# The Best 11g New Features
## Collaborate 2007 (Based on 11g Beta)

*Rich Niemiec, TUSC (www.tusc.com)*

# Audience Knowledge / Versions

- Oracle7 Experience ?
- Oracle8*i* Experience ?
- Oracle9*i* Experience ?
- Oracle10*g* Experience?
- Oracle Database 11*g* Experience?

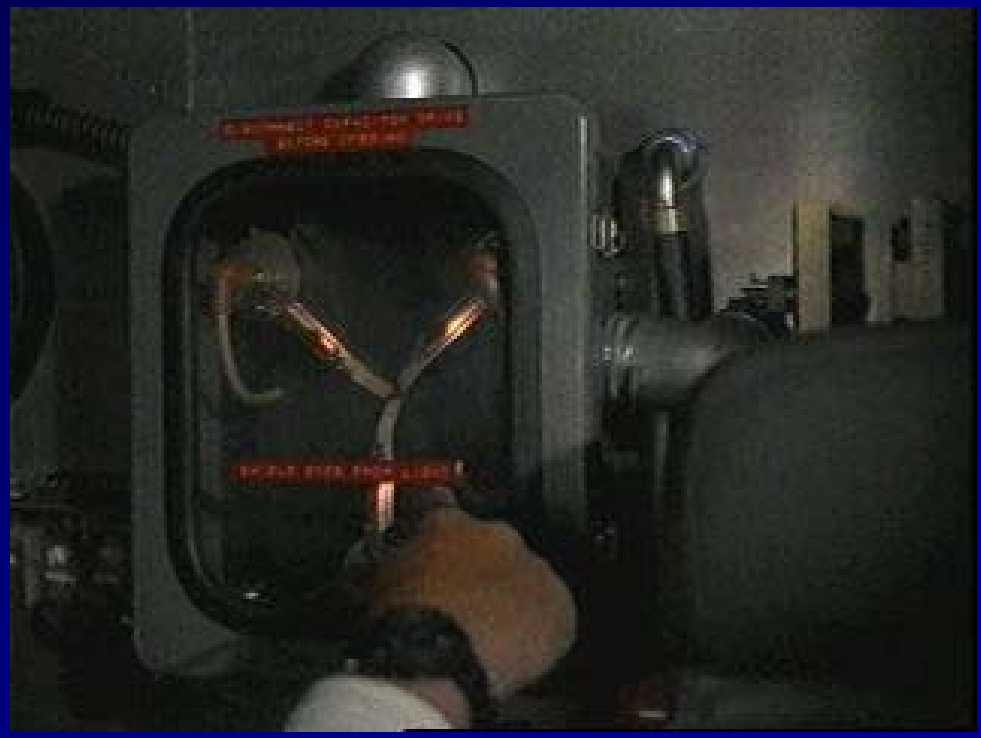

- Goals
  - Present NEW features in an EASY way
  - Focus on a few nice features of Oracle11g
- Non-Goals
  - Learn ALL aspects of Oracle11g

# Overview

- Start Me Up – Using Memory Target
- The Result Cache
- Invisible Indexes & Online Index Rebuilds
- Virtual Columns and other Nice Developer Tools
- ADDM Enhancements
- SQL Plan Management and SQL Plan Baselines
- SQL Access Advisor & Partition Advisor
- SQL Query Repair Advisor
- SQL Replay Advisor and DB Workload Capture & Replay
- Interval Partitioning & Partition Compression
- DBA Tools, ADR and DBMS_STATS Enhancements
- Grid Control & EM
- Security Enhancements & the Future Sizes
- Summary

# Testing the Future Version (Beta)

## Version 11.1.0.3.0 Beta of the Database



# The New Version – Life is Good!



$ sqlplus ***/***

**SQL*Plus: Release 11.1.0.3.0 - Beta** on Wed Mar 21 18:53:08 2007
Copyright (c) 1982, 2006, Oracle. All rights reserved.

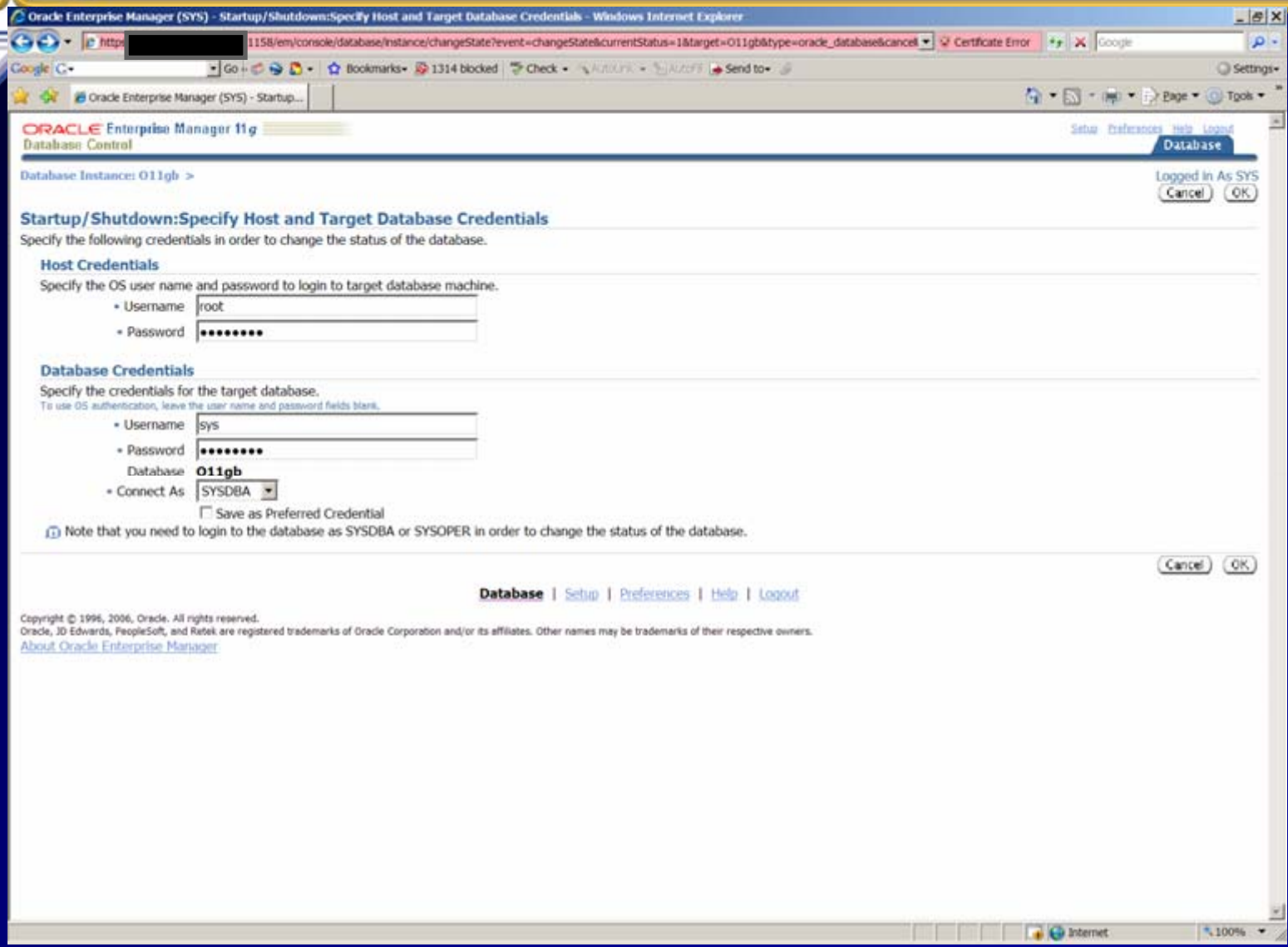
**Connected to:**
**Oracle Database 11g Enterprise Edition Release 11.1.0.3.0 - Beta**
With the Partitioning, OLAP and Data Mining options

SQL> startup
ORACLE instance started.

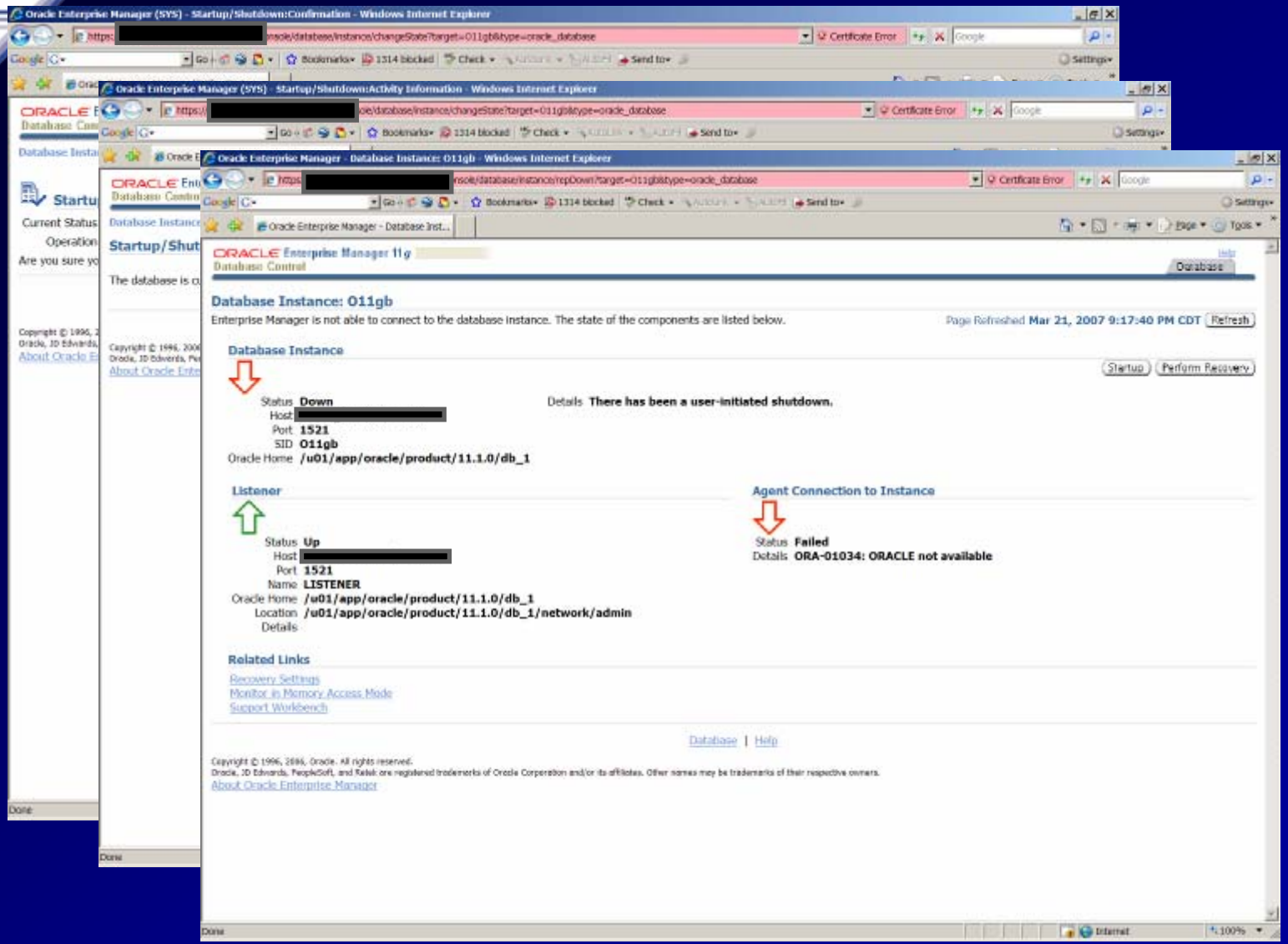
| | |
|---|---|
| Total System Global Area | 382779392 bytes |
| Fixed Size | 1297580 bytes |
| Variable Size | 363119444 bytes |
| Database Buffers | 12582912 bytes |
| Redo Buffers | 5779456 bytes |

Database mounted.
Database opened.

# Or... Use 11g EM...


Oracle Enterprise Manager (SYS) - Startup/Shutdown:Specify Host and Target Database Credentials - Windows Internet Explorer
ORACLE Enterprise Manager 11g
Database Control
Database
Database Instance: O11gb >
Logged In As SYS
Cancel
OK
Startup/Shutdown:Specify Host and Target Database Credentials
Specify the following credentials in order to change the status of the database.
Host Credentials
Specify the OS user name and password to login to target database machine.
• Username root
• Password ••••••••
Database Credentials
Specify the credentials for the target database.
To use OS authentication, leave the user name and password fields blank.
• Username sys
• Password ••••••••
Database O11gb
• Connect As SYSDBA
Save as Preferred Credential
Note that you need to login to the database as SYSDBA or SYSOPER in order to change the status of the database.
Cancel
OK
Database | Setup | Preferences | Help | Logout
Copyright © 1996, 2006, 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

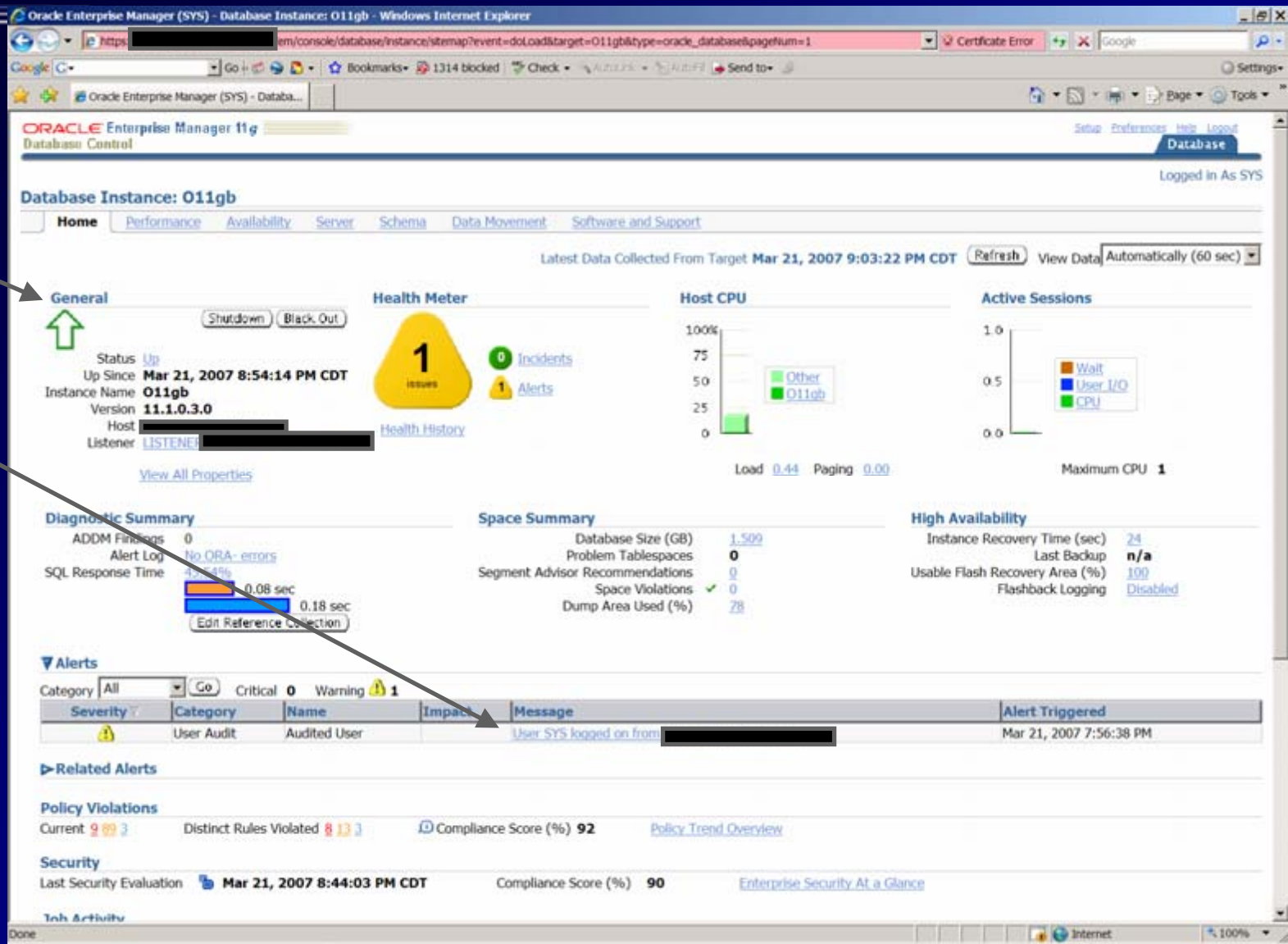
# Or... Use 11g EM...Status...



# Database Information – Back UP!

Monitor Database

We have an alert – we logged on as SYS



# MEMORY_TARGET
# &
# Automatic Memory Management

# Automatic Memory Management (AMM)
# MEMORY_TARGET in 11g



- 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)
- Now there is MEMORY_TARGET – 11g
  - SGA + PGA all in one setting; Still set minimums

# Init.ora Parameter Changes: SGA_MAX_SIZE




- Dynamic Memory allocation/resizing
  - INIT.ora changes are immediate (no shutdown)
  - Set an SGA_MAX_SIZE (varies by platform):
    - _ksm_granule_size=4M (if SGA<1G)
    - _ksm_granule_size=8M (if SGA>=1G) on Windows 32-bit
    - _ksm_granule_size=16M (if SGA>=1G)
  - SGA_MAX_SIZE is NOT dynamic!
  - MEMORY_MAX_TARGET is NOT dynamic!
- Altered Sizes of SGA components such as shared pool, buffer cache... get put into multiples of a granule size & must be <SGA_MAX_SIZE

# Setting a Minimum DB_CACHE_SIZE




- Upon Startup [K/M/G] with >1G SGA:
- ALTER SYSTEM SET DB_CACHE_SIZE=32M
  - Sets it to 32M (sets it to 33554432)
- ALTER SYSTEM SET DB_CACHE_SIZE=17M
  - Sets it to 32M (sets it to next highest multiple)
- ALTER SYSTEM SET DB_CACHE_SIZE=33M
  - Sets it to 48M (again sets it to next highest multiple of original size)
- ALTER SYSTEM SET DB_CACHE_SIZE=2M
  - Sets it to 16M (sets it to next highest/but at minimum of initial size)
- ALTER SYSTEM SET DB_CACHE_SIZE = 2000G

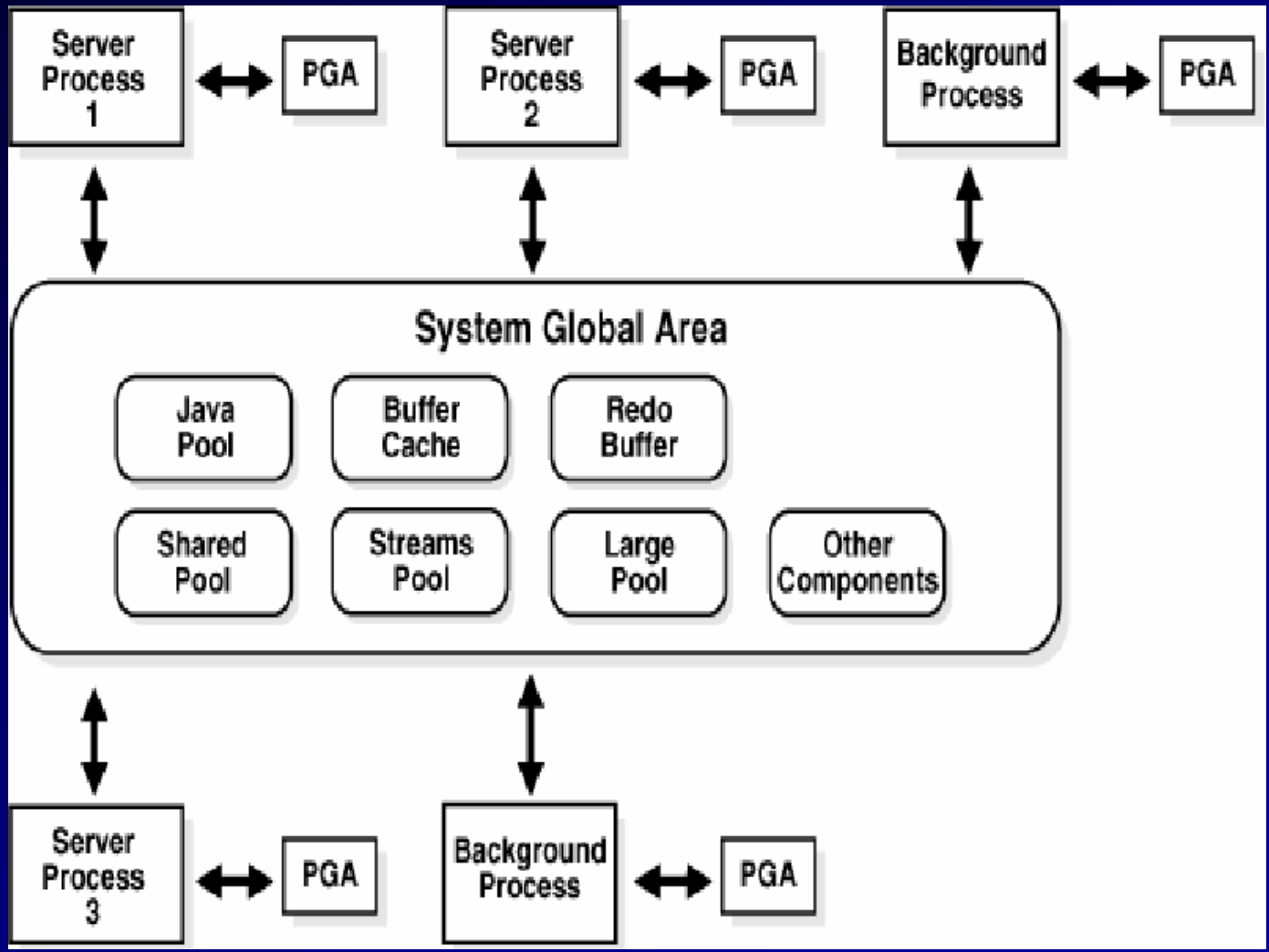
```
ALTER SYSTEM SET DB_CACHE_SIZE = 2000G
*
ERROR at line 1:
ORA-32017: failure in updating SPFILE
ORA-00384: Insufficient memory to grow cache
```

# SGA & PGA will both be in: MEMORY_TARGET

SGA: The System Global Area (SGA) are memory structures that control one instance and contain primarily the data in cache (DB_CACHE_SIZE) and statements (SHARED_POOL_SIZE).

PGA: The Program Global Area (PGA) contains the data and control information for a server PROCESS. It is non-shared memory allocated when a server process is started. Total allocated is the PGA_AGGREGATE_TARGET.

# SGA & PGA will be MEMORY_TARGET



# Automatically sized SGA Components that Use SGA_TARGET

| Component | Initialization Parameter |
|---|---|
| Fixed SGA | None |
| Shared Pool | SHARED_POOL |
| Large Pool | LARGE_POOL_SIZE |
| Java Pool | JAVA_POOL_SIZE |
| Buffer Cache | DB_CACHE_SIZE |
| Streams Pool | STREAMS_POOL_SIZE |

# Manually Sized SGA Components that Use SGA_TARGET

| Component | Initialization Parameter |
|---|---|
| Log buffer | LOG_BUFFER (pfile only in 10g) |
| Keep Pool | DB_KEEP_CACHE_SIZE |
| Recycle Pool | DB_RECYCLE_CACHE_SIZE |
| Block caches | DB_nK_CACHE_SIZE |

| Program Global Area (now in MEMORY_TARGET): | |
|---|---|
| Aggregate PGA | PGA_AGGREGATE_TARGET |

# Moving from to Manually Sizing to: SGA_TARGET (Sizing SGA)



```
SQL> COL SGA_TARGET FOR 999,999,999,999

SQL> SELECT ((select sum(value) from v$sga) -
          (select current_size from
           v$sga_dynamic_free_memory)) "SGA_TARGET"
         FROM DUAL;

SGA_TARGET
--------------------
       215,007,232
```

# Automatic Memory Management (AMM)
# MEMORY_TARGET in 11g


```
SQL> sho parameter sga_

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------
sga_max_size                         big integer 360M
sga_target                           big integer 0

SQL> sho parameter memory

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_max_target                    big integer 360M
memory_target                        big integer 360M
```

# Moving from SGA_TARGET to: MEMORY_TARGET



SQL> sho parameter target

| NAME | TYPE | VALUE |
|---|---|---|
| memory_max_target | big integer | 0 |
| memory_target | big integer | 0 |
| pga_aggregate_target | big integer | 110M |
| sga_target | big integer | 250M |

# Moving from SGA_TARGET to: MEMORY_TARGET



ALTER SYSTEM SET MEMORY_MAX_TARGET=360M SCOPE=SPFILE;
(shutdown/startup)
ALTER SYSTEM SET MEMORY_TARGET=360M SCOPE=SPFILE;
ALTER SYSTEM SET SGA_TARGET=0; (or set a minimum)
ALTER SYSTEM SET PGA_AGGREGATE_TARGET=0; (or set a minimum)

SQL> sho parameter target

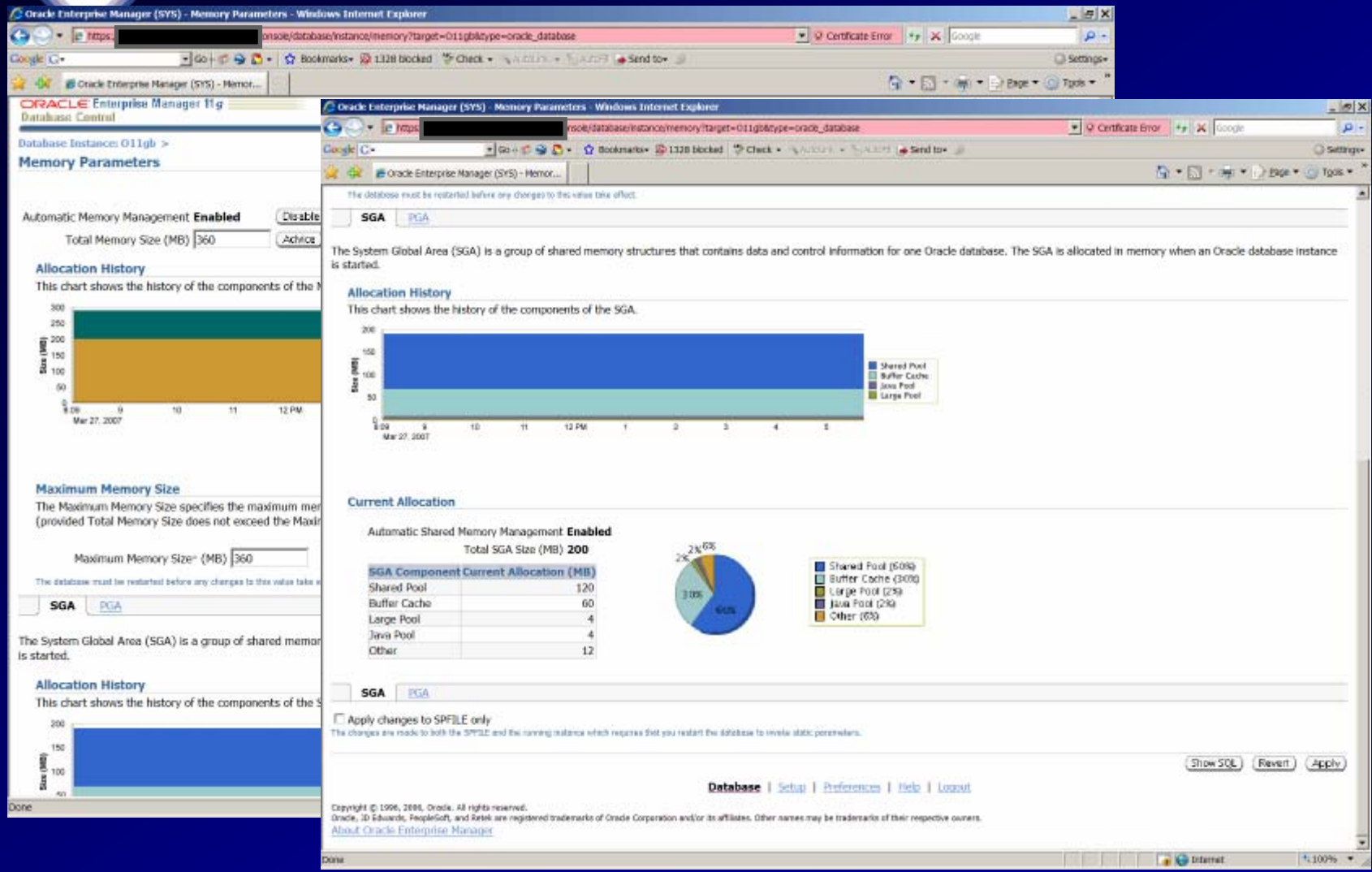
| NAME | TYPE | VALUE |
|---|---|---|
| memory_max_target | big integer | 360M |
| memory_target | big integer | 360M |
| pga_aggregate_target | big integer | 0 |
| sga_target | big integer | 0 |

# Moving from SGA_TARGET to: MEMORY_TARGET (set minimums)



```
ALTER SYSTEM SET SGA_TARGET=200;
ALTER SYSTEM SET PGA_AGGREGATE_TARGET=100;

SQL> sho parameter target

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_max_target                    big integer 360M
memory_target                        big integer 360M
pga_aggregate_target                 big integer 100M
sga_target                           big integer 200M
```

# Moving from SGA_TARGET to: MEMORY_TARGET - EM



# The Result Cache

# First, A quick review: Flush Buffer Cache




- The new 10g feature allows the flush of the buffer cache. It is NOT intended for production use, but rather for system testing purposes.
- This can help you in your tuning needs or as a band-aid if you have 'free buffer' waits (there are better ways to fix this like writing more often or increasing the DB_CACHE_SIZE)
- Note that any Oracle I/O not done in the SGA counts as a physical I/O. If your system has O/S caching or disk caching, the actual I/O that shows up as physical may indeed be a memory read outside of Oracle.
- To flush the buffer cache perform the following:

```
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;
```

# Flush Buffer Cache Example




```
select count(*) from tab1;

  COUNT(*)
----------
      1147

Execution Plan
----------------------------------------------------------------------------------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=1)
   1    0   SORT (AGGREGATE)
   2    1     TABLE ACCESS (FULL) OF 'TAB1' (TABLE) (Cost=4 Card=1147)

Statistics
----------------------------------------------------------
          0  db block gets
          7  consistent gets
          6  physical reads
```

# Flush Buffer Cache Example




select count(*) from tab1; **(Run it again and the physical reads go away)**

```
  COUNT(*)
----------
      1147

Execution Plan
----------------------------------------------------------------------------------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=1)
   1    0   SORT (AGGREGATE)
   2    1     TABLE ACCESS (FULL) OF 'TAB1' (TABLE) (Cost=4 Card=1147)

Statistics
----------------------------------------------------------
          0  db block gets
          7  consistent gets
          0  physical reads
```

# Flush Buffer Cache Example




```
ALTER SYSTEM FLUSH BUFFER_CACHE;
System altered.
```

select count(*) from tab1; **(Flush the cache and the physical reads are back)**

```
  COUNT(*)
----------
      1147

Execution Plan
----------------------------------------------------------------------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=1)
   1    0   SORT (AGGREGATE)
   2    1     TABLE ACCESS (FULL) OF 'TAB1' (TABLE) (Cost=4 Card=1147)

Statistics
----------------------------------------------------------
          0  db block gets
          7  consistent gets
          6  physical reads
```

# 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** (auto / **manual**)
- 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)




```
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;
```

# Result Cache Example Performance




Step 1 - In Session 1-

Executed query without hint and it returned an elapsed time of 3.80 seconds (not cached).

Step 2 - In Session 2 –

Executed query without hint and it returned an elapsed time of 3.20 seconds (not cached).

# Result Cache Example Performance




Step 3 - In Session 2

Executed query with the RESULT_CACHE hint and it returned an elapsed time of 3.18 seconds (cache it).

Step 4 - In Session 1

Executed query without the RESULT_CACHE hint, but with **RESULT_CACHE_MODE=force** and it returned an elapsed time of 0.86 seconds (cached!!).

# Result Cache Example Query From the Oracle Docs




- The RELIES_ON Clause specifies tables or views that the Function Results are dependent on.

```
-- Package specification
CREATE OR REPLACE PACKAGE HR IS
   ...
   type DeptInfoRec IS RECORD (avgSal NUMBER,
                               numberEmployees NUMBER);
-- Function declaration
   FUNCTION GetDeptInfo (dept_id NUMBER) RETURN DeptInfoRec
   RESULT_CACHE;
   ...
END HR;
```

# Result Cache Example Query From the Oracle Docs




```
PACKAGE BODY HR IS
  ...
  -- Function definition
  FUNCTION GetDeptInfo (dept_id NUMBER) RETURN DeptInfoRec
      RESULT_CACHE RELIES_ON (EMP);
  IS
      result DeptInfoRec;
  BEGIN SELECT AVG(sal), count(*) INTO result
                FROM EMP
               WHERE deptno = dept_id;
              RETURN result;
  END;
  ...
END HR;
```

# The Result Cache – V$ Views

- V$RESULT_CACHE_STATISTICS – Displays the amount of memory to help you determine memory currently allocated to the result cache.

