

Oracle9i for e-Business: Business Intelligence

*An Oracle Technical White Paper
June 2001*

Oracle9i for e-Business: Business Intelligence

INTRODUCTION

Oracle9i is the latest release of the leading relational database for data warehousing today. Oracle is most often chosen 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. Since data warehouses will store larger volumes of data, support more users, and require faster performance, these core requirements remain key factors in the successful implementation of data warehouses. Oracle9i continues to focus on these core requirements, with 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. Business intelligence systems often include more sophisticated analytic capabilities such as OLAP and data-mining functionality.

Many data warehouses today use the relational database primarily for managing data and executing basic queries. While these operations are fundamental to any data warehouse, Oracle9i broadens the footprint of the relational database so that Oracle9i is the scalable data engine for all server-based operations in a business intelligence system, not just loading and basic query operations. Oracle9i is designed to be the foundation not only of the data warehouse, but also of business intelligence. Oracle9i provides new server functionality in three areas: OLAP (On-

Line Analytic Processing), ETL (Extraction, Transformation, Loading), and data mining.

The rest of this paper has two main sections. The first section describes Oracle9i's new enhancements for core data-warehousing requirements. The following 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 Editions.

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 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 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 by the addition of new hardware resources.

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 new features in Oracle9i to support each of these requirements.

Performance

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.

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.

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.

FSuppose 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.

Automatic Memory Tuning

Oracle9i provides an automated mechanism for dynamically allocating *runtime memory* to each query. *Runtime memory* is memory which is allocated during query execution 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 runtime memory.

At first glance, automatic memory tuning seems like a manageability feature. However, while this feature undoubtedly improves manageability, the primary benefit of this feature is performance. Automatic memory tuning not only relieves database administrators of the burden of tuning runtime memory, but automatic memory tuning 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 runtime 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 runtime memory using a single parameter, PGA_AGGREGATE_TARGET. This parameter establishes a target

value for runtime 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 runtime 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 automatic memory tuning 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 a 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.

Enhancements to Materialized Views

Materialized views were introduced in Oracle8i. A fundamental feature for data warehousing, materialized views provide a mechanism for improving the performance of almost any type of query. A materialized view should be thought of as a special kind of view, which physically exists inside the database, can contain joins and/or aggregates, and exists to improve query execution time by pre-calculating expensive joins and aggregation operations prior to execution.

Query response time continues to decrease with enhancements to materialized views. Now even more queries are capable of using a materialized view with the ability to create a materialized view based on a subset of data e.g. only regions EMEA and Asia. The time required to maintain the materialized views has been reduced with many more of them now able to use the fast refresh method. To assist management and usage, two new packages are available which advise why a query did not use a materialized view and what it is capable of.

Materialized views have been enhanced in Oracle9i in several important ways, including more sophisticated query-rewrite mechanisms, such as support for data subsets.

and enhanced refresh mechanisms which allow fast, incremental refreshes for a wider variety of materialized views.

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.

Support of additional SQL syntax

Full Outer Joins

Oracle9i supports full outer joins. Previously, Oracle supported only one-sided outer joins. Oracle9i supports the ANSI join syntax, with complete support for one-sided and full outer joins. This feature both improves the performance of certain complex queries and additionally simplifies the task of expressing complex business questions using SQL.

WITH Clause

Oracle9i also supports the WITH clause. This new SQL construct can improve the performance of complex SQL queries which utilize the same subquery in multiple places.

Adaptive Direct IO Operations

Oracle has supported direct IO operations since Oracle7, Release 7.1 (in 1993). Direct IO operations increase performance of table scans, index scans, and bulk writes by bypassing the buffer cache. Further performance advantages are realized by using asynchronous IO operations (which have been support in Oracle since Oracle7, Release 7.3). In Oracle9i, these IO operations have been further enhanced. Oracle9i will make even more efficient use of direct , asynchronous IO by dynamically adjusting the number of IO buffers to ensure that asynchronous direct IOs are executing as efficiently as possible.

