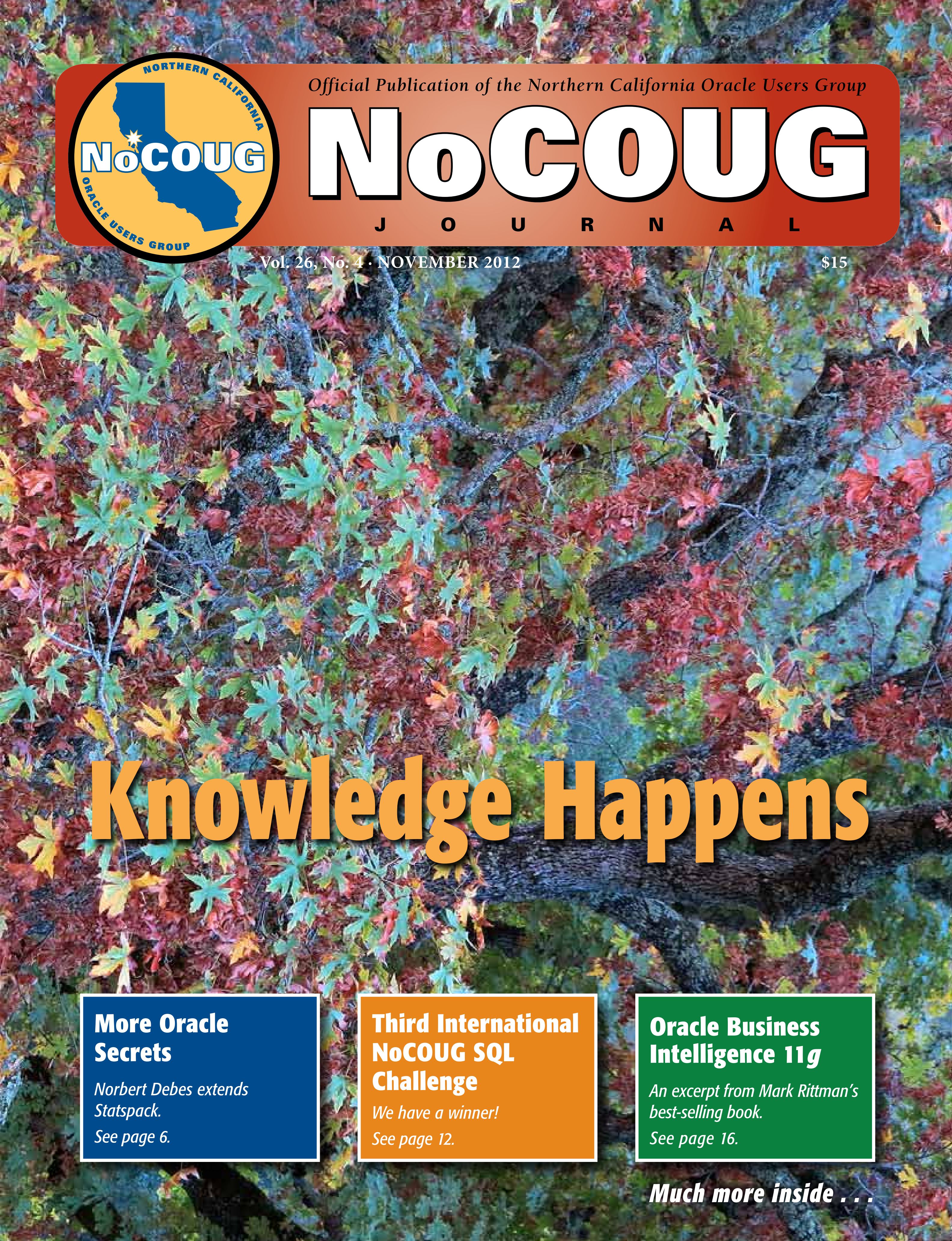


Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 26, No. 4 · NOVEMBER 2012 $15

Knowledge Happens

More Oracle Secrets

Norbert Debes extends Statspack.

See page 6.

Third International NoCOUG SQL Challenge

We have a winner!

See page 12.

Oracle Business Intelligence 11*g*

An excerpt from Mark Rittman's best-selling book.

See page 16.

Much more inside . . .

ORACLE ACCELERATION

YEAH, IT'S KIND OF LIKE THAT FAST!

- Accelerate Oracle databases up to 10x
- Deploy with minimal disruption to operations
- Extend the life of your IT infrastructure

Put Your Big Data on the Fast Track.

The GridIron Systems TurboCharger™ data acceleration appliance seamlessly integrates into your existing IT environment without changes to applications, databases, servers, storage or operational processes.

Learn more at **www.gridironsystems.com/oracle**.

GRIDIRON
Big Data. Fast

Thanking the Team

Take a moment to think about the huge amount of effort that goes into this publication. Your first thought might be about the care and attention required of the authors. Yes, writing is hard work. Now consider each author's years of hard-won experience; then add it up. The cumulative amount of time spent to acquire the knowledge printed in each issue is decades—maybe even centuries.

But let's take a moment to thank the people who make it possible for us to share this knowledge with you. Without the dedication and skill of our production team, all that we'd have is a jumble of Word files and a bunch of JPEGs. Copyeditor Karen Mead of Creative Solutions transforms our technobabble into readable English. Layout artist Kenneth Lockerbie and graphics guru Richard Repas give the *Journal* its professional layout.

Finally, what really distinguishes this *Journal* is that it is actually printed! Special thanks go to Jo Dziubek and Allen Hom of Andover Printing Services for making us more than just a magnetically recorded byte stream. ▲

—*NoCOUG Journal* Editor

Table of Contents

Publication Notices and Submission Format

The *NoCOUG Journal* is published four times a year by the Northern California Oracle Users Group (NoCOUG) approximately two weeks prior to the quarterly educational conferences.

Please send your questions, feedback, and submissions to the *NoCOUG Journal* editor at **journal@nocoug.org**.

The submission deadline for the upcoming February 2013 issue is November 30, 2012. Article submissions should be made in Microsoft Word format via email.

Copyright © 2012 by the Northern California Oracle Users Group except where otherwise indicated.

NoCOUG does not warrant the NoCOUG Journal *to be error-free.*

ADVERTISING RATES

The *NoCOUG Journal* is published quarterly.

Size	Per Issue	Per Year
Quarter Page	$125	$400
Half Page	$250	$800
Full Page	$500	$1,600
Inside Cover	$750	$2,400

Personnel recruitment ads are not accepted.

journal@nocoug.org

Optimization and Statistics

Maria Colgan

by Maria Colgan

Although the Optimizer often intimidates a lot of people, you definitely don't need magical powers to tune queries or to understand how the Optimizer works. I believe the mystique surrounding the Optimizer stems from the fact that there wasn't always a lot of information available on how the cost-based Optimizer (CBO) works and how statistics can affect an execution plan.

When we only had the rule-based Optimizer (RBO), things seemed a lot "easier," as the RBO chose its plans based on a set of fixed rules. Once you understood the rules, you really didn't need to know anything else to get the execution plan you wanted. For example, if you had the following query: 'select * from EMP where EMPNO < 5000', and there is an index on the EMPNO column, then one of the RBO's rules specifies that this query will always use the index. The plan chosen would be the same regardless of whether the EMP tables had ten rows or ten million rows and regardless of whether the 'EMPNO < 5000' predicate returned two rows or two million rows.

Then 20 years ago the cost-based Optimizer was introduced, and with it came the necessity for statistics and the fear of plan changes. The CBO generates multiple execution plans for a query, estimates the cost for each plan, and then chooses the plan with the lowest estimated cost. The cost model takes multiple inputs, including table, column, index, and system statistics. The use of statistics in the cost model implies that the CBO by its nature is more dynamic than the RBO. Collecting new statistics could lead to changes in execution plan if the new statistics are sufficiently different from the old ones. And although this behavior can seem scary, it is actually desirable since, as a database grows over time, the optimal execution plans for an application should also adapt.

The Secret of Optimal Execution Plans

Accurate statistics are the key to getting an optimal execution plan every time. Accurate statistics are a set of statistics that help the Optimizer to determine the correct number of rows it should expect from each operation in the execution plan. Accurate statistics do not necessarily need to be up-to-the-minute statistics.

Ideally the automatic statistics-gathering job will maintain the necessary statistics for most applications. The nightly job collects statistics for all database objects that are missing statistics or have stale statistics. Statistics are considered stale when 10% of the rows in the table have changed since statistics were last gathered. The automatic statistics-gathering job takes full advantage of the new performance-enhancing features in Oracle Database 11*g* to ensure that it gets to gather statistics on as many objects as possible in the limited time window. A good example of this is a new approximate NDV algorithm that will be used when the estimate percent parameter is allowed to default to AUTO_SAMPLE_SIZE. It uses a new hash-based sampling algorithm that provides deterministic statistics that have accuracy close to a 100% sample ("reading all data") but with the cost of, at most, a 10% sample.

When using the automatic statistics-gathering job, I strongly recommend that you monitor the frequency at which your larger tables have statistics gathered on them via the automatic job to see whether or not the default is good enough for these tables. If it's not, you can adjust the staleness threshold using the STALE_PERCENT table preference.

Let's take an example of a table that contains seven years' worth of sales data. New data is added daily. The statistics on this table won't be considered stale until 256 days of data have been loaded (10% of 7*365). Realistically, the statistics would be inaccurate long before that. Lowering the STALE_PERCENT to 1% would mean statistics would be gathered every 25 days.

There will always be cases where it is necessary to gather statistics manually. Typically this happens immediately after a data load. If you are going to manually gather statistics, we recommend that you use the default parameter values for the DBMS_STATS gather-statistics procedures. This will ensure a complete set of statistics in a timely manner.

There are also other key performance features that you should consider if you need to manually gather statistics in a timely manner. For example, you should take advantage of parallel execution, incremental statistics, and concurrent statistics-gathering techniques.

"Accurate statistics are the key to getting an optimal execution plan every time. Accurate statistics do not necessarily need to be up-to-the-minute statistics."

“The presence of a skew in the data, or correlation among the columns used together in the where clause predicates poses the biggest problem for the Optimizer. The Optimizer assumes that all data is uniform unless it is told otherwise.”

That said, it is the presence of a skew in the data, or correlation among the columns used together in the WHERE clause predicates, or in the GROUP BY clause of the SQL statement, that poses the biggest problem for the Optimizer. Unless the Optimizer is aware of the skew or the correlation, it is likely to pick a sub-optimal plan due to poor cardinality estimates. The Optimizer assumes that all data is uniform unless it is told otherwise. The good news is that you can make the Optimizer intimately aware of the nature of the data by providing additional column statistics, such as histograms to indicate skew and columns groups to demonstrate correlation.

If you are using an Exadata system, the same rules apply: you still need to provide the Optimizer with accurate information on the data you are accessing and the system you are running on so it can determine the optimal execution plans. In order to ensure that the Optimizer is fully aware of the performance that your Exadata system is capable of, you should gather system statistics in your environment using the new Exadata option. This will allow the Optimizer to fully understand the performance capabilities of the Exadata environment. Other than that you should operate just as you do on any other platform. Oracle will automatically take advantage of the Exadata features for each of the operations in the execution plan that will benefit from them.

New Features of Oracle Database 11*g*

The Optimizer development team is constantly working toward the goal of getting an optimal execution plan for every SQL statement, every time. As applications and SQL statements become more complex, we need to use new or alternative approaches to find optimal plans.

Sometimes, the nature of the SQL statements means that we can't make the call on whether or not we have the optimal plan until it's executed—hence the necessity to change the plan between executions in some cases. A good example of this is Adaptive Cursor Sharing in Oracle Database 11*g*. Adaptive Cursor Sharing allows multiple execution plans to be used for a single statement with bind variables. It relies on the monitoring of execution statistics to ensure that the correct plan is used for each bind value. On the first execution the Optimizer will peek at the bind value and determine the execution plan based on the bind values selectivity, just like it did in previous releases. The cursor will be marked bind sensitive if the Optimizer believes the optimal plan may depend on the value of the bind variable (for example, a histogram is present on the column or the predicate is a range) When a cursor is marked bind sensitive, Oracle monitors the behavior of the cursor, using different bind values, to determine if a different plan is called for. If a different bind value is used in a subsequent execution, it will use the same execution plan because Oracle initially assumes it can be shared. However, the execution statistics for this new bind value will be recorded and compared to the execution statistics for the previous value. If Oracle determines that the new bind value caused the data volumes manipulated by the query to be significantly different, it “adapts” and hard parses based on the new bind value on its next execution. The cursor is marked bind-aware. Each bind-aware cursor is associated with a selectivity range of the bind so that the cursor is only shared for a statement when the bind value in the statement is believed to fall within the range.

That said, the Optimizer team is aware of the pressure that DBAs face to ensure consistent response times, and the desire for plan stability on mission critical systems. For these types of environments we strongly recommend that you take advantage of SQL Plan Management (SPM) in Enterprise Edition to ensure plan stability. SPM incorporates the positive attributes of plan adaptability and plan stability while simultaneously avoiding their shortcomings.

