DB Time Performance Tuning: Theory and Practice

Graham Wood
Oracle America

DOAG, Nov 2013
Agenda

• Time
  • Database Time
  • Average Active Sessions

• Techniques
  • The DB Time Method

• Tools
  • ADDM
  • EM User Interface
  • Active Reports
Oracle Tuning Methods: A History

- Prehistoric (v5)
  - Debug code
- Dark Ages (v6)
  - Counters/Ratios
  - BSTAT/ESTAT
  - SQL*Trace
- Renaissance (v7/v8)
  - Introduction of Wait Event instrumentation
  - Move from counters to timers
  - STATSPACK
- Modernity (v10/v11)
  - DB Time Tuning – Tuning using fundamental notion of time spent in database
  - Multiple scoping levels
  - Always on, non-intrusive
  - Built into infrastructure: instrumentation, ASH, AWR, ADDM, EM
  - SQL monitoring
Why Do We Care About Time?

• Human time is critical to the enterprise

• Systems performance affects business goals
  • Human time + technology resource time

• “Time is money”

• Performance improvement means doing things faster

*Performance is always and only about time*
Database Time and Average Active Sessions
Database Time (DB Time)

- Total time in database calls by foreground sessions
- Includes CPU time, IO time and non-idle wait time
- DB Time <> response time
- Common currency for Oracle performance analysis

Database time is total time spent by user processes either actively working or actively waiting in a database call.
A Single Session

Single session with Database Black Box server

- Browse Books
- Read Reviews For One Book
- Add to Cart
- Checkout

TIME

= time spent in database
Fundamental Concepts

Database Time (DB Time) =
Total time session spent in all database calls

Active Session =
Session currently spending time in a database call

Average Activity of the Session (% Activity) =
The ratio of time active to total wall clock time

- Browse Books
- Read Reviews For One Book
- Add to Cart
- Checkout

= time spent in database

TIME
Multiple Sessions

DB Time = Sum of DB Time Over All Sessions

Avg. Active Sessions = Sum of Avg. Activity Over All Sessions

At time $t$ we have 2 active sessions

= time spent in database
Visualizing DB Time

Avg. Active Sessions = \frac{\text{Total Database Time}}{\text{Wall Clock (Elapsed) Time}}
• Active Sessions by wait class over time
• Colored area = amount of DB time
• “Click on the big stuff”
Average active sessions

\[ \text{= DB time / elapsed time} \]

- Time-normalized DB time
- Time units in numerator and denominator must synchronize to produce the proper metric
Host Performance and DB Time

• Host is CPU-bound
  • => foregrounds accumulate active run-queue time
    • => wait event times are artificially inflated
    • => DB time increases

_Tune for CPU before waits when CPU constrained_
CPU Run-queue and DB Time

DB time is inflated when host is CPU-bound
System performance and DB time

CPU or I/O problem?
Instrumentation: Where to find DB Time?

- **V$SYS_TIME_MODEL, V$SESS_TIME_MODEL**
  - STAT_NAME = ‘DB time’

- **V$SYSMETRIC_HISTORY**
  - “Database Time Per Second”, “CPU Usage Per Sec”
  - 10g units = centi-secs/sec (100xAvg. Active Sessions)
  - 11g new metric “Average Active Sessions”

- **V$SQL**
  - ELAPSED_TIME and CPU_TIME
  - Wait class times: APPLICATION, CONCURRENCY, CLUSTER, USER_IO

- **V$ACTIVE_SESSION_HISTORY**
Active Session History
Active Session History (ASH)

- All ‘Active’ sessions captured every second
  - Foregrounds and backgrounds are sampled
  - Active foregrounds contribute to DB Time

- In-memory: V$ACTIVE_SESSION_HISTORY
  - Sampling interval = 1 second

- On-disk: DBA_HIST_ACTIVE_SESS_HISTORY
  - Sampling interval = 10 second

- ASH is a system-wide record of database activity
COUNT(*) = DB Time

GROUP BY ?
ASH Math: COUNT(*)=DB Time

- ASH is a big fact table
  - Each row represents 1-second of active session time

- V$ACTIVE_SESSION_HISTORY
  - COUNT(*) = DB time in seconds

- DBA_HIST_ACTIVE_SESS_HISTORY
  - COUNT(*) * 10 = DB time in seconds
Estimating DB Time with ASH

- **ASH sample counts = DB Time** in seconds
  - Low sample counts are less reliable

