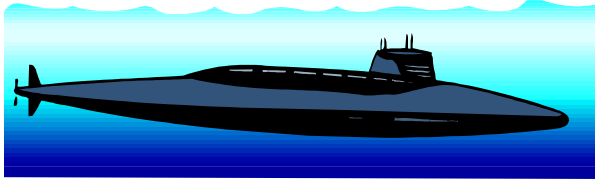


*"Ah, but a man's reach  
should exceed his grasp,  
or what's a heaven for?"  
Robert Browning*

Mike Ault  
Oracle Domain Specialist  
Quest Software  
SOUG 2008

# Michael R. Ault

## Oracle Domain Specialist







- Nuclear Navy 6 years
- Nuclear Chemist/Programmer 10 years
- Kennedy Western University Graduate
- Bachelors Degree Computer Science
- Certified in all Oracle Versions Since 6
- Oracle DBA, author, since 1990

**ORACLE** | CERTIFIED  
PROFESSIONAL

 **QUEST  
SOFTWARE**

# Oracle 11g Tuning Improvements

- Improved SQL Advisor
-  SQL Replay
-  Automatic SQL Tuning
-  SQL Statistics Management
-  SQL Plan Management








# First: SQL Advisor...Let's look Deeper!



# Improved SQL Advisor

- Incorporated into Automatic SQL Tuning
- Enhanced to perform faster with better recommendations
- Used by:
  -  – Automatic SQL Tuning Advisor
  -  – SQL Replay
  -  – RAT (Real Application Tuning)
  - OEM

# SQL Tuning Advisor

- The Automatic SQL Tuning capabilities utilized through the SQL Tuning Advisor.
- SQL Tuning Advisor takes a SQL Tuning Set and invokes the Automatic Tuning Optimizer to perform SQL tuning on the statements.
- The SQL Tuning Advisor output is advice or recommendations
  - Includes the rationale for each recommendation
  - Includes the expected benefit
- Recommendations relate to:
  - collection of statistics on objects
  - creation of new indexes
  - restructuring of the SQL statement
  - creation of SQL Profile.
- The user can choose to accept or reject the recommendation.

# SQL Tuning Set (STS)

- Required for tuning multiple statements.
- One to Many statements
- Is a database object that stores SQL statements along with their execution context.
- Can be created manually using command line APIs or automatically using Oracle Enterprise Manager

One...



Or Many!...



# SQL Advisor Inputs

- ADDM
- High-Load SQL Statements
- Cursor Cache
- STS



# SQL Advisor Options

- There are options to manage the scope and duration of a tuning task.
- The scope of a tuning task can be set to limited or comprehensive.
  - If the limited option is chosen, the SQL Tuning Advisor produces recommendations based on statistics checks, access path analysis, and SQL structure analysis. SQL Profile recommendations are not generated.
  - If the comprehensive option is selected, the SQL Tuning Advisor carries out all the analysis it performs under limited scope plus SQL Profiling.
- The comprehensive defaults to 30 minutes. You can change this, for complex statements it won't be enough!

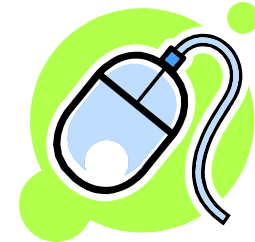
# SQL Advisor Outputs

- After analyzing the SQL statements, the SQL Tuning Advisor provides advice on:
  - optimizing the execution plan, the rationale for the proposed optimization, the estimated performance benefit, and the command to implement the advice.
- You simply have to choose whether or not to accept the recommendations to optimize the SQL statements.