SPM prevents performance regressions by ensuring that the Optimizer only uses known, verified plans. Any new plans found are recorded but are not used until they have been verified to perform better than the existing accepted plan, thus ensuring that all plan changes result in better performance. ▲

Maria Colgan is a Senior Principal Product Manager at Oracle Corporation and has been with the company since version 7.3 was released in 1996. Maria speaks regularly on query optimization and Optimizer statistics at NoCOUG conferences and—if the packed rooms are any indication—these are the most popular topics among conference attendees. Follow Maria on Twitter at @SQLMaria.

“If you are using an Exadata system, the same rules apply: you still need to provide the Optimizer with accurate information … In order to ensure that the Optimizer is fully aware of the performance that your Exadata system is capable of, you should gather system statistics in your environment using the new Exadata option.”

More Oracle Secrets Extending Statspack

by Norbert Debes

Norbert Debes

In this article, I will present an approach to extending the functionality of Statspack. Statspack is an alternative solution to the Active Workload Repository (AWR) for collecting instance-wide performance statistics. Statspack is included with the Oracle DBMS software at no extra cost, whereas AWR is a licensable component of the Enterprise Manager Diagnostics Pack.

Since the code for Statspack is not obfuscated (“wrapped”) using the Wrap utility, it is possible to modify the code. However modifying the package STATSPACK itself is not the approach I will propose herein. Rather, a separate package CSTATSPACK (“c” for custom) is created that will implement the extension. In this article I will propose a generic architecture for extending Statspack as well as a concrete example extension that persistently stores statistics pertaining to services that a DBMS instance provides to database clients.

Service Statistics and Instance Services

The dynamic performance view V$SERVICE_STATS was introduced with Oracle 10*g* Release 1. This view provides access to performance metrics on a per-service basis.

In this context a service is what I like to call an instance service that an RDBMS instance registers with a local or remote listener. Database clients use instance service names when they connect to an RDBMS instance. Using EasyConnect Naming, a client might use the following connect string format: <host>:<port>/<SERVICE_NAME>.

All services that an instance provides can be found in V$ACTIVE_SERVICES. The performance metrics of V$SERVICE_STATS are listed below:

- DB CPU
- DB time
- application wait time
- cluster wait time
- concurrency wait time
- db block changes
- execute count
- gc cr block receive time
- gc cr blocks received
- gc current block receive time
- gc current blocks received
- logons cumulative
- opened cursors cumulative
- parse count (total)
- parse time elapsed
- physical reads
- physical writes
- redo size
- session cursor cache hits
- session logical reads
- sql execute elapsed time
- user I/O wait time
- user calls
- user commits
- user rollbacks
- workarea executions - multipass
- workarea executions - onepass
- workarea executions - optimal

Note that all the client connections using the bequeath protocol have the instance service name SYS$USERS. Furthermore, connections over TCP/IP also get the instance service name SYS$USERS if they use SID instead of SERVICE_NAME in their connect descriptor (e.g., in tnsnames.ora).

It is also worth noting that the old-fashioned JDBC URL format jdbc:oracle:thin:@<host>:<port>:<ORACLE_SID> also results in a service name of SYS$USERS. Since the goal of using instance service names is to have a separate identifier for each application, the new JDBC URL format jdbc:oracle:thin:@<host>:<port>/<SERVICE_NAME> needs to be used. Unfortunately many application server setups (JBoss, Tomcat, etc.) still use the old format.

Architecture of Statspack

The database objects that implement Statspack reside in the schema PERFSTAT. This schema contains 72 tables in Oracle 11*g* Release 2 and an equal number of indexes. It also contains a single package called STATSPACK and a sequence called STATS$SNAPSHOT_ID, which is used for numbering snapshots. Whenever a snapshot is created, data is retrieved for the most part from dynamic performance views and stored in the tables.

A Statspack report is a SQL*Plus script that invokes the package STATSPACK to calculate the difference between the beginning and ending snapshots and creates a report using SELECT statements. The package STATSPACK contains several implementations of an overloaded subroutine called PURGE. This subroutine is used to remove outdated snapshots. Purging is done by invoking a single DELETE statement on the table STATS$SNAPSHOT. All of the tables containing actual snapshot data have a foreign key relationship with STATS$SNAPSHOT, which is created with ON DELETE CASCADE. Hence all the dependent rows are removed automatically when a row in the master table STATS$SNAPSHOT is deleted. The primary key of STATS$SNAPSHOT consists of the columns SNAP_ID (snapshot number), DBID (database ID) and INSTANCE_NUMBER.

Architecture of a Statspack Extension

Most conceivable extensions to Statspack will want to save information in V$ views and report on the differences ("deltas") between two snapshots. Other options exist too. For example, Statspack can be extended to gather workload system statistics in a statistics table outside of the data dictionary using DBMS_STATS.GATHER_SYSTEM_STATS.

In this article I will show how to augment Statspack with information from V$SERVICE_STATS. Oracle 11*g* Release 2 AWR reports contain four of 28 metrics available through V$SERVICE_STATS. The Statspack extension supports all 28 metrics listed above.

Any extension requires one or more tables to store information, which will come from V$ views most of the time. To integrate with the existing capabilities of Statspack, a new package CSTATSPACK ("C" for customized) is created. The new package accepts the same arguments as the procedure STATSPACK.SNAP. The implementation of CSTATSPACK.SNAP first invokes the function STATSPACK.SNAP to create a regular Statspack snapshot and to get the new snapshot number. Then it takes care of saving data in an additional table. Additional tables need to have the aforementioned foreign key relationship with STATS$SNAPSHOT to integrate with the purging approach of Statspack.

The Service Statistics Extension

Writing the extension involves creating a table for storing performance data from V$SERVICE_STATS, creating a package for taking snapshots and integrating the newly added data into Statspack reports.

The Table CSTATS$SERVICE_STATS

The table CSTATS$SERVICE_STATS will hold performance data from V$SERVICE_STATS. The DDL for creating the table is

```
create table perfstat.cstats$service_stats
(
  snap_id NUMBER NOT NULL,
  dbid NUMBER NOT NULL,
  instance_number NUMBER NOT NULL,
  service_name VARCHAR2(64),
  stat_name VARCHAR2(64),
  value NUMBER
);
```

The Package CSTATSPACK

The package CSTATSPACK is a wrapper around the original Statspack package that only contains the procedure SNAP, supports the same parameters as STATSPACK.SNAP, and implements one or more extensions. First of all a regular Statspack snapshot is taken using the function STATSPACK.SNAP. This returns a snapshot ID, which is then used to save performance measurements in the table CSTATS$SERVICE_STATS. The database ID and instance number of the snapshot are retrieved from STATS$SNAPSHOT. An excerpt of the package body is reproduced below:

```
v_snap_id := statspack.snap (
  i_snap_level,
  i_session_id,
  i_ucomment,
  i_num_sql,
  i_executions_th,
  i_parse_calls_th,
  i_disk_reads_th,
  i_buffer_gets_th,
  i_sharable_mem_th,
  i_version_count_th,
  i_seg_phy_reads_th,
  i_seg_log_reads_th,
  i_seg_buff_busy_th,
  i_seg_rowlock_w_th,
  i_seg_itl_waits_th,
  i_seg_cr_bks_rc_th,
  i_seg_cu_bks_rc_th,
  i_all_init,
  i_old_sql_capture_mth,
  i_pin_statspack,
  i_modify_parameter
 );

SELECT
  snap_id,
  dbid,
  instance_number
INTO
  v_snap_id,
  v_db_id,
  v_inst_nr
FROM perfstat.stats$snapshot
WHERE snap_id=v_snap_id;

INSERT INTO perfstat.cstats$service_stats
SELECT
  v_snap_id,
  v_db_id,
  v_inst_nr,
  service_name,
  stat_name,
  value
FROM v$service_stats
ORDER BY service_name;
```

Snapshots can now be taken by executing the procedure SNAP in the package CSTATSPACK:

```
EXEC perfstat.cstatspack.snap(i_snap_level => 0, i_ucomment => 'includes
service statistics')
```

Extending the Statspack Report

Statspack includes a configuration file called sprepcon.sql. However not all configurable parameters are set in sprepcon.sql. Some, such as top_n_events, are set in sprepins.sql. The file sprepins.sql is the main script that generates a report. One of the most important parameters set in sprepcon.sql is num_rows_per_hash, which controls the number of lines printed per SQL statement in Statspack reports. Its default is 4, which is much too low for investigating applications that often have SQL statements spanning 30 or more lines.

To avoid adapting sprepcon.sql in each and every ORACLE_HOME on many servers, I customarily copy all the Statspack SQL scripts (all of their names start with "sp") to a separate release-dependent directory, add the letter "c" (for custom) to the beginning of each file name (this has to be done inside the files too), and modify the files according to my preferences. Then I run all Statspack reports against, say, Oracle 11*g* Release 2 environments from that directory. Thus the main report script sprepins.sql becomes csprepins.sql.

Once a query that calculates the delta values for two snapshots has been written, it is very easy to integrate it into csprepins.sql. The original script sprepins.sql takes care of setting the database ID (dbid), instance number (inst_num), beginning snapshot ID (bid), and ending snapshot ID (eid) as SQL*Plus bind variables. An extension simply needs to use those bind variables to retrieve data pertaining to the snapshot range in question.

For the service statistics extension the query is:

```
SELECT
  b.service_name,
  b.stat_name,
  e.value,
  CASE
    WHEN b.stat_name='DB CPU' or b.stat_name LIKE '%time%'
    THEN to_char(round((e.value-b.value)/1000000, 3)) || ' s'
    ELSE to_char(e.value-b.value)
  END as delta_value
FROM perfstat.cstats$service_stats b, perfstat.cstats$service_stats e
WHERE b.dbid=:dbid
AND b.snap_id=:bid
AND b.instance_number=:inst_num -- always 1 for non-RAC
AND e.snap_id=:eid
AND e.dbid=b.dbid
AND e.instance_number=b.instance_number
AND b.service_name=e.service_name
AND b.stat_name=e.stat_name
ORDER BY b.service_name, b.stat_name;
```

In order to keep code for extensions separate from the Statspack scripts, this SELECT statement is saved in a separate file called spextend.sql. Some SQL*Plus commands for formatting the extended report section are included too.

Near the end of sprepins.sql there is a line that reads "prompt End of Report (&report_name)." All that is left to do is to call spextend.sql from just above the aforementioned line in sprepins.sql or, rather, csprepins.sql. The relevant section in csprepins.sql will read something like:

```
-- invoke Statspack extension
@@spextend.sql
-- end of Statspack extension
prompt
prompt End of Report ( &report_name )
```

Generating an Extended Statspack Report

The Statspack extension is now complete and ready to use by running the customized Statspack script cspreport.sql, which invokes csprepins.sql.

An excerpt that contains all 28 service statistics for the instance service name ELEVEN2.oradbpro.com is reproduced at the top of the following column.

Using the extension, performance metrics for individual applications are readily available, assuming that each application uses a separate instance service name. This information can be very helpful in determining which application causes

```
Instance Service Statistics  DB/Inst: ELEVEN2/eleven2  Snaps: 5-6

Instance Service Name   Statistic                        Value

ELEVEN2.oradbpro.com    DB CPU                           91.703 s
ELEVEN2.oradbpro.com    DB time                          94.858 s
ELEVEN2.oradbpro.com    application wait time            .003 s
ELEVEN2.oradbpro.com    cluster wait time                0 s
ELEVEN2.oradbpro.com    concurrency wait time            0 s
ELEVEN2.oradbpro.com    db block changes                 5266
ELEVEN2.oradbpro.com    execute count                    221871
ELEVEN2.oradbpro.com    gc cr block receive time         0 s
ELEVEN2.oradbpro.com    gc cr blocks received            0
ELEVEN2.oradbpro.com    gc current block receive time    0 s
ELEVEN2.oradbpro.com    gc current blocks received       0
ELEVEN2.oradbpro.com    logons cumulative                5
ELEVEN2.oradbpro.com    opened cursors cumulative        128460
ELEVEN2.oradbpro.com    parse count (total)              117336
ELEVEN2.oradbpro.com    parse time elapsed               10.857 s
ELEVEN2.oradbpro.com    physical reads                   21533
ELEVEN2.oradbpro.com    physical writes                  2
ELEVEN2.oradbpro.com    redo size                        287828
ELEVEN2.oradbpro.com    session cursor cache hits        60964
ELEVEN2.oradbpro.com    session logical reads            1444289
ELEVEN2.oradbpro.com    sql execute elapsed time         45.765 s
ELEVEN2.oradbpro.com    user I/O wait time               3.435 s
ELEVEN2.oradbpro.com    user calls                       867063
ELEVEN2.oradbpro.com    user commits                     18
ELEVEN2.oradbpro.com    user rollbacks                   0
ELEVEN2.oradbpro.com    workarea executions - multipass  0
ELEVEN2.oradbpro.com    workarea executions - onepass    0
ELEVEN2.oradbpro.com    workarea executions - optimal    73570
```

what load on an RDBMS instance. As smaller databases are consolidated into larger databases they share with several other applications, database administrators will probably value the additional information when they need to conduct performance analyses.

