

Oracle® SQL Developer

User's Guide

Release 2.1

E15222-02

February 2010

Provides conceptual and usage information about Oracle SQL Developer, a graphical tool that enables you to browse, create, edit, and delete (drop) database objects; run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export data; migrate third-party databases to Oracle; view metadata and data in third-party databases; and view and create reports.

Oracle SQL Developer User's Guide, Release 2.1

E15222-02

Copyright © 2006, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Chuck Murray

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

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

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

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

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

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

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

Contents

Preface	xv
Audience	xv
Documentation Accessibility	xv
Product Accessibility	xvi
Related Documents	xvi
Conventions	xvi
Third-Party License Information	xvi
1 SQL Developer Concepts and Usage	
1.1 Installing and Getting Started with SQL Developer	1-2
1.2 SQL Developer User Interface	1-2
1.2.1 Menus for SQL Developer	1-6
1.2.2 Restoring the Original "Look and Feel"	1-9
1.3 Database Objects	1-10
1.3.1 Applications (Application Express 3.0.1 and Later)	1-10
1.3.2 Cache Groups (Oracle TimesTen In-Memory Database)	1-11
1.3.3 Database Links (Public and Private)	1-11
1.3.4 Directories	1-11
1.3.5 Editions	1-11
1.3.6 Functions	1-12
1.3.7 Indexes	1-12
1.3.8 Java Sources	1-13
1.3.9 Jobs	1-13
1.3.10 Materialized Views	1-13
1.3.11 Materialized View Logs	1-13
1.3.12 Packages	1-14
1.3.13 Procedures	1-14
1.3.14 Queues	1-15
1.3.15 Queue Tables	1-15
1.3.16 Recycle Bin	1-15
1.3.17 Replication Schemes (Oracle TimesTen In-Memory Database)	1-15
1.3.18 Sequences	1-16
1.3.19 Synonyms (Public and Private)	1-16
1.3.20 Tables	1-16
1.3.20.1 Flashback Table Support	1-17

1.3.21	Triggers	1-18
1.3.22	Types.....	1-18
1.3.23	Users (Other Users)	1-18
1.3.24	Views	1-18
1.3.25	XML DB Repository.....	1-18
1.3.26	XML Schemas	1-19
1.3.27	Captured and Converted Database Objects (for Migration)	1-19
1.4	Database Connections	1-19
1.4.1	Using Folders to Group Connections	1-21
1.4.2	Sharing of Connections	1-22
1.4.3	Advanced Security for JDBC Connection to the Database	1-22
1.4.4	Connections with Operating System (OS) Authentication.....	1-22
1.4.5	Connections with Proxy Authentication.....	1-22
1.5	Entering and Modifying Data	1-23
1.6	Running and Debugging Functions and Procedures	1-25
1.6.1	Using Bookmarks When Editing Functions and Procedures.....	1-27
1.6.2	Remote Debugging.....	1-27
1.6.3	Displaying SQL Trace (.trc) Files.....	1-27
1.6.4	Using the PL/SQL Hierarchical Profiler	1-28
1.6.5	Setting Expression Watches	1-28
1.7	Using the SQL Worksheet.....	1-28
1.7.1	SQL*Plus Statements Supported and Not Supported in SQL Worksheet.....	1-31
1.7.2	Script Runner.....	1-32
1.7.3	Execution Plan.....	1-33
1.7.4	Autotrace Pane.....	1-33
1.7.5	DBMS Output Pane	1-33
1.7.6	OWA Output Pane	1-34
1.7.7	SQL History	1-34
1.7.8	Gauges: In the SQL Worksheet and User-Defined Reports.....	1-35
1.8	Using Snippets to Insert Code Fragments.....	1-36
1.8.1	User-Defined Snippets	1-36
1.9	Finding Database Objects	1-37
1.10	Using Versioning	1-37
1.10.1	About CVS and SQL Developer	1-38
1.10.1.1	Pending Changes (CVS)	1-38
1.10.2	About Subversion and SQL Developer	1-38
1.11	SQL Developer Reports.....	1-39
1.11.1	About Your Database reports	1-40
1.11.2	All Objects reports	1-40
1.11.3	Application Express reports.....	1-41
1.11.4	ASH and AWR reports.....	1-41
1.11.5	Charts reports.....	1-41
1.11.6	Database Administration reports	1-41
1.11.7	Data Dictionary reports	1-42
1.11.8	Jobs reports	1-42
1.11.9	PL/SQL reports.....	1-42
1.11.10	Security reports	1-43

1.11.11	Streams reports	1-43
1.11.12	Table reports.....	1-43
1.11.13	XML reports.....	1-45
1.11.14	Migration reports.....	1-45
1.11.15	User Defined reports	1-45
1.11.15.1	User-Defined Report Example: Chart.....	1-46
1.11.15.2	User-Defined Report Example: Dynamic HTML.....	1-47
1.12	SQL Developer Preferences	1-48
1.12.1	Environment.....	1-48
1.12.2	Code Editor.....	1-49
1.12.3	Compare and Merge.....	1-51
1.12.4	Database	1-52
1.12.5	Debugger.....	1-57
1.12.6	Extensions	1-58
1.12.7	External Editor	1-58
1.12.8	File Types	1-58
1.12.9	Global Ignore List	1-59
1.12.10	Migration.....	1-59
1.12.11	Mouseover Popups.....	1-62
1.12.12	Shortcut Keys (Accelerator Keys).....	1-62
1.12.13	Unit Test Parameters	1-63
1.12.14	Versioning.....	1-63
1.12.15	Web Browser and Proxy	1-66
1.13	Location of User-Related Information	1-66
1.14	Data Modeler Viewer (Read-Only)	1-67
1.15	Oracle TimesTen In-Memory Database Support.....	1-67
1.16	Using the Help.....	1-68
1.17	Tip of the Day	1-68
1.17.1	SQL History Shortcuts	1-68
1.17.2	Unshared Worksheets.....	1-69
1.17.3	SQL Worksheet Bookmarks	1-69
1.17.4	Formatted Display of SQL Trace (.trc) Files.....	1-69
1.17.5	Folders for Organizing Connections.....	1-69
1.17.6	Third-Party Databases and SQL Developer.....	1-69
1.17.7	Debugger Ports and Firewalls	1-69
1.17.8	Viewing Multiple Tables	1-69
1.17.9	Customizing SQL Developer Appearance.....	1-69
1.17.10	Maximizing Tab Panes.....	1-69
1.17.11	Default Path for Running Scripts	1-70
1.17.12	Shutting Down and Restarting the Database	1-70
1.17.13	Feature Requests	1-70
1.17.14	Discussion Forum	1-70
1.17.15	Help Text Font Size	1-70
1.17.16	Procedure and Function Signatures.....	1-70
1.17.17	Type-Ahead in Navigators.....	1-70
1.17.18	Extended Paste	1-71
1.17.19	Closing Tabbed Windows Using the Mouse Wheel.....	1-71

1.17.20	Go to Last Edit Location	1-71
1.17.21	Closing Tabbed Windows Using the Context Menu.....	1-71
1.17.22	List of All Open Windows.....	1-71
1.17.23	Go to Subprogram Implementation from Package Window	1-71
1.17.24	Select Multiple Table or Column Names in Completion Insight	1-71
1.18	For More Information.....	1-71

2 Migrating Third-Party Databases

2.1	Migration Quick Start.....	2-1
2.1.1	Standard Migration	2-2
2.1.2	Quick Migration.....	2-3
2.2	Overview of Migration.....	2-5
2.2.1	How Migration Works	2-5
2.2.2	Migration Implemented as SQL Developer Extensions.....	2-6
2.3	Preparing a Migration Plan	2-6
2.3.1	Task 1: Determining the Requirements of the Migration Project	2-6
2.3.2	Task 2: Estimating Workload	2-8
2.3.3	Task 3: Analyzing Operational Requirements	2-9
2.3.4	Task 4: Analyzing the Application.....	2-9
2.3.5	Task 5: Planning the Migration Project.....	2-10
2.4	Before You Start Migrating: General Information.....	2-10
2.4.1	Creating a Database User for the Migration Repository.....	2-11
2.4.2	Requirements for Creating the Destination Oracle Objects.....	2-11
2.5	Before You Start Migrating: Source-Specific Information.....	2-12
2.5.1	Before Migrating From IBM DB2	2-12
2.5.2	Before Migrating From Microsoft SQL Server or Sybase Adaptive Server.....	2-13
2.5.3	Before Migrating From Microsoft Access.....	2-14
2.5.3.1	Creating Microsoft Access XML Files.....	2-15
2.5.4	Before Migrating From MySQL.....	2-15
2.5.5	Before Migrating From Teradata.....	2-16
2.6	Capturing the Source Database	2-16
2.6.1	Online Capture.....	2-17
2.6.2	Offline Capture	2-17
2.6.2.1	IBM DB2 Offline Capture Notes.....	2-17
2.7	Creating and Customizing the Converted Model.....	2-18
2.7.1	Correcting Errors in the Converted Model.....	2-19
2.8	Generating the DDL for the Oracle Schema Objects.....	2-19
2.9	Migrating the Data.....	2-19
2.9.1	Transferring the Data Offline.....	2-19
2.9.1.1	Creating Data Files From Microsoft SQL Server or Sybase Adaptive Server...	2-20
2.9.1.2	Creating Data Files From Microsoft Access.....	2-20
2.9.1.3	Creating Data Files From MySQL.....	2-20
2.9.1.4	Populating the Destination Database Using the Data Files.....	2-21
2.10	Making Queries Case Insensitive	2-24
2.11	Testing the Oracle Database	2-25
2.11.1	Testing Methodology	2-25
2.11.2	Testing the Oracle Database.....	2-26

2.11.2.1	Guidelines for Creating Tests	2-28
2.11.2.2	Example of a Unit Test Case	2-28
2.12	Deploying the Oracle Database	2-29
2.12.1	Choosing a Rollout Strategy.....	2-29
2.12.1.1	Phased Approach	2-29
2.12.1.2	Big Bang Approach	2-30
2.12.1.3	Parallel Approach.....	2-30
2.12.2	Deploying the Destination Database	2-30
2.13	Using Migration Reports	2-31
2.14	SQL Developer User Interface for Migration.....	2-31
2.14.1	Migration Menu	2-33
2.14.2	Other Menus: Migration Items	2-33
2.14.3	Migration Preferences	2-33
2.14.4	Migration Log Panes	2-34
2.14.5	Using the Translation Scratch Editor	2-34

3 Unit Testing with SQL Developer

3.1	Overview of Unit Testing	3-1
3.2	SQL Developer User Interface for Unit Testing	3-2
3.2.1	Unit Test Submenu	3-3
3.2.2	Other Menus: Unit Test Items.....	3-3
3.2.3	Unit Test Preferences.....	3-4
3.3	Unit Test Repository	3-4
3.3.1	Managing Repository Users and Administrators.....	3-4
3.4	Editing and Running a Unit Test.....	3-5
3.5	Using a Dynamic Value Query for Seed Data	3-5
3.6	Using Lookups to Simplify Unit Test Creation	3-7
3.6.1	Providing Values for Input Fields.....	3-7
3.6.2	Automatically Creating Implementations	3-8
3.7	Using Variable Substitution in Validation Actions.....	3-8
3.8	Unit Test Library	3-9
3.9	Unit Test Reports	3-10
3.10	Exporting and Importing Unit Test Objects.....	3-10
3.11	Using the Command-Line Interface.....	3-11
3.12	Example of Unit Testing (Tutorial).....	3-11
3.12.1	Create the EMPLOYEES Table.....	3-13
3.12.2	Create the AWARD_BONUS Procedure.....	3-13
3.12.3	Create the Unit Testing Repository.....	3-14
3.12.4	Create a Unit Test	3-14
3.12.5	Run the Unit Test	3-15
3.12.6	Create and Run an Exception Unit Test	3-16
3.12.7	Create a Unit Test Suite.....	3-17
3.12.8	Run the Unit Test Suite	3-17

4 Tutorial: Creating Objects for a Small Database

4.1	Create a Table (BOOKS).....	4-2
-----	-----------------------------	-----

4.2	Create a Table (PATRONS)	4-3
4.3	Create a Table (TRANSACTIONS).....	4-4
4.4	Create a Sequence	4-6
4.5	Insert Data into the Tables.....	4-7
4.6	Create a View.....	4-8
4.7	Create a PL/SQL Procedure.....	4-8
4.8	Debug a PL/SQL Procedure	4-9
4.9	Use the SQL Worksheet for Queries	4-11
4.10	Script for Creating and Using the Library Tutorial Objects	4-12

5 Dialog Boxes for Creating/Editing Objects

5.1	Add Extension	5-1
5.2	Change Type.....	5-1
5.3	Check for Updates	5-2
5.4	Check Out from CVS	5-2
5.5	Choose Directory.....	5-3
5.6	Configure Component Palette	5-3
5.7	Create Palette Page	5-3
5.8	Confirm Drop Application	5-3
5.9	Confirm Running SQL	5-3
5.10	Connection Has Uncommitted Changes	5-3
5.11	Create New Object	5-4
5.12	Create/Edit CVS Connection.....	5-4
5.13	Create/Edit/Select Database Connection	5-5
5.14	Rename Model (Migration)	5-9
5.15	Rename Database Item (Migration)	5-10
5.16	Select Connection	5-10
5.17	Connection Information.....	5-10
5.18	No Connection Found	5-10
5.19	Connection Rename Error	5-10
5.20	New Folder (Connections).....	5-10
5.21	Continue After Pause	5-11
5.22	Select Library	5-11
5.23	Create Library.....	5-11
5.24	Import Data.....	5-11
5.25	Export/Import Connection Descriptors	5-12
5.26	Create/Edit Database Link.....	5-12
5.27	Create/Edit Index	5-13
5.28	Create Filter.....	5-14
5.29	Create/Edit Materialized View Log.....	5-14
5.30	Create PL/SQL Package	5-15
5.31	Create PL/SQL Subprogram (Function or Procedure)	5-16
5.32	Create/Edit Sequence.....	5-16
5.33	Create SQL File.....	5-17
5.34	Create/Edit Synonym	5-17
5.35	Create Table (quick creation)	5-18
5.36	Create/Edit Table (with advanced options)	5-19

5.37	Storage Options	5-27
5.38	Create Trigger	5-28
5.39	Create Type (User-Defined)	5-30
5.40	Create/Edit User	5-30
5.41	Create/Edit User Defined Report.....	5-31
5.42	Create/Edit User Defined Report Folder	5-32
5.43	Create/Edit View	5-32
5.44	Create XML Schema	5-38
5.45	Configure Extension	5-38
5.46	Configure File Type Associations.....	5-38
5.47	Copy Columns.....	5-39
5.48	Custom Filters	5-39
5.49	Database Copy (Schema Objects)	5-39
5.50	Database Schema Differences.....	5-40
5.51	DDL Panel for Creating or Editing an Object	5-41
5.52	Debugger - Attach to JPDA	5-41
5.53	Deploy or Import Application	5-41
5.54	Describe Object Window	5-42
5.55	Edit/View Value (Table Column or Other Data).....	5-42
5.56	Enter Bind Values	5-42
5.57	Erase from Disk.....	5-42
5.58	Error Writing to Export File	5-42
5.59	Export (Database Objects and Data)	5-43
5.60	Export: Advanced Data Filter	5-45
5.61	Export Error	5-45
5.62	Export Data	5-45
5.63	Export Table Data	5-45
5.64	External Locator Configuration	5-46
5.65	External Tools	5-46
5.66	Create/Edit External Tool	5-47
5.67	Choose Offline Options.....	5-48
5.68	Edit Join	5-48
5.69	Feature Required.....	5-48
5.70	Filter	5-49
5.71	Insert Macro	5-49
5.72	Externally Modified Files.....	5-49
5.73	Filter Object Types	5-49
5.74	Filter Schemas.....	5-50
5.75	Filter Error.....	5-50
5.76	Find/Replace Text	5-50
5.77	Find Result	5-51
5.78	Format Properties	5-51
5.79	Generate Oracle DDL	5-51
5.80	Generate Offline Data Move Files	5-51
5.81	Generate Patch.....	5-51
5.82	Go to Bookmark	5-52
5.83	Go to Line Number	5-52

5.84	Go to Line Number: Error.....	5-52
5.85	Import to CVS.....	5-52
5.86	Load Keyboard Scheme	5-53
5.87	Log In to CVS.....	5-53
5.88	Manage Columns	5-53
5.89	Modify Value	5-53
5.90	Data Move Details.....	5-54
5.91	New Procedure (Refactoring).....	5-54
5.92	No Object Found	5-54
5.93	No Object Selected	5-54
5.94	Object Preferences.....	5-55
5.95	Open File	5-55
5.96	Oracle-Only Report.....	5-55
5.97	Oracle Proxy Authentication.....	5-55
5.98	Paste	5-55
5.99	Privilege Warning for Migration	5-55
5.100	Query Builder	5-56
5.101	Recent Files	5-56
5.102	Create Repository.....	5-56
5.103	Delete or Truncate Repository	5-57
5.104	Capture Microsoft Access Exporter XML.....	5-57
5.105	Rename Local Variable.....	5-57
5.106	Rename Procedure.....	5-57
5.107	Select Current Repository	5-57
5.108	Cannot Capture Table	5-57
5.109	Reset Expired Password (Enter New Password)	5-58
5.110	Revision Lister	5-58
5.111	Run/Debug/Profile PL/SQL.....	5-58
5.112	Create/Edit Breakpoint.....	5-59
5.113	Save/Save As, or Select File	5-59
5.114	Save Files	5-59
5.115	Unable to Save Files.....	5-60
5.116	Save Style Settings	5-60
5.117	Schema Differences Source or Destination Error	5-60
5.118	Script Execution Failed.....	5-60
5.119	Script Generation Complete	5-60
5.120	Set Data Mapping	5-60
5.121	Add/Edit Rule.....	5-61
5.122	Set Encoding	5-61
5.123	Set Pause Continue	5-61
5.124	Sign In (checking for updates)	5-61
5.125	Single Record View.....	5-62
5.126	Save Snippet (User-Defined)	5-62
5.127	Edit Snippets (User-Defined)	5-62
5.128	Subversion: Add Property	5-62
5.129	Subversion: Add to Source Control.....	5-62
5.130	Subversion: Apply Patch	5-63

5.131	Subversion: Branch/Tag	5-63
5.132	Subversion: Check Out from Subversion	5-63
5.133	Subversion: Commit Resources	5-64
5.134	Subversion: Commit Working Copy.....	5-64
5.135	Subversion: Confirm Checkout.....	5-64
5.136	Subversion: Create Remote Directory.....	5-64
5.137	Subversion: Create Subversion Repository	5-65
5.138	Subversion: Create/Edit Subversion Connection	5-65
5.139	Subversion: Edit Configuration File.....	5-65
5.140	Subversion: Export Files	5-66
5.141	Subversion: Export Subversion Connections.....	5-66
5.142	Subversion: History	5-66
5.143	Subversion: Ignore	5-66
5.144	Subversion: Import Subversion Connections	5-66
5.145	Subversion: Import to Subversion	5-66
5.146	Subversion: Lock Resources	5-67
5.147	Subversion: Merge	5-67
5.148	Subversion: Pending Changes	5-68
5.149	Subversion: Properties	5-69
5.150	Subversion: Remove from Subversion.....	5-69
5.151	Subversion: Repository Browser.....	5-69
5.152	Subversion: Revert Local Changes	5-69
5.153	Subversion: Switch.....	5-69
5.154	Subversion: Unlock Resources	5-70
5.155	Subversion: Update Resources.....	5-70
5.156	Subversion: Update Working Copy	5-70
5.157	Subversion: Versioning Properties	5-70
5.158	Third-Party Database Objects	5-71
5.159	Unable to Connect.....	5-71
5.160	Unable to Open File	5-71
5.161	Unit Testing: Action Required	5-71
5.162	Unit Testing: Add Category	5-71
5.163	Unit Testing: Add Data Type	5-71
5.164	Unit Testing: Add Item to Library.....	5-71
5.165	Unit Testing: Add Test Implementation.....	5-71
5.166	Unit Testing: Add Test Suite	5-72
5.167	Unit Testing: Add Tests to Suite	5-72
5.168	Unit Testing: Copy or Rename Unit Test	5-72
5.169	Unit Testing: Create Unit Test.....	5-72
5.170	Unit Testing: Manage Users	5-74
5.171	Unit Testing: Result of Operation.....	5-74
5.172	Unsupported Database Version.....	5-74
5.173	Windows	5-74

List of Figures

1-1	SQL Developer Main Window	1-3
2-1	SQL Developer Migration Architecture	2-6
2-2	V-model with a Database Migration	2-26
2-3	Main Window for a Database Migration.....	2-32
3-1	Unit Test Navigator	3-3

List of Tables

1-1	Default Locations for User-Related Information.....	1-66
2-1	Complex and Simple Scenarios.....	2-7

Preface

This guide provides conceptual and usage information about Oracle SQL Developer, a graphical tool that enables you to browse, create, edit, and delete (drop) database objects; run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export data; and view and create reports.

Audience

This guide is intended for those using the Oracle SQL Developer tool.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

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

Product Accessibility

If you are using a screen reader (such as JAWS), SQL Developer must be running on the same system as the screen reader.

For more information about using a screen reader and Java Access Bridge with SQL Developer, see *Oracle SQL Developer Installation Guide*.

Related Documents

For information about installing Oracle SQL Developer, see the *Oracle SQL Developer Installation Guide*.

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

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

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

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

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

Conventions

The following text conventions are used in this document:

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

Third-Party License Information

Oracle SQL Developer 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.

Apache Regular Expression Package 2.0

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.

Antlr v 2.7.3

<http://www.antlr.org/rights.html>

OracleAS TopLink uses Antlr for EJB QL parsing. Antlr (ANother Tool for Language Recognition), is a language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions containing C++ or Java actions. The ANTLR parser and translator generator is fully in the public domain.

JGoodies Looks and Forms

Copyright © 2003 JGoodies Karsten Lentzsch. 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 JGoodies Karsten Lentzsch 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.

TMate Subversion Software

Copyright © 2004-2005 TMate Software. All rights reserved. This product includes software developed by TMate Software (<http://www.tmatesoft.com/>).

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.

- Redistributions in any form must be accompanied by information on how to obtain complete source code for the software that uses SVNKit and any accompanying software that uses the software that uses SVNKit. The source code must either be included in the distribution or be available for no more than the cost of distribution plus a nominal fee, and must be freely redistributable under reasonable conditions. For an executable file, complete source code means the source code for all modules it contains. It does not include source code for modules or files that typically accompany the major components of the operating system on which the executable file runs.
- Redistribution in any form without redistributing source code for software that uses SVNKit is possible only when such redistribution is explicitly permitted by TMate Software. Please, contact TMate Software at support@svnkit.com to get such permission.

THIS SOFTWARE IS PROVIDED BY TMATE SOFTWARE ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED.

IN NO EVENT SHALL TMATE SOFTWARE 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.

SQL Developer Concepts and Usage

Oracle SQL Developer is a graphical version of SQL*Plus that gives database developers a convenient way to perform basic tasks. You can browse, create, edit, and delete (drop) database objects; run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export data; and view and create reports.

You can connect to any target Oracle database schema using standard Oracle database authentication. Once connected, you can perform operations on objects in the database.

You can connect to schemas for selected third-party (non-Oracle) databases, such as MySQL, Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, and IBM DB2, and view metadata and data in these databases; and you can migrate third-party databases to Oracle.

This chapter contains the following major sections:

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

[Section 1.2, "SQL Developer User Interface"](#)

[Section 1.3, "Database Objects"](#)

[Section 1.4, "Database Connections"](#)

[Section 1.5, "Entering and Modifying Data"](#)

[Section 1.6, "Running and Debugging Functions and Procedures"](#)

[Section 1.7, "Using the SQL Worksheet"](#)

[Section 1.8, "Using Snippets to Insert Code Fragments"](#)

[Section 1.9, "Finding Database Objects"](#)

[Section 1.10, "Using Versioning"](#)

[Section 1.11, "SQL Developer Reports"](#)

[Section 1.12, "SQL Developer Preferences"](#)

[Section 1.13, "Location of User-Related Information"](#)

[Section 1.15, "Oracle TimesTen In-Memory Database Support"](#)

[Section 1.16, "Using the Help"](#)

[Section 1.17, "Tip of the Day"](#)

[Section 1.18, "For More Information"](#)

1.1 Installing and Getting Started with SQL Developer

To install and start SQL Developer, you simply 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 the *Oracle SQL Developer Installation Guide* before you perform the installation. After you have read the installation guide, the basic steps are:

1. Unzip the SQL Developer kit into a directory (folder) of your choice. This directory location will be referred to as `<sqldeveloper_install>`.

Unzipping the SQL Developer kit causes a directory named `sqldeveloper` to be created under the `<sqldeveloper_install>` directory. It also causes many files and folders to be placed in and under that directory.

If Oracle Database (Release 11 or later) is also installed, a version of SQL Developer is also included and is accessible through the menu system under Oracle. This version of SQL Developer is separate from any SQL Developer kit that you download and unzip on your own, so do not confuse the two, and do not unzip a kit over the SQL Developer files that are included with Oracle Database. Suggestion: Create a shortcut for the SQL Developer executable file that you install, and always use it to start SQL Developer.

2. To start SQL Developer, go to the `sqldeveloper` directory under the `<sqldeveloper_install>` directory, and do one of the following:

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

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

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

3. If you want to become familiar with SQL Developer concepts before using the interface, read the rest of this chapter before proceeding to the next step.
4. Create at least one database connection (or import some previously exported connections), so that you can view and work with database objects, use the SQL Worksheet, and use other features.

To create a new database connection, right-click the **Connections** node in the Connections navigator, select **New Connection**, and complete the required entries in the [Create/Edit/Select Database Connection](#) dialog box.

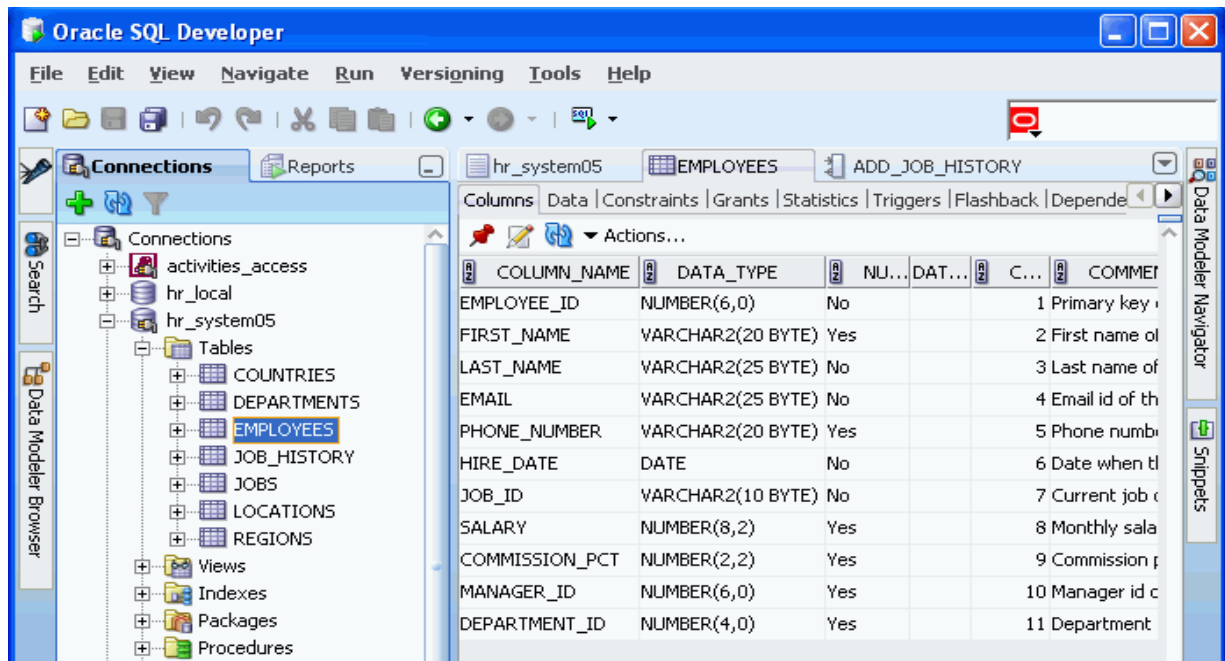
5. If you want to get started quickly with SQL Developer, do the short tutorial in [Chapter 4, "Tutorial: Creating Objects for a Small Database"](#), or work with your existing database objects.

1.2 SQL Developer User Interface

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

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

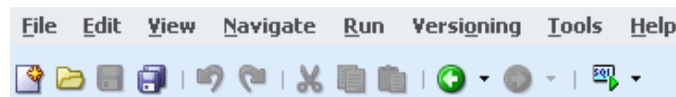
Figure 1–1 SQL Developer Main Window



Note: This text explains the default interface. However, you can customize many aspects of the appearance and behavior of SQL Developer by setting preferences (see [Section 1.12](#)). If you ever need to restore the default interface, see [Section 1.2.2, "Restoring the Original "Look and Feel"](#)".

Note: For migration of third-party databases to Oracle, see also [Section 2.14, "SQL Developer User Interface for Migration"](#).

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



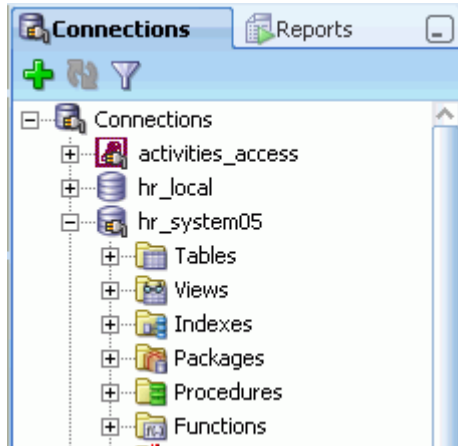
You can use **shortcut keys** to access menus and menu items: for example Alt+F for the File menu and Alt+E for the Edit menu; or Alt+H, then Alt+S for Help, then Search. You can also display the File menu by pressing the F10 key (except in the SQL Worksheet, where F10 is the shortcut for Explain Plan).

Icons under the menus perform various actions, including the following:

- **New** creates a new a new database object (see [Section 5.11, "Create New Object"](#)).
- **Open** opens a file (see [Section 5.95, "Open File"](#)).
- **Save** saves any changes to the currently selected object.
- **Save All** saves any changes to all open objects.

- **Back** moves to the pane that you most recently visited. (Or use the drop-down arrow to specify a tab view.)
- **Forward** moves to the pane after the current one in the list of visited panes. (Or use the drop-down arrow to specify a tab view.)
- **Open SQL Worksheet** opens the SQL Worksheet (see [Using the SQL Worksheet](#)). If you do not use the drop-down arrow to specify the database connection to use, you are asked to select a connection.

The left side of the SQL Developer window has tabs and panes for the Connections and Reports navigators, icons for performing actions, and a hierarchical tree display for the currently selected navigator, as shown in the following figure.



The **Connections navigator** lists database connections that have been created. To create a new database connection, import an XML file with connection definitions, or export or edit current connections, right-click the Connections node and select the appropriate menu item. (For more information, see [Section 1.4, "Database Connections"](#).)

The **Files navigator** (marked by a folder icon; not shown in the preceding figure) displays your local file system using a standard hierarchy of folders and files. You can double-click or drag and drop files to open them, and you can edit and save the files. For example, if you open a .sql file, it is displayed in a SQL Worksheet window. The Files navigator is especially useful if you are using versioning with SQL Developer (see [Section 1.10, "Using Versioning"](#)).

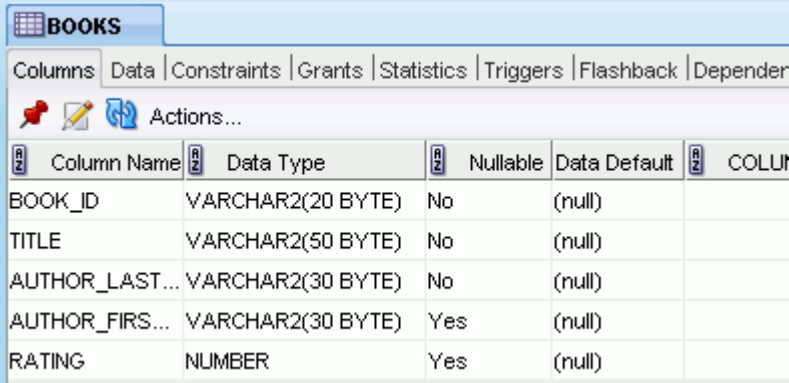
The **Reports navigator** lists informative reports provided by SQL Developer, such as a list of tables without primary keys for each database connection, as well as any user-defined reports. (For more information, see [Section 1.11, "SQL Developer Reports"](#).)

Icons under the Connections tab (above the metadata tree) perform the following actions on the currently selected object:

- **Refresh** queries the database for the current details about the selected object (for example, a connection or just a table).
- **Apply Filter** restricts the display of objects using a filter that you specify. For example, you can right-click the Tables node and specify a filter of EM% to see only tables that start with EM and to have the Tables node label be changed to *Tables (EM%)*. To remove the effects of applying a filter, right-click the node and select **Clear Filter**.

The metadata tree in the Connections pane displays all the objects (categorized by object type) accessible to the defined connections. To select an object, expand the appropriate tree node or nodes, then click the object.

The right side of the SQL Developer window has tabs and panes for objects that you select or open, as shown in the following figure, which displays information about a table named BOOKS. (If you hold the mouse pointer over the tab label -- BOOKS in this figure -- a tooltip displays the object's owner and the database connection.)



Column Name	Data Type	Nullable	Data Default	Column ID
BOOK_ID	VARCHAR2(20 BYTE)	No	(null)	
TITLE	VARCHAR2(50 BYTE)	No	(null)	
AUTHOR_LAST...	VARCHAR2(30 BYTE)	No	(null)	
AUTHOR_FIRS...	VARCHAR2(30 BYTE)	Yes	(null)	
RATING	NUMBER	Yes	(null)	

For objects other than subprograms, icons provide the following options:

- **Freeze View** (the **pin**) keeps that object's tab and information in the window when you click another object in the Connections navigator; a separate tab and display are created for that other object. If you click the pin again, the object's display is available for reuse.
- **Edit** displays a dialog box for editing the object.
- **Refresh** updates the display by querying the database for the latest information.
- **Actions** displays a menu with actions appropriate for the object. The actions are the same as when you right-click an object of that type in the Connections navigator, except the Actions menu does not include Edit.

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.

For tables and views, this information is grouped under tabs, which are labeled near the top. For example, for tables the tabs are Columns, Data (for seeing and modifying the data itself), Indexes, Constraints, and so on; and you can click a column heading under a tab to sort the grid rows by the values in that column. For most objects, the tabs include SQL, which displays the SQL statement for creating the object.

You can export data from a detail pane or from the results of a SQL Worksheet operation or a report by using the context menu and selecting **Export**.

The **Messages - Log** area is used for feedback information as appropriate (for example, results of an action, or error or warning messages). If this area is not already visible, you can display it by clicking View and then Log.

The **Compiler - Log** area is used for any messages displayed as a result of a Compile or Compile for Debug operation.

1.2.1 Menus for SQL Developer

This topic explains menu items that are specific to SQL Developer.

Edit menu

Extended Paste: Displays the [Paste](#) dialog box, in which you select a clipboard item (from potentially many) to be pasted into the current location.

Duplicate Selection: When you have selected text while editing a function or procedure, creates a copy of the selected text at the current location.

Wrap Selection: When you have selected text while editing a function or procedure, wraps the selected text.

View menu

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

Breakpoints: Displays the Breakpoints pane, which shows breakpoints, both system-defined and user-defined (see [Section 1.6, "Running and Debugging Functions and Procedures"](#)).

Debugger: Displays panes related to debugging (see [Section 1.6, "Running and Debugging Functions and Procedures"](#)).

Log: Displays the Messages - Log pane, which can contain errors, warnings, and informational messages.

Run Manager: Displays the Run Manager pane, which contains entries for any active debugging sessions.

Team: Lets you display the Versioning navigator (see [Section 1.10, "Using Versioning"](#)).

Connections: Displays the Connections navigator.

Data Modeler: Lets you display the Browser and Thumbnail Diagram panes of the [Data Modeler Viewer \(Read-Only\)](#).

Files: Displays the Files navigator, which is marked by a folder icon. You can use the Files navigator to browse, open, edit, and save files that are accessible from the local system.

Find DB Object: Displays the Find Database Object pane (see [Section 1.9, "Finding Database Objects"](#)).

History: Displays information about SQL statements that you have executed. You can select statements and append them to or overwrite statements on the worksheet (see [Section 1.7.7, "SQL History"](#)).

Migrations: Lets you display the Captured Models and Converted Models navigators (see [Section 2.14, "SQL Developer User Interface for Migration"](#)).

OWA Output: Displays Oracle Web Agent (MOD_PLSQL) output (see [Section 1.7.6, "OWA Output Pane"](#)).

Recent Objects: Displays a pane with names of recently opened objects. You can click a name in the list to go to its editing window.

Reports: Displays the Reports navigator (see [Section 1.11, "SQL Developer Reports"](#)).

Snippets: Displays snippets (see [Section 1.8, "Using Snippets to Insert Code Fragments"](#)).

Task Progress: Displays the Task Progress pane.

Show Status Bar: Controls the display of the status bar at the bottom of the SQL Developer window.

Show Toolbars: Controls the display of the toolbars: Main toolbar (under the SQL Developer menus), the Connections Navigator toolbar, and (if a package or subprogram is open) the Code Editor toolbar.

Navigate menu

Contains options for navigating to panes and in the execution of subprograms.

Back: Moves to the pane that you most recently visited.

Forward: Moves to the pane after the current one in the list of visited panes.

Toggle Bookmark: If you are editing a function or procedure, creates or removes a bookmark (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Remove Bookmarks from File: Removes bookmarks from the currently active editing window for a function or procedure (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Remove All Bookmarks: Removes bookmarks from open editing windows for functions and procedures (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Go to Bookmark: Displays a dialog box so that you can go to a specified bookmark (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Go to Next Bookmark: Goes to the next bookmark in the currently active editing window for a function or procedure (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Go to Previous Bookmark: Goes to the previous bookmark in the currently active editing window for a function or procedure (see [Section 1.6.1, "Using Bookmarks When Editing Functions and Procedures"](#)).

Go to Line: Goes to the specified line number and highlights the line in the editing window for the selected function or procedure. (To display line numbers, enable **Show Line Numbers** under the [Code Editor: Line Gutter](#) preferences.)

Go to Last Edit: Goes to the last line that was edited in the editing window for a function or procedure.

Go to Recent Files: Displays the [Recent Files](#) dialog box, in which you can specify a function or procedure to go to.

Run menu

Contains options relevant when a function or procedure is selected or when it is open for debugging.

Run [name]: Starts execution of the specified function or procedure.

Debug [name]: Starts execution of the specified function or procedure in debug mode.

The remaining items on the Debug menu match commands on the debugging toolbar, which is described in [Section 1.6, "Running and Debugging Functions and Procedures"](#).

Versioning menu

Contains options related to support for the Subversion version management and source control system, and for any other such systems (such as CVS) that you have

added as extensions to SQL Developer through the "check for updates" feature. See [Section 1.10, "Using Versioning"](#) for more information.

The commands on the Versioning menu depend on which version management and source control systems are available for use with SQL Developer.

Tools menu

Invokes SQL Developer tools.

Data Modeler: Starts the [Data Modeler Viewer \(Read-Only\)](#) if it not already active; otherwise, contains the commands **About Data Modeler**, **Design Rules**, and **General Options** (user preferences).

Migration: Displays the [Migration Menu](#), which contains commands related to migrating third-party databases to Oracle.

Unit Test: Displays the [Unit Test Submenu](#), which contains commands related to unit testing.

Configuration Palette: Displays the [Configure Component Palette](#) dialog box.

Database Copy: Enables you to copy objects from one database schema to another (see the [Database Copy \(Schema Objects\)](#) interface).

Database Diff: Enables you to compare two schemas to find differences between objects of the same type and name (for example, tables named CUSTOMERS) in two different schemas, and optionally to update the objects in the destination schema to reflect differences in the source schema (see the [Database Schema Differences](#) interface).

Database Export: Enables you to export some or all objects of one or more object types for a database connection to a file containing SQL statements to create these objects and optionally to export table data (see the [Export \(Database Objects and Data\)](#) interface).

Monitor SQL: Displays information about any query currently executing and queries that are done executing for a selected connection. To see detailed information about a query, right-click its row and select Show SQL Details. The information is especially useful for real-time monitoring of long-running SQL statements. Cursor statistics (such as CPU times and IO times) and execution plan statistics (such as number of output rows, memory, and temporary space used) are updated close to real-time during statement execution. (Internally, this feature calls the DBMS_SQLTUNE.REPORT_SQL_MONITOR subprogram.)

Monitor Sessions: Displays the status of one or more sessions, using information from the V\$RSRC_SESSION_INFO view, which shows how the session has been affected by the Oracle Database Resource Manager. For more information about session monitoring, see *Oracle Database Administrator's Guide*.

SQL Worksheet: Displays a worksheet in which you can enter and execute SQL and PL/SQL statements using a specified connection (see [Section 1.7, "Using the SQL Worksheet"](#)).

External Tools: Displays the [External Tools](#) dialog box, with information about user-defined external tools that are integrated with the SQL Developer interface. From this dialog box can add external tools (see [Section 5.66, "Create/Edit External Tool"](#)). The Tools menu also contains items for any user-defined external tools.

Preferences: Enables you to customize the behavior of SQL Developer (see [Section 1.12, "SQL Developer Preferences"](#)).

Help menu

Displays help about SQL Developer and enables you to check for SQL Developer updates.

Search: Displays the Help Center window.

Table of Contents: Displays the Help Center window. In this window you can click these icons:

- **Keep on Top:** Toggles whether to keep the Help Center window on top of the Data Modeler window.
- **Navigators:** Lets you select a help 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 Favorites list.
- **Find:** Lets you search for a string in the current help topic.

Start Page: Displays a page with links for options for learning about SQL Developer.

Tip of the Day (English locales only): Displays a suggestion for efficient use of SQL Developer. (See [Section 1.17](#), "Tip of the Day".)

Check for Updates: Checks for any updates to the selected optional SQL Developer extensions, as well as any mandatory SQL Developer extensions. The available updates may include the JTDS JDBC Driver for Microsoft SQL Server and the MySQL JDBC Driver, which enable you to create connections to third-party databases. (If the system you are using is behind a firewall, see the SQL Developer user preferences for [Web Browser and Proxy](#).)

About: Displays version-related information about SQL Developer and its components.

1.2.2 Restoring the Original "Look and Feel"

If you have made changes to the SQL Developer user interface ("look and feel"), such as accidentally repositioning navigators and panes, you can restore the interface to the way it was after SQL Developer was installed by following these steps:

1. If you are running SQL Developer, exit.
2. Create a backup copy of the folder or directory where your SQL Developer user information is stored, in case you want to restore any old user-defined reports, snippets, code templates, or SQL history. The default location is:
 - Windows: C:\Documents and Settings*<user-name>*\Application Data\SQL Developer
 - Linux or Mac OS X: ~/.sqldeveloper

If you have specified a nondefault location for your SQL Developer user information (see [Section 1.13](#)), create the backup copy of that folder or directory instead.

(If you do not want to use any old information or settings, you can skip creating a backup copy.)

3. Delete the original (not the backup) folder or directory where your user information is stored (explained in step 2).

4. Start SQL Developer.

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

If you have made changes to the SQL Developer 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**.

1.3 Database Objects

You can create, edit, and delete (drop) most types of objects in an Oracle database by using the context menu (right-click, or Shift+F10) in the Connections navigator or by clicking the **Actions** button in the detail pane display. For some objects, you can do other operations, as appropriate for the object type.

Note: The actions available from context menus and Actions buttons depend on the Oracle Database release number for the specified database connection. If an action mentioned in the text is not available with a connection, it may be that the feature was not available in that release of Oracle Database.

You can search for specific objects associated with an Oracle database connection by clicking the Search icon, as explained in [Section 1.9, "Finding Database Objects"](#).

If you have connected to any third-party (non-Oracle) databases, such as MySQL, Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, or IBM DB2, you can view their objects using the Connections navigator. (For information about connecting to third-party databases, see the SQL Developer user preferences for [Database: Third Party JDBC Drivers](#).)

1.3.1 Applications (Application Express 3.0.1 and Later)

Effective with Oracle Application Express 3.0.1, if you use SQL Developer to connect to a schema that owns any Application Express applications, the Connections navigator has an **Application Express** node. You can click an application name to display tabs (Application, Pages, LOVs, Lists, Templates, Breadcrumbs, and so on) with information about the application.

You can perform the following operations on an Application Express application by right-clicking the application name in the Connections navigator and selecting an item from the menu:

- **Import Application:** Imports an application from a specified file and installs the application.
- **Deploy Application:** Deploys an application into a specified target schema.
- **Drop:** Deletes the application.
- **Modify Application:** Enables you to change the alias, name (Rename), status, global notification, and proxy server for the application.
- **Export DDL:** Saves the DDL statements to create the application (or the selected component) to a file, a .zip file, a worksheet, or the system clipboard.
- **Refactor (in bulk):** Collects all anonymous blocks, refactors them into PL/SQL procedures, and places them in a package. The output of a refactor in bulk

operation is a PL/SQL script, which you can review and save, and which you can execute to create the package.

The following operations are available only by right-clicking the Application Express node in the Connections navigator and selecting an item from the menu:

- **Start EPG:** Starts the embedded PL/SQL gateway for Application Express. Displays a dialog box for executing the following statements: `BEGIN DBMS_EPG.map_dad('APEX', '/apex/*'); end;`
- **Stop EPG:** Stops the embedded PL/SQL gateway for Application Express. Displays a dialog box for executing the following statements: `BEGIN DBMS_EPG.unmap_dad('APEX'); end;`

1.3.2 Cache Groups (Oracle TimesTen In-Memory Database)

A cache group describes a collection of in-memory database tables that map to all or a subset of the tables in an Oracle database. A cache group can consist of all or a subset of the rows and columns in these tables. Multiple cache groups can be used to cache different sets of related tables in the Oracle database.

1.3.3 Database Links (Public and Private)

A database link is a database object in one database that enables you to access objects on another database. The other database need not be an Oracle Database system; however, to access non-Oracle systems you must use Oracle Heterogeneous Services. After you have created a database link, you can use it to refer to tables and views in the other database. The Connections navigator has a **Database Links** node for all database links (public and private) owned by the user associated with the specified connection, and a **Public Database Links** node for all public database links on the database associated with the connection. For help with specific options in creating a database link, see [Section 5.26, "Create/Edit Database Link"](#).

You can perform the following operations on a database link by right-clicking the database link name in the Connections navigator and selecting an item from the menu:

- **Test Database Link:** Validates the database link.
- **Drop:** Deletes the database link.

1.3.4 Directories

A directory object specifies 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. To create a directory (that is, a directory object), you can use SQL Developer or the SQL statement `CREATE DIRECTORY`.

You can use directory names when referring to BFILEs in your PL/SQL code and OCI calls, rather than hard coding the operating system path name, for management flexibility. All directories are created in a single namespace and are not owned by an individual schema. You can secure access to the BFILEs stored within the directory structure by granting object privileges on the directories to specific users.

1.3.5 Editions

Edition-based redefinition (introduced in Oracle Database Release 11.2) enables you to upgrade the database component of an application while it is in use, thereby minimizing or eliminating down time. To upgrade an application while it is in use, you copy the database objects that comprise the application and redefine the copied

objects in isolation. Your changes do not affect users of the application—they continue to run the unchanged application. When you are sure that your changes are correct, you make the upgraded application available to all users. For more information, see the chapter about edition-based redefinition in *Oracle Database Advanced Application Developer's Guide*.

To specify the current edition, right-click the edition name and select **Set Current Edition**. To create an edition under an existing edition, right-click the edition name and select **Create Edition**. To delete an edition (and optionally all editions under it), right-click the edition name and select **Drop Edition**.

1.3.6 Functions

A function is a type of PL/SQL subprogram, which is a programming object that can be stored and executed in the database server, and called from other programming objects or applications. (Functions return a value; procedures do not return a value.) For help with specific options in creating a PL/SQL subprogram, see [Section 5.31, "Create PL/SQL Subprogram \(Function or Procedure\)"](#).

You can perform the following operations on a function by right-clicking the function name in the Connections navigator and selecting an item from the menu:

- **Edit**: Displays the function text so that you can view and edit it.
- **Run**: Displays the [Run/Debug/Profile PL/SQL](#) dialog box, and then executes the function in normal (not debug) mode.
- **Compile**: Performs a PL/SQL compilation of the function.
- **Compile for Debug**: Performs a PL/SQL compilation of the procedure, with PL/SQL library units compiled for debugging.
- **Profile** (for an Oracle Database Release 11.1 or later connection): Displays the [Run/Debug/Profile PL/SQL](#) dialog box, and then executes the function and collects execution statistics.
- **Drop**: Deletes the function.
- **Grant**: Enables you to grant available privileges on the function to selected users.
- **Revoke**: Enables you to revoke available privileges on the function from selected users.
- **Format**: Reformats the text of the function definition.
- **Create Unit Test**: Creates a unit test (see [Chapter 3](#)) for the function.
- **Export DDL**: Enables you to export the DDL statement for creating the function to a file, a SQL Worksheet, or the clipboard.

1.3.7 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. For help with specific options in creating an index, see [Section 5.27, "Create/Edit Index"](#).

You can perform the following operations on an index by right-clicking the index name in the Connections navigator and selecting an item from the menu:

- **Drop**: Deletes the index.

- **Rebuild Index:** Re-creates the index or one of its partitions or subpartitions. If the index is unusable, a successful rebuild operation makes the index usable. For a function-based index, rebuilding also enables the index; however, if the function on which the index is based does not exist, the rebuild operation fails.
- **Rename Index:** Changes the name of the index.
- **Make Unusable:** Prevents the index from being used by Oracle in executing queries. An unusable index must be rebuilt, or dropped and re-created, before it can be used again.
- **Coalesce:** Merges the contents of index blocks, where possible, to free blocks for reuse.
- **Compute Statistics:** For a function-based index, collects statistics on both the index and its base table using the DBMS_STATS package. Such statistics will enable Oracle Database to correctly decide when to use the index.
- **Export DDL:** Saves the DDL statement to create the index to a file, a SQL Worksheet, or the system clipboard.

1.3.8 Java Sources

Java sources can be created and managed in the database. You can create a Java source object by right-clicking the Java node in the Connections navigator, selecting **Load Java**, and specifying the Java source name and either entering the source code or loading a Java source, class, or resource from a file (BFILE). (A CREATE OR REPLACE AND RESOLVE JAVA SOURCE statement is executed using the information you specify.) For information about Java concepts and stored procedures, see *Oracle Database Java Developer's Guide*.

1.3.9 Jobs

A job object (job) is a collection of metadata that describes a user-defined task. It defines what needs to be executed (the action), when (the one-time or recurring schedule or a triggering event), where (the destinations), and with what credentials. A job has an owner, which is the schema in which it is created. For information about jobs, see the chapter about Oracle Scheduler concepts in *Oracle Database Administrator's Guide*.

1.3.10 Materialized Views

A materialized view is a database object that contains the results of a query. The FROM clause of the query can name tables, views, and other materialized views. Collectively these objects are called master tables (a replication term) or detail tables (a data warehousing term). This reference uses "master tables" for consistency. The databases containing the master tables are called the master databases. For help with specific options in creating a materialized view, see [Section 5.43, "Create/Edit View"](#), especially the [View Information or Materialized View Properties](#) pane.

1.3.11 Materialized View Logs

A materialized view log is a table associated with the master table of a materialized view. When DML changes are made to master table data, Oracle Database stores rows describing those changes in the materialized view log and then uses the materialized view log to refresh materialized views based on the master table. This process is called incremental or fast refresh. Without a materialized view log, Oracle Database must

reexecute the materialized view query to refresh the materialized view. This process is called a complete refresh. Usually, a fast refresh takes less time than a complete refresh.

1.3.12 Packages

A package is an object that contains subprograms, which are programming objects that can be stored and executed in the database server, and called from other programming objects or applications. A package can contain functions or procedures, or both. For help with specific options in creating a package, see [Section 5.30, "Create PL/SQL Package"](#).

You can perform the following operations on a package by right-clicking the package name in the Connections navigator and selecting an item from the menu:

- **Edit:** Opens the package in a window, where you can modify the content and other information.
- **Run:** Lets you select a member in the package and run it.
- **Compile:** Performs a PL/SQL compilation of the members in the package.
- **Compile for Debug:** Performs a PL/SQL compilation of the members in the package, with PL/SQL library units compiled for debugging
- **Order Members By:** Orders the members of the package by location in the source, by name, or by type and by name within each type.
- **Use as Template:** Lets you create a new package using the selected package as the initial content.
- **Drop Package:** Deletes the package.
- **Create Body:** Displays a pane in which you can enter text for the package body.
- **Grant:** Lets you grant privileges on the package
- **Revoke:** Lets you revoke privileges on the package.
- **Save Package Spec and Body:** Saves the package specification and body to a file that you specify.
- **Export DDL:** Saves the DDL statement to create the package to a file, a SQL Worksheet, or the system clipboard.

1.3.13 Procedures

A procedure is a type of PL/SQL subprogram, which is a programming object that can be stored and executed in the database server, and called from other programming objects or applications. (Procedures do not return a value; functions return a value.) For help with specific options in creating a PL/SQL subprogram, see [Section 5.31, "Create PL/SQL Subprogram \(Function or Procedure\)"](#).

You can perform the following operations on a procedure by right-clicking the procedure name in the Connections navigator and selecting an item from the menu:

- **Edit:** Displays the procedure text so that you can view and edit it.
- **Run:** Displays the [Run/Debug/Profile PL/SQL](#) dialog box, and then executes the procedure in normal (not debug) mode.
- **Compile:** Performs a PL/SQL compilation of the procedure.
- **Compile for Debug:** Performs a PL/SQL compilation of the procedure, with PL/SQL library units compiled for debugging.

- **Profile** (for an Oracle Database Release 11.1 or later connection): Displays the [Run/Debug/Profile PL/SQL](#) dialog box, and then executes the procedure and collects execution statistics.
- **Drop**: Deletes the procedure.
- **Grant**: Enables you to grant available privileges on the procedure to selected users.
- **Revoke**: Enables you to revoke available privileges on the procedure from selected users.
- **Format**: Reformats the text of the procedure definition.
- **Create Unit Test**: Creates a unit test (see [Chapter 3](#)) for the procedure.
- **Export DDL**: Enables you to export the DDL statement for creating the procedure to a file, a SQL Worksheet, or the clipboard.

1.3.14 Queues

A queue is an object in which messages are enqueued and dequeued. Queues are managed by Oracle Streams Advanced Queuing (AQ). For information about using queues, see *Oracle Streams Advanced Queuing User's Guide*.

1.3.15 Queue Tables

A queue table is a table that holds messages to be used with Oracle Streams Advanced Queuing (AQ). For information about using queue tables, see *Oracle Streams Advanced Queuing User's Guide*, especially the information about managing queue tables in the chapter describing the Oracle Streams AQ administrative interface.

1.3.16 Recycle Bin

The Recycle bin (applicable only to Oracle Database Release 10g and later) holds objects that have been dropped (deleted). The objects are not actually deleted until a commit operation is performed. Before the objects are actually deleted, you can "undelete" them by selecting them in the Recycle bin and selecting **Undrop** from the context menu.

You can perform the following operations on an object in the Recycle bin by right-clicking the object name in the Recycle bin in the Connections navigator and selecting an item from the menu:

- **Purge**: Removes the object from the Recycle bin and deletes it.
- **Flashback to Before Drop**: Moves the object from the Recycle bin back to its appropriate place in the Connections navigator display.

1.3.17 Replication Schemes (Oracle TimesTen In-Memory Database)

A replication scheme is a configuration, using SQL statements and a transaction-based log, whereby committed changes are copied from their source to one or more subscriber databases. The goal is to enable high efficiency and low overhead during the replication.

1.3.18 Sequences

Sequences are used to generate unique integers. You can use sequences to automatically generate primary key values. For help with specific options in creating and editing a sequence, see [Section 5.32, "Create/Edit Sequence"](#).

1.3.19 Synonyms (Public and Private)

Synonyms provide alternative names for tables, views, sequences, procedures, stored functions, packages, materialized views, Java class database objects, user-defined object types, or other synonyms. The Connections navigator has a **Synonyms** node for all synonyms (public and private) owned by the user associated with the specified connection, and a **Public Synonyms** node for all public synonyms on the database associated with the connection. For help with specific options in creating and editing a synonym, see [Section 5.34, "Create/Edit Synonym"](#).

1.3.20 Tables

Tables are 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. To create a table, you can do one of the following:

- Create the table quickly by adding columns and specifying frequently used features. To do this, *do not check* the Advanced box in the Create Table dialog box. For help with options for creating a table using this quick approach, see [Create Table \(quick creation\)](#).
- Create the table by adding columns and selecting from a larger set of features. To do this, *check* the Advanced box in the Create Table dialog box. For help with options for creating a table with advanced features, see [Create/Edit Table \(with advanced options\)](#).
- Create the table automatically from a Microsoft Excel worksheet. To do this, right-click Tables under a connection in the Connections navigator, and select **Import Data**. When asked for the file, select a file of type .xls or .csv.

