USE ORACLE WAREHOUSE BUILDER TO BUILD YOUR OLAP-READY DATA WAREHOUSE

Patricia Fay, Jean-Pierre Dijcks, Oracle Corporation

INTRODUCTION

Technology attempts to make it easier to provide solutions for difficult problems. But, no matter how ingenious the technology is, the ability to implement it often becomes the deciding factor in whether the solution it offers is adoptable. A solution is most useful when it fits comfortably into the process and information flow of your business. This paper describes an OLAP solution that will comfortably fit into your processing environment.

One long-time problem that is looking for a solution is well known by businesses that have complex analytic, forecast and planning requirements. You already know if your business requires the analytic capabilities of a multidimensional analytic engine. In the past, that meant the purchase and maintenance of a database system which was different from the database system that warehoused the source data. And was probably different from the database system that hosted your Business Intelligence applications. The OLAP option of the Oracle*9i* Database provides the capability to perform all analysis tasks in the same database system that supports your BI applications.

Oracle*9i* also provides tools that support all of the OLAP activities from data preparation through the reporting of analytic findings. Oracle Warehouse Builder (OWB) is the Oracle ETL tool that supports the design, deployment, and maintenance of the data warehouse. You can use OWB to fully prepare and maintain your source data for OLAP processing. OWB provides ease of use through its design environment, and flexibility in its ability to deploy both relational and multidimensional data stores.

In addition, there are reporting and broadcast tools, such as Oracle Discoverer and Oracle Reports which format and distribute results of your analysis.

Now you need to know how to incorporate the OLAP capability as part of your business practices. This paper will describe how to use Oracle Warehouse Builder to prepare your data warehouse for Oracle*9i* OLAP processing.

The topics in this paper are covered in seven sections:

- The OLAP option in the Oracle*9i* Database
- Describing the OLAP Components
- Oracle Warehouse Builder
- Building the Data Warehouse
- A Scenario - Use OWB to build your OLAP Data Warehouse
- The Case Study
- The End Result – The OLAP-Ready Warehouse

AUDIENCE

This paper describes how to use Oracle Warehouse Builder to build a data warehouse that is OLAP ready. The information contained within will be of benefit to both business managers and data base administrators. It is assumed that OLAP processing is a business requirement of their business, and that the benefits are understood. Also assumed is a basic understanding of relational and multidimensional concepts.

SECTION 1: THE OLAP OPTION IN THE ORACLE*9i* DATABASE

The Oracle*9i* Database is an integrated Relational – Multidimensional database. The database contains the full power and advantages of the Relational engine and the OLAP engine. This architecture reduces cost and effort without compromising performance and analytic power.

Topics in this section are:

- The OLAP Option
- The Architecture of the Integrated Database
- Advantages of the Oracle Integrated Database

THE OLAP OPTION

OLAP is an option to the Oracle*9i* Database Enterprise Edition. It provides a simplified way to perform both simple and complex analysis.

Some components of OLAP are:

- A full-featured OLAP engine that supports the calculation, planning and forecasting for multidimensional data.
- Multidimensional data types.
- A SQL interface to the multidimensional data.
- An OLAP API to both the relational and multidimensional data.

THE ARCHITECTURE OF THE INTEGRATED DATABASE

Databases and their management systems can contain complicated concepts. The diagram in *Figure 1* is a simplified representation of some of the key features of the Oracel*9i* Relational – Multidimensional Database and can be used as a frame of reference as you construct a mental model of what this integrated OLAP system might look like.

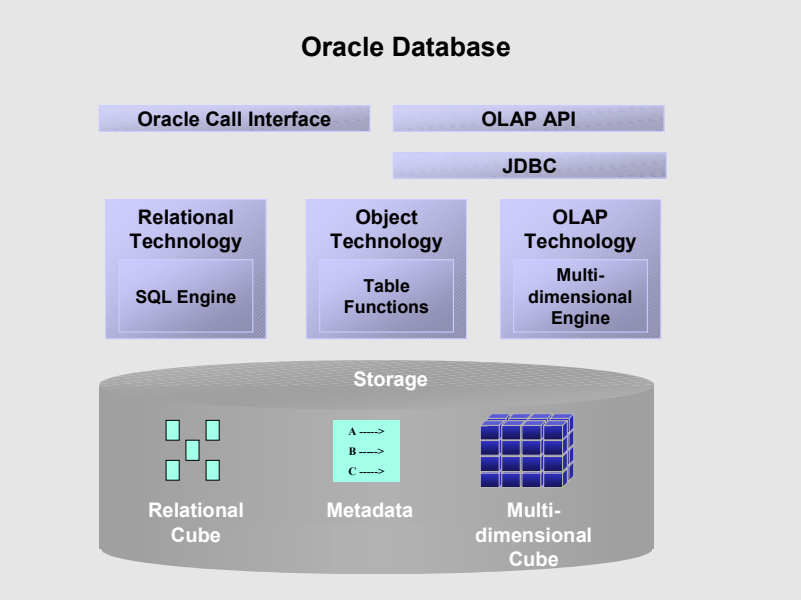


Figure 1: Oracle9i Integrated Relational – Multidimensional architecture

Starting at the bottom of the diagram, note that all data and metadata is stored within one database entity. Of course, the data storage could span multiple database instances if it needed to, but the point here is that different **types** of data are stored in the same entity. That is, relational and multidimensional data and metadata. This is a new concept.

Looking at the middle of the diagram, note that the relational engine, the multidimensional engine, and the object technology are hosted within the same database management system. Again, this innovative architecture is specific to the Oracle*9i* Database.

And at the top of the diagram, note that Oracle provides a Java API that can be used to access and manipulate data in all of the available storage formats.

ADVANTAGES OF THE ORACLE INTEGRATED DATABASE

Both relational database systems and multidimensional database systems offer their own advantages. Generally speaking relational systems offer outstanding data management features, and multidimensional management systems offer outstanding calculation and analytic features. Oracle*9i* is an integrated relational – multidimensional management system and offers the best features of each.

Some of the features and advantages in the Oracle*9i* integrated database are:

- **High availability, scalability, reliability**
- **The ability to process complex analytics, forecasts and planning** in the same database system that supports your Business Intelligence applications.
- **Security** for all data in the database, including multidimensional data.

- **Reduced update time** because all of the data is in the same database system.
- **Open access** to both relational and multidimensional data.
- **Expanded capabilities** of standard reporting tools. Multidimensional results are presented in a relational format and can be accessed through existing relational tools and applications.
- **Reduced maintenance** because all of the data is in the same database system.

SECTION 2: OLAP OBJECTS IN THE ORACLE *9i* DATABASE

The objects and features described in this section play a central role in OLAP processing. You will need to know how to create, populate and maintain them in the Oracle*9i* integrated database. Later, you will see how to use OWB to create and maintain these objects.

There are 3 different ways to create, populate, and maintain these objects.

1. **An entirely manual process.**
   The relational cube, metadata, table function, and the view can be created using SQL or PL/SQL commands. The Analytic Workspace (AW) and the multidimensional cube can be created using the OLAP DML.
2. **A combination of manual coding and tools.**
   The relational cube, the metadata, table function, and the view over the AW can all be created using Enterprise Manager. The AW and the multidimensional cube can be built using the AW Manager graphical tool.
3. **Entirely through the OWB design environment.**
   This is the option that will be illustrated later in this paper through a case study.

The following objects are described in this section:

- Relational Cube
- Multidimensional Cube
- The Analytic Workspace
- Metadata
- The OLAP Table Function
- A Different View on the Analytic Workspace

For more detailed information see the Oracle9*i* OLAP User's Guide.

RELATIONAL CUBE

A fundamental concept in OLAP processing is that of a cube. A cube is one way to represent data in an organized and somewhat intuitive way. The relational cube is modeled from a star schema. Star schemas are not a new concept, and are commonly used to support the analysis of data stored in relational tables. The table at the center of the star contains the factual data that is the target of analysis, for example SALES. The surrounding tables help put the fact data within a context; for example the context of a particular product, customer, geography and time.

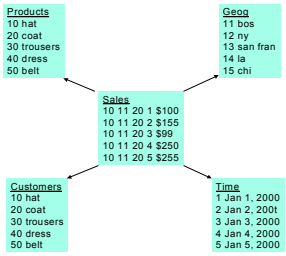


Figure 2: Star schema – the relational cube

MULTIDIMENSIONAL CUBE

The multidimensional cube can be visualized as in *Figure 3*. The factual data is within each cell of the cube. The contextual data runs along the edges. Multidimensional cubes are stored in Analytic Workspaces (AW). A particular AW may contain more than one cube.

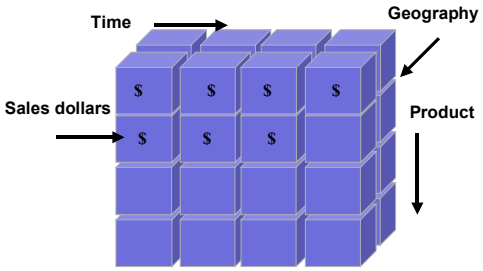


Figure 3: The OLAP multidimensional cube

THE ANALYTIC WORKSPACE

It is the Analytic Workspace (AW) and the OLAP engine that brings new analytic capabilities to the Oracle*9i* Database. Within the context of the Relational Database, the AW is a LOB which is a multidimensional data type, and it is stored in a column in a relational table. But in the context of OLAP the AW is the host for multidimensional data and process descriptions. Both the data and the processes are defined and stored in the format best suited for calculation and manipulation by the OLAP engine.

Some of the multidimensional objects in an AW represent metadata and contain metadata values. Other objects contain multidimensional data values. And there are objects that represent calculations, functions and programs.

Programmers can use the OLAP DML to create calculations, functions and programs to build analytic applications. The OLAP DML is a 4GL that has rich analytic functionality built into its commands and functions.

Later you will see that you can use OWB to automatically create an AW, create the multidimensional objects within it, and load data into it.

AWs can be populated with data from another AW, a flat file, or from relational tables. Through Warehouse Builder you can populate your AW from any of these sources.

METADATA

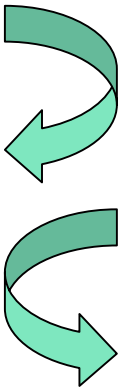
Metadata is data about data. It provides important information that helps turn data into business intelligence. Metadata contains both logical descriptions and physical mappings. The logical description of data includes information about dimensions, levels, hierarchies, attributes and relationships. Mappings associate the logical descriptions to the physical location of the data. For example, a product description can be mapped to a column in a table or view. In addition to providing the business context for your data, metadata isolates applications from changes to the underlying data. If the physical location of the data changes then the metadata mapping changes, the application stays the same.

Metadata for your warehouse is identified during the design phase, and is determined by its ability to support the user processing requirements. OWB provides easy-to-use wizards and graphical table editors for defining metadata, and the mappings, to the warehouse build process. During the build process the metadata definitions are written to the OLAP Catalog. Metadata for SQL access to all OLAP data, whether that data is stored in a relational or multidimensional format, is stored in the OLAP Catalog.

