PART 5

Appendixes

APPENDIX A

Downloadable Files

This appendix lists all the files used throughout the book. It also describes the test environment, which is important because some scripts will work correctly only when a specific configuration is in place.

Test Environment

My test server is a Dell PowerEdge 1900, equipped with a quad-core Xeon processor (E5320, 1.86GHz), 4GB of memory, two mirrored SATA disks (Samsung Spinpoint, 300GB, 7,200rpm) for the operating system and all other applications, and four striped SAS disks (Seagate Cheetah, 73GB, 15,000rpm) for the database files. The server is connected to my workstation and to the other test clients through a gigabit network and switch.

The operating system is CentOS[1] 4.4 x86_64. The following versions of the Oracle database engine are installed (actually, these are all the versions that are currently available for this platform):

- *Oracle9*i *Release 2*: 9.2.0.4, 9.2.0.6, 9.2.0.7, 9.2.0.8
- *Oracle Database 10*g *Release 1*: 10.1.0.3, 10.1.0.4, 10.1.0.5
- *Oracle Database 10*g *Release 2*: 10.2.0.1, 10.2.0.2, 10.2.0.3, 10.2.0.4
- *Oracle Database 11*g *Release 1*: 11.1.0.6

For each version, I installed two databases: Enterprise Edition without options and Enterprise Edition with all options.

Files Available for Download

Figure A-1 shows the structure of the distribution file you can download from `http://top.antognini.ch`. For each chapter (except Chapter 1), there is a directory containing the files related to it. In addition, for each database, there is a directory containing the files I used to build it.

1. See `http://www.centos.org` for additional information.

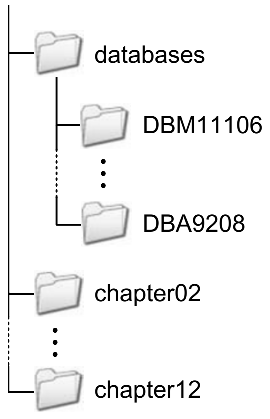

Figure A-1. *The directory structure of the distribution file*

Databases

Scripts were generated using the Database Configuration Assistant. For all databases, only the initialization parameters related to the name of the database, the location of the files, and the memory utilization are set. All other initialization parameters were left at the default value. The only exception is the initialization parameter `remote_os_authent`, which is set to `TRUE`. Note that you should never use this value for a database, except in a "playground" (a system that has no value and contains no data of any importance whatsoever).

The names of the databases define the type of the installation. For example, the names of the two databases shown in Figure A-1 mean the following:

- `DBM11106`: Database without options (`M` stands for "minimal") of version 11.1.0.6
- `DBA9208`: Database with all options (`A` stands for "all") of version 9.2.0.8

Every script has been tested against every database. If a specific script doesn't work in one of them, the header of the script indicates when this is the case.

Chapter 2

The files listed in Table A-1 are available for download for Chapter 2.

Table A-1. *Files for Chapter 2*

File Name	Description
`bind_variables.sql`	This script shows how and when bind variables lead to the sharing of cursors.
`bind_variables_peeking.sql`	This script shows the pros and cons of bind variable peeking.
`selectivity.sql`	This scripts provides the examples shown in the section "Defining Selectivity."
`sharable_cursors.sql`	This script shows examples of parent and child cursors that cannot be shared.

Chapter 3

The files listed in Table A-2 are available for download for Chapter 3.

Table A-2. *Files for Chapter 3*

File Name	Description
`DBM11106_ora_6334.trc`	This is the sample trace file used as the basis for explaining the TKPROF and TVD$XTAT output.
`DBM11106_ora_6334.txt`	This is the TKPROF output of the trace file `DBM11106_ora_9813.trc` used as the basis for explaining the format of the file generated by TKPROF.
`DBM11106_ora_6334.html`	This is the TVD$XTAT output of the trace file `DBM11106_ora_9813.trc` used as the basis for explaining the format of the file generated by TVD$XTAT.
`dbm10203_ora_24433.trc`	This is the sample trace file depicted in Figure 3-19. It shows how information is stored regarding multiple sections generated by a single session.
`dbm10203_s000_24374.trc`	This is the sample trace file shown in Figure 3-19. It shows how information is stored regarding three sessions connected through a shared server.
`dbms_profiler.sql`	This script shows how to profile a PL/SQL procedure and how to display the generated information.
`dbms_profiler_triggers.sql`	You can use this script to create two triggers enabling and disabling the PL/SQL profiler.
`LoggingPerf.java`	You can use this Java class to compare the average execution time of the methods `info` and `isInfoEnabled` of the log4j class `Logger`.
`makefile.mk`	This is the makefile I used to compile the C program given as an example.
`map_session_to_tracefile.sql`	You can use this script to map a session ID to a trace file.
`perfect_triangles.sql`	You can use this script to create the PL/SQL procedure `perfect_triangles` used as an example in the “Gathering the Profiling Data” section.
`session_attributes.c`	This C program shows how to set the client identifier, client information, module name, and action name through OCI.
`SessionAttributes.cs`	This C# class shows how to set the client identifier through ODP.NET.
`SessionAttributes.java`	This Java class shows how to set the client identifier, module name, and action name through JDBC.
`trcsess.awk`	You can use this awk script to make SQL trace files generated with Oracle9*i* compatible with the command-line tool `trcsess`.

Chapter 4

The files listed in Table A-3 are available for download for Chapter 4.

Table A-3. *Files for Chapter 4*

File Name	Description
clustering_factor.sql	This script creates a function that illustrates how the clustering factor is computed.
comparing_object_statistics.sql	This script shows how to compare current object statistics with object statistics stored in the history that are pending and stored in a backup table.
cpu_cost_column_access.sql	This script shows the CPU cost estimated by the query optimizer when accessing a column, depending on its position in the table.
dbms_stats_job_10g.sql	This script shows the actual configuration of the job aimed at automatically gathering object statistics, which is installed and scheduled during the creation of a 10*g* database.
dbms_stats_job_11g.sql	This script shows the actual configuration of the job aimed at automatically gathering object statistics, which is installed and scheduled during the creation of an 11*g* database.
delete_histogram.sql	This script shows how to delete a single histogram, without modifying the other statistics.
lock_statistics.sql	This script shows the working and behavior of locked object statistics.
mreadtim_lt_sreadtim.sql	This script shows the correction performed by the query optimizer when `mreadtim` is smaller or equal to `sreadtim`.
object_statistics.sql	This script provides an overview of all object statistics.
pending_object_statistics.sql	This script shows how to use pending statistics to test a new set of object statistics before publishing them.
system_stats_history_job.sql	This script can be used to create a table and a job to store the evolution of workload statistics over several days.
system_stats_history.sql	This script is used to extract workload statistics from the history table created by the script `system_stats_history_job.sql`. The output can be imported into the spreadsheet `system_stats_history.xls`.
system_stats_history.xls	This Excel spreadsheet can be used to compute average values and to draw charts showing trends of workload statistics extracted with the script `system_stats_history.sql`.

Chapter 5

The files listed in Table A-4 are available for download for Chapter 5.

Table A-4. *Files for Chapter 5*

File Name	Description
assess_dbfmbrc.sql	This script is used to test the performance of multiblock reads for different values of the initialization parameter db_file_multiblock_read_count. The idea is to determine the value of this parameter that provides optimal performance.
bug5015557.sql	This script shows that the initialization parameter optimizer_features_enable disables bug fixes as well as regular features.
dynamic_sampling_levels.sql	This script shows examples of queries taking advantage of dynamic sampling, for levels going from 1 to 4.
optimizer_index_caching.sql	This script shows the working and drawbacks of the initialization parameter optimizer_index_caching.
optimizer_index_cost_adj.sql	This script shows the drawbacks of setting the initialization parameter optimizer_index_cost_adj.
optimizer_secure_view_merging.sql	This script shows the working and drawbacks of the initialization parameter optimizer_secure_view_merging.

Chapter 6

The files listed in Table A-5 are available for download for Chapter 6.

Table A-5. *Files for Chapter 6*

File Name	Description
dbms_xplan_output.sql	This script generates a sample output containing the main information provided by the functions of dbms_xplan.
display.sql	This script shows examples of how to use the function display in the package dbms_xplan.
display_awr.sql	This script shows examples of how to use the function display_awr in the package dbms_xplan.
display_cursor.sql	This script shows examples of how to use the function display_cursor in the package dbms_xplan.
display_cursor_9i.sql	This script displays the execution plan of a cursor stored in the library cache. The cursor is identified by address, hash value, and child number.
execution_plans.sql	This script shows the different types of operations that execution plans are composed of.
parent_vs_child_cursors.sql	This script shows the relationship between a parent cursor and its children's cursors.

Table A-5. *Files for Chapter 6 (Continued)*

File Name	Description
`restriction_not_recognized.sql`	This script generates the output used for showing how to recognize inefficient execution plans, by checking actual cardinalities.
`wrong_estimations.sql`	This script generates the output used for showing how to recognize inefficient execution plans by looking at wrong estimations.

Chapter 7

The files listed in Table A-6 are available for download for Chapter 7.