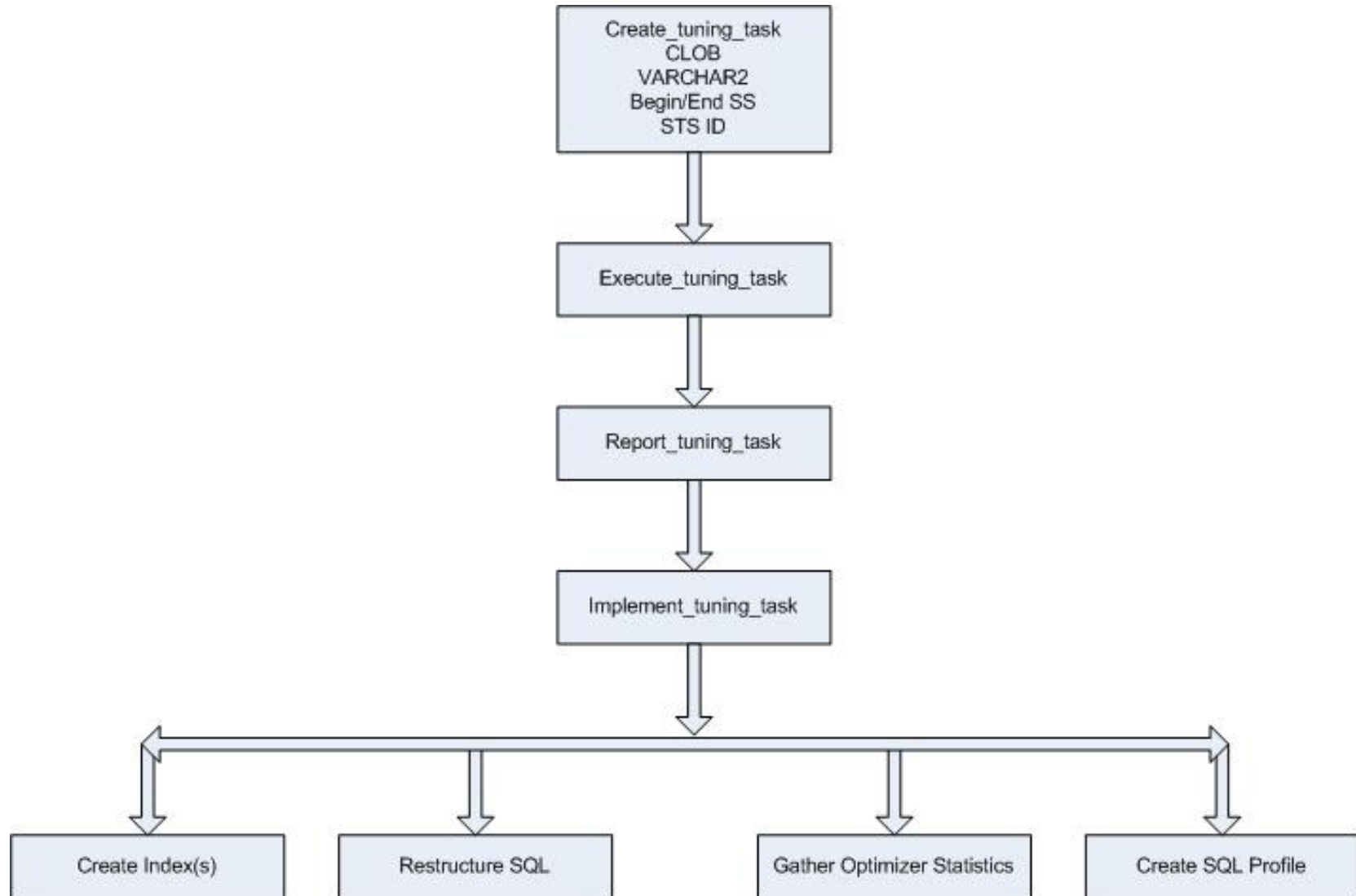


# Using SQL Advisor

- Enterprise Manager (Database Control)
  - Basically a point and click interface
  - Gives results per statement
  - Allows choice of quick or more complete analysis
  - Allows you the choice of implementing results
- SQL Advisor APIs
  - Manual control via DBA\_SQLTUNE
  - Can be called from other tools



# Manual SQL Advisor



# SQL Advisor and OEM



# SQL Advisor and OEM

Instance Throughput Rate  Per Second  Per Transaction

## Additional Monitoring Links

- [Historical SQL \(AWR\)](#)
- [Instance Activity](#)
- [Baseline Normalized Metrics](#)
- [Snapshots](#)
- [SQL Tuning Sets](#)

[Home](#) | [Performance](#) | [Administration](#) | [Maintenance](#)

## Related Links

<a href="#">Access</a>	<a href="#">Advisor Central</a>	<a href="#">Alert History</a>
<a href="#">Alert Log Content</a>	<a href="#">All Metrics</a>	<a href="#">Blackouts</a>
<a href="#">Deployments</a>	<a href="#">Execute SQL</a>	<a href="#">Jobs</a>
<a href="#">Metric and Policy Settings</a>	<a href="#">Metric Baselines</a>	<a href="#">Metric Collection Errors</a>
<a href="#">Monitoring Configuration</a>	<a href="#">Reports</a>	<a href="#">SQL History</a>
<a href="#">Target Properties</a>	<a href="#">User-Defined Metrics</a>	

[Home](#) | [Targets](#) | [Deployments](#) | [Alerts](#) | [Compliance](#) | [Jobs](#) | [Reports](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

# SQL Advisor and OEM

## Advisor Central

Page Refreshed Sep 13, 2007 1:41:28 PM EDT [Refresh](#)

### Advisors

[ADDM](#)  
[Segment Advisor](#)  
[Undo Management](#)

[Memory Advisor](#)  
[SQL Access Advisor](#)

[MTTR Advisor](#)  
[SQL Tuning Advisor](#)

### Advisor Tasks

[Change Default Parameters](#)

### Search

Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type	Task Name	Advisor Runs	Status	
All Types		Last Run	All	<a href="#">Go</a>

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

### Results

[View Result](#) [Delete](#) [Actions](#) [Re-schedule](#) [Go](#)

# SQL Advisor and OEM

## SQL Tuning Advisor Links

---

The SQL Tuning Advisor analyzes individual SQL statements and makes recommendations for improving their performance which will lead you to a data source where you can tune SQL statements using the SQL Tuning Advisor.

[Top Activity](#)

[Historical SQL \(AWR\)](#)

[SQL Tuning Sets](#)

[Snapshots](#)

[Preserved Snapshot Sets](#)

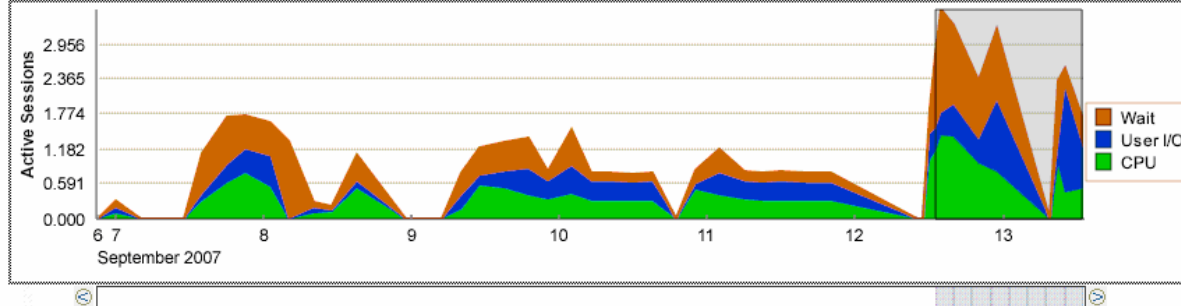
---

# SQL Advisor and OEM

## Historical SQL (AWR)

### Historical Interval Selection

Drag the shaded box to select the historical 24 hour interval for which you want to view data in the graphs below. Use the active sessions data to help with your selection.



### Detail for Selected 24 Hour Interval

Start Time **Sep 12, 2007 12:00:03 PM**

[Schedule SQL Tuning Advisor](#)

[Create SQL Tuning Set](#)

Previous

1-25 of 423

Next 25

[Select All](#) | [Select None](#)

Select	SQL ID	SQL Text	% of Total Elapsed Time	CPU Time (seconds)	Wait Time (seconds)	Elapsed Time Per Execution (seconds)	Module
<input type="checkbox"/>	27uudvf522jn6	SELECT s_name, s_address FROM	3.42	508.17	17114.33	8811.25	Agent.exe
<input type="checkbox"/>	3thb7by71m9n9	SELECT NATION, O_YEAR, SUM(AMO	2.68	7604.26	6190.96	13795.22	Agent.exe
<input type="checkbox"/>	btvxqt6vgfxzf	CREATE INDEX TPCH_LINEITEM_IDX	1.97	5291.82	4845.71	0.00	SQL*Plus
<input type="checkbox"/>	3thb7by71m9n9	SELECT NATION, O_YEAR, SUM(AMO	1.88	6055.42	3615.97	9671.39	Agent.exe
<input type="checkbox"/>	7zcfxggv196w2	SELECT s_name, count(*) numwai	1.86	3344.52	6250.80	9595.32	Agent.exe
<input type="checkbox"/>	7am96h8dkafxh	CREATE INDEX TPCH_LINEITEM_IDX	1.41	4242.73	3016.62	0.00	SQL*Plus
<input type="checkbox"/>	8zbx4z3qt2f3k	CREATE INDEX TPCH_H_LINEITEM_I	1.04	2616.76	2715.80	0.00	SQL*Plus
<input type="checkbox"/>	3yww31hvcm1xj	CREATE INDEX TPCH_H_LINEITEM_I	0.88	2162.99	2382.18	0.00	SQL*Plus
<input type="checkbox"/>	gbznrby7q10ya	SELECT O_YEAR, SUM(DECODE(NATI	0.69	1017.11	2531.18	1774.15	Agent.exe
<input type="checkbox"/>	gv5f67vn67tm	SELECT s_acctbal, s_name, n_na	0.52	215.48	2472.67	1344.08	Agent.exe
<input type="checkbox"/>	crzmannn5pzj5	/* SQL Analyze(141,1) */ selec	0.46	923.38	1461.62	0.00	DBMS_SCHEDULER

