

Oracle9i OLAP: A Scalable Web-Based Business Intelligence Platform

*An Oracle White Paper
April 2001*

Oracle9i OLAP: A Scalable Web-Based Business Intelligence Platform

EXECUTIVE SUMMARY	3
INTRODUCTION	4
A REVIEW OF THE BUSINESS INTELLIGENCE MARKET.....	6
Types of Business Intelligence Applications	6
Additional Trends in Business Intelligence.....	8
Traditional Analytic Servers	10
ORACLE9i - THE PLATFORM FOR ANALYTICAL APPLICATIONS.....	11
Oracle9i - A Single Database Platform	12
ORACLE9I OLAP	13
Adding Value to the Data Warehouse.....	14
Supporting Enterprise Analytical Applications.....	14
Overview of Oracle9i OLAP Architecture.....	19
Achieving Both Scalability and Performance	23
DEVELOPING AND DEPLOYING APPLICATIONS	28
Oracle Business Intelligence Beans.....	29
Java OLAP API or Business Intelligence Beans?	33
Development Environment.....	34
Deploying Applications Built with Business Intelligence Beans	36
CONCLUSION.....	36

On-line Analytic Processing in Oracle9i: A Scalable Web-Based Business Intelligence Platform

EXECUTIVE SUMMARY

In most organizations the deployment of business intelligence applications is bifurcated along the lines of technology. Less sophisticated reporting and ad-hoc query tools are usually deployed using data warehouses and SQL based reporting tools while more sophisticated analysis and planning applications are deployed using other specialized databases and tools.

At present, the use of different technologies for these different classes of applications is reasonable. Relational data warehouses are scalable and support very large data sets. Separate analytical databases provide necessary analytical functionality that is not available with relational databases.

This separation between data warehouses and analytical databases is, however, expensive. The cost of maintaining separate analytical databases involves additional hardware and database administrators. The process of replicating data into analytical databases also increases the amount of time it takes to make data available to analysts and decision makers.

As the size of data sets increase, these problems become more acute. It takes longer to replicate the data and query performance tends to degrade. These issues translate into opportunity costs associated with less timely and less effective analysis.

Although many of the less sophisticated business intelligence applications are now deployed using the Internet, most of the more capable business intelligence applications are deployed using client server architectures which offer richer graphical user interfaces.

The costs associated with client server deployment are well known: it is difficult to deploy and maintain applications. As a result, organizations tend to deploy business intelligence applications to relatively few users. In most organizations fewer than 5 percent of employees have access to business intelligence applications and they data they make available.

Many more employees could make use of this information if only they had access to business intelligence applications. This represents a tremendous opportunity cost because these employees lack the insight afforded by business intelligence applications and are therefore less effective. The larger and more geographically distributed the organization is, the larger this problem becomes.

It is possible for a highly motivated organization to overcome these barriers to insightful analysis. Some products do exist that offer useful analytical capabilities

using a data warehouse. They tend to offer fewer analytical functions and to perform poorly as compared to specialized analytical databases, but they are useful. It is also possible for an organization to absorb the costs associated with separate analytical databases. In each case, the organization will need to assemble the solution from a number of different vendors and endure the pains associated with acting a systems integrator.

With Oracle9i, Oracle eliminates these barriers by providing an analysis ready Oracle database and Internet based application development tools. These allow organizations to effectively analyze data in the data warehouse without requirement massive data replication and to easily deploy business intelligence applications to large, geographically distributed user communities.

Oracle9i OLAP, part of the Oracle9i database, provides the means to analyze data in large data warehouses. The Oracle Business Intelligence Beans, Oracle JDeveloper, and Oracle Internet Application Server provide the ability to build and deploy sophisticated business intelligence applications on the Internet to large and distributed user communities.

Because Oracle 9i OLAP, Oracle9i Business Intelligence Beans, Oracle JDeveloper, and Oracle9i Internet Application Server are part of an integrated product line your organization avoids the costs associated with playing the role of a system integrator.

The remainder of this paper discusses this solution, focusing on Oracle9i OLAP and the Oracle Business Intelligence Beans.

INTRODUCTION

Business intelligence applications are now a driving force within leading organizations. They provide the ability to gain insight into the organization's markets and internal operations and allow organizations to react quickly to a changing environment and to plan for the future. There are three primary factors that have acted to drive the adoption of business intelligence applications:

- Data warehouses are now widely used to store data sets suitable for analysis.
- The Internet offers the promise of allowing large numbers of users access business intelligence applications.
- The use of business intelligence applications has become a competitive necessity.

Prior to the widespread use of data warehouses, most organizations did not have an IT infrastructure to capture, cleanse and store large volumes of data that was well structured for analysis. Data warehouses have made it practical to make available large volumes of detailed information about the organization's markets and internal performance measurements. Users are demanding the means to make that data useful.

With the Internet, it is now possible to deploy business intelligence applications to large, geographically distributed user communities. Organizations need to distribute information both within the enterprise and externally to suppliers and customers.

Most organizations deploy relatively unsophisticated static and ad-hoc reporting tools against the data warehouse. They are easy to build and many can be deployed over the Internet. These types of applications do not, however, fully leverage the investment in the data warehouse by allowing truly insightful analysis. Instead, more advanced analysis and planning applications usually require specialized analytical databases.

Because specialized analytical databases are costly to deploy, their use is limited. They are typically used with smaller data sets and made available to only to selected managers and analysts within an organization; business intelligence applications requiring specialized analytical databases are rarely available to large numbers of line managers and non-analysts. Instead they simply go without good analysis and planning applications and use technologies such as spreadsheets to analyze small amounts of data.

The opportunity costs are staggering. Consider, for example, the cost of incorrectly predicting the demand for a new product. If an organization does not manufacture enough product sales opportunities are missed. If too much product is manufactured, capital is tied up in inventory and margins are pressured. What is the cost of not being able to predict the organization's profitability in the current fiscal quarter? If a problem isn't observed, corrective action can't be taken. If an opportunity isn't seen, it can't be seized.

Another key driver in business intelligence is the fact that the analysis and planning process includes people outside of the organization. A distributor might allow a manufacturer to analyze the distributor's sales order and shipments information for products that manufacturer supplies in order to allow the manufacturing to plan production of these products.

The trend toward more sophisticated analytical applications, the need to distribute these applications broadly both within the organization, and the need for collaborative analysis and planning accentuates the need for an enterprise platform that can provide a wide range of analytical services over the Internet.

Oracle9i provides a scalable, Internet based platform for developing and deploying the complete spectrum of business intelligence applications. The remainder of this paper will discuss different types of business intelligence applications, the analytical needs of these applications, how Oracle9i supports these analytical requirements, and how these applications are easily deployed on the Internet.

A REVIEW OF THE BUSINESS INTELLIGENCE MARKET

Before discussing Oracle9i, it is useful to review the market for analytic servers. This section discusses the four types of business intelligence applications and the current architectures that are used to support these applications.

Types of Business Intelligence Applications

Today's enterprise organizations require a variety of different business intelligence applications serving many different user communities. These applications generally fit into one of four categories:

- Reporting applications
- Ad-hoc query and reporting
- Multidimensional analysis
- Planning

Oracle9i supports the requirements of each type of application.

Reporting Applications

Reporting applications tend to provide static or parameterized reports. The audience tends to be broad, for example all sales managers within an organization.

Reporting applications that have minimal analytical requirements are typically based on relational databases and use SQL as the query language. Executive information systems (EIS) tend to have more complex analytical requirements often use specialized analytical databases.

Although the analytic requirements of reporting applications have tended to be minimal in the past, it is becoming more common for reporting applications to provide information that is the result of a complex analytical calculation or planning process. For example, an organization might wish to use a reporting application to distribute a final corporate budget. The reporting application cannot create the budget, but it can distribute the final budget once the budget data is placed in a database.

Ad-Hoc Query and Reporting

Ad-hoc query and reporting applications offer the user a high level of interaction. They allow the user to explore the data by using a variety of data selection and navigation techniques. Ad-hoc query and reporting applications typically are based on relational databases and offer a limited, but very useful, set of analytic capabilities. They are often, but not always, based on a multidimensional data model.

Ad-hoc query and reporting applications running on a relational database use SQL and typically support 'unidimensional' queries such as 'what are my top 10 customers' and 'how does this month's sales compare with sales a year ago'. This

range of analytic functionality is often sufficient for large user communities such as salespeople.

To date, ad-hoc query and reporting applications are typically the most sophisticated applications deployed directly against a data warehouse.

Oracle's Global Sales Reporting and Tracking System is an analytical application deployed using a data warehouse. Thousands of Oracle salespeople, market analysts, and managers worldwide use this application to analyze license and service revenues.

