

# Oracle9i for e-Business: Business Intelligence

*An Oracle Technical White Paper  
July 2002*

# Oracle9i for e-Business: Business Intelligence

Introduction .....	3
Enhancements to core capabilities .....	4
Performance .....	5
Scalability.....	8
Larger data volumes .....	8
More users .....	11
Manageability.....	14
Oracle9i: A Business Intelligence Platform .....	15
OLAP .....	15
Background .....	15
The solution: Oracle9i OLAP.....	16
Benefits.....	18
Data Mining.....	18
Data Mining API .....	19
Prediction and Classification: Naive Bayes and Decision Tree ..	19
Finding Associations: Clustering and Association Rules .....	19
Extraction, Transformation, Loading .....	20
Conclusion.....	22

# Oracle9i for e-Business: Business Intelligence

## INTRODUCTION

Oracle is the most popular database for data warehousing because of its success in satisfying the core requirements for data warehousing: performance, scalability, and manageability. Oracle7 (Release 7.3), Oracle8, and Oracle8i each introduced significant capabilities to meet these core requirements. Oracle9i extends this trend, with new features for storing larger volumes of data, supporting more users, and delivering faster performance. Oracle9i provides significant enhancements to every facet of Oracle8i's data warehouse capabilities.

Oracle9i is the first true 'business intelligence platform' – a platform which can handle the requirements not only of data warehousing, but also OLAP, data-mining and ETL operations.

However, Oracle9i goes far beyond these core data-warehousing requirements of performance, scalability, and manageability. Oracle9i is the first true 'business intelligence platform'. A business intelligence system is much broader than just a data warehouse. For example, business intelligence systems often include more sophisticated analytic capabilities such as OLAP and data-mining functionality. Traditionally, an enterprise-wide business intelligence system has utilized multiple products and data engines to handle the specific requirements of data-warehousing, ETL, data-mining, and OLAP. With Oracle9i, all of the server-side capabilities required for building business intelligence applications has been integrated into a single engine.

This integration provides three major benefits to the administrators, application developers, and end-users of business intelligence systems:

First, integration can vastly reduce costs. When a business-intelligence environment includes both a relational database and a multi-dimensional database, then this environment generally requires at least two administrators. With Oracle9i and integrated OLAP, a single database administrator can manage both environments because both environments are now integrated into a single platform. Additionally, the administration is

done with a single management tool, Oracle Enterprise Manager. Now imagine an environment not simply with two server products, but with four or more separate servers; rather than having administrators for four separate products, Oracle9i allows the entire business intelligence infrastructure to be built on a single product.

Second, integration extends the underlying strengths of the relational database platforms to the areas of OLAP, data-mining, and ETL. Supporting thousands of concurrent OLAP users is no longer an issue; with Oracle Real Application Clusters, OLAP applications can scale across multiple hardware nodes. Efficiently mining terabytes of raw data is no longer an issue; Oracle's parallel infrastructure can scale on any hardware platform and any hardware architecture. In a similar fashion, Oracle provides high-availability, disaster-recovery, and enterprise-level security to every aspect of business intelligence infrastructure.

Third, an integrated business intelligence infrastructure provides more consistent and more complete information to the end-users faster. Previously, with multiple data-server products, data must be transported through several separate servers before being accessed by the end-user. With Oracle9i, when all of the business-intelligence capabilities are integrated into a single platform, the data-latency associated with moving data from one server to another are virtually eliminated. The net result is that business users will get to analyze their data sooner with Oracle9i.

The rest of this paper has two main sections. The first section describes Oracle9i's new enhancements for core data-warehousing requirements and focuses on relational-database enhancements. The second section discusses the enhancements which support Oracle9i as a business intelligence platform, and introduces Oracle OLAP and Oracle Data Mining, two tightly integrated options to Oracle9i Enterprise Edition.

## **ENHANCEMENTS TO CORE CAPABILITIES**

There are three primary requirements of a relational database for data warehousing:

- Performance
- Scalability
- Manageability

The first, and probably most important, requirement is performance. An end-user typically accesses a data warehouse using a tool or application. The only characteristic of the database which the end-user can observe is performance: the end-users see how fast the results of a given query are processed and returned to

Oracle enhances the performance, scalability and manageability of the database in every release. The features discussed in this paper are simply the new features introduced in Oracle9i, built on top of all of Oracle's previous data warehousing features.

their tool or application. For this reason, performance is typically the most important requirement for a data warehouse database.

The second key requirement is scalability. Data warehouses often grow, both in terms of the volume of data and also in terms of the number of end-users. Therefore, the data server must be able to scale; that is, the data server must be able to handle larger volumes of data and/or more users.

The third key requirement is manageability. As a data-warehouse grows, it must continue to be simple to maintain. A data warehouse should not require additional database administrator resources simply because its data volume is growing or because the number of users is increasing.

The following sections discuss the key capabilities in Oracle9i to support each of these requirements, with a particular emphasis on the features introduced in Oracle9i Release 1 and Oracle9i Release 2.

## Performance

No two data warehouses have the exact same performance requirements. For example, one data warehouse may use a star-schema data model while another data warehouse uses a third-normal form data model. The users of one data warehouse may issue simple queries and demand response times in seconds, while users of another data warehouse may issue complex queries and be satisfied with response times of 30 minutes.

