Rational Rose 2000e
Using Rose Oracle8

Copyright © 1998-2000 Rational Software Corporation.
All rights reserved.

Part Number 800-023325-000
Revision 2.1, March 2000 (Software Release 2000e)

This document is subject to change without notice.

GOVERNMENT RIGHTS LEGEND: Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational Software Corporation license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

You may copy this manual for use internal to your company provided you include Rational's copyright notice and mark the copies "made by customer."

Trademark acknowledgments:

Rational, the Rational logo, and Rational Rose are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies. Oracle8 is a registered trademark of Oracle Corporation. The Oracle8 logo is used with permission.

Contents

List of Tables vii

Preface ix

How This Manual is Organized ix

Related Documentation x

Online Help and Tutorials xi

Online Manual xi

Chapter 1 Introduction 1

About Rational Rose Oracle8 1

Getting Started with a Tutorial 2

Chapter 2 How Rational Rose Oracle8 Models Schema Objects 3

About Rational Rose, UML, and Oracle8 3
- About Oracle8-Specific Class Stereotypes 4
- About Property Settings and Property Files 4

Object Types in a Rational Rose Model 5
- About Object Types 5
- How Object Types are Structured 5
  - About Attributes 6
  - About Methods 6

How Rational Rose Oracle8 Models Object Types 6
Rules for Using Object Types 8

Object Views in a Rational Rose Model 9
About Object Views 9
How Rational Rose Oracle8 Models Object Views 10
Rules for Using Object Views 11

Object Tables in a Rational Rose Model 11
About Object Tables 11
How Rational Rose Oracle8 Models Object Tables 12
Rules for Using Object Tables 12

VARRAYS in a Rational Rose Model 12
About VARRAYs 12
How Rational Rose Oracle8 Models VARRAYs 13
Rules for Using VARRAYs 14

Nested Tables in a Rational Rose Model 14
About Nested Tables 14
How Rational Rose Oracle8 Models Nested Tables 15
Rules for Using Nested Tables 16

Relational Tables in a Rational Rose Model 16
About Relational Tables 16
How Rational Rose Oracle8 Models Relational Tables 17
Rules For Using Relational Tables 20

Relational Views in a Rational Rose Model 20
About Relational Views 20
How Rational Rose Oracle8 Models Relational Views 21
Rules For Using Relational Views 21

Chapter 3 Reverse Engineering an Oracle8 Schema 23

What is Reverse Engineering? 23

How to Reverse Engineer a Schema 23

What You Can Do Next 24
- Display the Specification for Any Model Element 24
- Create a New Class Diagram 24
- Add New Schema Objects to the Model 25
- Updating Your Schema by Forward Engineering a Rational Rose Model 26

Chapter 4 Working with a Rational Rose Oracle8 Model 27

Adding Oracle8 Base Classes to a Model 27
- Using the Oracle8 Framework to Load Scalar Datatypes 28
- Importing the Scalar Datatypes into an Existing Model 29

How to Create New Schema Objects 29
- Creating an Object Type 30
- Creating an Object View 31
- Creating an Object Table 33
- Creating a VARRAY 33
- Creating a Nested Table 34
- Creating a Relational Table 35
- Creating a Relational View 37

Checking Model Syntax 38

Generating Reports 39

Viewing/Modifying the Order of Columns and Attributes 40

Creating/Editing Foreign Keys 40

Chapter 5 Forward Engineering a Rational Rose Model 43

What is Forward Engineering? 43

How to Forward Engineer a Rational Rose Oracle8 Model 44

Appendix A Rational Rose Oracle8 Mapping Reference 47

Appendix B Schema Generation Properties 51

Overview 51

Schema Generation Properties 52
Schema Generation Properties for Rational Rose Oracle8 Projects 52

Schema Generation Properties for Rational Rose Oracle8 Classes 56

Schema Generation Properties for Rational Rose Oracle8 Operations 56

Schema Generation Properties for Rational Rose Oracle8 Attributes 57

Schema Generation Properties for Rational Rose Oracle8 Roles 58

Schema Generation Properties for Rational Rose Oracle8 Module Specifications 58

Appendix C Quick Start Tutorial 59

Overview 59

How To Use This Tutorial 59

What You Need To Run The Tutorial 60

Lesson 1 Reverse Engineering a Relational Database 60

Lesson 2 Working with a Subset of Your Database 62

Lesson 3 Adding Oracle8 Objects to Your Model 66
Part I: Creating an Object Type 67
Part II: Creating an Object View 70
Part III: Generating a Report 74

Lesson 4 Updating Your Database Schema 74

Index 79

List of Tables

Table 1 Rational Rose to Oracle8 Mapping Reference 47
Table 2 Project Properties 52
Table 3 Class Properties 56
Table 4 Operation Properties 56
Table 5 Attribute Properties 57
Table 6 Role Properties 58
Table 7 Module Specification Properties 58

Preface

This guide, *Rational Rose 2000, Using Rose Oracle8*, is for anyone who wants to use Rational Rose to:

- Model Oracle8 schemas.
- Generate schemas from Rational Rose models.
- Reverse-engineer existing schemas into Rational Rose models.

The guide assumes that you are familiar with Oracle8 concepts and constructs, and that you are comfortable with basic Rational Rose concepts and procedures.

If you need to learn to use Rational Rose, you should run the Rational Rose tutorial included on your product CD. To learn more about Rational Rose Oracle8, see the *Getting Started* tutorial that is included in Appendix C of this guide, or run the online version of the tutorial that is packaged with the Rational Rose product CD.

How This Manual is Organized

This manual contains the following chapters and appendices:

- **Chapter 1—Introduction**

  Provides an overview of Rational Rose Oracle8.
- **Chapter 2—How Rational Rose Oracle8 Models Schema Objects**

  Describes how an Oracle8 schema presented in a Rational Rose model using UML notation.
- **Chapter 3—Reverse Engineering an Oracle8 Schema**

  Describes the reverse engineering step for creating a Rational Rose model from an Oracle8 schema.

- **Chapter 4—Working with a Rational Rose Oracle8 Model**

  Explains how to create schema objects in a model, how to check model syntax, generate reports, and use Rational Rose Oracle8 wizards to modify schema objects.
- **Chapter 5—Forward Engineering a Rational Rose Model**

  Explains how to update an Oracle8 schema using the DDL generated from a Rational Rose Oracle8 model.
- **Appendix A—Rational Rose Oracle8 Mapping Reference**

  Provides a summary of how Rational Rose Oracle8 models specific schema objects and structure.
- **Appendix B—Schema Generation Properties**

  Lists the specific properties that affect how Rational Rose Oracle8 generates the DDL for an Oracle8 schema.
- **Appendix C—Quick Start Tutorial**

  This is a tutorial you can use to become familiar with Rational Rose Oracle8.

Related Documentation

The information in this guide is also provided in the form of online help. In addition, you will find context-based online help as you complete procedures and work in the various Rational Rose Oracle8 dialog boxes.

After installation and before you begin using Rational Rose Oracle8, please review any **Readme.txt** files and **Release Notes** to ensure that you have the latest information about the product.

For additional resources, refer to the *Using Rational Rose* guide and online help. If you are new to Rational Rose, visual modeling, or the Unified Modeling Language (UML), you may also want to read the book, *Visual Modeling with Rational Rose and UML*, included with your product documentation.

Online Help and Tutorials

Rational Rose Oracle8 includes comprehensive online help with hypertext links and a two-level search index.

In addition, an online version of the Rational Rose Oracle8 *Getting Started* tutorial (included in printed form as Appendix C in this guide) is available on the product CDs that are packaged with the Rational Rose software.

Online Manual

Rational Rose includes all of the user manuals online. Please refer to the Readme.txt file (found in the Rational Rose installation directory) for more information.

Chapter 1

Introduction

About Rational Rose Oracle8

Rational Rose Oracle8 enables you to create object models from Oracle8 relational schemas and extend them to exploit Oracle8's object capabilities.

Rational Rose Oracle8 visualizes existing relational databases and facilitates the discovery and composition of existing business objects. This enables you to preserve your investment in existing relational databases while taking advantage of all the benefits that object modeling and development bring to desktop and server applications.

Rational Rose Oracle8 lets developers represent business objects as native types in the application implementation language and the database. Wizards assist in the evolution from relational to object-relational by guiding you through the definition of an object-relational model.

Rational Rose Oracle8 can model, generate, and visualize relational tables, triggers, object types, objects views, VARRAYs, nested tables, and other key objects featured in Oracle8.

Getting Started with a Tutorial

Appendix C of this manual is a quick start tutorial. By following its instructions, you can:

- Reverse engineer an Oracle8 schema
- Create a new Rose class diagram
- Create new schema objects in the model
- Forward engineer the new schema objects to an Oracle8 schema

The tutorial is based on the DEMO demonstration schema that is packaged with Oracle.

Chapter 2

How Rational Rose Oracle8 Models Schema Objects

This chapter describes how Rational Rose Oracle8 models the most common elements in an object-relational schema. These include:

- Object Types
- Object Views
- Object Tables
- VARRAYs
- Nested Tables
- Relational Tables
- Relational Views

About Rational Rose, UML, and Oracle8

As a modeling tool, Rational Rose is largely language-independent. It uses Unified Modeling Language (UML) notation to model business processes, objects, components, and (with the advent of Oracle8), object-relational database schemas.

To tailor modeling to a specific environment, particularly for forward and reverse engineering, Rational Rose uses two important mechanisms:

- Class Stereotypes. Rational Rose Oracle8 uses a set of Oracle8-specific class stereotypes to model schema objects.
- Property Files. A property file controls how items are mapped to schema-specific elements.

About Oracle8-Specific Class Stereotypes

Rational Rose uses classes, class stereotypes, and the relationships between classes to capture and model schema design. There are class stereotypes for:

- Object Types
- Object Views
- Object Tables
- VARRAYs
- Nested Tables
- Relational Tables
- Relational Views

This section describes how each of these schema objects is modeled as a stereotyped class. (For a quick summary of how Rational Rose Oracle8 uses classes, stereotypes and related elements to model a database schema, see Appendix A.)

About Property Settings and Property Files

Rational Rose Oracle8 supplies a default property file (oracle8.pty) with Oracle8-specific settings. The file is automatically attached and initialized when you start Rational Rose Oracle8. The file contains property settings for:

- Project
- Class
- Attribute
- Operation
- Role
- Module

For example, **IsPrimaryKey** is a Boolean property setting for an attribute that indicates if a column in a relational table is a primary key. While you can edit and customize property settings, this section describes some of the settings that are automatically determined for you when you:

- Create a Rational Rose model by forward engineering (analyzing) an existing Oracle8 schema.
- Use Rational Rose's Data Type Creation Wizard to create new schema objects in a Rational Rose model.

For a list of the property settings see Appendix B.

Object Types in a Rational Rose Model

About Object Types

An object type is a user-defined datatype that enables you to capture a complex, real-world entity as a single structured data unit that can be queried, updated, and stored in an Oracle8 database. The object types you create can be used the same way you use built-in (scalar) relational datatypes.

By creating object types in your schema you are able to:

- Package and store data together with its associated application logic.
- Enable multiple applications to access the data without having to generate the code needed to use it.
- Create a bridge between an existing relational database and an object-based application.

How Object Types are Structured

An object type has:

- Attributes that capture the structure and state of the entity the object type is modeling. For example, a CUSTOMER object type may have attributes such as NAME, ADDRESS, CUSTOMER-ID, etc.
- Methods, such as procedures, functions, and map or order comparison methods.

About Attributes

When you create an attribute for an object type, you provide the attribute's datatype, which can be:

- Scalar, using built in types such as CHAR, NUMBER, VARCHAR, DATE, etc. For example, the CUSTOMER object type may have a NAME attribute whose datatype is VARCHAR2.
- Another object type. For example, an attribute for the CUSTOMER object type may have another PURCHASE_ORDER object type as its datatype. This association can be made by creating a reference (REF) to PURCHASE_ORDER or it can be by value.
- A collection type—either a VARRAY or a nested table. VARRAYs and nested tables are structured collections of data. For example, the CUSTOMER object type may have a CUSTOMER_CONTACTS attribute whose datatype is a VARRAY of five names.

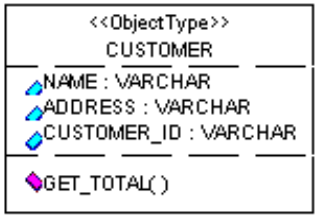
About Methods

By defining methods for an object, you are able to package application logic with your data, enabling object-based applications to use the data without requiring additional code. The methods you can define for an object type include:

- Functions and procedures
- Triggers
- Map or Order Comparison methods
- Constructor method (the method for creating the object itself)

How Rational Rose Oracle8 Models Object Types

Rational Rose Oracle8 models an object type as a class with a stereotype of Object Type. For example, the following is a sample CUSTOMER object type from a Rational Rose class diagram:



Attributes

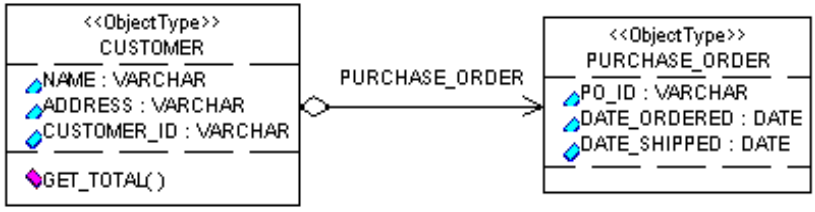
Object type attributes are modeled in Rational Rose as class attributes.

