



PostgreSQL Maestro

User's guide

Table of Contents

Foreword	0
I Welcome to PostgreSQL Maestro!	1
1 System Requirements	3
2 Installation	4
3 How can I purchase PostgreSQL Maestro?	5
4 License Agreement	6
5 About SQL Maestro Group	8
6 What's new	11
II Getting Started	12
1 Connect to a database	13
2 Connection parameters	14
3 Explaining user interface	18
First time started	19
Tabbed MDI	20
Switching between windows	22
4 Shortcut keys	23
III Databases and Database Profiles	24
1 Creating Database Profiles	26
Setting connection properties	26
Setting profile options	27
2 Editing Database Profile	30
Editing connection properties	30
Setting profile options	31
Setting default directories	33
Editing obligatory scripts to execute	34
Setting log options	34
Statistics	35
3 Create Database Wizard	37
Setting connection properties	37
Specifying database properties	38
4 Database Editor	40
IV Database Object Management	42
1 Create Objects	43
Create Database Object Dialog	43
Overview of Create Objects Wizards	43
Setting object name	44
Viewing common information	45
2 Edit Objects	46
Overview of Object Editors	46

Permissions of the Object.....	47
Object grants.....	48
Object Dependencies.....	49
Types of dependencies.....	49
SQL Definition.....	50
Parameter Editor.....	50
Executing functions and procedures.....	51
Modify Object Properties	52
Describe Objects	52
3 Duplicate Objects	53
Duplicate Object Wizard	53
Selecting source and destination databases.....	53
Selecting object to duplicate.....	54
Modifying new object definition.....	55
Duplicate Selected Object	55
Copy, Paste and Drag-n-Drop features	56
4 Browse Objects	58
Database Explorer	58
Filtering explorer content.....	61
Object Browser	61
Object Manager	63
Filter Builder Dialog	64

V Database Objects 66

1 Schemas	67
Create Schema Wizard	68
Schema Editor	70
2 Tables	73
Create Table Wizard	74
Table Editor	77
Editing table properties.....	77
Managing table data.....	79
Fields	82
Indexes	84
Foreign Keys	88
Checks	91
NOT NULL constraints	93
Triggers	95
Create Trigger Wizard.....	97
Trigger Editor.....	98
Rules	100
Create Rule Wizard.....	101
Rule Editor.....	103
Foreign Key References	105
Row Security Policies	106
3 Views	108
Create View Wizard	109
View Editor	114
Editing view properties.....	115
Viewing data.....	116
4 Materialized Vews	118
Create Materialized Views Wizard	119

Materialized Views Editor	123
5 Functions	125
Create Function Wizard	126
Function Editor	128
Editing properties	129
Viewing Function results.....	131
6 Domains	133
Create Domain Wizard	134
Domain Editor	136
7 Aggregates	139
Create Aggregate Wizard	140
Aggregate Editor	141
Editing aggregate properties.....	142
8 Sequences	144
Create Sequence Wizard	145
Sequence Editor	146
9 Types	148
Create Type Wizard	149
Type Editor	151
Editing type options.....	151
10 Composite and Enum Types	154
Create Type Wizard (Composite and Enum Types)	155
Type Editor (Composite and Enum Types)	157
Composite and Enum type properties	158
11 Range Types	159
Create Range Type Wizard	160
Range Type Editor	161
12 Operators	162
Create Operator Wizard	163
Operator Editor	164
Editing operator propties	165
13 Collations	167
14 Languages	168
Create Language Wizard	169
Language Editor	171
Editing language properties	171
15 Casts	172
Create Cast Wizard	173
Cast Editor	175
Editing cast properties	175
16 Database Variables	177
Database variable properties	178
17 Extensions	180
18 Foreign Tables	182
19 Foreign Servers	185
User Mappings	186

VI Server Objects

187

1 Server Editor	189
2 Databases	190
3 Server Variables	192
Variable Editor	192
4 Users	194
Create User Wizard	195
User Editor	196
5 Groups	198
Create Group Wizard	199
Group Editor	199
6 Roles	200
Create Role Wizard	201
Specifying role properties	202
Membership and environment.....	203
Role Editor	204
Editing role properties	205
Viewing role objects.....	207
7 Tablespaces	208
Create Tablespace Wizard	209
Tablespace Editor	210
Editing tablespace properties	210

VII Queries 212

1 SQL Editor	214
SQL Formatter	215
Executing query	216
Query Parameters	218
2 Visual Query Builder	219
Creating query diagram	220
Working with editor area	225
Executing query	226

VIII Data Management 228

1 Data View	229
Working with data grid	230
Working with info cards	235
Data input form	236
Data filtering	237
2 BLOB Editor	240
Editing as image	240
Editing as hexadecimal dump	241
Editing as plain text	241
Editing as HTML	242
Editing as PDF document	243
3 Export Data Wizard	245
Setting destination file name and format	245
Setting header and footer	246
Selecting fields for export	247
Adjusting data formats	247

Setting format-specific options	248
Setting common export options	251
4 Get SQL Dump	252
Selecting fields	252
Specifying dump options	253
5 Import Data Wizard	255
Setting source file name and format	256
Setting the accordance between source and target columns	258
Map builder	259
Data formats	260
Customizing common options	261

IX Database Tools

263

1 Script Runner	265
2 SQL Script Editor	266
3 Extract Database	269
Selecting the database and the target file name	269
Selecting objects to extract their structure	270
Selecting objects to extract their data	271
Customizing script options	272
4 Generate Database Report	274
Selecting reporting elements and setting other report options	274
Reporting objects and editing styles options	274
Editing database report style	275
5 BLOB Viewer	276
Viewing as hexadecimal dump	276
Viewing as plain text	277
Viewing as image	278
Viewing as HTML	279
Viewing as PDF	280
6 Diagram Viewer	282
Customizing diagram properties	283
Exporting diagram image	284
7 Data Analysis	286
Input SELECT query	287
Managing report data	288
8 Report Designer	291
Designer Tools and Objects	293
Object Inspector	295
9 Schema Designer	298
Designer Navigation Bar	300
Schema Designer Toolbox	300
10 PL/pgSQL Debugger	303
Debugging process	304
Debug information	306
11 Process Browser	307
12 Dependency tracker	308
13 SQL Generator	309

14 DML procedures generation	310
15 Generation of updatable views	312
16 Split table	313
17 Nullable Column Checker	315
18 Dialogs	317
Find Text dialog	317
Replace Text dialog	318

X Options 321

1 Application	322
Preferences	322
Confirmations	323
Directories	325
Tools	325
Explorer	327
Object Manager	328
SQL Editor	328
SQL Script Editor	329
Query Builder	330
Colors	332
BLOB Viewer	332
Data Export	333
Database Designer	334
Object Editors	335
Table	337
Data Grid	337
Options	339
Colors	340
Formats	341
Filter	342
2 Editors & Viewers	344
General	344
Display	345
SQL highlight	346
XML highlight	347
PHP highlight	348
Code Insight	349
Code Folding	350
3 Appearance	352
Bars and menus	352
Trees and lists	353
Edit controls	354
Check boxes	355
Buttons	356
Page controls	357
Group boxes	358
Splitters	359
4 Export Settings	361
Specifying destination file	361
Selecting setting categories	361

Selecting database profiles	362
Saving settings	362

Index	364
--------------	------------

1 Welcome to PostgreSQL Maestro!

PostgreSQL Maestro is the premier Windows GUI admin tool for PostgreSQL development and management. It allows you to make all the database operations easy and fast.

Basic PostgreSQL Maestro features

Support of the latest PostgreSQL features

Use PostgreSQL Maestro to work with PostgreSQL from 7.3 to 18. Among other features and objects implemented in the latest versions of the server, our software supports parameter scopes, connection limits, ascending and descending indexes, multirange data types and virtual generated columns, and a lot of other useful things.

Easy database management

PostgreSQL Maestro allows you to work with [remote servers](#)^[14] with restricted access via SSH or HTTP tunneling. PostgreSQL Maestro allows you to create new databases and drop existing ones. Database profiles give you the opportunity to connect to databases in one touch and work with the selected databases only. See the [Database Management](#)^[24] for details.

Powerful database object management

PostgreSQL Maestro provides you with an ability to manage database objects in various ways. For example, you can perform operations with a group of objects as well as with a single object in [Object Manager](#)^[63], sort, group and filter the database objects within [Object Browser](#)^[61], copy an object from one database to another by a drag-and-drop operation inside the explorer tree, use Windows clipboard to copy a set of objects and so on. For details turn to the [Database Object Management](#)^[66] section.

Working with tables and table subobjects

PostgreSQL Maestro wizards and editors allow you to create, edit and drop tables as well as their *fields*, *indexes*, and *foreign keys* in a couple of simple operations. See the [Tables](#)^[73] section to learn more.

Building and executing queries

PostgreSQL Maestro provides two powerful tools which allow you either to edit query text directly with syntax highlighting and code completion or to build a query diagram visually selecting tables and fields, setting links between tables and so on. You can find the detailed description in the [Queries](#)^[212] section.

Powerful data management tools

PostgreSQL Maestro puts at your disposal a complete set of data management tools with viewing, editing, grouping, sorting and filtering abilities, lookup editors, master-detail data view, BLOB Viewer/Editor, data export, data import and SQL dump modules and more. See the [Data Management](#)^[228] to learn the details.

Wide choice of additional tools

PostgreSQL Maestro provides you with a number of tools for working with database metadata and SQL scripts, including Script Runner, SQL Script Editor with code folding and script explorer. Moreover, it gives such tools as Schema Designer, BLOB Viewer, Diagram Viewer, Data Analysis, Dependency Tracker, SQL Generator, Report Designer, and a lot of others. To learn more, see the [Database Tools](#)^[263] section.

Security management

PostgreSQL Maestro gives you a comfortable access to PostgreSQL security features.

Easy PostgreSQL run-time configuration

PostgreSQL Maestro helps you to assign a set of option settings either to a user or to a database in a few mouse clicks. Overriding of the default server settings becomes easy too.

Full customization according to your preferences and needs

In PostgreSQL Maestro you can customize the behavior of all its tools, select a user interface scheme and set a lot of other preferences. All the options and their meanings are listed within the [Options](#)^[32] dialog description.

1.1 System Requirements

Client environment

- Pentium PC or higher;
- Windows NT4/2000/XP/Vista/Windows 7/Windows 8/Windows 10/Windows 11;
- 512 MB RAM (1 GB recommended);
- 25 MB of free hard disk space;
- SVGA-compatible video adapter.

Server environment

- PostgreSQL from 7.3 to 18.

1.2 Installation

To install **PostgreSQL Maestro** for the first time on your PC:

- download the PostgreSQL Maestro distribution package from the [download page](#) at our site;
- run setup.exe from the local folder and follow the instructions of the installation wizard;
- find the PostgreSQL Maestro shortcut in the corresponding program group of the Windows Start menu after the installation is completed.

To upgrade the installed copy of PostgreSQL Maestro to the latest version:

- download the PostgreSQL Maestro executable file from the [download page](#) at our site;
- unzip downloaded file to any local folder, e.g. *c:\unzipped*;
- exit from PostgreSQL Maestro if it is running;
- replace previous version of PostgreSQL Maestro by copying unzipped files to the PostgreSQL Maestro folder;
- run PostgreSQL Maestro using its shortcut in the Windows Start menu.

You can also use the full distribution package to upgrade your current version of PostgreSQL Maestro. In this case you should repeat the steps of the first-time installation. Note that the full distribution package is larger than a single executable file.

1.3 How can I purchase PostgreSQL Maestro?

Thank you for your interest in purchasing **PostgreSQL Maestro**!

You can select licensing options and register PostgreSQL Maestro at its [on-line order page](#). It is possible to purchase on-line, by fax, mail, toll-free phone call, or place a purchase order. We send the software activation key by email within 24 hours after completion of the order process. If you have not received the activation key within this period, please contact our [sales department](#).

All our products and bundles are shipped with 12 months of free upgrades (minor and major ones) or with 36 months of free upgrades for a quite small additional fee. After this period you may renew your license for the next 12(36) months with a 50% discount.

PostgreSQL Maestro has a free 30-day trial. Upon purchasing the product you confirm that you have tested it and you are completely satisfied with its current version.

To obtain technical support, please visit the [appropriate section](#) on our website or contact us by email to support@sqlmaestro.com.

1.4 License Agreement

Notice to users: carefully read the following legal agreement. The use of the software provided with this agreement (the "SOFTWARE") constitutes your acceptance of these terms. If you do not agree to the terms of this agreement, do not install and/or use this software. The use of this software is conditioned upon the user's compliance with the terms of this agreement.

- **License grant.** SQL Maestro Group grants you a license to use one copy of the version of this SOFTWARE on any single hardware product for as many licenses as you purchase. "You" means a company, an entity or an individual. "Use" means storing, loading, installing, executing or displaying the SOFTWARE. You may not modify the SOFTWARE or disable any licensing or control features of the SOFTWARE except as an intended part of the SOFTWARE's programming features. This license is not transferable to any other company, entity or individual. You may not publish any registration information (serial numbers, registration keys, etc.) or pass it to any other company, entity or individual.
- **Ownership.** The SOFTWARE is owned and copyrighted by [SQL Maestro Group](#). Your license confers no title or ownership of the SOFTWARE and should not be construed as a sale of any rights for the SOFTWARE.
- **Copyright.** The SOFTWARE is protected by the United States copyright law and international treaty provisions. You acknowledge that no title to the intellectual property in the SOFTWARE is transferred to you. You further acknowledge that title and full ownership rights to the SOFTWARE will remain the exclusive property of [SQL Maestro Group](#) and you will not acquire any rights to the SOFTWARE except as expressly set forth in this license. You agree that any copies of the SOFTWARE will contain the same proprietary notices which appear on and in the SOFTWARE.
- **License and distribution.** An unregistered copy of the SOFTWARE ("UNREGISTERED SOFTWARE") may be used for evaluation purposes. The UNREGISTERED SOFTWARE may be freely copied and distributed to other users for their evaluation. If you offer this UNREGISTERED SOFTWARE installation package for download, then you agree to:
 - replace existing version of the UNREGISTERED SOFTWARE installation package with the new package immediately after a new version of the SOFTWARE is released by SQL Maestro Group, or
 - delete an obsolete version of the UNREGISTERED SOFTWARE installation package immediately upon written email notice by SQL Maestro Group.

A registered copy of the SOFTWARE ("REGISTERED SOFTWARE") allows you to use the SOFTWARE only on a single computer and only by a single user at a time. If you wish to use the SOFTWARE for more than one user, you will need a separate license for each individual user. You are allowed to make one copy of the REGISTERED SOFTWARE for back-up purposes.

- **Reverse engineering.** You affirm that you will not attempt to reverse compile, modify, translate, or disassemble the SOFTWARE in whole or in part.
- **Unauthorized use.** You may not use, copy, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer the SOFTWARE except as provided in this agreement. Any such unauthorized use shall result in immediate and

automatic termination of this license.

- **No other warranties.** [SQL Maestro Group](#) does not warrant that the SOFTWARE is error-free. SQL Maestro Group disclaims all other warranties with respect to the SOFTWARE, either express or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose and noninfringement of third party rights. Some jurisdictions do not allow the exclusion of implied warranties or limitations on how long an implied warranty may last, or the exclusion or limitation of incidental or consequential damages, so the above given limitations or exclusions may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from jurisdiction to jurisdiction.
- **Limited warranty.** This SOFTWARE is provided on an "AS IS" basis. [SQL Maestro Group](#) disclaims all warranties relating to this SOFTWARE, whether expressed or implied, including but not limited to any implied warranties of merchantability or fitness for a particular purpose. Neither SQL Maestro Group nor anyone else who has been involved in the creation, production, or delivery of this SOFTWARE shall be liable for any indirect, consequential, or incidental damages arising out of the use or inability to use such SOFTWARE, even if SQL Maestro Group has been advised of the possibility of such damages or claims. The person using the SOFTWARE bears all risk as to the quality and performance of the SOFTWARE.

Some jurisdictions do not allow limitation or exclusion of incidental or consequential damages, so the above given limitations or exclusion may not apply to you to the extent that liability is by law incapable of exclusion or restriction.

- **Severability.** In the event of invalidity of any provision of this license, the parties agree that such invalidity shall not affect the validity of the remaining portions of this license.
- **No liability for consequential damages.** In no event shall [SQL Maestro Group](#) or its suppliers be liable to you for any consequential, special, incidental or indirect damages of any kind arising out of the delivery, performance or use of the SOFTWARE, even if SQL Maestro Group has been advised of the possibility of such damages. In no event will SQL Maestro Group's liability for any claim, whether in contract, tort or any other theory of liability, exceed the license fee paid by you, if any.
- **Entire agreement.** This is the entire agreement between you and [SQL Maestro Group](#) which supersedes any prior agreement or understanding, whether written or oral, relating to the subject matter of this license.
- **Reserved rights.** All rights not expressly granted here are reserved to [SQL Maestro Group](#).

1.5 About SQL Maestro Group

SQL Maestro Group is a privately-held company producing high-quality software for database administrators and developers. The united team of eminently qualified developers is pleased to create new software products for commercial, academic and government customers worldwide. We do our best to design and develop products that remove complexity, improve productivity, compress time frames, and increase database performance and availability. We are glad to realize that our products take usual chores upon themselves, so that our customers could have more time left for their creative work.

The company was founded in 2002 as an essential partner for every business that is trying to harness the explosive growth in corporate data. SQL Maestro Group employs an international team concentrating their efforts on cutting-edge DBA tools development.

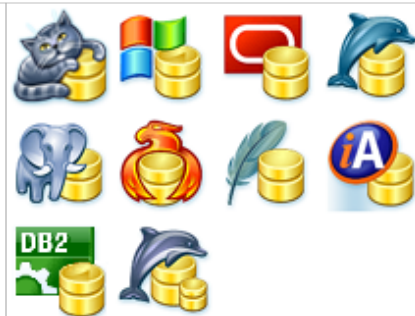
The slogan of our company is **The Shortest Path to SQL**. It is aimed to denote that we set to create easy-to-use products meant for those who appreciate comfort, friendly program interface and support when working with SQL servers.

- We are pleased to facilitate your job.
- We aim at being of considerable assistance to our clients.
- We feel contented doing our beloved work.

At present, our company offers a series of Windows GUI admin tools for SQL management, control and development of the following servers: **MySQL, Microsoft SQL Server, PostgreSQL, Oracle, SQL Anywhere, DB2, SQLite, Firebird, and MaxDB**. We also produce universal tools to be used for administering any database engine accessible via ODBC driver or OLE DB provider. Such products may be the clear-cut decision for those who constantly work with several database servers.

SQL Maestro is the premier Windows GUI admin tool for database development, management, and control.

It provides you with the ability to perform all the necessary database operations such as creating, editing, copying, extracting and dropping database objects; moreover, you can build queries visually, execute queries and SQL scripts, view and edit data including BLOBs, represent data as diagrams, export and import data to/from most popular file formats, manage users and their privileges (if possible), and use a lot of other tools designed for making your work with your server comfortable and efficient.



SQL PHP Generator is a powerful tool for creating database-driven web applications visually. It allows you to generate high-quality PHP scripts for working with tables, views and queries through the web. You needn't have any programming background to use it.



SQL Data Wizard is a high-capacity Windows GUI utility for managing your data.

It provides you with a number of easy-to-use wizards for performing the required data manipulation easily and quickly. The tool allows you to export data from PostgreSQL tables and queries to most popular formats, import data into the tables, generate SQL dump of selected tables, and export/import BLOB fields from/to files.



SQL Code Factory is a premier GUI tool aimed at the SQL queries and scripts development.

It allows you to manage SQL queries and scripts using such useful features as code folding, code completion and syntax highlighting, build query visually, execute several queries at a time, execute scripts from files, view and edit result data with filtering, sorting and grouping abilities, export data to as many as 14 file formats including Excel, RTF and HTML, import data from Excel, CSV, XML and text files, view and edit BLOBs in various way, build diagrams based on Oracle data, and much more.



Database Converter is a user friendly tool to migrate any local or remote ADO-compatible database to PostgreSQL.

Such tools transfer database schema and data and are equipped with native support for the most popular database servers.



Data Sync is a powerful and easy-to-use tool for database contents comparison and synchronization.

Such tools can be useful for database administrators, developers and testers that need a quick, easy and reliable way to compare and synchronize their data.



The software products are constantly optimized for the latest server versions support.

You can use the following contact information if necessary:

Our web-site www.sqlmaestro.com

Postal address: **SQL Maestro Group**
140 Broadway, Suite 706
New York City, New York 10005
United States

Thank you for your interest to our company!

1.6 What's new

Please find out the latest PostgreSQL Maestro news at <http://www.sqlmaestro.com/products/postgresql/maestro/news/>

2 Getting Started

The topics in this section provide some basic information about PostgreSQL Maestro, what it is for and what you can do with it.

How to get started:

- [Connect to a database with PostgreSQL Maestro](#) ¹³
- [Explaining user interface](#) ¹⁸
- [How PostgreSQL Maestro looks when you start it for the first time](#) ¹⁹
- [Shortcut keys](#) ²³

Learning more:

- ❑ Study the [Overview of Database Object Management](#) ⁴² section for detailed instructions on using PostgreSQL Maestro.
- ❑ See [Database Tools](#) ²⁶³ and [Queries](#) ²¹² sections for instructions on more advanced procedures!
- ❑ Find out more about [Working with Data in PostgreSQL Maestro](#) ²²⁸.
- ❑ Customize the way PostgreSQL Maestro works, see [Program Options](#) ³²¹ for full details.

2.1 Connect to a database

To manage an existing database with PostgreSQL Maestro, you have to [create the according database profile](#)^[26] first. A profile stores database connection settings, and some additional options to customize the way the software works with the database. After the creation database profiles appear as nodes in the Explorer tree on the left (profile properties can be later changed with [Database Profile Editor](#)^[30]).

When the profile is created you can connect to the database. To do so, select the database in the [Explorer tree](#)^[38], or either select the [Database | Connect to Database](#) main menu item or use the [Connect to Database](#) item of the popup menu. You can also double click the database node in the explorer tree. If connection succeeds, the database node expands displaying the tree of database objects (tables, views, procedures, etc). The database becomes ready for your activities.

How can I disconnect from a database?

In order to disconnect from a database you should first select the database in the explorer tree, then either

- select the [Database | Disconnect from Database](#) main menu item
- or
- use the [Disconnect from Database](#) item of the popup menu.

See also: [Connection parameters](#)^[14]

2.2 Connection parameters

PostgreSQL Maestro allows you to connect to PostgreSQL directly, or via Secure SHell (SSH) tunnel, or HTTP tunnel.

- **Direct connection**

It is the most natural and the most preferable connection mode. Use it each time it is possible.

- **SSH tunnel connection**

If your PostgreSQL server does not allow direct connections from your remote workstations, you can establish connection to an allowed intermediate SSH server and forward all PostgreSQL commands through the [Secure SHell \(SSH\) tunnel](#).

- **HTTP tunnel connection**

[HTTP tunneling](#) is a technique used in conditions of restricted network connectivity including firewalled networks, networks behind proxy servers, and NATs. It is the slowest way and is recommended to use if the others are impossible.

Irrespectively of a connection mode you should specify common credentials as follows:

Host

The host name of the PostgreSQL server.

Port number

The TCP/IP port to use. Default PostgreSQL port is 5432.

User name

The username used to connect to PostgreSQL.

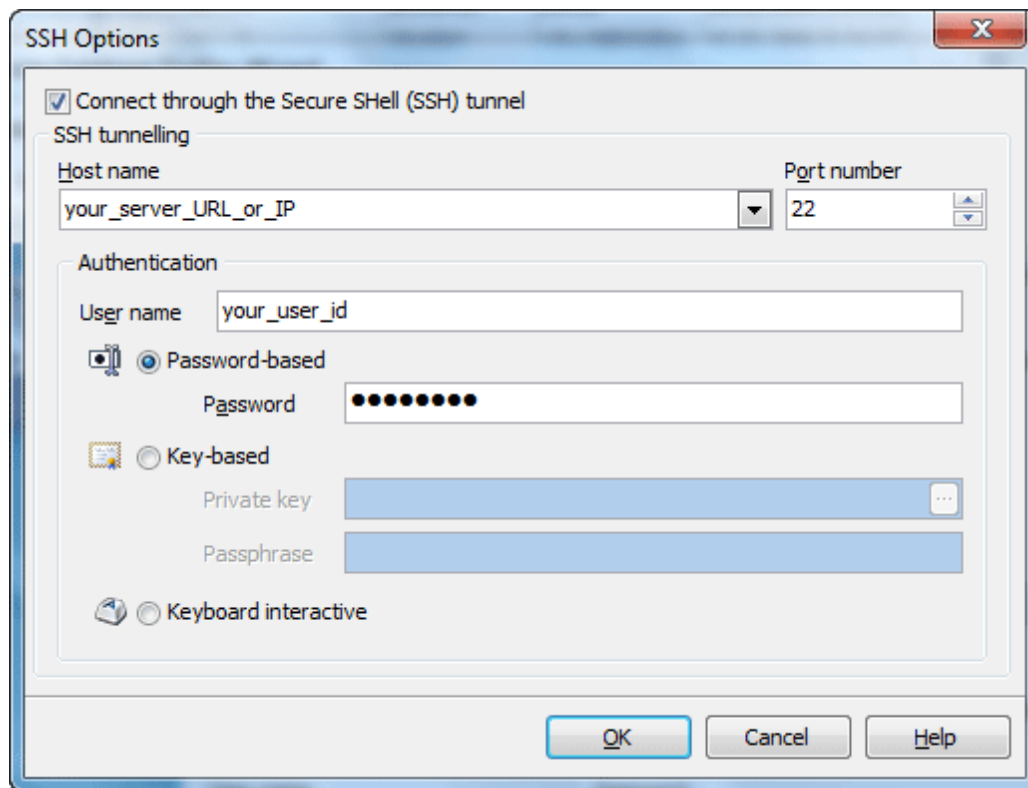
Password

The password for the user account on server.

📖 **More about SSH tunnel connection**

To establish connection to intermediate SSH server and forward all PostgreSQL commands through the secure tunnel, you need to:

1. Check [I can connect to the server directly or via SSH tunneling](#).
2. Follow the [Configure SSH options](#) link to open the [SSH Options](#) window.



3. Check [Connect through the Secure Shell \(SSH\) tunnel](#) and complete the following fields:

Host name

Specify the host name or IP of your site. Note, that PostgreSQL host name always should be set relatively to the SSH server. For example, if both of PostgreSQL and SSH servers are located on the same computer, you should specify localhost as Host name instead of server's external host name or IP address.

Port number

Enter the port number for the SSH server.

4. Enter valid [User name](#) for the remote server and select the [Authentication](#) method and set corresponding credentials.

Password-based

Set the [password](#) corresponding to the specified user.

Key-based

Specify the path to the [Private key](#) file with the corresponding [Passphrase](#) to log in to the remote server. PostgreSQL Maestro accepts keys in **ssh.com** or **OpenSSH** formats. To convert a private key from PuTTY's format to one of the formats supported by our software, [use the PuTTYgen utility](#) that can be freely downloaded from the [PuTTY website](#).

Keyboard interactive

Keyboard authentication is the advanced form of password authentication, aimed specifically at the human operator as a client. During keyboard authentication zero or more prompts (questions) is presented to the user. The user should give the answer to each prompt (question). The number and contents of the questions are virtually not limited, so certain types of automated logins are also possible.

More about connection via HTTP tunnel

To connect to a remote server using an HTTP tunnel, you need to:

1. Upload the connection PHP script to your website. The script is named *pgsql_tunnel.php* and can be found under the installation folder, usually *C:\Program Files\SQL Maestro Group\PostgreSQL Maestro*.
2. Select the [I have to use HTTP tunneling](#) radio button.
3. Enter the connection PHP script URL, e.g. *www.yoursite.com/files/pgsql_tunnel.php*. You can test the connection before the profile is created. Just use [Test script using default browser](#) to open connection script in your browser, enter all the required connection parameters and click the [Test connection](#) button.

Connection Script

Fields marked by * are required.

Host/Server name (or IP) *:	<input type="text" value="neptun"/>
User *:	<input type="text" value="postgres"/>
Password:	<input type="password" value="••••••••"/>
Port (if not 5432):	<input type="text" value="5433"/>
Database *:	<input type="text" value="adventure"/> <input type="button" value="v"/>
	<input type="button" value="Get Database List"/>
	<input type="button" value="Test Connection"/>
	<input type="button" value="ShowTables"/>

© 2002-2008 SQL Maestro Group

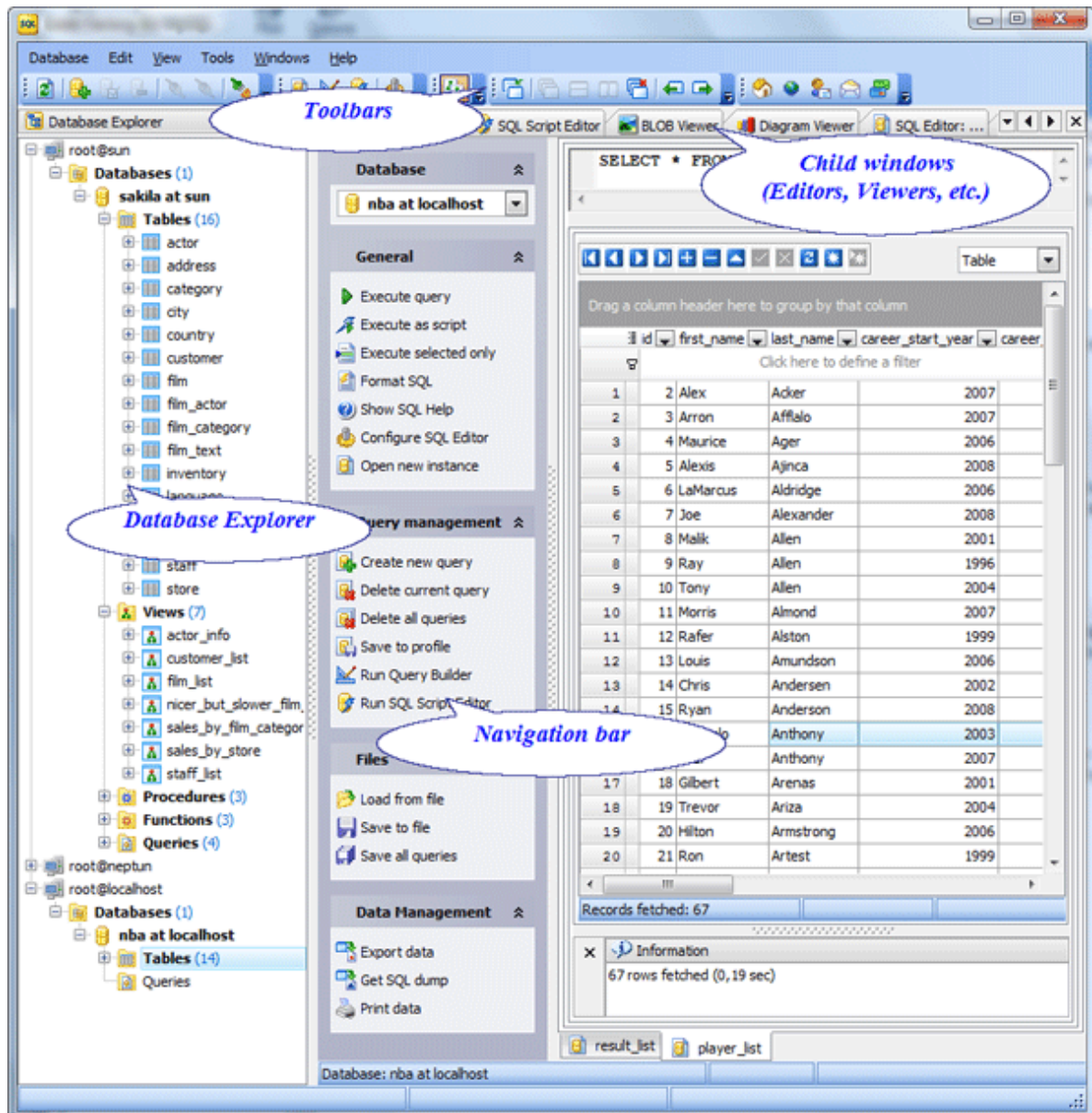
4. In case using of a proxy server use [Configure tunnelling options](#) to open the [HTTP tunnelling options](#) window and specify your [proxy server](#) connection parameters and [HTTP authentication](#).

Note: You are actually connecting to your database through the PHP script on the server, so in most cases the host/server name is "localhost" unless the target database server is not installed on the same computer as the Web server.

2.3 Explaining user interface

The SQL Maestro Group products are famous for their clear and intuitive user interface. The programs are built around the three-pane workspace that includes the [database explorer](#) and child windows consisting of the [navigation bar](#) and [work area](#).

This topic provides a brief guide to the components of PostgreSQL Maestro's user interface. For detailed descriptions, see below.



Database Explorer

The [Database Explorer](#)^[38] occupies the left side of PostgreSQL Maestro main window. It represents all the connected databases objects [including system objects](#)^[31].

The explorer provides the fastest way to reach the object properties, to perform the following operations with database profiles using the popup menu:

- create new objects (database profiles, database objects, table objects...);
- edit currently selected objects;
- remove currently selected objects from the explorer tree;
- duplicate objects;
- rename objects if available and edit object comments out of the object editor.

See also: [Filtering explorer content](#)^[61]

Editors and Viewers

According to the MDI style implementation the PostgreSQL Maestro tools and editors are opened in appropriate windows. Each window consists of a navigation bar and work area. The software supports Classic and Tabbed MDI.

See also: [Switching between windows](#)^[22], [Tabbed MDI](#)^[20]

Navigation bar

The [Navigation Bar](#) contains a set of logically grouped links provided to realize the corresponding actions. Just position the mouse over a link and wait for a second to display the appropriate action shortcut making it possible for experienced users to control the program almost entirely with the keyboard.