You can perform the following operations on a table by right-clicking the table name in the Connections navigator and selecting an item from the menu:

- **Edit:** Displays the [Create/Edit Table \(with advanced options\)](#) dialog box.
- **Table:** Table actions include Rename, Copy (create a copy using a different name), Drop (delete the table), Truncate (delete existing data without affecting the table definition), Lock (set the table lock mode: row share, exclusive, and so on), Comment (descriptive comment explaining the use or purpose of the table), Parallel (change the default degree of parallelism for queries and DML on the table), No Parallel (specify serial execution), and Count Rows (return the number of rows).
- **Column:** Column actions include Comment (descriptive comment about a column), Add, Drop, and Normalize.
- **Constraint:** Includes options for adding, dropping, enabling, and disabling constraints.
- **Index:** Options include Create (create an index on specified columns), Create Text (create an Oracle Text index on a column), Create Text (create a function-based index on a column), and Drop.

- **Constraint:** Options include Enable or Disable Single, Drop (delete a constraint), Add Check (add a check constraint), Add Foreign Key, and Add Unique.
- **Privileges:** If you are connected as a database user with sufficient privileges, you can Grant or Revoke privileges on the table to other users.
- **Statistics:** Options include Gather Statistics (compute exact table and column statistics and store them in the data dictionary) and Validate Structure (verifies the integrity of each data block and row, and for an index-organized table also generates the optimal prefix compression count for the primary key index on the table). Statistics are used by the Oracle Database optimizer to choose the execution plan for SQL statements that access analyzed objects.
- **Storage:** Options include Shrink Table (shrink space in a table, for segments in tablespaces with automatic segment management) and Move Table (to another tablespace). The Shrink Table options include Compact (only defragments the segment space and compacts the table rows for subsequent release, but does not readjust the high water mark and does not release the space immediately) and Cascade (performs the same operations on all dependent objects of the table, including secondary indexes on index-organized tables).
- **Trigger:** Options include Create, Create PK from Sequence (create a before-insert trigger to populate the primary key using values from a specified sequence), Enable or Disable All, Enable or Disable Single, and Drop (delete the trigger).
- **Import Data:** Enables you to import data from a Microsoft Excel worksheet (.xls or .csv file).
- **Export Data:** Enables you to export some or all of the table data to a file or to the system clipboard, in any of the following formats: XML (XML tags and data), CSV (comma-separated values including a header row for column identifiers), SQL Insert (INSERT statements), or SQL Loader (SQL*Loader control file). After you select a format, the [Export Table Data](#) dialog box is displayed.

You can perform the following operations on a column in a table by right-clicking the column name in the Connections navigator and selecting an item from the menu:

- **Rename:** Renames the column.
- **Drop:** Deletes the column (including all data in that column) from the table.
- **Comment:** Adds a descriptive comment about the column.
- **Encrypt** (for Oracle Database Release 10.2 and higher, and only if the Transparent Data Encryption feature is enabled for the database): Displays a dialog box in which you specify a supported encryption algorithm to be used for encrypting all data in the column. Current data and subsequently inserted data are encrypted.
- **Decrypt** (for Oracle Database Release 10.2 and higher, and only if the Transparent Data Encryption feature is enabled for the database): Decrypts data in the column that had been encrypted, and causes data that is subsequently inserted not to be encrypted.
- **Normalize:** Creates a new table using the distinct values in the specified column. You must specify names for the new table and its primary key column, as well as a sequence name and trigger name.

1.3.20.1 Flashback Table Support

For tables in Oracle Database Release 11.1 and later, the table display includes the **Flashback** tab, which provides a view of the modified and original data in the table. If you have appropriate privileges, you can click the **Undo SQL** subtab to select and

review the syntax required to undo changes. For information about using the Flashback Table feature, see *Oracle Database Backup and Recovery User's Guide*.

1.3.21 Triggers

Triggers are stored PL/SQL blocks associated with a table, a schema, or the database, or anonymous PL/SQL blocks or calls to a procedure implemented in PL/SQL or Java. Oracle Database automatically executes a trigger when specified conditions occur. For help with specific options in creating a trigger, see [Section 5.38, "Create Trigger"](#).

1.3.22 Types

A data type associates a fixed set of properties with the values that can be used in a column of a table or in an argument of a function or procedure. 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.

For help with specific options in creating a user-defined type, see [Section 5.39, "Create Type \(User-Defined\)"](#).

1.3.23 Users (Other Users)

Database users are accounts through which you can log in to the database. In the Connections navigator, you can see the **Other Users** in the database associated with a connection, but the database objects that you are allowed to see for each user are determined by the privileges of the database user associated with the current database connection.

If you are connected as a user with the DBA role, you can create a database user by right-clicking Other Users and selecting **Create User**, and you can edit an existing database user by right-clicking the user under Other Users and selecting **Edit User**. For help on options in creating and editing users, see [Create/Edit User](#).

1.3.24 Views

Views are virtual tables (analogous to queries in some database products) that select data from one or more underlying tables. Oracle Database provides many view creation options and specialized types of views (such as materialized views, described in [Section 1.3.10, "Materialized Views"](#)), to meet a variety of enterprise needs. For help with specific options in creating and editing a view, see [Create/Edit View](#).

You can perform the following operations on a view by right-clicking the view name in the Connections navigator and selecting an item from the menu:

- **Edit**: Displays the [Create/Edit View](#) dialog box.
- **Drop**: Deletes the view.
- **Compile**: Recompiles the view, to enable you to locate possible errors before run time. You may want to recompile a view after altering one of its base tables to ensure that the change does not affect the view or other objects that depend on it.

1.3.25 XML DB Repository

Oracle XML DB Repository is a component of Oracle Database that is optimized for handling XML data. The Oracle XML DB repository contains resources, which can be either folders (directories, containers) or files. For more information about Oracle XML

DB Repository, see *Oracle XML DB Developer's Guide* in the Oracle Database documentation library.

To create a subfolder of an existing folder, right-click the folder name and select **Create Subfolder**. To delete a folder (and optionally all subfolders under it), right-click the folder name and select **Drop Folder**.

1.3.26 XML Schemas

XML schemas are schema definitions, written in XML, that describe the structure and various other semantics of conforming instance XML documents. For conceptual and usage information about XML schemas, see *Oracle XML DB Developer's Guide* in the Oracle Database documentation library.

You can edit an XML schema by right-clicking the XML schema name in the Connections navigator and selecting **Edit** from the menu.

1.3.27 Captured and Converted Database Objects (for Migration)

If you are migrating a third-party database to Oracle, the Captured Models and Converted Models navigators can display models that include database objects, such as tables and procedures. A captured object represents an object in the captured third-party database, and a converted object represents an Oracle model of that object as it will be created in the Oracle database.

The context menu for each captured object includes **Convert to Oracle**, which creates a corresponding converted object. The context menu for each converted object includes **Generate**, which creates the corresponding Oracle Database object. (The context menus will contain other items as appropriate for the object.)

For information about the related Oracle Database objects, see the following:

- [Section 1.3.6, "Functions"](#)
- [Section 1.3.7, "Indexes"](#)
- [Section 1.3.13, "Procedures"](#)
- [Section 1.3.18, "Sequences"](#)
- [Section 1.3.20, "Tables"](#)
- [Section 1.3.21, "Triggers"](#)
- [Section 1.3.23, "Users \(Other Users\)"](#)
- [Section 1.3.24, "Views"](#)

1.4 Database Connections

A **connection** is a SQL Developer object that specifies the necessary information for connecting to a specific database as a specific user of that database. You must have at least one database connection (existing, created, or imported) to use SQL Developer.

You can connect to any target Oracle database schema using standard Oracle database authentication. Once connected, you can perform operations on objects in the database. You can also connect to schemas for selected third-party (non-Oracle) databases, such as MySQL, Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, and IBM DB2, and view metadata and data.

When you start SQL Developer and whenever you display the database connections dialog box, SQL Developer automatically reads any connections defined in the

tnsnames.ora file on your system, if that file exists. You can specify the tnsnames.ora location in the [Database: Advanced](#) preferences. By default, tnsnames.ora is located in the \$ORACLE_HOME/network/admin directory, but it can also be in the directory specified by the TNS_ADMIN environment variable or registry value or (on Linux systems) the global configuration directory. On Windows systems, if the tnsnames.ora file exists but its connections are not being used by SQL Developer, define TNS_ADMIN as a system environment variable. For information about the tnsnames.ora file, see the "Local Naming Parameters (tnsnames.ora)" chapter in *Oracle Database Net Services Reference*.

You can create additional connections (for example, to connect to the same database but as different users, or to connect to different databases). Each database connection is listed in the Connections navigator hierarchy.

To create a new database connection, right-click the Connections node and select **New Database Connection**. Use the dialog box to specify information about the connection (see [Section 5.13, "Create/Edit/Select Database Connection"](#)). You can also create a new database connection by selecting an existing connection in that dialog box, changing the connection name, changing other connection attributes as needed, and clicking Save or Connect.

To create database connection for each unlocked user account in the Oracle database instance on the local system, right-click the Connections node and select **Create Local Connections**. The generated connections are placed in a folder. You can then edit the connections as needed.

To edit the information about an existing database connection, right-click the connection name in the Connections navigator display and select **Properties**. Use the dialog box to modify information about the connection, but do not change the connection name. (See [Section 5.13, "Create/Edit/Select Database Connection"](#).)

To organize connection groups using folders, see [Section 1.4.1, "Using Folders to Group Connections"](#).

To export information about the existing database connections into an XML file that you can later use for importing connections, right-click Connections in the Connections navigator display and select **Export Connections**. Use the dialog box to specify the connections to be exported (see [Section 5.25, "Export/Import Connection Descriptors"](#)).

To import connections that had previously been exported (adding them to any connections that may already exist in SQL Developer), right-click Connections in the Connections navigator display and select **Import Connections**. Use the dialog box to specify the connections to be imported (see [Section 5.25, "Export/Import Connection Descriptors"](#)).

To perform limited database management operations if you are connected AS SYSDBA, right-click the connection name in the Connections navigator display and select **Manage Database**. You can click to refresh the read-only display of memory (SGA and PGA) and tablespace information. If a listener is running with a static listener configured for the database, you can also click to start and stop the database.

To perform remote debugging if you are using the Sun Microsystems's Java Platform Debugger Architecture (JPDA) and you would like the debugger to listen so that a debuggee can attach to the debugger, right-click the connection name in the Connections navigator display and select **Remote Debug**. Use the dialog box to specify remote debugging information (see [Section 5.52, "Debugger - Attach to JPDA"](#)).

To estimate or compute statistics for objects in a database schema, right-click the connection name in the Connections navigator display and select **Gather Schema Statistics**. Statistics are used to optimize SQL execution.

To generate documentation (in HTML format (comparable to Javadoc for Java classes) about a schema, right-click the connection name in the Connections navigator display and select **Generate DB Doc**. To view the generated documentation, open the index.html file in the output directory that you specified.

To reset an expired password for the database user associated with a connection, right-click the connection name in the Connections navigator display and select **Reset Password** (enabled only if an OCI (thick) driver is available).

To rename a connection, right-click the connection name in the Connections navigator display and select **Rename Connection**.

To delete a connection (that is, delete it from SQL Developer, *not* merely disconnect from the current connection), right-click the connection name in the Connections navigator display and select **Delete**. Deleting a connection does not delete the user associated with that connection.

To connect using an existing connection, expand its node in the Connections navigator, or right-click its name and select **Connect**. A SQL Worksheet window is also opened for the connection (see [Section 1.7, "Using the SQL Worksheet"](#)). To create a separate unshared worksheet for a connection, use Ctrl+Shift+N.

To disconnect from the current connection, right-click its name in the Connections navigator and select **Disconnect**.

To specify a preference for using an OCI (thick, Type 2) driver (if available) instead of a JDBC (thin) driver for basic and TNS (network alias) database connections, enable the **Use OCI/Thick driver** option under the [Database: Advanced](#) user preferences.

1.4.1 Using Folders to Group Connections

You can use folders in the Connections navigator to organize connections into groups: for example, one folder for connections on your local system, another for connections on the test system, and another for connections on the production system.

To create a folder to hold connections, right-click the name in the Connections navigator of a connection to be added to the folder, select **Add to Folder** and then **New Folder**, and specify the folder name (such as Local Connections).

To add more connections to a folder, right-click the name in the Connections navigator of a connection to be added to the folder, and select **Add to Folder** and then the name of the folder into which to add the connection.

To move a connection from one folder to another folder, right-click the connection name under its current folder, select **Add to Folder**, and then either the name of the destination folder or **New Folder** to move the connection to a new folder to be created.

To remove a connection from the folder, right-click the connection name under the folder and select **Remove from Folder**. (This does *not* delete the connection; it is moved to the top level in the Connections navigator hierarchy display.)

To remove a folder, right-click the folder name select **Remove Folder**. (This does *not* delete any connections that are in the folder; these connections are moved to the top level in the Connections navigator hierarchy display.)

To rename a folder, right-click the folder name, select **Rename Folder**, and specify the new name.

1.4.2 Sharing of Connections

By default, each connection in SQL Developer is shared when possible. For example, if you open a table in the Connections navigator and two SQL Worksheets using the same connection, all three panes use one shared connection to the database. In this example, a commit operation in one SQL Worksheet commits across all three panes. If you want a dedicated session, you must duplicate your connection and give it another name. Sessions are shared by name, not connection information, so this new connection will be kept separate from the original.

1.4.3 Advanced Security for JDBC Connection to the Database

You are encouraged to use Oracle Advanced Security to secure a JDBC or OCI connection to the database. Both the JDBC OCI and the JDBC Thin drivers support at least some of the Oracle Advanced Security features. If you are using the OCI driver, you can set relevant parameters in the same way that you would in any Oracle client setting. The JDBC Thin driver supports the Oracle Advanced Security features through a set of Java classes included with the JDBC classes in a Java Archive (JAR) file and supports security parameter settings through Java properties objects.

For more information about using Oracle Advanced Security, see *Oracle Database JDBC Developer's Guide*.

1.4.4 Connections with Operating System (OS) Authentication

When you create a connection to an Oracle database that is using operating system (OS) authentication, you can omit the user name and password; that is, specify a connection name and all the other necessary information, except do not specify a user name or password. For information about using external authentication, including the use of the OS_AUTHENT_PREFIX and REMOTE_OS_AUTHENT database initialization parameters, see *Oracle Database Security Guide*.

If you omit the user name and password trying to create a connection to a system that is not configured for external authentication, an error message is displayed.

1.4.5 Connections with Proxy Authentication

Proxy authentication enables one JDBC connection to act as a proxy for other JDBC connections. If you use the Proxy Connection option when you create a database connection, the connection will be used to connect as the specified user for the connection, but authenticated using the user name and either the password or distinguished name of the proxy user. For information about using a middle tier server for proxy authentication, see *Oracle Database Security Guide*.

For example, to create connection for a user named PROXY_USER but connecting using the user name and password of existing database user SCOTT, follow these steps.

1. Create the proxy user and grant it the appropriate privileges:

```
CREATE USER proxy_user IDENTIFIED BY <password>;
ALTER USER proxy_user GRANT CONNECT THROUGH scott AUTHENTICATED USING PASSWORD;
GRANT create session TO proxy_user;
. . .<Grant other privileges as needed.>
```

2. Create a new database connection. For example: connection name = proxy_conn, user name = scott, password = <password for scott>.
3. Enable (check) **Proxy Connection**.

4. In the Oracle Proxy Connection dialog box, select **User Name** for Proxy Type
5. For Proxy User, enter **PROXY_USER**; and for Proxy Password, enter the password for the PROXY_USER database user.
6. Click OK to close the Oracle Proxy Connection dialog box.
7. Complete any other necessary connection information, and click Connect to create the connection.

In this example, when you connect using the proxy_conn connection, the user name and password for user SCOTT are used to connect to the database, but the connection sees those database objects that the PROXY_USER user is permitted to see.

The preceding instructions cause two sessions to be started (one for the proxy user and one for the proxy client) when the connection is opened. If you want to have a single session (with no second password or distinguished name required) started, you can follow these steps instead after you create the proxy user and grant it appropriate privileges:

1. Create a new database connection. For example: connection name = proxy_conn, user name = proxy_user[scott], password = <password for proxy_user>.
2. For Connection Type, specify TNS.
(Note: Do *not* enable Proxy Connection.)
3. For Network Alias, select the network alias for the database for the connection.
4. Complete any other necessary connection information, and click Connect to create the connection.

1.5 Entering and Modifying Data

You can use SQL Developer to enter data into tables and to edit and delete existing table data. To do any of these operations, select the table in the Connections navigator, then click the **Data** tab in the table detail display. The following figure shows the Data pane for a table named BOOKS, with a filter applied to show only books whose rating is 10, and after the user has clicked in the Title cell for the first book.

BOOK_ID	TITLE	AUTHOR_LAST...	AUTHOR_F...	RATING
1 A.1111	Moby Dick	Melville	Herman	10
2 A.5555	Software Wi...	Abugov	D.	10

Icons and other controls under the Data tab provide the following options:

- **Freeze View** (the pin) keeps that object's tab and information in the window when you click another object in the Connections navigator; a separate tab and display are created for that other object. If you click the pin again, the object's display is available for reuse.
- **Refresh** queries the database to update the data display. If a filter is specified, the refresh operation uses the filter.
- **Insert Row** adds an empty row after the selected row, for you to enter new data.

- **Delete Selected Row(s)** marks the selected rows for deletion. The actual deletion does not occur until you commit changes.
- **Commit Changes** ends the current transaction and makes permanent all changes performed in the transaction.
- **Rollback Changes** undoes any work done in the current transaction.
- **Sort** displays a dialog box for selecting columns to sort by. For each column, you can specify ascending or descending order, and you can specify that null values be displayed first.
- **Filter** enables you to enter a SQL predicate (WHERE clause text without the WHERE keyword) for limiting the display of data. For example, to show only rows where the RATING column value is equal to 10, specify: `rating = 10`
- **Filter Column** enables you to enter a partial value (such as a number or a string; at least two characters for a string), to limit the dialog box display to items containing the partial value, so that you can then select the one item to appear in the grid. For example, entering EMP for column names might show a list of columns including EMPLOYEE_ID and IS_TEMP.
- **Actions** displays a menu with actions relevant to the table.

When you enter a cell in the grid, you can directly edit the data for many data types, and for all data types you can click the ellipsis (...) button to edit the data. For binary data you cannot edit the data in the cell, but must use the ellipsis button.

In the data grid, the context menu (right-click) includes the following commands:

- **Single Record View** displays the [Single Record View](#) dialog box, which enables you to edit data for a table or view, one record at a time.
- **Auto-fit All Columns** adjusts the width of all columns according to your specification (by column header, by column data, or best fit).
- **Auto-fit Selected Columns** adjusts the width of the selected columns according to your specification (by column header, by column data, or best fit).
- **Count Rows** displays the number of rows in the table.
- **Publish to Apex** (if Application Express is installed) creates a small Application Express application based on the data. It displays a dialog box in which you specify the following for the application to be created: workspace, application name, theme, page name, and SQL statement for generating the report.
- **Export Data** enables you to export some or all of the table data to a file or to the system clipboard, in any of the following formats: XML (XML tags and data), CSV (comma-separated values including a header row for column identifiers), SQL Insert (INSERT statements), or SQL Loader (SQL*Loader control file). After you select a format, the [Export Table Data](#) dialog box is displayed.

You can copy and paste data between table grid cells and cells in a Microsoft Excel worksheet.

To copy table data to the clipboard, click the column header (for all column data) or select specific cells and press Ctrl+C; to copy the column header text along with the table data, press Ctrl+Shift+C.

To sort the display of rows by values within a column, double-click the column header; to switch between ascending and descending sorting, double-click the up/down arrow in the column header.

In the Data pane for a table or view, you can **split** the display vertically or horizontally to see two (or more) parts independently by using the split box (thin blue rectangle), located to the right of the bottom scroll bar and above the right scroll bar.

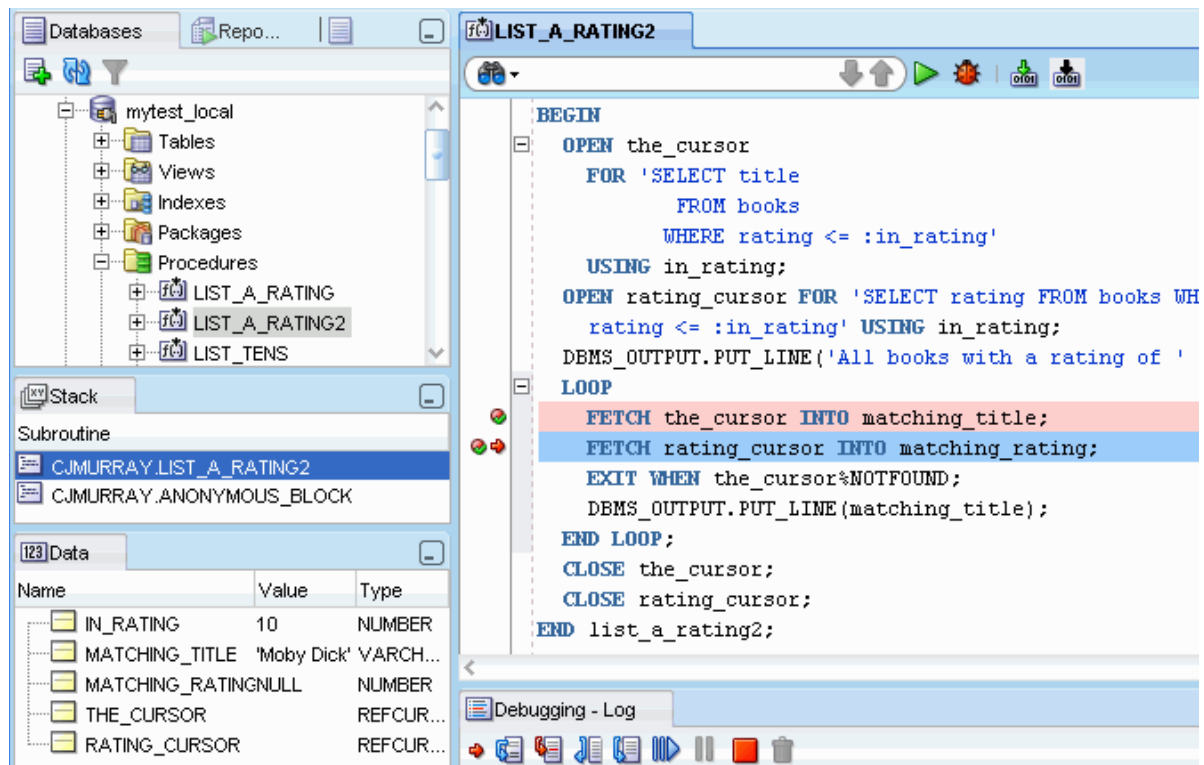
In the Data pane, the acceptable format or formats for entering dates may be different from the date format required by SQL*Plus.

1.6 Running and Debugging Functions and Procedures

You can use SQL Developer to run and debug PL/SQL subprograms (functions and procedures).

- To run a subprogram, click its name in the Connections navigator; then either right-click and select Run, or click the Edit icon and then click the Run icon above its source listing.
- To debug a subprogram, click its name in the Connections navigator. If the procedure in its current form has not already been compiled for debug, right-click and select Compile for Debug. Then click the Edit icon and click the Debug icon above its source listing.

In both cases, a code editing window is displayed. The following figure shows the code editing window being used to debug a procedure named LIST_A_RATING2, which is used for tutorial purposes in [Section 4.8, "Debug a PL/SQL Procedure"](#).



In the code editing window, under the tab with the name of the subprogram, is a toolbar, and beneath it is the text of the subprogram, which you can edit. You can set and unset breakpoints for debugging by clicking to the left of the thin vertical line beside each statement with which you want to associate a breakpoint. (When a breakpoint is set, a red circle is displayed.)

The toolbar under the tab for the subprogram name includes the icons shown in the following figure.



- **Run** starts normal execution of the subprogram, and displays the results in the **Running - Log** tab.
- **Debug** starts execution of the subprogram in debug mode, and displays the **Debugging - Log** tab, which includes the debugging toolbar for controlling the execution.
- **Make - Compile for Debug** performs a PL/SQL compilation of the subprogram so that it can be debugged.
- **Make - Compile** performs a PL/SQL compilation of the subprogram.
- **Profile** displays the [Run/Debug/Profile PL/SQL](#) dialog box.

The **Debugging - Log** tab under the code text area contains the debugging toolbar and informational messages. The debugging toolbar has the icons shown in the following figure.



- **Find Execution Point** goes to the execution point (the next line of source code to be executed by the debugger).
- **Step Over** bypasses the next subprogram (unless the subprogram has a breakpoint) and goes to the next statement after the subprogram. If the execution point is located on a subprogram call, it runs that subprogram without stopping (instead of stepping into it), then positions the execution point on the statement that follows the call. If the execution point is located on the last statement of a subprogram, Step Over returns from the subprogram, placing the execution point on the line of code that follows the call to the subprogram from which you are returning.
- **Step Into** executes a single program statement at a time. If the execution point is located on a call to a subprogram, Step Into steps into that subprogram and places the execution point on its first statement. If the execution point is located on the last statement of a subprogram, Step Into returns from the subprogram, placing the execution point on the line of code that follows the call to the subprogram from which you are returning.
- **Step Out** leaves the current subprogram and goes to the next statement.
- **Step to End of Method** goes to the last statement of the current subprogram.
- **Resume** continues execution.
- **Pause** halts execution but does not exit, thus allowing you to resume execution.
- **Terminate** halts and exits the execution. You cannot resume execution from this point; instead, to start running or debugging from the beginning of the subprogram, click the Run or Debug icon in the Source tab toolbar.

The **Breakpoints** tab displays breakpoints, both system-defined and user-defined.

The **Smart Data** tab displays information about variables, using your [Debugger: Smart Data](#) preferences. You can also specify these preferences by right-clicking in the Smart Data window and selecting **Preferences**.

The **Data** tab displays information about variables, using your [Debugger: Data](#) preferences. You can also specify these preferences by right-clicking in the Data window and selecting **Preferences**.

The **Watches** tab displays information about watches (see [Section 1.6.5, "Setting Expression Watches"](#)).

If the function or procedure to be debugged is on a remote system, see also [Section 1.6.2, "Remote Debugging"](#).

1.6.1 Using Bookmarks When Editing Functions and Procedures

When you are editing a long function or procedure, you may find it convenient to create bookmarks in the code so that you can easily navigate to points of interest.

To create or remove a bookmark, click **Navigate**, then **Toggle Bookmark**. When a bookmark is created, an icon appears to the left of the thin vertical line.

To go to a specific bookmark, click **Navigate**, then **Go to Bookmark**. To go to the next or previous bookmark, click **Navigate**, then **Go to Next Bookmark** or **Go to Previous Bookmark**, respectively.

To remove all bookmarks from the currently active editing window for a function or procedure or from all open editing windows, click **Navigate**, then **Remove Bookmarks from File** or **Remove All Bookmarks**, respectively.

You can also go to a specific line or to your last edit by clicking **Navigate**, then **Go to Line** or **Go to Last Edit**, respectively.

1.6.2 Remote Debugging

To debug a function or procedure for a connection where the database is on a different host than the one on which you are running SQL Developer, you can perform remote debugging. Remote debugging involves many of the steps as for local debugging; however, do the following before you start the remote debugging:

1. Use an Oracle client such as SQL*Plus to issue the debugger connection command. Whatever client you use, make sure that the session which issues the debugger connection commands is the same session which executes your PL/SQL program containing the breakpoints. For example, if the name of the remote system is `remote1`, use the following SQL*Plus command to open a TCP/IP connection to that system and the port for the JDWP session:

```
EXEC DBMS_DEBUG_JDWP.CONNECT_TCP('remote1', '4000');
```

The first parameter is the IP address or host name of the remote system, and the second parameter is the port number on that remote system on which the debugger is listening.

2. Right-click the connection for the remote database, select **Remote Debug**, and complete the information in the [Debugger - Attach to JPDA](#) dialog box.

Then, follow the steps that you would for local debugging (for example, see [Section 4.8, "Debug a PL/SQL Procedure"](#)).

1.6.3 Displaying SQL Trace (.trc) Files

If you have any SQL Trace (.trc) output files, you can display them in SQL Developer as an alternative to using the TKPROF program to format the contents of the trace file. To open a .trc file in SQL Developer and see an attractive, effective display of the

information, click **File**, then **Open**, and specify the file; or drag the file's name or icon into the SQL Developer window.

You can then examine the information in the List View, Statistics View, and History panes, with each pane including options for filtering and controlling the display.

For information about SQL Trace and TKPROF, see *Oracle Database Performance Tuning Guide*.

1.6.4 Using the PL/SQL Hierarchical Profiler

For an Oracle Database Release 11.1 or later connection, you can use the PL/SQL hierarchical profiler to identify bottlenecks and performance-tuning opportunities in PL/SQL applications. Profiling consists of the two steps: running the PL/SQL module in profiling mode, and analyzing the reports. In addition, some one-time setup work is required the first time you use profiling in SQL Developer.

To initiate profiling, right-click the name of the function or procedure in the Connections navigator hierarchy and select **Profile**, or click the Profile button on the PL/SQL source editor toolbar. After the function or procedure is run in profiling mode, the profiler reports are located at the Execution Profiles tab of the object viewer window. You can review subprogram-level execution summary information, such as:

- Number of calls to the subprogram
- Time spent in the subprogram itself (function time or self time)
- Time spent in the subprogram itself and in its descendent subprograms (subtree time)
- Detailed parent-children information, including all subprograms that a given subprogram called (that is, children of the given subprogram)

For more information about using the PL/SQL hierarchical profiler, see *Oracle Database Advanced Application Developer's Guide*.

1.6.5 Setting Expression Watches

A watch enables you to monitor the changing values of variables or expressions as your program runs. After you enter a watch expression, the Watches window displays the current value of the expression. As your program runs, the value of the watch changes as your program updates the values of the variables in the watch expression.

A watch evaluates an expression according to the current context which is controlled by the selection in the Stack window. If you move to a new context, the expression is reevaluated for the new context. If the execution point moves to a location where any of the variables in the watch expression are undefined, the entire watch expression becomes undefined. If the execution point returns to a location where the watch expression can be evaluated, the Watches window again displays the value of the watch expression.

To open the Watches window, click **View**, then **Debugger**, then **Watches**.

To add a watch, right-click in the Watches window and select **Add Watch**. To edit a watch, right-click in the Watches window and select **Edit Watch**.

1.7 Using the SQL Worksheet

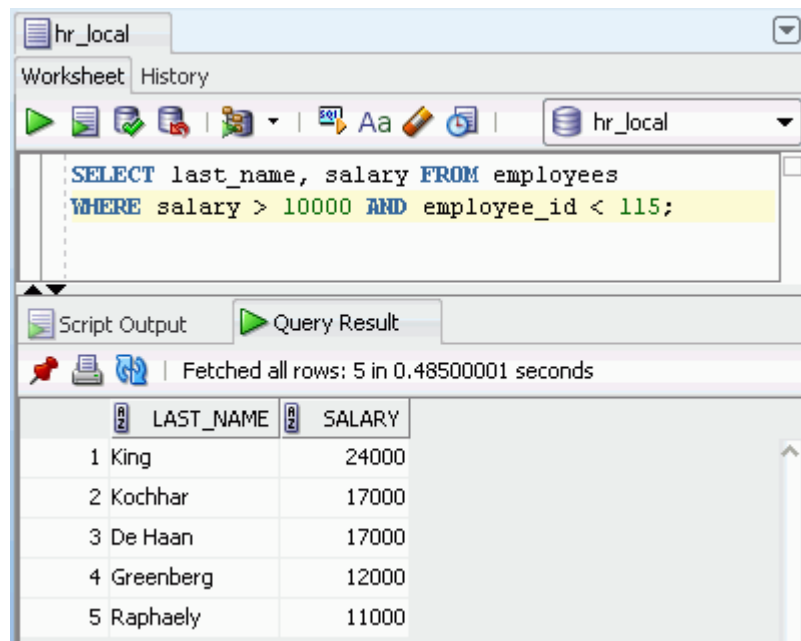
You can use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements. You can specify any actions that can be processed by the database

connection associated with the worksheet, such as creating a table, inserting data, creating and editing a trigger, selecting data from a table, and saving that data to a file.

You can display a SQL Worksheet by right-clicking a connection in the Connections navigator and selecting **Open SQL Worksheet**, by selecting **Tools** and then **SQL Worksheet**, or by clicking the **Use SQL Worksheet** icon under the menu bar. In the Select Connection dialog box, select the database connection to use for your work with the worksheet. You can also use that dialog box to create and edit database connections. (You can have a SQL Worksheet window open automatically when you open a database connection by enabling the appropriate SQL Developer user preference under Database Connections.)

To create a separate unshared worksheet for a connection, use Ctrl+Shift+N.

The SQL Worksheet has the user interface shown in the following figure:



SQL Worksheet toolbar (under the Worksheet tab): Contains icons for the following operations:

- **Execute Statement** executes the statement at the mouse pointer in the Enter SQL Statement box. The SQL statements can include bind variables and substitution variables of type VARCHAR2 (although in most cases, VARCHAR2 is automatically converted internally to NUMBER if necessary); a pop-up box is displayed for entering variable values.
- **Run Script** executes all statements in the Enter SQL Statement box using the [Script Runner](#). The SQL statements can include substitution variables (but not bind variables) of type VARCHAR2 (although in most cases, VARCHAR2 is automatically converted internally to NUMBER if necessary); a pop-up box is displayed for entering substitution variable values.
- **Commit** writes any changes to the database, and ends the transaction; also clears any output in the Results and Script Output panes.
- **Rollback** discards any changes without writing them to the database, and ends the transaction; also clears any output in the Results and Script Output panes.
- **Cancel** stops the execution of any statements currently being executed.

- **Monitor SQL Status** (Oracle Database Release 11.1 and later only) calls the real-time SQL monitoring feature of Oracle Database, enabling you to monitor the performance of SQL statements while they are executing.
- **Explain Plan** generates the execution plan for the statement (internally executing the EXPLAIN PLAN statement). To see the execution plan, click the Explain tab. For more information, see [Section 1.7.3, "Execution Plan"](#).
- **Autotrace** generates trace information for the statement. To see the [Autotrace Pane](#), click the Autotrace tab.
- **Clear** erases the statement or statements in the Enter SQL Statement box.
- To the right of these icons is a drop-down list for changing the **database connection** to use with the worksheet.

The context menu (right-click, or Shift+F10) includes the preceding SQL Worksheet toolbar operations, plus the following operations (some depending on the type of object displayed in the worksheet):

- **Print File** prints the contents of the Enter SQL Statement box.
- **Cut, Copy, Paste, and Select All** have the same meanings as for normal text editing operations.
- **Query Builder** opens the [Query Builder](#) dialog box, where you can create a SELECT statement by dragging and dropping table and view names and by graphically specifying columns and other elements of the query.
- **Refactoring** enables you to do the following on selected text: switch character case (to upper/lower/initcap), extract the sequence of PL/SQL statements to a procedure, or rename the local variable.
- **Format** formats the SQL statement (capitalizing the names of statements, clauses, keywords, and so on).
- **Advanced Format** displays a dialog box where you can specify the output destination and output type for the formal operation.
- **Quick Outline** displays the Outline pane with a graphical outline of the object displayed in the worksheet (if an outline is relevant for this type of object). You can click a node in the outline to go to the associated place in the text in the worksheet.
- **Popup Describe**, if the name of a database object is completely selected, displays a window with tabs and information appropriate for that type of object (see [Section 5.54, "Describe Object Window"](#)).
- **Save Snippet** opens the [Save Snippet \(User-Defined\)](#) dialog box with the selected text as the snippet text.

Enter SQL Statement: The statement or statements that you intend to execute. For multiple statements, each non-PL/SQL statement must be terminated with either a semicolon or (on a new line) a slash (/), and each PL/SQL statement must be terminated with a slash (/) on a new line. SQL keywords are automatically highlighted. To format the statement, right-click in the statement area and select **Format SQL**.

You can drag some kinds of objects from the Connections navigator and drop them into the Enter SQL Statement box:

- If you drag and drop a table or view, by default a SELECT statement is constructed with all columns in the table or view. You can then edit the statement, for example, modifying the column list or adding a WHERE clause.

- If you drag and drop a function or procedure, a snippet-like text block is constructed for you to edit when including that object in a statement.

To view details for any object, you can select its name in the Enter SQL Statement box and select **Popup Describe** from the context menu (or press Shift+F4). For example, if you select a table name and press Shift+F4, information about Columns, Constraints, Grants, and so on is displayed; or if you select a procedure name and press Shift+F4, information about Code, Grants, Dependencies, References, and Details is displayed.

Tabs display panes with the following information:

- **Results:** Displays the results of the most recent Execute Statement operation.
- **Explain:** Displays the output if you clicked the Explain Execution Plan icon (see [Section 1.7.3, "Execution Plan"](#)).
- **Script Output:** Displays the output if you clicked the Run Script icon (see [Section 1.7.2, "Script Runner"](#)).
- **DBMS Output:** Displays the output of DBMS_OUTPUT package statements (see [Section 1.7.5, "DBMS Output Pane"](#)).
- **OWA Output:** Displays Oracle Web Agent (MOD_PLSQL) output (see [Section 1.7.6, "OWA Output Pane"](#)).

To toggle the relative heights of the Enter SQL Statement area and the area for tabs and display panes, press Ctrl+Alt+L. You can also manually adjust the heights.

1.7.1 SQL*Plus Statements Supported and Not Supported in SQL Worksheet

The SQL Worksheet supports some SQL*Plus statements. SQL*Plus statements must be interpreted by the SQL Worksheet before being passed to the database; any SQL*Plus that are not supported by the SQL Worksheet are ignored and not passed to the database.

The following SQL*Plus statements are supported by the SQL Worksheet:

```
@
@@
/
acc[ept]
autotrace
clear screen
conn[ect]
def[ine]
desc[ribe]
doc[ument]
echo
errors
esc[ape]
exec[ute]
exit
feed[back]
ho[st]
pagesize
pau[se]
print
pro[mpt]
quit
rem[ark]
roll[back]
set pau[se] {ON | OFF}
```

```
sta[rt]
spo[ol]
term[out]
timi[ng]
undef[ine]
user
var[iable]
ver[ify]
whenever
xquery
```

The following SQL*Plus statements are *not* supported by the SQL Worksheet:

```
a[ppend]
archive
attr[ibute]
bre[ak]
bti[tile]
c[hange]
col[umn]
comp[ute]
copy
del
disc[onnect]
ed[it]
get
help
i[nput]
l[ist]
newpage
oradebug
passw[ord]
r[un]
recover
repf[ooter]
reph[eader]
sav[e]
startup
sho[w]
shu[tdown]
spo[ol]
startup
store
tti[tile]
```

1.7.2 Script Runner

The script runner emulates a limited set of SQL*Plus features. You can often enter SQL and SQL*Plus statements and execute them by clicking the **Run Script** icon. The Script Output pane displays the output.

The SQL*Plus features available in the script runner include @, @@, CONNECT, EXIT, QUIT, UNDEFINE, WHENEVER, and substitution variables. For example, to run a script named c:\myscripts\mytest.sql, type @c:\myscripts\mytest in the Enter SQL Statement box, and click the drop-down next to the Execute Statement icon and select Run Script.

The following considerations apply to using the SQL Developer script runner:

- You cannot use bind variables. (The Execute SQL Statement feature does let you use bind variables of type VARCHAR2, NUMBER, and DATE.)

- For substitution variables, the syntax `&&variable` assigns a permanent variable value, and the syntax `&variable` assigns a temporary (not stored) variable value.
- For EXIT and QUIT, commit is the default behavior, but you can specify rollback. In either case, the context is reset: for example, WHENEVER command information and substitution variable values are cleared.
- DESCRIBE works for most, but not all, object types for which it is supported in SQL*Plus.
- For SQL*Plus commands that are not supported, a warning message is displayed.
- SQL*Plus comments are ignored.
- For XMLType data, data in the column is displayed as "SYS.XMLDATA" if the database connection uses a JDBC Thin driver, but the expanded XML values are displayed if the connection uses an OCI (thick, Type 2) driver.

If you have SQL*Plus available on your system, you may want to use it instead of the script runner.

1.7.3 Execution Plan

The Execute Explain Plan icon generates the execution plan, which you can see by clicking the Explain Plan tab. The execution plan is the sequence of operations that will be performed to execute the statement. An execution plan shows a row source tree with the hierarchy of operations that make up the statement. For each operation, it shows the ordering of the tables referenced by the statement, access method for each table mentioned in the statement, join method for tables affected by join operations in the statement, and data operations such as filter, sort, or aggregation.

In addition to the row source tree, the plan table displays information about optimization (such as the cost and cardinality of each operation), partitioning (such as the set of accessed partitions), and parallel execution (such as the distribution method of join inputs). For more information, see the chapter about using EXPLAIN PLAN in *Oracle Database Performance Tuning Guide*.

1.7.4 Autotrace Pane

The Autotrace pane displays trace-related information when you execute the SQL statement by clicking the **Autotrace** icon. Most of the specific information displayed is determined by the [SQL Developer Preferences](#) for [Database: Autotrace/Explain Plan](#). If you cancel a long-running statement, partial execution statistics are displayed.

This information can help you to identify SQL statements that will benefit from tuning. For example, you may be able to optimize predicate handling by transitively adding predicates, rewriting predicates using Boolean algebra principles, moving predicates around in the execution plan, and so on. For more information about tracing and autotrace, see the chapter about tuning in *SQL*Plus User's Guide and Reference*.

To use the autotrace feature, the database user for the connection must have the SELECT_CATALOG_ROLE and SELECT ANY DICTIONARY privileges.

1.7.5 DBMS Output Pane

The PL/SQL DBMS_OUTPUT package enables you to send messages from stored procedures, packages, and triggers. The PUT and PUT_LINE procedures in this package enable you to place information in a buffer that can be read by another trigger, procedure, or package. In a separate PL/SQL procedure or anonymous block,

you can display the buffered information by calling the `GET_LINE` procedure. The DBMS Output pane is used to display the output of that buffer. This pane contains icons and other controls for the following operations:

- **Add New DBMS Output Tab:** Prompts you to specify a database connection, after which a tab is opened within the DBMS Output pane for that connection, and the `SET SERVEROUTPUT` setting is turned on so that any output is displayed in that tab. (To stop displaying output for that connection, close the tab.)
- **Clear:** Erases the contents of the pane.
- **Save:** Saves the contents of the pane to a file that you specify.
- **Print:** Prints the contents of the pane.
- **Buffer Size:** For databases before Oracle Database 10.2, limits the amount of data that can be stored in the `DBMS_OUTPUT` buffer. The buffer size can be between 1 and 1000000 (1 million).
- **Poll:** The interval (in seconds) at which SQL Developer checks the `DBMS_OUTPUT` buffer to see if there is data to print. The poll rate can be between 1 and 15.

1.7.6 OWA Output Pane

OWA (Oracle Web Agent) or `MOD_PLSQL` is an Apache (Web Server) extension module that enables you to create dynamic Web pages from PL/SQL packages and stored procedures. The OWA Output pane enables you to see the HTML output of `MOD_PLSQL` actions that have been executed in the SQL Worksheet. This pane contains icons for the following operations:

- **Add New OWA Output Tab:** Prompts you to specify a database connection, after which a tab is opened within the OWA Output pane for that connection, and entries written to the OWA output buffer are displayed in that tab. (To stop displaying output for that connection, close the tab.)
- **Clear:** Erases the contents of the pane.
- **Save:** Saves the contents of the pane to a file that you specify.
- **Print:** Prints the contents of the pane.

1.7.7 SQL History

You can click **View**, then **History** (or press **F8** in the SQL Worksheet) to view a dockable window with SQL statements and scripts that you have executed, and optionally select one or more statements to have them either replace the statements currently on the SQL Worksheet or be added to the statements currently on the SQL Worksheet.

You can click on a column heading to sort the rows by the values in that column.

The SQL history list will not contain any statement that can include a password. Such statements include (but are not necessarily limited to) `CONNECT`, `ALTER USER`, and `CREATE DATABASE LINK`.

You can control the maximum number of statements in the history by setting the SQL History Limit preference (see [Database: Worksheet](#) preferences).

Append: Appends the selected statement or statements to any statements currently on the SQL Worksheet. You can also append the selected statement or statements by

dragging them from the SQL History window and dropping them at the desired location on the SQL Worksheet.

Replace: Replaces any statements currently on the SQL Worksheet with the selected statement or statements.

Clear History: Removes the selected statement or statements (or all statements if no statements are selected) from the SQL history. (You will be asked to confirm this action.)

Filter: If you type a string in the text box and click Filter, only SQL statements containing that string are displayed.

1.7.8 Gauges: In the SQL Worksheet and User-Defined Reports

You can use graphical gauges to display query results in the SQL Worksheet and in user-defined reports. In both cases, you need to specify the name of the value column for the gauge data, and minimum and maximum values on the gauge, and the values to be shown as low and high on the gauge (usually between the minimum and maximum values). In the SQL Worksheet, the required structure for the value to be selected is:

```
'SQLDEV:GAUGE:<min>:<max>:<low>:<high>:' || <value-column>
```

For example, to display the last name and the salary in gauge format, where the gauge shows from 1000 to 30000 with below 10000 as low and above 18000 as high, for employees with ID numbers less than a number to be specified, connect to the supplied HR schema and execute the following query:

```
SELECT last_name, 'SQLDEV:GAUGE:1000:30000:10000:18000:' || salary
FROM employees WHERE employee_id < :employee_id
```

If you specify 104 as the bind variable value, the output appears as shown in the following figure:

The screenshot shows the SQL Developer Results window with a table of employee data. The table has two columns: LAST_NAME and a gauge column. The gauge column displays a horizontal bar for each employee, with a yellow segment indicating the salary value relative to the gauge scale. The gauge scale ranges from 1000 to 30000, with 10000 as the low value and 18000 as the high value. The employees listed are King, Kochhar, De Haan, and Hunold.

	LAST_NAME	'SQLDEV:GAUGE:1000:30000:10000:18000:' SALARY
1	King	[Gauge bar]
2	Kochhar	[Gauge bar]
3	De Haan	[Gauge bar]
4	Hunold	[Gauge bar]

For a user-defined gauge report, the query must specify only the value column, the minimum and maximum values, and the low and high values, and optionally a WHERE clause. The required structure for the query (before any optional WHERE clause) is:

```
SELECT <value-column>, <min>, <max>, <low>, <high> FROM <table-name>
```

For example, to create a report of salaries in gauge dial format, with the same values and WHERE clause as in the preceding query, right-click on User Defined Reports in the Reports navigator and select **Add Report**. In the Add Report dialog box, specify a report name; for **Style**, select **Gauge**; and for **SQL**, enter the following:

```
SELECT salary, 1000, 30000, 10000, 18000 FROM employees
WHERE employee_id < :EMPLOYEE_ID;
```

Click the **Chart Details** tab near the bottom of the box; for **Chart Type**, select DIAL; for **Query Based**, select true; and click **Apply**.

Use the Reports navigator to view the newly created user-defined report. For **Connection**, specify one that connects to the HR sample schema. For the bind variable value, specify 104. The report shows four semicircular dials, each with a label containing the salary amount and a "needle" pointing to an appropriate place on the dial.

1.8 Using Snippets to Insert Code Fragments

Snippets are code fragments, such as SQL functions, Optimizer hints, and miscellaneous PL/SQL programming techniques. Some snippets are just syntax, and others are examples. You can insert and edit snippets when you are using the SQL Worksheet or creating or editing a PL/SQL function or procedure.

To display snippets, from the **View** menu, select **Snippets**. In the snippets window (on the right side), use the drop-down to select a group (such as Aggregate Functions or Character Functions). In most cases, the fragments in each group do not represent all available objects in that logical grouping, or all formats and options of each fragment shown. For complete and detailed information, see the Oracle Database documentation.

To insert a snippet into your code in a SQL Worksheet or in a PL/SQL function or procedure, drag the snippet from the snippets window and drop it into the desired place in your code; then edit the syntax so that the SQL function is valid in the current context. To see a brief description of a SQL function in a tooltip, hold the pointer over the function name.

For example, you could type `SELECT` and then drag `CONCAT(char1, char2)` from the Character Functions group. Then, edit the `CONCAT` function syntax and type the rest of the statement, such as in the following:

```
SELECT CONCAT(title, ' is a book in the library.') FROM books;
```

1.8.1 User-Defined Snippets

You can create and edit snippets. User-defined snippets are intended mainly to enable you to supplement the Oracle-supplied snippets, although you are also permitted to replace an Oracle-supplied snippet with your own version.

When you create a user-defined snippet, you can add it to one of the Oracle-supplied snippet categories (such as Aggregate Functions) or to a category that you create. If you add a snippet to an Oracle-supplied category and if your snippet has the same name as an existing snippet, your snippet definition replaces the existing one. (If you later upgrade to a new version of SQL Developer and if you choose to preserve your old settings, your old user-defined snippets will replace any Oracle-supplied snippets of the same name in the new version of SQL Developer.)

To create a snippet, do any of the following:

- Open the Snippets window and click the **Add User Snippets** icon.
- Select text for the snippet in the SQL Worksheet window, right-click, and select **Save Snippet**.
- Click the **Add User Snippet** icon in the [Edit Snippets \(User-Defined\)](#) dialog box.

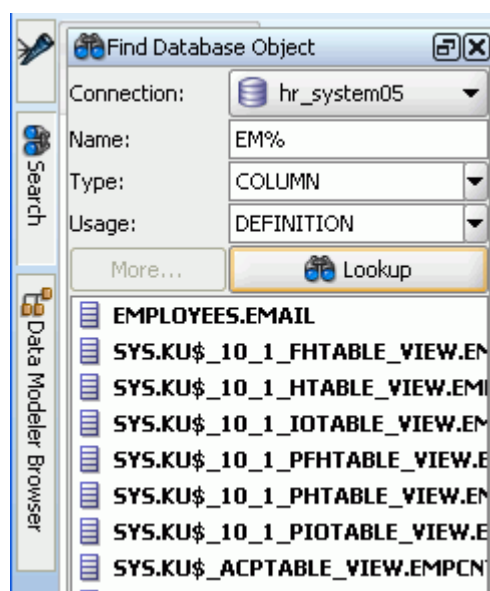
To edit an existing user-defined snippet, click the **Edit User Snippets** icon in the Snippets window.

Information about user-defined snippets is stored in a file named UserSnippets.xml under the directory for user-specific information. For information about the location of this information, see [Section 1.13, "Location of User-Related Information"](#).

1.9 Finding Database Objects

You can find various types of objects (tables, columns, declarations within functions or procedures, and so on) associated with an Oracle database connection and open editing panes to work with those objects. Click the **Search** icon on the left to display the Find Database Object window, where you can specify a connection name and search criteria (click **More** to display all criteria).

The following figure shows part of the Find Database Objects pane with results from a search for column definitions associated with a connection named hr_system05 where the column name starts with EM.



Connection: Database connection to use for the search.

Name: An object name or a string containing one or more wildcard characters. For example: EM% for all names starting with EM.

Type: Type of object for which to restrict the search.

Usage: Usage of the object. May or may not be relevant, depending on the type of object.

Click the **Lookup** icon to display objects that meet the specified criteria. To view or edit one of the objects (or the parent object that contains the specified object), double-click or right-click its name in the results display.

You can detach, move, and dock the Find Database Objects pane by clicking and holding the tab, and dragging and positioning the pane.

1.10 Using Versioning

SQL Developer provides integrated support for the Subversion versioning and source control system, and you can add support for other such systems as extensions by clicking Help, then Check for Updates. Available extensions include CVS (Concurrent Versions System), Serena Dimensions, and Perforce. The SQL Developer

documentation does not provide detailed information about the concepts and operations of such systems; it assumes that you know them or can read about them in the product documentation.

- For information about CVS, see <http://ximbiot.com/cvs/wiki/>. For the CVS manual (by Per Cederqvist and others), see <http://ximbiot.com/cvs/manual/>.
- For information about Subversion, see <http://subversion.tigris.org/>. For Subversion documentation, see <http://svnbook.red-bean.com/>.

To access the versioning features of SQL Developer, use the **Versioning** menu.

If you create any versioning repositories, you can use the hierarchical display in the Files navigator, which is marked by a folder icon. (If that navigator is not visible, click **View**, then **Files**.) You can also view a hierarchical display of repositories and their contents in the Versioning navigator.

1.10.1 About CVS and SQL Developer

CVS repositories can be created on a local PC or remote server. There can be more than one CVS repository. You need to create one or more CVS repositories.

Source files are held in a CVS repository. The source files in a CVS repository are grouped into modules. If you have new files, a wizard in SQL Developer will help you import them into the CVS repository and place them under version control. A copy is made of your original files and placed in a subdirectory (.backup) of the one from which you imported them.

Files to be worked on are checked out from the CVS repository. This makes a local copy of the files. You can see the contents of the CVS repository in the SQL Developer CVS Navigator and open read-only versions of files. You can then decide which files you want to check out and work on.

CVS creates a new directory populated with the copy of the source files. You can see the files in the System Navigator. You can also open them from here.

Source files have a status, depending on what operations have been carried out on them. A preference lets you choose whether the version control status of a file is shown in the System Navigator.

1.10.1.1 Pending Changes (CVS)

The Pending Changes window is displayed if you click **Versioning**, then **CVS**, 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** pane shows changes made locally, the **Incoming** 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.

1.10.2 About Subversion and SQL Developer

Before you can work with a Subversion repository through SQL Developer, 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 Subversion 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 (or moved into) SQL Developer must be stored in the Subversion working copy.

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

1.11 SQL Developer Reports

SQL Developer provides many reports about the database and its objects. You can also create your own user-defined reports. To display reports, click the Reports tab on the left side of the window (see [SQL Developer User Interface](#)). If this tab is not visible, select **View** and then **Reports**.

Individual reports are displayed in tabbed panes on the right side of the window; and for each report, you can select (in a drop-down control) the database connection for which to display the report. For reports about objects, the objects shown are only those visible to the database user associated with the selected database connection, and the rows are usually ordered by Owner. The detail display pane for a report includes the following icons at the top:

- **Freeze View** (the **pin**) keeps that report in the SQL Developer window when you click another report in the Reports navigator; a separate tab and detail view pane are created for that other report. If you click the pin again, the report's detail view pane is available for reuse.
- **Run Report** updates the detail view pane display by querying the database for the latest information.
- **Run Report in SQL Worksheet** displays the SQL statement used to retrieve the information for a report in a SQL Worksheet pane, where you can view, edit, and run the statement (see [Section 1.7, "Using the SQL Worksheet"](#)).

The time required to display specific reports will vary, and may be affected by the number and complexity of objects involved, and by the speed of the network connection to the database.

For most reports that contain names of database objects, you can double-click the object name in the report display pane (or right-click the object name and select **Go To**) to display that object in a detail view pane, just as if you had selected that object using the Connections navigator.

To export a report into an XML file that can be imported later, right-click the report name in the Reports navigator display and select **Export**. To import a report that had previously been exported, select the name of the report folder name (such as a user-defined folder) in which to store the imported report, right-click, and select **Import**.

You can create a **shared report** from an exported report by clicking **Tools**, then **Preferences**, and using the [Database: User Defined Extensions](#) pane to add a row with Type as **REPORT** and Location specifying the exported XML file. The next time you restart SQL Developer, the Reports navigator will have a Shared Reports folder containing that report.

Reports are grouped in the following categories:

[About Your Database reports](#) list release information about the database associated with the connection.

[All Objects reports](#) list information about all objects accessible to the user associated with the specified database connection, not just objects owned by the user.

[Application Express reports](#) list information about Oracle Application Express 3.0.1 (or later) applications, pages, schemas, UI defaults, and workspaces.

[ASH and AWR reports](#) list information provided by the Active Session History (ASH) and Automated Workload Repository (AWR) features.

[Database Administration reports](#) list usage information about system resources.

[Data Dictionary reports](#) list information about the data dictionary views that are accessible in the database. Examples of data dictionary views are ALL_OBJECTS and USER_TABLES.

[Jobs reports](#) list information about jobs running on the database.

[PL/SQL reports](#) list information about your PL/SQL objects and allow you to search the source of those objects.

[Security reports](#) list privilege-related information about the database.

[Streams reports](#) list information about stream rules.

[Table reports](#) list information about tables owned by the user associated with the specified connection. These reports can help you to better understand the metadata and data. The table reports include [Quality Assurance reports](#) that indicate possible logical design flaws and sources of run-time performance problems.

[XML reports](#) list information about XML objects.

[User Defined reports](#) are any customized reports that you have created.

Bind Variables for Reports

For some reports, you are prompted for **bind variables** before the report is generated. These bind variables enable you to further restrict the output. The default value for all 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, all matches are returned where the value string appears anywhere in the name of the relevant object type.

1.11.1 About Your Database reports

The About Your Database reports list release information about the database associated with the selected connection. The reports include Version Banner (database settings) and National Language Support Parameters (NLS_xxx parameter values for globalization support).

1.11.2 All Objects reports

All Objects reports list information about objects visible to the user associated with the database connection.

All Objects: For each object, lists the owner, name, type (table, view, index, and so on), status (valid or invalid), the date it was created, and the date when the last data definition language (DDL) operation was performed on it. The Last DDL date can help

you to find if any changes to the object definitions have been made on or after a specific time.

Collection Types: Lists information about for each collection type. The information includes the type owner, element type name and owner, and type-dependent specific information.

Dependencies: For each object with references to it, lists information about references to (uses of) that object.

Invalid Objects: Lists all objects that have a status of invalid.

