

Microolap Database Designer for PostgreSQL

User's guide

Contents

1. Introduction.....	1
2. Features.....	2
3. Other MicroOLAP Products.....	4
4. Installation.....	5
5. Interface User Guide.....	5
5.1. Diagram Window.....	7
5.2. Toolbars.....	7
5.3. Docking Windows.....	10
5.4. Keyboard Shortcuts.....	11
5.5. Environment Options.....	13
6. Diagram.....	20
6.1. Diagram notation.....	21
6.2. Creating a New Diagram.....	21
6.3. Open an Existing Diagram.....	22
6.4. Saving a Diagram.....	22
6.5. Selecting and Moving Objects.....	22
6.6. Copying and Pasting Objects.....	23
6.7. Using Snapping Grid.....	24
6.8. Export to Graphics.....	24
6.9. Diagram Display Preferences.....	25
6.10. Diagram Properties.....	32
6.10.1. Default Database Options.....	34
6.10.2. Diagram Naming.....	36
6.10.3. Diagram Pages.....	37
6.10.4. Diagram SQL Preview.....	37
6.10.5. Diagram Notes.....	37
6.10.6. Diagram Statistics.....	38
6.11. Zooming a Diagram.....	38
6.12. Auto Layout Diagram.....	38
6.13. Find Objects.....	39
7. Diagram Objects.....	41
7.1. Database.....	42
7.1.1. Database Editor.....	42
7.2. Tables.....	45
7.2.1. Creating a Table.....	45
7.2.2. Table Editor.....	46
7.2.3. Columns.....	50
7.2.3.1. Column Editor.....	50
7.2.3.2. Column Manager.....	52
7.2.4. Constraints.....	54

7.2.4.1. Constraint Editor.....	55
7.2.4.2. Constraint Manager.....	57
7.2.5. Indexes	59
7.2.5.1. Index Editor.....	59
7.2.5.2. Index Manager.....	62
7.2.6. Triggers	63
7.2.6.1. Trigger Editor.....	64
7.2.7. Rules	65
7.2.7.1. Rule Editor.....	66
7.2.8. Formatting Table	67
7.2.9. Storage Parameters	68
7.2.10. SQL Table Definition	70
7.2.11. Tables in Tree View Window	70
7.2.12. Table Manager	71
7.3. Stored Procedures and Functions.....	72
7.3.1. Creating a Stored Procedure or Function	72
7.3.2. Stored Routine Editor	73
7.3.3. Stored Routine Manager	80
7.4. Views.....	82
7.4.1. Creating a View	82
7.4.2. View Editor	82
7.4.3. View Manager	85
7.5. Schemas.....	86
7.5.1. Schema Manager	86
7.6. Domains & User defined Types.....	87
7.6.1. Domain & User Defined Type Manager	88
7.7. References and Foreign Keys.....	94
7.7.1. Creating a Reference	94
7.7.2. Creating a Many-to-Many Reference	95
7.7.3. Reference Editor	95
7.7.4. Reference Manager	98
7.7.5. Manual Reference Drawing	99
7.8. Roles.....	101
7.8.1. Roles Manager	101
7.9. Tablespaces.....	104
7.9.1. Tablespaces Manager	105
7.10. Sequences.....	106
7.10.1. Sequences Manager	106
7.11. Privileges.....	108
7.11.1. ACL Manager	108
7.12. Notes.....	110
7.13. Stamps.....	111

8. Diagram Functions.....	112
8.1. Check Diagram.....	112
8.2. Merge Diagram.....	115
8.3. Repair Diagram.....	115
9. Database Functions.....	116
9.1. New Database Wizard.....	116
9.2. Database Generation.....	120
9.3. Database Modification.....	129
10. Database Accessing Tools.....	132
10.1. Database Connection Manager.....	132
10.1.1. Connection Profile Editor	134
10.2. SQL Executor.....	136
10.3. Connect to a Database.....	137
10.4. Disconnect from a Database.....	137
11. Reverse Engineering And Import.....	137
11.1. Reverse Engineering PostgreSQL Database.....	138
11.2. Import from Access Database.....	140
11.3. Universal Reverse Engineering.....	141
11.4. SQL Reverse Engineering.....	145
12. Reports.....	147
12.1. Create a Report.....	148
13. Printing a Diagram.....	149
13.1. Page Setup.....	149
13.2. Print Preview.....	153
14. How to.....	154
14.1. How to Connect to a Database.....	155
14.2. How to Disconnect from a Database.....	155
14.3. How to Create a New Diagram.....	155
14.4. How to Create a Table.....	156
14.5. How to Create a Reference and Foreign Key.....	156
14.6. How to Check a Diagram.....	157
14.7. How to Edit Columns	160
14.8. How to Edit Table Index.....	162
14.9. How to Generate a Database.....	165
14.10. How to Modify a Database.....	174
14.11. How to Import from a PostgreSQL Database.....	177
14.12. How to Import from a Microsoft Access Database.....	179
14.13. How to Import from an SQL Script.....	180
14.14. How to Import from other Databases.....	182
14.15. How to Modify Multiple Tables.....	186
14.16. How to Execute an SQL Script.....	188
14.17. How to Merge Diagrams	189

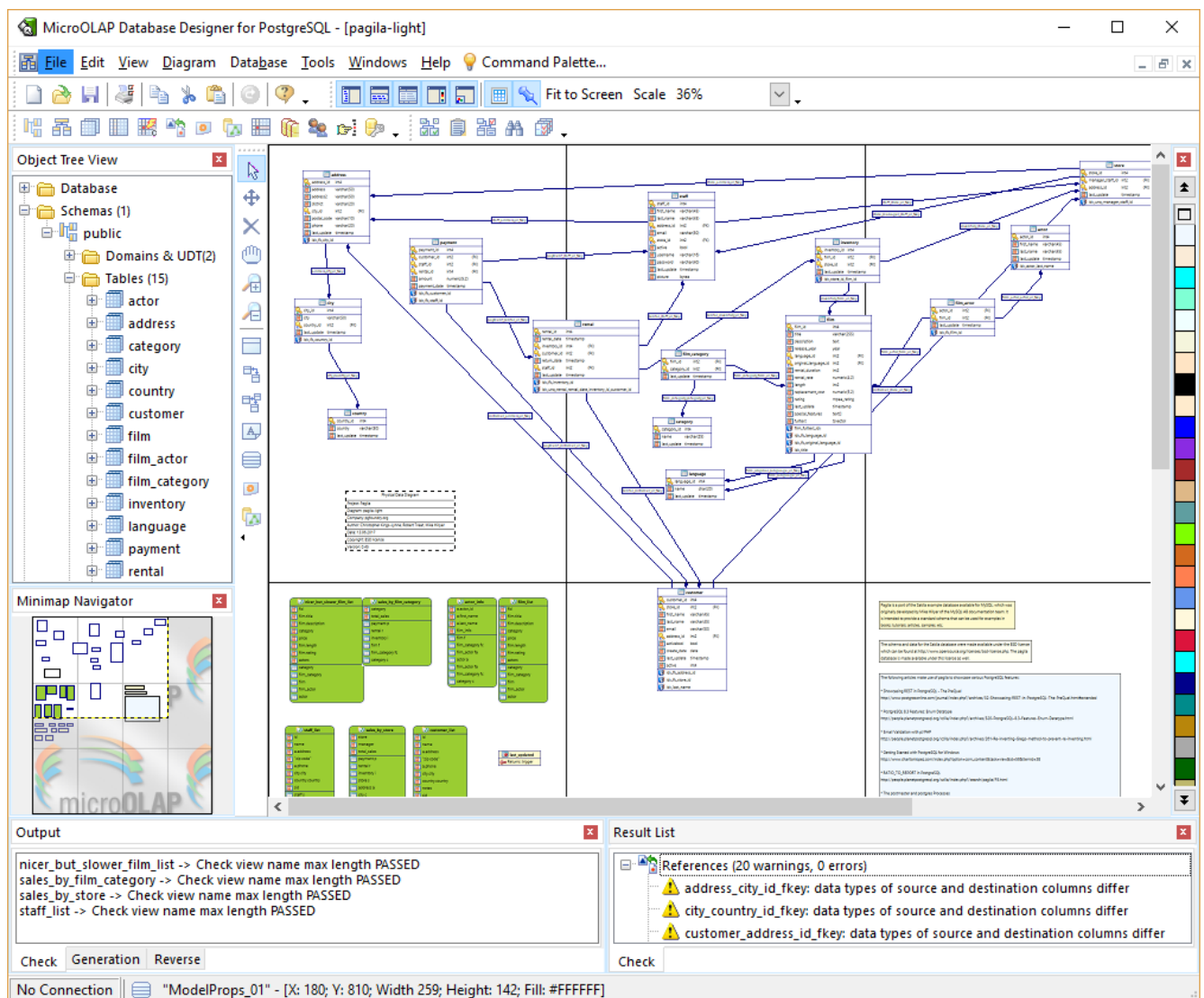
14.18. How to Find Errors in a Diagram.....	189
14.19. How to Find Objects	192
14.20. How to Print a Diagram.....	195
14.21. How to Read a Diagram.....	195
14.22. How to View an SQL Table Definition.....	196
14.23. How to Export a Diagram to Graphics.....	197
14.24. How to Create a Domain.....	197
14.25. How to Insert a Comment.....	202
14.26. How to Change Table Formatting.....	203
15. Wine Configuration.....	204
16. Purchase.....	205
17. Support.....	205
18. License Agreement.....	205

1. Introduction

Welcome to Database Designer for PostgreSQL

Database Designer for PostgreSQL is a powerful solution for visual database development: creation, modification and reverse engineering. It greatly simplifies the process of database development and maintenance and makes it clear. By working with graphical representation of tables, columns, relations and other objects, you can build a database model that can be used to generate a physical database or to modify an existing one. The model can be automatically created from an existing database by reverse engineering.

Database Designer for PostgreSQL is specially developed for PostgreSQL database and takes into account its features, has build-in support of schemas, tables, triggers, references, stored procedures, domains, indexes, foreign keys and other database objects.



The product allows to extract tables, attributes, relationships, indexes and other objects from existing databases including PostgreSQL, Microsoft Access and Sybase ASE/ASA, Oracle, Informix, MSSQL, DB2, DBF and many others that accessible through OLEDB or ODBC.

You can generate reports with complete information about a developed model in printer-friendly HTML view.

2. Features

Database Designer for PostgreSQL offers you many features that allow you to be creative and productive in your work on database design and maintenance:

PostgreSQL Native Support

Database Designer for PostgreSQL provides native support for PostgreSQL objects and data types. You can work with them in a very intuitive graphical interface. The central part of the application is a diagram that gives you a visual representation of the PostgreSQL database.

After you have completed designing the database diagram you can generate physical database directly on PostgreSQL server and even modify the already existing local or remote database. Please note that you can start working with the diagram by reverse engineering an existing PostgreSQL database (see [Diagram overview](#), [Main Window Organization](#), [Database Generation](#), [Reverse Engineering and Import overview](#)).

Easy Object Location and Modification

You can easily navigate through the diagram objects by using [Object Tree View window](#). It contains combined information about all objects of your diagram and displays the diagram in a hierarchical tree structure. Object Tree View window shows you: Domains, Tables, Columns, Indexes, References, Roles and Tablespaces. To locate any diagram table quickly, right-click on it in the Object Tree View window and select Go to object menu item from the context menu.

Check diagram Function

Database Designer for PostgreSQL can check your diagram for consistency and find typical errors, which may occur during the process of database design and maintenance. In terms of convenience, this feature can be compared to Microsoft Word's spell checker. It saves your time greatly and gives you the opportunity to concentrate on the main development tasks (see [Check Diagram function](#)).

Import From Multiple Data Sources

You can generate your diagram from an existing database structure. It's possible by reverse engineering a PostgreSQL database or importing a structure from other data sources.

Database Designer for PostgreSQL allows you to import database objects from PostgreSQL database, Microsoft Access, ADO (OLEDB)-compatible databases and SQL script (see [Reverse Engineering and Import Overview](#)).

Report Generator

Database Designer for PostgreSQL can generate comprehensive printable reports for you. This gives you the possibility of getting a hard copy of the report by printing it out. After that you can have your diagram approved by your colleagues and managers.

Generated reports contain information about all model objects.

You can get reports in HTML format (see [Reports overview](#)).

SQL Editor and Executor

With **Database Designer for PostgreSQL** you do not need any external tools for executing simple SQL statements and queries. The SQL Executor tool allows you to modify the existing structure of a local or remote database and even view data by running SELECT/INSERT/UPDATE/DELETE SQL statements (see [SQL Executor](#)).

Full Customization of Diagram and Objects Appearance

The appearance of the diagram can be adjusted according to your requirements and wishes. You can set the colors of the table background, table lines and references. You can configure the representation of the table information: enable showing of column and indexes icons, displaying indexes, keys, foreign keys and other ([Diagram Overview](#), [Diagram Display Preferences](#)).

Multi-document Interface

Database Designer for PostgreSQL has Multi-Document Interface. This means that you can edit multiple documents simultaneously. You can place more than one diagram onto your work space, switch between them quickly, compare them, and drag objects from one to another (see [Diagram Window](#)).

Export of diagram to Graphical File

The product can export your diagram into multiple graphical formats: bitmap (PNG, GIF, JPEG) and vector (EMF) by using **File | Export...** menu item. This allows you to use graphical representation of a diagram in your documents (i.e. use it in Microsoft Word, LaTeX), publish diagram to Web sites, create big posters with it (see [Export to Graphics](#)).

Database Generation

When you have finished creating the database diagram, you get the possibility of generating SQL script for it (**Database | Generate Database**) and you can build the database on a local or remote PostgreSQL server directly (see [Database Generation](#)).

Database Synchronization

The Synchronize Database (**Database | Modify Database...**) tool helps you not only generate a new database but, what is more important, synchronize the existing PostgreSQL database with the

diagram you are developing. **Database Designer for PostgreSQL** will automatically generate all required SQL statements. Which means that you do not have to create complex and inconvenient ALTER TABLE statements manually; moreover it significantly reduces the complexity and improves the reliability of all database modification tasks. Whatever complex diagram you have, you can synchronize the database with it literally in two clicks (see [Database Synchronization overview](#)).

Database Connection Manager

You can store the connection profiles for your database within the Database Connection Manager and access any of them quickly. You do not have to remember multiple parameters of connections — just click on the connection profile and get access to the database (see [Database Connection Manager overview](#)).

3. Other MicroOLAP Products

Database tools

PaGoDump

GUI tool for extracting a PostgreSQL database into SQL script file, archived SQL file (GZIP), TAR archive, or pg_restore custom archive (.backup).*

<http://microolap.com/products/database/pagodump/>

PaGoDump is a GUI Windows utility for backing up a PostgreSQL database built with Microolap PostgresDAC. It makes consistent backups even if the database is being used concurrently. PaGoDump does not block other users accessing the database (readers or writers), also it works with databases with any names (unicode) and dump them to any files (unicode again). Dumps can be output in script or archive file formats.

Database Designer for MySQL

Powerful MySQL-oriented database structure modeling, generation and modification

<http://www.microolap.com/products/database/mysql-designer/>

Database Designer for MySQL is a handy tool with intuitive interface. It allows you to build a clear and effective database structure visually, see the complete picture (diagram) representing all the tables, references between them, views, stored procedures within your database. Then you can easily generate a physical database on a server, modify it according to any changes you made to the diagram. The product has been developed specially for MySQL database taking into account all its features.

Database-related components

DAC for MySQL

<http://www.microolap.com/products/connectivity/mysqldac/>

DAC for MySQL (also known as **MySQLDAC**) is the most powerful component suite for Delphi/C++Builder/ MySQL. It allows you to create Delphi/BCB applications with direct access to MySQL DB without BDE/ODBC. No (even **libmysql.dll**) MySQL libraries are required.

PostgresDAC

<http://www.microolap.com/products/connectivity/postgresdac/>

PostgresDAC is the most powerful component suite for Delphi/C++Builder/MySQL. It allows you to create Delphi/BCB applications with direct access to MySQL DB without BDE/ODBC.

4. Installation

To install **Database Designer for PostgreSQL** on your PC:

- Unzip the downloaded file to any location you prefer;
- Run the .exe file and follow the installation application instructions.

The unregistered version of **Database Designer for PostgreSQL** (30-days trial) has the same features as the registered one.

 Before installation of **Database Designer for PostgreSQL**, please check for updates at microOLAP.com

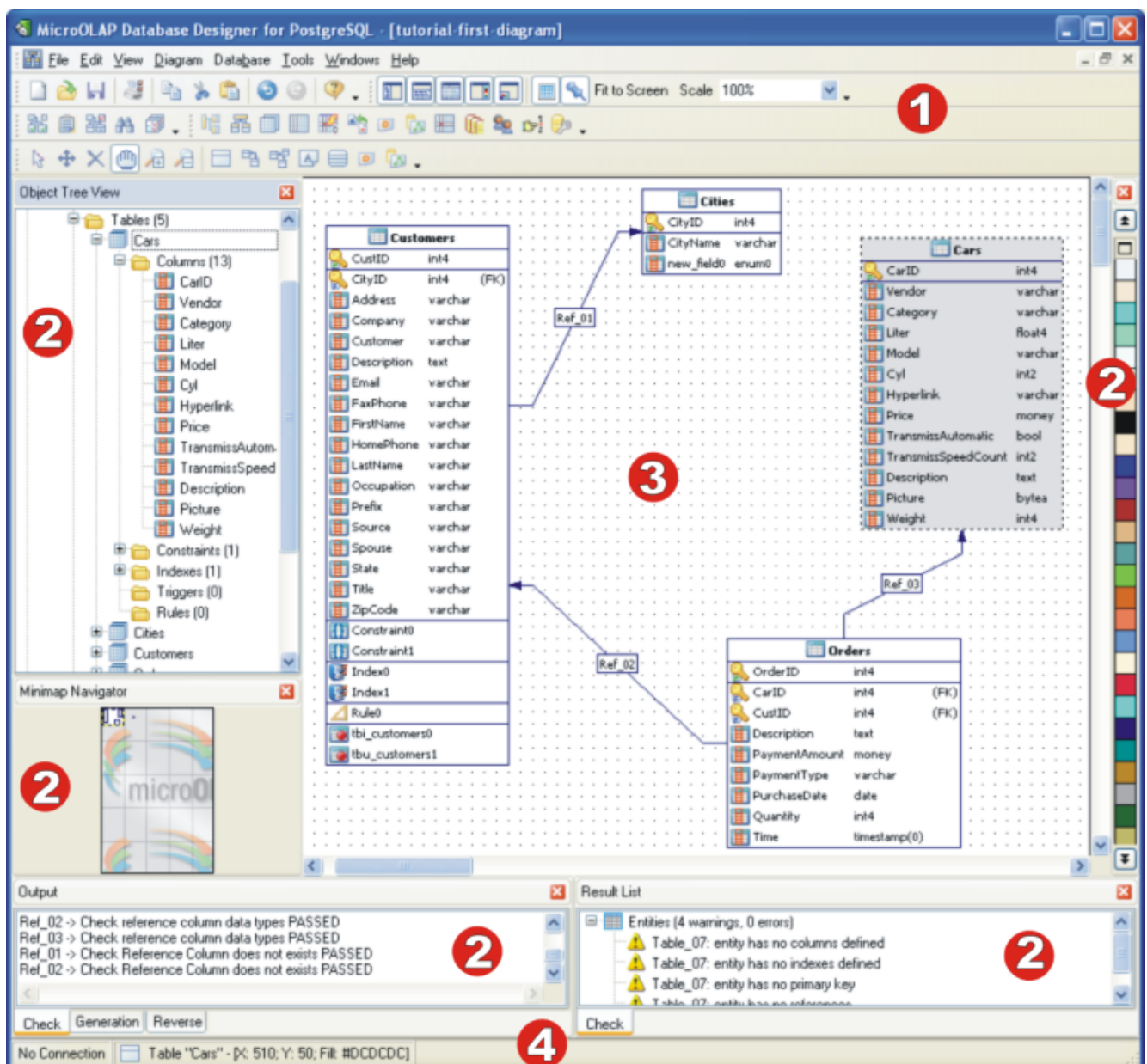
See also:

[Purchase](#)

5. Interface User Guide

The **Database Designer for PostgreSQL** user interface has been designed to provide you with convenience and flexibility in performing database designing tasks. You can easily organize various Tools and Windows to suit your preferences.

There are 4 main areas in the **Database Designer for PostgreSQL** user interface. They are illustrated in the following screenshot:



1. Menubars and [Toolbars](#)

The menubars and toolbars provide you with access to the **Database Designer for PostgreSQL** functions. The position and visibility of bars and tools can be customized.

2. [Docking Windows](#)

Docking windows are portions of the **Database Designer for PostgreSQL** interface containing specific functionality, such as the **Tree View Window**, **Minimap Navigator**, **Result List**, **Output** and **Color Palette**. Docking windows can be **Docked**, **Floating** and **Tab-Docked**.

3. [Diagram Window](#)

This area contains the database diagram, which is currently being edited. You can work with multiple diagrams at the same time.

4. Connection Statusbar

The Connection Statusbar shows you the information about database connection properties of the currently active diagram.

See also:

Diagram Objects: [Diagram Overview](#)

Diagram: [Notation](#)

Workspace: [Diagram Window](#) | [Toolbars](#) | [Docking Windows](#) | [Keyboard Shortcuts](#)

5.1. Diagram Window

The diagram window is the main working area of the application. It displays the active diagram content. You can open multiple diagrams and switch between them using Window submenu. Also you can dispose a number of diagrams at the working area. Please note that more than one diagram can be active.

To change diagrams disposition in the working area automatically, use the following menu items:

Window | Cascade

Window | Tile Horizontal

Window | Tile Vertical




To close all opened diagrams, use the **Window | Close All** menu item.








See also:

Diagram: [Creating a New Diagram](#) | [Notation](#) | [Export to Graphics](#)









5.2. Toolbars

Standard toolbar

Toolbar button	Shortcut	Function
	<i>Ctrl+N</i>	Create a new diagram
	<i>Ctrl+O</i>	Open an existing diagram
	<i>Ctrl+S</i>	Save this diagram

	<i>Ctrl+P</i>	Print this diagram
	<i>Ctrl+C</i>	Copy an object
	<i>Ctrl+V</i>	Paste an object
	<i>Ctrl+X</i>	Cut an object
	<i>Ctrl+Z</i>	Undo
	<i>Ctrl+Shift+Z</i>	Redo
	F1	Show help

Palette toolbar

Toolbar button	Shortcut	Function
		Pointer tool
		Move tool
		Delete object tool
		Hand tool
	<i>F6</i>	Zoom In
	<i>F7</i>	Zoom Out
		Create table tool
		Create a reference















		Create a many-to-many reference
		Create a text note
		Create a stamp
		Create a stored procedure or function

Diagram toolbar

Toolbar button	Shortcut	Function
	<i>F4</i>	Check diagram
		Create report
		Diagram properties
	<i>Ctrl-F</i>	Find objects
	<i>Alt-Enter</i>	Object properties

View toolbar

Toolbar button	Shortcut	Function
	<i>Alt-0</i>	Show/hide Object Tree View window
	<i>Alt-1</i>	Show/hide Output window
	<i>Alt-2</i>	Show/hide Result List window

	<i>Alt-3</i>	Show/hide Color Palette window
		Show grid
		Snap to grid
		Fit to screen
		Scale diagram

See also:

Workspace: [Main Window Organization](#)

5.3. Docking Windows

Database Designer for PostgreSQL user interface contains several Docking Windows. The product functionality is grouped in various docking windows.

In this topic you will find an overview of each Docking Window. Please see the detailed description below.

Object Tree View Window

The **Diagram Tree View** contains an organized view of the diagram tables, columns, constraints, indexes, triggers, rules, stored procedures/functions, User Defined Types (UDT), references, roles and tablespaces.

You can use the Tree View to:

- Browse through the diagram objects
- Search and locate the objects on the diagram
- Call the appropriate properties editor for the object

Minimap Navigator

The **Minimap Navigator** is intended to help you move quickly around a drawing.

Output Window

Output window contains run-time information, which is generated by such tools as [Check Diagram](#), [Generate Database](#) and [Reverse Engineer database](#).


Output window consists of several tabs (**Check, Generation, Reverse**), each of which is intended for representing the information for an appropriate tool.

Result List Window

Result List window is intended for showing the result of [Check Diagram](#) process.

Color Palette

Color Palette is intended for quick setting of the color to visual objects. Left click will set fill color for an object, right click will set line color. If several objects selected, color setting will be applied to all of them.

 You may reorder colors positions by dragging color boxes. Middle click will open **Custom color** dialog, so you can change any color in the palette to whatever you like. All these changes will remain after closing the **Database Designer for PostgreSQL**.

See also:

Workspace: [Main Window Organization](#)

5.4. Keyboard Shortcuts

This topic describes keyboard shortcuts, that allow you to access **Database Designer for PostgreSQL** functionality quickly.

Key	Function
<i>Ctrl-N</i>	Create a new diagram
<i>Ctrl-O</i>	Open an existing diagram
<i>Ctrl-S</i>	Save this diagram
<i>Ctrl-P</i>	Print this diagram

<i>Ctrl-F4</i>	Close this diagram
<i>Alt-F4</i>	Close the application
<i>Ctrl-Z</i>	Undo
<i>Ctrl-Shift-Z</i>	Redo
<i>Ctrl-C</i>	Clipboard Copy
<i>Ctrl-X</i>	Clipboard Cut
<i>Ctrl-V</i>	Clipboard Paste
<i>Ctrl-F</i>	Find diagram objects
<i>Ctrl-A</i>	Select all objects
<i>Shift-F5</i>	Redisplay diagram
<i>Alt-0</i>	Show/hide Object Tree View window
<i>Alt-1</i>	Show/hide Output window
<i>Alt-2</i>	Show/hide Result window
<i>Alt-3</i>	Show/hide Color Palette window
<i>Alt-4</i>	Show/hide Minimap Navigator window
<i>F5</i>	Set diagram actual size
<i>F6</i>	Zoom In diagram
<i>F7</i>	Zoom Out diagram
<i>F4</i>	Check Diagram

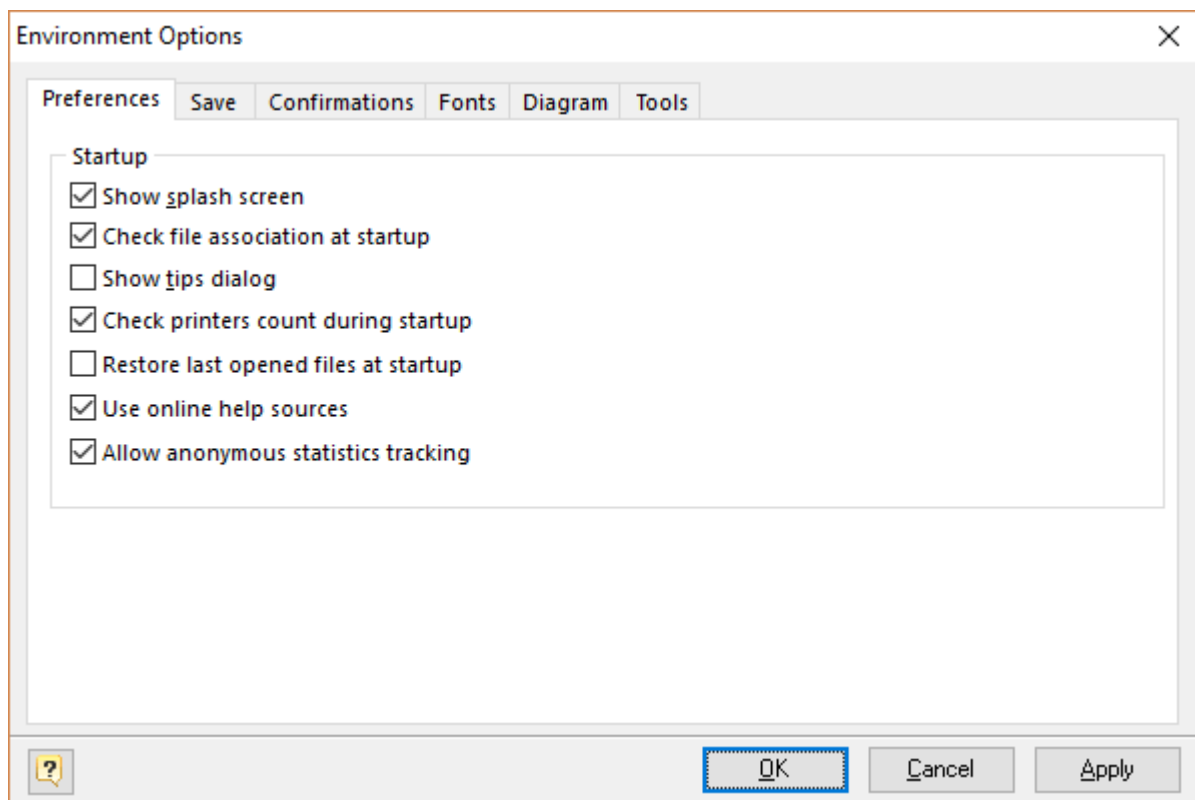
<i>Shift-F6</i>	Merge Diagram
<i>Ctrl-G</i>	Generate database
<i>Ctrl-R</i>	Reverse engineer PostgreSQL database
<i>Ctrl-M</i>	Database Modification
<i>Shift-Ctrl-E</i>	Execute SQL statement
<i>Shift-Ctrl-N</i>	Connect to database
<i>F1</i>	Show help

5.5. Environment Options

Environment Options allow you to customize general **Database Designer for PostgreSQL** options. To open the **Environment Options** dialog window select the **Tools | Environment Options** menu item.

The Environment Options dialog window consists of several tabs. Please see the detailed description below:

Preferences



In this tab you can adjust the standard behavior of the application on the startup or during the creation of new objects.

Show splash screen

This option enables you to view **Database Designer for PostgreSQL** welcome screen at the application startup.

Check file association at startup

This option enables you to check if the diagram files are associated with **Database Designer for PostgreSQL** at the program startup.

Show tips dialog

This option enables you to view Tips Of the Day dialog window with useful recommendations on the application startup.

Check printers count during startup

Enables reading of the printer parameters at the application startup. If you want to check the parameters, the startup may slow down.

Restore last opened files at startup

This option enables the opening of model files which you were working with, when Designer was closed last time.

Use online help sources

Enable this option to use online manual opened in the preferred browser instead of local .CHM manual.

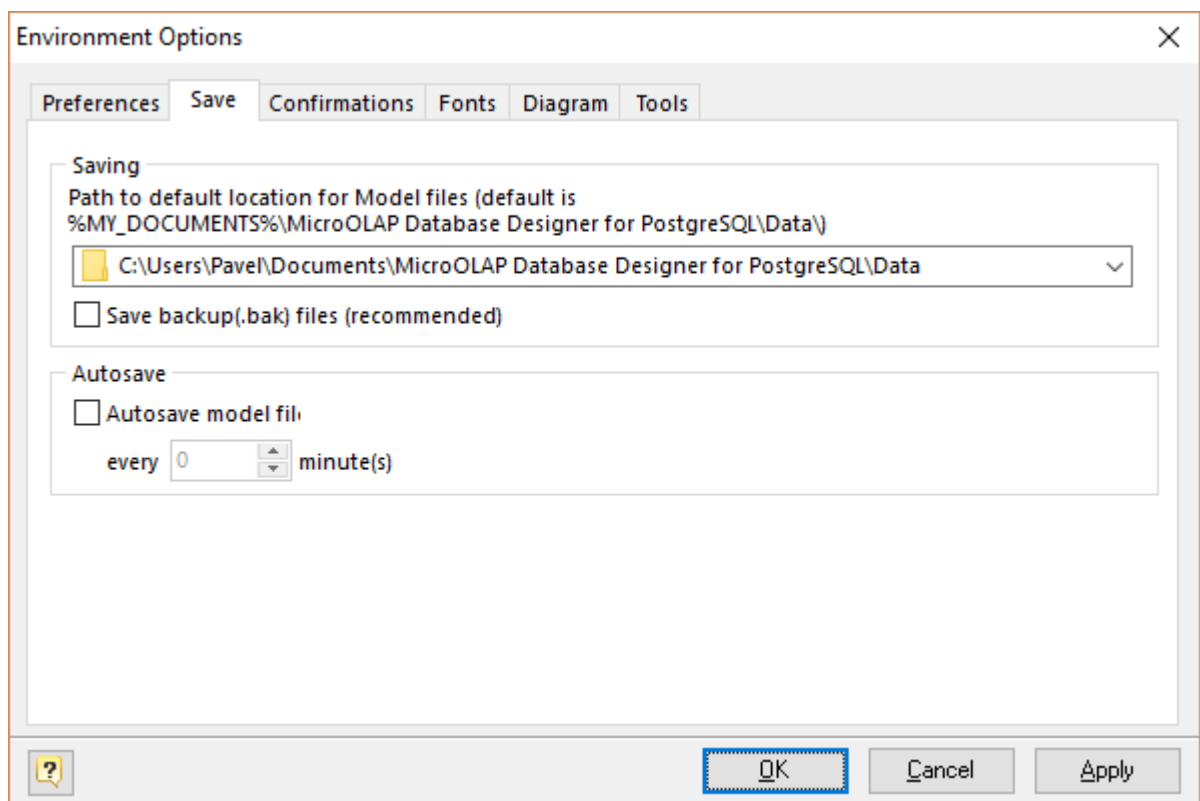
Enable undo

This option enables you to undo and redo functions of the application.

Undo limit

This option sets the limit of undo steps. The more the limit is set, the more memory is needed. The default value is set to 99.

Save



In this tab you can change Designer behavior in Save operations.

Path to default location for Model files

Set this option to change default path for Save and Open commands.

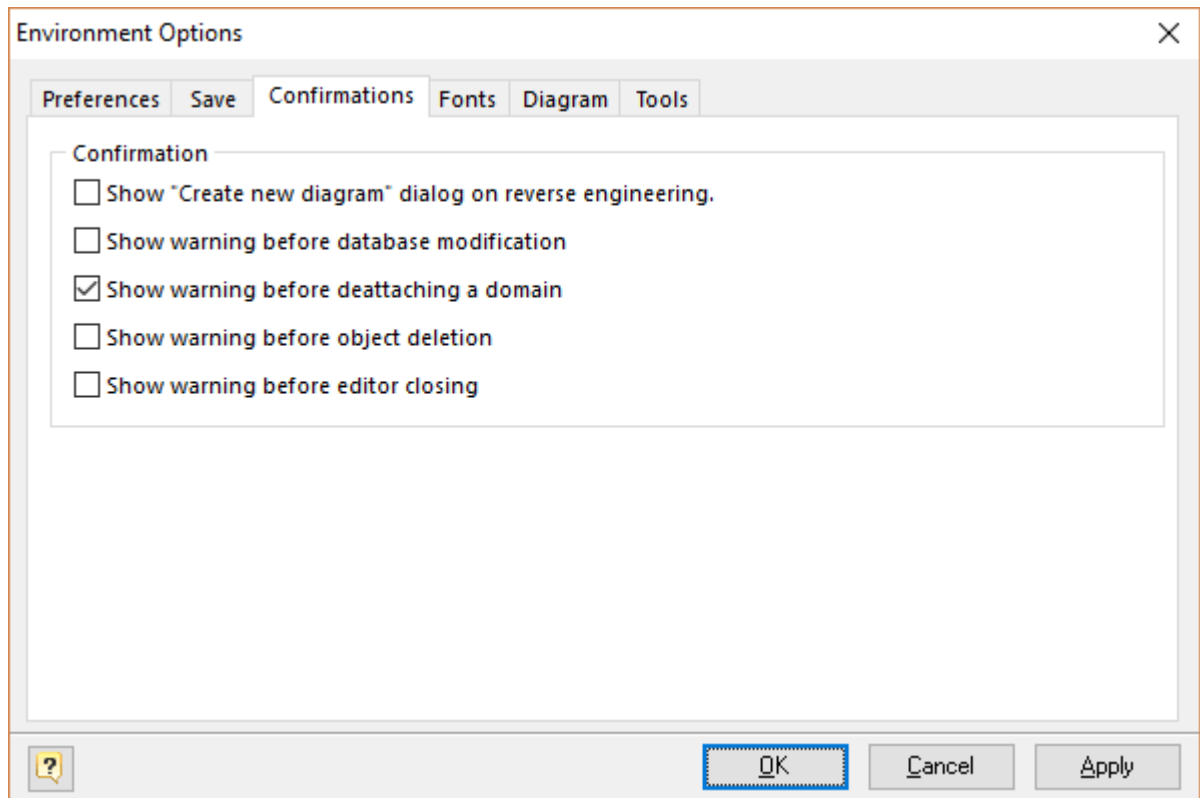
Save backup files (.bak)

This option enables the creation of backup files for the diagram.

Autosave model file

This option enables the scheduled model saving in the background.

Confirmations



This tab allows to enable or disable the confirmations, which pop up before some operations.

Show "Create new diagram" dialog box on reverse engineering

This option opens the dialog window that suggests you create a new diagram before [Reverse Engineering and Import](#).

Show warning before database modification

This option opens the dialog window, that shows warning before [Database Modification](#).

Show warning before deattaching a domain

This option opens the dialog window, that shows warning before modification of a domain-based table column.

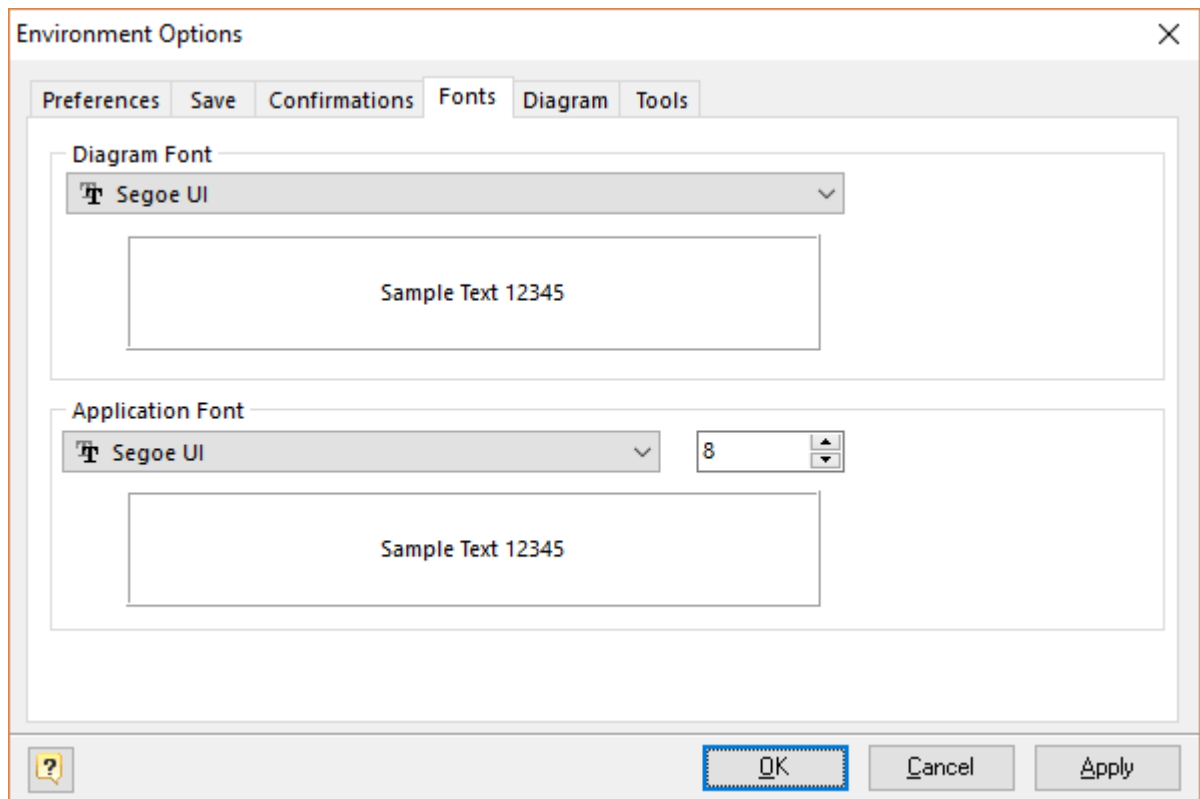
Show warning before object deletion

This option opens the dialog window that shows warning when you're going to delete an object.

Show warning before editor closing

This option opens the dialog window that shows warning when you're going to close object editor and some changes made.

Fonts



In this tab you can change the diagram and application fonts.

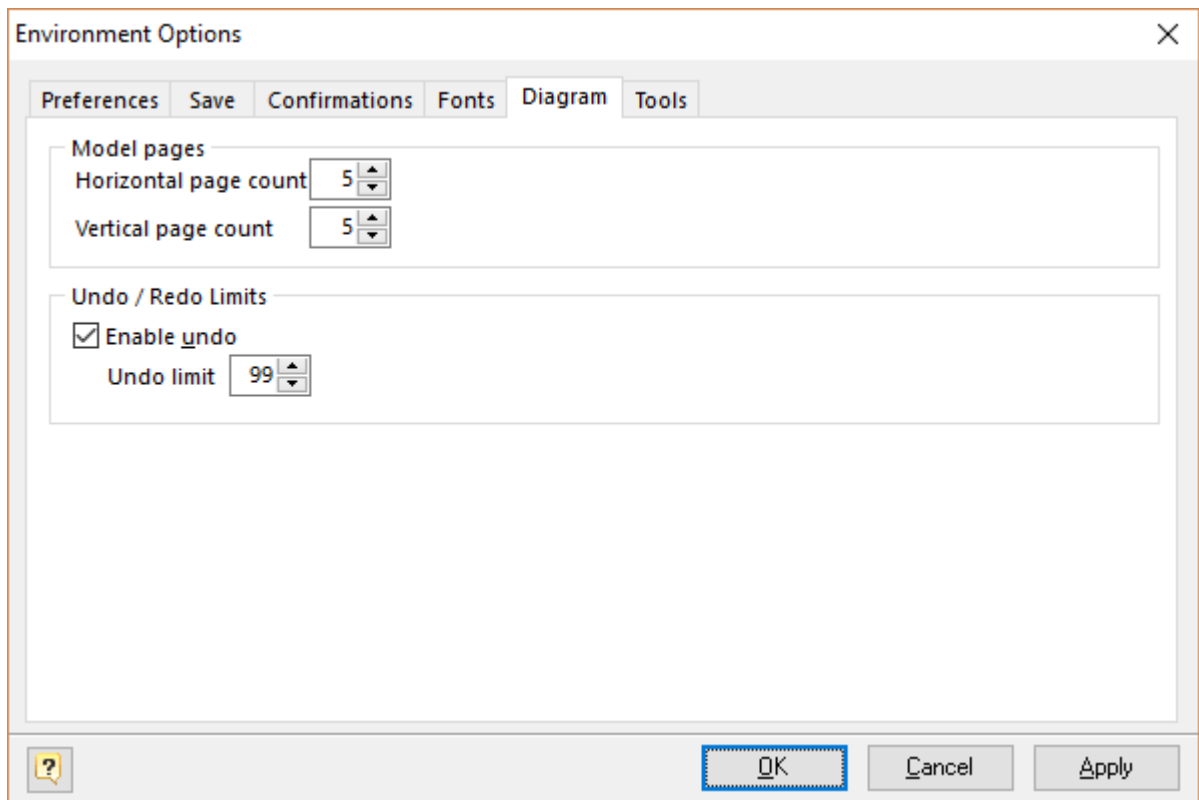
Diagram Font

You can select the font that is used in the diagram. The option changes the font in all objects' captions as well as in the captions of all diagram tables, references, etc.

Application Font

You can select the font that will be used in the application. The option changes the font of dialog windows, menus and other objects.

Diagram



In this tab you can change the quantity of pages that will be used in new diagrams.

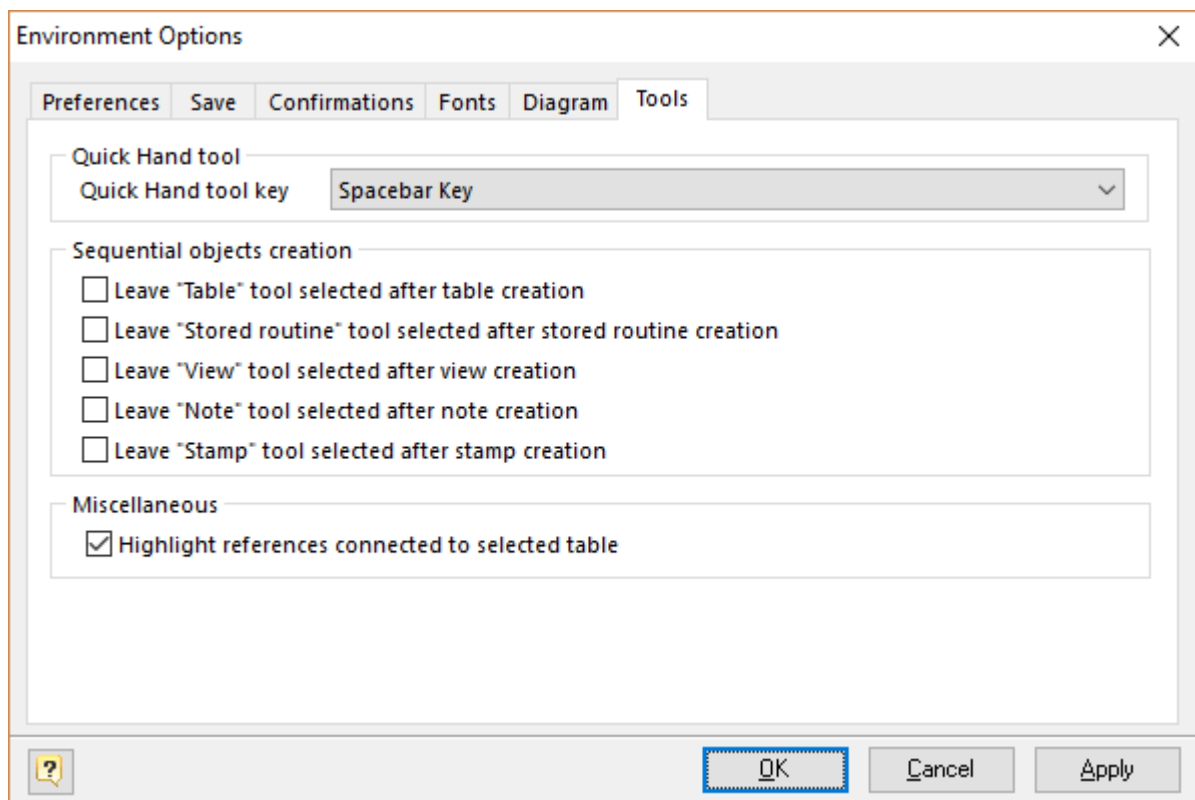
Horizontal Page Count

Horizontal size of the diagram, in pages.

Vertical Page Count

Vertical size of the diagram, in pages.

Tools



In this tab you can change the behavior of tools used in designer.

Quick Hand tool key


This option sets hot key for enabling Hand tool while it is pressed. You may choose Space bar or Control (Ctrl) key.

Sequential objects creation group

With this option group you can manage a tool changing behavior. If checkbox is unmarked, then this tool will be set to the Pointer mode after an object is created, otherwise it will remain active.

Highlight references connected to selected table

If this option is set references connected to selected table(s) will be drawn highlighted.

 Please note that highlighted references are not selected! It is just a visualization enhancement, but not functional. To select references connected to table object you should right click on the table and then use **Select...** -> Connected references menu item.

See also:

Diagram Objects: [Diagram Overview](#)

6. Diagram

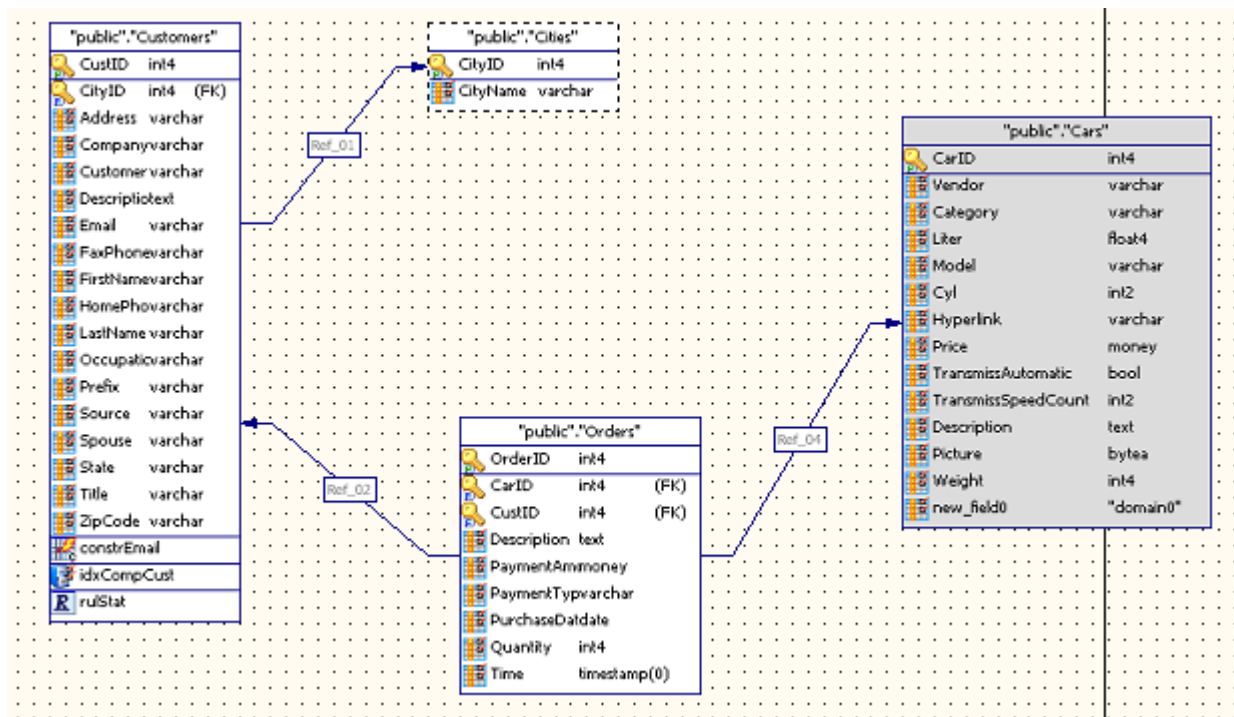
Database Designer for PostgreSQL allows you to create physical Entity Relationship Diagram, that will represent object in your PostgreSQL database.

A diagram represents the tables of your database and the relationships between them. At its core, a data diagram depicts the underlying structure of your database. It allows you to specify the data type to be used by each column in the table, and determine how tables will be stored in the database.

A diagram of your database can help you define operational aspects of your application logic that you might otherwise overlook. Also, a well-defined data diagram that accurately represents your tasks can be helpful in orienting employees to goals and operations. The data diagram can also serve as an invaluable communications tool for both internal and external constituents.

Moreover, the diagram you define does not only depict the underlying database structure, but with functionality of **Database Designer for PostgreSQL** allows you to generate appropriate databases and even bring existing databases to the state of your diagram.

In the following picture you can see four tables: *Customers*, *Cities*, *Orders*, *Cars*, their columns and the references between them. Also you may see that there are constraint, index and rule defined for table "Customers". All tables are objects of the public schema.

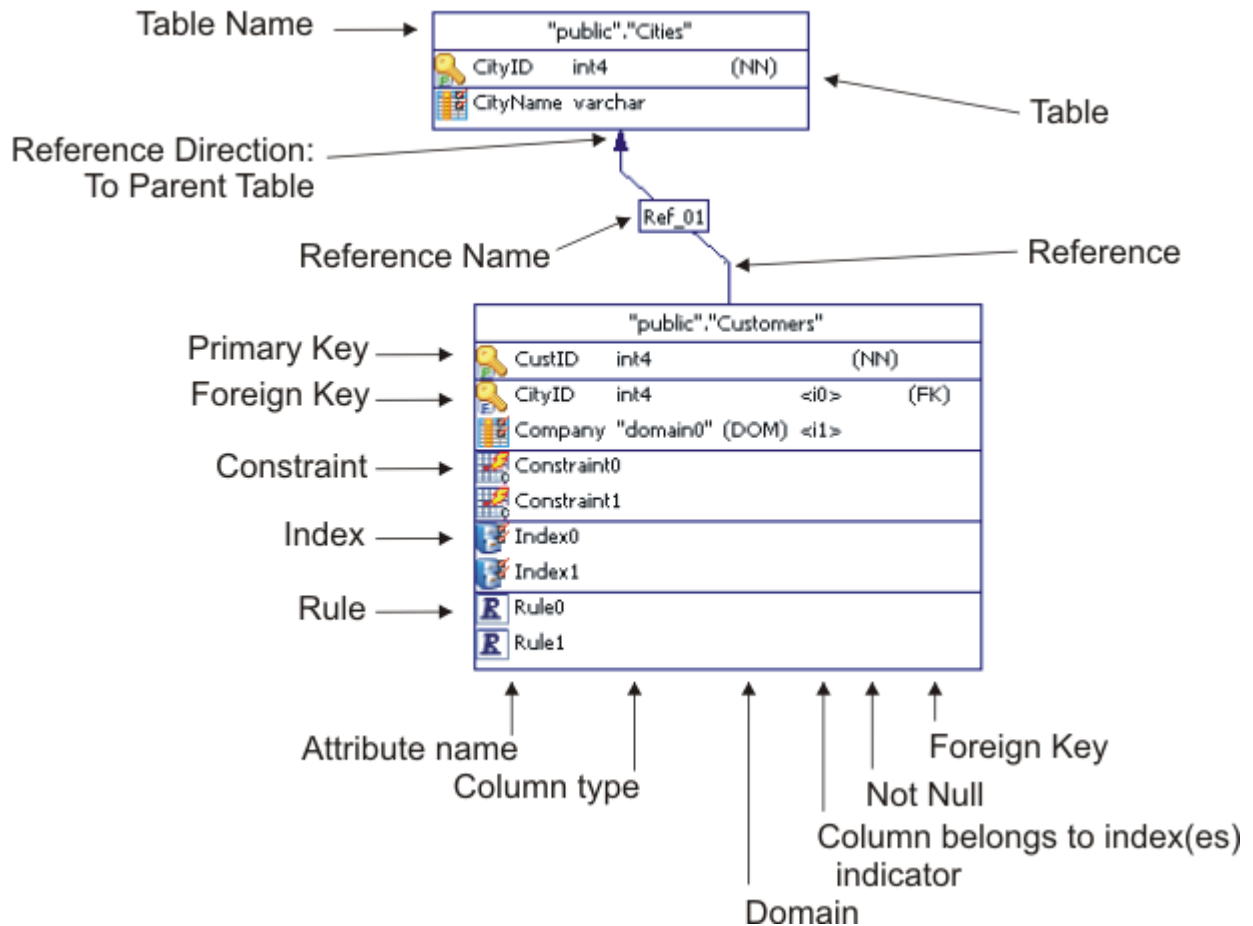


See also:

Diagram: [Notation](#) | [Creating a New Diagram](#)

6.1. Diagram notation

Please examine the following picture, which describes the diagram objects notation:




This way of diagram objects displaying is the most informative. You can change the displaying preferences according to your likes and dislikes. Please see the [Diagram Display Preferences](#) section for more information.

See also:

Entity Relationship Diagram: [Diagram Display Preferences](#)

6.2. Creating a New Diagram

The first step to use **Database Designer for PostgreSQL** is to create a new diagram.

To create a new diagram select the **File | New** menu item or press **Ctrl-N**. You can also use the New diagram button  on the **Standard toolbar**.

A new diagram will have a default name like "Noname1". You can change the name of your diagram in the [Diagram Properties](#) dialog window.

6.3. Open an Existing Diagram


To open an existing diagram file, select the **File | Open** command from the program menu. You can also use the shortcut **Ctrl-O** or press the **File Open** button in the **Standard toolbar**. The standard **Open File** dialog box will then be displayed. Browse to the diagram file you want to open and double click on it. Also, you can open a multiple diagram at once. For that purpose, please select the multiple diagram files you need in the **Open File** dialog window and click on the **Open** button.

Please note, that **Database Designer for PostgreSQL** diagram files have the **.pdd** extension.

See also:

Workspace: [Toolbars](#)

6.4. Saving a Diagram

You can save a diagram by choosing either the **Save** or **Save As** commands from the File menu. **Database Designer for PostgreSQL** saves the diagram into a specially formatted **.pdd** file, which contains the diagram and its properties. You can also save the current diagram by clicking on the **Save** () button in the **Standard toolbar**.

If you are saving a diagram for the first time, select the **Save** command, then type in a "file name" for your diagram and specify the location on your computer where you want to save it. Diagram file names must meet the requirements of a standard Windows filename.

If you are saving a diagram that was already saved, use the **Save** command to replace the file contents or the **Save As** command to save the diagram with a new name or to a location.

6.5. Selecting and Moving Objects

Once the object is in the diagram displaying area, you can select it to move it or make changes.

Selecting an object

To select an object, use the **Pointer** tool from the **Palette toolbar**, then click anywhere inside the object. The solid borders of the object will turn into the dashed ones.

Selecting multiple objects

To select multiple objects, use the **Pointer** tool from the **Palette toolbar**, then click the left mouse button and drag the selection rubberband so that the rubberband box encompasses the objects you want to select. Once all objects are within the rubberband, release the mouse button.

To add objects to the list of the already selected ones, use the **Pointer** tool from the **Palette toolbar**, then click anywhere inside the object holding the **Shift** button. If the object has already been selected, it will be deselected.

Selecting all objects

You can also select all objects in the diagram. For that purpose, click on **Select All** from the right-click shortcut menu. Another way to select all objects is to use the **Edit | Select** menu item or press **Ctrl-A**.

Deselecting objects

To deselect a selected object, click anywhere outside the selected object(s).

Moving objects

Once you have selected objects, you can change their location in the diagram. Just click within the selection with **Pointer** tool and drag the objects to the new location while holding down the left mouse button.

6.6. Copying and Pasting Objects

Copying objects to the clipboard

When one or more objects are selected, you can copy those objects to the clipboard. To do that, press **Ctrl+C**, or choose the **Copy** item from the **Edit** menu. Alternatively, you can choose the **Copy** item from the right-click shortcut menu.

Pasting objects to the diagram

When you have copied the objects to the clipboard, you can paste them either into the current diagram or into another one.

To paste the objects from the clipboard into the diagram, press **Ctrl-V** or choose the **Paste** item from the **Edit** menu, you can also paste the objects by choosing the **Paste** item from the right-click shortcut menu.

Please note, that you can paste the objects not only into the current diagram, but into any other diagram. To do that, please select the necessary diagram from the list of the opened ones (this list can be found in the Windows menu), or open an existing diagram by selecting **File | Open** menu item. Certainly, you can also paste the objects into a new diagram.

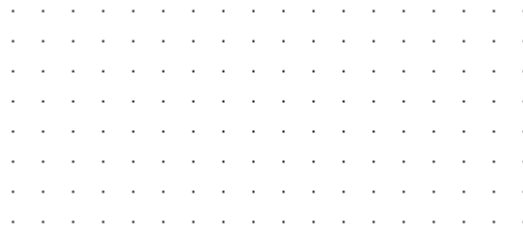
See also:

Diagram: [Creating a New Diagram](#) | [Open an Existing Diagram](#)

6.7. Using Snapping Grid

Very often it is necessary to line up some tables and objects in a column. One of the ways to do so is to take advantage of the grid. If you click on the **Display Grid** button on the **Display** toolbar, you enable the displaying of the grid, that helps to justify the diagram objects.

The following picture illustrates such a grid:



If you enable the **Snap to Grid** function by clicking on the **Snap to Grid** button on the **Display** toolbar, then when you are moving a table or some other object, its upper left corner will "snap" to the nearest grid point. This feature will help you line up objects both horizontally and vertically.

Diagram objects moved into a diagram can be automatically snapped to the grid points even if the grid is not displayed on the diagram.

It is possible to change the interval between the grid points. Choose **Tools | Display Preferences** to define both the displaying of the diagram grid and behavior of the diagram, make sure that you stay in the **General** tab of the [Diagram Display Preferences](#) dialog window:

- select **Show grid** checkbox to display the grid on the diagram
- select **Snap to grid** checkbox to enable snapping feature
- modify value of **Size** field to change the interval between the grid points.

See also:

Diagram: [Diagram Display Preferences](#)

6.8. Export to Graphics

You can export your diagram to an image file and then use its graphical representation in external applications. For example, insert a diagram image into your program documentation (e.g. use MS Word), create big posters with your diagram, publish them to Web.

Database Designer for PostgreSQL supports the following image types:

- Vector graphics: Windows Enhanced Metafile (EMF).
- Bitmap images: PNG, GIF, JPEG, BMP.

To export model to image, follow these steps:

1. Select **File | Export** to call the **Export Model** tool.
2. The export dialog window will appear. Please examine the dialog window controls:

File name

Enter the name for the file, in which you want to store the contents of the model.

Split into pages

Click on this option to create a separate image for each page in the model.

Show image on complete

Click on this option to open image in a default image viewer after the generation.

Click **OK** to export model.

