An Oracle White Paper
November 2011

Oracle OLAP Exadata X2-2 Performance Demonstration

Executive Overview

This paper describes a performance demonstration of the OLAP Option to the Oracle Database running on an X2-2 Exadata Database Machine half rack. It shows how Oracle OLAP cubes can be used to enhance the performance and analytic content of the data warehouse and business intelligence solutions, supporting a demanding user community with ultrafast query and rich analytic content.

The demonstration represents users of a business intelligence application using SQL to query an Oracle OLAP cube that has been enhanced with a variety of analytic measures. The cube contains data loaded from a fact table with more than 1 billion rows.

Utilizing Exadata features such as Smart Flash Cache, Oracle Database supported a community of 50 concurrent users querying the cube with queries that are typical of those executed from a business intelligence tool such as Oracle Business Intelligence Enterprise Edition.

Calendar Year	Calendar Quarter	All Products	Department	All Regions	All Channels	Sales	Sales Year Ago	Sales Change Year Ago	Sales Pct Change Year Ago	Sales YTD	Sales YTD Year Ago	Sales YTD Chg Year Ago	Sales YTD Pct Chg Year Ago
CY2009	Q3-CY2009	All Products	Cameras and Camcorders	All Regions	All Channels	3,192,047	2,463,581	728,466	30%	8,853,105	7,758,539	1,094,566	30%
			Computers	All Regions	All Channels	30,982,913	28,279,706	2,703,207	10%	93,341,138	80,544,054	12,797,085	10%
			Portable Music and Video	All Regions	All Channels	4,313,055	3,961,159	351,896	9%	12,995,844	11,882,202	1,113,642	9%

Figure 1. A sample query from the performance demonstration. This report includes one stored measure (sales) and seven calculated measures.

With each user querying the database non-stop (without waits between queries) with median query times ranged from .03 to .58 seconds, average query times ranged from .26 to 2.32 seconds, and 95 percent of queries returned in 1.5 to 5.5 seconds, depending on the type of query.

Query performance can be attributed to highly optimized data types and Exadata Smart Flash Cache. Cubes are designed for fast access to random data points, using features such as array-based storage, cost-based aggregation, and joined cube scans. Exadata Smart Flash Cache contributes significantly to cube query performance, virtually eliminating IO wait for the high volume, random IO typically seen with cube queries.

Introduction

According to the TDWI Next Generation Data Warehouse Platforms Report[1] , the two leading issues driving organizations to upgrade their data warehouse platform are query performance and the inability to support advance analytics[2]. The OracleExadata line of engineered systems, with tremendous computing capacity and advanced database features, accelerates performance of all workloads by at least 10x as compared to non-Exadata systems.

Exadata combines the raw power of a well balanced machine with many CPU cores, large amounts of memory, and InfiniBand fabric with smart database features such as Exadata Smart Scan, Exadata Smart Flash Cache, and Exadata Hybrid Columnar Compression. Combined, this power and these features dramatically improve query performance. Oracle also supports a wide variety of analytic features including window functions, statistical functions, in-database data mining and spatial intelligence.

The Oracle OLAP Option is a full featured on-line analytical processing engine embedded within Oracle Database Enterprise Edition. It includes a rich multidimensional business model, analytic expressions, ?and multidimensional data types (cubes and dimensions) that are highly optimized for the management of aggregate data and fast, ad-hoc query. Oracle cubes and dimensions can be queried using SQL, PL/SQL and an available MDX provider[3], making cubes easily accessible using a large variety of business intelligence tools and Microsoft Excel.

Cubes rely on optimized data storage, advanced compression, cost-based aggregation and smart query execution for accelerated query performance. This allows cubes have the ability to provide fast query with minimal use of system resources, letting cubes to support large numbers of concurrent users with maximum performance.

Combining the raw power of Exadata with smart cube-based data storage and query allows organizations to extend the analytic content of the data warehouse and provide business users with high performance queries, at times several times faster than existing Exadata solutions.

---

[1] Russom, Philip. *Next Generation Data Warehouse Platforms*. TDWI, 2009. Web. 14 Nov. 2011. <http://www.oracle.com/us/solutions/datawarehousing/040119.pdf>.

