# Oracle® SQL Developer

## Data Modeler User's Guide

Release 18.1
E94838-01
March 2018

Oracle SQL Developer Data Modeler User's Guide, Release 18.1

E94838-01

Copyright © 2008, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Celin Cherian

Contributing Authors: Chuck Murray

Contributors: Philip Stoyanov

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

# Contents

## Preface


Audience xii
Documentation Accessibility xii
Product Accessibility xii
Related Documents xii
Conventions xiii
Third-Party License Information xiii


## 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 1-8
1.3 Working with Data Modeler 1-10
1.3.1 Database Design 1-11
1.3.2 Data Types Model 1-11
1.3.2.1 Data Types Diagram and Subviews 1-12
1.3.2.2 Distinct Types 1-13
1.3.2.3 Structured Types 1-14
1.3.2.4 Collection Types 1-14
1.3.2.5 Logical Types 1-14
1.3.3 Process Model 1-15
1.3.3.1 Data Flow Diagrams 1-16
1.3.3.2 Transformation Processes and Packages 1-16
1.3.4 Logical Model 1-16
1.3.4.1 Logical Diagram and Subviews 1-17
1.3.4.2 Entities 1-18
1.3.4.3 Attributes 1-18
1.3.4.4 Unique Identifiers (UIDs) 1-18
1.3.4.5 Inheritances 1-18
1.3.4.6 Relations 1-18
1.3.4.7 Arcs 1-19

1.3.4.8 Type Substitution 1-19
1.3.4.9 Views 1-19
1.3.5 Relational Models 1-19
1.3.5.1 Relational Diagram and Subviews 1-20
1.3.5.2 Tables 1-21
1.3.5.3 Columns 1-21
1.3.5.4 Indexes 1-21
1.3.5.5 Relations 1-21
1.3.5.6 Relational Views 1-21
1.3.6 Physical Models 1-22
1.3.6.1 Clusters 1-22
1.3.6.2 Contexts 1-23
1.3.6.3 Dimensions 1-23
1.3.6.4 Directories 1-23
1.3.6.5 Disk Groups 1-23
1.3.6.6 External Tables 1-23
1.3.6.7 Indexes 1-23
1.3.6.8 Roles 1-23
1.3.6.9 Rollback Segments 1-24
1.3.6.10 Segments (Segment Templates) 1-24
1.3.6.11 Sequences 1-24
1.3.6.12 Snapshots 1-24
1.3.6.13 Stored Procedures 1-24
1.3.6.14 Synonyms 1-24
1.3.6.15 Structured Types 1-24
1.3.6.16 Tables 1-25
1.3.6.17 Tablespaces 1-25
1.3.6.18 Users 1-25
1.3.6.19 Views 1-25
1.3.7 Business Information 1-25
1.3.7.1 Contacts 1-26
1.3.7.2 Documents 1-26
1.3.7.3 Emails 1-26
1.3.7.4 Locations 1-26
1.3.7.5 Resource Locators 1-26
1.3.7.6 Responsible Parties 1-27
1.3.7.7 Telephones 1-27
1.4 Approaches to Data Modeling 1-27
1.4.1 Top-Down Modeling 1-27
1.4.2 Bottom-Up Modeling 1-29
1.4.3 Targeted Modeling 1-30

1.5 User Preferences for Data Modeler 1-31
1.5.1 Environment 1-31
1.5.2 Data Modeler 1-32
1.5.2.1 DDL 1-33
1.5.2.2 Diagram 1-36
1.5.2.3 Model 1-36
1.5.2.4 Reports 1-39
1.5.2.5 Search 1-40
1.5.2.6 Third Party JDBC Drivers 1-40
1.5.3 Format 1-40
1.5.4 Global Ignore List 1-41
1.5.5 Mouse Actions 1-41
1.5.6 Shortcut Keys (Accelerator Keys) 1-42
1.5.7 SSH (Secure Shell) 1-43
1.5.8 Usage Reporting 1-43
1.5.9 Versioning 1-43
1.5.10 Web Browser and Proxy 1-44
1.6 Saving, Opening, Exporting, and Importing Designs 1-45
1.6.1 Importing a DDL File 1-46
1.6.2 Importing Cube Views Metadata 1-46
1.6.3 Importing from Microsoft XMLA 1-46
1.6.4 Importing an ERwin File 1-46
1.6.5 Importing from a Data Dictionary 1-47
1.6.6 Importing an Oracle Designer Model 1-47
1.6.7 Importing a Data Modeler Design 1-47
1.6.8 Importing a Domain 1-47
1.7 Exporting and Importing Preferences and Other Settings 1-47
1.7.1 Restoring the Original Data Modeler Preferences 1-48
1.8 Data Modeler Reports 1-48
1.8.1 Generating Reports as RTF, HTML, or PDF Files 1-48
1.8.2 Using the Reporting Repository and Reporting Schema 1-50
1.8.3 Using SQL Developer to View Exported Reporting Schema Data 1-51
1.8.3.1 Design Content reports 1-52
1.8.3.2 Design Rules reports 1-52
1.9 Using Versioning 1-53
1.9.1 About Subversion and Data Modeler 1-53
1.9.1.1 Pending Changes 1-53
1.9.2 Basic Workflow: Using Subversion with a Design 1-54
1.10 For More Information About Data Modeling 1-55

## 2 Data Modeler Tutorial: 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 Data Modeler Dialog Boxes


3.1 Add Event 3-1
3.2 Add/Remove Objects 3-1
3.3 Advanced Properties (Connections) 3-2
3.4 Arc Properties 3-2
3.5 Attribute Properties 3-3
3.6 Change Subview Object Names Prefix 3-4
3.7 Change Request Properties 3-4
3.8 Change Requests Administration 3-5
3.9 Check for Updates 3-5
3.10 Choose Directory 3-6
3.11 Collection Type Properties 3-6
3.12 Column Properties 3-7
3.13 Common Information in Dialog Boxes 3-9
3.14 Compare Mapping 3-10
3.15 Compare Modeling Designs 3-10
3.16 Compare Models 3-10
3.17 Color Palette and Custom Colors 3-12
3.18 Connection Information 3-12
3.19 Contact Properties 3-12
3.20 Create Database Connection 3-13
3.21 Create Discovered Foreign Keys 3-14
3.22 Cube Properties 3-15
3.23 Custom Reports Template 3-17
3.24 Data Dictionary Connections 3-18
3.25 Data Dictionary Import (Metadata Extraction) 3-19
3.26 Database Connection Editor 3-20
3.27 DDL File Editor 3-20
3.28 DDL Generation Options 3-21

3.29 Design Properties 3-22
3.30 Design Rules 3-24
3.30.1 Design Rules 3-24
3.30.2 Custom Rules 3-25
3.30.3 Libraries 3-26
3.30.4 Transformations 3-26
3.31 Dimension Properties 3-26
3.32 Display Properties 3-28
3.33 Distinct Type Properties 3-28
3.34 Document Properties 3-28
3.35 Domain Properties (Domains Model) 3-29
3.36 Domains Administration 3-29
3.37 Email Properties 3-30
3.38 Engineering 3-30
3.39 Entity Properties 3-32
3.40 Event Properties 3-34
3.41 Export Cube Views Metadata 3-35
3.42 Export to Microsoft XMLA 3-35
3.43 Export to Oracle Analytic Workspaces 3-35
3.44 Export to Reporting Schema 3-36
3.45 Export Wizard 3-37
3.46 Export/Import Connections 3-38
3.46.1 Export Connections 3-38
3.46.2 Import Connections 3-39
3.47 External Agent Properties 3-39
3.48 External Data Properties 3-41
3.49 File Processing 3-41
3.50 Find Object (Search) 3-41
3.51 Flow Properties 3-42
3.52 Foreign Key Properties 3-43
3.53 Glossary Editor 3-44
3.54 Hierarchy Properties 3-46
3.55 Import Domains 3-47
3.56 Import Glossary (Naming Standard Definitions) 3-47
3.57 Import Mapped Models from VAR Files 3-48
3.58 Import Oracle Designer Model 3-48
3.59 Import Data Modeler Design 3-48
3.60 Import Database Connections 3-49
3.61 Import VAR File: Select Type of Import 3-49
3.62 Index, Primary Key, or Unique Key Properties 3-49
3.63 Information Store Properties 3-50

3.64 Information Structure Properties 3-52
3.65 Inheritance Relation Properties - <hierarchy-name> 3-53
3.66 Join Properties 3-53
3.67 Level Properties 3-54
3.68 Location Properties 3-55
3.69 Manage Features and Updates 3-56
3.70 Logical Type 3-56
3.71 Mask Templates Administration 3-57
3.72 Measure Folder Properties 3-57
3.73 Measure Properties 3-58
3.74 Measurement Properties 3-59
3.75 Method Properties 3-59
3.76 Model Properties - Business Information 3-60
3.77 Model Properties - <data-flow-diagram-name> 3-60
3.78 Model Properties - Data Types 3-61
3.79 Model Properties - Logical 3-61
3.80 Model Properties - <multidimensional-model-name> 3-61
3.81 Model Properties - Process Model 3-62
3.82 Model Properties - <name> (Relational) 3-62
3.83 Name Abbreviations 3-63
3.84 New/Edit SSH Connection 3-63
3.85 New/Update Database Connection 3-64
3.86 Object Names Administration 3-65
3.87 Process Properties 3-66
3.88 RDBMS Site Editor 3-68
3.89 Record Structure Properties 3-68
3.90 Relation Properties 3-69
3.91 Relational Models 3-70
3.92 Report Templates Management 3-70
3.93 Resource Locator Properties 3-71
3.94 Responsible Party Properties 3-71
3.95 Revision Lister 3-72
3.96 Role Properties 3-72
3.97 Rollup Link Properties 3-73
3.98 Rule Set Properties 3-73
3.99 Search Profile 3-73
3.100 Schema Properties 3-73
3.101 SELECT DDL Files 3-74
3.102 Select File 3-74
3.103 Select Models/Subviews to Export 3-74
3.104 Sensitive Type Properties 3-75

3.105 Set Classification Types 3-76
3.106 Set Common Properties 3-76
3.107 Set Data Type 3-76
3.108 Show/Hide Elements 3-76
3.109 Slice Properties 3-77
3.110 Spatial Definition Properties 3-78
3.111 SQL Access to Oracle AW Properties 3-78
3.112 Standard Reports Configurations 3-79
3.113 Structured Attribute Properties 3-80
3.114 Structured Type Properties 3-80
3.115 Subversion: Add Property 3-81
3.116 Subversion: Add to Source Control 3-81
3.117 Subversion: Apply Patch 3-81
3.118 Subversion: Branch/Tag 3-82
3.119 Subversion: Check Out from Subversion 3-82
3.120 Subversion: Commit Resources 3-83
3.121 Subversion: Commit Working Copy 3-83
3.122 Subversion: Confirm Checkout 3-83
3.123 Subversion: Create Remote Directory 3-84
3.124 Subversion: Create Subversion Repository 3-84
3.125 Subversion: Create/Edit Subversion Connection 3-84
3.126 Subversion: Delete Resources 3-85
3.127 Subversion: Edit Configuration File 3-85
3.128 Subversion: Export Files 3-85
3.129 Subversion: Export Subversion Connections 3-86
3.130 Subversion: History 3-86
3.131 Subversion: Ignore 3-86
3.132 Subversion: Import Subversion Connections 3-86
3.133 Subversion: Import to Subversion 3-87
3.134 Subversion: Lock Resources 3-88
3.135 Subversion: Merge 3-88
3.136 Subversion: Pending Changes 3-89
3.137 Subversion: Properties 3-89
3.138 Subversion: Remove from Subversion 3-89
3.139 Subversion: Repository Browser 3-90
3.140 Subversion: Revert Local Changes 3-90
3.141 Subversion: Switch 3-90
3.142 Subversion: Unlock Resources 3-91
3.143 Subversion: Update Resources 3-91
3.144 Subversion: Update Working Copy 3-91
3.145 Subversion: Versioning Properties 3-92

3.146 Subversion: XML Metadata Comparator 3-92
3.147 Subview Properties 3-92
3.148 Table Properties 3-93
3.149 Table to View 3-97
3.150 Table DDL Transformation Scripts 3-97
3.151 Telephone Properties 3-98
3.152 Transformation Package 3-98
3.153 Transformation <task-name> 3-98
3.154 Transformation Properties 3-99
3.155 Transformation Flow Properties 3-100
3.156 TSDP Policy Properties 3-100
3.157 Types Administration 3-101
3.158 Types to Domains 3-101
3.159 Unable to Connect 3-102
3.160 Unique Identifier (UID, or Key) Properties 3-102
3.161 View Properties (Logical Model) 3-102
3.162 View Properties (Relational Model) 3-103
3.163 View to Table 3-105
3.164 Windows 3-105

## Index

# 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## Product Accessibility

The *Oracle SQL Developer Data Modeler Accessibility Guide* provides information about the accessibility features for SQL Developer Data Modeler.

## Related Documents

To download free release notes, installation documentation, white papers, or other collateral, go to the Oracle Technology Network (OTN) at

`http://www.oracle.com/technetwork/`

The documentation section of the OTN site is at

`http://www.oracle.com/technetwork/indexes/documentation/`

# Conventions

The following text conventions are used in this document:

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

# 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, which has the following associated attribution text:

This product includes software developed by The Apache Software Foundation (`http://www.apache.org/`). This software contains code from the World Wide Web Consortium (W3C) for the Document Object Model API (DOM API) and SVG Document Type Definition (DTD). This software contains code from the International Organisation for Standardization for the definition of character entities used in the software's documentation. 2. Must include Apache License, Version 2.0 with any distribution of Apache Batik 1.7: 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. END OF TERMS AND CONDITIONS Appendix: How to Apply Apache License to Your Work To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives. Copyright [yyyy] [name of copyright owner] Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at `http://www.apache.org/licenses/LICENSE-2.0` Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## Log4j 1.2.9

This program contains Log4j 1.2.9, which has the following associated attribution text:

The Apache Software License, Version 1.1 * * Copyright (c) 2000 The Apache Software Foundation. All rights * reserved. * * Redistribution and use in source and binary forms, with or without * modification, are permitted provided that the following conditions * are met: * * 1. Redistributions of source code must retain the above copyright * notice, this list of conditions and the following disclaimer. * * 2. Redistributions in binary form must reproduce the above copyright * notice, this list of conditions and the following disclaimer in * the documentation and/or other materials provided with the * distribution. * * 3. The end-user documentation included with the redistribution, * if any, must include the following acknowledgment: * "This product includes software developed by the * Apache Software Foundation (Link1 /)." * Alternately, this acknowledgment may appear in the software itself, * if and wherever such third-party acknowledgments normally appear. * * 4. The names "Apache" and "Apache Software Foundation" must * not be used to endorse or promote products derived from this * software without prior written permission. For written * permission, please contact apache@apache.org. * * 5. Products derived from this software may not be called "Apache", * nor may "Apache" appear in their name, without prior written * permission of the Apache Software Foundation. * * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE * DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF * SUCH DAMAGE. * ========== * * This software consists of voluntary contributions made by many * individuals on behalf of the Apache Software Foundation. For more * information on the Apache Software Foundation, please see * <Link2 />. * * Portions of this software are based upon public domain software * originally written at the National Center for Supercomputing Applications, * University of Illinois, Urbana-Champaign. */

## MigLayout 4.0

Copyright (c) 2004, Mikael Grev, MiG InfoCom AB. (miglayout (at) miginfocom (dot) com)

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the MiG InfoCom AB nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## POI 2.5.1

This program contains POI 2.5.1, which has the following associated attribution text:

The following applies to all products licensed under the Apache 2.0 License: You may not use the identified files except in compliance with the Apache License, Version 2.0 (the "License.") You may obtain a copy of the License at `http://www.apache.org/`. A copy of the license is also reproduced below. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Apache License Version 2.0, January 2004 Link2 / 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. END OF TERMS AND CONDITIONS APPENDIX: How to apply the Apache License to your work. To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives. Copyright [yyyy] [name of copyright owner] Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at Link3 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Oracle must give recipients a copy of this license (doc).

## RSyntaxTextArea 2.0.3

This program contains RSyntaxTextArea 2.0.3, which has the following associated attribution text:

Copyright (c) 2012, Robert Futrell

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 1
# Data Modeler Concepts and Usage

SQL Developer Data Modeler (referred to as Data Modeler) is a data modeling and database design tool that provides an environment for capturing, modeling, managing, and exploiting metadata.

See the Related Topics to learn more about Data Modeler.

**Related Topics**


Installing and Getting Started with SQL Developer Data Modeler (page 1-1)

Data Modeler User Interface (page 1-2)

Working with Data Modeler (page 1-10)

Approaches to Data Modeling (page 1-27)

User Preferences for Data Modeler (page 1-31)

Saving, Opening, Exporting, and Importing Designs (page 1-45)

Exporting and Importing Preferences and Other Settings (page 1-47)

Data Modeler Reports (page 1-48)

For More Information About Data Modeling (page 1-55)

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)


## 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 `<datamodeler_install>`. For example, on a Windows system you might want to choose `C:\` as this location.

   Unzipping the Data Modeler kit causes a directory named `datamodeler` to be created under the `<datamodeler_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 `datamodeler` directory under the `<datamodeler_install>` directory, and do one of the following:

   On Linux and Mac OS X systems, run *sh datamodeler.sh*.

   On Windows systems, double-click `datamodeler64.exe` (Windows 64-bit systems) or `datamodeler.exe` (Windows 32-bit systems).

If you are asked to enter the full pathname for the JDK, click **Browse** and find it. For example, on a Windows system the path might have a name similar to `C:\Program Files\Java\jdk1.7.0_51`

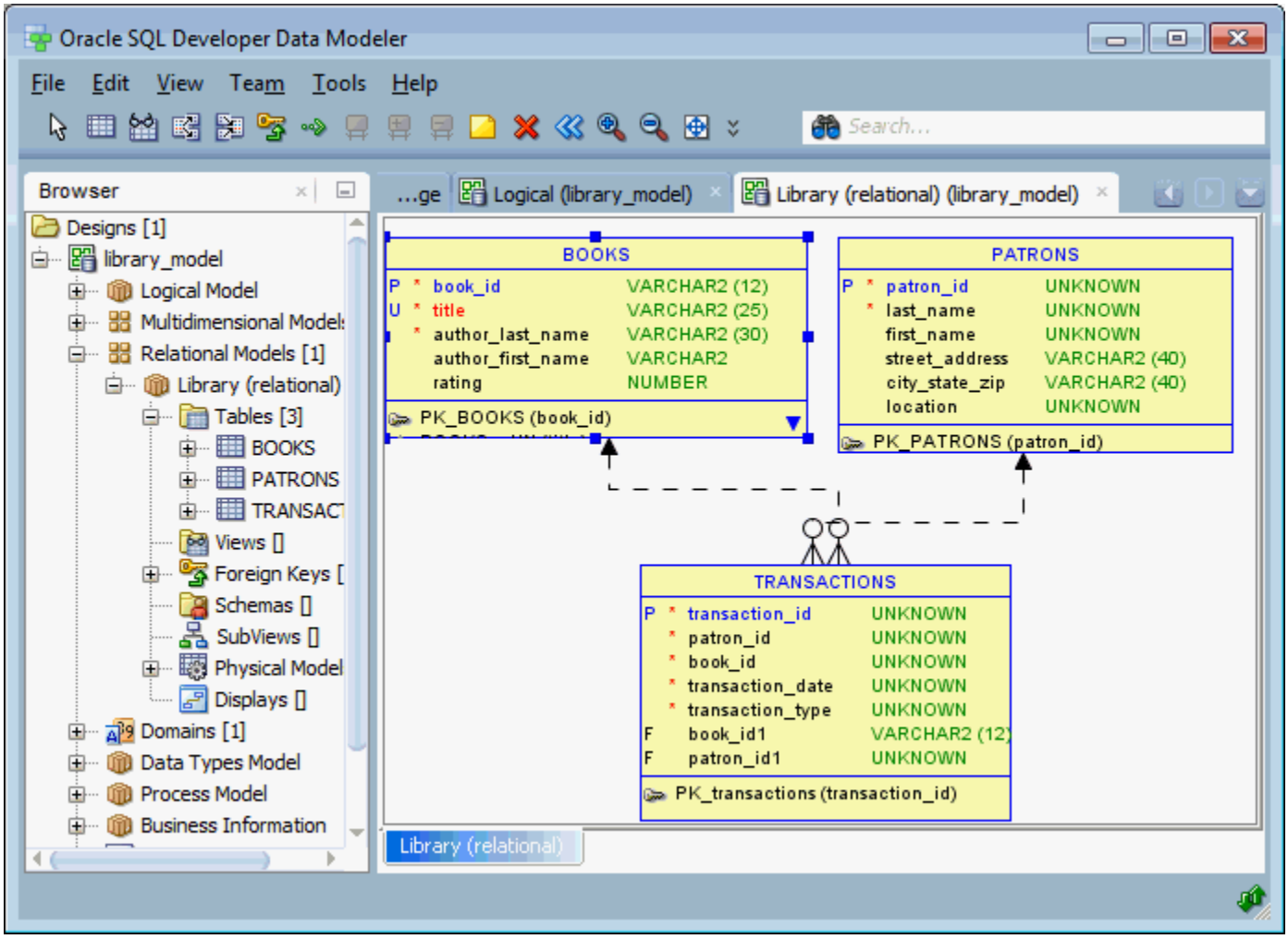
3. If you want to become familiar with data modeling concepts before using the interface, read the rest of this chapter before proceeding to the next step.
4. Do the short tutorial in Data Modeler Tutorial: Modeling for a Small Database (page 2-1). (For more advanced tutorials and other materials, see For More Information About Data Modeling (page 1-55).)

# 1.2 Data Modeler User Interface

The 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 (page 1-2) shows the main window.

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

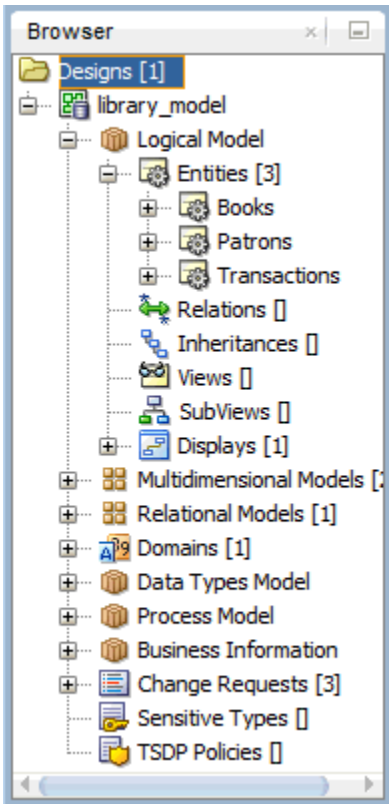


The menus at the top contain some standard entries, plus entries for features specific to Data Modeler (see Menus for Data Modeler (page 1-4)), as shown in the following figure.

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+S for Help, then Search. You can also display the File menu by pressing the F10 key.