- Enables DB Time analysis over many dimensions
  - Sqlid, session id, instance, service, module, action
  - 10gR2
    - Blocking_sid (10gR2)
    - XID
  - 11g
    - Row source
    - Execution ID
    - Operation type
      - Connect
      - Java/SQL/PLSQL
      - parse, bind, execute/fetch, close
Example: DB Time by SQL ID

```
select sql_id,
       count(*) DBTime,
       round(count(*)*100/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;
```
Example: DB Time by SQL ID

```sql
select sql_id
    , count(*) DBTime
    , round(count(*)*100/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;
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>DBTIME</th>
<th>PCTLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>6bmxrabnwwwsxd</td>
<td>60</td>
<td>63.83</td>
</tr>
<tr>
<td>azzsynnmz43nrr</td>
<td>8</td>
<td>8.51</td>
</tr>
<tr>
<td>28pb73sbwhmm8</td>
<td>5</td>
<td>5.32</td>
</tr>
<tr>
<td>58psyvgau23s2</td>
<td>3</td>
<td>3.19</td>
</tr>
<tr>
<td>amrq8hk767tuz</td>
<td>2</td>
<td>2.13</td>
</tr>
<tr>
<td>2r5qhb3fb63vm</td>
<td>1</td>
<td>1.06</td>
</tr>
<tr>
<td>f3919usqp5wj2</td>
<td>1</td>
<td>1.06</td>
</tr>
</tbody>
</table>
Avg Active Sessions and DB Time

Active sessions

ASH sample count is value of active sessions function at sample times

DB time is area under curve

$\Delta t = 1 \text{ sec}$
The calculus of DB time

• The number of active sessions at any time is the rate of change of the DB time function at that time.

\[
\frac{\delta DBtime}{\delta t} = ActiveSessions
\]

• DB time is the integral of the Active Session function.

\[
DBtime = \int_{t_0}^{t_1} ActiveSessions
\]
ASH Timing for Nano-Operations

• Some important operations are still too frequent and short-lived for timing
  • No “bind” wait event
• A session-level bit vector is updated in binary fashion before/after an operation
  • Cheaper than timer call
• The bit vector is sampled into ASH
• ASH math allows us to estimate time spent in these un-timed transient operations
Techniques:

The DB Time Method
Where is DB Time used?

- ADDM
- EM Performance page and drill downs
- ASH report
- AWR and AWR compare periods reports
- SYSMETRICS and Server-generated Alerts
The DB Time Method: Short Course

or

just ask ADDM
The DB Time Method: Process

1. Identify performance issue
2. Scope the issue
3. Set goals
4. Data capture (NO OP)
5. Investigate DB time distribution
   • Identify the largest potential for improvement
6. Modify system to tune for largest gain
7. Evaluate against goals
   • Repeat from step 4 if goals not met

*Performance tuning by removing excess DB time*
Investigate DB Time Distribution

- Identify uneven distributions of DB time (skew)
  - => Largest potential improvement within scope

- System scope:
  - Resource limits – is problem outside the DB?

- Application scope:
  - Service, module, action
  - Resource contention (e.g. latches)
  - SQLID, rowsource

- Session scope:
  - Long running SQL
  - Resource contention (e.g. enqueues)
DB time

Detail for Selected 5 Minute Interval
Start Time Oct 31, 2007 3:29:12 PM CDT

Top SQL
- SQL ID: 8vn5tn71d4a
  - SQL Type: SELECT
  - Activity (%): 12.59

- SQL ID: 6p15y6k8zwuh5
  - SQL Type: SELECT
  - Activity (%): 10.75

- SQL ID: 076tygk66k8ha
  - SQL Type: SELECT
  - Activity (%): 10.66

- SQL ID: 84k3v24910b
  - SQL Type: SELECT
  - Activity (%): 7.30

- SQL ID: 22183tatccs8d
  - SQL Type: SELECT
  - Activity (%): 4.95

- SQL ID: 93sgg7wmg35xy
  - SQL Type: SELECT
  - Activity (%): 4.95

- SQL ID: 5gvmnr6d21bb
  - SQL Type: SELECT
  - Activity (%): 4.20

- SQL ID: 9rauu0kprxwuf
  - SQL Type: SELECT
  - Activity (%): 3.86

Top Sessions
- Activity (%): 9.54
  - Session ID: 1306
  - User Name: AQLREP
  - Program: perl@atgebs.us.oracle.com (TNS V1-V3)

- Activity (%): 9.32