Download

A zip archive called cspcpkg-service_stats.zip contains all the SQL and PL/SQL code pertaining to this article. It is available for download at the following URL: http://www.nocoug.org/download/2012-11/cspcpkg-service_stats.zip.

Summary

Extending Statspack is fairly straightforward and simple. An extension consists of one or more additional tables for storing performance metrics. Those tables need to contain columns for the snapshot ID, database ID, and instance number, just like the original Statspack tables. A new package called CSTATSPACK is used to invoke the original Statspack package before populating the new tables that are part of the extension. The final step to complete an extension is to extend a customized copy of the Statspack script sprepins.sql to retrieve data from the tables of an extension. Bind variables identifying a snapshot range selected by the invoker of the report generation are made available in the script sprepins.sql and need to be used by SELECT statements that retrieve data from an extension's tables. ▲

Norbert Debes also wrote Secret ORACLE—Unleashing the Full Potential of the ORACLE DBMS by Leveraging Undocumented Features, *reviewed here in November 2008, which was subsequently updated and published by Apress as* Secrets of the Oracle Database. *He is focused on ORACLE DBMS performance, high availability, database security, backup/recovery, and training.*

Two Million Database Professionals Count on One Solution.

Simply the best for Oracle database professionals - Toad 11. Supported by over a decade of proven excellence, only Toad combines the deepest functionality available, extensive automation, and a workflow that enables database professionals of all skill and experience levels to work efficiently and accurately. Countless organizations empower their database professionals with Toad. The best just got better.

Watch the video at **www.quest.com/Toad11SimplyBetter**.

QUEST SOFTWARE®
Simplicity At Work™

Sometimes the Fastest UPDATE is an INSERT

by Tim Gorman

It's a problem many of us who live in Colorado have encountered: we want to drive from Winter Park to Denver. Easy: take US Highway 40 up and over Berthoud Pass, then down to I-70, and then take I-70 east all the way into Denver. Should only take an hour and a half, tops.

Except when Berthoud Pass has been closed due to winter weather. What then? Do we start heading east on US Highway 40 until the snow becomes higher than the windshield? Don't laugh. Plenty of people would do essentially this, and we find those folks when the snow melts.

Or, would you turn the other way, head in the opposite direction, westbound on US Highway 40, all the way northwest to Kremmling, then reverse direction down Highway 9 southbound to Silverthorne to reach I-70, then up and through the Eisenhower Tunnel, and so on into Denver. Yes, this route covers three times as many miles, but you get to Denver faster. It's not a great choice, but the first step is the most important: turning in the opposite direction and going that way as fast as you can. It's counter-intuitive, but sometimes it works.

When loading large volumes of data into an Oracle database, the same thing can be true: sometimes the fastest way to the destination seems to involve going in the opposite direction.

Let's suppose you have 100,000 rows of data to load into a slowly changing dimension table in a data warehouse in an Oracle database. Of those 100,000 rows, 50,000 are new, so they need to be inserted. The other 50,000 rows already have older copies in the table, so those rows will need to be updated with the new data. Often, this is called an "up-sert" or "merge" load, a combination of INSERTs and UPDATEs. Most dimension tables have new rows inserted or modified rarely, if at all. However, there are often dimensions that change a little (or a lot) with every load cycle. Typically, these dimensions have something to do with people, such as a dimension for employees or financial accounts or members.

There are a number of ways to merge this data; the most straightforward is to use the Oracle MERGE command, similar to the following:

```
MERGE INTO CURR_ACCT_DIM D
USING (SELECT acct_key FROM EXT_ACCT_DIM) X
ON (d.acct_key = x.acct_key)
WHEN MATCHED THEN UPDATE
SET d.eff_dt = x.eff_dt, d.attr_01 = x.attr_01, ..., d.attr_99 = x.attr_99
WHEN NOT MATCHED THEN
INSERT (d.acct_key, d.eff_dt, d.attr_01, d.attr_02, ..., d.attr_99)
VALUES (x.acct_key, x.eff_dt, x.attr_01, x.attr_02, ..., x.attr_99);
```

This does the job, but like the story about driving to Denver over Berthoud Pass, sometimes the most direct and straightforward route is not the fastest way to the destination. Especially when a large number of rows are being merged, and most particularly when a large number of rows are being updated.

In many ways, INSERTS are the fastest and most efficient type of data modification in any database, Oracle or otherwise.

➤ With an INSERT, there is no "search" phase to the operation, just placement of the row(s) into database blocks. From the perspective of transaction logging, not much redo (comparatively) is generated, just the "after image" of the newly inserted row.

➤ With a DELETE, there is a little more to do. There is a "search" phase to find the row, and then the row is removed. From the perspective of transaction logging, not much redo is generated—just the "before image" of the newly deleted row from the UNDO segment

➤ With an UPDATE, there is a lot to do. There is a "search" phase to find the row, and then there is the matter of changing data values in some or all of the columns. If the row grows in size and can no longer fit within the database block, then it must be migrated (similar to "chaining"). And, from the perspective of transaction logging, there is a lot of redo generated, both the "before-image" prior to the row being changed, stored within UNDO, and the "after-image" after the row has been updated.

Compared to DELETEs and UPDATEs, INSERT operations are a breeze. And they scale very well when run in parallel. There is even a special version of INSERT operations, called "direct-path," used for large volumes of data being inserted.

And, as it turns out, INSERTs are the very fastest way to accomplish a boatload of DELETEs or UPDATEs, if you're fortunate enough to be using table partitioning.

Here's how:

First, the easy part—the very fastest DELETE is an INSERT...

For each of the partitions in the table from which rows will be DELETEd . . .

1. Create a "scratch" table that has the same columns and physical storage attributes as the partition, using a CREATE TABLE . . . AS SELECT (CTAS) statement,

which queries the partition from which rows will be removed. The WHERE clause of the SELECT statement in the CTAS is the reverse condition of a DELETE statement, finding the rows that will be retained.

- ➤ The INSERT portion and the SELECT portion of the CTAS statement can both be run in PARALLEL.
- ➤ The INSERT portion can use the "direct-path" (aka APPEND) load mechanism in the CTAS statement.
- ➤ The NOLOGGING clause can be used in the CREATE attributes of the CTAS statement.
- ➤ The COMPRESS clause can be used in the CREATE attributes of the CTAS statement.

2. Perform ALTER TABLE . . . EXCHANGE PARTITION between the original partition and the newly created "scratch" table.

- ➤ As long as there are no global indexes, the EXCHANGE PARTITION operation is practically instantaneous.

3. Drop the "scratch" table.
4. Repeat for all partitions involved.

Fast, easy, and scalable. This is the way to delete hundreds of thousands or millions of rows from a table.

But UPDATEs aren't so easy. So I'm going to set up a scenario to help explain how it can be done. We have a dimension table named CURR_ACCT_DIM that is loaded into nightly. During these nightly loads, 50,000 rows are typically inserted and 50,000 rows are typically updated. The CURR_ACCT_DIM table has one unique B*Tree index supporting the primary key, a column named ACCT_KEY, and 30 single-column bitmap indexes. When the MERGE statement shown above was tested against this table, it took about 8 hours to complete, which is about three times too long.

What can be done?

Assuming that there is an external table (based on a "flat" file) named EXT_ACCT_DIM with the 100,000 rows of data to be up-serted or merged into the CURR_ACCT_DIM table, here is how we do it:

1. Range-partition the CURR_ACCT_DIM table with one (and only one) partition (called PDUMMY).

- ➤ The intent of doing this is only to use EXCHANGE-PARTITION during data loading, and not for partition-pruning during queries.
- ➤ Make sure that the partition key column is the primary key column, so that the unique index supporting the primary key constraint will be prefixed and thus able to be locally partitioned.

2. Create a non-partitioned "scratch" table (named SCRATCH_CURR_ACCT_DIM) that is "shaped" exactly like CURR_ACCT_DIM:

- ➤ The innermost subquery is a UNION-ALL, which queries all 100,000 of the rows from the EXT_ACCT_DIM and the existing contents of the CURR_ACCT_DIM. If CURR_ACCT_DIM previously had 10m rows already, then the output from this innermost subquery should be 10.1m rows (i.e., 10m plus 100k rows).

```
CREATE TABLE SCRATCH_CURR_ACCT_DIM
PARALLEL NOLOGGING COMPRESS AS
SELECT acct_key,
  eff_dt,
  attr_01, attr_02, ..., attr_99
FROM
  (SELECT acct_key,
    eff_dt,
    attr_01, attr_02, ..., attr_99,
    row_number() over (partition BY acct_key order by eff_dt DESC) rn
  FROM
    (SELECT acct_key, eff_dt, attr_01, attr_02, ..., attr_99
      FROM CURR_ACCT_DIM
    UNION ALL
    SELECT acct_key, eff_dt, attr_01, attr_02, ..., attr_99
      FROM EXT_ACCT_DIM
    )
  )
WHERE rn = 1;
```

- ➤ The middle subquery exists only to apply the ROW_NUMBER() window function to the results of the innermost subquery, so that we can easily identify the latest EFF_DT value for each ACCT_KEY value.
- ➤ The outermost subquery filters out any rows except for the ones with the latest EFF_DT for each ACCT_KEY value (i.e. "RN = 1") and inserts the remaining rows into the SCRATCH_CURR_ACCT_DIM table.

3. Gather table- and column-level statistics on SCRATCH_CURR_ACCT_DIM.
4. Create all LOCAL indexes that exist on CURR_ACCT_DIM onto SCRATCH_CURR_ACCT_DIM with the COMPUTE STATISTICS clause.
5. Finally, use "alter table CURR_ACCT_DIM exchange partition PDUMMY with table SCRATCH_CURR_ACCT_DIM" to publish the new contents of the dimension table for end users.

So in this way, the UPDATE/MERGE was performed with only INSERT/SELECT statements, and the resulting table can be compressed using Oracle 9*i* "basic" compression or, if on HCC-enabled storage, compressed using one of the HCC methods instead, thanks to the user of direct-path APPEND inserts. If need be, the NOLOGGING attribute can be set on the "scratch" table so that the direct-path APPEND loads don't generate redo, with the normal caveats and cautions when doing so.

Sometimes the fastest way to Denver is through Kremmling. And sometimes the fastest UPDATE is an INSERT. Trust me.

Have fun! ▲

Tim Gorman is the president of the Rocky Mountain Oracle Users Group. He is co-author of Oracle8 Data Warehousing *(Wiley, 1998),* Essential Oracle8*i* Data Warehousing *(Wiley, 2000),* Oracle Insights: Tales of the Oak Table *(Apress, 2004),* Expert Oracle Database Administration from the Oak Table *(Apress, 2009), and* Beginning Oracle SQL *(Apress, 2010).*

Third International NoCOUG SQL Challenge

Sponsored by Pythian—*Love Your Data*™

The Third International NoCOUG SQL Challenge was published in the May 2012 issue of the *NoCOUG Journal*. The winner, Lukasz Pluta of Poland, will receive an Amazon Kindle from contest sponsor Pythian and—in keeping with the pithy pronouncement of Steven Feuerstein that *"some people can perform seeming miracles with straight SQL, but the statements end up looking like pretzels created by somebody who is experimenting with hallucinogens"*—he will be made a knight of the August Order of the Wooden Pretzel.

Mind-Boggling Puzzle

The Wicked Witch of the West had invited six friends to the Third Annual Witching & Wizarding Ball at Pythian Academy of Es-Cue-El & No-Es-Cue-El. Burdock Muldoon and Carlotta Pinkstone both said they would come if Albus Dumbledore came. Albus Dumbledore and Daisy Dodderidge both said they would come if Carlotta Pinkstone came. Albus Dumbledore, Burdock Muldoon, and Carlotta Pinkstone all said they would come if Elfrida Clagg came. Carlotta Pinkstone and Daisy Dodderidge both said they would come if Falco Aesalon came. Burdock Muldoon, Elfrida Clagg, and Falco Aesalon all said they would come if Carlotta Pinkstone and Daisy Dodderidge both came. Daisy Dodderidge said she would come if Albus Dumbledore and Burdock Muldoon both came. The Wicked Witch of the West needed an Es-Cue-El or No-Es-Cue-El spell to determine whom she needed to persuade to attend the wizarding ball in order to ensure that all her invitees attend.

