



Sybase<sup>®</sup>  
PowerDesigner<sup>®</sup>  
Physical Data Model

Getting Started

Version 9.5  
38083-01-0950-01  
Last modified: July 2002

Copyright © 2002 Sybase, Inc. All rights reserved.

Information in this manual may change without notice and does not represent a commitment on the part of Sybase, Inc. and its subsidiaries.

Sybase, Inc. provides the software described in this manual under a Sybase License Agreement. The software may be used only in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, SYBASE (logo), AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, ASEP, Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC-GATEWAY, ECMAP, ECRTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, Financial Fusion, Financial Fusion Server, Formula One, Gateway Manager, GeoPoint, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MethodSet, ML Query, MobiCATS, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Rapport, Relational Beans, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RW-DisplayLib, RW-Library, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Versacore, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server and XP Server are trademarks of Sybase, Inc. or its subsidiaries.

All other trademarks are property of their respective owners.

# Contents

<b>About This Book .....</b>	<b>v</b>
<b>1        About the PDM Tutorial.....</b>	<b>1</b>
What you will do.....	2
How long it will take .....	2
What you will learn .....	3
Setting up .....	4
<b>2        How to Begin the PDM Tutorial .....</b>	<b>5</b>
Start PowerDesigner .....	6
Open a new PDM.....	7
Reverse engineer the tutorial PDM.....	9
Use the tools in the Palette .....	11
Define PDM preferences and options .....	16
Arranging symbols in the diagram window .....	19
Save the tutorial PDM .....	20
<b>3        Creating a Table in the PDM .....</b>	<b>21</b>
Add a table.....	23
Add columns .....	25
Creating a domain.....	29
Attaching columns to a domain.....	32
Create columns.....	34
Create a primary key index .....	36
Create an index for a non-key column.....	39
<b>4        Defining a Reference and Referential Integrity .....</b>	<b>43</b>
Create a reference .....	45
Define reference properties .....	46
Define referential integrity.....	48
<b>5        Creating a View .....</b>	<b>51</b>
Compose the view .....	52
Customize the view .....	53

<b>6</b>	<b>Using Referential Integrity Triggers .....</b>	<b>57</b>
	Defining trigger referential integrity .....	59
	Automatic creation of triggers .....	60
	Previewing a trigger .....	65
	Generate a script for triggers .....	67
<b>7</b>	<b>Using Abstract Data Types .....</b>	<b>71</b>
	Specifying an abstract data type as a Java class .....	73
	Create a PowerDesigner Object-Oriented Model .....	76
	Accessing Java class properties .....	77
<b>8</b>	<b>Generating a Database Script .....</b>	<b>81</b>
	Generate a database creation script .....	82
<b>9</b>	<b>Generating a Test Data Script .....</b>	<b>85</b>
	Import test data profiles .....	87
	Create a new test data profile .....	89
	Define a test data profile as a source for automatic test data generation .....	91
	Define a file as a source for test data values .....	94
	Assign test data profiles to selected columns .....	96
	Generate a test data creation script .....	100
<b>10</b>	<b>Designing a Data Warehouse Database Schema .....</b>	<b>103</b>
	Create new model and copy tables .....	104
	Adding a table to the model .....	107
	Define a data source in the current model .....	110
	Create a relational to relational mapping .....	112
	Generate an extraction script .....	119
<b>11</b>	<b>Using Multidimensional Features .....</b>	<b>121</b>
	Retrieve multidimensional objects .....	122
	Rebuilding Cubes .....	124
	Generate Cube Data .....	129
	Exit PowerDesigner .....	131
<b>Glossary</b>	<b>.....</b>	<b>133</b>
<b>Index</b>	<b>.....</b>	<b>137</b>

# About This Book

## Subject

This book contains step-by-step tutorials for the PowerDesigner Physical Data Model modeling environment. It shows you how to do the following:

- ◆ Reverse engineer a Physical Data Model (PDM) from a database creation script
- ◆ Build a Physical Data Model (PDM)
- ◆ Use referential integrity and check parameters
- ◆ Generate database scripts
- ◆ Generate test data scripts
- ◆ Design a data warehouse database
- ◆ Use the multidimensional diagram

## Audience

This book is for anyone who will be building data models with PowerDesigner Physical Data Model. Some familiarity with relational databases, SQL, and design methodology is helpful, but not required. For more information, see the Bibliography section at the end of this chapter.

## Documentation primer

The PowerDesigner modeling environment supports several types of models:

- ◆ **Conceptual Data Model (CDM)** to model the overall logical structure of a database, independent from any software or data storage structure considerations. A valid CDM can be converted to a PDM or an OOM
- ◆ **Physical Data Model (PDM)** to model the overall physical structure of a database, taking into account DBMS software or data storage structure considerations. A valid PDM can be converted to a CDM or an OOM
- ◆ **Object Oriented Model (OOM)** to model a software system using an object-oriented approach for Java or other object languages. A valid OOM can be converted to a CDM or a PDM
- ◆ **Business Process Model (BPM)** to model the means by which one or more processes are accomplished in operating business practices

- ◆ **Free Model (FEM)** to create any kind of chart, diagram, in a context-free environment

This book only contains the basics of the Physical Data Model. For information on other models or aspects of PowerDesigner, consult the following books:

**General Features Guide** To get familiar with the PowerDesigner interface before learning how to use any of the models.

**Conceptual Data Model Getting Started** To learn the basics of the CDM.

**Conceptual Data Model User's Guide** To build a CDM.

**Physical Data Model User's Guide** To build a PDM.

**Object Oriented Model Getting Started** To learn the basics of the OOM.

**Object Oriented Model User's Guide** To build an OOM.

**Business Process Model Getting Started** To learn the basics of the BPM.

**Business Process Model User's Guide** To build a BPM.

**Reports User's Guide** To create reports for any or all models.

**Repository User's Guide** To work in a multi-user environment using a central repository.

**Repository Getting Started** To learn the basics of the Repository.

## Typographic conventions

PowerDesigner documentation uses specific typefaces to help you readily identify specific items:

- ◆ monospace text (normal and **bold**)  
Used for: Code samples, commands, compiled functions and files, references to variables.  
Example: `declare user_defined...`, the **BeforeInsertTrigger** template.
- ◆ UPPER CASE  
Object codes, reversed objects, file names + extension.  
Example: The AUTHOR table appears in the Browser. Open the file OOMAFter.OOM.

- ◆ **bold text**  
Any new term.  
Example: A **shortcut** has a target object.
- ◆ **SMALL CAPS**  
Any key name.  
Example: Press the ENTER key.
- ◆ ***bold italic***  
Tabs, buttons, commands.  
Example: Click the ***Selection*** tab. Select ***File>Open***.

## Bibliography

### Information engineering

James Martin, Prentice Hall, 1990, three volumes of 178, 497, and 625 pages respectively; clothbound, ISBN 0-13-464462-X (vol. 1), 0-13-464885-4 (vol. 2), and 0-13-465501-X (vol. 3).

### Data Modeling Essentials

Graeme Simsion, Van Nostrand Reinhold, 1994, 310 pages; paperbound; ISBN 1850328773

### Celko95

Joe Celko, Joe Celko's SQL for Smarties (Morgan Kaufmann Publishers, Inc., 1995), 467 pages; paperbound; ISBN 1-55860-323-9.





## CHAPTER 1

# About the PDM Tutorial

This tutorial is a series of lessons that explain how to use PowerDesigner to create a Physical Data Model (PDM).

In this tutorial, you reverse engineer a database script into a PDM.

You learn how to denormalize the generated PDM so that you can archive data, speed up access to information in the database, and ensure database integrity. You also learn to design a data warehouse database that will be used as a data source for an OLAP database.

### What is a PDM?

The PDM is a database design tool for defining the implementation of physical structures and data queries.

Depending on the type of database you want to design, you will use different types of diagrams in the PDM.

Database	Diagram
Operational	Physical diagram to define the physical implementation of the database
Date warehouse or Data mart	Physical diagram to store business data
OLAP	Multidimensional diagram to define the possible queries to perform on the operational data

For more information on how to use a PDM, see chapter Physical Data Model Basics in the PDM User's Guide.

## **What you will do**

Chapter 2	<p>You will start PowerDesigner and open a new PDM. You will then reverse engineer a database script. This PDM script presents the physical structure of a publishing enterprise.</p> <p>You will specify model preferences, options, and properties. You will save the model.</p>
Chapter 3	<p>You will modify your PDM by adding a table, and assigning it columns.</p> <p>Next, you will create a primary key for the new table. You will also denormalize the PDM for performance enhancement by creating indexes.</p>
Chapter 4	<p>You will define a reference that indicates how this new table relates to the rest of the database. You will define referential integrity for another reference.</p>
Chapter 5	<p>You will create a view to allow users to view subsets of tables without giving them full access to the tables themselves.</p>
Chapter 6	<p>You will create referential integrity triggers for a selected table and then generate a trigger script.</p>
Chapter 7	<p>You will create an abstract data type and define it as a Java class. You will then link this Java class to a Java class in the PowerDesigner Object-Oriented Model to view its properties.</p>
Chapter 8	<p>You will generate a database creation script for the PDM.</p>
Chapter 9	<p>You will generate a test data script for the PDM.</p>
Chapter 10	<p>You will design a data warehouse database schema.</p>
Chapter 11	<p>You will use the PDM multidimensional diagram.</p>

## **How long it will take**

You can do this tutorial in one sitting of about two hours. You can also stop after any lesson, save your model, and continue at another time.

## What you will learn

You will learn basic PowerDesigner techniques for modifying a PDM, including:

- ◆ How to reverse engineer a database schema into a PDM
- ◆ How to add tables and columns to a PDM, and how to designate primary keys
- ◆ How to create references and define referential integrity
- ◆ How to create indexes
- ◆ How to create and customize a view
- ◆ How to create triggers for a table
- ◆ How to create an abstract data type and link it to a Java class in the PowerDesigner Object-Oriented Model
- ◆ How to generate a database creation script
- ◆ How to generate a test data script
- ◆ How to design a data warehouse database
- ◆ How to define a relational to relational mapping
- ◆ How to generate extraction scripts
- ◆ How to retrieve multidimensional objects
- ◆ How to rebuild cubes
- ◆ How to generate cube data

## Setting up

Before you begin, make sure that the files you need for the exercises are on your hard disk. When you install PowerDesigner, these files are installed in the PowerDesigner 9\Examples\Tutorial directory. When you have finished with this tutorial you can delete them if you want.

Also, if you want to open a PowerDesigner Object-Oriented Model (OOM), you need to install an object-oriented language such as Java.

The PDM tutorial uses the following files:

File	Description
PDMBEFORE.SQL	Starting tutorial PDM script
PDMAFTER.PDM	Finished tutorial PDM (physical features)
PDMBUSIN.PDM	Finished tutorial PDM (data warehouse features)

## CHAPTER 2

# How to Begin the PDM Tutorial

What is reverse engineering?

You will begin the tutorial by running PowerDesigner. You will open an empty PDM, and learn to use the Palette. Then you will reverse engineer the tutorial PDM from a creation script.

During this part of the tutorial, you will add and modify objects in the reversed PDM, and at the end you will generate a new creation script.

Reverse engineering is the process of generating a PDM from an existing database schema. The PDM can be generated from the database creation script, or using an ODBC data source.

Reverse engineering is used when maintaining or modifying an existing database. The PDM presents the database structure in a graphic format, which facilitates the organization and modification of tables, keys, indexes and other database objects.

In this chapter you will:

- ◆ Start PowerDesigner
- ◆ Open a new PDM
- ◆ Reverse engineer the tutorial PDM from a creation script
- ◆ Use the tools in the palette

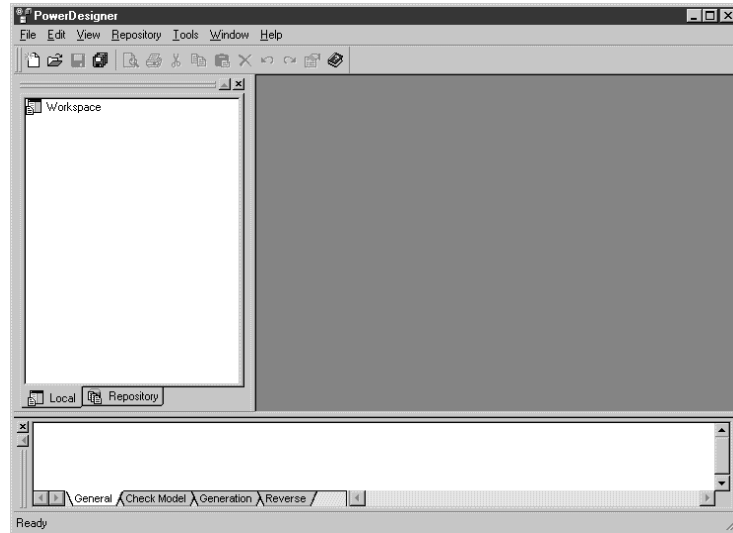
**How long will it take?**

About 10 minutes.

## Start PowerDesigner

- ◆ Click the *PowerDesigner* program icon.

The PowerDesigner main window appears. It contains an object browser window docked to the left, and an output window docked to the bottom of the main window.



The object browser window shows the contents of the workspace in a tree view. You can use the object browser to organize the objects in each of your models.

The **workspace** is the name for the current PowerDesigner session. Workspace is the default node in the object browser tree view. The new PDM that you will open will be created and saved in a workspace.

The output window shows the progression of any process that you run from PowerDesigner, for example the process of generating a database from your PDM is shown in this window.

## Open a new PDM

You will open a new PDM. Each time that you open a new PDM, you must choose a Database Management System (DBMS).

The DBMS definition in PowerDesigner is a set of values that define the SQL characteristics for all objects in your PDM.

- 1 Select **File→New**.

A selection window appears. It lists the types of models that you can open in the PowerDesigner main window.

- 2 Select **Physical Data Model**.

- 3 Click **OK**.

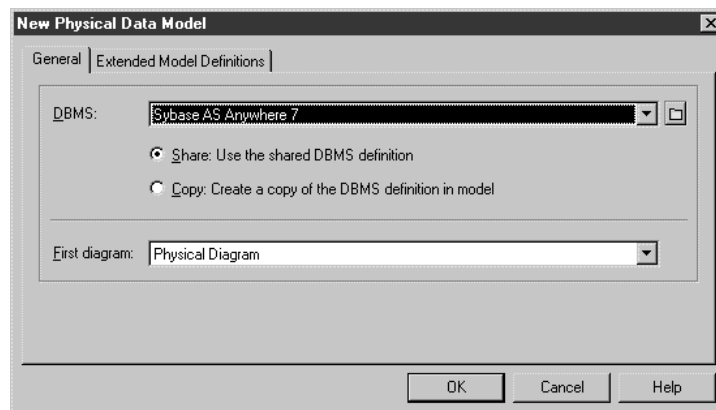
A dialog box appears asking you to choose a DBMS.

- 4 Select **Sybase AS Anywhere 7** from the DBMS dropdown listbox.

- 5 Select the **Share** radio button.

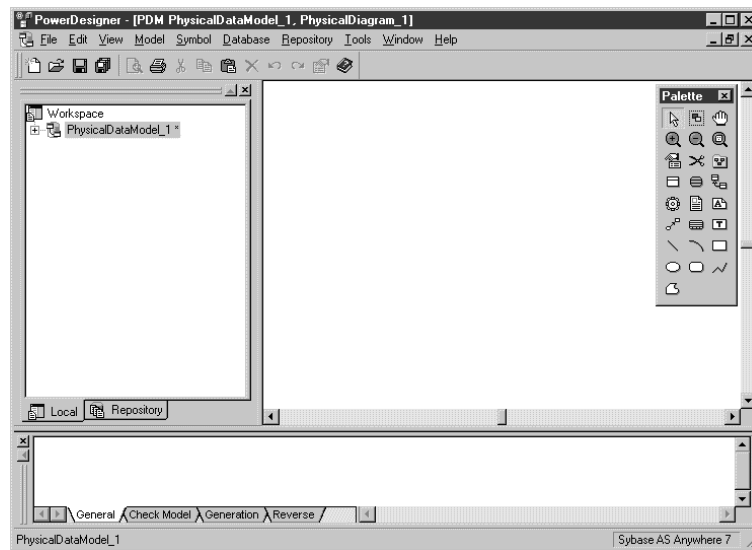
You will use the DBMS definition that is contained in the PowerDesigner DBMS directory.

- 6 Select **Physical Diagram** in the First Diagram dropdown listbox.



- 7 Click **OK**.

A PDM model window appears. It contains an empty diagram window, a palette, and the object browser and output windows are docked to the left and bottom of the screen respectively.



**Your screen looks different**

All the screen captures in this book were taken with a resolution that may be different from the one you use, as a result the appearance and proportions of the images on your screen may be slightly different.



## Reverse engineer the tutorial PDM

You will now reverse engineer the tutorial PDM from a creation script file. A creation script contains SQL creation statements for all the objects in a database. The PDM presents all the objects indicated in the creation script in a graphic format.

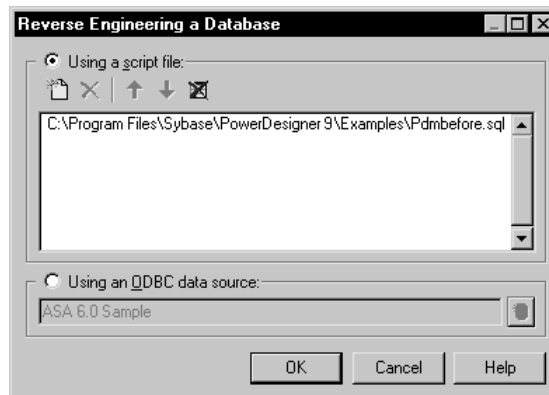
Once you have reversed engineered the tutorial PDM, you will learn to use the tool palette by creating and manipulating several objects in the diagram window.

- 1 Select **Database→Reverse Engineer Database**.

The Reverse Engineering a Database dialog box appears.

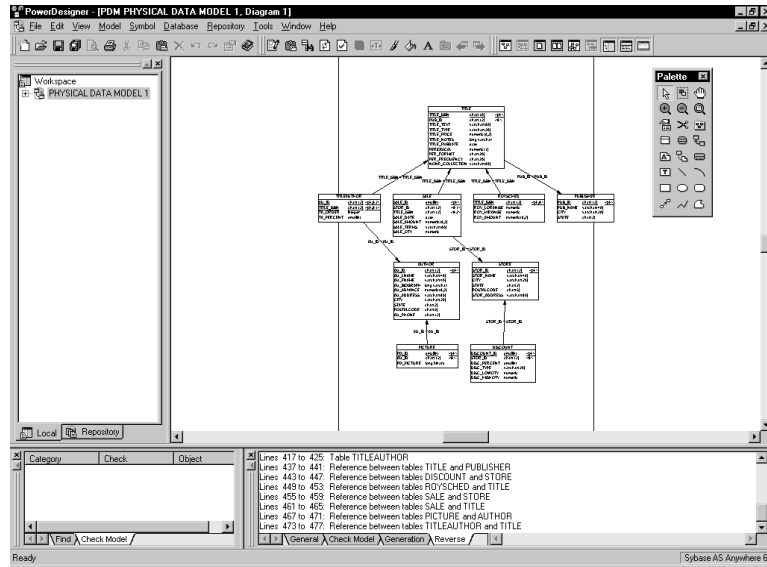
- 2 Select the **Using Script Files** radio button.
- 3 Select the **PDMBEFORE.SQL** file, using the Add Files tool if necessary.

If files other than **PDMBEFORE.SQL** appear in the list, delete them using the Delete File tool.



4 Click **OK**.

The Output window docked at the bottom of the main window shows the progress of the reverse engineering process. When the PDM has been generated, it appears in the diagram window.

**Adjust display scale**












You can choose your preferred display scale by clicking anywhere in the diagram window and selecting View→Scale and choosing a scale














## Use the tools in the Palette

The palette is a tool bar that can be docked anywhere in the workspace. The buttons in the PDM palette present all major functions needed to build and modify a PDM.



The following table indicates the names and use of each tool in the palette:

Tool	Name	Use
	Pointer	Select symbol
	Lasso	Select symbols in an area
	Grabber	Select and move all symbols
	Zoom In	Increase view scale
	Zoom Out	Decrease view scale
	Open Package Diagram	Display diagram for selected package
	Properties	Display property sheet for selected symbol
	Delete	Delete symbol
	Package	Insert package symbol
	Table	Insert table symbol
	View	Insert view symbol

Tool	Name	Use
	Reference	Insert reference symbol
	File	Insert a text file
	Note	Insert note symbol
	Link/Extended Dependency	<p>Draws a graphical link between symbols in the diagram</p> <p>Draws a note link between a Note and an object</p> <p>Draws an extended dependency between two objects that support extended dependencies</p>
	Title	Insert title symbol
	Text	Insert text
	Line	Draw a line
	Arc	Draw an arc
	Rectangle	Draw a rectangle
	Ellipse	Draw an ellipse
	Rounded rectangle	Draw a rounded rectangle
	Polyline	Draw a jagged line
	Polygon	Draw a polygon

You will learn how to use the tools by creating a few objects in the PDM using the palette.

- 1 Click the **Table** tool in the palette.

The cursor takes the form of a table once you move it into the diagram.

- 2 Click anywhere in the PDM diagram window.

A table symbol appears at the click position. The table has the name Table\_*n*, where *n* is a number assigned to the table in the order of creation of objects.



- 3 Click again in the PDM diagram window to create another table.

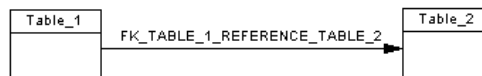


- 4 Click the **Reference** tool in the palette.

The Table tool is now released and the Reference tool is active.

- 5 Click inside the first table and while continuing to hold down the mouse button, drag the cursor to the second table. Release the mouse button inside the second table.