Table A-6. *Files for Chapter 7*

File Name	Description
`all_rows.sql`	This script shows how it is possible to switch the optimizer mode from `rule` to `all_rows` with a SQL profile.
`baseline_automatic.sql`	This script shows how the query optimizer automatically captures a SQL plan baseline.
`baseline_from_sqlarea1.sql`	This script shows how to manually load a SQL plan baseline from the library cache. The cursor is identified by the text of the SQL statement associated with it.
`baseline_from_sqlarea2.sql`	This script shows how to manually load a SQL plan baseline from the library cache. The cursor is identified by the SQL identifier of the SQL statement associated with it.
`baseline_from_sqlarea3.sql`	This script shows how to tune an application without changing its code. A SQL plan baseline is used for that purpose.
`baseline_from_sqlset.sql`	This script shows how to manually load a SQL plan baseline from a SQL tuning set.
`baseline_upgrade_10g.sql`	This script shows how to create and export a SQL tuning set on Oracle Database 10*g*. It is used along with the script `baseline_upgrade_11g.sql` to show how to stabilize execution plans during an upgrade to Oracle Database 11*g*.
`baseline_upgrade_11g.sql`	This script shows how to import and load a SQL tuning set into a SQL plan baseline. It is used along with the script `baseline_upgrade_10g.sql` to show how to stabilize execution plans during an upgrade from Oracle Database 10*g* to Oracle Database 11*g*.
`clone_baseline.sql`	This script shows how to move SQL plan baselines between two databases.
`clone_sql_profile.sql`	This script shows how to create a copy of a SQL profile.
`depts_wo_emps.sql`	This script was used to generate the execution plans used as examples in the section “Altering the Access Structures.”

Table A-6. *Files for Chapter 7*

File Name	Description
`exec_env_trigger.sql`	This script creates a configuration table and a database trigger to control the execution environment at the session level.
`first_rows.sql`	This script shows how it is possible to switch the optimizer mode from `all_rows` to `first_rows` with a SQL profile.
`object_stats.sql`	This script shows how it is possible to provide object statistics to the query optimizer with a SQL profile.
`opt_estimate.sql`	This script shows how it is possible to enhance the cardinality estimations performed by the query optimizer with a SQL profile.
`outline_editing.sql`	This script shows how to manually edit a stored outline.
`outline_edit_tables.sql`	This script creates the working tables and public synonym necessary to edit private outlines.
`outline_from_sqlarea.sql`	This script shows how to manually create a stored outline by referencing a cursor in the shared pool.
`outline_from_text.sql`	This script shows how to manually create a stored outline as well as how to manage and use it.
`outline_with_ffs.sql`	This script tests whether a stored outline is able to overwrite the setting of the initialization parameter `optimizer_features_enable`.
`outline_with_hj.sql`	This script tests whether a stored outline is able to overwrite the setting of the initialization parameter `hash_join_enabled`.
`outline_with_rewrite.sql`	This script tests whether a stored outline is able to overwrite the setting of the initialization parameter `query_rewrite_enabled`.
`outline_with_star.sql`	This script tests whether a stored outline is able to overwrite the setting of the initialization parameter `star_transformation_enabled`.
`tune_last_statement.sql`	This script is used to instruct the SQL Tuning Advisor to analyze the last SQL statement executed by the current session. When the processing is over, the analysis report is shown.

Chapter 8

The files listed in Table A-7 are available for download for Chapter 8.

Table A-7. *Files for Chapter 8*

File Name	Description
`bind_variables.sql`	This script shows how and when bind variables lead to the sharing of cursors.
`bind_variables_peeking.sql`	This script shows the pros and cons of bind variable peeking.
`lifecycle.sql`	This script shows the difference between implicit and explicit cursor management.
`long_parse.sql`	This script is used to carry out a parse lasting about one second. It also shows how to create a stored outline to avoid such a long parse.
`long_parse.zip`	This compressed archive contains two trace files generated by the execution of the script `long_parse.sql`. For each trace file, the output files generated by TKPROF and TVD$XTAT are available as well.
`ParsingTest1.c`, `ParsingTest2.c`, and `ParsingTest3.c`	These files contain C (OCI) implementations of test case 1, 2, and 3, respectively.
`ParsingTest1.cs` and `ParsingTest2.cs`	These files contain C# (ODP.NET) implementations of test case 1 and 2, respectively.
`ParsingTest1.java`, `ParsingTest2.java`, and `ParsingTest3.java`	These files contain Java implementations of test case 1, 2, and 3, respectively.
`ParsingTest1.sql`, `ParsingTest2.sql`, and `ParsingTest3.sql`	These scripts provide PL/SQL implementations of test case 1, 2, and 3, respectively.
`ParsingTest1.zip`, `ParsingTest2.zip`, and `ParsingTest3.zip`	These compressed archives contain several trace files generated by the execution of the Java implementation of test case 1, 2, and 3, respectively. For each trace file, the output files generated by TKPROF and TVD$XTAT are available as well.

Chapter 9

The files listed in Table A-8 are available for download for Chapter 9.

Table A-8. *Files for Chapter 9*

File Name	Description
`access_structures_1.sql`	This script compares the performance of different access structures in order to read a single row. It was used to generate the figures in Figure 9-3.
`access_structures_1000.sql`	This script compares the performance of different access structures in order to read thousands of rows. It was used to generate the figures in Figure 9-4.
`conditions.sql`	This script shows how you can use B-tree and bitmap indexes to apply several types of conditions.
`fbi.sql`	This script shows an example of a function-based index.
`full_scan_hwm.sql`	This script shows that full table scans read all blocks up to the high watermark.
`index_full_scan.sql`	This script shows examples of full index scans.
`iot_guess.sql`	This script shows the impact of stale guesses on logical reads.
`linguistic_index.sql`	This script shows an example of a linguistic index.
`pruning_composite.sql`	This script shows several examples of partition pruning applied to a composite-partitioned table.
`pruning_hash.sql`	This script shows several examples of partition pruning applied to a hash-partitioned table.
`pruning_list.sql`	This script shows several examples of partition pruning applied to a list-partitioned table.
`pruning_range.sql`	This script shows several examples of partition pruning applied to a range-partitioned table.
`read_consistency.sql`	This script shows how the number of logical reads might change because of read consistency.
`row_prefetching.sql`	This script shows how the number of logical reads might change because of row prefetching.

Chapter 10

The files listed in Table A-9 are available for download for Chapter 10.

Table A-9. *Files for Chapter 10*

File Name	Description
`block_prefetching.sql`	This script shows block prefetching for data and index blocks.
`hash_join.sql`	This script provides several examples of hash joins.
`join_elimination.sql`	This script provides an example of join elimination.
`join_trees.sql`	This script provides an example for each type of join tree.
`join_types.sql`	This script provides an example for each type of join.
`nested_loops_join.sql`	This script provides several examples of nested loop joins.
`merge_join.sql`	This script provides several examples of merge joins.
`outer_join.sql`	This script provides several examples of outer joins.
`outer_to_inner.sql`	This script provides an example of an outer join transformed into an inner join.
`pwj.sql`	This script provides several examples of partition-wise joins.
`pwj_performance.sql`	This script is used to compare the performance of different partition-wise joins. It was used to generate the figures found in Figure 10-15.
`star_transformation.sql`	This script provides several examples of star transformation.
`subquery_unnesting.sql`	This script provides several examples of subquery unnesting.

Chapter 11

The files listed in Table A-10 are available for download for Chapter 11.

Table A-10. *Files for Chapter 11*

File Name	Description
`array_interface.sql`, `array_interface.c`, `ArrayInterface.cs`, and `ArrayInterface.java`	These scripts provide examples of implementing the array interface with PL/SQL, OCI, JDBC, and ODP.NET.
`ArrayInterfacePerf.java`	This script shows that the array interface can greatly improve the response time of a large load. It was used to generate Figure 11-13.
`atomic_refresh.sql`	This script can be used to reproduce bug 3168840 in Oracle9*i*. The bug causes refreshes not to work correctly if a single materialized view is refreshed.

Table A-10. *Files for Chapter 11*

File Name	Description
`dpi.sql`	This script shows the behavior of direct-path inserts related to the utilization of the buffer cache, the generation of redo and undo, and the support of triggers and foreign keys.
`dpi_performance.sql`	This script is used to compare direct-path inserts with conventional inserts. It was used to generate Figure 11-10 and Figure 11-11.
`makefile.mk`	This is the makefile I used to compile the C programs given as an example.
`mv.sql`	This script shows the basic concepts of materialized views.
`mv_refresh_log.sql`	This script shows how fast refreshes based on materialized view logs work.
`mv_refresh_pct.sql`	This script shows how fast refreshes based on partition change tracking work.
`mv_rewrite.sql`	This script shows several examples of query rewrite.
`px_ddl.sql`	This script shows several examples of parallel DDL statements.
`px_dml.sql`	This script shows several examples of parallel DML statements.
`px_dop1.sql`	This script shows the impact of the initialization parameter `parallel_min_percent`.
`px_dop2.sql`	This script shows that hints do not force the query optimizer to use parallel processing. They simply override the default degree of parallelism.
`px_query.sql`	This script shows several examples of parallel queries.
`px_tqstat.sql`	This script shows what kind of information the dynamic performance view `v$pq_tqstat` displays.
`result_cache_query.sql`	This script shows an example of a query that takes advantage of the server result cache.
`result_cache_plsql.sql`	This script shows an example of a PL/SQL function that implements the PL/SQL function result cache.
`row_prefetching.sql`, `row_prefetching.c`, `RowPrefetching.cs`, `RowPrefetching.java`	These scripts provide examples of implementing row prefetching with PL/SQL, OCI, JDBC, and ODP.NET.
`RowPrefetchingPerf.java`	This script shows that row prefetching can greatly improve the response time of a query that retrieves many rows. It was used to generate Figure 11-12.