See also: [Shortcut keys](#)^[23]

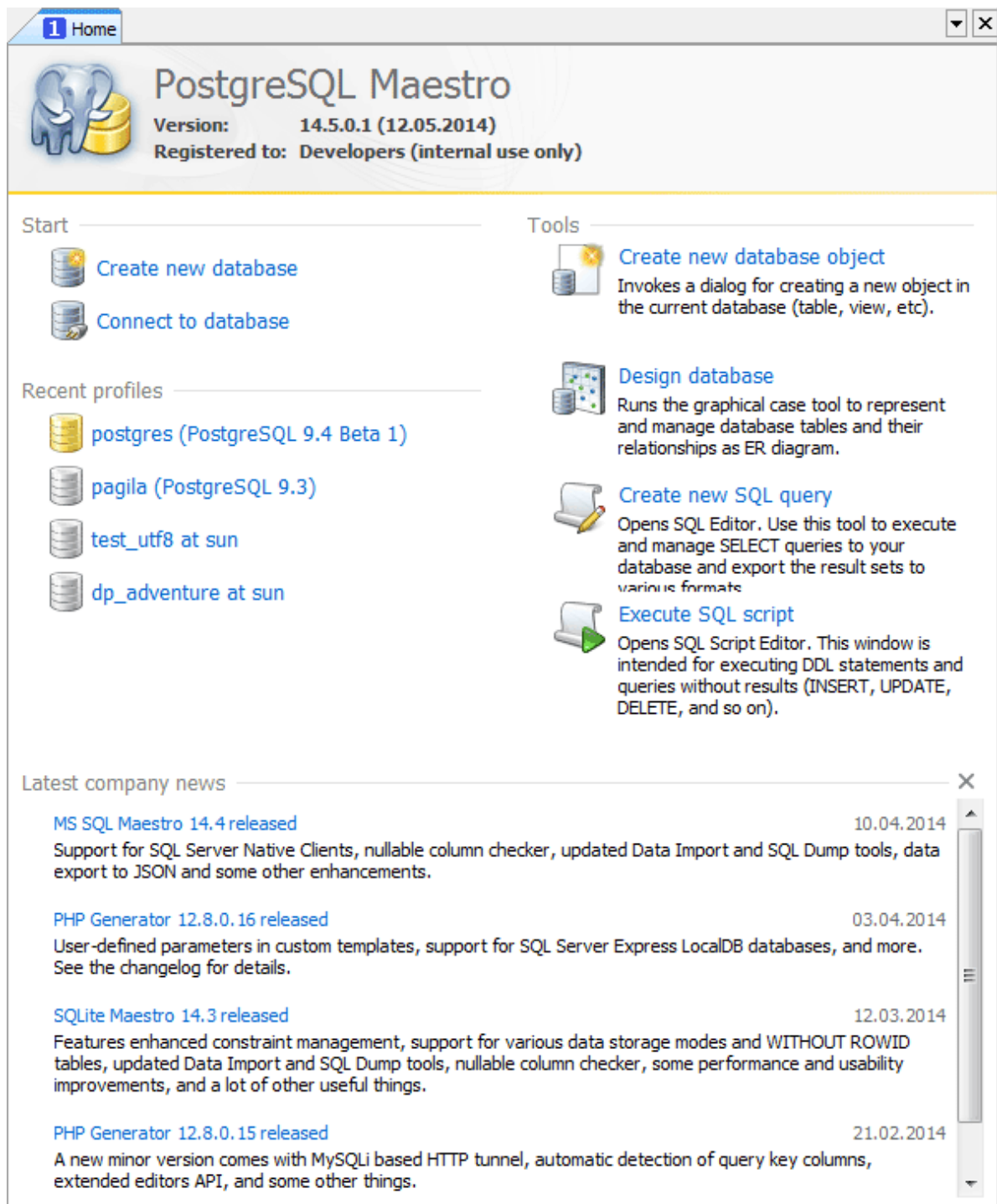
Toolbars

The bars occupy the top of the main window. The [Toolbars](#) provide quick access to the most frequently-used functions. Just position the mouse over a tool and wait for a second to display a brief text describing what it is for.

2.3.1 First time started

This is how PostgreSQL Maestro looks when you run it for the first time. The [Create new database](#)^[37] and [Connect to database](#)^[26] links allow you to start working with a new and existing databases.

The window provides you with quick access to the [Create Database Object](#)^[43] dialog, [Schema Designer](#)^[298], [SQL Editor](#)^[214], and [SQL Script Editor](#)^[266] recently used database profiles. At the bottom of the page the latest company news and current discount programs are represented.



2.3.2 Tabbed MDI

PostgreSQL Maestro provides you with a possibility to choose ([Options|Application](#)) your favorite UI. Among the **classic MDI style** the **tabbed MDI style** is also available.

Applying the style you'll get all the objects editors opening on separate sheets. You can move from one sheet to another by clicking the sheet tabs at the bottom of the working area. The tab for the active sheet is underlined in the color you choose; tabs for

inactive sheets are fully colored.

You can switch between the sheets with corresponding sheet tabs or using **Ctrl+Tab**. If you don't see the tool you want, click the tab scrolling buttons to display the tab, and then click the tab. You can also move the sheets.

The screenshot displays the SQL Maestro Group interface. On the left, a sidebar contains several sections: **Database** (with a dropdown for 'nba at localhost'), **General** (with options like 'Execute query', 'Execute as script', 'Execute selected only', 'Format SQL', 'Show SQL Help', 'Configure SQL Editor', and 'Open new instance'), **Query management** (with options like 'Create new query', 'Delete current query', 'Delete all queries', 'Save to profile', 'Run Query Builder', and 'Run SQL Script Editor'), **Files** (with options like 'Load from file', 'Save to file', and 'Save all queries'), and **Data Management** (with options like 'Export data', 'Get SQL dump', and 'Print data').

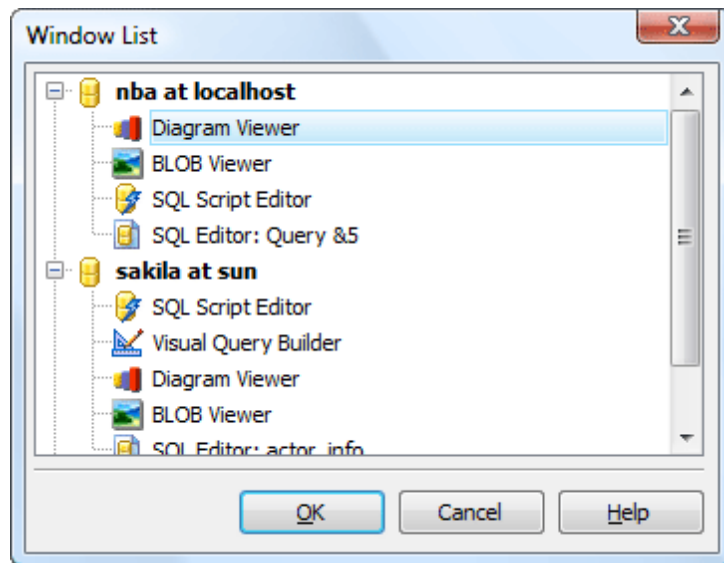
The main window shows a query editor with the text `SELECT * FROM player;`. Below the editor, a table of results is displayed. The table has columns: `id`, `first_name`, `last_name`, `career_start_year`, and `career_end_year`. The table contains 22 rows of data. The 15th row is highlighted. Below the table, a status bar indicates 'Records fetched: 67'. At the bottom, there is an 'Information' box showing '67 rows fetched (0,19 sec)'. The bottom of the window shows two tabs: 'result_list' and 'player_list'. The status bar at the very bottom indicates 'Database: nba at localhost'.

	id	first_name	last_name	career_start_year	career_end_year
1	2	Alex	Acker	2007	
2	3	Arron	Afflalo	2007	
3	4	Maurice	Ager	2006	
4	5	Alexis	Ajinca	2008	
5	6	LaMarcus	Aldridge	2006	
6	7	Joe	Alexander	2008	
7	8	Malik	Allen	2001	
8	9	Ray	Allen	1996	
9	10	Tony	Allen	2004	
10	11	Morris	Almond	2007	
11	12	Rafer	Alston	1999	
12	13	Louis	Amundson	2006	
13	14	Chris	Andersen	2002	
14	15	Ryan	Anderson	2008	
15	16	Carmelo	Anthony	2003	
16	17	Joel	Anthony	2007	
17	18	Gilbert	Arenas	2001	
18	19	Trevor	Ariza	2004	
19	20	Hilton	Armstrong	2006	
20	21	Ron	Artest	1999	
21	22	Darrell	Arthur	2008	

2.3.3 Switching between windows

The [Window List](#) dialog allows you to switch the child application windows quickly. To open the dialog select the [Windows | Window List...](#) item of the main menu or use the **Alt+O** hot keys combination.

Most of the windows are linked according to their active databases and displayed in the form of a tree, e.g. [Table Editor](#), [SQL Editor](#), [Diagram Viewer](#), etc. Windows which are common for the entire program are shown as separate nodes of the tree.



To activate the window you need, select one of the window tree items and click the **OK** button.

2.4 Shortcut keys

The following table describes the default shortcut keys in PostgreSQL Maestro.

Interface ¹⁸	
Window list	Alt+O
Previous Window	F6
Next Window	Ctrl+F6
Show Database Explorer	F11
Refresh	F5
Exit	Alt+F4
PostgreSQL Maestro help	F1
Clipboard	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Select all	Ctrl+A
Find	Ctrl+F
Replace	Ctrl+H
Search again	F3
Undo	Ctrl+Z
Redo	Shift+Ctrl+Z
SQL Editors ²¹⁴	
Open SQL Editor	Ctrl+E
Open SQL Script Editor	Ctrl+R
Open Visual Query Builder	Ctrl+Q
Execute query	(F9) or (F8)
Execute query as script	(Shift+F9) or (Shift+F8)
Execute selected only	(Alt+F9) or (Alt +F8)
Go to line	Ctrl+G
Format selected SQL	Ctrl+Alt+F
Create new query	Ctrl+N
Delete current query	Ctrl+Alt+D
Load script from file	Ctrl+O
Database management ²⁴	
Create a new database profile	Shift+Ctrl+P
Edit an existing database profile	Shift+Ctrl+E
Rename a database profile (object)	F2
Remove database profile	Shift+Ctrl+R
Connect to the database	Shift+Ctrl+C
Disconnect from the database	Shift+Ctrl+D
Create a database object	Shift+Ctrl+N
Object Browser	Shift+Ctrl+O
Open BLOB Viewer	Ctrl+B

3 Databases and Database Profiles

PostgreSQL Maestro allows you to manipulate databases by means of database profiles. Profile contains database connection settings and a set of options to automatize common manipulations with databases (a possibility to connect to the database at PostgreSQL Maestro startup, login prompt before connection, etc.). To start working with databases in PostgreSQL Maestro, you should create database profile(s) first.

Use the following links for details:

■ **How can I create a new database?**

Use for this purpose [Create Database Wizard](#)^[37]. In order to run the wizard you should either

- select the [Database | Create New Database...](#) main menu item
- or
- use the [Create New Database...](#) item of the popup menu.

Using [Create Database Wizard](#) set the [Create profile after creating the database](#) option to create a new profile and open the [Database Profile Properties](#) dialog after the database is created.

■ **How can I change attributes of an existing database?**

To edit a database:

- select the database to edit in the explorer tree;
- edit database properties within the appropriate tabs of [Database Editor](#)^[40].

■ **How can I drop an existing database?**

In order to drop a database you should first select the database to drop in the explorer tree and establish connection (if you are not connected to the database yet), then either

- select the [Database | Drop Database](#) main menu item
- or
- use the [Drop Database](#) item of the popup menu

and confirm dropping in the dialog window to complete the operation.

■ **How can I create new database profiles?**

In PostgreSQL Maestro database profiles are created within [Create Database Profiles Wizard](#)^[38]. In order to run the wizard you should either

- select the [Database | Create Database Profiles...](#) main menu item
- or
- use the [Create Database Profiles...](#) item of the popup menu.

Using [Create Database Profiles Wizard](#) set the necessary connection and authorization options and click the [Ready](#) button to complete the operation.

How can I edit existing database profile options?

Database connection properties and profile options are edited within the [Database Profile Properties](#)^[30] dialog window. In order to open the dialog for the selected database profile you should either

- select the [Database | Edit Database Profile...](#) main menu item
- or
- use the [Edit Database Profile...](#) item of the popup menu.

How can I remove database profiles?

In order to remove a database profile you should first select the database profile in the explorer tree, then either select the [Database | Remove Database Profile](#) main menu item, or use the [Remove Database Profile](#) item of the popup menu and confirm removing profile in the dialog window to complete the operation.

How can I connect to a database?

In order to connect to a database you should first select the database in the explorer tree, then either

- select the [Database | Connect to Database](#) main menu item
- or
- use the [Connect to Database](#) item of the popup menu.

How can I disconnect from a database?

In order to disconnect from a database you should first select the database in the explorer tree, then either

- select the [Database | Disconnect from Database](#) main menu item
- or
- use the [Disconnect from Database](#) item of the popup menu.

3.1 Creating Database Profiles

Create Database Profiles Wizard allows you to create a single database profile or several profiles from one host. To run the wizard, select the [Database | Create Database Profiles...](#) main menu item, or press the **Shift+Ctrl+P** hot keys combination. You can also use the [Create Database Profiles](#) button of the main toolbar.

- [Set connection properties](#)^[26]
- [Specify database profile options](#)^[27]

See also: [Edit Database Profile Dialog](#)^[30]

3.1.1 Setting connection properties

Specify PostgreSQL [connection properties](#)^[14] to be used on further connections.

Create Database Profiles Wizard

Create Database Profiles

Specify the connection parameters

Welcome to the Create Database Profiles Wizard!
This wizard allows you to set the connection parameters for the selected databases only once, giving you the ability to connect them quickly afterwards.

This wizard will guide you through the process of setting the connection parameters, selecting databases, and customizing the result profiles.

☒ I can connect to the server directly or via SSH tunneling
[Configure SSH options](#)

☐ I have to use HTTP tunneling
URL:
[Configure tunneling options](#) [Test script using default browser](#)

Host: Port number:

User name: Password:

☒ Create a single profile ☐ Hide already registered databases

[Help](#) [< Back](#) [Next >](#) [Cancel](#)

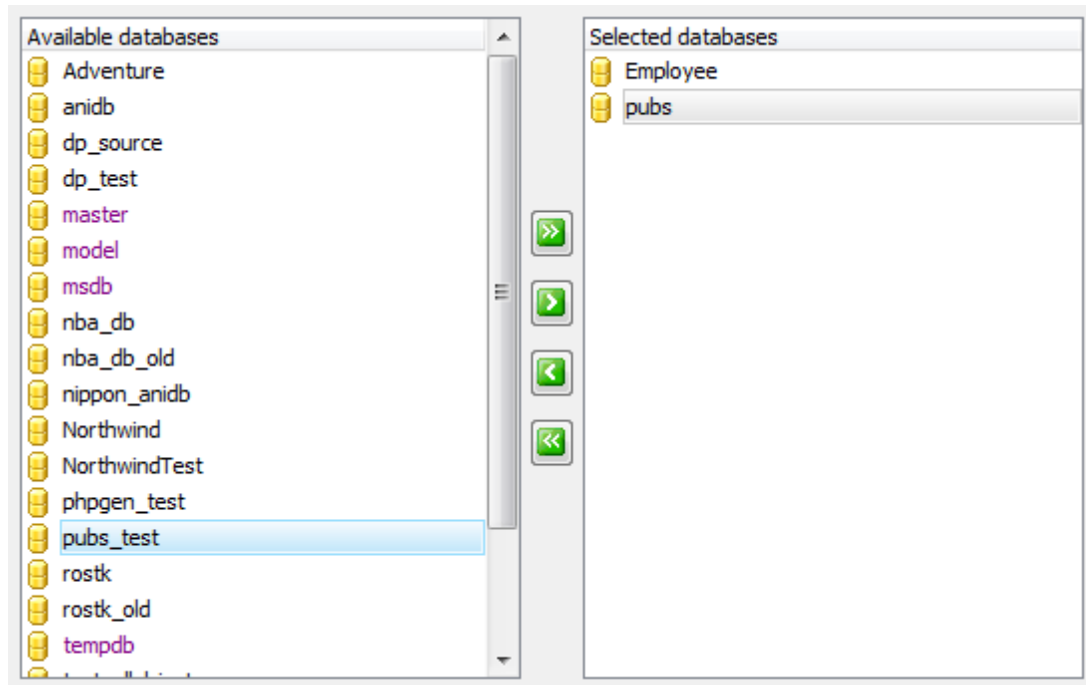
✓ Check the [Create a single profile](#) option to set the database name manually and create a single profile for this database.

✓ [Hide already registered databases](#)

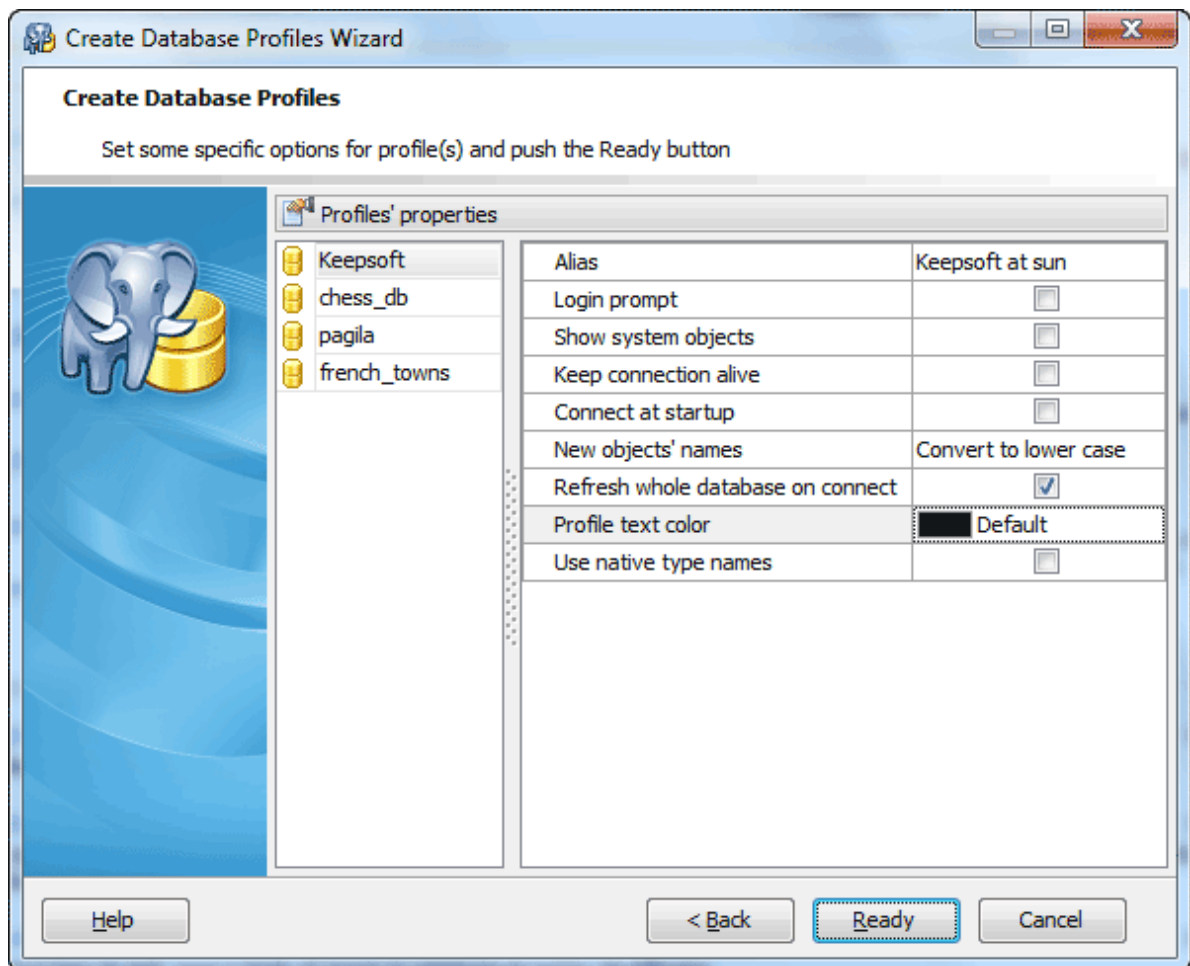
Check the box to shorten the databases list on the next wizard step.

3.1.2 Setting profile options

To create a new profile, select databases to be registered by moving them from the [Available databases](#) list to the [Selected databases](#) list. This step is available if the [Create a single profile](#) option is unchecked on the previous wizard step.



You can select several databases to set options for all the selected databases at once (except the alias which should be unique for each individual database).



☒ **Login prompt before connection**

Use the option to enable PostgreSQL Maestro to prompt for user name and password every time you connect to the database.

☒ **Show system objects**

Check the box to make system objects visible.

☒ **Keep connection alive**

Check the box for pinging server before each query execution.

☒ **Connect at startup**

With this option on connection to the profile database is automatically established at the application startup.

New objects' names (Don't change case, Convert to upper case, Convert to lower case)
The option allows you to specify the newly created objects case.

☒ **Refresh whole database on connect**

Use the option along with the [Show empty schemas](#) explorer options to hide/show empty schemas in the explorer tree.

Profile text color

Select the color to be used to represent the database profile name at the Explorer tree. For example this option may be useful to mark development and production databases in different colors in order to prevent casual metadata or data changes in the production.

Click the [Ready](#) button when done to start working with the selected databases in PostgreSQL Maestro.

3.2 Editing Database Profile

Use the [Edit Database Profile](#) dialog to edit the profile properties set on its creation. To open the dialog, select the database in the explorer tree, then select the [Database | Edit Database Profile...](#) main menu item or press the **Shift+Ctrl+E** hot key combination. You can also use the [Edit Database Profile](#) button of the main toolbar.

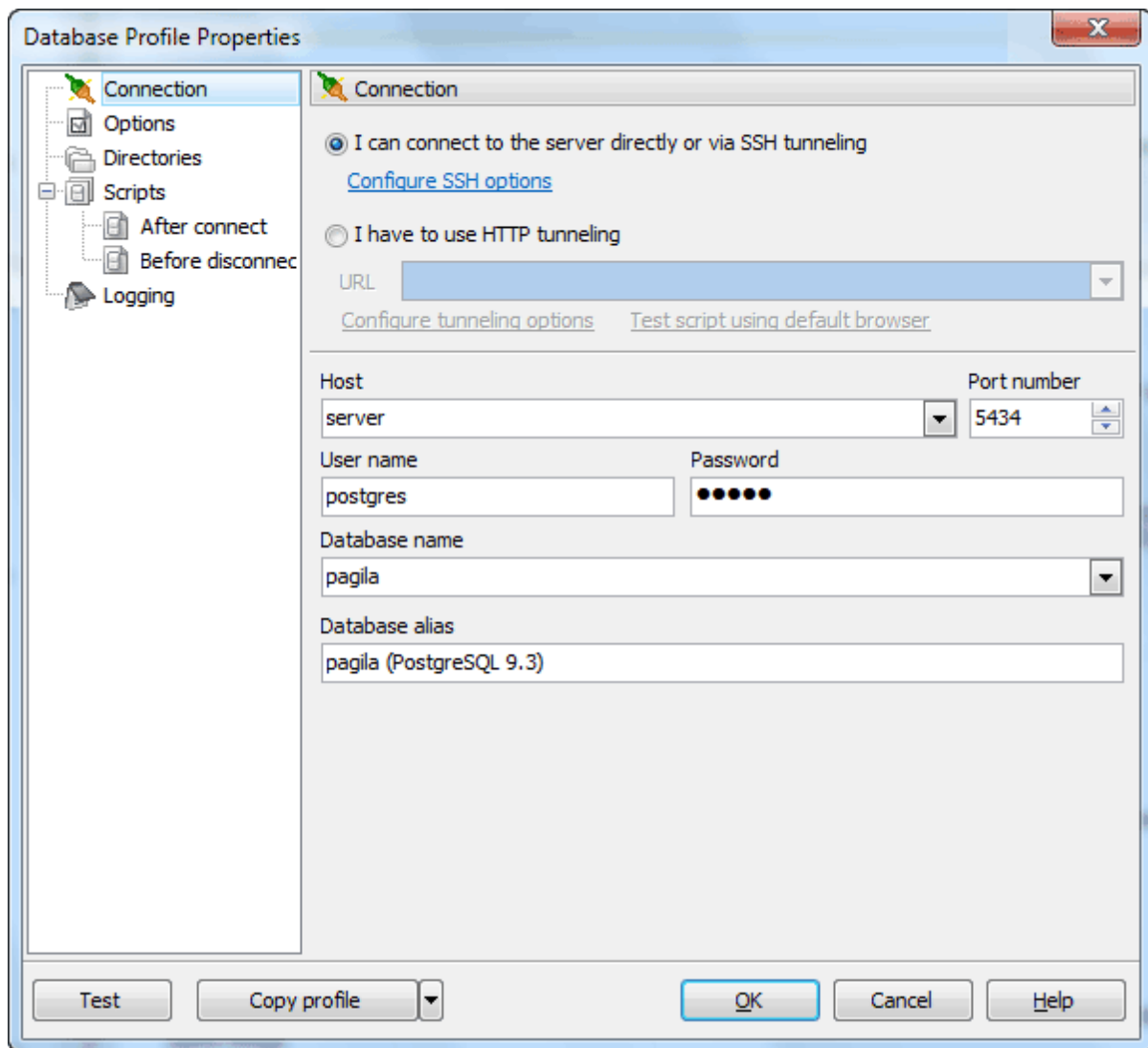
Instead of manual profile options editing you can copy all the options from the another existing profile with the [Copy profile](#) button.

- [Editing database connection properties](#) ^[30]
- [Settings database options](#) ^[31]
- [Setting default directories for database tools](#) ^[33]
- [Editing obligatory scripts to execute](#) ^[34]
- [Setting log options and file names](#) ^[34]

See also: [Create Database Profile Wizard](#) ^[26]

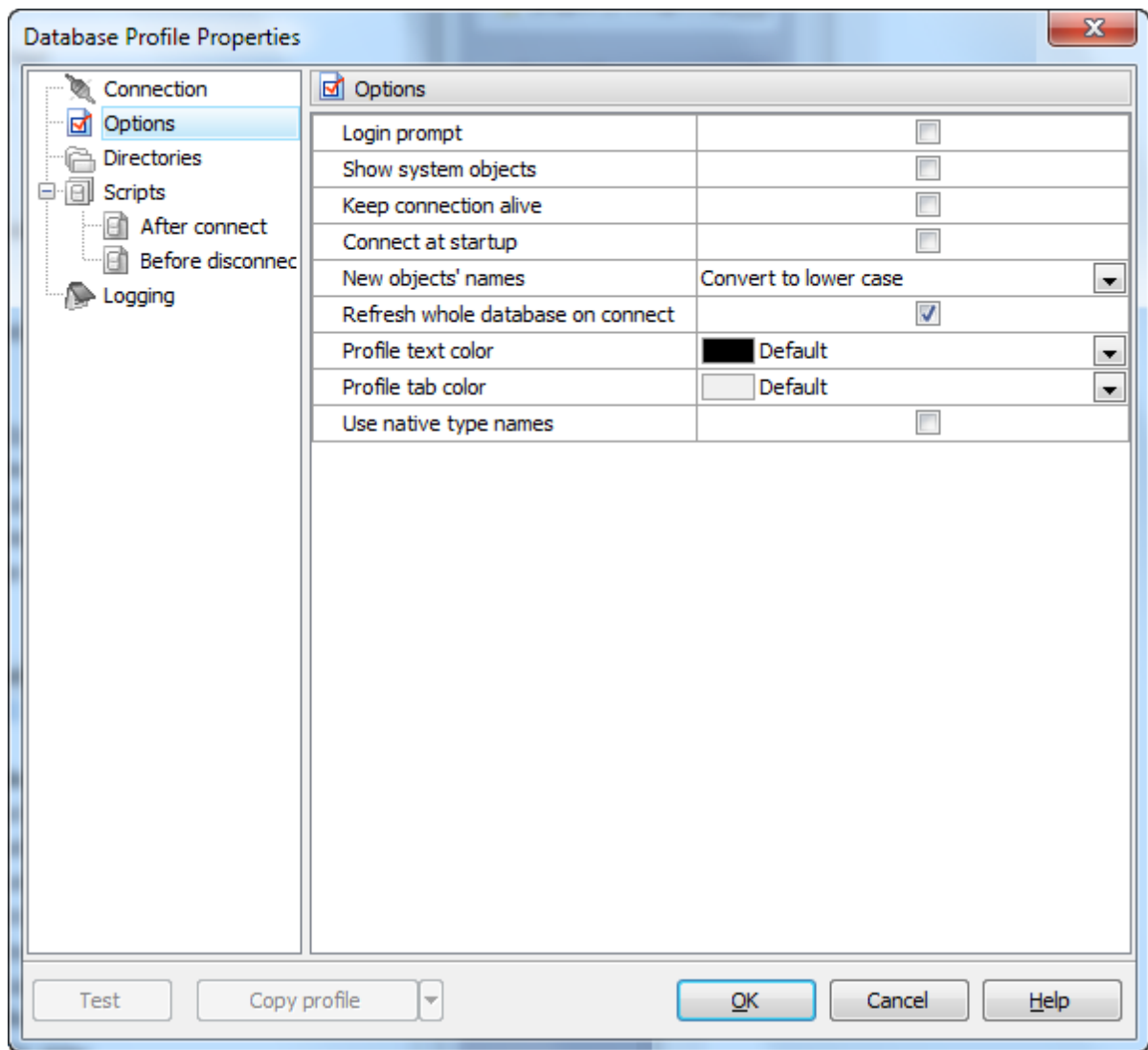
3.2.1 Editing connection properties

The tab allows you to change [connection properties](#) ^[14] of an existing database profile. Here you can change the database group, database info and edit the database alias, an optional name to display the database in the Explorer tree and in all the application tools.



3.2.2 Setting profile options

Customize database options according to your needs. The detailed description is given below.



☒ **Login prompt**

Use the option to enable PostgreSQL Maestro to prompt for user name and password every time you connect to the database.

☒ **Show system objects**

Check the option to make system objects visible.

☒ **Keep connection alive**

Check the box for pinging server before each query execution.

☒ **Connect at startup**

With this option on connection to the profile database is automatically established at the application startup.

New objects' names (Don't change case, Convert to upper case, Convert to lower case)
Use the option to change the case for newly created objects.

☒ **Refresh whole database on connect**

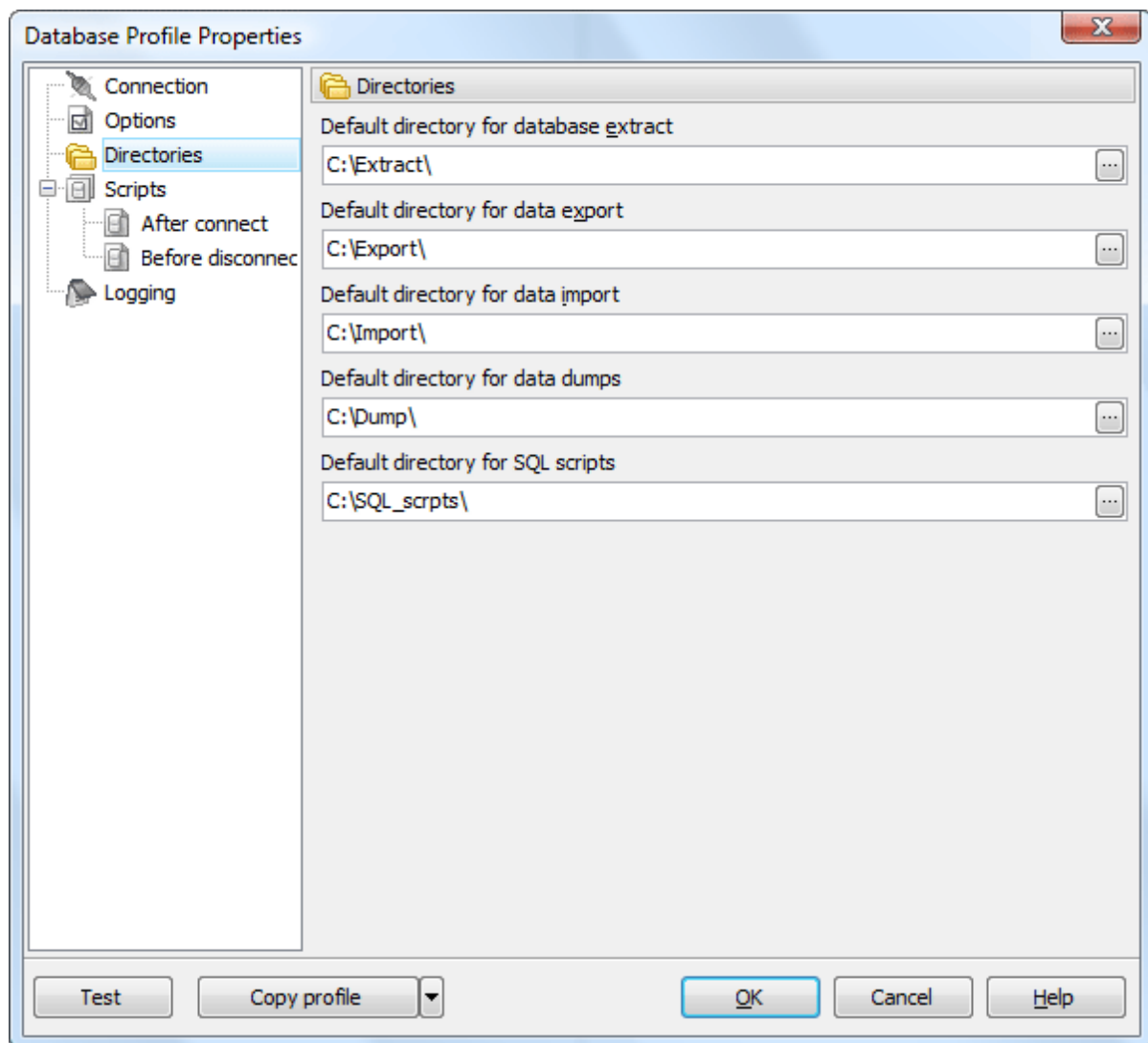
Use the option along with the [Show empty schemas](#)³²⁷ explorer options to hide/show empty schemas in the explorer tree.

