

Oracle TopLink

Developer's Guide, Volume 3

10g Release 3 (10.1.3)

B13593-02

February 2006

Understanding Relational Mappings

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States 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, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software–Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface	li
----------------------	-----------

Part I Building a TopLink Application

1 Understanding TopLink

What is TopLink?	1-1
Solving the Object-Persistence Impedance Mismatch	1-2
TopLink Key Features	1-4
TopLink Application Architectures.....	1-4

2 Understanding TopLink Application Development

Developing Your Application With TopLink.....	2-1
Typical Development Stages	2-1
Oracle Development Support.....	2-3
Designing Your Application With TopLink	2-3
Understanding TopLink Usage.....	2-3
Relational Database Usage	2-4
Object-Relational Database Usage.....	2-4
Oracle XML Database (XDB) Usage	2-4
Enterprise Information System (EIS) Usage.....	2-4
XML Usage.....	2-4
Understanding Target Platforms	2-4
Selecting an Architecture With TopLink.....	2-5
Tiers.....	2-5
Three Tier	2-5
Two Tier	2-6
Service Layer.....	2-6
EJB Session Beans.....	2-7
EJB Entity Beans	2-7
Plain Old Java Objects (POJO)	2-8
Data Access	2-8
Data Type	2-8
Multiple Data Sources	2-8
Isolating Data Access.....	2-8
Historical Data Access.....	2-9

Caching	2-9
Cache Type.....	2-9
Refreshing	2-9
Cache Coordination	2-9
Locking	2-10
Optimistic Locking	2-10
Pessimistic Locking.....	2-10
Building and Using the Persistence Layer	2-11
Implementation Options	2-11
Persistent Class Requirements	2-11
Persistence Layer Components	2-12
Mapping Metadata	2-12
Session Metadata.....	2-12
Cache.....	2-12
Queries and Expressions.....	2-13
Transactions	2-13
Using the Persistence Layer	2-13
Deploying the Application.....	2-14
Understanding Deployments	2-14
TopLink in a J2EE Application.....	2-14
Optimizing and Customizing the Application.....	2-15
Troubleshooting the Application.....	2-15
Understanding Object Persistence.....	2-15
Application Object Model.....	2-16
Data Storage Schema	2-16
Primary Keys and Object Identity	2-16
Mappings.....	2-16
Foreign Keys and Object Relationships	2-17
Inheritance.....	2-17
Concurrency.....	2-17
Caching	2-18
Nonintrusive Persistence	2-18
Indirection	2-18
Understanding TopLink Metadata	2-19
Advantages of the TopLink Metadata Architecture	2-19
Creating Project Metadata	2-20
Descriptors and Mappings	2-20
Data Source Login Information	2-20
Creating Session Metadata	2-21
Deploying Metadata	2-21
Understanding the Three-Tier Architecture	2-21
Example Implementations	2-22
Advantages and Disadvantages	2-22
Variation Using Remote Sessions	2-22
Technical Challenges	2-22
Understanding the Two-Tier Architecture	2-23
Example Implementations	2-24

Advantages and Disadvantages	2-24
Technical Challenges	2-24
Understanding the EJB Session Bean Facade Architecture.....	2-24
Example Implementation.....	2-25
Advantages and Disadvantages	2-25
Understanding Session Beans	2-25
Technical Challenges	2-26
Unit of Work Merge.....	2-26
Understanding the EJB Entity Beans with CMP Architecture	2-27
Example Implementation.....	2-28
Advantages and Disadvantages	2-28
Technical Challenges	2-28
External JDBC Pools	2-29
JTA/JTS Integration.....	2-29
Cache Coordination.....	2-29
Maintaining Bidirectional Relationships.....	2-29
Managing Dependent Objects in EJB 1.1	2-30
Managing Dependent Objects in EJB 2.0.....	2-32
Managing Collections of EJBObjects in EJB 1.1	2-32
Managing Collections of EJBObjects in EJB 2.0	2-33
Understanding the EJB Entity Beans With BMP Architecture	2-33
Example Implementations.....	2-35
Advantages and Disadvantages	2-35
Technical Challenges	2-35
External JDBC Pools	2-35
JTA/JTS Integration.....	2-36
Cache Coordination.....	2-36
Understanding the Web Services Architecture	2-36
Example Implementations.....	2-36
Advantages and Disadvantages	2-36
Technical Challenges	2-37

Part II Using TopLink Development Tools

3 Understanding TopLink Development Tools

Development Environment.....	3-2
TopLink Run-Time Environment	3-2

4 Using TopLink Workbench

Understanding TopLink Workbench	4-1
Configuring the TopLink Workbench Environment	4-2
Working With TopLink Workbench	4-3
Using the Menus	4-5
Menu Bar Menus.....	4-5
Context Menus	4-5
Using the Toolbars	4-6

Standard Toolbar	4-6
Context Toolbar	4-7
Using the Navigator.....	4-9
Using the Editor	4-10
Using the Problems Window	4-11
Using the Online Help.....	4-12
Working With TopLink Workbench Preferences	4-12
General Preferences	4-13
Help Preferences	4-14
Mappings Preferences	4-15
Class Preferences	4-16
EJB Preferences	4-16
Database Preferences	4-17
Sessions Configuration Preferences	4-18
New Names Preferences	4-19
Platform Preferences.....	4-20
Working With Databases	4-21
Working With Database Tables in the Navigator Window	4-21
Logging In and Out of a Database.....	4-21
Creating New Tables	4-22
Importing Tables From a Database	4-22
Removing Tables.....	4-24
Renaming Tables.....	4-24
Refreshing Tables From the Database.....	4-24
Working With Database Tables in the Editor Window	4-25
Working With Column Properties	4-25
Setting a Primary Key for Database Tables.....	4-26
Creating Table References	4-27
Creating Field Associations.....	4-28
Generating Data From Database Tables	4-29
Generating SQL Creation Scripts.....	4-29
Generating Classes and Descriptors From Database Tables	4-30
Generating EJB Entities and Descriptors From Database Tables.....	4-31
Generating Tables on the Database.....	4-32
Working With XML Schemas	4-33
Working With XML Schemas in the Navigator.....	4-33
Working With XML Schema Structure	4-34
Importing an XML Schema.....	4-34
Configuring XML Schema Reference	4-36
Using TopLink Workbench	4-36
Using Java	4-37
Configuring XML Schema Namespace.....	4-38
Using TopLink Workbench	4-38
Using Java	4-39
Working With Classes	4-40
Creating Classes	4-40
Using TopLink Workbench	4-40

Configuring Classes	4-41
Configuring Class Information	4-41
Configuring Class Modifiers	4-42
Configuring Class Interfaces	4-43
Adding Attributes	4-44
Configuring Attribute Modifiers	4-44
Configuring Attribute Type Information	4-45
Configuring Attribute Accessing Methods	4-46
Adding Methods	4-47
Configuring Method Modifiers	4-48
Configuring Method Type Information	4-48
Configuring Method Parameters	4-49
Importing and Updating Classes	4-50
Using TopLink Workbench	4-50
Managing Nondesoriptor Classes	4-51
Renaming Packages	4-52
Using TopLink Workbench	4-52
Integrating TopLink Workbench With Apache Ant	4-53
Configuring Ant to Use TopLink Workbench Tasks	4-53
Library Dependencies	4-53
Declaring TopLink Workbench Tasks	4-54
Understanding TopLink Workbench Ant Task API	4-54
Creating TopLink Workbench Ant Tasks	4-55
mappings.validate	4-56
Parameters	4-57
Parameters Specified as Nested Elements	4-57
Examples	4-57
session.validate	4-57
Parameters	4-58
Parameters Specified as Nested Elements	4-58
Examples	4-58
mappings.export	4-58
Parameters	4-58
Parameters Specified as Nested Elements	4-59
Examples	4-59
classpath	4-59
Parameters	4-60
Parameters Specified as Nested Elements	4-60
Examples	4-60
ignoreerror	4-60
Parameters	4-60
Parameters Specified as Nested Elements	4-60
Examples	4-61
ignoreerrorset	4-61
Parameters	4-61
Parameters Specified as Nested Elements	4-61
Examples	4-61

loginspec.....	4-62
Parameters.....	4-62
Parameters Specified as Nested Elements.....	4-62
Examples	4-62

5 Using an Integrated Development Environment

Configuring TopLink for Oracle JDeveloper	5-1
Using TopLink Mappings.....	5-1
Using TopLink Sessions	5-3
Configuring TopLink Workbench With Source Control Management Software.....	5-3
Using a Source Control Management System.....	5-3
Merging Files	5-4
Merging Project Files.....	5-4
Merging Table, Descriptor, and Class Files	5-5
Sharing Project Objects.....	5-6
Managing the ejb-jar.xml File.....	5-6
Working With Locked Files	5-6

6 Using the Schema Manager

Understanding the Schema Manager	6-1
Schema Manager Java and Database Type Conversion	6-3
Sequencing	6-3
Creating a Table Creator	6-4
Using TopLink Workbench During Development.....	6-4
Using the Default Table Generator at Run Time	6-4
Using Java.....	6-4
Creating a TableCreator Class.....	6-5
Creating a TableDefinition Class	6-5
Adding Fields to a TableDefinition.....	6-5
Defining Sybase and Microsoft SQL Server Native Sequencing.....	6-6
Creating Tables With a Table Creator.....	6-6
Automatic Database Table Creation.....	6-6

Part III Deploying a TopLink Application

7 Integrating TopLink With an Application Server

Application Server Support	7-1
Application Server Integration Concepts	7-2
Software Requirements	7-2
XML Parser Platform Configuration.....	7-2
Configuring XML Parser Platform	7-3
Creating an XML Parser Platform	7-3
XML Parser Limitations	7-3
Security Permissions.....	7-4
Persistence Manager Migration	7-4
Clustering	7-4

Oracle Containers for J2EE (OC4J)	7-5
CMP Integration.....	7-5
Migrating OC4J Orion Persistence to OC4J TopLink Persistence.....	7-5
Overview.....	7-6
Using the TopLink Migration Tool from TopLink Workbench.....	7-9
Using the TopLink Migration Tool From the Command Line.....	7-10
Post-Migration Changes.....	7-12
Troubleshooting Your Migration.....	7-13
JTA Integration.....	7-14
BEA WebLogic Server	7-14
Classpath	7-15
CMP Integration.....	7-15
Migrating BEA WebLogic Persistence to OC4J TopLink Persistence.....	7-16
Overview.....	7-16
Using the TopLink Migration Tool From TopLink Workbench.....	7-18
Using the TopLink Migration Tool From the Command Line.....	7-18
JTA Integration.....	7-20
Security Manager	7-20
IBM WebSphere Application Server	7-21
Classpath	7-21
Configuring Classpath for IBM WebSphere Application Server 4.0.....	7-21
Configuring Classpath for IBM WebSphere Application Server 5.0 and Later.....	7-21
CMP Integration.....	7-22
JTA Integration.....	7-22
Clustering on IBM WebSphere Application Server	7-22
Understanding Security Permissions	7-22
Permissions Required by TopLink Features	7-23
System Properties.....	7-23
Loading project.xml or sessions.xml Files.....	7-23
Cache Coordination.....	7-23
Accessing a Data Source by Port	7-24
Logging With java.util.logging	7-24
J2EE Application Deployment.....	7-24
Permissions Required When doPrivileged is Disabled	7-24
Disabling doPrivileged Operation.....	7-25
Configuring Miscellaneous EJB Options	7-25
Setter Parameter Type Checking.....	7-25
Unknown Primary Key Class Support.....	7-25
Single-Object Finder Return Type Checking	7-26

8 Creating TopLink Files for Deployment

Understanding TopLink Deployment File Creation	8-1
project.xml File	8-2
XSD File Format	8-2
Non-CMP Applications and Project Metadata.....	8-2
CMP Applications and Project Metadata	8-3
Creating project.xml With TopLink Workbench.....	8-3

Creating project.xml Programatically	8-3
EJB 3.0 and the project.xml File	8-3
sessions.xml File	8-3
XSD File Format	8-4
Non-CMP Applications and Session Metadata	8-4
CMP Applications and Session Metadata	8-4
EJB 3.0 and the sessions.xml File	8-5
ejb-jar.xml File	8-5
EJB 3.0 and the ejb-jar.xml File	8-5
<J2EE-Container>-ejb-jar.xml File	8-5
OC4J and the orion-ejb-jar.xml File	8-5
BEA WebLogic Server and the weblogic-ejb-jar.xml File	8-6
EJB 3.0 and the <J2EE-Container>-jar.xml File	8-6
toplink-ejb-jar.xml File	8-6
OC4J and the toplink-ejb-jar.xml File	8-6
BEA WebLogic Server and the toplink-ejb-jar.xml File	8-7
IBM WebSphere Application Server and the toplink-ejb-jar.xml File	8-8
EJB 3.0 and the toplink-ejb-jar.xml File	8-8
Java Applications	8-8
JavaServer Pages and Servlet Applications	8-8
Session Bean Applications	8-8
CMP Applications	8-9
BMP Applications	8-9
Configuring the orion-ejb-jar.xml File for OC4J	8-9
Configuring persistence-manager Entries	8-9
Configuring pm-properties	8-10
Configuring cache-synchronization Properties	8-11
Configuring default-mapping Properties	8-12
Configuring the weblogic-ejb-jar.xml File for BEA WebLogic Server	8-13
Configuring persistence-descriptor Entries	8-13
Unsupported weblogic-ejb-jar.xml File Tags	8-14

9 Packaging a TopLink Application

Java Applications	9-1
JavaServer Pages and Servlet Applications	9-2
TopLink Domain JAR	9-2
Session Bean Applications	9-3
TopLink Domain JAR	9-3
EJB JAR	9-4
CMP Applications	9-4
EJB JAR	9-4
BMP Applications	9-5
TopLink Domain JAR	9-6
EJB JAR	9-7
Packaging With TopLink Metadata File Resource Paths	9-7

10 Deploying a TopLink Application

Java Applications	10-1
JavaServer Pages and Servlets	10-1
Session Bean Applications	10-1
CMP Applications	10-2
Deploying a CMP Application to OC4J	10-2
Deploying a CMP Application to BEA WebLogic Server	10-2
Troubleshooting ejbc	10-3
Deploying a CMP Application to IBM WebSphere Application Server 4.0	10-3
Starting the Entity Bean	10-3
BMP Applications	10-4
Hot Deployment of EJB	10-4
Hot Deployment in a CMP Application	10-4
Hot Deployment in a non-CMP Application	10-4
Using the WebSphere Deploy Tool	10-5
Using the Deploy Tool on its Own	10-5
Using the Deploy Tool With WebSphere Studio Application Developer	10-5
Troubleshooting	10-6

Part IV Optimizing and Customizing a TopLink Application

11 Optimization

Understanding Optimization	11-1
Sources of Application Performance Problems	11-2
Measuring Performance With the TopLink Profiler	11-2
Configuring the TopLink Performance Profiler	11-3
Accessing the TopLink Profiler Results	11-3
Measuring Performance With the Oracle Dynamic Monitoring System (DMS)	11-4
Configuring the Oracle DMS Profiler	11-6
OC4J Applications	11-7
Non-OC4J Applications	11-7
Accessing Oracle DMS Profiler Data Using JMX	11-7
Accessing Oracle DMS Profiler Data Using the DMS Spy Servlet.....	11-7
General Performance Optimization	11-8
Schema Optimization	11-8
Schema Case 1: Aggregation of Two Tables into One	11-9
Schema Case 2: Splitting One Table Into Many	11-9
Schema Case 3: Collapsed Hierarchy	11-11
Schema Case 4: Choosing One out of Many	11-12
Mapping and Descriptor Optimization	11-13
Session Optimization	11-13
Cache Optimization	11-13
Data Access Optimization	11-14
JDBC Driver Properties Optimization.....	11-14
Data Format Optimization.....	11-15
Batch Writing	11-15

Parameterized SQL (Binding) and Prepared Statement Caching	11-15
Query Optimization	11-16
Parameterized SQL and Prepared Statement Caching.....	11-17
Named Queries.....	11-17
Batch and Join Reading	11-17
Partial Object Queries and Fetch Groups	11-17
JDBC Fetch Size	11-17
Cursored Streams and Scrollable Cursors.....	11-18
Read Optimization Examples.....	11-19
Reading Case 1: Displaying Names in a List	11-20
Reading Case 2: Batch Reading Objects.....	11-22
Reading Case 3: Using Complex Custom SQL Queries	11-24
Reading Case 4: Using View Objects	11-24
Reading Case 5: Inheritance Views	11-26
Write Optimization Examples.....	11-26
Writing Case: Batch Writes.....	11-27
Unit of Work Optimization	11-30
Application Server and Database Optimization	11-30

12 Customization

Overview	12-1
Creating Custom Data Types	12-1
Using Public Source.....	12-2

Part V Troubleshooting a TopLink Application

13 TopLink Exception Reference

Descriptor Exceptions (1 – 200)	13-2
Concurrency Exceptions (2001 – 2006)	13-25
Conversion Exceptions (3001– 3008)	13-26
Database Exceptions (4002 – 4018).....	13-27
Optimistic Lock Exceptions (5001 – 5008)	13-29
Query Exceptions (6001 – 6121)	13-30
Validation Exceptions (7001 – 7147)	13-42
EJB QL Exceptions (8001 – 8010)	13-55
Session Loader Exceptions (9000 - 9010)	13-56
EJB Exception Factory Exceptions (10001 - 10069)	13-58
Communication Exceptions (12000 - 12003).....	13-64
XML Data Store Exceptions (13000 - 13020)	13-64
Deployment Exceptions (14001 - 14033)	13-67
Synchronization Exceptions (15001 - 15025)	13-70
SDK Data Store Exceptions (17001 - 17006)	13-71
EIS Exceptions (17007 – 17025)	13-72
JMS Processing Exceptions (18001 - 18002).....	13-74
SDK Descriptor Exceptions (19001 - 19003)	13-75
Default Mapping Exceptions (20001 - 20008)	13-75

Discovery Exceptions (22001 - 22004).....	13-77
Remote Command Manager Exceptions (22101 - 22111)	13-77
Transaction Exceptions (23001 - 23011).....	13-79
XML Conversion Exceptions (25001 - 25016)	13-80
Migration Utility Exceptions (26001 - 26017).....	13-82
EJB JAR XML Exceptions (72000 – 72022)	13-84

14 TopLink Workbench Error Reference

Miscellaneous Errors (1 – 89).....	14-1
Project Errors (100 – 126)	14-2
Descriptor Errors (200 – 350).....	14-4
Mapping Errors (400 – 483)	14-11
Table Errors (500 – 610).....	14-15
XML Schema Errors (700 – 706)	14-20
Session Errors (800 – 812)	14-21
Common Classpath Problems.....	14-22
Data Source Problems	14-23
Database Connection Problems	14-23

15 Troubleshooting Application Deployment

Generating Deployment JAR Files	15-1
Common J2SE Deployment Exceptions.....	15-1
Classpath Exceptions	15-2
Communication Exceptions.....	15-2
Descriptor Validation Exceptions	15-2
Common BEA WebLogic Server Deployment Exceptions	15-3
Common BEA WebLogic Server 6.1 Exceptions.....	15-5
Development Exceptions	15-6
Deployment and Run Time Exceptions	15-6
Common BEA WebLogic 7.0 Exceptions.....	15-8
Development Exceptions	15-8
Deployment Exceptions	15-9
Common BEA WebLogic 8.1 Exceptions.....	15-10
Development Exceptions	15-10
Deployment Exceptions	15-11
Common IBM WebSphere Application Server Exceptions	15-12
Problems at Run Time	15-14
Common TopLink for IBM WebSphere Deploy Tool Exceptions.....	15-14

Part VI TopLink Tutorial

16 Planning the Tutorial Project

Overview	16-1
Before you Begin	16-1
Understanding the Tutorial Project.....	16-2
Source Files.....	16-2

Object Model Classes.....	16-2
Application Classes	16-3
Data Model.....	16-4
Creating and Populating Database Tables	16-7

17 Building the Tutorial Project

Step 1: Creating a Project	17-1
TopLink Workbench Project.....	17-1
Adding Java Classes	17-3
Step 2: Adding a Data Source.....	17-5
Creating a Database Development Login.....	17-5
Importing Database Tables	17-6
Project-Level Sequencing	17-8
Step 3: Associating Descriptors to Tables	17-9
Associating With Multiple Tables	17-10
Descriptor-Level Sequencing.....	17-11
Step 4: Mapping Attributes	17-12
Direct-to-Field Mappings.....	17-12
Using Automap	17-13
Additional Direct-to-Field Mappings	17-14
One-to-One Mappings.....	17-14
Additional One-to-One Mappings	17-16
One-to-Many Mappings.....	17-16
Additional One-to-Many Mappings	17-18
Object Type Mappings	17-18
Aggregate Object Mappings.....	17-19
Aggregate Descriptor	17-20
Aggregate Mapping.....	17-20
Direct Collection Mappings.....	17-21
Many-to-Many Mappings.....	17-21
Transformation Mappings.....	17-23
Step 5: Using Advanced Descriptor Properties	17-25
Inheritance.....	17-25
Root Parent Class	17-25
Child Class	17-26
Indicator Values	17-27
Locking Policy	17-28
Named Queries.....	17-29
Step 6: Verifying the Project	17-32

18 Deploying the Tutorial Project

Step 1: Generating Deployment Information.....	18-1
Creating the project.xml File	18-1
Step 2: Creating Sessions Configuration	18-3
Creating the sessions.xml File	18-3
Creating a Server Session.....	18-3
Associating the Project With the Session.....	18-4

Updating the data-sources.xml File.....	18-5
Creating a Database Deployment Login.....	18-6
Configuring Logging Options.....	18-7
Step 3: Packaging for Deployment.....	18-8
Domain JAR File.....	18-8
WAR File	18-8
EAR File.....	18-8
Step 4: Deploying to an Application Server.....	18-9
Step 5: Running the Tutorial Application	18-9

Part VII Mapping and Configuration Overview

19 Understanding TopLink Mapping and Configuration Concepts

Mapping and Configuration Concepts	19-1
Projects.....	19-1
Descriptors	19-2
Mappings.....	19-2

Part VIII Projects

20 Understanding Projects

TopLink Project Types	20-1
Project Concepts	20-2
Project Architecture	20-2
Relational and Nonrelational Projects.....	20-2
Persistent and Nonpersistent Projects.....	20-2
Projects and Login.....	20-3
Non-CMP Session Role: Session Login.....	20-3
CMP Deployment Role: Deployment Login	20-3
Development Role: Development Login	20-4
Projects and Platforms.....	20-4
Projects and Sequencing.....	20-4
Configuring how to Obtain Sequence Values.....	20-5
Configuring Where to Write Sequence Values.....	20-5
XML Namespaces.....	20-5
Relational Projects.....	20-6
Building Relational Projects for a Relational Database	20-6
Building Relational Projects for an Object-Relational Database	20-6
EIS Projects	20-7
Building EIS Projects with XML Records	20-9
Building EIS Projects With Indexed or Mapped Records	20-9
XML Projects	20-9
TopLink Support for Java Architecture for XML Binding (JAXB)	20-10
Understanding JAXB-Specific Generated Files	20-10
Understanding TopLink-Specific Generated Files.....	20-11
Using TopLink JAXB Compiler Generated Files at Run Time	20-13

JAXB Validation	20-14
Understanding the Project API	20-14
Project Inheritance Hierarchy.....	20-14
Understanding Sequencing in Relational Projects	20-14
Sequencing Configuration Options	20-15
Sequencing Types.....	20-16
Table Sequencing	20-16
Unary Table Sequencing	20-17
Query Sequencing.....	20-18
Default Sequencing.....	20-18
Native Sequencing With an Oracle Database Platform.....	20-18
Native Sequencing With a Non-Oracle Database Platform.....	20-19
Sequencing and Preallocation Size	20-20
Sequencing With CMP Entity Beans	20-21
Understanding XML Namespaces	20-22
TopLink Workbench Namespace Resolution	20-22
Element and Attribute Form Options	20-22
Element Form Default Qualified and Attribute Form Default Unqualified	20-23
Element and Attribute Form Default Unqualified.....	20-24
Element and Attribute Form Default Qualified	20-25
TopLink Runtime Namespace Resolution.....	20-26

21 Creating a Project

Project Creation Overview	21-1
Using TopLink Workbench	21-2
Creating New TopLink Workbench Projects.....	21-2
Using Java.....	21-3
Creating a Project for an Existing Object and Data Model	21-5
Using TopLink Workbench	21-5
Creating a Project From an Existing Object Model	21-5
Using TopLink Workbench	21-5
Creating a Project From an Existing Data Model	21-5
Using TopLink Workbench	21-6
Creating an XML Project From an XML Schema	21-6
Using TopLink Workbench	21-6
Using the Command Line	21-8
Creating a Project by Migrating an EAR to OC4J	21-9
Creating a Project From an OC4J EJB 2.0 CMP EAR at Deployment Time	21-10
Working With Projects	21-10
Opening Existing Projects	21-10
Saving Projects.....	21-11
Saving Projects With a New Name or Location	21-12
Generating the Project Status Report	21-12
Exporting Project Information	21-13
Exporting Deployment XML Information.....	21-14
Exporting Model Java Source.....	21-14
Exporting Project Java Source	21-14

Exporting Table Creator Files.....	21-15
Working With the ejb-jar.xml File	21-15
Writing to the ejb-jar.xml File.....	21-16
Reading From the ejb-jar.xml File.....	21-16

22 Configuring a Project

Configuring Common Project Options.....	22-1
Configuring Project Save Location	22-2
Using TopLink Workbench	22-2
Configuring Project Classpath.....	22-3
Using TopLink Workbench	22-3
Configuring Mapped Field Access at the Project Level.....	22-4
Using TopLink Workbench	22-4
Configuring Persistence Type	22-5
Using TopLink Workbench	22-6
Configuring Default Descriptor Advanced Properties.....	22-7
Using TopLink Workbench	22-8
Configuring Existence Checking at the Project Level.....	22-8
Using TopLink Workbench	22-9
Configuring Project Deployment XML Options.....	22-10
Using TopLink Workbench	22-10
Configuring Model Java Source Code Options.....	22-11
Using TopLink Workbench	22-11
Configuring Deprecated Direct Mappings	22-12
Using TopLink Workbench	22-13
Configuring Cache Type and Size at the Project Level	22-13
Using TopLink Workbench	22-14
Configuring Cache Isolation at the Project Level	22-16
Using TopLink Workbench	22-16
Configuring Cache Coordination Change Propagation at the Project Level.....	22-17
Using TopLink Workbench	22-18
Configuring Cache Expiration at the Project Level	22-19
Using TopLink Workbench	22-20
Configuring Project Comments.....	22-21
Using TopLink Workbench	22-21

23 Configuring a Relational Project

Relational Project Configuration Overview	23-1
Configuring Relational Database Platform at the Project Level.....	23-2
Using TopLink Workbench	23-2
Configuring Sequencing at the Project Level.....	23-3
Using TopLink Workbench	23-3
Using Java.....	23-4
Configuring Login Information	23-5
Using TopLink Workbench	23-5
Configuring Development and Deployment Logins	23-6

Using TopLink Workbench	23-6
Logging in to the Database	23-7
Configuring Named Query Parameterized SQL and Statement Caching at the Project Level.....	23-7
Using TopLink Workbench	23-8
Configuring Table Generation Options	23-9
Using TopLink Workbench	23-9
Configuring Table Creator Java Source Options	23-10
Using TopLink Workbench	23-10
Configuring Project Java Source Code Options.....	23-11
Using TopLink Workbench	23-11

24 Configuring an EIS Project

EIS Project Configuration Overview.....	24-1
Configuring EIS Data Source Platform at the Project Level.....	24-2
Using TopLink Workbench	24-2
Configuring EIS Connection Specification Options at the Project Level.....	24-2
Using TopLink Workbench	24-3

25 Configuring an XML Project

XML Project Configuration Overview	25-1
--	------

Part IX Descriptors

26 Understanding Descriptors

Descriptor Types	26-1
Descriptor Concepts.....	26-2
Descriptor Architecture.....	26-2
Descriptors and Inheritance.....	26-3
Descriptors and EJB	26-3
Nondeferred Changes	26-3
Creating a New Entity Bean and ejbCreate / ejbPostCreate Methods	26-4
Inheritance	26-4
Fetch Groups	26-4
Amendment and After-Load Methods	26-5
Descriptors and Aggregation	26-5
Aggregate and Composite Descriptors in Relational Projects	26-5
Root and Composite Descriptors in EIS Projects.....	26-8
Composite Descriptors in XML Projects.....	26-8
Descriptor Event Manager.....	26-8
Descriptor Query Manager.....	26-8
Descriptors and Sequencing	26-9
Descriptors and Locking	26-9
Default Root Element.....	26-9
Relational Descriptors.....	26-11
Object-Relational Descriptors	26-11

EIS Descriptors	26-11
XML Descriptors.....	26-12
Understanding Descriptors and Inheritance	26-12
Specifying a Class Indicator.....	26-13
Using Class Indicator Fields.....	26-13
Using Class Extraction Methods.....	26-14
Inheritance and Primary Keys (Relational and EIS Only).....	26-15
Single and Multi-Table Inheritance (Relational Only).....	26-15
Single Table Inheritance.....	26-15
Multitable Inheritance.....	26-16
Aggregate and Composite Descriptors and Inheritance	26-17
Inheritance and EJB.....	26-17
Understanding Descriptors and Locking	26-17
Optimistic Version Locking Policies	26-17
Optimistic Version Locking Policies and Cascading	26-18
Optimistic Locking and Rollbacks.....	26-20
Optimistic Field Locking Policies	26-20
Pessimistic Locking Policy.....	26-21
Locking in a Three-Tier Application	26-21
Optimistic Locking in a Three-Tier Application	26-21
Pessimistic Locking in a Three-Tier Application	26-22
Understanding the Descriptor API	26-22
Descriptor Inheritance Hierarchy	26-23

27 Creating a Descriptor

Descriptor Creation Overview	27-1
Creating a Relational Descriptor.....	27-1
Using TopLink Workbench	27-2
Relational Class Descriptors.....	27-2
Relational Aggregate Descriptors.....	27-2
Relational Interface Descriptors.....	27-2
Using Java.....	27-2
Creating an Object-Relational Descriptor.....	27-3
Using Java.....	27-3
Creating an EIS Descriptor.....	27-4
Using TopLink Workbench	27-4
EIS Root Descriptors.....	27-5
EIS Composite Descriptors	27-5
Using Java.....	27-5
Creating an XML Descriptor	27-5
Using TopLink Workbench	27-5
Using Java.....	27-5
Validating Descriptors	27-6
Generating Java Code for Descriptors	27-6

28 Configuring a Descriptor

Configuring Common Descriptor Options	28-1
Configuring Primary Keys	28-2
Using TopLink Workbench	28-3
Using Java.....	28-4
Relational Projects.....	28-4
EIS Projects.....	28-4
Configuring Read-Only Descriptors	28-4
Using Read-Only Entity Beans.....	28-5
Using TopLink Workbench	28-5
Using Java.....	28-6
Configuring Unit of Work Conforming at the Descriptor Level	28-6
Using TopLink Workbench	28-7
Using Java.....	28-7
Configuring Descriptor Alias	28-7
Using TopLink Workbench	28-8
Using Java.....	28-9
Configuring Descriptor Comments	28-9
Using TopLink Workbench	28-9
Configuring Named Queries at the Descriptor Level	28-10
Using TopLink Workbench	28-10
Adding Named Queries.....	28-12
Configuring Named Query Type and Parameters	28-13
Configuring Named Query Selection Criteria.....	28-14
Configuring Read All Query Order	28-15
Configuring Named Query Optimization.....	28-16
Configuring Named Query Attributes	28-17
Configuring Named Query Group/Order Options	28-19
Creating an EIS Interaction for a Named Query	28-20
Configuring Named Query Options.....	28-22
Configuring Named Query Advanced Options.....	28-24
Using Java.....	28-25
Configuring Query Timeout at the Descriptor Level.....	28-26
Using TopLink Workbench	28-26
Using Java.....	28-27
Configuring Cache Refreshing.....	28-27
Using TopLink Workbench	28-28
Using Java.....	28-29
Configuring Query Keys.....	28-29
Using TopLink Workbench	28-31
Using Java.....	28-31
Configuring Interface Query Keys	28-33
Using TopLink Workbench	28-34
Using Java.....	28-34
Configuring Cache Type and Size at the Descriptor Level.....	28-35
Using TopLink Workbench	28-35
Using Java.....	28-37

Configuring Cache Isolation at the Descriptor Level	28-37
Using TopLink Workbench	28-37
Using Java.....	28-38
Configuring Cache Coordination Change Propagation at the Descriptor Level	28-38
Using TopLink Workbench	28-39
Using Java.....	28-40
Configuring Cache Expiration at the Descriptor Level.....	28-40
Using TopLink Workbench	28-41
Using Java.....	28-42
Configuring Cache Existence Checking at the Descriptor Level	28-42
Using TopLink Workbench	28-42
Using Java.....	28-43
Configuring a Descriptor With EJB Information	28-44
Using TopLink Workbench	28-44
Using Java.....	28-46
Configuring CMP Information	28-46
Configuring BMP Information.....	28-47
Configuring Reading Subclasses on Queries	28-47
Using TopLink Workbench	28-48
Using Java.....	28-48
Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor	28-49
Using TopLink Workbench	28-49
Using Java.....	28-50
Configuring Inheritance for a Parent (Root) Descriptor.....	28-50
Using TopLink Workbench	28-51
Using Java.....	28-52
Configuring Inheritance Expressions for a Parent (Root) Class Descriptor	28-53
Using Java.....	28-55
Configuring Inherited Attribute Mapping in a Subclass.....	28-56
Using TopLink Workbench	28-56
Using Java.....	28-57
Configuring a Domain Object Method as an Event Handler.....	28-57
Using TopLink Workbench	28-58
Using Java.....	28-60
Configuring a Descriptor Event Listener as an Event Handler.....	28-60
Using Java.....	28-62
Configuring Locking Policy	28-62
Using TopLink Workbench	28-63
Using Java.....	28-65
Configuring an Optimistic Locking Policy	28-65
Configuring Optimistic Locking Policy Cascading	28-65
Configuring a Pessimistic Locking Policy	28-65
Configuring Returning Policy	28-65
Using TopLink Workbench	28-66
Using Java.....	28-67
Configuring Instantiation Policy	28-68
Using TopLink Workbench	28-68

Using Java.....	28-69
Configuring Copy Policy	28-69
Using TopLink Workbench	28-70
Using Java.....	28-70
Configuring Change Policy	28-70
Using Java.....	28-71
Configuring Deferred Change Detection Policy	28-71
Configuring Object Change Tracking Policy	28-71
Configuring Attribute Change Tracking Policy	28-72
Configuring a History Policy	28-73
Using Java.....	28-75
Configuring Write Responsibility	28-75
Configuring Wrapper Policy	28-75
Using Java.....	28-76
Configuring Fetch Groups.....	28-77
Using Java.....	28-78
Configuring Amendment Methods	28-78
Using TopLink Workbench	28-79

29 Configuring a Relational Descriptor

Relational Descriptor Configuration Overview.....	29-1
Configuring Associated Tables	29-2
Using TopLink Workbench	29-2
Using Java.....	29-3
Configuring Sequencing at the Descriptor Level	29-3
Using TopLink Workbench	29-4
Using Java.....	29-5
Configuring a Sequence by Name	29-5
Configuring the Same Sequence for Multiple Descriptors	29-5
Configuring the Platform Default Sequence	29-6
Configuring Custom SQL Queries for Basic Persistence Operations.....	29-6
Using TopLink Workbench	29-7
Using Java.....	29-8
Configuring Interface Alias	29-10
Using TopLink Workbench	29-11
Using Java.....	29-11
Configuring a Relational Descriptor as a Class or Aggregate Type.....	29-11
Using TopLink Workbench	29-12
Using Java.....	29-12
Configuring Multitable Information	29-13
Using TopLink Workbench	29-13
Using Java.....	29-14

30 Configuring an Object-Relational Descriptor

Object-Relational Descriptor Configuration Overview	30-1
Configuring Field Ordering	30-2
Using Java.....	30-2

31 Configuring an EIS Descriptor

EIS Descriptor Configuration Overview	31-1
Configuring Schema Context for an EIS Descriptor	31-2
Using TopLink Workbench	31-2
Choosing a Schema Context	31-3
Using Java	31-3
Configuring Default Root Element	31-3
Using TopLink Workbench	31-4
Choosing a Root Element	31-4
Using Java	31-5
Configuring Record Format	31-5
Using Java	31-6
Configuring Custom EIS Interactions for Basic Persistence Operations	31-6
Using TopLink Workbench	31-6
Using Java	31-8
Configuring an EIS Descriptor as a Root or Composite Type	31-8
Using TopLink Workbench	31-9
Using Java	31-9