Object Count by Type: For each type of object associated with a specific owner, lists the number of objects. This report might help you to identify users that have created an especially large number of objects, particularly objects of a specific type.

Public Database Links: Lists all public database links.

Public Synonyms: Lists all public synonyms.

1.11.3 Application Express reports

If you select a connection for a schema that owns any Oracle Application Express 3.0.1 (or later) applications, the Application Express reports list information about applications, pages, schemas, UI defaults, and workspaces. For information about Oracle Application Express, see the documentation for that product.

1.11.4 ASH and AWR reports

The ASH and AWR reports list information provided by the Active Session History (ASH) and Automated Workload Repository (AWR) features, which require special licensing. For information about using AWR, including how to use ASH reports, see the information about automatic performance statistics in *Oracle Database Performance Tuning Guide*.

1.11.5 Charts reports

Charts reports include a chart showing the distribution of objects of various object types (number of tables, indexes, and so on).

1.11.6 Database Administration reports

Database Administration reports list usage information about system resources. This information can help you to manage storage, user accounts, and sessions efficiently. (The user for the database connection must have the DBA role to see most Database Administration reports.)

All Tables: Contains the reports that are also grouped under [Table reports](#), including [Quality Assurance reports](#).

Cursors: Provide information about cursors, including cursors by session (including open cursors and cursor details).

Database Parameters: Provide information about all database parameters or only those parameters that are not set to their default values.

Locks: Provide information about locks, including the user associated with each.

Sessions: Provide information about sessions, selected and ordered by various criteria.

Storage: Provide usage and allocation information for tablespaces and data files.

Top SQL: Provide information about SQL statements, selected and ordered by various criteria. This information might help you to identify SQL statements that are being executed more often than expected or that are taking more time than expected.

Users: Provide information about database users, selected and ordered by various criteria. For example, you can find out which users were created most recently, which user accounts have expired, and which users use object types and how many objects each owns.

Waits and Events: Provide information about waits and events, selected by criteria related to time and other factors. For Events in the Last x Minutes, specify the number of minutes in the [Enter Bind Values](#) dialog box.

1.11.7 Data Dictionary reports

Data Dictionary reports list information about the data dictionary views that are accessible in the database. Examples of data dictionary views are ALL_OBJECTS and USER_TABLES.

Dictionary View Columns: For each Oracle data dictionary view, lists information about the columns in the view.

Dictionary Views: Lists each Oracle data dictionary view and (in most cases) a comment describing its contents or purpose.

1.11.8 Jobs reports

Jobs reports list information about jobs running on the database.

All Jobs: Lists information about all jobs running on the database. The information includes the start time of its last run, current run, and next scheduled run.

DBA Jobs: Lists information about each job for which a DBA user is associated with the database connection. The information includes the start time of its last run, current run, and next scheduled run.

Your Jobs: Lists information about each job for which the user associated with the database connection is the log user, privilege user, or schema user. The information includes the start time of its last run, current run, and next scheduled run.

1.11.9 PL/SQL reports

PL/SQL reports list information about PL/SQL packages, function, and procedures, and about types defined in them.

Program Unit Arguments: For each argument (parameter) in a program unit, lists the program unit name, the argument position (1, 2, 3, and so on), the argument name, and whether the argument is input-only (In), output-only (Out), or both input and output (In/Out).

Search Source Code: For each PL/SQL object, lists the source code for each line, and allows the source to be searched for occurrences of the specified variable.

Unit Line Counts: For each PL/SQL object, lists the number of source code lines. This information can help you to identify complex objects (for example, to identify code that may need to be simplified or divided into several objects).

1.11.10 Security reports

Security reports list information about users that have been granted privileges, and in some cases about the users that granted the privileges. This information can help you (or the database administrator if you are not a DBA) to understand possible security issues and vulnerabilities, and to decide on the appropriate action to take (for example, revoking certain privileges from users that do not need those privileges).

Auditing: Lists information about audit policies.

Encryption: Lists information about encrypted columns.

Grants and Privileges: Includes the following reports:

- **Column Privileges:** For each privilege granted on a specific column in a specific table, lists the user that granted the privilege, the user to which the privilege was granted, the table, the privilege, and whether the user to which the privilege was granted can grant that privilege to other users.
- **Object Grants:** For each privilege granted on a specific table, lists the user that granted the privilege, the user to which the privilege was granted, the table, the privilege, and whether the user to which the privilege was granted can grant that privilege to other users.
- **Role Privileges:** For each granted role, lists the user to which the role was granted, the role, whether the role was granted with the ADMIN option, and whether the role is designated as a default role for the user.
- **System Privileges:** For each privilege granted to the user associated with the database connection, lists the privilege and whether it was granted with the ADMIN option.

Policies: Lists information about policies.

Public Grants: Lists information about privileges granted to the PUBLIC role.

1.11.11 Streams reports

Streams reports list information about stream rules.

All Stream Rules: Lists information about all stream rules. The information includes stream type and name, rule set owner and name, rule owner and name, rule set type, streams rule type, and subsetting operation.

Your Stream Rules: Lists information about each stream rule for which the user associated with the database connection is the rule owner or rule set owner. The information includes stream type and name, rule set owner and name, rule owner and name, rule set type, streams rule type, and subsetting operation.

1.11.12 Table reports

Table reports list information about tables owned by the user associated with the specified connection. This information is not specifically designed to identify problem areas; however, depending on your resources and requirements, some of the information might indicate things that you should monitor or address.

For table reports, the owner is the user associated with the database connection.

Columns: For each table, lists each column, its data type, and whether it can contain a null value. Also includes **Datatype Occurrences:** For each table owner, lists each data type and how many times it is used.

Comments for tables and columns: For each table and for each column in each table, lists the descriptive comments (if any) associated with it. Also includes a report of tables without comments. If database developers use the COMMENT statement when creating or modifying tables, this report can provide useful information about the purposes of tables and columns

Constraints: Includes the following reports related to constraints:

- **All Constraints:** For each table, lists each associated constraint, including its type (unique constraint, check constraint, primary key, foreign key) and status (enabled or disabled).
- **Check Constraints:** For each check constraint, lists information that includes the owner, the table name, the constraint name, the constraint status (enabled or disabled), and the constraint specification.
- **Enabled Constraints and Disabled Constraints:** For each constraint with a status of enabled or disabled, lists the table name, constraint name, constraint type (unique constraint, check constraint, primary key, foreign key), and status. A disabled constraint is not enforced when rows are added or modified; to have a disabled constraint enforced, you must edit the table and set the status of the constraint to Enabled (see the appropriate tabs for the [Create/Edit Table \(with advanced options\)](#) dialog box).
- **Foreign Key Constraints:** For each foreign key constraint, lists information that includes the owner, the table name, the constraint name, the column that the constraint is against, the table that the constraint references, and the constraint in the table that is referenced.
- **Primary Key Constraints:** For primary key constraint, lists information that includes the owner, the table name, the constraint name, the constraint status (enabled or disabled), and the column name.
- **Unique Constraints:** For each unique constraint, lists information that includes the owner, the table name, the constraint name, the constraint status (enabled or disabled), and the column name.

Indexes: Includes information about all indexes, indexes by status, indexes by type, and unused indexes.

Organization: Specialized reports list information about partitioned tables, clustered tables, and index-organized tables.

Quality Assurance: (See [Quality Assurance reports](#).)

Statistics: For each table, lists statistical information, including when it was last analyzed, the total number of rows, the average row length, and the table type. In addition, specialized reports order the results by most rows and largest average row length.

Storage: Lists information about the table count by tablespace and the tables in each tablespace.

Triggers: Lists information about all triggers, disabled triggers, and enabled triggers.

User Synonyms: Displays information about either all user synonyms or those user synonyms containing the string that you specify in the Enter Bind Variables dialog box (uncheck Null in that box to enter a string).

User Tables: Displays information about either all tables or those tables containing the string that you specify in the Enter Bind Variables dialog box (uncheck Null in that box to enter a string).

Quality Assurance reports

Quality assurance reports are table reports that identify conditions that are not technically errors, but that usually indicate flaws in the database design. These flaws can result in various problems, such as logic errors and the need for additional application coding to work around the errors, as well as poor performance with queries at run time.

Tables without Primary Keys: Lists tables that do not have a primary key defined. A primary key is a column (or set of columns) that uniquely identifies each row in the table. Although tables are not required to have a primary key, it is strongly recommended that you create or designate a primary key for each table. Primary key columns are indexed, which enhances performance with queries, and they are required to be unique and not null, providing some "automatic" validation of input data. Primary keys can also be used with foreign keys to provide referential integrity.

Tables without Indexes: Lists tables that do not have any indexes. If a column in a table has an index defined on it, queries that use the column are usually much faster and more efficient than if there is no index on the column, especially if there are many rows in the table and many different data values in the column.

Tables with Unindexed Foreign Keys: Lists any foreign keys that do not have an associated index. A foreign key is a column (or set of columns) that references a primary key: that is, each value in the foreign key must match a value in its associated primary key. Foreign key columns are often joined in queries, and an index usually improves performance significantly for queries that use a column. If an unindexed foreign key is used in queries, you may be able to improve run-time performance by creating an index on that foreign key.

1.11.13 XML reports

XML reports list information about XML objects.

XML Schemas: For each user that owns any XML objects, lists information about each object, including the schema URL of the XSD file containing the schema definition.

1.11.14 Migration reports

Migration reports list information related to migrating third-party databases to Oracle. For more information, see [Section 2.13, "Using Migration Reports"](#).

1.11.15 User Defined reports

User Defined reports are any reports that are created by SQL Developer users. To create a user-defined report, right-click the User Defined node under Reports and select **Add Report**. A dialog box is displayed in which you specify the report name and the SQL query to retrieve information for the report (see [Section 5.41, "Create/Edit User Defined Report"](#)).

You can organize user-defined reports in folders, and you can create a hierarchy of folders and subfolders. To create a folder for user-defined reports, right-click the User Defined node or any folder name under that node and select **Add Folder** (see [Section 5.42, "Create/Edit User Defined Report Folder"](#)).

Information about user-defined reports, including any folders for these reports, is stored in a file named UserReports.xml under the directory for user-specific information. For information about the location of this information, see [Section 1.13, "Location of User-Related Information"](#).

For examples of creating user-defined reports, see:

- [Section 1.11.15.1, "User-Defined Report Example: Chart"](#)
- [Section 1.11.15.2, "User-Defined Report Example: Dynamic HTML"](#)
- [Section 1.7.8, "Gauges: In the SQL Worksheet and User-Defined Reports"](#)

1.11.15.1 User-Defined Report Example: Chart

This example creates a report displayed as a chart. It uses the definition of the EMPLOYEES table from the HR schema, which is a supplied sample schema.

Right-click on User Defined Reports and select **Add Report**. In the Add Report dialog box, specify a report name; for **Style**, select **Chart**; and for **SQL**, enter the following:

```
select m.department_id, e.last_name, e.salary
from employees m, employees e
where e.employee_id = m.employee_id
order by 1
```

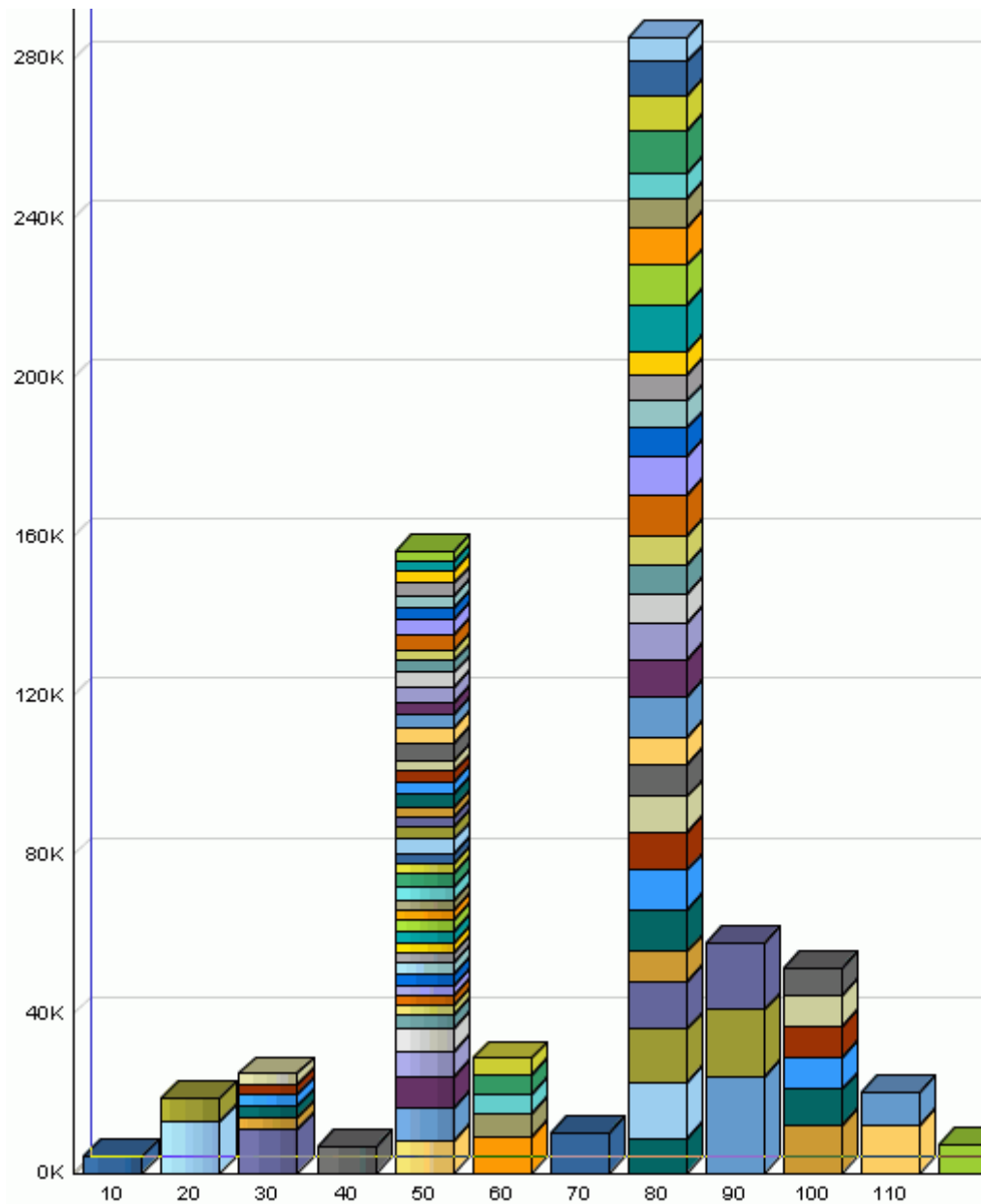
The preceding query lists the last name and salary of each employee in each department, grouping the results by department ID (10, 20, 30, ... 110). Note that the expected syntax for the SQL statement for a chart report is:

```
SELECT <group>, <series>, <value> FROM <table(s)>
```

Click the **Chart Details** tab near the bottom of the box; for **Chart Type**, select BAR_VERT_STACK (bar chart, stacked vertically); and click **Apply**.

Use the Reports navigator to view the newly created user-defined report. For **Connection**, specify one that connects to the HR sample schema.

The report is displayed as a chart, part of which is shown in the following illustration. For example, as you can see, department 50 has mainly employees with the lowest salaries, and department 90 consists of the three highest-paid employees.



1.11.15.2 User-Defined Report Example: Dynamic HTML

This example creates a report using one or more PL/SQL DBMS_OUTPUT statements, so that the report is displayed as dynamic HTML.

Right-click on User Defined Reports and select **Add Report**. In the Add Report dialog box, specify a report name; for **Style**, select **plsql-dbms_output**; and for **SQL**, enter the following:

```
begin
dbms_output.put_line ('<H1> This is Level-1 Heading </H1>');
dbms_output.put_line ('<H2> This is a Level-2 Heading </H2>');
dbms_output.put_line ('<p> This is regular paragraph text. </p>');
end;
```

Click **Apply**.

Use the Reports navigator to view the newly created user-defined report. For **Connection**, specify any from the list. (This report does not depend on a specific connection of table.).

The report is displayed as formatted HTML output.

1.12 SQL Developer Preferences

You can customize many aspects of the SQL Developer interface and environment by modifying SQL Developer preferences according to your preferences and needs. To modify SQL Developer preferences, select **Tools**, then **Preferences**.

Information about SQL Developer preferences is stored under the directory for user-specific information. For information about the location of this information, see [Section 1.13, "Location of User-Related Information"](#).

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.

1.12.1 Environment

The Environment pane contains options that affect the startup and overall behavior and appearance of SQL Developer. 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).

The undo level (number of previous operations that can be undone) and navigation level (number of open files) values involve slight increases or decreases system resource usage for higher or lower values.

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

Environment: Dockable Windows

The Dockable Windows pane configures the behavior of dockable windows and the shapes of the four docking areas of SQL Developer: 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: Local History

The Local History pane controls whether information about editing operations on files opened within SQL Developer is kept. If local history is enabled, you can specify how long information is retained and the maximum number of revisions for each file.

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.

1.12.2 Code Editor

The Code Editor pane contains general options that affect the appearance and behavior of SQL Developer when you edit functions, procedures, and packages.

Autopin PL/SQL Editors: Keeps the current PL/SQL editor open when you open another function, procedure, or package.

Max Open PL/SQL Editors: Specifies the maximum number of PL/SQL editors that can be kept open ("pinned").

Auto-Indent New Lines: Automatically indents a new line when you press Enter at the end of a line. The new line will automatically be indented at the same initial indentation as the line preceding it.

Perform Block Indent or Outdent for Selections: Performs a block indent or block outdent on a selection when you press Tab or Shift+Tab, respectively. With this option selected, when you press Tab on a selected block of text, the entire block will be indented to the current tab size. Shift+Tab on the same block would outdent it, as a block, to the current tab size.

Use Smart Home: Contextualizes the cursor's understanding of home (the beginning of the line). With this setting selected, pressing Home positions the cursor at the start of the line after any leading spaces or tabs. Pressing Home again repositions the cursor at the start of the line before any leading spaces or tabs. Continuing to press Home toggles the cursor between these two locations.

With this setting deselected, pressing Home simply places the cursor at the start of the line.

Use Smart End: Contextualizes the cursor's understanding of end of line. The behavior is analogous to that for Smart Home, except that the cursor responds to the End key, and its behavior regarding the end of the line and any trailing spaces is altered.

Use Jump Scrolling for Keyboard Navigation: Implement jump scrolling, which involves behavior of the keyboard arrow keys. With this setting selected, when you navigate off-screen using the keyboard arrow keys, the editor view will "jump" to recenter the cursor location in the middle of the editor view.

With this setting deselected, the editor view will scroll the editor view the minimum amount to bring the cursor back into view.

Use Change of Case As Word Boundary: Has change of case regarded as the boundary of a word, for example, when you double-click to select a word.

Enable Cut or Copy of Current Line with No Selection: Applies all cut and copy operations to the current line whenever there is no text selection in the editor.

Automatically Copy Paste Imports: Automatically add imports when references are introduced to objects that have not yet been imported.

Adjust Indentation When Pasting: Corrects the indentation of a pasted in item that includes indentation.

Reformat Code Block When Pasting: Reformats the code correctly when you paste it into a new location.

Escape When Pasting in String Literals: Includes the correct escape characters in pasted-in string literals.

Code Editor: Bookmarks

The Bookmarks pane contains options that determine the persistence and search behavior for bookmarks that you create when using the code editor.

Code Editor: Caret Behavior

The Caret Behavior pane contains options that determine the shape, color, and blinking characteristics of the caret (cursor) in the code editor.

Code Editor: Completion Insight

The Completion Insight pane contains options for the logical completion (autocomplete options) of keywords and names while you are coding in the SQL Worksheet.

When you pause for the auto-popup time (if the auto-popup is enabled) or when you press **Ctrl+Space**, code insight provides a context-sensitive popup window that can help you select parameter names. Completion insight provides you with a list of possible completions at the insertion point that you can use to auto-complete code you are editing. This list is based on the code context at the insertion point. To exit code insight at any time, press Esc or continue typing.

You can enable or disable automatic completion and parameter insight, as well as set the time delay for the popup windows.

Generate Column/Table Aliases Automatically: Automatically generates table aliases if you select multiple tables from the popup window; and if you then edit the column list, each column name in the popup window is prefixed with a table alias.

Autogenerate GROUP BY Clause: Automatically generates a GROUP BY clause if you manually enter (not copy/paste) a SELECT statement containing a COUNT function, and then edit the SELECT query.

Code Editor: Display

The Display pane contains general options for the appearance and behavior of the code editor.

Text Anti-Aliasing allows smooth-edged characters where possible.

Code Folding Margin allows program blocks in procedures and functions to be expanded and collapsed in the display.

Visible Right Margin renders a right margin that you can set to control the length of lines of code.

Automatic Brace Matching controls the highlighting of opening parentheses and brackets and of blocks when a closing parenthesis or bracket is typed.

Code Editor: Fonts

The Fonts pane specifies text font options for the code editor.

Display Only Fixed-Width Fonts: If this option is checked, the display of available font names is restricted to fonts where all characters have the same width. (Fixed-width fonts are contrasted with proportional-width fonts.)

Code Editor: Line Gutter

The Line Gutter pane specifies options for the line gutter (left margin of the code editor).

Show Line Numbers: If this option is checked, lines are numbered. (To go to a line number while you are using the SQL Worksheet, press Ctrl+G.)

Enable Line Selection by Click-Dragging: If this option is checked, you can select consecutive lines in the editor by clicking in the gutter and dragging the cursor without releasing the mouse button.

Code Editor: Printing

The Printing pane specifies options for printing the contents of the code editor. The Preview pane sample display changes as you select and deselect options.

Code Editor: Printing HTML

The Printing HTML pane specifies options for printing HTML files from the code editor.

Code Editor: Save Actions

The Save Actions pane specifies actions to be performed automatically during a save operation.

Code Editor: PL/SQL Syntax Colors

The PL/SQL Syntax Colors pane specifies colors for different kinds of syntax elements.

Code Editor: Undo Behavior

The Undo Behavior pane specifies options for the behavior of undo operations (Ctrl+Z, or Edit, then Undo). Only consecutive edits *of the same type* are considered; for example, inserting characters and deleting characters are two different types of operation.

Allow Navigation-Only Changes to be Undoable: If this option is checked, navigation actions with the keyboard or mouse can be undone. If this option is not checked, navigation actions cannot be undone, and only actual changes to the text can be undone.

1.12.3 Compare and Merge

The Compare and Merge pane defines options for comparing and merging two source files. For more information, see [Comparing Source Files](#).

For each type of option, you can specify a **Maximum File Size (KB)**: the maximum size of the file (number of kilobytes) for which the operation will be performed.

Ignore Whitespace: If this option is enabled, leading and trailing tabs and letter spacing are ignored when comparing files. Carriage returns are not ignored. Enabling this option makes comparing two files easier when you have replaced all the space

with hard tabs, or vice versa. Otherwise, every line in the two documents might be shown as different in the Compare window.

Show Character Differences: If this option is enabled, characters that are present in one file and not in another are highlighted. Red highlighting indicates a character that has been removed. Green highlighting indicates a character that has been added. The highlighting is shown only when you click into a comparison block that contains character differences.

Enable XML Compare: If this option is enabled, XML files can be compared.

Enable XML Merge: If this option is enabled, XML files can be merged.

Reformat Result: If this option is enabled, merged XML files can be reformatted.

Validate Result (May require Internet access): If this option is enabled, merged XML files will be validated.

Comparing Source Files

You can compare source files in the following ways:

- A file currently being edited with its saved version: Place the focus on the current version open in the editor, then select the History tab in the editor window. The saved file opens side by side with the file in the editor buffer.
- One file with another file outside the project: Place the focus on the file in the editor to be compared; from the main menu, choose **File**, then **Compare With Other File**; in the Select File to Compare With dialog, navigate to the file and click Open.
- Two files within the same project: In the navigator, select the two files to be compared; then from the main menu, choose **File**, then **Compare With Each Other**.

1.12.4 Database

The Database pane sets properties for the database connection.

Validate date and time default values: If this option is checked, date and time validation is used when you open tables.

Default path to store export in: Default path of the directory or folder under which to store output files when you perform an export operation. To see the current default for your system, click the **Browse** button next to this field.

Filename for connection startup script: File name for the startup script to run when an Oracle database connection is opened. You can click **Browse** to specify the location. The default location is the default path for scripts (see the [Database: Worksheet](#) preferences pane).

Database: Advanced

The Advanced pane specifies options such as the SQL array fetch size and display options for null values.

You can also specify Kerberos thin driver configuration parameters, which enables you to create database connections using Kerberos authentication and specifying the user name and password. For more information, see the Kerberos Authentication explanation on the [Oracle tab](#) in the [Create/Edit/Select Database Connection](#) dialog box. For information about configuring Kerberos authentication, see *Oracle Database Advanced Security Administrator's Guide*.

Use OCI/Thick driver: If this option is checked, and if an OCI (thick, Type 2) driver is available, that driver will be used instead of a JDBC (thin) driver for basic and TNS (network alias) database connections. If any connections use a supported Remote Authentication Dial In User Service (RADIUS) server, check this option.

Kerberos Thin Config: Config File: Kerberos configuration file (for example, `krb5.conf`). If this is not specified, default locations will be tried for your Java and system configuration.

Kerberos Thin Config: Credential Cache File: Kerberos credential cache file (for example, `krb5_cc_cache`). If this is not specified, a cache will not be used, and a principal name and password will be required each time.

Tnsnames Directory: Enter or browse to select the location of the `tnsnames.ora` file. If no location is specified, SQL Developer looks for this file as explained in [Section 1.4, "Database Connections"](#). Thus, any value you specify here overrides any `TNS_ADMIN` environment variable or registry value or (on Linux systems) the global configuration directory.

Database: Autotrace/Explain Plan

The Autotrace/Explain Plan pane specifies information to be displayed on the Autotrace and Explain Plan panes in the SQL Worksheet.

Database: Drag and Drop

The Drag and Drop Effects pane determines the type of SQL statement created in the SQL Worksheet when you drag an object from the Connections navigator into the SQL Worksheet. The SQL Developer preference sets the default, which you can override in the Drag and Drop Effects dialog box.

The type of statement (`INSERT`, `DELETE`, `UPDATE`, or `SELECT`) applies only for object types for which such a statement is possible. For example, `SELECT` makes sense for a table, but not for a trigger. For objects for which the statement type does not apply, the object name is inserted in the SQL Worksheet.

Database: NLS

The NLS pane specifies values for globalization support parameters, such as the language, territory, sort preference, and date format. These parameter values are used for SQL Developer session operations, such as for statements executed using the SQL Worksheet and for the National Language Support Parameters report. Specifying values in this preferences pane does *not* apply those values to the underlying database itself. To change the database settings, you must change the appropriate initialization parameters and restart the database.

Note that SQL Developer does not use default values from the current system for globalization support parameters; instead, SQL Developer, when initially installed, by default uses parameter values that include the following:

```
NLS_LANG, "AMERICAN"
NLS_TERR, "AMERICA"
NLS_CHAR, "AL32UTF8"
NLS_SORT, "BINARY"
NLS_CAL, "GREGORIAN"
NLS_DATE_LANG, "AMERICAN"
NLS_DATE_FORM, "DD-MON-RR"
```

Database: ObjectViewer Parameters

The ObjectViewer Parameters pane specifies whether to freeze object viewer windows, and display options for the output. The display options will affect the generated DDL on the SQL tab. The Data Editor Options affect the behavior when you are using the Data tab to edit table data.

Data Editor Options

Post Edits on Row Change: If this option is checked, posts DML changes when you perform edits using the Data tab (and the Set Auto Commit On option determines whether or not the changes are automatically committed). If this option is not checked, changes are posted and committed when you press the Commit toolbar button.

Set Auto Commit On (available only if Post Edit on Row Changes is enabled): If this option is checked, DML changes are automatically posted and committed when you perform edits using the Data tab.

Clear persisted table column widths, order, sort, and filter settings: If you click **Clear**, then any customizations in the Data tab display for table column widths, order, sort, and filtering are not saved for subsequent openings of the tab, but instead the default settings are used for subsequent openings.

Database: PL/SQL Compiler

The PL/SQL Compiler pane specifies options for compilation of PL/SQL subprograms.

Generate PL/SQL Debug Information: If this option is checked, PL/SQL debug information is included in the compiled code; if this option is not checked, this debug information is not included. The ability to stop on individual code lines and debugger access to variables are allowed only in code compiled with debug information generated.

Types of messages: You can control the display of informational, severe, and performance-related messages. (The ALL type overrides any individual specifications for the other types of messages.) For each type of message, you can specify any of the following:

- **No entry (blank):** Use any value specified for ALL; and if none is specified, use the Oracle default.
- **Enable:** Enable the display of all messages of this category.
- **Disable:** Disable the display of all messages of this category.
- **Error:** Enable the display of only error messages of this category.

Optimization Level: 0, 1, or 2, reflecting the optimization level that will be used to compile PL/SQL library units. The higher the setting of this parameter, the more effort the compiler makes to optimize PL/SQL library units. However, for a module to be compiled with PL/SQL debugging information, the level must be 0 or 1.

PLScope Identifiers: Specifies the amount of PL/Scope identifier data to collect and use (All or None).

Database: Reports

The Reports pane specifies options relating to [SQL Developer Reports](#).

Close all reports on disconnect: If this option is checked, all reports for any database connection are automatically closed when that connection is disconnected.

Database: SQL Editor Code Templates

The SQL Editor Code Templates pane enables you to view, add, and remove templates for editing SQL and PL/SQL code. Code templates assist you in writing code more quickly and efficiently by inserting text for commonly used statements. You can then modify the inserted text.

The template ID string is not used by SQL Developer; only the template content (Description text) is used, in that it is considered by completion insight (explained in [Code Editor: Completion Insight](#)) in determining whether a completion popup should be displayed and what the popup should contain. For example, if you define code template ID *mydate* as *SELECT sysdate FROM dual*, then if you start typing *select* in the SQL Worksheet, the auto-popup includes *SELECT sysdate FROM dual*.

Add Template: Adds an empty row in the code template display. Enter an ID value, then move to the Template cell; you can enter template content in that cell, or click the ellipsis (...) button to open the code editor to enter the template content.

Remove Template: Deletes the selected code template.

Database: SQL Formatter

The SQL Formatter pane controls how statements in the SQL Worksheet are formatted when you click Format SQL. 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 (if there is room) or on separate lines.

Import: Lets you import code style profile settings that you previously exported.

Export: Exports the current code profile settings to an XML file.

Autoformat PL/SQL in Procedures, Packages, Views, and Triggers: If this option is checked, the SQL Formatter options are applied automatically as you enter and modify PL/SQL code in procedures, packages, views, and triggers; if this option is not checked, the SQL Formatter options are applied only when you so request.

Panes for product-specific formatting options: Individual panes let you specify formatting options for Oracle and for other vendors (Microsoft Access, IBM DB2, Microsoft SQL Server, Sybase Adaptive Server). In each of these panes, you can click **Edit** to specify input/output, alignment, indentation, line breaks, CASE line breaks, white space, and other options.

Database: Third Party JDBC Drivers

The Third Party JDBC Drivers pane specifies drivers to be used for connections to third-party (non-Oracle) databases, such as IBM DB2, MySQL, Microsoft SQL Server, or Sybase Adaptive Server. (You do not need to add a driver for connections to Microsoft Access databases.) To add a driver, click **Add Entry** and select the path for the driver:

- For IBM DB2: the `db2jcc.jar` and `db2jcc_license_cu.jar` files, which are available from IBM
- For MySQL: a file with a name similar to `mysql-connector-java-5.0.4-bin.jar`, in a directory under the one into which you unzipped the download for the MySQL driver
- For Microsoft SQL Server or Sybase Adaptive Server: `jtds-1.2.jar`, which is included in the `jtds-1.2-dist.zip` download

- For Teradata: `tdgssconfig.jar` and `terajdbc4.jar`, which are included (along with a `readme.txt` file) in the `TeraJDBC__indep_indep.12.00.00.110.zip` or `TeraJDBC__indep_indep.12.00.00.110.tar` download

Alternative: As an alternative to using this preference, you can click **Help**, then **Check for Updates** to install the JTDS JDBC Driver for Microsoft SQL Server and the MySQL JDBE Driver as extensions.

To find a specific third-party JDBC driver, see the appropriate Web site (for example, <http://www.mysql.com> for the MySQL Connector/J JDBC driver for MySQL, <http://jtds.sourceforge.net/> for the jTDS driver for Microsoft SQL Server and Sybase Adaptive Server, or <http://www.teradata.com/DownloadCenter/Forum54-1.aspx> for the JDBC driver for Teradata). For MySQL, use the MySQL 5.0 driver, not 5.1 or later, with SQL Developer release 1.5.

You must specify a third-party JDBC driver or install a driver using the Check for Updates feature before you can create a database connection to a third-party database of that associated type. (See the tabs for creating connections to third-party databases in the [Create/Edit/Select Database Connection](#) dialog box.)

Database: User Defined Extensions

The User Defined Extensions pane specifies user-defined extensions that have been added. You can use this pane to add extensions that are not available through the Check for Updates feature. These extensions can be for user-defined reports, actions, editors, and navigators. (For more information about extensions and checking for updates, see [Section 1.12.6, "Extensions"](#).)

One use of the Database: User-Defined Extensions pane is to create a Shared Reports folder and to include an exported report under that folder: click Add Row, specify Type as REPORT, and for Location specify the XML file containing the exported report. The next time you restart SQL Developer, the Reports navigator will have a Shared Reports folder containing that report.

For more information about creating user-defined extensions, see:

- *How To create an XML User Defined Extension:*
<http://wiki.oracle.com/page/SQL+Dev+SDK+How+To+create+an+XML+User+Defined+Extension>
- *Creating XML Extensions for Oracle SQL Developer:*
http://www.oracle.com/technology/obe/sqldev_obe/extension/extensions.htm

Database: Worksheet

The Worksheet pane specifies options that affect the behavior of the SQL Worksheet.

Autocommit in SQL Worksheet: If this option is checked, a commit operation is automatically performed after each INSERT, UPDATE, or DELETE statement executed using the SQL Worksheet. If this option is not checked, a commit operation is not performed until you execute a COMMIT statement.

Open a worksheet on connect: If this option is checked, a SQL Worksheet window for the connection is automatically opened when you open a database connection. If this option is not checked, you must use the Open SQL Worksheet right-click command or toolbar icon to open a SQL Worksheet.

Close all worksheets on disconnect: If this option is checked, all SQL Worksheet windows for any database connection are automatically closed when that connection is disconnected.

Prompt for Save File on Close: If this option is checked, you are prompted to save changes when you close a SQL Worksheet if it contains any unsaved changes. If this option is not checked, any unsaved changes are discarded.

Max rows to print in a script: Limits the number of rows displayed.

SQL History Limit: Maximum number of statements that can be stored in the [SQL History](#). Must be greater than 0 (zero). If you enter an invalid value, no value is stored in this field.

Default path to look for scripts: The default directory where SQL Developer looks when you run a script (using @). If you type a path, you can specify multiple delimited locations; if you click Browse, you can select a single location. In addition to any path that you specify, SQL Developer looks in the location specified by the SQLPATH environment variable.

Save bind variables to disk on exit: If this option is checked, bind variables that you enter when running a script are saved on disk for reuse. If you do not want bind variable values stored on disk (for security or other reasons), be sure not to check this option.

1.12.5 Debugger

The Debugger pane contains general options for the SQL Developer debugger. Other panes contain additional specific kinds of debugger options.

Debugger: Breakpoints

The Breakpoints pane sets the columns to appear in the Breakpoints pane and the scope of each breakpoint.

Debugger: Breakpoints: Default Actions

The Breakpoints: Default Actions pane sets defaults for actions to occur at breakpoints. These actions are the same as on the [Actions tab](#) in the [Create/Edit Breakpoint](#) dialog box.

Debugger: Data

The Data pane enables you to control the columns to appear in the debugger Data pane and aspects of how the data is displayed.

Debugger: Inspector

The Inspector pane enables you to control the columns to appear in the debugger Inspector pane and aspects of how the data is displayed.

Debugger: Smart Data

The Smart Data pane enables you to control the columns to appear in the debugger Smart Data pane and aspects of how the data is displayed.

Debugger: Stack

The Stack pane enables you to control the columns to appear in the debugger Stack pane and other options.

Debugger: ToolTip

The ToolTip pane enables you to control the columns to appear in the debugger ToolTip pane.

Debugger: Watches

The Watches pane enables you to control the columns to appear in the debugger Watches pane and aspects of how the data is displayed.

1.12.6 Extensions

The Extensions pane determines which optional extensions SQL Developer uses when it starts. (SQL Developer also uses some mandatory extensions, which users cannot remove or disable.) If you change any settings, you must exit SQL Developer and restart it for the new settings to take effect.

For Versioning Support, the settings (selected or not, and configuration options if selected) affect whether the Versioning menu is displayed and the items on that menu.

Extensions to Use: Controls the specific optional SQL Developer extensions to use at startup.

Check for Updates: Checks for any updates to the selected optional SQL Developer extensions, as well as any mandatory extensions. (If the system you are using is behind a firewall, see the SQL Developer user preferences for [Web Browser and Proxy](#).)

Automatically Check for Updates: If this option is checked, SQL Developer automatically checks for any updates to the selected optional SQL Developer extensions and any mandatory extensions at startup.

1.12.7 External Editor

The External Editor pane determines which external editor is called by SQL Developer when you try to edit binary large object (BLOB) data, such as image files, video files, and other files created by certain applications. For each combination of MIME type and file extension, you can specify the executable application to be used to open and edit associated files.

MIME Type: MIME type of the data.

File Extension: File extension for files that contain BLOB data and that are associated with the MIME type.associated

Editor Location: Path to the editor to be used to open and edit files associated with this MIME type and file extension. To edit an existing path or to specify one if the cell is empty, click in the cell, and either modify the existing text or click Browse to find and select the executable file for the editor.

1.12.8 File Types

The File Types pane determines which file types and extensions will be opened by default by SQL Developer. The display shows each file extension, the associated file type, and a check mark if files with that extension are to be opened by SQL Developer by default, such as when a user double-clicks the file name.

Details area at bottom: You can modify the file type, content type (text or binary), and whether to open files with this extension automatically by SQL Developer.

To have files with a specific extension be opened by default by SQL Developer, click the file extension in the list, then check Open with SQL Developer in the Details area.

This overrides any previous application association that may have been in effect for that file extension.

To add a file extension, click **Add** and specify the file extension (including the period). After adding the extension, you can modify its associated information by selecting it and using the Details area.

1.12.9 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 SQL Developer 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 SQL Developer 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 SQL Developer; but if it is disabled (unchecked), the filter is not enforced.

1.12.10 Migration

The Migration pane contains options that affect the behavior of SQL Developer when you migrate schema objects and data from third-party databases to an Oracle database.

Default Repository: Migration repository to be used for storing the captured models and converted models. For information about migrating third-party databases to Oracle, including how to create a migration repository, see [Chapter 2](#).

Migration: Data Move Options

The Data Move Options pane contains options that affect the behavior when you migrate data from third-party databases to Oracle Database tables generated by the migration.

Online for all. Offline for MySQL, SQL Server, and Sybase Adaptive Server: Options that can be used for online data migration for all supported third-party databases, and for offline data migration for MySQL, SQL Server, and Sybase Adaptive Server.

Representation for 0 Length String: The value to which Oracle converts zero-length strings in the source data. Can be a space (' ') or a null value (NULL). Specific notes:

- For Microsoft Access offline migrations, a null value and a space are considered the same.
- For Sybase offline migrations, " is considered the same as a space (' ').
- For MySQL offline migrations, a null value is exported as 'NULL', which is handled as type VARCHAR2. You can specify another escape character by using the `--fields-escaped-by` option with the `mysqldump` command (for example,

specifying \N for null or \\ for \). For information about the mysqldump command, see [Section 2.9.1.3, "Creating Data Files From MySQL"](#).

For MySQL offline migrations, the data is exported to a file named *table-name.txt*; so if you are moving data from two or more tables with the same name but in different schemas, rename files as needed so that they are all unique, and modify the SQL*Loader .ctl file accordingly.

Online: The online data move options determine the results of files created when you click **Migration**, then **Migrate Data**.

Number of Parallel Data Move Streams: The number of internal connections created for simultaneous movement of data from the source database to the Oracle tables. Higher values may shorten the total time required, but will use more database resources during that time.

Number of Rows to Commit After: During the data move operation, Oracle pauses to perform an automatic internal commit operation after each number of rows that you specify are moved from the source database to Oracle tables.

Lower values will cause a successful move operation to take more time; but if a failure occurs, it is likely that more source records will exist in the Oracle tables and that if the move operation is resumed, fewer source records will need to be moved. Higher values will cause a successful move operation to take less time; but if a failure occurs, it is likely that fewer source records will exist in the Oracle tables and that if the move operation is resumed, more source records will need to be moved.

Offline: The offline data move options determine the results of files created when you click **Migration**, then **Generate Scripts**, then **Generate Data Move Scripts**.

Offline Data Script Directory: Default location for scripts for offline data move operations.

End of Column Delimiter: String to indicate end of column.

End of Row Delimiter: String to indicate end of row.

Date Mask: Format mask for dates.

Timestamp Mask: Format mask for timestamps.

Migration: Generation Options

The Generation Options pane contains options that determine the results of files created when you click **Migration**, then **Generate Scripts**, then **Generate Oracle DDL**.

One single file or A file per object: Determines how many files are created and their relative sizes. Having more files created might be less convenient, but may allow more flexibility with complex migration scenarios. (See also the Maximum Number of Lines option.)

Output Directory: Default location in which the files will be created.

Implement 'CREATE' as 'CREATE OR REPLACE': Causes CREATE statements in source database objects to be implemented using CREATE OR REPLACE when the Oracle syntax allows this.

Generate Comments: Generates comments in the Oracle SQL statements.

Generate Controlling Script: Generates a "master" script for running all the required files.

Maximum Number of Lines: Sets a maximum number of lines for each file; you then specify the number.

Least Privilege Schema Migration: For migrating schema objects in a converted model to Oracle, causes CREATE USER, GRANT, and CONNECT statements *not* to be generated in the output scripts. You must then ensure that the scripts are run using a connection with sufficient privileges. You can select this option if the database user and connection that you want to use to run the scripts already exist, or if you plan to create them.

Generate Data Move User: For data move operations, creates an additional database user with extra privileges to perform the operation. It is recommended that you delete this user after the operation. This option is provided for convenience, and is suggested unless you want to perform least privilege migrations or unless you want to grant privileges manually to a user for the data move operations. This option is especially recommended for multischema migrations, such as when not all tables belong to a single user.

Generate Failed Objects: Causes objects that failed to be converted to be included in the generation script, so that you can make any desired changes and then run the script. If this option is not checked, objects that failed to be converted are not included in the generation script.

Generate Stored Procedure for Migrate Blobs Offline: Causes a stored procedure named CLOBtoBLOB_sqldeveloper (with execute access granted to public) to be created if the schema contains a BLOB (binary large object); this procedure is automatically called if you perform an offline capture. If this option is not checked, you will need to use the manual workaround described in [Section 2.9.1.4, "Populating the Destination Database Using the Data Files"](#). (After the offline capture, you can delete the CLOBtoBLOB_sqldeveloper procedure or remove execute access from public.)

Migration: Identifier Options

The Identifier Options pane contains options that apply to object identifiers during migrations.

Prepended to All Identifier Names (Microsoft Access, Microsoft SQL Server, and Sybase Adaptive Server migrations only): A string to be added at the beginning of the name of migrated objects. For example, if you specify the string as XYZ_, and if a source table is named EMPLOYEES, the migrated table will be named XYZ_EMPLOYEES. (Be aware of any object name length restrictions if you use this option.)

Is Quoted Identifier On (Microsoft SQL Server and Sybase Adaptive Server migrations only): If this option is enabled, quotation marks (double-quotes) can be used to refer to identifiers (for example, SELECT "Col 1" from "Table 1"); if this option is not enabled, quotation marks identify string literals. **Important:** The setting of this option must match the setting in the source database to be migrated, as explained in [Section 2.5.2, "Before Migrating From Microsoft SQL Server or Sybase Adaptive Server"](#).

Migration: Translators

The Translators pane contains options that relate to conversion of stored procedures and functions from their source database format to Oracle format. (These options apply only to migrations from Microsoft Access, Microsoft SQL Server, and Sybase Adaptive Server.)

Default Source Date Format: Default date format mask to be used when casting string literals to dates in stored procedures and functions.

Query Assignment Translation: Option to determine what is generated for a query assignment: only the assignment, assignment with exception handling logic, or

assignment using a cursor LOOP ... END LOOP structure to fetch each row of the query into variables.

1.12.11 Mouseover Popups

The Mouseover Popups 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 during a debugging operation), **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.

1.12.12 Shortcut Keys (Accelerator Keys)

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

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: Select All or a specific category (Code Editor, Database, Debug, Edit, and so on), to control which actions are displayed.

Command: The actions for the selected 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.

1.12.13 Unit Test Parameters

Unit Test Parameters preferences affect the behavior of the SQL Developer unit testing feature (described in [Chapter 3](#)).

Configuration set to use for lookups: Lookup category to be used for automatically generating test implementations when you create a unit test, as explained in [Section 3.6.2](#). The list includes the default category and any user-added categories (see the [Unit Testing: Add Category](#) dialog box).

1.12.14 Versioning

Versioning preferences affect the behavior of the version control and management systems that you can use with SQL Developer. You can specify preferences for CVS and Subversion. For information about using versioning with SQL Developer, see [Section 1.10, "Using Versioning"](#).

Versioning: CVS

The CVS pane specifies options for use with CVS (Concurrent Versions System).

CVS Client: Internal to Oracle SQL Developer (installed with SQL Developer) or **External Executable** (separately installed CVS client, for which you must specify the name or path).

- **Name on System Path:** Name of the CVS server executable. The default (cvs) is correct for most installations. This option assumes that the name of the CVS server executable is on the system path.
- **Path from Environment:** Location of the CVS server executable, especially if there is more than one on the system path. The selection area will list all instances of the CVS server executable known to the local system. You may have more than one version of CVS installed: this option lets you specify which of them to use with SQL Developer.
- **Other Path:** Location of the CVS server executable, if it is not on the system path at all.

Run CVS in Edit/Watch Mode: If this option is enabled, you coordinate access to files by declaring an editor for them through CVS, after which they may be modified. Only those files that you check out after changing this preference will be affected. If this option is disabled, the edit and watch commands on the Versioning menu are disabled.

State Overlay Scheme: Scheme for the icons displayed alongside folder and file names in the navigators to indicate their versioning status.

Versioning: CVS: Commands

The CVS: Commands pane sets options for CVS source control. Some options are not available when using the internal CVS client.

Enable Advanced Controls: If this option is enabled, advanced CVS controls are shown in dialog boxes. If you find that you use only basic CVS features, you might wish to use SQL Developer without advanced controls, to reduce complexity and save screen space.

Global Options: Run Quietly: If this option is enabled, informational messages are suppressed.

Global Options: Do not Log Commands: If this option is enabled, CVS commands are not logged in the repository command history.

Global Options: Encrypt: If this option is enabled, all communication between the client and the server is encrypted. Encryption support is not available in CVS by default; it must be enabled using a special configuration option when you build CVS.

Set Compression Level (z): If this option is enabled, you can set the compression level for files sent between client and server. The level can be set from Minimum (high speed, low compression) to Maximum (low speed, high compression).

Keyword Substitution Mode: CVS uses keyword substitution modes to insert revision information into files when they are checked out or updated. This option controls the mode of replacement for keyword substitution in versioned files:

- **Automatic:** The default, recommended option.
- **Keyword-Only Mode:** Generates only keyword names in keyword strings and omits their values. This option is useful for disregarding differences due to keyword substitution when comparing different revisions of a file.
- **Keyword-Value Mode:** Generates keyword strings using the default form.
- **Keyword-Value-Locker Mode:** Like the keyword-value mode, except that the name of the locker is always inserted if the given revision is currently locked.
- **Old-Contents Mode:** Generates the old keyword string, present in the working file just before it was checked in.
- **Value-Only Mode:** Generates only keyword values for keyword strings. This can help generate files in programming languages where it is hard to strip keyword delimiters from a string. However, further keyword substitution cannot be performed once the keyword names are removed, so this option should be used with care.

On Commit: Use Comment Templates: If this option is enabled, your commit comments will be entered through template forms. The forms are set up by the CVS system administrator. There may be different forms for different circumstances and installations, and it may be that none of them are suitable for your commit comments. In this case, this preference lets you disable the use of all forms.

On Commit: Automatically Add Files: If this option is enabled, local files are added to the CVS repository whenever you perform a commit action.

Create Backup Files on Remove: If this option is enabled, backup copies are made of files that are removed through actions of the source control system.

Versioning: CVS: General

The CVS: 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 Make Files Editable: If this option is enabled, an editor is automatically used on a data file when you start to change it. (If you edit a file unintentionally, immediately use Versioning, then Unedit to revert.)

Operation Timeout: Maximum time allowed for CVS operations to complete.

Versioning: CVS: Navigator Labels

The CVS: Navigator Labels pane specifies formatting for CVS information appears on navigator nodes and tool tips. For a full explanation of keyword substitution modes, see the CVS documentation.

Versioning: CVS: Version Tools

The CVS: 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.

Versioning: Subversion

The Subversion pane specifies the Subversion client to use with SQL Developer.

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 Make Files Editable: If this option is enabled, an editor is automatically used on a data file when you start to change it. (If you edit a file unintentionally, immediately use Versioning, then Unedit to revert.)

Operation Timeout: Maximum time allowed for Subversion operations to complete.

Edit Subversion Configuration File: To modify the Subversion file directly, click **Edit "server"**.

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.

1.12.15 Web Browser and Proxy

The Web Browser and Proxy pane 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.

Browser Command Line: To specify a Web browser other than your default browser, specify the executable file to start that browser. To use your default browser, leave this field blank.

Use HTTP Proxy Server: Check your Web browser options or preferences for the appropriate values for these fields.

1.13 Location of User-Related Information

SQL Developer stores user-related information in several places, with the specific location depending on the operating system and certain environment specifications. User-related information includes user-defined reports, user-defined snippets, SQL Worksheet history, code templates, and SQL Developer user preferences. In most cases, your user-related information is stored outside the SQL Developer installation directory hierarchy, so that it is preserved if you delete that directory and install a new version.

The user-related information is stored in or under the IDE_USER_DIR environment variable location, if defined; otherwise as indicated in the following table, which shows the typical default locations (under a directory or in a file) for specific types of resources on different operating systems. (Note the period in the name of any directory named `.sqldeveloper`.)

The following table shows the typical default locations (under a directory or in a file) for specific types of resources on different operating systems. (Note the period in the name of any directory named `.sqldeveloper`.)

Table 1–1 Default Locations for User-Related Information

Resource Type	System (Windows, Linux, or Mac OS X)
User-defined reports	Windows: C:\Documents and Settings\ <i><user-name></i> \Application Data\SQL Developer\UserReports.xml Linux or Mac OS X: ~/.sqldeveloper/UserReports.xml

Table 1–1 (Cont.) Default Locations for User-Related Information

Resource Type	System (Windows, Linux, or Mac OS X)
User-defined snippets	Windows: C:\Documents and Settings\ <user-name>\Application Data\SQL Developer\UserSnippets.xml Linux: ~/.sqldeveloper/UserSnippets.xml Mac OS X: /Users/<Your user>/Library/Application Support/SQLDeveloper/UserSnippets.xml</user-name>
SQL history	Windows: C:\Documents and Settings\ <user-name>\Application Data\SQL Developer\SqlHistory.xml Linux: ~/.sqldeveloper/SqlHistory.xml Mac OS X: /Users/<Your user>/Library/Application Support/SQLDeveloper/SqlHistory.xml</user-name>
Code templates	Windows: C:\Documents and Settings\ <user-name>\Application Data\SQL Developer\CodeTemplate.xml Linux: ~/.sqldeveloper/CodeTemplate.xml Mac OS X: /Users/<Your user>/Library/Application Support/SQLDeveloper/CodeTemplate.xml</user-name>
SQL Developer user preferences	Windows: C:\Documents and Settings\ <user-name>\Application Data\SQL Developer\systemn.n.n.n.n Linux or Mac OS X: ~/.sqldeveloper/systemn.n.n.n.n</user-name>

If you want to prevent other users from accessing your user-specific SQL Developer information, you must ensure that the appropriate permissions are set on the directory where that information is stored or on a directory above it in the path hierarchy. For example, on a Windows system you may want to ensure that the SQL Developer folder and the `\<user-name>\Application Data\SQL Developer` folder under Documents and Settings are not shareable; and on a Linux or Mac OS X system you may want to ensure that the `~/.sqldeveloper` directory is not world-readable.

1.14 Data Modeler Viewer (Read-Only)

SQL Developer includes a read-only (viewer) version of SQL Developer Data Modeler. The Data Modeler viewer enables you to open, import, and view a database design, and to save it separate from the original design. However, you cannot create, modify, or delete any Data Modeler objects.

To display the Data Modeler viewer in a pane, click **Tools**, then **Data Modeler**. After that, the Data Modeler menu under Tools includes additional commands, for example, enabling you to specify design rules and general options (user preferences).

The online help for the Data Modeler viewer is the same as the help for the full-featured Data Modeler; thus, the help includes information about features that are not supported in the viewer.

1.15 Oracle TimesTen In-Memory Database Support

When you connect to an Oracle TimesTen In-Memory Database, the available types of objects that you can work with include several that apply to an Oracle Database, and the following that are specific to TimesTen:

- Cache groups
- Replication schemes

To create a connection to a TimesTen database, use the **TimesTen tab** in the **Create/Edit/Select Database Connection** dialog box.

For additional usage and reference information, see the following:

- *Oracle SQL Developer TimesTen In-Memory Database Support User's Guide* for information about using TimesTen-specific features in SQL Developer
- http://www.oracle.com/technology/products/timesten/timesten_sqldeveloper.html for an overview SQL Developer support for Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache, and links to related information
- <http://www.oracle.com/technology/products/timesten/> for links to TimesTen documentation, white papers, tutorials, case studies, and other resources

1.16 Using the Help

SQL Developer provides a Help menu and context-sensitive help (click the Help button or press the F1 key in certain contexts). Much of the help content is also in *Oracle SQL Developer User's Guide*, which is in the SQL Developer Documentation Library.

Help is displayed in the Help Center window, which has a Contents pane on the left, a Search box at the top right, and a help topic display pane under the Search box. You can move the horizontal divider to change the pane sizes (for example, to make the Contents pane narrower, to allow more room for the help topic content). You can also resize and reposition the Help Center window.

For Search, you can click the icon (binoculars) to see search options: case sensitive (Match case) or case insensitive; and whether to match topics based on all specified words, any specified words, or a Boolean expression.

The **Keep on Top** button toggles whether the Help Center window is kept on top of the display when you switch focus (click) back in the SQL Developer window.

To print a help topic, display it in the topic display pane and click the Print icon at the top of the pane.

To increase or decrease the size of the font in the help topic viewer, click the Change Font Size (A) icon in the Help Center topic display area toolbar, then select Increase Font Size or Decrease Font Size. This setting is preserved only for the duration of the current help pane or window; therefore, you may want to keep the Help Center window open after setting the help text font to your preferred size.

1.17 Tip of the Day

For English locales, you can display a random suggestion for effective use of SQL Developer by clicking **Help**, then **Tip of the Day**. The tip window is also displayed automatically when you start SQL Developer, unless you disable the **Show tips at startup** option in the tip window.

For convenience, this section lists the available tip topics. (There is no special order or organization for these topics.)

1.17.1 SQL History Shortcuts

Using **Ctrl+up-arrow** or **Ctrl+down-arrow** in the SQL Worksheet replaces the contents of the SQL Worksheet with lines of code from the SQL History. You can step up and down through the SQL History.

To view the SQL History in the SQL Worksheet, press **F8** or click **View**, then **History**.

1.17.2 Unshared Worksheets

To create a separate unshared worksheet for a connection, press **Ctrl+Shift+N**.

1.17.3 SQL Worksheet Bookmarks

If you have many SQL Worksheets open, you can assign a bookmark number to each and then easily navigate among them. To create a bookmark, click the worksheet's tab and press **Alt+Shift+number** (for example, **Alt+Shift+1**). The number now appears as a small superscript in the tab.

To switch to a worksheet that has a bookmark, press **Alt+number** (for example, **Alt+1**).

1.17.4 Formatted Display of SQL Trace (.trc) Files

To see a formatted display of a SQL Trace file, drag the *.trc file onto the area above the SQL Worksheet (or open it by clicking **File**, then **Open**).

1.17.5 Folders for Organizing Connections

You can group connections into folders. Right-click a connection name and select **Add to Folder**. See the help topic [Using Folders to Group Connections](#).

1.17.6 Third-Party Databases and SQL Developer

In addition to Oracle databases, SQL Developer works with several third-party databases, such as MySQL, Sybase, Microsoft Access, Microsoft SQL Server, and IBM DB2.

For information about connecting to third-party databases, or about migrating a third-party database to Oracle, see the help topics [Database Connections](#) and [Migrating Third-Party Databases](#).

1.17.7 Debugger Ports and Firewalls

The SQL Debugger by default uses ports 40000 to 49000. If you cannot get the debugger to start, make sure that you are not being blocked by a firewall on these ports.

1.17.8 Viewing Multiple Tables

You can have tabs open for more than one table. Just click the Freeze View button (it looks like a push pin) when you are viewing a table; and when you click to display another table, the tab for the first table will remain open.

