Editor’s Note: You would not drive a car in need of service nor play an out of tune piano? In his first ORAtips article, a two part series, Darrick Addison brings home the point that your biggest Oracle investment, the Oracle database, needs regular care and feeding to run at maximum efficiency and peak performance. Darrick discusses the pros and cons of Oracle’s new approach for 10g tuning, identifying no less than 17 problematic queries, and answers the questions “what do I look for and how do I make it better?”

Introduction

Performance tuning is a broad and complex topic area when it comes to Oracle databases. Two of the biggest questions faced by your average DBA concern where to start and what to do. All you may know is that someone reports a problem about a slow or poor performing application or query. Where do you even begin to start when faced with this situation? Oracle’s emphasis on this particular methodology changed when Oracle10g was released. The approach has gone from top-down in 8i and 9i to that of following principles in 10g.

The performance-tuning guide for Oracle10g (Release 2) identifies the overall process as The Oracle Performance Improvement Method. The steps have been expanded, but overall, remain the same.

1. Perform the following initial standard checks:
   - Get candid feedback from users. Identify the performance project’s scope and subsequent performance goals, as well as goals for the future.
   - Get a full set of operating system, database, and application statistics from the system when the performance is both good and bad.
   - Sanity-check the operating systems of all machines involved with user performance. By sanity-checking the operating system, you look for hardware or operating system resources that are fully utilized.
   - Redo log setup problems
   - Serialization of data blocks in the buffer cache due to lack of free lists, free list groups, transaction slots (INTRAN), or shortage of rollback segments.
   - Long full table scans
   - In disk sorting
   - High amounts of recursive (SYS) SQL
   - Schema errors and Optimizer problems
   - Use of non-standard initialization parameters

2. Check for the top ten most common mistakes with Oracle, and determine if any of these are likely to be the problem. List these symptoms for later analysis.
   - Bad connection management
   - Bad use of cursors and the shared pool
   - Getting database I/O wrong
   - Bad use of transactional resources
   - Serialization of data blocks or redo log operations

3. Build a conceptual model of what is happening on the system, using the symptoms as clues to understand what caused the performance problems.

4. Propose a series of remedy actions and the anticipated response of the system, and then apply them in the order that can benefit the application the most. A golden rule in performance work is that you only change one thing at a time and then measure the differences.

5. Validate that the changes made have had the desired effect, and see if the user’s perception of performance has improved.

6. Repeat the last three steps until performance goals are met or become impossible due to other constraints.
The Change Is Part of the Problem

The change from a top-down structured approach to a principle-based “make it stop hurting” one is part of the problem. Gathering statistics is obviously important, because how would you know if you have improved (or worsened) the problem? To some degree with either approach, you are left with the original two questions: what do I look for, and how do I make it better?

What would help the novice tuner is a list of items or areas to analyze (configure, diagnose, and tune) in each of the following:

- Tuning the Buffer Cache
- Tuning the Redo Log Buffer
- Tuning the Shared Pool Memory
- Tuning the Program Global Area
- Optimizing Data Storage
- Optimizing Table spaces
- Tuning Undo Segments
- Detecting Lock Contention
- Tuning SQL

These areas pretty much cover the Oracle RDBMS and instance from top to bottom.

Listed below are guidelines to assist you in avoiding common mistakes and bad coding.

17 Tips for Avoiding Problematic Queries

These tips provide a solid foundation for two outcomes: making a SQL statement perform better, and determining that nothing else can be done in this regard.

The 17 tips are as follow:

1. Avoid Cartesian products.
2. Avoid full table scans on large tables.
3. Use SQL standards and conventions to reduce parsing.
4. Avoid use of indexes on columns contained in the WHERE clause.
5. Avoid joining too many tables.
6. Monitor V$SESSION_LONGOPS to detect long running operations.
7. Use hints as appropriate.
8. Use the SHARED_CURSOR parameter.
9. Use the Rule-based optimizer if it is better than the Cost-based optimizer.
10. Avoid unnecessary sorting.
11. Monitor index browning (due to deletions; rebuild as necessary).
12. Use compound indexes with care (do not repeat columns).
14. Use different table spaces for tables and indexes (as a general rule: this is old-school somewhat, but the main point is reduce I/O contention).
15. Use table partitioning (and local indexes) when appropriate (partitioning is an extra cost feature).
16. Use literals in the WHERE clause (use bind variables).
17. Keep statistics up to date.

The list overall is thorough and accurate.

Note: Step 9, referring to the use of the Rule-based optimizer, may cause a reliance or dependency on a feature Oracle has identified as a future item to be deprecated.

Using Bind Variables

On any number of DBA help-type Web sites, a frequently seen bit of advice is to use bind variables, but rarely are the steps or instructions for this step included. Figure 1 illustrates a simple method for creating and using a bind variable.

Figures 2 and 3 compare querying for employee ID and name with and without the bind variable (with the output turned off using trace only).

Okay, so the difference is not that great (the cost went from 3 to 2), but this was a small example. The main point of the previous section is using and understanding bind variables. None of these steps is especially difficult to perform or implement. For
Oracle programmers used to using the “tableA.column_name = tableB.column_name” format for joins, moving to the use of natural joins saves quite a bit of typing, plus there is the benefit of having key column names match up. As shown, some measures may not have a big impact, but every little bit helps to improve performance.

**DataFactory**

The purpose of DataFactory is to “quickly create meaningful test data for multiple database platforms.”