32 Configuring an XML Descriptor

XML Descriptor Configuration Overview	32-1
Configuring Schema Context for an XML Descriptor	32-1
Using TopLink Workbench	32-2
Choosing a Schema Context	32-2
Using Java	32-3
Configuring for Complex Type of anyType	32-3
Using TopLink Workbench	32-4
Configuring Default Root Element	32-5
Using TopLink Workbench	32-5
Choosing a Root Element	32-5
Configuring Document Preservation	32-6
Using TopLink Workbench	32-6
Using Java	32-6

Part X Mappings

33 Understanding Mappings

Mapping Types	33-1
Mapping Concepts	33-2
Mapping Architecture	33-2
Example Mapping	33-3
Automatic Mappings	33-4
Automapping With TopLink Workbench at Development Time	33-4
Default Mapping in EJB 2.0 or 3.0 CMP Projects Using OC4J at Run Time	33-4
JAXB Project Generation at Development Time	33-5
Indirection	33-5

Value Holder Indirection.....	33-6
Transparent Indirect Container Indirection.....	33-8
Proxy Indirection	33-8
Indirection and EJB.....	33-9
Indirection, Serialization, and Detachment.....	33-9
Method Accessors and Attribute Accessors.....	33-10
Mapping Converters and Transformers	33-10
Serialized Object Converter.....	33-10
Type Conversion Converter	33-11
Object Type Converter	33-12
Simple Type Translator.....	33-12
Transformation Mappings.....	33-14
Mappings and XPath	33-15
XPath by Position.....	33-15
XPath by Path and Name.....	33-15
XPath by Name.....	33-16
Self XPath	33-16
Mappings and xsd:list and xsd:union Types	33-17
Mapping an xsd:union Type	33-17
Mapping an xsd:list Type	33-17
Mapping a List of Unions	33-18
Mapping a Union of Lists	33-19
Mapping a Union of Unions.....	33-19
Mappings and the jaxb:class Customization.....	33-20
all, choice, or sequence Structure.....	33-20
group Structure	33-21
sequence or choice Structure Containing a group	33-21
group Structure Containing a sequence or choice	33-22
group Structure Containing a group.....	33-23
Limitations of jaxb:class Customization Support.....	33-23
Mappings and JAXB Typesafe Enumerations.....	33-24
Understanding the Mapping API.....	33-25
Relational Mappings	33-25
Object-Relational Mappings.....	33-26
XML Mappings	33-27
EIS Mappings.....	33-27

34 Creating a Mapping

Mapping Creation Overview	34-1
Creating Mappings Manually During Development	34-1
Using TopLink Workbench	34-1
Creating Mappings Automatically During Development	34-2
Using TopLink Workbench	34-2
Creating Mappings Automatically During Deployment	34-2
Creating Mappings to Oracle LOB Database Objects.....	34-2
Using the Oracle JDBC OCI Driver or Server Driver.....	34-3
Using the Oracle JDBC Thin Driver	34-5

Removing Mappings	34-6
Using TopLink Workbench	34-6

35 Configuring a Mapping

Configuring Common Mapping Options.....	35-1
Configuring Read-Only Mappings.....	35-2
Using TopLink Workbench	35-3
Using Java.....	35-3
Configuring Indirection.....	35-3
Using TopLink Workbench	35-4
Using Java.....	35-5
Configuring ValueHolder Indirection	35-6
Configuring ValueHolder Indirection With Method Accessing.....	35-7
Configuring ValueHolder Indirection With EJB 3.0 on OC4J	35-8
Configuring IndirectContainer Indirection	35-8
Configuring Proxy Indirection.....	35-9
Configuring XPath	35-10
Using TopLink Workbench	35-11
Choosing the XPath	35-12
Configuring a Default Null Value at the Mapping Level.....	35-12
Using TopLink Workbench	35-13
Using Java.....	35-13
Configuring Method Accessing.....	35-14
Using TopLink Workbench	35-14
Using Java.....	35-15
Configuring Private or Independent Relationships.....	35-16
Using TopLink Workbench	35-17
Using Java.....	35-17
Configuring Mapping Comments.....	35-18
Using TopLink Workbench	35-18
Configuring a Serialized Object Converter	35-18
Using TopLink Workbench	35-19
Using Java.....	35-19
Configuring a Type Conversion Converter.....	35-20
Using TopLink Workbench	35-20
Using Java.....	35-21
Configuring an Object Type Converter.....	35-22
Using TopLink Workbench	35-22
Using Java.....	35-23
Configuring a Simple Type Translator.....	35-23
Using TopLink Workbench	35-24
Using Java.....	35-24
Configuring a JAXB Typesafe Enumeration Converter.....	35-25
Using Java.....	35-26
Configuring Container Policy.....	35-26
Using TopLink Workbench	35-27
Using Java.....	35-28

Configuring Attribute Transformer	35-29
Using TopLink Workbench	35-30
Using Java.....	35-31
Configuring Field Transformer Associations	35-31
Using TopLink Workbench	35-32
Specifying Field-to-Transformer Associations	35-32
Using Java.....	35-33
Configuring Mutable Mappings	35-33
Using TopLink Workbench	35-34
Using Java.....	35-34
Configuring Bidirectional Relationship	35-34
Using TopLink Workbench	35-35
Configuring the Use of a Single Node	35-36
Using TopLink Workbench	35-36
Using Java.....	35-37

Part XI Relational Mappings

36 Understanding Relational Mappings

Relational Mapping Types	36-1
Relational Mapping Concepts	36-2
Directionality	36-2
Converters and Transformers	36-3
Using a Direct Mapping	36-3
Using a Converter Mapping	36-3
Using a Transformation Mapping	36-3
Relational Mappings and EJB.....	36-3
Direct-to-Field Mapping	36-4
Direct to XMLType Mapping.....	36-4
One-to-One Mapping	36-5
One-to-One Mappings and EJB.....	36-6
Variable One-to-One Mapping	36-6
One-to-Many Mapping	36-7
One-to-Many Mappings and EJB.....	36-8
Many-to-Many Mapping	36-8
Many-to-Many Mappings and EJB.....	36-9
Aggregate Collection Mapping	36-10
Aggregate Collection Mappings and Inheritance	36-10
Aggregate Collection Mappings and EJB	36-11
Implementing Aggregate Collection Mappings	36-11
Direct Collection Mapping.....	36-11
Direct Map Mapping	36-12
Aggregate Object Mapping	36-12
Aggregate Object Mappings with a Single Source Object.....	36-13
Aggregate Object Mappings With Multiple Source Objects	36-14
Implementing an Aggregate Object Relationship Mapping	36-14
Transformation Mapping	36-15

37	Configuring a Relational Mapping	
	Configuring Common Relational Mapping Options.....	37-1
	Configuring a Database Field.....	37-2
	Using TopLink Workbench	37-4
	Configuring Reference Descriptor.....	37-5
	Using TopLink Workbench	37-5
	Configuring Batch Reading.....	37-6
	Using TopLink Workbench	37-7
	Using Java.....	37-7
	Configuring Query Key Order	37-8
	Using TopLink Workbench	37-8
	Configuring Table and Field References (Foreign and Target Foreign Keys)	37-8
	Using TopLink Workbench	37-9
38	Configuring a Relational Direct-to-Field Mapping	
	Relational Direct-to-Field Mapping Configuration Overview	38-1
39	Configuring a Relational Direct-to-XMLType Mapping	
	Relational Direct-to-XMLType Mapping Overview	39-1
	Configuring Read Whole Document.....	39-1
	Using TopLink Workbench	39-1
40	Configuring a Relational One-to-One Mapping	
	Relational One-to-One Mapping Configuration Overview	40-1
	Configuring Joining at the Mapping Level.....	40-1
	Using TopLink Workbench	40-2
41	Configuring a Relational Variable One-to-One Mapping	
	Relational Variable One-to-One Mapping Configuration Overview.....	41-1
	Configuring Class Indicator.....	41-1
	Using TopLink Workbench	41-2
	Configuring Unique Primary Key	41-3
	Understanding Unique Primary Key	41-3
	Using TopLink Workbench	41-3
	Using Java.....	41-4
	Configuring Query Key Association.....	41-4
	Using TopLink Workbench	41-4
42	Configuring a Relational One-to-Many Mapping	
	Relational One-to-Many Mapping Configuration Overview	42-1
43	Configuring a Relational Many-to-Many Mapping	
	Relational Many-to-Many Mapping Configuration Overview.....	43-1
	Configuring a Relation Table	43-1

Using TopLink Workbench	43-2
44 Configuring a Relational Aggregate Collection Mapping	
Relational Aggregate Collection Mapping Configuration Overview.....	44-1
45 Configuring a Relational Direct Collection Mapping	
Relational Direct Collection Mapping Configuration Overview	45-1
Configuring Target Table	45-1
Using TopLink Workbench	45-2
Configuring Direct Value Field.....	45-2
Using TopLink Workbench	45-2
46 Configuring a Relational Direct Map Mapping	
Relational Direct Map Mapping Configuration Overview	46-1
Configuring Direct Value Field.....	46-1
Using TopLink Workbench	46-1
Configuring Direct Key Field	46-2
Using TopLink Workbench	46-2
Configuring Key Converters.....	46-3
Using TopLink Workbench	46-3
Configuring Value Converters	46-4
Using TopLink Workbench	46-4
47 Configuring a Relational Aggregate Object Mapping	
Relational Aggregate Object Mapping Configuration Overview	47-1
Configuring Aggregate Fields	47-1
Using TopLink Workbench	47-2
Configuring Allowing Null Values.....	47-2
Using TopLink Workbench	47-2
48 Configuring a Relational Transformation Mapping	
Relational Transformation Mapping Configuration Overview	48-1
Part XII Object-Relational Mappings	
49 Understanding Object-Relational Mappings	
Object-Relational Mapping Types	49-1
Object-Relational Structure Mapping.....	49-2
Object-Relational Reference Mapping	49-2
Object-Relational Array Mapping	49-2
Object-Relational Object Array Mapping.....	49-2
Object-Relational Nested Table Mapping	49-3

50	Configuring an Object-Relational Mapping	
	Configuring Common Object-Relational Mapping Options	50-1
	Configuring Reference Class	50-2
	Using Java.....	50-2
	Configuring Attribute Name	50-2
	Using Java.....	50-3
	Configuring Field Name	50-3
	Using Java.....	50-3
	Configuring Structure Name	50-4
	Using Java.....	50-4
51	Configuring an Object-Relational Structure Mapping	
	Object-Relational Structure Mapping Configuration Overview.....	51-1
52	Configuring an Object-Relational Reference Mapping	
	Object-Relational Reference Mapping Configuration Overview	52-1
53	Configuring an Object-Relational Array Mapping	
	Object-Relational Array Mapping Configuration Overview	53-1
54	Configuring an Object-Relational Object Array Mapping	
	Object-Relational Object Array Mapping Configuration Overview.....	54-1
55	Configuring an Object-Relational Nested Table Mapping	
	Object-Relational Nested Table Mapping Configuration Overview	55-1
Part XIII EIS Mappings		
56	Understanding EIS Mappings	
	EIS Mapping Types	56-1
	EIS Mapping Concepts.....	56-2
	EIS Record Type	56-2
	Indexed Records.....	56-2
	Mapped Records	56-3
	XML Records	56-3
	XPath Support.....	56-3
	xsd:list and xsd:union Support	56-3
	jaxb:class Support.....	56-3
	Typesafe Enumeration Support	56-3
	Composite and Reference EIS Mappings	56-4
	Composite EIS Mappings	56-4
	Reference EIS Mappings	56-4
	EIS Mapping Architecture	56-5
	EIS Direct Mapping	56-5

EIS Composite Direct Collection Mapping.....	56-6
EIS Composite Object Mapping.....	56-7
EIS Composite Collection Mapping.....	56-7
EIS One-to-One Mapping.....	56-8
EIS One-to-One Mappings With Key on Source.....	56-9
EIS One-to-One Mappings With Key on Target.....	56-10
EIS One-to-Many Mapping.....	56-12
EIS One-to-Many Mappings With Key on Source.....	56-13
EIS One-to-Many Mappings With Key on Target.....	56-15
EIS Transformation Mapping.....	56-17
 57 Configuring an EIS Mapping	
Configuring Common EIS Mapping Options	57-1
Configuring Reference Descriptors.....	57-2
Using TopLink Workbench	57-2
Configuring Selection Interaction	57-3
Using TopLink Workbench	57-4
 58 Configuring an EIS Direct Mapping	
EIS Direct Mapping Configuration Overview	58-1
 59 Configuring an EIS Composite Direct Collection Mapping	
EIS Composite Direct Collection Mapping Configuration Overview.....	59-1
 60 Configuring an EIS Composite Object Mapping	
EIS Composite Object Mapping Configuration Overview.....	60-1
 61 Configuring an EIS Composite Collection Mapping	
EIS Composite Collection Mapping Configuration Overview	61-1
 62 Configuring an EIS One-to-One Mapping	
EIS One-to-One Mapping Configuration Overview.....	62-1
Configuring Foreign Key Pairs.....	62-1
Using TopLink Workbench	62-2
 63 Configuring an EIS One-to-Many Mapping	
EIS One-to-Many Mapping Configuration Overview	63-1
Configuring Foreign Key Pairs.....	63-1
Using TopLink Workbench	63-2
Configuring Delete All Interactions.....	63-3
Using TopLink Workbench	63-4
 64 Configuring an EIS Transformation Mapping	
EIS Transformation Mapping Configuration Overview.....	64-1

Part XIV XML Mappings

65 Understanding XML Mappings

XML Mapping Types	65-1
XML Mapping Concepts	65-2
Mapping to Simple and Complex Types.....	65-2
Mapping Order.....	65-3
XPath Support.....	65-3
xsd:list and xsd:union Support	65-3
xs:any and xs:anyType Support.....	65-4
jaxb:class Support.....	65-4
Typesafe Enumeration Support	65-4
Mapping Extensions	65-4
XML Direct Mapping	65-5
Mapping to a Text Node	65-5
Mapping to a Simple Text Node.....	65-5
Mapping to a Text Node in a Simple Sequence.....	65-6
Mapping to a Text Node in a Subelement.....	65-6
Mapping to a Text Node by Position	65-7
Mapping to an Attribute	65-8
Mapping to a Specified Schema Type.....	65-9
Mapping to a List Field With an XML Direct Mapping	65-10
Mapping to a Union Field With an XML Direct Mapping.....	65-10
Mapping to a Union of Lists With an XML Direct Mapping.....	65-12
Mapping to a Union of Unions With an XML Direct Mapping	65-12
Mapping With a Simple Type Translator.....	65-13
XML Composite Direct Collection Mapping	65-14
Mapping to Multiple Text Nodes	65-15
Mapping to a Simple Sequence.....	65-15
Mapping to a Sequence in a Subelement.....	65-15
Mapping to Multiple Attributes.....	65-16
Mapping to a Single Text Node With an XML Composite Direct Collection Mapping	65-17
Mapping to a Single Attribute With an XML Composite Direct Collection Mapping	65-18
Mapping to a List of Unions With an XML Composite Direct Collection Mapping.....	65-18
Mapping to a Union of Lists With an XML Composite Direct Collection Mapping.....	65-19
Specifying the Content Type of a Collection With an XML Composite Direct Collection Mapping	65-20
XML Composite Object Mapping	65-21
Mapping Into the Parent Record.....	65-21
Mapping to an Element.....	65-22
Mapping to Different Elements by Element Name.....	65-23
Mapping to Different Elements by Element Position	65-24
XML Composite Collection Mapping	65-25
XML Any Object Mapping	65-27
XML Any Collection Mapping	65-29
XML Transformation Mapping	65-31

66	Configuring an XML Mapping	
	Configuring Common XML Mapping Options	66-1
	Configuring Reference Descriptor.....	66-2
	Using TopLink Workbench	66-2
	Configuring Maps to Wildcard.....	66-3
	Using TopLink Workbench	66-3
67	Configuring an XML Direct Mapping	
	XML Direct Mapping Configuration Overview.....	67-1
68	Configuring an XML Composite Direct Collection Mapping	
	XML Composite Direct Collection Mapping Configuration Overview.....	68-1
69	Configuring an XML Composite Object Mapping	
	XML Composite Object Mapping Configuration Overview.....	69-1
70	Configuring an XML Composite Collection Mapping	
	XML Composite Collection Mapping Configuration Overview	70-1
71	Configuring an XML Any Object Mapping	
	XML Any Object Mapping Configuration Overview.....	71-1
72	Configuring an XML Any Collection Mapping	
	XML Any Collection Mapping Configuration Overview	72-1
73	Configuring an XML Transformation Mapping	
	XML Transformation Mapping Configuration Overview	73-1
Part XV Using TopLink Overview		
74	Understanding the Persistence Layer	
	Overview of the Persistence Layer.....	74-1
	Sessions	74-1
	Data Access	74-1
	Cache.....	74-2
	Queries and Expressions.....	74-2
	Transactions.....	74-3
Part XVI TopLink Sessions		
75	Understanding TopLink Sessions	
	Session Types.....	75-1
	Session Concepts	75-2

Session Architecture	75-2
Object Cache	75-3
Connection Pools	75-3
Query Mechanism.....	75-4
Java Object Builder	75-4
Session Configuration and the sessions.xml File.....	75-4
Session Customization	75-4
Acquiring a Session at Run Time With the Session Manager.....	75-5
Managing Session Events With the Session Event Manager	75-5
Session Event Manager Events	75-6
Session Event Listeners	75-7
Logging.....	75-7
Log Types	75-8
Log Output.....	75-10
Log Level.....	75-10
Logging SQL.....	75-10
Logging Chained Exceptions	75-10
Viewing TopLink Log Messages From the Application Server Control Console	75-10
Profiler	75-11
TopLink Profiler.....	75-11
Oracle Dynamic Monitoring System (DMS)	75-11
Integrity Checker.....	75-12
Exception Handlers.....	75-12
Registering Descriptors	75-12
Sessions and CMP	75-13
Sessions and Sequencing.....	75-13
Server and Client Sessions	75-13
Three-Tier Architecture Overview	75-14
Advantages of the TopLink Three-Tier Architecture	75-15
Shared Resources	75-15
Providing Read Access.....	75-15
Providing Write Access.....	75-16
Security and User Privileges	75-17
Concurrency.....	75-18
Connection Allocation.....	75-18
Unit of Work Sessions	75-19
Isolated Client Sessions	75-19
Isolated Client Sessions and Oracle Virtual Private Database (VPD)	75-20
VPD With Oracle Database Proxy Authentication	75-21
VPD Without Oracle Database Proxy Authentication	75-21
Isolated Client Session Life Cycle.....	75-22
Isolated Client Session Limitations.....	75-23
Historical Client Sessions.....	75-24
Historical Client Session Limitations	75-25
Session Broker and Client Sessions	75-25
Session Broker Architecture	75-26
Committing a Transaction with a Session Broker	75-27

Committing a Session with a JTA Driver: Two-Phase Commits	75-27
Committing a Session Without a JTA Driver: Two-Stage Commits.....	75-27
Session Broker Session Limitations	75-27
Many-to-Many Join Tables and Direct Collection Tables	75-27
Session Broker Alternatives.....	75-28
Database Linking	75-28
Multiple Sessions	75-28
Database Sessions	75-28
Remote Sessions	75-29
Architectural Overview.....	75-30
Application Layer	75-30
Transport Layer.....	75-30
Server Layer.....	75-31
Remote Session Concepts.....	75-31
Securing Remote Session Access	75-31
Queries.....	75-31
Refreshing	75-31
Indirection	75-32
Cursored Streams.....	75-32
Unit of Work	75-32
Sessions and the Cache	75-32
Server and Database Session Cache	75-32
Isolated Session Cache.....	75-33
Historical Session Cache	75-33
Understanding the Session API	75-33

76 Creating Sessions

Session Creation Overview	76-1
Creating a Sessions Configuration	76-1
Using TopLink Workbench	76-2
Configuring a Sessions Configuration	76-2
Using TopLink Workbench	76-2
Creating a Server Session	76-4
Using TopLink Workbench	76-4
Using Java.....	76-5
Creating Session Broker and Client Sessions	76-6
Using TopLink Workbench	76-6
Using Java.....	76-8
Creating Database Sessions	76-8
Using TopLink Workbench	76-8
Using Java.....	76-10
Creating Remote Sessions	76-10
Using Java.....	76-10
Server	76-11
Client.....	76-11

77 Configuring a Session

Configuring Common Session Options	77-1
Configuring a Primary Mapping Project.....	77-2
Using TopLink Workbench	77-3
Using Java.....	77-3
Configuring a Session Login.....	77-4
Configuring Logging	77-4
Using TopLink Workbench	77-5
Using Java.....	77-7
Using Session Logging API	77-7
Configuring a Session to use java.util.logging Package.....	77-8
Configuring Logging in a CMP Application	77-9
Configuring Multiple Mapping Projects.....	77-9
Using TopLink Workbench	77-9
Using Java.....	77-10
Configuring a Performance Profiler	77-10
Using TopLink Workbench	77-11
Using Java.....	77-12
Configuring an Exception Handler.....	77-12
Using TopLink Workbench	77-12
Using Java.....	77-12
Configuring Customizer Class	77-13
Using TopLink Workbench	77-14
Configuring the Server Platform.....	77-14
Using TopLink Workbench	77-15
Using Java.....	77-17
Configuring Session Event Listeners	77-17
Using TopLink Workbench	77-17
Using Java.....	77-18
Configuring the Integrity Checker	77-18
Using Java.....	77-19
Configuring Connection Policy	77-19
Using TopLink Workbench	77-20
Using Java.....	77-21
Configuring Named Queries at the Session Level	77-21
Using Java.....	77-22

78 Acquiring and Using Sessions at Run Time

Session Acquisition Overview	78-1
Understanding the Session Manager	78-2
Multiple Sessions.....	78-2
Acquiring the Session Manager	78-2
Acquiring a Session from the Session Manager	78-3
Loading a Session from sessions.xml Using Defaults.....	78-3
Loading a Session from sessions.xml with an Alternative Class Loader.....	78-4
Loading a Session from an Alternative Session Configuration File	78-4

Loading a Session Without Logging In.....	78-5
Reloading and Refreshing Session Configuration.....	78-5
Refreshing a Session when the Class Loader Changes.....	78-6
Acquiring a Client Session	78-6
Acquiring an Isolated Client Session	78-7
Acquiring a Historical Client Session.....	78-7
Acquiring a Client Session That Uses Exclusive Connections	78-7
Acquiring a Client Session that Uses Connection Properties	78-8
Acquiring a Client Session that Uses a Named Connection Pool.....	78-8
Acquiring a Client Session that Does Not Use Lazy Connection Allocation	78-9
Logging In to a Session	78-9
Using Session API	78-9
Logging Out of a Session	78-10
Storing Sessions in the Session Manager Instance	78-10
Destroying Sessions in the Session Manager Instance	78-10
 79 Configuring Server Sessions	
Server Session Configuration Overview	79-1
Configuring Internal Connection Pools	79-1
Configuring External Connection Pools	79-2
 80 Configuring Exclusive Isolated Client Sessions for Virtual Private Database	
Exclusive Isolated Client Session Configuration Overview.....	80-1
PostAcquireExclusiveConnection Event Handler	80-1
Using Java.....	80-2
PreReleaseExclusiveConnection Event Handler.....	80-2
Using Java.....	80-2
NoRowsModifiedSessionEvent Event Handler.....	80-3
Using Java.....	80-3
ValidationException Handler	80-3
 81 Configuring Historical Client Sessions	
Historical Client Session Configuration Overview.....	81-1
Configuring Historical Client Sessions Using an Oracle Platform	81-1
Configuring Historical Client Sessions Using a TopLink HistoryPolicy.....	81-1
 82 Configuring Session Broker and Client Sessions	
Session Broker and Client Session Configuration Overview	82-1
Removing, Renaming, or Adding Sessions	82-1
Using TopLink Workbench	82-2
 83 Configuring Database Sessions	
Database Session Configuration Overview.....	83-1
Configuring External Connection Pools	83-1

Part XVII Data Access

84 Understanding Data Access

Data Access Concepts	84-1
Externally Managed Transactional Data Sources	84-1
Data Source Login Types	84-2
DatabaseLogin	84-2
EISLogin	84-3
Data Source Platform Types	84-3
Database Platforms	84-3
EIS Platforms	84-4
Authentication	84-5
Simple JDBC Authentication	84-5
Oracle Database Proxy Authentication	84-5
Auditing	84-6
Connections	84-6
Connection Pools	84-7
Internal Connection Pools	84-7
External Connection Pools	84-8
Default (Write) and Read Connection Pools	84-8
Sequence Connection Pools	84-8
Application-Specific Connection Pools	84-9
Understanding Data Access API	84-9
Login Inheritance Hierarchy	84-10
Platform Inheritance Hierarchy	84-10

85 Configuring a Data Source Login

Configuring Common Data Source Login Options	85-1
Configuring User Name and Password	85-1
Using TopLink Workbench	85-2
Configuring Password Encryption	85-2
Using Java	85-2
Configuring External Connection Pooling	85-2
Using TopLink Workbench	85-3
Configuring Properties	85-4
Using TopLink Workbench	85-4
Using Java	85-5
Configuring a Default Null Value at the Login Level	85-5
Using Java	85-6

86 Configuring a Database Login

Database Login Configuration Overview	86-1
Configuring a Relational Database Platform at the Session Level	86-1
Using TopLink Workbench	86-1
Configuring Database Login Connection Options	86-2
Using TopLink Workbench	86-2

Configuring Sequencing at the Session Level.....	86-4
Using TopLink Workbench	86-4
Using Java.....	86-5
Using the Platform Default Sequence	86-5
Configuring Multiple Sequences	86-6
Configuring Query Sequencing	86-7
Configuring a Table Qualifier	86-8
Using TopLink Workbench	86-8
Configuring JDBC Options	86-9
Using TopLink Workbench	86-9
Using Java.....	86-11
Configuring Advanced Options	86-11
Using TopLink Workbench	86-11
Configuring Oracle Database Proxy Authentication	86-12
Using Java.....	86-13

87 Configuring an EIS Login

EIS Login Configuration Overview.....	87-1
Configuring an EIS Data Source Platform at the Session Level	87-1
Using TopLink Workbench	87-1
Configuring EIS Connection Specification Options at the Session Level.....	87-2
Using TopLink Workbench	87-2

88 Creating an Internal Connection Pool

Internal Connection Pool Creation Overview	88-1
Using TopLink Workbench	88-1

89 Configuring an Internal Connection Pool

Internal Connection Pool Configuration Overview	89-1
Configuring Connection Pool Sizes	89-1
Using TopLink Workbench	89-2
Configuring Properties.....	89-2
Using TopLink Workbench	89-2
Using Java.....	89-3
Configuring a Nontransactional Read Login.....	89-3
Using TopLink Workbench	89-3
Configuring Connection Pool Connection Options.....	89-4
Using TopLink Workbench	89-4
Configuring Exclusive Read Connections.....	89-6
Using TopLink Workbench	89-6

Part XVIII Cache

90 Understanding the Cache

Cache Architecture	90-1
Session Cache.....	90-2

Unit of Work Cache	90-2
Cache Concepts	90-2
Cache Type and Object Identity	90-3
Full Identity Map	90-3
Weak Identity Map	90-3
Soft and Hard Cache Weak Identity Maps.....	90-4
No Identity Map.....	90-4
Guidelines for Configuring the Cache and Identity Maps	90-4
Understanding the Internals of Soft and Hard Cache Weak Identity Map	90-5
Querying and the Cache.....	90-6
Handling Stale Data.....	90-6
Configure a Locking Policy	90-6
Configure the Cache on a Per-Class Basis	90-7
Force a Cache Refresh When Required on a Per-Query Basis.....	90-7
Configure Cache Invalidation	90-7
Configure Cache Coordination.....	90-7
Explicit Query Refreshes.....	90-7
Refresh Policy	90-8
EJB Finders and Refresh Policy	90-8
Cache Invalidation	90-8
Cache Coordination	90-9
Cache Isolation	90-9
Cache Locking and Transaction Isolation.....	90-9
Cache Optimization	90-9
Understanding Cache Coordination	90-10
When to use Cache Coordination.....	90-10
Coordinated Cache Architecture	90-11
Session	90-11
Descriptor	90-11
Unit of Work	90-12
Coordinated Cache Types.....	90-12
JMS Coordinated Cache	90-12
RMI Coordinated Cache	90-12
CORBA Coordinated Cache.....	90-13
Custom Coordinated Cache	90-13
Understanding the Cache API	90-13
Object Identity API	90-13
Cache Refresh API	90-14
Cache Invalidation API	90-14
Cache Coordination API	90-15

91 Configuring a Coordinated Cache

Configuring Common Coordinated Cache Options	91-1
Configuring the Synchronous Change Propagation Mode	91-2
Using TopLink Workbench	91-2
Configuring a Service Channel	91-3
Using TopLink Workbench	91-3

Configuring a Multicast Group Address	91-4
Using TopLink Workbench	91-5
Configuring a Multicast Port	91-5
Using TopLink Workbench	91-6
Configuring a Naming Service Type	91-7
Configuring JNDI Naming Service Information	91-7
Using TopLink Workbench	91-8
Configuring RMI Registry Naming Service Information	91-9
Using TopLink Workbench	91-10
Configuring an Announcement Delay	91-11
Using TopLink Workbench	91-12
Configuring Connection Handling	91-13
Using TopLink Workbench	91-13
Configuring Context Properties	91-14
Using TopLink Workbench	91-14
Configuring a Packet Time-to-Live	91-15
Using TopLink Workbench	91-16

92 Configuring a JMS Coordinated Cache

JMS Coordinated Cache Configuration Overview	92-1
Configuring a Topic Name	92-1
Using TopLink Workbench	92-1
Configuring a Topic Connection Factory Name	92-2
Using TopLink Workbench	92-2
Configuring a Topic Host URL	92-3
Using TopLink Workbench	92-3

93 Configuring an RMI Coordinated Cache

RMI Coordinated Cache Configuration Overview	93-1
---	------

94 Configuring a CORBA Coordinated Cache

CORBA Coordinated Cache Configuration Overview	94-1
---	------

95 Configuring a Custom Coordinated Cache

Custom Coordinated Cache Configuration Overview	95-1
Configuring Transport Class	95-1
Using TopLink Workbench	95-1

Part XIX Queries

96 Understanding TopLink Queries

Query Types	96-1
Query Concepts	96-2
Call.....	96-3
DatabaseQuery	96-3

Data-Level and Object-Level Queries	96-3
Summary Queries	96-3
Descriptor Query Manager	96-3
TopLink Expressions	96-3
Query Keys.....	96-4
Query Languages	96-4
SQL Queries.....	96-4
EJB QL Queries.....	96-5
XML Queries.....	96-5
EIS Interactions.....	96-6
Query by Example	96-6
Building Queries	96-6
Executing Queries	96-7
Handling Query Results	96-8
Collection Query Results.....	96-8
Report Query Results	96-8
Stream and Cursor Query Results	96-8
Session Queries	96-9
Read-Object Session Queries	96-9
Create, Update, and Delete Object Session Queries.....	96-10
Database Queries	96-10
Object-Level Read Query	96-11
ReadObjectQuery	96-11
ReadAllQuery.....	96-11
Partial Object Queries.....	96-11
Join Reading and Object-Level Read Queries.....	96-12
Fetch Groups and Object Level Read Queries	96-12
Data-Level Read Query	96-13
DataReadQuery	96-13
DirectReadQuery	96-13
ValueReadQuery	96-13
Object-Level Modify Query	96-13
WriteObjectQuery.....	96-14
UpdateObjectQuery	96-14
InsertObjectQuery.....	96-14
DeleteObjectQuery.....	96-14
UpdateAllQuery.....	96-14
Object-Level Modify Queries and Privately Owned Parts	96-14
Data-Level Modify Query	96-14
Report Query	96-15
Named Queries	96-15
Call Queries	96-16
SQL Calls	96-16
SQLCall.....	96-17
StoredProcedureCall.....	96-17
StoredFunctionCall	96-17
Oracle Extensions.....	96-17

EJB QL Calls	96-18
Enterprise Information System (EIS) Interactions	96-18
IndexedInteraction	96-19
MappedInteraction	96-19
XMLInteraction	96-19
XQueryInteraction	96-19
QueryStringInteraction	96-19
Redirect Queries	96-19
Historical Queries	96-20
Using an ObjectLevelReadQuery With an AsOfClause	96-21
Using an ObjectLevelReadQuery With Expression Operator asOf	96-21
Using an ObjectLevelReadQuery in a Historical Session.....	96-22
Interface and Inheritance Queries	96-22
Descriptor Query Manager Queries	96-22
Configuring Named Queries	96-22
Configuring Default Query Implementations	96-22
Configuring Additional Join Expressions	96-23
EJB Finders	96-23
Predefined Finders	96-24
Predefined CMP Finders for EJB 2.0	96-24
Predefined CMP Finders for EJB 1.1	96-25
Predefined BMP Finders	96-25
Default Finders	96-26
Call Finders	96-26
DatabaseQuery Finders.....	96-26
Named Query Finders.....	96-27
Primary Key Finders.....	96-27
Expression Finders.....	96-27
EJB QL Finders.....	96-27
SQL Finders.....	96-28
Redirect Finders.....	96-28
The ejbSelect Method	96-28
Queries and the Cache	96-29
Configuring the Cache	96-29
Using In-Memory Queries	96-30
Configuring Cache Usage for In-Memory Queries.....	96-30
Expression Options for In-Memory Queries.....	96-31
Handling Exceptions Resulting From In-Memory Queries.....	96-33
Primary Key Queries and the Cache	96-34
Disabling the Identity Map Cache Update During a Read Query	96-34
Refreshing the Cache	96-35
Object Refresh.....	96-35
Cascading Object Refresh	96-35
Refreshing the Identity Map Cache During a Read Query	96-35
Caching Query Results in the Session Cache	96-36
Caching Query Results	96-36
Internal Query Cache Restrictions.....	96-36

Caching and EJB Finders.....	96-36
Caching Options.....	96-37
Disabling Cache for Returned Finder Results	96-37
Refreshing Finder Results.....	96-38
Understanding the Query API.....	96-38

97 Understanding TopLink Expressions

Understanding the Expression Framework.....	97-1
Expressions Compared to SQL	97-1
Expression Components.....	97-2
Boolean Logic.....	97-3
Database Functions	97-3
Mathematical Functions	97-4
XMLType Functions	97-4
Platform and User-Defined Functions	97-4
Expressions for One-to-One and Aggregate Object Relationships.....	97-5
Expressions for Joining and Complex Relationships	97-5
Understanding Joins.....	97-5
Using TopLink Expression API For Joins.....	97-6
Parameterized Expressions.....	97-7
Expression Method getParameter.....	97-8
Expression Method getField.....	97-9
Query Keys and Expressions.....	97-9
Using Multiple Expressions	97-10
Subselects and Subqueries	97-10
Parallel Expressions	97-11
Data Queries and Expressions	97-11
getField	97-11
getTable	97-12
Creating an Expression.....	97-12
Using TopLink Workbench	97-12
Adding Arguments.....	97-14
Using Java.....	97-15
Creating and Using a User-Defined Function	97-16
Making a User-Defined Function Available to a Specific Platform.....	97-16
Making a User-Defined Function Available to All Platforms	97-17
Using a User-Defined Function	97-17

98 Using Basic Query API

Using Session Queries.....	98-1
Reading Objects With a Session Query	98-1
Reading an Object With a Session Query	98-2
Reading All Objects With a Session Query	98-2
Refreshing an Object with a Session Query	98-2
Creating, Updating, and Deleting Objects With a Session Query	98-3
Writing a Single Object to the Database with a Session Query.....	98-3