# SQL Advisor and OEM

Start Time **Sep 12, 2007 12:00:03 PM**

Schedule SQL Tuning Advisor

Create SQL Tuning Set

Previous

1-25 of 423

Next 25

Select All | Select None

Select	SQL ID	SQL Text	% of Total Elapsed Time	CPU Time (seconds)	Wait Time (seconds)	Elapsed Time Per Execution (seconds)	Module
<input checked="" type="checkbox"/>	27uudv522jn6	SELECT s_name,s_address FROM	3.42	508.17	17114.33	8811.25	Agent.exe
<input checked="" type="checkbox"/>	3thb7by71m9n9	SELECT NATION,O_YEAR,SUM(AMO	2.68	7604.26	6190.96	13795.22	Agent.exe
<input type="checkbox"/>	btvxqt6vgfxzf	CREATE INDEX TPCH.LINEITEM_IDX	1.97	5291.82	4845.71	0.00	SQL*Plus
<input checked="" type="checkbox"/>	3thb7by71m9n9	SELECT NATION,O_YEAR,SUM(AMO	1.88	6055.42	3615.97	9671.39	Agent.exe
<input checked="" type="checkbox"/>	7zcfxggv196w2	SELECT s_name,count(*) numwai	1.86	3344.52	6250.80	9595.32	Agent.exe
<input type="checkbox"/>	7am96h8dkafxh	CREATE INDEX TPCH.LINEITEM_IDX	1.41	4242.73	3016.62	0.00	SQL*Plus
<input type="checkbox"/>	8zbx4z3qt2f3k	CREATE INDEX TPCH.H_LINEITEM_I	1.04	2616.76	2715.80	0.00	SQL*Plus
<input type="checkbox"/>	3yww31hvcmlxj	CREATE INDEX TPCH.H_LINEITEM_I	0.88	2162.99	2382.18	0.00	SQL*Plus
<input checked="" type="checkbox"/>	gbznrby7q10ya	SELECT O_YEAR,SUM(DECODE(NATI	0.69	1017.11	2531.18	1774.15	Agent.exe
<input checked="" type="checkbox"/>	gv5r67vv67tm	SELECT s_acctbal,s_name,n_na	0.52	215.48	2472.67	1344.08	Agent.exe
<input type="checkbox"/>	crzmannn5pzj5	/* SQL Analyze(141,1) */ selec	0.46	923.38	1461.62	0.00	DBMS_SCHEDULER
<input checked="" type="checkbox"/>	7jm6vz9ggzxmj	SELECT c_name,c_custkey,o_orde	0.34	1173.87	558.78	866.32	Agent.exe
<input type="checkbox"/>	fa7zv72gg4wgu	/* SQL Analyze(141,1) */ selec	0.31	444.70	1168.44	0.00	DBMS_SCHEDULER
<input checked="" type="checkbox"/>	4qz3hwjc20s4w	SELECT SUM(l_extendedprice * l	0.19	17.62	983.46	500.54	Agent.exe
<input type="checkbox"/>	9h3g37134d0b4	CREATE INDEX TPCH.H_PART_IDX1	0.18	469.63	441.53	0.00	SQL*Plus

# SQL Advisor and OEM

## Create SQL Tuning Set

Cancel OK

\* Name

Description

SQL ID	SQL Text	% of Total Elapsed Time ▾	CPU Time (seconds)	Wait Time (seconds)	Elapsed Time Per Execution (seconds)	Module
27uudvf522jn6	SELECT s_name, s_address FROM	3.42	508.17	17114.33	8811.25	Agent.exe
3thb7by71m9n9	SELECT NATION, O_YEAR, SUM(AMO	2.68	7604.26	6190.96	13795.22	Agent.exe
3thb7by71m9n9	SELECT NATION, O_YEAR, SUM(AMO	1.88	6055.42	3615.97	9671.39	Agent.exe
7zcfxggv196w2	SELECT s_name, count(*) numwai	1.86	3344.52	6250.80	9595.32	Agent.exe
gbznrby7q10ya	SELECT O_YEAR, SUM(DECODE(NATI	0.69	1017.11	2531.18	1774.15	Agent.exe
gv5r67vv67tm	SELECT s_acctbal, s_name, n_na	0.52	215.48	2472.67	1344.08	Agent.exe
7jm6vz9ggzcmp	SELECT c_name, c_custkey, o_orde	0.34	1173.87	558.78	866.32	Agent.exe
4qz3hwjc20s4w	SELECT SUM(l_extendedprice * l	0.19	17.62	983.46	500.54	Agent.exe
g9x13g0jykubf	SELECT 100.00 * SUM(CASE W	0.11	138.83	438.42	288.63	Agent.exe
0pxv6xvm5vbbw	SELECT SUM(l_extendedprice)/Z.	0.03	5.10	157.42	162.52	Agent.exe

Cancel OK

[Home](#) | [Targets](#) | [Deployments](#) | [Alerts](#) | [Compliance](#) | [Jobs](#) | [Reports](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

# SQL Advisor and OEM



## Update Message

The SQL Tuning Set was created successfully.

## SQL Tuning Sets

Switch Database Instance

Page Refreshed Sep 13, 2007 1:44:16 PM EDT

A SQL Tuning Set is a collection of SQL Statements that can be used for tuning purposes.

Search

Filter on a name or partial name

Create SQL Tuning Set From

Select	Name	Schema	Description	Created	Last Modified
<input checked="" type="radio"/>	<a href="#">TOP_SQL_1189705414392</a>	SYSTEM	tpch_set	Sep 13, 2007 1:44:15 PM	Sep 13, 2007 1:44:16 PM

[Home](#) | [Targets](#) | [Deployments](#) | [Alerts](#) | [Compliance](#) | [Jobs](#) | [Reports](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

# SQL Advisor and OEM

Cluster: crs > Cluster Database: ault11g > Database Instance: ault11g\_ault11g1 >

Logged in As SYSTEM

## Schedule SQL Tuning Advisor

Enter the start date and time for the run of the advisor. A database job will be submitted at the time. You can also limit the amount of time for the run of the advisor. After reaching this limit, the advisor run will be interrupted and return partial results. You can check the status of any advisor run through Advisor Central.