Oracle's goal in data warehouse performance is to satisfy every organization's performance requirements. Oracle9i provides a broad set of performance features, so that every data warehouse, regardless of its design and its requirements, provides the appropriate response times to its business users.

Briefly, some of the key features introduced in earlier releases of Oracle include:

**Partitioning:** Partitioning typically provides many benefits for query performance. The most basic way in which partitioning improves query performance is through partition-pruning. For example, suppose that a table contained three years of historical data and was partitioned by month, and suppose that a query requested only the data for July of the current year. Oracle will automatically 'prune' the unnecessary partitions, so only the partition corresponding to July is accessed. In this example, the partition-pruning can result in a 36x gain in performance, since Oracle is scanning a single partition instead of 36 partitions.

**Parallelism:** Parallelism is the ability to apply multiple CPU and I/O resources to the execution of a single SQL command. Oracle's unique parallel architecture allows any query to execute with any degree of parallelism. Oracle intelligently chooses the degree of parallelism for each query, based upon the complexity of the query, the size of the tables in the query, the hardware configuration, and the currently level of activity on the system. Parallelism is a fundamental performance feature for executing queries over large volumes of data.

**Materialized views:** One of the techniques employed in data warehouses to improve performance is to pre-compute the results of commonly-executed queries. That is, rather than executing expensive joins and aggregates for every query, these results can be computed once and stored in materialized views. Oracle's materialized views are transparent to the end-user applications; the end-user does not need to write specific SQL to take advantage of materialized views; instead Oracle automatically transform the SQL queries to utilize materialized views where appropriate. For data-warehouse workloads in which the same joins and aggregations occur repetitively, materialized views often provide orders-of-magnitude performance gains.

**Bitmap indexes:** The most common type of index in an Oracle data warehouse is a bitmap index. Oracle's patented compression makes these bitmap indexes extremely small; bitmap indexes are typically an order of magnitude smaller than b-tree indexes. The benefit of this compression is that customers can create more indexes using the same amount of storage and with the same amount of maintenance costs. End-users can get better query performance because they can build indexes on more of their key columns with bitmap indexes.

**Star-query optimizations:** One typical type of query often found in data warehouses is a star query. Oracle has developed specific technology to address this common type of business query. Oracle supports star queries with its 'star transformation' technology, an innovative application of bitmap indexes and advanced query optimization. This proven technology has been broadly implemented by customers using Oracle8 and Oracle8i.

**Analytic SQL operations:** In Oracle8i, Oracle introduced a wide set of new SQL operations for executing more analytic operations in the database. These operations included ranking, moving averages, cumulative sums, ratio-to-reports, and period-over-period comparisons. Although some of these calculations were previously possible using SQL, the Oracle8i syntax (based on a recent addendum to the ANSI SQL standard) provides much better performance for these calculations.

Like every release, Oracle9i introduces further enhancements to Oracle's performance. These new features include:

#### ***Bitmap Join Index***

A 'join index' is an index structure which spans multiple tables, and improves the performance of joins of those tables. With materialized views, Oracle8i already provides a broad mechanism for improving join performance. Bitmap join indexes provide further improved performance for a more specific class of join-queries. Bitmap join indexes can be particularly useful for "star queries," and in some cases, bitmap join indexes can improve query performance by a factor of 30.

Bitmap join indexes are best understood by examining a simple example. Suppose that a data warehouse contained a star schema with a fact table named SALES and

a dimension table named CUSTOMER. Using bitmap join indexes, the following join index could be created on the SALES and CUSTOMER tables:

```
CREATE BITMAP INDEX cust_sales_bji
ON Sales(Customer.state)
FROM Sales, Customer
WHERE Sales.cust_id = Customer.cust_id;
```

This join index could be used to evaluate the following query. In this example query, the CUSTOMER table will not even be accessed; the query is executed using only the join index and the sales table.

```
SELECT SUM(Sales.dollar_amount)
FROM Sales, Customer
WHERE Sales.cust_id = Customer.cust_id
AND Customer.state = 'California';
```

If the CUSTOMER table is a large dimension table (and customer-based dimension tables are often tens of millions of records), then the bitmap join index can vastly improve performance by not requiring any access to the CUSTOMER table.

#### ***New analytic SQL functions***

Building on the foundation of Oracle8i's analytic SQL function, Oracle9i additionally provides SQL support for:

- Inverse Percentiles - These functions allow queries to find a specified the data which corresponds to a specified percentile value. For instance, users may find the median value of a data set by querying PERCENTILE\_DISC(0.5)
- Hypothetical Rank and Distributions - These functions allow queries to find what rank or percentile value a hypothetical data value would have if it were added to an existing data set
- Histograms - This functions creates a width-balanced histogram of the data. For each row, this function returns a number representing the bucket number
- FIRST/LAST aggregates - This allows comparisons between any element in a group and the first or last element. For example, this function could compute difference between the current balance and the balance on the first day of the month

#### ***SQL Aggregation Enhancements***

In Oracle8i, aggregations were enhanced with the addition of the CUBE and ROLLUP operators. These operators, extensions to the SQL GROUP-BY capabilities, enabled a single SQL query to calculate multiple levels of aggregation.