Nested Object Types

Rational Rose models an attribute whose datatype is another object type (a nested object type) as an association between the two object types. For example, if the object type CUSTOMER has an attribute whose datatype is the PURCHASE_ORDER object type, the attribute is modeled as follows:

In this case, the association is by *value*. If you create a REF (a pointer or reference similar to a foreign key in a relational table), Rational Rose Oracle8 models it as an *aggregate* association.

In this case, CUSTOMER is the Client (Role B in the association's Rational Rose specification) and PURCHASE_ORDER is the Supplier (Role A in the association's Rational Rose specification):



Rational Rose Oracle8 models the NULL constraint as a property setting for an attribute. By default, the **NullsAllowed** property is set to True.

Unique Constraint

Rational Rose Oracle8 models the Unique constraint as a property setting for an attribute. By default, the **IsUnique** property is set to False.

Methods

Rational Rose models methods as operations of a class. These can include:

- Triggers. These use the *Implementation* setting for *Export Control* to distinguish them from other methods.
- Procedures. These are methods *without* a return type. They are modeled as operations of a Rational Rose Class.
- Stored functions. These are methods *with* a return type. They are modeled as operations of a Rational Rose Class.
- Map or Order comparisons.
- Object constructor/destructor.

The type of method is captured by an operation's **MethodKind**.

Rules for Using Object Types

All Oracle8 rules for creating and using object types also apply in Rational Rose Oracle8. Please note these specific rules:

- An object type can have one MAP method or one ORDER method, but not both.
- An object type cannot have an attribute with a datatype of ROWID, LONG, LONG RAW, NCLOB, NCHAR, or NCHAR VARYING.
- You must use PRAGMAs to indicate the access level of member functions.
- You cannot define an INDEX on an object type or an object type's attributes unless it has a scalar datatype.

Object Views in a Rational Rose Model

About Object Views

An object view is a virtual object table. Its significant features include your ability to:

- "Objectify" the data currently stored in relational tables. By creating these object containers for existing relational data, your client applications can begin using object technology without the need to immediately change the underlying structure of your data.
- Begin to introduce objects into a relational database.
- Customize access to data.
- Possibly improve overall performance. (When stored as a row in an object view, relational data is retrieved as a single unit.)

Rational Rose Oracle8 greatly simplifies introducing object views to your schemas. Specifically, you can:

- Reverse engineer an existing relational schema into Rational Rose.
- Use Rational Rose Oracle8's Data Type Creation Wizard to build the object types and object views that will front-end your relational data with object-relational constructs.
- Forward engineer (generate schema) from your Rational Rose model and update your schema with the new object views and types.

To create an object view (whether you're using Rational Rose Oracle8 or generating the SQL yourself), you first need to create an object type that encapsulates the data from one or more relational tables. This object type serves as the bridge between the object view and relational data.

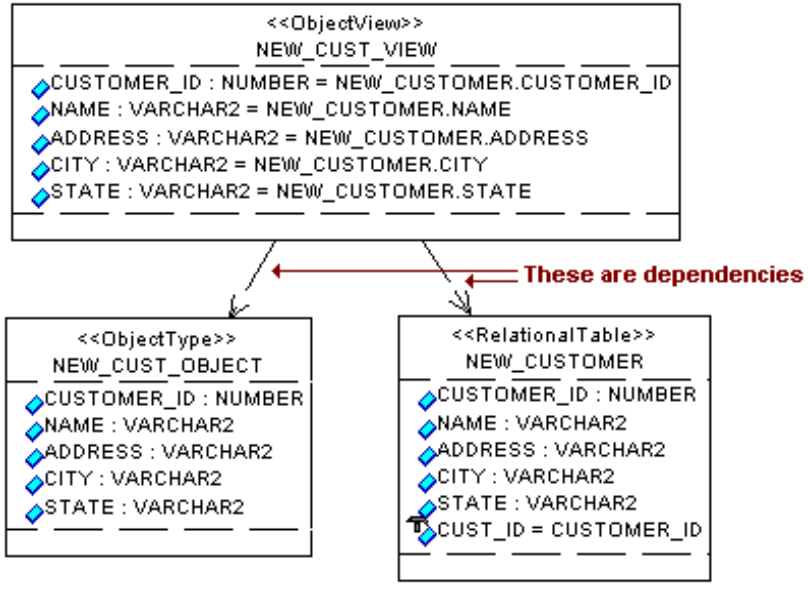
You also need to designate one or more of the object view's attributes as an object identifier that the view will use to enable REF's to point to objects (rows) in the view. (An object identifier can be a composite of more than one attribute.)

Since it generates the constructs you need, use Rational Rose Oracle8's Data Type Creation Wizard to create both the object types and object views that will extend your relational database.

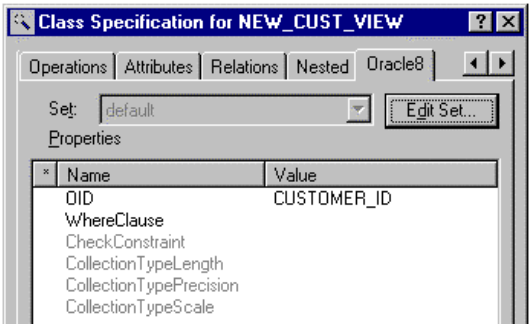
How Rational Rose Oracle8 Models Object Views

Rational Rose Oracle8 models an object view as a class with a stereotype of Object View. It represents the link between the underlying object types and relational tables as dependencies.

For example, the following shows an object view and its underlying object type and the relational table it front-ends:



The object view's attribute (or attributes, if a composite) that you selected as the Object Identifier is captured as a property setting. For example:



Rules for Using Object Views

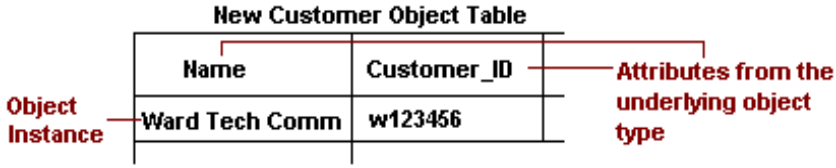
All Oracle8 rules for creating and using object views also apply in Rational Rose Oracle8. In addition, note that you cannot specify an alias when creating an object view.

Object Tables in a Rational Rose Model

About Object Tables

An object table enables you to place an object type in a relational construct. The columns in the object table correspond to the attributes in the underlying object type. Each row in the object table contains an object.

For example, consider a simple NEW CUSTOMER object type that has a NAME attribute and a CUSTOMER_ID attribute. Conceptually, an object table created from this object type would look like this:

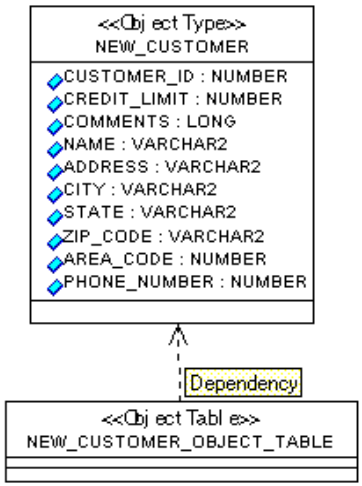


By packaging an object type in an object table, you can access the objects in using relational techniques. Note that another alternative for creating object packages that can be used in a relational environment, is to create object views.

How Rational Rose Oracle8 Models Object Tables

Rational Rose Oracle8 models an object table as a class with a stereotype of Object Table. Since object tables are built from underlying object types, Rational Rose Oracle8 models this relationship as a dependency.

For example:



Rules for Using Object Tables

All Oracle8 rules for creating and using object tables also apply in Rational Rose Oracle8.

VARRAYS in a Rational Rose Model

About VARRAYs

A VARRAY is a datatype you create to define an ordered collection of data elements.

All of the elements in a VARRAY must have the same datatype. The size of a VARRAY determines how many elements it can contain. Since a VARRAY defines an *ordered* collection, you can use it where the order of the elements is significant. (Each element in a VARRAY has an index that is based on the element's position in the array.) This is in contrast

to a nested table, which is an unordered collection type. VARRAYs are stored in a single column; they allow you to retrieve a collection as a whole.

You can use a VARRAY to define the datatype for:

- An attribute in an object type
- A column in a relational table

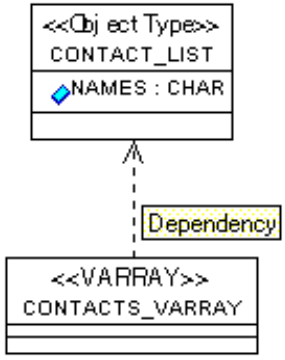
For example, in a CUSTOMER object type, you can use a CONTACT_NAME VARRAY to define the datatype for a CUSTOMER_CONTACTS attribute. If the size of the CONTACT_NAME VARRAY is five, up to five names can be in the array.

Since VARRAYs can be used to define the datatype of a column in a relational table, you are able to introduce this object construct into your existing relational schema.

How Rational Rose Oracle8 Models VARRAYs

Rational Rose Oracle8 models a VARRAY as a class with a stereotype of VARRAY. When you create a VARRAY in Rational Rose Oracle8, it can have a scalar datatype or it can be based on an object type.

If based on another object type, Rational Rose Oracle8 models this as a dependency between the object type and the VARRAY:



Rules for Using VARRAYs

All Oracle8 rules for creating VARRAYs also apply in Rational Rose Oracle8. Please note these specific rules:

- A VARRAY cannot have a datatype (directly or indirectly by nested object types) of BLOB, CLOB, NCLOB, NCHAR, or NCHAR VARYING. Note that this rule does not apply to object types that are included by reference.
- A VARRAY cannot have a nested table as a datatype, either directly or indirectly.
- A VARRAY does not support an index.

Nested Tables in a Rational Rose Model

About Nested Tables

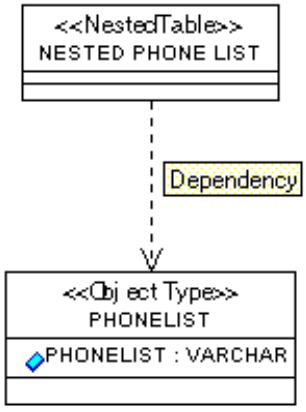
A nested table is a table of unordered data elements that is embedded as a column in another table. You can perform the same operations on a nested table that you can perform on other tables. All of the elements in a nested table must have the same datatype.

You can use a nested table to define the datatype for:

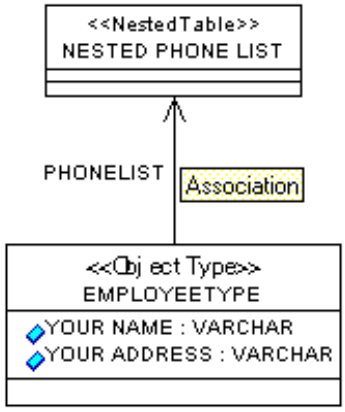
- An attribute in an object type
- A column in a relational table

How Rational Rose Oracle8 Models Nested Tables

Rational Rose Oracle8 models a nested table as a class with a stereotype of NestedTable. When you create a nested table in Rational Rose Oracle8, it is based on an object type. Rational Rose models the association between the nested table and its underlying object type as a dependency. For example:



When you use a nested table as the datatype for an object type attribute or a column in a relational table, Rational Rose models the nested table attribute as an association between the nested table schema object and the object type or relational table. For example:



Rules for Using Nested Tables

All Oracle8 rules for creating nested tables also apply in Rational Rose Oracle8. In addition, note that a nested table cannot be based on a scalar type of NCLOB, NCHAR, or NCHAR VARYING.

Relational Tables in a Rational Rose Model

About Relational Tables

Relational tables have been the fundamental schema structure for relational databases. With the introduction of object technology and Oracle8, conventional relational tables can also support:

- Using an object type, VARRAY, or nested table as the datatype for a column.
- Building object views that extend a relational database by packaging data as virtual objects.

Rational Rose Oracle8 smooths the transition to object-relational environments by:

- Enabling you to reverse engineer your current schemas into Rational Rose models.
- Create new schema objects, such as object views, as well as modify existing relational constructs.
- Create and execute DDL scripts based on the Rational Rose model.

How Rational Rose Oracle8 Models Relational Tables

Rational Rose Oracle8 models a relational table as a class with a stereotype of RelationalTable. For example, the following is a sample CUSTOMER relational table:

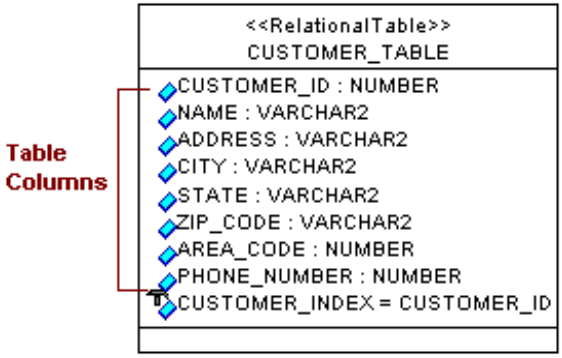
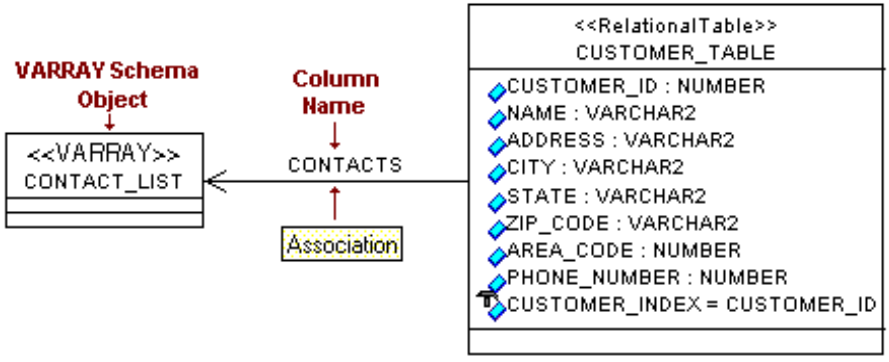


Table Columns

Rational Rose Oracle8 models table columns as attributes of the RelationalTable class.