\* Name   
Description   
Tuning Set Owner **SYSTEM**  
Tuning Set Name **TOP\_SQL\_1189705414392**  
Tuning Set Description **tpch\_set**

### SQL Statements

Previous 1-5 of 10 Next 5

SQL Text	Parsing Schema
SELECT SUM(l_extendedprice)/7.0 AVG_YEARLY FROM H_Lineitem, H_Part WHERE p_partkey = l_partkey AND p_brand = 'Brand#30' AND p_container = 'JUMBO CASE' AND l_quantity < (SELECT 0.2 * AVG(l_quant...	TPCH
SELECT s_name, s_address FROM H_Supplier, H_Nation WHERE s_suppkey in (SELECT ps_suppkey FROM H_Partsupp WHERE ps_partkey in (SELECT p_partkey FROM H_Part WHERE p_name like 'dark%') AND ps_availqty >...	TPCH
SELECT NATION, O_YEAR, SUM(AMOUNT) SUM_PROFIT FROM (SELECT n_name NATION, to_number(TO_CHAR(o_orderdate, 'YYYY')) AS O_YEAR, l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity AS ...	TPCH
SELECT NATION, O_YEAR, SUM(AMOUNT) SUM_PROFIT FROM (SELECT n_name NATION, to_number(TO_CHAR(o_orderdate, 'YYYY')) AS O_YEAR, l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity AS ...	TPCH
SELECT SUM(l_extendedprice * l_discount) REVENUE FROM H_Lineitem WHERE l_shipdate >= to_date('1997-01-01', 'YYYY-MM-DD') AND l_shipdate < add_months(to_date('1997-01-01', 'YYYY-MM-DD'), 12) ...	TPCH

### Scope

- Limited. Analysis without SQL Profile recommendation. Takes about 1 second per statement.
- Comprehensive. Complete analysis including SQL Profile. May take a long time.

Total Time Limit (minutes)

### Schedule

Time Zone

- Immediately
- Later

Date    
(example: Sep 13, 2007)

Time     AM  PM

# SQL Advisor and OEM

Logged in As SYSTEM

Cancel

SQL Tuning Advisor task is being submitted. This can take a while. Press Cancel to return to the previous page. The SQL Tuning Advisor task will continue to execute. You can check its status and view the recommendations from Advisor Central page.



Cancel

[Home](#) | [Targets](#) | [Deployments](#) | [Alerts](#) | [Compliance](#) | [Jobs](#) | [Reports](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

Copyright © 1996, 2007, Oracle. All rights reserved.

Oracle, JD 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](#)

# SQL Advisor and OEM

Logged in as SYS (LIM)

Cancel

SQL Tuning Advisor task is being submitted. This can take a while. Press Cancel to return to the previous page. The SQL Tuning Advisor task will continue to execute. You can check its status and view the recommendations from Advisor Central page.



- ✓ Creating a new SQL Tuning task
- ➔ Executing the task

Cancel

[Home](#) | [Targets](#) | [Deployments](#) | [Alerts](#) | [Compliance](#) | [Jobs](#) | [Reports](#) | [Setup](#) | [Preferences](#) | [Help](#) | [Logout](#)

Copyright © 1996, 2007, Oracle. All rights reserved.

Oracle, JD 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](#)

# SQL Advisor and OEM

## Advisor Central

Page Refreshed Sep 13, 2007 1:46:52 PM EDT [Refresh](#)

### Advisors

[ADDM](#)  
[Segment Advisor](#)  
[Undo Management](#)

[Memory Advisor](#)  
[SQL Access Advisor](#)

[MTR Advisor](#)  
[SQL Tuning Advisor](#)

### Advisor Tasks

[Change Default Parameters](#)

### Search

Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type Task Name Advisor Runs Status

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

### Results

Select	Advisory Type	Name	Instance Number	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)
<input checked="" type="radio"/>	SQL Tuning Advisor	<a href="#">SQL_TUNING_1189705496992</a>	UNUSED	tpch_tuning_run	SYSTEM	RUNNING	Sep 13, 2007 1:45:29 PM		30
<input type="radio"/>	ADDM	<a href="#">ADDM:2838543632_547</a>	UNUSED	ADDM auto run: snapshots [546, 547], , database id 2838543632	SYS	COMPLETED	Sep 13, 2007 1:00:30 PM	1	30
<input type="radio"/>	ADDM	<a href="#">ADDM:2838543632_1_547</a>	1	ADDM auto run: snapshots [546, 547], instance 1, database id 2838543632	SYS	COMPLETED	Sep 13, 2007 1:00:30 PM	0	30
<input type="radio"/>	Segment Advisor	<a href="#">SYS_AUTO_SPCADV_20021392007</a>	UNUSED	Auto Space Advisor	SYS	RUNNING	Sep 12, 2007 10:00:23 PM		29

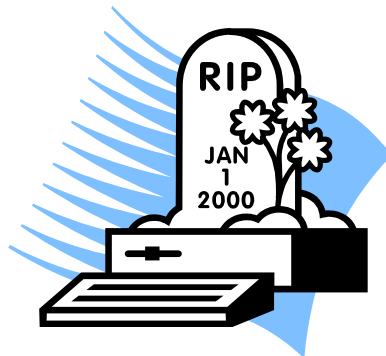
# SQL Advisor and OEM

- Upon which using XP SP2 with 10.2.0.3 of OEM I got....

SQL Advisor and OEM

# BLUE SCREEN OF DEATH

## BSODI



# SQL Advisor and OEM

**After a few choice curse words and a restart...**



# SQL Advisor and OEM

## Advisor Central

Page Refreshed Sep 14, 2007 6:40:02 PM EDT [Refresh](#)

### Advisors

[ADDM](#)  
[Segment Advisor](#)  
[Undo Management](#)

[Memory Advisor](#)  
[SQL Access Advisor](#)

[MTTR Advisor](#)  
[SQL Tuning Advisor](#)

### Advisor Tasks

[Change Default Parameters](#)

### Search

Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type Task Name Advisor Runs Status

