

Oracle9i Business Intelligence Beans

*An Oracle White Paper
September 2002*

Oracle9i Business Intelligence Beans

EXECUTIVE OVERVIEW	3
INTRODUCTION.....	4
ORACLE9i: PLATFORM FOR DEVELOPING ANALYTICAL APPLICATIONS.....	6
Oracle9i OLAP	6
Analysis Functions.....	6
Materialized Views	7
OLAP Catalog.....	8
Oracle OLAP API.....	8
Oracle9i Business Intelligence Beans	9
Core Components.....	10
Presentation Beans	10
OLAP Beans.....	11
Catalog Services.....	13
Application Development Using Oracle9i Business Intelligence Beans and Oracle JDeveloper.....	14
CONCLUSION.....	16

Oracle9i Business Intelligence Beans

EXECUTIVE OVERVIEW

Businesses in the modern economy need to transform themselves into e-businesses very quickly and smoothly. With this transformation, employees, from senior executives to junior analysts, will gain insight into real-time business performance to make better decisions for the future. They need the appropriate tools to obtain this information.

Oracle9i, the new release of Oracle's industry-leading database, provides a complete and integrated infrastructure for such business to e-business transformations. Oracle9i OLAP inside the database and Oracle9i Business Intelligence Beans (hereinafter referred to as "BI Beans") with Oracle9i JDeveloper (hereinafter referred to as "JDeveloper") are designed specifically to provide the platform to develop powerful business intelligence applications. IT developers, technical consultants and third party software providers can take advantage of the many BI Beans functionalities to build highly analytical custom applications for different target audiences. Built upon open technologies such as Java and XML, these applications are platform independent and J2EE-compliant. This paper will describe how BI Beans function within Oracle's integrated technology stack and how using BI Beans has a profound effect on the speed with which feature-rich business intelligence applications can be developed.

INTRODUCTION

The Business Intelligence (BI) market has been growing at a healthy pace for some time. We are seeing this growth across the different segments of the market: BI Applications, Multi-dimensional (OLAP) Analysis and End-User Query and Reporting.

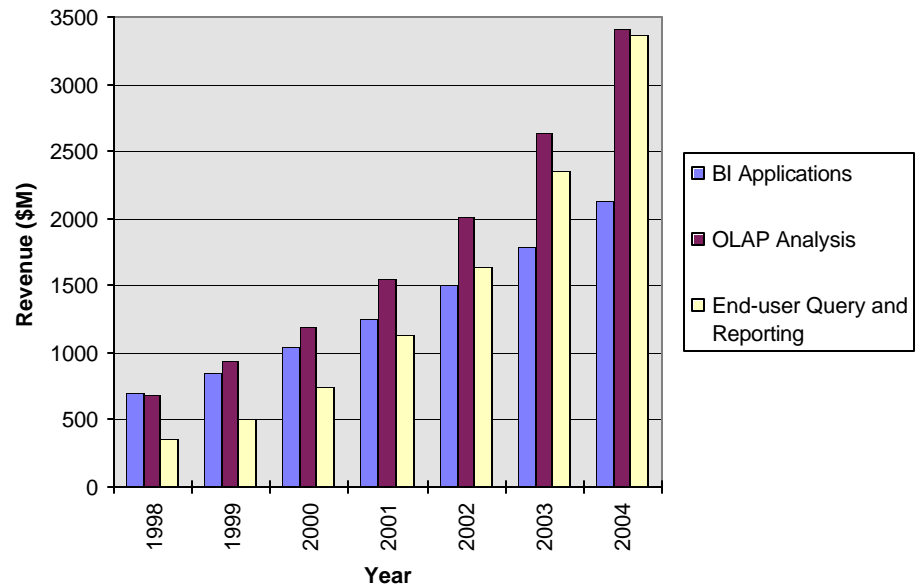


Figure 1: Worldwide Business Intelligence Market Growth (Source: IDC)

The reasons for this consistent growth are numerous. Corporations have historically focused their investment on OLTP applications, which they now are realizing must be extended through analytics. Examples of these applications include:

- Accounting – forecasting, budgeting, cost and profitability analysis, consolidation
- HR – HR/skills consolidation, labor scheduling, optimization
- Distribution – scheduling and optimization
- Sales Force Automation – cross selling, territory analysis
- Marketing – churn analysis, market-based analysis
- Retail – site location, demographic analysis
- Manufacturing – demand planning, forecasting
- Health Care – outcomes analysis
- Financial Services – risk management and assessment

These applications can be organized into two general categories: control-based and opportunity-based systems. Financial services and accounting applications, like Oracle Financial Analyzer, are systems that review and optimize business practices. These control-based systems generate a return on investment of 886% during an initial three-year period (source: IDC). Customer Relationship Management and Sales Force Automation analyses are examples of opportunity-based systems. These types of applications yield an average return of 452% over an initial three-year period (source: IDC).