  - Session ID: 1728
  - User Name: BGRAEF
  - Program: oracledb92@iasbde.us.oracle.com (TNS V1-V3)

- Activity (%): 6.38
  - Session ID: 2251
  - User Name: ARUDAS
  - Program: JDBC Thin Client

- Activity (%): 5.72
  - Session ID: 1682
  - User Name: BUGPATCH
  - Program: oracle@staip12 (TNS V1-V3)

- Activity (%): 5.65
  - Session ID: 1570
  - User Name: MFGOPSTM
  - Program: ? @ap615utl (TNS V1-V3)

- Activity (%): 5.36
  - Session ID: 2047
  - User Name: MFGOPSTM
  - Program: ? @ap615utl (TNS V1-V3)

- Activity (%): 4.62
  - Session ID: 1695
  - User Name: TOGEORGE
  - Program: OMS

- Activity (%): 3.96
  - Session ID: 1935
  - User Name: JSARICOS
  - Program: OMS
Modify System

- Start with the largest DB time issues first
  - Address root causes, not symptoms

- Match solution scope to problem scope
  - Don’t tweak optimizer parameters before tuning SQL

- Proceed iteratively one fix at a time
  - Concurrent fixes should be orthogonal

- Measure and validate results at each successive step

- Stop when goals are met
The DB Time Method: Advantages

• Tunes the one thing that affects users: Time

• Data capture scoping not necessary
  • ‘Always on’ data collection
  • No requirement to reproduce problem

• Works for concurrency problems such as locking

• Combines best of current methods
  • Less intrusive, more inclusive
Method Summary

• DB time is the fundamental performance metric

• The method allows DB time analysis at many scopes
  • Properly scoped problems and solutions are critical to success

• DB time based diagnosis removes value judgments
  • Scientific method, not sorcerer’s magic

• Performance improvement means doing the same work in less DB Time
Tools:

ADDM
Enterprise Manager
Active Reports
Tools for Applying DB Time Method

Two use-cases, one method:

1. Tuning steady-state performance
   - Improve overall workload throughput or response time
   - Best practice: use ADDM

2. Diagnosing transient performance problems
   - Confirm and investigate reported performance issues
   - Best practice: use EM real-time screens
Best Practice: Use ADDM

• Embedded expert system using the DB time method
  • Identifies root causes behind the symptoms

• Variably scoped:
  • Host to instance to SQL and even database block
  • Scoped to database for RAC (new in 11g)

• Findings prioritized by impact on DB time
  • Finding history allows flexible time scoping
  • Directives can filter findings

• Recommendations by benefit (reduction) to DB time
Automatic Database Diagnostic Monitor (ADDM)

Database Activity

The icon selected below the graph identifies the ADDM analysis period. Click on a different icon to select a different analysis period.

ADDM Performance Analysis

Task Name  ADDM:3132078998_1_1978

Task Owner  SYS  Average Active Sessions  10.2  Period Start Time  Apr 4, 2008 4:00:31 AM PDT  Period Duration (minutes)  60  Instance  emtarget_emtarget1

<table>
<thead>
<tr>
<th>Impact (%)</th>
<th>Finding</th>
<th>Occurrences (latest 24 hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.9</td>
<td>Top SQL by DB Time</td>
<td>24 of 24</td>
</tr>
<tr>
<td>39.5</td>
<td>Top SQL By I/O</td>
<td>0 of 24</td>
</tr>
<tr>
<td>36.3</td>
<td>Top Segments by I/O</td>
<td>1 of 24</td>
</tr>
<tr>
<td>12.9</td>
<td>Commits and Rollbacks</td>
<td>23 of 24</td>
</tr>
<tr>
<td>7.6</td>
<td>I/O Throughput</td>
<td>1 of 24</td>
</tr>
<tr>
<td>2.5</td>
<td>Undersized RO</td>
<td>0 of 24</td>
</tr>
</tbody>
</table>
Performance Finding Details: Top SQL by DB Time

Finding History

SQL statements consuming significant database time were found.

Impact (Active Sessions): 4.03
Impact (%): 52.8
Period Start Time: Apr 4, 2008 12:00:04 PM PDT
Period Duration (minutes): 60.2
Filtered: No

Recommendations