Scalability

List Partitioning

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. Partitioning is a key technology for data warehousing, where large tables are commonplace. Oracle's partitioning capabilities have been enhanced in Oracle9i with the addition of a new partitioning scheme, list partitioning.

This 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 partitioning in Oracle9i) may 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 by the addition of new hardware resources.

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.

Enhancements to Parallel Query

Several internal improvements have been made to parallel query in order to improve performance and scalability. Oracle dynamically subdivides the work of a single query so that it can be parallelized; this dynamic parallelism allows Oracle to execute any query with any degree of parallelism. Enhancements have been made to Oracle's internal parallelization strategies to provide even finer granularities of parallelization in order to enhance dynamic load-balancing during parallel operations.

Oracle9i also provides enhancements to improve the performance of inter-node parallel query. Inter-node parallel query is for queries on clustered or Massively Parallel Processor (MPP) hardware configurations using Real Application Clusters, in which a single query is parallelized across multiple hardware node.

Both of these enhancements are internal in nature, and are incremental improvements to Oracle's existing capabilities.

Manageability

Many data warehouses today have become established, important sources of information with their corporations. As these data warehouses have matured, more and more users have begun to directly access the data warehouse. Oracle9i addresses these growing workloads with enhanced capabilities for managing large numbers of users on a data warehouses, by ensuring that:

- An appropriate amount of resources is allocated to each query
- The throughput of the entire warehouse platform is maximized
- The warehouse administrator and users can view the status of ongoing jobs
- The database can automatically abort or queue queries, based upon conditions pre-specified by the database administrator, in order to maintain optimal system load

Oracle9i introduces several new features to support these requirements. One significant new feature is the Automatic Memory tuning (described above). Other features are described below.

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 for any operation submitted by a user in a given group.

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

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.

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.

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 by database administrator's using 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.

Execution plans in cache

The execution plans of all current queries, along with other runtime statistics, are now stored in Oracle's internal cursor cache. This will allow database administrators to view and/or capture the execution plans of queries which are currently executing in the database. This feature will be used not only by database administrators but will also be used by management tools in order to gather better explain plan information.

Summary Advisor and Materialized View Manageability

The summary advisor, introduced in Oracle8i, has been enhanced in Oracle9i. The summary advisor makes recommendations, based on schema characteristics and previous workload history. In Oracle9i, the summary advisor has been enhanced so that it supports a broader class of schemas and so that database administrators can specify their own workloads as input to the summary advisor.

Additionally, other enhancements have been made to materialized views so that it is simpler to manage environments with materialized views. A new query-rewrite mode has been introduced, in which a query must use a materialized view or the query will abort; this mode is introduced to ensure that end-users do not execute queries against large volumes of detail data.

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.

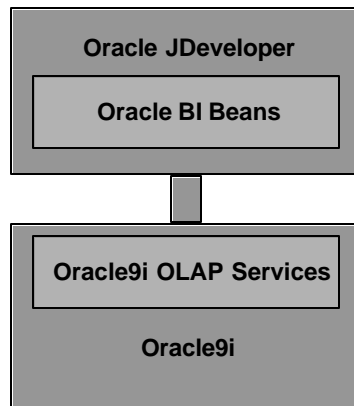
Oracle9i is the industry's first 'business intelligence platform'. The benefit of a business intelligence platform is integration—specifically the integration of the data—so that the same server infrastructure can be leveraged for all tasks which involve processing of large amounts of data.

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 API's 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

Oracle9i is the first and only OLAP-ready relational database, supplying efficient resolution of OLAP queries without adding administrative complexity. Oracle9i OLAP, an option to Oracle9i Enterprise Edition, provides valuable insight into business operations and markets using features previously found only in specialized OLAP databases. Because Oracle9i OLAP is fully integrated into the relational database, all data and metadata is stored and managed from within Oracle9i providing superior scalability, a robust management environment, and industrial-strength availability and security. A new API in Oracle9i, the Java OLAP API, provides the means of expressing complex multidimensional queries, and is designed for building internet-ready OLAP applications. Oracle Business Intelligence (BI) Beans and Oracle JDeveloper (both part of Oracle9i Developer Suite) provide a complete and highly productive development environment on top of Oracle9i OLAP.