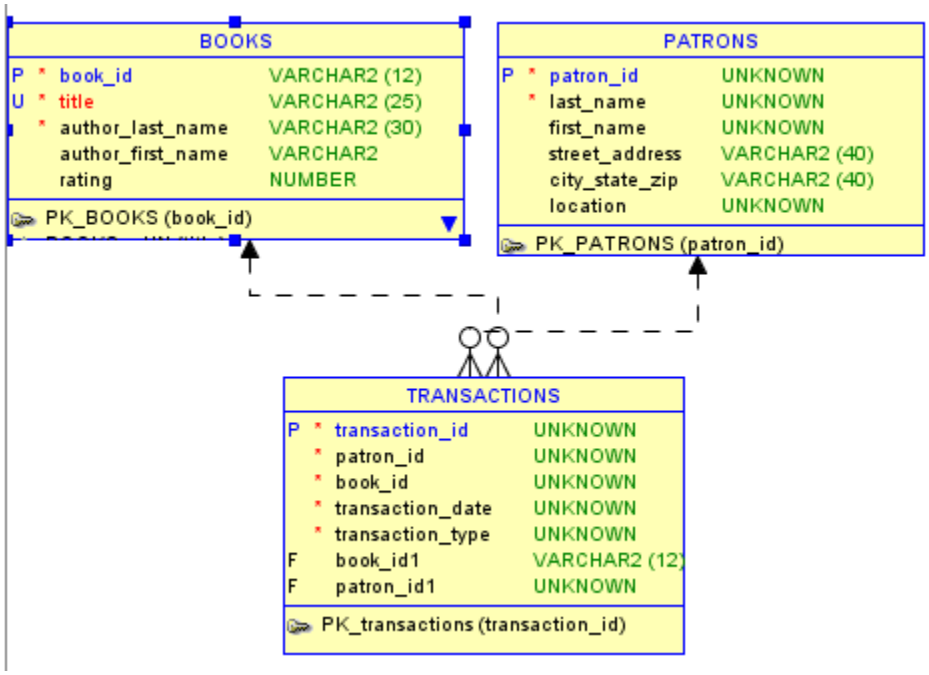
Icons under the menus perform actions relevant to what is currently selected for display on the right side of the window, such as the logical model, a relational model, or a data flow diagram. For example, for a relational model the icons include New Table, New View, Split Table, Merge Tables, New FK Relation, Generate DDL, Synchronize Model with Data Dictionary, and Synchronize Data Dictionary with Model. 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 Data Modeler Tutorial: Modeling for a Small Database (page 2-1)).




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.

**Related Topics**

Menus for Data Modeler (page 1-4)

Context Menus (page 1-8)

Data Modeler (page 1-32)

Data Modeler Concepts and Usage (page 1-1)

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)

## 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 Data Modeler design that had been saved or exported. For more information, see Saving, Opening, Exporting, and Importing Designs (page 1-45).

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

**Close All**: Closes all open designs without exiting Data Modeler.

**Import**: Lets you import models from a variety of sources. For more information, see Saving, Opening, Exporting, and Importing Designs (page 1-45).

**Export**: Lets you export models to files that can be imported into a variety of data modeling tools. For more information, see Saving, Opening, Exporting, and Importing Designs (page 1-45).

**Reports**: Lets you generate Data Modeler Reports (page 1-48).

**Page Setup**: Displays a dialog box where you can specify the following for any diagram print operations: media **Size** (Letter, Legal, or other predefined size) and **Source** (Automatically Select or a specified paper source), **Orientation** (Portrait, Landscape, Reverse Portrait, Reverse Landscape), and **Margins** (left, right, top, bottom).

**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, to a scalable vector graphics (.svg) file, or to an HTML/SVG (.html) file.

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

**Exit**: Closes any open designs and exits Data Modeler.

#### Edit menu

Contains standard Edit menu options related to cutting, copying, pasting, deleting, aligning, and finding objects.

#### View menu

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

**Show Status Bar**: Toggles the displaying of the status bar at the bottom of the Data Modeler window.

**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. The Navigator appears by default in the right side of the window.

**Log**: Displays the Messages - Log pane with a record of Data Modeler actions during the current invocation.

**External Log**: Displays a separate window with a record of all invocations of Data Modeler for the current full release number.

**External Log:** Displays a record of Data Modeler actions in an external viewer instead of in a pane within the Data Modeler window.

**Files**: Displays the Files pane for navigating the local file system.

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

**View Details**: Controls the level of detail in displays. Including Comments causes any Comments in RDBMS text to appear diagram displays for the logical model (entities and attributes) and relational models (tables and columns).

**DDL Preview** (relational diagram objects): Shows the DDL statements that would be generated to create the object. (When the DDL Preview window is displayed, you can click on other objects in the relational diagram to see the DDL statements that would be generated to create those objects.)

**DDL File Editor**: Lets you generate DDL statements for a selected physical model. Displays the DDL File Editor (page 3-20) dialog box. (This command is equivalent to clicking the Generate DDL icon, or clicking File, then Export, then DDL File.)

**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.

**Default Size** (and corresponding icon): Adjusts the shapes and text labels in the currently selected diagram to the default sizes.

**Find** (Search) Displays a dialog box for finding objects in the currently selected diagram. Useful for finding objects in large, complex diagrams. (See Find Object (Search) (page 3-41).)

> **Note:**
>
> To do a *global* search across all open models, use the search (***binoculars*** icon) box in the top-right area of the window.

**Refresh**: Updates the contents of the Data Modeler window to reflect current information.

**Full Screen:** Lets you toggle between a full-screen view of the Data Modeler window and the current or most recent non-full-screen view.

#### Team menu

Contains options related to support for the Subversion version management and source control system. See Using Versioning (page 1-53) for more information.

**Versions**: Lets you display the Versions Navigator and the Pending Changes (page 1-53) window.

The other commands on the Team menu depend on which version management and source control systems are available for use with Data Modeler.

#### Tools menu

Invokes 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 (page 3-29) dialog box.

**Types Administration**: Lets you view, modify, add, and delete logical types. Displays the Types Administration (page 3-101) 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 (page 3-68) dialog box.

**Mask Templates Administration**: Lets you create one or more "templates" that you can then associate with appropriate columns in tables in a relational model. Displays the Mask Templates Administration (page 3-57) dialog box.

**Table to View Wizard**: Lets you create views based on tables in a selected relational model. Displays the Table to View (page 3-97) wizard.

**View to Table Wizard**: Lets you create tables based on views in a selected relational model. Displays the View to Table (page 3-105) 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 (page 3-63) 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 (page 3-44) dialog box.

**Object Names Administration**: Lets you make the names of specified objects fixed (unchangeable) or changeable in dialog boxes for the properties of the objects. Displays the Object Names Administration (page 3-65) dialog box.

**Design Rules**: Lets you check your current design for violations of Data Modeler design rules. Displays the Design Rules (page 3-24) dialog box.

**Engineering Status**: Displays the Engineering (page 3-30) 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 (page 3-70) dialog box is displayed.

**General Options**: Lets you customize the behavior of Data Modeler. Displays the Data Modeler (page 1-32) dialog box.

**Help Menu**

Displays help about Data Modeler. The Help Center window includes the following icons in each tab:

- **Keep on Top**: Toggles whether to keep the Help Center window on top of the Data Modeler window.
- **Navigators**: Lets you select the Contents or Favorites navigator.
- **Print**: Prints the topic.
- **Change Font Size**: Lets you increase or decrease the font size for the display of the current help topic.
- **Add to Favorites**: Adds the topic to the list in the Favorites navigator.
- **Find**: Lets you search for a string in the current help topic.

**Search**: Displays the Help Center window, with focus in the Search (binoculars icon) box. You can enter one or more strings to be searched for in the online help.

**Table of Contents**: Displays the Help Center window, with the Contents tab selected.

**Start Page**: Displays a page with links for options for learning about Data Modeler. The options include a link to a page with **Sample Models and Scripts**.

**Release Notes**: Displays important information this release of Data Modeler, including requirements and some usage notes.

**About**: Displays version-related and other information about Data Modeler, its properties, and installed extensions.

## 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:

- **Change Subview Object Names Prefix**: Specifies the new prefix to replace a specified current prefix for selected types of objects. Displays the Change Subview Object Names Prefix (page 3-4) dialog box.
- **Apply Custom Transformation Scripts**: Displays the Custom Transformation Scripts dialog box, where you can select scripts to be applied. (For more information about custom transformation scripts, see Transformations (page 3-26).)
- **Discover Foreign Keys**: Lets you discover foreign key relationships among tables in the relational model, and to create foreign keys. (See Create Discovered Foreign Keys (page 3-14).)
- **Engineer to Relational Model** (with the logical model selected): 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** (with a relational model selected): Performs reverse engineering: updates the logical model from the selected relational model.

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

- **Create Discovered Foreign Keys** (relational model) Displays discovered hidden foreign key relationships in a relational model. (See Create Discovered Foreign Keys (page 3-14).)
- **Remove Discovered Foreign Keys** (relational model): Removes any discovered foreign keys from the relational model diagram.
- **Create Subview**: Creates a subview. (See also Logical Diagram and Subviews (page 1-17) and Relational Diagram and Subviews (page 1-20).)
- **Create Display**: Creates a separate display pane of the view or subview. Displays enable you to represent the same set of objects in different ways. For example, you can create a display, select it, and then experiment with changing the context-menu settings for View Details, Show (Grid, Page Grid, Labels, Legend), and Diagram Color.Displays are in the same window together with related main diagram or subview; however, you can find tabs for displays at the bottom of that window. To remove a display, right-click in it and select **Delete Display**.
- **Auto Route**: Toggles the setting of the Line Auto Route option (see Diagram (page 1-36) under Data Modeler (page 1-32)). 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.

- **Straighten Lines** (available only if Auto Route is disabled): Removes any elbows so that the line connects only the start and end points.
- **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.
- **View Details**: Lets you view all available details for objects or only selected details.
- **Show**: **Grid** displays a grid in the background, which can help you to align objects vertically and horizontally on the diagram; **Page Grid** displays where page margins will be in printed PDF output; **Labels** displays the foreign key names on relationship arrows and the flow names on flow lines in data flow diagrams; **Legend** displays a legend box (which you can drag to move) containing the diagram name, author, creation date, and other information.
- **Resize Objects to Visible**: Resizes objects in the diagram so that all are visible in the display area.
- **Diagram Color**: Displays a dialog box for selecting the color scheme for the background on diagrams.
- **Properties**: Displays the dialog box for viewing and editing properties of the model.

In diagrams, if you right-click a line connecting two objects, 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 connects 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 Logical Diagram and Subviews (page 1-17) and Relational Diagram and Subviews (page 1-20).)
- **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. (When the DDL Preview window is displayed, you can click on other objects in the relational diagram to see the DDL statements that would be generated to create those objects.)
- **Format**: Lets you specify colors and fonts for the selected objects.
- **Show/Hide Elements**: Lets you hide specified elements in the display.
- **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 Data Modeler

You can use 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 the toolbar icons.

  For example, to create an entity, select the Logical tab; click the New Entity toolbar icon; then define the entity in the Entity Properties (page 3-32) box. To edit an entity, either double-click its box in the diagram or right-click the box and select **Properties**.

Context Menus (page 1-8) (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:

- Database Design (page 1-11)
- Data Types Model (page 1-11)
- Process Model (page 1-15)

- Logical Model (page 1-16)
- Relational Models (page 1-19)
- Physical Models (page 1-22)
- Business Information (page 1-25)

## 1.3.1 Database Design

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 (with the extension `.dmd`) in a folder or directory that you specify, and subfolders or subdirectories are created as needed under it. The `.dmd` file contains pointers to information in these subfolders or subdirectories. For example, for a very basic design named `my_db_design`, the following hierarchy might be created starting at the folder or directory in which you created it:

```
my_db_design.dmd
my_db_design
   businessinfo
   datatypes
      subviews
   logical
      entity
      subviews
   mapping
   pm
   rdbms
   rel
      1
         subviews
         table
```

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

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 1.3.2 Data Types Model

Data Modeler supports supertypes and subtypes in its logical model, but it also provides the data types model, to be CWM (Common Warehouse Metamodel) 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 Data Modeler
- By importing from Oracle Designer repository. See Importing an Oracle Designer Model (page 1-47).

The data types model in 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

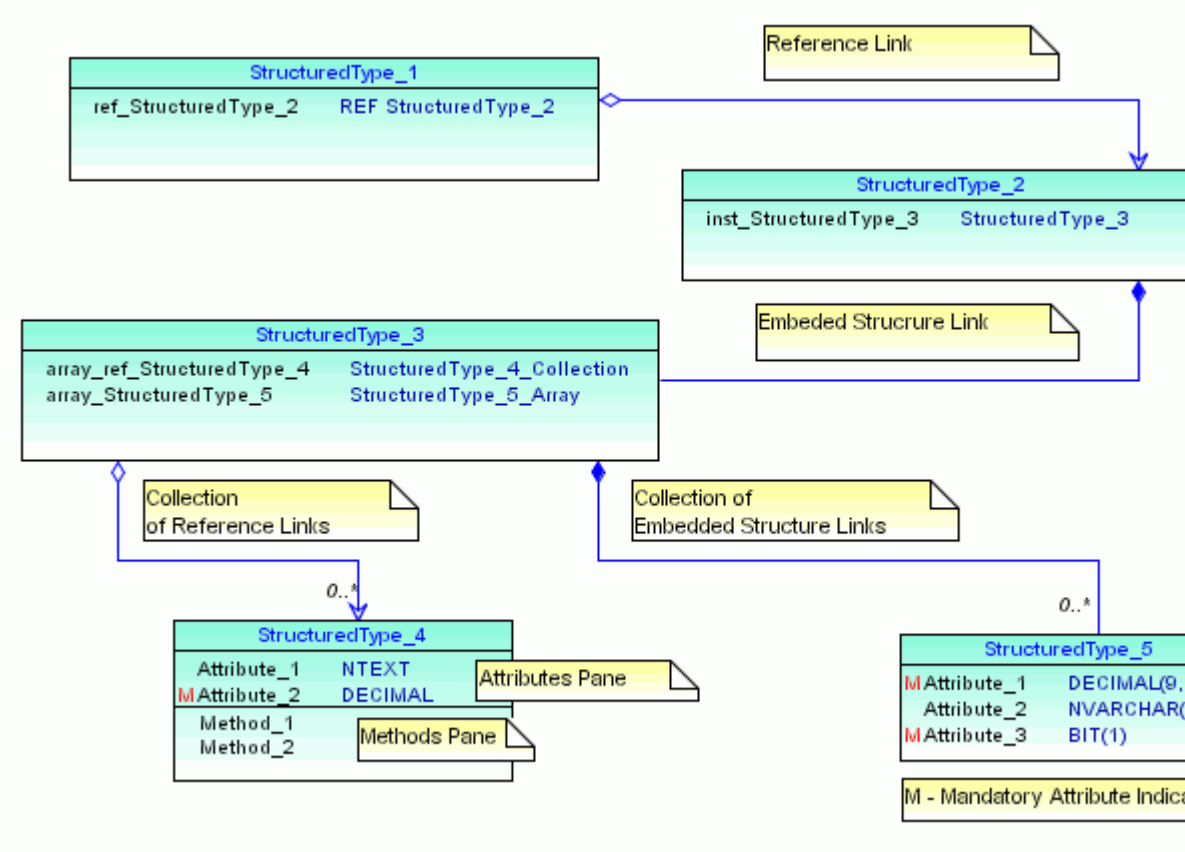
Subviews are considered as independent diagrams of the data types model, created to represent different subject areas.

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 (page 3-101) 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 that 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 and Graph 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 Types Administration (page 3-101)), and you can associate a domain with a logical type (see Domains Administration (page 3-29)).

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 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**. The data flow model allows you to drill down to child processes through a composite process. This means that a top-level process can drill down to another full data flow model.
- 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 Data Flow Diagrams (page 1-16).)
- 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 Transformation Processes and Packages (page 1-16).)
- 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 Processes and Packages

In a general data flow diagram, you may want to extract data from external sources and then transform the data before loading the it into the target store or database. You can build transformation packages for use with transformation processes.

For a **transformation process**, you need to create one or more transformation tasks in a transformation package. After you have the transformation task, you can include that in the main transformation process.

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 (On-Line Analytical Processing), 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 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 Data Modeler
- By importing models from a VAR file, such as those created by at least these versions of the following products: Sterling COOL:DBA V2.1 or Sterling Bsnteam V7.2, Cayenne Bsnteam V7.2
- By importing an existing model created by 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**. You can also right-click in the subview and select **Add/Remove Elements** to add objects to the subview and remove objects from the subview (using the Add/Remove Objects (page 3-1) dialog box).

**Diagraming 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 (page 1-32) (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.

### 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.

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 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 Data Modeler
- By forward engineering from the logical model or a subview of the logical model

- By importing models from a VAR file, such as those created by at least these versions of the following products: Sterling COOL:DBA V2.1 or Sterling Bsnteam V7.2, Cayenne Bsnteam V7.2
- By importing an existing model created by 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, and optionally to associate certain relational model objects with database schemas. 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.

To create a subview containing specific tables, you can select the desired entities in the logical model diagram, right-click, and select **Create Subview from Selected**. You can also right-click in the subview and select **Add/Remove Elements** to add objects to the subview and remove objects from the subview (using the Add/Remove Objects (page 3-1) dialog box).

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.

### 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.

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 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 11*g* and Microsoft SQL Server 2005). You can also use the RDBMS Site Editor (page 3-68) 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.

When you export to a DDL file, you specify the physical model to be applied. The generated DDL statements include clauses and keywords appropriate for features specified in that physical model (for example, partitioning for one or more tables).

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 (Oracle Call Interface) 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.

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 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, email, 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 Contact Properties (page 3-12).)

### 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 Document Properties (page 3-28).)

### 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 Email Properties (page 3-30).)

### 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 Location Properties (page 3-55).)

### 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 Resource Locator Properties (page 3-71).)

### 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 Responsible Party Properties (page 3-71).)

### 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 Telephone Properties (page 3-98).)

**Related Topics**

Working with Data Modeler (page 1-10)

Data Modeler Concepts and Usage (page 1-1)

## 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 (page 1-27): for designing a new database
- Bottom-Up Modeling (page 1-29): 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 (page 1-30): 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, email 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 (page 3-66) 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 (page 3-39) 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 (page 3-50) 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 (page 3-42) 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 (page 3-32) 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 (page 3-69) dialog box.

   c. **Apply design rules** to the logical model. Click **Tools**, then **Design Rules**, and use the Design Rules (page 3-24) dialog box to check for and fix any violations of the design rules.

   d. **Forward engineer the logical model to a relational model.** Right-click the logical model in the navigator, then select **Engineer to Relational Model**, and use the Engineering (page 3-30) 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 click the Split Table button
   b. Merge tables. To merge a table into another table (removing the merged table), click the Merge Tables button. 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 me 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 (page 3-20) 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) (page 3-19)).
   - **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 click the Split Table button. Use the Split Table wizard to copy or move source foreign keys and columns to the target table (the new table to be created).

      To *merge* a table into another table (removing the merged table), click the Merge Table button. Then, in the relational model diagram, first click the table whose columns are to be merged into the other table, and next click the table

into which to merge the columns from the first table that you clicked. (After the merge, the first table that you clicked will be removed, and the remaining table will include its original columns plus the columns that had been in the first table.)

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 (page 3-24) dialog box.

4. **Reverse engineer the logical model from a relational model.** Click the Engineer to Logical Model icon, or right-click the relational model, then select **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 (page 3-20) dialog box, select the physical model and click **Generate**. Specify any desired DDL Generation Options (page 3-21), 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 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 Bottom-Up Modeling (page 1-29).

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 (page 3-32) 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 Engineer to Relational Model icon or by right-clicking the logical model in the navigator, then selecting **Engineer to Relational Model**.

3. In the Engineering (page 3-30) 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 (page 3-93) 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 Engineer to Logical Model icon or by right-clicking the relational model name in the navigator, then selecting **Engineer to Logical Model**.
3. In the Engineering (page 3-30) dialog box, specify any desired filtering, then click **Engineer**.

# 1.5 User Preferences for Data Modeler

You can customize many aspects of the Data Modeler environment and interface by modifying user preferences according to your personal wishes and needs. To modify the user preferences, select **Tools**, then **Preferences**.

**Search** box: You can enter a string to limit the tree display to matching relevant preference groups.

Most preferences are self-explanatory, and this topic explains only those whose meaning and implications are not obvious. Some preferences involve performance or system resource trade-offs (for example, enabling a feature that adds execution time), and other preferences involve only personal aesthetic taste. The preferences are grouped in the following categories:

- Environment (page 1-31)
- Data Modeler (page 1-32)
- Format (page 1-40)
- Global Ignore List (page 1-41)
- Mouse Actions (page 1-41)
- Shortcut Keys (Accelerator Keys) (page 1-42)
- SSH (Secure Shell) (page 1-43)
- Versioning (page 1-43)
- Web Browser and Proxy (page 1-44)

## 1.5.1 Environment

The Environment pane contains options that affect the startup and overall behavior and appearance of Data Modeler. You can specify that certain operations be performed automatically at specified times, with the trade-off usually being the extra

time for the operation as opposed to the possibility of problems if the operation is not performed automatically (for example, if you forget to perform it when you should).

For example, changes to the undo level (number of previous operations that can be undone) and navigation level (number of open files) values may cause slight increases or decreases in system resource usage.

**Automatically Reload Externally Modified Files**: If this option is checked, any files open in Data Modeler that have been modified by an external application are updated when you switch back to Data Modeler, overwriting any changes that you might have made. If this option is not checked, changes that you make in Data Modeler overwrite any changes that might have been made by external applications.

**Silently Reload When File Is Unmodified**: If this option is checked, you are not asked if you want to reload files that have been modified externally but not in Data Modeler. If this option is not checked, you are asked if you want to reload each file that has been modified externally, regardless of whether it has been modified in Data Modeler.

**Environment: Dockable Windows**

The Dockable Windows pane configures the behavior of dockable windows and the shapes of the four docking areas of Data Modeler: top, bottom, left, and right.

**Dockable Windows Always on Top**: If this option is checked, dockable windows always remain visible in front of other windows.

**Windows Layout**: Click the corner arrows to lengthen or shorten the shape of each docking area.

**Environment: Log**

The Log pane configures the colors of certain types of log messages and the saving of log messages to log files.

**Save Logs to File**: If this option is checked, all output to the Messages - Log window is saved to log files, where the file name reflects the operation and a timestamp. You are also asked to specify a **Log Directory**; and if the specified directory does not already exist, it is created. Note that if you save log information to files, the number of these files can become large.

**Maximum Log Lines**: The maximum number of lines to store in each log file.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.2 Data Modeler

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

**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.

**Default Save Directory**: The default directory or folder in which to save files.

**Default System Types Directory**: The default directory or folder for storing type definition files.

**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.

**Use OCI/Thick Driver**: For Oracle Database connections, controls whether the oci8 (thick) driver is used by default if it is available, instead of the JDBC (thin) driver.

**Reload Last State**: Controls whether the last open design is reloaded when Data Modeler is restarted. (Regardless of the setting of this option, you can see any recently open designs by clicking **File > Recent Designs**.)

**Import**: Lets you import Data Modeler preferences and other settings that had previously been exported, as explained in Exporting and Importing Preferences and Other Settings (page 1-47).

**Export**: Saves Data Modeler preferences and other settings to an XML file, so that you can later import the information, as explained in Exporting and Importing Preferences and Other Settings (page 1-47).

Other Data Modeler preferences are grouped into the following categories:

- DDL (page 1-33)
- Diagram (page 1-36)
- Model (page 1-36)
- Reports (page 1-39)
- Search (page 1-40)
- Third Party JDBC Drivers (page 1-40)

### 1.5.2.1 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 Arc Constraints**: 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 (Updatable) option in the Foreign Key Properties (page 3-43) 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.

