NYOUG

Custom Performance Reporting Changes in Oracle 10g

Brian Doyle
BEZ Systems
VP, Product Service
Email: bdoyle@bez.com
(617) 532-8804

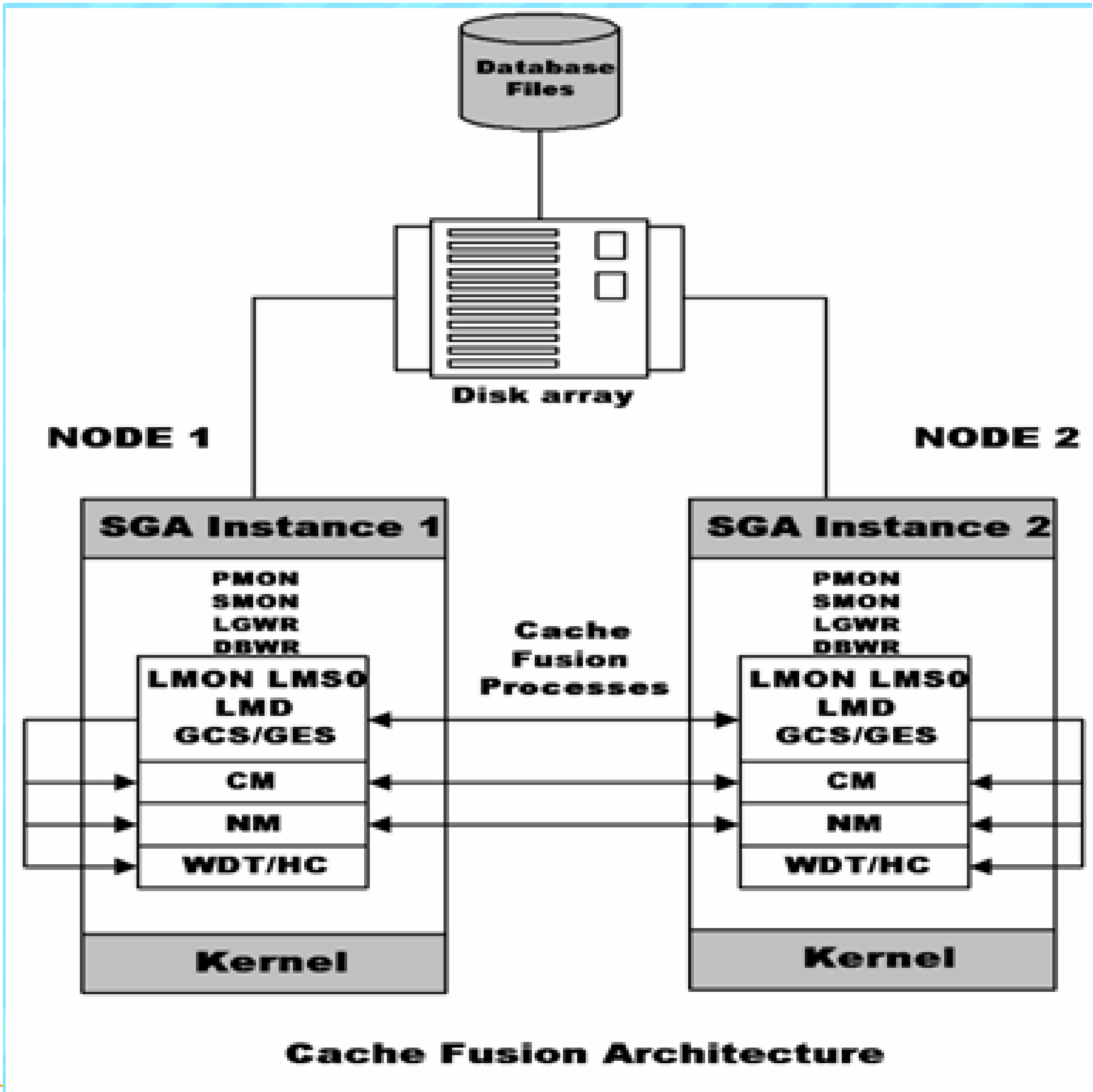
BEZ
a business lens on IT

Agenda

Topics to be discussed ….

- **RAC data capture using GV$ views**
- **Parallel Queries**
- **ASH / AWR**
- **Time Model Data**

RAC data capture using GV$ views



Cache Fusion Architecture

Oracle Dynamic Views

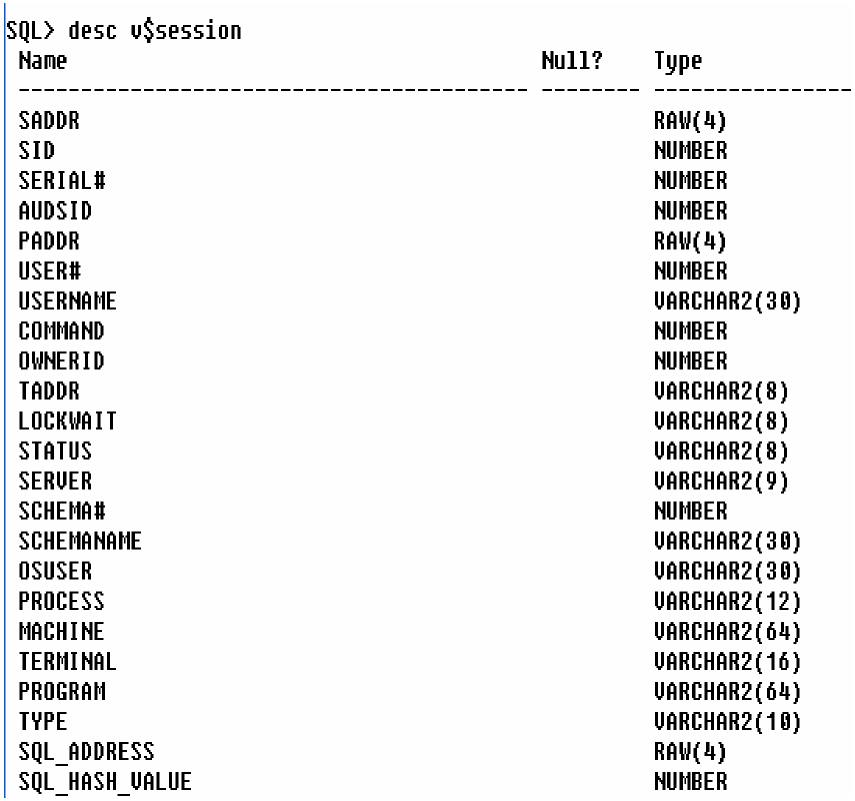
V$ views:

Provides access to the data in the internal memory cache for a single database instance.

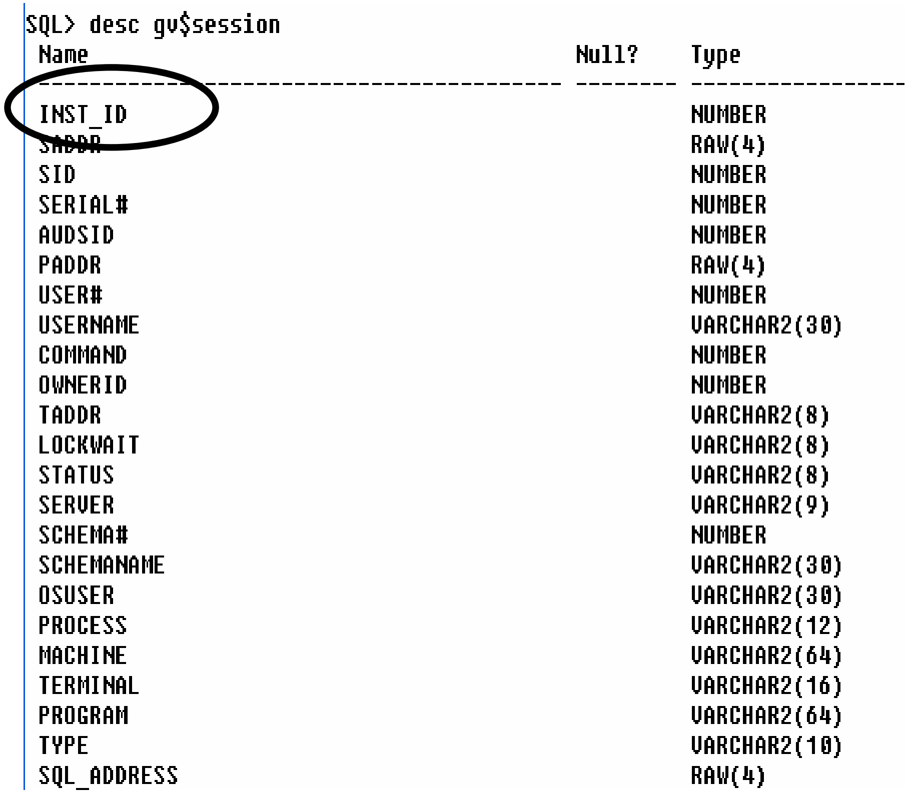
GV$ views:

Provides access to the data in the internal memory cache for an entire RAC cluster.

Example



```
SQL> desc v$session
 Name                                        Null?    Type
 ------------------------------------------- -------- ----------------
 SADDR                                                RAW(4)
 SID                                                  NUMBER
 SERIAL#                                              NUMBER
 AUDSID                                               NUMBER
 PADDR                                                RAW(4)
 USER#                                                NUMBER
 USERNAME                                             VARCHAR2(30)
 COMMAND                                              NUMBER
 OWNERID                                              NUMBER
 TADDR                                                VARCHAR2(8)
 LOCKWAIT                                             VARCHAR2(8)
 STATUS                                               VARCHAR2(8)
 SERVER                                               VARCHAR2(9)
 SCHEMA#                                              NUMBER
 SCHEMANAME                                           VARCHAR2(30)
 OSUSER                                               VARCHAR2(30)
 PROCESS                                              VARCHAR2(12)
 MACHINE                                              VARCHAR2(64)
 TERMINAL                                             VARCHAR2(16)
 PROGRAM                                              VARCHAR2(64)
 TYPE                                                 VARCHAR2(10)
 SQL_ADDRESS                                          RAW(4)
 SQL_HASH_VALUE                                       NUMBER
```



```
SQL> desc gv$session
 Name                                        Null?    Type
 ------------------------------------------- -------- -----------------
 INST_ID                                              NUMBER
 SADDR                                                RAW(4)
 SID                                                  NUMBER
 SERIAL#                                              NUMBER
 AUDSID                                               NUMBER
 PADDR                                                RAW(4)
 USER#                                                NUMBER
 USERNAME                                             VARCHAR2(30)
 COMMAND                                              NUMBER
 OWNERID                                              NUMBER
 TADDR                                                VARCHAR2(8)
 LOCKWAIT                                             VARCHAR2(8)
 STATUS                                               VARCHAR2(8)
 SERVER                                               VARCHAR2(9)
 SCHEMA#                                              NUMBER
 SCHEMANAME                                           VARCHAR2(30)
 OSUSER                                               VARCHAR2(30)
 PROCESS                                              VARCHAR2(12)
 MACHINE                                              VARCHAR2(64)
 TERMINAL                                             VARCHAR2(16)
 PROGRAM                                              VARCHAR2(64)
 TYPE                                                 VARCHAR2(10)
 SQL_ADDRESS                                          RAW(4)
```

