Oracle® Warehouse Builder

Concepts

11*g* Release 2 (11.2)

E10581-01

July 2009

ORACLE®

Oracle Warehouse Builder Concepts, 11*g* Release 2 (11.2)

E10581-01

Contents

2 Introduction to Oracle Warehouse Builder

3 User Interface Tour

4 Data Access and Movement

5 Data Objects

6 Data Transformation

7 Dependency and Change Management

8 Data Quality Management

9 Deployment and Execution

10 Scripting and Automation

Index

Preface

Oracle Warehouse Builder Concepts provides an architectural and conceptual overview for features and functionality of Oracle Warehouse Builder. This document lays a conceptual foundation for much of the practical information contained in other manuals.

This preface contains these topics:

Audience

Oracle Warehouse Builder Concepts, this guide, is primarily intended for database administrators and application developers who are new to Oracle Warehouse Builder, and who develop and maintain data integration systems. Readers of this document typically perform one or more of the following tasks:

- Plan for a data integration environment
- Configure data integration systems
- Administer a data integration environment
- Perform data integration tasks
- Deploy data integration systems

To use this document, you should be familiar with relational database concepts, distributed database administration, and the operating system under which you already run, or plan to run, an Oracle Warehouse Builder environment.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be

accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents

For more information about Oracle Warehouse Builder, see these Oracle resources:

- *Oracle Warehouse Builder Release Notes*
- *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*
- *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- *Oracle Warehouse Builder Sources and Targets Guide*
- *Oracle Warehouse Builder API and Scripting Reference*

For information about data warehousing, see these Oracle resources:

- *Oracle Database 2 Day + Data Warehousing Guide*
- *Oracle Database Data Warehousing Guide*

For information about licensing options, see:

- *Oracle Database Licensing Information*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

What's New in Oracle Warehouse Builder

This section summarizes the new high-value features in Oracle Warehouse Builder (OWB) for this release.

This section contains the following topics:

New Feature Highlights for Oracle Warehouse Builder 11*g* Release 2 (11.2)

While the new features of OWB for this release cover a number of different areas, significant changes for new and existing customers are:

- ETL support for non-Oracle databases, within and tightly integrated with the familiar flow-based ETL design paradigm
- SOA integration for OWB data integration and data quality functionality
- Extensive user interface redesign for enhanced usability and developer productivity

> **Note:** Go to the detailed sections under "Complete New Feature List for Oracle Warehouse Builder 11g Release 2 (11.2)" for links to the relevant documentation.
>
> Numerous smaller changes and improvements have been made throughout the product and, therefore, this list is not intended to be exhaustive.

New Features by Group for Oracle Warehouse Builder 11*g* Release 2 (11.2)

The major new features in OWB for this release can be grouped into the following categories:

- Administrator Usability Enhancements
- ETL Mapping Enhancements
- Process Flow Enhancements

Native Support for Heterogeneous Databases

OWB now provides extensive built-in support for non-Oracle databases. JDBC connectivity is added alongside previous support for ODBC and database gateways, and OWB now supports in-database ELT operations on non-Oracle databases. Other enhancements improve access to data from non-Oracle sources such as mainframe and flat file data.

Features in this area include:

- Code Template Mappings and JDBC Connectivity Support
- Enhanced Support for Flat File Imports
- Extensible Platform Framework
- Java-Based (J2EE) Control Center Agent
- Metadata Import from COBOL Copybooks

SOA Integration Enhancements for ETL and Data Quality

The ETL and data quality functionality in OWB can now be integrated into SOA-style architectures.

Features in this area include:

- Web Service and SOA Integration Support for ETL and Data Quality
- Java-Based (J2EE) Control Center Agent

Data Warehousing and Business Intelligence Enhancements

The data warehousing-specific support in OWB has improved. These improvements provide smarter dimensional object operators for ETL and support for more storage types for dimensional objects.

Features in this area include:

- Oracle Business Intelligence Enterprise Edition Integration
- Automated Orphan Management Policy for Loading Dimensional Objects
- OLAP Cube-Organized Materialized Views Support

Developer Usability Enhancements

The Design Center user interface in OWB has been extensively redesigned to improve developer productivity and make advanced features more accessible.

Features in this area include:

- Advanced Find Support in Mapping Editor
- Copy and Paste of Operators and Attributes in Mapping Editor
- Current Configuration Drop-down Box in Design Center Toolbar
- Experts Available in Editor Menus

- Expression Editing in Operator Edit Dialog
- Improved User Interface for Managing Locations (for Sources and Targets)
- Grouping and Spotlighting of Objects in Mapping Editor
- Improved Metadata Search and Find for Dependency Management
- New JDeveloper-Style User Interface
- Operator Comments Included in Generated PL/SQL Code
- Organizing Objects with User Folders
- Quick Mapper in Mapping Connection Dialog Box

Administrator Usability Enhancements

OWB administration tasks are simplified and improved by a number of features in this release. Administration has been extended to support new feature areas such as heterogeneous database support and Web services integration.

Features in this area include:

- Multiple Configuration Management Usability Improvements
- Web Service and SOA Integration Support for ETL and Data Quality
- Improved Management of Locations Registered in Multiple Control Centers
- Heterogeneous Audit and Reporting
- Repository Browser Changes
- Simplified Oracle Warehouse Builder Repository Upgrades

ETL Mapping Enhancements

ETL mappings have been enhanced to add new transformation capabilities and to improve the productivity of developers working with flat files and designing and debugging ETL mappings.

Features in this area include:

- Operator Comments Included in Generated PL/SQL Code
- Key Lookup Operator Enhancements
- Mapping Debugger Enhancements
- Enhanced Table Function Support
- Improved Code Generation for Extraction of Peoplesoft Application Data
- Support for Subqueries in Join Operator
- Web Service and SOA Integration Support for ETL and Data Quality

Process Flow Enhancements

OWB process flows have been enhanced to integrate with new activity types, out of the box, and to support integration of OWB ETL and data quality with SOA solutions.

Features that enhance process flows include:

- Support for EJB/Java Activity Type in Process Flows
- Support for OMB*Plus Activity Type In Process Flows

- Web Service and SOA Integration Support for ETL and Data Quality

Complete New Feature List for Oracle Warehouse Builder 11*g* Release 2 (11.2)

This section provides detailed descriptions of all major new features in OWB for this release.

Advanced Find Support in Mapping Editor

The Mapping Editor has been enhanced with advanced find capabilities to make it easier to locate and make updates to operators, groups, and attributes in a mapping diagram, in the Available Objects tab, and in the Selected Objects tab.

This feature enhances extraction, transformation, and loading (ETL) mapping developer productivity, especially on large and complex mappings and, for example, when working with complex data sources with large numbers of tables, views, or columns.

> **See Also:** "Overview of the Mapping Editor" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Oracle Business Intelligence Enterprise Edition Integration

OWB now supports integration with Oracle Business Intelligence Suite Enterprise Edition (OBI EE). This integration includes:

- Derivation of ready-to-use physical, business model and presentation layer metadata for OBI EE from a data warehouse design
- Visualization and maintenance of the derived objects from within OWB
- Deployment of the derived objects in the form of an RPD file that can be loaded into OBI EE.
- Inclusion of the derived objects in OWB data lineage and impact analysis, such that data lineage of objects in OBI EE reports can be traced down to the individual column level.

> **See Also:** *Oracle Database Licensing Information* and these topics in *Oracle Warehouse Builder Sources and Targets Guide*:
>
> - "Importing Warehouse Builder Data into Business Intelligence Applications"
> - "Creating Business Definitions for OBI EE"

OLAP Cube-Organized Materialized Views Support

OWB now supports OLAP cube storage in cube-organized materialized views. This brings the performance advantages of such storage to users of OWB data warehouse design.

> **See Also:** "Creating Cubes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Code Template Mappings and JDBC Connectivity Support

The new Code Template-based mapping framework enables implementation of data integration techniques and patterns for integration of content from non-Oracle databases with maximum performance and flexibility.

JDBC connectivity supports a wide variety of sources. Additionally, Oracle-supplied or user-developed Code Templates can use other native data integration techniques, such as bulk unloads and loads, for maximum performance on any platform.

Code Template mappings bring heterogeneous data integration support to the familiar flow-based data integration mapping paradigm that leverages existing developer skills with OWB. Code Template mappings that load Oracle targets still support the full range of transformation capabilities available in other OWB mappings.

See Also:

- "Creating Code Template Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Using Code Templates to Load and Transfer Data" in *Oracle Warehouse Builder Sources and Targets Guide*

Copy and Paste of Operators and Attributes in Mapping Editor

In the mapping editor, users can now copy and paste operators within a mapping or across mappings, including attribute settings.

This enhancement saves time and reduces errors in the development of complex ETL mappings that reuse common or similar elements.

See Also: "Copying Operators Across Mappings and Pluggable Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Current Configuration Drop-down Box in Design Center Toolbar

In the Design Center, there is now a drop-down box that displays the active configuration of the user. This feature improves usability of the multi-configuration feature.

See Also: "Activating Configurations" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

Enhanced Support for Flat File Imports

There are numerous improvements to support for importing flat files, including a simplified Flat File Sample Wizard, support for multi-character and hexadecimal format delimiters and enclosures, simplified support for fixed format fields, and support for bulk flat file loads into heterogeneous targets.

Flat files are frequently used for simple and high-performance data movement in ETL applications. These changes improve ETL developer productivity and provide more flexible handling of more types of flat files in more scenarios.

See Also: "Flat Files as Targets" and "Importing Definitions from Flat Files Using Sampling" in *Oracle Warehouse Builder Sources and Targets Guide*

Enhanced Table Function Support

OWB now has improved support for table functions, including importing metadata for existing table functions, a wizard for creating table functions from within OWB, and better support for table functions in mappings.

Improved support simplifies using table functions for much more flexible and powerful transformations, such as user-defined aggregations, data mining sampling operators, and so on.

> **See Also:** "Table Function Operator" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Experts Available in Editor Menus

It is now possible to add OWB experts to the mapping editor menu.

This feature makes it possible to enhance and extend the functionality of the mapping editor, improving developer productivity.

> **See Also:** "Overview of the Mapping Editor" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Expression Editing in Operator Edit Dialog

Expressions associated with operator attributes can now be entered directly into an Operator Edit Dialog or Expression Editor, rather than requiring that these expressions be entered into a property in the Property Inspector.

Developers can finish more of their work in one place when creating operators in ETL mappings, thus improving their productivity.

> **See Also:** "About Expressions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Extensible Platform Framework

Platform extensibility enables users to define new platforms, represent the native data types for those platforms, and create ETL mappings that manipulate that data according to the requirements of the platform.

This feature is part of the overall improved support for heterogeneous databases in this release.

> **See Also:**
>
> - "Using Functions In Non-Oracle Platforms" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
> - "Using Code Templates to Load and Transfer Data" in *Oracle Warehouse Builder Sources and Targets Guide*

Automated Orphan Management Policy for Loading Dimensional Objects

Orphan management policy for dimensions or cubes refers to the process of handling source rows that do not meet the requirements necessary to form a valid dimension or cube record.

OWB now supports the following orphan management policies:

- Assign a default parent

- Reject orphan rows
- No maintenance

Automated orphan management policies improve ETL developer and administrator productivity by addressing an important cause of cube and dimension load failures.

> **See Also:** "Performing ETL Using Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Grouping and Spotlighting of Objects in Mapping Editor

You can now temporarily or permanently group objects in the Mapping Editor so that they are collapsed to a single icon. This hides complexity in mappings. Users can also spotlight a single operator, which temporarily hides all objects in the mapping except for those objects that connect directly to the operator.

These features improve productivity for developers working with complex mappings with large numbers of operators.

> **See Also:** "Grouping Operators in Mappings and Pluggable Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Heterogeneous Audit and Reporting

Auditing and reporting on runtime jobs have been enhanced to show execution of all jobs required to support heterogeneous connectivity.

Users receive a unified view of all OWB job executions on both Oracle and non-Oracle platforms.

> **See Also:** "Monitoring Quality with Data Auditors and Data Rules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Improved Management of Locations Registered in Multiple Control Centers

The user interface for managing the registration of locations in control centers has been reworked to improve usability, especially when working with locations registered in multiple control centers.

This change improves productivity of OWB administrators responsible for managing locations across control centers.

> **See Also:** "Locations Registered in Multiple Control Centers" in *Oracle Warehouse Builder Sources and Targets Guide*

Improved Metadata Search and Find for Dependency Management

The Dependency Manager, which is used to browse data lineage and impact analysis information, now includes advanced metadata searching capabilities.

Users can now more easily locate specific objects in large and complex dependency graphs. This improves productivity by making it easier to find specific objects and their lineage, and discover impacts from design changes.

> **See Also:** "Managing Metadata Dependencies" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Improved User Interface for Managing Locations (for Sources and Targets)

The user interface for managing OWB locations has been reworked to improve usability and support access to non-Oracle data sources using newly supported connectivity methods.

These changes improve OWB administrator and developer productivity in heterogeneous and Oracle-only environments.

> **See Also:** "Designing Source and Target Schemas" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Java-Based (J2EE) Control Center Agent

The Control Center Agent provides a Java-based runtime environment that can be installed on Oracle and non-Oracle database hosts. Heterogeneous ETL mappings and Web services-related code are deployed to the Control Center Agent, and runtime audit metadata is accessible within OWB.

The Control Center Agent provides fundamental infrastructure for the heterogeneous, Code Template-based mapping support and Web services-related features of OWB for this release.

> **See Also:** "Enterprise Java Bean" and "Control Center Reports" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Key Lookup Operator Enhancements

Extensive changes have been made to the key lookup operator:

- More efficient use of screen real estate.
- Support for non-equality lookups.
- Dynamic lookups, where the lookup table may be modified during the mapping execution.

These changes make the lookup operator more powerful in many situations, including improving Type 2 slowly changing dimension support.

> **See Also:** "Lookup Operator" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Mapping Debugger Enhancements

There are numerous enhancements to the OWB Mapping Editor, including:

- Improved support for watch points and enabling and disabling of individual break points.
- Support for user-defined type columns.
- Enhanced support for numerous existing operators, such as `VARRAY`, `EXPAND`, and `CONSTRUCT`.
- Support for key lookup and table function operators.
- Support for correlated joins.
- Improved cleanup of debugger-specific objects.

These enhancements will improve productivity for ETL mapping developers, especially when working with complex mappings where the mapping debugger adds the most value.

> **See Also:** "Debugging Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Metadata Import from COBOL Copybooks

In this release, Oracle Warehouse Builder provides the ability to import metadata from COBOL Copybook definitions.

This improves developer productivity, by simplifying working with complex flat file data structures extracted from mainframe sources.

> **See Also:** "Importing Metadata Definitions from COBOL Copybooks" under "Using Flat Files as Sources or Targets" in *Oracle Warehouse Builder Sources and Targets Guide*

Multiple Configuration Management Usability Improvements

The OWB user interface for viewing and managing multiple configurations has been redesigned to simplify and clarify previously complex tasks, including:

- Editing the configuration values for an object across all configurations
- Configuring groups of objects at the same time
- Using configuration templates to set default configuration values for different object types
- Copying and pasting of configuration attribute values
- Side-by-side editing of attribute values for multiple configurations for an object

These improvements enable users to take full advantage of the flexibility provided by multiple configurations.

> **See Also:** "Configuring Objects" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

New JDeveloper-Style User Interface

The Oracle Warehouse Builder Design Center user interface has been updated to use the Fusion Client Platform, the same core Integrated Development Environment (IDE) platform as Oracle JDeveloper and Oracle SQL Developer.

The advantages of this user interface framework include:

- More efficient and flexible use of screen real estate.
- Support for opening multiple editors of the same type, for example, editing several ETL mappings at once in different windows.
- More consistent behavior across different parts of the OWB user interface.

This change brings the Design Center in Oracle Warehouse Builder in line with other development tools from Oracle.

> **See Also:** Chapter 3, "User Interface Tour" in this guide

Operator Comments Included in Generated PL/SQL Code

PL/SQL code generated for OWB ETL mappings now includes detailed comments to help developers associate specific operators in a mapping with sections of the generated code.

Developers can more easily troubleshoot issues with OWB-generated code that can only be detected when the code is deployed. This additional information enhances developer productivity.

> **See Also:** "Creating PL/SQL Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Organizing Objects with User Folders

Users can create hierarchically nested folders to logically group related objects. Folders can be created within Oracle and non-Oracle database modules, non-Oracle database modules, application modules. User folders can be nested as necessary to organize objects further.

Folders can be used to group related objects. Any object in one of the supported module types, such as a table or a mapping, can be moved into a folder.

For example, if a single database module contained tables, views, and ETL mappings for product and customer data, folders "Product" and "Customer" could be created, and the objects related to each category moved into the separate folders.

User folders can also be created to contain pluggable mappings. This allows organization of related pluggable mappings into groups.

User-created folders improve ETL developer productivity on complex projects, by making it easier to logically group and manage large numbers of objects.

> **See Also:**
>
> - "Creating User Folders" in *Oracle Warehouse Builder Sources and Targets Guide*
> - "Configuring Target Modules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Quick Mapper in Mapping Connection Dialog Box

In this release, OWB introduces the Mapping Connection dialog box, a spreadsheet-like "quick mapper" for connecting operators in a mapping. This functionality replaces the Auto Mapping dialog box in earlier releases.

This improvement saves developer time and reduces errors when working with operators with a large number of inputs or outputs.

> **See Also:** "Connecting Operators, Groups, and Attributes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Repository Browser Changes

The Repository Browser has been updated to support foldering, to expose the new types of metadata associated with the release 11.2 feature set, and to support OC4J 10.3.3.

These changes improve manageability for Oracle Warehouse Builder.

See Also: "Common Repository Browser Tasks" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Chunking for Parallelizing Large Table Updates

Chunking in Warehouse Builder automates the use of a "divide and conquer" approach to parallelize the processing of large updates. Users enable chunking for a mapping and define chunking criteria to partition the updates. OWB generates PL/SQL code for the mapping, and at execution time, updates are divided according to chunking criteria, a pool of threads is allocated, and the chunks are processed in parallel.

The benefits of applying chunking include:

- Chunking provides the only method of automatically parallelizing PL/SQL code in Warehouse Builder.
- Chunking avoids the need for large rollback segments. Set-based SQL statements for large updates require large rollback segments, because a single set-based statement does not perform intermediate commits.
- Large updates can be performed incrementally, and if interrupted, chunks already processed do not have to be processed again. Without chunking, if a large update terminates for some reason, all processing must be repeated.

See Also: The section on chunking data under "Using Oracle Source and Target Operators" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Simplified Oracle Warehouse Builder Repository Upgrades

The Repository Upgrade automatically upgrades an Oracle Warehouse Builder (OWB) repository to the current release with less user intervention.

This feature simplifies the task of upgrading from one release to the next.

See Also: "Upgrading an OWB 11g R1 Repository" and "Upgrading an OWB 10g R2 Repository" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

Support for EJB/Java Activity Type in Process Flows

A new process flow activity supports calling an EJB or Java program from within a process flow.

Customers benefit from being able to incorporate existing or new logic implemented in Java into their data integration processes.

See Also: "Enterprise Java Bean" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Improved Code Generation for Extraction of Peoplesoft Application Data

Oracle Warehouse Builder can now generate SQL*Plus code to extract data from database schemas supporting the deprecated LONG data type, such as occurs in PeopleSoft application data sources.

Support for LONG data types used in PeopleSoft data enables OWB users to integrate more effectively with PeopleSoft data or any other data source that uses the `LONG` data type.

See Also:

- "Creating Relational Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Importing Metadata from PeopleSoft Applications" in *Oracle Warehouse Builder Sources and Targets Guide*

Support for OMB*Plus Activity Type In Process Flows

Process flows now support an activity type for running an OMB*Plus script.

New process flow activity types increase the breadth of user-defined activities that can be incorporated into process flows and thus orchestrated and managed as part of your overall data integration process.

See Also: "OMB*Plus" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Support for Subqueries in Join Operator

The `JOIN` operator in OWB now supports several new behaviors related to the use of subqueries in joins:

- Specifying subqueries using `EXISTS`, `NOT EXISTS`, `IN`, and `NOT IN`.
- Specifying outer joins using the input role instead of the + (plus) sign.
- Generating ANSI SQL syntax for all join types instead of only outer joins.

More flexible handling for `JOIN` operations improves developer productivity and makes possible more flexible data transformations.

See Also:

- "Joiner Operator" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Creating and Editing Joins" in *Oracle Warehouse Builder Sources and Targets Guide*

Web Service and SOA Integration Support for ETL and Data Quality

Warehouse Builder ETL and data quality mapping, process flows, transformations, and data auditors can be published as Web services. OWB can also consume Web services in process flows.

This feature provides point-and-click integration of the ETL and data quality functionality of OWB into SOA-based designs, and facilitates integration with SOA-based process orchestration technologies such as Oracle BPEL Process Manager. (SOA stands for Service-Oriented Architecture).

See Also: "Creating and Consuming Web Services in Warehouse Builder" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

2

Introduction to Oracle Warehouse Builder

This section provides an introduction to Oracle Warehouse Builder (OWB), gets you started using OWB, and discusses the architectural components and objects that you create.

This section contains these topics:

Overview of Oracle Warehouse Builder and Its Benefits

Oracle Warehouse Builder (OWB) is a full-featured data integration, data warehousing, data quality and metadata management solution designed for the Oracle database. OWB is an integral part of Oracle Database 11*g* Release 2 (11.2) and is installed as part of every database installation (other than Oracle Database XE).

The major feature areas of OWB include:

- Data modeling
- Extraction, Transformation, and Load (ETL)
- Data profiling and data quality
- Metadata management
- Business-level integration of ERP application data
- Integration with Oracle business intelligence tools for reporting purposes
- Advanced data lineage and impact analysis

OWB is also an extensible data integration and data quality solutions platform. OWB can be extended to manage metadata specific to any application, and can integrate with new data source and target types, and implement support for new data access mechanisms and platforms, enforce your organization's best practices, and foster the reuse of components across solutions.

> **See Also:** *Oracle Database Licensing Information* for complete information about options for Oracle Warehouse Builder

Use Cases for Oracle Warehouse Builder

OWB can be used in a wide range of scenarios, centered on Oracle Database, and adds value as a solution for data integration, data movement, and data quality. The data systems you create with OWB are driven by rich metadata about sources and targets, and tight integration with, and awareness of, core features in Oracle Database. The ETL and data quality features provided by OWB add value in each of the use cases described in this section.

The most common use cases include:

- Business Intelligence and Data Warehousing
- Master Data Management
- Data Migration, Conversion, and Modernization
- Data Profiling and Quality Management

Business Intelligence and Data Warehousing

OWB can be used in the design of relational objects for your operational data store, and dimensional objects for the data warehouse performance layer. You can implement ETL processes for loading warehouses, including smart operators that simplify loading dimensional objects, even for complex loading processes required for slowly changing dimensions. OWB can implement business intelligence applications and data marts.

OWB can also be used to profile data sources and to develop or discover data rules. Data rules can be used to measure data quality, monitor, and enforce quality requirements during loading, or as an out-of-band process. Data cleansing logic can be incorporated into the warehouse loading process.

Master Data Management

OWB application adapters (or connectors) enable access to data stores representing critical business entities such as customers and products at a logical, rather than physical, level. This simplifies the design of data movement, data quality and data cleansing, and enrichment processes.

OWB data quality features can be used to discover, audit, and enforce the contents of your master data stores and their compliance with your data rules. Automated data cleansing and enrichment processes are easy to implement.

Data Migration, Conversion, and Modernization

OWB can be used to design the target for any migration or conversion process and can implement data movement processes. Data quality features offer high value in such scenarios as well. Data profiling of the source systems can reveal data quality issues before they are introduced into the new system. OWB can be used to profile source data, design the target system, and to implement and orchestrate complex data movement, transformation and cleansing processes without requiring custom code.

Data Profiling and Quality Management

Once you connect to your data sources in OWB (including Oracle databases, sources accessed through gateways, and flat file sources) you can apply full-featured data profiling to generate statistics about data quality, and to discover complex patterns, foreign key relationships, and functional dependencies. You can then design complex data rules and create data auditors to monitor compliance with those rules in any source or target system in your landscape, regardless of whether those sources are loaded using OWB or other ETL tools.

For customers who have selected solutions other than OWB for data profiling and data quality, these can be applied independently of the OWB ETL and design features.

> **Note:** Depending on how you utilize OWB, you may require licenses for additional database options and technologies. Refer to *Oracle Database Licensing Information* for complete details about OWB options.

Quick Start for Using Oracle Warehouse Builder

Once Oracle Database is installed, you do not need to take additional actions other than to unlock the `OWBSYS` and `OWBSYS_AUDIT` accounts, and run the Repository Assistant. This section provides

> **See Also:** These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*:
>
> - "To unlock OWBSYS and OWBSYS_AUDIT accounts"
> - "Overview of Installation and Configuration Architecture"

Oracle recommends that you start with the following steps to learn about using OWB:

1. Before You Begin
2. Configure a Project in the OWB Design Center
3. Import the Source Metadata
4. Profile Data and Ensure Data Quality
5. Design the Target Schema
6. Design ETL Logic
7. Deploy the Design and Execute the Data Integration Solution
8. Monitor Quality and Report on the Data System

The first time you start OWB, the Start Page is displayed with links to get you started using the product.

> **Note:** Standalone software for OWB is available with Oracle Database. Use the OWB standalone software to host the OWB repository on an earlier release of Oracle Database. Also, use the standalone software to install OWB on client computers. See "Working with the OWB Standalone Install Package" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

Before You Begin

Before you can use any of the OWB client components, first ensure you have access to an OWB workspace.