Other V$ views:

- V$RESULT_CACHE_MEMORY
- V$RESULT_CACHE_OBJECTS
- V$RESULT_CACHE_DEPENDENCY

# The Result Cache – FYI Only Digging Deeper



| KSPPINM | KSPPSTVL | KSPPDESC |
|---|---|---|
| **_result_cache_auto_execution_threshold** | **1** | **result cache auto execution threshold** |
| _result_cache_auto_size_threshold | 100 | result cache auto max size allowed |
| _result_cache_auto_time_threshold | 1000 | result cache auto time threshold |
| **_result_cache_block_size** | **1024** | **result cache block size** |
| **_result_cache_bypass** | **FALSE** | **bypass the result cache** |
| **_result_cache_hash_buckets** | **1024** | **hash bucket count** |
| _result_cache_invalid | 0 | post-invalidation usage allowance |
| _result_cache_max_result | 100 | maximum result size as percent of cache size |
| _result_cache_remote_expiration | 0 | maximum life time (min) for any result using a remote object |
| _result_cache_timeout | 60 | maximum time (sec) a session waits for a result |

# Tuning Tools – FYI Only
# DBMS_XPLAN




- Use DBMS_XPLAN to query the execution plan
  - Automatically queries the last plan in PLAN_TABLE
  - uses a TABLE() function with another pipelined function
  - Operation text truncation might be a problem
  - Will give additional information after plan
    - Highlight filter vs join conditions, if plan table is current
    - Displays warning message of old version plan table is being used
  - In 11g, a procedure for SQL Plan Baselines (we'll cover these later).

```
DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE (
      sql_handle   IN VARCHAR2 := NULL,
      plan_name    IN VARCHAR2 := NULL,
      format       IN VARCHAR2 := 'TYPICAL')   <'BASIC'/'ALL'>
RETURN dbms_xplan_type_table;
```

# Tuning Tools – FYI Only
# DBMS_XPLAN



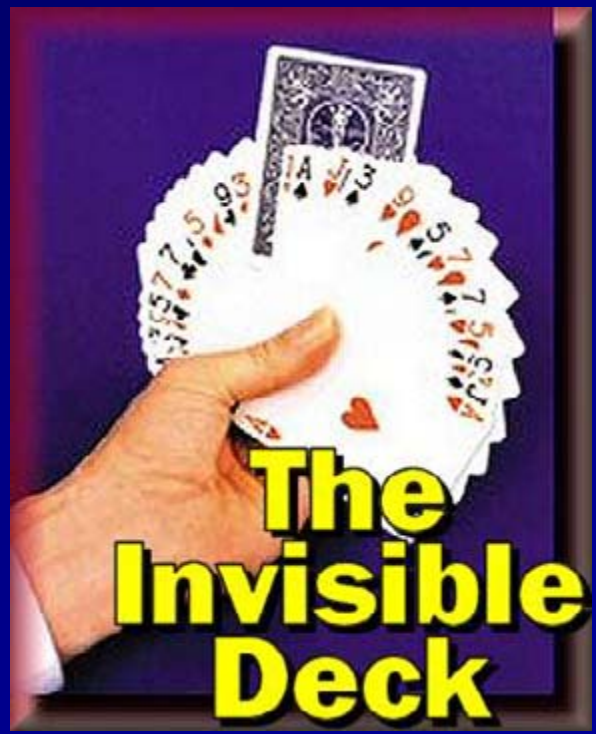

## DBMS_XPLAN Example:

Select *

from table (dbms_xplan.display);

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
| Id  | Operation             |  Name         | Rows  | Bytes | Cost  | Pstart| Pstop |
--------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT      |               |   328 |  2296 |     2 |       |       |
|   1 |  UPDATE               | JOURNAL_LINE  |       |       |       |       |       |
|   2 |   PARTITION RANGE ALL|               |       |       |       |     1 |     4 |
|   3 |    TABLE ACCESS FULL | JOURNAL_LINE  |   328 |  2296 |     2 |     1 |     4 |
--------------------------------------------------------------------------------
Note: cpu costing is off, 'PLAN_TABLE' is old version
11 rows selected
```

# The Invisible Index

# 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 INDEX (to override invisibility) or NO_INDEX (to override visibility) hints to override either setting.
- The index IS MAINTAINED during DML
- Great for testing!

# The Invisible Index

create index deptno_invisible_idx on dept_rich(deptno) invisible;

*Index created.*

select count(*) from dept_rich where deptno = 30; (doesn't see the index)

```
COUNT(*)
----------
       512

Execution Plan
----------------------------------------------------------
Plan hash value: 3024595593

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |     2 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |           |     1 |     2 |            |          |
|*  2 |   TABLE ACCESS FULL| DEPT_RICH |   512 |  1024 |     4   (0)| 0:00:01 |
-------------------------------------------------------------------------------
```

# The Invisible Index

select /*+ index(dept_rich dept_rich_inv_idx) */ count(*)
from dept_rich where deptno = 30; (forces the index with hint)

```
COUNT(*)
----------
       512

Execution Plan
----------------------------------------------------------
Plan hash value: 3699452051
-------------------------------------------------------------------------------------------
| Id  | Operation         | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                   |     1 |     2 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                   |     1 |     2 |            |          |
|*  2 |   INDEX RANGE SCAN| DEPT_RICH_INV_IDX |   512 |  1024 |1 (0)| 00:00:01
    |
-------------------------------------------------------------------------------------------
```

# The Invisible Index (set visible)

alter index dept_rich_inv_idx visible;

*Index altered.*

select count(*) from dept_rich where deptno = 30;

(it does see the index)

```
COUNT(*)
----------
       512

Execution Plan
----------------------------------------------------------
Plan hash value: 3699452051
--------------------------------------------------------------------------------
| Id  | Operation         | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                   |     1 |     2 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                   |     1 |     2 |            |          |
|*  2 |   INDEX RANGE SCAN| DEPT_RICH_INV_IDX |   512 |  1024 |1 (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

# The Invisible Index (set visible)

select /*+ no_index(dept_rich dept_rich_inv_idx) */ count(*)
from dept_rich
where deptno = 30; (forces *not* using the index with hint)

```
  COUNT(*)
----------
       512

Execution Plan
----------------------------------------------------------
Plan hash value: 3024595593

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |     2 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |           |     1 |     2 |            |          |
|*  2 |   TABLE ACCESS FULL| DEPT_RICH |   512 |  1024 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

# The Invisible Index (check it)

alter index dept_rich_inv_idx invisible;

*Index altered.*

select index_name, visibility
from dba_indexes (or go to USER_INDEXES)
where index_name = 'DEPT_RICH_INV_IDX';

| INDEX_NAME | VISIBILITY |
|---|---|
| DEPT_RICH_INV_IDX | INVISIBLE |

# Create & Rebuild Index Online

# 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.*

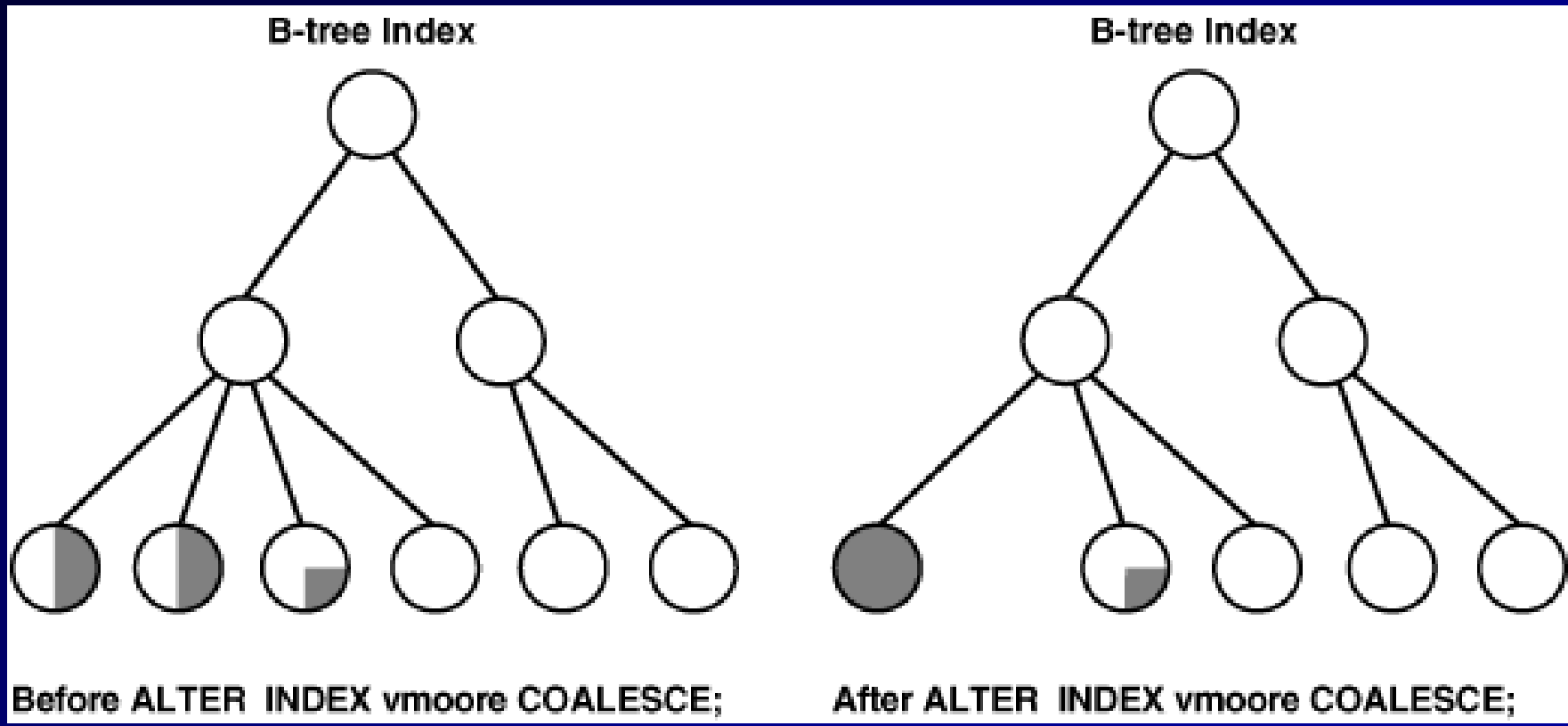
# Create Index Online (careful)

```
select  index_name, visibility
from  dba_indexes
where index_name = 'DEPT_RICH_INV_IDX';

INDEX_NAME                     VISIBILITY
------------------------------ ----------
DEPT_RICH_INV_IDX              INVISIBLE
```

```
alter index dept_rich_inv_idx rebuild tablespace users online;
Index altered.
```

```
select index_name, visibility
from dba_indexes
where index_name = 'DEPT_RICH_INV_IDX';

INDEX_NAME                     VISIBILITY
------------------------------ ----------
DEPT_RICH_INV_IDX              VISIBLE
```

# Rebuild Index or Coalesce (FYI)
# Coalesce Example from Oracle Doc.



# 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

# Alter Index Online (careful)

```
select  segment_name, segment_type, tablespace_name from dba_segments
where segment_name = 'DEPT_RICH_INV_IDX';

SEGMENT_NAME       SEGMENT_TYPE  TABLESPACE_NAME
------------------------------ ------------------------- ------------------------------
DEPT_RICH_INV_IDX  INDEX              SYSTEM
```

```
alter index dept_rich_inv_idx rebuild tablespace users online;
```

*Index altered.*

```
select  segment_name, segment_type, tablespace_name from dba_segments
where segment_name = 'DEPT_RICH_INV_IDX';

SEGMENT_NAME       SEGMENT_TYPE  TABLESPACE_NAME
------------------------------ ------------------------- ------------------------------
DEPT_RICH_INV_IDX  INDEX              USERS
```

# The Virtual Column

# The Virtual Column

- The value of the virtual column is a derived expression.
  - Can be derived from columns of the same table
  - Can be derived from constants
  - Can include SQL or user-defined PL/SQL functions
- You can create an index or partition on a virtual column
- Index Organized and External Tables can NOT have virtual columns.
- You can NOT explicitly write to a virtual column

# The Virtual Column

```
create table emp_rich
  (empno number(4),
   sal    number(7,2),
   yearly_sal generated always as (sal*12),
   deptno number(2));
```

*Table created.*

```
insert into emp_rich(empno, sal, deptno)
                select empno, sal, deptno from scott.emp;
```

*14 rows created.*

# The Virtual Column



```
select * from emp_rich;

 EMPNO      SAL YEARLY_SAL    DEPTNO
---------- ---------- ---------- ----------
      7369        800       9600         20
      7499       1600      19200         30
      7521       1250      15000         30
      7566       2975      35700         20
      7654       1250      15000         30
      7698       2850      34200         30
...
```

# Nice Developer Tools / Improvements


**DDL_LOCK_TIMEOUT**

**PL / SQL Expressions**

**Simple Integer**

**New PL / SQL Packages**

# The DDL Lock Timeout

- DDL Statements (Create / Alter / Drop) require exclusive locks and thus sometimes fail due to bad timing.
- The parameter DDL_LOCK_TIMEOUT specifies the amount of time (in seconds) the DDL statement will wait for the lock before timing out and failing.
- The default value is 0, the max value is 100000 (27.77 hours).
- Example:

```
alter session set DDL_LOCK_TIMEOUT = 30

Session altered.
```

# 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;
```

# Simple Integer Data Type

- Oracle added the new SIMPLE_INTEGER data type to be more efficient than PLS_INTEGER since the operations are done directly at the hardware level. There is also a built-in NOT NULL condition for SIMPLE_INTEGER.
- The performance is larger when the PLS_CODE_TYPE='NATIVE' vs. INTERPRETED
- We used a PL/SQL Block to loop through 1 million times incrementing a numeric variable by one. We executed the test for each of these three times.

Results:

NUMBER: 1.26s

PLS_INTEGER: 0.88s

**SIMPLE_INTEGER: 0.65s**

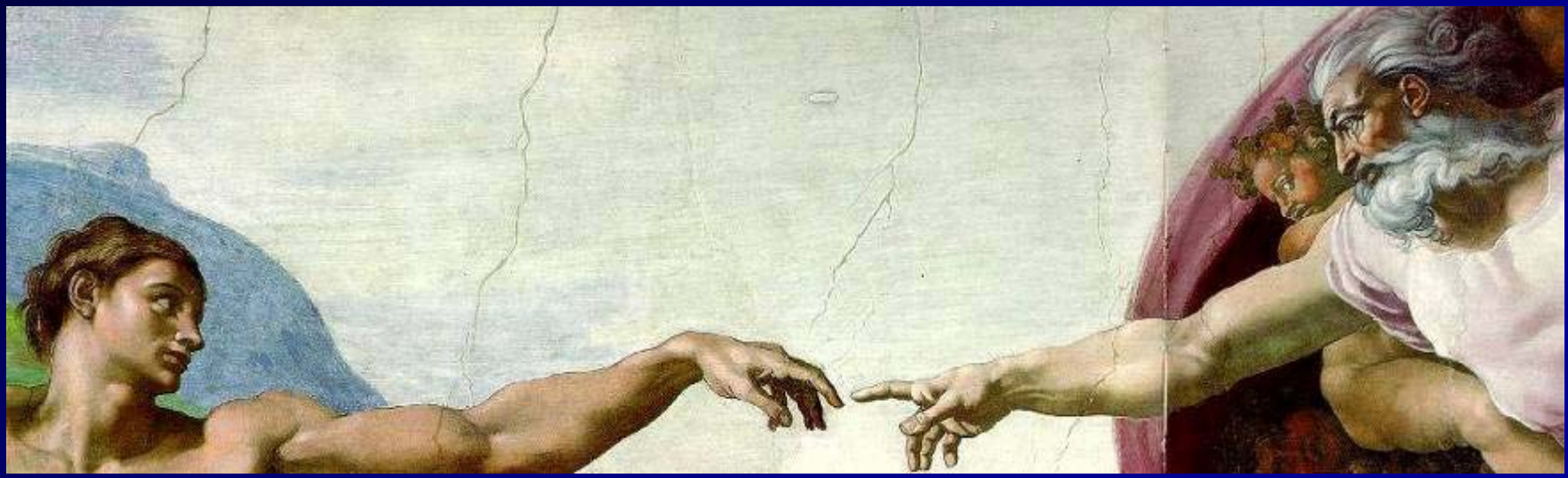
# Additional Enhancements
## New PL/SQL Packages

- DBMS_AUTO_TASK_ADMIN
- DBMS_COMPARISON
- DBMS_DG
- DBMS_EDITIONS_UTILITIES
- DBMS_HM (Health Monitor)
- DBMS_HPROF
- DBMS_MGD_ID_UTL
- DBMS_NETWORK_ACL_ADMIN
- DBMS_RESCONFIG
- DBMS_RESULT_CACHE
- DBMS_SQLDIAG (SQL Repair)
- DBMS_WORKLOAD_CAPTURE
- DBMS_WORKLOAD_REPLAY
- DBMS_XA
- DBMS_XDBADMIN
- DBMS_XEVENT
- DBMS_XMLDTD
- DBMS_XMLINDEX
- DBMS_XMLTRANSLATIONS
- SDO_RDF
- SDO_RDF_INFERENCE

# Additional Enhancements
## Enhanced PL/SQL Packages

- DBMS_ADVISOR
- DBMS_APPLY_ADM
- DBMS_AQ
- DBMS_AQADM
- DBMS_CAPTURE_ADM
- DBMS_CDC_PUBLISH
- DBMS_CDC_SUBSCRIBE
- DBMS_CQ_NOTIFICATION
- DBMS_DATA_MINING
- DBMS_DATA_MINING_TRANSFORM
- DBMS_DATAPUMP
- DBMS_EXPFIL
- DBMS_FLASHBACK
- DBMS_HS_PASSTHROUGH
- DBMS_LOB
- DBMS_LOGSTDBY
- DBMS_MGWADM
- DBMS_MVIEW
- DBMS_PREDICTIVE_ANALYTICS
- DBMS_RESOURCE_MANAGER
- DBMS_RLMGR
- DBMS_RULE_ADM
- DBMS_SCHEDULER
- DBMS_SERVER_ALERT
- DBMS_SESSION
- DBMS_SPACE
- DBMS_SQL
- DBMS_SQLTUNE
- DBMS_STATS
- DBMS_STREAMS_ADM
- DBMS_TRACE
- DBMS_UTILITY
- DBMS_WORKLOAD_REPOSITORY
- DBMS_XDB
- DBMS_XMLSCHEMA
- DBMS_XPLAN
- UTL_INADDR
- UTL_RECOMP
- UTL_SMTP
- UTL_TCP

# ADDM Enhancements



# ADDM enhancements

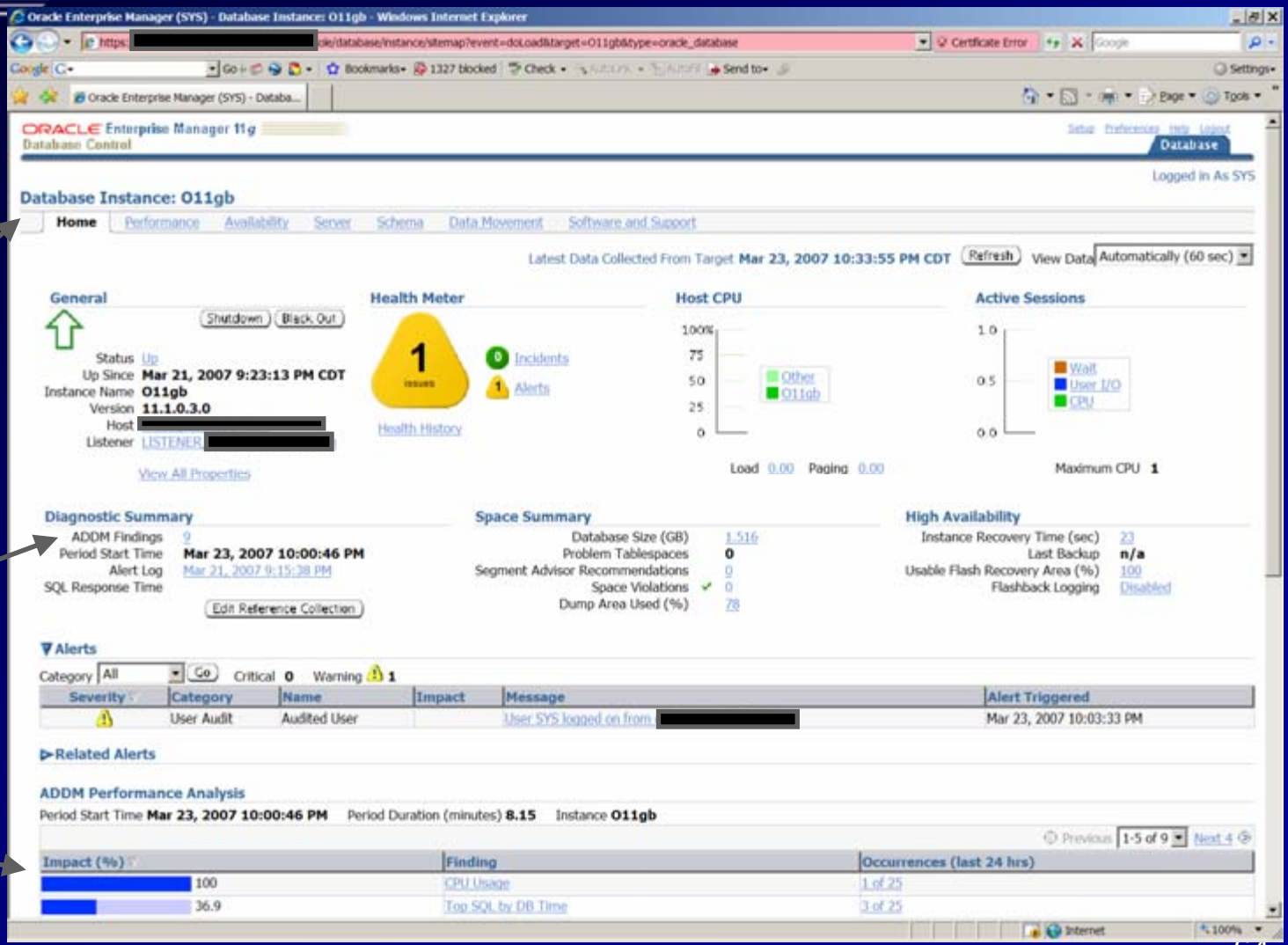
- Global ADDM so that Diagnostics are done across the entire cluster
- Emergency ADDM for use when database is hung
- On any granularity
  - Database Cluster
  - Database Instance
  - Specific Target (such as host, ASM...etc.)
- Over a specified time NOT tied to a pair of snapshots

# ADDM Briefly

Specific Database Instance

We have 9 ADDM Findings

Check them Here

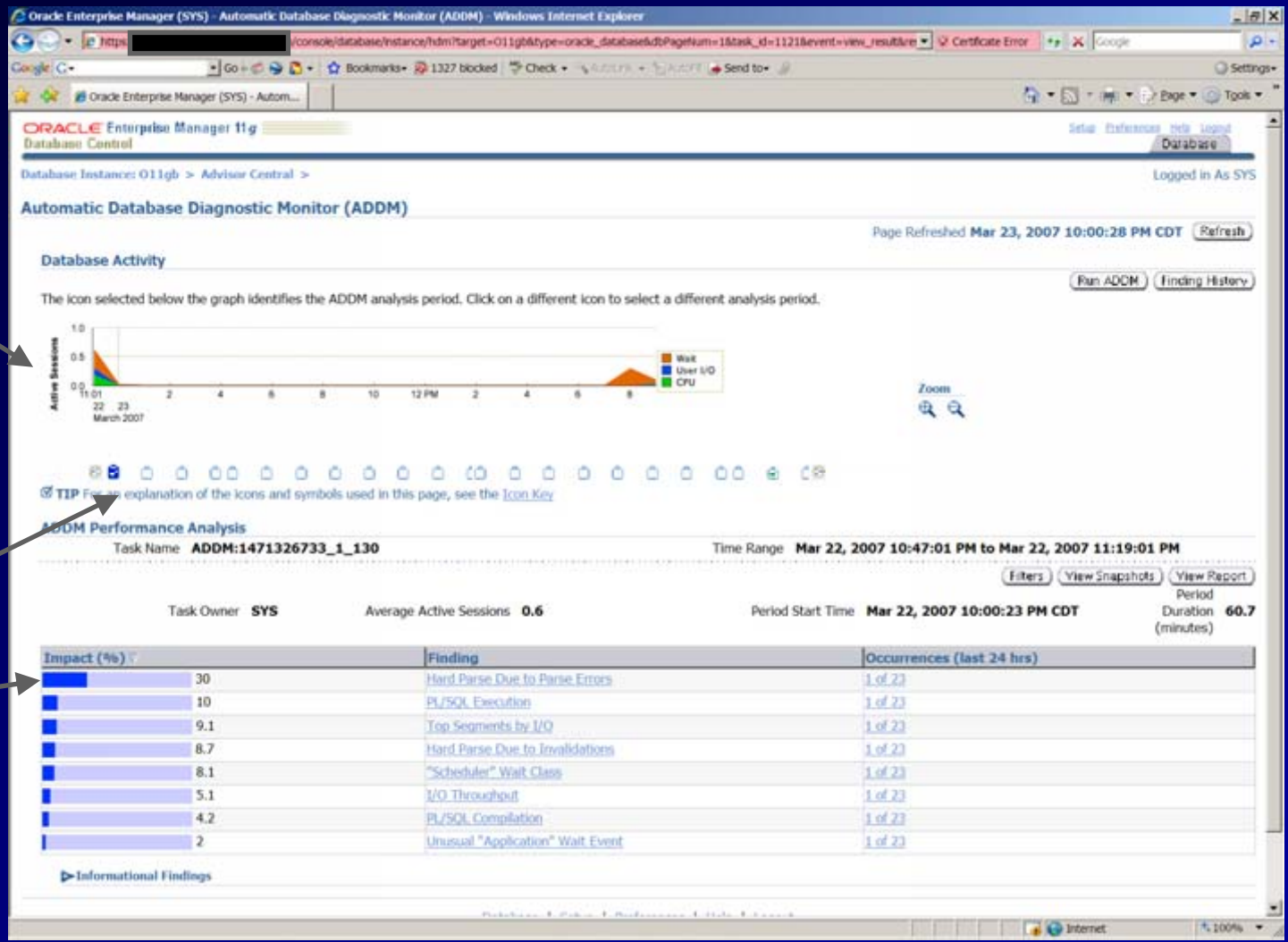


# ADDM Briefly

Top ADDM Findings

Click a Single Timeframe

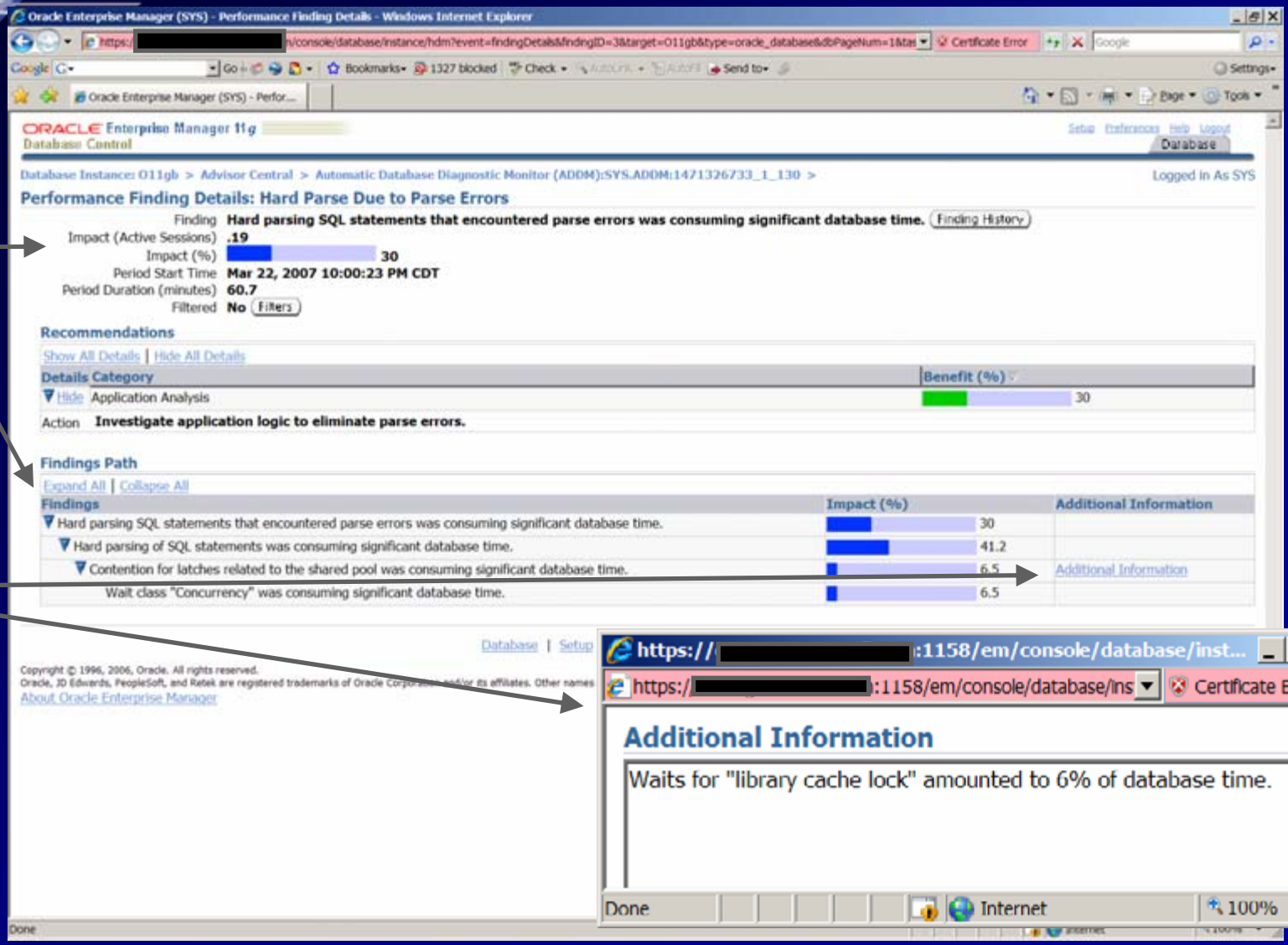
Let's Check the Hard Parse Issue



# ADDM Briefly

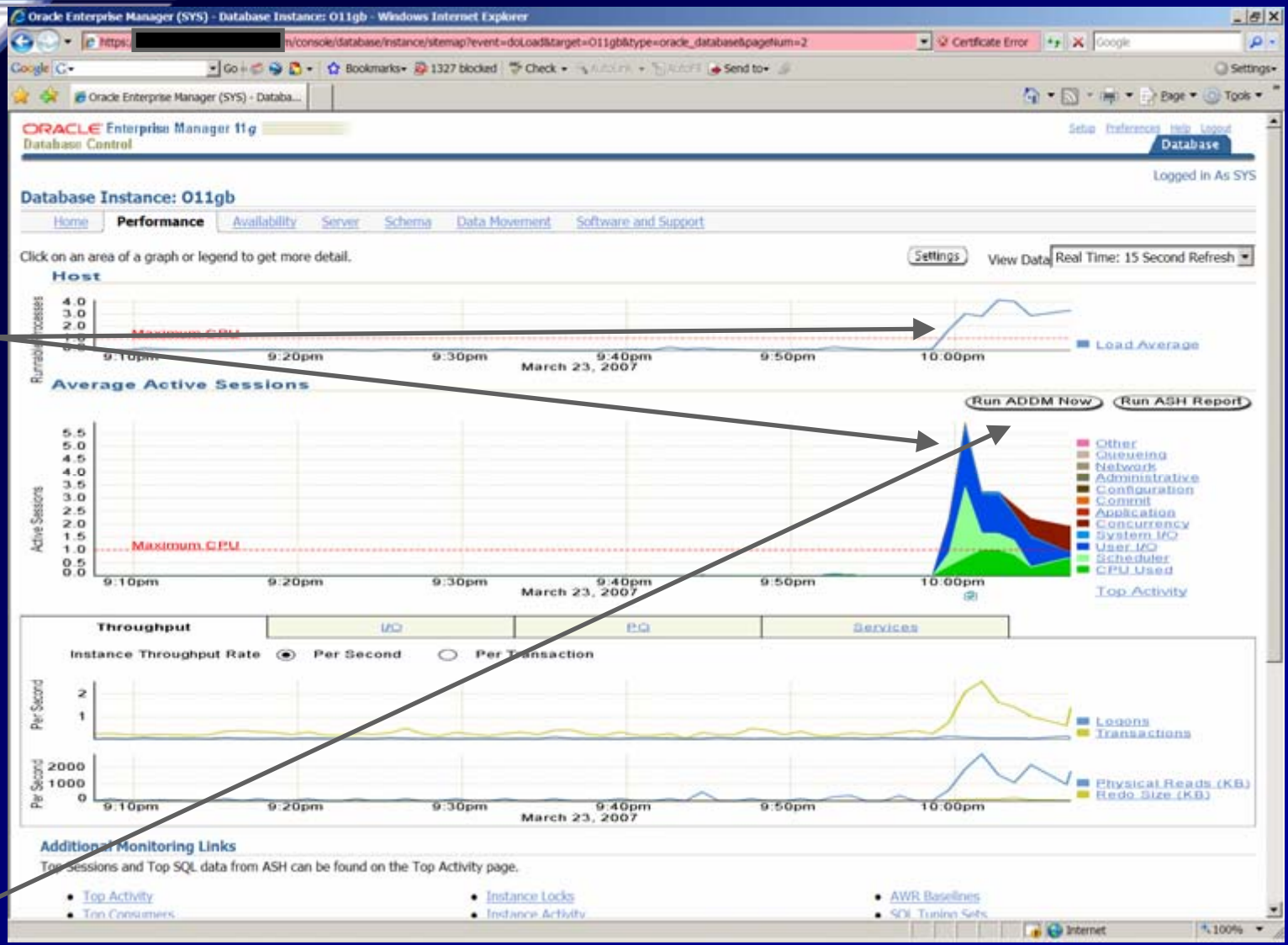
Detailed Info & Findings

Add'l Info



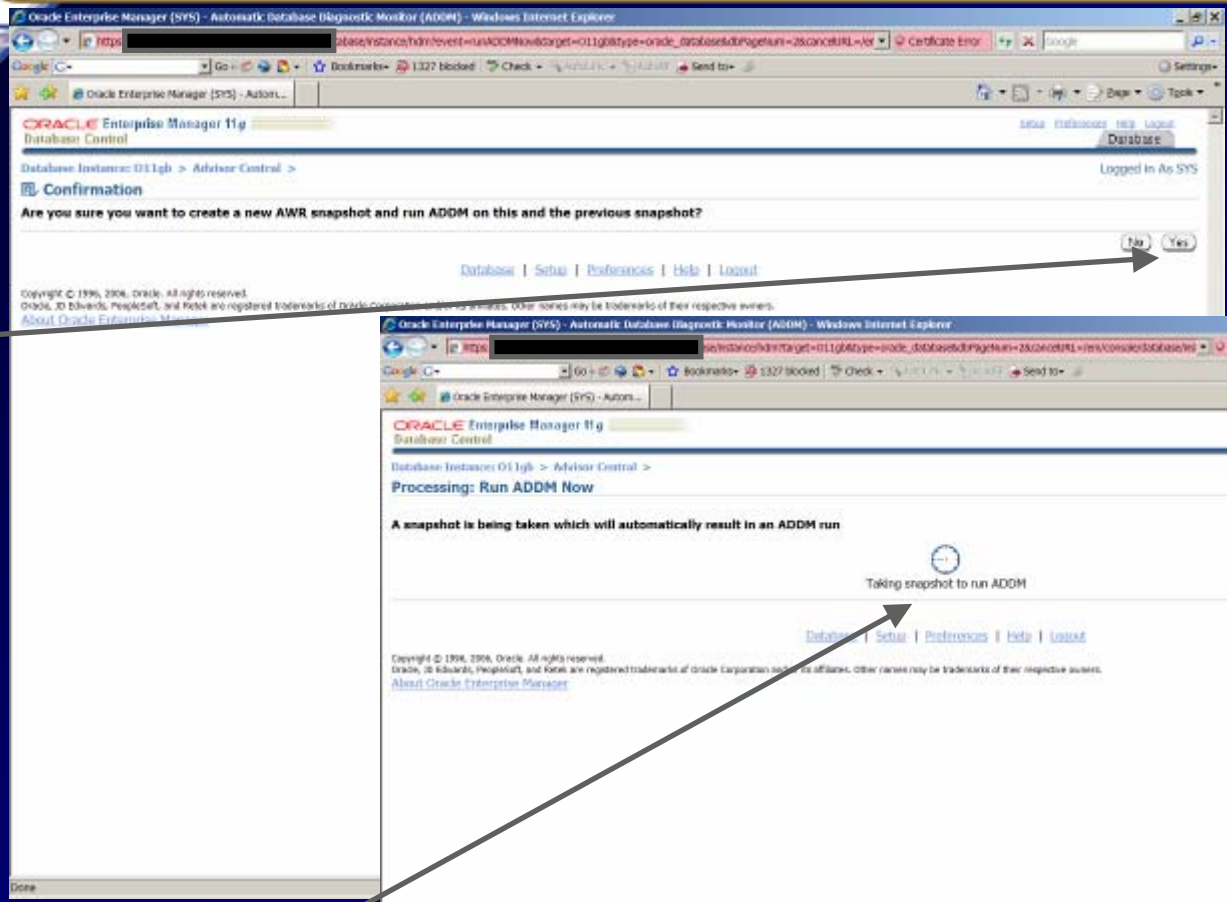
# ADDM - Run NOW!

A Big Problem Occurs




Run ADDM NOW!

# ADDM - Run NOW!



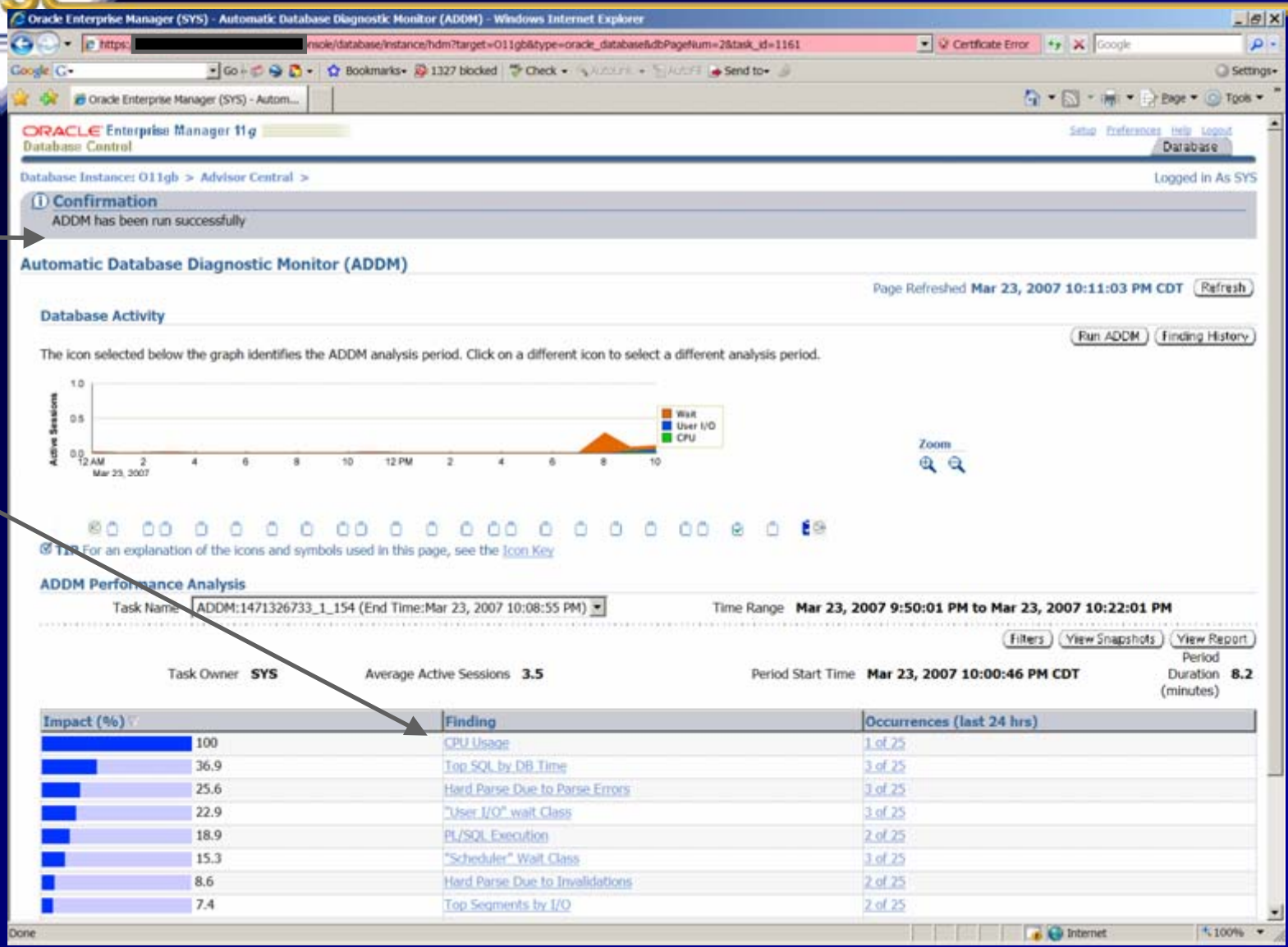

Are you Sure?