**Editor’s Note:** DataFactory is trial freeware from Quest Software, the same folks that created TOAD.

**Creating Tutorial Objects**

An excellent way to get to know the application is to use its tutorial objects, which includes:

- Create a project
- Create tables in a schema
- Run a script to load data

After starting DataFactory, you can start the tutorial. The instructions on how to load and for help are in HTML files.

Prior to creating the tables, you may want to create a separate schema in your database.

```sql
SQL> set autotrace traceonly
SQL> select employee_id, first_name, last_name
2  from employees
3  where department_id = 80;
14 rows selected.

<table>
<thead>
<tr>
<th>Execution Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 SELECT STATEMENT Optimizer=ALL_ROWS (Cost=0 Card=34 Bytes=748)</td>
</tr>
<tr>
<td>1 0 TABLE ACCESS (FULL) OF 'EMPLOYEES' (TABLE) (Cost=0 Card=34 Bytes=748)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 recursive calls</td>
</tr>
<tr>
<td>0 db block gets</td>
</tr>
<tr>
<td>0 consistent gets</td>
</tr>
<tr>
<td>0 physical reads</td>
</tr>
<tr>
<td>0 redo size</td>
</tr>
<tr>
<td>1527 bytes sent via SQL*Net to client</td>
</tr>
<tr>
<td>350 bytes received via SQL*Net from client</td>
</tr>
<tr>
<td>4 SQL*Net roundtrips to/from client</td>
</tr>
<tr>
<td>0 sorts (memory)</td>
</tr>
<tr>
<td>0 sorts (disk)</td>
</tr>
<tr>
<td>34 rows processed</td>
</tr>
</tbody>
</table>
```

*Figure 2: Querying Without a Bind Variable*

```sql
SQL> select employee_id, first_name, last_name
2  from employees
3  where department_id = :department_id;
14 rows selected.

<table>
<thead>
<tr>
<th>Execution Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2 Card=10 Bytes=2280)</td>
</tr>
<tr>
<td>1 0 TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES' (TABLE) (Cost=2 Card=10 Bytes=2280)</td>
</tr>
<tr>
<td>2 1 INDEX (RANGE SCAN) OF 'EMP_DEPARTMENT_IDX' (INDEX) (Cost=1 Card=10)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 recursive calls</td>
</tr>
<tr>
<td>0 db block gets</td>
</tr>
<tr>
<td>12 consistent gets</td>
</tr>
<tr>
<td>0 physical reads</td>
</tr>
<tr>
<td>0 redo size</td>
</tr>
<tr>
<td>1527 bytes sent via SQL*Net to client</td>
</tr>
<tr>
<td>350 bytes received via SQL*Net from client</td>
</tr>
<tr>
<td>4 SQL*Net roundtrips to/from client</td>
</tr>
<tr>
<td>0 sorts (memory)</td>
</tr>
<tr>
<td>0 sorts (disk)</td>
</tr>
<tr>
<td>34 rows processed</td>
</tr>
</tbody>
</table>
```

*Figure 3: Querying Using a Bind Variable*
To do this, use the navigation path: Tools > Create Tutorial Objects as shown in Figure 4 to execute the following steps.

Step 1: Start the Tutorial Setup Wizard as shown in Figure 5.

Step 2: Once finished, a list of tables appears as in Figure 6.

Step 3: Upon creation, DataFactory will display a success window (Figure 7).

Step 4: The project folder appears in the left frame (Figure 8).

The purpose of DataFactory is to “quickly create meaningful test data for multiple database platforms.”
Step 5: Click the Run button on the main menu. The ORA-02291 integrity constraint violated error will appear (Figure 9) quite a few times (in some tables more than once) because the loaded data in a foreign key-designated column does not correspond with data in a parent table.

To work around the integrity constraint violation, you can disable the constraint. The query and ALTER TABLE statement in Figure 10 show one method to identify and disable the problem constraint.

Instead of analyzing each table one at a time, use the DBMS_STATS built-in (which Oracle recommends using for most analyze operations). If you are using Oracle10g, you may want to add a WHERE dropped='NO' to prevent dropped tables from appearing in queries on user_tables or tab as in Figure 11.

Back in the project hierarchy or list of tables, selecting a table in the list will show its columns and their data types. (Figure 12)
uses it. If you are relying on referential integrity, your test data needs to support and honor parent table-child table relationships.

From a design standpoint, two best practices that were violated include failure to index foreign key columns, and failure to explicitly name three major items (primary keys, foreign keys, and indexes). A possible third violation concerns not having a primary key on every table. Does every table in a schema require a primary key? No, but for the most part, every table should, and if not, you should at least have a reason. In other words, to not normalize a table should be a conscious decision, not an oversight.

Darrick Addison, ASC Technologies, LLC - Darrick works as a Senior Software Engineer/Consultant for his consulting company, ASC Technologies, LLC. He has been designing and developing custom software applications since 1993. He has worked on the design and development of software ranging from database applications, network applications (TCP/IP client/server), GUI applications, and embedded systems applications in various commercial and government environments. He currently holds a BS degree in Computer Science and is pursuing his Master's degree in Computer Science/Telecommunications at Johns Hopkins University. Darrick may be contacted at Darrick.Addison@ERPtips.com.

Resources
Steve Callan, Columnist - Steve is an Oracle DBA (OCP 8i and 9i)/developer working in Denver. His Oracle experience also includes Forms and Reports, Oracle9iAS and Oracle9iDS.

Editor’s Note: Steve Callan’s article “How Much Is That Database In The Window” is featured in this issue’s CIO Corner.