Chapter 12

The files listed in Table A-11 are available for download for Chapter 12.

Table A-11. *Files for Chapter 12*

File Name	Description
`buffer_busy_waits.sql`	This script shows an example of processing that causes plenty of `buffer busy waits` events.
`buffer_busy_waits.zip`	This file contains the trace files and the output of TKPROF and TVD$XTAT used in the section “Block Contention.”
`column_order.sql`	This script shows that the position of a column in a row determines the amount of processing needed to access it. The script was used to generate the values represented in Figure 12-2.
`data_compression.sql`	This script shows that the performance of I/O bound processing might be improved thanks to data compression.
`reverse_index.sql`	This script shows that range scans on reverse indexes cannot be used to apply restrictions based on range conditions.
`wrong_datatype.sql`	This script shows that the decisions of the query optimizer are badly affected by the utilization of wrong datatypes.

APPENDIX B

Bibliography

Adams, Steve, "Oracle Internals and Advanced Performance Tuning." Miracle Master Class, 2003.

Alomari, Ahmed, *Oracle8i & Unix Performance Tuning*. Prentice Hall PTR, 2001.

Antognini, Christian, "Tracing Bind Variables and Waits." SOUG Newsletter, 2000.

Antognini, Christian, "When should an index be used?" SOUG Newsletter, 2001.

Antognini, Christian, Dominique Duay, Arturo Guadagnin, and Peter Welker, "Oracle Optimization Solutions." Trivadis TechnoCircle, 2004.

Antognini, Christian, "CBO: A Configuration Roadmap." Hotsos Symposium, 2005.

Antognini, Christian, "SQL Profiles." Trivadis CBO Days, 2006.

Antognini, Christian, "Oracle Data Storage Internals." Trivadis Traning, 2007.

Booch, Grady, *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, 1994.

Brady, James, "A Theory of Productivity in the Creative Process." IEEE Computer Graphics and Applications, 1986.

Breitling, Wolfgang, "A Look Under the Hood of CBO: the 10053 Event." Hotsos Symposium, 2003.

Breitling, Wolfgang, "Histograms—Myths and Facts." Trivadis CBO Days, 2006.

Breitling, Wolfgang, "Joins, Skew and Histograms." Hotsos Symposium, 2007.

Brown, Thomas, "Scaling Applications through Proper Cursor Management." Hotsos Symposium, 2004.

Chaudhuri, Surajit, "An Overview of Query Optimization in Relational Systems." ACM Symposium on Principles of Database Systems, 1998.

Dageville, Benoît and Mohamed Zait, "SQL Memory Management in Oracle9*i*." VLDB Conference, 2002.

Dageville, Benoît et al, "Automatic SQL Tuning in Oracle 10*g*." VLDB Conference, 2004.

Database Language – SQL. ANSI, 1992.

*Database Languag*e *– SQL – Part 2: Foundation*. ISO/IEC, 2003.

Date, Chris, *Database In Depth*. O'Reilly, 2005.

Dell'Era, Alberto, "Join Over Histograms." 2007.

Dyke, Julian, "Library Cache Internals." 2006.

Ellis, Jon and Linda Ho, *JDBC 3.0 Specification*. Sun Microsystems, 2001.

Engsig, Bjørn, “Efficient use of bind variables, cursor_sharing and related cursor parameters.” Miracle White Paper, 2002.

Foote, Richard, Richard Foote’s Oracle blog (`http://richardfoote.wordpress.com`).

Green, Connie and John Beresniewicz, “Understanding Shared Pool Memory Structures.” UKOUG Conference, 2006.

Goldratt, Eliyahu, *Theory of Constraints*. North River Press, 1990.

Gongloor, Prabhaker, Sameer Patkar, “Hash Joins, Implementation and Tuning.” Oracle Technical Report, 1997.

Gülcü Ceki, *The complete log4j manual*. QOS.ch, 2003.

Jain, Raj, *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

Kolk, Anjo, “The Life of an Oracle Cursor and its Impact on the Shared Pool.” AUSOUG Conference, 2006.

Knuth, Donald, *The Art of Computer Programming, Volume 3 – Sorting and Searching*. Addison-Wesley, 1998.

Kyte, Thomas, *Effective Oracle by Design*. McGraw-Hill/Osborne, 2003.

Lahdenmäki, Tapio and Michael Leach, *Relational Database Index Design and the Optimizers*. Wiley, 2005.

Lewis, Jonathan, *Cost-Based Oracle Fundamentals*. Apress, 2006.

Lewis, Jonathan, “Hints and how to use them.” Trivadis CBO Days, 2006.

Lewis, Jonathan, Oracle Scratchpad (`http://jonathanlewis.wordpress.com`).

Lilja, David, *Measuring Computer Performance*. Cambridge Universtity Press, 2000.

Machiavelli Niccoló, *Il Principe*. Einaudi, 1995.

Mahapatra, Tushar and Sanjay Mishra, *Oracle Parallel Processing*. O’Reilly, 2000.

Menon, R.M., *Expert Oracle JDBC Programming*. Apress, 2005.

Mensah, Kuassi, *Oracle Database Programming using Java and Web Services*. Digital Press, 2006.

Merriam-Webster online dictionary (`http://www.merriam-webster.com`)

Millsap, Cary, “Why You Should Focus on LIOs Instead of PIOs.” 2002.

Millsap, Cary with Jeff Holt, *Optimizing Oracle Performance*. O’Reilly, 2003.

Moerkotte, Guido, *Building Query Compilers*. 2006.

Nørgaard, Mogens et al, *Oracle Insights: Tales of the Oak Table*. Apress, 2004.

Oracle Corporation, “EVENT: 10046 ‘enable SQL statement tracing (including binds/waits).’” MetaLink note 21154.1, 2003.

Oracle Corporation, “Remove Functional Index need for QUERY_REWRITE/ QUERY_REWRITE_ENABLED.” MetaLink note 2799028.8, 2003.

Oracle Corporation, "Understanding Bitmap Indexes Growth while Performing DML operations on the Table." MetaLink note 260330.1, 2004.

Oracle Corporation, "Delete or Update running slow—db file scattered read waits on index range scan." MetaLink note 296727.1, 2005.

Oracle Corporation, "Interpreting Raw SQL_TRACE and DBMS_SUPPORT.START_TRACE output." MetaLink note 39817.1, 2007.

Oracle Corporation, "Table Prefetching causes intermittent Wrong Results in 9iR2, 10gR1, and 10gR2." MetaLink note 406966.1, 2007.

Oracle Corporation, "Tracing Sessions in Oracle Using the DBMS_SUPPORT Package." MetaLink note 62160.1, 2007.

Oracle Corporation, "Trace Analyzer TRCANLZR—Interpreting Raw SQL Traces with Binds and/or Waits generated by EVENT 10046." MetaLink note 224270.1, 2008.

Oracle Corporation, "The DBMS_SUPPORT Package." MetaLink note 62294.1, 2002.

Oracle Corporation, "Oracle JDBC Thin Driver generates different Execution Plan." MetaLink note 273635.1, 2007.

Oracle Corporation, "Handling and resolving unshared cursors/large version_counts." MetaLink note 296377.1, 2007.

Oracle Corporation, "CASE STUDY: Analyzing 10053 Trace Files." MetaLink note 338137.1, 2006.

Oracle Corporation, *Oracle9i Documentation, Release 2*.

Oracle Corporation, *Oracle Database Documentation, 10g Release 1*.

Oracle Corporation, *Oracle Database Documentation, 10g Release 2*.

Oracle Corporation, *Oracle Database Documentation, 11g Release 1*.

Oracle Corporation, *Oracle Database 10g: Performance Tuning*, Oracle University, 2006.

Oracle Corporation, "Query Optimization in Oracle Database 10g Release 2." Oracle white paper, 2005.

Oracle Corporation, "SQL Plan Management in Oracle Database 11g." Oracle white paper, 2007.

Quest Software, *JProbe Profiler: Developer's Guide, Version 7.0.1*.

Quest Software, *PerformaSure: User's Guide, Version 4.4*.

Senegacnik, Joze, "Advanced Management of Working Areas in Oracle 9i/10g." Collaborate, 2006.

Senegacnik, Joze, "How Not to Create a Table." Miracle Database Forum, 2006.

Shee, Richmond, "If Your Memory Serves You Right." IOUG Live! Conference, 2004.

Shee, Richmond, Kirtikumar Deshpande and K Gopalakrishnan, *Oracle Wait Interface: A Pratical Guide to Performance Diagnostics & Tuning*. McGraw-Hill/Osborne, 2004.

Shirazi, Jack, *Java Performance Tuning*. O'Reilly, 2003.

Sun Microsystems, *Java 2 Platform Standard Edition 5.0 Development Kit* documentation.

Vargas, Alejandro, “10g Questions and Answers.” 2007.