You can also change here the font color the profile name is represented at the Explorer tree.

To use native type names (i.e. *int2* as *smallint*, *int4* as *integer*), check the according box.

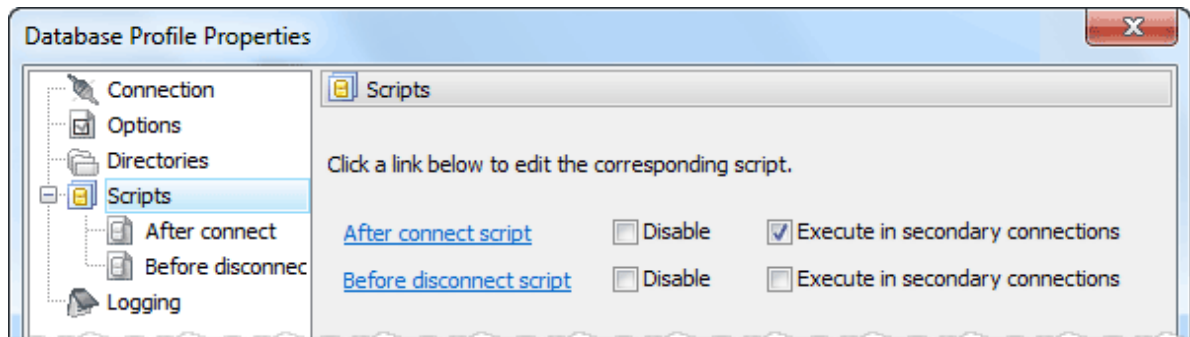
3.2.3 Setting default directories

Use the tab to specify the default directories respectively for database extract, data export, data import, and data dump.

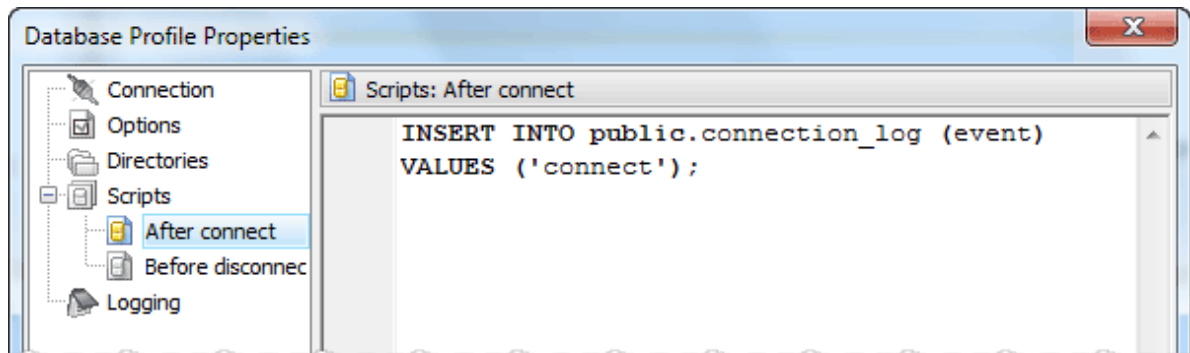


3.2.4 Editing obligatory scripts to execute

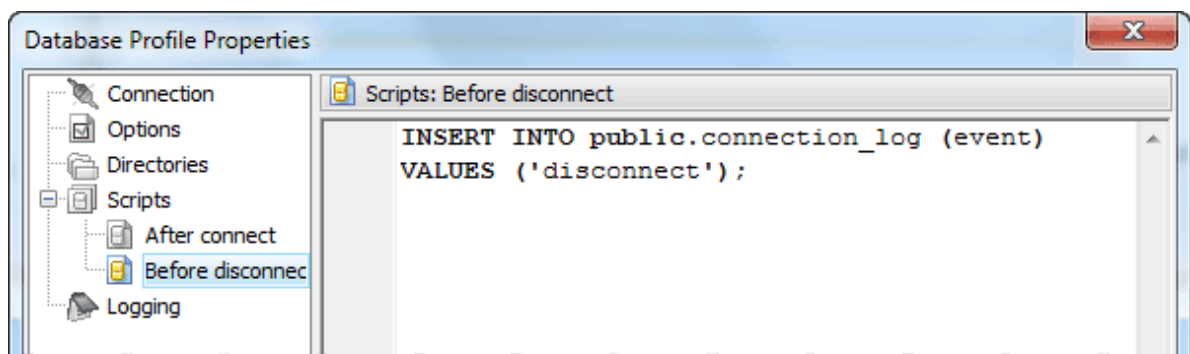
Use the tab to specify the obligatory scripts to execute in all database connections established by the software (on executing queries, browsing objects data, etc.). There is a possibility to enable/disable a written script.



Below you can find an example of an obligatory script to execute after PostgreSQL Maestro will connect to the database. The script writes a connect time to the log table.

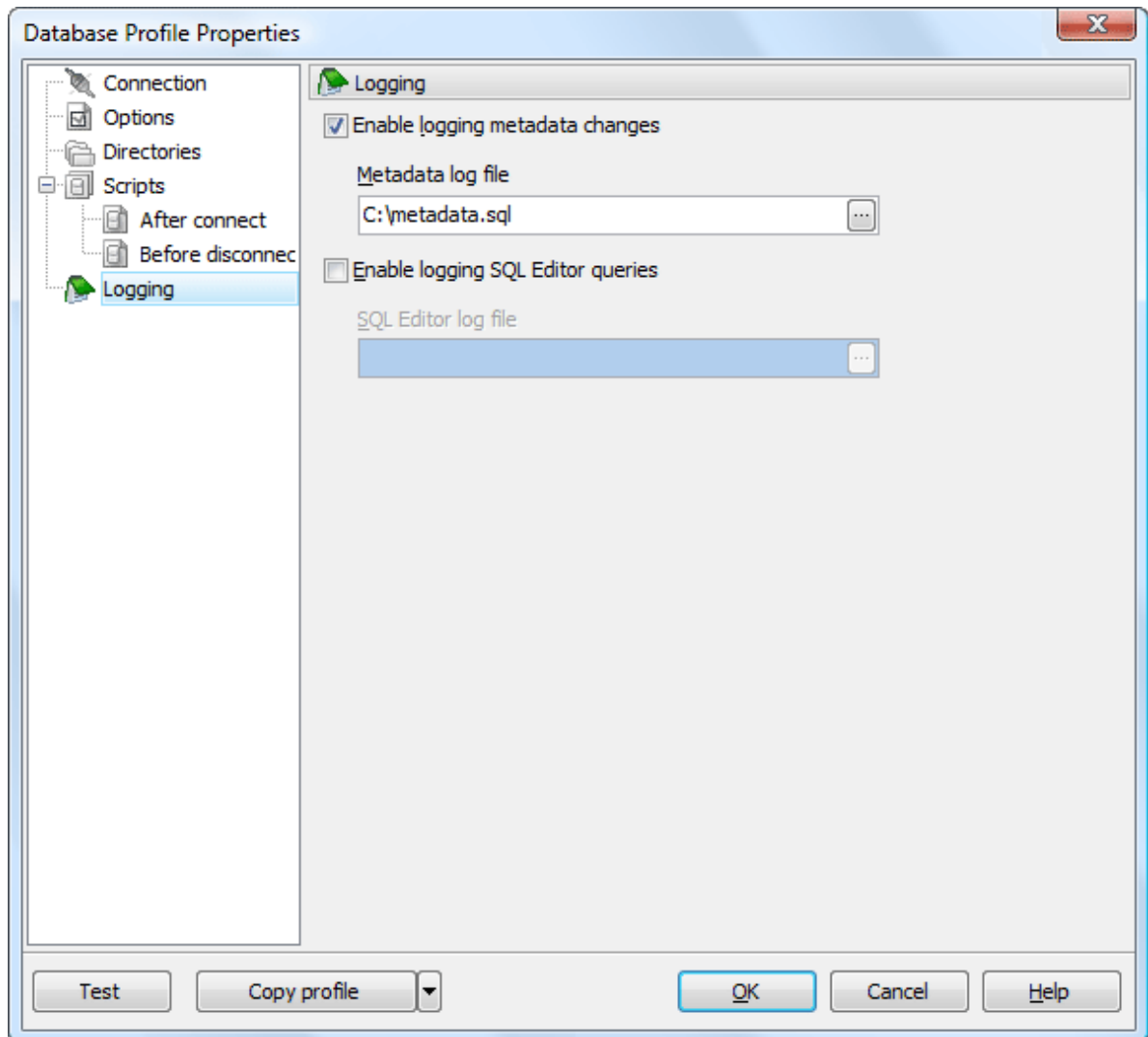


The next screen represents the example of an obligatory script to execute before PostgreSQL Maestro will disconnect from the database. The script writes a disconnect time to the log table.




3.2.5 Setting log options


Enable/disable [metadata changes logging](#) and [SQL query logging](#) and specify the corresponding log file names if necessary.



3.2.6 Statistics

This tab allows you to view usage statistics for the current profile. Click the **Reset Statistics** button to clear all the displayed values.

 Statistics

 **Statistics**

Creation time	N/A
Last modification time	N/A
Number of connections	6
Last connection time	18.08.2017 16:14:16
Total uptime	2:03:51:22

Reset statistics

3.3 Create Database Wizard

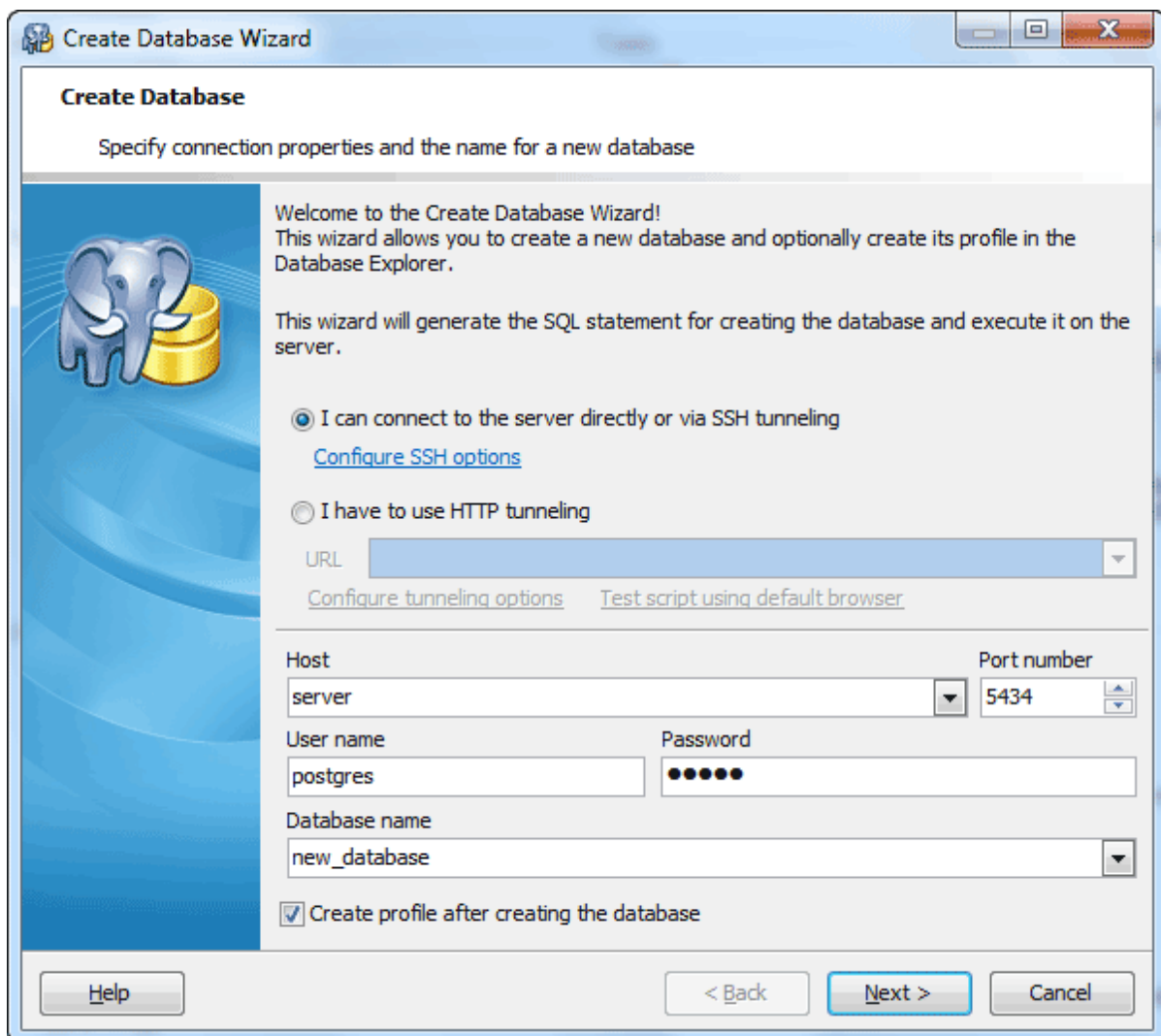
To run the [Create Database Wizard](#), select the [Database | Create New Database...](#) main menu item or click the [Create New Database](#) button on the main toolbar.

- [Setting database connection properties](#)^[37]
- [Specifying database properties](#)^[38]

See also: [Database Editor](#)^[40]

3.3.1 Setting connection properties

Set the [connection properties](#)^[14] of the new database. If the [Create profile after creating the database](#) option is checked, the [Edit Database Profile](#)^[30] dialog is opened after the new database is created.



The screenshot shows the 'Create Database Wizard' dialog box. The title bar says 'Create Database Wizard'. The main heading is 'Create Database' with the subtitle 'Specify connection properties and the name for a new database'. On the left is a blue background with a white elephant icon standing next to a yellow database cylinder. The text on the right says: 'Welcome to the Create Database Wizard! This wizard allows you to create a new database and optionally create its profile in the Database Explorer. This wizard will generate the SQL statement for creating the database and execute it on the server.' There are two radio buttons: 'I can connect to the server directly or via SSH tunneling' (selected) and 'I have to use HTTP tunneling'. Below the first radio button is a link 'Configure SSH options'. Below the second radio button is a 'URL' text box and links 'Configure tunneling options' and 'Test script using default browser'. There are four input fields: 'Host' (text box with 'server'), 'Port number' (spin box with '5434'), 'User name' (text box with 'postgres'), and 'Password' (password box with six dots). Below these is a 'Database name' dropdown menu with 'new_database' selected. At the bottom is a checked checkbox 'Create profile after creating the database'. The bottom of the dialog has three buttons: 'Help', '< Back', and 'Next >', and a 'Cancel' button on the far right.

3.3.2 Specifying database properties

The next wizard step allows you to set common database options. All fields below are optional, i.e. it is not obligatory for you to fill them.

Owner

You can specify here the name of the PostgreSQL server user who will own the new database, or leave this field blank to use the default user (namely, the user executing the command). By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Allow connections

If false then no one can connect to this database. This property is available for PostgreSQL 9.5 and higher.

Connection limit

Defines how many concurrent connections can be made to this database. -1 means no

limit.

Is template

If true, then this database can be cloned by any user with CREATEDB privileges; if false, then only superusers or the owner of the database can clone it. This property is available for PostgreSQL 9.5 and higher.

Encoding

Character set encoding to use in the new database. Select an encoding from the drop down list or leave this field blank to use the default encoding.

Template

Specify the name of the template from which the new database is to be created, or leave this field blank to use the default template (template1).

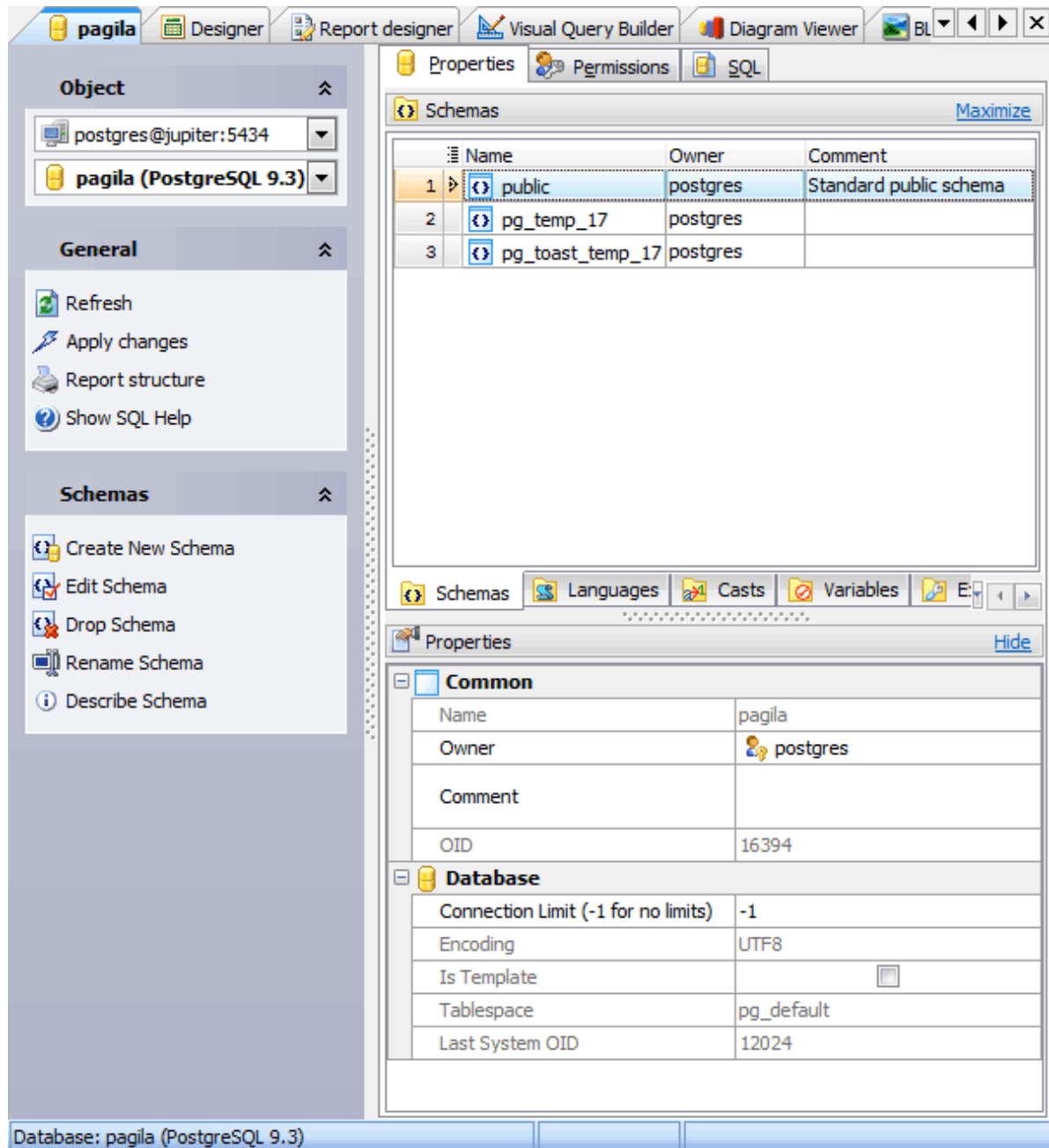
Tablespace

Set the tablespace name to associate with the new database, or leave this field blank to use the template database tablespace. This tablespace will be the default tablespace used for objects created in this database. The default tablespace is created using the template database's tablespace.

3.4 Database Editor

Database Editor allows you to browse, add, edit and delete all objects of the selected database and its main properties.

To open the editor, use popup menu of the database node at the Explorer tree.



Subitems

Every tab is intended for managing corresponding database objects (e.g. *tables*, *views*, *queries*, etc.). Open the object in its editor by double-clicking or pressing the **Enter** key. The popup menu allows you to create new, edit or drop the selected database

objects. Using this menu you can also create a copy of the object.

You can operate on several objects at a time. For this you have to select database objects with the **Shift** or the **Ctrl** key pressed. After the group of objects is selected, you can operate on it, e.g. delete several objects at once, as it was a single object.

The [Properties](#) tab displays available database parameters. Below you can find some of their descriptions.

OID

This field contains the database OID (object identifier). The latter can be defined as a serial number that is automatically added by PostgreSQL to all databases.

Connection Limit

Define the limit of connections to the database here.

Encoding

Character set encoding to use in the database.

Is Template

Set on if the database is template.

Tablespace





Edit the database tablespace name here. The default tablespace is created with the help of the template database tablespace.

Last System OID

Displays the last system OID (object identifier).

4 Database Object Management

PostgreSQL Maestro provides you with several tools to manage and navigate PostgreSQL objects. To browse and modify objects, at least one connection to a database should be established.

- [Browse Database Objects](#) 
- [Create New Objects](#) 
- [Edit Existing Objects](#) 
- [Duplicate Objects](#) 

The options to create or edit an object in PostgreSQL Maestro follow the parameters defined by PostgreSQL. If you need clarification on what an option means or how it should be used, see PostgreSQL's documentation for more information. The documentation provides detailed description of objects, including their purpose, properties, and restrictions. The PostgreSQL Maestro manual provides you with only brief review of PostgreSQL objects.

4.1 Create Objects

PostgreSQL Maestro provides a number of [Create Object Wizards](#)⁴³ to accomplish the most facile PostgreSQL object creation.

There are several ways to invoke the necessary Create Object Wizard:

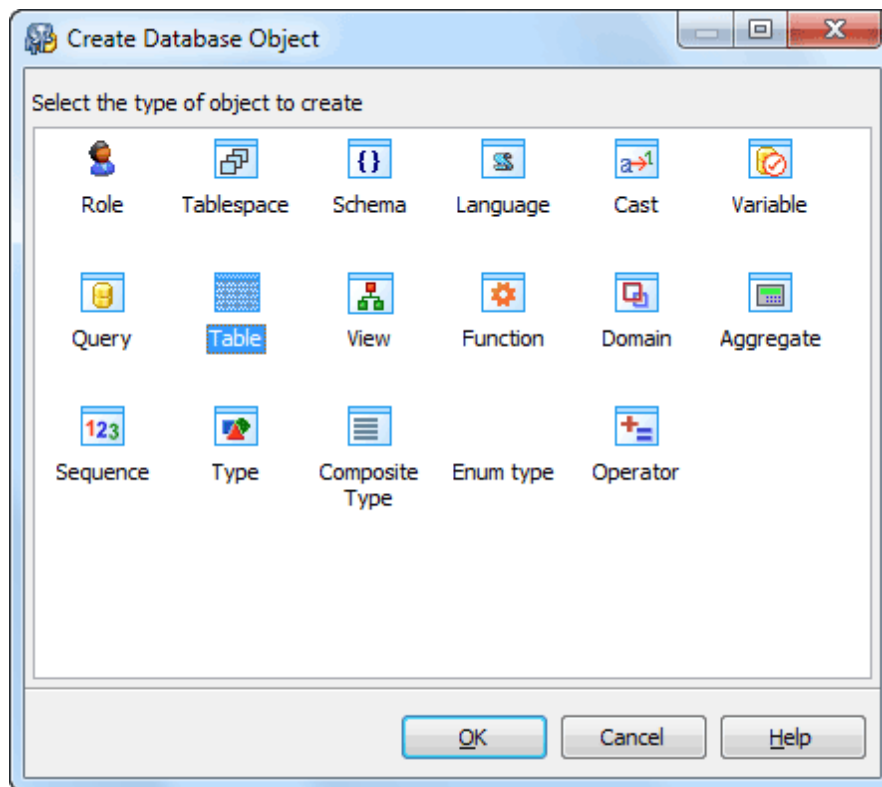
- select the **Object | Create Database Object...** main menu item;
- select the necessary icon (table, query, view, trigger, etc.) in the [Create Database Object](#)⁴³ dialog

or

- select the object list (Tables, Views, Triggers, etc.) or any object from that list in the Explorer tree (**Object Manager** and **Object Browser**);
- select the **Create New Table (View, Trigger, etc.)...** item from the popup menu or press **Insert**.

4.1.1 Create Database Object Dialog

The **Create Database Object** dialog allows you to create any type of database object supported by PostgreSQL Maestro. To open the dialog select the **Object | Create Database Object...** main menu item or use the **Shift+Ctrl+N** hot keys combination. Select an object type icon and click the **OK** button to invoke the corresponding wizard or dialog.



4.1.2 Overview of Create Objects Wizards

Several steps of Create Object Wizards are common for all of them. This part purpose is the formulation of the basic principles for the [Create Object Wizard](#) organization.

- On the [first wizard step](#) ^[44] you need to specify the new object name.
- On the second one you have to define all the object properties. To clear up the object properties meanings see the appropriate topic of the respective Create Object Wizard section.
- Some objects has subitems (e.g. each table contains fields, indexes, procedures have parameters, etc). In this case the next step allows you to manage such subobjects of the object being created. We recommend you to store the following shortcuts in order to speed your work: the Ins key adds a new subobject, the Enter key displays the subobject's editor, and the Del key drops the subobject.
- The [next wizard step](#) ^[46] is final. It is provided to sum up the [Create Object Wizard](#) operation.

Note: There are some objects to have an additional [Create Object Wizard](#) steps. The detailed description of the steps you can find at the appropriate topic of the corresponding section.

See also:

- [Create Schema Wizard](#) ^[68]
- [Create Table Wizard](#) ^[74]
- [Create View Wizard](#) ^[109]
- [Create Function Wizard](#) ^[126]
- [Create Domain Wizard](#) ^[134]
- [Create Aggregate Wizard](#) ^[140]
- [Create Sequence Wizard](#) ^[145]
- [Create Type Wizard](#) ^[149]
- [Create Composite Type Wizard](#) ^[155]
- [Create Operator Wizard](#) ^[163]
- [Create Language Wizard](#) ^[169]
- [Create Cast Wizard](#) ^[173]
- [Creation of a New Query](#) ^[212]
- [Create Database Variable Wizard](#) ^[178]
- [Create Tablespace Wizard](#) ^[209]

4.1.2.1 Setting object name

Select the container (table, schema, database, etc.) for the new object from the list of available containers and enter the new object [name](#) in the respective box.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

Welcome to the Create Table Wizard!
This wizard allows you to set up all the table properties, and create the new table according to these settings.

This wizard will guide you through the process of building the initial table structure and setting its properties.

Schema
NBA

Table name
PLAYER

4.1.2.2 Viewing common information

At this step common information about the object to be created is displayed. Select the [Open object editor after creating option](#) to open the appropriate [Object Editor](#) after the new object is created. Click the [Ready](#) button to complete creation of the object.

The following schema is selected:
NBA

The following table will be created:
PLAYER

☒ Open table editor after creating

Click "Ready" to create a new table.

4.2 Edit Objects

PostgreSQL Maestro allows you to view and modify existing database objects in several ways:

- edit object comment with the [Describe Object](#)^[52] dialog;
- briefly view and modify [object properties](#)^[52];
- view and modify the object including subitems within the object editor.

To open an [Object Editor](#)^[46], just double click its node in the [Database Explorer](#) tree. Of course this action is also available through popup menus, navigation bars, and so on.

4.2.1 Overview of Object Editors

[Database Object Editors](#) are the basic PostgreSQL Maestro tools for working with existing objects. The proper editor can be opened automatically after the object is created. You can also open the necessary object editor with the corresponding items of popup menus of the [Explorer Tree](#)^[58], [Object Manager](#)^[63] or [Object Browser](#)^[61].

The editors consist of a several tabs. Some tabs are similar for all editors. This part purpose is to formulate the basic principles of all [Object Editors in PostgreSQL Maestro](#).

- To edit object options such as name, owner, etc. use the [Properties](#) tab. To understand an option, see the appropriate topic of the corresponding [Object Editor](#) manual section and PostgreSQL documentation.

This tab also allows you to manage objects belonging to the selected one. To reset any tab to default settings, open it when holding the **Ctrl** key.

- Use the [Permissions](#)^[47] tab to manage access privileges (grants) of the corresponding object.
- In a similar manner, some objects called grantees (e.g. users or roles) can have rights to do something with other objects (e.g. a user can read data from a table). This relationship can be set up at the [Grants](#)^[48] tab.
- Object correlation with another PostgreSQL objects is represented on [Dependencies](#)^[49].
- Most of objects have a possibility to be created from an SQL script (SQL definition). If so, the corresponding script is available at the [SQL](#)^[50] tab of the editor.
- There is a [Result](#) tab in editors of such routines as functions and procedures that can take parameters, perform calculations or other actions, and return a result. You can [execute](#)^[51] any routine directly from its editor.

Note: Some object editors have additional tabs. The detailed description of them you can find at the appropriate topic of the corresponding section.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl**

+F9 or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

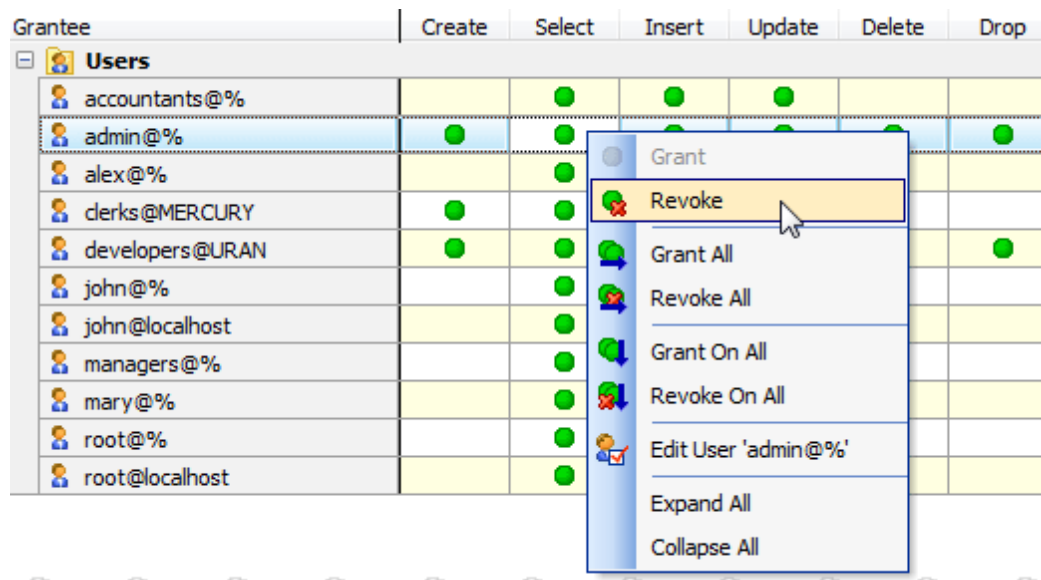
See also:

- [Schema Editor](#) ^[70]
- [Table Editor](#) ^[77]
- [View Editor](#) ^[114]
- [Function Editor](#) ^[128]
- [Domain Editor](#) ^[136]
- [Aggregate Editor](#) ^[141]
- [Sequence Editor](#) ^[146]
- [Type Editor](#) ^[151]
- [Composite Type Editor](#) ^[157]
- [Operator Editor](#) ^[164]
- [Language Editor](#) ^[171]
- [Cast Editor](#) ^[175]
- [Tablespace Editor](#) ^[210]

4.2.1.1 Permissions of the Object

The [Permissions](#) grid allows you to manage access privileges (grants) of users and groups of users.

Grants give specific privileges for an object (*table*, *view*, *sequence*, *database*, *function*, *procedural language*, *schema* or *tablespace*) to one or more users or groups of users.



Using the grid you can grant/revoke privileges as well as sort and filter displayed grantees.

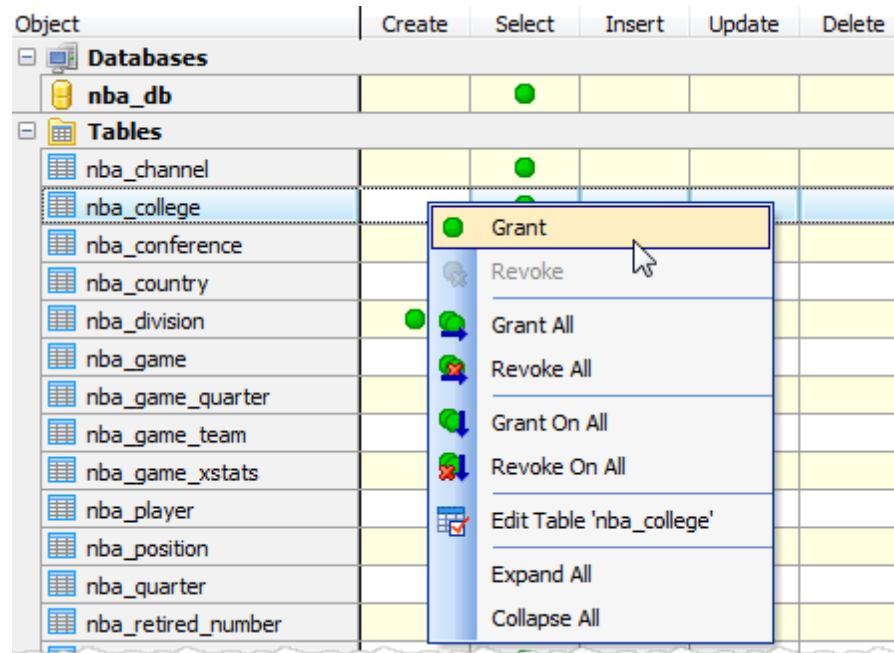
See also: [Users](#) ^[194], [Groups](#) ^[198]

4.2.1.2 Object grants

The Grants grid allows you to manage access privileges (grants) of the current object.

A grant gives specific privileges on an object (*table, view, procedure*) to the current one.

All objects are grouped by kind. Filter the object kinds using the checkboxes at the bottom of the window. Using the grid you may sort and filter data.



To grant the subject privilege on the object double-click an empty field; to revoke the privilege double-click a grant with grant option.

Use grid's popup menu to *grant*, *grant all*, *grant with grant option*, *grant all with grant option*, *grant on all*, *grant on all with grant option*, *revoke*, *revoke all* and *revoke on all*:

- select the **Grant** item to grant the subject privilege on the object;
- select the **Grant All** item to grant all the privileges on the object;
- if the **Grant With Grant Option** item is selected, the recipient of the privilege may in turn grant it to others (without a grant option specified, the recipient cannot do that; at present, grant options can only be granted to individual subject, not to groups or Public);
- select the **Grant All With Grant Option** item to grant with grant options the privilege on all the objects of the kind;
- select **Grant On All** or **Grant On All With Grant Option** to grant or grant with grant options respectively the subject privilege on all the objects;
- to revoke the privilege, all the privileges on the object or the privileges on all the server objects select the **Revoke**, **Revoke All** or **Revoke On All** items respectively.