Writing All Objects to the Database With a Session Query	98-3
Adding New Objects to the Database with a Session Query	98-4
Modifying Existing Objects in the Database with a Session Query	98-4
Deleting Objects in the Database with a Session Query	98-4
Using DatabaseQuery Queries	98-4
Reading Objects Using a DatabaseQuery	98-4
Basic DatabaseQuery Read Operations	98-5
Reading Objects Using Partial Object Queries	98-5
Reading Objects Using Report Queries	98-6
Reading Objects Using Query By Example	98-6
Specifying Read Ordering	98-8
Specifying a Collection Class	98-9
Specifying the Maximum Rows Returned	98-9
Configuring Query Timeout at the Query Level	98-10
Using Batch Reading	98-10
Using Join Reading	98-11
Creating, Updating, and Deleting Objects with a DatabaseQuery	98-12
Write Query Overview	98-12
UpdateAll Queries	98-13
Noncascading Write Queries	98-13
Disabling the Identity Map Cache During a Write Query	98-14
Reading Data With a DatabaseQuery	98-14
Using a DataReadQuery	98-15
Using a DirectReadQuery	98-15
Using a ValueReadQuery	98-15
Updating Data With a DatabaseQuery	98-15
Specifying a Custom SQL String in a DatabaseQuery	98-16
Specifying a Custom EJB QL String in a DatabaseQuery	98-16
Using Parameterized SQL and Statement Caching in a DatabaseQuery	98-17
Using Named Queries	98-17
Using SQL Calls	98-18
Using an SQLCall	98-18
Specifying a SQLCall Input Parameter	98-19
Specifying a SQLCall Output Parameter	98-19
Specifying a SQLCall Input / Output Parameter	98-20
Using a StoredProcedureCall	98-20
Specifying an Input Parameter	98-21
Specifying an Output Parameter	98-21
Specifying an Input / Output Parameter	98-22
Using an Output Parameter Event	98-22
Using a StoredFunctionCall	98-22
Using EJB QL Calls	98-23
Using EIS Interactions	98-24
Handling Exceptions	98-24
Handling Collection Query Results	98-25
Handling Report Query Results	98-25

99 Using Advanced Query API

Using Redirect Queries	99-1
Creating a Redirect Query	99-1
Using Historical Queries	99-2
Using Queries With Fetch Groups	99-2
Configuring Default Fetch Group Behavior	99-3
Querying With a Static Fetch Group	99-3
Querying With a Dynamic Fetch Group	99-4
Querying on Interfaces	99-4
Querying on an Inheritance Hierarchy	99-4
Appending Additional Join Expressions	99-4
Using Java	99-5
Using Queries on Variable One-to-One Mappings	99-5
Using Oracle Database Features	99-5
Oracle Hints	99-6
Hierarchical Queries	99-6
startWith Parameter	99-7
connectBy Parameter	99-7
orderSibling Parameter	99-7
Using EJB Finders	99-7
Creating a Finder	99-8
ejb-jar.xml Finder Options	99-9
Using DatabaseQuery Finders	99-10
Using Named Query Finders	99-10
Using Primary Key Finders	99-11
Using EJB QL Finders	99-11
Using SQL Finders	99-12
Using Redirect Finders	99-12
Using the ejbSelect Method	99-15
Handling Cursor and Stream Query Results	99-16
Cursors and Java Iterators	99-16
Traversing Data With Scrollable Cursors	99-16
Java Streams	99-17
Cursored Stream Support	99-17
Optimizing Streams	99-18
Using Cursors and Streams With EJB Finders	99-19
Building the Query	99-19
Executing the Finder From the Client in EJB 1.1	99-19
Executing the Finder From the Client in EJB 2.0	99-20
Using Queries and the Cache	99-21
Caching Results in a ReadQuery	99-21
Configuring Cache Expiration at the Query Level	99-22

Part XX Transactions

100 Understanding TopLink Transactions

Unit of Work Architecture	100-1
Unit of Work Transaction Context.....	100-2
Unit of Work Transaction Demarcation.....	100-2
JTA Controlled Transactions	100-3
OTS Controlled Transactions	100-3
CMP Controlled Transactions	100-3
Unit of Work Transaction Isolation	100-4
Unit of Work Concepts	100-4
Unit of Work Benefits	100-4
Unit of Work Life Cycle	100-5
Unit of Work and Change Policy	100-6
Deferred Change Detection Policy	100-7
Object-Level Change Tracking Policy	100-7
Attribute Change Tracking Policy	100-8
Change Policy Mapping Support	100-8
Clones and the Unit of Work	100-9
Nested and Parallel Units of Work	100-9
Nested Unit of Work	100-10
Parallel Unit of Work	100-10
Commit and Rollback Transactions	100-10
Commit Transactions	100-10
Rollback Transactions	100-11
Primary Keys	100-11
Unit of Work Optimization	100-11
Understanding the Unit of Work API	100-11
Unit of Work as Session	100-12
Reading and Querying Objects with the Unit of Work	100-12
Locking and the Unit of Work	100-12
Example Model Object and Schema	100-12

101 Using Basic Unit of Work API

Acquiring a Unit of Work	101-1
Creating an Object	101-2
Modifying an Object	101-2
Associating a New Target to an Existing Source Object	101-3
Associating Without Reference to the Cache Object	101-3
Associating With Reference to the Cache Object	101-4
Associating a New Source to an Existing Target Object	101-6
Associating an Existing Source to an Existing Target Object	101-7
Deleting Objects	101-7
Using privateOwnedRelationship	101-8
Explicitly Deleting from the Database	101-9
Understanding the Order in Which Objects Are Deleted	101-9

102 Using Advanced Unit of Work API

Registering and Unregistering Objects	102-1
Creating and Registering an Object in One Step	102-2
Using registerNewObject	102-2
Registering a New Object with registerNewObject	102-2
Associating New Objects With One Another	102-3
Using registerAllObjects	102-4
Using Registration and Existence Checking	102-5
Check Database	102-5
Assume Existence	102-5
Assume Nonexistence	102-5
Working with Aggregates.....	102-6
Unregistering Working Clones	102-6
Declaring Read-Only Classes	102-6
Configuring Read-Only Classes for a Single Unit of Work	102-6
Configuring Default Read-Only Classes	102-7
Read-Only Descriptors	102-7
Writing Changes Before Commit Time	102-7
Using Conforming Queries and Descriptors	102-8
Guidelines for Using Conforming	102-8
Use Only In-Memory Queries that Support Conforming.....	102-8
Consider How Conforming Affects Database Results	102-9
If You Want an Object in the Conformed Results, Register it in the Unit of Work.....	102-9
Using Conforming Queries.....	102-10
Using Conforming Descriptors	102-11
Conforming Query Alternatives	102-11
Using Unit of Work Method writeChanges Instead of Conforming.....	102-11
Using Unit of Work Properties Instead of Conforming	102-12
Merging Changes in Working Copy Clones	102-12
Resuming a Unit of Work After Commit	102-14
Reverting a Unit of Work	102-14
Using a Nested or Parallel Unit of Work	102-14
Parallel Unit of Work.....	102-15
Nested Unit of Work.....	102-15
Using a Unit of Work With Custom SQL	102-15
Controlling the Order of Delete Operations	102-16
Using the Unit of Work setShouldPerformDeletesFirst Method	102-16
Using the Descriptor addConstraintDependencies Method	102-16
Using deleteAllObjects Without addConstraintDependencies	102-16
Using deleteAllObjects with addConstraintDependencies	102-17
Using Optimistic Read Locking With forceUpdateToVersionField	102-17
Forcing a Check of the Optimistic Read Lock.....	102-17
Forcing a Version Field Update	102-18
Disabling forceUpdateToVersionField	102-20
Implementing User and Date Auditing with the Unit of Work	102-20
Integrating the Unit of Work With an External Transaction Service	102-20
Acquiring a Unit of Work With an External Transaction Service.....	102-21

Using a Unit of Work When an External Transaction Exists	102-21
Using a Unit of Work When No External Transaction Exists.....	102-22
Using the Unit of Work to Handle External Transaction Timeouts and Exceptions	102-23
External Transaction Commit Timeouts.....	102-23
External Transaction Commit Exceptions	102-23
Integrating the Unit of Work With CMP	102-24
CMP Transaction Attribute.....	102-24
Local Transactions.....	102-25
Nondeferred Changes	102-25
Database Transaction Isolation Levels	102-25
General Factors Affecting Transaction Isolation Level.....	102-26
External Applications	102-26
TopLink Coordinated Cache	102-26
DatabaseLogin Method setTransactionIsolation	102-27
Reading Through the Write Connection	102-27
Managing Cache Access.....	102-28
CMP and External Transactions	102-29
Read Uncommitted Level	102-29
Read Committed Level.....	102-30
Repeatable Read Levels.....	102-30
Serializable Read Levels	102-30
Troubleshooting a Unit of Work	102-30
Avoiding the Use of Post-Commit Clones	102-30
Determining Whether or Not an Object Is the Cache Object.....	102-31
Dumping the Contents of a Unit of Work	102-32
Handling Exceptions	102-33
Exceptions at Commit Time	102-33
Exceptions During Conforming.....	102-33
Validating a Unit of Work.....	102-34
Validating the Unit of Work Before Commit Time.....	102-34
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-2
.....	103-2

.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2

Index

Preface

Oracle TopLink Developer's Guide explains how to use Oracle TopLink to design, implement, deploy, and optimize an advanced persistence or object-to-XML transformation layer for a wide range of Java 2 Enterprise Edition (J2EE) and Java applications. It describes TopLink mapping metadata, sessions, data access, queries, transactions (both with and without an external transaction controller), and cache.

It describes how to use TopLink application development tools and how to integrate TopLink with a variety of J2EE containers. It also introduces the concepts with which you should be familiar to get the most out of TopLink.

Audience

Oracle TopLink Developer's Guide is intended for application developers creating applications that use TopLink to manage persistence.

This document assumes that you are familiar with the concepts of object-oriented programming, the Enterprise JavaBeans (EJB) specification, and your own particular Java development environment.

The document also assumes that you are familiar with your particular operating system (such as Windows, UNIX, or other). The general operation of any operating system is described in the user documentation for that system, and is not repeated in this manual.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documentation

For more information, see the following documents:

- *Oracle TopLink Release Notes*
- Oracle Application Server Release Notes
- *Oracle TopLink Getting Started Guide*
- *Oracle TopLink API Reference*

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.

Part XI

Relational Mappings

A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model.

This part contains the following chapters:

- [Chapter 36, "Understanding Relational Mappings"](#)
This chapter describes each of the different TopLink relational mapping types and important relational mapping concepts.
- [Chapter 37, "Configuring a Relational Mapping"](#)
This chapter explains how to configure TopLink relational mapping options common to two or more relational mapping types.
- [Chapter 38, "Configuring a Relational Direct-to-Field Mapping"](#)
This chapter explains how to configure a direct to field relational database mapping.
- [Chapter 39, "Configuring a Relational Direct-to-XMLType Mapping"](#)
This chapter explains how to configure a direct mapping to an Oracle XDB XML type field.
- [Chapter 40, "Configuring a Relational One-to-One Mapping"](#)
This chapter explains how to configure a one-to-one relational database mapping.
- [Chapter 41, "Configuring a Relational Variable One-to-One Mapping"](#)
This chapter explains how to configure a variable one-to-one relational database mapping.
- [Chapter 42, "Configuring a Relational One-to-Many Mapping"](#)
This chapter explains how to configure a one-to-many relational database mapping.
- [Chapter 43, "Configuring a Relational Many-to-Many Mapping"](#)
This chapter explains how to configure a many-to-many relational database mapping.
- [Chapter 44, "Configuring a Relational Aggregate Collection Mapping"](#)
This chapter explains how to configure an aggregate collection relational database mapping.
- [Chapter 45, "Configuring a Relational Direct Collection Mapping"](#)

This chapter explains how to configure a direct collection relational database mapping.

- [Chapter 46, "Configuring a Relational Direct Map Mapping"](#)

This chapter explains how to configure a direct map relational database mapping.

- [Chapter 47, "Configuring a Relational Aggregate Object Mapping"](#)

This chapter explains how to configure an aggregate object-relational database mapping.

- [Chapter 48, "Configuring a Relational Transformation Mapping"](#)

This chapter explains how to configure a transformation relational database mapping.

Understanding Relational Mappings

A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model.

Relational mappings transform object data members to relational database fields. Use them to map simple data types including primitives (such as `int`), JDK classes (such as `String`), and large object (LOB) values. You can also use them to transform object data members that reference other domain objects by way of association where data source representations require object identity maintenance (such as sequencing and back references) and possess various types of multiplicity and navigability. The appropriate mapping class is chosen primarily by the cardinality of the relationship.

Do not confuse relational mappings with object-relational mappings (see ["Understanding Object-Relational Mappings"](#) on page 49-1). An object-relational mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational databases such as Oracle9i Database Server. Object-relational mappings let you map an object model into an object-relational data model. In general, you can use relational mappings with any supported relational database. You can only use object-relational mappings with specialized object-relational databases optimized to support object-relational data source representations.

This chapter describes:

- [Relational Mapping Types](#)
- [Relational Mapping Concepts](#)

Relational Mapping Types

TopLink supports the relational mappings listed in [Table 36-1](#).

Table 36-1 *TopLink Relational Mapping Types*

Type	Description	Type	TopLink Workbench	Java
"Direct-to-Field Mapping" on page 36-4	Map a Java attribute directly to a database field.	Basic	✓	✓
"Direct to XMLType Mapping" on page 36-4	Map Java attributes to an XMLType column in an Oracle Database (introduced in version 9.2.0.1).	Advanced	✓	✓
"One-to-One Mapping" on page 36-5	Map a reference to another persistent Java object to the database.	Basic	✓	✓

Table 36–1 (Cont.) TopLink Relational Mapping Types

Type	Description	Type	TopLink Workbench	Java
"Variable One-to-One Mapping" on page 36-6	Map a reference to an interface to the database.	Basic	✓	✓
"One-to-Many Mapping" on page 36-7	Map Java collections of persistent objects to the database.	Advanced	✓	✓
"Many-to-Many Mapping" on page 36-8	Use an association table to map Java collections of persistent objects to the database.	Advanced	✓	✓
"Aggregate Collection Mapping" on page 36-10	Map Java collections of persistent objects to the database.	Basic		✓
"Direct Collection Mapping" on page 36-11	Map Java collections of objects that do not have descriptors.	Basic	✓	✓
"Direct Map Mapping" on page 36-12	Direct map mappings store instances that implement <code>java.util.Map</code> .	Basic	✓	✓
"Aggregate Object Mapping" on page 36-12	Create strict one-to-one mappings that require both objects to exist in the same database row.	Basic	✓	
"Transformation Mapping" on page 36-15	Create custom mappings where one or more fields can be used to create the object to be stored in the attribute.	Basic	✓	✓

Relational Mapping Concepts

This section introduces direct mapping concepts unique to TopLink, including the following:

- [Directionality](#)
- [Converters and Transformers](#)
- [Relational Mappings and EJB](#)

Directionality

The direction of a relationship may be either unidirectional or bidirectional. In a unidirectional relationship, only one entity bean has a relationship field that refers to the other. All TopLink relational mappings are unidirectional, from the class being described (the *source* class) to the class with which it is associated (the *target* class). The target class does not have a reference to the source class in a unidirectional relationship.

In a bidirectional relationship, each entity bean has a relationship field that refers to the other bean. Through the relationship field, an entity bean's code can access its related object. To implement a bidirectional relationship (classes that reference each other), use two unidirectional mappings with the sources and targets reversed.

Note: Maintenance of bidirectional relationships presents a number of technical challenges. For more information, see the following:

- [Maintaining Bidirectional Relationships](#)
 - [Configuring Bidirectional Relationship](#)
 - [Indirection](#)
-

Converters and Transformers

You can store object attributes directly in a database table as follows:

- [Using a Direct Mapping](#)
- [Using a Converter Mapping](#)
- [Using a Transformation Mapping](#)

Using a Direct Mapping

If the attribute type is comparable to a database type, the information can be stored directly simply by using a direct-to-field mapping (see ["Direct-to-Field Mapping"](#) on page 36-4).

Using a Converter Mapping

If the attribute type is comparable to a database type but requires conversion, the information can be stored directly by using a direct-to-field mapping (see ["Direct-to-Field Mapping"](#) on page 36-4) and an appropriate `Converter` instance.

In the previous release, TopLink provided subclasses of `DirectToFieldMapping` for object type direct mappings, serialized object direct mappings, and type conversion direct mappings. In this release, these subclasses are deprecated. In their place, Oracle recommends that you use the `DirectToFieldMapping` method `setConverter` and the corresponding `Converter` instance. [Table 36-2](#) summarizes these changes.

Table 36-2 *Using a Converter for Direct to Field Mappings*

Deprecated <code>DirectToFieldMapping</code> subclass...	Replaced by Converter instance...
<code>ObjectTypeMapping</code>	<code>ObjectTypeConverter</code> (see "Object Type Converter" on page 33-12)
<code>SerializedObjectMapping</code>	<code>SerializedObjectConverter</code> (see "Serialized Object Converter" on page 33-10)
<code>TypeConversionMapping</code>	<code>TypeConversionConverter</code> (see "Type Conversion Converter" on page 33-11)

If the application's objects contain attributes that cannot be represented as direct-to-field with an existing converter, use a direct-to-field mapping with a custom converter.

Using a Transformation Mapping

If there is no database primitive type that is logically comparable to the attribute's type, or, if an attribute requires data from multiple fields, it must be transformed on its way to and from the database.

In this case, use a transformation mapping (see ["Transformation Mapping"](#) on page 36-15).

Relational Mappings and EJB

Use direct mappings to map the (non-CMR) CMF attributes of a bean.

- In EJB 2.0 CMP projects, the bean class does not define real variables in the class – only abstract getter and setter methods. To map the bean's attributes, you must import `ejb-jar.xml` file into TopLink Workbench (see ["Configuring Persistence Type"](#) on page 22-5).

- In EJB 1.1 CMP and BMP projects, the bean's CMF attributes are mapped as normal.

You can map entity bean attributes using direct mappings without any special considerations.

Note: When you work with EJB, do not map the entity context attribute (type `javax.ejb.EntityContext`).

There are some special considerations when using one-to-one mappings (see ["One-to-One Mappings and EJB"](#) on page 36-6), one-to-many mappings (see ["One-to-Many Mappings and EJB"](#) on page 36-8), and many-to-many mappings (see ["Many-to-Many Mappings and EJB"](#) on page 36-9).

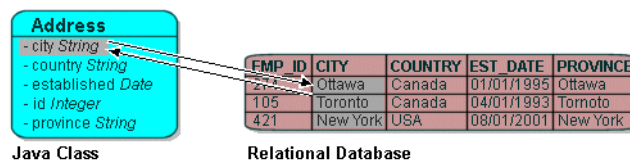
Direct-to-Field Mapping

Use direct-to-field mappings to map primitive object attributes, or non persistent regular objects, such as the JDK classes. For example, use a direct-to-field mapping to store a `String` attribute in a `VARCHAR` field.

Example 36-1 Direct-to-Field Mapping Example

[Figure 36-1](#) illustrates a direct-to-field mapping between the Java attribute `city` and the relational database column `CITY`. Similarly, direct-to-field mappings could be defined from `country` to `COUNTRY`, `id` to `ADDRESS_ID`, `established` to `EST_DATE`, and `province` to `PROVINCE`.

Figure 36-1 Direct-to-Field Mapping



You can use a direct-to-field mapping with any of the following `Converter` instances:

- see ["Object Type Converter"](#) on page 33-12
- see ["Serialized Object Converter"](#) on page 33-10
- see ["Type Conversion Converter"](#) on page 33-11

You can use a direct-to-field mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

See [Chapter 38, "Configuring a Relational Direct-to-Field Mapping"](#) for more information.

Direct to XMLType Mapping

Using a direct-to-XMLType mapping, you can map XML data in the form of a `String` or an `org.w3c.dom.Document` object to an `XMLType` column in an Oracle Database (introduced in version 9.2.0.1).

If you plan to use direct to XML type mappings in TopLink Workbench and the TopLink runtime, you must include the Oracle Database `xdb.jar` file in the TopLink

Workbench classpath (see ["Configuring the TopLink Workbench Environment"](#) on page 4-2).

The TopLink query framework provides a number of expression operators you can use to create queries based on the content of that XML data (see ["XMLType Functions"](#) on page 97-4).

See [Chapter 39, "Configuring a Relational Direct-to-XMLType Mapping"](#) for more information.

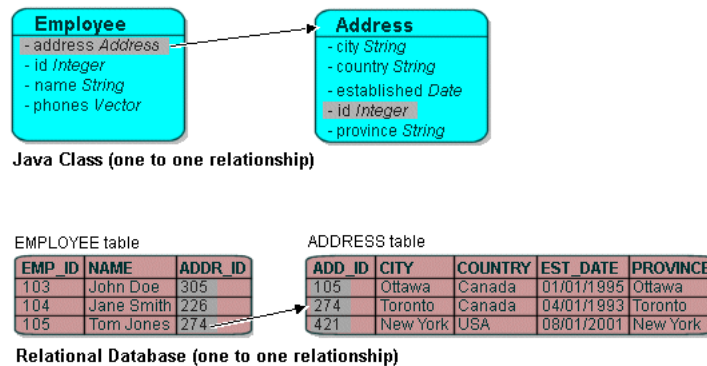
One-to-One Mapping

One-to-one mappings represent simple pointer references between two Java objects. In Java, a single pointer stored in an attribute represents the mapping between the source and target objects. Relational database tables implement these mappings using foreign keys.

[Figure 36-2](#) illustrates a one-to-one relationship from the `address` attribute of an `Employee` object to an `Address` object. To store this relationship in the database, create a one-to-one mapping between the `address` attribute and the `Address` class. This mapping stores the `id` of the `Address` instance in the `EMPLOYEE` table when the `Employee` instance is written. It also links the `Employee` instance to the `Address` instance when the `Employee` is read from the database. Because an `Address` does not have any references to the `Employee`, it does not have to provide a mapping to `Employee`.

For one-to-one mappings, the source table normally contains a foreign key reference to a record in the target table. In [Figure 36-2](#), the `ADDR_ID` field of the `EMPLOYEE` table is a foreign key.

Figure 36-2 One-to-One Mappings



You can also implement a one-to-one mapping where the target table contains a foreign key reference to the source table. In [Figure 36-2](#), the database design would change such that the `ADDRESS` row would contain the `EMP_ID` to identify the `Employee` to which it belonged. In this case, the target must also have a relationship mapping to the source.

The update, insert and delete operations, which are normally done for the target before the source for privately owned one-to-one relationships, are performed in the opposite order when the target owns the foreign key. Target foreign keys normally occur in bidirectional one-to-one mappings (see ["Directionality"](#) on page 36-2), because one side has a foreign key and the other shares the same foreign key in the other's table.

Target foreign keys can also occur when large cascaded composite primary keys exist (that is, one object's primary key is composed of the primary key of many other objects). In this case it is possible to have a one-to-one mapping that contains both foreign keys and target foreign keys.

In a foreign key, TopLink automatically updates the foreign key value in the object's row. In a target foreign key, it does not. In TopLink, use the **Target Foreign Key** option when a target foreign key relationship is defined.

When mapping a relationship, you must understand these differences between a foreign key and a target foreign key, to ensure that the relationship is defined correctly.

In a bidirectional relationship where the two classes in the relationship reference each other, only one of the mappings should have a foreign key. The other mapping should have a target foreign key. If one of the mappings in a bidirectional relationship is a one-to-many mapping, see ["Configuring a Relational Variable One-to-One Mapping"](#) on page 41-1 for details.

You can use a one-to-one mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

See ["Configuring a Relational One-to-One Mapping"](#) on page 40-1 for more information.

One-to-One Mappings and EJB

To maintain EJB compliance, the object attribute that points to the target of the relationship must be the remote (EJB 1.1) or local (EJB 2.0) interface type—not the bean class.

TopLink provides variations on one-to-one mappings that lets you define complex relationships when the target of the relationship is a dependent Java object. For example, *variable one-to-one mappings* enable you to specify variable target objects in the relationship. These variations are not available for entity beans, but are valid for dependent Java objects.

For more information, see the [Chapter 41, "Configuring a Relational Variable One-to-One Mapping"](#).

Variable One-to-One Mapping

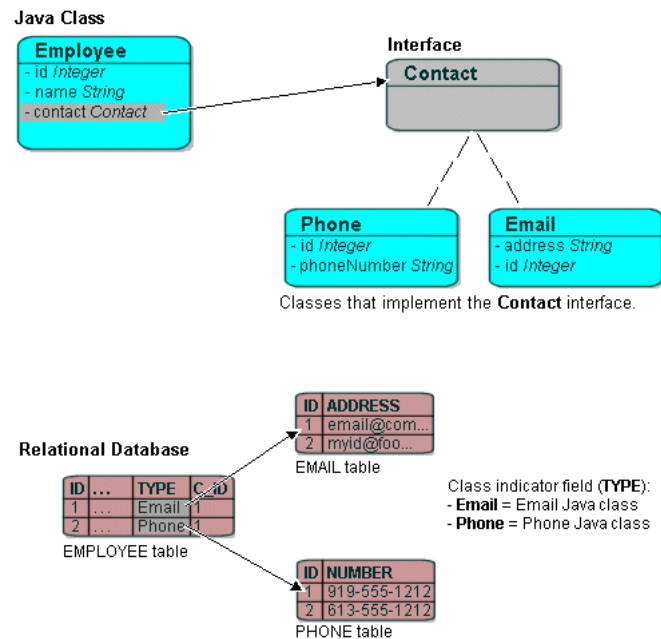
Variable class relationships are similar to polymorphic relationships, except that in this case the target classes are not related through inheritance (and thus not good candidates for an abstract table), but through an interface.

To define variable class relationships in TopLink Workbench, use the variable one-to-one mapping selection, but choose the interface as the reference class. This makes the mapping a variable one-to-one. When defining mappings in Java code, use the `VariableOneToOneMapping` class.

TopLink supports variable relationships only in one-to-one mappings. It handles this relationship in two ways:

- Through the class indicator field (see ["Configuring Class Indicator"](#) on page 41-1).
- Through unique primary key values among target classes implementing the interface (see ["Configuring Unique Primary Key"](#) on page 41-3).

Figure 36-3 Variable One-to-One Mappings with Class Indicator



See ["Configuring a Relational Variable One-to-One Mapping"](#) on page 41-1 for more information.

One-to-Many Mapping

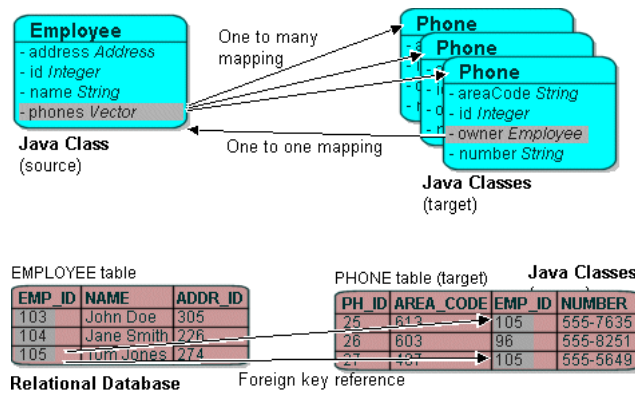
One-to-many mappings are used to represent the relationship between a single source object and a collection of target objects. They are a good example of something that is simple to implement in Java using a **Vector** (or other collection types) of target objects, but difficult to implement using relational databases.

In a Java **Vector**, the owner references its parts. In a relational database, the parts reference their owner. Relational databases use this implementation to make querying more efficient.

Note: See ["Configuring Container Policy"](#) on page 35-26 for information on using collection classes other than **Vector** with one-to-many mappings.

The purpose of creating this one-to-one mapping in the target is so that the foreign key information can be written when the target object is saved. Alternatives to the one-to-one mapping back reference include the following:

- Use a direct-to-field mapping to map the foreign key and maintain its value in the application. Here the object model does not require a back reference, but the data model still requires a foreign key in the target table.
- Use a many-to-many mapping to implement a logical one-to-many. This has the advantage of not requiring a back reference in the object model and not requiring a foreign key in the data model. In this model the many-to-many relation table stores the collection. It is possible to put a constraint on the join table to enforce that the relation is a logical one-to-many relationship.

Figure 36-4 One-to-Many Relationships

You can use a many-to-many mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

See ["Configuring a Relational One-to-Many Mapping"](#) on page 42-1 for more information.

One-to-Many Mappings and EJB

Use one-to-many mappings for relationships between entity beans or between an entity bean and a collection of privately owned regular Java objects. When you create one-to-many mappings, also create a one-to-one mapping from the target objects back to the source. The object attribute that contains a pointer to the bean must be the remote (EJB 1.1) or local (EJB 2.0) interface type—not the bean class.

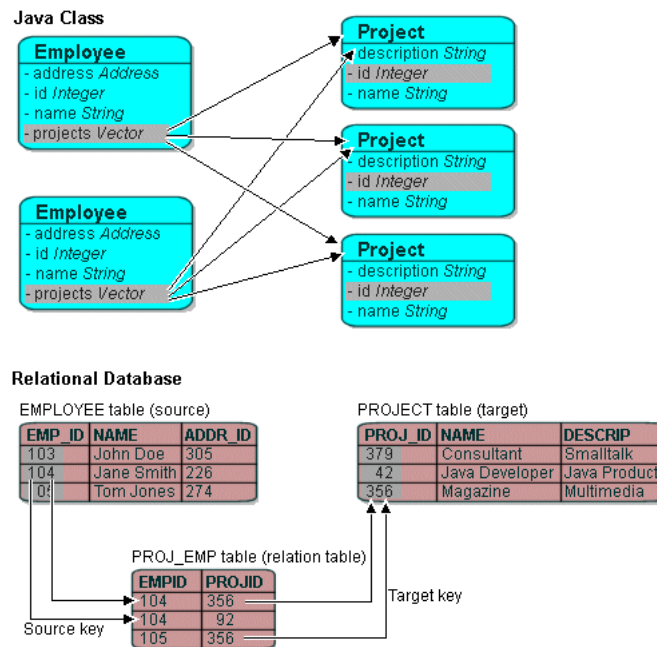
TopLink automatically maintains back-pointers when you create or update bidirectional relationships between beans.

For more information, see ["Configuring Bidirectional Relationship"](#) on page 35-34.

Many-to-Many Mapping

Many-to-many mappings represent the relationships between a collection of source objects and a collection of target objects. They require the creation of an intermediate table for managing the associations between the source and target records.

[Figure 36-5](#) illustrates a many-to-many mapping in Java and in relational database tables.

Figure 36-5 Many-to-many Relationships

Many-to-many mappings are implemented using a relation table. This table contains columns for the primary keys of the source and target tables. Composite primary keys require a column for each field of the composite key. The intermediate table must be created in the database before using the many-to-many mapping.

The target class does not have to implement any behavior for the many-to-many mappings. If the target class also creates a many-to-many mapping back to its source, then it can use the same relation table, but one of the mappings must be set to read-only. If both mappings write to the table, they can cause collisions.

Note: See ["Configuring Container Policy"](#) on page 35-26 for information on using `Collection` classes other than `Vector` with one-to-many mappings.

Indirection is enabled by default in a many-to-many mapping, which requires that the attribute have the `ValueHolderInterface` type or transparent collections.

You can use a many-to-many mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

See ["Configuring a Relational Many-to-Many Mapping"](#) on page 43-1 for more information.

Many-to-Many Mappings and EJB

When you use CMP, many-to-many mappings are valid only between entity beans, and cannot be privately owned. The only exception is when a many-to-many mapping is used to implement a logical one-to-many mapping with a relation table.

TopLink automatically maintains back-pointers when you create or update bidirectional relationships.

For more information, see ["Configuring Bidirectional Relationship"](#) on page 35-34.

Aggregate Collection Mapping

Aggregate collection mappings are used to represent the aggregate relationship between a single-source object and a collection of target objects. Unlike the TopLink one-to-many mappings, in which there should be a one-to-one back reference mapping from the target objects to the source object, there is no back reference required for the aggregate collection mappings, because the foreign key relationship is resolved by the aggregation.

Note: To use aggregate collections with TopLink Workbench, you must use an amendment method (see "[Configuring Amendment Methods](#)" on page 28-78), or manually edit the project source to add the mapping.

Although aggregate collection mappings are similar to one-to-many mappings, they are not replacements for one-to-many mappings. Use aggregate collections only in situations where the target collections are of a reasonable size and if having a one-to-one back mapping is difficult.

Because one-to-many relationships offer better performance and are more robust and scalable, consider using a one-to-many relationship rather than an aggregate collection. In addition, aggregate collections are privately owned by the source of the relationship and must not be shared or referenced by other objects.

Aggregate collections are privately owned by the source of the relationship and should not be shared or referenced by other objects.

This section describes the following:

- [Aggregate Collection Mappings and Inheritance](#)
- [Aggregate Collection Mappings and EJB](#)
- [Implementing Aggregate Collection Mappings](#)

See "[Configuring a Relational Aggregate Collection Mapping](#)" on page 44-1 for more information.

Aggregate Collection Mappings and Inheritance

Aggregate collection descriptors can use inheritance. You must also declare subclasses as aggregate collection. The subclasses can have their own mapped tables, or share the table with their parent class. See "[Descriptors and Inheritance](#)" on page 26-3 for more information on inheritance.

In a Java `Vector`, the owner references its parts. In a relational database, the parts reference their owners. Relational databases use this implementation to make querying more efficient.

Aggregate collection mappings require a target table for the target objects.

To implement an aggregate collection mapping, the following must take place:

- The descriptor of the target class must declare itself as an aggregate collection object. Unlike the aggregate object mapping, in which the target descriptor does not have a specific table to associate with, there must be a target table for the target object.
- The descriptor of the source class must add an aggregate collection mapping that specifies the target class.

Aggregate Collection Mappings and EJB

You can use aggregate collection mappings with entity beans if the source of the relationship is an entity bean or Java object, and the mapping targets are regular Java objects. Entity beans cannot be the target of an aggregate object mapping.

Implementing Aggregate Collection Mappings

To implement an aggregate collection mapping, the following must take place:

- The descriptor of the target class must declare itself to be an aggregate collection object. Unlike the aggregate object mapping, in which the target descriptor does not have a specific table to associate with, there must be a target table for the target object.
- The descriptor of the source class must add an aggregate collection mapping that specifies the target class.

Direct Collection Mapping

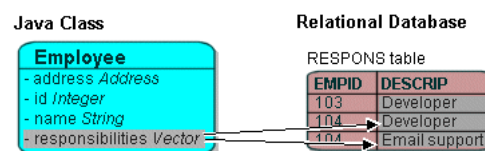
Direct collection mappings store collections of Java objects that are not TopLink-enabled. The object type stored in the direct collection is typically a Java type, such as `String`.

It is also possible to use direct collection mappings to map a collection of non-`String` objects. For example, it is possible to have an attribute that contains a collection of `Integer` or `Date` instances. The instances stored in the collection can be any type supported by the database and has a corresponding wrapper class in Java.

Support for primitive data types such as `int` is not provided, because Java `Vectors` hold only objects.

Figure 36–6 illustrates how a direct collection is stored in a separate table with two fields. The first field is the reference key field, which contains a reference to the primary key of the instance owning the collection. The second field contains an object in the collection and is called the direct field. There is one record in the table for each object in the collection.

Figure 36–6 Direct Collection Mappings



Note: The `responsibilities` attribute is of type `Vector`. When using JDK 1.2, it is possible to use a `Collection` interface (or any class that implements the `Collection` interface) for declaring the collection attribute. See ["Configuring Container Policy"](#) on page 35-26 for details.

Maps are not supported for direct collection because there is no key value.

You can use a direct collection mapping with any of the following `Converter` instances:

- see ["Object Type Converter"](#) on page 33-12
- see ["Serialized Object Converter"](#) on page 33-10
- see ["Type Conversion Converter"](#) on page 33-11

You can use a direct collection mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

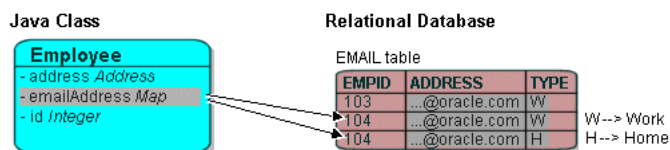
See ["Configuring a Relational Direct Collection Mapping"](#) on page 45-1 for more information.

Direct Map Mapping

Direct map mappings store instances that implement `java.util.Map`. Unlike one-to-many or many-to-many mappings, the keys and values of the map in this type of mapping are Java objects that do not have descriptors. The object type stored in the key and the value of direct map are Java primitive wrapper types such as `String` objects.

[Figure 36-7](#) illustrates how a direct map is stored in a separate table with three fields. The first field (EMPID) is the reference key field, which contains a reference to the primary key of the instance owning the collection. The second field (ADDRESS) contains an object in the collection and is called the direct value field. The third field (TYPE) contains the direct key field. In this example, the direct map uses a object type converter for the direct key field, converting the single character **W** in the database to the full string **Work** in the object (and **H** to **Home**).