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

### Results

Select	Advisory Type	Name	Instance Number	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)
<input checked="" type="radio"/>	ADDM	<a href="#">ADDM:2838543632_1_576</a>	1	ADDM auto run: snapshots [575, 576], instance 1, database id 2838543632	SYS	COMPLETED	Sep 14, 2007 6:00:23 PM	0	30
<input type="radio"/>	ADDM	<a href="#">ADDM:2838543632_576</a>	UNUSED	ADDM auto run: snapshots [575, 576], , database id 2838543632	SYS	COMPLETED	Sep 14, 2007 6:00:23 PM	0	30
<input type="radio"/>	SQL Tuning Advisor	<a href="#">SQL_TUNING_1189705496992</a>	UNUSED	tpch_tuning_run	SYSTEM	INTERRUPTED	Sep 14, 2007 1:30:00 AM	1801	29
<input type="radio"/>	Segment Advisor	<a href="#">SYS_AUTO_SPCADV_441721492007</a>	UNUSED	Auto Space Advisor	SYS	COMPLETED	Sep 13, 2007 10:17:44 PM	60	29

# SQL Advisor and OEM

## SQL Tuning Results:SQL\_TUNING\_1189705496992

### Update Message

The task has not completed

Page Refreshed Sep 14, 2007 6:42:59 PM

Status **INTERRUPTED**  
Started **Sep 14, 2007 1:30:00 AM**  
Completed **Sep 14, 2007 2:00:01 AM**

Tuning Set Owner **SYSTEM**  
Tuning Set Name **TOP\_SQL\_1189705414392**  
Time Limit (seconds) **1800**  
Running Time (seconds) **1801**

### Recommendations

Select SQL Text	Parsing Schema	SQL ID	Statistics	SQL Profile	Index	Restructure SQL	Miscellaneous Error
<input checked="" type="radio"/> SELECT s_name, s_address FROM H_Supplier, H_Nation WHERE s_suppkey in (SELECT ps_suppkey FROM H_Part...	TPCH	<a href="#">27uudv522jn6</a>			✓		✓

# SQL Advisor and OEM

## Recommendations for SQL ID:27uudvf522jn6

[Return](#)

Only one recommendation should be implemented.

### SQL Text

`SELECT s_name, s_address FROM H_Supplier, H_Nation WHERE s_suppkey in (SELECT ps_suppkey FROM H_Partsupp WHERE ps_partkey in ( SELECT p_partkey FROM H_Part WHERE p_name like 'dark%') AND ps_availqty >...`

### Select Recommendation

[Original Explain Plan \(Annotated\)](#)

[Implement](#)

Select	Type	Findings	Recommendations	Rationale	Benefit (%)	New Explain Plan	Compare Explain Plans
<input checked="" type="radio"/>	Index	The execution plan of this statement can be improved by creating one or more indices.	Consider running the Access Advisor to improve the physical schema design or creating the recommended index. TPCH.H_LINEITEM ("L_SUPPKEY", "L_SHIPDATE") TPCH.H_PART("P_NAME") TPCH.H_LINEITEM ("L_PARTKEY", "L_SUPPKEY") TPCH.H_NATION("N_NAME")	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.	99.96		
<input type="radio"/>	Miscellaneous	The optimizer could not merge the view at line ID 8 of the execution plan.					
<input type="radio"/>	Error	The current operation was interrupted because it timed out.					

[Return](#)

# SQL Advisor and OEM

## Compare Explain Plans

Line ID	Object	Object Type	Object Node	Order	Rows	Size (KB)	Cost	Time (sec)	CPU Cost	I/O Cost
0					32	1 0.048	8	1 61,017	8	
1					31					
2	SYS::TQ10003		:Q1003	30	1 0.048	8	1 61,017	8		
3			:Q1003	29	1 0.048	8	1 61,017	8		
4	<u>SYS.VM_NWWW_3</u>	VIEW	:Q1003	28	1 0.048	8	1 61,017	8		
5			:Q1003	27	1 0.301	8	1 61,017	8		
6			:Q1003	26	1 0.301	8	1 61,017	8		
7	SYS::TQ10002		:Q1002	25	1 0.301	8	1 61,017	8		
8			:Q1002	24						
9			:Q1002	23	1 0.301	8	1 61,017	8		
10			:Q1002	22	1 0.301	8	1 61,017	8		
11	SYS::TQ10001		:Q1001	21	1 0.301	8	1 61,017	8		
12			:Q1001	20	1 0.301	8	1 61,017	8		
13			:Q1001	19						
14			:Q1001	17	1 0.301	8	1 60,929	8		
15			:Q1001	15	1 0.267	8	1 56,779	8		
16			:Q1001	12	4 0.859	7	1 52,144	7		
17			:Q1001	9	4 0.348	6	1 43,060	6		
18			:Q1001	5						
19			:Q1001	4						
20	SYS::TQ10000				3					
21	<u>TPCH.H_PART</u>	TABLE			2	1 0.050	5	1 36,347	5	
22	IDX\$\$_07420002	INDEX			1	1	4	1 28,886	4	
23			:Q1001	8	4 0.148	1	1 6,712	1		

Object	Object Type	Object Node	Order	Rows	Size (KB)	Cost	Time (sec)	CPU Cost	I/O Cost
				30	1 0.094	25,896	311	1,533,713,024	25,795
				29					
SYS::TQ10003		:Q1003	28	1 0.094	25,896	311	1,533,713,024	25,795	
		:Q1003	27	1 0.094	25,896	311	1,533,713,024	25,795	
		:Q1003	26	1 0.094	25,896	311	1,516,875,520	25,795	
SYS::TQ10002		:Q1002	25	1 0.094	25,896	311	1,516,875,520	25,795	
		:Q1002	24	1 0.094	25,896	311	1,516,875,520	25,795	
		:Q1002	22	1 0.065	25,894	311	1,516,859,904	25,794	
SYS.VW_NSO_2	VIEW	:Q1002	19	1 0.006	25,893	311	1,500,021,248	25,794	
		:Q1002	18	1 0.121	25,893	311	1,500,021,248	25,794	
		:Q1002	17	1 0.121	25,893	311	1,500,021,248	25,794	
SYS::TQ10001		:Q1001	16	1 0.121	25,893	311	1,500,021,248	25,794	
		:Q1001	15						
		:Q1001	14	1 0.121	25,893	311	1,500,021,248	25,794	
		:Q1001	13	1 0.121	25,893	311	1,500,021,248	25,794	
SYS::TQ10000		:Q1000	12	1 0.121	25,893	311	1,500,021,248	25,794	
		:Q1000	11	1 0.121	25,893	311	1,500,021,248	25,794	
		:Q1000	10						
		:Q1000	8	2 0.242	25,893	311	1,500,021,120	25,794	
		:Q1000	6	4 0.348	25,891	311	1,500,003,328	25,792	
		:Q1000	2	1 0.050	25,890	311	1,500,000,000	25,791	
<u>TPCH.H_PART</u>	TABLE	:Q1000	1	1 0.050	25,890	311	1,500,000,000	25,791	
		:Q1000	5	4 0.148	1	1	6,712	1	
<u>TPCH.H_PARTSUPP</u>	TABLE	:Q1000	4	4 0.148	1	1	6,712	1	



## So....

- SQL Advisor can help you with tuning
- You must throw lots of resources at it
- Try not to do complex analysis during production
- It may not give you everything you want
- As implemented in 10.2.0.3 for 11g there may be some problems...BEWARE
- 11g R1 of OEM promised with Release 2 of 11g

# Let's Look at a Test Case!



## Using OEM...OOPS!

- SQL Replay sounds good in theory but I found the practice a bit harder.
- It would have been easier had I been able to use the wizards to perform the SQL Replay,
- During the last step of the install the Database Control on the Windows XP release refused to startup properly, complained about security certificates and fell over.
- I ended up doing my testing in manual using the PL/SQL APIs provided.