Users analyze data along a number of dimensions such as customers, products, lines of business, geographic areas, and time. They can use a variety of data selection techniques such as rankings, data exceptions, hierarchical relationships, and time series selections to navigate the data and explore trends and opportunities. The ability to create virtual dimension members and virtual facts allows analysts to create data needed for insightful analysis.

Because the global sales reporting and tracking system data is maintained in an Oracle data warehouse it offers timely access to data, is manageable, and offers access to the largest possible user community using a variety of reporting tools.

Analytical Applications

Analytical applications also support ad-hoc exploration of data, however they answer much more complex questions. Although the following query might appear to be rather complex, it is actually a typical question that might be asked by a marketing analyst:

'What was the percent change in total sales, as compared with the period a year ago, for each of the top 10 products, for each of my top 10 customers, based the percent change in total sales for each customer this year to date versus the equivalent year to date period a year ago'.

Note that this query is multidimensional. That is, calculations in the query occur along more than one dimension. Period ago and year to date calculations are made along the time dimension. Rankings occur along both the customer and product dimensions. The product ranking is nested within the customer ranking. A calculated measure is nested within the product and customer rankings.

An examination of a market or financial analyst's report catalog would typically reveal reports that include significantly more complex queries. For example, they might include forecasts and complex nesting such as rankings over virtual page dimension members which themselves are rankings.

In the past, most ad-hoc analytical applications have used stand alone analytical databases to support the required complex multidimensional queries.

Planning Applications

Planning applications allow the user to predict outcomes or results. They allow the user to ask questions such as "how profitable will my company be this quarter", "how will a price change influence unit sales and net margin" and "how many products should be manufactured to meet demand this month".

Planning applications are very different from query and reporting applications because they generate new data using analytical tools such as models, forecasts, specialized aggregation and allocation methods, and scenario management tools.

Common examples of planning applications include corporate budgeting and financial analysis, and demand planning systems.

Corporate budgeting and financial analysis systems generally serve three functions:

- They allow user to analyze past performance and answer questions such as 'what product lines are most profitable' and 'why have profits increased while cash flow has not'.

Oracle uses Oracle Financial Analyzer for global budgeting and financial analysis. Using Financial Analyzer, Oracle maintains one worldwide view of budgets and financial performance. The budgeting process allows Oracle management to establish profitability goals and allocate budgets to line managers. Line managers make adjustments using a variety of scenario management, forecasting, and modeling tools and submit completed budgets to a centralized database. These budgets are then consolidated and compared with profitability goals. The process can iterate frequently until a finished budget is produced.

Throughout the fiscal year, Oracle analyzes actual expenses and revenues to identify budget variances. Throughout each fiscal quarter, Oracle forecasts profitability and earnings per share. Oracle can predict whether earnings per share will be below, on, or exceed the targeted earnings per share. If earnings per share are below target expenses can be cut or deferred or incentives can be offered to improve revenue. If earnings per share exceed targets, Oracle sales managers can be informed that certain revenue can be deferred to the next fiscal quarter. This ability to predict earnings and take corrective actions to meet expectations is extremely critical in today's capital markets.

Oracle Financial Analyzer is based on Oracle Express Server, the predecessor to Oracle9i OLAP. A new version based on Oracle9i will be available next year.

- Budgeting systems allow organizations to build revenue and spending plans and manage towards profit goals. A typical system captures revenue data from sales force automation applications and spending data from general ledger applications. Spending can then be forecasted and profitability can be predicted based on the revenue estimates.

These estimates are typically aggregated for review by senior management. Sales quotas and profitability targets are then provided by senior management and allocated to line managers. This cycle might be repeated several times until a budget that meets revenue and profitability goals is reached. This function of a corporate budgeting system allows managers to answer the question 'what level of spending and investment is appropriate given certain revenue expectations and profitability goals'.

- They allow financial analysts to model the effects of change on the financial plan. For example, a financial analyst can model the effects of currency value fluctuations on revenues, costs, and profitability and prepare alternative budgets and investment plans.

Demand planning systems allow market analysts and product managers predict market demand based on sales history, sales pipelines, demographic factors, promotional plans, and other factors such as pricing models. They can model different scenarios that forecast product demand based on this type of analysis. These results can then be fed into the manufacturing processes.

Additional Trends in Business Intelligence

There are two additional trends in business intelligence applications:

- The expanded role of collaboration in analysis and planning.
- The convergence of operational and business intelligence applications.

These trends are discussed in the following topics.

Collaborative Analysis and Planning

In addition to the calculation capabilities of business intelligence applications, the ability to support large, geographically distributed user communities in a collaborative environment is key to the success of the application.

In order to support a collaborative environment for business intelligence applications, the application must:

- Provide access to a centralized data store
- Support sharing of user defined analytical objects
- Provide a convenient means for consolidating data
- Deploy applications in the Internet

In Oracle9i a centralized metadata and data store provides all members of the analysis and planning process with a common, up to date view of the data. All users have access to the same data model and the same data. Consider, for example, the confusion that can result if salespeople in different regions of the world analyze data using different dimensions and hierarchies or if 'unit sales growth' has one definition in Europe and a different definition in the North America. Or consider a budget process where managers did not have up to date revenue projections or budget allocations.

Users are most productive when they can share analytical objects such as reports, charts, and data selections. Think about the difference between sharing a printed copy of a report versus a live report. A live report allows the user to continue the analysis process while a printed copy conveys a limited set of information. Enterprise business intelligence applications that support centralized object repositories allow collaboration.

Users of planning applications must be able to quickly submit newly entered or generated data to a centralized repositories for consolidation, analysis, revision, review and approval. For example, line managers need to be able to submit budgets to a centralized data repository for review by their managers and budget analysts. This process needs to be efficient enough to allow the enterprise to make many iterations of the plan within a short period of time.

The Oracle9i Business Intelligence Beans allow applications to save user defined objects in a centralized catalog. Objects in this catalog can be shared among users and applications to provide support for collaborative analysis.

The requirement for collaborative analysis highlights significant shortcomings of applications that are reliant on desktop data stores such as spreadsheets. Desktop data stores lack instantaneous access to data produced by other users, do not support sharing of analytical objects, and require cumbersome consolidation processes.

Convergence of Operational and Business Intelligence Applications

Analytic and planning functions are typically provided in applications that are separate and apart from operational applications. For example, accounting systems provide basic reporting facilities but do not offer analysis tools. Users who need to analyze data use a different application specific to that purpose. Different applications are required because the technology supporting those applications are significantly different - accounting systems are not based on specialized analytical databases; analytical applications are not typically based on relational databases.

With Oracle9i it becomes possible for one database platform to provide support for both operational and business intelligence applications. This will cause a convergence of operation and business intelligence applications. For example, one application can support accounting, budgeting, and financial analysis. This one application will be more functional and less costly to maintain.

Traditional Analytic Servers

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. Replication to other specialized databases was not required. 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 very limited and performance tended to lag that of multidimensional databases.

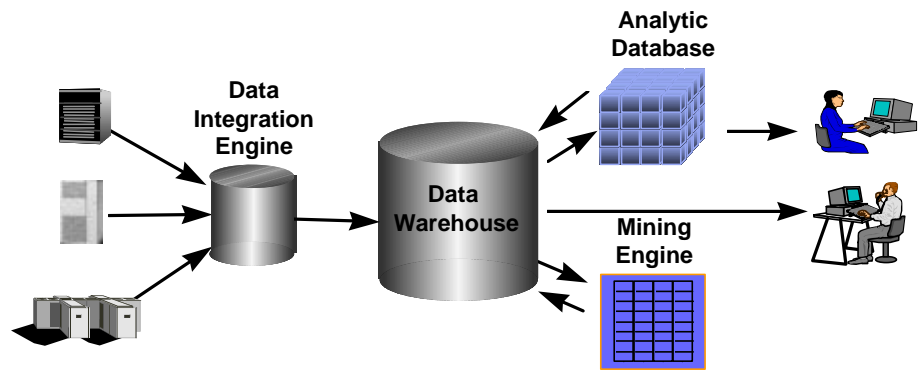
Specialized analytical databases offered a complete set of analytical functions and tended to provide better query performance, but maintaining a analytical database represented a significant expense. An analytical database required data replication and a separate management infrastructure. Data replication 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.

To make matters more complicated, RDBMS based data warehouses tend to scale very well in terms of data set sizes, while analytical databases tend to support only limited data set sizes. So not only did a developer of analytic applications need consider the types of calculations that are required and the required performance characteristics, but also the size of the data set.

As a result, developers tended to write applications that had only limited analytical requirements using SQL and applications that had more demanding analytical requirements using analytical databases. Organizations would need absorb the cost of managing a separate database system and replicated data.

Relational OLAP (ROLAP) technologies provided support for some analytic calculations using SQL, however these met only limited success due to limitations in analytical power and performance.