To begin using OWB:

1. Install the OWB software and create the necessary workspaces as described in "Installing Oracle Warehouse Builder on the Server" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*.

If an administrator has previously completed the installation, contact that person for the required connection information.

2. Start the Design Center.

   On a Windows platform, from the **Start** menu, select **Programs.** Select the Oracle home in which OWB is installed, then **OWB**, and then **Design Center.**

   On a Linux platform, run `owbclient.sh` located in the `owb/bin/unix` directory in the Oracle home for OWB.

Use the Projects Navigator to manage design objects for a given workspace. The design objects are organized under a **project**, which provide a means for structuring the objects for security and reusability. Each project contains nodes for each type of design object that you can create or import.

Use the Connections Navigator to establish connections between the OWB workspace and databases, data files, and applications.

Use the Globals Navigator to manage objects that are common to all projects in a workspace and to administer security.

Note: The Security node is visible to users who have an administrator role.

See Also: "Managing Security" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

Configure a Project in the OWB Design Center

In this procedure, you configure your project and access source and target data.

To configure a project in the Design Center:

1. In the Projects Navigator, identify the project to be used.
   - If you are satisfied with the single default project, MY_PROJECT, continue with the next step.
   - Alternatively, you can rename MY_PROJECT or define more projects. Each project you define is organized in the same fashion with nodes for databases, files, applications, and so on. See the procedure "To create a project" on page 3-3 For a different organization, consider creating optional collections as described in "Collections" on page 3-4.
2. Create locations in order to connect to source and target data objects.
   - To create a **location**, right-click the appropriate node and select **New**. Fill in the requested connection information and select **Test Connection**. In this step, you merely establish connections to sources and targets. You do not move data or metadata until subsequent steps.
   - In the Connections Navigator, establish these connections by defining locations. Expand the Location node and the nodes within it to gain a general understanding of the types of source and targets you can access from OWB.

   For more information about locations see "Locations Navigator" on page 3-12.
3. Identify the **target schema**.

   Although you can use a flat file as a target, the most common and recommended scenario is to use the Oracle Database as the target schema.

To define the target schema, begin by creating a module. **Modules** are grouping mechanisms in the Projects Navigator that correspond to locations in the Connections Navigator. The Oracle target module is the first of several modules you create in OWB.

In the Projects Navigator, expand the Databases node. Right-click **Oracle** and select **New.** The Create Module Wizard appears. Set the module type to Warehouse Target and specify whether the module will be used in development, quality assurance, or production. This module status is purely descriptive and has no bearing on subsequent steps you take.

When you complete the wizard, the target module displays with nodes for mappings, transformations, tables, cubes and the various other types of objects you utilize to design the target warehouse.

4. Create a separate Oracle module for the data sources. (Optional)

   At your discretion, you can either create another Oracle module to contain Oracle source data or proceed to the next step.

5. Identify the execution environment.

   Under the Connections Navigator, notice the Control Centers node. A Control Center is an Oracle Database schema that manages the execution of the ETL jobs you design in the Design Center in subsequent steps.

   During installation, OWB creates the `DEFAULT_CONTROL_CENTER` schema on the same database as the workspace.

   If you choose to utilize the default execution environment, continue to the next step. Alternatively, you can define new control centers at any time. For more information and procedures, see "Deploying to Target Schemas and Executing ETL Logic" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

6. Prepare development, test, and production environments. (Optional)

   Thus far, these instructions describe the creation of a single project corresponding to a single execution environment. You can, however, reuse the logical design of this project in different physical environments such as testing or production environments.

   Deploy a single data system to several different host systems or to various environments, by creating additional configurations. See "Managing Configurations" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*.

7. Adjust the client preference settings as desired or accept the default preference settings and proceed to the next step.

   From the main menu in the Design Center, select **Tools** and then **Preferences.**

   As a new user, you may be interested in setting the Environment Preferences, the locale under Appearance Preferences, and the naming mode under Naming Preferences. For information on all the preferences, see "About OWB Preferences" on page A-1.

Import the Source Metadata

In this section, you create modules for each type of design object into which you intend to import metadata.

1. In the Projects Navigator, select a node such as **Files**.

- For the selected node, determine the locations from which you intend to ultimately extract data.
- Then create a module for each relevant location by right-clicking on the node and select **New.**

2. Import metadata from the various data sources: right-click the module and select **Import** to extract metadata from the associated location. OWB displays a wizard to guide you through the process of importing data.

   For an example and additional information on importing data objects, see "Importing Warehouse Builder Data into Business Intelligence Applications" in *Oracle Warehouse Builder Sources and Targets Guide*.

3. For the metadata you imported, profile its corresponding data. (Optional)

   The next step uses the Data Profiling Option to ensure data quality as described in "Overview of Data Profiling" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

Profile Data and Ensure Data Quality

Data can only be transformed into actionable information when you are confident of its reliability. Before you load data into your target system, you must first understand the structure and the meaning of your data, and then assess the quality.

Consider using the Data Profiling Option to better understand the quality of your source data. With the Data Profiling Option, you can correct the source data and establish a means to detect and correct errors that may arise in the loading of transformed data.

See Also:

- *Oracle Database Licensing Information* for information about licensing the Data Profiling Option
- "Performing Data Profiling" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Design the Target Schema

In this section, you create and design the data objects for the Oracle target module. In previous steps, you may have already imported existing target objects.

To design the target schema:

1. To create data objects, you can either start the appropriate wizard or use the Data Object Editor. To use a wizard, right-click the node for the desired object and select **New.** After using a wizard, you may want to modify the object in the editor. In that case, right-click the object and select **Open Editor.**

   For additional information, see "Designing Target Schemas" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

2. As you design objects, be sure to frequently validate the design objects.
   - You can validate objects as you create them, or validate a group of objects together. In the Projects Navigator, select one or more objects or modules, then click the Validate icon.
   - Examine the messages in the Validation Results window. Correct any errors and try validating again.

- To redisplay the most recent validation results at a later time, select **Validation Messages** from the View menu.

For additional information, see "Validating Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

3. Configure the data objects.
   - To configure a data object, select the data object in the Projects Navigator and click the Configure icon. Or right-click the data object in the Projects Navigator and select **Configure**.
   - Configuring data objects specifies the physical properties of the object. You must not generate and deploy data objects without specifying the physical property values.
   - When you create data objects, OWB assigns default configuration property values based on the type of object. In most cases, these default values are appropriate. You can edit and modify the configuration property values of objects according to your requirement. For example, you configure a table to specify the name of the tablespace in which it is created.
4. When satisfied with the design of the target objects, generate the code.
   - In the Projects Navigator, select one or more objects or modules, then click the **Generate** icon. Examine the messages in the Generation Results window. To redisplay the most recent generation results at a later time, select **Generated Scripts** from the View menu.
   - Alternatively, in the Data Object Editor, you can generate code for a single object by clicking the **Generate** icon.
   - You can save the generated script as a file and optionally deploy it outside OWB.

   Generation produces a DDL or PL/SQL script to be used in subsequent steps to create the data objects in the target schema. For more information about generation, see "Generating Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

Design ETL Logic

This procedure describes how to design mappings that define the flow of data from a source to target objects.

To design ETL logic:

1. In the Projects Navigator, expand the Oracle target module, right-click the Mappings node and select **New.**
   - The Mapping Editor enables you to define the flow of data visually. You can drag-and-drop operators onto the canvas, and draw lines that connect the operators. Operators represent both data objects and functions such as filtering, aggregating, and so on.

   See detailed procedures in "Defining Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*, concluding with generating the code for the mapping.
2. Manage dependencies between mappings. See "Designing Process Flows" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

Deploy the Design and Execute the Data Integration Solution

Deployment is the process of copying the relevant metadata and code you generated in the Design Center to a target schema. This procedure is necessary to enable the target schema to execute ETL logic such as mappings.

To deploy and execute the generated code:

1. Deploy objects from either the Design Center or Control Center Manager.

   In this step, you define the objects in the target schema. You need do this only once.

   The simplest approach is to deploy directly from the Design Center by selecting an object and clicking the Deploy icon. In this case, OWB deploys the objects with the default deployment settings.

   Alternatively, if you want more control and feedback on how OWB deploys objects, from the Design Center menu select **Tools**, then **Control Center Manager**.

   Whether you deploy objects from the Design Center or the Control Center Manager, be sure to deploy all associated objects. For example, when deploying a mapping, also deploy the target data objects such as tables that you defined and any associated process flows or other mappings.

   For more information, see "Deploying to Target Schemas and Executing ETL Logic" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

2. Execute the ETL logic to populate the target warehouse.

   In this step, you move data for the first time. Repeat this step each time you want to refresh the target with new data.

   You have two options for executing the ETL logic in mappings and process flows. You can create and deploy a schedule as described in "Defining Schedules", or you can execute jobs manually as described in "Starting ETL Jobs" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

Monitor Quality and Report on the Data System

It is essential to ensure the quality of data entering your data warehouse over time. Data auditors enable you to monitor the quality of incoming data by validating incoming data against a set of data rules and determining if the data confirms to the business rules defined for your data warehouse.

> **See Also:** "Monitoring Quality with Data Auditors and Data Rules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Although the Control Center Manager displays histories for both deployment and execution, the Repository Browser is the preferred interface for monitoring and reporting on OWB operations.

> **See Also:** "About the Repository Browser" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Oracle Warehouse Builder Architecture

Oracle Database provides Oracle Warehouse Builder (OWB) as part of the standard software when the database is installed. OWB is an integral part of Oracle Database. OWB runs on all versions (Standard Edition, Standard Edition One, Enterprise Edition) and typically all platforms that Oracle Database is certified on and ported to.

The basic OWB architectural components on the server side are:

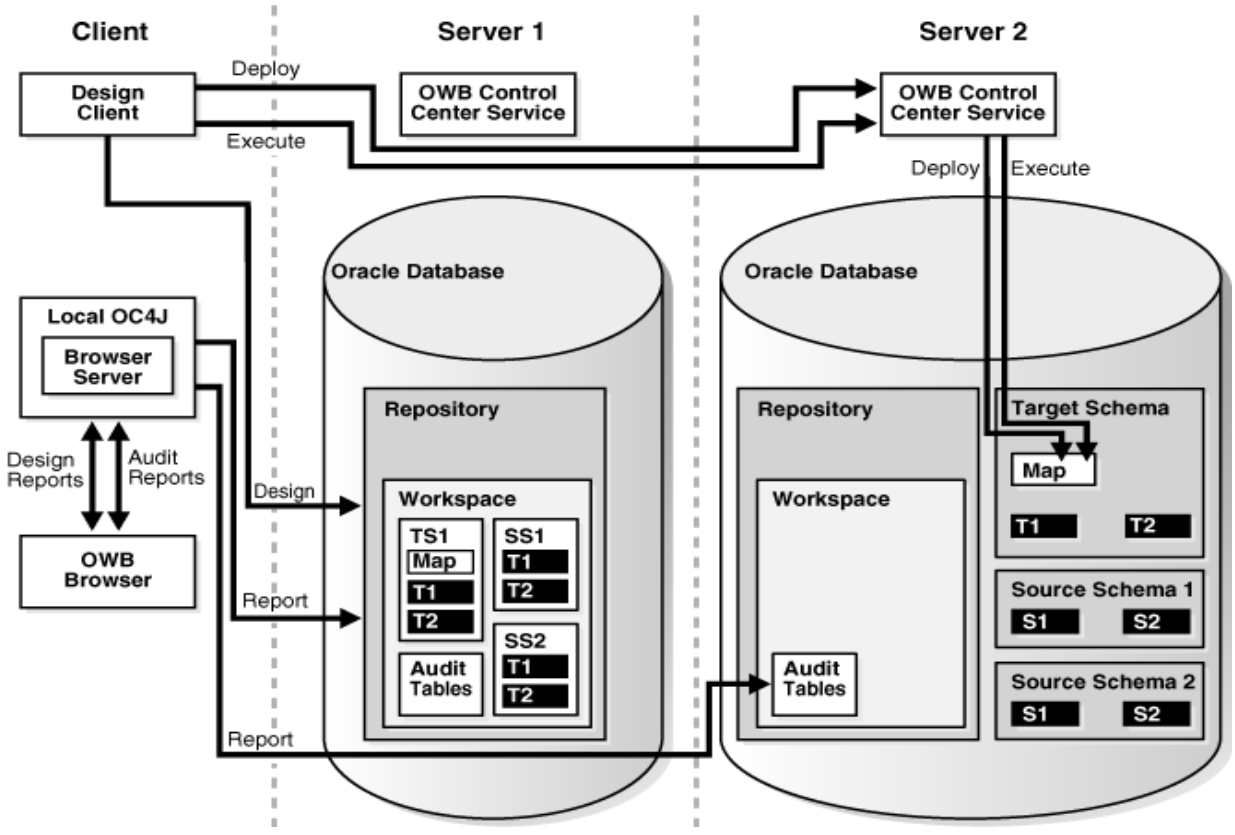
- The OWB Repository
- Workspaces
- Control Center Service
- Control Center Agent (J2EE Runtime)
- Target Schemas

The main OWB components on the client or desktop side, which are discussed in Chapter 3, "User Interface Tour" on page 3-1, are:

- Control Center Manager
- Design Center
- Repository Browser

Figure 2–1 illustrates the components that comprise OWB and where they reside and run on clients and servers.

Figure 2–1 OWB Components



This diagram shows the OWB components. It is divided into two main sections: Client and Server. The Client, shown to the left of the Server, consists of the Design Center box, which contains the Control Center Manager, and the Repository Browser, represented as a box. The Server part consists from left to right: the Control Center service and Oracle Database. The Oracle Database contains: the repository, which contains the language settings and Workspaces that contain the design metadata and the Control Center data. To the right of this box is the target schema, which contains

the generated code, cubes, dimensions, tables, views, mappings, and packages to execute the ETL process.

****

See Also: "Overview of Installation and Configuration Architecture" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux* for diagrams of additional configurations

The OWB Repository

A major feature of the architecture in OWB is the single, unified **OWB Repository** for the database instance, which is pre-seeded with a schema and database objects. The runtime environment and the design environment reside in this single repository. The repository schema, named OWBSYS, gets created when you install Oracle Database. Once the database is installed, you do not need to perform additional actions, other than unlocking the OWBSYS and OWBSYS_AUDIT accounts.

Note:

- OWB release 11*g* stores all repository objects in the OWBSYS schema, which is created as part of every Oracle release 11*g* database. The OWBSYS database user is also registered as an OWB user. Administrators and developers generally will register other database users and assign them required privileges, rather than using the OWBSYS account directly.
- You can create multiple repositories if you prefer to separate the runtime and design environments; however, this is not recommended.

See Also: These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*:

- "Installing Oracle Warehouse Builder on the Server"
- The procedure "To unlock OWBSYS and OWBSYS_AUDIT accounts"
- "Cleaning an Oracle Warehouse Builder Repository" about purging an existing repository
- "Creating an OWBSYS Schema with Repository Objects" about refreshing a repository

Workspaces

To start using OWB, you create at least one, new **workspace**. Users access their respective workspaces, instead of the repository as a whole. Thus, if you are the OWB administrator, instead of granting users access to a repository, you grant them access to one or more workspaces. Because all workspaces are stored in a single repository schema, creating workspaces is simplified.

In defining the repository, an administrator creates one or more workspaces, with each workspace corresponding to a set of users working on related projects. For example, a common practice is to create separate workspaces for Development, Testing, and Production. This practice provides team focus in addition to security. Users such as developers can have access to the Development and Testing workspaces, and can be

restricted from the Production workspace. Later in the implementation cycle, the Repository Assistant in OWB can be used to manage existing workspaces or to create new ones.

See Also:

- "Creating the First Workspace in the Repository" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*
- "Logging in to a Workspace" under "Opening the Repository Browser" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Workspaces" on page 3-1

Control Center Service

Each workspace has a default Control Center that points to itself, and it is started and stopped with its corresponding **Control Center Service**. A Control Center stores detailed information about every deployment and execution, which you can access either by object or by job.

You can use the default Control Center to deploy to the local system, or you can create additional Control Centers for deploying to different systems as needed. Only one Control Center is active at any given time, and this is the Control Center associated with the current active configuration.

See Also: These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*:

- "Setting up the Control Center Service"
- "Starting and Stopping the Control Center Service"

Control Center Agent (J2EE Runtime)

The **Control Center Agent (CCA)** runs on the Oracle Containers for J2EE (OC4J) server. Some capabilities of OWB related to accessing non-Oracle data, such as Code Templates and Web services, depend on Java code that executes outside the database, in an OC4J server called the Java or J2EE Runtime. For some heterogeneous data access scenarios, you may also need to install a standalone Java Runtime on hosts where there is no Oracle database installed.

You start the Control Center Agent with `ccastart` from the command line. OWB provides the `cca_admin` utility to enable dynamic changes to Control Center Agent settings, without the need to shut down and subsequently restart the run-time environment.

Note: A **Code Template (CT)** contains the knowledge required by Oracle Warehouse Builder to perform a specific set of tasks against a specific technology, system, or set of systems. You must start the Control Center Agent before you deploy Code Templates.

Refer to "About Code Templates" in *Oracle Warehouse Builder Sources and Targets Guide* and "About Prebuilt Code Templates Shipped with Warehouse Builder" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

See Also: These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*:

- "Starting and Stopping the Control Center Agent"
- "About the cca_admin Utility"
- "Modifying the OC4J Startup Script"
- "Installing Code Templates"

Target Schemas

The data in your OWB project is stored in a **target schema** within the server. This data is in the form of data objects such as tables, views, and **dimension** and **cube** objects. In a traditional data warehousing implementation, there is typically only one target schema, which is the data warehouse target. You can design both relational and dimensional target schemas. To design a target schema, you first create the target module that will contain all the data objects. A **target module** is a container that holds the metadata definitions of all your data warehouse objects. Each target module corresponds to a target location that represents the physical location where the objects are stored.

See Also: "Designing Source and Target Schemas" and these additional topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:

- "Designing Target Schemas"
- "Deploying to Target Schemas and Executing ETL Logic"

Getting Help for Oracle Warehouse Builder

In addition to context-sensitive help available with the F1 key, OWB provides a Help Menu with links to utilities, training, the discussion forum, Oracle Technology Network and more. The Help Menu also contains the Help Center, the online version of the complete OWB documentation library.

Help Menu

The Help menu available from the Design Center contains these menu items:

- Search. Provides a shortcut to the search facility for the online Help Center.
- Table of Contents. Opens the Help Center with the Table of Contents selected.
- Help Favorites. Opens the Help Center with your favorites selected, if you have configured favorite Help topics.
- Dynamic Help.
- Start Page. Accesses the Start Page after the first time you have run OWB.
- Extensions. A link to Oracle Warehouse Builder Utility Exchange on OTN. The purpose of the OWB Utility Exchange is to provide the user community with a forum where utilities, code samples, and tips and tricks can be exchanged. The utilities posted here are not part of any Oracle production release and are, therefore, free-of-charge and not supported.
- Training. Provides a shortcut to Oracle University where you can find out about training for OWB.

- Discussion Forum.
- Oracle Technology Network. Provides a shortcut to OWB on OTN for this release.
- Check for Updates. Checks for OWB product updates. When grayed out, no updates are available.
- Session Properties. Displays information about the current workspace session such as workspace owner and name; connection properties like host name, service name and database version; and roles granted to the workspace owner.
- About. Displays version information about the current software release.

Help Center

The Help Center contains the complete OWB documentation set in HTML format, available for online reading and searching. The Help Center opens with the Contents tab active. Click the plus symbol to expand the contents. Use the Search facility to enter topics on which to search.

Documentation Library for Oracle Warehouse Builder

Oracle Warehouse Builder provides the documentation described in Table 2–1.

Table 2–1 Oracle Warehouse Builder Documentation Library

Title	Description and Use
Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux	You will use "Part I: Installing and Configuring Oracle Warehouse Builder" to perform any necessary installation tasks and to configure the OWB repository. "Part II: Administering Oracle Warehouse Builder," starting with the chapter "Managing Configurations" provides detailed procedures for configurations, the Control Center and Repository, Control Center Agent, managing content, using the Metadata Loader, and managing security.
Oracle Warehouse Builder Release Notes	The release notes contain any late-breaking information about corrections, troubleshooting, and known issues. You can scan through the release notes during the set up processes to see the last-minute notes about this release.
Oracle Warehouse Builder Concepts	Similar to Oracle Database Concepts, this book provides a high-level explanation of the architecture, user interface, and components within OWB. It provides a user interface tour chapter and overviews of the processes and steps used to perform typical tasks within OWB. This book provides links to more detailed information and procedures within the other books.
Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide	This book provides comprehensive procedures for designing target schemas, performing data transformations, generating code, and doing all the tasks for optimizing and managing data quality.
Oracle Warehouse Builder Sources and Targets Guide	This book lists all of the supported sources and targets and provides procedures for importing from sources and deploying to targets.

Table 2–1 (Cont.) Oracle Warehouse Builder Documentation Library

Title	Description and Use
Oracle Database 2 Day + Data Warehousing Guide	This book is part of the Oracle Database 2 Day + series and provides a good starting place to understand the data warehousing features offered with the database in addition to OWB.
Oracle Warehouse Builder API and Scripting Reference	This book describes the scripting language available with OWB and provides a complete language reference.
Oracle Warehouse Builder Help (Only available as online help within OWB.)	The comprehensive help system that provides online versions of the complete documentation library, and context-sensitive help for all UI objects.

3

User Interface Tour

Oracle Warehouse Builder provides a dynamic workspace for your projects and one, common look-and-feel for all editors, including automatic layout, dockable panels, and zoom capabilities. The Property Inspector standardizes the properties inspection interface for all objects. This section introduces the general user interface elements that are essential to using Oracle Warehouse Builder (OWB)

This section contains these topics:

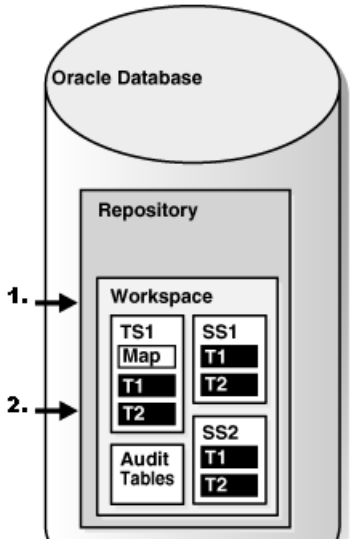
Workspaces

A **workspace** resides within the OWB repository on Oracle Database. The OWB repository can contain one or more workspaces, depending on how you want to organize your users and functional areas. The first step in building a new workspace is to assign it a name and to specify the path and directory where its source files will be saved. OWB organizes your workspace into projects and creates and organizes many of the configuration files required by the type of data system you are creating. For a description of projects, see "Projects" on page 3-2.

The workspace is hosted on an Oracle database. As a general user, you do not have full access to all the workspaces. Instead, you can access those workspaces to which you have been granted access and privileges. You log in to a workspace by starting the Design Center, which is the primary graphical user interface. The Design Center is used to import source objects, design ETL processes such as mappings, and ultimately define the integration solution. For information about the Design Center, see "Design Center" on page 3-6.

Figure 3–1 shows a representation of the OWB repository in Oracle Database. Within the repository, at least one workspace containing one project exists. In the figure, numeral **1.** points to a workspace and numeral **2.** points to several projects within that workspace. Multiple workspaces can reside within the OWB repository.

Figure 3–1 Oracle Database with a Workspace Contained within the OWB Repository



The diagram shows Oracle Database (symbolized as a cylinder) with the OWB repository residing within (as a rectangle). Within the repository is one OWB workspace (as a rectangle), which contains three projects: TS1, SS1, and SS2.

****

Projects

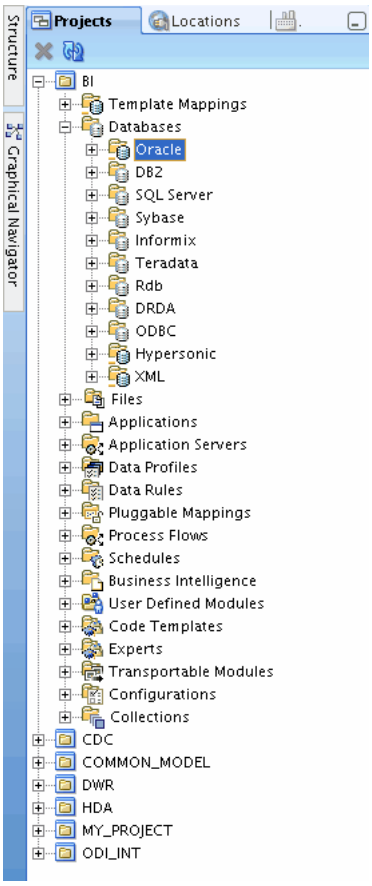
The **project** is the highest-level and largest object in the OWB workspace. Each project acts as the container for all objects in the data system that contains the sources and targets. The work you do on your data system is done within the context of a project. Projects store and organize related metadata definitions. You should include all the objects in a project that you think can or will share information. These definitions include data objects, mappings, and transformation operations. The definitions are organized into folders within the project. By creating multiple projects, you can organize the design and deployment of a large system.

You can create projects according to functional areas like Sales, HR, Customer Service, and so forth. Projects are organized in tree hierarchies with modules for specialized types of metadata and OWB objects. Multiple users can concurrently access an OWB project. Changes and updates are managed through a locking mechanism to ensure data integrity.