## SQL Replay Requirements

- You must first have the SQL you want to replay, any bind variables needed and be able to recreate the needed environment.
- The SQL statements must be contained in a STS and have the needed bind variables assigned.
- You can build the STS statement by statement manually and then use the API to populate the STS bind variables.
- My masochistic tendencies only reach so deep, I opted to use the option to collect the needed data from the AWR datasets.

## Collecting the SQL

So the first step in producing a SQL Replay is to create the STS:

```
SQL> set echo on  
SQL> exec dbms_sqltune.create_sqlset('tpcc_workload');  
PL/SQL procedure successfully completed.
```

Gee that was easy! Oracle knew automatically what SQL I needed and....no...not in your wildest dreams.

The above just creates a parking space for the STS to reside in.

## Now Populate the STS

So populate the STS with SQL and needed bind variable data:

```
SQL> declare
2 cur dbms_sqltune.sqlset_cursor;
3 begin
4 open cur for select value(p)
5 from table( dbms_sqltune.select_cursor_cache(
6 'parsing_schema_name = "TPCC" ',null, null, null, null,
7 1, null, 'ALL')) P;
8 dbms_sqltune.load_sqlset(sqlset_name=>'tpcc_workload',
9 populate_cursor=>cur);
10* end;
SQL> /
PL/SQL procedure successfully completed.
```

## Results

Well, that wasn't so hard. However, this was a brute force operation that grabbed every statement that was parsed by the TPCC user. So while there were maybe 20 statements I was interested in, the process actually grabbed:

```
SQL> select count(*) from  
table(dbms_sqltune.select_sqlset('tpcc_workload',  
'parsing_schema_name = "TPCC" '));
```

```
Count(*)
```

```
-----  
          36
```

## Pruning STS

In Database Control or in OEM we probably could have done this visually, I had to use the provided API calls (after reviewing what statements I wanted pruned):

```
SQL> exec  
dbms_sqltune.delete_sqlset('tpcc_workload','sql_text like  
"%OPT_DYN_SAMP%");  
PL/SQL procedure successfully completed.
```

Depending on the SQL I needed to prune I used a statement similar to the one above to remove the offending statements.

## First, do a Advisor Run

Now generate a SQL Advisor output so you know what needs tuning:

```
SQL> declare
```

```
 2 my_task_name varchar2(30);
```

```
 3 begin
```

```
 4 my_task_name :=
```

```
dbms_sqltune.create_tuning_task(sqlset_name=>'tpcc_work  
load',
```

```
 5 time_limit=>3600,
```

```
 6 scope=>'COMPREHENSIVE',
```

```
 7 task_name=>'tpcc_tuning_task',
```

```
 8 description=>'tpcc tuning task');
```

```
 9* end;
```

```
SQL> /
```

PL/SQL procedure successfully completed.

## Reset Some of the Parameters, Execute

```
SQL> begin
  2 dbms_sqltune.set_tuning_task_parameter(
  3 task_name=> 'tpcc_tuning_task',
  4 parameter => 'TIME_LIMIT', value=>7200);
  5 end;
  6 /
```

PL/SQL procedure successfully completed.

```
SQL> begin
  2 dbms_sqltune.execute_tuning_task(task_name =>
'tpcc_tuning_task');
  3 end;
  4 /
```

PL/SQL procedure successfully completed.

# Tuning Report Header

```
SQL> select dbms_sqltune.report_tuning_task('tpcc_tuning_task')
       2 from dual;
```

```
DBMS_SQLTUNE.REPORT_TUNING_TASK('TPCC_TUNING_TASK')
```

```
-----
GENERAL INFORMATION SECTION
```

```
-----Tuning
Task Name                : tpcc_tuning_task
Tuning Task Owner        : SYSTEM
Workload Type            : SQL Tuning Set
```

```
...
```

```
Number of Statements in the STS : 23
```

```
-----
Global SQL Tuning Result Statistics
```

```
-----
Number of SQLs Analyzed : 23
```

```
SQL> select count(*) from dba_advisor_findings;
```

```
COUNT(*)
```

```
-----
282
```

## Looking at Findings

Let's restrict the number of returned findings by the task name:

```
SQL> select message from dba_advisor_findings where  
task_name='tpcc_tuning_task';  
MESSAGE
```

```
-----  
A potentially better execution plan was found for this statement.  
Table "TPCC"."C_CUSTOMER" was not analyzed.  
Table "TPCC"."C_ORDER" was not analyzed.  
Table "TPCC"."C_NEW_ORDER" was not analyzed.  
Table "TPCC"."C_CUSTOMER" was not analyzed.  
Table "TPCC"."C_STOCK" was not analyzed.  
Table "TPCC"."C_ITEM" was not analyzed.  
An expensive "UNION" operation was found at line ID 2 of the execution plan.  
The optimizer could not merge the view at line ID 1 of the execution plan.  
Table "TPCC"."C_WAREHOUSE" was not analyzed.  
Table "TPCC"."C_DISTRICT" was not analyzed.  
45 rows selected.
```

SO...DBMS\_STATS for a start

## New Findings

After generating the statistics I reran the analysis and got:

A potentially better execution plan was found for this statement.

The execution plan of this statement can be improved by creating one or more indices.

An expensive "UNION" operation was found at line ID 2 of the execution plan.

The optimizer could not merge the view at line ID 1 of the execution plan.