1.17.9 Customizing SQL Developer Appearance

You can use the **Look and Feel** (platform) and **Theme** (color scheme) options under Environment preferences to customize the appearance of the SQL Developer window.

1.17.10 Maximizing Tab Panes

You can often maximize a display pane (such as a SQL Worksheet) by double-clicking its tab.

To restore the SQL Developer window to its original display, double-click the tab again.

1.17.11 Default Path for Running Scripts

You can set a default path for SQL Developer to use if you run a SQL script file (for example, *@my_script.sql*) without specifying the path. See the Database: Worksheet Parameters preferences.

1.17.12 Shutting Down and Restarting the Database

A user with SYSDBA privileges can shut down and restart the database from within SQL Developer, if a listener is running with a static listener configured for the database. Right-click the connection name and select **Manage Database**.

1.17.13 Feature Requests

Do you have a SQL Developer feature request? Log it at the Oracle Technology Network: go to <http://sqldeveloper.oracle.com> and select the **Feature Requests** link.

1.17.14 Discussion Forum

Would you like to share and search information, questions, and comments about SQL Developer? Visit our discussion forum at:

<http://forums.oracle.com/forums/forum.jspa?forumID=260>

1.17.15 Help Text Font Size

You can use a button in the Help Center window display pane (right side) to change the help text display size: click **Increase Font Size** or **Decrease Font Size** (repeatedly if necessary) until the size is right for you. (If the text appears blurry, try decreasing the size.)

Suggestion: Don't close the Help Center window, because font size changes will be in effect only for as long as the current window is open. Consider using the **Keep on Top** toggle in the Help Center window.

1.17.16 Procedure and Function Signatures

To see the signature (format, including parameters) of a procedure or function in a PL/SQL package, expand the package (under Packages in the Connections navigator), and place the mouse pointer over the procedure or function name.

1.17.17 Type-Ahead in Navigators

Many navigators that use a tree support type-ahead to find and open an object. For example, expand the Tables node under a connection and start typing a table name.

Note: This works only on nodes when the child nodes are visible. For example, if the Tables node is not expanded to display the individual tables, typing the name of a table will not find and open it.

1.17.18 Extended Paste

If you have cut or copied multiple things to the clipboard and want to paste something other than the most recent copy, you can use extended paste to display a dialog box to select which one to paste. Press **Ctrl+Shift+V**; or click **Edit**, then **Extended Paste**.

1.17.19 Closing Tabbed Windows Using the Mouse Wheel

To close a a tabbed editor or display window, click its tab with the mouse wheel.

1.17.20 Go to Last Edit Location

If you have made edits in several editing windows and are now in a different window, and if you want to return to where you made the last edit, press **Ctrl+Shift+Backspace**; or click **Navigate**, then **Go to Last Edit**.

1.17.21 Closing Tabbed Windows Using the Context Menu

To close a a tabbed editor or display window, right-click and select **Close** from the context menu.

1.17.22 List of All Open Windows

To see a list of all open tabbed windows, click the small button with the drop-down arrow, located to the right of the tabs and over the tabbed window vertical scroll bar.

To go to one of the listed windows, select it from the drop-down list.

1.17.23 Go to Subprogram Implementation from Package Window

In the window for a package definition, you can press **Ctrl+click** on a procedure or function name to perform the Open Declaration command, which opens the procedure or function implementation (body specification) in a new window.

1.17.24 Select Multiple Table or Column Names in Completion Insight

When entering or editing a SELECT query, you can select multiple tables and columns from the completion insight popup window. Aliases are provided for column and table names if the **Generate Column/Table Aliases Automatically** preference for Code Editor: Completion Insight is enabled.

1.18 For More Information

For more information about SQL Developer and related topics, you may find the following resources helpful:

- SQL Developer Start Page, which contains links for tutorials and OBE (Oracle By Example) lessons, online demonstrations, documentation, and other resources. (If the Start Page tab is not visible, click **Help**, then **Start Page**).
- SQL Developer home page (OTN), which includes links for downloads, white papers, tutorials, viewlets (demonstrations), blogs, a discussion forum, and other sources of information:
http://www.oracle.com/technology/products/database/sql_developer/
- PL/SQL page on OTN: http://www.oracle.com/technology/tech/pl_sql/

- Oracle Accessibility site: <http://www.oracle.com/accessibility/>
- Oracle Corporate site: <http://www.oracle.com/>

Migrating Third-Party Databases

Migration is the process of copying the schema objects and data from a source third-party (non-Oracle) database, such as MySQL, Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, or IBM DB2 (UDB), to an Oracle database. You can perform the migration in an efficient, largely automated way.

Thus, you have two options for working with third-party databases in SQL Developer:

- Creating database connections so that you can view schema objects and data in these databases
- Migrating these databases to Oracle, to take advantage of the full range of Oracle Database features and capabilities

This chapter contains the following major sections:

[Section 2.1, "Migration Quick Start"](#)

[Section 2.2, "Overview of Migration"](#)

[Section 2.3, "Preparing a Migration Plan"](#)

[Section 2.4, "Before You Start Migrating: General Information"](#)

[Section 2.5, "Before You Start Migrating: Source-Specific Information"](#)

[Section 2.6, "Capturing the Source Database"](#)

[Section 2.7, "Creating and Customizing the Converted Model"](#)

[Section 2.8, "Generating the DDL for the Oracle Schema Objects"](#)

[Section 2.9, "Migrating the Data"](#)

[Section 2.10, "Making Queries Case Insensitive"](#)

[Section 2.11, "Testing the Oracle Database"](#)

[Section 2.12, "Deploying the Oracle Database"](#)

[Section 2.13, "Using Migration Reports"](#)

[Section 2.14, "SQL Developer User Interface for Migration"](#)

2.1 Migration Quick Start

To migrate a third-party database to Oracle, the basic actions are: prepare for the migration, create or select associate a migration repository, capture the source database, convert the captured database, generate and run DDL for the new Oracle schema objects, and optionally move data from the source database to the new database.

There are two mechanisms for migrating third-party databases to Oracle: standard migration and quick migration.

2.1.1 Standard Migration

Standard migration involves capturing, converting, generating the database, and performing the data move in several distinct steps. This is the recommended approach when performing a migration. Any issues during these phases can be manually resolved and all objects can be inspected or modified to suit your needs.

Standard Migration: Prepare for Migration

1. Prepare for the migration by reading the appropriate related topics in [Chapter 2, "Migrating Third-Party Databases"](#).
2. Create a migration repository in a new or existing Oracle connection. You may find it simple and convenient to create a separate Oracle database user and connection for migration work. Then, select the connection and create the repository. For example:
 - a. Create an Oracle user named MIGRATIONS with default tablespace USER and temporary tablespace TEMP; and grant it at least the RESOURCE role and the CREATE SESSION and CREATE VIEW privileges. (For multischema migrations, you must grant the RESOURCE role with the ADMIN option; and you must also grant this user the CREATE ROLE, CREATE USER, and ALTER ANY TRIGGER privileges, all with the ADMIN option.)
 - b. Create a database connection named Migration_Repository that connects to the MIGRATIONS user.
 - c. Right-click the Migration_Repository connection, and select **Migration Repository**, then **Associate Migration Repository** to create the repository.
3. Create and open a database connection for the third-party database. (For migrations other than from Microsoft Access, you should set the third party JDBC driver preference before creating the connection.)

For example, create a database connection named Sales_Access to the Microsoft Access database named sales.mdb, and connect to it.

Standard Migration: Capture Source Schema Objects

There are two ways to capture source schema objects: online and offline. Online capture is suitable in most cases, so it is described here.

To perform online capture, right-click the connection name in the Connections navigator and select **Capture *database-type*** (for example, Capture MySQL, Capture Microsoft Access, Capture Microsoft SQL Server, or Capture Sybase Adaptive Server).

Selecting **Capture Microsoft Access** automatically invokes the Microsoft Access exporter tool to create XML files for migrating the schema and the table data.

However, if you want to run the exporter tool manually (for example, to control certain options), click **Migration**, then **Microsoft Access Exporter**, then the item for your version of Microsoft Access. Follow the steps for the exporter tool, which has its own online help.

After the capture, the Captured Models navigator displays an expandable node for the captured objects (for example, sales (Access) for the captured sales.mdb objects, as shown in the figure in [Section 2.14, "SQL Developer User Interface for Migration"](#)).

Standard Migration: Convert Captured Objects

To convert the captured objects to Oracle-format objects, right-click the appropriate node in the Captured Objects navigator and select **Convert to Oracle Model**, and accept the defaults for data mappings (or specify selected mappings if you need to).

After the conversion, the Converted Models navigator displays an expandable node for the converted objects (for example, Converted sales (Access)).

Standard Migration: Generate Oracle Database Objects

1. Generate a SQL*Plus script that creates the DDL statements to create the Oracle database objects that correspond to the source database objects: right-click the appropriate node in the Captured Models navigator and select **Generate**. A SQL Worksheet window opens containing the SQL*Plus statements.
2. In the SQL Worksheet window that was just opened, select (in the drop-down list on the right) an Oracle database connection in which to run the script (next step).
3. Examine the generated SQL*Plus statements, and optionally make any changes. For example, if the database user to own the generated objects already exists (as it will if you are following these quick-step instructions), delete or modify the CREATE USER and related statements.
4. Click the Run Script button in the SQL Worksheet window to run the script.
5. In the Connections navigator, create a connection to the user that was just created.

In the Connections navigator, you should now see the new database objects corresponding to the objects in the third-party database that you migrated.

Standard Migration: Move Data to Oracle Database

If you want, you can migrate (move) any existing data from the source database to the Oracle database. You have two options for data migration: online or offline.

- Online data move: Click **Migration**, then **Migrate Data**. In the dialog box, specify the Source Connection, the Target Connection, and the Converted Model. This method uses JDBC and therefore is constrained by the third-party implementations. This method is suitable for moving small data sets.
- Offline data move: Click **Migration**, then **Script Generation**, then **Generate Data Move Scripts**; specify the converted model and a directory into which to generate the files that you will use for unloading the data from the source database and for importing into Oracle using SQL*Loader. This method is designed for moving large volumes of data.

2.1.2 Quick Migration

Quick migration is a simplified approach that uses a wizard. It provides a quick solution when migrating a simple database; however, for more control of the migration process, you should use [Standard Migration](#).

Quick Migration: Prepare for Migration

1. Prepare for the migration by reading the appropriate related topics in [Chapter 2](#), "Migrating Third-Party Databases".
2. Create a migration repository in a new or existing Oracle connection. You may find it simple and convenient to create a separate Oracle database user and connection for migration work. Then, select the connection and create the repository. For example:

- a. Create an Oracle user named **MIGRATIONS** with default tablespace **USER** and temporary tablespace **TEMP**; and grant it at least **RESOURCE**, **CREATE SESSION**, and **CREATE VIEW** privileges. (For multischema migrations, you must grant the **RESOURCE** role with the **ADMIN** option; and you must also grant this user the **CREATE ROLE**, **CREATE USER**, and **ALTER ANY TRIGGER** privileges, all with the **ADMIN** option.)
 - b. Create a database connection named **Migration_Repository** that connects to the **MIGRATIONS** user.
 - c. Right-click the **Migration_Repository** connection, and select **Migration Repository**, then **Associate Migration Repository** to create the repository.
3. Create an Oracle user whose schema is to be used as the destination for the objects to be migrated, or use an existing Oracle user and schema. Grant sufficient privileges to this user.

For example, if you plan to migrate a Microsoft Access database named **sales.mdb**, you might create an Oracle user named **SALES**, in whose schema the Oracle database objects will be generated.
 4. Create and open an Oracle connection for the schema that you created or selected in the preceding step.

For example, create an Oracle connection named **Sales_Oracle** to the schema associated with user **SALES**, and connect to it.
 5. Create and open a database connection for the third-party database. (For migrations other than from Microsoft Access, you should set the third party JDBC driver preference before creating the connection.)

For example, create a database connection named **Sales_Access** to the Microsoft Access database named **sales.mdb**, and connect to it.

Quick Migration: Migrate Using the Wizard

1. Click **Migration**, then **Quick Migrate**.
2. For **Source Connection**, select the connection for the third-party database to be migrated. For example: **Sales_Access**
3. For **Target Connection**, select the connection for the Oracle Database schema to which the third-party database is to be migrated. For example: **Sales_Oracle**
4. For **Repository**, use the selected existing repository; or if no repository exists, allow SQL Developer to create a migration repository in the schema of the target connection.
5. Click **Verify** to start the pre-migration check.
6. After the pre-migration check completes satisfactorily, specify the **Migration Type**: **Migrate Tables**, **Migrate Tables and Data**, or **Migrate Everything** (all objects).
7. Click **Finish** in the Summary pane to perform the migration.

The specific operations performed depend on the migration type and the type of third-party database being migrated. For example, for a Microsoft Access database, the Exporter for Microsoft Access tool is automatically invoked. *Do not interrupt any of the migration operations.*

If any issues arise during the migration, the quick migration will stop. To proceed with migration, follow the [Standard Migration](#) approach, which will help identify the issues and allow you to modify the appropriate objects.

2.2 Overview of Migration

An Oracle database provides you with better scalability, reliability, increased performance, and better security than third-party databases. For this reason, organizations migrate from their current database, such as Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, or IBM DB2, to an Oracle database. Although database migration can be complicated, SQL Developer enables you to simplify the process of migrating a third-party database to an Oracle database.

SQL Developer captures information from the source database and displays it in the **captured model**, which is a representation of the structure of the source database. This representation is stored in a **migration repository**, which is a collection of schema objects that SQL Developer uses to store migration information.

The information in the repository is used to generate the **converted model**, which is a representation of the structure of the destination database as it will be implemented in the Oracle database. You can then use the information in the captured model and the converted model to compare database objects, identify conflicts with Oracle reserved words, and manage the migration progress. When you are ready to migrate, you generate the Oracle schema objects, and then migrate the data.

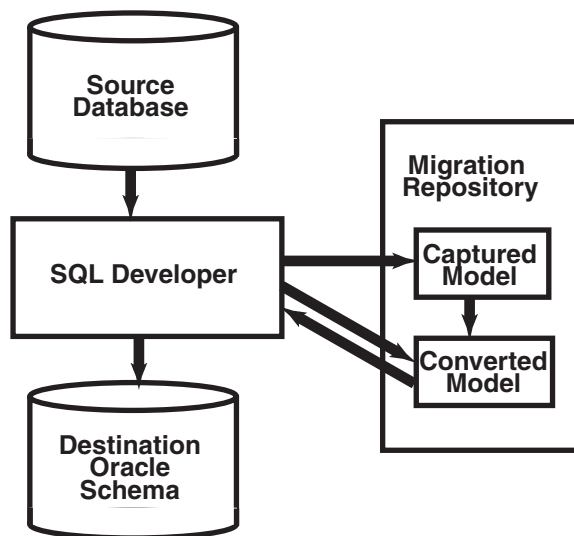
SQL Developer contains logic to extract data from the data dictionary of the source database, create the captured model, and convert the captured model to the converted model.

Using SQL Developer to migrate a third-party database to an Oracle database provides the following benefits:

- Reduces the effort and risks involved in a migration project
- Enables you to migrate an entire third-party database, including triggers and stored procedures
- Enables you to see and compare the captured model and converted model and to customize each if you wish, so that you can control how much automation there is in the migration process
- Provides feedback about the migration through reports

2.2.1 How Migration Works

The components of SQL Developer work together to migrate a third-party database to an Oracle database. [Figure 2-1, "SQL Developer Migration Architecture"](#) shows how SQL Developer reads the information from the source database and creates the Oracle database schema objects. SQL Developer uses the information stored in the migration repository to migrate to the Oracle schema. You can make changes to the captured model or the converted model, or both, before migrating. The information in the converted model is used to complete the migration, that is, to generate the database objects in the destination Oracle schema.

Figure 2–1 SQL Developer Migration Architecture

2.2.2 Migration Implemented as SQL Developer Extensions

Migration support is implemented in SQL Developer as a set of extensions. If you want, you can disable migration support or support for migrating individual third-party databases.

To view the installed extensions, and to enable or disable individual extensions, click **Tools**, then **Preferences**, then **Extensions**. Note that SQL Developer ships with all extensions and third-party database "plugins" available at the time of release, so to begin migrations other than for Microsoft Access, only the third-party drivers need be installed.

2.3 Preparing a Migration Plan

This topic describes the process of how to create a migration project plan. It identifies the sections to include in the migration plan, describes how to determine what to include for each section, and explains how to avoid the risks involved in a migration project. This information includes:

- [Task 1: Determining the Requirements of the Migration Project](#)
- [Task 2: Estimating Workload](#)
- [Task 3: Analyzing Operational Requirements](#)
- [Task 4: Analyzing the Application](#)
- [Task 5: Planning the Migration Project](#)

2.3.1 Task 1: Determining the Requirements of the Migration Project

In this task, you identify which databases you want to migrate and applications that access that database. You also evaluate the business requirements and define testing criteria.

To determine the requirements of the migration project:

1. Define the scope of the project.

There are several choices you must make about the third-party database and applications that access that database in order to define the scope of the migration project. To obtain a list of migration issues and dependencies, you should consider the following

- What third-party databases are you migrating?
 - What is the version of the third-party database?
 - What is the character set of the third-party database?
- What source applications are affected by migrating the third-party database to an Oracle database?
 - What is the third-party application language?
 - What version of the application language are you using?

In the scope of the project, you should have identified the applications you must migrate. Ensure that you have included all the necessary applications that are affected by migrating the database

- What types of connectivity issues are involved in migrating to an Oracle database?
 - Do you use connectivity software to connect the applications to the third-party database? Do you need to modify the connectivity software to connect the applications to the Oracle database?
 - What version of the connectivity software do you use? Can you use this same version to connect to the Oracle database?
- Are you planning to rewrite the applications or modify the applications to work with an Oracle database?

2. Use [Table 2–1](#) to determine whether you have a complex or simple source database environment. Identify the requirements based on the specific scenario.

If the migration project is a simple scenario, you may not have to complete all of the tasks listed in this guide. You make decisions based on your specific environment. For example, if you have a complex scenario, you may require extra testing based on the complexity of the application accessing the database.

Table 2–1 Complex and Simple Scenarios

Complex Scenario	Simple Scenario
More than one of the following:	Contains the following:
<ul style="list-style-type: none"> ■ Large database (greater than 25 GB) ■ Data warehouse ■ Large applications (greater than 100 forms, reports, and batch jobs) ■ Database is used by multiple lines of business ■ Distributed deployment ■ Large user base (greater than 100) ■ High availability requirement (such as a 24 X 7 X 365 environment) 	<ul style="list-style-type: none"> ■ Small database (less than 25 GB) ■ Simple online transaction processing (OLTP) ■ Small application (less than 100 forms, reports, and batch jobs) ■ Database is used by one department ■ Centralized deployment ■ Small user base (less than 100) ■ Average availability (business hours)

3. Determine whether the destination database requires additional hardware and rewriting of backup schedules.

4. Define testing and acceptance criteria.
Define tests to measure the accuracy of the migration. You then use the acceptance criteria to determine whether the migration was successful. The tests that you develop from the requirements should also measure stability, evaluate performance, and test the applications. You must decide how much testing is necessary before you can deploy the Oracle database and applications into a production environment.
5. Create a requirements document with a list of requirements for the migration project.
The requirements document should have clearly defined tasks and number each specific requirement, breaking these into sub-requirements where necessary.

2.3.2 Task 2: Estimating Workload

In this task, you use SQL Developer to make calculated decisions on the amount of work that can be automated and how much is manual.

To estimate the workload:

1. Capture the captured model, create the converted model, and migrate to the destination database.
You can analyze the source database through the captured model and a preview of the destination database through the converted model. After you have captured the source database, analyze the captured data contained in the captured model and the converted model. Ensure the content and structure of the migration repository is correct and determine how much time the entire process takes.
2. Use the Migration Log pane to evaluate the capture and migration process, categorize the total number of database objects, and identify the number of objects that can be converted and migrated automatically.
The migration log provides information about the actions that have occurred and record any warnings and errors. They identify the changes that have been made to the converted model so that you can evaluate if you should make changes to the applications that access the destination database.
3. Evaluate and categorize the issues that occurred. The migration log can help by providing information about:
 - Tables that did not load when you captured the source database
 - Stored procedures, views, and triggers that did not parse when you created the converted model
 - Syntax that requires manual intervention
 - Database objects that were not created successfully when you migrated the destination database
 - Data that did not migrate successfully when you migrated the destination database
4. For each error or warning in the migration log, evaluate the following:
 - Number of times an issue occurred
 - Time required to fix the issues, in person-hours
 - Number of resources required to fix the issue

After you have solved a complex problem, it should be easier and quicker to resolve the next time you have the same problem.

2.3.3 Task 3: Analyzing Operational Requirements

In this task, you analyze the operational requirements, as follows:

1. Evaluate the operational considerations in migrating the source database to a destination database. Consider the following questions:

Note: If the scope of the migration project is a complex scenario as defined in [Table 2-1](#), Oracle recommends that you answer all of these questions. If you have a simple scenario, determine the answers to the most appropriate questions.

- What backup and recovery changes do you require?
 - What downtime is required during the migration?
 - Have you met the performance requirements?
 - Are you changing the operational time window?
 - What effect does the downtime have on the business?
 - What training requirements or additional staff considerations are required?
 - Is it necessary to have the third-party and the Oracle database running simultaneously?
2. For each task, determine the resources and time required to complete.
 3. Create an initial project plan.

Use the information that you have gathered during the requirements and planning stage to develop an initial project plan.

2.3.4 Task 4: Analyzing the Application

In this task, you identify the users of the applications that run on the source database, what hardware it requires, what the application does, and how it interfaces with the source database. You also analyze the method the application uses to connect to the database and identify necessary modifications.

Note: If the migration project is a complex scenario as defined in [Table 2-1](#), Oracle recommends that you answer all of these questions. If you have a simple scenario, determine the answers to the most appropriate questions.

To analyze the application:

1. Determine whether changes to the application are required to make them run effectively on the destination database.
2. If changes are required to the application, determine whether it is more efficient to rewrite or modify the applications.

If you are rewriting the application to use the Oracle database, consider the following:

- a. Create the necessary project documentation to rewrite the application. For example, you need a design specification and requirements documentation.
- b. Rewrite the application according to the specification.
- c. Test the application works against the Oracle database.

If you are modifying the application to use the Oracle database, consider the following:

- a. Identify the number of connections to the database that are in the application and modify these connections to use the Oracle database.
You may need to change the connection information to use an ODBC or JDBC connection.
 - b. Identify the embedded SQL statements that you need to change in the application before you can test it against the Oracle database.
 - c. Test the application using the Oracle database.
3. Allocate time and resource to address each issue associated with rewriting or modifying the application.
 4. Update the general requirements document for the project that you created in Task 1.

2.3.5 Task 5: Planning the Migration Project

In this task, you evaluate the unknown variables that the migration project may contain, such as the difference in the technologies of the source database and the destination database. During the planning stage, you:

- Estimate the budget constraints of the project
- Gather information to produce a migration plan
- Estimate how much time the migration project should take
- Calculate how many resources are required to complete and test the migration

To plan a migration project:

1. Define a list of tasks required to successfully complete the migration project requirements of Task 1.
2. Categorize the list of tasks required to complete the migration project.
You should group these tasks according to your business. This allows you to schedule and assign resources more accurately.
3. Update and finalize the migration project plan based on the information that you have obtained from Task 3 and Task 4.
4. Make sure the migration project plan meets the requirements of the migration project.

The migration plan should include a project description, resources allocated, training requirements, migration deliverable, general requirements, environment analysis, risk analysis, application evaluation, and project schedule.

2.4 Before You Start Migrating: General Information

You may need to perform certain tasks before you start migrating a third-party database to an Oracle database. See the following for more information:

- [Section 2.4.1, "Creating a Database User for the Migration Repository"](#)
- [Section 2.4.2, "Requirements for Creating the Destination Oracle Objects"](#)

See also any information specific to the source database that you will be migrating, as explained in [Section 2.5](#).

Note: Oracle recommends that you make a complete backup of the source database before starting the migration. For more information about backing up the source database, see the documentation for that type of database.

If possible, begin the migration using a development or test environment, not a production database.

2.4.1 Creating a Database User for the Migration Repository

SQL Developer requires a migration repository to migrate a third-party database to an Oracle database. To use an Oracle database for the migration repository, you must have access to that database using a database user account. Oracle recommends that you use a specific user account for migrations. For example, you may want to create a user named MIGRATIONS, create a database connection to that user, and use that connection for the migration repository; and if you wish, you can later delete the MIGRATIONS user to remove all traces of the migration from the database.

When you create a user for migrations, specify the tablespace information as in the following example, instead of using the defaults for tablespaces:

```
CREATE USER migrations IDENTIFIED BY password
  DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp,
```

Do not use a standard account (for example, SYSTEM) for migration.

When SQL Developer creates a migration repository, it creates many schema objects that are intended only for its own use. For example, it creates tables, views, indexes, packages, and triggers, many with names starting with MD_ and MIGR. You should not directly modify these objects or any data stored in them.

2.4.2 Requirements for Creating the Destination Oracle Objects

The user associated with the Oracle database connection used to perform the migration (that is, to run the script containing the generated DDL statements) must have the following roles and privileges:

Note: You must grant these privileges directly to a user account. Granting the privileges to a role, which is subsequently granted to a user account, does not suffice. You cannot migrate a database as the user SYS.

Roles

```
CONNECT WITH ADMIN OPTION
RESOURCE WITH ADMIN OPTION
```

Privileges

```
ALTER ANY ROLE
ALTER ANY SEQUENCE
```

```
ALTER ANY TABLE
ALTER TABLESPACE
ALTER ANY TRIGGER
COMMENT ANY TABLE
CREATE ANY SEQUENCE
CREATE ANY TABLE
CREATE ANY TRIGGER
CREATE VIEW WITH ADMIN OPTION
CREATE PUBLIC SYNONYM WITH ADMIN OPTION
CREATE ROLE
CREATE USER
DROP ANY SEQUENCE
DROP ANY TABLE
DROP ANY TRIGGER
DROP USER
DROP ANY ROLE
GRANT ANY ROLE
INSERT ANY TABLE
SELECT ANY TABLE
UPDATE ANY TABLE
```

For example, you can create a user called `migrations` with the minimum required privileges required to migrate a database by using the following commands:

```
CREATE USER migrations IDENTIFIED BY password
  DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
```

```
GRANT CONNECT, RESOURCE, CREATE VIEW, CREATE PUBLIC SYNONYM TO
  migrations WITH ADMIN OPTION;
```

```
GRANT ALTER ANY ROLE, ALTER ANY SEQUENCE, ALTER ANY TABLE, ALTER TABLESPACE,
ALTER ANY TRIGGER, COMMENT ANY TABLE, CREATE ANY SEQUENCE, CREATE ANY TABLE,
CREATE ANY TRIGGER, CREATE ROLE, CREATE TABLESPACE, CREATE USER, DROP ANY
SEQUENCE, DROP ANY TABLE, DROP ANY TRIGGER, DROP TABLESPACE, DROP USER, DROP ANY
ROLE, GRANT ANY ROLE, INSERT ANY TABLE, SELECT ANY TABLE, UPDATE ANY TABLE TO
migrations;
```

After you have created the converted model is created and done first DDL generation done for the new database, it will be clear from the scripts which privileges will be required for your situation.

2.5 Before You Start Migrating: Source-Specific Information

Depending on the third-party database that you are migrating to an Oracle database, you may have to configure connection information and install drivers. For more information about specific third-party database requirements, see the following:

- [Section 2.5.1, "Before Migrating From IBM DB2"](#)
- [Section 2.5.2, "Before Migrating From Microsoft SQL Server or Sybase Adaptive Server"](#)
- [Section 2.5.3, "Before Migrating From Microsoft Access"](#)
- [Section 2.5.4, "Before Migrating From MySQL"](#)
- [Section 2.5.5, "Before Migrating From Teradata"](#)

2.5.1 Before Migrating From IBM DB2

To configure an IBM DB2 database for migration:

1. Ensure that the source database is accessible by the IBM DB2 database user that is used by SQL Developer for the source connection. This user must be able to see any objects to be captured in the IBM DB2 database; objects that the user cannot see are not captured. For example, if the user can execute a stored procedure but does not have sufficient privileges to see the source code, the stored procedure cannot be captured.
2. Ensure that you can connect to the IBM DB2 database from the system where you have installed SQL Developer.
3. Ensure that you have downloaded the `db2jcc.jar` and `db2jcc_license_cu.jar` files from IBM.
4. In SQL Developer, do the following:
 - a. Click **Tools**, then **Preferences**, then **Database**, then **Third Party JDBC Drivers**.
 - b. Click **Add Entry**.
 - c. Select the `db2jcc.jar` file.
 - d. Click **OK**.
 - e. Repeat steps b through d for the `db2jcc_license_cu.jar` file.

2.5.2 Before Migrating From Microsoft SQL Server or Sybase Adaptive Server

To configure a Microsoft SQL Server or Sybase Adaptive Server database for migration:

1. Ensure that the source database is accessible by the Microsoft SQL Server or Sybase Adaptive Server user that is used by SQL Developer for the source connection. This user must be able to see any objects to be captured in the Microsoft SQL Server or Sybase Adaptive Server database; objects that the user cannot see are not captured. For example, if the user can execute a stored procedure but does not have sufficient privileges to see the source code, the stored procedure cannot be captured.
2. Ensure that you can connect to the Microsoft SQL Server or Sybase Adaptive Server database from the system where you have installed SQL Developer.
3. Ensure that you have downloaded the JTDS JDBC driver from <http://jtds.sourceforge.net/>.
4. In SQL Developer, if you have not already installed the JTDS driver using Check for Updates (on the Help menu), do the following:
 - a. Click **Tools**, then **Preferences**, then **Database**, then **Third Party JDBC Drivers**.
 - b. Click **Add Entry**.
 - c. Select the jar file for the JTDS driver you downloaded from <http://jtds.sourceforge.net/>.
 - d. Click **OK**.
5. In SQL Developer, click **Tools**, then **Preferences**, then **Migration: Identifier Options**, and ensure that the setting is correct for the **Is Quoted Identifier On** option (that is, the setting reflects the database to be migrated).

If this option is enabled, quotation marks (double-quotes) can be used to refer to identifiers; if this option is not enabled, quotation marks identify string literals. As an example of the difference in behavior, consider the following T-SQL code:

```
select col1, "col 2" "column_alias"
```

```
from tablex "table_alias"
```

If the Is Quoted Identifier On option is enabled (checked), the following PL/SQL code is generated:

```
SELECT col1, col_2 "column_alias"  
FROM tablex "table_alias";
```

If the Is Quoted Identifier On option is disabled (not checked), the following PL/SQL code is generated:

```
SELECT col1, 'col 2' "column_alias"  
FROM tablex "table_alias";
```

2.5.3 Before Migrating From Microsoft Access

To configure a Microsoft Access database for migration:

1. Make backup copies of the database file or files.
2. Ensure that the necessary software (Microsoft Access, perhaps other components) is installed on the same system as SQL Developer.
3. Ensure that the Admin user has at least Read Design and Read Data permissions on the MSysObjects, MSysQueries, and MSysRelationships system tables, as explained in the information about the [Access tab](#) in the [Create/Edit/Select Database Connection](#) dialog box.
4. If security is enabled, you should turn it off by copying the contents of the secured database into a new database, as follows:

SQL Developer does not support the migration of Microsoft Access databases that have security enabled. By default, SQL Developer uses the name of the Microsoft Access MDB file as the user name for the destination Oracle user. If you create an Oracle user in this way, the password is ORACLE.

- a. From the **File** menu in Microsoft Access, select **New Database**.
- b. Select the **Blank Database** icon, then click **OK**.
- c. In the File New Database option, type a name for the database, then click **Create**.
- d. From the **File** menu within the new database, select **Get External Data**, then select **Import**.
- e. Select the secured Microsoft Access database that you want to import, then click **Import**.
- f. From the Import Objects dialog, click **Options**.
- g. Select the Relationships and Definition and Data options.
- h. From the Tables tab, choose **Select All**.
- i. Click **OK**.

All Microsoft Access objects are copied over to the new Microsoft Access database, except for the security settings.

5. If the application contains linked tables to other Microsoft Access databases, refresh these links by opening the application in Microsoft Access and performing the following:

From the **Tools** menu in Microsoft Access 97, select **Add Ins**, then select **Linked Table Manager**.

From the **Tools** menu in Microsoft Access 2000, select **Database Utilities**, then select **Linked Table Manager**.

6. Ensure that the Microsoft Access database is not a replica database, but a master database.

When you use the Exporter for Microsoft Access to export, an error message is displayed if the database is a replica. SQL Developer does not support the migration of a replica database.

7. From the **Tools** menu within Microsoft Access, select **Database**, then select **Compact Database** to compact the Microsoft Access database files.
8. Ensure that the Microsoft Access MDB file is accessible from the system where you have installed SQL Developer.
9. Use the Oracle Universal Installer to verify that you have the Oracle ODBC driver installed. If you need to install the driver, it is available on the Oracle Database Server or Database Client CD. You can also download the Oracle ODBC driver from the Oracle Technology Network (OTN) Web site:

<http://www.oracle.com/technology/software/tech/windows/odbc/>

Install the Oracle ODBC driver into an Oracle home directory that contains the Oracle Net Services. You can obtain the Oracle Net Services from the Oracle Client or Oracle Database CD. You install Oracle Net Services to obtain the Net Configuration Assistant and Net Manager. These allow you to create a net configuration in the `tnsnames.ora` file.

Note: For more information about installing the networking products needed to connect to an Oracle database, see the installation guide for your Oracle Database release.

2.5.3.1 Creating Microsoft Access XML Files

To prepare for capturing a Microsoft Access database, the Exporter for Microsoft Access tool must be run, either automatically or manually, as explained in [Section 2.6, "Capturing the Source Database"](#). This tool is packaged as a Microsoft Access MDE file and it allows you to export the Microsoft Access MDB file to an XML file.

Note: Do not modify any of the files created by the Exporter tool.

Each Microsoft Access database that you selected is exported to an XML file. The exporter tool currently does not support creating XML files from secured or replica databases.

2.5.4 Before Migrating From MySQL

To configure a MySQL database for migration, install MySQLConnector/J release 3.1.12 or 5.0.4 on the system where you have installed SQL Developer and set the appropriate SQL Developer preference. Follow these steps:

1. Ensure that you can connect to the MySQL database from the system where you have installed SQL Developer.

2. Ensure that you have downloaded the MySQLConnector/J API from the MySQL Web site at <http://www.mysql.com/>.
3. In SQL Developer, if you have not already installed the MySQL JDBC driver using Check for Updates (on the Help menu), do the following:
 - a. Click **Tools**, then **Preferences**, then **Database**, then **Third Party JDBC Drivers**.
 - b. Click **Add Entry**.
 - c. Select the jar file for the MySQL driver you downloaded from <http://www.mysql.com/>.
 - d. Click **OK**.
4. Ensure that the source database is accessible by the MySQL user that is used by SQL Developer for the source connection. This user must be able to see any objects to be captured in the MySQL database; objects that the user cannot see are not captured. For example, if the user can execute a stored procedure but does not have sufficient privileges to see the source code, the stored procedure cannot be captured.

2.5.5 Before Migrating From Teradata

Note that for the current release of SQL Developer, the following Teradata objects will *not* be migrated to Oracle: procedures, functions, triggers, views, macros, and BTEQ scripts.

To configure a Teradata database for migration:

1. Ensure that the source database is accessible by the Teradata database user that is used by SQL Developer for the source connection. This user must be able to see any objects to be captured in the Teradata database; objects that the user cannot see are not captured.
2. Ensure that you can connect to the Teradata database from the system where you have installed SQL Developer.
3. Ensure that you have downloaded the `tdgssconfig.jar` and `terajdbc4.jar` files from Teradata.
4. In SQL Developer, do the following:
 - a. Click **Tools**, then **Preferences**, then **Database**, then **Third Party JDBC Drivers**.
 - b. Click **Add Entry**.
 - c. Select the `tdgssconfig.jar` file.
 - d. Click **OK**.
 - e. Repeat steps b through d for the `terajdbc4.jar` file.

2.6 Capturing the Source Database

Before migrating a third-party database, you must extract information from the database. This information is a representation of the structure of the source database, and it is called the captured model. The process of extracting the information from the database is called capturing the source database.

The capture can be done online or offline:

- **Online capture** is done in a convenient guided sequence within the SQL Developer interface, as explained in [Section 2.6.1, "Online Capture"](#). You can use online capture with all supported third-party databases.
- **Offline capture** involves creating a script that you run later, as explained in [Section 2.6.2, "Offline Capture"](#). You can use offline capture with IBM DB2, MySQL, Microsoft SQL Server databases, and Sybase Adaptive Server.

After capturing the source database, you can view the source database information in the captured model in SQL Developer. If necessary, you can modify the captured model and change data type mappings.

Note: Oracle recommends that you do not change the default data type mappings unless you are an experienced Oracle database administrator.

2.6.1 Online Capture

To perform an online capture of the source database, you can have the capture performed automatically as part of the Quick Migrate option, or you can have it performed as a separate operation by right-clicking the connection name in the Connections navigator and selecting **Capture *product-name*** (for example, Capture MySQL, Capture Microsoft Access, Capture Microsoft SQL Server, or Capture Sybase Adaptive Server).

For a Microsoft Access database, selecting **Capture *product-name*** automatically invokes the Microsoft Access exporter tool to create XML files for migrating the schema and the table data. However, if you want to run the exporter tool manually (for example, to control certain options), click **Migration**, then **Microsoft Access Exporter**, then the item for your version of Microsoft Access. Follow the steps for the exporter tool, which has its own online help.

2.6.2 Offline Capture

To perform an offline capture of an IBM DB2, MySQL, Microsoft SQL Server, or Sybase Adaptive Server database, you create a set of offline capture scripts, run these scripts outside SQL Developer to create the script output (a dump of the third party metadata tables), and load the script output (the .ocp file containing the converted model) using SQL Developer.

- To create the script file (a Windows .bat file or a Linux or UNIX .sh file) and related files, click **Tools**, then **Migration**, then **Third Party Database Offline Capture**, then **Create Database Capture Scripts**.

When this operation completes, you are notified that several files (.bat, .sql, .ocp) have been created, one of which is the controlling script. You must run the controlling script (outside SQL Developer) to populate the object capture properties (.ocp) file with information about the converted model.

- To load the converted model from the object capture properties (.ocp) file generated by the offline capture controlling script, click **Tools**, then **Migration**, then **Third Party Database Offline Capture**, then **Load Database Capture Script Output**.

2.6.2.1 IBM DB2 Offline Capture Notes

Script files and the db2_x.ocp file are generated in the target folder. The main script is startDump.xxx, which you must execute to produce the schema dump. The script files

prompt you for the database name, user name, and password, and they use this information to connect to the local DB2 database. The scripts generate the schema dump for database objects within object-specific folders.

To capture the schema information in offline file format, use a command in the following format (with the `db2` executable in the run path):

```
db2 -x +o -r <file name> <schema query>
```

To export the schema data in offline file format, use a command in the following format (with the `db2` executable in the run path):

- For DB2 version 9 data export:

```
db2 export to <file name> of DEL modified by lobsinsefiles coldel"#"
timestampformat="\YYYY/MM/DD HH.mm.ss\" datesiso nochardel <select query>
```

- For DB2 version 8 data export:

```
db2 export to <file name> of DEL modified by coldel"#"
timestampformat="\YYYY/MM/DD HH.mm.ss\" datesiso nochardel <select query>
```

DB2 version 9 supports LOB data in separate files, which is better for migrating large data sizes. With version 8, to support large LOB data, you must modify the oracle `ctl` file command and `db2` command in `unload_script.bat` or `unload_script.sh`.

The table data is exported to files with names in the format `<catalog>.<schema>.<table>.dat`. The format of file is as follows: `data1#<COL_DEL>#data2#<COL_DEL>...<ROW_DEL>` where `COL_DEL` and `ROW_DEL` come from migration offline preference settings.

Before you execute the DB2 data dump script, you must log in by entering a command in the following format:

```
db2 connect to <catalog> user <user name> using <password>
```

You can then execute the script using the logged connection session.

2.7 Creating and Customizing the Converted Model

After you capture a third-party database, the next step is to convert it, creating the converted model. The converted model is a representation of the structure of the destination database. SQL Developer creates the converted model using the information from the captured model.

By default, all procedures, functions, triggers, and views are copied to the converted model during translation and translated to Oracle PL/SQL. However, if translation fails for any of the objects, those objects appear in the converted model but their original SQL code remains unchanged. Objects that remain in their original SQL code will not be used when the generation scripts are created. Therefore, to have any such objects migrated, you must either fix the problem in the original SQL code before generating the script or edit the generated script to replace the original SQL code with valid PL/SQL code.

To convert a captured model to a converted model, right-click the appropriate node in the Captured Models navigator and select **Convert to Oracle**, and specify or accept the defaults for data mappings.

The following topic describes how to modify the converted model, if this becomes necessary:

- [Correcting Errors in the Converted Model](#)

2.7.1 Correcting Errors in the Converted Model

If error messages with the prefix `Parse Exception` are listed in the migration log, manual intervention is required to resolve the issues. To complete the converted model:

1. Note the converted model schema object that failed.
2. Select that schema object in the converted model.
3. Copy the schema objects DDL and paste it into the translation scratch editor (displayed by clicking Migration, then Translation Scratch Editor).
4. Inspect the properties on the schema object in the translation scratch editor for possible causes of the error.
5. Modify a property of the schema object in the translation scratch editor.
For example, you might comment out one line of a stored procedure.
6. Translate using the appropriate translator.
7. If the error appears again, repeat steps 2 to 6.
8. If the error cannot be resolved in this way, it is best to modify the object manually in the converted model.

2.8 Generating the DDL for the Oracle Schema Objects

To generate the DDL statements to create the Oracle schema objects, you must already have captured the captured model and created the converted model. After you generate the DDL, you run the DDL statements to cause the objects to be created in the Oracle database. At this point, the database schema is migrated to Oracle.

After you generate and run the DDL statements to migrate the schema objects, you can migrate the data from the original source database, as explained in [Section 2.9](#).

2.9 Migrating the Data

After you have generated and run DDL statements to create the Oracle schema objects for the migrated database, you can migrate (move) any existing data from the source database to the Oracle database. You have two options for data migration: online or offline.

- **Online data move:** Click **Migration**, then **Migrate Data**. In the dialog box, specify the Source Connection, the Target Connection, and the Converted Model. This method uses JDBC and therefore is constrained by the third-party implementations. Online data moves are suitable for small data sets.
- **Offline data move:** Click **Migration**, then **Generate Offline Data Move Scripts**; specify the converted model and a directory into which to generate the files that you will use for unloading the data from the source database and for importing into Oracle using SQL*Loader. The offline data move approach is designed for moving large volumes of data.

2.9.1 Transferring the Data Offline

To transfer the data offline, you generate and use scripts to copy data from the source database to the destination database. During this process you must:

- Use SQL Developer to generate the data unload scripts for the source database and corresponding data load scripts for the destination database.

- Run the data unload scripts to create data files from the source database using the appropriate procedure for your source database:
 - [Creating Data Files From Microsoft SQL Server or Sybase Adaptive Server](#)
 - [Creating Data Files From Microsoft Access](#)
 - [Creating Data Files From MySQL](#)
 - For IBM DB2, see the chapter about offline data loading in *Oracle SQL Developer Supplementary Information for IBM DB2 Migrations*.
 - For Teradata, perform the offline data move using BTEQ and SQL*Loader.
- Run the data load scripts using SQL*Loader to populate the destination database with the data from these data files as described in [Section 2.9.1.4](#).

2.9.1.1 Creating Data Files From Microsoft SQL Server or Sybase Adaptive Server

To create data files from a Microsoft SQL Server or Sybase Adaptive Server database:

1. Copy the contents of the directory where SQL Developer generated the data unload scripts onto the computer where the source database is installed.
2. Edit the BCP extract script to include the name of the source database server.
 - On Windows, edit the `unload_script.bat` script to alter the bcp lines to include the appropriate variables.

The following shows a line from a sample `unload_script.bat` script:

```
bcp "AdventureWorks.dbo.AWBuildVersion" out  
"[AdventureWorks].[dbo].[AWBuildVersion].dat" -q -c -t "<EOFD>" -r "<EORD>"  
-U<Username> -P<Password> -S<ServerName>
```

3. Run the BCP extract script.

- On Windows, enter:

```
prompt> unload_script.bat
```

This script creates the data files in the current directory.

4. Copy the data files and scripts, if necessary, to the target Oracle database system, or to a system that has access to the target Oracle database and has SQL*Loader (Oracle Client) installed.

2.9.1.2 Creating Data Files From Microsoft Access

To create data files from a Microsoft Access database, use the Exporter for Microsoft Access tool.

Note: For information about how to create data files from a Microsoft Access database, see online help for the exporter tool.

2.9.1.3 Creating Data Files From MySQL

To create data files from a MySQL database:

1. Copy the contents of the directory where SQL Developer generated the data unload scripts, if necessary, onto the system where the source database is installed or a system that has access to the source database and has the `mysqldump` tool installed.

2. Edit the `unload_script` script to include the correct host, user name, password, and destination directory for the data files.
 - On Windows, edit the `unload_script.bat` script.
 - On Linux or UNIX, edit the `unload_script.sh` script.

The following shows a line from a sample `unload_script.bat` script:

```
mysqldump -h localhost -u <USERNAME> -p<PASSWORD> -T <DESTINATION_PATH>
--fields-terminated-by="<EOFD>" --fields-escaped-by=" "
--lines-terminated-by="<EORD>" "CarrierDb" "CarrierPlanTb"
```

Edit this line to include the correct values for *USERNAME*, *PASSWORD*, and *DESTINATION PATH*. Do not include the angle brackets in the edited version of this file.

In this command line, `localhost` indicates a loopback connection, which is required by the `-T` option. (See the `mysqldump` documentation for more information.)

3. Run the script.
 - On Windows, enter:


```
prompt> unload_script.bat
```
 - On Linux or UNIX, enter:


```
prompt> chmod 755 unload_script.sh
prompt> sh ./unload_script.sh
```

This script creates the data files in the current directory.

4. Copy the data files and scripts, if necessary, to the target Oracle database system, or to a system that has access to the target Oracle database and has SQL*Loader (Oracle Client) installed.

2.9.1.4 Populating the Destination Database Using the Data Files

To populate the destination database using the data files, you run the data load scripts using SQL*Loader:

1. Navigate to the directory where you created the data unload scripts.
2. Edit the `oracle_ctl.bat` (Windows systems) or `oractl_ctl.sh` (Linux or UNIX systems) file, to provide the appropriate user name and password strings.
3. Run the SQL Load script.
 - On Windows, enter:


```
prompt> oracle_ctl.bat
```
 - On Linux or UNIX, enter:


```
prompt> ./oracle_ctl.sh
```

For Microsoft SQL Server and Sybase migrations, if you are inserting into BLOB fields with SQL*Loader, you will receive the following error:

```
SQL*Loader-309: No SQL string allowed as part of LARGEOBJECT field specification
```

To handle situations indicated by this error, you can use either one of the following options:

- Enable the Generate Stored Procedure for Migrate Blobs Offline SQL Developer preference (see [Migration: Generation Options](#)).
- Use the following [Workaround](#).

Workaround

The workaround is to load the data (which is in hex format) into an additional CLOB field and then convert the CLOB to a BLOB through a PL/SQL procedure.

The only way to export binary data properly through the Microsoft SQL Server or Sybase Adaptive Server BCP is to export it in a hexadecimal (hex) format; however, to get the hex values into Oracle, save them in a CLOB (holds text) column, and then convert the hex values to binary values and insert them into the BLOB column. The problem here is that the HEXTORAW function in Oracle only converts a maximum of 2000 hex pairs. Consequently, write your own procedure that will convert (piece by piece) your hex data to binary. (In the following steps and examples, modify the START.SQL and FINISH.SQL to reflect your environment.

The following shows the code for two scripts, `start.sql` and `finish.sql`, that implement this workaround. Read the comments in the code, and modify any SQL statements as needed to reflect your environment and your needs.

Note: After you run `start.sql` and before you run `finish.sql`, run BCP; and before you run BCP, change the relevant line in the `.ctl` file from:

```
<blob_column> CHAR(2000000) "HEXTORAW (:<blob_column>)"
```

to:

```
<blob_column>_CLOB CHAR(2000000)
```

```
-- START.SQL
-- Modify this for your environment.

-- This should be executed in the user schema in Oracle that contains the table.
-- DESCRIPTION:
-- ALTERS THE OFFENDING TABLE SO THAT THE DATA MOVE CAN BE EXECUTED
-- DISABLES TRIGGERS, INDEXES AND SEQUENCES ON THE OFFENDING TABLE

-- 1) Add an extra column to hold the hex string;
alter table <tablename> add (<blob_column>_CLOB CLOB);

-- 2) Allow the BLOB column to accept NULLS
alter table <tablename> MODIFY <blob_column> NULL;

-- 3) Disable triggers and sequences on <tablename>
alter trigger <triggername> disable;

alter table <tablename> drop primary key cascade;

drop index <indexname>;

-- 4) Allow the table to use the tablespace
alter table <tablename> move lob (<blob_column>) store as (tablespace lob_
tablespace);

alter table <tablename> move lob (<blob_column>_clob) store as (tablespace lob_
```



```

tablespace);

COMMIT;

-- END OF FILE

-- FINISH.SQL
-- Modify this for your environment.

-- This should be executed in the table schema in Oracle.
-- DESCRIPTION:
-- MOVES THE DATA FROM CLOB TO BLOB
-- MODIFIES THE TABLE BACK TO ITS ORIGINAL SPEC (without a clob)
-- THEN ENABLES THE SEQUENCES, TRIGGERS AND INDEXES AGAIN

-- Currently we have the hex values saved as
-- text in the <blob_column>_CLOB column
-- And we have NULL in all rows for the <blob_column> column.
-- We have to get BLOB locators for each row in the BLOB column

-- put empty blobs in the blob column
UPDATE <tablename> SET <blob_column>=EMPTY_BLOB();

COMMIT;

-- create the following procedure in your table schema
CREATE OR REPLACE PROCEDURE CLOBTOBLOB
AS
inputLength NUMBER; -- size of input CLOB
offSet NUMBER := 1;
pieceMaxSize NUMBER := 2000; -- the max size of each peice
piece VARCHAR2(2000); -- these pieces will make up the entire CLOB
currentPlace NUMBER := 1; -- this is where were up to in the CLOB
blobLoc BLOB; -- blob locator in the table
clobLoc CLOB; -- clob locator pointsthis is the value from the dat file

-- THIS HAS TO BE CHANGED FOR SPECIFIC CUSTOMER TABLE
-- AND COLUMN NAMES
CURSOR cur IS SELECT <blob_column>_clob clob_column , <blob_column> blob_column
FROM /*table*/<tablename> FOR UPDATE;

cur_rec cur%ROWTYPE;

BEGIN

OPEN cur;
FETCH cur INTO cur_rec;

WHILE cur%FOUND
LOOP
--RETRIVE THE clobLoc and blobLoc
clobLoc := cur_rec.clob_column;
blobLoc := cur_rec.blob_column;

currentPlace := 1; -- reset evertime
-- find the lenght of the clob
inputLength := DBMS_LOB.getLength(clobLoc);

-- loop through each peice

```

```
LOOP
-- get the next piece and add it to the clob
piece := DBMS_LOB.subStr(clobLoc,pieceMaxSize,currentPlace);

-- append this piece to the BLOB
DBMS_LOB.WRITEAPPEND(blobLoc, LENGTH(piece)/2, HEXTORAW(piece));

currentPlace := currentPlace + pieceMaxSize ;

EXIT WHEN inputLength < currentplace;
END LOOP;

FETCH cur INTO cur_rec;
END LOOP;

END CLOBtoBLOB;
/

-- now run the procedure
-- It will update the blob column with the correct binary representation
-- of the clob column
EXEC CLOBtoBLOB;

-- drop the extra clob column
alter table <tablename> drop column <blob_column>_clob;

-- 2) apply the constraint we removed during the data load
alter table <tablename> MODIFY FILEBINARY NOT NULL;

-- Now re enable the triggers, indexes and primary keys
alter trigger <triggername> enable;

ALTER TABLE <tablename> ADD ( CONSTRAINT <pkname> PRIMARY KEY ( <column> ) );

CREATE INDEX <index_name> ON <tablename>( <column> );

COMMIT;

-- END OF FILE
```

2.10 Making Queries Case Insensitive

With several third-party databases, it is common for queries to be case insensitive. For example, in such cases the following queries return the same results:

```
SELECT * FROM orders WHERE sales_rep = 'Oracle';
SELECT * FROM orders WHERE sales_rep = 'oracle';
SELECT * FROM orders WHERE sales_rep = 'OrAcLe';
```

If you want queries to be case insensitive for a user in the Oracle database, you can create an AFTER LOGON ON DATABASE trigger, in which you set, for that database user, the NLS_SORT session parameter to an Oracle sort name with _CI (for "case insensitive") appended.

The following example causes queries for user SMITH to use the German sort order and to be case insensitive:

```
CREATE OR REPLACE TRIGGER set_sort_order AFTER LOGON ON DATABASE
DECLARE
    username VARCHAR2(30);
```

```
BEGIN
  username:=SYS_CONTEXT('USERENV','SESSION_USER');
  IF username LIKE 'SMITH' then
    execute immediate 'alter session set NLS_COMP=LINGUISTIC';
    execute immediate 'alter session set NLS_SORT=GERMAN_CI';
  END IF;
END;
```

2.11 Testing the Oracle Database

During the testing phase, you test the application and Oracle database to make sure that the:

- Migrated data is complete and accurate
- Applications function in the same way as the source database
- Oracle database producing the same results as the source database
- Applications and Oracle database meet the operational and performance requirements

You may already have a collection of unit tests and system tests from the original application that you can use to test the Oracle database. You should run these tests in the same way that you ran tests against the source database. However, regardless of added features, you should ensure that the application connects to the Oracle database and that the SQL statements it issues produces the correct results.

Note: The tests that you run against the application vary depending on the scope of the application. Oracle recommends that you thoroughly test each SQL statement that is changed in the application. You should also test the system to make sure that the application functions the same way as in the third-party database.

See also the following:

- [Section 2.11.1, "Testing Methodology"](#)
- [Section 2.11.2, "Testing the Oracle Database"](#)

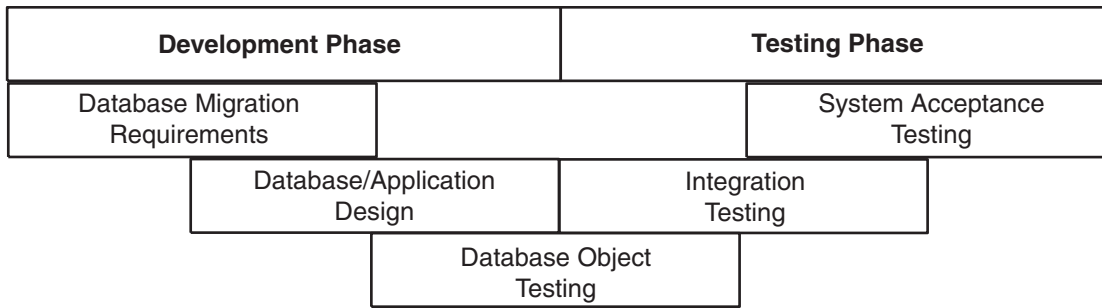
2.11.1 Testing Methodology

Many constraints shape the style and amount of testing that you perform on a database. Testing can contain one or all of the following:

- Simple data validation
- Full life cycle of testing addressing individual unit tests
- System and acceptance testing

You should follow a strategy for testing that suits your organization and circumstances. Your strategy should define the process by which you test the migrated application and Oracle database. A typical test method is the V-model, which is a staged approach where each feature of the database creation is mirrored with a testing phase.

[Figure 2-2, "V-model with a Database Migration"](#) shows an example of the V-model with a database migration scenario:

Figure 2–2 V-model with a Database Migration

There are several types of tests that you use during the migration process. During the testing stage, you go through several cycles of testing to enhance the quality of the database. The test cases you use should make sure that any issues encountered in a previous version of the Oracle database are not introduced again.

For example, if you have to make changes to the migrated schema based on test results, you may need to create a new version of the Oracle database schema. In practice, you use SQL Developer to create a base-line Oracle schema at the start of testing, and then edit this schema as you progress with testing.

Note: Oracle recommends that you track issues that you find during a testing cycle in an issue tracking system. Track these issues against the version of the database or application that you are testing.

2.11.2 Testing the Oracle Database

Use the test cases to verify that the Oracle database provides the same business logic results as the source database.

Note: Oracle recommends that you define completion criteria so that you can determine the success of the migration.

This procedure explains one way of testing the migrated database. Other methods are available and may be more appropriate to your business requirements.

To test the Oracle database:

1. Create a controlled version of the migrated database.

Oracle recommends that you keep the database migration scripts in a source control system.
2. Design a set of test cases that you can use to test the Oracle database from unit to system level. The test cases should:
 - a. Ensure the following:
 - All the users in the source database have migrated successfully
 - Privileges and grants for users are correct
 - Tables have the correct structure, defaults are functioning correctly, and errors did not occur during mapping or generation

- Describe the persistent after effect, if any
 - e. Attempt to fix the errors.
 - f. Return to step 1.
7. Identify acceptance tests that you can use to make sure the Oracle database is an acceptable quality level.

