Data Modeling

PDF download from SAP Help Portal:

Created on February 15, 2017

The documentation may have changed since you downloaded the PDF. You can always find the latest information on SAP Help Portal.

Note
This PDF document contains the selected topic and its subtopics (max. 150) in the selected structure. Subtopics from other structures are not included. The selected structure has more than 150 subtopics. This download contains only the first 150 subtopics. You can manually download the missing subtopics.
Table of content

1 Data Modeling
2 Building Data Models
2.1 Getting Started with Data Modeling
2.1.1 Conceptual Data Models
2.1.2 Logical Data Models
2.1.3 Physical Data Models
2.1.4 Creating a Data Model
2.1.4.1 Data Model Properties
2.1.4.1.1 Creating a Database in the Model
2.1.5 Customizing your Modeling Environment
2.1.5.1 Setting CDMLDM Model Options
2.1.5.1.1 Assertion Template
2.1.5.2 Setting PDM Model Options
2.1.5.2.1 Column and Domain Model Options
2.1.5.2.2 Reference Model Options
2.1.5.2.3 Other Object Model Options
2.1.5.3 Setting Data Model Display Preferences
2.1.5.4 Viewing and Editing the DBMS Definition File
2.1.5.4.1 Changing the DBMS
2.1.5.5 Extending your Modeling Environment
2.1.5.6 Traceability Links
2.2 Conceptual and Logical Diagrams
2.2.1 Supported CDM/LDM Notations
2.2.2 Conceptual Diagrams
2.2.2.1 Conceptual Diagram Objects
2.2.2.2 Example: Building a Data Dictionary in a CDM
2.2.3 Logical Diagrams
2.2.3.1 Logical Diagram Objects
2.2.3.2 Importing a Deprecated PDM Logical Model
2.2.3.3 Importing Multiple Interconnected PDM Logical Models
2.2.4 Data Items (CDM)
2.2.4.1 Creating a Data Item
2.2.4.2 Data Item Properties
2.2.4.3 Controlling Uniqueness and Reuse of Data Items
2.2.5 Entities (CDM/LDM)
2.2.5.1 Creating an Entity
2.2.5.2 Entity Properties
2.2.5.3 Copying Entities
2.2.5.4 Displaying Attributes and Other Information on an Entity Symbol
2.2.6 Attributes (CDM/LDM)
2.2.6.1 Creating an Attribute
2.2.6.2 Attribute Properties
2.2.6.3 Deleting Attributes (CDM)
2.2.7 Identifiers (CDM/LDM)
2.2.7.1 Creating an Identifier
2.2.7.2 Identifier Properties
2.2.8 Relationships (CDM/LDM)
2.2.8.1 Creating a Relationship
2.2.8.2 Relationship Properties
2.2.8.3 Enabling Many-to-many Relationships in an LDM
2.2.8.4 Creating a Reflexive Relationship
2.2.8.5 Defining a Code Option for Relationships
2.2.8.6 Changing a Relationship into an Associative Entity
2.2.8.7 Identifier Migration Along Relationships
2.2.9 Associations and Association Links (CDM)
2.2.9.1 Creating an Association with Links
2.2.9.2 Creating an Association Without Links
2.2.9.3 Association Properties
2.2.9.4 Association Link Properties
2.2.9.5 Creating a Reflexive Association
2.2.9.6 Defining a Dependent Association
2.2.9.7 Changing an Association into an Associative Entity
2.2.9.8 Creating an Association Attribute
2.2.10 Inheritances (CDM/LDM)
2.2.10.1 Creating an Inheritance
2.2.10.2 Inheritance Properties
2.2.10.3 Making Inheritance Links Mutually Exclusive
2.3 Physical Diagrams
2.3.1 Physical Diagram Objects
2.3.2 Tables (PDM)
2.3.2.1 Creating a Table
2.3.2.2 Table Properties
2.3.2.3 Linking a Table to an Abstract Data Type
2.3.2.4 Creating an XML Table or View
2.3.2.5 Specifying Table Constraints
2.3.2.6 Denormalizing Tables and Columns
2.3.2.6.1 Horizontal Partitions
2.3.2.6.2 Vertical Partitions
2.3.2.6.3 Table Collapsings
2.3.2.6.4 Column Denormalization
2.3.2.6.5 Denormalization Object Properties
2.3.2.6.6 Removing Partitionings and Table Collapsings
2.3.2.7 PowerBuilder DataWindow Extended Attributes
2.3.2.8 Displaying Column, Domain, and Data Type Information on a Table Symbol
2.3.2.9 Physical Options (PDM)
2.3.2.9.1 Defining Default Physical Options
2.3.3 Columns (PDM)
2.3.3.1 Creating a Column
2.3.3.2 Column Properties
2.3.3.3 Setting Data Profiling Constraints
2.3.3.3.1 Specifying Constraints Through Business Rules
2.3.3.3.2 Creating Data Formats For Reuse
2.3.3.3.3 Specifying Advanced Constraints
2.3.3.4 Populating Columns with Test Data
2.3.3.4.1 Test Data Profile Properties
2.3.3.4.2 Assigning Test Data Profiles to Columns
2.3.3.5 Creating a Computed Column
2.3.3.6 Attaching a Column to a Domain
2.3.3.7 Copying or Replicating a Column from Another Table
2.3.4 Primary, Alternate, and Foreign Keys (PDM)
2.3.4.1 Creating Primary Keys
2.3.4.1.1 Rebuilding Primary Keys
2.3.4.2 Creating Alternate Keys
2.3.4.3 Creating Foreign Keys
2.3.4.4 Key Properties
2.3.5 Indexes (PDM)
2.3.5.1 Creating Standard, Key, or Function-Based Indexes
2.3.5.2 Index Properties
2.3.5.3 Rebuilding Indexes
2.3.6 Views (PDM)
2.3.6.1 Creating a View
2.3.6.2 View Properties
2.3.6.3 View Queries
2.3.6.4 Materialized Views
2.3.6.5 Showing View Dependencies using Traceability Links
2.3.6.6 Defining a Generation Order for Views
2.3.7 Triggers (PDM)
2.3.7.1 Creating a Table or View Trigger
2.3.7.2 Creating Triggers from References
2.3.7.3 Creating a DBMS Trigger
2.3.7.4 Trigger and DBMS Trigger Properties
2.3.7.5   Trigger Naming Conventions
2.3.7.6   Calling a Related Procedure in a Trigger Template
2.3.7.7   Indicating Trigger Order for Multiple Triggers
2.3.7.8   Defining Triggers with Multiple Events
2.3.7.9   Rebuilding Triggers
2.3.7.10  Trigger Templates
2.3.7.11  Trigger Template Items
2.3.7.11.1 PowerDesigner Pre-Defined Trigger Template Items
2.3.7.12  Creating SQL/XML Queries with the Wizard
2.3.7.13  Generating Triggers and Procedures
2.3.7.13.1 Defining a Generation Order for Stored Procedures
2.3.7.13.2 Creating User-Defined Error Messages
2.3.7.13.2.1 Generating a User-Defined Error Message
2.3.8    Stored Procedures and Functions (PDM)
2.3.8.1   Creating a Stored Procedure or Function
2.3.8.2   Procedure Properties
2.3.8.3   Tracing Trigger and Procedure Dependencies
2.3.8.3.1 Creating Procedure Dependencies Manually
2.3.8.4   Attaching a Stored Procedure to a Table
2.3.8.4.1 Rebuilding Procedures Attached to Tables
2.3.8.5   Procedure Templates (PDM)
2.3.9    Users, Groups, and Roles (PDM)
2.3.9.1   Creating a User, Group, or Role
2.3.9.2   User, Group, and Role Properties
2.3.9.3   Assigning an Owner to an Object
2.3.9.4   Granting System Privileges
2.3.9.4.1 Generating Privileges
1 Data Modeling

A data model is a representation of the information consumed and produced by a system, which lets you analyze the data objects present in the system and the relationships between them.

Building Data Models

The chapters in this part explain how to model your data systems in SAP® PowerDesigner®.

In this section:
- Getting Started with Data Modeling
- Conceptual and Logical Diagrams
- Physical Diagrams
- Multidimensional Diagrams
- Checking a Data Model
- Generating and Reverse-Engineering Databases
- Generating Other Models from a Data Model
- Migrating from ERwin to PowerDesigner

2.1 Getting Started with Data Modeling

A data model is a representation of the information consumed and produced by a system, which lets you analyze the data objects present in the system and the relationships between them. PowerDesigner provides conceptual, logical, and physical data models to allow you to analyze and model your system at all levels of abstraction.

Suggested Bibliography


In this section:
- Conceptual Data Models
- Logical Data Models
- Physical Data Models
- Creating a Data Model
- Customizing your Modeling Environment

2.1.1 Conceptual Data Models

A conceptual data model (CDM) helps you analyze the conceptual structure of an information system, to identify the principal entities to be represented, their attributes, and the relationships between them. A CDM is more abstract than a logical (LDM) or physical (PDM) data model.

A CDM allows you to:
- Represent the organization of data in a graphic format to create Entity Relationship Diagrams (ERD).
- Verify the validity of data design.
- Generate a Logical Data Model (LDM), a Physical Data Model (PDM) or an Object-Oriented Model (OOM), which specifies an object representation of the CDM using the UML standard.

To create a CDM, see Creating a Data Model. For detailed information about conceptual diagrams, see Conceptual Diagrams.

2.1.2 Logical Data Models

A logical data model (LDM) helps you analyze the structure of an information system, independent of any specific physical database implementation. An LDM has migrated entity identifiers and is less abstract than a conceptual data model (CDM), but does not allow you to model views, indexes and other elements that are available in the more concrete physical data model (PDM).

You can use a logical model as an intermediary step in the database design process between the conceptual and physical designs:
- Start with a CDM containing entities, attributes, relationships, domains, data items and business rules. If need be, you may develop the CDM in several design steps starting from a high level model to a low level CDM
- Generate an LDM. Create indexes and specify FK column names and other common features
- Generate one or more PDMs, each targeted to a specific DBMS implementation

This design process allows you to keep everything consistent in a large development effort.

To create an LDM, see Creating a Data Model. For detailed information about logical diagrams, see Logical Diagrams.

2.1.3 Physical Data Models

A physical data model (PDM) helps you to analyze the tables, views, and other objects in a database, including multidimensional objects necessary for data warehousing. A PDM is more concrete than a conceptual (CDM) or logical (LDM) data model. You can model, reverse-engineer, and generate for all the most popular DBMSs.
PowerDesigner provides you with tools for modeling your operational and business intelligence environments:

- Operational/relational environment - modeled in physical diagrams (see Physical Diagrams). The physical analysis may follow a conceptual and/or logical analysis, and addresses the details of the actual physical implementation of data in a database, to suit your performance and physical constraints.
- Business intelligence environment:
  - Data warehouse or data mart database tables - can be modeled in physical diagrams and mapped to their source operational tables to generate data extraction scripts.
  - Data warehouse cubes (in ROLAP or HOLAP environments) - can be modeled in multidimensional diagrams (see Multidimensional Diagrams) and mapped to their source warehouse tables.
  - SAP® BusinessObjects™ Universes - can be generated from warehouse PDMs for direct consumption or for editing in BusinessObjects environments (see Generating an SAP BusinessObjects Universe).
  - OLAP cubes - can be modeled in multidimensional diagrams and mapped to their source operational or warehouse tables to generate cube data.

PowerDesigner provides support for a wide range of database families through DBMS definition files (*.xdb, located in Resource Files\DBMS inside your installation directory), which customize the metamodel to support the specific syntax of a DBMS, through extended attributes, objects, and generation templates. To view and edit the resource file for your DBMS, select Database > Edit Current DBMS. For detailed information about working with these files, see Customizing and Extending PowerDesigner > DBMS Definition Files.

### 2.1.4 Creating a Data Model

You create a new data model by selecting File > New Model.

**Context**

**Note**

In addition to creating a data model from scratch with the following procedure, you can also:

- create a CDM by importing an ERwin model (.ERX) or by generating it from another PowerDesigner model.
- create an LDM by generating it from another PowerDesigner model.
- create a PDM by reverse-engineering it from an existing database (see Reverse Engineering a Database into a PDM) or generating it from another PowerDesigner model.

The New Model dialog is highly configurable, and your administrator may hide options that are not relevant for your work or provide templates or predefined models to guide you through model creation. When you open the dialog, one or more of the following buttons will be available on the left hand side:

- **Categories**, which provides a set of predefined models and diagrams sorted in a configurable category structure.
- **Model types**, which provides the classic list of PowerDesigner model types and diagrams.
- **Template files**, which provides a set of model templates sorted by model type.
Procedure

1. Select File > New Model to open the New Model dialog.
2. Click a button, and then select a category or model type (Conceptual Data Model, Logical Data Model or Physical Data Model) in the left-hand pane.
3. Select an item in the right-hand pane. Depending on how your New Model dialog is configured, these items may be first diagrams or templates on which to base the creation of your model.
   Use the Views tool on the upper right hand side of the dialog to control the display of the items.
4. Enter a model name. The code of the model, which is used for script or code generation, is derived from this name using the model naming conventions.
5. [PDM only] Select a target DBMS, which customizes PowerDesigner's default modifying environment with target-specific properties, objects, and generation templates.
   By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the Embed Resource in Model button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.
6. [optional] Click the Select Extensions button and attach one or more extensions to your model.
7. Click OK to create and open the data model.

Note
Sample data models are available in the Example Directory.

In this section:
- Data Model Properties

Data Model Properties

You open the model property sheet by right-clicking the model in the Browser and selecting Properties.

Each data model has the following model properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the model. The name should clearly convey the model's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the model. By default, the code is auto-generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the (Code) field.</td>
</tr>
<tr>
<td>Filename</td>
<td>Specifies the location of the model file. This box is empty if the model has never been saved.</td>
</tr>
<tr>
<td>Author</td>
<td>Specifies the author of the model. If you enter nothing, the Author field in diagram title boxes displays the user name from the model property sheet Version Info tab. If you enter a space, the Author field displays nothing.</td>
</tr>
<tr>
<td>Version</td>
<td>Specifies the version of the model. You can use this box to display the repository version or a user-defined version of the model. This parameter is defined in the display preferences of the Title node.</td>
</tr>
<tr>
<td>DBMS</td>
<td>[PDM only] Specifies the model target.</td>
</tr>
<tr>
<td>Database</td>
<td>Specifies the database that is the target for the model. You can create a database in the model by clicking the Create tool to...</td>
</tr>
</tbody>
</table>
In this section:

- Creating a Database in the Model

## 2.1.4.1.1 Creating a Database in the Model

You can create a database from the General tab of the model property sheet or, if your DBMS supports multiple databases in a single model, from the list of databases in the Model menu.

### Context

Creating a database in your model allows you to specify physical options for it and to reference the database in the physical options of other objects. For example, you can specify that a table is created in a particular tablespace with the in 
```plaintext
<db> <tablespace>
```
physical option:

### Procedure

1. Select Model > Model Properties or right-click the diagram background and select Properties.  
2. Click the Create tool to the right of the Database field and click Yes in the confirmation dialog to open the database property sheet.  
3. Enter the following properties for the database as appropriate:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>DBMS</td>
<td>DBMS for the database</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- **Physical Options**: Specifies the physical options associated with the database (see Physical Options (PDM)).
- **Script**: Specifies begin and end scripts to bookend the database creation script (see Customizing Creation Statements).
- **Rules**: Specifies the business rules associated with the database (see Business Rules (CDM/LDM/PDM)).
- **Dependencies**: Lists the objects that reference the database in their physical options.

## 2.1.5 Customizing your Modeling Environment

The PowerDesigner data model provides various means for customizing and controlling your modeling environment. In this section:

- Setting CDM/LDM Model Options
- Setting PDM Model Options
2.1.5.1 Setting CDM/LDM Model Options

You can set CDM/LDM model options by selecting Tools > Model Options or right-clicking the diagram background and selecting Model Options. You can set the following options on the Model Settings page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code case sensitive</td>
<td>Specifies that the names and codes for all objects are case sensitive, allowing you to have two objects with identical names or codes but different cases in the same model. If you change case sensitivity during the design process, we recommend that you check your model to verify that your model does not contain any duplicate objects.</td>
</tr>
<tr>
<td>Enable links to requirements</td>
<td>Displays a Requirements tab in the property sheet of every object in the model, which allows you to attach requirements to objects (see Requirements Modeling).</td>
</tr>
<tr>
<td>Enforce non-divergence</td>
<td>Specifies that attributes attached to a domain must remain synchronized with the selected properties (see Controlling Non-Divergence from a Domain).</td>
</tr>
<tr>
<td>Use data type full name</td>
<td>Specifies that the complete data type is displayed in entity symbols.</td>
</tr>
<tr>
<td>Default data type</td>
<td>Specifies the default data type to apply to domains and attributes if none is selected for them.</td>
</tr>
<tr>
<td>External Shortcut Properties</td>
<td>Specifies the properties that are stored for external shortcuts to objects in other models for display in property sheets and on symbols. By default, all properties appear, but you can select to display only Name/Code to reduce the size of your model.</td>
</tr>
</tbody>
</table>

For more information about these notations, see Supported CDM/LDM Notations.

Notation

You can choose between the following notations:
- Entity / Relationship [Default – used throughout this manual] Entity/relationship notation connects entities with links representing one of four relationships between them. These relationships have properties that apply to both entities involved in the relationship.
- Merise - uses associations instead of relationships
- E/R + Merise - both entity/relationship and Merise are used in the same model
- IDEFIX - data modeling notation for relationships and entities. In this notation, each set of relationship symbols describes a combination of the optionality and cardinality of the entity next to it.
- Barker - inceptions (inheritances) are represented by placing the child entities inside the parent entity symbol, and relationships are drawn in two parts, each reflecting the multiplicity of the associated entity role.

For more information about these notations, see Supported CDM/LDM Notations.

Unique code

Requires that data items or relationships have unique codes.

Allow n-n relationships

[CDM only] Allows n-n relationships to be displayed.

Allow reuse

Allows the reuse of one data item as an attribute for more than one entity provided the attributes have same name and data type and do not belong to a primary key. If deselected or when the attribute belongs to a primary key, the data item cannot be reused. In this case, if the Unique code check box is selected, a new data item with identical name but different code is created, otherwise a new data item with identical name and code is created.

When you delete an entity or entity attributes, these options determine whether or not the corresponding data items are also deleted, as follows:
- Both – deletes the entity attribute.
- Unique Code only – deletes the entity attribute.
- Allow Reuse only – deletes the entity attribute and the corresponding data item (if it is not used by another entity).
- None – deletes the entity attribute and the corresponding data item.

For information about controlling the naming conventions of your models, see Core Features Guide > Modeling with PowerDesigner > Objects > Naming Conventions.

In this section:
- Assertion Template
- Migration Settings (LDM)

2.1.5.1.1 Assertion Template

The assertion template is a GTL template used to automatically generate sentences from the role names you specify on the Cardinalities tab of relationship property sheets. To review or edit the template, select Tools > Model Options > Assertion Template.

The PowerDesigner Generation Template Language (GTL) is used to generate text from the objects, properties, and relationships defined in the PowerDesigner metamodel and in extensions to it.

The GTL code in the template extracts various properties of the relationship object and the entities it connects to generate the assertion statements. The mandatory property and cardinalities are evaluated in each direction in order to generate the appropriate wording around the entity and role names.

You can edit the assertion template as necessary, to change the wording or to reference other properties. To reference extended attributes or other extensions, you must specify the extension file for the template to use in the Assertion Extension list.

A sample extension file, Relationship Assertion with Plural Entity Names, is provided, which provides support for using plural entity names in assertions. For information about attaching this or any other XEM to your model, see Extending your Modeling Environment.

For detailed information about working with GTL, see Customizing and Extending PowerDesigner > Customizing Generation with GTL.
2.1.5.1.2 Migration Settings (LDM)

To set migration settings, select **Tools** > **Model Options** and select the Migration settings sub-category under **Model Settings**.

These options control the migration of identifiers along relationships:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate attribute properties</td>
<td>Enables the domain, the checks or the rules to be kept when an attribute is migrated.</td>
</tr>
<tr>
<td>Foreign attribute name</td>
<td>Specifies the naming convention for migrated foreign identifiers. You can select one of the default templates from the list or enter your own using the following variables:</td>
</tr>
<tr>
<td></td>
<td>- %PARENT% - Name/Code of the parent entity</td>
</tr>
<tr>
<td></td>
<td>- %ATTRIBUTE% - Name/Code of the parent attribute</td>
</tr>
<tr>
<td></td>
<td>- %IDENTIFIER% - Name/Code of the identifier constraint attached to the relationship</td>
</tr>
<tr>
<td></td>
<td>- %RELATIONSHIP% - Name/Code of the relationship</td>
</tr>
<tr>
<td></td>
<td>- %PARENTROLE% - Role of the entity that generated the parent entity, this variable proceeds from the conceptual environment.</td>
</tr>
<tr>
<td></td>
<td>If no role is defined on the relationship, %PARENTROLE% takes the content of %PARENT% to avoid generating an attribute with no name.</td>
</tr>
<tr>
<td></td>
<td>The following example checks the %PARENTROLE% value: if it is equal to the parent name (which is the replacement value) then the template %PARENT%, %ATTRIBUTE% will be used, otherwise template %PARENTROLE% will be used because the user has entered a parent role for the relationship:</td>
</tr>
<tr>
<td></td>
<td>Note that customized naming templates reappear in the generation dialog box the next time you open it, but are not saved to the list of predefined templates.</td>
</tr>
<tr>
<td>Use template</td>
<td>Controls when the primary identifier attribute name template will be used. You can choose between:</td>
</tr>
<tr>
<td></td>
<td>- Always use template.</td>
</tr>
<tr>
<td></td>
<td>- Only use template in case of conflict.</td>
</tr>
</tbody>
</table>

2.1.5.2 Setting PDM Model Options

You can set PDM model options by selecting **Tools** > **Model Options** or right-clicking the diagram background and selecting **Model Options**.