[2] 45% of respondents indicated that "Poor query response" is a top driver. 40% indicated that "Can't support advanced analytics" is a top driver.

[3] MDX Provider for Oracle OLAP available from Simba Technologies. See <http://simba.com/MDX-Provider-for-Oracle-OLAP.htm>.

Overview

This performance demonstration is designed to represent a community of business users querying an analytically rich data set using a SQL-based business intelligence tool such as Oracle Business Intelligence Enterprise Edition. The demonstration measures the amount to time to build a new cube and query performance.

This demonstration is based on a retail sales reporting solution. Data are staged in a star schema (tables) and loaded into OLAP dimensions and a cube. The model includes four years of sales data that varies by day, customer, item and store, with hierarchies aggregating data to summary levels such as year, region, department and channel type.

Business users have the requirements of fast query and derived metrics including time series calculations (for example, year to date aggregations and current versus prior period comparisons). Many business users will query the solution simultaneously.

The following sections describe the data model, data sources, queries, and how the demonstration was executed.

Database Server

The performance demonstration was executed on an Oracle Exadata X2-2 Database Machine "half rack" with Oracle 11.2.0.2 (BP11). A half-rack system includes 4 database nodes, each with 12 CPU cores and 96GB memory, connecting to Oracle Exadata Storage Servers with Exadata Smart Flash Cache.

Data Model

The Oracle cube is designed from the perspective of a business model (that is, how data is presented to the business user). In this demonstration, there are four dimensions – Time, Product, Customer and Channel – and one cube. Within each dimension, data are organized by level of summarization into hierarchies. A member represents one distinct value of a dimension (that is, members are the primary key of a dimension).

Dimensions, hierarchies and levels used in this data model are described Table 1.

TABLE 1 – DIMENSIONS AND HIERARCHIES

DIMENSION	HIERARCHY	LEVELS
Time	Calendar	All Years (1 member) Calendar Year (4) Calendar Quarter (16) Month (48) Day (1,461)

Product	Departments	All Products (1 member) Department (4) Category (8) Type (32) Subtype (146) Item (2,713)
Customer	Regions	All Regions (1 member) Region level (6) Country level (64) State/Province level (181) City level (32,032) Customer level (~2,200,000)
Channel	Classes	All Channels (1 member) Class (2) Channel (11)

The Geographic Sales cube is dimensioned by Time, Product, Customer, and Channel and contains the measures listed in Table 2. Base measures are loaded from tables and stored in the cube. The base measures are aggregated within the cube using SUM. Derived measures are expressions that are calculated at runtime by the cube when queried.

TABLE 2. MEASURES IN THE GEOGRAPHIC SALES CUBE

MEASURE NAME	DESCRIPTION	MEASURE TYPE
SALES	Sales	BASE
UNITS	Units	BASE
SALES_CHG_YR_AGO	Sales Change from Year Ago	DERIVED
SALES_PCTCHG_YR_AGO	Sales Percent Change from Year Ago	DERIVED
SALES_RANK_WITHIN_PARENT	Sales Rank Within Parent	DERIVED
SALES_YTD	Sales YTD	DERIVED
SALES_YTD_CHG_YR_AGO	Sales YTD Change from Year Ago	DERIVED
SALES_YTD_PCTCHG_YR_AGO	Sales YTD Percent Change from Year Ago	DERIVED
SALES_YTD_YR_AGO	Sales YTD Year Ago	DERIVED
SALES_YR_AGO	Sales Year Ago	DERIVED

Data Sources

The database supports loading data into a cube from tables or views or generating data internally within the cube (for example, generating a forecast). In this demonstration, data are loaded from tables into dimensions and cubes. All dimension members and base measures are loaded from tables.

The source tables in this demonstration are in the form of a star schema, with one dimension table for each OLAP dimension and a fact table for the sales and units data. Data in the fact table is at the day, item, customer and channel (store) levels. Data is loaded into the same levels of the cube. The UNITS_FACT table contains approximately 1,000,000,000 unique rows.