**Extended Size for Characters for Oracle**: Controls whether the behavior of the MAX_STRING_SIZE = EXTENDED initialization parameter is available when generating DDL: that is, the size limit is 32767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. A VARCHAR2 or NVARCHAR2 data type with a declared size of greater than 4000 bytes, or a RAW data type with a declared size of greater than 2000 bytes, is an extended data type. Extended data type columns are stored out-of-line, leveraging Oracle's LOB technology. For more information, see the "Extended Data Types" section of the "Data Types" chapter in *Oracle Database SQL Language Reference*.

**Generate Short Form of NOT NULL Constraint**: Controls whether to use a `NOT NULL` constraint name in column definitions (in CREATE TABLE statements).

**Use Quoted Identifiers**: Controls whether object names are enclosed in double quotes in the generated DDL statements.

**Replace System Names During Import**: Controls whether constraint names originally assigned by Oracle Database (such as starting with *SYS_*) are kept during import operations or are replaced by names assigned by Data Modeler (such as starting with *T_PK* for a primary key). (For more information, see `http://www.thatjeffsmith.com/ask-a-question/#comment-49000`.)

**Create Domains During Import**: Controls whether domains are created from data types during import operations.

**Generate Comments in RDBMS**: Controls whether Comment in RDBMS text is included in the generated DDL statements

**Generate Inline Column Check Constraints**: Controls whether the column check constraint clause is included inline (in the CREATE TABLE statement) or as separate constraint definition (in an ALTER TABLE statement).

**Include Default Settings in DDL**: Controls whether default keywords are included in the generated DDL statements when you have not specified a corresponding setting. This option is useful if you want to see every keyword that will be used in the generated statements.

**Include Logging in DDL**: Controls whether logging information is included in the generated DDL statements.

**Include Schema in DDL**: Controls whether object names are prefixed with the schema name (for example, `SCOTT.EMP` as opposed to just `EMP`) in the generated DDL statements.

**Include Storage in DDL**: Controls whether storage information is included in the generated DDL statements.

**Include Tablespace in DDL**: Controls whether tablespace information is included in the generated DDL statements.

**Include Redaction in DDL**: Controls whether data redaction information is included in the generated DDL statements.(You should understand the concepts and techniques

for redaction, as explained in the "Using Transparent Sensitive Data Protection" chapter in *Oracle Database Security Guide*.

**Include Sensitive Data Protection in DDL**: Controls whether information about sensitive data is included in the generated DDL statements. (You should understand the concepts and techniques for redaction and sensitive data protection, as explained in the "Using Transparent Sensitive Data Protection" chapter in *Oracle Database Security Guide*.

**Include PROMPT Command (for Oracle Only)**: Controls whether a PROMPT command is included before each DDL statement in the generated DDL statements for an Oracle database. Each PROMPT command displays an informational message about the next statement, enabling someone viewing the output of script execution to follow the progress.

**SQL Formatting: Use SQL Developer Formatter**: Controls whether SQL formatting uses the SQL Developer defaults or the traditional Data Modeler defaults. (You can try DDL generation with the option on and off to see which DDL output better suits your personal preference.)

**Default DDL Files Export Directory**: Default directory in which generated DDL files are placed for export operations.

### DDL: DDL/Comparison

Lets you specify things to be considered or ignored in comparisons.

**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)).

**Use 'Schema' Property in Compare Functionality**: Controls whether the schema name associated with an object should be considered when comparing two objects.

**Use 'Columns Order' Property in Compare Functionality**: Controls whether the order of columns should be considered when comparing two tables.

**Case Sensitive Names in Compare Functionality**: Controls whether the letter case in object names should be considered when comparing two objects.

**Include System Names in Compare Functionality**: Controls whether system-generated constraint names (generated when a constraint name was not provided in the CREATE or ALTER statement) should be considered when comparing two constraint objects. (For a given constraint definition, the constraint names generated by two different systems will almost certainly be different.)

### DDL: DDL/Storage

Lets you specify storage options to be included in DDL for import and export operations. These options affect which clauses and keywords are included.

Including storage options provides produces more detailed DDL statements; this makes it clear exactly what options are in effect, and enables you to edit them in the DDL if you want. Excluding storage options produces more concise DDL statements, which you may find more readable.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

### 1.5.2.2 Diagram

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

**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: Grid Size**: Controls the relative distance between grid points when the grid is shown. (The point units reflect an internal coordinate system used by Data Modeler.)

**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.

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

**Diagram: Relational Model**

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

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

**Related Topics**

User Preferences for Data Modeler (page 1-31)

### 1.5.2.3 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).

**Data Type Compatibility for Foreign Keys: Allow Similar Data Types for Foreign Keys**: Controls whether, in a foreign key link, the data types of the two columns must be the same (strict match) or must merely be compatible. If this option is enabled, the data types can be the same or compatible; if it is disabled, the data types must be the same. For example, if one column is Number and the other is Integer, they are compatible; and if character data columns have different maximum lengths, they are compatible.

**Preferred Domains and Logical Types**: Enables you to limit the values displayed in drop-down lists of domains and logical types. (You can use this feature to prevent such lists from being "cluttered" with domains and logical types that you never specify.) To have a domain or logical type appear in drop-down lists, move it from the Preferred side to the All side.

### 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.

**Primary Key Option for Identifying Relationships: 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.

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

**FK Attribute Name Synchronization: Comments, Notes - Automatically propagate from PK attribute**: Controls whether to inherit comments and notes from the referenced primary key attribute in the definition of foreign key attributes.

**Default Surrogate Key Settings: Entity Create Surrogate Key**: Controls whether a surrogate primary key is created for entities. If this option is enabled, then by default when a new entity is created, a table will get a surrogate primary key when it is set to a related entity, when it is set to a relationship to use a surrogate key, or when the entity does not have a primary key and a relationship refers to that entity.

**Default Surrogate Key Settings: Relationship Use Surrogate Key**: If this option is enabled, then by default when a new relationship is created, it is set to use a surrogate key (and foreign key attributes are not maintained), rather than being bound to a specific unique identifier.

### Model: Physical

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

For **Oracle**, you can specify:

- Defaults for User and Tablespace for the table
- Whether to use a Table Template and/or Index Template; and if so, properties of the template to be used when generating tables and indexes.
- Autoincrement column template options: Names for triggers and sequences (and optionally add variables in the names), and the DDL implementation methods for Auto Increment and IDENTITY.

For information about Oracle Database table and index properties, see the `CREATE TABLE` and `CREATE INDEX` statements in *Oracle Database SQL Language Reference*.

For **DB2**, you can specify a Naming Rule for several options, and can add a variable to any naming rule string that you enter. For example, for **Create TableSpace for Each Table**, you might type `TBLS_` and then click **Add Variable** and select `{model}`, causing

each creation or renaming of a table to create or rename a corresponding tablespace to TBLS_ plus the table name.

**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 Data Modeler Tutorial: Modeling for a Small Database (page 2-1), 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.

**Allow Columns Reorder During Engineering**: If this option is enabled, Data Modeler can reorder the attributes of the associated entity when the table is engineered to the relational model, for example, to place attributes considered more important first. (This behavior can be especially useful with tables that contain many columns.) If this option is not enabled, entity attributes are placed in the same order as their associated columns in the table definition.

**Synchronize Remote Objects When Model Is Loaded**: If this option is enabled, then when a remote object (object in another model) is dropped into the diagram of a current model, the representation of the remote object is changed (synchronized) whenever the model is loaded to reflect any changes that may have been made to the remote object. If this option is not enabled, the representation of the remote object is not changed in the diagram.

**Surrogate Column Data Type**: Data type for the surrogate column (unique key that is not the primary key), if a table has a surrogate key.

**Database Synchronization: Use Source Connection**: If this option is enabled, then if the source and destination connections have different names, the source connection by default is used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Use Source Schema**: If this option is enabled, then if the source and destination schemas have different names, the source schema by default is used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Use Source Object**: If this option is enabled, then if the source and destination objects have different names, the source object by default is

used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Synchronize the Whole Schema**: If this option is enabled, then objects that exist in the database but not in the model will appear as new objects when the model is synchronized with database, and will appear as candidates to be deleted from the database when the database is synchronized with the model. If this option is not enabled, only objects that exist in the model are synchronized with database.

**Model: Synchronization Physical**

Contains options that apply to synchronizing a physical model of a specified database type with its associated relational model.

For each specified type of object, you can specify whether to synchronize it (that is, whether to have changes in the relational model to objects of the specified types be applied automatically in the associated physical models).

**Related Topics**

User Preferences for Data Modeler (page 1-31)

### 1.5.2.4 Reports

The Reports pane contains options that apply when generating reports.

**Company name**: Name of the company to appear in the report.

**Insert page break between objects**: Inserts a page break between objects in the report.

**Embed diagrams (HTML and PDF)**: Embeds all diagrams into the report. Click on an object in a diagram to view its details in the report.

**Embed main diagram (HTML and PDF)**: Embeds only the main diagram into the report.

**Generate HTML report TOC in separate file**: Generates more than one file such that the table of contents is a separate file. This option is only available for reports in the HTML format.

**Include select statement in views reports**: Includes the SQL statement used for the query to generate a report based on views.

**Group Objects by Schema**: Displays the objects in groups based on the schema in the left pane. If this option is not enabled, the objects are displayed in alphabetical order.

**Default Reports Directory**: The default directory or folder in which to generate Data Modeler Reports (page 1-48). If you do not specify a directory, the default is in `datamodeler/reports` or `datamodeler\reports` under the location where you installed Data Modeler. For example, on a Windows system this might be `C:\Program Files \datamodeler\datamodeler\reports`.

**Path to Saxon XSLT 2.0 Jar File**: Path to the Saxon 2.0 XSLT processor, if you have downloaded the file and want Data Modeler to use it for generating reports (for example: `C:\saxon9.3\saxon9he.jar`). If you do not specify this option, Data Modeler uses XSLT 1.0 in report generation. In general, Saxon can handle the generation of

much larger reports than XSLT 1.0, especially for PDF. (To read about and download the Saxon XSLT processor, go to `http://saxon.sourceforge.net/`.)

### 1.5.2.5 Search

The Search pane contains options that affect the behavior when you use the Find Object (Search) (page 3-41) feature

**Press Enter to Search**: Returns matches (reflecting the search string that you have typed so far) when you press Enter.

**Search as You Type**: Returns matches after you type the first specified number of characters (symbols). If you continue typing, the matches are automatically updated for each additional character that you type.

**Number of initial symbols to ignore**: For Search as Your Type, specifies how many characters you can type before possible matches are displayed. (The higher the number, the shorter the list of initial possible matches will probably be for a search; you might find this more convenient and less distracting as you type.)

**Search Profiles**: You can add and edit search profiles, which let you limit the set of properties of relational and logical model objects to be considered when searching for objects, thus potentially resulting in a shorter, more meaningful set of matches. Clicking the Add icon or double-clicking an existing profile name displays the Search Profile (page 3-73) dialog box.

**Related Topics**

Find Object (Search) (page 3-41)

User Preferences for Data Modeler (page 1-31)

### 1.5.2.6 Third Party JDBC Drivers

The Third Party JDBC Drivers pane specifies drivers to be used for connections to third-party (non-Oracle) databases. 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: `http://msdn.microsoft.com/en-us/data/aa937724.aspx`
- For IBM DB2/LUW, the IBM Data Server Driver for JDBC and SQLJ at: `http://www-01.ibm.com/software/data/db2/linux-unix-windows/downloads.html`

For each driver to be added, click the Add (+) icon and select the path for the driver.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.3 Format

The Format pane controls how statements are formatted in SQL scripts that are generated. The options include whether to insert space characters or tab characters when you press the Tab key (and how many characters), uppercase or lowercase for

keywords and identifiers, whether to preserve or eliminate empty lines, and whether comparable items should be placed on the same line or on separate lines.

The Advanced Format subpane lets you specify more detailed formatting options. It also includes these options:

- **Preview with Current Settings**: You specify changes on the left side, and the preview area on the right side reflects the changes.
- **Auto-Detect Formatter Settings**: You paste code with the desired formatting into the preview pane on the right side, and SQL Developer adjusts the settings on the left side to reflect what you pasted. (It automatically detects, or autodetects, your setting preferences.)

You can export these settings to a code style profile XML file, and can import settings from a previously exported code style profile file.

The Custom Format subpane lets you further customize your formatting settings. You can also export these settings to a custom formatter program XML file, and you can import settings from a previously exported custom formatter program file.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.4 Global Ignore List

The Global Ignore List pane specifies filters that determine which files and file types will not be used in any processing.

**New Filter**: A file name or file type that you want to add to the list of files and file types (in the Filter box) that Data Modeler will ignore during all processing (if the filter is enabled, or checked). You can exclude a particular file by entering its complete file name, such as `mumble.txt`, or you can exclude all files of the same type by entering a construct that describes the file type, such as `*.txt`.

**Add**: Adds the new filter to the list in the Filter box.

**Remove**: Deletes the selected filter from the list in the Filter box.

**Restore Defaults**: Restores the contents of the Filter box to the Data Modeler defaults.

**Filter**: Contains the list of files and file types. For each item, if it is enabled (checked), the filter is enforced and the file or file type is ignored by Data Modeler; but if it is disabled (unchecked), the filter is not enforced.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.5 Mouse Actions

The Mouse Actions pane specifies text to be displayed on hover-related mouse actions over relevant object names.

**Popup Name**: The type of information to be displayed: **Data Values** (value of the item under the mouse pointer, such as the value of a variable), **Documentation** (documentation on the item under the mouse pointer, such as Javadoc on a method

call), or **Source** (source code of the item under the mouse pointer, such as the source code of a method).

**Activate Via**: Use action with the mouse cursor to activate the display: Hover, or Hover while pressing one or two specified modifier keys.

**Description**: Description of the associated Popup Name entry.

**Smart Enabled**: If this option is checked, then the text for the relevant type of information is displayed if Smart Popup is also checked.

**Smart Popup**: If this option is checked, the relevant text for the first smart-enabled popup is displayed for the item under the mouse pointer.

**Related Topics**

Data Modeler (page 1-32)

## 1.5.6 Shortcut Keys (Accelerator Keys)

The Shortcut Keys pane enables you to view and customize the shortcut key (also called accelerator key) mappings for Data Modeler.

**Hide Unmapped Commands**: If this option is checked, only shortcut keys with mappings are displayed.

**More Actions**:

- **Export**: Exports the shortcut key definitions to an XML file.
- **Import**: Imports the shortcut key definitions from a previously exported XML file.
- **Load Keyboard Scheme**: Drops all current shortcut key mappings and sets the mappings in the specified scheme. (This option was called Load Preset in previous releases.) *If you have made changes to the mappings and want to restore the default settings, select* ***Default***.

**Category**: Lists commands and shortcuts grouped by specific categories (Code Editor, Compare, and so on), to control which actions are displayed.

**Command**: An action relevant to the specified category. When you select an action, any existing shortcut key mappings are displayed.

**Shortcut**: Any existing key mappings for the selected action. To remove an existing key mapping, select it and click Remove.

**New Shortcut**: The new shortcut key to be associated with the action. Press and hold the desired modifier key, then press the other key. For example, to associate Ctrl+J with an action, press and hold the Ctrl key, then press the j key. If any actions are currently associated with that shortcut key, they are listed in the Current Assignment box.

**Conflicts**: A read-only display of the current action, if any, that is mapped to the shortcut key that you specified in the New Shortcut box.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.7 SSH (Secure Shell)

SSH preferences are related to creating SSH (Secure Shell) connections.

**Use Known Hosts File**: If this option is checked, specify the file of known hosts to be used.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.8 Usage Reporting

In SQL Developer, Data Modeler, and some other applications, the Usage Reporting user preference and a related dialog box ask for your consent to Oracle usage reporting. If you consent, automated reports can occasionally be sent to Oracle describing the product features in use. No personally identifiable information will be sent and the report will not affect performance. You can review Oracle's privacy policy by clicking the **privacy policy** link.

**Allow automated usage reporting to Oracle**: Determines whether you consent to usage reporting.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

## 1.5.9 Versioning

Versioning preferences affect the behavior of the version control and management systems that you can use with Data Modeler. For information about using versioning with Data Modeler, see Using Versioning (page 1-53).

### Versioning: Subversion

The Subversion pane specifies the Subversion client to use with Data Modeler.

### Versioning: Subversion: Comment Templates

The Subversion: Comment Templates pane specifies templates for comments to be used with commit operations. For example, a template might contain text like the following:

```
Problem Description (with bug ID if any):
Fix Description:
```

You can add, edit, and remove comment templates, and you can export templates to an XML file or import templates that had previously been exported.

### Versioning: Subversion: General

The Subversion: General pane specifies environment settings and the operation timeout.

**Use Navigator State Overlay Icons**: If this option is enabled, state overlay icons are used. State overlay icons are small symbols associated with object names in the

navigators. They indicate the state of version-controlled files (for example, "up to date").

**Use Navigator State Overlay Labels**: If this option is enabled, state overlay labels are used. State overlay labels are tooltips associated with object names in the navigators.

**Automatically Add New Files on Committing Working Copy**: If this option is enabled, any new files you have created in your working copy are automatically added to the Subversion repository whenever you commit any individual file. Otherwise, Subversion will not add new files when you commit changes; you must continue to add new files to Subversion explicitly.

**Automatically Lock Files with svn:needs-lock Property After Checkout**: If this option is enabled, files you check out from the repository are automatically locked, preventing other team members from checking them out until you release the files.

**Use Merge Wizard for Subversion Merging**: If this option is enabled, the Merge wizard rather than the Merge dialog box is invoked for merge requests.

**Operation Timeout**: Maximum number of seconds, minutes, or hours allowed for Subversion operations to complete.

**Edit Subversion Configuration File**: To modify the Subversion file directly, click **Edit "server"**. You can make changes to server-specific protocol parameters such as proxy host, proxy port, timeout, compression, and other values. Lines beginning with # are interpreted as comments.

#### Versioning: Subversion: Version Tools

The Subversion: Version Tools pane specifies options for the pending changes window and the merge editor.

**Use Outgoing Changes Commit Dialog**: Enables you to make optimum use of limited screen space when the Pending Changes window is open. You can save screen space by not showing the Comments area of the Pending Changes window, but you might still want to add comments before a commit action. You can choose the circumstances under which the Commit dialog is opened: always, only when the Comments area of the Pending Changes window is hidden, or never.

**Incoming Changes Timer Interval**: The frequency at which the change status of files is checked.

**Merge Editor**: Specifies whether files are merged locally or at the server.

**Related Topics**

Using Versioning (page 1-53)

Data Modeler (page 1-32)

## 1.5.10 Web Browser and Proxy

The Web Browser and Proxy settings are relevant only when you use the Check for Updates feature (click **Help**, then **Check for Updates**), and only if your system is behind a firewall.

**Web Browsers**

Displays the available web browsers and the default browser for Check for Update operations. You can click under Default to change the default browser.

For each browser, you can determine whether it is the default, and you can see and optionally change its name , the path to the application's executable file, application command parameters, and the icon.

**Proxy Settings**

You can choose no proxy, system default proxy settings, or manually specified proxy settings for Check for Update operations. For manually specified settings, check your Web browser options or preferences for the appropriate values for these fields.

**Internet Files**

You can choose whether to enable Internet cookies for Check for Update operations.

**Clear All Cookies**: Clears all existing cookies.

**Related Topics**

User Preferences for Data Modeler (page 1-31)

# 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 Database Design (page 1-11)) are created for a new design or updated for the existing design, which is stored in 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 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 11*g*).

To use a design that had been saved by 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 following topics contain information about importing from specific types of files and other sources.

**Related Topics**

Importing a DDL File (page 1-46)

Importing Cube Views Metadata (page 1-46)

Importing from Microsoft XMLA (page 1-46)

Importing an ERwin File (page 1-46)

Importing from a Data Dictionary (page 1-47)

Importing an Oracle Designer Model (page 1-47)

Importing a Data Modeler Design (page 1-47)

Importing a Domain (page 1-47)

Select Models/Subviews to Export (page 3-74) (for Export > To Data Modeler Design)

## 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. You can create a connection to an Oracle Designer repository and import the entities, tables, and domains from a workspace in Designer.

In the Import Oracle Designer Model (page 3-48) wizard, 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 Data Modeler Design

Importing a Data Modeler design enables you to capture the logical model and any relational and data type models from a design previously exported from 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 (page 3-47) dialog box, select the domains to be imported and deselect (clear) the domains not to be imported.

## 1.7 Exporting and Importing Preferences and Other Settings

You can export and import the following Data Modeler information:

- User preferences (see User Preferences for Data Modeler (page 1-31))
- Connections, such as ones you create for importing from a data dictionary (see Data Dictionary Import (Metadata Extraction) (page 3-19))
- Recent designs (shown if you click **File**, then **Recent Designs**)

To export this information, click **Tools**, then **Preferences**, then **Data Modeler (page 1-32)**, and click **Export**. Be sure to specify a directory or folder that is not under the location where you installed Data Modeler, because the file will be lost if you later delete the current installation.

To import information that was previously exported, click **Tools**, then **Preferences**, then **Data Modeler (page 1-32)**, and click **Import**.

## 1.7.1 Restoring the Original Data Modeler Preferences

If you have made changes to Data Modeler preferences but want to restore all to their original default values, you can follow these steps:

1. If you are running Data Modeler, exit.
2. Delete the folder or directory where your Data Modeler user information is stored. The default location is a build-specific directory or folder under the following:
   - Windows: C:\Documents and Settings\<*user-name*>\Application Data\Oracle SQL Developer Data Modeler
   - Linux or Mac OS X: ~/.datamodeler

   For example, if the current build-specific folder is named `system3.1.0.678`, delete `C:\Documents and Settings\<user-name>\Application Data\Oracle SQL Developer Data Modeler\system3.1.0.678`.
3. Start Data Modeler.

   This creates a folder or directory where your user information is stored (explained in step 2), which has the same content as when Data Modeler was installed.

If you have made changes to the shortcut key (accelerator key) mappings, you can restore the mappings to the defaults for your system by clicking **Tools**, then **Preferences**, then **Shortcut Keys**, then **More Actions**, then **Load Keyboard Scheme**, and then selecting **Default**.

# 1.8 Data Modeler Reports

You can view reports on Data Modeler objects in the following ways:

- Generate reports as RTF, HTML, or PDF files on your local drive; they are also opened automatically. For more information, see Generating Reports as RTF, HTML, or PDF Files (page 1-48).
- Use the Data Modeler reporting repository, and use SQL Developer to view exported reports. For more information, see Using the Reporting Repository and Reporting Schema (page 1-50) and Using SQL Developer to View Exported Reporting Schema Data (page 1-51).

## 1.8.1 Generating Reports as RTF, HTML, or PDF Files

You can save individual reports as RTF (a Microsoft Word format), HTML, or PDF files, and view each report when it is opened automatically on creation and open the files for viewing later. For HTML, several separate files are generated. The files are stored in the location specified or defaulted for **Default Reports Directory** under Data Modeler (page 1-32) preferences.

Data Modeler ensures unique names for each file; for example, if you generate a report on all tables and if `AllTablesDetails_1.rtf` already exists, `AllTablesDetails_2.rtf` is created. (If you generate report files from the reporting repository in the reporting schema, the file names include *_rs*, for example, `AllTablesDetails_1_rs.rtf`.)

You can generate report files using either of the following approaches:

- Generate reports based on currently loaded designs. (This approach does not involve creating or using a reporting schema and reporting repository.)
- Generate reports based on information in the reporting repository in the reporting schema (which are explained in Using the Reporting Repository and Reporting Schema (page 1-50)).