In addition to the significant return on investment, another major reason for this BI market growth is the continued expansion of the target user community. No longer are analytical applications targeted to a small analyst community within the organization. The growth of the use of these tools is expanding to both lower levels and senior levels in an organization. Gartner Group expressed this sentiment well when they wrote:

"Instead of a small number of analysts spending 100 percent of their time analyzing data, all managers and professionals will spend 10 percent of their time analyzing the data themselves."

"Instead of a small number of analysts spending 100 percent of their time analyzing data, all managers and professionals will spend 10 percent of their time analyzing the data themselves."

Gartner Group

Now more than ever, companies can quickly respond to change. Virtual storefronts enable on-line retailers to quickly and effectively target promotions to a market of one – an individual who is visiting the retailer at a particular time. Correct use of business intelligence in this area can truly have a profound effect.

Customers are increasingly interested in purchasing an application rather than building one from scratch. Oracle is committed to providing the complete solution. Oracle's ERP and CRM applications offer the Oracle Business Intelligence System (BIS) – pre-packaged decision support for Oracle applications. BIS is a family of separate yet integrated intelligence modules for Financials, Purchasing, Operations, Human Resources, and Process Manufacturing. In addition, Oracle Financial Analyzer and Oracle Demand Planning target the corporate planners in an organization and provide all the necessary tools to effectively develop budgets and predict demand.

On the other hand, customers often want to extend the application, to provide capabilities that are unique to given situation. Or perhaps the end-user requirements are so unique that a completely custom application required. In either case, customers need a technology platform that is open – one that is based on scalable, secure, industry standard technology.

Oracle9i enables a development-ready application environment that leverages an industry-standard Java OLAP Application Program Interface (API) and BI Beans.

ORACLE9i: PLATFORM FOR DEVELOPING ANALYTICAL APPLICATIONS

Oracle9i is uniquely positioned and specifically designed to provide the core technology platform for developing analytical applications. Oracle9i provides a reliable and scalable analytic engine through the integrated Oracle9i OLAP. Oracle9i OLAP supports extensive analysis functions, materialized views, and an OLAP Catalog, which are accessible through an established Oracle9i OLAP API. BI Beans, by integrating with JDeveloper, leverages the Oracle9i OLAP features to provide a platform for the rapid development and deployment of e-business intelligence solutions. For the remainder of this paper, we will review in detail the different components that make the BI Beans/JDeveloper combination the development platform of choice for BI applications.

Oracle is uniquely positioned and completely committed to providing the core technology platform to develop analytical applications.

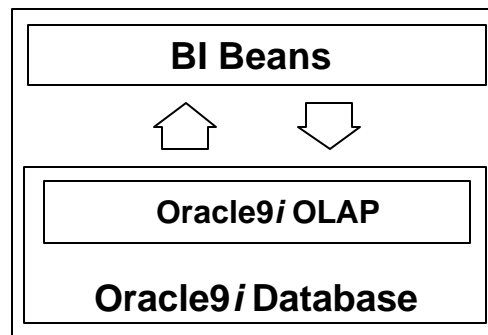


Figure 2: Interaction of BI Beans with Oracle9i through Oracle9i OLAP

Oracle9i OLAP

The analytical capabilities of Oracle9i and its support for highly performant queries continue to expand with the introduction of Oracle9i OLAP. These capabilities build on enhancements from previous Oracle8i releases and encompass extensions to the SQL language, support for materialized views, and the introduction of the Oracle OLAP API. BI Beans takes advantage of these enhancements through the Oracle OLAP API.