Using the popup menu you can also collapse or expand all the object kinds.

4.2.1.3 Object Dependencies

When you create complex database structures involving many tables with foreign key constraints, views, triggers, Functions, etc. you will implicitly create a net of dependencies between the objects. For instance, a table with a foreign key constraint depends on the table it references. We tried to create a software to make your work easier, i.e. to give you the tools for efficient database objects management. The [Dependencies](#) tab allows you to control the correlation of objects efficiently.

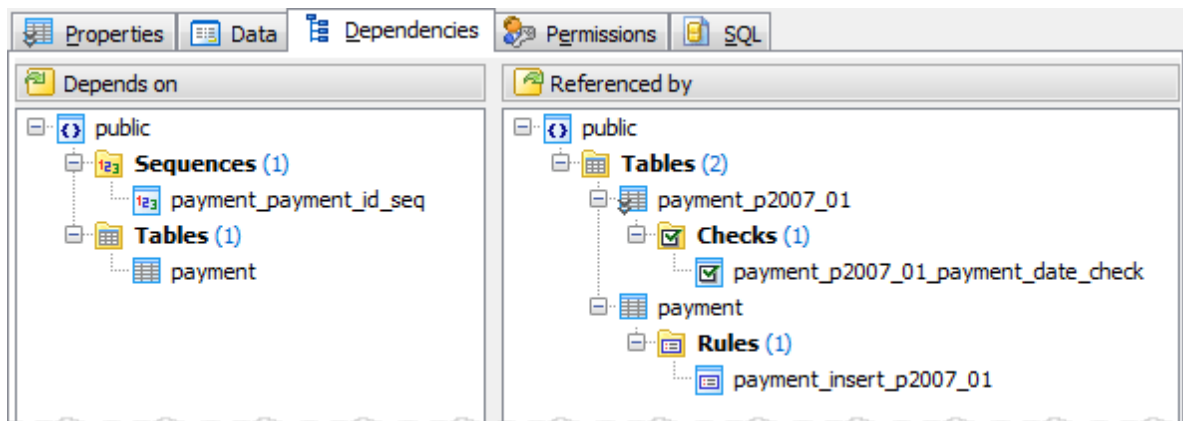
The [Depends on](#) window represents all the objects the current object depends on.

The [Referenced by](#) window contains the tree of database objects constituting the dependency relationship on the current object.

Use items of the popup menu to edit the selected object in [Object Editor](#) or to drop the object.

The optional [Show Details](#) clause of the popup menu specifies whether the types of dependencies are represented.

See also: [Types of dependencies](#)



4.2.1.4 Types of dependencies

This topic contains short descriptions for each type of database object dependencies.

Normal

A normal relationship between separately created objects. The dependent object may be dropped without affecting the referenced object. The referenced object may only be dropped by specifying CASCADE, in this case the dependent object is dropped as well. Example: *a table column* has a normal dependency on its *data type*.

Auto

The dependent object can be dropped separately from the referenced object, and should be automatically dropped (regardless of RESTRICT or CASCADE mode) if the referenced object is dropped. Example: *a named constraint on a table* is made auto dependent on the *table*, so that it will go away if the table is dropped.

Internal

The dependent object has been created as a part of creation of the referenced object, and is really just a part of its internal implementation. A DROP of the dependent object will be disallowed outright (we'll tell the user to issue a DROP against the referenced object instead). A DROP of the referenced object will be propagated through to drop the dependent object whether CASCADE is specified or not. Example: a *trigger* that's created to enforce a *foreign-key constraint* is made internally dependent on the constraint's *pg_constraint* entry.

Pin

There is no dependent object; this type of entry is a signal that the system itself depends on the referenced object, so that the object must never be deleted. Entries of this type are created only by *initdb*. The columns for the dependent object contain zeroes.

4.2.1.5 SQL Definition

The [SQL](#) tab displays the SQL definition for the object with all its properties. Bear in mind that this text is read-only. If you want to change the object definition, use the appropriate editor tabs instead, or copy the text to the Windows Clipboard to paste it in [SQL Editor](#) or [SQL Script Editor](#).

The SQL definition window allows you to browse the text effectively. The popup menu and the extensive system of hot keys give you the opportunity to search expressions within the text, to select the whole text for copying it to the Windows Clipboard, to save the definition to the **.sql* or **.txt* files, to print the document, etc.

You can customize the displayed definition using the [Editors & Viewers](#)³⁴⁴ options.

The [Properties](#) item of the popup menu displays the [Options](#) dialog in which you can establish optional settings concerning the current database.

The [Code Folding](#) item group makes it possible to view either the whole text or its logical parts (regions). Each region can be collapsed and extended.

In [extended mode](#) the whole text is displayed (set by default)

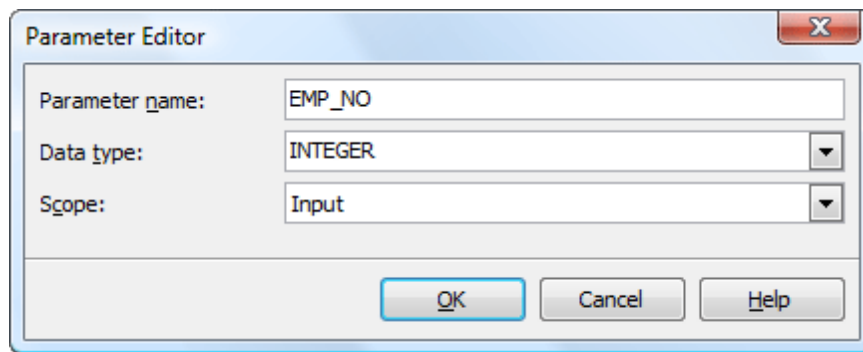
In [collapsed mode](#) the text is hidden behind one text line denoting the first line of the collapsed region.

[Navigation Bar](#) on the [SQL](#) tab allows you to copy the object's SQL definition (DDL) to the [SQL Script Editor](#) for future modifications.

4.2.1.6 Parameter Editor

The editor allows you to change the [Parameter name](#), [Data type](#).

The [Scope](#) box is available for PostgreSQL server 8.1+. Earlier versions support the INPUT value for the function parameters scope only. For PostgreSQL 8.4.+ it's possible to specify *Variadic* parameters. This allows you to write a function with an indefinite number of parameters, as long as they are all the same data type.



The **Parameter Editor** dialog box is shown. It contains three input fields: **Parameter name:** with the value `EMP_NO`, **Data type:** with the value `INTEGER`, and **Scope:** with the value `Input`. At the bottom, there are three buttons: **OK**, **Cancel**, and **Help**.

4.2.1.7 Executing functions and procedures

Function Editor provides an opportunity to execute current routine by opening the **Results** tab, by clicking the **Execute** item of the **Navigation Bar**, or by pressing the **F9** key.

If the Function has parameters, PostgreSQL Maestro will ask you to specify the values for these parameters in the **Input parameters** dialog which appears before the procedure execution. (for PostgreSQL server 8.1+, if the function has parameters marked as IN or INOUT) **Input parameters** dialog allows you to specify the values for all input parameters. After changes are made, click the **OK** button to execute the Function, or the **Cancel** button to abort the execution.

Parameter Values			
⬆⬆ @conference_id int	2		
⬆⬆ @division_id int	2		
⬆⬆ @team_id int	1		
History			
Date and time	@conference_id	@division_id	@team_id
18.02.2010 17:44:44	2	2	1
18.02.2010 17:44:19	1	1	4
18.02.2010 17:42:41	1	3	2
18.02.2010 17:42:18	1	2	7

PostgreSQL Maestro supports **Parameter History**. Values that have been set previously as the routine parameters are represented in the **History** tab of the **Input Parameter** dialog with a date and time of their last using. Double click a necessary set of values to set them as the routine parameters. You can manage the **Parameter History** with **Delete history** item and **Clear history** links.

The result of the successfully executed routine can be found within the **Results** tab of **Function Editor**.

Note: If any unsaved changes are applied to the routine being currently edited, the execution of the routine is impossible until changes are saved by the [Compile](#) procedure item of the [Navigation Bar](#).

4.2.2 Modify Object Properties

You can rename all objects those can be renamed with the corresponding option of the popup menu of the object at the Explorer tree. To edit other properties of the selected object without opening its editor, use the [Object Properties](#) dialog. To open this dialog, select the according item of the same popup menu. To clear up the object properties meanings, see the appropriate topic of the respective [Object Editor](#) section.

4.2.3 Describe Objects

Essentially a comment is the most often altered object property. To simplify it's editing, the PostgreSQL Maestro provides an ability to [Describe the object](#) within the [Database Explorer](#) immediately without opening of the object's editor.

Step-by-step:

- Select the necessary object in the explorer tree;
- Choose the [Describe Object...](#) item in the popup menu;
- Edit object comments within the [Describe Object](#) window;
- To commit the changes, push [OK](#) button.

4.3 Duplicate Objects

PostgreSQL Maestro offers several ways of objects duplicating.

1. **Duplicate Object Wizard.** The wizard is the most flexible tool of the coping. Along with a possibility to adjust the new object definition it allows you to copy data (for tables). But it consists of [several steps](#)^[53] and takes more time than other manners.
2. **Duplicate Object** window allows you to attune new object's SQL definition. It is preferred for creation a copy of selected object. [Here](#)^[56] you can find some additional info.
3. By [Drag-n-Drop](#)^[56] operation.

4.3.1 Duplicate Object Wizard

The [Duplicate Object Wizard](#) allows you to create a new database object with the same properties as the existing one. It is the most flexible tool of copying objects provided by PostgreSQL Maestro. It also allows you to copy data of the selected table to the new one.

To run the wizard select the [Object | Duplicate Database Object...](#) main menu item.

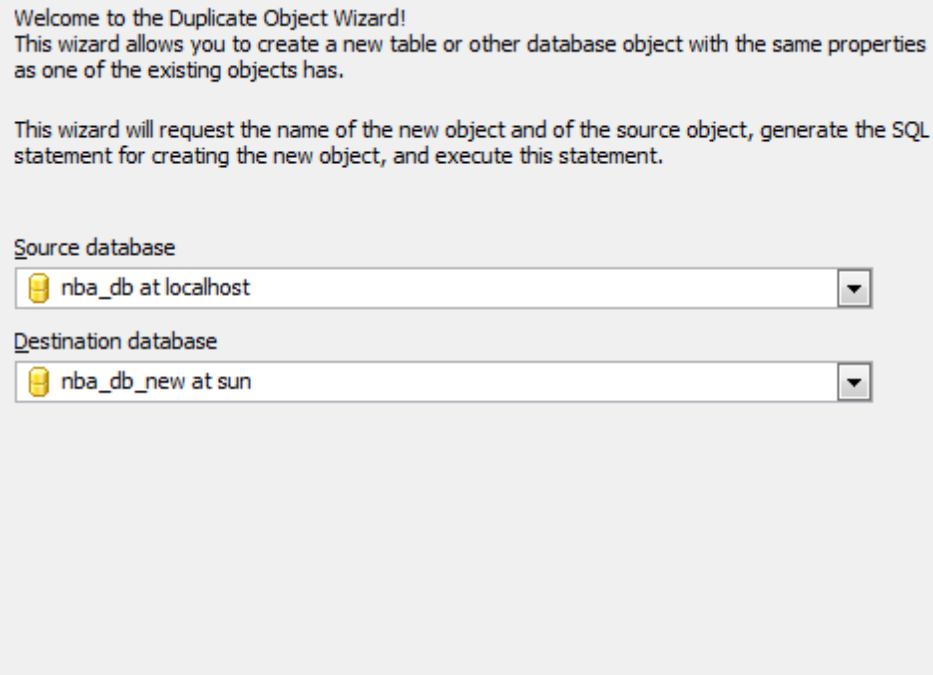
- [Selecting source and destination databases](#)^[53]
- [Selecting object to duplicate](#)^[54]
- [Modifying definition of a new object](#)^[55]

See also: [Create Database Object](#)^[43]

4.3.1.1 Selecting source and destination databases

Select the database containing a source object from the list of connected databases, and then specify the database for the duplicated object.

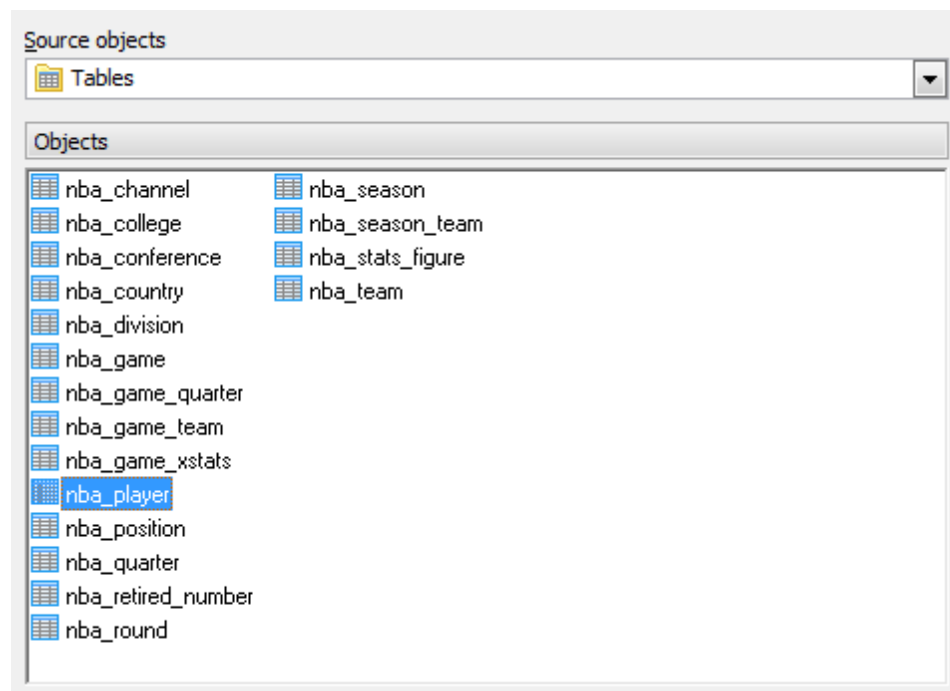
You should connect to the destination database beforehand (see [Database Management](#)^[24]).



4.3.1.2 Selecting object to duplicate

Specify a database object to create the new one with the same properties.

1. Select the type of the object to duplicate from the **Source objects** drop-down list.
2. Pick up the necessary object from the list.



4.3.1.3 Modifying new object definition

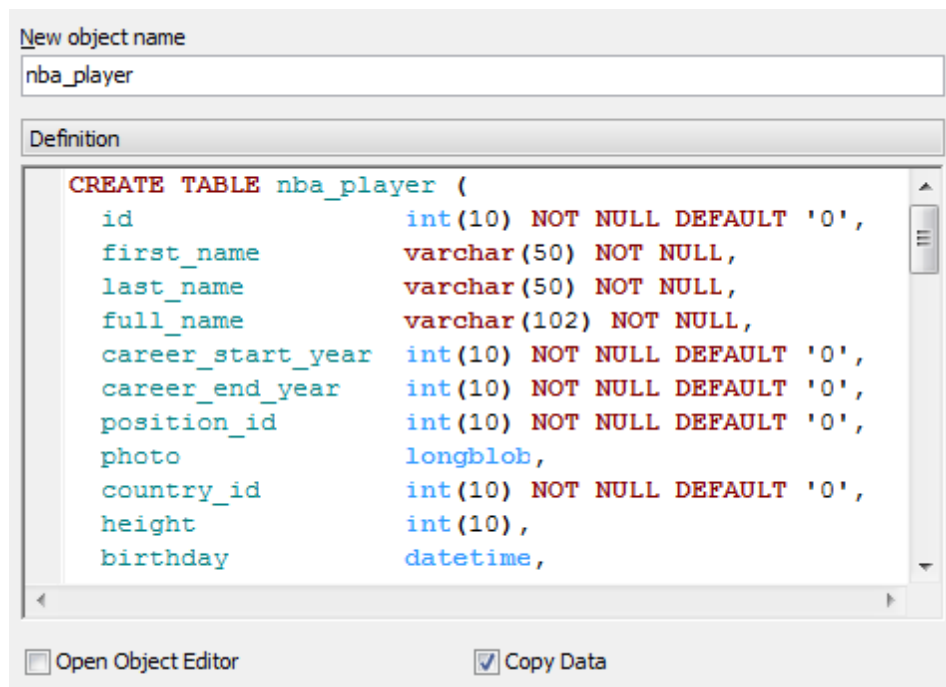
The last wizard step allows you to edit the new object definition directly.

Use this step to edit the name of object been creating ([New Object Name](#)). By default PostgreSQL Maestro generates the new object definition with the same name if the duplicating is to the source database, or like "%SOURCE_OBJECT_NAME%01" otherwise.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

You can edit the result SQL statement manually, add or remove fields, change field types, using the [New object definition](#) text area. Click the [Ready](#) button to complete the operation.

Check the according boxes to [Copy Data](#) (only for tables) and to [Open Object Editor](#) after the duplicating.



New object name

nba_player

Definition

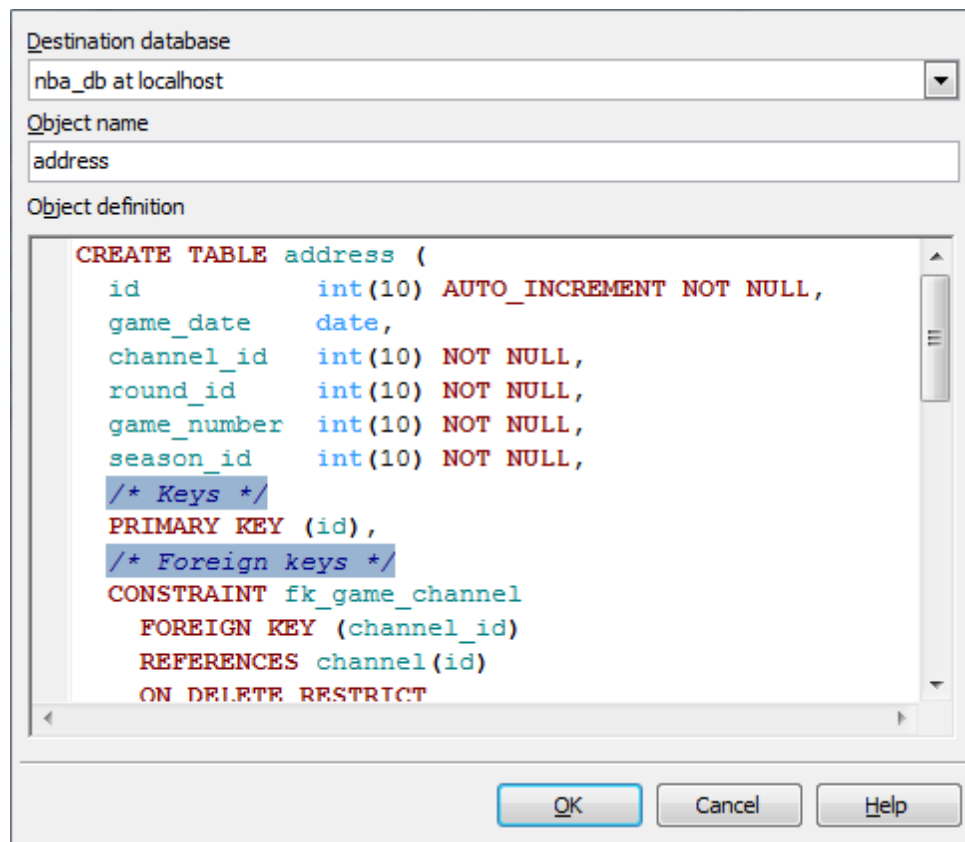
```
CREATE TABLE nba_player (  
  id                int(10) NOT NULL DEFAULT '0',  
  first_name        varchar(50) NOT NULL,  
  last_name         varchar(50) NOT NULL,  
  full_name         varchar(102) NOT NULL,  
  career_start_year int(10) NOT NULL DEFAULT '0',  
  career_end_year   int(10) NOT NULL DEFAULT '0',  
  position_id       int(10) NOT NULL DEFAULT '0',  
  photo             longblob,  
  country_id        int(10) NOT NULL DEFAULT '0',  
  height            int(10),  
  birthday          datetime,
```

☐ Open Object Editor ☒ Copy Data

4.3.2 Duplicate Selected Object

Within the [Duplicate Object](#) window you can duplicate a selected object fast and with some modifications.

It is available from the corresponding link of the object's popup menu at the [Database Explorer](#).



Select the [database](#) for a new object from the list of connected databases first.

Enter the [name](#) for the new object.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

You can also edit the SQL [definition](#) of the object if necessary (add or remove fields, change field types, etc.).

4.3.3 Copy, Paste and Drag-n-Drop features

PostgreSQL Maestro provides you with an ability of copying database objects within the database or even from one database to another (in this case you should connect to both the source and the destination databases first).

To copy an object, just drag the object in a source window (such as [Database Explorer](#), [Object Manager](#), [Object Browser](#)) and drop it to the target container in another window. You also can use the [Edit | Copy](#) and the [Edit | Paste](#) main menu items or the **Ctrl+C**/**Ctrl+V** hot keys combinations respectively. Copying several objects at a time is also available.

It is also possible to drag and drop objects between [Database Explorer](#), [Object Manager](#), [Object Browser](#) and [SQL Editor or SQL Script Editor](#). This works as follows:

SQL Editor: after dropping the object you will get a query to retrieve object data (e.g. `SELECT * FROM table_name`) or the full name of the object if it doesn't contain data (domains, indexes, etc.).

SQL Script Editor: after dropping the object you will get its SQL definition if applicable.

See also: Database Explorer, Object Manager, and Object Browser

4.4 Browse Objects

PostgreSQL Maestro allows to browse objects stored in a Remote Server database in several ways:

- [Database Explorer](#)^[58]: objects are represented as a hierarchy (grouped by kind and listed under the according PostgreSQL servers/database node, provided with subobjects if exist)
- [Object Browser](#)^[61]: an extension of explorer with ability to sort, group, filter and multiple select objects.
- [Object Manager](#)^[63]: an extension of the explorer with ability to select several objects at a time (to copy, drop, etc.)

All tool allows you to drag-and-drop between them and to perform all necessary operations upon database objects.

4.4.1 Database Explorer

Database Explorer is the basic feature of PostgreSQL Maestro which allows you to perform practically all necessary operations upon databases and their objects. The Database Explorer area occupies the left side of the PostgreSQL Maestro main window. All the objects at the Explorer tree are grouped by kind and listed under the according PostgreSQL servers/database node.

To start working with a database you need to create its profile first. The conception of database profiles gives you an opportunity to connect to databases in one touch and work with the selected databases only.

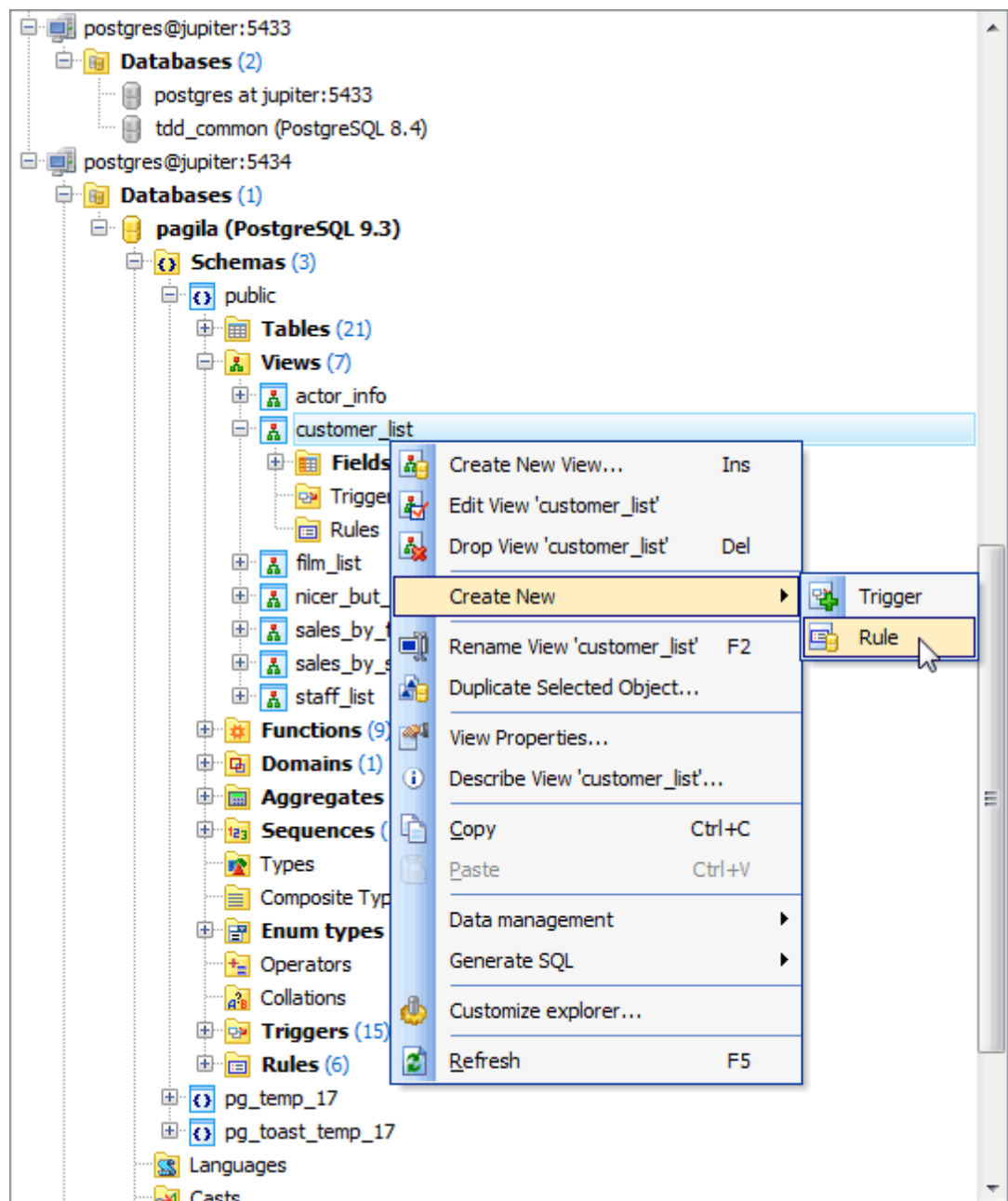
See also: [Object Manager](#)^[63], [Object Browser](#)^[61]

Note: In case your databases have a large quantity of objects you can speed up the object search by typing first letters of the object name in the explorer area.

Note: [Explorer options](#)^[327] allow you to hide/display table subobjects, represent system objects in different color, etc.

The sections below describe each of these actions in detail.

- [What operation can I accomplish upon database profiles within the Explorer Tree?](#)^[59]
- [How can I connect to a database?](#)^[58]
- [How can I disconnect from a database?](#)^[60]
- [What operations can I accomplish upon database objects within the Explorer Tree?](#)^[60]
- [Can I copy a database object from one database to another?](#)^[61]
- [Can I filter Explorer content?](#)^[61]
- [How can I create new/drop a database?](#)^[61]



Operations upon database profiles in the Explorer Tree

Using popup menu of the Explorer area you can realize the following operations:

- [create new database profiles](#) (the Create Database Profiles... item);
- [rename currently selected database profile](#) (the Rename Database Profile... item);
- [edit currently selected database profile](#) (the Edit Database Profile... item);
- [reorder existing database profiles](#) (the Reorder Databases...item of Databases node's popup menu or using drag-n-drop);
- [reorder servers](#) (the Reorder Servers...item of a server's popup menu);
- [remove currently selected database profile from the explorer tree](#) (the Remove Database Profile item);

- remove all profiles of selected server (the Remove all Profiles item of Databases node's popup menu).

In addition to these operations, Database Explorer gives you an ability to reorder existing profiles by performing drag-and-drop operations within the explorer tree.

How can I connect to a database?

You can establish connection to a database in Database Explorer by selecting the database profile and double-clicking it or pressing the Enter key (alternatively, you may use the Shift+Ctrl+C hot key combination). The same operation is also available through the Connect to Database item from the explorer popup menu, or through the Database | Connect to Database main menu item.

How can I disconnect from a database?

You can abort connection from a database in Database Explorer by selecting the database profile and pressing the Shift+Ctrl+D hot key combination. The same operation is also available through the Disconnect from Database item from the explorer popup menu, or through the Database | Disconnect from Database main menu item.

Operations upon database objects

Database Explorer allows you to perform the following operations with database objects using its popup menu (note that the popup menu contains object-specific items only when some database object is currently selected in the explorer tree):

- create a new database object (the Create New Object... item);
- edit currently selected database object (using the Edit Object... item, pressing the Enter key or double-clicking the database object);
- drop the selected object from the database (the Drop Object... item);
- rename the selected database object (the Rename Object... item);
- edit the database object properties (the Object properties ... item);
- duplicate the selected object (the Duplicate Object... item).
- run the Object Browser tool (the Browse ... item).

Can I copy a database object from one database to another?

Database Explorer provides you with an ability of copying database objects from one database to another. To perform this operation, you should connect to both the source and the destination databases first. After the connection is established, simply drag and drop an object to copy from the source database to the corresponding node (Tables, Queries, etc.) of the destination database.

Note: You also can use the Edit | Copy and the Edit | Paste main menu items to copy/paste a database object using Windows clipboard (alternatively, you may use the Ctrl+C/Ctrl+V hot keys combinations respectively).

How can I create new/drop a database?

To create a new PostgreSQL database (not existing on your PostgreSQL server) with Database Explorer, select the Create New Database... item from the popup menu and set

all the necessary options within the [Create Database Wizard](#)^[37].

To drop an existing database using Database Explorer, connect to the database you wish to drop, select the Drop Database item from the popup menu of the database and confirm dropping in the dialog window.

Note: Alternatively, you can use the Database | Create New (Drop) Database main menu item to perform these operations.

4.4.1.1 Filtering explorer content

PostgreSQL Maestro allows you to reduce the number of represented objects in the explorer tree. To hide seldom usable objects, filter your explorer content.

Filter Panel is available through the View | Show Filter Panel main menu item.

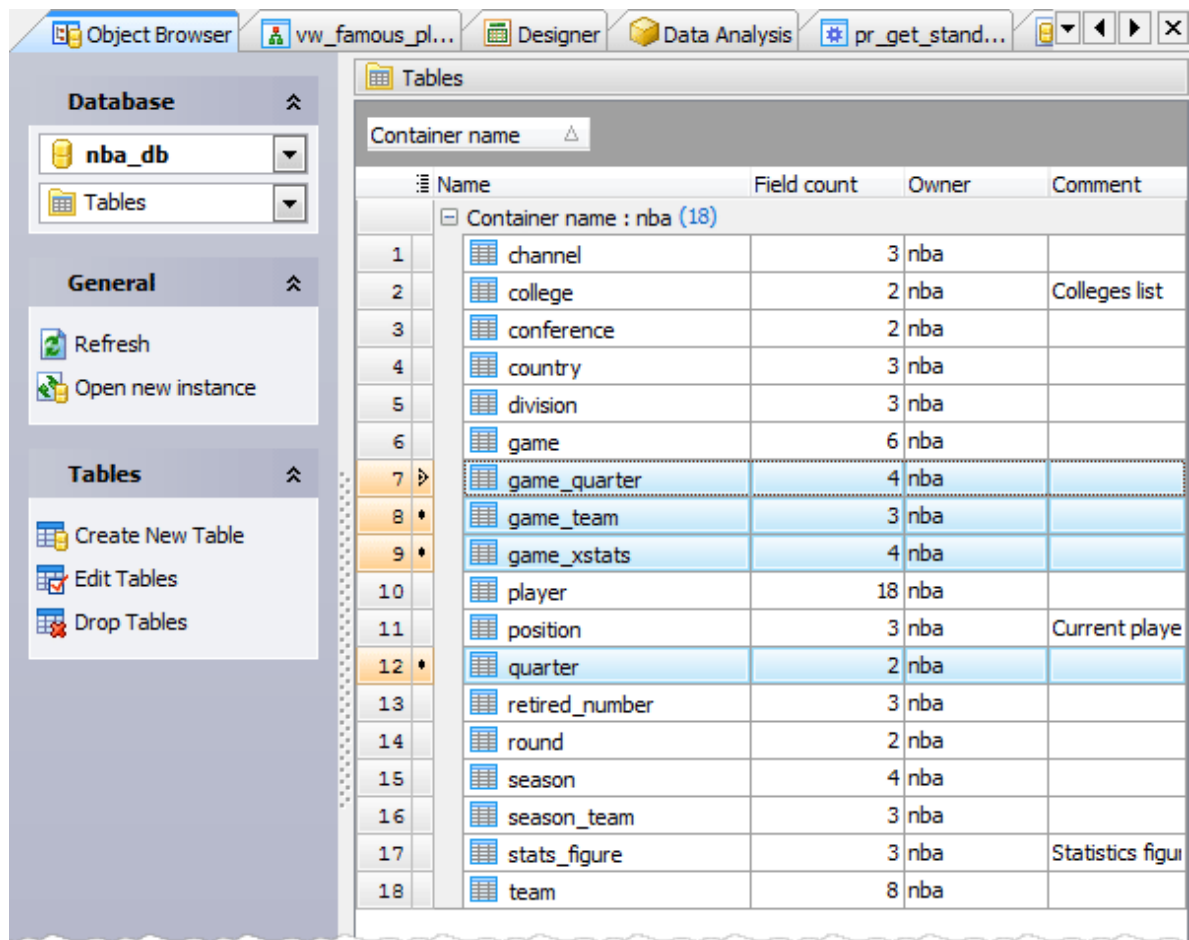
- Specify the Filter expression. The expression can contain any part of object name combined with an asterisk ('*') as a wildcard character and a question-mark ('?') as a mask character.
- Define the Filtered objects, object types for filtering in the explorer tree.
- Check the according radio button (Show by expression, Hide by expression) to define whether database objects will be shown or hidden in accordance with the filter expression.
- Click Apply button.

