

## **Oracle® SQL Developer**

Data Modeler User's Guide

Release 2.0

**E13677-01**

June 2009

Provides conceptual and usage information about Oracle SQL Developer Data Modeler, a data modeling and database design tool that provides an environment for capturing, modeling, managing, and exploiting metadata. It is a for-cost option of SQL Developer, and requires a separate license.

Oracle SQL Developer Data Modeler User's Guide, Release 2.0

E13677-01

Copyright © 2008, 2009, Oracle and/or its affiliates. All rights reserved.

Primary Author: Chuck Murray

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

---

# Contents

## 1 Data Modeler Concepts and Usage

1.1	Installing and Getting Started with SQL Developer Data Modeler .....	1-1
1.2	Data Modeler User Interface .....	1-2
1.2.1	Menus for Data Modeler.....	1-4
1.2.2	Context Menus .....	1-6
1.3	Working with SQL Developer Data Modeler .....	1-8
1.3.1	Database Design.....	1-8
1.3.2	Data Types Model.....	1-9
1.3.2.1	Data Types Diagram and Subviews.....	1-10
1.3.2.2	Distinct Types.....	1-10
1.3.2.3	Structured Types.....	1-11
1.3.2.4	Collection Types .....	1-11
1.3.2.5	Logical Types .....	1-11
1.3.3	Process Model.....	1-11
1.3.3.1	Data Flow Diagrams .....	1-12
1.3.3.2	Transformation Packages .....	1-13
1.3.4	Logical Model.....	1-13
1.3.4.1	Logical Diagram and Subviews.....	1-13
1.3.4.2	Entities.....	1-14
1.3.4.3	Attributes .....	1-14
1.3.4.4	Unique Identifiers (UIDs).....	1-14
1.3.4.5	Inheritances .....	1-14
1.3.4.6	Relations.....	1-15
1.3.4.7	Arcs.....	1-15
1.3.4.8	Type Substitution .....	1-15
1.3.4.9	Views .....	1-15
1.3.5	Relational Models .....	1-15
1.3.5.1	Relational Diagram and Subviews.....	1-16
1.3.5.2	Tables.....	1-16
1.3.5.3	Columns.....	1-17
1.3.5.4	Indexes .....	1-17
1.3.5.5	Relations.....	1-17
1.3.5.6	Relational Views .....	1-17
1.3.6	Physical Models .....	1-17
1.3.6.1	Clusters .....	1-18
1.3.6.2	Contexts .....	1-18

1.3.6.3	Dimensions .....	1-18
1.3.6.4	Directories.....	1-18
1.3.6.5	Disk Groups.....	1-19
1.3.6.6	External Tables.....	1-19
1.3.6.7	Indexes .....	1-19
1.3.6.8	Roles .....	1-19
1.3.6.9	Rollback Segments.....	1-19
1.3.6.10	Segments (Segment Templates).....	1-19
1.3.6.11	Sequences.....	1-19
1.3.6.12	Snapshots .....	1-19
1.3.6.13	Stored Procedures.....	1-20
1.3.6.14	Synonyms .....	1-20
1.3.6.15	Structured Types.....	1-20
1.3.6.16	Tables.....	1-20
1.3.6.17	Tablespaces .....	1-20
1.3.6.18	Users .....	1-20
1.3.6.19	Views .....	1-20
1.3.7	Business Information.....	1-21
1.3.7.1	Contacts.....	1-21
1.3.7.2	Documents.....	1-21
1.3.7.3	Emails .....	1-21
1.3.7.4	Locations .....	1-21
1.3.7.5	Resource Locators.....	1-22
1.3.7.6	Responsible Parties.....	1-22
1.3.7.7	Telephones.....	1-22
1.4	Approaches to Data Modeling .....	1-22
1.4.1	Top-Down Modeling.....	1-22
1.4.2	Bottom-Up Modeling .....	1-24
1.4.3	Targeted Modeling .....	1-25
1.5	Data Modeler Options (User Preferences).....	1-26
1.5.1	General .....	1-26
1.5.2	Model.....	1-26
1.5.3	Diagram.....	1-27
1.5.4	Naming Standard .....	1-28
1.5.5	DDL.....	1-29
1.5.6	Third Party JDBC Drivers.....	1-30
1.6	Saving, Opening, Exporting, and Importing Designs .....	1-30
1.6.1	Importing a DDL File .....	1-31
1.6.2	Importing Cube Views Metadata .....	1-31
1.6.3	Importing from Microsoft XMLA.....	1-31
1.6.4	Importing an ERwin File .....	1-31
1.6.5	Importing from a Data Dictionary .....	1-31
1.6.6	Importing an Oracle Designer Model.....	1-32
1.6.7	Importing a SQL Developer Data Modeler Design .....	1-32
1.6.8	Importing a Domain .....	1-32
1.7	Data Modeler Reports .....	1-32
1.7.1	Design Content reports .....	1-33

1.7.2	Design Rules reports .....	1-33
1.8	Read-Only Data Modeler Viewer .....	1-33
1.9	Additional Resources for Data Modeling.....	1-34

## 2 Tutorial: Data Modeling for a Small Database

2.1	Develop the Logical Model.....	2-1
2.1.1	Adding Domains.....	2-1
2.1.2	Creating the Books Entity .....	2-2
2.1.3	Creating the Patrons Entity .....	2-3
2.1.4	Creating the Transactions Entity .....	2-4
2.1.5	Creating Relations Between Entities.....	2-5
2.2	Develop the Relational Model.....	2-6
2.3	Generate DDL.....	2-6
2.4	Save the Design .....	2-7

## 3 Dialog Boxes for Data Modeler

3.1	Add Event .....	3-1
3.2	Arc Properties.....	3-1
3.3	Attribute Properties.....	3-2
3.4	Change Object Names Prefix.....	3-3
3.5	Change Request Properties.....	3-3
3.6	Collection Type Properties .....	3-4
3.7	Column Properties.....	3-4
3.8	Common Information in Dialog Boxes .....	3-5
3.9	Compare Mapping.....	3-6
3.10	Compare Modeling Designs.....	3-6
3.11	Compare Models.....	3-6
3.12	Contact Properties.....	3-7
3.13	Cube Properties.....	3-7
3.14	Data Dictionary Import (Metadata Extraction) .....	3-9
3.15	Database Connection Editor.....	3-10
3.16	DDL File Editor .....	3-10
3.17	DDL Generation Options.....	3-11
3.18	Design Properties.....	3-11
3.19	Design Rules .....	3-11
3.20	Dimension Properties.....	3-12
3.21	Display Properties.....	3-13
3.22	Distinct Type Properties .....	3-13
3.23	Document Properties.....	3-14
3.24	Domain Properties (Domains Model).....	3-14
3.25	Domains Administration .....	3-15
3.26	Email Properties.....	3-15
3.27	Engineering.....	3-16
3.28	Entity Properties.....	3-17
3.29	Event Properties .....	3-18
3.30	Export Cube Views Metadata .....	3-19

3.31	Export to Microsoft XMLA .....	3-19
3.32	Export to Oracle Analytic Workspaces .....	3-19
3.33	Export to Reporting Schema.....	3-20
3.34	External Agent Properties.....	3-20
3.35	External Data Properties .....	3-21
3.36	Find Object.....	3-22
3.37	Flow Properties .....	3-22
3.38	Foreign Key Properties.....	3-23
3.39	Glossary Editor.....	3-24
3.40	Hierarchy Properties .....	3-25
3.41	Import Domains .....	3-26
3.42	Import Glossary (Naming Standard Definitions) .....	3-26
3.43	Import Mapped Models from VAR Files.....	3-27
3.44	Import Oracle Designer Model .....	3-27
3.45	Import SQL Developer Data Modeler Design .....	3-27
3.46	Import SQL Developer Database Connections.....	3-27
3.47	Import VAR File: Select Type of Import.....	3-28
3.48	Index Properties .....	3-28
3.49	Information Store Properties .....	3-29
3.50	Information Structure Properties.....	3-30
3.51	Join Properties .....	3-31
3.52	Unique Identifier (UID, or Key) Properties.....	3-32
3.53	Level Properties.....	3-32
3.54	Location Properties.....	3-34
3.55	Logical Type.....	3-34
3.56	Measure Folder Properties .....	3-34
3.57	Measure Properties .....	3-35
3.58	Measurement Properties .....	3-36
3.59	Method Properties .....	3-36
3.60	Model Properties - Business Information.....	3-37
3.61	Model Properties - <data-flow-diagram-name> .....	3-37
3.62	Model Properties - Data Types .....	3-37
3.63	Model Properties - Logical.....	3-38
3.64	Model Properties - <multidimensional-model-name> .....	3-38
3.65	Model Properties - Process Model.....	3-38
3.66	Model Properties - <name> (Relational) .....	3-39
3.67	Name Abbreviations.....	3-39
3.68	New/Update Database Connection.....	3-40
3.69	Process Properties .....	3-42
3.70	RDBMS Site Editor.....	3-43
3.71	Record Structure Properties .....	3-44
3.72	Relation Properties.....	3-44
3.73	Relational Models .....	3-45
3.74	Resource Locator Properties.....	3-45
3.75	Responsible Party Properties .....	3-46
3.76	Role Properties .....	3-46
3.77	Rollup Link Properties .....	3-47

3.78	SELECT DDL Files .....	3-47
3.79	Set Classification Types.....	3-47
3.80	Set Data Type.....	3-47
3.81	Slice Properties .....	3-48
3.82	Spatial Definition Properties .....	3-49
3.83	SQL Access to Oracle AW Properties .....	3-49
3.84	Structured Attribute Properties .....	3-50
3.85	Structured Type Properties .....	3-50
3.86	Subview Properties.....	3-51
3.87	Table Properties.....	3-51
3.88	Table to View .....	3-53
3.89	Telephone Properties.....	3-54
3.90	Transformation Package .....	3-54
3.91	Transformation <task-name>.....	3-54
3.92	Transformation Properties.....	3-55
3.93	Transformation Flow Properties.....	3-55
3.94	Types Administration .....	3-56
3.95	View Properties.....	3-56





## List of Figures

1-1	SQL Developer Data Modeler Main Window .....	1-2
-----	--	-----



---

---

# Preface

This guide provides conceptual and usage information about SQL Developer Data Modeler, a data modeling and database design tool that provides an environment for capturing, modeling, managing, and exploiting metadata. SQL Developer Data Modeler is also referred to as Data Modeler.

## Audience

This guide is intended for those using SQL Developer Data Modeler. It assumes that you either have some familiarity with data modeling, or that you can find resources outside this guide for more advanced and detailed information about data modeling.

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

### Accessibility of Code Examples in Documentation

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

### Accessibility of Links to External Web Sites in Documentation

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

### Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at

<http://www.fcc.gov/cgb/consumerfacts/trs.html>, and a list of phone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>.

## Related Documents

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

To download free release notes, installation documentation, white papers, or other collateral, go to the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

<http://www.oracle.com/technology/membership>

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

<http://www.oracle.com/technology/documentation>

## Conventions

The following text conventions are used in this document:

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

## Third-Party License Information

SQL Developer Data Modeler contains third-party code. Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the third-party software, and the terms contained in the following notices do not change those rights.

### Batik Version 1.7

This program contains Batik version 1.7. Batik uses Apache License, Version 2.0, as follows:

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

## 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

## 2. Grant of Copyright License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.





---

---

# Data Modeler Concepts and Usage

SQL Developer Data Modeler is a data modeling and database design tool that provides an environment for capturing, modeling, managing, and exploiting metadata. It is based on the Zachman framework and the Object Management Group (OMG) MetaObject Facility (MOF) and Common Warehouse Metamodel (CMW) specifications.

SQL Developer Data Modeler is a for-cost option of SQL Developer that requires a separate license. However, a free viewer (read-only) version is also available, as explained in [Section 1.8](#).

This chapter contains the following major sections:

[Section 1.1, "Installing and Getting Started with SQL Developer Data Modeler"](#)

[Section 1.2, "Data Modeler User Interface"](#)

[Section 1.3, "Working with SQL Developer Data Modeler"](#)

[Section 1.4, "Approaches to Data Modeling"](#)

[Section 1.5, "Data Modeler Options \(User Preferences\)"](#)

[Section 1.6, "Saving, Opening, Exporting, and Importing Designs"](#)

[Section 1.8, "Read-Only Data Modeler Viewer"](#)

[Section 1.9, "Additional Resources for Data Modeling"](#)

## 1.1 Installing and Getting Started with SQL Developer Data Modeler

To install and start SQL Developer Data Modeler, the process is similar to that for SQL Developer: you download a .zip file and unzip it into a desired parent directory or folder, and then type a command or double-click a file name. You should read any Data Modeler release notes or "readme" file before you perform the following steps.

1. Unzip the Data Modeler kit into a directory (folder) of your choice. This directory location will be referred to as `<datamodeling_install>`. For example, on a Windows system you might want to choose `C:\Program Files` as this location.

Unzipping the Data Modeler kit causes a directory named `datamodeling` to be created under the `<datamodeling_install>` directory. It also causes many files and folders to be placed in and under that directory.

2. To start Data Modeler, go to the `datamodeling` directory under the `<datamodeling_install>` directory, and do one of the following:

On Linux and Mac OS X systems, run `sh datamodeling.sh`.

On Windows systems, double-click `datamodeling.exe`.

If you are asked to enter the full pathname for java.exe, click **Browse** and find java.exe. For example, on a Windows system the path might have a name similar to C:\Program Files\Java\jdk1.6.0\_06\bin\java.exe.

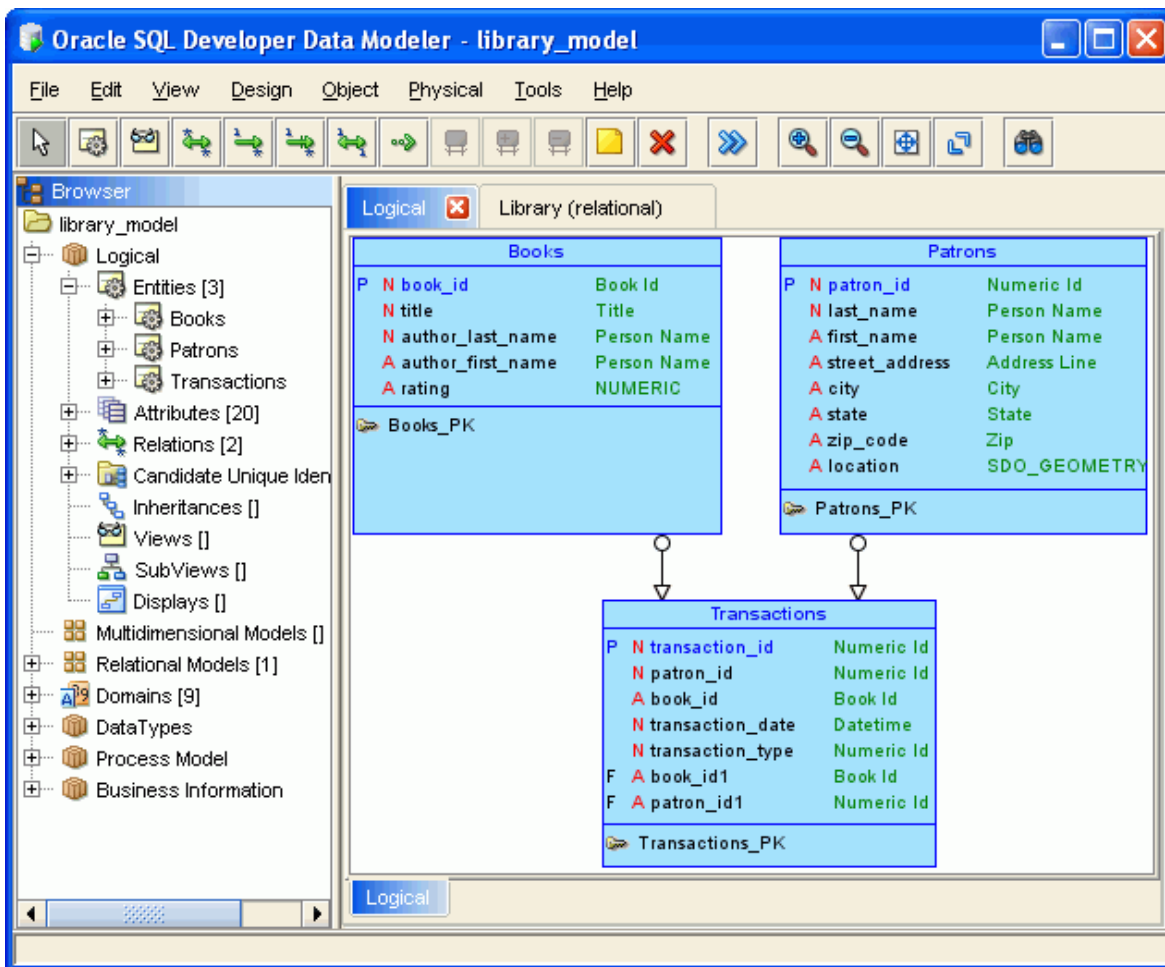
3. If you want to become familiar with data modeling concepts before using the interface, read the rest of this chapter before proceeding to the next step.
4. Do the short tutorial in [Chapter 2, "Tutorial: Data Modeling for a Small Database"](#). (For more advanced tutorials and other materials, see [Section 1.9, "Additional Resources for Data Modeling"](#).)

## 1.2 Data Modeler User Interface

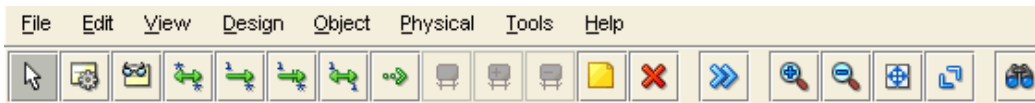
The SQL Developer Data Modeler window generally uses the left side for navigation to find and select objects, and the right side to display information about selected objects.

[Figure 1-1](#) shows the main window.

**Figure 1-1** SQL Developer Data Modeler Main Window



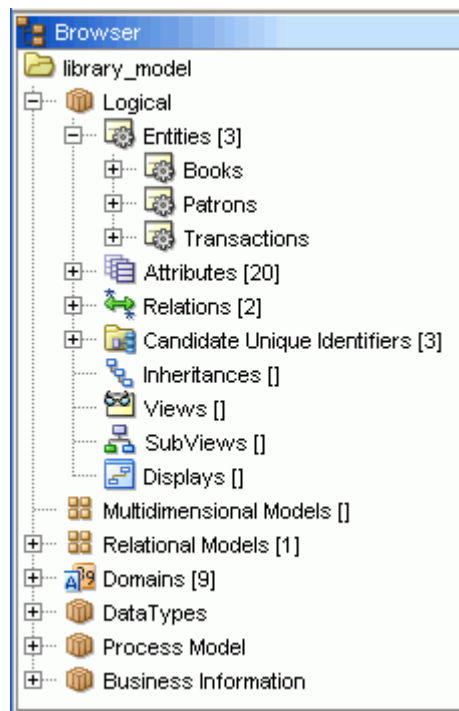
The menus at the top contain some standard entries, plus entries for features specific to SQL Developer Data Modeler (see [Section 1.2.1, "Menus for Data Modeler"](#)), as shown in the following figure.



You can use **shortcut keys** to access menus and menu items: for example Alt+F for the File menu and Alt+E for the Edit menu; or Alt+H, then Alt+C for Help, then Contents. You can also display the File menu by pressing the F10 key.

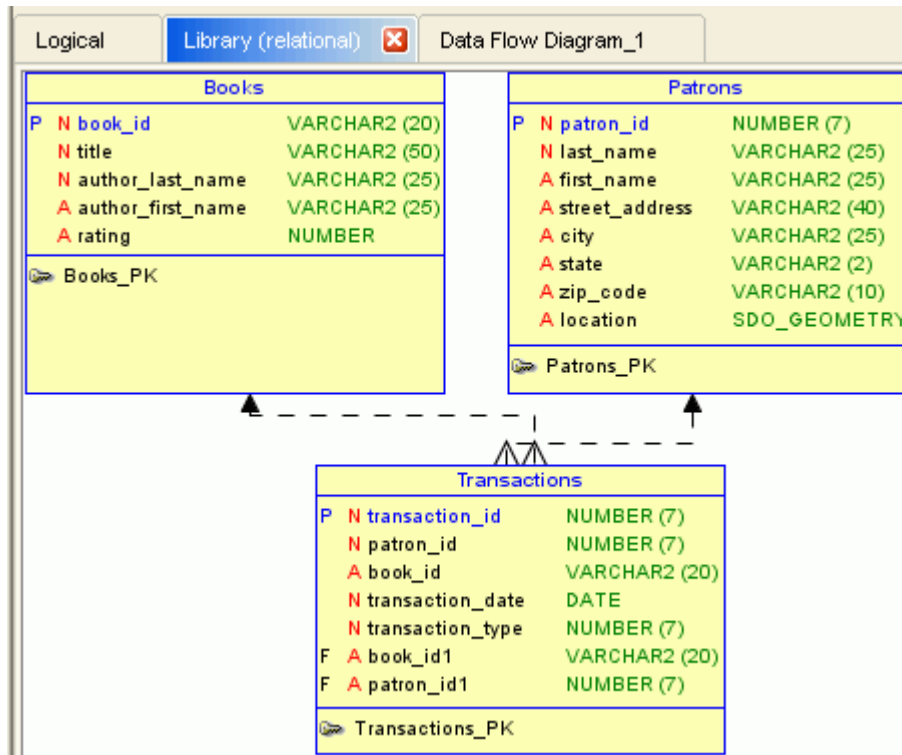
Icons under the menus perform actions relevant to what is currently selected for display on the right side of the window, such as the logical model, a relational model, or a data flow diagram. For example, for a relational model the icons include New Table, New View, Split Table, Merge Tables, New FK Relation, and Generate DDL. To see the name of any icon, hover the pointer over the icon. The actions for the icons are also available from the **Object** menu.

The left side of the Data Modeler window has an object browser with a hierarchical tree display for data modeling objects, as shown in the following figure.



To select an object in the object browser, expand the appropriate tree node or nodes, then click the object.

The right side of the Data Modeler window has tabs and panes for objects that you select or open, as shown in the following figure, which displays information about a deliberately oversimplified relational model for library-related data (the model developed in [Chapter 2, "Tutorial: Data Modeling for a Small Database"](#)).



To switch among objects, click the desired tabs; to close a tab, click the X in the tab. If you make changes to an object and click the X, you are asked if you want to save the changes.

## 1.2.1 Menus for Data Modeler

This topic explains menu items that are of special interest for Data Modeler .

### File menu

**Open:** Opens a SQL Developer Data Modeler design that had been saved or exported. For more information, see [Section 1.6, "Saving, Opening, Exporting, and Importing Designs"](#).

**Close:** Closes the current design without exiting SQL Developer Data Modeler.

**Import:** Lets you import models from a variety of sources. For more information, see [Section 1.6, "Saving, Opening, Exporting, and Importing Designs"](#).

**Export:** Lets you export models to files that can be imported into a variety of data modeling tools. For more information, see [Section 1.6, "Saving, Opening, Exporting, and Importing Designs"](#).

**Print:** Prints the currently selected diagram.

**Print Diagram:** Saves the currently selected diagram to an image file of the type associated with the file extension that you specify (.png or.jpg), to a PDF file, or to a scalable vector graphics (.svg) file.

**Recent Designs:** Lets you open a SQL Developer Data Modeler design that you recently worked on.

### View menu

Contains options that affect what is displayed in the SQL Developer Data Modeler interface.

**Browser:** Displays the object browser, which shows data modeling objects in a hierarchical tree format.

**Navigator:** Displays a graphical thumbnail representation of the view that is currently selected in the right side of the window.

**Log:** Displays the Log pane with a record of SQL Developer Data Modeler actions.

**View Details:** Controls the level of detail in displays.

**Logical Diagram Notation:** Controls whether Barker or Bachman notation is used to display the logical model.

**DDL File Editor:** Lets you generate DDL statements for a selected physical model. Displays the [DDL File Editor](#) dialog box. (This command is equivalent to clicking the Generate DDL icon when a relational model is selected, or clicking Object, then Relational, then Generate DDL.)

**Zoom In** (and corresponding icon): Displays more detail, and potentially fewer objects, in the currently selected diagram.

**Zoom Out** (and corresponding icon): Displays less detail, and potentially more objects, in the currently selected diagram.

**Fit Screen** (and corresponding icon): Makes all relevant objects fit in the window for the currently selected diagram, adjusting the sizes of shapes and text labels as needed.

**Find:** Displays a dialog box for finding objects in the currently selected diagram. Useful for finding objects in large, complex diagrams.

### Design menu

Contains options for generating models.

**Engineer to Relational Model:** Performs forward engineering: generates or updates a relational model from the logical model. You can also specify if the operation creates a subview.

**Engineer to Logical Model:** Performs reverse engineering: updates the logical model from the selected relational model.

### Object menu

Contains commands for performing the actions that are available for the currently selected diagram (for example, for the logical model, a relational model, or a data flow diagram). The icons are displayed under the menus and above the tabs for selecting diagrams to view.

### Physical menu

Contains commands for opening, closing, and saving a physical model.

### Tools menu

Invokes SQL Developer Data Modeler tools and lets you set certain options (user preferences).