You create a reference between the two tables.



- 6 Click the right mouse button.

You release the Reference tool.

#### Releasing a tool

A tool remains active until you release it. You can release a tool by selecting another tool or by clicking the right mouse button. By default, when you click the right mouse button, the Pointer tool is activated.

- 7 Click the **Lasso** tool in the palette.

The Lasso tool is now active.

- 8 Click the cursor above a corner of the first table and while continuing to hold down the mouse button, drag the cursor so that you draw a rectangle around the two tables.

Release the mouse button.

The tables and the reference are selected. Handles appear on the table symbols to show that they are selected.

- 9 Click one of the selected tables and drag it to a new position.

The selected symbols are moved.

- 10 Click the **Text** tool in the palette.

The Text tool is now active.

- 11 Click the cursor under the reference and while continuing to hold down the mouse button, drag the cursor to draw a small rectangle.  
Release the mouse button.

Some text appears in the area indicated by the rectangle.

- 12 Click the right mouse button.

You release the Text tool.

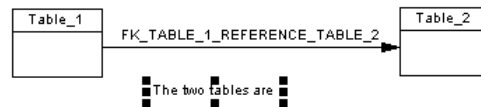
- 13 Double-click the text.

A text box appears.

- 14 Type a short text into the text box.

- 15 Click OK.

The text appears in the diagram. Handles appear around the text.

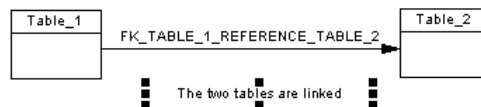


- 16 Click a handle at the right edge of the text and while continuing to hold down the mouse button, drag the cursor to the right until all the text appears.

Release the mouse button.

Click on the diagram background.

The handles around the text disappear.



- 17 Click the **Pointer** tool in the palette.

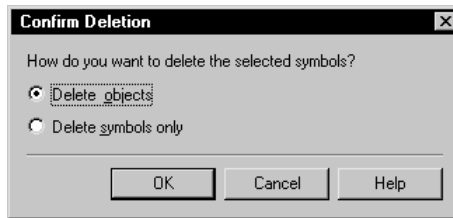
You will use this tool to select and delete one of the symbols.

- 18 Click on one of the table symbols.

This selects the object you want to delete.

- 19 Press the DEL key.

The Confirm Deletion message box appears, asking you how you want to delete the selection.



#### Deleting objects

If you select Delete object, you erase the graphic symbol and delete the object from the model. If you select Delete symbols only, you erase the graphic symbol, but keep the object in the model.

- 20 Click OK.

The table and associated reference are removed from the diagram. The objects are also deleted from the model.

- 21 Click in the remaining table.  
Press SHIFT while you click the text.

The two objects are selected.

- 22 Press the DEL key, and click OK when the deletion message appears.

The remaining table and text are erased.


#### What you learned

In this section, you learned how to use some of the tools in the palette. You can now:

- ◆ Select a tool
- ◆ Release the active tool either by selecting another tool or by clicking the right mouse button
- ◆ Select a group of objects
- ◆ Move graphic objects
- ◆ Create text to document the PDM
- ◆ Delete objects

## Define PDM preferences and options

Before you begin working, you will define certain display preferences and model options for the PDM.

 For a complete description of all PDM preferences and options, see the *General Features Guide*.

- 1 Select **Tools→Display Preferences** from the menu bar.

The Display Preferences dialog box appears.

- 2 Select the **Table** node, under the **Object view** node, in the **Category** tree view.

The Table page appears.

- 3 Select or clear the following display preferences:

Group box	Selected item
Table	Columns
	All columns
Table column	Key indicators
	Index indicators

For each table symbol, these preferences display all columns of a table, and identify all key and index columns.

- 4 Select the **Reference** node, under the **Object View** node, in the **Category** tree view

The Reference page appears.

- 5 Select or clear the following display preferences:

Group box	Selected item
Center Text	-
End Text	Cardinality
Display Mode	Relational

The cardinality of the reference is displayed for each reference symbol.

- 6 Select the **View** node, under the **Object View** node, in the **Category** tree view.

The View page appears.



- 7 Select or clear the following display preferences:

Group box	Selected item
View	Columns
	No formulas
	Tables
View columns	Name

For each view symbol, these preferences display all view columns, tables, and indicate the name for each view column.

- 8 Click the **Format** node.

The Format page appears.

- 9 Click the **Modify** button at the bottom right of the page.

The Symbol Format page appears.

- 10 Verify that the Auto adjust to text checkbox is selected.

- 11 Click **OK**.

You return to the Display Preferences dialog box.

- 12 Click **OK**.

You return to the PowerDesigner main window.

- 13 Select **Tools→Model Options** from the menu bar.

The Model Options dialog box appears. The Model node is selected by default in the Category tree view.

- 14 Select or clear the following model options:

Group box	Selected item
Column & Domain	Enforce non-divergence
	Data type
	Default data type: Undefined
Reference	Unique code
	Auto-migrate columns
	Default link on creation: Primary key

For the entire model, these preferences enforce non divergence of data type for columns and domain. References are set as having a unique code and auto-migrating columns is activated.

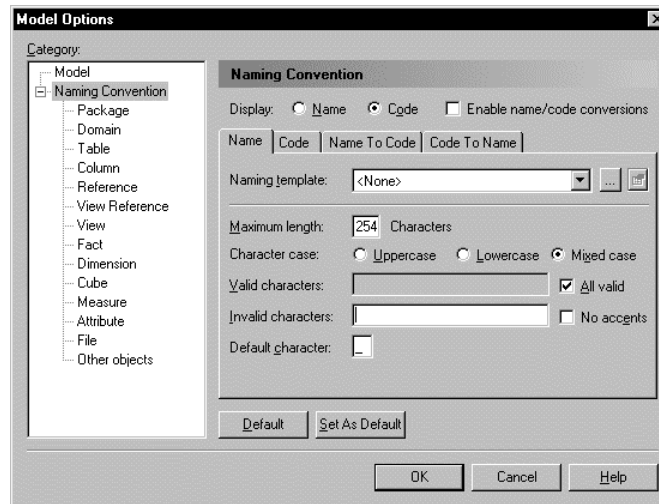
#### Auto-migrating foreign keys

When Auto-migrate columns is selected, the primary key in a parent table automatically migrates to the child table as a foreign key when you create a reference between the two tables

- 15 Click the **Naming Convention** node.

The Naming Convention page appears.

- 16 Select the Code radio button.



You display codes in the object symbols in the diagram.

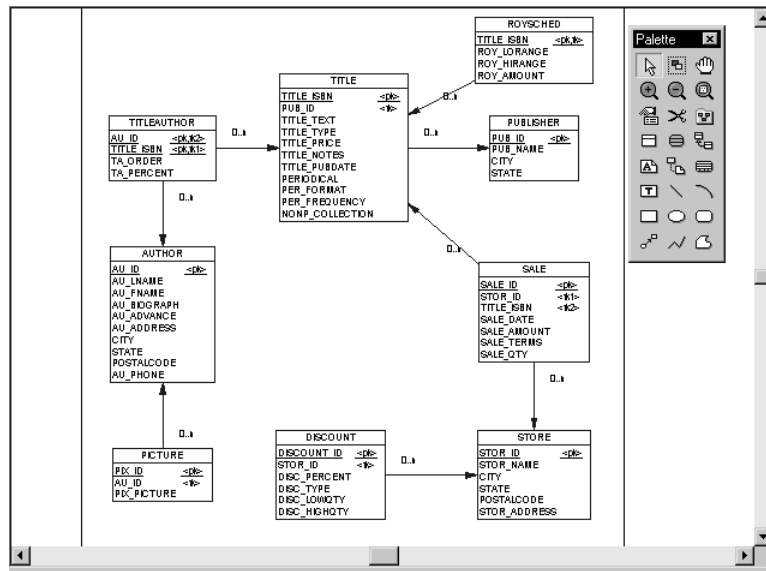
- 17 Click **OK**.

## Arranging symbols in the diagram window

You can arrange objects displayed in the diagram window by selecting a symbol and dragging it to a new position. When a model has been reversed engineered, you often need to rearrange the symbols to improve the readability of the display.

You will arrange the table symbols in the diagram.

- ◆ Arrange the table symbols in the diagram by clicking on a symbol and dragging it to a new position so the model appears in the diagram window as shown below:



### Aligning multiple symbols

To align multiple symbols, press **SHIFT** while clicking on the symbols that you want to arrange and then select **Symbol→Align** and choose an align option from the menu.

## Save the tutorial PDM

You will save the tutorial PDM in another file. This leaves the original tutorial PDM intact so you can use it again if you want to redo the exercises.

- 1 Select **File**→**Save As**.

The File Save As dialog box appears.

- 2 Type **TUTORIAL.PDM** in the File Name box.


This is the name of the file in which you will work and save your modifications.

- 3 Click **Save**.

### Save your work

Save your work periodically while doing these exercises by selecting File→Save.

## Creating a Table in the PDM

	<p>When maintaining a database, you often need to create new tables in a PDM to store new data as a result of development or modifications in the information system.</p> <p>You will create a table in the PDM. This table will be named HISTORY and will contain a record of all sales.</p>
Adding tables to the PDM	<p>The table HISTORY stores sales data but has no role in the functional structure of the database. When you must add tables that are part of the functional structure of a database, do it at the Conceptual Data Model (CDM) level before generating the new tables in a PDM.</p> <p> For more information on the CDM, see the <i>Conceptual Data Model User Guide</i>.</p>
About keys	<p>You will also add columns, designate a primary key for the table, create a domain, create a column, and create indexes.</p> <p>A key is a column, or several columns, in which each of the values correspond to one unique row in the table. A <b>primary key</b> is a key that has been designated as the primary identifier for a table. All tables must have a primary key composed of one or more of its columns.</p> <p>For example, the primary key for an author table can be author last names.</p>
About domains	<p>A domain defines a standard data structure that you can apply to multiple columns. When you modify a domain you globally update the columns associated with the domain. This makes it easier to standardize data characteristics and modify your model consistently when you need to make changes.</p> <p>For example, the domain for author last names can be defined as a set number of characters.</p>
About indexes	<p>An index is a data structure that speeds up access to data when you are searching for information in tables. Indexes are normally created for columns that are accessed regularly, and where response time is important.</p>

	<p>Indexes are most effective when they are used on columns that contain mostly unique values. When a column is attached to an index, the index orders the column rows in such a way so it is more efficient for the search process to find rows matching a search criteria, than to scan through the data contained in each row.</p> <p>For example, an index can be the primary key column containing author last names.</p>
Primary and foreign key indexes	<p>It is common to index primary key columns as they are often searched and contain only unique values. You will create an index for the primary key column in the HISTORY table.</p>
Indexes on non-key columns	<p>You can also create indexes for other columns, depending on the type of information you want to access in the database.</p> <p>For example, to find an author in the database, you can use author last names to index the columns which contain this information. You will create an index for the column Author Last Name in the Author table.</p> <p>In this chapter you will:</p> <ul style="list-style-type: none"><li>◆ Add a table</li><li>◆ Add columns to the table</li><li>◆ Designate a primary key</li><li>◆ Create a domain</li><li>◆ Add existing columns to the domain</li><li>◆ Create a column</li><li>◆ Create an index on a primary key</li><li>◆ Create an index on a non-key column</li></ul>

**How long will it take?**

About 15 minutes.

## Add a table

You will add the HISTORY table to the PDM.

- 1 Click the **Table** tool in the palette.
- 2 Click in the **diagram** underneath the TITLE table symbol.

A table symbol appears at the click position.

TITLE	
TITLE_ISBN	<pk>
TITLE_TEXT	
TITLE_TYPE	
TITLE_PRICE	
TITLE_NOTES	
TITLE_PUBDATE	
PERIODICAL	
PER_FORMAT	
PER_FREQUENCY	
NONP_COLLECTION	

Table_10

The table has the code Table\_*n*, where *n* is a number assigned in the order of creation of objects.

- 3 Click the **Pointer** tool in the palette.
- 4 Double-click the new table symbol.

The Table Properties dialog box appears.

- 5 Type **HISTORY** in the Name box.

This is the name of the table. HISTORY appears automatically in the Code box. This is the code for the table.

Table Properties - HISTORY (HISTORY)

Options Preview Mapping Notes Rules

Dependencies Extended Dependencies Version Info

General Columns Indexes Keys Triggers Procedures Check Script

Name: HISTORY

Code: HISTORY

Comment:

Owner: <None>

Number: Generate: ☒

Type: <None>

OK Cancel Apply Help

- 6 Click **OK**.

HISTORY

What you learned

In this section, you learned how to:

- ◆ Customize a PDM by adding a table to store data



## Add columns

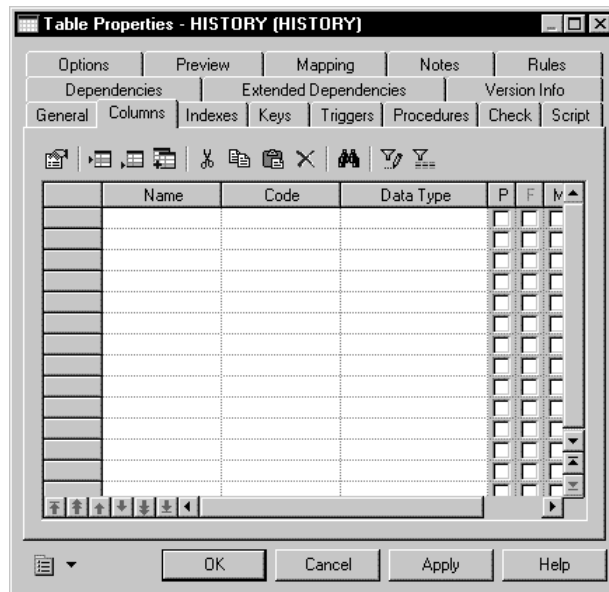
You will add the columns `TITLE_ISBN`, `TITLE_PRICE`, and `TITLE_TEXT` to the `HISTORY` table.

- 1 Double-click the **HISTORY** table symbol.

The Table Properties dialog box appears.

- 2 Click the **Columns** tab.

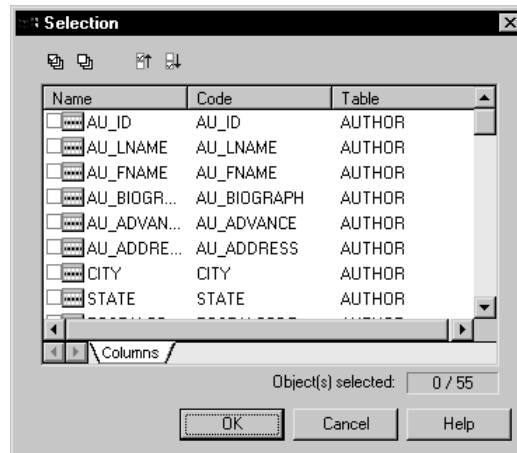
The Columns page appears. The table does not yet have any columns so the list is empty.



- 3 Click the **Add Columns** tool.



The Selection dialog box appears. It shows the name and code for each column in the model, and the table to which it belongs.



- 4 Click the **Table** column header to sort the columns alphabetically by table.
- 5 Scroll down the list to the columns in the **TITLE** table.
- 6 Select the **TITLE ISBN** checkbox.  
Select the **TITLE PRICE** checkbox.  
Select the **TITLE TEXT** checkbox.



- 11 Select the **Primary key** checkbox in the bottom part of the General page.

This indicates that Title ISBN is the primary key. The Foreign Key and Mandatory Key checkboxes are grayed. Mandatory values translate to a Not null field in most SQL databases.

- 12 Click **OK**.

You return to the property sheet for the HISTORY table.

- 13 Click **OK**.

This saves the column definition. Column TITLE\_ISBN has the symbol <pk> next to it. This indicates that it is a primary key column.

HISTORY	
TITLE_ISBN	<pk>
TITLE_PRICE	
TITLE_TEXT	

## What you learned

In this section, you learned how to:

- ◆ Add existing columns to a table in the PDM
- ◆ Define a primary key

## Creating a domain

You will create a domain to assign a standardized data type for money amounts.

- 1 Select **Model→Domains**.

The List of Domains appears. It lists all the domains defined in the model.

- 2 Click the **Add a Row** tool.



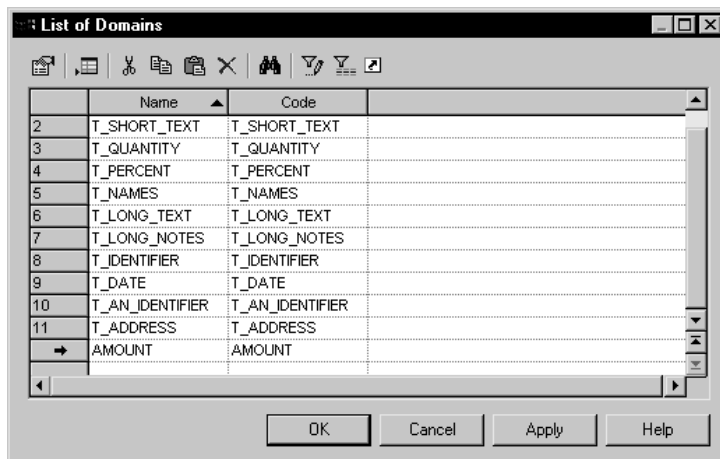
An arrow appears at the start of the first empty line and a default name and code are entered.

### A default name and code appear

When you create a new list item, a default name and code automatically appear for the new item. If the default name is selected it disappears when you start to type the item name. If the default name is not selected, select it and either type the new item name, or delete the default name before typing the new item name.

- 3 Type **AMOUNT** in the Name column.

This is the name of the domain.



- 4 Click **Apply**.

The creation of the new domain is committed.

**Names sorted alphabetically**

When you click **Apply** or **OK**, all names in the list are sorted alphabetically. The order of appearance of the names in the list will therefore change with either of these operations.

- 5 Click the new domain line.

An arrow appears at the beginning of the line.

- 6 Click the **Properties** tool.



or

Double click the arrow at the beginning of the line.

The property sheet for the new domain appears.

- 7 Click the **?** button next to the **Data Type** dropdown listbox.

The Standard Data Types dialog box appears. You can use this dialog box to specify the form of the data affected by the domain.

**Selecting a data type directly**

You can also select a data type directly by clicking the arrow at the end of the Data Type dropdown listbox, and selecting a data type from the dropdown list.

- 8 Click the **Number** radio button.

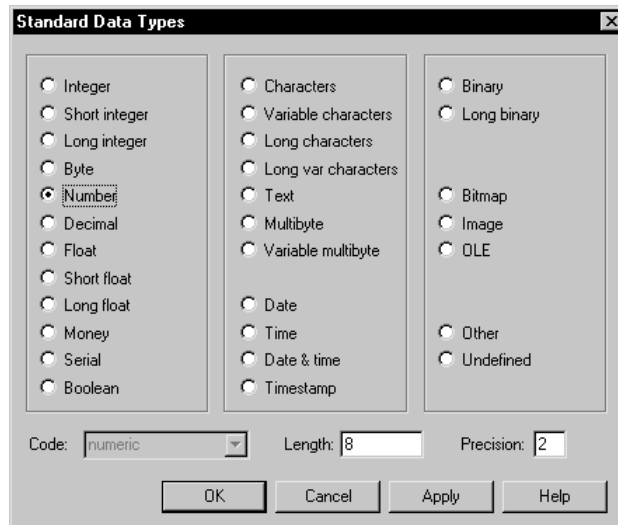
The domain now has a Number data type.

- 9 Type **8** in the Length box.

The maximum number of figures in a column attached to this domain will be 8.

- 10 Type **2** in the Precision box.

The maximum number of positions after the decimal point in a column attached to this domain will be 2.



- 11 Click **OK**.

You return to the domain property sheet. The value **numeric 8,2** appears in the Data Type dropdown listbox. Numeric is the code for a money data type, 8 indicates that an amount of money can have eight figures, 2 indicates that the amount has a decimal precision of two.

- 12 Click **OK** in each of the dialog boxes.

You return to the PDM diagram.

## Attaching columns to a domain

You will attach the AMOUNT domain to all columns that store amounts of money.

- 1 Select **Model→Columns**.

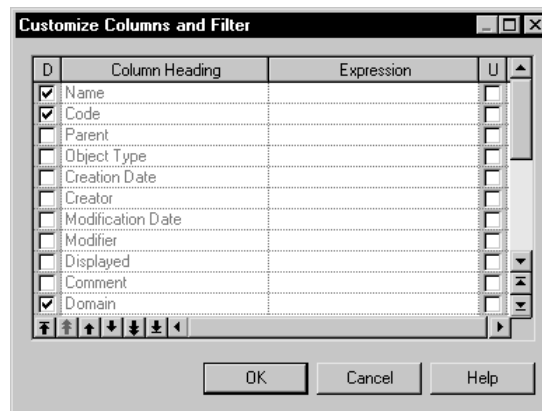
The List of Columns appears.

- 2 Click the **Customize Columns and Filter** tool from the tool bar at the top of the list.



The Customize Columns and Filter dialog box appears. You can select or clear check boxes for columns that you want to appear, or not to appear, in the list.

- 3 Select the **Domain** and **Table** checkboxes.



- 4 Click **OK**.

You return to the List of Columns. The column Domain appears in the list of columns. It lists the domain used by each column in the model.

### Enlarging dialog boxes

You can enlarge dialog boxes and lists by placing the cursor on a dialog box or list edge so that the cursor becomes a double headed arrow. Click and while continuing to hold down the mouse button, drag the edge in the direction that you want the dialog box or list to be enlarged.