The AW also contains metadata. If you tell OWB to put your data in an AW, then OWB will automatically generate the metadata for the AW based on definitions in the OLAP Catalog.

THE OLAP TABLE FUNCTION

The OLAP_TABLE table function provides the user with a simple way to perform technologically complex operations. One feature of the Oracle Database Object technology is the table function. It provides a means of accessing non-relational data.



For Oracle*9i*, a table function, called OLAP_TABLE, was implemented to support the new multidimensional technology within the database.

These are some of the characteristics and features of the OLAP_TABLE function.

- Provides a way to map the multidimensional data in the AW to a two-dimensional relational format for output.
- Its definition can be created and stored as a relational object, and then referred to in a select statement, or its definition can be imbedded in a select statement.
- It can be used wherever you would use the name of a table or view.
- It extracts the targeted AW data from the LOBs in which the data has been stored.
- It presents fully solved data that is either stored or calculated in the AW.

A DIFFERENT VIEW ON THE AW

Oracle*9i* provides a way access the multidimensional data in the AW through SQL. To enable SQL access to the data in an AW, you create a view that contains an OLAP table function in its definition. That's it! This very powerful feature of the Oracle*9i* Database is easy to implement, and you will see that with OWB even this easy task can be automated.

Once the view has been created the data in the AW can be accessed by applications that generate SQL, and by applications that write to the OLAP API. The OLAP API will generate the SQL against the view. Mappings in the metadata tell the API which views to select against.

In this architecture the view over the AW can be thought of as a materialized view (MV) without the overhead of the MV; the definition of the view is stored in the database, but it is only populated when a select statement is issued against it. One of the most valuable attributes of the AW/OLAP engine components is the efficient throughput of data requests. This efficiency contributes to the successful use of table function in this architecture.

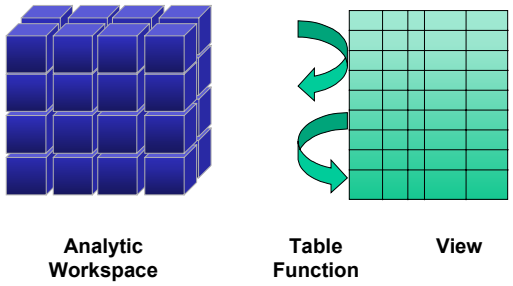


Figure 4: A view over the AW

THE OLAP STAGE IS SET

At this point you have seen the architecture of the Oracle*9i* Database with OLAP, and are familiar with some of the objects and concepts that support OLAP processing. The next section will describe how, by using OWB to build and maintain these objects in your data warehouse, you can incorporate Oracle*9i* OLAP functionality into your current business processes with minimal effort.

SECTION 3: WAREHOUSE BUILDER

Oracle*9i* Warehouse Builder (OWB) is all about turning data into business intelligence. It is the Oracle tool that provides a way to design and deploy Business Intelligence applications.

Warehouse Builder specializes in System Design and enterprise data integration. These are two of the features that allow you to more easily achieve the complex task of turning data into business intelligence. OWB is one of the few tools available that combines this functionality.

On the data integration side, OWB allows you to integrate data from disparate sources in a scalable and easy to understand way. On the system design side, the tool gives you the flexibility to design both relational and multi-dimensional schemas.

Topics in this section are:

- The Data Integration capabilities
- The Design capabilities
- The Infrastructure

THE DATA INTEGRATION CAPABILITIES

Moving data from a source system to a warehouse system is a complex task. Often the source is a transactional system. In this case, the data model would have been designed to support the efficient collection and storage of business transactions. A transaction data model is not the most efficient way to represent data that will be analyzed. Therefore, there is a need to transform the data as it is moved into the data warehouse. Some of the transformations change the data model, some change the number of rows that represent the information, and some change the actual data content. For the Extract, Transform, and Load activities, it is the Transform that is the most complex. The Warehouse Builder design environment provides a unique and intuitive graphical user interface to make this task simpler and faster.

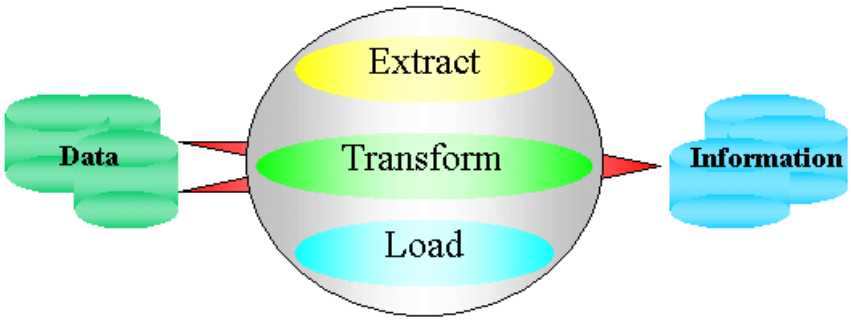


Figure 5: ETL, turning data into information

While the ease-of-use that the design interface provides is a welcome benefit, other key attributes of any data integration tool are scalability and performance. As an Oracle tool, OWB takes full advantage of the scalability and superior performance of the Oracle Database engine. OWB delivers new ETL features that further leverage and optimize your investment in Oracle technology.

THE DESIGN CAPABILITIES

One reason that the Warehouse Builder user interface is so intuitive is that its objects match the user's mental model of a multidimensional system. For example, the visual representation of a hierarchy presents a top to bottom arrangement of the levels, and a cube is presented as a fact table surrounded by its dimensions. As you will see later, even the mappings and transformations are accurately represented in a graphical manner.

Warehouse Builder provides wizards that guide you through the creation of complex objects such as dimensions and cubes, making the design process easy, logical, and fast. Once you have completed your design activities, OWB has a unique view of your whole system. The OWB repository contains both the ETL metadata, and the metadata that will support your application processing. OWB offers a web-based reporting environment that allows developers and business users to easily implement advanced metadata reporting.

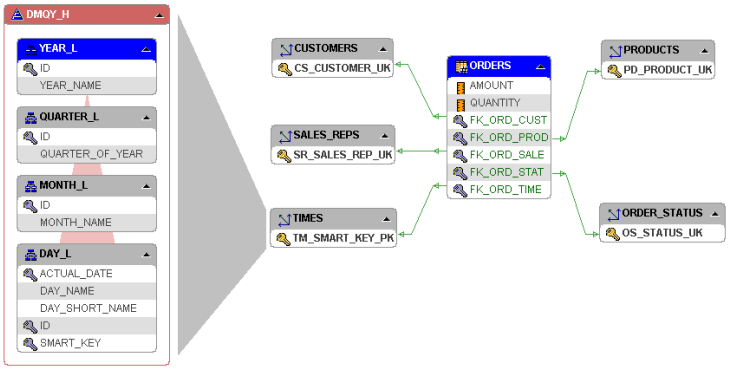


Figure 6: Graphical design capabilities

As you know, all database systems are subject to change, but a Business Intelligence system is more volatile than most. Constant enhancements, changes, and additions to the sources can make maintenance and interactive design a tough job. Not with OWB. OWB supports the full life cycle of the system. It provides an easy way to reconcile changes to both sources and targets, with the internal metadata, thereby reducing the effort and cost associated with making a change.

THE INFRASTRUCTURE

These are the components that support these advanced OWB capabilities:

- The Repository – A set of relational tables and views in an Oracle database that stores the metadata definitions
- A Client Application – This java application provides you with an easy-to-use graphical interface that guides you through the definition and design of the source data, application metadata, and data loads
- A Code Generator – This component generates scripts based on the metadata in the repository, which you then apply to your target database schema. The generator is designed to utilize the full feature set of the Oracle database, and the generated scripts provide optimal performance on the Oracle database
- A Runtime Environment – The target environment not only stores the data loaded from the sources in a schema designed and created by you; it also holds audit information on the loading processes that move the data into this schema

- A Runtime Audit Viewer – A Java application to inspect the audit information in the runtime environment
- A Reporting Environment – The reporting capabilities of Warehouse Builder allow you to view your metadata from a web-based application, and provide information to a larger audience. The information is aggregated for more business-oriented users

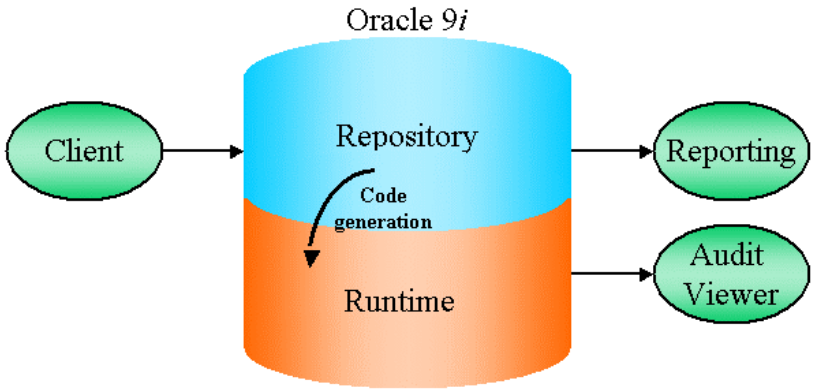


Figure 7: The infrastructure

For more detailed information on each of these components refer to the Oracle9*i* Warehouse Builder Architecture white paper, or the Oracle9*i* Warehouse Builder White Paper on OTN. With this infrastructure in place you are ready to build and manage your warehouse environment.

SECTION 4: BUILDING THE WAREHOUSE

Building a data warehouse with OWB is a logical process. Extending the functionality to include OLAP does not interrupt the logical flow. A typical sequence of steps is:

- Design your target Warehouse
- Build your ETL process to move data into the target
- Generate the code and create the warehouse
- Run the code to populate the Warehouse

DESIGN THE TARGET WAREHOUSE

The business requirements drive the design of the data warehouse. One of the most popular designs in data warehousing is the star schema. OWB supports this design, and also supports any normalized schema design. Using this design capability, OWB now supports the creation of multidimensional objects for the Analytic Workspace for those users whose business requires the analytic capabilities of a multidimensional analytic engine.

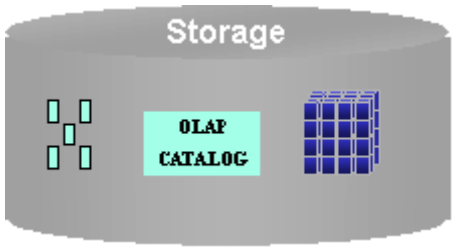


Figure 8: A target warehouse

Below are the relational objects supported in Warehouse Builder:

- Tables – Including constraints and indexes
- Dimensions – Including the relational dimension object and storage table
- Fact tables – Including bitmap indexes and foreign keys to the dimensions
- Views – Including the view query and constraints
- Materialized Views – Including the refresh options

Warehouse Builder is uniquely positioned to work with the Analytic Workspace constructs from a design perspective. The metadata definition is abstracted from the implementation. That means that the user can define a dimension once, and that definition can be implemented for both a relational schema and the multidimensional AW.