**Domains Administration:** Lets you view, modify, add, and delete domains. Displays the [Domains Administration](#) dialog box.

**Types Administration:** Lets you view, modify, add, and delete logical types. Displays the [Types Administration](#) dialog box.

**RDBMS Site Administration:** Lets you view RDBMS sites (names associated with supported types of databases), and to add your own names (aliases) for convenience in creating physical models. Displays the [RDBMS Site Editor](#) dialog box.

**Table to View Wizard:** Lets you create views based on tables in a selected relational model. Displays the [Table to View](#) wizard.

**Name Abbreviations:** Specifies a .csv file with strings to be changed in names of relational model objects (for example, to ensure the use of standard abbreviations or spellings). Displays the [Name Abbreviations](#) dialog box.

**Glossary Editor:** Lets you create a new glossary file (if you specify a file name that does not exist) or edit an existing glossary file. Displays a file selection dialog box, and then the [Glossary Editor](#) dialog box.

**Design Rules:** Lets you check your current design for violations of Data Modeler design rules. Displays the [Design Rules](#) dialog box.

**Engineering Status:** Displays the [Engineering](#) dialog box.

**Compare/Merge Models:** Lets you open a design file, compare a relational model from the file with a relational model in the current design, and merge objects from one model into the other. After you select the design file, the [Relational Models](#) dialog box is displayed.

**General Options:** Lets you customize the behavior of SQL Developer Data Modeler. Displays the [Data Modeler Options \(User Preferences\)](#) dialog box.

### Help menu

Displays help about SQL Developer Data Modeler.

**Contents:** Displays the help window.

**About:** Displays version-related information about SQL Developer Data Modeler.

## 1.2.2 Context Menus

The **context menus** (right-click menus) in the object browser and diagrams contain commands relevant for the object or objects selected.

In the object browser, if you right-click the logical model or a relational model, the context menu generally includes the following:

- **Set Classification Types:** For entities or tables in a multidimensional model, lets you set the classification type for each: Fact, Dimension, Logging, Summary, or Temporary. (You can also specify colors to be used in diagrams for each classification type in the [Diagram: Classification Types](#) user preference.)
- **Apply Naming Standards to Keys and Constraints** (relational models): Applies naming standardization rules specified for [Relational - Others](#) in the [Naming Standard](#) preferences to keys and constraints in the selected relational model.
- **Change Object Names Prefix:** Specifies the new prefix to replace a specified current prefix for selected types of objects. Displays the [Change Object Names Prefix](#) dialog box.

In diagrams, if you right-click outside any displayed object, the context menu generally includes the following:

- **AutoLayout** (relational and data flow diagrams): Rearranges the objects in the diagram to a layout that may be more meaningful and attractive. If you do not like the rearrangement, you can restore the previous layout by clicking Edit, then Undo AutoLayout.
- **Auto Route**: Toggles the setting of the Line Auto Route option (see [Diagram](#) under [Data Modeler Options \(User Preferences\)](#)). You must disable Auto Route before you can adjust lines in diagrams, such as clicking and dragging edges and elbows (vertices) to move them, or Ctrl+clicking and dragging on an edge to create a new elbow. Note: If you then enable Auto Route, any manual adjustments are lost.
- **View Details**: Lets you view all available details for objects or only selected details.
- **Model Properties**: Displays the dialog box for viewing and editing properties of the model.

In diagrams, if you right-click a line connecting two objects object, the context menu generally includes the following:

- **Delete**: Removes the line and deletes the relationship represented by the line.
- **Straighten Lines** (available only if Auto Route is disabled): Removes any elbows so that the line contains only the start and end points.
- **Format**: Lets you change the width and color of the line.
- **Add Elbow** (available only if Auto Route is disabled): Adds an elbow (vertex) at the selected point.
- **Remove Elbow** (available only if Auto Route is disabled): Removes the selected elbow (vertex).
- **Properties**: Displays the dialog box for viewing and editing properties of the relationship represented by the line.

In the logical and relational diagrams, if you select one or more entities or tables and right-click one of them, the context menu includes at least the following:

- **Create Synonym**: Creates a synonym object in the display.
- **Create Subview from Selected**: Creates a subview containing the selected objects. (See also [Section 1.3.4.1, "Logical Diagram and Subviews"](#) and [Section 1.3.5.1, "Relational Diagram and Subviews"](#).)
- **Select Neighbors**: Selects objects that are related to the selected object or objects. You can specify the selection direction: All (higher- and lower-level zones), Parent, or Child. You may want to select neighbors before creating a subview from the selection.
- **DDL Preview** (relational diagrams): Shows the DDL statement that would be generated to create the object.
- **Format**: Lets you specify colors and fonts for the selected objects.
- **Send to Back**: Sends the selected objects to the back of the diagram display, which may cause them to be partially or completely covered by other objects.
- **Properties**: Displays the dialog box for viewing and editing properties of the object.

In data flow diagrams, if you select one or more objects and right-click one of them, the context menu includes at least the following:

- **Delete**: Deletes the selected object.

- **Format:** Lets you specify colors and fonts for the selected objects.
- **Send to Back** (for objects not represented by lines): Sends the selected objects to the back of the diagram display, which may cause them to be partly or completely covered by other objects.
- **Properties:** Displays the dialog box for viewing and editing properties of the object.

## 1.3 Working with SQL Developer Data Modeler

You can use SQL Developer Data Modeler to create, edit, and delete objects at different hierarchy levels in different kinds of models. Many objects have similar properties, and the methods for performing operations are usually consistent and intuitive. To perform operations on objects (create, edit, delete), you can often use the context menu in the object browser or the toolbar or the Object menu after selecting a diagram.

- To perform an operation on an object using the object browser, right-click the appropriate node (or click the node and press Shift+f10) in the hierarchy, and select the command for the desired operation.

For example, to edit an entity, expand the Logical display so that all entities are visible, right-click the name of the entity to be edited, and select **Properties**.

- To perform an operation using a diagram, select the tab for the diagram, and use either the toolbar icons or the Object menu commands.

For example, to create an entity, select the Logical tab; either click the New Entity toolbar icon or click **Object**, then **Logical**, then **New Entity**; then define the entity in the [Entity Properties](#) box. To edit an entity, either double-click its box in the diagram or right-click the box and select **Properties**.

[Context Menus](#) (right-click menus) in diagrams contain commands relevant for either the diagram generally or the object or objects currently selected.

For conceptual and usage information about specific kinds of objects, see the following topics:

- [Section 1.3.1, "Database Design"](#)
- [Section 1.3.2, "Data Types Model"](#)
- [Section 1.3.3, "Process Model"](#)
- [Section 1.3.4, "Logical Model"](#)
- [Section 1.3.5, "Relational Models"](#)
- [Section 1.3.6, "Physical Models"](#)
- [Section 1.3.7, "Business Information"](#)

### 1.3.1 Database Design

SQL Developer Data Modeler works with one open database design, consisting of one logical model, optionally one or more relational models based on the logical model, and optionally one or more physical models based on each relational model. The database design can also include a data types model, and business information. To work on another database design, close the current design (click **File**, then **Close**), and create or import objects for the other database design.



When you save a database design, the structural information is stored in an XML file in a folder or directory that you specify, and subfolders or subdirectories are created as needed under it. The XML file contains pointers to information in these subfolders or subdirectories. For example, for a very basic design named `my_db_model`, the following hierarchy might be created starting at the folder or directory in which you created it:

```
my_db_model.xml
my_db_model
  businessinfo
  datatypes
  subviews
  logical
  entity
  subviews
  mapping
  pm
  rdbms
  rel
  1
  subviews
  table
```

Additional subfolders or directories may be created later, for example, `dataflows` under `pm` if you create any data flow diagrams in the process model.

### 1.3.2 Data Types Model

SQL Developer Data Modeler supports supertypes and subtypes in its logical model, but it also provides the data types model, to be CWM compliant and to allow modeling of SQL99 structured types, which can be used in the logical model and in relational models as data types.

Structured types are supported as named user-defined composite types with the possibility of building a supertype/subtypes inheritance hierarchy. You can create and visualize structured types and the inheritance hierarchies of structured types, defining distinct and collection (array) types.

Both logical and relational models can use definitions from the data types model to specify the data type for attributes and columns or to define that a table (entity) is of a certain structured type.

You can build the data types model in one or more of the following ways:

- Manually in SQL Developer Data Modeler
- By importing from Oracle Designer repository. See [Section 1.6.6, "Importing an Oracle Designer Model"](#).

The data types model in SQL Developer Data Modeler combines two kinds of data:

- One data types diagram, plus an optional set of subviews and auxiliary displays, each associated with the appropriate diagram/subview
- Data type object definitions

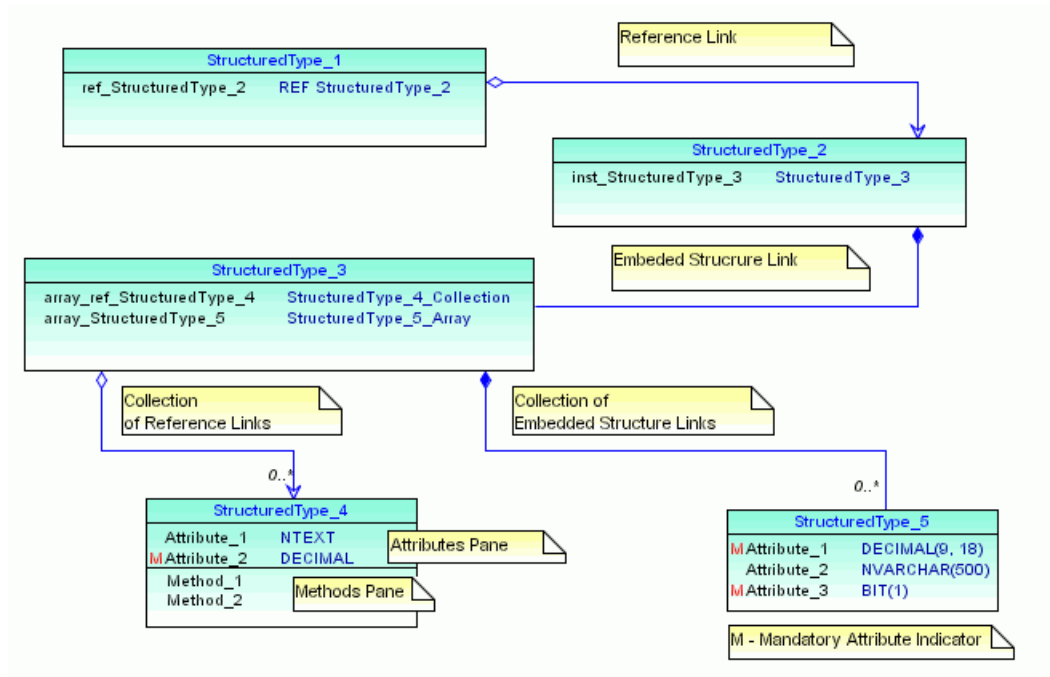
Subviews are considered as independent diagrams of the data types model, created to represent different subject areas.

The data types model enables you to create and manage object definitions of distinct, structured, collection, and logical types.

All data type model objects (except logical types) are displayed in the object browser tree, but only structured type objects and their interrelations are represented graphically on data types diagrams.

### 1.3.2.1 Data Types Diagram and Subviews

The data types diagram contains graphical representations of structured data types and links between them, as shown in the following figure.



A structured type box contains the name of the object, its defined attributes, and its methods (if any). Diagram links represent various kinds of attributes with a structured data type.

When you are working with a complicated data types model, you may want to create **subviews**, with each subview describing only a section of that model. You can define several data types subviews for a single data types model, and you can assign a structured type to more than one subview. However, links (references) between two structured types are displayed on the complete data types model and only on subviews to which both types have been assigned.

There is no difference between performing changes in a subview or in the complete data types model. Any changes made are immediately reflected in the complete model and any relevant subviews. However, you can remove a structured type from a subview without deleting it from the data types model.

### 1.3.2.2 Distinct Types

A user-defined distinct type is a data type derived from an existing logical type, defined in [Types Administration](#) dialog box. A distinct type shares its representation with an existing type (the source type), but is considered to be a separate and incompatible type.

A distinct type object can be accessed only in the Distinct Types subfolder of the Data Types folder.

You can create new distinct types or edit the properties of existing distinct types.

### 1.3.2.3 Structured Types

Structured types are user defined data types have attributes and methods. They also can be part of a supertype and subtype inheritance hierarchy. A structured type can be defined based on a basic data type, a distinct type, another structured type, or a reference to structured type, or it can be defined as a collection type.

A table or entity can be defined as based on a structured type. Type substitution enables you to describe (graphically on a diagram) instances of which subtypes can be accommodated by the table (entity).

Table column or entity attributes can be defined as based on a structured type, a reference to structured type, a collection type, a distinct type, and basic data types. Type substitution can be defined for a column based on a structured type, and a scope table can be defined for a column based on a reference to a structured type.

A structured type also includes a set of method specifications. Methods enable you to define behaviors for structured types. Like user-defined functions (UDFs), methods are routines that extend SQL. In the case of methods, however, the behavior is integrated solely with a particular structured type.

The expanded structured types subfolder lists all structured type objects, with the hierarchy of attributes and methods for each.

The Oracle Spatial SDO\_GEOMETRY type is predefined as a structured type. In addition, you can create new structured types or edit the properties of existing structured types.

### 1.3.2.4 Collection Types

Collection types represent arrays or collections of elements (basic type, distinct type, structured type, or another collection) and are mapped to the Oracle VARRAY and nested table types.

You can create new collection types or edit the properties of existing collection types.

### 1.3.2.5 Logical Types

Logical types are not actual data types, but names that can be associated with native types or with domains. The presupplied logical types include several from Oracle Multimedia (names starting with ORD); however, ORDIMAGE\_SIGNATURE is deprecated and should not be used for new definitions.

You can create logical types and edit their mappings to native types (see [Section 3.94, "Types Administration"](#)), and you can associate a domain with a logical type (see [Section 3.25, "Domains Administration"](#)).

## 1.3.3 Process Model

The process model represents a functional area of an information structures system. The process model, embodied graphically in one or more data flow diagrams, is an analysis technique used to capture the flow of inputs through a system (or group of processes) to their resulting output. The model shows the flow of information through a system, which can be an existing system or a proposed system.

All necessary elements for data flow diagramming are supported in the Data Modeler process model: primitive processes, composite processes with unlimited levels of decomposition, reusable transformation tasks, triggering events, information stores, external agents, record structure for describing external data elements, source-target

mapping of data elements, and CRUD (create, read, update, delete) dependencies between primitive process and data elements.

The following are important concepts for the process model:

- A **process** is an activity or a function that is performed for some specific reason. Ultimately each process should perform only one activity. A **primitive process** is a standalone process. A **composite process** consists of multiple **outer processes**.
- A **trigger** is something that happens which initiates the execution of a process.
- A **data flow** reflects the movement of single piece of data or logical collection of information. Flows describe the sequence of a data flow diagram. (For more information, see [Section 1.3.3.1, "Data Flow Diagrams"](#).)
- A **data store** is a collection of data that is permanently stored.
- An **external agent** is a person, organization, or system that is external to the system but interacts with it. External agents send information to and receive information from processes.
- An **information store** is a passive object that receives or stores information as entities and attributes in the data model. Ultimately, an information store corresponds with one or more entities of the data model.
- A **transformation task**, including input and output parameters, is an execution unit that communicates with surrounding environment that will execute it. An input parameter might be a date for which processing should be done. An output parameter might be a code that indicates whether the operation was successful or not. Transformation itself might involve reading, transforming, and saving information, some of which may not be directly tied to the input and output parameters. (For more information, see [Section 1.3.3.2, "Transformation Packages"](#).)
- A **role** is a set of defined privileges and permissions. Primitive processes connected to information stores (processes that create, read, update, and delete data elements) can be attached to a defined role, thus defining collaboration between roles and data elements. Later, role definitions can be transferred to any particular physical model such that appropriate database roles with defined Select, Insert, and Update permission will be created.

### 1.3.3.1 Data Flow Diagrams

A formal, structured analysis approach employs the data flow diagram (DFD) to assist in the functional decomposition process. A data flow diagram consists of the following components:

- External interactors, which are represented by rectangles
- Data stores, which are represented by open rectangles (two or three sides)
- Processes, which are represented by any rounded object (circle, oval, or square with rounded corners)

A process can represent a system function at one of various levels, from atomic through aggregate.

- Data flows, which are represented by arrows, and optionally with labels indicating their content.

### 1.3.3.2 Transformation Packages

A **transformation package** is a package as defined in the Object Management Group (OMG) *Common Warehouse Metamodel™ (CWM™) Specification, V1.1*. This specification introduces transformation packages as follows:

"A key aspect of data warehousing is to extract, transform, and load data from operational resources to a data warehouse or data mart for analysis. Extraction, transformation, and loading can all be characterized as transformations. In fact, whenever data needs to be converted from one form to another in data warehousing, whether for storage, retrieval, or presentation purposes, transformations are involved. Transformation, therefore, is central to data warehousing.

"The Transformation package contains classes and associations that represent common transformation metadata used in data warehousing. It covers basic transformations among all types of data sources and targets: object-oriented, relational, record, multidimensional, XML, OLAP, and data mining.

"The Transformation package is designed to enable interchange of common metadata about transformation tools and activities."

## 1.3.4 Logical Model

At the core of SQL Developer Data Modeler is the logical model (also called the entity-relationship diagram). It provides an implementation-independent view of enterprise information and acts as the mediator that maps definitions in the dimensional and process models to different physical implementations. A logical model, or a part of it (subject area, subview), can be transformed to one or more relational models.

You can build the logical model in any of the following ways:

- Manually in SQL Developer Data Modeler
- By importing models in a VAR file, such as those created by Sterling COOL:DBA V2.1 or Sterling Bsnteam V7.2, Cayenne Bsnteam V7.2, Rational Rose, TogetherJ, JDeveloper, MEGA, or PowerDesigner v.12
- By importing an existing model created by SQL Developer Data Modeler
- By reverse engineering from an imported relational model

The logical model combines two kinds of data:

- One logical diagram, plus an optional set of subviews and auxiliary displays, each associated with the appropriate diagram or subview
- Logical model object definitions

Subviews are considered as independent diagrams of the logical model, created to represent different subject areas.

The logical model enables you to create and manage object definitions for entities, logical views, attributes, unique identifiers, inheritances, relations, and arcs.

All logical model objects are displayed in the object browser tree.

### 1.3.4.1 Logical Diagram and Subviews

The logical model diagram contains graphical representations of entities, views, and links (relations and inheritances) between them.

When you are working with a complex logical model, you may want to create **Subviews**, each describing only a section of that model. You can define several logical

subviews for a single logical model, and you can assign entities and views to more than one subview. Links (relations) between two entities are displayed on the complete logical model and on logical subviews to which both referenced entities have been assigned.

There is no difference between performing changes in one of the subviews or in the complete logical model. Any changes made are immediately reflected in the complete logical model and any relevant subviews. However, you can remove entities and views from a subview without deleting them from the complete logical model.

To create a subview containing specific entities, you can select the desired entities in the logical model diagram, right-click, and select **Create Subview from Selected**.

### Diagramming Notation

Data Modeler supports the following alternatives for logical model diagramming notation:

- Bachman notation
- Barker notation

Detailed explanations and examples of each notation style are widely available in textbooks and on the Web. You can set the default notation type for new logical diagrams in the [Data Modeler Options \(User Preferences\)](#) (General Options, Diagram, Logical).

To switch from one notation type to the other (and to see the differences for a diagram), select the logical model diagram and click **View**, then **Logical Model Notation**, then the notation that is not the current one.

#### 1.3.4.2 Entities

An **entity** is an object or concept about which you want to store information. The structure of entity can be defined as collection of attributes or as based on structured type from the data types model. An entity may have candidate unique identifiers, one of which can be defined as primary unique identifier. Usually, an entity is mapped to table in the relational model.

#### 1.3.4.3 Attributes

A data **attribute** (property, data element, field) is a characteristic common to a particular entity. The data type of an attribute can be based on a logical data type, a domain, a distinct type, a collection type, or a structured type, or it can be a reference to structured type. If it a reference to a structured type, a scope entity can be defined. An attribute is mapped to a column in the relational model.

#### 1.3.4.4 Unique Identifiers (UIDs)

An entity **unique identifier** can be composed of one or more attributes. For each entity, you can define one **primary unique identifier** that uniquely identifies each entity occurrence. You can also specify one or more **foreign unique identifiers**, each of which points to (that is, must contain a value found in) a unique identifier in another entity.

#### 1.3.4.5 Inheritances

**Inheritance** defines a hierarchy of entities based on supertypes and subtypes. The supertype and subtype entities represent part of a system that has a recognizable subset of occurrences of an existing entity type. The subsets) are referred to as entity subtypes, with the original entity type being the supertype.

All attributes and relationships of the supertype must belong to all of its subtypes. However, some attributes and relationships of the subtype are added to those of the supertype. Subtypes are usefully defined where an identifiable group of entity occurrences has attributes in addition to those of the supertype.

#### 1.3.4.6 Relations

A **relation** (data relationship) is a natural association that exists between two or more entities. **Cardinality** defines the number of occurrences of one entity for a single occurrence of the related entity.

The relationship can be *identifying* or *not identifying*, and with a cardinality of 1:1 (one-to-one), 1:N (one-to-many), or N:M (many-to-many). A relationship with N:M cardinality is mapped to a reference table in the relational model. An **identifying** relationship indicates that the relationship is a component of the primary identifier for the target entity.

#### 1.3.4.7 Arcs

An **arc** is an exclusive relationship group, which is defined such that only one of the relationships can exist for any instance of an entity. For example, a seminar may be able to be taught by a staff member or an external consultant, but not by both. As examples, a seminar for new employees can be taught only by a corporate staff member, while a seminar in using Product XYX can be taught only by an external consultant with special qualifications.

All relations included in an arc should belong to the same entity and should have the same cardinality. Any foreign unique identifier (foreign UID) attributes belonging to relationships in an arc should be transferred as Allow Nulls during forward engineering. The meaning of mandatory relationships in an arc is that only one relationship must exist for a given instance of an entity.

To create an arc, do so after creating all the relationships to be included. Select the entity box, select all relationship lines to be included (hold Shift and click each line), and click the New Arc button in the toolbar or click **Object**, then **Logical**, then **New Arc**.

#### 1.3.4.8 Type Substitution

**Type substitution** is a subclassing mechanism that complements inheritance. Type substitution on the entity level take place only if the following are defined:

- Supertype/subtype inheritance between two structured types
- Entities based on the structured types which form a data type inheritance hierarchy (supertype/subtype inheritance)

#### 1.3.4.9 Views

A **view** is a named result set of a SQL query. A view selects the required data from one or more entities into a single virtual set. Views enable you to display different perspectives on the same database.

### 1.3.5 Relational Models

A relational model describes a database in terms of SQL tables, columns, and joins between tables. Each entity that you choose from the logical model is represented as a table in the relational model. Each row in a table represents a specific, individual occurrence of the corresponding entity. Each attribute of an entity is represented by a column in the table.

You can build a relational model in any of the following ways:

- Manually in SQL Developer Data Modeler
- By forward engineering from the logical model or a subview of the logical model
- By importing models in a VAR file, such as those created by Sterling COOL:DBA V2.1 or Sterling Bsnteam V7.2, Cayenne Bsnteam V7.2, Rational Rose, TogetherJ, JDeveloper, MEGA, or PowerDesigner v.12
- By importing an existing model created by SQL Developer Data Modeler
- By importing an Oracle Designer model
- By importing DDL files based on an existing database implementation
- By importing from the data dictionary of a supported database type and version

A relational model combines two kinds of data:

- One relational diagram, plus an optional set of subviews and auxiliary displays, each associated with the appropriate diagram or subview
- Relational model object definitions

Subviews are considered as independent diagrams of the relational model, created to represent different subject areas.

A relational model enables you to create and manage object definitions for tables, views, columns, indexes, and foreign keys. A relational model can contain one or more physical models.

All relational model objects are displayed in the object browser tree.

### 1.3.5.1 Relational Diagram and Subviews

The relational diagram contains graphical representations of tables, views, and links between them.

When you are working with a complex relational model, you may want to create subviews, each describing only a section of that model. You can define several relational subviews for a single relational model, and you can assign tables and views to more than one subview. Links (relations) between two tables are displayed on the complete relational model and on relational subviews to which both referenced tables have been assigned.

If you import from the data dictionary and select more than one schema to import, a relational model is created for all the schemas and a subview is created for each schema.

There is no difference between performing changes in one of the subviews or in the complete relational model. Any changes made are immediately reflected in the complete relational model and any relevant subviews. However, you can remove tables and views from a subview without deleting them from the complete relational model.

### 1.3.5.2 Tables

A **table** is an object in which you want to store information. The structure of table can be defined as a group of columns or as based on structured type from data types model. A table may have candidate keys, one of which can be defined as primary key. Usually, a table is mapped to entity from the logical model.



### 1.3.5.3 Columns

A table **column** is a characteristic common to a particular table. The data type of a column can be based on a logical data type, a domain, a distinct type, a collection type, or a structured type, or it can be a reference to structured type. If it is a reference to a structured type, a scope table can be defined. Usually, the columns in a table are mapped to the attributes of the corresponding entity from the logical model.