## Run First Stage Analysis

```
SQL> variable t_name varchar2(100);
SQL> exec :t_name :=
dbms_sqlpa.create_analysis_task(sqlset_name =>
'tpcc_workload', task_name => 'tpcc_sql_replay');
PL/SQL procedure successfully completed.
SQL> begin
  2  dbms_sqlpa.execute_analysis_task(task_name =>
'tpcc_sql_replay',
  3  execution_type => 'TEST EXECUTE',
  4  execution_name => 'before_change_tpcc');
  5  end;
  6  /
PL/SQL procedure successfully completed.
```

## Second Stage

I then created the indexes needed and reanalyzed the schema, after which I re-ran the STS through the second stage of analysis:

```
SQL> begin
  2  dbms_sqlpa.execute_analysis_task(task_name =>
    'tpcc_sql_replay',
  3  execution_type => 'TEST EXECUTE',
  4  execution_name => 'after_change_tpcc');
  5* end;
SQL> /
PL/SQL procedure successfully completed.
```

## Finally!

Ok, finally after a couple of hours effort I could finally run the report that would tell me if I improved my SQL or not:

```
SQL> exec :rep :=
dbms_sqlpa.report_analysis_task('tpcc_sql_replay', -
> 'text', 'all', 'all');
PL/SQL procedure successfully completed.
SQL> set long 100000 longchunksize 100000 linesize 130
SQL> print :rep
```

# Header

## General Information

---

### Task Information:

Task Name : tpcc\_sql\_replay  
Task Owner : SYSTEM  
Description :

### Workload Information:

SQL Tuning Set Name : tpcc\_workload  
SQL Tuning Set Owner : SYSTEM  
Total SQL Statement Count : 23

### Execution Information:

---

Execution Name	: tpcc_compare	Started	: 09/21/2007 09:54:37
Execution Type	: COMPARE PERFORMANCE	Last Updated	: 09/21/2007 09:54:37
Description	:	Global Time Limit	: UNLIMITED
Scope	: COMPREHENSIVE	Per-SQL Time Limit	: UNUSED
Status	: COMPLETED	Number of Errors	: 0

# Analysis Section

## Analysis Information:

-----  
Comparison Metric: ELAPSED\_TIME  
-----

Workload Impact Threshold: 1%

SQL Impact Threshold: 1%  
-----

### Before Change Execution:

-----  
Execution Name : before\_change\_tpcc  
Execution Type : TEST EXECUTE  
Description :  
Scope : COMPREHENSIVE  
Status : COMPLETED  
Started : 09/21/2007 09:09:57  
Last Updated : 09/21/2007 09:10:08  
Global Time Limit : UNLIMITED  
Per-SQL Time Limit : UNUSED  
Number of Errors : 0

### After Change Execution:

-----  
Execution Name : after\_change\_tpcc  
Execution Type : TEST EXECUTE  
Description :  
Scope : COMPREHENSIVE  
Status : COMPLETED  
Started : 09/21/2007 09:52:04  
Last Updated : 09/21/2007 09:52:11  
Global Time Limit : UNLIMITED  
Per-SQL Time Limit : UNUSED  
Number of Errors : 0

# Summary

## Report Summary

---

### Projected Workload Change Impact:

---

Overall Impact : 50.21%  
Improvement Impact : 50.21%  
Regression Impact : 0%

## SQL Statement Count

---

SQL Category	SQL Count	Plan Change Count
Overall	23	0
Improved	13	0
Unchanged	10	0

So, according to the report I got a 50% increase in overall performance. However, this was accomplished with no plan changes. Come again?



## Some things to remember about SQL Replay:

- Each statement is only executed once
- The database must be returned to the same state it was in before each test began (except for the test changes)
- Each statement is executed in the order they are provided in the STS.

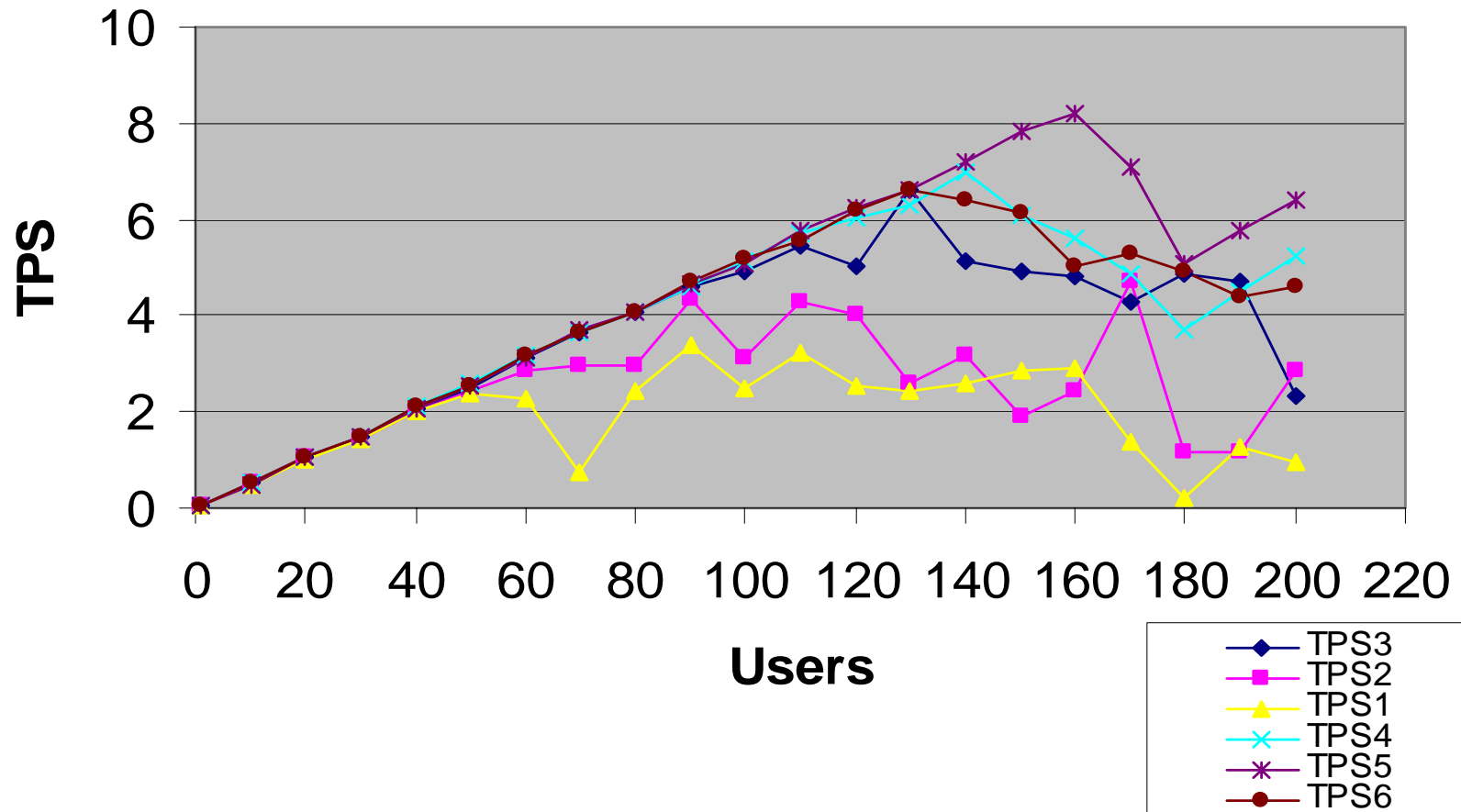
# Magic!