These are the design objects that are re-used:

- Dimensions – representing dimensions in OLAP
- Fact tables – representing cubes in OLAP

BUILD THE ETL PROCESS

To define the ETL process, you identify the source data, define data transformations, and map sources to targets.

Identify the source data - One characteristic of a data warehouse is that it is the repository for data that can come from multiple disparate sources. It is within the ETL process that these data sources are transformed into a common form and content. The first step in designing and building an ETL process then is to identify and chart the source data.

Warehouse Builder provides specific access methods for various sources:

- Oracle catalog (relational objects as well as Analytic Workspaces)
- Oracle Applications
- Flat Files
- SAP/R3
- Mainframes (via Oracle Transparent Gateways)
- Relational Databases (DB2, SQL Server, Sybase and Informix)
- ODBC Sources

You use the Warehouse Builder New Module Wizard to define sources, and the Metadata Import Wizard to execute the load. Once the source metadata is available in Warehouse Builder you can use the graphical interface to map the source elements to the intermediate structures or the target structures.

Note that the data warehouse can be built in one transform and load step, or in multiple steps. The approach that you take will be determined by your business requirements and practices.

Data transformation and mapping – Defining transformations and mappings are the activities that require the most thought, and involve the most complexity. The Warehouse Builder mapping interface, with its built-in features, makes it easier. The mapping editor provides a way for you to define all of the changes in the data as it is moved from the source to the target. This is also where you identify the data that is targeted for OLAP processing.

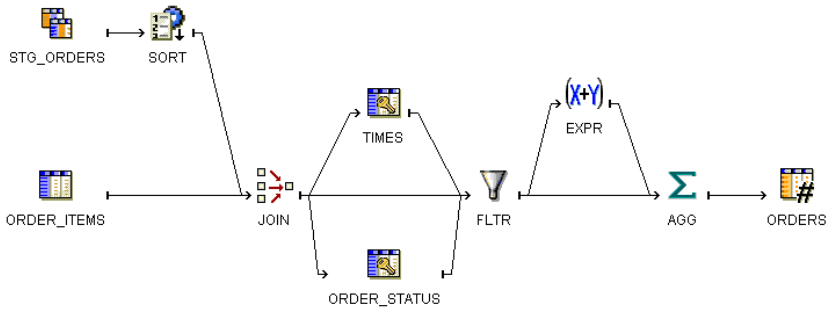


Figure 9: OWB mapping

Think of mappings as the way to determine or change the **cardinality** of the target rowsets, and think of transformations as the way to determine or change **content** of the target rowsets. Warehouse Builder provides a complete set of mapping operators and transformations to achieve even the most complex transformations. You can aggregate, sort, perform custom transformations and cleanse data, all using the same intuitive user interface. **This is where your data becomes information.**

While Warehouse Builder provides a rich set of operations and calculations for the relational data, there are some operations and calculations that can only be processed by the OLAP engine over data that is within an Analytic Workspace. Built-in AW functions can be used, or the application developer can use the OLAP DML to define operations and transformations to be performed by the OLAP engine. OWB can trigger these processes.

Some of the built-in functions of the OLAP DML are:

- Allocation
- Aggregation
- Forecasting – Straight-line Trend, Exponential Growth, and Holt-Winters extrapolation.
  In addition to these three forecasting functions, the OLAP engine provides very sophisticated forecast methods. For more information refer to the Oracle9*i* OLAP DML Guide.
- Numeric and Time Series Functions – such as Average, Cumulative Sums, Lead/Lag, Variance, Moving Average/Min/Max/Total, and Standard Deviation.
- Financial Functions – such as Depreciation, Growth Rate, Net Present Value and Internal Rate of Return.

You can also create custom calculations, operations and models. A model is a workspace element that contains a set of related equations that are based on the values of a specific dimension. They are commonly used with financial modeling. For more information on the calculation capabilities of the OLAP DML and the OLAP Engine refer to the Oracle9*i* OLAP DML Guide.

Validate your design - Prepare for the warehouse deployment step by running the OWB validation procedure. The validation procedure is run against the warehouse design to check for errors in the configuration. It verifies foreign key references, object references, data type matches, and other configuration properties. Warehouse Builder publishes the errors, which can then be corrected using the graphical editors. Once validated, the warehouse design can be deployed.

GENERATE THECODE AND CREATE THE WAREHOUSE

There are three steps to warehouse deployment. First, the build scripts are generated from the information contained in the design. Second, the warehouse is created, which causes the database objects to be instantiated as physical entities. The third step populates the objects with data. Populating the warehouse is covered in the next section.

Generate the scripts

OWB generates the following types of code to build the warehouse:

- SQL Data Definition Language representing the data objects
- PL/SQL programs representing the mappings
- SQL*Loader control files representing flat file mappings
- ABAP code representing SAP R/3 mappings
- Tcl scripts for scheduling mappings in Enterprise Manager

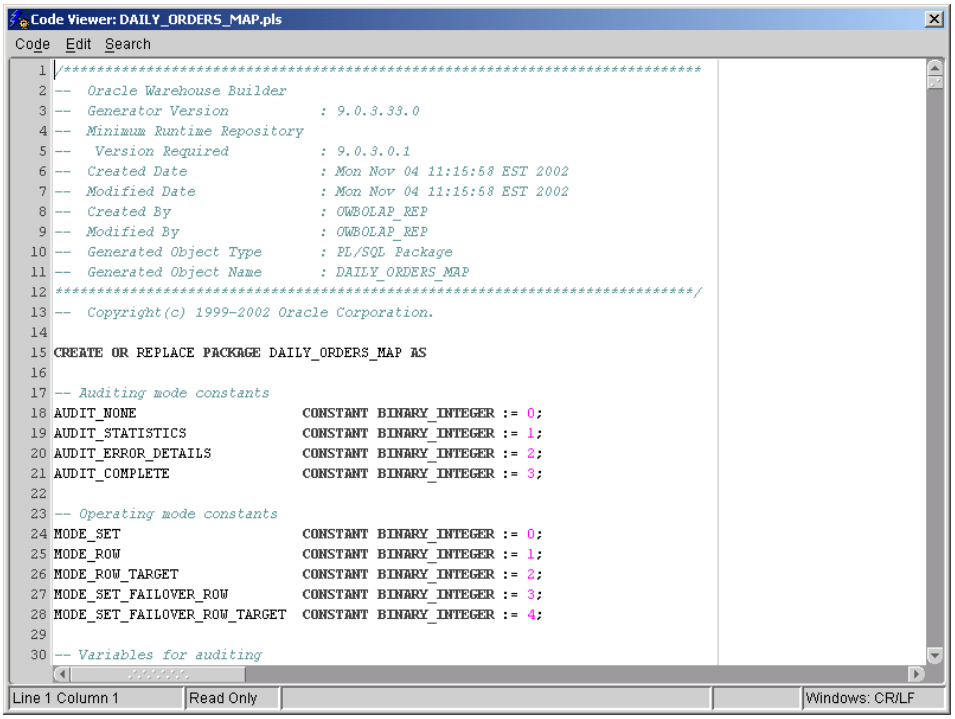


Figure 10: Sample build code

In this phase the OWB Code generator takes advantage of database features that significantly improve the performance of the extraction and loading process. Some of the built-in performance techniques that are used during the ETL process are:

- Oracle *9i* MERGE code
- Parallel processing in the database
- PL/SQL Bulk Processing

- Foreign Key Constraint Manipulation
- Partition Exchange Loading

Create the warehouse

Warehouse Builder provides a runtime environment into which all of the build code is deployed. During the deployment, data load information is automatically written to pre-defined structures. These structures can be easily viewed to determine the success of the loads.

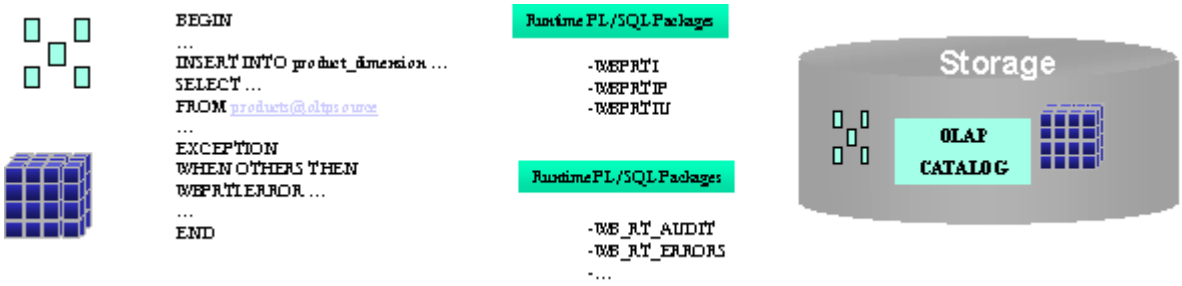


Figure 11: The OWB runtime environment

Warehouse Builder instantiates all of the relational objects and the multidimensional objects. These are the OWB tasks associated with preparing the warehouse for OLAP:

- Instantiate the multidimensional AW from the relational tables
- Construct the AW representation of the data from the same OLAP Catalog metadata that represents the relational data
- Build views over AWs to allow SQL access to the multidimensional data

RUN THE CODE TO LOAD THE WAREHOUSE

This last phase of the process uses the mappings you have created to load the data into the target. Some important features of the processing in this step are:

- Option to Load or Refresh the relational data objects from the sources
- Option to Load or Refresh the multidimensional objects from the relational data objects

> *Warehouse Builder uses the Oracle9i database to process ETL activities in a fast and efficient way. Using the latest techniques to load and synchronize the data, it has unrivalled performance. The value proposition here is worth pointing out. As mentioned earlier, the Oracle9i database architecture provides the ability to process both relational and multidimensional data in the same database instance, which eliminates the need for a separate multidimensional system. In addition, OWB makes it easy to incorporate the new capabilities into your existing business processes. The power of the database enhances the value of OWB, and the functionality and ease-of-use of OWB enhances the power of the database.*

Ready to use - Once the data is loaded, the warehouse is ready for OLAP processing. All of the objects that support OLAP in the Orcale9*i* database have been created. The case study will highlight the specific activities that enable the full Oracle OLAP functionality.

Users can choose from a broad range of tools when selecting a method for analyzing data and reporting on the results. Here is a list of some of the Oracle tools that are available.

- Business Intelligence Beans – Use the Java based BI Beans to rapidly develop custom analytic application. BI Beans are Java based, and are included in the Jdeveloper tool set.
- OLAP API – Develop custom Java applications using Oracle's OLAP aware API.
- Discoverer – Use Oracle's ad-hoc query tool to get the most out of your warehouse.
- Reports – Use Oracle Reports to create high definition enterprise class reports.