Nested Datatypes for Columns

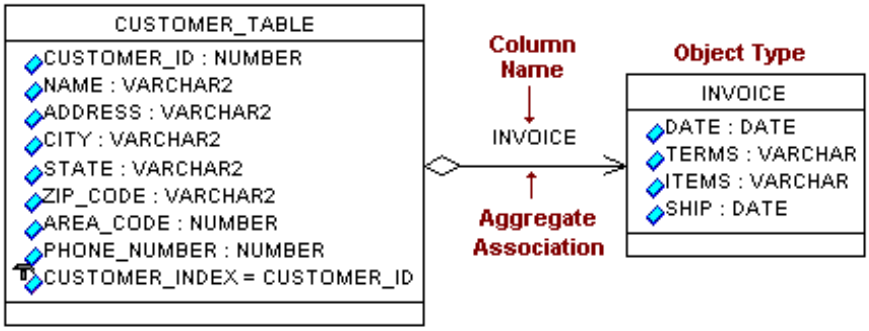
Rational Rose models a column whose datatype is a user-defined datatype (i.e., an object type, VARRAY, or nested table) as an association between the table and the datatype. For example, if the CUSTOMER relational table has a CONTACTS column whose datatype is a CONTACT_LIST VARRAY, it would be modeled as:



REFs

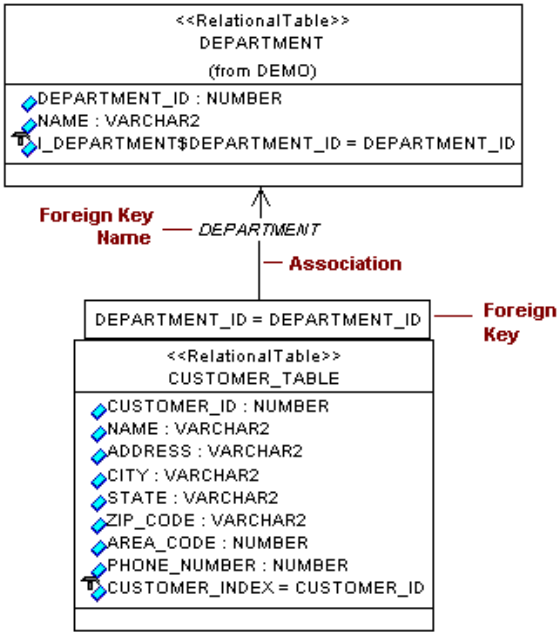
A REF is similar to a foreign key in that it serves as a reference to another entity, in this case, an object type. REFs enable you to derive the content of a column by "pointing" to an object type in the schema.

Rational Rose Oracle8 models a REF as an aggregate association. For example, if the CUSTOMER table has an INVOICE column that uses a REF to an INVOICE object type, it would be modeled as:



Foreign Keys

Rational Rose Oracle8 models foreign keys as associations. For example, if the CUSTOMER table has a foreign key to a DEPARMENT_ID in another table, it would be modeled as:



NULL and NOT NULL Constraints

Rational Rose Oracle8 models the NULL constraint as a model property associated with an attribute (column). By default, the **NullsAllowed** property is set to True.

Unique Constraint

Rational Rose Oracle8 models the Unique constraint as a model property for an attribute (column). By default, the **IsUnique** property is set to False.

Index

Rational Rose Oracle8 models an index as a class attribute and lists it with the other attributes (columns) associated with the table. The index attribute is shown with an implementation access control symbol.

Rules For Using Relational Tables

All Oracle8 rules for creating and using relational tables also apply in Rational Rose Oracle8.

Relational Views in a Rational Rose Model

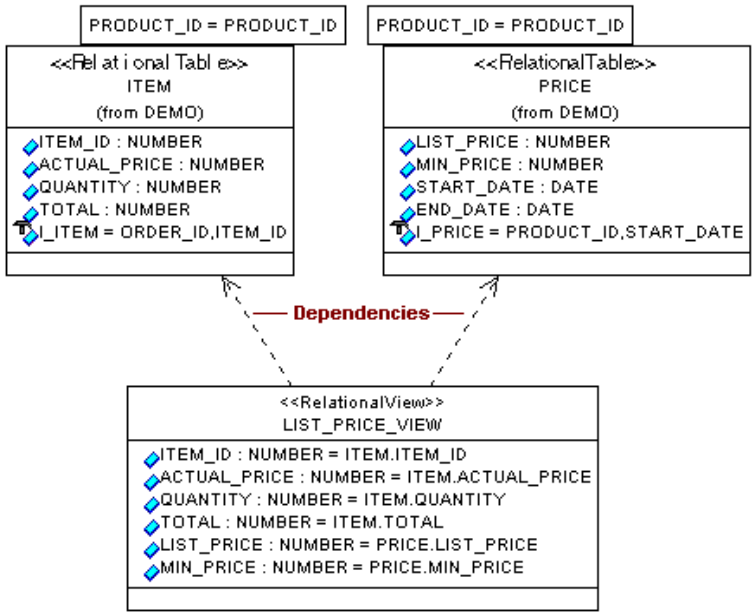
About Relational Views

A relational view is a standard Oracle construct for creating a virtual table based on one or more existing relational tables.

How Rational Rose Oracle8 Models Relational Views

Rational Rose Oracle8 models a relational view as a class with a stereotype of RelationalView. The columns of the view are modeled as attributes of the class. The view's ties to underlying tables are modeled as dependencies.

For example, consider a relational view that creates a virtual table based on the ITEM and PRICE relational tables:



Rules For Using Relational Views

All Oracle8 rules for creating and using relational tables also apply.

Chapter 3

Reverse Engineering an Oracle8 Schema

This chapter provides procedures for reverse engineering an existing Oracle8 schema, including:

- Connecting to the Oracle Database Server from Rational Rose.
- Analyzing a schema.
- Viewing the Rational Rose model that is generated from the schema.

What is Reverse Engineering?

Reverse engineering is the process of analyzing an Oracle8 schema and creating a Rational Rose model that captures the elements and structure of the schema. Rational Rose Oracle8 enables you to view and manipulate this model using the UML notation for object-oriented analysis and design.

How to Reverse Engineer a Schema

Follow these steps to reverse engineer an existing Oracle8 schema:

1. Start Rational Rose.
2. A tabbed display appears enabling you to choose a framework for a new model or to open an existing model. Since you are creating a new model, select the Oracle8 framework icon. (This automatically loads the base classes you need for using Oracle8 scalar datatypes in your model.) (Note that Unix platforms do not support frameworks.)
3. Click **Tools** > **Oracle8** > **Analyze Schema**.

4. On the dialog box that appears, enter the name of the Oracle schema that you want Rational Rose to analyze.
5. On the **Database Connect** dialog box, enter the **Database Server** name, your **UserName**, and **Password**.

   Note: *In order for Rational Rose Oracle8 to connect to the database, the OracleTNSListener80 service must be running on the host server.*

6. Rational Rose connects to the database and begins analyzing the schema you identified. It then creates a new class diagram that contains all of the schema elements. Rational Rose uses the schema name as the name of the class diagram.
7. Initially, the elements in the class diagram may overlap. To order the display, click **Tools** > **Layout Diagram**.
8. Look at the browser on the left side of the Rational Rose Oracle8 window. To view your schema, expand the **Logical View**. Each schema object is listed under the schema name.
9. Click **File** > **Save As** to name and save the model. Models are always saved with the **.mdl** file extension.

What You Can Do Next

These are among the functions you can perform with your model.

Display the Specification for Any Model Element

To display the specification for any element in the model, including relationships, you can double-click on the object in the diagram or on its name in the browser. An object's specification provides the details about how the object was analyzed and how it is modeled.

Create a New Class Diagram

When it reverse engineered your database, Rational Rose Oracle8 created a single class diagram that includes every element in the schema. You'll probably want to work with several separate diagrams, each one dealing with some logical part of the database.

When you create a new class diagram you are able to pull in the schema objects you want. Once you select a model element to copy to a new class diagram, Rational Rose Oracle8 can check the model for all other elements that are related to it and bring those elements along to the new diagram.

Follow these steps to create a new class diagram that shows a subset of your complete database model:

1. Select the  button on the toolbar to display the **Select Class Diagram** dialog box.
2. Select your schema name under Class Category and **<New>** under **Class Diagrams**.
3. Enter the name of a new diagram and click **OK** in the **Create Diagram** dialog box. This displays a new, blank diagram.
4. From the browser, drag and drop the objects from the browser to your new class diagram.
5. To have Rational Rose automatically include the objects that have a relationship to the objects you've dragged into the new diagram, click **Query > Expand Selected Elements**.
6. In the **Expand Selected Elements** dialog box, notice that you can check for **Clients**, **Suppliers**, or **Both** (the default). This enables you to control which other objects are brought into the class diagram. Click **Relations** to view the relations that Rational Rose Oracle8 can check. All relations are checked by default.

   (To understand how Rational Rose uses relationships to model the structure of your schema, see 2.)
7. Adjust the new class diagram display by clicking **Tools > Layout Diagram**.
8. Save the model.

Add New Schema Objects to the Model

You can begin to create new schema objects by first modeling them in Rational Rose. For example, if you're working with a relational database that you reverse-engineered to a Rational Rose model, you can begin to introduce object technology by creating object types and object views that are built upon the existing relational data.

The easiest (and recommended) way to create schema objects is to use Rational Rose Oracle8's Data Type Creation Wizard. The wizard steps you through the process and ensures that underlying structures (such as REFs, foreign keys, and other integrity constraints) are modeled correctly. If you intend to use Rational Rose to generate the DDL that will update your Oracle8 schema, it is important that objects and their relationships be modeled correctly.

For details about creating new objects, see 4.

Updating Your Schema by Forward Engineering a Rational Rose Model

By using Rational Rose Oracle8's forward engineering capability, you can generate and execute the DDL for all or part of a Rational Rose model. For example, if you use Rational Rose Oracle8 to create new object types and object views for existing relational data, you can use the forward engineering feature to add the new constructs to your Oracle8 schema. For more information about forward engineering, see 5.

Chapter 4

Working with a Rational Rose Oracle8 Model

This chapter describes how you use Rational Rose Oracle8 to:

- Load the base classes you need to model Oracle8 model elements.
- Create new schema objects.
- Check the syntax of your model.
- Generate reports from a model.
- Display or modify the order of columns and attributes.
- Edit or create Foreign Keys for existing relational tables.

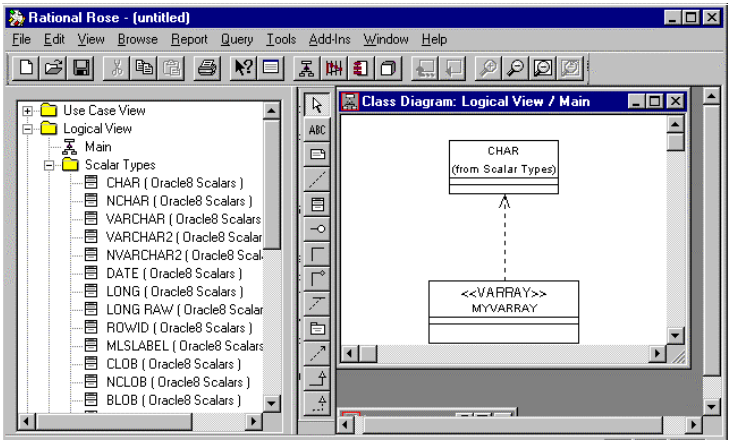
Adding Oracle8 Base Classes to a Model

For each of the Oracle8 scalar datatypes, there is a corresponding base class in Rational Rose. When you model a new Oracle8 element that relies on a scalar datatype, Rational Rose uses the datatype's underlying base class.

For example, suppose you created a VARRAY that has a scalar datatype of CHAR. On a diagram, if you were to expand all of the VARRAY's relationships, you would see a *dependency* relationship between the VARRAY and the underlying scalar base class. (To expand all of an

object's relationships, select the object and click **Query** > **Expand Selected Elements**. From the dialog box, select **Relationships** then select **All** for **Type**.)

For example:



Note that the browser has been expanded to display the Scalar Type classes.

There are two ways to add the Oracle8 base classes to a model:

- By selecting the Oracle8 framework when you are creating a new model.
- By importing the classes into an existing model, particularly a model that was created using a different framework.

Using the Oracle8 Framework to Load Scalar Datatypes

You can use the Oracle8 framework when you are creating new model. (Note that Unix platforms do not support frameworks.) By selecting the Oracle8 framework, the base classes for scalar datatypes are loaded into your model for you. To do this, follow these steps:

1. Start Rational Rose.
2. From the opening tabbed display, select the Oracle8 framework from the set of available frameworks.
3. The base classes are loaded into your model and can be displayed by using the browser.

Importing the Scalar Datatypes into an Existing Model

Rational Rose Enterprise supports multi-language models, thus enabling you to model real-world systems that incorporate different technologies.

To add the ability to model Oracle8 elements in an existing model, you need to import the Oracle8 base classes. To do this, follow these steps:

1. Start Rational Rose Enterprise and open a model.
2. Click **Tools** > **Oracle** > **Import Oracle Datatypes**.
3. The base classes are loaded into your model and can be displayed by using the browser.