Data in the CUSTOMER_DIM and UNITS_FACT tables are programmatically generated with approximately 2.2 million customers and 1 billion fact rows distributed evenly over a four year period. One new customer was created for each of the first 100,000 fact records. After the first 100,000 fact records, a new customer might be created or an existing customer might be reused.

Queries

Oracle OLAP cubes and dimensions are represented to SQL through a collection of dimension, hierarchy, and cube views in the form of a star schema. These views flatten the multidimensional cube structures into columns and rows, which can then be easily queried using SQL.

Queries are programmatically generated and stored in a table prior to query execution. This allows the same exact query test to be run multiple times using different query execution methods and different cube designs.

A stream of queries represents an interactive business user session. For each user, 80 queries are generated. The queries simulate a user displaying a summary level report in a dashboard application and then exploring data by drilling eight times to deeper levels of detail. This process is repeated five times for each of two different types of queries. Half the queries select only stored measures and half select both stored and calculated (derived) measures.

With the exception of the first query (that is, the first drill on the dashboard report), dimensions and dimension members are selected at random. On the first query, the time dimension is more likely to be selected than other dimensions. This enforces a balance between reports that tend to report over all four years and reports that query lower time periods.

The summary level dashboard report is not included in query timings. The following illustrations represent the query process.

Calendar Year	All Products	All Regions	All Channels	Sales	Sales Year Ago	Sales Change Year Ago	Sales Pct Change Year Ago	Sales YTD	Sales YTD Year Ago	Sales YTD Chg Year Ago	Sales YTD Pct Chg Year Ago
CY2007	All Products	All Regions	All Channels	120,335,758				120,335,758			
CY2008	All Products	All Regions	All Channels	138,960,159	120,335,758	18,624,401	15%	138,960,159	120,335,758	18,624,401	15%
CY2009	All Products	All Regions	All Channels	158,219,099	138,960,159	19,258,940	14%	158,219,099	138,960,159	19,258,940	14%
CY2010	All Products	All Regions	All Channels		158,219,099				158,219,099		

Figure 2. Summary level dashboard report (not included in query timings)

Calendar Year	Calendar Quarter	All Products	All Regions	All Channels	Sales	Sales Year Ago	Sales Change Year Ago	Sales Pct Change Year Ago	Sales YTD	Sales YTD Year Ago	Sales YTD Chg Year Ago	Sales YTD Pct Chg Year Ago
CY2009	Q1-CY2009	All Products	All Regions	All Channels	41,431,742	35,293,342	6,138,400	17%	41,431,742	35,293,342	6,138,400	17%
	Q2-CY2009	All Products	All Regions	All Channels	35,270,330	30,187,007	5,083,323	17%	76,702,072	65,480,349	11,221,723	17%
	Q3-CY2009	All Products	All Regions	All Channels	38,488,015	34,704,446	3,783,569	11%	115,190,087	100,184,795	15,005,292	11%
	Q4-CY2009	All Products	All Regions	All Channels	43,029,012	38,775,364	4,253,648	11%	158,219,099	138,960,159	19,258,940	11%

Figure 3. Drill 1, on Calendar Year CY2009.

Calendar Year	Calendar Quarter	All Products	All Regions	Region	All Channels	Sales	Sales Year Ago	Sales Change Year Ago	Sales Pct Change Year Ago	Sales YTD	Sales YTD Year Ago	Sales YTD Chg Year Ago	Sales YTD Pct Chg Year Ago
CY2009	Q3-CY2009	All Products	All Regions	Africa	All Channels	1,719,858	1,520,290	199,568	13%	5,046,219	4,549,285	496,934	13%
				Asia	All Channels	17,092,968	15,564,796	1,528,172	10%	50,784,102	44,627,447	6,156,655	10%
				Europe	All Channels	5,699,385	5,228,459	470,926	9%	17,267,877	15,187,250	2,080,627	9%
				North America	All Channels	9,677,541	8,666,862	1,010,678	12%	29,185,102	24,911,865	4,273,237	12%
				Oceania	All Channels	34,107	24,401	9,706	40%	75,965	77,545	-1,580	40%
				South America	All Channels	4,264,157	3,699,637	564,520	15%	12,830,822	10,831,403	1,999,419	15%

Figure 4. Drill 2, All Regions in Q3-CY2009. The drilling process is repeated for 8 queries.