You can set the following options on the **Model Settings** page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code case sensitive</td>
<td>Specifies that the names and codes for all objects are case sensitive, allowing you to have two objects with identical names or codes but different cases in the same model. If you change case sensitivity during the design process, we recommend that you check your model to verify that your model does not contain any duplicate objects.</td>
</tr>
<tr>
<td>Enable links to requirements</td>
<td>Displays a Requirements tab in the property sheet of every object in the model, which allows you to attach requirements to objects (see Requirements Modeling).</td>
</tr>
<tr>
<td>External Shortcut Properties</td>
<td>Specifies the properties that are stored for external shortcuts to objects in other models for display in property sheets and on symbols. By default, all properties appear, but you can select to display only Name/Code to reduce the size of your model.</td>
</tr>
<tr>
<td>Note</td>
<td>This option only controls properties of external shortcuts to models of the same type (PDM to PDM, EAM to EAM, etc). External shortcuts to objects in other types of model can show only the basic shortcut properties.</td>
</tr>
<tr>
<td>Notation</td>
<td>Specifies the use of one of the following notation types for the model. You can choose between:</td>
</tr>
<tr>
<td></td>
<td>- Relational - Arrow pointing to primary key. This option is the default, and is used in this manual.</td>
</tr>
<tr>
<td></td>
<td>- CODASYL - Arrow pointing to foreign key.</td>
</tr>
<tr>
<td></td>
<td>- Conceptual - Cardinality displayed in IE format (crow's feet).</td>
</tr>
<tr>
<td></td>
<td>- IDEF1X - Cardinality and mandatory status displayed on reference, primary columns in separate containers and dependent tables with rounded rectangles.</td>
</tr>
</tbody>
</table>

For information about controlling the naming conventions of your models, see Core Features Guide > Modeling with PowerDesigner > Objects > Naming Conventions.

In this section:
2.1.5.2.1 Column and Domain Model Options

To set model options for columns and domains, select Tools Model Options and select the Column & Domain sub-category in the left-hand Category pane.

You can set the following options on this tab:

<table>
<thead>
<tr>
<th>Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enforce non-divergence</td>
<td>Specifies that columns attached to a domain must remain synchronized with the selected properties (see Controlling Non-Divergence from a Domain).</td>
</tr>
<tr>
<td>Default data type</td>
<td>Specifies the default data type to be applied to columns and domains if none is selected for them.</td>
</tr>
<tr>
<td>Column / Domain: Mandatory by default</td>
<td>Specifies that columns or domains are created, by default, as mandatory and that they may must, therefore contain non-null values.</td>
</tr>
</tbody>
</table>

2.1.5.2.2 Reference Model Options

To set model options for references, select Tools Model Options, and select the Reference sub-category in the left-hand Category pane.

You can set the following options on this tab:

<table>
<thead>
<tr>
<th>Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique code</td>
<td>Requires that references have unique codes. If this option is not selected then different references can have the same code (except when two references share the same child table).</td>
</tr>
<tr>
<td>Auto-reuse / Auto-migrate columns</td>
<td>Enable the reuse of columns in child tables as foreign key columns and the migration of primary key columns to child tables during the creation of references (see Automatic Reuse and Migration of Columns).</td>
</tr>
<tr>
<td>Mandatory parent</td>
<td>Specifies that the relationship between child and parent tables is, by default, mandatory, i.e., each foreign key value in the child table must have a corresponding key value, in the parent table.</td>
</tr>
<tr>
<td>Change parent allowed</td>
<td>Specifies that a foreign key value can change to select another value in the referenced key in the parent table.</td>
</tr>
<tr>
<td>Check on commit</td>
<td>Specifies that referential integrity is checked only on commit, rather than immediately after row insertion. This feature can be useful when working with circular dependencies. Not available with all DBMSs.</td>
</tr>
<tr>
<td>Propagate column properties</td>
<td>Propagates changes made to the name, code, stereotype, or data type of a parent table column to the corresponding child column.</td>
</tr>
<tr>
<td>Default link on creation</td>
<td>Specifies how reference joins are created (see Automatic Reuse and Migration of Columns).</td>
</tr>
</tbody>
</table>
| Default implementation        | Specifies how referential integrity is implemented in the reference. You can select either:  
|                               | • Declarative – referential integrity is defined by constraint in foreign declarations  
|                               | • Trigger – referential integrity is implemented by triggers  
|                               | For more information on referential integrity, see Reference Properties. |
| Default Constraints: Update   | Controls how updating a key value in the parent table will, by default, affect the foreign key value in the child table. Depending on your DBMS, you can choose from some or all of the following settings:  
|                               | • None – no effect  
|                               | • Restrict – cannot update parent value if one or more matching child values exist (no effect)  
|                               | • Cascade - update matching child values  
|                               | • Set null - set matching child values to NULL  
|                               | • Set default – set matching child values to default value |
| Default Constraints: Delete   | Controls how deleting a key value in the parent table will, by default, affect the foreign key value in the child table. Depending on your DBMS, you can choose from some or all of the following settings:  
|                               | • None – no effect  
|                               | • Restrict – cannot delete parent value if one or more matching child values exist (no effect)  
|                               | • Cascade - delete matching child values  
|                               | • Set null - set matching child values to NULL  
|                               | • Set default – set matching child values to default value |

2.1.5.2.3 Other Object Model Options

To set model options for tables and views, indexes, join indexes, procedures, sequences, triggers, and database packages select Tools Model Options, and select the appropriate sub-category under Model Settings.

You can set the following options for these objects:

<table>
<thead>
<tr>
<th>Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default owner</td>
<td>Specifies a default owner for the specified object from the list of users (see Creating a User, Group, or Role). To create a user, click on the ellipsis button to open the List of Users, and click the Add a Row tool. If the user specified is subsequently deleted, this option (and the ownership of all associated objects) will be reset to none.</td>
</tr>
<tr>
<td>Ignore identifying owner</td>
<td>Specifies that the owner of a table or view is ignored for identification purposes. Since, by default, both the name/code and the owner are considered during a uniqueness check, this option enables you to enforce distinct names for these objects. For example, if a model contains a table called &quot;Table_1&quot;, which belongs to User_1, and another table, also called &quot;Table_1&quot;, which belongs to User_2, it will, by default, pass a uniqueness check because of the different owners.</td>
</tr>
</tbody>
</table>
Setting Data Model Display Preferences

PowerDesigner display preferences allow you to customize the format of object symbols, and the information that is displayed on them. To set data model display preferences, select `Tools > Display Preferences` or right-click the diagram background and select `Display Preferences`.

In the `Display Preferences` dialog, select the type of object in the list in the left pane, and modify its appearance in the right pane.

You can control what properties it will display on the `Content` tab, and how it will look on the `Format` tab. If the properties that you want to display are not available for selection on the `Content` tab, click the `Advanced` button and add them using the `Customize Content` dialog.

For detailed information about controlling the appearance and content of object symbols, see Core Features Guide > Modeling with PowerDesigner > Diagrams, Matrices, and Symbols > Display Preferences.

Viewing and Editing the DBMS Definition File

Each PDM is linked to a definition file that extends the standard PowerDesigner metamodel to provide objects, properties, data types, and generation parameters and templates specific to the language being modeled. Definition files and other resource files are XML files located in the `Resource Files` directory inside your installation directory, and can be opened and edited in the PowerDesigner Resource Editor.

**Caution**

The resource files provided with PowerDesigner inside the Program Files folder cannot be modified directly. To create a copy for editing, use the `New` tool on the resource file list, and save it in another location. To include resource files from different locations for use in your models, use the `Path` tool on the resource file list.

To open your model's definition file and review its extensions, select `Database > Edit Current DBMS`.

For detailed information about the format of these files, see Customizing and Extending PowerDesigner > DBMS Definition Files.

**Note**

Some resource files are delivered with "Not Certified" in their names. We will perform all possible validation checks, but we do not maintain specific environments to fully certify these resource files. We will support them by accepting bug reports and providing fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. You are invited to assist us by testing fixes and reporting any continuing inconsistencies.

Changing the DBMS

You can change the DBMS being modeled in your PDM at any time.

**Context**

If you change the DBMS being modeled, the model will be altered to conform with the new DBMS as follows:

- All data types specified in your model will be converted to their equivalents in the new DBMS.
- Any objects not supported by the new DBMS will be deleted.
- Certain objects, whose behavior is heavily DBMS-dependent may lose their values.

**Note**

You may be required to change the DBMS if you open a model and the associated definition file is unavailable.
2. Select a DBMS from the list.

By default, PowerDesigner creates a link in the model to the specified file. To copy the contents of the resource and save it in your model file, click the Embed Resource in Model button to the right of this field. Embedding a file in this way enables you to make changes specific to your model without affecting any other models that reference the shared resource.

3. [optional] Click the DBMS Preserve Options tab, and select the check boxes for the objects and options that you want to preserve:
   - Triggers and stored procedures – triggers are always rebuilt when you change DBMS.
   - Physical options - if the syntax of an option is incompatible with the new DBMS, the values will be lost, even if you have selected to preserve the physical option. For example, the physical option used by ASA is not supported by Oracle and any values associated with that option will be lost.
   - DBMS-specific objects - databases, storages, tablespaces, abstract data types, sequences.
   - Extended attributes - which are defined for a particular DBMS.

4. Click OK.

A message box opens to tell you that the DBMS has been changed.

5. Click OK to return to the model.

### 2.1.5.5 Extending your Modeling Environment

You can customize and extend PowerDesigner metaclasses, parameters, and file generation with extensions, which can be stored as part of your model or in separate extension files (*.xem) for reuse with other models.

To access extensions defined in a *.xem file, simply attach the file to your model. You can do this when creating a new model by clicking the Select Extensions button at the bottom of the New Model dialog, or at any time by selecting File > Extensions to open the List of Extensions and clicking the Attach an Extension tool.

In each case, you arrive at the Select Extensions dialog, which lists the extensions available, sorted on sub-tabs appropriate to the type of model you are working with:

---

1. **Note**

If you are changing DBMS within a database family, for example between SAP® Adaptive Server® Enterprise 12.5 and 15, all preserve options available are selected by default. The database objects not supported by the old and new DBMSs are disabled.
To quickly add a property or collection to an object from its property sheet, click the menu button in the bottom-left corner (or press F11) and select New Attribute or New List of Associated Objects. For more information, see Core Features Guide > Modeling with PowerDesigner > Objects > Extending Objects.

To create a new extension file and define extensions in the Resource Editor, select Model > Extensions, click Add a Row, and then click Properties. For detailed information about working with extensions, see Customizing and Extending PowerDesigner > Extension Files.

2.1.5.6 Traceability Links

Traceability links provide a flexible means for creating a connection between any object in any type of model and any other object in the same model or any other model of any type. Traceability links have no formal semantic meaning, but can be followed when performing an impact analysis or otherwise navigating through the model structure.

To create a traceability link between objects in the same diagram, select the Link/Traceability Link tool in the Toolbox. Click inside the symbol of the object that is dependent and, while continuing to hold down the mouse button, drag the cursor and release it on the symbol of the object on which it depends.

In the following example, the Work entity is shown as being dependent on School through a traceability link:

To create a traceability link to any object in any model that is open in the Workspace, open the property sheet of the dependent object, click its Traceability Links tab, and click the Add Objects tool. Use the Model list to select a different model, select the object to point to and click OK to create the link and return to the dependent object’s Traceability Links tab.

You can optionally specify a type for any traceability link in the Link Type column. Click the Types and Grouping tool to perform various actions on this tab:

- To make a link type available for selection in the Link Type column, click the Types and Grouping tool and select New Link Type. Enter a Name for the link type and, optionally, a Comment to explain its purpose, and then click OK.

  - To control the display and grouping of links, click the Types and Grouping tool and select:
    - Group by Object Type - to display links to different types of objects on separate sub-tabs. To add a link to a new object type, click the plus sign on the leftmost sub-tab.
    - Group by Link Type - to display different link types on separate sub-tabs. To add a new link type, click the plus sign on the leftmost sub-tab.

To see all of the objects that point to an object via traceability links, open its property sheet, click its Dependencies tab, and click the Incoming Traceability Links sub-tab.

2.2 Conceptual and Logical Diagrams

The data models in this chapter allow you to model the semantic and logical structure of your system.

PowerDesigner provides you with a highly flexible environment in which to model your data systems. You can begin with either a CDM (see Conceptual Diagrams) or an LDM (see Logical Diagrams) to analyze your system and then generate a PDM (see the Physical Diagrams) to work out the details of your implementation. Full support for database reverse-engineering allows you to take existing data structures and analyze them at any level of abstraction.

For more information about intermodel generation, see Generating Other Models from a Data Model.

In this section:
- Supported CDM/LDM Notations
- Conceptual Diagrams
- Logical Diagrams
- Data Items (CDM)
- Entities (CDM/LDM)
- Attributes (CDM/LDM)
- Identifiers (CDM/LDM)
- Relationships (CDM/LDM)
- Associations and Association Links (CDM)
2.2.1 Supported CDM/LDM Notations

PowerDesigner supports the most popular data modeling notations in the CDM and LDM. You can choose your notation by clicking **Tools > Model Options** and selecting it in the **Notation** list.

**Entity/relationship Notation**

In the Entity/relationship notation, entities are represented as rectangles and divided in three compartments: name, attributes, and identifiers.

The termination points of relationships indicate the cardinality as follows:

- **One to One**: 1-1
- **Many to One**: n-1
- **Many to Many**: n-n
- **Mandatory**: M
- **Dependent**: D

(Note that the Merise notation uses associations instead of relationships):

Inheritance symbols indicate if they are complete and if they have mutually exclusive children:

<table>
<thead>
<tr>
<th>Complete</th>
<th>Mutually exclusive</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**IDEF1X Notation**

In the IDEF1X notation, entity names are displayed outside the symbol, and dependent entities are drawn with round corners.
Relationship symbols indicate the cardinality as follows:

- One to One non-mandatory: 
- One to One mandatory: 
- One to Many non-mandatory: 
- One to Many mandatory: 
- Many to Many non-mandatory: 
- Many to Many mandatory: 
- Depends:

Inheritance symbols indicate if the inheritance is complete:

<table>
<thead>
<tr>
<th>Complete</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>☑️</td>
</tr>
<tr>
<td>No</td>
<td>☐️</td>
</tr>
</tbody>
</table>

**Barker Notation**

In the Barker notation, entities are drawn with round corners, and inheritances are displayed by placing children inside the parent entity.

Only attributes are listed and a symbol specifies whether each attribute is a key, a mandatory or an optional attribute as follows:

- ¶ Primary
- * Mandatory
- ○ Optional
2.2.2 Conceptual Diagrams

A conceptual data diagram provides a graphical view of the conceptual structure of an information system, and helps you identify the principal entities to be represented, their attributes, and the relationships between them.

Note
To create a conceptual diagram in an existing CDM, right-click the model in the Browser and select New > Conceptual Diagram. To create a new model, select File > New Model, choose Conceptual Data Model as the model type and Conceptual Diagram as the first diagram, and then click OK.

In the following conceptual diagram, the Teacher and Student entities inherit attributes from the Person parent entity. The two child entities are linked with a one-to-many relationship (a teacher has several students but each student has only one main teacher).

In addition:
- a teacher can teach several subjects and a subject can be taught by several teachers (many-to-many).
- a teacher can teach several lessons and a lesson is taught by only one teacher (one-to-many).
- a student attends multiple lessons and a lesson is followed by multiple students (many-to-many).
- a student studies multiple subjects and a subject can be studied by multiple students (many-to-many).

2.2.2.1 Conceptual Diagram Objects

PowerDesigner supports all the objects necessary to build conceptual diagrams.

<table>
<thead>
<tr>
<th>Object</th>
<th>Tool</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>[none]</td>
<td>[none]</td>
<td>Set of values for which a data item is valid. See Domains (CDDLMDPDM).</td>
</tr>
<tr>
<td>Data Item</td>
<td>[none]</td>
<td>[none]</td>
<td>Elementary piece of information. See Data Items (CDM).</td>
</tr>
<tr>
<td>Entity</td>
<td></td>
<td></td>
<td>Person, place, thing, or concept that is of interest to the enterprise.</td>
</tr>
</tbody>
</table>
2.2.2.2 Example: Building a Data Dictionary in a CDM

PowerDesigner supports the definition and maintenance of an enterprise data dictionary in a CDM. A data dictionary defines the data items, entities and attributes of the enterprise, and by managing it in a CDM and linking it (through generation or through the mapping editor) with your data and other models, you can ensure consistency of use and benefit from sophisticated impact analysis and "where used" reporting.

Context

Data dictionaries ensure consistency of use by providing a single authoritative definition for all common data elements used across the enterprise. They are used to standardize data content, context, and definitions and to achieve consistency and reusability while increasing the quality of the data used throughout the organization. By clearly defining and delineating the objects that comprise the enterprise and its systems, they enable:

- easier integration and communication between systems
- more standardized messaging between applications
- higher quality business intelligence and analytics
- better understanding between all subject matter experts
- more agile response to change and more complete impact analysis

A data dictionary defined in a PowerDesigner CDM provides:

- a unique list of entities and data items
- data items as descriptions of data artifacts
- entities connected to data items through attributes
- entity-to-entity relationships
- traceability from the data dictionary to logical and physical data models and other models
- impact analysis and "where used" reporting capabilities

Procedure

1. Select File → New to open the New Model dialog, select to create a new CDM and give it an appropriate name, for example, Enterprise Data Dictionary.
2. Select Model Data Items to open the List of Data Items and enter some concepts that you want to define. Each data item is an elementary piece of information, which represents a fact or a definition defined using business terms. Some examples of data items are Customer Name, Order Description, and Zip Code. Data items exist independently of any containing entity, which is important in a data dictionary as you are seeking to define atomic business data and terms, independent of how they may ultimately be used by entities. For more information about defining data items, see Data Items (CDM).

3. Select Model Entities to open the List of Entities and enter some of the entities that you want to define. Entities represent more complex business structures composed of one or more attributes (which are associated with data items). Some examples of entities are Customer, Product, Order. When you create entities, a symbol for each one will be created in the CDM diagram. While such a graphical representation is not strictly necessary for the purposes of creating a data dictionary, you may find this diagram useful to help you visualize the content and structure of business concepts.

4. Double-click an entity in the Browser or diagram to open its property sheet, and click the Attributes tab. Entity attributes provide the link between an entity and a data item.
Create a new attribute by reusing an existing data item by clicking the *Reuse Data Item* tool and selecting the data item that you want to reuse. By default, PowerDesigner allows you to reuse a data item for more than one entity attribute so that, for example, you can define a *Zip Code* data item once, and reuse it in whatever entities contain addresses. If you then update the data item, your changes will simultaneously cascade down to all the entity attributes that use it. This is a great way to enforce consistency throughout the data dictionary model.

You can also create data items in this list by clicking the *Add a Row* or *Insert a Row* tool to add a new line in the list and entering an appropriate name. PowerDesigner will create the attribute and an associated data item. You can also create a new attribute by creating a copy of an existing data item. Click the *Add Data Item* tool and select the data item that you want to copy. Any changes made through this attribute or directly to this copy of the data item will only affect this attribute and no others.

5. Double-click one of your entity symbols (or its Browser entry) to open its property sheet so that you can provide a precise definition for it. The *Comment* field on the *General* tab is intended for a simple, short description in unformatted text, while the *Description* field on the *Notes* tab allows for fully formatted text, and is intended to contain the complete, detailed definition from the business:

6. [optional] Select the *Relationship* tool in the pallet and create relationships between the entities in your data dictionary. Click and hold in one entity, then drag the cursor to a second entity and release the mouse button. Draw other relationships as necessary and then right-click anywhere in the diagram to drop the tool. Double-click a relationship line to open its property sheet and specify properties such as role name and cardinality.
7. The purpose of a data dictionary is to map the concepts that it defines to the concepts, logical entities, and physical tables that make up the implementation of these ideas in the enterprise. PowerDesigner provides two complementary methods for connecting the data dictionary with your other models:

- **Generation**: If you have no existing PDM, you can generate a new model from your data dictionary. Click [Tools] > [Generate Physical Data Model] to open the Generate dialog, select the [Generate new...](#) option, and specify a name for the model to generate. Click the [Selection] tab and select the concepts you want to generate to the new model, and then click [OK].

You can review the links created between the data dictionary and your other models in the Generation Links Viewer ([select](#) [Tools] > [Generation Links] > [Derived Models]).

You can regenerate whenever necessary to propagate updates or additions in the data dictionary to your other models. The Merge Models dialog (see Core Features Guide > Modeling with PowerDesigner > Comparing and Merging Models) will appear, which lets you review and approve (or reject) the changes that will be propagated from the data dictionary to the model.
For detailed information about generating models, see Generating Other Models from a Data Model.

- Mapping Editor - If you have an existing PDM or other model, it may be more appropriate to map your data dictionary concepts to your PDM objects using the Mapping Editor, which provides a finer degree of control and a simple drag and drop interface.

Open the model containing the objects you want to link with your data dictionary and select Tools > Mapping Editor. In the Data Source field, select Conceptual Model in the Model Type list, and click Next. Select your data dictionary CDM and click Next. Select the Create default mapping option to instruct PowerDesigner to auto-create mappings where possible based on shared names, and click Finish to open your model and the data dictionary in the Mapping Editor:

You can create additional mappings as necessary by dragging and dropping entities and attributes from the data dictionary onto objects in the target model. Note that mappings created in this way will not automatically propagate changes.

For detailed information about using the Mapping Editor, see Core Features Guide > Linking and Synchronizing Models > Object Mappings.

8. Once the data dictionary is established and linked to the other models used in the enterprise to define the information architecture, you will need to manage changes to it. New concepts will be added and existing elements updated due to refinements in understanding the business or changes to business operations. Some elements may also be removed (though this will probably be rare). Maintaining your data dictionary in a PowerDesigner CDM enables you to leverage sophisticated impact analysis tools to help you understand the time, cost and risk associated with proposed changes.

To launch an impact analysis, select one or more objects in a diagram or the Browser and select Tools > Impact and Lineage Analysis:

![Impact and Lineage Analysis](image)
You can edit the rule sets used to control the analysis and manually adjust the tree view by right-clicking items. Once the analysis view contains the level of detail you want, click the Generate Diagram button to create an impact analysis diagram. This diagram, which can be saved and compared to other impact analysis snapshots, shows the connections that link your dictionary concepts through intermediate objects and models to the physical objects that implement them, providing a graphical "where used" report:

The diagram helps you plan the implementation of a change, as everything defined in the diagram will require further assessment to ensure the change does not invalidate any specific work we have done at the implementation level.

For detailed information about working with impact analysis, see Core Features Guide > Linking and Synchronizing Models > Impact and Lineage Analysis.