How to Create New Schema Objects

Rational Rose Oracle8 features a Data Type Creation Wizard that enables you to add new schema objects to your Rational Rose model. The wizard takes the guesswork out of creating syntactically correct objects. For example, it:

- Creates the necessary classes, relationships and schema components (Rational Rose constructs) needed for schema generation (forward engineering).
- Sets the appropriate property settings and stereotypes for you.
- Places the visual elements on a Rational Rose diagram.

Using the wizard, you can create:

- Object Types
- Object Views
- Object Tables
- VARRAYs
- Nested Tables
- Relational Tables
- Relational Views

Creating an Object Type

For a description of object types and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From your class diagram, consider selecting (highlighting) any existing object types or relational tables whose attributes or columns you will be using to map to attributes of the new object type.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select **Object Type** from the list of schema object icons. Note that if you selected object types and/or tables before you started the wizard, they appear in the **Selected Items** list. Select **Next** to continue.
4. Assign a name to the new object type, identify the schema where the type will be created, and select the diagram where the new object type will appear. Select **Next**.
5. To define the object type's attributes you can map attributes or columns from existing object types or tables, or you can create new attributes.

   To select from existing tables and types, choose from the **Map From** list. Entries appear in the list automatically if they were selected *before* you started the wizard. Otherwise, display a list of available types and tables by using the **Map From** button.

   To rename a mapped attribute, double-click on its name and go to step 6. To create a new attribute, click **Create** and go to step 6. If you won't be creating or editing attributes, click **Next** and go to step 7.
6. To modify the name of a mapped attribute, use the **Name** field.

   Note: *Do not change any of the type information for a mapped attribute if the object type you're creating will be used to create an object view.*

   To create a new attribute, assign a name to the attribute and select the attribute's datatype. If you select **Scalar**, you will need to select a built-in type from the list of types, as well as the appropriate **Precision**, **Scale**, and **Length** values. If the attribute has a user-defined datatype, select either **Object Type**, **VARRAY**, or **Nested Table**. For **Type**, select from the list of existing user-defined types or

use **New** to create a new one on the fly. Use the **Nulls Allowed** checkbox to allow nulls for the attribute. Use the **Unique** checkbox to require unique values. Use the **Reference** checkbox to create a REF to another datatype. (The attribute will refer to another datatype for its value.)

Select **Add Attribute** to create the attribute and clear the fields for another definition. Click **Close** to return to the **Define Attribute** dialog box.

7. (Optional.) Define operations (methods) for the object type. Enter an operation name and select a type. **Normal** creates a function or a procedure. (A procedure will not have a **Return Type**.) For comparison methods, choose between **Map** and **Order**. Choose the appropriate **Return Type**. Optionally, specify arguments by entering a **Parameter Name**, selecting a **Direction** and a **Type**, then select **Parameter Add**. This clears the **Parameter** fields enabling you to define another set if needed. When the operation definition is complete, select the Operation **Add**. Repeat the step for each operation you're defining or click **Next** to go to the next step in the wizard.
8. (Optional.) You can change the order of the attributes in the object type by clicking and dragging items to the position you need. When you click **Finish** the new object type you created appears on the diagram you specified.

Creating an Object View

For a description of object views and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From class diagram, consider selecting (highlighting) any other existing object types or tables whose attributes or columns you will be using to create the new object view.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select **Object View** from the list of schema object icons. Note that if you selected object types or tables before you started the wizard, they appear in the **Selected Items** list. Click **Next**.
4. Assign a name to the new object view, identify the schema where the view will be created, and select the diagram where the view will appear. Click **Next**.

5. Select the **Object Type** for the object view from the list of available types. (If you selected an object type before you started the wizard, it appears as the selected type.) Or, create a new object type on the fly by clicking **New Object Type**. (The wizard will step you through the creation process.) Click **Next** to continue.

   You use the **Object View Map** dialog box to map attributes from an object type to attributes from one or more relational tables or other object view(s), as follows.

   Use the **Map To** button to select the object view(s) and/or relational tables whose attributes you will be mapping to. (This populates the selection box on the right side of the dialog box.)

   The attributes of the object type should already be displayed in the selection box on the left. To perform the mapping, click once on an attribute from the left, then click once on the corresponding attribute on the right. Click **Map**. This places your selection in the **View Map** below the two selection boxes. Note that you must map every attribute displayed on the left. (Shortcut: Instead of using the Map button, select one attribute and double-click on the second attribute. It doesn't matter which attribute you select first.)

   To add filtering criteria that will test data for inclusion in the view, use the **Where** button. This displays an area for you to enter SQL statements that are included when you generate the DDL for the schema.

   Note that once in the **View Map**, you can edit the attribute by double-clicking on the attribute name or by clicking once with the right mouse button. This displays an edit dialog box for creating expressions associated with the attribute/column.

   When you've finished mapping the view's attributes, click **Next**.

6. Select one or more attributes that will serve as an object-identifier for the view. Click **Finish**. The new object view appears on the diagram you specified.

Creating an Object Table

For a description of object tables and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From your class diagram, consider selecting (highlighting) the underlying object type for the object table you are creating (if the type already exists).
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select **Object Table** from the list of schema object icons. Note that if you selected an object type before you started the wizard, it appears in the **Selected Items** list in the opening dialog box. Select **Next**.
4. Assign a name to the new object table, identify the schema where the table will be created, and select the diagram where the table will appear. Click **Next**.
5. Select the **Object Type** for the object table from the list of available types. (If you selected an object type before you started the wizard, it appears as the selected type.) Or, create a new object type on the fly by clicking **New Object Type**. The wizard will step you through the creation process.
6. When you click **Finish**, the new object table you created appears on the class diagram you selected.

Creating a VARRAY

For a description of VARRAYs and how Rational Rose Oracle8 models them, see Section 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From the Rational Rose model, select (highlight) the object type for the VARRAY you are creating if the type already exists.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**. Note that if you selected an object type before you started the wizard, it appears in the **Selected Items** list. Click **Next**.
3. Select **VARRAY** from the list of schema object icons. Click **Next**.

4. Assign a name to the VARRAY, identify the schema where the VARRAY will be created, and select the diagram where the VARRAY will appear. Click **Next**.
5. Select the datatype for the VARRAY. The datatype can be **Scalar** or another object type. If **Scalar**, you will need to specify the appropriate **Precision, Length**, and **Cardinality** values. If the datatype is an object type, the drop down list will display all of the object types in the schema for you to choose from. If you selected an object type before you started the wizard, it is automatically displayed as the selected type. Note that you can create a new object type on the fly by selecting **New Object Type**. The wizard steps you through the creation process.
6. When you've completed the datatype designation, click **Finish.** The new VARRAY you created appears on the class diagram you selected.

Creating a Nested Table

For a description of nested tables and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From your class diagram, select (highlight) the underlying object type for the nested table you are creating if the type already exists.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**. Note that if you selected an object type before you started the wizard, it appears in the **Selected Items** list of the opening dialog box.
3. Select **Nested Table** from the list of schema object icons. Click **Next**.
4. Assign a name to the nested table, identify the schema where the table will be created, and select the diagram where the table will appear. Click **Next**.
5. Select the **Object Type** for the nested table from the list of available types. (If you selected an object type before you started the wizard, it appears as the selected type.) Or, create a new object type on the fly by clicking **New Object Type**.
6. When you click **Finish**, the new nested table you created appears on the class diagram you selected.

Creating a Relational Table

For a description of relational tables and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From your class diagram, consider selecting (highlighting) any other existing object types or tables whose attributes or columns you will be using to create the table.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select **Relational Table** from the list of schema object icons. Note that if you selected object types or tables before you started the wizard, they appear in the **Selected Items** list. Click **Next**.
4. Assign a name to the new relational table, identify the schema where the table will be created, and select the diagram where the table will appear. Click **Next**.
5. To define the relational table's columns you can map attributes or columns from existing object types or tables, or you can create new columns.

   To select from existing tables and types, choose from the **Map From** list. Entries appear in the list automatically if they were selected before you started the wizard. Otherwise, display a list of available types and tables by clicking **Map From**. To rename a mapped column, double-click on its name and go to step 6. To create a new column, click **Create** and go to step 6. If you won't be creating or editing columns, click **Next** and go to step 7.
6. Rename a mapped column by modifying the **Name** field. If you are creating a new column, assign a name to the new column and select the column's datatype.

   If you select **Scalar**, you will need to select a built-in type from the list of types, as well as provide the appropriate **Precision, Scale**, and **Length** values.

   If the column has a user-defined datatype, select either **Object Type, VARRAY**, or **Nested Table**. For **Type** select from the list of existing user-defined types or use **New** to create a new one on the fly. If the datatype is an object type and you want to create a REF, check the **Reference** checkbox.

Decide if the column will permit nulls and if the data in the column must be unique by checking the **Nulls Allowed** and **Unique** checkboxes.

When the column definition is complete, click **Add Column** to create it; this clears the fields for another column definition. When you've created all of the new columns, click **Close** to return to the **Define Column** dialog box.

7. (Optional.) You can define Foreign Keys for the table. Name the foreign key. Select the table your foreign key will reference. Once you have selected a table from the **Tables** selection box, a list of the columns with the Primary Key and Unique Key constraints is displayed in the **Columns** selection box. To select only the columns with a Primary Key, click **Select Primary Keys**. This acts as a filter.

   When you have selected the columns from the **Column** selection box, use the **Add Foreign Key** button to complete creating the Foreign Key. When you do this, the new Foreign Key is added to the **Foreign Keys Table** at the bottom of the dialog box, and the **FK Name** field is cleared for you to create another Foreign Key.

   The **Foreign Keys table** lists the Foreign Keys you have created for the new relational table. You can edit a column name by double-clicking on the name or by displaying the control menu via the right mouse button.

   Click **Next** to continue.

8. (Optional.) You can define indices for the table. Assign a name to the index. Select one or more columns and click **Add**. If the index should be a Primary Key index, check **Primary** *before* you use the **Create Index** button. When you select **Create Index**, the entry is added to the list of indices you've created for the table. This clears the definition fields enabling you to create another index. Click **Next** to continue.

9. (Optional.) You can change the column order of the table by clicking and dragging columns to the position you need. When you click **Finish**, the relational table you created appears on the diagram you specified.

Creating a Relational View

For a description of relational views and how Rational Rose Oracle8 models them, see 2. For a summary of property settings that affect a model, see Appendix B.

1. (Optional.) From your class diagram, consider selecting (highlighting) any other existing relational tables and/or views whose columns you will be using to create the new view.
2. Call the Data Type Creation Wizard. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select **Relational View** from the list of schema object icons. Note that if you selected tables and/or views before you started the wizard, they appear in the **Selected Items** list. Click **Next**.
4. Assign a name to the new view, identify the schema where the view will be created, and select the diagram where the view will appear. Select **Next**.
5. If you selected one or more relational tables or relational views before you started the wizard, they are displayed in the **Map From** selection box. Otherwise, you can use the **Map From** button to display a list of all of the relational tables and relational views currently in the schema.

   To add a column from the **Map From** box to the list of the new relational view's column list you can:

   - Double-click on a single column name.
   - Double-click on the relational table or relational view name to add all of the columns in the table or view.
   - Click **Add** to add a single column or to add all of the columns from a relational table or relational view.

   You can add filtering criteria to the view to determine when to include data. To do this, click **Where**. **Where** opens a *Where Clause* dialog box that lets you enter SQL statements that are carried forward into your Oracle8 schema when you perform schema generation.
6. Click **Next** to continue.
7. (Optional.) You can change the order of the view by clicking and dragging columns to the position you need. When you click **Finish**, the new relational view appears on the diagram you specified.

Checking Model Syntax

The Rational Rose Oracle8 Syntax Checker checks the Oracle8 model elements you select for possible DDL generation errors. You can use the Syntax Checker to make sure the schema objects in your model are defined appropriately and will generate the correct DDL for forward engineering. The syntax check is always the first line of defense for debugging. (In fact, when you generate DDL from a model, a syntax check is automatically performed for you as part of the generation process.)

The portions of the model that are checked are:

- All components in the model that have the <<Schema>> stereotype and all of the classes that are assigned to those components.
- All components in the model that have their **IsSchema** property set to True and all classes that are assigned to those components.
- Any classes in the model that have a valid Oracle8 stereotype (see 2) but that are not assigned to any component.

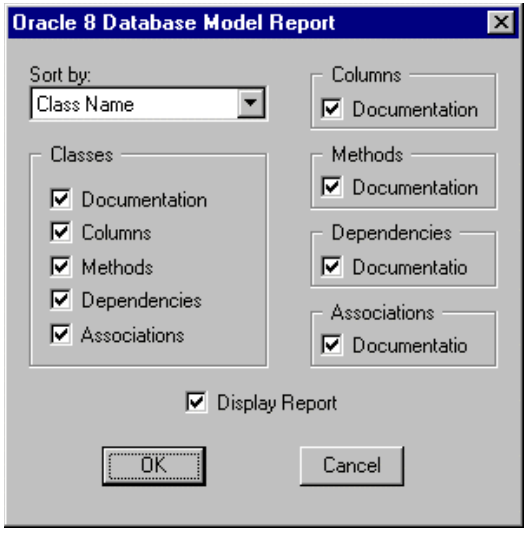
To check the syntax of your model, follow these steps:

1. Select (highlight) a model element on a class diagram.
2. Click **Tools** > **Oracle8** > **Syntax Checker**.
3. A message indicates if warning or errors have been found. View the results in the Rational Rose Log by clicking **Window > Log**.
4. Clear and/or save the contents of the log file by clicking **File** and the appropriate function from the Rational Rose menu. To return to your class diagram, use the **Window** menu.