Wikipedia encyclopedia (`http://www.wikipedia.org`).

Williams, Mark, *Pro .NET Oracle Programming*. Apress, 2005.

Williams, Mark, “Improve ODP.NET Performance.” *Oracle Magazine*, 2006.

Wustenhoff, Edward, *Service Level Agreement in the Data Center*. Sun BluePrints, 2002.

Zait, Mohamed, “Oracle10g SQL Optimization.” Trivadis CBO Days, 2006.

Index

Symbols

*** three stars, 75
+ operator, 440
_trace_files_public parameter, 73

A

accept_sql_profile procedure, 270, 271
acceptance testing, 7
access paths
  hints for, 255
  identifying/resolving inefficient, 339–349
  strong selectivity, SQL statements and, 372–408
  weak selectivity, SQL statements and, 350–372
access predicates, 211
access structures, altering, 248
action name attribute, 45
addBatch method, 524
address column, 200
administer sql management object privilege, 275, 303
advanced queuing processes (Qnnn), 71
advisor privilege, 275
aggregations, 492, 500
all-rows optimization, 441
all_rows hint, 276
Allround Automations' PL/SQLDeveloper, 105
alter any outline privilege, 289
alter any sql profile privilege, 275
ALTER INDEX RENAME statement, 383
ALTER INDEX statement, locking object statistics and, 157
ALTER MATERIALIZED VIEW statement, 473, 474
ALTER OUTLINE statement, 283
ALTER SESSION statement
  initialization parameters and, 70
  session-level execution environment alteration and, 261
ALTER SYSTEM statement, initialization parameters and, 70
ALTER TABLE statement, data compression and, 548
alter_sql_plan_baseline procedure, 300
alter_sql_profile procedure, 271
Analysis Engine (JProbe), 55
analysis road map, 37
ANALYZE INDEX statement, 399
ANALYZE statement, 109 157
ANALYZE TABLE… statement, 538
AND-EQUAL execution plan operation, 226
ANSI join syntax, 413, 481
anti-join subqueries, converting to regular joins, 449
anti-joins, 416, 441
Apache Logging Services Project, 43
APIs (application programming interfaces), 328–337
application benchmarks, 111
application code
  analysis road map and, 37
  instrumentation for, 42
  profiling analysis of, 48–58
application design, 6
application-monitoring tools, 9
application programming interfaces (APIs), 328–337
architectures, multitier, 35
A-Rows columns, 239, 241, 277
array interface, 522–525
ArrayBindCount property, 524

arrival rate, 5
ASM-related processes (Onnn), 71
ASSOCIATE STATISTICS statement, 120
Automatic Diagnostic Repository, 71
Automatic Tuning Optimizer, 265
Automatic Workload Repository (AWR), 195, 203, 219
aux_stats$, 112
AWR (Automatic Workload Repository), 195, 203, 219

B

background_dump_dest parameter, 70, 71
backup table, 109, 164
- creating/dropping, 166
- object statistics and, 146, 160
- system statistics and, 119

backward compatibility, ANALYZE statement and, 109
base tables, 461
basicfile storage method, 534
batch updates, 524
benchmarks, application/synthetic, 111
best practices
- for bind variables, 30
- for datatype selection, 533–535
- for object statistics gathering via dbms_stats package, 163

binary comparisons, 392
BINARY_DOUBLE datatype, 533
BINARY_FLOAT datatype, 533
bind variables
- advantages/disadvantages of, 22–30
- APIs and, 329
- best practices for, 30
- EXPLAIN PLAN statement and, 198
- graduating, 23
- peeking and, 25, 27
- prepared statements and, 317, 323

bit string datatypes, 534
BITMAP AND/BITMAP OR/BITMAP MINUS execution plan operations, 226
BITMAP CONVERSION FROM ROWIDS operation, 402
BITMAP CONVERSION TO ROWIDS operation, 382
BITMAP INDEX RANGE SCAN operation, 386
BITMAP INDEX SINGLE VALUE operation, 382, 386
bitmap indexes, 371, 378–389
- vs. B-tree indexes, 378
- composite, 400
- compression and, 398
- equality conditions and, 382
- IS NULL conditions and, 384
- min/max functions and, 389
- range conditions and, 386

bitmap-join indexes, 455
BITMAP KEY ITERATION execution plan operation, 227
bitmap_merge_area_size parameter, 190, 193
BLOB datatype, 534
block contention, 539–546
- identifying, 540
- resolving, 543–546

block prefetching, 422
block range granules, 491
blocking/nonblocking execution plan operations, 223, 225
blocks. *See* data blocks
bloom-filter pruning, 362
branch blocks, 133
broadcast distribution of rows, 493
B-tree indexes, 369, 378–389
- vs. bitmap indexes, 378
- bitmap plans for, 402
- composite, 395–399
- compression and, 398
- equality conditions and, 381
- IS NULL conditions and, 383
- min/max functions and, 389
- range conditions and, 384

bug 3168840, 474
bug 5759631, 283
build input, 434
BULK COLLECT statement, 519

bushy trees, 412
business perspective on performance optimization, 10

C

cache ID, 483
caching, 481
call graphs, 56
CallableStatement subclass, 524
cardinality, 13, 126
  checking estimated, 241–245
  wrong estimations and, 277
Cartesian product, 414
CBO (cost-based optimizer), 108
chained rows, 535
character string datatypes, 534
child cursors
  dynamic performance views and, 264
  identifying, 200
child_number column, 200
children, parent-child relationships and, 223, 226
clarity, of code, 6
clear_identifier procedure, 45
clear_used procedure, 290
client identifier attribute, 45
client information attribute, 45
client result cache, 482, 487, 488
client_id_trace_disable procedure, 67
client_id_trace_enable procedure, 67
client-side caches, 482, 487
client-side cursor caching, 331
client-side statement caching, 322, 326
  APIs and, 329
  JDBC and, 335
  ODP.NET and, 337
close_cursor procedure, 332
clustering factor, 375–378
coding, 6
column statistics, 122, 142, 143
column usage history, 145
columns, optimal order for, 527
comments, vs. hints, 254
comparing_object_statistics.sql, 158
components, analysis road map and, 39
composite indexes, 395, 398
composite partitioning, 365
compression, 398, 546
compulsive tuning disorder, 9, 309
COMPUTE STATISTICS statement, 158
concatenated indexes, 395
CONCATENATION execution plan operation, 226, 407
concise profiling analysis, 48–54
configure procedure, 304
configuring query optimizer, 169–193
  importance of, 170
  parameters for, 172–193
  step comprising, 170
CONNECT BY WITH FILTERING execution plan operation, 227, 233
CONNECT BY WITHOUT FILTERING execution plan operation, 226
Conquest Software's SQLDetective, 105
Console (JProbe), 55
constraints, 466, 508
consumers, 492, 494
container tables, 461
correlated columns, 131
cost-based optimizer (CBO), 108
cost-based transformations, 447
COUNT STOPKEY execution plan operation, 224
CPU cost model, 111
CPU costs, 117
cpu_cost_column_access.sql, 117
create any outline privilege, 289
create any sql profile privilege, 275
CREATE INDEX statement, locking object statistics and, 157
CREATE MATERIALIZED VIEW LOG statement, 476
CREATE MATERIALIZED VIEW statement, 461, 473, 474
CREATE OUTLINE statement, 281
CREATE PARTITION statement, 478
CREATE TABLE statement, data compression and, 548

create_outline procedure, 282
create_stgtab_sqlprof procedure, 273
create_stored_outlines parameter, 281
create_tuning_task function, 268
cross joins, 414, 424, 441
CTAS statements, 506
cursor_sharing parameter, 325
cursor_sharing_exact hint, 325
cursors
  client-side statement caching and, 322
  dynamic performance views and, 199
  dynamic SQL and, 331
  extended cursor sharing and, 27
  life cycle of, 15
  OCI and, 333
  ODP.NET and, 336
  parent/child, 18–22, 200
  row prefetching and, 519
  server-side statement caching and, 327
  sharing, 20–30, 325
  static SQL and, 330