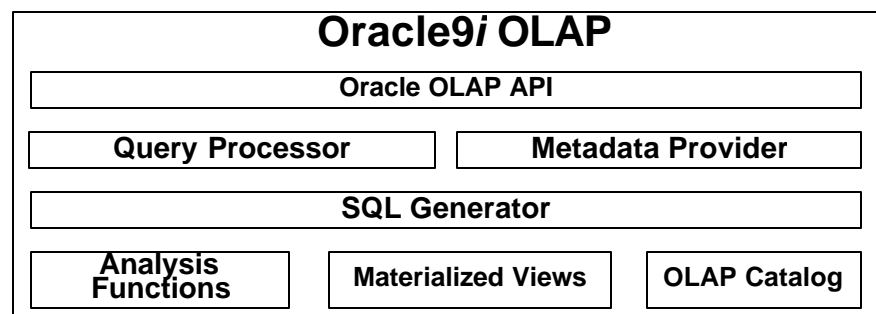


Figure 3: Oracle9i OLAP overview

Analysis Functions

In Oracle9i, enhancements to SQL to support analysis functions are significant. The goal of these enhancements is to make it easier to write highly performant

business intelligence queries against the database. Brief descriptions of the families of queries that have been added are listed below:

- **Ranking family** – This family supports business questions such as “Show the top 10 salespersons per region” or “Show, for each region, salespersons that make up 25% of the sales.”
- **Window Aggregate family** – This family addresses questions like “What is the 13-week moving average of a stock price?” or “What was the cumulative sum of sales per each region?”
- **Reporting Aggregate family** – One of the most common types of calculations is the comparison of values at different levels of aggregation. For instance, a user may want to know regional sales levels as a percentage of national sales. The reporting aggregate family makes these calculations simple: it allows one row to contain values calculated at different aggregation levels.
- **Lagging/Leading family** – Studying change and variation is at the heart of analysis. This necessarily involves comparing the values of different rows in a table. While such calculations have been possible in SQL, usually through self-joins, they have not been efficient or easy to formulate. This family enables queries to compare different rows of a table simply by specifying an offset from the current row.
- **Linear Regression family** – Oracle9i provides functions for linear regression calculations, including slope, intercept, correlation coefficient, and many other key values.
- **Inverse Percentile family** – Added in Oracle9i, these functions allow queries to find data that 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 Distribution family** – These functions, new in Oracle9i, allow queries to determine what rank or percentile value a hypothetical data value would have if it were added to an existing data set
- **FIRST/LAST Aggregates family** – An Oracle9i enhancement, this family enables queries to return the first or last value of a sorted aggregate group.

Materialized Views

One of the techniques that is employed in data warehouses to improve performance is the creation of summaries, or aggregates. Materialized views are special kind of aggregate views that improve query execution times by pre-calculating expensive joins and aggregation operations prior to execution, then storing the results in a table in the database. For example, a table could be created that contains the sum of sales by region and by product.

Prior to the introduction of materialized views, organizations spent a significant amount of time manually creating summaries, identifying which ones to create, indexing the summaries, updating them, and advising their users on which ones to use. The introduction of summary management in Oracle9i eases the workload of the DBA dramatically and means that the application developer no longer needs to know which summaries have been defined. The DBA creates one or more materialized views, which are the equivalent of a summary. The end-user queries the tables and views in the database, and the query rewrite mechanism in Oracle9i automatically rewrites the SQL query to use the summary tables. This results in a significant improvement in response time for returning results from the query and eliminates the need for the end-user or database application to be aware of the summaries that exist within the data warehouse.

OLAP Catalog

One of the most important advances in the BI industry is the establishment of a metadata standard. Metadata is used to represent a wide range of information, including information about the sources that are used to create the data warehouse, data transformations, security and business rules, etc. Metadata provides meaning to the data in the data-warehousing environment. In order for tools to interoperate, they must understand each other's metadata, and they must cooperate and use the metadata that each tool generates. The Common Warehouse Metadata (CWM) is a specification that describes metadata interchange among data warehousing, business intelligence, knowledge management, and portal technologies. The objective of CWM is to provide an open standard metamodel and application program interfaces (APIs) for data warehousing and decision support tools.