Projects contain these main structures, which are described in this section:

- Modules
- Folders
- Collections

Figure 3–2 Projects As They Appear in the Projects Navigator



The figure shows projects displayed as a tree hierarchy in the Projects Navigator, with the Business Intelligence (BI) project partially expanded with these object nodes: template mappings, databases, files, applications, application servers, data profiles, data rules, pluggable mappings, process flows, schedules, BI, user-defined modules, Code Templates, Experts, Transportable Modules, configurations, collections.

****

To create a project

1. In the Project Navigator, right-click a project, such as `MY_PROJECT`, and select **New**.

   The Create Project dialog box is displayed.

2. Click **Help** for additional instructions.

Each OWB workspace has a default project called `MY_PROJECT`. You can rename `MY_PROJECT`, or you can delete it after you create other projects. However, a workspace must contain at least one project at all times.

Because projects are the main design component in OWB, some restrictions are enforced to prevent you from deleting them unintentionally. You cannot delete:

- The currently active or expanded project.
- The only project in a workspace.

To delete a project

1. In the Project Navigator, collapse the project that you want to delete. You cannot delete the project when it is expanded.
2. Select and expand any other project.
3. Highlight the project you want to delete and, from the **Edit** menu, select **Delete.**

   or

   Right-click the project and select **Delete.**

   The Warehouse Builder Warning dialog box provides the option of putting the project in the recycle bin.
4. Click **OK** to delete the project.

Modules

A **module** is a container for actions that appears in the Projects Navigator, and that corresponds to a specific location in the Locations Navigator. A single location can correspond to one or more modules. However, a given module can correspond to only one metadata location and data location at a time. You can create a new module from the Projects Navigator.

The association of a module to a location enables you to perform certain actions more easily. For example, group or batch actions such as creating snapshots, copying, validating, generating, deploying, and so on, can be performed on all the objects within a module by selecting an action on the context menu when the module is selected.

To define the target schema, begin by creating a module. The Oracle target module is the first of several modules you create in OWB.

> **See Also:** "Creating Modules" in *Oracle Warehouse Builder Sources and Targets Guide* and "Configuring Target Modules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Folders

You can create **folders** to organize all or some objects in a target module based on specific object characteristics. Related tables and views that must be generated or deployed together can be placed under a common folder. For example, you may create user folders to group tables based on their functionality (sales, marketing, administration and so forth).

> **See Also:** These topics in *Oracle Warehouse Builder Sources and Targets Guide*:
>
> - "Creating User Folders"
> - "About Item Folders"

Collections

Collections are structures in OWB that store the metadata you want to export to other tools and systems. Collections enable you to perform the following tasks:

- Organize a large logical warehouse
- Validate and generate a group of objects

When you create a collection, you do not create new objects or copies of existing objects. You create shortcuts pointing to objects already existing in the project. You can use a collection to quickly access a base object and make changes to it.

You can define more than one collection within a project and an object can be referenced by more than one collection. For example, each user that accesses a project can create his own collection of frequently used objects. The users can also add the same objects (such as mappings, tables, or process flows) to their separate collections.

Each user can also delete either the shortcut or the base object. Shortcuts to deleted objects are deleted in the collection. When you open an object in a collection, you obtain a lock on that object. OWB prevents other users from editing the same object from another collection.

Use the Create Collection Wizard to define a collection as follows.

To define a new collection

1. Select and expand a project node in the Projects Navigator.
2. Right-click the Collections node and select **New.**

   OWB displays the Welcome page for the Create Collections Wizard. This page lists the steps to create a collection. Click **Next** to proceed.
3. Provide information on the following pages of the Create Collection Wizard:
   - Name and Description Page
   - Contents Page
   - Summary Page

Name and Description Page

Use the Name and Description page to provide a name and an optional description for the collection. The name should be unique within the module. In physical naming mode, type a name between 1 to 200 characters. Spaces are not allowed. In logical mode, the maximum number of characters is 200 and spaces are allowed.

Contents Page

The Contents page enables you to select the data objects that you want to refer to in the collection.

To add references to objects in a collection

1. Select and expand the project node in the left panel.

   The wizard displays a list of objects you can add to the collection.
2. Select objects from the **Available** section in the panel.

   Use the **Ctrl** key to select multiple objects. You can select objects at the object level or the module level. For example, under the Files node, you can add a specific file or add all the files in a given flat file module.

   If you add a module or another collection, OWB creates references to the module or collection and also creates references to objects contained within the module or collection.
3. Click the **right arrow**.

The wizard displays the list of objects under the Selected section on the right panel. You can remove objects from the list by selecting objects and clicking the left arrow.

Summary Page

The Summary page displays the objects selected for the collection. Review the objects and click **Back** to make changes to your selections. Click **Finish** to complete the collection definition. OWB creates the collection and adds it to the Projects Navigator.

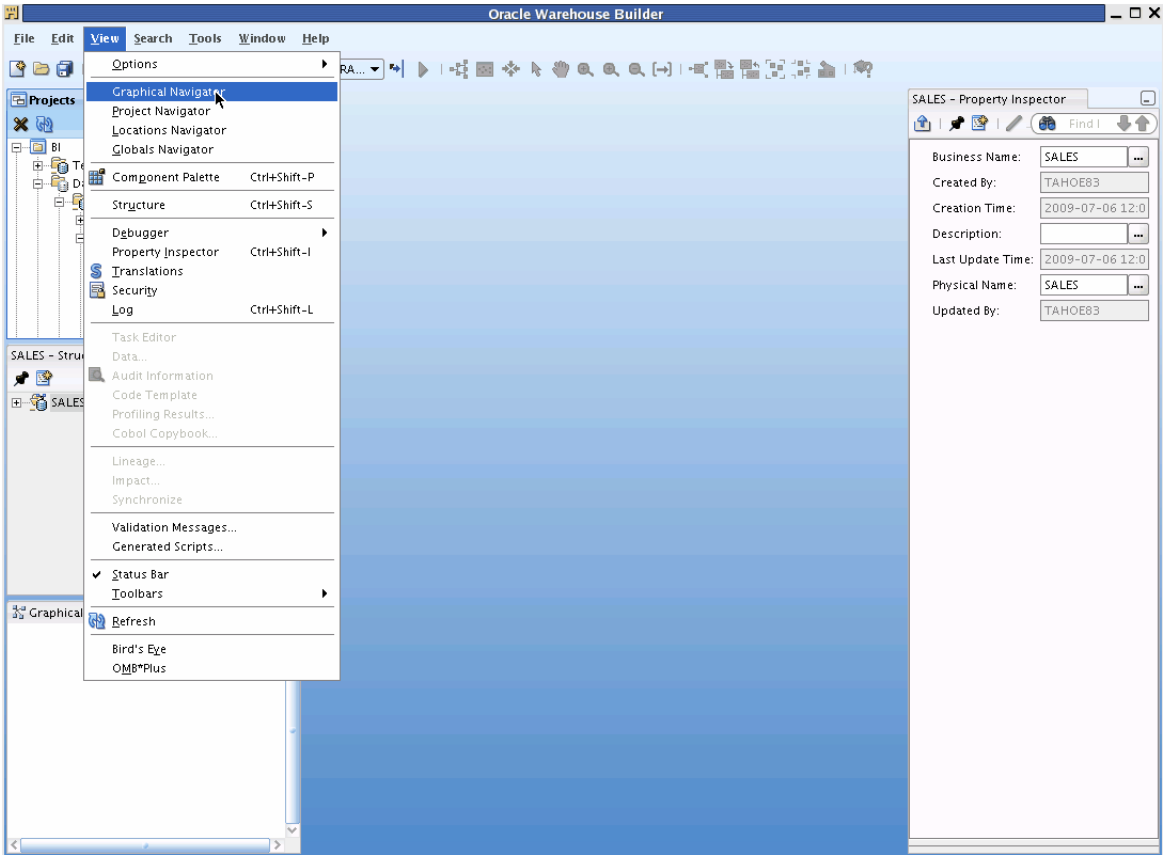
Design Center

The Design Center is the primary place for designing, managing, and building your data integration solution. You access all other tools from this central user interface and perform all commands for selected objects from here. The navigators in the Design Center are tree structures from which you select and manipulate projects and the different kinds of objects such as data locations, relational and dimensional data objects, flat files, ETL mappings to move and transform data, process flows to coordinate sequences of tasks, data profiles, rules and auditors for managing data quality, and so on. The Design Center also serves as a way for you to learn about the various kinds of components you may use in your projects, and also enables you to easily launch wizards to create most of the common object types.

Design Center hosts a number of specific editors and tools for working with the different objects. Most tools feature contextual linking, so that as you are working on an object such as a process flow, you can easily open the relevant editors on objects such as ETL mappings referenced in the process flow.

Figure 3–3 shows the Design Center canvas laid out as it usually appears the first time you start using OWB. In this figure, the Start Page has been closed and the View menu is open to show some of the options for adding to the canvas.

Figure 3–3 Design Center canvas



The figure shows the default Design Center layout with the View menu open. This is mainly a clean canvas with a blue background. The Projects Navigator is on the canvas at the left side and the Property Inspector is positioned on the right side.

****

Common Look and Feel with Oracle Application Development Framework

The Design Center user interface uses the Fusion Client Platform, the same core Integrated Development Environment (IDE) platform as Oracle JDeveloper and SQL Developer. The Design Center takes full advantage of usability research behind the Fusion Client Platform and provides developers with an environment that is consistent with other Oracle products.

Some of the productivity features of the Design Center and Java development environment are:

- Global Search: A Global Search field can be used to search across the entire Design Center IDE and your project for a word or phrase.
- Go to File: In addition to Go to Java Class (Alt+Minus), there is an option to Go to File (Ctrl+Alt+Minus) to enable you to quickly navigate to any file visible from your project.
- Gallery: You can store global objects in named collections, called galleries. You can create new galleries as needed.

Accessing Commands for Selected Objects in the Design Center

The Design Center provides several ways of accessing commands for selected objects:

- Right-click and select a command from the pop-up menu.
- Select a command from one of the menus on the menu bar.
- Click an icon on the toolbar.

> **Note:** After launching the Design Center, if classes cannot be found in their nominated directories, then the Messages area will have an Extended tab in addition to the Log tab. This tab will list as warnings those particular "errant" or missing classes. Two of these classes listed as missing can always be ignored: `help-share` and `ohj`.

Design Center Menus

The Design Center contains the following menus:

- Design
- Edit
- View
- Tools
- Window
- Help

> **Note:** Many of the commands available on the menu bar are also available on popup menus that you can access by right-clicking an object.

Design

The Design menu provides commands for creating a new object, and for configuring and editing parameters for the selected object. From this menu you import and export metadata to and from locations, objects, or folders. You validate the design of one or more selected objects. This menu provides commands for generating PL/SQL code and for deploying the PL/SQL code for one or more selected objects.

The Save command on the Design menu saves your changes to the workspace. The Revert to Saved command discards changes to the workspace.

See Also:

- Appendix A, "OWB Preferences"
- *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- *Oracle Warehouse Builder Sources and Targets Guide*

Edit

The Design Center Edit menu contains commands as follows.

- **Open Editor**: Opens the editor for the selected object.
- **Add/Remove Experts Here**: Displays the Add/Remove Experts dialog box, which you use to add the name of an expert to the menu. This command is displayed only to administrative users.

- **Cut**: Deletes an object as part of a cut-and-paste operation.
- **Copy**: Copies an object as part of a copy-and-paste operation.
- **Paste**: Pastes a cut or copied object into a new folder. For example, an administrator can cut an expert from the Expert folder in the Project Navigator to the Public Experts folder in the Global Navigator.
- **Delete**: Deletes one or more selected objects. A warning message provides the option of moving the object to the recycle bin.
- **Rename**: Enables you to edit the name of an object.
- **Find**: Displays the Find dialog box, which you use to search for an object within the current project. Searches are case-sensitive, and find only the objects that are visible (that is, in an expanded folder).
- **Properties**: Displays the Properties dialog box for the selected object.

View

The Design Center View menu contains the following commands:

- **Refresh**: Refreshes the objects displayed in the Project Navigator with the objects in the workspace. In a multiple user environment, this tool updates the Design Center with changes that other users have saved.
- **Expand**: Displays the folders or objects in the selected folder.
- **Expand All**: Displays all folders, subfolders, and objects in the selected folder.
- **Collapse**: Hides all folders and objects in the selected folder, but does not collapse the subfolders. The next time the folder is expanded, the subfolders are either collapsed or expanded, as they were before.
- **Collapse All**: Hides all folders and objects in the selected folder, and collapses any subfolders. The next time the folder is expanded, all of the subfolders are collapsed.
- **Data**: Displays the Data Viewer for a deployed and executed data object.
- **Profiling Results**: Displays the profile for the selected data object. You must have already created the profile.
- **Lineage**: Opens the Metadata Dependency Manager with the lineage analysis of the selected object.
- **Impact**: Opens the Metadata Dependency Manager with the impact analysis of the selected object.
- **Synchronize**: Updates an external table with any changes to its source file.
- **Validation Messages**: Displays the Validation Results dialog box with the results from validating the selected object.
- **Generated Scripts**: Displays the Generated Scripts dialog box with the results from generating the selected object.
- **Messages Log**: Displays the message log for the current session. The messages log area includes an Extensions tab when any extensions that are loaded into the ADF cause an error to occur. The Extensions tab only appears if there are errors; otherwise, no extensions tab appears. Click the Extensions tab (if it appears) to see any errors.

Tools

The Design Center Tools menu contains the following commands:

- **Control Center Manager**: Displays the Control Center Manager, from which you can deploy objects, start ETL jobs, and view the logs of previous jobs.
- **Job Monitor**: Displays the Control Center Jobs window for the selected Control Center or configuration.
- **Change Manager**: Displays the Metadata Change Management window, from which you can manage snapshots.
- **Metadata Dependency Manager**: Displays the Metadata Dependency Manager, from which you can evaluate the impact of changes to one or more objects on other objects in your data warehouse.
- **Repository Browser**: Displays the Repository Browser in your default Web browser, from which you can view reports about the repository or the Control Center. The browser listener must be running.
- **Optimize Repository**: Refreshes the statistics for the database schema where the repository is located. This refresh improves the execution time of the SQL commands that the product sends to the database, and thereby improves OWB performance. You should optimize the repository after importing or creating a large amount of metadata.
- **J2EE User Management**: Opens the J2EE User Management window, from which a user with administrative privileges can manage J2EE users.
- **SQL*Plus**: Opens a SQL*Plus window.
- **Recycle Bin**: Opens the recycle bin, from which you can restore deleted objects.
- **Clipboard**: Displays the Clipboard Contents window, which displays information about the object currently cut or copied to the clipboard.
- **Preferences**: Displays the preferences associated with your login name.

Window

The Design Center Window menu contains commands for displaying and hiding the navigators and OMB*Plus.

See Also:

- "Locations Navigator" on page 3-12
- "Globals Navigator" on page 3-12
- "OMB*Plus" on page 3-11 and Chapter 10, "Scripting and Automation"

Help

The Design Center Help menu provides access to the online help and other resources. Refer to "Getting Help for Oracle Warehouse Builder" on page 2-12 for a description.

The Design Center Toolbar

The Design Center toolbar provides easy access to the most commonly used commands, which are also available from the menu bar and on popup menus that you can access by right-clicking an object.

OMB*Plus

OMB*Plus is an optional panel that is available from the Design Center. You can display or hide it by choosing **OMB*Plus** from the main menu. The OMB*Plus command window enables you to manage objects in the design workspace using the OMB*Plus scripting language.

With the OMB*Plus scripting language you can:

- Perform complex actions.
- Define sets of routine operations.
- Perform batch operations.
- Automate a series of conditional operations.

> **See Also:** *Oracle Warehouse Builder API and Scripting Reference* for complete information about OMB*Plus, and Chapter 10, "Scripting and Automation" in this guide

Control Center Manager

You use the Control Center Manager to deploy design objects and subsequently execute the generated code. This is the client interface that interacts with the target schema through the Control Center Service, and provides access to the information stored in the active Control Center. As soon as you define a new object in the Design Center, the object is listed in the Control Center Manager under its deployment location. The Control Center Manager enables you to view and manage all aspects of deployment and execution, including status and results.

OWB uses only one Control Center Manager and corresponding Control Center Service for each database instance.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:
>
> - "Deploying Objects" for procedures on using the Control Center Manager for deployment
> - "Deploying to Target Schemas and Executing ETL Logic"
> - "Starting ETL Jobs" for procedures on using the Control Center Manager to execute generated code

Navigators

The Design Centers provides navigators to organize your workspace into categories that make it easier to find what you are looking for. Collapsible panels put common elements within easy reach. Objects and files that make up a composite node are exposed by clicking on the composite node.

The Navigator area in the Design Center contains the following tabs:

- Projects Navigator
- Locations Navigator
- Globals Navigator

Projects Navigator

The Projects Navigator provides a tree that organizes the various object categories in the workspace into folders or containers. Database sources and targets, flat files, external tables, ETL mappings, process flows, data rules, data auditors, and so forth are listed in the order in which they appear in a project. Within the workspace you can have one or more projects, and within each project are various categories of objects. When you select an object, you can create a new one like it, or edit, validate, and deploy it.

Every workspace must have at least one project, which initially is named `MY_PROJECT`. You can rename this project or use it as a template to create additional projects. When you create a new project, the Project Welcome page provides a guide for configuration.

Locations Navigator

Use the Locations Navigator to manage locations, connectors, and control centers for the entire workspace from one central location. You can display or hide it by choosing **Locations Navigator** from the View menu.

You can create locations and control centers while defining other objects. When you create a module, you can define its location. When you create a configuration, you can define a control center. You can create modules and configurations in the Projects Navigator. In contrast to these objects, you can only create a connector in the Locations Navigator.

Connectors Navigator of Locations

Provides a view of connectors to navigate, select, and manage.

Globals Navigator

The Globals Navigator provides users access to a set of shared objects:

- Public Transformations
- Public Application Servers
- Public Code Templates
- Public Experts
- Public User Defined Modules
- Public Data Rules
- Configuration Templates
- Icon Sets
- Security

Users with administrative rights can create a new object or edit an existing one most easily by right-clicking the object or object folder, then choosing a command from the pop-up menu. They can also copy objects from the Projects Navigator and paste them into an appropriate folder in the Globals Navigator. In this way, any user can develop an object for use by the team, and an administrator can publish it in the Globals Navigator.

The Security folder is displayed only to administrative users. It enables them to manage the access rights of all workspace users to objects in the workspace.

The Globals Navigator is an optional panel in the Design Center. You can display or hide it by choosing **Globals Navigator** from the View menu.

Editors

OWB provides contextual, JDev-style editors for accessing and editing the different types of data objects and document objects. Editors in OWB use one common look and feel across all objects, including automatic layout, dockable panels, and zoom capabilities. Many editors for components are fully-integrated, modeless editors. These appear in the editor area along with the other editors, and enable more productive navigation. Additionally, the property inspector standardizes the properties interface for all objects. The property inspector categories are used consistently throughout OWB to make using it more predictable.

The first time you open an editor, it displays with a menu bar, multiple tool bars, multiple windows along the left side, and a canvas or document area on the right.

> **See Also:** "Overview of the Mapping Editor" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* and Appendix B, "Object Editors"

Common Editor Components

In addition to the items listed in this section, editors typically contain Naming Mode, Rename Mode, Read/Write, and Validation Mode. The percent zoom and navigation mode are also indicated. You will find that editors are contextual and they can be customized. This section describes how editors appear when you first start using the Design Center, which is the main graphical user interface in OWB.

The following items are common to most editors in OWB:

- **Title Bar:** At the top of the editor, the title bar displays the name of the object, for example a mapping, and the access privileges you have on the object.
- **Menu Bar:** Below the title bar, the menu bar provides access to the editor commands. You can access the menu bar by clicking on one of its options or by using hot keys. For example, to access the Mapping menu, press **Alt +M.**
- **Toolbar:** Below the menu bar, the toolbar provides icons for commonly used commands.
- **Canvas:** The canvas provides the document area where you design and modify mappings.
- **Indicator Bar:** Along the lower edge of the editor there are mode icons, indicators, and descriptions.

Editor Windows

You can resize a window by placing your mouse on the border of the window, pressing the mouse button when the double sided arrow appears, and dragging your mouse to indicate the desired size. You can move a window by placing the mouse on the Title Bar, and dragging the mouse to the desired location. To show or hide windows, select **Window** from the menu bar and either activate or deactivate the check mark corresponding to the window.

Mapping Navigator

When you first start an editor, OWB displays a navigator in the upper left corner. The navigator provides a tree of all the activities on the canvas and their parameters. When you select an activity on the canvas, OWB goes to the activity on the navigator tree.

Properties Inspector

When you first start an editor, OWB displays the properties inspector in the lower left corner. The properties inspector displays the properties for the object, its operators, and attributes in the operators. Also, when you select an object either from the canvas or the navigator, OWB displays the properties in the properties inspector.

New Gallery

When you right-click an object, a wizard is invoked that guides you to make this object globally available in the object gallery for the JDev-based IDE. It presents a series of panels that capture the necessary information to create the object's component type. This enables you to specify the component name for the new component and to select the package into which you'd like to organize the component. If the package does not yet exist, the new component becomes the first component in that new package. When you click **Finish**, OWB creates a new JDeveloper-type component by saving its XML component definition file. If you have set your Java generation options to generate classes by default, OWB also creates the initial custom Java class files.

Palette

When you first start an editor, OWB displays a palette along the left side that contains activity icons you can drag and drop onto the canvas. The operator palette features collapsible panels and divider sections to organize related operators. A quick search field is provided to help locate operators. You can add commonly-used operators to your Favorites list for easier access later, and another panel keeps track of your recently used ones. You can relocate the palette anywhere on the editor. You can choose to hide or display the palette by clicking on **Operator Palette** listed under **View** in the menu bar.

Editor Toolbars

Editors typically provide the following task-oriented toolbars: general, graphic, generation, and palette. With the exception of the palette, the editor by default displays the toolbars below the menu bar. You can move, resize, or hide each of the toolbars.

- **General Toolbar:** Use this toolbar to call common operations such as save all, exporting diagram, validating, generating, and printing.
- **Diagram Toolbar:** Use this toolbar to navigate the canvas and change the zoom level of objects on the canvas.
- **Debug Toolbar:** Use this toolbar to call commands for debugging a mapping, for example.
- **Palette Toolbar:** The palette contains operator icons. To include an operator, drag an operator icon onto the editor canvas. As OWB includes over 50 operators, you may want to sort and display the operators based on type.

Editor Display Options

You can control how an editor displays objects on the canvas by selecting **View** from the menu bar and selecting **Options**.

Use the F1 key to see help for each of the options. You can either select or deselect any of the options.

Data Viewer

The data viewer area displays the data stored in the data object.

Generation

The Generation panel displays the generation and validation results for a data object. This panel is hidden when you first open the editor window. It appears the first time you generate or validate a data object.

The Generation window contains Script and Message tabs. The Script tab displays the generated scripts to implement the data object selected in the canvas. The Message tab displays the validation messages for the selected data object. Double-click a message to view the complete message text.

Wizards

OWB provides wizards to guide you in making numerous design decisions in defining objects and operators. Each wizard begins with a welcome page that provides an overview of the steps you must perform, and each wizard concludes with a summary page listing your selections.

> **See Also:** "About Operator Wizards" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* and these topics in *Oracle Warehouse Builder Sources and Targets Guide*:
>
> - "Using the Import Metadata Wizard"
> - "Using the Create Flat File Wizard"
> - "Using the Create Item Folder Wizard"
> - "Using the Create Business Area Wizard"
> - "Using the Create Logical Table Wizard"
> - "Using the Create Catalog Folder Wizard"

Repository Browser

The Repository Browser provides Web-based access to workspaces to view design details and metadata properties, and to generate reports about workspace objects and data.

With the Repository Browser you can also view reports for both high-level and detailed ETL runtime information as follows:

- Timings for each mapping and process flow.
- Details of activities for each process flow.
- Error details.
- Deployment information to manage separate target environments.

Note: In order to open the Repository Browser, you must have the `ACCESS_PUBLICVIEW_BROWSER` system privilege. You automatically have this privilege when you are the owner of the repository that you want to browse. If you are not the owner of the repository, then you must contact your database administrator to have this privilege set.

See Also:

- "About the Repository Browser" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Configuring the Repository Browser Environment" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

4

Data Access and Movement

This section discusses data sources and targets, and how OWB provides solutions for accessing and moving data among disparate systems with simple to complex requirements.

This section contains the following topics:

About Metadata on Source Data

Metadata is the data that describes the contents of a given object in a data source or target. For example, **metadata** for a table indicates the column names and data types for each column.

Before you import source metadata into OWB, first create a module that will contain these metadata definitions. The type of module you create depends on the source from which you are importing metadata. For example, to import metadata definitions from an Oracle database, you create or use an Oracle module. To import metadata definitions from flat files, you create a flat file module.

> **See Also:** "Modules and Locations" on page 4-2 in this guide, and "General Steps for Importing Metadata from Sources" in *Oracle Warehouse Builder Sources and Targets Guide*

OWB must have metadata for any source or target object that will be manipulated in your project. The most basic metadata needed by OWB can be created or derived in several ways:

- OWB can directly extract existing metadata from most database sources or targets. For example, when connecting to an Oracle database, OWB queries the database dictionary to extract all needed metadata on tables, views, sequences, dimensions, cubes, data types, PL/SQL packages, and so on.
- You can define and use SQL- or XML-based custom metadata stores to retrieve definitions of source and target objects such as tables and views.