9. Share your data dictionary with your modeling team and ensure that the latest version is always available to them, by checking it into your PowerDesigner repository library as a reference model (see Core Features Guide > Administering PowerDesigner > Deploying an Enterprise Glossary and Library).

10. Share your data dictionary with other members of your organization through the PowerDesigner Portal (see Core Features Guide > Storing, Sharing and Reporting on Models > Repository Web Clients > The PowerDesigner Portal) or by publishing it to HTML or RTF (see Core Features Guide > Storing, Sharing and Reporting on Models > Reports).

2.2.3 Logical Diagrams

A logical data diagram provides a graphical view of the structure of an information system, and helps you analyze the structure of your data system through entities and relationships, in which primary identifiers migrate along one-to-many relationships to become foreign identifiers, and many-to-many relationships can be replaced by intermediate entities.

**Note**

To create a logical diagram in an existing LDM, right-click the model in the Browser and select New Logical Diagram. To create a new model, select File New Model, choose Logical Data Model as the model type and Logical Diagram as the first diagram, and then click OK.

The following logical diagram represent the same system as that in our CDM example (see Conceptual Diagrams).
Primary identifiers have migrated along one-to-many relationships to become foreign identifiers, and many-to-many relationships are replaced with an intermediary entity linked with one-to-many relationships to the extremities.

In this section:
- Logical Diagram Objects
- Importing a Deprecated PDM Logical Model
- Importing Multiple Interconnected PDM Logical Models

### 2.2.3.1 Logical Diagram Objects

PowerDesigner supports all the objects necessary to build logical diagrams.

<table>
<thead>
<tr>
<th>Object</th>
<th>Tool</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>[none]</td>
<td>[none]</td>
<td>Set of values for which a data item is valid. See Domains (CDM/LDM/PDM).</td>
</tr>
<tr>
<td>Entity</td>
<td></td>
<td></td>
<td>Person, place, thing, or concept that is of interest to the enterprise. See Entities (CDM/LDM).</td>
</tr>
<tr>
<td>Entity Attribute</td>
<td>[none]</td>
<td>[none]</td>
<td>Elementary piece of information attached to an entity. See Attributes (CDM/LDM).</td>
</tr>
<tr>
<td>Identifier</td>
<td>[none]</td>
<td>[none]</td>
<td>One or many entity attributes, whose values uniquely identify each occurrence of the entity. See Identifiers (CDM/LDM).</td>
</tr>
<tr>
<td>Relationship</td>
<td></td>
<td></td>
<td>Named connection or relation between entities (ER modeling methodology). See Relationships (CDM/LDM).</td>
</tr>
<tr>
<td>n-n Relationship</td>
<td></td>
<td></td>
<td>(LDM only) Named cardinality represented with an intermediary entity. See Relationships (CDM/LDM).</td>
</tr>
<tr>
<td>Inheritance</td>
<td></td>
<td></td>
<td>Relationship that defines an entity as a special case of a more general entity. See Inheritances (CDM/LDM).</td>
</tr>
</tbody>
</table>

### 2.2.3.2 Importing a Deprecated PDM Logical Model

If you have previously created a PDM with the logical model DBMS, you will be invited to migrate to an LDM when you open it.

Procedure

1. Select File ➤ Open and browse to the PDM logical model to open.
2. Click Open to display the Import Logical Data Model dialog:
3. Choose one of the following options:
   - Convert the model to a logical data model – Note that only tables, columns, keys and references are preserved
   - Change the DBMS target to “ANSI Level 2” and open it as a PDM
4. Click **OK** to open the model.

**Results**

A PDM with the logical model DBMS that had been generated from a CDM will retain its links to the source CDM when you convert it to an LDM. However, for any PDM generated from the old LDM, you will need to restore the generation links by regenerating the PDM from the new LDM, using the Update existing PDM option (see Core Features Guide > Linking and Synchronizing Models > Generating Models and Model Objects).

### 2.2.3.3 Importing Multiple Interconnected PDM Logical Models

If you have previously created multiple PDMs with the logical model DBMS, and these models are connected by shortcuts and generation or other links, you can convert them en masse to logical data models and preserve their interconnections.

**Procedure**

1. Select **File > Import > Legacy Logical Data Models** to open the Import Logical Data Models dialog:

2. Click **Open**, browse to the legacy PDMs you want to import, select them, and then click **OK** to add them to the list. You can, if necessary, add multiple PDMs from multiple directories by repeating this step.

3. When you have added all the necessary PDMs to the list, click **OK** to import them into interconnected LDMs.

### 2.2.4 Data Items (CDM)

A **data item** is an elementary piece of information, which represents a fact or a definition in an information system, and which may or may not have any eventual existence as a modeled object.

You can attach a data item to an entity (see **Entities (CDM/LDM)**) in order to create an entity attribute (see **Attributes (CDM/LDM)**), which is associated with the data item.

There is no requirement to attach a data item to an entity. It remains defined in the model and can be attached to an entity at any time.

Data items are not generated when you generate an LDM or PDM.

**Example**

In the information system for a publishing company, the last names for authors and customers are both important pieces of business information. The data item **LAST NAME** is created to represent this information. It is attached to the entities **AUTHOR** and **CUSTOMER**, and becomes entity attributes of those entities.

Another piece of information is the date of birth of each author. The data item **BIRTH DATE** is created but, as there is no immediate need for this information in the model, it is not attached to any entity.

In this section:

- Creating a Data Item
- Data Item Properties
- Controlling Uniqueness and Reuse of Data Items
2.2.4.1 Creating a Data Item

You can create a data item from the Browser or Model menu. Data items are automatically created when you create entity attributes.

- Select Model > Data Items to access the List of Data Items, and click the Add a Row tool.
- Right-click the model (or a package) in the Browser, and select New > Data Item.
- Create an entity attribute (see Attributes (CDM/LDM)). A data item will be automatically created.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

2.2.4.2 Data Item Properties

To view or edit a data item's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated; and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Data type/Length/Precision</td>
<td>Specifies the form of data to be stored, such as numeric, alphanumeric, or Boolean, and, where appropriate, the maximum number of characters or numerals that can be stored, and the maximum number of places after the decimal point. Click the ellipsis button to choose from the list of standard data types (see PowerDesigner Standard Data Types).</td>
</tr>
<tr>
<td>Domain</td>
<td>Specifies the domain associated with the object (see Domains (CDM/LDM/PDM)). Use the tools to the right of this field to create or browse to a domain, or to open the property sheet of the selected domain.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- Standard Checks - Specifies constraints to control the range and format of permitted data (see Setting Data Profiling Constraints).
- Additional Checks - Displays an editable SQL statement, initialized with the standard checks, which can be used to generate more complex constraints (see Specifying Advanced Constraints).
- Rules - Lists the business rules associated with the object (see Business Rules (CDM/LDM/PDM)).

2.2.4.3 Controlling Uniqueness and Reuse of Data Items

You can control naming restraints and reuse for data items with CDM model options, by selecting Tools > Model Options.

Context

<table>
<thead>
<tr>
<th>Option</th>
<th>When selected</th>
<th>When cleared</th>
</tr>
</thead>
</table>
| Unique code     | Each data item must have a unique code. If you try to select this option and some existing data items are already sharing a code, the following error will be displayed: Unique Code option could not be selected because two data items have the same code: `<data_item_code>`.
|                 | To be able to select the option, you must first assign unique codes to all data items. | Multiple data items can have the same code, and you differentiate them by the entities that use them. The entities are listed in the Used By column of the list of data items. |
| Allow reuse     | One data item can be an entity attribute for multiple entities.               | Each data item can be an entity attribute for only one entity |

Note

To make an item visible in a list, click the Customize Columns and Filter tool in the list toolbar, select the appropriate check box from the list of filter options that is displayed, and click OK.

For more information about CDM model options, see Setting CDM/LDM Model Options.

2.2.5 Entities (CDM/LDM)

An entity represents an object about which you want to store information. For example, in a model of a major corporation, the entities created may include Employee and Division.

When you generate a PDM from a CDM or LDM, entities are generated as tables.

In this section:

- Creating an Entity
- Entity Properties
- Copying Entities
- Displaying Attributes and Other Information on an Entity Symbol

2.2.5.1 Creating an Entity

You can create an entity from the Toolbox, Browser, or Model menu.

- Use the Entity tool in the Toolbox.
2.2.5.2 Entity Properties

To view or edit an entity’s properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default, the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the estimated number of occurrences in the physical database for the entity (the number of records).</td>
</tr>
<tr>
<td>Generate</td>
<td>Specifies that the entity will generate a table in a PDM. When modeling in the Barker notation (see Supported CDM/LDM Notations), only leaf subtypes can be generated as PDM tables, and so this option is disabled on Barker supertype property sheets.</td>
</tr>
<tr>
<td>Parent Entity</td>
<td>[read-only] Specifies the parent entity. Click the Properties tool at the right of the field to open the parent property sheet.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:
- Attributes - lists the attributes associated with the entity (see Attributes (CDM/LDM)).
- Identifiers - lists the identifiers associated with the entity (see Identifiers (CDM/LDM)).
- Rules - lists the business rules associated with the entity (see Business Rules (CDM/LDM/PDM)).
- Subtypes - [Barker only] lists the subtypes that inherit from the entity.

2.2.5.3 Copying Entities

You can make a copy of an entity within the same model or between models. When you copy an entity, you create a new entity with a new name and code, attributes, and identifiers. Model options control whether you create new data items or reuse the data items that are attached to the original entity.

Procedure

1. Select an entity in the CDM/LDM, and then select Edit Copy (or press Ctrl+C).
2. Select the diagram or model to where you want to copy the entity and select Edit Paste (or press Ctrl+V).

The entity is copied and the new entity is displayed in the Browser and diagram.

**Note**
When copying an entity to the same model, a new entity with a new name and code, attributes, and identifiers is always created, but the creation of new data items is controlled by data item model options (see Setting CDM/LDM Model Options). Select:
- Allow reuse - to attach the original data items to the new entity attributes. If this option is not selected, the original data items will be copied and these copies will be attached to the new entity attributes.
- Unique code - to force all data items to have unique codes (though two or more data items can have the same name). If neither this option nor Allow reuse is selected, then duplicate data items will be created with the same names and codes.

2.2.5.4 Displaying Attributes and Other Information on an Entity Symbol

To set display preferences for entities, select Tools Display Preferences, and select the Entity sub-category in the left-hand Category pane.

**Entity**
By default the following properties can be displayed on entity symbols:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Display description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Specifies whether Attributes are displayed on entity symbols. If selected, you can choose between displaying:</td>
</tr>
<tr>
<td></td>
<td>• All attributes - All attributes:</td>
</tr>
<tr>
<td></td>
<td>• Primary attributes - Only primary identifier attributes:</td>
</tr>
</tbody>
</table>
Entity Attributes

By default the following properties can be displayed for entity attributes:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Display description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Data type for each entity attribute:</td>
</tr>
<tr>
<td>Data types</td>
<td>Displays only the data type, if any:</td>
</tr>
<tr>
<td>Data types and Domain</td>
<td>Displays both data type and domain, if any:</td>
</tr>
<tr>
<td>Domain</td>
<td>Domain for each entity attribute. You can only display domains when the Data type check box is selected.</td>
</tr>
<tr>
<td>Domain</td>
<td>Domain of an attribute in an entity. This display option interacts with the selection for Data types. As a result, there are four display options:</td>
</tr>
<tr>
<td></td>
<td>Data types: Displays only the data type, if any:</td>
</tr>
<tr>
<td></td>
<td>Domain: Displays only the domain, if any:</td>
</tr>
<tr>
<td></td>
<td>Data types and Domain: Displays both data type and domain, if any:</td>
</tr>
<tr>
<td></td>
<td>Data types and Replace by domains: Displays either data type or domain, if any, and domain if both are present:</td>
</tr>
<tr>
<td>Mandatory&lt;br&gt; &lt;M&gt;</td>
<td>&lt;M&gt; indicators are displayed next to each mandatory attribute:</td>
</tr>
</tbody>
</table>
2.2.6 Attributes (CDM/LDM)

In a CDM, attributes are data items attached to an entity, association, or inheritance. In an LDM, there are no data items, and so attributes exist in entities without a conceptual origin.

When you generate a PDM from a CDM or LDM, entity attributes are generated as table columns.

In this section:
- Creating an Attribute
- Attribute Properties
- Deleting Attributes (CDM)

2.2.6.1 Creating an Attribute

You can create an entity attribute from the Attributes tab in the property sheet of an entity, association, or inheritance.

You can use the following tools, available on the Attributes tab:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a Row</td>
<td>Creates a new attribute and associated data item.</td>
</tr>
<tr>
<td>Add Data Item (CDM)/Add Attributes (LDM)</td>
<td>Opens a Selection window listing all the data items/attributes available in the model. Select one or more data items/attributes in the list and then click OK to make them attributes to the object.</td>
</tr>
<tr>
<td>Reuse Data Item (CDM)</td>
<td>Opens a Selection window listing all the data items/attributes available in the model. Select one or more data items/attributes in the list and then click OK to make them attributes to the object.</td>
</tr>
</tbody>
</table>

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

2.2.6.2 Attribute Properties

To view or edit an attribute’s properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Entity/Association/Inheritance</td>
<td>[read-only] Specifies the parent object. Click the tool to the right of the field to open its property sheet.</td>
</tr>
<tr>
<td>Data Item</td>
<td>[CDM only, read-only] Specifies the related data item (see Data Items (CDM)). Click the tool to the right of the field to open its property sheet.</td>
</tr>
<tr>
<td>Inherited from</td>
<td>[LDM only, read-only] Specifies the parent entity from which the attribute is migrated through an inheritance.</td>
</tr>
</tbody>
</table>
The following tabs are also available:

- **Standard Checks** - Specifies constraints to control the range and format of permitted data (see Setting Data Profiling Constraints).
- **Additional Checks** - Displays an editable SQL statement, initialized with the standard checks, which can be used to generate more complex constraints (see Specifying Advanced Constraints).
- **Rules** - Lists the business rules associated with the object (see Business Rules (CDM/LDM)).

### 2.2.6.3 Deleting Attributes (CDM)

When you delete an attribute, model options determine whether or not the corresponding data items are also deleted:

<table>
<thead>
<tr>
<th>Model options selected</th>
<th>Result of deleting an attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Code and Allow Reuse</td>
<td>Does not delete corresponding data item</td>
</tr>
<tr>
<td>Unique Code only</td>
<td>Does not delete corresponding data item</td>
</tr>
<tr>
<td>Allow Reuse only</td>
<td>Deletes corresponding data item if it is not used by another entity</td>
</tr>
<tr>
<td>None</td>
<td>Deletes corresponding data item</td>
</tr>
</tbody>
</table>

### 2.2.7 Identifiers (CDM/LDM)

An **identifier** is one or many entity attributes, whose values uniquely identify each occurrence of the entity. Each entity must have at least one identifier. If an entity has only one identifier, it is designated by default as the primary identifier.

When you generate a PDM from a CDM or LDM, identifiers are generated as primary or alternate keys.

In this section:

- Creating an Identifier
- Identifier Properties

#### 2.2.7.1 Creating an Identifier

You can create an identifier from the property sheet of an entity:

- Open the **Attributes** tab in the property sheet of an entity, select one or more attributes, and click the **Create Identifier** tool. The selected attributes are associated with the identifier and are listed on the attributes tab of its property sheet.
- Open the **Identifiers** tab in the property sheet of an entity, and click the **Add a Row** tool.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

#### 2.2.7.2 Identifier Properties

To view or edit an identifier's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the <strong>Code</strong> field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Entity</td>
<td>Specifies the name of the entity to which the identifier belongs.</td>
</tr>
<tr>
<td>Primary identifier</td>
<td>Specifies that the identifier is a primary identifier.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- Attributes - lists the attributes (see Attributes (CDM/LDM)) associated with the identifier. Click the **Add Attributes** tool to add an attribute.

### 2.2.8 Relationships (CDM/LDM)

A relationship is a link between entities. For example, in a model that manages human resources, the **Member** relationship links the **Employee** and **Team**
entities and expresses that each employee works in a team, and each team has employees.

For example, the employee Martin works in the Marketing team is one occurrence of the Member relationship.

When you generate a PDM from a CDM or LDM, relationships are generated as references.

![Relationships](image)

**Note**

Relationships are used to link entities in the ER, Barker, and IDEF1X methodologies, while Merise uses associations (see Associations and Association Links (CDM)). PowerDesigner lets you use relationships or associations exclusively, or combine the two methodologies in the same model. The following examples use the ER format. For more information about the other notations, see Supported CDM/LDM Notations.

A one-to-many relationship links one instance of the first entity to multiple instances of the second entity. Additional properties can make one or both sides of this relationship mandatory and define identification rules:

<table>
<thead>
<tr>
<th>One-to-many relationship</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division may have zero or more employees. Each employee may belong to zero or one division.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division must have one or more employees. Each employee may belong to zero or one division.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division may have zero or more employees. Each employee must belong to one and only one division.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division must have one or more employees. Each employee must belong to one and only one division.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division may have zero or more employees. Each employee is identified uniquely by division number and employee number.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-many relationship" /></td>
<td>Each division must have one or more employees. Each employee is identified uniquely by division number and employee number.</td>
</tr>
</tbody>
</table>

A one-to-one relationship links one instance of the first entity with one instance of the second entity:

<table>
<thead>
<tr>
<th>One-to-one relationship</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="One-to-one relationship" /></td>
<td>Each team works on zero or one project. Each project is managed by zero or one team.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-one relationship" /></td>
<td>Each team works on one and one project only. Each project is managed by zero or one team.</td>
</tr>
<tr>
<td><img src="image" alt="One-to-one relationship" /></td>
<td>Each team works on zero or one project. Each project is managed by one and one team only.</td>
</tr>
</tbody>
</table>

A many-to-many relationship links multiple instances of the first entity to multiple instances of the second entity. This type of relationship is not permitted, by
default, in the LDM (see Enabling Many-to-many Relationships in an LDM):

<table>
<thead>
<tr>
<th>Many-to-many relationship</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Diagram](83x701 to 242x746)</td>
<td>Each division may have zero or more employees. Each employee may belong to zero or more divisions.</td>
</tr>
<tr>
<td>![Diagram](83x640 to 242x685)</td>
<td>Each division must have one or more employees. Each employee may belong to one or more divisions.</td>
</tr>
<tr>
<td>![Diagram](83x579 to 242x624)</td>
<td>Each division may have zero or more employees. Each employee must belong to one or more divisions.</td>
</tr>
<tr>
<td>![Diagram](83x518 to 242x563)</td>
<td>Each division must have one or more employees. Each employee must belong to one or more divisions.</td>
</tr>
</tbody>
</table>

In this section:
- Creating a Relationship
- Relationship Properties
- Enabling Many-to-many Relationships in an LDM
- Creating a Reflexive Relationship
- Defining a Code Option for Relationships
- Changing a Relationship into an Associative Entity
- Identifier Migration Along Relationships

### 2.2.8.1 Creating a Relationship

You can create a relationship from the Toolbox, Browser, or Model menu.
- Use the Relationship tool in the Toolbox. Click inside the first entity to be linked and, while continuing to hold down the mouse button, drag the cursor to the second entity. Release the mouse button inside the second entity.
- Select Model > Relationships to access the List of Relationships, and click the Add a Row tool.
- Right-click the model (or a package) in the Browser, and select New > Relationship.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

### 2.2.8.2 Relationship Properties

To view or edit a relationship’s properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Entity1</td>
<td>Specifies the two entities linked by the relationship. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Entity2</td>
<td>Specifies the two entities linked by the relationship. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Generate</td>
<td>Specifies that the relationship should be generated as a reference when you generate a PDM.</td>
</tr>
<tr>
<td>Cardinalities</td>
<td>Contains data about cardinality as the number of instances of one entity in relation to another entity.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

**Cardinalities Tab**

The Cardinalities tab allows you to specify the nature of the relationship between the two entities. The following properties are available:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cardinality

Specifies the number of instances (none, one, or many) of an entity in relation to another entity. You can choose from the following values:

- One-to-one (<1..1>) - One instance of entity A can correspond to only one instance of entity B.
- One-to-many (<1..n>) - One instance of entity A can correspond to more than one instance of entity B.
- Many-to-one (<n..1>) - More than one instance of entity A can correspond to the same one instance of entity B.
- Many-to-many (<n..n>) - More than one instance of entity A can correspond to more than one instance of entity B. To use n:n relationships in an LDM, see Enabling Many-to-many Relationships in an LDM.

For information about the termination points of the relationships in each of the supported notations, see Supported CDM/LDM Notations.

Dominant role

[one-to-one relationships only] Specifies one direction of the relationship as dominant. If you define a dominant direction, the one-to-one relationship generates one reference in a PDM, with the dominant entity as the parent table. If you do not define a dominant direction, the one-to-one relationship generates two references.

In the following example, the author is the dominant entity:

In a PDM, this relationship generates a reference with Author as the parent table, and its primary key migrated to the Picture table as a foreign key:

In addition, this tab contains a groupbox for each direction of the relationship, containing the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role name</td>
<td>Text that describes the relationship of EntityA to EntityB, and which is used to generate the assertion statements displayed at the top of this tab. You should use the infinitive phrase that describes the relationship of one entity to the other. For example: Each Order may &lt;contain&gt; one or more line, and Each line must &lt;belong to&gt; one and only one Order. To modify the sentences generated from your role names, edit your model's assertion template (see Assertion Template).</td>
</tr>
<tr>
<td>Dependent</td>
<td>Specifies that the entity is dependent on and partially identified by the other entity. In the following example, the task entity is dependent on the project entity. Each task is a part of a project and each project can contain zero or more tasks:</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Specifies that each instance of the entity requires at least one instance of the other entity. For example, the subcontract relationship is optional from customer to project, but mandatory from project to customer. Each project must have a customer, but each customer does not have to have a project. Implied by dependent</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the maximum and minimum number of instances of EntityA in relation to EntityB (if mandatory, at least 1). You can choose from the following values: 0..1 – Zero to one instances 0..n – Zero to many instances 1..1 – Exactly one instance 1..n – one to many instances</td>
</tr>
</tbody>
</table>

Joins Tab (LDM)

The Joins tab lists the joins defined between parent and child entity attributes. Joins can link primary, alternate, or foreign identifiers, or any user-specified attributes.