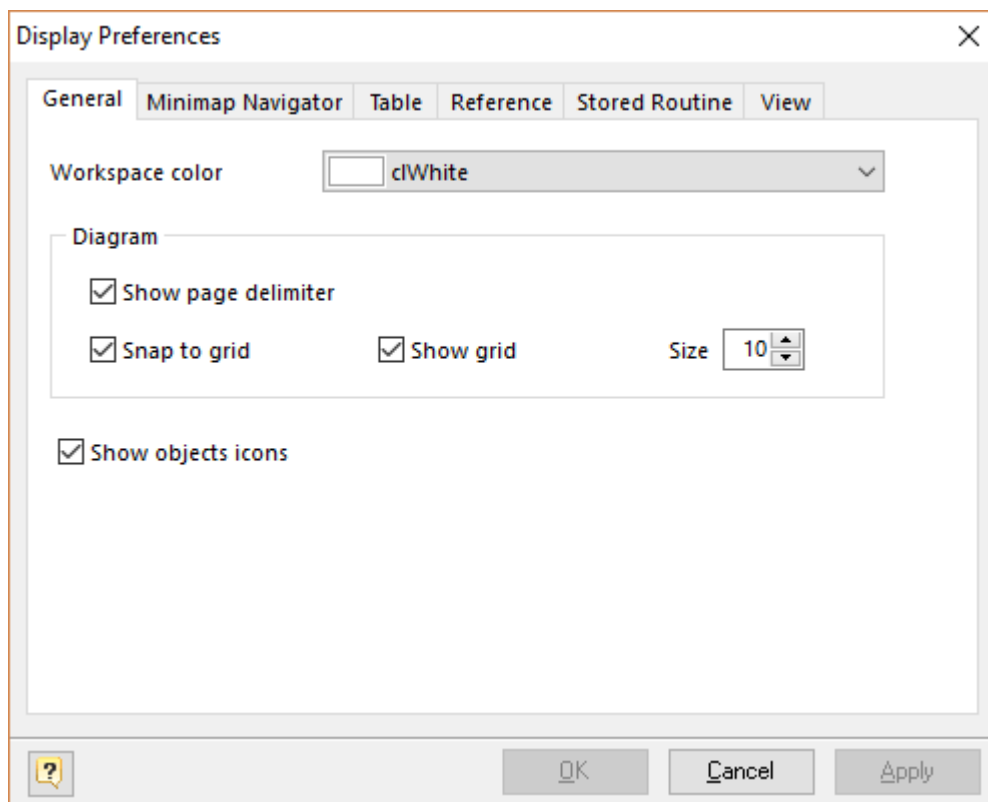
Please note, that exporting a diagram to the vector graphics format allows you to freely scale a diagram of any size without any noticeable distortions. It enables you to create big posters with the diagram to demonstrate the data logic of your application.

6.9. Diagram Display Preferences

Display Preferences editor allows you to change the appearance of the diagram, control the number of the object attributes, which are displayed in the diagram and define the diagram grid displaying and behavior.

To access the **Display Preferences editor**, select the **Tool | Display Preferences** item menu.

General



On this tab you can set the color of your diagram workspace, define the grid parameters and enable page delimiters displaying.

Workspace Color

Use this option to set the color of the diagram background. Choose the color you need from this drop-down list.

Show Page Delimiter

Makes page delimiters visible in the diagram. This helps you to understand how the diagram will be disposed on paper.

Snap to Grid

Enables snapping of the objects to the grid. If you enable this function, then when you move a table or some other object, its upper left corner will "snap" to the nearest grid point.

Show Grid

Shows the grid in your diagram.

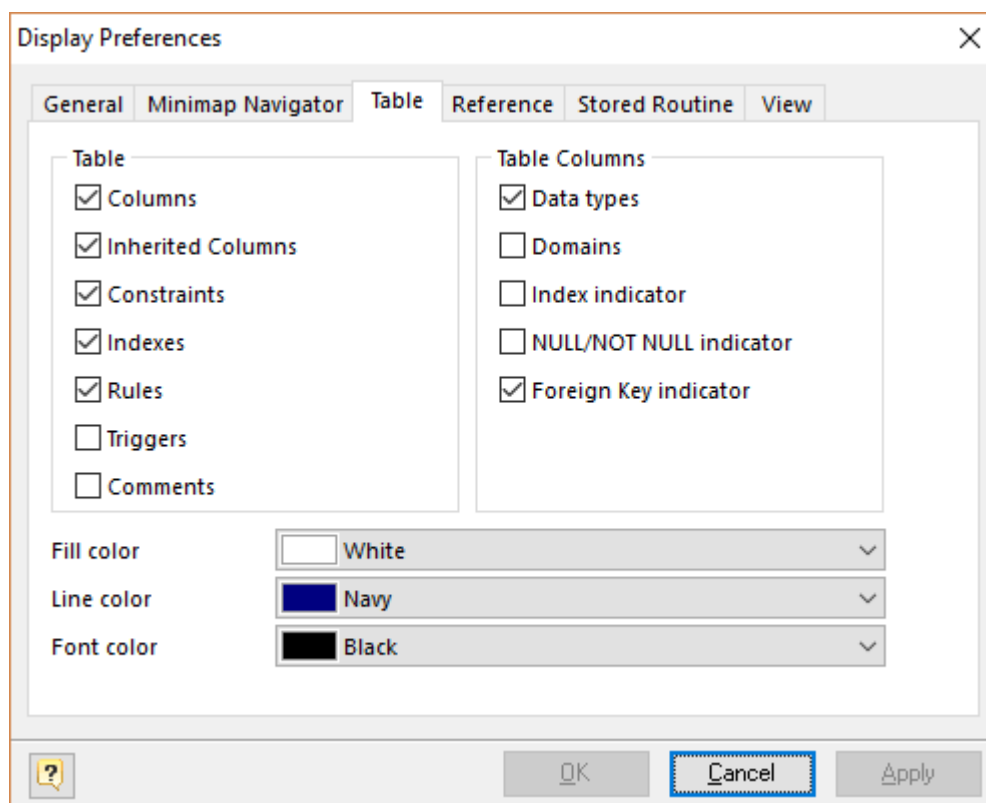
Size

Use this option to change the interval between the grid points.

Show objects icon

Shows the object icon to the left of object name.

Table



This tab allows you to define the table attributes to show in the diagram and set the default colors for the new tables. See [Notation](#) section to find out what the table parameters mean. Please note, that you can change some of the display parameters for a particular table using [Formatting Table](#) editor.

Columns

Enables displaying table columns on the diagram.

Inherited Columns

Enables displaying of table columns inherited from other tables on the diagram.

Constraints

Enables displaying of table constraints on the diagram.

Indexes

Enables displaying table indexes on the diagram.

Rules

Enables displaying of table rules on the diagram.

Triggers

Enables displaying table triggers on the diagram.

Comments

Enables displaying a table comments on the diagram.

Data types

Enables displaying a data type near a table column.

Domains

Show the domain indicator (DOM) near a domain-based table column.

Index indicator

Shows the indexes for which this column belongs to.

NULL/NOT NULL indicator

Shows (NN) indicator if column has NOT NULL status.

Foreign key indicator

Shows the availability of a foreign key for a column.

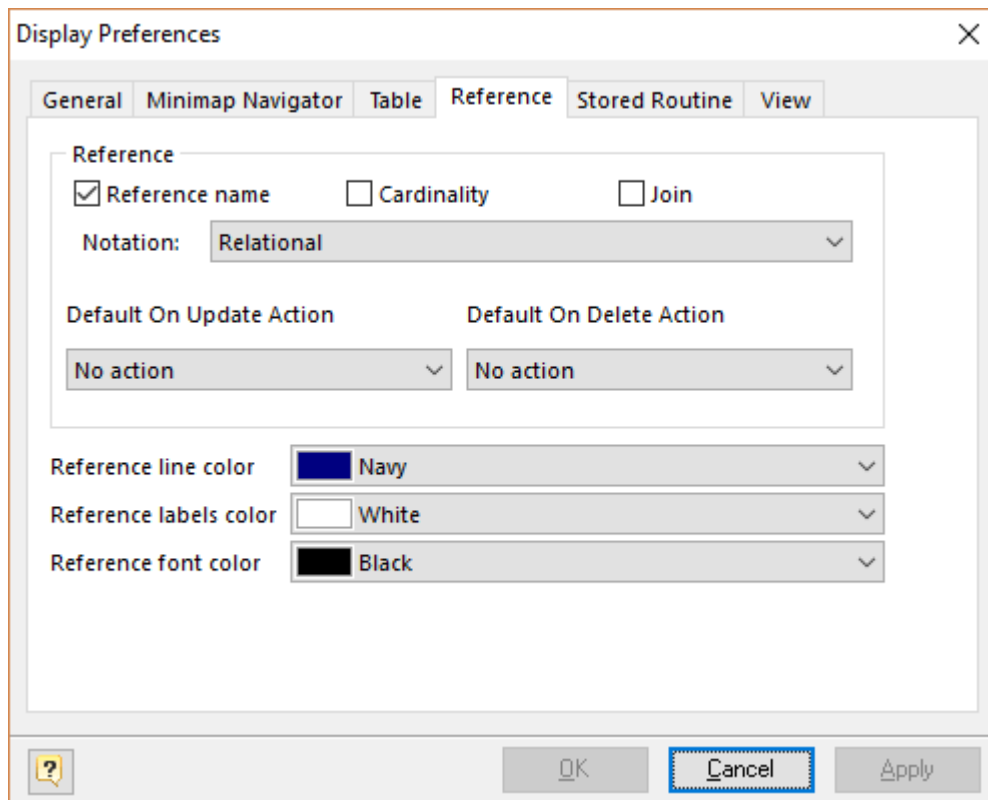
Fill Color

Use this option to set the color for a table background. Choose the color you need from this drop-down list.

Line Color

Use this option to set the border color of a table. Choose the color you need from this drop-down list.

Reference



This tab allows you to set the reference display preferences. Note, that you can apply changes to a particular reference using [Reference Editor](#).

Reference Name

This option enables displaying of the reference name box in the middle of the reference.

Cardinality

This option enables displaying of the reference cardinality.

Join

This option enables displaying of the table columns used in the reference join.

Notation

Use this drop-down list to select the reference notation style:

- **Relational.** This notation shows a reading direction for references.
- **Conceptual** (Cross Foot). This notation consists of splitting of a line into an approximation of the cardinality.

Default On Update Action

Choose On Update action that will be set for new references.

Default On Delete Action

Choose an On Delete action that will be set for new references.

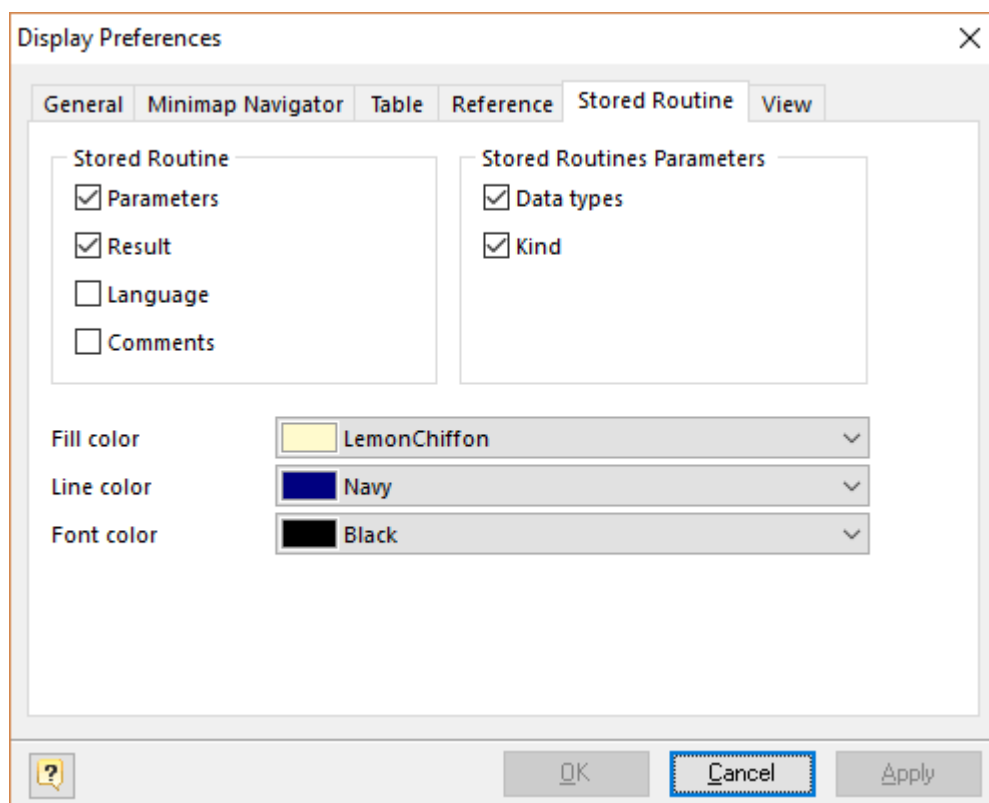
Reference Line Color

Use this option to set the diagram references color. Choose the color you need from this drop-down list.

Reference Labels Color

Use this option to set the color of join label. Choose the color you need from this drop-down list.

Stored Routines



This tab allows you to set the display preferences of stored procedures and functions. Note, that you can apply some changes to a particular stored procedure using [Stored Routine Editor](#).

Parameters

Enables displaying stored routine parameters on the diagram.

Result

Enables displaying stored routine result on the diagram.

Language

Enables displaying stored routine procedural language on the diagram.

Comments

Enables displaying stored routine comments on the diagram.

Data types

Enables displaying stored routine parameters data types on the diagram.

Kind

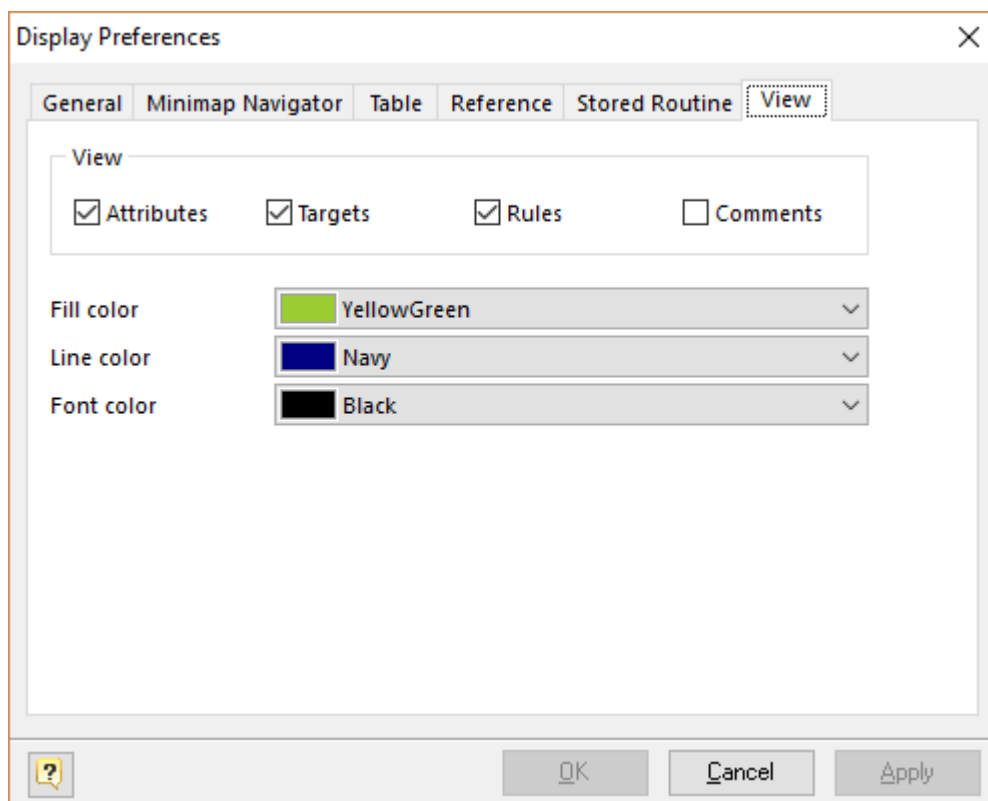
Enables displaying stored routine parameters kind (IN, OUT, INOUT, VARIADIC) on the diagram.

Fill Color

Use this option to set a default background color of a stored routine. Choose the color you need from this drop-down list.

Line Color

Use this option to set a default border color of a stored routine. Choose the color you need from this drop-down list.

Views

This tab allows you to set the display preferences of views. Note, that you can apply some changes to a particular view using [View Editor](#).

Attributes

Enables displaying view attributes (columns) on the diagram.

Targets

Enables displaying view targets (tables, functions etc.) on the diagram.

Rules

Enables displaying of view rules on the diagram.

Comments

Enables displaying view comments on the diagram.

Fill Color

Use this option to set a default background color of a view symbol. Choose the color you need from this drop-down list.

Line Color

Use this option to set a default border color of a view. Choose the color you need from this drop-down list.

See also:

Diagram: [Using Snapping Grid](#) | [Notation](#)

6.10. Diagram Properties

The **Diagram Properties** tool can be called by selecting **Diagram | Diagram Properties** from the program menu, or by clicking on the **Diagram Properties** button on the **Diagram** toolbar.

Diagram Properties editor is a tool for setting basic diagram overview information and getting diagram numerical statistics. **Diagram Properties editor** also allows you to view diagram SQL preview.

General diagram properties

The screenshot shows a dialog box titled "Diagram Properties - (pagila-light)". It has several tabs: "General", "Default Database Options", "Naming", "Pages", "Preview", "Notes", and "Statistics". The "General" tab is selected. It contains the following fields and options:

- Project:** pagila
- Diagram:** pagila-light
- Company:** pgfoundry.org
- Author:** Christopher Kings-Lynne, Robert Treat, Mike Hillyer
- Copyright:** BSD license
- Version:** 0.45
- ☒ Auto Increment version on save
- ☐ Include version to Generate & Modify scripts file names
- Created:** 08.04.2009 15:34:32
- Updated:** 13.06.2017 17:14:49

At the bottom, there are buttons for "?", "OK", "Cancel", and "Apply".

On **General** tab you can type in auxiliary information about the diagram creator, the diagram name and such. This information helps you to identify your diagram. It will be displayed on the [Stamp](#) diagram object and in the diagram reports.

Project

Use this option to set a project name diagram belongs to. It may be a name of a system being developed. E.g. *"Car Sales System"*, *"Web Forum"*, etc.

Diagram

Use this option to set the diagram name. Note, that the diagram name will be displayed in the title of **Database Designer for PostgreSQL** and in the list of the opened diagrams in the Windows program menu. We kindly advise you to set a short and descriptive name.

Company

Use this field to set the name of the company that develops the diagram.

Author

Use this field to set the name of the person who develops the diagram.

Copyright

Use this field to set the copyright information of the model.

Version

Use this field to set the diagram version. This helps you to identify the state of the diagram.

Auto increment version on save

Set this option on to enable automatic increment of the minor part of the diagram version when saving the diagram. In other words, to change the version number from 1.0 to 1.1, 1.2, etc...

Include version to Generate & Modify scripts file names

Set this option on to automatically add version info to script file names in Database Generation and Database Modification dialogs.

Created

This field shows you the date and time when the diagram was created.

Updated

This field shows you the date and time when the diagram was updated last.

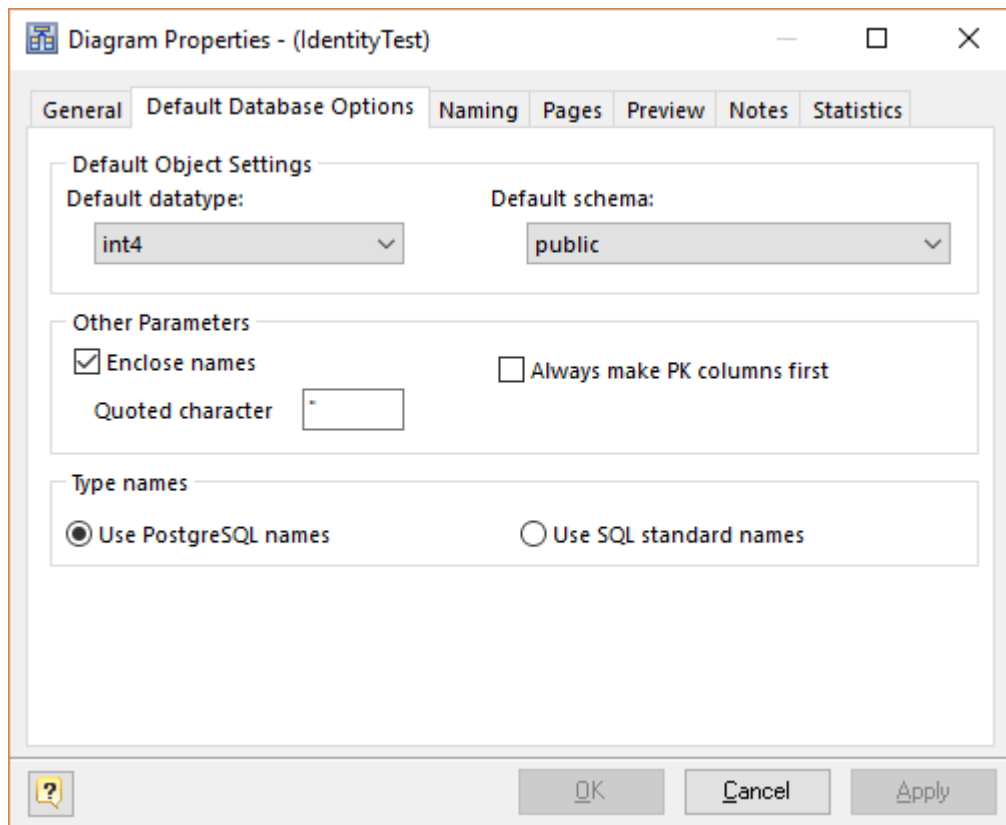
See also:

Diagram: [Database Options](#) | [Diagram SQL Preview](#) | [Diagram Notes](#) | [Diagram Statistics](#) | [Diagram Pages](#)

6.10.1. Default Database Options**Default database options**

This tab helps you to set the parameters, that will be applied as defaults for the newly created tables and generated SQL scripts.

To open this dialog window, please select **Diagram | Diagram Properties** and switch to the **Default Database Options** tab. You can also select **Diagram | Default Database Options** directly.



There are the following options:

Default Data type

Use this drop-down list to select the data type. This option will be applied as a default data type for the newly created columns.

Default schema

Use this drop-down list to select the existing schema. This option will be applied as a default schema for the newly created database objects: tables, stored routines, views, types etc.

Foreign Key columns prefix

Sets the prefix, which will be added to the names of new foreign key columns. You can use substitution %TABLENAME%, which will be replaced by the name of the master table.

Enclose Names

This option forces quoting of the diagram objects names by a quote character, i.e. it makes "Column" and "Table" instead of Column and Table in SQL script.

Quoted Character

This option sets the character that will be used on SQL script generation to quote table column names and values. Default PostgreSQL quoted symbol is '"'. We strongly recommend do not change this property.

Type names

Determines what type names will be used in the generated SQL, PostgreSQL (*int2*, *int4*, *int8*, *varchar*, *char*, *bool*, *float8*, *float4*) or standard ones (*smallint*, *integer*, *bigint*, *character varying*, *character*, *boolean*, *double precision*, *real*).

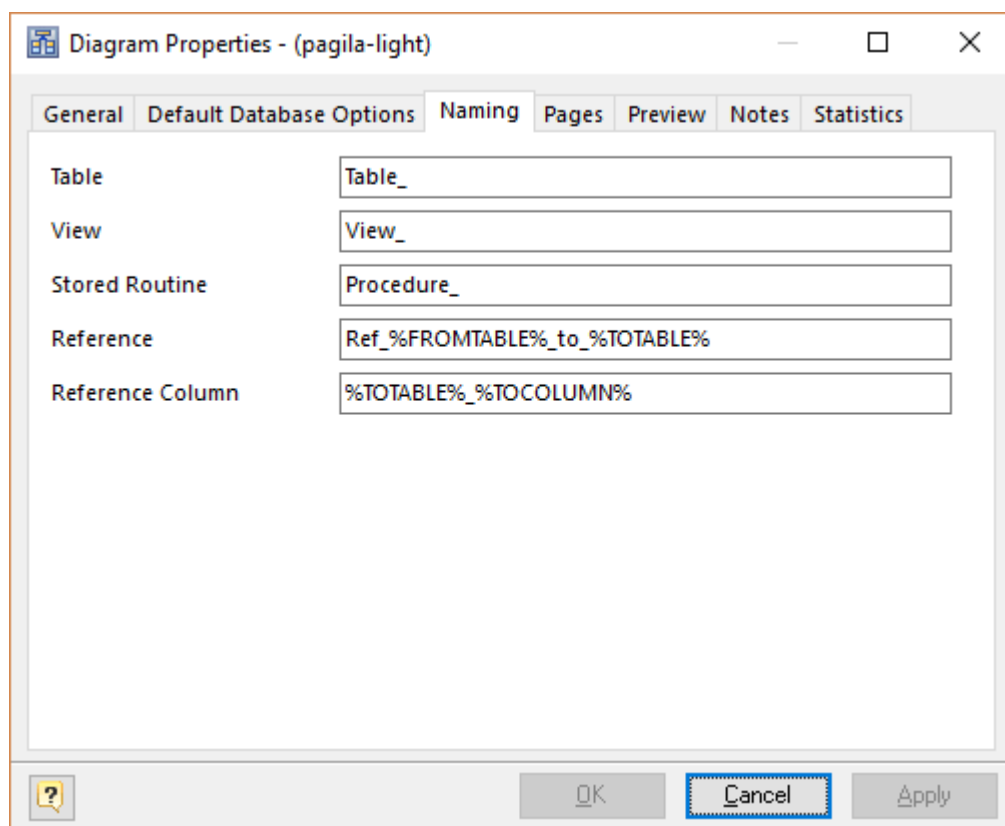
See also:

Diagram: [General Diagram Properties](#) | [Diagram SQL Preview](#) | [Diagram Notes](#) | [Diagram Statistics](#)

Diagram Objects: [Creating a Reference](#)

6.10.2. Diagram Naming

This tab helps you to set the patterns for newly created objects.



Table

Set the naming pattern for newly created tables. Default is "Table_". Number will be automatically added to the end, e.g. "Table_04".

View

Set the naming pattern for newly created views. Default is "View_". Number will be automatically added to the end, e.g. "View_12".

Stored Routine

Set the naming pattern for newly created stored routines. Default is "Procedure_". Number will be automatically added to the end, e.g. "Procedure_01".

Reference

Set the naming pattern for newly created references. Such format specifiers may be applied:

%FROMTABLE% - the argument is the referencing table name;

%TOTABLE% - the argument is the referenced table name.

Reference Column

Set the naming pattern for newly created referenced columns. Such format specifiers may be applied:

%TOTABLE% - the argument is the referenced table name;

%TOCOLUMN% - the argument is the referenced column name in the referenced table.

6.10.3. Diagram Pages

This tab helps you to set the size of the diagram. You can change the number of pages in the diagram.

Horizontal Page Count

Horizontal size of the diagram, in pages.

Vertical Page Count

Vertical size of the diagram, in pages.

6.10.4. Diagram SQL Preview

This tab shows you SQL representation of your diagram. To access this dialog window, please select **Diagram | Diagram Properties** and switch to the **Preview** tab. SQL syntax elements are colored.

It is possible to copy these definitions to clipboard.

See also:

Diagram: [General Diagram Properties](#) | [Database Options](#) | [Diagram Notes](#) | [Diagram Statistics](#)

6.10.5. Diagram Notes

This tab helps you to write descriptions and annotations for the diagram.

To access this dialog window, please select **Diagram | Diagram Properties** and switch to the **Notes** tab.

Use the bottom tabs to select between the description and annotation of the diagram.

See also:

Diagram: [General Diagram Properties](#) | [Database Options](#) | [Diagram SQL Preview](#) | [Diagram Statistics](#)

6.10.6. Diagram Statistics

This tab shows you the numerical statistics on the diagram objects.

You can see how many domains, tables, columns and other objects your diagram contains.



See also:

Diagram: [General Diagram Properties](#) | [Database Options](#) | [Diagram SQL Preview](#) | [Diagram Notes](#)

6.11. Zooming a Diagram

When a diagram is opened for the first time its zoom level is set to the normal size. You can change the zoom level of your database diagram.

There are several ways to zoom a diagram in the window:

- The most convenient variant is to use the mouse wheel holding **Ctrl** button. Turn up the mouse wheel to zoom in the digram, to zoom out turn down the mouse wheel.
- In the **View** menu you can find the **Scale** drop-down list with predefined scale factors. The same drop-down list is also placed in the **View** toolbar.
- Press **F6** to zoom in the diagram, and **F7** to zoom out. Press **F5** to return to actual size (100%) of the diagram.
- The **Zoom-in** () and **Zoom-out** () buttons are placed in the **Palette** toolbar .
- Select the **View | Zoom In** or **View | Zoom Out** menu items to zoom in and to zoom out respectively.

Please pay attention to the **Fit To Screen** function. It helps to display your diagram fully. You can call this function from the **View** menu by clicking on the **Fit To Screen** label.

6.12. Auto Layout Diagram

The **Auto-Layout** function automatically rearranges symbols in the active diagram. The symbols are arranged in rows from left to right horizontally. This function is useful when you want to line up multiple tables in the diagram automatically.

To call this function select **View | Auto-Layout**. Please note, that auto-layout operation is undoable in most cases.

See also:

Diagram: [Using Snapping Grid](#) | [Zooming a Diagram](#)

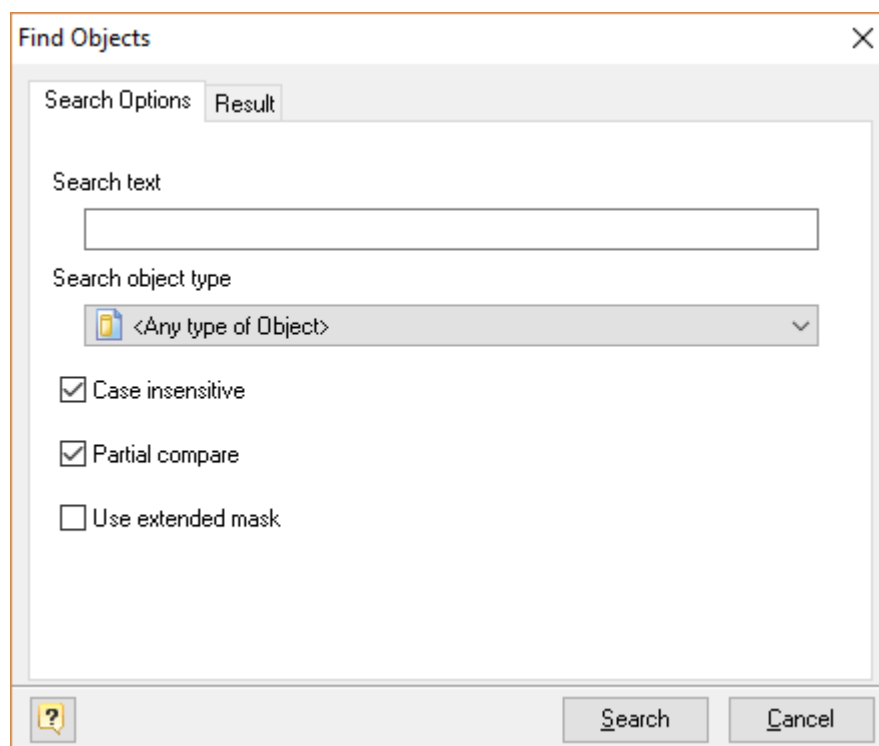
6.13. Find Objects

The **Find objects** feature lets you search objects within the entire **Database Designer for PostgreSQL** diagram so that you could locate the necessary objects in the diagram easily, find all the attributes related to a given object.

To call **Find Objects** dialog window, select the **Edit | Find Objects** menu item or press **Ctrl-F**.

Defining find parameters

You can define the search parameters in the **Find Objects** dialog window.



Search text

This option allows you to define the text that you want to search in the objects and attributes names.

Search object type

You can define the object type which you want to search for. Also you can search for any type of objects, please select one of the following object types:

Table
Column


Constraint
Index
Trigger
Rule
Reference
View
Stored Routine

Case insensitive

If this option is enabled, case matching is off and it will not affect on the search results.

Partial compare

If this option is enabled, you will find objects with partial name matching.

 If **Use extended mask** and **Partial compare** are checked, this means only that to **Search text** leading and trailing percent mark (%) will be added, e.g. %<Search Text>%.

Use extended mask

You can search objects which conform to the format specified by a **Search text**. A valid mask consists of literal characters, sets, and wildcards.

Each literal character must match a single character in the string.

Each set begins with an opening bracket ([) and ends with a closing bracket (]). Between the brackets are the elements of the set. Each element is a literal character or a range. Ranges are specified by an initial value, a dash (-), and a final value. Do not use spaces or commas to separate the elements of the set. A set must match a single character in the string. The character matches the set if it is the same as one of the literal characters in the set, or if it is in one of the ranges in the set. A character is in a range if it matches the initial value, the final value, or falls between the two values. If the first character after the opening bracket of a set is an exclamation point (!), then the set matches any character that is not in the set.

Wildcards are percent (%) or question marks (?). A percent matches any number of characters. A question mark matches a single arbitrary character.

Any character may be escaped using "\" character. To include "\" itself just double it, e.g. "\\"

Examples:

analy[sz]e

Conforms to both American and British spelling of "analyze" and "analyse"

[!a-fz]_column

Conforms to words where the first letter is not a..f or "z"

anal%

Conforms to analyzer, analog, analyze, analyse etc.

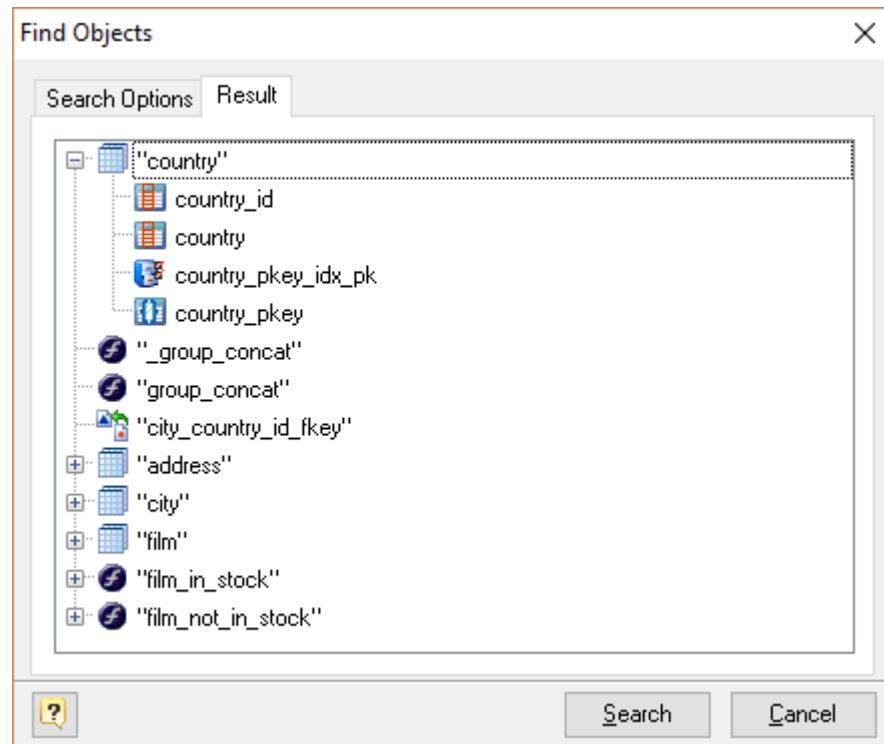
de?r

Conforms to dear, deer, deor, depr, etc.

`array\{?\}`

Conforms to array[a], array[5], array[_], array[!], etc.

Using the Result List



The **Result** tab displays the result of the search in the result tree. You can use the result tree to:

- learn to which objects the found objects (attributes) belong to;
- modify the found objects.

Double-click on the object in the result tree to call corresponding editor.

7. Diagram Objects

Database Designer for PostgreSQL allows to manage the following diagram objects:

[Databases](#)

[Tables](#)

[Columns](#)

[Constraints](#)

[Indexes](#)

[Triggers](#)

[Rules](#)

[Stored Procedures and Functions](#)

[Views](#)

[Schemas](#)

[Domains and User Defined Types](#)

[References and Foreign Keys](#)

[Roles](#)

[Tablespaces](#)

[Sequences](#)

[Privileges](#)

7.1. Database

Database is a virtual diagram object, that can be used during physical database generation process. You can set several basic parameters for the new database such as the database name, default encoding and so on.

This object is especially interesting to those who want to create a new database. You may skip this section if you have an already created physical database and you only want to generate its structure.

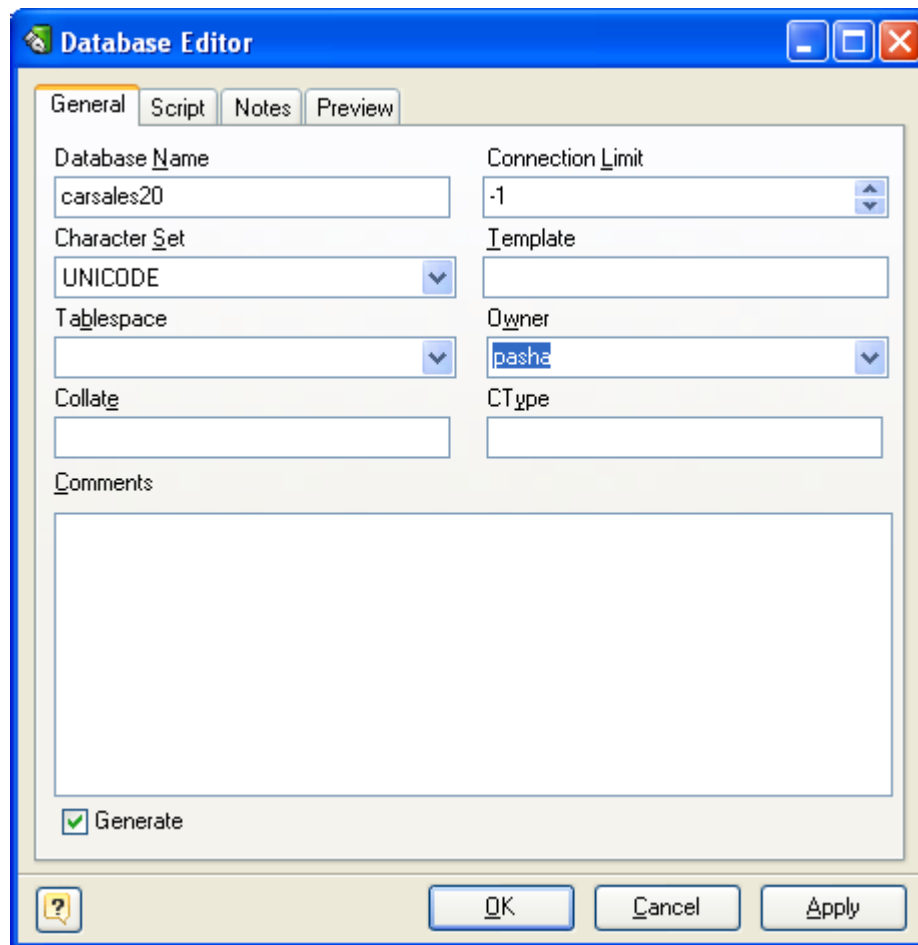
To set up a database object use [Database Editor](#).

7.1.1. Database Editor

Database Editor helps you to set up a virtual diagram object - a database. This object can be used during database generation. To open **Database Editor** select **Diagram | Database Editor** menu item.

Database Editor contains several tabs, please see the detailed description below.

General



This tab allows you to set the name of the database and other basic parameters.

Database Name

Use this field to set the database name.

Connection Limit

How many concurrent connections can be made to this database; -1 (the default) means no limit.

Character Set

This option specifies the default database character set.

Template

The **Template** parameter contains the name of the template you want to base your new database of. Use the DEFAULT keyword to specify the default template (usually *template1*).

Tablespace

The **Tablespace** parameter contains the name of a tablespace. A tablespace allows you to define an alternative location on the file system where the data files containing database objects (such as tables and indexes) may reside. You can manage tablespaces using SQL commands such as CREATE TABLESPACE and ALTER TABLESPACE.

Owner

The name of the database user who will own the new database, type DEFAULT to use the default user name (namely, the user executing the command).

Collate

Collation order (LC_COLLATE) to use in the new database. This affects the sort order applied to strings, e.g. in queries with ORDER BY, as well as the order used in indexes on text columns. The default is to use the collation order of the template database. See below for additional restrictions.

CType

Character classification (LC_CTYPE) to use in the new database. This affects the categorization of characters, e.g. lower, upper and digit. The default is to use the character classification of the template database. See below for additional restrictions.

Comments

A comment for the table.

Generate

Set this option off to disable generation of the physical database. The database objects will be generated in compliance with their own properties.

Script

This tab allows you to set SQL statements, which will be executed before (use **Begin** tab) and after (use **End** tab) generation of database.

Preview

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please note, that the text within the editor is read-only. The contents of this tab will be updated only when you press **Apply** button.

Notes

The **Note** tab allows you to define the description and annotation for the edited database.

See also:Diagram Objects: [Database](#)

7.2. Tables

Tables are the basic building blocks of a diagram. Every diagram is made up of one or more tables and various objects related to those tables.

Each table consists of a set of columns that contain the information about the type of data stored in the table. Each column must be assigned a name, data type, and length. A table can also have a set of [foreign keys](#), constraints, rules and [indexes](#). Since columns, indexes, constraints, rules and triggers are defined for a specific table, they are treated as properties of that table in a database diagram.


With **Database Designer for PostgreSQL** you can:

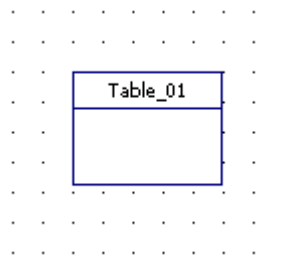
[Create a new table](#)[Modify the table parameters](#)[Modify multiple tables at once](#)[Manage table columns](#)[Manage table constraints](#)[Manage table indexes](#)[Manage table triggers](#)[Manage table rules](#)

See also:Diagram Objects: [Creating a Table](#) | [Table Editor](#) | [Table Manager](#) | [Column Manager](#) | [Index Editor](#) | [Constraint Editor](#) | [Trigger Editor](#) | [Rule Editor](#)

7.2.1. Creating a Table

To create a new table in a diagram:

1. Click on the **Table** () icon on the **Palette** toolbar. Your mouse cursor will change its appearance. Click on the diagram area to create a new table. Or right click on the necessary schema in the **Object Tree View** and choose **Create Table** item. Or right click on the diagram area and choose **Create Object -> Table**. An empty table will appear in the diagram:



2. (optional) Double click on the new table symbol in the diagram to display the [Table Editor](#) dialog window.
3. (optional) Enter the table name in the **Table name** field.
4. (optional) Click **OK** to save the changes.

See also:

Diagram Objects: [Table Editor](#)

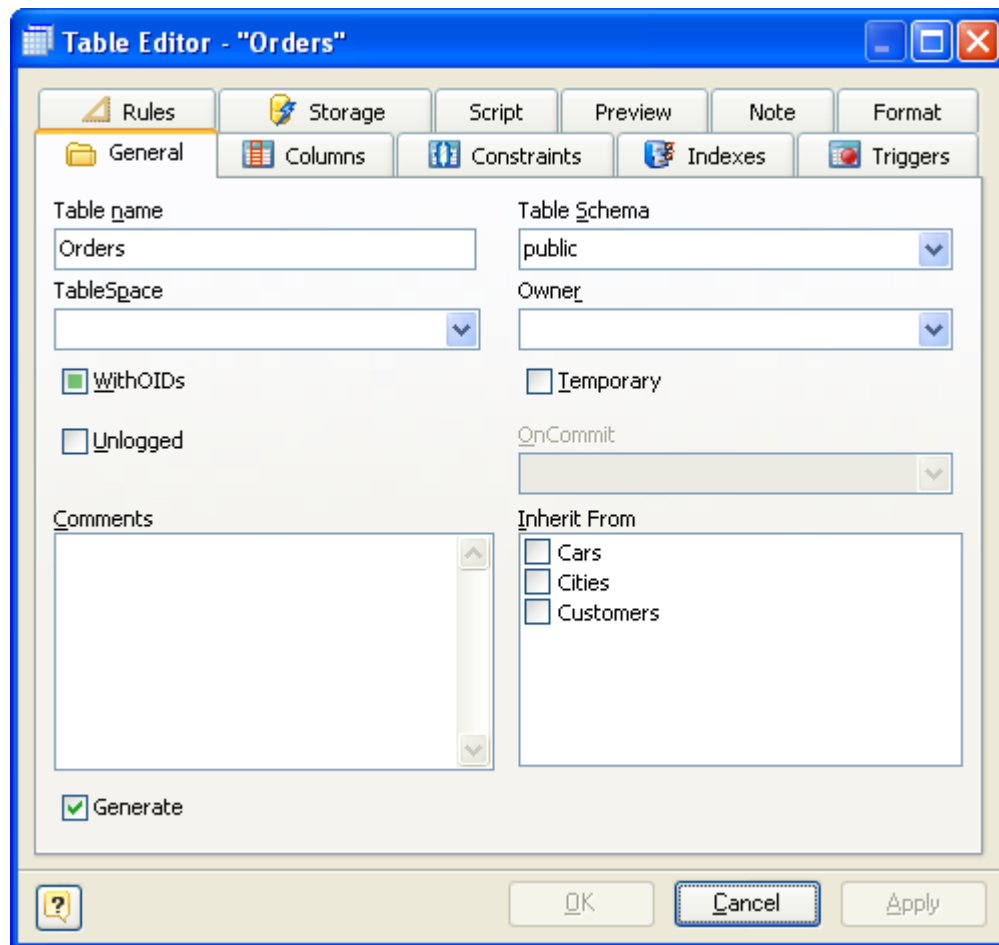
7.2.2. Table Editor

The **Table Editor** dialog window is intended for editing the following properties of a diagram table: general and database-specific table options (table name, etc.); table columns, table indexes, table constraints, table rules, display options and others.

To open **Table Editor**, simply double-click on a table in the diagram or select the **Properties** item from the table context menu.

Table Editor consists of several tabs, please see the detailed description below.

General



This tab allows you to adjust the basic table properties, which are used for generating CREATE TABLE or ALTER TABLE statements for executing on the PostgreSQL server.

Table name

The name of the table must be unique within the schema. To check your diagram for the uniqueness of table names use the [Check Diagram](#) tool.

Table schema

The name of the schema where your table will be created.

WithOIDs

This optional clause specifies whether the rows of your new table should have OIDs (object identifiers) assigned to them.

Owner

A role which will be owner of the object, or a role which will execute CREATE script in case of empty input field.

Tablespace

The name of a tablespace. A tablespace allows you to define an alternative location on the file system where the data files containing database objects (such as tables and indexes) may reside. You can manage tablespaces using SQL commands such as CREATE TABLESPACE and ALTER TABLESPACE.

Temporary

This option allows you to create a temporary table, i.e. a table which drops itself at the end of the session.

Fill Factor

The fillfactor for a table is a percentage between 10 and 100. 100 (complete packing) is the default. When a smaller fillfactor is specified, INSERT operations pack table pages only to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables smaller fillfactors are appropriate.

OnCommit

The behavior of temporary tables at the end of a transaction block. The three options are:

PRESERVE ROWS

No special action is taken at the ends of transactions. This is the default behavior.

DELETE ROWS

All rows in the temporary table will be deleted at the end of each transaction block. Essentially, an automatic TRUNCATE is done at each commit.

DROP

The temporary table will be dropped at the end of the current transaction block.

Comments

A comments for your table.

Inherit From

This list specifies tables from which the new table automatically inherits all columns.

Generate Table

Set this option off to exclude the table from the default selection of generated tables in the [Database Generation](#) and [Database Modification](#) tools.

Columns

Use the **Columns** tab for adding, modifying, and deleting table columns. Please see the topic [Column Editor](#) for the detailed information.

Constraints

Use the **Constraints** tab for managing constraints. Please see the [Constraint Editor](#) topic for the detailed information.

Indexes

The **Indexes** tab is intended for managing table indexes. Please see the [Index Editor](#) topic for the detailed information.

Triggers

The **Triggers** tab is intended for managing table triggers. Please see the [Trigger Editor](#) topic for the detailed information.

Rules

The **Rules** tab is intended for managing table rules. Please see the [Rule Editor](#) topic for the detailed information.

Script

This tab allows you to set SQL statements, which will be executed before (use **Begin** tab) and after (use **End** tab) generation of the table.

Preview

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using previous tabs. Note, that the text within the editor is read-only.

Note

The **Note** tab allows you to define a description and an annotation for the edited table. This properties will not affect the physical PostgreSQL database, but they can be useful for your diagram development.

Format

These options allow you to set table line and fill color for displaying on the diagram, different from the default table colors, which are defined within the [Diagram Display Preferences](#) dialog.

See also:

[Database Modification](#) | [SQL Table Definition](#)

Diagram Objects: [Column Editor](#) | [Domain Manager](#) | [Index Manager](#) | [Formatting Table](#) | [Tables in Tree View Window](#) | [Table Manager](#)

Database Functions: [Database Generation](#)

Diagram: [Diagram Display Preferences](#)

7.2.3. Columns

In a table data is arranged into columns. A column stores data element, such as a person name, a price, or any similar type of information. When columns are created in a table, they are given a name that identifies their purpose and role, such as `PersonName` or `Price`. Usually, you must also specify additional properties, such as data type and how long the longest entry in the column will be, other properties can include the column is the table primary key.

With **Database Designer for PostgreSQL** you can:

- [Edit table columns](#)
- [Edit multiple table columns at once](#)

See also:

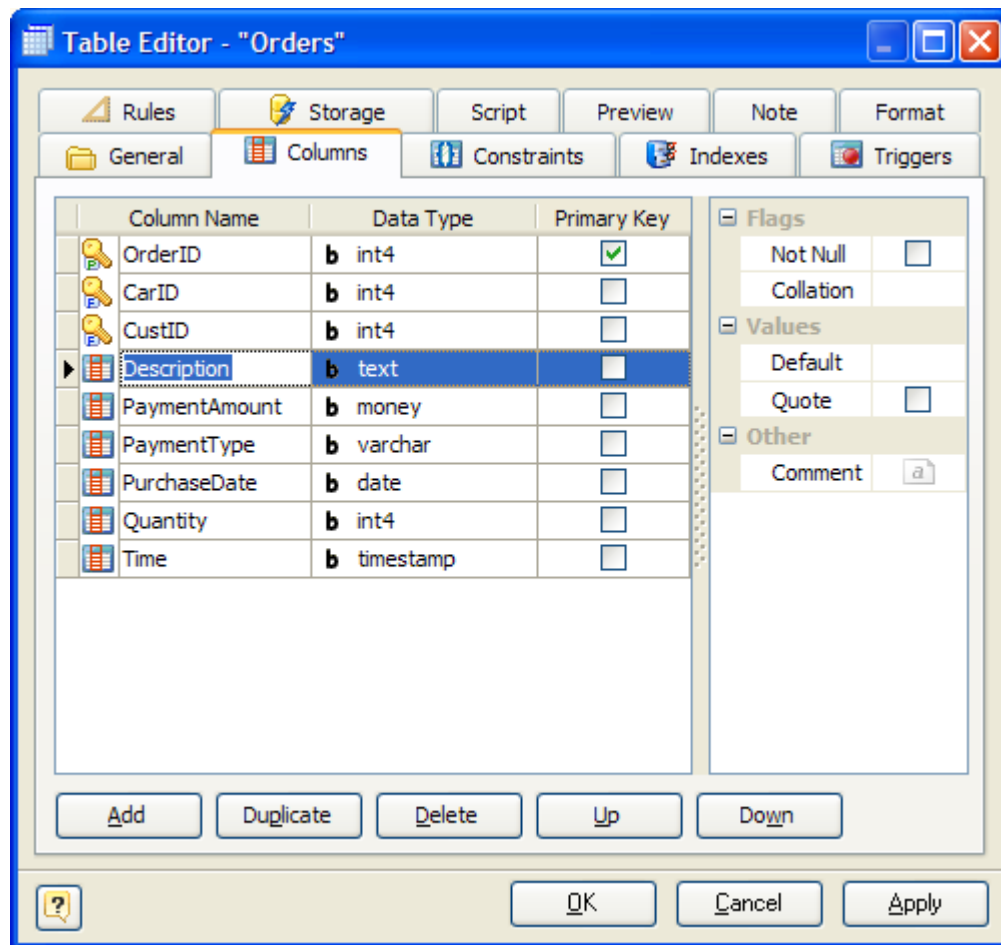
Diagram Objects: [Column Editor](#) | [Column Manager](#) | [Creating a Table](#)

7.2.3.1. Column Editor

The **Column Editor** is placed within the [Table Editor](#) dialog. It allows you to modify the list of table columns as well as column properties. Click the **Columns** tab of the **Table Editor** to manage table columns.

The **Column Editor** consists of the following areas:

- **Column List**
- **Properties Pane**
- **Button Pane**



Column List

The column list displays all the columns in the table and allows you to modify the following column properties:

- **Column name** - the name of the column, which must be unique within the table;
- **Data type** - the type of the column, which specifies data to store in the column;
- **Primary key** - specify this option to include the field into the table primary key;

Properties Pane

The properties pane allows you to define the advanced properties of the column, selected in the **Column List**. The appearance of this pane changes according to the data type of the column. These properties are:

- **Length** - this attribute defines the maximum allowed length of the stored values; it applies to all integer, decimal, and string types;
- **Decimals** - this attribute defines the number of digits, which follow the decimal point;
- **Not null** - this option indicates that the stored column value cannot be NULL;
- **Collation** option assigns a collation to the column (which must be of a collatable data type). If not specified, the column data type's default collation is used.
- **Autoinc** - this attribute makes the column value auto increment, i.e. each new value is set automatically according to the previous value; it applies to all integer values. Depending on the choice, this may generate a sequence using *SERIAL* keyword, or create *IDENTITY* column);
- **Default** - this attribute defines the default value, which the column accepts if no other is specified.
- **Quote** - this attribute defines the default value to be quoted or not.
- **Comment** - an arbitrary description for the column.

Buttons Pane

The buttons under the list of columns allow you to perform the following actions:

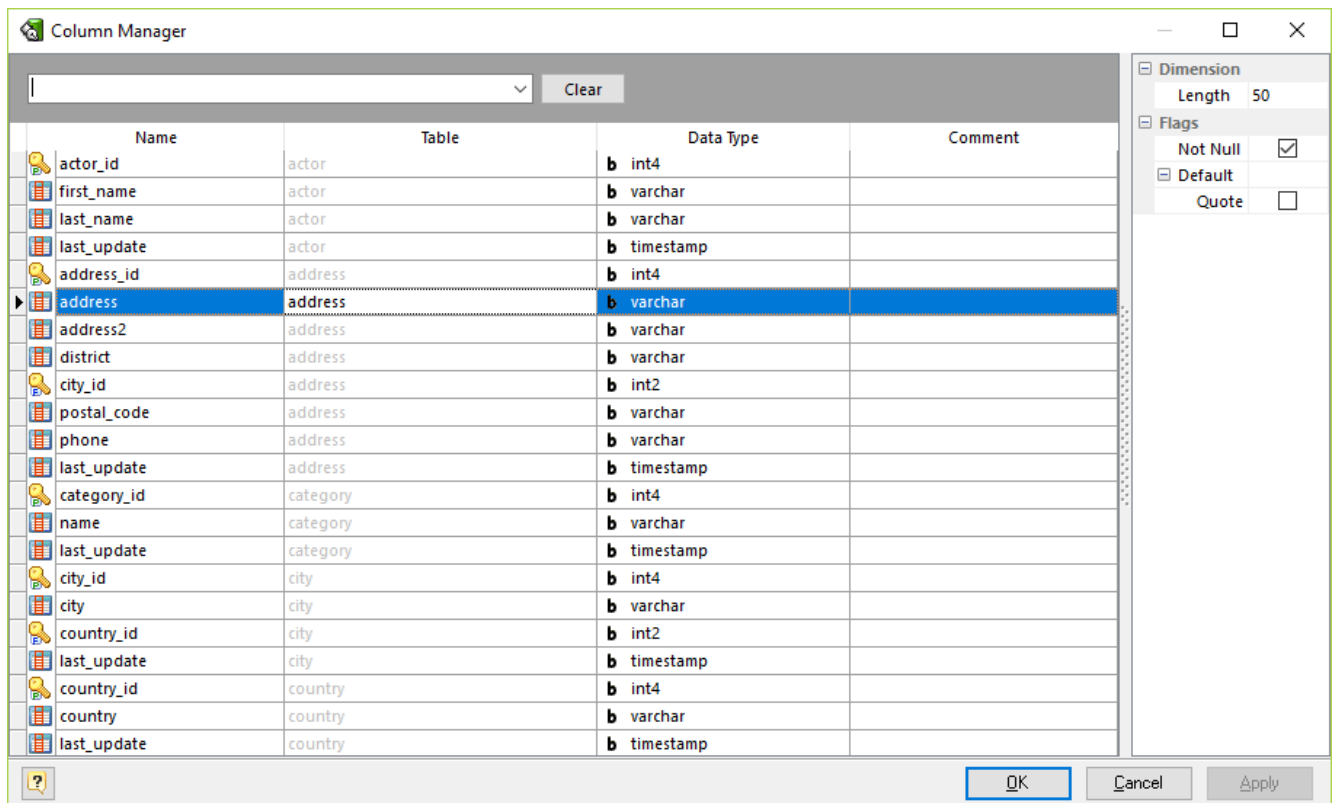
- **Add** - add a new column with the default properties to the end of the list;
- **Duplicate** - add a new column with the same properties as the selected column to the end of the list;
- **Delete** - remove the selected column from the list;
- **Up/Down** - move the selected column along the list.

See also:

Diagram Objects: [Domains](#) | [Table Editor](#) | [Indexes](#) | [Constraints](#) | [Triggers](#) | [Rules](#)

7.2.3.2. Column Manager

The **Column Manager** allows you to view and modify the basic parameters of all the table columns within the diagram. To open the **Column Manager** use the **Diagram | Column Manager** menu item.



The **Column Manager** consists of the following areas:

- **Column Grid**
- **Properties Pane**

Column Grid

The grid rows stand for the diagram columns, and the grid columns for the diagram column parameters. These parameters can be changed by using [Column Editor](#) for each table one by one, but with **Column Manager** you can do it much more quickly.

The grid allows you to modify the following column properties:

Name

The name of the column, which must be unique within the diagram;

Table

The name of the table which owns a column. This property can't be changed;

Data type

The data type of the column;

Comment

An arbitrary description for the column.

Properties Pane

The **Properties pane** allows you to define the advanced properties of the column, selected in the **Column Grid**. Please refer to [Column Editor](#) topic to see the meaning of properties.

To save your changes click the **OK** button. If you want to store changes and continue editing, click on the **Apply** button.

Searching column in the list

If your diagram contains large number of table columns, it may be important to search for a column in a most easy way. To find a column quickly by its name, press the **Ctrl + F**. Type in the name of column to find. If the column is found they will be highlighted.

Running the Column Editor

If you want to change some parameters of the column parent (e.g. add or remove columns or indexes), select the appropriate column in the dialog and press the **Ctrl + Enter**. The [Column Editor](#) will appear, where you can edit all the parameters for the selected column and for its parent table.

See also:

Diagram Objects: [Table Manager](#) | [Column Editor](#)

7.2.4. Constraints

Constraints give control over the data in your tables. If a user attempts to store data in a column that would violate a constraint, an error is raised. This applies even if the value came from the default value definition.

There are several types of constraints: check constraint, not null constraint, unique constraint, primary key constraint and foreign key constraint.

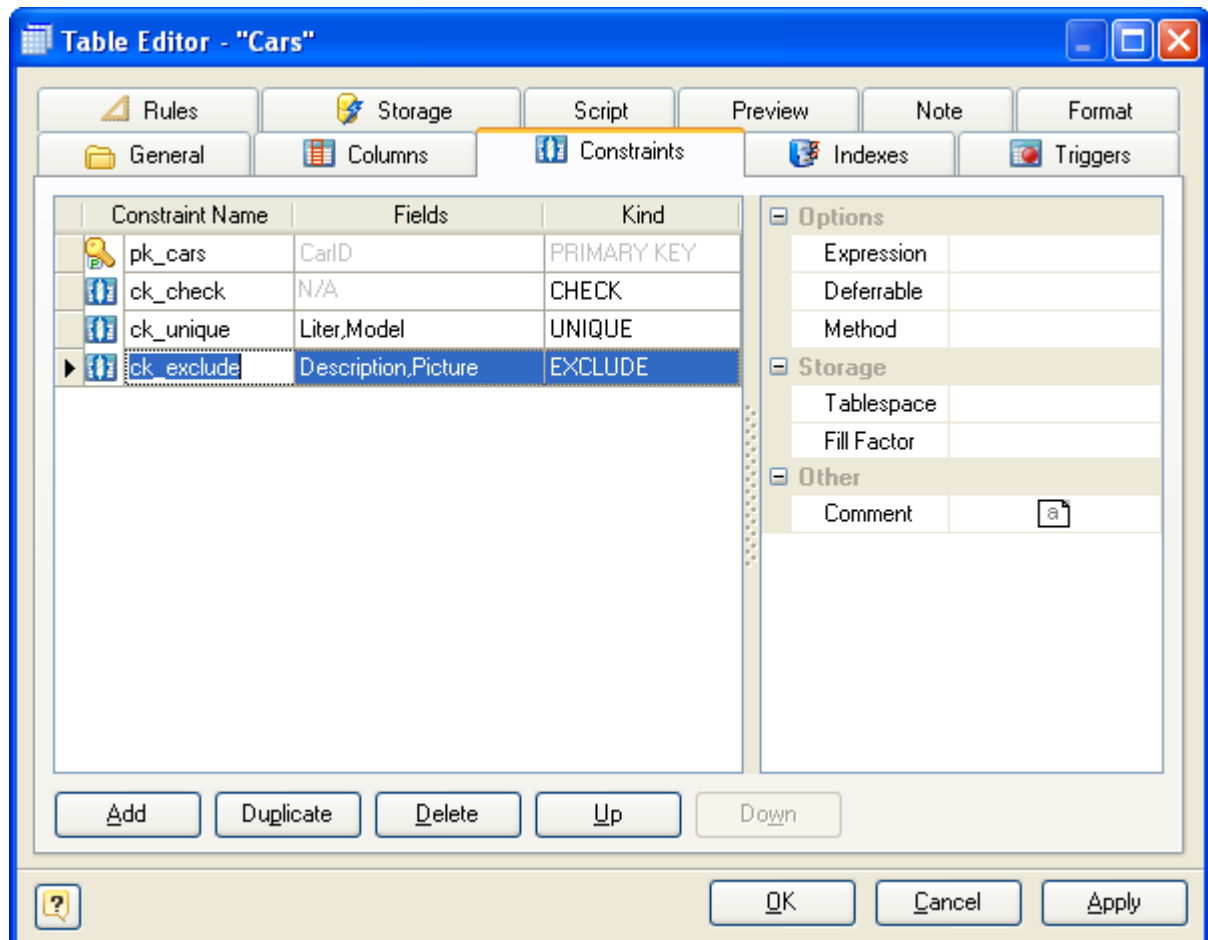
You can set a not null constraint and primary key constraint by using [Column Editor](#). A check and unique constraint can be defined using [Constraint Editor](#) and [Constraint Manager](#).