- When you design data objects that do not already exist, the metadata that describes the object is created by the design process.
- For data files extracted from some mainframe sources, OWB can interpret Cobol Copybook files that describes the structure of the data file, and create its source metadata based on that.
- OWB application adapters or application connectors provide additional metadata about ERP and CRM application sources.

The metadata management and reporting features in OWB, and data lineage and impact analysis, depend on, and leverage the metadata about the sources and targets and transformations that move data among them, which accumulates in your OWB projects over time.

> **See Also:** "Connecting to Sources and Targets in Oracle Warehouse Builder" in *Oracle Warehouse Builder Sources and Targets Guide*

The Import Metadata Wizard

The Import Metadata Wizard automates importing metadata from a database into a module in OWB. You can import metadata from Oracle Database and non-Oracle databases. Each module type that stores source or target data structures has an associated Import Wizard, which automates the process of importing the metadata to describe the data structures. Importing metadata saves time and avoids keying errors, for example, by bringing metadata definitions of existing database objects into OWB.

The Welcome page of the Import Metadata Wizard lists the steps for importing metadata from source applications into the appropriate module. The Import Metadata Wizard for Oracle Database supports importing of tables, views, materialized views, dimensions, cubes, external tables, sequences, user-defined types, and PL/SQL transformations directly or through object lookups using synonyms.

When you import an external table, OWB also imports the associated location and directory information for any associated flat files.

> **See Also:** These topics in *Oracle Warehouse Builder Sources and Targets Guide*:
>
> - "Connecting to Data Sources and Importing Metadata"
> - "Importing Metadata from Excel Using the Metadata Import Wizard"

Modules and Locations

A **module** is a container structure for the data objects in OWB. Modules are equivalent to schemas from a database perspective. A **location** stores credentials needed to access a schema. Locations are linked to modules to provide access to metadata and the data itself. A module can have many locations associated with it, but only one can be the configured location at any point in time.

The association of a module to a location enables you to perform certain actions more easily in OWB. For example, you can reimport metadata by reusing an existing module. Furthermore, when you deploy ETL processes in subsequent steps, modules enable you to deploy related objects together, such as process flows.

Modules are created by expanding the Projects Navigator until you find the node for the data object type for which you want to create the module. For example, if the source data is stored in an Oracle Database, then you expand the Databases node to

view the Oracle node. If the source data is in an SAP R/3 system, then you expand the Applications node to view the SAP node. By right-clicking the node, you can select New and launch the Create Module Wizard.

> **See Also:** "Modules" on page 3-4 in this guide, and "Creating Modules" in *Oracle Warehouse Builder Sources and Targets Guide*

> **See Also:** These topics about locations in *Oracle Warehouse Builder Sources and Targets Guide*:
>
> - "About Locations"
> - "Automatically Created Locations"
> - "Types of Locations"
> - "Creating Locations"

About Connectors

A **connector** is a logical link created by a mapping between a source location and a target location. The connector between schemas in two different Oracle Databases is implemented as a database link, and the connector between a schema and an operating system directory is implemented as a database directory.

You do not need to create connectors manually if your user ID has the credentials for creating these database objects. OWB will create them automatically the first time you deploy the mapping. Otherwise, a privileged user must create the objects and grant you access to use them. You can then create the connectors manually and select the database object from a list.

To create a database connector, from within the Connection Navigator, expand the Locations folder and the subfolder for the target location. Right-click **DB Connectors** and select **New**. The Create Connector wizard opens with prompts for creating the connection. You can create a directory connection by right-clicking **Directories** and selecting **New**, following the same steps.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about the `CREATE DATABASE LINK` command
> - *Oracle Database SQL Language Reference* for more information about the `CREATE DIRECTORY` command

Summary of Supported Sources and Targets

This section summarizes the supported sources and targets for each Location node as displayed in the Connections Navigator.

OWB supports sources from:

- Oracle Database releases 8.1 and later.
- Any database accessible through Oracle Heterogeneous Services (Gateways), including but not limited to DB2, DRDA, Informix, SQL Server, Sybase, and Teradata.
- Any data store accessible through the Code Templates (which use JDBC), including, but not limited to, DB2, SQL Server, Sybase, and Teradata.

- Any data store accessible through the ODBC Data Source Administrator, including but not limited to Excel and MS Access
- Delimited and fixed-length flat files.
- ERP and CRM applications such as Oracle E-Business Suite, Peoplesoft and Siebel, from which data can be extracted using SQL.
- SAP R/3, from which data is extracted using officially supported methods based on native ABAP code.

OWB supports the following targets:

- Oracle Database release 8.1 and later.
- Third-party databases accessed through Oracle gateways or ODBC.
- Comma-delimited and XML format flat files.
- Oracle BI tools, such as Oracle Business Intelligence Suite Enterprise Edition (OBI EE).
- OWB can deploy or execute process flows and schedules to Oracle Enterprise Manager and Oracle Workflow. In general, you can deploy a schedule in any Oracle Database location, release 10g or later.

> **See Also:** "Supported Sources and Targets" in *Oracle Warehouse Builder Sources and Targets Guide* for a detailed and complete list

Flat Files as Data Sources or Targets

OWB supports using **flat files** as data sources. Flat files are typically in plain text comma-separated or tab-separated format, or proprietary binary formats, and may be stored on different types of operating systems. You first ensure that you have direct access either locally, or by creating a network connection, through TCP/IP or NFS for example. OWB provides the Create Flat File Wizard to create a file object, which will contain the imported flat-file definitions.

The Create Flat File Wizard provides intuitive prompts for importing metadata. To start the wizard, you right-click the file module and select **Import**. You can filter the filenames from which to import by applying wildcards. The wizard creates definitions for the files and inserts the file names under the Flat File module in the Project Navigator.

The locations that correspond to this module appear as folders on your computer's file system. The metadata is imported into a file module in OWB and becomes visible in the workspace.

You can then sample the metadata from these flat files. The Flat File Sample Wizard enables you to view a sample of the flat file and define record organization and file properties. You can sample and define common flat file formats such as string and ASCII. The Flat File Sample Wizard also enables the importation of new data types such as GRAPHIC, RAW, and SMALLINT.

After you have created the flat-file locations and have imported the flat-file metadata, you are ready to import data. You introduce data from a flat file into an OWB mapping either through an external table or a flat-file operator. Depending on how the data is to be transformed, use one of the following options:

- External Table Option
- Flat File Operators

Note: The Create Flat File Wizard enables specifying the character set and defining single or multiple record types.

See Also: "Using Flat Files as Sources or Targets" in *Oracle Warehouse Builder Sources and Targets Guide* for procedures

External Table Option

If the data is to be joined with other tables or requires complex transformations, then use the External Table option. An external table can be used as a source and enables data from the associated flat file to be viewed from SQL as a table. When you use an external table in a mapping, its column properties are based on the SQL properties that you defined when importing the flat file. OWB generates SQL code to select rows from the external table. You can also get parallel access to the file through the table. You can either import an existing external table from another database or define a new external table.

You can also use an external table to combine the loading and transformation within a single set-based SQL DML statement. You do not have to stage the data before inserting it into the target table.

See Also: "Using External Tables" in *Oracle Warehouse Builder Sources and Targets Guide* for procedures

Flat File Operators

In cases where large volumes of data are to be extracted and little transformation is required, use the flat file operator. When you use a flat file operator, SQL*Loader code is generated. From the flat file operator, you can load the data to a staging table, add indexes, and perform transformations as necessary. The transformations you can perform on data introduced by a flat file operator are limited to SQL*Loader transformations only.

See Also: *Oracle Database Utilities* for more information about differences between external tables and SQL*Loader (flat file operators)

Data Systems Access with Code Templates

OWB achieves seamless management of JDBC-accessible data systems through its **Code Template (CT)** technology. Code Templates provide native heterogeneous connectivity to Oracle and JDBC-accessible data systems and disparate platforms. Code templates can be used as an alternative to Oracle Gateways as a means of accessing other databases. In addition to OWB being the best ETL solution for Oracle databases, with OWB you can move data that is located in non-Oracle systems into and out of your project quickly and easily. JDBC connectivity provides an alternative to Oracle Gateways as a means of accessing other databases.

Code Template technology in OWB also provides direct data movement among JDBC-accessible databases, without stopping in an Oracle database in between. For example, if you need to move data from DB/2 to SQL Server for some reason, then you can do so from OWB without moving the data through Oracle at all.

See Also: "Using Code Templates to Load and Transfer Data" in *Oracle Warehouse Builder Sources and Targets Guide*

Note: When getting ready to generate code for a CT, you must have only the editor open for the CT on which you are focused. Otherwise, when you generate code for the CT, you will get conflicting results.

Generic Connectivity and Oracle Database Gateways

Oracle alternatively provides the **generic connectivity agent** and optional Oracle Database Gateways for connecting to non-Oracle databases such as SQL Server, Sybase, Informix, Teradata, DRDA, ODBC, and other sources. OWB can communicate with non-Oracle systems using Oracle Database Heterogeneous Services and a complementary agent if you choose this route.

The generic connectivity agent is intended for low-end data integration solutions. The transfer of data is subject to the rules of specific ODBC or OLE DB drivers installed on the client computer. In this case, you do not need to purchase a separate transparent gateway. You use the generic connectivity agent included with Oracle Database. You must still create and customize an initialization file for your generic connectivity agent.

Oracle Database Gateways provide specific connection agents, designed and optimized for other databases, which you install and configure separately as needed. For example, for a Sybase data source, you install the Sybase-specific gateway. The non-Oracle system appears as a remote Oracle Database to which you can then create a connection and import its data into Oracle. This is especially useful for database environments that do not intend to harbor data marts or data warehouses, but that need integration with a set of other data sources.

See Also:

- `http://www.oracle.com/pls/db111/gateways` for installation instructions and information on specific gateway agents
- *Oracle Database Heterogeneous Connectivity User's Guide* for more information on Oracle Database Gateways

Transportable Modules for Moving Large Volumes of Data

A **transportable module** enables OWB to rapidly copy a group of related database objects from one database to another.

You use the Design Center to create a transportable module for which you specify the source database location and the target database location. Then you select the database objects to be included in the transportable module. The metadata of the selected objects are imported from the source database into the transportable module. The metadata is stored in the workspace. To physically move the data and metadata from source into target, you need to configure and deploy the transportable module to the target location. During deployment, both data and metadata are extracted from the source database and created in the target database.

See Also: "Moving Large Volumes of Data Using Transportable Modules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

5
Data Objects

The data in the data system that you implement with Oracle Warehouse Builder is stored in target schemas. This data is in the form of data objects such as tables, views, dimensional objects, and cubes. This section discusses relational and dimensional data objects that you design for your target system, and business intelligence objects for analytical views.

This section contains these topics:

Types of Data Objects

OWB uses relational and dimensional data objects and intelligence objects as follows:

- Relational objects rely on tables and table-derived objects to store and link all of their data. Relational objects include tables, views, materialized views, and sequences.
- Dimensional objects contain additional metadata to identify and categorize your data. Dimensional objects include dimensions and cubes.
- Intelligence objects enable you to store definitions of business views. You can deploy these definitions to Oracle Business Intelligence Suite Enterprise Edition (OBI EE). In OWB, intelligence objects are displayed under the Business Intelligence node in the Project Navigator.

See Also:

- "Designing Source and Target Schemas" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Overview of Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for a complete list and discussion of the data objects that you can use in OWB, and the data types that you can use and create
- "Defining Business Intelligence Objects" in *Oracle Warehouse Builder Sources and Targets Guide*

Data Object Editors

OWB provides contextual data object editors to create, edit, configure, validate, and deploy Oracle data objects. The data object editors work with relational, dimensional, and business intelligence objects.

Use data object editors to:

- Create, edit, and delete relational and dimensional objects.
- Create, edit, and delete the following business intelligence objects: Business Areas and Item Folders.
- Define relationships between Oracle data objects.
- Validate, generate, and deploy Oracle data objects.
- Define and edit all aspects of a data object such as its columns, constraints, indexes, partitions, data rules, and attribute sets.
- Define implementation details for dimensional objects with a relational implementation.

See Also:

- "Data Object and Document Editors" on page B-1 in this guide
- "Validating Data Objects" and "Generating Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Data Viewers

OWB Data Viewers are available for dimensions and cubes, as well as relational objects (tables, views, materialized views, sequences, external tables and so forth). The dimension and cube Data Viewers enable interactive, logical-level browsing of the contents of these objects at a logical level, regardless of the details of the underlying physical storage. The dimension Data Viewer lets you browse and drill into the dimension members organized by hierarchy and level. The cube Data Viewer enables interactive browsing of the contents of the cube, and pivoting and drilling down into the data along any dimension.

To access a Data Viewer, from the Projects Navigator, right-click a data object and select **Data**.

By default, the Data Viewer for the selected object displays the first hundred rows of data. To retrieve the next set of rows, click **Get More**. Alternatively, you can click **More** to perform the same action. The columns and column names displayed in the Data Viewer are taken directly from the location in which the actual table is deployed.

See Also: "Using the Data Viewer to View Data Stored in Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Dimensional Objects: Dimensions and Cubes

The term **dimensional object** refers to both dimensions and cubes. OWB provides wizards to create and maintain dimensions by answering simple questions. OWB supports two types of dimensional objects:

- **Dimensions**, discussed under "About Creating and Using Cubes and Dimensions" on page 5-3 in this guide.

- **Cubes**, discussed under "Cubes: Measures and Dimensionality" on page 5-15 in this guide.

> **See Also:** "Defining Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for complete procedures

ROLAP versus MOLAP Implementations

OWB separates the logical design of dimensional objects from their storage. The logical design, which consists of business rules, first focuses on the structure and the content of the dimensional object. You can then choose to store the dimensional object in a relational ROLAP or an analytic MOLAP implementation.

- ROLAP and relational implementations store the dimensional object in a relational schema in the database.
- A MOLAP implementation stores the dimensional object in analytic workspaces in the database.

OWB enables you to use the same metadata to create and manage both your relational and multidimensional data stores. Separating the logical design from the physical implementation has the advantage of making design of ETL logic transparent. Regardless of the physical storage implementation, the logic for loading dimensions and cubes is identical.

Uses for OLAP

Dimensional objects provide complex analytic power to your data warehouse. After you load data into dimensional objects, you can run complex analytical queries that answer your business questions. These analytic queries include time-series analysis, inter-row calculations, access to aggregated historical and current data, and forecasts. Multidimensional objects are more effective in answering these types of queries quickly.

> **See Also:** "ROLAP Implementation of Dimensional Objects" and "MOLAP Implementation of Dimensional Objects" under "Overview of Implementing Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

About Creating and Using Cubes and Dimensions

A **dimension** consists of a set of levels and a set of hierarchies defined over these levels. Working with cubes and dimensional objects consists of four high-level processes:

1. Dimension Creation
2. Dimension Implementation
3. Dimension Deployment
4. Dimension Loading

Dimension Creation

When you define dimensional objects such as cubes, you describe the logical relationships that help store data in a more structured format. For example, to define a dimension, you describe its attributes, levels, and hierarchies.

OWB provides the following two methods to define dimensions:

- **Create Dimension Wizard:** Creates a fully functional dimensional object along with implementation objects that store the dimension data. Provides default settings for the most common values. Use this wizard when you want to quickly create a dimension without manually specifying settings.

  See Also: "Defaults Used By the Create Dimension Wizard" under "Creating Dimensions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

- **Dimension Editor:** Use editors to create or edit dimensional objects. Use editors to create a dimensional object when you want to specify settings that are different from the default settings used by the wizards. Also use editors to create dimensional objects that use certain advanced options that are not available when you use wizards. For example, to create a relational dimension that uses a snowflake schema implementation, you must use the editor. When you use the wizard, the default implementation method used is the star schema. However, you can edit a dimension that you created using the Create Dimension Wizard and modify it to use a snowflake schema implementation.

Dimension Implementation

To implement a dimension is to create the physical structure of the dimensional object. OWB provides the following implementations for dimensions:

- Relational Implementation of Dimensions
- ROLAP Implementation of Dimensional Objects
- MOLAP Implementation of Dimensional Objects

Note: To use a MOLAP implementation, you must have the following software installed:

- Oracle Database 11*g* Enterprise Edition with the OLAP option
- OLAP 10.1.0.4 or higher

You set the Deployment Option configuration property to specify the type of implementation for a dimensional object. For more information on setting this property, see "Configuring Dimensions" and "Configuring Cubes" in *Warehouse Builder Online Help*.

Relational Implementation of Dimensions The dimensional data is stored in implementation objects that are typically tables. For relational dimensions, OWB can use a star schema, a snowflake schema, or a manual schema to store the implementation objects.

See Also: "Star Schema" on page 5-12 and "Snowflake Schema" on page 5-13 in this guide

ROLAP Implementation of Dimensional Objects In addition to creating DDL scripts that can be deployed to a database, a ROLAP implementation enables you to create CWM2 metadata for the dimensional object in the OLAP catalog.

See Also: "About the OLAP Catalog" on page 5-6 in this guide

MOLAP Implementation of Dimensional Objects The dimension data is stored in an **analytic workspace** in Oracle Database 11*g*. This analytic workspace, in turn, is stored in the database.

> **See Also:** *Oracle OLAP User's Guide* for more information about analytic workspaces

Dimension Deployment

To instantiate the dimensional objects in the database, you must deploy them. To specify the type of implementation for dimensional objects, you set the deployment option. The configuration parameter Deployment Options enables you to set the deployment option.

OWB provides the following deployment options for dimensional objects.

- **Deploy All:** For a relational or ROLAP implementation, the dimensional object is deployed to the database and a CWM definition to the OLAP catalog. For a MOLAP implementation, the dimensional object is deployed to the analytic workspace.
- **Deploy Data Objects Only:** Deploys the dimensional object only to the database. You can select this option only for dimensional objects that use a relational implementation.
- **Deploy to Catalog Only:** Deploys the CWM definition to the OLAP catalog only. Use this option if you want applications such as Oracle Business Intelligence Enterprise Edition to access the dimensional object data after you deploy data only. Use this option if you previously deployed with "Data Objects Only" and now want to deploy the CWM Catalog definitions without re-deploying the data objects again.

  > **See Also:** "About the OLAP Catalog" on page 5-6 in this guide

- **Deploy Aggregation:** Deploys the aggregations defined on the cube measures. This option is available only for cubes.

Deploying Dimensional Objects that Use a MOLAP Implementation

Dimensional objects that use a MOLAP implementation can be deployed just after you define them. You can use the Design Center or the Control Center Manager to deploy a dimensional object.

Deploying Dimensional Objects that Use a Relational or ROLAP Implementation

Before you deploy a relational or ROLAP dimensional object, ensure that the implementation details are specified. This means that the dimensional object should be bound to its implementation objects. Also ensure that the dimensional object is valid.

After you perform binding, deploy the dimensional object. Before you deploy a dimensional object, ensure that all its implementation objects are deployed. For a dimension, this includes the sequence that is used to generate the surrogate identifier of the dimension levels. Alternatively, you can deploy the implementation objects together with the dimensional object.

Dimension Loading

After you deploy a dimension, you load data into it by creating a mapping. Use the Mapping Editor to create the mapping, which loads data from the source objects into the dimensional object. You then deploy and execute this mapping.

See Also:

- "Dimension Operator as a Target" in *Warehouse Builder Online Help* for more information on loading dimensions
- "Cube Operator" in *Warehouse Builder Online Help* for information on loading cubes

About the OLAP Catalog

The OLAP catalog is the metadata repository provided for the OLAP option in the Oracle Database. This metadata describes the data stored in both relational tables and in analytic workspaces.

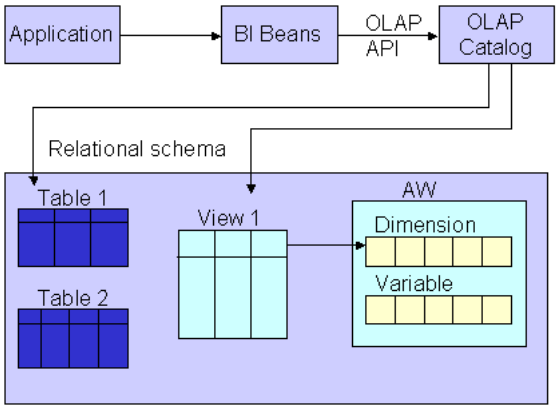
When you deploy a dimensional object using OWB, you can specify if the dimensional object metadata should be stored in the OLAP catalog.

OLAP metadata is dynamically projected through a series of views called the active catalog views (views whose names begin with ALL_OLAP2_AW).

In Oracle Database 11*g*, the OLAP catalog metadata is used by OLAP tools and applications to access data stored in relational star and snowflake schemas. External application such as Oracle Business Intelligence Enterprise Edition use the OLAP catalog to query relational and multidimensional data. The application does not need to be aware of whether the data is located in relational tables or in analytic workspaces, nor does it need to know the mechanism for accessing it.

Figure 5–1 describes how the OLAP catalog enables applications to access data stored in relational tables and analytic workspaces.

Figure 5–1 Using the OLAP Catalog to Access Dimensional Objects



This graphic illustrates the use of the OLAP catalog to access relational data. In this graphic, the data originates in an application, represented in the top left corner by a box labeled Application. To the right of this box are two boxes, from left to right, labeled BI Beans and OLAP Catalog. The data flows from the Application to BI Beans and then to the OLAP Catalog. The data flow is represented by arrows connecting these boxes.

In the lower part of the screen is the Relational Schema. It contains the following: Table 1, Table 2, View 1, and an AW that contains the following: Dimension and Variable. An arrow connects the OLAP Catalog to Table 1. An arrow also connects the OLAP Catalog to View 1. View 1 is connected to Dimension that is contained in the AW.

****

The OLAP catalog uses the metadata it stores to access data stored in relational tables or views. The OLAP catalog defines logical multidimensional objects and maps them to the physical data sources. The logical objects are dimensions and cubes. The physical data sources are columns of a relational table or view.

Orphan Management Policy for Dimensions

The orphan management policy in OWB enables you to manage orphan records in dimensions and cubes. An orphan record is one that does not have a corresponding, existing parent record.

Orphan records can occur when:

- A record that is loaded into a dimensional object does not have a corresponding parent record.
- A record is deleted from a dimensional object. This could result in the child records of the deleted record not having an existing parent record.

OWB enables you to specify a different orphan management policy for loading dimensional data and removing dimensional data.

> **See Also:** "Orphan Management for Dimensional Objects" under "Overview of Cubes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Relational Dimensions

A relational dimension is a structure that organizes data. Examples of commonly used dimensions are Customers, Time, and Products.

Relational dimensions provide improved query performance because users often analyze data by drilling down on known hierarchies. An example of a hierarchy is the Time hierarchy of year, quarter, month, day. The Oracle Database uses these defined hierarchies by rewriting queries that retrieve data from materialized views rather than detail tables.

Typical relational dimension tables have the following characteristics:

- A single column primary key populated with values called warehouse keys.

  Warehouse keys that provide administrative control over the dimension, support techniques that preserve dimension history, and reduce the size of cubes.
- One or more hierarchies that are explicitly defined as dimension objects. Hierarchies maximize the number of query rewrites by the Oracle server.

> **See Also:** "Creating Relational Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Rules for Dimension Objects

When you create a dimension object using OWB, the dimension must conform to the following rules:

- A dimension must have a surrogate identifier and a business identifier.
- The surrogate identifier can consist of only one attribute. However, the business identifier can consist of more than one attribute.
- Every dimension level must have at least one attribute.

- A dimension attribute can be either a surrogate identifier, a business identifier, a parent identifier, or a regular attribute.
- A regular attribute can also play only one of the following roles at a time: effective date, expiration date, or triggering attribute.
- A dimension that uses a relational or ROLAP implementation must have at least one level.
- Any database table or view that implements a dimension that uses a relational or ROLAP implementation must have only one `LONG`, `LONG RAW`, or `NCLOB` column.
- For a dimension that uses a relational or ROLAP implementation, all level attributes must bind to database tables or views only.
- A dimension that uses a relational or ROLAP implementation must be associated with a sequence that is used to load the dimension key attribute.
- The dimension key attribute of a dimension that uses a relational or ROLAP implementation must bind to the primary key of a table.
- A Type 2 Slowing Changing Dimension (SCD) must have the effective date, expiration date, and at least one triggering attribute.
- A Type 3 Slowing Changing Dimension (SCD) must have the effective date and at least one triggering attribute.

Note: For dimensions with a ROLAP implementation, there are implications and limitations related to the various dimension structures when either reporting on the underlying tables or deploying to the OLAP catalog. Refer to the topic, "Limitations of Deploying Dimensions to the OLAP Catalog" under "Overview of Dimensions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

See Also: "About the OLAP Catalog" under "Overview of Implementing Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

About Defining a Dimension

Creating a dimension consists of:

- Defining Dimension Attributes
- Defining Levels
- Defining Level Attributes
- Defining Hierarchies

See Also: "Defining Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Defining Dimension Attributes

A **dimension attribute** is a descriptive characteristic of a dimension member. It has a name and a data type. A dimension attribute is applicable to one or more levels in the dimension. They are implemented as level attributes to store data.

In OWB, you define dimension attributes when you define a dimension. The list of dimension attributes must include all the attributes that you may need for any of the levels in the dimension. Dimension attributes are the only attributes that are visible in Oracle Business Intelligence Suite Enterprise Edition (OBI EE) and other OLAP tools.

For example, the Products dimension has a dimension attribute called Description. This attribute is applicable to all the levels Total, Groups, and Products and stores the description for each of the members of these levels.