Oracle9i extends this capability farther with the introduction of **grouping sets**. This feature allows the a SQL query to specify the exact levels of aggregation of interest. Grouping sets are specified very easily by following the GROUP BY keyword with the term GROUPING SETS and a column specification list. For example, we can say:

```
SELECT
  year,
  region,
  product,
  sum(sales)
FROM
  salesTable
GROUP BY
  GROUPING SETS ((year, region, product),
  (year, product),
  (region, product));
```

The SQL above calculates aggregates over exactly 3 groupings - (year, region, product), (year, product), and (region, product).

The benefit of this feature is performance: rather than computing these three aggregates using three separate queries, all three aggregates can be computed in a single SQL query with a single scan of the underlying data.

### Scalability

Scalability is the capability of the database to support larger volumes of data and/or larger numbers of concurrent users.

One approach to addressing heavier database workloads is by adding hardware. Oracle has proven abilities to scale up within a single platform, up to the largest SMP and NUMA systems, and to scale up across multiple systems in a cluster or MPP environment using Real Application Clusters.

However, database scalability is not simply the characteristic of supporting larger hardware platform. Database scalability is also the ability to use these hardware capabilities as efficiently as possible, by providing algorithms and database features so that larger volumes of data and larger numbers of users can be accommodated without requiring additional hardware. These aspects of scalability are discussed in this section.

### Larger data volumes

For data warehousing, one of the most challenging areas of scalability is in supporting large data volumes. Data warehouses are typically the largest databases in the enterprise, and thus data management is a key requirement. There are two key capabilities necessary for supporting large volumes of data: partitioning and parallelism. Partitioning provides the capability to break operations on very large volumes of data into smaller operations; partitioning is essentially a 'divide-and-

The only characteristic of the database which the end-user can observe is performance: the end-users see how fast the results of a given query are processed and returned to their tool or application. Thus, performance is paramount for the database in a data warehouse.

conquer' technique for managing large tables and indexes. Parallelism provides the capability to apply multiple CPU resources to a single operation; parallelism is the feature that allows Oracle to take full advantage of all of the CPU power available on a system. The key to scalability is the combination of these two capabilities.

Oracle Partitioning, an option to Enterprise Edition first introduced with Oracle8, delivers significant improvements in the manageability, availability, and query performance of large tables and indexes. Oracle provides a broad set of partitioning capabilities, including range partitioning, hash partitioning, composite range-hash partitioning, list partitioning (new in Oracle9i) and composite range-list partitioning (new in Oracle9i).

The advantages of partitioning are most readily apparent in the data-warehouse load scenario. Suppose that a data warehouse table is loaded every week. In this environment, the warehouse administrator will likely choose to range-partition the fact table by time, so that each partition contains one week of data.

This partitioning scheme greatly simplifies the loading of the data warehouse. The new data is loaded into a single partition, which can be indexed and backed-up. Then, the new partition can be added to the partitioned table. This approach provides two key benefits. First, the new data has been loaded with minimal resource utilization. The indexes and backups only need to be created or maintained for a single partition, rather than for the entire table. Second, the new data has been loaded with minimal end-user unavailability. At no time would an index need to be dropped or unavailable, allowing ongoing queries to run entirely uninterrupted.

Key enhancements in Oracle9i for managing larger volumes of data include additional partitioning techniques, and an innovative new compression feature:

#### ***List and Composite Range-List Partitioning***

Oracle's partitioning capabilities have been enhanced in Oracle9i with the addition of a two new partitioning schemes, list partitioning and composite range-list partitioning

These new partitioning scheme provides even more choices to the data warehouse administrator in achieving the best combination of manageability and performance. While Oracle expects that the majority of data warehousing systems will utilize range partitioning, other partitioning schemes (hash partitioning, and composite range-hash partitioning introduced in Oracle8i, and now list and composite range-list partitioning in Oracle9i) offer distinct advantages in certain warehouse environments.

List partitioning gives data warehouse administrators precise control over which data belongs in each partition. For each partition, the data warehouse administrator can specify a list of possible values for the partitioning key of the rows in that partition.

List partitioning complements the functionality of range partitioning. Range partitioning is useful for segmenting a table along a continuous domain (most often, tables are range-partitioned by TIME, so that each range partition contains the data for a given range of TIME values such as one partition per month or per week). In contrast, list partitioning is useful for segmenting a table along a discrete domain. Each partition in a list partitioning scheme corresponds to a list of discrete values.

For example, suppose that a data warehouse for a large corporation contains data for many different countries. The data warehouse administrator could choose to list-partition the table by regions:

```
CREATE TABLE sales_history ( ... )
PARTITION BY LIST (country) (
PARTITION europe VALUES ('United Kingdom', 'Germany',
'France'),
PARTITION north_america VALUES ('United States',
'Canada', 'Mexico'),
PARTITION south_america VALUES ('Brazil',
'Argentina'),
PARTITION asia VALUES ('Japan', 'Korea');
```

Data warehouses often grow, both in terms of the volume of data and also in terms of the number of end-users accessing the data warehouse. Therefore, the data server must be able to scale; that is, the data server must be able to handle larger volumes of data and/or more users.