D

data access operations, 339–408, 459
  identifying/resolving inefficient access paths and, 339–349
  parallel processing and, 490
  strong selectivity, SQL statements and, 372–408
  weak selectivity, SQL statements and, 350–372
data blocks, 133
  block contention and, 543
  reading/writing, 30
  rows stored into, 527
data compression, 398, 546
data dictionary
  dbms_stats package and, 109
  SQL profiles and, 271, 275
data dictionary table, 112
data warehouses, materialized views and, 480
database administrators, compulsive tuning disorder and, 9
database calls
  instrumentation for, 44–48
  tracing, 59–100
database character set, 534
database tier, analysis road map and, 38
database version upgrades
  hints and, 261
  stored outlines and, 280
database_trace_disable procedure, 68
database_trace_enable procedure, 68
datatype selection
  best practices for, 533–535
  pitfalls in, 529–533
datatypes, optimal, 529–535
DATE datatype, 530, 534
datetime datatypes, 534
db_file_multiblock_read_count parameter, 171, 175–179
dba_sql_plan_baselines view, 296, 297
dba_sql_profiles view, 271, 275
dbms_application_info package, 46
dbms_monitor package, 65–69
dbms_mview package, 473
dbms_outln package, 282, 289, 290
dbms_outln_edit package, 285, 289
dbms_profiler package, 100–105
  installing, 101
  reporting data generated by, 103
  triggering, 103
dbms_refresh package, 474
dbms_result_cache package, 485
dbms_session package, 45, 63
dbms_spm package, 293, 297–304
dbms_sql package, 329, 332, 519, 524
dbms_sqltune package, 267–275, 278
dbms_stats package, 109
  common services in, 164–167
  configuring, 147
  deleting object statistics via, 161
  gathering object statistics via, 136–155, 163
  locking object statistics via, 155
  managing object statistics via, 162
dbms_stats_job_10g.sql, 152
dbms_stats_job_11g.sql, 154

dbms_support package, 65
dbms_system package, 63
dbms_xplan package, 208–221, 433
  display function and, 19, 213–2177
  display_awr function and, 203, 219
  display_cursor function and, 201, 217
  function output generation and, 208–219
  query block names and, 257
  SQL profiles and, 279
  stored outlines and, 290
DDL statements, parallel processing and, 505–509
debugging events, 62, 63
define_column procedure, 332
degree of parallelism (DOP), 496–500
DELETE statement
  block contention and, 544
  data compression and, 548
  high watermark and, 122
delete_pending_stats procedure, 151
delete_system_stats procedure, 116
deleting object statistics, 161
design, 6
  partitioning and, 367–369
  physical, optimizing 527–548
detailed profiling analysis, 55
diagnostic_dest parameter, 71
dimension tables, star schemas and, 451
dimensions, 466
direct reads, 31
direct writes, 31
direct-path inserts, 475, 513–517
dispatcher processes (Dnnn), 71
display function, 197, 213–217
display_awr function, 203, 208, 219
display_cursor function, 201, 208, 217
  caution for, 230
  checking cardinality via, 241
  using, 217
display_sql_plan_baseline function, 297
divide-and-conquer approach to identifying performance problems, 35
DML statements
  array interface and, 522
  parallel processing and, 503–505
Dnnn (dispatcher processes), 71
domain indexes, 374
DOP (degree of parallelism), 496–500
downloads, 551–562
  array scripts, 560
  bind variable scripts, 552, 558
  buffer cache scripts, 562
  clustering factor script, 554
  CPU cost script, 554
  data compression scripts, 562
  data structure scripts, 559
  DML statement scripts, 561
  execution plan scripts, 555
  histogram script, 554
  join scripts, 560
  object statistics scripts, 554
  parsing scripts, 558
  prefetching scripts, 560, 561
  query optimization scripts, 555
  reverse indexes scripts, 562
  shareable cursor scripts, 552
  selectivity/cardinality scripts, 552
  SQL plan baseline scripts, 556
  statistics scripts, 554
  stored outline scripts, 557
  subquery unnesting scripts, 560
  trace file scripts, 553
  TVD$XTAT profiler scripts, 91
  wrong datatypes scripts, 562
driving row source, 228, 418
drop any outline privilege, 289
drop any sql profile privilege, 275
DROP OUTLINE statement, 289
DROP PARTITION statement, 478
DROP PRIVATE OUTLINE statement, 289
drop_by_cat procedure, 289
drop_extended_stats procedure, 132
drop_sql_plan_baseline procedure, 303
drop_sql_profile procedure, 275

dynamic initialization parameters, 70
  query rewrite and, 463, 465, 469
  server result cache and, 484
dynamic performance views, 195, 199–203, 263
dynamic SQL, 329

E

Embarcadero's Rapid SQL, 105
Enterprise Manager, 195, 203
  evolutions and, 300
  SQL plan baselines and, 291
  SQL profiles and, 265, 267
  stored outlines and, 288
equality conditions, 381
equi-joins, 414, 441
E-Rows columns, 241, 277
estimations, by query optimizer
  performing, 199, 206, 209
  problems when wrong, 241, 277
event 10053, 205
event 10132, 206, 260
evolutions, 299
evolve_sql_plan_baseline function, 299
EXECUTE IMMEDIATE statement, 329, 331
executeBatch method, 524
execution environment, altering, 261–265
execution plans, 22, 107, 195–245, 285
  determining efficiency of, 241–244
  hash joins and, 435
  interpreting, 221–241
  merge joins and, 425
  nested loop joins and, 418–423
  object statistics gathering and, 163
  obtaining, 195–221
  outer join conversions and, 448
  outer joins and, 440
  partition-wise joins and, 445
  reading/decomposing, 236
  star transformations and, 452–456
  subquery unnesting and, 450
  TKPROF output and, 83
  tuning. *See* tuning
  TVD$XTAT output and, 98
  types of operations and, 223
execution statistics, 199, 341, 343
  TKPROF output and, 86
  TVD$XTAT output and, 95, 98
EXPLAIN PLAN statement, 126, 195–199
  caution with, 198
  checking assumptions and, 244
  query block names and, 257
explain_rewrite procedure, 469
extended cursor sharing, 27
extended SQL trace, 62
extended statistics, 131
extensions, 131
extent map blocks, block contention and, 546

F

fact tables, star schemas and, 451
false positives/negatives, 8
fast full table scans, parallel queries and, 500
fast refreshes, 475–480
  indexes and, 464
  pitfalls/fallacies of, 481
FetchSize property, 521
FILTER execution plan operation, 225, 227, 229, 449
filter predicates, 211
filtering conditions, 417
first-rows optimization, 441
fixed tables, 137
fixes, side effects and, 12
FORALL statement, 523
force logging, 516
Forms, 373
four-table joins
  hash joins and, 436
  merge joins and, 427
  nested loop joins and, 420
freelist blocks, block contention and, 546
freelist groups, 545
frequency histograms, 125–129
FROM statement, join syntax and, 413, 417
full index scans, 350, 369–372
full outer joins, 415
full partition scans, 350, 352, 500
full partition-wise joins, 443

full table scans, 350
  full partition scans and, 352
  parallel queries and, 500
full-text-match query rewrite, 465, 470
function-based indexes, 390

G

gather_database_stats procedure, 136
gather_database_stats_job_proc procedure, 153, 155
gather_dictionary_stats procedure, 136
gather_fixed_objects_stats procedure, 137
gather_index_stats procedure, 137
gather_schema_stats procedure, 137
gather_stats_prog, 153, 155
gather_system_statistics procedure, 113
gather_system_stats procedure, 113
gather_table_stats procedure, 137
gathering option parameters, 142–146
general query rewrite, 466
global hints, 256
global indexes, 405
graduating bind variables, 23
granularity, 139
granules, 491
The Grinder, 50

H

hard parses, 18
hash cluster access, 346, 407
hash distribution of rows, 493
HASH JOIN execution plan operation, 226, 361, 436, 445
hash joins, 434–439, 441
hash partitioning, 364, 368
hash_area_size parameter, 190, 192
hash_value column, 200
header blocks, block contention and, 545
height-balanced histograms, 125, 127
hierarchical queries, CONNECT BY WITH FILTERING execution plan operation for, 233
high (weak) selectivity, 345, 350–372
high watermark, 122
high_value, 123
hints, 211, 252–261, 339
  categories of, 255
  vs. comments, 254
  database/query optimizer version upgrades and, 261
  global, 256
  how they work, 252–259
  pitfalls/fallacies of, 259
  SQL plan baselines and, 291–304
  stored outlines and, 280–291
histograms, 124–131
  frequency vs. height-balanced, 129
  gathering, 143
  TVD$XTAT output and, 99
history
  of column usage, 145
  of object statistics, 158, 160
  of statistics, 164

I

I/O cost model, 111
I/O costs, 117
I/O subsystem, 111
ignore_optim_embedded_hints hint, 276
implicit conversion, wrong datatype selection and, 532
import_sql_profile procedure, 278
IN conditions, 357, 374, 386, 407
in-memory sorts, 428
index access, 374–407
index compression, 398
INDEX FAST FULL SCAN operation, 370
INDEX FULL SCAN (MIN/MAX) operation, 389
INDEX FULL SCAN operation, 370
index full scans, parallel queries and, 500
index joins, 439
index-only scans, 402, 404
index-organized tables, 404, 407
INDEX RANGE SCAN (MIN/MAX) operation, 389
INDEX RANGE SCAN DESCENDING operation, 382
INDEX RANGE SCAN operation, 382, 384
index range scans, parallel queries and, 500

INDEX SKIP SCAN DESCENDING
operation, 398
INDEX SKIP SCAN operation, 397
index skip scans, 397
index statistics, 133, 140
INDEX UNIQUE SCAN operation, 381
indexes
global/local/nonpartitioned, 405
parallel processing and, 507
reverse, 544
secondary, 404
structure of, 133
initialization parameters, 111, 171, 173–193
dynamic, 70
execution environment alteration
and, 261
hints for, 255
INLIST ITERATOR operation, 374, 386
inner joins converting to outer joins, 448
inner loop, 228, 418
INSERT INTO … SELECT … statement, 513
INSERT statement
block contention and, 544
data compression and, 548
direct-path inserts and, 513
high watermark and, 122
inserts, 513, 506
instrumentation, 7, 40, 41–48
for application code, 42
for database calls, 44–48
vs. profiling analysis, 40
pros/cons of, 40
integration, 7
inter-operation parallelism, 492, 496
internal log tables, 475
INTERSECTION execution plan
operation, 226
intra-block row chaining, 536
intra-operation parallelism, 492, 496
IS NULL conditions, 383
Isidian's SQL Insight, 105
iterative life cycle, 3, 8

