References

- Database Tuning: Principles, Experiments, and Troubleshooting Techniques
  - Authors: Dennis Shasha & Philippe Bonnet
  - Comment: Excellent text. Covers Oracle, DB2 and SQL Server

- Oracle Manuals. Available online at OTN
  - Oracle Performance planning
  - Short manual with check-lists of common problems
  - Oracle Database Performance Guide & Reference
    - Discusses Optimizer, Processes, Memory structures, data collection, OS issues, Wait Model
    - Very detailed

References: Oracle Specific Books

- Expert One-on-One Oracle
  - Author: Thomas Kyte
    - Covers all aspects on application tuning. Very readable
    - Very, very detailed

- Oracle Performance Tuning 101
  - Author: Gaja Vaidyanatha
    - Clearly describes a methodological approach to tuning the Oracle application, instance and OS
    - Many chapters available free electronically
    - veritas.com eBooks

- Oracle SQL High-Performance Tuning
  - Author: Guy Harrison
    - Focus on SQL statement tuning
Types of Tuning

- Tune the Application
  - Parsing (Oracle Specific)
  - Transactions (Oracle Specific)
- Tune the System
  - OS
  - Instance

Query Parsing, Optimization, and Execution

Oracle: SQL Processing Steps

- Parse
- Syntax check
- Semantic check
- View processing
- Subquery processing
- Predicate pushing
  - Into the view query block
- Constant folding
- Optimization
- Query Plan Generation
- Query Plan Execution
Should Every Query Go Through These Steps?

- **Scenarios:**
  - A User re-executes the same query
  - A User re-executes very similar queries
    - Show me the courses I took in Fall 93.
    - Show me the courses I took in Winter 93.
  - Multiple Users execute the same query
    - Show courses being offered in Winter 04 semester.

- Can some/most steps be avoided in such cases?

  - What’s needed is something for Plans similar to the caching of data blocks in buffer cache to avoid disk accesses.

---

**Memory Structures for SQL**

- **Shared Pool**
- **Block buffer**
- **Redo log buffer**

![Diagram of Memory Structures for SQL]

**Important point:** Only one Shared SQL Area for a “unique” statement. Created the first time the stmt is encountered.

---

What Constitutes a “Unique” Stmt?

- **Query 1 to 10:** For different literal values in the predicate, re-execute:
  ```sql
  select sname from sailors
  where sid = 1;
  ```
- **Query 11-20:** For different values substituted for the variable `my_var` re-execute:
  ```sql
  select sname from sailors
  where sid = :my_var;
  ```
- **Same SQL statements?**
  - Query 1 through 10 are different statements. 10 different Shared SQL Areas! Parsing, plan generation done 10 times.
  - Query 11-20 map to a unique statement. Same “Shared SQL Area.” No multiple parsing, plan generation, etc.
**Types of Parses**

- Oracle looks up the Shared Pool (in SGA) to determine if the statement has already been encountered:
  - A) Take a hash of the statement and check for a match
  - B) If match found, compare complete text of the two statements
- **Hard/Medium Parse**
  - Lookup failed
  - Carry out all the SQL processing steps
  - Allocate Shared SQL Area for this SQL statement
- **Soft Parse**
  - Lookup succeeded
  - Use already allocated Shared SQL Area
  - Substitute variable values (stored in the Private SQL Area) into the existing Plan (from the Shared SQL Area) and execute it

**Using (Bind) Variables**

- Critical to use these
  - Particularly in OLTP workloads
- Otherwise lots of parsing
  - CPU overhead
  - Contention within Oracle shared pool data structure
- When not to use bind variables?
  - Want specific plan for a particular value
    - E.g., want to use histograms if literal has a specific value.
  - Bind variables are not that important in DSS workloads.

**Best Scenario: No Parse**

- If the statement is going to be re-executed multiple times, don’t even de-allocate the Private SQL Area
  - Don’t close the “Cursor”
  - A cursor is a handle or name for a Private SQL Area
- E.g., when using JDBC
  - You instantiate a ‘Statement’ object
  - Don’t close if the query is going to be re-executed
    - Just bind the new variables (e.g., “Fall 03”, “Winter 04”) and re-execute
    - “Cursor”/Private SQL Area gets saved and is re-used
  - Doesn’t require even a soft parse
- Note: Cursors take up memory. So, do close if query is not going to be re-executed!
Relevant Performance Views

- **V$SQLAREA**
  - Statistics on shared SQL area, i.e., on SQL stmts that are in memory, parsed and ready for execution.
  - Contains one row per SQL string