The primary benefits of partitioning are often best realized when the partitioning strategy closely corresponds to the underlying business processes. List partitioning meets this objective. A data warehouse administrator might choose to use the above partitioning scheme if the data is often accessed or modified according to region or country. Since each region is in its own partition, these region-based operations will be much more efficient.

Composite range-list partitioning combines the advantages of list partitioning and range partitioning. Suppose that a DBA wished to partition the data by geographic region, but additionally planned to load data on a weekly basis. In this scenario, range-list partitioning would be ideal because it would allow the DBA to leverage both the advantages of range-partitioning to support the weekly data loads and the advantages of list-partitioning.

#### ***Table Compression***

Commercially available relational database systems have not heavily utilized compression techniques on data stored in relational tables. One reason is that the trade-off between time and space for compression is not always attractive for relational databases. A typical compression technique may offer space savings, but only at a cost of much increased query time against the data.

Oracle9i provides a unique compression technique that is particularly attractive for large data warehouses. Oracle9i compresses data by eliminating duplicate values in a database block. Because the compression algorithm is optimized for Oracle's internal database block structure, its reduction of disk space can be higher than

standard compression algorithms. More significantly, the compression has virtually no negative impact on the performance of queries; in fact, compression has a measurably positive impact on many queries accessing large amounts of data, as well as on data management operations like backup and recovery. While there is no performance impact on queries over compressed data, data-loading operations are considerably slower for compressed tables.

The typical compression ratios for large database warehouse tables range from 3:1 to 5:1. A 3:1 compression ratio would imply, for example, that a table which previously occupied 300 GB of database space could be stored using only 100GB when compressed. For data warehouses which are growing into the terabytes, compression can provide considerably cost savings, without sacrificing query performance.

#### **More users**

A second dimension of database scalability is supporting large numbers of concurrent users. The key to supporting large numbers of users is being able to provide each query with enough CPU, IO, and memory resources to execute effectively. A naïve approach is to provide each query with the same amount of database resources. That is, every query in the data warehouse could be executed with the same degree of parallelism, the same amount of memory, and the same priority. However, this approach is sub-optimal, since some queries may require many more resources than other queries. Moreover, some users may be much more important than other users, so the queries executed by important users should get more resources.

The challenge of managing large numbers of users is further complicated by the fact that, in any production system, the workload varies. At different parts of the day, there will be more or less concurrent queries.

Oracle9i addresses these requirements with enhanced capabilities for managing large numbers of users on a data warehouses:

- Oracle dynamically adjusts the execution memory and degree of parallelism for each query based on the current system load. When only a few users are executing queries on the data warehouse, each query receives relatively more memory and higher degrees of parallelism than when many users are concurrently executing queries.
- Oracle allows the database administrator to easily control 'runaway' queries, by providing a pro-active query governing capability. Additionally, the database administrator can view the status of ongoing jobs (including information on how much of the query is completed already and the expected completion time) in order to take corrective actions.
- Oracle provides an infrastructure for assigning each user to a specific 'resource profile', which specifies how much resources that user's queries can utilize.

Oracle9i introduces several new features to support these requirements. Two new capabilities, Self-Tuning SQL Execution Memory and the enhancements to the Database Resource Manager, are described in detail here.

#### ***Self-Tuning SQL Execution Memory***

Oracle9i provides an automated mechanism for dynamically allocating *SQL execution memory* for every query. *Execution memory* is memory which is allocated during query runtime for purposes such as sorting and hashing. In many data-warehouse environments, 70% or more of the data warehouse server's physical memory may be allocated for execution memory.

At first glance, self-tuning memory tuning seems like a manageability feature. However, while this feature undoubtedly improves manageability, the primary benefit of this feature is performance and scalability in multi-user environments. Self-tuning memory not only relieves database administrators of the burden of tuning execution memory, but this feature also chooses a more accurate memory allocation strategy than can be achieved by hand-tuning of memory parameters.

In Oracle8i, database administrators could tune the execution memory using parameters such as HASH\_AREA\_SIZE and SORT\_AREA\_SIZE (among others). These parameters controlled the amount of memory allocated for each individual query. These parameters were ideal for tuning an individual query, but were not optimal for an administrator whose data warehouse had hundreds of concurrent queries each with different memory requirements.

In Oracle9i, database administrators tune execution memory using a single parameter, PGA\_AGGREGATE\_TARGET. This parameter establishes a target value for execution memory consumption. Regardless of how many or few concurrent queries are running on the system, Oracle will seek to efficiently utilize all of the available memory as specified by this parameter.

By automating the allocation of execution memory, Oracle will improve the overall throughput of the data warehouse. The data warehouse will be able to support larger numbers of users at the same levels of performance, because the data warehouse is now using its memory much more effectively. Each query is allocated memory based upon its specific requirements, and Oracle9i dynamically adjusts memory allocation while a query is running to ensure good performance. The self-tuning memory feature will ensure that memory-intensive queries receive sufficient memory, while memory-light queries are not given too much memory. By making more effective use of memory, Oracle9i increases overall query performance.