- 5 Click the **Table** column header.  
The columns are ordered by table.
- 6 Click the **Domain** column in the **AU\_ADVANCE** line.  
An arrow appears in the domain column of the line.
- 7 Click the arrow.  
A dropdown list appears. It lists all the domains defined in the model.
- 8 Scroll down the list and select **AMOUNT**.  
AMOUNT is listed as the domain for the column AU\_ADVANCE.
- 9 Scroll down the list of columns to the **ROY\_AMOUNT** line.
- 10 Click the **Domain** column in the **ROY\_AMOUNT** line.  
An arrow appears in the domain column of the line.
- 11 Click the arrow.  
A dropdown list appears. It lists all the domains defined in the model.
- 12 Scroll down the list and select **AMOUNT**.
- 13 Repeat steps 9 to 12 for the following columns:

Column	Table	Domain
SALE_AMOUNT	SALE	AMOUNT
TITLE_PRICE	TITLE	AMOUNT

- 14 Click **OK**.  
You return to the model diagram.

#### What you learned

In this section, you learned how to:

- ◆ Create a new domain
- ◆ Customize the items displayed in a list
- ◆ Attach columns to the new domain

## Create columns

You will create a new column in the **HISTORY** table.

- 1 Double-click the **HISTORY** table symbol.

The table properties dialog box appears.

- 2 Click the **Columns** tab.

The Columns page appears.

- 3 Click the **Insert a row** tool.



An arrow appears at the start of the first empty line and a default name and code are entered.

- 4 Type **TOTAL SALES** in the **Name** column of the first blank line.

TOTAL\_SALES appears automatically in the Code column.

- 5 Click **Apply**.

The creation of the column is committed.

- 6 Click the **TOTAL SALES** line.

An arrow appears at the start of the line.

- 7 Click the **Properties** tool.



or

Double click the arrow at the beginning of the line.

The property sheet for the new column appears.

- 8 Select **AMOUNT** from the Domain dropdown listbox in the bottom part of the dialog box.

The data type column displays numeric 8,2 which is a data type available for the target database.

- 9 Click **OK** in each of the dialog boxes.

You return to the PDM diagram. The table symbol displays the new column.

HISTORY	
TITLE_ISBN	<pk>
TITLE_PRICE	
TITLE_TEXT	
TOTAL SALES	

### What you learned

In this section, you learned how to:

- ◆ Create a new column
- ◆ Identify a column by a name and a code
- ◆ Attach the new column to a domain

## Create a primary key index

You will define a primary key index for the HISTORY table.

- 1 Double-click the **HISTORY** table.

The Table Properties dialog box appears.

- 2 Click the **Indexes** tab.

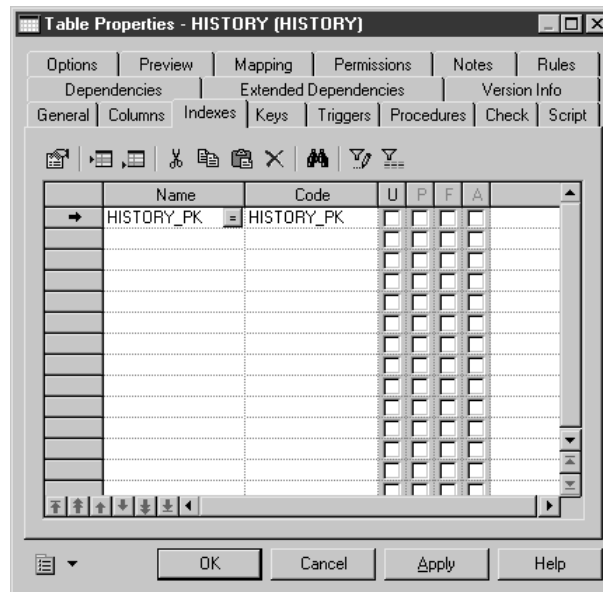
The Indexes page appears and is currently empty.

- 3 Click the first **blank line** in the list.

An arrow appears at the start of the line and a default name and code are entered.

- 4 Type **HISTORY\_PK** in the **Name** column.

HISTORY\_PK appears automatically in the Code column.



- 5 Click **Apply**.

The creation of the index is committed.

- 6 Click the **HISTORY\_PK** line.

An arrow appears at the start of the line.

- 7 Click the **Properties** tool.



or

Double click the arrow at the beginning of the line.

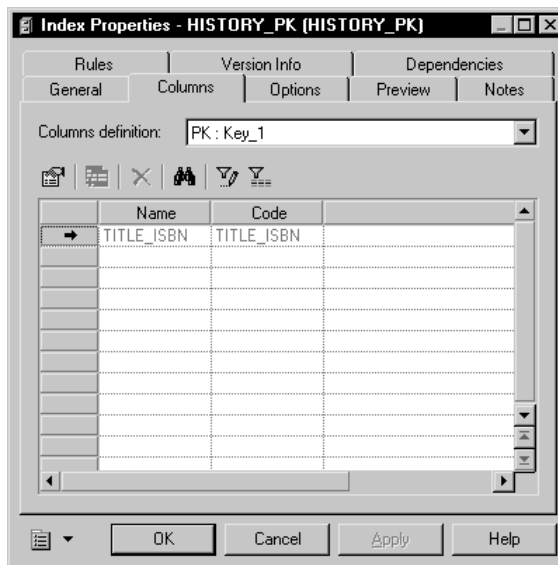
The property sheet for HISTORY\_PK opens to the General page.

- 8 Click the **Columns** tab.

The Columns page appears. It shows that there are no columns attached to the index. The Column Definition dropdown listbox lists the primary and alternate keys defined for the table. When a key is selected in this box, its columns are automatically listed in the columns list.

- 9 Select **PK: Key\_1** from the **Columns Definition** dropdown listbox.
- 10 Click **Apply**.

TITLE\_ISBN appears in the column list.



- 11 Click **OK**.

You return to the table property sheet.

- 12 Scroll to the right until the **P** column appears.

The **P** checkbox at the end of the HISTORY\_PK line is selected. This indicates that the indexed column is a primary key column.

**Make the P column visible**

If the P column is not visible, you can make it appear by selecting the Customize columns and filter tool from the list tool bar. From the selection box that appears, select or clear the appropriate checkbox for the item that you want to appear or not appear in the list.

13 Click **OK**.

You return to the PDM diagram. The index indicator <i> appears next to the primary key indicator <pk> in the table symbol.

HISTORY		
TITLE_ISBN	<pk>	<i>
TITLE_PRICE		
TITLE_TEXT		
TOTAL SALES		

## Create an index for a non-key column

The column that contains the last name of the author is not a primary key column. However, indexing this column would allow you to search for an author more quickly when you only know the author's last name. You will create an index for this column.

- 1 Double-click the **AUTHOR** table.

The Table Properties dialog box appears.

- 2 Click the **Indexes** tab.

The Indexes page appears. It shows that this table has no indexes.

- 3 Click the first **blank line** in the list.

An arrow appears at the start of the first empty line and a default name and code are entered.

- 4 Type **AU\_LNAME\_IDX** in the **Name** column.

AU\_LNAME\_IDX appears automatically in the Code column. This is the name of the index for the last names of the authors.

- 5 Click **Apply**.

The creation of the index is committed.

- 6 Click the **AU\_LNAME\_IDX** line.

An arrow appears at the start of the line.

- 7 Click the **Properties** tool.



or

Double click the arrow at the beginning of the line.

The Index property sheet appears.

- 8 Click the **Columns** tab.

The Columns page appears. The index does not yet have any columns attached, so the list is empty.

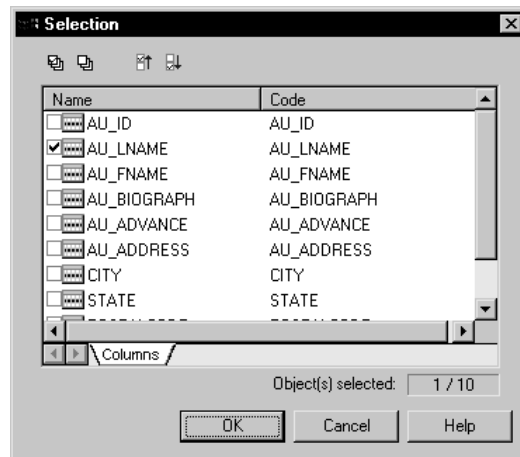
- 9 Click the **Add Columns** tool.



A selection box appears. It lists all the columns in the table.

- 10 Select **AU\_LNAME**.

This is the column to which you want to attach the AU\_LNAME\_IDX index.



- 11 Click **OK**.

You return to the columns page. AU\_LNAME is listed in the column list.

- 12 Scroll to the right until the **Sort** column appears.

**Make the sort column visible**

If the sort column is not visible, you can make it appear by selecting the Customize columns and filter tool from the list tool bar. From the selection box that appears, select or clear the appropriate checkbox for the item that you want to appear or not appear in the list.

- 13 Make sure **Ascending** is selected in the Sort column for AU\_LNAME line. This indicates that the index sorts the last names of authors in ascending alphabetical order from A to Z.



- 14 Click **OK** in each of the dialog boxes.

You return to the PDM diagram. The index indicator <i> appears next to the AU\_LNAME column in the table symbol.

AUTHOR	
AU_ID	<pk>
AU_LNAME	<i>
AU_FNAME	
AU_BIOGRAPH	
AU_ADVANCE	
AU_ADDRESS	
CITY	
STATE	
POSTALCODE	
AU_PHONE	

#### What you learned

In this section, you learned how to:

- ◆ Speed access to information in the database by creating indexes for the columns storing relevant data
- ◆ Create an index for a primary key and a non-key column
- ◆ Identify an index by a name and a code
- ◆ Select the columns you want to index
- ◆ Define how values inside the indexed columns are sorted



## CHAPTER 4

# Defining a Reference and Referential Integrity

### About references

A **reference** is a link between a parent table and a child table. It defines a referential integrity constraint between primary key, or alternate key, column pairs and a foreign key, or between user-specified columns in both tables.

You create a reference when you want one or more columns in a table to refer to one or more columns in another table.

Within a reference, a **join** links each foreign key column that refers to a matching primary key column (these are called a column pair). Depending on the number of columns in a primary key, a reference can contain one or more joins.

For this tutorial, you will create a reference that links the primary key in the TITLE table to a foreign key in the HISTORY table.

### About referential integrity

**Referential integrity** dictates what happens to a foreign key column in a child table when you update, or delete, the value of the corresponding primary key column in the parent table.

For example, in the tutorial PDM, a reference exists between the STORE and DISCOUNT tables.



Stor\_ID is the primary key column in the STORE table. It contains the unique identification code of a store. A reference links the value of Stor\_ID in the STORE table to the Stor\_ID column in the DISCOUNT table.

Using the referential integrity options, you can specify that if you delete a store from the STORE table, you also delete all of its corresponding records in the DISCOUNT table.

In this lesson you will:

- ◆ Create a reference
- ◆ Define the reference properties
- ◆ Define referential integrity

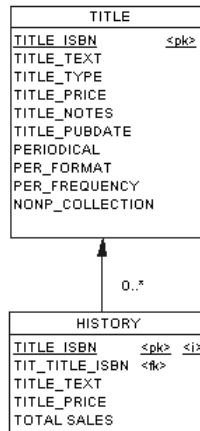
**How long will it take?**

About 5 minutes.

## Create a reference

You will create a reference between the HISTORY and TITLE tables. This reference represents the sales history for each title.

- 1 Click the **Reference** tool in the palette.
- 2 Click inside the HISTORY table and while continuing to hold down the mouse button, drag the cursor into the TITLE table and release the mouse button. You have created a **reference** link from HISTORY to TITLE.



HISTORY is the child table and TITLE is the parent table. TITLE\_ISBN is the primary key of both tables. TIT\_TITLE\_ISBN appears as a new foreign key in the HISTORY table as indicated by the symbol <fk>.

- 3 Click the Pointer tool to release the Reference tool.

### Auto-migrating a foreign key

The primary key TITLE\_ISBN is automatically migrated and linked to the HISTORY table as a foreign key when you create the reference. This is caused by having previously selected Auto-Migrate Columns (FK) as a model option.

### What you learned

In this section, you learned how to:

- ◆ Create a reference between two tables
- ◆ Identify the parent and child tables in a reference

## Define reference properties

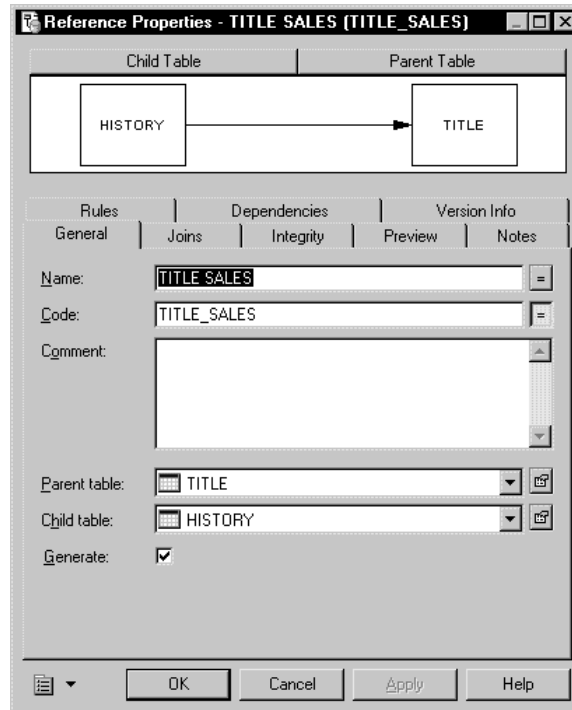
You can define a name for a reference to make it easier to identify.

- 1 Double-click the **reference** link between HISTORY and TITLE.

The Reference Properties dialog box appears.

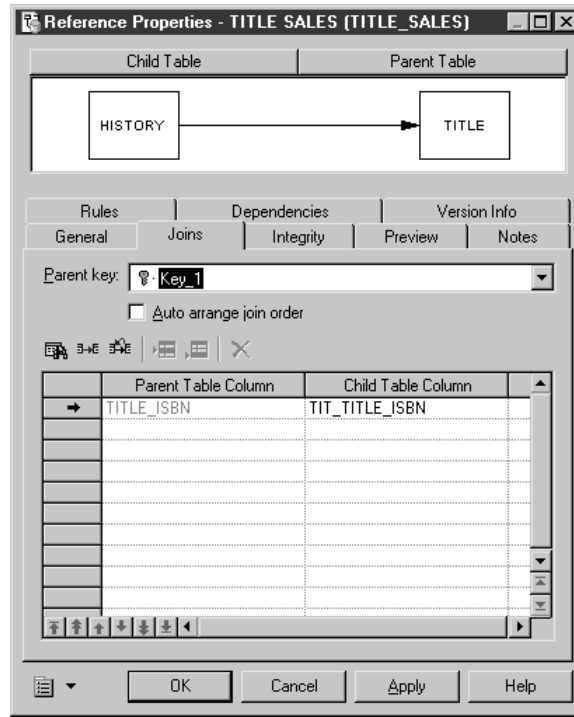
- 2 Type **TITLE SALES** in the Name box.

TITLE SALES appears automatically in the Code box.



- 3 Click the **Joins** tab.

The Joins page appears. The column list displays the parent table which contains the primary key column, and the child table which contains the foreign key column. These two columns are linked by the join within the reference.



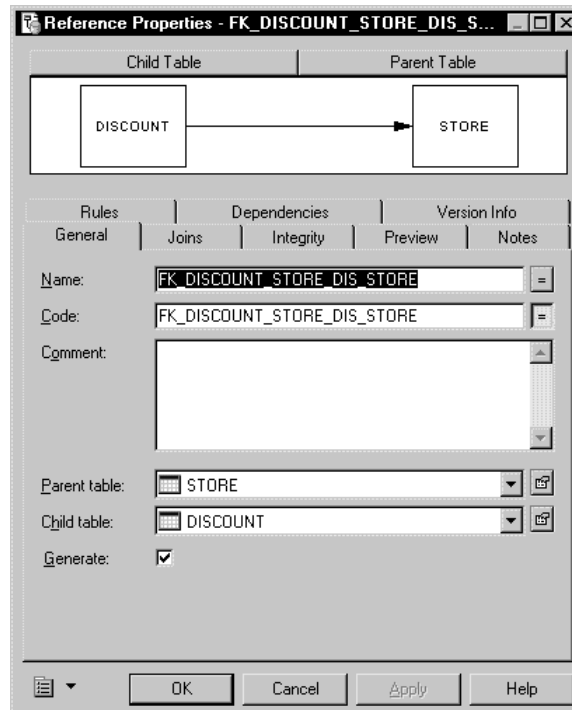
- 4 Click **OK**.

## Define referential integrity

You will use the referential integrity options to implement **cascade** update when a store is deleted. Since discounts are specific to store, if you delete a store you delete its associated discounts.

- 1 Double-click the *reference* link between DISCOUNT and STORE.

The reference property sheet appears.



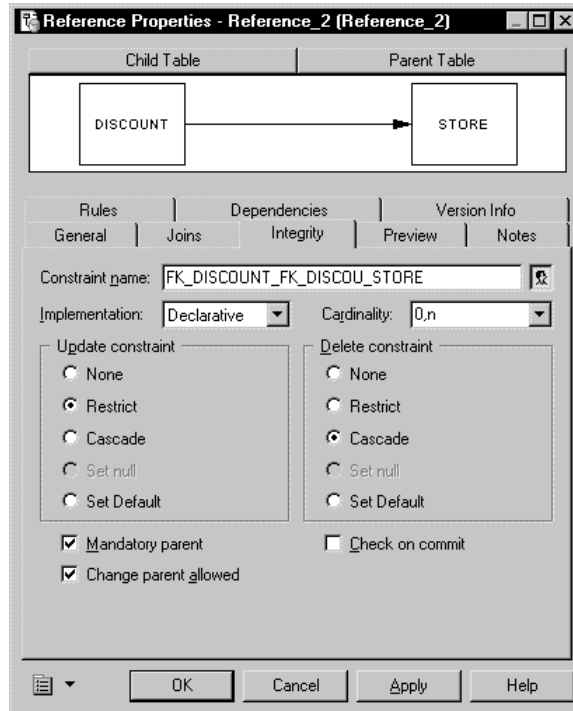
- 2 Click the *Integrity* tab.

The Integrity page appears. The Restrict radio buttons are selected by default in the Update Constraint and Delete Constraint group boxes. They indicate that an error message is produced when a value in a primary key column is updated, or deleted, while there are corresponding foreign key columns.



- 3 Click the **Cascade** radio button in the Delete Constraint groupbox.

Cascade indicates that when a primary key value is modified or deleted, the corresponding foreign key value is automatically updated or deleted. So, if you delete the Store ID in the STORE table, you also delete the corresponding records in the DISCOUNT table.



- 4 Click **OK**.

You return to the diagram.

#### What you learned

In this section, you learned how to:

- ◆ Define what happens to related values in child tables when you delete or modify a value in the parent table (referential integrity)



## CHAPTER 5

# Creating a View

### About views

A **view** is an alternative way of looking at the data in one or more tables. A view contains a subset of columns from one or more tables. You can create a view to allow users to see subsets of tables without giving them full access to the tables themselves.

### What happens when you create a view

Creating a view is equivalent to defining a SQL `SELECT` query to select objects in the database. When you create a view you select the tables and columns you want to include in the view. In this lesson, you will create a view on the `TITLE` and `SALES` tables. This view generates a SQL query which automatically selects all the columns in these tables and displays a graphic symbol representing the view.

In this lesson you will:

- ◆ Compose the view
- ◆ Customize the view

#### How long will it take?

About 5 minutes.

## Compose the view



You will compose a view of the **TITLE** and **SALES** tables.

- 1 Select the **TITLE** table symbol.
- 2 Press SHIFT while you click the **SALE** symbol and the reference between both tables.

Both table symbols are selected together with the reference symbol.

- 3 Select **Tools**→**Create View** from the menu bar.

A view appears in the PDM workspace. The view lists all the columns belonging to the selected tables. At the bottom of the view, it lists the tables.

View_1	
SALE_ID	
STOR_ID	
SALE_TITLE_ISBN	
SALE_DATE	
SALE_AMOUNT	
SALE_TERMS	
SALE_QTY	
TITLE_TITLE_ISBN	
PUB_ID	
TITLE_TEXT	
TITLE_TYPE	
TITLE_PRICE	
TITLE_NOTES	
TITLE_PUBDATE	
PERIODICAL	
PER_FORMAT	
PER_FREQUENCY	
NONP_COLLECTION	
 SALE	
 TITLE	

The view has the code VIEW\_*n*, where *n* is a number assigned in the order of creation of objects.

## Customize the view

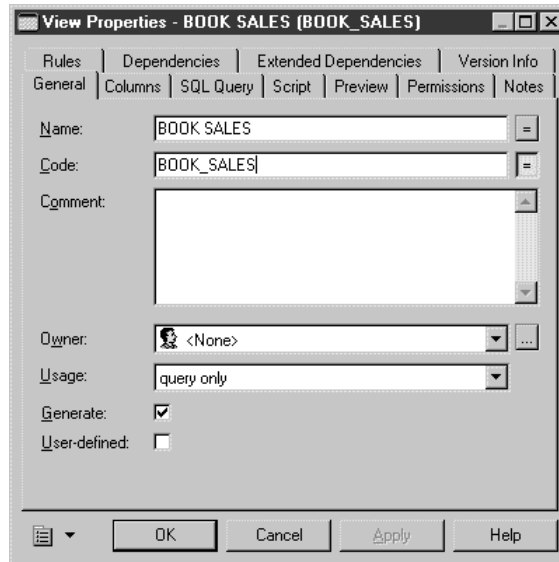
You will customize the view so that it contains only certain columns.

- 1 Double-click the View in the diagram.

The view property sheet appears.

- 2 Type **BOOK SALES** in the Name box.

BOOK\_SALES appears automatically in the Code box.