Using OWB to build your OLAP warehouse enhances the value of using using these Oracle tools. OWB efficiently uses the metadata gathered during the design phase to create the Discoverer and BI Beans environments.

SECTION 5: A SCENARIO - USE OWB TO BUILD YOUR OLAP DATA WAREHOUSE

This section describes a scenario and then presents a case study based on that scenario. The case study will be brought through the process of building a warehouse. The activities that transform the data and prepare the data warehouse for OLAP processing will be highlighted.

THE SCENARIO

In this scenario the source data will be transformed and used to build an OLAP ready data warehouse.

- The source is transactional data that is stored in a relational schema. It contains information for 6 dimensions and 3 facts in 16 tables.
- The data model of the source schema is a snowflake. That means the information for some of the dimensions spans multiple tables.
- The data model for the target schema will be a star. That means all information for a dimension will be contained in one table.
- In the new warehouse the transformed source data will be stored in a relational format.
- Data that is targeted for OLAP analysis in the new warehouse will be stored in a multidimensional format.
- Data in the multidimensional format (in an AW) will be analyzed using the built-in functionality of the OLAP engine.
- The results of the multidimensional analysis in the AW will be accessed through a SQL select against a SQL view.

SECTION 6: THE CASE STUDY

TEN STEPS TO AN OLAP-READY WAREHOUSE.

Here, some detail has been added to the high level steps.

- **Design the Target Warehouse**
  1. Review the source data.
  2. Determine the requirements of the target warehouse.
  3. Design the target schema and the data transformations.
- **Build your ETL process to move data into the target**
  4. Define the data source to OWB, and import the source metadata.
  5. Define the objects for the target schema – Dimensions, Levels, Hierarchies, Attributes, Facts, Transformations, and Calculations.
  6. Map the sources to the targets. Include the transformations and post-mapping processes.
  7. Validate the design.
- **Generate and create**
  8. Generate the code.
  9. Instantiate the objects.
- **Run the code to populate the Warehouse**
  10. Populate the warehouse.

DESIGN THE TARGET WAREHOUSE

Here we look at the shape and content of the source data, determine the requirements of the target warehouse, determine what needs to be done to transform the source data into the shape and content of the target schema, and design the warehouse objects for the target schema.

REVIEW THE SOURCE DATA

- Below is a representation of the source data that is stored in a relational snowflake schema. The schema contains information for 6 dimensions and 3 facts in 16 tables.

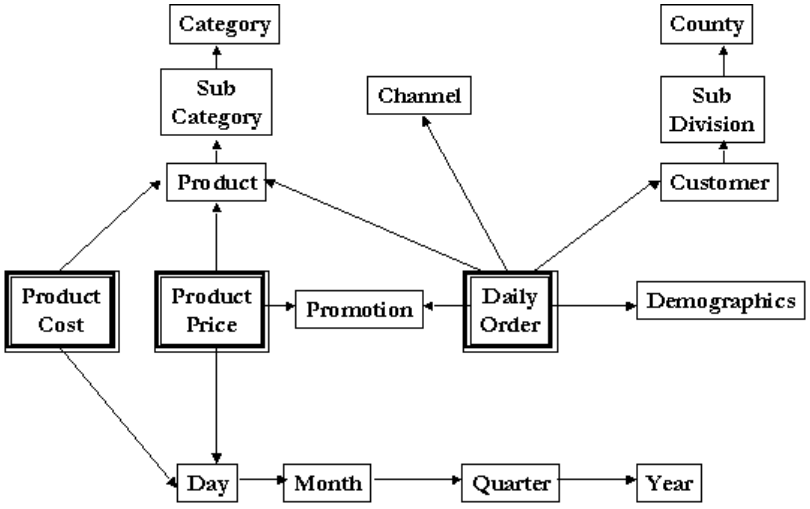


Figure 12: Review the source data model

DETERMINE THE REQUIREMENTS OF THE TARGET WAREHOUSE

The requirements of the target warehouse are:

- The data model to be used in the target warehouse is a star data model.
- All cubes will be targeted for OLAP analysis.

DESIGN THE TARGET SCHEMA AND THE DATA TRANSFORMATIONS

The data model for the target schema will be a star. The Product, Customer, and Time dimensions will be collapsed into one table.

Other transformations that will occur are:

- Time dimension – Some of the data will be converted to the DATE type. Some period names will be calculated.

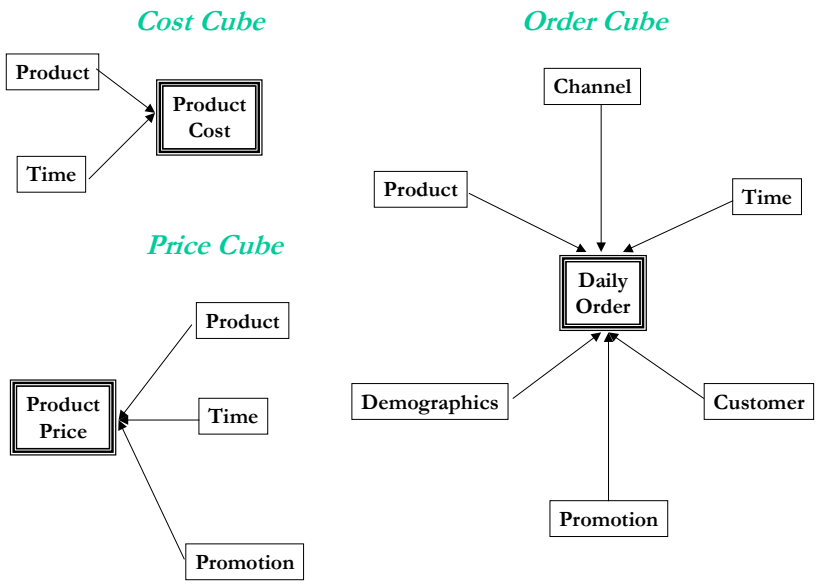


Figure 13: Design the target data model

BUILD THE ETL PROCESS

USE OWB TO IMPLEMENT THE DESIGN

Use the OWB design wizards and table editors to define the sources, targets, and transformations. The target definitions will include dimension, hierarchy, attribute and fact objects. The OWB interface is very intuitive, so all of the user actions will match the common concepts contained in a multidimensional data model.

DEFINE THE DATA SOURCE TO OWB

Use the OWB module wizard to define the source information, and to trigger the metadata import. This is what you need to provide:

- Name the module and identify it as type Data Source
- Select the application type, version, and integrator. For the case study the type will be Generic Oracle Database Application, version is *9i*, and the integrator defaults in as Oracle OWB Integrator for Oracle DB and Apps 3.0
- Tell OWB where the source metadata is stored. In this case it is the Oracle Data Dictionary.
- Provide connection information that includes the owner of the OWB repository, the username that has access to the source data, the name of the source schema, and the machine connect string.

The diagram below shows some of the source information, as well as the results of the subsequent source metadata import for the case study. Notice that the Source Module Properties pop-up window contains connection information. The listing on the left side of the main window contains the imported source metadata, which is presented as 16 relational tables. Use the Import Metadata wizard to bring the source metadata into your OWB project.

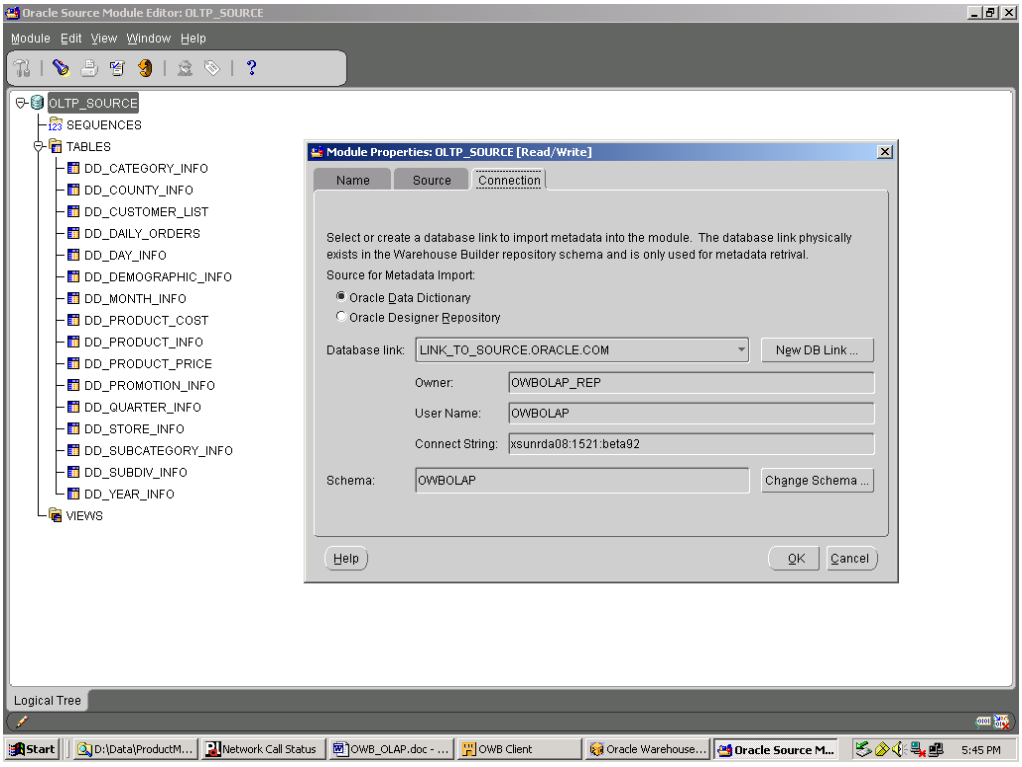


Figure 14: Define Oracle relational tables as a data source

DEFINE THE OBJECTS FOR THE TARGET SCHEMA

This is where the multidimensional metadata objects are defined. You will provide information for Dimensions, Levels, Hierarchies, Attributes, Facts, Transformations, and Calculations. Here we walk through the definition and mapping of one dimension and one cube.

DEFINE THE PRODUCT DIMENSION

Use the OWB Dimension Wizard to define dimensions. The Dimension Wizard leads you through a series of windows that prompt you for the following information:

- The name of the dimension
- The names of each of the attributes for each level
- The relationship of the levels within each hierarchy
- The names of each of the levels
- The name of the hierarchy(s)

Dimension editor - Once the dimension is created, all of its characteristics can be viewed though the Warehouse Builder Dimension Editor window. When you open the editor for a particular dimension, the initial window contains a graphical representation of the hierarchies that are defined for that dimension. Refer to the left side of the window in the diagram below. To open the pop-up window on the right side, select Edit/Object Properties from the menu bar. Here you can view all of the characteristics of the dimension.