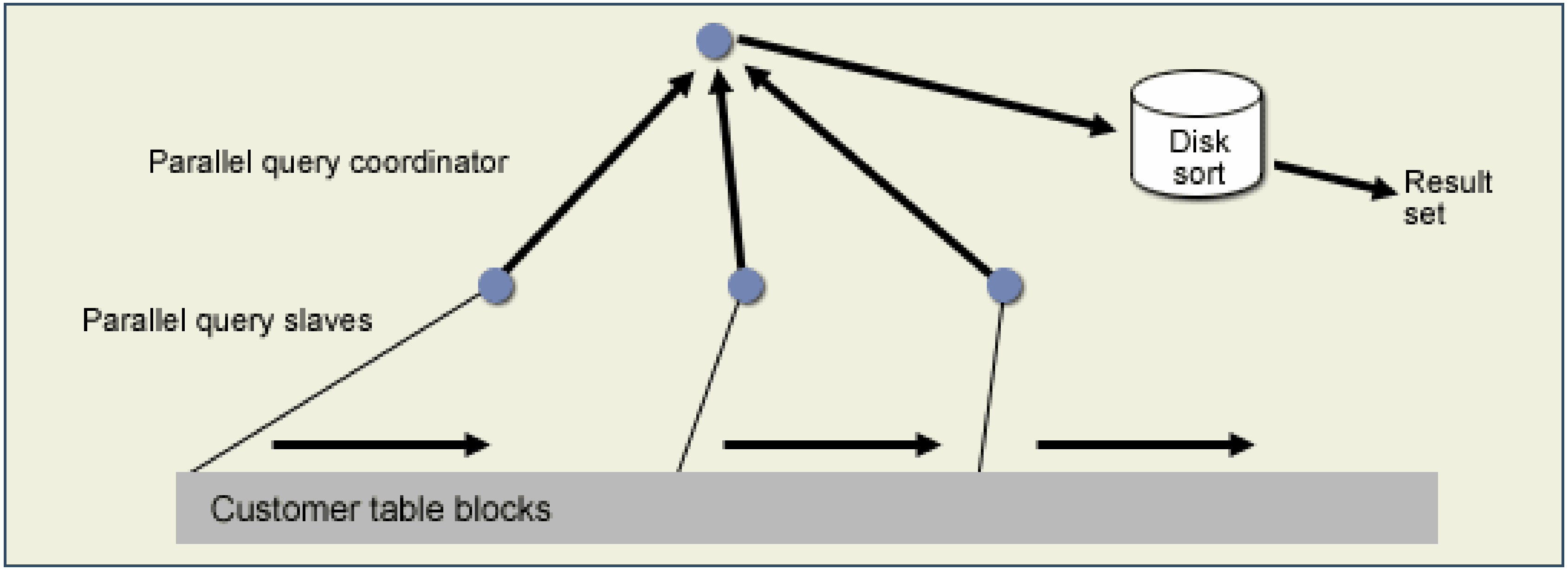
inst_id identifies the database instance that the data pertains to

Custom Reporting using GV$ Views

- GV$ views can be used even if a single instance is being reported on.
  ** inst_id will be the same for all rows

- When assembling the output results for a multi-instance RAC configuration, Oracle uses parallel query slave processes on each instance to gather the details.
  ** this can lead to problems if the OPQ settings are not adequate enough.

OPQ Architecture



Environment:

- PARALLEL_MAX_SERVER
- PARALLEL_AUTOMATIC_TUNING
- etc ...

Degree of Parallelism (DOP):

- Object Level Definition
- Statement Level (via hint)

Potential Problem with GV$ queries

ORA-12850 Could not allocate slaves on all specified instances

with cause: When executing a query on a gv$ fixed view, one or more instances failed to allocate a slave to process query

Why ??

1) **timeout occurred from master coordinator process waiting for slave process to report its results.**
2) **no parallel query slave process was available to accept the request.**

Note: The ORA-12850 is new with Oracle 10gR2. In previous releases there is no indication that the result set may be incomplete.

GV$ Summary

- **Pro's:**
  - For custom reporting, a single set of scripts can be written using the GV$ views. These can then be deployed against any Oracle environment; standalone or RAC.
  - inst_id will segment activity between instance of a multi-instance RAC cluster

- **Con's:**
  - When using GV$ queries make sure the OPQ settings are adequate to handle the parallel processing needs of the application as well as the GV$ reporting.
  - Code will have to check for the existence of the ORA-12850 error condition.

Parallel Query Operation

1. Parallel execution divides the task of executing a SQL statement into multiple small units, each of which is executed by a separate process. The user process that is going to execute a query in parallel takes on the role as parallel execution coordinator, or query coordinator.

2. The coordinator performs the parts of the plan that execute serially (such as accessing tables in serial if they are small or have no hint or degree of parallelism set). Ranging is also done serially to determine the ranges of keys to be distributed from producer slaves to consumer slaves who are sorting or otherwise must consume specific ranges of rows.

Parallel Query Coordinator

The **query coordinator** does the following:

- Parses the query and determines the degree of parallelism

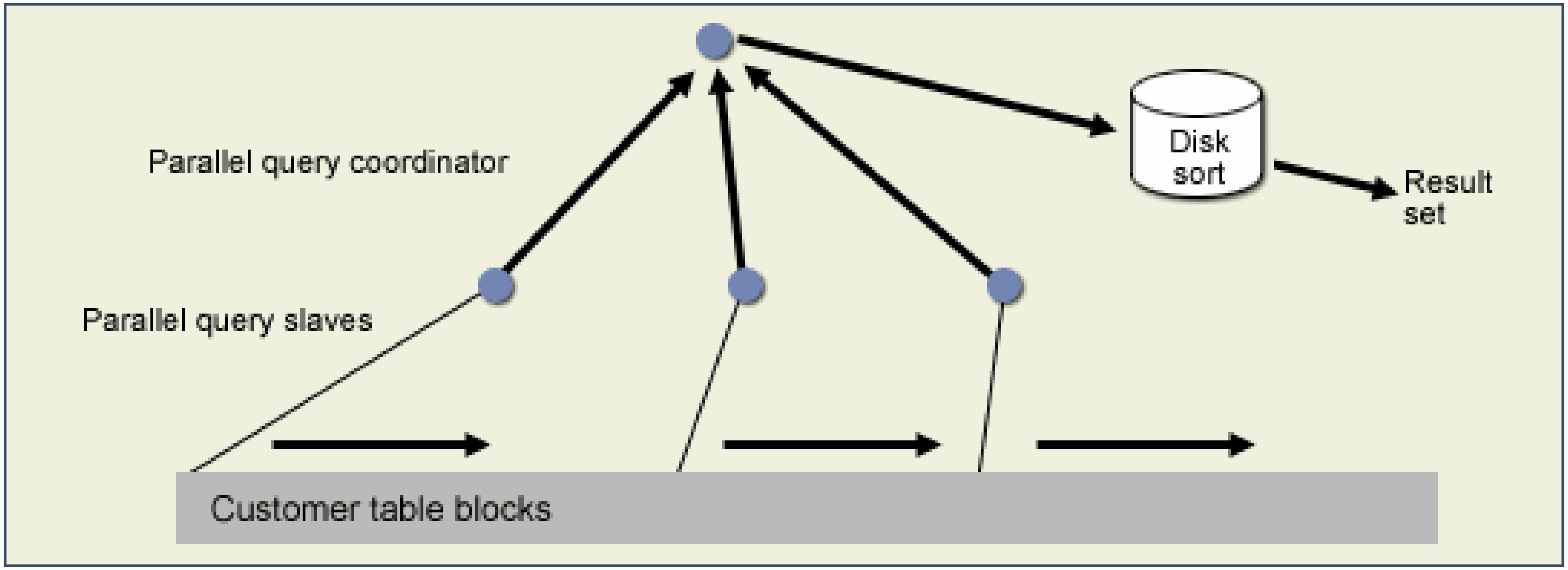
- Allocates one or two sets of slaves (threads or processes)

- Controls the query and sends instructions to the PQ slaves

- Determines which tables or indexes need to be scanned by the PQ slaves

- Produces the final output to the user

OPQ Architecture



Environment:

- PARALLEL_MAX_SERVER
- PARALLEL_AUTOMATIC_TUNING

- etc ...

Degree of Parallelism (DOP):

- Object Level Definition
- Statement Level (via hint)

Parallel Query Status

How many parallel servers are configured for the environment:

```
SELECT NAME, VALUE FROM v$parameter
WHERE NAME LIKE ‘%paral%max%’;
```

How many parallel servers are being used, and by whom:

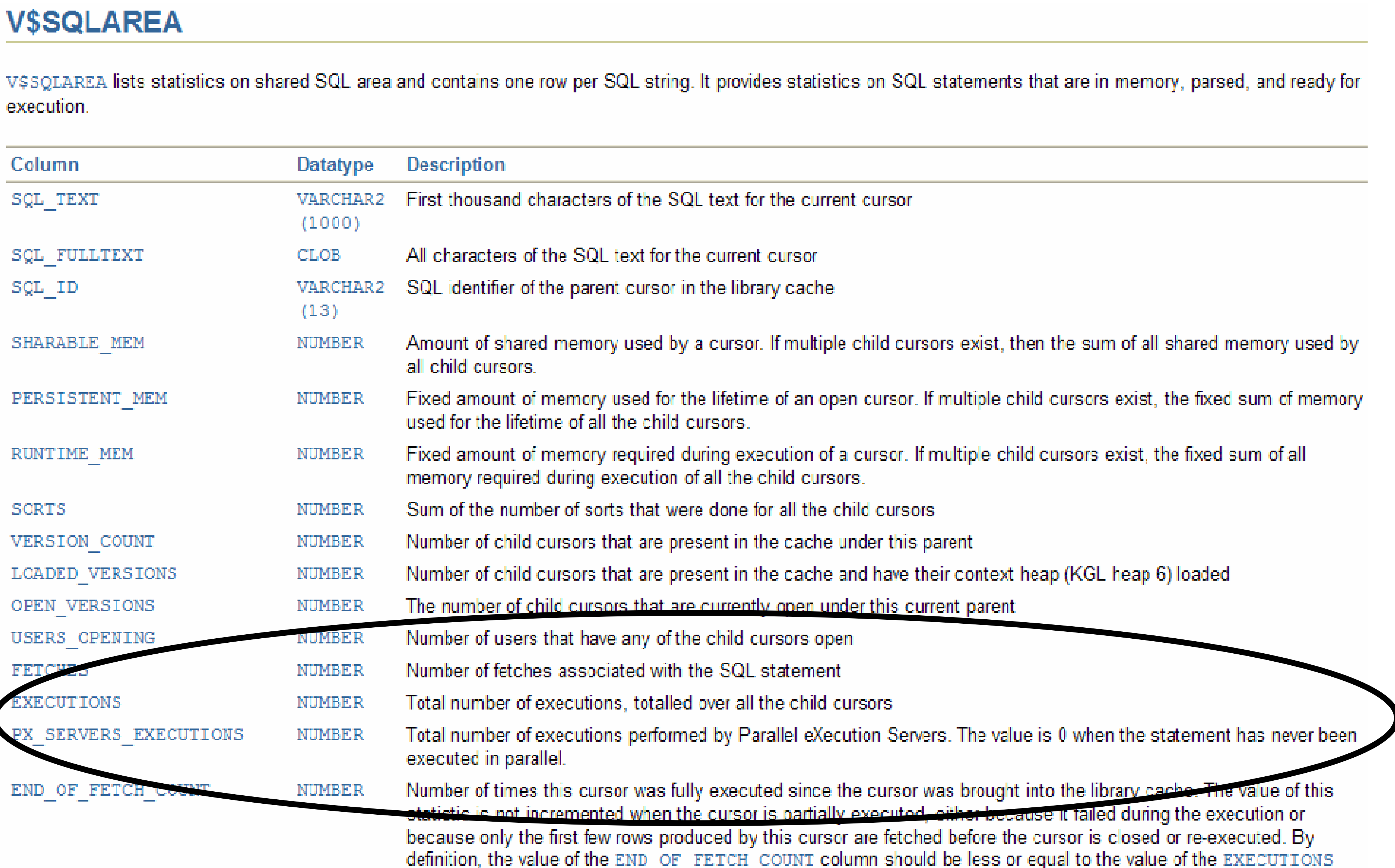
```
SELECT a.qcsid, a.qcserial#, y.osuser, COUNT(*)
FROM v$px_session a, v$session y
WHERE y.sid = a.qcsid AND y.serial# = a.qcserial#
GROUP BY a.qcsid, a.qcserial#, y.osuser;
```

Configuration Effectiveness

```
select name, value
from v$sysstat
where name like 'Parallel%'

Name                                          Value
-----------------------------------------     -------
Parallel operations not downgraded            18,694
Parallel operations downgraded to serial      476,364
Parallel operations downgraded 75 to 99 pct   15
Parallel operations downgraded 50 to 75 pct   137
Parallel operations downgraded 25 to 50 pct   214
Parallel operations downgraded 1 to 25 pct    85
```

- **Results like this would indicate that there are significant limitations in the OPQ configuration for this database**

PX_SERVERS_EXECUTION



V$SQLAREA

V$SQLAREA lists statistics on shared SQL area and contains one row per SQL string. It provides statistics on SQL statements that are in memory, parsed, and ready for execution.