To generate and view a report stored in RTF, HTML, PDF, XLS, or XLSX format, follow these steps:

1. Click **File**, then **Reports**.
2. In the **General** tab, for **Available Reports**, select one of the types of objects for which to report information: Tables, Entities, Domains, Glossaries, and so on.
3. In **Output Format**, select HTML, PDF, or RTF for standard templates and HTML, PDF, XLS, or XLSX for custom templates.
4. In **JVM Options**, specify the memory allocation for PDF reports. The default value is -Xmx768M.
5. For **Report Title**, enter a name for the report.
6. For **Report File Name**, enter a name for the report file.
7. For **Options**, see Reports (page 1-39).
8. Under **Templates**, optionally, select a report template to use. You can use a report template to customize the types of objects to be included in a report. The following tabs are available:
   - **Standard**: Leave blank to use the default standard report format for the report type, or select an available modified standard format (if any exist). To create and manage standard report types, click **Manage** to display the Report Templates Management (page 3-70) dialog box.
   - **Custom**: Lets you specify highly customized report formats. To create a new customized report format or to edit a selected existing one, click **Manage** to display the Report Templates Management (page 3-70) dialog box.

     For Custom reports, **Replace Boolean Values** let you select nondefault True and False values.
9. Click the **Objects** tab.
10. Click one of the following tabs (if the desired tab is not already selected):
    - **Loaded Designs**, to generate a report based on one or more currently loaded Data Modeler designs
    - **Reporting Schema**, to generate a report based on designs in the reporting repository in the reporting schema
11. For **Available Designs**, select the desired Data Modeler design.
12. For **Available Models,** select the desired model. (The list of models reflects the type of objects for the report.)
13. For **Report Configurations**, you can select a configuration to limit the report to specific subviews or specific objects of the selected Available Report type, or you can leave it blank to select all subviews (if any) and all objects of the selected Available Report type.

You can add, delete, or modify configurations by clicking Manage to display the Standard Reports Configurations (page 3-79) dialog box.

14. Click **Generate Report**.

    A message is displayed with the location and name of the file for the report.

15. Go to the file and open it.

**Related Topics**

Data Modeler Reports (page 1-48)

## 1.8.2 Using the Reporting Repository and Reporting Schema

The Data Modeler **reporting repository** is a collection of database schema objects for storing metadata and data about Data Modeler designs. The schema in which the reporting repository is stored is called the **reporting schema**.

To set up the Data Modeler reporting schema, perform these steps outside the Data Modeler interface:

1. Create or specify a database schema to hold the reporting repository.

   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.)

   If you want to continue using a reporting repository from an earlier version of Data Modeler, see the `Reporting_Schema_Upgrade_readme.txt` file in the `datamodeler\datamodeler\reports` folder.

2. Edit and run the `Reporting_Schema_Permissions.sql` script file, which is located in the `datamodeler\datamodeler\reports` folder.

   Before running the script, edit the file to replace *<USER>* and *<OS DIRECTORY>* with desired values:

   - *<USER>* is the schema to hold the reporting repository.
   - *<OS DIRECTORY>* is a temporary file system folder (directory) on the computer where the database is running. The script will create the definition in the database, but you must create this folder after you run the script.

Then, start Data Modeler and perform these steps:

1. Click **File**, then **Export**, then **To Reporting Schema**.
2. In the Export to Reporting Schema (page 3-36) dialog box, click the Add Connection (+) icon.
3. In the New/Update Database Connection (page 3-64) dialog box, enter a name for the connection (for example, `dm_reporting_repos_conn`), as well as the other information for the connection , including the user name and password for the database user associated with the reporting schema.
4. Optionally, click **Test** to test the connection. (If the test is not successful, correct any errors.)

5. Click **OK** to create the connection and to close the New/Update Database Connection (page 3-64) dialog box.
6. Select (click) the connection name in the list of connections near the top of the dialog box.
7. Click **OK** to create the reporting repository in the schema associated with the selected connection, and to have the information about the selected models exported to that repository.

To see the Data Modeler reports, use SQL Developer, as explained in Using SQL Developer to View Exported Reporting Schema Data (page 1-51).

To delete an existing reporting repository, follow these steps in Data Modeler:

1. Click **File**, then **Export**, then **To Reporting Schema**.
2. Select the connection for the schema associated with the reporting repository to be deleted.
3. In the Export to Reporting Schema (page 3-36) dialog box, click the **Maintenance** tab.
4. Click **Drop Repository**, then confirm that you want to drop the reporting repository.

   If you only want to deleted selected designs within the repository and not the entire repository, click **Delete Designs** and select the designs to be deleted.

For glossaries, you can perform the following operations using the **Glossary** tab of the Export to Reporting Schema (page 3-36) dialog box:

- **Export Glossary**: Enables you to specify a Data Modeler glossary file, to have its information exported to the reporting repository.
- **Delete Glossary**: Enables you to select a glossary in the reporting repository, to have its information deleted from the repository.

> **Note:**
>
> `datamodeler\datamodeler\reports\Reports_Info.txt` for additional technical details about Data Modeler reports.

**Related Topics**

Data Modeler Reports (page 1-48)

## 1.8.3 Using SQL Developer to View Exported Reporting Schema Data

You can use the reports feature in Oracle SQL Developer to view information that has been exported to the Data Modeler reporting repository. To export the information about a design to the reporting repository, follow the instructions in Using the Reporting Repository and Reporting Schema (page 1-50).

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 reporting 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.

Data Modeler reports are grouped in the following categories:

Design Content reports (page 1-52) list information about the design content (objects in the design).

Design Rules reports (page 1-52) list information about the design rules as they apply to the logical and relational models.

## 1.8.3.1 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 (page 1-11).

**Logical Model**: Contains reports related to the Logical Model (page 1-16).

**Relational Model**: Contains reports related to the Relational Models (page 1-19).

**Related Topics**

Data Modeler Reports (page 1-48)

## 1.8.3.2 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 (page 3-24) dialog box.)

**Logical Model**: Contains reports related to the Logical Model (page 1-16).

**Relational Model**: Contains reports related to the Relational Models (page 1-19).

**Related Topics**

Data Modeler Reports (page 1-48)

# 1.9 Using Versioning

Data Modeler provides integrated support for using the Subversion versioning and source control system with Data Modeler designs. You can store designs in a Subversion repository to achieve the usual version control benefits, including:

- Storing the "official" versions of designs in a central repository instead of in various folders or directories.
- Enabling multiple developers to work on the same design, coordinating their changes through the traditional Subversion checkout and commit processes.

The Data Modeler documentation does not provide detailed information about SVN concepts and operations; it assumes that you know them or can read about them. For information about Subversion, see `http://subversion.tigris.org/`. For Subversion documentation, see `http://svnbook.red-bean.com/`.

To access the versioning features of Data Modeler, use the Team menu (page 1-6).

If you create any versioning repositories or connect to any existing repositories, you can use the hierarchical display of repositories and their contents in the Versions navigator. (If that navigator is not visible, click **Team**, then **Versions**.)

**Related Topics**

About Subversion and Data Modeler (page 1-53)

Basic Workflow: Using Subversion with a Design (page 1-54)

Data Modeler Concepts and Usage (page 1-1)

## 1.9.1 About Subversion and Data Modeler

Before you can work with a Subversion repository through Data Modeler, you must create a connection to it. When you create a local Subversion repository, a connection to it is automatically created, and this can be seen in the Versions navigator. You can subsequently edit the connection details.

Existing files must be imported into the Subversion repository to bring them under version control. Files are then checked out from the Subversion repository to a local folder known as the "Subversion working copy". Files created in Data Modeler must be stored in the Subversion working copy.

Files newly created within Data Modeler must be added to version control. Changed and new files are made available to other users by committing them to the Subversion repository. The Subversion working copy can be updated with the contents of the Subversion repository to incorporate changes made by other users.

### 1.9.1.1 Pending Changes

The Pending Changes window is displayed if you click View, then Pending Changes, or when you initiate an action that changes the local source control status of a file.

This window shows files that have been added, modified or removed (locally or remotely), files whose content conflicts with other versions of the same file files that have not been added to source control files that are being watched, and files for which editors have been obtained. You can use this information to detect conflicts and to resolve them where possible.

The Outgoing Changes pane shows changes made locally, the Incoming Changes pane shows changes made remotely, and the Candidates pane shows files that have been created locally but not yet added to source control. You can double-click file names to edit them, and you can use the context menu to perform available operations.

## 1.9.2 Basic Workflow: Using Subversion with a Design

To use Subversion with a Data Modeler design, you must have the following:

- A folder or directory on your local system to serve as the working directory for the design. You create the design in this working directory, save the design to this working directory, and open the design from this working directory.
- A Subversion repository to which you can connect, and in which you can create under `branches` a branch for the initial version of the design (and later any subsequent versions).

The following are suggested basic steps. They are not the only possible steps or necessarily the "best" steps for a given project. These steps reflect the use of the Versions navigator and the Import wizard within Data Modeler to perform many actions; however, many actions can alternatively be performed using a separate SVN repository browser (such as the TortoiseSVN browser) and using SVN commands on your local system.

1. On your local system, create a directory or folder to serve as the parent for design-specific working directories. For example, on a Windows PC create:

   ```
   C:\designs
   ```

2. On your local system, create a directory or folder under the one in the preceding step to serve as the working directory for the design you plan to create. For example, for a design to be named `library`, create:

   ```
   C:\designs\library
   ```

3. In Data Modeler, create the design (for example, the library design in Data Modeler Tutorial: Modeling for a Small Database (page 2-1)), and save the design to the working directory that you created. For example, save the design to:

   ```
   C:\designs\library
   ```

   Saving the design causes the `.dmd` file and the related directory structure to be created in the working directory. (The `.dmd` file and the directory structure are explained in Database Design (page 1-11).)

4. Close the design. (Do not exit Data Modeler.)
5. Create an SVN connection to the repository that you want to use.
   a. In the Versions navigator, right-click the top-level node (Subversion) and select **New Repository Connection**.

   b. In the Subversion: Create/Edit Subversion Connection (page 3-84) dialog box, complete the information. Example repository URL: `https://example.com/svn/designs/`

6. Create a `branches` directory under the repository path.
   a. In the Versions navigator, right-click the repository path and select **New Remote Directory**.
   b. In the Subversion: Create Remote Directory (page 3-84) dialog box, complete the information, specifying the Directory Name as `branches`.
7. Create a project-specific branch under the `branches` directory.
   a. In the Versions navigator, right-click the `branches` directory and select **New Remote Directory**.
   b. In the Subversion: Create Remote Directory (page 3-84) dialog box, complete the information. Example Directory Name: `library`

      For example, if you plan to create the library design in Data Modeler Tutorial: Modeling for a Small Database (page 2-1), the URL in the repository for this branch might be:

      `https://example.com/svn/designs/branches/library`
8. Use the Subversion: Import to Subversion (page 3-87) wizard to import the design files into the repository. Click **Team**, then **Import Files**, and complete the wizard pages as follows.
   a. **Destination**: Specify the SVN connection and the repository path into which to import the files. Example: `root/branches/library`
   b. **Source**: Specify the source directory from which to import the files (that is, the directory containing the .dmd file and the design-specific folder hierarchy). Example: `C:\designs\library`
   c. **Filters**: Accept the defaults and click **Next**.
   d. **Options**: Accept the defaults and click **Next**.
   e. **Summary**: View the information and click **Finish**.

      The SVN Console Log shows the progress as files are added. After the files are added, the Handle New Files dialog box is displayed.
   f. In the Handle New Files dialog box, select **Do Not Open Files** and click **OK**.
   g. To see the files that have been added, click the Refresh icon in the Versions navigator tab.

For subsequent work on the design, follow the usual workflow for Subversion-based projects (SVN Update, SVN Lock, modify files, SVN Commit).

## 1.10 For More Information About Data Modeling

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

- Data Modeler Start Page, which contains links for tutorials, online demonstrations, documentation, and other resources. This page has two tabs: Get Started and Community. (If the Start Page tab is not visible, click **Help**, then **Start Page**).

- 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/technetwork/developer-tools/sql-developer/
- Object Management Group (OMG) site (http://www.omg.org/), especially the MetaObject Facility (MOF, http://www.omg.org/mof/) and Common Warehouse Metamodel (CWM, http://www.omg.org/spec/CWM/) specifications
- *United States Coast Guard Data Element Naming Standards Guidebook* (http://coastguardinstructions.tpub.com/CI_5230_42A/), especially concepts and recommendations relating to naming standards

**Related Topics**

Data Modeler Concepts and Usage (page 1-1)

# 2

# Data Modeler Tutorial: Modeling for a Small Database

In this tutorial, you will use 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 For More Information About Data Modeling (page 1-55).

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 (page 1-27) approach. (For information about the approaches, see Approaches to Data Modeling (page 1-27).)

You will perform the following major steps:

1. Develop the Logical Model (page 2-1).
2. Develop the Relational Model (page 2-6).
3. Generate DDL (page 2-6).
4. Save the Design (page 2-7).

**Related Topics**

Data Modeler Concepts and Usage (page 1-1)

Data Modeler User Interface (page 1-2)

## 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 Adding Domains (page 2-1).

### 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 (page 3-29) 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 `datamodeler/domains` directory or `datamodeler\domains` 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 Creating the Books Entity (page 2-2).

## 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 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 (page 3-32) dialog box is displayed.

4. Click **General** on the left, and specify as follows:

   **Name**: Books

5. 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 | (Author's first name; not mandatory, but enter it if the author has a first name.) |
| rating | Logical type: NUMERIC (Precision=2, Scale= 0) | (Librarian's personal rating of the book, from 1 (poor) to 10 (great).) |

6. Click **OK** to finish creating the Books entity.
7. Go to Creating the Patrons Entity (page 2-3).

## 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:

1. In the main area (right side) of the 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. (Suggestion: draw the box to the right of the Books box.) The Entity Properties (page 3-32) dialog box is displayed.
4. Click **General** on the left, and specify as follows:

   **Name**: Patrons

5. 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.) |
| street_address | Domain: Address Line | (Patron's street address.) |

| Attribute Name | Type | Other Information and Notes |
|---|---|---|
| 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 and Graph geometry object representing the patron's geocoded address. |

6. Click **OK** to finish creating the Patrons entity.
7. Go to Creating the Transactions Entity (page 2-4).

## 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 is 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:

1. In the main area (right side) of the 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. (Suggestion: Draw the box below and centered between the Books and Patrons boxes.) The Entity Properties (page 3-32) dialog box is displayed.
4. Click **General** on the left, and specify as follows:

   **Name**: Transactions
5. 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) |
| 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.) |

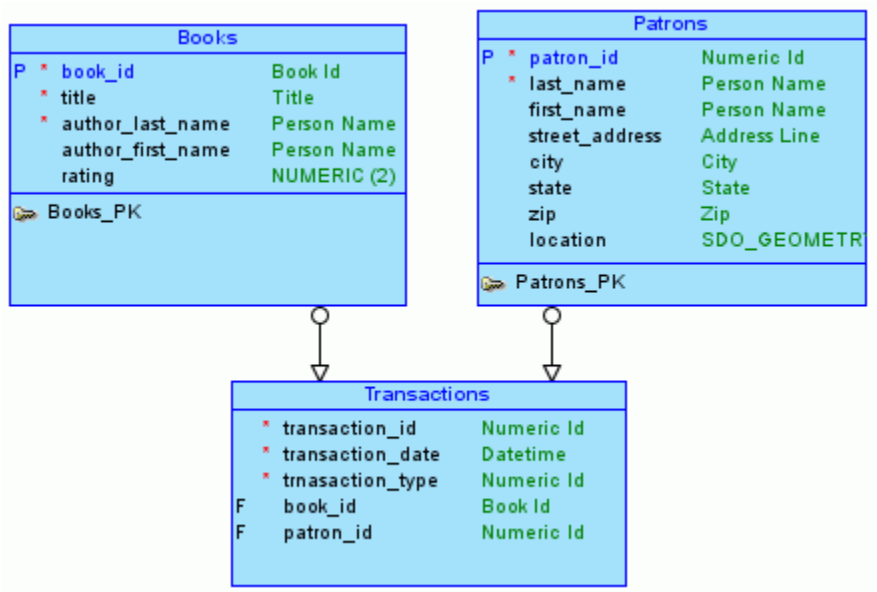
   Note that you do not explicitly define the patron_id and book_id attributes, because these will be automatically added to the Transactions entity after you create relations between the entities (see Creating Relations Between Entities (page 2-5)); they will be added as foreign keys when you generate the relational model (see Develop the Relational Model (page 2-6)).
6. Click **OK** to finish creating the Transactions entity.
7. Go to Creating Relations Between Entities (page 2-5).

## 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. Note that for this figure, Bachman notation is used (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 (page 3-69) information.
7. Go to Develop the Relational Model (page 2-6).

**Related Topics**

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)

Data Modeler User Interface (page 1-2)

# 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 the Engineer to Relational Model icon, or right-click the logical model in the navigator, then select **Engineer to Relational Model**. The Engineering (page 3-30) 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. Expand the Relational Models node in the object browser on the left side of the window, and expand Relational_1 and optionally 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) (page 3-62) dialog box specify **Name** as `Library (relational)`, and click **OK**.
5. Go to Generate DDL (page 2-6).

**Related Topics**

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)

Data Modeler User Interface (page 1-2)

# 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 11*g*) that you specify.

Develop the physical model as follows:

1. Optionally, view the physical model before you generate DDL statements:
   a. With the relational model selected and expanded, right-click the Physical Models node and select **New**. 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 11*g*), and click **OK**. A physical model reflecting the type of database is created under the Physical Models node.
   c. Expand the Physical Models node under the Library relational model, and expand the newly created physical model and the Tables node under it, to view the table objects that were 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 (page 3-21) 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 Save the Design (page 2-7).

**Related Topics**

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)

Data Modeler User Interface (page 1-2)

## 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 Database Design (page 1-11).

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 Data Modeler by clicking **File**, then **Exit**.

You can later open the saved design and continue working on it, as explained in Saving, Opening, Exporting, and Importing Designs (page 1-45).

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

Data Modeler Tutorial: Modeling for a Small Database (page 2-1)

Data Modeler User Interface (page 1-2)

# 3
# Data Modeler Dialog Boxes

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.

**Related Topics**

Data Modeler Concepts and Usage (page 1-1)

## 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 associate 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 Add/Remove Objects

This dialog box is displayed when you right-click in a subview display and select Add/Remove Elements. It lets you control which tables and views are included in the subview. There are separate tabs for Entities and Views for logical model subviews and for Tables and Views for relational model subviews.

(See also Logical Diagram and Subviews (page 1-17) and Relational Diagram and Subviews (page 1-20).)

**Select All** icon: Selects all displayed objects, to add them to the subview after you click OK.

**Deselect All** icon: Deselects all displayed objects, to remove them from the subview after you click OK

**Filter**: You can type a string to limit the display to objects whose names contain that string.

**Name**: Displays names of objects of the type appropriate for the tab.

**Include**: Lets you individually select objects to be added to the subview and deselect objects to be removed from the subview. (The additions and removals occur in the subview after you click OK.)

# 3.3 Advanced Properties (Connections)

This dialog box is displayed if you click Advanced in the dialog box for creating or editing a database connection. It has the following tabs:

- Proxy tab (page 3-2)
- SSH tab (page 3-2)

### Proxy tab

This information applies to proxy connections.

**Proxy Type**: *User Name* for authentication by proxy user name and password, or *Distinguished Name* for authentication by proxy user name and distinguished name.

**Proxy User**: Name of the user to be used for authentication for this connection.

**Proxy Password** (if Proxy Type is User Name): Password for the specified proxy user.

**Distinguished Name** (if Proxy Type is Distinguished Name): Distinguished name for the specified proxy user.

### SSH tab

This information applies to SSH (Secure Shell) connections.

**Use SSH**: Determines whether the SSH Tunnel will be used. If this option is not enabled, opening the connection will attempt to connect to the database directly.

**Host**: SSH server. SQL Developer will create an SSH session to this host, using the specified details.

**Port**: SSH port. The default port is 22.

**Username**: User name that will be used to authorize the SSH session.

**Use Key File**: Specifies that a key file should be used to provide authentication. The key file contains a private key that should correspond to a public key registered with the server. The server verifies that SQL Developer has access to the proper private key and thus the user is who he or she claims to be.

**Key File**: Path to the key file.

# 3.4 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 Arcs (page 1-19).

### 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 relationship or foreign key, double-click its item.

#### Comments, Notes, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.5 Attribute Properties

This dialog box displays the properties of an attribute, which is a component of an entity in the Logical Model (page 1-16).

#### 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 (page 3-30) 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 null value is allowed; if this option is disabled, 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 (page 3-29) 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, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Summary

See Common Information in Dialog Boxes (page 3-9).

#### Comments in RDBMS

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

## 3.6 Change Subview Object Names Prefix

This dialog box is displayed if you specify the Change Subview Object Names Prefix command on a relational model or subview. 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).

## 3.7 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**: Reason for the change request.

**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.

**Related Topics**:

Common Information in Dialog Boxes (page 3-9)

Change Requests Administration (page 3-5)

# 3.8 Change Requests Administration

Change Requests Administration enables you to manage change requests. This dialog box is displayed when you select **Change Requests Administration** from the Tools menu. You can associate a change request to one or more objects from one or more models. The object types depend on the selected model.

**Change Requests**: Lists the existing change requests. To associate a change request to a model object, select an existing change request from the left pane.

**Model**: Enables you to choose the model to display the available objects.

**Objects**: Enables you to select the object to associate with the change request. Click the blue downward arrow icon to move the object into Referenced Objects.

**Referenced Objects**: Displays the list of objects that are associated with the change request. To remove an object from this list, click the blue upward arrow icon.

**Related Topics**:

Common Information in Dialog Boxes (page 3-9) for information on creating change requests

Change Request Properties (page 3-4)

# 3.9 Check for Updates

When you click **Help** and then **Check for Updates**, you can check for and download available Data Modeler updates. The following pages may be displayed. (If you have

enabled the Data Modeler preference to check for updates automatically at startup, and if you click to see available updates at startup, the **Updates** page is displayed.)

If you are unable to check for updates because your system is behind a firewall, you may need to set the user preferences for Web Browser and Proxy (page 1-44).

1. **Source**: Select the source or sources to be checked for available updates: any or all of some specified online update centers, or a local ZIP file containing an update bundle. You can also click **Add** to add a user-defined update center.
2. **Updates**: If any updates are available from the selected source or sources, select those that you want to download.The available updates include certain third-party JDBC drivers, which require that you agree to the terms of their licenses.

   The **Show Upgrades Only** option restricts the display to upgrades of currently installed Data Modeler components. To enable the display of all new and updated components, whether currently installed or not, uncheck this option.

   After you click Next, you may be prompted to enter your Oracle Web Account user name and password. If you do not have an account, you can click the Sign Up link.
3. **License Agreements** (displayed only if you selected any updates that require a license agreement): For each update that requires you to agree to the terms of a license, review the license text and click **I Agree**. You must do this for each applicable license.
4. **Download**: If you selected any updates to download, this page displays the progress of the download operation.
5. **Summary**: Displays information about the updates that were downloaded. After you click Finish, you will be asked if you want to install the updates now and restart Data Modeler.

# 3.10 Choose Directory

This is a standard box for choosing a directory in which to place files: use **Location** to navigate to (double-clicking) the folder in which to save the files, or enter a directory name. If the directory does not already exist, it is created.

# 3.11 Collection Type Properties

This dialog box displays the properties of a collection type, which is part of the Data Types Model (page 1-11).

**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 Common Information in Dialog Boxes (page 3-9).

# 3.12 Column Properties

This dialog box displays the properties of a table column, which is part of the Relational Models (page 1-19).