Running

# ADDM – Run NOW!

Done.

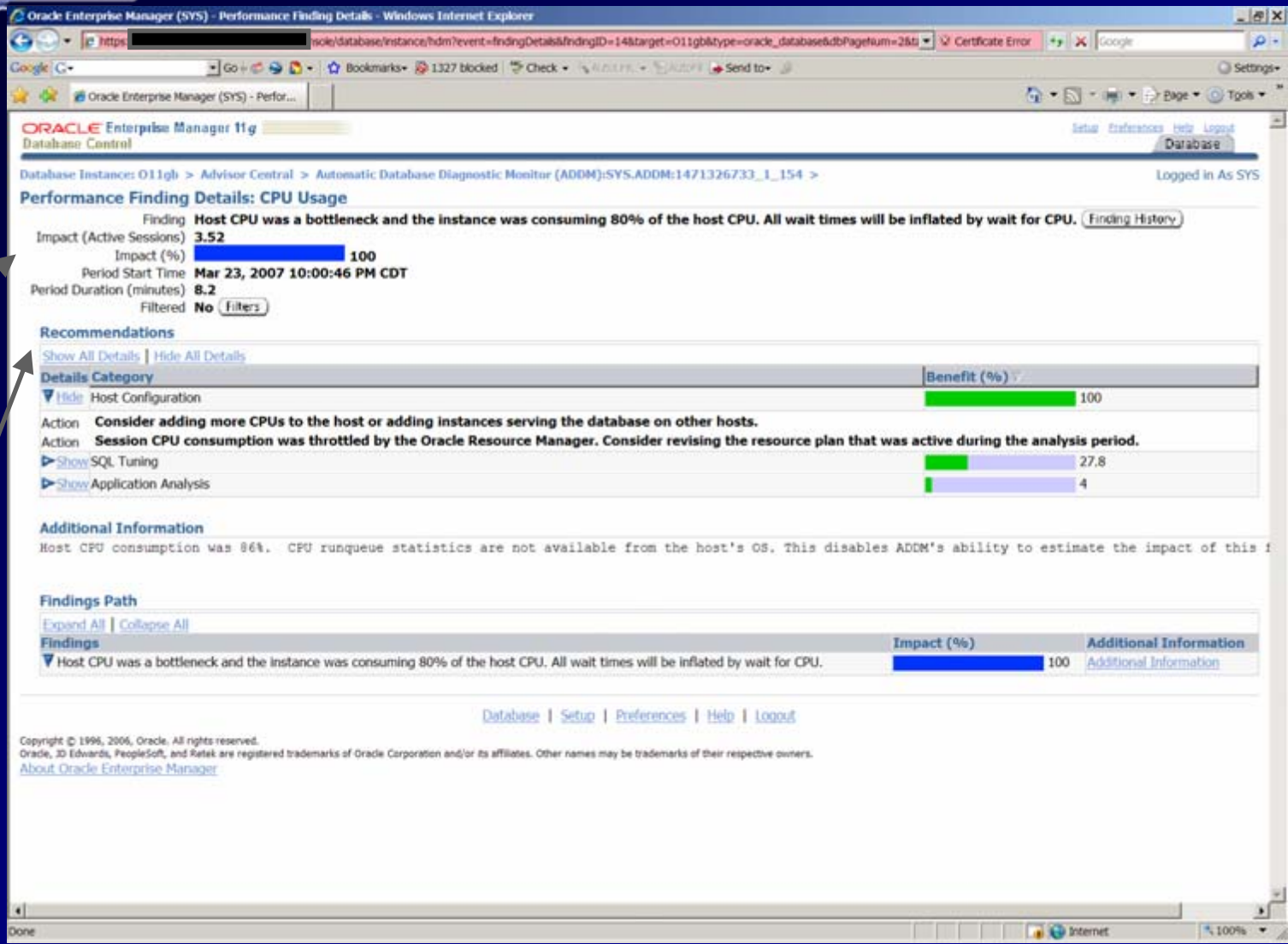
CPU Issue



# ADDM – Run NOW!

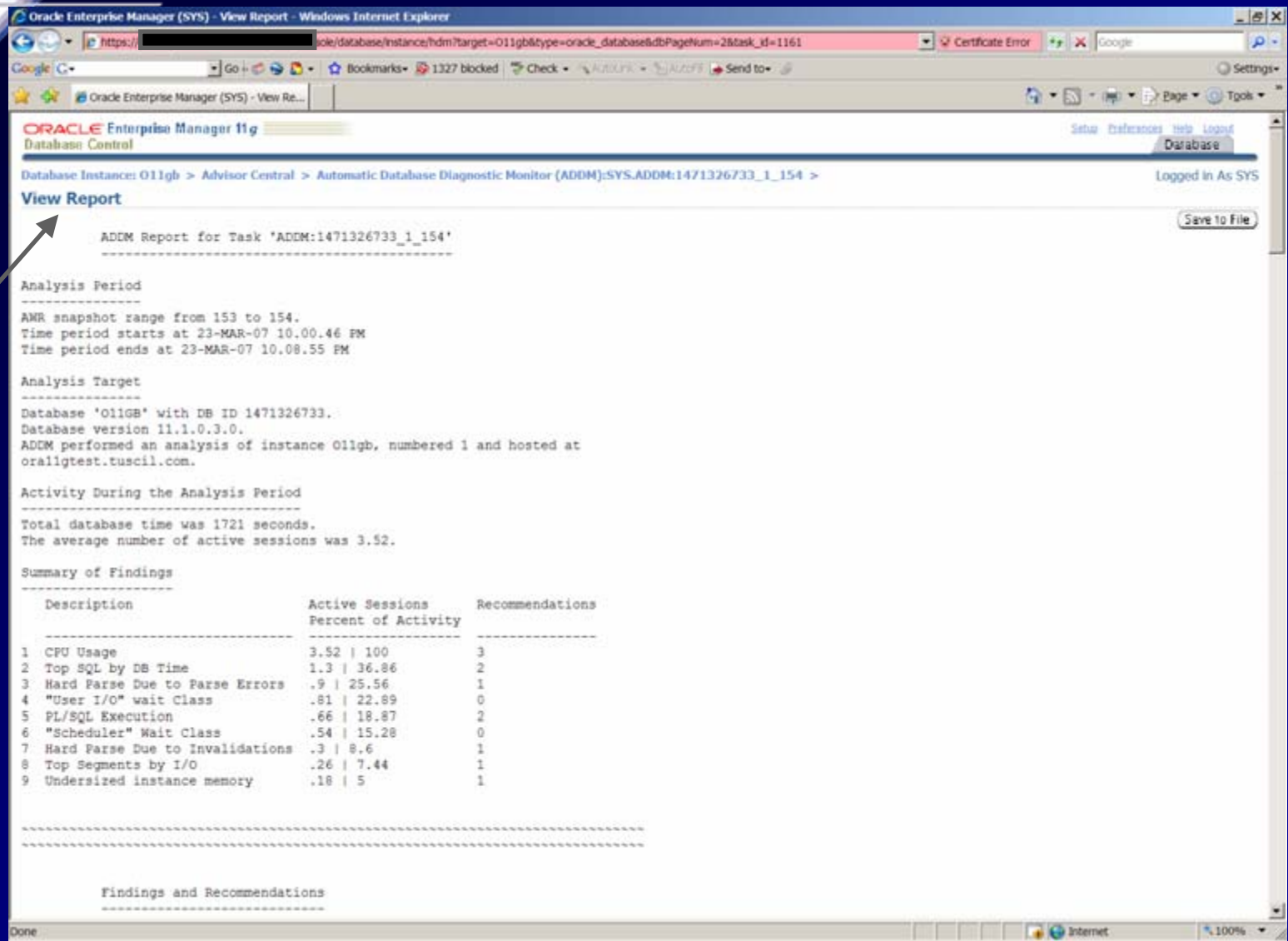
Detail on CPU Issue?

Suggested Fixes



# ADDM – Run NOW!

View The Report



# 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, Checkpoint.

# SQL Plan Management

# 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 11*g*. 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 better execution plan.

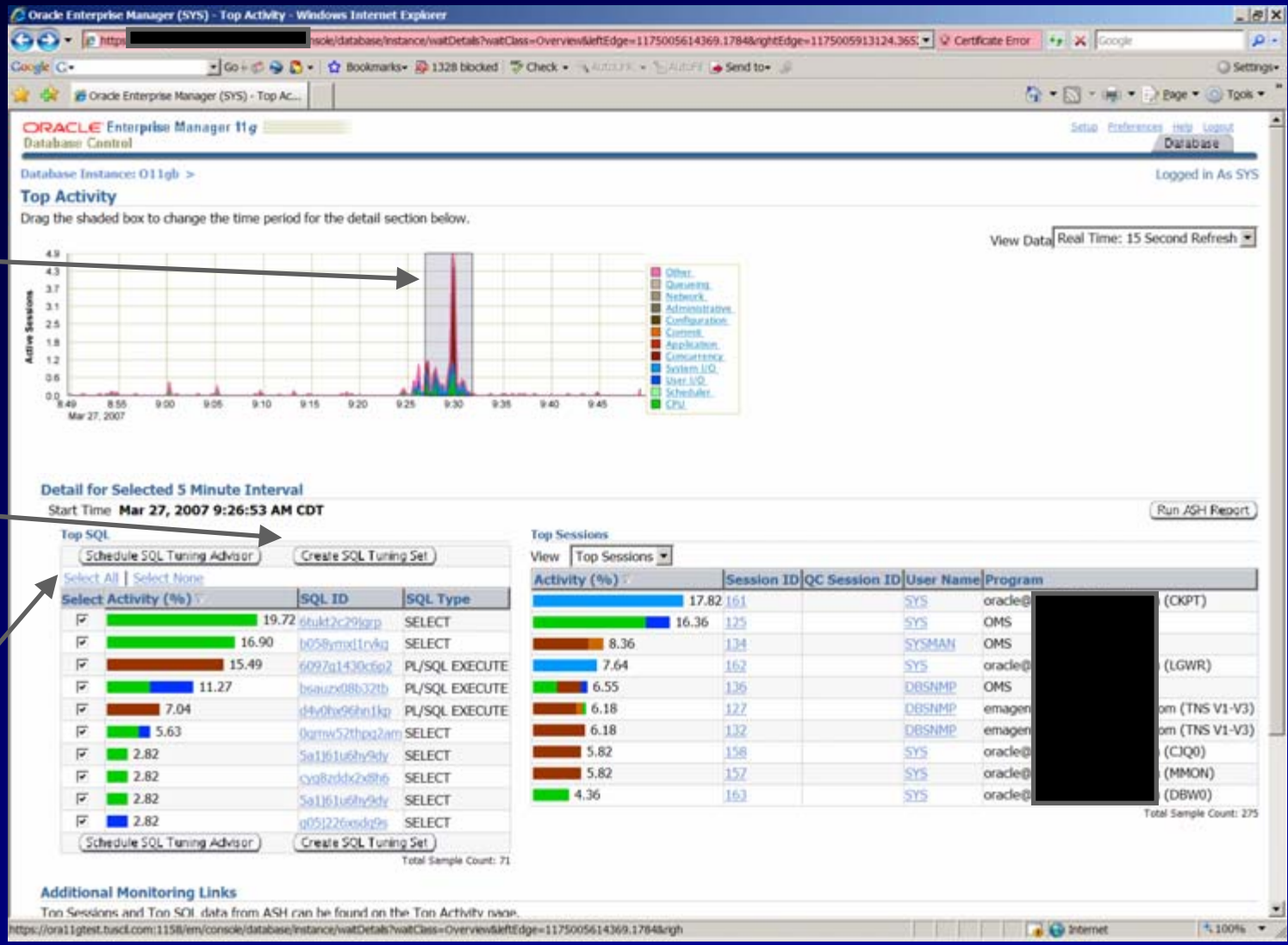
# SQL Plan Management

- **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.
- A SQL Profile is automatic in 11g.
- 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 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

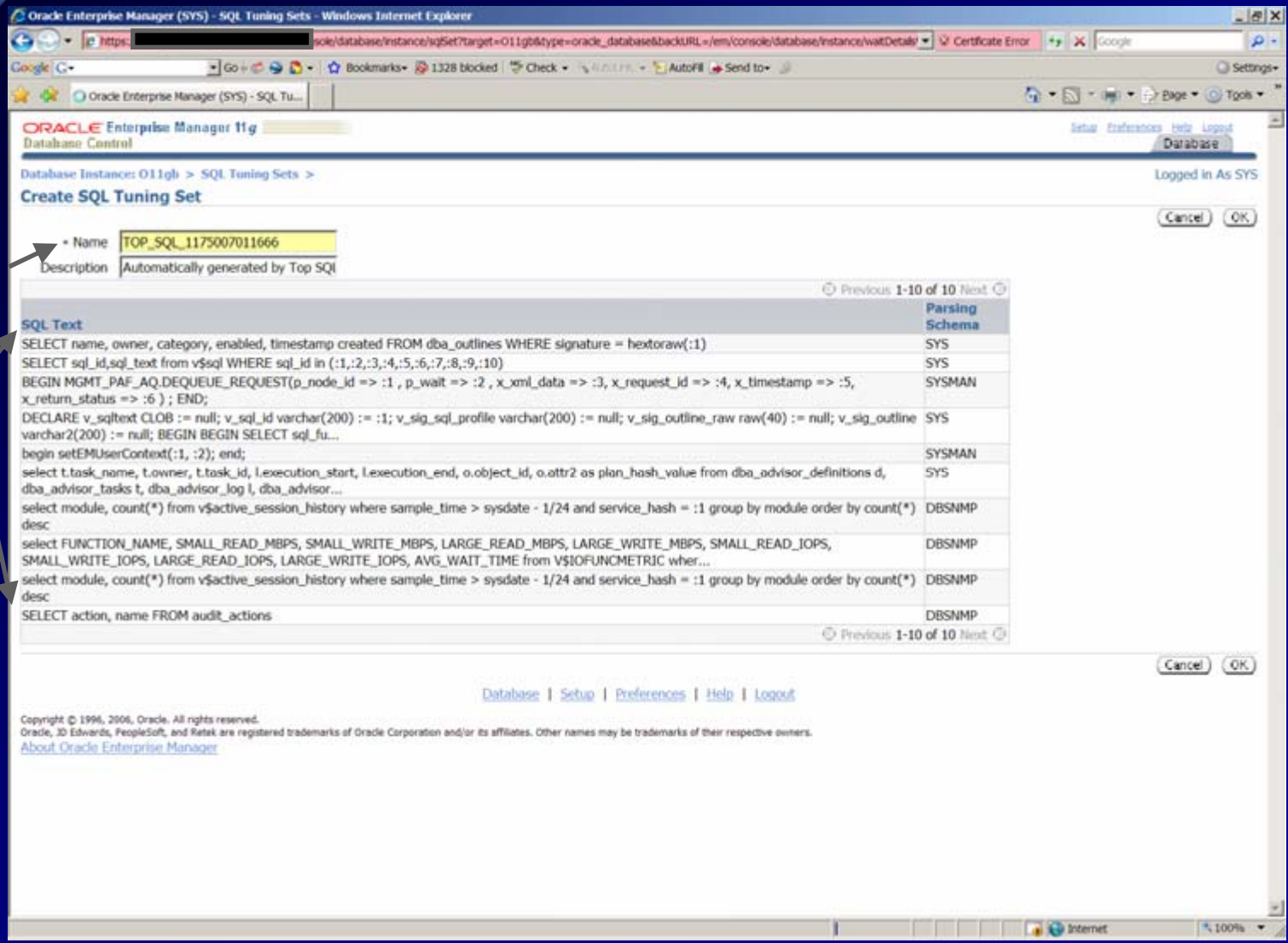
Create a Tuning Set from Top 10 SQL



# SQL Plan Management
# Create a SQL Tuning Set

Tuning Set Name

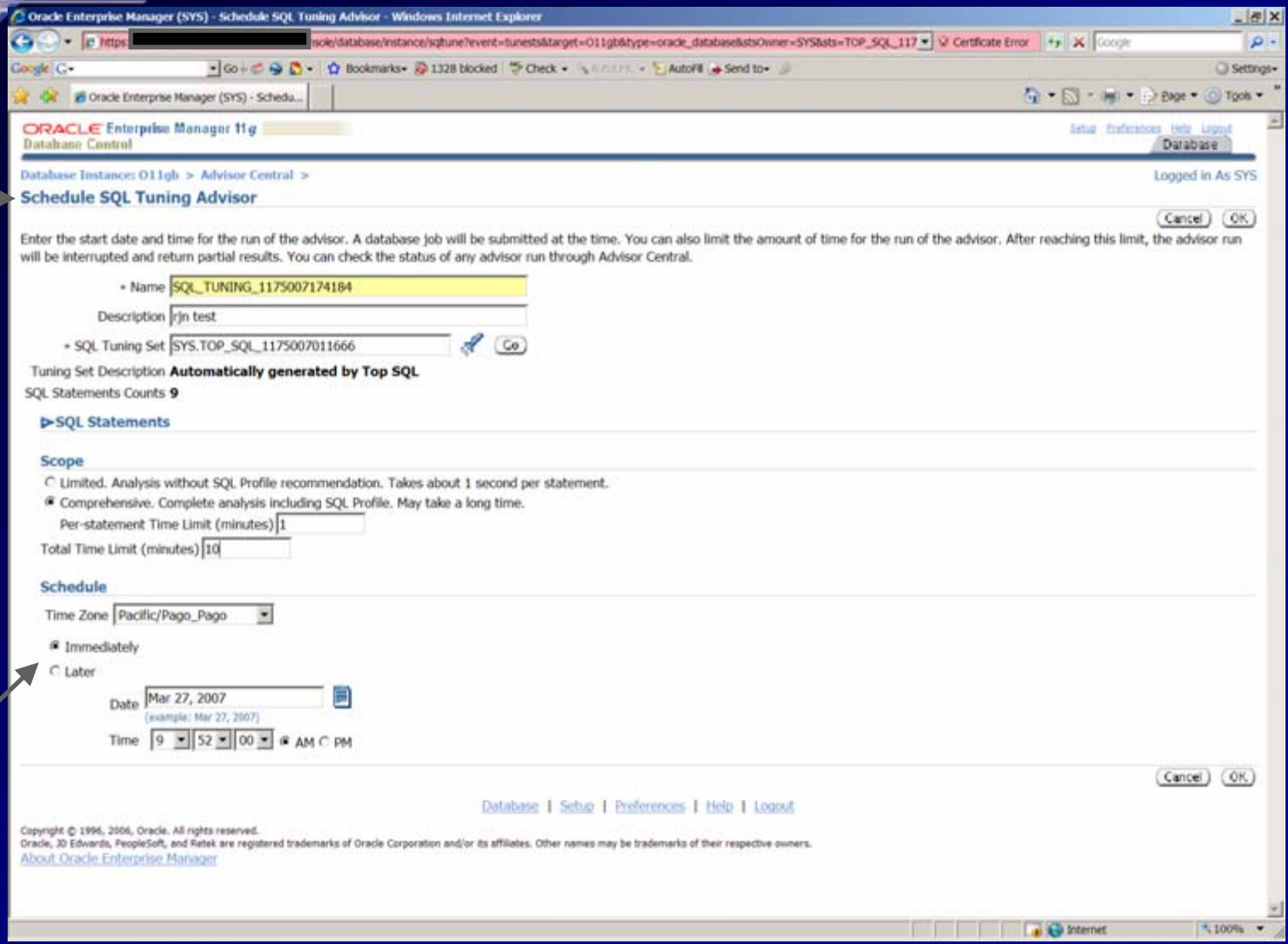
Queries



# SQL Plan Management
# Create a SQL Tuning Set

Run the Tuning Advisor on this SQL Tuning Set (STS)

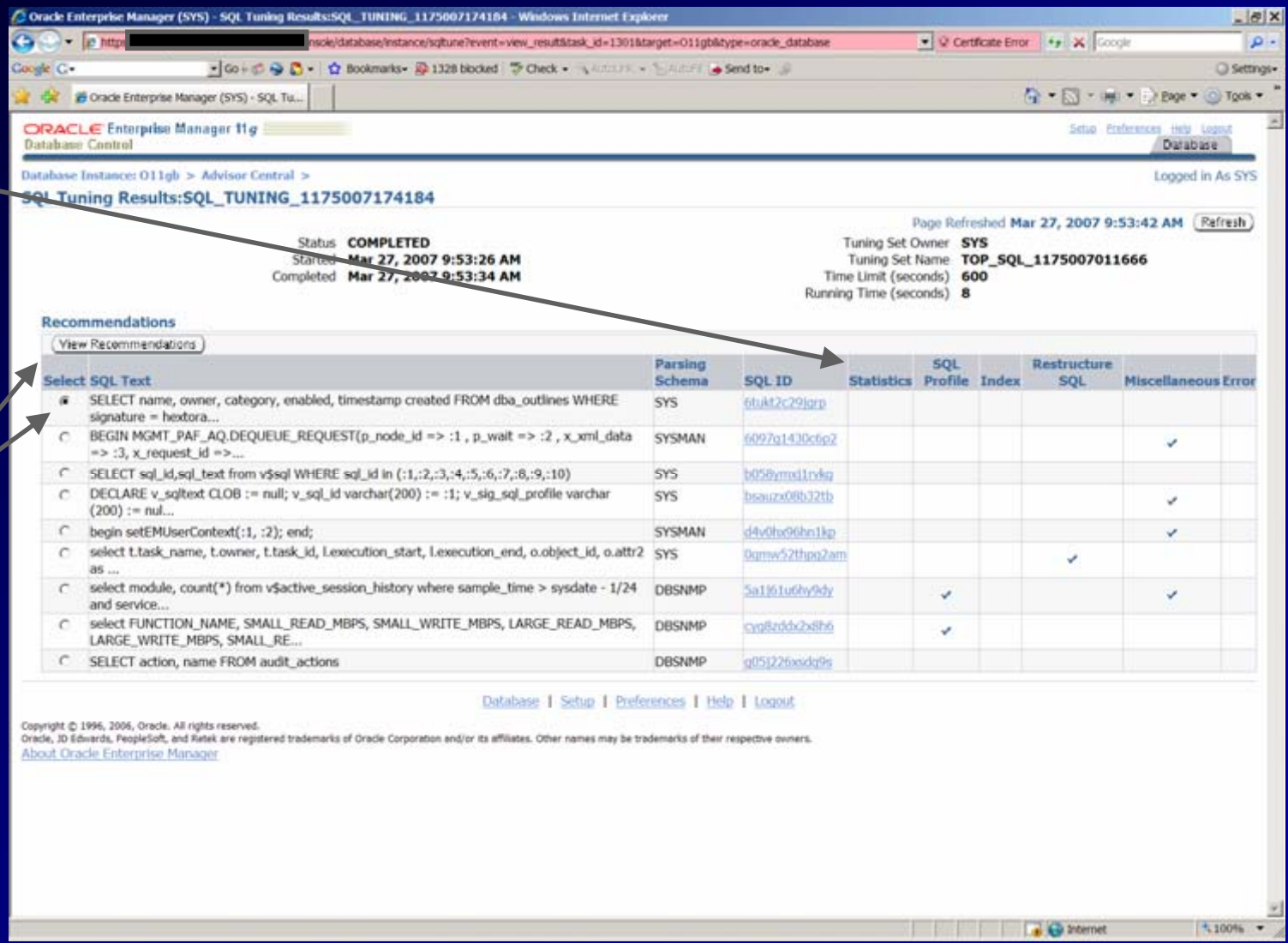
Run it NOW



# SQL Plan Management
# Create a SQL Tuning Set

Results

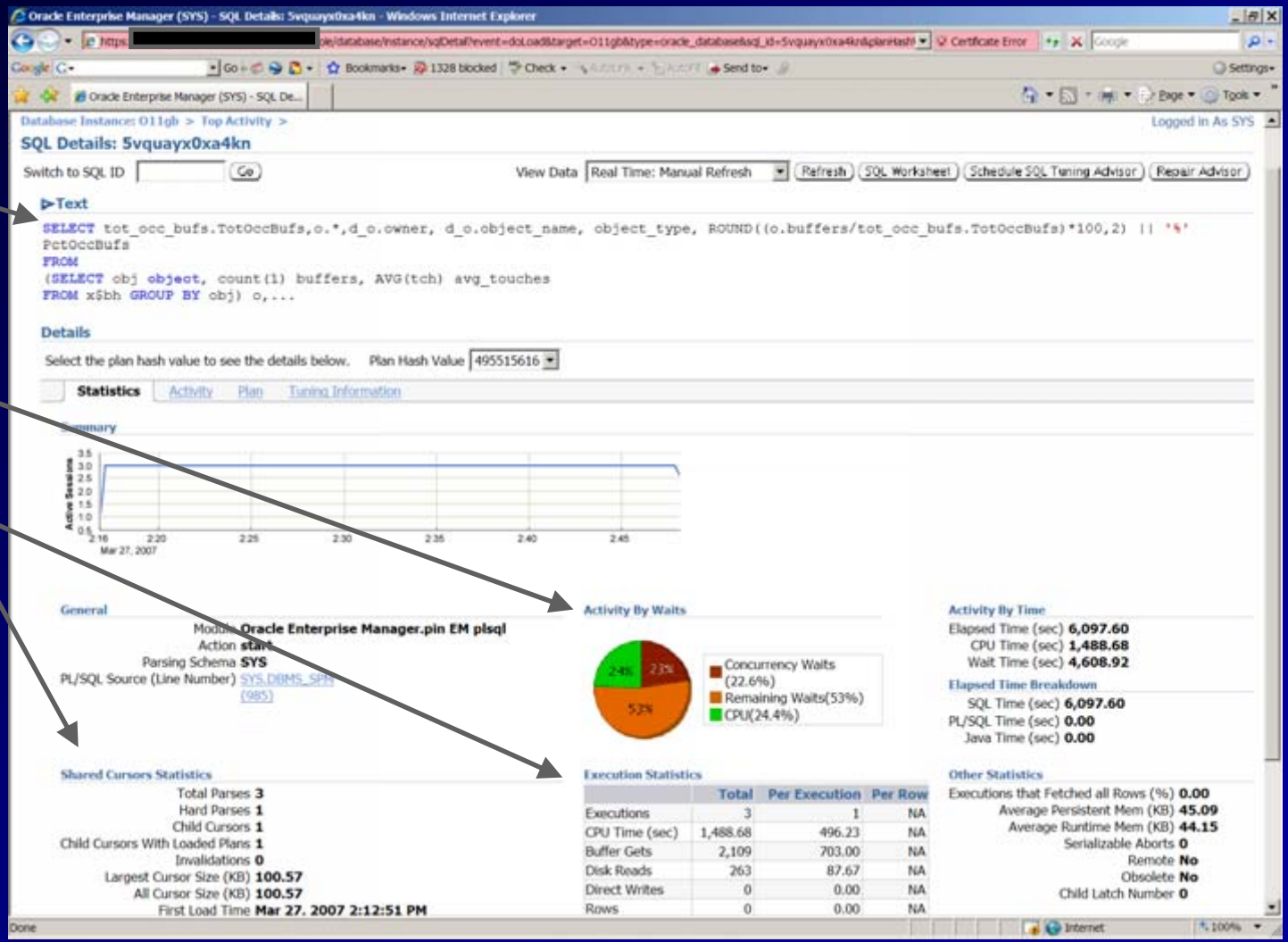
Select
One query
And click
View



# SQL Plan Management
# Click on any SQL ID

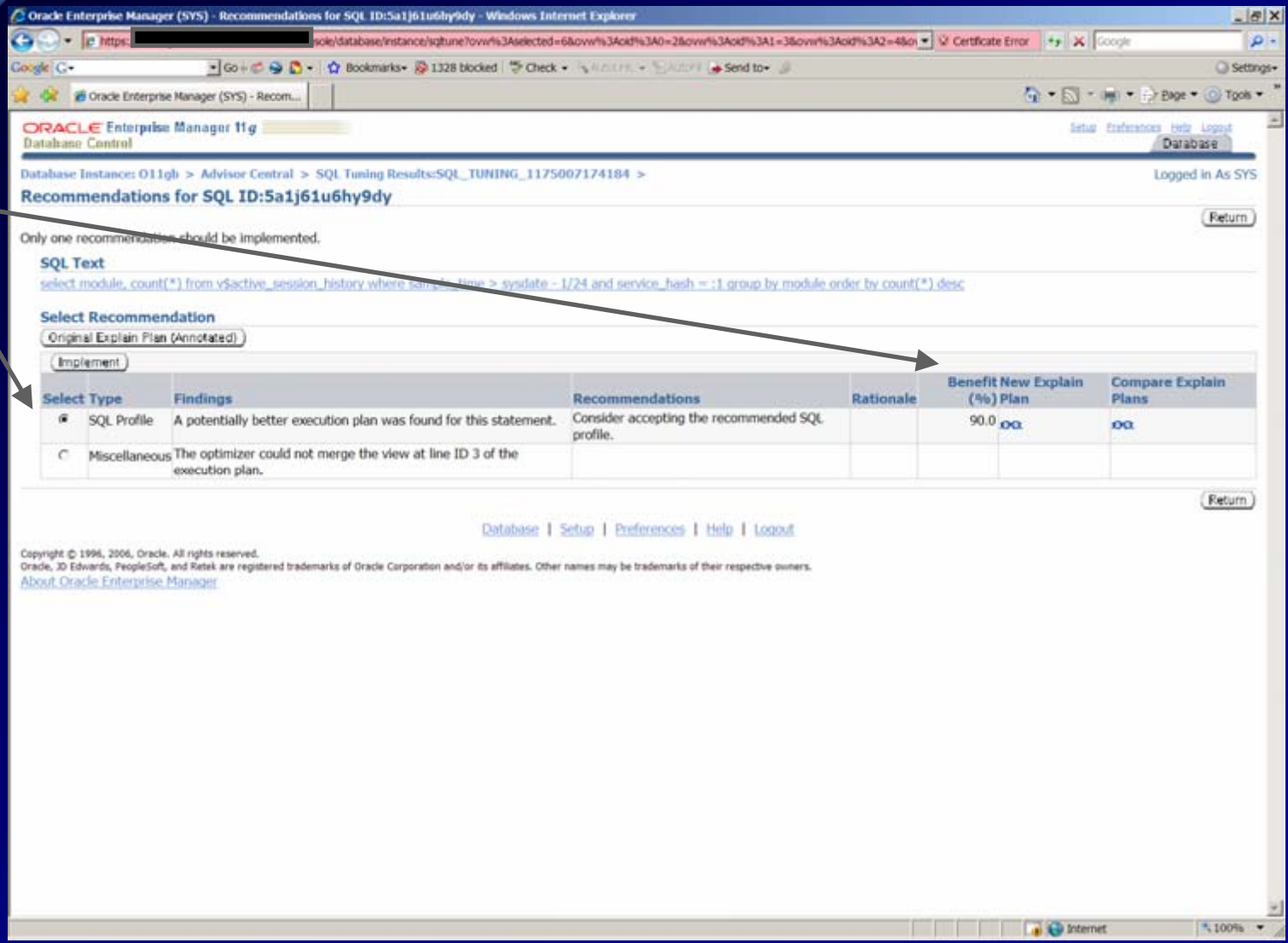
SQL Text

Waits & Statistics



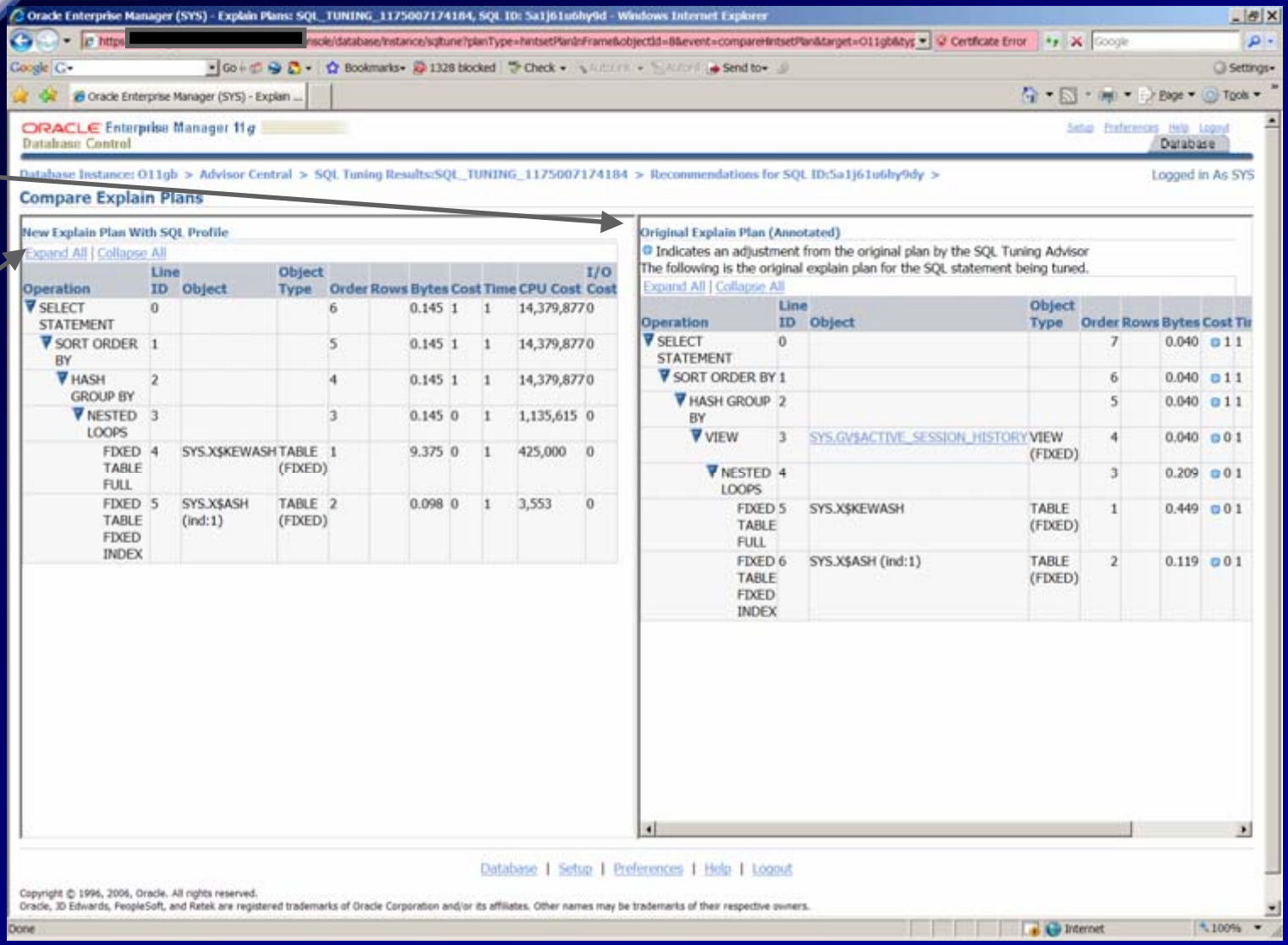
# SQL Plan Management
# Create a SQL Tuning Set

SQL Profile Will Help 90%



# SQL Plan Management
# Create a SQL Tuning Set

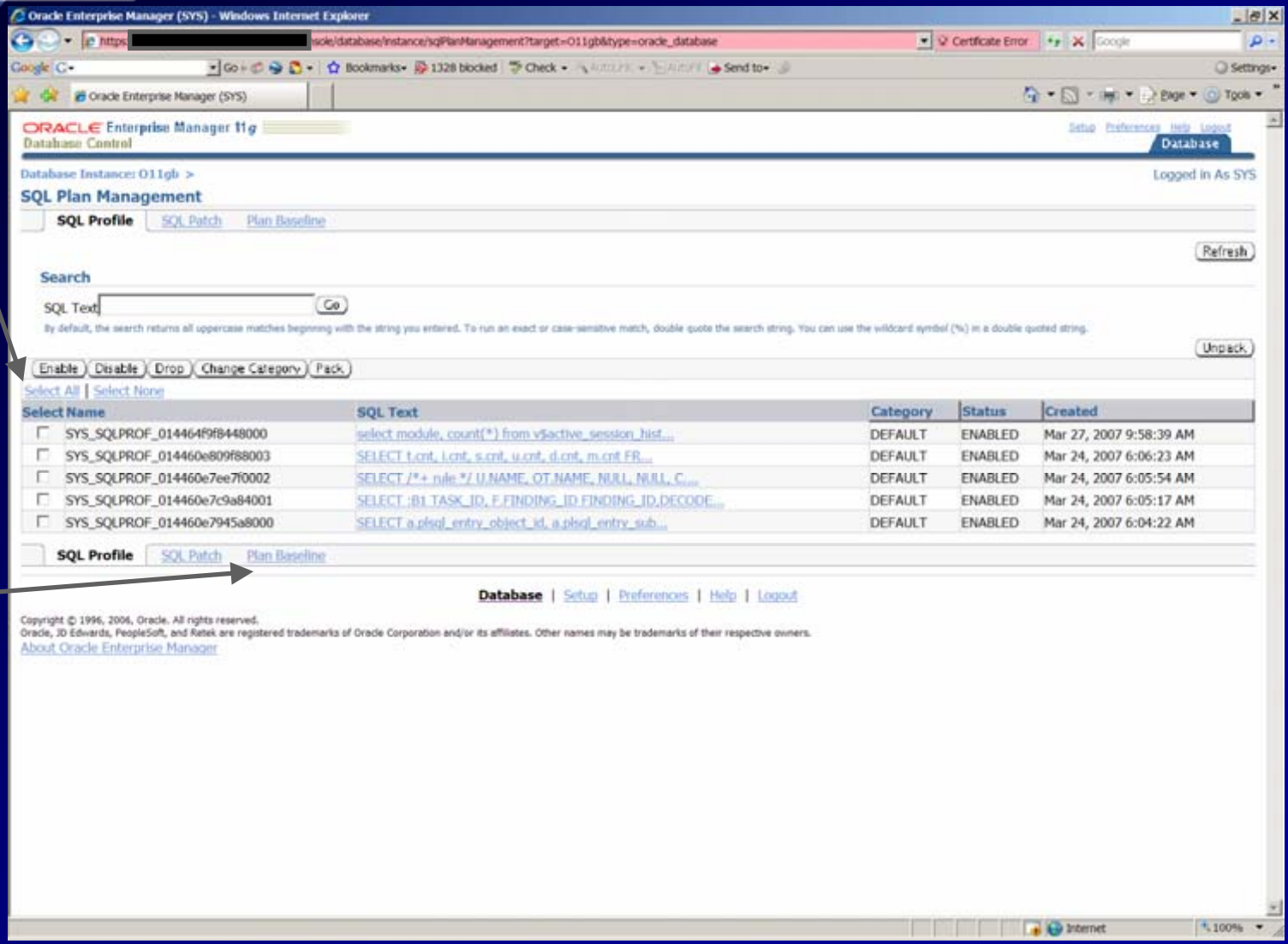
Compare Before & After



# SQL Plan Management
# SQL Profiles stored in the system

SQL Profiles

SQL Plan Baselines



# SQL Plan Management
## Capturing Baselines



- Capturing baselines (migrate stored outlines - hints)
- Plan history is only tracked for a SQL statement that executes more than once (no ad-hoc queries)
- Automatic Plan Capture:
  - OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
  - (set to TRUE – the default is FALSE)
- Enable the use of SQL Plan Baselines (could be session level of a tuning set & without the capture):
  - OPTIMIZER_USE_SQL_PLAN_BASELINES
  - (set to TRUE – the default is TRUE)

# SQL Plan Management
# Capturing Baselines

```
DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE (
        sql_id IN VARCHAR2,
        plan_hash_value IN NUMBER := NULL,
        sql_text IN CLOB,
        fixed IN VARCHAR2 := 'NO',
        enabled IN VARCHAR2 := 'YES')
 RETURN PLS_INTEGER;
```

- **Load one or more plans present in the cursor cache for a SQL statement.** You can also do this using the plan_handle and sql_text.