Note: A filter expression, if applied to the content of Database Explorer, is applied to the content of [Object Manager](#)^[38] and [Object Browser](#)^[39] as well.

4.4.2 Object Browser

[Object Browser](#) is a tool for operating on database objects designed as an extension of [Database Explorer](#)^[38] with ability to *sort*, *group* and *filter* the database objects. It also provides such operation as multiple selecting of objects (for *copying*, *dropping*, etc.) and the ability of using drag-and-drop operations between [Object Browser](#) and [Database Explorer](#). To open [Object Browser](#) select the [Object | Object Browser](#) main menu item.

Note: At least one connection to a database should be established to make [Object Browser](#) available.



Sorting database objects

Object Browser represents database objects in a grid. The object kind to display is defined on the top of the **Navigation bar**. The columns correspond to the objects properties and rows correspond to the objects. Click the column caption to sort objects by the values of this column in the ascending or descending mode. The navigation buttons allow you to open current object editor, create new or drop the existing one.

As **Object Manager** the browser allows you to operate on several objects at a time. You have an opportunity to select a batch of objects and after the object group is selected, you can operate on it (e.g. *drop several objects at once*) as if it were a single object.

The unique feature of the PostgreSQL Maestro is an opportunity of drag-and-drop operations between **Object Browser** and **SQL Editor**, **SQL Script Editor**. After the action objects are represented in **SQL Editor** as SQL queries (if they contain data) or as their full name in the database otherwise. **SQL Script Editor** displays the objects as SQL definition.

Grouping database objects

You can group grid objects by any of the columns by dragging the column header to the destination area. Now all the records are displayed as subnodes to the grouping row value as shown in the picture. To reverse grouping, just drag the column name from the upper area back.

Filtering database objects

You can filter objects in the grid using one of the following methods:

- use the drop-down button in the column caption area to filter objects by the value of the selected column
- click the drop-down button in the column caption area, then select the [Custom](#) item and build a simple filter within the dialog in the following way: select a logical operator for checking the column values (like is less than, is greater than, etc) and set the value to be checked by this operator in the neighboring box; then set the second condition if necessary in the following way and set the relation between these two conditions, whether both of them should be matched or just one of them; use the '_' character to represent any single symbol in the condition and the '%' character to represent any series of symbols in the condition

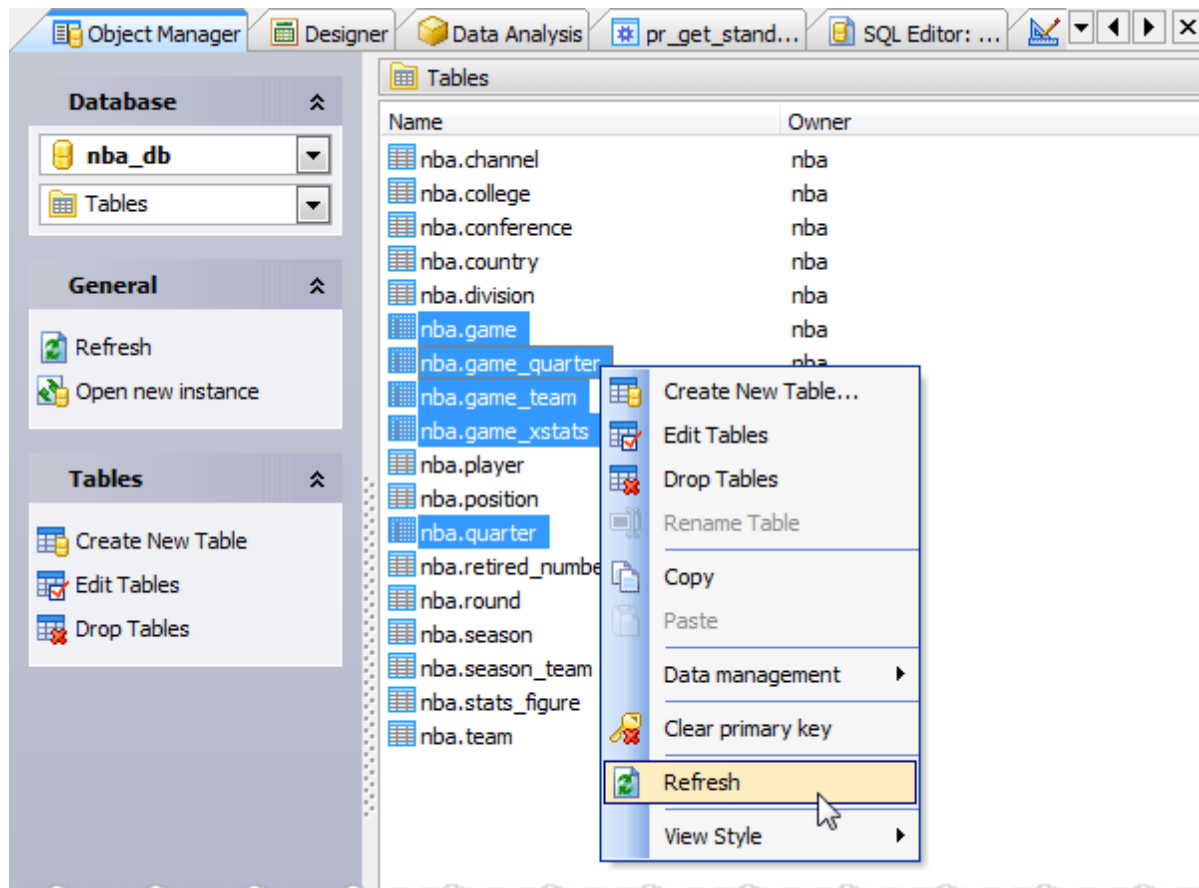
After you set a filter, the filtering panel becomes visible at the bottom of the grid where you can see the active filtering condition and easily enable or disable it by clicking the check box on the left. Using this panel you can also customize your filter in a more complicated way by clicking the [Customize](#) button and building your filter within the Filter Builder dialog.

See also: [Object Manager](#)^[63], [Data View](#)^[229]

4.4.3 Object Manager

[Object Manager](#) is a tool for operating on database objects designed as an extension of the [Database Explorer](#)^[38] with advanced features, such as multiple selecting of objects (for *copying*, *dropping*, etc.) and the ability of using drag-and-drop operations between [Object Manager](#) and [Database Explorer](#) as well as between two instances of the [Object Manager](#). To open [Object Manager](#) select the [Object | Object Manager](#) main menu item.

Note: At least one connection to a database should be established to make [Object Manager](#) available.



Using popup menu

The popup menu of [Object Manager](#) may have different content depending on the current selection. The common menu items allow you to switch the object list view between four standard modes (*large icons*, *small icons*, *list* and *report*), refresh the current view, and select all the objects in the view. If none of objects are currently selected, other menu items are unavailable to use, except of the one for creating a new object. If one or more objects are selected, clipboard operations (such as copy and paste) become available as well as the items for editing and dropping selected object(s). If the current object type of the Object Manager is "Tables", the *Empty Table(s)* menu item is also available.

Multiple selecting of database objects

[Object Manager](#) allows you to operate on several objects at a time. You have an opportunity to select a batch of objects and after the object group is selected, you can operate on it (e.g. *drop several objects at once*) as if it were a single object.

See also: [Object Browser](#) ⁶¹

4.4.4 Filter Builder Dialog

FilterBuilderDialog allows to limit represented objects according to specified conditions. It may be useful for filtering records in data grids of Table Editors, SQL Editor or Visual Query Builder as well as to filter database objects in Object Browser, and on setting a condition on anew view creating. All these cases are similar, see how it works on the

following example.

5 Database Objects


The following list contains database objects supported by PostgreSQL Maestro. To work with database objects you should [connect to the database](#)^[13] first.

- [Schemas](#)^[67]
- [Tables](#)^[73]
- [Views](#)^[108]
- [Functions](#)^[125]
- [Domains](#)^[133]
- [Aggregates](#)^[139]
- [Sequences](#)^[144]
- [Types](#)^[148]
- [Composite Types](#)^[154]
- [Operators](#)^[162]
- [Languages](#)^[168]
- [Casts](#)^[172]
- [Database Variables](#)^[177]
- [Extensions](#)^[180]
- [Foreign Tables](#)^[182]
- [Foreign Servers](#)^[185]
- [User Mappings](#)^[186]

5.1 Schemas

Schemas are a purely logical structure and the privilege system determines whether one can access them. A database is a collection of schemas and the schemas contain tables, views, Functions, etc. The full hierarchy is: server, database, schema, table (or some other kind of object, such as a Function). [Schemas](#) are being implemented in PostgreSQL Server version 7.3 and higher.

■ How can I add a new schema?

New schemas are created within [Create Schema Wizard](#). In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Schema](#) icon in the [Create Database Object](#) dialog
- or
- select the [Schemas](#) list or any object from that list in the explorer tree;
 - select the [Create New Schema...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Schemas](#) tab there;
 - press the **Insert** key or select the [Create New Schema](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new schema with the same properties as one of the existing schemas has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing schema?

Schemas are edited within [Schema Editor](#). In order to run the editor you should either

- select the schema for editing in the explorer tree (type the first letters of the schema name for quick search);
 - select the [Edit Schema...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Schemas](#) tab there;
 - select the schema to edit;
 - press the **Enter** key or select the [Edit Schema](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the schema using the [Rename Schema](#) dialog. To open the dialog you should either

- select the schema to rename in the explorer tree;
- select the [Rename Schema](#) item from the popup menu

or

- open the database in [Database Editor](#) and the [Schemas](#) tab there;
- select the schema to rename;
- select the [Rename Schema](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

📌 **How can I drop an existing schema?**

To drop a schema:

- select the schema to drop in the explorer tree;
- select the [Drop Schema](#) item from popup menu

or

- open the database in [Database Editor](#) and the [Schemas](#) tab there;
- select the schema to drop;
- press the **Delete** key or select the [Drop Schema](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.1.1 Create Schema Wizard

[Create Schema Wizard](#) guides you through the process of creating a new database schema.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

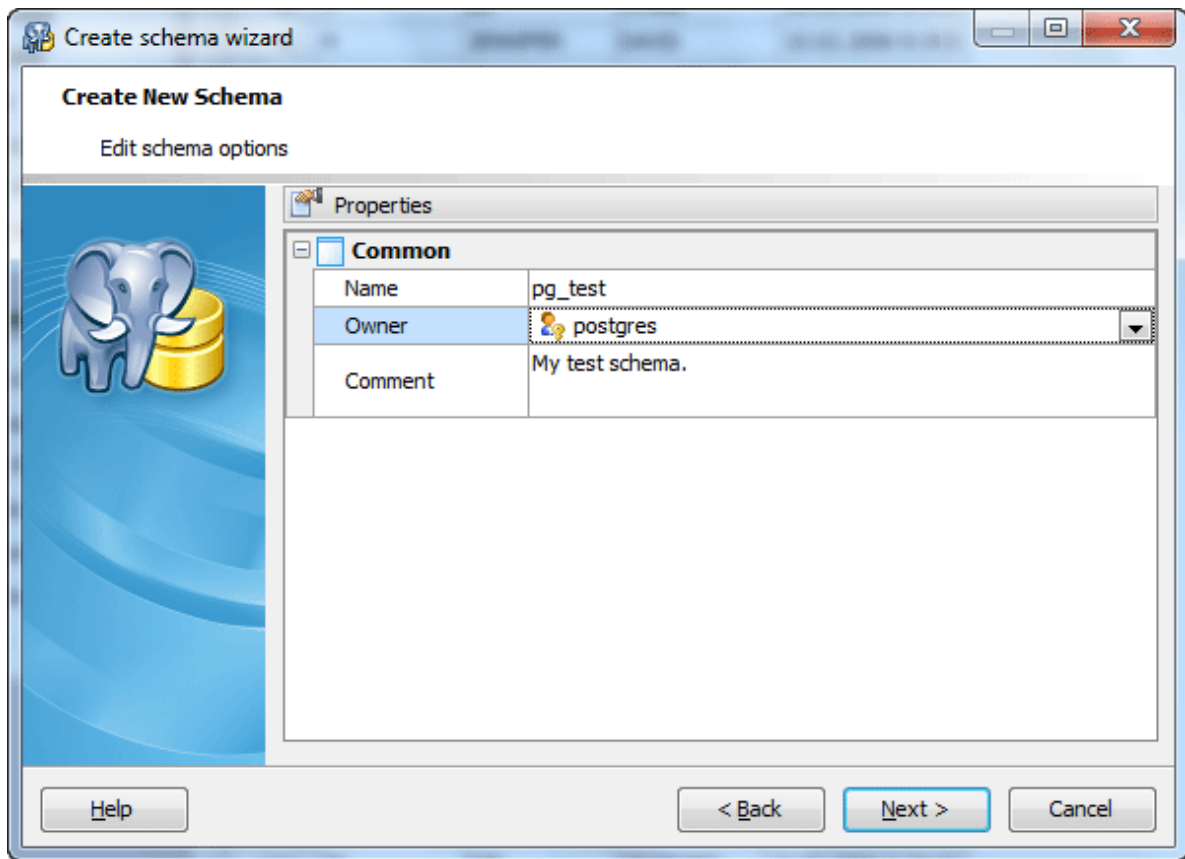
Schema options

[Owner](#)

Use the field to specify the owner of the new schema. The default owner is the user who have created the schema. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

[Comment](#)

The box allows you to set optional text describing the new schema.



Adding schema content

This wizard step allows you to create the new schema along with schema content. To add an object to the new schema:

- Open the corresponding tab ([Tables](#) - to manage schema tables, [Views](#) - to manage schema views, and so on);
- Use the [Create Object Wizard](#) link of the tab's pop-up menu or press **Insert**;
- Complete the corresponding create object wizard. To find out the wizards description, read the corresponding topics:

[Create Table Wizard](#) ⁷⁴

[Create Function Wizard](#) ¹²⁶

[Create Domain Wizard](#) ¹³⁴

[Create Aggregate Wizard](#) ¹⁴⁰

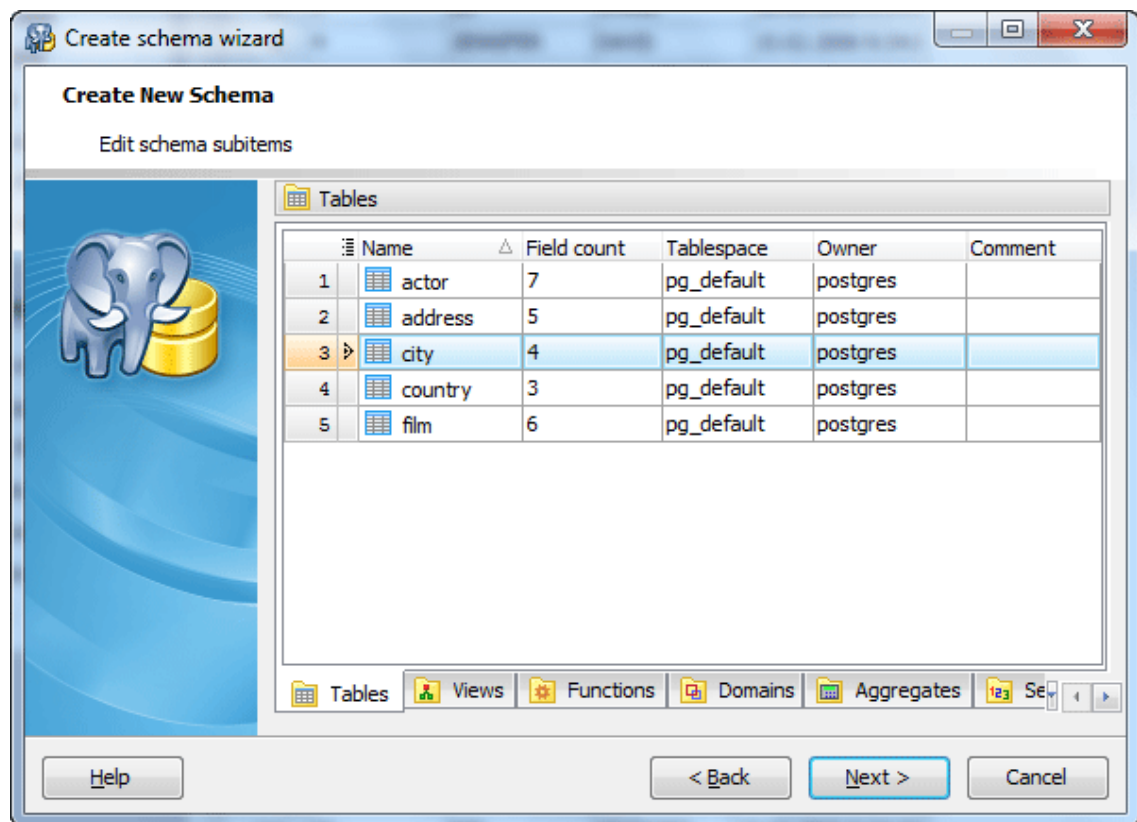
[Create Sequence Wizard](#) ¹⁴⁵

[Create Type Wizard](#) ¹⁴⁹,

[Create Composite Type Wizard](#) ¹⁵⁵

[Create Operator Wizard](#) ¹⁶³

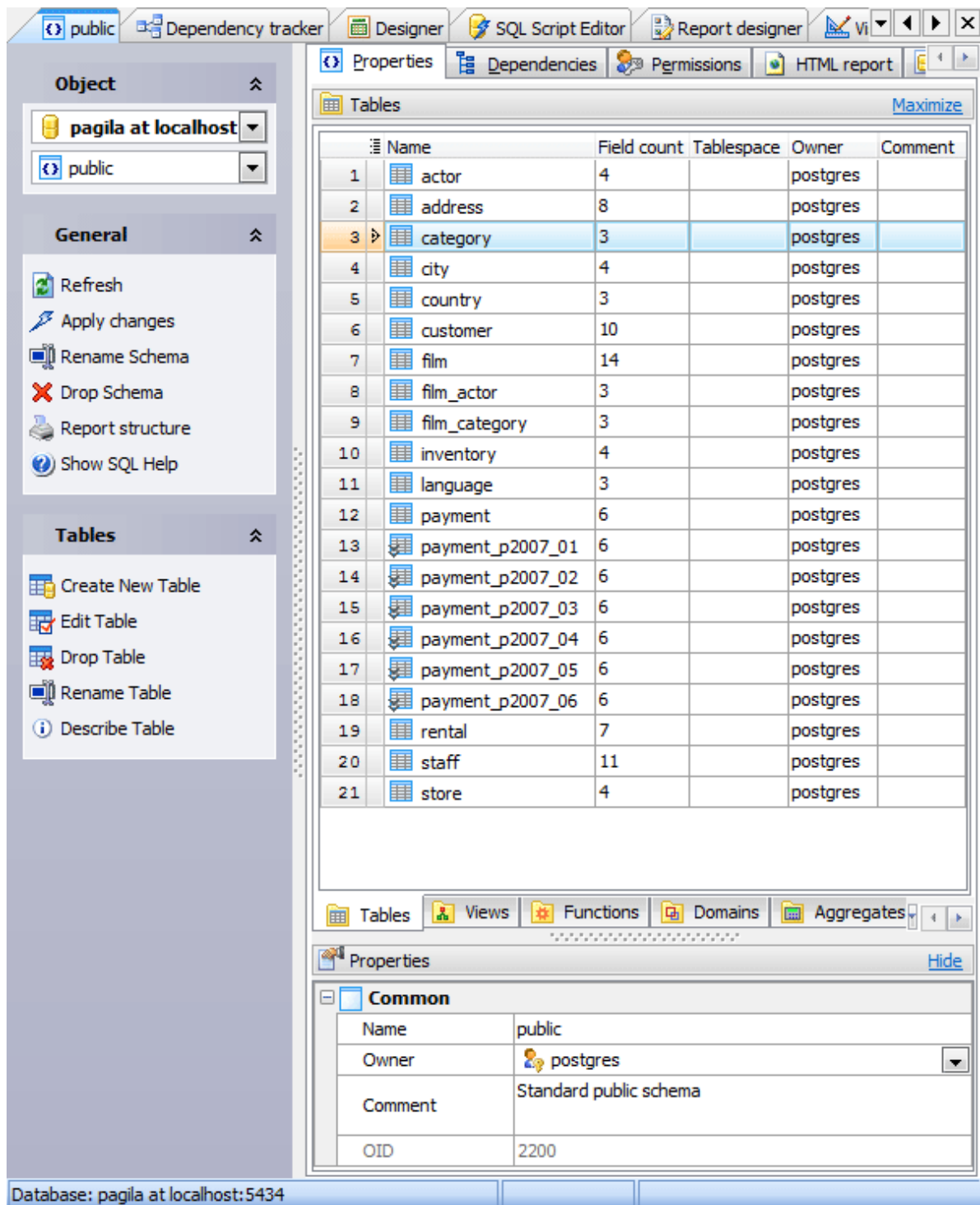
[Create View Wizard](#) ¹⁰⁹



5.1.2 Schema Editor

Schema Editor allows you to browse schema content, manage users permissions on the schema objects, and see the SQL definition of this schema. To open the editor, use the corresponding items of popup menus of the [Explorer Tree](#)^[58], [Object Manager](#)^[60] or [Object Browser](#)^[61].

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.



The **Properties** tab allows you to view schema options and to browse schema content divided into groups according to their types (*tables, views, functions, etc.*). The popup menu of each tab allows you to create new, edit, copy or drop the appropriate schema object. The grid allows you to operate with several objects at a time. For this purpose select objects with the **Shift** or the **Ctrl** key pressed. After a group of objects is selected you can operate with them, e.g. *delete several objects* at once, as if it is a

single object.

Name

Here you can change the schema name. The name of the schema must be unique among all the schema names in the database.

Owner

This field allows you to modify the schema owner. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Use the [Comment](#) field to set the schema description.

OID

In this field the schema OID (object identifier) is displayed. It is represented by a serial number that is automatically added by PostgreSQL to all schemas.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.2 Tables

PostgreSQL Maestro allows you to manipulate tables with easy: add new tables to the database, modify existing ones, browse table options and data. The sections below describe each of these actions in detail.

■ How can I add a new table?

New tables are created within [Create Table Wizard](#)^[74]. In order to run the wizard you should either

- select the **Object | Create Database Object...** main menu item;
- select the **Table** icon in the **Create Database Object** dialog

or

- select the **Tables** list or any object from that list in the explorer tree;
- select the **Create New Table...** item from the popup menu.

To create a new table with the same properties as one of the existing tables has:

- select the **Object | Duplicate Database Object...** main menu item;
- follow the instructions of **Duplicate Object Wizard**.

■ How can I work with an existing table?

Tables can be edited within [Table Editor](#)^[77]. In order to run the editor you should either

- select the table for editing in the explorer tree (type the first letters of the table name for quick search);
- select the **Edit Table...** item from the popup menu

or

- open **Schema (Database) Editor** and the **Tables** tab there;
- select the table to edit;
- press the **Enter** key or select the **Edit Table** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

You can also view and edit table properties without launching **Table Editor**:

- select the table for editing in the explorer tree (type the first letters of the table name for quick search);
- select the **Table Properties...** item from the popup menu;
- edit table properties within the **Table Properties** dialog.

You can change the name of the table using the **Rename Table** dialog. To open the dialog you should either

- select the table to rename in the explorer tree;

- select the [Rename Table](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Tables](#) tab there;
 - select the table to rename;
 - select the [Rename Table](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop the existing table?

To drop a table:

- select the table to drop in the explorer tree;
 - select the [Drop Table](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Tables](#) tab there;
 - select the table to drop;
 - press the **Delete** key or select the [Drop Table](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

Table Editor allows you to work with table data including [master-detail views](#)^[78], generate [simple SQL statements](#)^[309], [CRUD procedures](#)^[310] to work with this table, and [split the table](#)^[313] into two separate tables.

5.2.1 Create Table Wizard

[Create Table Wizard](#) guides you through the process of creating a new database table. The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

[Name](#)

The name of the table being created as it was specified at the previous step.

[Owner](#)

You can specify here the name of the PostgreSQL server user that will own the new table, or leave this field blank to use the default user (namely, the user executing the command). By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

[Comment](#)

Set the optional text to describe the new table.

[RLS policy \(PostgreSQL 9.5+\)](#)

Allows you to manage the row-level policy for the selected table. Possible values are Disabled, Enabled, Enabled and Forced.

☒ With OIDs

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

Tablespaces

Set the tablespace name to associate with the new table, or leave this field blank to use the template table tablespace. This tablespace will be the default one used for objects created in this table. The default tablespace is based on the template table tablespace.

Fill factor (Since PostgreSQL 8.2)

This storage parameter 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.

Partitioning (PostgreSQL 10+)

Allows you to specify partitioned properties for the table. Description of each property is listed below.

Is Partitioned

Turn this option ON if you want to create a partitioned table.

Strategy

Allows you to specify the partitioning strategy (Range or List)

Partition key

Enter column name or an expression to be used as partition key

Is Partition

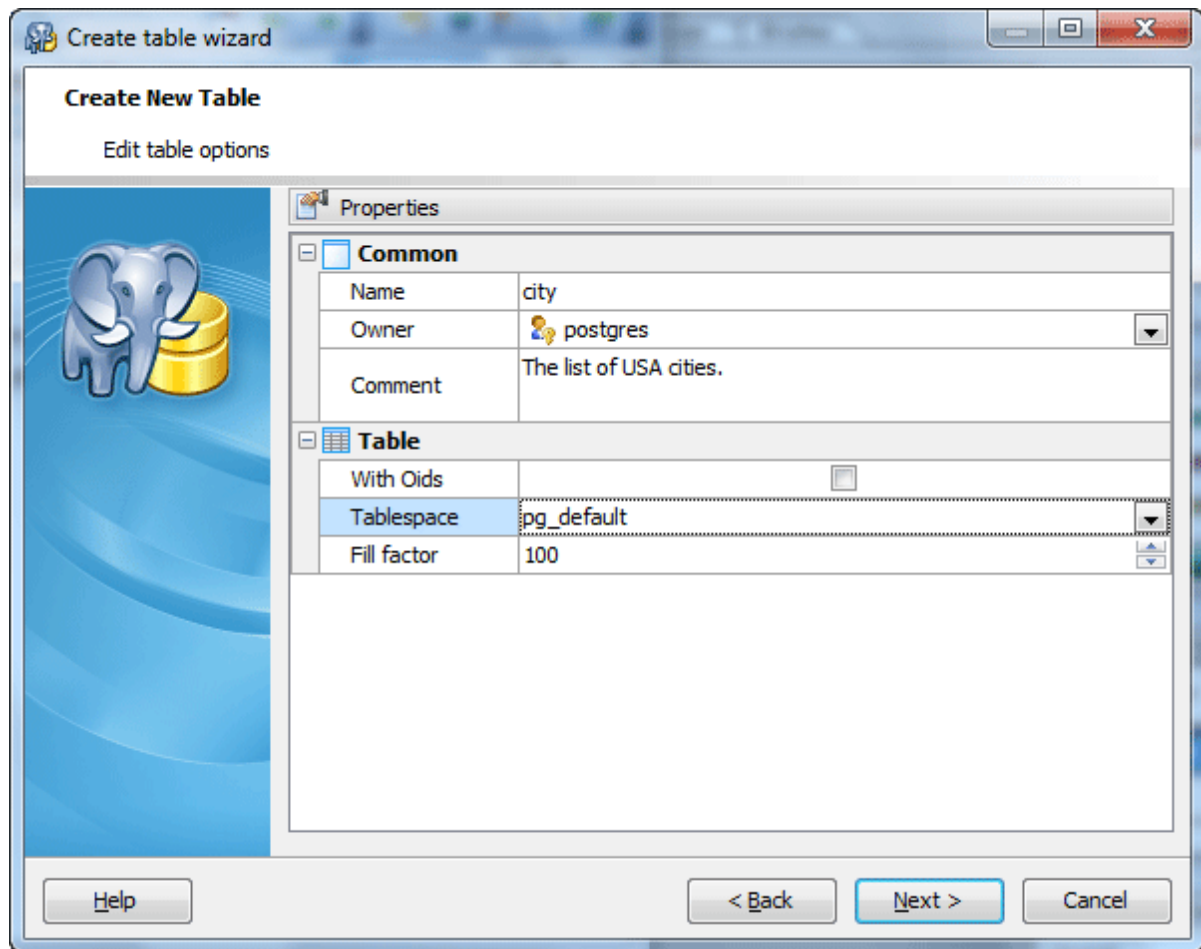
Turn this option ON if you want to create a partition.

Parent table

Specify a partitioned table for the new partition

Bound expression

Provide an expression for the new partition

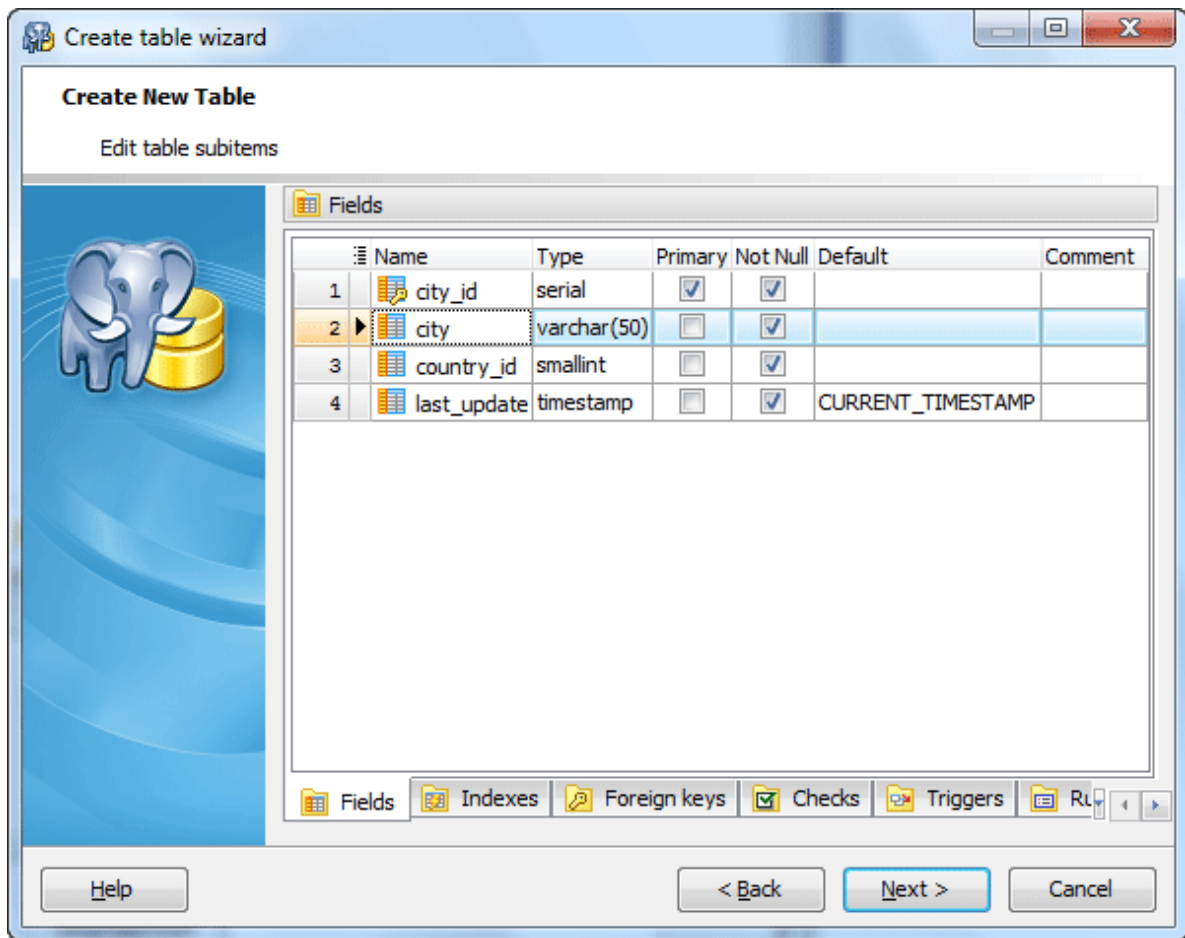


Adding table subitems

On this step of the wizard you can fulfill the new table with fields, indexes, and foreign keys. To add a new object:

- Choose the necessary page ([Fields](#) - to add table fields, [Indexes](#) - table indexes, and so on);
- Follow the corresponding link of the tab's pop-up menu;
- Specify properties of the new object. To find the description of [field](#)^[82], [foreign key](#)^[88], [check](#)^[91], [trigger](#)^[97], [rule](#)^[101], and [index](#)^[84], follow the according link. As Parent tables specify a list of tables from which the new table automatically inherits all columns.

The popup menu of each tab allows to edit, drop, reorder, and rename specified objects, etc.



Click [Add All](#) or [Add](#) to include table(s) to table definition. Use the [Remove](#) or [Remove All](#) items to exclude table(s) from the list.

5.2.2 Table Editor

[Table Editor](#) allows you to create, edit and drop table fields, indexes, foreign keys, manage table data and other table subobjects. It can be opened automatically after the table is created and is available on editing the table. To open [Table Editor](#), double-click the corresponding node at the [Explorer Tree](#) or [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

- [Editing table properties](#)^[77]
- [Viewing table data](#)^[78]

5.2.2.1 Editing table properties

The [Properties](#) section allows you to view general table properties and also to modify the table name, the table owner, and a comment for the table.

Properties Data Dependencies Permissions SQL

Fields [Maximize](#)

	Name	Type	Primary	Not Null	Default	Comment
1	customer_id	serial	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('customer_	
2	store_id	smallint	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	first_name	varchar(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	last_name	varchar(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
5	email	varchar(50)	<input type="checkbox"/>	<input type="checkbox"/>		
6	address_id	smallint	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
7	activebool	boolean	<input type="checkbox"/>	<input checked="" type="checkbox"/>	true	
8	create_date	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>	CURRENT_DATE	
9	last_update	timestamp without time	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMEST	
10	active	integer	<input type="checkbox"/>	<input type="checkbox"/>		

Fields Indexes Foreign keys Checks Triggers Rules References

Properties [Hide](#)

☐ **Common**

Name	customer
Owner	postgres
Comment	
OID	106896

☐ **Table**

RLS policy	Disabled
With Oids	<input type="checkbox"/>
Inherited from	
Tablespace	pg_default
Fill factor	100

☐ **Partitioning**

<input type="checkbox"/> Is partitioned	<input type="checkbox"/>
Strategy	Range
Partition key	
<input type="checkbox"/> Is partition	<input type="checkbox"/>
Parent table	
Bound expression	

Subitems

Every tab is intended for work with defined **objects** (*fields, indexes, etc.*). To modify any object, double click it or use grid's popup menu. The menu also allows you to add new, rename, describe, copy/paste, and drop selected objects. To operate with several objects at a time, select them with the **Shift** or the **Ctrl** key pressed. After a group of objects is selected you can operate with it, e.g. *delete several objects at once*, as if it is a single object.