On this tab, you can either:

- Select an identifier from the parent entity in the Parent field on which to base the join to autopopulate the list with its associated parent and child attributes. If necessary, you can modify the specified child attributes.
- Specify <None> in the Parent field and specify your own attribute pairs on which to base the join using the following tools:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuse Attributes</td>
<td>- Create a join by matching parent and child attributes that share the same code.</td>
</tr>
<tr>
<td>Migrate Attributes</td>
<td>- First specify attributes in the Parent Attribute column and then click this tool to migrate them to foreign identifier attributes in the child table. If the attributes do not exist in the child table, they are created.</td>
</tr>
<tr>
<td>Cancel Migration</td>
<td>- Remove any attributes migrated to the child table.</td>
</tr>
<tr>
<td>Insert a Row</td>
<td>- Inserts a row before the selected row in the list to specify another attribute to join on.</td>
</tr>
<tr>
<td>Add a Row</td>
<td>- Adds a row at the end of the list to specify another attribute to join on.</td>
</tr>
</tbody>
</table>
2.2.8.3 Enabling Many-to-many Relationships in an LDM

In an LDM, many-to-many relationships are, by default, not permitted and are represented with an intermediary entity. If you allow many-to-many relationships, you can select the many-to-many value in the cardinalities tab.

Procedure

1. Select Tools > Model Options.
2. Select the Allow n-n relationships check box in the Relationship groupbox, and then click OK to return to the model.

Note
When generating an LDM from a CDM, you can authorize the generation of many-to-many relationships by clicking the Configure Model Options button on the General tab of the generation dialog, and selecting the Allow n-n relationships option.

2.2.8.4 Creating a Reflexive Relationship

A reflexive relationship is a relationship between an entity and itself.

Context

In the following example, the reflexive relationship <Supervise> expresses that an employee (Manager) can supervise other employees.

Note
To obtain clean lines with rounded corners when you create a reflexive relationship, select Display Preferences > Format > Relationship and modify the Line Style with the appropriate type from the Corners list.

Procedure

1. Click the Relationship tool in the Toolbox.
2. Click inside the entity symbol and, while continuing to hold down the mouse button, drag the cursor a short distance within the symbol, before releasing the button.

A relationship symbol loops back to the same entity.

Results

Note
In the Dependencies page of the entity, you can see two identical occurrences of the relationship, this is to indicate that the relationship is reflexive and serves as origin and destination for the link.

2.2.8.5 Defining a Code Option for Relationships

You can control naming restraints for relationships so that each relationship must have a unique code.

Context

If you do not select Unique Code, two relationships can have the same code, and you differentiate them by the entities they link.

The following error message is displayed when the option you choose is incompatible with the current CDM:

<table>
<thead>
<tr>
<th>Error message</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Code option could not be selected because at least two relationships have the same code</td>
<td>Change the code of one relationship</td>
</tr>
</tbody>
</table>
Procedure

1. Select [Tools] > [Model Options] to open the Model Options dialog box:
2. Select or clear the Unique Code check box in the Relationship groupbox, and then click OK to return to the model.

2.2.8.6 Changing a Relationship into an Associative Entity

You can transform a relationship between two entities into an associative entity linked by two relationships, and then attach entity attributes to the associative entity that you could not attach to the relationship.

Procedure

1. Right-click a relationship symbol and select [Change to Entity].
   The original relationship is split in two and an associative entity is created between the two new relationships, taking the name and code of the original relationship.
2. Open the property sheet of the associative entity or one of the new relationships to modify their properties as appropriate.

2.2.8.7 Identifier Migration Along Relationships

Migrations are made instantaneously in an LDM or during generation if you generate a PDM from a CDM.

<table>
<thead>
<tr>
<th>Relationship type</th>
<th>Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent one-to-many</td>
<td>Foreign identifiers become attributes of the primary identifier of the child entity.</td>
</tr>
<tr>
<td>Many-to-many</td>
<td>No attributes are migrated.</td>
</tr>
<tr>
<td>Dominant one-to-one</td>
<td>Primary identifier migrate from the dominant attribute.</td>
</tr>
<tr>
<td>Mandatory one-to-many</td>
<td>If the child to parent role is mandatory, migrated attributes are mandatory.</td>
</tr>
</tbody>
</table>

2.2.9 Associations and Association Links (CDM)

In the Merise modeling methodology an association is used to connect several entities that each represents clearly defined objects, but are linked by an event, which may not be so clearly represented by another entity.

Each instance of an association corresponds to an instance of each entity linked to the association.

When you generate a PDM from a CDM, associations are generated as tables or references.

In the following example, three entities VIDEOK7, CLIENT, and STORE contain video cassette, client, and store information. They are linked by an association which represents a video cassette rental (K7RENTAL). The K7RENTAL association also contains the attributes DATE and STAFF_ID, which give the date of the rental, and the identity of the staff member who rented out the video cassette.

When you generate a PDM, K7RENTED is generated as a table with five columns, STORE_ID,K7_ID, CLIENT_ID, DATE, and STAFF_ID.

You can use associations exclusively in your CDM, or use both associations and relationships.
Association Links

An association is connected to an entity by an association link, which symbolizes the role and the cardinality between an association and an entity.

In this section:
- Creating an Association with Links
- Creating an Association Without Links
- Association Properties
- Association Link Properties
- Creating a Reflexive Association
- Defining a Dependent Association
- Changing an Association into an Associative Entity
- Creating an Association Attribute

### 2.2.9.1 Creating an Association with Links

The easiest way to create an association between entities is to use the Association Link tool, which will create the association and the necessary links as well.

**Procedure**

1. Click the [Association Link] tool in the Toolbox.
2. Click inside the first entity and while continuing to hold down the mouse button, drag the cursor to a second entity. Release the mouse button.

   An association symbol is created between the two entities.

### 2.2.9.2 Creating an Association Without Links

You can create an association without links from the Toolbox, Browser, or Model menu.

- Use the [Association] tool in the Toolbox.
- Select [Model] > [Associations] to access the List of Associations, and click the [Add a Row] tool.
- Right-click the model (or a package) in the Browser, and select [New] > [Association].

Once you have created the association, you can link it to the relevant entities by using the Association Link tool.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

### 2.2.9.3 Association Properties

To view or edit an association's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the estimated number of occurrences in the physical database for the association (the number of records).</td>
</tr>
<tr>
<td>Generate</td>
<td>Specifies that the association will generate a table in a PDM.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Specifies the data item attached to an association.</td>
</tr>
<tr>
<td>Rules</td>
<td>Specifies the business rules associated with the association.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

### 2.2.9.4 Association Link Properties

To view or edit an association link's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>Specifies the entity connected by the association link. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Association</td>
<td>Specifies the association connected by the association link.</td>
</tr>
<tr>
<td>Role</td>
<td>Specifies the label indicating the role of the association link.</td>
</tr>
</tbody>
</table>
2.2.9.5 Creating a Reflexive Association

A reflexive association is a relationship between an entity and itself.

Procedure

1. Click the Association Link tool in the Toolbox.
2. Click inside the entity symbol and, while continuing to hold down the mouse button, drag the cursor a short distance within the symbol, before releasing the button.
3. Drag the resulting association symbol away from the entity to make clear its two links to the entity:

In the example above, the reflexive association Manager expresses that an employee (Manager) can manage other employees.

2.2.9.6 Defining a Dependent Association

In a dependent association, one entity is partially identified by another. Each entity must have an identifier. In some cases, however, the attributes of an entity are not sufficient to identify an occurrence of the entity. For these entities, their identifiers incorporate the identifier of another entity with which they have a dependent association.

Context

An entity named Task has two entity attributes, TASK NAME and TASK COST. A task may be performed in many different projects and the task cost will vary with each project.

To identify each occurrence of TASK COST the unique Task entity identifier is the compound of its Task name entity attribute and the Project number identifier from the Project entity.

When you generate a POM, the TASK table contains the PROJECT NUMBER column as a foreign key, which is also a primary key column. The primary key therefore consists of both PROJECT NUMBER and TASK NAME columns.

Note

The same association can not have two identifier association links.

Procedure

1. Double-click an association link symbol to display the association link property sheet.
2. Select the Identifier check box and then click OK to return to the model.

The cardinality of the association link is enclosed in parenthesis to indicate that the association link is an identifier.
### 2.2.9.7 Changing an Association into an Associative Entity

You can transform an association into an associative entity linked by two associations. The associative entity gets the name and code of the association. The two new associations handle cardinality properties.

**Context**

Two entities `PROJECT MANAGER` and `CONTRACTOR` are linked by the association `WORKS ON PROJECT WITH`:

You can represent this association with an associative entity:

The two new associations can be represented as follows:

**Procedure**

Right-click an association symbol, and select Change to Entity from the contextual menu.

An associative entity that is linked to two associations replaces the original association. The associative entity takes the name of the original association.

### 2.2.9.8 Creating an Association Attribute

The tools used for creating association attributes on this tab are the same as those for creating entity attributes.

For more information, see [Creating an Attribute](#).

### 2.2.10 Inheritances (CDM/LDM)

An `inheritance` allows you to define an entity as a special case of a more general entity. The general, or supertype (or parent) entity contains all of the common characteristics, and the subtype (or child) entity contains only the particular characteristics.

In the example below, the `Account` entity represents all the bank accounts in the information system. There are two subtypes: checking accounts and savings accounts.

The inheritance symbol displays the inheritance status:

<table>
<thead>
<tr>
<th>IDEFIX</th>
<th>E/R and Merise</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬜</td>
<td></td>
<td>Standard</td>
</tr>
<tr>
<td>⬜</td>
<td>★</td>
<td>Mutually exclusive inheritance</td>
</tr>
<tr>
<td>⬜</td>
<td>★</td>
<td>Complete inheritance</td>
</tr>
<tr>
<td>⬜</td>
<td>★</td>
<td>Mutually exclusive and complete inheritance</td>
</tr>
</tbody>
</table>

**Note**

There is no separate inheritance object in the Barker notation (see [Supported CDM/LDM Notations](#)), as inheritances are represented by placing one entity symbol on top of another. Barker inheritances are always complete and mutually exclusive, and the supertype lists its subtypes on the `Subtypes` tab (see [Entity Properties](#)). Only leaf subtypes can be generated as PDM tables, and the `Generate` option is disabled on Barker supertype property sheets.
In this section:
- Creating an Inheritance
- Inheritance Properties
- Making Inheritance Links Mutually Exclusive

2.2.10.1 Creating an Inheritance

You can create an inheritance from the Toolbox, Browser, or Model menu.

- Select the Inheritance tool in the diagram Toolbox, click and hold inside the child entity and then drag to the parent entity and release the mouse button to create a link between the two entities with a half-circle symbol in the middle with the arrow pointing to the parent entity:

  ![Inheritance Diagram](image1)

  To add further child entities, click and hold inside the child entity and then drag to the inheritance half circle and release the mouse button:

  ![Inheritance Diagram](image2)

- Select Model > Inheritances to access the List of Inheritances, and click the Add a Row tool. You will be required to specify a parent entity.
- Right-click the model or package in the Browser, and select New > Inheritance. You will be required to specify a parent entity.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

2.2.10.2 Inheritance Properties

To view or edit an inheritance’s properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property/Code/Comment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Parent</td>
<td>Specifies the name of the parent entity. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Mutually exclusive child</td>
<td>Specifies that only one child can exist for one occurrence of the parent entity.</td>
</tr>
<tr>
<td>Complete</td>
<td>Specifies that all instances of the parent entity (surtype) must belong to one of the children (subtypes). For example, entity Person has 2 sub-types Male and Female; each instance of entity Person is either a male or a female.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

Generation Tab

This tab allows you to specify how the inheritance structure will be generated to a PDM, including which attributes will be inherited.

<table>
<thead>
<tr>
<th>Property/Code/Comment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation Mode</td>
<td>Specifies which parts of the inheritance will be generated. You can specify one or both of the following:</td>
</tr>
<tr>
<td></td>
<td>- Generate parent - Generates a table corresponding to the parent entity. If one or more child entities are not generated, the parent will take on their attributes and references.</td>
</tr>
<tr>
<td></td>
<td>- Generate children - Generates a table corresponding to each child entity. The primary key of each child table is the concatenation of the child entity identifier and the parent entity identifier. You must additionally choose between:</td>
</tr>
<tr>
<td></td>
<td>- Inherit all attributes - Each table inherits all the entity attributes of the parent entity</td>
</tr>
<tr>
<td></td>
<td>- Inherit only primary attributes - Each table inherits only the identifier of the parent entity</td>
</tr>
</tbody>
</table>

Note

- For LDM inheritances, primary identifiers of a parent entity always migrate to all child entities, even if the children are not selected for generation, and any changes you make on this tab will have an immediate effect on the inheritance of attributes in the LDM.

Note

- You can control the generation of individual child tables using the Generate option in the property sheet of each child entity (see Entity Properties).
2.2.10.3 Making Inheritance Links Mutually Exclusive

When an inheritance link is mutually exclusive, one occurrence of the parent entity cannot be matched to more than one child entity. This information is for documentation only and has no impact in generating the PDM.

To make an inheritance link mutually exclusive, open the inheritance property sheet and select the Mutually Exclusive Children check box. Then click OK to return to the diagram.

The mutually exclusive inheritance link displays an X on its half-circle symbol.

In the diagram below, the inheritance link is mutually exclusive, meaning that an account is either checking or savings, but never both.

---

2.3 Physical Diagrams

A physical data diagram provides a graphical view of your database structure, and helps you analyze its tables (including their columns, indexes, and triggers), views, and procedures, and the references between them.

**Note**

To create a physical diagram in an existing PDM, right-click the model in the Browser and select New Physical Diagram. To create a new model, select File New Model, choose Physical Data Model as the model type and Physical Diagram as the first diagram, and then click OK.

In the following example, the Employee table is shown in relation to the Team, Division, Material, Task, and Project tables.
2.3.1 Physical Diagram Objects

PowerDesigner supports all the objects necessary to build physical diagrams.

<table>
<thead>
<tr>
<th>Object</th>
<th>Tool</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td></td>
<td>![Table Symbol]</td>
<td>Collection of rows (records) that have associated columns (fields). See Tables (PDM).</td>
</tr>
<tr>
<td>Column</td>
<td>[none]</td>
<td>![Column Symbol]</td>
<td>Data structure that contains an individual data item within a row (record), model equivalent of a database field. See Columns (PDM).</td>
</tr>
<tr>
<td>Primary key</td>
<td>[none]</td>
<td>![Primary Key Symbol]</td>
<td>Column or columns whose values uniquely identify each row in a table, and are designated as the primary identifier of each row in the table. See Primary, Alternate, and Foreign Keys (PDM).</td>
</tr>
<tr>
<td>Alternate key</td>
<td>[none]</td>
<td>![Alternate Key Symbol]</td>
<td>Column or columns whose values uniquely identify each row in a table, and which is not a primary key. See Primary, Alternate, and Foreign Keys (PDM).</td>
</tr>
<tr>
<td>Foreign key</td>
<td>[none]</td>
<td>![Foreign Key Symbol]</td>
<td>Column or columns whose values depend on and migrate from a primary or alternate key in another table. See Primary, Alternate, and Foreign Keys (PDM).</td>
</tr>
<tr>
<td>Index</td>
<td>[none]</td>
<td>![Index Symbol]</td>
<td>Data structure associated with one or more columns in a table, in which the column values are ordered in such a way as to speed up access to data. See Indexes (PDM).</td>
</tr>
<tr>
<td>Default</td>
<td>[none]</td>
<td>![Default Symbol]</td>
<td>[certain DBMSs] A default value for a column. See Defaults (PDM).</td>
</tr>
<tr>
<td>Domain</td>
<td>[none]</td>
<td>![Domain Symbol]</td>
<td>Defines valid values for a column. See Domains (CDMLDM/PDM).</td>
</tr>
<tr>
<td>Sequence</td>
<td>[none]</td>
<td>![Sequence Symbol]</td>
<td>[certain DBMSs] Defines the form of incrementation for a column. See Sequences (PDM).</td>
</tr>
<tr>
<td>Abstract data type</td>
<td>[none]</td>
<td>![Abstract Data Type Symbol]</td>
<td>[certain DBMSs] User-defined data type. See Abstract Data Types (PDM).</td>
</tr>
<tr>
<td>Reference</td>
<td>![Reference Symbol]</td>
<td>![Reference Symbol]</td>
<td>Link between a primary or an alternate key in a parent table, and a foreign key of a child table. Depending on its selected properties, a reference can also link columns that are independent of primary or alternate key columns. See References (PDM).</td>
</tr>
</tbody>
</table>
2.3.2 Tables (PDM)

A table is used to store data in a set of columns. Each record in the table is represented as a row, which is uniquely identified by the values in its primary key column or columns.

Tables are generally defined using the following sub-objects:

- **Columns** - are named properties of a table that describe its characteristics (see Columns (PDM)).
- **Primary Keys** - Uniquely identify rows through the values in the column or columns with which they are associated (see Primary, Alternate, and Foreign Keys (PDM)). Each key can generate a unique index or a unique constraint in a target database.
- **Indexes** - Help improve search times by ordering the values in the column or columns with which they are associated (see Indexes (PDM)).
- **Triggers** - SQL code invoked automatically whenever there is an attempt to modify data in the tables (see Triggers (PDM)).

Tables are linked together by references (see References (PDM)).

In this section:

- Creating a Table
- Table Properties
- Linking a Table to an Abstract Data Type
- Creating an XML Table or View
- Specifying Table Constraints
- Denormalizing Tables and Columns
- PowerBuilder DataWindow Extended Attributes
- Displaying Column, Domain, and Data Type Information on a Table Symbol
- Physical Options (PDM)

2.3.2.1 Creating a Table

You can create a table from the Toolbox, Browser, or Model menu.

- Use the **Table** tool in the Toolbox.
- Select **Model > Tables** to access the list of Tables, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New > Table**.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

2.3.2.2 Table Properties

To view or edit a table's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Owner</td>
<td>Specifies the user who is the owner of the object. This is usually its creator. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Number</td>
<td>Specifies the estimated number of records in the table, which is used to estimate database size. This field is automatically populated during reverse engineering if you select the Statistics option (see Reverse Engineering from a Live Database).</td>
</tr>
</tbody>
</table>
2.3.2.3 Linking a Table to an Abstract Data Type

If your DBMS supports it, PowerDesigner allows you to base tables on abstract data types (ADT), where the table uses the properties of the ADT and the ADT attributes become table columns. To link a table to an ADT, open the table property sheet to the General tab, and select the ADT (of type Object, SQLJ Object, or Structured type) in the Based On field.

For detailed information about working with abstract data types, see Abstract Data Types (PDM).

2.3.2.4 Creating an XML Table or View

If your DBMS supports it, PowerDesigner allows you to create XML tables and views. An XML table does not contain columns, and instead stores an XML document. You must associate the table with a registered XML schema to validate the XML document stored in the table, and can specify a root element for the structure stored in your table.
When you select the XML in the **Type** field, the **Column** tab is removed and the following properties are added to the **General** tab:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Enter the target namespace or name of an XML model (see XML Modeling) or use the [Selected] tool to the right of the field to connect to the database and select a registered schema. The schema must be registered in the database to be used for validating XML documents.</td>
</tr>
<tr>
<td>Element</td>
<td>Allows you to specify a root element in the XML document. You can enter an element name or click the [Selected] tool to the right of the field to select an element from an XML model open in the workspace or from the schema registered in the database. If you select an element from a PowerDesigner XML model, the <strong>Schema</strong> property is set to the XML model target namespace.</td>
</tr>
</tbody>
</table>

### 2.3.2.5 Specifying Table Constraints

The table **Check** tab is initialized with the PowerDesigner **RULES** variable to generate validation rules specified on the **Rules** tab. You can edit the code on this tab by entering an appropriate SQL expression to supplement, modify, or replace these constraints.

You can override the default **Constraint name**. To revert to the default name, click to reset the **User-Defined** button to the right of the field:

For information about business rules, see Business Rules (CDM/LDM/PDM). For information about setting column constraints, see Setting Data Profiling Constraints.

### 2.3.2.6 Denormalizing Tables and Columns

Database normalization consists in eliminating redundancy and inconsistent dependencies between tables. While normalization is generally considered the goal of database design, denormalization, the deliberate duplication of certain data in order to speed data retrieval, may sometimes be more desirable.

PowerDesigner supports denormalization through:

- **Horizontal partitioning** - dividing a table into multiple tables containing the same columns but fewer rows.
- **Vertical partitioning** - dividing a table into multiple tables containing the same number of rows but fewer columns.
- **Table collapsing** - merging tables in order to eliminate the join between them.
- **Column denormalization** - repeating a column in multiple tables in order to avoid creating a join between them.

Horizontal and vertical partitioning involve tradeoffs in terms of performance and complexity. Though they can improve query response time and accelerate data backup and recovery, they require additional joins and unions to retrieve data from multiple tables, more complex queries to determine which table contains the requested data, and additional metadata to describe the partitioned table. Column denormalization can simplify queries but requires more maintenance and storage space as data is duplicated.

When deciding whether to denormalize, you should analyze the data access requirements of the applications in your environment and their actual performance characteristics. Often, good indexing and other solutions may more effectively address performance problems. Denormalization may be appropriate when:

- Critical queries rely upon data from more than one table.
- Many calculations need to be applied to columns before queries can be successfully answered.
- Tables need to be accessed in different ways by different kinds of users simultaneously.
- Certain columns are queried extremely frequently.

In this section:

- Horizontal Partitions
- Vertical Partitions
- Table Collapsings
- Column Denormalization
- Denormalization Object Properties
- Removing Partitionings and Table Collapsings

#### 2.3.2.6.1 Horizontal Partitions

Horizontal partitioning consists in segmenting a table into multiple tables each containing a subset of rows and the same columns in order to optimize data retrieval. You can use any column, including primary keys, as partitioning criteria.

**Procedure**

1. Select **Tools > Denormalization > Horizontal Partitioning** or right-click a table in the diagram and select **Horizontal Partitioning** to open the Horizontal Partitioning Wizard.
2. Select the table to partition, specify whether you want to keep the original table after partitioning, and then click **Next**.
3. Create as many partition tables as necessary using the **Insert** and **Add a Row** tools (specifying an appropriate name for each, which must be unique in the model), and then click **Next**.
4. Click the **Add Columns** tool to select one or more discriminant columns to use as partition criteria (these columns will be excluded from the partitions), and then click **Next**.
5. Specify a name and code for the transformation object that will be created to preserve information about the partitioning, and then click **Finish** to create a table for each partition, taking the name of the partition. All references to the original table are created on each partition table.