2.11.2.1 Guidelines for Creating Tests

You may already have a collection of unit tests and system tests from the original application that you can use to test the Oracle database. However, if you do not have any unit or system tests, you need to create them. When creating test cases, use the following guidelines:

- Plan, specify, and execute the test cases, recording the results of the tests.
The amount of testing you perform is proportional to the time and resources that are available for the migration project. Typically, the testing phase in a migration project can take anywhere from 40% to 60% of the effort for the entire project.
- Identify the components that you are testing, the approach to the test design and the test completion criteria.
- Define each test case so that it is reproducible.
A test that is not reproducible is not acceptable for issue tracking or for an audit process.
- Divide the source database into functions and procedures and create a test case for each function or procedure. In the test case, state what you are going to test, define the testing criteria, and describe the expected results.
- Record the expected result of each test case.
- Verify that the actual results meet the expected results for each test.
- Define test cases that produce negative results as well as those that you expect a positive result.

2.11.2.2 Example of a Unit Test Case

The following displays a sample unit test plan for Windows:

Name	Jane Harrison
Module	Table Test Emp
Date test completed	23 May 2007
Coverage log file location	mwb\database\TableTestEmp
Description	This unit test tests that the emp table was migrated successfully.
Reviewed by	John Smith

Task ID	Task Description	Expected Result	Verified (Yes/No)
1	<p>Run the following on the source database for each table:</p> <pre>select count(*) from emp</pre> <p>Run the following on the destination database for each table:</p> <pre>select count(*) from emp</pre>	<p>On the source database, the count(*) produces a number. In this case, the number is the number of rows in each table.</p> <p>On the destination database, the count(*) number corresponds to the number of rows in the new Oracle table.</p>	<p>Yes</p> <p>The number of rows in each table is the same in the source and destination databases.</p>
2	<p>Run the following on the source database for each table:</p> <pre>select sum(salary) from emp</pre> <p>Run the following on the destination database for each table:</p> <pre>select sum(salary) from emp</pre>	<p>On the source database, sum(salary) produces a check sum for the sum of the data in each table.</p> <p>On the destination database, sum(salary) corresponds to the sum of the salary in the emp table.</p>	<p>Yes</p> <p>The sum for each table is the same in the source and destination databases.</p>

2.12 Deploying the Oracle Database

Deploying the migrated and tested Oracle database within a business environment can be difficult. Therefore, you may need to consider different rollout strategies depending on your environment. Several rollout strategies are identified for you, but you may use another approach if that is recommended by your organization.

During the deployment phase, you move the destination database from a development to a production environment. A group separate from the migration and testing team, may perform the deployment phase, such as the in-house IT department.

Deployment involves the following:

- [Choosing a Rollout Strategy](#)
- [Deploying the Destination Database](#)

2.12.1 Choosing a Rollout Strategy

The strategy that you use for migrating a third-party database to an Oracle database must take into consideration the users and the type of business that may be affected during the transition period. For example, you may use the Big Bang approach because you do not have enough systems to run the source database and Oracle database simultaneously. Otherwise, you may want to use the Phased approach to make sure that the system is operating in the user environment correctly before it is released to the general user population. You can use one of the following approaches.

2.12.1.1 Phased Approach

Using the Phased approach, you migrate groups of users at different times. You may decide to migrate a department or a subset of the complete user-base. The users that you select should represent a cross-section of the complete user-base. This approach allows you to profile users as you introduce them to the Oracle database. You can reconfigure the system so that only selected users are affected by the migration and unscheduled outages only affect a small percentage of the user population. This

approach may affect the work of the users you migrated. However, because the number of users is limited, support services are not overloaded with issues.

The Phased approach allows you to debug scalability issues as the number of migrated users increases. However, using this approach may mean that you must migrate data to and from legacy systems during the migration process. The application architecture must support a phased approach.

2.12.1.2 Big Bang Approach

Using the Big Bang approach, you migrate all of the users at the same time. This approach may cause schedule outages during the time you are removing the old system, migrating the data, deploying the Oracle system, and testing that the system is operating correctly. This approach relies on you testing the database on the same scale as the original database. It has the advantage of minimal data conversion and synchronization with the original database because that database is switched off. The disadvantage is that this approach can be labor intensive and disruptive to business activities due to the switch over period needed to install the Oracle database and perform the other migration project tasks.

2.12.1.3 Parallel Approach

Using the Parallel approach, you maintain both the source database and destination Oracle database simultaneously. To ensure that the application behaves the same way in the production environment for the source database and destination database, you enter data in both databases and analyze the data results. The advantage of this approach is if problems occur in the destination database, users can continue using the source database. The disadvantage of the Parallel approach is that running and maintaining both the source and the destination database may require more resources and hardware than other approaches.

2.12.2 Deploying the Destination Database

There are several ways to deploy the destination database. The following task is an example that you should use as a guideline for deploying the destination database.

Note: If you have a complex scenario as defined in [Table 2-1](#), Oracle recommends that you complete all of the deployment tasks. However, if you have a simple scenario, you should choose the deployment tasks appropriate to your organization.

1. Configure the hardware, if necessary.

In a large scale or complex environment, you must design the disk layout to correspond with the database design. If you use redundant disks, align them in stripes that you can increase as the destination database evolves. You must install and configure the necessary disks, check the memory, and configure the system.

2. Make sure the operating system meets the parameters of the Oracle configuration.

Before installing any Oracle software, make sure that you have modified all system parameters. For more information about modifying system parameters, see the relevant installation guide for your platform, such as Solaris Operating System.

3. Install the Oracle software.

Aside from the Oracle software that allows you to create an Oracle database, you may need to install ancillary software to support the application, such as Extract Transformation and Load (ETL) Software for data warehousing.

4. Create the destination database from the source database and migrate the data to the Oracle database.

There are several ways of putting the destination database into production after testing it, such as:

- Place the successfully tested database into production. The test system is now the production system.
 - Use Oracle Export to extract the destination database from the successfully tested database and use Oracle Import to create that database within the production environment.
 - Use the tested migration scripts to create the Oracle database and populate it with data using SQL*Loader.
5. Perform the final checks on the destination database and applications.
 6. Place the destination database into production using one of the rollout strategies.
 7. Perform a final audit by doing the following:
 - Audit the integrity of the data
 - Audit the validity of the processes, such as back-up and recovery
 - Obtain sign-off for the project, if necessary

2.13 Using Migration Reports

Several SQL Developer reports provide information about objects that have been captured, converted, and generated during operations designed to migrate third-party databases to Oracle. Each report uses information from a selected migration project. These reports are listed in the Reports navigator: click **Migration Reports**.

Automatic Name Changes: Lists name changes that were automatically made when the converted model was generated. Some names are automatically changed during conversion, so that they are Oracle compliant and no collisions exist. For more information about the renaming of objects, see:

<http://wiki.oracle.com/page/SDMW+Identifier+Name+Conversion>

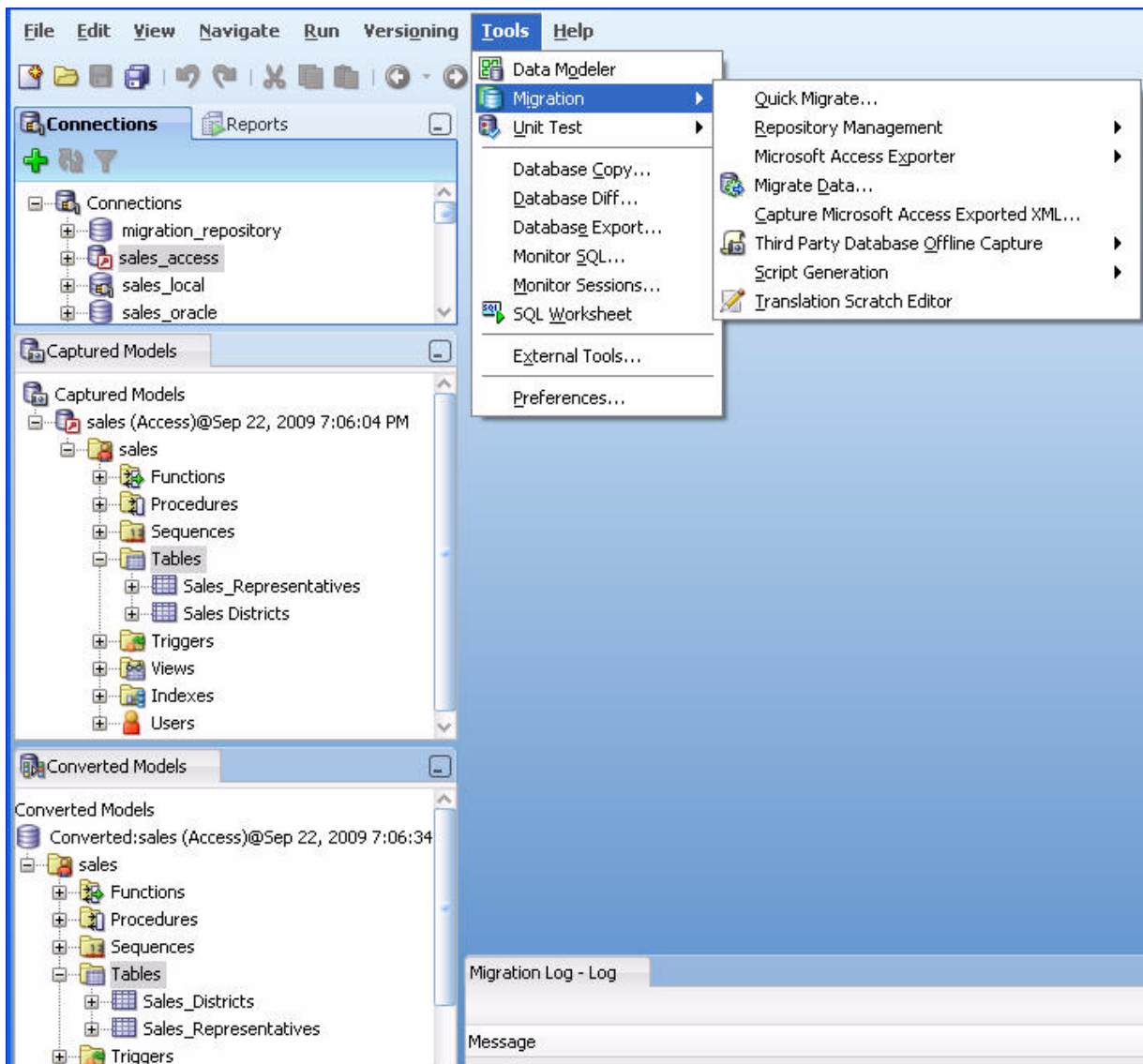
Migration Details: Lists all the objects and their status through each phase of the migration (capture, convert, generate).

Migration Summary: Gives a summary of the migration. Includes the total number of procedures, triggers, and views that were captured, converted, and generated successfully.

2.14 SQL Developer User Interface for Migration

If you are performing database migration, you need to use some migration-specific features in addition to those described in [Section 1.2, "SQL Developer User Interface"](#). The user interface includes some additional navigator tabs and panes (Captured Models and Converted Models) and a Migration menu, and many smaller changes throughout the interface. [Figure 2–3, "Main Window for a Database Migration"](#) shows the SQL Developer main window with objects reflecting the migration of a Microsoft Access application named sales.mdb. It also shows the [Migration Menu](#).

Figure 2-3 Main Window for a Database Migration



In this figure:

- The **Connections** navigator shows three database connections: `migration_repository` for a connection (to a user named `MIGRATION`) used for the migration repository, `sales_access` connected to a Microsoft Access database named `sales.mdb`, and `sales_oracle` connected to an Oracle user named `SALES` whose schema owns the migrated schema objects.
- The **Captured Models** navigator shows one captured model, which was created using an XML file created by the exporter tool for Access applications. (If the source database is a type other than Microsoft Access, the procedure for creating the captured model is different: you can generate it directly from the source database connection.)
- The **Converted Models** navigator shows one converted model, which is an Oracle representation of the source database. The converted model is created from the captured model, and the converted model is used to generate the schema objects that you can see using an Oracle database connection (`sales_oracle` in this figure).

2.14.1 Migration Menu

The Migration menu contains options related to migrating third-party databases to Oracle. To display the Migration menu, click **Tools**, then **Migration**.

Quick Migrate: Displays a dialog box for performing a quick migration using many default values.

Repository Management: Enables you to create, delete, or truncate (remove all data from) a migration repository; select the current migration repository; and disconnect from the current migration repository (which deactivates the current repository but does not disconnect from the database).

Microsoft Access Exporter: Contains submenu items from which you specify the version of the exporter tool to use to create an XML file to be used for creating the captured model. You can also use the exporter tool to export table data. Specify the exporter tool version for the version of Access that is on your PC and that was used to create the .mdb file.

Migrate Data: Displays a dialog box for performing an online migration of table data from the source database to the Oracle schema.

Script Generation: Generate Oracle DDL displays DDL (data definition language) statements in a SQL Worksheet window, where you can then run the script to create the Oracle schema and schema objects; **Generate Data Move Scripts** displays a dialog box for specifying the location in which to create files for performing an offline migration of table data from the source database to the Oracle schema.

Capture Microsoft Access Exporter XML: Creates a captured model of a Microsoft Access database from the XML file created by the exporter tool.

MySQL, SQL Server, and Sybase Offline Capture: Create Database Capture Scripts specifies options for creating an offline capture properties (.ocp) file, which you can later load and run; **Load Database Capture Script Output** enables you to select a script to be loaded and run.

Script Generation: Generate Oracle DDL specifies the converted model for which to generate Oracle DDL and produces a SQL*Plus script file that you use for offline generation (that is, you can run the script to create the appropriate objects in the Oracle database); **Generate Data Move Scripts** specifies the converted model and the destination directory if you are performing offline data migration.

Translation Scratch Editor: Displays the translation scratch editor, which is explained in [Section 2.14.5](#).

2.14.2 Other Menus: Migration Items

The **View** menu has the following items related to database migration:

- **Captured Models:** Displays the Captured Models navigator.
- **Converted Models:** Displays the Converted Models navigator.

2.14.3 Migration Preferences

The SQL Developer user preferences window (displayed by clicking **Tools**, then **Preferences**) contains a **Migration** pane with several related subpanes, and a **Translation** pane with a Translation Preferences subpane.

For information about these preferences, click **Help** in the pane, or see [Section 1.12.10](#), "Migration".

2.14.4 Migration Log Panes

Migration Log: Contains errors, warnings, and informational messages relating to migration operations.

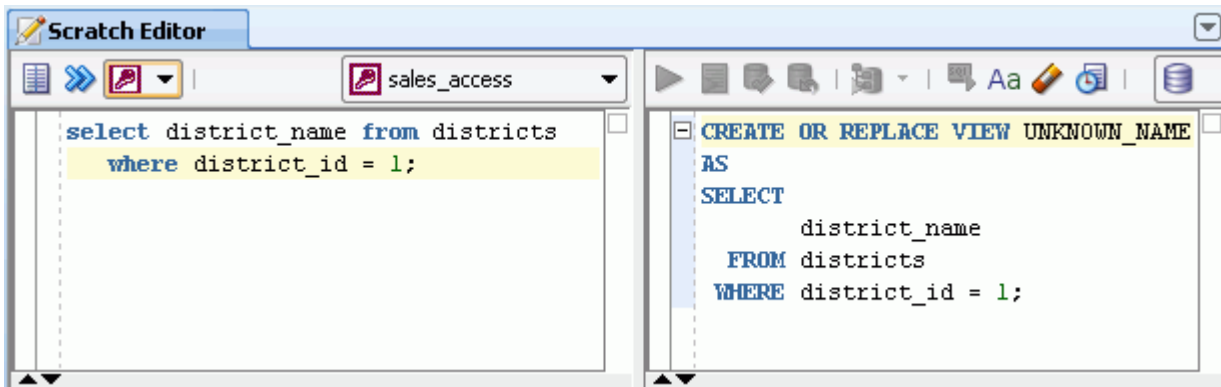
Logging Page: Contains an entry for each migrated-related operation.

Data Editor Log: Contains entries when data is being manipulated by SQL Developer. For example, the output of a Microsoft Excel import operation will be reported here as a series of INSERT statements.

2.14.5 Using the Translation Scratch Editor

You can use the translation scratch editor to enter third-party database SQL statements and have them translated to Oracle PL/SQL statements. You can specify translation from Microsoft SQL Server T-SQL to PL/SQL, from Sybase T-SQL to PL/SQL, or from Microsoft Access SQL to PL/SQL.

You can display the scratch editor by clicking **Tools**, then **Migration**, then **Translation Scratch Editor**. The scratch editor consists of two SQL Worksheet windows side by side, as shown in the following figure:



To translate a statement to its Oracle equivalent, select the type of translation, enter the third-party SQL statement or statements, then click the **Translate (>>)** icon to display the generated PL/SQL statement or statements.

SQL keywords are automatically highlighted.

Note: For a Microsoft SQL Server or Sybase Adaptive Server connection, the worksheet does not support running T-SQL statements. It only supports SELECT, CREATE, INSERT, UPDATE, DELETE, and DROP statements.

The first time you save the contents of either worksheet window in the translation scratch editor, you are prompted for the file location and name. If you perform any subsequent **Save** operations (regardless of whether you have erased or changed the content of the window), the contents are saved to the same file. To save the contents to a different file, click **File**, then **Save As**.

For detailed information about the worksheet windows, see [Section 1.7, "Using the SQL Worksheet"](#).

Unit Testing with SQL Developer

The SQL Developer **unit testing** feature provides a framework for testing PL/SQL objects, such as functions and procedures, and monitoring the results of such objects over time. You create tests, and for each you provide information about what is to be tested and what result is expected. The SQL Developer implementation of unit testing is modeled on the classic and well known xUnit collection of unit test frameworks.

The unit testing feature is part of the support within the SQL Developer family of products for major parts of the life cycle of database system development, from design (provided by Data Modeler) to development to testing.

This chapter contains the following major sections:

[Section 3.1, "Overview of Unit Testing"](#)

[Section 3.2, "SQL Developer User Interface for Unit Testing"](#)

[Section 3.3, "Unit Test Repository"](#)

[Section 3.4, "Editing and Running a Unit Test"](#)

[Section 3.5, "Using a Dynamic Value Query for Seed Data"](#)

[Section 3.6, "Using Lookups to Simplify Unit Test Creation"](#)

[Section 3.7, "Using Variable Substitution in Validation Actions"](#)

[Section 3.8, "Unit Test Library"](#)

[Section 3.9, "Unit Test Reports"](#)

[Section 3.10, "Exporting and Importing Unit Test Objects"](#)

[Section 3.11, "Using the Command-Line Interface"](#)

[Section 3.12, "Example of Unit Testing \(Tutorial\)"](#)

3.1 Overview of Unit Testing

The SQL Developer unit testing framework involves a set of sequential steps for each test case. The steps are as follows, including the user input for before the step is run and the framework activities for the step while the test is being run.

1. Identify the object to be tested.

User Input: Identify the object, such as a specific PL/SQL procedure or function.

Framework Activities: Select the object for processing.

2. Perform any startup processing.

User Input: Enter the PL/SQL block, or enter NULL for no startup processing.

Framework Activities: Execute the block.

3. Run the unit test object.

User Input: (None.)

Framework Activities: Execute the unit test.

4. Check and record the results.

User Input: Identify the expected return (result), plus any validation rules.

Framework Activities: Check the results, including for any validation, and store the results.

5. Perform any end processing (teardown).

User Input: Enter the PL/SQL block, or enter NULL for no teardown activities.

Framework Activities: Execute the block.

For each test, you enter the information called for in the preceding steps, to create a **test case**. A **unit test** is a group of test cases (one or more) on a specific PL/SQL object.

Each test case is an **implementation**. Each unit test has at least one implementation (named *Default* by default); however, you can add one or more other implementations. For example, you can have implementations that test various combinations of parameter values, including those that generate exceptions.

You can group unit tests into a **test suite** to be run as a grouped item, and the test suite can have its own startup and end processing in addition to any specified for test cases and unit tests.

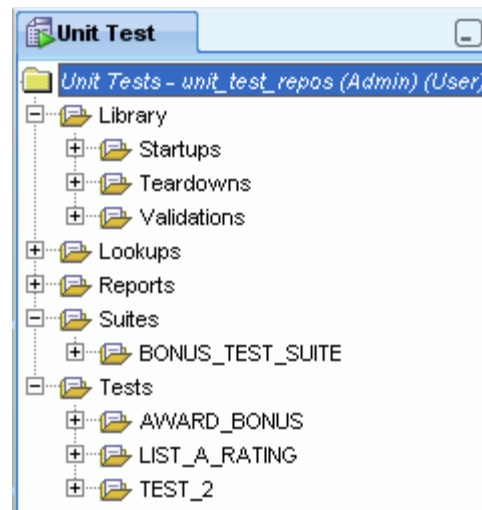
To learn more about unit testing with SQL Developer, take whichever approach suits your preference:

- Go to [Section 3.12, "Example of Unit Testing \(Tutorial\)"](#) and follow the steps, and then return to read the remaining conceptual information under [Unit Testing with SQL Developer](#).
- Read the remaining conceptual information under [Unit Testing with SQL Developer](#), finishing with [Section 3.12, "Example of Unit Testing \(Tutorial\)"](#).

3.2 SQL Developer User Interface for Unit Testing

The SQL Developer user interface for unit testing includes the Unit Test navigator, the Unit Test submenu, and other features.

[Figure 3–1, "Unit Test Navigator"](#) shows the Unit Test navigator, which includes the top-level nodes Library, Lookups, Reports, Suites, and Tests. (If this navigator is not visible, click **View**, then **Unit Test**.)

Figure 3–1 Unit Test Navigator

In the preceding figure, the top-level repository node shows the name of the connection being used (`unit_test_repos`) and whether the user associated with that connection has only User access to the repository or both Administrator and User access (here, both). (Administrator and User access are explained in [Section 3.3](#).)

The preceding figure also shows the types of actions under the Library node (Startups, Teardowns, Validations), one test suite, and several tests.

3.2.1 Unit Test Submenu

To display the Unit Test submenu, click **Tools**, then **Unit Test**. (The commands on the Unit Test submenu affect the [Unit Test Repository](#).)

Select Current Repository: Enables you to select the database connection to use for the unit testing repository, and to create a repository using that connection if no repository exists in the associated schema.

Deselect Current Repository: Disconnects from the current unit testing repository. To connect again to a unit testing repository (the same one or a different one), use Select Current Repository.

Create/Update Repository: Enables you to create a unit testing repository, to hold schema objects associated with the SQL Developer unit testing feature.

Drop Repository: Drops (deletes) the current unit testing repository.

Purge Repository: Deletes the contents of the current unit testing repository, but does not delete the repository metadata.

Manage Users: Enables you to select, add, and modify database connections to be used for the unit testing repository.

Select As Shared Repository: Makes the current repository a shared repository.

Deselect As Shared Repository: Makes the current repository an unshared repository.

3.2.2 Other Menus: Unit Test Items

The **View** menu has the following item related to unit testing:

- **Unit Test:** Toggles the display of the Unit Test navigator.

3.2.3 Unit Test Preferences

The SQL Developer user preferences window (displayed by clicking **Tools**, then **Preferences**) contains a **Unit Test Parameters** pane.

For information about specific preferences, click **Help** in the pane or see [Section 1.12.13](#).

3.3 Unit Test Repository

The unit test **repository** is a set of tables, views, indexes, and other schema objects that SQL Developer maintains to manage the use of the unit testing feature. (Most of these objects have `UT_` in their names.) You can create a separate database user for a repository or use the schema of an existing database user; but for simplicity and convenience in an environment with a single main shared repository, you may want to create a separate database user (as is done in [Section 3.12, "Example of Unit Testing \(Tutorial\)"](#)).

A repository can be unshared or shared, depending on how many and which database users are allowed to perform various types of unit testing operations:

- In an **unshared repository**, only the database user that owns the unit test repository schema objects can be used for operations than can modify the repository.

There can be multiple unshared repositories, for example, to allow individual developers to create private repositories.

- In a **shared repository**, the owner of the repository objects and any other user that has been granted Administrator access to the repository (specifically, `UT_REPO_ADMINISTRATOR` role) can perform administrative operations, such as managing users.

There can be at most one shared repository, and this is the typical case for a team development environment. A repository administrator can add users (as explained in [Section 3.3.1](#)) and can switch the repository status between shared and unshared. (When a repository is made shared, SQL Developer creates public synonyms for the appropriate repository objects.)

To change an unshared repository to shared, click **Tools**, then **Unit Test**, then **Repository**, then **Select As Shared Repository**. To change a shared repository to unshared, click **Tools**, then **Unit Test**, then **Repository**, then **Deselect As Shared Repository**.

3.3.1 Managing Repository Users and Administrators

To create and run unit tests and suites, you must use a connection for a database user that has been granted User access to the repository (specifically, `UT_REPO_USER` role). To perform repository administrative operations, such as managing users, you must use a connection for a database user that has been granted Administrator access to the repository (specifically, `UT_REPO_ADMINISTRATOR` role).

For example, you may want to allow users `SCOTT`, `JONES`, and `SMITH` to use the unit test capabilities and thus have User access to the shared repository, but to allow only `SYS` and the user that owns the repository objects (such as `UNIT_TEST_REPOS` in [Example of Unit Testing \(Tutorial\)](#)) to have Administrator access to the shared repository.

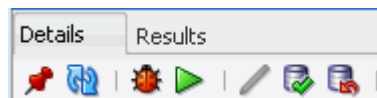
To grant either type of access to any database users, click **Tools**, then **Unit Test**, then **Repository**, then **Manage Users**. Select the database connection for the owner of the

repository objects or for any other user that has been granted Administrator access to the repository. The [Unit Testing: Manage Users](#) dialog box is displayed.

3.4 Editing and Running a Unit Test

To edit or run a unit test, click the unit test name in the Unit Test navigator and select the desired connection for running the unit test. A pane is displayed with two tabs: Details for the unit test specification, and Results for results if you run or debug the test.

The toolbar under the Details tab for the subprogram name has a toolbar that includes the icons shown in the following figure.



- **Freeze View** (the pin) keeps that pane in the SQL Developer window when you click another unit test in the Unit Test navigator; a separate tab and detail view pane are created for that other unit test. If you click the pin again, the unit test's detail view pane is available for reuse.
- **Refresh** refreshes the display in the pane.
- **Debug** starts execution of the first or next implementation of the unit test in debug mode, and displays the results in the Results tab.
- **Run** starts normal execution of the unit test, and displays the results in the Results tab.
- **Edit** (pencil icon) enables you to edit the unit test specification. (If you cannot modify the unit test, lick the Edit icon.)
- **Commit Changes** saves any changes that you have made to the unit test.
- **Rollback Changes** discards any unsaved changes that you have made to the unit test.

If you click the Edit icon, you can modify the Startup Process, Teardown Process, and details for each implementation.

You can also specify **Gather Code Coverage Statistics** to have SQL Developer collect statistics related to code coverage. To view any statistics that have been gathered from unit test runs, use the Test Runs Code Coverage report. In that report, you can click a row with summary information to display detailed information in the Code Coverage Details pane.

3.5 Using a Dynamic Value Query for Seed Data

As an alternative to specifying exact input data when creating a unit test, you can create a dynamic value query to use seed data from a table as input for the test. The query returns values from specified columns in one or more rows, and all sets of values returned are checked by any process validation that you have specified for the test. One common use of dynamic value queries is to perform "reasonableness" tests, such as checking that each salary or price resulting from a test is within a specified range.

To create a test that uses dynamic value queries, create and populate the table to be used by the query, create the test by specifying the object to be tested and any startup

and teardown actions, and specify a validation action (such as a query returning rows or no rows).

The following example assumes that you done at least the following in [Section 3.12, "Example of Unit Testing \(Tutorial\)"](#): created the EMPLOYEES table, created the AWARD_BONUS procedure, and created the unit test repository. It creates a unit test that checks to be sure that no salesperson would receive a bonus so large that his or her salary amount would be greater than 20000. Follow these steps:

1. Create and populate the table for the seed data by executing the following statements:

```
CREATE TABLE award_bonus_dyn_query (emp_id NUMBER PRIMARY KEY, sales_amt
NUMBER);
INSERT INTO award_bonus_dyn_query VALUES (1001, 5000);
INSERT INTO award_bonus_dyn_query VALUES (1002, 6000);
INSERT INTO award_bonus_dyn_query VALUES (1003, 2000);
commit;
```

2. In the Unit Test navigator, right-click the **Tests** node and select **Create Test**.

The [Unit Testing: Create Unit Test](#) wizard is displayed. Click **Next** to go from each step to the next; and when you are finished specifying the unit test, click **Finish**.

3. In [Select Operation](#), select the database connection for the schema that you used to create the AWARD_BONUS procedure; then expand the Procedures node and select AWARD_BONUS.
4. In [Specify Test Name](#), for **Test Name** specify AWARD_BONUS_WITH_SEED_DATA, and select **Seed/create implementations using lookup values**.

(You will not actually be using lookup values in this case, but this choice enables you to pass quickly through the remaining wizard steps before editing the unit test to use a dynamic value query.)

5. In [Specify Startup](#), select **Table or Row Copy** because you want to save the current data values in the EMPLOYEES table before any data is modified by the unit test.

When prompted, for **Source Table** specify EMPLOYEES, and for **Target Table** accept the default name provided for a temporary table that will be automatically created when it is needed and deleted when it is no longer needed.

6. In [Specify Parameters](#), click **Next** to go to the next page.
7. In [Specify Validations](#), click **Next** to go to the next page.
8. In [Specify Teardown](#), select **Table or Row Restore** because you want to restore the original data values in the EMPLOYEES table before any data was modified by the unit test. When prompted, accept the supplied values for **Target Table** (EMPLOYEES) and **Source Table** (the name of the temporary table).

9. In [Summary](#), review the information. If you need to change anything, click **Back** as needed and make the changes, then proceed to this Summary page. When you are ready to complete the unit test definition, click **Finish**.

10. In the Unit Test navigator, click the node for AWARD_BONUS_WITH_SEED_DATA under Tests, to display the test in an editing window.

11. In the Details pane, click the pencil icon next to Dynamic Value Query enter the following, and click OK:

```
SELECT emp_id, sales_amt FROM award_bonus_dyn_query;
```

12. For **Expected Result**, leave the value as **Success**.

13. In [Specify Validations](#), click the plus (+) icon and select **Query returning no rows**.

For the query, replace the SELECT statement in the Process Validation box with the following (any semicolon at the end of the statement is ignored):

```
SELECT * FROM employees WHERE salary_amt > 20000
AND commission_pct IS NOT NULL
```

That is, for all salespersons (employees whose commission percentage is not null), check whether the salary resulting from the unit test run is greater than 20000. If there are no such salespersons (that is, the query returns no rows), the result of the validation action is success.

14. Run the AWARD_BONUS_WITH_SEED_DATA unit test. (For the basic steps to run a unit test, see [Section 3.12.5](#).)

3.6 Using Lookups to Simplify Unit Test Creation

A lookup is an object that contains, for one or more data types, data values that can be tested. Lookups are mainly used for the following purposes:

- Providing lists of values (dropdown lists) for Input fields, as explained in [Section 3.6.1](#).
- Automatically creating test implementations based on lookup values, as explained in [Section 3.6.2](#).

To create a lookup:

1. In the Unit Test navigator, right-click the **Lookups** node and select **Add Category**.
2. Specify the category name (for example, EMP_ID_LOOKUP).
3. For each data type for which you want to specify lookup values (that is, valid and invalid data values for possible testing), right-click the category name and select **Add Datatype**, select the data type, and use the + (plus sign) icon to add as many data values as you want.

Note that (null) is automatically included in the list of values for each data type for each lookup that you create.

For example, for the environment described in [Section 3.12, "Example of Unit Testing \(Tutorial\)"](#), you could create lookups named EMP_ID_LOOKUP and SALES_AMT_LOOKUP. Each would have only one data type: NUMBER. For the NUMBER data for each lookup, use the + (plus sign) icon to add each of the following values on a separate line, and click the Commit Changes icon or press F11 when you are finished entering the set of numbers for each lookup:

- For EMP_ID_LOOKUP: -100, 99, 1001, 1002, 1003, 1004, 2000, 9999
- For SALES_AMT_LOOKUP: -1000, 0, 1000, 2000, 5000, 6000, 10000, 99999

You can delete and rename lookup categories by using the context (right-click) menu in the Unit Test navigator. You can also delete a data type under a lookup category; however, "deleting" in this case removes any currently specified data values for that type for the lookup category, and it makes the type available for selection in the [Unit Testing: Add Data Type](#) dialog box.

3.6.1 Providing Values for Input Fields

When you are specifying Input parameters for a unit test implementation, you can click the Lookup Category control to select a lookup category. When you then click in

a cell under Input, you can click the dropdown arrow to select a value from the specified lookup. (You can also enter a value other than one in the list.)

For example, if you created the EMP_ID_LOOKUP lookup category as explained in [Section 3.6](#), and if you select it as the lookup category when specifying parameters, then the values -100, 99, 1001, 1002, 1003, 1004, 2000, 9999, and (null) will be in the dropdown list for the Input cell for the EMP_ID parameter. (For the SALES_AMT parameter, use the SALES_AMT_LOOKUP category.)

3.6.2 Automatically Creating Implementations

If you know that you want implementations to test certain values for a data type, you can use a lookup category to generate these implementations automatically instead of creating them all manually. To do this, use either the DEFAULT lookup category or a user-created category, specify the values for the desired data type, then specify that lookup category for the **Configuration set to use for lookups** preference in the [Unit Test Parameters](#) preferences.

For example, assume that for NUMBER input parameters, you always want to check for a very high positive number (such as 9999), a very low negative number (such as -9999), 1, -1, and 0 (zero). Follow these steps:

1. In the Unit Test navigator, expand the **Lookups** node.
2. Right-click DEFAULT and select Add Datatype.
3. In the dialog box, specify NUMBER.
4. In the Lookups Editor for the NUMBER type, use the + (plus sign) icon to add each of the following as a separate item (new line).


```
9999
1.0
0
-1.0
-9999
```
5. Click the Commit Changes icon or press F11.
6. Click **Tools**, then **Preferences**, then **Unit Test Parameters**, and ensure that the configuration set to use for lookups is DEFAULT (the lookup category for which you just specified the values for the NUMBER data type).
7. Create the unit test in the usual way: in the Unit Test navigator, right-click the **Tests** node and select **Create Test**.

However, in the [Specify Test Name](#) step, select **Seed/Create implementations using lookup values** (that is, *not* "Create with single dummy representation").

For [Specify Startup](#) and [Specify Teardown](#), specify any desired action.

You cannot specify anything for [Specify Parameters](#) or [Specify Validations](#) now. An implementation (with a name in the form `Test Implementation n`) will automatically be created for each possible combination of input parameters of type NUMBER. For any validation actions, you must specify them later by editing each generated implementation.

3.7 Using Variable Substitution in Validation Actions

You can use variable substitution in validation actions to write dynamic validations that provide a result based on the values of input and output parameters of a

procedure or function, or on the return value of a function. You can specify strings in the following format in validation actions:

- For input parameters: {*PARAMETER_NAME*}

For example, if an input parameter is named EMP_ID:

```
SELECT ... WHERE employee_id = {EMP_ID} AND ...;
```

- For output parameters: {*PARAMETER_NAME*\$}

For example, if an output parameter is named SALARY:

```
SELECT ... WHERE {SALARY$} < old_salary;
```

- For the return value: {*RETURNS*\$}

For example, if a function returns a numeric value:

```
SELECT ... WHERE {RETURNS$} > 1;
```

What is actually substituted is the string representation of the parameter value. For example:

- If P1 is a parameter of type NUMBER and has the value 2.1, the string {P1} will be replaced by the string 2.1.
- If P1 is a parameter of type VARCHAR2 and has the value ABC, the string '{P1}' will be replaced by the string 'ABC'. (Note the single-quotation marks around {P1} in this example.)

You can use variable substitution for all types of validation actions except Compare Tables. For the applicable validation action types, variable substitution is performed as follows:

- For Query Returning Row(s) and Query Returning No Row(s), substitution is performed on the SQL query.
- For Compare Query Results, substitution is performed on both the source and target SQL queries.
- For Boolean Function and User PL/SQL Code, substitution is performed on the PL/SQL block.

3.8 Unit Test Library

The unit testing library enables you to store actions that you can reuse in the definitions of multiple unit tests. These user-defined actions are displayed under the Library node in the Unit Test navigator (which is explained in [Section 3.2](#)). You can store the following kinds of actions in the library, in the following categories:

- Dynamic value queries
- Startup actions
- Teardown actions
- Validation actions

Most categories have subcategories. For example, the *Startup Actions* node has subnodes for *Table or Row Copy* and *User PL/SQL Code*. You can add an entry to the library in the following ways:

- Expand the Library hierarchy to display the relevant lowest-level node (such as *User PL/SQL Code* under *Startups*); right-click and select **Add [action-type]**; specify

a name for the action; click the name of the newly created action; and complete the specification.

- Use the **Publish to Library** option when specifying the action when you are creating a unit test: enter a name for the action and click Publish. (The action will be added under the appropriate category and subcategory in the Library display in the Unit Test navigator.)

To use an action from the library when you are creating a unit test, select it from the list under **Library** on the appropriate page in the [Unit Testing: Create Unit Test](#) wizard.

3.9 Unit Test Reports

Several SQL Developer reports provide information about operations related to unit testing. These reports are listed in the Unit Test navigator under the Reports node. The available reports include:

- All Suite Runs
- All Test Implementation Runs
- All Test Runs
- Suite Runs Code Coverage
- Suite Test Implementation Runs
- Suite Test Runs
- Test Implementation Runs
- Test Runs Code Coverage
- User Test Runs (test runs grouped by user)

Each unit testing report contains a top pane with a summary information row for each item. To see detailed information about any item, click in its row to display the information in one or more detail panes below the summary information. For example, if you click in a summary row in the All Test Runs report, details about that test run are displayed under the Test Run Details and Most Recent Code Coverage tabs.

Some reports prompt you for bind variables, where you can accept the default values to display all relevant items or enter bind variables to restrict the display. (For more information, see [Bind Variables for Reports](#).)

3.10 Exporting and Importing Unit Test Objects

You can export and import unit tests, suites, and objects that are stored in the library (such as startup, validation, and teardown actions).

Exporting an object causes all dependent objects to be included in the resulting XML file. For example, if you export a suite, the resulting XML file includes all tests in that suite, as well as all startup, validation, and teardown actions within each test in that suite.

To export an object, right-click its name in the Unit Test navigator and select **Export to File**; then specify the location and name for the XML file that will include the definitions of the objects.

Importing unit test objects from an XML file causes all objects in the file to be created in the appropriate places in the Unit Test navigator hierarchy. If an object already

exists in the repository with the same name as an object of the same type in the XML file, it is replaced (overwritten) by the object definition in the XML file.

To import unit test objects, click **Tools**, then **Unit Test**, then **Import from File**; then specify the XML file to be used for the import operation.

3.11 Using the Command-Line Interface

In addition to running unit tests and suites, and exporting and importing unit test objects, within the SQL Developer graphical interface, you use the UtUtil batch file (Windows) or shell script (Linux) on the operating system command line. UtUtil is located in the `sqldeveloper\sqldeveloper\bin` folder or `sqldeveloper/sqldeveloper/bin` directory under the location where you installed SQL Developer.

UtUtil accepts these commands: `run` to run a test or suite, `exp` to perform an export operation, and `imp` to perform an import operation. For detailed information about the syntax and options, start by running UtUtil without any parameters at the system command prompt. For example:

```
C:\Program Files\sqldeveloper\sqldeveloper\bin>UtUtil

UtUtil -run ?
UtUtil -exp ?
UtUtil -imp ?
```

Then enter the command for information about the command that you want to use. For example: `UtUtil -run ?`

The following example runs a unit test named `AWARD_BONUS` in a Windows environment where SQL Developer is installed under Program Files. (Note that test and suite names are case sensitive for the command-line interface.) This example uses the repository connection for user `unit_test_repos` and runs the test as user `fred`.

```
> cd c:\Program Files\sqldeveloper\sqldeveloper\bin
> UtUtil -run -test -name AWARD_BONUS -repo unit_test_repos -db fred
```

The following example exports a unit test named `AWARD_BONUS`. It uses the repository connection for user `unit_test_repos` and stores the exported definitions in the file `C:\ut_xml\award_bonus_test.xml`.

```
> UtUtil -exp -test -name AWARD_BONUS -repo unit_test_repos -file c:\ut_xml\award_
bonus_test.xml
```

The following example imports object definitions from the file `C:\ut_xml\award_bonus_suite.xml`. It uses the repository connection for user `unit_test_repos`.

```
> UtUtil -imp -repo unit_test_repos -file c:\ut_xml\award_bonus_suite.xml
```

To check the results of any tests or suites that you run from the command line, you can start SQL Developer and view the All Test Runs and All Suite Runs reports (see [Section 3.9, "Unit Test Reports"](#)).

3.12 Example of Unit Testing (Tutorial)

This section presents a simplified example in which you create a table and a PL/SQL procedure, create unit tests with test cases for valid and invalid input data, run the unit tests, and create and run a unit test suite. It assumes that you have a table of employee data that includes salary information, and that you need to create a

procedure to award bonuses to sales representatives, whose pay consists of a base salary plus a commission-based bonus.

Note: An Oracle By Example (OBE) tutorial, *Performing a Unit Test of Your PL/SQL in Oracle SQL Developer 2.1*, is similar to this one, but it uses a copy of the EMPLOYEES table from the Oracle sample HR schema, which includes more columns and rows as well as different data. For information about SQL Developer OBEs, see the Start Page (click **Help**, then **Start Page**).

The EMPLOYEES table includes the following columns, all of type NUMBER:

- EMPLOYEE_ID: Employee identification (badge) number.
- COMMISSION_PCT: Commission percentage for the employee: a decimal fraction representing the percentage of the amount of sales by the employee, to be used to compute a bonus that will be added to the employee's base salary to determine the total salary. For example, 0.2 or .2 indicates a 20 percent commission, or 0.2 times the amount of sales.

Only employees in the Sales department have numeric COMMISSION_PCT values. Other employees (not "on commission") have null COMMISSION_PCT values.

- SALARY: Salary amount for the employee; includes base salary plus any bonus (which will be calculated by an award_bonus procedure, to be created during this example).

Assume that the following data exists in these columns in the EMPLOYEES table:

EMPLOYEE_ID	COMMISSION_PCT	SALARY
1001	0.2	8400
1002	0.25	6000
1003	0.3	5000
1004	(null)	10000

You create a procedure named AWARD_BONUS, which has two input parameters:

- emp_id: The employee ID of an employee.
- sales_amt: The amount of sales with which the employee is credited for the period in question.

This amount is calculated using the COMMISSION_PCT value for the specified employee, and the result is added to the SALARY value for that employee.

If the COMMISSION_PCT is null for the employee, no commission or bonus can be calculated, and an exception is raised. This scenario occurs if an attempt is made to add a commission-based bonus to the salary of an employee who is not in the Sales department.

The rest of this example involves the following major steps:

1. [Create the EMPLOYEES Table.](#)
2. [Create the AWARD_BONUS Procedure.](#)
3. [Create the Unit Testing Repository.](#)

4. [Create a Unit Test.](#)
5. [Run the Unit Test.](#)
6. [Create and Run an Exception Unit Test.](#)
7. [Create a Unit Test Suite.](#)
8. [Run the Unit Test Suite.](#)

3.12.1 Create the EMPLOYEES Table

This tutorial uses a table named EMPLOYEES, which must exist before you run any unit tests of the AWARD_BONUS procedure. This table contains some of the columns used in the HR.EMPLOYEES table that is included in the Oracle-supplied sample schemas, but it does not contain all of the columns, and it contains fewer rows and different data.

You can create this EMPLOYEES table in an existing schema and using an existing database connection, or you can create a new schema and connection for the table. To create and populate this table, enter the following statements in a SQL Worksheet or a SQL*Plus command window:

```
-- Connect as the database user that will be used to run the unit tests.
-- Then, enter the following statements:

CREATE TABLE employees (employee_id NUMBER PRIMARY KEY, commission_pct NUMBER,
salary NUMBER);
INSERT INTO employees VALUES (1001, 0.2, 8400);
INSERT INTO employees VALUES (1002, 0.25, 6000);
INSERT INTO employees VALUES (1003, 0.3, 5000);
-- Next employee is not in the Sales department, thus is not on commission.
INSERT INTO employees VALUES (1004, null, 10000);
commit;
```

3.12.2 Create the AWARD_BONUS Procedure

Create the AWARD_BONUS procedure in the same schema as the EMPLOYEES table. In a SQL Worksheet using the appropriate database connection, enter the following text:

```
create or replace
PROCEDURE award_bonus (
  emp_id NUMBER, sales_amt NUMBER) AS
  commission REAL;
  comm_missing EXCEPTION;
BEGIN
  SELECT commission_pct INTO commission
  FROM employees
  WHERE employee_id = emp_id;

  IF commission IS NULL THEN
    RAISE comm_missing;
  ELSE
    UPDATE employees
    SET salary = salary + sales_amt*commission
    WHERE employee_id = emp_id;
  END IF;
END award_bonus;
/
```

Click the Run Script icon (or press F5) to create the AWARD_BONUS procedure.

3.12.3 Create the Unit Testing Repository

You will need a unit testing repository in the database to hold schema objects that you create and that SQL Developer will maintain. You can create a separate database user for this repository or use the schema of an existing database user; however, to simplify your learning and any possible debugging you may need to do later, it is recommended that you use a separate schema for the unit testing repository, and the instructions in this section reflect this approach.

1. Create a database user (for example, UNIT_TEST_REPOS) for the unit testing repository. Using a database connection with DBA privileges, right-click **Other Users** in the Connections navigator and select **Create User**. Specify UNIT_TEST_REPOS as the user name, and complete any other required information.

For **Default Tablespace**, specify USERS; for **Temporary Tablespace**, specify TEMP.

For **System Privileges**, enable CREATE SESSION; then click **Apply**, then **Close**.

2. Create a database connection for the unit testing repository user that you created, as follows. Click **Tools**, then **Unit Test**, then **Manage Users**. In the **Select Connection** dialog box, click the plus (+) icon to create a new database connection (for example, unit_test_repos) for the unit testing repository user.

Click **Save** to save the connection, then **Cancel** to close the dialog box.

3. Create the repository in the schema of the user that you created, as follows. In the User Manager Connection dialog box, select the database connection that you just created (for example, unit_test_repos) and click **OK**.

SQL Developer will display several prompts so it can execute commands that grant the necessary privileges to the unit test repository user. In each case, click **Yes**, and enter the SYS account password when prompted.

3.12.4 Create a Unit Test

To create the first unit test, use the Unit Test navigator. If this navigator is not visible on the left side, click **View**, then **Unit Test**. The Unit Test navigator is described in [Section 3.2](#).

1. In the Unit Test navigator, right-click the **Tests** node and select **Create Test**.
The **Unit Testing: Create Unit Test** wizard is displayed. In the remaining steps, click **Next** to go from each step to the next; and when you are finished specifying the unit test, click **Finish**.
2. In **Select Operation**, select the database connection for the schema that you used to create the AWARD_BONUS procedure; then expand the Procedures node and select AWARD_BONUS.
3. In **Specify Test Name**, for **Test Name** specify AWARD_BONUS (same as the procedure name), and select **Create with single dummy representation**.
4. In **Specify Startup**, select **Table or Row Copy** because you want to save the current data values in the EMPLOYEES table before any data is modified by the unit test.

When prompted, for **Source Table** specify EMPLOYEES, and for **Target Table** accept the default name provided for a temporary table that will be automatically created when it is needed and deleted when it is no longer needed. (The target

table will be created; and if a table already exists with the name that you specify as the target table, *it will be overwritten.*)

5. In [Specify Parameters](#), change the values in the Input column to the following:

For Parameter EMP_ID: 1001

For Parameter SALES_AMT: 5000

For **Expected Result**, leave the value as `Success`.

6. In [Specify Validations](#), click the plus (+) icon and select **Query returning row(s)**.

For the query, replace the SELECT statement in the Process Validation box with the following (any semicolon at the end of the statement is ignored):

```
SELECT * FROM employees
WHERE employee_id = 1001 AND salary = 9400
```

That is, because employee 1001 has a 20 percent (0.2) commission and because the sales amount was specified as 5000, the bonus is 1000 (5000 * 0.2), and the new salary for this employee is 9400 (8400 base salary plus 1000 bonus). In this case, the query returns one row, and therefore the result of the validation action is success.

Note that you could have instead specified the SELECT statement in this step using variable replacement (explained in [Section 3.7](#)), as follows:

```
SELECT * FROM employees
WHERE employee_id = {EMP_ID} AND salary = 9400
```

However, in this specific example scenario, using variable substitution would provide no significant advantage.

7. In [Specify Teardown](#), select **Table or Row Restore** because you want to restore the original data values in the EMPLOYEES table before any data was modified by the unit test. When prompted, accept the supplied values for **Target Table** (EMPLOYEES) and **Source Table** (the name of the temporary table).
8. In [Summary](#), review the information. If you need to change anything, click **Back** as needed and make the changes, then proceed to this Summary page. When you are ready to complete the unit test definition, click **Finish**.

3.12.5 Run the Unit Test

To run the unit test, use the Unit Test navigator. If this navigator is not visible on the left side, click **View**, then **Unit Test**. The Unit Test navigator is described in [Section 3.2](#).

1. In the Unit Test navigator, expand the **Tests** node and click the AWARD_BONUS test.

A pane for the AWARD_BONUS test is displayed, with Details and Results tabs.

2. On the Details tab, near the top-right corner, select the database connection for the schema that you used to create the AWARD_BONUS procedure.

Do not change any other values. (However, if you later want to run the unit test with different specifications or data values, you can click the Edit (pencil) icon in the Code Editor toolbar at the top of the pane.)

3. Click the Run Test (green arrowhead) icon in the Code Editor toolbar (or press F9).

At this point, focus is shifted to the Results tab, where you can soon see that the AWARD_BONUS ran successfully.

If you want to check the EMPLOYEES table data, you will see that the salary for employee 1001 is the same as it was before (8400), because the startup action for the unit test copied the original data to the temporary table and the teardown action restored the original data to the EMPLOYEES table.

3.12.6 Create and Run an Exception Unit Test

Create another unit test for the exception condition where the COMMISSION_PCT value is null for the employee, and therefore no commission or bonus can be calculated. For this tutorial, the test data includes employee 1004 with a null commission percentage. (This condition could result from several possible scenarios, the most likely being an attempt to run the procedure on a salaried employee who is not eligible for commissions.)

The steps for creating this exception unit test are similar to those in [Section 3.12.4](#), except there are no startup or teardown steps because this test should not modify any table data, and there is no need for any validation action.

1. In the Unit Test navigator, right-click the **Tests** node and select **Create Test**.
The **Unit Testing: Create Unit Test** wizard is displayed. Click **Next** to go from each step to the next; and when you are finished specifying the unit test, click **Finish**.
2. In **Select Operation**, select the database connection for the schema that you used to create the AWARD_BONUS procedure; then expand the Procedures node and select AWARD_BONUS.
3. In **Specify Test Name**, for **Test Name** specify AWARD_BONUS_NO_COMM_EXC, and select **Create with single dummy representation**.
4. In **Specify Startup**, click **Next** to go to the next page.
5. In **Specify Parameters**, change the values in the Input column to the following:
EMP_ID: 1004
SALES_AMT: 5000
For **Expected Result**, change the value to `Exception` and leave the expected error number as `ANY`.
6. In **Specify Validations**, click **Next** to go to the next page.
7. In **Specify Teardown**, click **Next** to go to the next page.
8. In **Summary**, review the information. If you need to change anything, click **Back** as needed and make the changes, then proceed to this Summary page. When you are ready to complete the unit test definition, click **Finish**.

To run this unit test, follow the steps in [Section 3.12.5](#), except specify AWARD_BONUS_NO_COMM_EXC instead of AWARD_BONUS.

On the Results tab, you will see that the AWARD_BONUS_NO_COMM_EXC test ran successfully; and if you check the EMPLOYEES table data, you will see that the information for employee 1004 (and all the other employees) was not changed.

Design Consideration: As an alternative to creating a separate unit test for the exception condition, you could add it as an **implementation** to the AWARD_BONUS test (right-click AWARD_BONUS and select **Add Implementation**). Thus, the AWARD_BONUS unit test would have two implementations: the "Default" implementation using employee 1001, and the AWARD_BONUS_NO_COMM_EXC implementation using employee 1004.

The approach in this tutorial enables you to create a simple unit test suite using the two unit tests (see [Section 3.12.7](#)). However, in more realistic unit testing scenarios, it is probably better to use a unit test for each procedure, add implementations for each test case for a procedure, and group multiple unit tests (for individual procedures) into one or more test suites.

3.12.7 Create a Unit Test Suite

Create a unit test suite that groups together the two unit tests of the AWARD_BONUS procedure. If the Unit Test navigator is not visible on the left side, click **View**, then **Unit Test**. The Unit Test navigator is described in [Section 3.2](#).

1. In the Unit Test navigator, right-click the **Suites** node and select **Add Suite**.
2. In the [Unit Testing: Add Test Suite](#) dialog box, specify AWARD_BONUS_SUITE as the suite name.
3. In the Unit Test navigator, under Suites, click the AWARD_BONUS_SUITE node. An pane for the AWARD_BONUS_SUITE test suite is displayed.
4. Do not specify a Startup Process or Teardown Process, because neither is needed for this test suite.
5. Click the Add (+) icon to add the first test to the suite.
6. In the [Unit Testing: Add Tests to Suite](#) dialog box, click (select) AWARD_BONUS, check (select) **Run Test Startups** and **Run Test Teardowns** so that the startup and teardown actions for that unit test will be run, and click **OK**.
7. Click the Add (+) icon to add the next test to the suite.
8. In the [Unit Testing: Add Tests to Suite](#) dialog box, click (select) AWARD_BONUS_NO_COMM_EXC, and click **OK**. (The check **Run Test Startups** and **Run Test Teardowns** options are irrelevant here because the AWARD_BONUS_NO_COMM_EXC test does not perform any startup and teardown actions.)
9. Click the Commit Changes icon in the Code Editor toolbar at the top of the pane (or press F11).

3.12.8 Run the Unit Test Suite

To run the unit test suite, use the Unit Test navigator. If you are in the editing pane for the AWARD_BONUS_SUITE test suite, run the suite by clicking the Run Suite (green arrowhead) icon in the Code Editor toolbar. Otherwise, perform the following steps:

1. In the Unit Test navigator, expand the **Suites** node and click the AWARD_BONUS_SUITE test suite. A pane for the AWARD_BONUS_SUITE test is displayed, with Details and Results tabs.

2. In the Details tab, near the top-right corner, select the database connection for the schema that you used to create the AWARD_BONUS procedure.

Do not change any other values. (However, if you later want to run the unit test suite with different specifications, you can click the Edit (pencil) icon in the Code Editor toolbar at the top of the pane.)

3. Click the Run Suite (green arrowhead) icon in the Code Editor toolbar (or press F9).

After the suite is run, focus is shifted to the Results tab, where you can soon see that the AWARD_BONUS_SUITE test suite ran successfully.

Tutorial: Creating Objects for a Small Database

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

Note: Other SQL Developer tutorials, including Oracle By Example (OBE) lessons, are available from the Start Page. If the tab for that page is not visible, click **Help**, then **Start Page**.

The tables are deliberately oversimplified for this tutorial. They would not be adequate for any actual public or organizational library. For example, this library contains only books (not magazines, journals, or other document formats), and it can contain no more than one copy of any book.

You will perform the following major steps:

1. [Create a Table \(BOOKS\)](#).
2. [Create a Table \(PATRONS\)](#).
3. [Create a Table \(TRANSACTIONS\)](#).
4. [Create a Sequence](#).
5. [Insert Data into the Tables](#).
6. [Create a View](#).
7. [Create a PL/SQL Procedure](#).
8. [Debug a PL/SQL Procedure](#) (optional).
9. [Use the SQL Worksheet for Queries](#) (optional).

Note: To delete the objects that you create for this tutorial, you can use the DROP statements at the beginning of the script in [Section 4.10](#), "Script for Creating and Using the Library Tutorial Objects".

Related Topics

[Section 4.10, "Script for Creating and Using the Library Tutorial Objects"](#)

[Chapter 1, "SQL Developer Concepts and Usage"](#)

[Section 1.2, "SQL Developer User Interface"](#)

Section 1.3, "Database Objects"

4.1 Create a Table (BOOKS)

The BOOKS table contains a row for each book in the library. It includes columns of character and number types, a primary key, a unique constraint, and a check constraint. You will use the Create Table dialog box to create the table declaratively; the table that you create will be essentially the same as if you had entered the following statement using the SQL Worksheet:

```
CREATE TABLE books (
  book_id VARCHAR2(20),
  title VARCHAR2(50)
    CONSTRAINT title_not_null NOT NULL,
  author_last_name VARCHAR2(30)
    CONSTRAINT last_name_not_null NOT NULL,
  author_first_name VARCHAR2(30),
  rating NUMBER,
  CONSTRAINT books_pk PRIMARY KEY (book_id),
  CONSTRAINT rating_1_to_10 CHECK (rating IS NULL OR
    (rating >= 1 and rating <= 10)),
  CONSTRAINT author_title_unique UNIQUE (author_last_name, title));
```

To create the BOOKS table, connect to the database as the user in the schema you want to use for this tutorial. Right-click the Tables node in the schema hierarchy on the left side, select **New Table**, and enter the following information. (If a tab or field is not mentioned, do not enter anything for it. Be sure that the Advanced box is not checked when you start creating the table.)