Schedule SQL Tuning Advisor

Select All | Select None | Show All Details | Hide All Details

Select Details Category

- Hide SQL Tuning

Action Investigate the SQL statement with SQL_ID "66n44vwsmyknr" for possible performance improvements.
SQL Text:
```
select */ serial_guys */ p_brand, p_type, p_size, ...
```
SQL ID: 66n44vwsmyknr
Benefit (%): 15

Rationale: SQL statement with SQL_ID "66n44vwsmyknr" was executed 4 times and had an average elapsed time of 1031 seconds.

View Tuning History

Run SQL Tuning Advisor on the SQL statement with SQL_ID "4scj37xz190kp".
SQL Text:
```
select */ big_guys */ */ NO_GBY_PUSHDOWN */ s_name, s_address ...
```
SQL ID: 4scj37xz190kp
Benefit (%): 13.3

View Tuning History

Run Advisor Now | Filters

- Show SQL Tuning
- Show SQL Tuning
- Show SQL Tuning

Benefit (%): 10.2, 8, 6.6

Findings Path

Expand All | Collapse All
Finding History: Top SQL by DB Time

View: Apr 3, 2008

Drag the shaded box to change the time period for the detail section below.

Detail for Selected 3 Hour Interval

<table>
<thead>
<tr>
<th>Details</th>
<th>Finding Details</th>
<th>Impact (Active Sessions)</th>
<th>Start Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show</td>
<td>ADDM:3132078998_1_1986</td>
<td>4.03</td>
<td>Apr 4, 2008 12:00:04 PM PDT</td>
</tr>
<tr>
<td>Show</td>
<td>ADDM:3132078998_1_1987</td>
<td>4.55</td>
<td>Apr 4, 2008 1:00:18 PM PDT</td>
</tr>
<tr>
<td>Hide</td>
<td>ADDM:3132078998_1_1988</td>
<td>6.24</td>
<td>Apr 4, 2008 2:00:45 PM PDT</td>
</tr>
</tbody>
</table>

Action:
- Investigate the SQL statement with SQL_ID "2a6s3wn0nu91w" for possible performance improvements.
  SQL Text: select /* big_guys */ /* pq_distribute(supplier none partition) pq_map(supplie...
  SQL ID: 2a6s3wn0nu91w
- Investigate the SQL statement with SQL_ID "1pzsfgba2jm8" for possible performance improvements.
  SQL Text: select /* big_guys */ o_year, sum(case when nation='BRAZIL' then volume...  
  SQL ID: 1pzsfgba2jm8
- Investigate the SQL statement with SQL_ID "dt7umutdm8p67" for possible performance improvements.
  SQL Text: select /* big_guys */ supp_nation, cust_nation, year, ...  
  SQL ID: dt7umutdm8p67
- Investigate the SQL statement with SQL_ID "9sqv60uk9hjzw" for possible performance improvements.
  SQL Text: select /* big_guys */ o_orderpriority, count(*) as order_count from ...  
  SQL ID: 9sqv60uk9hjzw
- Investigate the SQL statement with SQL_ID "66n4fvsmyknw" for possible performance improvements.
  SQL Text: select /* big_guys */ /* pq_distribute(supplier none partition) pq_map(supplie...  
  SQL ID: 66n4fvsmyknw
Best Practice: EM Real-time Interface

• Transient (sub-hour) or immediate time scope
  • Requires interactivity of UI

• ‘Click on the big stuff’
  • Data visualizations display skew directly

• Takes some expertise to separate symptoms from root causes
SELECT */+ OPAQUE_TRANSFORM */
"RPTNO","RPTDATE","RPTD_BY","VERSION","UTILITY_VERSION","CATEGORY","STATUS","SUBJECT","UPD_BY","CUSTOMER"
FROM "BG"."RPTHEAD" "H" WHERE "RPTDATE">:1 AND "RPTD_BY">>'BATCH' AND "CUSTOMER" LIKE '%WPTG%' AND

Details
Select the plan hash value to see the details below. Plan Hash Value 301316116

Summary
Drag the shaded box to change the time period for the detail section below.

Detail for Selected 5 Minute Interval
Start Time Apr 5, 2008 10:29:32 AM

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SID</th>
<th>User</th>
<th>Program</th>
<th>Service</th>
<th>Plan Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2228</td>
<td>MOCONNEL</td>
<td>oracle@mlnxie01 (TNS V1-V3)</td>
<td>boracle.com</td>
<td>301316116</td>
</tr>
<tr>
<td></td>
<td>2203</td>
<td>MOCONNEL</td>
<td>oracle@moconnel-lnx (TNS V1-V3)</td>
<td>boracle.com</td>
<td>301316116</td>
</tr>
</tbody>
</table>
```sql