Figure 36-7 Direct Map Mappings



You can use a direct collection mapping with any of the following `Converter` instances:

- see ["Object Type Converter"](#) on page 33-12
- see ["Serialized Object Converter"](#) on page 33-10
- see ["Type Conversion Converter"](#) on page 33-11

You can use a direct map mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

See ["Configuring a Relational Direct Map Mapping"](#) on page 46-1 for more information.

Aggregate Object Mapping

Two objects—a source (parent or owning) object and a target (child or owned) object—are related by aggregation if there is a strict one-to-one relationship between them and all the attributes of the target object can be retrieved from the same table(s) as the source object. This means that if the source object exists, then the target object must also exist and if the source object is destroyed, then the target object is also destroyed.

An aggregate mapping allows you to associate data members in the target object with fields in the source object's underlying database tables.

You configure the aggregate mapping in the source object's descriptor. However, before doing so, you must designate the target object's descriptor as an aggregate (see ["Configuring a Relational Descriptor as a Class or Aggregate Type"](#) on page 29-11).

Aggregate objects are privately owned and should not be shared or referenced by other objects.

You cannot configure one-to-one, one-to-many, or many-to-many mappings from a nonaggregate object to an aggregate target object.

You can configure such mappings from an aggregate target object to another nonaggregate object. If you configure a one-to-many mapping from an aggregate target object to another nonaggregate object, you must configure a one-to-one mapping from the other object back to the source object that owns the aggregate (instead of to the aggregate target object itself). This is because the source object contains the table and primary key information of the aggregate target.

You can configure inheritance for a descriptor designated as an aggregate (see ["Descriptors and Inheritance"](#) on page 26-3), however, in this case, *all* the descriptors in the inheritance tree must be aggregates. Aggregate and class descriptors cannot exist in the same inheritance tree.

This section describes the following:

- [Aggregate Object Mappings with a Single Source Object](#)
- [Aggregate Object Mappings With Multiple Source Objects](#)
- [Implementing an Aggregate Object Relationship Mapping](#)

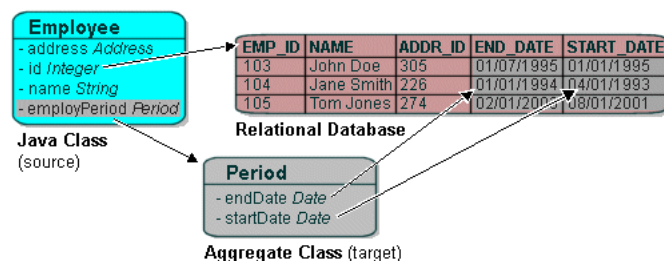
You can use an aggregate object mapping with a change policy (see ["Configuring Change Policy"](#) on page 28-70).

For more information on configuring an aggregate object relationship mapping, see ["Configuring a Relational Aggregate Object Mapping"](#) on page 47-1.

Aggregate Object Mappings with a Single Source Object

Figure 36-8 shows an example aggregate object mapping between source object `Employee` and target object `Period`. In this example, the target object is not shared by other types of source object.

Figure 36-8 *Aggregate Object Mapping with a Single Source Object*

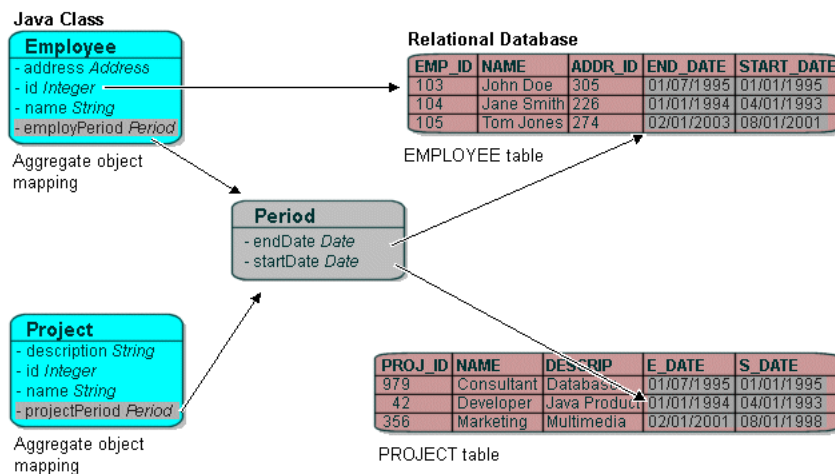


Aggregate target classes not shared among multiple source classes can have any type of mapping, including other aggregate object mappings.

Aggregate Object Mappings With Multiple Source Objects

Figure 36–9 shows an example aggregate object mapping in which different source objects—Employee and Project—map instances of the same type of target object, Period.

Figure 36–9 Aggregate Object Mapping with Multiple Source Objects



When you configure the aggregate object mapping in the source object, you choose the source object table for that particular mapping. This allows different source types to store the same target information within their tables. Each source object's table may use different field names. TopLink automatically manages the case where multiple source object tables use different field names.

For example, in Figure 36–9, The Employee attribute `employPeriod` is mapped by an aggregate object mapping to target object `Period`. This mapping associates `Period` attribute `startDate` with `EMPLOYEE` table field `START_DATE`. The Project attribute `projectPeriod` is also mapped by an aggregate object mapping to target object `Period`. This mapping associates `Period` attribute `startDate` with `PROJECT` table field `S_DATE`.

Aggregate target classes shared with multiple source classes cannot have one-to-many or many-to-many mappings.

Implementing an Aggregate Object Relationship Mapping

You must ensure that the following takes place:

- The descriptor of the target class declares itself to be an aggregate object. Because all its information comes from its parent's table(s), the target descriptor does not have a specific table associated with it. You must, however, choose one or more candidate table(s) from which you can use fields in mapping the target.

In the example above, you could choose the `EMPLOYEE` table so that the `START_DATE` and `END_DATE` fields are available during mapping.

- The descriptor of the source class adds an aggregate object mapping that specifies the target class.

In the example above, the `Employee` class has an attribute called `employPeriod` that would be mapped as an aggregate object mapping with `Period` as the reference class.

The source class must ensure that its table has fields that correspond to the field names registered with the target class.

- If a source object has a `null` target reference, TopLink writes `NULLs` to the aggregate database fields (see ["Configuring Allowing Null Values"](#) on page 47-2). When the source is read from the database, it can handle this `null` target in one of two ways:
 - Create an instance of the object with all its attributes equal to `null`.
 - Put a `null` reference in the source object without instantiating a target. (This is the default method of handling `null` targets.)

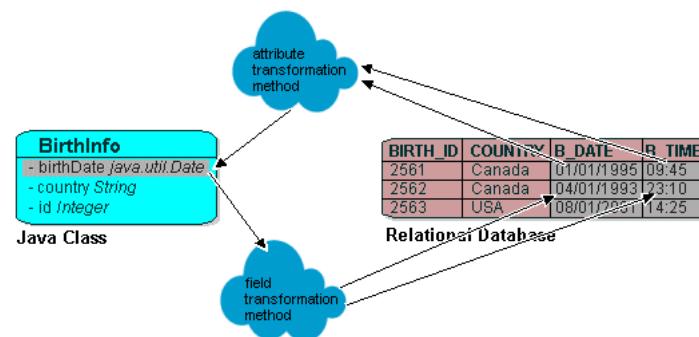
Transformation Mapping

Use transformation mappings for specialized translations for how a value is represented in Java and how it is represented in the database.

Tip: Use transformation mappings only when mapping multiple fields into a single attribute. Because of the complexity of transformation mappings, it is often easier to perform the transformation with a converter or getter and setter methods of a direct-to-field mapping. See [Chapter 38, "Configuring a Relational Direct-to-Field Mapping"](#) for more information.

Figure 36–10 illustrates a transformation mapping. The values from the `B_DATE` and `B_TIME` fields are used to create a `java.util.Date` to be stored in the `birthDate` attribute.

Figure 36–10 Transformation Mappings



Often, a transformation mapping is appropriate when values from multiple fields are used to create an object. This type of mapping requires that you provide an *attribute transformation* that is invoked when reading the object from the database. This must have at least one parameter that is an instance of `Record`. In your attribute transformation, you can use `Record` method `get` to retrieve the value in a specific column. Your attribute transformation can specify a second parameter, when it is an instance of `Session`. The `Session` performs queries on the database to get additional values needed in the transformation. The transformation should *return* the value to be stored in the attribute.

Transformation mappings also require a *field transformation* for each field, to be written to the database when the object is saved. The transformation returns the value to be stored in that field.

See [Chapter 48, "Configuring a Relational Transformation Mapping"](#) for more information.

Configuring a Relational Mapping

This chapter describes how to configure a relational mapping.

[Table 37–1](#) lists the types of relational mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

Table 37–1 *Configuring Relational Mappings*

If you are creating...	See...
Direct-to-Field Mapping	Chapter 38, "Configuring a Relational Direct-to-Field Mapping"
Transformation Mapping	Chapter 48, "Configuring a Relational Transformation Mapping"
Direct to XMLType Mapping	Chapter 39, "Configuring a Relational Direct-to-XMLType Mapping"
One-to-One Mapping	Chapter 40, "Configuring a Relational One-to-One Mapping"
Variable One-to-One Mapping	Chapter 41, "Configuring a Relational Variable One-to-One Mapping"
One-to-Many Mapping	Chapter 42, "Configuring a Relational One-to-Many Mapping"
Many-to-Many Mapping	Chapter 43, "Configuring a Relational Many-to-Many Mapping"
Aggregate Collection Mapping	Chapter 44, "Configuring a Relational Aggregate Collection Mapping"
Direct Collection Mapping	Chapter 45, "Configuring a Relational Direct Collection Mapping"
Direct Map Mapping	Chapter 46, "Configuring a Relational Direct Map Mapping"
Aggregate Object Mapping	Chapter 47, "Configuring a Relational Aggregate Object Mapping"

[Table 37–2](#) lists the configurable options shared by two or more relational mapping types.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Understanding Relational Mappings"](#) on page 36-1

Configuring Common Relational Mapping Options

[Table 37–2](#) lists the configurable options shared by two or more relational mapping types. In addition to the configurable options described here, you must also configure the options described for the specific [Relational Mapping Types](#), as shown in [Table 37–1](#).

Table 37–2 Common Relational Mapping Options

Option	Type	TopLink Workbench	Java
"Configuring a Database Field" on page 37-2	Basic	✓	✓
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Container Policy" on page 35-26	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring a Default Null Value at the Mapping Level" on page 35-12	Basic	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Basic	✓	✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring a Serialized Object Converter" on page 35-18	Advanced	✓	✓
"Configuring a Type Conversion Converter" on page 35-20	Advanced	✓	✓
"Configuring an Object Type Converter" on page 35-22	Advanced	✓	✓
"Configuring Bidirectional Relationship" on page 35-34	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Query Key Order" on page 37-8	Advanced	✓	✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Advanced	✓	✓

Configuring a Database Field

You can associate an object attribute with a database field.

Table 37–3 summarizes which relational mappings support this option.

Table 37–3 Relational Mapping Support for Database Field

Mapping	Using TopLink Workbench	Using Java
Direct-to-Field Mapping	✓	✓
Direct to XMLType Mapping	✓	✓

When choosing the database field, you must consider Java and database field type compatibility.

TopLink supports the following Java types:

- `java.lang`: Boolean, Float, Integer, String, Double, Long, Short, Byte, Byte[], Character, Character[]; all the primitives associated with these classes
- `java.math`: BigInteger, BigDecimal
- `java.sql`: Date, Time, Timestamp
- `java.util`: Date, Calendar

While executing reads, the mappings in Table 37–6 perform the simple one-way data conversions that Table 37–4 describes. For two-way or more complex conversions, You must use converters (see "Converters and Transformers" on page 36-3).

The mappings in [Table 37–3](#) also allow you to specify a null value. This may be required if primitive types are used in the object, and the database field allows null values. For more information, see ["Configuring a Default Null Value at the Mapping Level"](#) on page 35-12.

Table 37–4 Type Conversions Provided by Direct-to-Field Mappings

Java type	Database type
Integer, Float, Double, Byte, Short, BigDecimal, BigInteger, int, float, double, byte, short	NUMBER, NUMERIC, DECIMAL, FLOAT, DOUBLE, INT, SMALLINT, BIT, BOOLEAN
Boolean, boolean	BOOLEAN, BIT, SMALLINT, NUMBER, NUMERIC, DECIMAL, FLOAT, DOUBLE, INT
String	VARCHAR, CHAR, VARCHAR2, CLOB, TEXT, LONG, LONG VARCHAR, MEMO The following types apply only to Oracle9: NVARCHAR2, NCLOB, NCHAR
byte[]	BLOB, LONG RAW, IMAGE, RAW, VARBINARY, BINARY, LONG VARBINARY
Time	TIME
sql.Date	DATE (only applies to DB2)
Timestamp, util.Date, Calendar	TIMESTAMP (only applies to DB2)
sql.Date, Time, Timestamp, util.Date, Calendar	DATE, DATETIME (applies to Oracle, Sybase, SQL Server)
To use <code>oracle.sql.Timestamp</code> , see "Support for oracle.sql.Timestamp" on page 37-3.	

Support for oracle.sql.Timestamp

TopLink provides additional support for mapping Java date and time data types to Oracle database DATE, TIMESTAMP, and TIMESTAMPTZ data types when you use the Oracle JDBC driver with Oracle9i Database Server or later and the Oracle9i platform in TopLink.

In a direct-to-field mapping, you are not required to specify the database type of the field value; TopLink determines the appropriate data type conversion.

[Table 37–5](#) lists the supported direct-to-field mapping combinations.

Table 37–5 Supported Oracle Database Date and Time Direct-to-Field Mappings

Java Type	Database Type	Description
java.sql.Time	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Full bidirectional support.
java.sql.Date	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Full bidirectional support.
java.sql.Timestamp	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.

Table 37–5 (Cont.) Supported Oracle Database Date and Time Direct-to-Field Mappings

Java Type	Database Type	Description
	DATE	Nanoseconds are not stored in the database.
java.util.Date	TIMESTAMP	Full bidirectional support.
	TIMESTAMPZ	Full bidirectional support.
	DATE	Milliseconds are not stored in the database.
java.util.Calendar	TIMESTAMP	Native SQL or binding gives Calendar timezone. Note: The TIMESTAMP database value has no timezone – the Calendar object provides the local timezone by default. If the database is not in this timezone, you must obtain the database timezone by some other means and update the Calendar object accordingly. For this reason, TIMESTAMPZ may be a better choice.
	TIMESTAMPZ	Native SQL or binding stores timezone; standard SQL is based on the local timezone.
	DATE	Neither timezone nor milliseconds are stored in the database.

Note that some of these mappings result in a loss of precision: avoid these combinations if you require this level of precision. For example, if you create a direct-to-field mapping between a `java.sql.Date` attribute and a `TIMESTAMPZ` database field, there is no loss of precision. However, if you create a direct-to-field mapping between a `java.sql.Timestamp` attribute and a `DATE` database field, the nanoseconds or milliseconds of the attribute are not stored in the database.

Using TopLink Workbench

Use this procedure to select a specific database field for a direct mapping.

1. Select the direct mapping attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 37–1 Direct Mapping General Tab, Database Field Option

The screenshot shows the 'General' tab of the TopLink Workbench. The 'Database Field' dropdown menu is highlighted with a red circle and shows 'ADDRESS.CITY (VARCHAR2)'. Below this, there are several sections: 'Method Accessing' with 'Get Method' and 'Set Method' dropdowns (both set to '<none selected>'), 'Default Null Value' with 'Type' and 'Value' dropdowns (both set to '<none selected>'), 'Read-Only' checkbox, and a 'Comment' text area.

Use the **Database Field** field to select a field for this direct mapping. You must have previously associated the descriptor with a database table as described in ["Configuring Associated Tables"](#) on page 29-2.

Note: For direct to field mappings of a an Aggregate descriptor (see ["Configuring a Relational Descriptor as a Class or Aggregate Type"](#) on page 29-11), this field is for display only and cannot be changed.

Configuring Reference Descriptor

In relational mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping`, attributes reference other TopLink descriptors—not the data source. You can select any descriptor in the project.

[Table 37-6](#) summarizes which relational mappings support this option.

Table 37-6 *Relational Mapping Support for Reference Descriptor*

Mapping	Using TopLink Workbench	Using Java
One-to-One Mapping	✓	✓
Variable One-to-One Mapping	✓	✓
One-to-Many Mapping	✓	✓
Many-to-Many Mapping	✓	✓
Aggregate Collection Mapping		✓
Aggregate Object Mapping	✓	✓

Using TopLink Workbench

To specify a reference descriptor for a relational mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 37-2 General Tab, Reference Descriptor Field

The screenshot shows the 'General' tab of the 'Table Reference' dialog. The 'Reference Descriptor' field is highlighted with a red circle and contains a dropdown menu with 'Address (examples.servletjsp.model)' selected. Below this field are several checkboxes: 'Method Accessing' (unchecked), 'Read-Only' (unchecked), 'Private Owned' (unchecked), 'Batch Reading' (unchecked), 'Use Joining' (unchecked), and 'Use Indirection' (checked). There are also radio buttons for 'ValueHolder' (selected) and 'Proxy' (unchecked). At the bottom, there is a 'Maintains Bidirectional Relationship' checkbox (unchecked) and a 'Relationship Partner' dropdown menu (set to '<none selected>'). A 'Comment' text area is at the very bottom.

Use the **Reference Descriptor** field to select the descriptor referenced by this relationship mapping.

Note: For aggregate mappings the **Reference Descriptor** must be an *aggregate*. See ["Configuring a Relational Descriptor as a Class or Aggregate Type"](#) on page 29-11 for more information.

For variable one-to-one mappings, the Reference Descriptor must be an *interface*. See [Chapter 41, "Configuring a Relational Variable One-to-One Mapping"](#) for more information.

You can specify a reference descriptor that is not in the current TopLink Workbench project. For example, to create a mapping to an `Employee` class that does not exist in the current project, do the following:

1. Add the `Employee` class to your current project. See ["Working With Projects"](#) on page 21-10.
2. Create the relationship mapping to the `Employee` descriptor.
3. Deactivate the `Employee` descriptor. See ["Active and Inactive Descriptors"](#) on page 4-10.

When you generate the deployment XML for your project, the *mapping* to the `Employee` class will be included, but not the `Employee` class.

Configuring Batch Reading

Batch reading can be used in most of the relational mappings. This feature should be used only if it is known that the related objects are always required with the source object.

[Table 37-7](#) summarizes which relational mappings support this option.

Table 37-7 Relational Mapping Support for Batch Reading

Mapping	Using TopLink Workbench	Using Java
One-to-One Mapping	✓	✓
One-to-Many Mapping	✓	✓
Many-to-Many Mapping	✓	✓
Direct Collection Mapping	✓	✓
Direct Map Mapping	✓	✓
Aggregate Object Mapping		✓

Using TopLink Workbench

To use batch reading in a relationship mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 37-3 General Tab, Batch Reading Option

The screenshot shows the 'General' tab of the TopLink Workbench. The 'Reference Descriptor' is set to 'Address (examples.servletjsp.model)'. Under 'Method Accessing', 'Get Method' and 'Set Method' are both set to '<none selected>'. Under 'Read-Only', 'Private Owned', and 'Batch Reading' are unchecked. 'Batch Reading' is circled in red. 'Use Joining' is unchecked, and 'Use Indirection' is checked. Under 'ValueHolder', 'ValueHolder' is selected. 'Maintains Bidirectional Relationship' is unchecked, and 'Relationship Partner' is set to '<none selected>'. A 'Comment' field is at the bottom.

To specify that this mapping using batch reading, select the **Batch Reading** option.

Using Java

Example 37-1 Query Optimization Using Batching

The following code example illustrates using batch for query optimization.

```
// Queries on Employee are configured to always batch read Address
OneToManyMapping phoneNumbersMapping = new OneToManyMapping();
phoneNumbersMapping.setReferenceClass("
PhoneNumber.class")
phoneNumbersMapping.setAttributeName("phones");
```

```
phoneNumbersMapping.useBatchReading();
phoneNumbersMapping.privateOwnedRelationship();
```

Configuring Query Key Order

You can configure TopLink to maintain collections in order by query key.

Table 37–8 summarizes which relational mappings support this option.

Table 37–8 Relational Mapping Support for Query Key Order

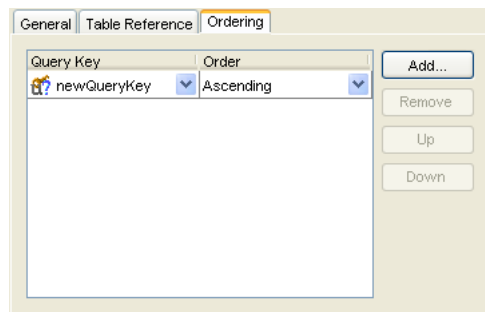
Mapping	Using TopLink Workbench	Using Java
Variable One-to-One Mapping	✓	✓
One-to-Many Mapping	✓	✓
Aggregate Collection Mapping		✓

Using TopLink Workbench

To specify the order of a mapping's query keys, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Ordering** tab. The Ordering tab appears.

Figure 37–4 Ordering Tab



3. Complete the **Ordering** options on the tab.

Field	Description
Query Key	Specify the query key to order by. Click Add to add query keys to, or Remove to remove query keys from the ordering operation. Click Up or Down to change the sort order of selected query keys.
Order	Specify if TopLink orders the selected query key in Ascending or Descending (alphabetical) order.

Configuring Table and Field References (Foreign and Target Foreign Keys)

A foreign key is a combination of one or more database columns that reference a unique key, usually the primary key, in another table. Foreign keys can be any number

of fields (similar to a primary key), all of which are treated as a unit. A foreign key and the parent key it references must have the same number and type of fields.

Mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping` use foreign keys to find information in the database so that the target object(s) can be instantiated. For example, if every `Employee` has an attribute `address` that contains an instance of `Address` (which has its own descriptor and table) then, the one-to-one mapping for the `address` attribute would specify foreign key information to find an `Address` for a particular `Employee`.

TopLink classifies foreign keys into two categories in mappings—**foreign keys** and **target foreign keys**:

- In a *foreign key*, the key is found in the table associated with the mapping's own descriptor. For example, an `Employee` foreign key to `ADDRESS` would be in the `EMPLOYEE` table.
- In a *target foreign key*, the reference is from the target object's table back to the key from the mapping's descriptor's table. For example, the `ADDRESS` table would have a foreign key to `EMPLOYEE`.

Caution: Make sure you fully understand the distinction between *foreign key* and *target foreign key* before defining a mapping.

The table reference is the database table that contains the foreign key references.

Table 37–9 summarizes which relational mappings support this option.

Table 37–9 Relational Mapping Support for Table Reference

Mapping	Using TopLink Workbench	Using Java
One-to-One Mapping	✓	✓
One-to-Many Mapping	✓	✓
Many-to-Many Mapping	✓	✓
Aggregate Collection Mapping		✓
Direct Collection Mapping	✓	✓
Direct Map Mapping	✓	✓

Using TopLink Workbench, you can either import this table from your database or create it. If you import tables from the database (see ["Importing Tables From a Database"](#) on page 4-22), TopLink creates references that correspond to existing database constraints (if supported by the driver). You can also define references in TopLink without creating similar constraints on the database.

Using TopLink Workbench

To specify a table for a mapping reference, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Table Reference** tab. The Reference tab appears.

Figure 37–5 Table Reference Tab, Table Reference Field

General | **Table Reference**

Table Reference: EMPLOYEE_EMPLOYEE (EMPLOYEE => EMPLOYEE) New...

Source Field	Target Field	Target Foreign Key
MANAGER_ID	EMP_ID	☐

Add Remove

Use the following information to select the field references on the tab:

Field	Description
Table Reference	Select an existing table, or click New to create a new table reference.
Source and Target Field	Click Add to create new foreign key reference. To delete an existing key pair reference, select the Source and Target fields and click Remove .
Source Field	Select the database field from the <i>source</i> table for this foreign key reference.
Target Field	Select the database field from the <i>target</i> table for this foreign key reference.
Target Foreign Key	Specify whether or not the reference is from the target object's table back to the key from the mapping's descriptor's table.

Configuring a Relational Direct-to-Field Mapping

This chapter describes the various components that you must configure in order to use a relational direct-to-field mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Direct-to-Field Mapping"](#) on page 36-4

Relational Direct-to-Field Mapping Configuration Overview

[Table 38–1](#) lists the configurable options for a relational direct-to-field mapping.

Table 38–1 Configurable Options for Relational Direct-to-Field Mapping

Option	Type	TopLink Workbench	Java
"Configuring a Database Field" on page 37-2	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring a Default Null Value at the Mapping Level" on page 35-12	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring a Serialized Object Converter" on page 35-18	Advanced	✓	✓
"Configuring a Type Conversion Converter" on page 35-20	Advanced	✓	✓
"Configuring an Object Type Converter" on page 35-22	Advanced	✓	✓

Configuring a Relational Transformation Mapping

This chapter describes the various components that you must configure in order to use a relational transformation mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Transformation Mapping"](#) on page 36-15

Relational Transformation Mapping Configuration Overview

[Table 48-1](#) lists the configurable options for a relational transformation mapping.

Table 48-1 Configurable Options for Relational Transformation Mapping

Option	Type	TopLink Workbench	Java
"Configuring Attribute Transformer" on page 35-29	Basic	✓	✓
"Configuring Field Transformer Associations" on page 35-31	Basic	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Mutable Mappings" on page 35-33	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓

Configuring a Relational Direct-to-XMLType Mapping

This chapter describes the various components that you must configure in order to use a relational direct-to-XMLType mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Direct to XMLType Mapping"](#) on page 36-4

Relational Direct-to-XMLType Mapping Overview

[Table 39-1](#) lists the configurable options for a relational direct-to-XMLType mapping.

Table 39-1 Configurable Options for Relational Direct-To-XMLType Mapping

Option	Type	TopLink Workbench	Java
"Configuring a Database Field" on page 37-2	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Read Whole Document" on page 39-1	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	

Configuring Read Whole Document

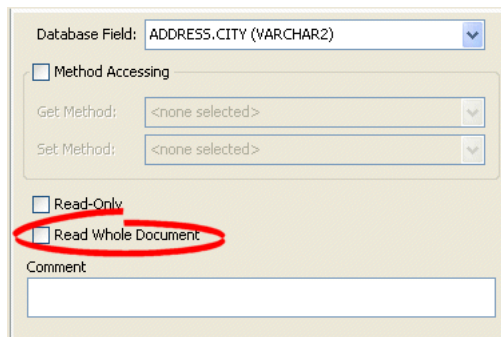
When mapping an XML Type to a Document Object Model (DOM), by default TopLink uses the database representation of the DOM. This allows for lazy loading of the XML data from the database.

However, if you require the entire DOM, (or if you require the DOM to be available in a disconnected fashion from the database connection) use the **Read Whole** option to retrieve the entire DOM from the database.

Using TopLink Workbench

To specify that this mapping reads the whole XML document, use this procedure:

1. Select the mapping in the **Navigator**. Its properties appear in the Editor.
2. Click **General**. The General tab appears.

Figure 39–1 Direct to XML Mapping Property Sheet, Read Whole Document Option

The image shows a 'Direct to XML Mapping Property Sheet' dialog box. At the top, there is a 'Database Field:' dropdown menu with 'ADDRESS.CITY (VARCHAR2)' selected. Below this is a section titled 'Method Accessing' which contains two dropdown menus: 'Get Method:' and 'Set Method:', both currently set to '<none selected>'. Underneath the 'Method Accessing' section are two checkboxes: 'Read-Only' and 'Read Whole Document'. The 'Read Whole Document' checkbox is circled in red. At the bottom of the dialog is a 'Comment' text area.

Choose the **Read Whole Document** option to read the whole XML document. If you do not select this option, the connection must remain open for TopLink to read the database values.

Configuring a Relational One-to-One Mapping

This chapter describes the various components that you must configure in order to use a relational one-to-one mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["One-to-One Mapping"](#) on page 36-5

Relational One-to-One Mapping Configuration Overview

[Table 40–1](#) lists the configurable options for a relational one-to-one mapping.

Table 40–1 Configurable Options for Relational One-to-One Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Joining at the Mapping Level" on page 40-1	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Bidirectional Relationship" on page 35-34	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Basic	✓	✓

Configuring Joining at the Mapping Level

TopLink supports configuring an inner join at the mapping level for one-to-one mappings only. When a class that owns the mapping is read, the TopLink runtime will always get the class and the target of the one-to-one mapping with one database hit.

Use this feature only if the target object is *always* required with the source object, or when indirection is *not* used.

You can also configure join reading (see ["Join Reading and Object-Level Read Queries"](#) on page 96-12) at the query level.

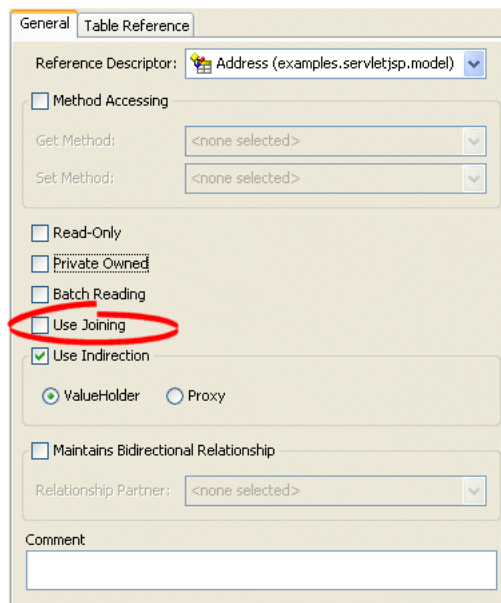
For more information about joins, see ["Expressions for Joining and Complex Relationships"](#) on page 97-5.

Using TopLink Workbench

To use joining in a relationship mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 40–1 General Tab, Use Joining Option



The screenshot shows the 'General' tab of a configuration window. The 'Reference Descriptor' is set to 'Address (examples.servletjsp.model)'. Under 'Method Accessing', 'Get Method' and 'Set Method' are both set to '<none selected>'. The 'Read-Only', 'Private Owned', and 'Batch Reading' options are unchecked. The 'Use Joining' option is checked and circled in red. The 'Use Indirection' option is also checked. Under 'ValueHolder' and 'Proxy', 'ValueHolder' is selected. The 'Maintains Bidirectional Relationship' option is unchecked, and the 'Relationship Partner' is set to '<none selected>'. A 'Comment' text area is at the bottom.

To use joining with this relationship, select the **Use Joining** option.

Configuring a Relational Variable One-to-One Mapping

This chapter describes the various components that you must configure in order to use a relational variable one-to-one mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Variable One-to-One Mapping"](#) on page 36-6

Relational Variable One-to-One Mapping Configuration Overview

[Table 41-1](#) lists the configurable options for a relational variable one-to-one mapping.

Table 41-1 Configurable Options for Relational Variable One-to-One Mapping

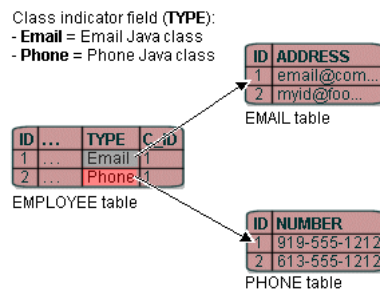
Option	Type	TopLink Workbench	Java
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Query Key Association" on page 41-4	Advanced	✓	✓
"Configuring Class Indicator" on page 41-1	Basic	✓	✓
"Configuring Unique Primary Key" on page 41-3	Basic	✓	✓

Configuring Class Indicator

In variable one-to-one mappings, you can use an indicator column in the source table to specify the correct target table, as illustrated in [Figure 41-1](#). This section includes information on implementing class indicators.

A source table has an indicator column that specifies the target table through the class indicator field.

[Figure 41-1](#) illustrates the EMPLOYEE table that has a TYPE column that indicates the target for the row (either PHONE or EMAIL).

Figure 41–1 Variable One-to-One Mapping using Class indicator Field

The principles of defining such a variable class relationship are similar to defining a normal one-to-one relationship, except:

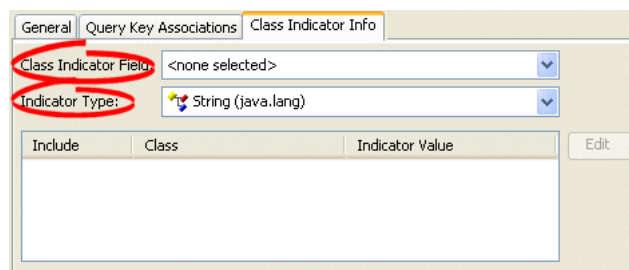
- The reference class is a Java *interface*, not a Java *class*. However, the method to set the interface is identical.
- You must specify a type indicator field.
- You specify the class indicator values on the mapping so that mapping can determine the class of object to create.
- You must specify the foreign key names and the respective abstract query keys from the target interface descriptor.

Alternatively, you can use unique primary keys to specify the correct target. See ["Configuring Unique Primary Key"](#) on page 41-3 for more information.

Using TopLink Workbench

To specify the class indicators for a variable one-to-one mapping, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.
2. Click the **Class Indicator Info** tab. The Class Indicator Info tab appears.

Figure 41–2 Class Indicator Info Tab

Use the **Class Indicator Field** to select the field on the database table (associated with the mapping's descriptor) to use as the indicator for the variable mapping.

Use the **Indicator Type** to specify the data type of the class indicator field by selecting the data type from the list.

To specify the specific class indicator field values for each (nonabstract) child class, click **Edit** and enter the appropriate value for each child class.

Configuring Unique Primary Key

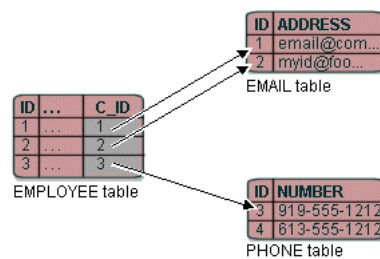
In variable one-to-one mappings, you can use a unique primary key in the source table to specify the correct target table, as illustrated in [Figure 41-3](#). This section includes information on implementing class indicators.

Alternatively, you can use a class indicator to specify the correct target. See ["Configuring Class Indicator"](#) on page 41-3 for more information.

Understanding Unique Primary Key

As [Figure 41-3](#) illustrates, the value of the foreign key in the source table (C_ID) mapped to the primary key of the target table is unique. No primary key values among the target tables are the same, so primary key values are not unique just in the table, but also among the tables.

Figure 41-3 Variable One-to-One Relationship with Unique Primary Key



If there is no indicator stored in the source table, TopLink cannot determine to which target table the foreign key value is mapped. Therefore, TopLink reads through all the target tables until it finds an entry in one of the target tables. This is an inefficient way of setting up a relation model. The class indicator is much more efficient as it reduces the number of reads performed on the tables to get the data. In the class indicator method, TopLink knows exactly which target table to look into for the data.

The principles of defining such a variable class relationship are similar to defining class indicator variable one-to-one relationships, except:

- A type indicator field is not specified.
- The class indicator values are not specified.

The type indicator field and its values are not needed, because TopLink goes through all the target tables until data is finally found.

Using TopLink Workbench

To specify the variable one-to-one mapping with a primary key, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.
2. Click the **Class Indicator Info** tab. The Class Indicator Info tab appears.

Figure 41–4 Class Indicator Info Tab, Configuring Primary Key

Use the **Class Indicator Field** to select none.

Use the **Indicator Type** to select none.

Use the **Indicator Value** column to specify none.

After choosing the reference descriptor for the mapping, deselect the **Include** check boxes.

Note: You cannot deselect the value in the Class Indicator Field, unless the foreign key values in the source table are unique.

Using Java

[Example 41–1](#) illustrates how to define a variable one-to-one mapping using a unique primary key in Java code.

Example 41–1 Defining a variable one-to-one mapping using a unique primary key

```
VariableOneToOneMapping variableOneToOneMapping = new VariableOneToOneMapping();
variableOneToOneMapping.setAttributeName("contact");
variableOneToOneMapping.setReferenceClass (Contact.class);
variableOneToOneMapping.setForeignQueryKeyName ("C_ID", "id");
variableOneToOneMapping.dontUseIndirection();
variableOneToOneMapping.privateOwnedRelationship();
```

Configuring Query Key Association

The API to configure query key associations is `oracle.toplink.mappings.VariableOneToOneMapping` method `addForeignQueryKeyName(String, String)`.