Mind-Blowing Solution

Master sorcerer Lukasz Pluta simply noted that if we start with a *minimal* set of invitees, then we can *augment* it—exactly one invitee at a time—by applying a sequence of rules until *everybody* has been included. Therefore, if we start with the *complete* list of invitees, we can *prune* it—exactly one invitee at a time—by applying the *same* sequence of rules but in *reverse* order until we find a *minimal* set of invitees. Lukasz assigned a power of two to each invitee (Albus = 1, Burdock = 2, Carlotta = 4, Daisy = 8, ...) and used binary arithmetic in a recursive SQL query; the anchor member of the query produces the complete list of invitees, while the recursive member prunes one invitee at a time. Here is his stunningly simple solution (with modifications for readability and efficiency); if there is more than one minimal set, then all of them are listed:

```
CREATE TABLE invitees
(
  invitee_id INTEGER,
  invitee_name VARCHAR2(128)
);

INSERT INTO invitees VALUES (1, 'Albus Dumbledore');
INSERT INTO invitees VALUES (2, 'Burdock Muldoon');
INSERT INTO invitees VALUES (4, 'Carlotta Pinkstone');
INSERT INTO invitees VALUES (8, 'Daisy Dodderidge');
INSERT INTO invitees VALUES (16, 'Elfrida Clagg');
INSERT INTO invitees VALUES (32, 'Falco Aesalon');

CREATE TABLE rules
(
  rule_id INTEGER,
  they_will_come INTEGER,
  if_they_come INTEGER
);

-- Burdock and Carlotta will come if Albus comes
INSERT INTO rules VALUES (1, 2 + 4, 1);

-- Albus and Daisy Dodderidge will come if Carlotta comes
INSERT INTO rules VALUES (2, 1 + 8, 4);

-- Albus, Burdock, and Carlotta will come if Elfrida comes
INSERT INTO rules VALUES (3, 1 + 2 + 4, 16);

-- Carlotta and Daisy will come if Falco comes
INSERT INTO rules VALUES (4, 4 + 8, 32);

-- Burdock, Elfrida, and Falco will come if Carlotta and Daisy come
INSERT INTO rules VALUES (5, 2 + 16 + 32, 4 + 8);

-- Daisy will come if Albus and Burdock come
INSERT INTO rules VALUES (6, 8, 1 + 2);
```

```
WITH working_sets(lvl, working_set, removed_id, is_leaf) AS
(
  -- The anchor member of the recursive query
  SELECT
    1 AS lvl,
    -- The complete list of invitees
    SUM(invitee_id) AS working_set,
    0 AS removed_id,
    0 AS is_leaf
  FROM invitees

  UNION ALL

  -- The recursive member of the query
  SELECT

    lvl + 1 AS lvl,

    -- Remove one invitee from the list
    CASE WHEN i.invitee_id IS NOT NULL
      THEN s.working_set - i.invitee_id
      ELSE s.working_set
    END AS working_set,

    invitee_id AS removed_id,

    -- Flag the leaf nodes
    CASE WHEN i.invitee_id IS NOT NULL
```

```
      THEN 0
      ELSE 1
    END AS is_leaf

  FROM

    working_sets s LEFT OUTER JOIN invitees i ON
    (
      -- The invitee is in the working set
      BITAND(s.working_set, i.invitee_id) = i.invitee_id

      -- The invitee will come since some others are coming
      AND EXISTS
      (
        SELECT * FROM rules r
        WHERE BITAND(r.they_will_come, i.invitee_id) = i.invitee_id
        AND BITAND(s.working_set, r.if_they_come) = r.if_they_come
      )
    )

  -- Terminate the recursion at leaf nodes
  WHERE s.is_leaf = 0
)
SEARCH DEPTH FIRST BY working_set SET seq
CYCLE lvl SET cycle TO 1 DEFAULT 0,

-- Eliminate duplicates
candidate_solutions AS
(
  SELECT DISTINCT working_set AS candidate_solution
  FROM working_sets
  -- Only consider leaf nodes
  WHERE is_leaf = 1
)

-- List minimal sets only
SELECT s1.candidate_solution AS solution
FROM candidate_solutions s1
WHERE NOT EXISTS
(
  SELECT * FROM candidate_solutions s2
  WHERE s2.candidate_solution < s1.candidate_solution
  AND BITAND(s1.candidate_solution, s2.candidate_solution)
    = s2.candidate_solution
);
```

The above solution limits the number of invitees to the number of bits in the numeric data type. The limitation can be overcome using multisets as follows:

```
CREATE TYPE invitees_t AS TABLE OF VARCHAR2(32)
/
CREATE TYPE solutions_t AS TABLE OF invitees_t
/

CREATE TABLE invitees
(
  invitee_id INTEGER,
  invitee VARCHAR2(32)
);

INSERT INTO invitees VALUES (1, 'A');
INSERT INTO invitees VALUES (2, 'B');
INSERT INTO invitees VALUES (3, 'C');
INSERT INTO invitees VALUES (4, 'D');
INSERT INTO invitees VALUES (5, 'E');
INSERT INTO invitees VALUES (6, 'F');

CREATE TABLE rules
(
  rule_id INTEGER,
  they_will_come invitees_t,
  if_they_come invitees_t
)
NESTED TABLE if_they_come STORE AS if_they_come_nt
NESTED TABLE they_will_come STORE AS they_will_come_nt;

INSERT INTO rules VALUES (1, invitees_t('B', 'C'), invitees_t('A'));
INSERT INTO rules VALUES (2, invitees_t('A', 'D'), invitees_t('C'));
INSERT INTO rules VALUES (3, invitees_t('A', 'B', 'C'), invitees_t('E'));
INSERT INTO rules VALUES (4, invitees_t('C', 'D'), invitees_t('F'));
INSERT INTO rules VALUES (5, invitees_t('B', 'E', 'F'), invitees_t('C', 'D'));
INSERT INTO rules VALUES (6, invitees_t('D'), invitees_t('A', 'B'));
```

```
WITH working_sets(lvl, working_set, removed, is_leaf) AS
(
  -- The anchor member of the recursive query
  SELECT
    1 AS lvl,
    -- The complete list of invitees
    CAST(MULTISET(SELECT invitee FROM invitees) AS invitees_t)
      AS working_set,
    NULL AS removed,
    0 AS is_leaf
  FROM dual

  UNION ALL

  -- The recursive member of the query
  SELECT

    lvl + 1 AS lvl,

    -- Remove one invitee from the list
    CASE WHEN i.invitee IS NULL
      THEN s.working_set
      ELSE s.working_set MULTISET EXCEPT invitees_t(i.invitee)
    END AS working_set,

    i.invitee AS removed,

    -- Flag the leaf nodes
    CASE WHEN i.invitee IS NULL
      THEN 1
      ELSE 0
    END AS is_leaf

  FROM

    working_sets s LEFT OUTER JOIN invitees i ON
    (
      -- The invitee is in the working set
      invitees_t(i.invitee) SUBMULTISET OF s.working_set

      -- The invitee will come since some others are coming
      AND EXISTS
      (
        SELECT * FROM rules r
        WHERE invitees_t(i.invitee) SUBMULTISET OF r.they_will_come
        AND r.if_they_come SUBMULTISET OF s.working_set
      )
    )

  WHERE s.is_leaf = 0
)
CYCLE lvl SET cycle_flag TO 1 DEFAULT 0,

-- Eliminate duplicates
candidate_solutions AS
(
  SELECT column_value AS candidate_solution
  FROM
    TABLE(SET(CAST(MULTISET(
      SELECT working_set FROM working_sets
      -- Only consider leaf nodes
      WHERE is_leaf = 1
    ) AS solutions_t)))
)

-- List minimal sets only
SELECT s1.candidate_solution AS solution
FROM candidate_solutions s1
WHERE NOT EXISTS
(
  SELECT * FROM candidate_solutions s2
  WHERE s2.candidate_solution SUBMULTISET OF s1.candidate_solution
  AND s2.candidate_solution != s1.candidate_solution

);
```

Lukasz Pluta becomes the fifth knight of the august order, following in the footsteps of Alberto Dell'Era (Italy), Andre Araujo (Australia), Rob van Wijk (Netherlands), and Ilya Chuhnakov (Russia).

Credits

The idea for the challenge came from a 1995 paper by C. J. Date titled *"Functional Dependencies are Fun: An Informal Look at the Formal World of FDs."* ▲

Relationally Speaking

by Iggy Fernandez

Iggy Fernandez

SQL comes in two distinct flavors—"relational calculus" and "relational algebra." Without sweating the technical details, let's just say that the relational calculus flavor is characterized by correlated subqueries—subqueries that refer to outside values—while the relational algebra flavor is characterized by set operations, such as JOIN, UNION, MINUS, and INTERSECT. And, as you have probably noticed, these flavors are often mixed. Consider the problem of listing all of the students who have enrolled in all of the courses required by their declared major. Here are the definitions and sample data for the three tables that are needed:

```
CREATE TABLE students
 (
   student_id INTEGER NOT NULL,
   major_id INTEGER NOT NULL,
   CONSTRAINT students_pk
     PRIMARY KEY (student_id)
 );

INSERT INTO students VALUES (1, 1);
INSERT INTO students VALUES (2, 1);
INSERT INTO students VALUES (3, 1);
INSERT INTO students VALUES (4, 1);
```

```
CREATE TABLE requirements
 (
   major_id INTEGER NOT NULL,
   course_id INTEGER NOT NULL,
   CONSTRAINT requirements_pk
     PRIMARY KEY (major_id, course_id)
 );

INSERT INTO requirements VALUES (1, 1);
INSERT INTO requirements VALUES (1, 2);
```

```
CREATE TABLE enrollments
 (
   student_id INTEGER NOT NULL,
   course_id INTEGER NOT NULL,
   CONSTRAINT enrollments_pk
     PRIMARY KEY (student_id, course_id),
   CONSTRAINT enrollments_fk1
     FOREIGN KEY (student_id) REFERENCES students
 );

INSERT INTO enrollments VALUES (1, 1);
INSERT INTO enrollments VALUES (1, 2);
INSERT INTO enrollments VALUES (2, 1);
INSERT INTO enrollments VALUES (3, 3);
INSERT INTO enrollments VALUES (4, 1);
INSERT INTO enrollments VALUES (4, 3);
```

Here are three solutions using the relational calculus flavor of SQL. In the first solution, we select those students for whom the count of enrolled required courses equals the count of required courses:

```
SELECT s.student_id
FROM students s
WHERE
 (
   SELECT COUNT(*)
   FROM requirements r, enrollments e
   WHERE r.major_id = s.major_id
   AND e.student_id = s.student_id
   AND e.course_id = r.course_id
 ) =
 (
   SELECT COUNT(*)
   FROM requirements r
   WHERE r.major_id = s.major_id
 );
```

In the second solution, we use double negation to select students such that there does not exist a required course in which they have not enrolled:

```
SELECT s.student_id
FROM students s
WHERE NOT EXISTS
 (
   SELECT *
   FROM requirements r
   WHERE r.major_id = s.major_id
   AND NOT EXISTS
   (
     SELECT *
     FROM enrollments e
     WHERE e.student_id = s.student_id
     AND e.course_id = r.course_id
   )
 );
```

In the third solution, we use object-relational techniques to select those students for whom the set of required courses is a subset of the set of enrolled courses:

```
CREATE TYPE list_type AS TABLE OF INTEGER;
/

SELECT s.student_id
FROM students s
WHERE

  CAST(MULTISET(
    SELECT r.course_id
    FROM requirements r
    WHERE r.major_id = s.major_id
  ) AS list_type)

  SUBMULTISET OF

  CAST(MULTISET(
    SELECT e.course_id
    FROM enrollments e
    WHERE e.student_id = s.student_id
  ) AS list_type);
```

Here is a solution that uses a mixed flavor of SQL (notice the use of the MINUS operation). We select those students for whom the set of required courses is a subset of the set of enrolled courses:

```
SELECT s.student_id
FROM students s
WHERE NOT EXISTS
  (
    SELECT r.course_id
    FROM requirements r
    WHERE r.major_id = s.major_id

    MINUS

    SELECT e.course_id
    FROM enrollments e
    WHERE e.student_id = s.student_id
  );
```

Here are three solutions using the relational algebra flavor of SQL. The first is the "textbook" solution, formed by converting the previous correlated subquery into an uncorrelated subquery:

```
SELECT student_id
FROM students

MINUS

SELECT student_id
FROM
  (
    SELECT s.student_id, r.course_id
    FROM requirements r, students s
    WHERE r.major_id = s.major_id

    MINUS

    SELECT e.student_id, e.course_id
    FROM enrollments e, students s
    WHERE e.student_id = s.student_id
  );
```

The second solution is due to Peter Johnson from Humboldt State University:

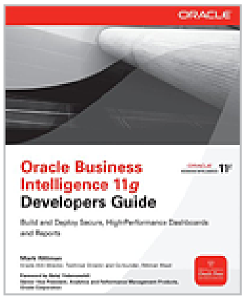
```
SELECT student_id FROM
(
  SELECT s.student_id, count(r.course_id)
  FROM students s, requirements r, enrollments e
  WHERE r.major_id  = s.major_id
  AND e.student_id = s.student_id
  AND e.course_id  = r.course_id
  GROUP BY s.student_id

  INTERSECT

  SELECT s.student_id, count(e.course_id)
  FROM students s, requirements e
  WHERE e.major_id = s.major_id
  GROUP BY s.student_id
);
```

The final solution is due to Fergal Taheny from Ireland:

```
SELECT s.student_id
FROM students s

INNER JOIN requirements r
ON (r.major_id = s.major_id)

LEFT OUTER JOIN enrollments e
ON (e.student_id = s.student_id AND e.course_id = r.course_id)

GROUP BY s.student_id
HAVING count(r.course_id) = count(e.course_id);
```

Although all of the above solutions are equivalent, they may not perform equally well, because equivalent SQL solutions of a problem are not guaranteed to use the same query plan and code paths. Depending on the data and circumstances, one solution may perform better than others. ▲

"The relational calculus flavor of SQL is characterized by correlated subqueries—subqueries that refer to outside values—while the relational algebra flavor is characterized by set operations, such as JOIN, UNION, MINUS, and INTERSECT."