### 1.3.5.4 Indexes

An **index** is an object that consists of an ordered set of pointers to rows in a base table. Each index is based on the values of data in one or more table columns. Defining indexes on frequently searched columns can improve the performance of database applications.

### 1.3.5.5 Relations

A **relation** (data relationship) is a natural association that exists between two or more tables. Relationships are expressed in the data values of the primary and foreign keys. **Cardinality** defines the number of occurrences in one table for a single occurrence in the related table.

An **identifying** relationship indicates that the relationship is a component of the primary identifier for the target table.

An exclusive relationship (**arc**) specifies that only one of the relationships can exist for a given instance in the table. For example, a seminar may be able to be taught by a staff member or an external consultant, but not by both. As examples, a seminar for new employees can be taught only by a corporate staff member, while a seminar in using Product XYX can be taught only by an external consultant with special qualifications.

All relationships in an arc should belong to the same table, and should have the same cardinality. Any foreign key (FK) attributes belonging to relationships in an arc should be transferred as Allow Nulls during forward engineering. The meaning of mandatory relationships in an arc is that only one relationship must exist for a given instance in the table.

To create an arc, do so after creating all the relationships to be included. Select the table box, select all relationship lines to be included (hold Shift and click each line), and click the New Arc button in the toolbar or click **Object**, then **Relational**, then **New Arc**.

### 1.3.5.6 Relational Views

A relational **view** is a named result set of a SQL query. A view selects the required data from one or more tables into a single virtual set. Views enable you to display different perspectives on the same database.

## 1.3.6 Physical Models

A physical model describes a database in terms of Oracle Database objects (tables, views, triggers, and so on) that are based on a relational model. Each relational model can have one or more physical models. The following shows a database design hierarchy with several relational and physical models:

```
Database design
  Logical model
    Relational model 1
      Physical model 1a
```

```
Physical model 1b
. . . (other physical models)
Relational model 2
Physical model 2a
Physical model 2b
. . . (other physical models)
. . . (other relational models)
```

Each physical model is based on an RDBMS site object. An **RDBMS site** is a name associated with a type of database supported by Data Modeler. Several RDBMS sites are predefined (for example, for Oracle 11g and Microsoft SQL Server 2005). You can also use the [RDBMS Site Editor](#) dialog box to create user-defined RDBMS sites as aliases for supported types of databases; for example, you might create sites named *Test* and *Production*, so that you will be able to generate different physical models and then modify them.

Physical models do not have graphical representation in the work area; instead, they are displayed in the object browser hierarchy. To create and manage objects in the physical model, use the Physical menu or the context (right-click) menu in the object browser.

The rest of this topic briefly describes various Oracle Database objects, listed in alphabetical order (not the order in which they may appear in an Oracle physical model display).

### 1.3.6.1 Clusters

A **cluster** is a schema object that contains data from one or more tables.

- An index cluster must contain more than one cluster, and all of the tables in the cluster have one or more columns in common. Oracle Database stores together all the rows from all the tables that share the same cluster key.
- In a hash cluster, which can contain one or more tables, Oracle Database stores together rows that have the same hash key value.

### 1.3.6.2 Contexts

A **context** is a set of application-defined attributes that validates and secures an application.

### 1.3.6.3 Dimensions

A **dimension** defines a parent-child relationship between pairs of column sets, where all the columns of a column set must come from the same table. However, columns in one column set (called a level) can come from a different table than columns in another set. The optimizer uses these relationships with materialized views to perform query rewrite. The SQL Access Advisor uses these relationships to recommend creation of specific materialized views.

### 1.3.6.4 Directories

A **directory** is an alias for a directory (called a folder on Windows systems) on the server file system where external binary file LOBs (BFILEs) and external table data are located.

You can use directory names when referring to BFILEs in your PL/SQL code and OCI calls, rather than hard coding the operating system path name, for management flexibility. All directories are created in a single namespace and are not owned by an

individual schema. You can secure access to the BFILEs stored within the directory structure by granting object privileges on the directories to specific users.

### 1.3.6.5 Disk Groups

A **disk group** is a group of disks that Oracle Database manages as a logical unit, evenly spreading each file across the disks to balance I/O. Oracle Database also automatically distributes database files across all available disks in disk groups and rebalances storage automatically whenever the storage configuration changes.

### 1.3.6.6 External Tables

An **external table** lets you access data in an external source as if it were in a table in the database. To use external tables, you must have some knowledge of the file format and record format of the data files on your platform.

### 1.3.6.7 Indexes

An **index** is a database object that contains an entry for each value that appears in the indexed column(s) of the table or cluster and provides direct, fast access to rows. Indexes are automatically created on primary key columns; however, you must create indexes on other columns to gain the benefits of indexing.

### 1.3.6.8 Roles

A **role** is a set of privileges that can be granted to users or to other roles. You can use roles to administer database privileges. You can add privileges to a role and then grant the role to a user. The user can then enable the role and exercise the privileges granted by the role.

### 1.3.6.9 Rollback Segments

A **rollback segment** is an object that Oracle Database uses to store data necessary to reverse, or undo, changes made by transactions. Note, however, that Oracle strongly recommends that you run your database in automatic undo management mode instead of using rollback segments. Do not use rollback segments unless you must do so for compatibility with earlier versions of Oracle Database. See *Oracle Database Administrator's Guide* for information about automatic undo management.

### 1.3.6.10 Segments (Segment Templates)

A **segment** is a set of extents that contains all the data for a logical storage structure within a tablespace. For example, Oracle Database allocates one or more extents to form the data segment for a table. The database also allocates one or more extents to form the index segment for a table.

### 1.3.6.11 Sequences

A **sequence** is an object used to generate unique integers. You can use sequences to automatically generate primary key values.

### 1.3.6.12 Snapshots

A **snapshot** is a set of historical data for specific time periods that is used for performance comparisons by the Automatic Database Diagnostic Monitor (ADDM). By default, Oracle Database automatically generates snapshots of the performance data and retains the statistics in the workload repository. You can also manually create snapshots, but this is usually not necessary. The data in the snapshot interval is then

analyzed by ADDM. For information about ADDM, see *Oracle Database Performance Tuning Guide*.

### 1.3.6.13 Stored Procedures

A **stored procedure** is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or perform a set of related tasks.

### 1.3.6.14 Synonyms

A **synonym** provides an alternative name for a table, view, sequence, procedure, stored function, package, user-defined object type, or other synonym. Synonyms can be public (available to all database users) or private only to the database user that owns the synonym).

### 1.3.6.15 Structured Types

A **structured type** is a non-simple data type that associates a fixed set of properties with the values that can be used in a column of a table. These properties cause Oracle Database to treat values of one data type differently from values of another data type. Most data types are supplied by Oracle, although users can create data types.

### 1.3.6.16 Tables

A **table** is used to hold data. Each table typically has multiple columns that describe attributes of the database entity associated with the table, and each column has an associated data type. You can choose from many table creation options and table organizations (such as partitioned tables, index-organized tables, and external tables), to meet a variety of enterprise needs.

### 1.3.6.17 Tablespaces

A **tablespace** is an allocation of space in the database that can contain schema objects.

- A **permanent tablespace** contains persistent schema objects. Objects in permanent tablespaces are stored in data files.
- An **undo tablespace** is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode. Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
- A **temporary tablespace** contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in temp files.

### 1.3.6.18 Users

A database **user** is an account through which you can log in to the database. (A database user is a database object; it is distinct from any human user of the database or of an application that accesses the database.) Each database user has a database schema with the same name as the user.

### 1.3.6.19 Views

A **view** is a virtual table (analogous to a query in some database products) that selects data from one or more underlying tables. Oracle Database provides many view creation options and specialized types of views.

## 1.3.7 Business Information

Business information objects define business-oriented information about model objects, such as responsible parties and information about how to contact them, and identification of relevant offline documentation.

A model object can have zero or more business information objects associated with it, and a business information object can be associated with zero or more model objects. For example, a single document can be used to describe many different entities and attributes, or a single person can be the responsible party for multiple events.

There can also be many-to-many relationships among business objects. For example, a responsible party can have multiple sets of contact information (contact objects), and a contact object can be associated with multiple responsible parties. Similarly, one or more telephone, e-mail, location, and URL objects can be associated with multiple contact objects.

The Data Modeler business information model is based on the Object Management Group (OMG) business information package, which is described in the OMG *Common Warehouse Metamodel™ (CWM™) Specification, V1.1* as follows: "The Business Information Metamodel provides general purpose services available to all CWM packages for defining business-oriented information about model elements. The business-oriented services described here are designed to support the needs of data warehousing and business intelligence systems; they are not intended as a complete representation of general purpose business intelligence metamodel. Business Information Metamodel services support the notions of responsible parties and information about how to contact them, identification of off-line documentation and support for general-purpose descriptive information."

The rest of this topic briefly describes business information objects, listed in alphabetical order (not the order in which they appear in the object browser under Business Information).

### 1.3.7.1 Contacts

A contact object groups the various types of related contact information. Each contact object can be associated with multiple email, location, URL, and telephone objects. Conversely, each email, location, URL, and telephone object can be associated with many contact objects. (See also [Section 3.12, "Contact Properties"](#).)

### 1.3.7.2 Documents

A document object represents externally stored descriptive information about some aspects of the modeled system. A document object can be associated with one or more model objects. (See also [Section 3.23, "Document Properties"](#).)

### 1.3.7.3 Emails

An email object identifies a single electronic mail address. Through the use of a contact object, you can associate an email address with one or more responsible parties. The sequence of email objects for a contact might be used to represent the order in which to try email addresses in attempting to communicate with a contact. (See also [Section 3.26, "Email Properties"](#).)

### 1.3.7.4 Locations

A location object identifies a single physical location. Through the use of a contact object, you can associate a location with one or more responsible parties. The sequence of contact objects for a location might be used to represent the order in which to try

contacting a person or group associated with a location. (See also [Section 3.54](#), "Location Properties".)

### 1.3.7.5 Resource Locators

A resource locator object provides a general means for describing a resource whose location is not defined by a traditional mailing address. For example, a resource locator could refer to anything from a Web address (such as "www.example.com") to a location within a building (such as "Room 317, third file cabinet, 2nd drawer"). (See also [Section 3.74](#), "Resource Locator Properties".)

### 1.3.7.6 Responsible Parties

A responsible party object represents a person, role, or organization that has a responsibility for, or should receive information about, one or more model objects. The precise meaning of the "responsibility" of a responsible object depends on the specific system being implemented. (See also [Section 3.75](#), "Responsible Party Properties".)

### 1.3.7.7 Telephones

A telephone object represents telephone contact information. A telephone object can be associated with one or more contacts. (See also [Section 3.89](#), "Telephone Properties".)

## 1.4 Approaches to Data Modeling

When modeling data, you can choose an approach best suited to the nature of the work to be done. The approaches to data modeling include the following: designing a new database, developing a design for an existing database, or performing maintenance on an existing database design

- [Top-Down Modeling](#): for designing a new database
- [Bottom-Up Modeling](#): for creating a database based on extracting metadata from an existing database or using the DDL code obtained from an implementation of an existing database
- [Targeted Modeling](#): for adapting a database to new requirements

### 1.4.1 Top-Down Modeling

Top-down modeling gathers information about business requirements and the internal environment, and proceeds to define processes, a logical model of the data, one or more relational models, and one or more physical models for each relational model. The steps and information requirements can range from simple to elaborate, depending on your needs. Top-down modeling can involve the following steps, but you can abbreviate or skip steps as appropriate for your needs.

1. **Develop the business information.**
  - a. **Create documents.** In the object browser, right-click **Logical** and select **Properties**, then click **Documents** and add items as appropriate.
  - b. **Create responsible parties** with contacts, e-mail addresses, locations, telephone numbers, and locations. In the object browser, right-click **Logical** and select **Properties**, then click **Responsible Parties** and add items as appropriate.
  - c. **Define any other information.** In the object browser, right-click **Logical** and select **Properties**, then modify other properties (Naming Options, Comments, Notes) as needed.

2. **Develop the process model**, using a data flow diagram. In the object browser under Process Model, right-click **Data Flow Diagrams** and select **New Data Flow Diagram**.
  - a. **Create processes**. For each process, click the New Process icon, click in the data flow diagram window, and enter information in the [Process Properties](#) dialog box.
  - b. **Create external agents**. For each external agent, click the New External Agent icon, click in the data flow diagram window, and enter information in the [External Agent Properties](#) dialog box.
  - c. **Create information stores**. For each process, click the New Information Store icon, click in the data flow diagram window, and enter information in the [Information Store Properties](#) dialog box.
  - d. **Create flows** with information structures. For each flow, click the New Flow icon, click the starting object (such as a process) in the data flow diagram window, and click the ending object for the flow; then double-click the flow arrow and modify information (as needed) in the [Flow Properties](#) dialog box.
3. **Develop the logical model**.
  - a. **Create entities**, and for each entity its **attributes** and **unique identifiers**. You can create all entities first and then the attributes and unique identifiers for each, or you can create the first entity with its attributes and unique identifiers, then the second, and so on.  
  
To create an entity, click the Logical tab, click the New Entity icon, click in the logical model window, and enter information in the [Entity Properties](#) dialog box. You can also enter attributes and unique identifiers using the appropriate panes in this dialog box.
  - b. **Create relations between entities**. For each relation, click the desired icon: New M:N Relation (many-to-many), New 1:N Relation (one-to-many) , New 1:N Identifying Relation (one-to-many, identifying), or New 1:1 Relation (one-to-one). Click the entity for the start of the relation, and click the entity for the end of the relation; then double-click the relation line and modify information (as needed) in the [Relation Properties](#) dialog box.
  - c. **Apply design rules** to the logical model. Click **Tools**, then **Design Rules**, and use the [Design Rules](#) dialog box to check for and fix any violations of the design rules.
  - d. **Forward engineer the logical model to a relational model**. Click **Design**, then **Engineer to Relational Model**, and use the [Engineering](#) dialog box to generate a relational model reflecting all or a specified subset of objects from the logical model.
4. **Develop the multidimensional model**, if needed.
  - a. Create cubes.
  - b. Create levels.
  - c. Create dimensions.
  - d. Create links.
  - e. Apply design rules for the multidimensional model.
  - f. Export the multidimensional model, as needed.
5. **Develop one or more relational models**, doing the following for each as needed.

- a. Split tables. To split one table into two, select the table on the relational model diagram, and either click the Split Table button or click **Object**, then **Relational**, then **Split Table**.
  - b. Merge tables. To merge a table into another table (removing the merged table), either click the Merge Table button or click **Object**, then **Relational**, then **Merge Tables**. Then, in the relational model diagram, first the table into which to merge columns from the other table, and next select the other table whose columns are to be merged. (After the merge, the second table will be removed.)
  - c. Check design rules for the relational model. Click **Tools**, then **Design Rules**.
6. **Develop one or more physical models for each relational model**, doing the following for each.
- a. Open a physical model.
  - b. Check design rules for the physical model. Click **Tools**, then **Design Rules**.
  - c. Generate DDL code, which can be used to generate the actual database objects. Click **View**, then **DDL File Editor**, and then use the [DDL File Editor](#) dialog box to select a physical model, generate DDL code, and save the code to a script file.

## 1.4.2 Bottom-Up Modeling

Bottom-up modeling builds a database design based on either metadata extracted from an existing database or a file with DDL code that implements an existing database. The resulting database is represented as a relational model and a physical model, and you reverse engineer the logical model from the relational model. Bottom-up modeling can involve the following steps, but you can abbreviate or skip some steps as appropriate for your needs.

1. Generate the relational model in either of the following ways:
  - **Extract metadata** directly from an existing database: click **File**, then **Import**, then **Data Dictionary**; then follow the instructions for the wizard (see [Data Dictionary Import \(Metadata Extraction\)](#)).
  - **Import DDL** code that reflects an existing database implementation. Click **File**, then **Import**, then **DDL File**.
2. **As needed, modify the relational model and create additional relational models.**
3. **As needed, denormalize the relational model or models.** Perform the following steps iteratively, as needed, on each model.
  - a. **Split or merge tables, or do both.**

To split one table into two, select the table on the relational model diagram, and either click the Split Table button or click **Object**, then **Relational**, then **Split Table**.

To merge a table into another table (removing the merged table), either click the Merge Table button or click **Object**, then **Relational**, then **Merge Tables**. Then, in the relational model diagram, first the table into which to merge columns from the other table, and next select the other table whose columns are to be merged. (After the merge, the second table will be removed.)
  - b. **Check the design rules for the model.** To view the design rules, click **Tools**, then **Design Rules**; select the desired relational model; and use the [Design Rules](#) dialog box.



4. **Reverse engineer the logical model from a relational model.** Click the Engineer to Logical Model icon, or click **Design**, then **Engineer to Logical Model**.
5. **As needed, modify the logical model.**
6. **Check design rules for the logical model.** Click **Tools**, then **Design Rules**.
7. **Save the design.**
8. **Generate DDL code**, and use it to create the database implementation. Click **View**, then **DDL File Editor**. In the **DDL File Editor** dialog box, select the physical model and click **Generate**. Specify any desired **DDL Generation Options**, then click **OK**.

### 1.4.3 Targeted Modeling

Targeted modeling involves maintaining an existing database by adapting it to new requirements.

---



---

**Note:** Maintaining a database with SQL Developer Data Modeler requires that the design and the actual database implementations be fully synchronized. If you are not sure if this is the case, you should consider the designs outdated and perform the procedures in [Section 1.4.2, "Bottom-Up Modeling"](#).

---



---

Depending on the kind of changes necessary, you can start with the logical model, one or more relational models, or one or more physical models, and then forward engineer or reverse engineer as appropriate.

#### To start with changes to the logical model:

1. For each logical model object (entity, attribute, relation, and so on) that you want to modify, modify its properties. For example, to add an attribute to an entity:
  - a. Double-click the entity's icon in the Logical diagram (or right-click the entity name in the object browser and select Properties).
  - b. In the **Entity Properties** dialog box, click **Attributes**.
  - c. Click the Add (+) icon and specify the attribute properties.
2. When you are finished modifying the logical model, forward engineer the changes to the relational model or models by clicking the Logical pane and clicking **Design**, then **Engineer to Relational Model**.
3. In the **Engineering** dialog box, specify any desired filtering, then click **Engineer**.

#### To start with changes to a relational model:

1. For each relational model object (table, column, and so on) that you want to modify, modify its properties. For example, to add a column to a table in a relational model:
  - a. Double-click the table's icon in the diagram for the relational model (or right-click the table name in the object browser and select Properties).
  - b. In the **Table Properties** dialog box, click **Columns**.
  - c. Click the Add (+) icon and specify the column properties.

2. When you are finished modifying the relational model, reverse engineer the changes to the logical model by clicking the pane for the relational model and clicking **Design**, then **Engineer to Logical Model**.
3. In the [Engineering](#) dialog box, specify any desired filtering, then click **Engineer**.

## 1.5 Data Modeler Options (User Preferences)

You can customize many aspects of SQL Developer Data Modeler by clicking **Tools**, then **General Options**.

Many preferences are self-explanatory, and this topic explains only those whose meaning and implications are not obvious. The preferences are grouped in the following categories:

- [General](#)
- [Model](#)
- [Diagram](#)
- [Naming Standard](#)
- [DDL](#)
- [Third Party JDBC Drivers](#)

### 1.5.1 General

The General pane contains options that affect the startup and overall behavior and appearance of SQL Developer Data Modeler.

**Default Designs Directory:** The default directory or folder from which to open a design or in which to create a design.

**Default Import Directory:** The default directory or folder from which to import designs.

**Show Log After Import:** Controls whether a Log window is displayed after an import operation. The window contains informational messages and any warning or error messages.

**Show "Select Relational Models" Dialog:** Controls whether the dialog box for selecting relational models to be included is displayed when you open a Data Modeler design. If this option is disabled, all relational models are included by default when you open a Data Modeler design.

**Show Properties Dialog on New Object:** Controls whether the Properties dialog box for objects of that type is displayed when you create a new model object.

### 1.5.2 Model

The Model pane contains options that apply to several types of models.

**Default RDBMS Type:** Default database type.

**Default RDBMS Site:** Default site within the default database type.

**Columns and Attributes Defaults: Nulls Allowed:** Controls whether new columns and attributes are allowed to have null values. If this option is disabled, new columns and attributes are by default mandatory (value required).

**Model: Logical**

Contains options that apply to the logical model.

**Relation Cardinality: Source Optional:** Controls whether the source entity in a relationship must, by default, contain one or more instances. If this option is enabled, source instances are not required for all relationship types; if this option is disabled, one or more source instances are required for all relationship types.

**Relation Cardinality: Target Optional:** Controls whether the target entity in a relationship must, by default, contain one or more instances. If this option is enabled, target instances are not required for all relationship types; if this option is disabled, one or more target instances are required for all relationship types.

**Use and Set First Unique Identifier as Primary Key:** Controls whether, by default, the first unique identifier attribute is set as the primary unique identifier when you create an entity.

**Foreign UID Attribute Name Synchronization: Keep as the Name of the Originating Attribute:** Controls whether the supertype or referenced attribute must be used in foreign unique identifier naming. To be able to specify some other name, deselect this option.

**Model: Relational**

Contains options that apply to a relational model.

**Delete FK Columns Strategy:** Specifies what Data Modeler should do when you attempt to delete a table that has one or more generated foreign key columns (columns in other tables) pointing to it: delete the foreign key columns, do not delete the foreign key columns, or ask to confirm the foreign key column deletions.

For example, using the relational model in [Chapter 2, "Tutorial: Data Modeling for a Small Database"](#), if you delete the Books table, the Transactions table contains the book\_id foreign key column that refers to the primary key of the Books table. Your choice for this option determines what happens to the Transactions.book\_id column if you delete the Books table.

**Default Foreign Key Delete Rule:** Specifies what happens if a user tries to delete a row containing data that is involved in a foreign key relationship:

- **No Action** causes an error message to be displayed indicating that deletion is not allowed; the deletion is rolled back.
- **Cascade** deletes all rows containing data that is involved in the foreign key relationship.
- **Set Null** sets the value to null if all foreign key columns for the table can accept null values.

**Model: Physical**

Contains options that apply to a physical model. Different options apply to each supported type of database.

### 1.5.3 Diagram

The Diagram pane contains general options that affect the appearance of model diagrams.

**General: Show Grid:** Controls whether a grid is displayed in the background on diagrams. Seeing the grid can help you to align objects vertically and horizontally on the diagram.

**General: Synchronize with Tree:** Controls whether the focus on an active diagram is automatically moved to reflect the selection of objects under that model in the object browser.

**General: Diagram Color:** Displays a dialog box for selecting the color scheme for the background on diagrams.

**General: Line Auto Route:** Controls whether lines representing relations, foreign key relations, inheritances, flows, and other relationships are automatically drawn on diagrams. If you deselect this option, you determine how these lines are drawn; for example, you may want to add or move break points manually, in order to enhance the clarity of your models. For more information about Auto Route and drawing lines, see the explanation of the Auto Route command in [Section 1.2.2, "Context Menus"](#).

**Process Model: Show Flow Name:** Controls whether the flow name is displayed in a box on each flow line in data flow diagrams.

### **Diagram: Logical Model**

Contains options that apply to the diagram of the logical model.

**Notation Type:** Notation type: Barker (sometimes called "crow's foot") or Bachman.

**Show Source/Target Name:** Controls whether the Name on Source and Name on Target values (in the Cardinality pane of the [Record Structure Properties](#) dialog box) are displayed. If they are displayed, you can format the text and move the boxes.

**Box-in-Box Presentation for Entity Inheritances:** Displays subtypes in a box inside their supertype's box.

**Domains Presentation:** Specifies what is displayed as the data type for an attribute based on a domain: Domain Name causes the domain name to be displayed; Used Logical Type causes the logical type used in the domain definition to be displayed.

### **Diagram: Relational Model**

Contains options that apply to a diagram of a relational model.

**Foreign Key Arrow Direction:** Controls whether arrowhead points toward the primary key or the foreign key in foreign key relationship arrows.

**Show Foreign Key Name:** Controls whether a text box containing the foreign key name is displayed on the foreign key relationship arrow.

### **Diagram: Classification Types**

Specifies colors for the display of different classification types in a multidimensional model.

### **Diagram: Format**

Specifies options for default fonts and colors for objects and default widths and colors for types of lines.

## **1.5.4 Naming Standard**

The Naming Standard pane lets you implement naming standardization: you can view, add, and modify naming standards for logical and relational model objects and for domains. These standards will be checked when you apply [Design Rules](#), and any violations of the standards will be reported as errors or warnings.

Do not confuse naming standardization with using the [Name Abbreviations](#) dialog box, which makes immediate name changes to enforce consistency in spellings and abbreviations, and which is limited to relational model name strings.

### Logical, Relational, Domains

For logical model entities and attributes, relational model tables and columns, and domains, you can add, rearrange, and make optional or mandatory any of the following components of object names: prime word, class word, modifier, and qualifier. The acceptable values of these components are specified in the glossary file or files that you specify in the [Glossary](#) pane.

**Title Case (Separator option):** Refers to capitalizing each "word" and not including spaces: for example, GovernmentAccounts. (Title case is sometimes called CamelCase.)