Column	Datatype	Description
SQL_TEXT	VARCHAR2 (1000)	First thousand characters of the SQL text for the current cursor
SQL_FULLTEXT	CLOB	All characters of the SQL text for the current cursor
SQL_ID	VARCHAR2 (13)	SQL identifier of the parent cursor in the library cache
SHARABLE_MEM	NUMBER	Amount of shared memory used by a cursor. If multiple child cursors exist, then the sum of all shared memory used by all child cursors.
PERSISTENT_MEM	NUMBER	Fixed amount of memory used for the lifetime of an open cursor. If multiple child cursors exist, the fixed sum of memory used for the lifetime of all the child cursors.
RUNTIME_MEM	NUMBER	Fixed amount of memory required during execution of a cursor. If multiple child cursors exist, the fixed sum of all memory required during execution of all the child cursors.
SORTS	NUMBER	Sum of the number of sorts that were done for all the child cursors
VERSION_COUNT	NUMBER	Number of child cursors that are present in the cache under this parent
LOADED_VERSIONS	NUMBER	Number of child cursors that are present in the cache and have their context heap (KGL heap 6) loaded
OPEN_VERSIONS	NUMBER	The number of child cursors that are currently open under this current parent
USERS_OPENING	NUMBER	Number of users that have any of the child cursors open
FETCHES	NUMBER	Number of fetches associated with the SQL statement
EXECUTIONS	NUMBER	Total number of executions, totalled over all the child cursors
PX_SERVERS_EXECUTIONS	NUMBER	Total number of executions performed by Parallel eXecution Servers. The value is 0 when the statement has never been executed in parallel.
END_OF_FETCH_COUNT	NUMBER	Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS

Simplified in Oracle 10g

Determining the Average Degree of Parallelism

1. PX_SERVERS_EXECTUTIONS makes determining the average DOP for a query a lot easier with Oracle 10g.

2. By dividing the PX_SERVERS_EXECTUTIONS by the number of EXECUTIONS for the statement, the average DOP for the query over time can be obtained.

3. Pre-Oracle10g the only way to obtain the data was to either:
   - Query the statement as it was running (current state).
   - Analyze to post-execution of the statement through the V$PQ_TQSTAT view

Altered in Oracle 10g

The ELAPSED_TIME calculation

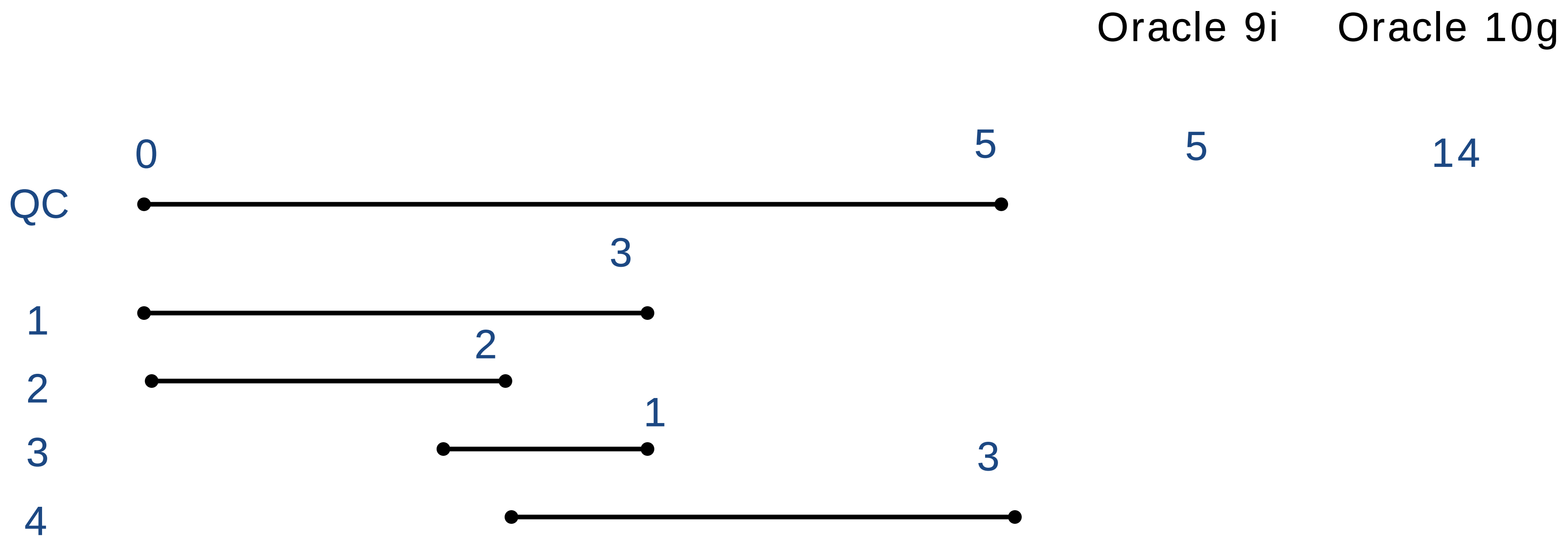
1. The ELAPSED_TIME column has been used in the past to calculate the average execution time, or wall clock time, for a query,

2. For the query coordinator session this value used to represent the wall clock value (end user experience) for the execution of the query

3. In Oracle 10g, this behavior changes; the ELAPSED_TIME value now includes the coordinator elapsed time and the aggregation of all the elapsed times for the slave queries.

4. The caution here is that using the ELPASED_TIME column for determining average execution time will result in an overstatement of time for parallel queries.

ELAPSED_TIME caution

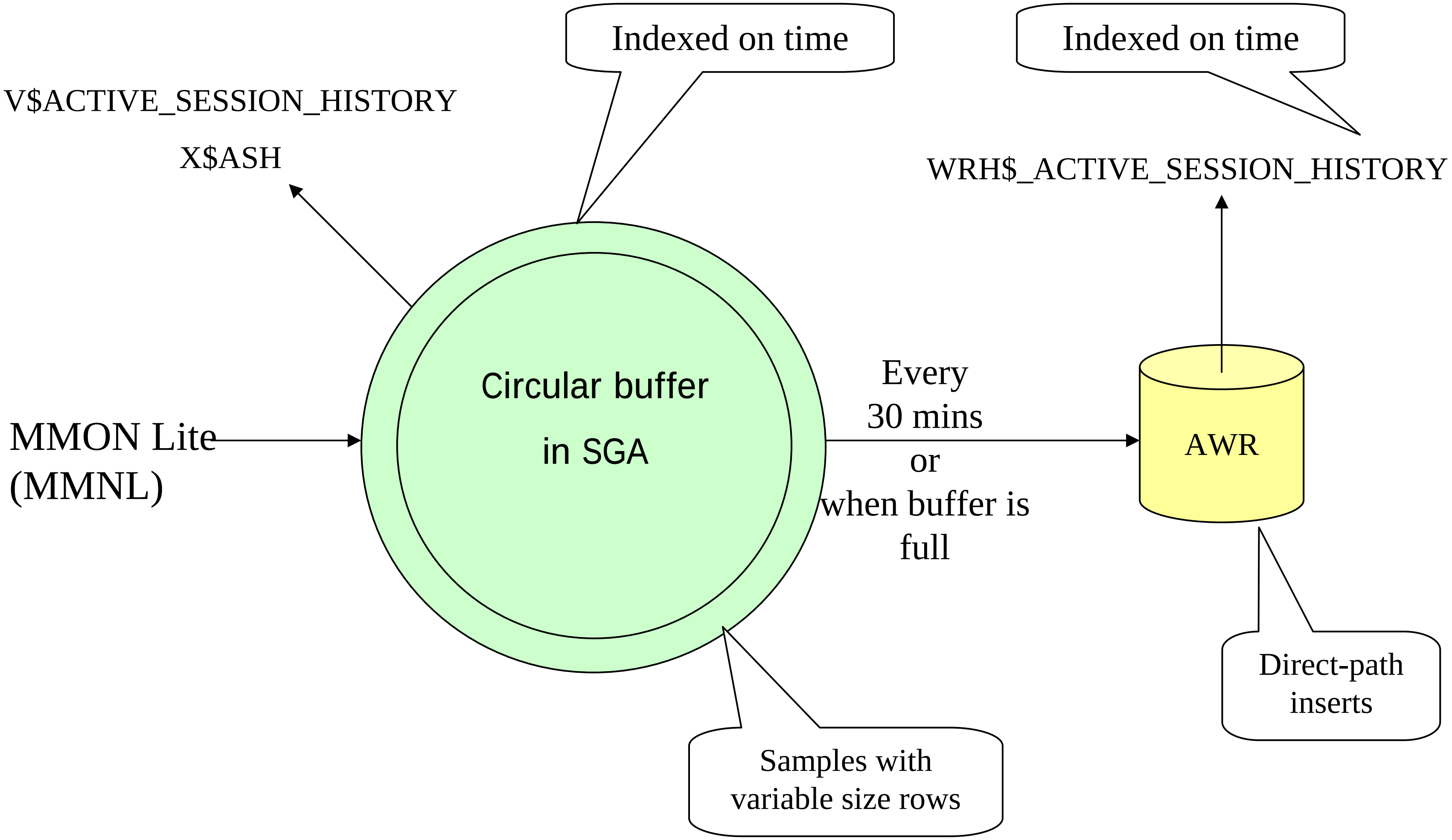


In Oracle 10g, ELAPSED_TIME reports the combined elapsed time for the query coordinator along with all the PX slave processes

ASH / AWR

- **New sources of Oracle database performance data available in 10*g***
- **ASH = V$SESSION_WAIT++ with History**
- **Provides historical information about <u>recently sampled</u> "active" sessions**
- **An *active session* is one which is in a user call**
  - **Parse**
  - **Execute**
  - **Fetch**
- **AWR is STATSPACK++**
- **Runs every 30 minutes (default) to create a snapshot**

ASH / AWR Architecture



Operational Details

- No installation or setup required
- 30-minute circular buffer in the SGA
- Dynamically adjusting session sampling algorithm uses < 0.1% of 1 CPU
- ASH on Disk persisted to the 10*g* workload repository (1 out of every 10 samples)
  - WRH$_ACTIVE_SESSION_HISTORY
- New Oracle Background Process
  - MMNL (MMON Lite)
- The sampler (MMNL) does not use any latches
- It supports dirty reads
- Can write to the in-memory buffer without any issues

Configuration Details

- Circular Buffer Sizing Formula: Max(Min (# of CPUs * 2MB, 5% of SHARED_POOL_SIZE, 30MB), 1MB)
- Default snapshot interval (30 minutes) can be adjusted by tweaking the INTERVAL parameter
- The shortest interval is 10 minutes
- For more frequent snapshots, execute DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE(...)
- init.ora
  - STATISTICS_LEVEL = TYPICAL (Default)
- Master Switch
  - _ACTIVE_SESSION_HISTORY = TRUE (Default)
- Automatically installed, populated and purged for 10*g* only databases
- Default retention for AWR is for 7 days

Components of AWR

- DBA_HIST* views – persistence across instance shutdowns
- New PL/SQL Package – DBMS_WORKLOAD_REPOSITORY.*
- Snapshot Data
  - Base statistics collection, Metrics collection, ASH on Disk
- AWR Report looks similar to a STATSPACK Report
- Master Switch – STATISTICS_LEVEL