Generating Reports

Rational Rose Oracle8 has a reporting feature that lets you view selected details about the schema objects in your model. Follow these steps to create a report:

1. Click **Tools > Reports**.
2. This displays the selection criteria you can set for the report, including the sort order and level of detail:



3. When you've selected your report criteria, use **OK** to generate the report. Rational Rose Oracle8 prompts you for a file where the report will be stored. Reports are automatically saved with the **.or8** file extension.
4. If you elected to display the report, Rational Rose Oracle8 opens Notepad (by default) and displays the report.

Viewing/Modifying the Order of Columns and Attributes

In some databases, the order of attributes or columns in a type, view, or table, may be significant. Use the Rational Rose Oracle8 Ordering Wizard to display how attributes/columns are ordered in an object type, relational table, or relational view. This is particularly useful if an attribute has a non-scalar attribute (such as another datatype or REF) since these are modeled as relationships with other schema objects.

To use the Ordering Wizard, follow these steps:

1. From your class diagram, select an object type, relational table, or relational view whose attributes/columns you want to view or whose order you want to rearrange.
2. Click **Tools** > **Oracle8** > **Ordering Wizard**.
3. On the dialog box, use your mouse to drag attributes/columns to the positions you want.
4. Click **Finish** when ordering is complete. This saves the new order you created and determines how the DDL is generated if you forward engineer the schema object.

Creating/Editing Foreign Keys

Rational Rose Oracle8 provides a Foreign Key Wizard that enables you to create and edit Foreign Keys for the columns in a relational table. (For details about how Rational Rose Oracle8 models Foreign Keys, see 2.)

When you identify a Foreign Key, you link the column content of the relational table you have selected to the column content (Unique column, usually a Primary Key) of another referenced table. By adding this constraint, you indicate that the data in the two columns must be identical. (The value of the Foreign Key must match the value of the Unique column.)

You can create composite keys that consist of multiple columns (up to 16) as long as the composite Foreign Key and composite Unique columns remain exactly the same (same datatypes).

To use the Foreign Key Wizard, follow these steps:

1. On your class diagram, select the relational table where you are adding/editing a foreign key.
2. Click **Tools** > **Oracle8** > **Edit Foreign Key.**
3. Complete the dialog box as follows:
   - **FK Name**

     This is the name of the Foreign Key you are creating/editing.
   - **Tables**

     The **Tables** selection box displays all of the tables that currently exist in your schema. Select the table your Foreign Key will reference.
   - **Columns**

     Once you have selected a table from the **Tables** selection box, a list of the columns with the Primary Key and Unique Key constraints is displayed in the **Columns** selection box. (Note that the column names are preceded with a U for Unique or P for Primary.)

     To select only the columns with a Primary Key, click **Select Primary Keys**. This acts as a filter.
   - **Add Foreign Key**

     When you have selected the columns from the **Column** selection box, click **Add Foreign Key** to complete creating the Foreign Key. When you do this:

     The new Foreign Key is added to the **Foreign Keys table** at the bottom of the dialog box, and

     The **FK Name** field is cleared for you to create another Foreign Key.
   - **Foreign Keys Table**

     This table lists the Foreign Keys you have created for the new relational table. You can edit the column names by double-clicking on the name or by displaying the control menu via the right mouse button.
4. Click **Finish** to complete the Foreign Key definition.

Chapter 5

Forward Engineering a Rational Rose Model

This chapter describes how you use Rational Rose Oracle8's forward engineering feature to update an Oracle8 schema from a Rational Rose model. It includes:

- What forward engineering is.
- How to create the DDL for selected model elements.
- How to execute the generated DDL against an existing Oracle8 schema.

What is Forward Engineering?

Forward engineering is the process of analyzing the elements in a Rational Rose Oracle8 model, generating a DDL script based on those elements, and executing the DDL against an Oracle8 database schema. For example, if you reverse-engineered a relational schema into a Rational Rose model, then created new schema objects based on the relational objects, you could use forward engineering to add the new schema objects to your original Oracle8 relational schema.

Rational Rose Oracle8's forward engineering feature enables you to:

- Generate the DDL for some or all of the elements in a model.
- Examine the generated DDL script.
- Connect to the Oracle8 database server to execute some or all of the generated statements, or
- Store the DDL script to a file for later execution.

How to Forward Engineer a Rational Rose Oracle8 Model

To forward engineer all or part of a Rational Rose Oracle8 model, follow these steps:

1. (Optional.) From your Rational Rose model, select (highlight) the schema objects for which you want to generate a DDL script. These are the Rational Rose elements that you will forward engineer to an Oracle8 schema. If you don't select the model elements now, you can do so from the **Schema Generation** dialog box.
2. Click **Tools** > **Oracle8** > **Schema Generation**. The Syntax Checker automatically checks the syntax of your model and the **Schema Generation** dialog box is displayed. If the Syntax Checker encountered errors, they are displayed in the lower half of the **Schema Generation** dialog box.
3. Complete the dialog box by supplying the following:

   File Name

   This displays the name of the default file where the generated DDL script is stored. To select a different file, enter the file name or click **Browse** to browse folders. If you enter the name of a file that doesn't exist, it will be created for you. Note that the default file name is controlled by a schema generation property for the project.

   Select Components

   Click on the schema icon to expand it. If you selected model elements on a class diagram before starting schema generation, they are selected on the list. (They appear in bold.)

   If you didn't select elements before, you can do so now, or you can change your prior selections. To select all of the components in the model, select the schema name.
4. Click **Generate** to create the DDL for the selected components. (Execution is a separate step.) This displays the **DDL Execution** dialog box.
5. The top half of the **DDL Execution** dialog box displays status; namely, any errors that were encountered during DDL generation (and later, during execution). The bottom half of the dialog box displays the generated DDL statements.

6. Initially, all of the generated statements are selected (highlighted). At this point, you can:
   - Use **Cancel** to exit the dialog box. The DDL script isn't saved or executed. The **DDL Execution** dialog box is closed, returning you to your class diagram.
   - Review and, if desired, edit the generated statements.
   - Use **Done** to save the DDL script without executing it. The script is saved to the file you specified in the previous dialog box and the **DDL Execution** dialog box is closed, returning you to your class diagram.
   - Select (highlight) some or all of the statements that you intend to execute, then use **Execute** to initiate a connection to the Oracle Database Server where your schema is located. *Only those statements that are highlighted can be executed.*(See Step 7.)
7. As a result of selecting **Execute,** Rational Rose Oracle8 displays a **Database Connect** dialog box. Enter the Database Server name, your UserName, and your Password.

   Note: *In order for Rational Rose Oracle8 to connect to the database, the OracleTNSListener80 service must be running on the host server.*

   When the connection is established, the DDL you selected is executed against the schema you named. Execution status appears in the upper half of the **DDL Execution** dialog box.

Appendix A

Rational Rose Oracle8 Mapping Reference

The following in a quick reference to how Rational Rose Oracle8 models Oracle8 schema objects.

Table 1 Rational Rose to Oracle8 Mapping Reference

Oracle Concept	Model Item	Description
Object Type	Class Stereotype= Object Type	Rational Rose models an Object Type as a class with an Object Type stereotype. The Object Type's attributes are modeled as class attributes; its methods are modeled as operations.
Relational Table	Class Stereotype= Relational Table	Rational Rose models a Relational Table as a class with a Relational Table stereotype. Table columns are modeled as class attributes.
Object Table	Class Stereotype= Object Table	Rational Rose models an Object Table as a class with an Object Table stereotype. It models the table's relationship with its underlying Object Type as a dependency.

Oracle Concept	Model Item	Description
Object View	ClassStereotype= Object View	Rational Rose models an Object View as a class with an Object View stereotype. View columns are modeled as attributes; methods are modeled as operations. The relationships between the Object View and its underlying object type(s) and relational table(s) are modeled as dependencies.
VARRAY	ClassStereotype= VARRAY	Rational Rose models a VARRAY as a class with a VARRAY stereotype. The relationship between the VARRAY and an underlying type is modeled as a dependency.
Nested Table	Class Stereotype= Nested Table	Rational Rose models a Nested Table as a class with a Nested Table stereotype. The relationship between the Nested Table and the underlying type is modeled as a dependency.
Relational View	ClassStereotype= Relational View	Rational Rose models a Relational View as a class with a Relational View stereotype. View columns are modeled as class attributes. Rational Rose uses dependencies to model relationships between the view and its underlying tables.
Column Name for Table or View	Attribute	The scalar type of the column is defined by using two model properties; one for the scalar type and a second for the length or size of the column (if applicable).
Attribute for an Object Type	Attribute	Scalar or another Object Type.

Oracle Concept	Model Item	Description
Object Identifier (Object Views)	Attribute	Rational Rose uses a model property to model the object identifier for an Object View.
Index	Attribute	The attribute name is equal to the index name and the initial value of the attribute is a comma delimited set of column names for the index. The **IsIndex** property is set to True and if this Index is a Primary Key Index, the **IsPrimaryKey** property is also set to True.
REF	Association	Rational Rose models a REF as an aggregate relationship between objects.
Foreign Key	Association	Rational Rose models a foreign key as an association relationship between two tables.
NULL and UNIQUE Constraints	Attribute	These constraints are determined by property settings for an attribute or table column, specifically, **IsUnique** and **NullsAllowed**.
Object or Relational View Column	Attribute	The attribute name is equal to the view column name or the view column alias, if an alias is provided. Where an alias is provided, the initial value of the attribute is set to the fully qualified column name.
Object Type Function (has a return type)	Operation	Rational Rose models methods as operations, with the **ModelKind** property set to Function.

Oracle Concept	Model Item	Description
Object Type Procedure (no return type)	Operation	Rational Rose models methods as operations, with the **ModelKind** property set to Procedure.
Trigger	Operation	The export control for these operations is set to implementation to signify that these types of class methods are triggers.
Schema	Component (Subprogram specification)	The stereotype is set to Schema.
Database Domain	Physical Package	The stereotype is set to Database Domain.

Appendix B

Schema Generation Properties

Overview

This appendix describes the schema generation properties available for Rational Rose Oracle8. It includes descriptions of these properties:

- Project (Model) properties
- Class properties
- Operation properties
- Attribute properties
- Role properties
- Module properties

For Oracle8 properties and property sets, you can:

- Display or modify schema generation property values
- Remove an overriding item level property
- Make a property be item-specific
- Create a new property set
- Delete a property set
- Display or edit a specific property set
- Reinstate the state and value of the last committed change

Schema Generation Properties

Schema generation properties provide Oracle8-specific information that is necessary for generating DDL script from a Rational Rose model.

When a model component (schema) is created, Rational Rose sets each of its properties to a default value that can be modified.

Schema Generation Properties for Rational Rose Oracle8 Projects

Project (or design-level) properties affect the entire Rational Rose model. The following table summarizes the schema generation properties, as well as any defined default values, for Rational Rose Oracle8 models.

Table 2 Project Properties

Property	Type	Description
DDLScriptFilename	string	The name of the default DDL script file that is created when you generate a schema from a Rational Rose model.
DropClause	Boolean	If true, generates a DROP statement for each Oracle8 entity. Default is False.
PrimaryKeyColumnName	string	Suffix for generated primary keys. The initial setting is blank. The recommended value is _ID.
PrimaryKeyColumnType	string	Default type and length for primary keys. The initial value is NUMBER(5,0)
SchemaNamePrefix	string	An optional naming standard that is added to the beginning of a component name for each generated schema. The initial setting is blank. The recommended value is S_.

Property	Type	Description
SchemaNameSuffix	string	An optional naming standard that is appended to the end of the component name for each generated schema. The initial setting is blank. The recommended value is _S.
TableNamePrefix	string	An optional naming standard that is added to the beginning of the class name for each generated table. The initial setting is blank. The recommended value is T_.
TableNameSuffix	string	An optional naming standard that is appended to the end of the class name for each generated table. The initial setting is blank. The recommended value is _T.
TypeNamePrefix	string	An optional naming standard that is added to the beginning of the class name for each generated object table. The initial setting is blank. The recommended value is O_.
TypeNameSuffix	string	An optional naming standard that is appended to the end of the class name for each generated object table. The initial setting is blank. The recommended value is _O.
ViewNamePrefix	string	An optional naming standard that is added to the beginning of the class name for all generated object and relational views. The initial setting is blank. The recommended value is V_.

Property	Type	Description
ViewNameSuffix	string	An optional naming standard that is appended to the end of the class name for all generated object and relational views. The initial setting is blank. The recommended value is _V.
VarrayNamePrefix	string	An optional naming standard that is added to the beginning of the class name for each generated Varray. The initial setting is blank. The recommended value is VA_.
VarrayNameSuffix	string	An optional naming standard that is appended to the end of the class name for each generated Varray. The initial setting is blank. The recommended value is _VA.
NestedTableNamePrefix	string	An optional naming standard that is added to the beginning of the class name for each generated nested table. The initial setting is blank. The recommended value is NT_.
NestedTableNameSuffix	string	An optional naming standard that is appended to the end of the class name for each generated nested table. The initial setting is blank. The recommended value is _NT.
ObjectTableNamePrefix	string	An optional naming standard that is added to the beginning of the class name for each generated object table. The initial setting is blank. The recommended value is OT_.

Property	Type	Description
ObjectTableNameSuffix	string	An optional naming standard that is appended to the end of the class name for each generated object table. The initial setting is blank. The recommended value is _OT.
AttributeNamePrefix	string	An optional naming standard that is added to the beginning of the attribute name for each generated attribute within an object type. The initial setting is blank. The recommended value is A_.
AttributeNameSuffix	string	An optional naming standard that is appended to the end of the attribute name for each generated attribute within an object type. The initial setting is blank. The recommended value is _A.
MemberNamePrefix	string	An optional naming standard that is added to the beginning of the operation name for each generated member within an object type. The initial setting is blank. The recommended value is M_.
MemberNameSuffix	string	An optional naming standard that is appended to the end of the operation name for each generated member within an object type. The initial setting is blank. The recommended value is _M.