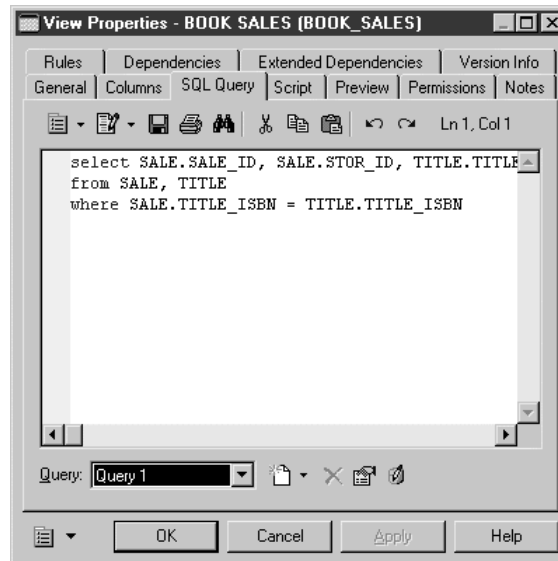


- 3 Select **Query Only** in the Usage dropdown listbox.

This limits access to consultation only.

- 4 Click the **SQL Query** tab.

The SQL Query page appears. It shows the definition of the view.



- 5 Click the **Properties** tool.



The Query Properties page appears.

- 6 Click the **Columns** tab.

The Columns page appears. It lists all the columns contained in the view.

- 7 Press CTRL and select all column names listed in the **Expression** column except for the following: **TITLE.TITLE\_ISBN**, **TITLE.TITLE\_TEXT**, **SALE.SALE\_AMOUNT**, and **SALE.SALE\_QTY**.

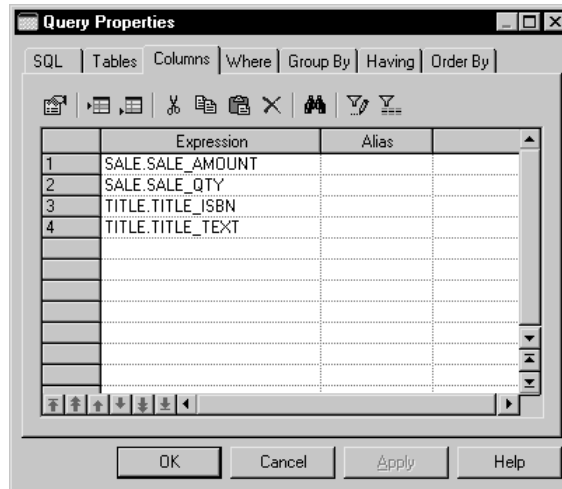
- 8 Click the **Delete** tool.



or

Press DEL.

The columns **TITLE.TITLE\_ISBN**, **TITLE.TITLE\_TEXT**, **SALE.SALE\_AMOUNT**, and **SALE.SALE\_QTY** are listed in the column list.

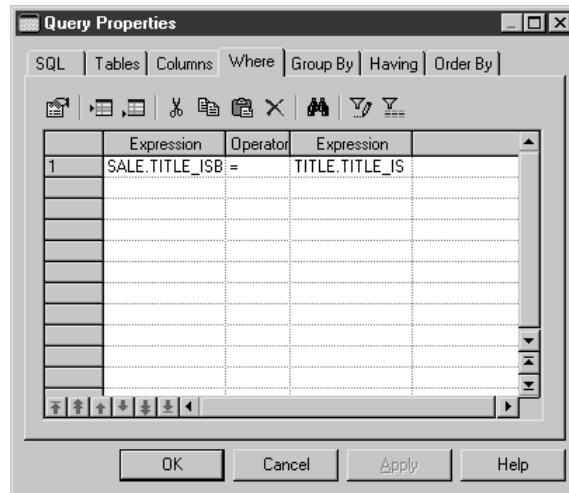


- 9 Click the **Apply** button.

The modifications are committed.

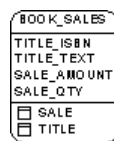
- 10 Click the **Where** tab.

The Where page shows the link between SALE and TITLE tables by TITLE\_ISBN.



- 11 Click **OK** in each of the dialog boxes.

You return to the PDM workspace. The view now only lists the selected columns.



## What you learned

In this section, you learned how to:

- ◆ Restrict access to information further by including only certain columns in a view



## Using Referential Integrity Triggers

### About triggers

A **trigger** is a segment of SQL code associated with a table, and stored in a database. It is invoked automatically whenever there is an attempt to modify data in the associated table with an insert, delete, or update command.

Triggers can enforce referential integrity. For example, by displaying an error message if you try to update a primary key column which has an update restriction.

PowerDesigner can create triggers for selected tables automatically based on trigger referential integrity defined for a reference. You can also define a trigger manually for a table.

You can generate a SQL script containing the triggers, or you can generate the triggers directly in the database via ODBC driver data sources.

### Automatic creation of referential integrity triggers

PowerDesigner provides the Rebuild Triggers function to automatically create referential integrity triggers for a selected table linked by a reference.

The referential integrity triggers are created from trigger templates defined either in the current DBMS file, or user-defined trigger templates listed in the list of Triggers.

Triggers that apply to the trigger referential integrity constraints defined for a reference, are created for selected tables attached to the reference.

#### **User-defined triggers and trigger templates**

All user-defined triggers and triggers based on user-defined trigger templates are created automatically independently of Trigger referential integrity constraints. However, for this tutorial you will only create triggers based on trigger referential integrity constraints defined for a reference.

In this lesson, you will use the Rebuild Trigger function to automatically create referential integrity triggers that apply to the table STORE.

About trigger templates


A trigger template is a pattern for creating triggers. PowerDesigner ships pre-defined templates for each supported DBMS. Depending on the current DBMS, there are pre-defined templates for insert, update, and delete trigger types. A trigger template can be stored in the model, or in the current DBMS file.

You can create your own trigger templates or customize an existing template, for example, by changing the type of error message displayed.

About template items

A template item is a reusable block of SQL script that can implement referential integrity, or do any other work on database tables. A template item is inserted into a trigger template script, or a trigger script. The template item calls a corresponding SQL macro which can implement an insert, update, delete, or error message constraint on one or more tables in the database.

PowerDesigner ships pre-defined template items that implement referential integrity constraints. These template items are inserted in the pre-defined trigger template scripts. When a trigger is created from the trigger template, the template item is generated in the trigger only if it implements the appropriate trigger referential integrity defined for a reference attached to a selected table.

 For more information on creating and using trigger templates and template items, see the *PowerDesigner Physical Data Model User's Guide*.

In this lesson you will:

- ◆ Define Trigger referential integrity for a reference
- ◆ Use Rebuild Triggers to automatically create triggers for the table STORE
- ◆ Preview the script for a trigger
- ◆ Generate a script for the triggers

**How long will it take?**

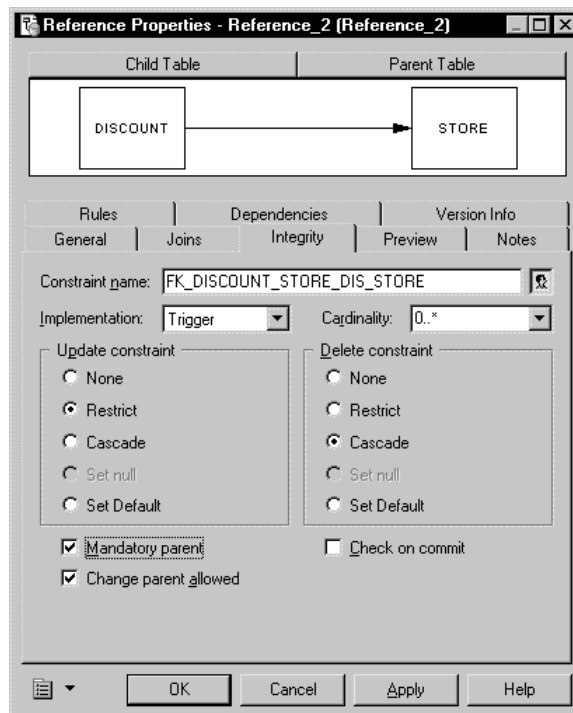
About 20 minutes.

## Defining trigger referential integrity

In a previous lesson you defined a cascade update for referential integrity when a store is deleted in the *STORE* table. You will now define trigger implementation for the same reference.

- 1 Double-click the reference that links the *STORE* and *DISCOUNT* tables.  
The reference property sheet appears.
- 2 Click the *Integrity* tab to display the Integrity page appears.
- 3 Click the down arrow at the end of the Implementation dropdown listbox.  
A dropdown list appears.
- 4 Select *Trigger* from the dropdown list.

You define trigger as the implementation for referential integrity for the reference.



- 5 Click **OK**.

You return to the PDM diagram.

## Automatic creation of triggers

You will use the Rebuild Triggers function to automatically create triggers for the table STORE. The triggers will be created using the PowerDesigner pre-defined trigger templates in the current DBMS file.

### Using PowerDesigner pre-defined trigger templates

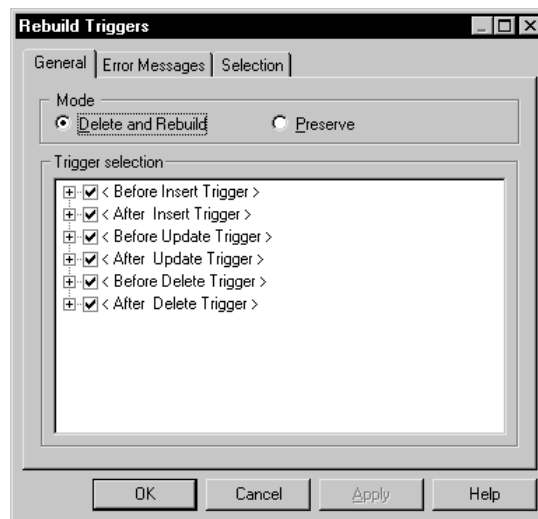
The trigger templates and template items used in the tutorial are the triggers shipped with PowerDesigner. You can copy and modify them to create new user-defined templates and template items.

For more information on creating and using trigger templates and templates items, see the chapter Using Triggers in the *PowerDesigner Physical Data Model User's Guide*.

Rebuild Triggers automatically creates a Delete trigger to cascade the deletion of a store from the STORE table to the DISCOUNT table.

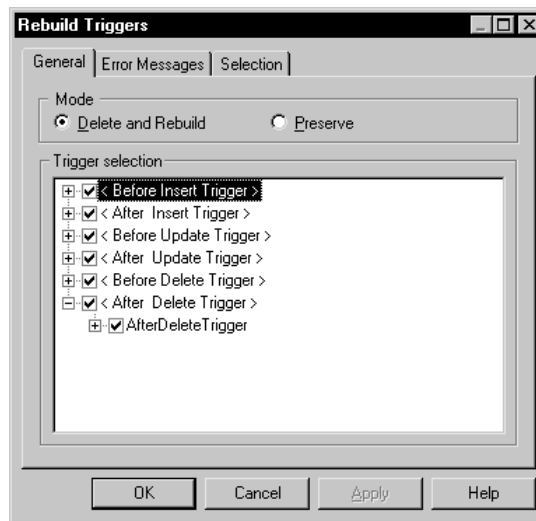
- 1 Select **Tools**→**Rebuild Objects**→**Rebuild Triggers**.

The Rebuild Triggers box appears. It displays the trigger types supported by the current DBMS organized in a tree view. When you expand a trigger type node, the next level that appears are the trigger templates that are available for that trigger type. The selected trigger templates at this level are the templates that are available to create the referential integrity triggers for that trigger type.



- 2 Expand the **<After Delete Trigger>** node at the bottom of the tree view.

The `AfterDeleteTrigger` node appears. This is the only PowerDesigner pre-defined trigger template available for this trigger type.

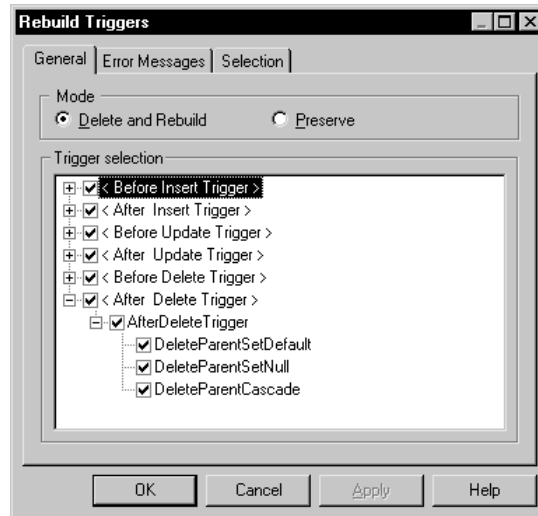


- 3 Expand the *AfterDeleteTrigger* node.

The three new nodes that appear are the PowerDesigner pre-defined template items as defined in the trigger template.

Each template item implements a type of referential integrity in `AfterDeleteTrigger`.

While each template item is defined in the trigger template, only the template items that are selected and able to implement the type of referential integrity defined for the reference are included in the trigger script being created.



- 4 Click the **Selection** tab.

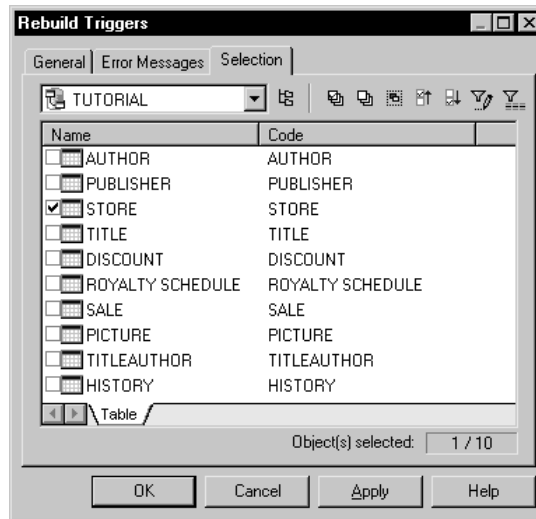
The Selection page appears. It shows checkboxes for each table in the PDM.

- 5 Click the **Deselect All** tool from the tool bar at the top of the page.



All the Checkboxes are cleared.

- 6 Select the **STORE** table checkbox.



- 7 Click **OK**.

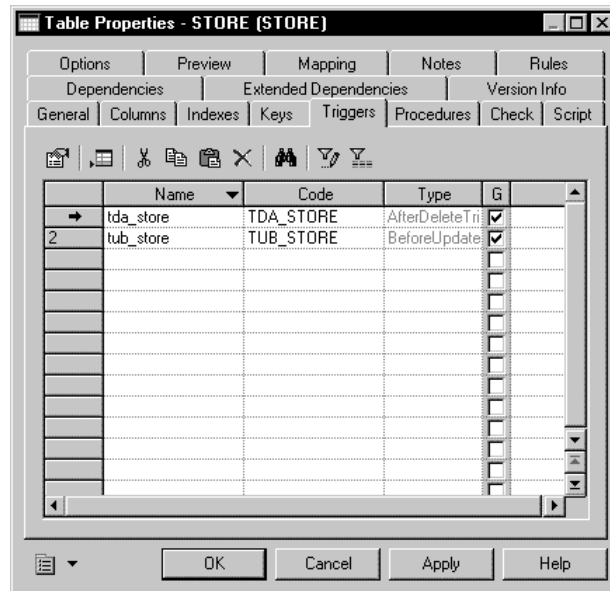
Rebuild triggers automatically creates referential integrity triggers for the **STORE** table based on the trigger referential integrity defined for the reference linking **STORE** and **DISCOUNT**.

- 8 Double-click the **STORE** table.

The table property sheet appears.

- 9 Click the **Triggers** tab.

The Triggers page appears. It shows that two triggers (AfterDeleteTrigger and BeforeUpdate Trigger) have been created. These two triggers correspond to the trigger referential integrity defined for the reference linking the STORE and DISCOUNT tables.



- 10 Click **Cancel**.

You return to the PDM diagram.



## Previewing a trigger

You will preview the trigger script for the AfterDeleteTrigger trigger created for the table STORE.

- 1 Double-click the **STORE** table.

The table property sheet appears.

- 2 Click the **Triggers** tab.

The Triggers page appears.

- 3 Click the **TDA\_STORE** line. This is the AfterDeleteTrigger trigger created in the last lesson.

An arrow appears at the start of the line.

- 4 Click the **Properties** tool.



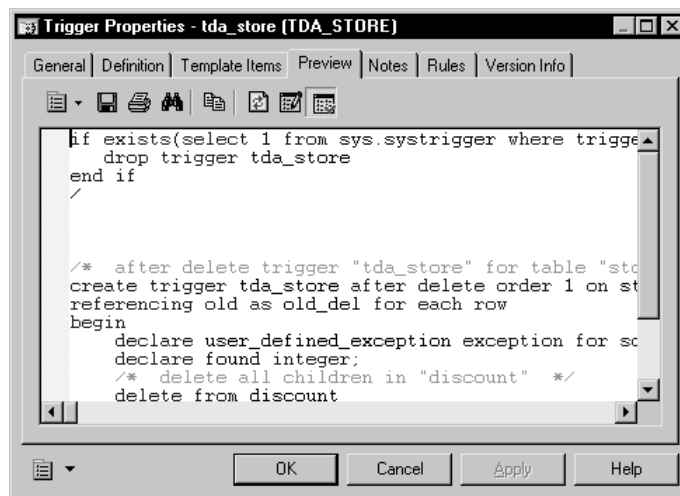
or

Double-click the arrow at the start of the line.

The trigger property sheet appears.

- 5 Click the **Preview** tab.

The Preview page appears. It shows the trigger script with trigger template variables instantiated with the values for the STORE table.



- 6 Click **Cancel** in each of the dialog boxes.

You return to the PDM diagram.

**What you learned**

In this section, you learned how to:

- ◆ Define trigger implementation for referential integrity for a reference
- ◆ Use Rebuild Triggers to automatically create triggers for a selected table
- ◆ Preview the trigger script


## Generate a script for triggers

You will define parameters to generate the script for the triggers and procedures you defined.

- 1 Select **Database→Generate Triggers and Procedures** from the menu bar.  
A dialog box containing the generation parameters appears.
- 2 Type **TUT\_TRIG.SQL** in the File Name box.
- 3 In the **Directory** box, type or browse to the **EXAMPLES** directory in the PowerDesigner path.
- 4 Select the **Script Generation** radio button.
- 5 Make sure all **generation parameters** are selected except for the following:

Groupbox	Deselected items
Triggers	Drop Trigger
Procedures	Permission

These parameters generate all triggers.

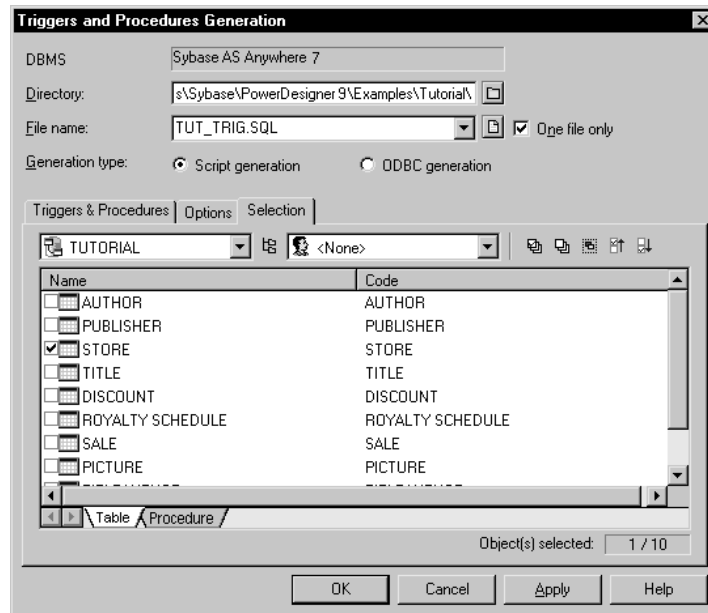
 For a complete description of generation parameters, see the *Physical Data Model User's Guide*.

- 6 Click the **Options** tab.  
The Options page appears. It lists script generation options.
- 7 Keep the default **options**.
- 8 Click the **Selection** tab.  
The Selection page appears.
- 9 Click the **Deselect All** tool from the tool bar at the top of the page.



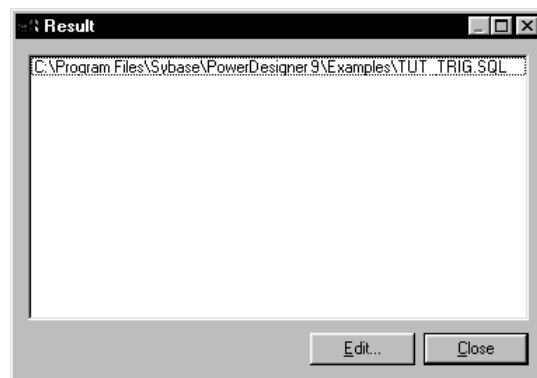
All the Checkboxes are cleared.

- 10 Select the **STORE** table checkbox as shown below.



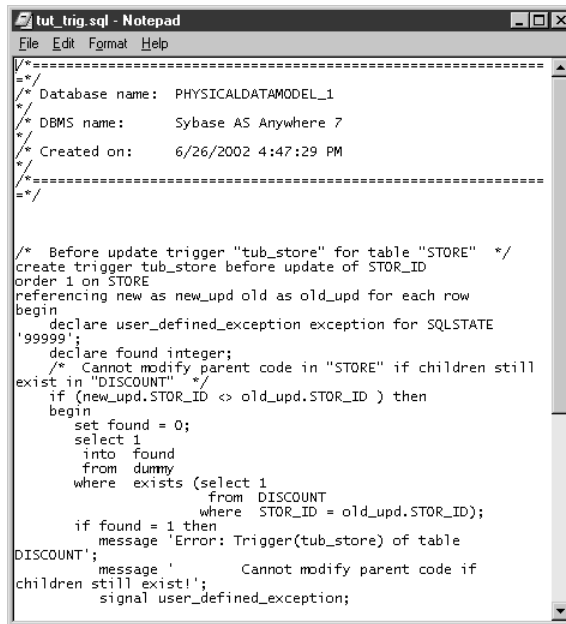
- 11 Click **OK**.