Uses for this data

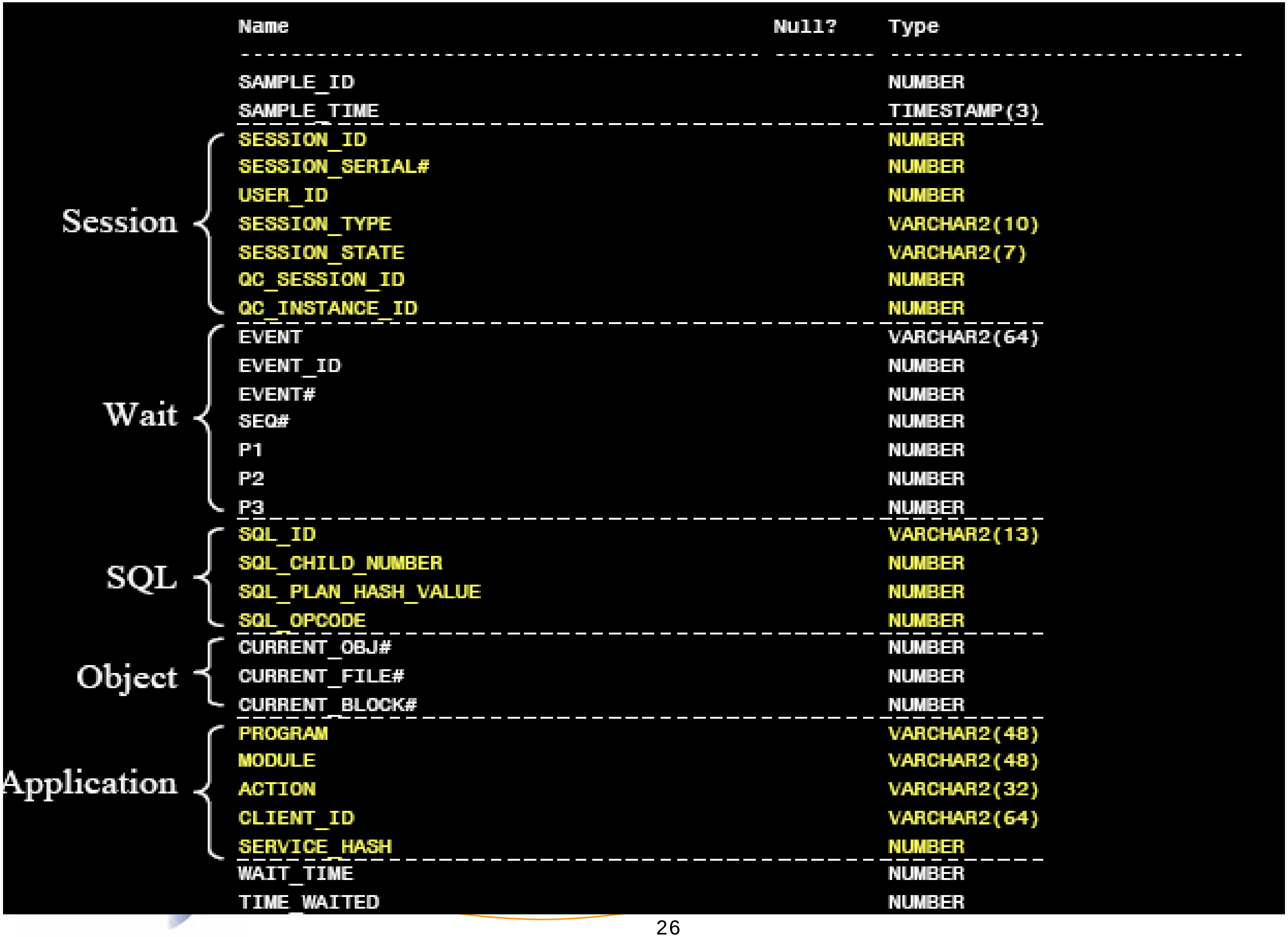
- Great for performance diagnostics
  - Logs wait events along with SQL details and session statistics in a circular buffer in memory
- Provides data for:
  - Automatic Database Diagnostic Monitor (ADDM)
  - Server-generated Alerts
  - Advisors (SQL, Memory, Undo, Index etc.)
  - Cost-based optimizer
  - EM Performance Reports
  - Ad Hoc Reporting **

** Diagnostic Pack license is required for ad-hoc access to the ASH / AWR data

Querying ASH data

- V$ACTIVE_SESSION_HISTORY
- X$ASH
- ASH on Disk persisted to the 10*g* workload repository (limited number of samples)
  - WRH$_ACTIVE_SESSION_HISTORY

V$active_session_history

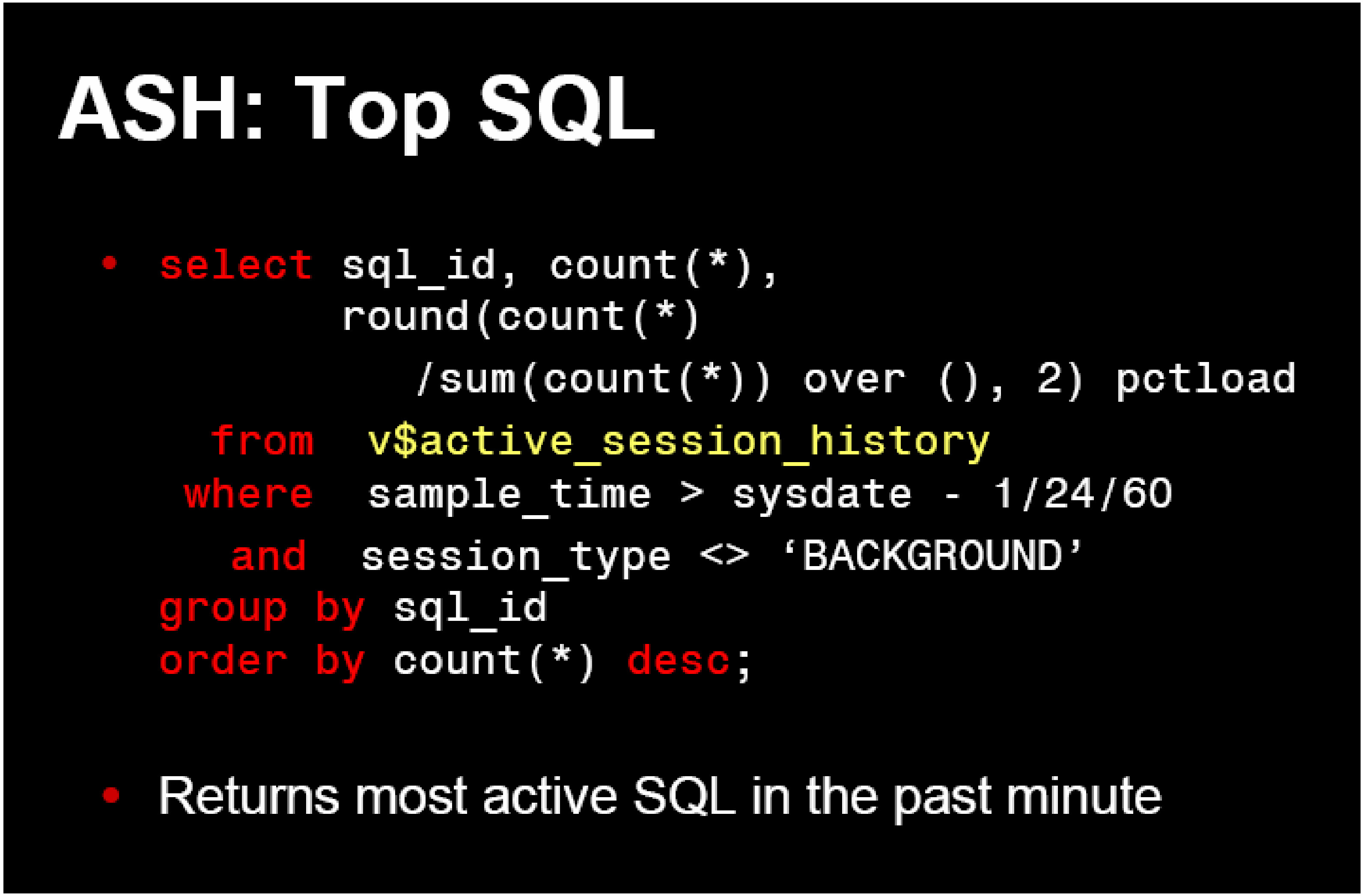


```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
SAMPLE_ID                                          NUMBER
SAMPLE_TIME                                        TIMESTAMP(3)
SESSION_ID                                         NUMBER
SESSION_SERIAL#                                    NUMBER
USER_ID                                            NUMBER
SESSION_TYPE                                       VARCHAR2(10)
SESSION_STATE                                      VARCHAR2(7)
QC_SESSION_ID                                      NUMBER
QC_INSTANCE_ID                                     NUMBER
EVENT                                              VARCHAR2(64)
EVENT_ID                                           NUMBER
EVENT#                                             NUMBER
SEQ#                                               NUMBER
P1                                                 NUMBER
P2                                                 NUMBER
P3                                                 NUMBER
SQL_ID                                             VARCHAR2(13)
SQL_CHILD_NUMBER                                   NUMBER
SQL_PLAN_HASH_VALUE                                NUMBER
SQL_OPCODE                                         NUMBER
CURRENT_OBJ#                                       NUMBER
CURRENT_FILE#                                      NUMBER
CURRENT_BLOCK#                                     NUMBER
PROGRAM                                            VARCHAR2(48)
MODULE                                             VARCHAR2(48)
ACTION                                             VARCHAR2(32)
CLIENT_ID                                          VARCHAR2(64)
SERVICE_HASH                                       NUMBER
WAIT_TIME                                          NUMBER
TIME_WAITED                                        NUMBER
```

View Definition

```
SQL> select view_definition from v$fixed_view_definition
  2* where view_name = 'GV$ACTIVE_SESSION_HISTORY';
     VIEW_DEFINITION
     -------------------------------------------------------
     SELECT  /*+ no_merge ordered use_nl(s,a) */
     a.inst_id, a.sample_id, a.sample_time, a.session_id,
     a.session_serial#, a.user_id, a.sql_id, a.sql_child_number,
     a.sql_plan_hash_value, a.sql_opcode, a.service_hash,
     decode(a.session_type, 1,'FOREGROUND',
     2,'BACKGROUND', 'UNKNOWN'), decode(a.wait_time, 0,
     'WAITING', 'ON CPU'), a.qc_session_id, a.qc_instance_id,
     a.seq#, a.event#, a.p1, a.p2, a.p3, a.wait_time,
     a.time_waited, a.current_obj#, a.current_file#,
     a.current_block#, a.program, a.module, a.action,
     a.client_id
     FROM  x$kewash s, x$ash a
     WHERE s.sample_addr = a.sample_addr and s.sample_id
     = a.sample_id
```