See also: [Fields](#)^[82], [Foreign Keys](#)^[88], [Checks](#)^[91], [Triggers](#)^[98], and [Indexes](#)^[84].

Owner

You can view and modify the name of the table [Owner](#).

By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

This field contains a comment to the table.

OID

In this field the table OID (object identifier) is displayed. It is represented by a serial number that is automatically added by PostgreSQL to all tables.

RLS policy (PostgreSQL 9.5+)

Allows you to manage the row-level policy for the selected table. Possible values are Disabled, Enabled, Enabled and Forced.

With OIDs

The checkbox determines whether the data rows will be created with OIDs. Physically it means that the table will have an additional column - OID.

Inherited from

Here is a list of tables the current table is inherited from, i.e. the list of tables which are used for implementing their fields in the new table upon its creation.

Tablespace

Tablespaces define locations in the file system where the files representing table objects can be stored. The blank field means that the table tablespace was created using the template table tablespace.

Fill factor (PostgreSQL 8.2+)

This storage parameter 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.

Partitioning (PostgreSQL 10+)

Values in this group display partitioning properties of the table. They are not applied to regular tables and cannot be changed for existing tables.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.2.2.2 Managing table data

The [Data](#) tab displays the table data as a grid or as info cards (see [Data View](#)²²⁹ for details). To edit/add a table record, use [Data Input Form](#) or type the new data directly

in the grid (card). To export/import/get SQL dump of the table data, invoke corresponding modules from the grid's popup menu. To view and edit the content of BLOB columns, run [BLOB Editor](#)^[240].

Lookup editors

Lookup editor displays the content of parent table's columns within the drop-down window. PostgreSQL Maestro enables a lookup editor for a column linked by a foreign key with a single column from another table. To get the corresponding data, double click the field or use **F2** shortcut and press **Alt+Down Arrow Key**.

1	7	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	1
2	8	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	1
1	9	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	1
2	address_id	address	district	city_id	postal_code
1	5	1913 Hanoi Way	Nagasaki	463	35200
2	6	1121 Loja Avenue	California	449	17886
2	7	692 Joliet Street	Attika	38	83579
1	8	1566 Inegl Manor	Mandalay	349	53561
2	9	53 Idfu Parkway	Nantou	361	42399
1	10	1795 Santiago de Compostela Way	Texas	295	18743
2	11	900 Santiago de Compostela Parkway	Central Serbia	280	93896
2	12	478 Joliet Way	Hamilton	200	77948
1					
2					
1	21	DONNA	THOMPSON	DONNA.THOMPSON@sakilacustomer.org	1

Master-Detail Data View

To get data in the [master-detail](#) view mode (multiple detail pages are displayed for a single master row), use the [Show/Hide details](#) link at the editor's navigation bar. This mode allows you add/edit/delete data of detail pages. To open/close the appropriate detail page click the +/- icon or use +/- shortcuts.

6	+	710	Mountain Bike Socks, L	SO-B909-L	White	3,3963	9,5
7	+	711	Sport-100 Helmet, Blue	HL-U509-B	Blue	13,0863	34,99
8	-	712	AWC Logo Cap	CA-1098	Multi	6,9223	8,99
1 SalesOrderDetail (ProductID)							
SalesOrderID SalesOrderDetailID OrderQty ProductID UnitPrice UnitPriceDisc							
Click here to define a filter							
1		71938	113283	1	712	5,394	
2		71897	112902	4	712	5,394	
3		71858	112375	3	712	5,394	
4		71902	112962	3	712	5,394	
5		71797	111053	6	712	5,394	
6		71816	111457	4	712	5,394	
7		71784	110761	10	712	5,394	
8		71783	110748	11	712	5,2142	
9		71782	110670	10	712	5,394	
9	+	713	Long-Sleeve Logo Jersey, S	LJ-0192-S	Multi	38,4923	49,99
10	+	714	Long-Sleeve Logo Jersey, M	LJ-0192-M	Multi	38,4923	49,99
11	+	715	Long-Sleeve Logo Jersey, L	LJ-0192-L	Multi	38,4923	49,99

Import from Clipboard

PostgreSQL Maestro supports data import from clipboard. It is supposed that columns within the data block are separated by the tabulation symbol, records are separated by newlines and the first line of the data block contains column headers.

Example:

ColHeader1 ColHeader2

R1C1 R1C2

R2C1 R2C2

The same data format is supported by a lot of other applications, so the ability allows you to copy data from MS Excel, another table or view, or even from a data set from a different DBMS especially if it is opened with an appropriate our product.

Uploading files as BLOBs

PostgreSQL Maestro allows you to upload files as BLOBs into a table. For this purpose the file names must contain the information on the record they need to be placed to: the files need to be named in the same manner and include content of one or several table columns that can uniquely identify each row. To import files, specify the file name template using file name tags (i.e. %id%, %user%, where 'id' and 'user' are the fact table columns). You can also set the default file to be uploaded to NULL fields.

Example:

Suppose we have a table 'employee' with Non-Blob data as follows:

Id User

1 Max

2 July

And we need to import the 1.jpg and 2.jpg files to a BLOB column of the table. The files are stored in the "D:\Images" directory. In this case we need to specify the "D:\Images\%Id%.jpg" file name template.

5.2.3 Fields

Table columns and composite type fields are created and edited within the [Field Editor](#). The information below are useful for editing of table fields. Composite types' fields admit to editing the comment only.

■ How to add a new column to a table?

To add a new table column, you should either:

- open the table in [Table Editor](#) and the [Fields](#) tab there;
- press the **Insert** key or select the [Add New Field...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))


or

- select the table in the explorer tree and use the [Create New Field](#) popup menu item

or

- select the table [Fields](#) node or any field within the table in the explorer tree and use the [Add New Field...](#) popup menu item.

■ How to edit an existing table field?

Table fields are edited within the [Field Editor](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Fields](#) tab there;
- press the **Enter** key or select the [Edit Field](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the field to edit in the explorer tree and use the [Edit Field](#) popup menu item.

You can change the name of the field using the [Rename Field](#) dialog. To open the dialog you should either

- select the field to rename in the explorer tree;
- select the [Rename Field](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Fields](#) tab there;
- select the field to rename;
- select the [Rename Field](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How to drop an existing table field?

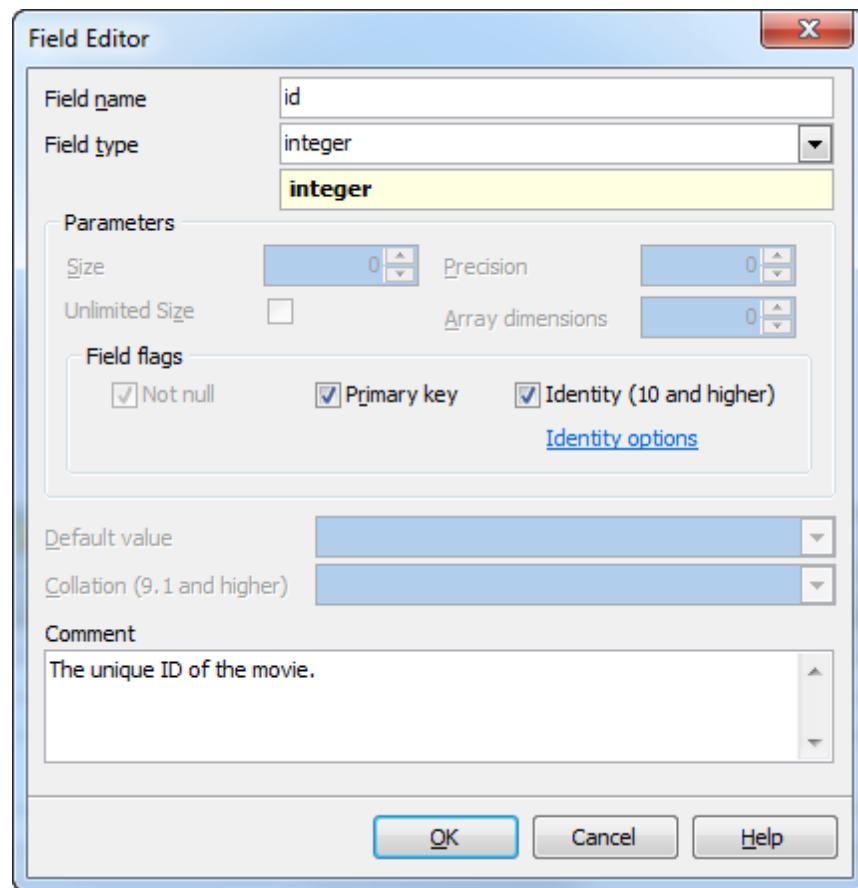
To drop the table field:

- select the field to drop in the explorer tree;
- select the [Drop Field](#) item from the popup menu

or

- open the table in Table Editor and the [Fields](#) tab there;
- press the **Delete** key or select the [Drop Field](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

The image shows a 'Field Editor' dialog box with a title bar and a close button. It contains several sections: 'Field name' with a text box containing 'id'; 'Field type' with a dropdown menu showing 'integer' and a list of other types below it; 'Parameters' with 'Size' and 'Precision' spinners, an 'Unlimited Size' checkbox, and 'Array dimensions' spinner; 'Field flags' with checkboxes for 'Not null', 'Primary key', and 'Identity (10 and higher)', plus a link for 'Identity options'; 'Default value' and 'Collation (9.1 and higher)' dropdowns; and a 'Comment' text area containing 'The unique ID of the movie.'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

To specify the [Data Type](#), select it from the drop-down list.

Note: the name of the object must be unique among all the object names in the table. You can use any identifier that is allowed by PostgreSQL server.

Parameters

Use the [Size](#) edit box to define the length of the field value for integer, float, char and other data types and use [Precision](#) to define the precision of the field value, e.g. for *float* data type.

Check the [Unlimited Size](#) option to remove value limitations for VarChar, Timestamp, TimestampTZ, Interval, Time, TimeTZ, VarBit, Decimal data types.

Use the [Array dimensions](#) to set the dimension of data array.

Check the [Use dimension](#) option to set the dimension properties for field types with unnecessary dimension properties (e.g. integer, float, or timestamp types).

Field flags

☒ Not Null

Forbids the NULL values for the field.

☒ Primary Key

With this option checked the field becomes the only field with a primary key. If you check this field, you will not be able to set this attribute for any other field in the table. Hence if you want to create a compound primary key, do not check this field but create a primary key through the Indexes tab of [Table Editor](#)^[77] or the appropriate step of [Create Table Wizard](#)^[74].

☒ Identity (PostgreSQL 10+)

Indicates that the new column is an identity column. Click the "Identity options" link to customize the appropriate properties.

[Expression](#) (PostgreSQL 12+)

To set a generated column, define here the generation expression. Check the [Stored](#) box to choose the stored kind of the generated column. To learn more about the implementation of generated columns in PostgreSQL, see the [PostgreSQL documentation](#).

[Collation](#) (PostgreSQL 9.1+)

Specify a collation to the column to define the sort order and character classification behavior of column data. The collation must be of a collatable data type. If not specified, the column data type's default collation is used.

[Default value](#)

Within the box you can assign a default value for the field column. The action is optional. If the default value was specified during the new row created and no values is specified for some of the columns, the columns will be filled with their respective default values.

The [Comment](#) box allows you to set optional text describing the field.

5.2.4 Indexes

[Indexes](#) are primarily used to enhance database performance (though inappropriate use may result in slower performance). The key field(s) for the index are specified as column names, or alternatively as expressions written in parentheses. Multiple fields can be specified if the index method supports multicolumn indexes.

How can I create a table index?

Table indexes are created within the [Index Properties](#)^[84] dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Indexes](#) tab there;
- press the **Insert** key or select the [Add New Index...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the table in the explorer tree and use the [Create New Index](#) popup menu item

or

- select the table [Indexes](#) node or any index within the table in the explorer tree and use the [Add New Index...](#) popup menu item.

■ How can I edit an existing index?

Table indexes are edited within the [Index Properties](#)⁸⁴ dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Indexes](#) tab there;
- press the **Enter** key or select the [Edit Index](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the index to edit in the explorer tree and use the [Edit Index](#) popup menu item.

You can change the name of the index using the [Rename Index](#) dialog. To open the dialog you should either

- select the index to rename in the explorer tree;
- select the [Rename Index](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Indexes](#) tab there;
- select the index to rename;
- select the [Rename Index](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a table index?

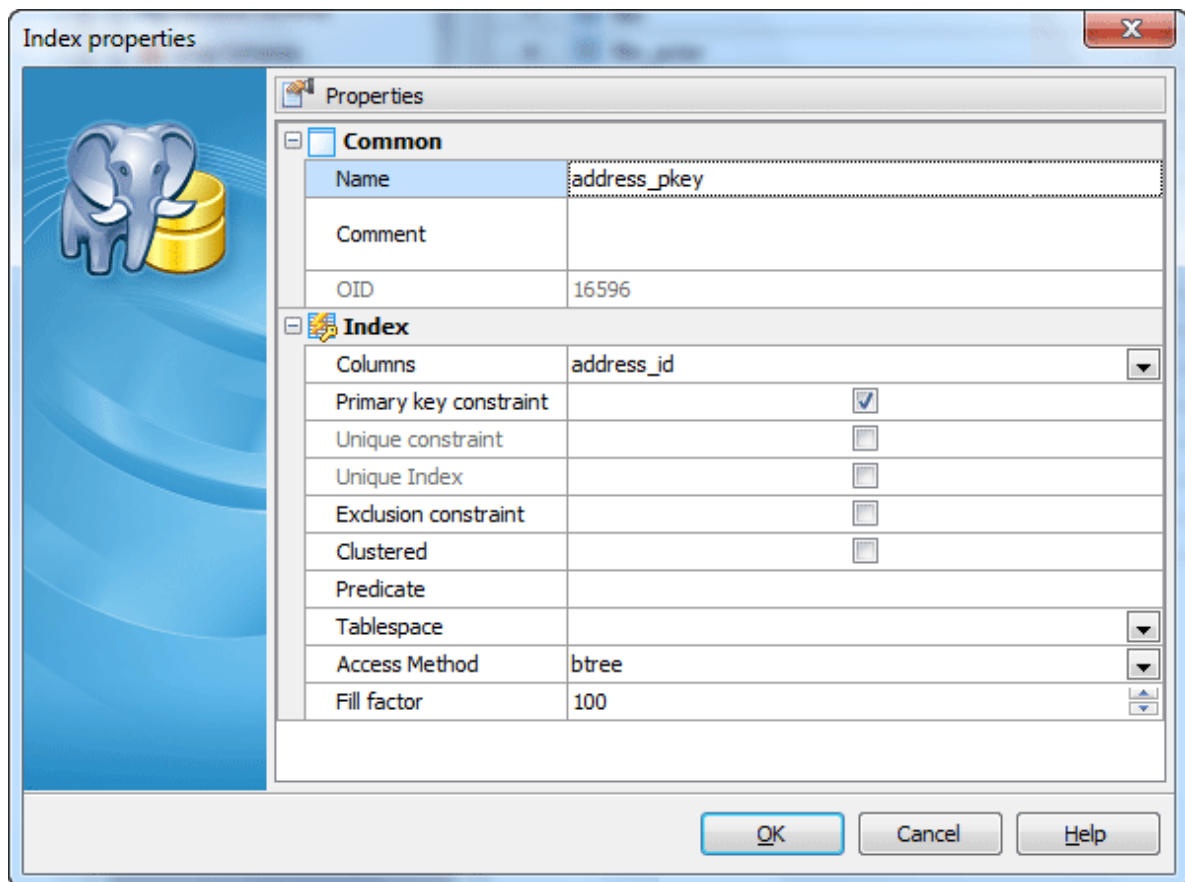
To drop the table index:

- select the index to drop in the explorer tree;
- select the [Drop Index](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Indexes](#) tab there;
- press the **Delete** key or select the [Drop Index](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.



Use the [Columns](#) drop-down list to select a key field(s) for the index. Specify the index sort order and Nulls order ([First](#) means the index sorts nulls before non-nulls, [Last](#) - conversely).

Expression

An index field can be an expression computed from the values of one or more columns of the table row. This feature can be used to obtain fast access to data based on some transformation of the basic data. For example, an index computed on `upper(col)` would allow the clause `WHERE upper(col) = 'JIM'` to use an index. To set the index expression, use the Add expression button and enter the expression in the corresponding window.

Example

Suppose, there is a table *Employee* with the *salary* and *premium* fields and queries as follows are executed very often.

```
SELECT *
FROM "Employee"
WHERE "salary"+"premium">5000;
```

To create an index making such queries faster, set *salary+premium* as the index expression.

☒ **Primary key constraint**

With this option checked this field becomes a compound primary key. It is useful in case the table has more than one primary key.

☒ **Unique constraint**

Check the option to permit no duplicate values. A unique column must also define the NOT NULL attribute. A table can have one or more unique keys.

☒ **Unique Index**

If checked, creates a unique index for the table, i.e. the database system ensures that no two rows of the specified table have the same values in the indexed columns. In this way, if two rows both contain the NULL value for all columns of an index, the two index values are not considered to be identical. If at least one column does not contain the NULL value, two rows that have the same value in all non-NULL columns are considered to be identical.

☒ **Clustered**

Fill the box to cluster the table based on the current index.

Note: When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. If one wishes, one can periodically recluster by issuing the command again.

Predicate

Set the constraint expression to create a partial index. A partial index is an index that contains entries for only a portion of a table, usually a portion that is more useful for indexing than the rest of the table. For example, if you have a table that contains both billed and unbilled orders where the unbilled orders take up a small fraction of the total table and yet that is an often used section, you can improve performance by creating an index on just that portion.

Tablespaces define locations in the file system where the files representing table objects can be stored.

Access method

PostgreSQL provides several index types: *B-tree*, *R-tree*, *Hash*, and *GiST*. Each index type uses a different algorithm that best suits different types of queries. By default, the CREATE INDEX command will create a B-tree index which fits the most common situations.

- *B-trees* can handle equality and range queries on data that can be sorted into some ordering.
- *R-tree* indexes are suited for queries on spatial data.
- *Hash* indexes can only handle simple equality comparisons. The query planner will consider using a hash index whenever an indexed column is involved in a comparison using the '=' operator.
- *GiST* indexes are not a single kind of index, but rather an infrastructure within which


many different indexing strategies can be implemented. Accordingly, the particular operators with which a GiST index can be used vary depending on the indexing strategy (the operator class).

5.2.5 Foreign Keys

A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table. In other words, a foreign key is a column or a combination of columns that is used to establish and enforce a link between the data in two tables.

Note: To create a foreign key constraint, it is necessary to have this privilege for both the referencing and referenced tables.

■ How can I add a new foreign key?

Foreign keys are created within the [Foreign Key Properties](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Foreign Keys](#) tab there;
- press the **Insert** key or select the [Add New Foreign Key...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))


or

- select the table in the explorer tree and use the [Create New Foreign Key](#) popup menu item

or

- select the table [Foreign Keys](#) node or any foreign key within the table in the explorer tree and use the [Add New Foreign Key...](#) popup menu item.

■ How can I edit an existing foreign key?

Foreign Keys are edited within the [Foreign Key Properties](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Foreign Keys](#) tab there;
- press the **Enter** key or select the [Edit Foreign Key](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the foreign key to edit in the explorer tree and use the [Edit Foreign Key](#) popup menu item.

You can change the name of the foreign key using the [Rename Foreign Key](#) dialog. To open the dialog you should either

- select the foreign key to rename in the explorer tree;
- select the [Rename Foreign Key](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Foreign Keys](#) tab there;
- select the foreign key to rename;
- select the [Rename Foreign Key](#) item from the popup menu

(alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a foreign key?

To drop the foreign key:

- select the foreign key to drop in the explorer tree;
- select the [Drop Foreign Key](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Foreign Keys](#) tab there;
- press the **Delete** key or select the [Drop Foreign Key](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

Set the Foreign Key [Name](#), select [Columns](#) from the [Available Fields](#) list to include into the foreign key, select the Foreign Table [Name](#) from the drop-down list and its fields from the list to include, set other foreign key properties and apply the changes by clicking the [OK](#) button.

Foreign key properties

Properties

☐ Common

Name	address_city_id_fkey
Comment	
OID	16674

☒ Foreign key

Columns	city_id
Foreign table	public.city
Foreign columns	city_id
On update rule	CASCADE
On delete rule	RESTRICT
Match type	Simple
Deferrable	<input type="checkbox"/>
Check time	After each statement

OK Cancel Help

All the fields which are included into the Foreign Key must be included into indexes as well. See [Indexes](#) ⁸⁴ for details.

Set rules **ON DELETE** and **ON UPDATE** from the respective drop-down lists.

- **NO ACTION** Produce an error indicating that the deletion or update will create a foreign key constraint violation. If the constraint is deferred this error will be produced at constraint check time if there still exist any referencing rows. This is the default action.
- **RESTRICT** Produce an error indicating that the deletion or update would create a foreign key constraint violation. This is the same as NO ACTION except that the check is not deferrable.
- **CASCADE** Delete any rows referencing the deleted row, or update the value of the referencing column to the new value of the referenced column, respectively.
- **SET NULL** Set the referencing column(s) to null.
- **SET DEFAULT** Set the referencing column(s) to their default values.

Match type

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.

- **Match Full** will not allow one column of a multi-column foreign key to be null unless all foreign key columns are null.
- **Match Simple**, which is also the default, allows some foreign key columns to be null while other parts of the foreign key are not null.

Deferrable

This controls whether the constraint can be deferred. A constraint that is not deferrable will be checked immediately after every command. Checking of foreign key constraints that are deferrable may be postponed until the end of the transaction.

Comment

The box allows you to set optional text describing the foreign key.

☒ Not validated

Turn this option ON to skip a scan of the table to verify that all existing rows in the table satisfy the new constraint. In this case the constraint will be applied against new inserts or updates. The database will not assume that the constraint holds for all rows in the table, until it will be validated manually via popup menu of the Foreign Keys tab in the Table Editor. Icons for not validated constraints are marked with a red exclamation mark in GUI.



☒ Not enforced

Turn this option ON to prevent the database system from checking the constraint. It is then up to the application code to ensure that the constraints are satisfied. The database system might still assume that the data actually satisfies the constraint for optimization decisions where this does not affect the correctness of the result. Not enforced constraints can be useful as documentation if the actual checking of the

constraint at run time is too expensive. Leave this option OFF to oblige the database system to ensure that the constraint is satisfied, by checking the constraint at appropriate times (after each statement or at the end of the transaction, as appropriate). Not enforced constraints are displayed with a grayscale icon in the GUI.

5.2.6 Checks

A [check](#) constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

The [Check Properties](#) editor allows you to add a new check constraint or edit an existing one. This dialog can be invoked from [Table Editor](#) , or via the popup menu of the corresponding nodes of the [explorer tree](#) .

■ How can I add a new check?

Checks are created within [Check Properties](#). In order to run the wizard you should either

- open the table in [Table Editor](#) and the [Checks](#) tab there;
- press the **Insert** key or select the [Add New Check...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))


or

- select the table in the explorer tree and use the [Create New Check...](#) popup menu item

or

- select the table [Checks](#) node or any check within the table in the explorer tree and use the [Add New Check...](#) popup menu item.

■ How can I edit an existing check?

Checks are edited within the [Check Properties](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) the [Checks](#) tab there;
- press the **Enter** key or select the [Edit Check](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the check to edit in the explorer tree and use the [Edit Check](#) popup menu item.

You can change the name of the check using the [Rename Check](#) dialog. To open the dialog you should either

- select the check to rename in the explorer tree;
- select the [Rename Check](#) item from the popup menu

or

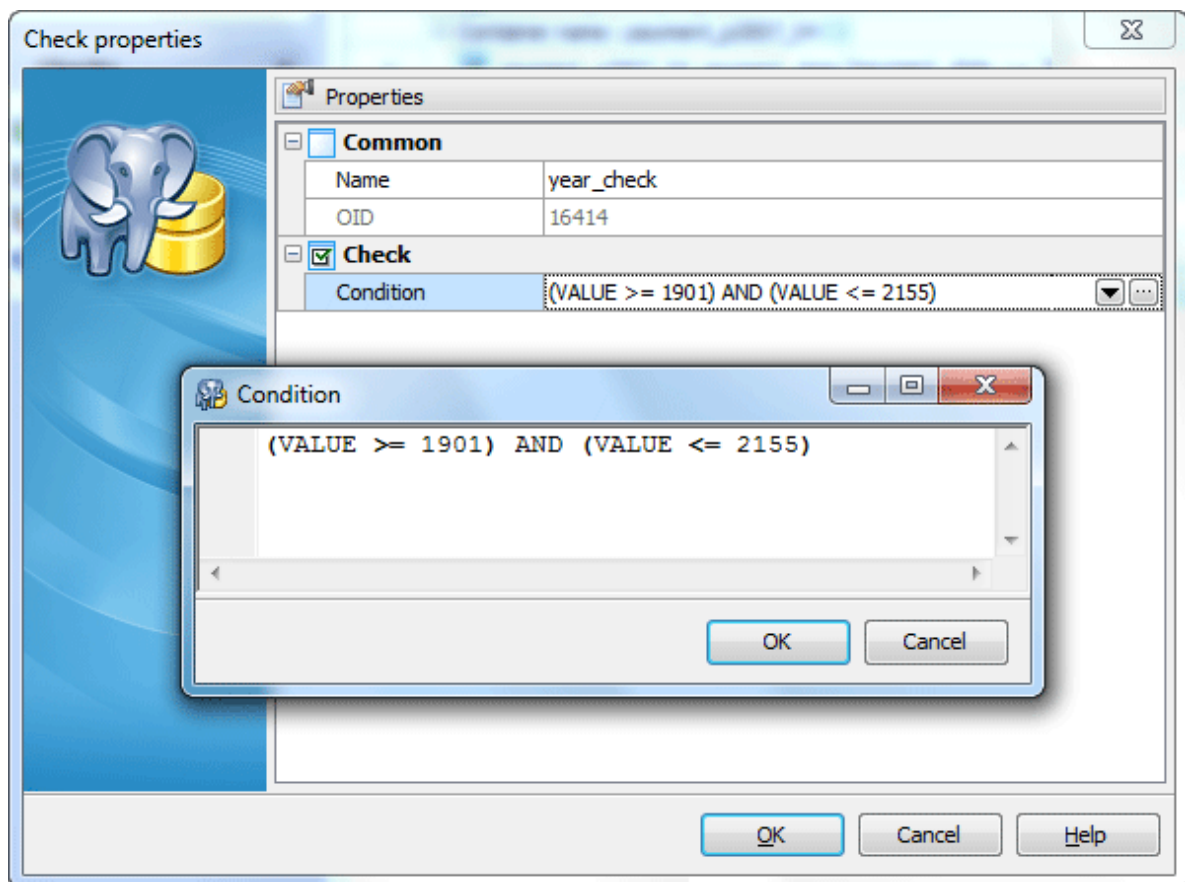
- open the table in [Table Editor](#) and the [Checks](#) tab there;
- select the check to rename;
- select the [Rename Check](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a check?

To drop the check:

- select the check to drop in the explorer tree;
 - select the [Drop Check](#) item from the popup menu
- or
- open the table in [Table Editor](#) and the [Checks](#) tab there;
 - press the **Delete** key or select the [Drop Check](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.



Name

The name of the check constraint.

Comment

This field contains an optional text describing the check.

OID

This field stores the check OID (object identifier). This is a serial number that is automatically added by PostgreSQL to all checks.

Condition

Specify an expression producing a Boolean result which new or updated rows must satisfy for an insert or update operation to succeed. Expressions evaluating to **True** or **Unknown** succeed. In case any row of an insert or update operation produce a **FALSE** result an error exception is raised and the insert or update does not alter the database.

Not validated

Turn this option ON to skip a scan of the table to verify that all existing rows in the table satisfy the new constraint. In this case the constraint will be applied against new inserts or updates. The database will not assume that the constraint holds for all rows in the table, until it will be validated manually via popup menu of the Checks tab in the Table Editor. Icons for not validated constraints are marked with a red exclamation mark in GUI.

Not enforced

Turn this option ON to prevent the database system from checking the constraint. It is then up to the application code to ensure that the constraints are satisfied. The database system might still assume that the data actually satisfies the constraint for optimization decisions where this does not affect the correctness of the result. Not enforced constraints can be useful as documentation if the actual checking of the constraint at run time is too expensive. Leave this option OFF to oblige the database system to ensure that the constraint is satisfied, by checking the constraint at appropriate times (after each statement or at the end of the transaction, as appropriate). Not enforced check constraints are displayed with a grayscale icon in the GUI.

5.2.7 NOT NULL constraints

A NOT NULL constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

The NOT NULL Constraint editor allows you to add a new check constraint or edit an existing one. This dialog can be invoked from [Table Editor](#)^[77], or via the popup menu of the corresponding nodes of the [explorer tree](#)^[58].

How can I add a new NOT NULL constraint?

NOT NULL constraints are created within [NOT NULL Constraint Properties](#) dialog window. In order to open the dialog you should:

- open the table in [Table Editor](#) and the [NOT NULL Constraints](#) tab there;

- press the **Insert** key or select the [Add New NOT NULL Constraint...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ **How can I edit an existing NOT NULL constraint?**

NOT NULL constraints are edited within the [NOT NULL constraints Properties](#) dialog window. In order to open the dialog you should:

- open the table in [Table Editor](#) and the [NOT NULL Constraints](#) tab there;
- press the **Enter** key or select the [Edit NOT NULL Constraints](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the NOT NULL constraint using the [Rename NOT NULL Constraint](#) dialog. To open the dialog you should:

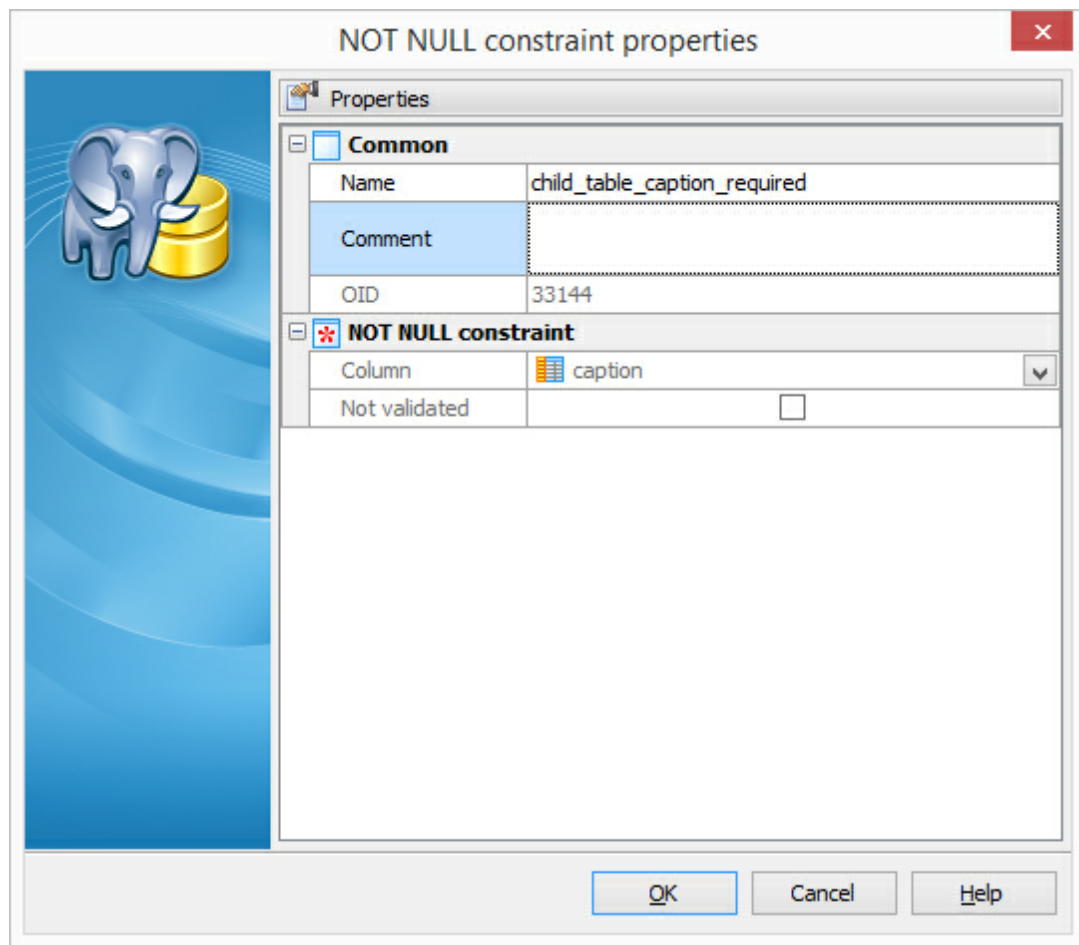
- open the table in [Table Editor](#) and the [NOT NULL Constraints](#) tab there;
- select the NOT NULL constraint to rename;
- select the [Rename NOT NULL Constraint](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ **How can I drop a NOT NULL constraint?**

To drop the NOT NULL constraint:

- open the table in [Table Editor](#) and the [NOT NULL Constraints](#) tab there;
- press the **Delete** key or select the [Drop NOT NULL Constraint](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.



Name

The name of the NOT NULL constraint.

Comment

The box allows you to set optional text describing the NOT NULL constraint.

Column

Select the table column this constraint to be applied.