### 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 null value is allowed; if this option is disabled, 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.

**Computed**: Indicates whether the column is a virtual column. A virtual column is not stored on disk. Rather, the database derives the values in a virtual column on demand by computing a set of expressions or functions.

**Column Expression**: For a derived or aggregate type, the expression for computing the value in the column.

**Auto Increment**: Controls whether a sequence is created for automatically incrementing values in this column when rows are inserted into the table. The properties of this sequence are determined by the Auto Increment (page 3-8) specifications.

**Identity Column**: Specifies whether the autoincrement column is an identity column. An identity column is an autoincrement column that can be used to identify a table row. Only one identity column can be specified for a table.

**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 (page 3-29) 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.

### Auto Increment

If Auto Increment is enabled under the General (page 3-7) column properties, specifies information that will be used to create a sequence for automatically incrementing values in this column when rows are inserted into the table.

**Start with**: Starting value of the sequence.

**Increment by**: Interval between successive numbers in the sequence.

**Min value**: Lowest possible value for the sequence. The default is 1 for an ascending sequence and -(10^26) for a descending sequence.

**Max value**: Highest possible value for the sequence. The default is 10^27 for an ascending sequence and -1 for a descending sequence.

**Cycle**: Indicates whether the sequence "wraps around" to reuse numbers after reaching its maximum value (for an ascending sequence) or its minimum value (for a descending sequence). If cycling of values is not enabled, the sequence cannot generate more values after reaching its maximum or minimum value.

**Disable Cache** and **Cache**: If Disable Cache is checked, sequence values are not preallocated in cache. If Disable Cache is not checked, sequence values are preallocated in cache, which can improve application performance; and Cache size indicates the number of sequence values preallocated in cache.

**Order**: Indicates whether sequence numbers are generated in the order in which they are requested. If no ordering is specified, sequence numbers are not guaranteed to be in the order in which they were requested.

**Sequence Name**: Name of the sequence. Must be unique within the database schema. If a sequence with the same name already exists in the physical model, the existing sequence is used.

**Trigger Name**: The name for the before-insert trigger that will be automatically created if Generate Trigger is enabled. This trigger uses the sequence to generate a new value for the primary key when a row is inserted. If a trigger with the same name already exists in the physical model, the existing trigger is used

**Generate Trigger**: Controls whether or not to generate the trigger automatically.

### Security

Specifies any relevant security-related properties for the column: whether it contains personally identifiable information (PII), contains sensitive information, or should be masked when displayed.

**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 Common Information in Dialog Boxes (page 3-9).

# 3.13 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.

**Impact Analysis**

Displays a tree representation of objects that are related in some way to the currently selected object. You can expand and collapse the tree and double-click some leaf-node objects in the tree, but you cannot add or delete related objects in the tree.

**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 (page 3-59).)

**Change Requests**

Lists any relevant change requests. To add a change request, click the Add (+) icon or right-click Change Requests in the browser and select New Change Request; to remove a change request from the object definition, select it and click the Remove (X) icon. (See also Change Request Properties (page 3-4).)

**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 (page 3-71).)

**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 (page 3-28).)

**Dynamic Properties**

Dynamic properties are name/value pairs that can be created and used during scripting . You can use the following methods relating to dynamic properties during scripting:

- void setProperty(String key, String value);
- String getProperty(String key);
- boolean hasProperty(String key)
- boolean hasProperty(String key, String value)
- void removeProperty(String key);
- void clearProperties();
- Iterator getPropertyNames();

**Summary**

Displays read-only summary information.

# 3.14 Compare Mapping

This dialog box is displayed if you click View Compare Mapping in the Relational Models (page 3-70) dialog box.

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

# 3.15 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.

# 3.16 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

**Stamp New Objects**: Associates new objects in the model (that have never been imported) with the database connection used for the Synchronize operation. That is, new objects in the model are synchronized with the data dictionary.

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

#### DDL Options

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

**Include Logging**: Controls whether logging information is included in the generated DDL statements.

**Include Schema**: Controls whether object names are prefixed with the schema name (for example, `SCOTT.EMP` as opposed to just `EMP`) in the generated DDL statements.

**Include Storage**: Controls whether storage information is included in the generated DDL statements.

**Include Tablespace**: Controls whether tablespace information is included in the generated DDL statements.

**DDL type**: Type of DDL statements to be generated: **Regular DDL**, **Advanced Interactive DDL**, or **Advanced CL DDL**

Advanced Interactive DDL and Advanced CL (command-line) DDL create a script with support for interactive (SQL*Plus or SQL Developer) or command-line (SQL*Plus only) setting of the following input parameters: start step, stop step, log file, and log level (1,2, or 3). Format for SQL*Plus execution of a resulting Advanced CL DDL script:

```
SQL> sqlplus user/password@name @script_name start_step stop_step log_file log_level
```

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

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

#### Compare Options

These options let you override defaults specified for the DDL: DDL/Comparison (page 1-35) user preferences.

#### Date/Time Format

These options let you specify formats for dates, timestamps, and timestamps with time zone.

#### Tables That Will Be Recreated

For each table that need to be re-created, lists the table name and its settings for backup strategy (**Backup** or **None**), data restoration (**Restore** or **None**), script execution, and the unload directory.

For **Script Execution**, the possible values for all DDL types include **Continue**; and for Advanced Interactive DDL and Advanced CL DDL, the values also include **Stop Before Backup**, **Stop After Backup**, and **Stop After Recreate**.

#### Data Type Conversion

For each column with different source and target data types, lists the column name, current data type, new data type, and whether to re-create the table.

#### Oracle Errors to Mask

For Advanced DDL and Advanced CL 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

## 3.17 Color Palette and Custom Colors

You can use the color palette editor to select a color from the supplied Available Colors or and saved Custom Colors.

You can also create by using a gradient box or by speechifying the RGB (Red, Green, Blue) values for the color.

## 3.18 Connection Information

To connect, you must select a database connection, and then specify the user name and password for the selected connection.

If the specified user name does not exist in the database associated with the connection, or if the specified password is not the correct one for that user, the connection is refused.

## 3.19 Contact Properties

This dialog box displays the properties of a contact object, which is a type of Business Information (page 1-25) 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 (page 3-30).)

### Locations

Lists any location objects. To add a location object, click the Add (+) icon; to remove a location object from the contact definition, select it and click the Remove (X) icon. (See also Location Properties (page 3-55).)

### 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 (page 3-98).)

### 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 (page 3-71).)

### Responsible Parties, Comments, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.20 Create Database Connection

Use this dialog box to create a database connection.

**Connection Name**: A descriptive name or alias for the connection. Example: HR_Local

**Connection type**: Type of database to which to connect: Oracle (JDBC) or JDBC ODBC Bridge.

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

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

**Role**: Database role (if any) associated with the user.

**Save Password**: Specify whether to save the password with the connection information. If the password is saved, users of the connection will not be promoted to enter the password.

### Oracle (JDBC) Settings

**Enter Custom JDBC URL**: If you select this option, enter the 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 Data Modeler is running must have an Oracle Client installation that contains the JDBC and orai18n libraries, is present on the path, and is version 10.2 or later.

**Driver**: `thin` (JDBC driver) or `oci8` (thick) (if available)

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

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

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

**Service Name**: Network service name of the database (for a remote database connection over a secure connection).

**JDBC-ODBC Bridge Settings**

**Datasource Name**: Name of an existing ODBC data source.

**Extra Parameters**: Additional parameters for the connection.

**Test Connection**: Performs a test of the connection, and indicates success or a specific error.

# 3.21 Create Discovered Foreign Keys

This dialog box is displayed if you right-click a relational model name in the browser and select Create Discovered Foreign Keys. It can help you to discover hidden foreign key relationships in the model. It is based on name and data type matching between columns in primary key columns, unique constraints, and other columns in tables.

You specify one or two foreign key column name policies to be used in the discovery process, and you can scan repeatedly with different policies and other options to see the keys that have been discovered.

If the relational model already contains foreign keys, then creating "discovered" foreign keys may create some foreign keys that seem "duplicates" of existing foreign keys (same basic information, but different foreign key names). All discovered foreign keys have the Name `createdByFKDiscoverer` and the Value `true` in the Dynamic Properties (page 3-44) pane of the Foreign Key Properties (page 3-43) dialog box.

**Column to Filter** and **Filter**: Lets you restrict the display based on a string in the name of the table, column, referred table, or referred key.

**Referred Table**: The table containing the key referred to by the specified column or columns in Table.

**Referred Key**: The key in Referred Table.

**Table**: The table in which the specified column or columns refer to the Referred Key in Referred Table.

**Columns**: The column or combination of columns in Table.

**Single Use of FK Column**: If this option is enabled, a foreign key column can be bound only to one foreign key.

**FK Column Name Policy**: The policy or policies to be applied when you click Scan Again:

- **Referred Column**: The foreign key column has the same name as the referred column.
- **PK Column Template**: The name of the foreign key column is equal to the name generated using the Foreign Key template defined under Naming Standard: Templates (page 3-24) in the Design Properties (page 3-22) dialog box.
- **Referred and Template**: First applies the Referred Column policy, then the PK Column Template policy.
- **Template and Referred**: First applies the PK Column Template policy, then the Referred Column policy.

**Scan Again**: Searches for foreign key relationships using the specified policies and options, and refreshes the list above in the dialog box.

**OK**: Creates foreign keys based on the results of the most recent scan, and closes the dialog box. (If you want to remove the discovered foreign keys, click **Edit**, then select **Remove Discovered Foreign Keys**.)

**Cancel**: Does not create foreign keys, and closes the dialog box.

# 3.22 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 (page 3-53).)

### 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 (page 3-26).)

**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 (page 3-77).)

#### 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 (page 3-78).)

#### Description

Description of the cube.

#### Partitioning Description

Description of the partitioning for the cube.

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

See Common Information in Dialog Boxes (page 3-9).

## 3.23 Custom Reports Template

This dialog box is displayed if you click **Manage** under Templates (Custom tab) when generating a report (described in Generating Reports as RTF, HTML, or PDF Files (page 1-48)). Use this dialog box to create, edit, delete, and save custom report templates, which provide you with substantial flexibility in specifying the report format. (Contrast this with the less flexible approach of creating a modified version of a standard report format, as explained in Report Templates Management (page 3-70).)

**Templates**: Lists custom report templates that have been created.

To create a new template click the **Add** icon. To edit a listed template, select it. To deleted a listed template, select it and click the **Remove** icon. To save a template after making any desired edits, click the **Save** icon.

**Template Name**: Name for the template. Suggestion: Choose a meaningful name, such as `Columns_and_Comments` or `Foreign_Keys_All` (if you select Foreign Keys - Referred From and Foreign Keys - Referring To).

**Template Description**: Optional descriptive text about the template.

**Available Collections**: Lists types of relevant information that can be added to the report layout. (Use the left and right arrow icons to move selected collections between Available Collections and Report Layout.)

**Report Layout**: Lists relevant types of information to be included in reports that use this template. (Use the up and down arrow icons to move selected items up and down in the list.)

**Available Properties**: Lists available properties for the selected Report Layout item. (Use the left and right arrow icons to move selected collections between Available Properties and Report Columns.)

**Report Columns**: Lists relevant properties to be included as columns in reports that use this template. For **Column Width**, accept the default (0.0) to have Data Modeler determine the best width, or specify a percentage value. For **Data Sort Order**, you can override the default column sort order (from the list) to specify sort order values for individual columns.

# 3.24 Data Dictionary Connections

This dialog box is displayed when you click Synchronize Model with Data Dictionary or Synchronize Data Dictionary with Model (represented by the blue arrow icons under the menu) for relational models.

**Select Connections for Synchronize**: Includes the default database connection.

**Redirect Connection**: Enables you to choose another database to synchronize.

**Database Synchronization: Use Source Connection as Filter**: If this option is enabled, then if the source and destination connections have different names, the source connection by default is used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Use Source Schema**: If this option is enabled, then if the source and destination schemas have different names, the source schema by default is used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Use Source Object**: If this option is enabled, then if the source and destination objects have different names, the source object by default is used as a filter for which objects to include in the synchronization. (You can specify differently for specific synchronization operations.)

**Database Synchronization: Synchronize the Whole Schema**: If this option is enabled, then objects that exist in the database but not in the model will appear as new objects when the model is synchronized with the database, and will appear as candidates to be deleted from the database when the database is synchronized with the model. If this option is not enabled, only objects that exist in the model are synchronized with the database.

**Database Synchronization: Exclude Remote Objects when synchronize Database with Model**: If this option is enabled, then remote objects (objects in another model) in the model are excluded when the database is synchronized with the model.

**Preferences**: Preferences dialog for synchronization options. This dialog contains options that apply to synchronizing a physical model of an Oracle database with its associated relational model. For each specified type of object, you can specify whether to synchronize it (that is, whether to have changes in the relational model to objects of the specified types be applied automatically in the associated physical models).

**Related Topics**:

Model (page 1-36)

# 3.25 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 (page 3-20) dialog box, or click **Import** to display the Import Database Connections (page 3-49) 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.)

**Select All** and **Deselect All** (this page and next page): Enable you to conveniently select or deselect all displayed items, and then deselect or select individual items.

**All Selected**: Controls whether all items are initially selected or deselected when you move to the next page (Select Objects to Import (page 3-19)).

**Options**: Displays a dialog box that lets you include or exclude specific things from the selected schemas during the import and processing:

- **Partitions**
- **Triggers**
- **Structured types** that are used
- **Secondary tables**
- **Oracle Spatial and Graph** properties

### 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 3-19) page and select at least one object.)

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

### Related Topics

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.26 Database Connection Editor

This dialog box is displayed if you add or edit a database connection while using the Data Dictionary Import (Metadata Extraction) (page 3-19) 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 (page 1-40) under General Options (see Data Modeler (page 1-32)).

**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.

# 3.27 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; if you click File, then Export, then DDL File; or if you click the Generate DDL toolbar icon.

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

**Generate**: Displays the DDL Generation Options (page 3-21) 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.

> **Note:**
>
> You should review, test, and (if necessary) edit any DDL that is generated before it is run against any database.
>
> This is especially important if you import from the data dictionary of a third-party database and then export or generate DDL for an Oracle Database physical model.

# 3.28 DDL Generation Options

This dialog box is displayed when you click Generate in the DDL File Editor (page 3-20) dialog box. These options control the content to be included in the generated script.

Options at the bottom:

- **Design Rules** displays the Design Rules (page 3-24) 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 (page 3-22) pane.
- **Advanced Interactive DDL** and **Advanced CL DDL** (command-line DDL) create a script with support for interactive (SQL*Plus or SQL Developer) or command-line (SQL*Plus only) setting of the following input parameters: start step, stop step, log file, and log level (1,2, or 3). Format for SQL*Plus execution of a resulting Advanced CL DDL script:

  ```
  SQL> sqlplus user/password@name @script_name start_step stop_step log_file
  log_level
  ```

- **Generate DDL in Separate Files**: Generates a separate file for each object to be created (as opposed to having one file with all the DDL statements). If option is enabled, a directory reflecting the model name is created under the output directory that you specify, and a directory hierarchy is created under that; the files are generated in appropriately named leaf directories.

### '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 and Advanced CL 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.

## 3.29 Design Properties

This dialog box displays the properties of the Database Design (page 1-11).

### General

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

### Settings

Contains panes with settings that affect the appearance and behavior of the design.

**Use Global Design Level Settings**: If selected, causes the current design properties to be modified to use any corresponding values in the global designs file (`datamodeler\datamodeler\types\dl_settings.xml` by default), which includes classification types, default fonts and colors, default line widths and colors, naming standard rules, and compare mappings. Also, for classification types that are used in logical or relational models in the current design and that are not currently included in the global designs file, those classification types are added to the global designs file.

**Import**: Lets you import a previously exported XML file with settings to be used for the design.

**Export**: Exports the current design's settings to an XML file.

**Save**: Saves the current settings.

### Compare Mappings

Contains options for comparing source and target objects. This information applies in special cases, such as if you have changed column properties (such as name, data type, or position in the table) and compare a model with a previous version of the same model; the mappings make it possible to know that it is the same column that is changed.

### Diagram: Classification Types

Specifies colors and optionally prefixes for the display of different classification types in a multidimensional model. You can also add (+ or Add icon) and delete (X or Remove icon) user-defined classification types.

### Diagram: Format

Specifies default object fonts and colors, and line widths and colors, for the display of different types of design objects.

### Diagram: Logical Model

**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.

### DDL

**Automatic Index Generation: Primary Key Constraint**: Controls whether an index is automatically generated for each primary key constraint.

**Automatic Index Generation: Unique Key Constraint**: Controls whether an index is automatically generated for each unique key constraint.

**Automatic Index Generation: Foreign Key Constraint**: Controls whether an index is automatically generated for each foreign key constraint.

**Preserve DDL Generation Options**: Controls whether to restore the original DDL generation options after a current DDL generation operation if you made any changes to the options for the current DDL generation operation.

### 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 specified for the old string.

### Naming Standard

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 (page 3-24), and any violations of the standards will be reported as errors or warnings. You can also apply the naming standards to primary and foreign keys, constraints, and indexes in a relational model by right-clicking the model name in the object browser and selecting **Apply Naming Standards to Keys and Constraints**.

Do not confuse naming standardization with using the Name Abbreviations (page 3-63) dialog box, which makes immediate name changes to enforce consistency in spellings and abbreviations, and which is limited to relational model name strings.

For an excellent discussion of naming standards, see the *United States Coast Guard Data Element Naming Standards Guidebook*.

**Logical Model: Separator**: Space, Title Case, or a specified Character; controls how "words" in names are separated. Title case refers to capitalizing each "word" and not

including spaces: for example, GovernmentAccounts. (Title case is sometimes called CamelCase.)

**Relational Model** and **Domain**: **Separator**: Character to be used to separate "words" in names.

**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).

**Glossary**: You can add one or more glossary files to be used in naming standardization. (For more information about glossaries, see Glossary Editor (page 3-44).)

**Naming Standard: Attribute, Column, Domain, Entity, Table**

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 for Glossary.

**Naming Standard: Templates**

For various kinds of constraints for tables and entities, 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).

**Comments, Notes, Summary**

See Common Information in Dialog Boxes (page 3-9).

# 3.30 Design Rules

Design rules are rules that can be applied to check and enforce the integrity and consistency of designs. Data Modeler comes with many predefined design rules, and you can add your own custom design rules. You can also define functions or methods to create custom transformations, and can group these transformations into libraries.

If you click **Tools** and then **Design Rules**, you can select one of the following:

- **Design Rules (page 3-24)**
- **Custom Rules (page 3-25)**
- **Libraries (page 3-26)**
- **Transformations (page 3-26)**

## 3.30.1 Design Rules

This dialog box enables you to apply Design Rules (page 3-24) and to group design rules into Rule Sets (page 3-25).

**Design Rules**

This tab 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.

#### Rule Sets

This tab enables you to create and edit named rule sets. A rule set is a collection of rules that you want to be able to check as a set.

To create a rule set, click the Add Rule Set (plus sign) icon, specify a name for the rule set, double-click the rule set number (or click the Rule Set Properties pencil icon), and use the Rule Set Properties (page 3-73) dialog box to move desired rules from the All Rules column into the Selected Rules column.

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

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

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

## 3.30.2 Custom Rules

The Custom Rules dialog box enables you to create custom design rules using a language supported by the rules execution engine, such as the Mozilla Rhino JavaScript implementation.

**Name**: Name of the custom rule.

**Object**: Type of the objects to which to apply this rule. Examples include `Table`, `Column`, `Entity`, and `Attribute`.

**Engine**: Implementation used for interpreting and executing the rule.

**Type**: Severity if a violation of the rule is detected (`Warning` or `Error`).

**Variable**: Name of the variable associated with the rule.

**Rule Script Library**: Library containing the method associated with the rule.

**Rule Script Method**: Method associated with the rule. (To modify a method, use the Custom Libraries (page 3-26) dialog box.)

You can save custom rules that you create, export custom rules that you have saved, and import design rules from an XML file.

### 3.30.3 Libraries

The Custom Libraries dialog box enables you to view, create, and delete libraries for custom design rules, and within a selected library to add, delete, and edit the methods associated with custom design rules.

If the JRuby Engine is missing, do the following:

1. If you have not already installed JRuby, download the appropriate kit from `http://jruby.org` and install it.
2. Under the location where you installed JRuby, find `jruby.jar` (in the `lib` directory).
3. Copy `jruby.jar` to the `ext` (extensions) directory under the JDK you are using with Data Modeler. For example, if you use the Windows kit that includes a JDK and if you unzipped the Data Modeler kit into `C:`, that location is:

   ```
   C:\datamodeler\jdk\jre\lib\ext
   ```

For information about design rules and custom design rules, see Design Rules (page 3-24).

### 3.30.4 Transformations

The Custom Transformation Scripts dialog box enables you to view, create, delete, and edit custom transformation scripts for implementing custom design rules.

**Name**: Name of the custom transformation.

**Object**: Type of models (logical or relational) to which to apply this transformation.

**Engine**: Implementation used for interpreting and executing the transformation.

**Variable**: Name of the variable associated with the transformation.

**Script Library**: Library containing the method associated with the transformation.

**Script Method**: Method associated with the transformation.

You can save custom transformations that you create, export custom transformations to an XML file, and import transformations from an XML file.

## 3.31 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 (page 3-46).)

### Levels

Lists levels associated with the dimension. To view or edit a level definition, double-click its item. (See also Level Properties (page 3-54).)

### 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 (page 3-3).)

### 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 member, click the Add (+) icon; to remove a calculated member 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 Common Information in Dialog Boxes (page 3-9).

# 3.32 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 Common Information in Dialog Boxes (page 3-9).

# 3.33 Distinct Type Properties

This dialog box displays the properties of a distinct type, which is part of the Data Types Model (page 1-11).

**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 Common Information in Dialog Boxes (page 3-9).

# 3.34 Document Properties

This dialog box lets you view and modify information for a document object, which is a type of Business Information (page 1-25) 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 Common Information in Dialog Boxes (page 3-9).

# 3.35 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 (page 3-29) 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 Common Information in Dialog Boxes (page 3-9).

# 3.36 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. (Note: To import domains for use, click **File**, then **Import**, then **Domains**.)

**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.

# 3.37 Email Properties

This dialog box displays the properties of an email object, which is a type of Business Information (page 1-25) 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 Common Information in Dialog Boxes (page 3-9).

# 3.38 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 unchecked objects are included or excluded in the tree display.

**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 if 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.

# 3.39 Entity Properties

This dialog box displays the properties of an entity, which is part of the Logical Model (page 1-16).

### General

**Name**: Name of the entity.

**Short Name**: Short form of the name, if any.

**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 (page 3-30) dialog box.

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

**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

**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.

**Create Surrogate Key**: Will create a unique identifier that is not the primary key.

### 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.

### Relationships

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

### Subtypes

Specifies options that are important when you are working with supertype and subtype entities.

**Subtree Generation**: Defines the strategy of engineering entities to relational model tables:

- Do not preset: Does not change the engineer-to properties.
- 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.

**Apply to Model**: Lets you apply the engineer-to properties to all models or a specified model.

**References**: When supertypes and subtypes are implemented, defines how relationships between tables are implemented in the relational model: **None**, **Identifying** (identifying foreign keys are created from subtype tables to the supertype table), or **Reverse arc** (only one child table record can exist for each supertype table record; required is the subtypes hierarchy is marked as complete [see the Complete Subtypes option]).

**Attributes Inheritance**: Defines how attributes of a supertype are inherited in both the supertype and subtype are implemented as tables (Primary Attributes Only or All Attributes).