SELECT /*+ OPAQUE_TRANSFORM */ "RPTNO", "RPTDATE", "RPTD_BY", "VERSION", "UTILITY_VERSION", "CATEGORY", "STATUS", "SUBJECT", "UPD_BY", "CUSTOMER" FROM "BG"."RPTHEAD" WHERE RPTDATE>:1 AND RPTD_BY<>'BATCH' AND "CUSTOMER" LIKE '%WPTG%' AND "CUSTOMER" NOT LIKE '%BO%' AND "CUSTOMER" NOT LIKE '%BTH%' AND "CUSTOMER" NOT LIKE '%BKG%'
```
<table>
<thead>
<tr>
<th>Status</th>
<th>Duration</th>
<th>SQL ID</th>
<th>User</th>
<th>Parallel</th>
<th>Database Time</th>
<th>IO Requests</th>
<th>Start</th>
<th>Ended</th>
<th>SQL Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>02xkn5u096e</td>
<td>FUSION</td>
<td></td>
<td>4.0h</td>
<td>23K</td>
<td>11:57:13 AM</td>
<td></td>
<td>WITH SAWITHD AS (select T9407234.C247.165484 as c1, T9407234.C247.165484 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a3pmr621apn</td>
<td>FUSION</td>
<td></td>
<td>6.2h</td>
<td>23K</td>
<td>10:04:05 AM</td>
<td></td>
<td>WITH SAWITHD AS (select T7850441.C160572348 as c1, T7850441.C160572348 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lgid1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>7.0h</td>
<td>23K</td>
<td>9:17:27 AM</td>
<td></td>
<td>WITH SAWITHD AS (select T18813.C165951942 as c1, T18813.C165951942 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lgid1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>6.7h</td>
<td>23K</td>
<td>7:34:00 AM</td>
<td></td>
<td>WITH SAWITHD AS (select T18813.C165951942 as c1, T18813.C165951942 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lgid1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>10.6h</td>
<td>24K</td>
<td>6:04:26 AM</td>
<td></td>
<td>DECLARE job BINARY_INTEGER := job; next_data TIMESTM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c20s74v769k1</td>
<td>SQLDBP</td>
<td></td>
<td>6:1m</td>
<td>14</td>
<td>Sat Nov 13, 2010 1</td>
<td></td>
<td>WITH SAWITHD AS (select T18813.C165951942 as c1, T18813.C165951942 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lgid1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>41.8h</td>
<td>14</td>
<td>Sat Nov 13, 2010 1</td>
<td></td>
<td>WITH SAWITHD AS (select T18813.C165951942 as c1, T18813.C165951942 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lgid1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>52.8h</td>
<td>819</td>
<td>Sat Nov 13, 2010 1</td>
<td></td>
<td>WITH SAWITHD AS (select T18813.C165951942 as c1, T18813.C165951942 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d2k68k3u306</td>
<td>FUSION</td>
<td></td>
<td>74.8h</td>
<td>22K</td>
<td>Fri Nov 12, 2010 1</td>
<td></td>
<td>WITH SAWITHD AS (select T031268.C247.165484 as c1, T031268.C247.165484 as c2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>11.4s</td>
<td>11.4s</td>
<td>4:13:47 PM</td>
<td>4:14:02 PM</td>
<td>begin := est_notify_wait_for_event_worker_fileid:= 2, p_instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>9.0s</td>
<td>11.4s</td>
<td>4:13:33 PM</td>
<td>4:13:47 PM</td>
<td>begin := est_notify_wait_for_event_worker_fileid:= 2, p_instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>32.0s</td>
<td>7.099</td>
<td>4:11:51 PM</td>
<td>4:12:23 PM</td>
<td>begin := est_notify_wait_for_event_worker_fileid:= 2, p_instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>13.3s</td>
<td>13.3s</td>
<td>4:11:01 PM</td>
<td>4:11:56 PM</td>
<td>begin := est_notify_wait_for_event_worker_fileid:= 2, p_instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>34.0s</td>
<td>34.0s</td>
<td>4:11:03 PM</td>
<td>4:11:17 PM</td>
<td>begin := est_notify_wait_for_event_worker_fileid:= 2, p_instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>14.0s</td>
<td>14.0s</td>
<td>3:50:47 PM</td>
<td>3:50:48 PM</td>
<td>create unique index FUSION_01_STAGE ID_342010_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>14.0s</td>
<td>14.0s</td>