The following illustration represents a typical database infrastructure used to support a variety of analytical applications.



In this typical environment, data is accessed from source systems (for example, a general ledger or sales order system), processed through a data integration engine, and loaded into a data warehouse in a relational database. Reporting and ad-hoc query and reporting applications could query the data warehouse directly and offered the most timely access to data.

Users of analytic applications built using an analytic database would need to wait until data was replicated in the analytic database. This created situations where data in the multidimensional database was stale and users needed to wait - sometimes hours or even days - for data to be replicated in the other database.

From the DBA perspective, there are:

- Three technologies to be learned (the data integration tool, the relational database, and the analytic database)
- Three metadata repositories to be managed
- Three management processes
- Two data stores (the warehouse and the multidimensional database)

Managing these three processes was very costly, both in terms of managing the system and the opportunity costs associated with less effective analysis due to the inability to immediately access fresh data.

Although separate analytic databases were costly, they were still often the best solution for analytical applications because they provided significant value to the organization in the form of added analytical capabilities and excellent query performance. Most organizations could justify the added costs since analytical capabilities and performance characteristics offered by analytic databases were more important than the added costs.

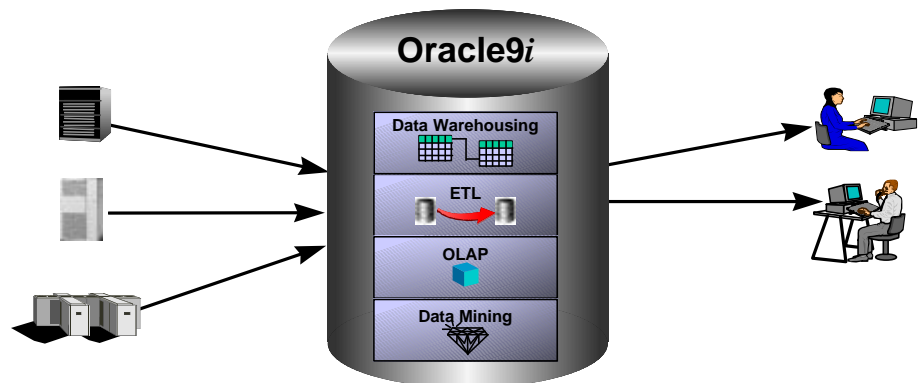
ORACLE9i - THE PLATFORM FOR ANALYTICAL APPLICATIONS

Oracle9i challenges the traditional view of the analytic server market by offering revolutionary function and performance in a more flexible, cost effective and manageable platform. Oracle9i offers:

- A complete range of analytic functionality that supports the complete spectrum of reporting, analytical, and planning applications
- Performance previously unavailable to data warehouse based analytic applications
- A flexible and manageable data platform that eliminates the need for wholesale data replication
- A more cost effective platform for analytic applications
- A platform for Internet based applications

Oracle9i - A Single Database Platform

Oracle9i 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 by both extending the analytical capability of SQL and by offering Oracle9i OLAP with the relational database.



Oracle9i extends and integrates ETL, OLAP, and data mining functionality into the Oracle database. The result is a platform that is easier to manage and more cost effective, supports the full range of analytical applications, and provides timely access to updated information.

In Oracle9i, there is

- A complete set of analytical functions
- One database and management tool set for ETL, data warehouse and analysis
- Centralized metadata
- A single permanent data store
- Built in support for the Internet

Oracle9i provides a full range of analytical functions to support all types of analytic applications. These functions range from SQL OLAP functions to highly specialized predictive analysis functions. Oracle9i offers traditional SQL based tools and applications new analytical opportunities with SQL OLAP functions and provides multidimensional analytical applications with a data warehouse based OLAP server to support analytical and planning applications.

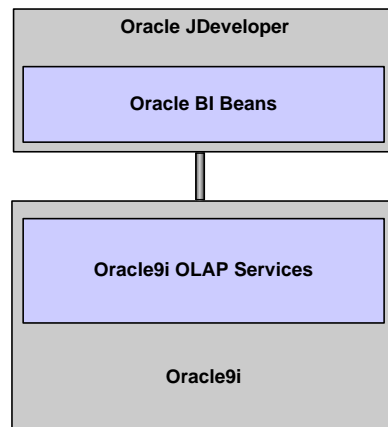
Since both SQL based and OLAP based applications can access the data warehouse they all share a single data repository and a common management infrastructure. This provides the most cost effective solution since the data is accessible to the most decision makers and the costs of managing the warehouse are minimized.

ORACLE9i OLAP

Oracle9i OLAP offers a complete platform for analytical applications ranging from standard reporting applications through planning applications. It is highly integrated with the Oracle relational database to minimize the need for data replication and to provide the extremely fast query response times normally associated with analytical databases. It is designed from the ground up for the Internet. It is easy to manage and is cost effective.

The Oracle9i product line provides a full software stack for developing and deploying enterprise-scale business intelligence applications. Each component of the product line is optimized to meet the demands of analytic applications. The major components include:

- Oracle Business Intelligence Beans and Oracle JDeveloper
- Oracle9i OLAP as a facility within Oracle9i
- Oracle9i RDBMS, the foundation of the platform



Business Intelligence Beans are Java Beans specifically designed to support the development of business intelligence applications. The Business Intelligence

Beans are highly integrated with Oracle JDeveloper9i and offering extremely productive development.

Oracle9i OLAP provides centralized analytic processing in a scalable and secure environment. Oracle9i's Java OLAP API provides the means of expressing complex multidimensional queries and a productive and powerful API for application development. Oracle9i OLAP also provides predictive analytical functions such as modeling, forecasts, and scenario management using a multidimensional data type known as an *analytic workspace*.

The Oracle9i relational database acts as the permanent data store, is the repository for all Oracle9i OLAP and Business Intelligence Beans metadata, provides summary management services, and provides SQL OLAP functions.

Adding Value to the Data Warehouse

Many organizations do not use analytical and planning applications that require analytic databases due to the costs associated with managing a separate data store. Instead, they access the data warehouse using batch reporting systems and ad-hoc reporting applications. Although manageable, the limited analytic capabilities prevent the organization from fully leveraging the data warehouse. They are incurring the opportunity cost of not fully understanding their customers, the marketplace, and the performance of their organization.

Since Oracle9i OLAP supports sophisticated analysis within the context of a data warehouse, organizations enjoy many benefits, including:

- Support for queries demanding complex multidimensional calculations
- Excellent query response times
- Support for planning applications, which require analytic features such as forecasting, models, consolidations (or aggregations), and support for scenario management
- Support for large numbers of geographically distributed users

Supporting Enterprise Analytical Applications

Specialized analytic databases do an excellent job providing support for complex analytical queries and query performance, however they do not typically meet all of the requirements needed for enterprise applications. All applications - whether they focus on analysis or not - share some common requirements:

- Scalability
- Open access
- Security
- Manageability

- High Availability

Scalability

Oracle9i OLAP stands out for its scalability. In today's competitive environment, there is tremendous growth along three dimensions of analytic applications: number of users, size of data, complexity of analysis. There are more users of analytical applications and they need access to more data to perform more sophisticated analysis.

A telephone company, for example, might require a customer dimension to include detail such as all telephone numbers as part of an application that is used to analyze customer turnover (churn). This would require support for multi-million row dimension tables and very large volumes of fact data.

Since Oracle9i OLAP can access data directly from the Oracle database, Oracle's architecture and scalability features translate directly into scalability for Oracle9i OLAP. Oracle9i supports multi-terabyte data sets using a proven architecture that includes parallel execution and partitioning, as well as support for NUMA and clustered systems.

Features such as Oracle's Data Resource Manager help DBAs support large numbers of concurrent users. Data Resource Manager provides a mechanism for allocating the resources of a data warehouse among different sets of end-users.

Consider an environment where the marketing department, the sales department and some executives share access to a data warehouse. Using the Database Resource Manager, a warehouse administrator could specify that:

- The marketing department receives at least 70 percent of the CPU resources of the machine and that only 50 concurrent queries are allowed (others are queued)
- The sales department receives 40 percent of the CPU resources and that only 25 concurrent queries are allowed (others are queued)
- Executives receive 80 percent of CPU and any number of concurrent queries are allowed
- Queries that are estimated to take longer than 10 minutes are put into a lower priority queue

Proactive query governance and automatic query prioritization features allow the DBA to create a user management scenario where executives have the highest priority and sales the lowest priority, but no one user or group could completely dominate the system with very long running queries.

Open Access

Oracle9i provides open access through the Internet, open APIs, and SQL. Because Oracle9i OLAP is designed for the Internet, applications anywhere on the

Internet can access Oracle9i (security policies allowing). Oracle9i's Java OLAP API is published and available to any application developer. Data stored in the data warehouse is accessible by both SQL and Java OLAP API clients, thereby leveraging the investment in the warehouse over the largest possible user community.