**Generate Discriminator**: If this option is enabled, a discriminator column is generated.

**Use Attribute**: Allows an existing attribute to be set for use as the discriminator column.

**Column Name**: Name of the generated discriminator column.

**Discriminator Value**: The value related to the current entity; can exist in the discriminator column. If this field is not specified, the entity short name or name is used.

**Complete Subtypes**: If this option is enabled, the list of subtypes is marked as complete, which affects generated arcs and the list of possible values for the discriminator column. If this option is not enabled, then optional arcs are generated and the value for the current entity is included in the list of possible values for the discriminator column.

### 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.

#### Classification Types

Classification type for the entity: Fact, External, Dimension, Logging, Summary, and Temporary.

You can specify colors to be used in diagrams for each classification type in the Diagram: Classification Types (page 3-22) design properties.

#### Comments, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.40 Event Properties

This dialog box displays the properties of an event, which is part of the Process Model (page 1-15).

#### 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 (page 3-42), 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 (page 3-66), 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 Common Information in Dialog Boxes (page 3-9).

# 3.41 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.

# 3.42 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.

# 3.43 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.

## 3.44 Export to Reporting Schema

Exports the design to the reporting repository in the reporting schema (explained in Data Modeler Reports (page 1-48)). 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 tab

**Export Diagrams as PDF**: If this option is enabled, diagrams in the design are stored in the reporting repository in PDF format.

#### Maintenance tab

**Drop Repository**: Enables you to drop (delete) the reporting repository in the schema of the selected database connection. After the existing repository is dropped, a new empty repository is automatically created in this schema.

**Delete Designs**: Enables you to select a repository connection and then one or more designs within that repository, to have the information about those designs deleted from the repository.

### Glossary tab

**Export Glossary**: Enables you to specify a Data Modeler glossary file, to have its information exported to the reporting repository.

**Delete Glossary**: Enables you to select a glossary in the reporting repository, to have its information deleted from the repository.

## 3.45 Export Wizard

This wizard exports database objects and optionally data. For a selected database connection, you can export some or all objects of one or more types of database objects to output files, a SQL Worksheet, or the clipboard. The output may contain SQL data definition language (DDL) statements to create these objects. It may also contain SQL statements to insert data into these objects and other formatted files.

The number of panes and the options available depend on the potential scope of the export operation.

### Source/Destination page

Contains up to four main areas for specifying the database connection and the DDL, data, and target options.

**Connection**: Select the database connection with the objects to be exported.

**Export DDL**: If this option is enabled, DDL statements are included in the export operation. Select the options to apply to the DDL that is generated.

**Export Data**: If this option is enabled, data is included in the operation. Select the options to apply to the data that is unloaded.

> **Note:**
>
> An **Export Format Error** is displayed if you attempt to export both DDL and Data when Format is `loader`.

**Format**: Select the desired output format for the data to be unloaded. Depending on the selected format, other options may appear. For example, for **xls** (Microsoft Excel file), you can specify worksheet names for the data and the SELECT statement.

For CLOB data, exporting is supported only if the format is `loader` (SQL*Loader) or `pdf` (PDF). Some export types export only a subset of the string followed by an ellipsis (...).

To ***paste unloaded data into a Microsoft Excel*** file, specify Export Data but not Export DDL, select **text** for Format, and select **Clipboard** for Save As; and after completing the unload, paste from the clipboard into Excel.

**Save As**: Specifies how or where target statements and data are to be saved:

- **Single File**: A single file contains both DDL and data. When you are unloading DDL, only Insert format can be specified for data.
- **Separate Files**: Each object is saved to a separate file in the specified directory.

- **Type Files**: Objects of the same type are saved to a single file in the specified directory.
- **Separate Directories**: A directory for each object type being exported is created. Files are created in the appropriate directory.
- **Worksheet**: Statements are sent to a SQL Worksheet window.
- **Clipboard**: Statements are copied to the clipboard.

**Encoding**: Character set to be used for encoding of the output.

#### Export Summary page

Contains a summary of the actions that will be taken. If you want to change anything, go back and make the changes, then finish the wizard steps.

## 3.46 Export/Import Connections

The Export Connections wizard exports information about one or more database connections to an XML file. The Import Connections wizard imports connections that have been exported. Database connections that you import are added to any connections that already exist.

- Export Connections (page 3-38)
- Import Connections (page 3-39)

### 3.46.1 Export Connections

#### Select Connections

You can select and deselect all connections or specific connections for the operation.

#### Destination File

**File Name**: Name of the XML file to contain definitions of the connections to be exported. Use the Browse button to specify the location.

#### Password Handling

Specify how passwords should be handled in the exported file:

- **Encrypt all passwords with a key**: Specify an Encryption Key value and verify that value. (Anyone who attempts to import connections from the exported file will need to know the encryption key in order to be able to use connections with saved passwords without being prompted for the password.)
- **Remove all passwords from the exported connections**: Removes any saved passwords from the exported connections. (If the connections are later imported, users will need to know the passwords for connections that they plan to use.)

#### Summary

Displays a summary of the options you specified. To make any changes, press **Back** as needed and change the information. To start the operation, click **Finish**.

## 3.46.2 Import Connections

**Source File**

**File Name**: Name of the XML file that contains definitions of the connections to be imported. Use the Browse button to specify the location.

**Password Handling**

Either specify the key that was used to encrypt the passwords when the connections were exported, or remove all passwords for the imported connections:

- **Use a key to decrypt all passwords**: Specify the **Encryption Key** value that was used to encrypt the passwords for the export operation. (If an encryption key was used and if you do not know it, use the option to remove all passwords from the exported connections.)
- **Remove all passwords from the exported connections**: During the import operation, removes any passwords that are saved in exported connections. (Users of the imported connections will need to know the passwords for connections that they plan to use.)

**Select Connections**

You can select and deselect all connections or specific connections for the operation.

**Duplicate Connections**: Determines what happens each existing connection that has the same name as a connection in the source file used for the import operation.

- **Rename**: Gives each duplicate connection a new name similar to the name of the existing connection.
- **Replace**: Replaces the information for the existing connection with the information for that imported connection.

**Summary**

Displays a summary of the options you specified. To make any changes, press **Back** as needed and change the information. To start the operation, click **Finish**.

# 3.47 External Agent Properties

This dialog box displays the properties of an external agent object, which is part of the Process Model (page 1-15).

**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 (page 3-42), 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 (page 3-42), 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, 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 Common Information in Dialog Boxes (page 3-9).

# 3.48 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 (page 1-15).

**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.

# 3.49 File Processing

This dialog box is displayed if you right-click a directory or file in the Files navigator and select Tag SQL. In the selected file or in files under the selected directory, all SQL statements that match a specified pattern are rewritten so that a comment is inserted after the SELECT keyword in the main query block that uniquely tags the statement.

For example, a PL/SQL source containing the statement "SELECT 1 FROM dual" would be rewritten into something like "SELECT /* PREFIX 00f7d2 */ FROM dual". This allows easy identification of problematic statements that appear in performance views such as V$SQL.

**Tag Prefix**: Text to appear after "PREFIX" in inserted comments.

**Selected Extensions**: Extensions of files on which to perform the SQL statement rewrite operations.

# 3.50 Find Object (Search)

This window is displayed if you:

- Click the **Find** (*binoculars* toolbar icon) when a diagram is selected, to search that diagram.
- Select **Find** from a context menu, to search the associated object.

You can search for strings or you can use regular expressions, specify simple or advanced mode, filter results by object type and by property values, and generate reports from the search results.

**Pattern**: Enter the pattern string for the search. The display is dynamically updated either as you type or when you press Enter, depending on the option specified in the Search (page 1-40) user preferences.

**Case Sensitive**: Controls whether the search is case sensitive or case insensitive with respect to alphabetic characters.

**Filter**: Select ALL to search all object types relevant to the Pattern; or select an object type to limit the search to objects of that type.

**Saved Searches**: Displays any saved searches, so that you can use a saved search now. To save a search, enter the specifics for the search, then enter a name for the saved search, click **More**, and select **Save**.

**Report** (available only if Filter specifies an object type): Displays the dialog box for generating Data Modeler Reports (page 1-48), to display and save a report on the selected objects.

**Properties** (available only if Filter specifies an object type): Displays the Set Common Properties (page 3-76) dialog box, in which you can specify new values for properties that the selected objects have in common.

**Simple Mode** or **Advanced Mode**: Simple Mode includes more basic search options; Advanced Mode includes options such as searching by object type and property optionally specifying a property and case sensitivity for each), and specifying OR or AND search logic.

**Stop on First**: Displays only the first occurrence if there are any exact duplicates under Name.

**Search Profile**: Displays any available **search profiles** that you can use for the search operation. (See the information about the Search (page 1-40) user preferences and the Search Profile (page 3-73) dialog box.)

# 3.51 Flow Properties

This dialog box displays the properties of a flow (data flow) object, which is part of the Process Model (page 1-15).

### 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 (page 3-52).)

**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 Common Information in Dialog Boxes (page 3-9).

# 3.52 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 (page 1-19).

**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 (shown by a crossing line in diagrams) performs no action on these rows; CASCADE (shown by an "X") deletes these rows; SET NULL (shown by a small circle) sets null all columns in those rows that can be set to a null value; RESTRICT (also shown by one crossing line) 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 (Updatable)**: Controls whether the foreign key relationship is transferable (that is, updatable). 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.

If the foreign key relationship is non-transferable, a diamond appears on the line in the diagram.

**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 Arcs (page 1-19).

#### Associated Columns

Lists each parent column and child column pair in the foreign key definition.

#### Dynamic Properties

For foreign keys that have been discovered (see Create Discovered Foreign Keys (page 3-14)), contains the **Name** `createdByFKDiscoverer` and the **Value** `true`.

See also Dynamic Properties in Common Information in Dialog Boxes (page 3-9).

#### Comments, Notes, Impact Analysis, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.53 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 (page 3-23) in the Design Properties (page 3-22) dialog box topic.)

A **glossary** in Data Modeler is a set of accepted terms that may or must be used in the design. Glossaries are used by Design Rules (page 3-24) to ensure that the model complies with your set naming standards or when engineering between the logical and relational models. You can create a new glossary from scratch, use existing glossaries, or generate a new glossary based on an existing logical model.

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 multiword 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.

**Plural**: Optional plural form of the associated entry (for example, AMOUNTS as the plural for AMOUNT). If a Plural value is specified, the singular and plural are considered the same during design rule validation and during transformation for entities and tables.

**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.

# 3.54 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 (page 3-54).)

### 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 (page 3-73).)

**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 Common Information in Dialog Boxes (page 3-9).

# 3.55 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 types directory (folder) under the location where you installed Data Modeler. The domain definitions in the `defaultdomains.xml` file are available whenever you use 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.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.56 Import Glossary (Naming Standard Definitions)

This dialog box is displayed if you click Import in the Glossary Editor (page 3-44) 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.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.57 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 (page 3-49) 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**.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.58 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.

For **Select Objects to Import**, if you are importing from a versioned Designer repository, only objects that have been checked in appear in the list of objects that you can import.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

Importing an Oracle Designer Model (page 1-47)

# 3.59 Import 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 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 (page 3-10) dialog box.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.60 Import Database Connections

This dialog box is displayed if you click Import to import database connections in the Data Dictionary Import (Metadata Extraction) (page 3-19) wizard.

**File Name**: Click **Browse** to find the XML file containing database connection definitions that have been exported from SQL Developer or Data Modeler.

**Connections**: Displays the names of database connections in the specified file.

Click **OK** to import the database connections.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.61 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.

# 3.62 Index, Primary Key, or Unique Key Properties

This dialog box displays the properties of an index, including several types of "index" such as primary key or unique key. These are objects that can be defined on one or more table columns in Relational Models (page 1-19).

**General**

**Name**: Name of the index object.

**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 Inde**x: 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.

#### Comments, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.63 Information Store Properties

This dialog box lets you view and modify information for an information store object, which is part of the Process Model (page 1-15).

### 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, 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 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 Common Information in Dialog Boxes (page 3-9).

## 3.64 Information Structure Properties

This dialog box lets you view and modify information for an information structure object, which is part of the Process Model (page 1-15).

#### 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 Common Information in Dialog Boxes (page 3-9).

## 3.65 Inheritance Relation Properties - <hierarchy-name>

This dialog box displays the inheritance properties between a subtype entity and a supertype entity in a hierarchy (see Inheritances (page 1-18) under Logical Model (page 1-16)).

**General**

**Name**: Name of the inheritance relationship.

**Long Name**: Long name showing the relationship between the entities.

**Supertype**: Supertype entity in the relationship.

**Subtype**: Subtype entity in the relationship.

**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 Common Information in Dialog Boxes (page 3-9).

## 3.66 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 Common Information in Dialog Boxes (page 3-9).

# 3.67 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 Common Information in Dialog Boxes (page 3-9).

## 3.68 Location Properties

This dialog box displays the properties of a location object, which is a type of Business Information (page 1-25) 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 Common Information in Dialog Boxes (page 3-9).

## 3.69 Manage Features and Updates

This dialog box is displayed when you click Tools, then Features. It lets you enable or disable features in the product.

**Check for Updates**: Lets you check for any available updates now and specify whether to check for updates automatically when the product is started.

**Search icon**: Enter text to filter the Available Features field.

#### Features tab

**Clear Cache:** Removes previously loaded features from the cache.

**Available Features**: Displays the features available in the product . Uncheck a feature to disable it. Click **Expand All** to expand all nodes and **Collapse All** to collapse all nodes in the tree.

#### Installed Updates tab

Lets you see and remove installed updates. Click **Expand All** to expand all nodes and **Collapse All** to collapse all nodes in the tree.

## 3.70 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.

# 3.71 Mask Templates Administration

This dialog box is displayed if you click Tools, then Mask Templates Administration. It lets you create one or more "templates" that you can then associate with appropriate columns in tables in a relational model.

Oracle Data Redaction, a part of Oracle Advanced Security, enables you to mask (redact) data that is sensitive. Redaction policies can be applied to table columns.

**Available Mask Templates**: Lists any mask templates that are available to be assigned to a column of a suitable data type (using the Security tab under Columns (page 3-94) under Table Properties (page 3-93)).

Buttons: **Add** lets you create a new mask template; **Remove** deletes the selected mask template; **Modify** lets you edit the selected template; **Apply** applies any changes you have made to the selected template.

**Name**: Name for the mask template

**Function Type**: `PARTIAL` (Partial redaction, redact a portion of the column data) or `REGEXP` (Regular expression based redaction).

- If Function Type is `PARTIAL`: **Datatype** can be **Character**, **Numeric**, or **Date**. (Remaining fields depend on the data type.)
- If Function Type is `REGEXP`, the remaining fields are **Pattern**, **Replace String**, **Position**, **Occurrence**, and **Match Parameter**.

For more detailed information about redaction and masks, see the following:

- "Oracle Data Redaction" section in the "Topics for Database Administrators and Developers" chapter in *Oracle Database Concepts*.
- `DBMS_REDACT` package information in *Oracle Database PL/SQL Packages and Types Reference*.

# 3.72 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 measure folders, 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 (page 3-58).)

### 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 Common Information in Dialog Boxes (page 3-9).

# 3.73 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 measures, 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 in 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.

**Addivity**: 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 *Mymeasure*

**Description**

Description of the measure. Example: Unit sales measure

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

See Common Information in Dialog Boxes (page 3-9).

# 3.74 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.

# 3.75 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 Common Information in Dialog Boxes (page 3-9).

## 3.76 Model Properties - Business Information

The Business Information (page 1-25) model has the following properties.

**General**

**Name**: Name of the business information model.

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

**Comments, Notes, Summary**

See Common Information in Dialog Boxes (page 3-9).

## 3.77 Model Properties - <data-flow-diagram-name>

Data Flow Diagrams (page 1-16), which are part of the Process Model (page 1-15), have the following properties.

**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 (page 3-66), 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 Common Information in Dialog Boxes (page 3-9).

## 3.78 Model Properties - Data Types

The Data Types Model (page 1-11) has the following properties.

**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 Common Information in Dialog Boxes (page 3-9).

## 3.79 Model Properties - Logical

The Logical Model (page 1-16) has the following properties.

**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 Common Information in Dialog Boxes (page 3-9).

## 3.80 Model Properties - <multidimensional-model-name>

A multidimensional model has the following properties.

**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 Common Information in Dialog Boxes (page 3-9).

# 3.81 Model Properties - Process Model

The Process Model (page 1-15) has the following properties.

**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 Common Information in Dialog Boxes (page 3-9).

# 3.82 Model Properties - <name> (Relational)

Relational Models (page 1-19) have the following properties.

**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 Physical Models (page 1-22).)

**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 Common Information in Dialog Boxes (page 3-9).

# 3.83 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 (page 3-24). Naming standardization is implemented through design properties, as explained in Naming Standard (page 3-23).

# 3.84 New/Edit SSH Connection

This dialog box enables you to create or edit an SSH (Secure Shell) connection.

**Host**: SSH server. SQL Developer will create an SSH session to this host, using the specified details.

**Port**: SSH port. The default port is 22.

**Username**: User name that will be used to authorize the SSH session.

**Use Key File**: Specifies that a key file should be used to provide authentication. The key file contains a private key that should correspond to a public key registered with the server. The server verifies that SQL Developer has access to the proper private key and thus the user is who he or she claims to be.

**Key File**: Path to the key file.

# 3.85 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 Data Modeler Reports (page 1-48)).

When you have finished entering the connection information, test the connection, as explained in After Specifying the Connection Information (page 3-65).

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 (page 3-64) or Other databases (third-party) tabs (page 3-65).

### 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 Data Modeler is running must have an Oracle Client installation that contains the JDBC and orai18n 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**.

## 3.86 Object Names Administration

This dialog box is displayed if you click Tools, then Object Names Administration. It lets you make the names of specified objects fixed (unchangeable) or changeable in dialog boxes for the properties of the objects. For example, if the name of the Employees table is fixed, then it appears grayed out (not editable) in the Table Properties (page 3-93) dialog box for the Employees table in the relational model.

To see the objects whose names you can specify as fixed or changeable, select each tab for the available types. The page for each tab includes Select All and Deselect All

icons, a Filter box, and rows for each available object (including a Fixed box in each displayed row).

**Select All** icon: Selects all displayed objects.

**Deselect All** icon: Deselects all displayed objects.

**Filter**: Lets you enter a string to restrict the display to those objects with names containing that string.

**Name**: Name of an object of the type associated with the selected tab.

**Fixed**: If this option is selected, the object name in not changeable in the properties dialog box for the object; if this option is not selected, the object name is changeable.

**Apply**: You must click Apply to make your specifications effective.

# 3.87 Process Properties

This dialog box displays the properties of a process object, which is part of the Process Model (page 1-15).

### 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.

### Short Definition

Text for a short definition of the process.

### 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 (page 3-52), 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 (page 3-34), 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 (page 3-42), 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 (page 3-42), 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 (page 3-3), 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 (page 3-32), double-click an entity item or select the item and click the Properties icon.

### 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 (page 3-67) pane.)

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

See Common Information in Dialog Boxes (page 3-9).

# 3.88 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 Physical Models (page 1-22).)

**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**.

# 3.89 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 Common Information in Dialog Boxes (page 3-9).

# 3.90 Relation Properties

This dialog box displays the properties of a relationship, which is part of the Logical Model (page 1-16).

**General**

**Name**: Name of the relationship.

**Use Surrogate Keys**: Controls whether surrogate keys are used to ensure uniqueness by adding a level prefix to the members.

**Source Cardinality**

**Source**: Source entity for the relationship.

Source Key:

**Name on Source**: Text describing the role of the source entity in the relationship (for example, "has"). This text is displayed in the logical model diagram if you select **Show**, then **Labels** from the diagram context menu.

**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 relationship 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.

Transferable:

**Target Cardinality**

**Target**: Target entity for the relationship.

Target Key:

**Name on Target**: Text describing the role of the target entity in the relationship (for example, "any number of"). This text is displayed in the logical model diagram if you select **Show**, then **Labels** from the diagram context menu.

**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 relationship 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.

Transferable:

**Dominant Role**: Entity with the dominant role.

**Identifying**: Controls whether this is an identifying relationship. When there is an identifying relationship between a parent entity and a child entity, when the relational model is generated, the following occurs in the child table: the foreign key to the parent becomes part of the primary key of the child. (In non-identifying relationships, the

foreign key to the parent table is just another column in the child table and is not part of the primary key.)

**Delete Rule**: Action to take automatically for the child end of the relationship; used in engineering to the relational model. When a row in the child table is deleted and rows with that value exist in the parent table, 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.

#### Attributes

Lets you define any attributes of the relationship.

**Name**: Name of the attribute.

**Datatype**: Data type of the attribute.

**Mandatory**:

#### Engineer To

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

**Engineer**: Controls whether the relationship 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.

#### Comments, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.91 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 (page 3-10) dialog box.

When you click OK, the Compare Models (page 3-10) dialog box is displayed, in which you can apply a filter to restrict the types of objects and specific objects to be merged.

## 3.92 Report Templates Management

This dialog box is displayed if you click **Manage** under Templates (Standard tab) when generating a report (described in Generating Reports as RTF, HTML, or PDF Files

(page 1-48)). Use this dialog box to create, edit, delete, and save modified versions of standard report templates, which let you specify the types of objects to be included in a report. (Contrast this with the more flexible approach of creating a custom report format, as explained in Custom Reports Template (page 3-17).)

**Templates**: Lists report templates that have been created.

To create a new template click **Add**. To edit a listed template, select it and click **Edit**. To deleted a listed template, select it and click **Remove**. To save a template after making any desired edits, click **Save**.

**Template Name**: Name for the template. Suggestion: Choose a meaningful name, such as `Columns_and_Comments` or `Foreign_Keys_All` (if you select Foreign Keys - Referred From and Foreign Keys - Referring To).

**Report Sections**: Select (check) items that indicate the kinds of information to be included in this report template. For example, you might choose to select Columns, Column Comments, Constraints, Foreign Keys - Referred From, Foreign Keys - Referring To, and Indexes.

# 3.93 Resource Locator Properties

This dialog box displays the properties of a resource locator object, which is a type of Business Information (page 1-25) 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 Common Information in Dialog Boxes (page 3-9).

# 3.94 Responsible Party Properties

This dialog box displays the properties of a responsible party object, which is a type of Business Information (page 1-25) 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 Common Information in Dialog Boxes (page 3-9).

## 3.95 Revision Lister

This dialog box is displayed if you click List Revisions in the Subversion: Branch/Tag (page 3-82) dialog box. It contains a list of revisions in the repository.

Select the desired revision to use, and click **OK**.

## 3.96 Role Properties

This dialog box displays the properties of a role object, which is part of the Process Model (page 1-15).

#### 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 Common Information in Dialog Boxes (page 3-9).

# 3.97 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 the online help for Analytic Workspace Manager.

# 3.98 Rule Set Properties

This dialog box displays the properties of rule set that is selected on the Rule Sets (page 3-25) tab of the Design Rules (page 3-24) dialog box. To specify design rules to be included in this rule set, select the desired rules and move them from the All Rules column into the Selected Rules column.

# 3.99 Search Profile

This dialog box is displayed when you click the Add icon under Search Profiles or double-click an existing profile name on the Search (page 1-40) preferences pane.

**File Name**: Name of the XML file containing the search profile definition. The default file extension is `.sposdm`.

**Path**: Directory path to the search profile file.

**Description**: Optional descriptive text about the search profile, such as its purpose or goal.

**Active**: Controls whether the search profile appears as selectable when you perform a search in Data Modeler.

**Objects**: The Relational Model and Logical Model tabs identify the types of objects for which you can specify properties to be included in the search profile.