For detailed information about the table dialog box and its tabs, see [Section 5.35, "Create Table \(quick creation\)"](#) and [Section 5.36, "Create/Edit Table \(with advanced options\)"](#).

Schema: Specify your current schema as the schema in which to create the table.

Name: BOOKS

Create the table columns using the following information. After creating each column except the last one (rating), click **Add Column** to add the next column. (If you accidentally click OK instead of Add Column, right-click the BOOKS table in the Connections navigator display, select Edit, and continue to add columns.)

Column Name	Type	Size	Other Information and Notes
book_id	VARCHAR2	20	Primary Key (Automatically checks Not Null; an index is also created on the primary key column. This is the Dewey code or other book identifier.)
title	VARCHAR2	50	Not Null
author_last_name	VARCHAR2	30	Not Null
author_first_name	VARCHAR2	30	
rating	NUMBER		(Librarian's personal rating of the book, from 1 (poor) to 10 (great))

After you have entered the last column (rating), check **Advanced** (next to Schema). This displays a pane for more table options. For this table, you will use the Unique Constraints and Check Constraints panes.

Unique Constraints pane

Click **Add** to add a unique constraint for the table, namely, that the combination of author_last_name and title must be unique within the table. (This is deliberately oversimplified, since most major libraries will have allow more than one copy of a book in their holdings. Also, the combination of last name and title is not always a "foolproof" check for uniqueness, but it is sufficient for this simple scenario.)

Name: author_title_unique

In **Available Columns**, double-click TITLE and then AUTHOR_LAST_NAME to move them to Selected Columns.

Check Constraints pane

Click **Add** to add a check constraint for the table, namely, that the rating column value is optional (it can be null), but if a value is specified, it must be a number from 1 through 10. You must enter the condition using SQL syntax that is valid in a CHECK clause (but do not include the CHECK keyword or enclosing parentheses for the entire CHECK clause text).

Name: rating_1_to_10

Condition: rating is null or (rating >= 1 and rating <= 10)

Click **OK** to finish creating the table.

Go to [Section 4.2, "Create a Table \(PATRONS\)"](#) to create the next table.

4.2 Create a Table (PATRONS)

The PATRONS table contains a row for each patron who can check books out of the library (that is, each person who has a library card). It includes an object type (MDSYS.SDO_GEOMETRY) column. You will use the Create Table dialog box to create the table declaratively; the table that you create will be essentially the same as if you had entered the following statement using the SQL Worksheet:

```
CREATE TABLE patrons (
  patron_id NUMBER,
  last_name VARCHAR2(30)
  CONSTRAINT patron_last_not_null NOT NULL,
  first_name VARCHAR2(30),
  street_address VARCHAR2(50),
  city_state_zip VARCHAR2(50),
  location MDSYS.SDO_GEOMETRY,
  CONSTRAINT patrons_pk PRIMARY KEY (patron_id));
```

The use of single city_state_zip column for all that information is not good database design; it is done here merely to simplify your work in the tutorial.

The location column (Oracle Spatial geometry representing the patron's geocoded address) is merely to show the use of a complex (object) type.

To create the PATRONS table, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Tables node in the schema hierarchy on the left side, select **New Table**, and enter the following information. (If a tab or field is not mentioned, do not enter anything for it. Be sure that the Advanced box is not checked when you start creating the table.)

Schema: Specify your current schema as the schema in which to create the table.

Name: PATRONS

Create most of the table columns using the following information. After creating each column except the city_state_zip column, click **Add Column** to add the next column. (If you accidentally click OK instead of Add Column, right-click the PATRONS table in the Connections navigator display, select Edit, and continue to add columns.)

Column Name	Type	Size	Other Information and Notes
patron_id	NUMBER		Primary Key. (Unique patron ID number, with values to be created using a sequence that you will create)
last_name	VARCHAR2	30	Not Null
first_name	VARCHAR2	30	
street_address	VARCHAR2	30	
city_state_zip	VARCHAR2	30	

The last column in the table (location) requires a complex data type, for which you must use the Columns tab with advanced options. Check **Advanced** (next to Schema). This displays a pane for selecting more table options.

In the Columns pane, click the city_state_zip column name, and click the Add Column (+) icon to add the following as the last column in the table.

Column Name	Type	Other Information and Notes
location	Complex type Schema: MDSYS Type: SDO_GEOMETRY	(Oracle Spatial geometry object representing the patron's geocoded address)

After you have entered the last column (location), click **OK** to finish creating the table.

Go to [Section 4.3, "Create a Table \(TRANSACTIONS\)"](#) to create the next table.

4.3 Create a Table (TRANSACTIONS)

The TRANSACTIONS table contains a row for each transaction involving a patron and a book (for example, someone checking a book out or returning a book). It includes two foreign key columns. You will use the Create Table dialog box to create the table declaratively; the table that you create will be essentially the same as if you had entered the following statement using the SQL Worksheet:

```
CREATE TABLE transactions (
  transaction_id NUMBER,
  patron_id CONSTRAINT for_key_patron_id
    REFERENCES patrons (patron_id),
  book_id CONSTRAINT for_key_book_id
    REFERENCES books (book_id),
  transaction_date DATE
    CONSTRAINT tran_date_not_null NOT NULL,
  transaction_type NUMBER
    CONSTRAINT tran_type_not_null NOT NULL,
  CONSTRAINT transactions_pk PRIMARY KEY (transaction_id));
```

To create the TRANSACTIONS table, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Tables node in the schema hierarchy on the left side, select **New Table**, and enter the

following information. (If a tab or field is not mentioned, do not enter anything for it. Be sure that the Advanced box is not checked when you start creating the table.)

Schema: Specify your current schema as the schema in which to create the table.

Name: TRANSACTIONS

Create the table columns using the following information. After creating each column except the last one (transaction_type), click **Add Column** to add the next column. (If you accidentally click OK instead of Add Column, right-click the TRANSACTIONS table in the Connections navigator display, select Edit, and continue to add columns.)

Column Name	Type	Size	Other Information and Notes
transaction_id	NUMBER		Primary Key. (Unique transaction ID number, with values to be created using a trigger and sequence that will be created automatically)
patron_id	NUMBER		(Foreign key; must match a patron_id value in the PATRONS table)
book_id	VARCHAR2	20	(Foreign key; must match a book_id value in the BOOKS table)
transaction_date	DATE		(Date and time of the transaction)
transaction_type	NUMBER		(Numeric code indicating the type of transaction, such as 1 for checking out a book)

After you have entered the last column (transaction_type), check **Advanced** (next to Schema). This displays a pane for selecting more table options. For this table, you will use the Column Sequences and Foreign Keys panes.

Column Sequences pane

You have already specified TRANSACTION_ID as the primary key, and you will use this pane only to specify that the primary key column values are to be populated automatically. This convenient approach uses a trigger and a sequence (both created automatically by SQL Developer), and ensures that each transaction ID value is unique.

Column: TRANSACTION_ID

Sequence: New Sequence

Trigger: TRANSACTIONS_TRG (The default; a before-insert trigger with this name will be created automatically.)

Foreign Keys tab

1. Click **Add** to create the first of the two foreign keys for the TRANSACTIONS table.

Name: for_key_patron_id

Referenced Schema: Name of the schema containing the table with the primary key or unique constraint to which this foreign key refers. Use the schema you have been using for this tutorial.

Referenced Table: PATRONS

Referenced Constraint: PATRONS_PK (The name of the primary key constraint for the PATRONS table. Be sure that the **Referenced Column on PATRONS** displayed value is PATRON_ID.)

Associations: Local Column: PATRON_ID

Associations: Referenced Column on PATRONS: PATRON_ID

2. Click **Add** to create the second of the two foreign keys for the TRANSACTIONS table.

Name: for_key_book_id

Referenced Schema: Name of the schema containing the table with the primary key or unique constraint to which this foreign key refers. Use the schema you have been using for this tutorial.

Referenced Table: BOOKS

Referenced Constraint: BOOKS_PK (The name of the primary key constraint for the BOOKS table. Be sure that the **Referenced Column on BOOKS** displayed value is BOOK_ID.

Associations: Local Column: BOOK_ID

Associations: Referenced Column on BOOKS: BOOK_ID

3. Click **OK** to finish creating the table.

You have finished creating all the tables. To create a sequence for use in generating unique primary key values for the PATRONS table, go to [Section 4.4, "Create a Sequence"](#).

4.4 Create a Sequence

Create one sequence object, which will be used in INSERT statements to generate unique primary key values in the PATRONS table. (You do not need to create a sequence for the primary key in the TRANSACTIONS table, because you used the SQL Developer feature that enables automatic population of primary key values for that table.) You will use the Create Sequence dialog box to create the sequence declaratively; the sequence that you create will be essentially the same as if you had entered the following statements using the SQL Worksheet:

```
CREATE SEQUENCE patron_id_seq
  START WITH 100
  INCREMENT BY 1;
```

After creating the sequence, you can use it in INSERT statements to generate unique numeric values. The following example uses the patron_id_seq sequence in creating a row for a new patron (library user), assigning her a patron ID that is the next available value of the patron_id_seq sequence:

```
INSERT INTO patrons VALUES (patron_id_seq.nextval,
  'Smith', 'Jane', '123 Main Street', 'Mytown, MA 01234', null);
```

To create the sequence, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Sequences node in the schema hierarchy on the left side, select **New Sequence**, and enter information using the Create Sequence dialog box.

Schema: Specify your current schema as the schema in which to create the sequence.

Name: patron_id_seq

Increment: 1

Start with: 100

Min value: 100

Click **OK** to finish creating the sequence.

To insert sample data into the tables, go to [Section 4.5, "Insert Data into the Tables"](#).

4.5 Insert Data into the Tables

For your convenience in using the view and the PL/SQL procedure that you will create, add some sample data to the BOOKS, PATRONS, and TRANSACTIONS tables. (If you do not add sample data, you can still create the remaining objects in this tutorial, but the view and the procedure will not return any results.)

Go to the SQL Worksheet window associated with the database connection you have been using. (For information about using the SQL Worksheet, see [Section 1.7, "Using the SQL Worksheet"](#).) Copy and paste the following INSERT statements into the **Enter SQL Statement** box:

```
INSERT INTO books VALUES ('A1111', 'Moby Dick', 'Melville', 'Herman', 10);
INSERT INTO books VALUES ('A2222', 'Get Rich Really Fast', 'Scammer', 'Ima', 1);
INSERT INTO books VALUES ('A3333', 'Finding Inner Peace', 'Blissford', 'Serenity',
null);
INSERT INTO books VALUES ('A4444', 'Great Mystery Stories', 'Whodunit', 'Rodney',
5);
INSERT INTO books VALUES ('A5555', 'Software Wizardry', 'Abugov', 'D.', 10);

INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Smith', 'Jane', '123 Main Street', 'Mytown, MA 01234', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Chen', 'William', '16 S. Maple Road', 'Mytown, MA 01234', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Fernandez', 'Maria', '502 Harrison Blvd.', 'Sometown, NH 03078', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Murphy', 'Sam', '57 Main Street', 'Mytown, MA 01234', null);

INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A1111', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A2222', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (101, 'A3333', SYSDATE, 3);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (101, 'A2222', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A3333', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (103, 'A4444', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A4444', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A2222', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A5555', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
```

```
VALUES (101, 'A2222', SYSDATE, 1);
```

Click the Run Script icon, or press the F5 key.

To create a view, go to [Section 4.6, "Create a View"](#).

4.6 Create a View

Create a view that returns information about patrons and their transactions. This view queries the PATRONS and TRANSACTIONS tables, and returns rows that contain a patron's ID, last name, and first name, along with a transaction and the transaction type. The rows are ordered by patron ID, and by transaction type within patron IDs.

To create the patrons_trans_view view, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Views node in the schema hierarchy on the left side, select **New View**, and enter the following information. (If a tab or field is not mentioned, do not enter anything for it.)

Schema: Specify your current schema as the schema in which to create the view.

Name: patrons_trans_view

SQL Query tab

In the SQL Query box, enter (or copy and paste) the following statement:

```
SELECT p.patron_id,
       p.last_name,
       p.first_name,
       t.transaction_type,
       t.transaction_date
FROM   patrons p, transactions t
WHERE  p.patron_id = t.patron_id
ORDER BY p.patron_id, t.transaction_type
```

Then click **Test Syntax**, and ensure that you have not made any syntax errors. If you made any errors, correct them and click Test Syntax again.

DDL

Review the SQL statement that SQL Developer will use to create the view. If you want to make any changes, go back to the SQL Query tab and make the changes there.

If you want to save the CREATE VIEW statement to a SQL script file, click **Save** and specify the location and file name.

When you are finished, click **OK**.

You have finished creating the view. If you inserted data to the underlying tables, as described in [Section 4.5, "Insert Data into the Tables"](#), you can see the data returned by this view as follows: in the Connections navigator, expand Views, and select PATRONS_TRANS_VIEW, then click the Data tab.

To create a procedure that lists all books with a specified rating, go to [Section 4.7, "Create a PL/SQL Procedure"](#).

4.7 Create a PL/SQL Procedure

Create a procedure that lists all books with a specified rating. You can then call this procedure with an input parameter (a number from 1 to 10), and the output will be all the titles of all books with that rating.

To create the procedure, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Procedures node in the schema hierarchy on the left side, select **New Procedure**, and enter the following information using the Create PL/SQL Procedure dialog box.

Object Name: list_a_rating

Click **OK**. A source window for the new procedure is opened. Enter (or copy and paste) the following procedure text, replacing any existing text:

```
CREATE OR REPLACE
PROCEDURE list_a_rating(in_rating IN NUMBER) AS
    matching_title VARCHAR2(50);
    TYPE my_cursor IS REF CURSOR;
    the_cursor my_cursor;
BEGIN
    OPEN the_cursor
    FOR 'SELECT title
        FROM books
        WHERE rating = :in_rating'
    USING in_rating;
    DBMS_OUTPUT.PUT_LINE('All books with a rating of ' || in_rating || ':');
    LOOP
        FETCH the_cursor INTO matching_title;
        EXIT WHEN the_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(matching_title);
    END LOOP;
    CLOSE the_cursor;
END list_a_rating;
```

This procedure uses a cursor (named the_cursor) to return only rows where the book has the specified rating (in_rating parameter), and uses a loop to output the title of each book with that rating.

Click the Save icon to save the procedure.

As a usage example, after creating the procedure named LIST_A_RATING, if you have inserted data into the BOOKS table (for example, using the INSERT statements in [Section 4.5, "Insert Data into the Tables"](#)), you could use the following statement to return all books with a rating of 10:

```
CALL list_a_rating(10);
```

To run this procedure within SQL Developer, right-click LIST_A_RATING in the Connections navigator hierarchy display and select **Run**. Under **PL/SQL Block** in the Run PL/SQL dialog box, change IN_RATING => IN_RATING to IN_RATING => **10**, and click **OK**. The Log window display will now include the following output:

```
All books with a rating of 10:
Moby Dick
Software Wizardry
```

4.8 Debug a PL/SQL Procedure

If you want to practice debugging a PL/SQL procedure with SQL Developer, create a procedure that is like the list_a_rating procedure that you created in [Section 4.7, "Create a PL/SQL Procedure"](#), but with a logic error. (The coding is also deliberately inefficient, to allow the display of the rating in a variable.)

Before you can debug the procedure, you must ensure that the user associated with the database connection has the DEBUG CONNECT SESSION and DEBUG ANY PROCEDURE privileges.

To create this procedure, if you are not already connected, connect to the database as the user for the schema you are using for this tutorial. Right-click the Procedures node in the schema hierarchy on the left side, select **New Procedure**, and enter the following information using the Create PL/SQL Procedure dialog box.

Object Name: list_a_rating2

Click **OK**. A source window for the new procedure is opened. Enter (or copy and paste) the following procedure text, replacing any existing text:

```
CREATE OR REPLACE
PROCEDURE list_a_rating2(in_rating IN NUMBER) AS
    matching_title VARCHAR2(50);
    matching_rating NUMBER;
    TYPE my_cursor IS REF CURSOR;
    the_cursor my_cursor;
    rating_cursor my_cursor;
BEGIN
    OPEN the_cursor
        FOR 'SELECT title
            FROM books
            WHERE rating <= :in_rating'
        USING in_rating;
    OPEN rating_cursor FOR 'SELECT rating FROM books WHERE
        rating <= :in_rating' USING in_rating;
    DBMS_OUTPUT.PUT_LINE('All books with a rating of ' || in_rating || ':');
    LOOP
        FETCH the_cursor INTO matching_title;
        FETCH rating_cursor INTO matching_rating;
        EXIT WHEN the_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(matching_title);
    END LOOP;
    CLOSE the_cursor;
    CLOSE rating_cursor;
END list_a_rating2;
```

This procedure contains a logic error in the definition of the `the_cursor`: it selects titles where the rating is less than or equal to a specified rating, whereas it should select titles only where the rating is equal to the specified rating.

Click the Save icon to save the procedure.

Assume that you wanted to run this procedure and list all books with a rating of 10. Right-click LIST_A_RATING2 in the Connections navigator hierarchy display and select **Run**. Under **PL/SQL Block** in the Run PL/SQL dialog box, change, change IN_RATING => IN_RATING to IN_RATING => **10**, and click **OK**. In the Log window, however, you see unexpected output: many titles are listed, including some with ratings other than 10. So, you decide to debug the procedure.

To debug the procedure, follow these steps:

1. Click the Compile for Debug icon in the toolbar under the LIST_A_RATING2 tab.
2. Set two breakpoints by clicking in the left margin (left of the thin vertical line) beside each of these two lines:

```
FETCH the_cursor INTO matching_title;
FETCH rating_cursor INTO matching_rating;
```

Clicking in the left margin toggles the setting and unsetting of breakpoints. Clicking beside these two lines will enable you to see the values of the `matching_title` and `matching_rating` variables as execution proceeds in debug mode.

3. On the **Debug** menu, select **Debug LIST_A_RATING2**. Ensure that the line `IN_RATING => IN_RATING` has been changed to `IN_RATING => 10`, and click **OK**.
4. Click **View**, then **Debugger**, then **Data** to display the Data pane. (Tip: Expand the Name column width so that you can see `MATCHING_RATING`.)
5. Press the **F9** key (or click **Debug**, then **Resume**) to have execution proceed, stopping at the next breakpoint.
6. Repeatedly press the **F9** key (or click **Debug**, then **Resume**), noticing especially the value of `MATCHING_RATING` as each row is processed. You will notice the first incorrect result when you see that the title *Get Rich Really Fast* is included, even though its rating is only 1 (obviously less than 10). (See the screen illustration with debugging information in [Section 1.6, "Running and Debugging Functions and Procedures"](#).)
7. When you have enough information to fix the problem, you can click the Debugging - Log tab, and Terminate icon in the debugging toolbar.

From this debugging session, you know that to fix the logic error, you should change `rating <= :in_rating` to `rating = :in_rating` in the definition of the `cursor`.

4.9 Use the SQL Worksheet for Queries

You can use the SQL Worksheet to test SQL statements using a database connection. To display the worksheet, from the **Tools** menu, select **SQL Worksheet**. In the Select Connection dialog box, select the database connection that you used to create the `BOOKS`, `PATRONS`, and `TRANSACTIONS` tables for the tutorial in [Chapter 4, "Tutorial: Creating Objects for a Small Database"](#).

The SQL Worksheet has the user interface shown in [Section 1.7, "Using the SQL Worksheet"](#).

In the **Enter SQL Statement** box, enter the following statement (the semicolon is optional for the SQL Worksheet):

```
SELECT author_last_name, title FROM books;
```

Notice the automatic highlighting of SQL keywords (`SELECT` and `FROM` in this example).

Click the Execute SQL Statement icon in the SQL Worksheet toolbar. The results of the query are displayed on the **Results** tab under the area in which you entered the SQL statement.

In the **Enter SQL Statement** box, enter (or copy and paste) the following statement, which is the same as the `SELECT` statement in the view you created in [Create a View](#):

```
SELECT p.patron_id,
       p.last_name,
       p.first_name,
       t.transaction_type,
       t.transaction_date
FROM patrons p, transactions t
WHERE p.patron_id = t.patron_id
ORDER BY p.patron_id, t.transaction_type;
```

Click the Execute SQL Statement icon in the SQL Worksheet toolbar, and view the results of the query.

Click the Execute Explain Plan icon in the SQL Worksheet toolbar to see the execution plan (displayed on the Explain tab) that Oracle Database follows to execute the SQL

statement. The information includes the optimizer strategy and the cost of executing the statement. (For information about how to generate and interpret execution plans, see *Oracle Database Performance Tuning Guide*.)

4.10 Script for Creating and Using the Library Tutorial Objects

The following statements create and use the database objects that you have created (or will create) for the tutorial in [Chapter 4, "Tutorial: Creating Objects for a Small Database"](#). You can view these commands to help you understand the library database objects that are covered in the tutorial.

```
-- Clean up from any previous tutorial actions.
DROP TABLE transactions;
DROP TABLE books;
DROP TABLE patrons;
DROP SEQUENCE patron_id_seq;
DROP SEQUENCE transactions_seq;
DROP TRIGGER transactions_trg;
DROP VIEW patrons_trans_view;
DROP PROCEDURE list_a_rating;
DROP PROCEDURE list_a_rating2;

set serveroutput on

-- Create objects.

CREATE TABLE books (
  book_id VARCHAR2(20),
  title VARCHAR2(50)
  CONSTRAINT title_not_null NOT NULL,
  author_last_name VARCHAR2(30)
  CONSTRAINT last_name_not_null NOT NULL,
  author_first_name VARCHAR2(30),
  rating NUMBER,
  CONSTRAINT books_pk PRIMARY KEY (book_id),
  CONSTRAINT rating_1_to_10 CHECK (rating IS NULL OR
    (rating >= 1 and rating <= 10)),
  CONSTRAINT author_title_unique UNIQUE (author_last_name, title));

CREATE TABLE patrons (
  patron_id NUMBER,
  last_name VARCHAR2(30)
  CONSTRAINT patron_last_not_null NOT NULL,
  first_name VARCHAR2(30),
  street_address VARCHAR2(50),
  city_state_zip VARCHAR2(50),
  location MDSYS.SDO_GEOMETRY,
  CONSTRAINT patrons_pk PRIMARY KEY (patron_id));

CREATE TABLE transactions (
  transaction_id NUMBER,
  patron_id CONSTRAINT for_key_patron_id
  REFERENCES patrons(patron_id),
  book_id CONSTRAINT for_key_book_id
  REFERENCES books(book_id),
  transaction_date DATE
  CONSTRAINT tran_date_not_null NOT NULL,
  transaction_type NUMBER
  CONSTRAINT tran_type_not_null NOT NULL,
```

```

CONSTRAINT transactions_pk PRIMARY KEY (transaction_id);

CREATE SEQUENCE patron_id_seq
  START WITH 100
  INCREMENT BY 1;

-- The sequence for the transaction_id
-- in the tutorial is created automatically,
-- and may have the name TRANSACTIONS_SEQ.
CREATE SEQUENCE transactions_seq
  START WITH 1
  INCREMENT BY 1;

-- The before-insert trigger for transaction ID values
-- in the tutorial is created automatically,
-- and may have the name TRANSACTIONS_TRG.
CREATE OR REPLACE TRIGGER transactions_trg
  BEFORE INSERT ON TRANSACTIONS
  FOR EACH ROW
  BEGIN
    SELECT TRANSACTIONS_SEQ.NEXTVAL INTO :NEW.TRANSACTION_ID FROM DUAL;
  END;
/

CREATE VIEW patrons_trans_view AS
  SELECT p.patron_id,
         p.last_name,
         p.first_name,
         t.transaction_type,
         t.transaction_date
  FROM patrons p, transactions t
  WHERE p.patron_id = t.patron_id
  ORDER BY p.patron_id, t.transaction_type;

-- Procedure: List all books that have a specified rating.
CREATE OR REPLACE PROCEDURE list_a_rating(in_rating IN NUMBER) AS
  matching_title VARCHAR2(50);
  TYPE my_cursor IS REF CURSOR;
  the_cursor my_cursor;
BEGIN
  OPEN the_cursor
  FOR 'SELECT title
      FROM books
      WHERE rating = :in_rating'
  USING in_rating;
  DBMS_OUTPUT.PUT_LINE('All books with a rating of ' || in_rating || ':');
  LOOP
    FETCH the_cursor INTO matching_title;
    EXIT WHEN the_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(matching_title);
  END LOOP;
  CLOSE the_cursor;
END;
/
show errors;

-- Insert and query data.

INSERT INTO books VALUES ('A1111', 'Moby Dick', 'Melville', 'Herman', 10);
INSERT INTO books VALUES ('A2222', 'Get Rich Really Fast', 'Scammer', 'Ima', 1);

```

```
INSERT INTO books VALUES ('A3333', 'Finding Inner Peace', 'Blissford', 'Serenity',
null);
INSERT INTO books VALUES ('A4444', 'Great Mystery Stories', 'Whodunit', 'Rodney',
5);
INSERT INTO books VALUES ('A5555', 'Software Wizardry', 'Abugov', 'D.', 10);

INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Smith', 'Jane', '123 Main Street', 'Mytown, MA 01234', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Chen', 'William', '16 S. Maple Road', 'Mytown, MA 01234', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Fernandez', 'Maria', '502 Harrison Blvd.', 'Sometown, NH 03078', null);
INSERT INTO patrons VALUES (patron_id_seq.nextval,
'Murphy', 'Sam', '57 Main Street', 'Mytown, MA 01234', null);

INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A1111', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A2222', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (101, 'A3333', SYSDATE, 3);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (101, 'A2222', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A3333', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (103, 'A4444', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (100, 'A4444', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A2222', SYSDATE, 2);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (102, 'A5555', SYSDATE, 1);
INSERT INTO transactions (patron_id, book_id,
transaction_date, transaction_type)
VALUES (101, 'A2222', SYSDATE, 1);

-- Test the view and the procedure.
SELECT * FROM patrons_trans_view;
CALL list_a_rating(10);
```

Dialog Boxes for Creating/Editing Objects

SQL Developer uses dialog boxes for creating and editing database connections and objects in the database (tables, views, procedures, and so on). The dialog boxes sometimes have multiple tabs, each reflecting a logical grouping of properties for that type of object.

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

The dialog boxes are not presented here in any rigorous order, because the help for each box is an independent piece of information and is normally seen when you click Help or press F1 in that box.

Note: For all Name fields, any name that you type is automatically converted to and stored in the database metadata in uppercase, unless you enclose the name in quotation marks (" "). (Names of database objects in SQL and PL/SQL statements are not case-sensitive.)

To include lowercase characters, special characters, or spaces in object names, enclose the name in quotation marks (" ") when you type it.

Example: "My table"

5.1 Add Extension

This dialog box is displayed when you click Add in the [File Types](#) pane of [SQL Developer Preferences](#).

Extension: Specify the file extension, including the period (for example, .xyz).

After you click OK, you can select that extension and modify its details, including the file type, content type, and whether to have files with the extension automatically opened by SQL Developer.

5.2 Change Type

Use this dialog box to change the data type of a column in a captured model before you perform the migration.

Source Data Type: Specify the new data type for the column.

Any remaining fields in the dialog box depend on the Source Data Type that is selected.

5.3 Check for Updates

When you click **Help** and then **Check for Updates**, you can check for and download available SQL Developer updates. The following pages may be displayed. (If you have enabled the SQL Developer 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 SQL Developer user preferences for [Web Browser and Proxy](#).

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 SQL Developer 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 SQL Developer.

5.4 Check Out from CVS

Use this dialog box to check out modules from a CVS repository.

Connection Name: Name of the connection to the repository

Module Name: Name of the module to be checked out.

Path: Path to the module.

Get/Refresh Module List: Displays the list of modules or updates the current display.

Destination Folder: Folder into which to place the checked out files.

Use Revision or Tag: If this option is checked, the revision or tag that you specify in the text box is used. To see the available tags, click the binoculars icon.

Prune Empty Folders: If this option is checked, empty folders are removed from the working directory.

5.5 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.

5.6 Configure Component Palette

Lets you configure the component palette. Note that some page types cannot be edited or removed, and most existing component types cannot be added to, edited, or removed.

Add: Displays the [Create Palette Page](#) dialog box.

Remove: Deletes the selected page from the palette.

Rename: Renames a specified page.

5.7 Create Palette Page

Lets you create a new page for the component palette. Specify a name of the page, and select the type of page from a list.

Page Name: Name of the page. Suggestion: Include the type of page in the name, perhaps naming pages in the form *name_type_page*.

Page Type: Page type, selected from the list.

5.8 Confirm Drop Application

This dialog box is displayed when you right-click an Application Express application and select Drop. To drop the application, click **Yes**; to keep (not drop) the application, click **No**.

If the application contains an uninstall script, that script is run before the application is dropped.

5.9 Confirm Running SQL

This dialog box is displayed in certain situations when SQL Developer needs to run a setup script on the server. The script is displayed in a text box, where you can view or edit the contents. To allow the script to run, click **Yes**; to prevent the script from running, click **No**.

5.10 Connection Has Uncommitted Changes

This dialog box is displayed if you try to end the active database session while there are transactions to be committed. Select the appropriate option and click **OK**.

To commit the changes and end the session, select **Commit Changes**. To roll back the changes and end the session, select **Rollback Changes**. To cancel the attempt to end the session, select **Abort Connection Disconnect**. (Selecting Abort Connection Disconnect and clicking OK has the same effect as clicking Cancel.)

5.11 Create New Object

This dialog box is displayed if you click File, then New. Specify the type of object to create. After you click OK, the dialog box for creating that type of object is displayed.

5.12 Create/Edit CVS Connection

This information applies to creating or editing a CVS (Concurrent Versions System) connection. For information about SQL Developer support for versioning and CVS, see [Section 1.10](#).

Connection

Access Method: The method by which the client will gain access to and authenticate against the server. The methods available depend on which CVS preferences you have set; the available methods might include External, Password Server, Secure Shell via SSH2, and [Other].

Most of the remaining Connection fields apply only to specific Access Method values.

User Name: A CVS user name known to the repository.

Host Name: Qualified host name or IP address of the CVS server system.

Port: TCP/IP port number on which the repository is listening.

Repository Path: The location of the CVS repository software. The seeded / can be overwritten with a path in the format suitable for your operating system, for example c:\cvs. A simple formatting error, such as a forward slash instead of a backslash, will result in a message asking you to enter a valid repository path.

SSH2 Key: Path and file name for the SSH2 private key file for this connection. You can generate a SSH2 private key file using Generate SSH2 Key Pair.

Generate SSH2 Key Pair: Displays a dialog box for generating an SSH2 key pair (that is, a private key file and a public key file). You specify the private key file in the SSH2 Key box. You add the details of the public key file to the list of public keys on the CVS server system

Use HTTP Proxy Settings: Check (enable) this option if you are behind a firewall and need to use HTTP to access the CVS server.

External Locator Configuration: Displays the [External Locator Configuration](#) dialog box, in which you can edit the details of the remote shell client and remote server program.

Root

Value of CVSROOT: CVS root variable made up from the information that you have already provided. This variable provides the client with access details when contacting the server. The format of the seeded variable is:

```
:accessmethod:username@serverlocation:repositorypath
```

You would not normally need to change this value. One instance when you would change this value is when you are attempting to connect to a CVSNT server through a firewall. In this case, you would add proxy information to the beginning of the username portion, so that the CVS root variable would take the following form:

```
:accessmethod:proxy=proxyname;proxyport=portnumber:username@serverlocation:repositorypath
```


Test

Test Connection: Attempts to establish a connection to the CVS repository.

Status: Displays the result of the test (success or an error message).

Name

Connection Name: Name to identify the connection to the CVS repository. The default name is the same as the CVSROOT value.

Summary

Displays the connection information that you have specified. To make any corrections, click **Back** as needed and modify the information. To create the connection, click **Finish**.

5.13 Create/Edit/Select Database Connection

The database connection dialog box displays any existing connections. Depending on the context, you can select a connection to connect to the database, edit the information about existing connections, or specify information while creating a new connection. (See [Creating and Editing 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.

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, 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.

Information for Database-Specific Tabs:

- [Oracle tab](#)
- [TimesTen tab](#)
- [Access tab](#)
- [DB2 tab](#)
- [JDBC tab](#)
- [MySQL tab](#)
- [SQL Server and Sybase tabs](#)
- [Teradata tab](#)

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, LDAP (Lightweight Directory Access Protocol), or Advanced. (The display of fields changes to reflect any change in connection type.)

OS Authentication: If this option is checked, control of user authentication is passed to the operating system (OS). This allows the specified user to connect to the database by authenticating that user's OS username in the database. No password is associated with the connection since it is assumed that OS authentication is sufficient. For information about using OS authentication, see *Oracle Database JDBC Developer's Guide*.

Kerberos Authentication: If this option is checked, credentials can be shared across many Kerberos-enabled applications (for example, to have the same username and password for both the operating system and Oracle Database). Thick driver configuration is done through `sqlnet.ora` (`sqlnet.authentication_services=(KERBEROS)` and related parameters), so no username and password are needed. Thin driver configuration uses the configuration (`.conf`) file and the credentials cache, and uses a service principal and password. For more information about Kerberos authentication options, see [Database: Advanced](#). For information about configuring Kerberos authentication, see *Oracle Database Advanced Security Administrator's Guide*.

Proxy Connection: If this option is checked, proxy authentication will be used, as explained in [Section 1.4.5, "Connections with Proxy Authentication"](#). Displays the [Oracle Proxy Authentication](#) dialog box.

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.

LDAP connection type

Enterprise users are authenticated with the Lightweight Directory Access Protocol (LDAP) server. The user login information must be configured in the LDAP server and mapped to a schema in the database. Support for LDAP-compliant directory servers provides a centralized vehicle for managing and configuring a distributed Oracle network. The directory server can replace client-side and server-side localized `tnsnames.ora` files.

LDAP Server: Select from the list (from `<DIRECTORY_SERVER>` entries in the `ldap.ora` file); or enter the directory server location and port (either SSL or non-SSL), for example: `system123.example.com:389:636` (`ldap-system:nonssl-port:ssl-port`)

Context: LDAP administrative context. The contexts available in the selected server are listed.

DB Service: Database connection information: click **Load** to display a list of database services associated with the selected context. (If an error is displayed, no database services are associated with this context.) If a connection uses the OCI/Thick driver (see the **Use OCI/Thick** preference under [Database: Advanced](#)), the system on which

SQL Developer is running must have an Oracle Client installation that contains the JDBC and orai18n libraries, these libraries must be present on the path, and the Oracle Client installation must be version 10.2 or later.

Advanced connection type

Custom JDBC URL: URL for connecting directly from Java to the database; overrides any other connection type specification. If you are using TNS or a naming service with the OCI driver, you must specify this information: Example:

```
jdbc:oracle:thin:scott/@localhost:1521:orcl
```

Note that in this example, the "/" is required, and the user will be prompted to enter the password.

To use a custom JDBC URL, the system on which SQL Developer is running must have an Oracle Client installation that contains the JDBC and orai18n libraries, is present on the path, and is version 10.2 or later.

TimesTen tab

The following information applies to a connection to an Oracle TimesTen In-Memory Database.

For **Username** and **Password**, specify the user name and password of the user account in the TimesTen database.

DSN: Data source name. Select an existing DSN (if any are displayed), or *User-specified* to create a new DSN. A DSN is a character string that identifies a TimesTen database and includes connection attributes to be used when connecting to the database. A DSN has the following characteristics: its maximum length is 32 characters; it cannot contain spaces; and it consists of ASCII characters except for the following: [{};?*=@\

Connection Type (if DNS is user-specified): *C/S* for client-server mode or *Direct* for direct mode

Connection String: Connection attributes including database attributes, first connection attributes, general connection attributes, NLS attributes, and Cache Connect attributes. (See the *TimesTen Cache Connect to Oracle Guide* for information about the attributes.)

Oracle Password (for Cache): The password for the TimesTen user account on the Oracle Database. (See the TimesTen documentation for more information.)

For more information about SQL Developer support for TimesTen, see [Section 1.15](#).

For detailed usage and reference information about Oracle TimesTen, see the online documentation that is included in the TimesTen installation. For additional information, go to:

<http://www.oracle.com/technology/products/timesten/>

Access tab

For a connection to a Microsoft Access database, click Browse and find the database (.mdb) file. However, to be able to use the connection, you must first ensure that the system tables in the database file are readable by SQL Developer, as follows:

1. Open the database (.mdb) file in Microsoft Access.
2. Click **Tools**, then **Options**, and on the **View** tab ensure that **System Objects** are shown.

3. Click **Tools**, then **Security**, and, if necessary, modify the user and group permissions for the MSysObjects, MsysQueries, and MSysRelationships tables as follows: select the table and give the Admin user at least Read Design and Read Data permission on the table.
4. Save changes and close the Access database file.
5. Create and test the connection in SQL Developer.

DB2 tab

The following information applies to a connection to an IBM DB2 database.

Note that to connect to an IBM DB2 database, you must first download the db2jcc.jar and db2jcc_license_cu.jar files, and then click **Tools**, then **Preferences**, and use the SQL Developer user preference pane for [Database: Third Party JDBC Drivers](#) to add these files.

Platform: UDB or iSeries.

Host Name: Host system for the IBM DB2 database.

Port: TCP/IP Port on which the IBM DB2 server will listen.

Enter Database: Name of the IBM DB2 database.

JDBC tab

The following information applies to a JDBC connection.

JDBC-ODBC Bridge or Other Third Party Driver: Indicates a JDBC to ODBC bridge driver or another third-party driver.

Data Source (JDBC-ODBC Bridge): Name of an existing ODBC data source.

Extra Parameters (JDBC-ODBC Bridge): Additional parameters for the connection.

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).

MySQL tab

The following information applies to a connection to a MySQL database.

Note that to connect to a MySQL database, you must first download the appropriate MySQL connection driver, and then click **Tools**, then **Preferences**, and use the SQL Developer user preference pane for [Database: Third Party JDBC Drivers](#) to add the driver.

Host Name: Host system for the MySQL database.

Port: TCP/IP Port on which the MySQL server will listen.

Choose Database: Name of the MySQL database.

Zero Date Handling: Because the MySQL JDBC driver cannot handle the default 0000-00-00 date, specify one of the following options for handling this date: **Set to NULL** to set it to a null value, or **Round to 0001-01-01** to set it to 0001-01-01.

SQL Server and Sybase tabs

The following information applies to a connection to a Microsoft SQL Server or Sybase Adaptive Server database.

Note that to connect to a Microsoft SQL Server or Sybase Adaptive Server database, you must first download the appropriate connection driver, and then click **Tools**, then **Preferences**, and use the SQL Developer user preference pane for [Database: Third Party JDBC Drivers](#) to add the driver.

Host Name: Host system for the Microsoft SQL Server or Sybase Adaptive Server database.

Port: TCP/IP Port on which Microsoft SQL Server or Sybase Adaptive Server will listen.

Retrieve Database: Name of the Microsoft SQL Server or Sybase Adaptive Server database.

Teradata tab

The following information applies to a connection to a Teradata database.

Note that to connect to a Teradata database, you must first download the tdgssconfig.jar and a terajdbc4.jar files, and then click **Tools**, then **Preferences**, and use the SQL Developer user preference pane for [Database: Third Party JDBC Drivers](#) to add these files. (See also the readme.txt file that is included with the tdgssconfig.jar and a terajdbc4.jar files.)

Host Name: Host system for the Teradata database.

DBS Port: TCP/IP Port on which the Teradata server will listen.

Charset: Character set for the data.

TMODE: Transaction mode: ANSI, TERA (Teradata), or DEFAULT.

To add a connection parameter to the list in the box, click **Add**; to delete a connection parameter from the list, click **Delete**.

Creating and Editing Connections

*To create a new connection when no connections exist, enter the connection information and click **Connect**. To test the connection before you create it, click **Test**.*

*To create a new connection when one or more connections already exist, click to select an existing connection, change the Connection Name to the desired name, edit other connection information as needed, and click **Save** or **Connect** to create the new connection. To test the connection before you create it, click **Test**.*

*To edit an existing connection, click in its entry in the Connection Name column, change any connection information except the connection name, and click **Save** or **Connect**. To test the connection before you save changes to it, click **Test**.*

5.14 Rename Model (Migration)

This dialog box is displayed when you right-click a captured or converted model and select **Rename Model**. To rename the model, change the name and click **OK**.

5.15 Rename Database Item (Migration)

This dialog box is displayed when you right-click a database object under a captured or converted model and select Rename. To rename the object, change the name and click **OK**.

5.16 Select Connection

Use this dialog box to select a database connection for use with a specific SQL Developer feature (for example, the SQL worksheet or the Reports navigator). After you click **OK**, the interface for the component is displayed, with the current user the same as the one specified in the connection.

To create a new database connection, click the plus (+) icon; to edit the selected database connection, click the pencil icon. In both cases, a dialog box for specifying connection information is displayed (see [Section 5.13, "Create/Edit/Select Database Connection"](#)).

5.17 Connection Information

Use this dialog box to specify the user name and password for the selected database 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.

5.18 No Connection Found

This dialog box is displayed when you attempt to perform an operation that requires a database connection, but no connection currently exists for that operation. For example, you might have opened a SQL file but not selected a connection, or the connection might have disconnected; or you might have tried to perform a schema copy operation without specifying both the From Schema and To Schema connections.

To select a connection in the SQL Worksheet, click **OK** to close this dialog box, then select a connection from the drop-down list in the SQL Worksheet icon bar.

5.19 Connection Rename Error

This dialog box is displayed when you attempt to rename a database connection to a name that is already used for another connection. For example, you might have forgotten to enter a new name for the connection that you want to rename.

To rename the connection, click **OK** to close this dialog box, then specify a unique connection name.

5.20 New Folder (Connections)

This dialog box enable you to create or rename a folder for organizing database connections. If you are creating a folder, enter the name of the new folder; if you are renaming a folder, replace the existing name with the desired new name. For information about using folders, see [Section 1.4.1, "Using Folders to Group Connections"](#).

5.21 Continue After Pause

This dialog box is displayed when a PAUSE statement is encountered in a script that you are running in the SQL Worksheet.

To continue execution at the statement after the PAUSE statement, click **Yes**. To stop execution and not continue with the statement after the PAUSE statement, click **No**.

5.22 Select Library

This dialog box is displayed when you click Browse in the [Database](#) pane when setting [SQL Developer Preferences](#). Use this box to select the library for the specified JDBC driver class.

5.23 Create Library

This dialog box is displayed when you click New in the Select Library dialog box, which is displayed when you click Browse in the [Database](#) pane when setting [SQL Developer Preferences](#). Use this box to create the library for the specified JDBC driver class.

5.24 Import Data

This dialog box is displayed when you right-click the Tables node or a table name in the Connections navigator, select Import Data, and specify the .xls or .csv file from which to import data. It enables you to create a table and import data into it from an Microsoft Excel file, or to import Microsoft Excel data into an existing table.

Data Preview

Worksheet: Name of a worksheet in the Microsoft Excel file.

Header row?: If this option is checked, the first row in the selected Microsoft Excel worksheet is considered a row with text for the column headings. If this option is not checked, the first row is considered to contain worksheet data.

Locale: Language for any text data in the worksheet.

Choose Columns

Available Columns: Lists the Microsoft Excel worksheet columns from which you can select for import into columns in the table. To select one or more worksheet columns, use the arrow buttons to move columns from Available to Selected.

Selected Columns: Lists the columns whose data is to be imported into columns in the database table. To change the order of a selected column in the list for the import operation, select it and use the up and down arrow buttons.

Column Definition

Enables you to specify the name of the database table and information about the columns in that table.

Table Name: Name of the database table into which to import the Excel data.

Source Data Columns and Target Table Columns: You can select a source (Excel) data column to display its target (Oracle) column properties. For **Data Type**, select one of the supported types for this import operation. For a VARCHAR2 or NUMBER column, you must specify an appropriate **Size/Precision** value. You can specify

whether the column value can be null (**Nullable?**), and you can specify a default value (**Default**).

Finish

Verify: You must verify the import parameters. If any test fails, the Information column contains a brief explanation, and you must go back and fix any errors before you can click Finish.

Send to Worksheet: Does not immediately perform the import operation, but instead opens a SQL Worksheet with statements that will be used after you click the Run Script icon in the worksheet.

To perform the import operation, or to send the statements to a SQL Worksheet if you so specified, click **Finish**.

5.25 Export/Import Connection Descriptors

The Export Connection Descriptors dialog box exports information about one or more database connections to an XML file. The Import Connection Descriptors dialog box imports connections that have been exported. Connections that you import are added to any connections that already exist in SQL Developer.

File Name: Name of the XML file to contain exported information or that contains information to be imported. Use the Browse button to specify the location.

Connections: Names of connections that you can select for the export or import operation.

5.26 Create/Edit Database Link

The following information applies to a database link, which is a database object in one database that enables you to access objects on another database, as explained in [Section 1.3.3, "Database Links \(Public and Private\)"](#).

Public: If this option is checked, the database link is public (available to all users). If this option is not checked, the database link is private and is available only to you.

Schema: Database schema in which to create the database link.

Name: Name of the database link. Must be unique within a schema.

Host Name: The service name of a remote database. If you specify only the database name, Oracle Database implicitly appends the database domain to the connect string to create a complete service name. Therefore, if the database domain of the remote database is different from that of the current database, you must specify the complete service name.

Current User: Creates a current user database link. The current user must be a global user with a valid account on the remote database. If the database link is used directly, that is, not from within a stored object, then the current user is the same as the connected user.

Fixed User: Creates a fixed user database link, for which you specify the user name and password used to connect to the remote database.

Shared: If this option is checked, a single network connection is used to create a public database link that can be shared among multiple users. In this case, you must also specify the Authentication information.

Authentication - User Name and Password: The user name and password on the target instance. This information authenticates the user to the remote server and is required for security. The specified user and password must be a valid user and password on the remote instance.

DDL tab

You can review and save the SQL statement that SQL Developer will use to create the database link.

5.27 Create/Edit Index

The following information applies to an index, which is a database object that contains an entry for each value that appears in the indexed column or columns of the table or cluster and provides direct, fast access to rows, as explained in [Section 1.3.7, "Indexes"](#). For detailed information about all index-related options, see the CREATE INDEX reference section in *Oracle Database SQL Language Reference*.

Advanced: If this option is checked, the dialog box changes to enable you to set advanced properties (select Advanced on the left side).

Schema: Database schema that owns the table associated with the index.

Table: Name of the table associated with the index.

Name: Name of the index. Must be unique within a schema.

Index Type: **Normal** for a standard Oracle index, in which case you also specify non-unique, unique, or bitmap, as well as one or more index expressions; or **Text** for an Oracle Text index (created with INDEXTYPE IS CTXSYS.CONTEXT), in which case you specify the column to be indexed.

Non-unique means that the index can contain multiple identical values; **Unique** means that no duplicate values are permitted; **Bitmap** stores rowids associated with a key value as a bitmap.

Index: A list of index expressions, that is, the table columns or column expressions in the index. To add an index expression, click the Add Column Expression (+) icon; this adds a column name here and in Column Expression, where you can edit it. To delete an index expression, click the Remove Column Expression (X) icon; to move an index expression up or down in the list, click the Move Column Up and Move Column Down icons. An index must have at least one index expression.

For example, to create an index on the AUTHOR_LAST_NAME column of the BOOKS table from the tutorial (see [Section 4.1, "Create a Table \(BOOKS\)"](#)), click the + icon, and select AUTHOR_LAST_NAME in Column Name or Expression (next field), which changes BOOKS to AUTHOR_LAST_NAME in the Index field.

Column Name or Expression: A column name or column expression. A column expression is an expression built from columns, constants, SQL functions, and user-defined functions. When you specify a column expression, you create a function-based index.

Order: ASC for an ascending index (index values sorted in ascending order); DESC for a descending index (index values sorted in descending order).

Properties

Enables you to specify index properties such as compression, parallelism, and storage options.

Compress: If this option is checked, key compression is enabled, which eliminates repeated occurrence of key column values and may substantially reduce storage. If this option is checked, you can enter an integer to specify the prefix length (number of prefix columns to compress).

Parallel: If this option is checked, parallel creation of the index is enabled. You can also enter an integer in the text box to specify the degree of parallelism, which is the number of parallel threads used in the parallel operation. (Each parallel thread may use one or two parallel execution servers.) If you specify Parallel without entering an integer, the optimum degree of parallelism is automatically calculated.

Storage Options: Enables you to specify storage options for the index. Displays the [Storage Options](#) dialog box.

Partitions

When applicable, enables you to specify whether the index is not partitioned, locally partitioned, or globally partitioned. If you specify Local or Global, additional fields are displayed relevant to the selected partitioning option.

Local: Specifies that the index is partitioned on the same columns, with the same number of partitions and the same partition bounds as its associated table. Oracle Database automatically maintains local index partitioning as the underlying table is repartitioned.

Global: Specifies that the partitioning of the index is user defined and is not equipartitioned with the underlying table. You can partition a global index by range or by hash. In both cases, you can specify up to 32 columns as partitioning key columns. The partitioning column list must specify a left prefix of the index column list. If the index is defined on columns a, b, and c, then for the columns you can specify (a, b, c), or (a, b), or (a, c), but you cannot specify (b, c) or (c) or (b, a). If you omit the partition names, then Oracle Database assigns names of the form SYS_P*n*.

5.28 Create Filter

This dialog box is displayed when you click New to add a user-defined exclusion filter when importing files into a repository.

Filter: Shell filename pattern, which can contain both normal characters and meta-characters, including wildcards. (See the supplied Selected Filters list for typical patterns.) For example, to exclude files with the extension *xyz*, enter the following:
*.xyz

When you click OK, the specified filter is added to the Selected Filters list.

5.29 Create/Edit Materialized View Log

User this dialog box to create or edit a materialized view log, which is a table associated with the master table of a materialized view. For more information, see [Section 1.3.11, "Materialized View Logs"](#).

Schema: Database schema in which to create the materialized view log.

Name: Name of the master table of the materialized view to be associated with this materialized view log.

Properties tab

Tablespace: Tablespace in which the materialized view log is to be created.

Logging: LOGGING or NOLOGGING, to establish the logging characteristics for the materialized view log.

Row ID: Yes indicates that the rowid of all rows changed should be recorded in the materialized view log; No indicates that the rowid of all rows changed should not be recorded in the materialized view log.

Primary Key: Yes indicates that the primary key of all rows changed should be recorded in the materialized view log; No indicates that the primary key of all rows changed should not be recorded in the materialized view log.

New Values: INCLUDING saves both old and new values for update DML operations in the materialized view log; EXCLUDING disables the recording of new values in the materialized view log. If this log is for a table on which you have a single-table materialized aggregate view, and if you want the materialized view to be eligible for fast refresh, you must specify INCLUDING.

Cache: For data that will be accessed frequently, CACHE specifies that the blocks retrieved for this log are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This attribute is useful for small lookup tables. NOCACHE specifies that the blocks are placed at the least recently used end of the LRU list.

Parallel: If this option is checked, parallel operations will be supported for the materialized view log.

Object ID: For a log on an object table only: Yes indicates that the system-generated or user-defined object identifier of every modified row should be recorded in the materialized view log; No indicates that the system-generated or user-defined object identifier of every modified row should not be recorded in the materialized view log.

Sequence: Yes indicates that a sequence value providing additional ordering information should be recorded in the materialized view log; No indicates that a sequence value providing additional ordering information should not be recorded in the materialized view log. Sequence numbers (that is, Yes for this option) are necessary to support fast refresh after some update scenarios.

Available Filter Columns: Additional columns, which are non-primary-key columns referenced by subquery materialized views, to be recorded in the materialized view log. To select one or more filter columns, use the arrow buttons to move columns from Available to Selected.

DDL tab

You can view a SQL CREATE statement that reflects the current definition of the object, or a SQL ALTER statement to modify an existing object to reflect your changes.

To save the SQL statement to a script file, click **Save** and specify the location and file name.

5.30 Create PL/SQL Package

Use this dialog box to create a package to contain PL/SQL subprograms (functions or procedures, or a combination).

Schema: Database schema in which to create the PL/SQL package.

Name: Name of the package. Must be unique within a schema.

Add New Source in Lowercase: If this option is checked, new text is entered in lowercase regardless of the case in which you type it. This option affects only the appearance of the code, because PL/SQL is not case-sensitive in its execution.

The package is created and is displayed in the Editor window, where you can enter the details.

5.31 Create PL/SQL Subprogram (Function or Procedure)

Use this dialog box to create a PL/SQL subprogram (function or procedure). A function returns a value; a procedure does not return a value.

Specify the information for the package and for each parameter, then click OK to create the subprogram and have it displayed in the Editor window, where you can enter the details.

Schema: Database schema in which to create the PL/SQL subprogram.

Name: Name of the subprogram. Must be unique within a schema.

Add New Source in Lowercase: If this option is checked, new text is entered in lowercase regardless of the case in which you type it. This option affects only the appearance of the code, because PL/SQL is not case-sensitive in its execution.

Parameters tab

For each parameter in the procedure to be created, specify the following information.

Name: Name of the parameter.

Type: Data type of the parameter.

Mode: IN for input only, OUT for output only, or IN OUT for input and output (that is, the output is stored in the parameter overwriting its initial input value).

Default Value: Optionally, the default value if the parameter is omitted or specified as null when the subprogram is called.

To add a parameter, click the Add (+) icon; to delete a parameter, click the Remove (X) icon; to move a parameter up or down in the list, click the up-arrow or down-arrow icon.

DDL tab

You can view a SQL CREATE statement that reflects the current definition of the object, or a SQL ALTER statement to modify an existing object to reflect your changes.

5.32 Create/Edit Sequence

The following information applies to a sequence, which is an object from which multiple users may generate unique integers. You can use sequences to automatically generate primary key values.

Schema: Database schema in which to create the sequence.

Name: Name of the sequence. Must be unique within a schema.

Increment: Interval between successive numbers in a sequence.

Start with: Starting value of 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.

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

DDL tab

You can review the SQL statement that SQL Developer will use to create a new sequence or that reflects any changes you have made to the sequence properties.

5.33 Create SQL File

Use this dialog box to create a SQL script file and to open the file in a SQL Worksheet for editing.

File Name: Name and extension of the file to be created. The default and recommended extension is .sql.

Directory Name: Directory path for the file. To specify a directory, you can click Browse. The default directory is the [Location of User-Related Information](#).

5.34 Create/Edit Synonym

The following information applies to a synonym, which is an alternative name for a table, view, sequence, procedure, stored function, package, materialized view, Java class database object, user-defined object type, or another synonym.

Public: If this option is checked, the synonym is accessible to all users. (However each user must have appropriate privileges on the underlying object in order to use the synonym.) If this option is not checked, the synonym is a private synonym, and is accessible only within its schema.

Schema: Database schema in which to create the synonym.

Name: Name of the synonym. A private synonym must be unique within its schema; a public synonym must be unique within the database.

For - Referenced Schema: Schema containing the object or name to which this synonym refers.

Object Based: Specify the object to which this synonym refers.

Name Based: Enter the name of the object to which this synonym refers.

DDL tab

You can review the SQL statement that SQL Developer will use to create a new synonym or that reflects any changes you have made to the synonym properties.

5.35 Create Table (quick creation)

This dialog box (if you do not check the Advanced box) creates a new table quickly by specifying columns and some frequently used features. (If you need to add or change features after you create the table, you can edit the table by clicking the Modify icon while viewing the table or by right-clicking its name in the Connections navigator and selecting Properties, which displays the [Create/Edit Table \(with advanced options\)](#) dialog box.)

To create a new table, the only things you **must** do are specify the schema and the table name, add the necessary columns, and click OK. Although it is not required, you should also specify a primary key.

Advanced: If this option is checked, the dialog box changes to include an extended set of features for creating the table. For example, you must check this option if you want to create a partitioned table, an index-organized table, or an external table.

Schema: Database schema in which to create the table.

Name: Name of the table. Must be unique within a schema.

Table tab (quick creation)

Specifies properties for each column in the table.

Columns: Lists the columns currently in the table.

Note: To add a column after the currently selected column, click **Add Column**; to delete a column, select it and click **Remove Column**.

Column Name: Name of the column. Must be unique within the table. Suggestion: For a new column, replace any default name, such as COLUMN1.

Type: Data type for the column. The drop-down list includes only selected frequently used data types. To specify any other type for the column, you must use the Columns panel of the [Create/Edit Table \(with advanced options\)](#) dialog box.

Size: For VARCHAR2 data, the maximum size of the column data; for NUMBER data, the maximum number of digits.

Not Null: If this option is checked, the column must contain data; you cannot specify no value or an explicit null value for this column when you insert a row. If this option is not checked, the column can contain either data or no data.

Primary Key: If this option is checked, the column is the primary key, or part of the primary key, for the table. The primary key is the column, or set of columns, that uniquely identifies each row in the table. A primary key column cannot be null.

If you want to have the primary key values automatically populated by a convenient method that uses a before-insert trigger and a sequence, then before you finish creating the table, you must check the Advanced box and use the Primary Key tab, starting with the Populate Primary Key Column field.

To add another column, click **Add Column**. When you are finished adding columns, either click **OK** or click the DDL tab to review the CREATE TABLE statement.

DDL tab (quick creation)

You can review and save the CREATE TABLE statement that SQL Developer will use to create a new table or that reflects any changes you have made to the table

properties. If you want to make any changes, go back to the Table tab and make the changes there.

When you are finished, click **OK**.