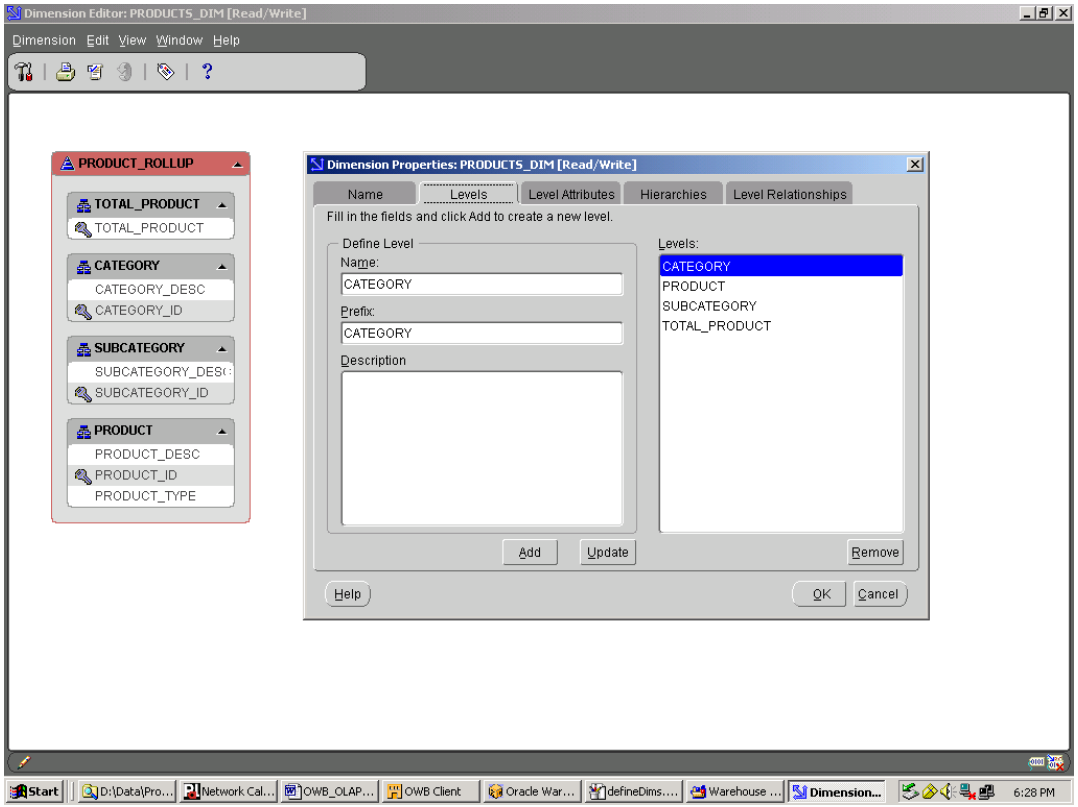


Figure 15: Use the Dimension Editor to inspect the Product Dimension

MAP SOURCES TO TARGETS

After the multidimensional objects for the target warehouse are defined, use the OWB Mapping Editor to map the sources to the targets. This mapping defines the ETL process. It describes both the relational ETL process and the OLAP ETL process. The Toolbox in the Mapping Editor makes this an easy task. For the case study we will show the mapping and transformations for the product dimension, and the daily orders cube.

MAP THE PRODUCT DIMENSION

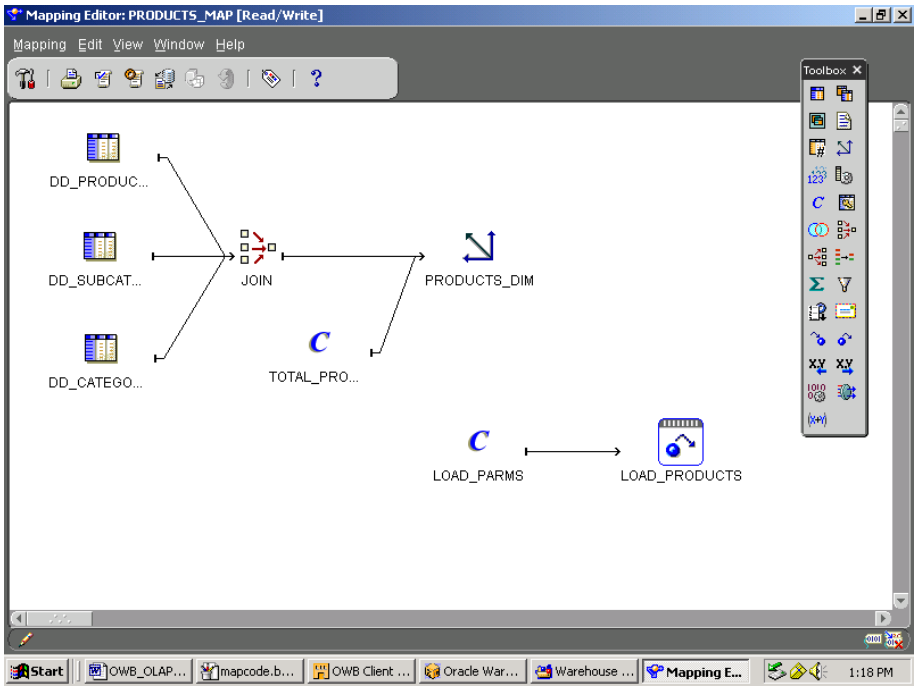


Figure 16: Use the Mapping Editor to describe the ETL process for the Product Dimension

Some characteristics of this Product Dimension mapping are:

- One transformation that needs to occur during the build process is the collapsing of the multiple source product tables into one product dimension table in the target. Notice that the JOIN operator is used to collapse multiple inputs into one output.
- Another transformation for the product dimension is that a total column will be added. It will have the same data content for every row. Therefore, we create a constant with a value of "Total Product," and add it to every row in the Product dimension table. Below you can see the contents of the JOIN operator and the TOTAL_PRODUCT constant.
- The Product Dimension is targeted for OLAP processing, so the mapping contains information for the OLAP ETL.

The diagrams that follow illustrate how this information is described to the mapping process.

INSPECT THE PRODUCT DIMENSION JOIN OPERATOR

Below is the Expression Builder window. This editor is accessed from the Operator Properties Inspector or the Attribute Properties Inspector. Here, on the left side of the window, you can see that the relevant columns from each of the three product tables have been mapped into the join operator. Notice the join conditions on the upper right side of the window. And, at the lower right, see the results of the validation procedure.

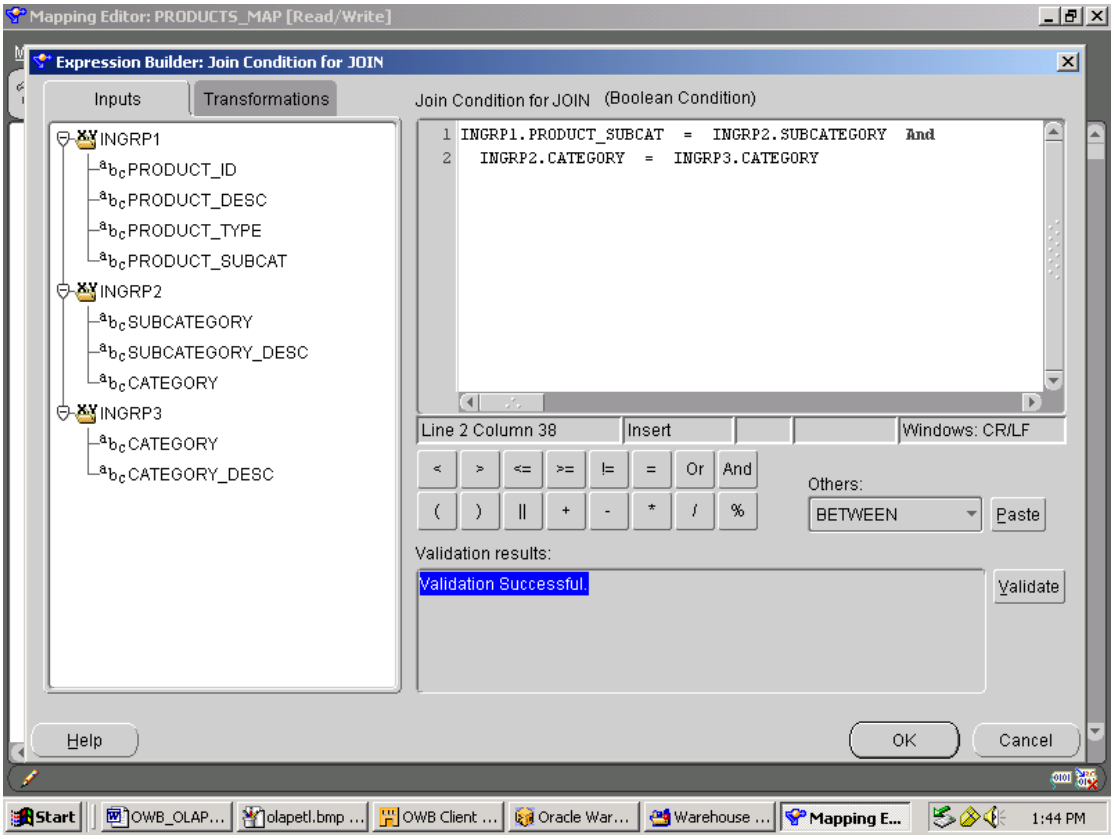


Figure 17: Contents of the JOIN operator for the Product Dimension

INSPECT THE PRODUCT DIMENSION CONSTANT OPERATOR

In the window below, the constant, TOTAL_PRODUCT_CONST, is expanded to show that it contains one attribute called TOTAL_PRODUCT_ATTR. The pop-up window below it is the Attributes Properites Inspector. Notice that the value "Total Product" is entered in the Expression field. This is the value that will be put in the TOTAL_PRODUCT_ID in the Product dimension table as the mapping indicates. The Mapping Editor window also shows the mapping of the other input columns to the Product dimension.