Security

The expanded number of users and web-based access of enterprise level business intelligence applications calls for the strongest possible security. Just as Oracle developed robust features for scalability, manageability and backup and recovery, Oracle has created industry-leading security features. The security features in Oracle have also reached the highest levels of U.S. government certification for database trustworthiness.

There are two aspect of security that are of interest to analytic applications: authentication and authorization.

Authentication

All users of applications based on Oracle9i OLAP are authenticated using either Oracle database authentication or LDAP. When database authentication is used, the user is authenticated against the user catalog in the Oracle data dictionary.

Oracle9i makes extensive use of LDAP functionality in its security features. For instance, Oracle enables centralized user and privilege management in LDAP-based directories such as Oracle Internet Directory. Support for standards-based public key infrastructure enables existing PKI credentials to be shared by an Oracle Wallet. Wallets can be downloaded from LDAP directories to support mobile or "hot-desked" users. Oracle Internet Directory also supports an IETF LDAP standard for handling both generally used and customized password encryption schemes.

Authorization

It is typical for users of business intelligence applications to be allowed access to only a subset of data within the data warehouse. Users might be limited in the types of data they are allowed to see (for example, certain facts) or they might be limited the actual data they can access (for example, certain dimension members).

A common scenario might be one in which a data warehouse contains both sales and financial data. Salespeople usually do not have access to financial measures and often do not have access to all products or all geographical areas related to the sales measures. As a result, the DBA needs the means to limit what elements of the multidimensional model are exposed to the salespeople and what dimensions members they can see.

Access to elements of the multidimensional model are controlled through access privileges to the *OLAP catalog*. The OLAP catalog describes the multidimensional model (for example, dimensions, levels, attributes, and measures). Because

elements of the model belong to a schema in the Oracle database, DBAs can control access to the model by granting privileges in the relational database to objects referenced by the model. For example, to provide access to a dimension, the user must be granted SELECT privileges to the dimension table.

Oracle's Virtual Private Database feature enables cell-level security for OLAP users. This provides the means to control what dimension members and fact data a user can see. The Virtual Private Database is a server-enforced, fine-grained access control.

Since security policies are defined once, in the database, the cost and burden of enforcing security policies is removed from individual applications. By enforcing both authentication and authorization in the database, no matter how the user accesses the data, security remains in force regardless of whether the access method is a SQL based tool or through an Oracle9i OLAP application.

Virtual Private Database enables a single database to logically separate data sets with the security that formerly required multiple physical databases.

Manageability

Oracle Enterprise Manager (OEM) provides a centralized, comprehensive management tool. OEM enables administrators to monitor all aspects of the database, including Oracle9i OLAP.

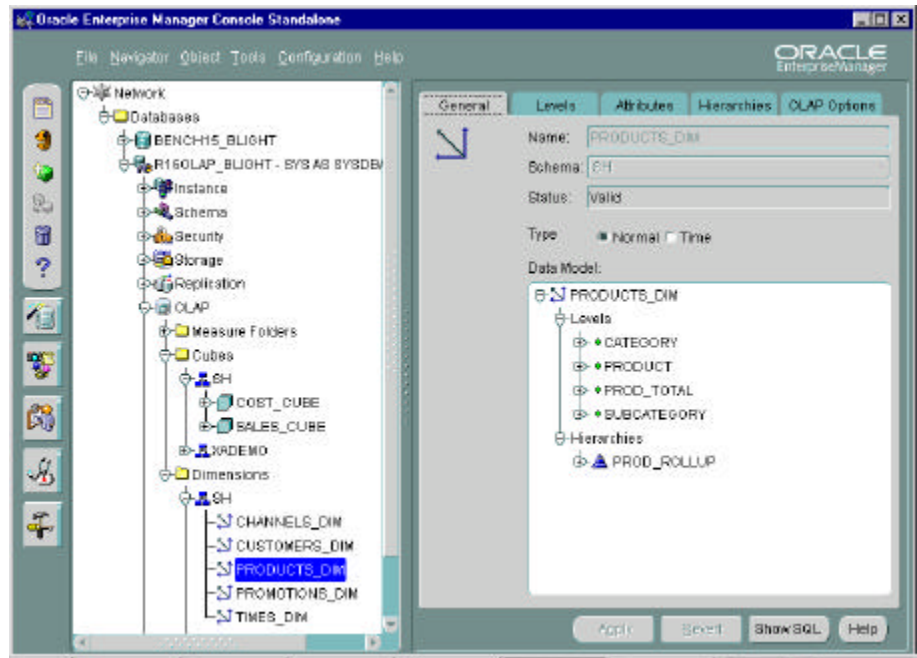
Oracle Enterprise Manager provides management services to Oracle9i OLAP including:

- Instance management
- Session management
- Configuration management
- Data modeling
- Performance monitoring
- Job scheduling

Although Oracle9i OLAP is most often deployed on the same server as the relational database, it is a separate process that can be deployed on a different server as a middle tier. Since an Oracle9i OLAP instance is a separate process, Oracle9i OLAP Instance Manager is provided to start and stop instances and to manage configuration settings. Instance manager also allow the DBA to manage connections to Oracle9i OLAP through its' session manager feature.

The multidimensional model used by Oracle9i OLAP is managed using Oracle Enterprise Manager's OLAP tool. The OLAP tools defines the elements of the multidimensional model (for example, cubes, facts, dimensions, hierarchies, levels, and attributes) and maps them to the data sources in the data warehouse (dimension and fact tables).

In the following illustration we see the OLAP tool outline on the left side panel (note cubes and dimensions) and detail of a product dimension on the right side panel (note levels and hierarchies).



Performance monitoring facilities are provided by Oracle Enterprise Manager's performance monitoring tools. Job scheduling facilities are provided by Oracle Enterprise Manager's job scheduler.

Availability

The demands made on enterprise-level OLAP systems include higher availability requirements. Oracle9i includes many features that support high availability. One of the most significant is Oracle partitioning.

Partitioning allows management of precise subsets of tables and indexes, so that management operations affect only small pieces of these data structures. Since smaller portions of the database are off line at any one time, overall availability can be greatly increased. For instance, when data is added to a large table, the new rows can be placed in their own partition and the existing data remains available throughout the data load.

The indexes associated with the table could be managed similarly: indexes for the new data could be placed in their own partition. If existing indexes are unaffected by the new data, only the new data will need indexing and the existing indexes will always be available.

Backup and Recovery

As the importance of analytic applications grows, so does the need for bulletproof backup and recovery processing. Likewise, the acceptable recovery time for analytic applications shrink as more and more users depend on these systems. Oracle's heritage in enterprise applications means that it has long included a strong set of backup and recovery features. Oracle8 introduced an intelligent, server-managed infrastructure for backup, restore, and recovery tasks that enables simpler, safer operations at terabyte scale. Highlights of this technology include:

- Storage of details related to backup, restore, and recovery operations are automatically maintained by the server in a recovery catalog and automatically used as part of these operations. This reduces administrative burden and minimizes the possibility of human errors.
- Backup and recovery operations are fully integrated with partitioning. Individual partitions, when placed in their own tablespaces, can be backed up and restored independently of the other partitions of a table.
- Oracle includes support for incremental backup and recovery, enabling operations to be completed efficiently within times proportional to the amount of changes, rather than the overall size of the database.
- The backup/recovery technology is highly scalable and provides high performance interfaces to industry-leading media management subsystems. This provides for efficient operations that can scale up to handle very large volumes of data.

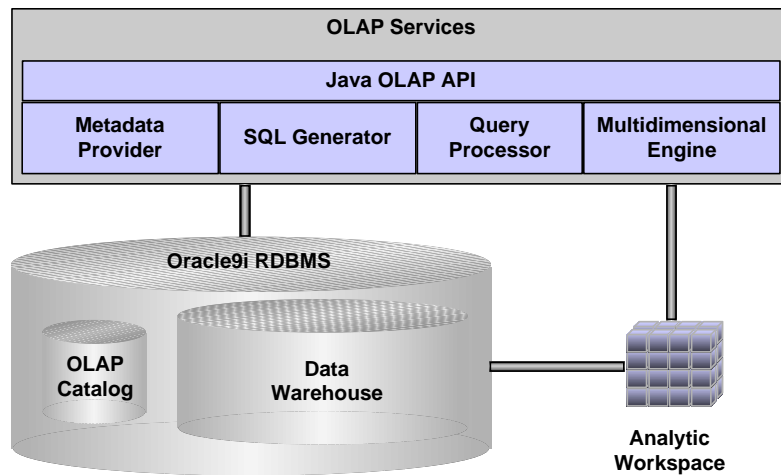
Oracle includes many other features ensuring high speed recovery. For instance, Oracle9i provides a two-pass recovery algorithm that ensures only the blocks which need to be processed are read from and written to data files. It also includes a new "mean time to recover" parameter which lets administrators more easily set an upper limit on crash recovery time. For preventing and handling disk corruption, Oracle9i adds features such as block media recovery, enabling only all but the corrupt portions of a table to stay online.