5.36 Create/Edit Table (with advanced options)

The table dialog box is used for creating a new table or editing an existing table. The table properties are grouped under several tabs.

To create a new table, the only things you **must** do are specify the schema and the table name, add the necessary columns, and click **OK**. Although it is not required, you should also specify a primary key using the [Primary Key pane](#). For other table-related features, use the appropriate tabs; the order in which you visit tabs usually does not matter, although you might find it convenient to visit them in the sequence in this topic. If you are editing an existing table, you can visit the tabs in any order.

If you click **OK** before you are finished creating or editing the table, right-click the table name in the Connections navigator, select **Edit**, and continue creating or editing the table.

Schema: Database schema in which to create the table.

Name: Name of the table. Must be unique within a schema.

Type: The type of table:

- **Normal:** A regular database table. It can be partitioned (see [Partitioning pane](#), [Subpartition Templates pane](#), and [Partition Definitions pane](#)).
- **External:** An external table (see [External Table Properties pane](#)).
- **Index Organized:** An index-organized table (see [Index Organized Properties pane](#)).
- **Temporary Table:** A temporary table, which is not stored permanently in the database. The temporary table definition persists in the same way as the definition of a regular table, but the table segment and any data in the temporary table persist only for the duration of either the transaction (**Transaction** option) or the session (**Session** option).

The following panes may also be available. (Some panes are available only for tables of specific types or with specific features.)

Columns pane

Specifies properties for each column in the table.

Columns: Lists the columns currently in the table. To add a column, click the Add Column (+) icon; to delete a column, select it and click the Remove Column (X) icon; to move a column up or down in the table definition, select it and use the up-arrow and down-arrow buttons.

To copy in column definitions from another table, click the Copy Columns icon to display the [Copy Columns](#) dialog box.

Note: After you add a column, **to add another column**, click the Add Column (+) icon.

Name: Name of the column. Must be unique within the table. Suggestion: For a new column, replace any default name, such as COLUMN1.

Datatype: **Simple** indicates a simple (non-object) data type; **Complex** indicates an object type. For a complex type, you must specify the schema and the type name (for example, MDSYS and SDO_GEOMETRY for the Oracle Spatial geometry type).

Type: Name of the data type. Most of the remaining information depends on the specific type.

Precision: For numeric data, the precision (total number of significant digits that can be represented) of the column data.

Scale: For numeric data, the scale (number of digits after the decimal point) of the column data.

Size: For character data, the maximum size of the column data.

Units: For character data, the units represented by the Size: **BYTE** for bytes or **CHAR** for characters. This attribute is important if the database can contain data in Unicode format, with multiple bytes for each character.

Default: For relevant types, the default value inserted into the column if no value is specified when a row is inserted.

Cannot be NULL: If this option is checked, the column must contain data; you cannot specify no value or an explicit null value for this column when you insert a row. If this option is not checked, the column can contain either data or no data. A primary key column (see [Primary Key pane](#)) cannot be null.

Comment: Optional descriptive comment about the column.

To add another column, click the Add Column (+) icon.

Primary Key pane

Specifies the primary key for the table. The primary key is the column, or set of columns, that uniquely identifies each row in the table.

An index is automatically created on the primary key.

Name: Name of the constraint to be associated with the primary key definition. Must be unique within the database.

Enabled: If this option is checked, the primary key constraint is enforced: that is, the data in the primary key column (or set of columns) must be unique and not null.

Available Columns: Lists the columns that are available to be added to the primary key definition.

Selected Columns: Lists the columns that are included in the primary key definition.

To add a column to the primary key definition, select it in Available Columns and click the Add (>) icon; to remove a column from the primary key definition, select it in Selected Columns and click the Remove (<) icon. To move all columns from available to selected (or the reverse), use the Add All (>>) or Remove All (<<) icon. To move a column up or down in the primary key definition, select it in Selected Columns and use the arrow buttons.

The remaining fields (Populate Primary Key Column through Trigger Name) appear *only* when you are creating a table. They are not available when you are editing an existing table.

Populate Primary Key Column: When you are creating a table, if you want to use a trigger and a sequence to have a unique value automatically inserted into the primary key column when you insert a new row, specify the primary key column.

From: An existing sequence that you select, or a new sequence whose name you enter. (For a new sequence, SQL Developer creates the sequence automatically using the name that you enter.)

Trigger Name: The name for the before-insert trigger that will be automatically created. This trigger uses the sequence to generate a new value for the primary key when a row is inserted. For an example of using this technique, see the tutorial section [Section 4.3, "Create a Table \(TRANSACTIONS\)"](#).

Unique Constraints pane

Specifies one or more unique constraints for the table. A unique constraint specifies a column, or set of columns, whose data values must be unique: each data value must not be null, and it must not be the same as any other value in the column.

For a multicolumn unique constraint, the combination of values must be unique, and no column in the constraint definition can have a null value. For example, if you specify the office_name and city columns for a unique constraint, you could not have two Sales offices in Chicago, but you could have a Sales office in Chicago and a Sales office in Atlanta.

Unique Constraints: Lists the unique constraints currently defined on the table. To add a unique constraint, click the Add button; to delete a unique constraint, select it and click the Remove button.

Note: After you add a unique constraint, **to add another unique constraint**, click the **Add** button.

Name: Name of the unique constraint. Must be unique within the database.

Enabled: If this option is checked, the unique constraint is enforced.

Available Columns: Lists the columns that are available to be added to the unique constraint definition.

Selected Columns: Lists the columns that are included in the unique constraint definition.

To add a column to the unique constraint definition, select it in Available Columns and click the Add (>) icon; to remove a column from the unique constraint definition, select it in Selected Columns and click the Remove (<) icon. To move all columns from available to selected (or the reverse), use the Add All (>>) or Remove All (<<) icon. To move a column up or down in the unique constraint definition, select it in Selected Columns and use the arrow buttons.

Foreign Keys pane

Specifies one or more foreign keys for the table. A foreign key specifies a column ("local column"), each of whose data values must match a value in the primary key or unique constraint of another table.

Foreign Keys: Lists the foreign keys currently defined on the table. To add a foreign key, click the Add button; to delete a foreign key, select it and click the Remove button.

Note: After you add a foreign key, **to add another foreign key**, click the **Add** button.

Name: Name of the foreign key definition. Must be unique within the database.

Enabled: If this option is checked, the foreign key is enforced.

Referenced Schema: Name of the schema containing the table with the primary key or unique constraint to which this foreign key refers.

Referenced Table: Name of the table with the primary key or unique constraint to which this foreign key refers.

Referenced Constraint: Name of the primary key or unique constraint to which this foreign key refers.

Associations: Local Column: Lists the column in the currently selected (local) table that is included in the foreign key definition. For each local column in the foreign key definition, select the name of a column in the local table.

Associations: Referenced Column on [table]: For each local column, identifies the column in the other (foreign) table that must have a value matching the value in the local column.

Check Constraints pane

Specifies one or more check constraints for the table. A check constraint specifies a condition that must be met when a row is inserted into the table or when an existing row is modified.

Check Constraints: Lists the check constraints currently defined on the table. To add a check constraint, click the Add button; to delete a check constraint, select it and click the Remove button.

Note: After you add a check constraint, **to add another check constraint**, click the **Add** button.

Name: Name of the check constraint definition. Must be unique within the database.

Enabled: If this option is checked, the check constraint is enforced.

Condition: Condition that must be met for a row. Can be any valid CHECK clause (without the CHECK keyword). For example, to indicate that the value in a numeric column named RATING must be from 1 to 10, you can specify: `rating >=1 and rating <= 10`

To add another check constraint, click the **Add** button.

Indexes pane

Specifies properties for each index on the table.

Indexes: Lists the indexes currently defined on the table. To add an index, click the Add Index (+) icon; to delete an index, select it and click the Remove Index (X) icon.

Note: After you add an index, **to add another index**, click the Add Index (+) icon.

Name: Name of the index. Must be unique within the schema.

Index: A list of index expressions, that is, the table columns or column expressions in the index. To add an index expression, click the Add Column Expression (+) icon; this adds a column name here and in Column Expression, where you can edit it. To delete

an index expression, click the Remove Column Expression (X) icon; to move an index expression up or down in the list, click the Move Column Up and Move Column Down icons. An index must have at least one index expression.

For example, to create an index on the AUTHOR_LAST_NAME column of the BOOKS table from the tutorial (see [Create a Table \(BOOKS\)](#)), click the + icon, and select AUTHOR_LAST_NAME in Column Name or Expression (next field), which changes BOOKS to AUTHOR_LAST_NAME in the Index field.

Column Name or Expression: A column name or column expression. A column expression is an expression built from columns, constants, SQL functions, and user-defined functions. When you specify a column expression, you create a function-based index.

Order: ASC for an ascending index (index values sorted in ascending order); DESC for a descending index (index values sorted in descending order).

Column Sequences pane

Enables you to specify sequences and before-insert triggers to be used in populating a column with values. This approach is especially convenient for automatically populating primary key column values with unique values.

Column: Name of the column for which a sequence and a trigger are to be used to insert unique values. The data type of the column must be numeric.

Sequence: None causes no sequence to be used; Existing Sequence uses the sequence that you specify; New Sequence creates a new sequence with a default or specified name.

Trigger: Before-insert trigger that automatically inserts the next value of the specified sequence into the column when a new row is inserted.

Table Properties pane

Enables you to specify table properties such as compression, parallelism, and storage options.

Compress (heap-organized tables only): If this option is checked, data segments are compressed to reduce disk use. This clause is especially useful in environments such as data warehouses, where the amount of insert and update operations is small, and in OLTP environments.

Parallel: If this option is checked, parallel creation of the table is enabled, and the default degree of parallelism is set for queries and the DML INSERT, UPDATE, DELETE, and MERGE statements after table creation. You can also enter an integer in the text box to specify the degree of parallelism, which is the number of parallel threads used in the parallel operation. (Each parallel thread may use one or two parallel execution servers.) If you specify Parallel without entering an integer, the optimum degree of parallelism is automatically calculated.

Storage Options: Enables you to specify storage options for the table. Displays the [Storage Options](#) dialog box.

LOB Parameters pane

Specifies storage options for LOB (large object) columns, enabling you to override the default storage options.

Column: Name of the LOB column.

LOB Parameters: If this option is checked, the specified values for the remaining field are used. If this option is not checked, the default values for all fields are used.

Segment: LOB segment ID.

Tablespace: Name of the tablespace for the LOB data.

Store in Row: If this option is checked, the LOB value is stored in the row (inline) if its length is less than approximately 4000 bytes minus system control information.

Cache: Specifies how Oracle Database should store blocks in the buffer cache:

- **CACHE:** For data that is accessed frequently, indicates that the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This attribute is useful for small lookup tables.
- **NOCACHE:** For data that is not accessed frequently, indicates that the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed. NOCACHE is the default for LOB storage.
- **CACHE READS:** LOB values are brought into the buffer cache only during read operations but not during write operations.

Retention: If this option is checked, old versions of this LOB column are retained. You can specify this option only if the database is running in automatic undo mode and if you do not specify a Pct Version value.

Logging: <DEFAULT> means to use the Oracle Database default. ON means that the table creation and any subsequent direct loader (SQL*Loader) and direct-path INSERT operations against the table, partition, or LOB storage are logged in the redo log file. OFF means that these operations are not logged in the redo log file.

Chunk: The number of bytes to be allocated for LOB manipulation. If the value is not a multiple of the database block size, then the database rounds up in bytes to the next multiple. The maximum value is 32768 (32K), which is the largest Oracle Database block size allowed. The default CHUNK size is one Oracle Database block.

Pct Version: Specifies the maximum percentage of overall LOB storage space used for maintaining old versions of the LOB. The default value is 10, meaning that older versions of the LOB data are not overwritten until they consume 10% of the overall LOB storage space. You can specify a Pct Version value whether the database is running in manual mode (where it is the default) or automatic undo mode (where Retention is the default). You cannot specify both a Pct Version value and the Retention option.

Free Pools: Specifies the number of groups of free lists for the LOB segment, usually the number of instances in a Real Application Clusters environment or 1 for a single-instance database. You can specify this option only if the database is running in automatic undo mode. You cannot specify both a Free Pools value and the Free Lists fields.

Extents - Initial: Size of the first extent of the table. Specify K (kilobytes) or M (megabytes) for the unit associated with the number.

Extents - Next: Size of the next extent to be allocated to the table. Specify K (kilobytes) or M (megabytes) for the unit associated with the number.

Extents - Min: Minimum number of extents allocated when the table is created.

Extents - Max: Maximum number of extents allocated when the table is created.

Unlimited (if checked) means that there is no maximum (and any specified maximum is ignored).

Extents - Pct Increase: Percentage that each extent grows over the previous extent.

Buffer Pool: <DEFAULT> means to use the Oracle Database default. KEEP means to put blocks from the segment into the Keep buffer pool; maintaining an appropriately sized Keep buffer pool lets Oracle retain the database object in memory to avoid I/O operations. RECYCLE means to put blocks from the segment into the Recycle pool; an appropriately sized Recycle pool reduces the number of objects whose default pool is the Recycle pool from taking up unnecessary cache space.

Free Lists: Number of free lists for each of the free list groups for the table. The default and minimum value for this parameter is 1, meaning that each free list group contains one free list.

Free List Groups: Number of groups of free lists for the table. The default and minimum value for this parameter is 1. Oracle uses the instance number of Real Application Clusters instances to map each instance to one free list group.

Partitioning pane

Specifies partitioning options for a **partitioned table**, which is a table that is organized into smaller and more manageable pieces called partitions. SQL queries and DML statements do not need to be modified in order to access partitioned tables; however, after partitions are defined, DDL statements can access and manipulate individual partitions rather than entire tables or indexes. Also, partitioning is entirely transparent to applications.

Partition By: The type of partitioning: **RANGE** partitions the table on ranges of values from the column list (which for an index-organized tablet must be a subset of the primary key columns of the table); **HASH** partitions the table using the hash method (rows assigned to partitions using a hash function on values found in columns designated as the partitioning key); **LIST** partitions the table on lists of literal values from column (useful for controlling how individual rows map to specific partitions).

Available: Lists the columns whose values are available to be used in assigning rows to partitions.

Selected: Lists the column whose values are to be used in assigning rows to partitions.

To add a column to the partitioning definition, select it in Available Columns and click the Add (>) icon; to remove a column from the partitioning definition, select it in Selected Columns and click the Remove (<) icon. To move all columns from available to selected (or the reverse), use the Add All (>>) or Remove All (<<) icon. To move a column up or down in the partitioning definition, select it in Selected Columns and use the arrow buttons.

Subpartition By: The partitioning type to be used to create subpartitions within each range partition. Use the Available and Selected column lists select and deselect a column for subpartitioning.

Subpartition Templates pane

Specifies subpartitioning options for a partitioned table. The options depend on the subpartition type, and might include the following.

Hash Quantity: Hash subpartition quantity.

Tablespaces: Available and Selected tablespaces for storage of the data in a subpartition.

Subpartition Templates: Specifications (subpartition templates) to control the placement of rows in each subpartition. Click the Add (+) icon to add a subpartition template that is appropriate for the subpartition type.

Subpartition Details: For each subpartition template, specify a name and (if relevant) a value or set of values that is appropriate for the subpartition type.

Storage: Enables you to specify a tablespace for the subpartition.

Partition Definitions pane

Defines each partition for a partitioned table. The options depend on the partition type, and might include the following.

Partitions: Specifications to control the placement of rows in each partition. Click the Add (+) icon to add a partition specification that is appropriate for the partition type.

Partition Details: For each partition specification, specify a name and (if relevant) a value or set of values that is appropriate for the subpartition type.

Storage: Enables you to specify a tablespace for the partition.

Subpartitions: Enables you to specify subpartition information.

Index Organized Properties pane

Specifies options for an **index-organized table**, which is a table in which the rows, both primary key column values and nonkey column values, are maintained in an index built on the primary key. Index-organized tables are best suited for primary key-based access and manipulation.

PCTTHRESHOLD: The percentage of space reserved in the index block for an index-organized table row; must be large enough to hold the primary key. All trailing columns of a row, starting with the column that causes the specified threshold to be exceeded, are stored in the overflow segment. PCTTHRESHOLD must be a value from 1 to 50; the default is 50.

Key Compression: If this option is checked, key compression is enabled, which eliminates repeated occurrence of primary key column values in index-organized tables. In the box to the right of this field, you can specify the prefix length, which is the number of prefix columns to compress. (This value can be from 1 to the number of primary key columns minus 1; the default prefix length is the number of primary key columns minus 1.)

Include Column: Column at which to divide an index-organized table row into index and overflow portions. The primary key columns are always stored in the index. The Include Column can be either the last primary key column or any non-primary-key column. All non-primary-key columns that follow the Include Column are stored in the overflow data segment.

Mapping Table: If this option is checked, SQL Developer creates a mapping of local to physical ROWIDs and store them in a heap-organized table. This mapping is needed in order to create a bitmap index on the index-organized table. If the index-organized table is partitioned, then the mapping table is also partitioned and its partitions have the same name and physical attributes as the base table partitions.

Overflow: Specifications for the overflow segment. The options are the same as in the [Storage Options](#) dialog box.

External Table Properties pane

Specifies options for an **external table**, which is a read-only table whose metadata is stored in the database but whose data is stored outside the database. Among other capabilities, external tables enable you to query data without first loading it into the database.

Access Driver: The access driver of the external table. The access driver is the API that interprets the external data for the database: ORACLE_LOADER or ORACLE_DATAPUMP. You must specify the ORACLE_DATAPUMP access driver if you specify the AS subquery clause to unload data from one Oracle database and reload it into the same database or a different Oracle database.

Access Type: Type of data to be automatically converted during loads and unloads: BLOB or CLOB.

Default Directory: A default directory object corresponding to a directory on the file system where the external data sources may reside. The default directory can also be used by the access driver to store auxiliary files such as error logs.

Project Column: Determines how the access driver validates the rows of an external table in subsequent queries. **ALL** processes all column values, regardless of which columns are selected, and validates only those rows with fully valid column entries. If any column value would raise an error, such as a data type conversion error, the row is rejected even if that column was not referenced in the select list. **REFERENCED** processes only those columns in the select list.

The ALL setting guarantees consistent result sets. The REFERENCED setting can result in different numbers of rows returned, depending on the columns referenced in subsequent queries, but is faster than the ALL setting. If a subsequent query selects all columns of the external table, then the settings behave identically.

Reject Limit: The number of conversion errors can occur during a query of the external data before an Oracle Database error is returned and the query is aborted.

Access Parameters: Values to the parameters of the specific access driver for this external table.

Location Specifications: One or more external data sources. Each is usually a file, but it need not be. Oracle Database does not interpret this clause; it is up to the access driver to interpret this information in the context of the external data. Use the Add (+) icon to add each location specification.

Comment pane

Optional descriptive comment about the table.

DDL pane

You can review and save the CREATE TABLE statement that SQL Developer will use to create a new table or that reflects any changes you have made to the table properties. If you want to make any changes, go back to the relevant tabs and make the changes there.

To save the SQL statement to a script file, click **Save** and specify the location and file name.

When you are finished, click **OK**.

5.37 Storage Options

This dialog box is displayed if you click Storage Options in the Properties pane when creating or editing a table or an index. It enables you to override the default storage options.

Tablespace: Name of the tablespace for the table or index.

Pct Free: Percentage of space in each of the data blocks of the table or index reserved for future updates. You can enter a value from 0 through 99.

Pct Used: Minimum percentage of used space that Oracle maintains for each data block. A block becomes a candidate for row insertions when its used space falls below the Pct Used value. You can enter a value from 1 through 99.

Logging: <DEFAULT> means to use the Oracle Database default. ON means that the table creation and any subsequent direct loader (SQL*Loader) and direct-path INSERT operations against the table, partition, or LOB storage are logged in the redo log file. OFF means that these operations are not logged in the redo log file.

Ini Trans: Number of update transaction entries for which space is initially reserved in the data block header.

Max Trans: Number of transaction entries that could concurrently use data in a data block. This parameter has been deprecated. Oracle Database now automatically allows up to 255 concurrent update transactions for any data block, depending on the available space in the block.

Extents - Initial: Size of the first extent of the table or index. Specify K (kilobytes) or M (megabytes) for the unit associated with the number.

Extents - Next: Size of the next extent to be allocated to the table or index. Specify K (kilobytes) or M (megabytes) for the unit associated with the number.

Extents - Min: Minimum number of extents allocated when the table or index is created.

Extents - Max: Maximum number of extents allocated when the table or index is created. **Unlimited** (if checked) means that there is no maximum (and any specified maximum is ignored).

Pct Increase: Percentage that each extent grows over the previous extent.

Buffer Pool: <DEFAULT> means to use the Oracle Database default. KEEP means to put blocks from the segment into the Keep buffer pool; maintaining an appropriately sized Keep buffer pool lets Oracle retain the database object in memory to avoid I/O operations. RECYCLE means to put blocks from the segment into the Recycle pool; an appropriately sized Recycle pool reduces the number of objects whose default pool is the Recycle pool from taking up unnecessary cache space.

Free Lists: Number of free lists for each of the free list groups for the table or index. The default and minimum value for this parameter is 1, meaning that each free list group contains one free list.

Free List Groups: Number of groups of free lists for the table or index. The default and minimum value for this parameter is 1. Oracle uses the instance number of Real Application Clusters instances to map each instance to one free list group.

5.38 Create Trigger

The following information applies to a trigger, which is which is a stored PL/SQL block associated with a table, a schema, or the database, or an anonymous PL/SQL block or a call to a procedure implemented in PL/SQL or Java. The trigger is automatically executed when the specified conditions occur.

Schema: Database schema in which to create the trigger.

Name: Name of the trigger. Must be unique within the database.

Add New Source in Lowercase: If this option is checked, new text is entered in lowercase regardless of the case in which you type it. This option affects only the appearance of the code, because PL/SQL is not case-sensitive in its execution.

Trigger tab

Trigger Type: The type of object on which to create the trigger: TABLE, VIEW, SCHEMA, or DATABASE. (The remaining items depend on the type of trigger.)

Table Owner or View Owner: For a trigger on a table or a view, the name of the owner of the table or the view.

Table Name or View Name : For a trigger on a table or a view, the name of the table or the view.

Before or After: For a trigger on a table, select Before to cause the database to fire the trigger before executing the triggering event, or select After to cause the database to fire the trigger after executing the triggering event.

Statement Level or Row Level: For a trigger on a table, Statement Level fires the trigger once before or after the triggering statement that meets the optional trigger constraint defined in the WHEN condition; Row Level fires the trigger once *for each row* that is affected by the triggering statement and that meets the optional trigger constraint defined in the WHEN condition.

Insert, Update, Delete: For a trigger on a table or a view, Insert fires the trigger whenever an INSERT statement adds a row to a table or adds an element to a nested table; Update fires the trigger whenever an UPDATE statement changes a value in one of the columns specified in Selected Columns (or in any column if no columns are specified); Delete fires the trigger whenever a DELETE statement removes a row from the table or removes an element from a nested table.

Referencing - Old: For a trigger on a table, the correlation names in the PL/SQL block and WHEN condition of a row trigger to refer specifically to old value of the current row.

Referencing - New: For a trigger on a table, the correlation names in the PL/SQL block and WHEN condition of a row trigger to refer specifically to new value of the current row.

Available Columns: For a trigger on a table, lists the columns from which you can select for use in an Update trigger definition.

Selected Columns: For a trigger on a table, lists the columns used in an Update trigger definition.

When: For a trigger on a table, an optional trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger. This condition must contain correlation names and cannot contain a query.

Schema: For a trigger on a schema, the name of the schema on which to create the trigger.

Available Events: For a trigger on a schema or database, lists events from which you can select for use in the trigger definition.

Selected Events: For a trigger on a schema or database, lists events used in the trigger definition.

DDL tab

You can view a SQL CREATE statement that reflects the current definition of the object, or a SQL ALTER statement to modify an existing object to reflect your changes.

5.39 Create Type (User-Defined)

This dialog box is displayed when you right-click Types in the Connections navigator and select Create Type to create a user-defined type. After you complete the information in this dialog box and click OK, a SQL Worksheet is displayed in which you must specify the appropriate definition of the type.

Schema: Database schema in which to create the type.

Name: Name of the type. Must be unique within its schema.

Type: Select the type of data type to be created: array type, object type specification, object type specification and type body, or table type.

For more information about creating a user-defined type, see the CREATE TYPE statement in *Oracle Database SQL Language Reference*.

5.40 Create/Edit User

The user dialog box is used for creating a new database user or editing an existing database user. The user properties are grouped under several tabs.

To create or edit a database user, the user associated with your database connection must have the DBA role. You should also be familiar with the main concepts and techniques documented in *Oracle Database Administrator's Guide*.

User tab

Specifies general properties for the database user.

User Name: The user name string. For an existing user, this field is read-only; to change the name, you must drop the user and create a new user with the desired name.

New Password: Password string for the new user, or new password for an existing user. You must also type the same password string for **Confirm Password**.

Password Expired: If this option is checked, the password is marked as expired, and the user must change the password before being permitted to connect to the database.

Account Locked: If this option is checked, the user will not be permitted to connect to the database until a DBA user unlocks the account associated with this user.

Roles tab

Specifies roles to be granted to the user. For each role, you can check **Granted** to grant the role, **Admin** to permit the user to grant the role to other users, and **Default** to use the default settings for Granted and Admin.

For convenience, you can click buttons to affect all settings (Grant All, Revoke All, Admin All, Admin None, Default All, Default None); then, you can specify other settings for individual roles.

System Privileges tab

Specifies privileges to be granted to the user. For each privilege, you can check **Granted** to grant the privilege, and **Admin Option** to permit the user to grant the privilege to other users.

For convenience, you can click buttons to affect all settings (Grant All, Revoke All, Admin All, Admin None); then, you can specify other settings for individual privileges.

Quotas tab

Specifies disk usage limits on specified tablespaces for the user. If you check Unlimited, there is no disk usage limit on the tablespace.

SQL tab

Displays the SQL statements that SQL Developer will use to create (after executing a CREATE USER statement) a new user or to edit an existing user. This display is read-only; if you want to make any changes, go back to the relevant tabs and make the changes there.

5.41 Create/Edit User Defined Report

The following information applies to a user-defined report. For information about how to create a user-defined report, as well as examples of creating such reports, see [Section 1.11.15, "User Defined reports"](#).

Add Child: Add a child report in this report.

Test: Tests the report definition by running it in a separate window. This feature enables you to test the report before creating it.

Details tab

Name: Name of the user-defined report.

Style: Report style: *Table* (default), *Code* (formats the code in the output), *Chart* (bar or pie chart; see [Section 1.11.15.1, "User-Defined Report Example: Chart"](#) for an example), *Gauge* (dial or status meter; see [Section 1.7.8, "Gauges: In the SQL Worksheet and User-Defined Reports"](#)), *plsql-dbms_output* (dynamic HTML; see [Section 1.11.15.2, "User-Defined Report Example: Dynamic HTML"](#) for an example), or *Script* (executable script).

Description: Optional description of the report.

ToolTip: Optional tooltip text to be displayed when the mouse pointer stays briefly over the report name in the Reports navigator display.

SQL Statement: The complete SQL statement for retrieving the information to be displayed in the user-defined report. As a trivial example, the statement SELECT user "Current User" FROM DUAL displays Current User as the heading and the name of the user associated with the current database connection.

Suggestion: Look at the SQL statements for various SQL Developer-supplied reports; check the Messages - Log pane below the report results, or click the SQL icon under the Report Results tab.

Binds tab

Name: Name of the bind variable.

Prompt: String displayed when the user is prompted to enter a value. Example: *Table name*

Default: Default value if the user does not enter a value at the prompt. To accept the Oracle SQL value, specify NULL_VALUE.

ToolTip: Optional tooltip text to be displayed when the mouse pointer stays briefly over the bind variable name.

Chart Details tab

Available if the report type is Chart.

Chart Type: Bar chart with horizontal or vertical bars, or pie chart.

3D Graph: *True* for a three-dimensional appearance; *False* for a two-dimensional appearance.

Gradient Effect: *True* for a gradient effect; *False* for no gradient effect.

Chart Style: Thematic name for the overall appearance of the chart.

Show Grid: *True* to show the grid lines; *False* to hide the grid lines.

Show Legend: *True* to show the chart legend; *False* to hide the chart legend.

Gauge Details tab

Available if the report type is Gauge.

Gauge Type: Dial (like a fuel gauge in an automobile) or Status meter (bar representation).

Query Based: *True* if the minimum, maximum, low, and high values are specified in the SQL query; *False* to specify the minimum, maximum, low, and high values in the remaining fields.

Min: Minimum value displayed on the gauge.

Max: Maximum value displayed on the gauge.

Low: "Low" value; usually greater than Min and less than High.

High: "High" value; usually greater than Low and less than Max.

5.42 Create/Edit User Defined Report Folder

The following information applies to a folder for organizing user-defined reports. Each folder can contain reports and other folders (subfolders). For example, you can create a folder named Sales, and then under that folder create folders named Sales by District and Sales by Product.

For information about how to create user-defined reports and folders for these reports, see [Section 1.11.15, "User Defined reports"](#).

Name: Name of the folder.

Description: Optional description of the folder.

ToolTip: Optional tooltip text to be displayed when the mouse pointer stays briefly over the folder name in the Reports navigator display.

5.43 Create/Edit View

The view dialog box is used for creating or editing a view or materialized view. You can use the SQL Query tab or a series of panes to specify the query part of the view definition, and you can use one or more other panes (depending on the type of view) for other parts of the definition.

If you click OK before you are finished creating or editing the view, right-click the view name in the Connections navigator, select Edit, and continue creating or editing the view.

Schema: Database schema in which to create the view.

Name: Name of the view. Must be unique within a schema.

Advanced: If this option is checked, the dialog box changes to include a pane that provides an extended set of features for creating the view.

SQL Query tab or pane

As a tab (if you did not check the Advanced box), it contains the SQL code for the query part of the view definition, using the SELECT and FROM keywords and usually a WHERE clause with whatever syntax is needed to retrieve the desired information.

As a pane (if you checked the Advanced box), it presents options for building specific parts of the query.

For example, the following query, from the [Create a View](#) tutorial topic, selects columns from the PATRONS and TRANSACTIONS tables, ordering them first by values in the PATRON_ID column in the PATRONS table and then by values in the TRANSACTION_TYPE column in the TRANSACTIONS table. The result is a listing by patron ID of all patrons who had transactions, and for each listed patron the transaction information listed by transaction type

```
CREATE VIEW patrons_trans_view AS
  SELECT p.patron_id, p.last_name, p.first_name,
         t.transaction_type, t.transaction_date
  FROM patrons p, transactions t
  WHERE p.patron_id = t.patron_id
  ORDER BY p.patron_id, t.transaction_type;
```

SQL Parse Results: If you click Test Syntax, displays any SQL syntax errors, or displays a message indicating no errors if there are no syntax errors.

Revert: Cancels any edits you have made in the SQL Query box, and displays the contents of the box before these edits.

Test Syntax: Checks the statement in the SQL Query box for any SQL syntax errors.

Test Query: Displays a dialog box that runs the SQL query and indicates the result. If there is an error in the query, such as an invalid character or a missing expression, the error is displayed.

Quick-Pick Objects pane

Specifies objects that you can use in the SELECT, FROM, and WHERE clauses of the view definition. Identify the tables and views on which this view is based, and the columns in those tables and views that are used in the definition of this view. To see the results of your quick-pick specification, either check Auto-Query or click Query.

Schema: Database schema containing the objects to be selected.

Type Filter - Filter Types: Enables you to limit the display of objects available for selection to certain types of database objects (for example, to show only tables or views).

Name Filter: Enables you to limit the display of objects available for selection according to a character string in the name, with the percent sign (%) as a wildcard character. For example, to limit the display of available tables and views to those whose names start with the string EM, specify the following name filter: EM%

Auto-Query: If this option is enabled, the display of available objects is automatically refreshed when you specify or change the Type Filter or Name Filter value.

Query: Refreshes the display of available objects based on the Type Filter and Name Filter values.

Available: Lists the objects (typically, tables and views in a hierarchical display) from which you can select objects to use in the SELECT, FROM, and WHERE clauses of the view definition.

Selected: Lists the objects (typically, columns) that you can use in the SELECT, FROM, and WHERE clauses of the view definition.

To add an object as selected, select it in Available and click the Add (>) icon; to remove an object as selected, select it in Selected and click the Remove (<) icon. To move all objects from selected to available, use the Remove All (<<) icon. To move an object up or down in the selected list, select it in Selected and use the arrow buttons.

For the example in [DDL tab or pane](#), select the DEPTNO and SAL columns from the EMP table.

FROM Clause pane

Specifies the tables and views that you can use in the FROM clause of the view definition.

Type Filter - Filter Types: Indicates whether or not you have limited the types of database objects to be displayed in the Available List (by clicking the **Filter Types** and selecting any combination of Materialized Views, Tables, and Views).

Naming Filter: Substring for limiting object names to appear as available. For example, to display only objects with names that start with EM, specify EM% (with the percent sign as a wildcard character).

Auto-Query: If this option is enabled, the display of available objects is automatically refreshed when you specify or change the Type Filter or Name Filter value.

Query: Refreshes the display of available objects based on the Type Filter and Name Filter values.

Available: Lists the tables and views that are available to be selected for use in the FROM clause of the view definition.

Selected: Lists the tables and views that you can use in the FROM clause of the view definition.

To add an object as selected, select it in Available and click the Add (>) icon; to remove an object as selected, select it in Selected and click the Remove (<) icon. To move all objects from available to selected, use the Add All (<<) icon; to move all objects from selected to available, use the Remove All (<<) icon.

If you attempt to remove any objects that have dependencies in the SQL query, a Confirm Remove box warns you that the dependencies will be removed with the FROM expression; to cancel the remove operation, click **No**.

Alias: Alias for the table or view.

For the example in [DDL tab or pane](#), select the EMP table.

Join: If you select two tables and click this button, the [Edit Join](#) dialog box is displayed.

SELECT Clause pane

Specifies objects that you can use in the SELECT clause of the view definition.

SELECT List: Lists the objects (typically, columns) that you can currently use in the SELECT clause. To add an object, click the Add (+) icon; to delete an object, select it and click the Delete (X) icon; to move an object up or down in the view definition, select it and use the up-arrow and down-arrow buttons.

Note: After you add an object, **to add another object**, click the Add (+) icon.

Expression: Column name or an expression. For expressions, you can type them, or you can use the **Expression Palette** to add object names and function names.

Validate: Checks the validity of the Expression entry.

For the example in [DDL tab or pane](#), select DEPTNO column and the MIN(emp.sal) and MAX(emp.sal) functions.

WHERE Clause pane

Specifies the WHERE clause of the view definition.

WHERE: The text of the WHERE clause, without the WHERE keyword. You can type the text completely; or you can type some of the text and use the **Expression Palette** to add object names, function names, and operators.

Example (from the [Create a View](#) tutorial exercise): `p.patron_id = t.patron_id`

GROUP BY Clause pane

Specifies a clause to be used to group the selected rows based on the value of columns for each row and return a single row of summary information for each group. The GROUP BY clause groups rows but does not guarantee the order of the result set; to order the groupings, use the ORDER BY clause.

Available: Lists the tables and views, and the columns in each, that are available to be selected for use in the GROUP BY clause of the view definition.

Selected: Lists the tables and views, and the columns in each, that you can use in the GROUP BY clause of the view definition.

To add an object as selected, select it in Available and click the Add (>) icon; to remove an object as selected, select it in Selected and click the Remove (<) icon. To move all objects from available to selected, use the Add All (<<) icon; to move all objects from selected to available, use the Remove All (>>) icon.

HAVING Clause pane

Specifies an expression that must be satisfied for rows to be processed by the GROUP BY clause. For example, `HAVING MIN(salary) < 30000` causes the GROUP BY clause to consider only rows where the minimum value of the relevant salary values is less than 30000.

HAVING: You can type the complete expression text, or you can use the **Expression Palette** to add object names, function names, and operators to the expression text.

ORDER BY Clause pane

Specifies one or more columns or column expressions whose values will be used to sort the results returned by the view. Without an ORDER BY clause, no guarantee exists that the same query executed more than once will retrieve rows in the same order.

ORDER BY List: Lists the objects (typically, columns) that you can currently use in the ORDER BY clause. To add an object, click the Add (+) icon; to delete an object, select it and click the Delete (X) icon; to move an object up or down in the view definition, select it and use the up-arrow and down-arrow buttons.

Note: After you add an object, **to add another object**, click the Add (+) icon.

ORDER BY Expression Filter: For each column or column expression, you can type the text completely into the **Expression** box; or you can type some of the text and use the **Expression Palette** to add object names, function names, and operators.

Validate: Tests the validity of the syntax for the expression.

Order: ASC for ascending (expression values sorted in ascending order); DESC for descending (expression values sorted in descending order).

Nulls Ordering: NULLS FIRST to have null expression values appear before non-null values; NULLS LAST to have null expression values appear after non-null values. ("Before" and "after" positions are determined by the Order value.)

View Information or Materialized View Properties pane

Options for a standard view:

Restrict Query: If this option is checked, you can enable one of the following options

- **Read Only:** Prevents the view from being used to add, delete, or change data in the underlying table or tables.
- **Check Option:** If this option is checked, it prohibits any changes to the underlying table or tables that would produce rows that are not included in this view.

Force on create: If this option is checked, the view is created even if it has errors in its definition. This option is useful if you want to create the view regardless of any errors, and go back and correct the errors later. If this option is not checked, the view is not created if its definition contains any errors.

Options for a materialized view:

Refresh Options:

Method: The method of refresh operation to be performed:

- **Complete Refresh:** Executes the defining query of the materialized view, even if a fast refresh is possible.
- **Fast Refresh:** Uses the incremental refresh method, which performs the refresh according to the changes that have occurred to the master tables. The changes for conventional DML changes are stored in the materialized view log associated with the master table. The changes for direct-path INSERT operations are stored in the direct loader log.
- **Force Refresh:** Performs a fast refresh if one is possible; otherwise, performs a complete refresh.
- **Never:** Do not perform refresh operations.

When: The type of refresh operation to be performed:

- **On Demand:** Performs a refresh when one of the DBMS_MVIEW refresh procedures is called.
- **On Commit:** Performs a fast refresh whenever the database commits a transaction that operates on a master table of the materialized view. This may increase the time taken to complete the commit, because the database performs the refresh operation as part of the commit process.

- **Specify:** Performs refresh operations according to what you specify in the *Start on* and *Next* fields.
- **Never:** Does not perform a refresh operation.

Type: Refresh type, which determines the type of materialized view:

- **Primary Key:** Creates a primary key materialized view, which allows materialized view master tables to be reorganized without affecting the eligibility of the materialized view for fast refresh.
- **Row ID:** Creates a rowid materialized view, which is useful if the materialized view does not include all primary key columns of the master tables.

Start on: Starting date and time for the first automatic refresh operation. Must be in the future.

Next: Time for the next automatic refresh operation. The interval between the *Start on* and *Next* times establishes the interval for subsequent automatic refresh operations. If you do not specify a value, the refresh operation is performed only once at the time specified for *Start on*.

Constraints: If this option is checked, more rewrite alternatives can be used during the refresh operation, resulting in more efficient refresh execution. The behavior of this option is affected by whether you select *Enforced* or *Trusted*.

Enforced: Causes only enforced constraints to be used during the refresh operation.

Trusted: Enables the use of dimension and constraint information that has been declared trustworthy by the database administrator but that has not been validated by the database. If the dimension and constraint information is valid, performance may improve. However, if this information is invalid, then the refresh procedure may corrupt the materialized view even though it returns a success status.

Materialized View Options:

Parallel: If this option is checked, parallel operations will be supported for the materialized view, and you can specify a number for the default degree of parallelism for queries and DML on the materialized view after creation.

Enable Cache: If this option is checked, the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This setting is useful for small lookup tables. If this option is not checked, the blocks are placed at the least recently used end of the LRU list.

Build Type: Specifies when to populate the materialized view. **Immediate** indicates that the materialized view is to be populated immediately. **Deferred** indicates that the materialized view is to be populated by the next refresh operation. If you specify *Deferred*, the first (deferred) refresh must always be a complete refresh; until then, the materialized view has a staleness value of unusable, so it cannot be used for query rewrite.

Enable Query Rewrite: If this option is checked, the materialized view is enabled for query rewrite, an optimization technique that transforms a user request written in terms of master tables into a semantically equivalent request that includes one or more materialized views.

Prebuilt Option: If this option is checked, an existing table is registered as a preinitialized materialized view. This option is particularly useful for registering large materialized views in a data warehousing environment. The table must have the same name and be in the same schema as the resulting materialized view, and the table should reflect the materialization of a subquery. **Reduced Precision** authorizes the loss

of precision that will result if the precision of the table or materialized view columns do not exactly match the precision returned by subquery. **No Reduced Precision** requires that the precision of the table or materialized view columns match exactly the precision returned by subquery, or the create operation will fail.

Index Storage Options:

Use Index: If this option is checked, a default index is created and used to speed up incremental (fast) refresh of the materialized view. If this option is not checked, this default index is not created. (For example, you might choose to suppress the index creation now and to create such an index explicitly later.)

Use Tablespace: If this option is checked, you can specify the tablespace in which the materialized view is to be created. If this option is not checked, the materialized view is created in the default tablespace of the schema containing the materialized view.

DDL tab or pane

You can view a SQL CREATE statement that reflects the current definition of the object, or a SQL ALTER statement to modify an existing object to reflect your changes.

To save the SQL statement to a script file, click **Save** and specify the location and file name.

5.44 Create XML Schema

This dialog box enables you to specify the URL of an XML schema that can be associated with XML document instances.

Schema: Name of the schema in which to create the XML schema object.

Name: URL of the XML schema.

5.45 Configure Extension

This dialog box, which is displayed if you click Configure for Versioning Support in the Extensions preferences pane, enables you to select from among available versioning support extensions for SQL Developer. For information about using versioning with SQL Developer, see [Section 1.10](#).

If you change any existing settings, you will need to restart SQL Developer.

5.46 Configure File Type Associations

This dialog box, which is displayed the first time you start SQL Developer, enables you to associate certain file types with SQL Developer. If a file type is associated with SQL Developer, files with that type's extension will automatically be opened by SQL Developer when you double-click the file name. Any previous association for that file type is replaced.

If you do not associate a file type with SQL Developer, any existing association for that file is unchanged.

After you close this box, you can change the associations for these file types and many others by clicking **Tools** and then **Preferences**, and selecting **File Types** (see [Section 1.12.8, "File Types"](#)).

5.47 Copy Columns

This dialog box is displayed if you click the Copy Columns icon when specifying column definitions for a table.

Schema: The schema that owns the table from which to copy column definitions.

Table: The table within the selected schema.

Columns: A list of the columns in the table. Select one or more columns to be copied into the table that you are creating or editing, and click **OK**.

5.48 Custom Filters

This dialog box is displayed if you right-click and select Customize Filters in the History tab for a SQL Trace (.trc) file. You can modify an existing filter or create a new filter.

Filter List: Names of the available filters. To edit an existing filter, select its name; the details for that filter are displayed in the dialog box. To remove an existing filter, select its name and click **Remove**.

To create a new filter, click **Add** and specify the filter name.

To restore the filters to those at SQL Developer installation, click **Restore Defaults**. This deleted any filters that have been added since installation.

Simple Expression: Create the filter by selecting a column and operator and by specifying a value. To select a value from a list, click **Insert**.

Complex Expression: Create the filter by entering a complex expression.

5.49 Database Copy (Schema Objects)

This dialog box is displayed if you click Tools, then Database Copy. Specify the type of operation, and the connections for the source and destination schemas. All database objects are copied from the source schema to the destination schema, subject to any restrictions depending on the type of operation, which determines the behavior if objects of the same name exist in the destination schema.

Source/Destination pane

Source Connection: Database connection for the schema from which to copy the objects.

Destination Connection: Database connection for the schema to which to copy the objects

Create Objects: Copies the objects to new objects in the destination schema only if an existing object of that type with the same name does not already exist.

Truncate Objects: Deletes existing rows in any existing table with the same name, and then loads rows from the source.

Drop Objects: Drops any existing table with the same name, and then creates and loads it from the source.

Copy Summary pane

You can review the SQL statements that will be used to perform the copy operation according to your specifications.

To go back and make any changes, click **Back**.

To perform the copy operation, click **Finish**. After the copy operation completes, a log file is displayed.

5.50 Database Schema Differences

This interface is displayed if you click Tools, then Database Diff. You can find differences between objects of the same type and name (for example, tables named CUSTOMERS) in two different schemas, and optionally update the objects in one schema (destination) to reflect differences in the other schema (source).

Use the [Source/Destination pane](#) to specify the source and destination database connections. Database objects in the schemas associated with these connections will be compared. The schemas for the source and destination connections can be in the same database or different databases.

Source/Destination pane

Source Connection: Database connection for the source schema (the schema in which selected objects are to be compared with objects in the destination schema).

Destination Connection: Database connection for the database that contains the destination schema (the schema containing one or more objects of the same type and name as those selected in the source schema). The selected connection can be the same as, or different from, the connection for the source schema.

Diff Objects: Check the types of objects that you want to be compared in the source and destination connections. You can click *Toggle All* to check and uncheck all individual types. You must select at least one object type.

Proceed to Summary: If this option is checked, clicking Next takes you directly to the [Diff Summary pane](#).

Specify Objects pane

You can limit the types or objects, and the objects within selected types, for the comparison operation.

Object Type: Select All for all object types, or a specific type of object.

Go: Click Go to display a list of objects that meet the selection criteria for the selected connection. Use the arrow keys to move selected objects or all objects from the available objects box to the selected objects box.

Diff Summary pane

You can review the information that will be used to compare the source and destination connections, according to your specifications.

To go back and make any changes, click **Back** as needed.

To perform the comparison, click **Finish**. The results are displayed in a *Diff Report* window, where you can see the DDL statements to update the objects in the destination schema to reflect differences in the source schema. To create a file containing these DDL statements, click the **Generate Script** icon in that window. To toggle the display between all objects and only those objects with differences, click **Show Equal Objects**.

5.51 DDL Panel for Creating or Editing an Object

You can review and save the SQL statement that SQL Developer will use to create or edit the object, to reflect any changes you have made to the object's properties. If you want to make any changes, go back to the relevant panels and make the changes there.

To save the SQL statement to a script file, click **Save** and specify the location and file name.

5.52 Debugger - Attach to JPDA

This dialog box is displayed when you right-click a database connection name and select Remote Debug. Use this dialog box if you are using the Sun Microsystems's Java Platform Debugger Architecture (JPDA) and you would like the debugger to listen so that a debuggee can attach to the debugger. For more information about remote debugging, see [Section 1.6.2, "Remote Debugging"](#).

Host: Name or IP address of the remote host on which SQL Developer should listen for the database to connect.

Port: Listening port number on the remote host. You can choose any valid port number that is not in use by another process.

Timeout: The number of seconds that SQL Developer will wait for the remote database to make a debugging connection.

Don't Show Dialog Box Before Connecting: If this option is checked, this dialog box will not be displayed before future connections for remote debugging.

5.53 Deploy or Import Application

Use this wizard to deploy or import an Application Express application into a specified target schema.

Deploy to Connection or Specify File to Import

Choose Connection to Deploy Application: For a deploy operation, specify the database connection for the target schema into which to deploy the application.

Specify File to Import: For an import operation, specify the location and name of the SQL file containing the exported application (usually the output of a previous "export application" operation).

Choose Import Options

Specifies options for the application to be deployed or imported.

Workspace: Name of the Application Express workspace.

Parse As Schema: Schema against which all of the application's SQL and PL/SQL will be parsed.

Application Name: Name of the application.

Application Alias: Alias for the application. It is recommended that you never hard code the application ID into your application, but instead use the application alias or a built-in substitution string (such as APP_ID and APP_ALIAS).

Build Status: RUN_ONLY or RUN_AND_BUILD

Application ID: Specify whether to have an application ID assigned automatically, to use an existing listed ID, or to specify a new ID. Use these options to avoid application

ID conflicts, such as when you need to have two versions of the same application in the same instance. For example, you might be migrating an application to a production instance and still need to maintain the development version.

ID Currently Used by and Overwrite: If the specified Application ID is currently used by another application, you can enable **Overwrite** to have the application ID instead associated with the application being deployed or imported.

Summary

Displays the selected options for the application to be deployed or imported. To make any changes, click Back. To perform the operation, click Finish.

5.54 Describe Object Window

This window is displayed when you select a database object name in the SQL Worksheet, right-click, and select **Describe**. The information is read-only, and is displayed using tabs that are appropriate for the type of object.

For example, if the display is for a table, the information displayed is similar to that in the [Create/Edit Table \(with advanced options\)](#) dialog box.

5.55 Edit/View Value (Table Column or Other Data)

This dialog box enables you to view and edit data in a cell in the table Data grid or in some other grid display (for example, to edit the value of a single column within a row). If you are permitted to modify the data, you can change the data value and then click **OK**. (If you are not permitted to modify the data, the Value display is read-only.)

The specific options for editing table data available depend on the data type of the column associated with that cell in the grid. For example, for BLOB data you can invoke the external editor associated with the MIME type and file extension (see the preferences for [External Editor](#)).

5.56 Enter Bind Values

This dialog box enables you to enter values for each bind variable. If the NULL option is checked, you cannot enter a value in this dialog box.

5.57 Erase from Disk

This dialog box asks you to confirm your action if you select one or more files in the Files navigator and press the Delete key. To perform the deletion, click **Yes**; to cancel the deletion, click **No**.

5.58 Error Writing to Export File

This box is displayed if you tried to export table data to a file, but the directory or folder path does not exist.

Click **OK** to close the box, then enter a valid path in the Export Data dialog box and click **Apply**.

5.59 Export (Database Objects and Data)

This interface is displayed when you click Tools and then Database Export to export 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 a file containing SQL data definition language (DDL) statements to create these objects. To specify options for the export operation, use the [Types to Export pane](#). To specify the objects or types of objects to export, use the [Specify Objects pane](#).

In several panes, if you select **Proceed to summary**, clicking Next takes you to the [Export Summary pane](#).

Source/Destination pane

Specify the output file, the database connection, and options that affect the content (DDL statements) of the output file.

File: Specify the name of the script file to contain the DDL statements for creating the objects to be exported and the INSERT statements if you will also be exporting data (for example, `my_tables.sql`). You can click **Browse** to select a directory for this file. (The default file path for export operations is specified in the SQL Developer user preferences for [Database](#).)

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

DDL Options: Options that affect the DDL statements in the output file:

- **Show Schema:** If this option is checked, the schema name is included in CREATE statements. If this option is not checked, the schema name is not included in CREATE statements, which is convenient if you want to re-create the exported objects under a schema that has a different name.
- **Storage:** If this option is checked, any STORAGE clauses in definitions of the database objects are preserved in the exported DDL statements. If you do not want to use the current storage definitions (for example, if you will re-create the objects in a different system environment), uncheck this option.
- **Terminator:** If this option is checked, a line terminator character is inserted at the end of each line.
- **Pretty Print:** If this option is checked, the statements are attractively formatted in the output file, and the size of the file will be larger than it would otherwise be.
- **Include BYTE Keyword:** If this option is checked, column length specifications refer to bytes; if this option is not checked, column length specifications refer to characters.
- **Add Force to Views:** If this option is checked, the FORCE option is added to any CREATE VIEW statements, causing each view to be created even if it contains errors.
- **Include Drop Statement:** If this option is checked, a DROP statements is included before each CREATE statement, to delete any existing objects with the same names. However, you may want to uncheck this option, and create a separate drop script that can be run to remove an older version of your objects before creation. This avoids the chance of accidentally removing an object you did not intend to drop.
- **Include Grants:** If this option is checked, GRANT statements are included for any grant objects on the exported objects. (However, grants on objects owned by the SYS schema are never exported.)

- **Automatically Include Dependent Objects:** If this option is checked, objects that are dependent on the objects specified for export are also exported. For nonprivileged users, only dependent objects in their schema are exported; for privileged users, all dependent objects are exported.

Types to Export pane

Specify object types to be exported and options for the export operation.

Object Types: Check the types of objects that you want to export. You can click *Toggle All* to check and uncheck all individual types. You must select at least one object type. Note also the following:

- **Dependencies** (under Tables): If this option is checked, constraints for each table are defined as inline constraints in the CREATE TABLE statement; and if any indexes or triggers exist for a table, they are also included in the CREATE TABLE statement.
- **Constraints:** If this option is checked, any constraints for each table are defined in separate ALTER TABLE statements instead of in the CREATE TABLE statement.
- **Data:** If this option is checked, statements are included to insert the data for an exported table or view. If this option is not checked, statements are not included to insert the data for an exported table or view; that is, only the DDL statements are included. If you check Data, statements are included to insert all data in all tables in the selected schema, unless you use the Filter Data tab to limit the data to be migrated.

Specify Objects pane

You can limit the types or objects, and the objects within selected types, for the export operation.

Object Type: Select All for all object types, or a specific type of object.

Go: Click Go to display a list of objects that meet the selection criteria for the selected connection. Use the arrow keys to move selected objects or all objects from the available objects box to the selected objects box.

Specify Data pane

You can limit the data for the export operation.

Go: Click Go to display a list of available tables, and use the arrow keys to move selected tables or all tables from the available box to the selected box.

Then, select a table, enter the filter text (a WHERE clause without the WHERE keyword), and click **Apply Filter**.

Export Summary pane

You can review the information that will be used to create the output file, which will contain statements to export database objects and data according to your specifications.

To go back and make any changes, click **Back** as needed.

To create the output file, click **Finish**. The file is also displayed in a SQL Worksheet window, where you can run it as a script and perform other operations.

5.60 Export: Advanced Data Filter

This dialog box is displayed when you click **Advanced** in the [Export \(Database Objects and Data\)](#) dialog box.

Schema: Select the name of the schema to see the available objects on which you can specify a filter.

Filter: A WHERE clause specifying the condition or conditions for filtering data from the selected object.

Apply Filter: Click to apply the specified filter.

When you are finished applying any filters, click **Apply**.

5.61 Export Error

This dialog box is displayed when you tried to export some or all objects of one or more types of database objects to a file containing SQL statements, but did not include some essential information, which might include one or more of the following:

- The database connection. For **Connection**, select the database connection from which the objects will be exported.
- The name of the output file. Look at the **Options** tab, and be sure that you specified a file.
- One or more objects or types of objects. Look at the **Objects** tab, and be sure that you selected (checked) at least one object or type of object.

5.62 Export Data

This dialog box prompts you to specify the location and name of a text file to contain the output of the export operation, such as data values during a debug operation.

5.63 Export Table Data

This dialog box is displayed when you right-click a table name, a table data display, a SQL Worksheet result set, or report output, and select **Export** and then an export format. You can export some or all of the data to a file or to the system clipboard. To restrict the output to specified columns, use the [Columns](#) tab. To restrict the output based on a WHERE clause condition, use the [Where](#) tab.

Format tab

Format: Determines the format of entries written in the specified output file: *CSV* for comma-separated values including a header row for column identifiers, *Fixed* for a file where records are the same byte length, *HTML* for an HTML file containing a table with the data, *Insert* for SQL INSERT statements, *Loader* for a SQL*Loader control file, *Text* for a text file, *TTBulkCP* for a data files to be used with the TimesTen ttbulkcp command line utility, *XLS* for a Microsoft Excel worksheet, or *XML* for XML tags and data.

For exporting *large tables to Microsoft Excel* files:

- If you encounter problems, try adding the following line to the sqldeveloper.conf file to increase heap size and then restarting SQL Developer: `AddVMOption -Xmx1024M`

- If the number of table rows exceeds 65,536, SQL Developer writes the rows to multiple worksheets within the .xls file.

Output: File writes the output to a file that you specify; Clipboard places the output on the system clipboard, so that you can paste it into a file, a command line, or other location appropriate for the format.

File: If the output is to a file, click **Browse** to select the directory or folder and to specify the file name and extension. The file path is then placed in the File box. (The default file path for export operations is specified in the SQL Developer user preferences for [Database](#).) Standard file extensions include .sql for Insert format, .xls for Microsoft Excel format, .xml for XML format, .ctl for SQL LOADER format, and .csv for CSV format.

Columns tab

You can specify whether the output should include data from all columns or just from the checked columns. (Note: For CLOB columns, only the first 32 KB of any CLOB is exported.)

Where tab

You can restrict the output by entering a valid WHERE clause for a query on the table, without the WHERE keyword. For example, to restrict the exported data to rows where a column named RATING contains a value greater than 5, specify: `rating > 5`

5.64 External Locator Configuration

This dialog box is displayed if you click External Locator Configuration when creating a CVS repository. Specify the information required to connect to the remote repository when the method by which the client will gain access to and authenticate against the server is External.

Set Remote Shell: If this option is checked, external repositories are accessed through a remote shell utility, usually rsh (the default) or ssh.

Set Remote Shell: If this option is checked, you specify the name of the CVS program on the remote server. (It is unlikely to need to be changed from the default, and should only be changed in cooperation with the administrator of the CVS remote server.)

5.65 External Tools

This dialog box is displayed when you click Tools and then External Tools. It displays information about user-defined external tools that are integrated with the SQL Developer interface.

Find Tools: Checks for any tools that Oracle offers for your consideration, and adds them to the list if they are not already included.

New: Starts a wizard for defining a new external tool (see [Section 5.66, "Create/Edit External Tool"](#)).

Edit: Displays a dialog box for editing the selected external tool (see [Section 5.66, "Create/Edit External Tool"](#)).