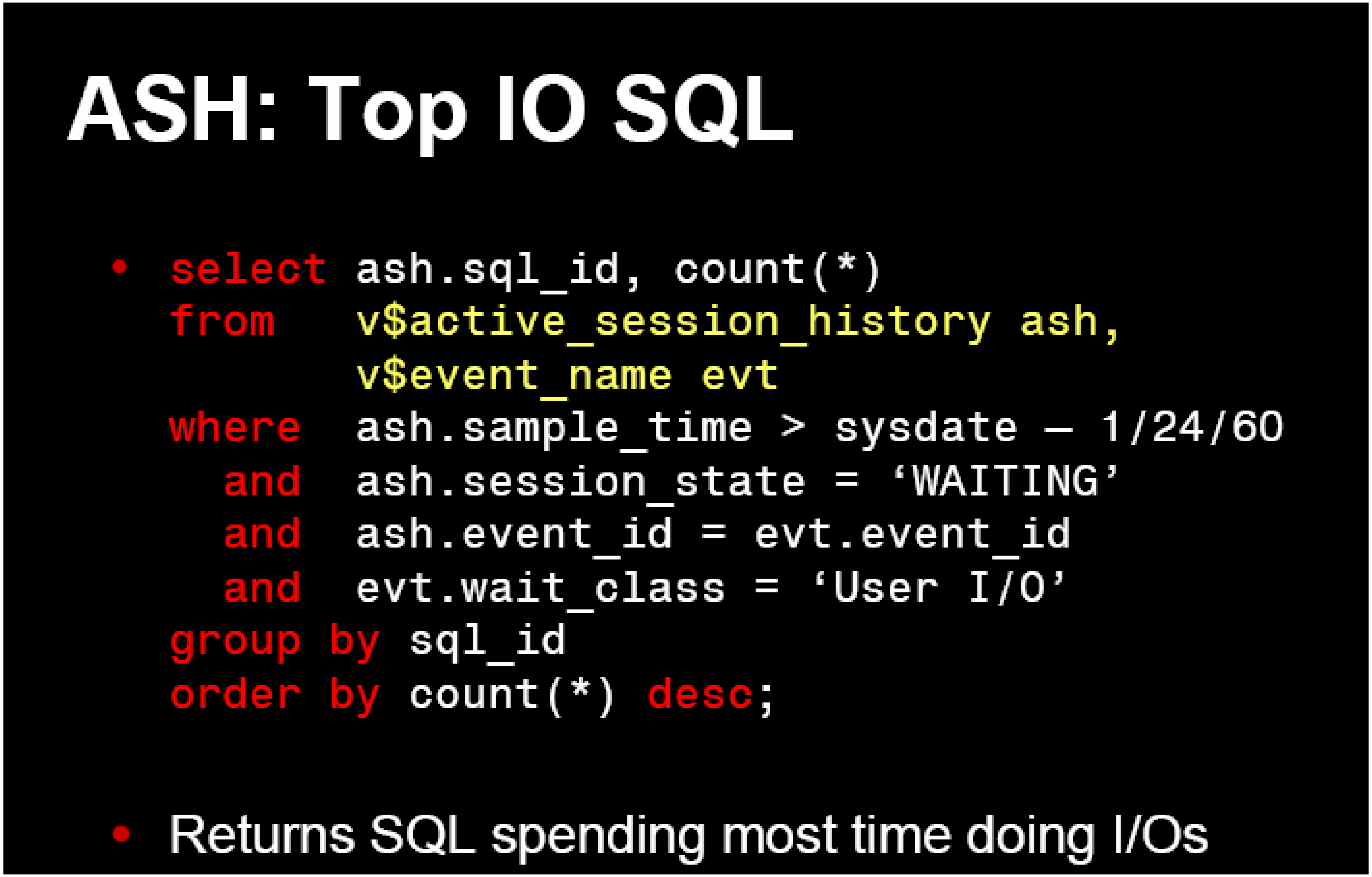
Ad-Hoc Queries



ASH: Top SQL

-
  ```
  select sql_id, count(*),
         round(count(*)
            /sum(count(*)) over (), 2) pctload
    from  v$active_session_history
   where  sample_time > sysdate - 1/24/60
     and  session_type <> 'BACKGROUND'
  group by sql_id
  order by count(*) desc;
  ```
- Returns most active SQL in the past minute

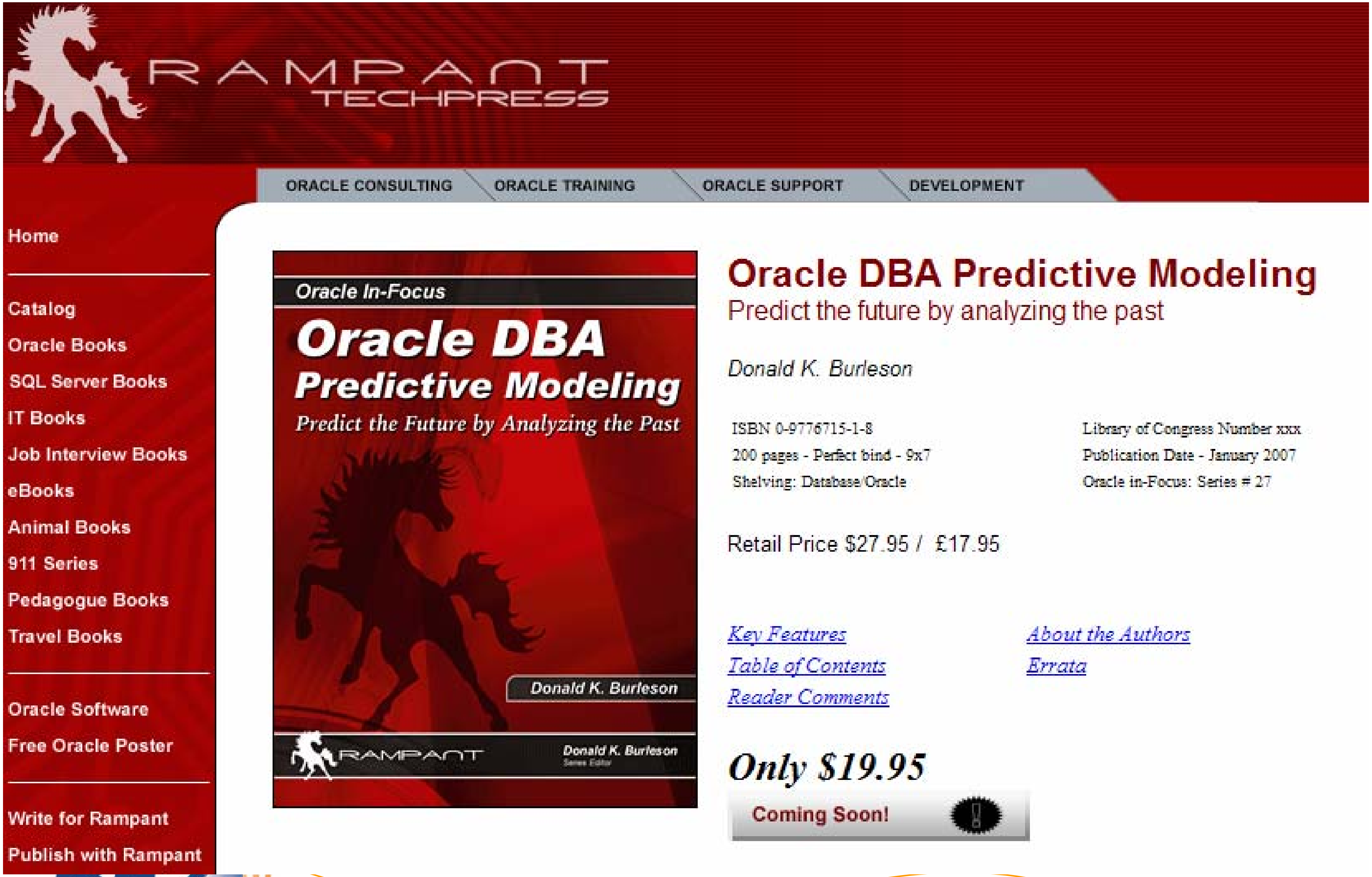
Ad-Hoc Queries



ASH: Top IO SQL

-
  ```
  select ash.sql_id, count(*)
  from   v$active_session_history ash,
         v$event_name evt
  where  ash.sample_time > sysdate – 1/24/60
    and  ash.session_state = ‘WAITING’
    and  ash.event_id = evt.event_id
    and  evt.wait_class = ‘User I/O’
  group by sql_id
  order by count(*) desc;
  ```
- Returns SQL spending most time doing I/Os

Lots of Reference Material



Oracle DBA Predictive Modeling

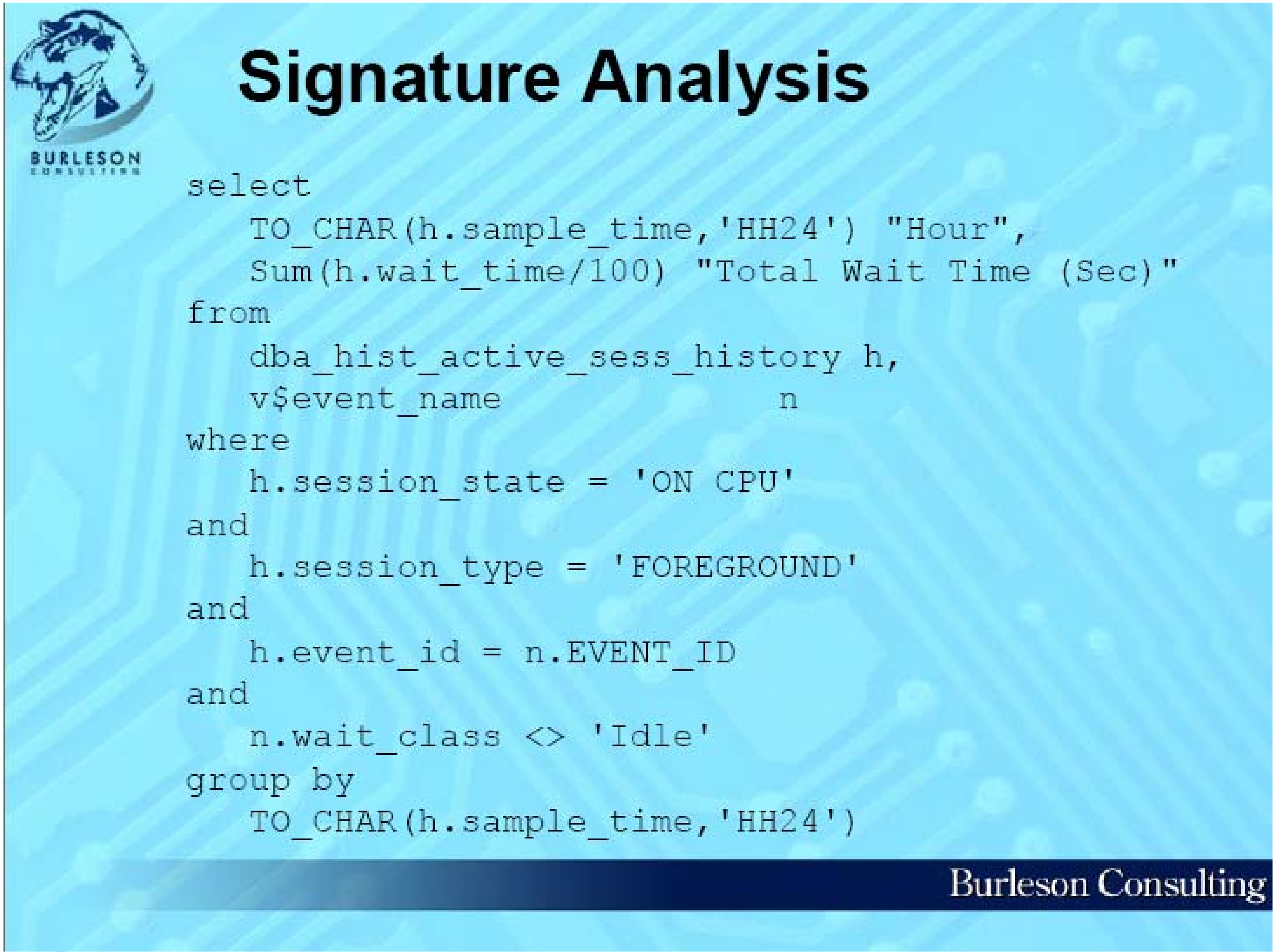
Predict the future by analyzing the past

Donald K. Burleson

ISBN 0-9776715-1-8	Library of Congress Number xxx
200 pages - Perfect bind - 9x7	Publication Date - January 2007
Shelving: Database/Oracle	Oracle in-Focus: Series # 27