☒ Not validated

Turn this option ON to skip a scan of the table to verify that all existing rows in the table satisfy the constraint. In this case the constraint will be applied against new inserts or updates. The database will not assume that the constraint holds for all rows in the table, until it will be validated manually via popup menu of the NOT NULL constraints tab in the Table Editor. Icons for not validated constraints are marked with a red exclamation mark in GUI.

5.2.8 Triggers

A **trigger** is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. A trigger can be defined to execute before or after an INSERT, UPDATE, or DELETE operation, either once per

modified row, or once per SQL statement. If a trigger event occurs, the trigger fires. You can also specify a trigger to fire only when a specified column(s) are affected by the query and to set a condition to control over whether a trigger is fired (PostgreSQL 9.0).

How can I add a new trigger?

Triggers are created within [Create Trigger Wizard](#)^[97]. In order to run the wizard you should either

- open the table in [Table Editor](#) and the [Triggers](#) tab there;
- press the **Insert** key or select the [Add New Trigger...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the table in the explorer tree and use the [Create New Trigger...](#) popup menu item

or

- select the table [Triggers](#) node or any trigger within the table in the explorer tree and use the [Add New Trigger...](#) popup menu item.

How can I edit an existing trigger?

Triggers are edited within the [Trigger Editor](#)^[98] dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Triggers](#) tab there;
- press the **Enter** key or select the [Edit Trigger](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the trigger to edit in the explorer tree and use the [Edit Trigger](#) popup menu item.

You can change the name of the trigger using the [Rename Trigger](#) dialog. To open the dialog you should either

- select the trigger to rename in the explorer tree;
- select the [Rename Trigger](#) item from the popup menu

or

- open the table in [Table Editor](#) and the [Triggers](#) tab there;
- select the trigger to rename;
- select the [Rename Trigger](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

How can I drop a trigger?

To drop the trigger:

- select the trigger to drop in the explorer tree;

- select the [Drop Trigger](#) item from the popup menu
- or
- open the table in [Table Editor](#) and the [Triggers](#) tab there;
 - press the **Delete** key or select the [Drop Trigger](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.2.8.1 Create Trigger Wizard

[Create Trigger Wizard](#) guides you through the process of creating of a new table trigger.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)⁴³. Below you will find a description of wizard steps that are unique for the current object.

To define a new trigger, you need to set its [Name](#), [Type](#) (*Before*, *After*). This option determines whether the function is called before or after the event.

[Comment](#)

This field contains a comment to the table trigger.

[Event](#)

One of *Insert*, *Update*, or *Delete*; this specifies the event that will fire the trigger.

[For Each](#) (*Row*, *Statement*)

This specifies whether the trigger procedure should be fired once for every row affected by the trigger event, or just once per SQL statement.

[Function](#)

Select the function to be executed when the trigger fires.

Note: 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.

[Arguments](#)

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be entered here too, but all of them will be converted to strings.

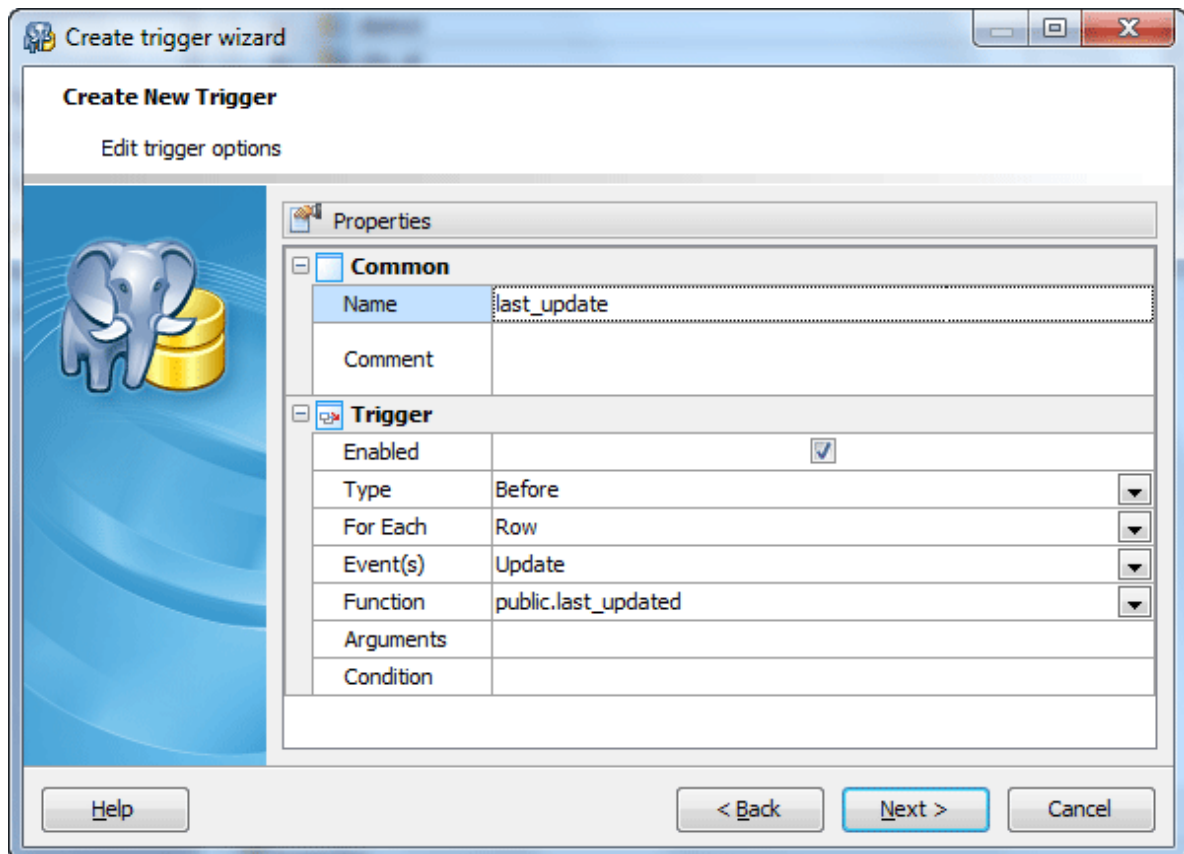
Note: Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

[Condition](#) (PostgreSQL 9.0)

Specify here a Boolean expression that determines whether the trigger function will actually be executed.

[Fields to update](#) (PostgreSQL 9.0)

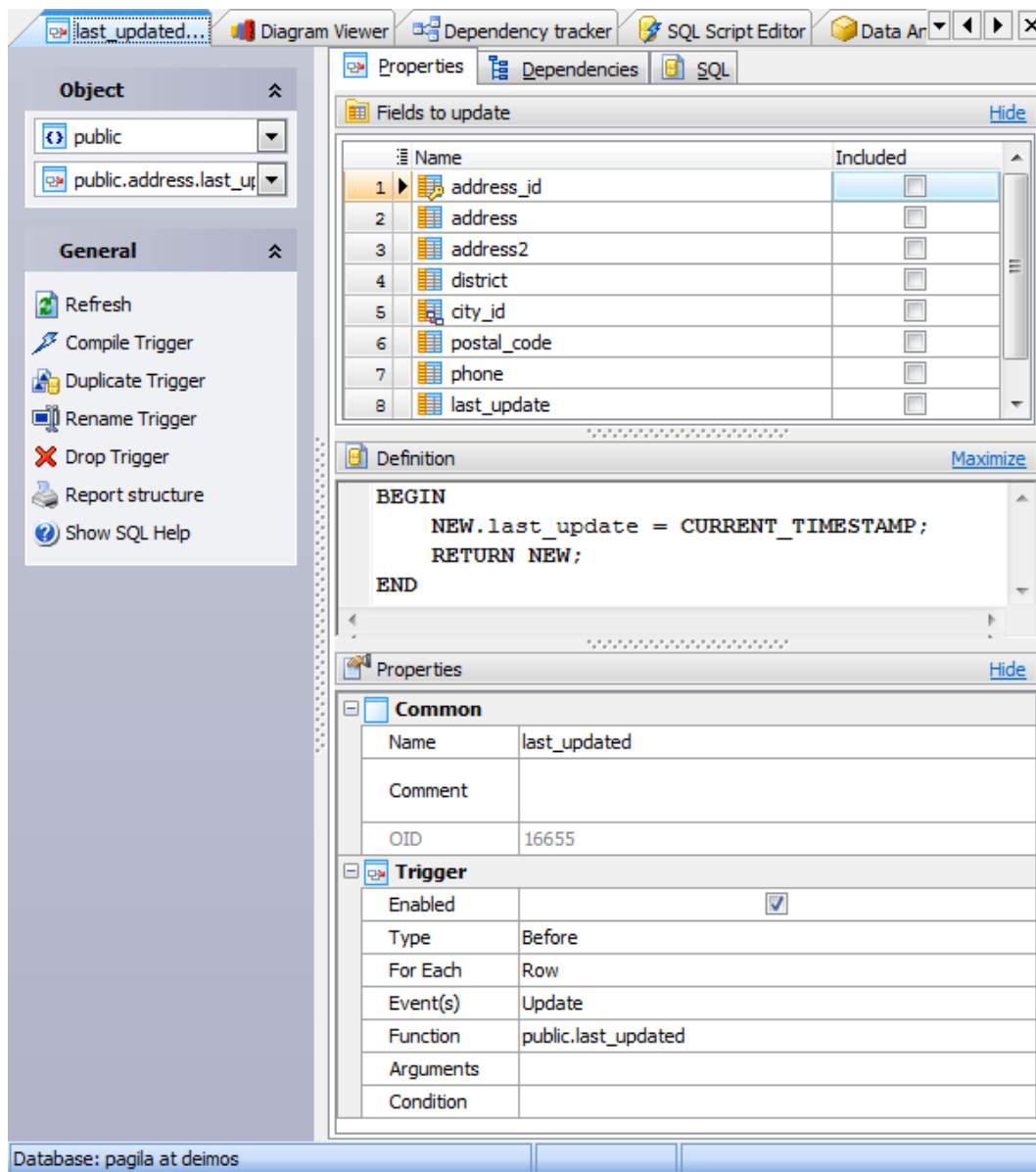
Check table columns to set the trigger to fire only when one of the specified columns is updated. If all columns are unchecked, the trigger is fired when any column of the associated table is updated.



5.2.8.2 Trigger Editor

[Trigger Editor](#) can be opened automatically after the trigger is created and is available on editing the trigger.

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)⁴⁶. Below you will find a description of editor tabs that are unique for the current object.



The main tab of the editor consists of several parts: a list of fields to be updated by the trigger,, trigger definition, and trigger properties.

Fields to update (PostgreSQL 9.0)

The tab contains the list of table columns the trigger to fire only when one of the specified columns is updated. If all columns are unchecked, the trigger is fired when any column of the associated table is updated.

Definition

Defines the trigger conditions and actions.

Properties

Name

Here you can view and change the trigger name.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

Comment

This field contains a comment to the trigger.

OID

This field stores the trigger OID (object identifier). This is a serial number that is automatically added by PostgreSQL to all triggers.

For table

Here is the field where you can specify the name of the table the trigger is defined for.

For Each (Row, Statement)

This specifies whether the trigger procedure should be fired once for every row affected by the trigger event, or just once per SQL statement.

Function

The function is executed when the trigger fires.

The [Arguments](#) list contains the list of the arguments to be provided for the function when the trigger is executed.

Type(Before, After)

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

Condition (PostgreSQL 9.0)

The field contains a Boolean expression that determines whether the trigger function will actually be executed.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.2.9 Rules

The PostgreSQL rule system allows 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.

■ How can I add a new table rule?

Rules are created within [Create Rule Wizard](#)^[101]. In order to run the wizard you should either

- open the table in [Table Editor](#) and the [Rules](#) tab there;
- press the [Insert](#) key or select the [Add New Rule...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the table in the explorer tree and use the [Create New Rule...](#) popup menu item

or

- select the table [Rules](#) node or any rule within the table in the explorer tree and use the [Add New Rule...](#) popup menu item.

■ **How can I edit an existing table rule?**

Rules are edited within the [Rule Editor](#)^[103] dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#) and the [Rules](#) tab there;
- press the [Enter](#) key or select the [Edit Rule](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

or

- select the rule to edit in the explorer tree and use the [Edit Rule](#) popup menu item.

■ **How can I drop a table rule?**

To drop a rule:

- select the rule to drop in the explorer tree;
- select the [Drop Rule](#) item from the popup menu

or

- open the table in [Table Editor](#)^[77] and the [Rules](#) tab there;
- select the rule to drop;
- press the **Delete** key or select the [Drop Rule](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.2.9.1 Create Rule Wizard

[Create Rule Wizard](#) guides you through the process of creating a new table rule.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Specifying rule properties

[Comment](#)

The box allows you to set optional text to describe the new rule.

[Event](#)

Select the event that will fire the rule. The event is one of *Select*, *Insert*, *Update*, or *Delete*.

☒ [Instead](#)

The checkbox indicates that the specified commands will be executed instead of the original command.

Condition

Here you can set 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.

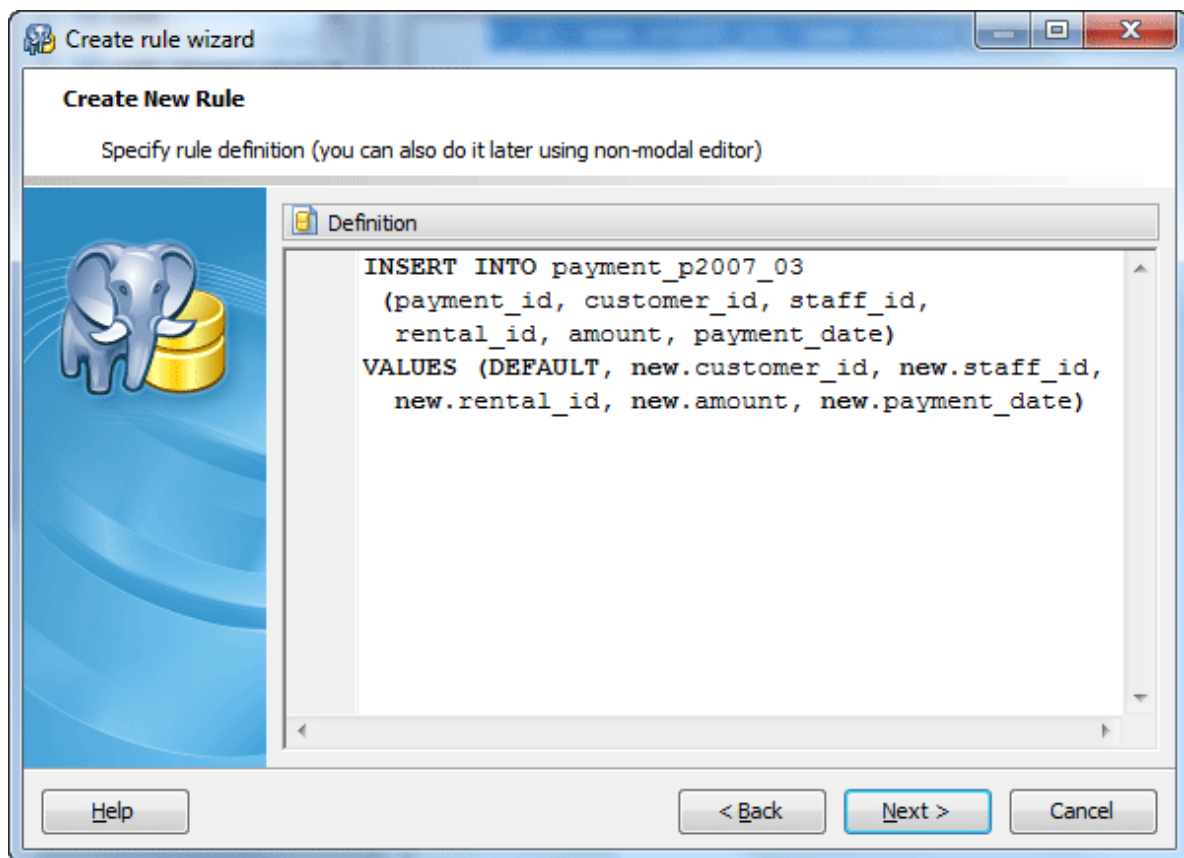
The screenshot shows the 'Create rule wizard' dialog box, specifically the 'Create New Rule' step. The dialog has a title bar with a wizard icon and the text 'Create rule wizard'. Below the title bar, there's a section titled 'Create New Rule' with a sub-option 'Edit rule options'. On the left side, there's a blue background with a white elephant icon and a yellow database cylinder icon. The main area is divided into two sections: 'Properties' and 'Rule'. The 'Properties' section has a 'Common' tab with fields for 'Name' (containing 'payment_insert_p2007_03') and 'Comment'. The 'Rule' section has a 'Rule' tab with fields for 'Event' (set to 'Insert'), 'Instead' (with a checked checkbox), and 'Condition' (containing the SQL expression: '((new.payment_date >= '2007-03-01 00:00:00'::timestamp without time zone) AND (new.payment_date < '2007-04-01')). At the bottom, there are buttons for 'Help', '< Back', 'Next >', and 'Cancel'.

Properties	
Common	
Name	payment_insert_p2007_03
Comment	

Rule	
Event	Insert
Instead	<input checked="" type="checkbox"/>
Condition	((new.payment_date >= '2007-03-01 00:00:00'::timestamp without time zone) AND (new.payment_date < '2007-04-01

Entering rule definition

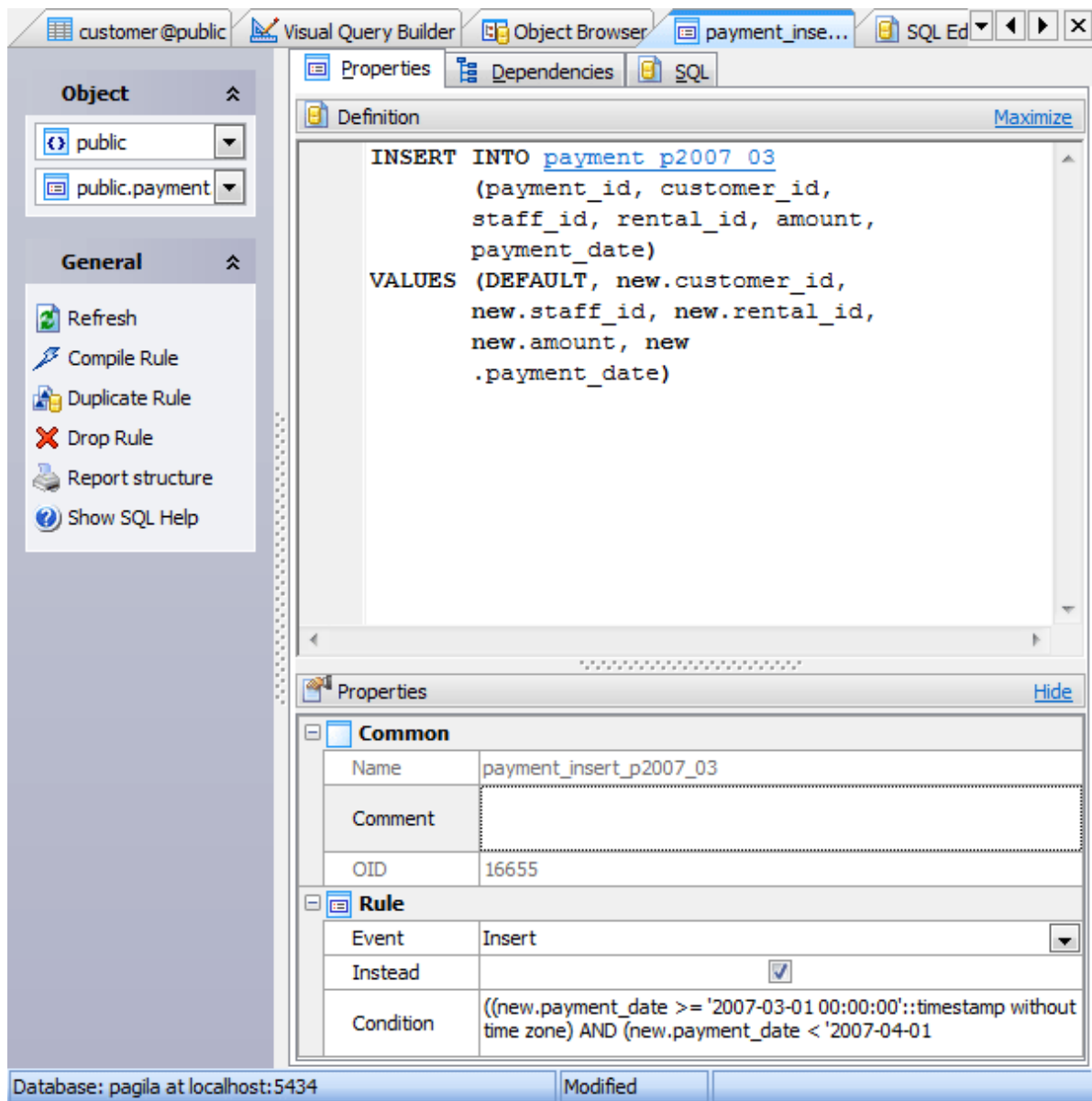
At this step you can specify the SQL definition for the new rule. For any event (except the [SELECT](#)) the step is optional: you can do it later using a non-modal editor.



5.2.9.2 Rule Editor

[Rule Editor](#) can be open automatically after the rule is created and is available on editing the rule. You can open a rule in the editor from the [Explorer Tree](#) or via the [Rules](#) tab of [Object Manager](#). Alternatively, you may use the corresponding link of the [Navigation Bar](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)⁴⁶. Below you will find a description of editor tabs that are unique for the current object.



The main tab of [Rule Editor](#) consists of two parts: Definition and Properties.

Definition

Properties

Body

The command or commands that make up the rule action. Valid commands are *Select*, *Insert*, *Update*, *Delete* or *Notify*.

Name

Here you can change the rule name.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need

unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

Comment

This field stores a comment to the rule.

OID

The field stores the rule OID (object identifier). The latter can be defined as a serial number that is automatically added by PostgreSQL to all rules.

Event

Here is the event that will fire the rule.

☒ Instead

The checkbox indicates that the specified commands will be executed instead of the original command.

Condition

The field contains the rule conditional expression.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.2.10 Foreign Key References


A foreign key specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. The [Foreign Key References](#) tab allows you to manage foreign keys created in other tables and reference for columns of the current one. Table objects are managed within the corresponding tab of [Table Editor](#) [7]. Unlike *tables* or *views*, Foreign Key References are actually not database objects. These are only references to foreign keys. They are designed specially for easy foreign keys management.

References Maximize				
	Name	Referencing table	Referencing columns	Referenced columns
1	payment_customer_id_fkey	public.payment	customer_id	customer_id
2	payment_p2007_01_customer_id_fkey	public.payment_p2007_01	customer_id	customer_id
3	payment_p2007_02_customer_id_fkey	public.payment_p2007_02	customer_id	customer_id
4	payment_p2007_03_customer_id_fkey	public.payment_p2007_03	customer_id	customer_id
5	payment_p2007_04_customer_id_fkey	public.payment_p2007_04	customer_id	customer_id
6	payment_p2007_05_customer_id_fkey	public.payment_p2007_05	customer_id	customer_id
7	payment_p2007_06_customer_id_fkey	public.payment_p2007_06	customer_id	customer_id
8	rental_customer_id_fkey	public.rental	customer_id	customer_id

Foreign keys
 Checks
 Triggers
 Rules
 References
 Parent tables

See also: [Foreign Keys](#) 

■ How can I add a new foreign key reference?


Table foreign key references are edited within the [Foreign Key Editor](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#);
- open the [Subitems](#) item and the [Foreign Keys References](#) tab there;
- press **Enter** key or select the [Edit Foreign Key](#) item from the popup menu

or

- select the foreign key to edit in the appropriate table group of the explorer tree and use the [Edit Foreign Key](#) popup menu item.

■ How can I edit an existing foreign key reference?

Table foreign key references are edited within the [Foreign Key Editor](#)  dialog window. In order to open the dialog you should either

- open the table in [Table Editor](#);
- open the [Subitems](#) item and the [Foreign Keys References](#) tab there;
- press **Enter** key or select the [Edit Foreign Key](#) item from the popup menu

or

- select the foreign key to edit in the appropriate table group of the explorer tree and use the [Edit Foreign Key](#) popup menu item.

■ How can I drop a foreign key reference?

To drop the foreign key reference:

- open the table in [Table Editor](#);
- open the [Subitems](#) item and the [Foreign Keys References](#) tab there;
- press **Delete** key or select the [Drop Foreign Key](#) item from the popup menu;

or

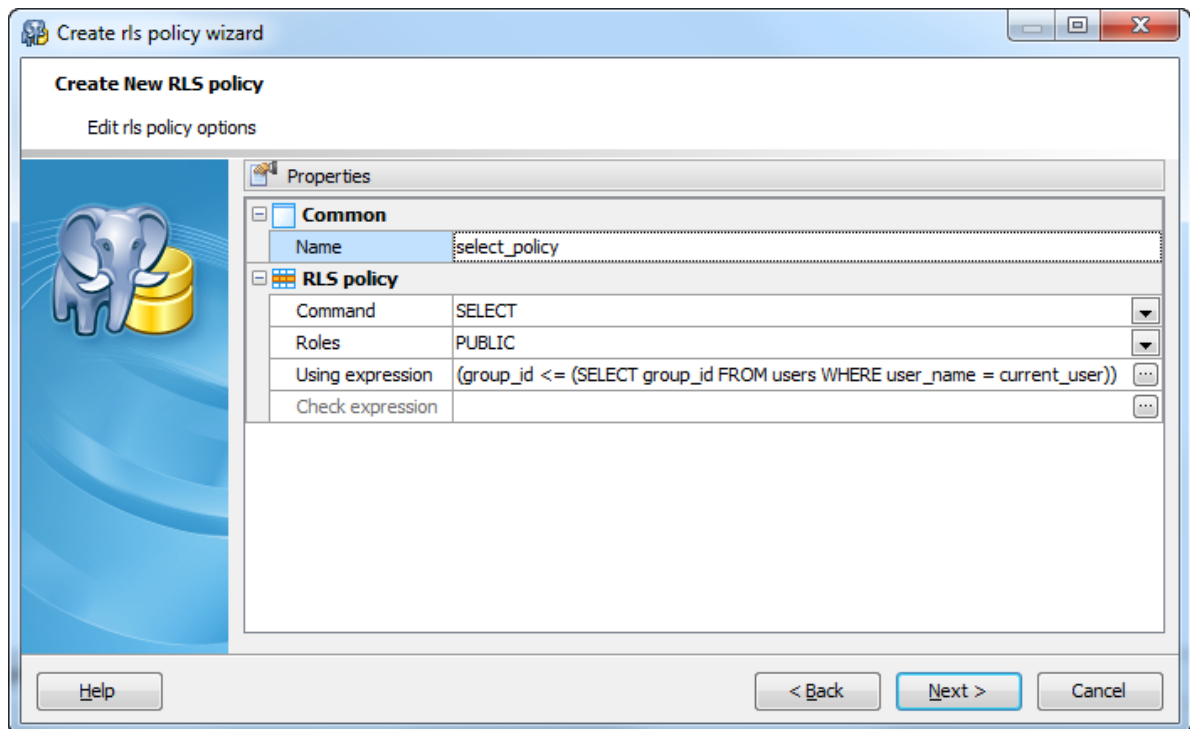
- select the foreign key to drop in the appropriate table group of the explorer tree and use the [Drop Foreign Key](#) popup menu item.

and confirm dropping in the dialog window.

5.2.11 Row Security Policies

Row Security Policies have been implemented in PostgreSQL 9.5. They allow you to restrict, on a per-user basis, which rows can be returned by normal queries or inserted, updated, or deleted by data modification commands. Note that row-level security must

be enabled on the table (using ALTER TABLE ... ENABLE ROW LEVEL SECURITY) in order for created policies to be applied.



Command

The command to which the policy applies. Valid options are ALL, SELECT, INSERT, UPDATE, and DELETE. ALL is the default.

Roles

The role(s) to which the policy is to be applied. The default is PUBLIC, which will apply the policy to all roles.

Using Expression

Any SQL conditional expression (returning boolean). This expression will be added to queries that refer to the table if row level security is enabled. Rows for which the expression returns true will be visible. Any rows for which the expression returns false or null will not be visible to the user (in a SELECT), and will not be available for modification (in an UPDATE or DELETE).

Check Expression

Any SQL conditional expression (returning boolean). This expression will be used in INSERT and UPDATE queries against the table if row level security is enabled. Only rows for which the expression evaluates to true will be allowed.

5.3 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).

■ How can I create a new view?

New views are created within [Create View Wizard](#)^[109]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [View](#) icon in the [Create Database Object](#) dialog
- or
- select the [Views](#) list or any object from that list in the explorer tree;
 - select the [Create New View...](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Views](#) tab there;
 - press the **Insert** key or select the [Create New View](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new view with the same properties as one of the existing views has:

- select the [Object | Duplicate Database Object...](#) main menu item.
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing view definition?

Views can be edited within [View Editor](#)^[114]. In order to run the editor you should either

- select the view for editing in the explorer tree (type the first letters of the view name for quick search);
 - select the [Edit View...](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Views](#) tab there;
 - select the view to edit;
 - press the **Enter** key or select the [Edit View](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the view using the [Rename View](#) dialog. To open the dialog you should either

- select the view to rename in the explorer tree;
 - select the [Rename View](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Views](#) tab there;

- select the view to rename;
- select the [Rename View](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

📌 How can I drop a view?

To drop a view:

- select the view to drop in the explorer tree;
 - select the [Drop View](#) item from the popup menu
- or
- open [Schema \(Database\) Editor](#) and the [Views](#) tab there;
 - select the view to drop;
 - press the **Delete** key or select the [Drop View](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

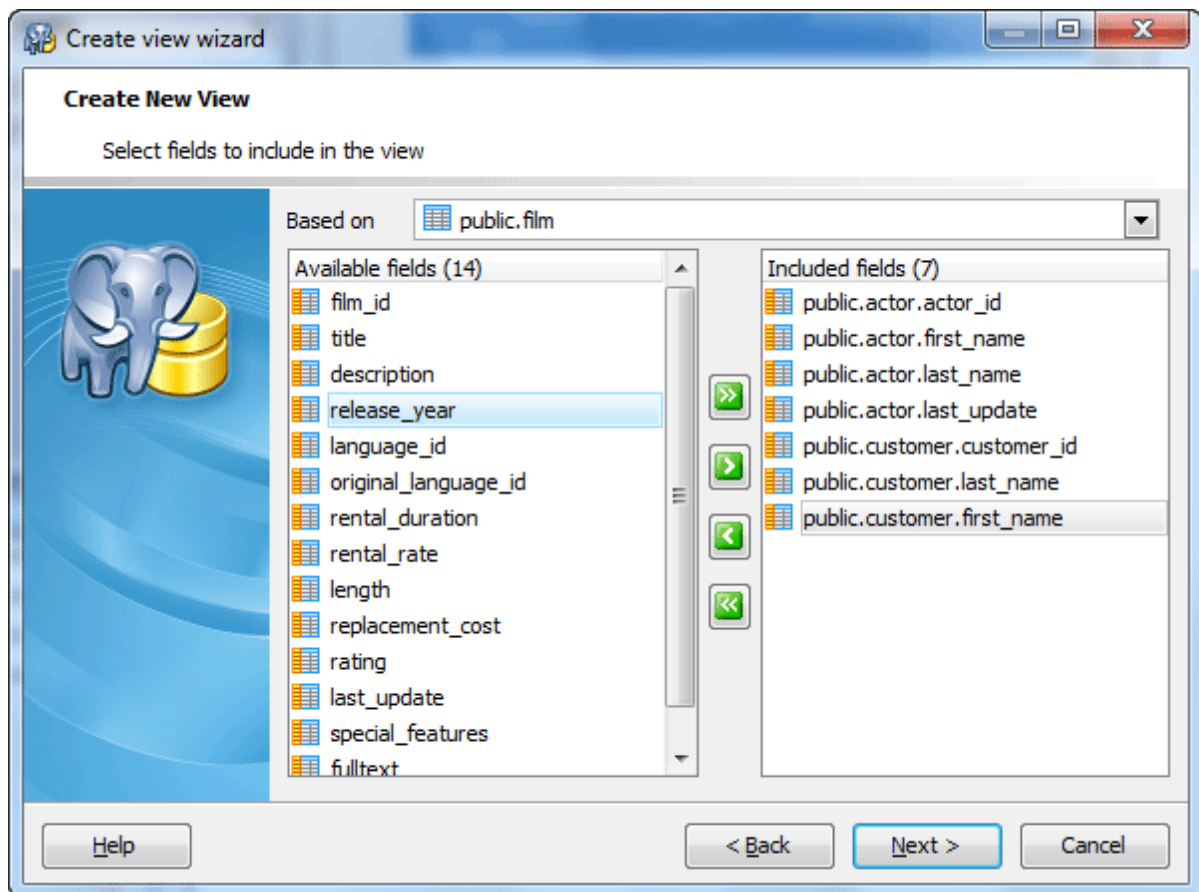
5.3.1 Create View Wizard

[Create View Wizard](#) guides you through the process of creating a new view. See [How To Create View](#)^[108] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Selecting fields for a new view

Select a table or a view from the [Based on](#) drop-down menu. Then specify which fields will be used in the new view. Use [Add All](#) or [Add Selected](#) buttons to include field(s) into view definition. Use the [Remove Selected](#) or [Remove All](#) items to exclude field(s) from the view's field list. Click the [Next](#) button to proceed.



Specifying view options

Name

You may specify here the name of the view being created.