OLAP Catalog is the collection of CWM-compliant runtime-metadata that is used for describing the data for analytics. This infrastructure enables end user customers to dramatically lower the initial and ongoing costs of building, administering and using data warehouses for decision support by providing an open object model and an open, network-oriented API. Moreover, the OLAP Catalog is maintainable through Oracle Enterprise Manager, providing a single point of management for the entire database.

Oracle OLAP API

Oracle9i OLAP introduces a powerful new Java™-based, objected-oriented query and calculation API. The Oracle OLAP API is a pure Java API for the J2EE environment, which allows Java programs such as BI Beans to tap into the powerful analytic engine of the Oracle9i database. It provides a number of related services:

- The OLAP Object Model – Oracle OLAP API presents complex warehouse schema in business terms by reading the definitions of the warehouse out of the OLAP Catalog. The Oracle OLAP API defines an OLAP-specific logical

model, which it exposes as a set of metadata objects. These objects include items such as “measures,” “dimensions,” “hierarchies,” which directly reflect the types of information that are prevalent in a business intelligence application. The client application can use these objects to understand what data is available. For example, what is the list of key indicators that my end users can analyze? What are the hierarchies that an end user can use to navigate through the geography dimension?

- **Metadata Support** – Due to the complexity of the warehouse design and the questions that end users ask, many interrelated lookup tables and fact tables may be used to satisfy even the simplest OLAP query. The optimized SQL that is required to satisfy the user’s simple question may be thousands of lines long. The Oracle OLAP API enables a client application to specify complex multidimensional or relational style queries and calculations using metadata. Complex queries are simplified significantly by directly manipulating the metadata.
- **OLAP Navigation** – OLAP methods – such as drill, page, and pivot – are provided as methods on query objects. Again, this is an example of the API reflecting the types of activities that business intelligence users perform.
- **Query Cache:** The Oracle OLAP API returns the results of query executions to the client application. These results are exposed to the client as “virtual cursors,” and the API manages the underlying data cache. This means that large query result sets are effectively managed by Oracle9i OLAP – providing optimal performance to client applications.

In summary, Oracle9i OLAP enables analytic applications in the database, while the Oracle OLAP API brings OLAP applications to the Web through BI Beans.

Oracle9i Business Intelligence Beans

Oracle9i Business Intelligence Beans leverages the Oracle9i OLAP to provide Java components for the development of analytical applications.

The key objective of BI Beans is to provide the reusable application components and services that will enable the rapid development of business intelligence applications. This development process is enhanced through integration with JDeveloper. Together, JDeveloper and BI Beans provide a highly productive development platform. IT developers from large corporations and third party application providers alike can build completely custom applications very quickly based on the integrated platform. They can also embed analytics into their existing applications because of the open technology and well documented API’s and samples. Because JDeveloper is itself a Java application, a developer can build BI Beans applications on multiple platforms including the many varieties of Windows and UNIX. BI Beans applications are also J2EE-compliant which means they can run on any J2EE-compliant server such as Oracle9i Application Server. The

BI Beans provide key reusable application components and services to enable the rapid development of business intelligence applications

remaining sections of this paper outline the functions of each core component of BI Beans and the benefits of its integration with JDeveloper.

Core Components

The components that comprise BI Beans fall into three general categories: 1) presentation beans, 2) OLAP beans, and 3) catalog services.