Internal testing has shown that this feature can improve the performance of high-concurrency memory-bound system by 20% or more. Moreover, this feature is very simple to implement (since it only involves changing a few initialization parameters), so that this feature should be a key consideration for all customers who are upgrading to Oracle9i from earlier releases of Oracle.

### *Database Resource Manager*

The Database Resource Manager was introduced in Oracle8i, and provides a mechanism for allocating the resources of a data warehouse among multiple populations of end-users. These groups, called Resource Consumer Groups, are specified by the database administrator, and then the administrator can control how resources are allocated to each group. In Oracle8i, the Database Resource Manager provided a mechanism for controlling the amount of the CPU allocated to each group, and for limiting the maximum degree of parallelism.

In Oracle9i, the database resource manager has been enhanced with several other capabilities which are especially valuable for data warehouses.

First, the number of active sessions for each Resource Consumer Group can be limited. For example, in a data warehouse, the administrator may specify that one group of users is limited to twenty (20) concurrently-running queries. Once the limit on the number of active sessions is reached, if a user in that group submits a new query then that query will be queued; the query will be executed after other queries from the Resource Consumer Group are completed.

A second significant enhancement of the resource manager is a query-governing capability. For each Resource Consumer Group, the database administrator can specify the maximum estimated execution time. If a user in that group submits a query which is expected to take longer than the limit, then that query will be aborted and an Oracle error will be returned. This enhancement prevents excessively long-running queries from even starting, and thus prevents unnecessary allocation of resources to queries which would use too much resources.

The third enhancement is the ability of the resource manager to automatically change the Resource Consumer Group of a given session based on database administrator-specified criteria. For example, the database administrator could specify that if a query from a given Resource Consumer Group runs for more than 5 minutes, then that query should be automatically migrated to a different Resource Consumer Group (and the new Resource Consumer Group may, for example, receive less CPU, so that this long-running query has been dynamically 'de-prioritized' based on the database administrator's criteria). This strategy would ensure that long-running queries are not dominating the resources of the data warehouse platform, and this would be an appropriate strategy for an environment in which the majority of queries runs in at most a couple of minutes, and only an occasional query takes more than 5 minutes.

Using the Database Resource Manager, a database administrator can very precisely control how much resources is being allocated to each group of users (so that more important processes will receive more resources than less important ones), and can furthermore place careful limits on each group of users to ensure that no group or individual query can unduly impact the overall performance of the data warehouse

## **Manageability**

The third key requirement for data warehouse is manageability. More than most database applications, data warehouses tend to grow rapidly and evolve. This places a particular importance on ensuring the data warehouses are manageable. The database administrator should be primarily concerned with ensuring that the data warehouse meets the growing business requirements of the data warehouse, rather than being overly burdened with day-to-day administrative tasks.

Oracle's goal in manageability is to automate and eliminate these administrative tasks. The self-tuning SQL execution memory feature, described in the previous section, is a good example of a feature that improves manageability (in addition to improving scalability and performance); while database administrators previously had to carefully tune the database's memory parameters based upon their knowledge of the number of users and the types of queries running on the system, with Oracle9i the database administrator simply needs to specify the total amount of memory available to Oracle for SQL execution.

Some of the other key manageability features introduced in Oracle9i include:

### ***Automatic Undo Management***

In order to guarantee read-consistency during concurrent transactions and queries, as well as assist in recovery, Oracle stores 'undo' information, which are the older values of data that have been updated. Oracle9i introduces Automatic Undo Management which enables the database server to automatically manage allocation and management of Undo (Rollback) space. Administrators merely need to create an UNDO tablespace (Using the CREATE UNDO TABLESPACE...command) with sufficient disk space. This considerably eases the burden of database administrators in managing undo space. With earlier releases of Oracle, the database administrator would need to determine how many rollback segments to create, and determine the size and storage attributes for each rollback segment. With Oracle9i, the process is entirely automated; the database administrator simply creates a single UNDO tablespace.

### ***Enhanced Statistics Gathering***

Oracle's query optimizer uses statistics about the objects in the database (such as the number of rows and blocks in each table). These statistics are gathered when the database administrator runs the DBMS\_STATS facility. In Oracle9i, the DBMS\_STATS package has been enhanced in order to make it easier for database administrators to gather the appropriate sets of statistics. With a single command in Oracle9i, a database administrator can update all of the statistics for an entire schema. Oracle9i automatically determines which tables have been extensively modified, so that Oracle9i knows which tables require updated statistics. Moreover, Oracle9i automatically determine the appropriate sampling percentage as well as the appropriate columns for histograms. These enhancements greatly simplify to database administrator's task in gathering accurate statistics.

## **ORACLE9i: A BUSINESS INTELLIGENCE PLATFORM**

While Oracle9i extends Oracle's lead in providing enterprise-level performance, scalability and manageability for data warehousing, Oracle9i breaks new ground in several key areas for business intelligence.

Technologies such as OLAP, ETL, and data-mining are hardly new to data warehousing and business intelligence (indeed, some of these technologies arguably preceded data warehousing). Data warehousing practitioners have been able to purchase products with each of these capabilities for years. However, OLAP products typically have their own calculation engine and data storage, ETL products have their own transformation engine, and data mining products have their own mining engines. In short, the business-intelligence software industry was maintaining at least four 'data engines', each requiring its own infrastructure and tools for managing data, its own availability and recovery strategies, its own security mechanisms, and its own parallelism and scalability infrastructure. Not surprisingly, many products lacked robustness in one or more areas. Moreover, these products were not integrated, so that a complete business intelligence system required the resources to implement and manage multiple server products.