# SQL Plan Management
# Capturing Baselines

```
DBMS_SPM.LOAD_PLANS_FROM_SQLSET (
        sqlset_name IN VARCHAR2,
        sqlset_owner IN VARCHAR2 := NULL,
        basic_filter IN VARCHAR2 := NULL,
        fixed IN VARCHAR2 := 'NO',
        enabled IN VARCHAR2 := 'YES'
        commit_rows IN NUMBER := 1000)
RETURN PLS_INTEGER;
```

- **Manually load plans stored in SQL Tuning Sets (STS) into plan baselines.**
- Note that plan history is only tracked for a SQL statement that executes more than once (no ad-hoc queries)

# SQL Plan Management
# Capturing Baselines

- Set the retention to 100 weeks to retain unused plans before they are purged (53 is the default). Shown for example, 100 is not recommended, 53 is better since it will include monthly/yearly runs.

```
begin
   dbms_spm.configure('plan_retention_weeks',100);
end;
/
```

*PL/SQL procedure successfully completed.*

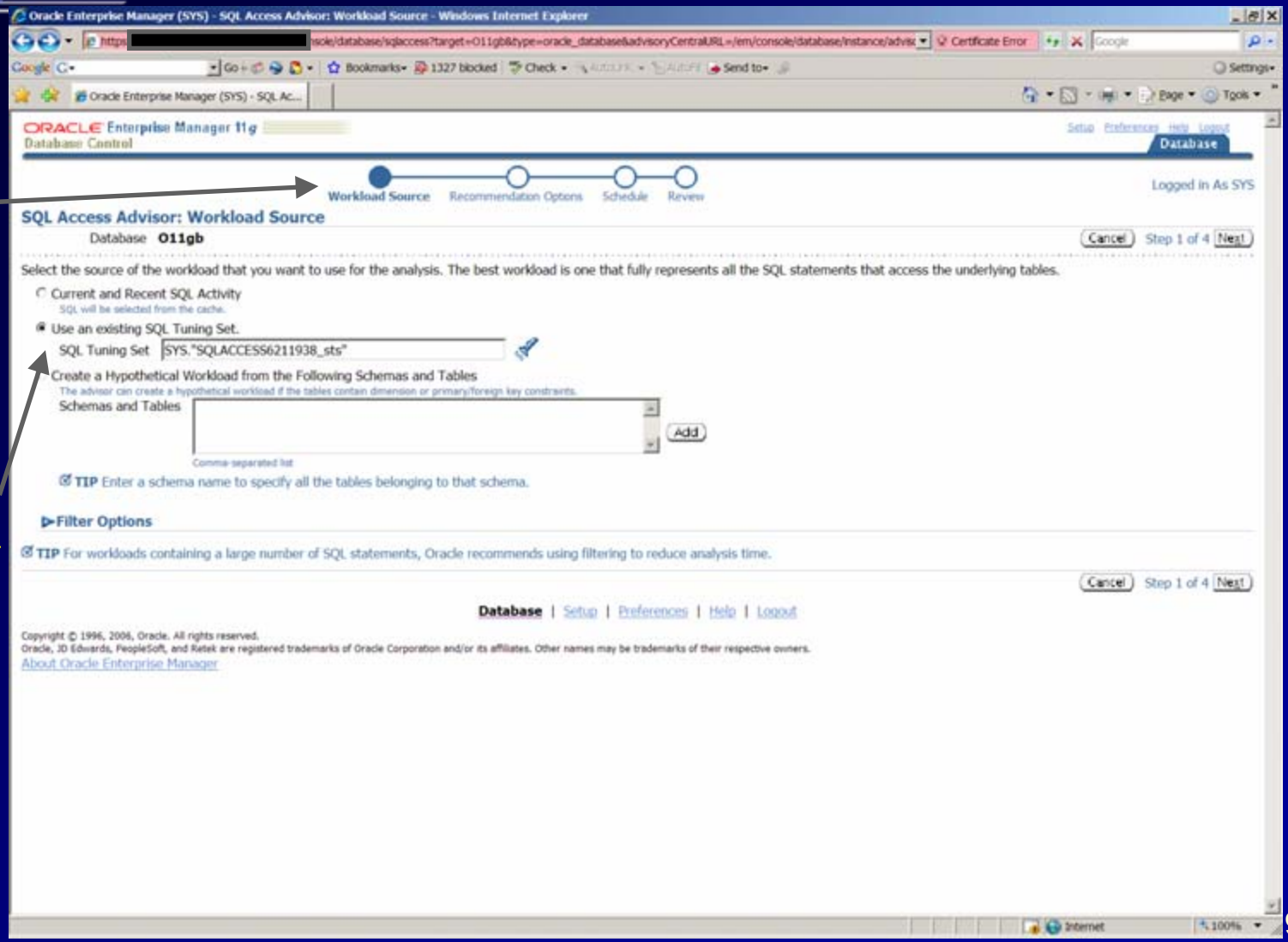
- You can also purge individual plans with the purge_sql_plan_baseline function.
- You can also query dba_sql_plan_baselines as well as use DBMS_XPLAN.display_sql_plan_baseline to view stored plans.

# SQL Access Advisor
# NEW Partition Advisor

# SQL Access Advisor & NEW Partition Advisor

- The SQL Advisor now combines the functionality of the SQL Tuning Advisor, SQL Access Advisor and the new Partition Advisor.
  - Recommends Partitioning Needs
  - Utilize a previous SQL Tuning Set
  - Take SQL straight from what's currently in the CACHE.
  - Create a hypothetical workload
  - SQL Access Advisor checks Indexes, Partitions or Materialized Views (**schema related issues**)

# SQL Access Advisor & NEW Partition Advisor



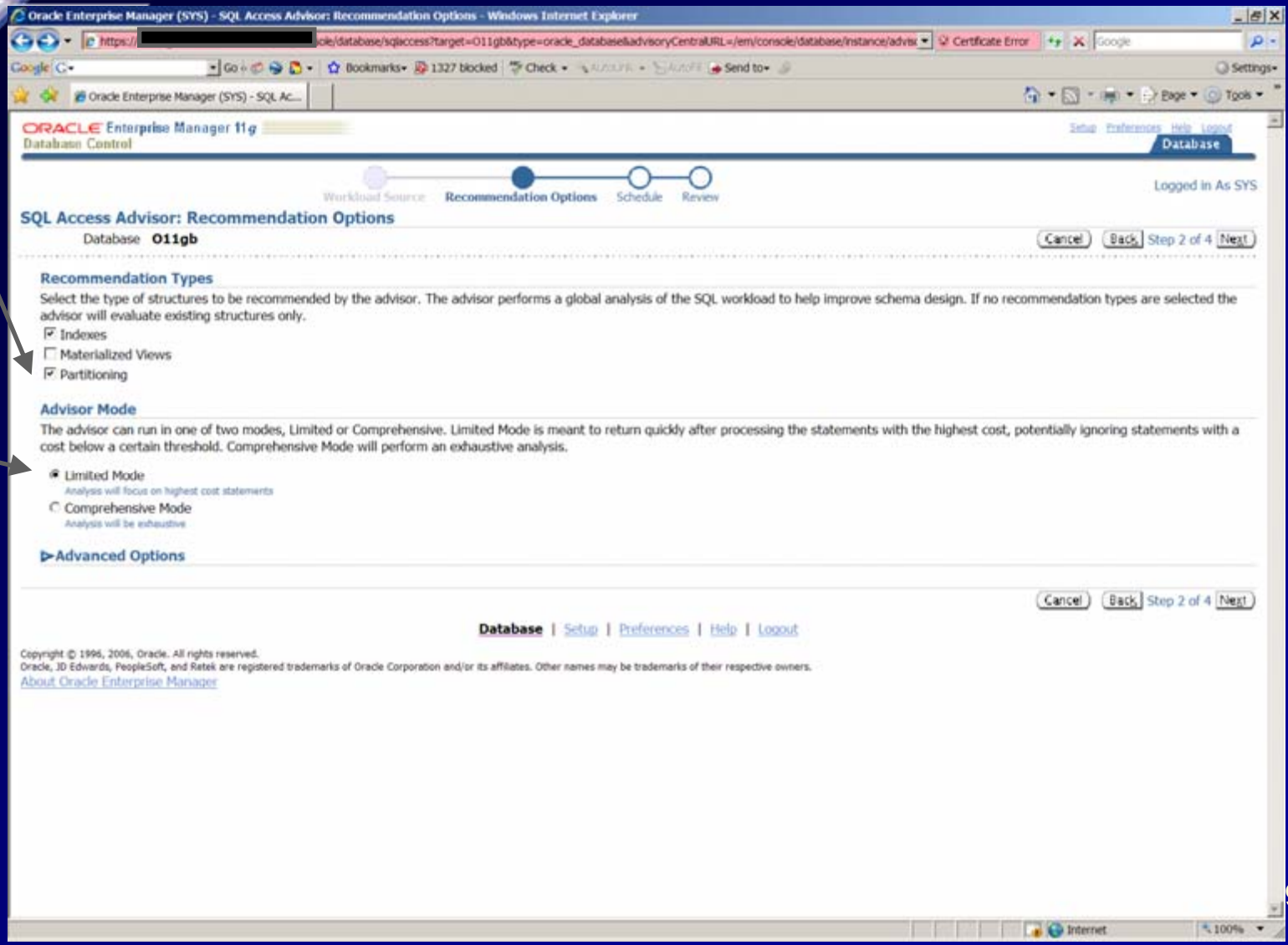

Step One

Use a SQL Tuning Set

# SQL Access Advisor & NEW Partition Advisor

Look at Partitions

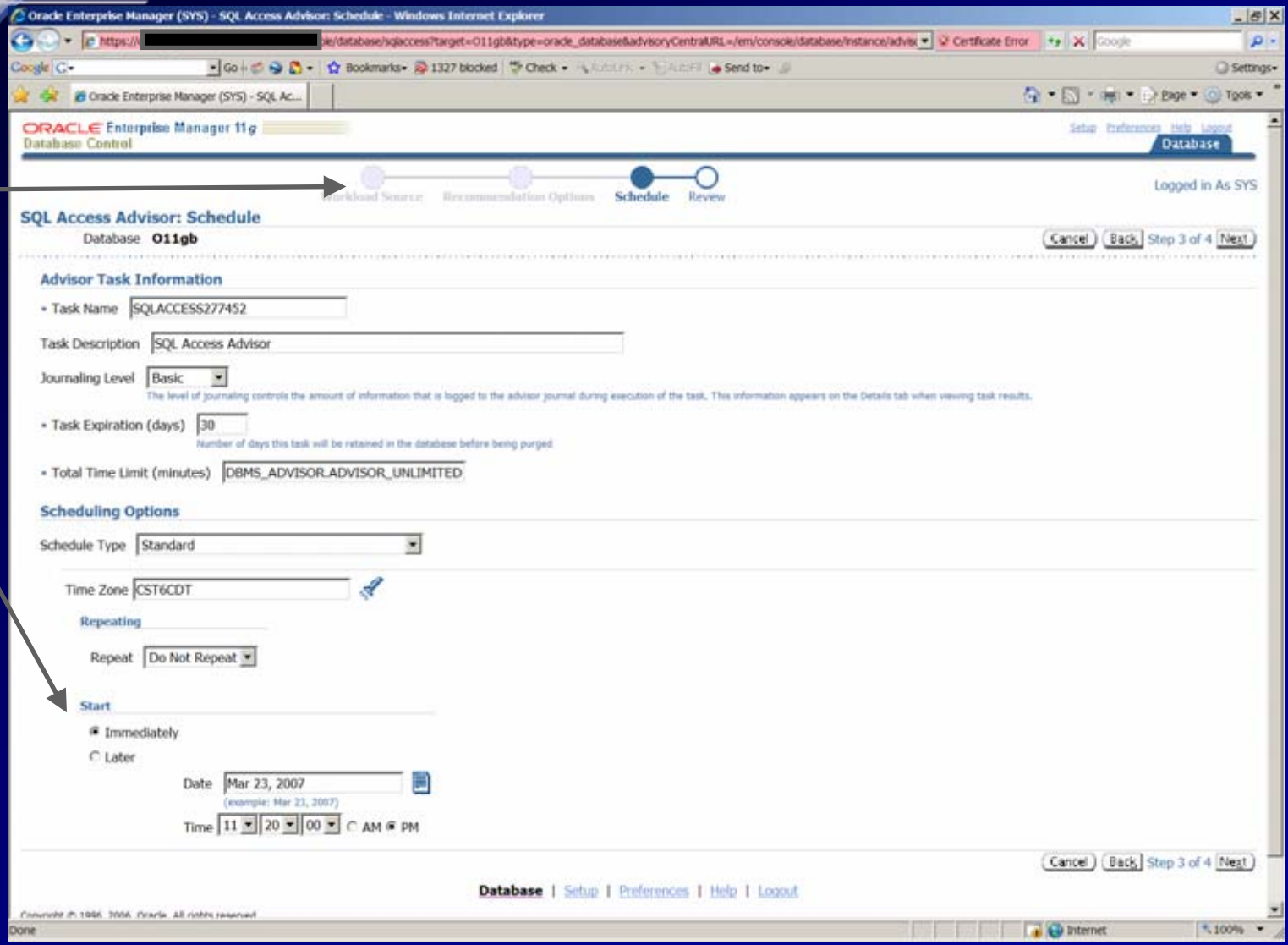
Quick Solution



# SQL Access Advisor & NEW Partition Advisor

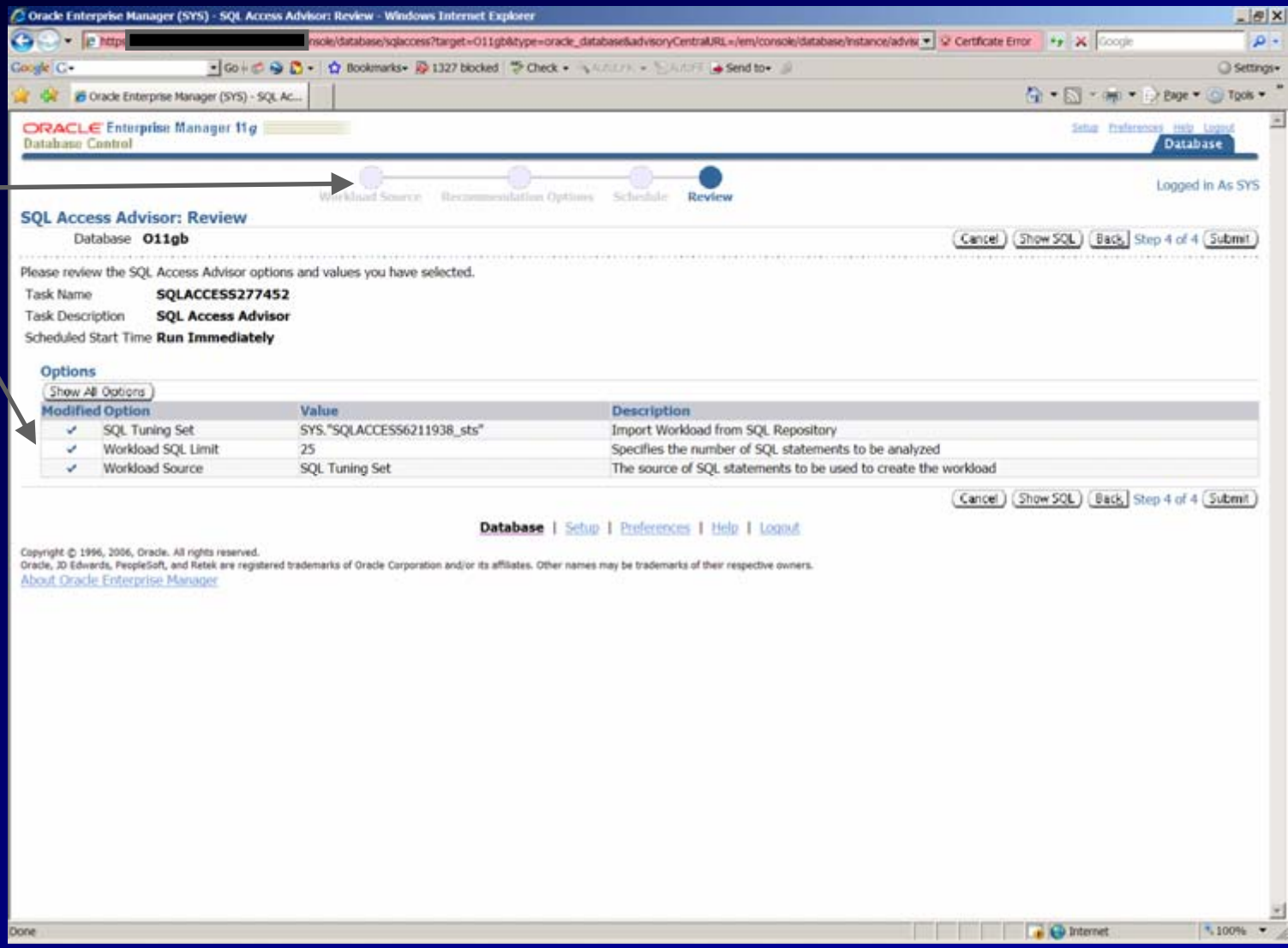
Step 3 Schedule it.

Run it Now!



# SQL Access Advisor & NEW Partition Advisor

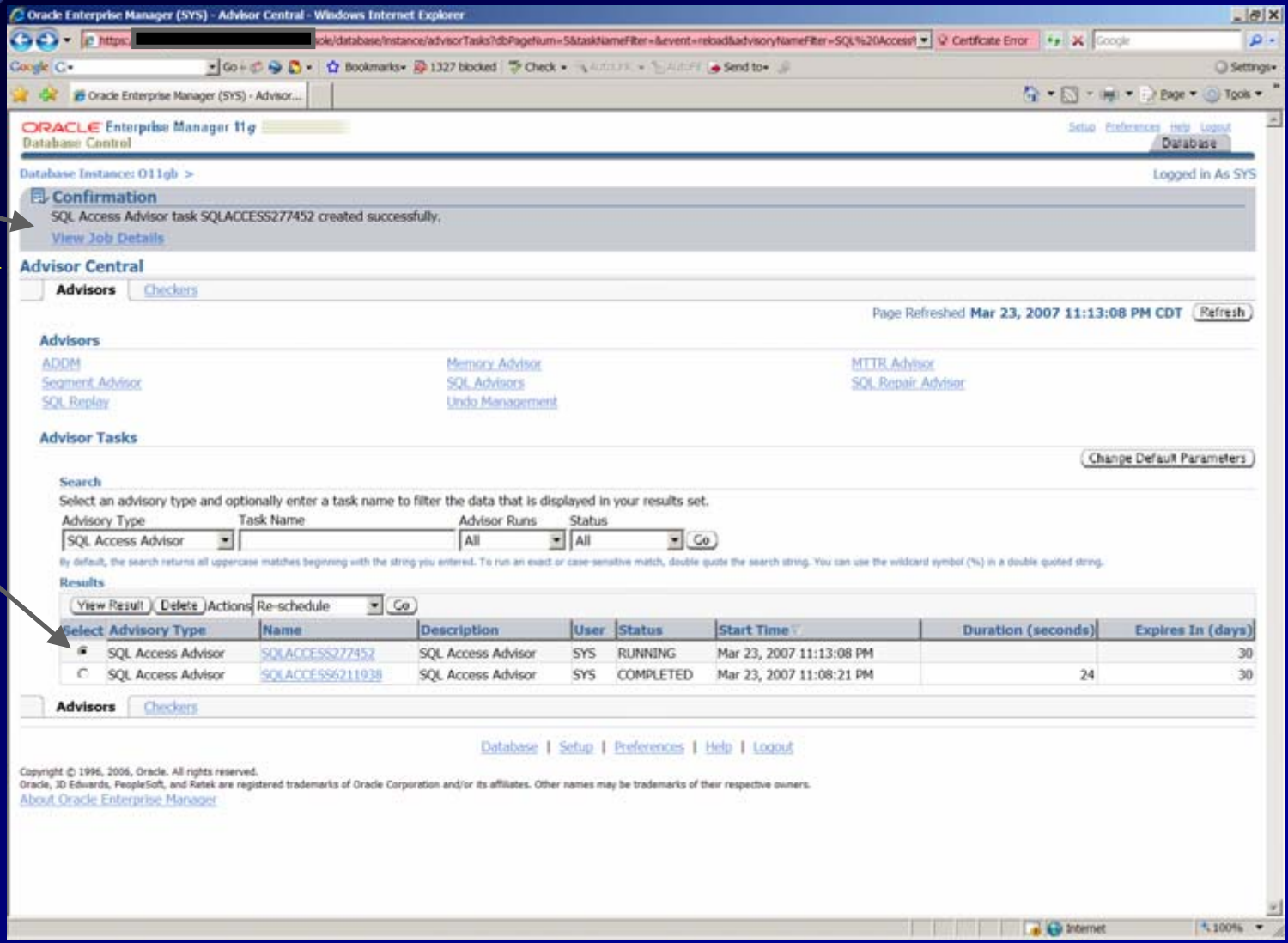
Final Review



# SQL Access Advisor & NEW Partition Advisor

Job Submitted

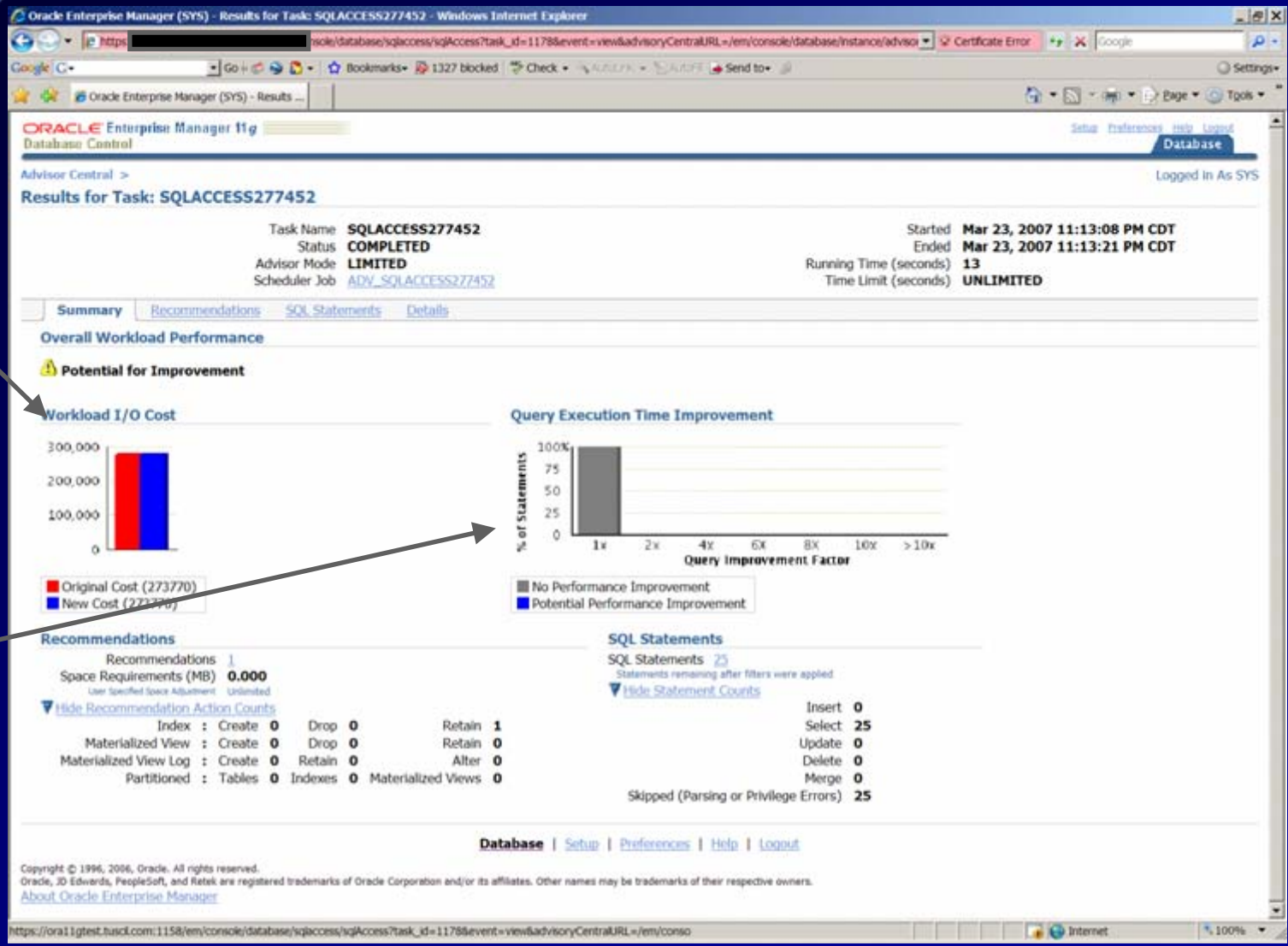
Job Running Now.



# SQL Access Advisor & NEW Partition Advisor

Improve I/O

Improve Execution Time



# The SQL Repair Advisor


**ORA-600**

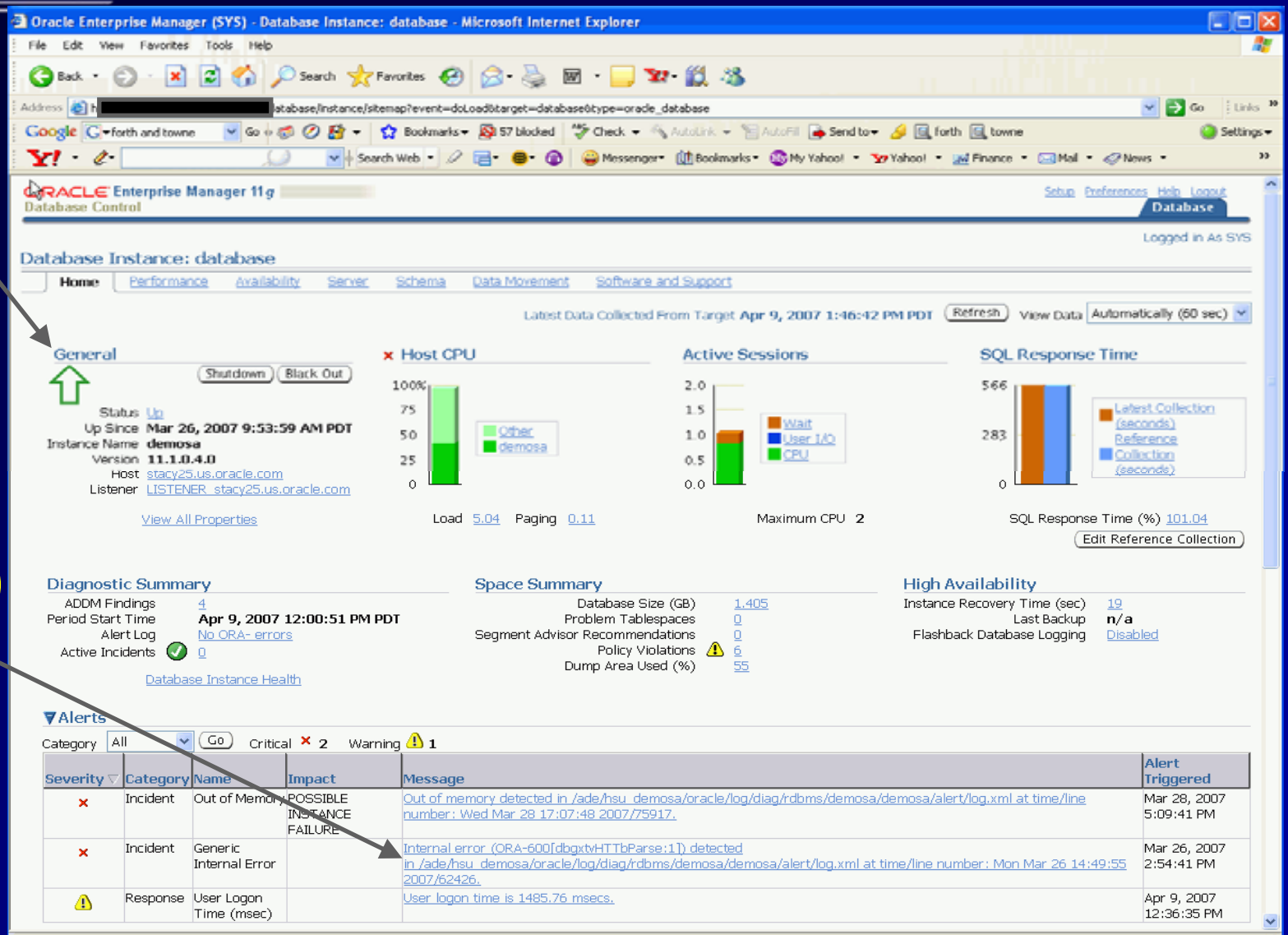
# SQL Repair Advisor

- 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

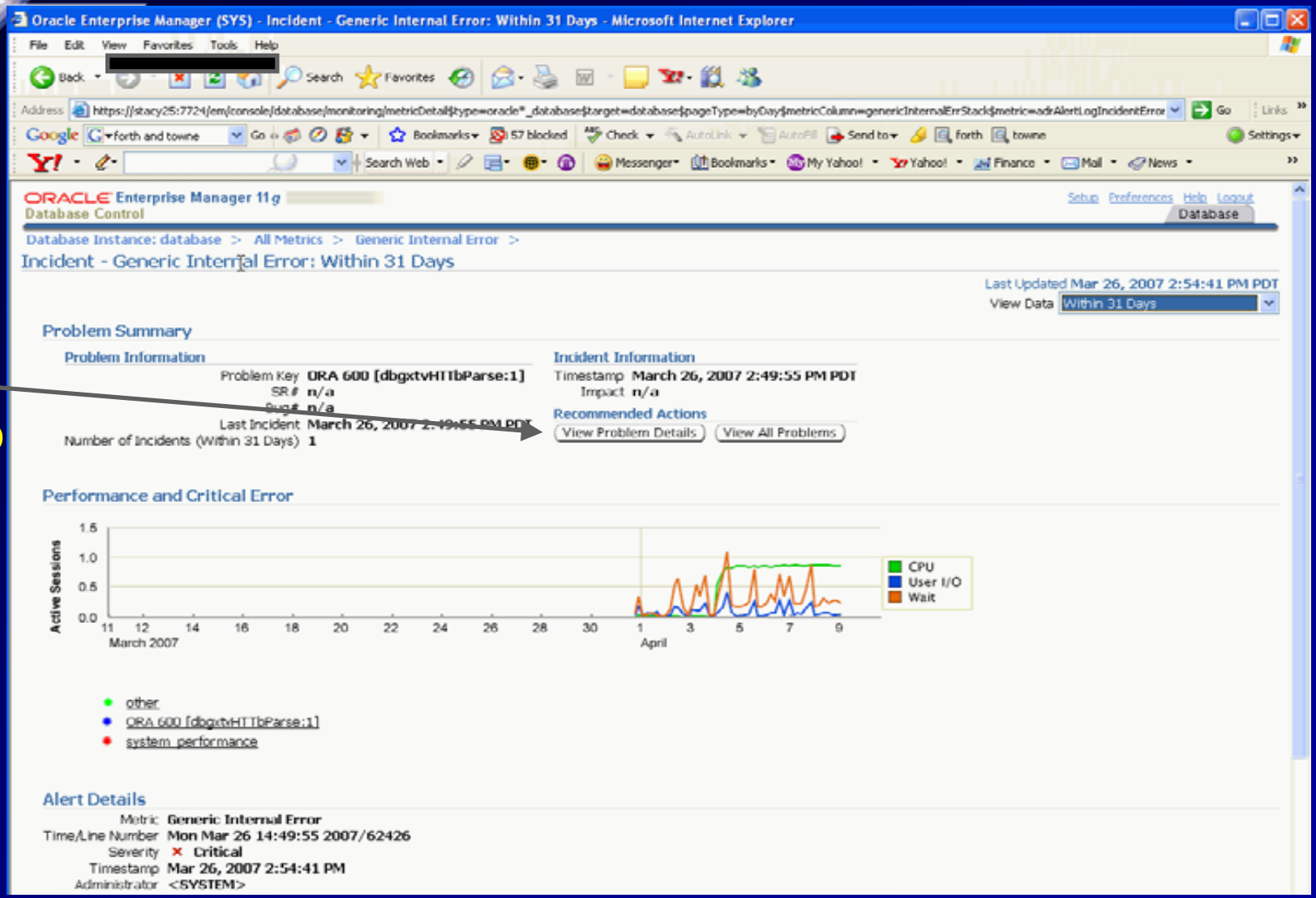
Go to the Database Instance

Click Alert (ORA-600) message text to see details



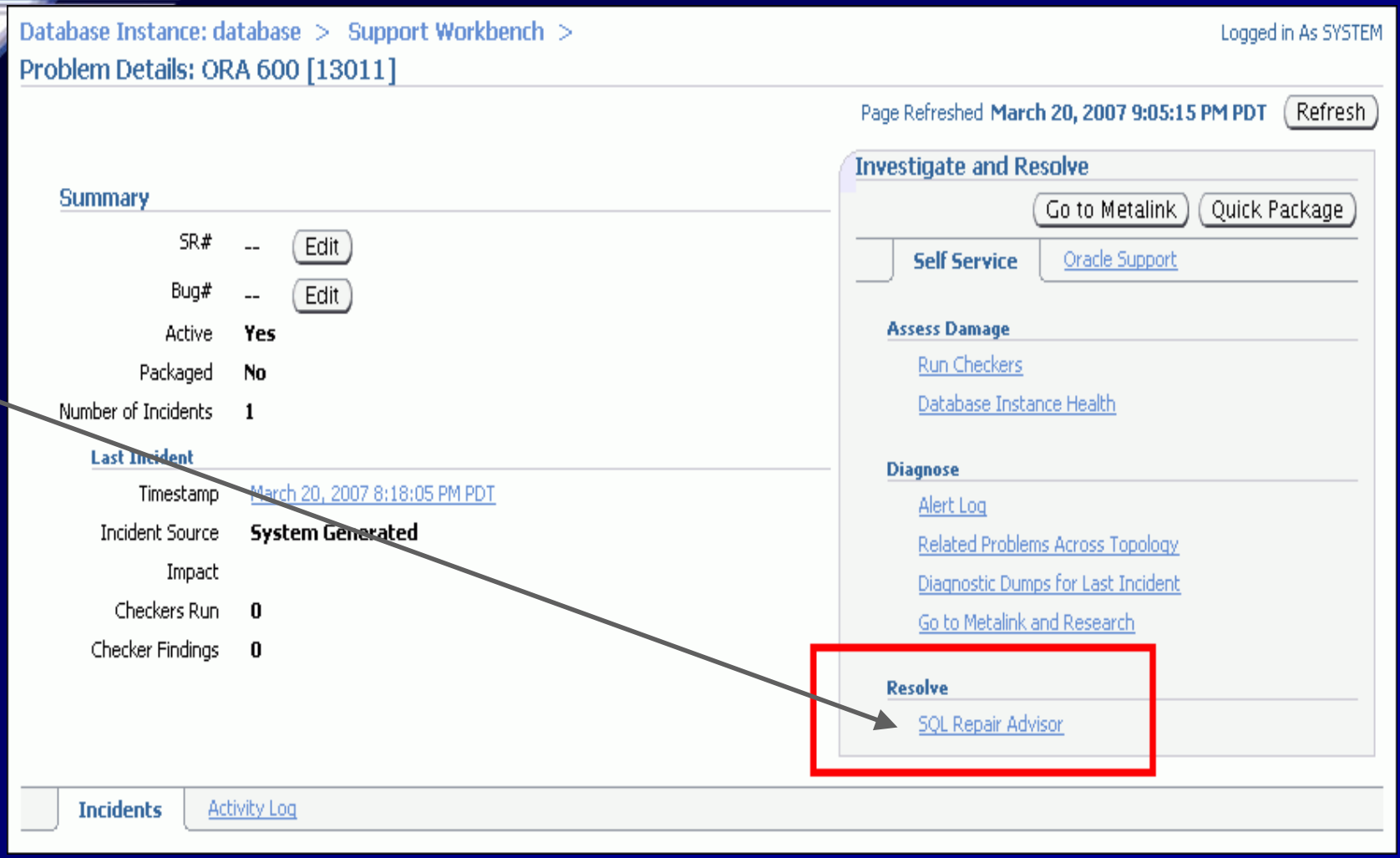
# SQL Repair Advisor – View Problem Details