Overview of Oracle9i OLAP Architecture

Oracle9i OLAP is an integrated part of Oracle9i, yet provides the flexibility to work with both relational and multidimensional data sources. This provides Oracle9i OLAP with the ability to provide sophisticated multidimensional analysis of data within an Oracle data warehouse and to support specialized planning functions that are required by planning applications.

Software Components

The major components of Oracle9i OLAP are illustrated below.



The Java OLAP API provides the OLAP interface to Oracle9i. It processes the API calls, passes queries on to the query processor, and maintains multidimensional cursors.

The query processor determines the query execution plan. Depending on the data source - the relational database or an analytic workspace - the query processor chooses to use the SQL generator or the multidimensional engine to access data and perform calculations.

The metadata provider reads the metadata repository and represents the multidimensional data model and data sources to the Java OLAP API. There are two metadata providers in Oracle9i OLAP. One metadata provider is for the OLAP catalog and the other is for the analytic workspace.

The SQL generator generates SQL that is used to access data and resolve analytic queries in the relational database. The Oracle9i OLAP SQL generator is tuned specifically to the Oracle9i database. It uses a number of new features in Oracle9i to perform calculations and achieve superior performance.

The multidimensional engine manages data within analytic workspaces, a special data type that is used to solve certain predictive analytic calculations. The multidimensional engine supports the OLAP DML, a multidimensional data manipulation language that supports custom multidimensional functions and predictive functions.

Data Sources

Oracle9i OLAP supports two data sources: the Oracle9i relational database and Oracle9i OLAP analytic workspaces.

Oracle Relational Database

The Oracle9i relational database is the primary data source and generally acts as the permanent repository for all data within Oracle9i. Oracle9i OLAP accesses data in Oracle9i that is in a data warehouse environment, specifically a star schema.

Unlike other analytical servers based on hybrid (relational and multidimensional) databases, Oracle9i can resolve complex multidimensional queries directly in the relational database without requiring that data be cached in the multidimensional data store. As a result, it is perfectly suited for analytic applications requiring support for complex multidimensional calculations.

Analytic Workspaces

Analytic workspaces are multidimensional data files. They do not use Oracle data files as storage, instead they are separate files on the operating system. Analytic workspaces serve three specific functions:

- They provide Oracle Express Server customers a migration path (Oracle Express Server's multidimensional data files can be attached to an Oracle9i OLAP session as an analytic workspace)
- They support planning applications by supporting predictive analytic functions such as forecasting, modeling, and scenario management (what-if)
- They allow application developers to extend the Java OLAP API by defining custom multidimensional analytic functions using the OLAP DML

The OLAP DML is a stored procedure language specifically designed to support the definition of multidimensional calculations. (Although the languages are actually very different, you could think of the OLAP DML as being the multidimensional equivalent to PL/SQL.)

To use an analytic workspace, you cache data from the relational database into it using functions provided by Oracle9i OLAP. You can then manipulate data using the OLAP DML and expose data to applications through the Java OLAP API. The results of calculations made in the analytic workspace can also be committed back to the data warehouse for permanent storage.

Choosing a Data Source

Most applications will use the Oracle9i relational database as OLAP Service's data source since it offers many advantages, including:

- The Oracle database is a highly scalable data store capable of supporting multi terabyte data warehouses with excellent query performance
- It provides an open data store since both SQL and Oracle9i OLAP based applications have full access to data in the Oracle database
- An integrated security model
- Simplified management since all administration occurs within the context of the Oracle relational database

Planning applications that require predictive functions such as forecasting, modeling and scenario management will use analytic workspaces as temporary data source for work in progress. For example an application that requires solving a

complex financial model can use the analytic workspace to solve the model, but the results of the calculation can be committed to the data warehouse for permanent storage.

Data Model and Metadata

Oracle9i OLAP using a multidimensional data model. The multidimensional model contains the following elements:

- Measures
- Cubes
- Dimensions
- Levels
- Hierarchies
- Attributes

This multidimensional model is familiar to end users because it describes data in business terms. For example, end users ask questions such as how many units of a product did we sell in a particular geographic area during a specific time period. In this example, we have a measure (Units) and three dimensions (Product, Geography, and Time).

A *measure* represents factual data. Dollar Sales and Unit Sales are examples of measures. A measure corresponds to a fact column in a fact table, although it has a much richer set of metadata that describes it.

A *cube* describes a group of measures that shares common dimensionality and the relationship between fact tables and dimensions. For example a Sales cube might have several facts including Dollar Sales and Unit Sales that are dimensioned by Product, Geography, and Time. An Oracle9i Cube is a metadata object, it does not actually contain any data. A Cube's data is stored in a fact table and a materialized view.

Dimensions provide logical indexes into the data. Examples include Product, Geography, Time, and Distribution Channel. Dimensions are made of up levels, representing levels of summarization (for example Month, Quarter, and Year) that are organized into one or more *hierarchies*. Dimensions can also contain *attributes*, which are non-hierarchical properties (for example, color or size).

Metadata describing the multidimensional model used by Oracle9i OLAP is saved in Oracle9i's *OLAP catalog*. The OLAP catalog is closely linked to the Oracle data dictionary and provides Oracle9i OLAP with a high performance method of accessing multidimensional metadata. The OLAP catalog, containing both the multidimensional model and mapping to physical data sources, completes the definition of the star schema.

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 modern, Java object-oriented API which provides encapsulation, abstraction and inheritance. The Java OLAP API supports a more productive development environment as compared to string based query languages such as SQL and competing multidimensional APIs.

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 tables or analytic workspaces - or how to access the data.

This insulation allows the DBA complete freedom to manage physical storage in the relational database without disturbing Oracle9i OLAP applications. For example, the DBA is free to tune the database by changing the warehouse schema without being concerned for whether this will affect the applications ability to run.

The Java OLAP API also insulates the Oracle9i OLAP application developer from SQL. This has two important benefits:

- 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 Java OLAP API which is designed for the ground up for OLAP. They can leave the task of generating optimal SQL up to Oracle9i OLAP.
- The Oracle9i OLAP application developer writing to the Java OLAP API does not need to be concerned with updates to SQL in later releases of the Oracle database. As new analytical and performance features are introduced into the Oracle database, Oracle9i OLAP SQL generator will be updated and applications written to the Java OLAP API will enjoy the benefits without re-coding.

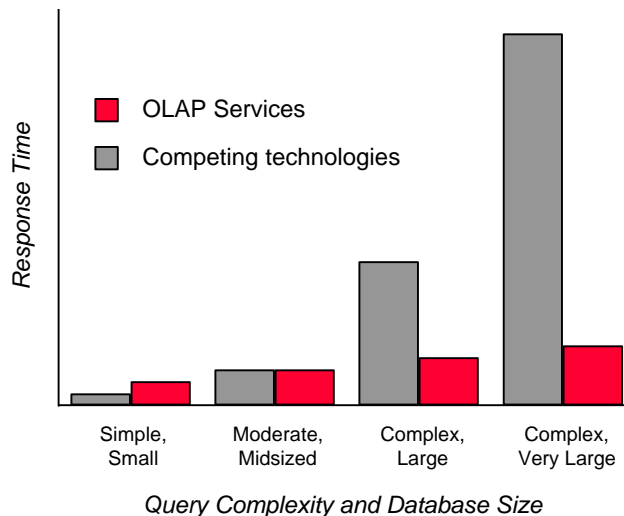
Achieving Both Scalability and Performance

Key to Oracle9i OLAP success is its ability to provide rapid response times for complex multidimensional queries with very large data sets and to manage summary data efficiently.

Query Performance

Oracle9i eliminates the tradeoff between analytical complexity and support for large databases. In general, users can expect the following runtime query performance characteristics:

- When queries are very simple and data sets are small (for example, less than 100 GB) both multidimensional databases and Oracle9i OLAP will deliver sub second response times. Multidimensional databases will, however, often have some advantage in this scenario but this doesn't matter because the end user will be satisfied with performance of both.
- When queries begin to show some complexity and data sets are mid sized (for example, 100- 250 GB), performance of multidimensional databases and Oracle9i OLAP will tend to be similar.
- As queries become more complex and the data set grows larger (for example, 250 GB to 1 TB) Oracle9i OLAP query performance will be superior to that of multidimensional databases. In this case the difference is very noticeable to the end user.
- When data sets are very large (larger than 1 TB) and queries are complex, performance of competing products will degrade while Oracle9i OLAP continues to perform well.