A result box appears showing the file path of the generated trigger script file.



- 12 Click the **file path** to select it, and then click the **Edit** button.

The generated trigger script appears in a text editor.



```

/*=====
/* Database name:  PHYSICALDATAMODEL_1
/* DBMS name:      Sybase AS Anywhere 7
/* Created on:     6/26/2002 4:47:29 PM
/*=====
*/

/* Before update trigger "tub_store" for table "STORE" */
create trigger tub_store before update of STOR_ID
order 1 on STORE
referencing new as new_upd old as old_upd for each row
begin
  declare user_defined_exception exception for SQLSTATE
  '99999';
  declare found integer;
  /* Cannot modify parent code in "STORE" if children still
  exist in "DISCOUNT" */
  if (new_upd.STOR_ID <> old_upd.STOR_ID ) then
    begin
      set found = 0;
      select 1
      into found
      from dummy
      where exists (select 1
                    from DISCOUNT
                    where STOR_ID = old_upd.STOR_ID);
      if found = 1 then
        message 'Error: Trigger(tub_store) of table
DISCOUNT';
        message '      Cannot modify parent code if
children still exist!';
        signal user_defined_exception;
    end
  end if;
end

```

- 13 Select **File**→**Exit** to close the text editor. If you are prompted to save the file, click the No button.
  - 14 Click **Close** to close the Result window.
  - 15 Close the Result List window. This is the Check Model result window.
- You return to the PDM diagram.

#### What you learned

In this section, you learned how to:

- ◆ Select options that will generate triggers
- ◆ Generate a script for triggers



## Using Abstract Data Types

**You must have an Object-Oriented Model for this lesson**

This lesson can only be done if you have access to the PowerDesigner Object-Oriented Model (OOM). Certain versions of the PowerDesigner Physical Data Model (PDM) may not be bundled with the OOM. If you do not have access to the PowerDesigner Object-Oriented Model, you can skip this lesson and go directly to the next lesson

An **abstract data type (ADT)** is a user-defined data type which encapsulates a range of data values and functions. The functions can be both defined on, and operate on the set of values.

Depending on the current DBMS, PowerDesigner supports the following types of abstract data types:

- ◆ Arrays
- ◆ Lists
- ◆ Java classes
- ◆ Objects
- ◆ Structured

You can specify and reverse engineer the definition for abstract data types into a PDM. Once an abstract data type has been specified in the PDM, it can be used by columns and domains in the same way as standard data types. In this tutorial you will work with Java classes.

### Java classes

In a PDM, abstract data types of the type **Java** class can be linked to Java classes in the PowerDesigner Object-Oriented Model (OOM). This allows you to access the property sheet of linked Java classes from the PDM.

To access Java class properties from the PDM, the Java class must exist in the OOM. A Java class can be created directly, or reverse engineered into an OOM.

Why access a Java class?

You can reverse engineer a database that contains Java classes that are used as data types for columns and domains into a PDM. These Java classes often contain much more complex information than the length and precision properties of standard data types. You may need to have more information about the operations that the Java classes contains to understand the actions that it performs in the database.

You can access this information by reverse engineering the Java classes that are used in the PDM into an OOM. The Java classes in the PDM can be linked to the corresponding Java classes in the OOM. You can then access the property sheet for a Java class. You can study its operations and get a much more complete understanding of how a Java class operates as a column data type for a table in the database.

In this lesson you will:

- ◆ Specify an abstract data type of type Java class in the PDM
- ◆ Open an Object-Oriented Model (OOM)
- ◆ Create a link between the Java class specified in the PDM and the corresponding Java class in the OOM

**How long will it take?**

About 5 minutes.



## Specifying an abstract data type as a Java class

When you reverse engineer a database creation script into a PDM, the name of a Java class is included in the column or domain definition to which it is attached. However the definition of the Java class is not reverse engineered into the PDM.

To access a Java class in the OOM, you need to specify an abstract data type as a Java class in the List of Abstract Data Types in the PDM before you can link it to the corresponding Java class in the OOM.

- 1 Select **Model→Abstract Data Types**.

The list of abstract data types appears.

- 2 Click the **Add a Row** tool.



An arrow appears at the start of the first empty line and a default name and code are entered.

- 3 Type **STORE** in the Name column.

This is the name of the abstract data type.

- 4 Click **Apply**.

The creation of the new abstract data type definition is committed.

- 5 Click the new abstract data type line.

An arrow appears at the beginning of the line.

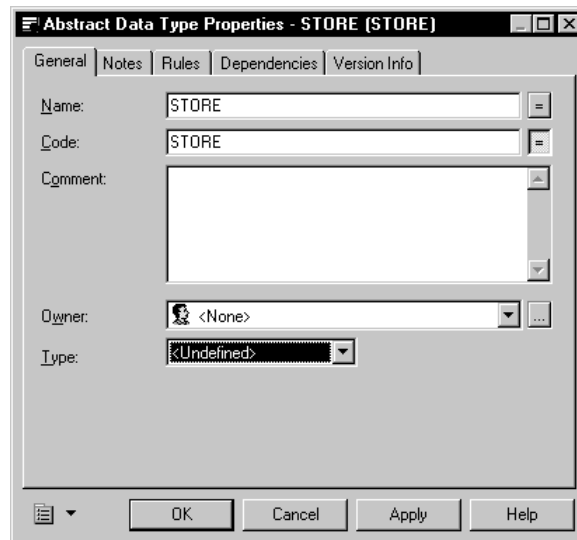
- 6 Click the **Properties** tool.



or

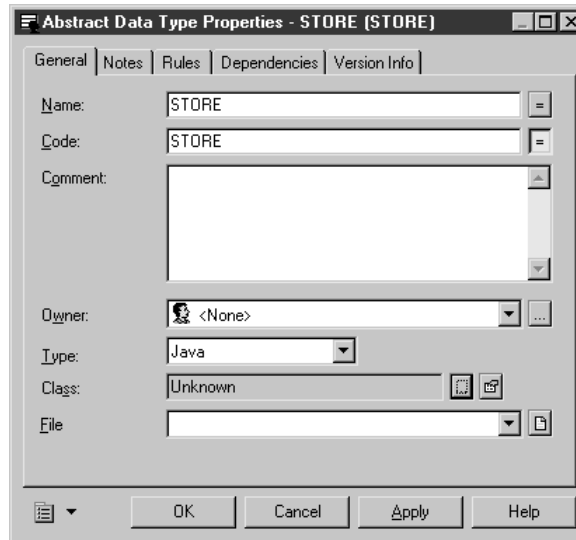
Double click the arrow at the beginning of the line.

The property sheet for the new abstract data type appears.



- 7 Click the down arrowhead at the end of the Type dropdown listbox.  
A dropdown list appears.
- 8 Select **JAVA** from the dropdown list.

The Class and Filename boxes appear at the bottom of the property sheet. The Class box indicates the Java class in the OOM that is linked to the Java class in the PDM. The Filename box indicates the path of a Java class installed in the database. Both these boxes are empty when you first define the Java class.



- 9 Click **OK**.  
You return to the List of Abstract data Types.
- 10 Click **OK**.  
You return to the PDM diagram.

## Create a PowerDesigner Object-Oriented Model

You need to create an OOM. The new OOM should contain a class called **STORE** which contains operations that can return information about an instance of a store in a table called **STORE**.

In this lesson, you will create a basic class without going any further in its definition.

- 1 Select **File**→**New**.


The New dialog box appears.

- 2 Select Object-Oriented Model and click OK.

The New Object-Oriented Model dialog box appears. It contains a dropdown listbox from which you select the object language for the new model.

- 3 Select **Java** in the Object Language dropdown listbox.
- 4 Select the **Share** radio button.
- 5 Select Class Diagram in the First Diagram dropdown listbox.
- 6 Click **OK**.
- 7 Create a class in the model and name it **STORE**.

You do not need to further define this class.

 For more information on the OOM and classes, see the *OOM User's Guide*.

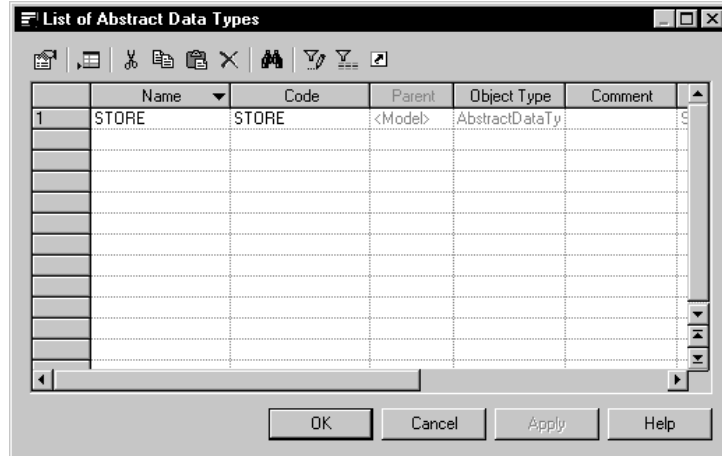
- 8 Select **Window**→\...\b>TUTORIAL.PDM to return to the PDM tutorial model.

## Accessing Java class properties

You will define a link between the STORE Java class in the PDM with the STORE Java class in the OOM. You will then access the property sheet for the Java class in the OOM to view its properties.

- 1 Select *Model*→*Abstract Data Types*.

The list of abstract data types appears. It shows the `STORE` Java class that you specified in a previous lesson.



- 2 Click the **STORE** line.

An arrow appears at the start of the line.

- 3 Click the *Properties* tool.



*or*

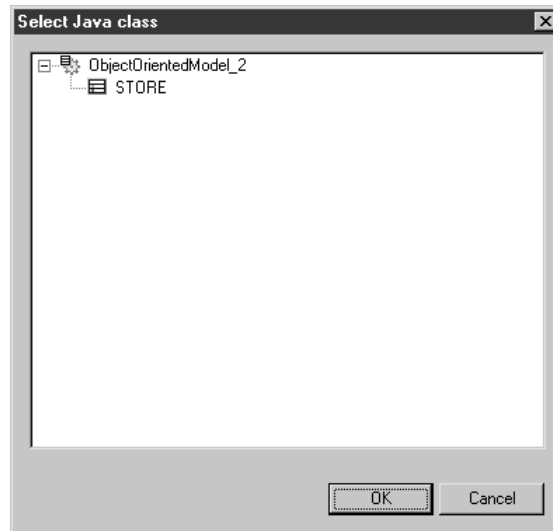
Double-click the arrow at the start of the line.

The property sheet for the **STORE** abstract data type appears.

- 4 Select the ***Ellipsis*** button at the end of the Class box.



The Select Java Class box appears. It displays a tree view of the Java classes that are available in the OOM. The Java class that you select will be linked to the Java class specified in the PDM.



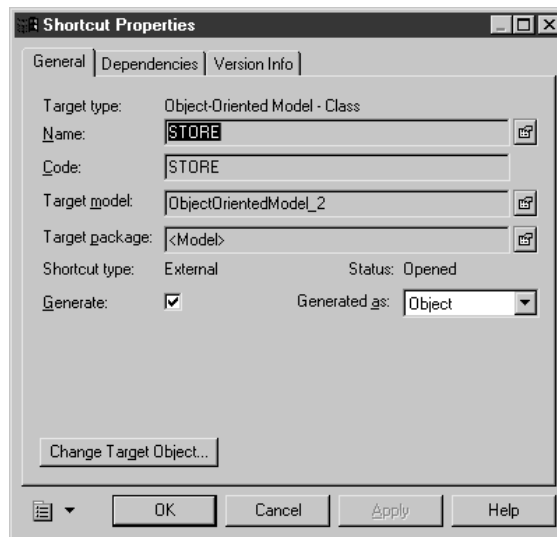
- 5 Click the **STORE** node.
- 6 Click **OK**.

You return to the abstract data type property sheet. **STORE** now appears in the Class box. This indicates that the Java class **STORE** in the PDM is linked to the Java class **STORE** in the OOM.

- 7 Click the **Properties** tool at the end of the Class box.



A shortcut property sheet appears. It indicates that the target object (referenced object) is the STORE class, and that the target model (referenced model) is the OOM.



- 8 Click **OK** in each of the dialog boxes.

You return to the PDM diagram.

#### What you learned

In this section, you learned how to:

- ◆ Specify an abstract data type as a Java class
- ◆ Open a PowerDesigner Object-Oriented Model.
- ◆ Link a Java class specified in the PDM with a Java class in an OOM and view its property sheet.





## Generating a Database Script

You can generate a database directly from a PDM, or generate a database script which you can run in your DBMS environment. In this chapter you will create a script for the current database.

The generation parameters which are available depend on the target DBMS you select. By default, the current DBMS is the one you select when you open the PDM, but you can select a different DBMS before generating the script.

In this chapter you will:

- ◆ Generate a database creation script
- ◆ Save and close the PDM

**How long will it take?**

About 5 minutes.

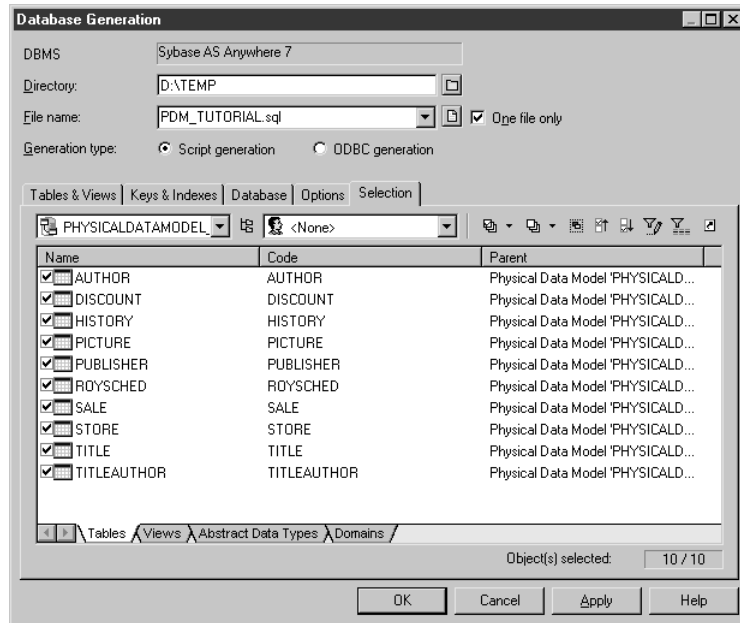
## Generate a database creation script

- 1 Select **Database**→**Generate Database**.  
The Database Generation dialog box appears. It displays generation parameters. Certain parameters are already selected.
- 2 Type **PDM\_TUTORIAL.SQL** in the File Name box.
- 3 In the **Directory** box, type a path.
- 4 Select the **Script Generation** radio button.
- 5 Select the **One File Only** check box.  
All check boxes in the lower part of the dialog box should be selected.
- 6 Click the **Selection** tab.  
The Selection page appears.
- 7 Click the **Tables** tab at the bottom of the page.  
The Table page lists all the tables available for selection in the model.

- 8 Click the **Select All** tool.

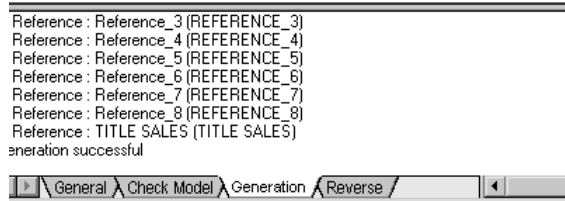


This selects all the table check boxes.



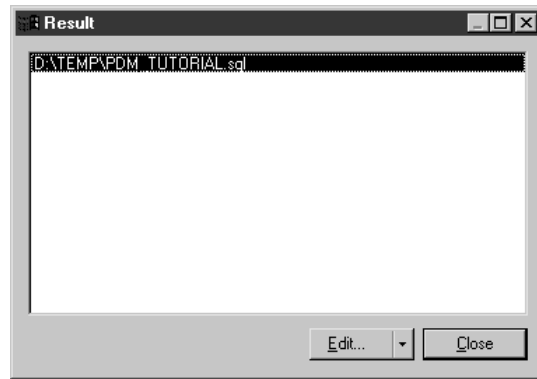
- 9 Repeat steps 7 and 8 for the **View** and **Domain** tabs.  
10 Click **OK**.

A result list appears showing the results of the Check Model. By default PowerDesigner verifies a model before generation. The Generation process is shown in the Output window in the lower part of the main window.



For a complete description of database generation parameters, see the *Physical Data Model User's Guide*.

At the end of the generation process a result box appears . It displays the file path of the generated script file.



- 11 Select the displayed file path and click the **Edit** button.

A window displays the generated script.

- 12 Select **File→Exit** to close the script window.
- 13 Click **Close** to close the Result box.
- 14 Close the result window.

You return to the PDM diagram.

# Generating a Test Data Script

## About test data

**Test data** is sample data that you can define and generate for one or more tables in a PDM. When you generate test data, the generation process automatically does the following:

- ◆ Creates SQL statements to insert data into tables
- ◆ Creates rows of data in the tables
- ◆ Inserts rows of data in the database through the ODBC.

You can use test data to verify database performance when it contains large amounts of data or when many users or applications are accessing it simultaneously.

You can generate test data directly from the PDM, or with a data script that you can execute in ODBC.

You can also obtain test data by importing values from a CSV file.

## About test data profiles

You generate test data for a table based on test data profiles. A **test data profile** defines the set of values to be generated according to a data type (character, number, or time and date).

You can assign a test data profile to one or more selected columns. Test data is generated for the columns using the data source defined for each test data profile.

In this chapter you will:

- ◆ Import test data profiles to the tutorial model
- ◆ Create new test data profiles in the tutorial model
- ◆ Define a test data profile as a source for automatic test data generation
- ◆ Define a file as a source for test data values
- ◆ Assign test data profiles to selected columns in tables
- ◆ Generate a test data creation script for tables
- ◆ Exit the PDM and PowerDesigner

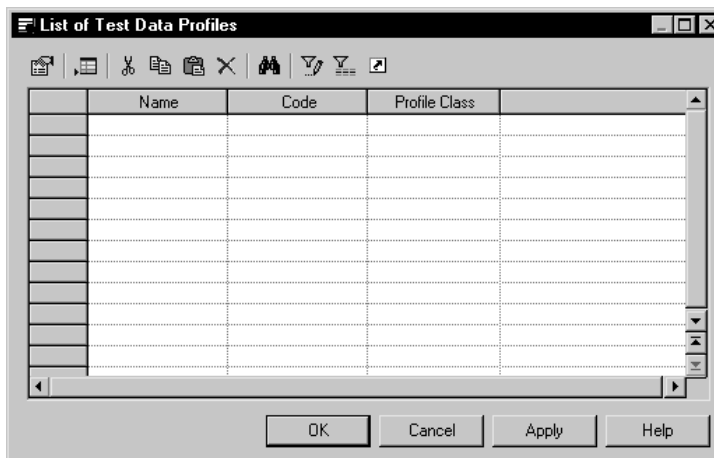
**How long will it take?**  
About 20 minutes.

## Import test data profiles

Test data profiles can be imported from or exported to another PDM as XPF files. For this tutorial, a number of test data profiles have been pre-defined in the file TUTORIAL.XPF in the TUTORIAL directory. You will import these data profiles to your tutorial model.

- 1 Select *Model*→*Test Data Profiles*.

The list of test data profiles appears. The list is empty. You will import test data profiles contained in the TUTORIAL.XPF file to this list.



- 2 Click *Cancel*.

You return to the PDM diagram.

- 3 Select **Tools**→**Test Data Profile**→**Import**.

A standard file selection box appears.

- 4 Browse to the **EXAMPLES/TUTORIAL** directory.

- 5 Select the file *Tutorial.XPF* and click *Open*.

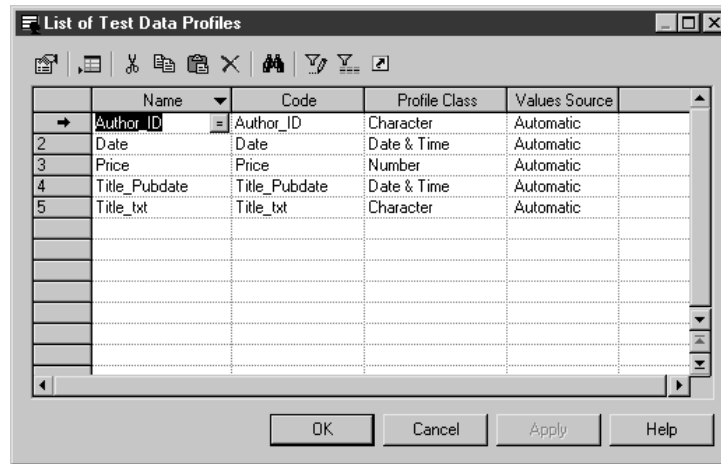
The Output window informs you that the profiles defined in the file TUTORIAL.XPF have been successfully imported into your model.

- 6 Click **OK**.

You return to the PDM diagram.

- 7 Select **Model** → **Test Data Profiles**.

The list of test data profiles displays the imported data profiles.



## Display the column you need

If you don't see the column you need, display it with the Customize Columns and Filter tool.

- 8 Click **Cancel**.

You return to the PDM diagram.



## Create a new test data profile

In this lesson you will create an additional test data profile which is added to the list you imported from the file *tutorial.XPF*.

The new test data profile is *Name\_ID*. This profile can represent a column that has a character data type.

- 1 Select **Model→Test Data Profiles** from the menu bar.

The list of test data profiles appears and shows existing profiles.