Retail Price $27.95 / £17.95

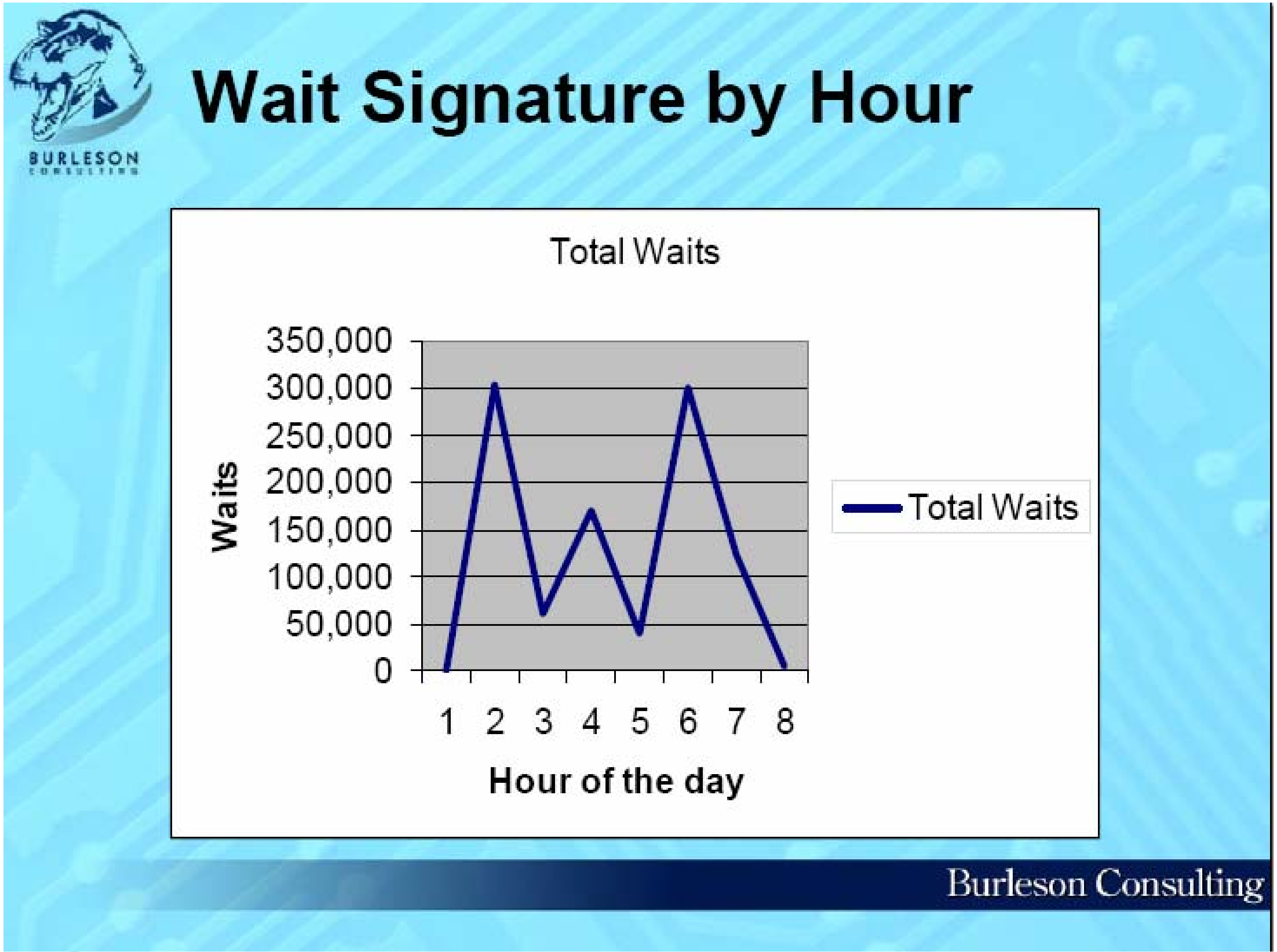
Only $19.95



Signature Analysis

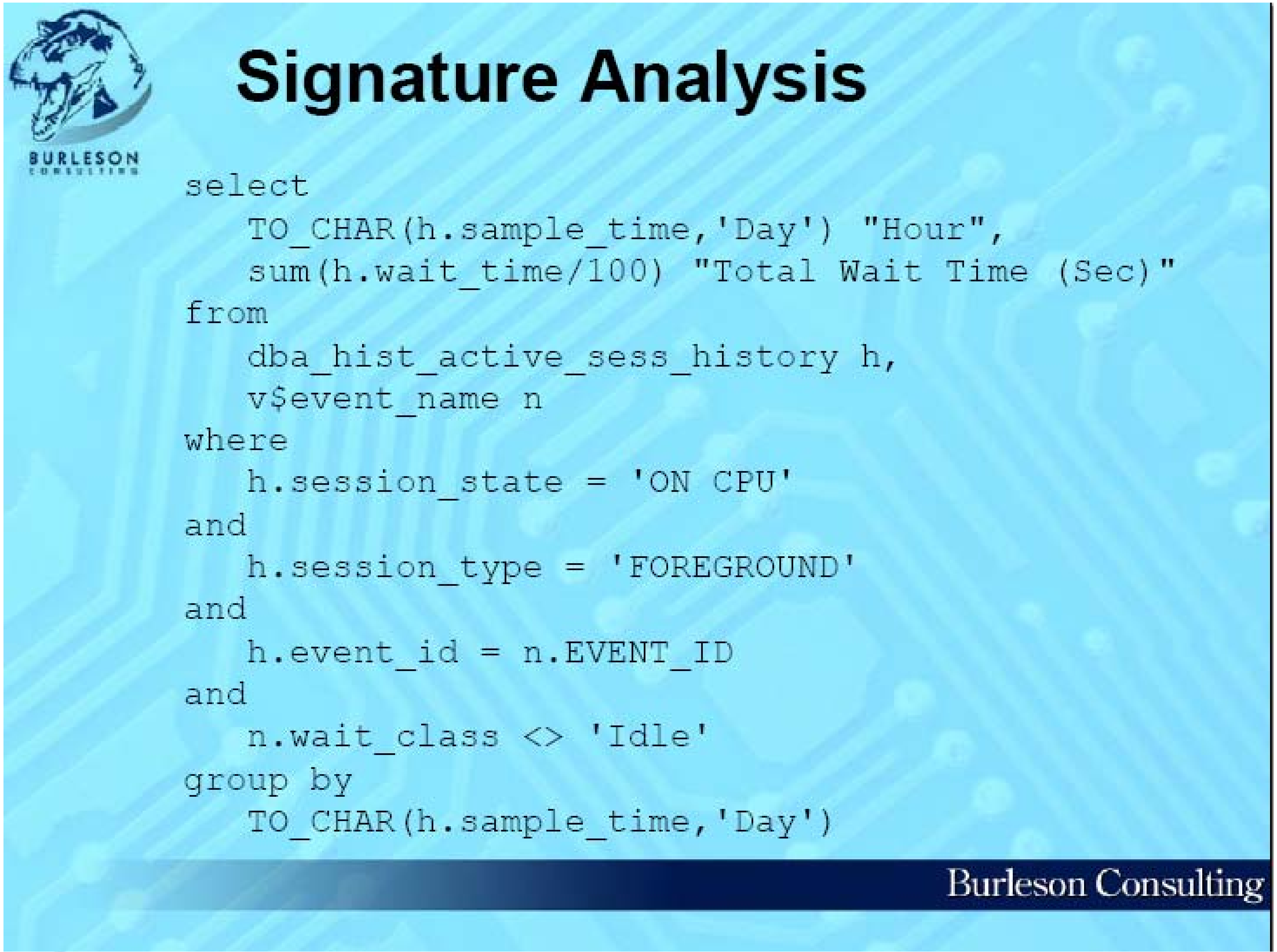
```
select
   TO_CHAR(h.sample_time,'HH24') "Hour",
   Sum(h.wait_time/100) "Total Wait Time (Sec)"
from
   dba_hist_active_sess_history h,
   v$event_name             n
where
   h.session_state = 'ON CPU'
and
   h.session_type = 'FOREGROUND'
and
   h.event_id = n.EVENT_ID
and
   n.wait_class <> 'Idle'
group by
   TO_CHAR(h.sample_time,'HH24')
```

Burleson Consulting



Wait Signature by Hour

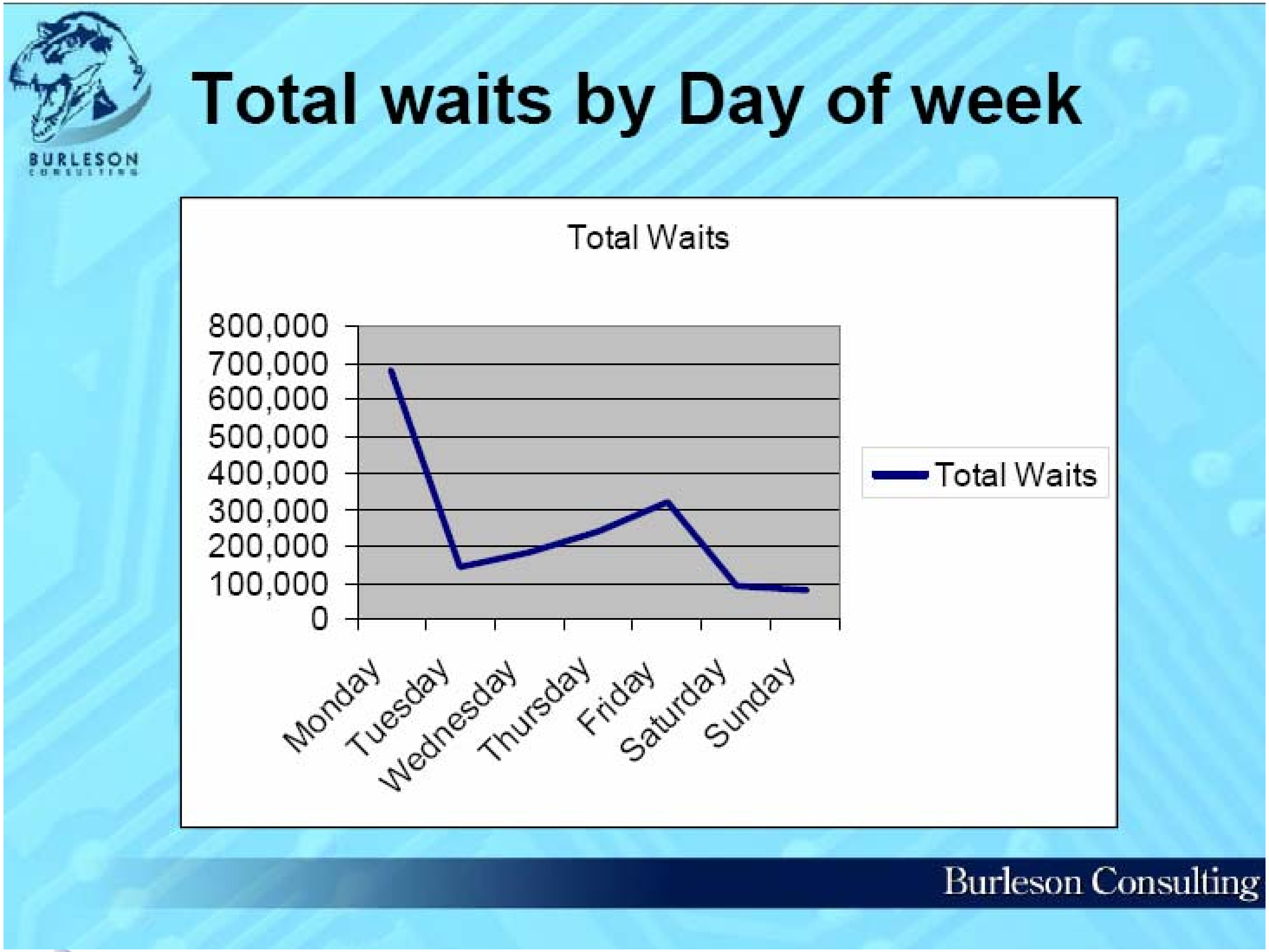
Burleson Consulting



Signature Analysis

```
select
   TO_CHAR(h.sample_time,'Day') "Hour",
   sum(h.wait_time/100) "Total Wait Time (Sec)"
from
   dba_hist_active_sess_history h,
   v$event_name n
where
   h.session_state = 'ON CPU'
and
   h.session_type = 'FOREGROUND'
and
   h.event_id = n.EVENT_ID
and
   n.wait_class <> 'Idle'
group by
   TO_CHAR(h.sample_time,'Day')
```

Burleson Consulting



Total waits by Day of week

Burleson Consulting

Limitations of this solution

- Querying V$ACTIVE_SESSION_HISTORY needs a session
  - Logins may be impossible on a fully loaded system.
- Querying V$ACTIVE_SESSION_HISTORY requires all relevant latches in the SQL layer
  - ASH will impose even more overhead on *shared pool* and *library cache latches*
  - This can result in a significant increase in waits for these latches.
- Each database has its own AWR repository
- Stored in your “production database”
- Any access to it incurs a database layer overhead on the production box
- You cannot modify/drop AWR tables
- **Ad hoc querying of the collected data requires the Diagnostic Pack license.**

Diagnostic Pack

Database Diagnostic Pack includes:

- use of **DBMS_WORKLOAD_REPOSITORY** package
- use of DBMS_ADVISOR PACK if:
- when using any ADDM prefix for the value of the ADVISOR_NAME parameter
- when using any ADDM prefix for the value of the TASK_NAME parameter
- use of the view **V$ACTIVE_SESSION_HISTORY**
- use of any Data Dictionary view the **DBA_HIST_** prefix in the name of the view
- use of any Data Dictionary view the DBA_ADVISOR_ prefix in the name of the view if query to these views retuns values from the ADDM or ADVISOR_NAME column or a value of ADDM* inthe TASK_NAME column or the corresponding TASK_ID

Time Model Views

- Another new source of Oracle database performance data available in 10*g*
- Determines the relative time allocation breakdown for end user activities.
- Data is available at the session or the instance level.
  - V$SESS_TIME_MODEL
  - V$SYS_TIME_MODEL
- High level view to pinpoint where the investigation should begin.
- Hierarchy of metrics for analysis.