Please refer to [References and Foreign Keys](#) for information about a foreign key constraint.

7.2.4.1. Constraint Editor

Constraint Editor allows you to define and modify check, unique and primary key constraints on the table. The **Constraint Editor** is placed within the [Table Editor](#) dialog. Click the **Constraints** tab of the **Table Editor** to manage table constraints.

The main area of the editor is the list of constraints defined on table. The columns of the list allow you to modify the properties of the particular constraint.



These properties are:

Constraint Name

On optional name for the constraint. If you don't specify a name, the system chooses a name for you.

Fields

A list of columns on which the current constraint is defined. This field is actual only for primary keys, unique and exclude constraints.

💡 For EXCLUDE constraints one may add expressions not only columns on which constraint will be built. Comparison operators are specified using drop-down field editor either.

Kind

A kind of the constraint. You can choose one of the following kinds:

EXCLUDE. Exclusion constraint guarantees that if any two rows are compared on the specified column(s) or expression(s) using the specified operator(s), not all of these comparisons will return TRUE.

CHECK. Is the most generic constraint type. It allows you to specify that the value in certain columns must satisfy a Boolean (truth-value) expression.

UNIQUE. The unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

PRIMARY KEY. Simply put, it's a combination of a unique constraint and a not-null constraint. Please note, that you could define primary keys right in the [Column Editor](#).

The properties pane allows you to define the advanced properties of the constraint, selected in the Constraint List. The appearance of this pane changes according to the kind of the constraint. These properties are:

Expression

This field can be used for setting boolean expression for a check constraint. The check expression should involve the column thus constrained, otherwise the constraint would not make too much sense. For exclusion constraint it allows you to specify an constraint on a subset of the table; internally this creates a partial index. Example expression:

```
discounted_price > 0
```

Deferable

This controls whether the constraint can be deferred. A constraint that is not deferrable will be checked immediately after every command. Checking of constraints that are deferrable can be postponed until the end of the transaction. Currently, only UNIQUE, PRIMARY KEY, and EXCLUDE constraints accept this clause. CHECK constraints are not deferrable.

Method

Exclusion constraints are implemented using an index, so each specified operator must be associated with an appropriate operator class for the index access method.

Fill Factor

Specifies storage parameter for underlying index. The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages. For B-trees, leaf pages are filled to this percentage during initial index build, and also when extending the index at the right (largest key values). If pages subsequently become completely full, they will be split, leading to gradual degradation in the index's efficiency. B-trees use a default fillfactor of 90, but any value

from 10 to 100 can be selected. If the table is static then fillfactor 100 is best to minimize the index's physical size, but for heavily updated tables a smaller fillfactor is better to minimize the need for page splits. The other index methods use fillfactor in different but roughly analogous ways; the default fillfactor varies between methods.

Tablespace

The tablespace in which to create the underlying index.

FK Referenced

Read only checkbox showing if unique or primary key constraint is used in the reference join and cannot be modified.

Comment

An arbitrary description for the constraint.

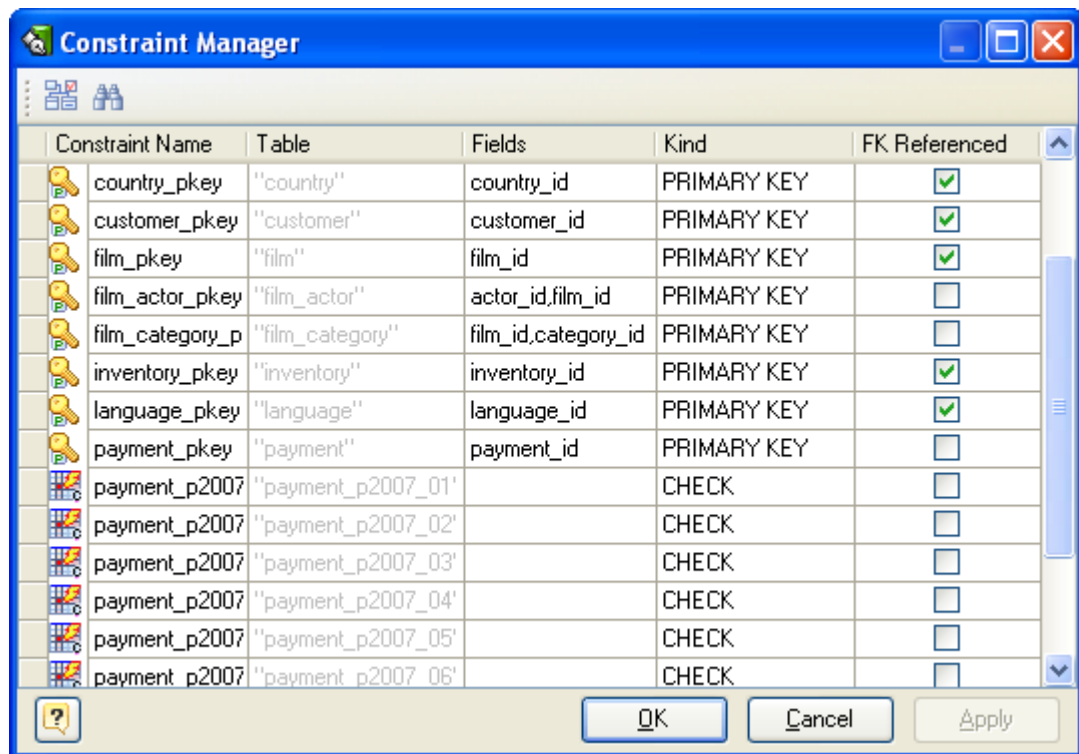
The buttons under the list allow you to perform the following actions:

- **Add** - add a new constraint to the end of the list;
- **Duplicate** - add a new constraint with the same properties as the selected constraint to the end of the list;
- **Delete** - remove the selected constraint from the list;
- **Up/Down** - move the selected constraint along the list.

7.2.4.2. Constraint Manager

Constraint Manager can be used to manage all check, primary and unique constraints within the diagram. To open **Constraint Manager** use the **Diagram | Constraint Manager** menu item.

The main area of the manager is a grid that represents all the constraints and their properties defined in the diagram excluding foreign key constraints and not null constraints. These properties can be changed by using [Constraint Editor](#) for each constraint one by one, but using **Constraint Manager** helps you to do it much more quickly.



You can modify the constraint properties by using the grid:

Constraint Name

Optional name for the constraint. If you don't specify a name, the system chooses a name for you.

Table

The name of the table on which the constraint is defined. This property can't be changed.

Kind

A kind of the constraint. You can choose one of the following kinds:

CHECK. Is the most generic constraint type. It allows you to specify that the value in certain columns must satisfy a Boolean (truth-value) expression.

UNIQUE. The unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

PRIMARY KEY. Simply put, it's a combination of a unique constraint and a not-null constraint. Please note, that you could define primary keys right in the [Column Editor](#).

Fields

A list of columns on which the current constraint is defined. This field is actual only for primary key and unique constraints.

To save your changes click the **OK** button. If you want to store changes and continue editing, click on the **Apply** button.

7.2.5. Indexes

In diagrams, you can create, edit, or delete table indexes, which gain fast access to specific information in a table. As a general rule, you should create an index on a table only if the data in the indexed columns will be queried frequently. Indexes take up disk space and slow the adding, deleting, and updating of rows. In most situations, the speed advantages of indexes for data retrieval greatly outweigh these disadvantages. However, if your application updates data very frequently or if you have disk space constraints, you might want to limit the number of indexes.

Before creating an index, you must determine what columns to use and what type of index to create.

You can create indexes based on a single column or on multiple columns in a database table. Multiple-column indexes enable you to distinguish between rows in which one column may have the same value. Indexes are also helpful if you often search or sort by two or more columns at a time. For example, if you often set criteria for last name and first name columns in the same query, it makes sense to create a multiple-column index on those two columns. To determine the usefulness of an index:

- Examine the WHERE and JOIN clauses of your queries. Each column included in either clause is a possible candidate for an index.
- Experiment with the new index to examine its effect on the performance of running queries.
- Consider the number of indexes already created on your table. It is best to avoid a large number of indexes on a single table.
- Examine the definitions of the indexes already created on your table. It is best to avoid overlapping indexes that contain shared columns.
- Examine the number of unique data values in a column and compare that number with the number of rows in the table. The result is the selectivity of that column, which can help you decide if a column is a candidate for an index and, if so, what type of index.

To modify table indexes, use [Index Editor](#).

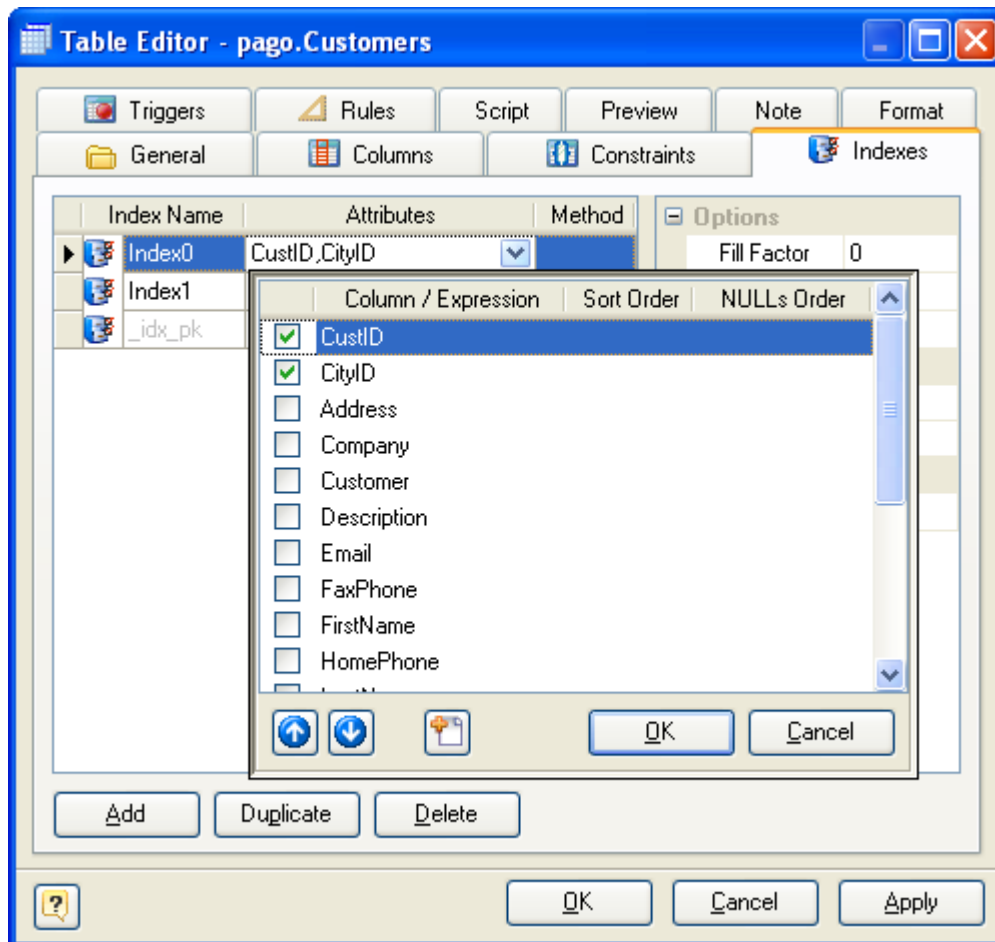
See also:

Diagram Objects: [Index Editor](#) | [Index Manager](#)

7.2.5.1. Index Editor

The **Index Editor** is placed within the [Table Editor](#) dialog. It allows you to modify the list of table indexes as well as index properties.

The main element of the editor is the index list, which displays all indexes available within the table. The columns of the list allow you to modify the properties of the selected index.



These properties are:

Index Name

The name of the index, which must be unique within the table.

Attributes

The names of a columns of the table or expressions based on one or more columns of the table. The expression must not be written with surrounding parentheses, because Designer will add them automatically.

Method

The name of the method to be used for the index. Choices are btree, hash, rtree, and gist. The default method is btree.

Sort Order

Specifies index attributes sort order.



Since an ordered index can be scanned either forward or backward, it is not normally useful to create a single-column DESC index — that sort ordering is already available with a regular index. The value of these options is that multicolumn indexes can be created that match the sort ordering requested by a mixed-ordering query, such as `SELECT ... ORDER BY x ASC, y DESC`.

NULLs Order

Specifies that nulls sort before or after non-nulls.



The NULLs Order options are useful if you need to support “nulls sort low” behavior, rather than the default “nulls sort high”, in queries that depend on indexes to avoid sorting steps.

Predicate

The constraint expression for a partial index.

Tablespace

The tablespace in which to create the index. If not specified, default is used, or the database's default tablespace if server's parameter `default_tablespace` is an empty string.

Fast Update

This setting controls usage of the fast update technique for GIN indexes. Updating a GIN index tends to be slow because of the intrinsic nature of inverted indexes: inserting or updating one heap row can cause many inserts into the index (one for each key extracted from the indexed value). As of PostgreSQL 8.4, GIN is capable of postponing much of this work by inserting new tuples into a temporary, unsorted list of pending entries. See PostgreSQL manual for details.

Fillfactor

The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages. For B-trees, leaf pages are filled to this percentage during initial index build, and also when extending the index at the right (largest key values). If pages subsequently become completely full, they will be split, leading to gradual degradation in the index's efficiency. B-trees use a default fillfactor of 90, but any value from 10 to 100 can be selected. If the table is static then fillfactor 100 is best to minimize the index's physical size, but for heavily updated tables a smaller fillfactor is better to minimize the need for page splits. The other index methods use fillfactor in different but roughly analogous ways; the default fillfactor varies between methods.

Unique

Defines the UNIQUE constraint for the selected columns. I.e. the combination of the included field values must be unique within the table;

System

An indicator that shows if the index is system and can't be modified by a user.

Comment

Specifies comment for index object.

The buttons under the list of indexes allows you to perform the following actions:

- **Add** - add a new index with the default properties to the end of the list;
- **Duplicate** - add a new index with the same properties as the selected index to the end of the list;
- **Delete** - remove the selected index from the list;
- **Up/Down** - move the selected index along the list.

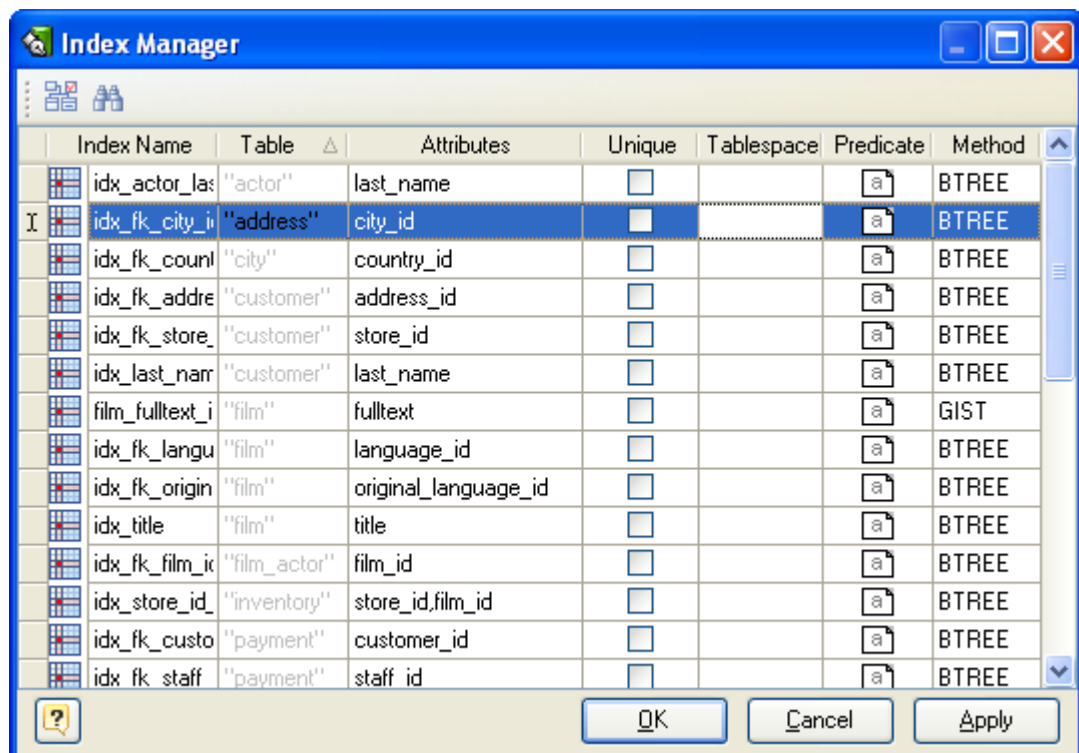
See also:

Diagram Objects: [Index Manager](#)

7.2.5.2. Index Manager

The **Index Manager** allows you to view and modify the basic parameters of all the table indexes within the diagram.

To open the **Index Manager** use the **Diagram | Index Manager** menu item.




Modifying parameter values

This grid represents the basic parameters of all diagram indexes. The grid rows stand for the diagram indexes, and the grid columns for the index parameters. These parameters can be changed by using [Index Editor](#) for each table one by one, but with **Index Manager** you can do it much more quickly.


Click on the cell to modify the parameter value. Depending on the parameter type, the activated in-place editor can be a text box, a drop-down list, a check box, etc. Please refer to the [Index Editor](#) topic to find out more about all index parameters. Note, that the **Table** parameter indicates the name of the table a column belongs to, and cannot be changed.

To save your changes click the **OK** button. If you want to store changes and continue editing, click on the **Apply** button.

Searching index in the list

If your diagram contains large number of table indexes, it may be important to search for an index in a most easy way. To find an index quickly by its name, click the **Find** button () at the dialog toolbar. The **Find Name** dialog will appear. Type in the name of an index to find and click **OK**. If the index is found the grid cursor positions on the appropriate row.

Running the Index Editor

If you want to change some parameters of the index parent (e.g. add or remove columns or indexes), select the appropriate index in the dialog and click the **Edit** button () of the dialog toolbar. The [Index Editor](#) will appear, where you can edit all the parameters for the selected index and for its parent table.

See also:

Diagram Objects: [Index Editor](#)

7.2.6. Triggers

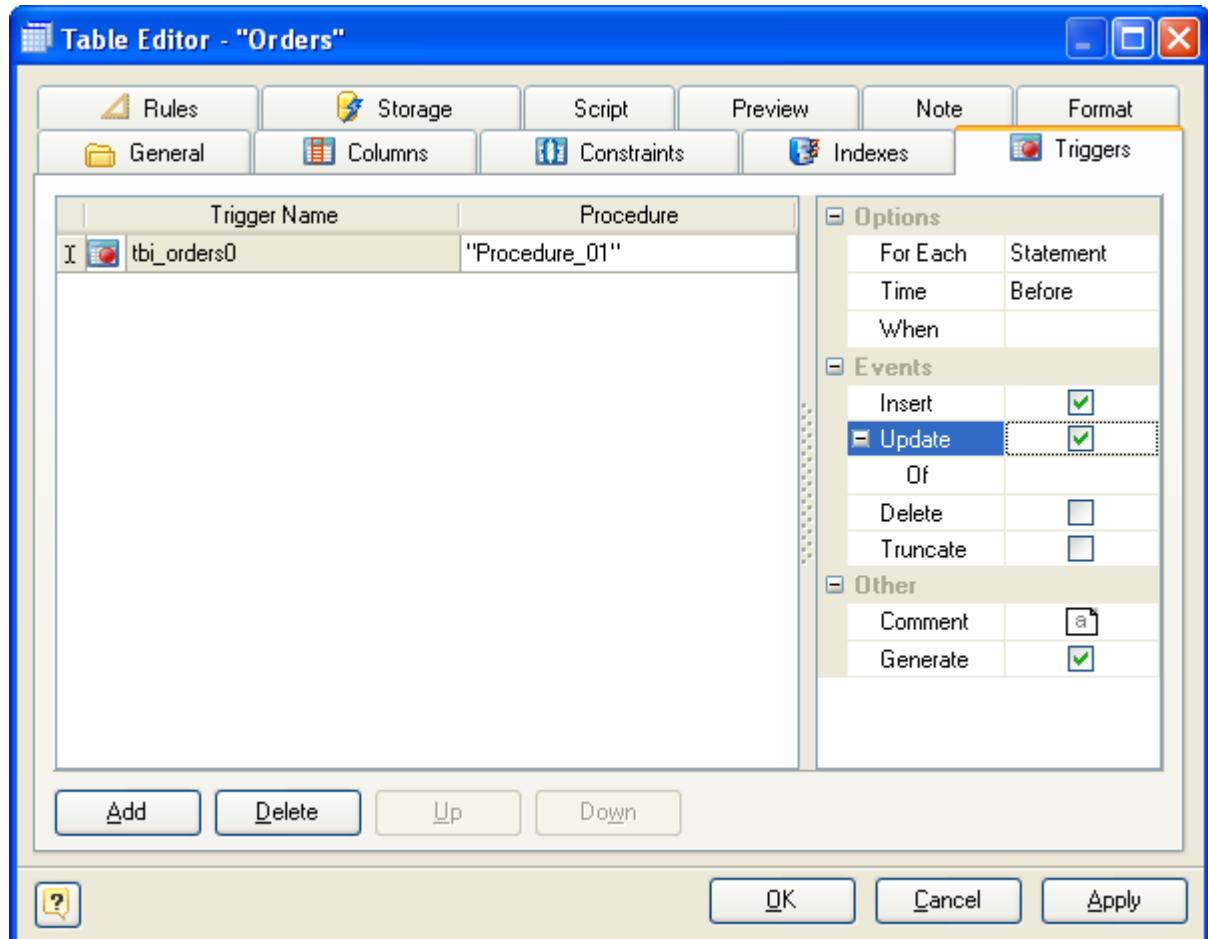
A trigger is a named database object that is associated with a table and that activates when a particular event occurs for the table. Activation of a trigger means execution of a set of commands attached to that trigger. For example, BEFORE trigger for INSERT statements could be used to check the values to be inserted into new rows.

Trigger functions can be written in C or in any of the available procedural languages. It is not currently possible to write a trigger function in SQL. The trigger function must be defined before the trigger itself can be created. The [trigger function](#) must be declared as a function taking no arguments and returning type trigger. In the **Database Designer for PostgreSQL** such functions are called procedures.

With **Database Designer for PostgreSQL** you can [edit](#) table triggers.

7.2.6.1. Trigger Editor

Trigger Editor is placed within the [Table Editor](#) dialog. It allows you to modify table triggers.



The main element of the editor is the trigger list, which displays all the triggers in the table and their properties. These properties are as follows:

Trigger Name

The name of the trigger. It must be different from any other trigger name of the same table.

Procedure

Select [stored procedure](#) from the list of the defined ones within the diagram. This function will be executed when the trigger fires.

For Each

It specifies whether the trigger procedure will be fired once for every row affected by the trigger event, or just once per SQL statement. If neither is specified, then the trigger procedure FOR EACH STATEMENT is the default.

Time

Determines whether the function is called before or after the event.


When

A boolean expression that determines whether the trigger function will actually be executed. If WHEN is specified, the function will only be called if the condition returns true. In FOR EACH ROW triggers, the WHEN condition can refer to columns of the old and/or new row values by writing OLD.column_name or NEW.column_name respectively. Of course, INSERT triggers cannot refer to OLD and DELETE triggers cannot refer to NEW.

 Currently, WHEN expressions cannot contain subqueries.

Events

Indicates the kind of statement that activates the trigger. It can be INSERT, UPDATE, DELETE or TRUNCATE.

 For UPDATE triggers, it is possible to specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the update.

Comment

Defines comment for trigger object.

Generate

Enables trigger creation during [Database Generation](#) and [Database Modification](#).

The buttons under the list of triggers allow you to perform the following actions:

- **Add** - add a new trigger to the end of the list;
- **Delete** - remove the selected trigger from the list;
- **Up/Down** - move the selected trigger along the list.

See also:

[Table Editor](#) | [Triggers](#)

7.2.7. Rules

The PostgreSQL rule system allows one to define an alternate action to be performed on insertions, updates, or deletions in database tables. Roughly speaking, a rule causes additional commands to be executed when a given command on a given table is executed. Alternatively, an INSTEAD rule can replace a given command by another, or cause a command not to be executed at all.

Rules are used to implement table views as well. It is important to realize that a rule is really a command transformation mechanism, or command macro. The transformation happens before the execution of the commands starts. If you actually want an operation that fires independently for each physical row, you probably want to use a trigger, not a rule.

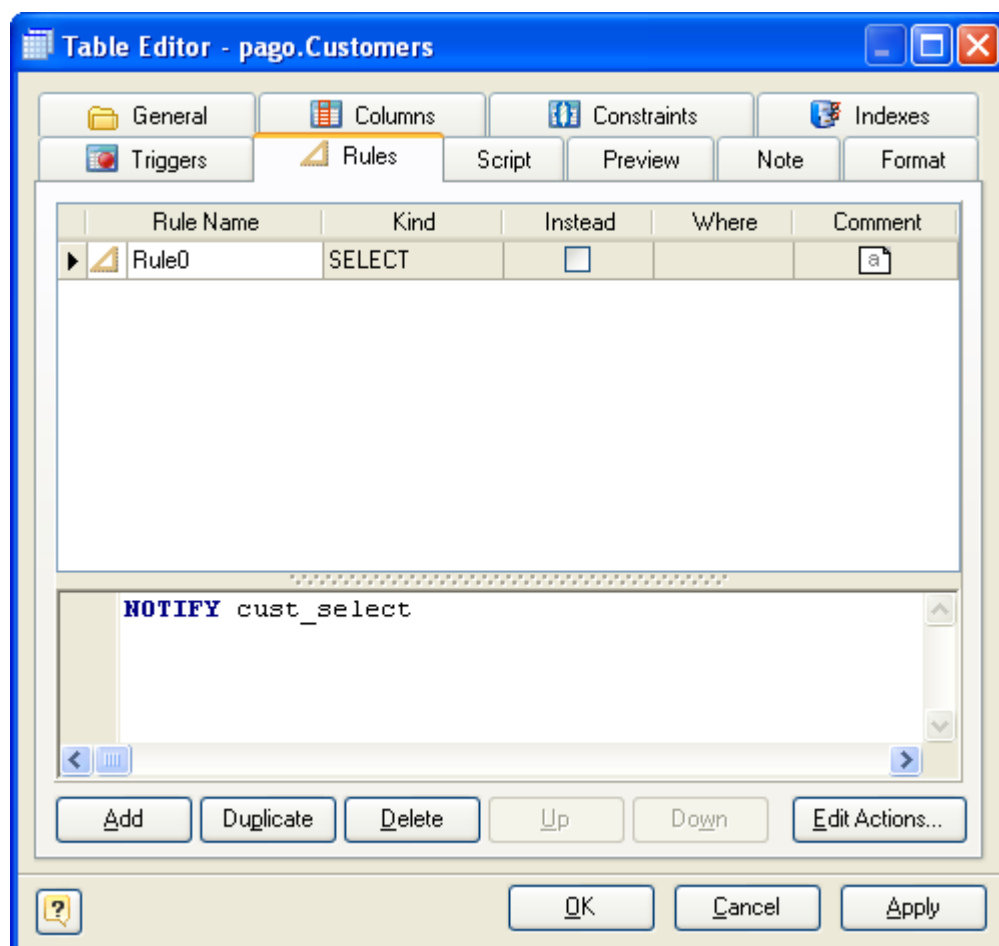
See also:

Database Designer for PostgreSQL allows to [edit table rules](#).

Diagram Objects: [Rule Editor](#)

7.2.7.1. Rule Editor

Rule Editor is placed within the [Table Editor](#) dialog. It allows you to modify table rules.



The main element of the editor is the rules list, which displays all the rules in the table and their properties. These properties are as follows:

Rule Name

The name of the rule. It must be different from any other rule name of the same table.

Kind

Indicates the kind of statement that activates the rule. It can be SELECT, UPDATE, INSERT or DELETE.

Instead

Indicates should rule actions be done instead of statement that activates rule.

Where

Any SQL conditional expression (returning boolean). The condition expression may not refer to any tables except NEW and OLD, and may not contain aggregate functions. NEW is valid in ON INSERT and ON UPDATE rules to refer to the new row being inserted or updated. OLD is valid in ON UPDATE and ON DELETE rules to refer to the existing row being updated or deleted.

Comment

Defines comment for rule object.

Actions memo

The command or commands that make up the rule action. Valid commands are SELECT, INSERT, UPDATE, DELETE, or NOTIFY.

The buttons under the list of rules allow you to perform the following actions:

- **Add** - add a new ruler to the end of the list;
- **Delete** - remove the selected rule from the list;
- **Up/Down** - move the selected rule along the list.

See also:

[Table Editor](#) | [Rules](#)

7.2.8. Formatting Table

You can set table line and fill color for displaying on the diagram, different from the default table colors, which are defined within the [Diagram Display Preferences](#) dialog.

Double click on the table symbol and [Table Editor](#) will appear. Go to the **Format** tab.

Fill Color

Use this option to set a background color of the table. Choose the color you need from this drop-down list.

Line Color

Use this option to set a border color of the table. Choose the color you need from this drop-down list.

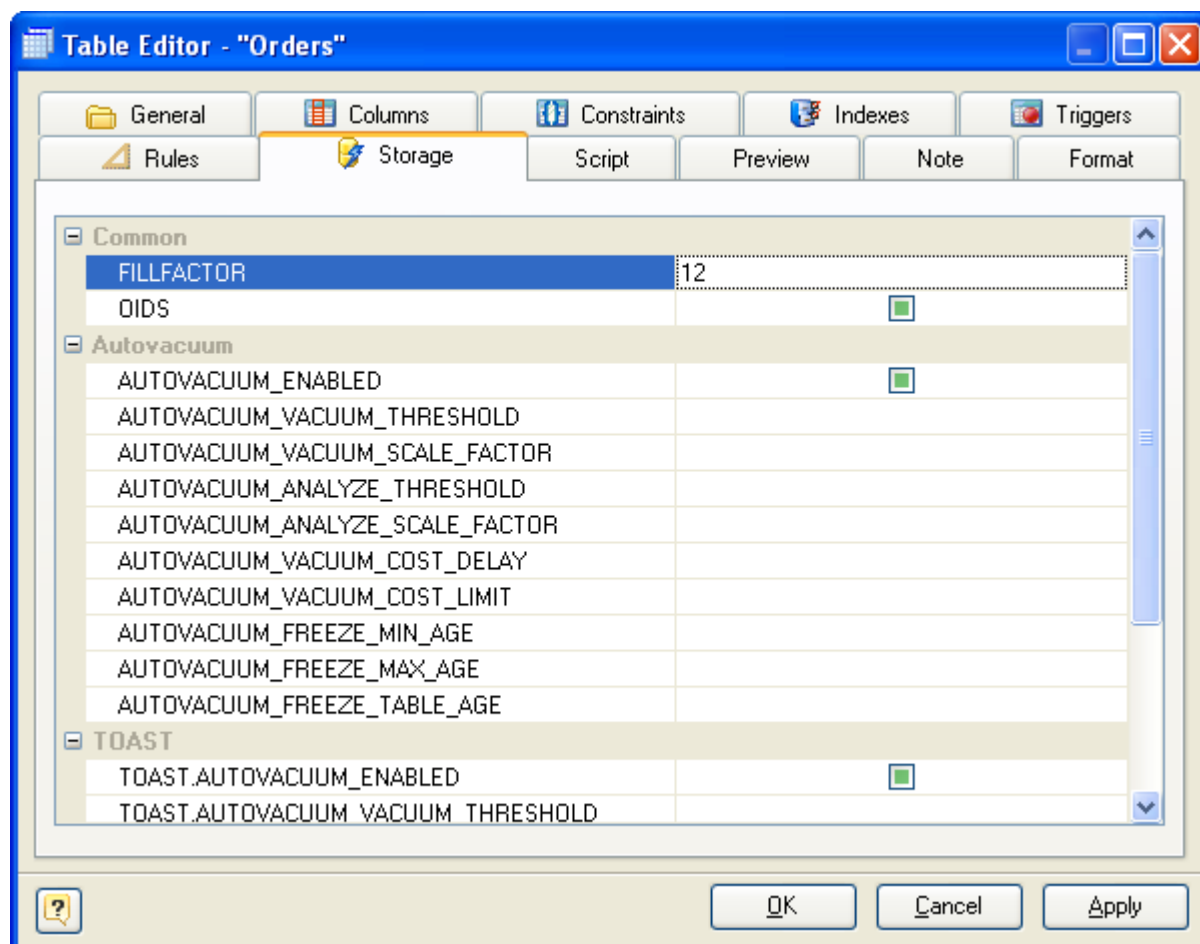
See also:

Diagram Objects: [Table Editor](#)

Diagram: [Diagram Display Preferences](#)

7.2.9. Storage Parameters

You can set and specify storage parameters for tables. The storage parameters currently available for tables are listed below. For each parameter, unless noted, there is an additional parameter with the same name prefixed with *toast.*, which can be used to control the behavior of the table's secondary TOAST table, if any. Note that the TOAST table inherits the *autovacuum_** values from its parent table, if there are no *toast.autovacuum_** settings set. For details see the PostgreSQL manual.



fillfactor

The fillfactor for a table is a percentage between 10 and 100. 100 (complete packing) is the default. When a smaller fillfactor is specified, INSERT operations pack table pages only to the

indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables smaller fillfactors are appropriate. This parameter cannot be set for TOAST tables.

autovacuum_enabled, toast.autovacuum_enabled

Enables or disables the autovacuum daemon on a particular table.

autovacuum_vacuum_threshold, toast.autovacuum_vacuum_threshold

Minimum number of updated or deleted tuples before initiate a VACUUM operation on a particular table.

autovacuum_vacuum_scale_factor, toast.autovacuum_vacuum_scale_factor

Multiplier for reltuples to add to autovacuum_vacuum_threshold.

autovacuum_analyze_threshold

Minimum number of inserted, updated, or deleted tuples before initiate an ANALYZE operation on a particular table.

autovacuum_analyze_scale_factor

Multiplier for reltuples to add to autovacuum_analyze_threshold.

autovacuum_vacuum_cost_delay, toast.autovacuum_vacuum_cost_delay

See PostgreSQL manual.

autovacuum_vacuum_cost_limit, toast.autovacuum_vacuum_cost_limit

See PostgreSQL manual.

autovacuum_freeze_min_age, toast.autovacuum_freeze_min_age

See PostgreSQL manual.

autovacuum_freeze_max_age, toast.autovacuum_freeze_max_age

See PostgreSQL manual.

autovacuum_freeze_table_age, toast.autovacuum_freeze_table_age

See PostgreSQL manual.

See also:

Diagram Objects: [Table Editor](#)

7.2.10. SQL Table Definition

Table SQL preview

You can see the SQL representation of your table, it includes columns, indexes, foreign keys, etc.

To see SQL representation of the table, double click on the table symbol and [Table Editor](#) will appear. Go to the **Preview** tab. SQL syntax elements are colored. It is possible to copy this definition to clipboard.

See also:

Diagram Objects: [Table Editor](#)

Diagram: [Diagram SQL Preview](#)

7.2.11. Tables in Tree View Window

You can access to the diagram tables and browse through their objects by mean of the [Object Tree View](#), that represent all the diagram objects, including tables, in a tree-like structure.

To expand table, click on the + sign near the table name, it will show you table columns and indexes. Click on the "-" sign to roll up the list.

Examine the following table-related features in the [Object Tree View](#):

- Double click on a table to call [Table Editor](#) for it.
- Right-click on a table to see shortcut menu. Select the **Goto** object item to select the table on the diagram and scroll diagram to it.
- Double-click on a column to call the [Column Editor](#).
- Double-click on a constraint to call the [Constraint Editor](#).
- Double-click on an index to call the [Index Editor](#).
- Double-click on a trigger to call the [Trigger Editor](#).
- Double-click on a rule to call the [Rule Editor](#).

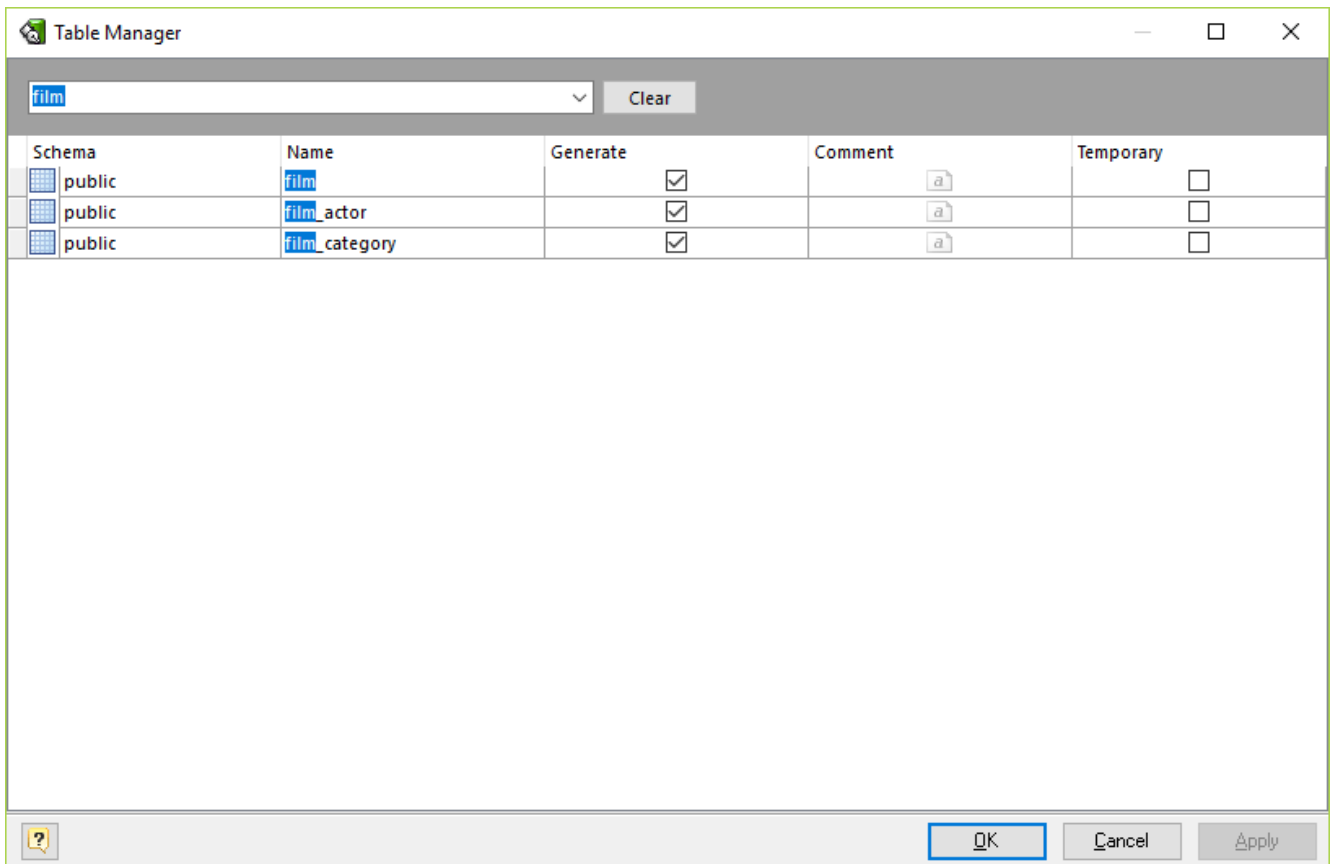
See also:

Workspace: [Docking Windows](#)

Diagram Objects: [Column Editor](#) | [Table Editor](#) | [Index Editor](#)

7.2.12. Table Manager

Database Designer for PostgreSQL has a great feature that allows you to modify the basic parameters for multiple tables at once. This is the **Table Manager**. To run this tool use the **Diagram | Table Manager** menu item.



Modifying parameter values

The grid in **Table Manager** represents the basic parameters of all diagram tables. The rows stand for the tables, and columns for the table parameters. These parameters can be changed by using [Table Editor](#) for each table one by one, but with **Table Manager** you can do it much more quickly.

Click on the cell to modify the parameter value. Depending on the parameter type, the activated in-place editor can be a text box, a drop-down list, a check box, etc. Please refer to the [Table Editor](#) topic to find out more about all table parameters.

To save your changes click the **OK** button. If you want to store changes and continue editing, click on the **Apply** button.

Searching table in the list

If your diagram contains large number of tables, it may be important to search for a table in a most easy way. To find a table quickly by its name, press the **Ctrl + F**. Type in the name of table to find. Found tables will be filtered and highlighted.

Running the Table Editor

If you want to change some unavailable table parameters and attributes (such as columns and indexes), select the appropriate table in the dialog and press **Ctrl + Enter**. The [Table Editor](#) will appear, where you can edit all the parameters for the selected table.

See also:

Diagram Objects: [Table Editor](#) | [Column Manager](#) | [Domain Manager](#) | [Index Manager](#)

7.3. Stored Procedures and Functions

A stored function is a set of statements that can be stored in the server. Once this has been done, clients don't need to keep reissuing the individual statements but can refer to the stored procedure instead.

Database Designer for PostgreSQL helps to build and manage stored procedures and functions in a powerful graphical environment.


With **Database Designer for PostgreSQL** you can:

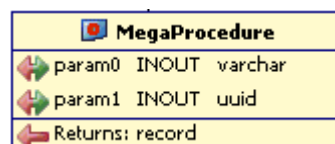
- [Create](#) and [Edit](#) stored procedure or function;
- [Manage all stored procedures](#) and functions within the diagram.

See also:

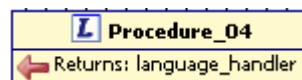
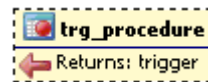
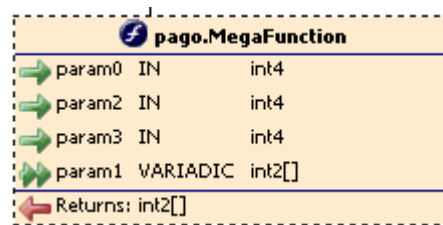
[Creating a Stored Procedure or Function](#) | [Stored Routine Editor](#) | [Stored Routine Manager](#)

7.3.1. Creating a Stored Procedure or Function

To create a new procedure (later on you can switch it to a stored function), click on the **Stored Procedure** () icon on the **Palette** toolbar. Your mouse cursor will change its appearance. Click on the diagram area to create a new stored procedure. A rectangle with the name of the stored procedure will appear in the diagram:



Visual representation may vary: if an object has at least one output parameter, then it will be shown as above (procedure notation). If return type is trigger, then it will be shown in trigger procedure notation. If return type is *language_handler*, then it will be shown in **Language Handler** procedure notation. In other cases it will have function notation representation.



Basically, PostgreSQL has only functions. To edit parameters of the stored procedures and functions use [Stored Routine Editor](#).

See also:

[Stored Procedures and Functions](#) | [Stored Routine Manager](#)

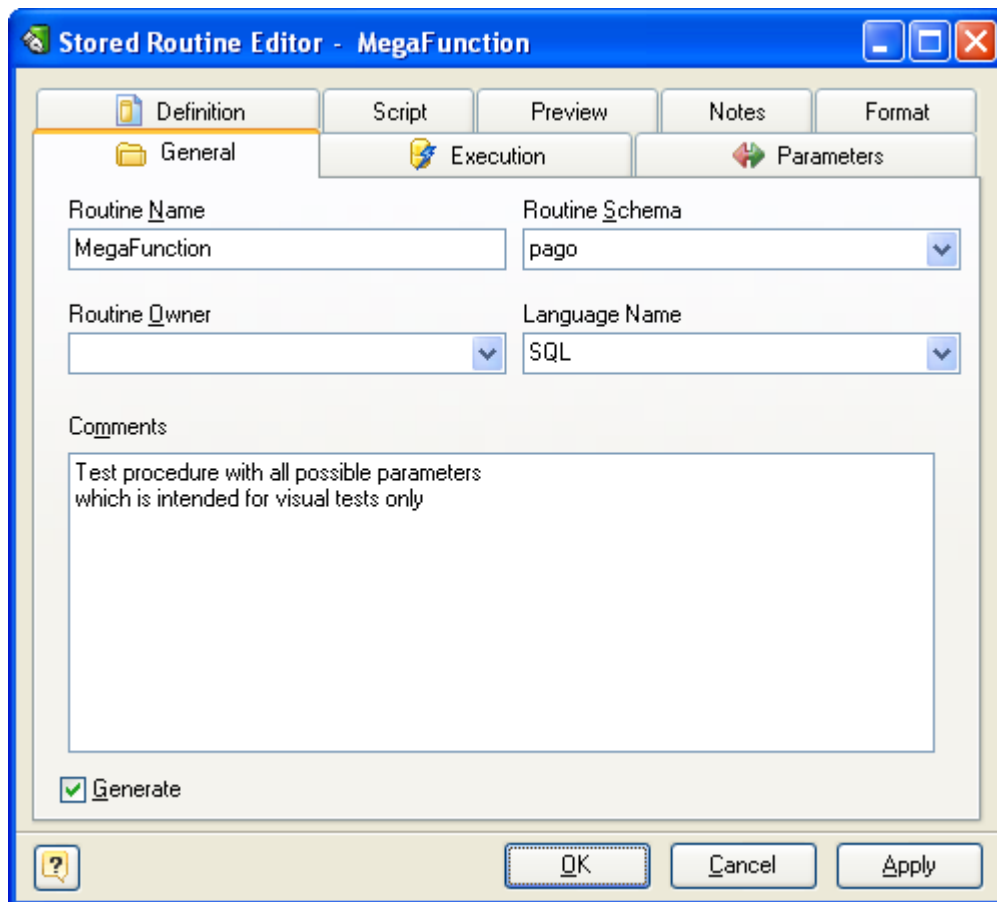
7.3.2. Stored Routine Editor

The **Stored Routine Editor** is provided for altering stored procedures/functions. All the parameters of stored procedure are valid for stored function, so you can switch between them easily.

To open the the **Stored Routine Editor**, simply double-click a stored procedure or function on the diagram or select the Properties item from the context menu.

The **Stored Routine Editor** consists of several tabs, each of which will be described below.

General



This tab allows you to tune the basic properties of a stored procedure or function.

Routine Name

The name of the stored routine must be unique within a schema. To check your diagram for the uniqueness of names use the [Check Diagram tool](#).

Routine Schema

A name of a database schema where the stored routine will be placed.

Routine Owner

A role which will be owner of the object, or a role which will execute CREATE script in case of empty input field.

Language Name

The name of the language that the function is implemented in. May be SQL, C, internal, or the name of a user-defined procedural language.

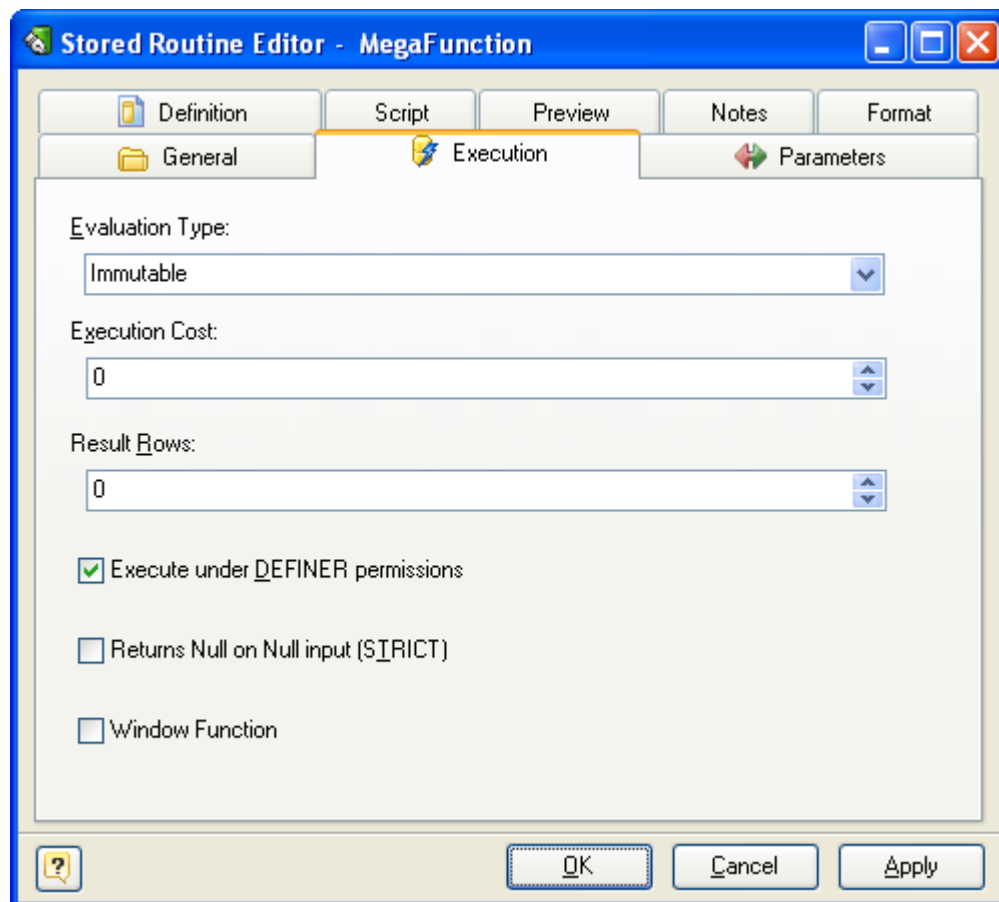
Comment

A native database comment for your routine.

Generate

Set this option off to exclude the routine from the default selection in the [Database Generation](#) and [Database Modification](#) tools.

Execution



Evaluation Type

This option informs the system if it is safe to replace multiple evaluations of the function with a single evaluation, for run-time optimization. At most one choice may be specified. If none of these appear, VOLATILE is the default assumption.

IMMUTABLE indicates that the function always returns the same result when given the same argument values.

STABLE indicates that within a single table scan the function will consistently return the same result for the same argument values, but that its result could change across SQL statements.

VOLATILE indicates that the function value can change even within a single table scan, so no optimizations can be made.

Parallel Mode

UNSAFE indicates that the function can't be executed in parallel mode and the presence of such a function in an SQL statement forces a serial execution plan. This is the default.

RESTRICTED indicates that the function can be executed in parallel mode, but the execution is restricted to parallel group leader.

SAFE indicates that the function is safe to run in parallel mode without restriction.

Functions should be labeled parallel unsafe if they modify any database state, or if they make changes to the transaction such as using sub-transactions, or if they access sequences or attempt to make persistent changes to settings (e.g. *setval*). They should be labeled as parallel restricted if they access temporary tables, client connection state, cursors, prepared statements, or miscellaneous backend-local state which the system cannot synchronize in parallel mode (e.g. *setseed* cannot be executed other than by the group leader because a change made by another process would not be reflected in the leader). In general, if a function is labeled as being safe when it is restricted or unsafe, or if it is labeled as being restricted when it is in fact unsafe, it may throw errors or produce wrong answers when used in a parallel query. C-language functions could in theory exhibit totally undefined behavior if mislabeled, since there is no way for the system to protect itself against arbitrary C code, but in most likely cases the result will be no worse than for any other function. If in doubt, functions should be labeled as UNSAFE, which is the default.

Execution Cost

A positive number giving the estimated execution cost for the function, in units of *cpu_operator_cost* configuration parameter. If the function returns a set, this is the cost per returned row. If the cost is not specified, 1 unit is assumed for C-language and internal functions, and 100 units for functions in all other languages. Larger values cause the planner to try to avoid evaluating the function more often than necessary.

Result Rows

A positive number giving the estimated number of rows that the planner should expect the function to return. This is only allowed when the function is declared to return a set (see Parameters tab). If function return only one row value is ignored. The default assumption is 1000 rows.

Execute under DEFINER permissions

This option can be used to specify whether the routine should be executed using the permissions of the user who creates the routine or the user who invokes it. The default value is DEFINER. The creator or invoker must have permission to access the database with which the routine is associated.

On NULL Arguments behavior

Determines how the function works with null input values. Set RETURNS NULL ON NULL INPUT or STRICT if you want the function always returns null whenever any of its arguments are null.

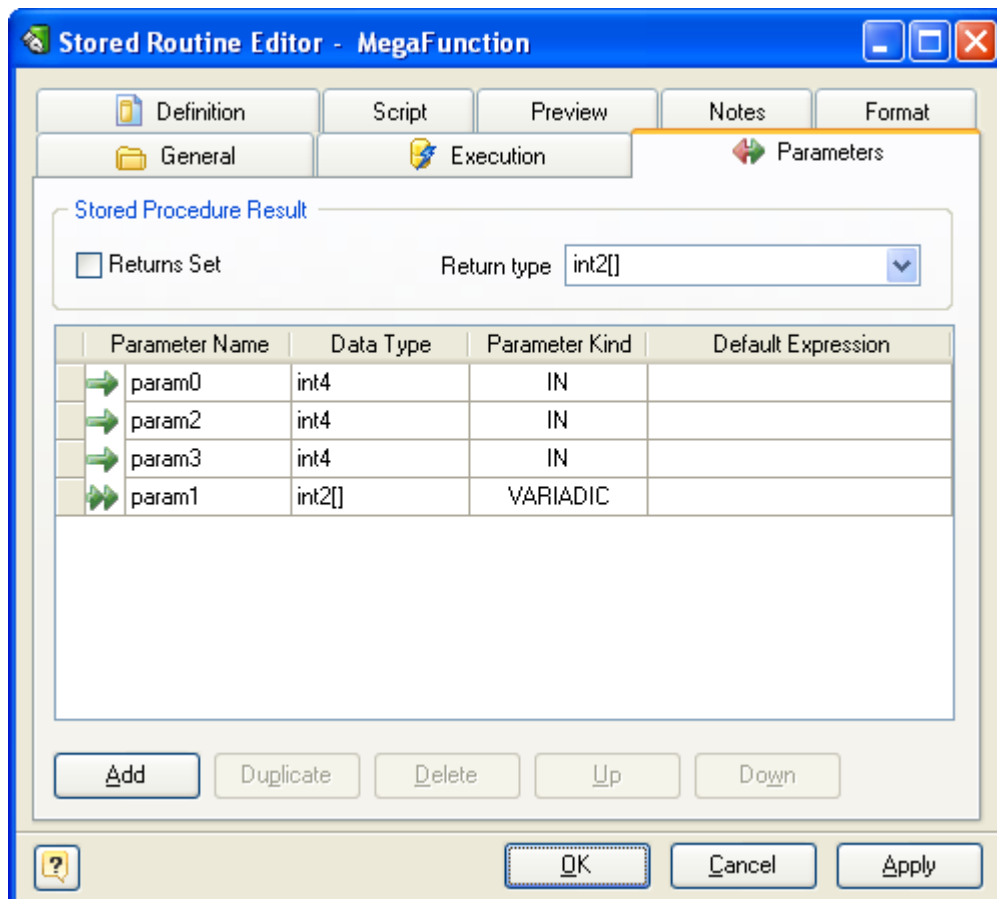
Window Function

Indicates that the function is a window function rather than a plain function. This is currently only useful for functions written in C. The WINDOW attribute cannot be changed when replacing an existing function definition.

LeakProof Function

Indicates that the function has no side effects. It reveals no information about its arguments other than by its return value. For example, a function which throws an error message for some argument values but not others, or which includes the argument values in any error message, is not leakproof. The query planner may push leakproof functions (but not others) into views created with the security_barrier option. This option can only be set by the superuser.

Parameters



The **Parameters** tab allows you to choose the routine result type and determine the list of parameters.

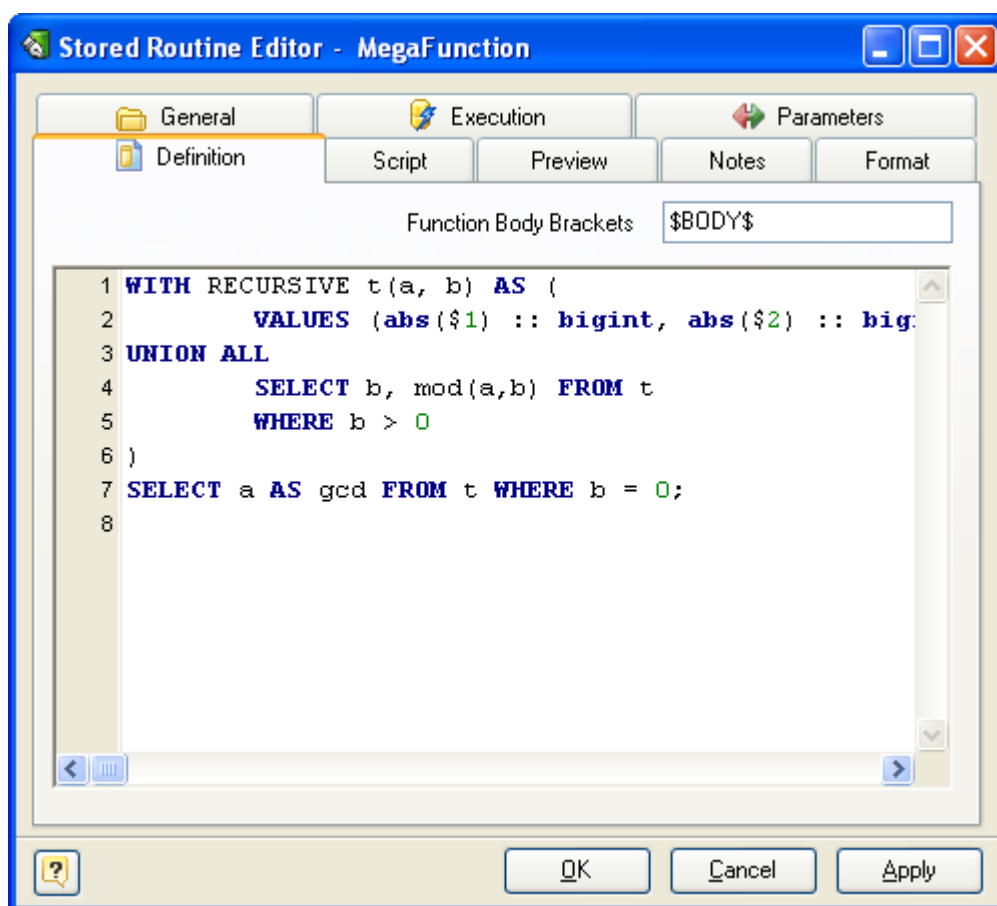
Return type

It sets the type of variable being returned by the function. If there are at least two OUT parameters, then changes return type to RECORD.

Parameters

This grid contains the list of all procedure parameters. Buttons pane below the parameters list work in the same way as the [Column Editor](#) one.

Definition



The **Definition** tab allows you to edit routine text. It will be visible only if the **Language Name** is set to SQL or to one of the user-defined procedural languages, e.g. **PIPgSQL**.

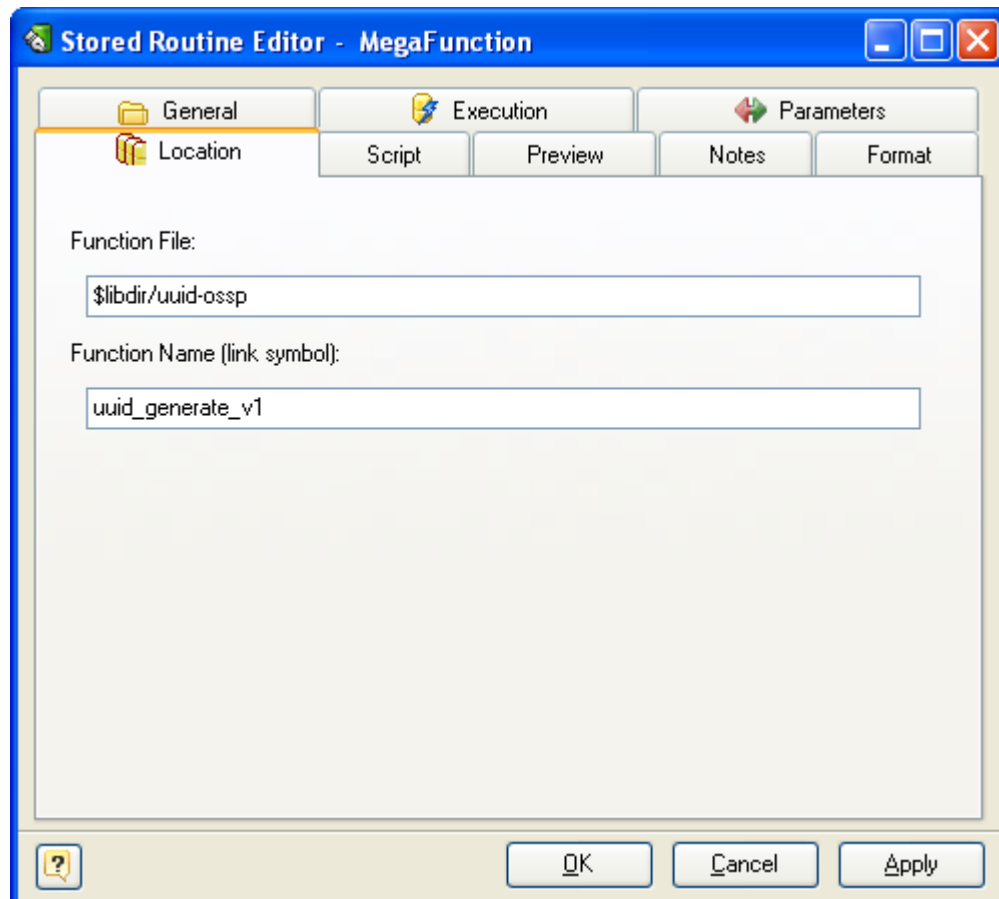
Function Body Brackets

Dollar-Quoted string surrounding function body text, or quote character.

Routine Body Memo

This text editor represents the routine text, which appears between BEGIN and END keywords in CREATE FUNCTION statement.

Location



The Location tab used for dynamically loadable C language functions or functions which language is defined as internal. In other cases tab will be invisible.

Function File

Specifies the name of the file containing the dynamically loadable object.

Function Name (link symbol)

Specifies the function link symbol, i.e. the name of the function in the C language source code. If the link symbol is omitted, it is assumed to be the same as the name of the SQL function being defined.

Script

This tab allows you to set SQL statements, which will be executed before (use **Begin** tab) and after (use **End** tab) generation of the stored routine.

Preview

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using previous tabs. Note, that the text within the editor is read-only.