J
Java Management Extensions (JMX), 42
Java Virtual Machine Tool Interface (JVMTI), 55
Java with JDBC, bind variables and, 22
java.sql.PreparedStatement class, 335
java.sql.Statement class, 334
JDBC, 334
array interface and, 524
bind variables and, 22
row prefetching and, 520
session attributes for, 47
JMX (Java Management Extensions), 42
Jnnn (job queue processes), 71
join conditions, 417
join-dependent expressions, 479
join eliminations, 447
join-filter pruning, 362
join syntax, materialized views and, 481
join trees, 409–413
joins, 409–457
basic methods for, 418–439
choosing methods for, 417, 441
defined, 409–417
hints for, 255
optimizing, 422, 439, 442–456, 459
parallel processing and, 492, 500
PARTITION RANGE SUBQUERY
operation and, 360–362
restrictions (filtering conditions), vs. join
conditions, 417
syntax for, 413, 417
types of, 413–416
JPetStore (sample application),
instrumentation and, 4, 41
JProbe (Quest Software), 55
JVMTI (Java Virtual Machine Tool
Interface), 55

K
key performance indicators, 5

L
leaf blocks, 133
left-deep trees, 410, 438
left input data, 409

legacy syntax, for joins, 413, 417
library cache, parent/child cursors stored in, 18
licensing, SQL Tuning Advisor and, 278
life cycles
  of cursors, 15
  sequential/iterative, 3, 8
LIKE conditions, 387
LIMIT statement, 519
linguistic comparisons, 392
linguistic indexes, 392–395
list partitioning, 364, 368
literals, 24
  cursor sharing and, 325
  replaced with bind variables, 317, 325
load_plans_from_cursor_cache function, 293
load_plans_from_sqlset function, 297
local indexes, 405
lock_schema_stats procedure, 155
lock_table_stats procedure, 155
locking object statistics, 155–158
log4j logging framework, 43
logging frameworks, 42
logical reads, 30, 340–352
  row prefetching and, 343
  row retrieval tests and, 346
logical rowids, 404
logs, 167
LONG datatype (deprecated), 534
long parses, 309, 314, 324
low (strong) selectivity, 345, 372–408
low_value, 123

M

master tables, 461
MAT_VIEW ACCESS FULL access path, 461
MAT_VIEW REWRITE ACCESS FULL access path, 463
materialized view logs, 475–478, 481
materialized views, 459–481
  assembled into refresh groups, 474
  pitfalls/fallacies of, 481
  query rewrite and, 463, 465–472
  refreshes of, 472–480
  when to use, 480
max_dump_file_size parameter, 70
memory manager, 190
memory work areas, 200
memory_max_target parameter, 190
memory_target parameter, 190
MERGE JOIN execution plan operation, 226, 361, 426
merge joins, 424–434
  choosing compared with other join methods, 441
  four-table, 427
  two-table, 425–427
MERGE statement, 513
migrated rows, 535
min/max functions, 389
minimal logging, 514
MINUS execution plan operation, 226
Mnnn (MMON slave processes), 71
module name attribute, 45
multicolumn indexes, 395
multicolumn pruning, 363
multipass sorts, 433
MULTI-TABLE INSERT execution plan operation, 226
multithreaded servers, 190
multitier architectures, 35

N

national character set, 534
natural joins, 414
NCHAR datatype, 534
NCLOB datatype, 534
nested loop joins, 418–423
  choosing compared with other join methods, 441
  four-table, 420–422
  two-table, 418–420
NESTED LOOPS execution plan operation, 227, 228, 360, 419
NEVER REFRESH, 472
Nexus (PerformaSure), 49
no_cpu_costing hint, 111
nonpartitioned indexes, 405

NOPARALLEL statement, 496
noworkload statistics, 111, 113
nullable/not-nullable columns, 371
NUMBER datatype, 533
NVARCHAR2 datatype, 534

O

object statistics, 119–164
  comparing, 158
  deleting, 161
  gathering, 136–155, 162
  inaccurate/missing, 278
  keeping up-to-date, 162
  locking, 155–158
  purging, 164
  query optimizer and, 119, 171
  restoring from history, 165
  staleness of, 140
  types of, 120
object_statistics.sql, 120
OCI, 333, 513
  array interface and, 524
  row prefetching and, 519, 521
  session attributes for, 46
OCIBindByName method, 524
OCIBindByPos method, 524
OCIStmtExecute method, 524
ODP.NET, 336
  array interface and, 524
  row prefetching and, 521
  session attributes for, 47
on-disk sorts, 428, 430
one-pass sorts, 433
Onnn (ASM-related processes), 71
OPEN/FETCH/CLOSE statements, 329, 331
open_cursor procedure, 332
open_cursors parameter, 322, 328, 331
operations logs, 167
operator (+), in predicates, 440
optimizer mode, switching, 276
optimizer_capture_sql_plan_baselines parameter, 292, 295, 303
optimizer_dynamic_sampling parameter, 179–183
optimizer_features_enable parameter, 174, 276, 290
optimizer_index_caching parameter, 111, 172, 186
optimizer_index_cost_adj parameter, 111, 172, 183
optimizer_mode parameter, 171, 173
optimizer_secure_view_merging parameter, 187
optimizer_use_sql_plan_baselines parameter, 301
optimizing
  data access operations, 339–408, 459
  joins, 422, 439, 442–456, 459
  physical design, 527–548
OR conditions, 359
OR pruning, 359
Oracle Data Provider for .NET, 521
Oracle Database 10*g*
  Automatic Tuning Optimizer and, 265
  AWR repository and, 203
  BINARY_FLOAT/BINARY_DOUBLE datatypes and, 533
  bitmap indexes and, 379
  block prefetching and, 422
  child cursors and, 200
  COMPUTE STATISTICS statement and, 158
  CPU costs and, 117
  dbms_profiler package and, 101
  dbms_stats package configuration and, 147
  display_awr function and, 219
  display_cursor function and, 217
  dynamic performance views and, 263
  event 10132 and, 260
  granularity and, 139
  hints and, 275
  join elimination and, 447
  linguistic comparisons and, 392
  locking statistics and, 155
  MAT_VIEW ACCESS FULL access path and, 461
  MAT_VIEW REWRITE ACCESS FULL access path and, 463

multicolumn pruning and, 363
noworkload statistics and, 113
object statistics and, 136, 158, 163
OR pruning and, 359
plan table and, 197
row prefetching and, 518
Scheduler and, 152
scheduling statistics and, 152
server-side statement caching and, 327
session_cached_cursors parameter and, 331
SQL profiles and, 279
SQL trace and, 63, 65
SQL tuning sets and, 268
sql_id and, 18, 200, 207
statistics history and, 164
stored outlines and, 290
time model statistics and, 39
trace file format changes and, 75
TRCSESS tool and, 76
working tables and, 287
Oracle Database 11*g*
Automatic Diagnostic Repository and, 71
BLOB storage methods and, 534
data compression and, 548
dbms_stats package configuration and, 148
execution plans and, 423
extended cursor sharing and, 27
extended statistics and, 131
join-filter pruning and, 362
result caches and, 481
scheduling statistics and, 154
server-side statement caching and, 327
virtual columns and, 133
Oracle Enterprise Manager. *See* Enterprise Manager
Oracle9*i*
bind variable peeking and, 25
bug 3168840 and, 474
child cursors and, 200
client-side cursor caching and, 331
COMPUTE STATISTICS statement and, 158
CPU cost model and, 111
display function and, 197
granularity and, 139
hints and, 275
PARTITION RANGE SINGLE operation, caution for, 355
query optimizer and, 170
SQL trace and, 63
statistics history and, 164
TRCSESS tool and, 76
working tables and, 287
OracleCommand class, 521
OracleConnection class, 337
OracleDataReader class, 521
OracleDataSource class, 520
OracleDriver class, 520
OracleParameter class, 336
OraSRP, 77
ORDER BY statement, 370, 385, 405, 461, 467
outer joins, 415, 439
converting to inner joins, 448
join methods for, 441
outer loop, 228, 418
outlines. *See* hints
overindexation, 398