View Content

V$SESS_TIME_MODEL and V$SYS_TIME_MODEL Contents

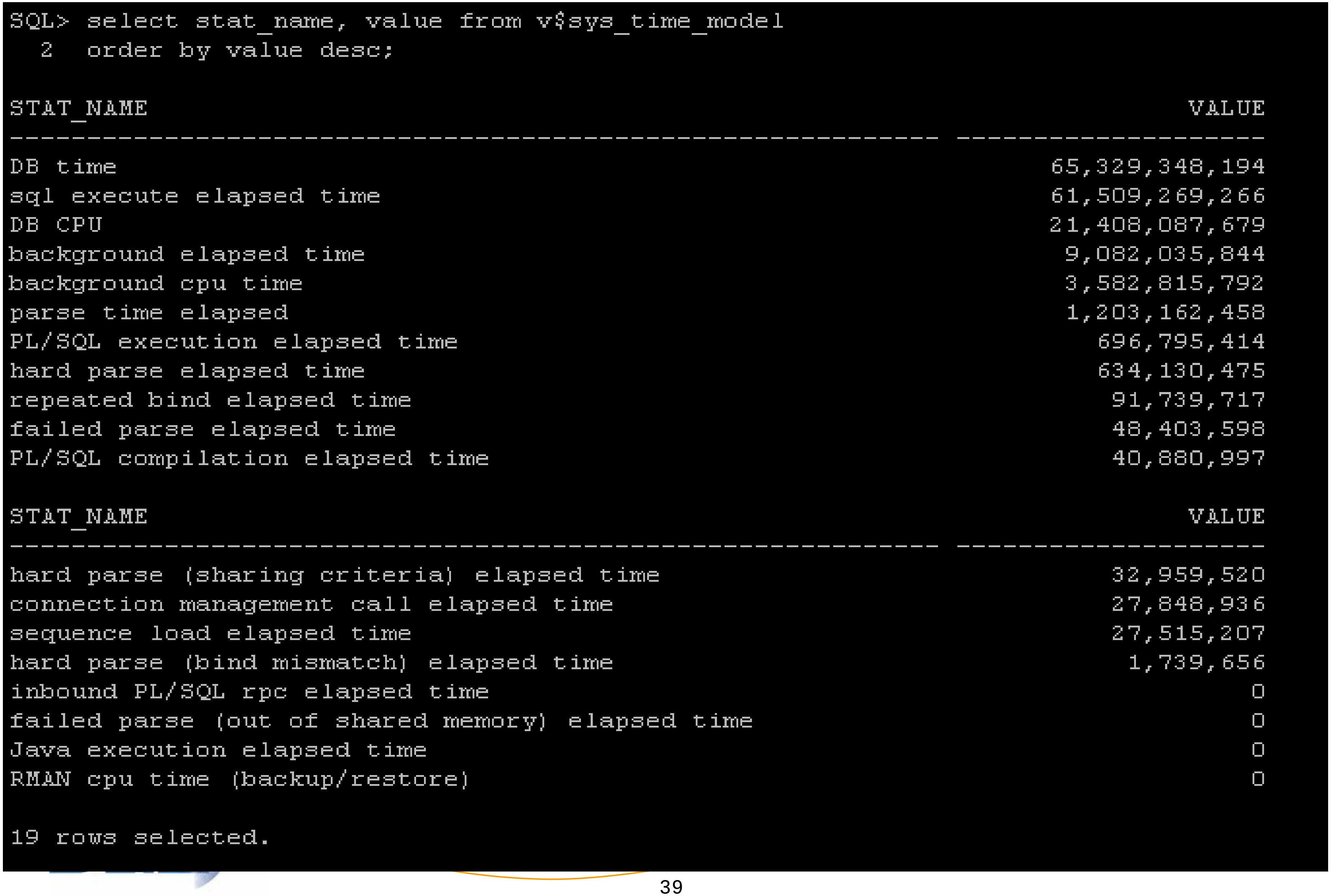
•**SID (NUMBER)** - Session Identifier. This is the same value that you will find in all views that record information about individual sessions. This column is useful if you want to join V$SESS_TIME_MODEL to V$SESSION to retrieve additional information on the session being evaluated. As stated previously, since V$SYS_TIME_MODEL records information at the instance level, you won't find this column in the view.

•**STAT_ID (NUMBER)** Statistic identifier for the time statistic.

•**STAT_NAME (VARCHAR2 64) Name of the statistic being recorded. A listing of all of the statistics is provided later in the presentation.**

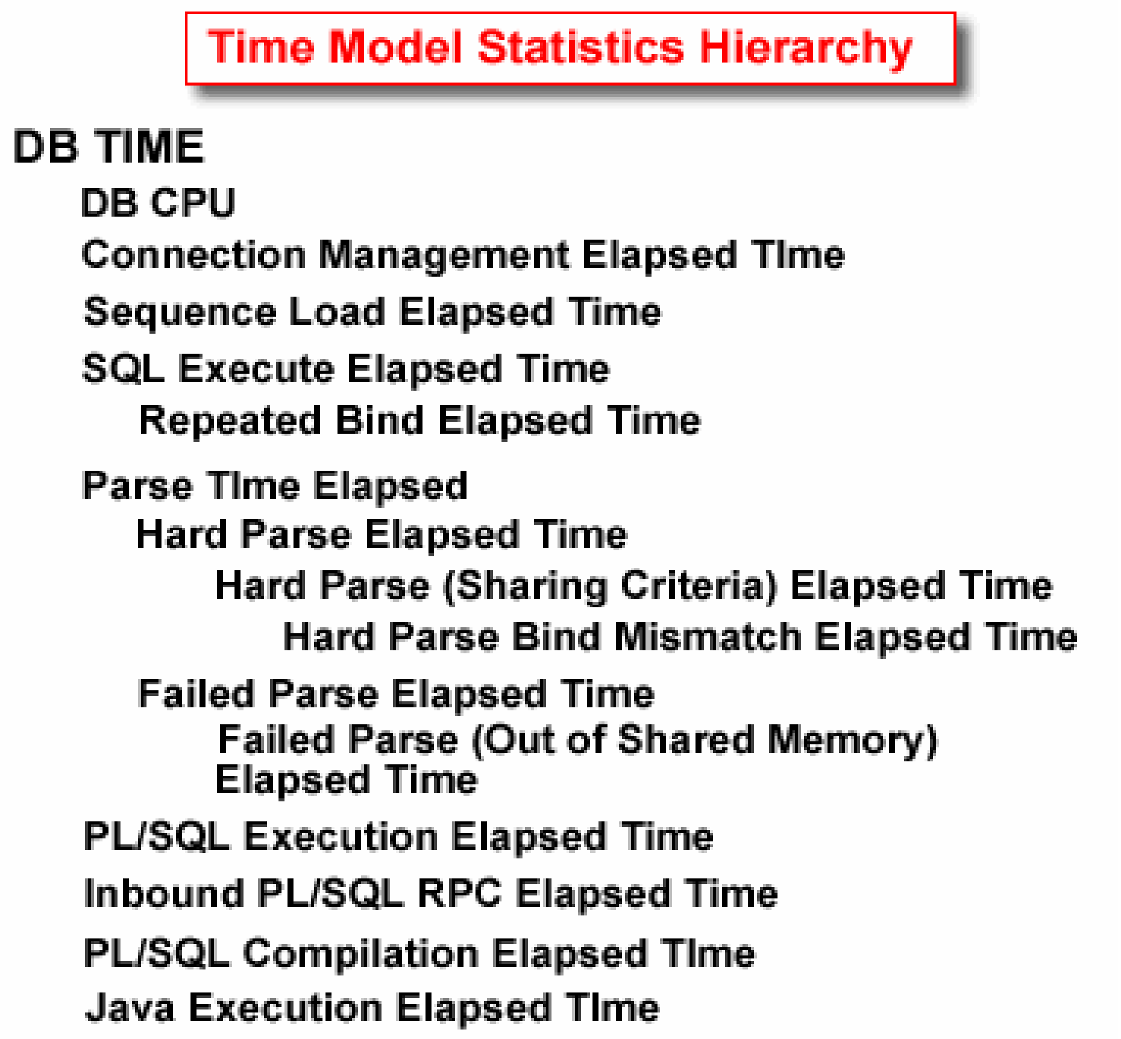
•**VALUE (NUMBER)** - Amount of time, in microseconds, the session has spent performing the operation identified in the STAT_NAME column.

Statistical Categories



```
SQL> select stat_name, value from v$sys_time_model
  2  order by value desc;

STAT_NAME                                                                   VALUE
------------------------------------------------------------ --------------------
DB time                                                            65,329,348,194
sql execute elapsed time                                           61,509,269,266
DB CPU                                                             21,408,087,679
background elapsed time                                             9,082,035,844
background cpu time                                                 3,582,815,792
parse time elapsed                                                  1,203,162,458
PL/SQL execution elapsed time                                         696,795,414
hard parse elapsed time                                               634,130,475
repeated bind elapsed time                                             91,739,717
failed parse elapsed time                                              48,403,598
PL/SQL compilation elapsed time                                        40,880,997

STAT_NAME                                                                   VALUE
------------------------------------------------------------ --------------------
hard parse (sharing criteria) elapsed time                             32,959,520
connection management call elapsed time                                27,848,936
sequence load elapsed time                                             27,515,207
hard parse (bind mismatch) elapsed time                                 1,739,656
inbound PL/SQL rpc elapsed time                                                 0
failed parse (out of shared memory) elapsed time                                0
Java execution elapsed time                                                     0
RMAN cpu time (backup/restore)                                                  0

19 rows selected.
```

Statistics Hierarchy



Time Model Statistics Hierarchy

- DB TIME
  - DB CPU
  - Connection Management Elapsed TIme
  - Sequence Load Elapsed Time
  - SQL Execute Elapsed Time
    - Repeated Bind Elapsed Time
  - Parse TIme Elapsed
    - Hard Parse Elapsed Time
      - Hard Parse (Sharing Criteria) Elapsed Time
        - Hard Parse Bind Mismatch Elapsed Time
    - Failed Parse Elapsed Time
      - Failed Parse (Out of Shared Memory) Elapsed Time
  - PL/SQL Execution Elapsed Time
  - Inbound PL/SQL RPC Elapsed Time
  - PL/SQL Compilation Elapsed TIme
  - Java Execution Elapsed TIme

DB TIME Definition

DB time –
Amount of time spent performing operations in the database. You compare this value against all of the other values contained in this table to determine where the bulk of the time is being spent.