Notes

The **Note** tab allows you to define a description and an annotation for the edited stored routine. This properties will not affect the physical database, but they can be useful for your diagram development.

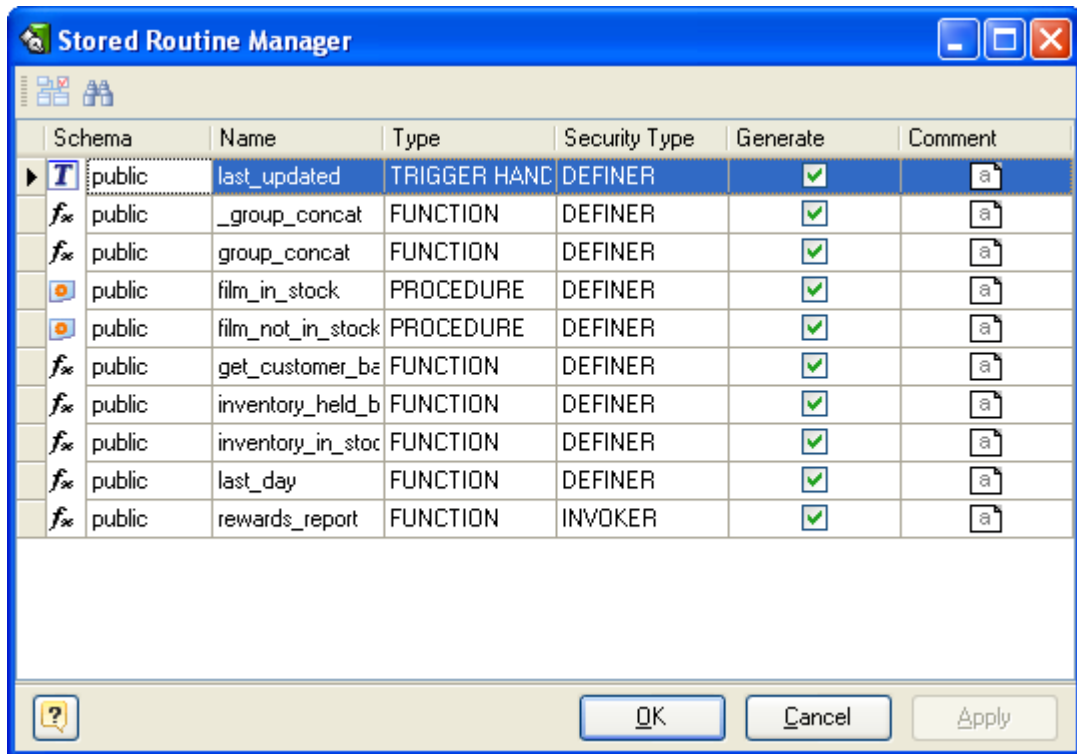
Format

These options allow you to set the routine line and fill color for displaying on the diagram, different from the default colors.

7.3.3. Stored Routine Manager

Stored Routine Manager allows you to view and modify the basic parameters of all stored procedures and functions within the diagram.

To open **Stored Routine Manager** use the **Diagram | Stored Routine Manager** menu item.



The main part of the dialog window is the stored routines grid. The grid rows stand for the stored procedures and functions, and the grid columns stand for their parameters.

These parameters can be changed by using [Stored Routine Editor](#) for each routine one by one, but with the manager you can do it much more quickly.

The grid allows you to modify the following properties:

- **Name** - the name of the routine;
- **Security type** - this option can be used to specify whether the routine should be executed using the permissions of the user who creates the routine or the user who invokes it. The default value is DEFINER. The creator or invoker must have permission to access the database with which the routine is associated;
- **Generate** - includes the routine to the default selection in the [Database Generation](#) and [Database Modification](#) tools;
- **Comment** - an arbitrary description of the routine.

See also:

[Stored Routine Editor](#) | [Creating a Stored Procedure or Function](#)

7.4. Views

Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table).

You can think of a view as virtual table. It does not physically exist, but works like a real table. Usually it is created by a query joining one or more tables.

With **Database Designer for PostgreSQL** you can:

- [Create](#) it;
- [Edit](#) a view;
- [Manage](#) all view within the diagram.

See also:

Diagram Objects: [Creating a View](#) | [View Editor](#) | [View Manager](#)

7.4.1. Creating a View

To create a new view please do the following:

1. Click on the **Create View** () icon on the **Palette** toolbar. Your mouse cursor will change its appearance. Click on the diagram area to create a new view. Also you may right-click on some schema in the **Tree View** object and choose **Create View** item. A rectangle with the view name will appear on the diagram:



2. (optional) Double click on the new view symbol in the diagram to call the [View Editor](#). You can set the name of the view and define its SQL query.

See also:

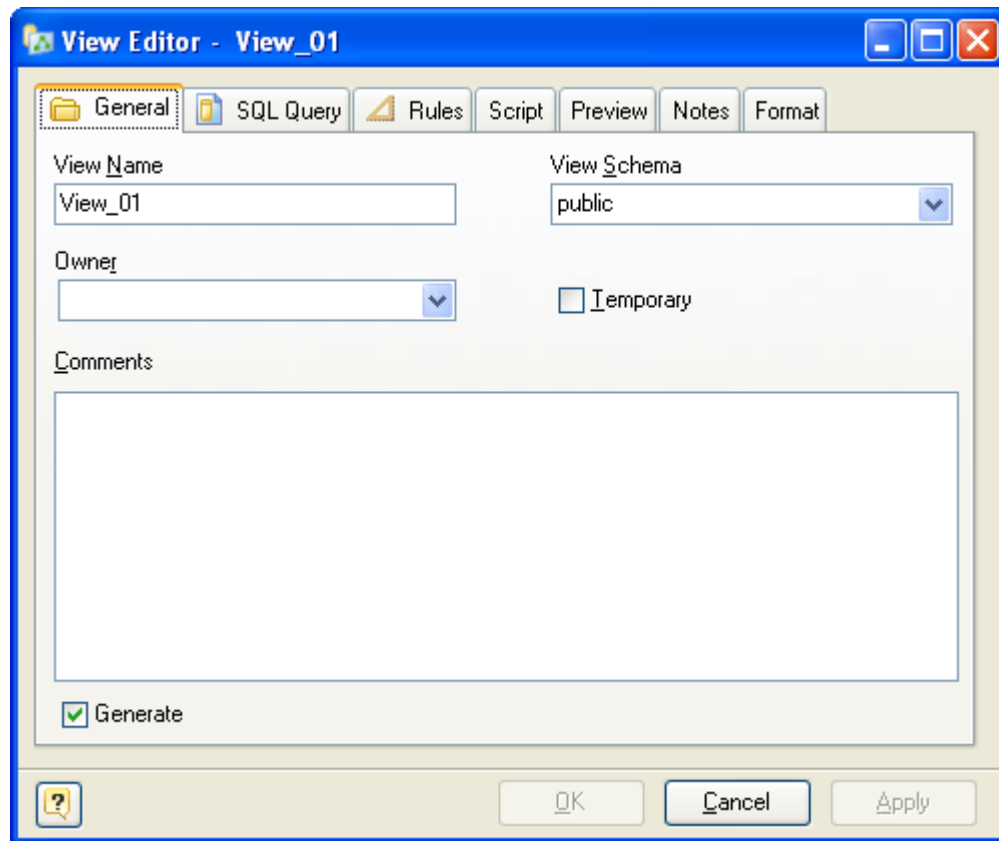
[View Editor](#)

7.4.2. View Editor

The **View Editor** is provided for altering SQL views. To open the **View Editor**, simply double-click a view on the diagram or select the **Properties** item from the context menu.

The **View Editor** contains several tabs, each of which will be described below.

General



This tab allows you to set the name of the view and write comments for it. There are the following fields on the tab:

View Name

Sets the name of the view. To check your diagram for the uniqueness of names use the [Check Diagram tool](#).

View Schema

The name of the schema where the view will be placed.

Owner

A role which will be owner of the object, or a role which will execute CREATE script in case of empty input field.

Temporary

If specified, the view is created as a temporary view. Temporary views are automatically dropped at the end of the current session. Existing permanent relations with the same name are not visible

to the current session while the temporary view exists, unless they are referenced with schema-qualified names.



If any of the tables referenced by the view are temporary, the view is created as a temporary view (whether TEMPORARY is specified or not).

Security Barrier

This option should be enabled when a view is intended to provide row-level security.

Comments

A native comment for the view.

Generate

Set this option off to exclude the view from the default selection in the [Database Generation](#) and [Database Modification](#) tools.

SQL Query

This tab allows to set the SELECT statement that provides the definition of the view. The optional column list can be given to define explicit names for the view columns.

A view can be created from many kinds of SELECT statements. For example, the SELECT can refer to a single table, a join of multiple tables, or a UNION. The SELECT need not even refer to any tables.

An example of SELECT statement, which could be used here:

```
SELECT product_id, product_name FROM Products
```

Rules

Rules Editor allows you to modify view rules. See [Rules Editor](#) for tables description for details.

Script

This tab allows you to set SQL statements, which will be executed before (use **Begin** tab) and after (use **End** tab) generation of the view.

Preview

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using previous tabs. Note, that the text within the editor is read-only. The content of this tab updates only then your press **Apply** button.

Notes

The **Note** tab allows you to define a description and an annotation for the edited view. This properties will not affect the physical database, but they can be useful for your diagram development.

Format

These options allow you to set the view symbol line and fill color for displaying on the diagram, different from the default colors.

See also:

[Creating a View](#)

7.4.3. View Manager

The **View Manager** allows you to view and modify the basic parameters of all SQL views within the diagram. To open the **View Manager** use the **Diagram | View Manager** menu item. The main part of the dialog is the view grid. The grid rows stand for the views, and the grid columns for their parameters.

These parameters can be changed by using [View Manager](#) for each view one by one, but with the manager you can do it much more quickly.

The grid allows you to modify the following properties:

Name

The name of the view;

Generate

Includes the view to the default selection in the [Database Generation](#) and [Database Modification](#) tools;

Comment

An arbitrary description for the view;

SQL Query

SELECT statement that provides the definition of the view.

See also:

[View Editor](#)

7.5. Schemas

A database in PostgreSQL contains one or more schemas, each of them has a certain name. The schemas contain tables and other kinds of objects, including data types, functions and operators. The same object name can be used in different database schemas and no conflict will arise; for example, both **schema1** and **myschema** may contain tables named *mytable*. Unlike databases, schemas are not rigidly separated: a user may access the necessary objects in any of the schemas in the database he is connected to, if he has the privileges to do so.

Database schemas can be compared to directories at the operating system level, the only difference is that schemas cannot be nested.

See also:

[Schema Manager](#)

7.5.1. Schema Manager

Schema Manager allows you to add, edit and delete schemas within the diagram. To open **Schema Manager**, select the **Diagram | Schema Manager** menu item.

Schema is a virtual object of a diagram and has no graphical representation. After defining a schema, it appears in the object editors, so that you could attach other diagram objects to it.

Schemas tab

Schema Manager contains a grid that represents schemas available in the diagram and their properties. The schema properties you can change in the grid are as follows:

Schema name

The name of a schema to be created. The name cannot begin with *pg_*, as such names are reserved for system schemas.

Comment

Comment to the schema.

Owner


The name of the user who will own the schema. If omitted, defaults to the user executing the command. Only superusers may create schemas owned by users other than themselves.

Generate

Set this option off to disable generation of the schema during database generation.

System

This flag shows if the schema is system and can't be changed by a user.

 Despite the fact that *public* schema marked as *system* there is still opportunity to change it's *Comment*, *Owner* and *Generate* options.

The buttons below the list of schemas allow you to perform the following actions:

- **Add** - add a new schema with the default properties to the end of the list;
- **Duplicate** - add a new schema with the same properties as the selected schema to the end of the list;
- **Delete** - remove the selected schema from the list.

Preview tab

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please note that the text within the editor is read-only. The content of this tab updates only when you press the **Apply** button.

7.6. Domains & User defined Types

A **domain** is a storage for some column attributes. Using domains you can create a column with all the required properties already defined (e.g. data type, length, precision, etc.), and modify a number of columns at once.

To create a domain and set all the domain properties use the [Domain & User Defined Type Manager](#). This tool allows you to add, modify, and remove domains and UDT's.

After you have created diagram domains, it becomes much easier to add columns to the diagram tables - you can simply select the appropriate domain for the column within the [Column Editor](#), and this assigns all domain attributes to the corresponding column attributes. It is also much more convenient to modify the domain-based columns than usual columns. Modifying any of the domain attribute updates all the columns, based on this domain.

The **composite type** is specified by a list of attribute names and data types. This is essentially the same as the row type of a table, but using UDT Manager avoids the need to create an actual table when all that is wanted is to define a type. A stand-alone composite type is useful as the argument or return type of a function.

The **Domain & User Defined Type Manager** allows creation of a new base type (scalar type). You must register two or more functions (using [Stored Procedure Editor](#)) after defining the type. A user-defined type must always have input and output functions. These functions determine how the type appears in strings (for input by the user and output to the user) and how the type is organized in memory. The input function takes a null-terminated character string as its argument and returns the internal (in memory) representation of the type. The output function takes the internal representation of the type as argument and returns a null-terminated character string. If we want to do anything

more with the type than merely store it, we must provide additional functions to implement whatever operations we'd like to have for the type.

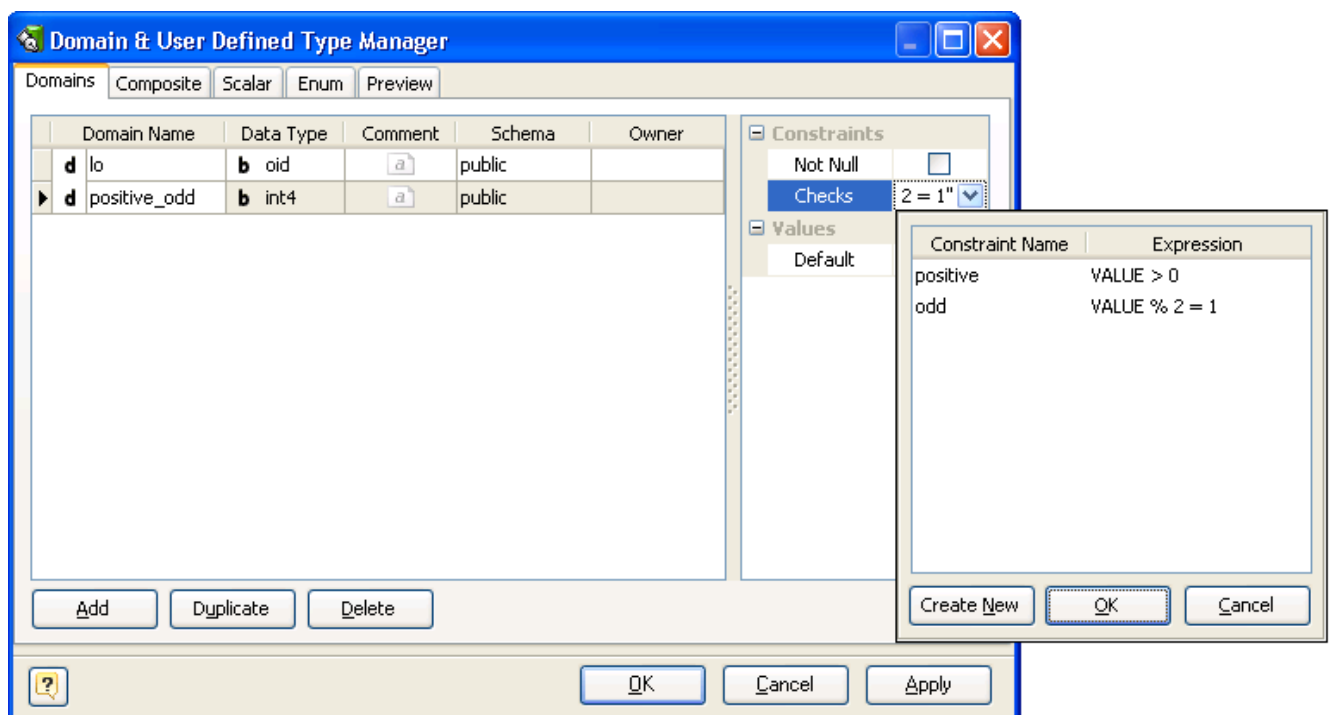
See also:

Diagram Objects: [Column Editor](#) | [Domain & User Defined Type Manager](#) | [Stored Routine Editor](#)

7.6.1. Domain & User Defined Type Manager

The **Domain & User Defined Type Manager** is intended for managing diagram domains, composite and scalar user defined types, which can be used for faster creating and modifying table columns, stored procedures etc.

To open the **Domain & User Defined Type Manager**, select the **Diagram | Domain & User Defined Type Manager** menu item.



The **Domain** page consists of the following areas:

Domain List

The **Domains** list displays all the domains in the diagram and allows you to modify the following domain properties:

- **Domain name** - the name of the domain, which must be unique within the schema;

- **Data type** - the data type of the domain;
- **Comment** - an arbitrary description for the domain.
- **Schema** - the database schema that domain belongs to.

Properties Pane

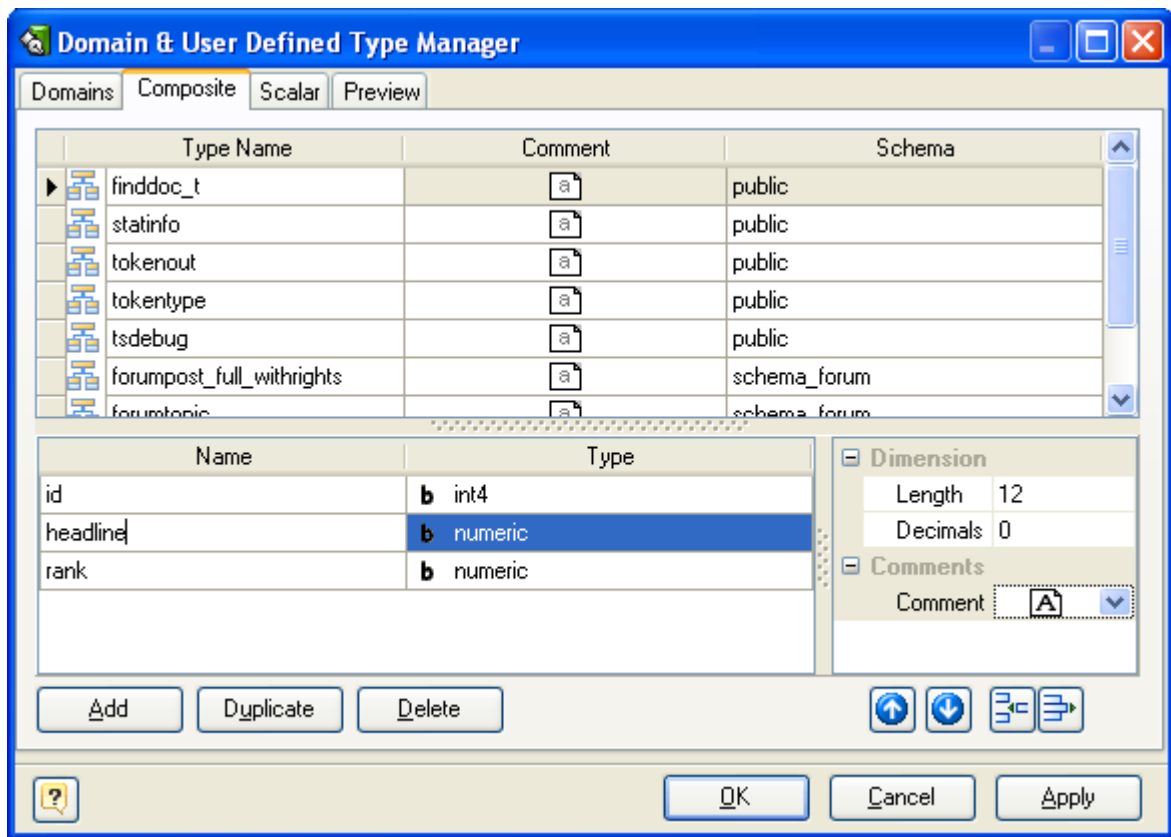
The properties pane allows you to define the advanced properties of the domain, selected in the Domain List. The appearance of this pane changes according to the data type of the domain. These properties are:

- **Not null** - this option indicates that a domain-based column value cannot be NULL;
- **Default** - this attribute defines the default value, which the domain-column accepts if no other is specified;
- **Checks** - this property specifies integrity constraints or tests which values of the domain must satisfy. Each constraint must be an expression producing a Boolean result. It should use the name VALUE to refer to the value being tested.

Buttons Pane

The buttons under the list of domains allows you to perform the following actions:

- **Add** - add a new domain with the default properties to the end of the list;
- **Duplicate** - add a new domain with the same properties as the selected domain to the end of the list;
- **Delete** - remove the selected domain from the list.



The **Composite** page consists of the following areas:

Composite List

The **Composite** list displays all the composites in the diagram and allows you to modify the following composite properties:

- **Type name** - the name of the composite type, which must be unique within the schema;
- **Comment** - an arbitrary description for the composite type.
- **Schema** - the database schema that composite type belongs to.

Type Attribute (Column) List

Type Attribute (Column) List displays all the columns in the selected composite type and allows you to modify the following properties:

- **Attribute name** - the name of the composite type attribute, which must be unique within the parent type;

- **Type** - the data type of the attribute.

Properties Pane

The properties pane allows you to define the advanced properties of the attribute, selected in the Type Attribute List. The appearance of this pane changes according to the data type of the attribute. These properties are:

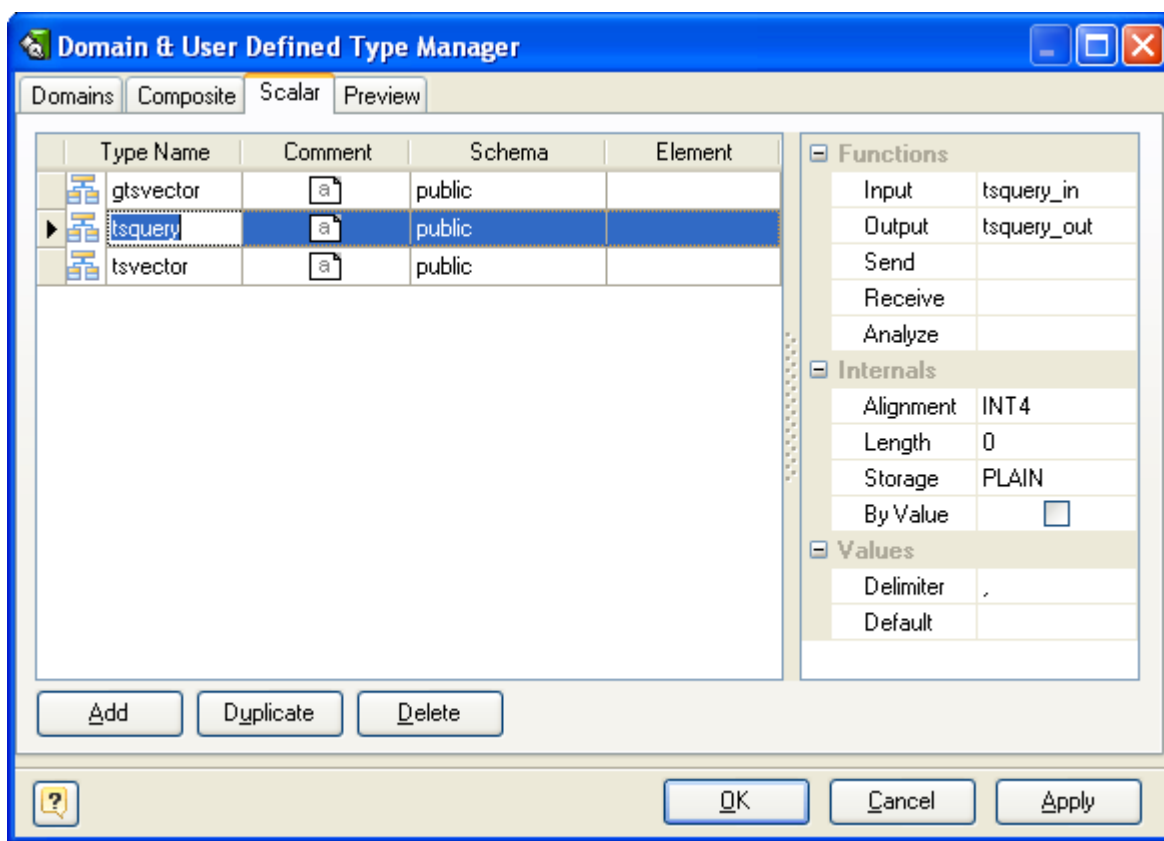
- **Length** - this option indicates maximum allowed length for a column;
- **Decimals** - this attribute defines the number of decimal digits in the fractional part of a numeric attribute;
- **Comment** - an arbitrary description for the composite type attribute.

Buttons Pane

The Buttons Pane is similar to Buttons Pane on the Domain Page, but have additional buttons:

The buttons under the list of columns allow you to perform the following actions:

- **Up/Down** - move the selected column along the list;
- **Add** - add a new column with the default properties to the end of the list;
- **Delete** - remove the selected column from the list.



The **Scalar** page consists of the following areas:

Scalar List

The scalar list displays all the scalars in the diagram and allows you to modify the following scalar properties:

- **Scalar name** - the name of the scalar, which must be unique within the schema;
- **Comment** - an arbitrary description for the scalar;
- **Schema** - the database schema that scalar belongs to;
- **Element** - specifies that scalar type being created is an array; this specifies the type of the array elements.

Properties Pane

The properties pane allows you to define the advanced properties of the scalar, selected in the Scalar List. These properties are:

- **Input** - the name of a function that converts data from the type's external textual form to its internal form;
- **Output** - the name of a function that converts data from the type's internal form to its external textual form;
- **Receive** - the name of a function that converts data from the type's external binary form to its internal form;
- **Send** - the name of a function that converts data from the type's internal form to its external binary form;
- **Analyze** - the name of a function that performs statistical analysis for the data type;
- **Alignment** - the storage alignment requirement of the data type. It must be CHAR, INT2, INT4, or DOUBLE; the default is INT4;
- **Length** - a numeric constant that specifies the length in bytes of the new type's internal representation. The default assumption is that it is variable-length;
- **Storage** - the storage strategy for the data type. If specified, must be PLAIN, EXTERNAL, EXTENDED, or MAIN; the default is PLAIN;
- **By Value** - indicates that values of this data type are passed by value, rather than by reference;
- **Delimiter** - the delimiter character to be used between values in arrays made of this type;
- **Default** - the default value for the data type. If this is omitted, the default is null.

Buttons Pane

The buttons under the list of scalars allows you to perform the following actions:

- **Add** - add a new scalar with the default properties to the end of the list;
- **Duplicate** - add a new scalar with the same properties as the selected scalar to the end of the list;
- **Delete** - remove the selected scalar from the list.

See also:

Diagram Objects: [Domains & User Defined Types](#) | [Columns](#) | [Stored Routine Editor](#)

7.7. References and Foreign Keys

Foreign key constraints are responsible for data referential integrity in your database. Simply put, referential integrity means that when a record in a table refers to a corresponding record in another table, that corresponding record must exist. So, a foreign key constraint specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table.


A reference sets a foreign key constraint on the referencing table.


See also:

Diagram Objects: [Creating a Reference](#) | [Reference Editor](#)

7.7.1. Creating a Reference

To create a reference between tables:

1. Click the  icon on the **Palette** toolbar. Your mouse cursor will change its appearance.
2. Click on the table (a referencing table) that will have a foreign key.
3. Then click on the second table (referenced table) whose constraint (e.g. Primary key) will be referenced by the new foreign key.
4. The **Join** tab of the [Reference Editor](#) will be shown. You can choose columns of a referenced and referencing table participated in the reference. You can choose *<Auto Create Column>* on the list of referencing table columns to **Database Designer for PostgreSQL** create a new column in the referencing table that will have a foreign key set on it. The properties of auto column will be copied from the respective column of the referenced table.

To create a self reference (that links columns at the same table), click the  icon on the **Palette** toolbar. Then click the same table two times. To create **N:M** references, please refer to [Creating a Many-to-Many Reference](#).

Reference Creation in Details

On reference creation, **Database Designer for PostgreSQL** performs the following actions:

1. Creates new column(s) in the referencing table, their parameters (name, data type) will be copied from the primary key constraint of the referenced table. If the referencing table already has column(s) analogous to the primary key(s) of the referenced table, this column(s) will be used as foreign-key column(s).
2. If there is no primary key or unique constraints in the referenced table, a standard primary key column will be created in the referenced table.
3. Creates foreign key constraint in the referencing table that refers to the referenced table primary key(s).

See the diagram [database options](#) to find out more about disabling some of above actions.

See the [Notation](#) topic to find out more about reference symbol on the diagram.

See also:

Diagram: [Notation](#) | [Database Options](#) | [Creating a Many-to-Many Reference](#)

Diagram Objects: [Reference Editor](#)

7.7.2. Creating a Many-to-Many Reference

A *many-to-many* reference can relate one record in either table to many records in the other table.


The only way to create many-to-many (**N:M**) reference between two tables in PostgreSQL is using an intermediate table, which records set association between primary keys of first table and primary keys of second table.

Database Designer for PostgreSQL helps you set many-to-many reference between two tables easily:

- it automatically creates intermediate table with proper columns and indexes;
- and then creates regular references between columns of intermediate table and primary keys of tables being linked.

Remember, that tables being linked must have primary key.

How to create...

To create a *many-to-many* reference between two tables, click the  icon on the **Palette** toolbar. Your mouse cursor will change its appearance. Then consistently click two tables you want to link with many-to-many reference.

See also:

[Creating a Reference](#)

7.7.3. Reference Editor

Reference Editor is intended for editing the properties of a foreign key constraint, integrity rules, and choosing referenced columns.

To open the **Reference Editor**, simply double-click a reference on the diagram or select the **Properties** item from the reference context menu.

The dialog consists of several tabs, each of which is described in details below.

General

This tab allows you to set up the basic reference properties.

Name

You can enter reference name, which will be displayed in the caption in the middle of the reference on the diagram. It is desirable, than name describe relationship role between tables.

Comment

The field to describe the reference role in full.

Parent table

Shows the referenced table's name.

Child table

Shows the referencing table's name.

Generate

Set this option off to exclude the table from the default selection of references in the [Database Generation](#) and [Database Modification](#) tools.

Joins

This tab allows to choose an active constraint of the referenced table and columns participated in the reference.

Parent Constraints

This allows to select one of the primary key or unique constraints defined on the referenced column. Choosing a constraint will define columns of referenced table that will be used in the reference. The columns on which the selected constraint is defined fill the grid placed on the left side of the dialog.

In the left side of the grid there is list of columns of referenced table. In the right side there is a list of corresponding foreign key columns of the *referencing table*.

To change assignment of foreign key column of the referencing table:

1. Click on the Child table column cell, that corresponds to one of the columns of the *referenced table*.
2. The drop-down menu with the list of the *referencing table* columns will appear.
3. Then click on the required column of the *referencing table*.

Integrity

In this tab you can change integrity rules for foreign key constraint.

When record in referenced table is changed (updated or deleted), it is possible to automatically modify associated records in the referencing table. This is the function of delete/update rules of the constraint.

For example, there are two tables: *Orders* and *Customers*. The *Orders* table has the foreign key column *CustID* which refer to the *Customers* table. You can delete a record from the *Customers* table and corresponding records from the *Orders* table, using one DELETE statement. It is possible because of cascade delete rule.

You can assign a rule both to update and delete event. In the left side there is list of update rules. In the right side there is list of delete rules. Click on the appropriate list item to change a rule.

Update constraint. Sets a rule that will be executed on update of a referenced table's record

Restrict. Disallow update of the referenced table record if associated records in the child table exist.

Cascade. Update associated records in compliance with referenced table row update.

Set null. Set foreign key columns of associated records to null.

No action. Do nothing with associated records.

Set default. Set foreign key columns of associated records to default column value. This default value can be set in the [Table Columns Manager](#).

Delete constraint. Sets a rule that will be executed on delete of a referenced table's record

Restrict. Disallow delete of referenced table record if associated records in the child table are exists.

Cascade. Delete all associated records.

Set null. Set foreign key columns of associated records to NULL.

No action. Do nothing with associated records.

Set default. Set foreign key columns of associated records to default column value. This default value can be set in the [Table Columns Manager](#).

A value inserted into the referencing column(s) is matched against the values of the referenced table and referenced columns using the given match type. There are three match types: MATCH FULL, MATCH PARTIAL, and MATCH SIMPLE, which is also the default.

Match Full will not allow one column of a multicolumn foreign key to be null unless all foreign key columns are null.

Match Simple allows some foreign key columns to be null while other parts of the foreign key are not null.

Match Partial will be implemented in the future versions of PostgreSQL.

You can also set transaction check mode.

Not deferrable. This controls whether the constraint reference can be deferred. A constraint that is not deferrable will be checked immediately after every command. Checking of constraints that are deferrable may be postponed until the end of the transaction (using the SET CONSTRAINTS command). NOT DEFERRABLE is the default. Only foreign key constraints currently accept this clause. All other constraint types are not deferrable.

Deferrable initially immediate. In this case constraint will be checked after each statement. The constraint check time can be altered with the SET CONSTRAINTS command inside the transaction block.

Deferrable initially deferred. In this case constraint will be checked only at the end of the transaction. The constraint check time can be altered with the SET CONSTRAINTS command inside the transaction block.

Note

The **Note** tab allows you to define a description and an annotation for the edited reference. This properties will not affect the physical database, but they can be useful for your diagram development.

Format

In this tab you can choose the color which will be used for displaying edited reference on the diagram. Click on the field to select appropriate color.

See also:

Diagram Objects: [Column Editor](#) | [Creating a Reference](#)

7.7.4. Reference Manager

Reference Manager allows you to view and modify basic parameters of table references within the diagram.

To open the **Reference Manager** use the **Diagram | Reference Manager** menu item or press **Ctrl-5**.

The dialog contains a grid with the list of references that exist within the diagram. The grid shows reference names and their properties.

The grid allows you to modify the following properties:

Reference - the name of a reference;

On update - sets a rule that will be executed on update of a referenced table's record;

On delete - sets a rule that will be executed on delete of a referenced table's record;

Generate - Includes the table to the default selection of references in the [Database Generation](#) and [Database Modification](#) tools;

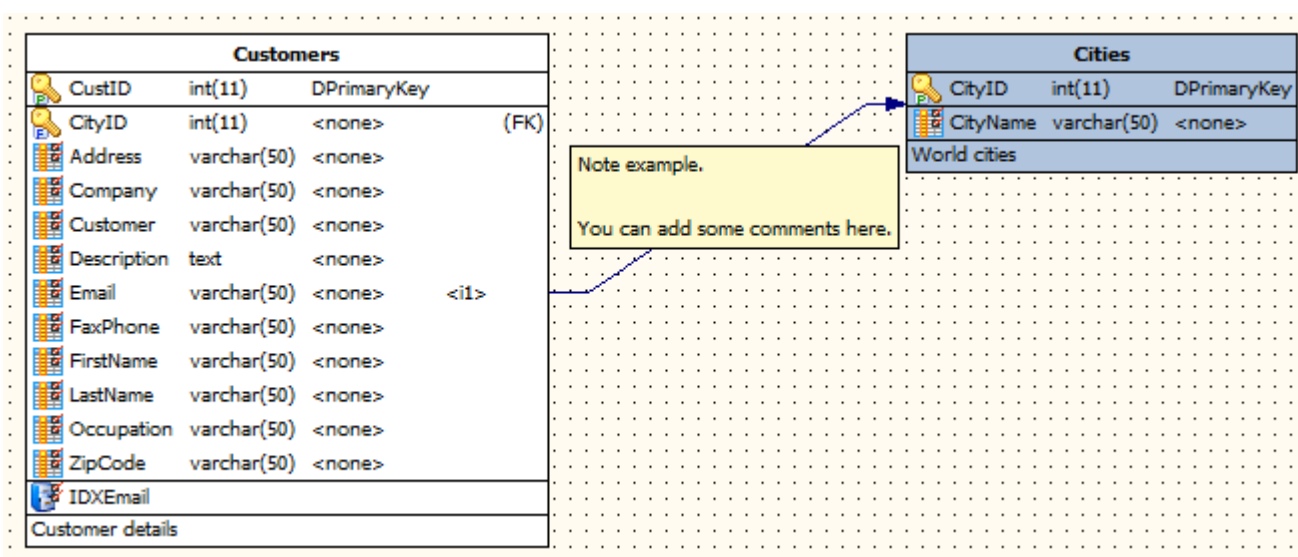
The **Parent Table** and **Child Table** columns of the grid display relationships between columns of participated tables.

See also:

[Reference Editor](#)

7.7.5. Manual Reference Drawing

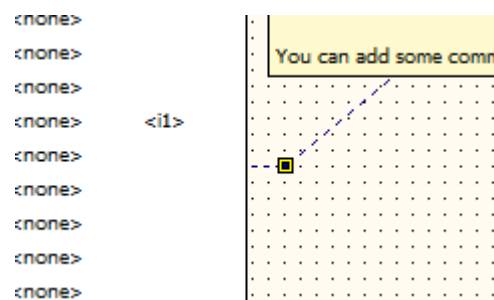
By default **Database Designer for PostgreSQL** routes references automatically using shortest line between two table shapes. Sometimes this is inconvenient because some other database objects can appear above reference or its label.



However you can draw references manually, change label position, start and end tails position. You can add additional intermediate points to route your references in some other way.


Manual Start / End Tail Position

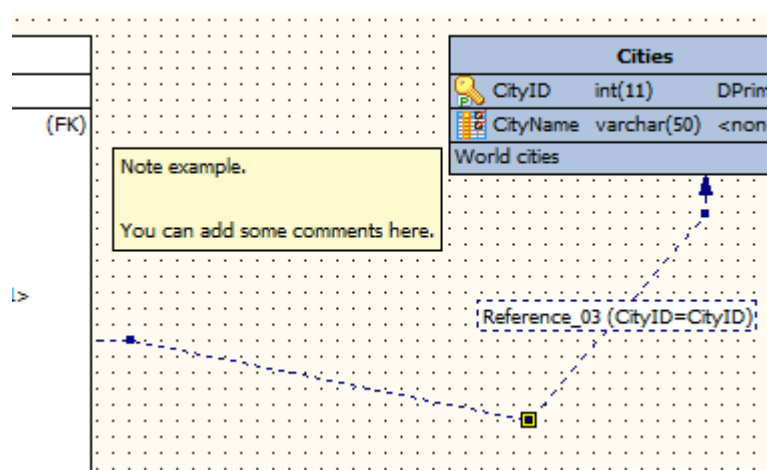
To allow changing of tails position use **Manual Start Tail Position** and **Manual End Tail Position** items from reference context menu (right mouse button click on reference). After enabling **Manual Start/End Tail Position** item small marker will appear at reference tail. You can click this marker by left mouse button and drag it to preferred location. Marker will be marked with black-yellow frame while moving.



Click **Manual Start/End Tail Position** item of context menu again to disable manual tail positioning and let **Database Designer for PostgreSQL** calculate its position.

User Points

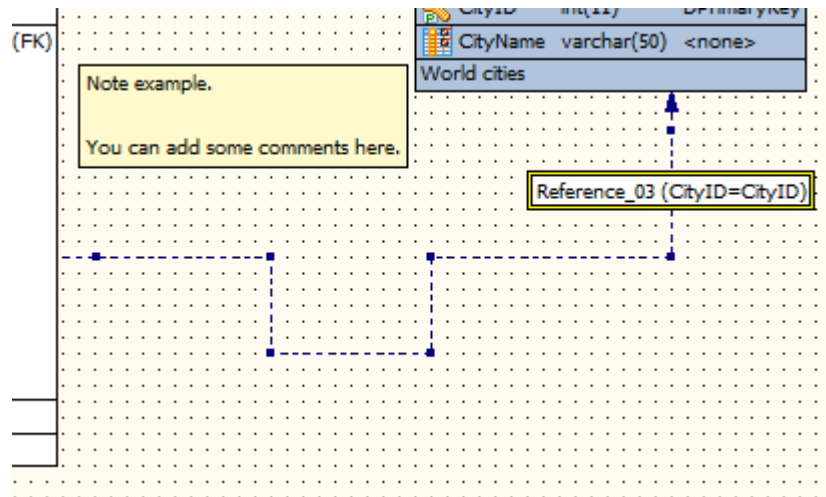
Use **Add Point** () item of reference context-menu to add user point to reference. This points are displayed with small square markers when reference is selected. You can use left mouse button to drag such user point to any location you prefer. You can add such many user points as you need.



You can delete user points by using **Delete Point** () item of reference context-menu.

Reference Label Repositioning

Another new feature added is an ability to move reference's label. Just click it by left mouse button and drag to any other location you prefer on reference. Label is marked with black-yellow frame while moving.



See also:

Diagram Objects: [References and Foreign Keys](#) | [Creating a Reference](#)

7.8. Roles

Database roles are conceptually completely separate from operating system users. In practice it might be convenient to maintain a correspondence, but this is not required. Database roles are global across a database cluster installation (and not per individual database).

Every connection to the database server is made in the name of some particular role, and this role determines the initial access privileges for commands issued on that connection. The role name to use for a particular database connection is indicated by the client that is initiating the connection request in an application-specific fashion.

Since the role identity determines the set of privileges available to a connected client, it is important to carefully configure this when setting up a multiuser environment.

To create a Role and set all the Role properties use the [Role Manager](#). This tool allows you to add, modify, and remove Roles.

After you have created database Roles, you may specify owners for any diagram object using appropriate object editor.

See also:

[Role Manager](#)

7.8.1. Roles Manager

Role Manager allows you to add, edit and delete Roles within the diagram. To open **Role Manager**, select the **Diagram | Role Manager** menu item.

Role is a virtual object of a diagram and has no graphical representation. After defining a **Role**, it appears in the object editors, so that you could attach other diagram objects to it.

Roles tab

Role Manager contains a grid that represents **Roles** available in the diagram and their properties. The Role properties you can change in the grid are as follows:

Role name


The name of a **Role** to be created. The name cannot begin with *pg_*, as such names are reserved for system **Roles**.

Login

Only roles that have the LOGIN attribute can be used as the initial role name for a database connection. A role with the LOGIN attribute can be considered the same thing as a "database user".


Inherit

These option determine whether a role "inherits" the privileges of roles it is a member of. A role with the INHERIT attribute can automatically use whatever database privileges have been granted to all roles it is directly or indirectly a member of. Without INHERIT, membership in another role only grants the ability to SET ROLE to that other role; the privileges of the other role are only available after having done so.

 The INHERIT attribute governs inheritance of grantable privileges (that is, access privileges for database objects and role memberships). It does not apply to the special role attributes set by CREATE ROLE and ALTER ROLE. For example, being a member of a role with CREATEDB privilege does not immediately grant the ability to create databases, even if INHERIT is set.

Connection Limit

If role can log in, this specifies how many concurrent connections the role can make. -1 (the default) means no limit.

 The CONNECTION LIMIT option is only enforced approximately; if two new sessions start at about the same time when just one connection "slot" remains for the role, it is possible that both will fail. Also, the limit is never enforced for superusers.

SuperUser


A database superuser bypasses all permission checks. This is a dangerous privilege and should not be used carelessly; it is best to do most of your work as a role that is not a superuser. You must do this as a role that is already a superuser.

Create DB

A role must be explicitly given permission to create databases (except for superusers, since those bypass all permission checks).


Create Role

A role must be explicitly given permission to create more roles (except for superusers, since those bypass all permission checks). A role with `CREATEROLE` privilege can alter and drop other roles, too, as well as grant or revoke membership in them. However, to create, alter, drop, or change membership of a superuser role, superuser status is required; `CREATEROLE` is not sufficient for that.

 Be careful with the `CREATEROLE` privilege. There is no concept of inheritance for the privileges of a `CREATEROLE`-role. That means that even if a role does not have a certain privilege but is allowed to create other roles, it can easily create another role with different privileges than its own (except for creating roles with superuser privileges). For example, if the role “user” has the `CREATEROLE` privilege but not the `CREATEDB` privilege, nonetheless it can create a new role with the `CREATEDB` privilege. Therefore, regard roles that have the `CREATEROLE` privilege as almost-superuser-roles.


Password

A password is only significant if the client authentication method requires the user to supply a password when connecting to the database. The *password*, *md5*, and *crypt* authentication methods make use of passwords. Database passwords are separate from operating system passwords.

 It is good practice to create a role that has the `CREATEDB` and `CREATEROLE` privileges, but is not a superuser, and then use this role for all routine management of databases and roles. This approach avoids the dangers of operating as a superuser for tasks that do not really require it.

Valid

The `Valid` option sets a date and time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.

 This option defines an expiration time for a password only, not for the role per se. In particular, the expiration time is not enforced when logging in using a non-password-based authentication method.

In Role

Option lists one or more existing roles to which the new role will be immediately added as a new member.

Roles

Option lists one or more existing roles which are automatically added as members of the new role.

Admins

This option is like Roles, but the named roles are added to the new role WITH ADMIN OPTION, giving them the right to grant membership in this role to others.

Generate

Set this option off to disable generation of the Role during database generation.

The buttons below the list of Roles allow you to perform the following actions:

Add - add a new Role with the default properties to the end of the list;

Duplicate - add a new Role with the same properties as the selected Role to the end of the list;

Delete - remove the selected Role from the list.

Preview tab

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please note that the text within the editor is read-only. The content of this tab updates only when you press the **Apply** button.

7.9. Tablespaces

Tablespaces in PostgreSQL allow database administrators to define locations in the file system where the files representing database objects can be stored. Once created, a tablespace can be referred to by name when creating database objects.

By using tablespaces, an administrator can control the disk layout of a PostgreSQL installation. This is useful in at least two ways:

First, if the partition or volume on which the cluster was initialized runs out of space and cannot be extended, a tablespace can be created on a different partition and used until the system can be reconfigured.

Second, tablespaces allow an administrator to use knowledge of the usage pattern of database objects to optimize performance. For example, an index which is very heavily used can be placed on a very fast, highly available disk, such as an expensive solid state device. At the same time a table storing archived data which is rarely used or not performance critical could be stored on a less expensive, slower disk system.

To create a tablespace and set all the tablespace properties use the [Tablespace Manager](#). This tool allows you to add, modify, and remove tablespaces.

After you have created diagram tablespaces, you may specify tablespaces for tables and indexes using [Table Editor](#).

See also:

Diagram Objects: [Table Editor](#) | [Tablespace Manager](#)

7.9.1. Tablespaces Manager

Tablespace Manager allows you to add, edit and delete Tablespaces within the diagram. To open **Tablespace Manager**, select the **Diagram | Tablespace Manager** menu item.

Tablespace is a virtual object of a diagram and has no graphical representation. After defining a **Tablespace**, it appears in the object editors, so that you could attach other diagram objects to it.

Tablespaces tab

Tablespace Manager contains a grid that represents Tablespaces available in the diagram and their properties. The Tablespace properties you can change in the grid are as follows:

Tablespace name

The name of a Tablespace to be created. The name cannot begin with pg_, as such names are reserved for system Tablespaces.

Tablespace path

The directory that will be used for the tablespace. The directory must be empty and must be owned by the PostgreSQL system user. The directory must be specified by an absolute path name.

Owner

The name of the user who will own the tablespace. If omitted, defaults to the user executing the command. Only superusers can create tablespaces, but they can assign ownership of tablespaces to non-superusers.

Generate

Set this option off to disable generation of the Tablespace during database generation.

The buttons below the list of Tablespaces allow you to perform the following actions:

Add - add a new Tablespace with the default properties to the end of the list;

Duplicate - add a new Tablespace with the same properties as the selected Tablespace to the end of the list;

Delete - remove the selected Tablespace from the list.

Preview tab


The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please

note that the text within the editor is read-only. The content of this tab updates only when you press the **Apply** button.

7.10. Sequences

Sequence objects (also called sequence generators or just sequences) are special single-row tables created with CREATE SEQUENCE. A sequence object is usually used to generate unique identifiers for rows of a table.

The sequence name must be distinct from the name of any other sequence, table, index, or view in the same schema.

 **Database Designer for PostgreSQL** has a simplified interface for creating auto-increment columns. Instead of creating sequences, you can simply set the Autoinc property for the appropriate column by using the [Column Editor](#).

7.10.1. Sequences Manager

Sequence Manager allows you to add, edit and delete Sequences within the diagram. To open **Sequence Manager**, select the **Diagram | Sequence Manager** menu item.

Sequence is a virtual object of a diagram and has no graphical representation. After defining a **Sequence**, it appears in the object editors, so that you could attach other diagram objects to it.

Sequences tab

Sequence Manager contains a grid that represents Sequences available in the diagram and their properties. The Sequence properties you can change in the grid are as follows:

Sequence name

The name of a Sequence to be created. The name cannot begin with *pg_*, as such names are reserved for system Sequences.

Schema

The database schema that sequence belongs to.

Comment

Comment to the Sequence.

Owner

The name of the user who will own the Sequence. If omitted, defaults to the user executing the command. Only superusers may create Sequences owned by users other than themselves.

Increment

Specifies which value is added to the current sequence value to create a new value. A positive value will make an ascending sequence, a negative one a descending sequence. If omitted then the default value is 1.

MinVal

Determines the minimum value a sequence can generate. If this clause is not supplied then defaults will be used. The defaults are 1 and -263-1 for ascending and descending sequences, respectively.

MaxVal

Determines the maximum value for the sequence. If this clause is not supplied then default values will be used. The defaults are 263-1 and -1 for ascending and descending sequences, respectively.

Start

Allows the sequence to begin anywhere. The default starting value is *minvalue* for ascending sequences and *maxvalue* for descending ones.

Cache

Specifies how many sequence numbers are to be preallocated and stored in memory for faster access. The minimum value is 1 (only one value can be generated at a time, i.e., no cache), and this is also the default.

Cycle

Allows the sequence to wrap around when the *maxvalue* or *minvalue* has been reached by an ascending or descending sequence respectively. If the limit is reached, the next number generated will be the *minvalue* or *maxvalue*, respectively. If unchecked, any calls to *nextval* after the sequence has reached its maximum value will return an error.

The buttons below the list of Sequences allow you to perform the following actions:

Add - add a new Sequence with the default properties to the end of the list;

Duplicate - add a new Sequence with the same properties as the selected Sequence to the end of the list;

Delete - remove the selected Sequence from the list.

Preview tab

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please note that the text within the editor is read-only. The content of this tab updates only when you press the **Apply** button.

7.11. Privileges

When an object is created, it is assigned an owner. The owner is normally the role that executed the creation statement. For most kinds of objects, the initial state is that only the owner (or a superuser) can do anything with the object. To allow other roles to use it, privileges must be granted. There are several different kinds of privilege: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE, and USAGE.

To assign, modify and remove model objects' privileges, the [ACL Manager](#) is used.



ACL stands for "Access Control List". We will use this term along with "Privileges" term.

The special name PUBLIC can be used to grant a privilege to every role on the system.

The special privileges of an object's owner (i.e., the right to modify or destroy the object) are always implicit in being the owner, and cannot be granted or revoked. But the owner can choose to revoke his own ordinary privileges, for example to make a table read-only for himself as well as others.

See also:

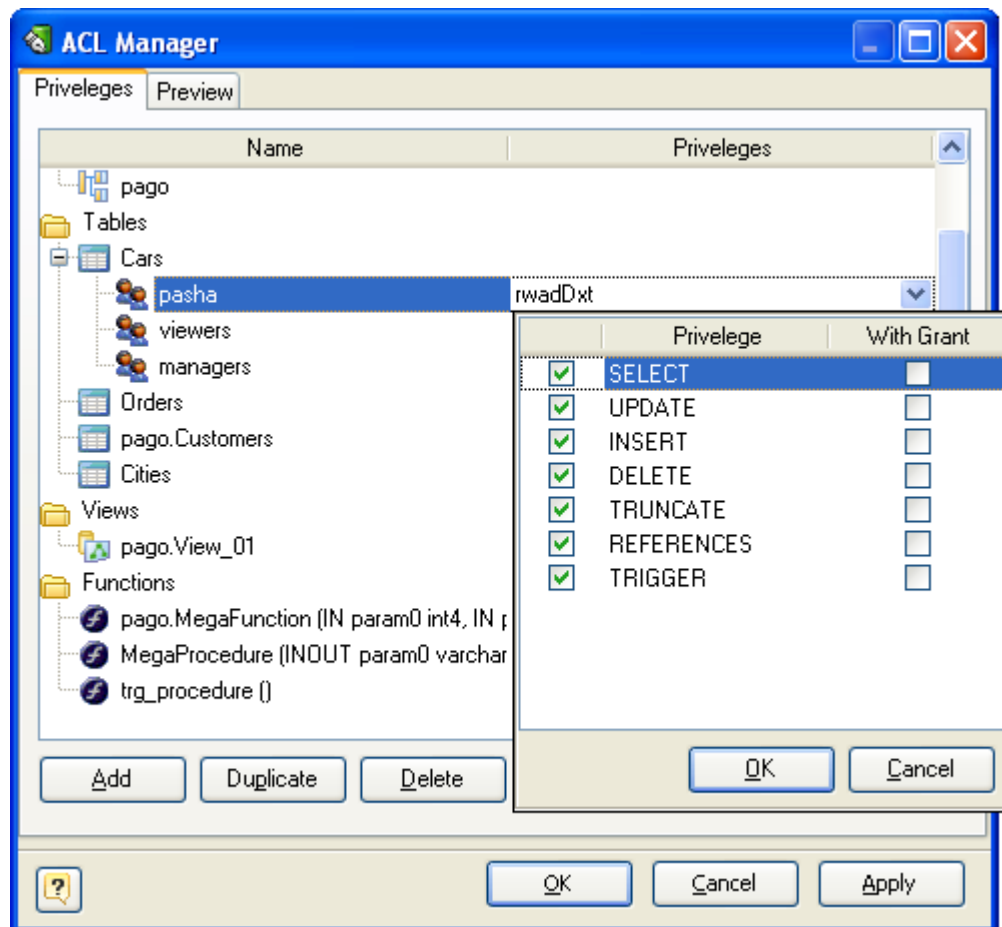
Diagram Objects: [ACL Manager](#)

7.11.1. ACL Manager

ACL Manager allows you to add, edit and delete objects' privileges within the diagram. To open ACL Manager, select the **Diagram | Privileges Manager** menu item.

Privileges tab

ACL Manager contains a grid that represents objects available in the diagram and their access control lists (ACL). The columns in the grid are as follows:



Name

The name of an object and role names privileges for which to be created.

If the role name is empty this stands for PUBLIC - the special name used to grant a privilege to every role on the system.

Privileges

ACL (Access Control List) value in the PostgreSQL representation:

- r** - SELECT ("read")
- w** - UPDATE ("write")
- a** - INSERT ("append")
- d** - DELETE
- D** - TRUNCATE
- x** - REFERENCES
- t** - TRIGGER

X - EXECUTE

U - USAGE

C - CREATE

c - CONNECT

T - TEMPORARY

***** - grant option for preceding privilege.

The buttons below the list allow you to perform the following actions:

Add - add a new ACL entry with the default properties to the selected object;

Duplicate - add a new ACL entry with the same properties as the selected ACL entry;

Delete - remove the selected ACL entry from the list.

Up - move the selected ACL entry up one position.

Down - move the selected ACL entry down one position.



Using **Up** and **Down** buttons you can move ACL entry from one object to other.

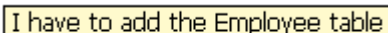
Preview tab

The **Preview** tab displays the SQL statement, which will be executed during the database generation. This statement is made up according to the changes you have made using the previous tabs. Please note that the text within the editor is read-only. The content of this tab updates only when you press the **Apply** button.

7.12. Notes


You can type in the **Note** box any comment concerning your diagram. **Note** is useful for writing various comments, such as object functionality or role description, things to do, etc in your diagram.

The following picture demonstrates a sample Note:



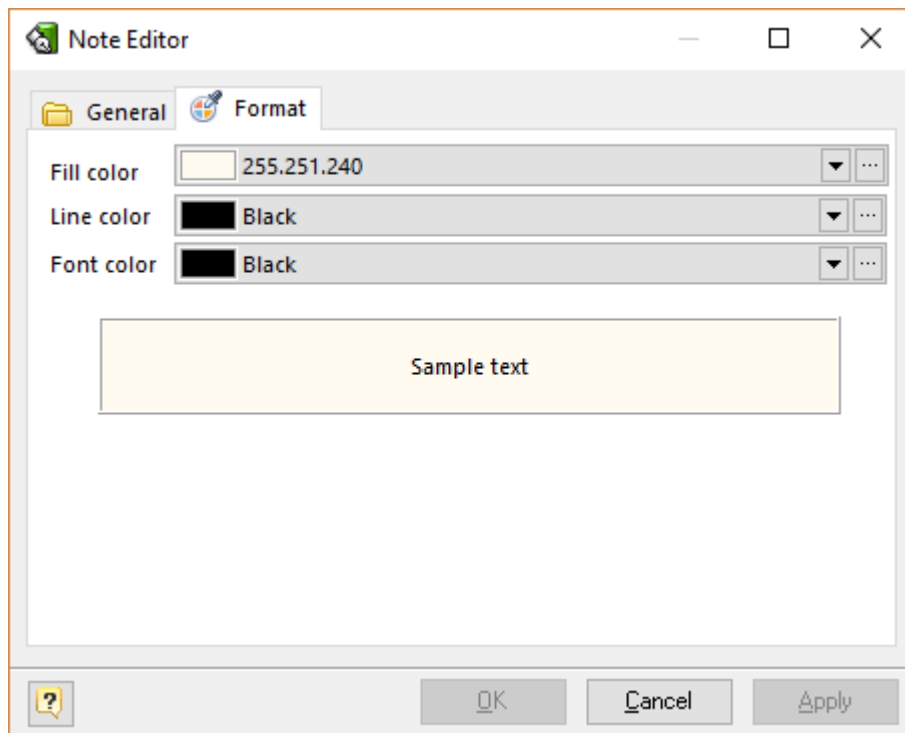
I have to add the Employee table

Placing Note to the diagram

To place a note in the diagram, select the **Note** tool () from the **Palette** toolbar. Then click anywhere in the diagram. Certainly, you can place multiple notes in one diagram.

Modify the text in Note

To modify the information, that is displayed in **Note**, double click on it. Then the following **Note editor** will appear:



In the drop-down menu select the color that you want to use for the background in your note. Enter the text you want to be shown inside the Note object in the diagram.

 Please note, that **Note** is auxiliary object, it doesn't affect database generation.


7.13. Stamps

A **Stamp** box displays essential information about the diagram such as: the name of the diagram itself, the author and version of the diagram and the date of modification. This information is retrieved from the diagram properties.

The following picture demonstrates sample stamp:

Physical Data Diagram
Project: ABC Aircraft
Diagram: Resellers
Company: ABC Inc.
Author: Petya Pupkin
Date: 25.06.2006
Copyright: ABC Inc., 2005-2006
Version: 1.1

Placing Stamp to the diagram

To place **Stamp** on the diagram, select the Stamp tool () from the Palette toolbar. Then click on any place of diagram. Note, that you can place multiple stamps on one diagram.

Modify Stamp information

To modify the information, that is displayed on the stamp, double click on it or select **Diagram | Diagram properties**. [Diagram Properties editor](#) will be shown. Edit the appropriate fields to modify the stamp and diagram information.

 Please note, that **Stamp** is an auxiliary object, and it doesn't affect database generation.

See also:

Diagram: [General Diagram Properties](#)

Diagram Objects: [Notes](#)

8. Diagram Functions

Database Designer for PostgreSQL supports the following diagram functions:

[Merge Diagram](#), that allows to merge content of two diagrams, and

[Check Diagram](#), that allows to check your database diagram for most typical errors and defects.

[Repair Diagram](#), that allows to check and repair underlying physical structure of model file (*.pdd) if some reference integrity was broken.

8.1. Check Diagram

The **Check Diagram** tool of the **Database Designer for PostgreSQL** allows you to check your database diagram for most typical errors and defects. The result of the check goes in a well structured form, using which you can easily bring your diagram to correspondence with the common standards of database modeling.

To open the **Check Diagram** dialog press **F4** or select the **Diagram | Check Diagram** menu item.

Use the **Select diagram** drop-down list to select the diagram from the list of currently opened diagrams.

The tree list below allows you to select what warnings and errors should be taken into account during the check. All warnings and errors are divided into categories, which correspond to the diagram objects. Remove selection from the warning/error or from the whole category to exclude it from the check.

These are the descriptions for all available warnings and errors:

Table

Error "Table Name Uniqueness"

Check the diagram for the uniqueness of each table name within a schema;

Warning "Table Name Max Length"

PostgreSQL allows only 63 characters in table names and cuts names if they are longer than this;

Warning "Column Definition"

Check if each table within the diagram owns at least one column;

Warning "Index Definition"

Check if each table within the diagram owns at least one index;

Warning "Primary Key Definition"

Check if a primary key is defined for each table within the diagram;

Warning "Reference Definition"

Check if each table within the diagram is linked with other tables;

Error "Auto-increment columns"

Check if each table within the diagram has no more than one auto-increment column, as otherwise PostgreSQL will not allow such table to be created;

Table Columns

Error "Column Name Uniqueness"

Check diagram tables for the uniqueness of each column name within the table;

Warning "Column Name Max Length"

PostgreSQL allows only 63 characters in column names and cuts names if they are longer than this;

Warning "Auto-increment Column Definition"

Check if each auto-increment column within a table is a part of a primary key;

Table Indexes

Error "Index Name Uniqueness"

Check diagram tables for the uniqueness of each index name within the table;

Warning "Index Name Max Length"

PostgreSQL allows only 63 characters in index names and cuts names if they are longer than this;

Warning "Duplicate Index Column"

Check if each table column is indexed only once;

References**Error "Reference Column Data Types"**

Check whether the linked columns are of the same data type for each diagram reference;

Domains**Error "Domain Name Uniqueness"**

Check diagram for the uniqueness of each domain name within a schema.

Stored Routine**Error "Stored Routine Name Uniqueness"**

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "Stored Routine Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Views**Error "View Name Uniqueness"**

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "View Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Error "View was created on a not existing table"

Check existence of tables, which are used in the view;

Error "View was created on a not existing column"

Check existence of table columns, which are used in the view.

After you click **OK** the check process will be displayed within the **Output** window and the result of the check will be displayed within the **Result** window in the same categorized view as described above.