Query Execution

Queries are executed programmatically on the server using either a PL/SQL program or by running queries from a script using SQL*Plus. The PL/SQL program method captures detailed performance information in log tables and represent the official timings reported in this paper. The SQL*Plus method cross-checks the query timings for consistency.

Analytic Workspace Attach Methods

Applications can explicitly attach to an analytic workspace for query (a "hard attach") or the analytic workspace can be attached and detached automatically as part of query processing (a "soft attach"). When a hard attach is used, query processing is simplified and the query will run a little more quickly.

Any application that can issue a PL/SQL command once prior to querying the cube can explicitly attach to an analytic workspace. This might be done, for example, by a batch reporting system that makes a single connection and creates many reports. Applications such as Oracle Business Intelligence Enterprise Edition (OBIEE) typically do not explicitly attach to an analytic workspace at the beginning of a session, however OBIEE can issue a hard attach in a connection script.

Both attach methods were used in this performance demonstration. Results for each attach method are reported separately.

Exadata Smart Flash Cache and DB Cache

Exadata Smart Flash Cache is ideal for the small, random I/O that is typical of cube queries and was used for all query tests. In previous demonstrations, Exadata Smart Flash Cache was proven to

significantly improve median and average query performance (making fast queries even faster), but had the greatest impact on the most difficult, longest running queries (which tend to require more reads).

All queries were run immediately after a full build of the cube when the database cache was cold. Running queries against a warm cache would tend to improve query times, particularly for longer running queries.

Query Performance Observations

Query performance is summarized in Table 3. Results are presented for both hard and soft attach modes. The the number of rows returned by reports ranged from 0 to 59,097 with the average being 171. The average number of rows returned for the 12 queries running 60 seconds or longer was greater than 17,000.

TABLE 3. CUBE QUERY PERFORMANCE

	ATTACH TYPE	ALL QUERY TYPES	STORED MEASURES	TIME SERIES MEASURES
Concurrent users (no wait time between queries)		50	50	50
Total queries executed		4,000	2,000	2,000
Median query time (seconds per query)	Soft	.49	.41	.58
	Hard	.05	.03	.08
Average query time (seconds per query)	Soft	1.54	.76	2.32
	Hard	.91	.31	1.51
95% of queries return in *n* seconds or less	Soft	3.5	2.5	5.5
	Hard	1.5	1.5	1.5
Queries running longer than 60 seconds	Soft	12 (0.3%)	0 (0.0%)	12 (0.6%)
	Hard	8 (0.2%)	0 (0.0%)	8 (0.4%)

User Scalability

When interpreting query performance and user scalability, it is important to understand that Oracle cubes achieve fast query performance using cost-based cube aggregation, smart query execution and resolution of calculations within the cube. Cubes do not rely on parallel query: all queries are executed in a single process.

As a general rule, maximum query performance can be maintained at least until the number of concurrent users equals the number of available CPU cores, providing that other resources such as memory and IO do not become constraining factors. The Exadata X2-2 Database Machine with Smart Flash Cache, ample memory, and fast disk allowed the cube to maintain maximum query performance

under heavy load. Smart Flash Cache plays a very important role in query performance by reducing or eliminating IO wait during query.

Also consider that all 50 users started at the same time and that there were no waits between queries. While this is convenient for the execution of a performance demonstration because it places a high load on the server, actual business user behavior will introduce think time between many queries. A user might execute several queries quickly as they drill to an area of interest and them spend a some of time viewing the report. As such, 50 concurrently users is in this demonstration likely represents a considerably larger user community.

Designing and Building Cubes

Cubes and dimensions were designed using only Analytic Workspace Manager and well-known best practices. The cube was built (processed) using the DBMS_CUBE.BUILD procedure, the standard method for maintaining cubes and dimensions. The default LOAD_AND_AGGREGATE cube script was used to load data from relational tables into the cube and to partially pre-aggregate data within the cube.

See Appendix A for more detailed information about cube build processing.