Using TopLink Workbench

To specify the query keys used for a variable one-to-one mapping, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.
2. Click the **Query Key Associations** tab. The Query Key Associations tab appears

Figure 41-5 Query Key Associations Tab

The screenshot shows a software window with three tabs: 'General', 'Query Key Associations', and 'Class Indicator Info'. The 'Query Key Associations' tab is active. It contains a table with two columns: 'Foreign Key' and 'Query Key Name'. The 'Foreign Key' column has a dropdown menu with the text 'EMPLOYEE.ADDR_ID (NUMBER)'. The 'Query Key Name' column has a dropdown menu with the text 'id'. To the right of the table are two buttons: 'Add' and 'Remove'.

Use the following information to enter data in each field on the tab:

Use the **Indicator Type** to specify the data type of the class indicator field by selecting the data type from the list.

Field	Description
Foreign Key	The field from the database table to use as the foreign key in this relationship.
Query Key Name	The name of the query key (from the referenced descriptor) for this association. See "Configuring Query Keys" on page 28-29 for more information.

Configuring a Relational One-to-Many Mapping

This chapter describes the various components that you must configure in order to use a relational one-to-many mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["One-to-Many Mapping"](#) on page 36-7

Relational One-to-Many Mapping Configuration Overview

[Table 42-1](#) lists the configurable options for a relational one-to-many mapping.

Table 42-1 *Configurable Options for Relational One-to-Many Mapping*

Option	Type	TopLink Workbench	Java
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Bidirectional Relationship" on page 35-34	Advanced	✓	✓
"Configuring Container Policy" on page 35-26	Basic	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Basic	✓	✓
"Configuring Query Key Order" on page 37-8	Advanced	✓	✓

Configuring a Relational Many-to-Many Mapping

This chapter describes the various components that you must configure in order to use a relational many-to-many mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Many-to-Many Mapping"](#) on page 36-8

Relational Many-to-Many Mapping Configuration Overview

[Table 43–1](#) lists the configurable options for a relational many-to-many mapping.

Table 43–1 Configurable Options for Relational Many-to-Many Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Bidirectional Relationship" on page 35-34	Basic	✓	✓
"Configuring Container Policy" on page 35-26	Basic	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring a Relation Table" on page 43-1	Basic	✓	✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8 (Source)	Basic	✓	✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8 (Target)	Basic	✓	✓
"Configuring Query Key Order" on page 37-8	Advanced	✓	✓

Configuring a Relation Table

The relation table contains the columns for the primary keys of the source table and target table involved in the many-to-many mapping. You must create this table in the

database before completing the mapping. See ["Working With Databases"](#) on page 4-21 for information on creating database tables.

In [Figure 36-5](#) on page 36-9, the PROJ_EMP table serves as the relation table between the PROJECT and EMPLOYEE tables.

Using TopLink Workbench

To select a relation table for a mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 43-1 Table Reference Tab, Relation Table Option

The screenshot shows the 'Table Reference' tab in the TopLink Workbench. The 'Relation Table' dropdown is highlighted with a red circle and contains the text 'PROJ_EMP'. Below this, there are two sections: 'Source Reference' and 'Target Reference'. The 'Source Reference' section has a 'Table Reference' dropdown set to 'PROJEMP_EMP (PROJ_EMP => EMPLOYEE)', a table with 'Source Column' (EMP_ID) and 'Target Column' (EMP_ID), and buttons for 'New...', 'Add', and 'Remove'. The 'Target Reference' section has a 'Table Reference' dropdown set to 'PROJ_EMP_PROJECT (PROJ_EMP => PROJECT)', a table with 'Source Column' (PROJ_ID) and 'Target Column' (PROJ_ID), and buttons for 'New...', 'Add', and 'Remove'.

Use the **Relation Table** field to select a database table to define this mapping.

Configuring a Relational Aggregate Collection Mapping

This chapter describes the various components that you must configure in order to use a relational aggregate collection mapping.

Note: To use a relational aggregate collection mapping with TopLink Workbench, you must use an amendment method (see ["Configuring Amendment Methods"](#) on page 28-78).

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Aggregate Collection Mapping"](#) on page 36-10

Relational Aggregate Collection Mapping Configuration Overview

[Table 44–1](#) lists the configurable options for a relational aggregate collection mapping.

Table 44–1 Configurable Options for Relational Aggregate Collection Mapping

Option	Type	TopLink Workbench	Java
"Configuring a Database Field" on page 37-2	Basic		✓
"Configuring Reference Descriptor" on page 37-5	Basic		✓
"Configuring Container Policy" on page 35-26	Basic		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring Read-Only Mappings" on page 35-2	Advanced		✓
"Configuring Batch Reading" on page 37-6	Advanced		✓
"Configuring Bidirectional Relationship" on page 35-34	Advanced		✓
"Configuring Query Key Order" on page 37-8	Advanced		✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Advanced		✓

Configuring a Relational Direct Collection Mapping

This chapter describes the various components that you must configure in order to use a relational direct collection mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Direct Collection Mapping"](#) on page 36-11

Relational Direct Collection Mapping Configuration Overview

[Table 45-1](#) lists the configurable options for a relational direct collection mapping.

Table 45-1 Configurable Options for Relational Direct Collection Mapping

Option	Type	TopLink Workbench	Java
"Configuring Target Table" on page 45-1	Basic	✓	✓
"Configuring Direct Value Field" on page 45-2	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Container Policy" on page 35-26	Basic	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring a Serialized Object Converter" on page 35-18	Advanced	✓	✓
"Configuring a Type Conversion Converter" on page 35-20	Advanced	✓	✓
"Configuring an Object Type Converter" on page 35-22	Advanced	✓	✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Basic	✓	✓

Configuring Target Table

Each direct collection stores reference information in a target table. In [Figure 36-6](#) on page 36-11, the `RESPONS` table contains the primary key and object of the instance owning the collection. You must create this table in your database.

Using TopLink Workbench

To specify the direct collection specifics, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 45–1 General Tab, Target Table Options

The screenshot shows the 'General' tab of a configuration window. The 'Target Table' dropdown is highlighted with a red circle. Below it is the 'Direct Value Field' dropdown. Further down are checkboxes for 'Method Accessing', 'Read-Only', and 'Batch Reading'. There are also 'Get Method' and 'Set Method' dropdowns. At the bottom, there is a 'Use Indirection' checkbox (checked) and radio buttons for 'ValueHolder' (selected) and 'Transparent'. An 'Advanced...' button and a 'Comment' text area are at the very bottom.

Use the **Target Table** list to select the table that contains the reference fields for the direct collection mapping.

Configuring Direct Value Field

The direct value field, located in the reference table, stores the primitive data value. In [Figure 36–6](#) on page 36-11, the DESCRIP field stores the collection.

Using TopLink Workbench

To specify the direct collection specifics, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 45–2 General Tab, Direct Value Field

The screenshot shows a configuration window with three tabs: 'General', 'Converter', and 'Table Reference'. The 'General' tab is active. It contains the following elements:

- Target Table:** A dropdown menu currently showing '<none selected>'.
- Direct Value Field:** A dropdown menu currently showing '<none selected>'. This label and its dropdown are circled in red.
- Method Accessing:** A checkbox that is unchecked.
- Get Method:** A dropdown menu currently showing '<none selected>'.
- Set Method:** A dropdown menu currently showing '<none selected>'.
- Read-Only:** A checkbox that is unchecked.
- Batch Reading:** A checkbox that is unchecked.
- Use Indirection:** A checkbox that is checked.
- ValueHolder:** A radio button that is selected.
- Transparent:** A radio button that is unselected.
- Advanced...** A button.
- Comment:** A text area.

Use the **Direct Value Field** list to select the field from the **Target Table** table that contains the object of the collection.

Configuring a Relational Direct Map Mapping

This chapter describes the various components that you must configure in order to use a relational direct map mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Direct Map Mapping"](#) on page 36-12

Relational Direct Map Mapping Configuration Overview

[Table 46–1](#) lists the configurable options for a relational direct map mapping.

Table 46–1 Configurable Options for Relational Direct Map Mapping

Option	Type	TopLink Workbench	Java
"Configuring Target Table" on page 45-1	Basic	✓	✓
"Configuring Direct Value Field" on page 46-1	Basic	✓	✓
"Configuring Direct Key Field" on page 46-2	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Batch Reading" on page 37-6	Advanced	✓	✓
"Configuring Indirection" on page 35-3	Advanced	✓	✓
"Configuring Container Policy" on page 35-26	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Key Converters" on page 46-3	Advanced	✓	✓
"Configuring Value Converters" on page 46-4	Advanced	✓	✓
"Configuring Table and Field References (Foreign and Target Foreign Keys)" on page 37-8	Basic	✓	✓

Configuring Direct Value Field

The direct value field in the reference table stores the primitive data value of the map value. If the value's object value and database value are different types, use a converter (see ["Configuring Value Converters"](#) on page 46-3).

Using TopLink Workbench

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.

2. Click the **General** tab. The General tab appears.

Figure 46–1 General Tab, Direct Value Field

The screenshot shows a configuration dialog with three tabs: General, Converter, and Table Reference. The General tab is active. It contains several dropdown menus and checkboxes. The 'Direct Value Field' dropdown is circled in red. Below it are 'Target Table', 'Direct Key Field', 'Method Accessing' (checkbox), 'Get Method', 'Set Method', 'Read-Only' (checkbox), 'Batch Reading' (checkbox), 'Use Indirection' (checked checkbox), 'ValueHolder' (selected radio button), and 'Transparent' (radio button). At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

Use the **Direct Value Field** list to select the field from the **Target Table** table that contains the object of the direct map mapping.

Configuring Direct Key Field

The direct key field in the reference table stores the primitive data value of the map key. If the key's object value and database value are different types, use a converter (see "[Configuring Key Converters](#)" on page 46-3).

Using TopLink Workbench

To specify the direct key field in the reference table, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 46–2 General Tab, Direct Key Field

The screenshot shows the 'General' tab of a configuration dialog. At the top are three tabs: 'General', 'Converter', and 'Table Reference'. Below them are three dropdown menus: 'Target Table:' (set to '<none selected>'), 'Direct Value Field:' (set to '<none selected>'), and 'Direct Key Field:' (set to '<none selected>'). The 'Direct Key Field:' dropdown is circled in red. Below these is a section for 'Method Accessing' with a checkbox (unchecked) and two dropdowns for 'Get Method:' and 'Set Method:' (both set to '<none selected>'). Further down are checkboxes for 'Read-Only' (unchecked), 'Batch Reading' (unchecked), and 'Use Indirection' (checked). Below these are two radio buttons: 'ValueHolder' (selected) and 'Transparent' (unselected). At the bottom is an 'Advanced...' button and a 'Comment' text area.

Use the **Direct Key Field** list to select the key from the **Target Table** table that contains the object of the direct map mapping.

Configuring Key Converters

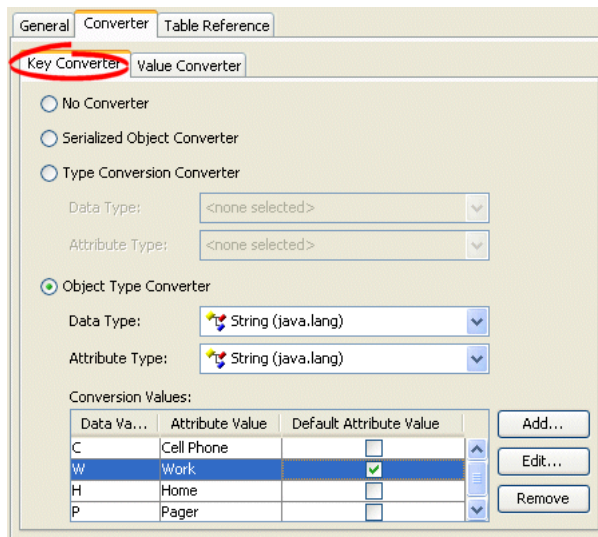
If the key's object value and database value are different types, use a converter. TopLink supports the following key converters:

- see ["Serialized Object Converter"](#) on page 33-10
- see ["Type Conversion Converter"](#) on page 33-11
- see ["Object Type Converter"](#) on page 33-12

Using TopLink Workbench

Use this procedure to specify the converter for a direct map mapping key:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.
3. Click the **Key Converter** tab. The Key Converter tab appears.

Figure 46-3 Converter Tab, Key Converter Subtab

Use the following information to select the converter for the direct map mapping key:

Converter	Description
No Converter	Do not use a Key Converter for this mapping.
Serialized Object Converter	See "Configuring a Serialized Object Converter" on page 35-18.
Type Conversion Converter	See "Configuring a Type Conversion Converter" on page 35-20.
Object Type Converter	See "Configuring an Object Type Converter" on page 35-22.

Configuring Value Converters

If the value's object value and database value are different types, use a converter. TopLink supports the following value converters:

- see ["Serialized Object Converter"](#) on page 33-10
- see ["Type Conversion Converter"](#) on page 33-11
- see ["Object Type Converter"](#) on page 33-12

Using TopLink Workbench

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.
3. Click the **Value Converter** tab. The Value Converter tab appears.

Figure 46-4 Converter Tab, Value Converter Subtab

General Converter Table Reference

Key Converter Value Converter

☐ No Converter

☐ Serialized Object Converter

☐ Type Conversion Converter

Data Type: <none selected>

Attribute Type: <none selected>

☒ Object Type Converter

Data Type: String (java.lang)

Attribute Type: String (java.lang)

Conversion Values:

Data Value	Attribute Value	Default Attribute...
C	Cell Phone	☐
W	Work	☒
H	Home	☐
P	Pager	☐

Add... Edit... Remove

4. Select the appropriate Value Converter.

Use the following information to select the converter for the direct map mapping value:

Converter	Description
No Converter	Do not use a Value Converter for this mapping.
Serialized Object Converter	See "Configuring a Serialized Object Converter" on page 35-18.
Type Conversion Converter	See "Configuring a Type Conversion Converter" on page 35-20.
Object Type Converter	See "Configuring an Object Type Converter" on page 35-22.

Configuring a Relational Aggregate Object Mapping

This chapter describes the various components that you must configure in order to use a relational aggregate object mapping.

Note: You configure the relational aggregate object mapping in the source object's descriptor. However, before doing so, you must designate the target object's descriptor as an aggregate (see ["Configuring a Relational Descriptor as a Class or Aggregate Type"](#) on page 29-11).

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Aggregate Object Mapping"](#) on page 36-12

Relational Aggregate Object Mapping Configuration Overview

[Table 47-1](#) lists the configurable options for a relational aggregate object mapping.

Table 47-1 Configurable Options for Relational Aggregate Object Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Descriptor" on page 37-5	Basic	✓	✓
"Configuring Method Accessing" on page 35-14	Advanced	✓	✓
"Configuring Read-Only Mappings" on page 35-2	Advanced	✓	✓
"Configuring Allowing Null Values" on page 47-2	Advanced	✓	✓
"Configuring Mapping Comments" on page 35-18	Advanced	✓	
"Configuring Aggregate Fields" on page 47-1	Basic	✓	✓

Configuring Aggregate Fields

When you designate a descriptor as an aggregate, TopLink allows you to specify a mapping type for each field in the target class, but defers associating the field with a database table until you configure the aggregate object mapping in the source class descriptor. In other words, the target class descriptor defines how each target class field is mapped but the source class descriptor defines where each target class field is mapped.

This section explains how to configure the source class descriptor to define where each target class field is mapped.

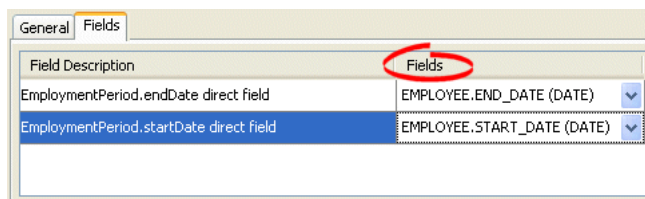
For more information on how to configure the target class descriptor to define how each target class field is mapped, see ["Configuring a Relational Descriptor as a Class or Aggregate Type"](#) on page 29-11.

Using TopLink Workbench

To specify the mapped fields of an aggregate mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Fields** tab. The Fields tab appears.

Figure 47-1 Fields Tab



3.

Use the following information to complete each field on the tab:

Field	Description
Field Description	This column shows the name of the fields from the target object, whose descriptor is designated as an aggregate (see "Configuring a Relational Descriptor as a Class or Aggregate Type" on page 29-11). These are for display only and cannot be changed.
Fields	Use this column to select the source object database table field that TopLink will map to the corresponding target object field.

Configuring Allowing Null Values

If all the fields in the database row for the aggregate object are `null`, then, by default, TopLink places `null` in the appropriate source object, as opposed to filling an aggregate object with `null` values.

Using TopLink Workbench

To allow a mapping to contain a null value, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

Figure 47-2 General Tab, Allow Null Option

The screenshot shows a configuration window with two tabs: 'General' and 'Fields'. The 'General' tab is active. It contains a 'Reference Descriptor' dropdown menu set to 'EmploymentPeriod (examples.servletjsp.model)'. Below this is a 'Method Accessing' section with two dropdown menus for 'Get Method' and 'Set Method', both set to '<none selected>'. Further down are two checkboxes: 'Read-Only' and 'Allows Null'. The 'Allows Null' checkbox is checked and circled in red. At the bottom is a 'Comment' text area.

Select the **Allows Null** option to allow this mapping to contain a null value.

Part XII

Object-Relational Mappings

An object-relational mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational databases (such as Oracle Database). Object-relational mappings let you map an object model into an object-relational data model.

This part contains the following chapters:

- [Chapter 49, "Understanding Object-Relational Mappings"](#)
This chapter describes each of the different TopLink object-relational mapping types and important object-relational mapping concepts.
- [Chapter 50, "Configuring an Object-Relational Mapping"](#)
This chapter explains how to configure TopLink object-relational mapping options common to two or more object-relational mapping types.
- [Chapter 52, "Configuring an Object-Relational Reference Mapping"](#)
This chapter explains how to configure a reference object-relational mapping.
- [Chapter 55, "Configuring an Object-Relational Nested Table Mapping"](#)
This chapter explains how to configure a nested table object-relational mapping.
- [Chapter 54, "Configuring an Object-Relational Object Array Mapping"](#)
This chapter explains how to configure an object array object-relational mapping.
- [Chapter 51, "Configuring an Object-Relational Structure Mapping"](#)
This chapter explains how to configure a structure object-relational mapping.
- [Chapter 53, "Configuring an Object-Relational Array Mapping"](#)
This chapter explains how to configure an array object-relational mapping.

Understanding Object-Relational Mappings

An object-relational mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational databases (such as Oracle Database). Object-relational mappings let you map an object model into an object-relational data model.

Do not confuse object-relational mappings with relational mappings (see ["Understanding Relational Mappings"](#) on page 36-1). A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model. In general, you can use relational mappings with any supported relational database. You can only use object-relational mappings with specialized object-relational databases optimized to support object-relational data source representations.

This chapter describes:

- [Object-Relational Mapping Types](#)

Object-Relational Mapping Types

TopLink supports the object-relational mappings listed in [Table 49-1](#).

These mappings allow for an object model to persist in an object-relational data model. Currently, TopLink Workbench does not support object-relational mappings—they must be defined in code or through amendment methods.

Table 49-1 *TopLink Object Relationship Mapping Types*

Type	Description	Type	TopLink Workbench	Java
"Object-Relational Structure Mapping" on page 49-2	Map to object-relational aggregate structures (the <code>Struct</code> type in JDBC and the <code>OBJECT</code> type in Oracle Database).	Advanced		✓
"Object-Relational Reference Mapping" on page 49-2	Map to object-relational references (the <code>Ref</code> type in JDBC and the <code>REF</code> type in Oracle Database).	Advanced		✓
"Object-Relational Array Mapping" on page 49-2	Map a collection of primitive data to object-relational array data types (the <code>Array</code> type in JDBC and the <code>VARRAY</code> type in Oracle Database).	Advanced		✓
"Object-Relational Object Array Mapping" on page 49-2	Map to object-relational array data types (the <code>Array</code> type in JDBC and the <code>VARRAY</code> type in Oracle Database).	Advanced		✓
"Object-Relational Nested Table Mapping" on page 49-3	Map to object-relational nested tables (the <code>Array</code> type in JDBC and the <code>NESTED TABLE</code> type in Oracle Database).	Advanced		✓

Object-Relational Structure Mapping

In an object-relational data model, **structures** are user-defined data types or object types. This is similar to a Java class—it defines attributes or fields in which each attribute is one of the following:

- A primitive data type
- Another structure
- Reference to another structure

TopLink maps nested structures with the `StructuringMapping` class. The structure mapping supports null values and shared aggregates without requiring additional settings (because of the object-relational support of the database).

See ["Configuring an Object-Relational Structure Mapping"](#) on page 51-1 for more information.

Object-Relational Reference Mapping

In an object-relational data model, structures reference each other through **refs**—not through foreign keys (as in a traditional data model). Refs are based on the target structure's `ObjectID`. They represent an object reference in Java.

TopLink maps refs with the `ReferenceMapping` class. The reference mapping does not require foreign key information (because of the object-relational support of the database).

See ["Configuring an Object-Relational Reference Mapping"](#) on page 52-1 for more information.

Object-Relational Array Mapping

In an object-relational data model, structures can contain **arrays** (collections of other data types). These arrays can contain primitive data types or collections of other structures.

TopLink maps arrays of primitive data types with the `ArrayMapping` class. An array mapping maps to object-relational array data types (the `Array` type in JDBC and the `VARRAY` type in Oracle Database). To map a collection of aggregate structures, use an object array mapping (see ["Object-Relational Object Array Mapping"](#) on page 49-2).

The object-relational database stores the arrays with their parent structure in the same table. To store information in a separate table from the parent structure's table, use a nested table mapping (see ["Object-Relational Nested Table Mapping"](#) on page 49-3).

All elements in the array must be the same data type. The number of elements in an array controls the size of the array. An Oracle Database allows arrays of variable sizes (the `VARRAY` type).

See ["Configuring an Object-Relational Array Mapping"](#) on page 53-1 for more information.

Object-Relational Object Array Mapping

In an object-relational data model, structures can contain *arrays* (collections of other data types). These arrays can contain primitive data types or collections of other structures.

TopLink maps arrays of structures with the `ObjectArrayMapping` class. An object array mapping defines a collection-aggregated relationship, in which the target objects share the same row as the source object.

You must associate this mapping to an attribute in the parent class.

See ["Configuring an Object-Relational Object Array Mapping"](#) on page 54-1 for more information.

Object-Relational Nested Table Mapping

Nested table types model an unordered set of elements. These elements may be built-in or user-defined types. You can view a nested table as a single-column table or, if the nested table is an object type, as a multicolumn table (with a column for each attribute of the object type).

TopLink maps nested tables with the `NestedTableMapping` class. It represents a collection of object references in Java. Because of the object-relational support of the database, nested table mapping does not require foreign key information (as with a one-to-many mapping) or a relational table (as with a many-to-many mapping).

Typically, nested tables represent a one-to-many or many-to-many relationship of references to another independent structure. They support querying and joining better than the `VARRAY` types that are in-lined to the parent table. TopLink supports mapping a nested table of `REF` types only. TopLink does not support nested tables of basic or other structured data types—use array (see ["Object-Relational Array Mapping"](#) on page 49-2) or object array (see ["Object-Relational Object Array Mapping"](#) on page 49-2) mappings instead.

See ["Configuring an Object-Relational Nested Table Mapping"](#) on page 55-1 for more information.

Configuring an Object-Relational Mapping

This chapter describes how to configure an object-relational mapping.

[Table 50–1](#) lists the types of object-relational mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

Table 50–1 *Configuring Object-Relational Mappings*

If you are creating...	See Also...
Object-Relational Structure Mapping	Chapter 51, "Configuring an Object-Relational Structure Mapping"
Object-Relational Reference Mapping	Chapter 52, "Configuring an Object-Relational Reference Mapping"
Object-Relational Array Mapping	Chapter 53, "Configuring an Object-Relational Array Mapping"
Object-Relational Object Array Mapping	Chapter 54, "Configuring an Object-Relational Object Array Mapping"
Object-Relational Nested Table Mapping	Chapter 55, "Configuring an Object-Relational Nested Table Mapping"

[Table 50–2](#) lists the configurable options shared by two or more object-relational mapping types.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Understanding Object-Relational Mappings"](#) on page 49-1

Configuring Common Object-Relational Mapping Options

[Table 50–2](#) lists the configurable options shared by two or more object-relational mapping types. In addition to the configurable options described here, you must also configure the options described for the specific [Object-Relational Mapping Types](#), as shown in [Table 50–1](#).

Table 50–2 *Common Options for Object-Relational Mappings*

Option	Type	TopLink Workbench	Java
"Configuring Reference Class" on page 50-2	Basic		✓
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓

Table 50–2 (Cont.) Common Options for Object-Relational Mappings

Option	Type	TopLink Workbench	Java
"Configuring Structure Name" on page 50-4	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Basic		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring Indirection" on page 35-3	Advanced		✓
"Configuring Container Policy" on page 35-26	Advanced		✓

Configuring Reference Class

When mapping an attribute that involves a relationship to another class, you must specify the reference class—the Java class to which the mapped attribute refers.

[Table 50–3](#) summarizes which object-relational mappings support this option.

Table 50–3 Mapping Support for Reference Class

Mapping	Using TopLink Workbench	Using Java
Object-Relational Structure Mapping		✓
Object-Relational Reference Mapping		✓
Object-Relational Array Mapping		
Object-Relational Object Array Mapping		✓
Object-Relational Nested Table Mapping		✓

Using Java

Use `oracle.toplink.mappings.ForeignReferenceMapping.setReferenceClass` to specify the target class of the attribute being mapped.

[Example 50–1](#) shows how to use this method with a `ReferenceMapping` that maps the `manager` attribute of the `Employee` class.

Example 50–1 Configuring Reference Class in Java

```
ReferenceMapping managerMapping = new new ReferenceMapping();
managerMapping.setReferenceClass("Employee.class");
managerMapping.setAttributeName("manager");
```

Configuring Attribute Name

All object-relational mappings map an attribute in a Java object to field in the database. The attribute name is the name of the attribute being mapped. The name is as specified in the reference class (see ["Configuring Reference Class"](#) on page 50-2).

[Table 50–4](#) summarizes which object-relational mappings support this option.

Table 50–4 Mapping Support for Attribute Name

Mapping	Using TopLink Workbench	Using Java
Object-Relational Structure Mapping		✓

Table 50–4 (Cont.) Mapping Support for Attribute Name

Mapping	Using TopLink Workbench	Using Java
Object-Relational Reference Mapping		✓
Object-Relational Array Mapping		✓
Object-Relational Object Array Mapping		✓
Object-Relational Nested Table Mapping		✓

Using Java

Use `oracle.toplink.mappings.DatabaseMapping.setAttributeName` to specify the name of the attribute being mapped.

[Example 50–2](#) shows how to use this method with a `ReferenceMapping` that maps the `manager` attribute of the `Employee` class.

Example 50–2 Configuring Attribute Name in Java

```
ReferenceMapping managerMapping = new ReferenceMapping();
managerMapping.setReferenceClass("Employee.class");
managerMapping.setAttributeName("manager");
```

Configuring Field Name

All object-relational mappings require the name of database field to which their specified attribute is mapped. This field name can be the column name of a database table or the name of a field in an object type created on the database.

[Table 50–5](#) summarizes which object-relational mappings support this option.

Table 50–5 Mapping Support for Field Name

Mapping	Using TopLink Workbench	Using Java
Object-Relational Structure Mapping		✓
Object-Relational Reference Mapping		✓
Object-Relational Array Mapping		✓
Object-Relational Object Array Mapping		✓
Object-Relational Nested Table Mapping		✓

Using Java

Use the object-relational mapping method `setFieldName` to specify the database field to which the attribute is mapped.

[Example 50–3](#) shows how to use this method with an `ObjectArrayMapping` that maps the `Employee` class attribute `phone` to database field name `PHONE_NUMBER`.

Example 50–3 Configuring Field Name in Java

```
ObjectArrayMapping phonesMapping = new ObjectArrayMapping();
managerMapping.setReferenceClass("Employee.class");
phonesMapping.setAttributeName("phone");
phonesMapping.setFieldName("PHONE_NUMBER");
```

Configuring Structure Name

Certain object-relational mappings require the specification of the data type or structure name of the field being mapped. The structure name is the name of the array or table type that defines the field.

Table 50–6 summarizes which object-relational mappings support this option.

Table 50–6 Mapping Support for Structure Name

Mapping	Using TopLink Workbench	Using Java
Object-Relational Structure Mapping		
Object-Relational Reference Mapping		
Object-Relational Array Mapping		✓
Object-Relational Object Array Mapping		✓
Object-Relational Nested Table Mapping		✓

Using Java

Use the object-relational mapping method `setStructureName` to specify the structure of the attribute being mapped.

Example 50–4 shows how to use this method with an `ObjectArrayMapping` that maps the `Employee` class attribute `phones` to database field name `PHONE_NUMBERS` of type `PHONE_ARRAY_TYPE`.

Example 50–4 Configuring Structure Name in Java

```
ObjectArrayMapping phonesMapping = new ObjectArrayMapping();
managerMapping.setReferenceClass("Employee.class");
phonesMapping.setAttributeName("phones");
phonesMapping.setFieldName("PHONE_NUMBERS");
phonesMapping.setStructureName("PHONE_ARRAY_TYPE");
```

Configuring an Object-Relational Structure Mapping

This chapter describes the various components that you must configure in order to use an object-relational structure mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Object-Relational Structure Mapping"](#) on page 49-2

Object-Relational Structure Mapping Configuration Overview

[Table 51-1](#) lists the configurable options for an object-relational structure mapping.

Table 51-1 *Configurable Options for Object-Relational Structure Mapping*

Option	Type	TopLink Workbench	Java
"Configuring Reference Class" on page 50-2	Basic		✓
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Advanced		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓

Configuring an Object-Relational Reference Mapping

This chapter describes the various components that you must configure in order to use an object-relational reference mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Object-Relational Reference Mapping"](#) on page 49-2

Object-Relational Reference Mapping Configuration Overview

[Table 52–1](#) lists the configurable options for an object-relational reference mapping.

Table 52–1 Configurable Options for Object-Relational Reference Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Class" on page 50-2	Basic		✓
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Advanced		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring Private or Independent Relationships" on page 35-16	Advanced		✓
"Configuring Indirection" on page 35-3	Advanced		✓

Configuring an Object-Relational Array Mapping

This chapter describes the various components that you must configure in order to use an object-relational array mapping.

Note: To map a collection of aggregate structures, use an object-relational object array mapping (see ["Object-Relational Object Array Mapping"](#) on page 49-2). To store information in a separate table from the parent structure's table, use an object-relational nested table mapping (see ["Object-Relational Nested Table Mapping"](#) on page 49-3).

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Object-Relational Array Mapping"](#) on page 49-2

Object-Relational Array Mapping Configuration Overview

[Table 53–1](#) lists the configurable options for an object-relational array mapping.

Table 53–1 Configurable Options for Object-Relational Array Mapping

Option	Type	TopLink Workbench	Java
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓
"Configuring Structure Name" on page 50-4	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Basic		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring a Serialized Object Converter" on page 35-18	Advanced		✓
"Configuring a Type Conversion Converter" on page 35-20	Advanced		✓
"Configuring an Object Type Converter" on page 35-22	Advanced		✓
"Configuring Container Policy" on page 35-26	Advanced		✓

Configuring an Object-Relational Object Array Mapping

This chapter describes the various components that you must configure in order to use an object-relational object array mapping.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Object-Relational Object Array Mapping"](#) on page 49-2

Object-Relational Object Array Mapping Configuration Overview

[Table 54–1](#) lists the configurable options for an object-relational object array mapping.

Table 54–1 Configurable Options for Object-Relational Object Array Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Class" on page 50-2	Basic		✓
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓
"Configuring Structure Name" on page 50-4	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Basic		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring Container Policy" on page 35-26	Advanced		✓

Configuring an Object-Relational Nested Table Mapping

This chapter describes the various components that you must configure in order to use an object-relational nested table mapping.

Note: For an equivalent mapping for basic or other structured data types, use object-relational array (see ["Object-Relational Array Mapping"](#) on page 49-2) or object array (see ["Object-Relational Object Array Mapping"](#) on page 49-2) mappings.

For more information, see the following:

- ["Mapping Creation Overview"](#) on page 34-1
- ["Object-Relational Nested Table Mapping"](#) on page 49-3

Object-Relational Nested Table Mapping Configuration Overview

[Table 55–1](#) lists the configurable options for an object-relational nested table mapping.

Table 55–1 Configurable Options for Object-Relational Nested Table Mapping

Option	Type	TopLink Workbench	Java
"Configuring Reference Class" on page 50-2	Basic		✓
"Configuring Attribute Name" on page 50-2	Basic		✓
"Configuring Field Name" on page 50-3	Basic		✓
"Configuring Structure Name" on page 50-4	Basic		✓
"Configuring Read-Only Mappings" on page 35-2	Basic		✓
"Configuring Method Accessing" on page 35-14	Advanced		✓
"Configuring Indirection" on page 35-3	Advanced		✓
"Configuring Container Policy" on page 35-26	Advanced		✓

Glossary

This glossary contains terms and abbreviations that you should be familiar with when using Oracle TopLink.

attribute

A variable of a class or object. In TopLink, an *attribute* describes all instance variables of a class. Every attribute contains a single mapping. Attributes store primitive data such as integers, and simple Java types such as `String` or `Date`.

authentication

The means by which a data source validates a user's identity and determines whether or not the user has sufficient privileges to perform a given action.

bean class

The implementation of the bean. The bean is accessed from the client using the home and remote interfaces.

bean-managed persistence (BMP)

A scheme for persisting entity beans that requires the developer to manually code the methods that perform the persistence.

Compare to [container-managed persistence \(CMP\)](#).

branch class

Has a persistent superclass and also has subclasses. By default, queries performed on the branch class return instances of the branch class and any of its subclasses. However, the branch class can be configured so that queries on it return only instances of itself without instances of its subclasses.

Compare to [leaf class](#).

class

A category of objects. Classes allow data and method to be grouped together.

class indicator field

A field in the table of the root class that indicates which subclass should be instantiated

client session broker

A collection of client sessions, one from each server session associated with the session broker.

connection pool

A collection of reusable connections to a single data source.

container-managed persistence (CMP)

A scheme for persisting entity beans that uses information supplied by the developer or deployer to perform the persistence

Compare to [bean-managed persistence \(BMP\)](#).

custom SQL

Refers to any non-TopLink-generated SQL used through TopLink. This includes hard-coded SQL and stored procedure calls.

data definition language (DDL)

The data definition part of the structured query language (SQL). TopLink Workbench can generate DDL creation scripts that can be used to create tables on the desired database.

database session

A database session provides a client application with a single data store connection, for simple, standalone applications in which a single connection services all data store requests for one user.

default mapping

A relational persistence framework term that refers to making the framework automatically generate the object descriptor metadata (including such things as mappings, login data, database platform, locking, and foreign keys). Default mapping is available for TopLink projects using EJB 2.0 CMP applications with OC4J.

dependent class path (IBM WebSphere)

Location where nonbean classes are specified. TopLink requires that the bean classes be included here since they are referenced by the project.

deployment descriptor

A set of XML files that provide the additional required information to install an EJB within its server. Typically, this includes security, transaction, relationship, and persistence information.

Compare with TopLink [descriptors](#).

descriptors

An TopLink object that describes how an object's attributes and relationships are to be represented in relational database table(s). An "TopLink descriptor" is not the same as a [deployment descriptor](#), although it plays a similar role.

direct access

By default, TopLink accesses public attributes directly when writing the attributes of the object to the database or reading the attributes of the object from the database.

Compare to [method access](#).

direct mapping

There are two basic ways of storing object attributes directly in a table:

-
- The information can be stored directly if the attribute type is comparable to a database type.
 - If there is no database primitive type that is logically comparable to the attribute's type, it must be transformed on its way to and from the database

TopLink provides five classes of direct mappings.

Compare to [relationship mapping](#).

Enterprise Java Beans (EJB)

EJB are server-side domain objects that fit into a standard component-based architecture for building enterprise applications with Java. They are objects that become distributed, transactional, and secure components. TopLink Workbench uses three types of EJB: [session beans](#), [entity beans](#), and [message-driven beans](#).

expressions

The TopLink equivalent of an SQL conditional clause. TopLink expressions are specified using the `Expression` and `ExpressionBuilder` classes.

entity beans

EJB that represent a persistent data object. TopLink uses two schemes for persisting entity beans: [bean-managed persistence \(BMP\)](#) and [container-managed persistence \(CMP\)](#).

fetch group

A performance enhancement that defines a subset of object attributes to be loaded initially and ensures that all other attributes are loaded on demand.

hub

A common connection point for devices in a network.

identity map

Used to cache objects for performance and to maintain object identity.