**Abbreviated Only:** If this option is enabled, non-abbreviated words cannot be used in relational model object names (that is, only abbreviated words can be used).

For explanations of other terms, see [Section 3.39, "Glossary Editor"](#). For an excellent discussion of naming standards, see the *United States Coast Guard Data Naming Element Standards Guidebook*.

### Relational - Others

For various kinds of constraints in relational models, you can edit the format string and add variable string elements.

**Example:** To see a sample name in a currently specified format, select the desired constraint type (for example, Foreign Key).

### Glossary

You can add one or more glossary files to be used in naming standardization. (For more information about glossaries, see [Section 3.39, "Glossary Editor"](#).)

## 1.5.5 DDL

The DDL pane contains general options for Data Definition Language (DDL) statements in code to be generated.

**Statement Termination Character for DB2 and UDB:** Termination character for DDL for IBM DB2 and UDB databases.

**Create Type Substitution Triggers for Oracle and UDB:** Controls whether triggers are created for type substitutions in Oracle and IBM UDB physical models.

**Create Triggers for FK Arc Constraint:** Controls whether triggers are created in generated DDL code to implement foreign key arc constraints.

**Create Triggers for Non Transferable FK:** Controls whether triggers are created for non-transferable foreign key relationships. (Whether a foreign key relationship is transferable is controlled by the Transferable (Updateable) option in the [Foreign Key Properties](#) dialog box.)

**Show CHAR/BYTE Unit for Oracle Varchar2 and Char Types:** Controls whether, for attributes of Oracle type CHAR or VARCHAR2, the unit (CHAR or BYTE) associated with the attribute length is included for columns based on the attribute in relational model diagrams and in generated CREATE TABLE statements.

**Use 'Data Type Kind' Property in Compare Functionality:** Controls whether the data type kind (such as domain, logical type, or distinct type) should be considered to

prevent types of different kinds from generating the same native data type (for example, preventing a domain and a logical type from resulting in Number(7,2)).

**DDL: DDL/Migration**

Lets you specify one or more pairs of string replacements to be made when DDL statements are generated. Each pair specifies the old string and the new string with which to replace the old string.

**Selected:** Controls whether the specified replacement is enabled or disabled.

**Case Sensitive:** Controls whether the replacement is done only if the case of the old string in the DDL exactly matches the case specifies for the old string.

## 1.5.6 Third Party JDBC Drivers

The Third Party JDBC Drivers pane specifies drivers to be used for connections to third-party (non-Oracle) databases. SQL Developer Data Modeler needs to use a JDBC driver for some operations, such as obtaining metadata from the third-party database.

Oracle does not supply non-Oracle drivers. To access any non-Oracle databases that require the use of drivers other than ODBC/JDBC (which are included in Java), you must download the files for the necessary drivers, and then add them using this pane. To download drivers, use the appropriate link at the third-party site. For example:

- For Microsoft SQL Server 2000:  
<http://www.microsoft.com/downloads/details.aspx?familyid=07287B11-0502-461A-B138-2AA54BFDC03A&displaylang=en> (Be sure to install all three .jar files: msbase.jar, mssqlserver.jar, and msutil.jar.)
- For Microsoft SQL Server 2005:  
<http://www.microsoft.com/downloads/details.aspx?familyid=C47053EB-3B64-4794-950D-81E1EC91C1BA&displaylang=en>
- For IBM DB2/LUW, the IBM Data Server Driver for JDBC and SQLJ at:  
<http://www-01.ibm.com/software/data/db2/express/additional-downloads.html>

For each driver to be added, click the Add (+) icon and select the path for the driver. Third-party databases and their required driver files include:

- Microsoft SQL Server 2000: msbase.jar, mssqlserver.jar, and msutil.jar
- Microsoft SQL Server 2005: sqljdbc.jar
- IBM DB2/UDB: db2jcc.jar

## 1.6 Saving, Opening, Exporting, and Importing Designs

To store a design (or parts of a design) that you are working on, you can save or export it.

- **Saving** a design enables you to save all elements of the design: the logical model, relational models, physical models, process model, and business information. An XML file and a directory structure (described in [Section 1.3.1, "Database Design"](#)) are created for a new design or updated for the existing design, which is stored in SQL Developer Data Modeler format.

To save a design, click **File**, then **Save**. If the design was not previously saved, specify the location and XML file name. To save a design in a different file and directory structure. click **File**, then **Save As**.

- **Exporting** a design enables you to save parts of the design (logical model, relational models but no physical models, and data types model) to a file. You can export in a variety of formats, both non-Oracle and Oracle. Thus, exporting provides flexibility in output formats, but saving enables you to save more design objects if you only need SQL Developer Data Modeler output.

To export a design, click **File**, then **Export**, then the output format.

To use a design that had been saved, you can **open** it by clicking **File**, then **Open**. Opening a design makes all models and objects in the saved design available for you to work on. Any saved physical models are not initially visible in the object browser; however, you can make a physical model visible by right-clicking Physical Models under the desired relational model, selecting Open, and then specifying the database type (such as Oracle 11g).

To use a design that had been saved by SQL Developer Data Modeler, or exported or saved by another data modeling tool, you can **import** it by clicking **File**, then **Import**, then the type of design to be imported. Usually, you specify a file, and then use a wizard that enables you to control what is imported.

Any text file that you open or import must be encoded in a format supported by the operating system locale setting. For information about character encoding and locales, see *Oracle Database Globalization Support Guide*.

The rest of this section contains information about importing from specific types of files and other sources.

### 1.6.1 Importing a DDL File

Importing a DDL files enables you to create a relational model based on an existing database implementation. DDL files can originate from any supported database type and version. The file to be imported usually has the extension `.ddl` or `.sql`.

The import process creates a new relational model with the name of the imported DDL file and opens a physical model reflecting the source site.

### 1.6.2 Importing Cube Views Metadata

Importing Cube Views metadata enables you to create a multidimensional model based on an existing implementation, as reflected in a specified XML file.

### 1.6.3 Importing from Microsoft XMLA

Importing from Microsoft XMLA enables you to create a multidimensional model stored in the Microsoft XMLA file format.

### 1.6.4 Importing an ERwin File

Importing an ERwin file enables you to capture models from the ERwin modeling tool. Specify the XML file containing definitions of the models to be imported.

### 1.6.5 Importing from a Data Dictionary

Importing from a data dictionary enables you to create a relational model and a physical model based on an existing database implementation. The data dictionary can be from any supported database type and version.

In the wizard for importing from a data dictionary, you must either select an existing database connection or create (add) a new one, and then follow the instructions to select the schema or database and the objects to be imported.

After you import from a data dictionary, you can edit the relational and physical models as needed, and you can reverse engineer the logical model from the relational model.

### 1.6.6 Importing an Oracle Designer Model

Importing an Oracle Designer model enables you to create a relational model and a physical model based on an existing Oracle Designer model.

In the wizard for importing an Oracle Designer model, you must either select an existing database connection or create (add) a new one, and then follow the instructions to select the work areas, application systems, and objects to be imported. (Note that you cannot import Oracle Designer dataflow diagrams.)

After you import the Oracle Designer model, you can edit the relational and physical models as needed, and you can reverse engineer the logical model from the relational model.

### 1.6.7 Importing a SQL Developer Data Modeler Design

Importing a SQL Developer Data Modeler design enables you to capture the logical model and any relational and data type models from a design previously exported from SQL Developer Data Modeler.

### 1.6.8 Importing a Domain

Importing a domain enables you to change and extend the existing domain definitions. In the [Import Domains](#) dialog box, select the domains to be imported and deselect (clear) the domains not to be imported.

## 1.7 Data Modeler Reports

You can export the information about a Data Modeler design to a database schema (called the *reporting repository*), and you can then use the reports feature in SQL Developer to view reports about that Data Modeler design.

The **reporting repository** is a collection of schema objects for storing metadata and data about Data Modeler designs. It is recommended that you create a separate database user for the Data Modeler reporting repository, and use that schema only for the reporting repository. For example, create a user named DM\_REPORT\_REPOS, and grant that user at least CONNECT and RESOURCE privileges. (You can create the reporting repository in an existing schema that is also used for other purposes, but you might find that more confusing to keep track of.)

To export the information about a design to the reporting repository, click **File**, then **Export**, then **To Reporting Schema**.

To view the reports in SQL Developer, you must do the following:

1. In SQL Developer, check to see if the Reports navigator already includes a child node named **Data Modeler Reports**. If it does include that node, go to the next step; if it does not include that node, install the Data Modeler Reports extension, as follows:



Click **Help**, then **Check for Updates**. In the Check for Updates wizard, specify **Install From Local File**, and specify the following local file in the location where you installed Data Modeler:

datamodeler\reports\oracle.sqldeveloper.datamodeler\_reports.*nn.nn*.zip  
(Windows systems) or datamodeler/reports/oracle.sqldeveloper.datamodeler\_reports.*nn.nn*.zip (Linux systems), where *nn.nn* reflects a build number.

2. In SQL Developer, open the Reports navigator, expand the **Data Modeler Reports** node, plus nodes under it as desired.

For each report that you want to view:

1. Double-click the node for the report name.
2. Select the database connection that you used for the reports repository.
3. Complete the Bind Variables dialog information, and click OK. For the bind variables, the default values represent the most typical case: display all available information for the most recent version of the design.

The bind variables enable you to restrict the output. The default value for most bind variables is null, which implies no further restrictions. To specify a bind variable, select the variable name and type an entry in the Value field. Any bind variable values that you enter are case insensitive. Bind variable values can contain the special characters % (percent sign) to mean any string and \_ (underscore) to mean any character.

### 1.7.1 Design Content reports

Design Content reports list information about the design content (objects in the design).

**Data Types Model:** Contains reports related to the [Data Types Model](#).

**Logical Model:** Contains reports related to the [Logical Model](#).

**Relational Model:** Contains reports related to the [Relational Models](#).

### 1.7.2 Design Rules reports

Design Rules reports list information about the design rules as they apply to the logical and relational models. (See the information about the [Design Rules](#) dialog box.)

**Logical Model:** Contains reports related to the [Logical Model](#).

**Relational Model:** Contains reports related to the [Relational Models](#).

## 1.8 Read-Only Data Modeler Viewer

SQL Developer Data Modeler Viewer is a free read-only version of SQL Developer Data Modeler that enables you to open, import, and view a database design, and to save it separate from the original design. However, you cannot create, modify, or delete any objects.

The online help is the same for both SQL Developer Data Modeler and the free viewer; thus, the help includes information about features that are not supported in the viewer.

## 1.9 Additional Resources for Data Modeling

See the following for more information, including advanced materials, about data modeling:

- SQL Developer home page (OTN), which includes links for white papers, viewlets (screen demonstrations), Oracle by Example (OBE) tutorials, and other materials: [http://www.oracle.com/technology/products/database/sql\\_developer/](http://www.oracle.com/technology/products/database/sql_developer/)
- Object Management Group (OMG) site (<http://www.omg.org/>), especially the MetaObject Facility (MOF) and Common Warehouse Metamodel (CMW) specifications
- *United States Coast Guard Data Naming Element Standards Guidebook* ([http://www.uscg.mil/directives/ci/5000-5999/CI\\_5230\\_42A.pdf](http://www.uscg.mil/directives/ci/5000-5999/CI_5230_42A.pdf)), especially concepts and recommendations relating to naming standards

---

---

## Tutorial: Data Modeling for a Small Database

In this tutorial, you will use SQL Developer Data Modeler to create models for a simplified library database, which will include entities for books, patrons (people who have library cards), and transactions (checking a book out, returning a book, and so on).

This tutorial uses the same entities as for the tutorial provided with the SQL Developer online help. The model is deliberately oversimplified and would not be adequate for any actual public or organizational library. For more advanced tutorials and other materials, see [Section 1.9, "Additional Resources for Data Modeling"](#).

If the instructions do not mention a particular dialog box, tab, or field, then do not specify anything for it.

This simplified tutorial uses only a subset of the possible steps for the [Top-Down Modeling](#) approach. (For information about the approaches, see [Section 1.4, "Approaches to Data Modeling"](#).)

You will perform the following major steps:

1. [Develop the Logical Model](#).
2. [Develop the Relational Model](#).
3. [Generate DDL](#).
4. [Save the Design](#).

### 2.1 Develop the Logical Model

The logical model for the database includes three entities: Books (describes each book in the library), Patrons (describes each person who has a library card), and Transactions (describes each transaction involving a patron and a book). However, before you create the entities, create some domains that will make the entity creation (and later DDL generation) more meaningful and specific.

To start developing the logical model, go to [Section 2.1.1, "Adding Domains"](#).

#### 2.1.1 Adding Domains

In planning for your data needs, you have determined that several kinds of fields will occur in multiple kinds of records, and many fields can share a definition. For example, you have decided that:

- The first and last names of persons can be up to 25 characters each.
- Street address lines can be up to 40 characters.

- City names can be up to 25 characters.
- State codes (United States) are 2-character standard abbreviations.
- Zip codes (United States postal codes) can be up to 10 characters (*nnnnn-nnnn*).
- Book identifiers can be up to 20 characters.
- Other identifiers are numeric, with up to 7 digits (no decimal places).
- Titles (books, articles, and so on) can be up to 50 characters.

You therefore decide to add appropriate domains, so that you can later use them to specify data types for attributes when you create the entities. (These added domains will also be available after you exit Data Modeler and restart it later.)

1. Click **Tools**, then **Domains Administration**.
2. In the **Domains Administration** dialog box, add domains with the following definitions. Click **Add** to start each definition, and click **Apply** after each definition.

Name	Logical Type	Other Information
Person Name	VARCHAR	Size: 25
Address Line	VARCHAR	Size: 40
City	VARCHAR	Size: 25
State	VARCHAR	Size: 2
Zip	VARCHAR	Size: 10
Book Id	VARCHAR	Size: 20
Numeric Id	NUMERIC	Precision: 7, Scale: 0
Title	VARCHAR	Size: 50

3. When you have finished defining these domains, click **Save**. This creates a file named `defaultdomains.xml` in the `domains` directory (folder) under the location where you installed Data Modeler.
4. Optionally, copy the `defaultdomains.xml` file to a new location (not under the Data Modeler installation directory), and give it an appropriate name, such as `library_domains.xml`. You can then import domains from that file when you create other designs.
5. Click **Close** to close the dialog box.
6. Go to [Section 2.1.2, "Creating the Books Entity"](#).

## 2.1.2 Creating the Books Entity

The Books entity describes each book in the library. Create the Books entity as follows:

1. In the main area (right side) of the SQL Developer Data Modeler window, click the Logical tab.
2. Click the New Entity icon.
3. Click in the logical model pane in the main area; and in the Logical pane press, diagonally drag, and release the mouse button to draw an entity box. The [Entity Properties](#) dialog box is displayed.
4. Click **General** on the left, and specify as follows:

**Name:** Books

- Click **Attributes** on the left, and use the Add (+) icon to add the following attributes, one at a time. (For datatypes, select from the Domain types except for Rating, which is a Logical type.)

Name	Datatype	Other Information and Notes
book_id	Domain: Book Id	Primary UID (unique identifier). (The Dewey code or other book identifier.)
title	Domain: Title	M (mandatory, that is, must not be null).
author_last_name	Domain: Person Name	M (mandatory, that is, must not be null).
author_first_name	Domain: Person Name	25 characters maximum.
rating	Logical type: NUMERIC (Precision=2, Scale= 0)	(Librarian's personal rating of the book, from 1 (poor) to 10 (great).)

- Click **OK** to finish creating the Books entity.
- Go to [Section 2.1.3, "Creating the Patrons Entity"](#).

### 2.1.3 Creating the Patrons Entity

The Patrons entity describes each library patron (that is, each person who has a library card and is thus able to borrow books). Create the Patrons entity as follows:

- In the main area (right side) of the SQL Developer Data Modeler window, click the Logical tab.
- Click the New Entity icon.
- Click in the logical model pane in the main area; and in the Logical pane press, diagonally drag, and release the mouse button to draw an entity box. (Suggestion: draw the box to the right of the Books box.) The [Entity Properties](#) dialog box is displayed.
- Click **General** on the left, and specify as follows:
 

**Name:** Patrons
- Click **Attributes** on the left, and use the Add (+) icon to add the following attributes, one at a time. (For datatypes, select from the Domain types, except for location, which uses the structured type SDO\_GEOMETRY.)

Attribute Name	Type	Other Information and Notes
patron_id	Domain: Numeric Id	Primary UID (unique identifier). (Unique patron ID number, also called the library card number.)
last_name	Domain: Person Name	M (mandatory, that is, must not be null). 25 characters maximum.
first_name	Domain: Person Name	(Patron's first name.)

Attribute Name	Type	Other Information and Notes
street_address	Domain: Address Line	(Patron's street address.)
city	Domain: City	(City or town where the patron lives.)
state	Domain: State	(2-letter code for the state where the patron lives.)
zip	Domain: Zip	(Postal code where the patron lives.)
location	Structured type: SDO_ GEOMETRY	Oracle Spatial geometry object representing the patron's geocoded address.

- Click **OK** to finish creating the Patrons entity.
- Go to [Section 2.1.4, "Creating the Transactions Entity"](#).

## 2.1.4 Creating the Transactions Entity

The Transactions entity describes each transaction that involves a patron and a book, such as someone checking out or returning a book. Each record a single transaction, regardless of how many books the patron brings to the library desk. For example, a patron returning two books and checking out three books causes five transactions to be recorded (two returns and three checkouts). Create the Transactions entity as follows:

- In the main area (right side) of the SQL Developer Data Modeler window, click the Logical tab.
- Click the New Entity icon.
- Click in the logical model pane in the main area; and in the Logical pane press, diagonally drag, and release the mouse button to draw an entity box. (Suggestion: Draw the box below and centered between the Books and Patrons boxes.) The [Entity Properties](#) dialog box is displayed.
- Click **General** on the left, and specify as follows:  
**Name:** Transactions
- Click **Attributes** on the left, and use the Add (+) icon to add the following attributes, one at a time. (For datatypes, select from the Domain types, except for transaction\_date, which uses a Logical type.)

Attribute Name	Type	Other Information and Notes
transaction_id	Domain: Numeric Id	Primary UID (unique identifier). (Unique transaction ID number)
patron_id	Domain: Numeric Id	M (mandatory, that is, must not be null). Must match a patron_id value in the Patrons entity.
book_id	Domain: Book Id	M (mandatory, that is, must not be null). Must match a book_id value in the Books entity.
transaction_date	Logical type: Datetime	M (mandatory, that is, must not be null). Date and time of the transaction.
transaction_type	Domain: Numeric Id	M (mandatory, that is, must not be null). (Numeric code indicating the type of transaction, such as 1 for checking out a book.)

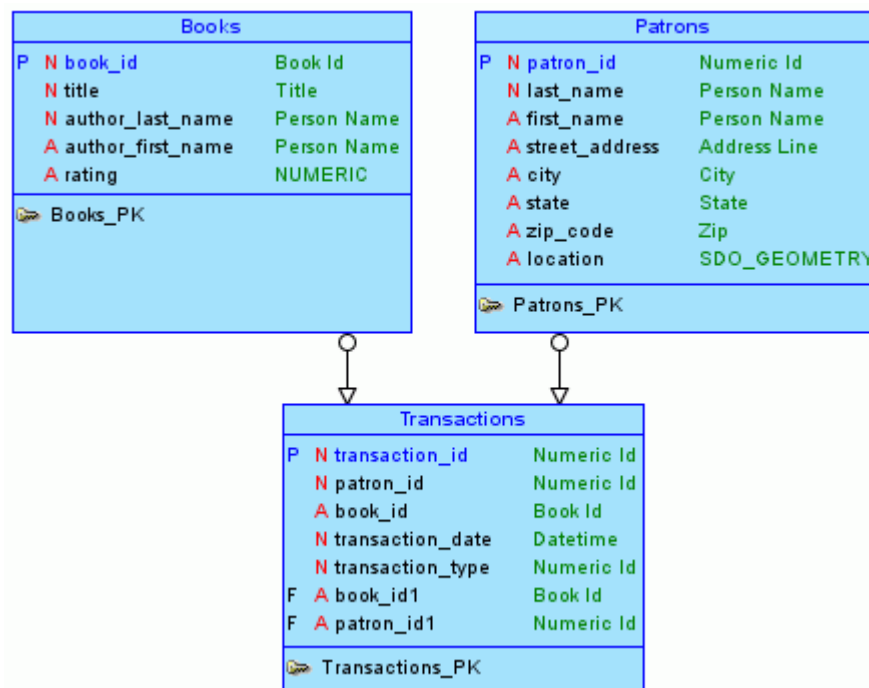
6. Click OK to finish creating the Transactions entity.
7. Go to [Section 2.1.5, "Creating Relations Between Entities"](#).

## 2.1.5 Creating Relations Between Entities

Relations show the relationships between entities: one-to-many, many-to-one, or many-to-many. The following relationships exist between the entities:

- Books and Transactions: one-to-many. Each book can be involved in multiple sequential transactions. Each book can have zero or one active checkout transactions; a book that is checked out cannot be checked out again until after it has been returned.
- Patrons and Transactions: one-to-many. Each patron can be involved in multiple sequential and simultaneous transactions. Each patron can check out one or many books in a visit to the library, and can have multiple active checkout transactions reflecting several visits; each patron can also return checked out books at any time.

Create the relationships as follows. When you are done, the logical model pane in the main area should look like the following figure (using Bachman notation, which you can change to Barker by clicking View, then Logical Diagram Notation, then Barker Notation):



1. In the logical model pane in the main area, arrange the entity boxes as follows: Books on the left, Patrons on the right, and Transactions either between Books and Patrons or under them and in the middle. (If the pointer is still cross-hairs, click the Select icon at the top left to change the pointer to an arrow.)

Suggestion: Turn off auto line routing for this exercise: right-click in the Logical pane, and ensure that Auto Route is not checked.

2. Click the New 1:N Relation icon.
3. Click first in the Books box, then in the Transactions box. A line with an arrowhead is drawn from Books to Transactions.

4. Click the New 1:N Relation icon.
5. Click first in the Patrons box, then in the Transactions box. A line with an arrowhead is drawn from Patrons to Transactions.
6. Optionally, double-click a line (or right-click a line and select Properties) and view the [Relation Properties](#) information.
7. Go to [Section 2.2, "Develop the Relational Model"](#).

## 2.2 Develop the Relational Model

The relational model for the library tutorial database consists of tables that reflect the entities of the logical model (Books, Patrons, and Transactions) and all attributes of each entity. In the simplified data model for this tutorial, a single relational model reflects the entire logical model; however, for other data models you can create one or more relational models, each reflecting all or a subset of the logical model. (To have a relational model reflect a subset of the logical model, use the "filter" feature in the dialog box for engineering a relational model.)

Develop the relational model as follows:

1. With the logical model selected, click **Design**, then **Engineer to Relational Model**. The [Engineering](#) dialog box is displayed.
2. Accept all defaults (do not filter), and click **Engineer**. This causes the Relational\_1 model to be populated with tables and other objects that reflect the logical model.
3. Optionally, expand the Relational Models node in the object browser on the left side of the window, and expand Relational\_1 and nodes under it that contain any entries (such as Tables and Columns), to view the objects created.
4. Change the name of the relational model from Relational\_1 to something more meaningful for diagram displays, such as Library (relational). Specifically, right-click Relational\_1 in the hierarchy display, select **Properties**, in the General pane of the [Model Properties - <name> \(Relational\)](#) dialog box specify **Name** as `Library (relational)`, and click **OK**.
5. Go to [Section 2.3, "Generate DDL"](#).

## 2.3 Generate DDL

Generate Data Definition Language (DDL) statements that you can use to create database objects that reflect the models that you have designed. The DDL statements will implement the physical model (type of database, such as Oracle Database 11g) that you specify.

Develop the physical model as follows:

1. Optionally, view the physical model before you generate DDL statements:
  - a. With the Library logical model selected, click **Physical**, then **Open Physical Model**. A dialog box is displayed for selecting the type of database for which to create the physical model.
  - b. Specify the type of database (for example, Oracle Database 11g), and click **OK**. In the hierarchy display on the left side of the window, a Physical Models node is added under the Library relational model node, and a physical model reflecting the type of database is created under the Physical Models node.



- c. Expand the Physical Models node under Library (the relational model), and expand the newly created physical model and nodes under it that contain any entries (such as Tables and Columns), to view the objects created.
2. Click **File**, then **Export**, then **DDL File**.
3. Select the database type (for example, Oracle Database 11g) and click **Generate**. The [DDL Generation Options](#) dialog box is displayed.
4. Accept all defaults, and click **OK**. A DDL file editor is displayed, with SQL statements to create the tables and add constraints. (Although you can edit statements in this window, do not edit any statements for this tutorial exercise.)
5. Click **Save** to save the statements to a .sql script file (for example, `create_library_objects.sql`) on your local system.  
  
Later, run the script (for example, using a database connection and SQL Worksheet in SQL Developer) to create the objects in the desired database.
6. Click **Close** to close the DDL file editor.
7. Go to [Section 2.4, "Save the Design"](#).

## 2.4 Save the Design

Save the design by clicking **File**, then **Save**. Specify the location and name for the XML file to contain the basic structural information (for example, `library_design.xml`). A directory or folder structure will also be created automatically to hold the detailed information about the design, as explained in [Section 1.3.1, "Database Design"](#).