Double-click on a warning or an error in the list opens the editor window for the appropriate object ([Table Editor](#), [Index Editor](#), or [Domain Manager](#)).

See also:

Diagram Objects: [Table Editor](#) | [Column Editor](#) | [Reference Editor](#) | [Index Editor](#)

8.2. Merge Diagram

The **Merge Diagram** tool of the **Database Designer for PostgreSQL** allows you to merge content of two diagrams. This allows you to create new cumulative diagram, which will include content of your two diagrams.

To merge two diagrams, please follow these steps:

1. [Open](#) two diagrams you want to merge. One of them will accumulate its own content and content of other diagram.
2. Start **Merge Diagram** tool by selecting the **Diagram | Merge Diagram** menu item.
3. Select diagrams you want to merge. Make sure that you have chosen **Options** tab of the **Merge Diagram** tool.

From diagram

Choose the *source diagram* from drop-down menu, which contains list of opened diagrams. Objects of this diagram will be added to the *destination* diagram.

To diagram

Choose the *destination diagram* from the drop-down menu, which contains the list of the opened diagrams. This diagram will contain its own objects and objects of *source diagram*. Set the name of the database user who will own the new database, or type DEFAULT to use the default (namely, the user executing the command).

4. Select the *source diagram* objects to merge. Go to the Objects tab. Click on the appropriate checkboxes to select objects you want to add to the *destination diagram*. To select/deselect all objects of particular type, click on appropriate checkbox near them. There are the following objects types: Domains, Notes, Tables, References (by default all objects are already selected).
5. Click **OK** to add the selected objects of the *source diagram* to the *destination diagram*.
6. Save the *resulted (destination)* diagram to a new file.

8.3. Repair Diagram

The **Repair Diagram** tool of the **Database Designer for PostgreSQL** allows to check and repair underlying physical structure of model file (*.pdd) if some reference integrity was broken.

During file opening Database Designer automatically checks file for reference corruption, and if **Repair** needed **Model Repairing** dialog appears. You may **Cancel** this operation. However, it is strongly recommended to proceed. After Repairing you may use **Save As** dialog to leave a backup of old model file.

To open manually the **Repair Diagram** dialog select the **Diagram | Repair Diagram** menu item.

Check **Color tables with duplicated IDs** to distinguish entities with non unique identifiers.

Use the **Fill Color** drop-down list to select the desired color for such action.

Check **Color references which are joined to tables with duplicated IDs** to mark the references which can not be unambiguously related to single table.

Use the **Fill Color** drop-down list to select the desired color for such action.

9. Database Functions

Database Designer for PostgreSQL supports the following database related functions:

[New Database Wizard](#), that allows you to create a new physical database;

[Database Generation](#), that allows to generate SQL script, that represents the diagram you developed and executes it on the database server;

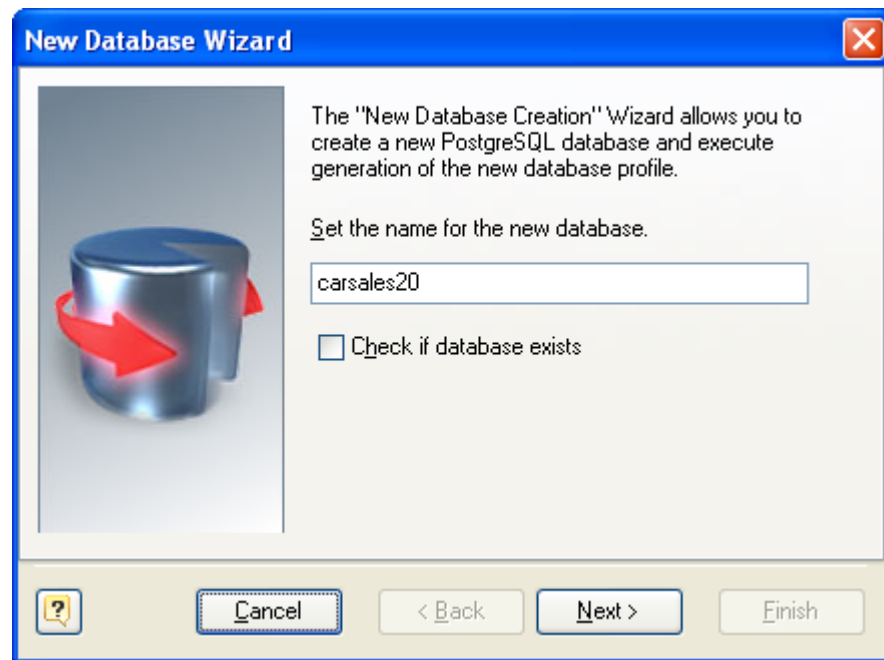
[Database Modification](#), that allows to generate SQL script that leads your database to the current state of your diagram.

9.1. New Database Wizard

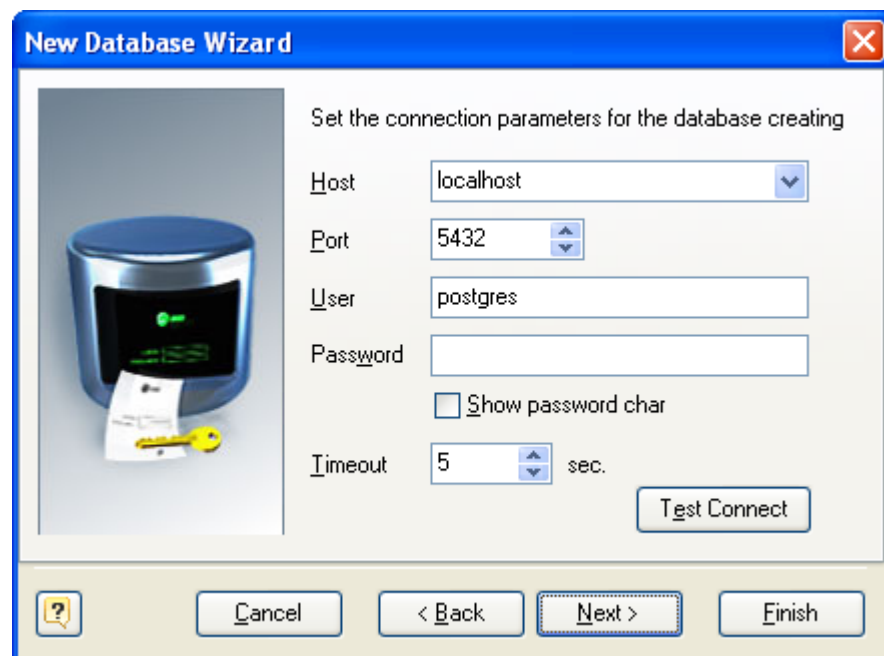
New Database Wizard allows you to create a new physical database. Later on you could generate its structure represented in the diagram by using [Database Generation](#). New Database Wizard takes the preferences you set for database using [Database Editor](#) and guides you through steps where you tune server connection parameters and other options for the new database. As a result the wizard creates a physical database and a new [connection profile](#) for it.

To open the wizard select the **Database | New Database Wizard** menu item. The wizard can be launched during [Database Generation](#) if you tick Run New Database Wizard checkbox there.

Let's explore the steps of the wizard.



1. The first step of the wizard prompts for the new database name. If you set the **Check if database exists** option on, then the new database will be created only if no database with such name exists on the server.



2. The next step of the wizard allows you to define the connection properties for connecting to the new database server. You should use user that have enough rights to create a new database on the server. The properties you tune on this step are:

Host

Sets the PostgreSQL server address.

Port

Sets the PostgreSQL server port.

User

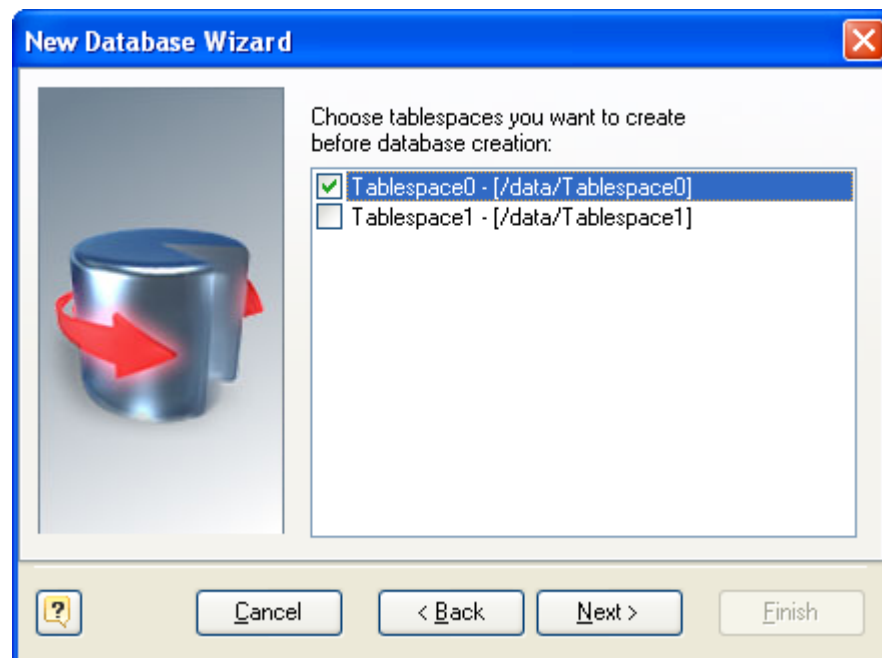
Sets the PostgreSQL user login.

Password

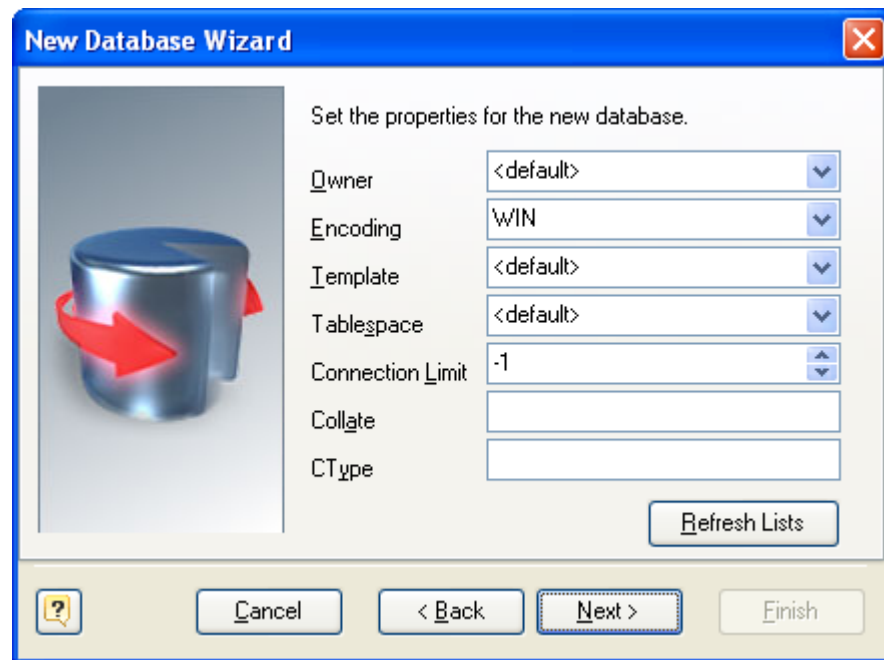
Sets the PostgreSQL user password

Timeout

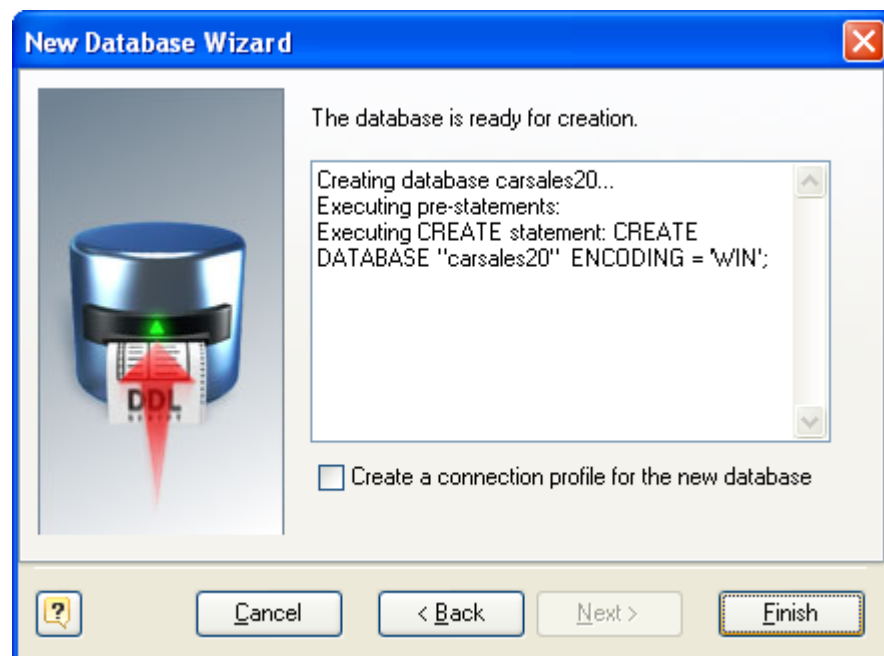
This option defines an interval of time that the **Database Designer for PostgreSQL** will try to connect to the PostgreSQL server.



3. Here you can choose what tablespaces should be created before CREATE DATABASE statement execution.



4. This step allows you to tune parameters of a new database. The dialog shows the parameters you already set by using [Database Editor](#). But it's possible to tune them here.



4. The final step of the wizard allows you to chose if you want to create a new connection profile for the database. Additionally, it shows the process and result of database creation.

To start creating the database with all the parameters you have set, click the **Finish** button.

After wizard has finished, the [Database Generation](#) process will continue if the wizard had been launched from it.

9.2. Database Generation

The **Database Generation** tool can generate SQL script, that represents the diagram you developed and executes it on the database server.

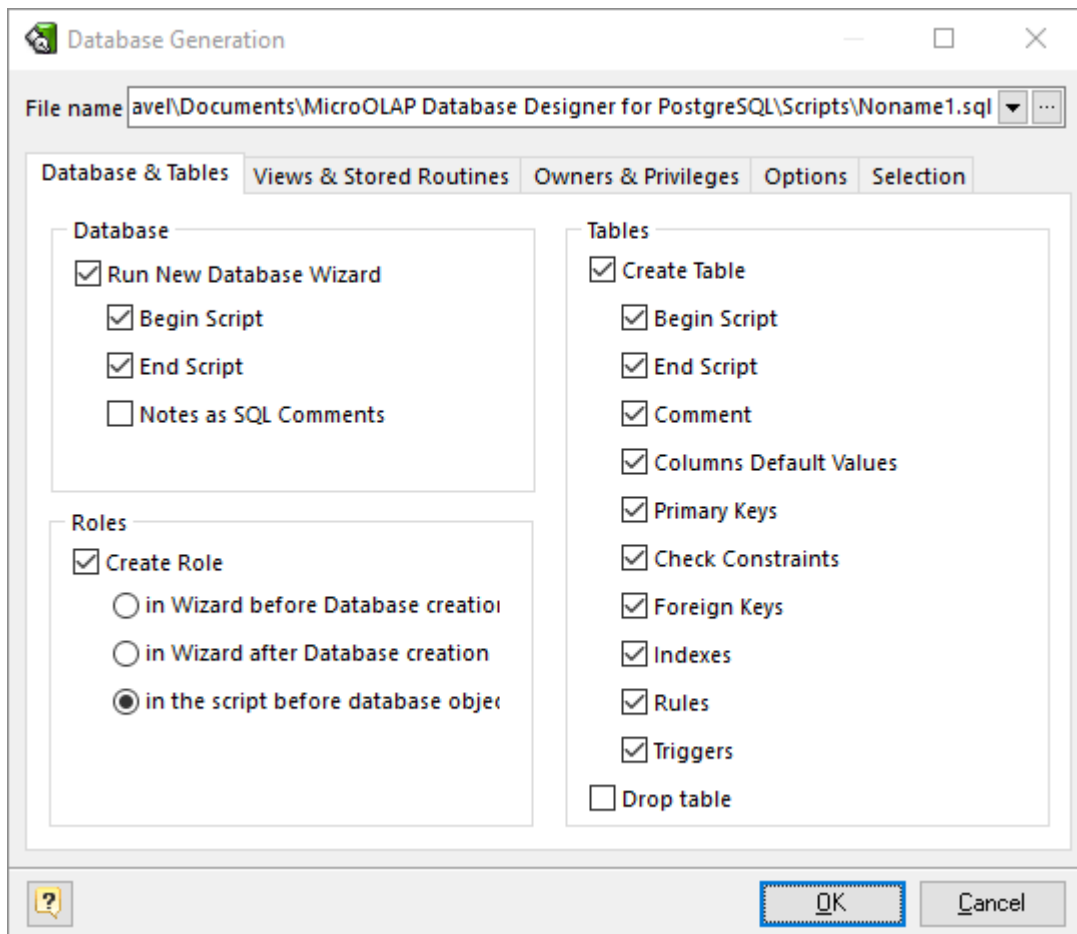
You can generate database in two ways:

- Directly execute a script on a PostgreSQL server. Please examine [Connect to a Database](#) section to explore the database connection process;
- Generate a script to be executed on PostgreSQL server at a later time.

In both cases, the database generation commands are saved in a script file. You must always provide path to the script file.

To generate database, start the **Database Generation** tool by selecting the **Database | Generate Database** menu item or pressing **Ctrl-G**. **Database Generation** tool consist of five tabs, which contain SQL generation options. Let's explore them. The following pictures demonstrate **Database Generation** tool interface.

Database Generation



File name

This field allows you to set file, in which generated SQL statements will be stored. Click on the ... button near the field to browse to file on the file system.

Database Group

Run 'Create New Database Wizard'

Mark this checkbox if you want to create a new physical PostgreSQL database before generating the database structure presented in the diagram.

Begin Script

This option enables inserting the begin script before the CREATE DATABASE statement.

End Script

This option enables inserting the end script after the CREATE DATABASE statement.

Notes as SQL Comments

This option enables using notes of database as comments in SQL script.

Role Group

Create Role

This option enables generation of model roles.

in Wizard before Database creation

This option enables executing the CREATE ROLE statements before the CREATE DATABASE statement right in **New Database Wizard**.

in Wizard after Database creation

This option enables executing the CREATE ROLE statements after the CREATE DATABASE statement right in **New Database Wizard**.

in the script before database objects

This option enables executing the CREATE ROLE statements in the script before tables, views, stored routines etc.

Tables Group

Create Tables

This option enables generation of tables.

Begin Script

This option enables inserting the begin script (it can be set using [Table Editor](#)) before the CREATE TABLE statement.

End Script

This option enables inserting the end script (it can be set using [Table Editor](#)) after the CREATE TABLE statement.

Comment

This option enables inserting of table comments in SQL script.

Columns Default Values

This option enables generation of columns' default values.

Create Primary Keys

This option enables generation of table primary keys.

Create Check Constraints

This option enables generation of check constraints for the tables.

Create Foreign Keys

This option enables generation of foreign keys for the tables.

Create Indexes

This option enables generation of indexes for the tables.

Create Rules

This option enables generation of rules for the tables.

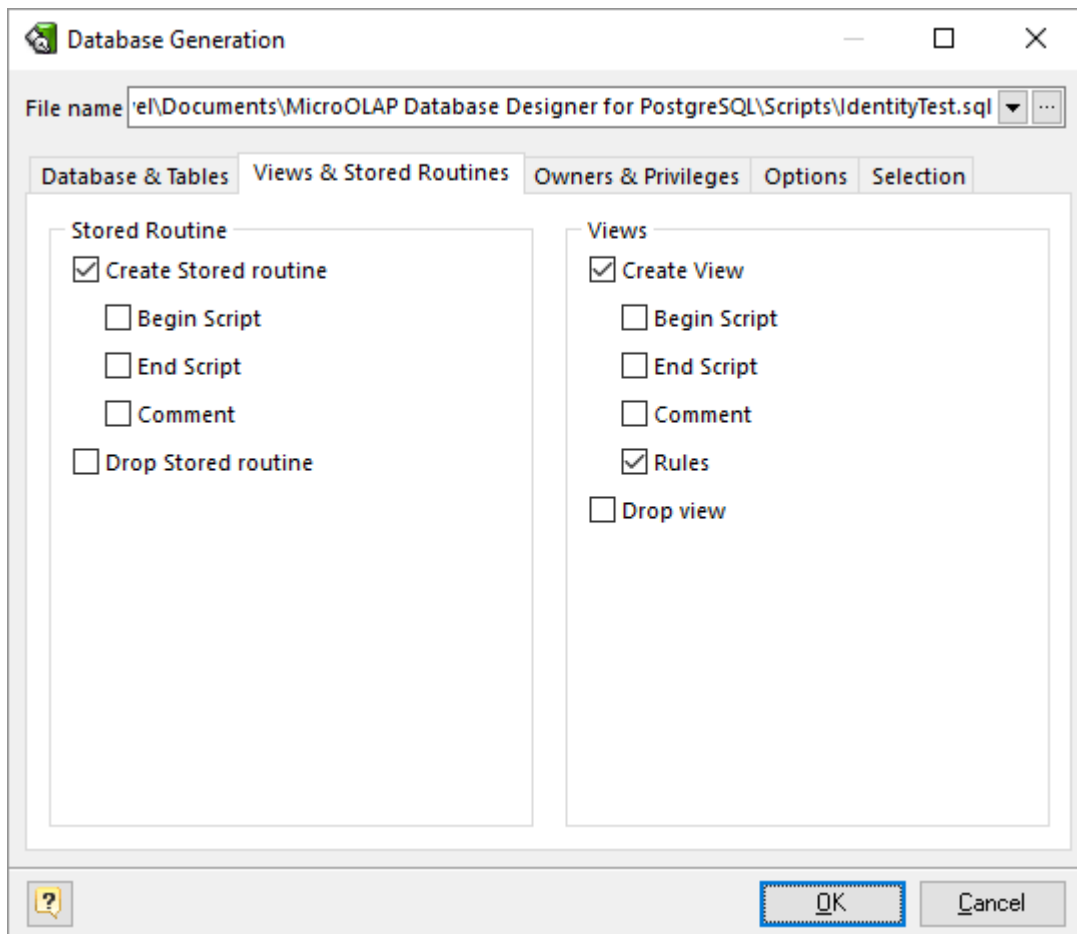
Create Triggers

Generate triggers set on table.

Drop table

This option enables dropping old tables if they were already exist. I.e. enables generating DROP TABLE "<TABLE_NAME>" statement before CREATE TABLE statement.

Views & Stored Routines



Stored Routines Group

Create Stored Routine

This option enables generation of stored routines (i.e. stored functions and procedures).

Begin Script

This option enables inserting the begin script before the SQL generation statement.

End Script

This option enables inserting the end script after the SQL generation statement.

Comments

This option enables showing of stored routines comments in SQL script.

Drop Stored Routine

This option enables dropping old stored routines if it were already exist.

Views Group

Create View

This option enables generation of database views.

Begin Script

This option enables inserting the begin script before the SQL generation statement.

End Script

This option enables inserting the end script after the SQL generation statement.

Comment

This option enables showing of view comments in SQL script.

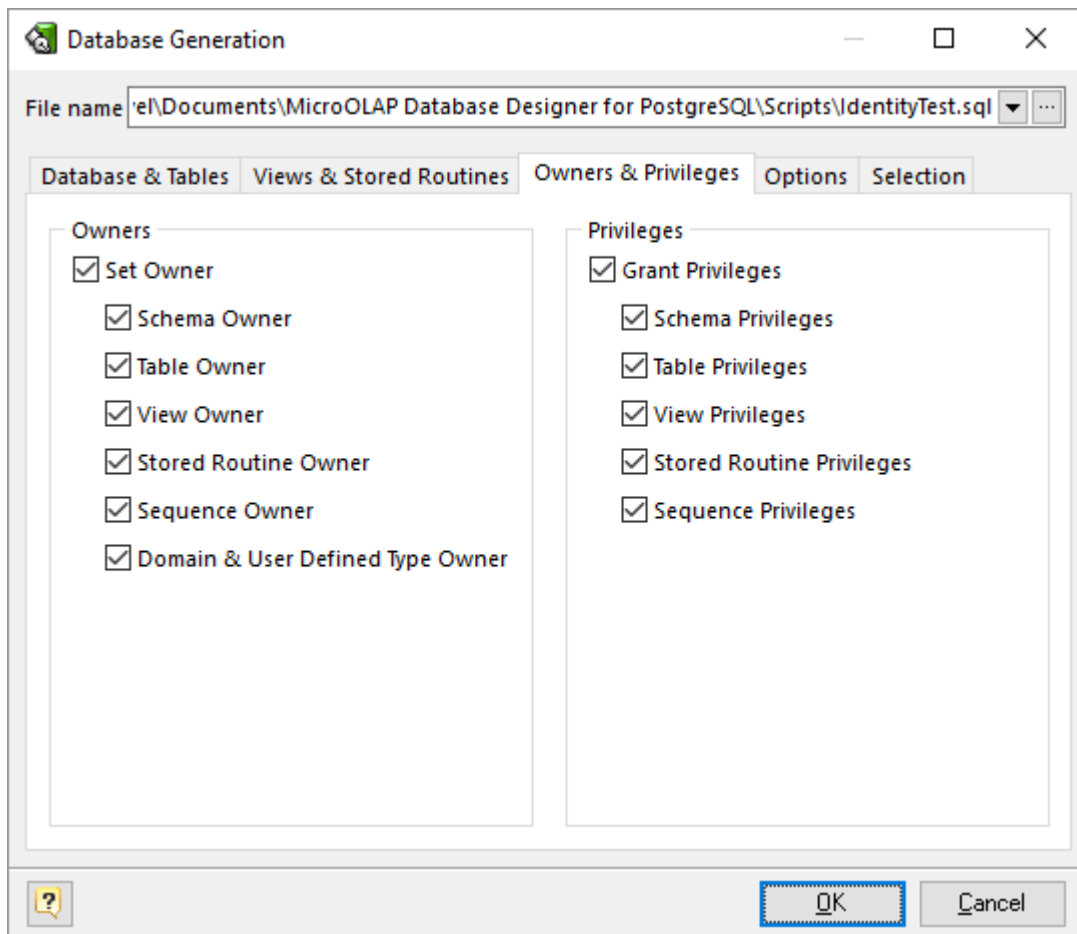
Create Rules

This option enables generation of rules for the views.

Drop Views

This option enables dropping old views if it were already exist.

Owners & Privileges

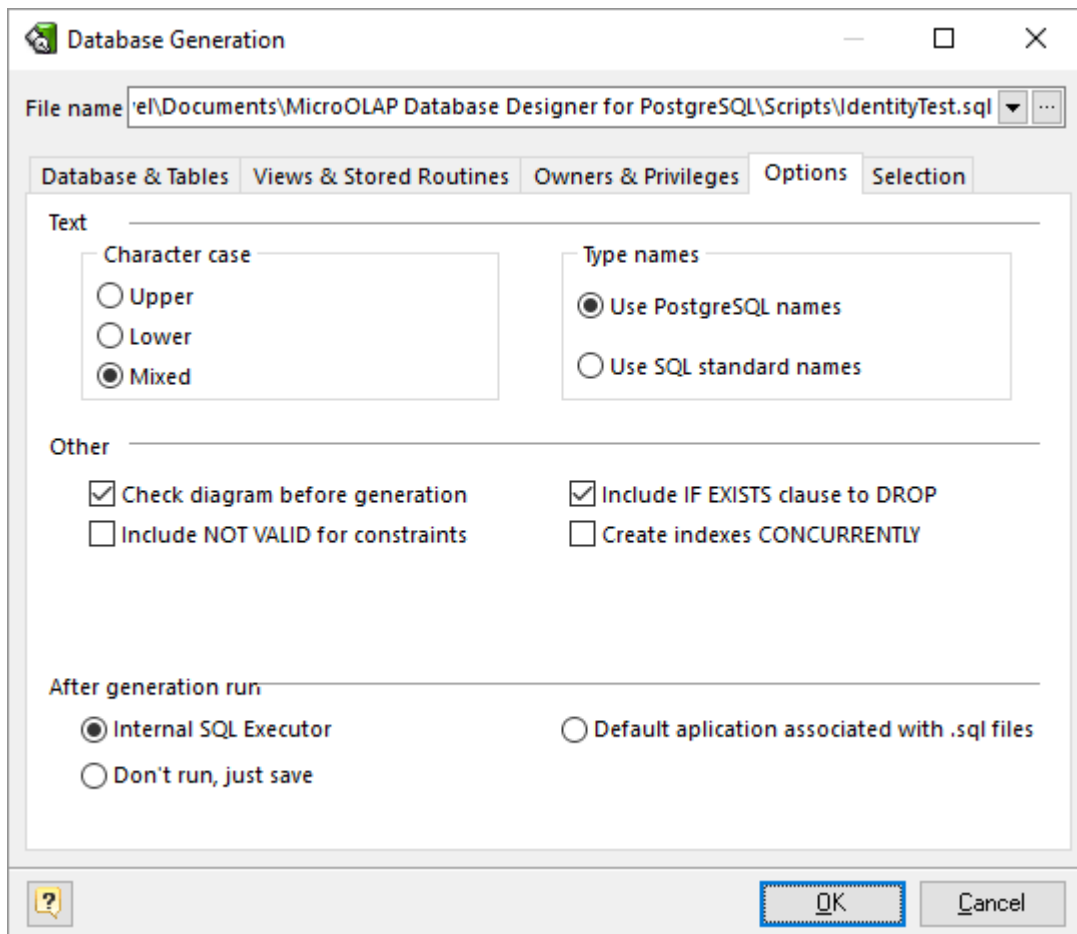
**Owners**

Defines for which objects owner information should be generated.

Privileges

Defines for which objects privileges information should be generated.

Options



This tab allows set generation-related options.

Character case

Defines case of characters, which will be used in generated script. Use Mixed option to leave the characters unmodified.

Type names

Determines what type names will be used in the generated SQL, PostgreSQL (*int2, int4, int8, varchar, char, bool, float8, float4*) or standard ones (*smallint, integer, bigint, character varying, character, boolean, double precision, real*).

Before generation:

Check diagram

Enables checking the diagram before generation.

Drop objects

Include IF EXISTS clause

Do not throw an error if the object does not exist. A notice is issued in this case while executing generated script. Server must be 8.2.x or higher.

After generation, run**Internal SQL Executor**

Send the generated SQL statements into the internal [SQL Executor](#).

Default application associated with SQL files

Send the generated SQL statements into the application associated with SQL files

Don't run, just save

Do nothing with output script, just save it on disk.

Selecting objects to generate

You can select diagram objects you want to generate in SQL script or database. Use the **Selection** tab of the **Database Generation** tool for it.

There are several subtabs: **Tables**, **Stores Routines**, **Views**, **Types & Domains** and **Sequences**. Each of which allows you to select appropriate diagram objects to generate.

To enable particular objects generation, click on the checkbox near it.

The default selection of objects to generate depends on their **Generate** property.

Pay attention to the buttons on the **Selection** tab:

Select ALL - checks on all checkboxes.

Deselect ALL - checks off all checkboxes.

Use graphical selection - checks on checkboxes for objects, depending on [diagram selection](#).

You can change the order of tables in which they will be placed in the generated SQL script. Use the buttons with arrows for this.

Generating, customizing and executing SQL

Click on the **OK** button on the **Database Generation** tool to generate SQL script. The generated SQL script will be stored in the file you have set.

If you have established connection to database, the [SQL Executor](#) with generated SQL statements will appear.

You can easily customize statements for your needs. And then send them to the database server by clicking on the **Execute SQL** button.

Please, examine [SQL Executor](#) section to know more about it.

See also:

Database Accessing: [SQL Executor](#)

Database Functions: [Database Modification](#)

9.3. Database Modification


Overview

Once you have changed your diagram, it's usually necessary to apply these changes to your database. It's easy to do with the **Database Modification** tool. The **Database Modification** tool can generate SQL script that leads your database to the current state of your diagram.

You can synchronize database in two ways:

- Directly execute a modification script on a PostgreSQL server. Please examine [Connect to a Database](#) section to explore the database connection process;
- Generate a modification script for executing at a PostgreSQL server some time later.

In both cases, the database modification commands are saved in a script file. You must always provide the path to the script file.

 Please note, that database modification usually cause multiple complex statements for database structure modification. It is possible that some of them may not execute correctly due to some physical reason. It's recommended to make a backup of your database before applying structure changes to database.


That's how the Database Modification tool works:

1. Reverse engineers your existing PostgreSQL database
2. Compares the result with your current database diagram
3. Creates a list that contains differences between database objects and diagram objects
4. After analyzing the difference list, creates necessary SQL statements, that modify database structure.

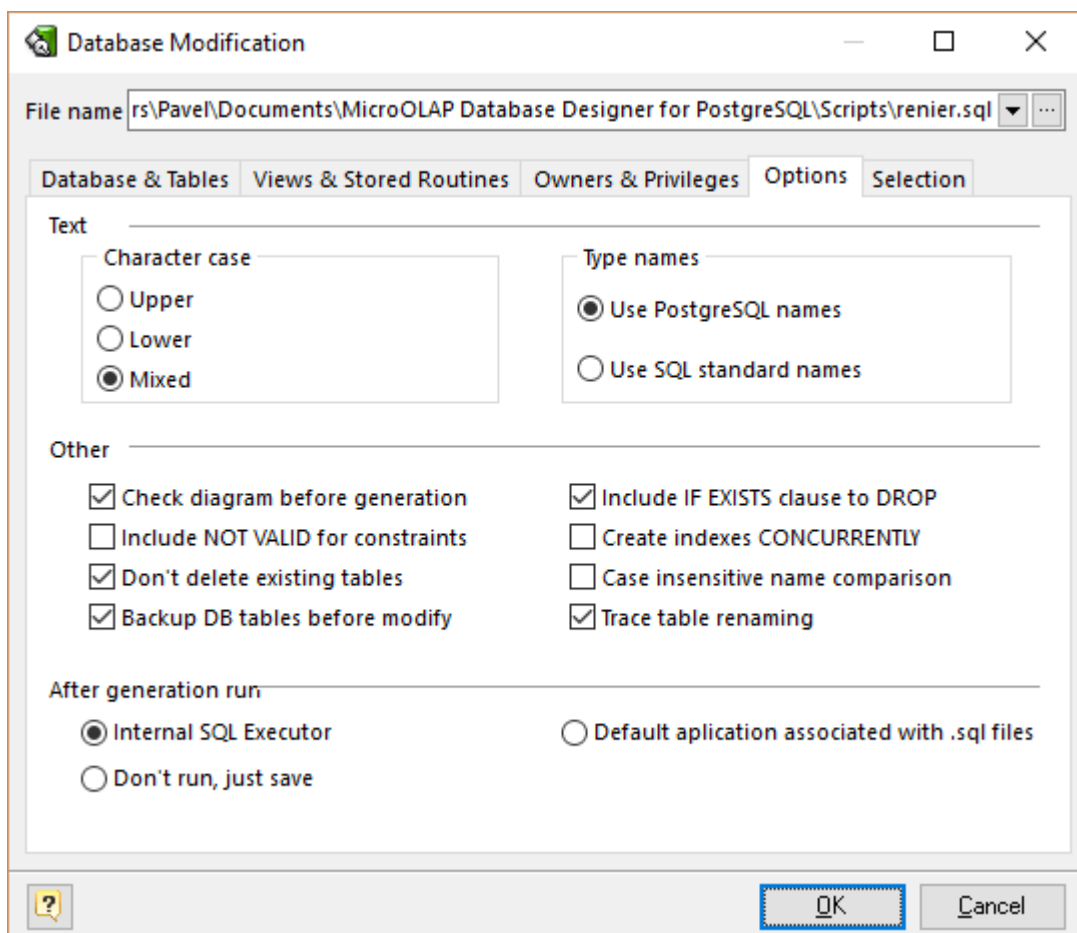
If some table is going to be modified, **Database Designer for PostgreSQL** makes a backup copy of that table, so you can restore data and table structure later on if there will be some errors during the table structure alteration.

Database Modification Tool

To modify your database, start Database Modification tool by selecting the **Database | Modify Database** menu item or pressing Ctrl-M.

 Interface of the **Database Modification** dialog is much the same as the [Database Generation](#) dialog. The only differences are persist on the **Options** tab.

Modification Options



In the Options tab of the Database Modification tool you can set modification options.

Don't delete existing tables

This option disables deleting tables that already exist in the database, but don't exist in your diagram.

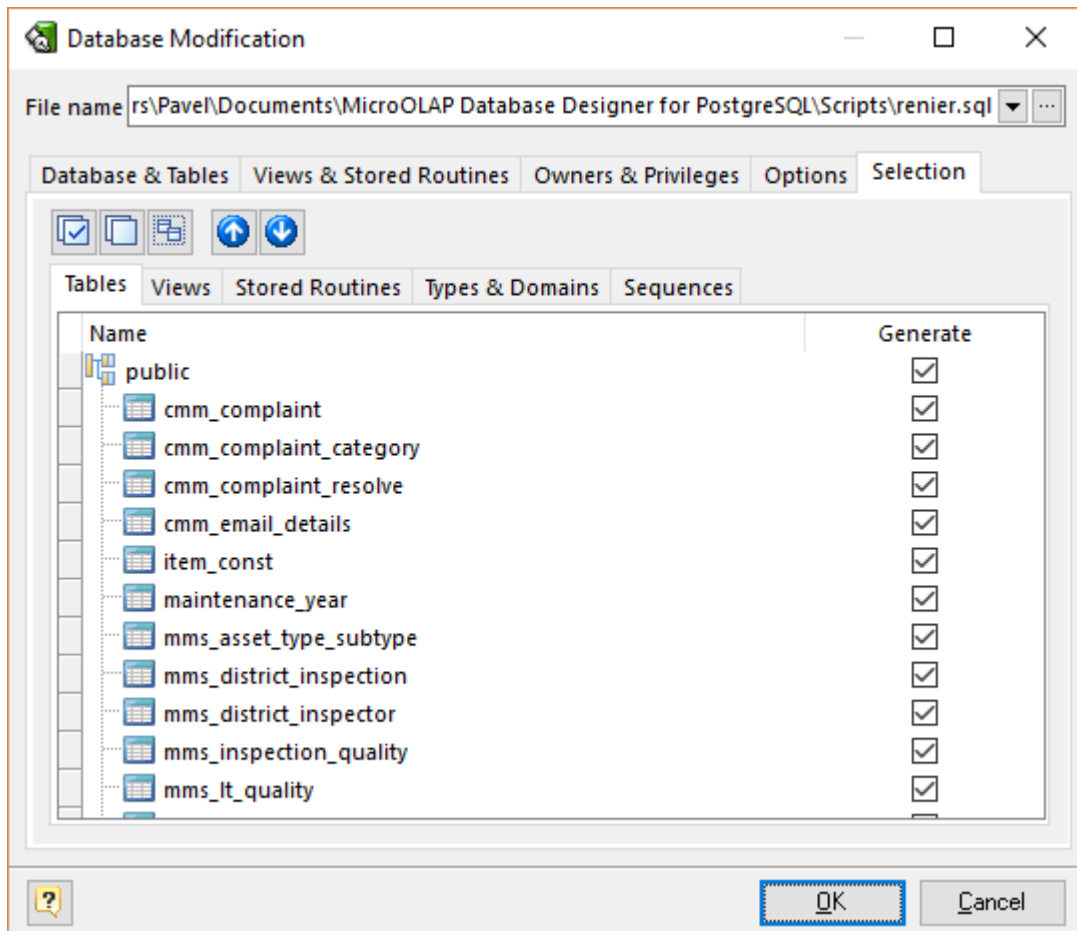
Backup DB tables before modify

This option enables creating backup of tables to be modified.

Trace table renaming

Try to find tables that has been renamed in the diagram by comparing their structure with the structure of the tables in the database. If a structure of a table in the database is identical to a structure of a table in the diagram, but such tables have different names, a table in the database is going to be renamed to match the diagram table's name.

Selecting objects to generate



You can select diagram objects which need to be modified in the database. Use the **Selection** tab of the **Database Modification** tool for this.

There are several subtabs: **Tables**, **Views**, **Stores Routines**, **Types & Domains** and **Sequences**. Each of which allows to select appropriate diagram objects to modify.

To enable particular objects generation, click on the checkbox near it.

The default selection of objects to generate depends on their Generate property. Pay attention to the buttons on the Selection tab:

Select ALL - checks on all checkboxes

Deselect ALL - checks off all checkboxes

Use graphical selection - checks on checkboxes for objects, depending on [diagram selection](#).

You can change the order of tables in which they will be placed in the generated SQL script. Use the buttons with arrows for this.

Generating, customizing and executing SQL

Click on the **OK** button on the **Database Modification** tool to generate SQL script. The generated SQL script will be stored in the file you have set. Also [SQL Executor](#) with generated SQL statements for database modifications will appear.

You can easily customize statements according to your wants and wishes. And then send them to the database server by clicking on the **Execute SQL** button. Please examine [SQL Executor](#) section to know more about it.

See also:

Database Accessing: [SQL Executor](#) | [Connect to a Database](#)

10. Database Accessing Tools

Database Designer for PostgreSQL supports the following database accessing tools:

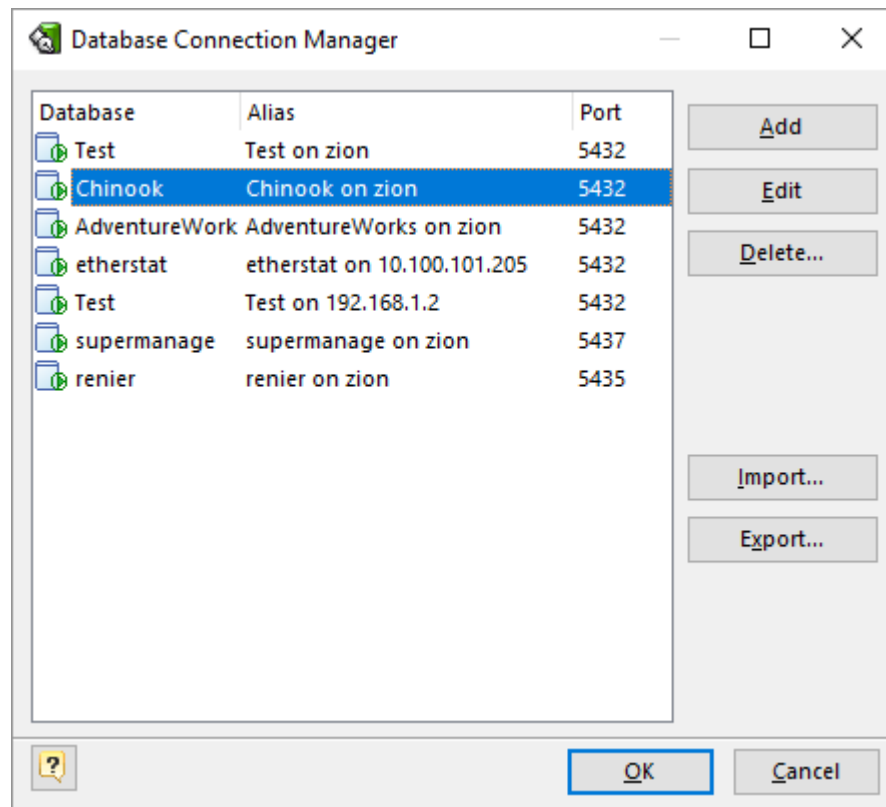
[Database Connection Manager](#), which allows you to connect to a local or remote PostgreSQL database for Database Generation, Reverse Engineering, Database Modification, or running SQL scripts.

[SQL Executor](#), that allows to send SQL queries to the connected PostgreSQL database and display the result.

10.1. Database Connection Manager

Database Connection Manager is a useful tool, which allows you to connect to a local or remote PostgreSQL database for [Database Generation](#), [Reverse Engineering](#), [Database Modification](#), or [running SQL scripts](#).

Database Connection Manager uses connection profiles, which allow you to set all the connection parameters for a specific database only once, and quickly connect to this database later.




The dialog contains a list of available connection profiles. To connect to a database, select one of the defined profiles in the list, and click **OK**. Connection is assigned to your current diagram, so that each opened diagram could have its own connection.

To add a profile to the list, click the **Add** button and enter the connection properties using [Connection Profile Editor](#) launched.

Click **Edit** to change the profile connection properties.

To remove a profile from the list, click the **Delete** button.

You may share your profiles with others and vice versa - for this use **Import** and **Export** buttons.

 Please note, that connection can be established with any PostgreSQL server that works under any OS, including Windows, Linux, FreeBSD and others.

See also:

[Connection Profile Editor](#)

Database Functions: [Database Generation](#) | [Database Modification](#)

Reverse Engineering and Import: [Reverse Engineering and Import Overview](#)

Database Accessing Tools: [SQL Executor](#)

10.1.1. Connection Profile Editor

The **Connection Profile Editor** allows you to set up the database connection parameters and other additional preferences. It can be launched during the process of a new profile creation and also can be used through the [Database Connection Manager](#).

The **Profile Editor** consists of several tabs. Please see the detailed description below.

Connection

The **Connection** tab allows you to define the PostgreSQL server connection parameters and select the required database.

Host

Sets PostgreSQL server address.

Port

Sets PostgreSQL server port.

User

Sets PostgreSQL server user login.

Password

Sets PostgreSQL server user password.

Show password chars


This option enables showing real password chars instead of asterisks in the Password field.

Timeout

This option defines the time interval for the **Database Designer for PostgreSQL** to connect to the PostgreSQL database.

Database Encoding

Sets the encoding of current database connection.

 Please note, that list that represents possible encodings consist of only one row: <default>. You may manually type in preferred value, e.g. SQL_ASCII. Or you may set all connection options and click on the **Test Connect** button. Then all available encodings will appear in the list.

Database

Selects a database from the list of the ones, which are available on the server.

SSL

The following options are used to establish secure SSL connection.

Use SSL Connection

Enable SSL connection.

SSL Client Certificate

Allows to select the file containing the client certificate.

SSL Private key

Allows to select the file containing the client private key.

SSH

Connecting through SSH tunnel gives you two main advantages:

1. You can connect to PostgreSQL server even if the direct connections to it are not allowed. It's a typical situation on a shared hosting.
2. The traffic between the server and **Database Designer for PostgreSQL** is secured.

If you set the SSH tunnel, this may request you to change the settings in the **Connection** tab: imagine that your current computer is the server you set the SSH tunnel with. So, perhaps you will have to change the value of the **Host** field to localhost.

Please note, that using SSH tunnel requires SSH service available on server you are connecting with.

The SSH tab allows you to establish connection to PostgreSQL using secure SSH tunneling. To connect to PostgreSQL through SSH check the **Enable SSH Tunneling** option and specify the SSH connection parameters.

SSH Host

Sets SSH server address.

SSH Port

Sets SSH server port.

SSH User

Sets SSH server user login.

SSH Password

Sets SSH server user password.

Show password chars

This option enables showing real password chars in the SSH Password field instead of asterisks.

Use compression

Check this option to use the SSH compression, defined by the Compression Level option

Compression Level

Set Compression to any value between 1 and 9 (1 for minimum compression, 9 for maximum compression). Typically, SSH clients use compression level 6 for optimum performance.

SSH Timeout

This option defines an interval of time that the **Database Designer for PostgreSQL** will try to connect to the SSH host.

See also:

[Database Connection Manager](#)

10.2. SQL Executor


You can send SQL queries to the connected PostgreSQL database and display the result. The queries can contain any possible statements, e.g. UPDATE, DELETE, INSERT, SELECT statements etc. Note, that it is possible to run multiple SQL queries simultaneously.

The result of the SELECT-containing queries will be shown in the grid-based dialogues.

To execute SQL queries:

1. Connect to the database using [Database Connection Manager](#)
or
Use the already established database connection.
2. Call the **SQL Executor** by selecting **Database | SQL Executor** menu item or pressing **Ctrl-Shift-E**.
3. Enter one or more SQL queries.
4. Click the **Validate SQL** button on the dialog window toolbar to check your queries. **Database Designer Validator** can check SQL only for grammar correctness. This means that it can't check correctness of the database objects name using, i.e. if you've been mistaken in the name of some table. However, you may execute an SQL without validating it: it will be checked before executing automatically.
5. Press **F9** or click the **Execute SQL** button on the dialog window toolbar to execute your queries. A data grid window will be displayed for each SELECT-based query. To close the data grid window, click on the **Close** button. The execution status for each query will be displayed at the bottom of the **SQL Executor** dialog window.

6. Press **Alt-F9** or click **Execute SQL in Single Transaction** button on the dialog window toolbar to execute your script in the context of the single transaction without pre-validation. This means the whole script will be sent to server without parsing and validation.

You can save your queries into a file. Click on the Save () button on the SQL Executor toolbar.

Please note, that SQL Executor dialogue window is used by [Database Generation](#) and [Database Modification](#) tools.

See also:

Database Generation: [Database Generation](#)

Database Functions: [Database Modification](#)

10.3. Connect to a Database

To start working with a database using [Generate Database](#), [Database Modification](#), [SQL Execute](#) and [Reverse Engineering functions](#), you have to connect to it first. To do so, please follow these steps:

1. Select the **Database | Connect** file menu item, or press **Ctrl-Shift-N**. The [Database Connection Manager](#) will appear.
2. Select one of the defined database profiles in the list or add a new one.
3. Click **OK** to establish connection to the database.

See also:

[Database Modification](#)

Database Generation: [Database Generation](#)

Database Tools: [SQL Executor](#) | [Database Connection Manager](#)

Reverse Engineering and Import: [Reverse Engineering PostgreSQL Database](#)

10.4. Disconnect from a Database

To break the current connection from the database, use the **Database | Disconnect** menu item.

11. Reverse Engineering And Import

With **Database Designer for PostgreSQL** you can extract database tables, attributes, relationships, indexes and other objects from existing databases.

Database Designer for PostgreSQL supports a number of different communication methods: direct connection to database, ADO/ODBC support. This enables you import different objects from multiple databases.

Examine these import facilities:

1. [Reverse Engineering PostgreSQL Database](#). You can connect to PostgreSQL database directly and reverse engineer its objects.
2. [Import from Access Database](#). This tool allows you to import tables, attributes, relationships, indexes to your diagram from Microsoft Access database file.
3. [Universal Reverse Engineering](#). Use this tool to reverse engineer/import the structure of various databases. It allows you to reverse engineer the following databases: Sybase ASE and ASA, Oracle, Informix, MSSQL, DB2, DBF and many others that accessible by OLEDB or ODBC.

See also:

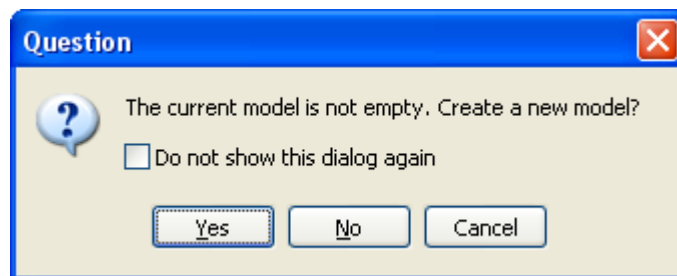
Reverse Engineering and Import: [Reverse Engineering PostgreSQL Database](#) | [Import from Access Database](#) | [Universal Reverse Engineering](#)

11.1. Reverse Engineering PostgreSQL Database

You can reverse engineer an existing PostgreSQL database. This means that you can extract the database tables, attributes, relationships, indexes and other objects from the database to your diagram.

To reverse engineer PostgreSQL database:

1. Select **File | Reverse Engineer | PostgreSQL database** or press **Ctrl-R**.
2. If the currently opened diagram already contains some objects, a warning dialog box will appear:

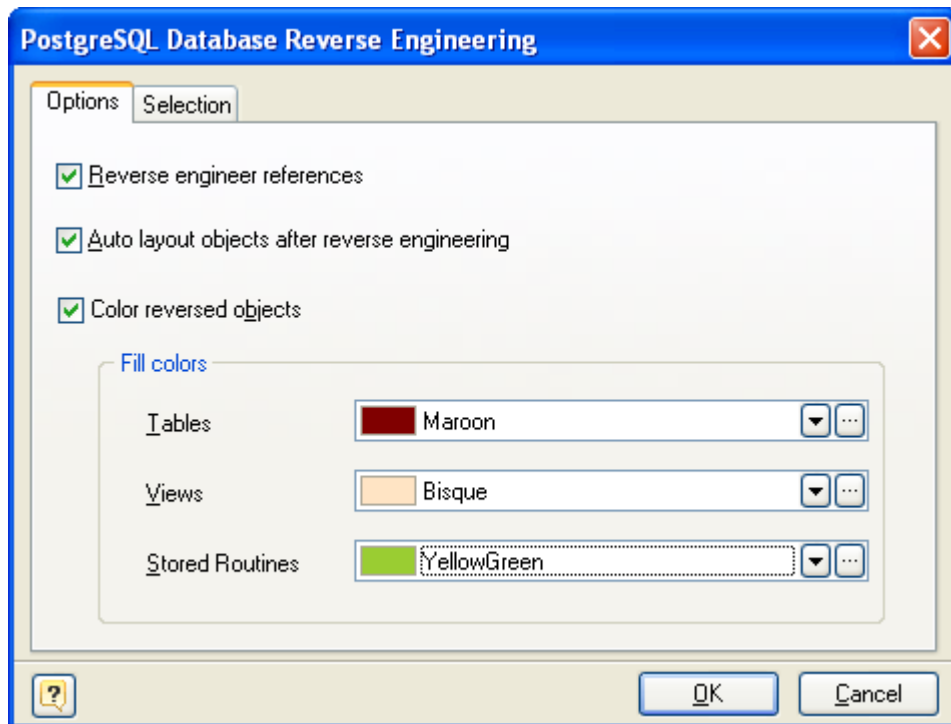


It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram.

Click **Yes** to create a new diagram or **No** to use the currently opened diagram. Click **Cancel** to cancel reverse engineering.

Select **Do not show this dialog again** to disable future notifications.

3. If connection for the current diagram has not been established, [Database Connection Manager](#) will be shown. Select a profile from the list of the available ones or create a new profile to connect to the database you want to reverse engineer.
4. The **PostgreSQL Database Reverse Engineering** tool will be shown. You can set the reverse engineering options in the **Options** tab.



Reverse Engineer references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

Auto layout objects after reverse engineering

Select this option if you want to Designer perform auto layout of reversed objects

Color reversed objects

This option allows you to set fill colors for newly reversed objects. It may be useful if you're performing Reverse Engineering into non empty diagram, thus you may distinguish old objects from reversed ones.

5. In the Selection tab of the **PostgreSQL Database Reverse Engineering** tool you can choose the objects you want to reverse engineer.

There are several subtabs: **Tables**, **Views**, **Stores Procedures** and **Types & Domains**. Each of them allows you to select appropriate database objects. Click on the checkbox near the object to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all objects in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

7. The reversed database objects will be placed in your diagram.

See also:

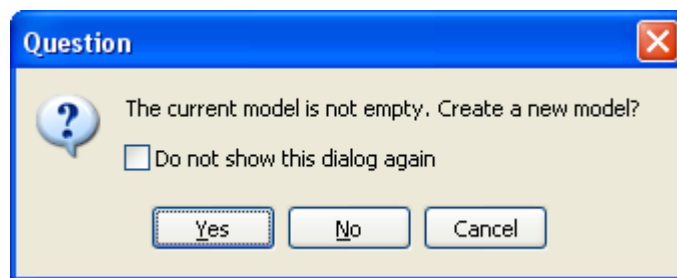
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Import from Access Database](#) | [Universal Reverse Engineering](#) | [SQL Reverse Engineering](#)

11.2. Import from Access Database

You can reverse engineer an existing Microsoft Access database. This means that you can extract the database tables, attributes, relationships, indexes and other objects from Microsoft Access database file to your diagram.

To reverse engineer Microsoft Access database:

1. Select **File | Reverse Engineer | Microsoft Access database**.
2. If the currently opened diagram already contains some objects, a warning dialog box will appear:



It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram.

Click **Yes** to create a new diagram, **No** to use the currently opened diagram. **Cancel** to cancel reverse engineering.

Select **Do not show this dialog again** to disable future notifications.

3. The **Access Reverse Engineering** tool will be shown. First, type in the full path to Access database in the **Access File** tab.

4. You can set the reverse engineering options in the **Options** tab.

Tables only

Reverse engineer only tables ignoring views.

Tables and Views

Reverse engineer both tables and views.

Garbage symbols remove/replace

Defines the symbols that will be replaced in the names of the objects.

Replace with

Defines the garbage replacement symbol.

Build references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

Automatically rebuild references when no reference is reversed

If there are no physical references extracted, it is possible to build them from logical structure of the database.

Enabling this feature leads to automatic reconstruction of references. Such reconstruction works by the following scheme: each column of the table is being compared with all primary keys of other tables, and if the column name and data type match one of the primary keys, a reference between the source column and the key column will be created.

This option is available for modification only if Build references is checked.

Tables in a Diagram row

This option defines how reversed tables will be disposed in the diagram. Reversed tables will be placed in the diagram in rows with equal distance, this option determines how many tables maximum there will be in one row.

5. In the **Selection** tab of the **Access Reverse Engineering** tool you can choose the tables you want to reverse engineer.

Click on the checkbox near the table to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all tables in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

7. The reversed database objects will be placed on your diagram.

See also:

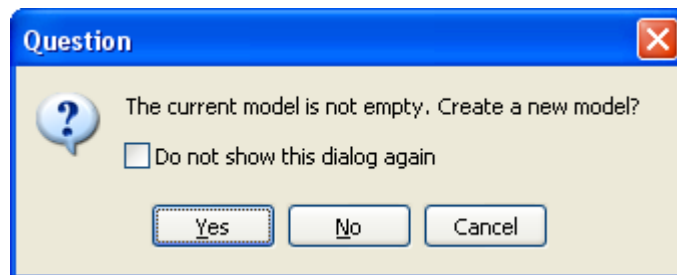
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Universal Reverse Engineering](#) | [SQL Reverse Engineering](#)

11.3. Universal Reverse Engineering

You can reverse engineer a number of databases, such as Sybase ASA and ASE, Oracle, Informix, MSSQL and others. This means that you can extract tables, attributes, relationships, indexes and other objects. It is possible with the **Universal Reverse Engineering** tool, which connects to a number of databases through OLEDB or ODBC link. So, to reverse engineer a particular database, you need a corresponding OLEDB provider, or ODBC driver. In most cases, such libraries are installed in the system with the help of database client applications.

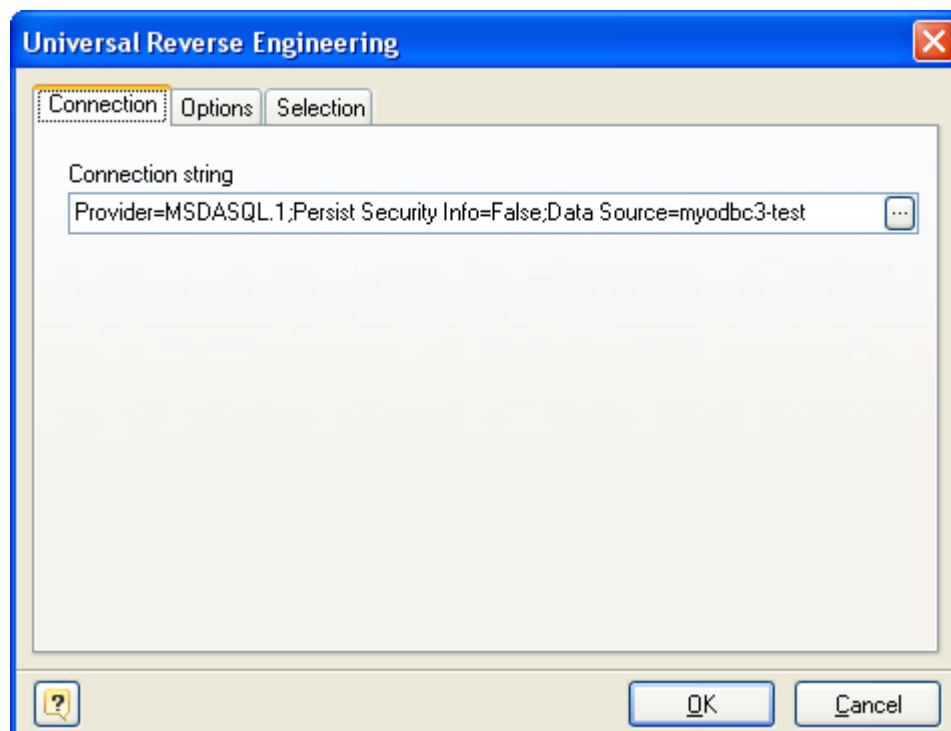
To reverse engineer a database through OLEDB or ODBC link:

1. Select **File | Reverse Engineer | Universal Reverse Engineering** menu item.
2. If the currently opened diagram already contains some objects, the following dialog box will appear:

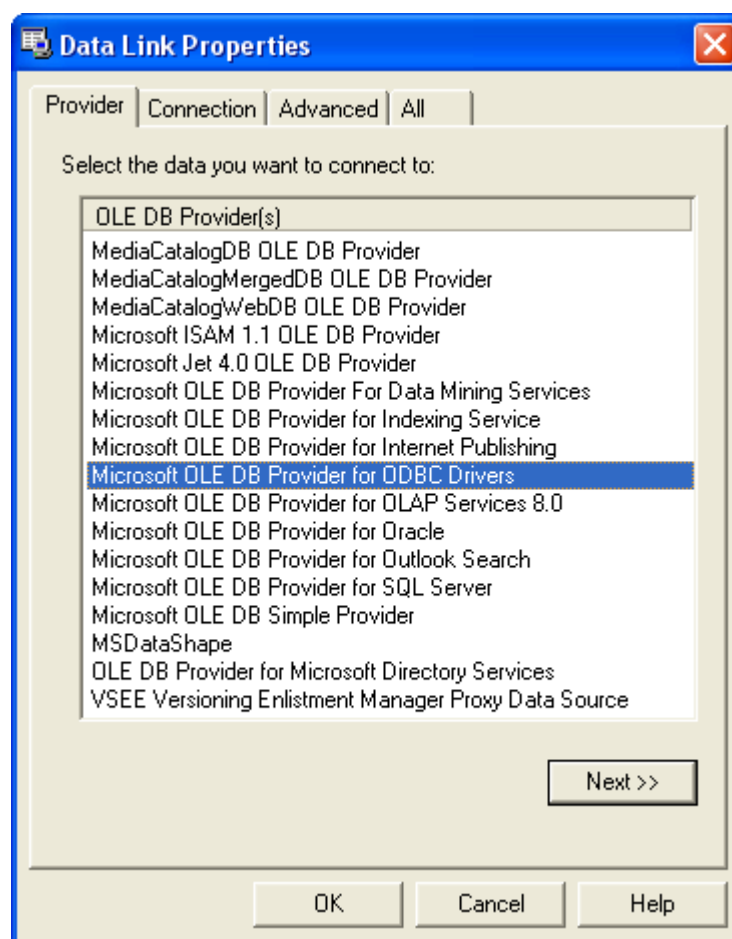


It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram. Click **Yes** to create a new diagram, **No** to use the currently opened diagram, or **Cancel** to cancel reverse engineering. Select **Do not show this dialog again** to disable future notifications.