- 2 Click the **Add a Row** tool.



An arrow appears at the beginning of the first empty line and a default name and code are entered.

- 3 Type **Name\_ID** in the first line of the Name column.

This is the name of the test data profile. The code equals the test data profile name.

- 4 Click in the Profile Class column on the same line.

- 5 Click the Down Arrow button.

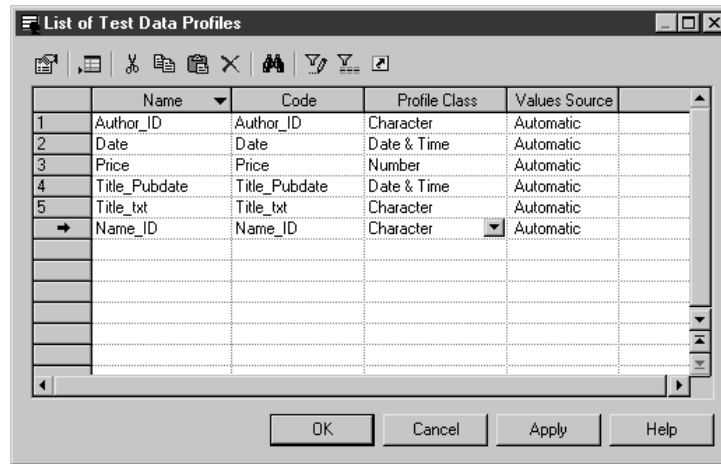
The profile class dropdown listbox appears.

### Display the column you need

If you don't see the column you need, display it with the Customize Columns and Filter tool.

- 6 Select **Character** from the dropdown list in the Profile Class column.

This defines the character class for the data profile Name\_ID.



- 7 Click **OK**.

You return to the PDM diagram.

## Define a test data profile as a source for automatic test data generation

To generate test data you need to define a data source for each test data profile. PowerDesigner can generate test data values automatically, using generation parameters that are defined for each test data profile.

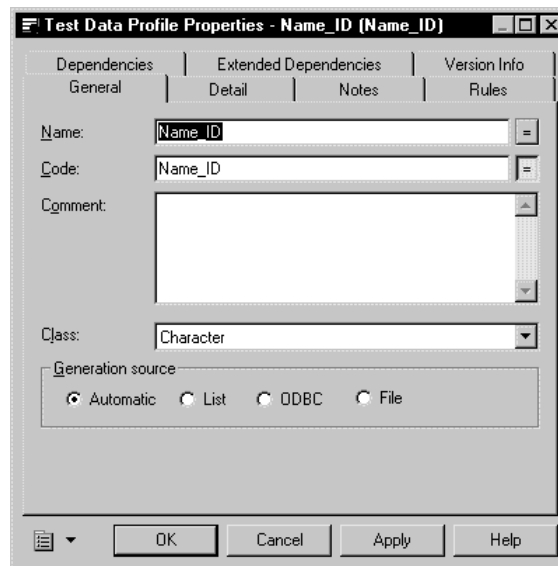
You will define automatic test data generation parameters for the data profiles `Name_ID` and `Price`.

- 1 Select **Model** → **Test Data Profiles** from the menu bar.

The list of test data profiles appears and shows existing profiles.

- 2 Double-click the **Name\_ID** line in the list.

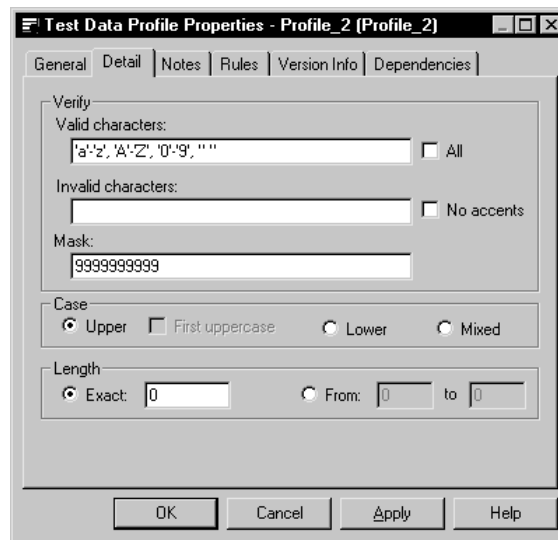
The test data profile property sheet opens to the General page. The Automatic radio button is selected when you select a data profile that does not currently have a data generation source defined.



- 3 Click the **Detail** tab.

The Detail page appears. You will define a mask for the test data profile `Name_ID`. A mask is a placeholder for character strings. PowerDesigner will generate data that respects the mask characters that you define for `Name_ID`.

- 4 Click the **Mask** textbox.  
Type **9999999999** in the textbox (ISBN numbers are composed of 10 characters).



- 5 Click the **Exact** textbox.  
Type **10** in the Exact textbox.  
PowerDesigner will generate a random number for each mask character 9.
- 6 Click **OK**.  
You return to the list of test data profiles.
- 7 Double-click the **Price** line in the list.  
The test data profile property sheet opens to the General page. The Automatic radio button is selected when you select a data profile that does not currently have a data generation source defined.
- 8 Click the **Detail** tab.  
The Detail page appears.
- 9 Select the **Random** radio button.
- 10 Type **10** in the From textbox and **1000** in the To textbox.  
Select the **Generate decimal numbers** checkbox.  
Type the number **2** in the Decimal digits number textbox.

You define a range of decimal numbers between 10.00 and 1000.00 for the random test numbers that will be generated for the data profile Price.

The screenshot shows a dialog box titled "Test Data Profile Properties - Price [Price]". It has several tabs: General, Detail, Notes, Rules, Version Info, and Dependencies. The "General" tab is active. Inside the dialog, there are three main sections: "Type", "Range", and "Decimal Numbers". In the "Type" section, the "Random" radio button is selected. In the "Range" section, the "From" field contains "10", the "to" field contains "1000", and the "Step" field is empty. In the "Decimal Numbers" section, the checkbox "Generate decimal numbers" is checked, and the "Decimal digits number" field contains "2". At the bottom of the dialog, there are four buttons: "OK", "Cancel", "Apply", and "Help".

- 11 Click **OK** in each of the dialog boxes.

## Define a file as a source for test data values

You can specify a CSV format file which contains values that can be used to create test data. You will define a CSV file as a test data source for the data profile `Title Txt`. The directory `TESTDATA` in the PowerDesigner path contains several CSV files that can be used as test data generation source files.

- 1 Select **Model** → **Test Data Profiles** from the menu bar.

The list of test data profiles appears and shows existing profiles.

- 2 Double-click the **Title Txt** line in the list.

The test data profile property sheet opens to the General page. The Automatic radio button is selected when you select a data profile that does not currently have a data generation source defined.

- 3 Select the **File** radio button from the Generation Source group box.

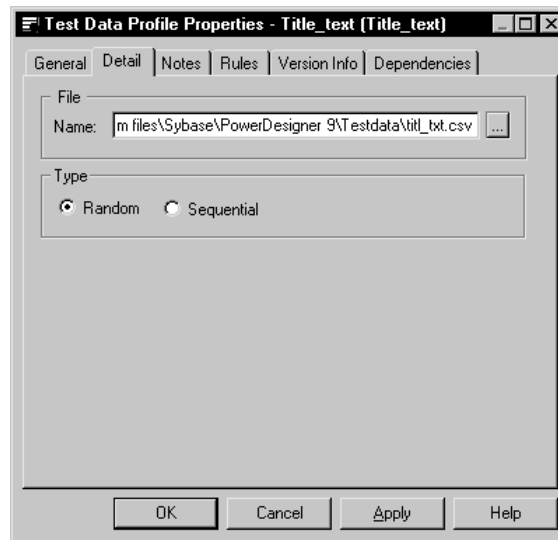
- 4 Click the **Detail** tab.

The Detail page appears.

- 5 Select or browse to the directory **TESTDATA** in the PowerDesigner path.

- 6 Select the file **titl\_txt.csv**.

This file contains a number of book titles.



- 7 Click **OK** in each of the dialog boxes.  
You return to the PDM diagram.

## Assign test data profiles to selected columns

You can assign test data profiles directly to selected table columns. You can do this for columns that are not attached to a domain, or if you want to generate test data that is specific to a column.

You cannot assign a test data profile to a foreign key column. It automatically takes the data profile of the primary key column in the parent table.

You will assign test data profiles to selected columns for the **TITLE** table in the PDM model. The same procedure can be used to assign test data profiles to any selected column in any table in the PDM model.

- 1 Select **Model** → **Tables** from the menu bar.

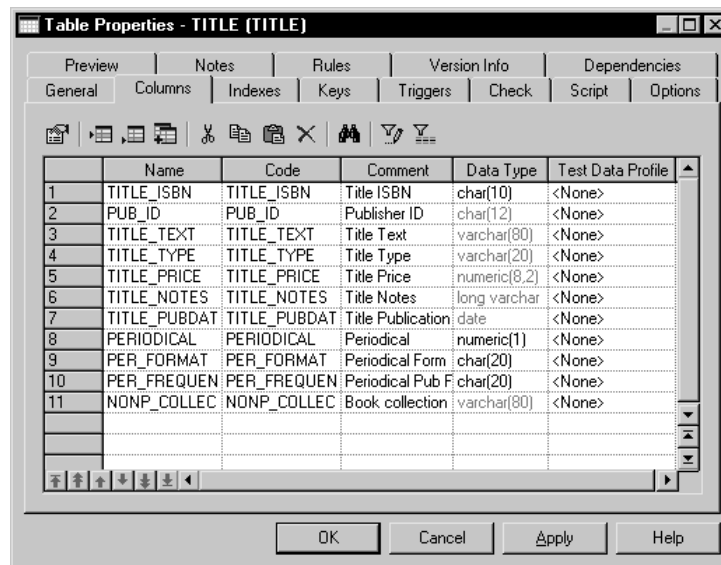
The list of tables appears.

- 2 Double-click the **TITLE** table.

The table property sheet opens to the General page.

- 3 Click the **Columns** tab.

The Columns page appears.



### Display the column you need

If you don't see the column you need, display it with the Customize Columns and Filter tool.



TITLE\_ISBN has the data type CHAR(10). Based on this data type, and the sort of data that you want to generate for the column, you will assign the Name\_ID data profile to this column.

- 4 Double-click the **TITLE\_ISBN** line in the list of columns.

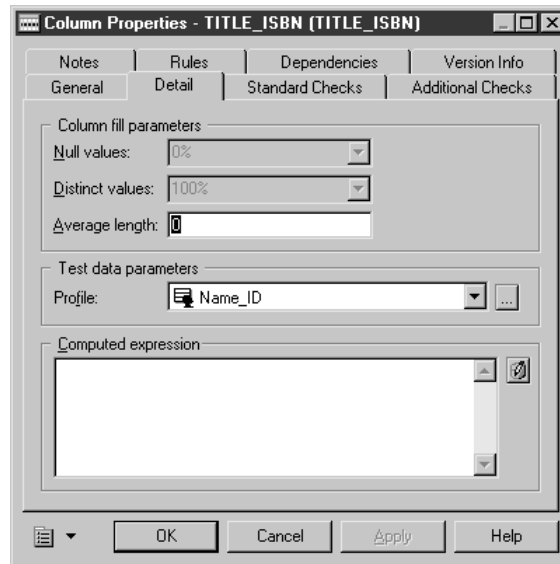
The column property sheet opens to the General page.

- 5 Click the **Detail** tab.

The Detail page appears.

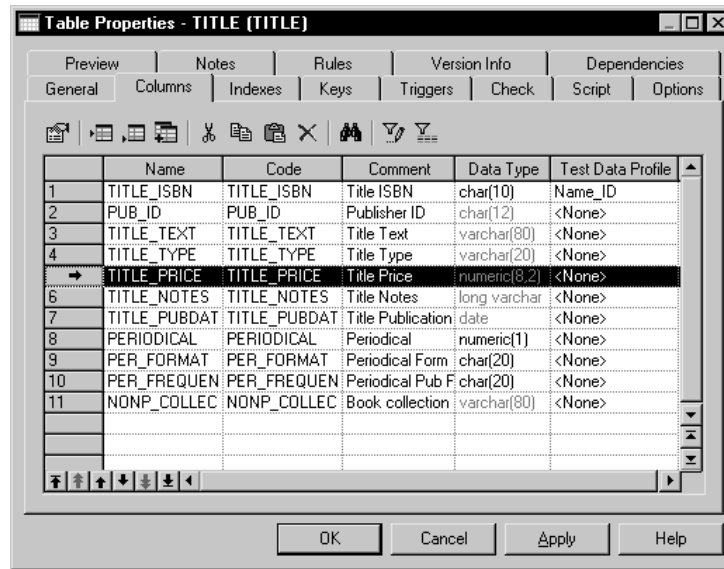
- 6 Select **Name\_ID** in the dropdown list of the Profile textbox.

The profile is attached to the column.



- 7 Click **OK**.

You return to the Columns page of the table property sheet.



TITLE\_PRICE has the data type NUMERIC(8,2). Based on this data type, and the sort of data that you want to generate for the column, you will assign the name data profile to this column.

- 8 Double-click the **TITLE\_PRICE** line in the list of columns.

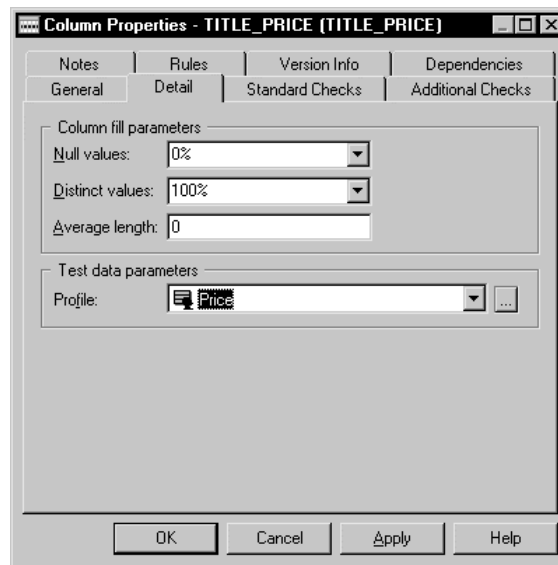
The column property sheet opens to the General page.

- 9 Click the **Detail** tab.

The Detail page appears.

- 10 Select **Price** in the dropdown list of the Profile textbox.

The profile is attached to the column.



- 11 Click **OK**.

You return to the columns property sheet.

- 12 Repeat steps 8 to 11 using **TITLE\_PUBDAT** in the list of columns but select **Date** in the dropdown list of the Profile textbox.
- 13 Click **OK** in each of the dialog boxes.

You return to the PDM diagram.

## Generate a test data creation script

You will generate a test data script for the tables TITLE, AUTHOR, and TITLEAUTHOR. This script is named author.sql. You must execute the test data script in a database that contains the same tables that you use to generate the script.

- 1 Select **Database**→**Generate Test Data**.

The Test Data Generation dialog box opens to the Parameters page. It displays the test data generation parameters. Certain parameters are already selected.

- 2 Select a destination directory.
- 3 Type **author.sql** in the File name box.
- 4 Select the **Script generation** radio button.
- 5 Select the **Delete Old Data** checkbox.

This option deletes all existing data in the tables so that they can be filled with new test data.

- 6 Type **15** in the **Default Number of Rows** textbox.

The default number of rows is the number of rows of test data that will be generated for tables that do not have a Number value defined in their property sheets.

- 7 Select **Price** from the **Default Number Profile** dropdown list.  
Select **Name\_ID** from the **Default Character Profile** dropdown list.  
Select **Date** from the **Default Date Profile** dropdown list.

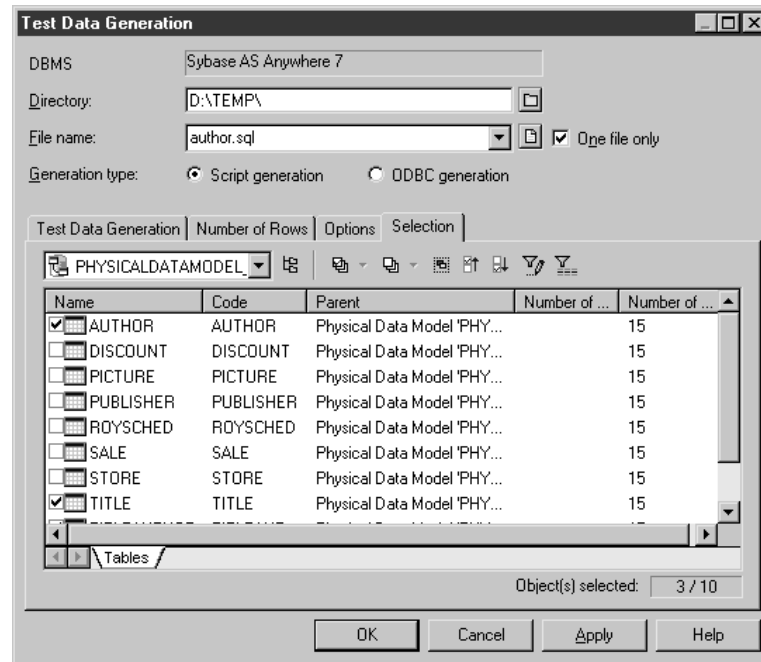
The screenshot shows the 'Test Data Generation' dialog box. At the top, the title bar reads 'Test Data Generation'. Below it, the 'DBMS' field is set to 'Sybase AS Anywhere 7'. The 'Directory' field is 'D:\TEMP\' and the 'File name' is 'author.sql'. The 'Generation type' is set to 'Script generation'. The 'Selection' tab is selected, showing a 'Delete old data' checkbox, a 'Default number of rows' field set to '15', and three dropdown menus for 'Default number profile' (set to 'Price'), 'Default character profile' (set to 'Name\_ID'), and 'Default date profile' (set to 'Date'). At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

- 8 Click the Selection tab.  
The Selection page appears.
- 9 Click the **Deselect All** tool from the tool bar at the top of the page.



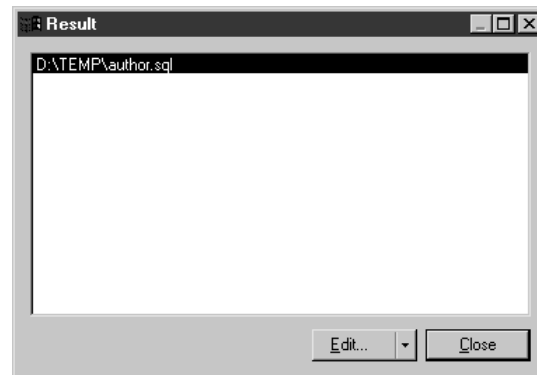
All the checkboxes are cleared.

- 10 Select the **AUTHOR**, **TITLE** and **TITLEAUTHOR** tables from the list.



- 11 Click **OK** to start the generation.

A Progress window briefly shows the test data generation progress. A Result dialog box asks you if you want to Edit or Close the newly generated file.



- 12 Click **Close** to close the Test Data Generation dialog box.  
You return to the PDM diagram.

## Designing a Data Warehouse Database Schema

What is a data warehouse?

In this lesson, you will design a data warehouse database schema.

A **Data warehouse database** is used to store historical business data from heterogeneous systems. Data warehouse databases are populated with data proceeding from different operational sources. In this lesson, you will consider that operational data proceed from the model PUBLISHING INDUSTRY (PDMAFTER.PDM).

The purpose of maintaining such type of database is to gather in a central area all the information that may be needed in an OLAP database where queries for business analysis and decision making are performed.

You will use tables from an operational model to create a new data warehouse database schema. This schema will be used in the next lesson to design the structure of the OLAP database.

In this lesson you will:

- ◆ Create a new model and add tables and references to the model
- ◆ Create a data source in the new model
- ◆ Define a relational to relational mapping between the operational and the data warehouse models
- ◆ Generate an extraction script

**How long will it take?**

About 20 minutes.

## Create new model and copy tables

PUBLISHING INDUSTRY (PDMAFTER.PDM) is going to be used as an operational model in the following lessons. You will create a new model and copy tables from PUBLISHING INDUSTRY to design a data warehouse database corresponding to the operational model.

To keep these lessons short, you will focus on one aspect of the operational model, that is book sales per author.

- 1 Click the **New** tool in the toolbar.

The New dialog box appears.

- 2 Select Physical Data Model and click OK.

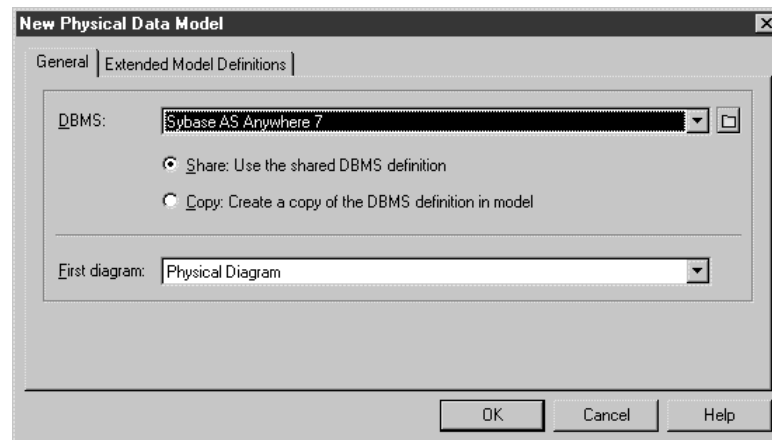
The Choose DBMS dialog box appears.

- 3 Select **Sybase AS Anywhere 7** from the DBMS dropdown listbox.

- 4 Select the **Share** radio button.

You will use the DBMS definition that is contained in the PowerDesigner DBMS directory.

- 5 Select **Physical Diagram** in the First Diagram dropdown listbox.





- 6 Click **OK**.

The new model appears in the Browser and displays a physical diagram in the diagram window.