OLAP API

As an object-oriented, platform-independent, network-based and secure language Java is fast superseding C++ and Visual Basic as the language of choice business intelligence applications. Oracle9i's Java OLAP API is designed from the ground up for Java and the Internet. Using Java, application developer can write applications, applets, and servlets that provide the means for deploying applications over the Internet to large, distributed user communities on a variety of devices.

Oracle9i's OLAP API is a Java object-oriented API which provides the following key benefits:

- Designed from the ground up to support Internet applications
- Object-oriented: provides encapsulation, abstraction and inheritance
- Leverage multidimensional metadata model
- Provides multidimensional cursors

- Supports a complete set of OLAP calculation functions
- Provides a simple mechanism for expressing complex OLAP calculations (which can be difficult to express in SQL)

Oracle's Business Intelligence Beans are built using Oracle9i's Java OLAP API.

New Relational Features to Support OLAP

With Oracle9i OLAP, all data resides in the relational database, and the vast majority of calculations are satisfied entirely using SQL. Many of the key ingredients for successful OLAP processing using SQL (such as fast cell access, efficient aggregation, and management of summary data) was already present in Oracle8i in features such as bitmap indexes, star-query optimizations and materialized views.

However, in order to meet the demanding requirements of OLAP processing, Oracle9i contains substantial enhancements to both the functionality and performance of SQL. Specifically, Oracle9i provides an extensive set of SQL capabilities for new types of analytic functions, as well as substantial enhancements for aggregation.

SQL Analytic Enhancements

For analytic functions, Oracle9i builds upon the functions introduced in an earlier release of Oracle8i that included calculations for:

- Ranking ("find the top 10 sales reps in each region")
- Moving-window aggregates (cumulative sums and moving averages)
- Period-over-period comparisons ("what is the percentage growth from January, 2000 over January, 1999?")
- Ratio-to-report ("What is January's sales as a percentage of the entire year's?")
- Statistical functions (linear regression, correlations)

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

Oracle has substantially enhanced both the functionality and performance of SQL to address the requirements of typical OLAP operations.

- Histograms - This function 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

Aggregations are also a key component of OLAP operations. 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).

Grouping sets are further enhanced with the concept of **concatenated grouping sets**. Concatenated grouping sets offer a concise way to generate larger combinations of groupings. Groupings specified with concatenated grouping sets yield the cross-product of groupings from each grouping set. The cross-product operation enables even a small number of concatenated groupings to generate a large number of final groups.

Here is an example of concatenated grouping sets:

```
GROUP BY GROUPING SETS (month, year), GROUPING
  SETS (region, country)
```

The SQL above defines the following groupings:

(month, region), (month, country), (year, region) and (year, country)

This example illustrates one of the most important uses for concatenated groups sets in an OLAP environment: to generate the aggregates needed for a hierarchical cube of data. This is a very basic operation in OLAP environments, and concatenated grouping sets provides an intuitive syntax for this operation.

Leveraging both Express Server technology and Oracle's relational database capabilities, Oracle OLAP provides a robust, scalable platform for delivering internet-ready analytic applications.

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

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

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

Oracle9i Data Mining provides the Naive Bayes data mining algorithm for making predictions and classifications. This algorithm is applicable to a variety of data mining problems and provides high accuracy. By finding patterns in data, companies can 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. 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.

Finding Associations: Association Rules

Oracle9i Data Mining also provides the Association Rules data mining algorithm to detect 'associated' or co-occurring events hidden in databases. 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 opens the door for integrating sophisticated data-mining capabilities, once the domain of specialized servers, into mainstream business intelligence applications.

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 provides a toolkit of ETL features, to address all aspects of the ETL process. These features 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

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

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 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

Oracle9i 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, Oracle9i 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 Oracle9i from a relational database used primarily for warehouse query-processing to a complete business intelligence platform capable of supporting all business intelligence workloads.



Oracle9i e-Business: Business Intelligence
October 2000
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

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 Oracle Corporation
All rights reserved.