Oracle Business Intelligence 11g Developers Guide
Build and Deploy Secure, High-Performance Dashboards and Reports
Mark Rittman

Oracle Business Intelligence Overview and Architecture

by Mark Rittman

This article is an excerpt from the book Oracle Business Intelligence 11g Developers Guide *published by Oracle Press, Sep. 2012, ISBN 0071798749; Copyright 2012. Reprinted with permission.*

Oracle Business Intelligence 11*g* Release 1 is a platform for delivering business intelligence across a wide range of data sources and to a wide range of audiences within the enterprise. You can consider it a "toolkit" in that, in itself, it does not come with any prebuilt reports, data, or other content, although as we will see later in this chapter, you can license content from Oracle Corporation and other providers that can be used with Oracle Business Intelligence.

Introducing Oracle Business Intelligence

As an end user, your first encounter with Oracle Business Intelligence would be when logging in to, and interacting with, a web-based dashboard such as that shown in Figure 1-1. Oracle Business Intelligence dashboards are made up of pages of analyses, displayed as tables, pivot tables, charts, gauges, or other views using data from potentially many sources. These can be interacted with, allowing the end user to, for example, start with a set of summarized figures and then progressively drill into more detail. Oracle Business Intelligence dashboards are typically highly graphical and provide a familiar, point-and-click environment for users to explore their data.

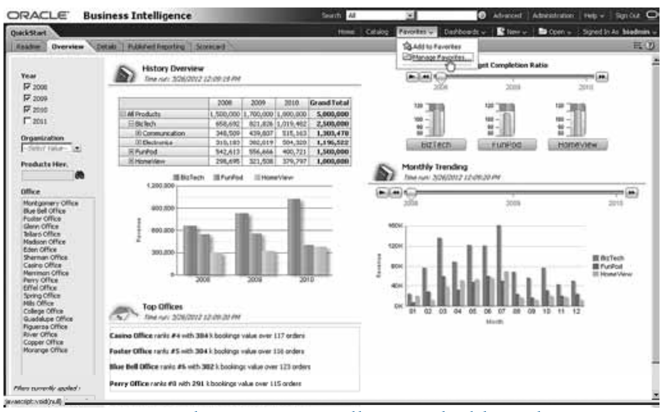


Figure 1: An Oracle Business Intelligence dashboard

For end users who wish to create their own reports, or those who wish to investigate their data in more detail, Oracle Business Intelligence allows users to create ad-hoc reports using data items taken from subject areas described using familiar business terms. Figure 1-2 shows a typical ad-hoc report, with a typical subject area made up of tables, columns, and hierarchies on the left-hand side of the screen that can be used to create table, chart, and other data views displayed on the right-hand side.

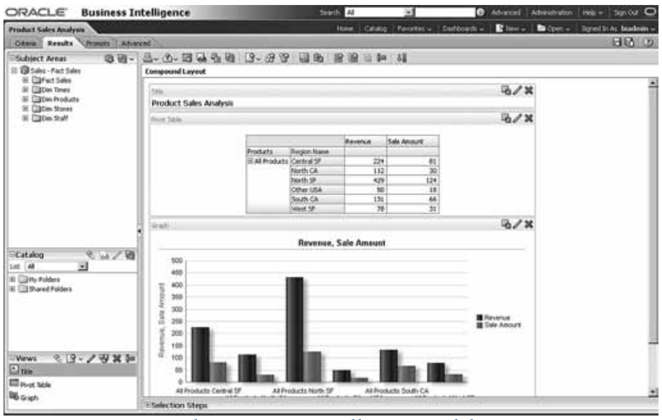


Figure 2: An Oracle Business Intelligence ad-hoc query

As well as providing an ad-hoc query environment suitable for data exploration, Oracle Business Intelligence also comes with tools for publishing reports in formats such as Adobe PDF and distributing them to large numbers of recipients through technologies such as e-mail. Analyses created using Oracle Business Intelligence can also be accessed through collaboration and office products such as Microsoft Outlook, Microsoft Word, Microsoft PowerPoint, and Microsoft Excel, or can be embedded directly into applications such as Oracle E-Business Suite, Siebel Customer Relationship Management (CRM), or in the new Fusion Applications from Oracle Corporation.

The 11.1.1.6 release of Oracle Business Intelligence 11*g*, on which the examples in this book are based, introduces new capabilities and visualization options, including the ability to create scorecards and key performance indicators, display data in the form of maps, and integrate with applications and business processes through a feature called the Action Framework. We will look at these capabilities in more detail in later chapters of this book.

The Development Toolkit

As developers, you have a number of tools within Oracle Business Intelligence that you can use to develop business intelligence solutions. The main tool that you will use is the Oracle Business Intelligence Administration tool, a Microsoft Windows–based tool that is used to define and then maintain the business metadata layer known as the Oracle BI Repository, used for creating reports and analyses. Several chapters of this book are devoted to this tool, and as a developer you will need to understand in detail the functionality of this tool and the Oracle BI Repository.

Figure 1-3 shows the Oracle Business Intelligence Administration tool, with a repository open for editing. The tool is also used for managing connection details through to data sources, defining security policies that control users' access to data, and performing a number of administration tasks such as defining variables, managing caching, and checking the status of the cluster.

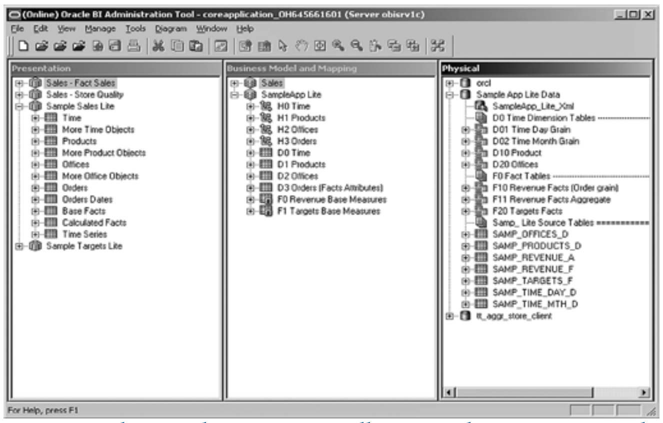


Figure 3: The Oracle Business Intelligence Administration tool

From the 11*g* release of Oracle Business Intelligence, a number of systems administration tasks previously carried out using the Oracle Business Intelligence Administration tool are now performed using Oracle Enterprise Manager Fusion Middleware Control. These tasks include enabling and disabling caching, setting the cache size, and managing the status of the system components in a cluster.

Other tools for developers provided as part of Oracle Business Intelligence 11*g* Release 1 include the Catalog Manager, a Java application used for managing the catalog of reports, dashboards, and other business intelligence objects; Oracle Enterprise Manager Fusion Middleware Control, for administering the business intelligence platform; and Oracle WebLogic Server Administration Console, for controlling the functionality of the WebLogic Server application server.

Reports, analyses, and dashboards are created using web-based authoring tools that do not require any separate software to be installed on developers' desktops. Oracle Business Intelligence 11*g* Release 1 brings together all report-authoring tools into a single integrated environment using the same semantic model, and later chapters in this book will show you how easy it is to create compelling, interactive dashboards using these tools.

Platform Support

At the time of writing this book, Oracle Business Intelligence is at release 11.1.1.6 and can be installed on a number of Microsoft Windows, Linux, and Unix platforms, running on both 32-bit and 64-bit processors. Because the report and dashboard-authoring tools within Oracle Business Intelligence are mostly web-based, any operating system that supports these web browsers can be used to create reports. There are separate certifications for server and client tools within Oracle Business Intelligence, and you should refer to the Oracle Technology Network web site, and in particular the System Requirements and Supported Platforms for Oracle Business Intelligence Suite Enterprise Edition 11*g*R1 document, to obtain the latest list of supported platforms and operating systems.

Note The list of certified operating systems and platforms can change from release to release, and you should check the System Requirements and Supported Platforms for Oracle Business Intelligence Suite Enterprise Edition 11*g*R1 document, available on the Oracle Technology Network web site (**http://otn.oracle.com**), for your particular version of Oracle Business Intelligence.

How Does Oracle Business Intelligence Work?

So you now know that Oracle Business Intelligence comes with a number of end-user tools for developing and viewing reports, together with developer tools for administration, creating the semantic model, and maintaining the system. But how does Oracle Business Intelligence work, how does it access your various data sources, and what use does it make of other business intelligence systems such as data warehouses, online analytical processing (OLAP) servers such as Oracle Essbase, or data that might be of interest in your applications or company databases?

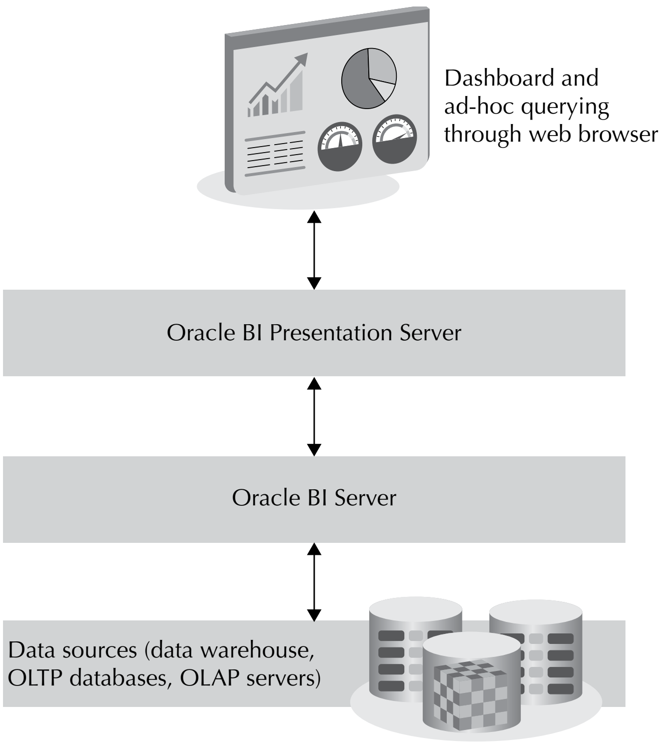

Dashboard and ad-hoc querying through web browser
Oracle BI Presentation Server
Oracle BI Server
Data sources (data warehouse, OLTP databases, OLAP servers)

Figure 4: High-level Oracle Business Intelligence schematic

At a high level, Oracle Business Intelligence uses a four-tier architecture that provides access to your data through two main servers and a semantic model. Figure 1-4 shows a high-level schematic for Oracle Business Intelligence, with your data being accessed through two servers—the Oracle Business Intelligence (BI) Server and the Oracle BI Presentation Server—before it is presented to end users through a web browser.

Considering this four-tier architecture from the perspective of an end user requesting a dashboard of business information, the components within Oracle Business Intelligence perform the following high-level functions to return data to the user:

1. The web browser requests a dashboard of data, consisting of analyses, published reports, and other BI content.
2. This request is received by the Oracle BI Presentation Server, which translates requests for individual analyses and reports into logical SQL queries. These logical queries are then passed to the Oracle BI Server.
3. The Oracle BI Server takes these logical SQL queries, written against the semantic model contained in the Oracle BI Repository, and translates them into native SQL and MDX queries that are then sent to the underlying data sources.
4. The underlying data sources process the native SQL and MDX queries, and return results to the Oracle BI Server.
5. The Oracle BI Server returns a data result set to the Oracle BI Presentation Server. In instances where more than one data source is needed to satisfy the query, the BI Server is capable of combining multiple data sets into a single set of results.
6. Finally, the Oracle BI Presentation Server presents the results back to the end user, in the form of analyses, published reports, dashboards, and other BI content.

Unlike many other business intelligence tools that combine data presentation with query generation in a single server, Oracle Business Intelligence splits these functions into two separate servers:

- **Oracle BI Server** This server provides simultaneous connectivity to heterogeneous data sources, a calculation and data federation engine, access to a semantic (metadata) model, and a security layer.
- **Oracle BI Presentation Server** This server connects to the Oracle BI Server and provides users with a catalog of analyses, reports, and dashboards that they can use to analyze their data.

Oracle Business Intelligence 11*g* actually makes use of other servers to handle clustering, scheduling of reports, and other services; but, for now, consider these two servers the "core" of the Oracle Business Intelligence's functionality.

So Where Does the Data Come From?

As you will have seen from the above schematic, Oracle Business Intelligence does not itself hold data; instead, it uses a metadata layer to create a "virtual dimensional model" over one or more data sources and then generates SQL and MDX queries to retrieve data, on demand, from these data sources for presentation back to the user. As such, it leverages any investment you have made in data warehouse technology such as Oracle Database Enterprise Edition, or in OLAP technology such as Oracle Essbase, rather than replacing the need for them.