- **Key metric from which all other metrics are based.**
- **Use DB TIME as the denominator when determining the percentage of time spent in a specific operation.**
- **“DB CPU” / “DB TIME” would indicate the percentage of end user processing time attributed to CPU activity.**

Statistics Definitions

- **DB CPU** - Amount of CPU time spent performing operations in the database. Like DB TIME, DB CPU only records user workload. If you see a relatively high value in this time look for complex calculations and poor SQL plans that perform a high level of buffer gets.

- **Connection Management Call Elapsed Time** - This value represents the amount of time processes spent performing CONNECT and DISCONNECT calls. This time should be much lower than most of the other values contained in this table. If it is high in V$SYS_TIME_MODEL, review program code to ensure that the application isn't attempting to make a connection and disconnection for each interaction with the database. This is a very common problem for applications built using a middle tier application server.

- **Sequence Load Elapsed Time** - Amount of time spent obtaining the next sequence number from Oracle's Data Dictionary. If the sequence is cached, the time is not recorded. A sequence is a user created object that generates numbers according to a specific pattern. It is most often used to generate unique identifiers to identify a specific object.

- **SQL Execute Elapsed Time** - This is a very important measurement. It records the amount of time that SQL statements are executing. This will also record the amount of time SELECT statements spend fetching the query results. Typically, user processes spend the bulk of their time accessing data. If it is extremely high, look for poorly performing SQL statements. If the user process is having performance problems and this indicator does not make up the majority of overall DB Time, check for poorly written programs, parsing problems, poor connection management, network issues.

- **Repeated Bind Elapsed Time** - Elapsed time spent on re-binding.

- **Parse Elapsed Time** - This records the amount of time the process spent hard parsing and soft parsing SQL statements. Oracle parses each statement before it is executed. The parse process includes syntax checking (making sure you spelled "WHERE" right), ensuring the objects being accessed are actually in the database, security checking, execution plan creation and loading the parsed representations into the shared pool.

Statistics Definitions

- **Hard Parse Elapsed Time** - The amount of time spent hard parsing a SQL statement before execution. Before a statement enters the parse phase, Oracle matches the statement being executed to statements that are already parsed and stored in the shared pool. When Oracle finds a matching statement in the shared pool, it will do a soft parse on the SQL statement. If Oracle does not find any matching SQL in the shared pool, it will perform a hard parse, which requires that more steps be performed than its soft parse counterpart. As with almost everything else, fewer steps = faster performance.

- **Hard Parse (Sharing Criteria) Elapsed Time** - This value represents the amount of elapsed time the database was forced to hard parse a SQL statement because it was unable to find an existing cursor in the SQL Cache. It is a subset of the Hard Parse Elapsed time. If this value is high, look for programs that don't use bind variables.

- **Hard Parse (Bind Mismatch) Elapsed Time** - Signifies the amount of elapsed time spent performing hard parses because the bind variable's type or bind size did not match existing cursors in the cache. Oracle documentation also states that bind type mismatches often causes indexes not to be used. If this value is high, look for poor programming standards.

- **Failed Parse Elapsed Time** - Records the time spent attempting, and ultimately failing, to parse a statement before execution. Check the program code to ensure that the statements are syntactically correct.

- **Failed Parse (Out of Shared Memory)** - Pretty self explanatory. The amount of elapsed time recorded when a parse failed because of a lack of adequate resources allocated to the shared pool. If this time is high, you need to add more memory to the shared pool, identify if bind variables are being used, etc.

Statistics Definitions

- <u>PL/SQL Execution Elapsed Time</u> - Amount of elapsed time spent running the PL/SQL interpreter. It does not include the time executing and parsing SQL statements or the amount of time the process spent recursively executing the Java VM.

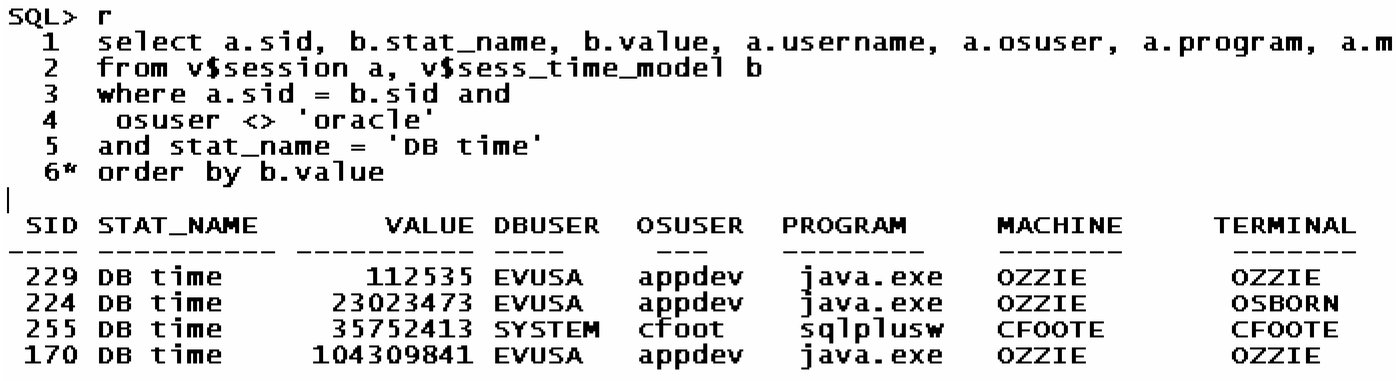
- <u>PL/SQL Compilation Elapsed Time</u> - Elapsed time spent running the PL/SQL compiler. It is the compiler's job to transform PL/SQL source code into machine-readable code (m-code).

- <u>Inbound PL/SQL RPC Elapsed Time</u> - Records the elapsed time PL/SQL remote procedure calls spent executing including executing SQL and JAVA.

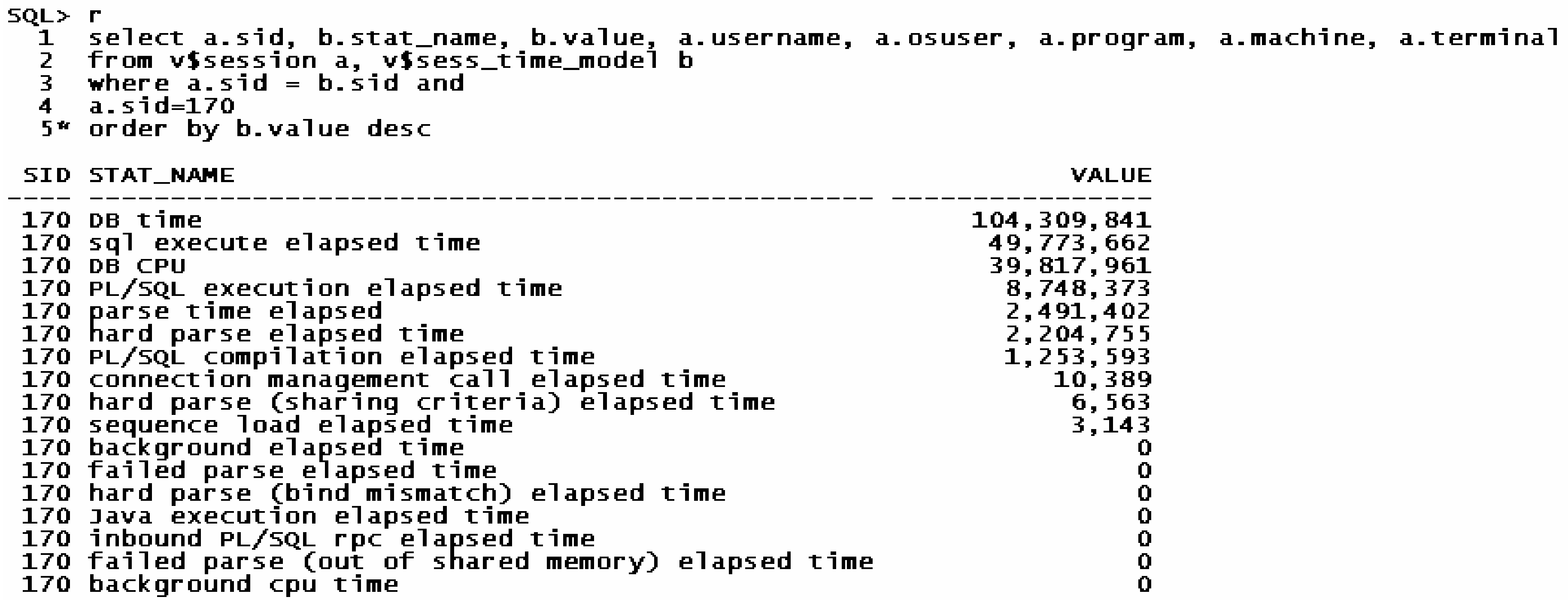
- <u>Java Execution Elapsed Time</u> - Amount of time spent running the JAVA machine. This time does

  not include the time spent executing and parsing SQL statements or recursively executing PL/SQL.

Session Summary

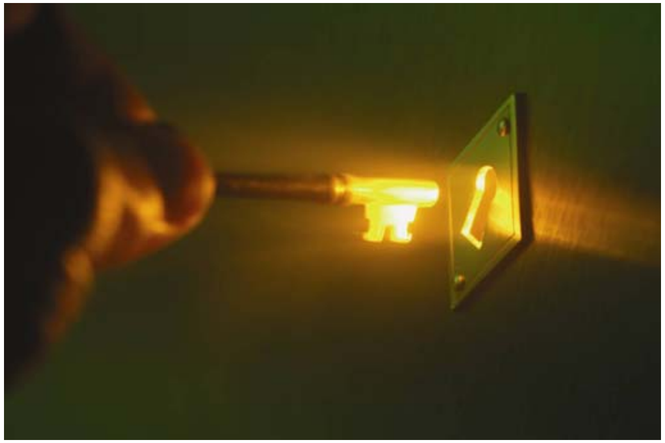


```
SQL> r
  1  select a.sid, b.stat_name, b.value, a.username, a.osuser, a.program, a.machine, a.terminal
  2  from v$session a, v$sess_time_model b
  3  where a.sid = b.sid and
  4   osuser <> 'oracle'
  5  and stat_name = 'DB time'
  6* order by b.value

 SID STAT_NAME        VALUE DBUSER   OSUSER   PROGRAM     MACHINE     TERMINAL
---- ---------- ----------- ----     ---      ---------   -------      -------
 229 DB time         112535 EVUSA    appdev    java.exe   OZZIE        OZZIE
 224 DB time       23023473 EVUSA    appdev    java.exe   OZZIE        OSBORN
 255 DB time       35752413 SYSTEM   cfoot     sqlplusw   CFOOTE       CFOOTE
 170 DB time      104309841 EVUSA    appdev    java.exe   OZZIE        OZZIE
```



```
SQL> r
  1  select a.sid, b.stat_name, b.value, a.username, a.osuser, a.program, a.machine, a.terminal
  2  from v$session a, v$sess_time_model b
  3  where a.sid = b.sid and
  4  a.sid=170
  5* order by b.value desc

 SID STAT_NAME                                                   VALUE
---- ------------------------------------------------ ----------------
 170 DB time                                               104,309,841
 170 sql execute elapsed time                               49,773,662
 170 DB CPU                                                 39,817,961
 170 PL/SQL execution elapsed time                           8,748,373
 170 parse time elapsed                                      2,491,402
 170 hard parse elapsed time                                 2,204,755
 170 PL/SQL compilation elapsed time                         1,253,593
 170 connection management call elapsed time                    10,389
 170 hard parse (sharing criteria) elapsed time                  6,563
 170 sequence load elapsed time                                  3,143
 170 background elapsed time                                         0
 170 failed parse elapsed time                                       0
 170 hard parse (bind mismatch) elapsed time                         0
 170 Java execution elapsed time                                     0
 170 inbound PL/SQL rpc elapsed time                                 0
 170 failed parse (out of shared memory) elapsed time                0
 170 background cpu time                                             0
```

Thank You!



Please visit our booth in the vendor area.

www.BEZ.com | 617-532-8800 | info@bez.com