Cube Build Performance Observations

To measure build performance, a new analytic workspace is created and data for all four years is loaded and processed (that is, partially pre-aggregated). The full build ran using 48 parallel processes. The total elapsed time was 66 minutes.

Conclusion

This Oracle OLAP Exadata X2-2 performance demonstration shows how Oracle cubes can support a large user community with fast, analytically rich queries. Oracle cubes, with stored and calculated measures, are managed centrally in Oracle Database and are easily accessible using SQL and an MDX provider. Every organization should consider that cubes are a valuable tool for improving the performance and analytic content of business intelligence solutions.

Oracle cubes run best on Exadata. The processing power, available memory, and fast disk allow large cubes to be processed quickly and efficiently. Although that same power benefits query performance, Exadata Smart Flash Cache provides extra query performance not available in any other system.

Note: This is one example of cube performance on Exadata. Different data sets, which differing data volumes and data models, can result in different performance profiles.

Appendix A – More Detail for the DBA

The following sections contain more detailed information that will be of interest to the database administrator.

Query Processing

It is useful to understand how queries are executed against the cube when interpreting query performance. This understanding provides useful background information and helps explain some important differences between cubes and tables.

First, as previously noted, the database relies on smart data types and query execution for maximum query performance. It does not rely on parallel query.

The database will optimize SQL queries that select from a cube by pushing as much of the query into the cube as possible. This allows measures and aggregates to be computed in the cube, and filters and joins to be pushed into the cube. Processing that cannot be pushed into the cube will be done on data that is returned from the cube to the SQL engine.

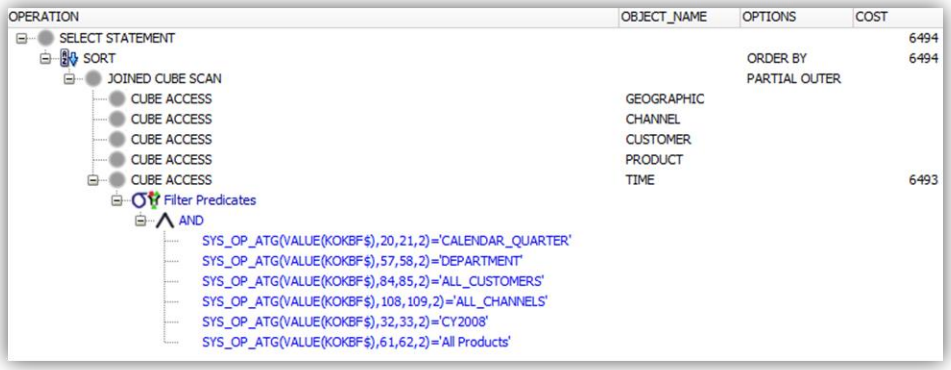

Figure 5. A typical SQL execution plan for a query selecting from Oracle dimensions and a cube.

In this example, predicates are pushed into the cube and SQL joins are transformed into a joined cube scan. This execution plan allows the database to optimize the query within the cube, processing data as efficiently as possible and returning the minimal number of rows to the SQL engine. Depending on the query, it is possible that some processing will occur in the SQL engine (in this example, note the sort operation).

Dedicated Tablespace for the Analytic Workspace

As the number of concurrent users querying the cube increase, the amount of time spent in cluster wait events can increase when running on a cluster (RAC) database. This is due to the increased volume of requests from the global buffer cache. Some cluster wait events can be significantly reduced or

eliminated by creating the analytic workspace in a separate tablespace (one that has no other objects) and making that tablespace read only when it is being queried.