While Oracle Business Intelligence can optionally be configured to hold a cache of data to enable faster display of results, queries that it generates are otherwise sent directly to your underlying data sources. It follows, therefore, that your underlying data sources should be as optimized for queries as possible. Furthermore, the Oracle BI Server can also take advantage of any analytic functionality that is available on a particular data source to enable more efficient processing by "passing down" calculations to the underlying data source. Therefore, the recommended, optimal data source for Oracle Business Intelligence would be an enterprise data warehouse running on a database platform such as Oracle Database Enterprise Edition, potentially supplemented or enhanced by an OLAP server such as Oracle Essbase or the OLAP Option for the Oracle Database Enterprise Edition, to provide fast access to aggregated data.

However, Oracle Business Intelligence also has the ability to connect to more than one data source and "join together" results from each one into a single data set, giving you the ability to create "virtual" data warehouses made up of data taken in real time from separate databases, which can be departmental data warehouses, data marts, or even online transaction processing (OLTP) databases or nonrelational database sources such as OLAP servers, files, or sources such as Microsoft Excel spreadsheets. This ability to work with "federated" data sources gives you great flexibility in how you design your reporting system, allowing you to, for example, source the majority of your reporting data from a data warehouse but supplement it with data sourced in real time from a range of applications, file sources, and OLAP servers, as shown in Figure 1-5.

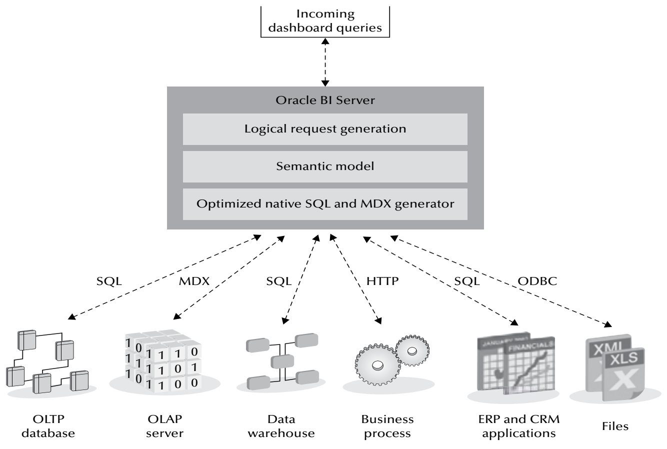


Figure 5: The Oracle BI Server accessing heterogeneous data sources

This approach becomes possible due to two key features provided by the Oracle BI Server:

- The semantic model, which can create a metadata model over multiple data sources from different vendors, presenting users with a single, unified view over their data regardless of the data source
- The ability of the BI Server to generate native, optimized queries for each data source and to combine the results returned into a single result set

So, it's clear that the BI Server and the semantic model that it uses are key to how Oracle Business Intelligence provides access to data. With this in mind, let's take a look at how the semantic model works and how it structures data so that it is optimized for querying.

The Oracle Business Intelligence Semantic Model

The Oracle Business Intelligence semantic model has three main objectives:

- To represent your enterprise's data as a logical dimensional model
- To map this logical dimensional model onto the data sources used by the enterprise
- To provide personalized views over the logical dimensional model for subsets of users giving them access to just the data they need while preserving, "under the covers," a single unified business intelligence data model

Semantic models in the Oracle Business Intelligence repository therefore have three distinct layers:

- **Physical layer** This layer contains metadata on the physical databases and other data sources that provide data for the semantic model.
- **Business Model and Mapping layer** This layer contains the logical dimensional model defined for the business.
- **Presentation layer** This layer provides personalized subsets of the logical dimensional model tailored for different audiences.

Data flows through the semantic model, as shown in Figure 1-6, from the Physical layer, through mappings into the Business Model and Mapping layer, and is eventually accessed by end users through the Presentation layer. The semantic model is defined and maintained using the Oracle Business Intelligence Administration tool and stored in the Oracle BI Repository, and Chapters 3 and 4 show you how semantic models can be created that access data from relational, OLAP, files, and other nonrelational sources.

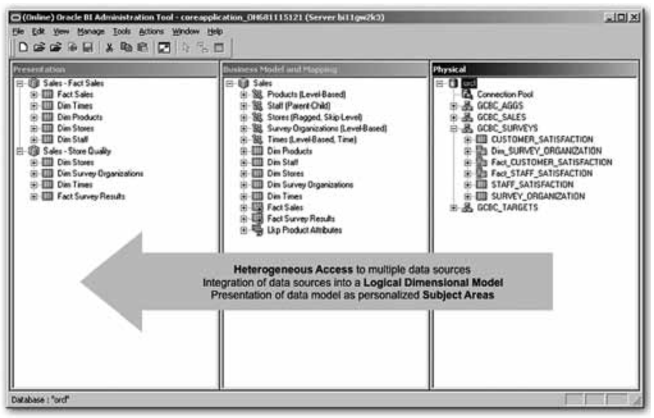


Figure 6: The Oracle Business Intelligence semantic model

Packaged Business Intelligence Solutions

The fact that Oracle Business Intelligence is, in effect, a toolkit that does not in itself provide any data or reports that you can use "out of the box" means that, realistically, you will need to spend a significant amount of time developing a solution before your users can start analyzing their data. In addition, because many organizations have standardized on packaged enterprise resource planning (ERP) systems such as Oracle E-Business Suite and PeopleSoft Enterprise or customer relationship management (CRM) suites such as Siebel CRM, the work you would be doing might essentially be "reinventing the wheel," as many organizations would have had requirements similar to yours in the past and created similar business intelligence solutions to deliver similar dashboards and reports.

Oracle Corporation, as well as third-party vendors such as Noetix, have addressed this opportunity by developing packaged sets of dashboards, data models, and data extraction routines that you can install, along with Oracle Business Intelligence, to provide dashboards and reports within days or weeks rather than the usual months that are required to create a custom solution. The Oracle Business Intelligence Applications, from Oracle Corporation, are a suite of packaged BI products built around Oracle Business Intelligence that provides applications such as the following:

- Financial Analytics
- Human Resources Analytics
- Project Analytics
- Procurement Analytics
- Supply Chain Analytics

In addition to these, it provides other, industry-specific "vertical" packaged applications for the financial services, pharmaceuticals, and other industries. Figure 1-7 shows the relationship between Oracle Business Intelligence Applications and Oracle Business Intelligence, and how data for this combined system is either accessed directly from application and database data sources, or through a prebuilt data warehouse fed by predefined data extraction routines.

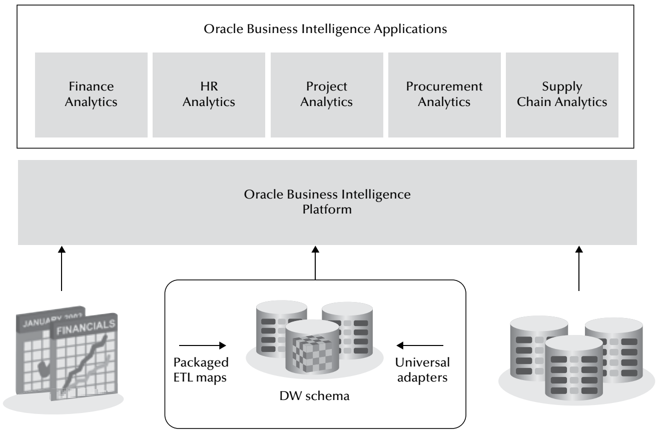


Figure 7: The Oracle Business Intelligence Applications architecture

These packaged business intelligence applications provide a prebuilt and extensible data warehouse; data extraction routines that provide preconfigured access to Oracle E-Business Suite, SAP, Oracle PeopleSoft, and Oracle Siebel applications; as well as content for use with Oracle Business Intelligence. As these data models and extraction routines are based on industry-standard tools and databases, they can be customized after installation; many customers use them as the starting point for their entire business intelligence solution, using the packaged data model and data extraction routines as the starting point, and then customizing and extending them to cover the full range of their enterprise data.

The use and deployment of packaged solutions is outside the scope of this book, which instead focuses on the Oracle Business Intelligence platform itself and the creation of custom solutions. However, once you are familiar with these topics you should take time to investigate these packaged solutions, which you may wish to deploy along with your custom development in order to bring down the total time, and cost, of your business intelligence deployment.

Oracle Exalytics In-Memory Machine

For customers looking for "speed-of-thought" analysis of very large sets of detail-level data, Oracle Exalytics In-Memory Machine is a combination of hardware and Oracle Business Intelligence software that uses an in-memory database cache and special management tools to manage the cache.

Figure 1-8 shows the architecture for Oracle Exalytics In-Memory Machine, which comes with Oracle Business Intelligence and Oracle Essbase installed on the Exalytics hardware device, along with Oracle TimesTen for Exalytics as the in-memory database. Exalytics is typically used by those customers looking to interactively analyze large volumes of data in a very graphically rich environment and is sold as an "appliance" that you can add to your data center and connect, via InfiniBand to Oracle Exadata or via Ethernet, to your corporate databases and data warehouses.

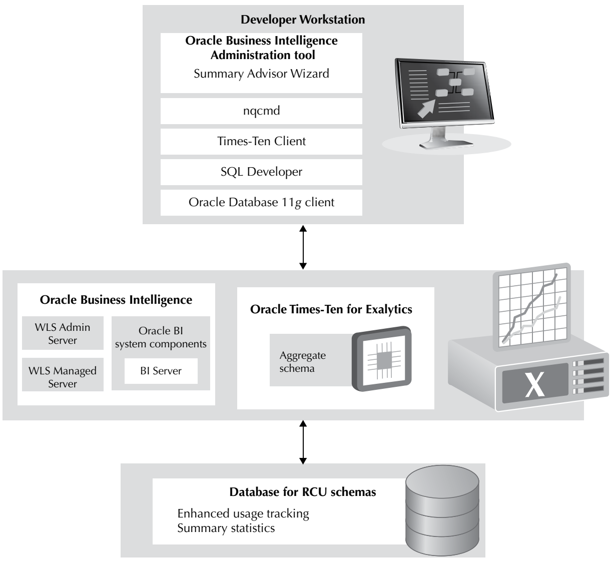


Figure 8: Oracle Exalytics In-Memory Machine architecture

We will look in more detail at Oracle Exalytics In-Memory Machine in the final chapter of this book, including how it is configured, how the in-memory cache is used, and the new visualization types that are enabled by this high-performance platform for your BI system.

What Does Oracle Business Intelligence "Not Do"?

When considering a new software tool, it's worth understanding also what it does "not do." While Oracle Business Intelligence is a suite of products that provides a wide range of analysis tools, data source adapters, and—with the Oracle Business Intelligence Applications—prebuilt content for a wide range of business applications, it is worth understanding what it is not:

- It is not a replacement for Microsoft Excel. Microsoft Excel provides a free-form environment for analyzing and reporting on data, is programmable, and places few restrictions on how data is presented, input, analyzed, and distributed. Oracle Business Intelligence, in contrast, provides a structured analysis environment based around a dimensional model, primarily providing analytical views of data through a web-based dashboard. Individual analyses can, however, be imported into Microsoft Excel, either through an export from the dashboard or through a plug-in to Microsoft Excel, and the BI Server can also act as an ODBC data source for Excel (and other clients).
- Because it uses a logical dimensional model for presenting data to users, it is not generally suitable for querying unstructured data, being better suited to reporting on transactional, data warehouse, and OLAP data sources.
- While it can report against the transactional databases used by applications such as Oracle E-Business Suite, due to the complexity and existing workload on these databases, repositories created directly against them are generally not recommended; instead, data from the transactional databases used by these applications is generally extracted into a data warehouse, or OLAP cube, and then queried from that location by Oracle Business Intelligence.
- Although it can access data in Oracle Essbase or other OLAP servers, Oracle Business Intelligence is not itself a multidimensional OLAP server; instead, it could be thought of as a relational OLAP (ROLAP) server, providing access to a dimensional model through (virtual) relational data structures, themselves mapped to either relational, file or multidimensional physical data structures.

A History of Oracle Business Intelligence

There have been several "business intelligence" products released by Oracle Corporation over the years, and you may have come to this book planning an upgrade from a previous generation of tools such as Oracle Discoverer or Oracle Reports. These tools were developed in-house by Oracle Corporation's developers and were designed to work primarily with Oracle's own database, application server, and security products. As Oracle Corporation moved from being solely focused on database technologies and started to make acquisitions in the middleware, applications, and infrastructure industry sectors, it became clear that it needed a "next-generation" business intelligence platform that did not have such a dependency on Oracle database technologies but that could still take advantage of them if, as is often the case, the customer had used Oracle technology for their data warehouse.