Schema Generation Properties for Rational Rose Oracle8 Classes

The following are the schema generation properties for classes.

Table 3 Class Properties

Property	Type	Description
OID	string	The Object ID of the Object View.
WhereClause	string	Used by object views and relational views.
CheckConstraint	string	Used to indicate a CHECK constraint.
CollectionTypeLength	string	Used by Varrays of Scalar type.
CollectionTypePrecision	string	Used by Varrays of Scalar type.
CollectionTypeScale	string	Used by Varrays of Scalar type.

Schema Generation Properties for Rational Rose Oracle8 Operations

The following are the schema generation properties for operations.

Table 4 Operation Properties

Property	Type	Description
MethodKind	Enumerated	Can be Function, Procedure, Operator, Constructor, Destructor, Trigger, MapMethod, OrderMethod, Calculated Column Order Number if Calculated View Column. Initial value is Function.
OverloadID	String	Supported for Functions.
OrderNumber	String	The column order, if a calculated View column.
IsReadNoDataState	Boolean	Initial setting is False.

Property	Type	Description
IsReadNoProcessState	Boolean	Initial setting is False.
IsWriteNoDataState	Boolean	Initial setting is False.
IsWriteNoProcessState	Boolean	Initial setting is False.
IsSelfish	Boolean	Initial setting is False.
TriggerType	Enumerated	Can be AFTER, BEFORE, INSTEAD OF.
TriggerEvent	Enumerated	Can be INSERT, UPDATE, DELETE, INSERT OR UPDATE, INSERT OR DELETE, UPDATE OR DELETE, INSERT OR UPDATE OR DELETE.
TriggerText	String	Part of trigger definition.
TriggerRefererencingNames	String	Used for the trigger Referencing option.
TriggerForEach	Enumerated	Can be ROW, STATEMENT.
TriggerWhenClause	String	Parameter for a trigger.

Schema Generation Properties for Rational Rose Oracle8 Attributes

The following are the schema generation properties for attributes.

Table 5 Attribute Properties

Property	Type	Description
OrderNumber	String	Column order for tables.
IsUnique	Boolean	If set to False (the default), the attribute is not required to be unique.
NullsAllowed	Boolean	If set to True (the default) attribute is required to have a value. A value is required for NOT NULL settings.

Property	Type	Description
Length	string	Used for scalar datatypes such as CHAR, VARCHAR, etc.
Precision	string	Used by NUMBER scalar datatype.
Scale	string	Used by NUMBER scalar datatype.
IsIndex	Boolean	Identifies whether the attribute is part of an index. Default is FALSE.
IsPrimaryKey	Boolean	Marks the attribute as the Primary Key or part of the Primary Key. The attribute must be a scalar type. If more than one Primary Key attribute is identified, a concatenated primary key is generated. Default is False.
CompositeUnique	Boolean	Identifies if attribute is part of a composite. The default is False.
CheckConstraint	string	Used to indicate a CHECK constraint.

Schema Generation Properties for Rational Rose Oracle8 Roles

The following is the schema generation property for a role.

Table 6 Role Properties

Property	Type	Description
OrderNumber	string	Order of the table column.

Schema Generation Properties for Rational Rose Oracle8 Module Specifications

The following is the schema generation property for module specification.

Table 7 Module Specification Properties

Property	Type	Description
IsSchema	Boolean	Identifies if the component is a schema.

Appendix C

Quick Start Tutorial

Overview

Welcome to the Rational Rose Oracle8 Quick Start Tutorial! Rational Rose Oracle8 is the tool of choice for evolving relational databases to object-relational databases.

By simply running the Rational Rose Oracle8 Analyzer you get a complete model of your existing database schema.

When you're ready to add new Oracle8 object functionality to your database, the Rational Rose Oracle8 Data Type Creation Wizard walks you through the process of adding objects to your database model.

As you add objects to the model, let Rational Rose Oracle8 automatically check for syntax errors, generate your updated schema and apply it to your database.

How To Use This Tutorial

This tutorial provides you with two ways to get up and running with Rational Rose Oracle8. Choose the one that suits you best:

- The Printed Tutorial

  The printed version (this appendix) of the Quick Start tutorial provides you with step-by-step instructions, illustrated with screen shots of the actual Rational Rose Oracle8 application. Follow the instructions in this appendix while seated at your computer with Rational Rose Oracle8 up and running. Compare your results with the screen shots to keep yourself on track.

- The Online Tutorial

  The online version of the Quick Start tutorial comes to you as an online help file. Like the printed tutorial, the online tutorial provides complete step-by-step instructions to get you up and running in the Rational Rose Oracle8 environment. It also gives you something extra: Movies. At the end of each lesson, you'll find a videocam icon. Click on the icon and you'll see a movie that demonstrates the entire lesson.

Both versions are comprised of four lessons. Each lesson builds upon the activities of the previous lessons, so be sure to do the lessons in order.

What You Need To Run The Tutorial

Here's a list of what you need to run the Rational Rose Oracle8 Quick Start Tutorial:

- A Windows 95 or Windows NT system with:
  - Oracle8 client software
  - Rational Rose with the Oracle8 add in active
- Access to an Oracle8 server with the DEMO database

Note: *The DEMO database is a sample database that is packaged with the Oracle8 server software.*

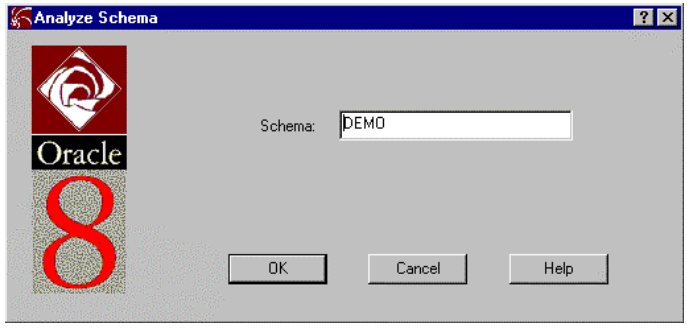
Lesson 1 Reverse Engineering a Relational Database

Let's assume that you have brought an existing Oracle database into the Oracle8 environment and you're ready to enter the world of objects. To start off, you want to create a model from your database. We call this *reverse engineering.*

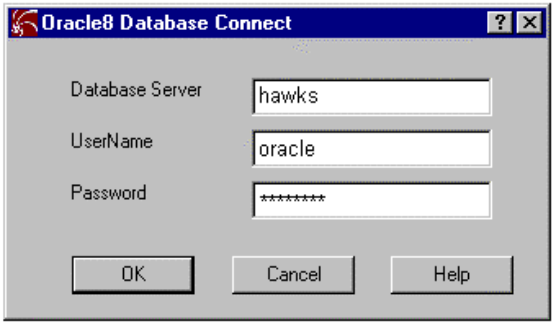
Now, you could review your schema and create the model yourself, or you could do it the easy way, by letting Rational Rose Oracle8 do it for you.

Follow these simple instructions to let Rational Rose Oracle8 reverse engineer your current database:

1. Start Rational Rose.
2. A tabbed display appears enabling you to choose a framework for a new model or to open an existing model. Since you are creating a new model, select the Oracle8 framework icon. (This automatically loads the base classes you need for using Oracle8 scalar datatypes in your model.)
3. Click **Tools** > **Oracle8** > **Analyze Schema.**
4. Use **Analyze Schema** to supply Rational Rose with the information needed to select the Demo schema and connect to the database server.
   - Enter **DEMO** in the **Schema** field; (DEMO is a sample provided with your Oracle8 system. If this database is not available to you, you can use another Oracle database.



   - Enter your own database server name, username and password in the Database Connect dialog box.



5. Click **OK**.

   Note: *In order for Rational Rose Oracle8 to connect to the database, the OracleTNSListener80 service must be running on the host server.*

6. Watch as the Analyzer reads your database, creates a model, and displays a class diagram called **Demo Schema Coverage**.
7. Adjust the class diagram display by clicking **Tools > Layout Diagram**.
8. Look at the browser in the upper left portion of the Rational Rose Oracle8 window. Notice that **Demo** has been added to the **Logical View**. (If the browser is not displayed, click **View > Browser** from the menu to display it.)
9. Click the **+** sign next to the **Demo** package to expand it in the browser and view the database elements it contains.
10. Click **File > Save As** to name and save the Demo model.

Lesson 2 Working with a Subset of Your Database

When it reverse engineered your database, Rational Rose Oracle8 created a single class diagram that covers every element in the schema. That class diagram provides a valuable view of your database, but it's a lot to work with at one time.

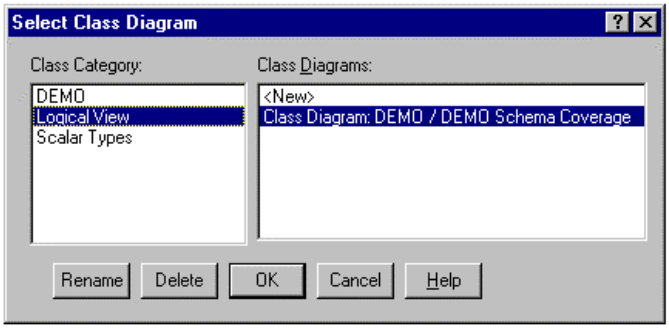
Rational Rose provides an easy solution to organizing a model into manageable diagrams. You'll probably want to work with several separate diagrams, each one dealing with some logical part of the database. Once you select a model element to copy to a new class diagram, Rational Rose Oracle8 can check the model for all other elements that are related to it and bring those elements along to the new diagram.

Follow these steps to create a new class diagram that shows a subset of your complete database model:

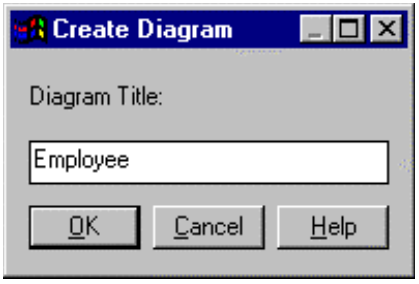
1. If the model you created in Lesson 1 is no longer open, click **File > Open** to open it.
2. Display the **DEMO Schema Coverage** diagram by doing the following:
   - Click the  button on the toolbar to display the **Select Class Diagram** dialog box.

- Select **DEMO** under **Class Category** and **DEMO Schema Coverage** under **Class Diagrams**.
- Click **OK**.

3. Click the  button on the toolbar to redisplay the **Select Class Diagram** dialog box.



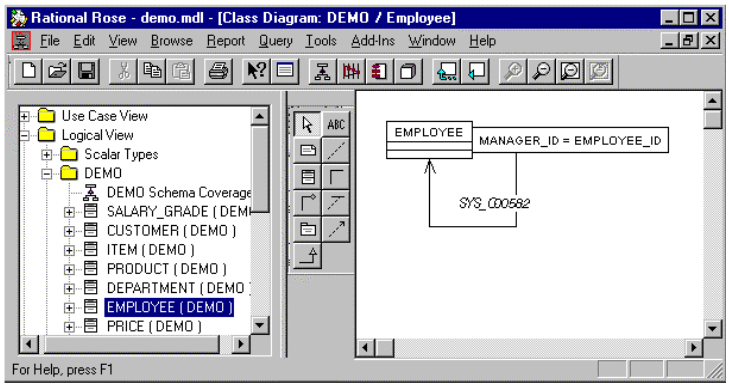
4. Click **DEMO** under **Class Category** and **<New>** under **Class Diagrams**, and click **OK**.
5. Enter **`Employee`** and click **OK** in the **Create Diagram** dialog box.



Rational Rose Oracle8 displays a new, blank class diagram called **Employee**.

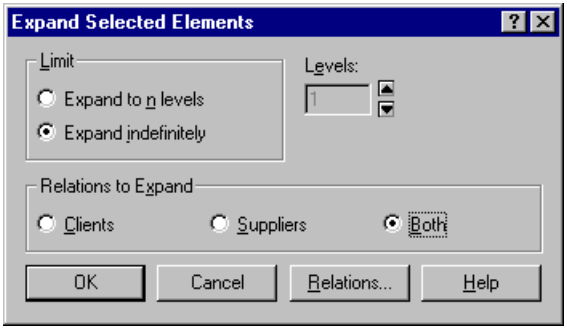
6. Go to the browser and locate the **Employee** object.
7. Drag and drop the **Employee** object from the browser to your new **Employee** class diagram.

8. Select the **Employee** class in the **Employee** diagram. (You can select as many classes as you like, but for this example, just select the one.)

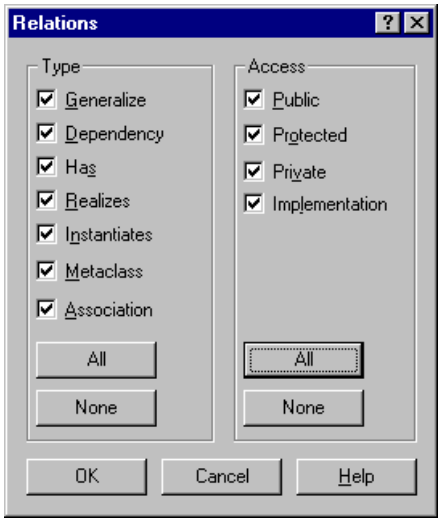


Note that Employee is a relational table that has a foreign key to a Manager-Id in another table. As you will see, we can pull in the other, related objects into this diagram.