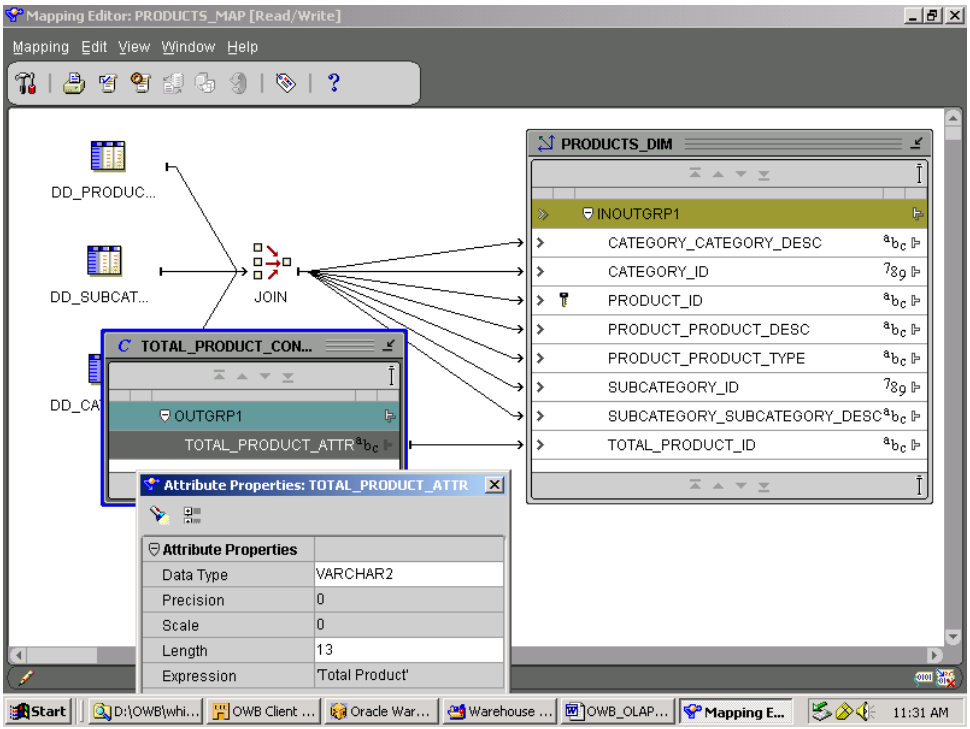


Figure 18: Contents of the TOTAL_PRODUCT_CONST for the Product Dimension mapping

INSPECT THE OLAP ETL FOR THE PRODUCT DIMENSION

Two components in the Product Dimension mapping represent the OLAP ETL process. The object titled LOAD_PRODUCTS is a post-mapping process that represents the OWB procedure, WB_OLAP_LOAD_DIMENSION, which loads dimensions into AWs. The object LOAD_PARMS provides values for the procedure input parameters.

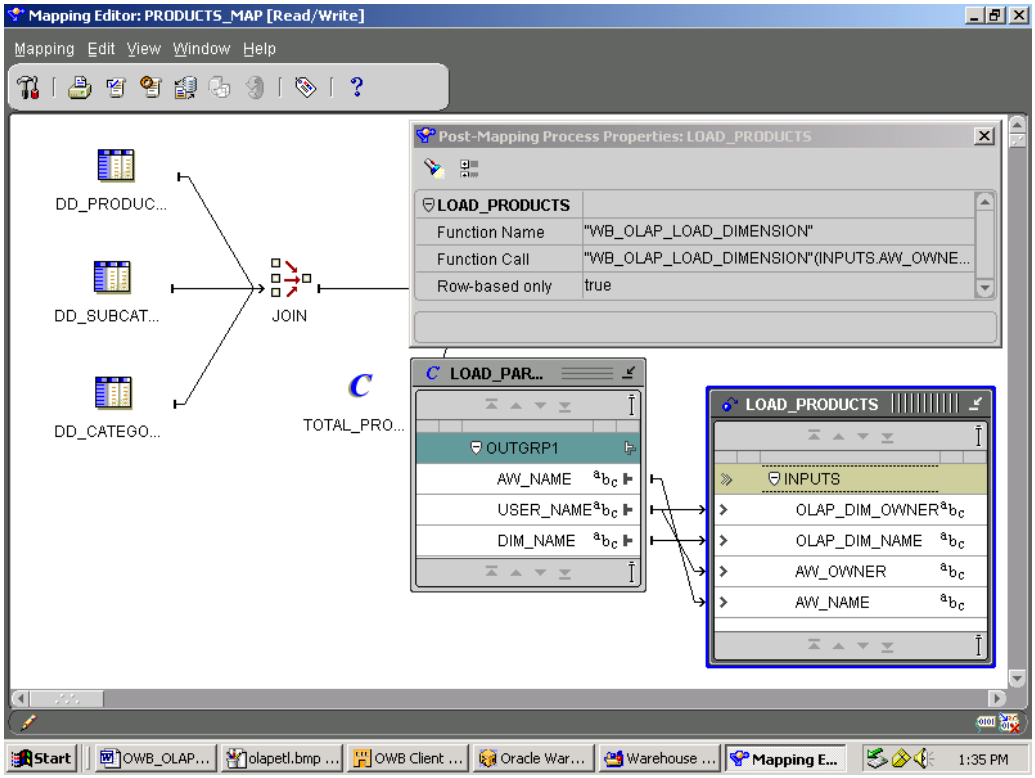


Figure 19: The Product Dimension post-mapping process

DEFINE THE DAILY ORDERS FACT

Use the Fact wizard to create the Daily Orders fact. Provide the fact name, identify the foreign keys, and add the measures and their type. Once all of the information is entered, the fact is graphically presented as a star data model. It is important to note that this graphical representation conveys the relationship between the data, and it does not imply a particular implementation of the data model. This diagram effectively represents both a relational implementation and a multidimensional implementation. This is one example of how OWB abstracts the metadata from the implementation.

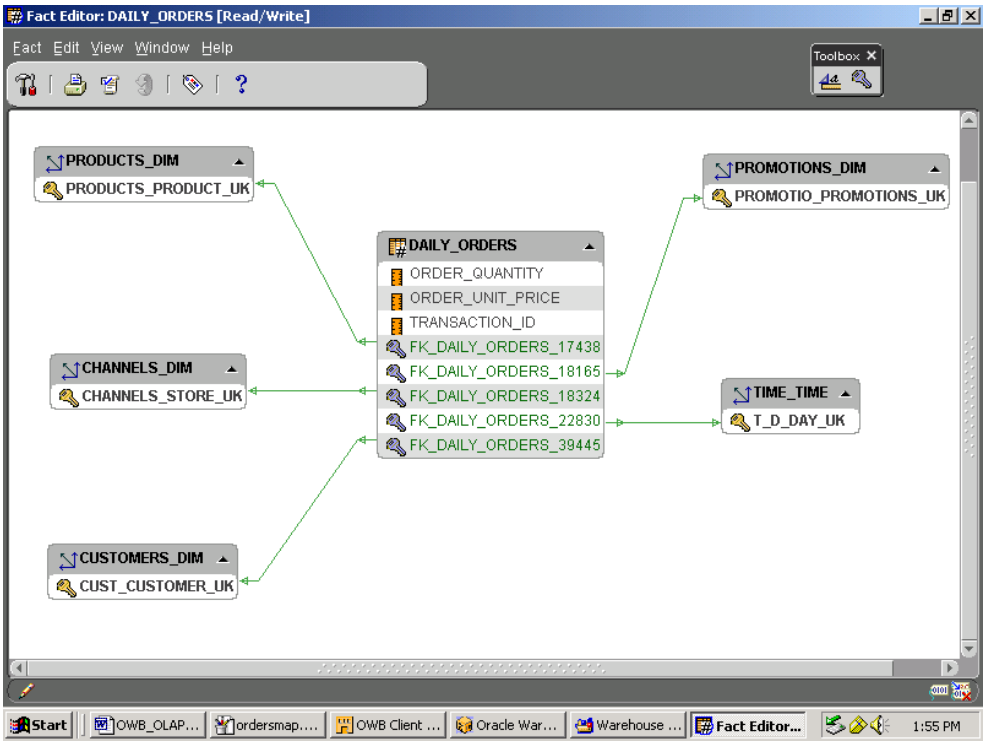


Figure 20: Define the Daily Orders fact

MAP THE DAILY ORDERS CUBE

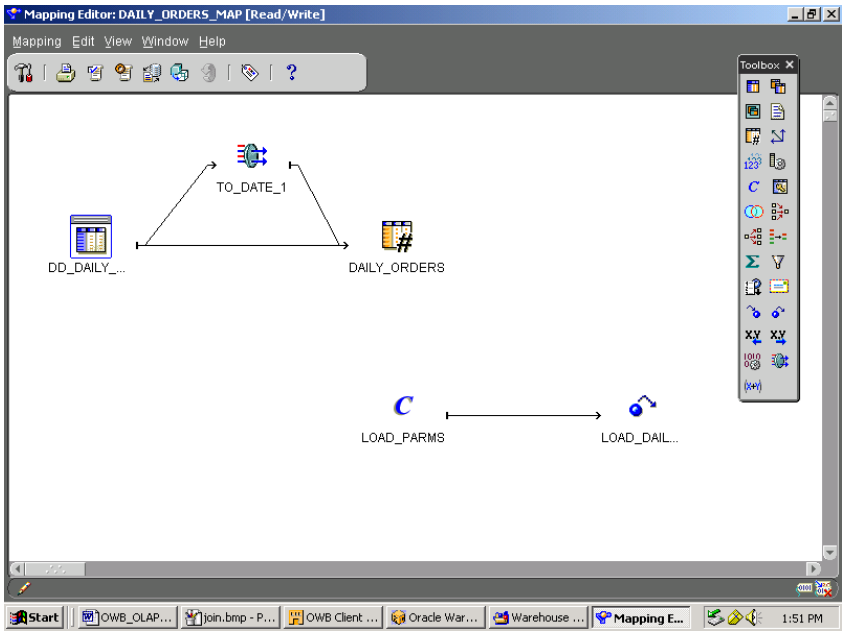


Figure 21: Map the Daily Orders cube

You can see that there is only one transformation defined for the Daily Orders cube. It is called TO_DATE_1, and is a user defined function that implements an Oracle defined function. See the TO_DATE_1 function listed below on the left side of the window under Transformation Libraries. The pop-up window on the right shows the implementation code. In this case, the user used PL SQL code to add a static definition for two of the input parameters of the standard TO_DATE function.

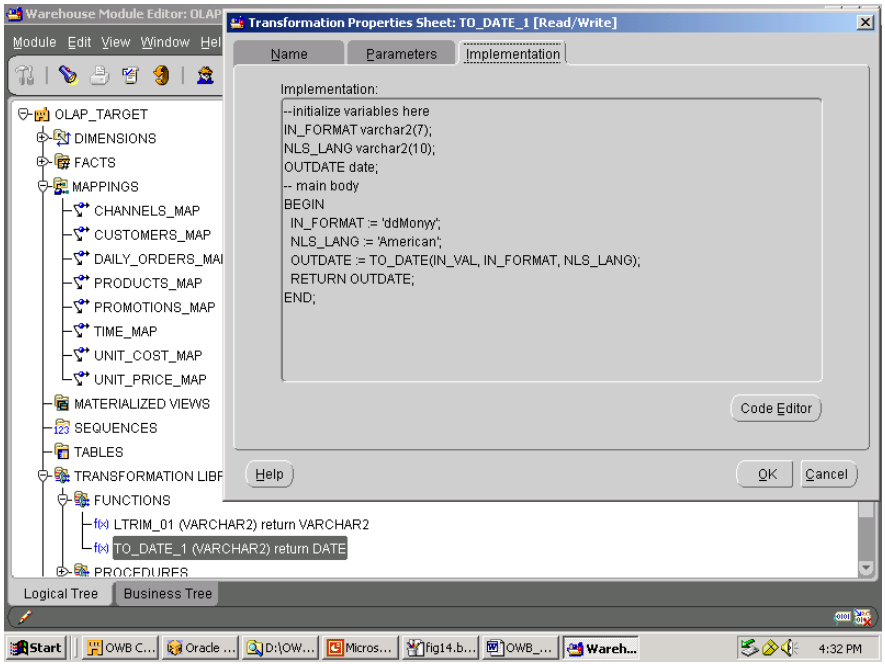


Figure 22: Implementation code for the TO_DATE_1 function

VALIDATE THE DESIGN

After all of the objects for the target warehouse are defined and mapped, use the validation procedure to validate the warehouse design.