- 7 Select **Model→Model Properties** to display the model property sheet.

- 8 Type **Business Intelligence** in the Name box.

This is the name of the model. BUSINESS\_INTELLIGENCE appears automatically in the Code box. This is the code of the model.

- 9 Click **OK**.

- 10 Select **File→Open** and select PDMAFTER.PDM in the PowerDesigner tutorial directory.

The model appears in the Browser and the diagram displays model objects.

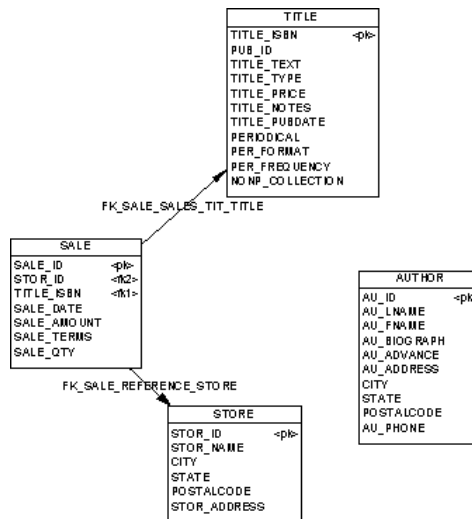
- 11 Press the SHIFT key and click on tables **AUTHOR**, **TITLE**, **SALE**, **STORE** with the references between **SALE** and **TITLE** and between **SALE** and **STORE**.

- 12 Select **Edit→Copy**.

- 13 Use the **Window** menu to display again the model **Business Intelligence**.

- 14 Select **Edit→Paste**.

The tables and their references are copied into the new model.



- 15 Select **File**→**Save As**.

The File Save as dialog box appears.

- 16 Type **TUTORIAL2.PDM** in the File name box.

This is the name of the file in which you will work and save your modifications.

- 17 Click **Save**.

#### What you learned

In this section, you learned how to:

- ◆ Create a new PDM to design a data warehouse database
- ◆ Copy tables from an operational model into a data warehouse model

## Adding a table to the model

You will add the table SALE\_AUTHOR to the data warehouse model.

SALE\_AUTHOR will be used in the data warehouse database to gather information from the operational model. It will provide the sales amount and quantity per author in the data warehouse database.

You will create a reference between the new table and table AUTHOR.

- 1 In the physical diagram in the BUSINESS INTELLIGENCE model, click the **Table** tool in the palette.

- 2 Click in the physical diagram.

A table symbol appears at the click position.

- 3 Click the right mouse button.

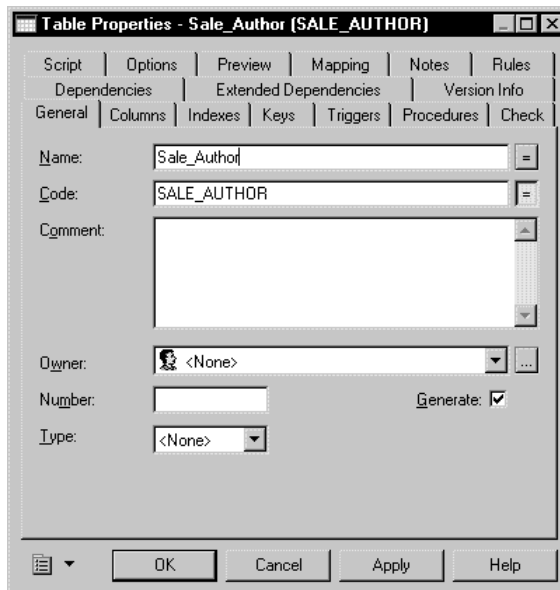
You release the **Table** tool.

- 4 Double-click the table symbol.

The table property sheet appears.

- 5 Type **Sale\_Author** in the Name box.

This is the name of the table. SALE\_AUTHOR appears automatically in the Code box. This is the code of the table.



The image shows a dialog box titled "Table Properties - Sale\_Author (SALE\_AUTHOR)". It has a tabbed interface with the following tabs: Script, Options, Preview, Mapping, Notes, Rules, Dependencies, Extended Dependencies, Version Info, General, Columns, Indexes, Keys, Triggers, Procedures, and Check. The "General" tab is selected. The dialog contains the following fields and controls:

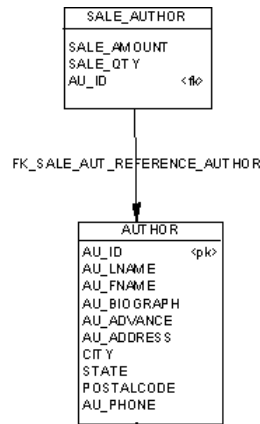
- Name:** A text box containing "Sale\_Author" with a "=" button to its right.
- Code:** A text box containing "SALE\_AUTHOR" with a "=" button to its right.
- Comment:** A large text area.
- Owner:** A dropdown menu showing "<None>" with a user icon and a "... button to its right.
- Number:** A text box.
- Generate:** A checkbox that is checked.
- Type:** A dropdown menu showing "<None>" with a dropdown arrow.

At the bottom of the dialog are four buttons: OK, Cancel, Apply, and Help.

- 6 Click the **Columns** tab to display the Columns page.
- 7 Click the **Add Columns** tool and select the following columns in the list:

Name	Data type	Primary key
SALE_AMOUNT	numeric(8,2)	—
SALE_QTY	numeric	—
AU_ID	char(12)	—

- 8 Click **OK** in the Selection list.  
The columns appear in the list of columns of the table SALE\_AUTHOR.
- 9 Click **OK**.
- 10 Click the **Reference** tool in the palette.
- 11 Click inside the SALE\_AUTHOR table and while continuing to hold down the mouse button, drag the cursor into the AUTHOR table and release the mouse button.

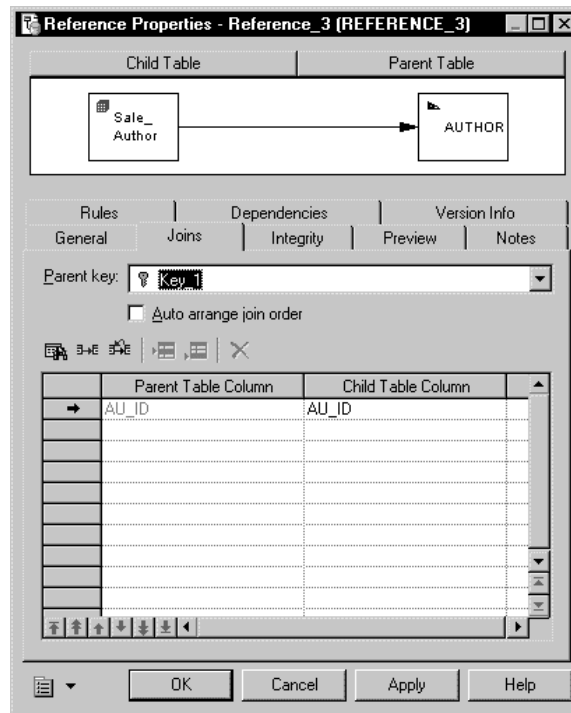


- 12 Click the **Pointer** tool in the palette to release the **Reference** tool.
- 13 Double-click the reference link between SALE\_AUTHOR and AUTHOR.

The reference property sheet appears.

- 14 Click the Joins tab.

The Joins page appears. You can verify that the parent and child columns are linked by a join within the reference.



- 15 Click **Cancel**.
- 16 Click the **Save** tool in the toolbar to save your model.

#### What you learned

In this section, you learned how to:

- ◆ Create a new table
- ◆ Add columns from a selection list
- ◆ Create a reference between tables

## Define a data source in the current model

You will create a data source in the new model BUSINESS INTELLIGENCE; this model is a data warehouse schema that needs to be populated with information from the operational model PUBLISHING INDUSTRY (PDMAFTER.PDM). PUBLISHING INDUSTRY is the data source for the new model.

The process of linking operational to data warehouse tables is called relational to relational mapping. When you create a relational to relational mapping, first you have to define a data source. A data source is used to define where data should be extracted to be transferred to another database.

### Operational model in the workspace

Make sure PDMAFTER.PDM is opened in the workspace, otherwise it will be impossible to define it as the data source of the new model.

- 1 Select **Model→Data Sources**.

The List of Data Sources appears.

- 2 Click the **Add a Row** tool.

An arrow appears at the beginning of the first empty line and a default name and code are entered.

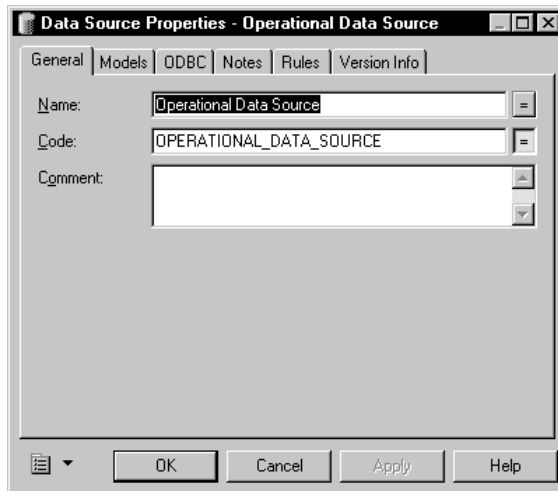
- 3 Type **Operational Data Source** in the Name column.

OPERATIONAL\_DATA\_SOURCE appears automatically in the Code column.

- 4 Click **Apply**.

- 5 Click the **Properties** tool.

The property sheet of the data source appears.



6 Click the **Models** tab to display the Models page.

7 Click the **Add Models** tool.

A selection list appears to let you select the models to include in the data source.

8 Select **Publishing Industry** and click OK.

The model appears in the list of models of the data source.

9 Click the **ODBC** tab to display the ODBC page.

The ODBC page allows to define the ODBC connection parameters to your database where the PUBLISHING INDUSTRY data are stored. To retrieve information from the operational database, you should have generated PUBLISHING INDUSTRY in a database.

10 Define the ODBC connection parameters.

11 Click **OK** in each of the dialog boxes.

#### What you learned

In this section, you learned how to:

- ◆ Create a data source
- ◆ Add a model to a data source

## Create a relational to relational mapping

You will map the table `SALE_AUTHOR` to tables in the operational model using the new data source. `SALE_AUTHOR` is designed to summarize the total sales amount and quantity per author in the data warehouse database. To do so, you have to define a relational to relational mapping between `SALE_AUTHOR` and operational tables.

- 1 Double-click the ***SALE\_AUTHOR*** table.

The table property sheet appears.

- 2 Click the ***Mapping*** tab.

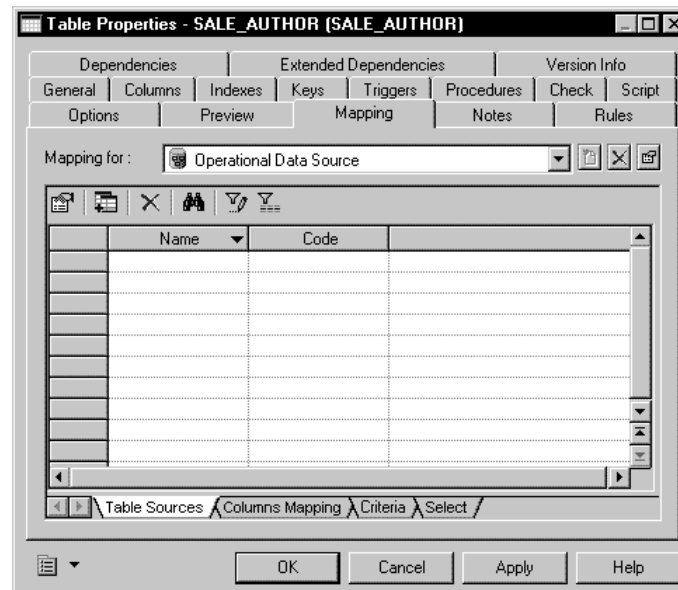
The Mapping page appears. The first time you define a mapping for a table, you have to define the data source for the current table.

- 3 Click the ***Add a Mapping for a Data Source*** tool beside the Mapping For box.

A data source selection dialog box appears.

- 4 Select ***Operational Data Source*** and click ***OK***.

The data source appears in the Mapping For box.



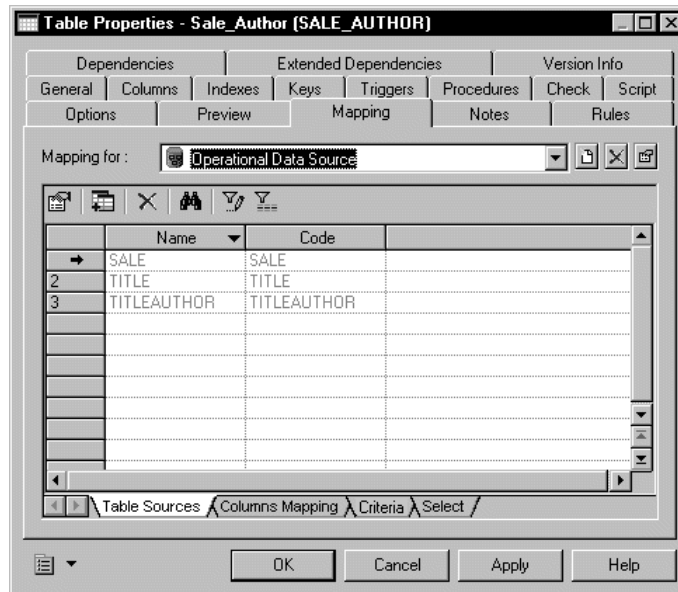


- 5 Click the **Add Objects** tool in the Table Sources page.

A table selection dialog box appears to let you select the tables in the operational model that will be linked to the table SALE\_AUTHOR.

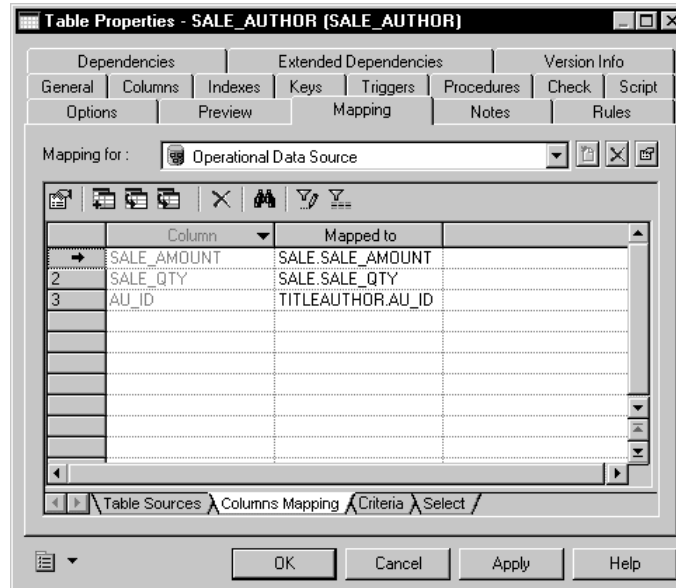
- 6 Select the tables **SALE**, **TITLE**, and **TITLEAUTHOR** and click OK in the selection dialog box.

The name and the code of the tables appear in the list of table sources.



- 7 Click the **Columns Mapping** tab to display the Columns Mapping page.

The list of columns is automatically filled with the columns having the same name and code in the current table and in the mapped tables.



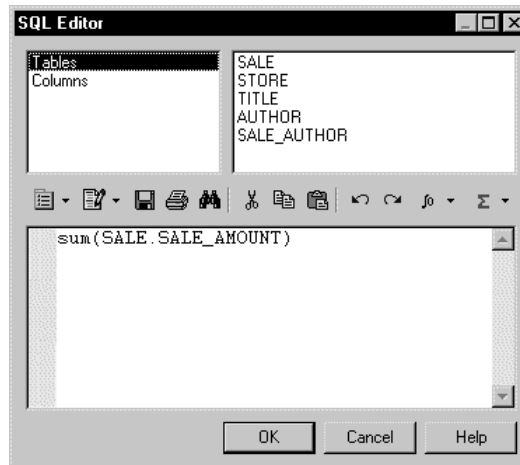
- 8 Click in the Mapped to column on the SALE\_AMOUNT line and click the *Ellipsis* button in the Mapped To column.

The SQL Editor appears. By default, the name of the column in the data source appears in the query script textbox. You will use the **sum** function to obtain the total amount of sales per author.

- 9 Type ***sum(SALE.SALE\_AMOUNT)*** in the query script textbox.

#### Functions in the SQL Editor

You can use the Functions tool in the SQL Editor dialog box to access functions to insert into statements



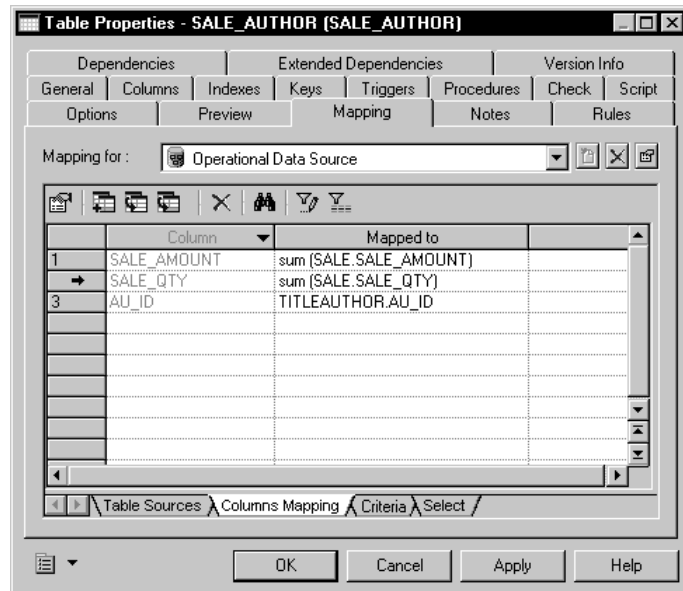
- 10 Click ***OK*** in the SQL Editor.
- 11 Click in the Mapped to column on the SALE\_QTY line and click the ***Ellipsis*** button in the Mapped To column.

The SQL Editor appears. You will also use the ***sum*** function to obtain the total quantity of sales per author.

- 12 Type ***sum(SALE.SALE\_QTY)*** in the query script textbox.

- 13 Click **OK** in the SQL Editor.

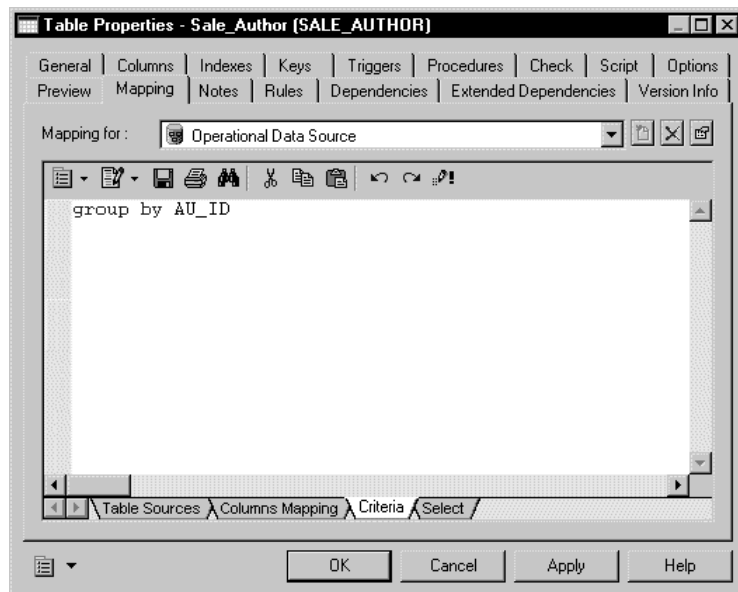
The Columns Mapping page displays the list of mapped columns.



- 14 Click the **Criteria** tab to display the **Criteria** page.

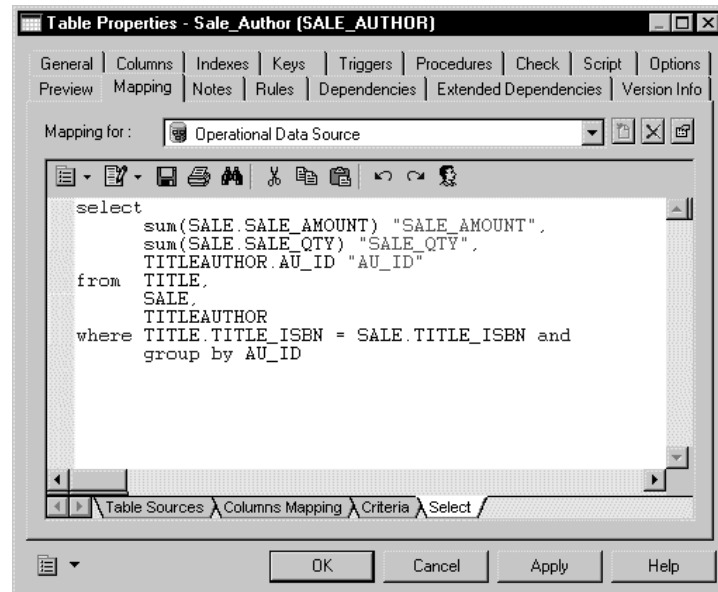
In this page, you can define join criteria between source tables. You will define a clause that will group sales amounts or sales quantities by author.

- 15 Type *group by AU\_ID*.



- 16 Click the *Select* tab to display the Select page and view the entire SQL statement.

The Select page displays the entire SQL statement that will be used to select data in the operational database to populate the data warehouse.



- 17 Click **OK**.
- 18 Click the **Save** tool in the toolbar to save your model.

#### What you learned

In this section, you learned how to:

- ◆ Define a relational to relational mapping between a table in the current model and several tables in the data source
- ◆ Sum column values to obtain global figures
- ◆ Insert a **group by** clause

## Generate an extraction script

The Generate Extraction Script feature allows to generate script files that will be used to fill and update tables in a data warehouse database.