Defining Levels

The levels in a dimension represent the level of aggregation of data. A dimension must contain at least one level, except in the case of a dimension that contains a value-based hierarchy. Every level must have level attributes and a level identifier.

For example, the dimension Products can have the following levels: Total, Groups, and Product.

Surrogate, Business, and Parent Identifiers

Every level must have two identifiers: a surrogate identifier and a business identifier. When you create a dimension, each level must implement the dimension attributes marked as the surrogate identifier and business identifier (attributes, in the case of a composite business identifier) of the dimension.

Surrogate Identifiers A surrogate identifier uniquely identifies each level record across all the levels of the dimension. It must be composed of a single attribute. Surrogate identifiers enable you to hook facts to any dimension level as opposed to the lowest dimension level only.

For a dimension that has a relational or ROLAP implementation, the surrogate identifier should be of the data type `NUMBER`. Because the value of the surrogate identifier must be unique across all dimension levels, you use the same sequence to generate the surrogate identifier of all the dimension levels.

For a relational implementation, the surrogate identifier serves the following purposes:

- If a child level is stored in a different table from the parent level, each child level record stores the surrogate identifier of the parent record.
- In a fact table, each cube record stores only the surrogate identifier of the dimension record to which it refers. By storing the surrogate identifier, the size of the fact table that implements the cube is reduced.

Business Identifiers A business identifier consists of a user-selected list of attributes. The business identifier must be unique across the level and is always derived from the natural key of the data source. The business identifier uniquely identifies the member. For example, the business identifier of a Product level can be its Universal Product Code (UPC), which is a unique code for each product.

> **Note:** For a dimension that has a MOLAP implementation, the business identifier can consist of only one attribute.

The business identifier does the following:

- Identifies a record in business terms
- Provides a logical link between the fact and the dimension or between two levels
- Enables the lookup of a surrogate key

When you populate a child level in a dimension, you must specify the business identifier of its parent level. When you populate a cube, you must specify the business identifier of the dimension level to which the cube refers.

Parent Identifier A parent identifier is used to annotate the parent reference in a value-based hierarchy. For more information on value-based hierarchies, see "Value-based Hierarchies" on page 5-11.

For example, an `EMPLOYEE` dimension with a value-based hierarchy, has the following dimension attributes: `ID`, `FIRST_NAME`, `LAST_NAME`, `EMAIL`, `PHONE`, `JOB_ID`, `HIRE_DATE`, and `MANAGER_ID`. In this dimension, ID is the surrogate identifier and `MANAGER_ID` is the parent identifier.

Defining Level Attributes

A level attribute is a descriptive characteristic of a level member. Each level in the dimension has a set of level attributes. To define level attributes, you select the dimension attributes that the level will implement. A level attribute has a distinct name and a data type. The data type is inherited from the dimension attribute that the level attribute implements. The name of the level attribute can be modified to be different from that of the dimension attribute that it implements.

Every level must implement the attribute marked as the surrogate identifier and the business identifier in the set of the dimension attributes.

Defining Hierarchies

A dimension hierarchy is a logical structure that uses ordered levels or a set of data values (for a value-based hierarchy) as a means of organizing data. A hierarchy describes parent-child relationships among a set of levels. A level-based hierarchy must have at least one level. A level can be part of more than one hierarchy.

For example, the Time dimension can have the following two hierarchies:

Fiscal Hierarchy: Fiscal Year > Fiscal Quarter > Fiscal Month > Fiscal Week > Day

Calendar Hierarchy: Calendar Year > Calendar Quarter > Calendar Month > Day

All hierarchies must be strict 1:n relationships. One record in a parent level corresponds to multiple records in a child level. But one record in a child level corresponds to only one parent record within a hierarchy.

Dimension Roles

A dimension role is an alias for a dimension. In a data warehouse, a cube can refer to the same dimension multiple times, without requiring the dimension to be stored multiple times. Multiple references to the same dimension may cause confusion. So you create an alias for each reference to the dimension, thus allowing the joins to be instantly understandable. In such cases, the same dimension performs different dimension roles in the cube.

For example, a sales record can have the following three time values:

- Time the order is booked
- Time the order is shipped
- Time the order is fulfilled

Instead of creating three time dimensions and populating them with data, you can use dimension roles. Model one time dimension and create the following three roles for the time dimension: order booked time, order shipped time, and order fulfillment

time. The sales cube can refer to the order time, ship time, and fulfillment time dimensions.

When the dimension is stored in the database, only one dimension is created and each dimension role references this dimension. But when the dimension is stored in the OLAP catalog, OWB creates a dimension for each dimension role. Thus, if a time dimension has three roles, three dimensions are created in the OLAP catalog. However, all three dimensions are mapped to the same underlying table. This is a workaround because the OLAP catalog does not support dimension roles.

Note: Dimension roles can be created for dimensions that have a relational implementation only.

Level Relationships

A level relationship is an association between levels in a dimension hierarchy. Level relationships are implemented using level attributes that store the reference to the parent level in the hierarchy.

For example, the Products dimension has the following hierarchy: Total > Groups > Product. OWB creates two level relationships: Product to Groups and Groups to Total. Two new attributes implement this level relationship: one in the Product level and one in the Groups level. These attributes store the surrogate ID of the parent level.

Control Rows

When you load data into a dimension, OWB creates control rows. Control rows link fact data to a dimension at any level, thus enabling the reuse of a dimension in different cubes.

> **See Also:** "Using Control Rows" in Chapter 3, "Defining Dimensional Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Value-based Hierarchies

A value-based hierarchy is a dimension in which hierarchical relationships are defined by a parent dimension attribute and a child dimension attribute. This is different from a level-based hierarchy, referred to as a hierarchy in this chapter, in which the hierarchical relationships are defined between levels.

You create a value-based hierarchy when the parent-child relationships cannot be grouped into meaningful levels. A value-based hierarchy has no levels. When you create the dimension attributes, you must specify which dimension attribute is the parent attribute.

For example, consider an `EMPLOYEE` dimension that has the following dimension attributes: `ID`, `FIRST_NAME`, `LAST_NAME`, `EMAIL`, `PHONE`, `JOB_ID`, `HIRE_DATE`, `DESCRIPTION`, and `MANAGER_ID`. This dimension contains a parent-child relationship in which the `MANAGER_ID` attribute identifies the manager of each employee. But these relationships may not form meaningful levels across the organization. This is because the number of levels between an employee and the CEO is not the same for all employees. There may be four levels between employee A and the CEO, whereas, there may be six levels between employee B and the CEO. In such cases, you create a value-based hierarchy with `MANAGER_ID` as the parent identifier.

You can create value-based hierarchies using the Data Object Editor only. For more information about specifying a parent attribute, see "Attributes Tab" in *Warehouse Builder Online Help*.

> **Note:** Value-based hierarchies can be created only in dimensions that use a MOLAP implementation.

Implementing a Dimension

Implementing a dimension consists of specifying how the dimension and its data are physically stored. You can choose either a relational implementation, ROLAP implementation, or MOLAP implementation for a dimension. For more information about setting the implementation method, see "Dimension Implementation" on page 5-4.

When you store dimension data in a relational form, you can implement the dimension using one of the following methods:

- Star Schema
- Snowflake Schema

Star Schema

In a star schema implementation, OWB stores the dimension data in a single table. Because the same table or view stores data for more than one dimension level, you must specify a dimension key column in the table. The dimension key column is the primary key for the dimension. This column also forms the foreign key reference to the cube.

Each level implements a subset of dimension attributes. By default, the level attribute name is the same as the dimension attribute name. To avoid name conflicts caused by all level data being stored in the same table, OWB uses the following guidelines for naming in a star table:

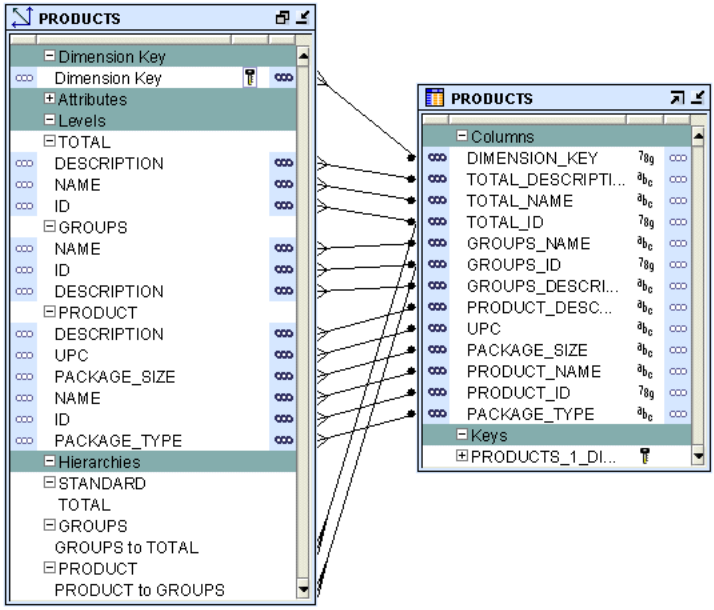
- If the level attribute name is not unique, OWB prefixes it with the name of the level.
- If the level attribute name is unique, OWB does not use any prefix.

> **Note:** To ensure that no prefixes are used, you must explicitly change the level attribute name in the Create Dimension wizard or the Data Object Editor.

For example, if you implement the Products dimension using a star schema, OWB uses a single table to implement all the levels in the dimension.

Figure 5–2 displays the star schema implementation of the Products dimension. The attributes in all the levels are mapped to different columns in a single table called `PRODUCTS`. The column called `DIMENSION_KEY` stores the surrogate ID for the dimension and is the primary key of the table.

Figure 5–2 Star Schema Implementation of Products Dimension



The screenshot is a diagrammatic representation showing the table columns to which each attribute in the Products dimension is mapped. On the left is the PRODUCTS dimension. On the right is the PRODUCTS implementation table. There are arrows from attributes in the PRODUCTS dimension to the columns in the PRODUCTS implementation table.

****

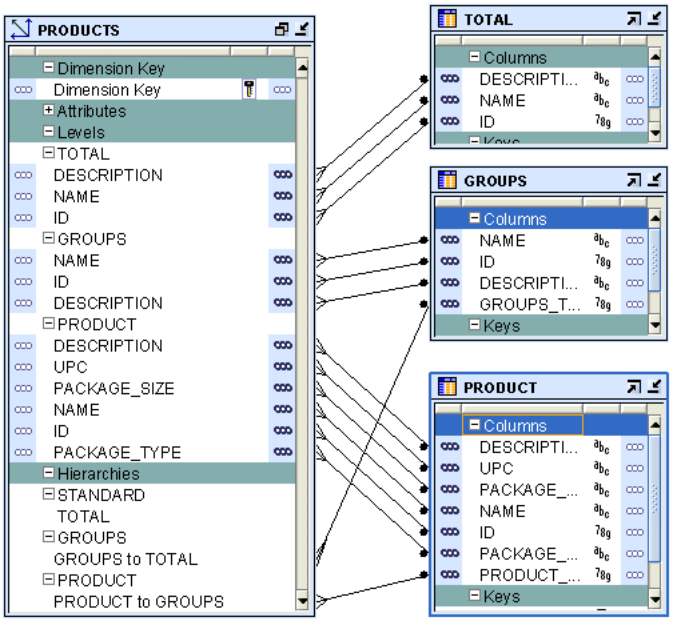
For relational or ROLAP dimensions that use a star implementation, you can bind attributes from more than one levels to the same database column. A database column that is bound to attributes from more than one dimension levels is referred to as a *shared column*. For a Type 2 SCD, you cannot set the level attributes that are bound to a shared column as triggering attributes.

Snowflake Schema

In a snowflake schema implementation, OWB uses more than one table to store the dimension data. Separate database tables or views store the data pertaining to each level in the dimension.

Figure 5–3 displays the snowflake implementation of the `PRODUCTS` dimension. Each level in the dimension is mapped to a different table.

Figure 5–3 Snowflake Schema Implementation of the Products Dimension



This screenshot is a diagrammatic representation of the attribute bindings of the dimension when it is implemented using a snowflake schema. To the left is the PRODUCTS dimension. To the right are three tables ordered from top to bottom: TOTAL, GROUPS, and PRODUCT. There are arrows from attributes in the PRODUCTS dimension to columns in the tables TOTAL, GROUPS, and PRODUCTS.

****

Slowly Changing Dimensions (SCDs)

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. In data warehousing, there are three commonly recognized types of SCDs: Type 1, Type 2, and Type 3.

With the appropriate licensing, you can use OWB to define, deploy, and load all three types of SCDs. You can create slowly changing dimensions only for dimensions that use a relational implementation.

Note: Type 1 does not require additional licensing; however, Type 2 and Type 3 SCDs require the Warehouse Builder Enterprise ETL Option. Refer to *Oracle Database Licensing Information*.

See Also: "Overview of Slowly Changing Dimensions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for complete information about the types of SCDs and how to use them

About Type 1 Slowly Changing Dimensions

In a Type 1 Slowly Changing Dimension (SCD), the new data overwrites the existing data. Typically, this type is not considered an SCD and most dimensions are of this type. Thus the existing data is lost as it is not stored anywhere else. This is the default type of dimension you create. You need not specify any additional information to

create a Type 1 SCD. Unless there are specific business reasons, you must assume that a Type 1 SCD is sufficient.

> **See Also:** "Creating Slowly Changing Dimensions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for detailed information about all types of SCDs

Time Dimensions

A time dimension is a dimension that stores temporal data. Time dimensions are used extensively in data warehouses. OWB enables you to create and populate time dimensions. You can use OWB to create both fiscal and calendar time dimensions.

When you create a time dimension using the wizard, OWB creates the mapping for you to execute to populate the time dimension. Also, the data loaded into the time dimension conforms to the best practices recommended by OWB for a time dimension.

> **See Also:** "Creating Time Dimensions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Cubes: Measures and Dimensionality

A **cube** is a data object that contains measures, and links to one or more dimensions. The axes of a cube contain dimension members, and the body of the cube contains measure values. Most measures are additive. For example, sales data can be organized into a cube whose edges contain values for Time, Products, and Promotions dimensions and whose body contains values from the measures Value sales, and Dollar sales.

Note: In the relational implementation of a cube, the cube is linked to dimension tables over foreign key constraints. Since data integrity is vital, these constraints are critical in a data warehousing environment. The constraints enforce referential integrity during the daily operations of the data warehouse.

> **See Also:** "Overview of Cubes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Cube Definitions

A cube consists of the set of measures defined over a set of dimensions as follows.

- Cube Measures
- Cube Dimensionality

Cube Measures

A measure is data, usually numeric and additive, that can be examined and analyzed. Examples of measures include sales, cost, and profit. A cube must have one or more measures. You can also perform aggregation of measures. Only numeric measures can be aggregated.

Cube Dimensionality

A cube is defined by a set of dimensions. A cube can refer to a level that is not the lowest level in a dimension.

For cubes that use a pure relational implementation, you can reuse the same dimension multiple times with the help of dimension roles. For more information on dimension roles, see "Dimension Roles" on page 5-10.

Before you validate a cube, ensure that all the dimensions that the cube references are valid.

To define a dimension reference, specify the following:

- The dimension and the level within the dimension to which the cube refers.

  For a cube that uses a relational implementation, you can refer to intermediate levels in a dimension. However, for cubes that use a MOLAP implementation, you can only reference the lowest level in the dimension. OWB supports a reference to the non surrogate identifier of a level, for example, the business keys.

- For dimensions that use a relational or ROLAP implementation, a dimension role for each dimension to indicate what role the dimension reference is performing in the cube. Specifying the dimension role is optional.

When you define a MOLAP cube, the order in which you define the dimension references is important. The physical ordering of dimensions on disk is the same as the order in which you define the dimension references. The physical ordering is tightly coupled with the sparsity definition. Define the dimension references in the order of most dense to least dense. Time is usually a dense dimension, and listing it first expedites data loading and time-based analysis. For more information on defining dimension references, see "Dimensions Page" or "Dimensions Tab" in *Warehouse Builder Online Help*. For more information on sparsity, see Advanced Dialog Box" in *Warehouse Builder Online Help*.

Default Aggregation Method

You can define aggregations that should be performed on the cube. For ROLAP cubes, you can only define a single aggregation method for the cube. For MOLAP cubes, you can define a different aggregation method for each dimension of each measure. OWB enables you to use the same aggregation function for all the cube measures or specify different aggregate functions for each measure.

OWB supports the following default aggregation methods: SUM, SSUM (scaled SUM), AVERAGE, HAVERAGE (hierarchical average), MAX, MIN, FIRST, LAST, AND, OR, HIERARCHICAL_FIRST and HIERARCHICAL_LAST. If you do not want to perform aggregation, select NOAGG. The methods AND and OR are not applicable for cubes that use a multidimensional implementation.

Note: You cannot define aggregation for pure relational cubes.

Implementing a Cube

When you implement a cube, you specify the physical storage details for the cube. You can implement a cube in a relational form or a multidimensional form in the database.

The types of implementation are:

- Relational and ROLAP Implementation of a Cube
- MOLAP Implementation of a Cube

To set the type of implementation for a cube, use the **Deployment Option** configuration property.

> **See Also:** "Creating Cubes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Relational and ROLAP Implementation of a Cube

The database object used to store the cube data is called a fact table. A cube must be implemented using only one fact table. The fact table contains columns for the cube measures and dimension references. For more information on setting the implementation option for a cube, see "Dimension Implementation" on page 5-4.

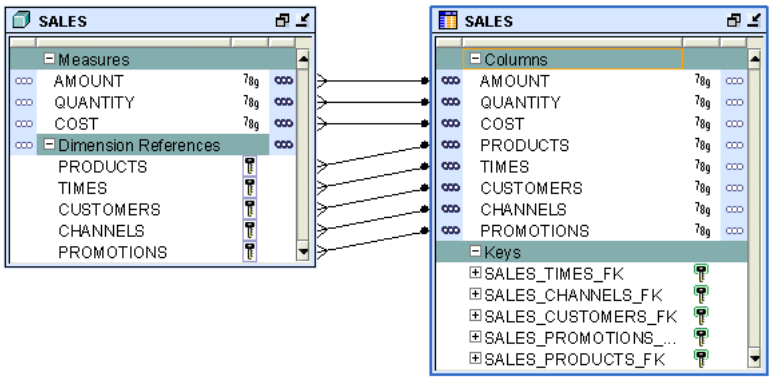
To implement a cube:

- Select a table or materialized view that will store the cube data.
- For each measure, select a column that will store the measure data.
- For each dimension reference, select a column that will store the dimension reference.

  Each dimension reference corresponds to a column on the fact table and optionally a foreign key from the fact table to dimension table. The 1:n relationships from the fact tables to the dimension tables must be enforced.

Figure 5–4 displays the bindings for the relational implementation of the SALES cube. The data for the SALES cube is stored in a table called SALES.

Figure 5–4 Implementation of the Sales Cube



The screenshot shows the relational implementation of the SALES cube. To the left is the SALES cube. To the right is a SALES table. There are arrows from the SALES cube to the SALES table connecting the cube attributes and dimensions to the table columns.

****

MOLAP Implementation of a Cube

Storing the cube and its data in an analytic workspace is called a MOLAP implementation. You can store multiple cubes in the same analytic workspace. For more information on OLAP implementation, see "MOLAP Implementation of Dimensional Objects" on page 5-5.

Solve Dependency Order of Cube

Certain business scenarios may require the dimensions in a cube to be evaluated in a particular order. The order in which the dimensions are evaluated is called the solve dependency order of the cube. For example, in the Sales cube, the Time dimension may need to be evaluated before the Products dimension. For each dimension of the cube, you can specify a dependency on another dimension of the cube.

The advantage of specifying the dependency order is that it enables OWB to optimize the query speed of calculating the joins of the dimension and cubes. For example, retrieving results from the sales cube based on Time criteria may be more selective than retrieving result based on Products criteria. In this case, you can specify that for the Sales cube, the Products dimension depends on the Time dimension.

Specifying the solve dependency order is optional. If you do not specify a dependency order, the optimizer determines the solve-order with additional flexibility.

> **See Also:** "Configuring Cubes" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

6

Data Transformation

This section discusses basic concepts related to design and implementation of data extraction, transformation and loading (ETL) mappings in Oracle Warehouse Builder.

This section contains the following topics:

Data Transformation with OWB Mappings

Data transformation is the term for converting data from a source data format into a destination data format. Data transformations typically require two steps: a) data mapping (from source to target) to capture any transformations that must occur, and b) code generation to create the actual transformation process. After you import your source data and define the target, you decide how to transform the source data into the output desired for the target. The Mapping Editor in OWB guides you on how to transform the data by designing mappings. A mapping describes the sequence of operations required to extract data from sources, transform the data, and load the data into one or more targets.

Transformations are PL/SQL functions, procedures, packages, and types that enable you to transform data. You use transformations when designing mappings and process flows that define ETL processes.

> **See Also:** "About Data Transformation in Warehouse Builder" and "Overview of Oracle Warehouse Builder Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Data Flow and Transformation-Code Generation in Mappings

Mappings provide a visual representation of the flow of the data and the operations performed on the data. Based on the ETL logic that you define in a mapping, OWB generates the code required to implement your design. OWB can generate code for the following languages:

- **PL/SQL:** PL/SQL stands for Procedural Language/Standard Query Language. It extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL.

- **SQL*Loader**: SQL*Loader is an Oracle tool for loading data from files into Oracle Database tables. It is the most efficient way to load large volumes of data from flat files.
- **SAP ABAP**: ABAP is a programming language for developing applications for the SAP R/3 system, a business application subsystem.
- **Code Templates (CT mappings):** For Code Template (CT) mappings, Warehouse Builder generates data extraction or other mapping code based on the contents of a Code Template.

Note: You can create and define mappings using OMB*Plus, the scripting interface for OWB as described in *Oracle Warehouse Builder API and Scripting Reference*.

See Also: These sections in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:

- "Designing Process Flows"
- "Best Practices for Designing PL/SQL Mappings"
- "Best Practices for Designing SQL*Loader Mappings"
- "Retrieving Data From the SAP System"
- "Creating Code Template Mappings"
- "Mappings and Process Flows Reference"
- "Configuring Mappings Reference"

Mapping Operators

The **mapping operator** is the basic design element for a mapping. As you design a mapping, you select operators from the Mapping Editor palette, and you can visually drag them onto the work area or canvas. Operators handle how to represent sources and targets in the data flow. Operators also define how to transform the data from source to target. The operators you select affect how you will design the mapping.

Based on the operators you select, OWB assigns the mapping to one of the following Mapping Generation Languages:

- PL/SQL. OWB generates PL/SQL code for all mappings that do not contain either a flat file operator as a source, or a SAP/R3 source. Design considerations for PL/SQL mappings depend on whether you specify a row-based or set-based operating mode.
- SQL*Loader. When you define a flat file operator as a source, OWB generates SQL*Loader code. To design a SQL*Loader mapping correctly, follow the guidelines described in "Using Flat File Source and Target Operators" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.
- ABAP (SAP-based script). When you define a SAP/R3 source, OWB generates ABAP code. For mapping design considerations for SAP sources, see "Creating SQL*Loader, SAP, and Code Template Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.

Each of these languages require you to adhere to certain rules when designing a mapping.

See Also: The following topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:

- "Creating PL/SQL Mappings"
- "Creating SQL*Loader Mappings to Extract Data from Flat Files"
- "Creating SAP Extraction Mappings"
- "Mapping Operators that are Only Supported Directly in Oracle Target CT Mappings"

Pluggable Mappings

A **pluggable mapping** is a reusable grouping of mapping operators that works as a single operator. Pluggable mappings are similar to functions in programming languages such as SQL*Plus and C, and enable you to re-use the ETL logic contained within.

Once defined, a pluggable mapping appears as a single mapping operator, nested inside a mapping. You can use a pluggable mapping more than once in the same mapping, or in other mappings. You can include pluggable mappings within other pluggable mappings.

Like any operator, a pluggable mapping has a signature, which consists of input and output attributes that enable you to connect it to other operators in various mappings. The signature is similar to the input and output requirements of a function in a programming language.

A pluggable mapping can be either *reusable* or *embedded*:

- **Reusable pluggable mapping:** A pluggable mapping is reusable if the metadata it references can exist outside of the mapping within which it is contained.
- **Embedded pluggable mapping:** A pluggable mapping is embedded if the metadata it references is owned only by the mapping or pluggable mapping in which it is contained.

Note: The use of pluggable mappings requires the Oracle Warehouse Builder Enterprise ETL Option. Refer to *Oracle Database Licensing Information* for details about this option.

See Also: "Using Pluggable Mappings" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for procedures

Transformations for Designing Mappings

Transformations are PL/SQL functions, procedures, table functions, and packages that enable you to transform data. You use transformations when designing mappings and process flows that define ETL processes.

Transformations are organized as follows:

- Predefined Transformations and Custom Transformations
- Transformation Libraries

Predefined Transformations and Custom Transformations

OWB provides a set of **predefined transformations** that enable you to perform common transformation operations. These predefined transformations are part of the Oracle Library that consists of built-in and seeded functions and procedures. You can directly use these predefined transformations to transform your data.

A **custom transformation** is one that is created by the user. Custom transformations can use predefined transformations as part of their definition. You can also import PL/SQL packages. Although you can modify the package body of an imported package, you cannot modify the package header, which is the signature for the function or procedure.

> **See Also:** "About Transformations" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Transformation Libraries

A **transformation library** consists of a set of reusable transformations. Each time you create a project, OWB creates a Transformation Library containing transformation operations for that project. This library contains the standard Oracle Library and an additional library for each Oracle module defined within the project.