For the case study, the procedure has validated that all of the objects, transformations, and mappings are syntactically and logically correct. The target warehouse is now ready to be deployed and populated.

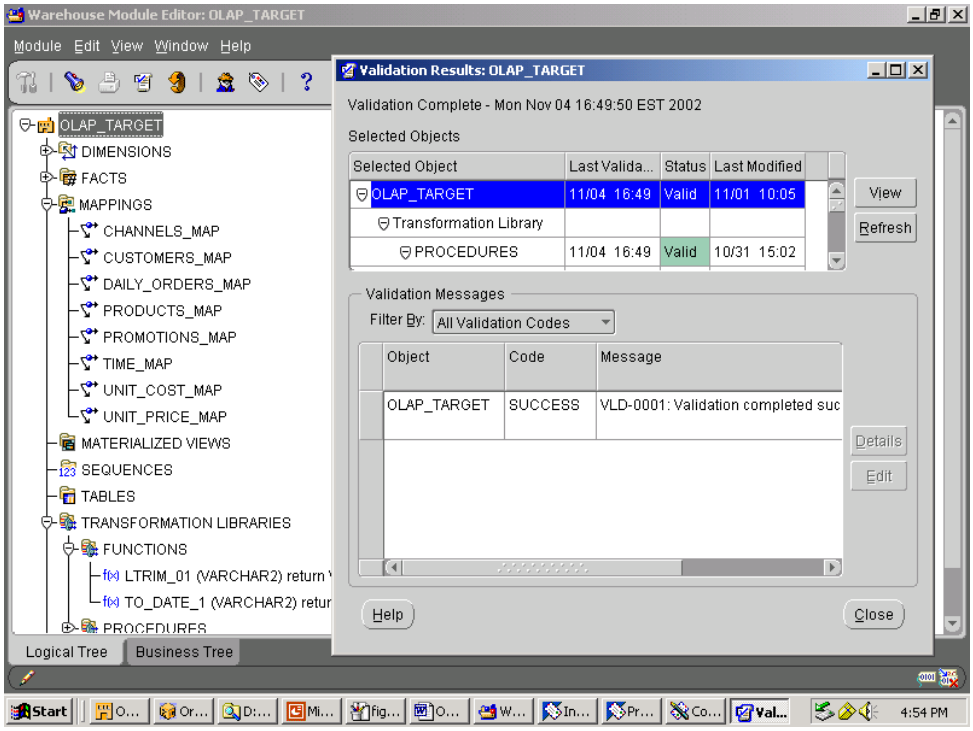


Figure 23: Validation procedure results window

GENERATE AND CREATE

GENERATE THE CODE

Generate is an option that is available from multiple locations in the OWB interface, including the Project menu and various object menus. This is where you specify whether this is a new build or an update to an existing warehouse.

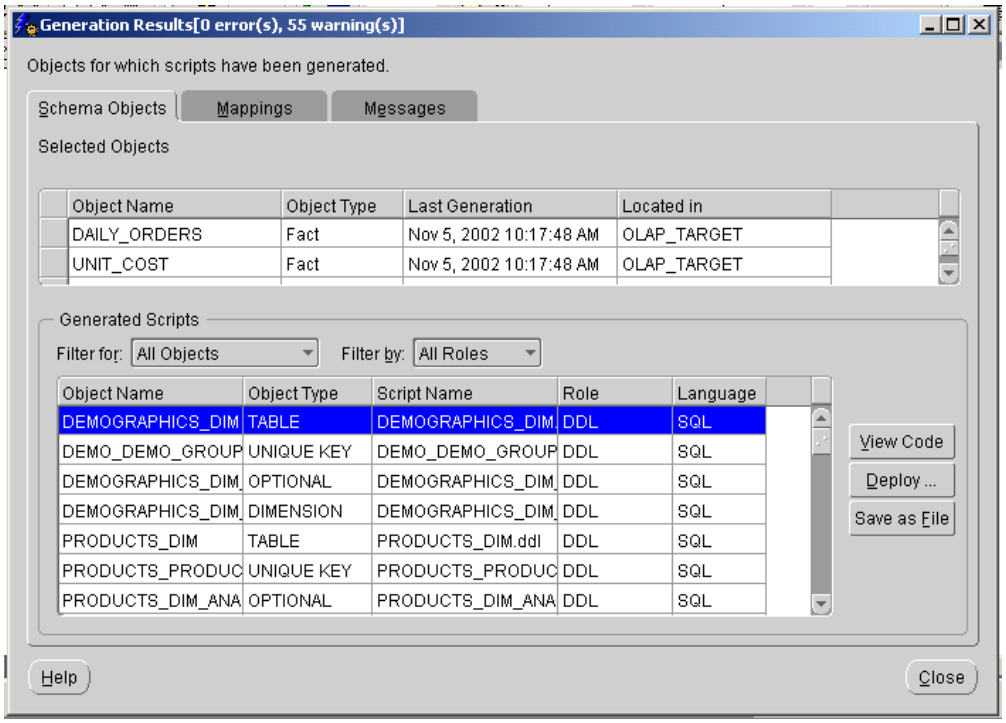


Figure 24: Results window for object script generation

CREATE THE OBJECTS

The initial Generate results window presents the Schema Objects tab, which contains the list of scripts that will instantiate the database objects. From here you can view the code, deploy the scripts, or save the scripts and execute them at will.

When you select Deploy, all of the relational objects are instantiated. You then use the Metadata Transfer wizard to trigger the deployment of the OLAP objects and metadata definitions.

One ease-of-use feature in OWB is the ability to define a Business Area. A Business Area can be thought of a folder that contains some, or all, of your objects. When you create a Business Area, you simply give it a name, and check the names or types of those objects that will be included in it. For the case study there is a Business Area named OLAP_BUSINESS_AREA, and it contains reference to all of the dimensions and facts in the OLAP_TARGET module. You provide the name of the Business Area as one of the parameters of the metadata export, as seen below.

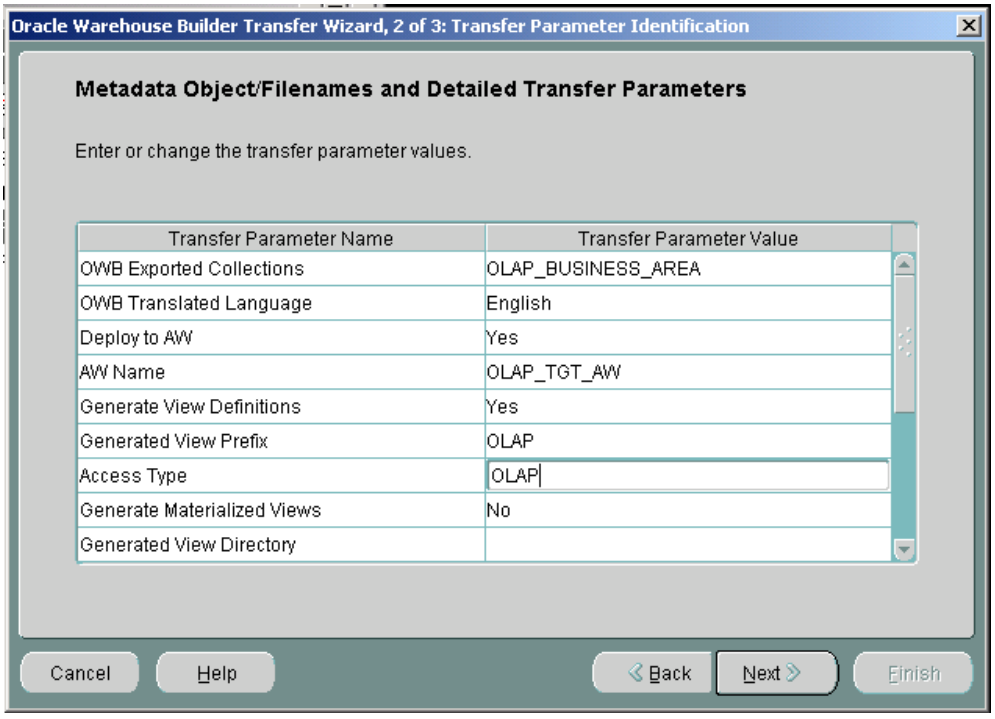


Figure 25: Metadata Transfer wizard

REVIEW THE STRUCTURES

Once you have moved the metadata and deployed your objects you can use tools to review the structures.

Relational structures: Use SQL*Plus or Enterprise Manager to review the relational structures created by OWB.

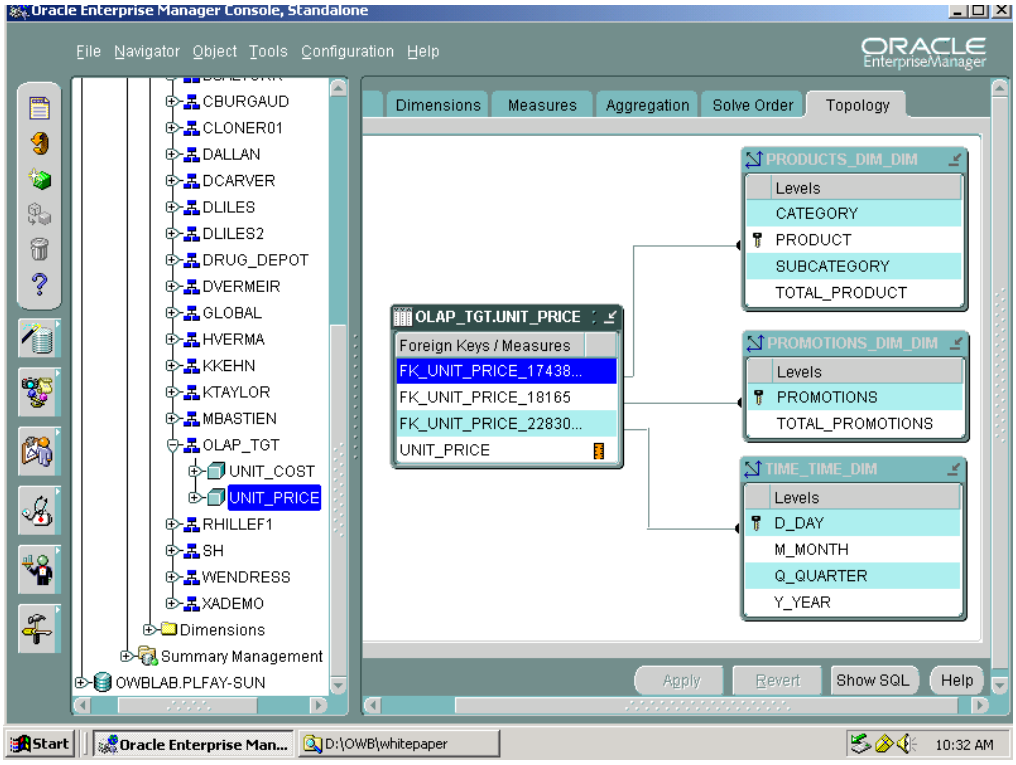


Figure 26: View the UNIT_PRICE Relational Metadata from Enterprise Manager

Multidimensional structures: Use the OLAP Worksheet or the Analytic Workspace Manager to review the multidimensional structures created by OWB. Here we use the AW Manager to view some of the metadata that was created for the AW. The right side of the window displays information about the channels dimension.