Continue creating and modifying design objects, if you wish. When you are finished, save the design again if you have made any changes, then exit SQL Developer Data Modeler by clicking **File**, then **Exit**.

You can later open the saved design and continue working on it, as explained in [Section 1.6, "Saving, Opening, Exporting, and Importing Designs"](#).



---

---

## Dialog Boxes for Data Modeler

This chapter contains reference information about dialog boxes for SQL Developer Data Modeler. The dialog boxes sometimes have multiple panes, each reflecting a logical grouping of properties for that type of object.

For an explanation of any dialog box, click the **Help** button or press the **F1** key.

### 3.1 Add Event

This dialog box is displayed when you click the Add (+) icon in an Events pane for an object for which events are relevant.

**New Event:** Lets you create a new event and associated this event with the object.

**Available Event:** Lets you select an existing event to be associated with the object.

**Name** (new event): Name of the new event.

**Type** (new event): Type of the new event: Clock/Calendar, Person, or Storage.

**All Events:** List of events from which to select for an available event.

### 3.2 Arc Properties

This dialog box displays the properties of an exclusive relationship group (arc). Arcs are displayed in a logical model or relational model diagram, and are explained in [Section 1.3.4.7](#).

#### General

**Name:** Name of the arc.

**Entity** (logical model) or **Table** (relational model): Name of the logical model entity or relational model table associated with this arc.

**Mandatory** (relational model): Controls whether there must exist only one relationship for each instance of the entity or table.

**Include into DDL Script** (relational model): Controls whether the arc requirements are included in any generated DDL statements.

#### Relations (logical model) or Foreign Keys (relational model)

Lists the logical model relations or relational model foreign keys in this arc. To view or edit the properties of a listed relation or foreign key, double-click its item.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

### 3.3 Attribute Properties

This dialog box displays the properties of an attribute, which is a component of an entity in the [Logical Model](#).

**General**

**Name:** Name of the attribute.

**Synonym:** Synonym for the attribute.

**Preferred Abbreviation:** Name that will be used for any corresponding table column during forward-engineering *if* the Use Preferred Abbreviations option is enabled in the [Engineering](#) dialog box.

**Long Name:** Long name in the format: entity-name.attribute-name

**Allow Nulls:** Controls whether null values are allowed for the attribute. If this option is enabled, a non-null value is mandatory.

**Datatype:** Enables you to specify a domain, logical type, distinct type, collection type, or structured type as the data type of the attribute. You can click the ellipsis (...) button to specify further details for the selected type.

**Entity:** Name of the entity with which the attribute is associated.

**Source Name:** User-specified name of the source for this attribute.

**Source Type:** Manual, System, Derived, or Aggregate.

**Formula Description:** For a derived or aggregate source type, the formula for the attribute.

**Scope:** For a structured type with Reference enabled, limits the scope by specifying the table in which the type is implemented.

**Type Substitution:** For a structured type with Reference disabled, or for a structured type applied to an entity, controls whether a substitutional structured type is generated in the DDL.

**Default and Constraint**

**Constraint Name:** Name of the constraint.

**Default Value:** Default value for the attribute.

**Use Domain Constraints:** Controls whether the properties defined in [Domains Administration](#) for the associated domain are used. If this option is disabled, you can use the remaining fields to specify the database type for the constraint and the ranges or a list of values.

**Constraint:** Enables you to specify a constraint for one or more types of databases.

**Ranges:** Enables you to specify one or more value ranges for the attribute.

**Value List:** Enables you to specify a list of valid values for the attribute.

**Permitted Subtypes**

For a structured data type, lists all subtypes for the attribute, and lets you specify whether each is permitted for the attribute.

**Engineer To**

Enables you to specify the relational models to which this attribute should be propagated in forward engineering operations.

**Engineer:** Controls whether the attribute is propagated to the specified **Relational Design** (model) during forward engineering operations.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

## 3.4 Change Object Names Prefix

This dialog box is displayed if you specify the Change Object Names Prefix command on the logical model or a relational model. Specify the new prefix to replace a specified current prefix. You can restrict the changes to selected types of objects.

**Current Prefix:** Prefix to be changed to the new prefix.

**New Prefix:** Prefix to replace all occurrences of the current prefix in names for the selected types of objects.

**Case Sensitive:** Controls whether case sensitivity is applied in searching for old strings to be replaced by new strings.

**Apply to:** Specify the types of objects for which to perform the prefix replacement. The types of objects depend on the type of model (logical or relational).

## 3.5 Change Request Properties

Change requests allow details of proposed changes affecting model objects to be recorded. They may also be used to keep a historical record of implemented or rejected changes. A change request object can be associated with one or more model objects.

**Name:** Name of the change request.

**Comment:** Optional descriptive comment text.

**Notes:** Optional note text, such as background information or implementation notes.

**Reason:**

**Status:** Current status of the change request: Proposed, Agreed (accepted), Implemented, Implementing (implementation in progress), or Rejected.

**Completed:** Indicates whether the work on the change request is completed.

**Request Date:** The date when the change request was made.

**Completion Date:** The date when the work on the change request was completed.

**Implementation Note:** Optional note about the implementation of the change request.

**Summary:** Displays read-only summary information.

## 3.6 Collection Type Properties

This dialog box displays the properties of collection type, which is part of the [Data Types Model](#).

### General

**Name:** Name of the collection type.

**Collection Type:** ARRAY for a Varray collection type (an ordered collection of elements), or COLLECTION for a nested table collection type (can have any number of elements).

**Max Element:** For a Varray type, the maximum number of elements it can contain.

**Datatype:** The data type of each element.

### Comments, Notes, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.7 Column Properties

This dialog box displays the properties of a table column, which is part of the [Relational Models](#).

### General

**Name:** Name of the column.

**Long Name:** Long name in the format: table-name.column-name

**Engineer:** Controls whether the column will be considered during reverse engineering operations. If this option is disabled, this column and its properties are not considered when the table is reverse engineered into the associated entity in the logical model.

**Allow Nulls:** Controls whether null values are allowed for the column. If this option is enabled, a non-null value is mandatory.

**Table:** Name of the table with which the column is associated.

**Datatype:** Enables you to specify a domain, logical type, distinct type, collection type, or structured type as the data type of the column. You can click the ellipsis (...) button to specify further details for the selected type.

**Type:** Manual, System, Derived, or Aggregate.

**Formula Description:** For a derived or aggregate type, the formula for the attribute.

**Scope:** For a structured type with Reference enabled, limits the scope by specifying the table in which the type is implemented.

**Type Substitution:** For a structured type with Reference disabled, or for a structured type applied to an entity, controls whether a substitutional structured type is generated in the DDL.

### Default and Constraint

**Constraint Name:** Name of the constraint.

**Uses Default:** Controls whether the default value is used for this column.

**Default Value:** Default value for the column.

**Use Domain Constraints:** Controls whether the properties defined in [Domains Administration](#) for the associated domain are used. If this option is disabled, you can use the remaining fields to specify the database type for the constraint and the ranges or a list of values.

**Constraint:** Enables you to specify a constraint for one or more types of databases.

**Ranges:** Enables you to specify one or more value ranges for the column.

**Value List:** Enables you to specify a list of valid values for the column.

### **Permitted Subtypes**

For a structured data type, lists all subtypes for the attribute, and lets you specify whether each is permitted for the attribute.

### **Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

### **Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.8 Common Information in Dialog Boxes**

Many dialog boxes let you include comments and notes about a modeling object, and to see a summary of the current information. You can also often view, add, and delete business model objects (documents, responsible parties, and so on) that are associated with the object.

### **Comments**

Optional descriptive comment text about the modeling object.

Note that for some objects you can specify both Comments and Comments in RDBMS. Any Comments in RDBMS text is included in DDL statements that are generated for creating the object in a database.

### **Notes**

Optional note text, such as background information or implementation notes.

### **Measurements**

Lists any relevant measurements. To add a measurement, click the Add (+) icon; to remove a measurement from the object definition, select it and click the Remove (X) icon. (See also [Measurement Properties](#).)

### **Change Requests**

Lists any relevant change requests. To add a change request, click the Add (+) icon; to remove a change request from the object definition, select it and click the Remove (X) icon. (See also [Change Request Properties](#).)

**Responsible Parties**

Lists any relevant responsible parties. To add a responsible party, click the Add (+) icon; to remove a responsible party from the object definition, select it and click the Remove (X) icon. (See also [Responsible Party Properties](#).)

**Documents**

Lists any relevant documents. To add a document, click the Add (+) icon; to remove a document from the object definition, select it and click the Remove (X) icon. (See also [Document Properties](#).)

**Summary**

Displays read-only summary information.

## 3.9 Compare Mapping

This dialog box is displayed if you click View Compare Mapping in the [Relational Models](#) dialog box.

It displays any mappings between source and target columns; and for each mapping you can select, edit, or delete it.

## 3.10 Compare Modeling Designs

Lets you apply a filter to restrict the types of objects and specific objects to be included when models are imported.

**Filter:** You can include all objects, or only new, deleted, or modified objects.

## 3.11 Compare Models

This dialog box lets you control the types of objects and specific objects to be merged when two relational models are compared and merged. Objects from the selected model in the file that you opened are on the left, and objects from the selected model in the currently open design are on the right.

You can expand and collapse the display of the relational models. If you select a specific table or view, the Details, Options, and perhaps other panes at the bottom display information for that object.

**Filter:** Lets you display all objects in both models, or display only new, deleted, or modified objects

To update the selected relational model in the current design by merging objects from the selected relational model in the external file, click **Merge**.

**Options**

**Include Comments:** Controls whether comments are included in the compare and merge operation.

**DDL type:** Type of DDL statements to be generated.

**Replace Existing Files:** Controls whether any existing DDL files will automatically be replaced.

**Unload Directory:** Directory or folder in which to save DDL files.



## 3.12 Contact Properties

This dialog box displays the properties of a contact object, which is a type of [Business Information](#) object.

### General

**Name:** Name of the contact object. For example, it might be the name of a person, a role, a group, a department, or a company.

### Emails

Lists any email objects. To add an email object, click the Add (+) icon; to remove an email object from the contact definition, select it and click the Remove (X) icon. (See also [Email Properties](#).)

### Locations

Lists any location objects. To add a location object, click the Add (+) icon; to remove a location object to remove a location object from the contact definition, select it and click the Remove (X) icon. (See also [Location Properties](#).)

### Telephones

Lists any telephone objects. To add a telephone object, click the Add (+) icon; to remove a telephone object from the contact definition, select it and click the Remove (X) icon. (See also [Telephone Properties](#).)

### URLs

Lists any relevant URLs. To add a URL, click the Add (+) icon; to remove a URL from the contact definition, select it and click the Remove (X) icon. (See also [Resource Locator Properties](#).)

### Responsible Parties, Comments, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.13 Cube Properties

This dialog box displays the properties of a cube, which is part of the multidimensional model. It is displayed when you view the properties of a cube or when you import cube views metadata. (Importing cube views metadata enables you to create a multidimensional model based on an existing implementation.)

Cubes are first-class database objects that store data in a dimensional format. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the cube.

**Virtual:** Indicates whether this is a virtual cube or an actual cube. A virtual cube combines multiple actual cubes into a single logical cube.

**Partitioned:** Controls whether the cube is partitioned. Partitioning is a method of physically storing the measures in a cube. It improves the performance of large measures.

**Part. Dimension:** Dimension for partitioning the cube. (For example, if your partitioning strategy is driven primarily by life-cycle management considerations, then

you should partition the cube on the Time dimension.) The dimension must have at least one level-based hierarchy and its members should be distributed evenly, such that every parent at a particular level has roughly the same number of children.

**Part. Hierarchy:** Hierarchy to be used for partitioning. If the dimension has multiple hierarchies, choose the one that has the most members; it should be defined as the default hierarchy.

**Part. Level:** Level to be used for partitioning. Each dimension member at that level is stored in a separate partition, along with its descendants. Any dimension members that are at higher levels or are not in the hierarchy are stored together in the top partition. The size of the top partition should not exceed the size of the level-based partitions.

**Global Composites:** Controls whether the cube will use one global composite or multiple composites. An unpartitioned cube always has one composite for the cube, whether it is compressed or uncompressed. A partitioned compressed cube always has a composite for each partition. A choice between single (global) and multiple composites is available only for uncompressed, partitioned cubes.

When in doubt, do not choose this option. The cube will have one composite for each partition.

**Compressed Composites:** Controls whether composites are compressed or uncompressed. Cubes that are very sparse often use a compressed composite.

**Full Cube Materialization:** Materialized view for a cube that has been enhanced with materialized view capabilities. A cube materialized view can be incrementally refreshed at prescheduled times or on demand through the Oracle Database materialized view subsystem. It can also serve as a target for transparent rewrite of queries against the source tables.

### Entities

Lists unselected, available entities for the cube on the left and selected cubes on the right. Use the arrow keys to move selected entities from one side to the other.

**Use Fact Entities Only:** Controls whether only tables in a star schema that contain facts are displayed.

### Joins

Lists any joins for the cube. To add a join, click the Add (+) icon; to remove a join from the cube definition, select it and click the Remove (X) icon. (See also [Join Properties](#).)

### Dimensions

Lists any dimensions for the cube. To add a dimension, click the Add (+) icon; to remove a dimension from the cube definition, select it and click the Remove (X) icon. (See also [Dimension Properties](#).)

**Default Operator:** Default operator assigned to the dimension.

**IsSparse:** If selected, null values and empty fields are excluded, in order to manage space.

### Measures

Calculated measures can add information-rich data to a cube. The data set is calculated on the fly, so no data is stored. You can add as many calculated measures as you like without increasing the size of the database.

**Precalculated Slices**

Lists any slices for which values are precalculated (precomputed) and stored in the cube during data maintenance. To add a precalculated slice, click the Add (+) icon; to remove a precalculated slice from the cube definition, select it and click the Remove (X) icon. (See also [Slice Properties](#).)

**Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Mycubes* if the Short Name is *Mycube*

**SQL Access to Oracle AW**

Lists any relevant SQL Access to Oracle Analytic Workspaces (AW) objects. To add a SQL Access to Oracle AW object, click the Add (+) icon; to remove a SQL Access to Oracle AW object from the cube definition, select it and click the Remove (X) icon. To edit a SQL Access to Oracle AW object, double-click its item, or click its item and click the Properties icon. (See also [SQL Access to Oracle AW Properties](#).)

**Description**

Description of the cube.

**Partitioning Description**

Description of the partitioning for the cube.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.14 Data Dictionary Import (Metadata Extraction)

This wizard is displayed when you click File, then Import, then Data Dictionary. It enables you to connect to an existing database (Oracle or supported third-party) and to create one or more relational models based on the metadata in that database.

The specific steps and fields depend on the type of database that you connect to. This topic describes information for an Oracle Database connection.

**Connect to Database**

Displays a list of database connections, from which you must select one. If no connections exist or if the one you want is not displayed, click **Add** to display the [Database Connection Editor](#) dialog box.

**Select Schema/Database**

Select the schemas from which to import. Note: Metadata will be extracted only for schemas that the user associated with the connection is authorized to access. A separate subview will be created within the relational model for each Oracle Database schema from which metadata is extracted. (To see a subview, expand the Subviews node, right-click the subview name, and select Show Diagram.)

**Secondary Tables:** Includes secondary tables in the metadata extraction.

**Spatial Properties:** Includes Oracle Spatial properties in the metadata extraction.

**Select Objects to Import**

Select the objects for the metadata extraction. For each type of object to be imported, click its associated tab (Tables, Views, Users, and so on), and select the desired objects. For each tab, you can select or deselect all, or you can select and deselect individual objects of that type.

**Generate Design**

Displays a summary page with the number of objects of specified types to be imported. (If no objects or any type are to be imported, go back to the [Select Objects to Import](#) page and select at least one object.)

To perform the import operation, click **Finish**.

## 3.15 Database Connection Editor

This dialog box is displayed if you add or edit a database connection while using the [Data Dictionary Import \(Metadata Extraction\)](#) wizard. The specific fields depend on the connection type that you select; this topic describes the fields for an Oracle connection.

**Type:** Type of database to which to connect: Oracle, Microsoft SQL Server 2000 or 2005, IDB DB2 or UDB, or JDBC ODBC Bridge.

For a third-party database connection, you must have specified the appropriate driver in [Third Party JDBC Drivers](#) under General Options (see [Section 1.5](#)).

**Name:** A descriptive name or alias for the connection. Example: HR\_Local

**Host:** Host system for the Oracle database. Example: localhost

**Port:** Listener port. Example: 1521

**SID:** Database name. Example: orcl

**Username:** Name of the database user for the connection. This user must have sufficient privileges to perform the tasks that you want perform while connected to the database.

**Password:** Password associated with the specified database user.

## 3.16 DDL File Editor

This dialog box is displayed after you select a relational model for which to generate Data Definition Language (DDL) statements. It is also displayed if you click View, then DDL File Editor, or if you click the Generate DDL toolbar icon when a relational model diagram is displayed.

**Physical model (list):** Select the physical model (type of database) for which to generate DDL statements. Example: Oracle 11g

**Generate:** Displays the [DDL Generation Options](#) dialog box. After you select the options, the resulting DDL statements are displayed in this dialog box.

**Clear:** Clears the display of generated DDL statements. If the generated DDL code does not meet your needs, click **Clear**, make any necessary changes to the model, and start the generation process again.

**Find:** Displays a box for finding a text string in the generated DDL statements.

**Save:** Lets you save the generated DDL statements to a script file.

## 3.17 DDL Generation Options

This dialog box is displayed when you click Generate in the [DDL File Editor](#) dialog box. These options control the content to be included in the generated script.

Buttons at the bottom: **Design Rules** displays the [Design Rules](#) dialog box, enabling you to check your current design for any violations of predefined design rules before you generate the DDL. **Include Comments** causes comments to be included in the statements. **Apply Name Substitution** causes old name strings to be replaced with new strings as specified in the [Name Substitution](#) pane. You can also select Regular DDL, Advanced Interactive, or Advanced CL DDL.

### 'Create' Selection

Lets you select types of objects and specific objects for which to generate CREATE statements. You can select Tree View for an expandable display of all available objects, or a tab (Tables, PK and UK Constraints, and so on) for a subset of objects.

### 'Drop' Selection

Adds DROP statements for objects of the selected types before any CREATE statements, to drop any existing objects with the same name before creating new objects.

You can include drop dependencies to add CASCADE to the DROP statements.

### Name Substitution

Lets you specify old strings to be replaced with new strings in object names when the DDL statements are generated.

**Selected:** Controls whether the item is enforced in the generation process.

**Case Sensitive:** Controls whether case sensitivity is applied in searching for old strings to be replaced by new strings.

### Oracle Errors to Mask

For Advanced DDL only: Specifies any Oracle errors to be ignored during script execution. For the error Number or Type, specify a hyphen and significant digits without leading zeroes; for example, specify -942 for ORA-00942. The error description is informational only and does not affect the script execution.

## 3.18 Design Properties

This dialog box displays the properties of the [Database Design](#).

### General

**Name:** Name of the database design.

### Comments, Notes, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.19 Design Rules

This dialog box enables you to check your current design (logical, relational, and process models) for any violations of predefined Data Modeler design rules. You can check all design rules or selected rules; each violation of the specified rules results in a warning or error message, as appropriate. You are encouraged to check your models

against the design rules often, especially before any forward or reverse engineering operations and before generating DDL statements for a physical model.

Examples of design rule warnings and errors include a primary key with the wrong naming standards, a flow without information structures, and an external agent without data elements.

For any error, you can double-click its item to display the properties for the associated object, where you can correct the problem.

**Expand All:** Expands the display to show all design rules in all categories and subcategories. When one or more categories are expanded, you can select individual rules, ranges of rules, or a combination.

**Collapse All:** Collapses the display to show only the design rule major category names (General, Logical, Relational, Process Model, Physical).

**Apply All:** Applies all design rules to the current design; displays a warning or error message for each violation of a rule.

**Apply Selected:** Applies only the selected design rules to the current design; displays a warning or error message for each violation of a rule.

**Clear:** Clears any displayed warning and error messages.

## 3.20 Dimension Properties

This dialog box displays the properties of a dimension in a multidimensional model.

A dimension (or more precisely, a cube dimension) is a first-class database object. It stores a list of values that serves as an index to the data stored in a cube. These values, or dimension members, represent all levels of aggregation. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the dimension. Examples: Customer, Product, Channel, Time

**Use Natural Keys:** Controls whether natural (source) keys or surrogate keys are used from the relational sources for the dimension members. If this option is enabled, source keys are read from the relational sources without modification. If this option is disabled, surrogate keys ensure uniqueness by adding a level prefix to the members while loading them into the analytic workspace.

**Time Dimension:** Controls whether the dimension is a Time dimension or a User dimension. If this option is enabled, you must define a Time dimension with at least one level to support time-based analysis, such as a custom measure that calculates the difference from the prior period.

### Hierarchies

Lists hierarchies associated with the dimension. To view or edit a hierarchy definition, double-click its item. (See also [Hierarchy Properties](#).)

### Levels

Lists levels associated with the dimension. To view or edit a level definition, double-click its item. (See also [Level Properties](#).)

**Slow Changing Attributes**

For a Slowly Changing Dimension (SCD), lists slowly changing attributes associated with the dimension. To view or edit an attribute definition, double-click its item. (See also [Attribute Properties](#).)

**Calculated Members**

Lists calculated members associated with the dimension. A calculated member enables the summation (simple plus or minus) or aggregation (using the Aggregation operator for the cube) of a set of dimension members that are specified by a user. To add a calculated members, click the Add (+) icon; to remove a calculated members from the dimension definition, select it and click the Remove (X) icon.

**Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Mydimensions* if the Short Name is *Mydimension*

**Description**

Description of the dimension.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.21 Display Properties

This dialog box displays the properties of a display object.

**General**

**Name:** Name of the display object.

**Visible:** (Does not apply to this model.)

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.22 Distinct Type Properties

This dialog box displays the properties of a distinct type, which is part of the [Data Types Model](#).

**General**

**Name:** Name of the user-defined distinct type.

**Logical Type:** The logical type from which the distinct type is derived.

(The remaining fields -- **Size**, **Precision**, and **Scale** -- are available only if they apply to this distinct type.)

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.23 Document Properties

This dialog box lets you view and modify information for a document object, which is a type of [Business Information](#) object.

**General**

**Name:** Name of the document object.

**Reference:** Descriptive phrase with reference information about the object, such as its location.

**Type:** Descriptive phrase indicating the type of document, such as General Procedure or Technical Specification.

**Elements**

Lists the objects currently associated with this document. To view the properties of any listed object, double-click its entry.

**Comments, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.24 Domain Properties (Domains Model)

This dialog box displays information about an existing domain. Much of the information is read-only, and some fields apply only to certain kinds of domains. To add a user-defined domain or to remove a domain, click **Tools**, then **Domains Administration**, and use the [Domains Administration](#) dialog box.

**General**

**Name:** Name of the domain object.

**Synonym:** Synonym for the domain.

**Long Name:** Long name for the domain.

**File Name:** Name of the file containing the domain definition.

**Logical Type:** Logical type of the domain.

**Size:** Maximum size of data in columns based on the domain.

**Precision:** For a numeric domain, the maximum number of significant decimal digits.

**Scale:** For a numeric domain, the number of digits from the decimal point to the least significant digit.

**Check Constraint:** Constraints, both generic and database product-specific, on data values permitted for the domain. If constraints apply to the domain, you can double-click to edit the generic and product-specific constraints.

**Ranges:** Ranges of data values permitted.



**Value List:** List of data values permitted.

**Used In**

Lists attributes (logical model) and columns (relational model) that are based on this domain.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.25 Domains Administration

This dialog box is displayed if you click Tools, then Domains Administration.

**Domains File:** XML file containing the domain definitions (if other than the default). You can click **Select** to search for a file.

**Available Domains:** Displays the available domains (types). Select a domain to view or edit its properties. (Some properties are relevant only to specific domains.)

**Name:** Name of the selected domain (if one is selected).

**Logical Type:** Logical type of the selected domain.

**Size:** Maximum size of the data for this type.

**Precision:** For a numeric domain, the maximum number of significant decimal digits.

**Scale:** For a numeric domain, the number of digits from the decimal point to the least significant digit.

**Synonym:** Synonym for the domain.

**Comment:** Comment about the domain.

**Check Constraint:** Lets you view and edit constraints (general and specific to physical models) for the domain.

**Ranges:** Lets you specify one or more data value ranges for the domain.

**Value List:** Lets you specify data values for the domain.

## 3.26 Email Properties

This dialog box displays the properties of an email object, which is a type of [Business Information](#) object.

**General**

**Name:** Name of the email object.

**Email Address:** Email address in the standard format. Example: smith@example.com

**Email Type:** Descriptive phrase indicating the type of email object, using any naming scheme suited to your needs. Examples: Work, Personal

**Contacts**

Lists the contact objects currently associated with this email object. To view the properties of any listed contact object, double-click its entry.

**Comments, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.27 Engineering

Depending on the context, this dialog box lets you update the logical model from a selected relational model or update a relational model from the logical model, control which parts of the logical model are reflected in a relational model, or restrict the types of objects and specific objects to be merged when two relational models are compared and merged.