Click on View Problem Details to go to the Support Bench



# Support Workbench - Details

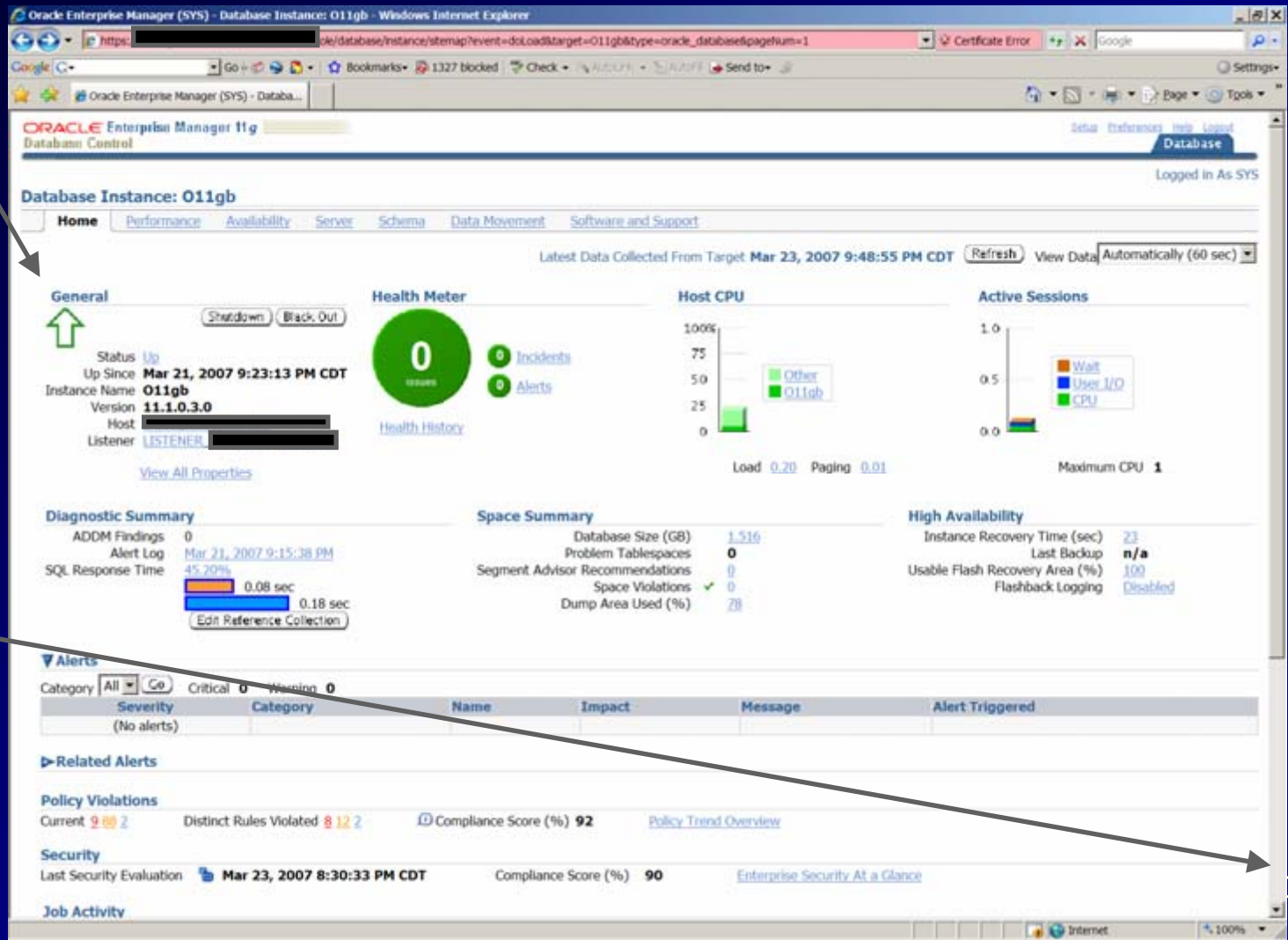
Click on SQL Repair Advisor



# SQL Repair Advisor – 2nd Method
## Go Into Advisor Central

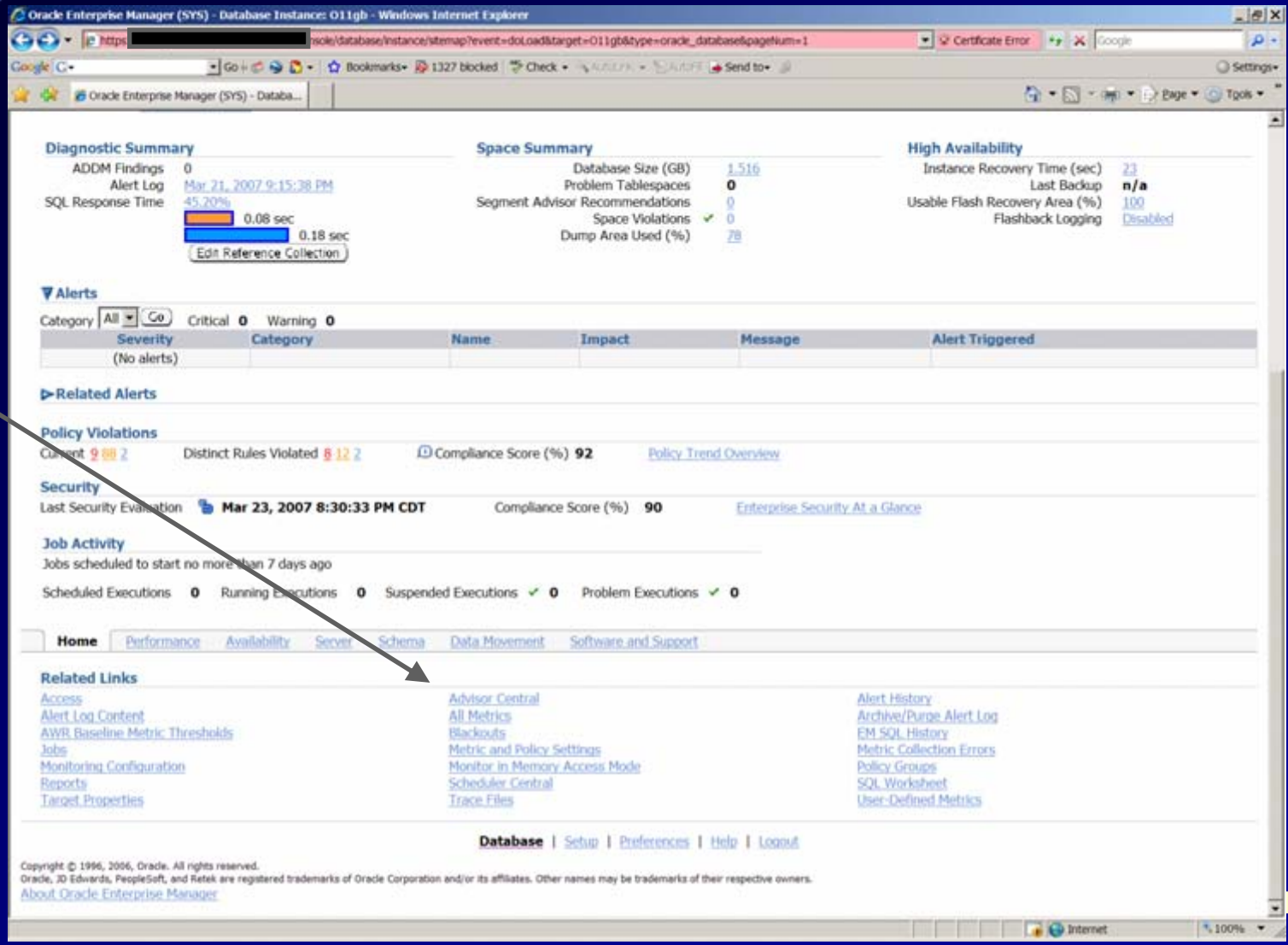
Go to the Database Instance

Scroll Down



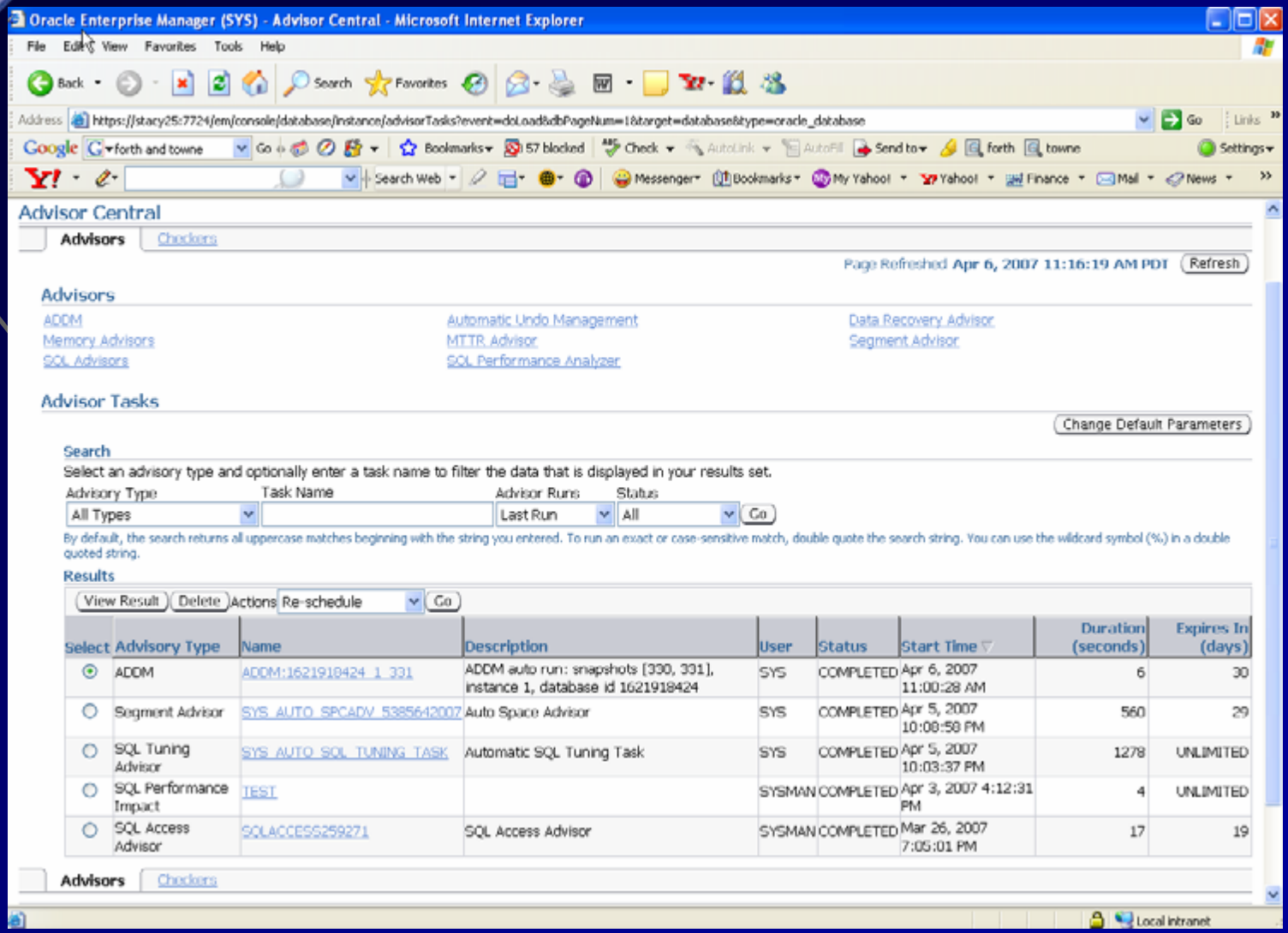
# SQL Repair Advisor – Go Into Advisor Central

Click on The Advisor Central



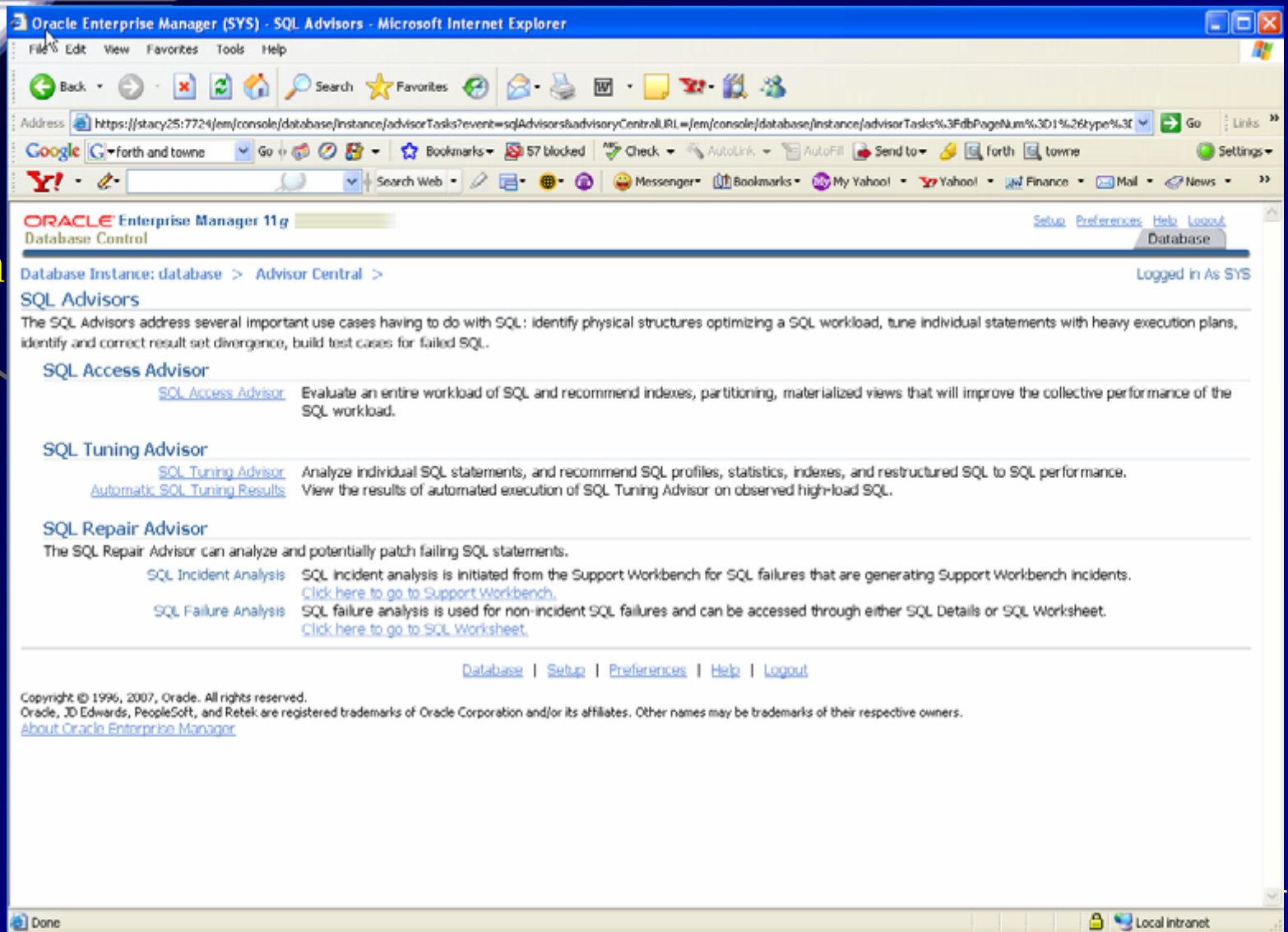
# SQL Repair Advisor – Advisor Central

Click on the SQL Advisor



# SQL Repair Advisor – SQL Advisors

Click on The Support Bench

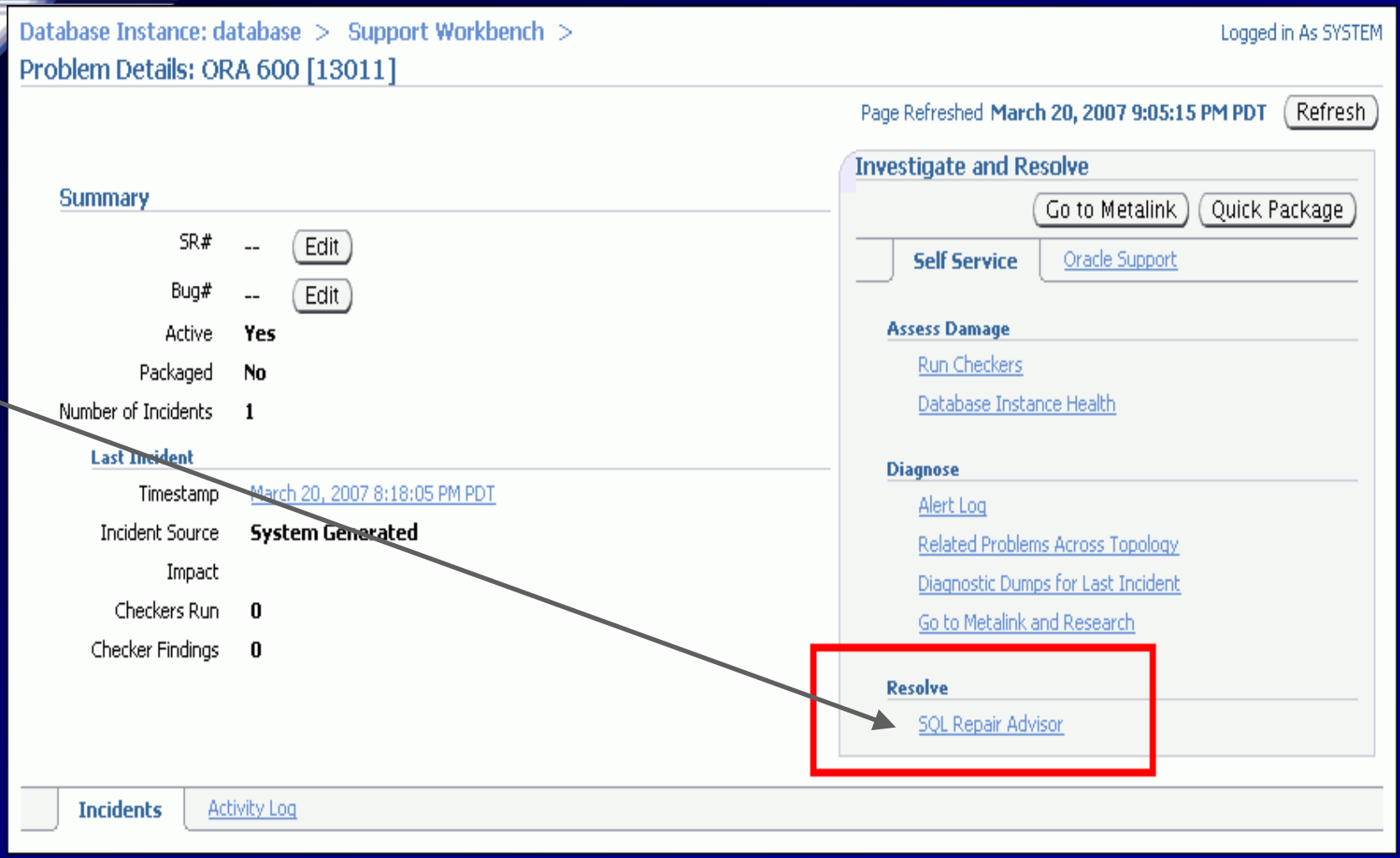


# SQL Repair Advisor – Support Workbench

Click on The ID number to Open the detail

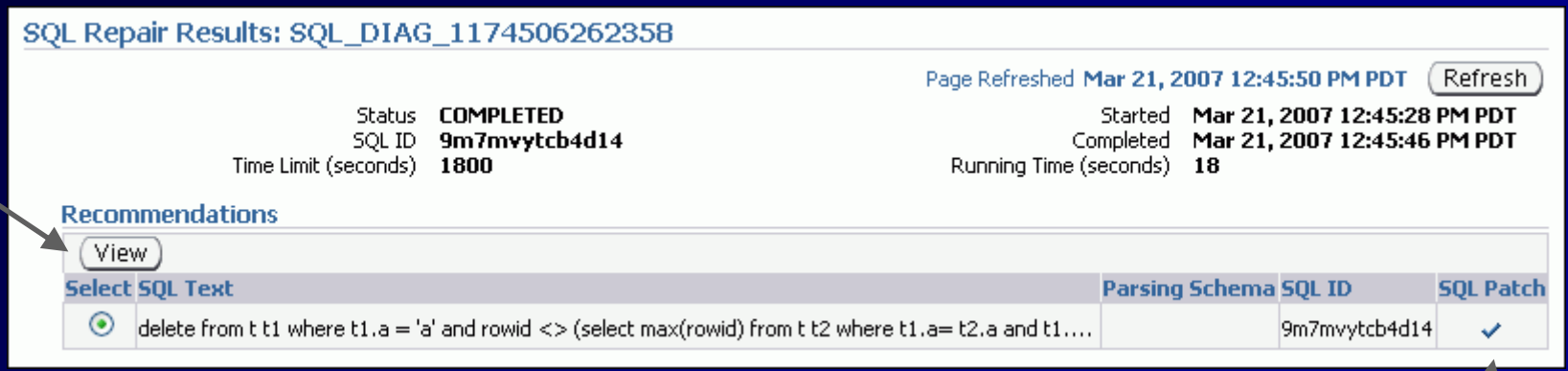
# SQL Repair Advisor – Support Workbench Details

Click on SQL Repair Advisor to run it



# Results from SQL Repair Advisor

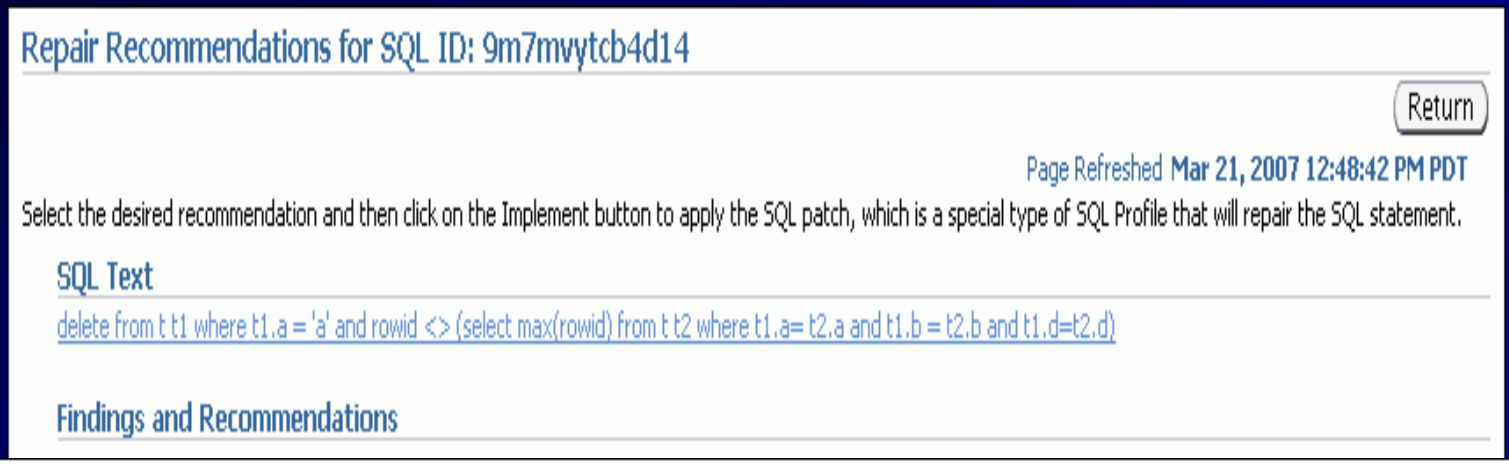
Click on View to Get the Detail finding of the Advisor

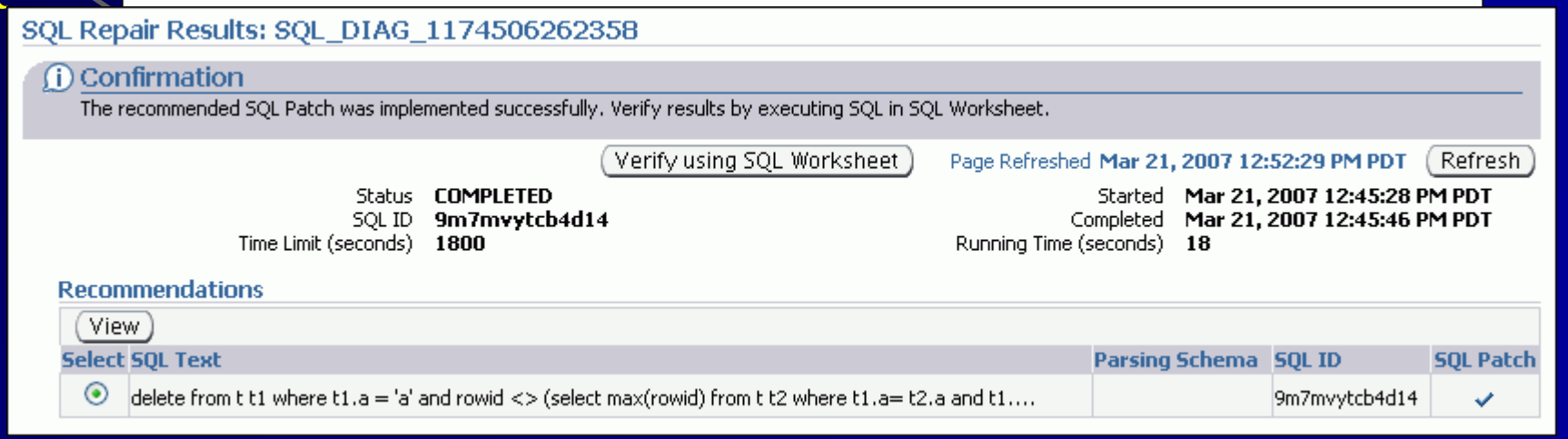



Note a SQL Patch (FIX for the SQL) has been generated

# SQL Repair Advisor
## Recommendation / Confirmation

Click on Implement To accept the SQL Patch



# SQL Replay Advisor
# SQL Performance Analyzer (renamed)

# SQL Replay Advisor

- 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

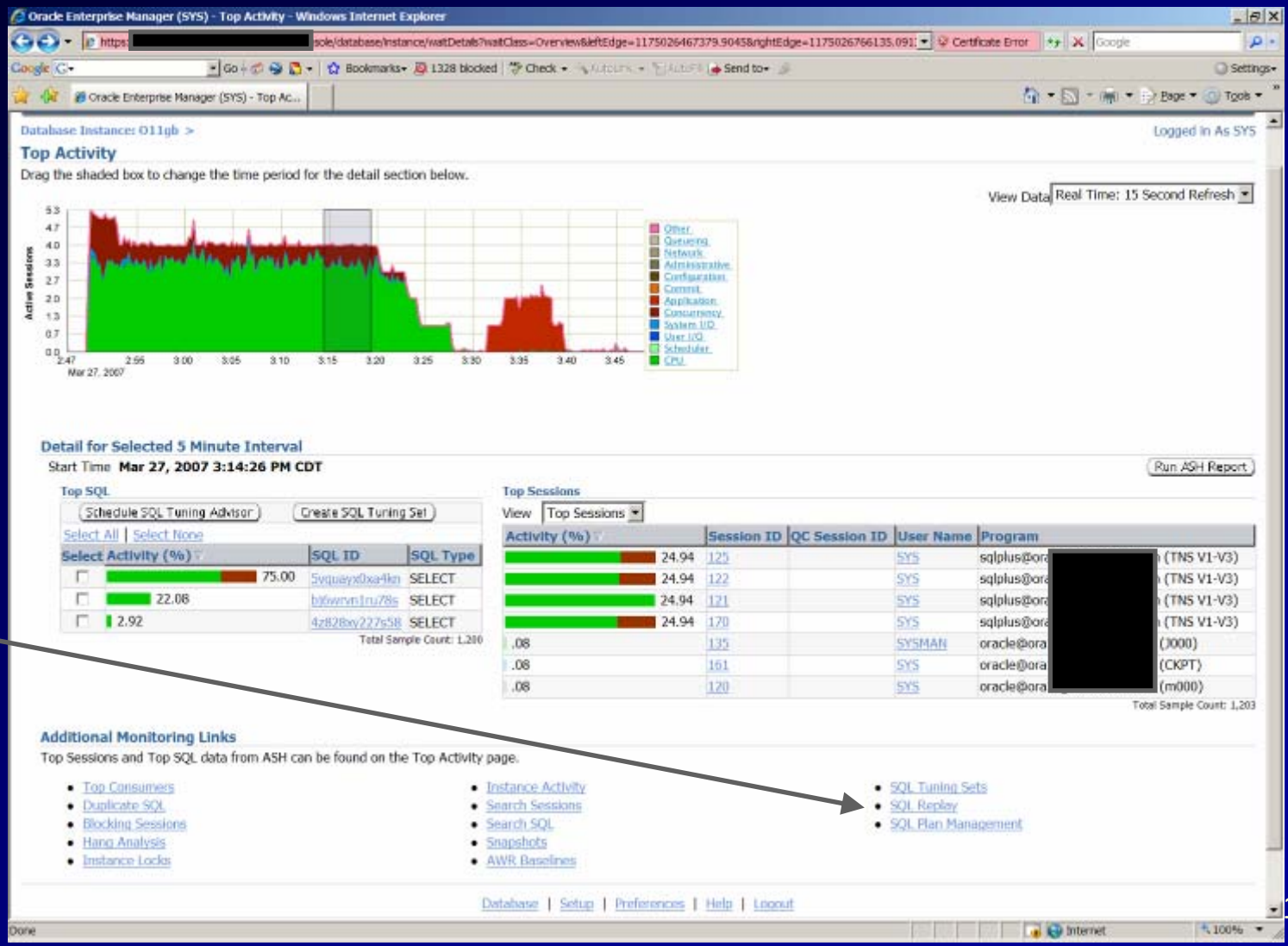
# SQL Replay Advisor



Easy to run – SQL Focus (Test SGA settings):

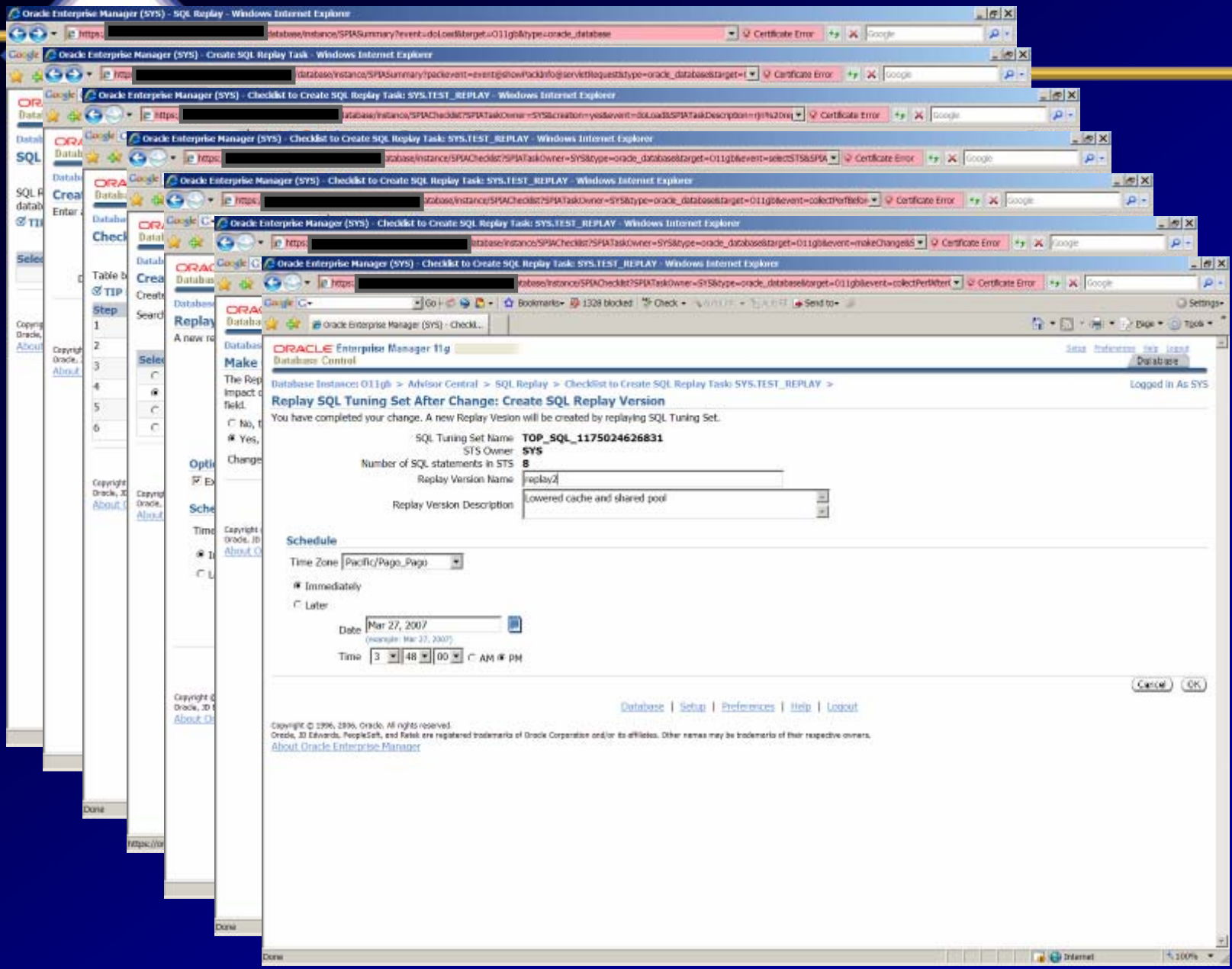
- Capture SQL
- Transport SQL
- Create a Replay Task
- Set up the environment to Test
- Make any changes to Test (such as SGA changes)
- Compare before and after performance
- Tune the problems!