Multidimensional databases tend to have extremely simple execution plans. As a result, they have a modest performance advantage when queries are relatively simple and data sets are small. Oracle9i OLAP and the Oracle database spend more time constructing an optimum execution plan, which pays off as the query becomes more complex and the data set grows larger.

Oracle9i provides the foundation for Oracle9i OLAP and the ability to achieve excellent query response times. Some important parts of this foundation - star join

transformations, bitmap indexes, parallel query, function indexes and some SQL OLAP functions - first became available in Oracle8. Oracle9i introduces many additional performance features that are used by Oracle9i OLAP including bitmap join indexes, grouping sets, WITH clause, additional SQL OLAP functions, automatic memory tuning, and enhanced cursor sharing. A brief overview of these features follows.

Bitmap Join Indexes

Bitmap join indexes pre-joins several tables and stores the result in a single bitmap index. Bitmap join indexes improve query performance by eliminating the need to calculate joins of dimension tables to fact tables at runtime.

Grouping Sets

Grouping Sets allow Oracle to select data from multiple levels of summarization in a single select statement. Previously, a separate Select statement (with a single Group By) was required for data at each separate level of summarization. Multiple select statements are inefficient because each select statement requires a execution plan and each might require expensive operations such as table scans and sorts. In Oracle9i, there only one execution plan needs to be developed. If the table needs to be scanned or sorted, it only needs to be scanned or sorted once.

WITH Clause

The WITH clause provides Oracle with the ability to create temporary results for the duration of the query and use these results as part of a computation. This eliminates the need for Oracle9i OLAP to use temporary tables.

SQL OLAP Functions

SQL OLAP functions such as Moving Average, Cumulative Sum, and Lead/Lag were first introduced in Oracle8. In Oracle9i a new set of functions are provided that are used by Oracle9i OLAP to efficient resolve analytical queries. These new functions include inverse percentile functions, hypothetical rank and distribution functions, and first and last aggregate functions.

Automatic Memory Management

Automatic memory management features provide the correct amounts of memory during memory intensive tasks such as hash joins, sorting, and bitmap operations. Oracle provides enough memory to allow the operation to be completed efficiently. It also prevents over allocating memory, which increases overall performance of the database by allowing other users and operations the memory they need.

Enhanced Cursor Sharing

Enhanced cursor sharing eliminates the need to recompile queries when another similar query has already been run. This saves CPU and time.

Summary Management

Oracle9i uses *materialized views* to manage summary data within the data warehouse. Materialized views are scalable and easy to maintain.

As compared to summary tables, materialized views offer several advantages:

- They are transparent to applications using the warehouse
- They manage staleness of data
- Materialized views can automatically update themselves when source data has changed

Like Oracle tables, they are also highly scalable.

Unlike summary data stored in proprietary multidimensional cubes, data in materialized views are equally accessible to all applications using the Oracle data warehouse.

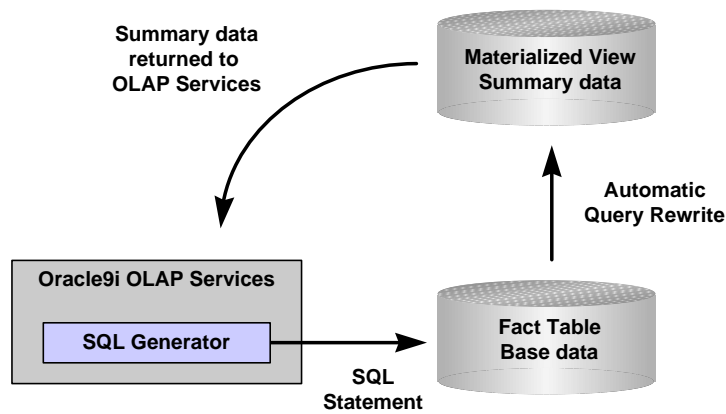
Transparency

Base level data is the data at the lowest levels of a cube. For example, if a Time dimension contains Months, Quarters and Years, Month is the base level for Time. If Geography dimension contains Store, City, and Region, Store is the base level for Geography.

All other levels are summary levels. That is, they represent a summarization of level data. Quarters and Years are summary levels for Time; City and Region are Summary levels for Geography.

In Oracle9i, base level data is stored in a fact table and summary level data can be calculated and stored in materialized views. The Cube in the OLAP catalog maps the base level in a dimension object to the fact table. Oracle9i OLAP does not need to be aware of materialized views since the Oracle database automatically rewrites queries written to the base fact table to access data in materialized views.

When Oracle9i OLAP requires summary level data it issues a Select statement that asks for an aggregation of base level data. Oracle automatically recognizes that the summary level data has already been calculated and stored in a materialized view and satisfies the Oracle9i OLAP query with data from the materialized view.



Materialized views allow the DBA flexibility in maintaining summary data. Since applications do not need to be aware of materialized views the DBA is free to create, change and remove them without effecting the ability of the application to access data.

Managing Staleness

Summary data is said to be 'stale' when lower level data has been updated but the summary data has not yet been recalculated. Materialized views understand when data has changed in the source fact table and can automatically alter the behavior of query rewrite as per policies set by the DBA.

Staleness policies set by the DBA control whether query rewrite is allowed to use the materialized view when data has changed in the source fact table. Consider two cases:

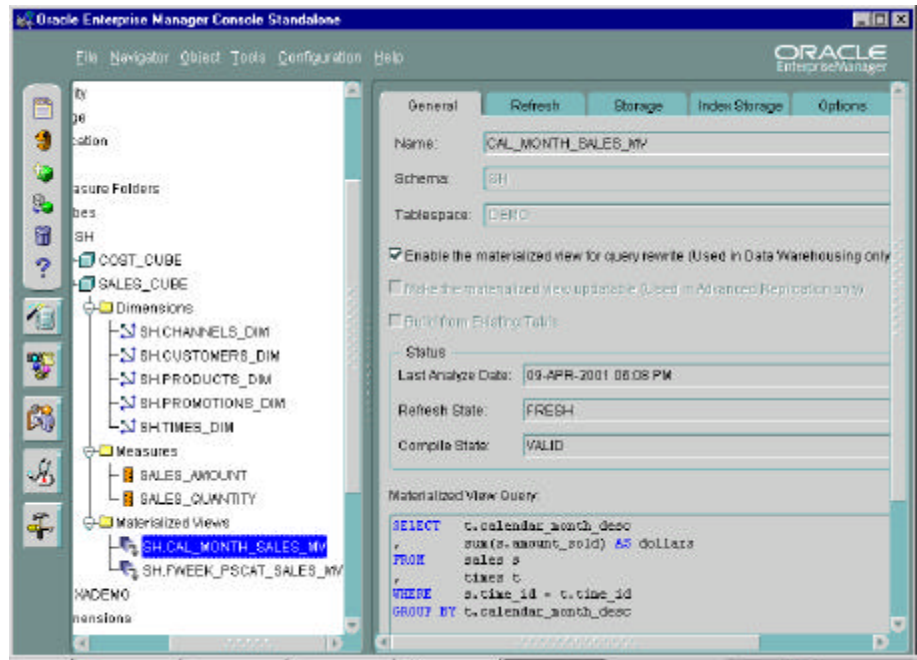
- A warehouse containing sales data is updated to reflect a small correction to the number of units of a product sold to a specific customer. In this case, that small update it not likely to materially affect the accuracy of marketing analysis conducted against that that data. In this case, the DBA would probably choose to allow the materialized view to be used for query rewrite even though it contained some stale data. As a result, the users can continue to enjoy the performance benefits of the materialized view.
- A warehouse used for financial reporting is updated to reflect new revenue and expense data. In this case, the changes to the data might have a significant affect on the financial results reported by the organizations. The DBA would probably choose not to allow the materialized view to be used for query rewrite while the materialized view contained stale data. Instead, the query would be satisfied with data in the fact table. Although the query would run more slowly, the user is guaranteed of seeing the results based on the updated data in the fact table.

DBAs only need to set staleness policies once. Oracle can automatically disable query rewrite to a particular materialized view when data becomes stale and automatically enable query rewrite after the materialized view is refreshed.

Building and Updating Materialized Views

Materialized views can be built using SQL commands and by using OLAP tools in Oracle Enterprise Manager. When using Oracle9i OLAP, Oracle Enterprise Manager's OLAP tool is the preferred method. The OLAP tool will automatically create a materialized view that is perfectly tuned to provide the best query performance with Oracle9i OLAP. Materialized views created by Oracle Enterprise Manager's OLAP tool are also perfectly well suited for use by other applications.

In the following illustration, we see materialized views that are created for the Sales cube in Oracle Enterprise Manager.