Transformation libraries are available under the Public Transformations node of the Global Navigator in the Design Center.

Transformation libraries are one of the following types:

- Oracle Library, a collection of predefined functions from which you can define procedures for your Global Shared Library.
- Global Shared Library, a collection of reusable transformations created by the user.

> **See Also:** "About Transformation Libraries" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Table Functions

OWB provides the ability to define table function operators in mappings. Use table function operators to represent a **table function** in a mapping. Table function operators enable you to manipulate a set of input rows and return another set of rows of the same or different cardinality. Using table functions can greatly improve performance when loading your data warehouse.

> **See Also:** "Table Function Operator" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

7

Dependency and Change Management

This section discusses how metadata is managed and made secure in Oracle Warehouse Builder and introduces the Metadata Loader, snapshots, the Metadata Dependency Manager, and lineage and impact analysis.

This section contains these topics:

Metadata Security Concepts

OWB enables you to design security on the metadata you store in the design repository. The design repository is an Oracle Database with users, roles, and access privileges already defined. OWB metadata security operates in addition to the Oracle Database security. The Oracle Database provides security for data while OWB provides security for the metadata.

In addition to being registered in the repository, all users must also be database users in the design repository database. Database users have access to the data in the database by using SQL Plus. However, they do not have access to OWB and its metadata unless the users are also registered in OWB.

Metadata security is optional and flexible. You can:

- Apply no metadata security controls or define a metadata security policy.
- Define multiple users and apply full security.
- Implement your own security strategy based on the Security Service.

Also, after you define a security strategy, you can later adapt the strategy to be more or less restrictive. The following sections describe how to implement metadata security using the Design Center.

Security policies can be managed from within the Repository Assistant or by using OMB*Plus commands.

See Also:

- "About Metadata Security" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*
- "Managing Metadata Dependencies" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

About the Security Service

Only users with administrative privileges can access the Security Service and change security policy in OWB.

When you install OWB and then use the Repository Assistant to create a design repository, OWB assigns the design repository owner you define to be the default administrator. The first time you start the Design Center after installation, you must log in as the design repository owner. You can then define additional administrators or other users as necessary.

Log in to the OWB Design Center as the design repository owner and OWB displays the Globals Navigator.

Under the security node, there are two predefined roles, ADMINISTRATOR and EVERYONE. The one predefined user is the design repository owner, *REPOS_OWNER* in this example, which is assigned the ADMINISTRATOR role by default.

To perform actions under the Security node, select an object and right-click to view all of the possible operations. Or select an object and select **Edit** from the menu bar.

See Also: These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*:

- "Managing Security"
- "Managing Configurations"

Metadata Lifecycle Management Concepts

OWB provides several solutions for copying and moving metadata for the purposes of backup, history management, and version management. You can either take snapshots of your metadata or use the Metadata Loader (MDL) utility.

Snapshots enable you to capture the metadata definitions of design objects created in the Project Navigator. You can capture all the design objects in a project or selectively choose objects in a project to include in a snapshot.

The Metadata Loader utility enables you to copy and move all types of metadata objects in a repository. With this utility, you can move metadata between OWB repositories that reside on platforms with different operating systems.

With both solutions, you can export the metadata into a third-party version control tool such as ClearCase or SourceSafe.

See Also: These topics in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

- "About Snapshots and Metadata Loader"
- "Comparison of Snapshots and Metadata Loader"

The Metadata Loader

Using the Metadata Loader (MDL) utility, you can import and export metadata from any object in the Project Navigator, Global Navigator, and Connection Navigator. You can then move exported files into a third-party version control tool such as Oracle Repository, ClearCase, or SourceSafe. You can enter annotations for your MDL export file to keep track of the information contained in the file.

The Metadata Loader enables you to populate a new repository and transfer, update, or restore a backup of existing repository metadata. You can copy or move metadata objects between repositories, even if those repositories reside on platforms with different operating systems.

You can use the Design Center to run the Metadata Loader utilities. The Design Center provides a graphical interface that guides you through the process of exporting and importing metadata.

> **See Also:** "Using the Metadata Loader" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*

Metadata Snapshots

A **snapshot** captures all the metadata information about the selected objects and their relationships at a given point in time. While an object can only have one current definition in a workspace, it can have multiple snapshots that describe it at various points in time. Snapshots are stored in the Oracle Database, in contrast to Metadata Loader exports, which are stored as separate disk files. You can, however, export snapshots to disk files. Snapshots are also used to support the recycle bin, providing the information needed to restore a deleted metadata object.

When you take a snapshot, you capture the metadata of all or specific objects in your workspace at a given point in time. You can use a snapshot to detect and report changes in your metadata. You can create snapshots of any objects that you can access from the Projects Navigator.

Note that a snapshot of a collection is not a snapshot of just the shortcuts in the collection but a snapshot of the actual objects.

> **See Also:**
>
> - "Using Snapshots" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux*
> - *Oracle Warehouse Builder API and Scripting Reference* for information about creating and managing snapshots using scripts

About Managing Snapshots

You can manage your snapshots from the Metadata Change Management window in the Design Center. To open this window, select **Change Manager** from the Tools menu.

The Metadata Change Management window contains a menu bar and a toolbar. You can start most tasks in several different ways, either by using the menus, clicking the tools, or right-clicking a snapshot or a component.

> **See Also:** "Managing Snapshots" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and Linux* for complete procedures

The Metadata Dependency Manager (MDM)

The **Metadata Dependency Manager** (MDM) enables you to plan your project by previewing the impact of the changes or future changes for "what-if" analysis. When you plan to introduce changes to your source systems, you can gauge the impact of that change on your warehouse design. If changes have already been introduced, then you can plan the time required to update your ETL design and rebuild your data warehouse.

> **See Also:** "Managing Metadata Dependencies" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for complete procedures

The Metadata Dependency Manager Menus and Commands

The Metadata Dependency Manager is a graphical interface that represents the dependencies between objects. Knowing that an object has dependencies with another object helps identify potential impacts when things change.

The Metadata Dependency Manager has the following components:

- Menu Bar
- Analysis
- Edit
- View
- Window
- Toolbars
- DM Tree
- Property Inspector
- Canvas

Menu Bar

The Metadata Dependency Manager menu bar provides commands for performing various tasks. Some of these commands are also available on the toolbars.

Analysis

The Analysis menu contains the following commands:

- **Close**: Closes the Dependency Manager.
- **Export Diagram**: Exports the active diagram to the local file system as an SVG or JPEG file.
- **Print Options**: Provides Print Setup, Preview, and Print options for printing the diagram.

Edit

The Edit menu contains the following commands:

- **Open Editor**: Opens the Editor for the currently selected object.
- **Hide**: Removes the selected objects from the canvas. Use the Refresh command on the View menu to restore the hidden objects.
- **Select All**: Selects all objects displayed on the canvas.

- **Propagate Changes**: Displays the Propagate Changes dialog box for a selected attribute. Use it to change the value of an attribute and to propagate that change to all objects downstream. For example, you can select a column in a table object and change its metadata such as its name or data type.
- **Group Selected Objects**: Creates a group containing the selected objects on the canvas. A folder icon represents all objects in the group. Double-click the icon to display the individual objects in the group. Grouping enables you to reduce clutter on the canvas when there are many objects.
- **Ungroup Selected Objects**: Eliminates the selected group so that all objects are represented individually.
- **Group By Module**: Automatically groups all objects by module. A folder icon represents the module and all objects in the module. Double-click the icon to display the individual objects in the group.
- **Ungroup Modules**: Eliminates the module groups so that all objects are represented individually.

View

The View menu contains the following commands:

- **Toolbars**: Displays or hides the Graphic tools or the Edit tools.
- **Mode**: Sets the pointer for one of these actions:
  - **Select**: Selects one or more objects on the canvas.
  - **Pan**: Moves the entire diagram on the canvas.
  - **Interactive Zoom**: Expands the diagram as you move the pointer down, or shrinks the diagram as you move the pointer up.
  - **Navigate Edge**: Selects the next object in the flow.
- **Zoom**: Displays a list of percentages that expand or shrink the diagram.
- **Fit in Window:** Automatically chooses a size for the diagram so that it fits on the canvas.
- **Auto Layout**: Organizes the objects and displays the diagram at its default size.
- **Center:** Centers the diagram on the canvas.
- **Show Full Impact**: Generates the full impact diagram of the selected object.
- **Show Full Lineage**: Generates the full lineage diagram of the selected object.
- **Show Lineage**: Displays the next level of objects in the lineage diagram of the selected object.
- **Hide Lineage**: Hides the lineage of the selected object.
- **Show Impact**: Displays the next level of objects in the impact diagram of the selected object.
- **Hide Impact**: Hides the impact of the selected object.
- **Expand**: Expands the selected icon in the diagram. The expanded icon shows the details of the object, such as the columns in a table or the attributes in a dimension.
- **Expand All**: Expands all the object icons in the diagram.
- **Collapse**: Collapses the expanded icon for a selected object.
- **Collapse All**: Collapses all the expanded icons on the canvas.

- **Refresh**: Refreshes the Dependency Manager diagram to reflect recent changes in the workspace.

Window

The Metadata Dependency Manager Window menu contains commands for toggling between displaying and hiding the following windows:

- **Property Inspector**
- **Bird's Eye View**
- **Tree View**

Toolbars

The Metadata Dependency Manager provides two toolbars as shortcuts to frequently used commands:

- **Graphic toolbar**: Provides icons for commands on the View menu. Refer to "View" on page 7-5 for descriptions of these commands.
- **Edit toolbar**: Provides icons for commands on the Edit menu. Refer to "Edit" on page 7-4 for descriptions of these commands.

Bird's Eye View

Use the Bird's Eye View to quickly change the portion of the diagram currently displayed on the canvas. This view displays a miniature version of the diagram on the canvas, with a scrollable box that represents the dimensions of the canvas. Drag the box to the area of the diagram currently of interest to you.

DM Tree

Use the DM Tree to change the content of the canvas. The DM Tree has these tabs:

- **DM Context Tree**: Lists objects in the current project. Right-click an object, and use the menu to generate a new diagram or to add the object to the current diagram. You can also drag-and-drop an object onto the current diagram.
- **DM Graph Tree**: Lists the current diagrams. You can close a diagram on the canvas, then use this list to redisplay the diagram without regenerating it.

Property Inspector

Use the Property Inspector to view an object's properties.

To change the current object in the Property Inspector, right-click the object on the canvas and select **Update Property Inspector**.

For a description of a property, select the property in the Inspector. The description appears at the bottom of the window.

Canvas

Use the canvas to display one or more lineage and impact diagrams. Each diagram is displayed on a separate tab.

You can use these techniques to create and manipulate diagrams on the canvas:

- To open a new diagram, right-click an object in the DM Context Tree and select **Show Lineage** or **Show Impact**.
- To close a diagram, click the **X** on the tab. You can redisplay the diagram from the DM Graph Tree until you close the Metadata Management window.

- To add an object to the canvas, drag-and-drop it from the DM Context Tree.
- To display more of the lineage of an object, click the plus sign (+) to the left of its icon. To hide the lineage, click the minus sign (-) to the left.
- To display more of the impact of an object, click the plus sign (+) to the right of its icon. To hide the impact, click the minus sign (-) to the right.
- To display the details of an object, double-click it. You can then select an individual property and edit it by choosing **Propagate Changes** from the Edit menu.
- To edit an object, right-click its icon and select **Open Editor**.

Metadata Dependencies and Lineage and Impact Analysis

OWB provides the Metadata Dependency Manager to detect and resolve the impact of the changes made to the object definitions or the metadata in the workspace. The Metadata Dependency Manager generates lineage and impact diagrams for any data object. A lineage diagram traces the data flows for an object back to the sources and displays all objects along those paths. An impact diagram identifies all the objects that are derived from selected object.

This type of information can help you in many circumstances as follows:

- Comply with government regulations or other audits that require tracking of financial data.
- Modify the system design because of changes in the source data.
- Modify the system design because of new requirements for reporting and analysis.
- Assess the impact of design changes in a pluggable mapping that is used throughout a project.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
>
> - "About the Metadata Dependency Manager"
> - "Opening an LIA Diagram"

Lineage and Impact Analysis Diagrams

Lineage and Impact Analysis (LIA) diagrams show the relationships among objects managed by OWB. These relationships are constructed by mappings and structural relationships (for example, a primary key and foreign key relationship). The lineage diagram for a particular object shows its source objects, and the impact diagram shows its targets.

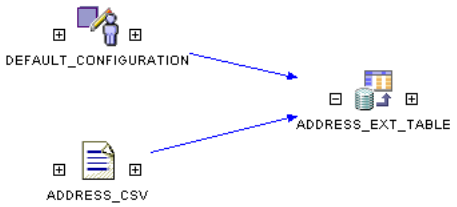
Lineage and impact are mirror images of each other. If Object A is part of the lineage diagram of Object B, then Object B is part of the impact diagram of Object A. When you read a diagram from left to right, you are seeing impact. When you read it from right to left, you are seeing lineage.

> **Tip:** "Managing and Exploring Objects in an LIA Diagram" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

For example, you might have a mapping that extracts data from a file and loads it into a table by way of an external table. This is the relationship:

```
flat_file > external_table > table
```

Figure 7–1 Lineage Analysis Diagram for ADDRESS_EXT_TABLE



Lineage Analysis Diagram: The lineage diagram for ADDRESS_EXT_TABLE includes DEFAULT_CONFIGURATION and ADDRESS_CSV.

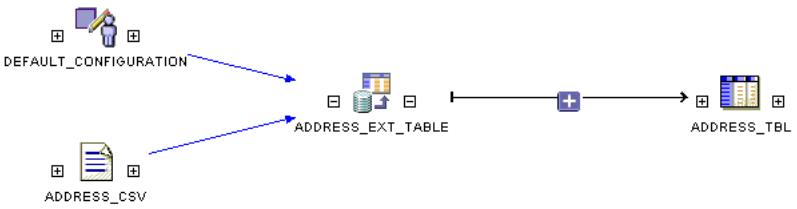
****

Figure 7–2 Impact Analysis Diagram for ADDRESS_EXT_TABLE

Impact Analysis Diagram: The impact diagram for ADDRESS_EXT_TABLE includes ADDRESS_TBL. A mapping is between the two objects.

****

Figure 7–3 Lineage and Impact Analysis Diagram



Lineage and Impact Analysis Diagram: DEFAULT_CONFIGURATION and ADDRESS_CSV impact ADDRESS_EXT_TABLE, which impacts ADDRESS_TBL. The lineage for ADDRESS_TBL is ADDRESS_EXT_TABLE, and its lineage is DEFAULT_CONFIGURATION and ADDRESS_CSV.

****

8

Data Quality Management

This section discusses data quality management and data profiling. Through data management capabilities, OWB ensures consistent, dependable data quality. Data profiling is the first step for any organization to improve information quality and provide better decisions.

This section contains these topics:

About Data Quality Management Processes and Phases

OWB offers a set of features that assist you in creating data systems that provide high quality information to your business users. With OWB you can implement a process that assesses, designs, transforms, and monitors data quality. The aim of building a data warehouse is to have an integrated, single source of data that can be used to make business decisions. Since the data is usually sourced from a number of disparate systems, it is important to ensure that the data is standardized and cleansed before loading into the data warehouse.

Using OWB for data management provides the following benefits:

- Provides an end-to-end data quality solution
- Enables you to include data quality and data profiling as an integral part of your data integration process.
- Stores metadata regarding the quality of your data alongside your data definitions.
- Automatically generates the mappings that you can use to correct data. These mappings are based on the business rules that you choose to apply to your data and decisions you make on how to correct data.

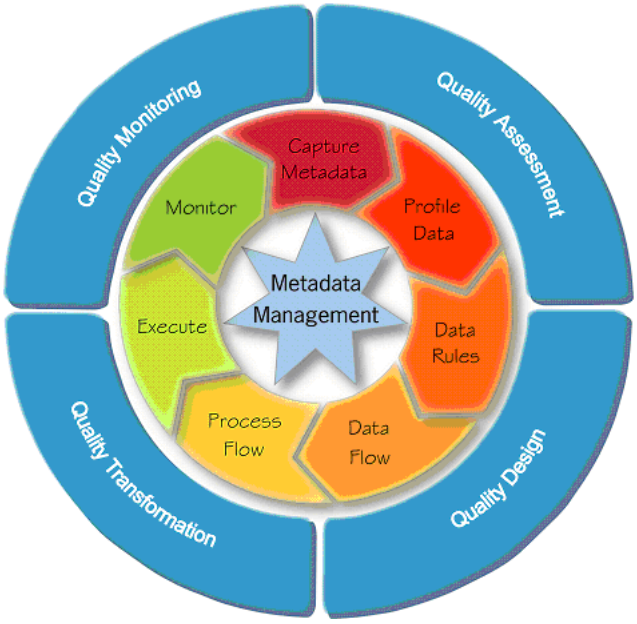
The rest of this section is devoted to discussing the phases of implementing and using data quality processes.

Phases in the Data Quality Lifecycle

Ensuring data quality involves the following phases:

- Quality Assessment
- Quality Design
- Quality Transformation
- Quality Monitoring

Figure 8–1 Phases Involved in Providing Quality Information



This screenshot shows the Data Quality Process. The outermost circle is divided into four parts. They are:

- Quality Assessment
- Quality Design
- Quality Transformation
- Quality Monitoring

The inner circle is divided into seven parts. They are:

- Capture Metadata
- Profile Data
- Data Rules
- Data Flow
- Process Flow
- Execute
- Monitor

In the Center of the inner circle is a star that represents metadata management.

****

Quality Assessment

In the quality assessment phase, you determine the quality of the source data. The first step in this phase is to import the source data, which could be stored in different sources, into OWB. You can import metadata and data from both Oracle and non-Oracle sources.

After you load the source data, you use data profiling to assess its quality. Data profiling is the process of uncovering data anomalies, inconsistencies, and redundancies by analyzing the content, structure, and relationships within the data. The analysis and data discovery techniques form the basis for data monitoring. For a quick summary of data profiling, see "Data Profiling: Assessing Data Quality" on page 8-3.

Quality Design

The quality design phase consists of designing your quality processes. You can specify the legal data within a data object or legal relationships between data objects using data rules. For more information about data rules, see "Data Rules: Enforcing Data Quality" on page 8-4.

Quality Transformation

As part of the quality design phase, you also design the transformations that ensure data quality. These transformations could be mappings that are generated by OWB as a result of data profiling or mappings you create. The quality transformation phase consists of running the correction mappings you designed to correct the source data.

Quality Monitoring

Data monitoring is the process of examining warehouse data over time and alerting you when the data violates business rules set for the data. For more information about data monitoring, see "Data Auditors: Monitoring Data Quality" on page 8-4.

Data Profiling: Assessing Data Quality

Data profiling enables you to assess the quality of your source data before you use it in data integration scenarios and systems. OWB provides the Data Profile Wizard to guide you through creating a data profile, and the Data Profile Editor to configure and manage data profiles. Because data profiling is integrated with the ETL features in OWB and other data quality features, such as data rules and built-in cleansing algorithms, you can also generate data cleansing mappings and schema correction scripts. This enables you to automatically correct any inconsistencies, redundancies, and inaccuracies in both the data and metadata.

Data profiling enables you to discover many important things about your data. Some common findings include:

- A domain of valid product codes
- A range of product discounts
- Columns that hold the pattern of an e-mail address
- A one-to-many relationship between columns
- Anomalies and outliers within columns
- Relations between tables even if they are not documented in the database

To begin the process of data profiling, you first use the Data Profile Wizard to create a data profile from within the Design Center. You then use the Data Profile Editor to run data profiling on the objects contained in the data profile, and to create correction tables and mappings.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:
>
> - "Overview of Data Profiling," which includes the sources that OWB supports for data profiling and a section on the Data Profile Editor
> - "Performing Data Profiling"
> - "Configuring Data Profiles"

Data Rules: Enforcing Data Quality

A **data rule** is a definition of valid data values and relationships, which determine legal data within a table or legal relationships between tables. Data rules can be applied to tables, views, dimensions, cubes, materialized views, and external tables. They are used in many situations including data profiling, data and schema cleansing, and data auditing.

The metadata for a data rule is stored in the workspace. To use a data rule, you apply the data rule to a data object. For example, you could create a data rule called `gender_rule`, which could specify that valid values are 'M' and 'F'. You could then apply this data rule to the `emp_gender` column of the `Employees` table. Applying the data rule ensures that the values stored for the `emp_gender` column are either 'M' or 'F'. You can view the details of the data rule bindings on the Data Rule tab of the Data Object Editor for the `Employees` table.

A data rule can be derived from the results of data profiling, or it can be created using the Data Rule Wizard or OMB*Plus scripting commands.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:
>
> - "Overview of Data Rules"
> - "Using Data Rules"
> - "Deriving Data Rules From Data Profiling Results"

Data Auditors: Monitoring Data Quality

OWB provides a way to create custom **data auditors**, which are processes that provide data monitoring by validating data against a set of data rules to determine which records comply and which do not. Data auditors gather statistical metrics on how well the data in a system complies with a rule by auditing and marking how many errors are occurring against the audited data. The monitoring process builds on your data profiling and data quality initiatives.

- Data auditors have thresholds that allow you to create logic based on the fact that too many non-compliant records can divert the process flow into an error or notification stream. Based on this threshold, the process can choose actions. In addition, the audit results can be captured and stored for analysis purposes.
- Data auditors can be deployed and executed ad-hoc, but they are typically run to monitor the quality of the data in an operational environment like a data

warehouse or ERP system. Therefore, they can be added to a process flow and scheduled. When run, the data auditor sets several output values. One of these output values is called the audit result.

- Data auditors also set the actual measured values such as Error Percent and Six Sigma values. Data auditors are a very important tool in ensuring that data quality levels are up to the standards set by the users of the system. It also helps determine spikes in bad data allowing events to the tied to these spikes.

  See Also: These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for procedures for using data auditors:

  - "Monitoring Data Quality Using Data Auditors"
  - "Configuring Data Auditors"

OWB Data Profiling Features

Using the data profiling features in OWB enables you to:

- Profile data from any source or combination of sources that OWB can access.
- Explore data profiling results in tabular or graphical format.
- Drill down into the actual data related to any profiling result.
- Derive data rules, either manually or automatically, based on the data profiling results.
- Derive quality indices such as six-sigma valuations.
- Profile or test any data rules you want to verify before putting in place.

  See Also: "Performing Data Profiling" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for detailed information and procedures

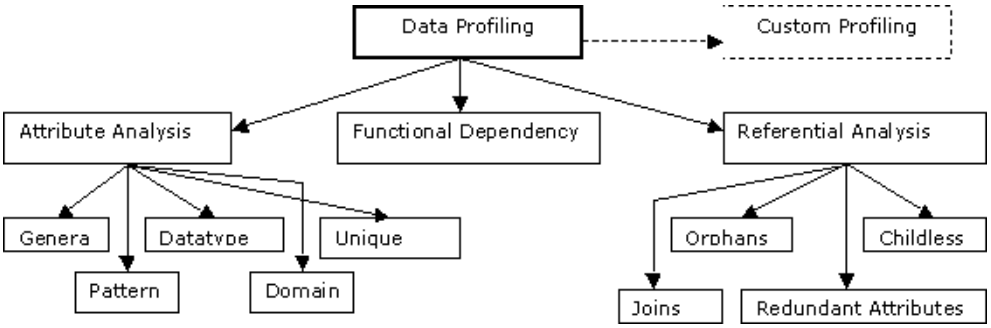
Types of Data Profiling

The selection of data objects determines which aspects of that data that you can profile and analyze.

Data profiling offers these main types of analysis:

- Attribute Analysis
- Functional Dependency Analysis
- Referential Analysis
- Custom Profiling with Data Rules

Figure 8–2 Data Profiling Overview



This illustration shows the types of analysis performed in each type of data profiling. At the top is a box labeled Data Profiling. Below this box are three boxes labeled, from left to right, Attribute Analysis, Functional Dependency, and Referential Analysis. Arrows connect Data Profiling to each of these boxes.

At the bottom of the image are two rows of boxes. The first row contains the following boxes, from left to right: General, Datatype, Unique, Orphans, and Childless. In the next row are the following boxes, from left to right: Pattern, Domain, Joins, and Redundant Attributes. Arrows connect the box labeled Attribute Analysis to the following boxes: General, Datatype, Unique, Pattern, and Domain. Arrows connect the box labeled Referential Analysis to the following boxes: Orphans, Childless, Joins, and Redundant Analysis.

****

Attribute Analysis

Attribute analysis seeks to discover both general and detailed information about the structure and content of data stored within a given column or attribute.

Attribute analysis consists of:

- Pattern Analysis
- Domain Analysis
- Data Type Analysis
- Unique Key Analysis

Pattern Analysis

Pattern analysis attempts to discover patterns and common types of records by analyzing the string of data stored in the attribute. It generates several regular expressions that match many of the values in the attribute, and reports the percentages of the data that comply with each candidate regular expression. OWB can also search for data that conforms to common regular expressions, such as dates, e-mail addresses, telephone numbers and Social Security numbers.

Table 8–1 shows a sample attribute, Job Code, that could be used for pattern analysis.

Table 8–1 Sample Columns Used for Pattern Analysis

Job ID	Job Code
7	337-A-55
9	740-B-74

Table 8–1 (Cont.) Sample Columns Used for Pattern Analysis

Job ID	Job Code
10	732-C-04
20	43-D-4

Table 8–2 shows the possible results from pattern analysis, where D represents a digit and X represents a character. After looking at the results and knowing that it is company policy for all job codes be in the format of DDD-X-DD, you can derive a data rule that requires all values in this attribute to conform to this pattern.