**Properties**: Lists the available properties for the selected type of object.

- **Filter**: Lets you type a string to limit the display to property names that contain that string.
- **Property Name**: Name of the property.
- **Use**: Controls whether the property is included in the search profile.

# 3.100 Schema Properties

This dialog box displays the properties of a database schema, which can be associated with tables and other objects in Relational Models (page 1-19). If you associate a relational model object with a schema, the schema name appears in diagrams and it is used during DDL generation.

**General**

**Name**: Name of the database schema.

**Tables, Views, Indexes**

For tables, views, and indexes, All (available) objects of that type and Selected objects of that type are displayed. To associate an object with the schema, select it under All and click the Add (right-arrow) icon to move it to the Selected column. To remove an association, select it under Selected and click the Remove (left-arrow) icon to move it to the All column.

**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**

See Common Information in Dialog Boxes (page 3-9).

# 3.101 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**: This option (if available) determines which definitions are merged into which definitions, and which model appears in the left and right panes of the Relational Models (page 3-70) dialog box for merging (if that box is displayed). For the ensuing import and merge operation, if this option is enabled, a script is generated in which your current relational model is merged into the specified database (DDL files); but 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 (page 3-70) dialog box is not displayed before the import operation occurs.

# 3.102 Select File

This is a standard box for selecting a file for an operation: use **Location** to navigate to the folder in which to save or open the file, then select or specify the file name (including any extension) and, if necessary, the file type.

# 3.103 Select Models/Subviews to Export

This dialog box is displayed if you click **File**, then **Export**, then **To Data Modeler Design**. Use this dialog box to create, edit, delete, and save export configurations, which let you customize the models and subviews to be exported.

**Export Configurations**: Lists any configurations that have been created.

To create a new configuration, click **Add**. To edit a listed configuration, select it and click **Edit**. To deleted a listed configuration, select it and click **Remove**. To save a configuration after making any desired edits, click **Save**.

**Export Name**: Name for the export configuration.

**Description**: Optional descriptive text.

**Models: Filter**: You can start typing to limit the list of available models to those with names starting with the typed characters. To select from the available models, click the desired name or names and click the right-arrow icon to move them to the Selected list.

**Subviews: Filter**: You can start typing to limit the list of available subviews to those with names starting with the typed characters. To select from the available subviews, click the desired name or names and click the right-arrow icon to move them to the Selected list.

**Related Topics**

Saving, Opening, Exporting, and Importing Designs (page 1-45)

# 3.104 Sensitive Type Properties

This dialog box displays the properties of a sensitive type, which can be associated with TSDP (Transparent Sensitive Data Protection) policies. You should be familiar with the concepts and techniques explained in the "Using Transparent Sensitive Data Protection" chapter in *Oracle Database Security Guide*.

### General

**Name**: Name of the sensitive type.

**Generate in DDL**: Controls whether the sensitive type creation is included when DDL statements are generated.

**Enable**: controls whether the sensitive type is selectable for association with an attribute or column.

### Description

Optional descriptive text about the sensitive type.

### Used In

Lets you specify attributes and columns in which the sensitive type is to be used.

**Design**: Design name.

**Model**: Name of a logical or relational model.

**List of entities or tables**: Click one to display its attributes or columns.

**Columns/Attributes**: List of the attributes or columns of the selected entity or table. Use the down arrow icon to move the select attributes or columns to **Selected Elements**.

### Comments, Notes, Summary

See Common Information in Dialog Boxes (page 3-9).

**Related Topics**

TSDP Policy Properties (page 3-100)

# 3.105 Set Classification Types

This dialog box is used to specify, for entities or tables in a multidimensional model, the classification type for each: Fact, External, Dimension, Logging, Summary, or Temporary.

You can specify colors to be used in diagrams for each classification type in the Diagram: Classification Types (page 3-22) design properties.

# 3.106 Set Common Properties

This dialog box is displayed if you click Properties in the Find Object (Search) (page 3-41) window.

For each property that the selected objects have in common, the list displays the **Property Name**, **Old Value** (value when the dialog box was displayed), **New Value** (if you have specified a new value), and **Type** (data type).

To specify new values for any listed properties, for **New Value** enter the desired new value if possible; otherwise, an ellipsis (**...**) button appears next to Type, and you can click that button to select the desired new value.

# 3.107 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 (page 3-56) 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.

# 3.108 Show/Hide Elements

This dialog box lets you show and hide elements in the display for one or more objects in a diagram. For example, you could specify that for the Products table in the relational model diagram, the UnitsInStock column should be hidden (not shown). You can hide or unhide all elements or specific elements.

**Apply to This View Only**: Controls whether the show/hide specifications are applied in this view or all views.

**Hide All Elements**: Hides all listed elements.

**Unhide All Elements**: Unhides all listed elements.

**Filter**: Limits the list of elements to those containing the specified substring.

**Type**: Type of element. For example, for a table the elements might include columns, indexes, and the primary key.

**Element Name**: Name of the element.

**Hidden**: Indicates whether the element is to be hidden (if checked) or not.

# 3.109 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 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 (page 3-54).)

### 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 (page 3-26).)

### 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 (page 3-78).)

**Description**

Description of the slice.

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

See Common Information in Dialog Boxes (page 3-9).

# 3.110 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 and Graph Developer's Guide*.

**Name**: Name of the spatial 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 and Graph 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).

# 3.111 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 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 that reflects the current specifications. On the SQL Statement tabs in this box, the Show Formatted Limit Map option enables dimension information to be formated (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 Common Information in Dialog Boxes (page 3-9).

## 3.112 Standard Reports Configurations

This dialog box is displayed if you click **Manage** under Designs in the Reports dialog box (see Generating Reports as RTF, HTML, or PDF Files (page 1-48)). Use this dialog box to create, edit, delete, and save report configurations, which let you customize the subviews and objects to be included in a report.

**Standard Reports Configurations**: Lists any configurations that have been created.

To create a new configuration, click **Add**. To edit a listed configuration, select it and click **Edit**. To deleted a listed configuration, select it and click **Remove**. To save a configuration after making any desired edits, click **Save**.

**Report Name**: Name for the configuration.

**Include All Objects**: Causes the report to contain information about all objects of the specified Available Reports type.

**Choose Subview(s) Objects**: Limits the report to objects of the specified Available Reports type in the specified subview or subviews.

**Description**: Optional descriptive text.

**Subviews: Filter**: You can start typing to limit the list of available subviews to those with names starting with the typed characters. To select from the available subviews, click the desired name or names and click the right-arrow icon to move them to the Selected list.

**Objects: Filter**: You can start typing to limit the Object Name list to objects with names starting with the typed characters.

**Use**: You can check or uncheck each listed Object Name to have it included or excluded, respectively, in the report.

# 3.113 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 Common Information in Dialog Boxes (page 3-9).

# 3.114 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 Common Information in Dialog Boxes (page 3-9).

# 3.115 Subversion: Add Property

Use this dialog box to add a versioning property for the currently selected file or folder.

**Related Topics**

Using Versioning (page 1-53)

# 3.116 Subversion: Add to Source Control

Use this dialog box to bring a new file under Subversion control.

**Files List**: Lists the names and physical locations of the files that will be added to Subversion.

**Related Topics**

Using Versioning (page 1-53)

# 3.117 Subversion: Apply Patch

Use this dialog box to apply a previously generated patch. A patch must be applied to the same revision/tag from which it was generated.

The name and location are displayed for the project or set of files to which the patch will be applied.

**Patch Source**: Specify where the patch will be obtained from: the system clipboard or a file.

**System Clipboard**: Obtains the patch from the system clipboard.

**File**: Obtains the patch from a file. If you select this option, a suggested location and name for the patch file is already entered in the text box. You can change this by typing a new location and name into the box, or by using the **Browse** button to find a new location

**Related Topics**

Using Versioning (page 1-53)

# 3.118 Subversion: Branch/Tag

This dialog box is displayed when you right-click a remote directory in the Subversion repository and select Branch/Tag. Create a branch by copying the current working copy or a revision from the repository to a selected location in the repository.

**From**: Location of the working copy or revision.

**Working Copy**: Causes the current working copy to be copied.

**HEAD Revision**: Causes the HEAD revision (the latest revision in the repository) to be copied.

**Use Revision**: Causes the revision specified in the text box to be copied. To see a list of revisions from which you can choose, click **List Revisions**.

**To**: Destination location.

**Comment**: Optional descriptive comment.

**Switch to new branch/tag**: If this option is checked, the existing working copy is switched to the new branch.

After you click **OK**, the SVN Console - Log pane is displayed at the bottom, with messages about commands that were executed.

**Related Topics**

Using Versioning (page 1-53)

# 3.119 Subversion: Check Out from Subversion

Use this dialog box to check out modules from a Subversion repository, to create the initial local copies of files and folders that are being stored within a Subversion repository. It is these local copies, stored outside the Subversion repository, that you work on. This location and the files within it constitute the Subversion "working copy".

**Note**: With Subversion, there is no "check in" procedure, so you do not check in files that you have updated and check them out again when you next want to work on them. Instead, when you have finished working on your local copies of files, you **commit** them to the Subversion repository to bring the files held there up to date. Also, if you want to incorporate into your local copies changes that other developers have committed to the Subversion repository, you **update** them.

**Destination**: Directory or folder into which to place the checked out files. If this destination is not empty, a warning message will be displayed asking if you are sure you want to check out into this directory. (Attempting to check out files into a non-empty destination might reflect a mistake in specifying the destination, or it might be your intention.)

**Use Revision**: If this option is checked, the revision you specify in the text box is used. To see the available revisions, click the binoculars icon.

**Depth**: The level of recursion for selecting files to be checked out, from Infinity (all children at all levels under the selected item in the Versions browser hierarchy) through Empty (only this item and no children).

**Related Topics**

Using Versioning (page 1-53)

## 3.120 Subversion: Commit Resources

Use this dialog box to commit individual files to the Subversion repository. If a file is a child of a changed parent that has not been committed, you must either first commit the parent or instead commit the working copy.

The committed files will replace those in the repository as the most recent. Other developers who subsequently check out or update from these files will see the file changed in comparison with the previous version held in the repository.

**Files List**: Lists the names and physical locations of the files that will be committed to the Subversion repository.

**Keep Locks**: Retains the locks that you previously obtained on the files that you are about to commit. This will mean that other developers will still not be able to commit changes they may have made to the files.

**Comments**: Comments to accompany the commit action. You will later be able to see these comments when viewing the list of versions of a particular file.

**Template or Previous Comments**: A template with comment text for the Comments box.

**Related Topics**

Using Versioning (page 1-53)

## 3.121 Subversion: Commit Working Copy

Use this dialog box to commit the working copy to the Subversion repository. The committed files will replace those in the repository as the most recent. Other developers who subsequently check out or update from these files will see the file changed in comparison with the previous version held in the repository.

**Files List**: Lists the names and physical locations of the working copy that will be committed to the Subversion repository.

**Keep Locks**: Retains the locks that you previously obtained on the files that you are about to commit. This will mean that other developers will still not be able to commit changes they may have made to the files.

**Comments**: Comments to accompany the commit action. You will later be able to see these comments when viewing the list of versions of a particular file.

**Related Topics**

Using Versioning (page 1-53)

## 3.122 Subversion: Confirm Checkout

This dialog box is displayed if you attempt to check out from the repository root, as opposed to from the branches, tags, or trunk of the repository. To proceed with the checkout from the root, click **Yes**; to cancel this request, click **No**.

**Skip This Message Next Time**: If you enable this option, on future requests to check out from the repository root, this dialog box will not be displayed and the operation will proceed as if you had clicked Yes.

**Related Topics**

Using Versioning (page 1-53)

# 3.123 Subversion: Create Remote Directory

Use this dialog box to create a remote directory for a connection in a Subversion repository.

**Directory Name**: Directory name to be associated with the specified URL.

**Comments**: Optional descriptive comment.

**Related Topics**

Using Versioning (page 1-53)

# 3.124 Subversion: Create Subversion Repository

This information applies to creating a Subversion repository. A connection to the repository will be created automatically. For information about Data Modeler support for versioning and Subversion, see Using Versioning (page 1-53).

**Repository Path**: Location for the new Subversion repository. You can **Browse** to select the location.

**File System Type**: Data storage system type for the repository. For information about choosing a system, see "Version Control with Subversion" at `http://svnbook.red-bean.com/`.

- **Native**: The file system type being used by the operating system.
- **Berkeley DB**: Causes a Berkeley DB database to be used as the data storage system.

**Connection Name**: Name for this connection. If you leave this box blank, the connection will be given a name based on the URL of the repository location.

**Related Topics**

Using Versioning (page 1-53)

# 3.125 Subversion: Create/Edit Subversion Connection

This information applies to creating or editing a Subversion connection. For information about Data Modeler support for versioning and Subversion, see Using Versioning (page 1-53).

**Repository URL**: Full, valid URL for the location of the Subversion repository. The following are URL schemas and the access methods they map to:

- file:/// -- Direct repository access (on local disk)
- http://-- Access via WebDAV protocol to Subversion-aware Apache server

- https:// -- Same as http://, but with SSL encryption
- svn:// -- Access via custom protocol to an svnserve server
- svn+ssh:// -- Same as svn://, but through an SSH tunnel

**Connection Name**: Name for this connection. If you leave this box blank, the connection will be given a name based on the URL of the repository location.

**User Name**: User name known to the repository, if the repository requires user and password validation.

**Password**: Password for the specified user, or blank if a password is not required.

**Test Read Access**: Attempts to establish a connection for read access to the Subversion repository.

**Status**: Displays the result of the test (success or an error message).

**Related Topics**

Using Versioning (page 1-53)

## 3.126 Subversion: Delete Resources

Use this dialog box to delete the selected resources (such as a file or directory) in the repository.

**Comments**: Comment explaining the deletion.

**Related Topics**

Using Versioning (page 1-53)

## 3.127 Subversion: Edit Configuration File

This dialog box is displayed if you click Edit "server" in the Versioning: Subversion: General (page 1-43) preferences pane. You can modify the Subversion configuration file directly.

**Reset**: Discards any changes that you have made and leaves the dialog box open.

To save any changes and close the box, click **OK**; to discard any changes and close the box, click **Cancel**.

**Related Topics**

Using Versioning (page 1-53)

## 3.128 Subversion: Export Files

Use this dialog box to copy files from the Subversion repository to a local file system directory, or to copy working copies to a local file system directory.

**Working Copy Path**: The location of the files that will be copied for export. Only files that are under Subversion control will be exported.

**Destination Path**: A path that includes the directory where you want the files to be copied to.

**Related Topics**

Using Versioning (page 1-53)

# 3.129 Subversion: Export Subversion Connections

Use this dialog box to export the details of one or more current Subversion connections to a file. The details can subsequently be imported from the file to re-create the connections.

**File Name**: The location and name for the file that will contain the connection details, or browse to a file/location using the **Browse** button.

**Connections**: Select one or more connections whose details will be exported.

**Related Topics**

Using Versioning (page 1-53)

# 3.130 Subversion: History

This dialog box displays version history information about Subversion files.

**Related Topics**

Using Versioning (page 1-53)

# 3.131 Subversion: Ignore

Use this dialog box to mark a file, or a pattern that identifies common file names, as content that Subversion should ignore. (This dialog box sets the `svn:ignore` property for the specified content.)

Often, a directory contains files that should not be kept under version control. For example, log files from a debug or batch operation do not need to be tracked or merged, yet they are often in the same directory as the shared code for a project. Such files should be marked to be ignored by Subversion.

**Related Topics**

Using Versioning (page 1-53)

# 3.132 Subversion: Import Subversion Connections

Use this dialog box to import the details of Subversion connections from a previously created file.

**File Name**: The location and name for the file that contains the connection details, or browse to a file/location using the **Browse** button.

**Connections**: Select one or more connections whose details will be imported. If a connection to be imported already exists with the same URL, you will be asked to confirm whether you want to overwrite the existing connection details with the details in the imported connection.

**Related Topics**

Using Versioning (page 1-53)

# 3.133 Subversion: Import to Subversion

Use this wizard to import source files into the Subversion repository To go from one step to the next, click Next; to go back to the previous step, click Back.

### Destination

Use to identify the Subversion repository, and directory within the repository, where the imported files will be stored.

**Repository Connection**: The connection for the Subversion repository in which you want to store the imported files.

**Path**: The directory within the Subversion repository for storing the imported files.

### Source

**Source Directory**: The directory containing the source files that you want to import into Subversion. Initially contains a path based on the item that was selected when you launched the wizard.

**Comments**: Comment text to accompany the imported files. The comments are recorded with the files in the Subversion repository and will be viewable with the version history of the files. You must enter some comment text; otherwise, an error will occur when you click Finish to attempt to perform the import operation.

### Filters

Filters that will be applied to the import operation. If you do not want one or more of the filters to be applied, move them from Selected Filters to Available Filters using the left arrow keys. If necessary, you can use the right arrow keys to move filters from Available Filters to Selected Filters.

**New**: Displays a dialog box in which you can create a new filter that will be applied to the import operation. New filters are added to the Selected Filters list.

### Options

You can configure options specific to the import operation.

**Do Not Recurse**: If this option is enabled, it prevents files being imported from directories subordinate to the one you identified on the Source page.

**Perform Checkout** : If this option is enabled, the imported source files will be checked out after import.

### Summary

Displays the selected options for the import operation. To make any changes, click Back. To perform the operation, click Finish.

### Related Topics

Using Versioning (page 1-53)

# 3.134 Subversion: Lock Resources

Use this dialog box to perform a Subversion lock operation on one or more checked out files (working copies).

**Files List**: Lists the names and physical locations of the files to be locked. You can individually select and deselect files.

**Steal Lock**: Breaks any existing locks and relocks the files for your use. Causes the `--force` option to be added to the underlying `svn lock` command.

**Comments**: Comments to accompany the action.

**Related Topics**

Using Versioning (page 1-53)

# 3.135 Subversion: Merge

A merge operation copies changes made in one branch to another branch, or copies changes from a branch to the trunk (main line of development). It is typically used to bring another developer's work into your own files, and to merge private development back into the main line of development.

The merge is created by comparing the content of two revisions within the Subversion repository, and applying the differences to a Subversion working copy. If you subsequently want to use the result of the merge in the main line of development, you commit the working copy to the Subversion repository in the usual way.

Specify the following:

- **Merge Type**: *Merge Selected Revision Range*, *Reintegrate a Branch*, or *Merge Two Different Trees*.
- **Merge Resource**
- **Merge Options**

Your selection for Merge Type affects the content of subsequent displays, which can include the following.

**From URL and its (start) revision to merge**: The resource that is the basis of the comparison. (The resource entered in the To URL box will be compared against the resource entered here.)

**HEAD Revision from Repository**: Causes the comparison to be made against the most recently committed resources in the Subversion repository.

**Use Revision**: Causes the comparison to be made against a resource in the Subversion repository with a particular revision number. When selected, the accompanying text box becomes available. You can then enter a revision number into the text box, or click the List Revisions button to select the revision that you require.

**To URL and its (end) revision to merge**: The resource that will be compared with the base resource selected in the From URL box.

**Same as "From" URL**: Uses the same base repository location for both elements of the comparison.

**Ignore Ancestry**: Ignores any relationships between resources in the Subversion repository when comparing the start and end revisions. The effect of this will be to retain resources that have names identical to those they are being compared with, even though the resources have no common ancestry. The alternative is that a resource that predates an identically named one may be deleted and replaced with the later resource.

**Dry Run**: Causes the comparison to be performed without the changes being applied to the Subversion working copy. The results of the comparison are displayed in the Messages - Log window.

**Related Topics**

Using Versioning (page 1-53)

## 3.136 Subversion: Pending Changes

This window shows files that have been added, modified or removed, either locally or remotely; files whose content conflicts with other versions of the same file; and files that have not been added to source control. This window is opened automatically when you first initiate an action that changes the local source control status of a file. You can also open this window manually.

The window shows any **outgoing changes** (files that have been added, modified or removed locally, and local files whose content conflicts with remote files), **candidates** (files that have been created locally but not yet added to source control), and **incoming changes** (files that have been added, modified or removed at a remote location).

You can restrict the files shown in the Pending Changes window by selecting a **scope** from the drop-down list. The default scope is Active Application, which will show files from the application currently selected in the navigator.

You can carry out appropriate source control actions on the files listed in the window, using the buttons in the toolbar. Clicking a button will either immediately initiate the operation, or open a dialog box through which you can choose options before proceeding.

**Related Topics**

Using Versioning (page 1-53)

## 3.137 Subversion: Properties

This dialog box is displayed if you right-click a node under a connection in the Versions navigator and select Properties. It displays properties and property values for the selected object.

**Related Topics**

Using Versioning (page 1-53)

## 3.138 Subversion: Remove from Subversion

Use this dialog box to begin the process of removing the listed files from the Subversion repository.

After you have clicked **OK**, the listed files will appear on the Outgoing tab of the Pending Changes window. The files will be removed from the Subversion dialog when you next commit the individual files or the working copy that they are part of.

**Related Topics**

Using Versioning (page 1-53)

# 3.139 Subversion: Repository Browser

Use this dialog box to select the location of a Subversion repository when using the branching, merging, and switching facilities. Locations in this dialog are shown as directories and objects. The chosen location is ultimately returned from this dialog as a URL.

**Repository Connection**: If the required location already exists, select it from the browser tree.

To create a new location, navigate to a parent directory, then select the Create New Remote Directory icon. This opens a dialog box that will show the location of the parent object (in the form of a URL) and let you name a directory beneath that one that will become the new location.

**Related Topics**

Using Versioning (page 1-53)

# 3.140 Subversion: Revert Local Changes

Use this dialog box to revert files to their previous state.

If the contents of a file have been changed, the file will be reverted to its base revision. If a file has been added but not yet committed, it will revert to unadded status. If a file is scheduled for removal (in the Pending Changes window), it will be added back to the navigator and given its previous status

**Files List**: Lists the names and physical locations of the files that will be reverted.

**Recursive**: Select if you want the revert operation to recurse into child objects of those selected.

**Related Topics**

Using Versioning (page 1-53)

# 3.141 Subversion: Switch

Use this dialog box to update the current working copy of the specified file from the specified repository and revision.

**From URL**: Full URL for the repository location associated with the current working copy.

**To URL**: Full URL for the repository location to use to update the current working copy.

**HEAD Revision**: Causes the HEAD revision (the latest revision in the repository) to be used for the update operation.

**Use Revision**: Causes the revision specified in the text box to be used for the update operation. To see a list of revisions from which you can choose, click **List Revisions**.

**Related Topics**

Using Versioning (page 1-53)

## 3.142 Subversion: Unlock Resources

Use this dialog box to perform a Subversion unlock operation on one or more locked, checked out files (working copies).

**Files List**: Lists the names and physical locations of the files to be unlocked. You can individually select and deselect files.

**Force Unlock**: Breaks any existing locks and unlocks the files. Causes the `--force` option to be added to the underlying `svn unlock` command.

**Related Topics**

Using Versioning (page 1-53)

## 3.143 Subversion: Update Resources

Use this dialog box to incorporate into your local copies changes that other developers have committed to the Subversion repository.

**Files List**: Lists the names and physical locations of the files that will be updated with content from the Subversion repository.

**Use Revision**: Updates the files with content from a particular revision within the Subversion repository. Enter the revision number in the adjacent text box. If not selected, the files will be updated from the HEAD revision.