See also [object identity](#).

independent relationship

A relationship in which the source and target are public objects that exist independently; the destruction of one object does not necessarily imply the destruction of the other.

Compare to [private relationship](#).

indirection

An indirection object is one that acts as a stand-in for another object. In TopLink, indirection is implemented through value holders, which delay database access through acting as substitute for any object relationships.

inheritance

Describes how a child class inherits the characteristics of its parent class. TopLink supports multiple approaches to database implementations that preserve the inheritance relationship.

in-memory query

A query that is run against the shared session cache.

instantiate

Create an instance of a Java class.

J2C

The J2EE Connector architecture (J2C) adapter is a way to persist Java objects to a nonrelational data source, such as XML.

J2SE

The Java 2 Platform, Standard Edition (J2SE) is the core Java technology platform. It provides software compilers, tools, runtimes, and APIs for writing, deploying, and running applets and applications in Java.

J2EE

The Java 2 Platform, Enterprise Edition (J2EE) is an environment for developing and deploying enterprise applications. J2EE includes a set of services, APIs, and protocols for developing multitiered web-based applications.

J2EE Containers

A J2EE container is a run-time environment for Enterprise Java Beans (EJB) that includes such basic functions as security, life cycle management, transaction management, and deployment services. J2EE containers are usually provided by a J2EE server, such as Oracle Containers for J2EE.

Java Messaging Service (JMS)

The JMS API is a protocol for communication that provides asynchronous communication between components in a distributed computing environment.

Java Naming and Directory Interface (JNDI)

The JDBC specification recommends using a JNDI naming service to acquire a connection to a database. TopLink supports acquiring a database connection in this fashion. To take advantage of this feature, construct and configure an instance of `oracle.toplink.jndi.JNDIConnector` and pass it to the project login object using the `setConnector` method.

Java Transaction API (JTA)

The Java Transaction API (JTA) specifies the interfaces between a transaction manager, a resource manager, an application server, and transactional applications involved in a distributed transaction system.

leaf class

Has a persistent superclass in the hierarchy but does not have subclasses; queries performed on the leaf class can return only instances of the leaf class.

Compare to [branch class](#).

locking policy

A mechanism that ensures one user does not overwrite another users's work. TopLink descriptors support optimistic and pessimistic locking policies.

mappings

Describe how individual Java objects and attributes relate to a data source.

message-driven beans

An EJB that processes asynchronous Java Messaging Service (JMS) messages. For TopLink clients, a message-driven bean is simply a JMS consumer with no conversational state and no home or remote interfaces.

method access

The application registers accessor methods for the attribute.

Compare to [direct access](#).

named query

A TopLink query that is created and stored, by name, in a session for later retrieval and execution

object identity

Ensures that each object is represented by one and only one instance in the application; that is, multiple retrievals of the same object return references to the same object instance, not multiple copies of the same object. Violating object identity can corrupt the object model.

See also [identity map](#).

optimistic locking

Also known as write locking; allows unlimited read access to objects. A client can write an object to the database only if the object has not changed since it was last read.

Compare to [pessimistic locking](#).

packet

A piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in addition to the data.

packet time-to-live

A number of hops that session data packets can take before expiring. The default is 2.

See also [packet](#).

persist

In object technology, the storage of an Java object by a data source.

pessimistic locking

Objects are locked before they are edited, which ensures that only one client is editing the object at any given time.

Compare to [optimistic locking](#).

primary key

A field (or combination of fields) that uniquely identifies a record in the data source.

private relationship

A relationship in which the target object is considered to be a private component of the source object; the target object cannot exist without the source and is accessible only through the source object; furthermore, if the source object is destroyed, the target object is destroyed as well.

Compare to [independent relationship](#).

query manager

An object, owned by a descriptor, that controls the way the descriptor accesses the database. The query manager generates its own default SQL to access the database in a transparent manner.

query optimization

TopLink supports two forms of query optimization: joining and batch reading. Their purpose is to optimize database access through reducing the number of database calls required to read a group of objects.

relationship

In TopLink, a reference between two TopLink-enabled objects.

relationship mapping

Persistent objects use relationship mappings to store references to instances of other persistent classes. The appropriate mapping class is chosen primarily by the cardinality of the relationship. TopLink provides five classes of relationship mappings.

Compare to [direct mapping](#).

Remote Method Invocation (RMI)

A set of protocols that enable Java objects to communicate remotely with other Java objects.

remote session

A remote session is a client-side session that communicates over RMI with a corresponding client session and server session on the server side. Remote sessions handle object identity and marshalling and unmarshalling between client side and server side.

service channel

A name of the TopLink coordinated cache channel to which sessions subscribe in order to participate in the same coordinated cache.

session beans

EJB that represent a business operation, task, or process. TopLink can use session beans to make the regular Java objects they access persistent, or to wrap other legacy applications.

stale data

An artifact of caching, in which an object in the cache is not the most recent version committed to the data source.

TopLink session broker

A mechanism that enables client applications to transparently access multiple databases through a single TopLink session.

unit of work

A transactional TopLink session that allows for a transaction to occur at the object level not only the database level. Changes to objects are not visible globally until the unit of work is committed.

value holder

A wrapping object used by TopLink to delay database access.

A

access

- data access, 84-1
- direct, 35-32
- modifiers, 4-42
- optimizing data access, 11-14
- remote sessions, 75-31

access method

- direct, 35-14
- generating, 4-30
- mappings, 35-15
- method, 35-14
- specifying, 35-32

access modifiers, classes, 4-42

acquiring

- client sessions, 78-6
- sessions, at runtime, 75-5
- unit of work, 101-1

activating descriptors, 4-10

Add Named Query dialog, 28-12, 28-18, 28-20

Add New Class dialog, 4-40, 4-53

Add New Table button, 4-22

Add or Refresh Class button, 4-50

addConstraingDependencies(), 102-16

address

- multicast group, 91-4
- multicast port, 91-5

Add/Update Existing Tables from Database

- button, 4-23

advanced properties for descriptors, 22-7

After Load tab, 28-79

aggregate collection relational mappings

- and EJB, 36-11
- configuring, 44-1
- understanding, 36-10

aggregate descriptors

- about, 26-5
- aggregate object mapping, 26-5
- EIS projects, 26-8
- EJB 3.0, 26-6
- inheritance, 26-17
- relational projects, 26-5
- XML projects, 26-8

aggregate object relational mapping

- aggregate descriptors, and, 26-5

aggregate object relational mappings

- configuring, 47-1
- understanding, 36-12

aggregation, isolated client sessions, 75-24

AllFieldsLockingPolicy, 26-20

allows none, 28-18, 97-14

allows null, 28-18, 97-14

amending descriptors, 2-20, 26-5, 28-78

- see also* after load

announcement delay, 91-11

Ant, integrating with Oracle TopLink

- Workbench, 4-53

any collection XML mappings

- configuring, 72-1
- understanding, 65-29

any object XML mappings

- configuring, 71-1
- understanding, 65-27

application development

- deploying, 10-1
- mapping, 2-20
- querying, 2-13, 74-2
- troubleshooting, 15-1

application layer, remote sessions, 75-30

application servers

- EJB support, customizing, 7-25
- integrating with Oracle TopLink, 7-1
- logging, 75-9
- optimization, 11-30
- setter parameter type checking, 7-25
- single-object finder return type checking, 7-25, 7-26
- software requirements, 7-2
- target platforms, 2-5
- unknown primary key class support, 7-25

architectures

- application, 1-4
- BMP, 1-5, 2-33
- cache, 90-1
- choosing, 2-3
- CMP, 1-5, 2-27
- EIS, 2-4, 56-5
- EJB entity beans, 1-5, 2-27, 2-33
- EJB session bean facade, 1-5, 2-24
- locking, 2-10
- optimistic locking, 2-10

- Oracle TopLink, 1-1
- pessimistic locking, 2-10
- selecting, 2-5
- session brokers, 75-26
- sessions, 75-2
- three-tier, 1-4, 2-21
- two-tier, 1-6, 2-23
- unit of work, 100-1
- web services, 1-5, 2-36
- arguments, binding in query, 29-6
- array
 - dimensionality, 4-45, 4-49
 - object-relational mappings, 49-2
- AsOfClause, 99-2
- asynchronous change propagation, 91-2
- AttributeChangeTrackingPolicy
 - about, 100-8
 - OC4J CMP integration, 100-8
 - OC4J EJB 1.x CMP integration, 100-8
 - OC4J EJB 2.x CMP integration, 100-8
 - OC4J EJB 3.0 CMP integration, 100-8
 - other application servers, 100-8
- attributes
 - adding to descriptors, 4-44
 - array dimensionality, 4-45, 4-49
 - changes, tracking, 28-72
 - final, 4-45, 4-48
 - in TopLink Workbench, Navigator window, 4-9
 - lazy loading. *see* fetch groups
 - static, 4-45, 4-48
 - transforming, 35-29, 35-31, 36-15
 - transient, 4-45, 4-48
 - unmapping, 34-6
 - volatile, 4-45, 4-48
- Attributes tab, 4-44, 4-48
- Attunity Connect platform, 84-3
- auditing
 - authentication, 84-6
 - unit of work, 102-20
- authentication
 - about, 84-5
 - auditing, 84-6
 - proxy authentication, 84-5
 - simple JDBC authentication, 84-5
 - three-tier architecture, 84-5
 - two-tier architecture, 84-5
- Automap, 34-2
- automapping descriptors
 - about, 34-2
 - see also* mappings
- automatic table generation
 - about, 33-4
 - configuring, 8-12

B

- Base64 encoded strings, 35-25
- batch options
 - mappings, 37-7
 - writing, 11-15

- batch reading
 - in query objects, 98-10
 - read optimization, and, 11-19
- batch writing
 - about, 11-15, 11-26, 86-11
 - dynamic, 11-15
 - dynamic, `setMaxBatchWritingSize()`, 11-15
 - MySQL4 platform, 86-9
 - non-parameterized, 11-15
 - parameterized, 11-15
 - `setMaxBatchWritingSize()`, 11-15
- BEA WebLogic
 - deploying to, 10-2
 - deployment exceptions, 15-3
 - modifying persistence descriptor, 8-13
 - setting classpath, 7-15
 - setting shared library, 7-15
 - transport layer, 75-31
 - using a security manager, 7-20
- @Bean fetch=lazy, 33-9
- beans
 - session beans, 2-25
 - stateful beans, 2-25
 - stateless beans, 2-25
- bidirectional relationships
 - about, 36-2
 - generating, 4-31
 - in one-to-one mappings, 35-34
 - target keys, 36-5
 - with indirection, 35-35
- `bindAllParameters()` method, 23-7
- bindings
 - arguments, 29-6
 - input paramters, 98-19
 - JAXB, 20-10
 - LOB mappings, 34-3
 - see* parameter binding
- BLOB
 - mapping to, 34-2
- BMP
 - and EJB 1.1, 2-34
 - and EJB 2.0, 2-34
 - and TopLink, 2-33
 - deployment files, 8-9
 - descriptors, 26-3, 28-44
 - packaging for deployment, 9-5
- boolean logic in expressions, 97-3
- branch classes, 26-12, 26-13
- buttons. *see* toolbars
- bye array binding, LOB mappings, 34-3
- Byte array Base64, 35-25

C

- cache
 - about, 2-12, 2-18, 90-1
 - architecture, 90-1
 - configuring, 90-6, 99-21
 - coordination, 90-9, 91-1, 92-1, 93-1
 - descriptor level, 28-35

- disabling during read query, 96-34
- distributed, 90-9
- expiration, 22-19, 28-40, 99-22
- expression limitations, 96-31
- identity maps, using, 75-29
- in-memory queries, 96-30, 96-31
- internal query object cache, 96-36, 99-21, 99-22
- invalidation, 22-19, 28-40, 90-7, 90-8
- isolated client sessions, 75-24
- isolation, 90-6, 90-9
- object cache, 96-36
- object cascading refresh, 96-35
- object refresh, 96-35
- optimizing, 11-13
- project level, 22-13
- queries, 90-6, 96-29
- query cache, 96-36
- `readObject` method, and, 98-2
- refreshing, 28-27, 90-8, 96-35
- restrictions, 96-36
- service channel, 91-3
- sessions, 75-3, 75-32, 90-2
- stale data, 90-6
- storing query results, 96-36, 99-21
- type and size, 22-13, 28-35
- unit of work cache, 90-2
- cache coordination
 - about, 90-9
 - application server clustering, and, 7-4
 - avoiding stale data, 90-7
 - CMP projects, 8-11
 - EJB Entity Beans with BMP architecture, and, 2-36
 - explicit query refreshes, 90-7
 - JMS, 92-1
 - `orion-ejb-jar.xml`, 8-11
 - packet time-to-live, 91-15
 - permissions, 7-23
 - RMI, 93-1
- cache invalidation, avoiding stale data, 90-7
- cache synchronization. *see* cache coordination
- `cacheAllStatements()`, 23-7
- `cacheQueryResults()`, 28-25
- cache-synchronization property, 8-11
- Caching tab, 28-35, 28-38, 28-39, 28-42
- call
 - call queries, 96-16
 - EIS, 98-24
 - EJBQLCall, 98-23
 - SQLCall, 98-18
 - StoredFunctionCall, 98-22
 - StoredProcedureCall, 98-20
- Call object, queries, 96-3
- cascading
 - object refresh, 96-35
 - optimistic version locking, 26-18
 - write queries, compared to non-cascading, 96-14, 98-13
- catalog, database, 4-22
- `catchExcpetions()`, 77-19
- change policy
 - about, 28-70
 - attribute change tracking, configuring, 28-72
 - deferred change detection, configuring, 28-71
 - empty transaction, 100-6
 - object change tracking, configuring, 28-71
 - unit of work, 100-6
- change tracking
 - attribute, configuring, 28-72
 - deferred, configuring, 28-71
 - object, configuring, 28-71
- changed items, displaying in TopLink Workbench Navigator window, 4-9
- `ChangedFieldsLockingPolicy`, 26-20
- changing package names, 4-52
- `checkDatabase()`, 77-18
- checking in/out projects, 5-3
- `checkInstantiationPolicy()`, 77-19
- Choose a Schema Context dialog box, 31-3, 32-2
- Choose Query Key dialog box, 97-14
- Choose Relationships to Generate dialog box, 4-30, 4-31
- Choose Root Element dialog box, 31-4, 32-5
- class extraction method
 - about, 26-14
 - inheritance, 26-15
- class indicator
 - about, 26-13
 - class extraction method, 26-14
 - class indicator field, 26-13, 41-1
- class loader
 - host application, 2-5
 - loading session, 78-4
- class modifiers, 4-42
- Class tab, 4-41, 4-42, 4-43
- classes
 - access modifiers, 4-42
 - adding and refreshing, 4-50
 - branch, 26-13
 - creating, 4-40, 4-52
 - `CursoredStream`, optimizing, 99-18
 - Database Exception, 98-24
 - `DatabaseMapping`, 33-25
 - default null values, 35-36
 - `DeleteObjectQuery`, 98-13
 - `ExpressionBuilder`, 97-15
 - generating from database, 4-30
 - `InsertObjectQuery`, 98-13
 - `InsertObjectQuery`, 100-12
 - interfaces, 4-43
 - leaf, 26-13
 - merging files, 5-5
 - methods, adding, 4-47
 - naming, 4-41
 - non-descriptor classes, 4-51
 - object model, 2-16
 - Performance Profiler, 11-3
 - persistent requirements, 2-11
 - preferences, 4-16
 - refreshing, 4-50

- removing, 4-51
- root, 26-12
- troubleshooting, 14-22
- tutorial, 16-2
- unit of work, 100-12
- UpdateObjectQuery, 98-13, 98-14
- ValueHolderInterface, 2-11, 33-7, 36-9
- VariableOneToOneMapping, 36-6
- see also* specific class name
- classpath
 - adding, 22-3
 - BEA WebLogic, 7-15
 - configuring, 4-2, 7-15, 7-21
 - connector.jar, 4-2
 - custom Collection class, 4-3
 - DRIVER_CLASSPATH, Oracle TopLink Workbench, 4-2
 - IBM WebSphere, 7-21
 - J2C adapter, 4-2
 - JDBC driver, 4-2, 4-23
 - JDBC_CLASSPATH, 4-2
 - Oracle TopLink Workbench DRIVER_CLASSPATH, 4-2
 - relative, 22-3
 - setting for BEA WebLogic, 7-15
 - troubleshooting, 14-22
 - xdb.jar file, 4-3
- class-table relationships, 16-6
- client sessions
 - about, 75-1, 75-13, 75-25
 - acquiring, 78-6
 - configuration, 82-1
 - example, 75-15
 - shared resources, 75-15
- client-controlled transactions, 102-24
- client-server architecture. *See* two-tier
- CLOB
 - mapping to, 34-2
- clones
 - copying methods, 28-69
 - merging changes, 102-12
 - post-commit, avoiding, 102-30
 - unit of work, 100-2, 100-9
- Cloudscape platform, 84-3
- clustering, integrating TopLink with, 7-4
- CMP
 - and EJB 1.1, 2-27
 - and EJB 2.x, 2-27
 - and EJB 3.0, 2-27
 - and TopLink, 2-27
 - CMPPolicy, 26-3, 28-44
 - deploying, 8-9, 10-2
 - descriptors, inheritance, 26-4
 - external transactions, 100-3
 - isolated client sessions, 75-24
 - local transaction, 102-25
 - non-deferred write, 102-25
 - OC4J persistence, 7-5
 - packaging for deployment, 9-4
 - setter parameter type checking, 7-25
 - single-object finder return type checking, 7-26
 - transaction attribute, 102-24
 - unit of work, 100-3, 102-24
 - unknown primary key class support, 7-25
- code generation, optimizing, 11-8
- collapsing items in Navigator window, 4-9
- collection class
 - specifying, 98-9
 - specifying in query objects, 98-9
- collections
 - persistent requirements for mappings, 2-11
 - query results, 96-8
- comments
 - descriptors, 28-8, 28-9
 - mappings, 35-18
 - projects, 22-21
- commit
 - and Java Transaction API, 100-10
 - failure, resuming after, 102-14
 - resuming unit of work after, 102-14
- Communication Exceptions, 13-64
- composite collection EIS mappings
 - configuring, 61-1
 - example, 56-8
 - understanding, 56-7
- composite collection XML mappings
 - configuration, 70-1
 - configuring, 70-1
 - understanding, 65-25
- composite descriptors
 - about, 26-5
 - composite object mapping, 26-8
 - EIS projects, 26-8
 - XML projects, 26-8
- composite direct collection EIS mappings
 - configuring, 59-1
 - understanding, 56-6
- composite direct collection XML mappings
 - configuring, 68-1
 - understanding, 65-14
- composite EIS descriptors, 27-5
- composite object EIS mappings
 - composite descriptors, and, 26-8
 - configuring, 60-1
 - understanding, 56-7
- composite object mappings
 - composite descriptors, and, 26-8
- composite object XML mappings
 - configuration, 69-1
 - configuring, 69-1
 - understanding, 65-21
- composite primary key, 36-9
- concrete class. *see* container policy
- concurrency
 - about, 2-17
 - exceptions, 13-25
 - server session, 75-18
- Concurrency Exceptions, 13-25
- configurations
 - about, 76-2

- creating, 76-2
- development environment, 5-1
- new, 76-2
- Oracle JDeveloper, 5-1
- session, 76-2, 76-4, 76-6, 76-9
- conforming
 - about, 102-8
 - alternatives, UnitOfWork method
 - writeChanges, 102-11
 - alternatives, UnitOfWork properties, 102-12
 - descriptors, and, 102-11
 - in-memory queries, 102-9
 - queries, alternatives to, 102-11
- ConnectBy, 99-7
- connection policy
 - configuring, 77-19
 - exclusive connections, 77-19
 - lazy connection acquisition, 77-20
- connection pool
 - about, 84-7
 - connection count, 89-1
 - ConnectionPolicy, 75-20
 - external, 84-8
 - internal, 84-7, 84-8, 84-9
 - lazy connection allocation, 75-18, 78-9
 - named, 84-9
 - parameter binding, 11-16
 - prepared statement caching, 11-16
 - read, 84-8
 - sequence, 20-5, 84-8
 - server session, 75-18
 - sessions, and, 75-3
 - size, 89-1
 - write, 84-8
- Connection Specifications tab, 24-3
- Connection tab, 24-3
- connections
 - about, 84-6
 - connection pool, 84-7
 - exclusive write connection, 77-19
 - lazy acquisition, 77-20
 - reading through the write connection, 102-27
- connector.jar, 4-2, 20-7
- container configuration file, 8-5
- container policy
 - about, 35-26
 - custom Collection class, 4-3
 - sorting, in memory, 35-26
- container-controlled transactions, 102-24
- context
 - JAXB path, 20-14
 - menus, 4-5
 - schema, 31-2, 32-2
- Context.SECURITY_CREDENTIALS, 91-9
- Context.SECURITY_PRINCIPAL, 91-9
- Conversion Exception, 13-26
- Converter tab
 - object type mappings, 35-22
- converters
 - custom, 36-3

- object type, 35-22
- coordinated announcement delay, 91-11
- coordinated cache, 8-11
 - configuring, 91-1
 - naming service, 91-7
 - service channel, 91-3
- copy policy
 - about, 28-69
 - method, 28-69
 - setting, 28-70
- copying project objects, 5-6
- Copying tab, 28-70
- CORBA
 - Oracle TopLink transport layer support, 75-31
 - Transaction Service *see* OTS
- Create New Project button, 21-2
- Create New Project dialog box, 21-2
- Create New Session dialog, 76-4, 76-6, 76-8
- Create Project from JAXB dialog, 21-6
- Create Project from OC4J dialog, 7-9
- creating
 - configurations, 76-2
 - expressions, 97-12
 - sessions, 76-4, 76-6, 76-8, 88-1
- Crimson XML parser, 7-3
- cursored streams
 - example, 99-18
 - optimizing, 99-18
 - remote sessions, 75-32
 - usage example, 75-32
- cursors
 - as query results, 96-8
 - traversing scrollable, 99-16
- Custom Calls tab, 31-6
- custom SQL
 - Custom SQL tab, 29-7
 - unit of work, 102-15
- customization
 - about, 12-1
 - data types, 12-1
 - EIS, 12-1
 - mapping extensions, 12-1
 - overview, 2-15
 - XML, 12-1

D

- data access
 - about, 84-1
 - authentication, 84-5
 - connection pool, 84-7
 - connections, 84-6
 - optimizing, 11-14, 11-15
 - platforms, 84-3
- data level queries
 - example, 97-12
 - in expressions, 97-11
- data source platform
 - about, 84-3
 - Attunity Connect database, 84-3

- Cloudscape database, 84-3
- databases, 84-3
- DB2 database, 84-3
- EIS, 84-4
- HSQL database, 84-3
- Informix database, 84-3
- J2C adapter, 84-4
- JDBC drivers, 84-3
- Microsoft Access database, 84-3
- MySQL4 database, 84-3
- Oracle database, 84-3
- Oracle8 database, 84-3
- Oracle9 database, 84-3
- PointBase database, 84-3
- SQLAnywhere database, 84-3
- SQLServer database, 84-3
- SybasePlatform database, 84-3
- data sources
 - configuring, 85-1
 - nontransactional, 84-1
 - transactional, 84-1
 - troubleshooting, 14-23
- Database Exceptions, 13-27, 98-24
- database fields, configuring, 37-2
- database functions, in expressions, 97-3
- database login
 - parameter binding, 86-9
 - prepared statement caching, 86-9
- Database Preferences, 4-17
- database queries
 - about, 96-10, 98-4
 - fetch groups, 96-12
 - join reading, 96-12
 - object level modify query, 96-13, 96-14
 - object level read query, 96-11, 96-13
 - partial object query, 96-11
 - read all query, 96-11
 - read object query, 96-11
 - report query, 96-15
- database schema
 - tables, 4-22
 - tutorial, 16-4
- database sessions
 - about, 75-2, 75-28
 - cache, 75-32
 - configuration, 83-1
 - creating, 76-8
- database tables
 - about, 4-21
 - adding to database, 4-22
 - creating, 4-22
 - descriptors and classes, generating, 4-30
 - EJB entity generation, 4-31
 - fields, 4-25
 - generating, 4-29, 4-30, 4-32
 - importing, 4-22
 - Java source generation, 21-15
 - JDBC driver classpath, 4-23
 - properties, 4-25
 - references, 4-27, 4-28
 - removing, 4-24
 - renaming, 4-24
 - schema, 4-22
 - SQL generation, 4-29
 - TopLink Workbench, Navigator window, 4-9
- DatabaseException class, 98-24
- DatabaseLogin, 84-2
- DatabaseMapping class, 33-25
- DatabaseQuery, 96-3
- DatabaseRow, 36-15
- databases
 - catalog, 4-22
 - common problems, 14-23
 - connect to, 4-21
 - creating reference tables on, 4-27
 - custom drivers, 4-18
 - disconnect from, 4-21
 - drivers, 4-18
 - exceptions, 98-24
 - fields, configuring, 37-2
 - for project, 21-3
 - Java type conversion, 6-3
 - linking, 75-28
 - log out of, 4-22
 - logging into, 4-22, 23-7
 - logins, 86-1
 - mapping. *See* mappings
 - platform, 21-3, 23-2, 24-2, 86-1, 87-1
 - preferences, 4-17
 - schema, 4-22
 - schema manager, 6-1
 - tables, 4-21
 - TopLink Workbench, Navigator window, 4-9
 - troubleshooting, 14-23
 - type conversion, schema manager, 6-3
 - using with Oracle TopLink Workbench, 4-21
- DatabaseSession class
 - logging SQL and messages, 75-10
- DB2
 - platform, 84-3
 - schema manager type conversion, 6-3
- DBase platform, 84-3
- default mapping
 - about, 33-4
 - automatic table generation, and, 33-4
 - configuring, 8-12
 - default table generator, 6-6
- Default Mapping Exception, 13-75
- default table generator
 - default mapping, 6-6
 - table creator, creating, 6-4
- defaults
 - login level null values, 85-5
 - mapping level null values, 35-12
 - null values, 35-12, 35-36, 85-5
 - optimization, 11-8
 - root, 32-5
 - see also* preferences
- DefaultSequence, 86-6
- deferred change detection

- configuring, 28-71
- DeferredChangeDetectionPolicy, 100-7
- Delete All Interaction tab, 63-3
- deleteObject(), 29-8
- DeleteObjectQuery, 98-13
- deletes
 - controlling order, 102-16
 - delete operation, 98-4
 - queries, EIS mappings, 63-3
- demarcation of unit of work transactions, 100-2
- dependent objects
 - non-deferred write, 26-4
- deploy tool
 - about, 10-5
 - troubleshooting, 10-6
 - using with WebSphere Studio Application Developer, 10-5
- deploying
 - about, 2-14, 10-1
 - application server requirements, 7-1
 - BEA WebLogic, 10-2
 - CMP applications, 10-2, 10-4
 - database login, 23-7
 - entity beans overview, 2-14
 - generating XML for, 8-3
 - hot deployment, 10-4
 - IBM WebSphere, 10-3
 - Java applications, 10-1
 - JSP and Servlet applications, 10-1
 - modifying BEA WebLogic persistence
 - descriptor, 8-13
 - non-CMP applications, 10-4
 - packaging, 9-1
 - Session Bean applications, 10-1
 - troubleshooting, 15-1
- deployment descriptors, 26-3
- Deployment Exceptions, 13-67
- deployment exceptions
 - BEA WebLogic deployment, 15-3
 - Deployment Exceptions (14001-14033), 13-67
 - IBM WebSphere deployment, 15-12
- deployment files
 - BMP applications, 8-9
 - CMP applications, 8-9
 - creating, 8-1
 - descriptors, 26-3
 - EJB 3.0, 8-2, 9-1
 - JARs, troubleshooting, 15-1
 - Java applications, 8-8
 - JSP and Servlet applications, 8-8
 - Session Bean applications, 8-8
 - XML, generating, 8-3
- deployment XML, exporting, 21-14
- DeploymentXMLGenerator, 8-3
- Descriptor Event Listener, 28-60
- Descriptor Event Manager
 - about, 26-8
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - event types, 28-57

- handlers, 26-8, 28-57, 28-60
- handlers, Descriptor Event Listener, 28-60
- understanding, 26-8
- descriptor events
 - about, 26-8
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - handlers, 26-8, 28-57, 28-60
 - types of, 28-57
 - understanding, 26-8
- Descriptor Exceptions, 13-2, 13-29
- Descriptor Info tab, 28-3, 28-5, 28-7, 29-2, 29-4, 31-2, 31-4, 32-2, 32-4, 32-5, 32-6
- DescriptorEventListener, 28-60
- descriptors
 - about, 19-2, 26-1
 - advanced properties, default, 22-7
 - aggregate, 26-5, 27-2
 - aggregate, EJB 3.0 and, 26-6
 - aggregate, relational projects and, 26-5
 - amending, 2-20, 26-5, 28-78
 - API, 26-22
 - architecture, 26-2
 - attributes, adding, 4-44
 - automapping, 34-2
 - automatically mapping, 34-2
 - BMP, 26-3, 28-44
 - cache refreshing, 28-27
 - change policy, 28-70
 - child inheritance, 28-49
 - class, 27-2
 - CMP, 26-3, 28-44
 - CMPPolicy, 26-3, 28-44
 - comments, 28-8, 28-9
 - composite, 26-5, 26-8, 27-5
 - composite EIS, 27-5
 - configuring, 28-1
 - conforming, 102-8
 - creating, 27-1, 27-2, 27-5
 - custom EIS interactions for basic persistence, 31-6
 - custom SQL queries for basic persistence, 29-6
 - deactivating, 4-10
 - default mappings, 33-4
 - default root, 31-4
 - deployment information, 26-3
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - EIS, 26-11, 27-5, 31-1
 - EIS projects, 26-8
 - EJB, 26-3, 28-44
 - EJB information, 26-3, 28-44
 - errors, 4-10, 4-11, 14-4
 - event handlers, 26-8, 28-57, 28-60
 - events, 26-8, 28-57
 - existence checking, 11-13, 28-42
 - fetch groups, 28-77
 - files, merging, 5-5
 - generating from database, 4-30
 - hierarchy, inheritance, 26-23
 - history policy, 28-73

- identity maps, 22-14, 28-35, 28-37, 28-39
- in Java, 26-22
- inactive, 4-10
- inheritance, 26-3, 26-12, 28-49, 28-50
- instantiation, 11-13
- interface, 27-2, 28-31, 28-33
- mapping, 28-3, 29-2, 33-4, 34-1, 34-2
- merging, 5-5
- named queries, 28-10
- nondesoriptor classes, 4-51
- object-relational, 26-11, 30-1
- optimizing, 11-13
- parent inheritance, 28-50
- projects, 19-2, 26-1
- query key interfaces, 28-31
- query timeout, 28-24, 28-26
- read only, 28-4
- registering with sessions, 75-12, 77-2, 77-3, 77-10
- relational, 26-11, 27-2, 29-1
- removing, 4-51
- returning policy, 28-65
- root EIS, 27-5
- root element, 32-5
- schema context, 31-2, 32-2, 32-3
- sequencing, 26-9, 29-3
- TopLink Workbench, Navigator window, 4-9
- types of, 26-1
- validating, 27-6
- XML, 26-12, 32-1
- XML projects, 26-8
- detachement indirection, 33-9
- developing applications with Oracle TopLink, 2-1
- development environments
 - about, 3-2
 - configuring, 5-1
 - database logins, 23-7
- development process
 - about, 2-1
 - additional support, 2-3
 - stages of, 2-1
 - with Oracle TopLink, 2-1
- development tools
 - about, 3-1
 - profiler, 11-2
 - schema manager, 6-1
- dimensionality, array, 4-45, 4-49
- direct access
 - about, 22-4, 35-14
 - specifying, 35-32
- direct collection relational mappings
 - configuring, 45-1
 - example, 36-11
 - understanding, 36-11
- direct collections
 - session broker limitations, 75-27
- direct EIS mappings
 - configuring, 58-1
 - understanding, 56-5
- direct field
 - in direct collection mappings, 45-2
- direct key fields, 46-2
- direct map relational mappings
 - configuring, 46-1
 - direct keys, 46-2
 - direct value, 46-1
 - understanding, 36-12
- direct mappings
 - generating deprecated, 22-12
 - with EJB, 36-4
- direct value fields, 46-1
- direct XML mappings
 - configuring, 67-1
 - understanding, 65-5
- directionality in mappings, 36-2
- direct-to-field mappings
 - ObjectTypemapping deprecated, 36-3
 - SerializedObjectMapping deprecated, 36-3
 - type conversions, 37-3
 - TypeConversionMapping deprecated, 36-3
- direct-to-field relational mappings
 - configuring, 38-1
 - options, 38-1
 - timestamp support, 37-3
 - understanding, 36-4
- direct-to-XMLType relational mappings
 - configuring, 39-1
 - understanding, 36-4
- Discovery Exception, 13-77
- DMS profiler
 - about, 11-4, 75-11
 - accessing with JMX, 11-7
 - and JMX
 - nouns, 11-4, 75-11
 - selecting, 77-11, 77-14, 77-17, 77-20
 - sensors, 11-4, 75-11
- document information in XML schemas, 4-34, 4-36, 4-38
- documentation
 - hosted, 4-14
 - See also* Help
- does exist write object, 11-27
- dontOptimizeDataConversion(), 11-15
- doPrivileged(), 7-4
- DRIVER_CLASSPATH
 - Oracle TopLink Workbench environment, 4-2
- drivers, custom database, 4-18
- dynamic batch writing
 - about, 11-15
 - setMaxBatchWritingSize(), 11-15
- dynamic fetch groups, querying with, 99-4

E

- Editor window, about, 4-4, 4-10
- EIS
 - about, 20-8
 - architecture, 2-4
 - call, 98-24
 - custom interactions for basic persistence, per descriptor, 31-6

- indexed records, configuring, 31-5
- interactions, 31-6, 96-6, 98-24
- mapped records, configuring, 31-5
- mappings, 20-8, 56-2
- projects, 21-3
- queries, 96-6
- record format, configuring, 31-5
- XML records, configuring, 31-5
- EIS descriptors
 - composite, 27-5
 - configuring, 31-1
 - default root, 31-4
 - locking policy, 28-62
 - root descriptor, 27-5
 - schema context, 31-2
 - setDataTypeName, 31-5
 - understanding, 26-11
- EIS mappings
 - about, 56-1, 56-2
 - architecture, 56-5
 - composite collection, 56-7, 61-1
 - composite direct collection, 56-6, 59-1
 - composite object, 56-7, 60-1
 - configuring, 57-1
 - direct, 56-5, 58-1
 - jaxb:class support, 56-3
 - list support, 56-3
 - one-to-many, 56-12, 63-1
 - one-to-many, key on source, 56-13
 - one-to-many, key on target, 56-15
 - one-to-one, 56-8, 62-1
 - one-to-one, key on source, 56-9
 - one-to-one, key on target, 56-10
 - transformation, 56-17, 64-1
 - types of, 56-1
 - union support, 56-3
 - xsd:list, 56-3
 - xsd:union, 56-3
- EIS projects
 - configuring, 24-1
 - connector.jar, 20-7
 - indexed records, 20-9
 - mapped records, 20-9
 - sequencing, 20-5
 - understanding, 20-7
 - XML records, 20-9
- EIS queries, 96-6
- EIS record types, supported, 56-2
- EISLogin, 84-2
- EJB
 - descriptors, 21-10
 - isolated client sessions, 75-24
 - setter parameter type checking, 7-25
 - single-object finder return type checking, 7-25
 - unknown primary key class support, 7-25
- EJB 1.1
 - and BMP, 2-34
 - and CMP, 2-27
- EJB 1.x
 - indirection, 33-9
 - serialization, 33-9
- EJB 2.x
 - and BMP, 2-34
 - and CMP, 2-27
 - default mapping, 33-4
 - indirection, 33-9
 - serialization, 33-9
- EJB 3.0
 - <J2EE-Container>-jar.xml file, 8-6
 - and CMP, 2-27
 - attribute change tracking policy, OC4J CMP integration, 7-5, 100-8
 - Bean annotation, fetch=lazy, 33-9
 - default mapping, 33-4
 - deployment files, 8-2, 9-1
 - deployment files, OC4J CMP integration, 7-5
 - detachment, 33-9
 - Embedded annotation, 26-6
 - packaging for deployment, 9-1
 - packaging, OC4J CMP integration, 7-5
 - projects.xml file, 8-3, 8-5
 - serialization, 33-9
 - sessions.xml file, 8-5
 - toplink-ejb.xml file, 8-8
 - value holder indirection, 33-9
- EJB descriptors, opening projects with, 21-10
- EJB entities
 - BMP indirection, 33-9
 - CMP hot deployment, 10-4
 - deployment overview, 2-14
 - EJB 1.x indirection, 33-9
 - EJB 2.x indirection, 33-9
 - EJB 3.0 indirection, 33-9
 - generating, 4-31
 - hot deployment, 10-4
 - indirection, 33-9
 - inheritance, 26-4, 26-17
 - inserting after ejbCreate, 26-4
 - inserting after ejbPostCreate, 26-4
 - mapping, 22-6
 - non-CMP hot deployment, 10-4
 - non-deferred write, understanding, 26-3, 102-25
 - sequencing, 20-21
- EJB entity beans
 - and EJB 1.1, 2-27, 2-34
 - and EJB 2.x, 2-27, 2-34
 - and EJB 3.0, 2-27
 - with BMP architecture, 1-5, 2-33
 - with CMP architecture, 1-5, 2-27
- EJB Exception Factory Exceptions, 13-58
- EJB finders
 - about, 96-23
 - Call finders, 96-26
 - creating, 99-8
 - DatabaseQuery finders, about, 96-26
 - default finders, about, 96-26
 - default finders, creating, 99-8
 - EJB QL finders, about, 96-27
 - ejb-jar.xml options, 99-9
 - ejbSelect method, 96-28