In this lesson, you will generate an extraction script corresponding to the tables of the model BUSINESS INTELLIGENCE. One table in BUSINESS INTELLIGENCE is mapped with tables in PUBLISHING INDUSTRY via a data source called OPERATIONAL DATA SOURCE.

The extraction script is generated for OPERATIONAL DATA SOURCE. It contains the **select** order defined for the table SALE\_AUTHOR mapped to tables in the data source.

You will be able to use this extraction script with an extraction tool to create and fill the tables in your data warehouse database.

- 1 Select **Database**→**Generate Extraction Scripts**.

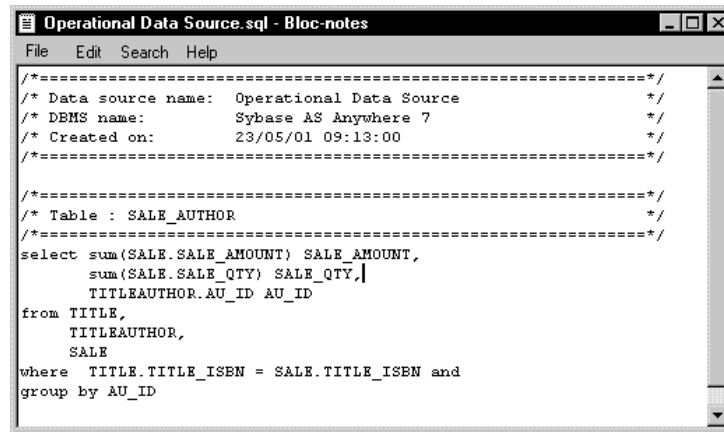
The Extraction Script Generation dialog box appears.

- 2 Select a destination directory in the **Directory** box.
- 3 Select or clear the following generation options:

Group box	Selected item
Text	Character set: Windows (ANSI)
	Character case: Mixed
Usage	Title

- 4 Click **OK**.  
The Output window indicates that the generation was successful.
- 5 In your Windows Explorer, browse to the directory where the extraction script was generated and open the file OPERATIONAL DATA SOURCE.sql.

The extraction script contains the select order defined in the Mapping tab of SALE\_AUTHOR.



```
Operational Data Source.sql - Bloc-notes
File Edit Search Help
/*=====*/
/* Data source name: Operational Data Source */
/* DBMS name: Sybase AS Anywhere 7 */
/* Created on: 23/05/01 09:13:00 */
/*=====*/

/*=====*/
/* Table : SALE_AUTHOR */
/*=====*/
select sum(SALE.SALE_AMOUNT) SALE_AMOUNT,
       sum(SALE.SALE_QTY) SALE_QTY,
       TITLEAUTHOR.AU_ID AU_ID
from TITLE,
     TITLEAUTHOR,
     SALE
where TITLE.TITLE_ISBN = SALE.TITLE_ISBN and
group by AU_ID
```

- 6 Select *File*→*Close*.



# Using Multidimensional Features

In this lesson, you will use the multidimensional diagram to design the structure of your OLAP database. You will use the data warehouse database structure defined in the model BUSINESS INTELLIGENCE.

Retrieve  
multidimensional  
objects

The Retrieve Multidimensional Objects feature allows to identify fact and dimension tables in the data warehouse database schema. It assigns multidimensional types to the model tables.

Rebuild Cube

The Rebuild Cubes feature creates cubes and dimensions from fact and dimension tables. You will use this functionality to switch to the multidimensional diagram modeling environment.

Generate Cube  
Data

The Generate Cube Data feature allows to generate text files containing the data of a cube. The cube query is used to extract data from the data warehouse database and to insert them into the generated text file. These text files can be used by OLAP tools to create and populate tables in the OLAP engine.

To make an efficient use of the Generate Cube Data feature, you need to have the data warehouse database generated in order to recover data from the database and fill the text file.

In this lesson you will:

- ◆ Retrieve multidimensional objects in the physical diagram
- ◆ Rebuild cubes and observe multidimensional objects
- ◆ Generate a cube data in text file

## How long will it take?

About 10 minutes.

## Retrieve multidimensional objects

You will use the Retrieve Multidimensional Objects feature to automatically assign multidimensional types to the tables of model BUSINESS INTELLIGENCE.

- 1 Select **Tools**→**Multidimension**→**Retrieve Multidimensional Objects**.

The Multidimensional Object Retrieval dialog box appears.

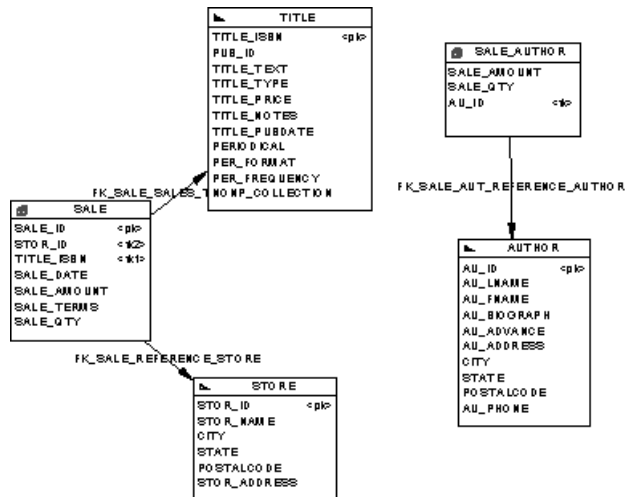
- 2 Select both **Retrieve Fact** and **Retrieve Dimension** check boxes in the General Page.

By default, all tables are selected in the Selection page.



3 Click **OK**.

Child tables (SALE and SALE\_AUTHOR) become fact tables and parent tables (TITLE, STORE and AUTHOR) become dimension tables as shown below.



What you learned

In this section, you learned how to:

- ◆ Use the Retrieve Multidimensional Objects feature
- ◆ Assign types to tables

## Rebuilding Cubes

You will rebuild cubes to automatically create cubes and dimensions from the fact and dimension tables of your model. The new facts and dimensions are displayed in a multidimensional diagram.

The Rebuild Cubes automatically creates a relational to multidimensional mapping between data warehouse tables and multidimensional facts and dimensions.

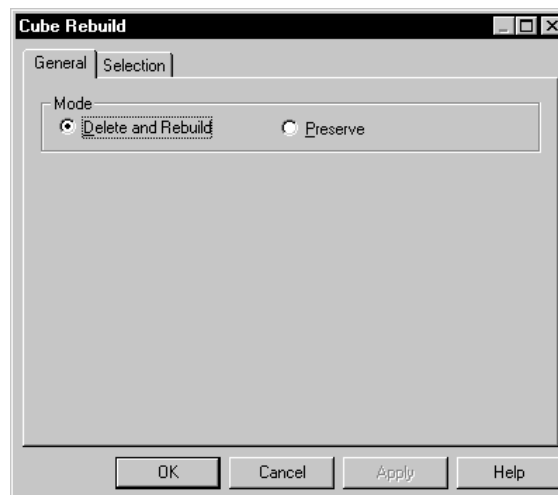
When you rebuild cubes, a new data source is automatically created in the current model to allow the transfer of data from the data warehouse database (in the physical diagram) to the OLAP database (in the multidimensional diagram).

- 1 Select **Tools**→**Multidimension**→**Rebuild Cubes**.

The Cube Rebuild dialog box appears.

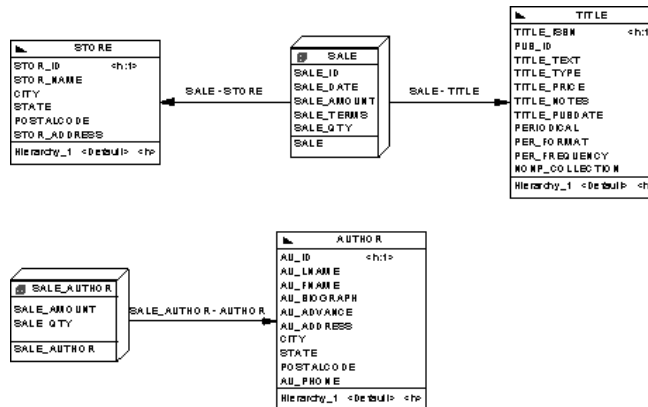
- 2 Select the **Delete and Rebuild** radio button in the Mode groupbox.

By default, all fact tables are selected in the Selection page. These fact tables will become cubes in the multidimensional diagram.



3 Click **OK**.

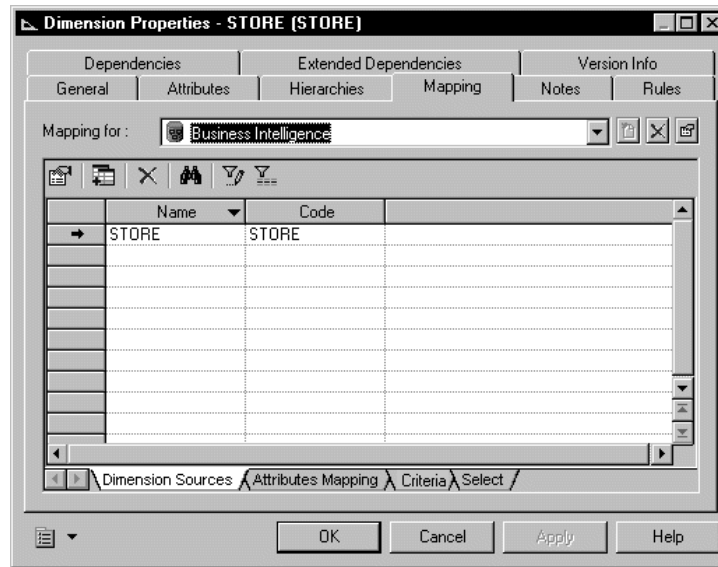
A new multidimensional diagram appears and displays the symbols of the new objects as shown below:



When you use rebuild cubes, the tables are automatically mapped to the new cubes and dimensions. A new data source called Business Intelligence is automatically created to express the link between the data warehouse and multidimensional objects. By default, this data source has no ODBC connection defined, you should specify these parameters if you want to retrieve data from the data warehouse database.

4 Double-click the **Store** dimension to display the property sheet.

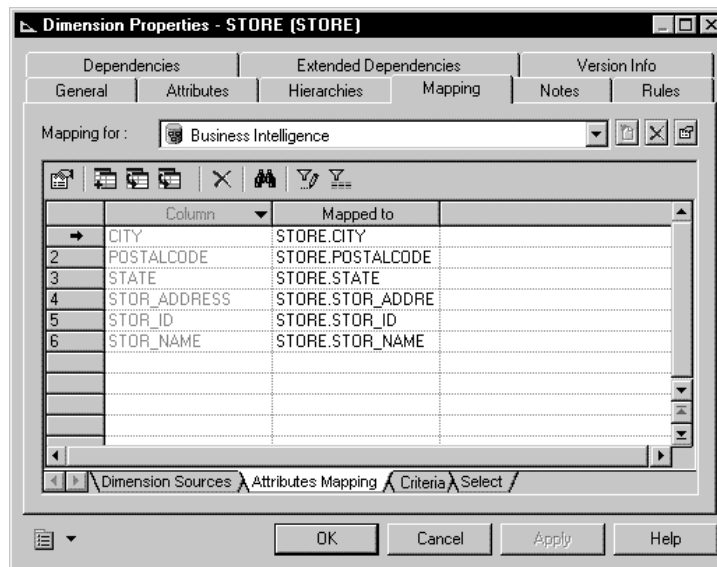
- 5 Click the *Mapping* tab.



As you can see, the current dimension is mapped to the table **STORE** in the physical diagram. The model **BUSINESS INTELLIGENCE** is its own data source.

- Click the **Attributes Mapping** tab in the **Mapping** page.

The table columns are mapped to the attributes of the new dimension.



- 7 Click **Cancel**.
- 8 Double-click the **Sale\_Author** cube to display the property sheet.
- 9 Click the **Properties** tool beside the Fact box to display the fact property sheet.

10 Click the **Mapping** tab.



In the Fact Sources page, you can check that SALE\_AUTHOR is mapped to table SALE\_AUTHOR. The measures are also mapped to table columns as you can check in the Measures Mapping page.

11 Click **Cancel** in each of the dialog boxes.

#### What you learned

In this section, you learned how to:

- ◆ Use the Rebuild Cubes feature
- ◆ Observe automatic mapping created after rebuild cube



## Generate Cube Data

You will use the Generate Cube Data feature to create text files for each cube in the multidimensional diagram. This text file can be used by an OLAP tool to create and populate cubes in the OLAP engine.

### Generated data warehouse database

You need to have generated the data warehouse database in order to recover data from the database and fill the text file. To do so, you can use the extraction scripts with your data warehouse extraction tool.

You also need to define the ODBC connection parameters in the data source property sheet to access the generated data warehouse database

- 1 Select **Tools**→**Generate Cube Data**.

The Cube Rebuild dialog box appears.

- 2 Select a destination directory in the **Directory** box.
- 3 Select or clear the following generation options:

Group box	Selected item
Text	Character set: Windows (ANSI)
	Character case: Mixed
File	Header
	Extension: txt
	Separator: ;
	Delimiter: "

- 4 Click the Selection tab to display a cube and data source selection list.
- 5 Make sure **SALE** and **SALE\_AUTHOR** are selected in the Cubes tab.
- 6 Click the Data Source tab and clear the **Operational Data Source** check box.

By doing so, you will generate one file per cube for the Business Intelligence data source.

- 7 Click **OK**.

The Output window indicates that the generation was successful. The files **SALE - Business Intelligence.txt** and **SALE\_AUTHOR - Business Intelligence.txt** are generated in the destination directory.

What you learned

In this section, you learned how to:

- ◆ Use the Generate Cube Data feature

## Exit PowerDesigner

You will save and close the PDM, then exit PowerDesigner.

- 1 Select **File**→**Save**.

This saves the PDM.

- 2 Select **File**→**Close**.

This closes the model.

- 3 Select **File**→**Exit**.

A confirmation box asks you if you want to save the Workspace.

- 4 Click the **No** button.

You exit the PowerDesigner application.



# Glossary

abstract data type (ADT)	User defined data type which encapsulates a range of data values and functions. The functions are both defined on, and operate on the set of values
alternate key	Column or columns whose values uniquely identify a row in the table and are not primary key columns
association	Shows the axis of investigation of the dimension in the cube
attribute	Qualifies dimensions used in queries. For example, the Time dimension can contain attributes Year, Quarter, Month, and Week
binary long object	BLOB, a data type used for large binary objects such as images
business rule	Written statement specifying what the information system must do or how it must be structured to support business needs
clustered index	Index in which the physical order and the logical (indexed) order is the same
column	Data structure that contains an individual data item within a row (record), model equivalent of a database field
column denormalization	Used to eliminate frequent joins between tables and improve query performance
Conceptual Data Model (CDM)	Entity-relationship diagram that models the information system without considering the details of physical implementation
constraint	Named check that enforces data requirements, default values, or referential integrity on a table or a column
cube	Collection of measures corresponding to values stored into each of its data cell. The measures are organized into dimensions to provide for faster retrieval and drill-down

data source	Identification of the data to access, its operating system, DBMS, and network platform
DBMS	Text file in XML format that contains all the SQL syntax and specifications for a Database Management System
dimension	Axis of analysis in a multidimensional structure
dimension table	Table that stores data related to the axis of investigation of a fact. For example, geography, time, product
domain	Set of values for which a data item is valid
fact	Set of measures manipulated by the cube. For example, Sale, Revenue, Budget are facts
fact table	Table that stores variable numerical values related to aspects of a business
foreign key	Column or columns whose values depend on and migrate from a primary key, or an alternate key, in another table
hierarchy	Defines navigational and consolidation paths through dimensions
horizontal partitioning	Consists in segmenting a table into multiple tables each containing a subset of rows and the same columns as the partitioned table.
index	Data structure that is based on a key and that speeds access to data and controls unique values
join	Within a reference, a link between each foreign key column referring to a matching primary key column
measure	Variable that corresponds to the focus of an investigation, it describes the meaning of the analytical values stored in each data cell of the cube. Measures are most of the time numeric values like for example Price or Total
ODBC	Open Database Connectivity (ODBC) interface which gives PowerDesigner access to data in database management systems (DBMS)
ODBC driver	Part of the Open Database Connectivity (ODBC) interface that processes ODBC functions calls, submits SQL requests to a specific data source, and returns results to the application

OLAP	Online Analytical Processing is a class of analytic application software that exposes business data in a multidimensional format
permission	Usage restrictions set on a particular database object for a particular user, group or role
Physical Data Model (PDM)	Table-reference diagram that models the information system including the details of physical implementation
primary key	Column or columns whose values uniquely identify a row in a table
privilege	Set of rights assigned to a database user, group, or role
property sheet	Window that displays the properties of an object
reference	Link between a parent table and a child table. A reference can link tables by shared keys or by specified columns.
referential integrity	Rules governing data consistency, specifically the relationships among primary keys and foreign keys of different tables
storage	Named partition that stores tables and indexes on a storage device
synonym	Alternative name for various types of objects used to mask the name and owner of an object, provide location transparency for remote objects of a distributed database, and simplify SQL statements for database users
table	Collection of rows (records) that have associated columns (fields)
table collapsing	Consists in merging tables into a single table in order to eliminate joins and to improve query performance
tablespace	Named partition that stores tables and indexes in a database
trigger	Special form of stored procedure that goes into effect when you insert, delete, or update a specified table or column
user	Name that identifies a person or group working with objects in a PDM, and which has the ability to own other objects in the PDM
unique index	Index in which no two rows can have the same index value, including NULL

vertical partitioning	Consists in segmenting a table into multiple tables each containing a subset of columns and the same number of rows as the partitioned table.
view	Alternate way of looking at the data in one or more tables. Usually created as a subset of columns from one or more tables



# Index

## A

- abstract data type
  - define 71, 73
  - Java class 73
  - specify 73
- add
  - column 25
  - table 23, 107
- ADT *See* abstract data type
- alternate key
  - create 39
  - index 39
- assign
  - data profile 96
  - test data profile 85
- auto-migrate foreign key 16

## C

- close PDM 131
- column
  - add 25
  - create 34
  - foreign key 21
  - mapping 112
  - primary key 21
  - sort 27
- copy table 104
- create
  - alternate key 39
  - column 34
  - index 36, 39
  - model 104
  - primary key 36
  - reference 45
  - table 21
  - test data profile 89
  - trigger 57
  - view 51, 52

- cube 124
  - generate cube data 129
  - rebuild 124
- cube data 129
- customize view 53

## D

- data profile
  - assign 96
  - automatic data source 94
- data source 124
  - define 110
- data warehouse 103
- database
  - creation script 82
  - generate 71, 81, 82
- define
  - abstract data type 71, 73
  - cascade 48
  - data source 110
  - index 21
  - option 16
  - preference 16
  - reference 43, 46
  - referential integrity 43, 48
  - test data 85
  - test data profile 85
  - test data script 85
- delete
  - object 15
  - symbol 15
- detach symbol 15
- dimension 124
- dimension table 122
- domain
  - sort 30

## E

- extraction script 119

## F

fact table 122  
foreign key  
    auto-migrate 16  
    column 21  
    index 21  
    reference 43  
    referential integrity 43

## G

generate  
    automatic test data 91  
    cube data 129  
    database 71, 81, 82  
    extraction script 119  
    test data from file 94  
    test data script 100  
    trigger script 67

## I

implement trigger referential integrity 59  
index  
    create 36, 39  
    define 21  
    foreign key 21  
    primary key 21, 36  
    search 21  
item 57  
item template 57

## J

Java class  
    access property 73, 77  
    link to OOM 71

## M

mapping 124  
    column 112  
    relational to relational 112  
    table 112  
multidimensional diagram 121

## N

name  
    reference 46

## O

Object-Oriented Model  
    access Java class 77  
    create 76  
OOM *See* Object-Oriented Model  
    link Java class 71  
open  
    PDM 5, 6  
option  
    define 16  
    PDM 16

## P

PDM 1  
    close 131  
    denormalize 21  
    file 4  
    open 5, 6  
    option 16  
    preference 16  
    save 20, 131  
preference  
    define 16  
    PDM 16  
primary key  
    column 21  
    create 36  
    index 21, 36  
    reference 43  
    referential integrity 43  
procedure script 67  
property  
    reference 46  
    table 23

## R

rebuild cubes 124  
    automatic mapping 124  
    data source 124  
rebuild trigger 57

- reference
  - create 45
  - define 43, 46
  - foreign key 43
  - join 43
  - name 46
  - primary key 43
  - property 46
  - table 45
- referential integrity
  - define 43, 48
  - foreign key 43
  - primary key 43
  - trigger 57, 59
- relational to relational mapping 112
- retrieve multidimensional objects 122

## S

- save PDM 20, 131
- script
  - create database 71, 81
  - create trigger 57
  - generate test data 100
  - generate trigger 67
  - procedure 67
  - trigger 67
  - view 100
- sort
  - columns 27
  - domain 30
- specify abstract data type 73
- symbol
  - delete 15
  - detach 15

## T

- table
  - add 23, 107
  - copy 104
  - create 21
  - mapping 112
  - property 23
  - reference 45
- template
  - item 57
  - trigger 57

- test data 85
  - assign data profile 96
  - assign profile 85
  - automatic generation source 91
  - create profile 89
  - data profile 85
  - file generation source 94
  - generate data source 91, 94
  - generate script 100
- test data profile
  - assign 85
  - automatic data source 91
  - create 89
  - define 85
- test data script 85
- text file 129
- trigger 57
  - create 57
  - implementation 59
  - rebuild 57
  - referential integrity 57
  - script 57, 67
  - template 57

## V

- view
  - create 51, 52
  - customize 53
  - define 51