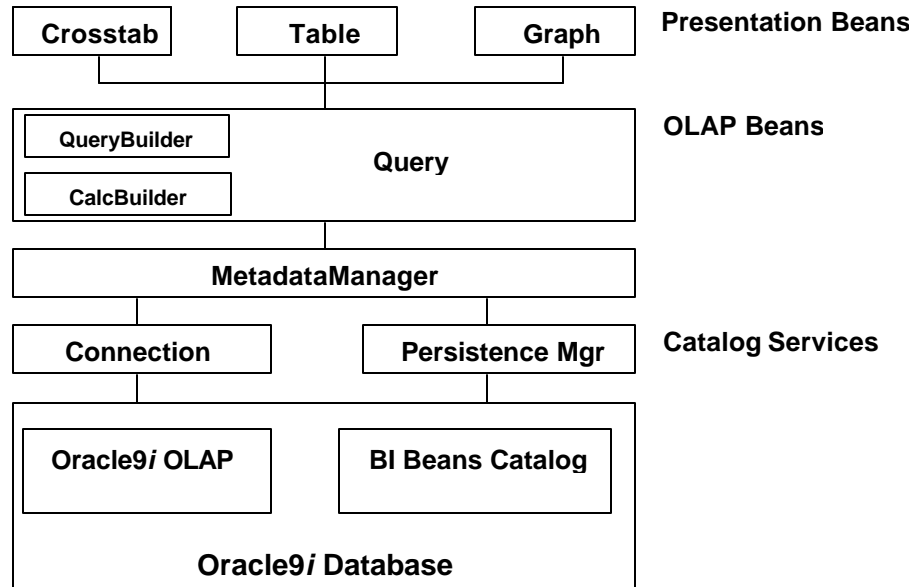


Figure 4: BI Beans Overview

Presentation Beans

The Presentation beans provide the display of complex query results in user-friendly, attractive graphs, tables, and crosstabs. The beans achieve this function by providing data-aware rendering components that are highly customizable both at design time and at runtime through a comprehensive API. The default user interfaces for these beans have undergone extensive usability testing formally and through their use in production-level Oracle products such as Oracle9iAS Discoverer and Oracle9i Reports.

Graph

Using the graph bean, you can create “boardroom quality” presentations. The graph bean supports 70+ graph types, with special customizers for formatting various graph components (layout, legend, series, axes). With the intuitive custom formatting tools, a novice end user can easily create compelling presentations for senior management. Another benefit is the support for printing, with zoom capabilities that allow users to specify the range to print (for example, all page items or any subset of page items). Graphs are also interactive, allowing users to explore the data by drilling or pivoting.

Table and Crosstab

Like graphs, tables and crosstabs provide formatting and printing support. They also allow users to navigate through the data by drilling, paging, pivoting, and so on. In addition, users can define cell-level formatting that is dependent on the data and/or dimension values for a cell. Such rich formatting alerts users to important aspects of their data and makes it easy to identify “problem” areas. Developers do not have to worry about the sizing of tables and crosstabs because they are auto-sized based on the frame or HTML window. This means that users will always have the best view of the data.

Java-Client versus Thin-Client

Depending on the nature and requirements of the application, presentations can be Java-based or HTML-based. For example, high-end analytical users, who spend a large percentage of their workweek analyzing past business performance or developing forecasts of future performance, need a highly interactive environment. The Java versions of the Presentation beans enable a rich, interactive experience. For example, users can rotate the dimensions in a presentation by dragging and dropping or format a presentation directly from the extensive toolbar.

Conversely, for users with slower connections, the thin client version of the Presentation beans will be more productive. These thin Presentation beans display presentations in a web browser. By creating plain HTML, these beans have the inherent benefit of not requiring extensive Java code to be downloaded to the client, which saves valuable bandwidth. They can be integrated easily with existing HTML applications. Also, graphs are exported to an image file on the middle tier (such as GIF, PNG, JPEG) and then inserted to an HTML page just like any other image. Imagemap generation enables data tips and drill down (or custom) action on different presentation components, and a simplified toolbar allows users to modify presentation properties at runtime. Pooled graph instances are maintained and shared across multiple users to increase scalability.

OLAP Beans

While the Presentation beans are responsible for rendering the data in different formats, the business logic is provided by the OLAP beans. This separation of the business logic from the presentation is extremely important, because it enables multiple clients (such as Java, HTML and WAP) to access the same application code.

The following components comprise the OLAP beans.

Connection Bean

The Connection bean provides the connection to Oracle9i. An application may access multiple databases; there is no specific limit to the number of servers that

are accessible to an application. This capability enables, for example, one report to be derived from a Sales History database while another is derived from the General Ledger database.

Query

Behind any presentation is a query. The query specifies exactly what the user is viewing in the presentation: the measures (such as Sales and Costs), the dimensions (such as Product, Geography, and Time), the selections for each dimension (such as the last 6 months), and the layout of the dimensions (such as Product in the rows, Time in the Columns, and paging on Geography).

Much of the business logic for a business intelligence application is contained in the query. The query provides the OLAP navigation capabilities on the presentation objects. A user might think that drilling down on the Northeast Region in a crosstab is an operation that takes place on the crosstab bean. In reality, the operation is affecting the query that the underlying crosstab is displaying. As mentioned earlier, the crosstab is simply a rendering engine that displays query results. User interactions such as paging and pivoting are methods on the query.