- ejbSelect, creating, 99-15
 - ejbSelect, using, 99-15
 - expression finders, about, 96-27
 - named query finders, about, 96-27
 - predefined, about, 96-24
 - primary key finders, about, 96-27
 - redirect finders, about, 96-28
 - redirect finders, using, 99-12
 - single-object finder return type checking, 7-25
 - SQL finders, about, 96-28
- EJB Info tab, 28-44
- EJB JAR XML Exception, 13-84
- EJB Preferences, 4-16
- EJB QL
 - exceptions, 13-55
 - queries, 29-6, 96-5
 - query language, 96-5
- EJB session bean facade architecture
 - about, 1-5, 2-24
 - understanding, 2-24
- EJB Session Beans, 75-31
- ejbc
 - about, 10-2
 - troubleshooting, 10-3
- ejbCreate, 26-4
- ejb-jar.xml file
 - about, 8-5, 21-15
 - configuring, 8-5
 - corresponding to Oracle TopLink Workbench functions, 21-15
 - EJB finder options, 99-9
 - location, 22-7
 - managing, 5-6
 - preferences, 4-16
 - synchronization under EJB 2.0, 8-5
 - updating from, 21-16
 - writing, 21-16
- ejbPostCreate, 26-4
- @Embeddable, 26-6
- @Embedded, 26-6
- empty unit of work transactions, 100-6
- encrypting login passwords, 21-13
- enhanced validation exceptions, 8-4
- Enterprise Information Systems. *see* EIS
- entity beans
 - deployment, 2-14
 - descriptor information, 26-3
 - direct mappings, 36-4
 - indirection, BMP, 33-9
 - indirection, EJB 1.x, 33-9
 - indirection, EJB 2.x, 33-9
 - indirection, EJB 3.0, 33-9
 - sequencing with, 20-21
- environment
 - configuring, 4-2
 - JAVA_HOME, 4-2
 - JDBC_CLASSPATH, 4-2
 - proxy, 4-13, 4-14, 21-16
- error codes
 - 1-176, 13-2
 - 1-99, 14-1
 - 100-199, 14-2
 - 200-399, 14-4
 - 400-599, 14-11
 - 500-699, 14-15
 - 700-799, 14-20
 - 800-899, 14-21
 - 10001-10047, 13-58
 - 12000-12004, 13-64
 - 13000-13020, 13-64
 - 14001-14027, 13-67
 - 15001-15024, 13-70
 - 17001-17006, 13-71
 - 18001-18002, 13-74
 - 19001-19003, 13-75
 - 22001-22004, 13-77
 - 22101-22105, 13-77
 - 3001-3007, 13-26
 - 4002-4018, 13-27
 - 5001-5008, 13-29
 - 6001-6098, 13-30
 - 7001-7104, 13-42
 - 72000-72023, 13-84
 - 8001-8010, 13-55
 - 9000-9009, 13-56
- errors
 - about, 13-1
 - codes and descriptions, 13-1, 14-1
 - descriptors, 4-10, 4-11
 - migration, 7-13
 - Oracle TopLink Workbench, 14-1
- Event Manager, 75-5
- events
 - about, 26-8
 - client session, 75-6
 - database access, 75-6
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - handlers, 26-8, 28-57, 28-60
 - listeners, sessions, 75-7
 - session, 75-5
 - session manager, 75-6
 - sever session, 75-6
 - types of, 28-57
 - unit of work, 75-6
- examples
 - composite collection EIS mapping, 56-8
 - context menu, 4-5
 - cursor streams, 99-17
 - direct collection mappings, 36-11
 - direct-to-field mappings, 36-4
 - exception handler, 77-13
 - indirection, 33-6
 - inheritance, 26-12
 - Oracle TopLink Workbench, 4-3
 - performance optimization, 11-22, 11-24
 - proxy indirection in code, 35-9
 - READALL finders, 99-10
 - report query, 98-6
 - scrollable cursors, 99-17

- serialized mapping, 33-10
- stored procedure call, 98-21, 98-22
- transformation mapping, 36-15
- transformation XML mapping, 56-18, 65-31
- Unit of Work, 100-6, 101-7
- write, write all, 98-3
- exception handler
 - about, 75-12
 - example, 77-13
 - selecting, 77-12
- exceptions
 - chained, 75-10
 - communication exceptions, 13-64
 - conversion exceptions, 13-26
 - database exceptions, 13-27, 98-24
 - Default mapping exception, 13-75
 - deployment exceptions, 13-67
 - descriptor exceptions, 13-2
 - discovery exceptions, 13-77
 - EJB exceptions factory, 13-58
 - EJB JAR XML exceptions, 13-84
 - EJB QL exceptions, 13-55
 - enhanced validation, 8-4
 - java.security.AccessControlException, 15-12, 15-13
 - JMS processing exceptions, 13-74
 - Migration utility exception, 13-82
 - optimistic locking, 13-29
 - query exceptions, 13-30
 - remote command manager exceptions, 13-77
 - SDK data store exceptions, 13-71
 - SDK descriptor exceptions, 13-75
 - selecting exception handler, 77-12
 - session loader exceptions, 13-56
 - synchronization exceptions, 13-70
 - Transaction exception, 13-79
 - validation exceptions, 13-42
 - XML conversion exception, 13-80
 - XML data store exceptions, 13-64
- exclusive connections
 - about, 102-28
 - internal read connection pool, 89-6
 - isolated sessions, 75-20, 77-19
 - named queries, 28-24
- existence checking, 102-5
 - descriptors, 28-42
 - projects, 22-8
- expanding items in Navigator window, 4-9
- expiration of objects in the cache, 22-19, 28-40
- explicit query refreshes, cache coordination, 90-7
- exporting
 - deployment XML, 21-14
 - Java model, 21-14
 - Java source, 21-14
 - preferences, 4-12
 - projects, 21-13
- Expression Builder, 97-12, 97-15
- Expression Builder dialog box, 97-12
- Expression class, 97-1
- ExpressionMath class, 97-1

- expressions
 - about, 96-3, 97-1
 - allows none, 28-18, 97-14
 - allows null, 28-18, 97-14
 - building, 97-12
 - comparing with SQL, 97-1
 - components, 97-2
 - creating, 97-12
 - data level queries, 97-11
 - database functions, 97-3
 - in relationships, 97-5
 - in-memory queries, limitations, 96-31
 - mathematical functions, 97-4
 - multiple, 97-10
 - one-to-one mappings, 97-5
 - outer joins, 98-11
 - parallel expressions, 97-11
 - parameterized, 97-7
 - platform functions, 97-16
 - query keys, 97-9
 - subqueries and subselects, 97-10
 - user-defined functions, 97-4, 97-16
 - using Boolean logic, 97-3
 - XML Type functions, 97-4
 - see also* queries
- external
 - applications, 102-26
 - connection pools, 84-8
 - controller, transaction, 100-1
 - JDBC pools, 2-35
 - transactions, 100-1
- external transaction controller
 - configuration, sessions, 77-14
 - session, 100-2

F

- factory name, JMS connection, 92-2
- failure, resuming unit of work after commit, 102-14
- features of Oracle TopLink, 1-4
- fetch groups
 - about, 26-4, 28-77
 - configuring, 28-77, 99-3
 - default, 28-77, 99-3
 - disabling, 99-3
 - dynamic, 99-4
 - object level read query, 96-12
 - read optimization, and, 28-77
 - size, 11-17
 - static, 99-3
- field references, 37-8
- Field uses XML Schema "type" attribute
 - option, 33-12
- fields in database tables, 4-25
- Fields tab, 47-2
- field-to-object attribute transformation, 35-29, 35-31
- files
 - JAXB-specific, 20-10
 - TopLink-specific, 20-11
 - see also* specific file name

- final attributes, 4-45, 4-48
- findAll, using, 96-26
- finders
 - caching options, 96-37
 - disabling cache, 96-37
 - managing large result sets, 99-19
 - refreshing results, 96-38
 - see also* EJB finders
- flashback queries
 - about, 96-18
 - historical client sessions, 81-1
- forceUpdateToVersionField(), 102-17
- foreign keys
 - about, 2-17, 37-8
 - configuring in EIS mappings, 62-1
 - EIS mappings, 63-1
 - multiple tables, 29-14
 - one-to-many mappings, 36-7
 - one-to-one mappings, 35-34, 36-5
 - parameterized expressions, 97-7
 - references, 14-12, 14-21
 - target, 36-6, 37-9
 - troubleshooting, 14-12, 14-21
- full identity map, 90-3

G

- garbage collection, managing, 101-8
- General Preferences dialog, 4-13
- Generate Classes and Descriptors dialog, 4-30
- Generate EJB Entity Classes and Descriptors dialog, 4-31
- generating
 - access method, 4-30
 - deployment JARs, troubleshooting, 15-1
 - deprecated direct mappings, 22-12
 - see also* exporting
- getCatalogs(), 4-22
- getField(), 97-11
- getImportedKeys(), 4-22
- getParameter, 97-8
- getPrimaryKeys(), 4-22
- getTable(), 97-12
- getTables(), 4-22
- getTableTypes(), 4-22
- getValue(), 33-7
- getValue() method, 33-7

H

- hard cache weak identity map
 - about, 90-4
 - when to use, 90-5
- help
 - about, 4-12
 - displaying, 4-12
- Help button., 4-12
- Help Preferences, 4-14, 4-16
- hierarchical queries
 - about, 96-18

- described, 99-6
- hints, Oracle Hints in queries, 96-18
- historical client sessions
 - about, 75-1, 75-24
 - cache, 75-33
 - limitations of, 75-25
- historical queries, 99-2
 - about, 96-20
 - see also* AsOfClause
- history policy, configuring, 28-73
- holders, value, 33-6
- host URL, JMS topic, 92-3
- hosted
 - documentation, 4-14
 - XSD files, 8-2, 8-4
- hot deployment
 - about, 10-4
 - CMP applications, 10-4
 - non-CMP applications, 10-4
- HSQL platform, 84-3

I

- IBM Informix Database native sequencing, 20-19
- IBM WebSphere
 - deploy tool, 10-5
 - deploying to, 10-3
 - deployment exceptions, 15-12
 - setting classpath, 7-21
- identity
 - about, 2-16, 90-3
 - cache, and, 90-3
 - using cache to preserve, 90-3
 - see also* identity map
- identity map cache
 - disabling during a write query, 98-14
 - refresh in read query, 96-35
- identity maps
 - about, 28-35, 28-37, 28-39, 75-29
 - cascading refresh during read query, 96-35
 - descriptors, 22-14, 28-35
 - example, 96-35
 - full, 90-3
 - guidelines for choosing type, 90-4
 - hard cache weak identity map, 90-4, 90-5
 - isolated client sessions, 75-24
 - no identity map, 90-4
 - refreshing during read query, 96-35
 - soft cache weak identity map, 90-4, 90-5
 - soft cache weak identity map and read optimization, 11-19
 - specifying, 28-35, 28-37, 28-39
 - weak, 90-3
 - weak identity map and read optimization, 11-19
- Identity tab. *see* Caching tab
- impedance mismatch, solving, 1-2
- Implementors tab, 28-33
- Import Tables from Database dialog, 4-23
- importing
 - classes, 4-16

- preferences, 4-12
- inactive descriptors
 - about, 4-10
 - mapping to, 37-6, 57-3
- independent relationships, 35-16
- indexed records, 56-2
- indirection
 - about, 2-18, 33-5
 - bidirectional relationships, 35-35
 - choosing the correct type, 35-4
 - configuring, 35-3
 - EJB, 33-9
 - EJB 1.x CMP, 33-9
 - EJB 2.x CMP, 33-9
 - EJB 3.0 CMP, 33-9
 - example, 33-6
 - many-to-many mappings, 36-9
 - nontransparent, 2-11
 - one-to-many mappings, 35-34
 - proxy indirection, 33-8
 - remote sessions, 75-32
 - serialization, 33-9
 - transparent, 2-11, 33-8
 - value holder, 33-6
 - ValueHolderInterface, 2-11
 - see also* proxy indirection, transparent indirection
- Informix platform, 84-3
- inheritance
 - about, 2-17, 26-3, 26-12
 - aggregate classes, 26-17
 - aggregate collection mappings, 36-10
 - branch classes, 26-12, 26-13
 - child descriptors, 28-49
 - class extraction, 26-14, 26-15
 - class indicator, 26-13, 26-14
 - descriptors, 26-3, 26-12, 28-49, 28-50
 - finding subclasses, 26-13
 - instantiating subclasses, 26-13
 - isolated client sessions, 75-24
 - leaf classes, 26-12, 26-13, 99-4
 - primary keys, 26-15
 - queries, 96-22
 - querying on hierarchy, 99-4
 - relational parent, 28-50
 - root class, 26-12, 28-49, 28-51
 - root class subclasses, finding in inheritance, 26-13
 - supporting with multiple tables, 26-16
 - supporting with one table, 26-15
 - transformed to relational model, 11-11
 - using with EJB, 26-4, 26-17
- inheritance hierarchies
 - descriptors, 26-23
 - querying on, 99-4
- Inheritance tab, 28-49, 28-51
- inherited subclasses, mapping, 28-56
- in-memory query
 - about, 90-6
 - check cache using exact primary key, 96-30
 - check cache using primary key, 96-30
 - check database if not in cache, 96-30
 - conform results in unit of work, 96-31
 - conforming, 102-9
 - expression limitations, 96-31
 - supported, 96-30
 - using, 96-30
- inner join, 97-5
- insert operation, 98-3, 98-4
- insertObject(), 29-7
- instantiation policy
 - about, 28-68
 - setting, 28-68
- Instantiation tab, 28-68
- integrity checker, 27-6
 - about, 75-12
 - configuring, 77-18
- interactions
 - about, 98-24
 - creating, 31-6
- interface alias
 - about, 29-10
 - creating, 29-11
- Interface Alias tab, 29-11
- interfaces
 - classes, implementing, 4-43
 - customizing, 4-13
 - descriptors, 27-2, 28-31, 28-33
 - queries, 96-22
 - query keys, 28-31
 - querying on, 99-4
- internal connection pool
 - about, 84-7
 - named, 84-9
 - read, 84-8
 - sequence, 84-8
 - write, 84-8
- internal query object cache
 - about, 96-36
 - configuring, 99-21, 99-22
 - expiration, 99-22
 - restrictions, 96-36
- internal transactions, 100-1
- invalidation of objects in the cache, 22-19, 28-40, 90-8
- IP address for multicast group, 91-4
- isolated client sessions
 - about, 75-1, 75-19, 80-1
 - configuration, 80-1
 - life cycle, 75-22
 - limitations of, 75-23
 - session event handlers, 75-21
 - with Oracle Virtual Private Database (VPD), 75-20
- isolated session
 - cache, 75-33
 - ConnectionPolicy, 75-20
 - exclusive connections, 75-20
 - supported databases, 75-20
- isolation
 - cache, 90-6
 - transaction levels, 100-2
 - unit of work transactions, 100-4

Iterator interface, 99-16

J

J2C adapters

- about, 84-4
- configuring for Oracle TopLink Workbench, 4-2
- EISLogin, 84-2
- selecting, 84-4
- with EIS, 21-3

<J2EE-Container>-ejb-jar.xml file, 8-5

<J2EE-Container>-jar.xml file

EJB 3.0, 8-6

J2EE

- parameter binding, 11-16
- prepared statement caching, 11-16
- web applications, 1-4

Java

- database tables, 21-15
- exporting to, 21-14
- integration with any datasource, 1-2
- iterators, 99-16
- object model, 2-11

Java applications

- deploying, 10-1
- deployment files, 8-8
- packaging for deployment, 9-1

Java Cryptography Extension, 21-13

Java Management Extensions. *see* JMX

Java Naming and Directory Interface. *See* JNDI

Java Object Builder, 75-4

Java streams

- described, 99-17
- optimizing, 99-18
- support for, 99-17

Java Transaction API

- and unit of work commit, 100-10
- and unit of work rollback, 100-11
- see also* JTA

Java Transaction Service *see* JTA

JAVA_HOME, 4-2

java.security.AccessControlException, 15-12, 15-13

java.util.Collection interface, 35-26

java.util.Map interface, 35-26

java.util.Set interface, 35-26

javax.ejb.EntityBean interface, 4-31

JAXB

- creating projects from, 21-6
- files, 20-10
- generating project from the command line, 21-8
- jaxb:class, and EIS mappings, 56-3
- jaxb:class, and XML mappings, 65-4
- proxy configuration, 21-8
- tljxb.cmd, 21-8
- tljxb.sh, 21-8
- typesafe enumeration converter, 35-25
- understanding, 20-10
- validation, 20-14
- XML projects, 20-10

JAXB typesafe enumeration converter

- configuring, 35-25
- understanding, 33-24

JAXBContext, 20-14

JCE. *see* Java Cryptography Extension

JConnect, 11-14

JDBC

- adaptor for EIS, 56-5
- database gateway for EIS, 56-5
- driver classpath, 4-23
- JConnect, 11-14
- Sybase JConnect, 11-14

JDBC drivers

- about, 84-3
- configuring for Oracle TopLink Workbench, 4-2
- fetch size, 11-14, 11-17
- general properties, 11-14
- mapping LOBs, 34-3
- selecting, 84-3

JDBC pools

- external with EJB Entity Beans with BMP architecture, 2-35

JDBC_CLASSPATH, 4-2

JDeveloper. *See* Oracle JDeveloper

JMS

- connection factory name, 92-2
- coordinated cache, 92-1
- Processing Exceptions, 13-72, 13-74
- topic host URL, 92-3
- topic name, 92-1

JMX

- about
- and DMS profiler
- DMS profiler, 11-7

JNDI naming service, 91-7

joining

- about, 96-12
- expressions, and, 97-5
- mappings, and, 40-1
- one-to-many, about, 98-11
- one-to-one mappings, 40-1, 98-11
- optimizing reads, 11-19
- queries, and, 98-11
- QueryManager expressions, 99-4
- read queries, 11-19, 98-11

JSP and Servlet applications

- deploying, 10-1
- deployment files, 8-8
- packaging for deployment, 9-2

JTA

- about, 100-3
- and unit of work, 100-3
- isolated client sessions, 75-24
- unit of work, 100-1

JTA/JTS

- using with EJB Entity Beans with BMP architecture, 2-36

just-in-time reading. *see* indirection

K

Key Converter tab, 46-3

key pairs

database table reference, 4-28

troubleshooting, 14-12, 14-21

keys

about, 2-16

foreign, 2-17, 35-34, 36-5

foreign, target, 36-6

inheritance, 26-15

multiple tables, 29-13

primary, 26-15, 29-13, 35-2, 36-6, 36-9, 41-3

read-only settings, 35-2

reference key field, 36-11, 36-12

variable class relationships, 36-6, 41-3

L

large result sets, managing in finders, 99-19

lazy attribute loading and read optimization, 28-77

lazy connection

acquisition, 77-20

allocation, 75-18, 78-9

lazy loading. *see* indirection

lazy reading. *see* indirection

leaf classes, 26-12, 26-13, 99-4

life cycle of unit of work, 100-5

LOB

mapping to, 34-2

local

documentation, 4-14

transactions, 102-25

locked files, 5-6

locking policy

AllFieldsLockingPolicy, 26-20

ChangedFieldsLockingPolicy, 26-20

configuring, 28-62

field locking, 26-20

optimistic, 26-9, 26-18, 26-20

optimistic version locking, 26-17

OptimisticLockException, 26-18, 26-21

pessimistic locking policy, 26-9, 26-21

SelectedFieldsLockingPolicy, 26-21

stale data, and, 90-6

three-tier architectures, optimistic locking
and, 26-21

three-tier architectures, pessimistic locking
and, 26-22

TimestampLockingPolicy, 26-18

understanding, 26-9, 26-17

version locking, 26-18

VersionLockingPolicy, 26-18

Locking tab, 28-63

log into database, 4-22

Log Out of Database, 4-22

log out of database, 4-22

logging

application server, 75-9

chained exceptions, 75-10

java.util.logging, 75-9

log level, 75-10

Oracle Enterprise Manager 10g, 75-10

output, 75-10

permissions, 7-24

sessions, 75-7, 77-4, 77-5

TopLink native logging, 75-8

types, 75-8

Logging tab, 77-5

login

CMP deployment, 20-3

database, 23-5, 23-7, 86-1

deployment, 20-3

development, 20-4

platforms, and, 20-4

projects, and, 20-3, 74-1, 84-2

role in project, 20-3, 84-2

session, 20-3, 77-4

session role, non-CMP, 20-3

logMessages method, 75-7

look and feel, specifying, 4-13

M

Manage Non-Descriptor Classes dialog, 4-51

management, source control, 5-3

manager, session events, 75-5

many-to-many mappings

relation table, 36-9

relation tables, 43-1

session broker limitations, 75-27

many-to-many relational mappings

configuring, 43-1

EJB, 36-9

understanding, 36-8

mapped records, 56-3

mapping extensions

custom data types, 12-1

JAXB typesafe enumeration converter, 33-24,
35-25

object type converter, 33-12, 35-22

serialized object converter, 33-10, 35-18

simple type translator, 33-12, 35-23

transformation mappings, 33-14

type conversion converter, 33-11, 35-20

mappings

about, 2-15, 2-16, 19-1, 19-2, 33-1, 34-1

access types, 35-15

aggregate collection mappings and EJB, 36-11

anyType mapping, 66-3

as part of the application development
process, 2-20

automatic, 34-2

batch options, 37-7

class hierarchy, 33-25

comments, 35-18

configuring, 34-1, 35-1

database field, 37-4

default mapping, 33-4

deprecated, generating, 22-12

direct access, 11-13, 35-14

- directionality, 36-2
- EIS mappings, 20-8, 33-27, 56-1, 57-1
- EJB 2.0 entities, 22-6
- errors, 14-11
- example, 33-3
- extensions, about, 33-10
- hierarchy, 33-25
- inactive descriptors, 37-6, 57-3
- indirection, 11-13, 33-5, 35-3
- isolated client sessions, 75-23
- manually configuring, 34-1
- many-to-many, 36-9
- many-to-many, with EJB, 36-9
- method access, 35-14, 35-32
- null values, 35-12, 35-36
- object-relational, 49-1, 50-1
- one-to-many object, with EJB, 36-8
- one-to-one with EJB, 36-6
- optimizing, 11-13
- OX mappings, 33-27
- projects, and, 19-2
- read only, 35-2, 35-3
- relation tables, 43-1, 43-2
- relational, 36-1, 37-1
- removing, 34-6
- to tables, 28-3, 29-2
- TopLink Workbench, Navigator window, 4-9
- types of, 33-1
- XML mappings, 65-1, 66-1
- mathematical functions, in TopLink
 - expressions, 97-4
- menu bar, 4-5
- menus
 - about, 4-4, 4-5
 - context menus, 4-5
 - menu bar, 4-5
- merging
 - changes in clones, 102-12
 - Oracle TopLink Workbench project files, 5-4
 - project files, 5-4
- messages, error, 13-1, 14-1
- metadata
 - about, 2-12, 2-19
 - advantages, 2-19
 - creating, 2-20, 2-21
 - mapping and configuration, 19-1
 - project metadata, 2-20
 - session metadata, 2-21
- Metalink, 2-3
- method access
 - about, 22-4, 35-14
 - setting, 35-32
- methods
 - adding, 4-47
 - getValue(), 33-7
 - setValue(), 33-7
 - wrapper policy, 28-76
 - see also* specific method name
- Microsoft Access
 - platform, 84-3

- schema manager type conversion, 6-3
- Microsoft SQL Database native sequencing, 20-19
- migrating
 - error messages, 7-13
 - OC4J persistence to TopLink, 7-5
 - Oracle TopLink Workbench projects, 21-10
 - troubleshooting, 7-13
- Migration Utility Exception, 13-82
- model source, exporting, 21-14
- modifiers, class, 4-42
- multicast group address, coordinated cache, 91-4
- multicast port, coordinated cache, 91-5
- multiple sessions, 75-28, 78-2
- multiple tables
 - about, 29-13
 - specifying for descriptors, 29-13
- multiplicity in relationships, resolving, 7-14
- multi-processing, 11-29
- Multitable Info tab, 28-48, 29-13
- mutable mappings, 35-33
- .mwp file, 4-1, 21-2
- MySQL4
 - batch writing, 86-9
 - platform, 84-3
 - primary key restrictions, 20-15
 - schema manager type conversion, 6-3

N

- named connection pools, 84-9
- named queries
 - about, 28-10, 77-21, 96-15
 - configuring, 28-10, 77-21
 - descriptor level, 28-10
 - exclusive connections, 28-24
 - options, advanced, 28-24
 - parameter binding, 28-22
 - prepared statement caching, 28-22
 - redirect query, 96-16
 - session level, 77-21
 - using, 98-17
 - when not to use, 96-16
 - when to use, 96-16
- namespaces
 - about, 20-5, 20-22
 - configuring, 4-38
- naming service
 - coordinated cache, 91-7
 - JNDI, 91-7
 - RMI, 91-9
- native sequencing
 - IBM Informix Database, 20-19
 - Microsoft SQL Database, 20-19
 - Microsoft SQL Server, 6-6
 - non-Oracle database, 20-19
 - Oracle Database, 20-18, 20-20
 - Oracle Database SEQUENCE object, 20-19
 - Sybase Database, 6-6, 20-19
- Navigator window
 - about, 4-4, 4-9

- attribute and mapping, 4-9
- database, 4-9
- database tables in, 4-21
- descriptor, 4-9
- example, 4-9
- package, 4-9
- project, 4-9
- unsaved or changed item, 4-9
- NCHAR, 33-12
- NCLOB, 33-12
- neediness warnings. *See* troubleshooting
- nested table object-relational mappings
 - configuring, 55-1
 - understanding, 49-3
- nested unit of work, 100-9, 102-15
- new projects, 21-2
- New Reference dialog box, 4-27
- New Session button, 76-4, 76-6, 76-8
- New Sessions Configuration, 76-2
- New Table dialog box, 4-22
- newInstance method, 102-2
- no identity map, 90-4
- non-cascading write queries
 - compared to cascading, 96-14, 98-13
 - creating using dontCascadeParts ()
 - method, 96-14, 98-13
- non-deferred write
 - configuring, 102-11
 - dependent objects, 26-4
 - understanding, 26-3, 102-7, 102-25
- nonintrusive persistence, 2-18
- nonpersistent projects, 20-2
- nonrelational projects, 20-2
- nontransactional data sources, 84-1
- nontransparent indirection, 2-11
- nouns
 - DMS profiler, 11-4, 75-11
- null values
 - default, 35-12, 35-36, 85-5
 - in expressions, 97-13
 - login level, 85-5
 - mapping level, 35-12
- NVARCHAR2, 33-12

O

- object array object-relational mappings
 - configuring, 54-1
 - understanding, 49-2
- object cache, 96-36
- object cache, sessions, 75-3
- object change tracking
 - configuring, 28-71
- object identity, 75-29
 - about, 2-16, 90-3
 - cache, and, 90-3
 - using cache to preserve, 90-3
 - see also* identity map
- object indirection
 - read optimization, as, 11-19

- object level modify query
 - about, 96-13, 96-14
- object level read query
 - about, 96-11, 96-13
 - fetch groups, 96-12
 - join reading, 96-12
 - partial object query, 96-11
 - read all query, 96-11
 - read object query, 96-11
- object model
 - about, 2-16
 - generating with t ljaxb.cmd, 21-9
 - optimization, 11-8
 - Oracle TopLink requirements, 2-11
 - tutorial, 16-2
- object type converter
 - about, 12-1, 33-12
 - configuring, 35-22
- object type mappings
 - configuring, 35-22
- ObjectLevelChangeTrackingPolicy, 100-7
- object-relational descriptors
 - configuring, 30-1
 - locking policy, 28-62
 - understanding, 26-11
- object-relational mappings
 - about, 49-1
 - array, understanding, 49-2
 - configuring, 50-1
 - nested table, 49-3, 55-1
 - object array, 49-2, 54-1
 - overview, 2-15
 - reference, 49-2, 52-1, 53-1
 - structure, 49-2, 51-1
- object-relational projects
 - about, 20-6
 - sequencing, 20-5
- objects
 - cascading refresh in cache, 96-35
 - creating and registering, 102-2
 - query, 98-6
 - refreshing in cache, 96-35
 - registering and unregistering, 102-1
- ObjectTypingMapping
 - see* ObjectTypeConverter
- OC4J. *See* Oracle Containers for J2EE
- one-to-many EIS mappings
 - configuring, 63-1
 - key on source, 56-13
 - key on target, 56-15
 - understanding, 56-12
- one-to-many relational mappings
 - configuring, 42-1
 - understanding, 36-7
- one-to-one EIS mappings
 - configuring, 62-1
 - key on source, 56-9
 - key on target, 56-10
 - understanding, 56-8
- one-to-one relational mappings

- configuring, 40-1
 - expressions, 97-5
 - joining, 40-1
 - understanding, 36-5
 - with EJB, 36-6
- online help, 4-14
- Open Project button, 21-10
- opening projects, 21-10
- operators
 - boolean logic, 97-3
- optimistic locking
 - about, 26-9
 - application architecture, 2-10
 - cascading locking policy, 26-18, 28-65
 - database exception, 98-24
 - exceptions, 13-29
 - field locking policy, about, 26-20
 - rollbacks, 26-20
 - version locking policy, 26-17, 26-18, 28-65
 - with `forceUpdateToVersionField()`
 - method, 102-17
- optimistic locking policy
 - field locking, about, 26-20
 - version locking, 26-17, 26-18, 28-65
- `OptimisticLockException`, 26-18, 26-21
- optimization
 - about, 11-1
 - application bottlenecks, 11-2
 - application server, 11-30
 - batch reading, 11-17
 - batch writing, 11-15
 - CMP partial object queries, 11-17
 - code generation, 11-8
 - data access, 11-14, 11-15
 - data format, 11-15
 - database, 11-30
 - descriptors, 11-13
 - DMS profiler, 11-4, 75-11
 - existence checking, 11-13
 - fetch groups, 11-17
 - fetch size, JDBC, 11-17
 - general, 11-8
 - inheritance, 26-16
 - instantiation, 11-13
 - JDBC driver, 11-14, 11-17
 - join reading, 11-17
 - mappings, 11-13
 - named queries, 11-17
 - object model, 11-8
 - overview, 2-15
 - parameter binding, 11-15
 - partial object queries, 11-17
 - prepared statement caching, 11-15
 - profiler, 11-4, 75-11
 - optimization
 - TopLink Profiler**, 11-2
 - queries, 11-16
 - reading, 11-19
 - `ReadQuery` method `setMaxRows`, 11-18
 - schema, 11-8
 - `setMaxRows`, 11-18
 - understanding, 11-1
 - unit of work, 11-30
 - writing, 11-26
- Oracle
 - development support, 2-3
 - remote session support, 75-31
- Oracle Containers for J2EE
 - creating projects from, 7-9
 - migrating to TopLink, 7-5
- Oracle Database
 - date and timestamp mappings, 37-3
 - native sequencing, 20-20
 - Oracle Database
 - SEQUENCE** object, 20-19
 - platform, 84-3
 - schema manager type conversion, 6-3
- Oracle extensions
 - hierarchical queries, 99-6
 - Oracle Hints, 99-6
- Oracle Hints, using with TopLink queries, 99-6
- Oracle JDeveloper
 - configuring with Oracle TopLink, 5-1
 - TopLink sessions, 5-3
- Oracle TopLink
 - about, 1-1, 4-1
 - application architectures, 1-4
 - architectures, 1-1
 - deploy tool for IBM WebSphere, 10-5
 - development, 2-1, 3-1
 - features, 1-4
 - integrating with application server, 7-1, 7-2
 - mapping types, 33-1
 - optimization, 11-1
 - packaging your application, 9-1
 - public source, 12-2
 - runtime components, 3-2
 - understanding, 1-1
- Oracle TopLink Sessions Editor. *see* sessions
- Oracle TopLink Workbench
 - about, 4-1
 - Ant integration, 4-53
 - classpath, 4-2
 - creating projects, 21-1
 - development process, 4-1
 - `DRIVER_CLASSPATH`, 4-2
 - environment, 4-2
 - error messages, 14-1, 14-2
 - `JDBC_CLASSPATH`, 4-2
 - parts of, 4-3
 - preferences, 4-12
 - project, 4-1, 21-2
 - proxy, 4-13, 4-14, 21-16
 - sample, 4-3
 - table creator, creating, 6-4
 - upgrading projects, 21-10
- Oracle Virtual Private Database (VPD)
 - isolated client sessions, 75-20
 - proxy authentication, 75-21, 84-6, 86-13
- `oracle.sql.TimeStamp`, 37-3

- order
 - query keys, 37-8
 - relational mappings, 37-8
- OrderSibling, 99-7
- orion-ejb-jar.xml file
 - about, 8-9
 - entity-deployment attribute pm-name, 8-9
 - modifying for Oracle TopLink, 8-9
 - persistence-manager attribute
 - class-name, 8-10
 - persistence-manager attribute
 - descriptor, 8-10
 - persistence-manager attribute name, 8-9
 - persistence-manager subentry
 - forpm-properties
 - cache-synchronizations, 8-11
 - customization-class, 8-11
 - db-platform-class, 8-11
 - default-mapping, 8-11
 - project-class, 8-10
 - remote-relationships, 8-11
 - session-name, 8-10
- OTN (Oracle Technology Network), 1-4, 2-3
- OTS (Object Transaction Service)
 - about, 100-3
 - unit of work, 100-3
- outer joins
 - in expressions, 98-11
 - inheritance, 28-47
- output parameter event in stored procedures, 98-22
- OX mappings
 - about, 33-27
 - extensions, simple type translator, 33-13, 33-14
 - read conversions, 33-13
 - write conversions, 33-14