**Ignore Externals**: Select if you do not want the update operation to apply to external working copies created as the result of externals definition handling. Externals definitions are used to pull data from multiple repositories. See the Subversion documentation for details.

**Recursive**: Deselect if you do not want the update operation to recurse into child objects of those selected.

**Related Topics**

Using Versioning (page 1-53)

## 3.144 Subversion: Update Working Copy

Use this dialog box to update individual files with content from the Subversion repository.

**Files List**: Lists the names and physical locations of the files that will be updated with content from the Subversion repository.

**Use Revision**: Updates the files with content from a particular revision within the Subversion repository. Enter the revision number in the adjacent text box. If not selected, the files will be updated from the HEAD revision.

**Ignore Externals**: Select if you do not want the update operation to apply to external working copies created as the result of externals definition handling. Externals definitions are used to pull data from multiple repositories. See the Subversion documentation for details.

**Recursive**: Deselect if you do not want the update operation to recurse into child objects of those selected.

**Related Topics**

Using Versioning (page 1-53)

## 3.145 Subversion: Versioning Properties

This dialog box displays general and versioning information about the currently selected file or folder.

**Related Topics**

Using Versioning (page 1-53)

## 3.146 Subversion: XML Metadata Comparator

Use this dialog box to resolve conflicts when there are conflicts in the values for certain properties.

**Filter**: Lets you control the list of properties displayed.

**Consider "Changed Time" Property**: If this option is enabled, two properties with identical content but different timestamps for the last change are considered to be different and thus potentially in conflict. If this option is not enabled, creation timestamps are ignored in comparing properties.

**Details** tab: For each property, shows the property name, a Selected box indicating whether it is to be merged if you click Merge, and the values for the property in the left and right columns shown at the top of the box.

**Merge**: Merges the selected property definitions

**Close**: Closes the dialog box without performing a merge.

**Related Topics**

Using Versioning (page 1-53)

## 3.147 Subview Properties

This dialog box displays the properties of a subview. (See also Logical Diagram and Subviews (page 1-17) and Relational Diagram and Subviews (page 1-20).)

**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 Common Information in Dialog Boxes (page 3-9).

# 3.148 Table Properties

This dialog box displays the properties of a table, which is part of Relational Models (page 1-19).

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.

**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.

**Generate in DDL**: Controls whether statements to create the table are included when DDL is generated.

**Engineer as Relationship**: Controls whether relationship attributes are created during engineering. (Relationships can be shown or hidden on diagrams.) If the table complies with the criteria for an intersection table, selecting this option will cause the table to be engineered as a many-to-many (m:n) relationship in which non-foreign-key columns become attributes of the relationship.

**Allow Columns Reorder During Engineering**: If this option is enabled, Data Modeler can reorder the attributes of the associated entity when the table is engineered to the logical model, for example, to place attributes considered more important first. (This behavior can be especially useful with tables that contain many columns.) If this option is not enabled, entity attributes are placed in the same order as their associated columns in the table definition.

**Materialized Query Table**: Indicates whether the table has an associated SQL query. For Oracle databases. this option indicates whether the table is implemented as a materialized view.

### Columns

#### Details tab

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.

#### Overview tab

Lists each column, its data type, and if it is a primary key, foreign key, or mandatory.

#### Security tab

Lists each column and any relevant security-related properties for each: whether it contains personally identifiable information (PII), contains sensitive information, should be masked when displayed, and the mask template to use (with the list containing any appropriate mask templates; see Mask Templates Administration).

### Primary Key

Shows the current primary key (if any) of the table, and lets you change the primary key.

### Unique Constraints

Lists any unique constraints. You can add, modify, and delete unique constraints. For each constraint, specify the column whose values must be unique or multiple columns that must have unique combinations of values.

### 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.

### Table Level Constraints

Lists any table-level constraints that are defined by a validation rule (an expression that must evaluate to true for the data to be valid).

### 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.

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

### 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 and Graph 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 (page 3-78).

### Column Groups

Displays information about column groups, which can be used to group related columns for possible use in generating a user interface. For example, a column group named *Name* could include columns first_name and last_name, and a column group named *Address* could include columns street_address, city, state, and postal_code.

To add a column group, click the Add (+) icon, specify the column group name, select the desired columns and move them to the right side, and optionally enter descriptive text in the Notes box; to delete a column group, select its entry and click the Remove (X) icon.

### Comments in RDBMS

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

### Scripts

Enables you to specify SQL statements 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. For example, for After Create for a table named MY_TABLE, you might specify the following statement (and also check **Include into DDL Script**):

```
INSERT INTO my_table SELECT * FROM your_table
```

**Include into DDL Script**: Controls whether the text of the specified statements is included in the generated DDL script for creating the table.

### Classification Types

Classification type for the entity: Fact, External, Dimension, Logging, Summary, and Temporary.

You can specify colors to be used in diagrams for each classification type in the Diagram: Classification Types (page 3-22) design properties.

### Redaction Policy

**Policy Name**: Name for the redaction policy.

**Enabled**: Indicates if the policy is enabled.

**Generate in DDL**: Indicates whether the redaction policy is considered when generating DDLs.

**Expression**: Expression defined to determine whether any real data is displayed to users. If this field is left blank, data that is not masked is displayed to users.

> **See Also:**
>
> http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/sqldevdm/r40/Mask/LAB_12cDataModeler.html

### Comments, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Summary

See Common Information in Dialog Boxes (page 3-9).

# 3.149 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 *jv1*, *jv2*, 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.

# 3.150 Table DDL Transformation Scripts

Lets you create, test, and debug scripts that can be included in generated DDL for a table. It is displayed when you select Tools > Design Rules and Transformations > Table DDL Transformations.

**Active Script Set**: Lists available script sets, and lets you add and delete sets in the list.

**Set Details**: Shows details about the selected script set, and lets you modify that set's name and description, and define one or more custom scripts for **Before Create**, **Instead of Create**, **After Create**, and **End of Script**. (Instead of Create suppresses the built-in generation of DDL for tables and related components, so that you can implement DDL generation for nonsupported databases.)

**Script**: Lets you display and modify details for a script within the selected script set. Select the table to test, its library (if any), and the method to use (if any). Enter and edit the actual script code in the text box.

Buttons:

- **Test**: Tests the script; opens a DDL preview window and shows the generated DDL.
- **Debug**: Allows debugging; does not include a DDL preview.
- **Open**: Opens a DDL preview window.

For more detailed information, see: https://community.oracle.com/community/database/developer-tools/sql_developer_data_modeler/blog/2015/03/12/oracle-sql-developer-data-modeler-41-user-defined-ddl-generation-using-transformation-scripts

# 3.151 Telephone Properties

This dialog box displays the properties of a telephone object, which is a type of Business Information (page 1-25) 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 (page 3-12).)

### Comments, Summary

See Common Information in Dialog Boxes (page 3-9).

# 3.152 Transformation Package

This dialog box displays the properties of a transformation package object, which is part of the Process Model (page 1-15) and is described in Transformation Processes and Packages (page 1-16).

**Name**: Name of the transformation package.

**Comment**: Optional descriptive comment text.

# 3.153 Transformation <task-name>

This dialog box displays the properties of a transformation task object, which is part of the Process Model (page 1-15).

### General

**Name**: Name of the transformation task.

**Comment**: Optional descriptive comment text.

**Visible**: Controls whether the transformation task is displayed in the Data Modeler window.

### 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 (page 3-99).)

## 3.154 Transformation Properties

This dialog box displays the properties of a transformation, which is part of the Process Model (page 1-15).

### 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 (page 3-52).)

### Processed Attributes

Lists any attributes processed by the transformation. To view the properties of an attribute, double-click its name. (See also Attribute Properties (page 3-3).)

### Processed Entities

Lists any entities processed by the transformation. To view the properties of an entity, double-click its name. (See also Entity Properties (page 3-32).)

### 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 Common Information in Dialog Boxes (page 3-9).

# 3.155 Transformation Flow Properties

This dialog box displays the properties of a transformation flow, which is part of the Process Model (page 1-15).

### 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 (page 3-99) 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 (page 3-52).)

### 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 Common Information in Dialog Boxes (page 3-9).

# 3.156 TSDP Policy Properties

This dialog box displays the properties of a TSDP (Transparent Sensitive Data Protection) policy. You should be familiar with the concepts and techniques explained in the "Using Transparent Sensitive Data Protection" chapter in *Oracle Database Security Guide*. You should also be familiar with the `DBMS_TSDP_PROTECT` package (described in *Oracle Database PL/SQL Packages and Types Reference*), which provides an interface to configure TSDP policies in conjunction with the `DBMS_TSDP_MANAGE` package.

### General

**Name**: Name of the policy.

**Generate in DDL**: Controls whether the policy creation is included when DDL statements are generated.

**Subpolicies**

You can specify one or more specific policies (subpolicies) to make up the TSDP policy. For each subpolicy, specify the name, expression, mask template, length, parent schema, and parent table.

You can optionally enter comments and notes about each subpolicy.

**Comments, Notes, Summary**

See Common Information in Dialog Boxes (page 3-9).

**Related Topics**

Sensitive Type Properties (page 3-75)

# 3.157 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 and 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 Data Modeler logical types.

# 3.158 Types to Domains

Creates domains based on data types in the selected models (logical, relational, or a combination). This dialog is displayed if you click Tools, then Types to Domains Wizard.

This wizard provides a convenient way to generate domains based on the types associated with attributes of entities in the logical model and columns of tables in relational models.

If you later want to edit or delete any of the generated domain definitions, you can do so using the Domains Administration (page 3-29) dialog box.

**Create New Domains**: Controls whether existing domain definitions are overwritten if a generated new domain has the same name as an existing domain. If this option is enabled, those existing domain definitions are overwritten; if this option is not enabled,

existing domains are not overwritten if the generated new domain would have the same name.

## 3.159 Unable to Connect

This box informs you that SQL Developer is unable to connect to the Internet. The cause might be that the connection information for the specified HTTP proxy server is invalid or the server in not available.

## 3.160 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 (page 1-16).

**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.

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

See Common Information in Dialog Boxes (page 3-9).

## 3.161 View Properties (Logical Model)

This dialog box displays the properties of a view in a Logical Model (page 1-16).

**General**

**Name**: Name of the view.

**Query Builder**: Click to display a query builder interface, in which you can create a view from the query by specifying tables and columns, as well as other information. In the query builder:

- You can drag and drop tables into the main area; click to select columns to be included; view, copy, and paste the query SQL; and edit the SQL, or make it read-only reflecting the main diagram.
- **Filter Metadata Objects by Diagram** makes it easier to find objects when there are very many metadata objects.
- **Show Structure Tree** shows a structure view of the query.
- **Show Criteria List** lets you specify criteria for the SELECT statement. Your specifications can affect the 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.

**View Type**: Type of view: entity view or named view.

#### Engineer To

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

**Engineer**: Controls whether the view 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.

#### Comments, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Dynamic Properties, Summary

See Common Information in Dialog Boxes (page 3-9).

## 3.162 View Properties (Relational Model)

This dialog box displays the properties of a view in Relational Models (page 1-19).

In relational model diagrams, a view might have an icon next to its name indicating its status:

- (No icon): The view is parsed and valid.
- Yellow triangle with exclamation mark: "Older style view" created with an earlier Data Modeler version, or imported; not yet parsed. To parse such a view, right-click its box in the diagram and select **Parse Older Style Views**.
- Red triangle with exclamation mark: Invalid view. For example, the name of an object in the view was changed after parsing, or the view contains incorrect syntax. To validate a view, right-click its box in the diagram and select **Validate Selected Views**.

#### General

**Name**: Name of the view.

**Query Builder**: Click to display a query builder interface, in which you can create a view from the query by specifying tables and columns, as well as other information. In the query builder:

- You can drag and drop tables into the main area; click to select columns to be included; view, copy, and paste the query SQL; and edit the SQL, or make it read-only reflecting the main diagram.
- **Filter Metadata Objects by Diagram** makes it easier to find objects when there are very many metadata objects.
- **Show Structure Tree** shows a structure view of the query.
- **Show Criteria List** lets you specify criteria for the SELECT statement. Your specifications can affect the 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.

**Schema**: Schema of the database user that owns the view. If not specified, the current schema is assumed.

**Include Schema Name in Query**: If a schema is specified, you can have objects in the view's query prefixed by that schema name.

**Use Objects Only From**: Name of the relational model from which to use objects specified in the view.

**Allow Type Substitution**: If applicable, controls whether a substitutional structured type generation in the DDL is allowed.

**Generate in DDL**: controls whether the view definition is included when DDL is generated.

**Test Query**: Displays a dialog box in which you can specify a database connection and test the view's query.

### 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.

### 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, Impact Analysis, Measurements, Change Requests, Responsible Parties, Documents, Dynamic Properties, Summary**

See Common Information in Dialog Boxes (page 3-9).

# 3.163 View to Table

Creates tables based on views in a selected relational model. This dialog is displayed if you click Tools, then View to Table Wizard.

The views in the selected relational model are displayed. Select all views or a subset, optionally specify any naming rule, and click **Generate**. A table is created for each selected view. Each table selects all columns from the view, although you can later modify the view definition as needed. By default (if no Naming Rule is specified), each table name is its associated view name, prefixed with *T_*. For example, if a view name is BOOKS, the associated table name by default will be T_BOOKS.

**Naming Rule**: Displays any variables (relational model name or view name, or both in the specified order) that will be prefixed to the table name instead of *T_*. 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 view name is BOOKS:

- If you specify only view name, the name of the generated table will be BOOKS. That is, all tables will have the same name as their associated views.
- If you specify only model name, the name of the generated table will be Library. You should probably not choose this option, because all tables will have names starting with the model name and *jt1*, *jt2*, and so on added to names after the first one.
- If you specify model name and view name, the name of the generated table will be LibraryBOOKS.

# 3.164 Windows

This dialog box is displayed if you right-click a tab on the right side of the Data Modeler window and select Windows. It lists all currently open editing panes.

**Activate**: Moves the focus to the selected pane and leaves the dialog box open.

**Close**: Closes the selected pane and leaves the dialog box open.

If you double-click a name, focus is moved to that pane and the dialog box is closed.

# Index

## A


accelerator (shortcut) keys, *1-42*, *1-43*
accelerator keys
    for menus, *1-3*
Apply Custom Transformation Scripts command, *1-8*
arcs
    definition, *1-19*
    properties, *3-2*
attributes
    definition of, *1-18*
    properties, *3-3*
auto route
    adjusting lines in diagrams, *1-8*
    context menu command, *1-8*
AutoLayout
    context menu command, *1-8*


## B


Bachman notation, *1-17*
background color
    in diagrams, *1-9*
Barker notation, *1-17*
Berkeley DB
    data storage for Subversion repository, *3-84*
bind variables
    for reports, *1-52*
blogs
    SQL Developer, *1-56*
bottom-up modeling, *1-29*
business information, *1-25*


## C


cardinality
    definition of, *1-18*
change request
    administration, *3-5*
    properties, *3-4*
Change Subview Object Names Prefix command, *1-8*
Check for Updates feature, *3-5*
classification types
classification types *(continued)*
    specifying colors for in diagrams, *3-22*
collection type
    properties, *3-6*
collection types, *1-14*
color
    background in diagrams, *1-9*
columns
    properties, *3-7*
comparing models, *1-4*
composite process
    definition of, *1-15*
configurations
    export, *3-74*
    report, *3-79*
connections
    creating, editing, or selecting, *3-64*
    Oracle Database, *3-64*
contacts
    definition of, *1-26*
    properties, *3-12*
context menus (right-click menus), *1-8*
cubes
    properties, *3-15*
custom design rules, *3-25*
custom report templates, *3-17*
custom transformation scripts
    applying, *1-8*
custom transformations, *3-26*
customizing Data Modeler
    setting preferences, *1-31*


## D


data flow
    definition of, *1-15*
data flow diagrams, *1-16*
Data Modeler preferences, *1-31*
data modeling
    concepts and usage information, *1-1*
data redaction
    mask templates administration, *3-57*
data store
    definition of, *1-15*
data types

data types *(continued)*
administering, *1-4*, *3-101*
model, *1-11*
database connections
creating, editing, or selecting, *3-64*
Oracle Database, *3-64*
DDL file editor, *3-20*
DDL files
selecting for import, *3-74*
DDL generation
options, *3-21*
default size
adjusting sizes and shapes to fit diagram window, *1-6*
defaultdomains.xml file, *3-47*
design rules
checking a design for violations, *1-7*
custom, *3-25*
explanation and usage information, *3-24*
diagram color
background, *1-9*
diagrams
adjusting lines in (auto route setting), *1-8*
dimensions
properties, *3-26*
directories (directory objects), *1-23*
Discover Foreign Keys command, *1-8*
discussion forum
SQL Developer, *1-56*
displays, *1-8*
distinct types, *1-13*
properties, *3-28*
documents
definition of, *1-26*
properties, *3-28*
domains
administering, *1-4*, *3-29*
importing, *3-47*
preferred, *1-36*
properties, *3-29*
types to domains wizard, *3-101*
dynamic properties, *3-9*

## E

elbows
adding vertices in diagrams, *1-8*
disabling Auto Route before adding vertices in diagrams, *1-8*
removing vertices in diagrams, *1-8*
emails
definition of, *1-26*
properties, *3-30*
entities
definition of, *1-18*
entities *(continued)*
properties, *3-32*
events
adding, *3-1*
properties, *3-34*
export configurations, *3-74*
external agents
definition, *1-15*
properties, *3-39*
external data
properties, *3-41*

## F

F10 key
for File menu, *1-3*
fit screen
adjusting sizes and shapes to fit diagram window, *1-6*
fixing (making unchangeable) object names, *3-65*
flows
properties, *3-42*
showing name in diagrams, *1-9*
foreign keys
discovering, *1-8*
properties, *3-43*
forward engineering, *1-8*

## G

glossaries
concepts (explanations), *3-44*
creating, *1-4*
editing, *1-4*
glossary
importing, *3-47*
glossary editor, *3-44*
graphical user interface (GUI)
Data Modeler, *1-2*
grid
showing and hiding, *1-8*
grid size
option, *1-36*

## H

hierarchies
properties, *3-46*

## I

identifying relationship, *3-69*
identity column, *3-7*
indexes

indexes *(continued)*
definition of, *1-23*
properties, *3-49*
information store
definition of, *1-15*
information stores
properties, *3-50*
information structures
properties, *3-52*
inheritance
definition of, *1-18*

## J

JDBC drivers (third party), *1-40*
joins
properties, *3-53*

## K

keyboard shortcuts, *1-42*, *1-43*
keys
properties, *3-102*

## L

layout
automatically rearranging in diagrams (AutoLayout), *1-8*
levels
properties, *3-54*
libraries
transformation, *3-26*
line auto route
adjusting lines in diagrams, *1-8*
context menu command, *1-8*
locations
definition of, *1-26*
properties, *3-55*
logical model, *1-16*
logical types, *1-14*
preferred, *1-36*

## M

mask templates
administering, *3-57*
measure folders
properties, *3-57*
measurements
properties, *3-59*
measures
properties, *3-58*
merging models, *1-4*
methods
properties, *3-59*
modeling
data modeling concepts and usage information, *1-1*
Multimedia
Oracle Multimedia logical types, *1-14*

## N

name abbreviations, *3-63*
specifying file with, *1-4*
name translation
applying during engineering, *3-30*
naming standard definitions
importing, *3-47*
neighbors
selecting in diagrams, *1-8*

## O

object names
administration, *3-65*
fixing (making unchangeable), *3-65*
object names prefix
changing, *1-8*
older style views
parsing, *3-103*
Oracle by Example (OBE)
Data Modeler, *1-56*
Oracle Database connections, *3-64*
Oracle Multimedia
logical types, *1-14*
ORD* types
Oracle Multimedia logical types, *1-14*
OTN page
SQL Developer, *1-56*

## P

page grid
showing and hiding, *1-8*
physical models, *1-22*
preferences
customizing Data Modeler, *1-31*
exporting and importing, *1-47*
restoring to original settings, *1-48*
preferred domains, *1-36*
preferred logical types, *1-36*
prefixes
Change Subview Object Names Prefix command, *1-8*
primitive process
definition of, *1-15*
process model, *1-15*

processes
definition, 1-15
properties, 3-66

## Q

query builder, 3-102, 3-103

## R

RDBMS site
definition of, 1-22
editing, 3-68
RDBMS sites
administering, 1-4
record structures
properties, 3-68
redaction
mask templates administration, 3-57
relational models, 1-19
relations
definition of, 1-18, 1-21
properties, 3-69
relationship
identifying, 3-69
report configurations, 3-79
report templates, 3-17, 3-70
reporting repository, 1-48
reports, 1-48
bind variables for, 1-52
generating as RTF, HTML, or PDF, 1-48
resource locators
definition of, 1-26
properties, 3-71
responsible parties
definition of, 1-27
properties, 3-71
reverse engineering, 1-8
right-click menus (context menus), 1-8
role
definition of, 1-15
roles
properties, 3-72
rollup links
properties, 3-73
rule sets, 3-24
properties, 3-73

## S

Saxon XSLT 2.0 .jar file
specifying, 1-39
schemas
properties, 3-73
search profiles, 3-73
select neighbors
context menu command, 1-8
Select Relational Models dialog box
controlling display of, 1-33
sensitive type
properties, 3-75
sequences
definition of, 1-24
settings
exporting and importing, 1-47
shortcut keys, 1-42, 1-43
for menus, 1-3
restoring to default scheme, 1-42
slices
properties, 3-77
spatial definitions
properties, 3-78
SQL Access to Oracle AW
properties, 3-78
standard report templates
modifying, 3-70
Start Page for Data Modeler, 1-55
structured types, 1-14
Subversion
Data Modeler support for, 1-53
preferences for, 1-43
subviews
adding elements to, 3-1
creating from selected objects, 1-8
properties, 3-92
synchronize with tree
option, 1-36
synonyms
creating, 1-9

## T

table to view wizard, 1-4, 3-97, 3-105
tables
definition of, 1-21, 1-25
properties, 3-93
targeted modeling, 1-30
telephones
definition of, 1-27
properties, 3-98
templates
report, 3-17, 3-70
third party JDBC drivers, 1-40
top-down modeling, 1-27
transformation flows
properties, 3-100
transformation packages, 1-16
properties, 3-98
transformation processes, 1-16
transformation tasks

transformation tasks *(continued)*
definition, *1-15*
properties, *3-98*
transformations
custom, *3-26*
libraries, *3-26*
properties, *3-99*
TSDP (Transparent Sensitive Data Protection) policies
properties, *3-100*
sensitive type properties, *3-75*
tutorial
creating models for a small database, *2-1*
type substitution, *1-19*
types
administering, *1-4*, *3-101*
data types model, *1-11*
types to domains wizard, *3-101*

## U

unique identifiers (UIDs)
definition of, *1-18*
properties, *3-102*
updates
checking for SQL Developer updates, *3-5*
URLs
resource locator objects, *1-26*
resource locator properties, *3-71*
URLs *(continued)*
usage reporting, *1-43*
user interface (UI)
Data Modeler, *1-2*

## V

versioning
Data Modeler support for, *1-53*
preferences for, *1-43*
vertices (elbows)
disabling Auto Route before adding elbows in diagrams, *1-8*
view to table wizard, *1-4*
viewlets
Data Modeler, *1-56*
views
definition of, *1-25*
older style, *3-103*
properties, *3-102*, *3-103*
table to view wizard, *3-97*, *3-105*
validating, *3-103*
virtual columns (Computed property for a column), *3-7*

## Z

zoom in, *1-6*
zoom out, *1-6*