3. The Universal Reverse Engineering tool will be shown. First, provide full OLEDB connection string in the **Connection tab.**

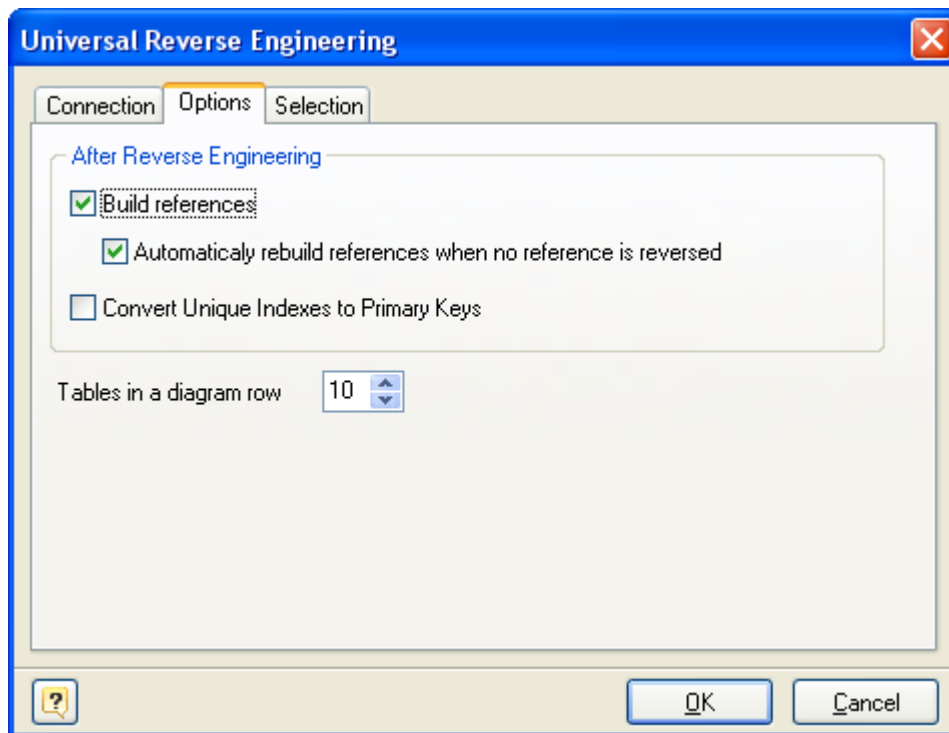


Click on the ... button to call the standard system **Data Link Properties** dialog window, which helps you to build OLEDB connection string:



To use ODBC drivers, click on the Microsoft OLE DB Provider for ODBC Drivers. To use the native OLEDB provider, click on the appropriate item in the list. Browse through tabs to set other tabs and connection properties. Click on the **Help** button in the bottom of the dialog window to learn more about OLEDB connections. Click on the **OK** button to store the data link properties in the **Connection string** field of the **Universal Reverse Engineering** dialog.

4. You can set reverse engineering options in the Options tab:



Build references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

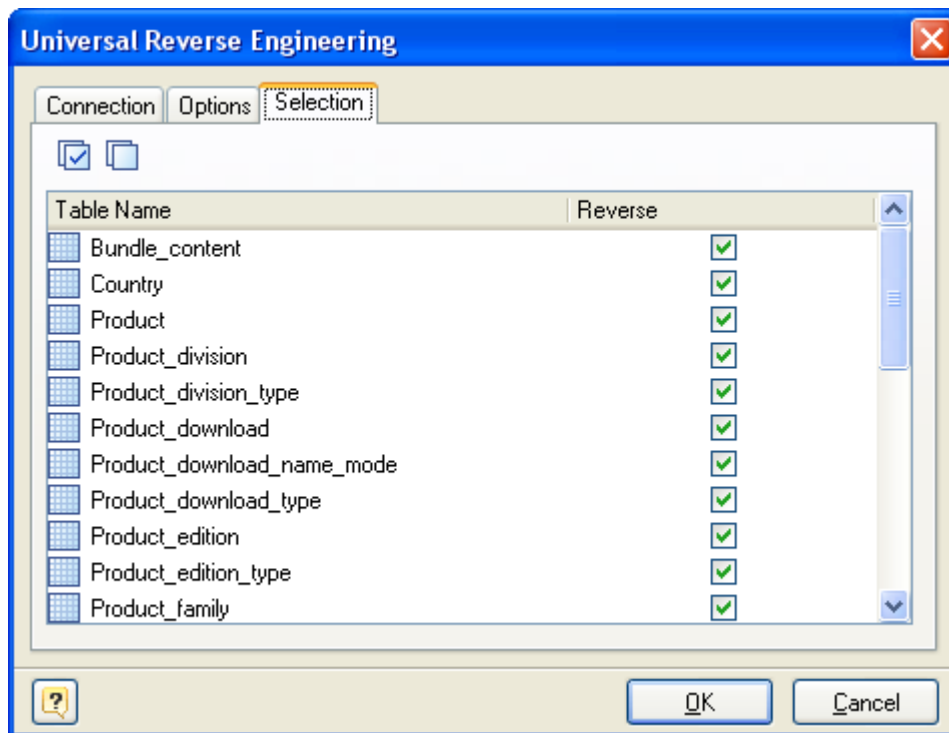
Automatically rebuild references when no reference is reversed

If there are no physical references extracted, it is possible to build them from the logical structure of the database. Enabling this feature leads to automatic reconstruction of references. Such reconstruction works by the following scheme: each column of a table is being compared with all primary keys of other tables, and if the column name and data type match one of the primary keys, a reference between the source column and the key column will be created. This option is available for modification only if Build references is checked.

Tables in a Diagram row

This option determines how reversed tables will be disposed on the diagram. Reversed tables will be placed on the diagram in rows. They will be displayed at an equal distance from each other. This option determines how many tables maximum there will be in one row.

5. In the **Selection** tab of the **Universal Reverse Engineering** tool you can choose the tables you want to reverse engineer.



Click on the checkbox near the table to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all tables in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

7. The reversed database objects will be placed in your diagram.

See also:

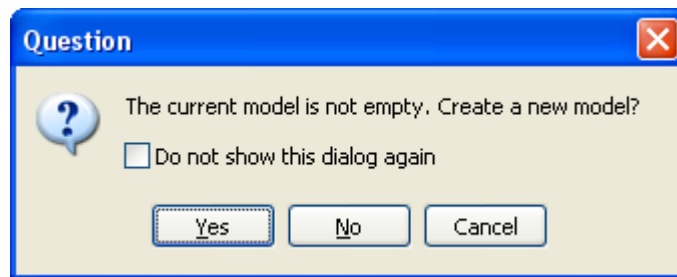
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Import from Access Database](#) | [SQL Reverse Engineering](#)

11.4. SQL Reverse Engineering

You can reverse engineer an SQL script. This means that you can extract tables, attributes, relationships, indexes and other objects. It is possible with the SQL **Reverse Engineering tool**, which parses an SQL Script using internal parser fully compatible with PostgreSQL standards.

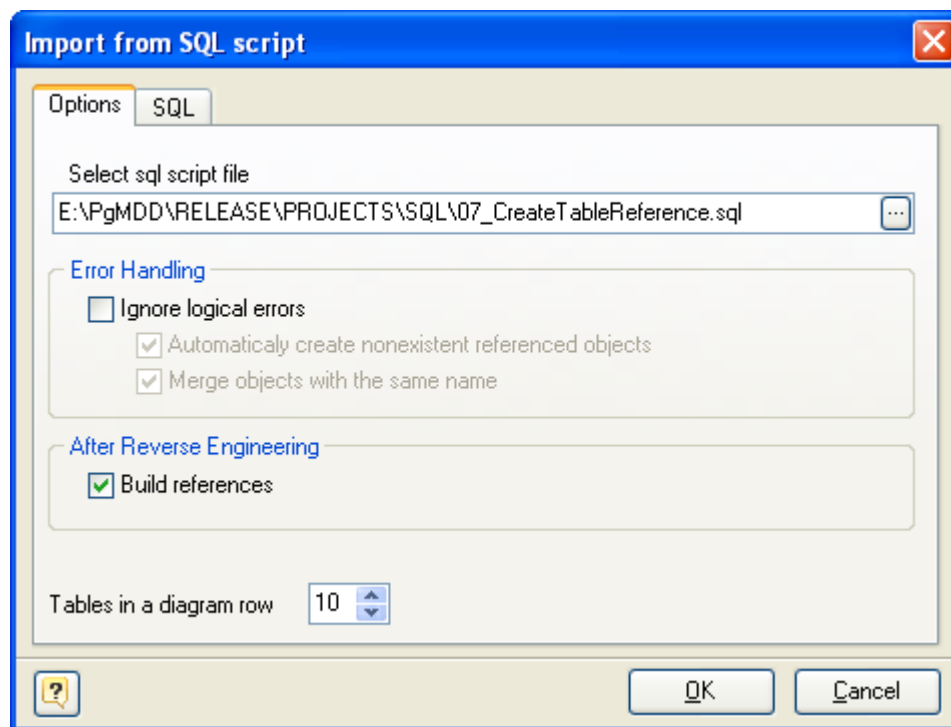
To reverse engineer a script:

1. Select **File | Reverse Engineer | SQL Script...** menu item.
2. If the currently opened diagram already contains some objects, the following dialog box will appear:



It requests you to create a new diagram to place the reversed objects into it, or use the currently opened diagram. Click **Yes** to create a new diagram, **No** to use the currently opened diagram, or **Cancel** to cancel reverse engineering. Select **Do not show this dialog again** to disable future notifications.

3. The **SQL Reverse Engineering** dialog will be shown. First, you should provide options needed for desired behavior of the parser in the **Options** tab.



Click on the ... button to call the standard system **Open File** dialog window.

You can set the following SQL reverse engineering options:

Ignore logical errors

This will cause parser to ignore non syntax errors, e.g. missing referenced objects, name duplicates etc. However, use this option carefully. This may bring some troubles keeping in mind, that the model will not be well-formed in this case.

Automatically create nonexistent referenced objects

Designer will take a try to create necessary missing objects referenced by other ones.

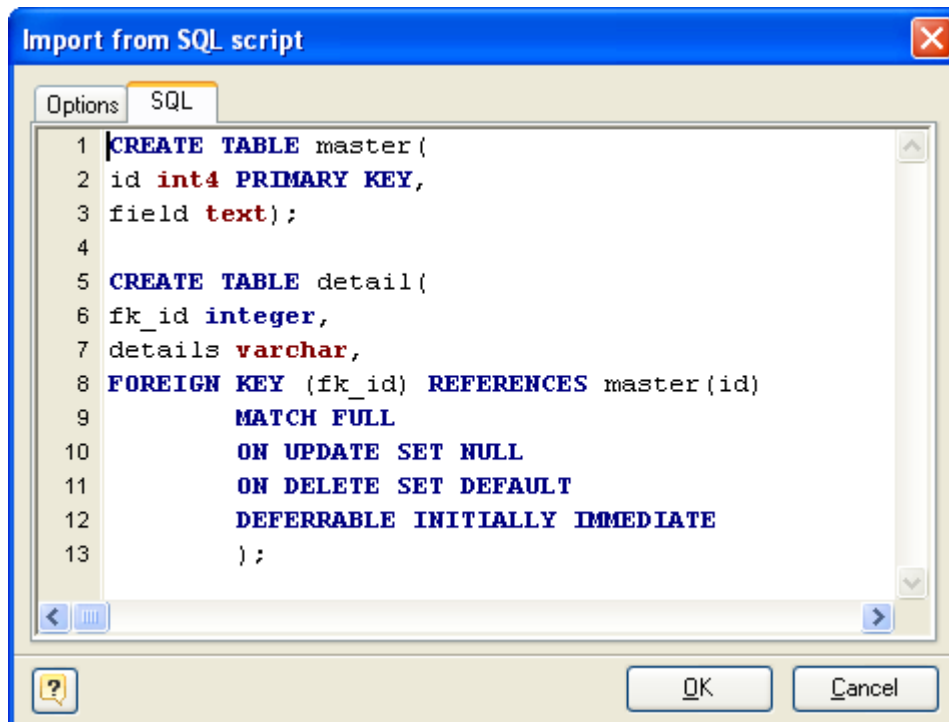
Merge objects with the same name

Objects with the same name will be merged in case this option is checked. The last object in the script has an advantage if some properties can not be merged, e.g. TABLESPACE property of an index or a table and so on.

Build references

This option enables extracting foreign keys from the script and creating corresponding references in your diagram.

4. On the **SQL** tab you can manually edit script loaded from file.



5. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.
6. The reversed database objects will be placed in your diagram.

See also:

Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Import from Access Database](#) | [Universal Reverse Engineering](#)

12. Reports

Database Designer for PostgreSQL can generate comprehensive printable reports for you. This gives you the possibility to get a hard copy of the report by printing it out. After that you get the opportunity to have it approved by your colleagues and managers. Generated reports contain the information about all tables, their indexes, columns and references.

See also:

Reports: [Create a Report](#)

12.1. Create a Report

You can generate an HTML report, which will contain detailed information about the objects placed in the diagram, and diagram picture.

To generate an HTML report, follow these steps:

1. Select **Tools | Create Report** to call the **Create HTML Report** tool.
2. The report generation dialog window will appear.

Please examine the dialog window controls:

File name

Enter the name for the file, in which you want to store the contents of the generated report.

Report style

Select the style of the report from this drop down menu. A style applies graphical environment to the report, such as font color, background, etc.

Include the diagram in the Report

Check this option to insert diagram image into the report.

Show report on complete

Click on this option to open an HTML report in a default browser window after the generation.

In the **Selection** tab you can choose the objects you want to export.

There are several subtabs: **Tables**, **Views** and **Stores Routines**. Each of them allows you to select appropriate database objects. Click on the checkbox near the object to enable its export. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all objects in the list.

Click **OK** to generate the report.

See also:

Reports: [Reports](#)

13. Printing a Diagram


Printing a database diagram gives you a picture of your database structure to refer to or distribute.

Before you start printing, arrange the objects in the database diagram to your satisfaction. You can change the shape, and position of the objects in the diagram without affecting their definitions in the database.

To change the layout of the diagram, move, size, and shape the objects. For example, you can use the mouse to move tables, or use the [Auto Layout Diagram](#) command to automatically reposition the objects.

You can change physical parameters of your paper, on which diagram will be printed out, specify margins, page headers and footers information using [Page Setup](#) dialog.

Before printing out a diagram, you can see how it will look paper, use the [Print Preview](#) tool for that.

To print out a diagram, select the **File | Print** menu item or press **Ctrl-P**. You also can call the **Print Setup** dialog by clicking on the **Print** () icon on the Standard toolbar.

The **Print Setup** dialog is displayed. In the **Print Setup** dialog select the printer you want to print on, and then click the **OK** button to print a diagram.

See also:

Printing: [Page Setup](#) | [Print Preview](#)

Diagram: [Auto Layout Diagram](#)

13.1. Page Setup

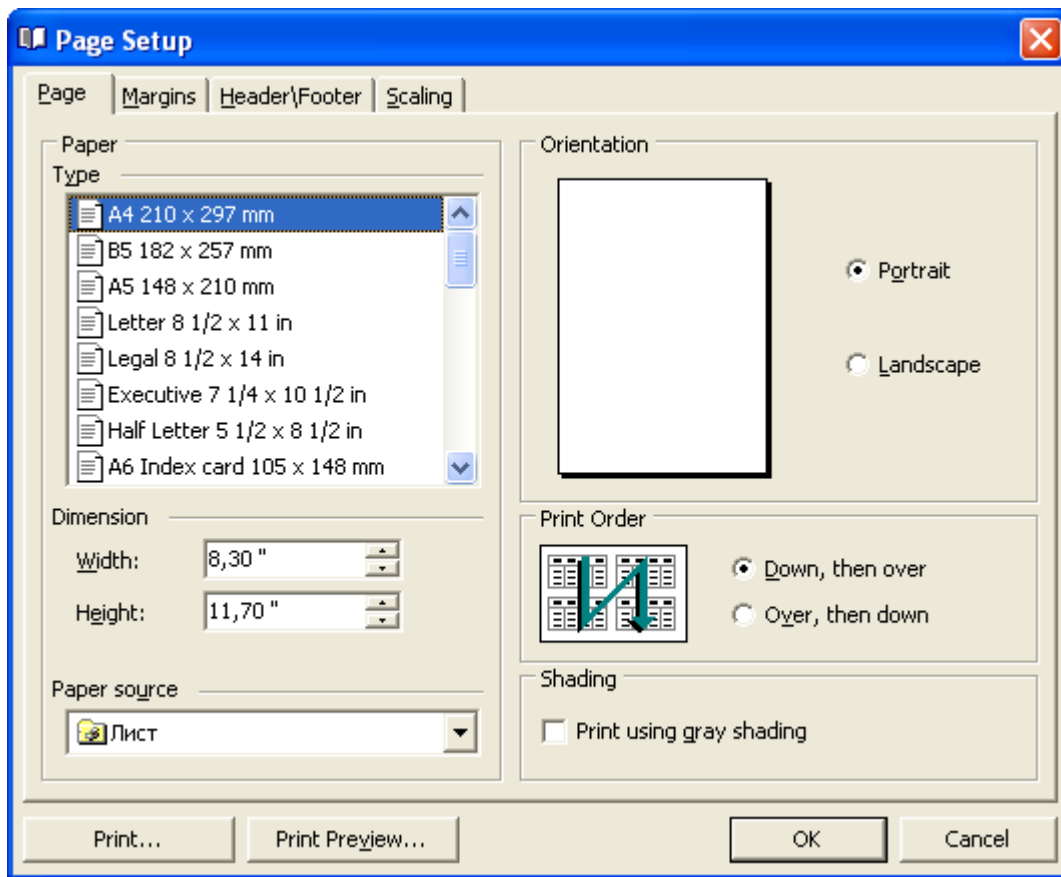
The **Page Setup** dialog window allows you to specify printing parameters, select the paper size and page orientation.

To call the **Page Setup** dialog window, select the **File | Page Setup** menu item.

The **Page Setup** dialog window consists of several tabs, each of which is described in detail below.

Page

This tab allows you to specify page parameters.

**Paper Type**

Allows you to choose one of the predefined paper types.

Dimension Width & Height

Allows you to specify non-standard dimensions for output.

Paper Source

Available paper sources (printer specific).

Orientation Portrait

Vertical page orientation.

Orientation Landscape

Horizontal page orientation.

Print Order

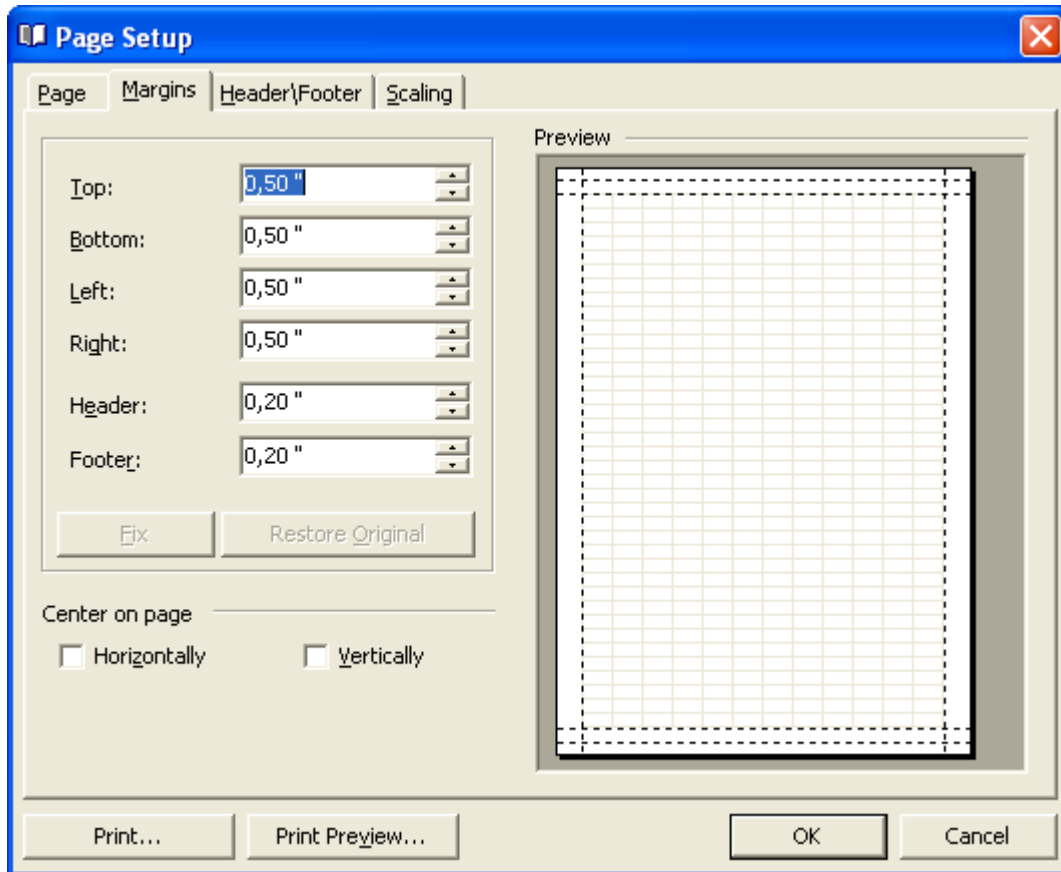
Specifies the order in which pages will be printed.

Shading

Print using gray shading.

Margins

This tab allows you to set the margins and distances.



Top

Adjusts the amount of space that appears between the top of the page and the top of the header.

Header

Sets the height of the header.

Left

Adjusts the whitespace that appears between the left edge of the page and the left edge of the diagram.

Right

Adjusts the whitespace that appears between the right edge of the page and the right edge of the diagram.

Bottom

Adjusts the whitespace that appears between the bottom of the page and the bottom of the footer.

Footer

Sets the height of the footer.

Horizontally

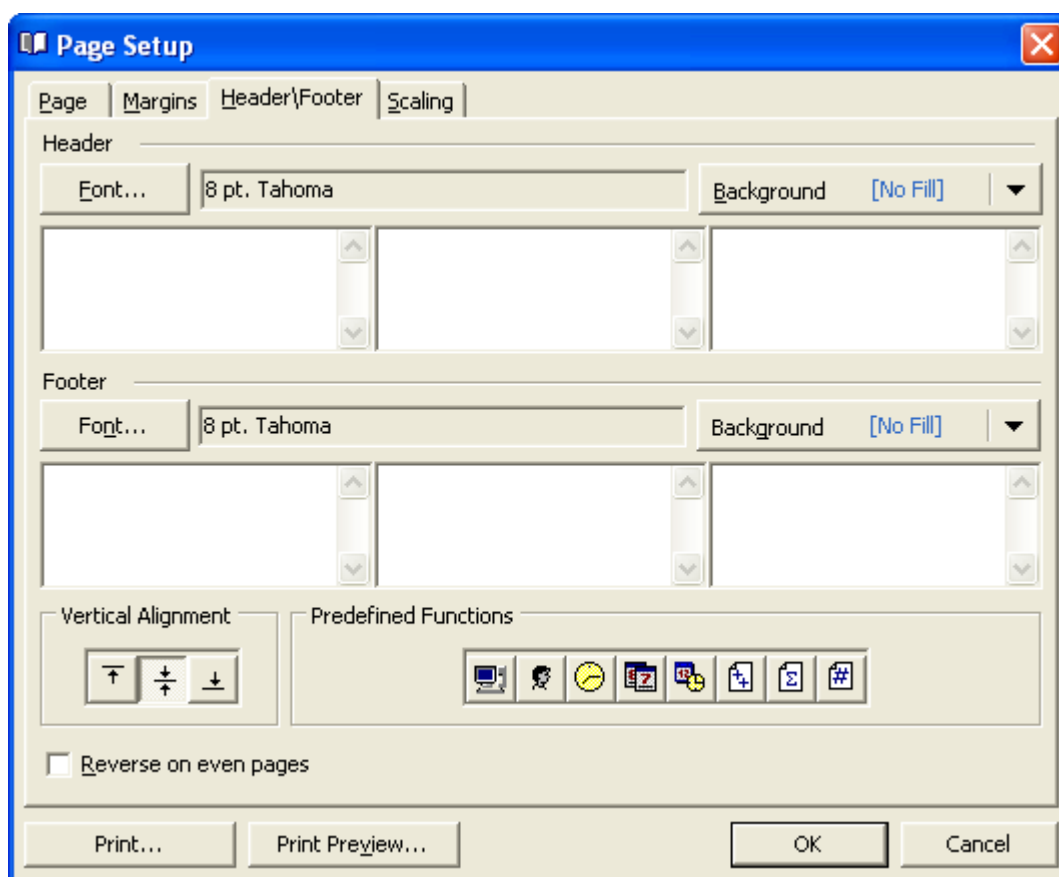
Enable this option to align the content of the page to the center horizontally.

Vertically

Enable this option to align the content of the page to the center vertically.

Header/Footer

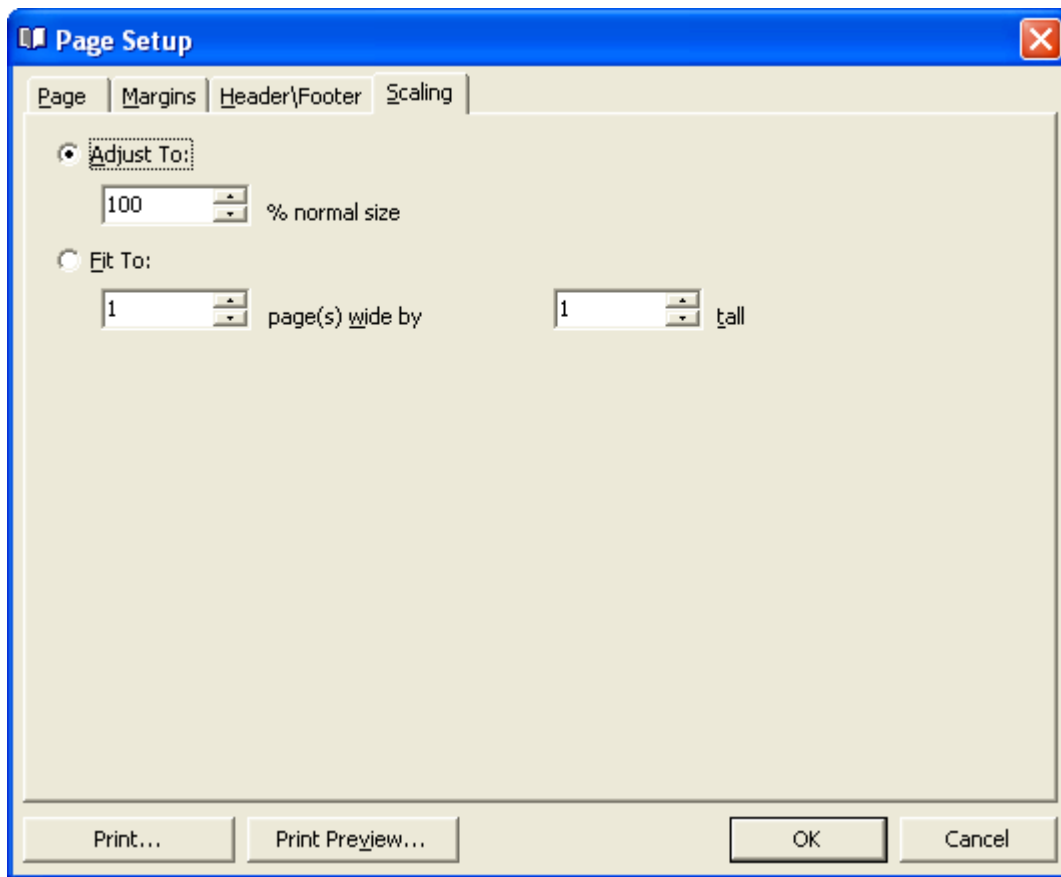
This tab allows you to specify the content of the header and footer, which will be displayed on each printed page.



You can create the content for each header and footer using a list of predefined functions.

Scaling

This tab allows you to specify scale factors for printing.

**Adjust To:**

Set this value to print source page with necessary percentage.

Fit To:

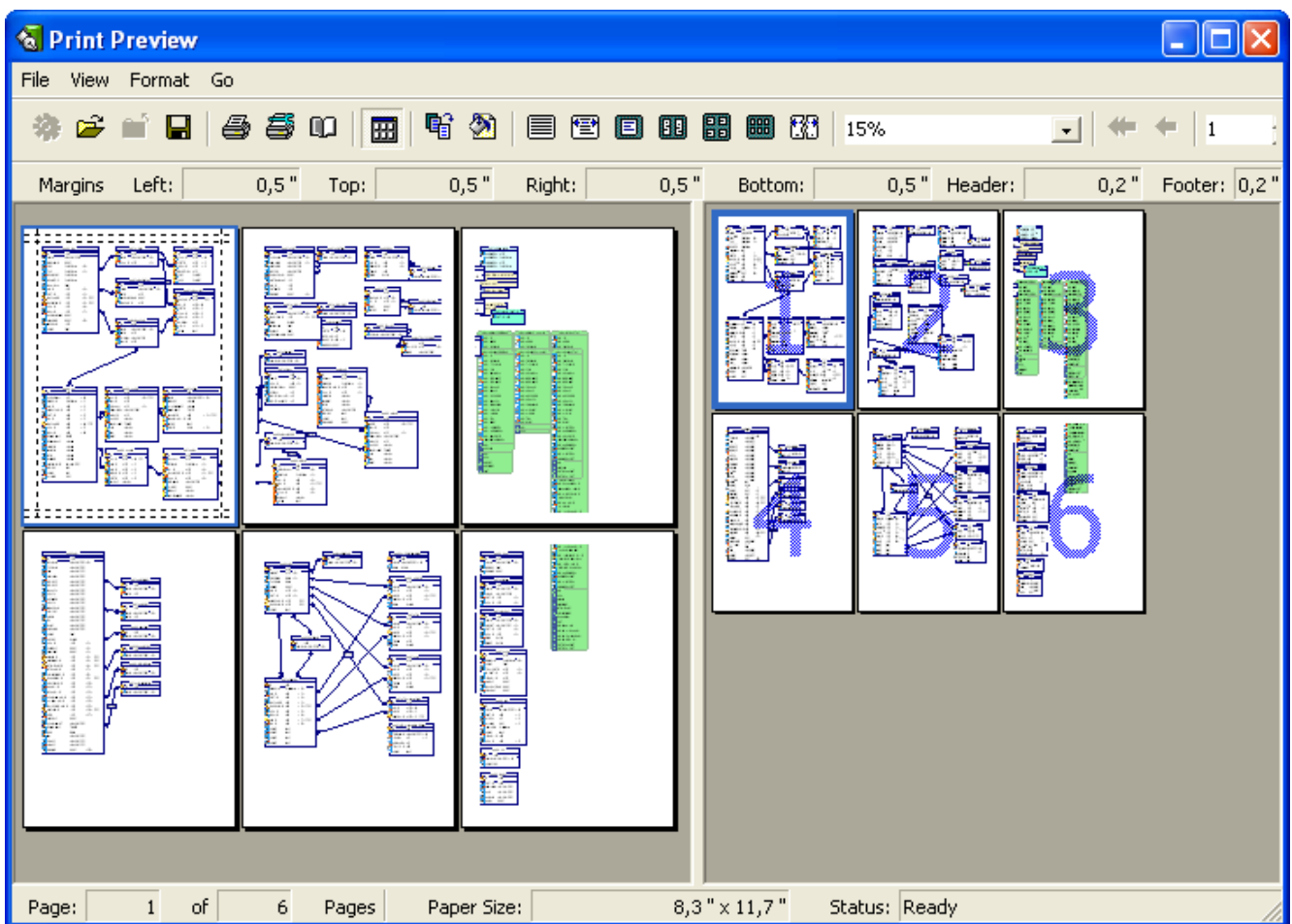
Set these values to print several source pages to one physical paper page.

See also:

Printing: [Printing a Diagram](#) | [Print Preview](#)

13.2. Print Preview

It is often necessary to preview a diagram before printing it out. It is possible with the Print Preview window. To call it, use the **Print | Print Preview** menu item. The following picture demonstrates how the Print Preview window looks like:

**See also:**

Printing: [Page Setup](#)

14. How to...

How to ...

[... Connect to a Database](#)

[... Create a Domain](#)

[... Create a New Diagram](#)

[... Create a Reference and Foreign Key](#)

[... Create a Table](#)

[... Disconnect from a Database](#)

[... Edit Columns](#)

[... Edit Table Index](#)

[... Execute an SQL Script](#)

[... Export a Diagram to Graphics](#)
[... Find Errors in a Diagram](#)
[... Find Objects](#)
[... Generate a Database](#)
[... Import from a Microsoft Access Database](#)
[... Import from a PostgreSQL Database](#)
[... Import from other Databases](#)
[... Insert a Comment](#)
[... Merge Diagrams](#)
[... Modify a Database](#)
[... Modify Multiple Tables](#)
[... Print a Diagram](#)
[... Read a Diagram](#)
[... View an SQL Table Definition](#)

14.1. How to Connect to a Database

To start working with a database using [Generate Database](#), [Database Modification](#), [SQL Execute](#) and [Reverse Engineering functions](#), you have to connect to it first. To do so, please follow these steps:

1. Select the **Database | Connect** file menu item, or press **Ctrl-Shift-N**. The [Database Connection Manager](#) will appear.
2. Select one of the defined database profiles in the list or add a new one.
3. Click **OK** to establish connection to the database.

See also:

[Database Modification](#)

Database Generation: [Database Generation](#)

Database Tools: [SQL Executor](#) | [Database Connection Manager](#)


Reverse Engineering and Import: [Reverse Engineering PostgreSQL Database](#)

14.2. How to Disconnect from a Database

To break the current connection from the database, use the **Database | Disconnect** menu item.

14.3. How to Create a New Diagram


The first step to use **Database Designer for PostgreSQL** is to create a new diagram.

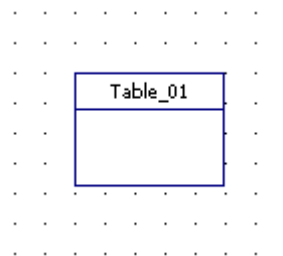
To create a new diagram select the **File | New** menu item or press **Ctrl-N**. You can also use the New diagram button  on the **Standard toolbar**.

A new diagram will have a default name like "Noname1". You can change the name of your diagram in the [Diagram Properties](#) dialog window.

14.4. How to Create a Table

To create a new table in a diagram:

1. Click on the **Table** () icon on the **Palette** toolbar. Your mouse cursor will change its appearance. Click on the diagram area to create a new table. Or right click on the necessary schema in the **Object Tree View** and choose **Create Table** item. Or right click on the diagram area and choose **Create Object -> Table**. An empty table will appear in the diagram:




2. (optional) Double click on the new table symbol in the diagram to display the [Table Editor](#) dialog window.
3. (optional) Enter the table name in the **Table name** field.
4. (optional) Click **OK** to save the changes.


See also:

Diagram Objects: [Table Editor](#)

14.5. How to Create a Reference and Foreign Key

To create a reference between tables:

1. Click the  icon on the **Palette** toolbar. Your mouse cursor will change its appearance.
2. Click on the table (a referencing table) that will have a foreign key.
3. Then click on the second table (referenced table) whose constraint (e.g. Primary key) will be referenced by the new foreign key.
4. The **Join** tab of the [Reference Editor](#) will be shown. You can choose columns of a referenced and referencing table participated in the reference. You can choose *<Auto Create Column>* on the list of referencing table columns to **Database Designer for PostgreSQL** create a new column in the referencing table that will have a foreign key set on it. The properties of auto column will be copied from the respective column of the referenced table.

To create a self reference (that links columns at the same table), click the  icon on the **Palette** toolbar. Then click the same table two times. To create **N:M** references, please refer to [Creating a Many-to-Many Reference](#).

Reference Creation in Details

On reference creation, **Database Designer for PostgreSQL** performs the following actions:

1. Creates new column(s) in the referencing table, their parameters (name, data type) will be copied from the primary key constraint of the referenced table. If the referencing table already has column(s) analogous to the primary key(s) of the referenced table, this column(s) will be used as foreign-key column(s).
2. If there is no primary key or unique constraints in the referenced table, a standard primary key column will be created in the referenced table.
3. Creates foreign key constraint in the referencing table that refers to the referenced table primary key(s).

See the diagram [database options](#) to find out more about disabling some of above actions.

See the [Notation](#) topic to find out more about reference symbol on the diagram.

See also:

Diagram: [Notation](#) | [Database Options](#) | [Creating a Many-to-Many Reference](#)

Diagram Objects: [Reference Editor](#)

14.6. How to Check a Diagram

The **Check Diagram** tool of the **Database Designer for PostgreSQL** allows you to check your database diagram for most typical errors and defects. The result of the check goes in a well structured form, using which you can easily bring your diagram to correspondence with the common standards of database modeling.

To open the **Check Diagram** dialog press **F4** or select the **Diagram | Check Diagram** menu item.

Use the **Select diagram** drop-down list to select the diagram from the list of currently opened diagrams.

The tree list below allows you to select what warnings and errors should be taken into account during the check. All warnings and errors are divided into categories, which correspond to the diagram objects. Remove selection from the warning/error or from the whole category to exclude it from the check.

These are the descriptions for all available warnings and errors:

Table

Error "Table Name Uniqueness"

Check the diagram for the uniqueness of each table name within a schema;

Warning "Table Name Max Length"

PostgreSQL allows only 63 characters in table names and cuts names if they are longer than this;

Warning "Column Definition"

Check if each table within the diagram owns at least one column;

Warning "Index Definition"

Check if each table within the diagram owns at least one index;

Warning "Primary Key Definition"

Check if a primary key is defined for each table within the diagram;

Warning "Reference Definition"

Check if each table within the diagram is linked with other tables;

Error "Auto-increment columns"

Check if each table within the diagram has no more than one auto-increment column, as otherwise PostgreSQL will not allow such table to be created;

Table Columns

Error "Column Name Uniqueness"

Check diagram tables for the uniqueness of each column name within the table;

Warning "Column Name Max Length"

PostgreSQL allows only 63 characters in column names and cuts names if they are longer than this;

Warning "Auto-increment Column Definition"

Check if each auto-increment column within a table is a part of a primary key;

Table Indexes

Error "Index Name Uniqueness"

Check diagram tables for the uniqueness of each index name within the table;

Warning "Index Name Max Length"

PostgreSQL allows only 63 characters in index names and cuts names if they are longer than this;

Warning "Duplicate Index Column"

Check if each table column is indexed only once;

References**Error "Reference Column Data Types"**

Check whether the linked columns are of the same data type for each diagram reference;

Domains**Error "Domain Name Uniqueness"**

Check diagram for the uniqueness of each domain name within a schema.

Stored Routine**Error "Stored Routine Name Uniqueness"**

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "Stored Routine Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Views**Error "View Name Uniqueness"**

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "View Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Error "View was created on a not existing table"

Check existence of tables, which are used in the view;

Error "View was created on a not existing column"

Check existence of table columns, which are used in the view.

After you click **OK** the check process will be displayed within the **Output** window and the result of the

check will be displayed within the **Result** window in the same categorized view as described above.

Double-click on a warning or an error in the list opens the editor window for the appropriate object ([Table Editor](#), [Index Editor](#), or [Domain Manager](#)).

See also:

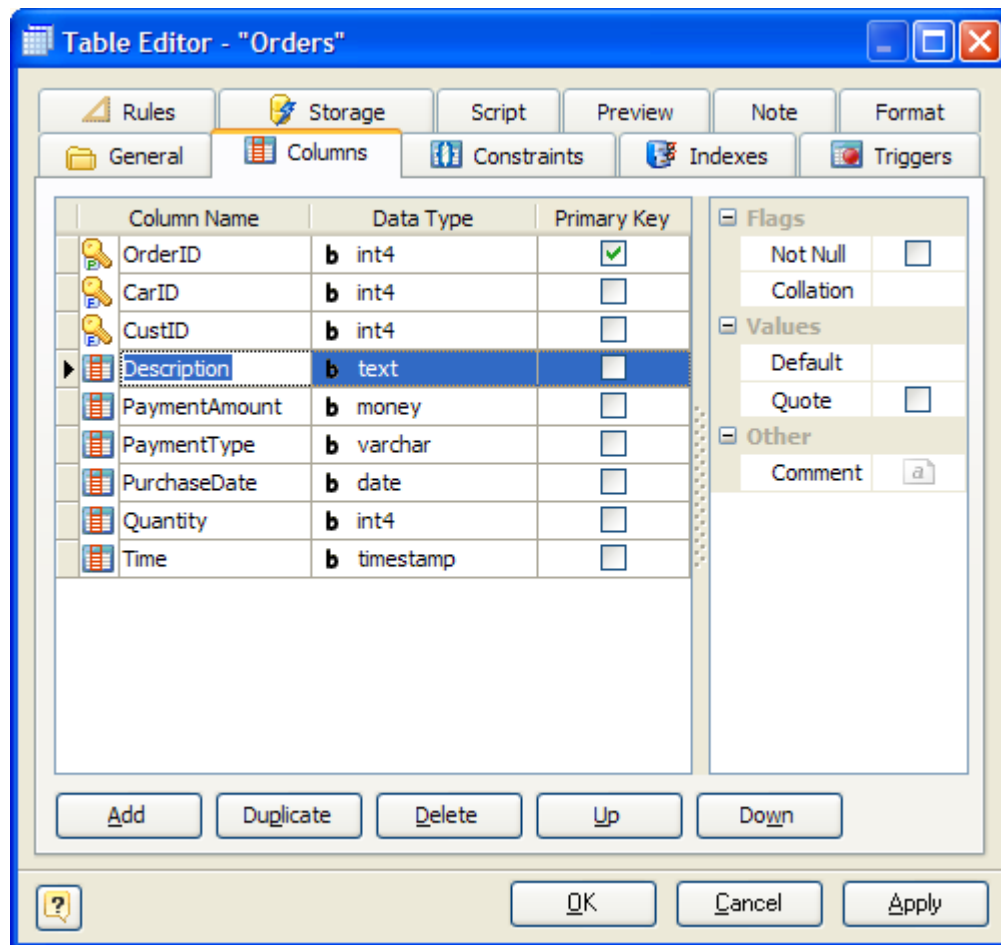
Diagram Objects: [Table Editor](#) | [Column Editor](#) | [Reference Editor](#) | [Index Editor](#)

14.7. How to Edit Columns

The **Column Editor** is placed within the [Table Editor](#) dialog. It allows you to modify the list of table columns as well as column properties. Click the **Columns** tab of the **Table Editor** to manage table columns.

The **Column Editor** consists of the following areas:

- **Column List**
- **Properties Pane**
- **Button Pane**



Column List

The column list displays all the columns in the table and allows you to modify the following column properties:

- **Column name** - the name of the column, which must be unique within the table;
- **Data type** - the type of the column, which specifies data to store in the column;
- **Primary key** - specify this option to include the field into the table primary key;

Properties Pane

The properties pane allows you to define the advanced properties of the column, selected in the **Column List**. The appearance of this pane changes according to the data type of the column. These properties are:

- **Length** - this attribute defines the maximum allowed length of the stored values; it applies to

all integer, decimal, and string types;

- **Decimals** - this attribute defines the number of digits, which follow the decimal point;
- **Not null** - this option indicates that the stored column value cannot be NULL;
- **Collation** option assigns a collation to the column (which must be of a collatable data type). If not specified, the column data type's default collation is used.
- **Autoinc** - this attribute makes the column value auto increment, i.e. each new value is set automatically according to the previous value; it applies to all integer values. Depending on the choice, this may generate a sequence using *SERIAL* keyword, or create *IDENTITY* column);
- **Default** - this attribute defines the default value, which the column accepts if no other is specified.
- **Quote** - this attribute defines the default value to be quoted or not.
- **Comment** - an arbitrary description for the column.

Buttons Pane

The buttons under the list of columns allow you to perform the following actions:

- **Add** - add a new column with the default properties to the end of the list;
- **Duplicate** - add a new column with the same properties as the selected column to the end of the list;
- **Delete** - remove the selected column from the list;
- **Up/Down** - move the selected column along the list.

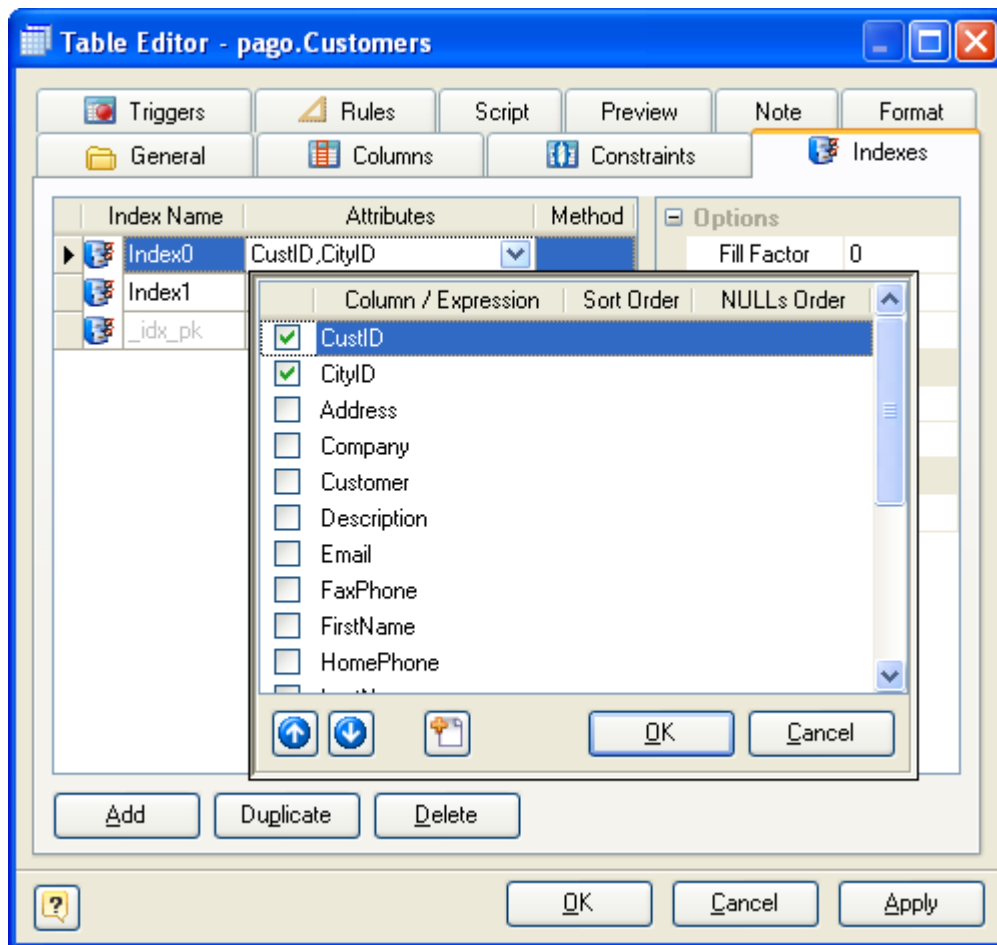
See also:

Diagram Objects: [Domains](#) | [Table Editor](#) | [Indexes](#) | [Constraints](#) | [Triggers](#) | [Rules](#)

14.8. How to Edit Table Index

The **Index Editor** is placed within the [Table Editor](#) dialog. It allows you to modify the list of table indexes as well as index properties.

The main element of the editor is the index list, which displays all indexes available within the table. The columns of the list allow you to modify the properties of the selected index.



These properties are:

Index Name

The name of the index, which must be unique within the table.

Attributes

The names of a columns of the table or expressions based on one or more columns of the table. The expression must not be written with surrounding parentheses, because Designer will add them automatically.

Method

The name of the method to be used for the index. Choices are btree, hash, rtree, and gist. The default method is btree.

Sort Order

Specifies index attributes sort order.

💡 Since an ordered index can be scanned either forward or backward, it is not normally useful to create a single-column DESC index — that sort ordering is already available with a regular index. The value of these options is that multicolumn indexes can be created that match the sort ordering requested by a mixed-ordering query, such as `SELECT ... ORDER BY x ASC, y DESC`.

NULLs Order

Specifies that nulls sort before or after non-nulls.

💡 The NULLs Order options are useful if you need to support “nulls sort low” behavior, rather than the default “nulls sort high”, in queries that depend on indexes to avoid sorting steps.

Predicate

The constraint expression for a partial index.

Tablespace

The tablespace in which to create the index. If not specified, default is used, or the database's default tablespace if server's parameter `default_tablespace` is an empty string.

Fast Update

This setting controls usage of the fast update technique for GIN indexes. Updating a GIN index tends to be slow because of the intrinsic nature of inverted indexes: inserting or updating one heap row can cause many inserts into the index (one for each key extracted from the indexed value). As of PostgreSQL 8.4, GIN is capable of postponing much of this work by inserting new tuples into a temporary, unsorted list of pending entries. See PostgreSQL manual for details.

Fillfactor

The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages. For B-trees, leaf pages are filled to this percentage during initial index build, and also when extending the index at the right (largest key values). If pages subsequently become completely full, they will be split, leading to gradual degradation in the index's efficiency. B-trees use a default fillfactor of 90, but any value from 10 to 100 can be selected. If the table is static then fillfactor 100 is best to minimize the index's physical size, but for heavily updated tables a smaller fillfactor is better to minimize the need for page splits. The other index methods use fillfactor in different but roughly analogous ways; the default fillfactor varies between methods.

Unique

Defines the UNIQUE constraint for the selected columns. I.e. the combination of the included field values must be unique within the table;

System

An indicator that shows if the index is system and can't be modified by a user.

Comment

Specifies comment for index object.

The buttons under the list of indexes allows you to perform the following actions:

- **Add** - add a new index with the default properties to the end of the list;
- **Duplicate** - add a new index with the same properties as the selected index to the end of the list;
- **Delete** - remove the selected index from the list;
- **Up/Down** - move the selected index along the list.

See also:

Diagram Objects: [Index Manager](#)

14.9. How to Generate a Database

The **Database Generation** tool can generate SQL script, that represents the diagram you developed and executes it on the database server.

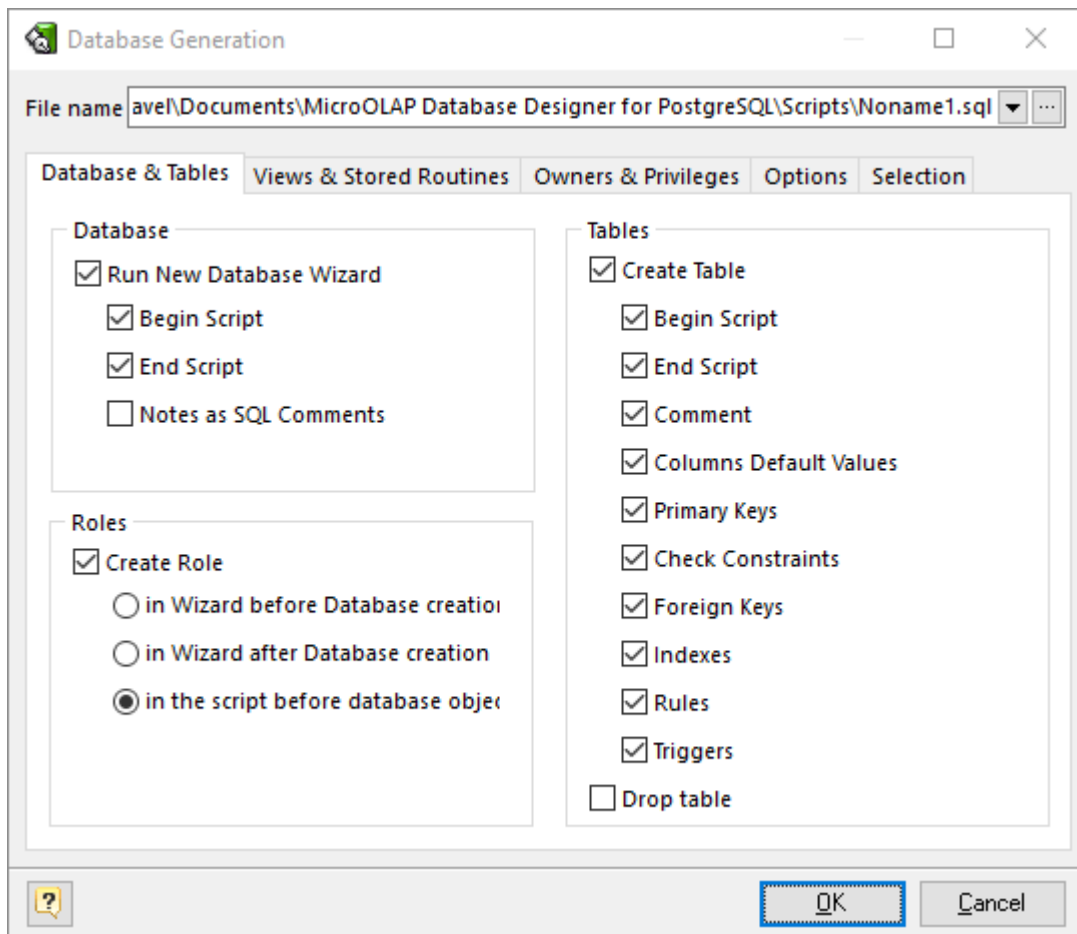
You can generate database in two ways:

- Directly execute a script on a PostgreSQL server. Please examine [Connect to a Database](#) section to explore the database connection process;
- Generate a script to be executed on PostgreSQL server at a later time.

In both cases, the database generation commands are saved in a script file. You must always provide path to the script file.

To generate database, start the **Database Generation** tool by selecting the **Database | Generate Database** menu item or pressing **Ctrl-G**. **Database Generation** tool consist of five tabs, which contain SQL generation options. Let's explore them. The following pictures demonstrate **Database Generation** tool interface.

Database Generation



File name

This field allows you to set file, in which generated SQL statements will be stored. Click on the ... button near the field to browse to file on the file system.

Database Group

Run 'Create New Database Wizard'

Mark this checkbox if you want to create a new physical PostgreSQL database before generating the database structure presented in the diagram.

Begin Script

This option enables inserting the begin script before the CREATE DATABASE statement.

End Script

This option enables inserting the end script after the CREATE DATABASE statement.

Notes as SQL Comments

This option enables using notes of database as comments in SQL script.

Role Group

Create Role

This option enables generation of model roles.

in Wizard before Database creation

This option enables executing the CREATE ROLE statements before the CREATE DATABASE statement right in **New Database Wizard**.

in Wizard after Database creation

This option enables executing the CREATE ROLE statements after the CREATE DATABASE statement right in **New Database Wizard**.

in the script before database objects

This option enables executing the CREATE ROLE statements in the script before tables, views, stored routines etc.

Tables Group

Create Tables

This option enables generation of tables.

Begin Script

This option enables inserting the begin script (it can be set using [Table Editor](#)) before the CREATE TABLE statement.

End Script

This option enables inserting the end script (it can be set using [Table Editor](#)) after the CREATE TABLE statement.

Comment

This option enables inserting of table comments in SQL script.

Columns Default Values

This option enables generation of columns' default values.

Create Primary Keys

This option enables generation of table primary keys.

Create Check Constraints

This option enables generation of check constraints for the tables.

Create Foreign Keys

This option enables generation of foreign keys for the tables.

Create Indexes

This option enables generation of indexes for the tables.

Create Rules

This option enables generation of rules for the tables.

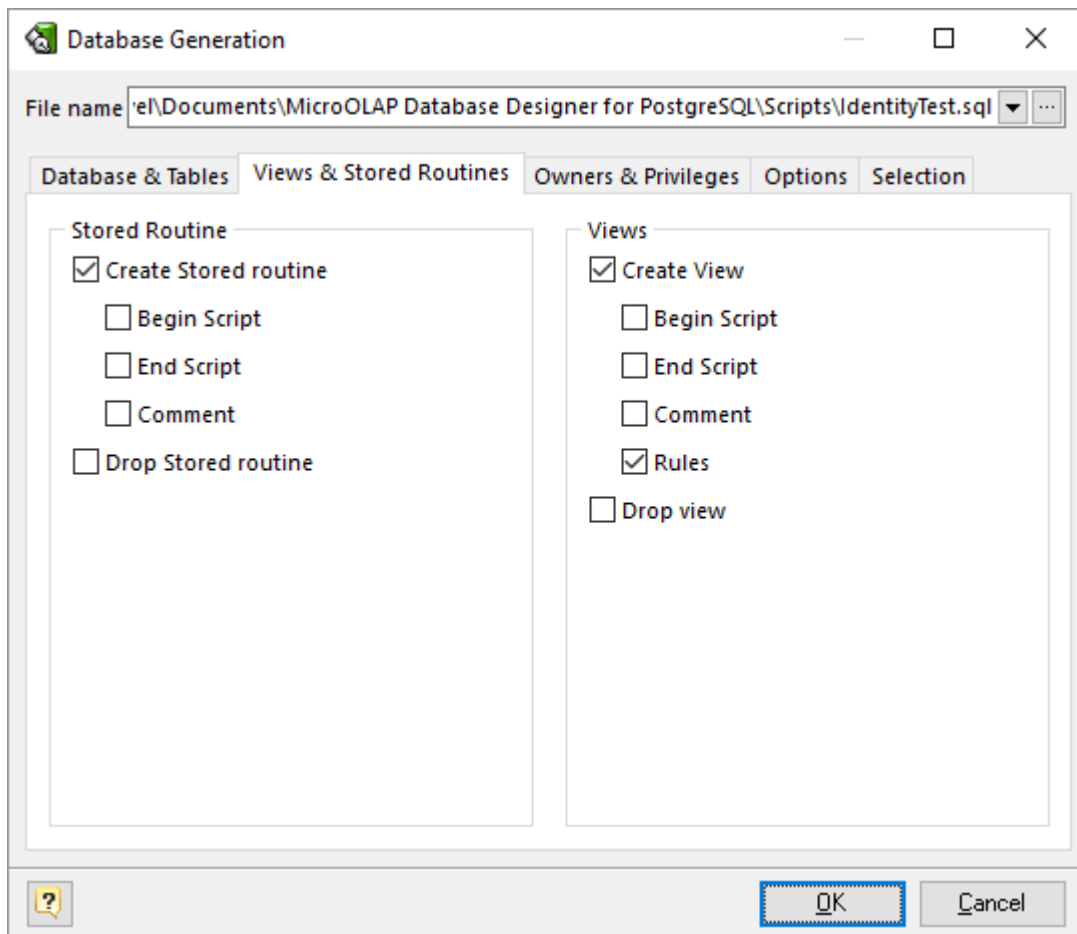
Create Triggers

Generate triggers set on table.

Drop table

This option enables dropping old tables if they were already exist. I.e. enables generating DROP TABLE "<TABLE_NAME>" statement before CREATE TABLE statement.

Views & Stored Routines



Stored Routines Group

Create Stored Routine

This option enables generation of stored routines (i.e. stored functions and procedures).

Begin Script

This option enables inserting the begin script before the SQL generation statement.

End Script

This option enables inserting the end script after the SQL generation statement.

Comments

This option enables showing of stored routines comments in SQL script.

Drop Stored Routine

This option enables dropping old stored routines if it were already exist.

Views Group

Create View

This option enables generation of database views.

Begin Script

This option enables inserting the begin script before the SQL generation statement.

End Script

This option enables inserting the end script after the SQL generation statement.

Comment

This option enables showing of view comments in SQL script.

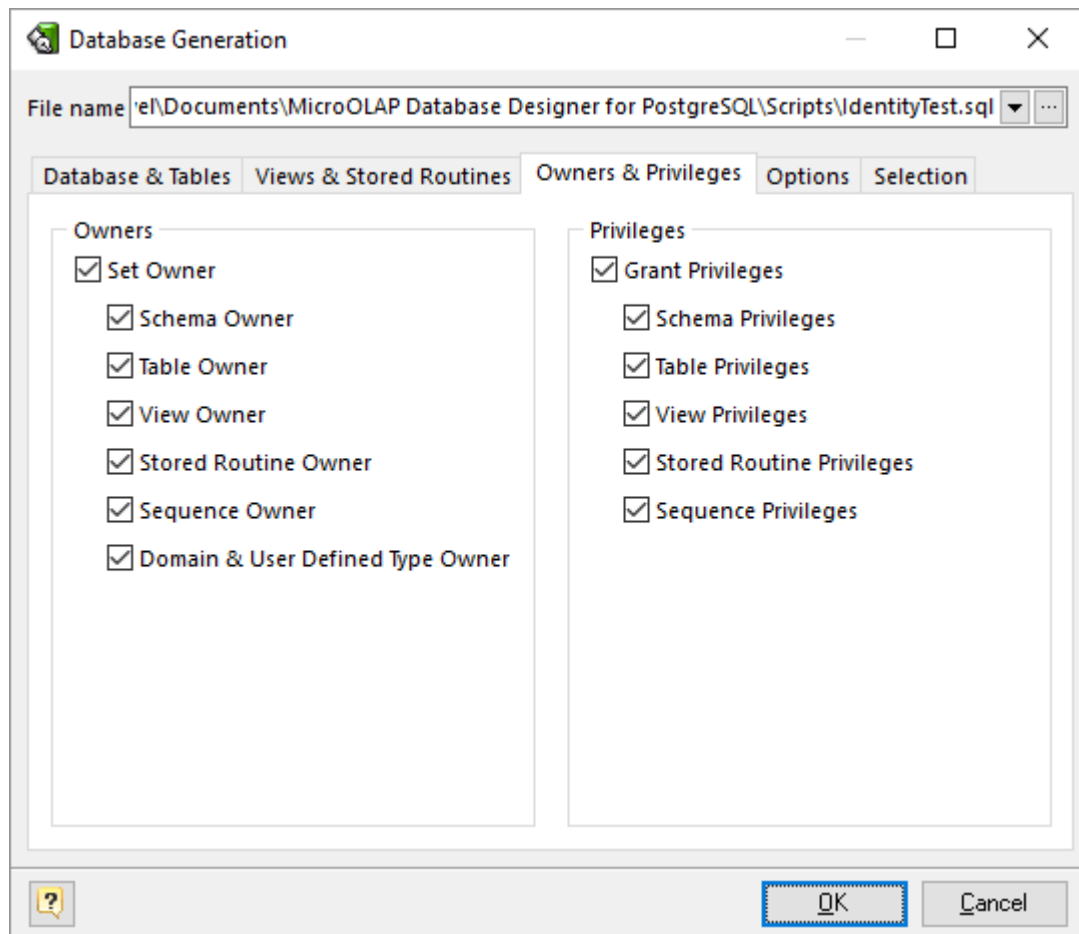
Create Rules

This option enables generation of rules for the views.