P

- package names
 - generating, 4-30
 - renaming, 4-52
 - TopLink Workbench, Navigator window, 4-9
 - see also* classes
- packaging for deployment
 - about, 9-1
 - BMP applications, 9-5
 - CMP applications, 9-4
 - EJB 3.0, 9-1
 - Java applications, 9-1
 - JSP and Servlet applications, 9-2
 - Session Bean applications, 9-3
- packet time-to-live cache coordination, 91-15
- parallel
 - expressions, 97-11
 - unit of work, 100-9
- parameter binding
 - about, 11-15
 - byte arrays, 11-15
 - configuring, 23-7, 28-22, 86-9, 98-17
 - database login level, 86-9
 - descriptor level, 28-22
 - external connection pools, 11-16
 - internal connection pools, 11-16
 - J2EE, 11-16
 - named queries, 28-22
 - optimizing, 11-15
 - project level, 23-7
 - queries, 98-17
 - streams, 11-15
 - strings, 11-15
 - trouble shooting, 11-15
- parameterized batch writing
 - about, 11-15
 - setMaxBatchWritingSize(), 11-15
- parameterized expressions
 - about, 97-7
 - example, 97-8
- parameterized SQL
 - enabling on queries, 98-17
 - Oracle TopLink optimization features, 11-26
 - See also* parameter binding
- parser conflicts, XML, 7-3
- partial object reading optimization, 11-19
- passwords, encryption, 21-13
- performance optimization
 - about, 11-1
 - application bottleneck, 11-2
 - examples, 11-20
 - JConnect method isClosed, 11-14
 - using Performance Profiler, 11-2
- Performance Profiler
 - about, 11-2
 - class, 11-3
- persistence
 - about, 2-18
 - BEA WebLogic deployment, 8-13
 - by reachability, 101-6
 - components of, 2-12
 - descriptor, 8-13
 - implementation options, 2-11
 - manager, 7-4
 - OC4J, 7-5
 - projects, 20-2
 - types, 22-5
 - using a persistence layer, 2-13
- persistence manager
 - default, 7-4
 - migration, 7-4
 - restrictions, 7-4
- persistent classes
 - requirements, 2-11
 - types, 22-7
- pessimistic locking
 - about, 26-9, 26-21
 - application architecture, 2-10
 - policy, 26-21
- phantom reads, preventing, 102-30
- platforms
 - Attunity Connect database, 84-3
 - Cloudscape database, 84-3

- data source, 84-3
- database, 21-3, 23-2, 24-2, 84-3, 86-1, 87-1
- DB2 database, 84-3
- EIS, 84-4
- functions in expressions, 97-16
- HSQL database, 84-3
- Informix database, 84-3
- J2C adapter, 84-4
- JDBC drivers, 84-3
- Microsoft Access database, 84-3
- MySQL4 database, 84-3
- Oracle database, 84-3
- Oracle8 database, 84-3
- Oracle9 database, 84-3
- parser, XML, 7-2
- PointBase database, 84-3
- projects, and, 20-4
- server, 77-15
- session configuration, 77-15
- SQLAnywhere database, 84-3
- SQLServer database, 84-3
- SybasePlatform database, 84-3
- XML parser, 7-2
- see also* target platforms
- PointBase platform, 84-3
- pop-up menus. *see* context menus
- ports
 - multicast group, 91-5
 - permissions, 7-23, 7-24
- post-commit clones, avoiding, 102-30
- Potential EJB Descriptors dialog box, 21-10
- pre-allocating sequence numbers, 20-20, 23-4, 86-5, 87-2
- preferences
 - class import, 4-16
 - database, 4-17
 - EJB, 4-16
 - general, 4-13
 - help, 4-14
 - importing and exporting, 4-13
 - Oracle TopLink Workbench, 4-12
 - sessions, 4-18, 4-19
- Preferences - Class dialog, 4-16
- Preferences - EJB dialog, 4-16
- Preferences - General dialog, 4-13
- Preferences - Help dialog, 4-14
- Preferences - Mappings dialog, 4-15
- Preferences dialog, 4-13
- prepared statement caching
 - about, 11-15
 - configuring, 23-7, 28-22, 86-9, 98-17
 - database login level, 86-9
 - descriptor level, 28-22
 - external connection pools, 11-16
 - internal connection pools, 11-16
 - J2EE, 11-16
 - named queries, 28-22
 - optimizing, 11-15
 - project level, 23-7
 - queries, 98-17
 - query level, 98-17
- preserving XML documents, 32-6
- primary key
 - about, 2-16
 - composite, 36-9
 - inheritance, 26-15
 - multiple tables, 29-13
 - primkey in ejb-jar.xml file, 21-15
 - read-only settings, 35-2
 - restrictions, 20-15
 - setting, 4-26, 28-3, 29-2
 - unit of work, 100-11
 - unknown, 28-45
 - variable class relationships, 36-6, 41-3
- private relationships, 35-16
- Problems window
 - about, 4-4, 4-11
 - sample, 4-11
 - see also* error messages
- profiler
 - about, 77-10
 - development tool, 11-2
 - DMS, 11-4, 75-11
 - Oracle TopLink, 11-2, 11-3, 75-11
 - selecting, 77-11, 77-14, 77-17, 77-20
- Project - Multiple Projects tab, 77-10
- Project Status Report dialog box, 21-13
- projects
 - about, 19-1, 20-1, 20-2, 21-2
 - architecture, 20-2
 - cache type and size, 22-13, 28-35
 - comments, 22-21
 - configuring, 22-1
 - copying objects, 5-6
 - creating, 7-9, 21-1, 21-2, 21-3, 21-6
 - deployment login, and, 20-3
 - deployment overview, 2-14
 - descriptors, 19-2
 - development login, and, 20-4
 - direct access to mapped fields, 22-4
 - EIS, 20-7, 20-9, 24-1
 - errors, 14-2
 - existence checking, 22-8
 - exporting, 21-13, 21-14
 - for sessions, 77-9
 - indexed records, 20-9
 - Java, 21-3
 - JAXB, 21-6
 - locked, 5-6
 - login, 20-3, 74-1, 84-2
 - login, and, 20-3
 - mapped field access, default, 22-4
 - mapped records, 20-9
 - mapping projects, creating, 21-2
 - mappings, 19-2, 21-2
 - merging, 5-4
 - metadata, 2-20
 - method access to mapped fields, 22-4
 - model, exporting, 21-14
 - .mwp file, 4-1

- nondesoriptor classes, 4-51
- non-persistent, 20-2
- nonrelational, 20-2
- object-relational, types supported, 20-6
- OC4J, creating from, 7-9
- open, 21-10
- Oracle TopLink Workbench, 21-2
- packages, renaming, 4-52
- persistence type, 20-2, 22-7
- platforms, and, 20-4
- prior TopLink versions, 21-10
- recently opened, 21-10
- relational, 20-2, 20-6, 23-1
- renaming, 21-12
- reopening, 21-10
- saving, 21-11
- sequencing, 20-4, 20-5, 23-3, 86-4
- session login, and, 20-3
- sharing, 5-6
- status report, 21-12
- team development, 5-3
- TopLink Workbench, Navigator window, 4-9
- types of, 20-1, 21-2, 26-1
- updating from `ejb-jar.xml`, 21-16
- upgrading from 2.x or 3.x, 21-10
- writing `ejb-jar.xml` file, 21-16
- XML, 20-9, 25-1
- XML records, 20-9
- `projects.xml` file
 - about, 8-2
 - EJB 3.0, 8-3, 8-5
 - schema, 8-2
 - XSD file, 8-2
- propagation mode, cache, 91-2
- proxies. *see* wrapper policy
- proxy authentication
 - about, 84-5
 - applications, 84-5
 - Oracle Virtual Private Database, 75-21, 84-6, 86-13
 - session events, 75-6
 - use cases, 84-5
- proxy indirection
 - about, 33-8
 - example, 35-9
 - restrictions, 33-8
- proxy settings, preferences, 4-13
- public source code, 12-2

Q

- qualified names, database tables, 4-23, 4-24
- queries
 - about, 96-1
 - application development process, 2-13, 74-2
 - building, 96-6
 - cache, 96-29
 - `Call` queries, 96-3, 96-16
 - cascading, 96-14, 98-13
 - concepts, 96-2

- conforming, 102-8
- database queries, 96-10, 98-4
- `DatabaseQuery`, 96-3
- descriptor query manager, 96-3
- EIS interactions, 96-6
- EJB finders, 96-23
- EJB QL query language, 96-5
- `ejb-jar.xml` file, 28-10, 29-6, 31-6
- executing, 96-7
- expressions, 96-3
- fetch groups, 96-12
- flashback queries, 81-1, 96-18
- hierarchical queries, Oracle extensions, 96-18
- hints, Oracle extensions, 96-18
- historical, 96-20, 99-2
- interface and inheritance queries, 96-22
- joining, 96-12
- languages, about, 96-4
- named queries, 96-15
- object level modify query, 96-13, 96-14
- object level read query, 96-11, 96-13
- on inheritance hierarchies, 99-4
- on interfaces, 99-4
- optimizing, 11-16
- Oracle database features, 99-5
- Oracle extensions, 96-17, 96-18
- parameter binding, 98-17
- partial object query, 96-11
- performance, 11-16
- prepared statement caching, 98-17
- query keys, 96-4
- read all query, 96-11
- read object query, 96-11
- redirect queries, 96-19
- remote sessions, 75-31
- report, 11-21, 96-8
- report query, 96-15
- results, 96-8
- returning policy, 96-17
- session queries, 96-9, 98-1
- SQL query language, 96-4
- stored functions, Oracle extensions, 96-18
- subqueries, 97-10
- summary queries, 96-3
- timeout, 28-24, 28-26, 98-10
- types, 96-1
- `UpdateAll`, 96-14
- XML query language, 96-5
- Queries tab, 28-26, 28-28
- query by example, 98-6
- query cache, 96-36
- Query Exception, 13-30
- query keys
 - about, 28-29, 28-33, 96-4
 - adding, 28-31, 28-34
 - and expressions, 97-9
 - creating, 28-31, 28-34
 - direct mappings, 28-30
 - generating, 28-30
 - in expressions, 97-9

- interface descriptors, 28-31, 28-33
- Java implementation, 28-31
- modifying, 28-30
- order, 37-8
- relationship mappings, 28-30
- specifying, 28-31, 28-34
- unmapped attributes, 28-30, 28-31
- Query Keys tab, 28-31
- query object query. *See* DatabaseQuery
- query objects
 - batch reading, 98-10
 - cache expiration, 99-22
 - caching results, 96-36, 99-21
 - examples, 98-5
 - ordering for ReadAll queries, 98-8
 - report query, 11-21
 - specifying collection class, 98-9
- query results
 - about, 96-8
 - caching, 96-36
 - collections, 96-8
 - cursors, 96-8
 - reports, 96-8
 - streams, 96-8
- query timeout example, 98-10
- QueryManager
 - about, 26-8
 - joining expressions, 99-4
- QuerySequence, 86-8

R

- read access
 - providing in sessions, 75-15
- read all operation, 98-2
- read conversions
 - simple type translator, 33-13
- read only
 - descriptors, 28-4
 - files, 5-6
 - mappings, 35-2, 35-3
- read operation, 98-2
- read optimization
 - about, 11-19
 - batch reading, 11-19
 - fetch groups, 28-77
 - joining, 11-19
 - lazy attribute loading, 28-77
 - object indirection, 11-19
 - partial object reading, 11-19
 - report query, 11-20
 - soft cache weak identity map, 11-19
 - unit of work, 11-19
 - weak identity map, 11-19
- read queries
 - cascading refresh of identity maps, 96-35
 - identity map cache refresh, 96-35
 - refreshing identity maps, 96-35
- ReadAll finders
 - executing, 99-10
- ReadAll queries
 - ordering in query objects, 98-8
- readAllObjects()
 - about, 29-8
 - example, 98-2
- reading
 - ejb-jar.xml, 21-16
 - just-in-time reading, 33-8
 - whole XML documents, 39-1
- reading through the write connection, 102-27
- read-locking, 26-9
- readObject()
 - example, 98-2
- recently opened projects, 21-10
- redirect queries
 - about, 96-19, 99-1
 - creating, 99-1
 - finders, 99-13
 - using, 99-12
- reference key field, 36-11, 36-12
- reference object-relational mappings
 - configuring, 52-1, 53-1
 - understanding, 49-2
- ReferenceMapping class, 49-2
- references
 - database tables, 4-27, 4-28
 - foreign keys, 14-12, 14-21
- Refresh from Database button, 4-24
- refresh policy
 - cache, 90-8
 - EJB finders, 90-8
- refreshing
 - cache, 28-27
 - classes, 4-50
 - refreshObject(), 98-2
 - remote sessions, 75-31
 - sessions, 78-5
- registering objects, 102-1
- registerNewObject method, 102-2
- reimporting schemas, 4-36
- relation tables
 - about, 36-9
 - many-to-many mappings, 43-1
 - mappings, 43-2
- relational descriptors
 - associated table, 29-2
 - configuring, 29-1
 - locking policy, 28-62
 - understanding, 26-11
- relational mappings
 - about, 36-1
 - aggregate collection, 36-10, 44-1
 - aggregate object, 36-12, 47-1
 - configuring, 37-1
 - converters, 36-3
 - direct collection, 36-11, 45-1
 - direct map, 36-12, 46-1
 - direct-to-field, 36-4, 38-1
 - direct-to-XMLType, 36-4, 39-1
 - many-to-many, 36-8, 43-1

- one-to-many, 36-7, 42-1
- one-to-one, 36-5, 40-1
- options, 37-1
- order, 37-8
- transformation, 36-3, 36-15, 48-1
- variable one-to-one, 36-6, 41-1
- relational projects
 - about, 20-2
 - configuring, 23-1
 - object-relational databases, 20-6
 - relational databases, 20-6
 - understanding, 20-6
- relationships
 - about, 2-17
 - bidirectional, 4-31, 35-34, 36-5
 - expressions, 97-5
 - in `ejb-jar.xml` file, 21-16
 - unexpected multiplicity, 7-14
- relative locations
 - about, 22-2
 - classpath, 22-3
- Remote Command Manager Exception, 13-77
- remote connection using RMI, example, 76-10
- remote sessions
 - about, 75-2, 75-29
 - application layer, 75-30
 - creating, 76-10
 - limitations of, 75-31
 - securing access, 75-31
 - server layer, 75-31
 - transport layer, 75-30
 - unit of work, 75-32
- Remove Class button, 4-51
- Remove Table button, 4-24
- removing sessions from brokers, 82-1
- Rename dialog, 4-24
- renaming
 - packages, 4-52
 - projects, 21-10, 21-12
- reopening projects, 21-10
- report query
 - about, 96-15
 - query objects, 11-21
 - read optimization, and, 11-20
 - using, 11-21, 96-8
- reports
 - project status, 21-12
 - query results, 96-8
 - see also* status reports
- resuming unit of work
 - after commit, 102-14
 - after commit failure, 102-14
- returning policy
 - configuring, 28-65
 - SQLCall, 96-17
- Returning tab, 28-66
- RMI
 - coordinated cache, 93-1
 - naming service, 91-9
 - remote session support, 75-31

- rollback
 - and Java Transaction API, 100-11
 - overview, 100-11
 - with optimistic locking, 26-20
- root class
 - about, 26-12
 - inheritance, 26-12
- root EIS descriptor, 27-5
- root element, descriptor, 32-5
- runtime
 - acquiring sessions, 75-5
 - components, 3-2
 - services, configuring sessions, 77-14
 - troubleshooting, 15-14

S

- Save All button, 21-11
- Save button, 21-11
- saving projects, 22-2
- schema context
 - descriptors without, 32-3
 - EIS descriptors, 31-2
 - XML descriptors, 32-2
- Schema Document Info tab, 4-36, 4-38
- schema manager
 - about, 6-1
 - automatic table creation, 6-6
 - creating a table creator, 6-4
 - DB2, 6-3
 - default table generator, 6-4
 - Java table creator, 6-4
 - MS Access, 6-3
 - MySQL, 6-3
 - Oracle, 6-3
 - Oracle TopLink Workbench table creator, 6-4
 - sequencing, 6-3, 6-6
 - Sybase, 6-3
 - table creator, 6-2, 6-4, 6-6
 - table definition, 6-2
 - type conversion, 6-3
 - usage, 6-1
- Schema Structure tab, 4-34
- schemas
 - about, 2-16
 - context for EIS descriptors, 31-2
 - context for XML descriptor, 32-2
 - data storage, 2-16
 - database, 4-22
 - default root for EIS descriptors, 31-4
 - details, 4-34
 - document information, 4-36, 4-38
 - errors, 14-20
 - importing, 4-35
 - optimizing, 11-8
 - properties, 4-36
 - reimporting, 4-36
 - schema manager, 6-1
 - structure, 4-34
 - XML schemas, 4-34

- SCM. *see* source control management
- scripts
 - SQL, generating, 4-29
 - see also* SQL
- scrollable cursor
 - traversing, 99-16
 - using for ReadAllQuery, 99-16
- SDK
 - Data Store Exception, 13-71
 - Descriptor Exception, 13-75
- security
 - cache coordination, permissions, 7-23
 - data source access, permissions, 7-24
 - disabling doPrivileged(), 7-25
 - doPrivileged(), 7-4
 - EJB, permissions, 7-24
 - enabling doPrivileged(), 7-25
 - J2EE application, permissions, 7-24
 - java.util.logging, permissions, 7-24
 - loading project.xml, permissions, 7-23
 - loading sessions.xml, permissions, 7-23
 - logging, permissions, 7-24
 - password encryption, 21-13
 - permissions by feature, 7-23
 - permissions when doPrivileged()
 - disabled, 7-24
 - port permissions, 7-23, 7-24
 - SecurityManager integration, 7-4
 - system properties, permissions, 7-23
 - understanding permissions, 7-22
 - with BEA WebLogic, 7-20
- Select Classes dialog box, 4-50
- SelectedFieldsLockingPolicy, 26-21
- sensors, DMS profiler, 11-4, 75-11
- SEQ_COUNT column in sequence table, 20-20
- sequence connection pools, 84-8
- sequence numbers, write optimization
 - features, 11-26
- SEQUENCE objects in Oracle native
 - sequencing, 20-19
- sequencing
 - about, 20-14
 - BEA WebLogic, 20-21
 - BEA WebLogic single column sequence
 - table, 20-17, 23-3, 86-4
 - configuring, 20-15
 - connection pools, 84-8
 - default, 86-5, 86-6
 - DefaultSequence class, 86-6
 - descriptors, 26-9, 29-3
 - entity beans, 20-21
 - IBM WebSphere, 20-21
 - isolated client sessions, 75-24
 - Java configuration, 20-15
 - Microsoft SQL Server, 6-6
 - native, 20-18, 20-19, 20-20
 - non-Oracle database native, 20-19
 - Oracle Database native, 20-18, 20-20
 - Oracle TopLink Workbench configuration, 20-15
 - overriding default, 86-6
 - platform default, 86-5
 - preallocation, 11-28, 20-20
 - QuerySequence class, 86-8
 - schema manager, 6-6
 - SEQ_COUNT, 20-20
 - sequence type, configuring, 23-3, 86-4
 - sessions, and, 75-13
 - stored procedures, 20-18, 86-7
 - Sybase Database, 6-6
 - table, 20-16
 - table, default column and table names, 20-17
 - unary table, 20-17
 - with stored procedures, 86-7
- serialization
 - descriptor exceptions, 102-13
 - EJB 1.x, 33-9
 - EJB 2.x, 33-9
 - EJB 3.0, 33-9
 - indirection, 33-9
 - merging into session cache with unit of
 - work, 102-12
- serialized object converter
 - about, 12-2, 33-10
 - configuring, 35-18
- serialized object mappings, 35-19
- SerializedObjectMapping. *see*
 - SerializedObjectConverter
- server layer, 75-31
- server platform
 - external transaction controller, 77-14
 - runtime services, 77-14
 - session configuration, 77-14
 - session event listener, 77-17
- Server Platform tab, 77-15
- server sessions
 - about, 75-1, 75-13, 75-15
 - cache, 75-32
 - connection options, 75-18
- service channel, coordinated cache, 91-3
- session accessor
 - about, 2-30
 - managing dependent objects with, 2-30
- session beans
 - about, 2-25
 - deploying, 8-8, 9-3, 10-1
 - model, 2-23, 2-25
 - remote session support for, 75-31
- session brokers
 - about, 75-2, 75-25
 - adding sessions to, 82-1
 - alternatives, 75-28
 - architecture, 75-26
 - configuration, 82-1
 - limitations of, 75-27
 - renaming, 82-1
 - two-phase commits, 75-27
 - two-stage commits, 75-27
- session configuration file
 - about, 76-1
 - loading alternative, 78-4

- preferences, 4-18, 4-19
 - see also* sessions.xml file
- session customizer, 75-4
- Session Event Manager, 75-5
- session events
 - example, 75-5, 75-6
 - isolated sessions, and, 75-5
 - manager, 75-5
 - proxy authentication, and, 75-6
- Session Loader Exceptions, 13-56
- Session Manager
 - about, 78-1, 78-2
 - acquiring, 78-2, 78-3
 - defaults, 78-3
 - destroying sessions, 78-10
 - J2EE defaults, 78-3
 - sessions, acquiring, 75-5
 - storing sessions, 78-10
- sessions
 - about, 2-12, 74-1, 75-1, 75-2, 76-1
 - acquiring at runtime, 75-5, 78-1
 - adding to session brokers, 82-1
 - additional mapping projects, 77-9
 - API, 75-33
 - application server logging, 75-9
 - architecture, 75-2
 - cache, 75-32, 90-2
 - client, 75-13
 - configuring, 77-1, 77-17
 - connection policy, 77-19
 - creating, 2-21, 76-1, 76-4, 76-6, 76-8, 88-1
 - customization, 75-4
 - database sessions, 75-28, 76-8
 - destroying in Session Manager, 78-10
 - errors, 14-21
 - event listeners, 75-7, 77-17
 - events, 75-6
 - external transaction controller, 77-14, 100-2
 - historical client sessions, 75-24
 - in sessions.xml file, 75-4
 - isolated client sessions, 75-19, 80-1
 - loading with alternative class loader, 78-4
 - logging, 75-7, 75-9, 75-10, 77-4, 77-5
 - logging into, 78-9
 - logging out of, 78-10
 - logins, 77-4
 - management of, 78-1
 - metadata, about, 2-21
 - multiple sessions, 75-28, 78-2
 - named queries, 77-21
 - object cache, 75-3
 - optimizing, 11-13
 - preferences, 4-18, 4-19
 - queries, 96-7, 96-9, 98-1
 - refreshing, 78-5
 - registering descriptors, 75-12, 77-2, 77-3, 77-10
 - remote sessions, 75-29, 76-10
 - sequencing, about, 75-13
 - server, 75-13
 - server platform, 77-14, 77-15
 - SQL and messages, 75-10
 - storing in Session Manager, 78-10
 - three-tier architecture, 75-14
 - transformation mappings, 36-15
 - types, 75-1
 - unit of work, 75-1, 75-19
- sessions.xml file
 - about, 8-3
 - acquiring, 75-5
 - CMP applications, 8-4, 75-4
 - configuring with JDeveloper, 5-3
 - creating, 76-1
 - default location, 75-5
 - EJB 3.0, 8-5
 - loading alternative configuration file, 78-4
 - non-CMP applications, 8-4
 - schema, 8-4
 - sessions, 75-4
 - XSD file, 8-4
- setAdditionalJoinExpression(), 99-5
- setenv.cmd file, 4-2
- setenv.sh file, 4-2
- setMaxBatchWritingSize(), 11-15
- setMultipleTableJoinExpression(), 99-5
- setShouldPerformDeletesFirst(), 102-16
- Settings tab, 28-26, 28-28
- setValue() method, 33-7
- shared library
 - setting for BEA WebLogic, 7-15
- shared library for BEA WebLogic, 7-15
- simple type translators
 - about, 12-2, 33-12
 - configuring, 35-23
 - in Java, 35-25
 - read conversions, 33-13
 - write conversions, 33-14
- soft cache weak identity map
 - about, 90-4
 - when to use, 90-5
- sorting, in memory, 35-26
- source code, public, 12-2
- source control management
 - projects, 5-3
 - with Oracle TopLink Workbench, 5-3
 - see also* team development
- source table, reference, 4-28
- source.jar, 12-2
- splash screen, 4-13
- SQL
 - call, 98-18
 - comparing with expressions, 97-1
 - custom queries for basic persistence, per descriptor, 29-6
 - EJBQLCall, 98-23
 - generating from database tables, 4-29
 - parameter binding, 11-15
 - parameterized, 11-26, 98-17
 - prepared statement caching, 11-15
 - queries, 96-4
 - scripts with binding arguments, 29-6

- SQLCall, 98-18
 - StoredFunctionCall, 98-22
 - StoredProcedureCall, 98-20
 - unit of work, 102-15
- SQL Creation Script dialog box, 4-29
- SQL DISTINCT, 11-24
- SQL Exception, 13-27
- SQLAnyWhere platform, 84-3
- SQLCall
 - about, 96-17
 - binding input parameters, 98-19
 - input parameters, 98-19
 - input-output parameters, specifying, 98-20
 - output parameters, 98-19
 - Returning Policy, 96-17
 - using, 98-18
- SQLServer platform, 84-3
- stages of development with Oracle TopLink, 2-1
- stale data
 - cache, 90-6, 90-7
 - coordination, cache, 90-7
 - invalidating the cache, 90-7
 - locking policy, and, 90-6
 - per-class cache configuration, 90-7
 - per-query cache refresh, 90-7
- StartWith, 99-7
- stateful
 - beans, 2-25
 - comparing with stateless, 2-25
- stateless
 - comparing with stateful, 2-25
- static attributes, 4-45, 4-48
- static fetch groups, querying with, 99-3
- status report, generating, 21-12
- stored functions
 - about, 96-18
 - using, 98-22
- stored procedures
 - output parameter event, 98-22
 - queries, 98-20
 - sequencing, and, 20-18, 86-7
- streams
 - as query results, 96-8
 - cursor, 75-32, 99-18
- string bindings, CLOB mappings, 34-4
- structure object-relational mappings
 - configuring, 51-1
 - understanding, 49-2
- StructureMapping class, 49-2
- subqueries
 - multiple expressions, 97-10
 - subselects in expressions, 97-10
- subselects, in expressions, 97-10
- summary queries, 96-3
- superclass, 4-41
- Sybase
 - database schema manager type conversion, 6-3
 - native sequencing, 20-19
 - platform, 84-3
- Synchronization Exceptions, 13-70

- synchronous change propagation, 91-2
- system properties
 - oracle.j2ee.security.usedoprivileged, 7-25
 - oracle.j2ee.toplink.security.usedoprivileged, 7-25
 - toplink.cts.collection.checkParameters, 7-25
 - toplink.xml.platform, 7-3

T

- table creator
 - about, 6-2
 - creating, 6-4
 - using, 6-6
- Table Creator dialog, 6-4
- table generation. *see* automatic table generation
- table sequence
 - about, 20-16
 - default column and table names, 20-17
- table-class relationships, 16-6
- tables
 - adding database, 4-22
 - associating with relational descriptors, 29-2
 - database, 4-21
 - defining schema, 6-2
 - errors, 14-15
 - generating, automatic, 33-4
 - import filter, 4-23
 - mapping to descriptors, 28-3, 29-2
 - merging files, 5-5
 - multiple, 29-13
 - primary key, 4-26
 - references, 37-8
 - relation tables for mappings, 43-2
 - TableDefinition class, 6-2
 - see also* database tables
- target foreign keys
 - about, 36-6, 37-9
 - configuring, 37-9
- target platforms
 - about, 2-5
 - choosing, 2-3
- target tables
 - in direct collection mappings, 45-1
 - reference, 4-28
- team development, 5-3
- technical support, 2-3
- three-tier architecture
 - about, 1-4, 2-21
 - authentication, 84-5
 - migrating to scalable architecture, 75-29
 - overview, 2-21
 - sessions, 75-14
- timestamp support
 - about, 37-3
 - direct to field mappings, 37-3
 - Oracle Database, 37-3
 - TIMESTAMP timezone, 37-4
- TimestampLockingPolicy, 26-18
- timezone, with TIMESTAMP, 37-4

- tljxb.cmd file, 21-9
- toolbars, 4-4, 4-6
- topic name, 92-1
- TopLink. *see* Oracle TopLink
- TopLink expressions. *see* expressions
- TopLink profiler
 - about, 11-2
 - selecting, 77-11, 77-14, 77-17, 77-20
- TopLink Workbench. *see* Oracle TopLink Workbench
- toplink-ejb-jar.xml file, 8-6, 19-2
- toplink-ejb.xml file, 8-8
- Transaction Exception, 13-79
- transactional data sources, 84-1
- transactions
 - client-controlled, 102-24
 - CMP, 100-3, 102-24, 102-25
 - container-controlled, 102-24
 - demarcation, 100-2
 - external transaction controller, 77-14, 100-2
 - external, integrating, 100-2, 102-20
 - isolated client sessions, 75-24
 - isolation, 100-2, 100-4
 - JTA, 100-3
 - JTS, 100-3
 - local, 102-25
 - local, CMP, 102-25
 - OTS, 100-3
 - overview, 2-13, 74-3, 100-1
 - see also* unit of work
- transformation EIS mappings
 - configuring, 64-1
 - understanding, 56-17
- transformation mappings
 - about, 33-14
 - attribute transformation, 35-29, 35-31
 - mutable, 35-33
- transformation relational mappings
 - configuring, 48-1
 - understanding, 36-15
- transformation XML mappings
 - configuring, 73-1
 - understanding, 65-31
- transient attributes, 4-45, 4-48
- transparent indirection
 - about, 33-8
 - persistent class requirements, 2-11
- transport layer, 75-30
- troubleshooting
 - BEA WebLogic deployment, 15-3
 - data sources, 14-23
 - deploy tool, 10-6
 - deployment, 15-1
 - ejbc, 10-3
 - IBM WebSphere deployment, 15-12
 - migration from OC4J persistence, 7-13
 - Oracle TopLink Workbench, 14-1
 - unit of work, 102-30, 102-34
- tutorial
 - about, 16-1
 - database schema, 16-4
 - planning, 16-1
 - two-phase commits, 75-27
 - two-stage commits, 75-27
 - two-tier architecture
 - about, 1-6, 2-23
 - authentication, 84-5
 - understanding, 2-23
 - type conversion
 - automatic, 37-2
 - NCHAR, 33-12
 - NCLOB, 33-12
 - NVARCHAR2, 33-12
 - oracle.sql.TimeStamp, 37-3
 - schema manager, 6-3
 - String to TIMESTAMP, 33-12
 - TIMESTAMP to String, 33-12
 - type conversion converter
 - about, 12-2, 33-11
 - configuring, 35-20
 - provided by direct-to-field mappings, 37-3
 - TypeConversionMapping
 - see* TypeConversionConverter
 - types of mappings, 33-1
 - typesafe enumeration, in EIS mappings, 56-3

U

- unary table sequence
 - about, 20-17
 - BEA WebLogic single column sequence
 - table, 20-17, 23-3, 86-4
- undeployment, 10-4
- unexpected relationship multiplicity, 7-14
- unidirectional relationships, 36-2
- unit of work
 - about, 100-1, 100-2, 100-4
 - acquiring, 101-1
 - API, 100-11
 - architecture, 100-1
 - auditing, 102-20
 - benefits of, 100-4
 - cache, 90-2
 - change policy, 100-6
 - clones, 100-9
 - CMP integration, 100-3
 - commit and Java Transaction API, 100-10
 - commit, writing changes before, 102-7, 102-11, 102-23
 - conform results of in-memory query, 96-31
 - creating objects, 101-2
 - deleting objects, 101-7
 - example, 100-6
 - external transaction controller, 77-14, 100-2
 - external transactions, 100-2, 102-20
 - integrating with CMP, 102-24
 - isolation, 100-4
 - JTA integration, 100-3
 - JTS integration, 100-3
 - life cycle, 100-5
 - modifying objects, 101-2

- mutable mappings, 35-33
- nested, 100-9, 102-15
- `newInstance` method, 102-2
- optimization, 11-30, 100-11
- OTS integration, 100-3
- parallel, 100-9, 102-15
- pre-commit validation, 102-34
- primary keys, 100-11
- proxy indirection, 33-8
- queries, 100-12
- read optimization, 11-19
- read-only classes, 102-6, 102-7
- `registerNewObject` method, 102-2
- remote sessions, 75-32
- resuming, 102-14
- reverting, 102-14
- rollback, 100-11
- sessions, 75-1, 75-19
- transaction demarcation, 100-2
- transactions, 100-4
 - unit of work
 - demarcation, 100-2
- troubleshooting, 102-30
- validating objects, 102-34
- with custom SQL, 102-15
- write optimization, 11-26
- writing changes before commit, about, 102-7
- writing changes before commit, and external
 - transaction exceptions, 102-23
- writing changes before commit, and external
 - transaction timeouts, 102-23
- writing changes before commit, as alternative to
 - conforming, 102-11
- unmapping, 34-6
- unregistering objects, 102-1
- unsaved items, displaying in TopLink Workbench
 - Navigator window, 4-9
- update
 - operation, 98-3, 98-4
 - projects from prior versions, 21-10
- `UpdateAll` query, 96-14
- `updateObject()`, 29-8
- Use XML Schema "type" attribute,
 - configuring, 35-23
- `useBatchWriting()`, 86-11
- user-defined functions, in expressions, 97-16

V

- validating
 - descriptors, 27-6
 - JAXB, 20-14
 - projects, 21-12
 - unit of work, 102-34
- Validation Exceptions, 13-42
- Value Converter tab, 46-4
- value holders
 - about, 33-6
 - `ValueHolder` class, 33-6
- `ValueHolderInterface` class, 2-11, 33-7, 36-9

- variable one-to-one relational mappings
 - class indicator, 41-1
 - configuring, 41-1
 - primary key, unique, 41-3
 - understanding, 36-6
- `VariableOneToOneMapping` class, 36-6
- Varray in Oracle database. *see* array mappings
- version control, 5-6
- Version Control Assistance dialog box, 5-6
- `VersionLockingPolicy`, 26-18
- volatile attributes, 4-45, 4-48
- VPD. *see* Oracle Virtual Private Database

W

- warning icon, 4-10, 4-11
- weak identity map, 90-3
- web browser, specifying, 4-14
- web services architecture, 1-5, 2-36
- WebLogic. *see* BEA WebLogic
- `weblogic-ejb-jar.xml` file
 - described, 8-5
 - modifying for Oracle TopLink, 8-13
 - unsupported tags, 8-14
- WebSphere. *see* IBM WebSphere
- welcome screen, 4-14
- wildcard, 66-3
- wrapper policy
 - about, 28-75
 - implementing in Java, 28-76
- write
 - conversions, simple type translator, 33-14
 - write all operation, 98-3
- write query
 - disabling identity map cache, 98-14
 - non-cascading, 98-13
 - objects, 98-12
 - overview, 98-12
- write-locking, 26-9
- writing
 - batch, 11-26, 86-11
 - `ejb-jar.xml` file, 21-16
 - optimization, 11-26
 - sessions write access, 75-16

X

- `xdb.jar` file, 4-3
- XML
 - descriptor, schema context, 32-2
 - generating deployment, 8-3
 - mappings, concepts, 65-2
 - preserving data, 32-6
 - projects, 21-3
 - query language, 96-5
 - reading whole documents, 39-1
 - records, 56-3
 - schemas, 4-34
- XML Conversion Exception, 13-80
- XML Data Store Exceptions, 13-64

- XML descriptors
 - configuring, 32-1
 - schema context, 32-2
 - understanding, 26-12
- XML mappings
 - about, 65-1
 - any collection, 65-29, 72-1
 - any object, 65-27, 71-1
 - any type support, 65-4
 - composite collection, 65-25, 70-1
 - composite direct collection, 65-14, 68-1
 - composite object, 65-21, 69-1
 - concepts, 65-2
 - configuring, 66-1
 - default conversion pairs, customizing, 65-11
 - direct, 65-5, 67-1
 - extensions, 65-4
 - jaxb:class support, 65-4
 - list support, 65-3
 - reference descriptor, configuring, 66-2
 - transformation, 65-31, 73-1
 - types of, 65-1
 - union support, 65-3
 - xsd:list, 65-3
 - xsd:union, 65-3
- XML parser platform
 - about, 7-2
 - configuring, 7-3
 - creating, 7-3
 - Crimson, 7-3
 - default, 7-3
 - limitations, 7-3
 - parser conflicts, 7-3
 - toplink.xml.platform system property, 7-3
 - used by, application server, 7-2
 - used by, Oracle TopLink, 7-2
- XML projects
 - configuring, 25-1
 - JAXB support, 20-10
 - sequencing, 20-5
 - understanding, 20-9
- XML queries, 96-5
- XML schema
 - jaxb:class, and EIS mappings, 56-3
 - jaxb:class, and XML mappings, 65-4
 - jaxb:class, understanding, 33-20
 - type, 35-23
 - xs:any, understanding, 65-4
 - xs:anyType, understanding, 65-4
 - xsd:list, understanding, 33-17
 - xsd:union, understanding, 33-17
 - see also* schemas
- XML Type functions, 97-4
- XPath
 - by name, 33-16
 - by position, 33-15
 - mapping Java attributes, 35-10
 - support in OX mappings, 33-15
 - support in XML mappings, 56-3, 65-3
- XSD file

- projects.xml file, 8-2
- sessions.xml file, 8-4

Z

- zero-argument constructors
 - editing, 4-16