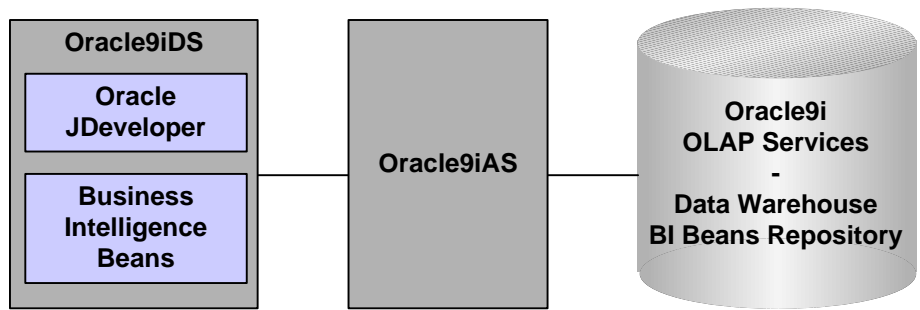


Since materialized views are a special form of an Oracle table, all of Oracle performance and scalability features are applicable to them. For example, materialized views can be partitioned, partitions can be updated in parallel, and grouping sets can be used for aggregations.

Materialized views can be updated by several different methods. They can automatically update themselves when data changes in a source table, they can update themselves on a scheduled basis and they can be updated explicitly as the result of a SQL command. Like when materialized views are first created, partitioning and parallelism are used during the update process.

DEVELOPING AND DEPLOYING APPLICATIONS

Oracle9i, Oracle9iAS, and Oracle9iDS provide a complete platform for the development and deployment of business intelligence applications.



Oracle9i serves as the database platform providing data storage, metadata, summary management and Oracle9i OLAP (including the Java OLAP API).

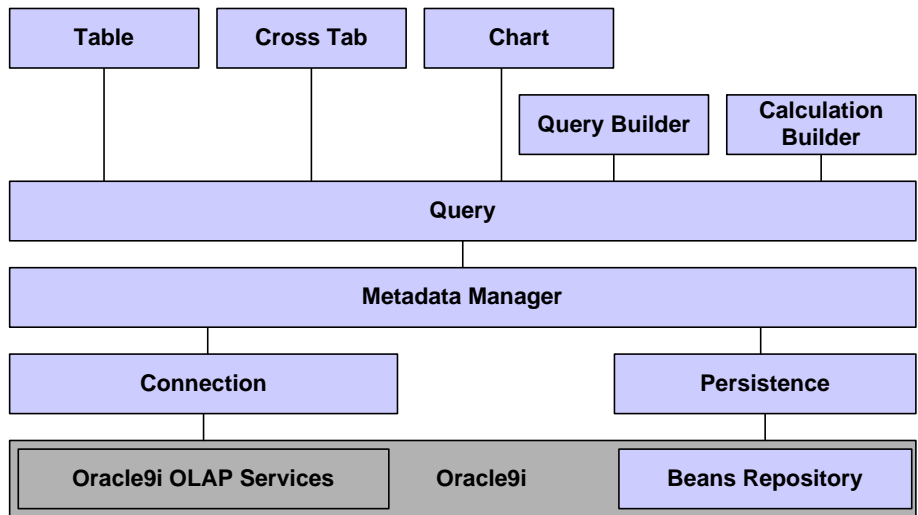
Oracle9i Development Suite (Oracle9iDS) provides application development tools including Oracle JDeveloper and the Business Intelligence Beans.

Oracle9i Application Server (Oracle9iAS) provides the runtime infrastructure used to deploy applications built using the Oracle Business Intelligence Beans. Oracle9iAS also includes other tools useful in the deployment of business intelligence applications such as Oracle Discoverer, Oracle Reports, and Oracle Portal.

Together Oracle9i, Oracle9iDS, and Oracle9iAS provide a complete and integrated solution for the development and deployment of business intelligence applications. Since it is an integrated product stack you do not need to play the role of system integrator.

Oracle Business Intelligence Beans

The Oracle Business Intelligence Beans are Java Beans that provide analysis-aware application building blocks designed for Oracle9i OLAP. The following illustration shows the relationships among the beans.



There are three types of Business Intelligence Beans: presentation, data, and persistence services

Presentation Beans

Presentation beans include table, cross tab and chart. Table provides a row oriented view of data. Cross tab offers a multidimensional view of data and provides services such as drilling, pivoting, and the selection of page dimension members. The chart bean offers graph views of data. A very simple sample application that uses the cross tab bean as a data display is pictured below.

	2000		1999	
	Sales	Quota	Sales	Quota
Total Regions of the World	46,412,112.00	42,667,472.00	109,802,640.00	101,209,600.00
Total Areas in the Americas	13,919,578.00	12,773,550.00	32,654,836.00	30,093,278.00
Total Australia	3,967,199.50	3,655,550.50	9,230,257.00	8,484,204.00
Total Europe	20,029,270.00	18,415,906.00	48,266,256.00	44,415,004.00
Total Asia	8,498,066.00	7,822,465.50	19,661,294.00	18,217,116.00

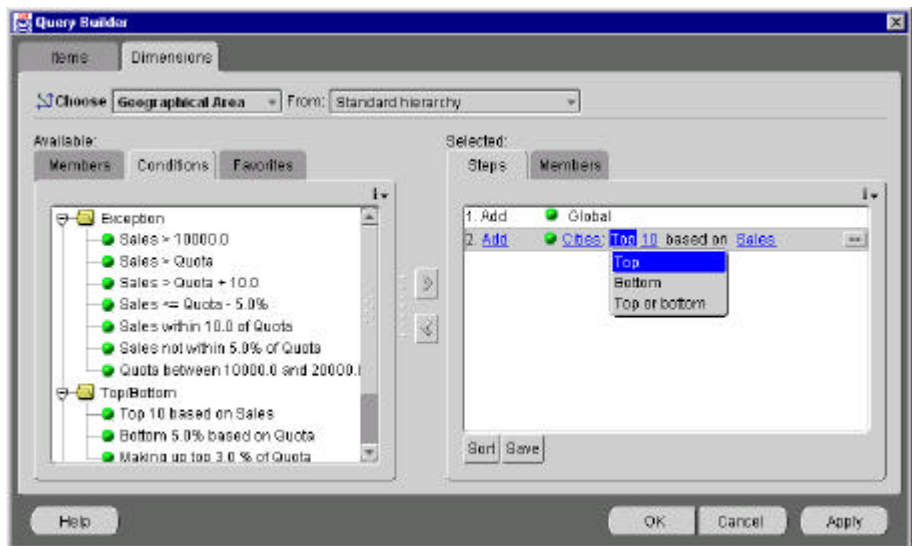
Presentation beans offer a wide variety of formatting and data manipulation tools. For example, data driven formatting allows users to highlight data by automatically using foreground and ground colors based on data values. Users who are accustomed presentation quality formatting in desktop tools such as spreadsheets will be pleased with applications that use the Business Intelligence Beans.

Since the Business Intelligence Beans use a layered architecture, the presentation beans are not 'hard wired' to Oracle9i OLAP. Instead, they use Data beans to access data through Oracle9i OLAP. Oracle takes advantage of this architecture by using the presentation beans in a variety of products such as Oracle Discoverer, Oracle Reports, and Oracle Portal. This is particularly important since it allows organizations to build intelligence web portals that access data from a variety of sources that provide a common user interface regardless of the data source.

Data Beans

Data beans manage the application's interaction with Oracle9i OLAP and eliminate the need for the application developer to write directly to the Java OLAP API. Instead, the data beans generate calls to the Java OLAP API. Since the Business Intelligence Beans provides a higher level API, the task of developing applications is greatly simplified.

The Query Builder bean provides user interface that allows users to select data in a multidimensional context using a variety of powerful, analytically aware selection tools. The Calculation Builder allows users to define virtual facts ("custom measures").



The Query Builder, pictured above, allows end users to easily select data in a multidimensional context. Users can select from lists of dimension members or, as pictured above, define selection rules. Selection rules are built from examples (see the left side of the Query Builder) that can be customized (see the right side of Query Builder).

In the example above, the user has selected a ranking selection example (“Add Top 10 Cities based on Sales”). The user can modify the selection by selecting values using hyper links. For example, the user might change Cities to Regions, Top to Bottom, 10 to 20, and Sales to Units to arrive at ‘Add the Bottom 20 Regions based on Unit Sales’.

In addition to ranking and exception tools, other selection tools include level selections, selection by attribute, hierarchical selections, and time based selections.

One key concept that contributes to ease of use is editable queries. Users build selections using discrete steps that can later be edited. For example, a user might build a selection such as:

1. Select all products at the brand level.
2. Keep all products that had sales of 0 in 1999
3. Keep all products that had sales greater than 0 in 2000
4. Keep the top 10 products based on unit sales.