**Filter:** You can include all objects, or only new, deleted, or modified objects.

### Details

Displays detailed information about the selected model object.

### General Options

**Show Engineering in the Main Browser:** Controls whether a check box is added for each object in the browser window, with the box checked if the object has been engineered.

**Engineer Coordinates:** Controls whether to position the tables resulting from the forward engineering process in the same way as their source entities.

**Engineer Only Objects Created in "<model>":** Controls whether only objects that were explicitly created in the specified model are engineered, or whether those objects plus other objects that were created as a result of previous forward or reverse engineering operations should be engineered.

**Apply Name Translation:** Controls whether formal names are translated to abbreviated names when the logical model is forward engineered to a relational model, and whether abbreviated names are translated to format names when a relational model is reverse engineered to the logical model. Name translation is applied only for valid names. In addition, translations between the words *entity/attribute/key* and *table/column/index* are performed.

**Use Preferred Abbreviations:** Controls whether any specified preferred abbreviations for entities and attributes are used for corresponding table and column names when the logical model is engineered to create a relational model.

### Copy/Compare Options

Lets you select types of objects (*entity/table*, *attribute/column*, *key/index*, *relation/foreign key*, *entity view/view*), and for each pair indicate which of its properties are selected for use in the forward or reverse engineering operation. You can also specify the following options.

**Show Selected Properties Only:** Controls whether all properties for displayed objects are shown or only properties enabled (checked) under Selected.

**Don't Apply for New Objects:**

**Exclude Unchecked Objects from Tree:** Controls whether

**Update Tree:** Updates the logical and relational model tree displays to reflect the currently selected options.

### Synchronization of Deleted Objects

Lets you select and deselect pairs of objects, with each pair containing a deleted object and the associated object to be deleted in the generated model in the pair is selected.

**Overlapping and Folding Unique Identifiers (UIDs)**

For engineering from the logical model to a relational model, displays information about any keys that overlap or are overlapped.

## 3.28 Entity Properties

This dialog box displays the properties of an entity, which is part of the [Logical Model](#).

**General**

**Name:** Name of the entity.

**Synonyms:** Synonyms for the entity.

**Synonym to display:** Synonym to display for the entity.

**Preferred Abbreviation:** Name that will be used for any corresponding table during forward-engineering *if* the Use Preferred Abbreviations option is enabled in the [Engineering](#) dialog box.

**Long Name:** Long name to be used in *entity-name.attribute-name* displays.

**FWD Engineer Strategy:** This field is important when you are working with supertype and subtype entities. It defines the strategy of engineering entities to relational model tables:

- Single Table: The subtype entity metadata are included in the supertype table. There are no subtype tables.
- Table per Child: The supertype entity metadata is included in the subtype tables. There is no supertype table.
- Table for each Entity: A separate table is generated for each supertype and subtype entity.

**Based on Structured Type:** If the entity is based on a structured type, select it from the list. The entity will contain the attributes as defined in the structured type.

**Super Type:** If the entity is a subtype, select its supertype from the list. The entity inherits all attributes from the specified supertype.

**Source:** Description of the source type. Examples: DDL, COBOL Copybook, IDMS Schema

**Type:** Classification type, from the list defined in the General Options. Examples: Fact, Dimension, Logging, Summary, Temporary.

**Scope:** For an entity classified as Temporary, you can specify a scope, such as Session or Application.

**Allow Type Substitution:** For a structured type with Reference disabled, or for a structured type applied to an entity, controls whether a substitutional structured type generation in the DDL is allowed.

**Attributes**

Lists the attributes currently defined for the entity. The properties for each attribute include its name and data type, and whether it is the primary unique identifier (Primary UID), a relation unique identifier (Relation UID, comparable to foreign key), or a required field (M, for mandatory).

To add an attribute, click the Add (+) icon; to delete an attribute, select it and click the Remove (X) icon; to view the properties of an attribute, double-click in the cell to the left of the name.

**Unique Identifiers**

Lists the unique identifiers (UIDs) currently defined for the entity. The properties for each key include its name and whether it is included in the primary unique identifier (PUID).

To make the attribute a unique identifier, click the Add (+) icon; to make the attribute not a unique identifier, select it and click the Remove (X) icon; to view the properties of a unique identifier, double-click its name.

**Relations**

Lists any relations associated with the entity, To view the properties of a relation, double-click its name.

**Volume Properties**

**Volumes: Minimum:** Minimum data volume for the entity.

**Volumes: Expected:** Expected or typical data volume for the entity.

**Volumes: Maximum:** Maximum data volume for the entity.

**Growth Rate: Percent:** Expected growth rate percentage for the entity, for each period as specified in the next field.

**Growth Rate: Year/Month/Day:** The period (year, month, or day) to which the expected growth rate applies.

**Normal Form:** The required normal form (database normalization) for the entity: None, First, Second, Third, or Fourth.

**Adequately Normalized?:** YES indicates that the model is sufficiently normalized. NO indicates that the model is not sufficiently normalized, and that additional normalization may be required on the relational model.

**Engineer To**

Enables you to specify the relational models to which this entity should be propagated in forward engineering operations.

**Engineer:** Controls whether the entity is propagated to the specified **Relational Design** (model) during forward engineering operations.

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.29 Event Properties

This dialog box displays the properties of an event, which is part of the [Process Model](#).

**General**

**Name:** Name of the event.

**Synonym:** Synonym for the event.

**Flow:** Flow object that is associated with the event. To view or edit the [Flow Properties](#), click the flow name.

**Event Text:** Application-defined or user-defined event text string.

**Type:** Application-defined or user-defined string identifying the type of event.

**Process Triggered**

Lists the process triggered by the event. To view the [Process Properties](#), double-click its entry.

**Days When Run**

Specify options for days when the event is to run: unlimited or limited number of occurrences; day or days of the week, or day of one or all months; specific quarter or all; and specific fiscal or calendar year, or all years. Zero (0) in a field means all or unlimited.

**Times When Run**

Specify options for times during the day when the event is to run: open or close of the business day, when convenient during the day, or at a specified time or at a specified interval.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

### 3.30 Export Cube Views Metadata

By exporting cube views metadata, you can deploy them in a UDB v8.1 physical environment.

**Default Schema:** Name of the default schema.

**Dimensional Model:** Name of the multidimensional model to be exported.

**Relational Model:** Name of the relational model.

**Physical (DB2 or UDB) Model:** Name of the physical model.

### 3.31 Export to Microsoft XMLA

Exports the design to a Microsoft XMLA (XML for Analysis) file.

**Database Name:** Name of the database.

**Dimensional Model:** Name of the multidimensional model.

### 3.32 Export to Oracle Analytic Workspaces

By exporting to Oracle Analytic Workspace (AW), you can create the analytic workspace based on a multidimensional model. For more information about working with analytic workspaces, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

**General**

**Default Schema:** Name of the default schema.

**Dimensional Model:** Name of the multidimensional model.

**Relational Model:** Name of the relational model.

**Oracle Model:** Name of the Oracle model.

**Oracle AW Name:** Name of the Oracle analytic workspace.

**Export Mode:** Mode or scope of the export operation: re-create the analytical workspace, export all or only new calculated measures, or export only metadata.

**Populate Created Cubes:** Controls whether the created cubes are populated with their data.

**Output XML File:** Name of the XML file to contain the exported definition. To select an output directory or folder, click the ellipsis (...) button.

**JDBC Connection:** JDBC connection to use for the export operation.

**New JDBC Connection:** Enables you to create a new JDBC connection. To create a new connection, click **Create**.

**Test Selected Connection:** To test the selected JDBC connection, click **Test**. A message is displayed indicating either that the connection is available or that an error occurred.

**Cubes and Slices**

Displays cubes and slices selected for the export operation.

### 3.33 Export to Reporting Schema

Exports the design to the reporting schema (explained in [Section 1.7, "Data Modeler Reports"](#)). To perform the export, you must select a connection and click **OK**.

**Connections:** Displays the names of database connections, from which you must choose one to serve as the reporting repository. To create a new database connection, click the Add Connection (+) icon; to delete an existing connection, select it and click the Remove Connection (X) icon; to edit an existing connection, double-click its entry, or select it and click the Connection Properties icon.

**Comments:** Optional descriptive text or notes about the connections.

**Options: Export Diagrams as PDF:** If this option is enabled, diagrams in the design are exported in PDF format and stored as BLOBs; if this option is not enabled, diagrams are exported in XML format.

### 3.34 External Agent Properties

This dialog box displays the properties of an external agent object, which is part of the [Process Model](#).

**General**

**Name:** Name of the external agent.

**Synonym:** Synonym for the external agent.

**Type:** Type of external agent: Organization Unit, System, Role, or Other.

**Incoming Flows**

Displays flows coming into the external agent. To view the [Flow Properties](#), double-click the flow item or select the item and click the Properties icon. (To add a flow, use the New Flow icon on the data flow diagram.)

**Outgoing Flows**

Displays flows going out from the external agent. To view the [Flow Properties](#), double-click the flow item or select the item and click the Properties icon. (To add a flow, use the New Flow icon on the data flow diagram.)

**Data File Specification**

**Owner:** Owner of the external agent.

**Source:** Source of the external agent: where the information for the external agent comes from.

**File Name:** Name of the external agent file.

**Location:** Location of the external agent file.

**File Type:** Type of file: CSV (comma-separated values), fixed-length fields, Excel spreadsheet, or plain text.

**Field Separator:** Character (for example, the comma) that separates fields.

**Data Capture Type:** Specifies whether the data capture is a full refresh or is limited to a specified type of capture operation.

**Self Describing:** Indicates whether the field is self-describing.

**Skip Records:** Number of records to be skipped at the top of the data file.

**Text Delimiter:** Text delimiter character that is used in text strings (single or double quotes).

**Data Elements**

Lists the external data elements currently defined for the external agent. To add an element, click the Add (+) icon; to remove an element from the definition, select it and click the Remove (X) icon; to view the properties of an element, double-click it or select it and click the Properties icon.

**External Data:** Name of the external data object.

**Type:** Logical type of the external data.

**Starting Position:** Starting position of the element in the data file.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.35 External Data Properties

This dialog box displays the properties of a data element for an external agent object, which is part of the [Process Model](#).

**General**

**Name:** Name of the external data object. You might want it to include a sequence number, for example, MyExtData\_1.

**Comment:** Comment about this data element.

**Logical Type:** Logical type associated with this external data object.

**Starting Position:** Starting position of the external data object in the data file, where 0 (zero) is the first position.

**Description:** Optional description of the external data object.

**Structured:** Indicates whether this external data object is linked to a record structure.

**Record Structure:** If Structured is enabled, lists the available record structures; select one for this external data object.

### 3.36 Find Object

This dialog box is displayed if you click the Find icon when a diagram is selected. Specify an object name and an object type, then click **Find**.

**Object to Find:** Name of the object to find. You must specify a complete string match. For example, specifying *cust* or *customer* will not find a table named *customers*.

**Case Sensitive:** Controls whether the search for the object to find is case sensitive.

**Object Type:** Limits the search for objects to the specified type of object. (The available types depend on the type of diagram.)

### 3.37 Flow Properties

This dialog box displays the properties of a flow (data flow) object, which is part of the [Process Model](#).

#### General

**Name:** Name of the flow.

**Synonym:** Synonym for the flow.

**Source:** Source of the flow, for example, a process.

**Destination:** Destination of the flow, for example, an information store.

**Parent Flow:** Name of the parent flow, if any.

**Logging Flow:** Controls whether logging is activated for the flow.

**Event:** Flow event. You can click **New** to create a new flow event.

**Operations:** Options that control the types of operations that can be performed in the flow (create, read, update, delete).

#### Component Flows

Lists any component flows associated with the flow. To view or edit a component flow, double-click its item.

#### Information Structures

Lists any information structures associated with the flow. To add an information structure, click the Add (+) icon; to remove an information structure from the flow definition, select it and click the Remove (X) icon. (See also [Information Structure Properties](#).)



**External Data**

Lists any external data objects associated with the flow. To view or edit an external data object, double-click its item.

**System Objective**

Description of the system objective for this flow.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.38 Foreign Key Properties

This dialog box displays the properties of a foreign key relation, which is part of the definition of a table in [Relational Models](#).

**General**

**Name:** Name of the foreign key.

**Table:** Name of the table containing this foreign key (a column whose value must match a value in a primary key or unique key column in another table).

**PK/UK Index:** The primary key, unique key, or index to which this foreign key refers (for example, the primary key column in another value that must contain a matching value for each value in the foreign key column).

**Delete Rule:** Action to take automatically when a row in the referenced table (in PK/UK Index) is deleted and rows with that value exist in the table containing this foreign key: NO ACTION performs no action on these rows; CASCADE deletes these rows; SET NULL sets null all columns in those rows that can be set to a null value; RESTRICT prevents those rows from being deleted.

**Source Table Synonym:** Name or synonym of the table containing the primary or unique key to which this foreign key refers.

**Target Table Synonym:** Name or synonym of the table containing this foreign key column.

**Mandatory:** Controls whether referential integrity is enforced. If this option is enabled, referential integrity is enforced (that is, a matching value in the table for the referenced primary or unique key is mandatory; and if a matching value does not exist, a record cannot be created in the current table). If this option is not enabled, referential integrity is not enforced, and applications that use the database must deal with any potential problems caused by inconsistent data.

**Transferable (Updateable):** Controls whether the foreign key relationship is transferable (that is, updateable). In a non-transferable relationship, each foreign key value cannot be changed; for example, if a line item has a non-transferable relationship to an order, a line item cannot be reassigned later to another order. In a transferable relationship, a foreign key value can be changed; for example, an employee could be transferred later from one department to another.

**Generate in DDL:** Controls whether the foreign key creation is included when DDL statements are generated to be used to create the database.

**In Arc:** Controls whether the foreign key relationship should be included in an exclusive relationship group (arc). For more information, see [Section 1.3.4.7, "Arcs"](#).

**Associated Columns**

Lists each parent column and child column pair in the foreign key definition.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.39 Glossary Editor

This dialog box is displayed if you click Tools, then Glossary Editor, and either specify a file name that does not exist (for example, my\_terms.glossary) in a selected location to create a new glossary file, or select an existing glossary file for editing. (See also [Naming Standard](#) under [Data Modeler Options \(User Preferences\)](#).)

Do not modify glossary files (XML format) in an external text editor. Instead, create and modify glossaries using the glossary editor within Data Modeler.

**Name:** Name of the glossary. Example: Project XYZ glossary

**Description:** Brief description of the glossary.

**Incomplete Modifiers:** Controls whether all terms used in names do or do not need to be defined in the glossary. If this option is enabled, it is *not* mandatory that modifiers and qualifiers be defined in the glossary; as a consequence, name validation will succeed if name parts that are not in the glossary correspond to a modifier or qualifier in the name structure.

**Case Sensitive:** Controls whether name validation using this glossary will be case sensitive. For example, if this option is enabled, "Code" and "CODE" are considered different values.

**Unique Abbreviations:** Controls whether uniqueness of abbreviations is required. If this option is disabled, uniqueness is not required, thus allowing one abbreviation to be used for all forms of a single word. In this case, for example, ADMIN could be an abbreviation for Administrator, Administration, and Administrative (that is, three terms with the same abbreviation). Such definitions should be maintained carefully because name validation (and name translation) will return correct results only if all terms have the same classification settings. If this option is enabled, name validation will report non-unique abbreviations (alternate ones are also included) and words without abbreviations.

**Separator and Sep. char.:** Define a word separator for multi-word terms. Note that separator settings are checked when a glossary is loaded into the glossary editor; and if the separator is not a space character, a warning is displayed, and you can change the separator to a space.

**Filter:** Lets you display all glossary entries or restrict the display to entries that include a specified classification. ("Unclassified" shows only entries that have no classification.)

**Words:** To add a glossary entry, click the Add (+) icon and complete the information on the new line for the entry; to delete a glossary entry, select its entry and click the Remove (X) icon; to edit an entry, select its entry and modify the information.

**Name (of entry):** Glossary term associated with this entry.

**Abbreviation:** Abbreviation for the glossary term.

**Alt Abbreviation:** Alternative abbreviation for the glossary term.

**Prime, Class, Modifier, Qualifier:** Specify as many classification types (prime word, class word, modifier, qualifier) as apply to the glossary entry. Name patterns can be defined for entities, attributes, tables, columns, and domains. A name pattern defines the structure of the name in the form of an unrestricted sequence of permitted word types and their optionality.

**Prime:** The prime word identifies the object or element being defined. Typically, these objects represent a person, place, thing, or event about which an organization wishes to maintain information. Prime words may act as primary search identifiers when querying a database system and provide a basic list of keywords for developing a general-to-specific classification scheme based on business usages. CUSTOMER in Customer Address is an example of a prime word.

**Class:** A class word is the most important noun in a data element name. Class words identify the use or purpose of a data element. Class words designate the type of information maintained about the object (prime word) of the data element name. ADDRESS in Customer Address is an example of a class word

**Modifier:** A modifier gives additional information about the class word or prime word. Modifiers may be adjectives or nouns. DELIVERY in Customer Delivery Address is an example of a modifier. Other modifier examples: ANNUAL, QUARTERLY, MOST, LEAST

**Qualifier:** A qualifier is a special kind of modifier that is used with a class word to further describe a characteristic of the class word within a domain of values, or to specify a type of information which can be attached to an object. Examples: FEET, METERS, SECONDS, WEEKS

**Short Description:** Short description of the entry.

**Import:** Lets you specify a glossary file and import its entries.

**Export:** Lets you export the glossary to a comma-separated values (CSV) file.

**Save:** Saves the current entries into the currently open glossary file.

**Save As:** Saves the current entries in a glossary file that you specify.

## 3.40 Hierarchy Properties

This dialog box is displayed when you view the properties of a hierarchy in a multidimensional model.

A hierarchy is a way to organize data at different levels of aggregation. Hierarchies are used to define data aggregation; for example, in a Time dimension, a hierarchy might be used to aggregate data from days to months to quarters to years. Hierarchies are also used to define a navigational drill path. In a relational table, hierarchies can be defined as part of a dimension object.

For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the hierarchy.

**Value Based Hierarchy:** Controls whether this is a value-based hierarchy, that is, a hierarchy defined only by the parent-child relationships among dimension members. The dimension members at a particular distance from the base level do not form a meaningful group for analysis, so the levels are not named.

**Time Based Hierarchy:** Controls whether this is a time-based hierarchy, that is, a hierarchy composed of time-related levels such as Month, Quarter, and Year.

**Ragged Hierarchy:** Controls whether this is a ragged hierarchy, that is, a hierarchy in which leaf nodes can be located at different levels.

**Default Hierarchy:** Controls whether this is the default hierarchy for the dimension.

#### **Levels**

Lists levels associated with the hierarchy. To view or edit a level definition, double-click its item. (See also [Level Properties](#).)

#### **Rollup Links**

Lists rollup links associated with the hierarchy. To view or edit a rollup link definition, double-click its item. (See also [Rollup Link Properties](#).)

#### **Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Myhierarchies* if the Short Name is *Myhierarchy*

#### **Description**

Description of the hierarchy.

#### **Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.41 Import Domains**

This dialog box is displayed if you click File, then Import, then Domains. You can import domains from an XML file, for example, to use domains that you created for another design.

**Open Domain File:** Displays a dialog box for specifying the XML file containing domain definitions to be imported.

**Import in Default Domains:** Controls whether the imported domains will be saved in a file named `defaultdomains.xml` and placed in the domains directory (folder) under the location where you installed Data Modeler.

**Domains in File:** Displays the domains in the domain file that you opened.

**Filter:** Lets you control whether all domains are displayed, or only new or modified domains.

**Corresponding Domains in Design <name>:** Displays any corresponding domains in the current design.

## **3.42 Import Glossary (Naming Standard Definitions)**

This dialog box is displayed if you click Import in the [Glossary Editor](#) and specify a glossary file with glossary terms (naming standard definitions) to be imported. You can import definitions from an external glossary and then modify the details.

The possible words to be imported and the corresponding words in the current glossary are displayed. You can select and deselect words to import.

Separator settings are checked when the terms from the external glossary are loaded into the glossary editor. If the separator is not the space character, a report showing all terms with a space in the name is displayed, and you are given the option to replace the space with a defined glossary separator.

### 3.43 Import Mapped Models from VAR Files

This dialog box is displayed if you specify to import mapped models in the [Import VAR File: Select Type of Import](#) dialog box.

**Logical Model:** Select the file containing the logical model.

**Relational Models:** Lists the relational models to be included. To add a model, click **Add**; to remove a model, click its name and click **Remove**.

### 3.44 Import Oracle Designer Model

This wizard enables you to import an Oracle Designer design into Data Modeler.

#### Connect to Database

Specify information to connect to the database with the Oracle Designer design. Select an existing connection, or click **Add** to specify information for a new connection.

To test a connection, select it and click **Test Connection**.

To move to the next wizard page, select a connection and click **Next**.

Follow the instructions on the remaining wizard pages to select the work area, select application systems, select objects to import, and generate the design. Click **Next** to move to a next page, and click **Finish** on the last page to perform the import operation.

### 3.45 Import SQL Developer Data Modeler Design

This wizard is displayed if you click **File**, then **Import**, then **Data Modeler Design**, and then specify an XML file containing a SQL Developer Data Modeler design.

#### Select Models to Import

Lists the logical model and any relational models in the specified file, and lets you select which models (individually or all) to include in the import operation.

#### Generate Design

Identifies the number of logical and physical designs that will be generated. To continue the import operation, click **Finish**, which displays the [Compare Modeling Designs](#) dialog box.

### 3.46 Import SQL Developer Database Connections

This dialog box is displayed if you click **Import** to import database connections in the [Data Dictionary Import \(Metadata Extraction\)](#) wizard.

**File Name:** Click **Browse** to find the XML file containing database connection definitions that have been exported from SQL Developer.

**Connections:** Displays the names of database connections in the specified file.

Click **OK** to import the database connections.

### 3.47 Import VAR File: Select Type of Import

This dialog box is displayed when you click File, then Import, then VAR File. You can import model definitions from a VAR file.

**Import Single Model:** Imports only a single model describing a logical or relational model.

**Import Mapped Models:** Imports a mapped model describing a logical model and a relational model, as well as the mappings between them.

**Import Process Model:** Causes the process model to be included in the import operation.

**Select VAR Type:** Type of VAR file: Sterling COOL:DBA V2.1, Sterling Bnsteam V7.2, or Cayenne Bnsteam V7.2.

### 3.48 Index Properties

This dialog box displays the properties of an index, which is an object defined on one or more table columns in [Relational Models](#).

#### General

**Name:** Name of the index.

**Long Name:** Long name for display purposes.

**Engineer:** Controls whether the index will be considered during reverse engineering operations. If this option is disabled, this index and its properties are not considered when the relational model is reverse engineered into the logical model

**Table:** Table containing the column or columns to be indexed.

**State:** State or purpose of the index: Plain Index, Primary Constraint, Unique Constraint, or Unique Plain Index.

**Sort Order Columns:** Displays the column names and their sort order in the index.

**Index Expression:** Indicates whether this is a function-based index, which is an index based on an expression.

**Spatial Index:** Indicates whether this is a spatial index (with INDEXTYPE of MDSYS.SPATIAL\_INDEX).

**Generate in DDL:** Controls whether the index creation is included when DDL statements are generated to be used to create the database.

#### Columns

Displays columns that are available to be added to the index definition on the left, and columns that are included in the index definition on the right. You can select columns and use the arrow keys to move them from one side to the other.

#### Functional

Specification of the expression for a function-based index (if Index Expression is enabled under General properties).

**Spatial Properties**

Spatial properties for a spatial index.

**Constrain to Layer Type:** Constrains the index to a specified type of layer.

**Geodetic Index:** Indicates whether the index is on geodetic or projected data.

**Number of Dimensions:** Number of spatial dimensions to be indexed. Example: 2 for longitude and latitude

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.49 Information Store Properties

This dialog box lets you view and modify information for an information store object, which is part of the [Process Model](#).

**General**

**Name:** Name of the information store.

**Synonym:** Synonym for the information store.

**Type:** Where the data for the information store is stored: RDBMS for a relational database, File for an external file, Object for an object, or Temporary for temporary storage.

**RDBMS Type:** If Type is RDBMS, the database type (physical model) for the information store.

**Object Type:** If Type is Object, the object type.

**Implementation Name:** If Type is Object, the implementation name.

**Scope:** If Type is Temporary, the scope of the data: when the data is available (for example, Session or Application).

**Attributes**

Lists the attributes associated with this information store. To view the properties of any listed attribute, double-click its entry.

**Entities**

Lists each entity and corresponding table associated with this information store. To view the properties of any listed entity, double-click its entry

**Processes**

Lists the processes associated with this information store. To view the properties of any listed process, double-click its entry.

**Information Flows**

Lists the information flows associated with this information store. To view the properties of any listed information flow, double-click its entry.

**System Objective**

Description of the system objective for this information store.