Oracle9i is the industry's first 'business intelligence platform'. The benefit of a business intelligence platform is integration—specifically the integration of the data—so that the same server infrastructure can be leveraged for all tasks which involve processing of large amounts of data.

Oracle9i is the industry's first 'business intelligence platform'. The benefit of a business intelligence platform is integration—specifically the integration of the data—so that the same server infrastructure can be leveraged for all tasks which involve processing of large amounts of data.

Oracle has extended the relational database's capabilities and the relational database's language (SQL) in order to be able to meet the requirements of a business intelligence platform, and Oracle has introduced two new Java-based APIs on top of Oracle9i to support the specific requirements of OLAP and data-mining.

The next three sections briefly describe the enhancements in Oracle9i which support these new areas.

### **OLAP**

#### **Background**

In the past, developers of analytic applications had to make a fundamental decision: should an application be based on a relational database accessing data in a data warehouse, or should an application utilize a specialized analytical database? The tradeoffs are not insignificant.

The relational database offered the most cost effective method of managing the data and the most open access to the widest variety of applications. Data could be managed in a centralized location. Since all data was stored in a data warehouse and could be queried using SQL, any SQL-based application could access the data. Unfortunately, the analytical capabilities of SQL were limited and performance tended to lag that of specialized analytical databases.

Specialized analytical databases offered a complete set of analytical functions and tended to provide better query performance, but maintaining an analytical database represented a significant expense. An analytical database required data replication and a separate management infrastructure. Data replication is a costly process and can cause a significant delay in the availability of data. The separate management process is also costly since it requires separate data modeling, ETL processes, security procedures and disaster recovery plans. Most analytic servers do not provide high availability features such as site failover.

**The solution: Oracle9i OLAP**

Oracle9i challenges the traditional view of the analytic servers by offering an integrated relational-multidimensional database. The Oracle9i RDBMS-MDDS eliminates the trade-off between manageability versus performance and analytic power. Oracle9i simplifies the process and reduces the cost of maintaining data while retaining the ability to support complex analytical queries and provide excellent performance. Oracle9i accomplishes this both by offering Oracle9i OLAP as an integrated part of the relational database.

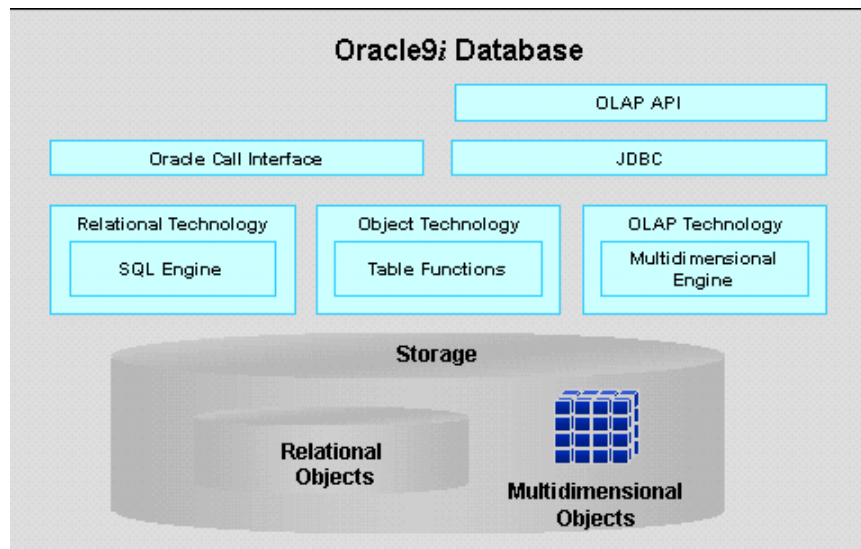
The Oracle9i relational-multidimensional database is composed of three technologies:

Relational technology manages relational database objects and provides a SQL interface to the data.

Object technology allows the database to process non-relational data.

OLAP technology provides high-end analytic functionality in the context of a multidimensional model.

This is illustrated below:



Relational objects and multidimensional objects are stored in the Oracle9i database. Both of these data-storage models can be accessed by SQL or by the OLAP API, a new API in Oracle9i tailored for OLAP applications. Object technology provides the ability to represent non-relational data types as relational data types to SQL. In the context of the Oracle9i OLAP option, the object technology enables the SQL interface to multidimensional data.

These core technologies are supported by a management infrastructure provided by Oracle Enterprise Manager and Oracle Warehouse Builder, and metadata in the form of the Oracle data dictionary and the OLAP catalog. Business intelligence tools such as Oracle Reports, Oracle Discoverer, and Business Intelligence Beans leverage the integrated database to provide a tools that can be used to query the database and analyze data.