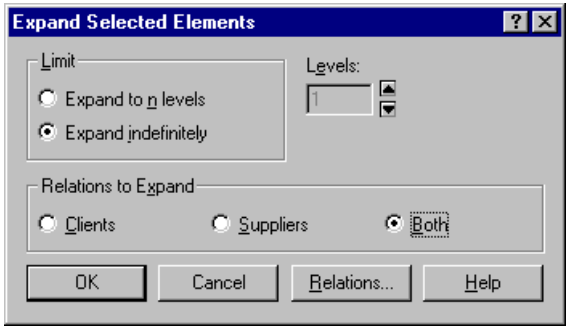
9. Click **Query > Expand Selected Elements**.
10. In the **Expand Selected Elements** dialog box, do the following:
    - Notice that you can check for **Clients**, **Suppliers**, or **Both** (the default). Leave these settings as they are.



- Click **Relations** to view the relations that Rational Rose Oracle8 can check. All relations are checked by default.

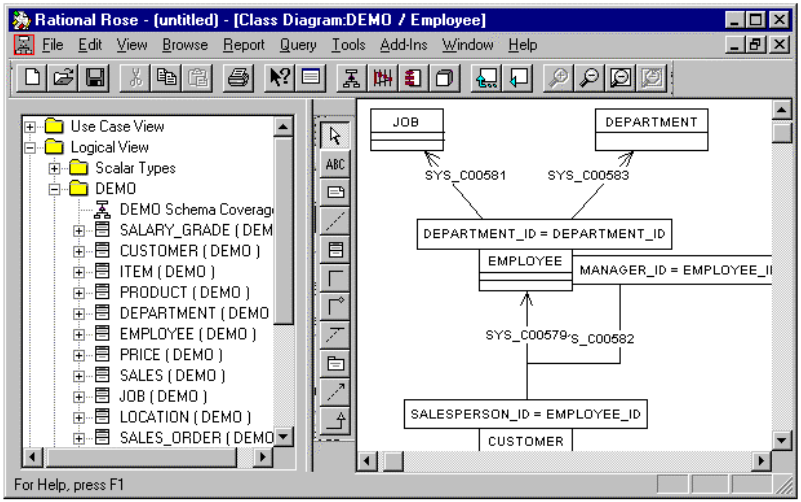


11. Leave the default **Relations** settings as they are and click **OK** to return to the **Expand Selected Elements** dialog box.



12. Click **OK** to add any classes related to the **Employee** diagram. In this case, you'll see that the **Job**, **Department**, and **Customer** classes have been added to the diagram.

13. Adjust the class diagram display by clicking **Tools > Layout Diagram**. You now have a class diagram that shows only those parts of your database schema that are related to **Employee**:



14. Save the model.

Lesson 3 Adding Oracle8 Objects to Your Model

Oracle8 defines a number of database-specific objects to extend your current relational database into an object-relational database. Like all objects, each Oracle8 object has a set of attributes, operations, and, optionally, relationships that define it.

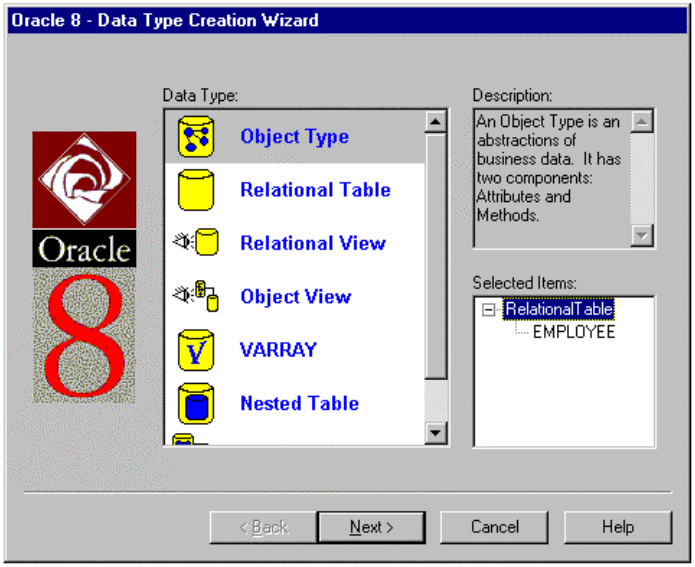
Rational Rose Oracle8's Data Type Creation Wizard makes it easy for you to add objects to your database model, and then to the database itself. You can create new objects from information that is already in your database, as well as create them from scratch.

To add objects to your model, you start up the Wizard, answer questions about the object you want to add, and let the Wizard do the rest. For example, you can create an object view that "objectifies" the data currently stored in relational tables. This is a two step process, because you must first create an object type that maps the relational table's columns to the type's attributes.

This lesson is split into parts for each stage of the creation process. In addition, it shows you how to create a report based on your model.

Part I: Creating an Object Type

1. If your model is no longer open, click **File > Open** to open it.
2. Display the **Employee** diagram by doing the following:
   - Click the  button on the toolbar to display the **Select Class Diagram** dialog box.
   - Click **DEMO** under **Class Category** and **Employee** under **Class Diagrams**.
   - Click **OK**.
3. Select **Employee** in the **Employee** diagram.
4. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
5. Select the object to add to the diagram, in this case **ObjectType.**




RoseTip

Notice that the Employee table appears in the Selected Items list. That's because you selected it in your diagram in Step 3.

6. Click **Next** to continue.

7. Fill in the information to tell the Data Type Creation Wizard the name of the object you are creating and where to put it in the model:
   - Enter the Name of the ObjectType you want to add, in this case, **EMPLOYEE_TYPE.**

   Use underscores instead of spaces when you enter your information.

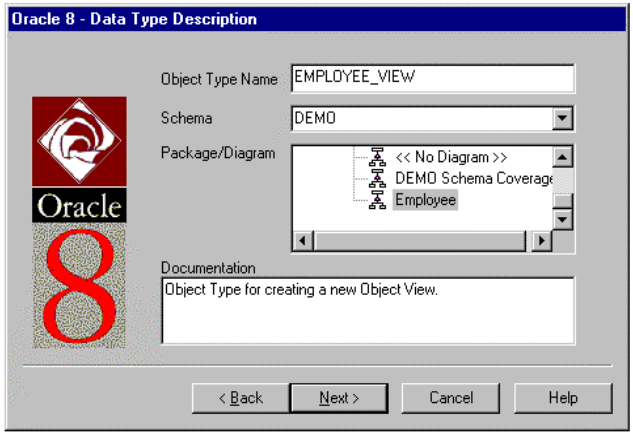

RoseTip

   - Leave the **Schema** set to **DEMO** and **Logical Package** set to **DEMO**.

   *When Rational Rose Oracle8 reverse engineered the **DEMO** database (schema), it created the **DEMO Schema Coverage** diagram and placed in a logical package called **DEMO**.*

   - Select **Employee** as the **Diagram** to which you are adding the ObjectType. (You are adding the object to the class diagram you created in Lesson 2.)
   - (Optional.) Enter **`Object Type for creating a new Object View`** as the description of the ObjectType in the **Documentation** field.

   The Wizard dialog box should look like this:



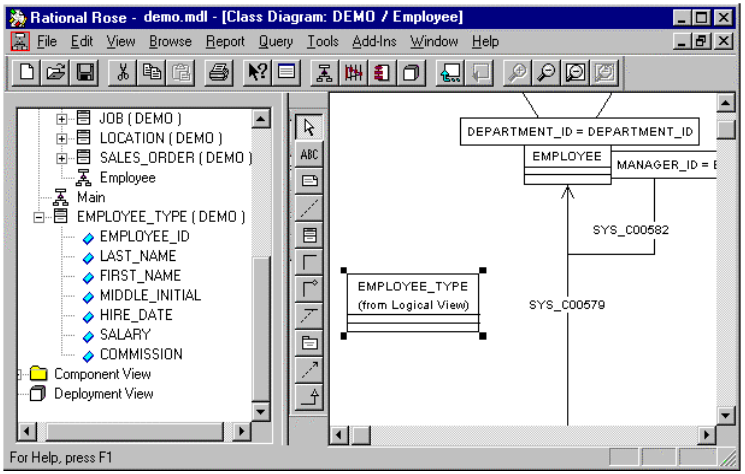
8. Click **Next** to display the **Define Attribute** dialog box.
9. Select the EMPLOYEE table name under Map From and click **Add**. This adds all of the table's columns to the object type's list of attributes. Note that the EMPLOYEE table is displayed here

because you selected it before starting the wizard. If you hadn't selected it, you could use the Map From button to display a list of all of the tables in the Demo schema.

The **Define Attribute** page of the wizard dialog box should now look like this:

10. Click **Next** to display the **Define Operations** dialog box. At this point, you could use this dialog box to create methods for the object type, including functions, procedures, comparison methods, etc. For our example, we won't be creating new methods.
11. Click **Next** to display the **Column/Attribute Ordering** dialog box. This dialog box displays the order in which the object type's attributes are defined. For our example, the order is fine.
12. Click **Finish** to add the **`EMPLOYEE_TYPE`** ObjectType to the **Employee** diagram**.**
13. Click **Tools > Layout Diagram**.
14. Go to the browser window and do the following:
    - Expand **Logical** and notice that the **`EMPLOYEE_TYPE`** object has been added to the model.

- Expand the **EMPLOYEE_TYPE** object and notice that its attributes have been added to the model.



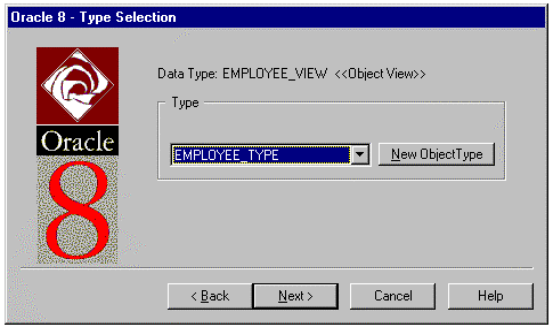
15. Save the model.

Part II: Creating an Object View

1. To create the object view that will “objectify” the Employee data, select the Employee relational table and the new Employee_Type object type on the Employee class diagram. (To select more than one item on a diagram, press and hold the CTRL key while you click on objects in the diagram.)
2. Click **Tools** > **Oracle8** > **Data Type Creation Wizard**.
3. Select the object to add to the diagram, in this case ObjectView. Note that since you selected objects before starting the wizard, they appear in the Selected Items list.
4. Click **Next** to continue.
5. Fill in the information to tell the Data Type Creation Wizard the name of the object you are creating and where to put it in the model:
   - Enter the Name of the ObjectView you want to add, in this case, EMPLOYEE_VIEW.
   - Leave the Schema set to DEMO and Logical Package set to DEMO.

- Select Employee as the Diagram to which you are adding the Object View.
- (Optional.) Enter Object View for relational table EMPLOYEE as the description of the Object View in the Documentation field.

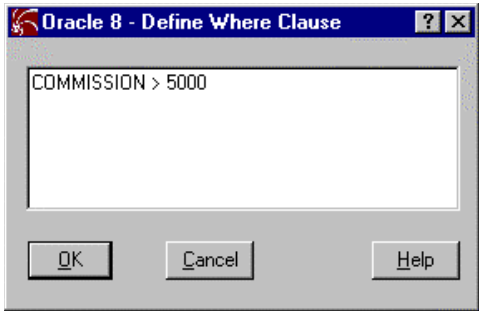
6. Click **Next** to continue.
7. From the **Type Selection** dialog box, select the Object Type the new Object View will be based on. In this case, the Object Type is EMPLOYEE_TYPE. Since you selected it before starting the wizard, its name should already appear as the selected type.



8. Click **Next** to continue.
9. The **Object View Map** dialog box enables you to map the specific Object Type attributes to the correct columns from the underlying relational table. The EMPLOYEE_TYPE Object Type's attributes are displayed in the list on the left. The columns of the EMPLOYEE Relational Table appear in the list on the right. Map the Object Type's attributes to the correct table column by:
   - Clicking on the attribute
   - Clicking on its corresponding column name, then
   - Clicking **Map**

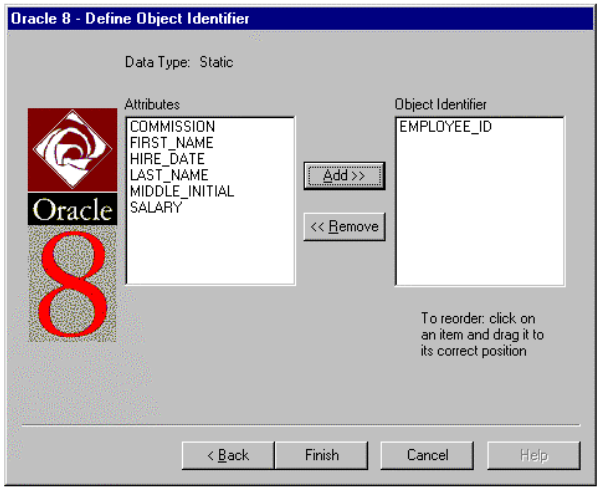
As you map attributes to columns, the attribute list empties. *You must map every attribute to a column (the list on the left must be emptied). The View Map at the bottom of the dialog box displays the mapping you created.*

10. Define a WHERE clause for the view by clicking **Where**.
11. On the **Define Where Clause** dialog box, enter the expression: COMMISSION > 5000.