In this example, the table **Annual Sales**, which contains a very large amount of data is horizontally partitioned on the **Year** column:

<table>
<thead>
<tr>
<th>Before</th>
<th>After 2010</th>
<th>After 2011</th>
<th>After 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Type</td>
<td>Column</td>
<td>Type</td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td>Amount</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Year</td>
<td>DATE</td>
<td>Amount</td>
<td>INTEGER</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Horizontal partitions created in a PDM generated from another model are preserved when applying changes from the original model. The absence of discriminant columns in the target PDM is respected in the Merge dialog (see Core Features Guide > Modeling with PowerDesigner > Comparing
2.3.2.6.2 Vertical Partitions

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. The partition tables share the same primary key.

Procedure

1. Select > Tools > Denormalization > Vertical Partitioning or right-click a table in the diagram and select Vertical Partitioning to open the Vertical Partitioning Wizard.
2. Select the table to partition, specify whether you want to keep the original table after partitioning, and then click Next.
3. Create as many partition tables as necessary using the Insert and Add a Row tools (specifying an appropriate name for each, which must be unique in the model), and then click Next.
4. Drag columns from under the original table in the Available columns pane, to the appropriate partition table in the Columns distribution pane, (or select source and target tables and use the Add and Remove buttons), and then click Next.
5. Specify a name and code for the transformation object that will be created to preserve information about the partitioning, and then click Finish to create a table for each partition, taking the name of the partition. All references to the original table are created on each partition table.

In this example, the table Customer, is divided into two tables, each of which details one type of information about the customer:

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Customer</td>
</tr>
<tr>
<td>CustomerID INT</td>
<td>CustomerID INT</td>
</tr>
<tr>
<td>FName VARCHAR(40)</td>
<td>FName VARCHAR(40)</td>
</tr>
<tr>
<td>LName VARCHAR(40)</td>
<td>LName VARCHAR(40)</td>
</tr>
<tr>
<td>Contact VARCHAR(40)</td>
<td>Contact VARCHAR(40)</td>
</tr>
<tr>
<td>Address VARCHAR(40)</td>
<td>Address VARCHAR(40)</td>
</tr>
<tr>
<td>CreditCardNo INT</td>
<td>CreditCardNo INT</td>
</tr>
<tr>
<td>AccountNo INT</td>
<td>AccountNo INT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note

Vertical partitionings created in a PDM generated from another model are preserved when applying changes from the original model. The columns absent from each partition table in the target PDM are shown but not selected in the Merge dialog (see Core Features Guide > Modeling with PowerDesigner > Comparing and Merging Models). Any changes in the source model are proposed, where appropriate, to each of the partition tables, and you should deselect the change for those partitions to which you do not want to apply it.

2.3.2.6.3 Table Collapsings

Table collapsing consists in merging tables in order to eliminate joins and to improve query performance. You can collapse tables related to each other with a reference or tables with identical primary keys.

Procedure

1. Select > Tools > Denormalization > Table Collapsing or right-click a reference between the tables to collapse and select Table Collapsing to open the Table Collapsing Wizard.
2. Specify a name and code for the table to be created, and then click Next.
3. Click the Add Tables tool to select tables to collapse into the new table, specify whether you want to keep the original tables after collapsing, and then click Next.
4.Specify a name and code for the transformation object that will be created to preserve information about the collapsing, and then click Finish to collapse the selected tables into a single unified table (with graphical synonyms replacing each original table symbol in the diagram to minimize disruption of references).

In this example, the tables Customer and Order are collapsed together to eliminate the join and optimize data retrieval. The result is a single table (with 2 synonym symbols) with the primary key of the child table:

5. [optional] Delete one of more of the synonyms. References will redirect to the remaining symbol.

2.3.2.6.4 Column Denormalization

Column denormalization consists in replicating columns from one table to another to reduce the number of joins needed for frequently called queries. Though it
can provide improved performance, column denormalization requires more maintenance and disk space as the data in the replicated column is stored twice.

Procedure

1. Select **Tools** > **Denormalization** > **Column Denormalization** or right-click the table to which you want to replicate columns and select **Column Denormalization** to open the Column Denormalization Wizard.  
2. Specify the table to which you want to replicate columns, and then click **Next**.  
3. Select one or more columns, and then click **Finish** to replicate them to the selected table.

**Note**  
Replicas are, by default, read-only copies of objects. Any changes made to the original column are automatically propagated to the replica. This synchronization is controlled by a replication object for each replica, a list of which is available by selecting **Model** > **Replications**. To revert a column denormalization, simply delete the duplicated column from the target table property sheet. For detailed information about working with replicas and replications, see Core Features Guide > Linking and Synchronizing Models > Shortcuts and Replicas.

In this example, to obtain the division name on the pay slip of each employee without requiring a link to the **Division** table, the **DivisionName** column is replicated to the **PaySlip** table:

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram Before" /></td>
<td><img src="image2.png" alt="Diagram After" /></td>
</tr>
</tbody>
</table>

### 2.3.2.6.5 Denormalization Object Properties

A denormalization transformation object is automatically created when you partition a table using the Horizontal or Vertical Partitioning Wizard or collapse tables with the Table Collapsing Wizard. To access the property sheet of this object, select **Model** > **Transformations** to open the List of Transformations, select the appropriate denormalization, and then click the **Properties** tool.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Partitioned table</td>
<td>[partitionings only] Specifies the name of the table used to create the table partitions.</td>
</tr>
<tr>
<td>Discriminant Columns</td>
<td>[horizontal partitionings] Specifies the name and code of the columns used as partition criteria.</td>
</tr>
<tr>
<td>Target table</td>
<td>[collapsings] Specifies the name of the table resulting from the collapsing of the tables.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- **Partitions** - [partitionings] Lists the tables associated with the partitioning. You can create or delete partition tables, and edit their properties. If you delete a partition, you are prompted to specify whether you want to delete the corresponding table.
- **Partition Columns** - [vertical partitionings] Displays the distribution of columns between the partition tables. You can drag and drop columns between tables.
- **Source Tables** - [table collapsings] Lists the tables that were collapsed. These tables will no longer exist unless you selected to keep them in the wizard.

### 2.3.2.6.6 Removing Partitionings and Table Collapsings

You can remove partitionings or table collapsings and either keep or remove the associated tables.

Select **Model** > **Transformations** to open the List of Transformations. The following tools are available for removing transformations:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Delete" /></td>
<td>Removes the denormalization but retains any tables created by it.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Cancel" /></td>
<td>Removes the denormalization and any tables created by it. You can recover the original table by regenerating it from the source model.</td>
</tr>
</tbody>
</table>

**Note**  
You cannot move or paste a denormalization object to another model or package.
2.3.2.7 PowerBuilder DataWindow Extended Attributes

When designing tables to be used in a SAP® PowerBuilder® DataWindow, you can manage the extended attributes which PowerBuilder uses to store application-based information, such as label and heading text for columns, validation rules, display formats, and edit styles.

Context

PowerDesigner supports the modeling of this information through an extension file. To enable the PowerBuilder extensions in your model, select Model ➤ Extensions ➤ click the Attach an Extension tool, select the PowerBuilder file (on the General Purpose tab), and click OK to attach it.

When this extension file is attached, additional properties for two PowerBuilder system tables (PBCatTbl for tables and PBCatCol for columns) are available on the PowerBuilder tab of tables and columns:

![Table Properties](image)

To import the PowerBuilder extended attributes contained in your database to your PDM, select Tools ➤ PowerBuilder ➤ Reverse Extended Attributes ➤ click the Connect to a Data Source tool, select a machine or file data source and click Connect. Select the tables you want to reverse-engineer, and click OK.

To update the PowerBuilder extended attribute system tables in your database, select Tools ➤ PowerBuilder ➤ Generate Extended Attributes ➤ click the Connect to a Data Source tool, select a machine or file data source and click Connect. Select the tables you want to generate, and click OK. Reversed extended attributes are compared with the translated default values in the PowerBuilder extension file. If these attributes match, the reversed value is replaced by the default value from the extension file.

2.3.2.8 Displaying Column, Domain, and Data Type Information on a Table Symbol

To set display preferences for tables, select Tools ➤ Display Preferences ➤ and select the Table sub-category in the left-hand Category pane.

Columns

Keys and indexes are represented by indicators in the table symbol. Each key and index indicator is assigned a number. You can use these numbers to keep track of the different groups of alternate keys, foreign keys, and indexes in your model.

By default, the following information about columns can be displayed on table symbols:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Displays</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data types</td>
<td>Data type for each column</td>
<td></td>
</tr>
<tr>
<td>Replace by domains</td>
<td>Domain codes for each column attached to a domain</td>
<td></td>
</tr>
<tr>
<td>Domains</td>
<td>Domain of an attribute in the table. This display option interacts with the selection for Data types. As a result, there are four display options</td>
<td>See the Display Domain and Data Type section below for options and examples.</td>
</tr>
<tr>
<td>Key Indicators</td>
<td>&lt;pk&gt;, &lt;fk&gt;, and &lt;ak&gt; indicators next to primary key</td>
<td></td>
</tr>
</tbody>
</table>
### Display Domain and Data Type

You can display the domain of an attribute in the symbol of a table. There are four display options available:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Displays</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data types</td>
<td>Only the data type, if it exists</td>
<td><img src="SALE" alt="Example" /></td>
</tr>
<tr>
<td>Domains</td>
<td>Only the domain, if it exists</td>
<td><img src="SALE" alt="Example" /></td>
</tr>
<tr>
<td>Data types and Domains</td>
<td>Both data type and domain, if they exist</td>
<td><img src="SALE" alt="Example" /></td>
</tr>
<tr>
<td>Data types and Replace by domains</td>
<td>If domain exists and data type does not exist, then displays domain. If domain does not exist and data type exists, then displays data type.</td>
<td><img src="SALE" alt="Example" /></td>
</tr>
</tbody>
</table>

**Note**

For information about selecting other properties to display, see Core Features Guide > Modeling with PowerDesigner > Diagrams, Matrices, and Symbols > Display Preferences.

### 2.3.2.9 Physical Options (PDM)

Physical options are DBMS-specific parameters that specify how an object is optimized or stored in a database, and are included at the end of the object’s `Create` statement. Physical options are defined in the DBMS definition file, and may be available for tables, columns, indexes, tablespaces, and other objects. You can specify default physical options for all objects of a particular type and for individual objects (overriding the default, if one is specified).

#### Context

There are two different interfaces for specifying physical options for individual objects, both of which are accessible through tabs on the object’s property sheet. Changes made on either of these tabs will be reflected on the other:

- **Physical Options (Common)** – this tab is displayed by default (along with the `Partition` tab, if applicable), and lists the most commonly-used physical options as a standard property sheet tab. Select or enter values for the appropriate options and click **OK**.
- **Physical Options** – this tab is hidden by default, and lists all the available physical options for the object in a tree format. To display this tab, click the Property Sheet Menu button and select > Physical Options (All). Follow the procedure in Defining Default Physical Options, to specify options and set values for them.

Physical options can vary widely by DBMS. For example, in Oracle, you specify the tablespace where the table is stored with the `Tablespace` keyword, while in SAP® SQL Anywhere®, you use `In`. When you change DBMS, the physical options selected are preserved as far as possible. If a specific physical option is not available in the new DBMS, it will be removed.
option was selected, the default value is preserved for the option in the new DBMS. Unselected physical options are reset with the new DBMS default values. For detailed information about the syntax of physical options and how they are specified, see Customizing and Extending PowerDesigner > DBMS Definition Files > Physical Options.

In Oracle, the storage composite physical option is used as a template to define all the storage values in a storage entry to avoid having to set values independently each time you need to re-use them same values in a storage clause. For this reason, the Oracle physical option does not include the storage name (%s).

In this section:
- Defining Default Physical Options

## 2.3.2.9.1 Defining Default Physical Options

You can define default physical options for all the objects of a particular type in the model.

**Procedure**

1. Select Database > Default Physical Options to open the Default Physical Options dialog. There is a tab for each kind of object that supports physical options. The Table tab opens by default. The Syntax sub-tab in the left pane lists the physical options available in the DBMS, and the Items sub-tab in the right pane lists the physical options that have been selected for the object.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Action when clicked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds physical option selected in Syntax tab (left pane) to Items tab (right pane)</td>
</tr>
<tr>
<td>Align</td>
<td>Aligns a selected physical option in the Items tab with the corresponding physical option in the Syntax tab</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes physical option selected in Items tab</td>
</tr>
</tbody>
</table>

2. To add a default option for the object, select it in the Syntax pane and click the Add tool to copy it to the Items pane. To add only a sub-parameter for the option, expand the option in the Syntax pane, select the required parameter and then click the Add tool.

3. To set a default value for a physical option parameter, select it in the Items pane and enter or select the appropriate value in the field below the pane. The entered value will then be displayed against the parameter in the Items list.

4. Repeat the above steps as many times as necessary to specify all your required physical options. By default, these options will be applied to all tables created subsequently in the model. To apply them to existing tables, click the Apply to button to select the tables to which you want to apply the options, and then click OK.

5. Select the other tabs to specify physical options for other object types. (Note that the Apply to button is not available on the Database tab).

6. Click OK to close the dialog and return to your model.

To override the default physical options for a particular object, set the appropriate values on the the object's Physical Options (Common) or Physical Options tab.

You can view the physical options set for an object in its Preview tab.
### 2.3.3 Columns (PDM)

A column is a set of values of a single type in a table. Each row of the table contains one instance of each column. Each table must have at least one column, which must have a name and code and to which you can assign a data type, either directly, or via a domain.

In this section:
- Creating a Column
- Column Properties
- Setting Data Profiling Constraints
- Populating Columns with Test Data
- Creating a Computed Column
- Attaching a Column to a Domain
- Copying or Replicating a Column from Another Table

#### 2.3.3.1 Creating a Column

You can create a column from the property sheet of, or in the Browser under, a table.

- Open the Columns tab in the property sheet of a table, and click the Add a Row or Insert a Row tool.
- Right-click a table in the Browser, and select New Column.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

#### 2.3.3.2 Column Properties

To view or edit a column's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default, the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Table</td>
<td>Specifies the table which contains the column.</td>
</tr>
<tr>
<td>Data type/Length/Precision</td>
<td>Specifies the form of data to be stored, such as numeric, alphanumeric, or Boolean, and, where appropriate, the maximum number of characters or numerals that can be stored, and the maximum number of places after the decimal point. Click the ellipsis button to choose from the list of standard data types (see PowerDesigner Standard Data Types). To review the data types permitted by your DBMS, select Database &gt; Edit Current DBMS, and navigate to PhysDataType. The following variables specify length and precision requirements:</td>
</tr>
<tr>
<td>Domain</td>
<td>Specifies the domain associated with the object (see Domains (CDMD/MPDMD)). Use the tools to the right of this field to create or browse to a domain, or to open the property sheet of the selected domain.</td>
</tr>
</tbody>
</table>
The Detail tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Column fill parameters| The fields in this groupbox model the size and distinctness of data values that you expect to store in the column and are used in conjunction with test data profiles (see Populating Columns with Test Data). You can specify:
  - **Null values** - [Default: 0%] Specifies the percentage of values to leave empty.
  - **Distinct values** - [Default: 100%] Specifies the percentage of values that must be unique.
  - **Average Length** - [read only] Used for estimating the size of the database (see Estimating Database Size). The default value is the maximum length for the data type defined for the column.
  - You can enter values by hand or obtain them from your database by selecting the **Statistics** option in the Reverse Engineering dialog (see Reverse Engineering from a Live Database).
  - To refresh the values in these fields for all a table’s columns at any time, right-click the table symbol or its entry in the Browser and select **Update Statistics**. To update the column statistics for all the tables in a model, select the **Tools** > **Update Statistics** (see Reverse Engineering Database Statistics).
| Profile               | Specifies a test data profile to use to generate test data (see Populating Columns with Test Data). Use the tools to the right of this field to create or browse to a profile, or to open the property sheet of the selected profile.
| Computed Expression   | Specifies an expression used to compute data for the column (see Creating a Computed Column).

The following tabs are also available:
- **Standard Checks** - Specifies constraints to control the range and format of permitted data (see Setting Data Profiling Constraints)
- **Additional Checks** - Displays an editable SQL statement, initialized with the standard checks, which can be used to generate more complex constraints (see Specifying Advanced Constraints).
- **Rules** - Lists the business rules associated with the object (see Business Rules (CDM/LDM/PDM)).

### 2.3.3.3 Setting Data Profiling Constraints

PowerDesigner allows you to define data profiling constraints to control the range and format of data allowed in your database. You can specify constraints on the **Standard Checks** and **Additional Checks** tabs of table columns in your PDM, entity attributes in your CDM or LDM, and domains. You can also specify data quality rules on the **Rules** tab of PDM tables and columns, CDM/LDM entities and attributes, and domains.

The following constraints are available on the **Standard Checks** tab of PDM columns, CDM/LDM entity attributes, and CDM/LDM/PDM domains:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Values                | Specifies the range of acceptable values. You can set a:
  - **Minimum** - The lowest acceptable numeric value
  - **Maximum** - The highest acceptable numeric value
  - **Default** - The value assigned in the absence of an expressly entered value. For the PDM, you can directly enter a default value or select a keyword (defined in the Script\Sql\Keywords\ReservedDefault entry of the DBMS definition file) from the list. Default objects (see **Defaults (PDM)**) are also available for selection if your DBMS supports them.
| Characteristics       | These properties are for documentation purposes only, and will not be generated. You can choose a:
  - **Format** - A number of standard formats are available in the list. You can enter a new format directly in the field or use the tools to the right of the field to create a data format for reuse elsewhere.
  - **Unit** - A standard measure.
  - **No space** - Space characters are not allowed.
  - **Cannot modify** - The value cannot be updated after initialization.
| Character case        | Specifies the acceptable case for the data. You can choose between:
  - **Mixed case** [default]
  - **Uppercase**
  - **Lowercase**
  - **Sentence case**
  - **Title case**
| List of values        | Specifies the various values that are acceptable.
  - When specifying strings in the list of values, single or double quotation marks (depending on the DBMS) will be added around the values in the generated script unless:
    - You surround the value by the appropriate quotation marks.
    - You surround the value by tilde characters.
    - The value is a keyword (such as NULL) defined in the DBMS.
In this section:
- Specifying Constraints Through Business Rules
- Creating Data Formats For Reuse
- Specifying Advanced Constraints

### 2.3.3.3.1 Specifying Constraints Through Business Rules

In addition to the constraints specified on the Standard Checks tab, you can specify business rules of type Validation or Constraint to control your data. Both types of rule contain SQL code to validate your data, and you can attach them to tables and table columns in your PDM, entities and entity attributes in your CDM or LDM, and domains.

**Context**

You can use the following PowerDesigner variables when writing your rule expression:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%COLUMN%</td>
<td>Code of the column to which the business rule applies</td>
</tr>
<tr>
<td>%DOMAIN%</td>
<td>Code of the domain to which the business rule applies</td>
</tr>
<tr>
<td>%TABLE%</td>
<td>Code of the table to which the business rule applies</td>
</tr>
<tr>
<td>%MINMAX%</td>
<td>Minimum and maximum values for the column or domain</td>
</tr>
<tr>
<td>%LISTVAL%</td>
<td>List values for the column or domain</td>
</tr>
<tr>
<td>%RULES%</td>
<td>Server validation rules for the column or domain</td>
</tr>
</tbody>
</table>

To attach a business rule (see Business Rules (CDM/LDM/PDM)) to a table, column, entity, attribute, or domain, open the object's property sheet, select the Rules tab, and click the Add Objects tool.

At generation time, business rules of type validation are concatenated together into a single constraint, while rules of type Constraint will be generated as separate constraints if your DBMS supports them.

### 2.3.3.3.2 Creating Data Formats For Reuse

You can create data formats to reuse in constraints for multiple objects by clicking the New button to the right of the Format field on the Standard Checks tab. Data formats are informational only, and are not generated as constraints.

#### Data Format Properties

To view or edit a data format's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
</tbody>
</table>
| Type | Specifies the type of the format. You can choose between:  
- Date/Time  
- String  
- Regular Expression |
| Expression | Specifies the form of the data to be stored in the column; For example, 9999.99 would represent a four digit number with two decimal places. |
| Keywords | Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas. |
2.3.3.3 Specifying Advanced Constraints

The Additional Checks tab is initialized with PowerDesigner variables to generate the data profiling constraints specified on the Standard Checks tab and the validation rules specified on the Rules tab. You can edit the code on this tab by entering an appropriate SQL expression to supplement, modify, or replace these constraints.

Context

For columns, you can override the default Constraint name. To revert to the default name, click to reset the User-Defined button to the right of the field:

The following variables are inserted by default:
- \%MINMAX% - Minimum and maximum values specified on the Standard Checks tab
- \%LISTVAL% - List of values specified on the Standard Checks tab
- \%CASE% - Character case specified on the Standard Checks tab
- \%RULES% - Constraint and validation rules specified on the Rules tab

2.3.3.4 Populating Columns with Test Data

You can use test data to quickly fill your database with large amounts of data in order to test its performance and estimate its size. You can also use test data as the basis for data profiling. PowerDesigner allows you to create test data profiles, which generate or provide lists of data items and are assigned to columns or domains. You can create test data profiles that contain number, character, or date/time data.

For example, you could create a test data profile called Address that specifies character data appropriate to represent addresses, and then associate that profile with the columns Employee Location, Store Location, and Client Address.

If you associate a test data profile with a domain, its data will be generated to all columns that are attached to the domain. If you specify a data profile as the default for its type, its data will be generated to all columns that are not associated with another profile.

To generate test data with or without test data profiles, see Generating Test Data to a Database

You can create a test data profile in any of the following ways:
- Select Model Test Data Profiles to access the List of Test Data Profiles, and click the Add a Row tool
- Right-click the model (or a package) in the Browser, and select New Test Data Profile

Note

You can import and export test data profiles to reuse them across multiple models by using the commands under the Tools Test Data Profile menu. The *.xpf file format can contain one or more test data profiles.