Cube Build Script

The following script was used to build the cube.

```
-- Create the analytic workspace from a template that is stored
-- in a table.
DECLARE
  v_xmlClob CLOB;
BEGIN
  SELECT aw_template
  FROM aw_xml_catalog
  WHERE template_name = 'geog_mon_0_40.xml' INTO v_xmlClob;
  DBMS_CUBE.IMPORT_XML(v_xmlClob);
END;
/
-- Keep the cube in cell flash cache (Exadata Smart Cache)
ALTER TABLE aw$salestrack storage(cell_flash_cache keep)';
ALTER TABLE aw$salestrack MODIFY lob (awlob)(storage(cell_flash_cache
keep))';
-- Turn off logging on the AW$ table.
ALTER TABLE aw$salestrack MODIFY lob (AWLOB) (NOCACHE NOLOGGING)';
ALTER TABLE aw$salestrack MODIFY partition ptn1 lob (AWLOB) (NOCACHE
NOLOGGING)';
ALTER TABLE aw$salestrack MODIFY partition ptnn lob (AWLOB) (NOCACHE
NOLOGGING)';
-- Drop the index on the aw$ table.
DROP INDEX aw$salestrack_i$;
-- Make sure the tablespace is read write.
ALTER tablespace OLAP_PERF_AW read write;
-- Distribute parallel build jobs evenly across database nodes.
ALTER system SET job_queue_processes = 13 scope=memory;
-- Build the cube.
BEGIN
  DBMS_CUBE.BUILD('GEOGRAPHIC',  -- cube name
                  'S',           -- fast refresh
                  false,         -- refresh after errors
                  48,            -- parallel processes
                  false,         -- atomic refresh
                  true,          -- automatic order
                  false          -- add dimensions
                  );
END;
/
```

Exadata Smart Flash Cache

For the best query performance, particularly with a cold database buffer cache, the cube should be kept in Exadata Smart Flash Cache. Smart Flash Cache is ideal for the high volume random IO patterns typical of cube queries.

JOB_CUBE_PROCESSES

The cube was partitioned by month (resulting in 48 partitions), allowing 48 parallel processes to be used to process the cube. In this case, DBMS_CUBE.BUILD creates 48 separate sessions using the Oracle job scheduler with each session processing a partition. The JOB_QUEUE_PROCESSES parameter was used to control the maximum number of jobs per node, distributing jobs evenly across nodes.

A value of 12 for JOB_QUEUE_PROCESSES could be used to ensure a perfectly even distribution of jobs across the four database nodes (48 partitions divided by 4 nodes) providing that no other jobs are running while the cube is being built. This demonstration used a value of 13 to accommodate each of the 48 jobs processing cube partitions and 4 other jobs that might be running on the cluster (for example, system jobs that collect statistics or perform other database maintenance activities). While this approach could slow the cube build by overloading a node with more jobs than available cores, it better represents how a production database might be managed.

ALTER TABLESPACE olap_perf_aw READ WRITE

The ALTER TABLESPACE olap_perf_aw read-write command was used to ensure that the tablespace was updatable during the cube build.

Logging

Turning off logging on the AW$ table improves cube build performance by eliminating time spent writing to redo logs. This is a reasonable strategy with the cube because a failed or aborted cube build will typically be restarted from the beginning.

DROP INDEX aw$salestrack_i$;

On Exadata, the index on the AW$ table can be dropped to eliminate LOB index contention that can occur when many parallel processes are used to build a cube. Cell flash cache, balanced I/O, and fast disk more than compensate for the lack of an index on the AW$ table.

Database Parameters Related to Memory Allocation

As a general rule, cube build and query performance benefit from more available memory. Table 5 lists memory parameters and values used in this performance demonstration.

TABLE 5. DATABASE PARAMETERS RELATED TO MEMORY ALLOCATION

PARAMETER	VALUE	NOTES
DB_CACHE_SIZE	25G	Improves cube aggregation performance.
SGA_TARGET	50G	Improves cube aggregation performance.
PGA_AGGREGATE_TARGET	25G	Improves cube aggregation performance and improves query performance with many concurrent users.

Other Database Parameters

Several other database parameter settings were either changed from a value typically found in a data warehouse environment or are otherwise worth noting are described in Table 6.

TABLE 6. OTHER DATABASE PARAMETERS

PARAMETER	VALUE	NOTES
DEFERRED_SEQMENT_CREATION	False	Reduces TS enqueue contention wait events
PARALLEL_FORCE_LOCAL	True	Improved performance of loading data into the cube.
UNDO_RETENTION	21600	Set to a value that exceeds the cube build time to prevent 'snapshot to old' errors.
_BUFFER_BUSY_WAIT_TIMEOUT	2	Reduces gc buffer acquire wait events
_OLAP_PARALLEL_UPDATE_SERVER_NUM	5	Controls the number of parallel slave processes that are used during the UPDATE command in a cube build. Note: this does not affect the number of processes that might be used for table queries and other operations.)