P

P->P (parallel to parallel operation), 493, 507
P->S (parallel to serial operation), 493, 506
pack_stgtab_baseline function, 302
pack_stgtab_sqlprof procedure, 273
parallel joins, 442
parallel processing, 489–513
creating/rebuilding indexes and, 507
DDL statements and, 505–509
disabling, 496
DML statements and, 503–505
hints for, 255
inserts/queries and, 506
parallel operations, relationships between, 493
pitfalls/fallacies of, 509–513
when to use, 509
parallel queries, 500

parallel slave processes (Pnnn), 71
PARALLEL statement, 496, 505
parallelism, 442
  degree of parallelism and, 496–500
  inter-operation/intra-operation, 492, 496
parent cursors, 200, 221–241
parent-child relationships
  stand-alone execution plan operations and, 223
  unrelated-combine execution plan operations and, 226
parse procedure, 332
parses, main types of, 309
parsing
  how it works, 18–30
  TKPROF output and, 83
parsing problems, 309–338
  identifying, 309–317
  resolving/minimizing, 317–324
  workarounds for, 324
PART JOIN FILTER CREATE operation, 362
partial partition-wise joins, 446
partial-text-match query rewrite, 465, 470
partition change tracking (PCT), 478
partition granules, 491
PARTITION HASH ALL operation, 364, 445
PARTITION HASH INLIST operation, 364
PARTITION HASH ITERATOR operation, 364
PARTITION HASH JOIN-FILTER operation, 364
PARTITION HASH SINGLE operation, 364
PARTITION HASH SUBQUERY operation, 364
partition keys, 478
PARTITION LIST ALL operation, 364
PARTITION LIST EMPTY operation, 364
PARTITION LIST INLIST operation, 364
PARTITION LIST ITERATOR operation, 364
PARTITION LIST JOIN-FILTER operation, 364
PARTITION LIST OR operation, 364
PARTITION LIST SINGLE operation, 364
PARTITION LIST SUBQUERY operation, 364
partition markers, 478
partition pruning, 352–369, 405
  composite partitioning and, 365
  hash/list partitioning and, 364
  join-filter pruning and, 362
  multicolumn pruning and, 363
  OR pruning and, 359
  range partitioning and, 352–364
  subquery pruning and, 361
PARTITION RANGE ALL operation, 358, 360, 406
PARTITION RANGE EMPTY operation, 359
PARTITION RANGE INLIST operation, 357
PARTITION RANGE ITERATOR operation, 356, 360, 362
PARTITION RANGE JOIN-FILTER operation, 362
PARTITION RANGE MULTI-COLUMN operation, 363
PARTITION RANGE OR operation, 359
PARTITION RANGE SINGLE operation, 354
PARTITION RANGE SUBQUERY operation, 360–362
partition-wise joins, 442–447
partitioned outer joins, 416, 441
partitioning, 339, 350, 352
  composite, 365
  design considerations and, 367–369
  hash, 364, 368
  list, 364, 368
  partition change tracking and, 478
  range, 352–364, 368
pause_profiler routine, 102
PCT (partition change tracking), 478
PCWC (parallel combined with child operation), 493
PCWP (parallel combined with parent operation), 493
pending statistics, 150
performance
  bind variables and, 22
  block contention and, 539–546
  datatype selection and, 533–535
  index compression and, 399

index fast full scans and, 370
in-memory/on-disk sorts and, 428
one-pass/multipass sorts and, 433
partial partition-wise joins and, 447
planning for, 3–8
problems with, 3–12, 35–106
row-based processing and, 375
row migration/row chaining and, 537
rowid access and, 373
transformations and, 447–456
performance figures, 4, 82
performance tuning. *See* tuning
PerformaSure (Quest Software), 48–54
PGA (program global area), 189–193
pga_aggregate_target parameter, 172, 190
phases in application development, 3
physical design, optimizing, 527–548
physical reads, 30
physical rowids, 404
physical writes, 31
pins, 539
PL/SQL, 329–333
array interface and, 523
bind variables and, 22
profiling data and, 100–105
row prefetching and, 518
session attributes for, 45
PL/SQL Developer (Allround Automations), 105
PL/SQL function result cache, 482, 486–488
placeholders, 335
plan baselines. *See* SQL plan baselines
plan stability. *See* stored outlines
plan table, 195, 197, 213
Pnnn (parallel slave processes), 71
popular values, 128
precision, 533
predicates, 211
preferences, 148
prepared statements, 317, 323, 329
PreparedStatement subclass, 524
private stored outlines, 285
privileges
SQL plan baselines and, 303
SQL profiles and, 275
stored outlines and, 289
probe input, 434
producers, 492, 494
profilers, 40
dbms_profiler package, 100–105
for analyzing trace file content, 77
JProbe, 55
key functionalities of, 48
PerformaSure, 48–54
TKPROF. *See* TKPROF profiler
TVD$XTAT. *See* TVD$XTAT profiler
profiling analysis, 40, 48–58
concise, 48–54
detailed profiling and, 55
vs. instrumentation, 40
pros/cons of, 40
program global area (PGA), 189–193
public stored outlines, 285
publish_pending_stats procedure, 151
purging object statistics, 164
PX PARTITION HASH ALL partition operation, 445
PX SEND execution plan operation, 447

Q

QC Order distribution of rows, 493
QC Random distribution of rows, 493
Qnnn (queuing processes), 71
queries, parallelizing, 506
query block names, 257
query blocks, 210
query coordinator, 490
query optimizer, 107
bind variables and, 24, 30
configuring, 169–193
correlated columns and, 131
histograms and, 124
inefficient access paths and, 345
object statistics and, 119, 163
query rewrite and, 462, 465–472

search space and, 18
system statistics and, 117
version upgrades, hints and, 261
wrong datatype selection and, 530
query result cache, 482
query rewrite, 462, 465–472
enabling, 463, 464
methods of, 465, 470
query_rewrite_enabled parameter, 290
query transformation techniques, hints for, 255
Quest Software
JProbe, 55
PerformaSure, 48–54
SQL Navigator, 105
Toad, 105, 195
queuing processes (Qnnn), 71
quick parses, 309, 310–313
resolving/minimizing parsing problems and, 317–323
TKPROF/TVD$XTAT and, 310–314

R

range conditions, 384
range distribution of rows, 493
range partitioning, 352–364, 368
Rapid SQL (Embarcadero), 105
Rational Unified Process, 3
RAW datatype, 534
RBO (rule-based optimizer), 108
read consistency, 343
reading data blocks, 30
real benchmarks, 111
REFRESH COMPLETE, 472
REFRESH FAST, 472
REFRESH FORCE, 472
refresh groups, 474
refresh procedure, 473
refresh_all_mviews procedure, 473
refresh_dependent procedure, 473
refreshes, 463, 472–480
automating, 474
methods of, 472
related-combine execution plan operations, 223, 227–236
remap_stgtab_sqlprof procedure, 273
report_tuning_task function, 268
requirements analysis, 4
reset_param_defaults procedure, 148
resource utilization, by code, 6
resources for further reading
ASSOCIATE STATISTICS statement, 120
AWR repository, 203
dbms_stats package, services in, 166
debugging events, 63
domain indexes, 374
The Grinder, 50
hints, 255
JDBC, statement caching and, 335
minimizing logical reads, 347
OraSRP, 77
plan table, 197
query optimizer, 107
SQL tuning sets, 268
Statspack, 203
Trace Analyzer, 77
trace files, 62, 206
wait events, 86
response time, 4
broken up into major components, 36
data compression and, 546
monitoring, tools for, 9
restoring object statistics from history, 165
restrictions (filtering conditions), 243, 417
result caches, 481–489
limitations of, 485, 487
pitfalls/fallacies of, 489
types of, 482
when to use, 488
resume_profiler routine, 102
RETURNING statement, 519
reverse indexes, 544
rewrite/no_rewrite hints, 462
right-deep trees, 411, 437
right input data, 409
right-outer hash joins, 440
robots, 9

robustness, of code, 6
root block, 133
round-robin distribution of rows, 493
row chaining, 535, 538
row-level security, 18, 212
row migration, 535, 538
row prefetching, 375–378, 517–521
  logical reads and, 343
  pitfalls/fallacies of, 521
row source operation, 199, 209
rowids, 478
  logical/physical, 404
  rowid access and, 346, 372
rows
  retrieval tests and, 346
  stored into blocks, 527
rule-based optimizer (RBO), 108
rules
  for parent-child relationships, 221, 238
  for stand-alone execution plan operations, 223

S
s= semi-join operator, 450
S->P (serial to parallel operation), 493, 506
scale, 533
Scheduler, 152
scheduling statistics, 152
search space, 18
secondary indexes, 404
securefile storage method, 534
security, trace files and, 73
segment header blocks, block contention and, 545
SELECT statement
  block contention and, 544
  query rewrite and, 465, 467
selectivity, 13, 345
  strong, 372–408
  weak, 350–372
self-joins, 415
semi-join subqueries, converting to regular joins, 449
semi-joins, 416, 441
sequential life cycle, 3, 8
serial processing, 489
serv_mod_act_trace_disable procedure, 67
serv_mod_act_trace_enable procedure, 67
server processes, 489, 494
server result cache, 482–485, 488
server-side caches, 482, 488
server-side statement caching, 326
service level agreements (SLAs) 5, 9
service time, 4
session analysis
  with JProbe, 56
  with PerformaSure, 51
session-level execution environment alteration, 261
session recordings
  with JProbe, 56
  with PerformaSure, 51
session_cached_cursors parameter, 327, 331
session_trace_disable procedure, 66
session_trace_enable procedure, 66
set operators, parallel queries and, 500
set_action procedure, 46
set_client_info, set_module procedure, 46
set_database_prefs procedure, 148
set_ev procedure, 64
set_identifier procedure, 45
set_module procedure, 46
set_param procedure, 147, 148
set_schema_prefs procedure, 148
set_sql_trace procedure, 63
set_sql_trace_in_session procedure, 63
set_table_prefs procedure, 148
setFetchSize method, 520
SGA (system global area), 190
shared server processes (Snnn), 71
shared servers, 190
Show Only Focus Path feature, 57
side effects, 12
Sign-on action (in JPetStore sample application), 41, 52, 56
single-row retrieval test, 346
single-table hash clusters, 407
SLAs (service level agreements), 5, 9