The acquisition of Siebel CRM Systems, Inc., though primarily for their Siebel CRM platform and their extensive customer base, presented Oracle with an opportunity to update their business intelligence platform through another, lesser-known product that Siebel offered; Siebel Analytics. Though well regarded in the industry, Siebel Analytics was not as well known as similar products from vendors such as Business Objects (now part of SAP) and Cognos (now part of IBM), but the Siebel Analytics platform met many of Oracle Corporation's requirements for a next-generation business intelligence platform in that it could access data from many different data sources, had an industry-leading metadata layer (the "semantic model"), very user-friendly dashboards and reporting tools, and also came ready-integrated with popular ERP and CRM systems such as SAP, Oracle E-Business Suite, PeopleSoft—and, of course, Siebel CRM—in a package called Siebel Business Analytics.

Oracle announced in 2005 that what was previously called Siebel Analytics would now be adopted by Oracle Corporation as their strategic business intelligence platform, and they renamed it Oracle Business Intelligence Enterprise Edition. The existing Oracle Discoverer and Oracle Reports tools would be packaged as Oracle Business Intelligence Standard Edition, and while customers would not be forced to upgrade from the older toolset to the new one, in time upgrade tools and services would be made available to make this process easier for those customers who chose to do so.

Siebel Analytics was itself, though, developed outside of Siebel CRM Systems, Inc., and was in fact originally developed by a technology startup out of Minneapolis, MN, called nQuire. Led by Larry Barbetta and a number of ex-Platinum Software engineers and product managers, nQuire released the nQuire Server, the predecessor to what eventually became the Oracle BI Server, back in the late 1990's as a stand-alone analytics and search server that featured connectivity to a wide range of data sources. The nQuire Query Server featured a metadata model (which eventually became the semantic model) that provided a virtual logical dimensional model over these data sources, and over time the nQuire Query Server was joined by nQuire Answers and nQuire Delivers, giving us the core of what is now Oracle Business Intelligence. nQuire was itself acquired by Siebel CRM Systems, Inc., in October 2001, and it developed the product further and licensed data models and data extraction routines from Informatica Corporation that now form the core of the Oracle Business Intelligence Applications. So while what you know as Oracle Business Intelligence may be a product that is only a few years old, the core of the product itself can be traced back to groundbreaking work done by the nQuire team back in the mid-1990's.

Oracle Product Release History

Shortly after Oracle Corporation acquired Siebel Systems, what was Siebel Analytics was renamed Oracle Business Intelligence Enterprise Edition, whilst Siebel Business Analytics was renamed Oracle BI Applications. The initial release of Oracle Business Intelligence was the 10g 10.1.3.2 version, with subsequent major releases of Oracle Business within the 10*g* timeline:

- **Oracle Business Intelligence 10.1.3.2** First "Oracle-branded" release of Oracle Business Intelligence; introduced Oracle BI Publisher as a replacement for Actuate, a new Oracle "look and feel" with 64-bit support; time-series functions and features for multiuser development.
- **Oracle Business Intelligence 10.1.3.3.x** MS Office integration, support for metadata import through the Oracle Call Interface; support for embedded database functions; initial support for Oracle Essbase as a data source.
- **Oracle Business Intelligence 10.1.3.4.x** Integration with Hyperion Workspace; integration with Oracle Smart View and Oracle Smart Space; the introduction of a utility to upgrade Oracle Discoverer End-User Layers to Oracle Business Intelligence repositories.

In addition, at the time of writing this book there have been three major releases as part of the 11*g* Release 1 timeline:

- **Oracle Business Intelligence 11.1.1.3** Initial 11*g* release, provided new "look and feel," support for KPIs and scorecards, the Action Framework, and other new features. Platform support limited to Microsoft Windows, Linux, and IBM AIX, with Oracle WebLogic Server as the sole JEE (Java Platform, Enterprise Edition) application server.
- **Oracle Business Intelligence 11.1.1.5** Oracle Extension of platform support to HP/UX and Sun Solaris, introduction of iOS (Apple iPhone, Apple iPad) native clients, and restoration of data sources temporarily desupported in the 11.1.1.3 release.
- **Oracle Business Intelligence 11.1.1.6** Support for the Exalytics In-Memory Machine platform and integration with version control tools. New visualization options and new certified data sources, including Oracle TimesTen.

In addition, there are three editions of Oracle Business Intelligence, the first two of which in the following list are within the scope of this book, and the third that is not:

- **Oracle Business Intelligence Enterprise Edition** The full set of business intelligence tools and servers. This book will concern itself primarily with this edition.
- **Oracle Business Intelligence Standard Edition** One "Departmental" or budget version of Oracle Business Intelligence that comes with certain restrictions on, for example, the number of allowable users and CPUs. Check with your Oracle representative or **http://www.oracle.com** for up-to-date details on this and other product packages.
- **Oracle Business Intelligence Standard Edition** Somewhat confusingly, a different family of products altogether. This is a container for "legacy" Oracle Business Intelligence tools such as Oracle Discoverer and Oracle Reports. You can use upgrade tools to migrate Discoverer metadata to Oracle Business Intelligence Enterprise Edition and Standard Edition One; but, otherwise, this edition is outside the scope of this book.

Now that you know a little more about Oracle Business Intelligence's background and a little of its history, let's take a look in more detail at the individual products within the platform, its architecture, and how it works "under the covers."

Oracle Business Intelligence 11*g* Release 1 Architecture

Earlier in this chapter, we looked how the "heart" of Oracle Business Intelligence is the Oracle BI Server and the Oracle BI Presentation Server. The Oracle BI Server provides native, federated access to data sources, together with security, calculations, and data navigation. The Oracle BI Presentation Server connects to the BI Server to obtain data, which it presents to users in the form of analyses, reports, and dashboards.

In addition to these two servers, there are three other servers that work with them to provide core Oracle Business Intelligence functionality:

- **Oracle BI Cluster Controller** This server provides a central point of access for the Oracle BI Presentation Server when two or more BI Servers are working together in a cluster, together with load-balancing, failover, and other cluster services.
- **Oracle BI Java Host** This server works alongside the BI Presentation Server to provide connectivity to Java tasks and the Java-based Oracle BI Publisher, as well as to support chart generation.
- **Oracle BI Scheduler** This server is used to schedule and automate the production and distribution of analyses, as well as to automate workflow tasks based around business intelligence functionality.

These three servers, together with the Oracle BI Server and the Oracle BI Presentation Server, are known in Oracle Business Intelligence 11*g* Release 1 terminology as system components, and they run as services and servers directly on the host platform. They are operating system executables written in C-based languages.

To create the link between an end user's web browser and the dashboards, analyses, and reports provided by the Oracle BI Presentation Server, a Java application called the Oracle BI Analytics Plug-In runs in a Java application server and routes incoming requests through to the BI Presentation Server. (Currently, only Oracle WebLogic Server is supported, but later releases of Oracle Business Intelligence should support other application servers.) A simplified schematic of the Oracle Business Intelligence 11*g* system components, together with the Oracle BI Plug-In, is shown in Figure 1-9.

This basic, internal architecture has stayed consistent since the days of nQuire and Siebel Analytics and is still at the core of the 11*g* Release of Oracle Business Intelligence. It, together with two additional Java server applications for publishing reports and connecting to Microsoft office, is largely the architecture of the 10g release of Oracle Business Intelligence and would run fairly comfortably on a smaller server, desktop computer, or laptop. Because the product has been adopted within Oracle Corporation as their strategic business intelligence platform and, in particular, because it has been integrated over time into their wider Oracle Fusion Middleware platform due to customer requirements for BI to integrate into wider business processes and applications, these core components have been built on and enhanced with additional Java

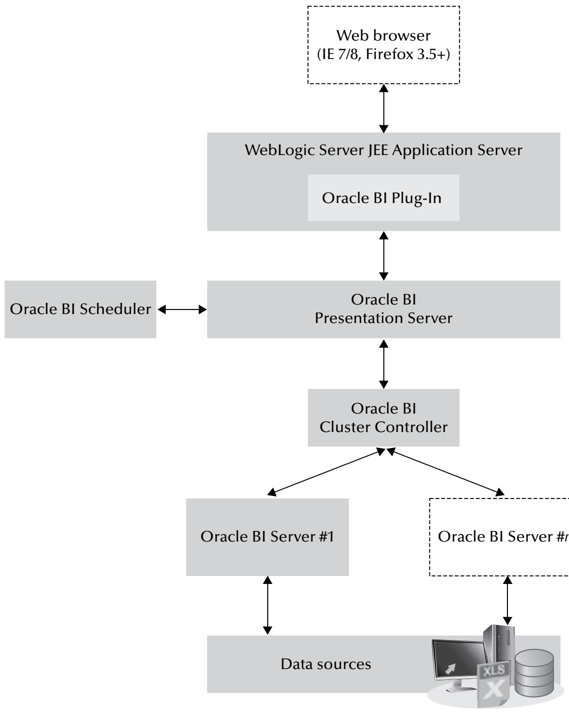


Figure 9: Oracle Business Intelligence system components schematic

components to form the more complete architecture used in the 11*g* Release 1 version.

Oracle Business Intelligence 11*g* and Oracle Fusion Middleware

While the core components within Oracle Business Intelligence remain the Oracle BI Server and Oracle BI Presentation Server, supported by the Oracle BI Scheduler, Oracle Java Host, and Oracle BI Cluster Controller, these have been supplemented in the 11*g* Release 1 release by Java-based Oracle Fusion Middleware technologies based around the Oracle WebLogic Server application server (with plans to extend this to other non-WebLogic application servers in future releases). While the previous, 10g release of Oracle Business Intelligence made limited use of application server technology to, for example, host the Oracle BI Plug-In and Java-based applications such as Oracle BI Publisher and Oracle BI Office, the 11*g* release of Oracle Business Intelligence leverages Fusion Middleware and WebLogic technologies in areas such as the following:

- *Security and authentication,* which are now delegated to Oracle Fusion Middleware 11g, with users and groups now held, by default, in the WebLogic Server LDAP directory rather than the Oracle BI Repository
- *Systems administration,* now centralized using Oracle Enterprise Manager Fusion Middleware Control
- *Connectivity* to outside applications and processes through the Java-based Action Service, with security and credentials handled via Fusion Middleware's credential and policy stores
- *Administration scripting* using the Oracle WebLogic Scripting Tool and JMX MBeans (Java managed beans that provide the core administration functionality be-

(continued on page 26)

amazon web services

Innovation. Powered by the AWS Cloud.

Low Cost

Open & Flexible

Instant Elasticity

Secure

Managing Oracle databases made simple.
aws.amazon.com/rds/oracle

Ignite8

Turn Up the Heat on Oracle Performance

Database Monitoring and Performance Analysis

Only Ignite delivers a complete picture of database performance from an end user perspective. It analyzes response time, queries, sessions, and server resources to show both historical and real-time performance.

Go to www.Confio.com and try Ignite 8 today!

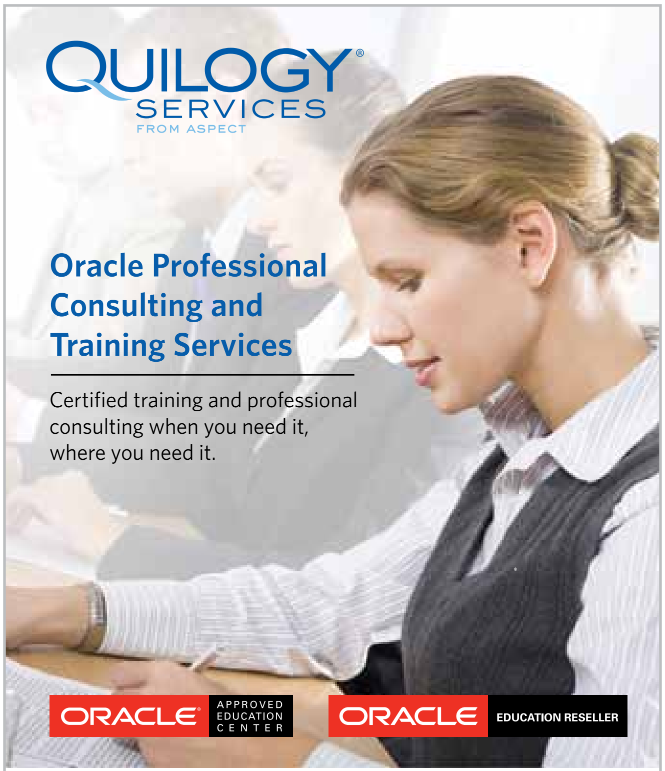
CONFIO
Confio Software, Boulder Colorado.

DELPHIX

Database Virtualization Software

- Consolidate Infrastructure.
- Instantly Provision and Refresh.
- Maximize Performance.

QUILOGY SERVICES
FROM ASPECT



Oracle Professional Consulting and Training Services

Certified training and professional consulting when you need it, where you need it.

ORACLE APPROVED EDUCATION CENTER

ORACLE EDUCATION RESELLER

www.quilogyservices.com
education@aspect.com
866.784.5649

Aspect

Many Thanks to Our Sponsors

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low-cost memberships. If your company is able to offer sponsorship at any level, please contact NoCOUG's president, Iggy Fernandez, at **iggy_fernandez@hotmail.com**. ▲

Long-term event sponsorship:

CHEVRON

ORACLE CORP.