5.66 Create/Edit External Tool

This interface is displayed as a wizard if you are creating a new external tool, and as a dialog box if you are editing an existing external tool (see [Section 5.65, "External Tools"](#)).

External Program Options

Program Executable: Path of the program executable for the tool.

Arguments: Arguments (parameters) to be passed to the program. You can click **Insert** to insert a macro for the argument (see [Section 5.71, "Insert Macro"](#)).

Run Directory: Directory in which to run the program. You can click **Insert** to insert a macro for the directory (see [Section 5.71, "Insert Macro"](#)).

Command Sample: A read-only sample display of the command to run the program.

Display Options

Specify how the external tool should appear when displayed in menu or toolbar items.

Caption for Menu Items: The text string that will appear for any menu item that calls the external tool. To indicate the mnemonic character, use the ampersand before the character. For example: &Mytool for the "M" to be underlined and used as the mnemonic

ToolTip Text: Text for the tooltip to be displayed when the mouse pointer hovers over the icon for the tool in a toolbar.

Icon Location: File path of the icon associated with the tool. Click **Browse** to specify a graphics file, or **Use Default** to use the default icon (if you previously specified a nondefault icon).

Preview: A read-only display of the menu item and its associated icon.

Integration Options

Specify how the external tool will be integrated with SQL Developer.

Add Items to Menus: Check any menus on which you want to include an item for this tool.

Add Buttons to Toolbars: To add the icon for this tool to the SQL Developer main toolbar, check **Main Toolbar**.

After Tool Exits: To have SQL Developer reload any open files after the tool exits, check **Reload Open Files**.

Availability Options

Specify when the external tool is enabled. In contexts where the tool is not enabled, its menu item and icon are grayed out.

Always: Makes the tool always available.

When a File is Selected or Open in the Editor: Makes the tool available only when a file is selected or open, such as when the SQL Worksheet is open.

When Specific File Types are Selected: Makes the tool available only when files of the specified type or types are selected. Use the arrow buttons to move desired types from **Available Types** to **Selected Types**.

5.67 Choose Offline Options

This dialog box is displayed when you click Tools, then Migration, then Third Party Database Offline Capture, then Create Database Capture Scripts. It specifies options for creating an offline capture properties (.ocp) file, which you can later load and run by clicking Tools, then Migration, then Third Party Database Offline Capture, then Load Database Capture Script Output.

Output Directory: Converted model containing tables whose data is to be moved to the corresponding Oracle database tables.

Generate for: *Windows Batch File* generates a .bat file to be run on Windows systems; *Linux Shell Scripts* generates .sh files to be run on Linux systems.

For a MySQL migrations, if you generate .sh files, you must also execute the following command to make the .sh files executable and the .ocp file writable:

```
chmod 755 *
```

Platform: The IDM DB2, MySQL, Microsoft SQL Server, or Sybase Adaptive Server version for which to generate the scripts.

5.68 Edit Join

This dialog box lets you edit the join specification for a join view.

Swap: Reverses the order of the tables.

Join Type:

Natural: If this option is checked, a natural join will be performed. A natural join is based on all columns in the two tables that have the same name. It selects rows from the two tables that have equal values in the relevant columns. When specifying columns that are involved in the natural join, do not qualify the column name with a table name or table alias.

On: Using the ON clause to specify a join condition lets you specify join conditions separate from any search or filter conditions in the WHERE clause.

Using: When you are specifying an equijoin of columns that have the same name in both tables, the USING *column* clause indicates the columns to be used. You can use this clause only if the join columns in both tables have the same name. Within this clause, do not qualify the column name with a table name or table alias. In an outer join with the USING clause, the query returns a single column which is a coalesce of the two matching columns in the join.

5.69 Feature Required

This dialog box is displayed if you try to use a SQL Developer feature that requires the licensing of the specified feature for Oracle Database. If you do not have a license for the specified feature, you must click **No**.

If you have a license for the feature on the database or databases on which you plan to use the feature this time, you can click **Yes**. If you have a license for the feature on all databases on which you plan to use the feature now and in the future, you can enable **Skip This Message Next Time** and click **Yes**.

To purchase any required license, contact your Oracle sales representative or authorized Oracle Reseller, or go to the Oracle Store to buy online.

5.70 Filter

This dialog box is displayed when you right-click a connection node or an object type node (such as Tables) in the Connections navigator and select **Apply Filter**. Use this box to limit the number of objects of that type that are displayed, according to one or more filter criteria that you specify. For each criterion, specify the following:

- Criterion name (list always includes NAME; other criteria depend on the object type)
- Operator (for example, LIKE)
- Value for comparison (for example EM%)
- Case-sensitive option for character data comparison

For example, to display only tables with names that start with EM, specify NAME, LIKE, and EM% (with the percent sign as a wildcard character).

Override Schema Filter (object type node filters): If this option is checked, any filter criterion specified at the connection level is ignored, and only the object type node filter criteria are applied.

Include Synonyms (object type node filters): If this option is checked, synonyms for objects of this object type are included.

To add another filter criterion, click the Add (+) icon; to delete a criterion, select it and click the Delete (X) icon; to move a criterion up or down in the list, select it and use the arrow icons.

To apply the filter criteria to the Connections navigator display, click **OK**.

To remove the effects of applying a filter, right-click the object type node in the Connections navigator display and select **Clear Filter**.

5.71 Insert Macro

This dialog box is displayed when you click **Insert** when specifying external program options (see [Section 5.66, "Create/Edit External Tool"](#)). It enables you to insert a sample text string into the relevant field for the external program option; you can then edit that string to suit your needs. (This is somewhat analogous to using snippets to insert text strings into the SQL Worksheet.)

Select the desired type of macro, read its description to ensure that it is what you want, and click **OK**. For some macros, a sample expansion is included.

5.72 Externally Modified Files

This dialog box is displayed when an external application has modified a file that you have open in SQL Developer. You are asked if you want to reload the externally modified file.

If you click **Yes**, the externally modified file overwrites any changes that you might have made in SQL Developer. If you click **No**, the externally modified file will be overwritten by your version when you save the file in SQL Developer.

5.73 Filter Object Types

This dialog box filters (restricts) the types of objects to be displayed for the schema associated with the selected user.

Available Object Types: Lists the types of objects that are available to be added to the display.

Displayed Object Types: Lists the types of objects that are included in the display.

To add a type of object to the display, select it in Available Object Types and click the Add (>) icon; to remove a type of object from the display, select it in Displayed Object Types and click the Remove (<) icon. To move all types of objects from available to displayed (or the reverse), use the Add All (>>) or Remove All (<<) icon.

5.74 Filter Schemas

This dialog box enables you to restrict the schemas that are displayed under Other Users in the Connections navigator.

Available Schemas: Lists the schemas that are not currently displayed under Other Users in the Connections navigator, but that are available to be added to the list of displayed users.

Displayed Schemas: Lists the schemas that are to be included in the display under Other Users in the Connections navigator.

To add a schema to the display, select it in Available Schemas and click the Add (>) icon; to remove a schema from the display, select it in Displayed Schemas and click the Remove (<) icon. To move all schemas from available to displayed (or the reverse), use the Add All (>>) or Remove All (<<) icon.

Only display schemas with visible objects: Limits the display to available schemas that have any database objects that are visible to the database user associated with the current connection.

5.75 Filter Error

This dialog box is displayed if you did not specify any data for an export operation. Be sure to specify Filter Data options that select some data for the export operation.

5.76 Find/Replace Text

This dialog box specifies a text string to find, optionally a replacement text string, and search options.

Text to Search For: Text string to search for.

Replace With: If you check this option, enter a text string to replace the text string that is being searched for.

Options: Options to control the search behavior: **Match Case** makes the search case-sensitive; **Search from Beginning** starts the search at the beginning instead of at the text cursor; **Highlight All Occurrences** highlights all occurrences of the search string instead of just the first one; **Wrap Around** searches across line breaks; **Whole Word Only** find the search string only if it is a complete word and not just part of a word; **Regular Expressions** means that the search string is a regular expression; **Selected Text Only** means to search only in the text block that you have selected.

Direction: **Forward** starts the search from the cursor in the direction of normal text flow; **Backward** starts the search from the cursor in the opposite direction of normal text flow.

5.77 Find Result

This box is displayed if you specify text to search for in the [Find/Replace Text](#) dialog box that is not in the SQL Worksheet.

If you think that the text is in the worksheet, retry your query, and check the spelling of the text to search for.

5.78 Format Properties

This box is displayed if you right-click and select **Advanced Format** in the editor for a subprogram, package, view, or trigger. You can specify a set of formatting rules different from those if you had selected **Format** (that is, different from the [Database: SQL Formatter](#) user preferences).

Output Destination: **Editor** applies the selected Output Type formatting in the current editing pane; **Clipboard** applies the formatting on the clipboard, so that you can paste it into a pane or window of your choice.

Output Type: A set of formatting rules associated with an output type: SQL or another type from the list.

5.79 Generate Oracle DDL

This dialog box is displayed when you click **Migration**, then **Script Generation**, then **Generate Oracle DDL**. It specifies the converted model for which to generate Oracle DDL (data definition language) statements. The operation produces a SQL*Plus script file that you use for offline generation: that is, you can run the script to create the appropriate objects in the Oracle database.

When the operation is in progress, a box displays object types and the number of objects of each type for which DDL statements are being generated.

After the operation finishes, a box displays the directory in which two files have been created: a .sql file containing DDL statements (such as CREATE TABLE and CREATE OR REPLACE VIEW) that create the migrated schema objects in the Oracle database, and a .ctl file containing the SQL*Plus @ statement to invoke the .sql file.

Converted Models: Converted model containing objects for which to generate Oracle DDL statements.

5.80 Generate Offline Data Move Files

This dialog box is displayed when you click **Migration**, then **Script Generation**, then **Generate Data Move Scripts**. It specifies the converted model and the destination directory if you are performing offline data migration, which is explained in [Section 2.9.1](#).

Converted Model: Converted model containing tables whose data is to be moved to the corresponding Oracle database tables.

Directory: Path in which to create files containing the data and the SQL*Loader specifications.

5.81 Generate Patch

You can generate a patch containing changes that have been made to files. The patch can be applied to another set of checked out CVS files so that your changes are

incorporated into them. A patch must be applied to the same revision/tag from which it was generated.

Source Files or Project: Name and location of the project or set of files will be the subject of the patch.

Patch Target: Where the generated patch will be sent: **System Clipboard** (from which you can paste the patch), **File** (accept the shown name and location, or specify different ones), or **Open Patch File Editor** (where you can edit and then save it).

Output Format: One of the standard diff formats: **Unified**, **Context**, or **Standard**.

5.82 Go to Bookmark

Use this box to specify the bookmark to go to in the selected function or procedure. After you enter the bookmark and click **Go**, the line associated with that bookmark is highlighted.

5.83 Go to Line Number

Use this box to specify the line number to go to in the selected function or procedure. After you enter the line number and click the Go icon, that line is highlighted.

5.84 Go to Line Number: Error

This error box tells you that you entered an invalid line number in the Go to Line Number box, probably because you entered a line number greater than that of the last line in the function or procedure.

5.85 Import to CVS

This interface is displayed when you click Versioning, then CVS, then Import Module. It enables you to import local files into the repository as a CVS module.

Module

Select the connection name, enter a name for the module, and optionally enter a descriptive comment about the import operation.

Tags

Select the connection name, enter a name for the module, and optionally enter a descriptive comment about the import operation.

Sources

Source Folder: Location from which files will be copied for the import operation.

Filters

You can configure filters to be used for excluding folders and files from the import operation. Use the arrow keys to move selected filters or all filters between Available Filters and Selected Filters.

To create a filter and add it to the Selected Filters list, click **New** to display the [Create Filter](#) dialog box.

Options

You can specify options to be used during the import operation.

Use File Modification Time: If this option is checked, the file's modification time is used as the time of import. If this option is not checked, the time when the import operation is performed is used as the time of import.

Perform Module Checkout: If this option is checked, the modules are checked out after they are imported.

Summary

You can review the information that will be used to perform the import operation.

To go back and make any changes, click **Back** as needed.

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

5.86 Load Keyboard Scheme

This dialog box is displayed when you select Load Keyboard Scheme from More Actions when specifying shortcut key preferences for SQL Developer. You can load a set of predefined key mappings for certain systems and external editing applications. If you load any preset key mappings that conflict with changes that you have made, your changes are overwritten.

You can specify Default to restore the shortcut key mappings to the SQL Developer defaults for your system.

5.87 Log In to CVS

Use this dialog box to log in to the specified CVS repository. You must know the password for the specified user.

Connect Automatically on Startup: If this option is checked, a login operation is performed when you start SQL Developer.

5.88 Manage Columns

Use this dialog box to reorder, hide, or show columns in the display when you are editing a table or data in a table. To move a column higher or lower in the display order, click the column and then click the appropriate icon on the left. To hide a column or to show a hidden column, click the column if necessary and click the appropriate icon between the Shown Columns and Hidden Columns lists.

When you are finished, click **OK**. (Your action affects only the current display; it does not change the table definition or any data in the table.)

5.89 Modify Value

This dialog box is displayed when you right-click a variable in the Data or Smart Data pane during debugging and select Modify Value. You can modify the value for the selected data item (primitive value, string, or reference pointer) during debugging. Note: You cannot undo the action after you click OK, so be careful when making any changes.

Current Value: The value of the data item.

New Value: The new value for the data item (enter or select from a drop-down list).

- For a primitive value, you can enter a new value.
- For a reference pointer, you can enter the memory address of an existing object or array. To set a reference pointer to null, enter 0 as a memory address.
- For a string, you can enter either a new string value or the memory address of an existing string.

Interpret New Value as Object Address: If this option is checked, the New Value entry is interpreted as a memory address pointer to an object or array in the heap of the program you are debugging. For a string, this box must be checked check if the value you enter in the New Value field is the memory address of an existing string

5.90 Data Move Details

This dialog box is displayed when you click Migration, then Migrate Data. It specifies the source and target information for online data migration, which is explained in [Section 2.9](#).

Source Connection: Database connection from which data is to be migrated.

Target Connection: Database connection to which data is to be migrated.

Converted Model: Converted model containing tables whose data is to be moved to the corresponding Oracle database tables.

Use qualified names from converted model for insert: If this option is checked, object names are qualified by the schema name.

5.91 New Procedure (Refactoring)

This dialog box is displayed if you are editing a procedure, select one or more PL/SQL statements, right-click, and select Refactoring, then Extract Procedure. The selected statements are encapsulated into the procedure to be created.

Defined Locally: For a standalone procedure, defines the newly refactored code in the definition section of the original procedure.

Stored: For a standalone procedure, defines the newly refactored code in a new standalone procedure.

Name: Name of the procedure to encapsulate the selected statements. For a packaged procedure, the newly extracted procedure text is placed immediately after the current procedure.

5.92 No Object Found

This dialog box is displayed if no objects could be found to satisfy the requested operation, such as trying to perform a "Describe" operation when the currently selected object is not valid for a SQL*Plus DESCRIBE statement.

5.93 No Object Selected

This dialog box is displayed if no object was selected for the requested operation, such as trying to perform a "Describe" operation when no object is selected in the SQL Worksheet.

5.94 Object Preferences

This dialog box lets you specify preferences for the display of data output during debugging. For a specified data type and its subclasses, you can control what to display in value columns, what to show when expanding the object (expressions, fields, or both), and (for fields) which fields to show and which to hide.

5.95 Open File

This is a standard box for selecting a file to open: use **Location** to navigate to (double-clicking) the folder with the file to open, then click the file to select it.

5.96 Oracle-Only Report

This dialog box is displayed if you select a non-Oracle (third-party) database connection for a report that applies only to Oracle database connections. Be sure to select an Oracle connection.

5.97 Oracle Proxy Authentication

This dialog box is displayed if you enable the Proxy Connection option in the [Create/Edit/Select Database Connection](#) dialog box. For an explanation of proxy authentication, see [Section 1.4.5, "Connections with Proxy Authentication"](#).

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.

5.98 Paste

This dialog box is displayed if you click **Edit**, then **Extended Paste**. It shows a list of clipboard items, so that you can select the content to be pasted. Click **OK** to paste the selected content into the current location.

Clipboard Items: Clipboard items with content from copy operations. Usually displays the first line of the content.

Item Content: The content of the selected clipboard item.

5.99 Privilege Warning for Migration

This dialog box is displayed if you click **Verify** in the **Quick Migrate** box and the database user for the connection does not have all privileges necessary for a multischema migration. For multischema migrations, this user must be granted the **RESOURCE** role with the **ADMIN** option; and this user must also be granted the **CREATE ROLE**, **CREATE USER**, and **ALTER ANY TRIGGER** privileges, all with the **ADMIN** option.

If you are performing a single-schema migration, you can ignore this warning.

5.100 Query Builder

The Query Builder box is displayed when you right-click in the SQL Worksheet and select **Query Builder**. You can use this box to create a SELECT statement by dragging and dropping table and view names and by graphically specifying columns and other elements of the query. When you finish building the query, the resulting SELECT statement is inserted into the SQL Worksheet.

The Query Builder capabilities are grouped under the following tabs.

Select Columns

Use the Select Columns tab to select tables and views, then columns within them, to be used in the query. Use the connections tree on the left to find the desired tables and views under the appropriate schema or schemas, and double-click each desired table and view.

Within each selected table or view, click to select the desired columns (all or specific ones) to include in the query.

Create Where Clause

Use the Create Where Clause tab to select, for each column in the WHERE clause, the column name, operator, and value. For example, you might want to select only rows where AUTHOR_LAST_NAME contains Melville or where RATING > 5.

Show SQL

Use the Show SQL tab to see a read-only display of the query reflecting what you have specified so far.

View Results

Use the View tab to test the query in its current form. Click the Execute Statement icon to execute the query.

Refresh: Specifies the refresh interval: the number of seconds between each time the query is automatically re-executed and the results display is updated. A value of zero (0) means that the query is not automatically re-executed after the initial execution.

5.101 Recent Files

This dialog box displays files recently opened in SQL Developer.

Files: A list of files opened in SQL Developer, with the most recent file first. The Show All option determines whether the list includes only files opened implicitly or files opened implicitly or explicitly.

Show All: If this option is checked, the list includes both explicitly and implicitly opened files; if this option is not checked, the list includes only implicitly opened files. Explicitly opened files are those that you opened directly; implicitly opened files are those that SQL Developer opened to support your work (for example, while you were debugging).

5.102 Create Repository

This dialog box is displayed if you click Migration, then Repository Management, then Create Repository.

Create Repository: Name of the database connection to use to create a migration repository. The objects associated with the migration repository are created in the schema of the user associated with the selected connection.

5.103 Delete or Truncate Repository

The Delete Repository dialog box is displayed if you click Migration, then Repository Management, then Delete Repository; the Truncate Repository dialog box is displayed if you click Migration, then Repository Management, then Truncate Repository.

Deleting a repository removes all schema objects that are used for the migration repository. Truncating a repository deletes all data from schema objects that are used for the migration repository, but does not delete the schema objects themselves, effectively leaving you with an empty repository.

Repository: Name of the database connection in which to delete or truncate the migration repository.

5.104 Capture Microsoft Access Exporter XML

This dialog box is displayed if you click Migration, then Capture Exporter XML.

File Path: File path to the .xml file that was produced when you ran the appropriate version of the exporter tool for Microsoft Access (when you clicked Migrations, then Microsoft Access Exporter, then the appropriate version for your version of Microsoft Access).

5.105 Rename Local Variable

This dialog box is displayed if you right-click a variable name in the display of the source code for a function or procedure, and select Refactoring and then Rename Local Variable. Specify the desired new name for the variable.

5.106 Rename Procedure

This dialog box is displayed if you try to rename a procedure. Specify a unique new name for the procedure.

5.107 Select Current Repository

This dialog box is displayed if you click Migration, then Repository Management, then Select Current Repository. You can use this dialog box to reconnect to a migration repository after you have disconnected (using Migration, then Repository Management, then Disconnect Migration Repository). In addition, if you have multiple migration repositories, and you can use this dialog box to switch from one to another.

Select Current Repository: Name of the database connection with the migration repository to be used for all operations relating to migrating third-party databases to Oracle.

5.108 Cannot Capture Table

This dialog box is displayed if you try to capture a third-party database before establishing and connecting to a current migration repository.

If no migration repository exists, create one by clicking **Migration**, then **Repository Management**, then **Create Repository**.

To make an existing migration repository the current one, right-click its connection in the Connections navigator and select **Associate Migration Repository**.

To open a connection to the migration repository, expand the node for its connection in the Connections navigator.

5.109 Reset Expired Password (Enter New Password)

This dialog box is displayed if you attempt to create a new database connection or open an existing connection, and if the password associated with the used for the connection has expired. It is also displayed only if an OCI (thick) driver is available; if an OCI driver is not available, an error message is displayed instead of this dialog box.

To reset the password, enter the current password for the specified user, enter the new password, confirm the password (type the same new password), and click **OK**.

5.110 Revision Lister

This dialog box is displayed if you click List Revisions in the [Subversion: Branch/Tag](#) dialog box. It contains a list of revisions in the repository.

Select the desired revision to use, and click **OK**.

5.111 Run/Debug/Profile PL/SQL

Use this box to specify parameter values for running, debugging, or profiling a PL/SQL function or procedure. (If you specify a package, select a function or procedure in the package.) A profile operation runs the function or procedure and collects execution statistics; it also requires auxiliary structures in the user schema. For information, see [Section 1.6.4, "Using the PL/SQL Hierarchical Profiler"](#).

Comment (Profile only): Descriptive comment to be included in the execution profile.

Target: Name of the function or procedure to run or to run in debug mode. (You have a choice only if you specified a package that has more than one subprogram.)

Parameters: List of each parameter for the specified target. The mode of each parameter can be IN (the value is passed in), OUT (the value is returned, or IN/OUT (the value is passed in, and the result of the function or procedure's action is stored in the parameter).

PL/SQL Block: A block of PL/SQL code created by SQL Developer. You should change the formal IN and IN/OUT parameter specifications in this block to actual values that you want to use for running or debugging the function or procedure.

For example, to specify 10 as the value for an input parameter named `in_rating`, change `IN_RATING => IN_RATING` to `IN_RATING => 10`.

When you click **OK**, SQL Developer runs the function or procedure.

If you are debugging a function or procedure, the debugging toolbar and one or more windows for debug-related information are displayed, as explained in [Section 1.6, "Running and Debugging Functions and Procedures"](#).

5.112 Create/Edit Breakpoint

Use this box to create or edit a breakpoint to use when debugging a PL/SQL function or procedure.

Definition tab

Specify the definition of the breakpoint.

Breakpoint Type: Type of breakpoint, indicating when the breakpoint will occur. Options include breaking when one of the following occurs: a specific line of code (Source); exception class or other class; method, file, or watch.

Breakpoint Details: Options depend on the breakpoint type.

Breakpoint Group Name: Breakpoint group in which to include this breakpoint. Breakpoint groups can be edited, enabled, and disabled.

Conditions tab

Specify any conditions that apply to the breakpoint.

Condition: A SQL condition (WHERE clause without the WHERE keyword) restricting when the breakpoint occurs. For example, to specify that the condition should occur only when `status_code` is greater than 10, specify:

```
status_code > 10
```

Thread Options: You can specify whether the breakpoint occurs for all threads, or only when the breakpoint is hit by threads that either do or do not have a specified name.

Pass Count: The number of times the debugger should allow execution to pass over the breakpoint before the breakpoint occurs.

Actions tab

Specify the actions to be taken when the breakpoint occurs. The options you specify override any default values on the [Debugger: Breakpoints: Default Actions](#) pane for [SQL Developer Preferences](#).

Halt Execution: Pauses execution when the breakpoint occurs.

Beep: Beeps when the breakpoint occurs.

Log Breakpoint Occurrence: Sends a message to the log window when the breakpoint occurs. You can also specify the following to be included in each display: a tag, and a condition to be evaluated.

Enable/Disable a Group of Breakpoints: Enables or disables the specified breakpoint group when this breakpoint occurs.

5.113 Save/Save As, or Select File

This is a standard box for saving information to a file or for selecting a file: use **Location** to navigate to (double-clicking) the folder in which to save or open the file, then specify the file name (including any extension) and, if necessary, the file type.

5.114 Save Files

This box asks if you want to save the specified files before another action occurs (for example, saving procedures you had been editing before disconnecting).

5.115 Unable to Save Files

This box informs you that SQL Developer is unable to save the specified file or files. To cancel the attempt to save the files and to return to edit the relevant object, click **Cancel**.

5.116 Save Style Settings

This dialog box is displayed when you click **Save As** in the [Code Editor: PL/SQL Syntax Colors](#) pane when setting [SQL Developer Preferences](#). You can save the specified color settings as a named color scheme, which adds it to the drop-down list for **Scheme** in that pane.

5.117 Schema Differences Source or Destination Error

This error box is displayed if you click **Apply** before specifying the source or the destination, or both, for a schema differences operation.

Click **OK** to close the error box, then follow the instructions for performing the schema differences operation, as explained in [Section 5.50, "Database Schema Differences"](#).

5.118 Script Execution Failed

This error box is displayed if the script generated by the Quick Migrate procedure fails before it completes its execution. The Build pane displays the error that caused the failure.

To close the error box and open the script in a SQL Worksheet window, where you can edit the text and run the corrected script, click **Yes**; or to close the error box without opening the script in a SQL Worksheet window, click **No**.

5.119 Script Generation Complete

This information box is displayed after you generate the controlling script and related files for performing an offline capture of a third-party database, as explained in [Section 2.6.2, "Offline Capture"](#).

Click **OK** to close the error box. Later, run the controlling script to generate output containing the converted model.

5.120 Set Data Mapping

This dialog box is displayed if you right-click a captured model and select **Set Data Mapping**. You can use this dialog box to specify source data type mappings when migrating the specified third-party database to Oracle. If you are editing an existing mapping, you can change only the Oracle data type, precision, and scale information.

Show only data types used in the source model: If you check this option, only data types used in the selected captured model are shown. If you do not check this option, all valid data types for the source (third-party) database are shown.

Source Data Type: Data type name in the third-party database.

Oracle Data Type: Data type name in Oracle Database.

Type: *System* for a system-defined data type, or *User* for a user-defined data type.

Add New Rule: Displays the [Add/Edit Rule](#) dialog box, for specifying a mapping for another data type.

Edit Rule: Displays the [Add/Edit Rule](#) dialog box, for editing the selected mapping.

Remove Rule: Deletes the selected mapping.

5.121 Add/Edit Rule

This dialog box is displayed if you click Add New Rule or Edit Rule in the [Set Data Mapping](#) dialog box, which is used for specifying source data type mappings when migrating a specified third-party database to Oracle.

Source Data Type: Data type name in the third-party database.

Oracle Data Type: Data type name in Oracle Database.

Precision and Scale: Precision and scale values to be used for the source data type and Oracle data type during the conversion.

5.122 Set Encoding

This dialog box is displayed if you right-click a CVS connection and select Set Encoding. Specify a character set for the connection. The character set that you choose is applied to the encoding of files under CVS control through that connection.

Platform Default (Newline Conversions): Uses the character set specified for the platform/operating system. Newline conversions for files crossing different platforms are handled automatically.

IDE Global Setting: Uses the default character set for the integrated development environment (IDE).

Other: Uses the selected character set.

5.123 Set Pause Continue

This dialog box is displayed if you enter the SQL*Plus statement SET PAUSE ON in the SQL Worksheet and then run the worksheet contents as a script. After the SET PAUSE ON statement is processed, execution pauses (and this dialog box is displayed) after each statement until the SET PAUSE OFF statement is processed.

To have execution continue at the next statement, click **OK**.

5.124 Sign In (checking for updates)

This dialog box is displayed if any of the updates that you selected during the check for updates process are on a remote site that requires you to log in. Currently, all updates are on the Oracle Technology Network (OTN), so you must enter your OTN user name and password.

User Name: Your user name at the remote site.

Password: Your password at the remote site.

Sign Up: If you do not have an account at the remote site, click this link.

Find Password: If you have an account at the remote site but cannot remember your password, click this link.

5.125 Single Record View

The main use for this box, which is displayed by right-clicking the display grid for an object and selecting Single Record View, is to edit data for a table or view, one record at a time. After you change data in any cells in a row, you can apply the changes by clicking Apply or by navigating to another record. (For non-Data grids, the cells are read-only.)

Navigation icons: First (<<) moves to the first record, Previous (<) moves to the previous record, Next (>) moves to the next record, and Last (>) moves to the last record.

Apply: Applies changes made to the current data record.

Cancel: Cancels changes made to the current data record, and closes the box.

5.126 Save Snippet (User-Defined)

Use this box to create a user-defined snippet. For information about how to create user-defined snippets, including options for snippet categories, see [Section 1.8.1, "User-Defined Snippets"](#).

Category: Existing or new category in which to place the snippet. To create a new (user-defined) category, type the category name instead of selecting a category name from the list.

Name: Name of the snippet, as it will be displayed when users see the list of available snippets in the specified category. If an existing Oracle-supplied snippet has the same name in the same category, the user-defined snippet definition replaces the Oracle-supplied definition.

ToolTip: Optional tooltip text to be displayed when the mouse pointer stays briefly over the snippet name in the display of snippets in the specified category.

Snippet: Text that will be inserted for this snippet.

5.127 Edit Snippets (User-Defined)

This box displays any existing user-defined snippets, and enables you to add, edit, or delete user-defined snippets.

To edit an existing user-defined snippet, select its row and click the **Edit User Snippet** icon, which displays the [Save Snippet \(User-Defined\)](#) dialog box.

To create a new user-defined snippet, click the **Add User Snippet** icon, which displays the [Save Snippet \(User-Defined\)](#) dialog box.

To delete a user-defined snippet, select its row and click the **Delete User Snippet** icon.

5.128 Subversion: Add Property

Use this dialog box to add a versioning property for the currently selected file or folder.

5.129 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.

5.130 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.

5.131 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.

5.132 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 Versioning browser hierarchy) through Empty (only this item and no children).

5.133 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.

5.134 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.

5.135 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.

5.136 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.

5.137 Subversion: Create Subversion Repository

This information applies to creating a Subversion repository. A connection to the repository will be created automatically. For information about SQL Developer support for versioning and Subversion, see [Section 1.10](#).

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.

5.138 Subversion: Create/Edit Subversion Connection

This information applies to creating or editing a Subversion connection. For information about SQL Developer support for versioning and Subversion, see [Section 1.10](#).

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).

5.139 Subversion: Edit Configuration File

This dialog box is displayed if you click Edit "server" in the [Versioning: Subversion: General](#) 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**.

5.140 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.

5.141 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.

5.142 Subversion: History

This dialog box displays version history information about Subversion files.

5.143 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.

5.144 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.

5.145 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 wish 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 prevent 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.

5.146 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.

5.147 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 wish 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.

5.148 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).

5.149 Subversion: Properties

This dialog box is displayed in you right-click a node under a connection in the Versioning navigator and select Properties. It displays properties and property values for the selected object.

5.150 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.

5.151 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.

5.152 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.

5.153 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**.

5.154 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.

5.155 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.

5.156 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.

5.157 Subversion: Versioning Properties

This dialog box displays general and versioning information about the currently selected file or folder.

5.158 Third-Party Database Objects

You have requested help about a type of object in the context of a third-party (non-Oracle) database connection. See the documentation for the third-party database for information about database objects as they apply to that product.

5.159 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 is not available.

5.160 Unable to Open File

This box informs you that SQL Developer is unable to perform the export operation to the location and file that you specified. The cause might be that you do not have permission to write to that location.

5.161 Unit Testing: Action Required

This dialog box is displayed if you are creating a repository but the database user for the repository does not have the necessary privileges.

SQL Developer will issue one or more prompts. For each, click **Yes**, enter the password for the SYS account, and allow the commands to execute.

5.162 Unit Testing: Add Category

Use this dialog box to create a lookup category for unit testing. (Using lookups is explained in [Section 3.6, "Using Lookups to Simplify Unit Test Creation"](#).)

5.163 Unit Testing: Add Data Type

Use this dialog box to add a data type to a lookup category for unit testing. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

5.164 Unit Testing: Add Item to Library

Use this dialog box to add a startup, teardown, or validation specification to the unit testing library, depending on which node you have selected in the Unit Test navigator. After you specify a name and click OK, specify a connection, and then complete the definition. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

5.165 Unit Testing: Add Test Implementation

Use this dialog box to add an implementation for a unit test. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

When you create a unit test, a default implementation is created; however, you can specify one or more additional implementations.

5.166 Unit Testing: Add Test Suite

Use this dialog box to create a test suite for unit testing. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

5.167 Unit Testing: Add Tests to Suite

Use this dialog box to add one or more unit tests to a test suite. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

List of tests: Select one or more tests to be added to the suite.

Run Test Startups: If this option is checked, the startup action defined for each specified test is run when the suite is run.

Run Test Teardowns: If this option is checked, the teardown action defined for each specified test is run when the suite is run.

Any startup and teardown actions for a test are specified when you create the unit test (see [Unit Testing: Create Unit Test](#)).

5.168 Unit Testing: Copy or Rename Unit Test

Use this dialog box to copy or rename a unit test. (The SQL Developer unit testing feature is explained in [Chapter 3](#).)

If you specified Copy, a copy of the selected unit test is created and is given the name you specify.

If you specified Rename, the name of the selected unit test is changed to the name you specify.

5.169 Unit Testing: Create Unit Test

Use this wizard to create a unit test. (The SQL Developer unit testing feature is explained in [Chapter 3](#).) To go from one step to the next, click Next; to go back to the previous step, click Back.

Select Operation

Select a database connection on the right, then use the hierarchy tree to select an object to be tested. For example, to test a procedure, expand the Procedures node and select the desired procedure.

Specify Test Name

Test Name: Name for the unit test. Can be the same as the name of the database object (for example, procedure or function) to be tested.

Create with single dummy representation: Creates a single (sometimes called "one-off") test case for which you must specify the input parameter values when you run the test.

Select/create implementation using lookup values: Generates multiple test cases using sets of input parameter values that you will specify.

Specify Startup

Specify the action to perform at the start of the test, before any of the actual test operations are performed. Reasons for using a startup action might include the following:

- You want to create copies of the entire table or specific rows that will be modified by the test, so that you can restore the original values later during teardown.
- You want to perform some special operations before the test is run.

None: Perform no startup action.

Row Copy: Copy rows from a specified table. You will be asked to specify the table and q query for selecting the rows.

Table Copy: Copy a specified table. You will be asked to specify the table. For example, you might want to copy the EMPLOYEES table to a table named EMPLOYEES_ORIGINAL.

User PL/SQL Code: Run a script. You will be asked to specify the script file.

Specify Parameters

If you are creating a single test case, specify the parameter values in the Input column for each input parameter. For example, for the example `award_bonus` procedure, you might test the case where employee ID 1001 sold 5000 dollars work of goods or services (EMP_ID Input: 1001, SALES_AMT Input: 5000).

If you are creating multiple test cases, specify the information for each parameter for each test case. For example, to create a test with 5 test cases for the example `award_bonus` procedure, you need to enter 10 rows on this page (2 input parameters times 5 test cases).

Specify Validations

Specify one or more validation actions to perform after the test case is run, to check whether the desired result was returned. For example, if a test case for a procedure was supposed to increase the salary for employee number 1001 to \$2000, specify a Query returning row(s) validation of `SELECT salary FROM employees WHERE employee_id = 1001 AND salary = 2000`.

For all validation action options except None, you will be prompted to specify the required information.

To add a validation action for a test case, click the Add (+) icon; to delete a validation action for a test case, select it and click the Remove (X) icon.

None: Perform no validation action.

Boolean function: Validation succeeds if Boolean TRUE is returned by the specified function.

Query returning no row(s): Validation succeeds if the specified query returns no rows.

Query returning row(s): Validation succeeds if the specified query returns one or more rows that satisfy the specified query.

Result Set Compare: Validation succeeds if the results returned by the specified query match the values that you specify.

Table Compare: Validation succeeds if the target table (the table modified by the test case) is identical to the specified table.

User PL/SQL Code: Validation succeeds if the specified script runs successfully.

Specify Teardown

Specify the action to perform after the validation action (or the test case in no validation action was specified) is finished. Reasons for using a teardown action might include the following:

- You want to restore a table to its original values, effectively "rolling back" any changes that were made by the test case.
- You want to perform some special operations after the test is run.

None: Perform no teardown action.

Table Drop: Drop (delete) the specified table.

Table Restore: Replace all rows in the specified target table (the table modified by the test case) with the rows in the specified source table. For example, you might want to replace the contents of EMPLOYEES (target) table with a source table named EMPLOYEES_ORIGINAL.

User PL/SQL Code: Run a script. You will be asked to specify the script file.

Summary

Displays the selected options for the application to be deployed or imported. To make any changes, click Back. To perform the operation, click Finish.

5.170 Unit Testing: Manage Users

Use this dialog box to add or remove database users as unit test users (Users tab) or administrators (Administrators tab). (Managing unit test users and administrators is explained in [Section 3.3.1](#).)

On each tab, use the icons to move selected database users (> or <) or all listed database users (>> or <<) into or out of the column on the right, which identifies the unit testing users or administrators.

With Admin: If this option is checked, the users in the column on the right can grant the specified unit test role (User or Administrator) to other database users.

5.171 Unit Testing: Result of Operation

This informational box indicates either that the operation was successful or that an error occurred. If an error occurred, a brief explanation is included.

5.172 Unsupported Database Version

This box is displayed if you try to create a connection to a database release that is not supported by SQL Developer, such as Oracle Database release 8.1. For information about database releases supported by SQL Developer, see *Oracle SQL Developer Installation Guide*.

5.173 Windows

This dialog box is displayed if you right-click the tab for a window in the display area of the SQL Developer main window.

Windows: A list of the windows in the display area.

Activate: Makes active (switches focus to) the selected window.

Close: Closes the selected window.

A

- accelerator (shortcut) keys, 1-62
 - for menus, 1-3
- Access (Microsoft) connections, 5-7
- Active Session History (ASH) reports, 1-41
- Advanced Security for JDBC connection, 1-22
- analyzing tables, 1-17
- application
 - deploying (Application Express), 5-41
 - importing (Application Express), 5-41
- Application Express
 - applications, 1-10
 - deploying an application, 5-41
 - importing an application, 5-41
 - Publish to Apex option, 1-24
 - refactor in bulk command, 1-10
 - starting and stopping EPG (embedded PL/SQL gateway), 1-11
- applications
 - Application Express, 1-10
- Apply Filter
 - to display of objects, 1-4, 5-49
- ASH (Active Session History) reports, 1-41
- associations
 - file types, 5-38
- authentication
 - Kerberos, 5-6
 - Database Advanced Parameters, 1-52
 - OS (operating system), 5-6
- autocommit
 - preference for database connections, 1-56
- autocomplete preferences, 1-50
- Automated Workload Repository (AWR)
 - reports, 1-41
- autotrace
 - Autotrace pane, 1-33
 - preferences for, 1-53
- AWR (Automated Workload Repository)
 - reports, 1-41

B

- Berkeley DB
 - data storage for Subversion repository, 5-65
- bind variables

- for reports, 1-40
- saving to disk on exit, 1-57

- blogs
 - SQL Developer, 1-71
- breakpoints
 - creating and editing, 5-59

C

- cache groups, 1-11
- case insensitivity for queries
 - setting, 2-24
- charts
 - user-defined report example, 1-46
- check constraints, 5-22
- Check for Updates feature, 5-2
- CLOBtoBLOB_sqldeveloper stored procedure, 1-61
- close all SQL Worksheets automatically on disconnect
 - preference for database connections, 1-57
- coalescing an index, 1-12
- code coverage
 - gathering and viewing statistics, 3-5
- code fragments, 1-36
- code insight, 1-50
- column sequences, 5-23
- columns
 - decrypting, 1-17
 - encrypting, 1-17
 - normalizing column data, 1-17
- Compact option for shrinking a table, 1-17
- compiling
 - function, 1-12, 1-14
 - for debug, 1-12, 1-14
 - view, 1-18
- completion
 - SQL Developer preferences, 1-50
- completion insight, 1-50
- configuring, 2-13
 - third-party databases, 2-12
- configuring file type associations, 5-38
- connections
 - creating, editing, or selecting, 5-5
 - explanation, 1-19
 - IBM DB2, 5-8
 - JDBC, 5-8
 - Microsoft Access, 5-7

- Microsoft SQL Server, 5-9
- MySQL, 5-8
- Oracle Database, 5-5
- Oracle TimesTen, 5-7
- separate unshared worksheet for, 1-21, 1-29
- Sybase Adaptive Server, 5-9
- Teradata, 5-9
- using folders to group, 1-21
- constraints
 - check, 5-22
 - disabled, 1-44
 - unique, 5-21
- converted model
 - correcting errors in, 2-19
 - creating and customizing, 2-18
- Create Local Connections command, 1-20
- Create Unit Test wizard, 5-72
- customizing SQL Developer
 - setting preferences, 1-48
- CVS (Concurrent Versions System)
 - preferences for, 1-63
 - SQL Developer support for, 1-37

D

- data
 - entering and modifying, 1-23
- data move
 - generating user, 1-61
 - number of parallel streams, 1-60
 - options, 1-59
- data types
 - creating, 5-30
- database connections, 5-9
 - creating, editing, or selecting, 5-5
 - explanation, 1-19
 - IBM DB2, 5-8
 - JDBC, 5-8
 - Microsoft Access, 5-7
 - Microsoft SQL Server, 5-9
 - MySQL, 5-8
 - Oracle Database, 5-5
 - Oracle TimesTen, 5-7
 - Sybase Adaptive Server, 5-9
 - Teradata, 5-9
- database links, 1-11
 - creating or editing, 5-12
- database objects, 1-10
 - exporting, 5-43
 - finding (searching for), 1-37
- database schema differences
 - differences between two database schemas, 5-40
- DB2
 - before migrating, 2-12
- DB2 (IBM) connections, 5-8
- DBMS_OUTPUT
 - user-defined report example, 1-47
- DBMS_OUTPUT pane, 1-33
- debugging PL/SQL function or procedure, 1-25
 - dialog box, 5-58
 - remote debugging, 1-27

- decryption
 - data in a table column, 1-17
- deployment
 - after database migration, 2-29
- destination schema
 - database schema differences, 5-40
- dialog boxes and tabs, 5-1
- directories (directory objects), 1-11
- disabled constraints, 1-44
- discussion forum
 - SQL Developer, 1-71
- documentation
 - generating for schema, 1-21
- downloads
 - SQL Developer, 1-71
- drag and drop effects, 1-53
- drivers
 - third-party JDBC, 1-55
- dynamic HTML
 - user-defined report example, 1-47

E

- editions, 1-11
- embedded PL/SQL gateway (EPG) for Application Express
 - starting and stopping, 1-11
- encryption
 - data in a table column, 1-17
- entering data in tables, 1-23
- EPG (embedded PL/SQL gateway) for Application Express
 - starting and stopping, 1-11
- errors
 - parse exceptions, 2-19
- Excel
 - importing data, 5-11
- execution plan, 1-33
- expired password
 - resetting, 1-21
- EXPLAIN PLAN
 - execution plan, 1-33
- exporting
 - applying filters, 5-45
 - database objects, 5-43
 - reports, 1-39
 - table data, 1-17, 5-45
 - unit test objects, 3-10
- expression watches, 1-28
- extensions, 5-47
 - SQL Developer, 1-58
 - user-defined, 1-56
- external editor
 - SQL Developer preferences, 1-58
- external tables
 - properties, 5-26
- external tools, 1-8, 5-46, 5-47

F

- F10 key
 - for File menu, 1-3
- F4 key, 1-31
- failed objects
 - generating, 1-61
- file types
 - associating with SQL Developer, 5-38
- file-oriented development
 - SQL Worksheet right-click operations, 1-30
- Files navigator, 1-4
- filter
 - applying, 1-4, 5-49
 - clearing, 1-4, 5-49
- finding database objects, 1-37
- Flashback Table support, 1-17
- folders
 - for user-defined reports, 1-45, 5-32
 - in Connections navigator, 1-21
- font
 - increasing or decreasing size for help text, 1-68
- foreign keys, 5-21
- Freeze View
 - pinning an object's display, 1-5, 1-23, 1-39
- FROM clause, 5-34
- full pathname for java.exe, 1-2
- functions
 - compiling, 1-12, 1-14
 - compiling for debug, 1-12, 1-14
 - creating, 5-16
 - debugging, 5-58
 - running, 5-58

G

- Gather Code Coverage Statistics option, 3-5
- Gather Schema Statistics menu command, 1-21
- gauges
 - in SQL Worksheet and user-defined reports, 1-35
- generate data move user option, 1-61
- Generate DB Doc menu command, 1-21
- generate failed objects option, 1-61
- graphical user interface (GUI), 1-2, 2-31
 - unit testing, 3-2
- GROUP BY clause, 5-35
- groups
 - connections in folders, 1-21

H

- HAVING clause, 5-35
- help
 - increasing or decreasing help text font size, 1-68
 - using the online help, 1-68
- hierarchical profiler (PL/SQL), 1-28
- HTML
 - dynamic (user-defined report), 1-47

I

- IBM DB2
 - before migrating, 2-12
- IBM DB2 connections, 5-8
- IDE_USER_DIR location, 1-66
- implementation
 - adding for a unit test, 5-71, 5-72
- importing
 - Microsoft Excel data, 5-11
 - reports, 1-39
 - unit test objects, 3-10
- indexes, 5-22
 - coalescing, 1-12
 - creating or editing, 5-13
 - explanation, 1-12
 - making unusable, 1-12
 - rebuilding, 1-12
- index-organized tables
 - properties, 5-26
- introduction
 - SQL Developer, 1-1

J

- Java sources, 1-13
- java.exe
 - pathname for, 1-2
- JDBC connections, 5-8
- JDBC drivers, 1-55
- jobs, 1-13
- jTDS
 - JDBC driver for Microsoft SQL Server and Sybase Adaptive Server, 1-55

K

- Kerberos authentication, 5-6
 - Database Advanced Parameters, 1-52
- keyboard shortcuts, 1-62

L

- least privilege schema migration, 1-61
- library
 - adding an item for unit testing, 5-71
 - unit testing, 3-9
- link
 - database, 5-12
- LOB parameters, 5-23
- local connections
 - creating, 1-20
- locking a table, 1-16
- lookups
 - adding for unit testing, 5-71
- lowercase characters in object names
 - quotation marks required, 5-1

M

- materialized view logs, 1-13

- creating and editing, 5-14
- materialized views, 1-13
 - options when creating or editing, 5-36
- Microsoft Access
 - capturing using exporter tool, 2-15
 - configuring before migrating, 2-14
 - creating data files, 2-20
 - turning off security, 2-14
- Microsoft Access connections, 5-7
- Microsoft Excel
 - importing data, 5-11
- Microsoft SQL Server, 2-13
 - connections, 5-9
 - creating data files, 2-20
 - third-party JDBC drivers, 1-55
- migration
 - architecture, 2-5
 - before you start (general), 2-10
 - before you start (source-specific), 2-12
 - deployment after migrating, 2-29
 - introduction and main topics, 2-1
 - overview, 2-5
 - planning for, 2-6
 - quick start, 2-1
 - required roles and privileges, 2-11
- migration plan
 - preparing, 2-6
- migration repository
 - creating database user for, 2-11
- MIME type
 - specifying external editor for BLOB data, 1-58
- MOD_PLSQL
 - OWA output, 1-34
- modifying data in tables, 1-23
- Monitor Sessions command, 1-8
- Monitor SQL command, 1-8
- moving a table to another tablespace, 1-17
- MySQL
 - configuring before migrating, 2-15
 - creating data files, 2-20
 - third-party JDBC drivers, 1-55
- MySQL connections, 5-8
 - zero date handling, 5-8

N

- normalizing
 - column data, 1-17

O

- OBEs (Oracle By Example lessons), 1-71
- objects
 - database, 1-10
 - exporting, 5-43
- OCI driver
 - database advanced parameters preference, 1-53
- offline data transfer, 2-19
- offline generation of Oracle DDL, 5-51
- online help

- using, 1-68
- open SQL Worksheet automatically
 - preference for database connections, 1-56
- operating system authentication, 5-6
- Oracle Application Express
 - applications, 1-10
- Oracle By Example (OBE) lessons, 1-71
- Oracle Database connections, 5-5
- Oracle Web Agent (OWA), 1-34
- ORDER BY clause, 5-35
- OS authentication, 5-6
- OTN page
 - SQL Developer, 1-71
- overview
 - SQL Developer, 1-1
 - OWA_OUTPUT pane, 1-34

P

- packages
 - creating, 5-15
 - debugging, 5-58
 - running, 5-58
- parallel data move streams, 1-60
- parameter insight, 1-50
- parse exception errors, 2-19
- partitioned tables
 - partition definitions, 5-26
 - partitioning options, 5-25
 - subpartitioning options (templates), 5-25
- password
 - resetting expired, 1-21
- pathname
 - for java.exe, 1-2
- pinning an object's display, 1-5, 1-23, 1-39
- PL/SQL hierarchical profiler, 1-28
- PL/SQL packages
 - creating, 5-15
- PL/SQL subprograms
 - creating, 5-16
 - debugging, 5-58
 - breakpoints, 5-59
 - running, 5-58
- plsql-dbms_output
 - user-defined report example, 1-47
- preferences
 - customizing SQL Developer, 1-48
- preferences (options), 1-63
- primary key, 5-20
- private database links, 1-11
- private synonyms, 1-16, 5-17
- privileges
 - required for migration, 2-11
- procedures
 - creating, 5-16
- profiler (PL/SQL hierarchical), 1-28
- profiling a PL/SQL function or procedure, 5-58
- projects
 - creating plan, 2-9
- proxy authentication, 1-22

- proxy connections, 1-22, 5-6
- public database links, 1-11
- public synonyms, 1-16, 5-17
- Publish to Apex (Application Express) option, 1-24

Q

- query builder, 5-56
- quotation marks
 - for name with lowercase characters, special characters, or spaces, 5-1

R

- Raptor
 - former code name for SQL Developer, 1-1
- read-only data modeling viewer, 1-67
- rebuilding an index, 1-12
- recompiling
 - view, 1-18
- Recycle bin, 1-15
- refactor in bulk, 1-10
- remote debugging, 1-27
- replication schemes, 1-15
- report navigator, 1-39
- reports, 1-39
 - bind variables for, 1-40
 - exporting, 1-39
 - importing, 1-39
 - shared, 1-39
 - unit testing, 3-10
 - user-defined
 - chart example, 1-46
 - creating and editing, 5-31
 - dynamic HTML example, 1-47
 - explanation, 1-45
 - folders for, 1-45, 5-32
 - gauge example, 1-35
 - UserReports.xml, 1-45
- repository
 - unit test, 3-4
- Reset Password menu command, 1-21
- roles
 - required for migration, 2-11
- running PL/SQL function or procedure, 1-25
 - dialog box, 5-58

S

- schema
 - XML, 1-19, 5-38
- schema differences
 - differences between two database schemas, 5-40
- schema documentation
 - generating, 1-21
- schema objects, 1-10
- schema statistics
 - gathering, 1-21
- scratch editor, 2-34
- script runner, 1-32
- scripts

- running, 1-32
- search
 - for database objects, 1-37
- security
 - Advanced Security for JDBC connection, 1-22
- SELECT clause, 5-34
- sequences, 1-16
 - creating and editing, 5-16
 - for populating table columns, 5-23
- sessions
 - monitoring, 1-8
- shared reports, 1-39
- shared repository, 3-4
- shortcut keys, 1-62
 - for menus, 1-3
 - restoring to default scheme, 1-62
- shrinking a table, 1-17
- single record view, 5-62
- size of help text
 - increasing or decreasing, 1-68
- snippets, 1-36
 - user-defined, 1-36
- source database
 - capturing, 2-16
- source schema
 - database schema differences, 5-40
- spaces in object names
 - quotation marks required, 5-1
- special characters in object names
 - quotation marks required, 5-1
- split
 - data pane for a table or view, 1-25
- SQL
 - monitoring execution, 1-8
 - SQL Developer preferences, 1-48
- SQL file
 - creating, 5-17
- SQL scripts
 - running, 1-32
- SQL Server
 - configuring, 2-13
 - creating data files, 2-20
 - third-party JDBC drivers, 1-55
- SQL Server connections, 5-9
- SQL Trace files (.trc), 1-27
- SQL Worksheet
 - closing automatically on disconnect, 1-57
 - opening automatically on database connection, 1-56
 - unshared for a connection, 1-21, 1-29
 - using, 1-28
- Start Page for SQL Developer, 1-71
- statistics
 - computing table and column, 1-17
 - estimating table and column, 1-17
 - gathering schema, 1-21
- storage options, 5-27
- stored procedures
 - generating for Migrate Blobs Offline, 1-61
- subpartitions

- templates, 5-25
- subprograms
 - creating, 5-16
 - debugging, 5-58
 - running, 5-58
- substitution variables, 1-32
- Subversion
 - preferences for, 1-65
 - SQL Developer support for, 1-37
- Sybase Adaptive Server, 2-13
 - connections, 5-9
 - creating data files, 2-20
 - third-party JDBC drivers, 1-55
- synonyms, 1-16
 - creating and editing, 5-17

T

- tables, 1-16
 - analyzing, 1-17
 - compacting, 1-17
 - computing statistics, 1-17
 - creating and editing, 5-19
 - creating quickly, 5-18
 - decrypting column data, 1-17
 - encrypting column data, 1-17
 - entering and modifying data, 1-23
 - estimating statistics, 1-17
 - exporting data, 1-17, 5-45
 - external
 - properties, 5-26
 - index-organized
 - properties, 5-26
 - locking, 1-16
 - moving to another tablespace, 1-17
 - normalizing column data, 1-17
 - partitioned
 - partition definitions, 5-26
 - partitioning options, 5-25
 - subpartitioning options (templates), 5-25
 - shrinking, 1-17
 - splitting the data pane, 1-25
 - temporary, 5-19
 - truncating, 1-16
 - validating structure, 1-17
- temporary tables, 5-19
- Teradata
 - before migrating, 2-16
 - connections, 5-9
- test case
 - definition of, 3-2
- test suite
 - definition of, 3-2
- thick (OCI) driver
 - database advanced parameters preference, 1-53
- third-party databases
 - configuring before migrating, 2-12
- third-party JDBC drivers, 1-55
- TimesTen
 - cache groups, 1-11

- database connections, 5-7
- replication schemes, 1-15
- support for in SQL Developer, 1-67
- Tip of the Day, 1-68
- TKPROF
 - opening in SQL Developer as an alternative to using, 1-27
- tnsnames.ora file, 1-19
- tools
 - external, 1-8, 5-46, 5-47
- trace files (.trc), 1-27
- transferring
 - data offline, 2-19
- translation scratch editor, 2-34
- .trc files, 1-27
- triggers, 1-18
 - creating and editing, 5-28
- truncating a table, 1-16
- T-SQL
 - not supported in worksheet for SQL Server or Sybase connection, 2-34
 - translation to PL/SQL, 2-34
- tutorial
 - creating objects for a small database, 4-1
- types
 - creating, 5-30

U

- unique constraints, 5-21
- unit test
 - definition of (as related to test cases), 3-2
- unit testing, 1-63
 - adding a category for lookups, 5-71
 - adding an implementation, 5-71, 5-72
 - adding an item to a library, 5-71
 - command-line interface, 3-11
 - copying or renaming a unit test, 5-72
 - Create Unit Test wizard, 5-72
 - editing and running a unit test, 3-5
 - example, 3-11
 - exporting objects, 3-10
 - graphical user interface (GUI), 3-2
 - importing objects, 3-10
 - introduction and main topics, 3-1
 - library, 3-9
 - managing users, 5-74
 - overview, 3-1
 - reports, 3-10
 - repository, 3-4
 - tutorial, 3-11
 - variable substitution, 3-8
- unshared worksheet for a connection, 1-21, 1-29
- unusable indexes, 1-12
- updates
 - checking for SQL Developer updates, 5-2
 - user information directory (IDE_USER_DIR), 1-66
 - user interface
 - restoring to original settings, 1-9
 - user interface (UI), 1-2, 2-31

- unit testing, 3-2
- user-defined extensions, 1-56
- user-defined reports
 - chart example, 1-46
 - creating and editing, 5-31
 - dynamic HTML example, 1-47
 - explanation, 1-45
 - folders for, 1-45, 5-32
 - gauge example, 1-35
 - UserReports.xml, 1-45
- user-defined snippets, 1-36
- user-defined types
 - creating, 5-30
- UserReports.xml, 1-45
- users
 - creating and editing, 5-30
- UT_REPO_ADMINISTRATOR role, 3-4
- UT_REPO_USER role, 3-4
- UtUtil executable or shell script, 3-11

V

- variable substitution
 - unit testing, 3-8
- versioning
 - preferences for, 1-63
 - SQL Developer support for, 1-37
- viewer (read-only) data modeling tool, 1-67
- viewlets
 - SQL Developer, 1-71
- views, 1-18
 - compiling, 1-18
 - creating and editing, 5-32
 - FROM clause, 5-34
 - GROUP BY clause, 5-35
 - HAVING clause, 5-35
 - options when creating or editing, 5-36
 - ORDER BY clause, 5-35
 - recompiling, 1-18
 - SELECT clause, 5-34
 - splitting the data pane, 1-25
 - WHERE clause, 5-35

W

- watches, 1-28
- WHERE clause, 5-35

X

- XML DB Repository, 1-18
- XML schema, 1-19
 - creating, 5-38
- xUnit frameworks, 3-1

Z

- zero date handling
 - MySQL connections, 5-8