# SQL Replay Advisor

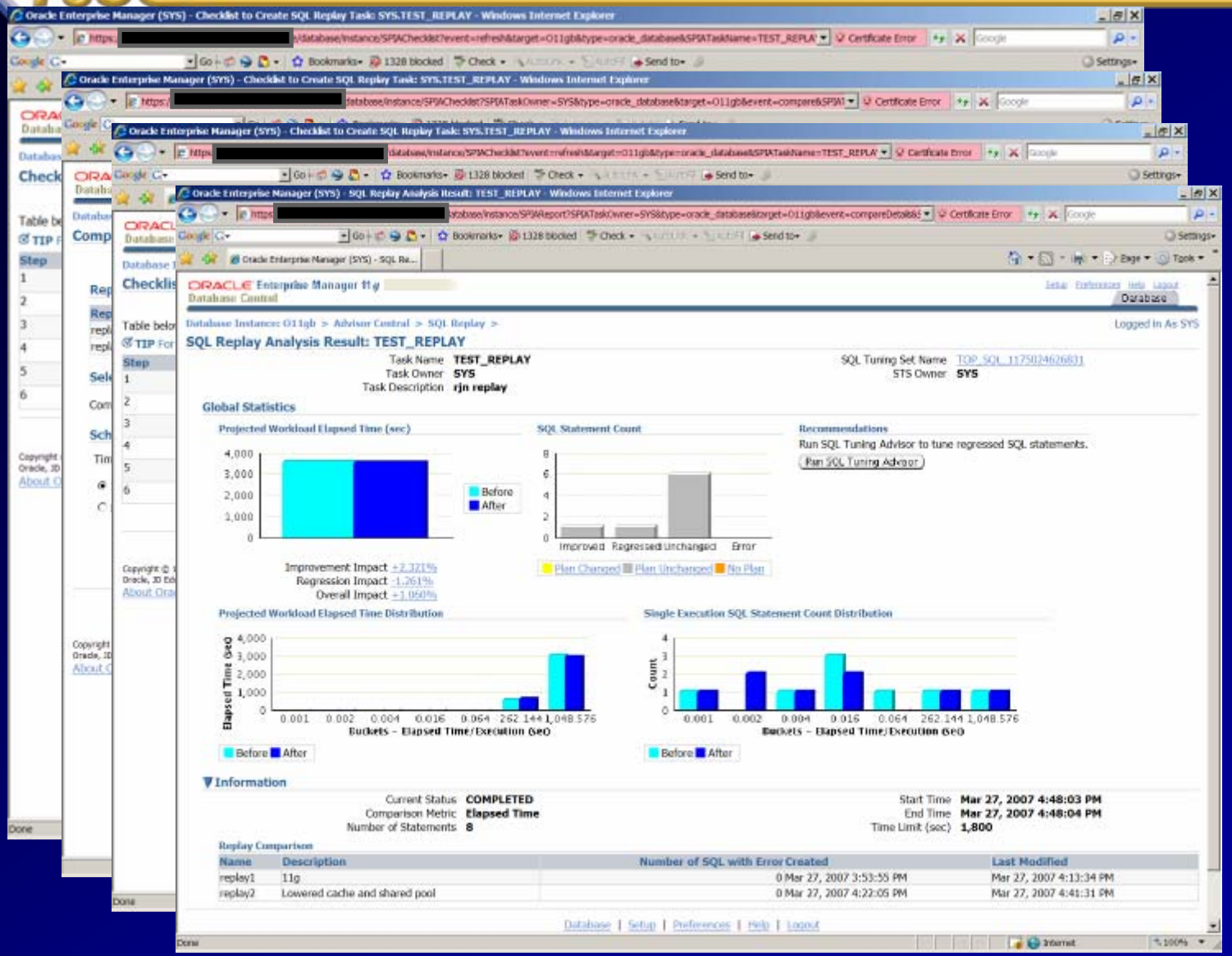



SQL Replay

# SQL Replay Advisor



# SQL Replay Advisor



# Database workload capture and replay


# Database workload capture and replay

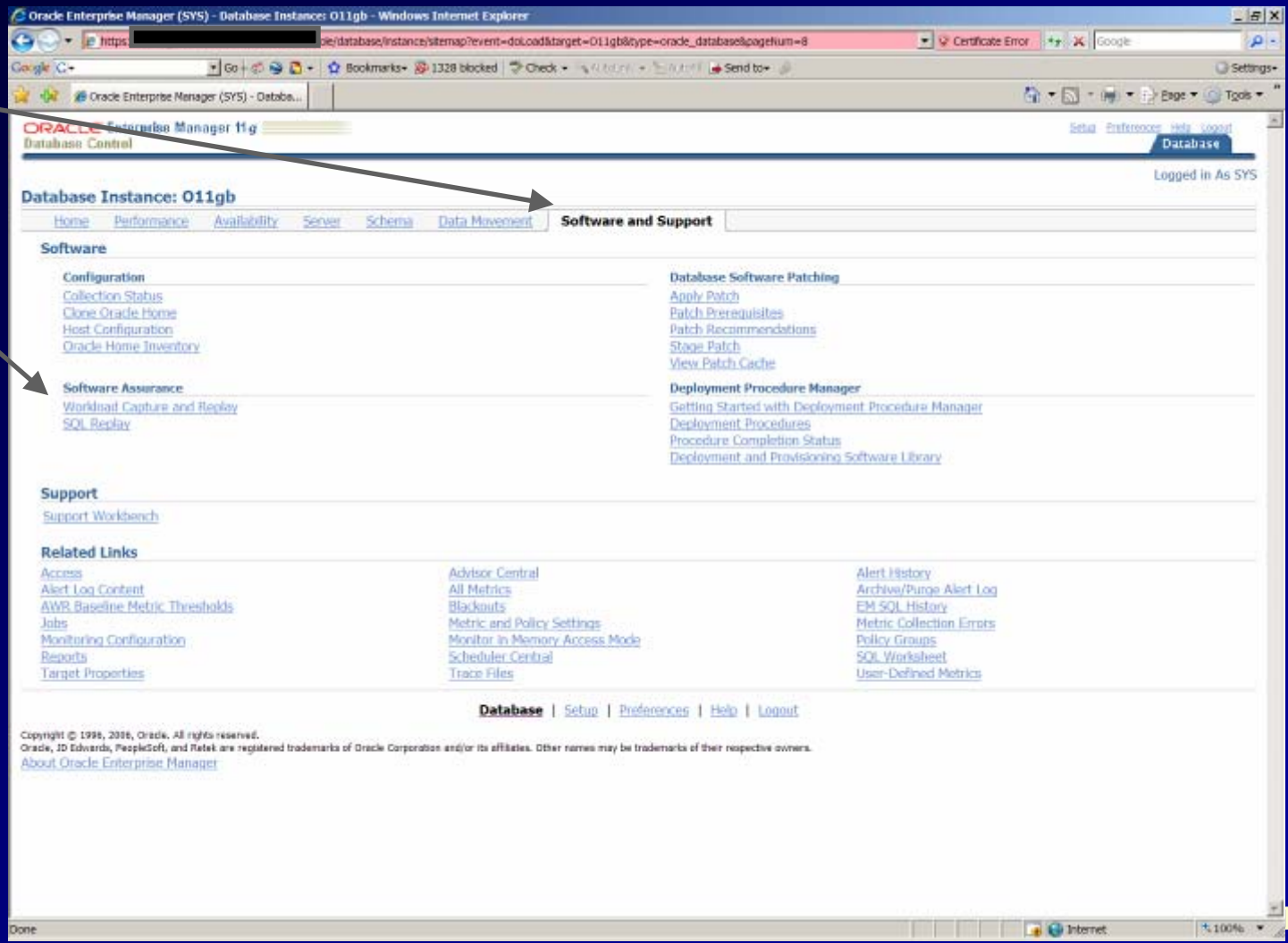
- Used to capture **database workload** on one system and replay later on a different system. Useful to compare two different systems.
- Could rival LoadRunner in the future (may be more precise!)

Brief Steps:

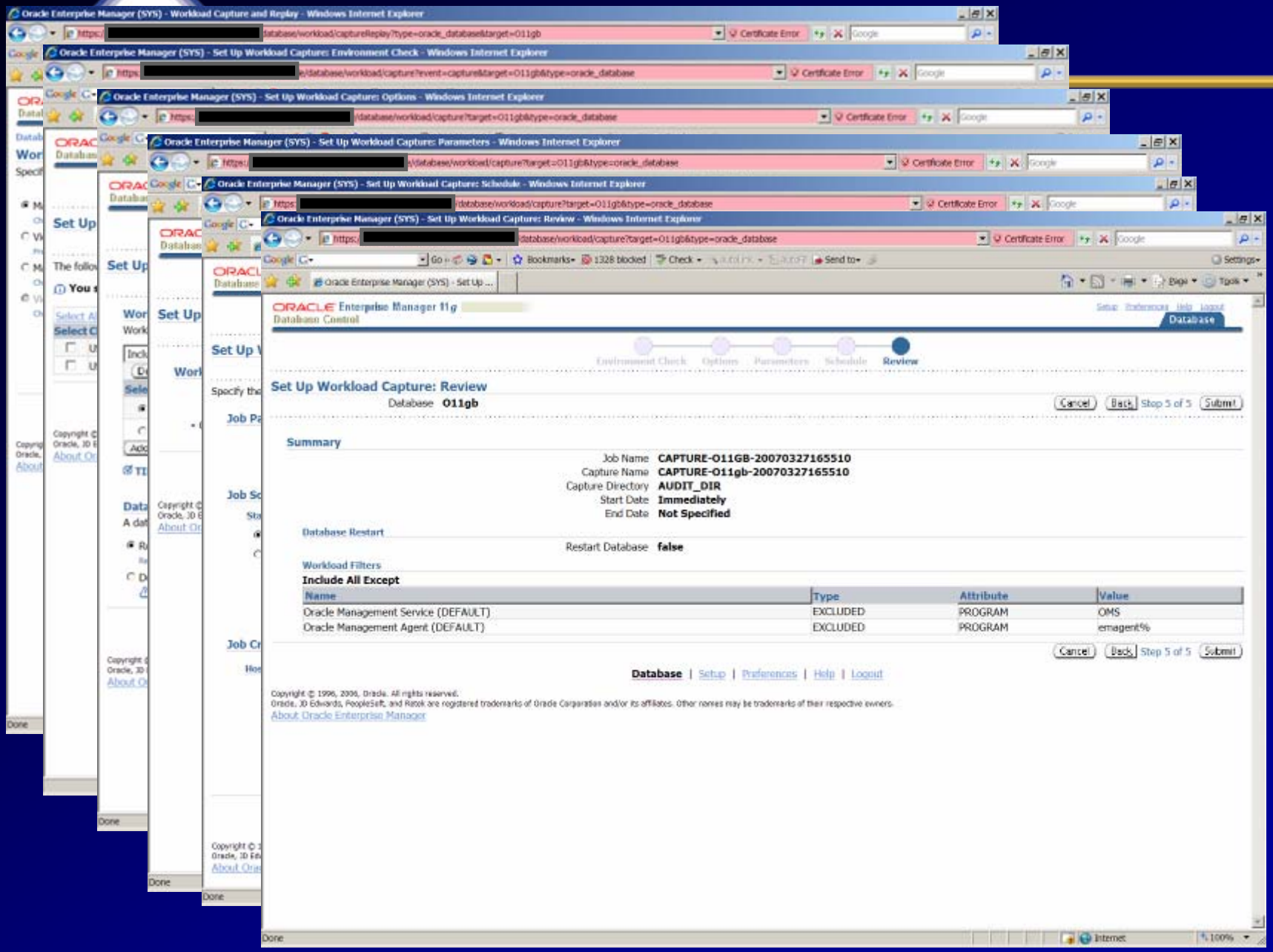
- Capture workload on a database (We'll see how to do this)
- Restore the database on a test system to the SCN when capture begins
- Perform upgrade and make changes to the test system as needed
- Preprocess the captured workload if it is not preprocessed
- Configure the test system for replay (I don't 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 an upgrade to 11g (Test with 10g then 11g)

# Database workload capture and replay

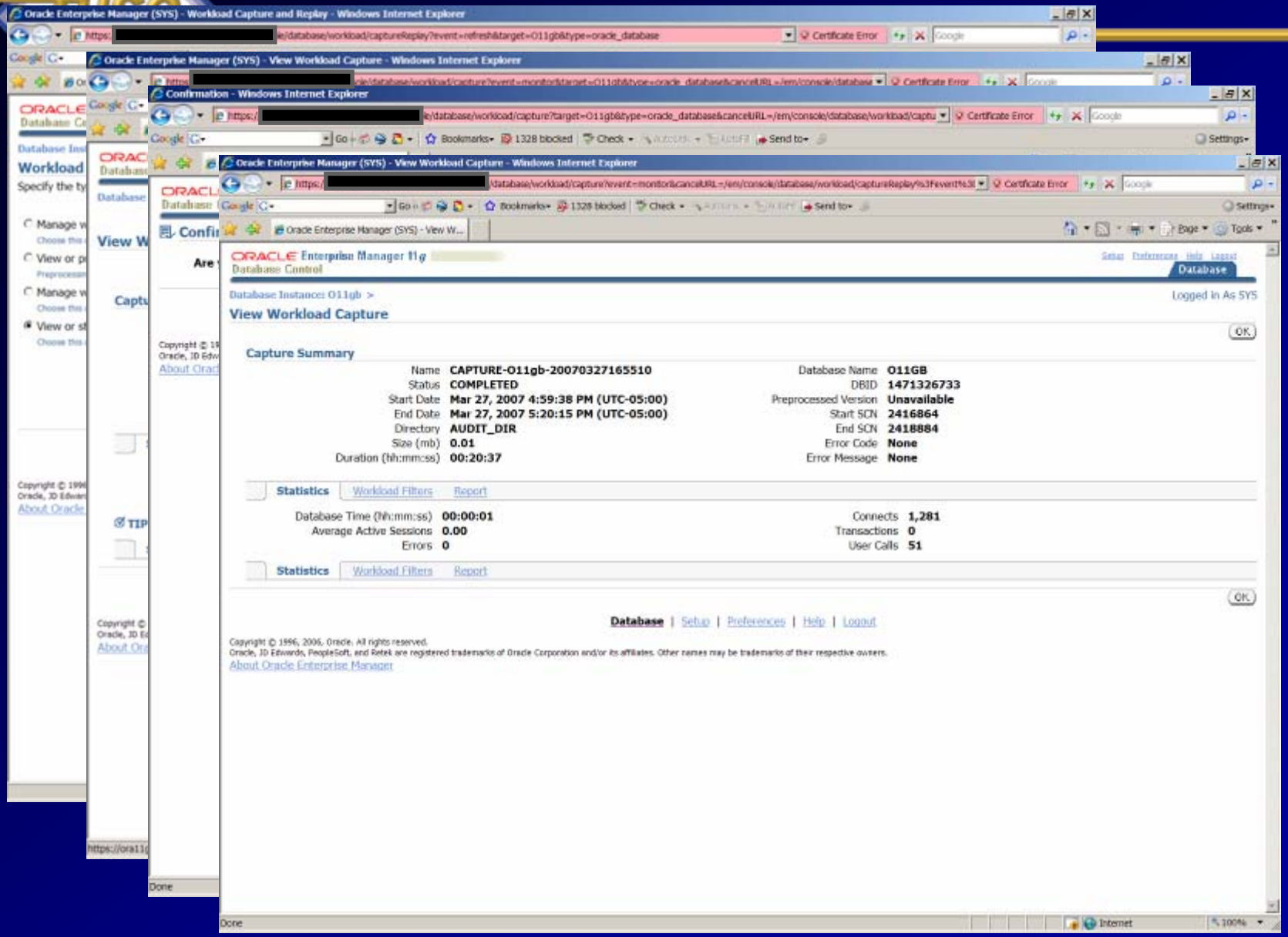
Workload Capture and Replay



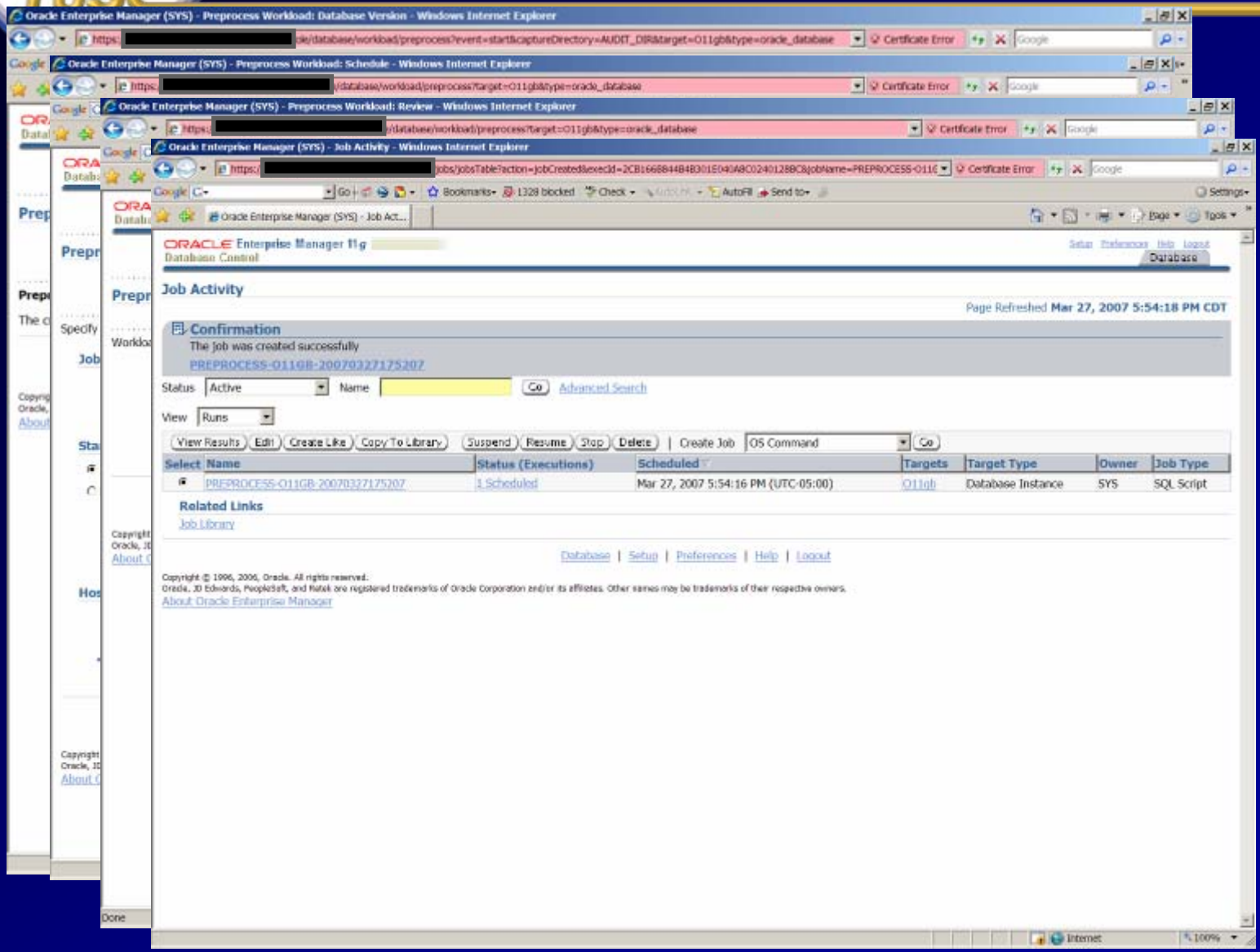
# Database workload capture and replay



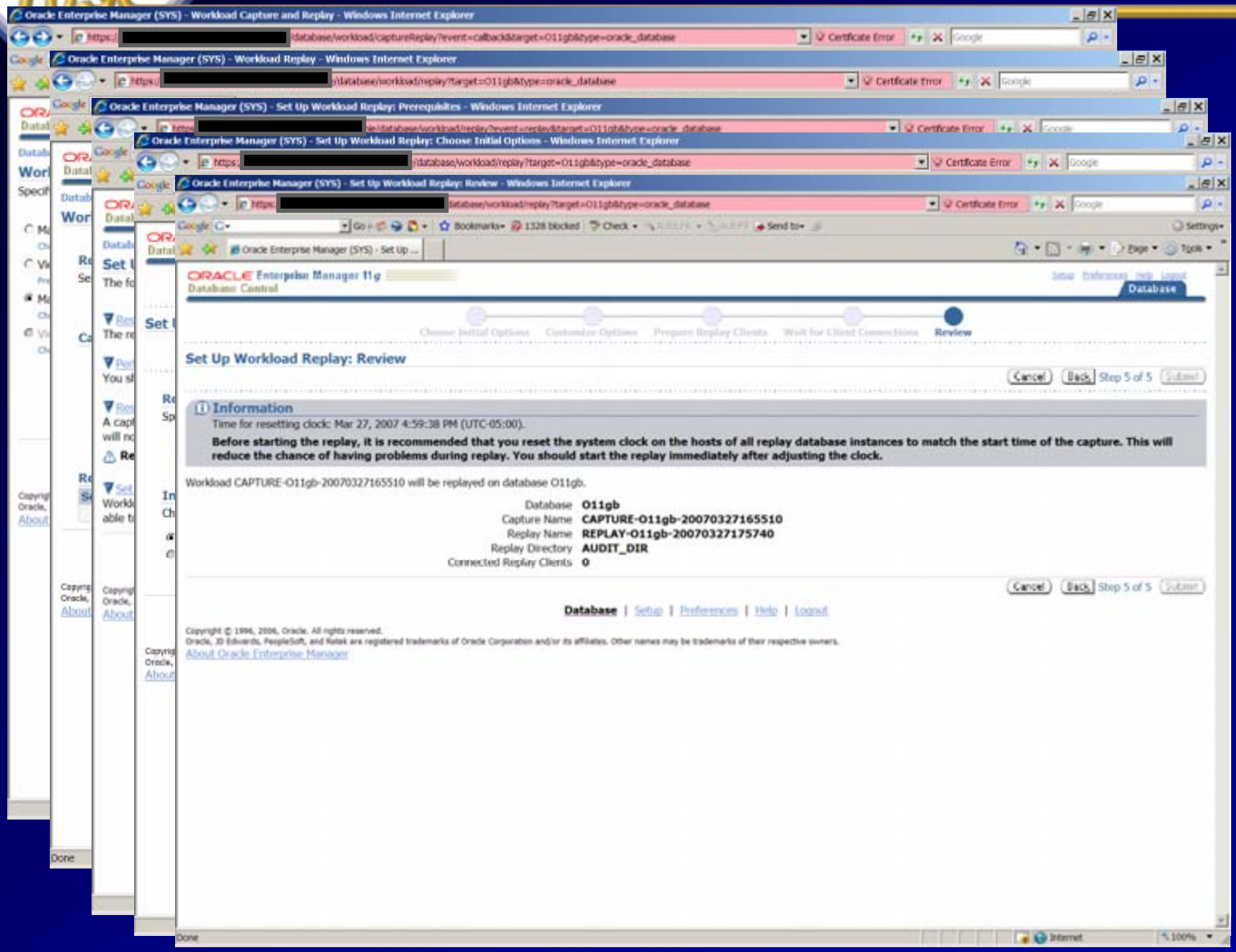
# Database workload capture and replay



# Database workload - Preprocess



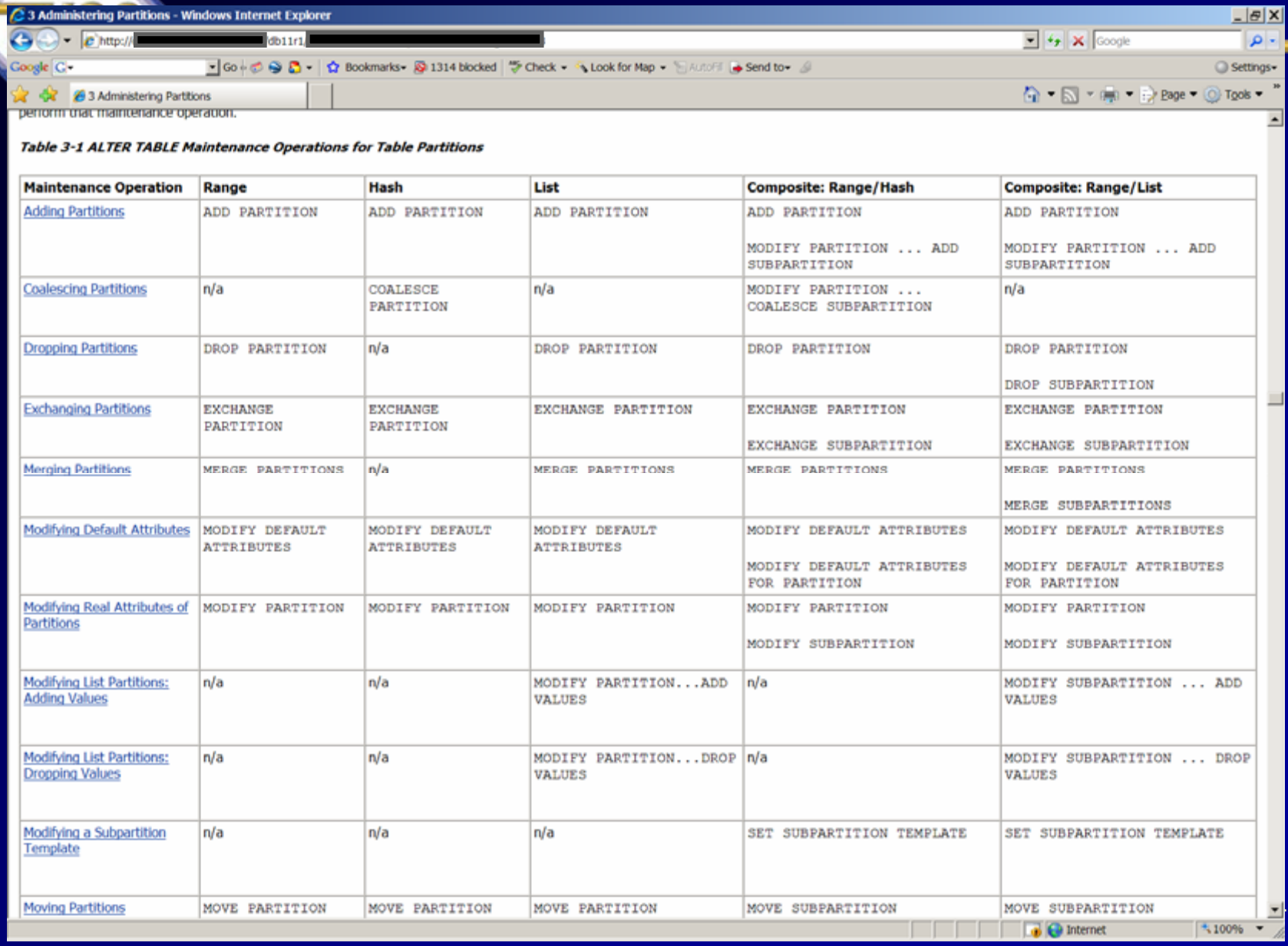
# Database workload capture and replay On the NEW system (shortened steps)



# Partitioning: (FYI & 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

# The Rules – See Partitioning Guide




perform that maintenance operation.

***Table 3-1 ALTER TABLE Maintenance Operations for Table Partitions***

| Maintenance Operation | Range | Hash | List | Composite: Range/Hash | Composite: Range/List |
|---|---|---|---|---|---|
| Adding Partitions | ADD PARTITION | ADD PARTITION | ADD PARTITION | ADD PARTITION<br><br>MODIFY PARTITION ... ADD SUBPARTITION | ADD PARTITION<br><br>MODIFY PARTITION ... ADD SUBPARTITION |
| Coalescing Partitions | n/a | COALESCE PARTITION | n/a | MODIFY PARTITION ... COALESCE SUBPARTITION | n/a |
| Dropping Partitions | DROP PARTITION | n/a | DROP PARTITION | DROP PARTITION | DROP PARTITION<br><br>DROP SUBPARTITION |
| Exchanging Partitions | EXCHANGE PARTITION | EXCHANGE PARTITION | EXCHANGE PARTITION | EXCHANGE PARTITION<br><br>EXCHANGE SUBPARTITION | EXCHANGE PARTITION<br><br>EXCHANGE SUBPARTITION |
| Merging Partitions | MERGE PARTITIONS | n/a | MERGE PARTITIONS | MERGE PARTITIONS | MERGE PARTITIONS<br><br>MERGE SUBPARTITIONS |
| Modifying Default Attributes | MODIFY DEFAULT ATTRIBUTES | MODIFY DEFAULT ATTRIBUTES | MODIFY DEFAULT ATTRIBUTES | MODIFY DEFAULT ATTRIBUTES<br><br>MODIFY DEFAULT ATTRIBUTES FOR PARTITION | MODIFY DEFAULT ATTRIBUTES<br><br>MODIFY DEFAULT ATTRIBUTES FOR PARTITION |
| Modifying Real Attributes of Partitions | MODIFY PARTITION | MODIFY PARTITION | MODIFY PARTITION | MODIFY PARTITION<br><br>MODIFY SUBPARTITION | MODIFY PARTITION<br><br>MODIFY SUBPARTITION |
| Modifying List Partitions: Adding Values | n/a | n/a | MODIFY PARTITION...ADD VALUES | n/a | MODIFY SUBPARTITION ... ADD VALUES |
| Modifying List Partitions: Dropping Values | n/a | n/a | MODIFY PARTITION...DROP VALUES | n/a | MODIFY SUBPARTITION ... DROP VALUES |
| Modifying a Subpartition Template | n/a | n/a | n/a | SET SUBPARTITION TEMPLATE | SET SUBPARTITION TEMPLATE |
| Moving Partitions | MOVE PARTITION | MOVE PARTITION | MOVE PARTITION | MOVE SUBPARTITION | MOVE SUBPARTITION |

# Range Partitioning

```
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');
```

# Range Partitioning (Multi-Column)




```
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);
```

# Hash Partitioning (Multi-Column)

```
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);
```

# Composite Partitioning

```
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')));
```

# List Partitioning (Allowed since 9i)




```
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.

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.

# Interval Partitioning – 11g

```
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

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

| SEGMENT_NAME | PARTITION_NAME |
|---|---|
| DEPT_NEW | D1 |
| DEPT_NEW | D2 |
| DEPT_NEW | D3 |

# Interval Partitioning – 11g

```
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.

# Interval Partitioning – 11g

```
insert into dept_new2 values(35,null);
insert into dept_new2 values(45,null);
insert into dept_new2 values(55,null);

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_P61
DEPT_NEW2            SYS_P62
DEPT_NEW2            SYS_P63
```

# Interval Partitioning – 11g

```
select  table_name, partition_count
from    dba_part_tables
where table_name in ('DEPT_NEW','DEPT_NEW2');
```

| TABLE_NAME | PARTITION_COUNT |
|---|---|
| DEPT_NEW2 | 1048575 |
| DEPT_NEW | 3 |

# 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
- ALTER TABLE... NOCOMPRESS
- Table compression now supported for OLTP

# Partition Compression

```
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.*

# Partition Compression

```
insert into dept_new3 values(10,null);
1 row created.
insert into dept_new3 values(20,null);
1 row created.
insert into dept_new3 values(30,null);
1 row created.
insert into dept_new3 values(60,null);
1 row created.
insert into dept_new3 values(90,null);
1 row created.
```

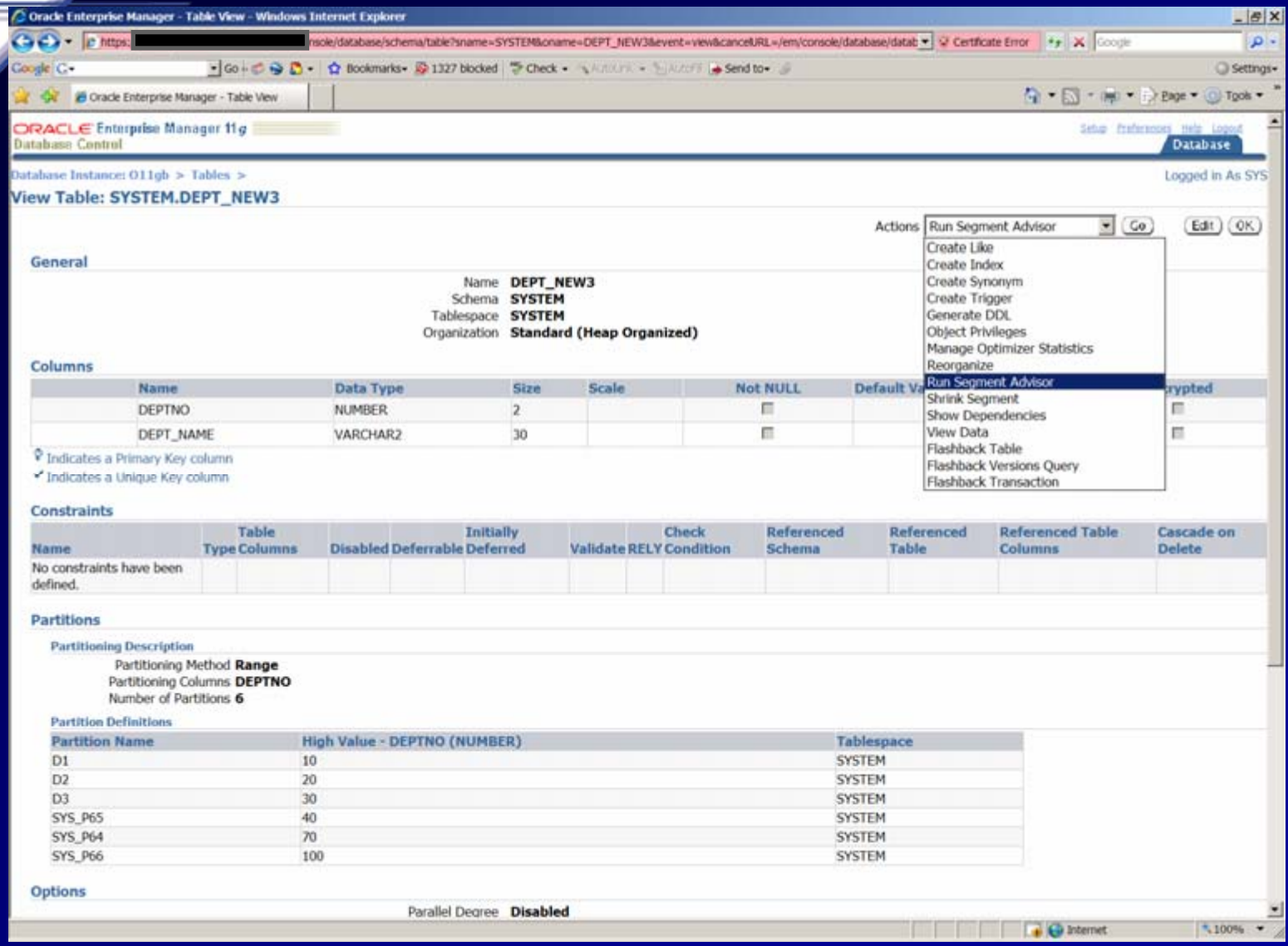
# Partition Compression

```
select  table_name, partition_name, compression
from    dba_tab_partitions
where table_name = 'DEPT_NEW3';
```