Table 8–2 Pattern Analysis Results

Job Code	% Occurred
DDD-X-DD	75%
DD-X-D	25%

Domain Analysis

Domain analysis identifies a domain or set of commonly used values within the attribute by capturing the most frequently occurring values. For example, the Status column in the Customers table is profiled and the results reveal that 90% of the values are among the following values: "MARRIED", "SINGLE", "DIVORCED". Further analysis and drilling down into the data reveal that the other 10% contains misspelled versions of these words with few exceptions. Configuration of the profiling determines when something is qualified as a domain; therefore, be sure to review the configuration before accepting domain values. You can then let OWB derive a rule that requires the data stored in this attribute to be one of the three values that were qualified as a domain.

Data Type Analysis

Data type analysis enables you to discover information about the data types found in the attribute. This type of analysis reveals metrics such as minimum and maximum character length values as well as scale and precision ranges. In some cases, the database column is of data type `VARCHAR2`, but the values in this column are all numbers. Then you may want to ensure that you only load numbers. Using data type analysis, you can have OWB derive a rule that requires all data stored within an attribute to be of the same data type.

Unique Key Analysis

Unique key analysis provides information to assist you in determining whether or not an attribute is a unique key. It does this by looking at the percentages of distinct values that occur in the attribute. You might determine that attributes with a minimum of 70% distinct values should be flagged for unique key analysis. For example, using unique key analysis you could discover that 95% of the values in the `EMP_ID` column are unique. Further analysis of the other 5% reveals that most of these values are either duplicates or nulls. You could then derive a rule that requires that all entries into the `EMP_ID` column be unique and not null.

Functional Dependency Analysis

Functional dependency analysis reveals information about column relationships. This enables you to search for things such as one attribute determining another attribute within an object.

Table 8–3 shows the contents of the Employees table in which the attribute Dept. Location is dependent on the attribute Dept. Number. Note that the attribute Dept. Number is not dependent on the attribute Dept. Location.

Table 8–3 Employees Table

ID	Name	Salary	Dept Number	Dept Location
10	Alison	1000	10	SF
20	Rochnik	1000	11	London
30	Meijer	300	12	LA
40	John	500	13	London
50	George	200	13	London
60	Paul	600	13	London
70	Ringo	100	13	London
80	Yoko	600	13	London
90	Jones	1200	10	SF

Referential Analysis

Referential analysis attempts to detect aspects of your data objects that refer to other objects. The purpose behind this type of analysis is to provide insight into how the object you are profiling is related or connected to other objects. Because you are comparing two objects in this type of analysis, one is often referred to as the parent object and the other as the child object. Some of the common things detected include orphans, childless objects, redundant objects, and joins. Orphans are values that are found in the child object, but not found in the parent object. Childless objects are values that are found in the parent object, but not found in the child object. Redundant attributes are values that exist in both the parent and child objects.

Table 8–4 and Table 8–5 show the contents of two tables that are candidates for referential analysis. Table 8–4, " Employees Table (Child)" is the child object, which inherits from Table 8–5, " Department Table (Parent)", the parent object.

Table 8–4 Employees Table (Child)

ID	Name	Dept. Number	City
10	Alison	17	NY
20	Rochnik	23	SF
30	Meijer	23	SF
40	Jones	15	SD

Table 8–5 Department Table (Parent)

Dept. Number	Location
17	NY

Table 8–5 (Cont.) Department Table (Parent)

Dept. Number	Location
18	London
20	SF
23	SF
55	HK

Referential analysis of these two objects would reveal that Dept. Number 15 from the Employees table is an orphan and Dept. Numbers 18, 20, and 55 from the Department table are childless. It would also reveal a join on the Dept. Number column.

Based on these results, you could derive referential rules that determine the cardinality between the two tables.

Custom Profiling with Data Rules

In addition to attribute analysis, functional dependency analysis, and referential analysis, OWB offers data rule profiling. Data rule profiling enables you to create rules to search for profile parameters within or between objects.

This is very powerful as it enables you to validate rules that apparently exist and are defined by the business users. By creating a data rule, and then profiling with this rule you can verify if the data actually complies with the rule, and whether or not the rule needs amending or the data needs cleansing.

For example, the HR department might define a rule that states that Income = Salary + Bonus for the Employee table shown in Table 8–6. You can then catch errors such as the one for employee Alison.

Table 8–6 Sample Employee Table

ID	Name	Salary	Bonus	Income
10	Alison	1000	50	*1075 (error)*
20	Rochnik	1000	75	1075
30	Meijer	300	35	335
40	Jones	1200	500	1700

See Also: "Data Cleansing and Correction with Data Rules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Six Sigma Methodology

OWB provides Six Sigma results and metrics embedded within the other data profiling results to provide a standardized approach to data quality.

See Also: "Viewing Profile Results" under "Performing Data Profiling" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for information on Six Sigma results

What is Six Sigma?

Six Sigma is a methodology that attempts to standardize the concept of quality in business processes. It achieves this goal by statistically analyzing the performance of

business processes. The goal of Six Sigma is to improve the performance of these processes by identifying the defects, understanding them, and eliminating the variables that cause these defects.

Six Sigma metrics give a quantitative number for the number of defects for each 1,000,000 opportunities. The term "opportunities" can be interpreted as the number of records. The perfect score is 6.0. The score of 6.0 is achieved when there are only 3.4 defects for each 1,000,000 opportunities. The score is calculated using the following formula:

- Defects Per Million Opportunities (DPMO) = (Total Defects / Total Opportunities) * 1,000,000
- Defects (%) = (Total Defects / Total Opportunities)* 100%
- Yield (%) = 100 - %Defects
- Process Sigma = NORMSINV(1-((Total Defects) / (Total Opportunities))) + 1.5

  where NORMSINV is the inverse of the standard normal cumulative distribution.

Six Sigma Metrics for Data Profiling

When you perform data profiling, the number of defects and anomalies discovered are shown as Six Sigma metrics. For example, if data profiling finds that a table has a row relationship with a second table, the number of records in the first table that do not adhere to this row-relationship can be described using the Six Sigma metric.

Six Sigma metrics are calculated for the following measures in the Data Profile Editor:

- **Aggregation:** For each column, the number of null values (defects) to the total number of rows in the table (opportunities).
- **Data Types:** For each column, the number of values that do not comply with the documented data type (defects) to the total number of rows in the table (opportunities).
- **Data Types:** For each column, the number of values that do not comply with the documented length (defects) to the total number of rows in the table (opportunities).
- **Data Types:** For each column, the number of values that do not comply with the documented scale (defects) to the total number of rows in the table (opportunities).
- **Data Types:** For each column, the number of values that do not comply with the documented precision (defects) to the total number of rows in the table (opportunities).
- **Patterns:** For each column, the number of values that do not comply with the common format (defects) to the total number of rows in the table (opportunities).
- **Domains:** For each column, the number of values that do not comply with the documented domain (defects) to the total number of rows in the table (opportunities).
- **Referential:** For each relationship, the number of values that do not comply with the documented foreign key (defects) to the total number of rows in the table (opportunities).
- **Referential:** For each column, the number of values that are redundant (defects) to the total number of rows in the table (opportunities).

- **Unique Key:** For each unique key, the number of values that do not comply with the documented unique key (defects) to the total number of rows in the table (opportunities).
- **Foreign Key:** For each foreign key, the number of rows that are childless (defects) to the total number of rows in the table (opportunities).
- **Data Rule:** For each data rule applied to the data profile, the number of rows that fail the data rule to the number of rows in the table.

Automated Data Cleansing

OWB enables you to automatically create corrected data objects and correction mappings based on the results of data profiling. On top of these automated corrections that make use of the underlying OWB architecture for data quality, you can create your own data quality mappings to correct and cleanse source data.

> **See Also:** "Overview of Automatic Data Correction and Data Rules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Automatic Data Correction Based on Data Profiling Results

When you perform data profiling, OWB generates corrections for the objects that you profiled. You can then decide to create corrected objects based on results of data profiling. The corrections are in the form of data rules that can be bound to the corrected object.

> **See Also:** "Generating Corrections Based on Data Profiling Results" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Types of Corrections for Source Data

You can perform the following types of corrections on source data objects:

- **Schema correction**

  Schema correction creates scripts that you can use to create a corrected set of source data objects with data rules applied to them. The corrected data objects adhere to the data rules derived from the results of data profiling.

  The correction tables have names that are prefixed with TMP__. For example, when you profile the `EMPLOYEES` table, the correction table will be called `TMP__EMPLOYEES`.

- **Data Correction**

  Data correction is the process of creating correction mappings to remove anomalies and inconsistencies in the source data before loading it into the corrected data objects. Correction mappings enforce the data rules defined on the data objects. While moving data from the old "dirty" tables in the profile source tables into the corrected tables, these mappings correct records that do not comply with the data rules.

  The name of the correction mapping is the object name prefixed with M_. For example, the correction mapping for the `EMPLOYEE` table is called `M_EMPLOYEE`.

Quick Summary of Performing Data Correction

To perform data correction on source data, you specify the following information:

- Data Correction Actions

- Cleansing Strategies for Data Correction

> **See Also:** "Data Cleansing and Correction with Data Rules" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for complete procedures

Data Correction Actions

Based on the data profiling results, OWB derives a set of data rules that you can use to cleanse the source data. You can automatically generate corrections based on these data rules by performing data correction actions.

For each data rule derived as a result of data profiling, you must choose a correction action that specifies how data values that are not accepted due to data rule enforcement should be handled. The correction actions you can choose are:

- **Ignore:** The data rule is ignored and, therefore, no values are rejected based on this data rule.
- **Report:** The data rule is run only after the data has been loaded for reporting purposes. It is similar to the Ignore option, except that a report is created that contains the values that do not adhere to the data rules. This action can be used for some rule types only.
- **Cleanse:** The values rejected by this data rule are moved to an error table where cleansing strategies are applied. When you select this option, you must specify a cleansing strategy.

Cleansing Strategies for Data Correction

When you decide to automatically generate corrected objects based on data profiling results, you must specify how inconsistent data from the source should be cleansed before being stored in the corrected object. To do this, you specify a cleansing strategy for each data rule that is applied to the correction object. Error tables are used to store the records that do not conform to the data rule.

The cleansing strategy you use depends on the type of data rule and the rule configuration.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:
>
> - "Generating Corrections Based on Data Profiling Results" for details about strategies and procedures
> - "Cleansing and Transforming Source Data Based on Data Profiling Results" for procedures on deploying corrections

Match and Merge Operations

OWB provides general-purpose data matching and merging capabilities that can be applied to any type of data, including data deduplication features. **Matching** determines which records refer to the same logical data. Warehouse Builder provides a variety of match rules to compare records. Match rules range from a simple exact match to sophisticated algorithms that can discover and correct common data entry errors. **Merging** consolidates matched records into a single consolidated "golden standard" record based on survivorship rules called **merge rules** that you select or define for creating a merged value for each column.

See Also : "Matching, Merging, and Deduplication" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide* for complete procedures

9

Deployment and Execution

This section discusses the concepts of deploying design objects to a target schema, and executing the ETL logic defined in the deployed objects.

This section contains these topics:

- About Deploying and Executing Design Objects
- Overview of Deployment and Execution Procedures
- Runtime Auditing for Deployments and Executions
- About Scheduling Design Objects to Execute

About Deploying and Executing Design Objects

After you design your ETL and data quality processes, you deploy and execute the resulting design objects in order to implement the design in the target schema. The Control Center Manager provides a comprehensive deployment console for viewing and managing all aspects of deployment and execution. It provides access to the information stored in the active Control Center.

The following topics provide overviews of deploying and executing design objects:

- Deployment Concepts
- About Executing Design Objects

> **See Also:** "Deploying to Target Schemas and Executing ETL Logic" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Deployment Concepts

Deployment is the process of creating physical objects in a target location according to the logical objects in an OWB workspace.

For example, when you create a table using the Design Center, the metadata for this table is stored in the workspace. If the table described in your design does not already exist in the database schema referenced by the specified location, then you must create the table by deploying it. Similarly, after you design a PL/SQL mapping, you must generate code for it (which creates a PL/SQL package implementing the mapping logic), then deploy the generated code to the specified location, which loads the generated PL/SQL package to the referenced schema. You can deploy objects from within the Design Center, or use the Control Center Manager. You can also use OMB*Plus commands to deploy objects.

As soon as you define a new object in the Design Center, the object is listed in the Control Center Manager under its deployment location.

Deploying a mapping or a process flow includes these steps:

- Generate the PL/SQL, SQL*Loader, or ABAP script, if necessary.
- Register the required locations and deploy any required connectors. This ensures that the details of the physical locations and their connectors are available at runtime.
- Transfer the PL/SQL, XPDL, SQL*Loader, or **ABAP script** from the Design Center to the Control Center. (**XPDL** refers to XML Process Definition Language, a format standardized by the Workflow Management Coalition.)

Note:

- After deploying a mapping or a process flow, you must explicitly start the scripts, as described in "Starting ETL Jobs" *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*.
- Workspace users can deploy only those objects for which they have the `COMPILE` privilege. By default, his privilege is granted on all objects in the workspace. However, the workspace owner may have instituted a different security policy.

See Also: "Deploying Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Deployment Status: Viewing Results

After you deploy an object, OWB assigns a deployment status to it. You can view the deployment status in the Control Center Manager.

The status represents the result of the deployment as follows:

- **Not Deployed:** Indicates that the object has not yet been deployed to the target schema.
- **Success:** Indicates that the object has been successfully deployed to the target schema.
- **Warning:** Indicates that some warnings were generated during the deployment of the object. Double-click the status to view details about the warning.
- **Failed:** Indicates that deployment of the object failed. Double-click the job in the Control Center jobs window to view information about why the deployment failed. The Job Log in the Design Center also captures details that you can view.

About Executing Design Objects

Execution is the process of starting the ETL logic defined in the deployed objects. For example, you define a mapping that sources data from a table, performs transformations on the source data, and loads it into the target table. When you deploy this mapping, the PL/SQL code generated for this mapping is stored in the target schema. When you execute this mapping, the job for the ETL logic is started and the data is picked up from the source table, transformed, and loaded into the target table.

Overview of Deployment and Execution Procedures

During the lifecycle of a data system, you typically will take these steps in the deployment process to create your system and perform the execution process to move data into your system:

Step 1 Select a named configuration

The Control Center for the selected configuration specifies the execution environment for the objects.

Step 2 Deploy objects to the target location

You can deploy them individually, in stages, or all at once.

Step 3 Review the results of the deployment

If an object fails to deploy, then fix the problem and try again.

Step 4 Start the ETL process for the target location

Execute the mappings or process flows that contain the ETL logic for the objects.

Step 5 Revise the design of target objects, if needed

Accommodate user requests, changes to the source data, and so forth.

Step 6 Set the deployment action

Set the modified objects to Upgrade or Replace.

> **See Also:** These topics in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*:
>
> - "Steps in the Deployment and Execution Process"
> - "Deploying Objects"
> - "Starting ETL Jobs"

Runtime Auditing for Deployments and Executions

Whenever you deploy and execute a design object, auditing information is generated and stored for you to view and manage.

The auditing information is specific to the type of object:

- PL/SQL runtime auditing is the auditing information for OWB objects deployed using PL/SQL scripts. Data objects, mappings, and process flows use PL/SQL deployments.
- Heterogeneous runtime auditing refers to auditing information about objects that are deployed to an OC4J server or a heterogeneous database such as DB2 or SQL Server. This type of auditing includes auditing from more than one Control Center Agent (CCA). This type of auditing also includes information about Code Templates, Web services, and Code Template mappings.

You view and manage the audit information through these methods:

- Control Center Manager. The Control Center Jobs panel displays a list of the deployment and executions jobs. When you double-click a job, you can see the job details.

- Repository Browser or Heterogeneous Repository Browser. You can access audit information The Heterogeneous Repository Browser enables you to access auditing information on systems that do not have Oracle Warehouse Builder installed.
- Public Views. As an alternative to using the Repository Browser, you can access the same information through the public views. To do this, start a SQL*Plus session and query the public views. Refer to *Oracle Warehouse Builder Sources and Targets Guide* for a list of public views.

See Also: "Auditing Deployments and Executions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

About Scheduling Design Objects to Execute

OWB provides these scheduling options:

- Integrated job scheduling in OWB (This feature is available if you have the Enterprise ETL option)
- Scheduling with Oracle Enterprise Manager to run ETL jobs
- Integration with third party schedulers

Integration with Third-party Schedulers

Integration with third-party schedulers depends on the features of the third party scheduler. For example, mappings and process flows in OWB are PL/SQL packages, and if there is a way in a third party scheduler to invoke a PL/SQL package, then refer to the documentation for that third party scheduler. You can also expose a mapping or a process flow as a Web Service if you are using a product like Oracle BPEL that can invoke Web services as part of orchestrating complex processes.

See Also:

- "Scheduling ETL Jobs" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- "Publishing Warehouse Builder Objects" as Web Services in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*
- *Oracle Database Licensing Information*

10

Scripting and Automation

This section introduces OMB*Plus and Experts, the scripting and automation features that contribute to making Oracle Warehouse Builder an extensible platform.

This section contains the following topics:

- OMB*Plus Concepts
- Overview of Experts in OWB
- About Guided Assistance Experts

> **See Also:** *Oracle Warehouse Builder API and Scripting Reference* for complete information about OMB*Plus

OMB*Plus Concepts

The OMB*Plus scripting language is built around a Java extension of the Tcl language called Jacl. OMB*Plus provides Tcl-based basic language features such as variables, procedures, and control structures. Additionally, OWB-specific command syntax provides scripting commands for batch processing and for manipulating the user interface in OWB:

- The OMB-prefixed commands ('B' for batch component).
- The OMU-prefixed commands ('U' for UI component).

You can write syntactic constructs such as variable support, conditional and looping control structures, error handling, and standard library procedures. You can also use Tcl to perform tasks on the host that is running Design Center, such as file system I/O.

A key benefit of OMB*Plus is its extensibility. For example, you can execute SQL*Plus statements and use PL/SQL packages from OMB*Plus.

You can use the OMB*Plus scripting interface to:

- Perform complex actions directly in OWB, without launching the OWB client user interface.
- Define sets of routine operations to be executed in OWB.
- Query OWB repository metadata and report or act on the results.
- Perform OWB batch operations.
- Automate a series of conditional operations in OWB.

OMB*Plus is accessed as described in these topics:

- OMB*Plus from the Design Center

- OMB*Plus from the Command Line

See Also: *Oracle Warehouse Builder API and Scripting Reference*

OMB*Plus from the Design Center

From within OWB you can access OMB*Plus from the Design Center as follows:

- Select **View**, then **OMB*Plus**.

The Design Center displays the OMB*Plus panel.

Note:

- When running OMB*Plus from within the Design Center, OMU commands are disallowed. They can only be used when running OMB*Plus from command line mode.
- You cannot access `public_project` or its child objects from the OMB*Plus view. Instead, use OMB*Plus from the command line to access these objects.

OMB*Plus from the Command Line

You launch the OMB*Plus console according to the procedure specific to your operating system.

- OMB*Plus and Windows Systems
- OMB*Plus and Linux and UNIX

OMB*Plus and Windows Systems

To start OMB*Plus on Windows systems, you run OMB*Plus from the Start menu under the Warehouse Builder program group.

- Select **Start**, then **Warehouse Builder** and **Administration**, and then **OMB*Plus**.

To access the OWB repository, you use the `OMBCONNECT` statement.

OMB*Plus and Linux and UNIX

To start OMB*Plus on Linux and UNIX systems, you run `OMBPlus.sh` from the command prompt under the Oracle home.

To access the OWB repository, you use the `OMBCONNECT` statement.

Overview of Experts in OWB

Experts are mini-applications or task-flows that developers build to carry out specific tasks in OWB. Experts enable you to capture your knowledge, expertise, and best practices in OWB and make them repeatable. The most basic use for Experts is to guide a user through a sequence of related tasks in Design Center. Experts enable the re-use the graphical UI components and capabilities of OWB. For example, the Flat File Sampler Wizard in OWB may be re-used as a component within an Expert, where the developer needs the user to define the structure of a file. The developer can have full control over how the information and metadata that is returned by the wizard is used, and can call other components, write and embed scripts, call SQL or Java, and so forth.

Uses for Experts include:

- Encapsulating common practices, to ensure that developers implement tasks in a consistent fashion
- Automating the performance of complex tasks to increase developer productivity
- Creating user interfaces to functionality implemented as OMB*Plus scripts or in Java

Combined with the OMB*Plus scripting features and other extension mechanisms, Experts facilitate using OWB as a platform for delivering complex, reusable data integration solutions.

This section includes these additional topics about Experts:

- Design Environment for Experts
- Storage of Experts: Public and Private
- About Guided Assistance Experts

> **See Also:** The chapter "About Experts" in *Oracle Warehouse Builder API and Scripting Reference* for details and procedures for developing Experts

Design Environment for Experts

The design environment for Experts is in the context of projects in Design Center, primarily with the Expert Editor, which is similar to the environment for process flows: a graph in which task objects are represented by icons, flow of control is represented by connecting lines, and design activities are performed by drag-and-drop and setting up object properties. Tasks can include creating or modifying objects in your projects, invoking commonly used built-in dialogs, creating custom dialog boxes, and invoking OMB*Plus scripts for complex flow of control and interaction with OWB. Variables are available to preserve state among tasks in an Expert. Nested Experts are also supported, so that logic implemented in one Expert can be called in others.

Experts can also invoke arbitrary Java code in JAR files that you set up to be loaded with Warehouse Builder. As a result, you can implement new functionality in Java and integrate it into the Warehouse Builder user interface.

At runtime, Experts follow the same paradigm as traditional OWB functionality, in that code (in this case, OMB and OMU scripts) is generated from the model/logical design, and is then executed.

Storage of Experts: Public and Private

In a workspace, Experts can be created in a public Experts folder or in the private Experts folder that is included in each project. Private Experts are typically used only in the context of its parent project. As with other public objects, public Experts are stored in the globals area and can be used across all projects in the workspace. Public Experts typically encapsulate functionality that is useful across different types of projects.

Depending upon how they are implemented, Experts can be invoked in several different contexts:

- Some Experts are designed to be right-clicked and executed from the Experts or Public Experts folder.
- Experts can be added to the right-click menus associated with most objects in Warehouse Builder, as well as being added as new menu commands in Design

Center. In such a case, the Expert executes in a context where the object from which it was invoked is passed as a parameter.

- OWB can also be launched from the command line in a mode that hides Design Center from the user and runs a single Expert, then exits. These two capabilities allow the creation of simplified, constrained interfaces to OWB functionality for users who are not frequent OWB users or ETL developers.

About Guided Assistance Experts

The Guided Assistance feature enables the creation of a set of Experts in an OWB workspace, such that when a user logs in to the workspace, they are immediately offered a point-and-click list of Experts to choose from to perform one of several common tasks. Placing Experts in a folder called STARTUP in the Public Experts folder causes them to appear in the Guided Assistance Experts list. Guided Assistance Experts can also be accessed through a Help menu item in Design Center. Administrators can use this feature to provide developers on a project a set of Experts and encourage their use for consistency and time-saving on common tasks.

> **See Also:** The topic "Guided Assistance" under "About Experts" in *Oracle Warehouse Builder API and Scripting Reference*

A

OWB Preferences

This appendix describes the preferences available in OWB.

This appendix contains the following topics:

About OWB Preferences

OWB provides a set of user preferences that enable you to customize your user interface and environment. You set preferences from the Design Center by selecting an object and clicking **Design** on the Design Center menu.

Preferences are organized under categories, and each preference contains values that you set according to your design needs. You may also view and set preferences from the OWB client console menu under **Tools>Preferences**.

Appearance Preferences

The Appearance category contains the Locale preference. Use the **Locale** list to set the language you want the client text to display. This list displays the language options for the client. OWB prompts you to restart the computer in order to use the new language setting.

> **Note:** The Locale selection does not define the character set of the OWB repository. The Locale appearance preference only affects the text and menu options on the client user interface. The repository character set is determined by the settings in Oracle Database.

Control Center Monitor Preferences

Use the Control Center Monitor category to set preferences that control the display of components in the Control Center. When you use the Control Center to deploy or execute objects, the Job Details window displays the results of deployment or execution. The Control Center Monitor preferences enable you to control which items to display in the object tree of the Job Details window.

> **Note:** OWB displays the Job Details window only if you select the **Show Monitor** preference under the Process node of the Deployment preferences category.
>
> If this option is not selected, then you can view the Job Details window by double-clicking the **row** representing the deployment or execution job in the **Control Center Jobs** panel of the Control Center.

The Control Center Monitor category contains the following preferences:

- **Show Project:** Select this option to display the project name in the Job Details window object tree. When this option is selected, the object tree displays a node for the named project, and all the objects are displayed under this project node.
- **Show Module:** Select this option to display the name of the module to which the object being deployed or executed belongs in the Job Details window. When this option is selected, the object tree displays a node for the module. Expand the module node to view the object details.
- **Show Location:** Select this option to display the location name in the object tree of the Job Details window.
- **Show Action:** Select this option to display and access the actions that can be performed on the object in the object tree of the Job Details window. The actions that can be performed are: Create, Drop, Deploy, and Upgrade.
- **Show Type:** Select this option to display the type of object in the object tree of the Job Details window. When you select this option, a node is displayed for the type of object and the objects are listed under the respective nodes.