The query uses the Oracle OLAP API to retrieve and manipulate data to provide the advanced analytic capabilities that are offered in Oracle9i. The Oracle OLAP API achieves this by generating highly tuned SQL to resolve the request from the query.

QueryBuilder

The QueryBuilder provides a simple user interface to define sophisticated queries. The user interface is patented and extensively tested for ease of use. The QueryBuilder is a very powerful tool that enables users to specify the following query properties without needing to know the underlying query language:

- Measures and dimensions – Chosen using a list tool which displays a hierarchical list.
- Dimension selections – Specified by defining exceptions, rankings, top/bottom selections, hierarchical selections (children and ancestors), text matching, and so on. For example, “Top 5 Products sold in each City”.
- The layout of the dimensions within the presentation.

End-users across an organization have different skill sets and analytical requirements. The QueryBuilder is designed to be highly customizable and componentized to meet these various needs. Developers can turn off capabilities to simplify the user interface or enable it to be used in different contexts. For example, the application might simply provide the list tool or only show the user's favorite selections. Developers can also use components of the QueryBuilder

(such as condition templates) in other parts of an application or add and remove condition types for different purposes.

The QueryBuilder also makes it easy to reuse selections by allowing users to save and retrieve defined selections as favorites. Users often want to use a popular selection or query (such as “Top 5 Products”) in multiple reports or presentations.

A key strength of the QueryBuilder is that the end-user does not need to understand a query language to define the query. Powerful queries are made simple by presenting the query definition in business terms, which end users can modify to meet their needs. For example, a default query definition can be “Show the Top 5 Products based on Sales”. An end-user can then modify the word “Top” to “Bottom” in the definition to show the bottom 5 products based on sales.

CalcBuilder

The CalcBuilder provides a user interface to create derived calculations. Like the QueryBuilder, end-users define new metrics using templates, which eliminates the need to understand the underlying SQL. For example, an end-user might be interested in seeing Sales Growth compared to last year for a particular product line, even though Sales Growth has not been defined by the database administrator. The CalcBuilder enables the user to define this calculation. The calculation templates are organized by type. In our example, the end-user selects the appropriate Time Series template to define the new measure.

Catalog Services

BI Beans Catalog

The BI Beans Catalog provides a secure, searchable, and extensible object storage mechanism that enables end users to save personal analyses and share analyses with other members of the user community. The catalog stores the definitions of tables, crosstabs, graphs, calculations, and queries in XML format.

These objects are organized into folders, where access privileges are granted at the user and user group level. When saving a report, for instance, an end-user could save the document into a Finance Group folder, which would then be accessible to his or her peers in the finance organization. The catalog also supports pluggable authentication schemes and storage mechanisms. For example, developers may want to authenticate users against a company LDAP server or save objects to a data store other than the two supported by default – the Oracle database and the local file system.

The catalog is searchable based on keyword, name, description, author, or date and is filterable by object type (for example, show graphs only). This allows users to concentrate on the objects of interest in a large pool of defined objects.

The catalog is also extensible. Applications can define their own persistable objects by implementing the persistence interfaces. For example, a developer might create a compound object that contains a graph and a crosstab. This new object may contain layout information, page titles, and so on. By implementing the persistence interface, this new object can behave just like any native object; for example, the object can be saved and searched.

MetadataManager Bean

The MetadataManager bean provides an integrated view of all the metadata that is used in the definition of analytical applications. It merges at runtime the metadata that has been defined in the OLAP Catalog with the metadata that is stored in the BI Beans Catalog.

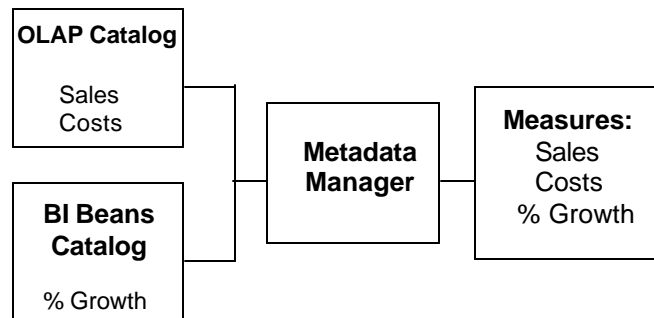


Figure 5: MetadataManager Overview

End users need to access common pieces of information (such as Sales and Costs); this information is stored in the OLAP Catalog within Oracle9i OLAP. Users also need access to personal calculations (such as % Growth) whose definitions are stored in the BI Beans Catalog. These personal calculations might not be meant for general use, in which case they will not have been defined in the common schema.