**Data File Specification**

**Owner:** Owner of the information store.

**Source:** Source of the information store: where the information for the information store comes from.

**File Name:** Name of the information store file.

**Location:** Location of the information store file.

**File Type:** Type of file: CSV (comma-separated values), fixed-length fields, Excel spreadsheet, or plain text.

**Field Separator:** Character (for example, the comma) that separates fields.

**Data Capture Type:**

**Self Describing:** Indicates whether the field is self-describing.

**Skip Records:** Number of records to be skipped at the top of the data file.

**Text Delimiter:** Text delimiter character that is used in text strings (single or double quotes).

**Data Elements**

Lists the external data elements currently defined for the external agent. To add an element, click the Add (+) icon; to delete an element, select it and click the Remove (X) icon; to view the properties of an element, double-click it or select it and click the Properties icon.

**External Data:** Name of the external data object.

**Type:** Logical type of the external data.

**Starting Position:** Starting position of the element in the data file.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.50 Information Structure Properties

This dialog box lets you view and modify information for an information structure object, which is part of the [Process Model](#).

**General**

**Name:** Name of the information structure.

**Synonym:** Synonym for the information structure.



**Attributes**

Lists the attributes associated with this information structure. To view the properties of any listed attribute, double-click its entry.

**Entities**

Lists each entity and corresponding table associated with this information structure. To view the properties of any listed entity, double-click its entry.

**Information Flows**

Lists the information flows associated with this information structure. To view the properties of any listed information flow, double-click its entry.

**Volume Information**

**Volume:** Initial or current data volume for the information structure.

**Growth Rate: Percent:** Percent of growth expected in a time period specified in the next field. For example, if you expect growth of 5 percent each month, enter 5 in this field and select Month for the next field.

**Growth Rate: Year/Month/Day:** Time period associated with the expected growth rate.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.51 Join Properties

This dialog box is displayed when you view the properties of a join in a cube in a multidimensional model. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

**General**

**Name:** Name of the join object.

**Left Entity:** Left entity in the join operation.

**Right Entity:** Right entity in the join operation.

**Existing Relation:**

**Cardinality:** Cardinality relationship between the left and right entities: 1:1 (one to one), 1:\* (one to many), or \*:1 (many to one).

**Dominant Role:** Entity with the dominant role.

**Attribute Pairs**

Lists the left and right entity for each attribute pair. You can add and remove pairs. For each pair, you can specify the left and right entities.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.52 Unique Identifier (UID, or Key) Properties

This dialog box displays the properties of a candidate key, which is an object defined in the [Logical Model](#).

### General

**Name:** Name of the key.

**Synonym:** Synonym for the key.

**Long Name:** Long name for display purposes.

**State:** State or purpose of the key: Primary Key or Unique Key.

### Attributes and Relations

Displays attributes and relations that are available to be added to the key definition on the left, and attributes and relations that are included in the index definition on the right. You can select attributes and relations and use the arrow keys to move them from one side to the other.

### Engineer To

Enables you to specify the relational models to which this key should be propagated in forward engineering operations.

**Engineer:** Controls whether the key is propagated to the specified **Relational Design** (model) during forward engineering operations.

### Impact Analysis

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

### Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.53 Level Properties

This dialog box is displayed when you view the properties of a level in a multidimensional model.

For business analysis, data is typically summarized by level. For example, the database may contain daily snapshots of a transactional database. Days are the base level. You might summarize this data at the weekly, quarterly, and yearly levels. Thus, levels provide a convenient way of identifying the dimension members at a particular distance from the detail data.

For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the level. Examples: customer, product

**Level Entity:** Entity associated with this level. Example: *customers* in *customers.customer*

**Value Based Hierarchy:** Controls whether the hierarchy is value-based or level-based. (the latter being more common). If this option is enabled, a value-based hierarchy is

used, in which the parent-child relations do not have named levels. If this option is disabled, a value-based hierarchy is used.

**Root Identification:** For a value-based hierarchy, specify one of the following: Parent is Null, Root has Value, or Parent has Value.

**Default Attribute:** Default descriptive attribute (if required).

### **Selection Criteria**

Selection criteria for this level.

### **Selection Criteria Description**

Description of the selection criteria for this level.

### **Level Key**

Displays any attributes that are keys for the level. To add a level key object, click the Add (+) icon; to remove a level key object from the level definition, select it and click the Remove (X) icon. To edit a level key object, double-click its item, or click its item and click the Properties icon.

### **Descriptive Attributes**

**Name:** Name of the attribute.

**Attribute:** Fully qualified name of the attribute.

**Indexed:** Controls whether the attribute is indexed.

**Slow Changing:** Indicates whether this is a Slowly Changing Attribute; and if so, what type. None means it is not a Slowly Changing Attribute; and Type 1, Type 2, or Type 3 means it is a Slowly Changing Attribute of the specified type.

### **Parent Key**

For a level with a value-based hierarchy, lists the attributes of the parent entity.

### **Calculated Attributes**

Lets you add calculated attributes to the level definition and remove them from the definition. For each calculated attribute, you can specify the associated expression.

### **Oracle AW Attributes**

Displays attributes of the level for Oracle Analytic Workspaces.

### **Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Mylevels* if the Short Name is *Mylevel*

### **MS Olap**

Lets you specify the name and value columns for use with Microsoft OLAP.

### **Description**

Description of the level.

**Partitioning Description**

Description of the partitioning for the level.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.54 Location Properties

This dialog box displays the properties of a location object, which is a type of [Business Information](#) object.

**General**

**Name:** Name of the location object.

**Location Type:** Descriptive phrase indicating the type of location object, using any naming scheme suited to your needs.

**Address:** Street address, in a format appropriate for the locale.

**City:** Name of the city or town.

**Post Code:** Postal code, in a format appropriate for the area or country.

**Area:** Part of the country, such as the two-character state abbreviation for a United States address.

**Country:** Country code or name.

**Contacts**

Lists the contact objects currently associated with this email object. To view the properties of any listed contact object, double-click its entry.

**Comments, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.55 Logical Type

This dialog box is used to specify a logical data type for a user-defined attribute.

**Logical Type:** Name of the logical type.

**Size:** Maximum size of the data for this type.

**Precision:** For a numeric type, the maximum number of significant decimal digits.

**Scale:** For a numeric type, the number of digits from the decimal point to the least significant digit.

## 3.56 Measure Folder Properties

Measure folders in the multidimensional model group measures together so that they can be identified and accessed easily. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

**General**

**Name:** Name of the measure folder object.

**Measures**

Lists any relevant measures. To add a measure, click the Add (+) icon; to remove a measure from the measure folder definition, select it and click the Remove (X) icon. (See also [Measure Properties](#).)

**Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Myfolders* if the Short Name is *Myfolder*

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.57 Measure Properties

This dialog box displays the properties of a measure in a multidimensional model.

Measures store the facts collected about your business, such as Costs or Units Sold. Each measure belongs to a particular cube, and thus shares particular characteristics with other measures in the cube, such as the same dimensions. The default characteristics of a measure are inherited from the cube.

For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

**General**

**Name:** Name of the measure. Example: Sales

**Is Formula:** Controls whether the measure is defined by the formula specified in the Formula field. If you do not specify a formula, the function specified in the Aggregation Function field is used.

**Custom Formula:** Identifies whether the formula is a custom formula.

**Formula:** Formula if this is a formula-based measure.

**Formula Type:** Type of formula: none, Base, OLAP, or Microsoft-computed.

**Based On Fact:** Name of the fact associated with the measure.

**Additivity:** Fully-Additive (additive across all dimensions), Semi-Additive (additive across some dimensions), or Non-Additive.

**Aggregation Function:** Function to be used for aggregation.

**Where Clause:** WHERE clause limiting the aggregation.

**Aggregation Functions**

**Functions:** Lists the aggregation functions and measure aliases. To add an aggregation function, click the Add (+) icon; to remove an aggregation function from the measure property definition, select it and click the Remove (X) icon. To set summary levels for a measure alias, select its item and click the Set Oracle AW Presummarized Levels icon.

**Oracle OLAP Measure**

For a measure based on an Oracle OLAP formula, specifies the OLAP operator and other information.

**Oracle Names**

**Short Name:** Short descriptive name that can be used by applications instead of the long name. Example: Sales

**Long Name:** Long descriptive name, typically used for display. Example: Unit Sales

**Plural Name:** Plural name (for reporting purposes). For example: *Mymeasures* if the Short Name is *Mymmeasure*

**Description**

Description of the measure. Example: Unit sales measure

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.58 Measurement Properties

Measurements allow metrics to be defined for model objects, indicating the value of some object attribute. For example, they may be used to hold actual, estimated, or planned values for the size of a table, the number of rows in a table, the number of pages in an index, the number of different values in a column, and so on. This flexibility allows for product or project extensions, without changing the model.

**Name:** Name of the measurement object.

**Comment:** Optional descriptive comment text.

**Notes:** Optional note text, such as background information or implementation notes.

**Value:** Numeric value associated with the measure. Example: 10 if the measure is for 10 meters

**Unit:** Unit of measurement associated with the value. Example: Meter

**Type:** Type of value: Measure (measured), Estimate (estimated), Plan (planned), Minimum, Maximum, or Average.

**Creation Date:** The date when the measurement object was established.

**Effective Date:** The date when the measurement object is effective. For measured values, the effective and creation dates should be the same. For estimated actual values, the creation date may be later than the effective date. For planned values, the effective date is normally later than the creation date.

**Summary:** Displays read-only summary information.

## 3.59 Method Properties

This dialog box defines the properties of a method used to implement a structured data type.

**General**

**Name:** Name of the method.

**Constructor:** Controls whether a constructor is created for the method.

**Overridden Method:** The method that this method overrides (if any).

**Return Value:** NO RETURN if there is no return value, or the data type of the returned value.

#### **Parameters**

Lists the name and data type of each parameter for the method.

To add a parameter, click the Add (+) icon; to delete a parameter select it and click the Remove (X) icon; to move a parameter up or down in the list, select it and click the appropriate arrow.

#### **Body**

Code that implements the method.

#### **Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

### 3.60 Model Properties - Business Information

This dialog box displays the properties of the [Business Information](#) model.

#### **General**

**Name:** Name of the business information model.

**Visible:** (Does not apply to this model.)

#### **Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

### 3.61 Model Properties - <data-flow-diagram-name>

This dialog box displays the properties of [Data Flow Diagrams](#), which are part of the [Process Model](#).

#### **General**

**Name:** Name of the data flow diagram.

**Visible:** Controls whether the data flow diagram is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the data flow diagram name in the object browser.

#### **Process Order/Number**

Displays any processes associated with the data flow diagram. To view or edit the [Process Properties](#), select it and click the Properties icon. To move a process up or down in the order within the data flow, select it and use the arrow icons.

#### **Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

### 3.62 Model Properties - Data Types

This dialog box displays the properties of the [Data Types Model](#).

### **General**

**Name:** Name of the data types model.

**Visible:** Controls whether the data types model diagram is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the data types model in the object browser.

### **Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.63 Model Properties - Logical**

This dialog box displays the properties of the [Logical Model](#).

### **General**

**Name:** Name of the logical model.

**Visible:** Controls whether the diagram for the logical model is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the logical model name in the object browser.

### **Naming Options**

You can specify the following naming rules for entities, attributes, and views:

**Max Name Length:** Maximum number of characters in the name.

**Character Case:** Controls whether you can use only uppercase or lowercase characters, or both uppercase and lowercase (that is, mixed case).

**Valid Characters:** Specify either **All Valid** (no restrictions), or disable All Valid and then select the valid set of characters.

### **Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.64 Model Properties - <multidimensional-model-name>**

This dialog box displays the properties of a multidimensional model.

### **General**

**Name:** Name of the multidimensional model.

**Visible:** Controls whether the multidimensional model diagram is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the multidimensional model name in the object browser.

### **Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.65 Model Properties - Process Model**

This dialog box displays the properties of the [Process Model](#).



**General**

**Name:** Name of the process model.

**Visible:** Controls whether the diagram for the process model is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the process model name in the object browser.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.66 Model Properties - <name> (Relational)

This dialog box displays the properties of [Relational Models](#).

**General**

**Name:** Name of the relational model.

**Visible:** Controls whether the diagram for this relational model is displayed in the Data Modeler window. You can also control the visibility by selecting Show or Hide on the context menu after you right-click the relational model name in the object browser.

**RDBMS Type:** Type of database.

**RDBMS Site:** RDBMS site. You can select a site of the specified RDBMS Type. (RDBMS sites are explained in [Section 1.3.6, "Physical Models"](#).)

**Naming Options**

You can specify the following naming rules for tables, columns, and views:

**Max Name Length:** Maximum number of characters in the name.

**Character Case:** Controls whether you can use only uppercase or lowercase characters, or both uppercase and lowercase (that is, mixed case).

**Valid Characters:** Specify either **All Valid** (no restrictions), or disable All Valid and then select the valid set of characters.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.67 Name Abbreviations

This dialog box is displayed after you click Tools, then Name Abbreviations. You must specify a .csv (comma-separated values) file containing pairs in the following format: *string-to-use, string-to-be-changed* (with or without a space after the comma). In each pair, the second term is the spelling that is to be changed to the first term in names or parts of names (separated by underscores or spaces) in relational models of the current design after you click **OK**.

As an example, the following lines in a .csv file cause the strings CUS and CUSTOMER to be changed to CUST:

```
CUST, CUSTOMER
CUST, CUS
```

For example, a column named `customer_id` would be changed to `cust_id`, and a column named `cus_last_name` would be changed to `cust_last_name`. (Thus, the resulting names are not necessarily "abbreviated"; they are just made consistent.)

**Select a .csv File:** Find the .csv file containing the comma-separated value pairs.

**All Objects:** Controls whether the name changes are applied to tables, columns, indexes, and views, or only to the types of objects that you specify.

**Keep Letter Case:** Controls whether the letter case of the current name is retained when the name string is changed. If this option is disabled, any spelling (case) of the name to be changed uses the case as specified in the pair in the .csv file. For example, if this option is disabled and if the string pair is `CUST , CUSTOMER`, a name of `customer_first_name` would be changed to `CUST_first_name`.

Note: Do not confuse "Name abbreviations" with naming standardization, which is wider in scope and is integrated with the checking of [Design Rules](#). Naming standardization is implemented through Data Modeler user options, as explained in [Section 1.5.4, "Naming Standard"](#).

## 3.68 New/Update Database Connection

This dialog box is displayed when you attempt to add or edit a database connection for use in an import or export operation. For import operations from a data dictionary, you can connect to an Oracle database or a supported third-party database. For export operations, you can connect to an Oracle Database, for use as the Data Modeler *reporting repository* (which is explained in [Section 1.7, "Data Modeler Reports"](#)).

When you have finished entering the connection information, test the connection, as explained in [After Specifying the Connection Information](#).

Some of the following fields apply only to certain kinds of database connections.

**Connection Name:** An alias for a connection to the database using the information that you enter. (The connection name is not stored in the database, and the connection is not a database object.) Suggestion: Include the database name (SID) and user name in the connection name. Example: `personnel_herman` for connecting to the personnel database as user Herman.

**User Name:** Name of the database user for the connection. This user must have sufficient privileges to perform the tasks that you want perform while connected to the database, such as creating, editing, and deleting tables, views, and other objects.

**Password:** Password associated with the specified database user.

**Save Password:** If this option is checked, the password is saved with the connection information, and you will not be prompted for the password on subsequent attempts to connect using this connection.

The remaining fields are grouped under tabs according to the database type. See [Oracle tab](#) or [Other databases \(third-party\) tabs](#).

### Oracle tab

The following information applies to a connection to an Oracle database.

**Role:** The set of privileges to be associated with the connection. For a user that has been granted the SYSDBA system privilege, you can specify a connection that includes the privilege.

**Connection Type:** Select Basic, TNS, or Advanced. (The display of fields changes to reflect any change in connection type.)

**Basic connection type**

**Host Name:** Host system for the Oracle database.

**Port:** Listener port.

**SID:** Database name.

**Service Name:** Network service name of the database (for a remote database connection over a secure connection).

**TNS connection type**

**Network Alias:** Oracle Net alias for the database. (The list for selecting a network alias is initially filled from the tnsnames.ora file on your system, if that file exists.)

**Connect Identifier:** Oracle Net connect identifier.

**Advanced connection type**

**Custom JDBC URL:** URL for connecting directly from Java to the database; overrides any other connection type specification. If you are using TNS or a naming service with the OCI driver, you must specify this information: Example:

```
jdbc:oracle:thin:scott/@localhost:1521:orcl
```

Note that in this example, the "/" is required, and the user will be prompted to enter the password.

To use a custom JDBC URL, the system on which SQL Developer is running must have an Oracle Client installation that contains the JDBC and ora18n libraries, is present on the path, and is version 10.2 or later.

**Other databases (third-party) tabs**

The following information applies to a connection to a third-party (non-Oracle) database.

**Host Name:** Host system for the database.

**Port:** Listener port.

**Database:** Database name.

**JDBC-ODBC Bridge or Other Third Party Driver (JDBC ODBC Bridge tab):** Indicates a JDBC to ODBC bridge driver or another third-party driver.

**ODBC Alias (JDBC-ODBC Bridge):** Name of an existing ODBC data source.

**JDBC URL (Other Third Party Driver):** URL for connecting directly from Java to the database; overrides any other connection type specification.

**Driver Class (Other Third Party Driver):** The name of the driver class that will be used for the connection (for example, `com.microsoft.jdbc.sqlserver.SQLServerDriver`). This name can be found in the JDBC driver specification (usually shipped with the driver).

**After Specifying the Connection Information**

To test the connection using the specified information, click **Test Connection**. A message is displayed indicating the result of the test.

To add the new connection or to complete any edits to an existing connection, click **OK**.

## 3.69 Process Properties

This dialog box displays the properties of a process object, which is part of the [Process Model](#).

### General

**Name:** Name of the process.

**Synonym:** Synonym for the process.

**Sources Filter Condition:** A filter for the data on the source level, using SQL valid in a WHERE clause (but without including the WHERE keyword). Example: `dept_id = 'ENG'`

**Sources Join Condition:** A join between two or more sources used in the process, using SQL valid in a WHERE clause (but without including the WHERE keyword). Example: `table1.dept_id = table2.dept_id`

**Type:** Type of process: Primitive (standalone), Composite (consisting of multiple outer processes, or Use transformation task (as specified in the next field).

**Use Transformation Task:** Select the transformation task to use.

### Mode

**Interactive/Batch: Mode:** Batch, Interactive, Manual, or Unknown.

**Minimum Acceptable Throughput: Batch Min. Transactions:** For batch mode, minimum acceptable number of processing operations for each specified Minimum Acceptable Throughput Batch Time Unit. Example: 100 for 100 operations each hour.

**Minimum Acceptable Throughput: Batch Time Unit:** Unit of time for the specified minimum acceptable number of processing operations. Example: Hour for 100 operations each hour.

**Longest Acceptable Response Time:** Numeric unit of the longest acceptable response time. Example: 5 for 5 seconds.

**Longest Acceptable Response Time Unit:** Time measure unit of the longest acceptable response time. Example: Second for 5 seconds.

### Frequency/Priority

**Expected Frequency Times:** Number of times the process is expected to be used within each Expected Frequency Time Unit. Example: 50 for 50 times each day.

**Expected Frequency Time Unit:** Time measure unit for the Expected Frequency Times. Example: Day for 50 times each day.

**Priority:** Descriptive term for the priority of the process: Low, Medium, High, or None.

### Peak Periods

You can specify, for each day the process is run, either no peak periods (times with a high level or activity or demand) or one or more one-hour intervals.

### Information Structures

Displays information structures associated with the process. To view the [Information Structure Properties](#), double-click an information structure item or select the item and click the Properties icon.

### Events

Displays events associated with the process. Includes a separate area for flow events. To view the [Event Properties](#), double-click an information structure item or select the item and click the Properties icon. To add an event, click the Add (+) icon; to remove an event from the process definition, select it and click the Remove (X) icon.

### Incoming Flows

Displays flows coming into the process. To view the [Flow Properties](#), double-click the flow item or select the item and click the Properties icon. (To add a flow, use the New Flow icon on the data flow diagram.)

### Outgoing Flows

Displays flows going out from the process. To view the [Flow Properties](#), double-click the flow item or select the item and click the Properties icon. (To add a flow, use the New Flow icon on the data flow diagram.)

### Processed Attributes

Displays attributes processed by the process. To view the [Attribute Properties](#), double-click an attribute item or select the item and click the Properties icon.

### Processed Entities

Displays entities processed by the process. To view the [Entity Properties](#), double-click an entity item or select the item and click the Properties icon.

### Footnote

Text for a footnote about the process.

### Task Input Params Mapping

Displays mappings between task input parameters and their sources.

### Source-Target Mapping

Lets you specify, for each target element, the type of transformation: As it is, Derivation, Aggregation, Summarization, or Complex Formula. (Each Target Element matches an item from the [Processed Attributes](#) pane.)

### Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.70 RDBMS Site Editor

This dialog box is displayed when you click Tools, then RDBMS Site Administration. It lets you view RDBMS sites, add sites, and edit and remove sites that you added. You cannot remove or change the properties of any predefined sites. (For an explanation of RDBMS sites, see [Section 1.3.6, "Physical Models"](#).)

### Current Design

Lists currently defined RDBMS sites. If you select a site, its name and RDBMS type are displayed. To add a user-defined site, click **Add**, specify the name and RDBMS type, and click **Apply**; to remove a user-defined site, click its entry under RDBMS Sites and click **Remove**.

**External RDBMS File**

Lets you specify an external XML file containing RDBMS site definitions, after which its sites are displayed. If you select a site, its name and RDBMS type are displayed. To add a site, click **Add**, specify the name and RDBMS type, and click **Apply**; to remove a site, click its entry under RDBMS Sites and click **Remove**. To save in the XML file any changes that you have made in this dialog box, click **Save**.

## 3.71 Record Structure Properties

This dialog box displays the properties of a record structure.

**General**

**Name:** Name of the record structure.

**Data Elements**

Displays external data objects associated with the record structure. To view the external data object properties, double-click an item or select the item and click the Properties icon. To add an external data object, click the Add (+) icon; to remove an external data object from the record structure definition, select it and click the Remove (X) icon.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.72 Relation Properties

This dialog box displays the properties of a relation, which is part of the [Logical Model](#).

**General**

**Name:** Name of the relation.

**Long Name:** Long name for display purposes.

**Cardinality**

**Source:** Source entity for the relation.

**Name on Source:** Text describing the role of the source entity in the relation (for example, "has"). This text is displayed in the logical model diagram if you enable the Show Source/Target Name option for the Logical Model diagram in the [Diagram](#) user options.

**Source Entity Synonym:** Synonym for the source entity.

**Source to Target Cardinality:** Cardinality of source records to target records with the same key value: \* for many, or 1 for one.

**Source Optional:** Controls whether the source entity in the relation must contain one or more instances. If this option is enabled, there can be zero source instances; if this option is disabled, one or more source instances are required.

**Target:** Target entity for the relation

**Name on Target:** Text describing the role of the target entity in the relation (for example, "any number of"). This text is displayed in the logical model diagram if you

enable the Show Source/Target Name option for the Logical Model diagram in the [Diagram](#) user options.

**Target Entity Synonym:** Synonym for the target entity.

**Target to Source Cardinality:** Cardinality of target records to source records with the same key value: \* for many, or 1 for one

**Target Optional:** Controls whether the target entity in the relation must contain one or more instances. If this option is enabled, there can be zero target instances; if this option is disabled, one or more target instances are required.

**Dominant Role:** Entity with the dominant role.

**Identifying:** Controls whether this is an identifying relationship.

### **Engineer To**

Enables you to specify the relational models to which this relation should be propagated in forward engineering operations.

**Engineer:** Controls whether the relation is propagated to the specified **Relational Design** (model) during forward engineering operations.

### **Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

### **Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

### **Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## **3.73 Relational Models**

The Relational Models dialog box identifies two relational models to be compared and merged. This box is displayed if you click Tools, then Compare/Merge Models.

**From:** Source relational model to be merged into the destination model.

**To:** Destination relational model into which objects from the source relational model are to be merged.

**Database Type:** Select the Oracle physical model whose data types are to be used.

**View Compare Mapping:** Displays the [Compare Mapping](#) dialog box.

When you click OK, the [Compare Models](#) dialog box is displayed, in which you can apply a filter to restrict the types of objects and specific objects to be merged.

## **3.74 Resource Locator Properties**

This dialog box displays the properties of a resource locator object, which is a type of [Business Information](#) object.

### General

**Name:** Name of the resource locator object.

**URL:** Uniform Resource Locator, if the resource locator is a Web address.

### Contacts

A resource locator object can have multiple contact objects associated with it, with the contact objects in an order that can indicate the level of responsibility for the resource. For example, if two contacts are listed for a URL, the first one listed might be the primary webmaster.

### Comments, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.75 Responsible Party Properties

This dialog box displays the properties of a responsible party object, which is a type of [Business Information](#) object.

### General

**Name:** Name of the responsible party object. Examples: a person's name, a role (such as Project Leader), or a department (such as Quality Assurance).

**Responsibility:** Brief description of the nature of the responsibility. Example: "Overall project management"

### Contacts

A responsible party object can have multiple contact objects associated with it, with the contact objects in an order that can indicate the sequence in which to attempt to contact the party. For example, if the office, mobile, and home contacts are listed in that order, that may mean that the responsible party should be called first at the office, then on his or her mobile phone, and finally at home.

### Comments, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.76 Role Properties

This dialog box displays the properties of a role object, which is part of the [Process Model](#).