Data Profiling Preferences

Use the Data Profiling category to set the preferences for data profiling.

This category contains the following preferences:

- **Data Rule Folder Name:** Use this option to set the name of the folder that contains the data rules as a result of data profiling.
- **Default Profile Location:** Use this option to set the default location that is used to store the data profiling results. You can override this setting by selecting a different location as your profile location. In the Data Profile Editor, from the Edit menu, select **Properties**. Use the Data Locations tab to change the default profile location.

Deployment Preferences

The Deployment category enables you to set deployment preferences such as displaying the deployment monitor, prompting for execution parameters, and showing completion messages. This enables you to control some of the popup windows that are displayed by the Control Center Manager during object deployment.

Deployment preferences are divided into two sections: Process and Monitor. Expand the **Deployment** node in the Preferences dialog box. Click the node for which you want to set preferences.

Process

The Process node sets the following deployment options:

- **Pause After Compile:** Select this option to pause deployment after script generation. This means that you must explicitly deploy an object after it is successfully generated.
- **Prompt for Commit:** Select this option to prompt the user to commit design time changes before a deployment. When you deploy objects from the Design Center, if there are any unsaved design changes, then OWB prompts you to save these changes by displaying the OWB Warning dialog box. Click **Save** to commit unsaved design changes. Click **Cancel** to terminate the deployment.

  If you do not set this option, then OWB saves any design changes before the deployment job.
- **Prompt for Job Name:** Select this option to prompt the user for the name of a deployment job. When this option is not selected, OWB assigns a default job name.
- **Prompt for Execution Parameters:** Select this option to prompt the user for the values of execution parameters. If you do not select this option, then OWB uses the default value of parameters during the execution. OWB does not prompt you to provide the parameter values.
- **Show Monitor:** Select this option to display the Job Details window when you deploy or execute an object. This dialog box displays details of the objects being deployed, deployment progress, and deployment status.
- **Show Deployment Completion Message:** Select this option to display an alert indicating that the deployment job has completed.
- **Show Design Center Deployment Job:** Select this option to display the Control Center Jobs dialog box when you deploy an object from the Design Center. The Control Center Jobs dialog box, which is similar to the Jobs panel of the Control Center Manager, contains the Deployment, Execution, and Scheduled tabs. Use this option to view the status of a deployment job while deploying using the Design Center.

Monitor

The Monitor node sets the following deployment monitoring options:

- **Show Monitor Tree:** Select this option to show the Job Details window when you perform a deployment or execution.
- **Show Monitor Results:** Select this option to display the deployment or execution results in Control Center Manager.
- **Show Monitor Logfile:** Select this option to display the log file in the Control Center Manager.

Environment Preferences

The Environment category enables you to set generic preferences that control the client environment such as displaying welcome pages for wizards and recycle bin preferences.

You can set the following environment preferences:

- **Recycle Deleted Objects:** Select this option to move deleted objects to the recycle bin. If this option is not selected, any objects you delete are lost and you have no way of recovering them.
- **Allow Optimize Repository Warning on Startup:** Select this option to collect schema statistics when you log in to OWB. Collecting schema statistics improves repository performance. If this option is selected, at log on, OWB determines if statistics must be gathered for the repository schema. If statistics must be gathered, a warning dialog box is displayed asking if you want to gather statistics now. Click **Yes** to collect schema statistics and optimize the repository.

  If you do not select this option, you can still collect schema statistics from the Design Center by selecting **Optimize Repository** from Tools menu.
- **Show Delete Confirmation Dialog Box:** Select this option to display a dialog box that asks for a confirmation before deleting an object. When this option is selected, if you delete an object, the OWB Warning dialog box is displayed. Click **Yes** to delete the object. Click **No** to cancel the Delete operation and retain the object.
- **Hide All Wizard Welcome pages:** Select this option to hide the welcome page of all wizards. Every wizard in OWB starts with a Welcome page that summarizes the steps to follow to complete that task. To display the Welcome page for all wizards, deselect this preference.
- **Empty Recycle Bin on Exit:** Select this option to empty the contents of the recycle bin when you exit the OWB client. Deselect this option to save the recycle bin objects across sessions.
- **Default Workspace:** Select this option to set the default workspace for the logged in user to use. A drop-down list shows the workspaces for which the logged in user is registered. The initial setting is for the first workspace that you own or for which you were registered.

Note: You can access public views from your default workspace. From your default workspace, when you log in to SQL*Plus, you can access public views (design-time public views or runtime public views) without needing to call a separate procedure.

If you try to access public views from any workspace other than the default, then you must call: `WB_workspace_management.set_workspace(<wksp_name>, <wksp_owner>)`.

If you want to switch to a workspace other than the default one, then you can call the `WB_workspace_management.set_workspace` procedure. You must have the `ACCESS_PUBLICVIEW_BROWSER` system privilege to retrieve useful information from the pubic views. Otherwise, you will get "0 rows returned." You may need to ask the workspace owner or workspace admin to grant the system privilege `ACCESS_PUBLICVIEW_BROWSER`.

Generation/Validation Preferences

The Generation/Validation category enables you to set preferences related to generation and validation of OWB design objects. Use these preferences to control what is displayed in the Generation Results dialog box or Validation Results dialog box. These dialog boxes are displayed when you generate or validate an object from the Design Center.

The Generation/Validation category contains the following preferences:

- **Show Project:** Select this option to display the project node in Validation Results dialog box or the Generation Results dialog box.
- **Show Module:** Select this option to display the module node in Validation Results dialog box or the Generation Results dialog box.
- **Show Location:** Select this option to display the location node in Validation Results dialog box or the Generation Results dialog box.
- **Show Action:** Select this option to display the action node in Validation Results dialog box or the Generation Results dialog box.
- **Show Type:** Select this option to display the type node in Validation Results dialog box or the Generation Results dialog box.

Code Template Editor Preferences

The Code Template Editor preferences provide settings for how the editor will be laid out on screen. This enables management of screen real estate while you are editing a Code Template.

- **Vertical Layout Horizontal Spacing:** Specify the screen width for vertical layout. The default is 80.
- **Horizontal Layout Horizontal Spacing:** Specify the screen width for horizontal layout. The default is 25.
- **Direction of Auto Layout:** Specify how the editor will automatically be laid out on screen when opened. The default is Vertical.
- **Vertical Layout Vertical Spacing:** Specify the screen height for vertical layout. The default is 5.
- **Horizontal Layout Vertical Spacing:** Specify the screen height for horizontal layout. The default is 40.

Logging Preferences

The Logging Preferences category enables you to set log file options such as file location, file size, and types of messages saved to any log file. The log file contains messages relating to your design process. By default a message log is saved to the default location.

The Logging Preferences category contains the following preferences:

- **File Path:** This option specifies the location where the log files are saved. Enter the complete path or use the Browse button to select the location. The default location is `<OWB_ORACLE_HOME>\owb\bin\admin`.
- **File Name:** This option specifies the name of the log file. Do not include a file extension when you specify a file name.
- **Maximum Size (Kb):** This option specifies the maximum file size for the log file(s) in Kb. There are two log files: `<logfilename>.0`, and `<logfilename>.1`. When the maximum size of the first log file `<logfilename>.0` is reached, OWB starts writing to the second log file, `<logfilename>.1`. When the maximum size of the second one is reached, OWB starts overwriting the first log file.
- **Log Error Messages:** Select this option to write all error messages to the log file.
- **Log Warning Messages:** Select this option to write all warning messages to the log file.

- **Log Information Messages:** Select this option to write all information messages to the log file.

Naming Preferences

The Naming Preferences category enables you to set naming preferences by selecting whether you want to view objects in Business Name or Physical Name mode. You can also set how you want to propagate object name changes, and whether you want to synchronize names with the target.

The Naming Preferences category contains the following preferences:

- **Naming Mode:** Selects whether to display objects using their physical or business names.
- **Propagate Name Changes:** Propagates name changes from the current naming mode to the other naming mode (*physical* to *business* naming mode, and vice versa.)
- **Synchronize:** Selects whether to synchronize names with the target object, or to keep names as-is.

About Naming Modes

OWB maintains a business name and a physical name for each object stored in the repository. A business name is a descriptive logical name for an object. A physical name is the actual object name recognized by the repository.

When you generate DDL scripts for a named object, the physical names are used. Physical names must conform to the syntax rules for basic elements as defined in the *Oracle Database SQL Language Reference*.

Names must be unique within their category:

- Module names must be unique within a project.
- Warehouse object names must be unique within a warehouse module. This includes the names of tables, dimensions, cubes, mappings, materialized views, sequences, views and indexes.
- Transformation names must be unique within a transformation package.

Business Name Mode Use Business Name mode to create a business name for an object or to change the business name of an existing object. When this mode is selected, OWB editors, wizards, and property sheets display the business names of objects.

A business name must conform to these rules:

- The length of a name cannot exceed 200 characters.
- The name must be unique within its category.
- All source modules reflect the case of the imported source and are subject to the double-quotes rules as defined in the *Oracle Database SQL Language Reference*.

Copy operations from a source to a target in a mapping are not case-sensitive.

When you create a business name, OWB generates a corresponding, valid physical name that resembles the business name. If you create a business name that duplicates an existing physical name, then OWB appends an underscore and a number in order to create a unique name.

Physical Name Mode Use Physical Name mode to create a physical name for an object or to change the physical name of an existing object. When this mode is selected, OWB editors, wizards, and property sheets display the physical names of objects. Physical names are converted to UPPERCASE.

An object's physical name must conform to these rules:

- Contain no more than 30 characters.
- Conform with the basic syntax rules for schema objects as defined by *Oracle Database SQL Language Reference*.

> **Note:** A collection can have a physical name containing up to 200 characters.

OWB prevents you from entering an invalid physical name. For example, you cannot enter a duplicate name, a name with too many characters, nor a name that is a reserved word.

Setting the Name Mode To create or change a business name for an object, OWB must be in Business Name mode. To create or change a physical name for an object, OWB must be in Physical Name mode.

The default naming preferences for OWB are as follows:

- **Mode:** The default setting for the mode is physical name mode.
- **Propagation:** The default propagation setting is to propagate physical name to business name.

Icons for the name mode and name propagation settings are located in the lower-left corner of the editors. These icons indicate the current naming preference setting.

OWB saves your naming preferences across sessions. The Name Mode preference is stored in a file on the client workstation. If you use OWB from another workstation, your preferences may be different.

> **See Also:** *Oracle Database SQL Language Reference*

Security Parameters

> **IMPORTANT: Only administrators can edit the security preferences.**

The Security Parameters category contains the following settings:

- Persist Location Password in Metadata
- Share Location Password During Runtime
- Default Metadata Security Policy

Persist Location Password in Metadata

This option determines whether or not location passwords are persisted across OWB design sessions.

By default, this option is deselected, which is the more secure setting. OWB retains location passwords for the length of the design session only. That is, the first time you start tools such as the Data Viewer or Debugger, you must enter the appropriate location passwords.

If this option is selected, then OWB persists encrypted versions of location passwords in the workspace. The result is that you can start tools such as the Data Viewer and Debugger without entering passwords each time.

> **See Also:** "Encrypting Passwords to Warehouse Builder Locations" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and UNIX* for more information about the encryption methodology

Share Location Password During Runtime

This parameter determines whether or not the location passwords that users enter during the design phase can be shared with other users. For example, assume that user `Dev1` designs mapping `MAP1`. To access the sources and targets for this mapping, `Dev1` defines the locations to each source and target, including a username and password. When other users subsequently attempt to execute `MAP1` or view data associated with it, the OWB preference determines whether or not each user must enter the password each time in the Design Center, or the first time in the Control Center.

Share Location Password During Runtime works in conjunction with Persist Location Password in Metadata. The most secure mode, and the default behavior, is for both options to be deactivated. In this case, each user, including `Dev1`, must enter their password once for each Design Center session and for the first time they attempt to use that location in the Control Center. Depending on your security requirements, you may want each user to define their own location for a given source or target

If both Share Location Password During Runtime and Persist Location Password in Metadata are activated, then any user can access a schema given that any user previously defined the location. Therefore, user `Oper2` can execute `MAP1` given that `Dev1` or any other user previously defined the location with valid credentials.

Default Metadata Security Policy

This parameter specifies the default security policy to be applied. Minimum security enables all users full control over objects that any newly registered user creates. Maximum security, however, restricts access to the newly registered user who created the object, and to OWB administrators.

This setting is not retroactive. That is, if you change this setting in an existing OWB implementation, then the setting does not affect existing users and existing objects. You must manually change the security settings on existing objects.

> **See Also:** "Managing Security" in *Oracle Warehouse Builder Installation and Administration Guide for Windows and UNIX* for more information about manually changing the security settings

B

Object Editors

This appendix provides information about the Graphical Navigator and object editors in OWB.

This appendix contains the following topics:

The Graphical Navigator and Editors

In design mode, OWB provides the Graphical Navigator from which you can launch object and document editors. Objects from the component palette and tree panel can be dropped onto this navigator. The OWB object editors also support keyboard navigation, including tabbing and shortcuts.

When you double-click an object such as a table, a document panel opens containing one or more viewers and editors that are specific to that object type. These editors appear as tabs at the bottom of the document window. When you select an object, a property inspector and menu options are available for that object, and you can include related objects. Additionally, you can drag and drop appropriate objects from the Navigator tree to the Graphical canvas and then relate these objects with rubber bands.

> **See Also:** "Using the Graphical Navigator" in *Oracle Warehouse Builder Sources and Targets Guide*

Data Object and Document Editors

This sections provides a list of the various editors in OWB:

- Item Folder Editor

Cube Editor Tabs

- Cube Name. Contains fields for the name and description for the cube.
- Cube Storage. Supports the specification of the storage type (ROLAP or MOLAP) and associated details. Changing the storage type from ROLAP to MOLAP, or vice versa, affects the Cube Dimensions Editor, Cube Measures Editor and Cube Aggregation Editor.
- Cube Dimensions. Supports the specification of the cube's dimension references. For a MOLAP cube, the panel has an additional Advanced button, which allows sparsity, partitioning, and indexing to be specified.
- Cube Measures. Supports the specification of the cube's measures. For a MOLAP cube, the panel has an additional Generate Calculated Measures button, which may be used to specify which calculations to generate for each measure.
- Cube Aggregation. Supports the specification of aggregation options for the cube. The options presented depend on whether the storage type is ROLAP or MOLAP.
- Physical Binding. For cubes, a physical binding tab graphically shows the object and the objects it is bound to. From here, you can open additional editors for the cube objects. Double clicking a cube and selecting Physical Bindings opens a graphical view of the cube with links to the bound objects. A property inspector shows the properties for the selected cube. (Operations on this tab are only available if the manual binding property has been set.)
- Cube Data Viewer. Invokes the Data Viewer for the cube.

> **See Also:** "Creating Cubes" and "Editing Cube Definitions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Dimension Editor Tabs

- Dimension Name. Contains fields for the name and description for the dimension. It also supports specification of any dimension roles. For a time dimension, it supports additional fields specifying the start year and number of years supported by the time dimension.
- Dimension Storage. Supports the specification of the storage type (ROLAP or MOLAP) and associated details.
- Dimension Attributes. supports the specification of the Dimension's Attributes. The panel has an additional field to specify the Sequence to be used to populate Dimension keys.
- Dimension Levels. Supports the specification of the Dimension's Levels, including the Level Attributes available at each Level.
- Dimension Hierarchies. Supports the specification of the Dimension's Hierarchies, including the Levels available in each Hierarchy. The panel for a Time Dimension displays additional "Fiscal Settings" and "Create Map" buttons.
- Dimension SCD. The Dimension SCD Editor supports the specification of Slowly Changing Dimensions. This panel is not present for Time Dimensions. (Also, OWB 11.1 does not allow these details to be updated for MOLAP Dimensions.)
- Physical Binding. For dimensions, the physical binding tab graphically shows the object and the objects it is bound to.
- Dimension Data Viewer. Invokes the Data Viewer for the Dimension.

See Also: "Creating Dimensions" and "Editing Dimension Definitions" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Table Editor

These tabs are available for table objects.

- Table Name
- Table Columns
- Table Constraints
- Table Indexes
- Table Partitions
- Table Attribute Sets
- Table Data Rules

See Also: "Defining Tables" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Data Viewer

These tabs in the Data Viewer are available for viewing objects.

- Structure Panel.
- View Name.
- View Columns
- View Query
- View Constraints
- View Attribute Sets
- View Data Rules
- View Data Viewer

See Also: "Using the Data Viewer to View Data Stored in Data Objects" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Materialized View Editor

These editor tabs are available for materialized view objects.

- Structure Panel.
- Materialized View Name Editor.
- Materialized View Columns Editor
- Materialized View Query Editor
- Materialized View Constraints Editor
- Materialized Indexes Editor
- Materialized View Partitions Editor
- Materialized View Attribute Sets Editor

- Materialized View Data Rules Editor
- Materialized View Data Viewer Editor

> **See Also:** "Configuring Materialized Views" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Object Type Editor

Contains these tabs:

- Structure Panel.
- Object Type Name Editor
- Object Type Columns Editor

> **See Also:** "Defining Object Types" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

VArray Editor

Contains these tabs:

- Structure Panel. Displays the VArray structure with its objects.
- VARRAY Name Editor
- VARRAY Details Editor

> **See Also:** "Defining Varrays" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Nested Table Editors

- Structure Panel. Displays the nested table structure with its objects.
- Nested Table Name Editor
- Nested Table Details Editor

> **See Also:** "Defining Nested Tables" in *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*

Create Business Area Wizard and Business Area Editor

Contains these tabs:

- Structure Panel. Displays the business area structure with its objects: business name and item folders.
- Contents Editor. Shows the item folders that are in the business area.
- Business Area Name Editor.
- Business Area Item Folders Editor.

> **See Also:** "Creating a Business Area" and "Using the Create Business Area Wizard" in *Oracle Warehouse Builder Sources and Targets Guide*

Item Folder Editor

Contains these tabs:

- Structure Panel. Displays the item folder name and its contained items.

- Composition Editor. Displays the objects that are used in the construction of the Item Folder
- Item Folder Name Editor
- Item Folder Source Items Editor
- Item Folder Items Editor
- Item Folder Joins Editor
- Item Folder Conditions Editor

See Also: "Editing an Item Folder" in *Oracle Warehouse Builder Sources and Targets Guide*

Glossary

ABAP script

A script that can be generated in OWB that extracts and loads data from SAP systems.

activity

A unit of work for a process flow. See also **process flow**.

analytic workspace

A container within Oracle Database that stores data in a multidimensional format. Analytic workspaces provide the best support for OLAP processing.

Code Template (CT)

A cross-platform, reusable object that contains the information required to perform a specific set of tasks against a specific technology or set of technologies, for example data integration or data transformation tasks.

Code Template mapping

A mapping that contains an association with a Code Template. Typically used to extract or load data (both with and without transformations) from non-Oracle databases, such as IBM DB2 and Microsoft SQL Server.

connector

A logical link created by a mapping between a source location and a target location.

Control Center Agent (CCA)

The agent that runs the Code Templates in the Oracle Containers for J2EE (OC4J) server. You must start the Control Center Agent before you deploy Code Templates or CT mappings. Also referred to as the J2EE Runtime.

Control Center Manager

The graphical console of the Control Center Service for centrally viewing and managing all aspects of deployment and execution. Provides access to the information stored in the active configuration. Includes update capabilities to enable management of your data system's life cycle.

Control Center Service

A service that runs outside the database, which can monitor and execute things that cannot be run directly in the database, such as: PL*SQL scripts, SQL*Loader, and shell scripts. Enables deployment of OWB mappings and processes to targets (databases, etc.), and the execution of these mappings and processes.

control row

A row that links fact data to a dimension at any level.

CT mapping

See **Code Template mapping**.

cube

A data object that contains measures, and links to one or more dimensions. The axes of a cube contain dimension members, and the body of the cube contains measure values.

data auditors

Processes that provide data monitoring by validating data against a set of data rules to determine which records comply and which do not.

data rule

Metadata (as definitions) about data profiling results, which can be bound to the profiled data objects, and then be available in any context in which the profiled objects are used in ETL.

data transformation

A set of operations, which are specified in a mapping, that change source data into consistent, compatible output for a target.

DDL script

A script that can be generated in OWB that creates or drops database objects.

deployable parameter

The parameter for an object that specifies it is to be deployed. By default this parameter is selected. To prevent an object from being deployed, clear this parameter.

deployable parameter

The parameter for an object that specifies it is to be deployed. By default this parameter is selected. To prevent an object from being deployed, clear this parameter.

deployment

The process of creating physical objects in a target location according to the logical objects defined in an OWB workspace.

dimension

An object that contains additional metadata to identify and categorize data. Same as dimensional object. Can be a cube.

dimension attribute

A descriptive characteristic of a dimension member, having a name and a data type.

dimensional object

Refer to **dimension**.

ETL

The process of extracting data from its source location, transforming it as defined in a mapping, and loading it into target objects (or schemas). ETL stands for *extract*, *transform*, and *load*.

execution

The process of running the code for the ETL logic that is defined in the deployed objects in order to instantiate the logic within the objects.

Expert

Mini-applications or task-flows that perform a specific sequence of tasks in OWB.

flat files

Non-hierarchical, non-object-oriented file structures in plain text comma-separated or tab-separated format, ASCII format, or proprietary binary formats.

folders

Structures in which to organize all or some objects within a target module based on specific object characteristics. For example, you may create user folders to group tables based on their functionality (sales, marketing, administration and so forth).

hierarchy

A structure that uses ordered levels to organize data. OWB uses hierarchies to define relationships between adjacent levels in time dimensions.

item folder

A result set of data, similar to a database view. Item folders store information in the same manner as tables.

J2EE Runtime

Refer to **Control Center Agent (CCA)**. Also, sometimes called Java Runtime.

level attribute

For time dimensions, a descriptive characteristic of a level value.

location

Object that stores the connection information to the various files, databases, and applications that OWB accesses for extracting and loading data. Locations also store connection information to ETL management tools and Business Intelligence tools.

logical definition

The data objects created when you design a target schema.

mapping

An object that contains operations for extraction, transformation, and loading (ETL) that moves data from sources to targets.

mapping operator

The representation of an operation for a distinct task you want to perform in a mapping. For example, operations include extracting data, loading data, and transforming data.

match bin set

A match record set consisting of one or more similar records.

metadata

Data that describes the contents of objects in a data set.

module

A container object that appears in the Projects Navigator and that corresponds to a specific location in the Locations Navigator. A module can correspond to only one metadata location and data location at a time.

ODI

See **Oracle Data Integrator**.

OMB*Plus

The OWB scripting API, which is based on the Java implementation of Tcl called Jacl.

OMB

The OMB-prefixed commands ('B' for batch component) of OMB*Plus.

OMU

A subset of OMB*Plus that provides scripting commands for manipulating the user interface in OWB. Also OMU-prefixed commands ('U' for UI component).

operator

OWB contains pre-built operators for transformations, mappings, names and addresses, and so forth. Operators in OWB are customizable and take advantage of the library of PL/SQL functions, procedures, package functions, and package procedures for Oracle Database. See also **mapping operator**.

Oracle Data Integrator

Oracle Data Integrator (ODI). See *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

OWB Repository

The single, unified repository for the database instance, which is pre-seeded with a schema and database objects. The runtime environment and the design environment reside in this single repository. The repository schema, named OWBSYS, gets created when you install Oracle Database.

pluggable mapping

A reusable grouping of mapping operators that works as a single operator. Similar in concept to a function in a programming language.

process flow

An object that describes dependencies and activities between OWB mappings and external processes, applets, or applications. Process flows begin with a start activity and conclude with an end activity and can also start other process flows. Compare to *schedule*. See also **activity**.

process flow module

A container for a grouping of process packages.

process flow package

A container for a grouping of process flows.

project

The highest-level and largest object in the OWB workspace. Each project contains the metadata and definitions for objects in the data system that contains the sources and targets.

relational target schema

A target schema that contains relational data objects such as tables, views, materialized views, and sequences. All of the data for a data store or data warehouse is contained in these objects.

repository

Refer to **OWB Repository**.

SQL*Loader control file

A file or script that can be generated in OWB that extracts and transports data from file sources.

table function

A set of operators that enable manipulation of a set of input rows, which return another set of rows of the same or different cardinality. Can return a set of output rows that can be queried like a physical table.

target module

A container that holds the metadata definitions of all your data warehouse objects. Each target module corresponds to a target location that represents the physical location where the objects are stored.

target schema

A schema that contains the data objects that store your data warehouse data. You can design a relational target schema or a dimensional target schema.

transportable module

Type of module that enables rapid copying of a group of related database objects in one database, to be pasted or inserted into another database.

transformation operators

Prebuilt operators that enable commonly performed operations such as filtering, joining, and sorting. OWB also includes prebuilt operators for complex operations such as merging data, cleansing data, or profiling data.

user folder

A folder you can create to organize all or some objects in a target module based on specific object characteristics. Related tables and views that must be generated or deployed together can be placed under a common folder. For example, you may create user folders to group tables based on their functionality (sales, marketing, administration and so forth).

validation

The process of verifying metadata definitions and configuration parameters to ensure that data object definitions are complete and that scripts can be generated and deployed.

value-based hierarchy

A dimension in which hierarchical relationships are defined by a parent dimension attribute and a child dimension attribute.

workspace

The OWB structure that contains all the related projects and their objects. Graphically displayed as the canvas in the Design Center where OWB windows, navigators, wizards, and dialog boxes are laid out to create a work environment that one or more users log in to.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X