In this section:
- Test Data Profile Properties
- Assigning Test Data Profiles to Columns

2.3.3.4.1 Test Data Profile Properties

To view or edit a test data profile's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
</tbody>
</table>
If you have selected to automatically generate number data on the General tab, you must define the following properties on the Detail tab:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Specifies whether the data is to be generated randomly or sequentially.</td>
</tr>
<tr>
<td>Range</td>
<td>Specifies the range of numbers to generate from and, if a sequential type is specified, the step value to use when traversing the range.</td>
</tr>
<tr>
<td>Decimal numbers</td>
<td>Specifies that the numbers to be generated are decimal, and the number of digits after the decimal point to generate.</td>
</tr>
</tbody>
</table>

If you have selected to automatically generate character data on the General tab, you must define the following properties on the Detail tab:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid characters</td>
<td>Specifies the characters that can be generated (by default, all alphanumeric characters and spaces), separated by commas. You can specify:</td>
</tr>
<tr>
<td></td>
<td>Single characters or strings of characters - surrounded by double quotes. For example, &quot;a&quot;, &quot;bcd&quot;, &quot;e&quot;.</td>
</tr>
<tr>
<td></td>
<td>Character intervals - in which the boundary characters are surrounded by single quotes and separated by a dash. For example, 'a'-'z', 'A'-'Z'.</td>
</tr>
<tr>
<td></td>
<td>To allow any character, select the All checkbox.</td>
</tr>
<tr>
<td>Invalid characters</td>
<td>Specifies the characters that cannot be generated, using the same syntax as for the valid characters. To disallow accented characters, select the No accents checkbox.</td>
</tr>
<tr>
<td>Mask</td>
<td>Specifies the mask characters used to tell users what kind of character they must enter in a given context. By default the test data profile uses the following mask characters:</td>
</tr>
<tr>
<td></td>
<td>A - Letter</td>
</tr>
<tr>
<td></td>
<td>9 - Number</td>
</tr>
<tr>
<td></td>
<td>? - Any character</td>
</tr>
<tr>
<td>Case</td>
<td>Specifies the case in which to generate the data. If you select Lower or Mixed case, select the First Uppercase checkbox to require that each word begin with a capital letter.</td>
</tr>
<tr>
<td>Length</td>
<td>Specifies the length of character strings to generate. You can specify either an exact required length or a range.</td>
</tr>
</tbody>
</table>

If you have selected to automatically generate date and time data on the General tab, you must define the following properties on the Detail tab:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date range</td>
<td>Specifies the upper and lower limits of the date range within which data can be generated.</td>
</tr>
<tr>
<td>Time range</td>
<td>Specifies the upper and lower limits of the time range within which data can be generated.</td>
</tr>
<tr>
<td>Step</td>
<td>Specifies step values for use when traversing the date and time ranges, if sequential values are generated.</td>
</tr>
<tr>
<td>Values</td>
<td>Specifies whether the values are to be generated randomly or sequentially.</td>
</tr>
</tbody>
</table>

The format in which date and time data is generated can be controlled by DBMS items in the Script/Sql/Format category (see Customizing and Extending PowerDesigner > DBMS Definition Files > Script/Sql Category).

If you have selected to provide list data on the General tab, enter as many value-label pairs as necessary on the Detail tab.

If you have selected to provide data from a database on the General tab, you must define the following properties on the Detail tab:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
If you have selected to provide data from a file on the General tab, you must define the following properties on the Detail tab:

### Assigning Test Data Profiles to Columns

You can associate a test data profile directly to a column or to a domain.

**Context**

**Note**
To assign a test data profile to a domain (see Domains (CDM/LDM/PDM), open the domain property sheet and select the appropriate test data profile in the Profile list on the General tab. A test data profile assigned to a domain will generate test data for all the columns attached to the domain.

**Procedure**

1. Open the property sheet of a column and click the Detail tab.
2. Select the appropriate test data profile.
3. **[optional]** Adjust the following properties in the Column fill parameters group box as appropriate:
   - **Null values** - [Default: 0%] Specifies the percentage of values to leave empty.
   - **Distinct values** - [Default: 100%] Specifies the percentage of values that must be unique. For example, if you set this field to 100% for one column and to 80% for a second column, and then generate the table with 10 rows, all 10 rows in the first column will have different values, while 2 values in the second column will be repeated. This is a maximum value, and can change depending on the referential integrity parameters of primary key columns. Alternately, you can enter a specific value without a percentage sign, to indicate the exact number of column rows that should contain unique entries.
   - **Average Length** - [read only] Used for estimating the size of the database (see Estimating Database Size). The default value is the maximum length for the data type defined for the column.

   **Note**
   - If you use a test data profile with a list generation source to a column with a given percentage of distinct values, PowerDesigner uses the values from the test data profile list. If there are not enough values declared in the list, a warning message is displayed in the Output window to inform you that the distinct value parameter cannot be enforced due to lack of distinct values in the list of values.
   - The **Mandatory** (M) - Specifies that the column must contain a value and sets Null values to 0%.
   - **Unique** (U) - Specifies the column must contain a unique value and sets Null value to 0% and Distinct values to 100%.
   - **Foreign** (F) - The column is a foreign key column and takes the values of the corresponding primary key column in the parent table.

4. Click OK to close the column property sheet and return to the model.

5. **[optional]** Generate your test data (see Generating Test Data to a Database).

### Creating a Computed Column

Computed columns are columns whose content is computed from values in other columns in the table. Computed columns are not supported by all DBMSs.

**Procedure**

1. Open the table property sheet and click the Columns tab.
2. Click the Add a Row tool, and then click the Properties tool to open the property sheet for the new column.
3. On the General tab, select the Computed checkbox, and then click the Detail tab.

Simple computed expressions can be entered directly in the Computed expression field. For more complex expressions, click the Edit tool to the right...
of the field to access the SQL Editor (see Writing SQL Code in PowerDesigner).

In the following example a column must be filled with the total sales of widgets computed by multiplying the number of widgets by the widget price:

![SQL Editor](image)

4. Click OK to return to the column property sheet.

The expression is displayed in the Computed Expression pane.

2.3.3.6 Attaching a Column to a Domain

You can attach a column to a domain, and have the domain specify the data type, check parameters, and business rules for the column. Domains can help with data consistency across columns storing similar types of data.

Procedure

1. Double-click a table to open its property sheet, and click the Columns tab.
2. Select the required column and then click the Properties tool to open its property sheet.
3. Select a domain from the Domain list and then click OK.

For detailed information about working with domains, see Domains (CDM/LDM/PDM).

2.3.3.7 Copying or Replicating a Column from Another Table

You can reuse existing columns from other tables by copying or replicating them using the tools on the table property sheet Columns tab or by drag and drop. If your table already contains a column with the same name or code as the copied column, the copied column is renamed.

Context

Copying a column creates a simple copy that you can modify as you wish. Replicating a column creates a synchronized copy which remains synchronized with any changes made to the original column (see Core Features Guide > Linking and Synchronizing Models > Shortcuts and Replicas).

Procedure

1. Open the property sheet of the table you want to copy or replicate the columns to, and click the Columns tab.
2. Click the Add Columns or Replicate Columns to open a selection box listing the columns attached to all other tables in the model.
2.3.4 Primary, Alternate, and Foreign Keys (PDM)

A key is a column, or a combination of columns, that uniquely identifies a row in a table. Each key can generate a unique index or a unique constraint in a target database. You can create the following types of keys:

- **Primary keys** - Contain one or more columns whose combined values uniquely identify every row in a table. Each table can have only one primary key.
- **Alternate keys** - Contain one or more columns whose combined values uniquely identify every row in a table.
- **Foreign keys** - Contain one or more columns whose values match a primary or alternate key in some other table.

In the following example, the **TITLE** table has a primary, alternate and foreign key:

- The primary key, **TITLE_ID** contains the column **TITLE ISBN**, and uniquely identifies each book in the table.
- The alternate key, **TITLE_NAME**, contains the columns **TITLE NAME** and **TITLE TYPE**, and enforces a constraint that no two titles of the same type can have the same name.
- The foreign key contains the column **PUBLISHER ID** and references the primary key column in the **Publisher** table.

In this section:

- Creating Primary Keys
- Creating Alternate Keys
- Creating Foreign Keys
- Key Properties

### 2.3.4.1 Creating Primary Keys

A primary key is the primary identifier for a table, and is attached to one or more columns whose combined values uniquely identify every row in the table. Every table must have a primary key.

**Procedure**

1. Open the property sheet of the table and click the **Columns** tab, which lists all the columns defined for the table (see **Columns (PDM)**).
2. Select the check box in the **P** column for one or more columns in the list to associate them with the primary key.
3. [optional] Click the **Keys** tab and rename the key or select it and click the **Properties** tool to open its property sheet.
4. Click **OK** to close the property sheet and return to the diagram.

In the following example, **Employee number** is the primary key for the table **Employee**, and each employee must have a unique employee number.
2.3.4.1.1 Rebuilding Primary Keys

Rebuilding primary keys in a physical diagram updates primary keys for tables by creating primary keys for tables that have no key and a single unique index. Rebuilding primary keys is useful when not all of the primary keys could be reverse engineered from a database, or if you did not select the rebuild option for primary keys during reverse engineering.

Procedure

1. Select Tools > Rebuild Objects > Rebuild Primary Keys to open the Rebuild Primary Keys dialog box, which lists all the tables in the current model.

![Primary Key Rebuild dialog box](image)

**Note**
To rebuild the primary keys in a package, select the package from the list at the top of the tab. To rebuild the primary keys in a sub-package, click the Include Sub-Packages tool, and then select a sub-package from the dropdown list.

2. Select the tables containing the primary keys that you want to rebuild and then click OK.

2.3.4.2 Creating Alternate Keys

An alternate key is a key associated with one or more columns whose values uniquely identify every row in the table, but which is not the primary key. For example, where the primary key for a table may be the employee id, the alternate key might combine the first, middle, and last names of the employee. Each alternate key can generate a unique index or a unique constraint in a target database.

Procedure

1. Open the property sheet of a table and select the Columns tab.
2. Select the column or columns to associate with the alternate key and click the Create Key tool.
3. Enter a name for the key. Alternate keys are conventionally named AK_<x>_<ColumnCodes> (for example AK1_CUSNAME).
4. [optional] Modify the default Constraint Name.
5. Click OK to complete the creation of your alternate key and return to the table property sheet.

**Note**
You can also create an alternative key using the Add a Row tool on the table property sheet Keys tab, click the Properties tool to open its property sheet, and select the Columns tab to manually associate columns with the key.

2.3.4.3 Creating Foreign Keys

A foreign key is a primary or alternate key that migrates from another table. Foreign keys are generally migrated automatically when you draw a reference from a child to a parent table.

The columns that are defined in a foreign key can also be user-specified at creation and changed at any time from the Joins tab of the reference property sheet (see References (PDM)). For information about auto-migration of foreign keys, see Automatic Reuse and Migration of Columns.

2.3.4.4 Key Properties

To view or edit a key's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
The following tabs are also available:
- Columns - Lists the columns associated with the key. Use the Add Columns tool to associate additional columns with the key.

2.3.5 Indexes (PDM)

An index is a data structure associated with one or more columns ordered by the column values. Indexes are typically created for columns that you are frequently searched on to improve response times. Most types of index are more effective when applied to columns with high cardinality.

For example, in an Author table, you might create an index for the primary key ID and another for the LastName column, as it is regularly searched on, but you will probably not create an index for the BirthCity column, as it is not often searched on.

### 2.3.5.1 Creating Standard, Key, or Function-Based Indexes

You can create indexes by selecting columns on a table property sheet Columns tab and clicking the Create Index tool.

**Procedure**

1. Open the property sheet of a table and select the Columns tab.
2. Select the column or columns on which to base the index and click the Create Index tool.
3. Enter a name for the index and then click the Columns tab.
4. PowerDesigner supports the creation of the following types of index:
   - **Standard indexes** are associated with one or more columns containing high-cardinality values that are frequently searched on. Use the arrow buttons at the bottom of the list to reorder the columns in order of descending cardinality.
   - **Key indexes** are associated with a primary, foreign, or alternate key and based on the same columns as the key. Select the appropriate key from the Columns definition field above the list to empty the list and replace it with the columns associated with the key.

   **Note**
   Key indexes are conventionally named after the table with a _PK, _FK, or AK suffix (for example, Project_AK).

   - **Function-based indexes** (if supported by the DBMS) are populated with values derived from a function or expression based on one or more columns, and provide an efficient mechanism for evaluating statements that contain functions in their WHERE clauses. Click the Add a Row tool, then click in the Expression column and click the ellipsis button to open the SQL Editor to specify an expression.
5. Select an ascending or descending sort order for each column using the list's Sort column.
6. Click OK to complete the creation of your index and return to the table property sheet.

### 2.3.5.2 Index Properties

- **Name/Code/Comment** Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.

- **Stereotype** Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.

- **Table** Specifies the name of the table where the key is defined.

- **Constraint name** Specifies the name of the key constraint. A primary key constraint is a named check that enforces the uniqueness and the presence of values in a primary key column. PowerDesigner automatically creates a default constraint name for a key, which you can modify. To return to the default click to release the User-Defined button. You can use the following variables:
  - %AK% and %AKNAME% - Code and name of the alternate key.
  - %TABLE%, %PARENT%, %CHILD% - Code of the table, the parent table, and the child table.
  - %REFCODE% and %REFRNAME% - Code and name of the reference.

For a complete list of PDM variables, see Customizing and Extending PowerDesigner > DBMS Definition Files > PDM Variables and Macros.

- **Primary key** Specifies that the key is the primary key of the table. There can be only one primary key in a table, so selecting this key as the primary key will deselect any existing primary key.

- **Cluster** Specifies that the key constraint is a clustered constraint (for those DBMSs that support clustered indexes).

- **Keywords** Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

This section includes:
- Creating Standard, Key, or Function-Based Indexes
- Index Properties
- Rebuilding Indexes
The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Name/Code/Comment     | Identify the object. The name should clearly convey the object's purpose to
                        | non-technical users, while the code, which is used for generating        |
                        | code or scripts, may be abbreviated, and should not normally include spaces. |
                        | You can optionally add a comment to provide more detailed information     |
                        | about the object. By default the code is generated from the name by       |
                        | applying the naming conventions specified in the model options. To        |
                        | decouple name-code synchronization, click to release the button to the   |
                        | right of the Code field.                                                |
| Stereotype            | Extends the semantics of the object. You can enter a stereotype directly   |
                        | in this field, or add stereotypes to the list by specifying them in an    |
                        | extension file.                                                          |
| Owner                 | Specifies the user who is the owner of the object. This is usually its     |
                        | creator. Use the tools to the right of the list to create, browse for, or|
                        | view the properties of the currently selected object.                    |
| Table                 | Specifies the table to which the index belongs.                            |
| Type                  | Specifies the type of index (if supported by your DBMS). For information    |
                        | about SAP® IQ index types, see Indexes (IQ).                            |
| Unique                | Specifies that the index cannot contain duplicate values.                 |
| Cluster               | Specifies that the index is a clustered index. A table cannot have more   |
                        | than one clustered index.                                                |
| Keywords              | Provide a way of loosely grouping objects through tagging. To enter        |
                        | multiple keywords, separate them with commas.                           |

The following tabs are also available:
- **Columns**: Lists the columns with which the index is associated (see Creating Standard, Key, or Function-Based Indexes). Use the following tools to specify columns:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Columns definition - Select the appropriate key to empty the list and replace</td>
</tr>
<tr>
<td></td>
<td>it with the columns associated with the key.</td>
</tr>
<tr>
<td>Add Columns</td>
<td>Choose columns from the table to add to the list.</td>
</tr>
<tr>
<td>Add / Insert a Row</td>
<td>[if your DBMS supports function-based indexes] - Create a new row, then</td>
</tr>
<tr>
<td></td>
<td>click in the Expression column and click the ellipsis button to open the</td>
</tr>
<tr>
<td></td>
<td>SQL Editor to specify an expression. For example, to define an index to</td>
</tr>
<tr>
<td></td>
<td>convert all names to lowercase to simplify searching, you could enter an</td>
</tr>
<tr>
<td></td>
<td>expression such as: lower(SURNAME)</td>
</tr>
</tbody>
</table>

2.3.5.3 Rebuilding Indexes

You can rebuild indexes at any time to reflect any changes that you have made to primary keys, foreign keys, or alternate keys in your model.

**Procedure**

1. Select **Tools** ➤ **Build Objects** ➤ **Rebuild Indexes** and enter the appropriate options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary key</td>
<td>Rebuilds primary key indexes. The field displays the naming convention for</td>
</tr>
</tbody>
</table>
<pre><code>                    | primary key indexes, which is by default STABLES_PK. You can use the        |
                    | following variables:                                                       |
                    | - STABLES - Generated code of the table. The generated code of a variable   |
                    | is the code defined in the object property sheet, but may be truncated if   |
                    | it contains characters not supported by the DBMS.                         |
                    | - STRNAME, VTCODE, VSTLABL - Table name, code, and comment.                |
</code></pre>
<p>| Other keys          | Rebuilds alternate key indexes. The field displays the naming convention    |
| for alternate key indexes, which is by default KEYS_AL.                    |
| Foreign key indexes | Rebuild foreign key indexes. The field displays the naming convention for   |
| foreign key indexes, which is by default REFRA_PK. You can use the          |
| following variables:                                                       |
| - REFRA, VTPARENTS, VSCHILD - Generated code of the reference, parent, and |
| child table.                                                               |
| - SPAREMS, VPCODES, VQUALIFIERS - Parent table name, code, and qualifier.  |
| - SACSNAME, VCCODES, VEQUALITIES - Child table name, code, and qualifier.  |
| - SREPNAME, VREFCODES - Reference name or code.                           |
| Foreign key threshold| Specifies the minimum number of estimated records in a table (specified in  |
| the Number field in the table properties sheet) that are necessary before  |
| a foreign key index can be created. If the Number field is empty, foreign |
| key indexes are generated.                                                 |
| Mode                | Specifies the type of rebuild. You can select:                              |
| - Delete and Rebuild - Delete and rebuild all indexes attached to primary,  |
| alternate, and foreign keys.                                              |
| - Add missing indexes - Preserve existing key indexes and add any missing   |
| key indexes.                                                              |</p>

2. [optional] Click the Selection tab to specify which tables you want to rebuild indexes for.
3. Click **OK**. If you selected the Delete and Rebuild mode, a confirmation box asks you to confirm your choice. Click **Yes** to confirm the deletion and rebuild of the selected indexes.

2.3.6 Views (PDM)

A view is a query that provides access to all or a subset of the data in a table or multiple tables connected by joins. Views do not copy the data from their underlying tables and are updated when data in those tables changes. Views can reference other views, can order or filter data as necessary, and may be
indistinguishable from tables for users accessing them.

In this section:
- Creating a View
- View Properties
- View Queries
- Materialized Views
- Showing View Dependencies using Traceability Links
- Defining a Generation Order for Views

2.3.6.1 Creating a View

You can create a view populated with columns from selected tables and other views via the Tools menu. Alternately, you can create an empty view from the Toolbox, Browser, or Model menu.

Procedure

1. [optional] Select one or more tables and views in the diagram. You can select multiple objects by holding down the Shift key while you select them.

2. Select Tools > Create View.

   If you have not selected any tables or views, then a selection box opens, allowing you to select the objects to be included in the view. Select the appropriate objects and then click OK.

   A view symbol is created in the diagram, displaying all the columns in each of the tables and views selected for the view. The names for the tables and views appear at the bottom of the view symbol.

3. [optional] Edit the view’s query to remove unwanted columns or otherwise modify the view (see View Queries).

   Alternatively, you can create an empty view, which you should complete by specifying a query (see View Queries) in the following ways:
   - Use the View tool in the Toolbox.
   - Select Model > Views to access the List of Views, and click the Add a Row tool.
   - Right-click the model (or a package) in the Browser, and select New View.

2.3.6.2 View Properties

To view or edit a view’s properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object’s purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the view, where supported by your DBMS (see Materialized Views and Creating an XML Table or View).</td>
</tr>
<tr>
<td>Owner</td>
<td>Specifies the user who is the owner of the object. This is usually its creator. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Usage</td>
<td>Specifies how the view will be used. You can choose between: Query only - Consultation only. The view cannot update tables. Updatable - Consultation and update of underlying tables. With Check options - Implements controls on view insertions.</td>
</tr>
<tr>
<td>Dimensional type</td>
<td>Specifies the type of the view for purposes of creating star or snowflake schemas containing fact tables and dimensions. You can choose between: Fact - see Facts (PDM) Dimension - see Dimensions (PDM) Exclude - PowerDesigner will not consider the view when identifying or generating multidimensional objects. You can instruct PowerDesigner to complete this field for you (see Identifying Fact and Dimension Tables). PowerDesigner’s support for the generation of BusinessObjects universes (see Generating an SAP BusinessObjects Universe) and of facts and dimensions in a multidimensional diagram (see Generating Cubes) depends on the value of this field.</td>
</tr>
<tr>
<td>Generate</td>
<td>Selects the view for generation to the database.</td>
</tr>
<tr>
<td>User-defined</td>
<td>By default, the view query is updated to reflect changes to model objects on which it is based. Selecting this option freezes the view and protects your manual changes.</td>
</tr>
</tbody>
</table>
The following tabs are also available:

- **Columns**: Lists the columns in the view based on the SELECT orders from the queries. You can modify column properties in this list but to add or remove columns, you must modify the appropriate view query. View column properties are initialized from the properties of their source columns. The read-only `Expression` column specifies the qualified name of the view column.

- **Indexes**: Lists the indexes defined on the materialized view (see Indexes (PDM)).

- **SQL Query**: Displays the SQL code for all the queries associated with the view. You can edit this code directly in this tab or access the property sheets of individual queries (see View Queries).

- **Triggers**: Lists the triggers associated with the view (see Triggers (PDM)). You can define a trigger to fire when one or more attributes of a table view column are modified.

- **Preview**: Displays the SQL code to be generated for the view (see Previewing SQL Statements).

### 2.3.6.3 View Queries

You can edit queries associated with a view from the **SQL Query** tab of the view property sheet.

Any number of queries may be associated with a view, and the totality of their SQL statements is shown in this tab, linked by any of the standard SQL constructs, such as `Union`, etc.