The above selection might be used to select the top 10 brands of those brands that where new in the year 2000.

If the user wanted to instead select the top ten new items, they could edit step one and change it from “Select all products at the brand level” to “Select all products at the item level”.

The Query bean provides a view of the multidimensional model to the presentation beans, query builder bean, and calculation bean and generates Java OLAP API queries. This dramatically improves the application developers productivity by eliminating the need to write to the lower level Java OLAP API.

The Connection bean manages connections with Oracle9i OLAP. For example, the connection bean provides services to make the initial connection and authenticate the user.

Persistence Services

Persistence Services manages the Business Intelligence Beans repository. The beans repository is used to store all developer and user defined analytical objects such as reports, graphs, saved selections and virtual measures. Applications can use the Persistence Services to save, retrieve, and otherwise manage all analytical objects in the BI Beans repository.

Since the BI Beans repository is in the Oracle database, it is scalable and secure. It is designed to support large distributed user communities who share analytical objects in collaborative environments.

Java OLAP API or Business Intelligence Beans?

Applications developers have the choice of writing applications using Oracle9i's Java OLAP API or the Business Intelligence Beans. Which you use will depend on the type of application you wish to develop.

The Java OLAP API is the lowest level interface to Oracle9i OLAP. It is very powerful and extremely flexible. Using the Java OLAP API you can create highly customized Java applications, Java applets (applets run in the context of browsers), Java beans, and servlets.

In general, developers of third party commercial software products will access Oracle9i OLAP by writing directly to the Java OLAP API. Examples:

- The developer of an ad-hoc reporting application could use Oracle9i OLAP to extend the analytical capabilities of their product. Since this developer already has an application with a user interface suitable for this application, they would write directly to the Java OLAP API to use Oracle9i OLAP as an analytically aware data source.
- A developer of a multidimensional analysis application could use Oracle9i OLAP to replace a proprietary multidimensional engine.
- The developer of Java beans which provide specialized visualization methods such as geography mapping would write directly to the Java OLAP API to access Oracle9i OLAP.

Most corporate IT organizations who build their own applications will use the Business Intelligence Beans. The Business Intelligence Beans provide the application components that are needed by most business intelligence applications

- tables, cross tabs, charts, data selection, and data calculation tools. The Business Intelligence beans offer a high level API that insulates the application developer from the lower level Java OLAP API. As a result, applications can be built more easily in less amount of time and at a reduced cost.

Development Environment

Applications can be built using the Business Intelligence Beans using any Java development environment that supports Java beans. Oracle recommends using Oracle JDeveloper because the Business Intelligence Beans are highly integrated with JDeveloper and offers many advantages. These include:

- Wizards in Oracle JDeveloper that can automatically generate Java code that is used to build applications the use the Business Intelligence Beans. For example, Oracle JDeveloper's Business Intelligence Beans wizards can automatically generate Java application code for an application that displays a report, a chart, has access to the Query Builder and Calculation Builder, and includes a menu with commands that allow users to save and open reports and charts.
- Oracle JDeveloper provides a live connection to Oracle9i OLAP during the application design session. This allows the application developer to see data in the application at design time. This is particularly important with business intelligence applications because the data content often affects how the analytical objects are designed. For example, being able to see the data at design time allows the developer to make decisions about data selections and formatting in a report. (Could you imagine trying to format a spreadsheet if you couldn't see the data?)
- Oracle JDeveloper understands the Business Intelligence Beans repository. This makes it very easy to share analytical objects between many different applications. For example, several different applications could all share the same report. When the report needs to be altered, JDeveloper can be used to make the changes. All the applications that use that report then automatically see the updated report. You can do this using other development environments, but JDeveloper makes the task almost trivial.

Development Process

There are two basic tasks in the process of developing an application using the Business Intelligence Beans. The first is to define analytical objects such as reports, graphs, and data selections. The second is to develop the application code that will display these objects to the end user.

The procedure for creating analytical objects involves three steps:

1. Create a new report or graph.

2. Graphically choose the data and layout that you want to appear in the report or graph.
3. Save the object to the Business Intelligence Beans repository.

These steps are easily accomplished using in Oracle JDeveloper or an application that you create. Oracle JDeveloper is very convenient, particularly if the creator of the analytical objects is also the application developer. If an end user (for example, a marketing analyst) will be the creator of the analytical objects they can do so in an application created using the Business Intelligence Beans.

After the analytical objects are created (or at least placeholders for them) they can be used by a Java applications, a Java applets, a servlets or JSP.

Java applications run on the users desktop (like a Windows application). Java applets run within a web browser. Both Java applications and applets offer full featured GUIs that are run on the client machine.

Servlets and JSP are Java applications that run on a server. Servlets generate HTML code that is displayed in the user's browser. As a result, servlets do not require that any Java code be downloaded from the client machine and are therefore very fast. Servlets, however, do not provide as rich of a GUI to the end user.

Since the Business Intelligence Beans support Java applications, Java applets and servlets you are free to choose what type of application is best for your users. In general, Java applications work well for users who spend a considerable amount time using an application (for example, a market or financial analyst) and servlets are best for occasional users or users who access the system using relatively slow connections such as a modem. Java applets work well when you need to embed a rich GUI in an HTML page.

Since analytical objects such as reports and graphs are saved in the Business Intelligence Beans repository they are not embedded into the applications source code. Instead, the applications reference an object in the Business Intelligence Beans repository. As a result, it is possible for many applications to reference the same report or graph. This offers several benefits:

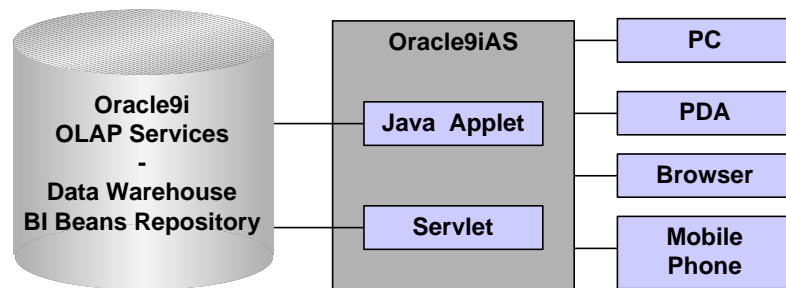
- The application developer and the designer of the analytical objects can be different individuals. The application developer doesn't necessarily need to understand the content of the analysis. Developers of the analytical objects do not need to understand how to build applications. This is very important since the designers of analytical object are likely to be more like a marketing or financial analyst than a Java developer.
- Since analytical objects can be reused, the development process is more efficient.
- Since analytic objects are maintained in central repository, any updates to an objects is automatically reflected in every application that uses that object.

For example, a financial analyst might update a report that is used in several different applications. This reduces the cost of maintaining applications.

Deploying Applications Built with Business Intelligence Beans

Because applications built with the Business Intelligence Beans are built using Java they can be deployed anywhere on the Internet. Java applications can be deployed on any device that can run Java. Servlets can service any device that supports a browser. The application logic is written by the developer on the middle-tier and may be reused by any client application: Java applications, servlets, JSPs, etc. enabling support for a variety of devices such as PCs and browsers, PDAs and even telephones.

When using the Business Intelligence Beans, deployment services are provided by Oracle9i Application Server (Oracle9iAS).



An increasingly common deployment method is to use business intelligence portals. Business intelligence portals can be built using Oracle Portal, which is included with Oracle9iAS. Since the Business Intelligence presentation beans are used throughout Oracle's business intelligence product line, a business intelligence portal built using Oracle Portal, the Business Intelligence Beans, Oracle Reports, and Oracle Discoverer provides the user with a consistent user interface experience within the portal.

CONCLUSION

The days of large replicated data stores, client server business intelligence applications and organizations acting as system integrators are past.

The Oracle9i product line provides a scalable and manageable solution for data warehouse based business intelligence applications. Oracle9i OLAP provides the means to effectively analyze data in the data warehouse. As a result, it is no longer required to replicate large amounts of data into specialized databases and incur the monetary costs of maintain such as infrastructure and the opportunity costs of data latency.

Oracle Business Intelligence Beans and Oracle JDeveloper provide a highly productive development environment for building attractive and easy to use business intelligence applications that contain rich analytic content. Oracle9i

Application Server provides integrated deployment services for thin client applications on the Internet.

This product line provides organizations with the opportunity to build sophisticated business intelligence solutions around centrally managed data warehouses and deploy them to large, distributed user communities. Total cost of ownership is minimized by avoiding costs associated with assembling multi-vendor solutions on site and maintaining them on an ongoing basis.



Oracle9i OLAP: A Scalable Web-Base Business Intelligence Platform

April 2001

Author: Bud Endress

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 © 2001 Oracle Corporation

All rights reserved.