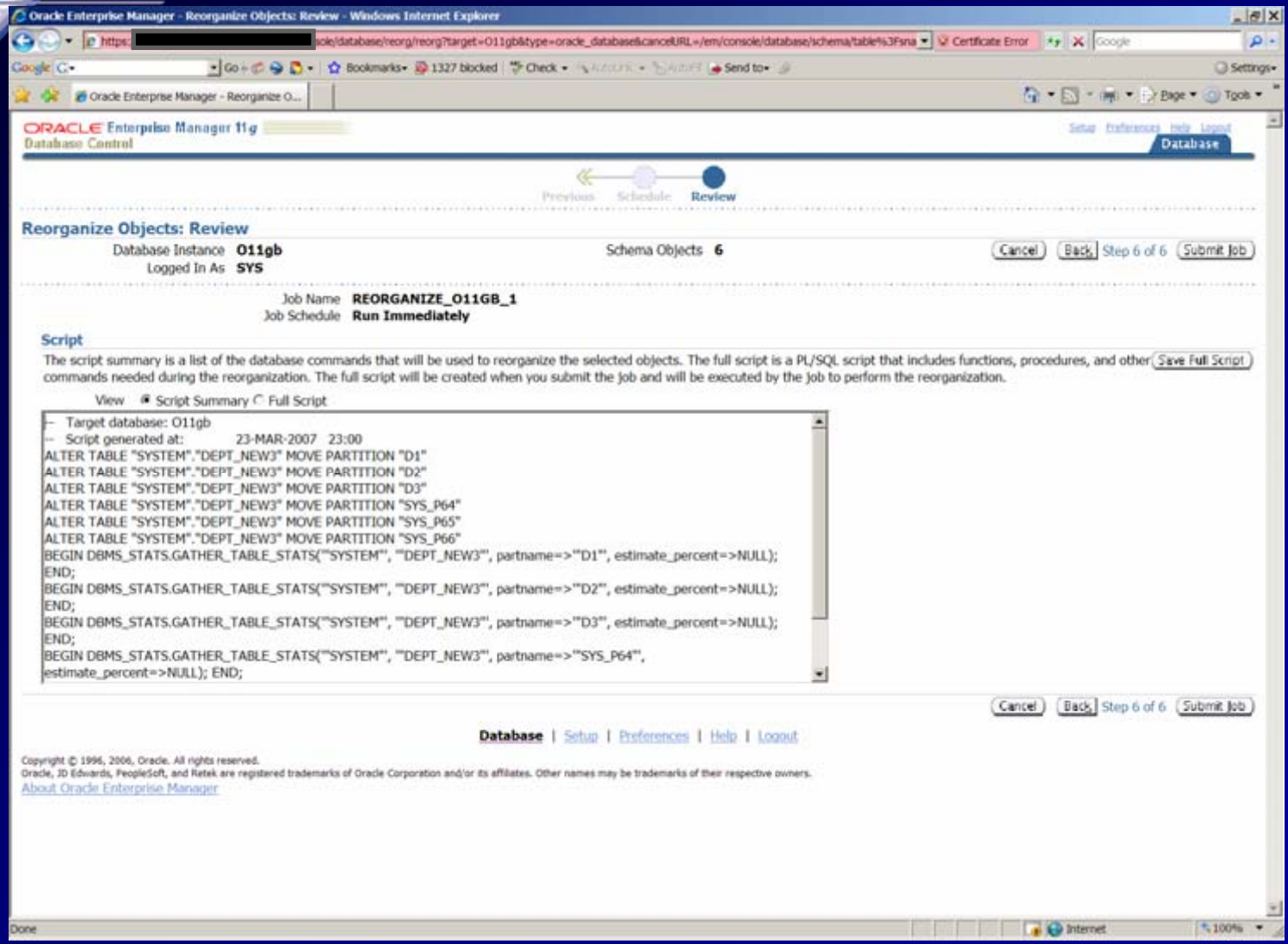
| TABLE_NAME | PARTITION_NAME | COMPRESS |
|---|---|---|
| DEPT_NEW3 | D1 | ENABLED |
| DEPT_NEW3 | D2 | DISABLED |
| DEPT_NEW3 | D3 | ENABLED |
| DEPT_NEW3 | SYS_P64 | ENABLED |
| DEPT_NEW3 | SYS_P65 | ENABLED |
| DEPT_NEW3 | SYS_P66 | ENABLED |

*6 rows selected.*

# Object Maintenance



# Object Maintenance – Reorganize



# Nice DBA Tools / Improvements


**Read Only Tables**
**Automated Maintenance**
**Additional Features**

# Read Only Tables

- You could already have a Tablespace in Read Only Mode in prior versions of Oracle.
- NOW You can have a TABLE in Read Only Mode in Oracle 11g (*alter table x read only*).
- COMPATIBLE must be set to 11.0 or higher
- Once put in READ ONLY mode:
  - No DML is allowed
  - No DDL other than DROP TABLE and an ALTER TABLE to READ WRITE mode is allowed.
  - SELECT FOR UPDATE is NOT allowed

# Read Only Tables

```
alter table dept_new read only;
Table altered.

insert into dept_new values (10, 'ADMIN');
            *
ERROR at line 1:
ORA-12081: update operation not allowed on table "SYSTEM"."DEPT_NEW"


alter table dept_new read write;
Table altered.

insert into dept_new values (10, 'ADMIN');
1 row created.
```

# Automated Maintenance Tasks

- Automatic Optimizer Statistics Collection
- Automatic Segment Advisor
- Automatic SQL Tuning Advisor (DBMS_SQLTUNE)
- Disable / Enable Automated Tasks:

DBMS_AUTO_TASK_ADMIN.DISABLE (ENABLE)

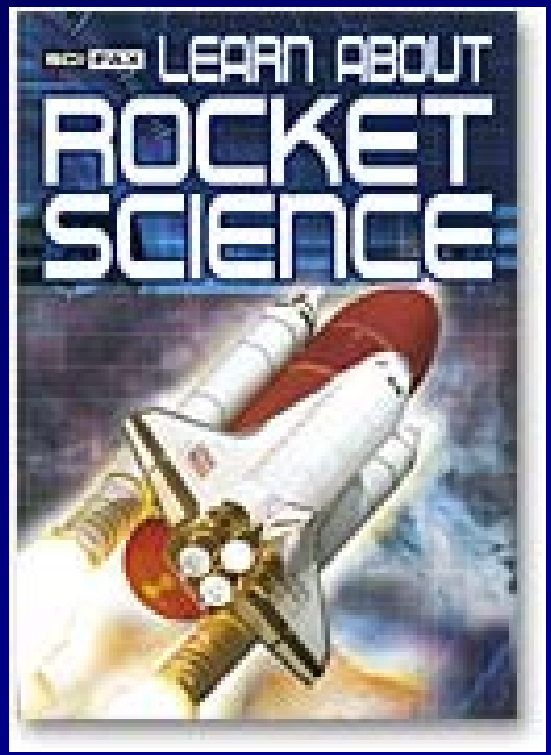
- Setting up Maintenance Windows

DBMS_SCHEDULER.CREATE_WINDOW

# Additional Enhancements

- Ability to online redefine tables that have materialized view logs:
  - Tables with materialized view logs can now be redefined online.
  - Materialized view logs are now one of the dependent objects that can be copied to the interim table with the DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS package procedure.
- DBMS_STATS performance has been improved.

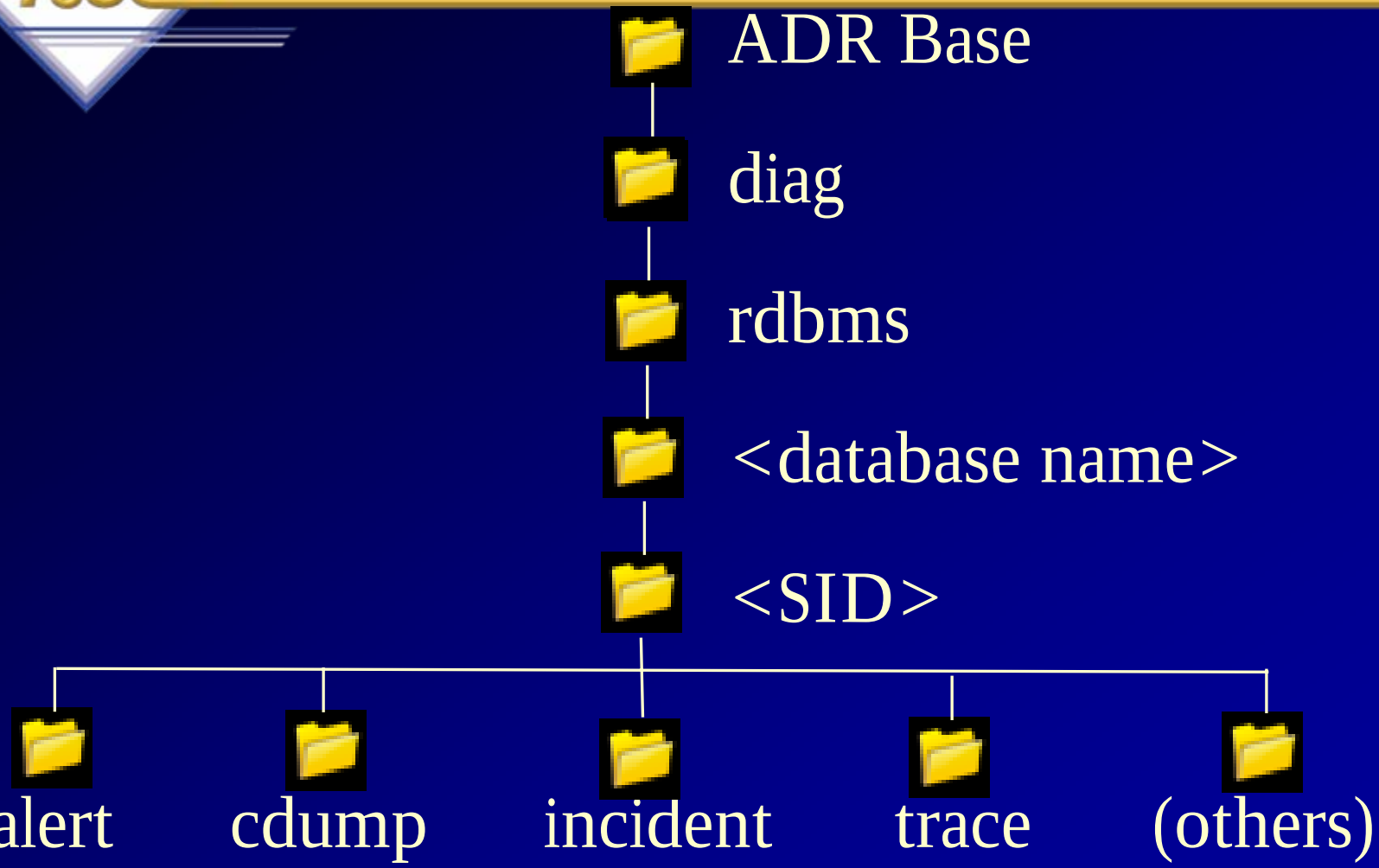
# Automatic Diagnostic Repository (ADR)



# Automatic Diagnostic Repository (ADR)

- Oracle 11g includes a Fault Diagnosability Infrastructure to prevent, detect, diagnose, resolve issues related to bugs, corruption, etc.
- When a critical error occurs it is assigned an incident number and all diagnostic data tagged with this in ADR.
- ADR is a file based repository outside of the database
- ADR helps detect problems proactively
- ADR helps limit the damage of interruptions
- ADR helps reduce problem diagnostic time
- ADR simplifies Oracle Support / Customer interaction
- The ADR also contains Health Reports, Trace Files, Dump Files, SQL Test Cases and Data Repair Records

# ADR Directory Structure for a Database Instance




**Alert Log: /u01/app/oracle/diag/rdbms/o11gb/O11gb/trace**
**ORACLE_HOME: /u01/app/oracle/product/11.1.0/db_1**

# ADR – V$ Diagnostic Info



select name, value
from v$diag_info;

```
NAME                           VALUE
------------------------------ ----------------------------------------
Diag Enabled                   TRUE
ADR Base                       /u01/app/oracle
ADR Home                       /u01/app/oracle/diag/rdbms/o11gb/O11gb
Diag Trace                     /u01/app/oracle/diag/rdbms/o11gb/O11gb/trace
Diag Alert                     /u01/app/oracle/diag/rdbms/o11gb/O11gb/alert
Diag Incident                  /u01/app/oracle/diag/rdbms/o11gb/O11gb/incident
Diag Cdump                     /u01/app/oracle/diag/rdbms/o11gb/O11gb/cdump
Health Monitor                 /u01/app/oracle/diag/rdbms/o11gb/O11gb/hm
Default Trace File             /u01/app/oracle/diag/rdbms/o11gb/O11gb/trace/O11gb_ora_16676.trc
Active Problem Count           0
Active Incident Count          0

11 rows selected.
```

# Optimizer Statistics
# &
# Other Optimizer Advances

Special Thanks: Maria Colgan & 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 sampling.
  - Accuracy is comparable to COMPUTE

# Incremental Statistics Maintenance - Stats by Partition vs. table



- 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 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!

# Manage New Statistics
## 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
--------------------
FALSE
```

# Manage New Statistics
## Gather Stats but make them PENDING


```
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

execute dbms_stats.gather_table_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_NAME ANALYZE_T   NUM_ROWS     BLOCKS AVG_ROW_LEN
---------- --------- ---------- ---------- -----------
CUST
```

# Manage New Statistics
# PUBLISH Stats after Testing Complete

alter session set optimizer_private_statistics = true;

*(Then run your query – If ready/better – publish the new stats)*

exec dbms_stats.publish_private_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_NAME  ANALYZE_T    NUM_ROWS     BLOCKS AVG_ROW_LEN
----------  ---------  ----------  ----------  -----------
CUST        13-APR-07       55500        1451          180
```

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

# 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.
- 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';
```

| | | |
|---|---|---|
| CORVETTE | 40,000 | RED |
| CORVETTE | 60,000 | BLACK |
| CORVETTE | 50,000 | SILVER |

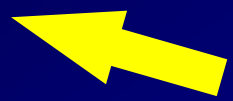
- Three records selected.
- Single column statistics are accurate

| Make | Price | Color |
|---|---|---|
| CORVETTE | 40,000 | RED |
| CORVETTE | 60,000 | BLACK |
| CORVETTE | 50,000 | SILVER |
| CADILLAC | 90,000 | RED |
| JEEP | 35,000 | BLACK |
| JEEP | 45,000 | SLIVER |

# Example, cont.

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

| | | |
|---|---|---|
| CORVETTE | 40,000 | RED |



| Make | Price | Color |
|---|---|---|
| CORVETTE | 40,000 | RED |
| CORVETTE | 60,000 | BLACK |
| CORVETTE | 50,000 | SILVER |
| CADILLAC | 90,000 | RED |
| JEEP | 35,000 | BLACK |
| JEEP | 45,000 | SLIVER |

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

# Example, cont.

```
SELECT make, price, color
FROM   cars_dot_com
WHERE  make = 'CORVETTE'
AND    PRICE > 35000;
```

| | | |
|---|---|---|
| CORVETTE | 40,000 | RED |
| CORVETTE | 60,000 | BLACK |
| CORVETTE | 50,000 | SLIVER |

- Three records selected.
  - Correlated columns
  - Additional predicate has no effect
  - **Single column statistics are NOT sufficient**

| Make | Price | Color |
|---|---|---|
| CORVETTE | 40,000 | RED |
| CORVETTE | 60,000 | BLACK |
| CORVETTE | 50,000 | SILVER |
| CADILLAC | 90,000 | RED |
| JEEP | 35,000 | BLACK |
| JEEP | 45,000 | SLIVER |

# Manage New Statistics
# 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

# Manage New Statistics
## After normal Statistics Creation


```
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.
```

# Manage New Statistics
## Create EXTENDED Statistic Group

- Now lets create the extended statistics group & 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.
```

# Manage New Statistics
## Now there are Extended Statistics


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                     146 NONE
CUST_CITY_ID                            620 HEIGHT BALANCED
CUST_CITY                               621 NONE
CUST_LAST_NAME                          755 NONE
CUST_FIRST_NAME                         987 NONE
CUST_ID                               55605 NONE
...
SYS_STUJGVLRVH5USVDU$XNV4_IR#4          146 NONE

24 rows selected.
```

# Manage New Statistics
# 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';

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.
```

# Adaptive Cursor Sharing

- The optimizer peeks at user-defined bind values during plan selection on the hard parse.
- Initial value of the binds determines the plan for all future binds (hopefully the first peek covers most queries)
- Same execution plan shared regardless of future bind values
- One plan is not always appropriate for all bind values for a given SQL statement
  - Where job= 'PRESIDENT' (use an index – only one row)
  - Where job = 'OPERATOR' (don't use an index – 90% of the table)
- If Oracle "peeks" and sees the President, it will use the index. Future queries also use the index without peeking after that (bad for the OPERATOR query).

# Consider a Telephone Company…


**SELECT Ename, Empno, Job**
**FROM Emp**
**WHERE Job = :B1**
**Value of B1 = 'OPERATOR';**

| Ename | Empno | Job |
|---|---|---|
| SMITH | 6973 | OPERATOR |
| ALLEN | 7499 | OPERATOR |
| WARD | 7521 | OPERATOR |
| KING | 8739 | PRESIDENT |
| SCOTT | 7788 | OPERATOR |
| CLARK | 7782 | OPERATOR |

| Ename | Empno | Job |
|---|---|---|
| SMITH | 6973 | OPERATOR |
| ALLEN | 7499 | OPERATOR |
| WARD | 7521 | OPERATOR |
| SCOTT | 7788 | OPERATOR |
| CLARK | 7782 | OPERATOR |

- If 'OPERATOR' is the bind value at hard parse, most records will be selected. Execution plan will be a full table scan
- If 'PRESIDENT' is the bind value at hard parse, few records will be selected. Execution plan will be an index search

# Adaptive Cursor Sharing

**Solution:**

- In 11g, Oracle uses bind-aware cursor matching.
- Share the plan when binds values are “equivalent”
  - Plans are marked with selectivity range
  - If current bind values fall within range they use the same plan
- Create a new plan if binds are not equivalent
  - Generating a new plan with a different selectivity range

# Bind Peeking
# Cursor Sharing (cs) Statistics

select sql_id, peeked, executions, rows_processed, cpu_time
from v$sql_cs_statistics; (using the peeked value on the 2nd+ execution)

| SQL_ID | P | EXECUTIONS | ROWS_PROCESSED | CPU_TIME |
|---|---|---|---|---|
| 5wfj3qs71nd7m | Y | 3 | 1 | 0 |
| 2rad83pp613m1 | Y | 3 | 3 | 0 |
| dr78c03uv97bp | N | 1 | 3 | 0 |
| dr78c03uv97bp | N | 1 | 3 | 0 |
| dr78c03uv97bp | Y | 1 | 3 | 0 |
| 9qv6tq9ag5b80 | Y | 3 | 3 | 0 |
| a2k4qkh681fzx | Y | 3 | 2 | 0 |
| 413zr99jf9h72 | N | 1 | 1 | 0 |
| 413zr99jf9h72 | N | 1 | 1 | 0 |
| 413zr99jf9h72 | Y | 1 | 1 | 0 |
| fd69nfzww1mhm | Y | 6 | 0 | 0 |

# Bind Peeking – V$SQL

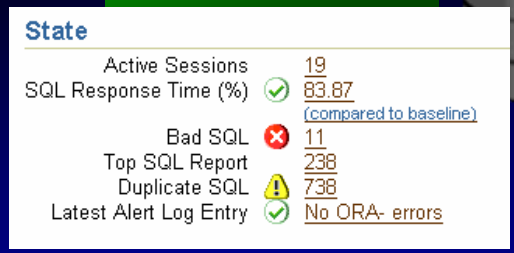
```
select sql_id, executions, is_bind_sensitive, is_bind_aware
from  v$sql;

SQL_ID        EXECUTIONS I I
------------- ---------- - -
9ugwm6xmvw06u          11 Y N
bdfrydpbzw07g          11 Y N
57pfs5p8xc07w          20 N N
...
```

- is_bind_sensitive – If 'Y', then Oracle is using multiple plans depending on bind variable.
- is_bind_aware – Oracle knows that the different data patterns may result depending on bind value. Oracle switches to a bind-aware cursor and may hard parse the statement.

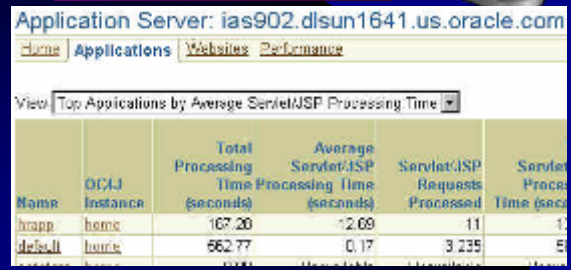
# Enterprise Manager for the Grid

**Host and Hardware**

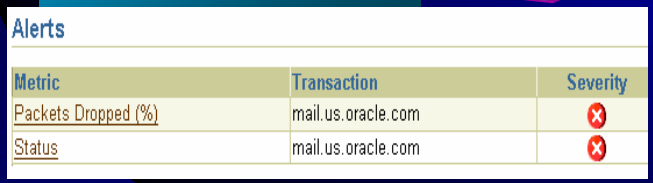

**Database**




**Oracle9iAS**




**Network and Load Balancer**



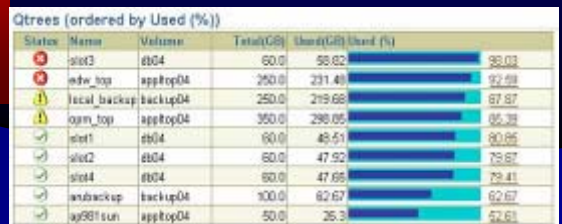

**Administration**
**Monitoring**
**Provisioning**
**Security**

Enterprise Manager

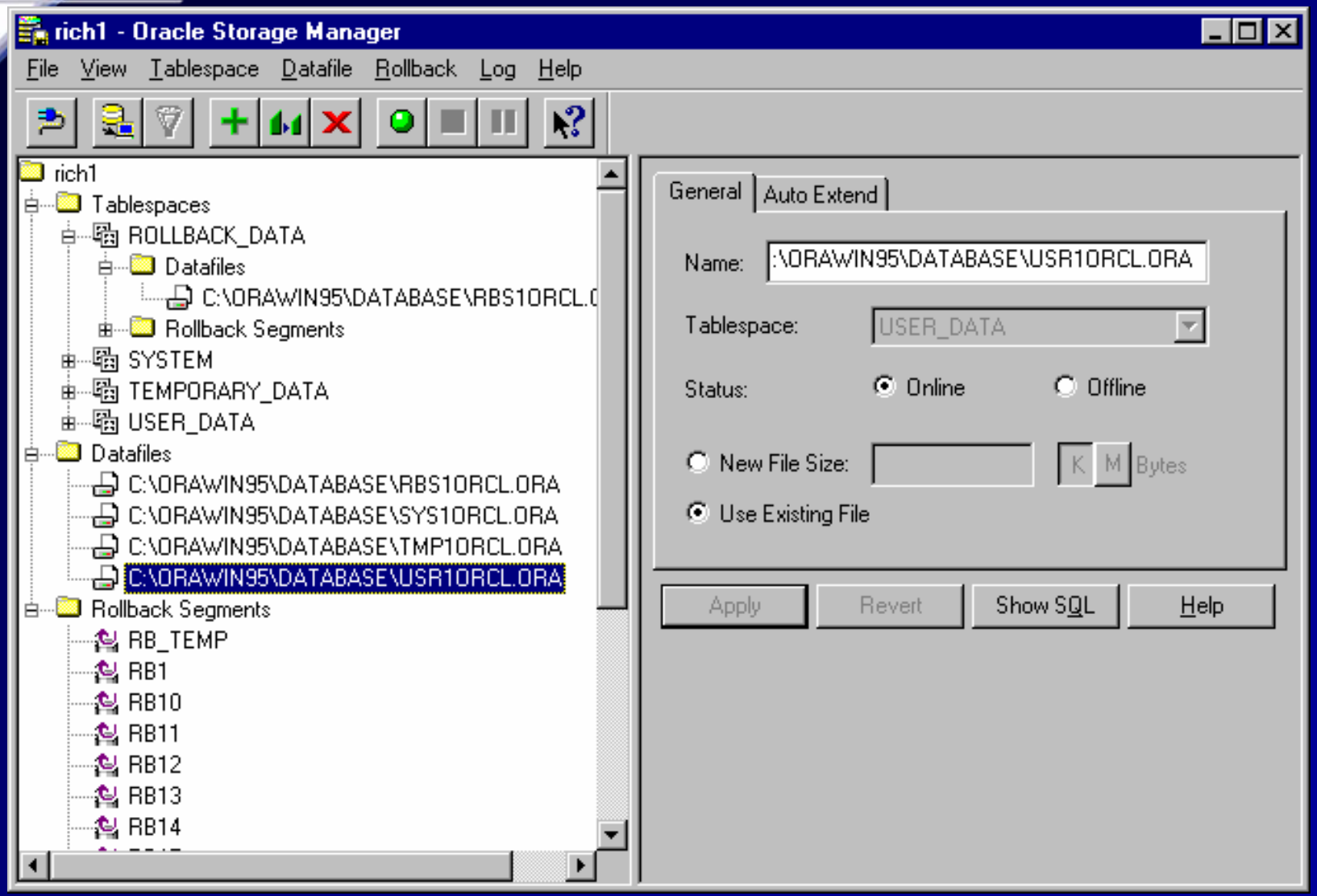
**Applications**



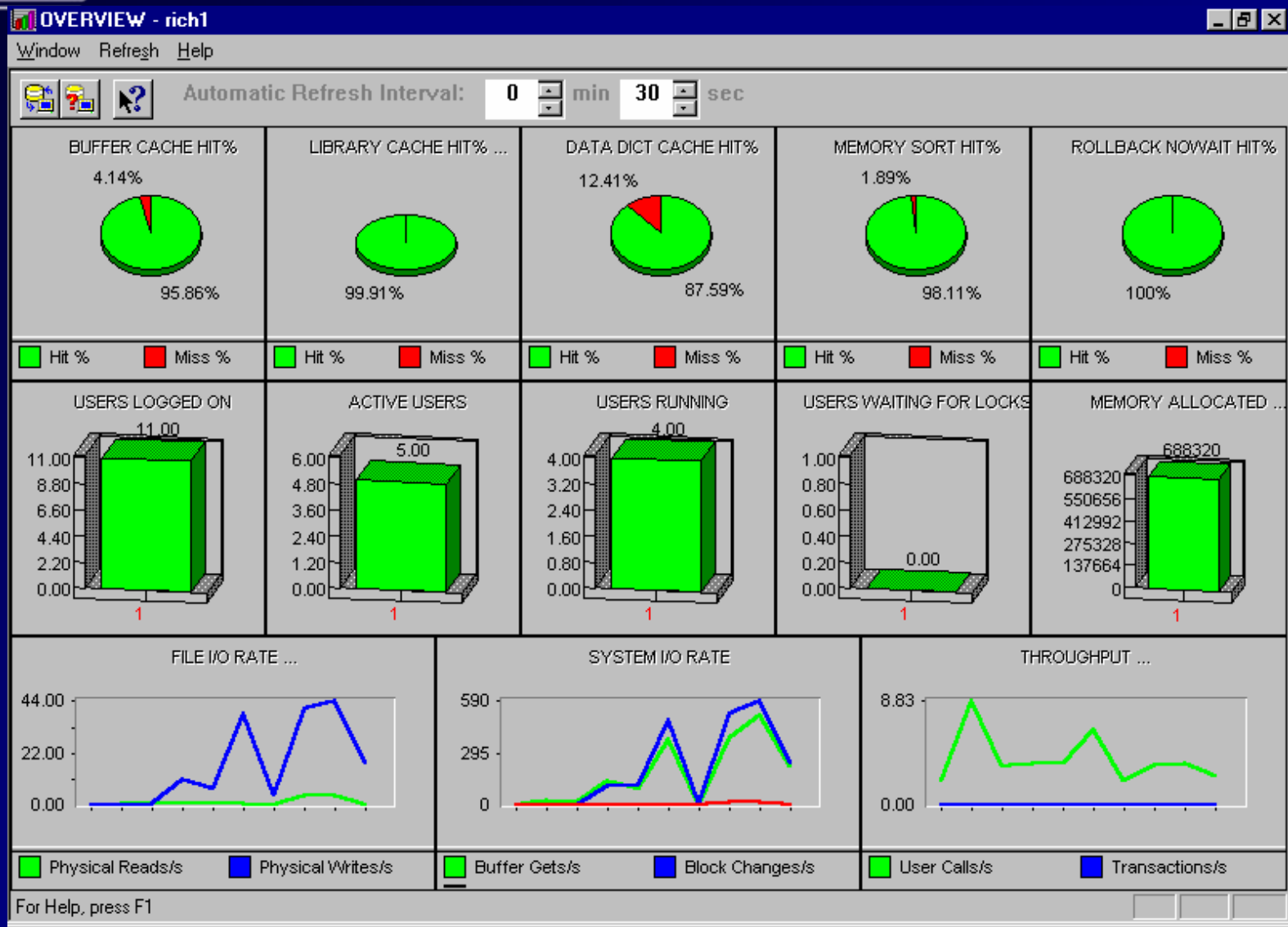
**Storage**



# Enterprise Manager: Back in Time!

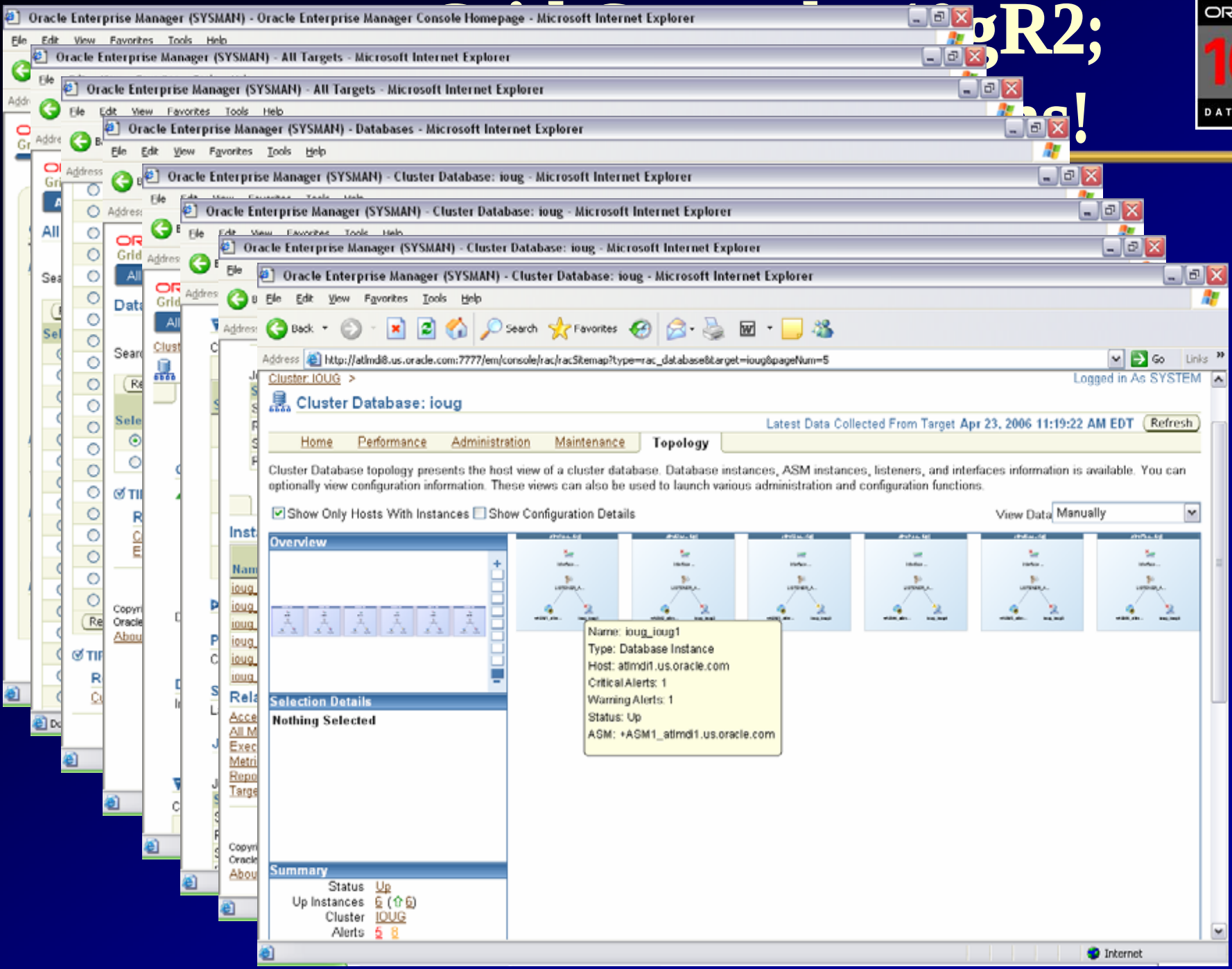


# Performance Manager : Back in Time!

# gR2;

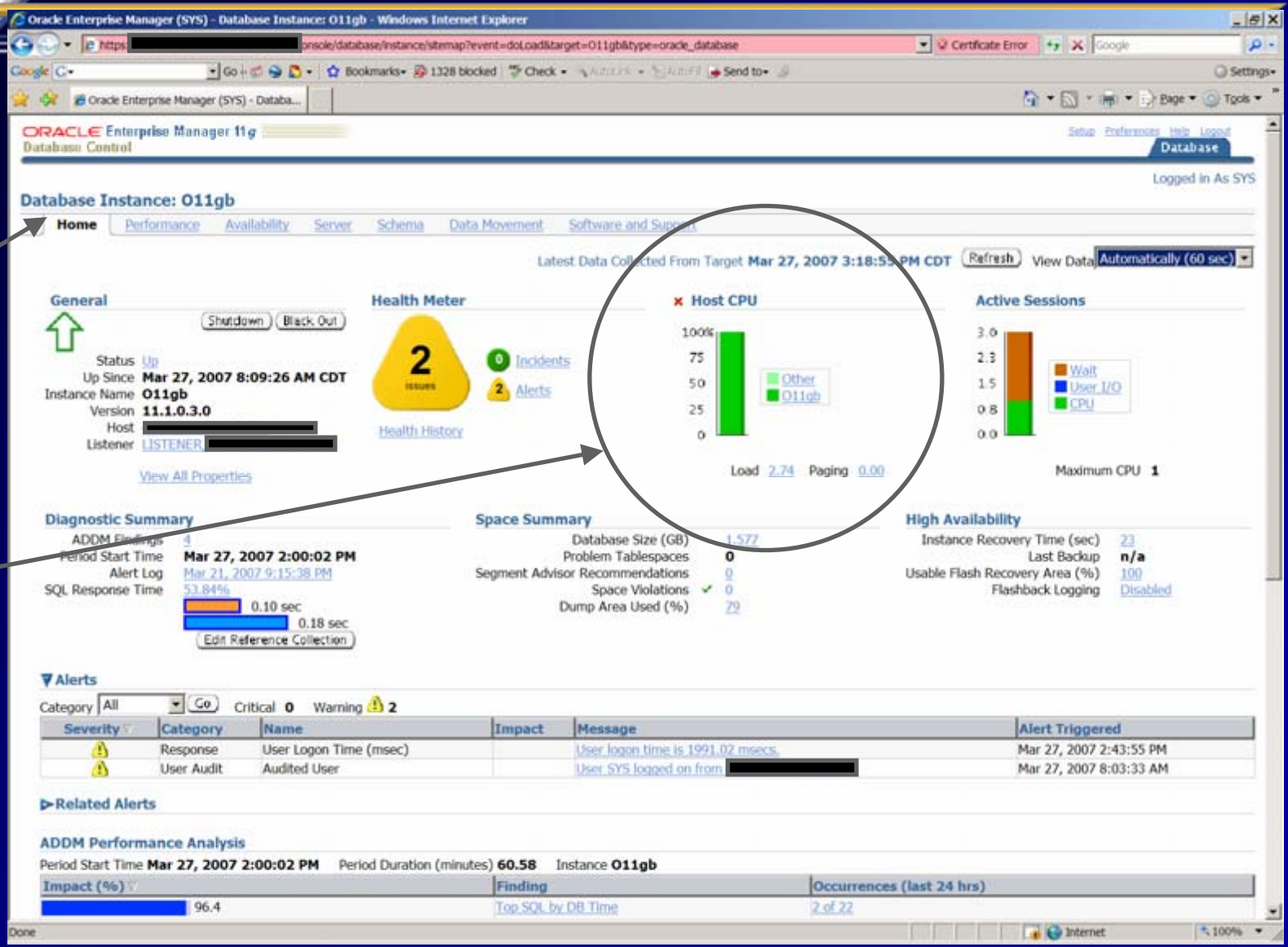
g!