<td>3:56:34 PM</td>
<td>3:56:44 PM</td>
<td>create unique index FUSION_01_STAGE ID_342010_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>14.0s</td>
<td>14.0s</td>
<td>3:54:43 PM</td>
<td>3:54:44 PM</td>
<td>create unique index FUSION_01_STAGE ID_342010_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6z1aq8b800mc</td>
<td>FUSIONORA</td>
<td></td>
<td>14.0s</td>
<td>14.0s</td>
<td>3:54:43 PM</td>
<td>3:54:44 PM</td>
<td>create unique index FUSION_01_STAGE ID_342010_10</td>
</tr>
</tbody>
</table>

Mon Nov 15, 2010 3:54:43 PM
<table>
<thead>
<tr>
<th>TABLE ACCESS BY INDEX ROWID</th>
<th>FND_TREE_VERSION</th>
<th>1</th>
<th>2</th>
<th>52K</th>
<th>54K</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX RA...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>1</td>
<td>52K</td>
<td>69K</td>
</tr>
<tr>
<td>INDEX RANGE...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>54K</td>
<td>212K</td>
</tr>
<tr>
<td>INDEX RANGE...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>212K</td>
<td>522K</td>
</tr>
<tr>
<td>TABLE ACCESS B...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>52K</td>
<td>643K</td>
</tr>
<tr>
<td>VIEW</td>
<td>index1_jmd_036</td>
<td>1</td>
<td>3</td>
<td>514K</td>
<td>3,549</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>151K</td>
</tr>
<tr>
<td>INDEX FAST FU...</td>
<td>HZ_CUST_ACCOUNTS</td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>1,313</td>
</tr>
<tr>
<td>INDEX FAST FU...</td>
<td>HZ_CUST_ACCOUNTS</td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>1,560</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>39</td>
<td>1,731</td>
<td>4,293M</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>36</td>
<td>4,631K</td>
<td></td>
</tr>
<tr>
<td>NESTED LOGO...</td>
<td></td>
<td>1</td>
<td>30</td>
<td></td>
<td>3,731</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>10</td>
<td>901KB</td>
<td></td>
</tr>
<tr>
<td>NESTED...</td>
<td></td>
<td>1</td>
<td>10</td>
<td>1,414</td>
<td></td>
</tr>
<tr>
<td>TABLE...</td>
<td>FND_TREE_VERSION</td>
<td>19</td>
<td>18</td>
<td>1,414</td>
<td>1,414</td>
</tr>
<tr>
<td>INDEX A...</td>
<td>FND_TREE_UI</td>
<td>1</td>
<td>1</td>
<td>1,414</td>
<td>1,414</td>
</tr>
<tr>
<td>TABLE A...</td>
<td>FND_TREE_STRUCTURE</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TABLE AC...</td>
<td>AR_PAY_REL_ASSIGN</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>1</td>
<td>64</td>
<td>3,732</td>
</tr>
<tr>
<td>INDEX RA...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>64</td>
<td>3,732</td>
</tr>
<tr>
<td>TABLE ACCESS...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>3,732</td>
<td>4,631K</td>
</tr>
<tr>
<td>INDEX RANGE...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>3,732</td>
<td>4,631K</td>
</tr>
<tr>
<td>INDEX RANGE...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>3,732</td>
<td>4,631K</td>
</tr>
<tr>
<td>TABLE ACCESS B...</td>
<td>FND_TREE_VERSION</td>
<td>1</td>
<td>2</td>
<td>3,732</td>
<td>4,631K</td>
</tr>
<tr>
<td>VIEW</td>
<td>index1_jmd_036</td>
<td>1</td>
<td>3</td>
<td>514K</td>
<td>3,549</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>151K</td>
</tr>
<tr>
<td>INDEX FAST FU...</td>
<td>HZ_CUST_ACCOUNTS</td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>1,313</td>
</tr>
<tr>
<td>INDEX FAST FU...</td>
<td>HZ_CUST_ACCOUNTS</td>
<td>1</td>
<td>1</td>
<td>514K</td>
<td>1,560</td>
</tr>
<tr>
<td>FAST DUAL</td>
<td></td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>INDEX RANGE SCAN</td>
<td>AR_PAYMENT_SCHEDU</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>INDEX UNIQUE SCAN</td>
<td>AR_CONS_INV_ALL</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RA_CUST_TRN_TYPES</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
ORACLE IS THE INFORMATION COMPANY