Drop Views

This option enables dropping old views if it were already exist.

Owners & Privileges

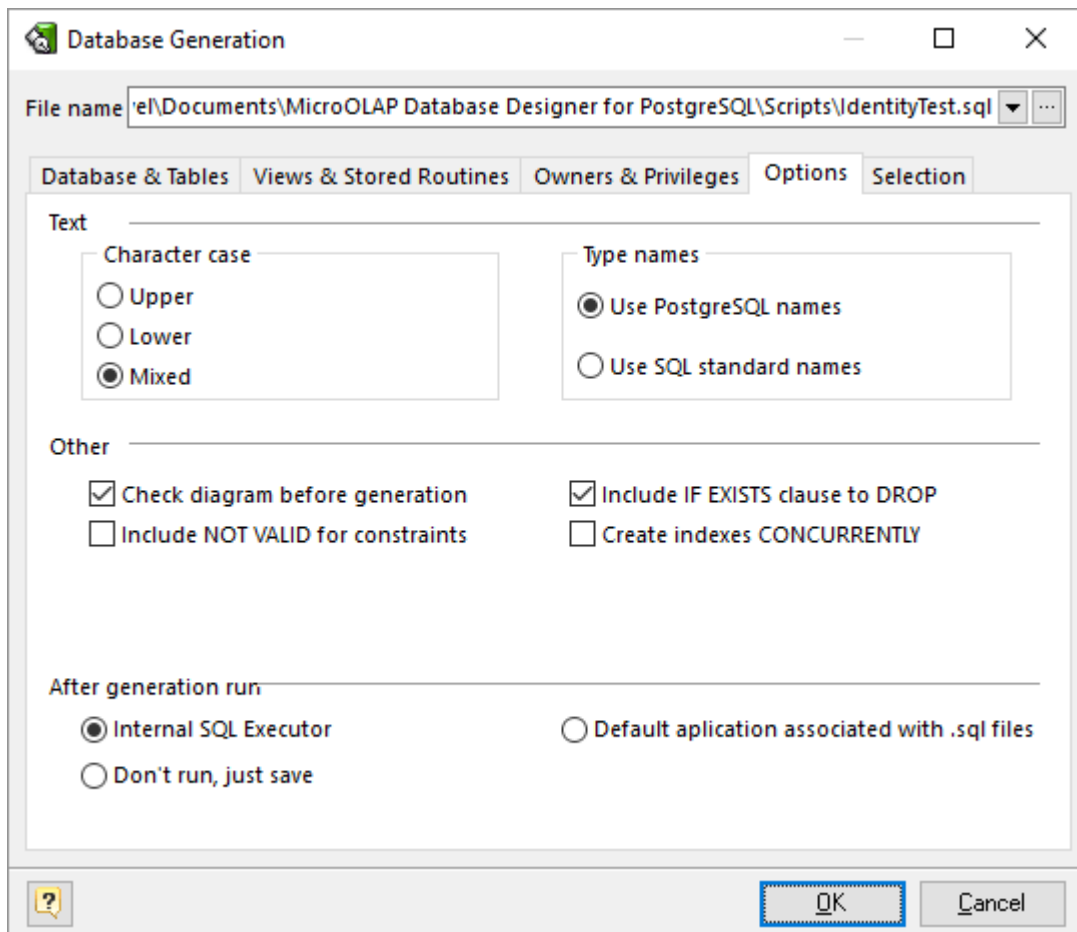
**Owners**

Defines for which objects owner information should be generated.

Privileges

Defines for which objects privileges information should be generated.

Options



This tab allows set generation-related options.

Character case

Defines case of characters, which will be used in generated script. Use Mixed option to leave the characters unmodified.

Type names

Determines what type names will be used in the generated SQL, PostgreSQL (*int2, int4, int8, varchar, char, bool, float8, float4*) or standard ones (*smallint, integer, bigint, character varying, character, boolean, double precision, real*).

Before generation:

Check diagram

Enables checking the diagram before generation.

Drop objects

Include IF EXISTS clause

Do not throw an error if the object does not exist. A notice is issued in this case while executing generated script. Server must be 8.2.x or higher.

After generation, run**Internal SQL Executor**

Send the generated SQL statements into the internal [SQL Executor](#).

Default application associated with SQL files

Send the generated SQL statements into the application associated with SQL files

Don't run, just save

Do nothing with output script, just save it on disk.

Selecting objects to generate

You can select diagram objects you want to generate in SQL script or database. Use the **Selection** tab of the **Database Generation** tool for it.

There are several subtabs: **Tables**, **Stores Routines**, **Views**, **Types & Domains** and **Sequences**. Each of which allows you to select appropriate diagram objects to generate.

To enable particular objects generation, click on the checkbox near it.

The default selection of objects to generate depends on their **Generate** property.

Pay attention to the buttons on the **Selection** tab:

Select ALL - checks on all checkboxes.

Deselect ALL - checks off all checkboxes.

Use graphical selection - checks on checkboxes for objects, depending on [diagram selection](#).

You can change the order of tables in which they will be placed in the generated SQL script. Use the buttons with arrows for this.

Generating, customizing and executing SQL

Click on the **OK** button on the **Database Generation** tool to generate SQL script. The generated SQL script will be stored in the file you have set.

If you have established connection to database, the [SQL Executor](#) with generated SQL statements will appear.

You can easily customize statements for your needs. And then send them to the database server by

clicking on the **Execute SQL** button.

Please, examine [SQL Executor](#) section to know more about it.

See also:

Database Accessing: [SQL Executor](#)

Database Functions: [Database Modification](#)

14.10. How to Modify a Database


Overview

Once you have changed your diagram, it's usually necessary to apply these changes to your database. It's easy to do with the **Database Modification** tool. The **Database Modification** tool can generate SQL script that leads your database to the current state of your diagram.

You can synchronize database in two ways:

- Directly execute a modification script on a PostgreSQL server. Please examine [Connect to a Database](#) section to explore the database connection process;
- Generate a modification script for executing at a PostgreSQL server some time later.

In both cases, the database modification commands are saved in a script file. You must always provide the path to the script file.

 Please note, that database modification usually cause multiple complex statements for database structure modification. It is possible that some of them may not execute correctly due to some physical reason. It's recommended to make a backup of your database before applying structure changes to database.


That's how the Database Modification tool works:

1. Reverse engineers your existing PostgreSQL database
2. Compares the result with your current database diagram
3. Creates a list that contains differences between database objects and diagram objects
4. After analyzing the difference list, creates necessary SQL statements, that modify database structure.

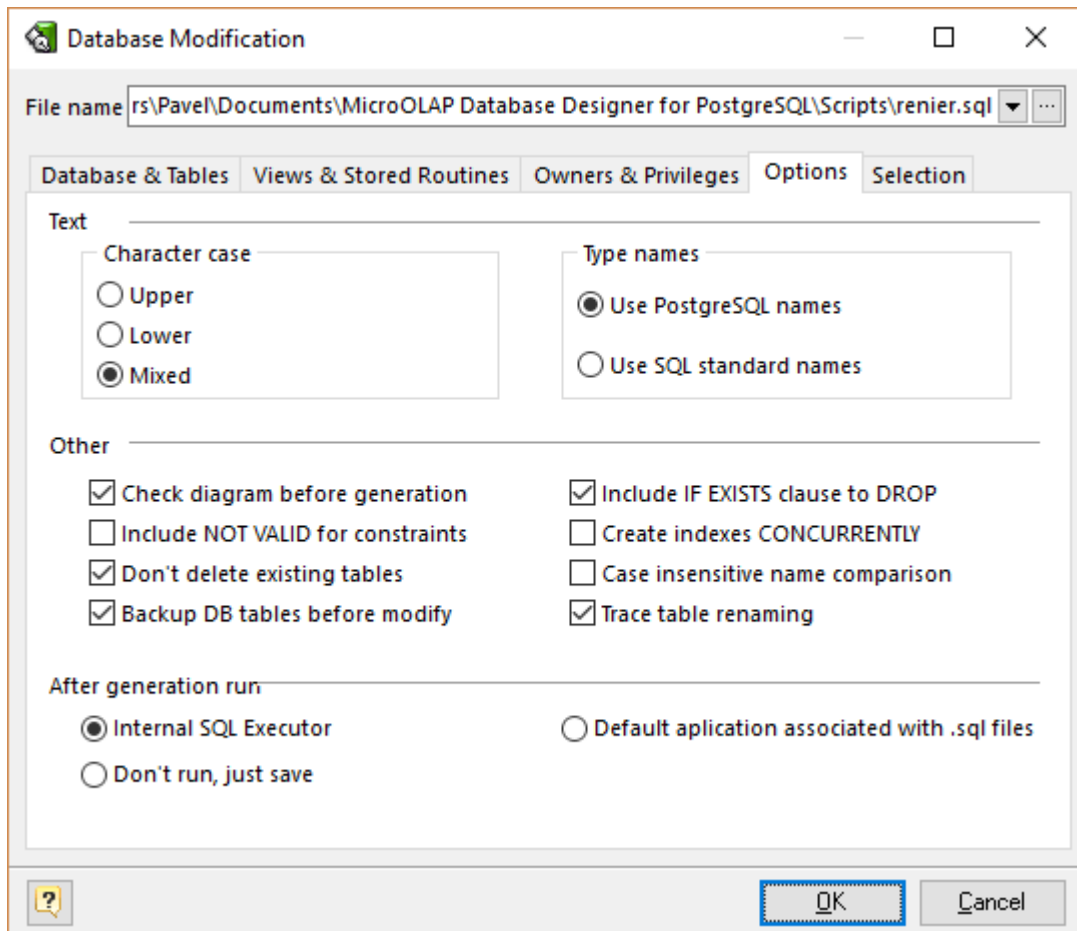
If some table is going to be modified, **Database Designer for PostgreSQL** makes a backup copy of that table, so you can restore data and table structure later on if there will be some errors during the table structure alteration.

Database Modification Tool

To modify your database, start Database Modification tool by selecting the **Database | Modify Database** menu item or pressing Ctrl-M.

 Interface of the **Database Modification** dialog is much the same as the [Database Generation](#) dialog. The only differences are persist on the **Options** tab.

Modification Options



In the Options tab of the Database Modification tool you can set modification options.

Don't delete existing tables

This option disables deleting tables that already exist in the database, but don't exist in your diagram.

Backup DB tables before modify

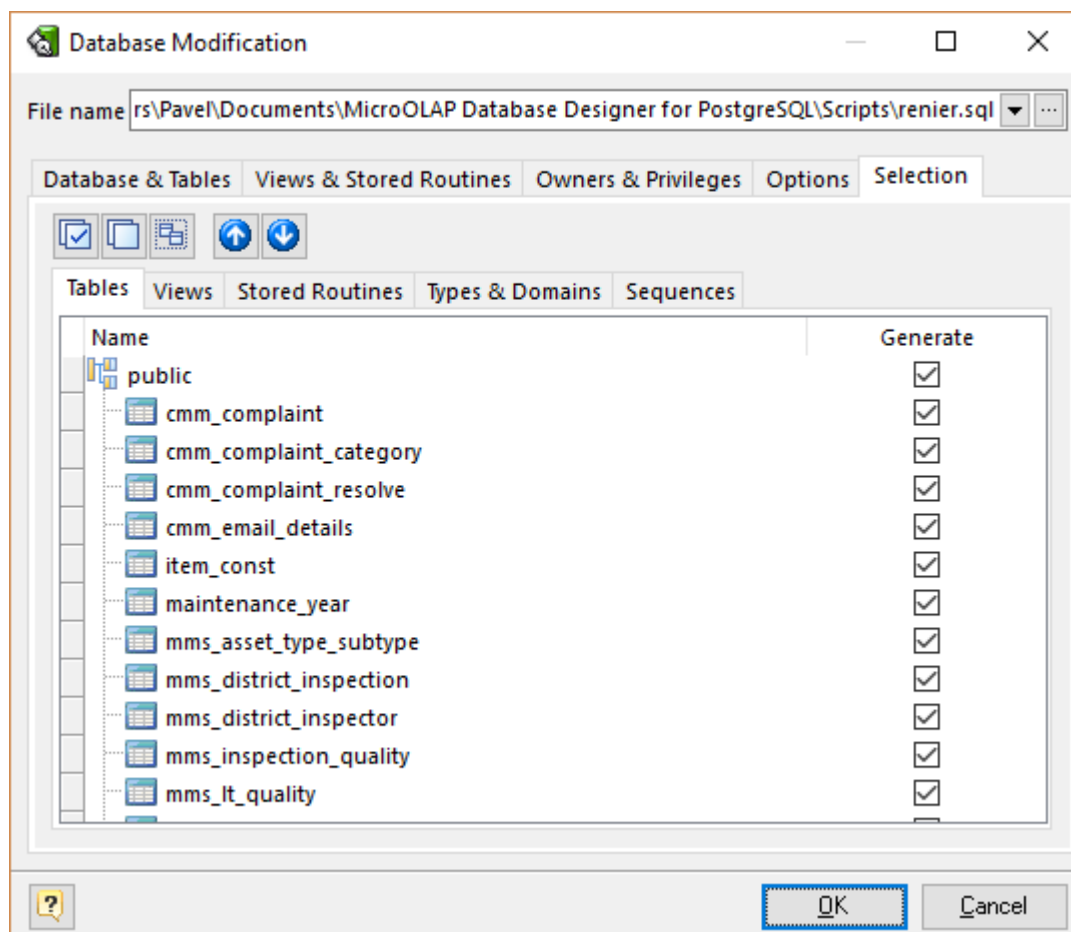
This option enables creating backup of tables to be modified.

Trace table renaming

Try to find tables that has been renamed in the diagram by comparing their structure with the structure of the tables in the database. If a structure of a table in the database is identical to a

structure of a table in the diagram, but such tables have different names, a table in the database is going to be renamed to match the diagram table's name.

Selecting objects to generate



You can select diagram objects which need to be modified in the database. Use the **Selection** tab of the **Database Modification** tool for this.

There are several subtabs: **Tables**, **Views**, **Stores Routines**, **Types & Domains** and **Sequences**. Each of which allows to select appropriate diagram objects to modify.

To enable particular objects generation, click on the checkbox near it.

The default selection of objects to generate depends on their Generate property. Pay attention to the buttons on the Selection tab:

Select ALL - checks on all checkboxes

Deselect ALL - checks off all checkboxes

Use graphical selection - checks on checkboxes for objects, depending on [diagram selection](#).

You can change the order of tables in which they will be placed in the generated SQL script. Use the buttons with arrows for this.

Generating, customizing and executing SQL

Click on the **OK** button on the **Database Modification** tool to generate SQL script. The generated SQL script will be stored in the file you have set. Also [SQL Executor](#) with generated SQL statements for database modifications will appear.

You can easily customize statements according to your wants and wishes. And then send them to the database server by clicking on the **Execute SQL** button. Please examine [SQL Executor](#) section to know more about it.

See also:

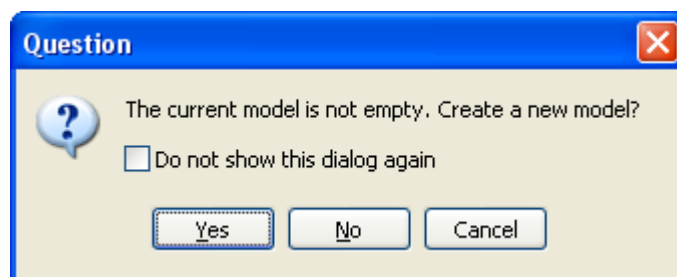
Database Accessing: [SQL Executor](#) | [Connect to a Database](#)

14.11. How to Import from a PostgreSQL Database

You can reverse engineer an existing PostgreSQL database. This means that you can extract the database tables, attributes, relationships, indexes and other objects from the database to your diagram.

To reverse engineer PostgreSQL database:

1. Select **File | Reverse Engineer | PostgreSQL database** or press **Ctrl-R**.
2. If the currently opened diagram already contains some objects, a warning dialog box will appear:



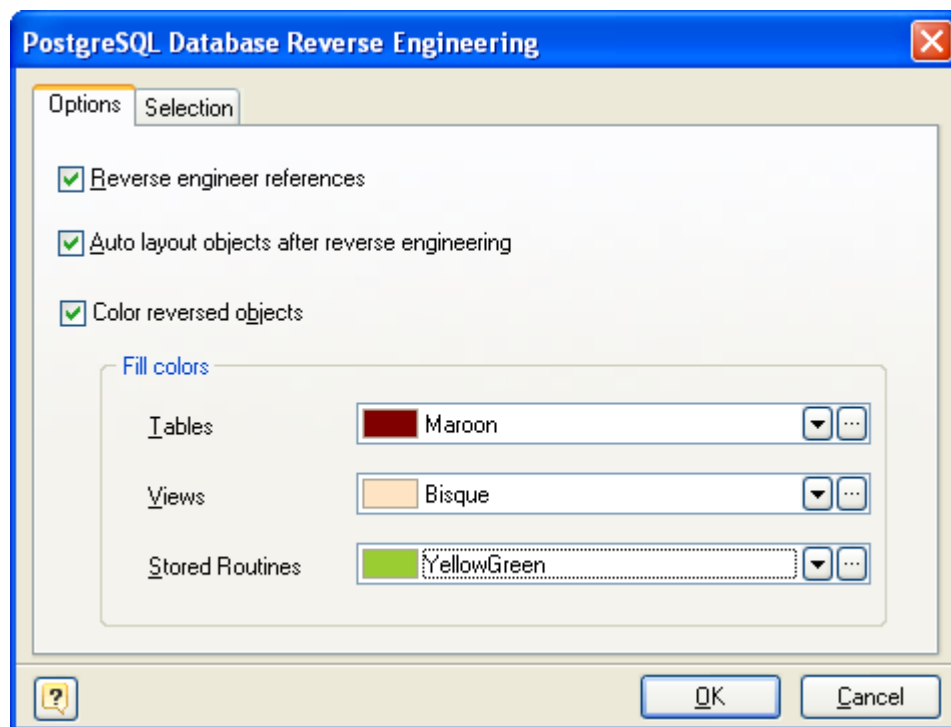
It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram.

Click **Yes** to create a new diagram or **No** to use the currently opened diagram. Click **Cancel** to cancel reverse engineering.

Select **Do not show this dialog again** to disable future notifications.

3. If connection for the current diagram has not been established, [Database Connection Manager](#) will be shown. Select a profile from the list of the available ones or create a new profile to connect to the database you want to reverse engineer.

4. The **PostgreSQL Database Reverse Engineering** tool will be shown. You can set the reverse engineering options in the **Options** tab.



Reverse Engineer references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

Auto layout objects after reverse engineering

Select this option if you want to Designer perform auto layout of reversed objects

Color reversed objects

This option allows you to set fill colors for newly reversed objects. It may be useful if you're performing Reverse Engineering into non empty diagram, thus you may distinguish old objects from reversed ones.

5. In the Selection tab of the **PostgreSQL Database Reverse Engineering** tool you can choose the objects you want to reverse engineer.

There are several subtabs: **Tables**, **Views**, **Stores Procedures** and **Types & Domains**. Each of them allows you to select appropriate database objects. Click on the checkbox near the object to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all objects in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

7. The reversed database objects will be placed in your diagram.

See also:

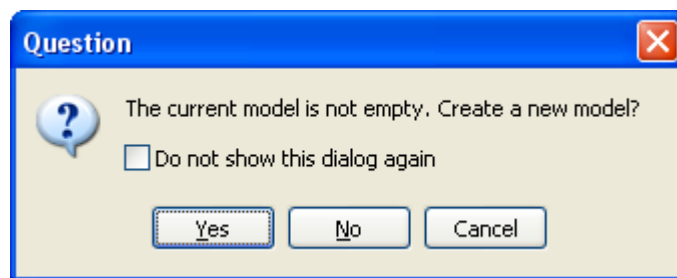
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Import from Access Database](#) | [Universal Reverse Engineering](#) | [SQL Reverse Engineering](#)

14.12. How to Import from a Microsoft Access Database

You can reverse engineer an existing Microsoft Access database. This means that you can extract the database tables, attributes, relationships, indexes and other objects from Microsoft Access database file to your diagram.

To reverse engineer Microsoft Access database:

1. Select **File | Reverse Engineer | Microsoft Access database**.
2. If the currently opened diagram already contains some objects, a warning dialog box will appear:



It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram.

Click **Yes** to create a new diagram, **No** to use the currently opened diagram. **Cancel** to cancel reverse engineering.

Select **Do not show this dialog again** to disable future notifications.

3. The **Access Reverse Engineering** tool will be shown. First, type in the full path to Access database in the **Access File** tab.

4. You can set the reverse engineering options in the **Options** tab.

Tables only

Reverse engineer only tables ignoring views.

Tables and Views

Reverse engineer both tables and views.

Garbage symbols remove/replace

Defines the symbols that will be replaced in the names of the objects.

Replace with

Defines the garbage replacement symbol.

Build references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

Automatically rebuild references when no reference is reversed

If there are no physical references extracted, it is possible to build them from logical structure of the database.

Enabling this feature leads to automatic reconstruction of references. Such reconstruction works by the following scheme: each column of the table is being compared with all primary keys of other tables, and if the column name and data type match one of the primary keys, a reference between the source column and the key column will be created.

This option is available for modification only if Build references is checked.

Tables in a Diagram row

This option defines how reversed tables will be disposed in the diagram. Reversed tables will be placed in the diagram in rows with equal distance, this option determines how many tables maximum there will be in one row.

5. In the **Selection** tab of the **Access Reverse Engineering** tool you can choose the tables you want to reverse engineer.

Click on the checkbox near the table to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all tables in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

7. The reversed database objects will be placed on your diagram.

See also:

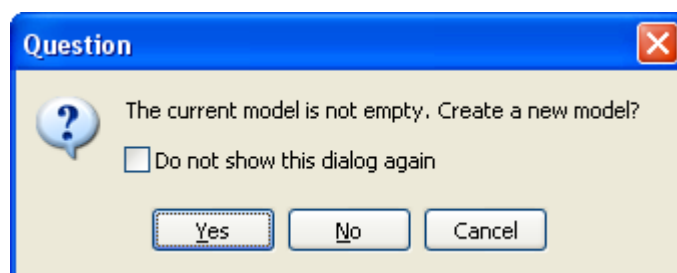
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Universal Reverse Engineering](#) | [SQL Reverse Engineering](#)

14.13. How to Import from an SQL Script

You can reverse engineer an SQL script. This means that you can extract tables, attributes, relationships, indexes and other objects. It is possible with the **SQL Reverse Engineering tool**, which parses an SQL Script using internal parser fully compatible with PostgreSQL standards.

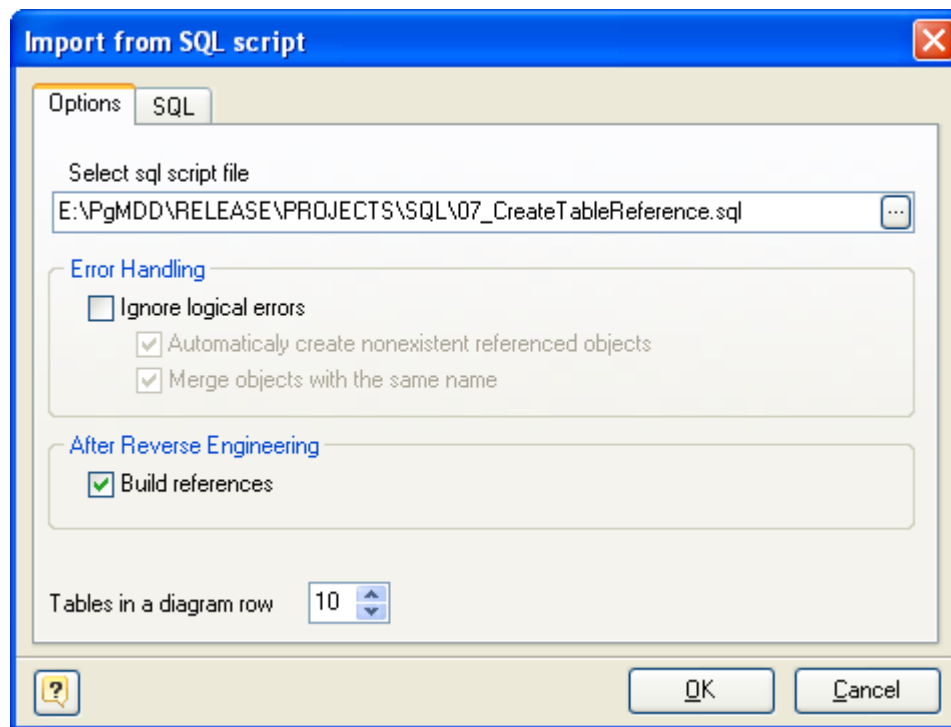
To reverse engineer a script:

1. Select **File | Reverse Engineer | SQL Script...** menu item.
2. If the currently opened diagram already contains some objects, the following dialog box will appear:



It requests you to create a new diagram to place the reversed objects into it, or use the currently opened diagram. Click **Yes** to create a new diagram, **No** to use the currently opened diagram, or **Cancel** to cancel reverse engineering. Select **Do not show this dialog again** to disable future notifications.

3. The **SQL Reverse Engineering** dialog will be shown. First, you should provide options needed for desired behavior of the parser in the **Options** tab.



Click on the ... button to call the standard system **Open File** dialog window.

You can set the following SQL reverse engineering options:

Ignore logical errors

This will cause parser to ignore non syntax errors, e.g. missing referenced objects, name duplicates etc. However, use this option carefully. This may bring some troubles keeping in mind, that the model will not be well-formed in this case.

Automatically create nonexistent referenced objects

Designer will take a try to create necessary missing objects referenced by other ones.

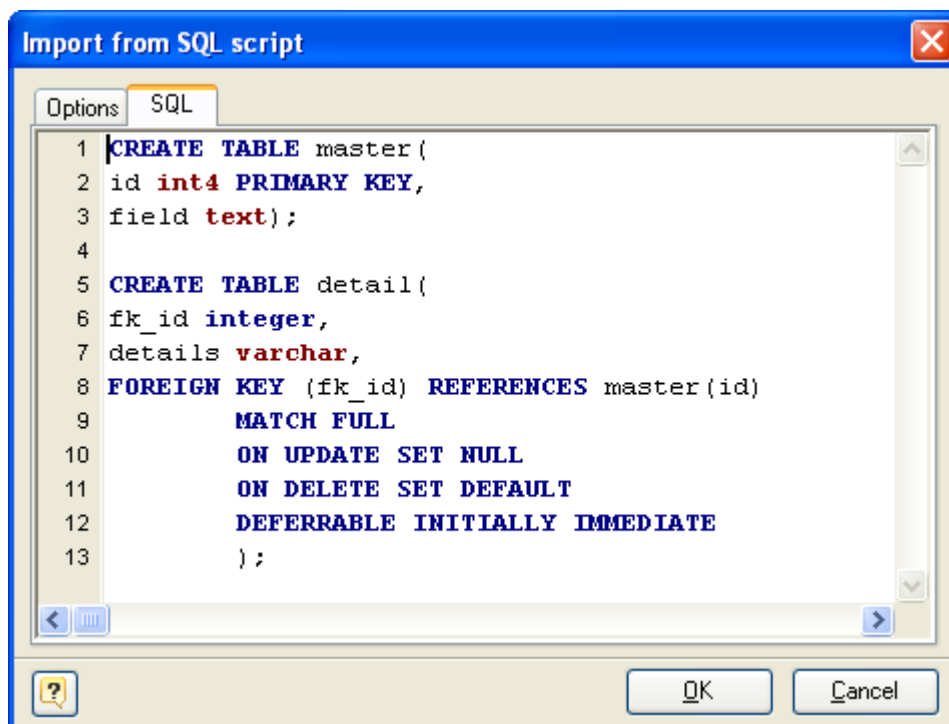
Merge objects with the same name

Objects with the same name will be merged in case this option is checked. The last object in the script has an advantage if some properties can not be merged, e.g. TABLESPACE property of an index or a table and so on.

Build references

This option enables extracting foreign keys from the script and creating corresponding references in your diagram.

4. On the **SQL** tab you can manually edit script loaded from file.



5. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

6. The reversed database objects will be placed in your diagram.

See also:

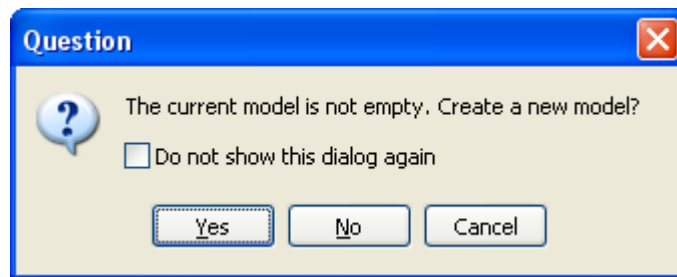
Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Import from Access Database](#) | [Universal Reverse Engineering](#)

14.14. How to Import from other Databases

You can reverse engineer a number of databases, such as Sybase ASA and ASE, Oracle, Informix, MSSQL and others. This means that you can extract tables, attributes, relationships, indexes and other objects. It is possible with the **Universal Reverse Engineering** tool, which connects to a number of databases through OLEDB or ODBC link. So, to reverse engineer a particular database, you need a corresponding OLEDB provider, or ODBC driver. In most cases, such libraries are installed in the system with the help of database client applications.

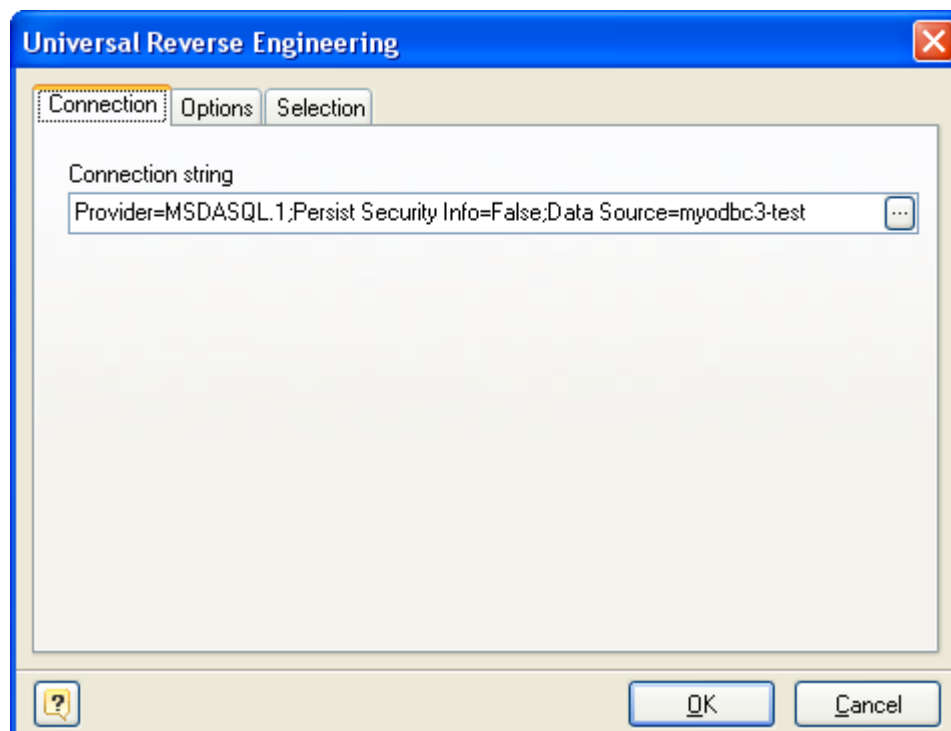
To reverse engineer a database through OLEDB or ODBC link:

1. Select **File | Reverse Engineer | Universal Reverse Engineering** menu item.
2. If the currently opened diagram already contains some objects, the following dialog box will appear:

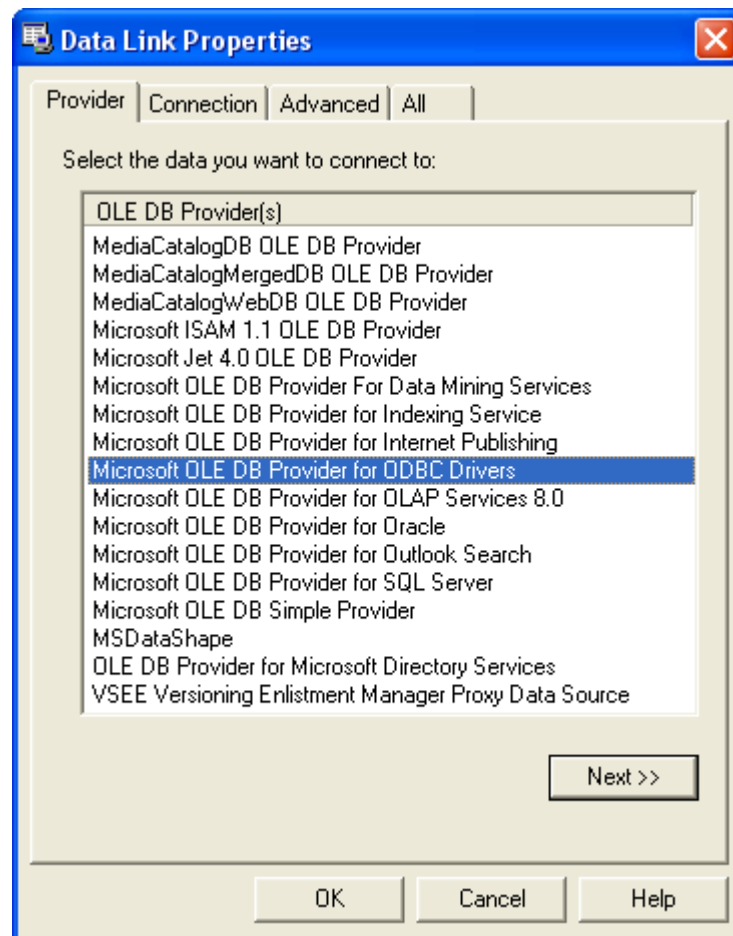


It requests you to create a new diagram to place the reversed objects into or use the currently opened diagram. Click **Yes** to create a new diagram, **No** to use the currently opened diagram, or **Cancel** to cancel reverse engineering. Select **Do not show this dialog again** to disable future notifications.

3. The Universal Reverse Engineering tool will be shown. First, provide full OLEDB connection string in the **Connection tab.**

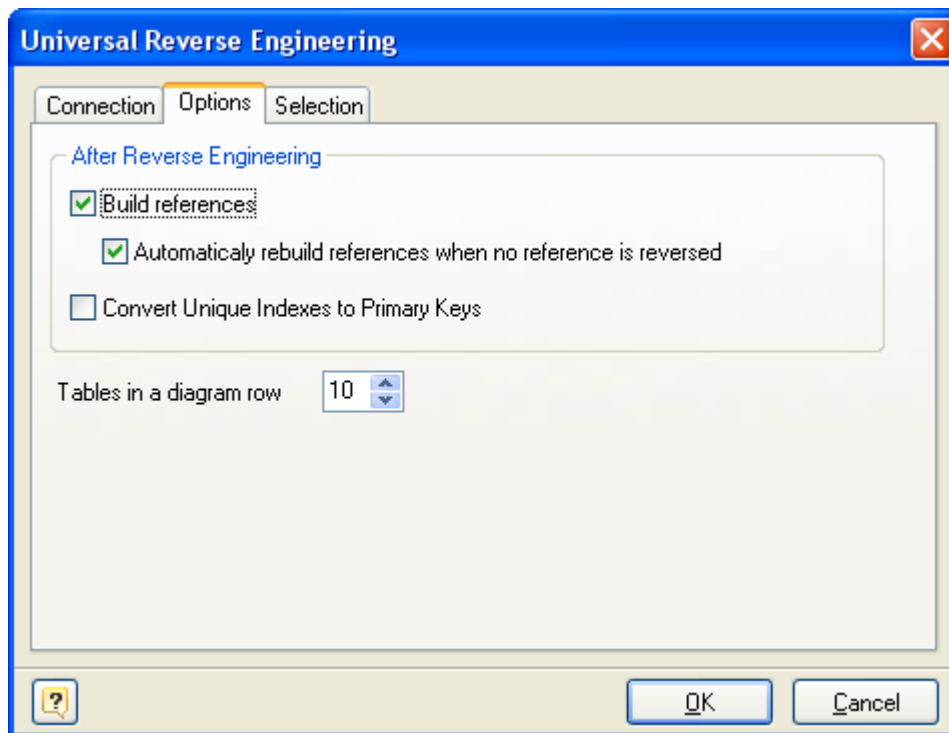


Click on the ... button to call the standard system **Data Link Properties** dialog window, which helps you to build OLEDB connection string:



To use ODBC drivers, click on the Microsoft OLE DB Provider for ODBC Drivers. To use the native OLEDB provider, click on the appropriate item in the list. Browse through tabs to set other tabs and connection properties. Click on the **Help** button in the bottom of the dialog window to learn more about OLEDB connections. Click on the **OK** button to store the data link properties in the **Connection string** field of the **Universal Reverse Engineering** dialog.

4. You can set reverse engineering options in the Options tab:



Build references

This option enables extracting foreign keys from the database and creating appropriate references in your diagram.

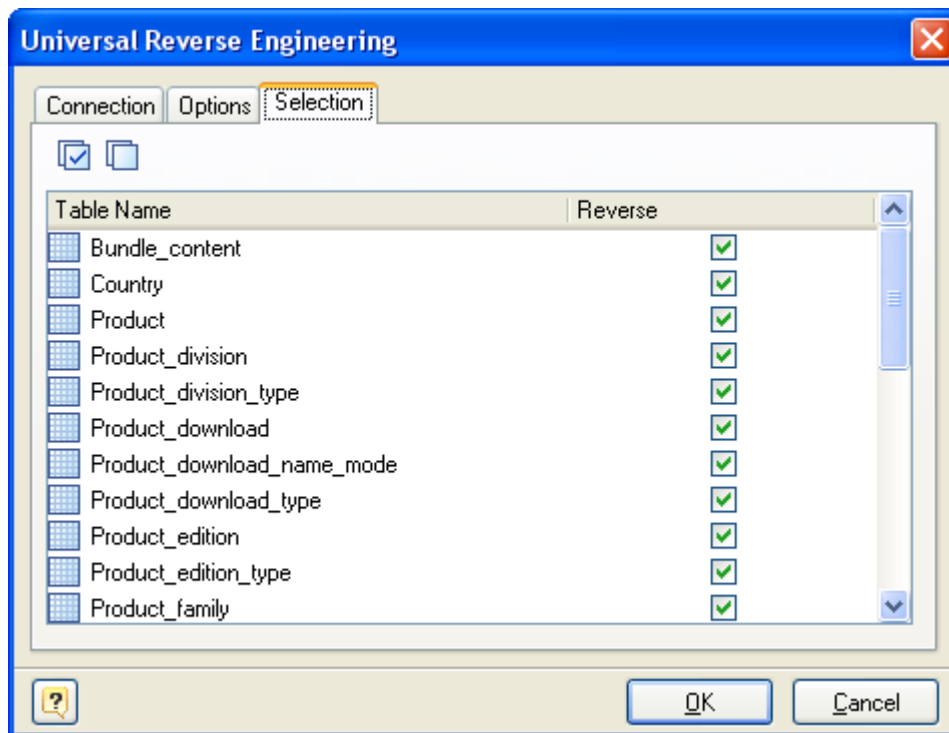
Automatically rebuild references when no reference is reversed

If there are no physical references extracted, it is possible to build them from the logical structure of the database. Enabling this feature leads to automatic reconstruction of references. Such reconstruction works by the following scheme: each column of a table is being compared with all primary keys of other tables, and if the column name and data type match one of the primary keys, a reference between the source column and the key column will be created. This option is available for modification only if Build references is checked.

Tables in a Diagram row

This option determines how reversed tables will be disposed on the diagram. Reversed tables will be placed on the diagram in rows. They will be displayed at an equal distance from each other. This option determines how many tables maximum there will be in one row.

5. In the **Selection** tab of the **Universal Reverse Engineering** tool you can choose the tables you want to reverse engineer.



Click on the checkbox near the table to enable its reverse engineering. Pay attention to the **Select All** and **Deselect All** buttons on the tab, they allow you to select/deselect all tables in the list.

6. Click **OK** to start the database reverse engineering process. The **Output -> Reverse** docking window will display the state of the process.

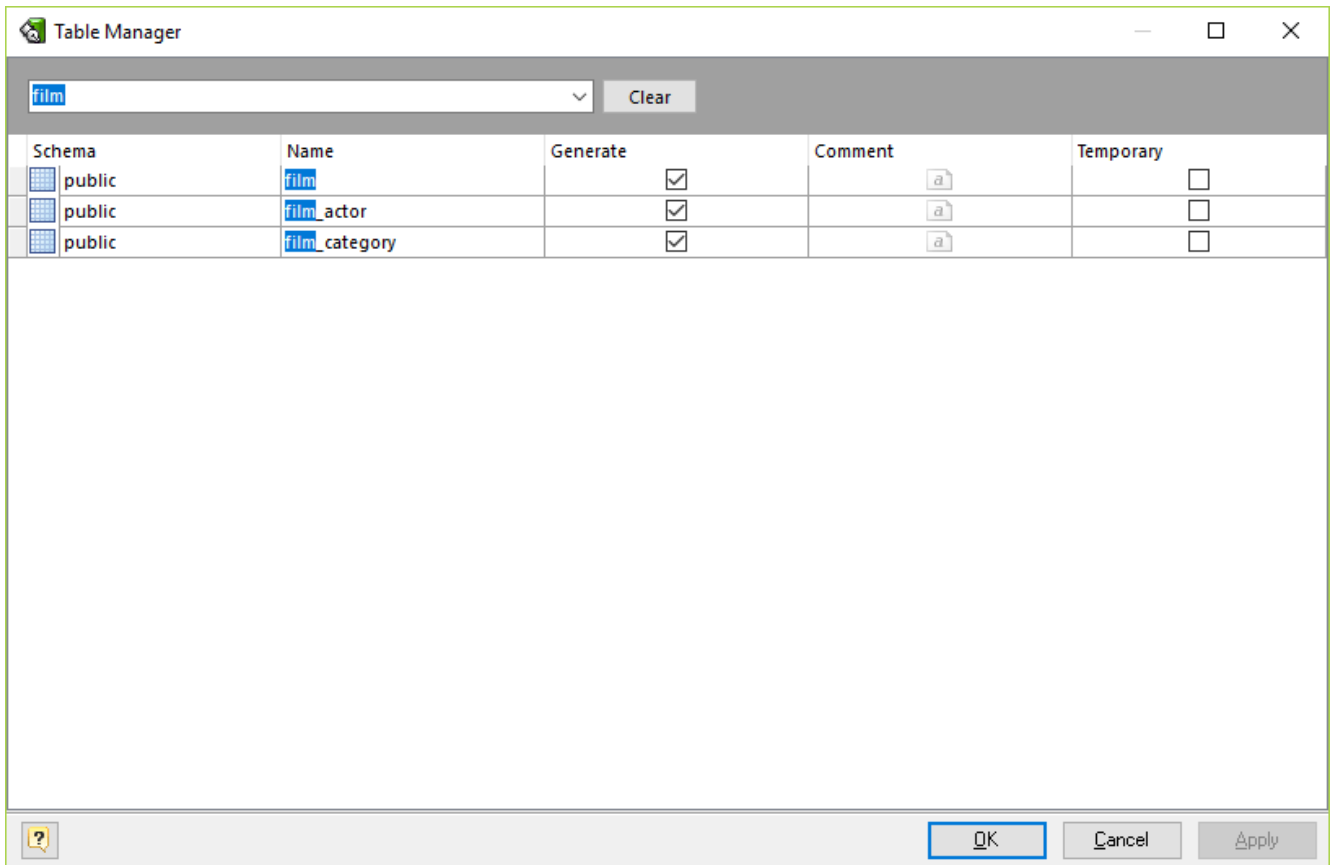
7. The reversed database objects will be placed in your diagram.

See also:

Reverse Engineering and Import: [Reverse Engineering and Import Overview](#) | [Reverse Engineering PostgreSQL Database](#) | [Import from Access Database](#) | [SQL Reverse Engineering](#)

14.15. How to Modify Multiple Tables

Database Designer for PostgreSQL has a great feature that allows you to modify the basic parameters for multiple tables at once. This is the **Table Manager**. To run this tool use the **Diagram | Table Manager** menu item.



Modifying parameter values

The grid in **Table Manager** represents the basic parameters of all diagram tables. The rows stand for the tables, and columns for the table parameters. These parameters can be changed by using [Table Editor](#) for each table one by one, but with **Table Manager** you can do it much more quickly.

Click on the cell to modify the parameter value. Depending on the parameter type, the activated in-place editor can be a text box, a drop-down list, a check box, etc. Please refer to the [Table Editor](#) topic to find out more about all table parameters.

To save your changes click the **OK** button. If you want to store changes and continue editing, click on the **Apply** button.

Searching table in the list

If your diagram contains a large number of tables, it may be important to search for a table in a most easy way. To find a table quickly by its name, press the **Ctrl + F**. Type in the name of table to find. Found tables will be filtered and highlighted.

Running the Table Editor

If you want to change some unavailable table parameters and attributes (such as columns and indexes), select the appropriate table in the dialog and press **Ctrl + Enter**. The [Table Editor](#) will appear, where you can edit all the parameters for the selected table.

See also:

Diagram Objects: [Table Editor](#) | [Column Manager](#) | [Domain Manager](#) | [Index Manager](#)


14.16. How to Execute an SQL Script

You can send SQL queries to the connected PostgreSQL database and display the result. The queries can contain any possible statements, e.g. UPDATE, DELETE, INSERT, SELECT statements etc. Note, that it is possible to run multiple SQL queries simultaneously.

The result of the SELECT-containing queries will be shown in the grid-based dialogues.

To execute SQL queries:

1. Connect to the database using [Database Connection Manager](#)
or
Use the already established database connection.
2. Call the **SQL Executor** by selecting **Database | SQL Executor** menu item or pressing **Ctrl-Shift-E**.
3. Enter one or more SQL queries.
4. Click the **Validate SQL** button on the dialog window toolbar to check your queries. **Database Designer Validator** can check SQL only for grammar correctness. This means that it can't check correctness of the database objects name using, i.e. if you've been mistaken in the name of some table. However, you may execute an SQL without validating it: it will be checked before executing automatically.
5. Press **F9** or click the **Execute SQL** button on the dialog window toolbar to execute your queries. A data grid window will be displayed for each SELECT-based query. To close the data grid window, click on the **Close** button. The execution status for each query will be displayed at the bottom of the **SQL Executor** dialog window.
6. Press **Alt-F9** or click **Execute SQL in Single Transaction** button on the dialog window toolbar to execute your script in the context of the single transaction without pre-validation. This means the whole script will be sent to server without parsing and validation.

You can save your queries into a file. Click on the Save () button on the SQL Executor toolbar.

Please note, that SQL Executor dialogue window is used by [Database Generation](#) and [Database Modification](#) tools.

See also:

Database Generation: [Database Generation](#)

Database Functions: [Database Modification](#)

14.17. How to Merge Diagrams

The **Merge Diagram** tool of the **Database Designer for PostgreSQL** allows you to merge content of two diagrams. This allows you to create new cumulative diagram, which will include content of your two diagrams.

To merge two diagrams, please follow these steps:

1. [Open](#) two diagrams you want to merge. One of them will accumulate its own content and content of other diagram.
2. Start **Merge Diagram** tool by selecting the **Diagram | Merge Diagram** menu item.
3. Select diagrams you want to merge. Make sure that you have chosen **Options** tab of the **Merge Diagram** tool.

From diagram

Choose the *source diagram* from drop-down menu, which contains list of opened diagrams. Objects of this diagram will be added to the *destination* diagram.

To diagram

Choose the *destination diagram* from the drop-down menu, which contains the list of the opened diagrams. This diagram will contain its own objects and objects of *source diagram*. Set the name of the database user who will own the new database, or type DEFAULT to use the default (namely, the user executing the command).

4. Select the *source diagram* objects to merge. Go to the Objects tab. Click on the appropriate checkboxes to select objects you want to add to the *destination diagram*. To select/deselect all objects of particular type, click on appropriate checkbox near them. There are the following objects types: Domains, Notes, Tables, References (by default all objects are already selected).
5. Click **OK** to add the selected objects of the *source diagram* to the *destination diagram*.
6. Save the *resulted (destination)* diagram to a new file.

14.18. How to Find Errors in a Diagram

The **Check Diagram** tool of the **Database Designer for PostgreSQL** allows you to check your database diagram for most typical errors and defects. The result of the check goes in a well structured form, using which you can easily bring your diagram to correspondence with the common standards of database modeling.

To open the **Check Diagram** dialog press **F4** or select the **Diagram | Check Diagram** menu item.

Use the **Select diagram** drop-down list to select the diagram from the list of currently opened diagrams.

The tree list below allows you to select what warnings and errors should be taken into account during the check. All warnings and errors are divided into categories, which correspond to the diagram objects. Remove selection from the warning/error or from the whole category to exclude it from the check.

These are the descriptions for all available warnings and errors:

Table

Error "Table Name Uniqueness"

Check the diagram for the uniqueness of each table name within a schema;

Warning "Table Name Max Length"

PostgreSQL allows only 63 characters in table names and cuts names if they are longer than this;

Warning "Column Definition"

Check if each table within the diagram owns at least one column;

Warning "Index Definition"

Check if each table within the diagram owns at least one index;

Warning "Primary Key Definition"

Check if a primary key is defined for each table within the diagram;

Warning "Reference Definition"

Check if each table within the diagram is linked with other tables;

Error "Auto-increment columns"

Check if each table within the diagram has no more than one auto-increment column, as otherwise PostgreSQL will not allow such table to be created;

Table Columns

Error "Column Name Uniqueness"

Check diagram tables for the uniqueness of each column name within the table;

Warning "Column Name Max Length"

PostgreSQL allows only 63 characters in column names and cuts names if they are longer than this;

Warning "Auto-increment Column Definition"

Check if each auto-increment column within a table is a part of a primary key;

Table Indexes

Error "Index Name Uniqueness"

Check diagram tables for the uniqueness of each index name within the table;

Warning "Index Name Max Length"

PostgreSQL allows only 63 characters in index names and cuts names if they are longer than this;

Warning "Duplicate Index Column"

Check if each table column is indexed only once;

References

Error "Reference Column Data Types"

Check whether the linked columns are of the same data type for each diagram reference;

Domains

Error "Domain Name Uniqueness"

Check diagram for the uniqueness of each domain name within a schema.

Stored Routine

Error "Stored Routine Name Uniqueness"

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "Stored Routine Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Views

Error "View Name Uniqueness"

Check diagram stored procedures and functions for the uniqueness of name within a schema.

Warning "View Name Max Length"

PostgreSQL allows only 63 characters in stored routine names and cuts names if they are longer than this;

Error "View was created on a not existing table"

Check existence of tables, which are used in the view;

Error "View was created on a not existing column"

Check existence of table columns, which are used in the view.

After you click **OK** the check process will be displayed within the **Output** window and the result of the check will be displayed within the **Result** window in the same categorized view as described above.

Double-click on a warning or an error in the list opens the editor window for the appropriate object ([Table Editor](#), [Index Editor](#), or [Domain Manager](#)).

See also:

Diagram Objects: [Table Editor](#) | [Column Editor](#) | [Reference Editor](#) | [Index Editor](#)

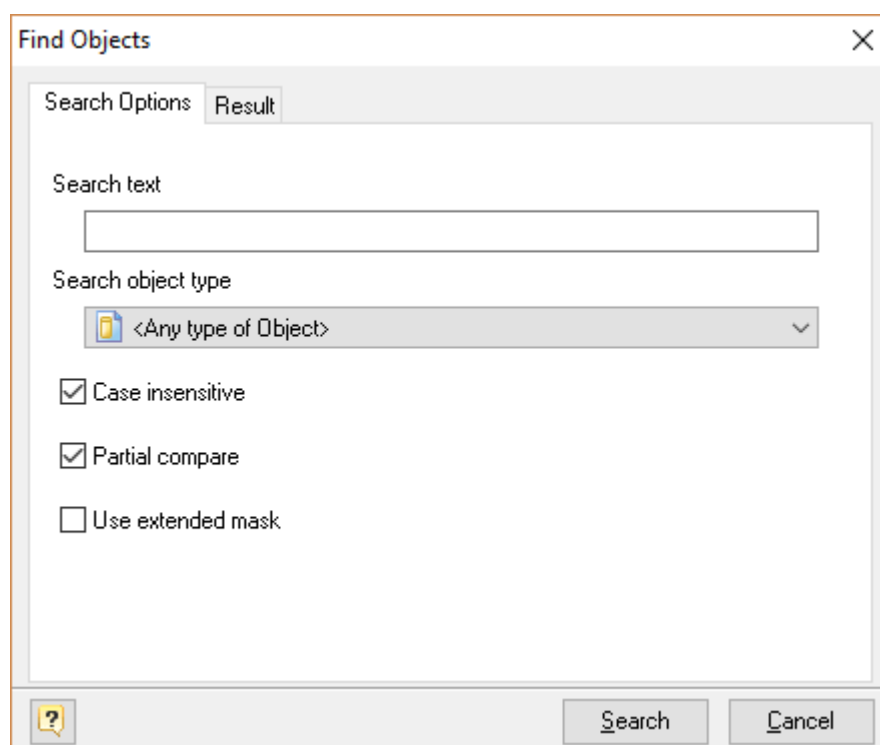
14.19. How to Find Objects

The **Find objects** feature lets you search objects within the entire **Database Designer for PostgreSQL** diagram so that you could locate the necessary objects in the diagram easily, find all the attributes related to a given object.

To call **Find Objects** dialog window, select the **Edit | Find Objects** menu item or press **Ctrl-F**.

Defining find parameters

You can define the search parameters in the **Find Objects** dialog window.



Search text

This option allows you to define the text that you want to search in the objects and attributes names.

Search object type

You can define the object type which you want to search for. Also you can search for any type of objects, please select one of the following object types:


- Table
- Column
- Constraint
- Index
- Trigger
- Rule
- Reference
- View
- Stored Routine

Case insensitive

If this option is enabled, case matching is off and it will not affect on the search results.

Partial compare

If this option is enabled, you will find objects with partial name matching.

 If **Use extended mask** and **Partial compare** are checked, this means only that to **Search text** leading and trailing percent mark (%) will be added, e.g. %<Search Text>%.

Use extended mask

You can search objects which conform to the format specified by a **Search text**. A valid mask consists of literal characters, sets, and wildcards.

Each literal character must match a single character in the string.

Each set begins with an opening bracket ([) and ends with a closing bracket (]). Between the brackets are the elements of the set. Each element is a literal character or a range. Ranges are specified by an initial value, a dash (-), and a final value. Do not use spaces or commas to separate the elements of the set. A set must match a single character in the string. The character matches the set if it is the same as one of the literal characters in the set, or if it is in one of the ranges in the set. A character is in a range if it matches the initial value, the final value, or falls between the two values. If the first character after the opening bracket of a set is an exclamation point (!), then the set matches any character that is not in the set.

Wildcards are percent (%) or question marks (?). A percent matches any number of characters. A question mark matches a single arbitrary character.

Any character may be escaped using "\" character. To include "\"" itself just double it, e.g. "\""

Examples:**analy[sz]e**

Conforms to both American and British spelling of "analyze" and "analyse"

[!a-fz]_column

Conforms to words where the first letter is not a..f or "z"

anal%

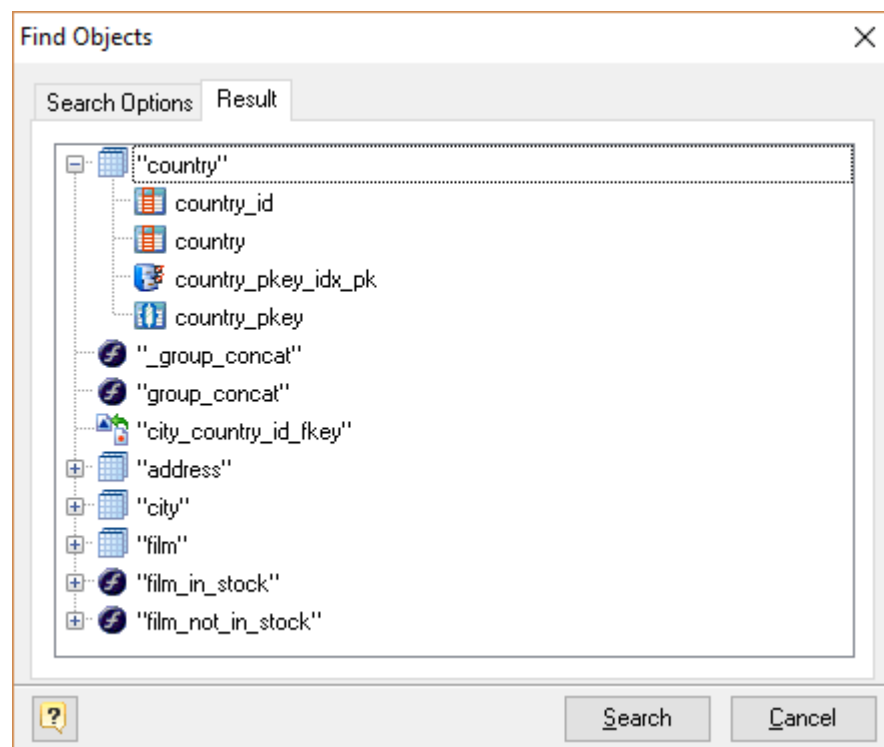
Conforms to analyzer, analog, analyze, analyse etc.

de?r

Conforms to dear, deer, deor, depr, etc.

array\[?\]

Conforms to array[a], array[5], array[_], array[!], etc.

Using the Result List

The **Result** tab displays the result of the search in the result tree. You can use the result tree to:

- learn to which objects the found objects (attributes) belong to;
- modify the found objects.

Double-click on the object in the result tree to call corresponding editor.

14.20. How to Print a Diagram


Printing a database diagram gives you a picture of your database structure to refer to or distribute.

Before you start printing, arrange the objects in the database diagram to your satisfaction. You can change the shape, and position of the objects in the diagram without affecting their definitions in the database.

To change the layout of the diagram, move, size, and shape the objects. For example, you can use the mouse to move tables, or use the [Auto Layout Diagram](#) command to automatically reposition the objects.

You can change physical parameters of your paper, on which diagram will be printed out, specify margins, page headers and footers information using [Page Setup](#) dialog.

Before printing out a diagram, you can see how it will look paper, use the [Print Preview](#) tool for that.

To print out a diagram, select the **File | Print** menu item or press **Ctrl-P**. You also can call the **Print Setup** dialog by clicking on the **Print** () icon on the Standard toolbar.

The **Print Setup** dialog is displayed. In the **Print Setup** dialog select the printer you want to print on, and then click the **OK** button to print a diagram.

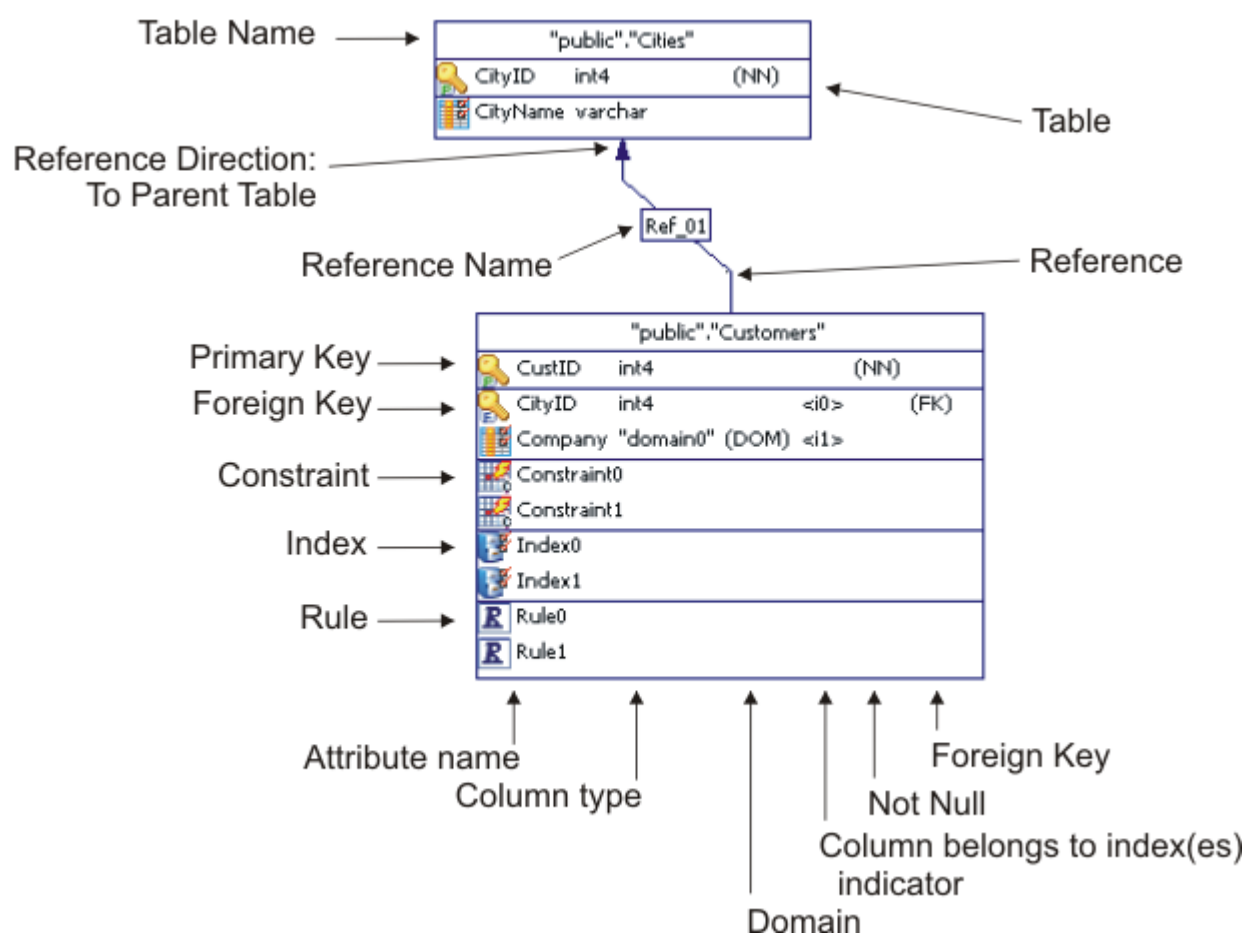
See also:

Printing: [Page Setup](#) | [Print Preview](#)

Diagram: [Auto Layout Diagram](#)

14.21. How to Read a Diagram

Please examine the following picture, which describes the diagram objects notation:



This way of diagram objects displaying is the most informative. You can change the displaying preferences according to your likes and dislikes. Please see the [Diagram Display Preferences](#) section for more information.