Oracle Enterprise Manager (SYSMAN) - Cluster Database: ioug - Microsoft Internet Explorer
Cluster: IOUG >
Logged in As SYSTEM
Cluster Database: ioug
Latest Data Collected From Target Apr 23, 2006 11:19:22 AM EDT
Refresh
Home
Performance
Administration
Maintenance
Topology
Cluster Database topology presents the host view of a cluster database. Database instances, ASM instances, listeners, and interfaces information is available. You can optionally view configuration information. These views can also be used to launch various administration and configuration functions.
Show Only Hosts With Instances
Show Configuration Details
View Data
Manually
Overview
Selection Details
Nothing Selected
Summary
Status Up
Up Instances 6 (⇧0)
Cluster IOUG
Alerts 5 8
Name: ioug_ioug1
Type: Database Instance
Host: atlmdi1.us.oracle.com
Critical Alerts: 1
Warning Alerts: 1
Status: Up
ASM: +ASM1_atlmdi1.us.oracle.com

# Database Performance

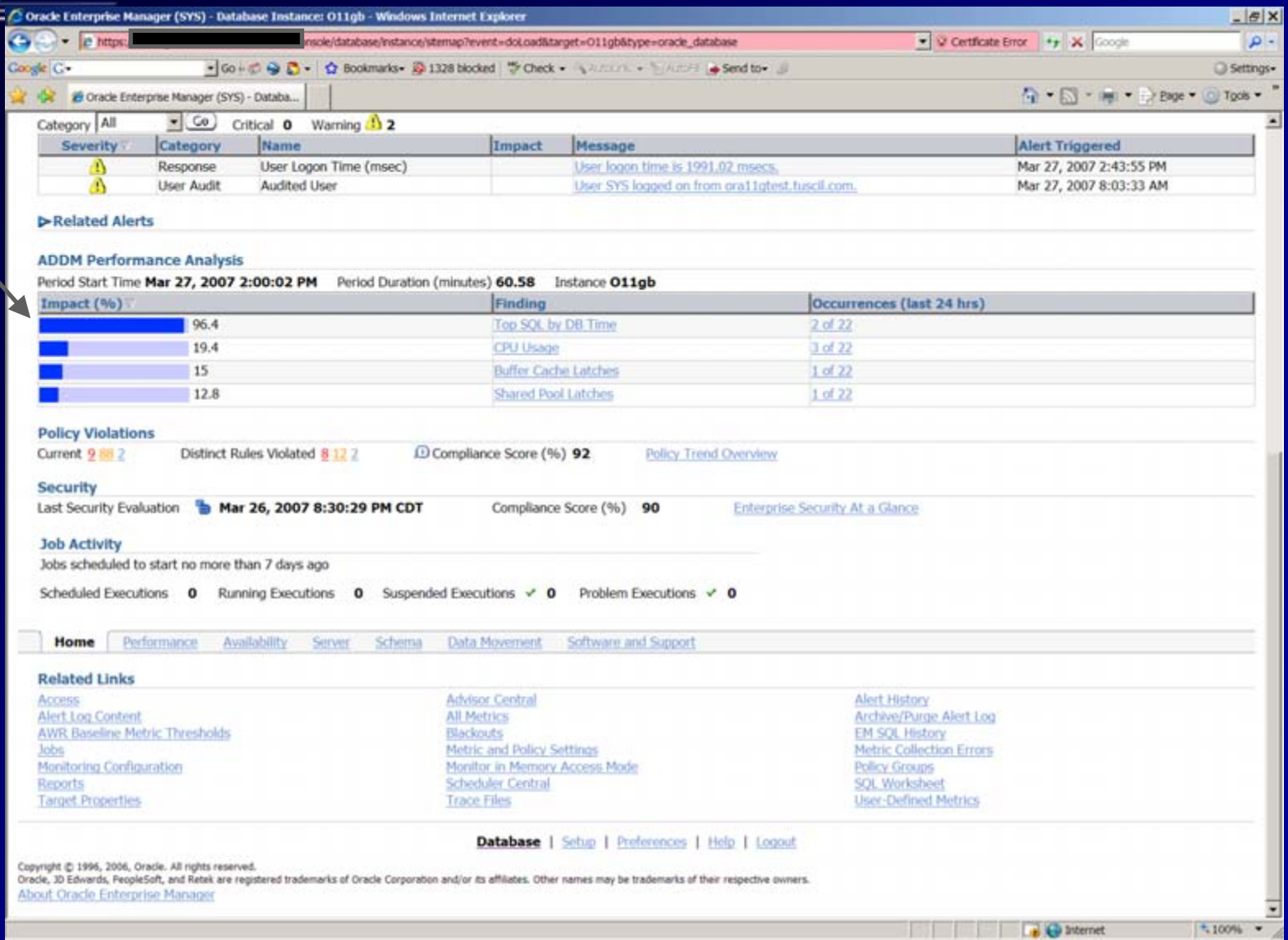
Monitor Database

We have a CPU issue!



# Database Performance

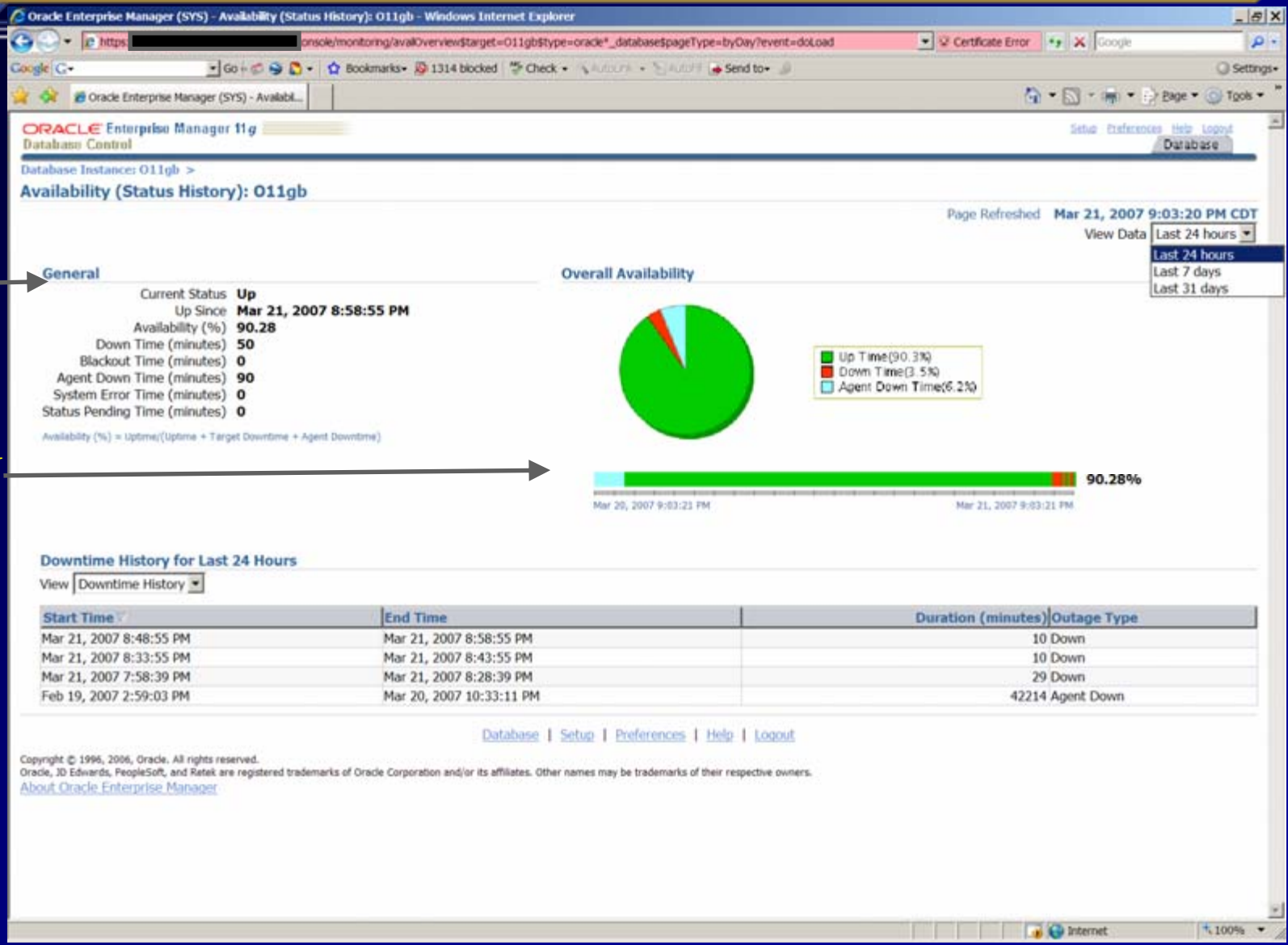
Top SQL the Issue



# Database Performance

Current Status

History of Availability

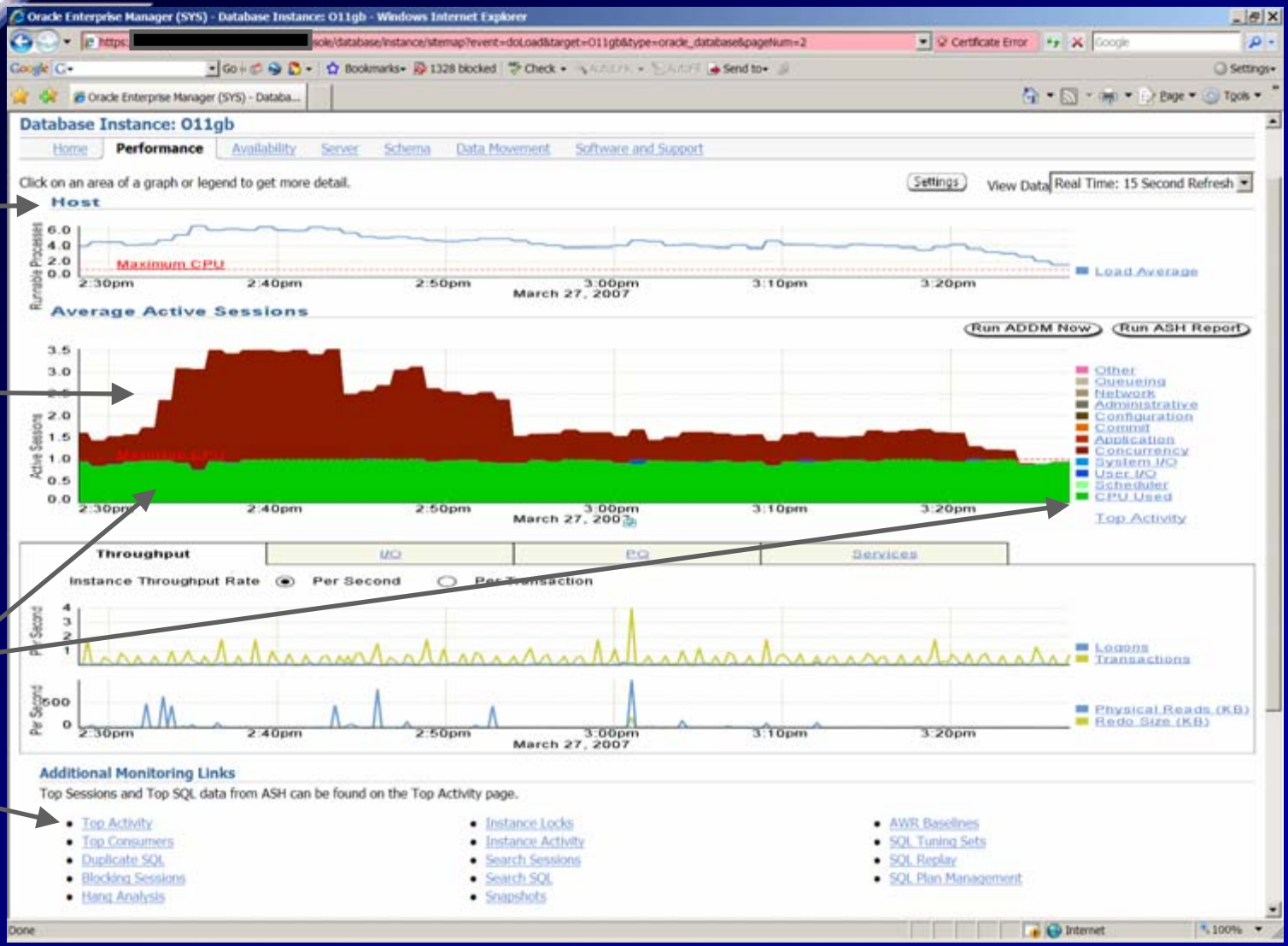


# Database Performance

Check Host

Check Database Active Sessions (CPU)

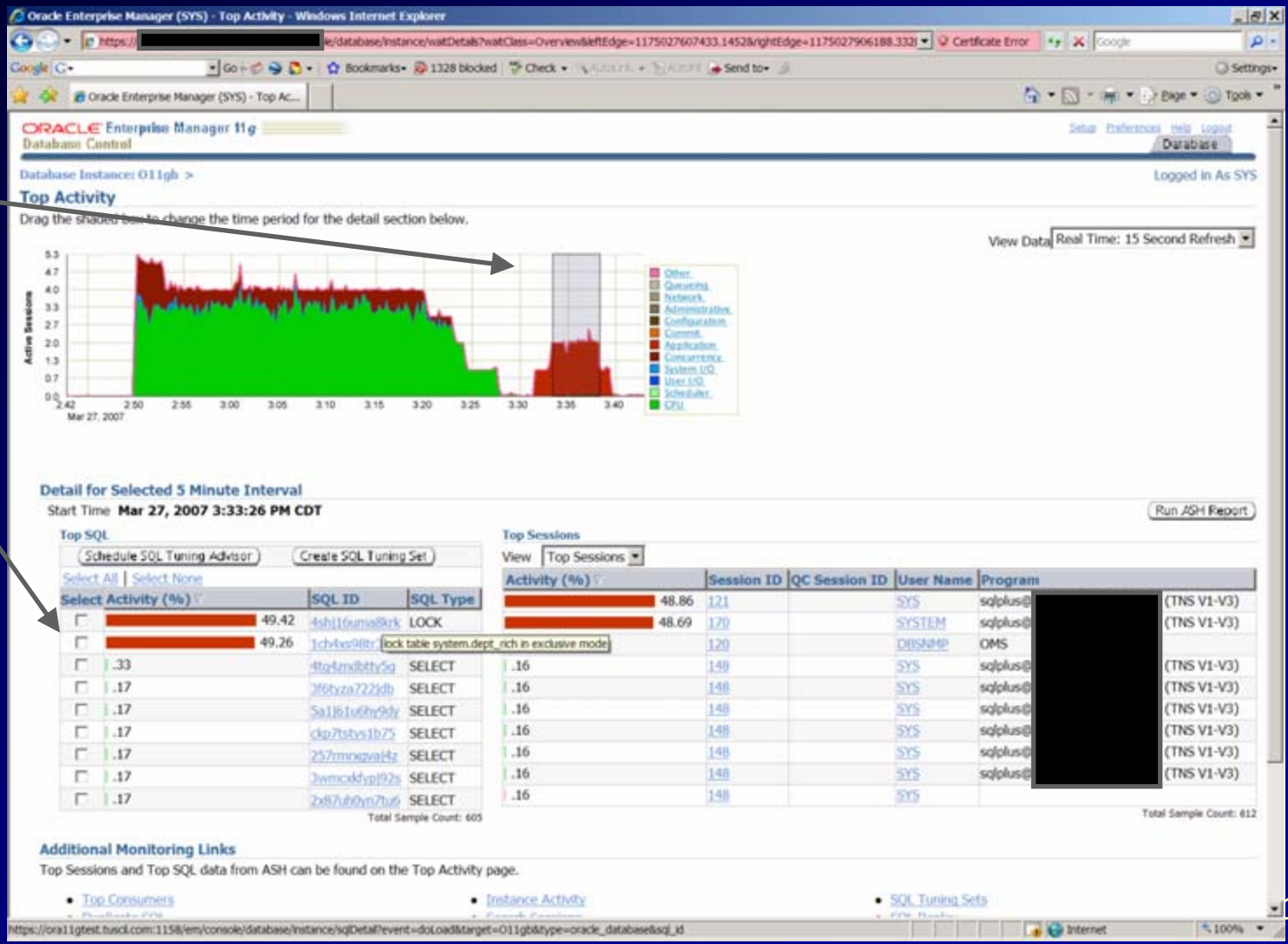
Top Activity



# Database Performance

Top Activity

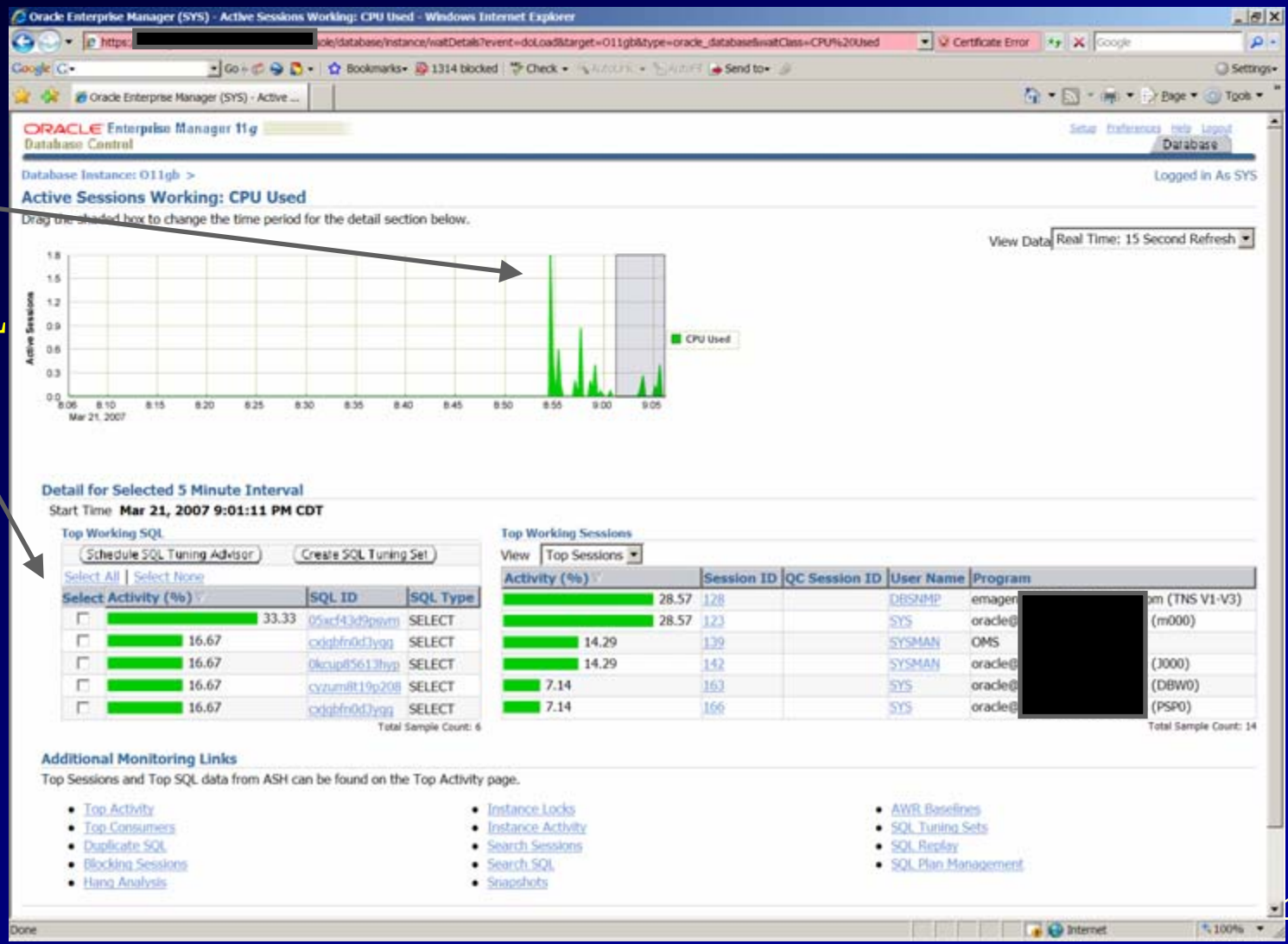
Lock Table Issues



# Database Performance

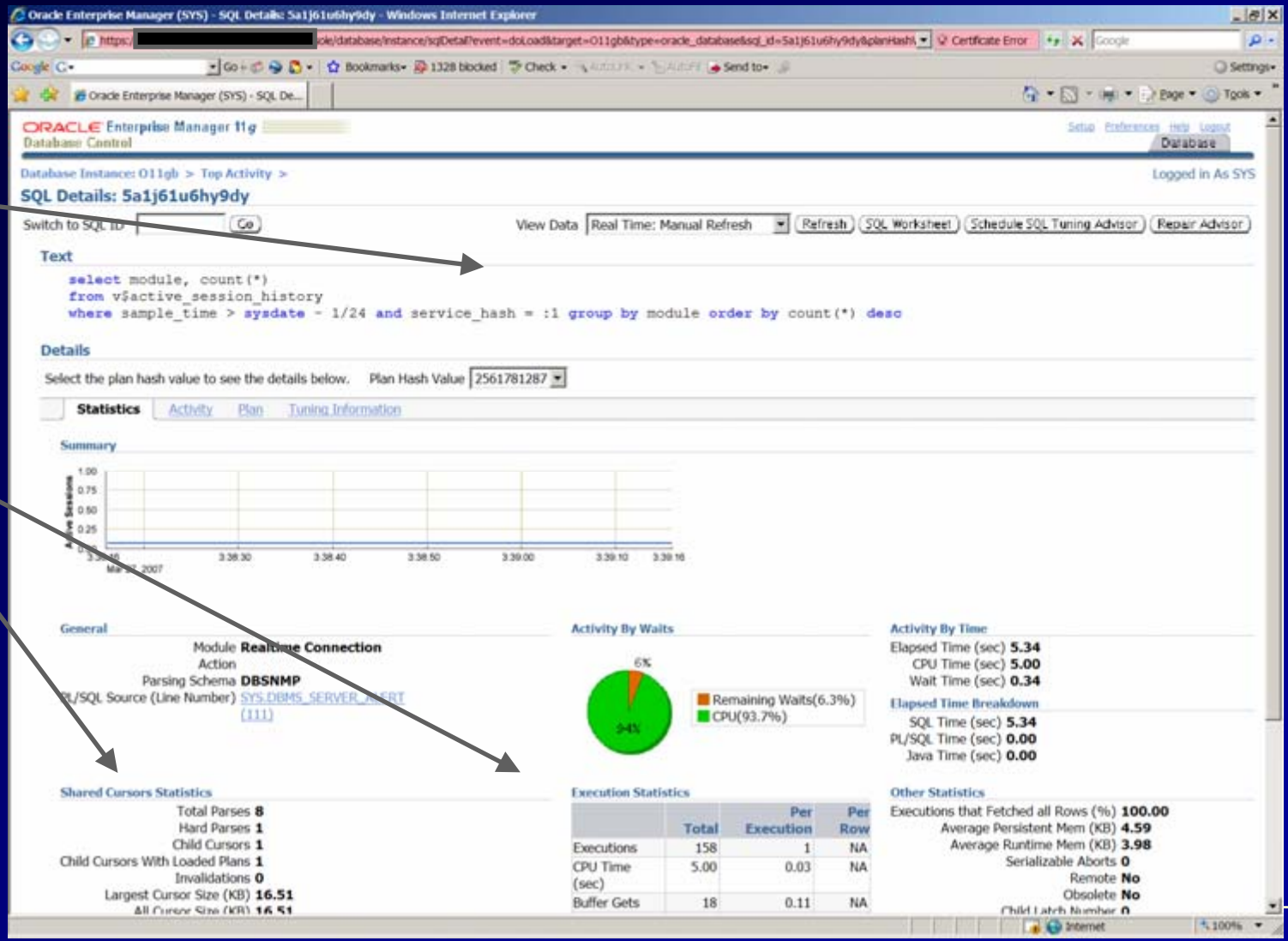
CPU

Worst SQL by CPU

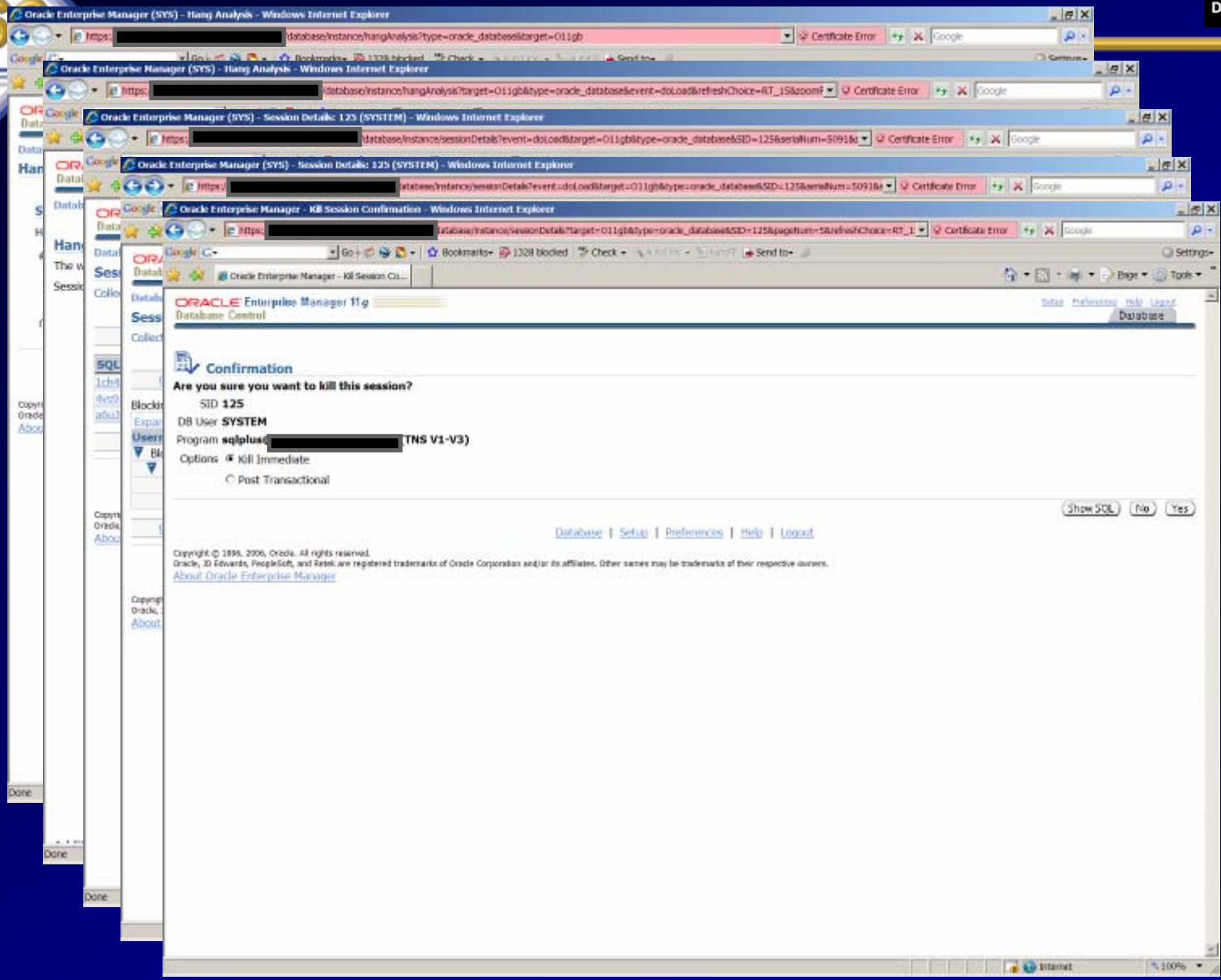


SQL

Statistics for this SQL



# Hang Analysis!



# 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 TABLLE, 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.

# 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 2006 (5E)**

*or*

- **All Email generated in the world in 2006 (6E)**

*or*

- **1 Mount Everest filled with Documents (approx.)**

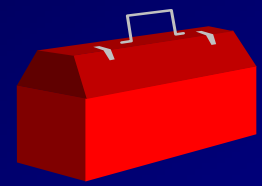
# Impact Tuning with Oracle

## 99.8% Less Data Accessed / 96.8% Time Reduction

| Option | Before | After |
|---|---|---|
| Partitions | 120 sec - 310M | 0.43 sec - 200k |
| Partitions / Tuned | 120 sec - 310M | 0.01 sec - 8k |
| Parallel Query (20 Proc.) | 230 sec | 18 sec |
| Function-Based Index | 1206 sec - 3G | 7 sec - 8k |
| Materialized View | 28 sec | 3 sec |
| Cursor_Sharing | 240 sec | 0.01 sec |
| Truncate | 510 sec / 8G | 0.40 sec / 32k |
| Driving Table | 900 sec | 1 second |
| SGA Sizing | 30 sec | 0.01 sec |
| 750,000 Query Mix | 5.1 T / 540 hours | 9 G / 23 hours |

**NOW Oracle will do all of this for you!!!**

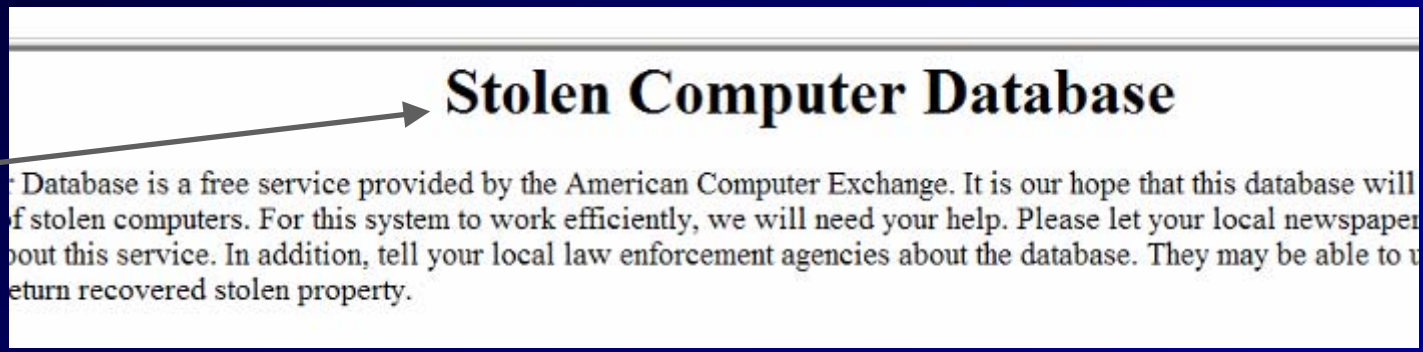
# V$ Views over the years



| Version | V$ Views | X$ Tables |
|---|---|---|
| 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 |
| **11.1.0.3 (beta)** | **482 (+22%)** | **782 (+22%)** |

# Oracle 11g
# Keeps the Database Up & Secure!

DBA Problem




Celebrity Problem


*Wouldn't you rather be a DBA...*

# Overview

- Start Me Up – Using Memory Target
- The Result Cache
- Invisible Indexes & Online Index Rebuilds
- Virtual Columns and Nice Developer Tools
- ADDM Enhancements
- SQL Plan Management & SQL Plan Baselines
- SQL Access Advisor & Partition Advisor
- SQL Query Repair Advisor
- SQL Replay and DB Capture & Replay
- Interval Partitioning & Partition Compression
- DBA Tools, ADR and DBMS_STATS
- Grid Control & EM
- Security Enhancements & the Future Sizes
- Summary

# Kramer doesn't have a Backup

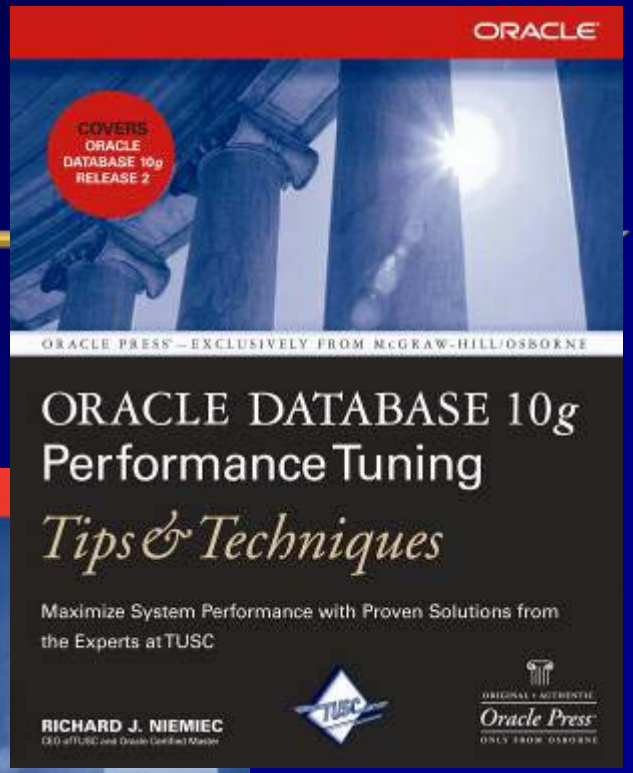
***My Junior DBA is getting the backup right now!***

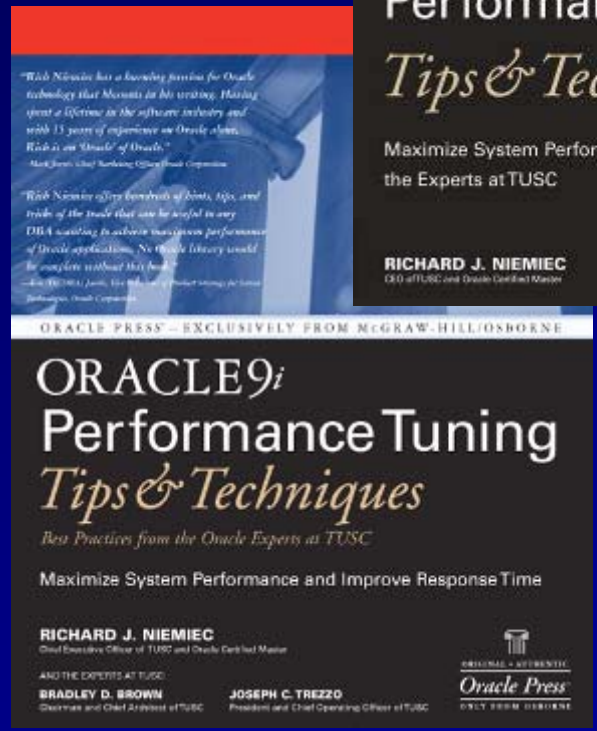
# Statspack - Still nice; Some new 10g features


*Do Developers think of this when they think of their DBA?*

# 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*

# References

- www.tusc.com
- *Oracle10g Performance Tuning Tips & Techniques; Richard J. Niemiec; Oracle Press*
- www.ioug.org, www.oracle.com & technet.oracle.com
- Thanks Dan M., Bob T., Brad, Joe, Heidi, Mike K., Debbie
- All companies and product names are trademarks or registered trademarks of the respective owners.
- Database Secure Configuration Initiative: Enhancements with Oracle Database 11g, www.oracle.com
- All Oracle11g Documentation from Oracle Beta Site
- 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 Implementations / Upgrades
  - Oracle Applications Tuning
- Managed Services
  - 24x7x365 Remote Monitoring & Management
  - Functional & Technical Support
- Training & Mentoring
- Oracle Authorized Reseller

# Copyright Information

- Neither TUSC nor the author guarantee this document to be error-free. Please provide comments/questions to rich@tusc.com.
- 

Contact Information

Rich Niemiec: rich@tusc.com

**www.tusc.com**

# www.tusc.com

ORACLE
PartnerNetwork
* PARTNER OF THE YEAR 2002
Internet Platform


*"Success usually comes to those that are too busy to be looking for it."*

*- Henry David Thoreau*