- **V$LIBRARYCACHE**
  - Statistics on ‘Library cache’
    - ‘Library cache’ is an area in the ‘Shared Pool’
    - ‘Shared SQL Area’ is a component of ‘Library Cache’

Types of Tuning

- **Tune the Application**
  - Parsing (Oracle Specific)
  - Transactions (Oracle Specific)

- **Tune the System**
  - OS
  - Instance
**Read Consistency**

- Oracle uses multi-version concurrency control
- Oracle also provides read-consistency
  - Query result is consistent with respect to a point in time
- Stmt level read consistency:
  - Query result is consistent with respect to time query started
  - Provided with READ COMMITTED isolation level
- Transaction level read consistency
  - All queries in the txn get results consistent with respect to time txn started (somewhat like REPEATABLE READ isolation level)
  - Provided with READ ONLY transaction mode
- For multi-version concurrency control
  - The "old" copy of the data needs to be retained
  - UNDO records

**Locking in Oracle**

- Recall no read locks
- Very rarely will the User need to explicitly request a lock
- Main types of locks that User transactions implicitly get:
  - TX and TM
- TX locks:
  - One lock implicitly given to an update transaction
- TM locks:
  - Also called DML locks
  - To ensure that the table is not modified while data is being inserted/deleted/modified rows
  - One lock implicitly given for each table into which the txn is inserting/deleting/modifying rows

**Oracle Locking Example 1**

- T1 starts: **INSERT INTO SAILORS**
  - Oracle will grant
    - A (shared) TM lock on the table SAILORS
    - A TX lock. This lock is associated with the UNDO segment
    - If T1 now inserts 20 more rows, NO more locks are needed
    - T1 has not committed
- T2 starts: **INSERT INTO SAILORS**
  - T2 also gets on TX lock and a (shared) TM lock on table SAILORS
  - Inserts the row successfully
  - T2 commits. Gives up its locks
- T3 starts: **DROP TABLE SAILORS**
  - DROP requires an exclusive lock on table SAILORS
  - T3's action will be rolled back
    - with an Oracle error (ORA-00054)
Oracle Locking Example 2

**T1:** UPDATE SAILORS SET RATING = 10
WHERE SID = 3
Oracle will grant
- A (shared) TM lock on the table SAILORS
- A TX lock. This lock is associated with the UNDO segment for this transaction.
- T1 has not committed

**T2:** UPDATE SAILORS SET RATING = 11
WHERE SID = 3
Oracle will grant
- A (shared) TM lock on the table SAILORS
- T2 will block on a TX lock. T2 will remain blocked until T1 gives up the lock on the row.

Locking Implementation in Oracle

- No data structure that has information on all the locks
- Row level locking information stored in the data block where the updated row exists
- To lock a row, Oracle does the following:
  1. Find the address of the row to be locked
  2. Go to the row
  3. Lock it. If it is already locked, wait!

Should Explicit Lock Request Ever be Made?

- Scenario:
  - Program queries a table and shows data to User, who can then update the data.
  - T1 reads row X. Shows it to User 1.
  - T2 reads row X. Show it to User 2.
  - User 1 decides to update row X. T1 updates row X. T2 commits.
  - User 2 decides to update row X. T2 updates row X. T2 commits.

- Possible solution: Pessimistic approach.
  - Try to lock row before showing to a User.
    - SELECT * FROM sailors WHERE sid = 3 FOR UPDATE NOWAIT
Relevant Performance Views

- **V$LOCK**
  - Lists locks requested as well as granted
  - Can be used to find which Session is blocking which other session

Types of Tuning

- Tune the Application
- Tune the System
  - OS
  - Oracle Instance

Tuning vs. System Configuration

- Configuring a system
  - Allocating resources in an ordered manner to make system configuration functional
- Performance tuning requires a different method
- Bottleneck:
  - Resource in the system that is being over-used
- Key performance tuning idea:
  - Identify the most significant bottlenecks
  - Break the bottlenecks
    - By changes in application, Oracle, host hardware


**General Comments**

- Tuning should be part of the entire life-cycle
- Generally done reactively
  - Sometimes in panic-mode
    - E.g., application has become so slow that users are complaining
- Goal of the lecture:
  - To highlight important points
  - For more details, look up the references

**Resources & Consumers in the System**

- Machine and Operating System
  - CPU
  - Disk
  - Memory
    - Primary resources
- Database sub-systems
  - Processes
  - Files
    - Memory structures
    - Intermediate resources as well as consumers
- User processes
  - Execute SQL online or batch
    - High level consumers

**Use Symptoms to Identify Causes**

- Need to identify the resources or consumers that are effects or causes of the performance problem
  - What might seem like the cause may actually be the effect of a problem caused by something else
- Problems with host?
  - Enough resources for the desired performance goal?
    - Configured correctly given the expected load?
- Problems with Oracle Instance?
  - Are DBMS resources configured correctly?
- Problems with Application?
  - Are applications and queries making good use of DBMS resources?
- To identify and solve any of these problems, important to capture relevant data
### Host: CPU
- Just adding more processing power will **not** help the performance if the system is not CPU bound!