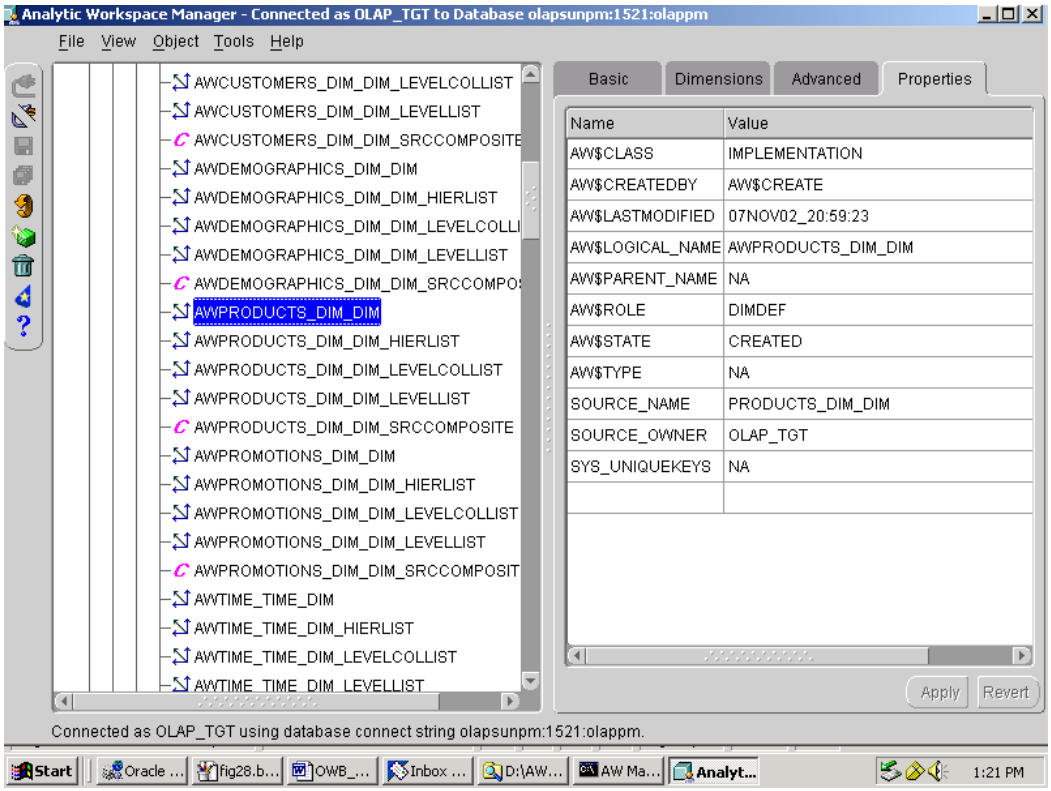


Figure 27: View the PRODUCTS_DIM Metadata in the AW from AW Manager

POPULATE THE WAREHOUSE

After a successful object deployment and metadata export, click on the Mappings tab. From here you have the same View, Deploy, or Save options that you have for the objects. If you choose to save the scripts, you will have the option of registering the script with Oracle Enterprise Manager and Oracle Workflow, so that the eventual running of the scripts can be scheduled. Click on the Run button to populate your target warehouse.

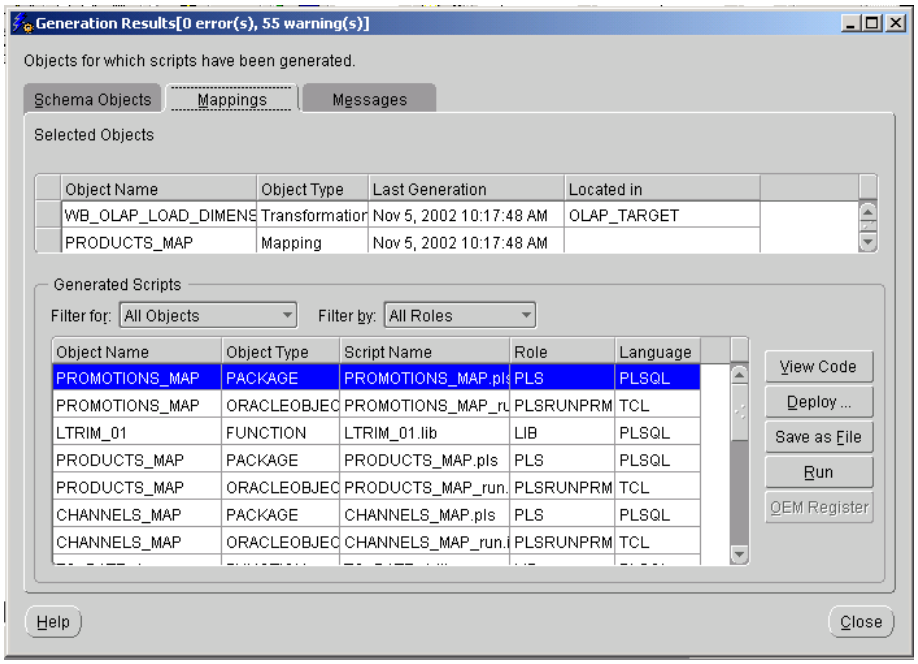


Figure 28: Results window for mapping script generation

VERIFY THE DATA LOAD FROM OWB

OWB's Runtime Audit Viewer can be used to verify the data load . This tool shows the load information and allows you to investigate any errors that may have occurred during the mapping.

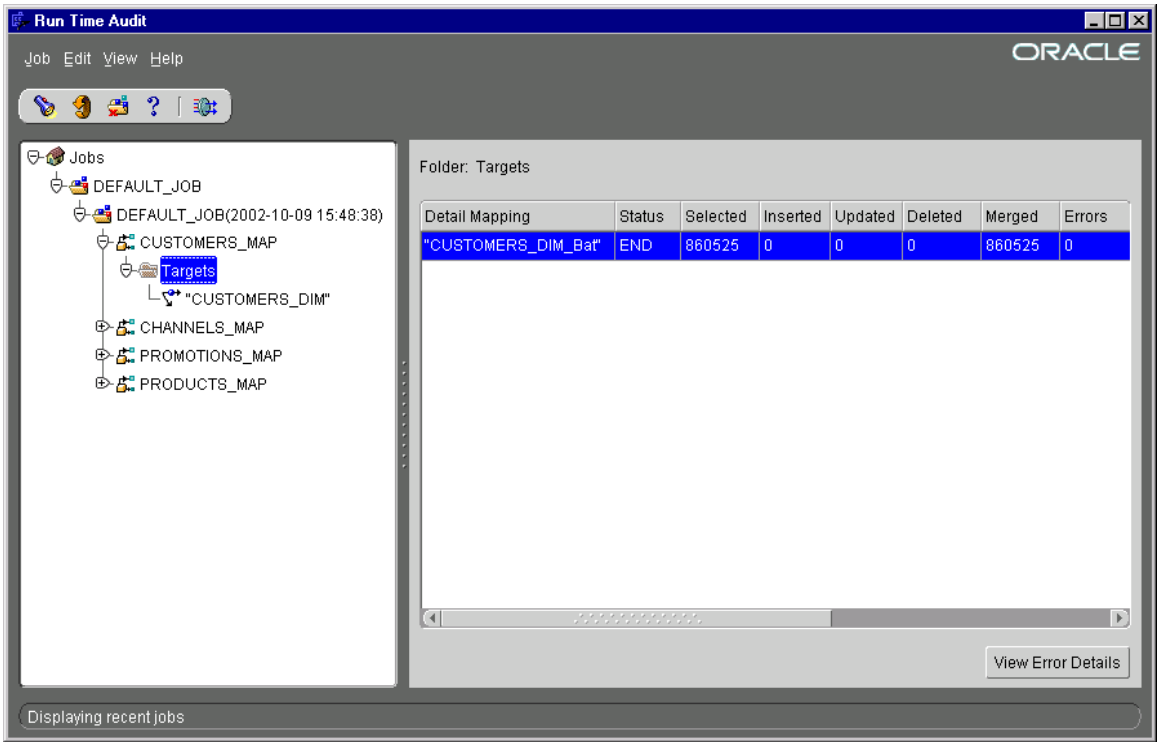


Figure 29: Use OWB's Runtime Audit Viewer to verify the data load

SECTION 7: THE END RESULT - THE OLAP-READY WAREHOUSE

Successful execution of the mapping scripts results in an OLAP-ready data warehouse. The preparation for OLAP was incorporated into the existing Warehouse Builder process flow. The added steps, which are restated below, took minimal effort, and did not interrupt the logical flow of the process.

The first OLAP specific action was to add a post-mapping process to the mappings. This step calls OLAP procedures that load data into multidimensional storage. The second step that was OLAP specific was the metadata export step. Based on the parameter values, this step creates an AW, writes the metadata for that AW to the OLAP Catalog, and creates views over the AW to enable SQL access. This step must be run before the mappings can be run. That's it.

Now that you have designed, deployed and populated your warehouse, you are ready to enable access to the information that your end-users require. The Oracle9i OLAP-ready warehouse supports multiple access methods. All of these methods can manipulate and access data in both relational and multidimensional storage formats.

- SQL access – Oracle tools, such as Discoverer and Reports. Third party tools that generate SQL code are able to access the relational data, as well as the data in an AW.
- Oracle's BI Beans – Delivered with Oracle's JDeveloper tool, the BI Beans provide a Java rapid application development environment.
- OLAP API – The OLAP API provides an OLAP-aware Java API for developing analytic applications and tools.
- OLAP DML – Application developers can use the OLAP DML to directly manipulate the data and processes in an AW.

  The OLAP DML can be used to extend the built-in functionality in an AW. Applications for budgeting, forecasting, or planning might need this extended functionality. The OLAP DML is a 4GL that is rich with analytic content. As well, Oracle OLAP provides multiple UI tools to facilitate the use of the DML.

CHANGE LOG:

Version	Date	Description
1.0	November 8, 2002	Initial version