You can edit the code shown in the **SQL Query** tab:

- **Directly in the tab**.
- **Click the `Edit with SQL Editor` tool** to edit the code in the PowerDesigner SQL Editor (see Writing SQL Code in PowerDesigner).
- **Click the `Edit with` tool (CTRL+E)** to open the code in your favorite editor.

Any edits you make in this tab will propagate to the property sheets of the associated individual queries, which are available from the **Query** list at the bottom of the tab. Use the tools to the right of this list to create a new query (with the appropriate linking construct), delete the selected query, or open the property sheet of the selected query.

The following SQL constructs are available (if supported by your DBMS) for linking queries:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Result</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Union</code> [default]</td>
<td>Displays all the data retrieved by both the queries, except where results are repeated.</td>
<td>SELECT 1: ABC SELECT 2: BCD Result: ABCD</td>
</tr>
<tr>
<td><code>Union All</code></td>
<td>Displays all the data retrieved by both the queries, including repeated results.</td>
<td>SELECT 1: ABC SELECT 2: BCD Result: ABCBDCD</td>
</tr>
<tr>
<td><code>Intersect</code></td>
<td>Displays only the data retrieved by both the queries.</td>
<td>SELECT 1: ABC SELECT 2: BCD Result: BC</td>
</tr>
<tr>
<td><code>Minus</code></td>
<td>Displays only the data retrieved by one or other of the queries, but not by both.</td>
<td>SELECT 1: ABC SELECT 2: BCD Result: AD</td>
</tr>
</tbody>
</table>

The following tabs are available:

- **SQL** tab - displays the SQL code for the query. You can edit the query directly in this tab or in PowerDesigner's built-in SQL Editor (see Writing SQL Code in PowerDesigner) by clicking the `Edit with SQL Editor` tool or in an external editor by clicking the `Edit with` tool (CTRL+E). Any edits you make in this tab will propagate to the query's other tabs and the **SQL Query** tab of the parent view, as changes made in other tabs will propagate here and to the parent view.

- **Tables** tab - lists the tables in the `FROM` clause. You can add or delete tables in the list, and reorder the list using the arrows at the bottom of the tab. For each line, select a table or click the ellipsis button to enter a more complex expression in the SQL Editor and, optionally, enter an alias in the **Alias** column. For the second and subsequent lines in the list you can specify an appropriate join condition keyword, and then specify the join condition.

- **Columns** tab - lists the columns in the `SELECT` clause. You can add or delete columns in the list, specify aliases for them, and reorder the list using the arrows at the bottom of the tab.

- **Where** tab - lists the expressions in the `WHERE` clause. You can add or delete expressions in the list, and reorder the list using the arrows at the bottom of the tab. For each line, select a column in each of the two `Expression` columns (or click the ellipsis button to specify a more complex expression), and select the appropriate operator between them. You can optionally enter a prefix and suffix.
Group By tab - lists the columns in the `GROUP BY` clause. You can add or delete columns in the list, and reorder the list using the arrows at the bottom of the tab.

Having tab - lists the expressions in the `HAVING` clause. You can add or delete expressions in the list, and reorder the list using the arrows at the bottom of the tab. For each line, select a column in each of the two `Expression` columns (or click the ellipsis button to specify a more complex expression), and select the appropriate operator between them. You can optionally enter a prefix and suffix.

Order By tab - lists the columns in the `ORDER BY` clause. You can add or delete columns in the list, and reorder the list using the arrows at the bottom of the tab. For each line, select a column (or click the ellipsis button to specify a more complex expression), and select ASC or DESC for the sort direction.

2.3.6.4 Materialized Views

A materialized view is a table containing the results of a query. PowerDesigner supports materialized views for the DB2, HP Neoview, Netezza, Oracle, and SQL Anywhere DBMS families.

Materialized views are supported in the following ways:

- DB2 - Select `materialized query table` (or for earlier versions, `summary table`) in the `Type` list on the `General` tab of a view property sheet.
- HP Neoview - Use the List of Materialized Views (available from `Model > Materialized Views`).
- Netezza - Use the List of Materialized Views (available from `Model > Materialized Views`).
- Oracle - Use the List of Materialized Views (available from `Model > Materialized Views`).
- SQL Anywhere - Select `Materialized View` in the `Type` list on the `General` tab of a view property sheet to display the `DB space` field, and specify the dbspace in which to create the materialized view. The default is the current dbspace.

2.3.6.5 Showing View Dependencies using Traceability Links

You can use traceability links to make the relationships between views and tables clearer. These links are not interpreted and checked by PowerDesigner.

In the following example, the `Book Sales` view is shown as depending on the `Title` and `Sale` tables via two traceability links with their type set to `depends on`:

![Diagram showing traceability links between views and tables]

For detailed information about traceability links, see Core Features Guide > Modeling with PowerDesigner > Objects > Traceability Links.

2.3.6.6 Defining a Generation Order for Views

You can define the order of the generation of views by using traceability links with a type of `DBCreateAfter`. The view from which you start the traceability link is dependent on the view you link it to, and this influent view will be generated before the dependent view.

Context

For example you create the view `DEPARTMENT STORE` from the table `STORE`, and then another view called `COMPUTER COUNTER` from the view `DEPARTMENT STORE` to show only part of the department store offer.

By default, views are generated in alphabetical order, so the generation of `COMPUTER COUNTER` will fail since the view `DEPARTMENT STORE` on which it depends is not yet generated. To bypass this problem, you should create a traceability link of type `<<DBCreateAfter>>` from `COMPUTER COUNTER` to `DEPARTMENT STORE` to ensure that `DEPARTMENT STORE` is generated before `COMPUTER COUNTER`.

Note

There is a model check to warn you if you create a reflexive or circular set of traceability links of type `DBCreateAfter`. If you generate without correcting this error, views will be generated in alphabetical order, without taking into account the generation order.
### Procedure

1. Select the Traceability Links tool in the toolbox.
2. Click inside the dependent view and, while holding down the mouse button, drag the cursor into the influent view. Release the mouse button.
3. Double-click the traceability link to open the property sheet of the dependent object at the Traceability Links tab.
   - The influent view is displayed in the Linked Object column.
4. Click in the Link Type column, click the down arrow and select DBCreateAfter.
5. Click OK to close the property sheet and return to your model.

---

### 2.3.7 Triggers (PDM)

A trigger is a segment of SQL code associated with a table or a view, which is invoked automatically whenever there is an attempt to modify data in the associated table or view with an insert, delete, or update command. A DBMS trigger is not associated with any table or view, and fires on modifications to the database structure itself, such as the creation or dropping of a table or events like startup, shutdown, login etc. You can use triggers to enforce referential integrity (where declarative constraints are not sufficient) and to implement sequences for columns.

In the PowerDesigner interface, table and view triggers are called simply triggers, while DDL or database triggers are called DBMS triggers. View and DBMS triggers are not supported by all DBMSs.

PowerDesigner provides trigger templates to generate triggers (see Trigger Templates) and template items, which are reusable blocks of SQL script that can be inserted into triggers or trigger templates (see Trigger Template Items) and you can modify these templates and items and create your own.

In this section:

- Creating a Table or View Trigger
- Creating Triggers from References
- Creating a DBMS Trigger
- Trigger and DBMS Trigger Properties
- Trigger Naming Conventions
- Calling a Related Procedure in a Trigger Template
- Indicating Trigger Order for Multiple Triggers
- Defining Triggers with Multiple Events
- Rebuilding Triggers
- Trigger Templates
- Trigger Template Items
- Creating SQL/XML Queries with the Wizard
- Generating Triggers and Procedures

#### 2.3.7.1 Creating a Table or View Trigger

You can create a trigger for a table from its property sheet and base it on a PowerDesigner template, or on a template of your own, or write it from scratch.

### Procedure

1. Open the table or view property sheet, and then click the Triggers tab.
2. Click the Add a Row tool to create a new trigger, enter a name and code, and then click the Properties tool to open its property sheet.
3. Click the Definition tab, and select a trigger template (see Trigger Templates) from the Template list. The time and event fields will be set and the template code copied into the definition editor.
4. [optional] Modify the trigger definition code. You can insert trigger template items (see Trigger Template Items), use PDM variables and macros and various other tools available from the toolbar (see Writing SQL Code in PowerDesigner).

If you edit the code, then the trigger will be marked as user-defined and will be excluded from most forms of rebuilding (see Rebuilding Triggers).

5. Click OK to return to your model.

### 2.3.7.2 Creating Triggers from References

You can create triggers to enforce referential integrity individually or instruct PowerDesigner to create them by default.

**Procedure**

1. Create a reference between two tables, and then double click the reference symbol to open its property sheet.
2. Click the **Integrity** tab, and then select **Trigger** from the **Implementation** list.
3. Specify the form of Update and Delete constraints using the radio buttons (see Reference Properties), and then click OK to return to the diagram.
4. If you have set the **Automatically rebuild triggers** model option (see Reference Model Options,) then triggers will have been created automatically in the parent and child tables. To verify this open the table property sheet and click the **Triggers** tab. If the triggers are not present, you will need to rebuild your triggers manually (see Rebuilding Triggers).

**Note**

To instruct PowerDesigner to implement referential integrity between tables using triggers by default whenever you create a reference, select **Tools** > **Model Options**, click **Model Settings** > **Reference** in the Category list, select **Trigger** in the Default implementation list.

### 2.3.7.3 Creating a DBMS Trigger

DBMS triggers are not associated with any table or view. You create them directly under the model.

**Context**

You can create a DBMS trigger in any of the following ways:

- Select **Model** > **Triggers** > **DBMS Triggers** to access the List of DBMS Triggers, and click the **Add a Row** tool.
- Right-click the model (or a package) in the Browser, and select **New** > **DBMS Trigger**.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

### 2.3.7.4 Trigger and DBMS Trigger Properties

To view or edit a trigger's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The **General** tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
</tbody>
</table>
Definition Tab

This tab allows you to enter code for the trigger. For information about the tools available on the toolbar, see Writing SQL Code in PowerDesigner. The following properties are available:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Specifies the template on which the trigger is based (see Trigger Templates). The User-defined button is automatically depressed when you modify the definition of a trigger. Click the button to release it and restore the template trigger definition.</td>
</tr>
<tr>
<td>Time</td>
<td>Specifies when the trigger will fire in relation to the associated event. The content of the list depends on the values defined in the trigger template and in the Time entry in the Trigger category of the DBMS.</td>
</tr>
<tr>
<td>Event</td>
<td>Specifies the event that will cause the trigger to fire. Click the ellipsis tool to the right of this field to select multiple events (see Defining Triggers with Multiple Events). For table and view triggers, this field is a list, the content of which depends on the values defined in the trigger template and in the Event entry in the Trigger category of the DBMS. You can add your own events to this entry and they will appear in this list. For DBMS triggers, this field allows you to enter any text.</td>
</tr>
<tr>
<td>Order</td>
<td>[table and view triggers only] Specifies the firing order of trigger.</td>
</tr>
</tbody>
</table>

The following tabs are also available:
- **Template Items** - lists the trigger template items available for use in the trigger definition (see Trigger Template Items).
- **Preview** - displays the SQL code that will be generated for the trigger (see Previewing SQL Statements).

2.3.7.5 Trigger Naming Conventions

The pre-defined trigger templates that ship with PowerDesigner indicate naming conventions for the trigger scripts that it generates. The naming convention consists of a prefix indicating the trigger type followed by the table code.

**Context**

The default naming conventions include a variable (%L:TABLE). The name of the resulting trigger script replaces this variable with a lower-case table code. For example, a resulting trigger script may have the name ti_employee.

You can change the trigger naming convention in PowerDesigner pre-defined DBMS trigger templates from the Trigger Templates tab of the DBMS property sheet.

**Procedure**

1. Select [Database] Edit Current DBMS to open the DBMS definition file in the Resource Editor, and then click the Trigger Template tab.
2. Click a trigger template in the list, and then click the Properties tool to open its property sheet.
3. Type a new trigger name in the Trigger Name text box at the bottom of the tab. For example, mytempl_%TABLE%
4. Click OK in each of the dialog boxes.

2.3.7.6 Calling a Related Procedure in a Trigger Template

Some target databases do not accept code within a trigger statement. For these databases, a trigger template can call a related procedure as a parameter, which is defined in a procedure template. In these cases, procedure templates are listed in the list of trigger templates.

**Example**

Informix does not accept code in trigger templates. The template InsertTrigger calls the procedure in the form of the variable %PROC%, as follows:

```
-- Insert trigger "$QUALIFIER$\$TRIGGER$" for table "$QUALIFIER$\$TABLE$"
create trigger "$QUALIFIER$\$TRIGGER$" insert on "$QUALIFIER$\$TABLE$
  referencing new as new_ins
  for each row (execute procedure %PROC%(FKCOLN("new_ins.%COLUMN%", ",", ",", ");))
/
```

The template InsertProc defines the procedure, as follows:

```
-- Insert procedure "$PROC$" for table "$QUALIFIER$\$TABLE$
```
2.3.7.7 Indicating Trigger Order for Multiple Triggers

Some DBMSs allow you to have multiple triggers for the same insert, update, or delete event at the same time. You can indicate the order in which each trigger within the group fires.

Context

In the following example, a company is considering candidates for various positions, and must ensure that new employees are offered a salary that is within the range of others working in the same field, and less than their prospective manager.

The EMPLOYEE table contains two Before Insert triggers to perform these tests:

```sql
create trigger tibTestSalry1 before insert order 1 on EMPLOYEE
referencing new as new_ins for each row
begin
  [Trigger code]
end;
create trigger tibTestSalry2 before insert order 2 on EMPLOYEE
begin
  [Trigger code]
end;
```

Procedure

1. Open the trigger property sheet and click the Definition tab.
2. Select a number from the Order list to indicate the position in which the trigger fires.
3. Click OK to return to your model.

2.3.7.8 Defining Triggers with Multiple Events

Some DBMSs support multiple events on triggers. If such is the case, the Ellipsis button to the right of the Event box on the trigger definition tab is available.

You can click the Ellipsis button to open the Multiple Events Selection box. If you select several events and click OK, the different events will be displayed in the Event box, separated by the appropriate delimiter.

2.3.7.9 Rebuilding Triggers

PowerDesigner can rebuild triggers to ensure that they are attached to all tables joined by references to ensure referential integrity. You can instruct PowerDesigner to automatically rebuild triggers whenever a relevant change is made and you can manually rebuild triggers at any time.

Context

The Rebuild Triggers function creates new triggers based on template items that correspond to trigger referential integrity defined for references and sequence implementation for columns.

To instruct PowerDesigner to automatically rebuild triggers, select Tools > Model Options > Model Settings > Trigger, select Automatically rebuild triggers, and click OK. PowerDesigner rebuilds all triggers and will, from now on, rebuild triggers whenever you make a relevant change in the model.

To rebuild triggers manually:

Procedure

1. Select Tools > Rebuild Objects > Rebuild Triggers
2. Specify a rebuild mode. You can choose between:
   - Delete and Rebuild – all triggers attached to templates are deleted and rebuilt, including those to which you have made modifications.
Preserve – only those triggers attached to templates that have not been modified are deleted and rebuilt. Any triggers that you have modified are preserved.

3. The Trigger selection box shows an expandable tree view of trigger types. Expand the tree and select the types to rebuild. There are three levels in this tree:
   - All trigger types supported by the current DBMS
   - All trigger templates corresponding to the trigger types
   - All template items defined for each trigger template

For example, in the list below, the two template items InsertChildParentExist and InsertTooManyChildren are used in the BeforeInsertTrigger template that is, in turn, used in all triggers with a time of Before and an event type of Insert:

4. [optional] Click the Error Messages tab to define the types of error messages to generate (see Generating a User-Defined Error Message).
5. [optional] Click the Selection tab to specify which tables to rebuild the triggers for.
6. Click OK to begin the rebuild process.

Progress is shown in the Output window. You can view the triggers that have been created from the Triggers tab of the table property sheet, or from the List of Triggers.

Note
If you change the target DBMS family, for example from Oracle to IBM DB2, triggers are automatically rebuilt.

For information about rebuilding dependencies between triggers and other objects, see Tracing Trigger and Procedure Dependencies.

### 2.3.7.10 Trigger Templates

PowerDesigner trigger templates allow you to write trigger code in a modular reusable fashion. We provide basic templates for before, after, and with insert, update, and delete events and for other types of triggers where supported by the DBMS. You can modify the code specified in these templates or create your own templates in the DBMS definition file or in your model.

To apply a trigger template to your trigger definition, select the template from the list on the trigger property sheet Definition tab (see Trigger and DBMS Trigger Properties).

To review or modify the provided trigger templates, select Database Edit Current DBMS, and then click the Trigger Templates tab. You cannot delete or rename these templates.

Caution
The resource files provided with PowerDesigner inside the Program Files folder cannot be modified directly. To create a copy for editing, use the New tool on the resource file list, and save it in another location. To include resource files from different locations for use in your models, use the Path tool on the resource file list.

To create a new template, click the Create from Trigger Template tool (to copy the code of an existing template to your new template) or the Add a Row tool (to start from scratch).

Note
You can, alternatively, create trigger templates in your model by selecting Model Triggers Trigger Templates, but these templates will not be accessible from other models.

### Trigger Template Properties

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>DBMS</td>
<td>Specifies the parent DBMS.</td>
</tr>
</tbody>
</table>
The following tabs are also available:

- **Definition** - Contains a field for entering the trigger code for the template. You can use trigger template items, PDM variables and macros and other tools available from the toolbar (see Writing SQL Code in PowerDesigner).
- **Template Items** - Lists the template items (see Trigger Template Items) that are defined in the trigger template and that will be generated when a trigger is generated from the template.

**Note**

If you delete a template item from this list, it is not deleted from the template definition, but is excluded from generation when rebuilding triggers. PowerDesigner-provided template items listed on this tab are generated in a trigger if they match the trigger implemented referential integrity defined for a reference attached to the table. User-created template items are always generated regardless of trigger referential integrity constraints.

### 2.3.7.11 Trigger Template Items

Trigger template items are named reusable blocks of script that can be inserted into triggers or trigger templates. In a generated trigger script, a template item calls a macro that implements a trigger referential integrity constraint or does any other updating work on tables in the database.

To insert a trigger template item into your trigger or template definition, click the **Add Trigger Item from Model** or **Add Trigger Item from DBMS** tool, select the items from the list and click OK. The item is inserted with a dot followed by its name, and is also added to the list on the Template Items tab. For example, the following script contains two template items: `InsertChildParentExist` and `InsertTooManyChildren`:

```sql
/* Before insert trigger "%TRIGGER%" for table "[%QUALIFIER%]%TABLE%" */
create trigger %TRIGGER% before insert order %ORDER% on [%QUALIFIER%]%TABLE%
referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
  declare found integer;
  .InsertChildParentExist
  .InsertTooManyChildren
end
/
```

**Note**

Certain DBMSs require that a cursor and variables are declared for each template item before the template item name is used in the script. You can use the following format to declare a template item:

```
.DECL <template item name>
```

For example, the trigger definition for Oracle 8 declares and then inserts the `DeclInsertChildParentExist` template item:

```sql
-- Before insert trigger "%QUALIFIER%\%TRIGGER%" for table "%QUALIFIER%\%TABLE%"
create trigger %QUALIFIER%\%TRIGGER% before insert
on %QUALIFIER%\%TABLE% for each row
declare
  integrity_error exception;
  errno integer;
  errmsg char(200);
  dummy integer;
  found boolean;
  .DeclInsertChildParentExist
begin
  .InsertChildParentExist
  -- Errors handling
  exception
  when integrity_error then
    raise_application_error(errno, errmsg);
  end;
/
```

To review or modify the provided trigger template items, select **Database > Edit Current DBMS**, and then click the **Trigger Template Items** tab. You cannot delete or rename these items.

**Caution**

The resource files provided with PowerDesigner inside the Program Files folder cannot be modified directly. To create a copy for editing, use the **New** tool on the resource file list, and save it in another location. To include resource files from different locations for use in your models, use the **Path** tool on the resource file list.

To create a new template item, click the **Create from DBMS Trigger Item** tool (to copy the code of an existing item to your new item) or the **Add a Row** tool (to start from scratch).

**Note**

You can, alternatively, create trigger template items in your model by selecting **Model > Triggers > Trigger Template Items**, but these templates will not be accessible from other models.
Trigger Template Item Properties

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the [Code] field.</td>
</tr>
<tr>
<td>DBMS</td>
<td>Specifies the parent DBMS.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- **Definition** - Contains a field for entering the trigger code for the item. You can use PDM variables and macros and other tools available from the toolbar (see Writing SQL Code in PowerDesigner).
- **Declaration** - Contains a field for entering the declaration for the item in trigger scripts.

In this section:

- **PowerDesigner Pre-Defined Trigger Template Items**

2.3.7.11.1 PowerDesigner Pre-Defined Trigger Template Items

PowerDesigner provides pre-defined template items for the trigger templates defined in each DBMS. The Rebuild Triggers function uses both pre-defined and user-defined trigger templates to automatically create triggers for selected tables.

In the pre-defined trigger templates, each pre-defined template item corresponds to a referential integrity constraint. Although a pre-defined template item is defined in a trigger template, it is only generated in a trigger script if it implements the trigger referential integrity defined for a reference. The item is available for generation if it is present on the **Template Items** tab of a trigger property sheet and will be generated if it is present on the **Template Items** tab of a trigger template property sheet.

Insert Constraints

The template items below implement referential integrity in insert trigger templates.

<table>
<thead>
<tr>
<th>Template item</th>
<th>Integrity constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeclInsertChildParentExist</td>
<td>Mandatory parent</td>
<td>Parent must exist when inserting a child</td>
</tr>
<tr>
<td>InsertChildParentExist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclInsertTooManyChildren</td>
<td>Cannot exceed maximum cardinality constraint</td>
<td>Cannot insert a child if maximum cardinality has been reached</td>
</tr>
<tr>
<td>InsertTooManyChildren</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclInsertSequenceColumn</td>
<td>Select value in sequence list for column</td>
<td>Select a value for the column from a list of sequences</td>
</tr>
<tr>
<td>InsertSequenceColumn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Update Constraints

The template items below implement referential integrity in update trigger templates.