- What to measure?
  - Run queue length
  - Greater than 2 per CPU means CPU is a bottleneck
  - CPU used by DBMS vs. other processes in non-dedicated env.
  - May be the problem is caused by other non-DBMS applications running on the same machine
  - CPU time spent in Kernel mode vs. User mode
    - Is the OS spending a lot of time on its internal work?

- How to measure?
  - OS dependent
    - Unix: sar, top, vmstat

- What to do?
  - Problem could be in application: e.g., too much parsing

### Host: Disk
- What to measure?
  - Current response time
  - Average size of waiting queue
  - Compare waits from each disk
    - Will indicate I/O is not balanced across disks

- How to measure?
  - OS dependent
    - Unix: iostat

- What to do?
  - Move data if problem is with I/O resource
  - Problem could be in application: e.g., poorly tuned SQL,…
  - Problem could be in DBMS sub-systems: e.g., buffer cache too small

### Host: Memory
- What to measure?
  - Number of page faults per second
    - Ideally no paging
    - Certainly ensure no thrashing

- How to measure?
  - OS dependent:
    - Unix: vmstat
Database: Where is the Resource Problem?

- A methodological way to find problems in DBMS sub-system is to use **Wait Model**
- Oracle can keep statistics on what is going on for each session in the DBMS
  - Is the session on CPU?
  - Is the session *waiting* for some resource?
    - Is session waiting for I/O to complete?
    - Is session waiting for a lock held by another session?
    - Is session waiting for a lock (latch) in a data structure in SGA?
    - ...

Use Wait Statistics to Identify Bottlenecks

- Statistics on time spent waiting on different wait events
  - **V$SYSTEM_EVENT**
- This information (and *supporting* information) can be used to deduce what is causing the problem
  - Waits for I/O? -> I/O problem
  - Waits for locks? -> Concurrency problem
  - Waits for free buffers? -> DBWR/buffer cache problem

V$_EVENTS

```sql
SQL> desc v$system_event;
EVENT         VARCHAR2(64)  TOTAL_WAITS      NUMBER  TIME_WAITED      NUMBER

SQL> desc v$session_event;
SID            NUMBER    EVENT         VARCHAR2(64)    TOTAL_WAITS      NUMBER  TIME_WAITED      NUMBER
```
**Fixing the Application**

- Again the Wait model helps identify symptoms
- Fix may be to tune the Oracle Instance
  - In-adequately size buffer cache
  - In-adequately sized shared pool (recall SQL Area)
  - Need more DBWR processes
- But fix in most cases will be “Tune the Application!”
  - Un-tuned SQL
  - Missing indexes
  - Unnecessary locking
  - SQL does not use variables
  - Missing or stale Optimizer stats
  - ...

**Finding out Offending SQL & Sessions**

- If problem is indeed in the Application, we still need to find out the SQL statements and/or the Sessions whose resource consumption is causing the problem
- Can use snapshots of performance data to identify top resource consuming SQL statements over a period
- Can use V$SESSION to identify top resource consuming sessions

**V$SESSION**

SQL> desc v$session;

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SERIAL#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>USERNAME</td>
<td>VARCHAR2 (30)</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR2 (8)</td>
</tr>
<tr>
<td>OSUSER</td>
<td>VARCHAR2 (30)</td>
</tr>
<tr>
<td>PROCESS</td>
<td>VARCHAR2 (12)</td>
</tr>
<tr>
<td>MACHINE</td>
<td>VARCHAR2 (64)</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>VARCHAR2 (48)</td>
</tr>
<tr>
<td>SQL_ADDRESS</td>
<td>RAW (8)</td>
</tr>
<tr>
<td>SQL_HASH_VALUE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>LOGON_TIME</td>
<td>DATE</td>
</tr>
</tbody>
</table>
Relevant Performance Views

- **V$SYSTEM_EVENT**
  - System-wide wait information
- **V$SESSION_EVENT**
  - Wait information per session
- **V$SESSION_WAIT**
  - Details on Session Wait
- **V$SYSSTAT**
  - System-wide statistics
    - E.g., number of transactions, redo generated, physical I/O...
- **V$SESSTAT**
  - Statistics per session
- **V$SQL**
  - Statistics of SQL statements
    - E.g., number of executions, physical I/O,...