- How can we get a 50% increase in performance with no plan changes?
- Obviously we have some auto-tuning going on under the covers that isn't being reported or documented in these reports.
- Or...it didn't take into account the effect of caching!

## Auto Tuning?

- In fact, looking at the overall transactions per second for a 200 user load against this TPCC database we see a very interesting trend, in the next graph:
  - TPS1 is the first TPCC run with no indexes and no analysis
  - TPS2 is after the analysis but before indexes
  - TPS3-6 are post analysis and indexes, but with no additional manual tuning performed, but, the database was allowed to perform automated tuning operations between each run.

## 200 User TPS



## Interesting...

- Supposedly no new plans are implemented unless they are 3 times better performance wise than their previous incarnation.
- Yet in my testing performance improved (TPS increase) as did scalability (Number of stable users) over three days time with no manual, DBA changes implemented to the database.

# Auto Tuning

- The data in the graph was generated using the Benchmark Factory (BMF) tool with a scaling factor of 4 and a maximum user load of 200 users with default timing parameters using a standard TPCC database (also create by the BMF tool.)
- Scalability (as measured by stable, increasing performance with no large deviations) improved from 50 to 80 users with the analysis.
- Adding indexes resulted in 110, then 140 and then a peak of 160 users to act against the test database (which by the way was a Viao Laptop with 1 gigabyte of memory and a 3 GHz CPU with a 500 megabyte SGA on Oracle 11.1.0.6 running on Windows XP SP2)
- An astounding 200 percent increase in scalability by just analyzing the schema and adding a couple of indexes!

# Statistics Management

- Everything is running fine
- Then comes the nightly/weekly/monthly analyze
- POOF! Several critical SQLs performances go down the drain
- New statistics caused new execution plans...oops!
- 11g offers statistics management

## Statistics Management

- **DBMS\_STATS** has been greatly extended including **SET\*PREFS** set of procedures
- **SET\*PREFS** allows setting of **PUBLISH** to **TRUE** or **FALSE**
- **TRUE** is default
- **FALSE** places stats in **PENDING** mode
- **DIFF\_TABLE\_STATS\_IN\_PEN** allows comparisons

# Pending Statistics Views

- DBA\_TAB\_PENDING\_STATS
- DBA\_IND\_PENDING\_STATS

# Checking Status

- `Select dbms_stats.get_prefs('PUBLISH')`  
`publish from dual;`

# Setting Objects

- Exec `dbms_stats.set_table_prefs('SH', 'CUSTOMERS', 'PUBLISH', 'false');`



## Testing Pending Statistics

If you want the optimizer to use the newly collected pending statistics, set the initialization parameter:

`OPTIMIZER_PENDING_STATISTICS`

to TRUE (the default value is FALSE), and run a workload against the table or schema:

```
alter session set optimizer_pending_statistics = TRUE;
```



## Publishing Statistics

If the pending statistics are valid, they can be made public by executing the following statement:

```
Exec dbms_stats.publish_pending_stats(null, null);
```

## By Object

You can also publish the pending statistics for a specific database object by using the following statement:

Exec

```
dbms_stats.publish_pending_stats('SH','CUSTOMERS');
```

If you do not want to publish the pending statistics, delete them by executing the following statement:

Exec

```
dbms_stats.delete_pending_stats('SH','CUSTOMERS');
```

Using NULL,NULL publishes or deletes all pending statistics.



## Alternate testing

You can export pending statistics using `dbms_stats.export_pending_stats` function.

Exporting pending statistics to a test system enables you to run a full workload against the new statistics using SQL or Database replay.

# SQL Plan Management

- Baselines are used to capture plans
- The baseline plans are then used as the execution plans
- Only “repeatable” SQL plans are baselined



## SQL Plan Management

To enable automatic plan capture, set the `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES` initialization parameter to `TRUE`.

By default, this parameter is set to `FALSE`.

# Evolving SQL Plan Baselines

- When a new plan for a SQL statement is found, the plan is added as a non-accepted plan.
- The plan is then verified for performance relative to the SQL plan baseline.
- When it is verified to not cause a performance regression, it is changed to an accepted plan and integrated into the SQL plan baseline.
- A successful verification consists of comparing the performance to that of a plan selected from the SQL plan baseline and ensuring that it delivers better performance.

## DBMS\_SPM.EVOLVE\_SQL\_PLAN\_BASELINE

- Used to evolve plans
- You provide the name of the plan to use to evolve

```
SET SERVEROUTPUT ON
SET LONG 10000
DECLARE
report clob;
BEGIN
report := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
sql_handle => 'SYS_SQL_593bc74fca8e6738' );
DBMS_OUTPUT.PUT_LINE(report);
END;
/
```

The report will tell if the plan was evolved.

## Seeing Plans

To view the plans stored in the SQL plan baseline for a given statement, use the `DISPLAY_SQL_PLAN_BASELINE` function of the `DBMS_XPLAN` package:

```
select * from table(  
  dbms_xplan.display_sql_plan_baseline(  
    sql_handle=>'SYS_SQL_209d10fabbedc741',  
    format=>'basic' ));
```

# Displaying Plan Baseline Data

You can also display SQL plan baseline information using a **SELECT** statement directly on the **DBA\_SQL\_PLAN\_BASELINES** view:

```
select sql_handle, plan_name, enabled, accepted, fixed from  
dba_sql_plan_baselines;
```

SQL_HANDLE	PLAN_NAME	ENA	ACC	FIX
SYS_SQL_209d10fabbedc741	SYS_SQL_PLAN_bbedc741a57b5fc2	YES	NO	NO
SYS_SQL_209d10fabbedc741	SYS_SQL_PLAN_bbedc741f554c408	YES	YES	NO

## Plan Storage

By default uses 10% of SYSAUX

Can be changed using:

```
BEGIN  
DBMS_SPM.CONFIGURE('space_budget_percent', 30);  
END;  
/
```

Can also use a different retention period for unused plans  
(default is 53 weeks)

Questions?



# Contact Data

- [Mike.ault@quest.com](mailto:Mike.ault@quest.com)
- [www.toadworld.com](http://www.toadworld.com)
- 770-754-9057 (Office line stateside)