slave processes, 490–512
  configuring pools of, 494
  creating/rebuilding indexes and, 507
  degree of parallelism and, 496–500
  parallel processing pitfalls/fallacies and, 511
Snnn (shared server processes), 71
soft parses, 18
SORT AGGREGATE operation, 371
SORT ORDER BY operation, 385
sort_area_retained_size parameter, 190, 192
sort_area_size parameter, 190, 191
sort-merge joins, 424–434
sorts, parallel processing and, 492, 500
speed, of code, 6
SQL, static/dynamic, 329–333
SQL Developer, 373
SQL handle, 296
SQL Insight (Isidian), 105
SQL MODEL execution plan operation, 226
SQL Navigator (Quest Software), 105
SQL plan baselines, 291–304
  how they work, 291–303
  pitfalls/fallacies of, 304
  stored as accepted/nonaccepted, 293, 297, 299
  stored outlines and, 280, 291
SQL plan names, 295
SQL profiles, 244, 265–279
  creating manually, 278
  how they work, 265–278
  pitfalls/fallacies of, 279
  SQL Tuning Advisor for, 265, 267
SQL statement-level execution environment alteration, 262
SQL statements
  altering, 250
  parallel processing and, 489–513
  SQL plan baselines and, 291–304
  SQL tuning sets and, 268
  stored outlines and, 280–291
  strong selectivity and, 372–408
  tuning. *See* tuning
  weak selectivity and, 350–372
SQL trace, 59–73, 341
  enabling/disabling, 63–69
  extended SQL trace and, 62
  trace files and, 70–63
  triggering, 69
SQL Tuning Advisor, 265, 267, 278
SQL tuning sets, 268
sql_id, 18, 200, 203, 207, 208
sql_plan_baseline column, 297
SQLDetective (Conquest Software), 105
sqltext_to_signature function, 272
sqltune_category parameter, 273, 279
staging tables, 273, 276, 301
stand-alone execution plan operations, 223
  CONNECT BY WITH FILTERING execution plan operation and, 234
  FILTER execution plan operation and, 229
  NESTED LOOPS execution plan operation and, 229
  UPDATE execution plan operation and, 232
star schemas, 451
star transformations, 451–456
star_transformation_enabled parameter, 290, 452, 454
stars (***), trace files and, 75
start_profiler routine, 102
Starts columns, 239, 241
Statement class, 520, 524
static SQL, 330
statistics
  dbms_stats package for, 109, 164–167
  object, 119–164
  pending, 150
  query optimizer configuration and, 171
  system, 111–119
  time model, 39
statistics_level parameter, 70
Statspack, 195, 203
stop_profiler routine, 102
stored outlines, 280–291, 324
  how they work, 280–289
  pitfalls/fallacies of, 289
  public/private, 285
stress tests, 7

strong (low) selectivity, 345, 372–408
subquery pruning, 361
subquery unnesting, 449
syntax, for joins, 413, 417
synthetic benchmarks, 111
sysaux tablespace, 304
SYSSTATS_INFO, 112
SYSSTATS_MAIN, 112
SYSSTATS_TEMP, 112
system global area (SGA), 190
system monitoring, tools for, 8
system perspective on performance optimization, 10
system privileges
  SQL plan baselines and, 303
  SQL profiles and, 275
  stored outlines and, 289
system response time. *See* response time
system statistics, 111–119
  charts of, 116
  deleting old, 116
  gathering, 113–117
  query optimizer and, 117, 171
  snapshots of, 114
  types of, 111

T

TABLE ACCESS BY GLOBAL INDEX ROWID operation, 406
TABLE ACCESS BY INDEX ROWID operation, 374, 381
TABLE ACCESS BY LOCAL INDEX ROWID operation, 406
TABLE ACCESS BY USER ROWID operation, 374
TABLE ACCESS FULL operation, 350, 354, 461, 463
TABLE ACCESS HASH operation, 407
table aliases, 260
table queues, 494, 495
table statistics, 121
target object parameters, 138
TEMP TABLE TRANSFORMATION execution plan operation, 226
temporary tables, 454
test cases, 313, 317
  client-side statement caching and, 322
  cursor sharing and, 325
  dbms_sql package and, 332
  EXECUTE IMMEDIATE statement and, 331
  JDBC and, 334
  OCI and, 333
  ODP.NET and, 336
  OPEN/FETCH/CLOSE statements and, 331
  static SQL and, 330
test environment, 551
testing, 6, 7
text normalization, 271
theta joins, 414, 441
thousands-of-rows retrieval test, 348
time, unaccounted-for time and, 85, 93
time model statistics, 39
timed_statistics parameter, 70, 85
TIMESTAMP datatype, 534
TIMESTAMP WITH LOCAL TIME ZONE datatype, 534
TIMESTAMP WITH TIME ZONE datatype, 530, 534
timing information, in trace files, 70
TKPROF profiler, 77, 78–90
  arguments for, 78–81
  block contention and, 540
  for long parses, 314
  interpreting output of, 81–90
  for quick parses, 310
  tracing database calls via, 60
Toad (Quest Software), 105, 195
tools. *See* utilities
top-n queries, COUNT STOPKEY execution plan operation for, 224
Trace Analyzer, 77
trace files, 195, 205, 310, 314
  confidential information and, 73
  finding location of, 70
  information about, 89, 93
  limiting size of, 70
  SQL trace and, 59–63, 70–73

structure of, 73
timing information in, 70
TKPROF profiler, for analyzing content of, 78–90
TRCSESS, for extracting information from, 76
TVD$XTAT profiler, for processing, 90–100
unaccounted-for time and, 85, 93
Trace Kernel PROfiler. *See* TKPROF profiler
tracefile_identifier parameter, 72
tracing database calls, 59–100
transformations, 447–456
TRCSESS tool, 76
Trivadis Extended Tracefile Analysis Tool. *See* TVD$XTAT profiler
troubleshooting, by tackling one problem at a time, 11
tuning, 247–305
access structures, altering, 248
block contention and, 543
execution environment, altering, 261–265
hints and, 252–261
SQL plan baselines and, 291–304
SQL profiles and, 265–279
SQL statements, altering, 250
stored outlines and, 280–291
techniques for (table), 247
tuning tasks, 267
TVD$XTAT profiler, 77, 90–100
arguments for, 91
block contention and, 541
installing, 90
interpreting output of, 93–100
for long parses, 315
for quick parses, 312
two-table joins
hash joins and, 435
merge joins and, 425
nested loop joins and, 418

U
UGA (user global area), 15
unaccounted-for time, 85, 93
undo blocks, block contention and, 545
undo header blocks, block contention and, 545
UNION-ALL execution plan operation, 226
unit testing, 6
unlock_schema_stats procedure, 156
unlock_table_stats procedure, 156
unpack_stgtab_sqlprof procedure, 273
unrelated-combine execution plan operations, 223, 226
UPDATE execution plan operation, 227, 231
UPDATE statement, 231
block contention and, 544
data compression and, 548
update_by_cat procedure, 283
use_stored_outlines parameter, 289
user global area (UGA), 15
user productivity, performance and, 3
user response time, 4
user_dump_dest parameter, 70, 71
user_outline_hints view, 282
user_outlines view, 282
users, application dissatisfaction and, 8
utilities
dbms_profiler package, 100–105
Forms, 373
GUI, for PL/SQL profiling, 105
JProbe, 55
Oracle Enterprise Manager, 195, 203
PerformaSure, 48–54
Quest TOAD, 105, 195
response-monitoring, 9
SQL Developer, 373
SQL trace, 59–73
system-monitoring, 8
TKPROF profiler. *See* TKPROF profiler
TRCSESS, 76
TVD$XTAT profiler. *See* TVD$XTAT profiler

V
v$ses_optimizer_env dynamic performance view, execution environment and, 263
v$sql view, 297

v$sql_optimizer_env dynamic performance view, execution environment and, 264
v$sql_plan dynamic performance view, 199
  caution for, 230, 234, 240
  running queries against, 201
v$sql_plan_statistics dynamic performance view, 199
v$sql_plan_statistics_all dynamic performance view, 200
  caution for, 230, 234, 240
  running queries against, 201
v$sql_workarea dynamic performance view, 200
v$sys_optimizer_env dynamic performance view, execution environment and, 263
variable-length datatypes, 533
variable peeking, 25, 27
version upgrades
  hints and, 261
  stored outlines and, 280
views, 459
  hints and, 261
  materialized, 459–481
virtual columns, 133
VPD predicates, 18, 212

W
wait events, 422
  TKPROF output and, 84
  TVD$XTAT output and, 99
wait time, 4
waterfall life cycle model, 3
weak (high) selectivity, 345, 350–372
WHERE statement
  column usage history and, 145
  join syntax and, 413, 417
work areas, 189, 200
  hash joins and, 438
  merge joins and, 428–434
workarea_size_policy parameter, 172, 189, 428
working tables, 285
workload statistics, 111, 113, 119
Workstation (PerformaSure), 49
writing data blocks, 30
wrong datatype selection, 529–535

Z
zig-zag trees, 412, 437