In Oracle9i all data - relational and multidimensional - is stored in Oracle data files. There are no separate multidimensional files to manage. Relational and multidimensional data types can coexist in the same data files. All data files, including those with multidimensional data, are managed as standard Oracle data files. Multidimensional data is stored in the database as 'analytic workspaces'. An analytic workspace can contain a number of multidimensional database objects and could be thought of as a multidimensional schema.

As with any other data in the Oracle database, OLAP analytic workspaces belong to an Oracle schema. Standard RDBMS access control methods (for examples, GRANTS) apply to analytic workspaces, thereby providing one level of access control. Analytic workspaces can be partitioned, that is, distributed across disks to maximize performance of I/O operations.

Since all Oracle data - relational and multidimensional - is stored in Oracle data files and is equally accessible by SQL- and OLAP API-based applications, data never needs to be replicated. DBAs can choose the storage method that is best for their applications.

Access to OLAP data, whether stored in relational or multidimensional form, is either via SQL or the Java OLAP API. This API is an object-oriented Java API that provides encapsulation, abstraction, and inheritance. The Java OLAP API provides connection, multidimensional navigation, data selection, analysis functions, and cursor management features.

The Java OLAP API insulates the business intelligence application from both the physical data store and the access methods required to obtain data from the data source. The application does not need to be aware of the data source — relational or multidimensional tables — or how to access the data. This insulation allows the DBA complete freedom to manage physical storage in the database without disturbing OLAP API-based applications.

The Java OLAP API also insulates the business intelligence application from the underlying complexity of some of the analytic operations. When data is stored in

relational tables, the SQL necessary to resolve multidimensional queries can be extremely complex. Even the most experienced SQL-literate application developer would find it difficult to generate SQL for these types of queries. Generating SQL that performs well is even more challenging. Application developers are more productive writing to the OLAP API, which is designed from the ground up for OLAP. They can leave the task of generating optimal SQL up to Oracle9i OLAP.

Another key OLAP capability is OLAP DML, the OLAP data manipulation language. OLAP DML is specifically designed to support the definition of multidimensional calculations. Although the languages are actually very different, one can think of the OLAP DML as being the multidimensional equivalent to PL/SQL. The OLAP DML has a substantial inventory of analytic functions that can be used to produce just about any type of multidimensional calculation. Since it is a procedural language, the calculations can include conditional logic. This allows calculation rules to vary depending on the data values. An application could, for example, apply one calculation rule when sales values are less than 100,000 and another when sales values are greater than 100,000.

#### **Benefits**

There are many benefits to an fully integrated RDBMS-MDDS database as compared to separate, stand-alone multidimensional databases. These include:

- Simplified management
- High availability
- Improved security
- Open access to both SQL and OLAP API clients
- Reduced information cycle time
- Improved data reliability

#### **Data Mining**

Oracle9i Data Mining, an option to Oracle9i Enterprise Edition, allows companies to build advanced business intelligence applications that mine corporate databases to discover new insights, and integrate those insights into business applications. Oracle9i Data Mining embeds data-mining functionality into the Oracle9i database, for making classifications, predictions, and associations.

Oracle9i Data Mining allows application developers to integrate data-mining capabilities into their business intelligence applications to support such activities as:

- Preventing customer attrition
- Cross-selling to existing customers
- Acquiring new customers
- Detecting fraud

Oracle9i Data Mining executes all data-mining operations within the database, directly against relational tables.

- Identifying the most profitable customers
- Profiling customers with more accuracy

Oracle9i Data Mining is completely integrated into the relational database. All model-building, scoring, and metadata management operations are initiated via a Java-based API and occur entirely within the relational database.

#### **Data Mining API**

Application developers access Oracle9i Data Mining's functionality through a Java-based API. Programmatic control of all data mining functions enable automation of data preparation, model building, and model scoring operations.

Java Data Mining is an emerging data mining standard, following Sun's Java Community Process as a Java Specification Request. Oracle9i Data Mining's API provides an early look at concepts and approaches being proposed for Java Data Mining. Ultimately, Oracle9i Data Mining will comply with the standard once the standard is published.

#### **Prediction and Classification: Naive Bayes and Decision Tree**

Oracle9i Data Mining provides both Naïve Bayes and Decision Tree (adaptive Bayes network) data mining algorithms for making predictions and classifications. These algorithm is applicable to a variety of data mining problems and provides high accuracy. By finding patterns in data, companies can, for example, make predictions about the future behavior of customers with similar characteristics -- using the past as a predictor of the future. Typical prediction applications estimate the probability of an outcome, such as "0, 1" or "yes, no" or "A, B, C or D." For example, a campaign management system may want to know the probability of a given customer responding to a given offer. The Naive Bayes algorithm can estimate this probability, and based on this probability, a company can target its campaigns at those customers who are most likely to respond. The Decision Tree algorithm provide human readable "rules" that are useful for explanation; the output of the Decision Tree algorithm could be "if age is between 30 and 40 and income class is 8, then this individual will be interested in an offer related to BMW automobiles with a confidence of 85%".

#### **Finding Associations: Clustering and Association Rules**

Oracle9i Data Mining also provides Clustering and Association Rules data mining algorithms to detect naturally occurring clusters, or 'associated' or co-occurring events hidden in databases.