Appendix B – Sample Queries

This section provides two sample queries, one that selects only stored measures and another that selects stored and calculated measures.

Stored Measure Query

The following query selects stored measures at the Quarter, All Products, Region, and Class levels.

```
SELECT d1.calendar_year_long_descr,
  d1.calendar_quarter_long_de,
  d1.calendar_quarter_end_dat,
  d2.all_products_long_descri,
  d3.all_customers_long_descr,
  d3.region_long_description,
  d4.all_channels_long_descri,
  d4.class_long_description,
  f.sales sales
FROM time_calendar_view d1,
  product_departments_view d2,
  customer_regions_view d3,
  channel_classes_view d4,
  geographic_view f
WHERE d1.dim_key                = f.time
AND d2.dim_key                  = f.product
AND d3.dim_key                  = f.customer
AND d4.dim_key                  = f.channel
AND d1.level_name               = 'CALENDAR_QUARTER'
AND d2.level_name               = 'ALL_PRODUCTS'
AND d3.level_name               = 'REGION'
AND d4.level_name               = 'CLASS'
AND d1.calendar_year_long_descr = 'CY2005'
AND d2.all_products_long_descri = 'All Products'
AND d3.all_customers_long_descr = 'All Customers'
AND d4.all_channels_long_descri = 'All Channels'
ORDER BY d1.calendar_year_long_descr,
  d1.calendar_quarter_long_de,
  d1.calendar_quarter_end_dat,
  d2.all_products_long_descri,
  d3.all_customers_long_descr,
  d3.region_long_description,
  d4.all_channels_long_descri,
  d4.class_long_description;
```

Stored and Calculated Measure Query

The next query selects both stored and time series measures at the Year, Category, Country, and All Channels levels.

```
SELECT d1.calendar_year_long_descr,
  d1.calendar_year_end_date,
  d2.all_products_long_descri,
  d2.department_long_descript,
  d2.category_long_descriptio,
  d3.all_customers_long_descr,
  d3.region_long_description,
  d3.country_long_description,
  d4.all_channels_long_descri,
  f.sales sales,
  f.sales_yr_ago sales_yr_ago,
  f.sales_chg_yr_ago sales_chg_yr_ago,
  ROUND(f.sales_pctchg_yr_ago,2) AS sales_pctchg_yr_ago,
  f.sales_ytd sales_ytd,
  f.sales_ytd_yr_ago sales_ytd_yr_ago,
  f.sales_ytd_chg_yr_ago sales_ytd_chg_yr_ago,
  ROUND(f.sales_ytd_pctchg_yr_ago,2) AS sales_ytd_pctchg_yr_ago
FROM time_calendar_view d1,
  product_departments_view d2,
  customer_regions_view d3,
  channel_classes_view d4,
  geographic_view f
WHERE d1.dim_key                = f.time
AND d2.dim_key                  = f.product
AND d3.dim_key                  = f.customer
AND d4.dim_key                  = f.channel
AND d1.level_name               = 'CALENDAR_YEAR'
AND d2.level_name               = 'CATEGORY'
AND d3.level_name               = 'COUNTRY'
AND d4.level_name               = 'ALL_CHANNELS'
AND d1.calendar_year_long_descr = 'CY2007'
AND d2.all_products_long_descri = 'All Products'
AND d2.department_long_descript = 'Computers'
AND d2.category_long_descriptio = 'Total Personal Computers'
AND d3.all_customers_long_descr = 'All Customers'
AND d3.region_long_description  = 'South America'
ORDER BY d1.calendar_year_long_descr,
  d1.calendar_year_end_date,
  d2.all_products_long_descri,
  d2.department_long_descript,
  d2.category_long_descriptio,
  d3.all_customers_long_descr,
  d3.region_long_description,
  d3.country_long_description,
  d4.all_channels_long_descri;
```

Oracle OLAP Exadata X2-2 Performance Demonstration
November 2011
Author: Bud Endress
Contributing Authors: Jameson White

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together