When creating a new query or calculation, users need an integrated view of both personal and common metadata. The MetadataManager provides that integrated view of information.

Application Development Using Oracle9i Business Intelligence Beans and Oracle JDeveloper

Using BI Beans within JDeveloper provides the most productive development environment for analytical applications. BI Beans are standard JavaBeans™ that can be used in any Java development environment that supports JavaBeans components. However, JDeveloper is the development environment of choice because of its integrated support for BI Beans.

The development of BI applications using Java is a two-stage process. First, the developer defines the application objects, which can be graphs, tables, crosstabs, queries, or calculations. Then, the developer defines the application logic or code,

which manipulates the application objects to create useful BI applications. JDeveloper provides several tools to facilitate the development process during both stages.

Defining the Application Object

JDeveloper provides wizards that guide developers through the process of creating new BI objects. The BI Designer wizard establishes a live connection to the data in a Oracle9i database. Developers can visually create a data presentation, a query, or a custom calculation using wizards and without writing a single line of code. This simplifies and speeds up the definition process significantly.

JDeveloper ensures increased productivity by providing live access to data while the presentation objects are created and modified. The presentation editor enables users to modify presentation properties with a live data source. This means, for example, that the developer does not need to compile and run the application to view the results of a query in a presentation. This is very important because the format of the presentation is typically driven by the data itself. Consider the definition of a graph; you need to interact with the graph to select the correct graph type, color format, and so on.

Another key benefit of using BI Beans with JDeveloper is the integration of the BI Beans Catalog with the development environment. A given application may contain hundreds of BI application objects. To make such an application scalable, the definitions of application objects need to be stored in the catalog – not in application code. This integration also ensures the reusability of objects, because objects are stored and shared through public folders.

Developing the Application Code

Developers who are familiar with JDeveloper concepts will quickly recognize and understand how to develop applications using BI Beans. The development of BI applications uses generic JDeveloper concepts whenever possible, and extends those concepts when required. This is abundantly clear when you view the JDeveloper tools palette, project navigator, and wizards; BI Beans fit seamlessly into the environment.

JDeveloper provides wizards to make the development of applications simpler, including those that guide developers, in a step-by-step process, through the creation of Java applications or applets for Java clients and Java Servlets or JavaServer Pages (JSPs) for thin-clients. Such application wizards greatly reduce the amount of time required to develop an application and the wizards provide the flexibility in deployment. Developers can easily create applets for advanced users and servlets for less-demanding users that are based on the same application objects.

JDeveloper also provides a custom JSP tag library, which contains JSP tags that encapsulate BI logic. For example, instead of writing many lines of custom code to display a graph, a developer can include a simple presentation JSP tag in their application. The custom tag library allows the rapid development of BI applications.

CONCLUSION

By working within the JDeveloper environment with pre-built wizards and JSP tag libraries, a developer can easily and quickly create platform-independent, J2EE-compliant analytical solutions for the desktop and for the Web using BI Beans.

As a modern business revolutionizes itself into an e-business, it requires tools to ensure its success. When going through such a transformation, companies undergo intense pressure not only to understand their customers and drive new business value from these relationships, but also to generate positive return on investments. As a result, more and more such companies are turning to business intelligence systems for customer, supply chain, and operational analysis. Oracle9i provides a business intelligence platform through the integrated Oracle9i OLAP and BI Beans. BI Beans are specifically designed for the rapid development of sophisticated analytical applications. Based on the JavaBeans architecture, BI Beans promote the creation and maintenance of modular, reusable components. By working within the JDeveloper environment with pre-built wizards and JSP tag libraries, a developer can easily and quickly create platform-independent, J2EE-compliant analytical solutions for the desktop and for the Web using BI Beans.



Oracle9i for e-Business: Business Intelligence Beans
September 2002
Author: Kelly Chan

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 © 2001-2002 Oracle Corporation
All rights reserved.