Thank you! Year 2012 Gold Vendors:

- ➤ Confio Software
- ➤ Database Specialists
- ➤ Delphix
- ➤ GridIron Systems
- ➤ Quilogy Services

For information about our Gold Vendor Program, contact the NoCOUG vendor coordinator via email at: **vendor_coordinator@nocoug.org**.

TREASURER'S REPORT

Naren Nagtode, *Treasurer*

Beginning Balance		
June 1, 2012		**$ 64,888.40**
Revenue		
Membership Dues	1,404.91	
Meeting Fees	725.00	
Vendor Receipts	4,050.00	
Advertising Fee	–	
Training Day Fees	2,100.00	
Conference Sponsorship	–	
Interest	2.62	
Paypal balance	–	
Total Revenue		**$ 8,282.53**
Expenses		
Regional Meeting	7,620.90	
Journal	2,710.93	
Membership	68.93	
Administration	1,708.16	
Board Meeting	409.43	
Marketing	–	
Insurance	–	
Vendor expenses	77.50	
Membership S/W subscription	–	
Training Day	–	
IOUG-Rep	–	
Server and S/W Fees	–	
Bank Service Charges	–	
Total Expenses		**$ 12,595.85**
Ending Balance		
September 30, 2012		**$ 60,575.08**

NEED TO APPLY DATA QUALITY TO ORACLE DATA?

DBMoto® can help!

HiT® SOFTWARE

A BackOffice Associates, LLC Company

DBMoto® — Data Replication and Change Data Capture

DBMoto® is the best independent data replication and data integration solution available today for real-time Change Data Capture across multiple databases. No consulting required — data replication functions, mapping and verification are all available through prebuilt graphical interfaces!

- Use DBMoto Data Replication and Change Data Capture to send changes to a file to be shared with your data quality, data auditing or data analysis programs
- Connect data change and auditing information to your Business Intelligence/Reporting applications through low cost data marts
- Use DBMoto's APIs to build Data Replication and Change Data Capture into your system or architecture

DOWNLOAD FREE PAPER:

"Five Ways Change Data Capture Can Help Improve Data Quality"

http://info.hitsw.com/NOCOUG4

DATABASES SUPPORTED:

Oracle (v.9 and above) · MySQL · Microsoft SQL Server · IBM DB2 (all versions) · Netezza · Informix · Sybase ASE · SQL Anywhere · Ingres · PostgreSQL

T +1.408.345.4001 **www.hitsw.com** **info@hitsw.com**

ORACLE® Gold Partner

(continued from page 22)

hind Oracle Enterprise Manager Fusion Middleware Control)

- *Process management* of the system components (BI Server, BI Presentation Server, and so on) through the Oracle Process Manager and Notification Server
- *High availability* through clustering of WebLogic Managed Servers within a WebLogic domain

Figure 1-10 shows the logical architecture for Oracle Business Intelligence 11*g* Release 1, which together is called an Oracle BI domain.

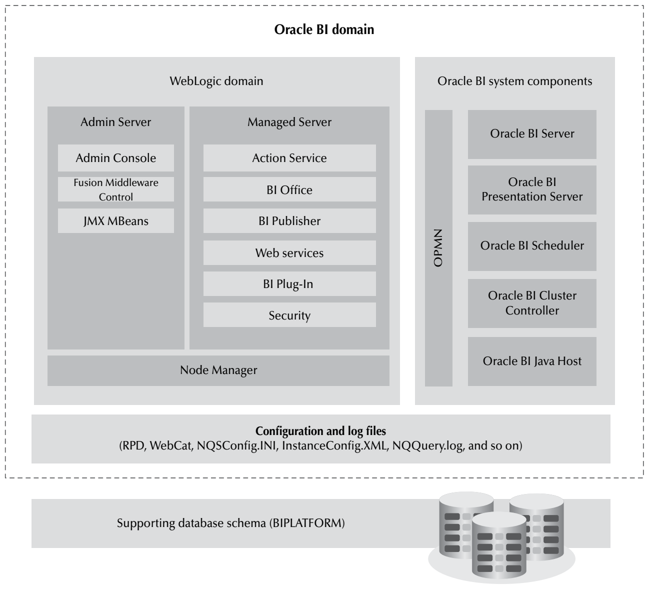

Oracle BI domain
WebLogic domain
Admin Server
Admin Console
Fusion Middleware Control
JMX MBeans
Managed Server
Action Service
BI Office
BI Publisher
Web services
BI Plug-In
Security
Node Manager
Oracle BI system components
OPMN
Oracle BI Server
Oracle BI Presentation Server
Oracle BI Scheduler
Oracle BI Cluster Controller
Oracle BI Java Host
Configuration and log files
(RPD, WebCat, NQSConfig.INI, InstanceConfig.XML, NQQuery.log, and so on)
Supporting database schema (BIPLATFORM)

Figure 10: The Oracle Business Intelligence 11g logical architecture

This logical architecture is made up of a number of components:

- **Oracle BI domain** The complete set of Java and non-Java components that make up a single Oracle Business Intelligence environment
- **WebLogic domain** Houses the Java components within the architecture
- **WebLogic Server Admin Server** A JEE container (server) that contains a dedicated Java Virtual Machine used for monitoring and managing the system
- **WebLogic Server Managed Server** Another dedicated JEE container that, in this case, is used to house the Java applications used by Oracle Business Intelligence
- **Java components** Java applications such as the Oracle BI Analytics Plug-In, Oracle BI Publisher, the Action Service, and Oracle BI Office that work alongside the traditional system components described previously
- **System components** The term now used for the original BI Server, BI Presentation Server, and other servers used in the core Oracle Business Intelligence architecture
- **WebLogic Server Administration Console** An application that runs in the Admin Server and is used for controlling WebLogic Server
- **Oracle Enterprise Manager Fusion Middleware Control** For managing the Oracle Business Intelligence system across a single or multiple cluster nodes
- **Supporting database schemas** Created using the Oracle Fusion Middleware Repository Creation Utility, which contains relational tables used for storing additional Oracle Business Intelligence metadata (BIPLATFORM) and metadata used by Oracle Fusion Middleware's Metadata Services (MDS)

Note that it is possible to install Oracle Business Intelligence in a "simple" architecture configuration that does away with the WebLogic Managed Server and Node Manager, to reduce the software and memory footprint for laptop-style installations. This type of installation does, however, restrict you in how you can "scale out" your installation afterward, so it should only be used for demo or proof-of-concept scenarios. Chapter 2 details the various installation options and what these mean in terms of installed components on your server.

This distinction between Java components that are written in Java and managed by WebLogic Server, and System Components that run as operating system services and are written in C++, is due to the history of Oracle Business Intelligence. The core server components (system components, in 11*g* terminology) were written in C++, but most new development of distinct new functionality carried out by Oracle Corporation has been carried out in Java as part of Oracle Fusion Middleware. Rather than convert all of the legacy servers to Java applications from C++, however, these have been "lifted" into the new 11*g* architecture but allowed to run stand-alone, outside of the Oracle WebLogic Server domain structure, and are now called system components. This approach is actually common to many Oracle Fusion Middleware products that need to combine long-standing C-based server products with more recent, Java-based applications, and allows both sets of products to benefit from the same management and integration infrastructure based around Oracle Enterprise Manager Fusion Middleware Control and Oracle WebLogic Server.

While the distinction between system components and Java components becomes obvious to experienced Oracle Business Intelligence developers, both types of components are managed at a system level by the same application: Oracle Enterprise Manager Fusion Middleware Control. While each component and type of component has its own unique configuration tools, Fusion Middleware Control provides a centralized, web-based console for controlling the business intelligence system.

In addition to the Java and system components, two additional servers are used to control and monitor the running of these component categories:

- **WebLogic Server Node Manager** This server is used to start, stop, restart, and monitor the Managed Server, or multiple Managed Servers if WebLogic Server is running clustered (not available for "Simple" install types, which do not need to stop and start any Managed Servers).
- **Oracle Process Manager and Notification Server** (OPMN) This server performs a similar role for the system components—stopping, starting, and restarting them, and reporting their status to monitoring applications. ▲

Database Specialists: DBA Pro Service

DBA PRO BENEFITS

- *Cost-effective and flexible extension of your IT team*
- *Proactive database maintenance and quick resolution of problems by Oracle experts*
- *Increased database uptime*
- *Improved database performance*
- *Constant database monitoring with Database Rx*
- *Onsite and offsite flexibility*
- *Reliable support from a stable team of DBAs familiar with your databases*

CUSTOMIZABLE SERVICE PLANS FOR ORACLE SYSTEMS

Keeping your Oracle database systems highly available takes knowledge, skill, and experience. It also takes knowing that each environment is different. From large companies that need additional DBA support and specialized expertise to small companies that don't require a full-time onsite DBA, flexibility is the key. That's why Database Specialists offers a flexible service called DBA Pro. With DBA Pro, we work with you to configure a program that best suits your needs and helps you deal with any Oracle issues that arise. You receive cost-effective basic services for development systems and more comprehensive plans for production and mission-critical Oracle systems.

DBA Pro's mix and match service components

Access to experienced senior Oracle expertise when you need it
We work as an extension of your team to set up and manage your Oracle databases to maintain reliability, scalability, and peak performance. When you become a DBA Pro client, you are assigned a primary and secondary Database Specialists DBA. They'll become intimately familiar with your systems. When you need us, just call our toll-free number or send email for assistance from an experienced DBA during regular business hours. If you need a fuller range of coverage with guaranteed response times, you may choose our 24 x 7 option.

24 x 7 availability with guaranteed response time
For managing mission-critical systems, no service is more valuable than being able to call on a team of experts to solve a database problem quickly and efficiently. You may call in an emergency request for help at any time, knowing your call will be answered by a Database Specialists DBA within a guaranteed response time.

Daily review and recommendations for database care
A Database Specialists DBA will perform a daily review of activity and alerts on your Oracle database. This aids in a proactive approach to managing your database systems. After each review, you receive personalized recommendations, comments, and action items via email. This information is stored in the Database Rx Performance Portal for future reference.

Monthly review and report
Looking at trends and focusing on performance, availability, and stability are critical over time. Each month, a Database Specialists DBA will review activity and alerts on your Oracle database and prepare a comprehensive report for you.

Proactive maintenance
When you want Database Specialists to handle ongoing proactive maintenance, we can automatically access your database remotely and address issues directly — if the maintenance procedure is one you have pre-authorized us to perform. You can rest assured knowing your Oracle systems are in good hands.

Onsite and offsite flexibility
You may choose to have Database Specialists consultants work onsite so they can work closely with your own DBA staff, or you may bring us onsite only for specific projects. Or you may choose to save money on travel time and infrastructure setup by having work done remotely. With DBA Pro we provide the most appropriate service program for you.

DatabaseSpecialists

CALL 1-888-648-0500 TO DISCUSS A SERVICE PLAN

NoCOUG
P.O. Box 3282
Danville, CA 94526

NoCOUG Fall Conference Schedule

Thursday, November 15, 2012—CarrAmerica Conference Center, Pleasanton, CA

Please visit **http://www.nocoug.org** for updates and directions, and to submit your RSVP.
Cost: $50 admission fee for non-members. Members free. Includes lunch voucher.

8:00 a.m.–9:00	Registration and Continental Breakfast—Refreshments served
9:00–9:30	**Welcome:** Iggy Fernandez, NoCOUG president
9:30–10:30	**Keynote:** *Tom's Top Twelve Things About the Latest Generation of Database Technology* —Tom Kyte, Oracle Corporation
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Auditorium: *Five SQL and PL/SQL Things in the Latest Generation of Database Technology* —Tom Kyte, Oracle Corporation
	Tassajara: *Fine Tune Oracle Execution Plans for Performance Gains*—Janis Griffin, Confio
	Diablo: *Scaling to Infinity: Partitioning Data Warehouses on Oracle* —Tim Gorman, Evergreen Database Technologies
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Auditorium: *Harnessing the Power of Optimizer Hints*—Maria Colgan, Oracle Corporation
	Tassajara: *Internals of Active DataGuard*—Saibabu Devabhaktuni, PayPal
	Diablo: *MySQL 5.6: The Next Generation Database*—Lynn Ferrante, Oracle Corporation
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Auditorium: *Scaling to Infinity: Making Star Transformations Sing* —Tim Gorman, Evergreen Database Technologies
	Tassajara: *Sherlock Holmes for the DBA*—Kellyn Pot'Vin, Enkitec
	Diablo: *The Hitchhiker's Guide to the Cloud for the DBA*—Ben Prusinski, VCE
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Auditorium: *Database Virtualization and Flexible Cloning*—Kyle Hailey, Delphix
	Tassajara: *Database Protection Best Practices for Oracle Database*—Gurmeet Goindi, Oracle Corporation
	Diablo: *TBD*
5:00–	**NoCOUG Networking and No-Host Happy Hour**

RSVP *required* at http://www.nocoug.org

The *NoCOUG Journal* design and production: Giraffex, Inc., S.F.
Front cover photo: Bigleaf Maple (Acer Macrophyllum) in Mt. Diablo State Park, CA by Dave Abercrombie.