See also:

Entity Relationship Diagram: [Diagram Display Preferences](#)

14.22. How to View an SQL Table Definition

Table SQL preview

You can see the SQL representation of your table, it includes columns, indexes, foreign keys, etc.

To see SQL representation of the table, double click on the table symbol and [Table Editor](#) will appear. Go to the **Preview** tab. SQL syntax elements are colored. It is possible to copy this definition to clipboard.

See also:

Diagram Objects: [Table Editor](#)

Diagram: [Diagram SQL Preview](#)

14.23. How to Export a Diagram to Graphics

You can export your diagram to an image file and then use its graphical representation in external applications. For example, insert a diagram image into your program documentation (e.g. use MS Word), create big posters with your diagram, publish them to Web.

Database Designer for PostgreSQL supports the following image types:

- Vector graphics: Windows Enhanced Metafile (EMF).
- Bitmap images: PNG, GIF, JPEG, BMP.

To export model to image, follow these steps:

1. Select **File | Export** to call the **Export Model** tool.
2. The export dialog window will appear. Please examine the dialog window controls:

File name

Enter the name for the file, in which you want to store the contents of the model.

Split into pages

Click on this option to create a separate image for each page in the model.

Show image on complete

Click on this option to open image in a default image viewer after the generation.

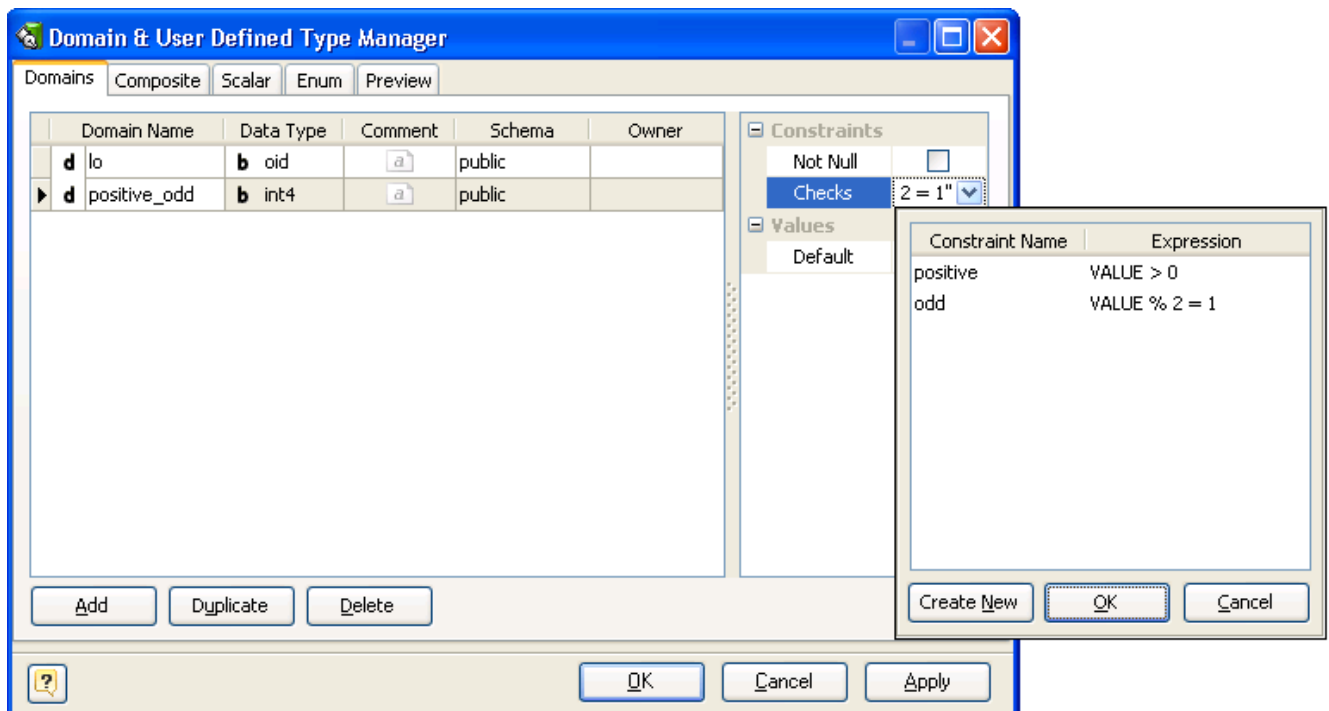
Click **OK** to export model.

Please note, that exporting a diagram to the vector graphics format allows you to freely scale a diagram of any size without any noticeable distortions. It enables you to create big posters with the diagram to demonstrate the data logic of your application.

14.24. How to Create a Domain

The **Domain & User Defined Type Manager** is intended for managing diagram domains, composite and scalar user defined types, which can be used for faster creating and modifying table columns, stored procedures etc.

To open the **Domain & User Defined Type Manager**, select the **Diagram | Domain & User Defined Type Manager** menu item.



The **Domain** page consists of the following areas:

Domain List

The **Domains** list displays all the domains in the diagram and allows you to modify the following domain properties:

- **Domain name** - the name of the domain, which must be unique within the schema;
- **Data type** - the data type of the domain;
- **Comment** - an arbitrary description for the domain.
- **Schema** - the database schema that domain belongs to.

Properties Pane

The properties pane allows you to define the advanced properties of the domain, selected in the Domain List. The appearance of this pane changes according to the data type of the domain. These properties are:

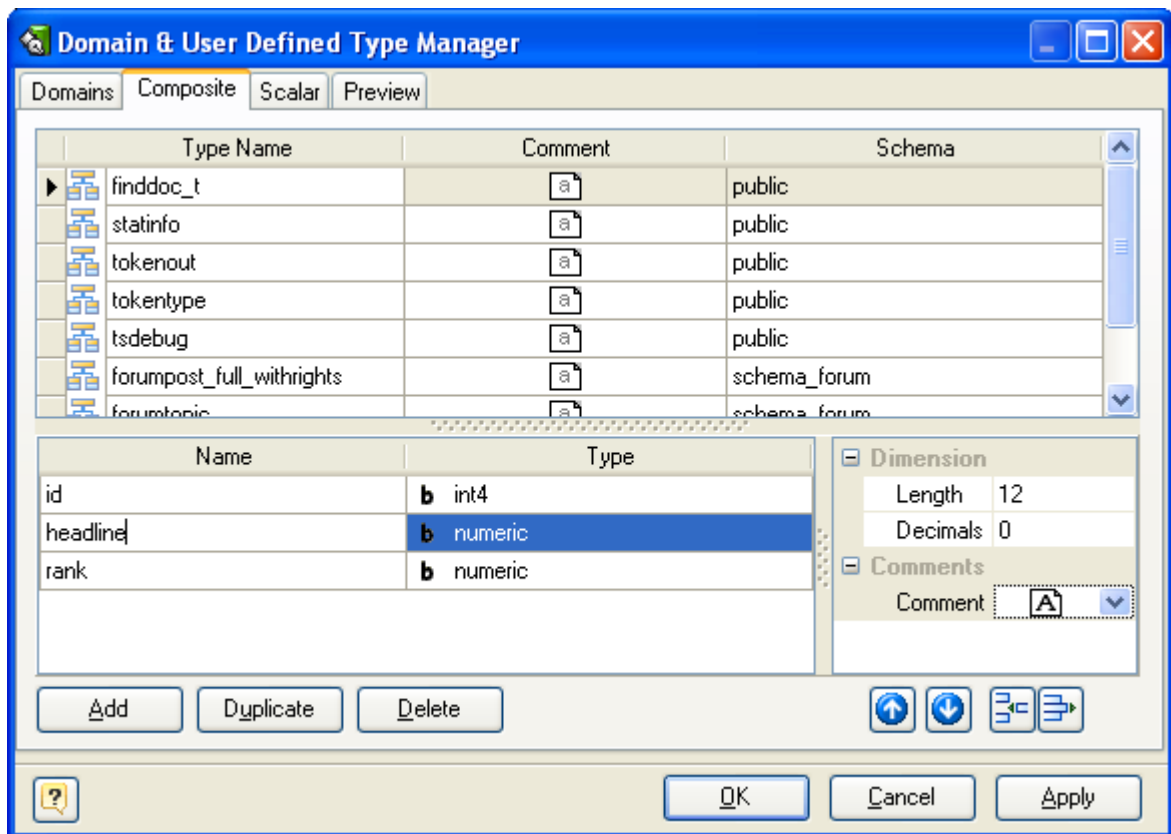
- **Not null** - this option indicates that a domain-based column value cannot be NULL;
- **Default** - this attribute defines the default value, which the domain-column accepts if no other is specified;

- **Checks** - this property specifies integrity constraints or tests which values of the domain must satisfy. Each constraint must be an expression producing a Boolean result. It should use the name VALUE to refer to the value being tested.

Buttons Pane

The buttons under the list of domains allows you to perform the following actions:

- **Add** - add a new domain with the default properties to the end of the list;
- **Duplicate** - add a new domain with the same properties as the selected domain to the end of the list;
- **Delete** - remove the selected domain from the list.



The **Composite** page consists of the following areas:

Composite List

The **Composite** list displays all the composites in the diagram and allows you to modify the following composite properties:

- **Type name** - the name of the composite type, which must be unique within the schema;
- **Comment** - an arbitrary description for the composite type.
- **Schema** - the database schema that composite type belongs to.

Type Attribute (Column) List

Type Attribute (Column) List displays all the columns in the selected composite type and allows you to modify the following properties:

- **Attribute name** - the name of the composite type attribute, which must be unique within the parent type;
- **Type** - the data type of the attribute.

Properties Pane

The properties pane allows you to define the advanced properties of the attribute, selected in the Type Attribute List. The appearance of this pane changes according to the data type of the attribute. These properties are:

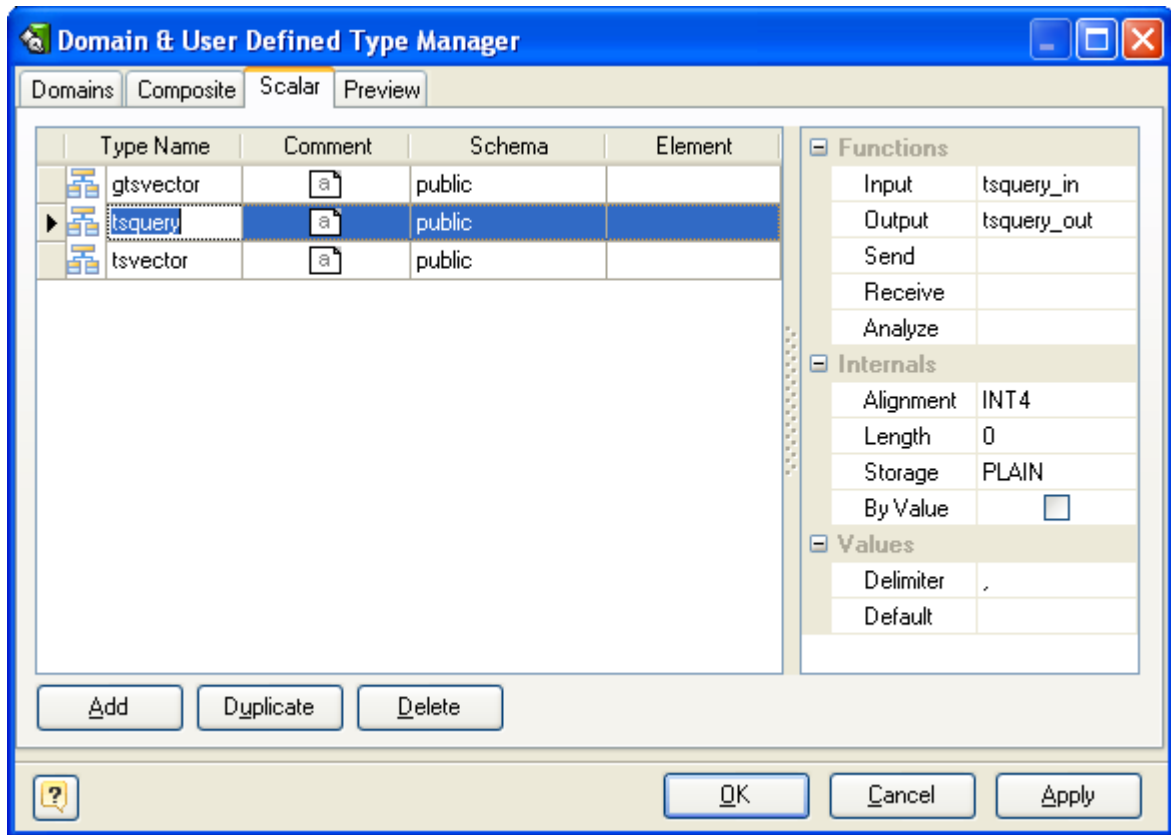
- **Length** - this option indicates maximum allowed length for a column;
- **Decimals** - this attribute defines the number of decimal digits in the fractional part of a numeric attribute;
- **Comment** - an arbitrary description for the composite type attribute.

Buttons Pane

The Buttons Pane is similar to Buttons Pane on the Domain Page, but have additional buttons:

The buttons under the list of columns allow you to perform the following actions:

- **Up/Down** - move the selected column along the list;
- **Add** - add a new column with the default properties to the end of the list;
- **Delete** - remove the selected column from the list.



The **Scalar** page consists of the following areas:

Scalar List

The scalar list displays all the scalars in the diagram and allows you to modify the following scalar properties:

- **Scalar name** - the name of the scalar, which must be unique within the schema;
- **Comment** - an arbitrary description for the scalar;
- **Schema** - the database schema that scalar belongs to;
- **Element** - specifies that scalar type being created is an array; this specifies the type of the array elements.

Properties Pane

The properties pane allows you to define the advanced properties of the scalar, selected in the Scalar List. These properties are:

- **Input** - the name of a function that converts data from the type's external textual form to its internal form;
- **Output** - the name of a function that converts data from the type's internal form to its external textual form;
- **Receive** - the name of a function that converts data from the type's external binary form to its internal form;
- **Send** - the name of a function that converts data from the type's internal form to its external binary form;
- **Analyze** - the name of a function that performs statistical analysis for the data type;
- **Alignment** - the storage alignment requirement of the data type. It must be CHAR, INT2, INT4, or DOUBLE; the default is INT4;
- **Length** - a numeric constant that specifies the length in bytes of the new type's internal representation. The default assumption is that it is variable-length;
- **Storage** - the storage strategy for the data type. If specified, must be PLAIN, EXTERNAL, EXTENDED, or MAIN; the default is PLAIN;
- **By Value** - indicates that values of this data type are passed by value, rather than by reference;
- **Delimiter** - the delimiter character to be used between values in arrays made of this type;
- **Default** - the default value for the data type. If this is omitted, the default is null.

Buttons Pane

The buttons under the list of scalars allows you to perform the following actions:

- **Add** - add a new scalar with the default properties to the end of the list;
- **Duplicate** - add a new scalar with the same properties as the selected scalar to the end of the list;
- **Delete** - remove the selected scalar from the list.

See also:

Diagram Objects: [Domains & User Defined Types](#) | [Columns](#) | [Stored Routine Editor](#)

14.25. How to Insert a Comment


You can type in the **Note** box any comment concerning your diagram. **Note** is useful for writing

various comments, such as object functionality or role description, things to do, etc in your diagram.

The following picture demonstrates a sample Note:

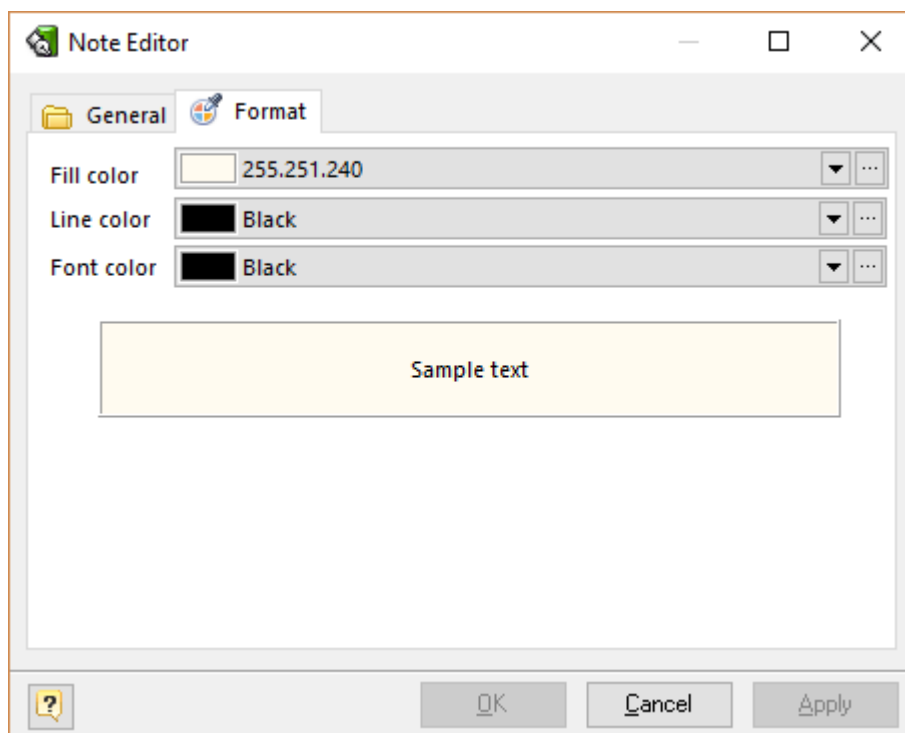
I have to add the Employee table

Placing Note to the diagram

To place a note in the diagram, select the **Note** tool () from the **Palette** toolbar. Then click anywhere in the diagram. Certainly, you can place multiple notes in one diagram.

Modify the text in Note

To modify the information, that is displayed in **Note**, double click on it. Then the following **Note editor** will appear:



In the drop-down menu select the color that you want to use for the background in your note. Enter the text you want to be shown inside the Note object in the diagram.

 Please note, that **Note** is auxiliary object, it doesn't affect database generation.

14.26. How to Change Table Formatting

You can set table line and fill color for displaying on the diagram, different from the default table colors, which are defined within the [Diagram Display Preferences](#) dialog.

Double click on the table symbol and [Table Editor](#) will appear. Go to the **Format** tab.

Fill Color

Use this option to set a background color of the table. Choose the color you need from this drop-down list.

Line Color

Use this option to set a border color of the table. Choose the color you need from this drop-down list.

See also:

Diagram Objects: [Table Editor](#)

Diagram: [Diagram Display Preferences](#)

15. Wine Configuration

To run **Database Designer for PostgreSQL** on your non-Windows box under Wine you should use at least:

- [Wine](#) 1.0.1
- [Winetricks](#) version 20091022.

Database Designer for PostgreSQL uses GDI+ library in the work. To install appropriate files please use the command:

```
$sh winetricks gdiplus
```

Since **Database Designer for PostgreSQL** uses Microsoft Data Access Components for its Universal and MS Access Database Reverse Engineering appropriate redistributable runtime libraries should be installed:

```
$sh winetricks jet40 mdac28
```

jet40 - MS Jet 4.0 Service Pack 8

mdac28 - MS MDAC 2.8: Microsoft ODBC drivers, etc.



Database Designer for PostgreSQL will function normally even without these runtime libraries: only Universal and MS Access Database Reverse Engineering functions will be disabled.

After this application already can be used with its full functionality. However, fonts used in Designer appear ridged. To fix this install Microsoft Core Fonts (Arial, Courier, Times etc.), but make sure [cabextract](#) utility is present in the system.

```
$ssh winetricks corefonts
```

16. Purchase

Thank you for your interest in purchasing **Database Designer for PostgreSQL!**

You can select the purchasing method (CC online, Purchase Order, Fax or wire transfer) and licensing options you prefer and also register for **Database Designer for PostgreSQL** at <http://www.microolap.com/products/database/postgresql-designer/order/>.

17. Support

We provide our customers with efficient and high-professional support, which is absolutely free. Whether you have any questions about **Database Designer for PostgreSQL** or you want to report a bug, feel free to do it by using our [on-line Support System](#).

18. License Agreement

NOTICE TO USER:

THIS IS AN AGREEMENT GOVERNING YOUR USE OF THE SOFTWARE TITLED **Microolap Database Designer for PostgreSQL**, FURTHER DEFINED HEREIN AS "PRODUCT," AND THE LICENSOR OF THE PRODUCT IS WILLING TO PROVIDE YOU WITH ACCESS TO THE PRODUCT ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS AND CONDITIONS CONTAINED IN THIS AGREEMENT. BELOW, YOU ARE ASKED TO ACCEPT THIS AGREEMENT AND CONTINUE TO INSTALL OR, IF YOU DO NOT WISH TO ACCEPT THIS AGREEMENT, TO DECLINE THIS AGREEMENT, IN WHICH CASE YOU WILL NOT BE ABLE TO INSTALL OR OPERATE THE PRODUCT. BY ACKNOWLEDGING YOUR CONSENT HERETO AND BY INSTALLING AND OPERATING THIS PRODUCT YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. For purposes hereof **"Operating"** shall mean accessing, storing, loading, installing, Using (as defined below), and copying the Product into the memory of a Client Device, as defined below. **"Using"** shall mean executing and displaying the product on a Client Device or otherwise benefiting from utilizing, deploying or using the Product or its functionality.

This Electronic End User License Agreement (the **"Agreement"**) is a legal agreement between you (either an individual or an entity), the Licensee, and MicroOLAP Technologies Ltd., (the **"Licensor"**), regarding the Product and related support service you are about to install and Operate and/or other related services, including without limitation:

- a) all of the contents of the files, including disk(s), CD-ROM(s) or other media with which this Agreement is provided and including all forms of code, such as Source Code, Object Code, dynamic or static libraries, and/or executable files as provided and in a form that is provided by Licensor to you (the **"Software"**). For the avoidance of doubt, by way of example, but not exclusion, if a specific file is provided by Licensor in Object Code only, the Source Code for such files shall not be deemed a part of the Software provided by Licensor to you. For purposes hereof **"Source Code"** shall mean the human-readable form of the computer programming code and related system documentation including all comments and any procedural code such as job control language and **"Object Code"** shall mean computer programs assembled or compiled in magnetic or electronic binary form on software media, which are readable and usable by machines, but not generally readable by humans without reverse-assembly, reverse-compiling, or reverse-engineering.

b) all support services provided to you by Licensor in connection with the Software (the "**Services**");

c) and all successor upgrades, modified versions, modified modules, revisions, patches, enhancements, fixes, modifications, copies, additions or maintenance releases of the Software, if any, licensed to you by the Licensor (collectively, the "**Updates**"), and

d) related user documentation and explanatory materials or files provided in written, "online" or electronic form (the "**Documentation**" and together with the Software, Samples, Updates, and Services the "**Product**").

For the purposes of this Agreement, "**Licensor Site**" shall mean the Internet website maintained by or on behalf of Licensor from which the Software is available for download pursuant to a license from Licensor. The Licensor Site is currently located at <http://www.microolap.com>.

You are subject to the terms and conditions of this End User License Agreement whether you access or obtain the Product directly from the Licensor, or through any other source. For purposes hereof, "**you**" or "**Licensee**" means the individual person installing or using the Product on his or her own behalf; or, if the Product is being downloaded or installed on behalf of an organization, such as an employer, "**you**" means the organization for which the Product is downloaded or installed, then the person accepting this agreement represents hereby that such organization has authorized such person to accept this agreement on the organization's behalf. For purposes hereof the term "**organization**," without limitation, includes any partnership, limited liability company, corporation, association, joint stock company, trust, joint venture, labor organization, unincorporated organization, or governmental authority.

If you do not agree to the terms and conditions of this Agreement, the Licensor is unwilling to license the Product to you. In such event, you may not Operate the Product in any way.

IF THE LICENSOR AND YOU HAVE AGREED ON AND PROPERLY EXECUTED A SEPARATE CONTEMPORANEOUS OR SUBSEQUENT TERMS OF USE OR EXHIBITS (the "Terms of Use**"), WHICH ARE SUPPLEMENTAL, DIFFERENT OR INCONSISTENT WITH THE TERMS OF THIS AGREEMENT, SUCH TERMS OF USE SHALL CONTROL, PROVIDED THAT (i) SUCH TERMS OF USE SPECIFICALLY ACKNOWLEDGE AND REFER TO THIS AGREEMENT, AND (ii) ALL OTHER TERMS AND CONDITIONS OF THIS AGREEMENT REMAIN IN FULL FORCE AND EFFECT.**

BEFORE YOU PUT A CHECKMARK AT THE "I ACCEPT THE AGREEMENT" BUTTON AND PRESS "NEXT," PLEASE CAREFULLY READ THE TERMS AND CONDITIONS OF THIS AGREEMENT, AS SUCH ACTIONS ARE A SYMBOL OF YOUR SIGNATURE AND BY CLICKING ON THE "I ACCEPT THE AGREEMENT" AND "NEXT" BUTTONS, YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PARTY TO THIS AGREEMENT AND AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE "CANCEL" BUTTON AND THE PRODUCT WILL NOT BE INSTALLED ON YOUR CLIENT DEVICE, AS SUCH TERM IS DEFINED BELOW.

For your reference, you may refer to the copy of this Agreement that can be found in installed files of the Software as **license.rtf**.

You may also receive an electronic copy of this Agreement by contacting Licensor at sales@microolap.com.

1. Proprietary Rights and Non-Disclosure.

1.1. Ownership Rights. You agree that the Product and the authorship, systems, ideas, methods of operation, derivative Documentation and other information contained in the Product, are proprietary intellectual properties and or the valuable trade secrets of the Licensor and are protected by applicable civil and criminal law, and by the law of copyright, trade secret, trademark and patent and international treaties. You may use trademarks only insofar as to identify printed output produced by the Product in accordance with accepted trademark practice, including identification of trademark owner's name. Such use of any trademark does not give you any rights of ownership in that trademark. The Licensor and its suppliers own and retain all right, title, and interest in and to the Product, including all copyrights, patents, trade secret rights, trademarks, and other intellectual property rights therein. Your possession, installation or use of the Product does not transfer to you any title to the intellectual property in the Product, and you will not acquire any rights to the Product except as expressly set forth in this Agreement. All copies of the

Product made hereunder must contain the same proprietary notices that appear on and in the Product. Except as stated herein, this Agreement does not grant you any intellectual property rights in the Product.

1.2. Source Code and Modifications. You acknowledge that the source code for the Product is proprietary to the Licensor and constitutes trade secrets of the Licensor. Except as otherwise specifically provided herein or in Terms of Use, you agree not to disassemble, decompile or "unlock", decode or otherwise reverse-translate or reverse-engineer, or attempt in any manner to reconstruct or discover any source code or underlying algorithms of the Product or any part thereof provided solely in Object Code form but you may change, add or delete any files of the licensed copy of the Products and you may adapt or modify the Source Code solely for purposes of Operating a licensed copy of the Product and as expressly permitted pursuant to the type of the License purchased hereunder *provided* that you may not, in any event, remove or alter any copyright notices or other proprietary notices on any copies of the Product, whether so modified or not, *and further provided* that any such change, addition, deletion, adaptation or modification voids any express warranty provided herein and terminates any right to support services.

1.3. Confidential Information. You agree that, unless otherwise specifically provided herein the Product, including the specific design and structure of individual programs and the Product, constitute confidential proprietary information of the Licensor or its suppliers and/or licensors. You agree not to transfer, copy, disclose, provide or otherwise make available such confidential information in any form to any third party. For purposes hereof, **"License Key"** or **"Registration Key"** shall mean a file or a unique sequence of digit and/or symbols provided to you by the Licensor confirming the purchase of the license from the Licensor, which may carry the information about the License, i.e. its type, the user name and the number of licenses purchased, and enabling the full functionality of the Product in accordance with the License granted under this Agreement. You agree to implement reasonable security measures to protect such confidential information. If you download the Software from the Internet or similar on-line source, you must include the copyright notices resident on the Software with any on-line distribution and on any media you distribute that includes the Software.

2. Grant of License.

2.1. License. Unless otherwise specifically indicated under a valid Terms of Use, the Licensor grants you a non-exclusive and non-transferable license without the right to sublicense (the **"License"**) and Licensee hereby accepts such License as follows, *provided* that unless otherwise agreed by Licensor or specified by Terms of Use, each License is granted per one Licensee:

a) Trial License. If you have received, downloaded and/or installed a Trial Version of the Product (the **"Trial Version"**) and are hereby granted a Trial License for the Software and you may Operate the Product only for evaluation and demonstration purposes and only during the single applicable evaluation period of thirty (30) days (the **"Trial Period"**), unless otherwise indicated, from the date of the initial installation. Any use of the Product for other purposes or beyond the applicable evaluation period is strictly prohibited, *provided however* that, subject to the restrictions contained herein, you may copy and distribute the Trial Version of the Software without any modifications whatsoever, and including this Agreement, to any third party. Licensee shall have no technical support rights during the Trial Period. The Licensor shall not be required to provide any support and Updates, as stated below, for the Trial Version of the Product. During the Trial Period, the Licensor provides no warranty whatsoever and assumes and bears no liability whatsoever for the Trial Version of the Product.

b) Personal Use License. If the Product is licensed under Personal License with Software provided in Object Code only upon the terms specified in the applicable registration, Terms of Use, invoicing or packaging for the Product, you personally may use of the Product solely for Personal Use (**"Personal License"**). **"Personal Use"** shall mean personal non-commercial, non-business, non-government Use, and not on behalf or for the benefit of any clients and excludes any commercial purposes whatsoever, which include without limitation: advertising marketing and promotional materials/services on behalf of an actual client, employer, employee or for your own benefit, any products that are commercially distributed, whether or not for a fee, any materials or services for sale or for which fees or charges are paid or received. Upon payment for the License and registration of the Product, you are granted a non-exclusive and non-transferable personal License to (i) install one (1) copy of the Product on up to three (3) Client Devices owned by you, and (ii) subject to the payment of the

applicable fees and your compliance with the terms hereof, to Use one (1) copy of the specified version of the Product during the Term of this Agreement, on one (1) of such Client Devices at any given time. Additionally, the individual licensing terms may specify other terms, conditions and restrictions of Operating the Product.

c) Site License. If the Product is licensed with site license terms specified in the applicable product invoicing or packaging for the Product, you may Operate and Use the Product on an unlimited number of Client Devices within a single building owned or leased by your company. Additionally, the individual licensing terms may specify other terms, conditions and restrictions of Using the Product (the "**Site License**").

d) Business License. If the Product is licensed under Business License with the Software provided in Object Code only upon the terms specified in the applicable Terms of Use, invoicing or packaging for the Product, you may Use of the Product solely for Business Use (the "**Business License**"). For purposes hereof, "**Business Use**" shall mean business, commercial, government Use only for internal business purposes of such entity without the right to distribute the applications, frameworks or components developed Using the Software (the "**Results**") to third parties *provided that* nothing herein shall be construed as creating any obligations of the Licensor to any end user of the Results. Under the Business License, any Use or Operating of the Product or transfer of the results of Using of the Product on a computer device owned by a third party is strictly prohibited unless a separate License is purchased therefor.

e) Commercial License. If the Product is licensed under Commercial License with the Software provided in Object Code only upon the terms specified in the applicable Terms of Use, invoicing or packaging for the Product, you may Use of the Product solely for Commercial Use (the "**Commercial License**"). For purposes hereof, "**Commercial Use**" shall mean business, commercial, government Use with the right to distribute the Results to third parties. Licensee may Use the Product licensed under the Commercial License on an unlimited number of web servers and domains owned, rented or leased by Licensee. Under the Commercial License, any Use or Operating of the Product or transfer of the results of Using of the Product on a computer device owned by a third party is strictly prohibited unless a separate License is purchased therefor.

f) Intranet License. If the Product is licensed with Intranet License terms specified in the applicable product invoicing or packaging for the Product, you may Operate and Use the Product on an unlimited number of Client Devices within a single local area network and/or private computer network under your control (the "**Intranet License**"). Additionally, the individual licensing terms may specify other terms, conditions and restrictions of Using the Product.

g) Internet License. If the Product is licensed with Internet License terms specified in the applicable product invoicing or packaging for the Product, you may Operate and Use the Product on an unlimited number of Client Devices on a single server accessible via internet (the "**Internet License**"). Additionally, the individual licensing terms may specify other terms, conditions and restrictions of Using the Product.

h) Compiled Units. If you are granted a Commercial License pursuant to Section 2.1(e) hereof, in addition to the licenses and rights granted therein, Licensor grants you a nonexclusive, deployment-free, royalty-free right to reproduce and distribute the Object Code version of those portions of the Software which are identified in the Documentation as '*compiled units*' (the "**Compiled Units**") *provided that* you comply with all of the following requirements:

- i). you distribute the Compiled Units in Object Code form only in conjunction with and as part of your software application product which adds significant and primary functionality and when the absence of Compiled Units will make your software application inoperable;
- ii). you do not use Licensor name, logo or trademarks to market your software application product; and
- iii). you include a valid copyright, trademark or any other proprietary notices on your Software identifying the Licensor as the owner of the Compiled Units.

i) Source Code License. The Licensor, pursuant to Terms of Use or invoicing terms, and in conjunction with one of Licenses granted under Section 2 hereof, may grant you certain rights to Software provided in Source Code as follows (the "**Source Code License**"):

i). For purposes hereof, "Source Code" shall mean the human-readable form of the computer programming code and related system documentation including all comments and any procedural code such as job control language. Provided you have purchased a license to a part of the Software supplied in Source Code form, you may make modifications, enhancements, derivative works and/or extensions to that licensed Source Code provided to you under the terms set forth in this Section 2.1(f).

ii). While the Licensor does not claim any ownership rights in the Results, in the event you develop any modifications, enhancements, derivative works and/or extensions to the licensed Source Code (the "**Derivatives**"), either independently or jointly with the Licensor, such Derivatives and all rights associated therewith will be the exclusive property of the Licensor.

iii). You shall not grant, either expressly or by implication, any rights, title, interest, or licenses to any Derivatives to any third party. You will, however, be entitled to use such Derivatives under the terms set forth in this Agreement. You hereby assign all right, title and interest in and to such Derivatives to the licensed Source Code to the Licensor.

iv). You also agree to execute, acknowledge and deliver to the Licensor all documents and instruments and do all things and actions Licensor deems necessary or desirable, at no cost to you but at Licensor's expense, to enable the Licensor to obtain and secure such Derivatives anywhere in the World. You agree to secure all necessary rights and obligations from relevant employees, or third parties in order to satisfy the above obligations. You may not distribute the Licensor's Source Code, or any Derivatives, in Source Code form.

v). Under no circumstances may any portion of the Source Code be distributed, disclosed or otherwise made available to any third party without the express, prior written consent of the Licensor. Under no circumstances may the Source Code be used in whole or in part, as the basis for creating a product that provides the same, or substantially the same, functionality as any of the Licensor's product. You will not take any action, or assist or otherwise aid anyone else in taking any action that would, in any way, limit the Licensor's independent development, sale, assignment, licensing or use of its Software or any Derivatives thereof. You will not modify or delete, in whole or part, any copyright, trade secret, proprietary, confidential or other notice thereon or therein, including a prominent notice on the Results "**Powered by Microolap Database Designer for PostgreSQL**" without the express, prior written consent of the Licensor.

vi). YOU UNDERSTAND, ACKNOWLEDGE AND AGREE THAT SOURCE CODE IS LICENSED "AS IS," AND THAT THE LICENSOR DOES NOT PROVIDE ANY TECHNICAL SUPPORT FOR SOURCE CODE.

j) Business License with Source Code License. If the Product is licensed under Business License with Source Code License upon the terms specified in the applicable Terms of Use, invoicing or packaging for the Product, your rights to Use the Product shall be the rights granted under the Business License together with the rights granted under the Source Code License (the "**Business License with Source Code License**").

k) Commercial License with Source Code License. If the Product is licensed under Commercial License with Source Code License upon the terms specified in the applicable Terms of Use, invoicing or packaging for the Product, your rights to may Use of the Product shall be the rights granted under the Commercial License together with the rights granted under the Source Code License (the "**Commercial License with Source Code License**").

l) Customized Licenses. The Licensor may grant you a specific customized terms of License pursuant to a valid Terms of Use signed by both parties hereto in which case such the terms and conditions of the Terms of Use shall be controlling and supersede any conflicting terms of this Agreement.

m) Educational Purpose License, Educational Institution Classroom License, and Educational Institution Site License. If the Product is licensed under an Educational Purpose License upon the terms specified in the applicable Educational Purpose License invoicing or packaging for the Product, you may make use of the Product solely for Educational Purpose. "**Educational Purpose**" means any non-commercial study or research that is undertaken solely in furtherance of one's education, whether or not completed by a student in pursuit of an educational degree, certificate or diploma and as used by teachers or facilitates teaching of a

class, and all administrative staff, faculty and employees, of any college, university, trade school or other school ("**Educational Institution**"). With the acquisition of an Educational Institution Classroom License, Licensee may install and Operate the Product by a number of Users determined by the applicable invoicing terms within one Educational Institution in one classroom. Within these limitations, you may install the Product as a "Network" Product and run the Product from any networked Client Devices on Licensee's LAN, provided that such Client Devices are located exclusively within one classroom. With the acquisition of an Educational Institution Site License, Licensee may install and Operate the Product by a number of Users determined by the applicable invoicing terms within one Educational Institution in one geographic location. Within these limitations, you may install the Product as a "Network" Product and run the Product from any networked Client Devices on Licensee's LAN, provided that such Client Devices are located exclusively within one office complex within one geographic location. Educational License may be granted exclusively at the discretion of the Licensor upon your submission of a written request discussing your and your employer/employees activities, when applicable, and your reasons for and purposes of Operating the Product.

2.2. Multiple Environment Product; Multiple Language Product; Dual Media Product; Multiple Copies; Bundles. If you use different versions of the Product or different language editions of the Product, if you receive the Product on multiple media, if you otherwise receive multiple copies of the Product, or if you received the Product bundled with other software, the total permitted number of your Client Devices on which all versions of the Product are installed shall correspond to the number of licenses you have obtained from the Licensor provided that unless the licensing terms and the License Key provides otherwise, each purchased license entitles you to install and Use the Product on one (1) Client Device. You may not rent, lease, sublicense, lend or transfer any versions or copies of the Product regardless of whether you use the Product or not, *provided that* the terms specified in the applicable product invoicing or packaging for the Product specify otherwise.

2.3. Run-time License. If the Product is licensed under Commercial License in accordance to Sections 2.1(e) or 2.1(k), in addition to the licenses and rights granted therein, Licensor grants Licensee a right to display, install, copy and distribute a portion of Licensor Software (the "**Run-time Edition**") on Client Devices of its customers as part of the Licensee's bundled products (the "**Run-time License**") in accordance with terms and conditions specified in the License Key, and/or invoicing terms, including the limitations relating to the number Client Devices of such customers or the number of such customers as the case may be, *provided that*, Licensee shall include a valid and prominent copyright, trademark or any other proprietary notices in its bundled products identifying the Licensor as the owner of the Run-time Edition as may be specified by Licensor from time to time.

2.4. Updates. During the Term of this Agreement, you may download Updates to the Product when and as the Licensor publishes them in its website or through other online services. Notwithstanding any provision to the contrary herein, nothing in this Agreement shall be construed as to grant you any rights or licenses with regard to the New Releases of the Product or to entitle you to any New Release. This Agreement does not obligate the Licensor to provide any Updates. Notwithstanding the foregoing, any Updates that you may receive become part of the Product and the terms of this Agreement apply to them (unless this Agreement is superseded by a further Agreement accompanying such Update or modified version of to the Product).

2.5. Material Terms and Conditions. You specifically agree that each of the terms and conditions of this Section 2 are material and that failure of you to comply with these terms and conditions shall constitute sufficient cause for Licensor to immediately terminate this Agreement and the License granted under this Agreement. The presence of this Section 2.5 shall not be relevant in determining the materiality of any other provision or breach of this Agreement by either party hereto.

3. Additional Covenants; Assignment of Intellectual Property Rights.

3.1. Additional Limitations. Notwithstanding anything to the contrary herein, you may not Operate, Use, or modify the Product in any way as to form the basis for creating a product that provides the same, or substantially the same, functionality as the Product; and in the event you develop any modifications, enhancements, derivative works and/or extensions to the Product, either independently or jointly with Licensor, such modifications, enhancements, derivative works and/or extensions and all rights associated therewith will be the exclusive property of Licensor. You will

not grant, either expressly or impliedly, any rights, title, interest, or licenses to any such modifications, enhancements, derivative works and/or extensions to any third party. You will, however, be entitled to use such modifications, enhancements, derivative works and/or extensions under the terms set forth in this Agreement. You hereby assign all right, title and interest in and to such modifications, enhancements, derivative works and/or extensions to the Product to Licensor. You also agree to execute, acknowledge and deliver to Licensor all documents and do all things Licensor deems necessary or desirable, at no cost to but at Licensor expense, to enable Licensor to obtain and secure such modifications, enhancements, derivative works and/or extensions anywhere in the world. You agree to secure all necessary rights and obligations from relevant employees or third parties in order to satisfy the above obligations.

3.2. Reservations of all Rights. The Licensor reserves all rights not expressly granted herein.

3.3. Back-up Copies. You can make one (1) copy of the Product for backup and archival purposes, *provided, however*, that the original and each copy is kept in your possession or control, and that your installation, Operation and Use of the Product does not exceed that which is allowed in Section 2 hereof.

3.4. Additional Protection Measures. Solely for the purpose of preventing unlicensed and/or unauthorized Use of the Product, including without limitation Run-time Edition, the Software may collect certain non-personal information relating to the hardware of your or your customer's Client Devices and/or install on your or your customer's Client Devices certain technological measures that are designed to prevent unlicensed and/or unauthorized Use and Operation of the Product, and the Licensor may use this technology to confirm that you have a licensed copy of the Product. Such installation or collection of information or updates of these technological measures may occur through and/or during the installation or activation of the Product or Updates. The Product and/or Updates will not install or may fail to Operate if installed contrary to the rights granted under the License or if attempted to be installed or Operated on unlicensed copies of the Product. If you are not using a licensed copy of the Product, you are not allowed to install the Updates. The Licensor will not collect any personally identifiable information from your computer during this process.

4. Term and Termination.

4.1. Term. The term of this Agreement ("**Term**") shall begin when you download, access or install the Product, whichever is earlier, and shall continue in perpetuity unless otherwise designated in the purchase order, Terms of Use, exhibit or unless otherwise terminated pursuant hereto. Without prejudice to any other rights, this Agreement will terminate automatically if you fail to comply with any of the limitations or other requirements described herein. Upon any termination or expiration of this Agreement, you must immediately cease Operating the Product and all of its components and destroy, uninstall and erase all copies of the Product and all of its components, including without limitation on all systems and all types of media and in computer memory.

4.2. No Rights upon Termination. Upon termination of this Agreement you will no longer be authorized to Operate or Use the Product in any way.

5. Support and Updates.

5.1. Terms of Support. During the Warranty Period as defined below, you are entitled to technical services and support for the Product which is provided to you by Licensor free of charge during the regular business hours (GMT+3), except for locally-observed holidays, and includes the support provided through a special technical support section of the Licensor Site (<http://www.microolap.com/support/>) ("**Warranty Support**"). During the Warranty Period, Warranty Support is unlimited and includes technical and support questions and patch fixes. After the expiration of the Warranty Period, you may purchase additional support services from Licensor at current rates as listed at Licensor Site (<http://www.microolap.com/support/>).

5.2. Updates. During the Warranty Period hereunder and, if your purchasing terms provide for a certain extended period of time during which you are entitled to Updates free of charge (i.e., one year from the time of the purchase of the License), then for such extended period of time, you may download Updates to the Product when and as the Licensor publishes them on the Licensor Site, or through other online services. If the Product is an Update to a previous version of the Product, you must possess a valid license to such previous version in order to use the Update. You may continue to use the previous version of the Product on your Client Device after you receive the Update to assist you in the transition to the Update, provided that: (i) the Update and

the previous version are installed on the same computer device; (ii) the previous version or copies thereof are not transferred to another party unless all copies of the Update are also transferred to such party; and (iii) you acknowledge that any obligation the Licensor may have to support the previous version of the Product may be ended upon availability of the Update. Except for the rights to free Updates during the Warranty Period, as further defined herein, nothing in this Agreement shall be construed as to grant you any rights or licenses with regard to the new version or releases of the Product or to entitle you to any new version, upgrade or release. This Agreement does not obligate the Licensor to provide any Updates. Notwithstanding the foregoing, any Updates that you may receive become part of the Product and the terms of this Agreement apply to them (unless this Agreement is superseded by a succeeding agreement accompanying such Update or modified version of the Product). Notwithstanding anything to the contrary herein, nothing herein shall be deemed to entitle you to any Licensor's Optional Updates that are provided by Licensor for a fee, or to any new releases, versions or substitutes of the Product.

5.3. Additional Support and Updates. In addition to the free Support and free Updates provided for in [Sections 5.1](#) and [5.2](#), you may purchase additional Services, additional Updates or additional Support beyond the applicable period at the ongoing rates and prices published or provided by Licensor.

6. Restrictions.

6.1. No Transfer of Rights. Except as otherwise specifically provided herein or in the applicable Terms of Use, you may not transfer any rights pursuant to this Agreement nor rent, sublicense, lease, loan or resell the Product or permanently or temporarily transfer the Product in any other manner. You may not permit third parties, including any subcontractors, to benefit from the use or functionality of the Product including, without limitations, via a timesharing, service bureau or other arrangement, except to the extent such use is specified in the application price list, purchase order or product packaging for the Product. Except as otherwise provided in [Section 1.2](#) hereof, you may not, without the Licensor's prior written consent, reverse engineer, decompile, disassemble or otherwise reduce any part of the Product to human readable form nor permit any third party to do so, except to the extent the foregoing restriction is expressly prohibited by applicable law. Notwithstanding the foregoing sentence, decompiling the Software is permitted to the extent the laws of your jurisdiction expressly give you the non-waivable right to do so to obtain information necessary to render the Software interoperable with other software; provided, however, that you must first request such information from the Licensor and the Licensor may, in its discretion, either provide such information to you (subject to confidentiality terms) or impose reasonable conditions, including a reasonable fee, on such use of the Software to ensure that the Licensor's and its affiliates' proprietary rights in the Software are protected. Except for the modification permitted under [Section 1.2](#), you may not modify, or create derivative works based upon the Product in whole or in part.

6.2. No Extraction for Separate Use. You shall not have the right to extract or to Use any functionality of this Software, including any DLLs or other compiled units or Object Code fragments other than as part of normal Operation of the Product described in the Documentation and as integral part of Operation and functionality of the Product as a whole.

6.3. Proprietary Notices and Copies. You may not remove any proprietary notices or labels on the Product. You may not copy the Product except as expressly permitted in [Section 2](#) above.

6.4. Compliance with Law. You agree that in Operating the Product and in using any report or information derived as a result of Operating this Product, you will comply with all applicable international, national, state, regional and local laws and regulations, including, without limitation, privacy, trademark, patent, copyright, export control and obscenity law and you shall not use the Product for unethical or illegal business practices or in violation of any obligation to a third party in using, Operating, accessing or running any of the Product and shall not knowingly assist any other person or entity to so violate any obligation to a third party.

7. WARRANTIES AND DISCLAIMERS.

7.1. Limited Warranty. The Licensor warrants that for two (2) months (the "**Warranty Period**") from the date the Product has been downloaded by you or was made otherwise available to you by Licensor if the Product was supplied to you on other media, the media on which Product has been provided will be free from defects in materials and workmanship and that the Software will perform substantially in accordance with the Documentation or generally conform to the Product's

specifications published by the Licensor. Non-substantial variations of performance from the Documentation do not establish a warranty right. THIS LIMITED WARRANTY DOES NOT APPLY TO UPDATES AS APPLIED TO ANY MODIFIED PRODUCT, WHETHER OR NOT SUCH MODIFICATION IS PERMISSIBLE HEREUNDER, TRIAL AND DEMO OR EVALUATION VERSIONS, UPDATES, PRE-RELEASE, TRYOUT, PRODUCT SAMPLER, OR NOT FOR RESALE (NFR) COPIES OF PRODUCT. This limited warranty is void and your support right terminates if the defect or damage has resulted from accident, abuse, or misapplication or any modification, whether or not such modification is permitted hereunder. Notwithstanding anything to the contrary herein, any and all modifications shall be deemed derivative works within the meaning of copyright law and remain subject to the terms of this Agreement, including without limitation [Section 1.1](#) hereof, *provided however*, that any damage or claims relating to or arising out of any modifications made by Licensee, whether or not permitted hereunder, shall be the sole responsibility of Licensee. To make a warranty claim, you must return the Product to the location where you obtained it along with proof of purchase within such sixty (60) day period of the license fee you paid for the Product. THE LIMITED WARRANTY SET FORTH IN THIS SECTION GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE ADDITIONAL RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

7.2. Customer Remedies. The Licensor and its suppliers' entire liability and your exclusive remedy for any breach of the foregoing warranty shall be at the Licensor's option: (i) return of the purchase price paid for the License, if any, (ii) replacement of the defective media in which the Product is contained, or (iii) correction of the defects, "bugs" or errors within reasonable period of time. You must return the defective media to the place of purchase at your expense with a copy of your receipt. Any replacement media will be warranted for the remainder of the original Warranty Period.

7.3. NO OTHER WARRANTIES. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT TO WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, THE PRODUCT IS PROVIDED "AS-IS" WITHOUT ANY WARRANTY WHATSOEVER AND THE LICENSOR MAKES NO PROMISES, REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESSED OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE, REGARDING OR RELATING TO THE PRODUCT OR CONTENT THEREIN OR TO ANY OTHER MATERIAL FURNISHED OR PROVIDED TO YOU PURSUANT TO THIS AGREEMENT OR OTHERWISE. YOU ASSUME ALL RISKS AND RESPONSIBILITIES FOR SELECTION OF THE PRODUCT TO ACHIEVE YOUR INTENDED RESULTS, AND FOR THE INSTALLATION OF, USE OF, AND RESULTS OBTAINED FROM THE PRODUCT. THE LICENSOR MAKES NO WARRANTY THAT THE PRODUCT WILL BE ERROR FREE OR FREE FROM INTERRUPTION OR FAILURE, OR THAT IT IS COMPATIBLE WITH ANY PARTICULAR HARDWARE OR SOFTWARE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, INTEGRATION, SATISFACTORY QUALITY OR FITNESS FOR ANY PARTICULAR PURPOSE WITH RESPECT TO THE PRODUCT AND THE ACCOMPANYING WRITTEN MATERIALS OR THE USE THEREOF. SOME JURISDICTIONS DO NOT ALLOW LIMITATIONS ON IMPLIED WARRANTIES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU. YOU HEREBY ACKNOWLEDGE THAT THE PRODUCT MAY NOT BE OR BECOME AVAILABLE DUE TO ANY NUMBER OF FACTORS INCLUDING WITHOUT LIMITATION PERIODIC SYSTEM MAINTENANCE, SCHEDULED OR UNSCHEDULED, ACTS OF GOD, TECHNICAL FAILURE OF THE SOFTWARE, TELECOMMUNICATIONS INFRASTRUCTURE, OR DELAY OR DISRUPTION ATTRIBUTABLE TO VIRUSES, DENIAL OF SERVICE ATTACKS, INCREASED OR FLUCTUATING DEMAND, AND ACTIONS AND OMISSIONS OF THIRD PARTIES. THEREFORE, THE LICENSOR EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY REGARDING SYSTEM AND/OR SOFTWARE AVAILABILITY, ACCESSIBILITY, OR PERFORMANCE. THE LICENSOR DISCLAIMS ANY AND ALL LIABILITY FOR THE LOSS OF DATA DURING ANY COMMUNICATIONS AND ANY LIABILITY ARISING FROM OR RELATED TO ANY FAILURE BY THE LICENSOR TO TRANSMIT ACCURATE OR COMPLETE INFORMATION TO YOU.

7.4. LIMITED LIABILITY; NO LIABILITY FOR CONSEQUENTIAL DAMAGES. YOU ASSUME THE ENTIRE COST OF ANY DAMAGE RESULTING FROM YOUR USE OF THE PRODUCT AND THE INFORMATION CONTAINED IN OR COMPILED BY THE PRODUCT, AND THE INTERACTION (OR FAILURE TO INTERACT PROPERLY) WITH ANY OTHER HARDWARE OR SOFTWARE WHETHER PROVIDED BY THE LICENSOR OR A THIRD PARTY. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT WILL THE LICENSOR OR ITS SUPPLIERS OR LICENSORS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF DATA, LOSS OF GOODWILL, WORK STOPPAGE, HARDWARE OR SOFTWARE DISRUPTION IMPAIRMENT OR FAILURE, REPAIR COSTS, TIME VALUE OR OTHER

PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT, OR THE INCOMPATIBILITY OF THE PRODUCT WITH ANY HARDWARE SOFTWARE OR USAGE, EVEN IF SUCH PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL LICENSOR'S TOTAL LIABILITY TO YOU FOR ALL DAMAGES IN ANY ONE OR MORE CAUSE OF ACTION, WHETHER IN CONTRACT, TORT OR OTHERWISE EXCEED THE AMOUNT PAID BY YOU FOR THE PRODUCT. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT THAT APPLICABLE LAW PROHIBITS SUCH LIMITATION. FURTHERMORE, BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. Indemnification

8.1. Indemnification for Violations. In Operating the Product, you agree to use only those materials for which you have the necessary patent, copyright and other permissions, licenses, and/or clearances. You agree to indemnify, defend and hold harmless the Licensor and its respective officers, directors, employees, agents, successors, and assigns (the "**Licensor Indemnitees**") from any and all losses, liabilities, damages and claims, and all related expenses including without limitation reasonable legal fees and disbursements and costs of investigation, litigation, settlement, judgment, interest and penalties and costs related to, arising from, or in connection with any third-party claim related to, arising from, or in connection with the actual or alleged: (i) infringement by you or by Compiled Units (except when such breach is exclusively attributable to Product) of any third-party intellectual property and/or proprietary right, including, but not limited to, patent, trademark, copyright, trade secret, publicity and/or privacy, (ii) personal injury (including death) or property damage due to gross negligence or intentional misconduct of the Licensee, and (iii) breach by the Licensee of any of its representations, warranties, obligations, and/or covenants set forth herein. You shall promptly notify the Licensor in writing after you become aware of any such claims, but failure to give such notice shall not relieve you of indemnity obligations hereunder. You shall have exclusive control over the settlement or defense of such claims or actions, except that Licensor may appear in the action, at its own expense, through counsel reasonably acceptable to you, only in the event it is mutually determined by the parties that an actual conflict of interest would exist by your representation of the Licensor and you in such action. Licensor shall give you, at your expense, all information and assistance reasonably requested by you to settle or defend such claims or actions. You shall be entitled to retain all monetary proceeds, attorneys' fees, costs and other rewards you receive as a result of defending or settling such claims. In the event you fail to promptly indemnify and defend such claims and/or pay Licensor's expenses, as provided above, Licensor shall have the right to defend itself, and in that case, you shall reimburse the Licensor Indemnitees for all of their attorneys' fees, costs and damages incurred in settling or defending such claims within thirty (30) days of each of Licensor's written requests. Nothing in this Section 3.2 or this Agreement shall be interpreted as to exclude any possible legal recourse against the Licensee.

9. U.S. Government-Restricted Rights.

9.1. Notice to U.S. Government End Users. The Product and accompanying Documentation are deemed to be "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," respectively, as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights, including any use, modification, reproduction, release, performance, display or disclosure of the Product and accompanying Documentation, as are granted to all other end users pursuant to the terms and conditions herein. Unpublished rights are reserved under the copyright laws of the United States.

9.2. Export Restrictions. You acknowledge and agree that the Product may be subject to restrictions and controls imposed by the Export Administration Act and the Export Administration Regulations of the United States (the "**Acts**"). You agree and certify that neither the Product nor any direct product thereof is being or will be used for any purpose prohibited by the Acts. You may not Operate, download, export, or re-export the Product (a) into, or to a national or resident of, any country to which the United States has embargoed goods, or (b) to anyone on the United States Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Deny Orders. By downloading or using the Product, you are representing

and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list. You acknowledge that it is your sole responsibility to comply with any and all government export and other applicable laws and that the Licensor has no further responsibility for such after the initial license to you. You warrant and represent that neither the U.S. Commerce Department, Bureau of Export Administration nor any other U.S. federal agency has suspended, revoked or denied your export privileges. For more information on the U.S. Export Administration Regulations (EAR), 15 C.F.R. Parts 730-774, and the Bureau of Export Administration ("**BXA**"), please see the BXA homepage (<http://www.bxa.doc.gov>).

10. Your Information and the Licensor's Privacy Policy

10.1. Privacy Policy. You acknowledge receipt of and agree to the Licensor's privacy statement which is made available to you in connection with installation and is set forth in full at <http://www.microolap/about/privacy/>. You hereby expressly consent to the Licensor's processing of your personal data (which may be collected by the Licensor or its distributors) according to the Licensor's current privacy policy as of the date of the effectiveness hereof which is incorporated into this Agreement by reference. By entering into this Agreement, you agree that the Licensor may collect and retain information about you, including your name, email address and credit card information. The Licensor may employ other companies and individuals to perform certain functions on its behalf. Examples include fulfilling orders, delivering packages, sending postal mail and e-mail, removing repetitive information from customer lists, analyzing data, providing marketing assistance, processing credit card payments, implementing fraud check policies, and providing customer service. Such companies and individuals may have access to personal information needed to perform their functions, but may not use it for other purposes. The Licensor publishes a privacy policy on the Licensor Site and may amend such policy from time to time in its sole discretion. You should refer to the Licensor's privacy policy prior to agreeing to this Agreement for a more detailed explanation of how your information will be stored and used by the Licensor. If "you" are an organization, you will ensure that each member of your organization (including employees and contractors) about whom personal data may be provided to the Licensor has given his or her express consent to the Licensor's processing of such personal data. Personal data will be processed by the Licensor or its distributors in the country where it was collected. The relevant laws in such jurisdictions regarding processing of personal data may be less or more stringent than the laws in your jurisdiction.

10.2. Public Announcements. The Licensor may identify you to the public as a customer of the Licensor and describe in a customer case study the services and solutions delivered by the Licensor to you. The Licensor may also issue one or more press releases, containing an announcement of the execution and delivery of this Agreement and/or the implementation of the Product by you. Nothing contained in this Section 10.2 shall be construed as an obligation by you to disclose any of your proprietary or confidential information to any third party. In addition, you may opt-out from this Section 10.2 by writing an opt-out request to the Licensor at sales@microolap.com.

11. Miscellaneous.

11.1. Governing Law ; Jurisdiction and Venue. This Agreement shall be governed by and construed and enforced in accordance with the laws of the British Virgin Islands without reference to conflicts of law rules and principles. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly disclaimed and excluded. The courts within the British Virgin Islands shall have exclusive jurisdiction to adjudicate any dispute arising out of this Agreement. You agree that this Agreement and any action, dispute, controversy, or claim that may be instituted based on this Agreement, or arising out of or related to this Agreement or any alleged breach thereof, shall be prosecuted exclusively in the courts of the British Virgin Islands and you, to the extent permitted by applicable law, hereby waive the right to change venue to any other state, county, district or jurisdiction; *provided, however*, that the Licensor as claimant shall be entitled to initiate proceedings in any court of competent jurisdiction.

11.2. Period for Bringing Actions. No action, regardless of form, arising out of the transactions under this Agreement, may be brought by either party hereto more than one (1) year after the cause of action has occurred, or was discovered to have occurred, except that an action for infringement of intellectual property rights may be brought within the maximum applicable statutory period.

11.3. Entire Agreement; Severability; No Waiver. This Agreement is the entire agreement between you and Licensor and supersedes any other prior agreements, proposals, communications or advertising, oral or written, with respect to the Product or to subject matter of this Agreement. You acknowledge that you have read this Agreement, understand it and agree to be bound by its terms. If any provision of this Agreement is found by a court of competent jurisdiction to be invalid, void, or unenforceable for any reason, in whole or in part, such provision will be more narrowly construed so that it becomes legal and enforceable, and the entire Agreement will not fail on account thereof and the balance of the Agreement will continue in full force and effect to the maximum extent permitted by law or equity while preserving, to the fullest extent possible, its original intent. No waiver of any provision or condition herein shall be valid unless in writing and signed by you and an authorized representative of Licensor provided that no waiver of any breach of any provisions of this Agreement will constitute a waiver of any prior, concurrent or subsequent breach. Licensor's failure to insist upon or enforce strict performance of any provision of this Agreement or any right shall not be construed as a waiver of any such provision or right.

11.4. Contact Information. Should you have any questions concerning this Agreement, or if you desire to contact the Licensor for any reason, please contact our Customer Department at <http://microolap.com/support/>.

© 1999-2018, Microolap Technologies. All rights reserved. The Product, including the Software and any accompanying Documentation, are copyrighted and protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.