12. Click **OK** to continue. This returns you to the Object View Map dialog box.
13. Click **Next** to continue.

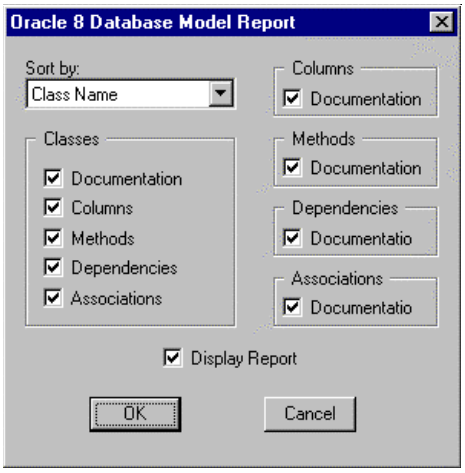
14. From the **Define Object Identifier** dialog box, select EMPLOYEE_ID as the Identifier to use for the new view. The Object View uses the Object Identifier to enable REF's to point to objects (rows) in the view. (An Object Identifier can be a composite of more than one attribute.)



15. Click **Finish** to add the new Object View to the EMPLOYEE diagram.

Part III: Generating a Report

1. To create a report of the objects in your schema, click **Tools** > **Oracle8** > **Reports**.
2. On the Oracle 8 **Database Model Report** dialog box, click **OK**.



3. From the **Save** dialog box, enter MYDEMO as the name of the saved report and click Save. By default, Rational Rose always saves reports with the .or8 file extension.
4. The report is saved and is displayed in Notepad.
5. Close Notepad when you've completed viewing the report.
6. Save your model.

Lesson 4 Updating Your Database Schema

At this point, you have used Rational Rose Oracle8 to:

- Analyze your current database and create a model of its schema
- Create a diagram of a subset of the database with which to work
- Add Oracle8 objects to your database model
- Generated a report based on your model

What's next? What you really want is an updated database schema that includes the new Oracle8 objects that you added to the model.

Rational Rose Oracle8 will check the syntax of your updated model and then let you *forward engineer your new objects to the Oracle8 Demo schema.*

Rational Rose Oracle8 does this by generating a DDL script, connecting to your Oracle database server, then executing the script against your schema.

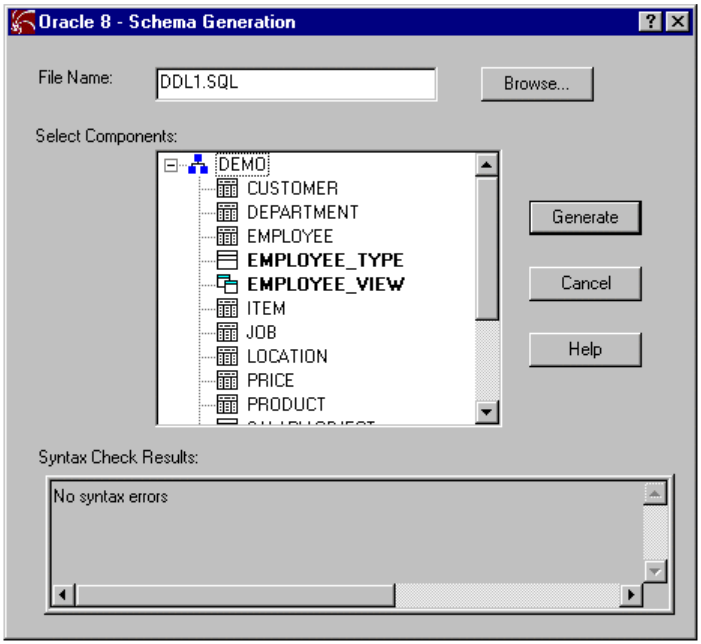
Follow these steps to update your database schema from Rational Rose Oracle8:

1. If your model is no longer open, Click **File > Open** to open it.
2. Display the Employee diagram by doing the following:
   - Click the  button on the toolbar to display the **Select Class Diagram** dialog box.
   - Select **DEMO** under **Class Category** and **Employee** under **Class Diagrams**.
   - Click **OK**.

   Select the EMPLOYEE_TYPE Object Type and EMPLOYEE_VIEW Object View in the Employee diagram. These are the objects we will add to the Oracle8 Demo schema. Click **Tools** > **Oracle8** > **Generate Schema**.
3. When the **Schema Generation** dialog box appears, expand the tree in the Select Components area of the window.

   Notice that the entire model structure is shown and the EMPLOYEE_TYPE and EMPLOYEE_VIEW objects that you selected in the diagram appear in boldface type. Also notice that the entire

model was checked for syntax automatically as a result of selecting Schema Generation. The results of the syntax check are displayed in the lower half of the dialog box.

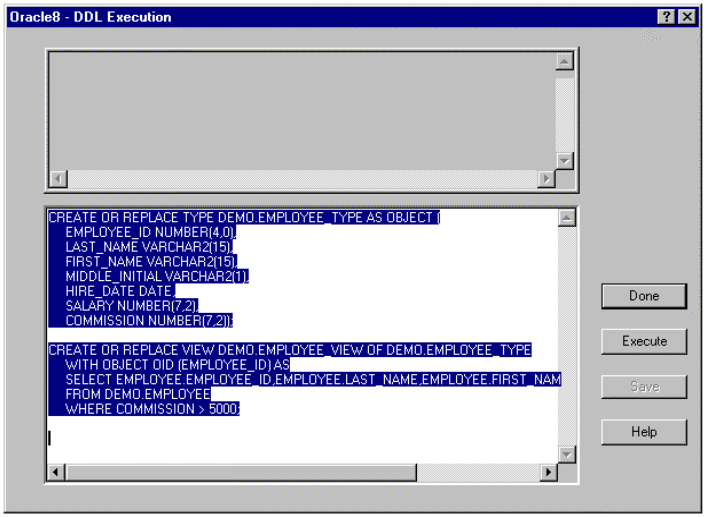



RoseTip

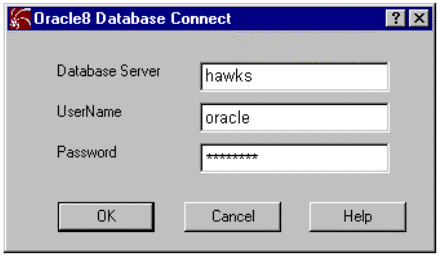
You can use this window to view the objects that are currently selected for schema generation and you can change those selections, if you need to.

4. Click **Generate** to continue.
5. This displays the **DDL Execution** dialog box. The top half of the dialog box displays any errors that the generator encountered. The bottom half displays the DDL script.

To execute the DDL, leave the statements highlighted. (Statements must be selected in order to be executed.



6. Click **Execute**.
7. The **Oracle8 Database Connect** dialog box prompts you for the Database Server Name, your User Name, and Password. Click **OK** once you completed these fields.



8. A connection is made to your database and the statements are executed. The status appears in the upper half of the **DDL Execution** dialog box.

   Generation is complete when the status indicates that the statements have been successfully executed against your schema.
9. Save your model.

Index

A

aggregate association 7
    in a relational table 18
alias 11
analyzing a Rose model 43
analyzing an Oracle8 schema 23
associations
    aggregate, for REFs 7
    for a nested table 15
    in relational tables 17
AttributeNamePrefix property 55
AttributeNameSuffix property 55
attributes
    and relational tables 17
    creating in an object type 30
    how modeled in Rose 7
    in a relational view 21
    in an object table 11
    in an object view 10
    in object types 5, 6
    index in a relational table 20
    mapping for an object view 32
    mapping in an object type 30
    re-ordering 40
    schema generation properties 51, 57
    using a nested table 14
    using a VARRAY 13
    viewing 40

C

CheckConstraint property 56, 58
checking model syntax 38
class
    base, loading 27
class diagram, creating a new 25
class properties 51
class stereotypes, about 3, 4
collection datatype 12
collection type 14
collection types 6
CollectionTypeLength property 56
CollectionTypePrecision property 56
CollectionTypeScale property 56
columns
    in a relational table 17
    in a relational view 21
    indexing 20
    mapping in a relational table 35
    nested datatypes 17
    re-ordering 40
    schema generation properties (attributes) 57
    viewing 40
comparison methods 6
CompositeUnique property 58
connecting to an Oracle8 Database Server 24

connecting to the Oracle Database Server 45
constraints
  defining in a relational table 36
  defining in an object type 31
  foreign keys 19
  NULL and NOT NULL 7, 19
  REFs 7
  unique 19
create a new class diagram 25
creating a nested table 34
creating a new object view 31
creating a new relational table 35
creating a new relational view 37
creating a new VARRAY 33
Creating an Object Type 30

D

Data Type Creation Wizard
  about 29
DDL
  and forward engineering 43
  generation errors 38
DDLScriptFilename property 52
dependencies
  in a relational view 21
  in a VARRAY 13
  in an object table 12
  in object views 10
diagrams
  creating a new class 25
DropClause property 52

F

FK 36
Foreign Key Wizard 40
Foreign Key Wizard, about 40
foreign keys
  creating 40
  creating in a relational table 36
  editing existing 40
  how modeled in Rose 19
forward engineering
  about 43
  steps for completing 44
framework
  loading scalar datatype classes 28
functions 5, 6, 8

G

generating reports 39

I

identifier, object 9
implementation access control 20
import
  scalar data types 29
index
  creating in a relational table 36
  how modeled in Rose 20
  schema generation property 58
IsIndex property 58
IsPrimaryKey property 58
IsReadNoDataState property 56
IsReadNoProcessState property 57
IsSchema property 38, 58
IsSelfish property 57
IsUnique property 19, 57
IsWriteNoDataState property 57
IsWriteNoProcessState property 57

L

Length property 58

M

map methods 5
MemberNamePrefix property 55
MemberNameSuffix property 55
MethodKind property 8, 56
Methods
  in object types 5, 6
methods
  defining for an object type 31
model properties 52
model, checking the syntax 38
module properties 51

N

nested object type 7
nested table
  creating a new 34
  how modeled in Rose 15
  in a relational table 17
NestedTableNamePrefix property 54
NestedTableNameSuffix property 54
NOT NULL constraint 7, 19
NULL constraint 7, 19
NullsAllowed property 7, 19, 57

O

Object Identifier
  property setting 10
object identifier
  about 9
object table
  creating a new 33
object type
  about 5
  as a VARRAY datatype 13
  creating a new 30
  how modeled in Rose 6
  in a nested table 15
  in an object table 11
  in an object view 9, 10
  nested 7
  using to build object views 9
  viewing/modifying the attribute order 40
object view
  about 9
  creating a new 31
  how modeled in Rose 10
object-identifier
  creating 32
ObjectTableNamePrefix property 54
ObjectTableNameSuffix property 55
OID property 56
operation properties 51
operations
  defining for an object type 31
oracle8.pty 4
order methods 5
ordered collection 12
Ordering Wizard
  about 40
  steps for using 40
OrderNumber property 56, 57, 58
OverloadID property 56

P

Precision property 58
PrimaryKeyColumnName property 52
PrimaryKeyColumnType property 52
procedures 5, 6, 8
project (Model) properties 51
project properties 52
properties
  about 51
  AttributeNamePrefix 55
  AttributeNameSuffix 55
  CheckConstraint 56, 58
  classes 56
  CollectionTypeLength 56
  CollectionTypePrecision 56
  CollectionTypeScale 56

CompositeUnique 58
DDLScriptFilename 52
DropClause 52
IsIndex 58
IsPrimaryKey 58
IsReadNoDataState 56
IsReadNoProcessState 57
IsSchema 58
IsSelfish 57
IsUnique 57
IsWriteNoDataState 57
IsWriteNoProcessState 57
Length 58
MemberNamePrefix 55
MemberNameSuffix 55
MethodKind 56
NestedTableNamePrefix 54
NestedTableNameSuffix 54
NullsAllowed 57
ObjectTableNamePrefix 54
ObjectTableNameSuffix 55
OID 56
operations 56
OrderNumber 56, 57, 58
OverloadID 56
Precision 58
PrimaryKeyColumnName property 52
PrimaryKeyColumnType 52
project 52
Scale 58
schema generation 52
SchemaNamePrefix 52
SchemaNameSuffix 53
TableNamePrefix 53
TableNameSuffix 53
TriggerEvent 57
TriggerForEach 57
TriggerRefererencingNames 57
TriggerText 57
TriggerType 57
TriggerWhenClause 57
TypeNamePrefix 53
TypeNameSuffix 53
VarrayNamePrefix 54
VarrayNameSuffix 54
ViewNamePrefix 53
ViewNameSuffix 54
WhereClause 56
property file
about 3, 4
property settings
about 4

R

REFs
and object identifiers 9
creating in a relational table 35
creating in an object type 31
in a relational table 18
in object types 7
using the Ordering Wizard to view 40
relational table
about 16
creating a new 35
creating new foreign keys 40
editing foreign keys 40
how modeled in Rose 17
in an object view 9, 10
using a nested table in 14
using VARRAYs in 13
viewing/modifying the column order 40
relational view
about 20
creating a new 37
how modeled in Rose 21
viewing/modifying the column order 40
reports, generating 39
return type 8

reverse engineering
about 23
steps for completing 23
role properties 51

S

scalar datatypes 6
using a framework to load 28
scalar datatypes, loading in a model 27
Scale property 58
Schema Generation 44
schema generation properties 56
about 51
attributes 57
classes 56
module specifications 58
operations 56
project 52
role 58
SchemaNamePrefix property 52
SchemaNameSuffix property 53
specification, displaying 24
stereotypes, class 3, 4
Syntax Checker 38

T

TableNamePrefix property 53
TableNameSuffix property 53
TriggerEvent property 57
TriggerForEach property 57
TriggerRefererencingNames property 57
triggers 6, 8
TriggerText property 57
TriggerType property 57
TriggerWhenClause property 57
TypeNamePrefix property 53
TypeNameSuffix property 53

U

UML 3, 23
unique constraint 8, 19
unordered collection 14

V

VARRAY
about 12
creating a new 33
how modeled in Rose 13
in a relational table 16, 17
VarrayNamePrefix property 54
ViewNamePrefix property 53
ViewNameSuffix property 54

W

WhereClause property 56