Cluster analysis is popular for discovering groupings within the data that may reveal some additional insight. For example, an clustering algorithm might subdivide a given company's customers into several separate clusters. An example output of clustering might reveal that the average age of Cluster 1 is 20% higher

than the average age of Cluster. A company could then use their knowledge of these clusters to implement more effective marketing campaigns.

Association analysis is often used to find popular product bundles (e.g. market basket analysis), such as 'milk' and 'cereal' being associated with 'bananas'.

Associations can also be used to identify co-occurring items or events such as:

- what manufactured parts and equipment setting are associated with failure events?
- which patient and drug attributes are associated with which outcomes?
- which items or products is this person most likely to buy or like?

Associations can be used to predict the next item placed into the shopping basket which can be helpful to satisfy customers and increase average order value.

Oracle9i Data Mining provides a powerful, scalable infrastructure for building applications that automate the extraction of business intelligence and its integration into other applications. The Naïve Bayes, Association Rules, Decision Trees and Clustering data mining algorithms can solve a wide variety of business problems, and thus Oracle9i Data Mining opens the door for integrating sophisticated data-mining capabilities, once the domain of specialized servers, into mainstream business intelligence applications. By automating, integrating, and “operationalizing” the extraction and distribution of new business intelligence insights, companies can leverage their investment in data, operate more effectively, and obtain greater competitive advantage.

### **Extraction, Transformation, Loading**

Oracle9i provides a robust set of server functionality to address the typical requirements of **ETL** (Extraction, Transformation, Loading) processes. This functionality is focused on providing a scalable ETL infrastructure, to allow the processing of large volumes of data into the data warehouse.

Oracle Warehouse Builder (included in Oracle9i Developer Suite) along with Oracle9i provides a complete solution for ETL. Oracle Warehouse Builder provides a graphical interface for designing and implementing the ETL process while Oracle9i provides the infrastructure for scalability data transformations.

The key features in Oracle9i for ETL include:

#### ***External tables***

The external table feature allows external data, such as flat files, to be exposed within the database just like a regular database table. External tables can be accessed via SQL, so that external files can be queried directly and in parallel using the full power of SQL, PL/SQL, and Java. External tables will commonly be used in the ETL process to combine data-transformations (via SQL) with data-loading into a single step. External tables are a very powerful feature with many possible

Oracle9i provides the capability to stream data into the database through multiple transformations in parallel; Oracle9i is a scalable, function-rich ETL engine.

applications in ETL as well as other database environments in which flat-files are processed.

#### ***Upsert functionality***

MERGE is a new Data Manipulation Language (DML) command which allows a row (or sets of rows) to be conditionally updated or inserted, also known as 'Upsert'. A common data warehouse scenario is to receive a set of rows which are either modifications to previously existing rows (updates) or are new rows (inserts). The MERGE command can process both types of rows in a single SQL command. The benefit of MERGE is improved performance, since these DML operations can be done in a single logical operation, rather than a separate update and insert command.

#### ***Multi-table inserts***

Multi-table inserts allows data to be inserted into multiple target tables. SQL predicates control which rows are inserted into each target table. For example, a multi-table insert command could insert pending sales orders into a separate table from closed sales orders. Like the upsert feature, the primary benefit of multi-table inserts is improved performance; a single multi-table insert statement performs much better than multiple single-table insert statements.

#### ***Table functions***

Data transformations can be arbitrarily complex; therefore, it is necessary to provide an extensible framework in order to implement transformations which cannot be implemented using SQL. Oracle9i's Table Functions provide the support for pipelined and parallel execution of such transformations implemented in PL/SQL, Java, C, C++ (any language supported by Oracle8i).

Not only do each of the above features extend the functionality of the Oracle Server for ETL processing, but also each of these features focused on scalability, since all of these operations can be fully parallelized. Scalability is a key advantage provided by Oracle9i when compared to some of the alternative ETL solutions available today.

These ETL features are particularly powerful because they can all be used in conjunction with each other. For example, a single SQL operation could select data from a flat file using the external table feature, join this flat file data to other lookup tables within the database, apply additional complex transformations using table functions, and insert the results into multiple target tables. All of these operations are done in parallel, and within a single SQL statement. In summary, Oracle9i provides the capability to stream data into the database through multiple transformations in parallel; Oracle9i is a scalable, function-rich ETL engine.

## **CONCLUSION**

Oracle9*i* continues to enhance the industry-leading Oracle database platform, increasing Oracle's capabilities to support business intelligence systems containing increasingly more data and more users, all with improved manageability.

Moreover, Oracle9*i* introduces genuinely revolutionary new business intelligence functionality, by providing the first OLAP and data-mining servers which are completely embedded in the relational database. These new capabilities, along with significant enhancements in the ETL areas, propel Oracle9*i* from a relational database used primarily for warehouse query-processing to a complete business intelligence platform capable of supporting all business intelligence workloads.



Oracle9i for e-Business: Business Intelligence

July 2002

Author: George Lumpkin

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  
[www.oracle.com](http://www.oracle.com)

Oracle Corporation provides the software  
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various  
product and service names referenced herein may be trademarks  
of Oracle Corporation. All other product and service names  
mentioned may be trademarks of their respective owners.

Copyright © 2000-2002 Oracle Corporation  
All rights reserved.