<table>
<thead>
<tr>
<th>Template item</th>
<th>Integrity constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeclUpdateChildParentExist</td>
<td>Mandatory parent</td>
<td>Parent must exist when updating a child</td>
</tr>
<tr>
<td>UpdateChildParentExist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateChildChangeParent</td>
<td>Change parent not allowed</td>
<td>Cannot modify parent code in child</td>
</tr>
<tr>
<td>UpdateChildChangeParent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateParentRestrict</td>
<td>Restrict on update</td>
<td>Cannot modify parent if child exists</td>
</tr>
<tr>
<td>UpdateParentRestrict</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateParentCascade</td>
<td>Cascade on update</td>
<td>Modify parent code in all children</td>
</tr>
<tr>
<td>UpdateParentCascade</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateChangeColumn</td>
<td>Non-modifiable column</td>
<td>Cannot modify column</td>
</tr>
<tr>
<td>UpdateChangeColumn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateParentSetNull</td>
<td>Set null on update</td>
<td>Set parent code to null in all children</td>
</tr>
<tr>
<td>UpdateParentSetNull</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateParentSetDefault</td>
<td>Set default on update</td>
<td>Set parent code to default in all children</td>
</tr>
<tr>
<td>UpdateParentSetDefault</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclUpdateTooManyChildren</td>
<td>Cannot exceed maximum cardinality constraint</td>
<td>Cannot update a child if maximum cardinality has been reached</td>
</tr>
<tr>
<td>UpdateTooManyChildren</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Delete Constraints

The template items below implement referential integrity in delete trigger templates.
Constraint Messages
You can insert the following template items in any trigger template. They generate error messages that indicate the violation of an integrity constraint.

<table>
<thead>
<tr>
<th>Template item</th>
<th>Integrity constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeclDeleteParentRestrict</td>
<td>Restrict on delete</td>
<td>Cannot delete parent if child exists</td>
</tr>
<tr>
<td>DeleteParentRestrict</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclDeleteParentCascade</td>
<td>Cascade on delete</td>
<td>Delete parent code in all children</td>
</tr>
<tr>
<td>DeleteParentCascade</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclDeleteParentSetNull</td>
<td>Set null on delete</td>
<td>Delete in parent sets child to null</td>
</tr>
<tr>
<td>DeleteParentSetNull</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeclDeleteParentSetDefault</td>
<td>Set default on delete</td>
<td>Delete in parent sets child to default</td>
</tr>
<tr>
<td>DeleteParentSetDefault</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**2.3.7.12 Creating SQL/XML Queries with the Wizard**

You can use the SQL/XML Wizard to insert a SQL/XML query in the definition of a trigger, stored procedure, or function to store or retrieve data, in an XML format, from relational databases supporting SQL/XML. The wizard allows you to select tables and views from a PDM to build a mapped XML model. This XML model (which does not appear in the workspace) is used to generate SQL/XML queries from global elements.

**Procedure**

1. Open the trigger property sheet, click the **Definition** tab and position the cursor in the trigger definition where you want to insert the SQL/XML query:

![Trigger Properties](image)

2. Click the **SQL/XML Wizard** tool to launch the wizard at the **Tables and Views Selection** page:
3. Select the tables and views that you want to include in your query and click Next to go to the XML Hierarchy Design page:

On this tab, you construct the XML hierarchy that you want to generate:

- The left-hand pane lists the tables and views that you have selected.
- The right-hand pane displays the XML hierarchy to be generated, containing a default root element.

4. You can build your XML hierarchy using the following techniques:

- Specify whether columns will be generated as elements or attributes by using the radio buttons above the panes.
- Drag and drop a table, view, or column onto a node in the XML hierarchy. You must respect the PDM hierarchy: You cannot create an XML hierarchy between two elements if there is no reference between their corresponding tables, and a parent table cannot be placed beneath one of its children.
- Right-click a table, view, or column and select Add from the contextual menu to add it to the last selected node in the XML hierarchy.
- Rename an element or attribute by clicking its node and typing a new name.
- Create new elements and attributes not in the PDM, and Sequence, Choice and All group particles, by right-clicking an XML node and selecting New from the contextual menu.
- Delete an XML node by right-clicking it and selecting Delete from the contextual menu.

5. When you have finished building your hierarchy, click Next to go to the Query tab:
6. Review your query and click Back, if necessary, to make revisions in your hierarchy. When you are satisfied, click Finish to close the wizard and insert the SQL/XML query in the trigger definition.

7. [optional] Add code to complete the SQL/XML query:
8. Click OK to close the trigger property sheet:

2.3.7.13 Generating Triggers and Procedures

You can create or modify database triggers to a script or to a live database connection.

Procedure

1. Select Database > Generate Database to open the Database Generation window, and specify the standard options, including whether you want to generate to a script or to a live database connection.

For detailed information about using this window, see the Generating a Database from a PDM.

2. Select "Triggers & Procedures (with Permissions)" from the Settings set list in the Quick Launch groupbox at the bottom of the window. This settings set specifies standard options for generating triggers and procedures.

or:

Click the Options tab and click on Trigger in the left-hand pane to display the trigger generation options. Change the default options as appropriate.

For detailed information about settings sets, see Quick Launch Selection and Settings Sets.

3. [optional] Click the Selection tab and select the Table or Procedure subtab at the bottom of the tab. Select the tables or procedures that you want to generate for. Note that if you want to generate a trigger script for tables owned by a particular owner, you can select an owner from the Owner list.

4. Click OK to begin the generation.

In this section:
- Defining a Generation Order for Stored Procedures
- Creating User-Defined Error Messages

2.3.7.13.1 Defining a Generation Order for Stored Procedures

You can define the order of the generation of stored procedures by using traceability links with a type of DBCreateAfter. The procedure from which you start the traceability link is dependent on the procedure you link it to, and this influent procedure will be generated before the dependent procedure.

Context

For example, a publisher may decide to sell certain books at a reduced rate (15%) when a customer's order is above 10 000$. The GENERAL CHECK stored procedure verifies orders globally by checking availability, the order amount, if a discount rate is required, and so on. This procedure calls the DISCOUNT CALC procedure to calculate the 15% discount rate. Consequently, DISCOUNT CALC must be generated before GENERAL CHECK, and you can enforce this by creating a traceability link of type DBCreateAfter from GENERAL CHECK to DISCOUNT CALC.

Note

There is a model check to warn you if you create a reflexive or circular set of traceability links of type DBCreateAfter. If generate without correcting this error, procedures will be generated in alphabetical order, without taking into account the generation order.

Procedure

1. Open the property sheet of the dependent stored procedure and click the Traceability Links tab.

2. Click the Add Objects tool, click the Procedure sub-tab in the Add Object selection dialog, select the influent stored procedure, and click OK.

3. Click in the Link Type column, click the down arrow and select DBCreateAfter.
4. Click **OK** to close the property sheet and return to your model.

**Note**

You can also create `DBCreateAfter` traceability links using the `Traceability Links` tool (see Defining a Generation Order for Views). For detailed information about traceability links, see Core Features Guide > Modeling with PowerDesigner > Objects > Traceability Links.

### 2.3.7.13.2 Creating User-Defined Error Messages

You can create a message table in your database to store user-defined error messages. When you select trigger generation parameters, you can choose to generate an error message from this table.

#### Procedure

1. Create a table with columns to store the following information:

<table>
<thead>
<tr>
<th>Column to store...</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error number</td>
<td>Number of the error message that is referenced in the trigger script</td>
</tr>
<tr>
<td>Message text</td>
<td>Text of message</td>
</tr>
</tbody>
</table>

2. Generate the table in your database.
3. Select **Tools** > **Rebuild Objects** > **Rebuild Triggers**.
4. Select the **Error Messages** tab, and select the **User-defined** radio button.
5. Enter the name of the table that contains the error message, the name of the column that contains the error number, and the name of the column that contains the error message text.

### 2.3.7.13.2.1 Generating a User-Defined Error Message

You can choose to generate a user-defined error message from the trigger generation parameters box.

#### Procedure

1. Select **Tools** > **Rebuild Objects** > **Rebuild Triggers**.
2. Click the **Error Messages** tab, and select the **User-defined** radio button.
3. Enter the name of the table that contains the error message, the name of the column that contains the error number, and the name of the column that contains the error message text.
4. Click the General tab and select the mode and triggers to create.
5. Click the Selection tab and select the tables for which you want to create triggers.
   For more information on rebuilding triggers, see Rebuilding Triggers.
6. Click OK.
   The trigger rebuilding process is shown in the Output window.
7. Select Database, Generate Database, select generation parameters as required (see Generating Triggers and Procedures), and click OK.

2.3.8 Stored Procedures and Functions (PDM)

You can define stored procedures and functions for any DBMS that supports them.

A stored procedure is a precompiled collection of SQL statements stored under a name and processed as a unit. Stored procedures are stored within a database; can be executed with one call from an application; and allow user-declared variables, conditional execution, and other programming features.

The use of stored procedures can be helpful in controlling access to data (end-users may enter or change data but do not write procedures), preserving data integrity (information is entered in a consistent manner), and improving productivity (statements in a stored procedure only need to be written one time).

A user-defined function is a form of procedure that returns a value to the calling environment for use in queries and other SQL statements.

In this section:
- Creating a Stored Procedure or Function
- Procedure Properties
- Tracing Trigger and Procedure Dependencies
- Attaching a Stored Procedure to a Table
- Procedure Templates (PDM)

2.3.8.1 Creating a Stored Procedure or Function

You can create a stored procedure or function from a table property sheet or from the Toolbox, Browser, or Model menu.

Context
- Use the Procedure tool in the diagram Toolbox.
- Open the Procedures tab in the property sheet of a table, and click the Add a Row tool.
- Select File > Model > Procedures to access the List of Procedures, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select New > Procedure.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

You can create a procedure based on one of the PowerDesigner templates or on a template of your own.

Procedure
1. Double-click a table symbol to open its property sheet, and then click the Procedures tab.
2. Click the Add a Row tool to create a new procedure, and type a name and code.
3. Click Apply to commit the creation of the new procedure, and then click the Properties tool to open its property sheet.
4. Click the Definition tab.
5. [optional] Select a procedure template from the Template list (see Procedure Templates (PDM)).

6. Modify the procedure definition code. You can use PDM variables and macros and various other tools available from the toolbar (see Writing SQL Code in PowerDesigner).

7. You can also modify the procedure's other properties. For a full list of the properties available, see Procedure Properties.

8. Click OK in each of the dialog boxes.

Results

Note

When using the PowerDesigner Eclipse plug-in, you can right-click a procedure in the Browser or diagram and select Edit in SQL Editor from the contextual menu to open it in the Eclipse SQL Editor. You can optionally connect to your database in order to obtain auto-completion for table names. The procedure definition is added to the Generated SQL Files list in the Workspace Navigator.

2.3.8.2 Procedure Properties

To view or edit a procedure's properties, double-click its diagram symbol or Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.</td>
</tr>
<tr>
<td>Owner</td>
<td>Specifies the name of the procedure owner.</td>
</tr>
<tr>
<td>Table</td>
<td>Specifies the table to which the procedure is attached. Use the tools to the right of the list to create, browse for, or view the properties of the currently selected object.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- Definition - allows you to enter the SQL code for the procedure. For information about the tools available, see Writing SQL Code in PowerDesigner.

2.3.8.3 Tracing Trigger and Procedure Dependencies

When you write a trigger or procedure, PowerDesigner automatically creates dependencies to any table, view, procedure, or database package referenced in the code. These dependencies are taken into account when performing an impact analysis prior to deleting the trigger or procedure or objects on which they depend. For procedures, if the procedure has a symbol in your diagram, then any dependencies will be shown graphically by way of arrows linking the procedure to these objects.

The diagram below shows a procedure, ProcChangeLocation, which is dependent on a number of other objects:
Its Traceability Links tab lists the objects upon which it depends, and the link type of `DBCreateAfter (computed)` shows that PowerDesigner has determined that it can only be created after these objects:

- Employee table
- ProcChangeLocation

The Employee table Dependencies tab shows that ProcChangeLocation is dependent upon it, and if you were to perform an impact analysis prior to deleting the Employee table, you would be warned of the procedure's dependency on it.

In this section:
- Creating Procedure Dependencies Manually
- Rebuilding Trigger and Procedure Dependencies

### 2.3.8.3.1 Creating Procedure Dependencies Manually

Since procedures have diagram symbols, you can manually add dependencies for them using the Traceability Links tool in the toolbox.

In the diagram below, ProcChangeLocation has a dependency on a new procedure, ProcOccupancy:
Since ProcOccupancy is not directly referenced in ProcChangeLocation, you must manually set the type of the link to DBCreateAfter on the Traceability Links tab of the ProcChangeLocation property sheet:

### 2.3.8.3.2 Rebuilding Trigger and Procedure Dependencies

Trigger and procedure dependencies are rebuilt automatically after the following actions:

**Context**

- Importing a PDM created with a former version of PowerDesigner
- Reverse engineering a database into a PDM
- Merging PDMs

You can also manually rebuild trigger and procedure dependencies at any time.

**Procedure**

1. Select **Tools** > **Rebuild Objects** > **Rebuild Triggers and Procedure Dependencies** to open the Procedures Dependencies window.
2. Specify a rebuild mode for each of Procedures and Triggers. You can choose between the following options:
   - **Delete and Rebuild** – all triggers and/or procedures attached to templates are deleted and rebuilt, including those to which you have made modifications
   - **Preserve** – only those triggers and/or procedures attached to templates that have not been modified are deleted and rebuilt. Any triggers and/or procedures that you have modified are preserved.
3. [optional] Click the Selection tab and specify the tables, views, procedures, and (for Oracle only) database packages for which you want to rebuild dependencies. By default all are selected.
4. Click OK to begin the rebuild process.

2.3.8.4 Attaching a Stored Procedure to a Table

You can attach a stored procedure to a table when your current DBMS supports stored procedures. This feature lets you update the table or retrieve information from this table.

Context

For example, the stored procedure TABLE_ADDROW can be attached to a table in which you need to insert rows.

When you generate an OOM from a PDM, the procedures attached to a table become operations with the <<procedure>> stereotype in the generated class. By attaching procedures to tables, you are able to define class operations in the generated OOM.

When you generate a PDM from an OOM, class operations with the <<procedure>> stereotype become stored procedures attached to the generated table. The operation body is generated as a comment in the procedure definition.

You can attach a table to a procedure from the property sheet of a procedure or the property sheet of a table.

Procedure

1. Open the table property sheet and click the Procedures tab.
2. Click the Add Objects tool to open a selection box, choose the stored procedure you want to attach to the table and click OK.
   The stored procedure is displayed in the list of stored procedures.
3. Click OK.

In this section:

- Rebuilding Procedures Attached to Tables

2.3.8.4.1 Rebuilding Procedures Attached to Tables

You can rebuild procedures attached to tables at any time.
Procedure

1. Select Tools > Rebuild Objects > Rebuild Table Stored Procedures to open the Rebuild Table Stored Procedures window.

2. Specify a rebuild mode. You can choose between the following options:
   - Delete and Rebuild – all procedures attached to tables are deleted and rebuilt
   - Add missing table stored procedures – adds procedures to any selected tables that do not presently have them.

3. [optional] Click the Selection tab to specify for which tables you want to rebuild stored procedures.

4. Click OK to begin the rebuild process.

2.3.8.5 Procedure Templates (PDM)

PowerDesigner procedure templates allow you to write table procedures in a modular reusable fashion. We provide basic templates for insert, select, update, and delete procedures. You can modify the code specified in these templates or create your own templates in the DBMS definition file.

To apply a procedure template to your procedure definition, select the template from the list on the procedure property sheet Definition tab (see Procedure Properties).

To review or modify the provided procedure templates, select Database > Edit Current DBMS, and then click the Procedure Templates tab. You cannot delete or rename these templates.

Caution

The resource files provided with PowerDesigner inside the Program Files folder cannot be modified directly. To create a copy for editing, use the New tool on the resource file list, and save it in another location. To include resource files from different locations for use in your models, use the Path tool on the resource file list.

To create a new template, click the Add a Row tool.

Procedure Template Properties

The General tab contains the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Code/Comment</td>
<td>Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.</td>
</tr>
<tr>
<td>DBMS</td>
<td>Specifies the parent DBMS.</td>
</tr>
<tr>
<td>Function</td>
<td>Specifies whether the template defines procedures or functions.</td>
</tr>
<tr>
<td>Procedure Name</td>
<td>Specifies the conventions for naming procedures based on the template.</td>
</tr>
<tr>
<td>Linked to table</td>
<td>Specifies whether the resulting procedure will be linked to a table.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.</td>
</tr>
</tbody>
</table>

The following tabs are also available:

- Definition - Contains a field for entering the procedure code for the template. You can use PDM variables and macros and other tools available from the toolbar (see Writing SQL Code in PowerDesigner).

2.3.9 Users, Groups, and Roles (PDM)

A user is a database object that identifies a person who can login or connect to the database. Groups and roles are used to simplify the granting of rights to users, as privileges and permissions granted to a group or role are inherited by users who belong to that group or incarnate that role.

Not all DBMSs support each of the concepts of user, role, and group.
2.3.9.1 Creating a User, Group, or Role

You can create a user, group, or role from the Browser or Model menu. You can also create a user from the Owner field of various objects.

- Select Model > Users and Roles to access the appropriate model object list, and click the Add a Row tool.
- Right-click the model (or a package) in the Browser, and select New <Type>.
- ([users]) Click the Create tool to the right of the Owner field on the General tab of a table (see Table Properties) or other object that allows you to specify an owner.

For general information about creating objects, see Core Features Guide > Modeling with PowerDesigner > Objects.

2.3.9.2 User, Group, and Role Properties

To view or edit a user, group, or role's properties, double-click its Browser or list entry. The property sheet tabs and fields listed here are those available by default, before any customization of the interface by you or an administrator.

The General tab contains the following properties:

- Name/Code/Comment: Identify the object. The name should clearly convey the object's purpose to non-technical users, while the code, which is used for generating code or scripts, may be abbreviated, and should not normally include spaces. You can optionally add a comment to provide more detailed information about the object. By default the code is generated from the name by applying the naming conventions specified in the model options. To decouple name-code synchronization, click to release the = button to the right of the Code field.
- Stereotype: Extends the semantics of the object. You can enter a stereotype directly in this field, or add stereotypes to the list by specifying them in an extension file.
- Password: [users and groups] Password used for database connection.
- Keywords: Provide a way of loosely grouping objects through tagging. To enter multiple keywords, separate them with commas.

The following tabs are also available:

- Privileges - lists the system privileges granted to the user (see Granting System Privileges).
- Permissions - lists the operations that the user is permitted to perform on various database objects (see Granting Object Permissions).
- Users - [groups and roles] Lists the users belonging to the group or role.
- Groups - [groups and roles] Lists the groups belonging to the group or role.
- Roles - [roles] Lists the roles belonging to the role.

2.3.9.3 Assigning an Owner to an Object

The database user who creates an object (table, view, stored procedure, etc) is the owner of the object and is automatically granted all permissions on it. In a PDM, you can specify the owner of an object by attaching a user to it. Each object can have only one owner. Where other users must access the object, you can restrict object modifications to the owner and grant Select or other permissions for the other users.

Context

- Note: To automatically assign a default owner for any type of object that supports the concept of ownership, select Tools > Model Options, choose the appropriate object type in the left-hand pane, and select the appropriate user in the Default owner field (see Other Object Model Options).

Procedure

1. Open the property sheet of the object to the General tab.
2. Select a user in the Owner field. To create a new user, click the Create tool to the right of this field.
3. Click OK to return to your model.

Note: When generating to your database (see Generating a Database from a PDM), you can restrict the tables and other objects generated to only those belonging to a particular owner, by selecting the owner on the Database Generation dialog Selection tab.
2.3.9.4 Granting System Privileges

System privileges are granted to users, groups, and roles to give them the right to perform particular types of action in the database. By default, a user belonging to a group or having a role inherits the group or role privileges and these inherited privileges are identified as such in the Privileges tab of the user property sheet. A user with an administrative profile is also allowed to revoke a privilege.

Context

System privileges are used in association with object permissions (see Granting Object Permissions) to evaluate the rights of a user, group, or role. For example, even if a user has the Modify privilege, he cannot modify an object on which he has no Update permission.

Note

In some DBMSs, system privileges are called permissions. In PowerDesigner, the term privilege is reserved for any right granted to a user, a group, or a role. Permissions are defined for objects.

Procedure

1. Open the property sheet of a user, role, or group, and click the Privileges tab.
2. [optional] Click the Show/Hide All Inherited Privileges tool to show privileges that have been inherited from a group. Inherited privileges are red, while privileges directly granted to the user are blue.
3. Click the Add Objects tool to choose one or more of the privileges available in the DBMS, and click OK to grant them to the user, role, or group. System privileges are defined in the DBMS definition file. To review and edit the list of available privileges, select Database > Edit Current DBMS > Script > Objects > Privilege > System, and edit the list as appropriate. The Privilege category also contains entries that define the syntax for the necessary SQL statements for granting and revoking privileges. For more information, see Customizing and Extending PowerDesigner > DBMS Definition Files > Script/Objects Category.
4. [optional] To change the state of a privilege (whether granted directly, or inherited from a group), click the State column to cycle through the available states, or click on the appropriate tools in the Privilege state group box at the bottom of the tab.

Privilege Description

- Grant – [default] Assigns the privilege to the user.
- Inherited/None – Reverts the cell to the inherited state.
- Revoke – Revokes the privilege inherited from a group or role for the current user or group.
- Grant with admin option – Assigns the privilege to the user, and allows the recipient to pass on the privilege to other users, groups, or roles. For example, you assign the CREATE TABLE privilege for user Designer_1 and then click the Grant With Admin Option button to permit Designer_1 to grant this privilege to other users.

5. When the privileges are correct, click OK to return to the model.

In this section:

- Generating Privileges

2.3.9.4.1 Generating Privileges

You can generate privileges to a script or to a live database connection.

Procedure
1. Select Database Generate Database to open the Database Generation window, and specify the standard options, including whether you want to generate to a script or to a live database connection. For detailed information about using this window, see Generating a Database from a PDM.

2. Select "Users & Groups (with privileges)" from the Settings set list in the Quick Launch groupbox at the bottom of the window. This settings set specifies standard options for generating privileges.
   
or:
   Click the Options tab and click on User in the left-hand pane to display the user generation options. Change the default options as appropriate. For detailed information about settings sets, see Quick Launch Selection and Settings Sets.

3. [optional] Click the Selection tab and select the Users sub-tab at the bottom of the tab. Select the users that you want to generate for.

4. Click OK to begin the generation.