### General

**Name:** Name of the role.

**Synonym:** Synonym for the role.

**Description:** Description of the role.

### Processes

Lets you add, delete, and edit processes associated with the role.

### Entities

Lets you view and modify entities associated with the role and operations permitted on that entity (create, read, update, delete) by the role.



### Comments in RDBMS

Comment text to be included in database objects that are generated based on this modeling object.

### Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.77 Rollup Link Properties

This dialog box displays the properties of a rollup link in a multidimensional model. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

## 3.78 SELECT DDL Files

This dialog box is displayed if you click File, then Import, then DDL File to import definitions from one or more data definition language (DDL) files into a relational model. To add a file to the list to be imported, click the Add DDL Files (+) icon; to remove a file from the list, select it and click the Remove DDL Files (X) icon.

### Options

**Import to:** Name of the relational model into which to import the definitions in the specified file or files.

**Swap Target Model:** If this option is enabled, a script is generated in which your current relational model is merged into the specified database (DDL files). If this option is not enabled, the definitions in the imported files are merged into your relational model.

**Skip Merge Dialog:** If this option is enabled, the [Relational Models](#) dialog box is not displayed before the import operation occurs.

## 3.79 Set Classification Types

This dialog box is used to specify, for entities or tables in a multidimensional model, the classification type for each: Fact, Dimension, Logging, Summary, or Temporary.

You can specify colors to be used in diagrams for each classification type in the [Diagram: Classification Types](#) user preference.

## 3.80 Set Data Type

This dialog box is used to specify the data type for a user-defined attribute. Select the category for this type (**Logical**, **Distinct**, **Collection**, or **Structured**), then select the specific type within that category.

If you specify **Logical Type** and click the displayed type, the [Logical Type](#) dialog box is displayed.

**Reference:** For a structured type, controls whether the type is created as a REF (reference). A REF is a logical pointer to a row object that is constructed from the object identifier (OID) of the referenced object and is an Oracle built-in data type. REFs and collections of REFs model associations among objects, particularly many-to-one relationships, thus reducing the need for foreign keys.

## 3.81 Slice Properties

This dialog box displays the properties of a slice of a cube in a multidimensional model. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the slice.

**Read Only:** Controls whether the data in the slice is read-only.

### Measures

Calculated measures can add information-rich data to a cube. The data set is calculated on the fly, so no data is stored. You can add as many calculated measures as you like without increasing the size of the database.

### Dimensions, Levels

Lists any dimension/hierarchy/level combinations for the slice. To add a level, click the Add (+) icon; to remove a level from the slice definition, select it and click the Remove (X) icon. (See also [Level Properties](#).)

### Selected Attributes

Lists any dimensions for the slice. To add a dimension, click the Add (+) icon; to remove a dimension from the slice definition, select it and click the Remove (X) icon. (See also [Dimension Properties](#).)

### Slice to Entity Mappings

Lists any attributes and their mappings for a specified entity.

### Selection Criteria

Lets you specify one or more selection criteria (name, WHERE clause, and other details for each).

### Oracle Names

**Short Name:** Short descriptive name that can be used by applications instead of the long name.

**Long Name:** Long descriptive name, typically used for display.

**Plural Name:** Plural name (for reporting purposes). For example: *Myslices* if the Short Name is *Myslice*

### SQL Access to Oracle AW

Lists any relevant SQL Access to Oracle Analytic Workspaces (AW) objects. To add a SQL Access to Oracle AW object, click the Add (+) icon; to remove a SQL Access to Oracle AW object from the slice definition, select it and click the Remove (X) icon. To edit a SQL Access to Oracle AW object, double-click its item, or click its item and click the Properties icon. (See also [SQL Access to Oracle AW Properties](#).)

### Description

Description of the slice.

### Comments, Notes, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.82 Spatial Definition Properties

Displays the properties of the spatial column or columns in a spatial table. The properties correspond to attributes and options for the SDO\_GEOMETRY type, which is documented in *Oracle Spatial Developer's Guide*.

**Name:** Name of the specify definition object.

**Spatial Column:** Name of the spatial column.

**Use Function:** Identifies whether a function is used to compute the column value.

**Function Expression:** Function expression if Use Function is enabled.

**Coordinate System ID:** Oracle Spatial SRID value. Example: 8307 for the WGS 84 longitude/latitude system that uses the Greenwich prime meridian.

**Create Spatial Index:** Controls whether a spatial index is created.

**Spatial Index Name:** Name for the spatial index.

**Dimensional Information:** For each dimension, specify the name, lower and upper boundaries, and the tolerance. Example: Name = longitude, Low boundary = -180, Upper boundary = 180, Tolerance = 10 (for 10 meters).

## 3.83 SQL Access to Oracle AW Properties

This dialog box displays the properties of a SQL Access to Oracle Analytic Workspaces (AW) object in a multidimensional model. For more information about working with multidimensional data, see *Oracle OLAP User's Guide* and the online help for Analytic Workspace Manager.

### General

**Name:** Name of the object.

**AW Name:** Analytic workspace where the source data is stored.

**Include GIDs:** Controls whether the grouping ID for each dimension member is included.

**Use Object Types:** Controls whether the next two fields (Object Type Name and Table Type Name) are accessible.

**Object Type Name:** Name of the object type.

**Table Type Name:** Name of the table type.

**Use Model Clause:** Controls whether the Include RowToCell default statement will be included in the SQL statement.

**Include RowToCell:** Controls whether the Use Model Clause default statement will be included in the SQL statement.

**SQL Statements:** Displays a dialog box in which you can view, modify, and change the order of attributes, and also see the SQL statement the reflects the current specifications. On the SQL Statement tabs in this box, the Show Formatted Limit Map option enables dimension information to be formatted (divided over several lines).

### Dimensions and Attributes

Lists any dimensions associated with the object. For each dimension, you can specify predefined attributes and hierarchies. To add a dimensions and attributes item, click

the Add (+) icon and select the elements to be included; to remove a dimensions and attributes item from the object definition, select it and click the Remove (X) icon.

**Measures**

Calculated measures can add information-rich data to a cube. The data set is calculated on the fly, so no data is stored. You can add as many calculated measures as you like without increasing the size of the database.

**Description**

Description of the SQL Access to Oracle AW object.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.84 Structured Attribute Properties

This dialog box defines properties of an attribute of a structured data type.

**General**

**Name:** Name of the attribute.

**Datatype:** Data type of the attribute.

**Mandatory:** Controls whether a value will be required in all columns that are based on this data type.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.85 Structured Type Properties

This dialog box is displayed when you create a new data type or edit a structured type listed in the display for Data Types.

**General**

**Name:** Name of the structured type.

**Super Type:** If this is a subtype, name of its supertype.

**Final:** Controls whether objects of this type can be inherited from in the definition of another type. If this option is enabled, this type cannot be inherited from (it cannot be a supertype in a type definition).

**Instantiable:** Controls whether objects of this type can be created. If this option is enabled, objects of this type can be created.

**Attributes**

Lists the attributes currently defined for the structured type. The properties for each attribute include its name and data type.

To add an attribute, click the Add (+) icon; to delete an attribute, select it and click the Remove (X) icon; to move an attribute up or down in the list, select it and click the appropriate arrow; to view the properties of an attribute, double-click in the cell to the left of the name.

**Methods**

Lists the methods currently defined for the structured type.

To add a method, click the Add (+) icon; to delete a method, select it and click the Remove (X) icon; to move a method up or down in the list, select it and click the appropriate arrow; to view the properties of a method, double-click its name.

**Comments, Notes, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.86 Subview Properties

This dialog box displays the properties of a subview. (See also [Section 1.3.4.1, "Logical Diagram and Subviews"](#) and [Section 1.3.5.1, "Relational Diagram and Subviews"](#).)

**General**

**Name:** Name of the subview.

**Visible:** Controls whether the subview diagram is displayed in the Data Modeler window. You can also control the visibility by selecting Show Diagram or Hide Diagram on the context menu after you right-click the subview in the object browser.

**Comments, Notes, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.87 Table Properties

This dialog box displays the properties of a table, which is part of [Relational Models](#).

You can click **Naming Rules** to apply the current naming rules to specified types of objects related to this table definition. For example, if you applied naming rules to Check Constraints and if a table-level constraint was named PRODUCTS\_Check, the name would be changed to PRODUCTS\_CK (if the table name was PRODUCTS).

**General**

**Name:** Name of the table.

**Long Name:** Long name in the form: relational-model-name:table-name

**Engineer:** Controls whether the table will be considered during reverse engineering operations. If this option is disabled, this table and its properties are not considered when the relational model is reverse engineered into the logical model.

**PK Name:** Name of the primary key of the table.

**Type:** Classification type, from the list defined in the General Options. Examples: Fact, Dimension, Logging, Summary, Temporary.

**Temp Table Scope:** For a table classified as Temporary, you can specify a scope, such as Session or Dimension.

**Register as Spatial Table:** For a table with a column of type SDO\_GEOMETRY, creates the spatial index and inserts the appropriate entry in the USER\_SDO\_GEOM\_METADATA view.

**Allow Type Substitution:** For a structured type with Reference disabled, or for a structured type applied to a table, controls whether a substitutional structured type generation in the DDL is allowed.

**Columns**

Lists the columns currently defined for the table. The properties for each column include its name and data type, and whether it is the primary key (PK), a foreign key (FK), or a required field (M, for mandatory).

To add a column, click the Add (+) icon; to delete a column, select it and click the Remove (X) icon; to view the properties of a column, double-click in the cell to the left of the name.

**Indexes**

Lists the indexes currently defined for the table. The properties for each index include its name, its state, and whether to generate the index when the table is created.

To add an index, click the Add (+) icon; to delete an index, select it and click the Remove (X) icon; to view the properties of an index, double-click in the cell to the left of the name.

**Foreign Keys**

Lists the foreign keys currently defined for the table. The properties for each key include its name, its parent table, its delete rule, and whether to generate a foreign key constraint for it when the table is created.

**Nested Columns**

For each column based on a structured data type that has attributes, lists each attribute (in *column-name.attribute-name* format). For each attribute, you can specify whether it is the primary key (PK), a foreign key (FK), or a required field (M, for mandatory).

**Oid Options and PK Columns**

Displays any OID (object identifier) settings and primary key columns based on a structured type.

**OID Is Primary Key:** Indicates whether the OID is the primary key of the table.

**User Defined or System Generated:** Indicates whether the OID is user-defined or generated by the database system.

**PK Columns for Table Based on Structured Type:** Displays the column name and data type for primary key columns that are based on a structured type.

**Volume Properties**

**Volumes: Minimum:** Minimum data volume for the table.

**Volumes: Expected:** Expected or typical data volume for the table.

**Volumes: Maximum:** Maximum data volume for the table.

**Growth Rate: Percent:** Expected growth rate percentage for the table, for each period as specified in the next field.

**Growth Rate: Year/Month/Day:** The period (year, month, or day) to which the expected growth rate applies.

**Normal Form:** The required normal form (database normalization) for the table: None, First, Second, Third, or Fourth.

**Adequately Normalized?:** YES indicates that the model is sufficiently normalized. NO indicates that the model is not sufficiently normalized, and that additional normalization may be required on the relational model.

**Spatial Properties**

Displays any currently defined Oracle Spatial properties, each being a data column (type SDO\_GEOMETRY or a function that evaluates to an SDO\_GEOMETRY object) in the table. You can double-click an item's name to display its [Spatial Definition Properties](#).

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

**Scripts**

Enables you to specify scripts to be run automatically at specified times or stages: before the table is dropped or renamed, before the table is created, after the table is created, and at the end of any script specified for the table.

**Include into DDL Script:** Controls whether the text of the specified scripts is included into the DDL script for operations on the table.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.88 Table to View

Creates views based on tables in a selected relational model. This dialog is displayed if you click Tools, then Table to View Wizard.

The tables in the selected relational model are displayed. Select all tables or a subset, optionally specify any naming rule, and click **Generate**. A view is created for each selected table. Each view selects all columns from the table, although you can later modify the view definition as needed. By default (if no Naming Rule is specified), each view name is its associated table name, prefixed with V\_. For example, if a table name is BOOKS, the associated view name by default will be V\_BOOKS.

**Naming Rule:** Displays any variables (relational model name or table name, or both in the specified order) that will be prefixed to the view name instead of V\_. To display a dialog box for selecting variables for the naming rule, click **Add Variable**. For example, assume that the relational model name is Library and the table name is BOOKS:

- If you specify only table name, the name of the generated view will be BOOKS. That is, all views will have the same name as their associated tables.
- If you specify only model name, the name of the generated view will be Library. You should probably not choose this option, because all views will have names starting with the model name and *lv1*, *lv2*, and so on added to names after the first one.
- If you specify model name and table name, the name of the generated view will be LibraryBOOKS.

## 3.89 Telephone Properties

This dialog box displays the properties of a telephone object, which is a type of [Business Information](#) object.

### General

**Name:** Name of the telephone object.

**Phone Number:** Telephone number, in any format appropriate for your needs. For example, you may want to include the country code for international dialing.

**Phone Type:** Type of phone, in any format appropriate for your needs. Examples: Mobile, Office, Home.

### Contacts

Lists any relevant contacts. To view the properties of a contact, double-click its name. (See also [Contact Properties](#).)

### Comments, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.90 Transformation Package

This dialog box displays the properties of a transformation package object, which is part of the [Process Model](#) and is described in [Section 1.3.3.2, "Transformation Packages"](#).

**Name:** Name of the transformation package.

**Comment:** Optional descriptive comment text.

## 3.91 Transformation <task-name>

This dialog box displays the properties of a transformation task object, which is part of the [Process Model](#).

### General

**Name:** Name of the transformation task.

**Comment:** Optional descriptive comment text.

**Visible:**

### Sources

Displays all sources and selected sources side by side, and lets you select items and use the arrow icons to move items from one side to the other.

### Targets

Displays all targets and selected targets side by side, and lets you select items and use the arrow icons to move items from one side to the other.

### Primary Transformations

Displays the primary transformation for the transformation task. To view the properties of the primary transformation, double-click its name. (See also [Transformation Properties](#).)



## 3.92 Transformation Properties

This dialog box displays the properties of a transformation, which is part of the [Process Model](#).

### General

**Name:** Name of the transformation.

**Synonym:** Synonym for the transformation.

**Sources Filter Condition:**

**Sources Join Condition:**

**Primary:** Controls whether this is the primary transformation for the associated transformation task.

### Information Structures

Lists any relevant information structures. To view the properties of an information structure, double-click its name. (See also [Information Structure Properties](#).)

### Processed Attributes

Lists any attributes processed by the transformation. To view the properties of an attribute, double-click its name. (See also [Attribute Properties](#).)

### Processed Entities

Lists any entities processed by the transformation. To view the properties of an entity, double-click its name. (See also [Entity Properties](#).)

### Source-Target Mapping

Displays any targets and sources for the transformation. For each target element, includes the type of transformation: As it is, Derivation, Aggregation, Summarization, or Complex Formula.

### Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.93 Transformation Flow Properties

This dialog box displays the properties of a transformation flow, which is part of the [Process Model](#).

### General

**Name:** Name of the transformation flow.

**Synonym:** Synonym for the transformation flow.

**Source:** Click to display the input parameters.

**Destination:** Click to display the [Transformation Properties](#) of the associated transformation.

**Logging Flow:** Controls whether the flow is specifically for logging operations. Creating a separate logging flow might simplify keeping track of information.

**Operations:** Specifies types of operations that the transformation flow can perform (create, read, update, delete).

#### **Information Structures**

Lists any relevant information structures. To view the properties of an information structure, double-click its name. (See also [Information Structure Properties](#).)

#### **External Data**

Lists any relevant external data objects. To view the properties of an external data object, double-click its name.

#### **System Objective**

Description of the system objective for this transformation flow.

#### **Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).

## 3.94 Types Administration

This dialog box is displayed when you click Tools, then Types Administration. It enables you to manage the mappings between logical data types and native data types for specific supported database products, and to add and remove logical types.

#### **Logical Types to Native Types**

For each logical type, you can select the type in the list on the left to view its mappings to a type in each supported database product.

To add a logical type and specify its mappings, click **Add**. To delete a logical type, select the type click **Remove**. To modify the mappings for a logical type (predefined or user-defined), select the type, click **Modify**, and specify the mapping information for any desired database products.

#### **Native Types to Logical Types**

For each supported database product, you can view the mappings between its native types and SQL Developer Data Modeler logical types.

## 3.95 View Properties

This dialog box displays the properties of a view, which is part of [Relational Models](#).

#### **General**

**Name:** Name of the view.

**View Builder:** Click to display a view builder interface, in which you can create a view by specifying tables and columns, a WHERE clause, and any GROUP BY, HAVING, and ORDER BY clauses.

**Based on Structured Type:** For a view based on a structured type, the name of the type.

**OID Columns:** Object identifier (OID) column name or names.

**Allow Type Substitution:** If applicable, controls whether a substitutional structured type generation in the DDL is allowed

**Reference Elements**

Displays any currently defined columns in the view.

**SQL for View on Structured Type**

For a view on a structured type, the SQL statement for the view.

**Comments in RDBMS**

Comment text to be included in database objects that are generated based on this modeling object.

**Impact Analysis**

Enables you to view and specify information to be used by Oracle Warehouse Builder for impact analysis.

**Scripts**

**Before Drop/Rename:** Statements to execute before the view is dropped or renamed.

**Before Create:** Statements to execute before the view is created.

**After Create:** Statements to execute after the view is created.

**End of Script:** Statements to execute after all other statements in the script have been executed.

**Include into DDL Script:** Enables user-defined scripts to be included in the generated DDL at defined events, such as before drop or end of script.

**Comments, Notes, Measurements, Change Requests, Responsible Parties, Documents, Summary**

See [Section 3.8, "Common Information in Dialog Boxes"](#).



## A

---

accelerator keys  
  for menus, 1-3  
Apply Naming Standards to Keys and Constraints  
  command, 1-6  
arcs  
  definition, 1-15  
  properties, 3-1  
attributes  
  definition of, 1-14  
  properties, 3-2  
auto route  
  adjusting lines in diagrams, 1-7  
  context menu command, 1-7  
  line auto route option, 1-28  
AutoLayout  
  context menu command, 1-7

## B

---

Bachman notation, 1-14  
background color  
  in diagrams, 1-28  
Barker notation, 1-14  
bind variables  
  for reports, 1-33  
blogs  
  SQL Developer, 1-34  
bottom-up modeling, 1-24  
business information, 1-21

## C

---

cardinality  
  definition of, 1-15  
Change Object Names Prefix command, 1-6  
change request  
  properties, 3-3  
classification types  
  setting for entities and tables, 1-6  
  specifying colors for in diagrams, 1-28  
collection type  
  properties, 3-4  
collection types, 1-11  
color

  background in diagrams, 1-28  
columns  
  properties, 3-4  
comparing models, 1-6  
composite process  
  definition of, 1-12  
connections  
  creating, editing, or selecting, 3-40  
  Oracle Database, 3-40, 3-41  
contacts  
  definition of, 1-21  
  properties, 3-7  
context menus (right-click menus), 1-6  
cubes  
  properties, 3-7

## D

---

data flow  
  definition of, 1-12  
data flow diagrams, 1-12  
data modeling  
  concepts and usage information, 1-1  
data store  
  definition of, 1-12  
data types  
  administering, 1-6, 3-56  
  model, 1-9  
database connections  
  creating, editing, or selecting, 3-40  
  Oracle Database, 3-40, 3-41  
DDL file editor, 3-10  
DDL files  
  selecting for import, 3-47  
DDL generation  
  options, 3-11  
design rules  
  checking a design for violations, 1-6  
  explanation and usage information, 3-11  
diagram color  
  background, 1-28  
diagrams  
  adjusting lines in (auto route setting), 1-7  
dimensions  
  properties, 3-12  
directories (directory objects), 1-18

discussion forum  
SQL Developer, 1-34

distinct types, 1-10  
properties, 3-13

documents  
definition of, 1-21  
properties, 3-14

domains  
administering, 1-5, 3-15  
importing, 3-26  
properties, 3-14

## E

---

elbows  
adding vertices in diagrams, 1-7  
disabling Auto Route before adding vertices in  
diagrams, 1-7  
removing vertices in diagrams, 1-7

emails  
definition of, 1-21  
properties, 3-15

entities  
definition of, 1-14  
properties, 3-17

events  
adding, 3-1  
properties, 3-18

external agents  
definition, 1-12  
properties, 3-20

external data  
properties, 3-21

## F

---

F10 key  
for File menu, 1-3

fit screen  
adjusting sizes and shapes to fit diagram  
window, 1-5

flows  
properties, 3-22  
showing name in diagrams, 1-28

foreign keys  
properties, 3-23

forward engineering, 1-5

## G

---

glossaries  
concepts (explanations), 3-24  
creating, 1-6  
editing, 1-6

glossary  
importing, 3-26

glossary editor, 3-24  
graphical user interface (GUI)  
Data Modeler, 1-2

grid  
showing and hiding, 1-27

## H

---

hierarchies  
properties, 3-25

## I

---

indexes  
definition of, 1-19  
properties, 3-28

information store  
definition of, 1-12

information stores  
properties, 3-29

information structures  
properties, 3-30

inheritance  
definition of, 1-14

## J

---

JDBC drivers (third party), 1-30

joins  
properties, 3-31

## K

---

keys  
properties, 3-32

## L

---

layout  
automatically rearranging in diagrams  
(AutoLayout), 1-7

levels  
properties, 3-32

line auto route  
adjusting lines in diagrams, 1-7  
context menu command, 1-7  
option, 1-28

locations  
definition of, 1-21  
properties, 3-34

logical model, 1-13  
logical types, 1-11

## M

---

measure folders  
properties, 3-34

measurements  
properties, 3-36

measures  
properties, 3-35

merging models, 1-6

methods  
properties, 3-36

modeling  
data modeling concepts and usage  
information, 1-1

Multimedia  
Oracle Multimedia logical types, 1-11

## N

---

name abbreviations, 3-39  
specifying file with, 1-6  
name translation  
applying during engineering, 3-16  
naming standard definitions  
importing, 3-26  
naming standards  
applying to keys and constraints, 1-6  
specifying, 1-28  
neighbors  
selecting in diagrams, 1-7

## O

---

object names prefix  
changing, 1-6  
Oracle by Example (OBE)  
Data Modeler, 1-34  
Oracle Database connections, 3-40, 3-41  
Oracle Multimedia  
logical types, 1-11  
ORD\* types  
Oracle Multimedia logical types, 1-11  
OTN page  
SQL Developer, 1-34

## P

---

physical models, 1-17  
preferences  
for data modeling, 1-26  
prefixes  
Change Object Names Prefix command, 1-6  
primitive process  
definition of, 1-12  
process model, 1-11  
processes  
definition, 1-12  
properties, 3-42

## R

---

RDBMS site  
definition of, 1-18  
editing, 3-43  
RDBMS sites  
administering, 1-6  
read-only data modeling viewer, 1-33  
record structures  
properties, 3-44  
relational models, 1-15  
relations  
definition of, 1-15, 1-17  
properties, 3-44  
reporting repository, 1-32  
reports, 1-32

bind variables for, 1-33  
resource locators  
definition of, 1-22  
properties, 3-45  
responsible parties  
definition of, 1-22  
properties, 3-46  
reverse engineering, 1-5  
right-click menus (context menus), 1-6  
role  
definition of, 1-12  
roles  
properties, 3-46  
rollup links  
properties, 3-47

## S

---

select neighbors  
context menu command, 1-7  
Select Relational Models dialog box  
controlling display of, 1-26  
sequences  
definition of, 1-19  
shortcut keys  
for menus, 1-3  
slices  
properties, 3-48  
spatial definitions  
properties, 3-49  
SQL Access to Oracle AW  
properties, 3-49  
SQL Developer preferences  
for data modeling, 1-26  
structured types, 1-11  
subviews  
creating from selected objects, 1-7  
properties, 3-51  
synchronize with tree  
option, 1-28  
synonyms  
creating, 1-7

## T

---

table to view wizard, 1-6, 3-53  
tables  
definition of, 1-16, 1-20  
properties, 3-51  
targeted modeling, 1-25  
telephones  
definition of, 1-22  
properties, 3-54  
third party JDBC drivers, 1-30  
top-down modeling, 1-22  
transformation flows  
properties, 3-55  
transformation packages, 1-13  
properties, 3-54  
transformation tasks

- definition, 1-12
- properties, 3-54
- transformations
  - properties, 3-55
- tutorial
  - creating models for a small database, 2-1
- type substitution, 1-15
- types
  - administering, 1-6, 3-56
  - data types model, 1-9

## U

---

- unique identifiers (UIDs)
  - definition of, 1-14
- URLs
  - resource locator objects, 1-22
  - resource locator properties, 3-45
- user interface (UI)
  - Data Modeler, 1-2

## V

---

- vertices (elbows)
  - disabling Auto Route before adding elbows in diagrams, 1-7
- viewer (read-only) data modeling tool, 1-33
- viewlets
  - Data Modeler, 1-34
- views
  - definition of, 1-20
  - properties, 3-56
  - table to view wizard, 3-53

## Z

---

- Zachman framework, 1-1
- zoom in, 1-5
- zoom out, 1-5