Owner

Defines the owner of the new view. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

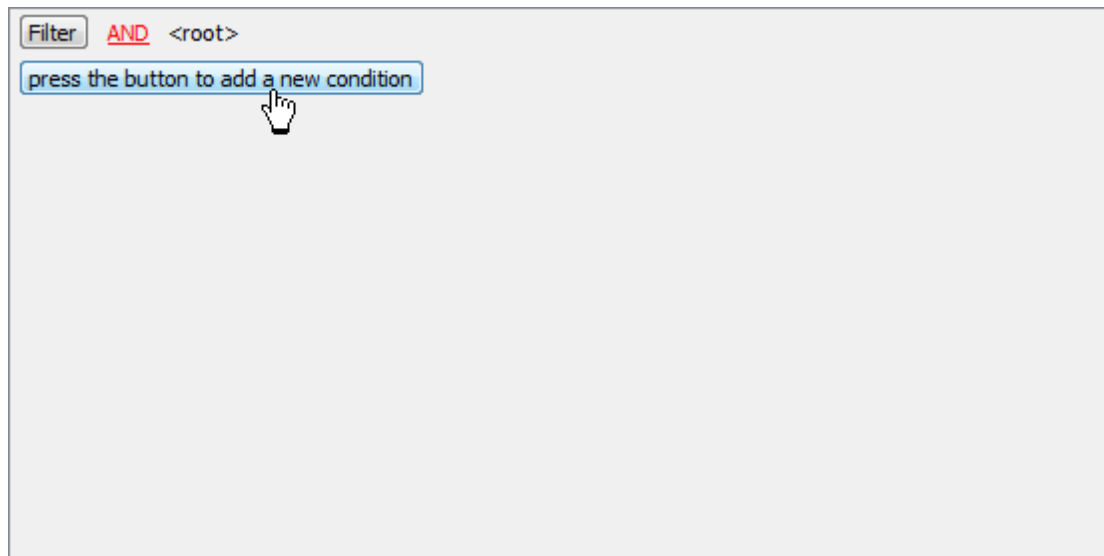
The box allows you to set optional text describing the view.

Specifying the WHERE condition

PostgreSQL Maestro provides the [Filter Builder](#) dialog to facilitate a creating of the WHERE condition.

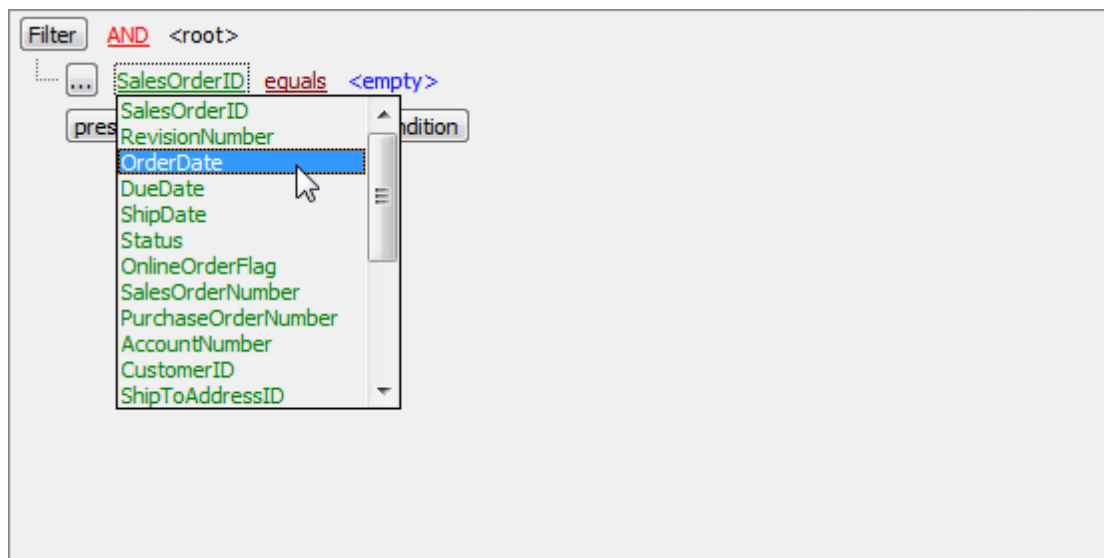
■ Adding a new condition to the filter

Suppose we need to select orders from the sample table *Orders* made between 01.02.2010 and 10.02.2010. These criteria are applied to the *OrderDate* column. Press the button to add this condition. Alternatively, you can use the [Filter](#) button and select the [Add Condition](#) option from the drop-down menu.



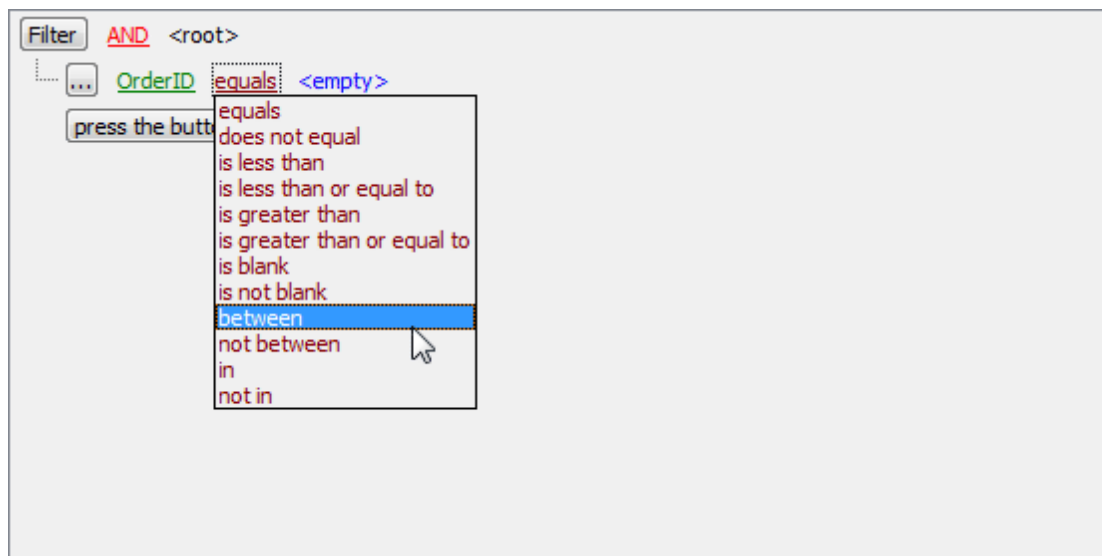
■ Setting a filter criteria in the condition

Select the *OrderDate* column in the drop-down list of the available columns.



■ Setting an operator in the condition

Set the proper operator. In our example it is BETWEEN.



■ Setting criteria values in the condition

Next, you need to specify the range values for the selected operator. The editor used in value boxes is determined by the editor type assigned to the corresponding column.



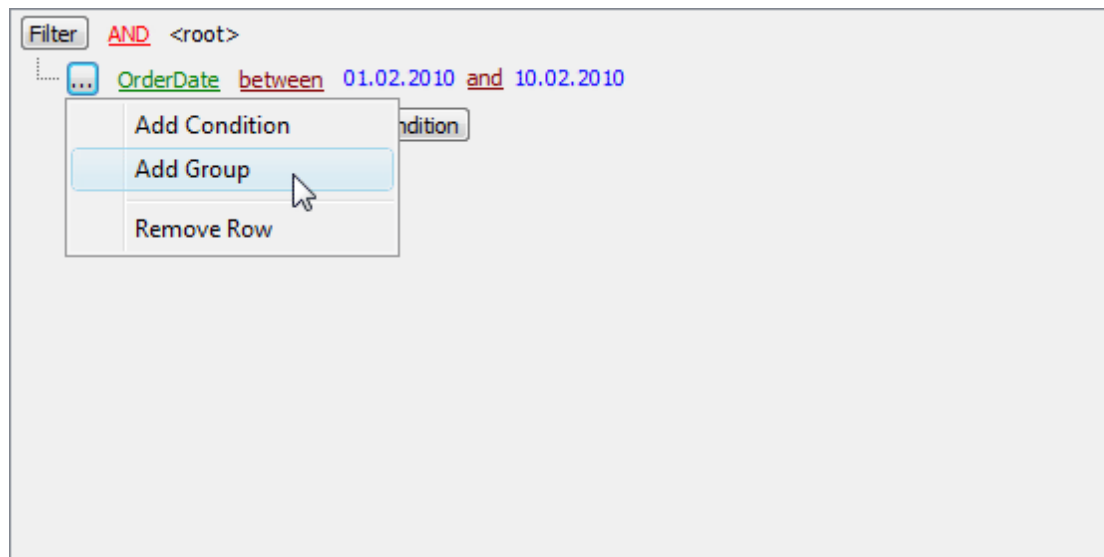
Now use the [Apply](#) button to see the filter result.

You can add additional conditions to the same root level to be combined by the AND operator.

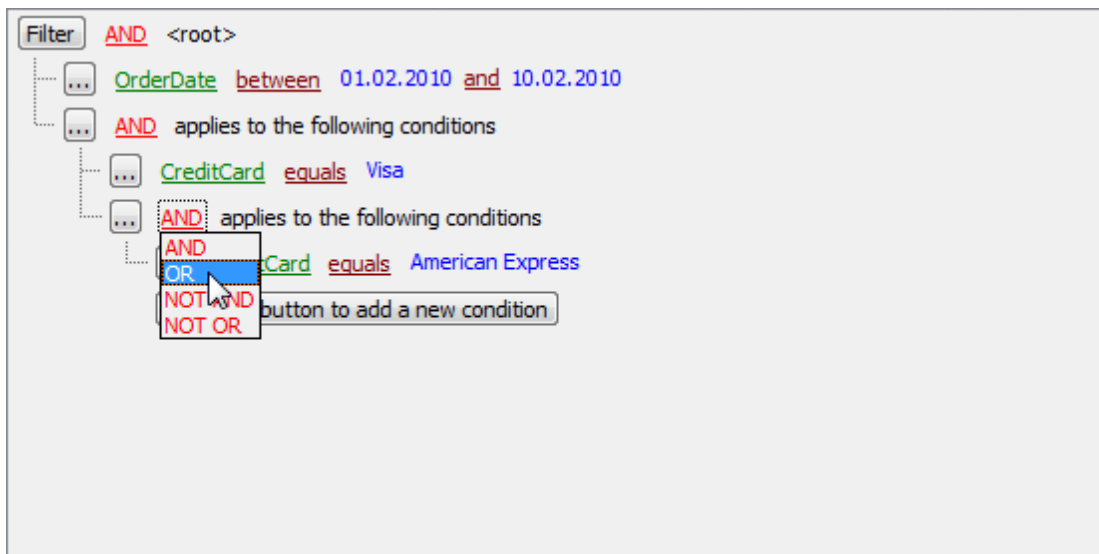
■ Adding a new group

Suppose we need to select orders made between 01.02.2010 and 10.02.2010 and paid via 'Visa' or 'American Express'. This is a complex filter condition combining

two simple conditions with the OR operator. Conditions from the same root level are combined by the AND operator. To add a condition combined with the previous one with the OR (NOT AND, NOT OR) operator, use a new group of conditions.



The next screen represents the finished filter conditions for this example.



Adding view subitems

On this step of the wizard you can specify subobjects of the new view.
To add a new object:

- Choose a necessary page ([Triggers](#) - to manage view triggers, [Rules](#) - to add view rules);
- Press [Insert](#) or use pop-up menu to open the appropriate [Create Object Wizard](#) ([triggers](#)^[97], or [rules](#)^[101]);
- Specify new object properties.

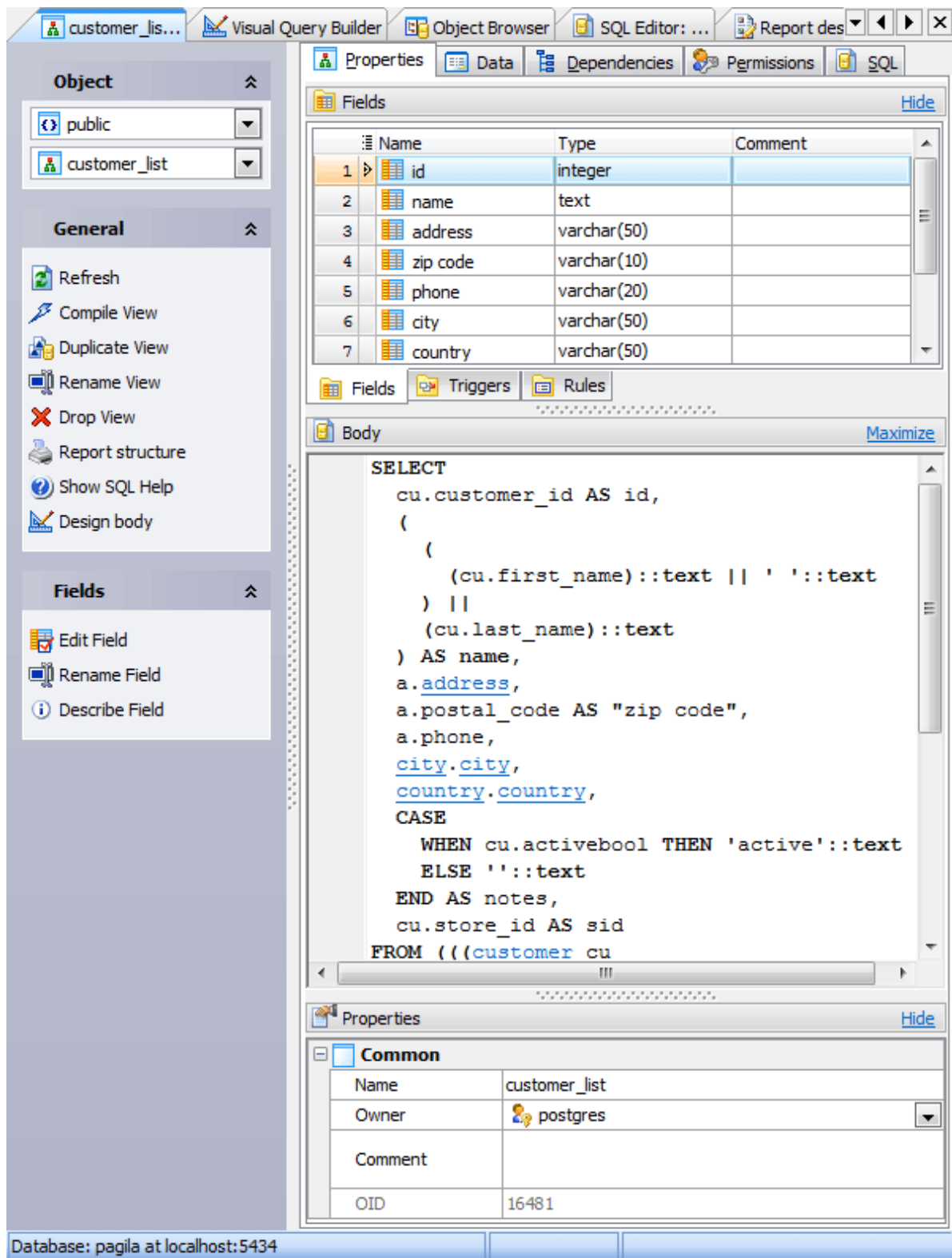
5.3.2 View Editor

[View Editor](#) allows you to edit the existing view definition (view name and the SELECT statement it implements).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

- [Editing view properties](#)^[115]
- [Viewing data](#)^[116]

See also: [Create View Wizard](#)^[109]



5.3.2.1 Editing view properties

View Editor provides you with an ability to edit view properties. The **Properties** tab allows you to change the view name, view definition, the view owner and the comment

for the view. You can also find the body of the view and the view OID there.

Subitems

Every tab is intended for managing some view [subitems](#) (e.g. *fields*, *rules*). Each object can be opened in its editor. Use grid's popup menu to create new, edit or drop the selected view subitems. Using the popup menu you can also copy the selected objects to clipboard or paste previously copied objects.

You can operate on several objects at a time. For this you have to select view objects with the **Shift** or the **Ctrl** key pressed. After a group of objects is selected you can operate on it, e.g. *delete several objects at once*, as if it were a single object.

See also: [Fields](#)^[82], [Rules](#)^[100]

Body

You can edit the view definition in this box.

Use the [Name](#) field to specify the view name.

Note: the name of the object must be unique among all the object names in its container. Moreover, all the objects that are source of data need unique names among themselves. You can use any identifier that is allowed by PostgreSQL server.

Owner

Represents the view owner. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

This field stores a comment to the view.

OID

This field stores the view OID (object identifier). This is a serial number that is automatically added by PostgreSQL to all views.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.3.2.2 Viewing data

The [Data](#) tab displays the data represented in the view as a grid (see Data View for details). The popup menu of this tab and the Data Management navigation bar allow you to export data, get SQL dump, set the value of the selected record to *Null* or to *Now* (for [Date](#) values). In tables with BLOB fields you can also call BLOB Editor to view and edit the BLOB fields.

The screenshot displays a database management application interface. On the left, there is a sidebar with sections for 'Object' (showing 'public' and 'customer_list') and 'General' (containing actions like Refresh, Compile View, Duplicate View, Rename View, Drop View, Report structure, and Show SQL Help). Below this is a 'Data Management' section with options like Generate query, Data input form, Export data, Get SQL dump, Import data, and Print data. The main window is titled 'Visual Query Builder' and contains a 'Data' tab. The data is presented in a table with columns 'id', 'name', and 'address'. The table contains 20 rows of customer data. At the bottom of the table, it says 'Records fetched: 599/599'. The status bar at the very bottom indicates 'Database: pagila at localhost:5434'.

	id	name	address
1	1	MARY SMITH	1913 Hanoi Way
2	2	PATRICIA JOHNSON	1121 Loja Avenue
3	3	LINDA WILLIAMS	692 Joliet Street
4	4	BARBARA JONES	1566 Inegl Manor
5	5	ELIZABETH BROWN	53 Idfu Parkway
6	6	JENNIFER DAVIS	1795 Santiago de Compostela Way
7	7	MARIA MILLER	900 Santiago de Compostela Parkway
8	8	SUSAN WILSON	478 Joliet Way
9	9	MARGARET MOORE	613 Korolev Drive
10	10	DOROTHY TAYLOR	1531 Sal Drive
11	11	LISA ANDERSON	1542 Tarlac Parkway
12	12	NANCY THOMAS	808 Bhopal Manor
13	13	KAREN JACKSON	270 Amroha Parkway
14	14	BETTY WHITE	770 Bydgoszcz Avenue
15	15	HELEN HARRIS	419 Iligan Lane
16	16	SANDRA MARTIN	360 Toulouse Parkway
17	17	DONNA THOMPSON	270 Toulon Boulevard
18	18	CAROL GARCIA	320 Brest Avenue
19	19	RUTH MARTINEZ	1417 Lancaster Avenue
20	20	SHARON ROBINSON	1688 Okara Way

Records fetched: 599/599

Database: pagila at localhost:5434

5.4 Materialized Views

A materialized view is a view that has been computed and stored on disk. Materialized views have characteristics of both views (they are defined using a query specification), and of tables (they allow most table operations to be performed on them). The main differences between a materialized view and a table are that the materialized view cannot subsequently be directly updated and that the query used to create the materialized view is stored in exactly the same way that a view's query is stored, so that fresh data can be generated for the materialized view with the special command.

■ How can I create a new materialized view?

New materialized views are created within [Create Materialized view Wizard](#)^[119]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Materialized view](#) icon in the Create Database Object dialog
- or
- select the [Materialized view](#) list or any object from that list in the explorer tree;
 - select the [Create New Materialized view...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Materialized view](#) tab there;
 - press the **Insert** key or select the [Create New Materialized view](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new materialized view with the same properties as one of the existing defaults has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing materialized view definition?

Materialized views can be edited within Materialized view Editor. In order to run the editor you should either

- select the materialized view for editing in the explorer tree (type the first letters of the materialized view name for quick search);
 - select the [Edit Materialized view ...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Materialized view](#) tab there;
 - select the materialized view to edit;
 - press the **Enter** key or select the [Edit Materialized view](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the materialized view using the [Rename Materialized view](#) dialog. To open the dialog you should either

- select the materialized view to rename in the explorer tree;
 - select the [Rename Materialized view](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Materialized view](#) tab there;
 - select the materialized view to rename;
 - select the [Rename Materialized view](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a materialized view?

To drop a materialized view:

- select the materialized view to drop in the explorer tree;
 - select the [Drop Materialized view](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Materialized view](#) tab there;
 - select the materialized view to drop;
 - press the **Delete** key or select the [Drop Materialized view](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.4.1 Create Materialized Views Wizard

[Create Materialized View Wizard](#) guides you through the process of creating a new view. See [How To Create Materialized View](#) ^[118] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#) ^[43]. Below you will find a description of wizard steps that are unique for the current object.

Selecting fields

To specify fields to be used in the view, select a table or a view from the [Based on](#) drop-down menu and specify which fields will be used in the new view. Use [Add All](#) or [Add Selected](#) buttons to include field(s) into view definition. Use the [Remove Selected](#) or [Remove All](#) items to exclude field(s) from the view's field list. Click the [Next](#) button to proceed. If column names are not provided, they are taken from the output column names of the query.

Specifying options

Use this step to set [Owner](#), [Tablespace](#), and [Comment](#) for the new materialized view. If not specified, default values are consulted.

Is populated

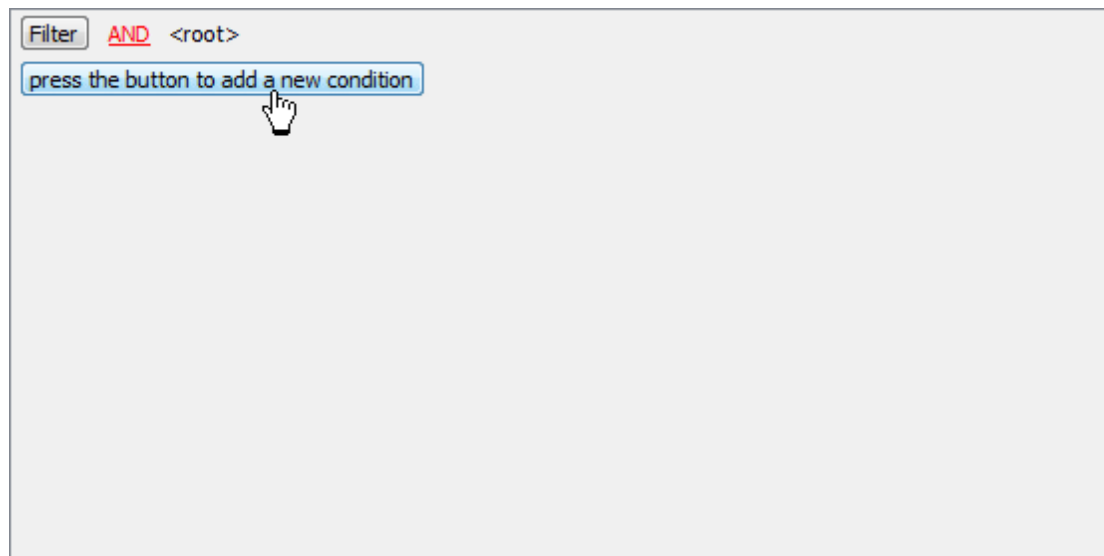
This clause specifies whether or not the materialized view should be populated at

creation time. If not, the materialized view will be flagged as unscannable and cannot be queried until the [Refresh materialized view](#)^[123] is used.

Specifying WHERE condition

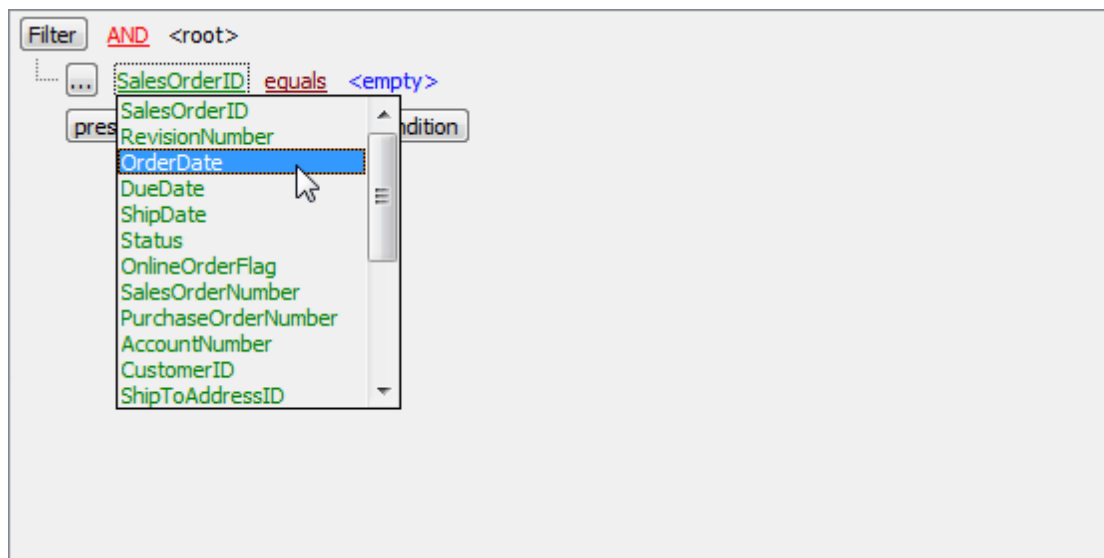
■ Adding a new condition to the filter

Suppose we need to select orders from the sample table *Orders* made between 01.02.2010 and 10.02.2010. These criteria are applied to the *OrderDate* column. Press the button to add this condition. Alternatively, you can use the [Filter](#) button and select the [Add Condition](#) option from the drop-down menu.



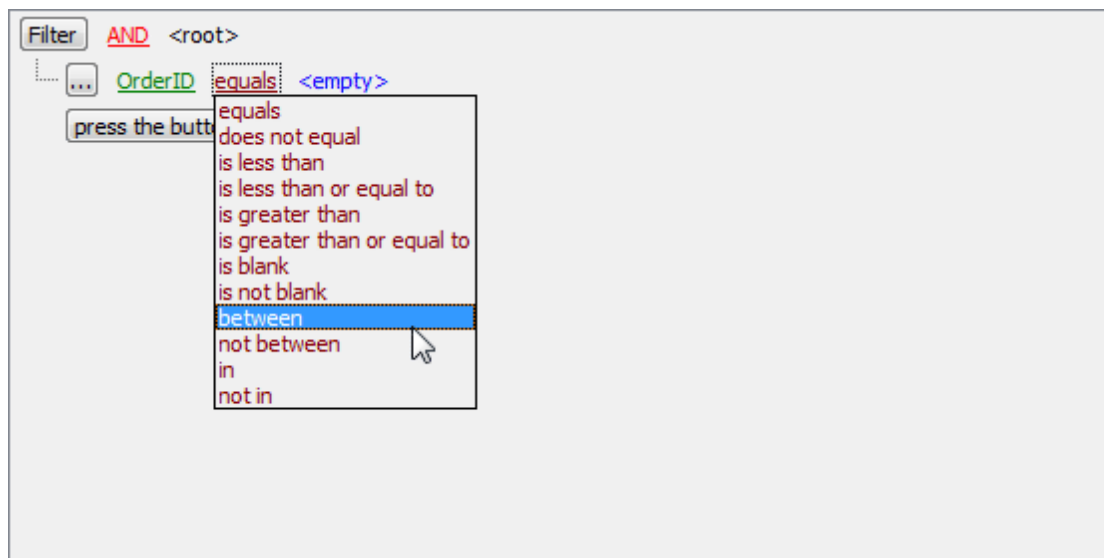
■ Setting a filter criteria in the condition

Select the *OrderDate* column in the drop-down list of the available columns.



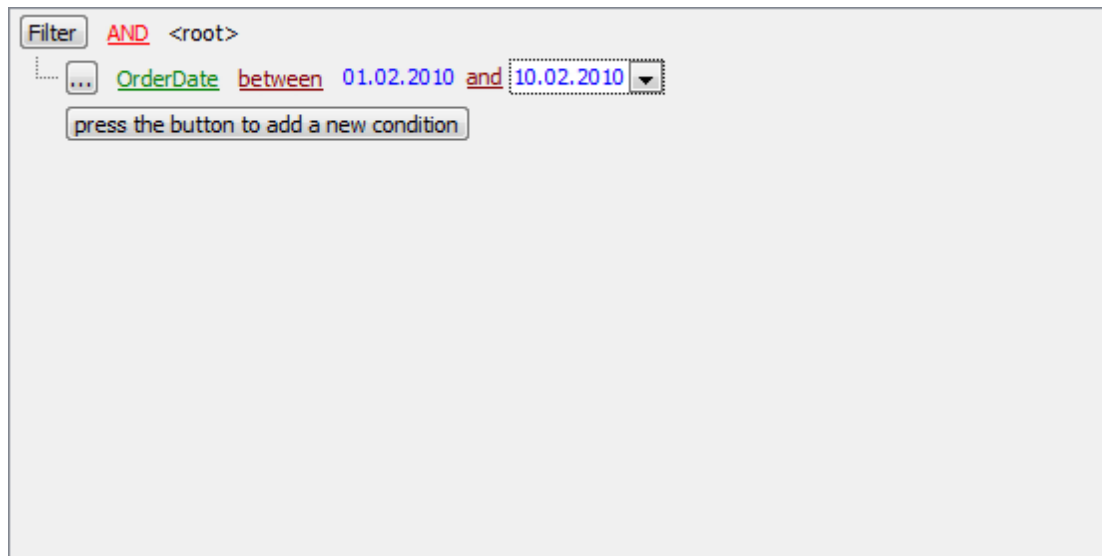
■ Setting an operator in the condition

Set the proper operator. In our example it is BETWEEN.



■ Setting criteria values in the condition

Next, you need to specify the range values for the selected operator. The editor used in value boxes is determined by the editor type assigned to the corresponding column.

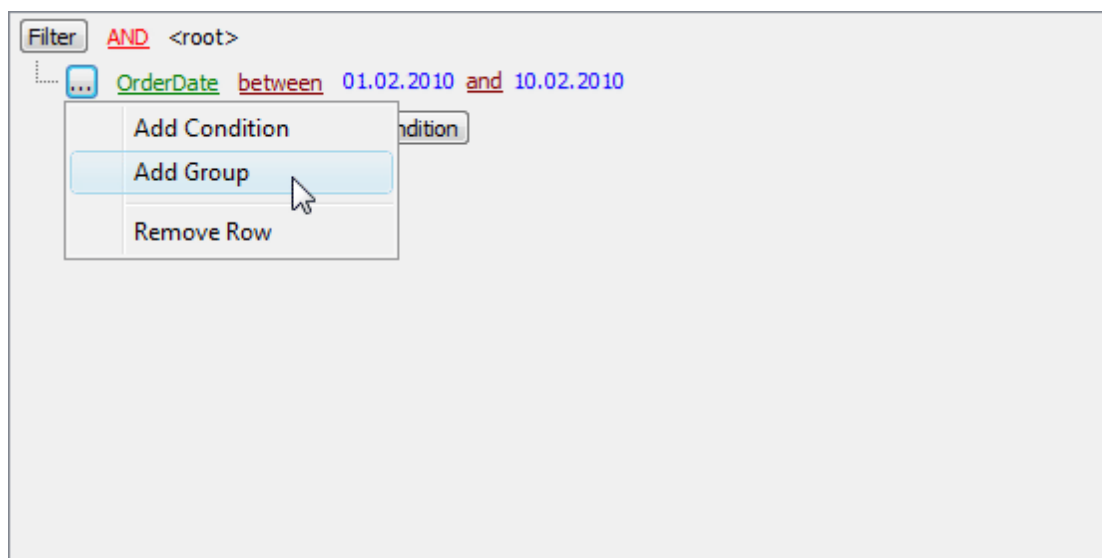


Now use the [Apply](#) button to see the filter result.

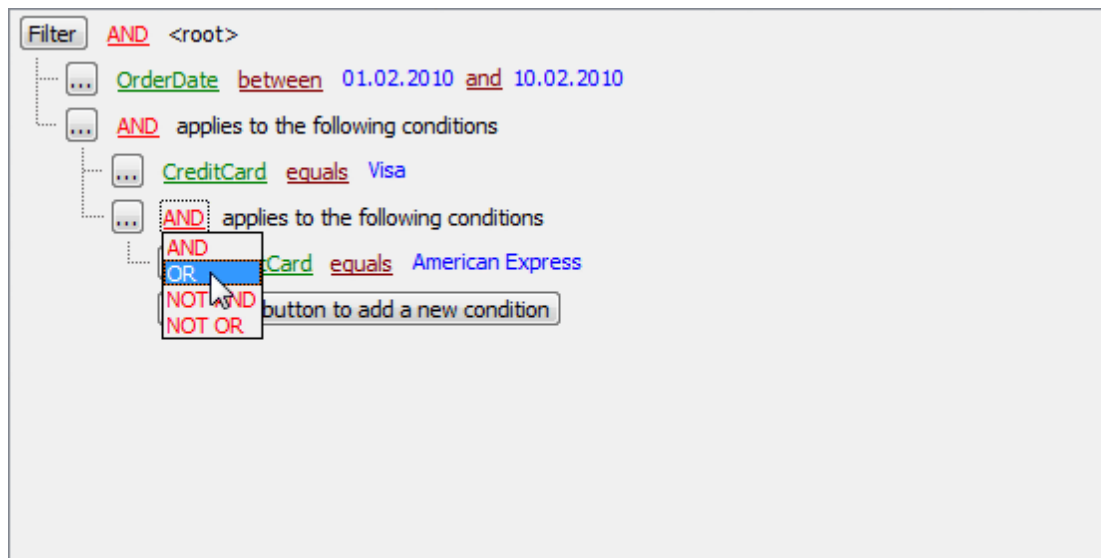
You can add additional conditions to the same root level to be combined by the AND operator.

Adding a new group

Suppose we need to select orders made between 01.02.2010 and 10.02.2010 and paid via 'Visa' or 'American Express'. This is a complex filter condition combining two simple conditions with the OR operator. Conditions from the same root level are combined by the AND operator. To add a condition combined with the previous one with the OR (NOT AND, NOT OR) operator, use a new group of conditions.



The next screen represents the finished filter conditions for this example.



See also: [Materialized View Editor](#)^[123]

5.4.2 Materialized Views Editor

Materialized View Editor allows you to edit the existing view definition (view name and the SELECT statement it implements), browse and update the view data.

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

The **Properties** tab allows you to rename the view, change the definition, the owner, the tablespace and the comment of the view.

The **Fields** tab represents fields included in the materialized view. Use grid's popup menu to describe or rename fields.

The **Body** area contains the query used to populate the view. To change the query, modify the SQL statement and use the **Compile materialized view** at the **Navigation bar**.

Is populated

This option is read-only and shows whether or not the materialized view is populated. If not, the materialized view is flagged as unscannable and cannot be queried until the **Refresh materialized view** is used.

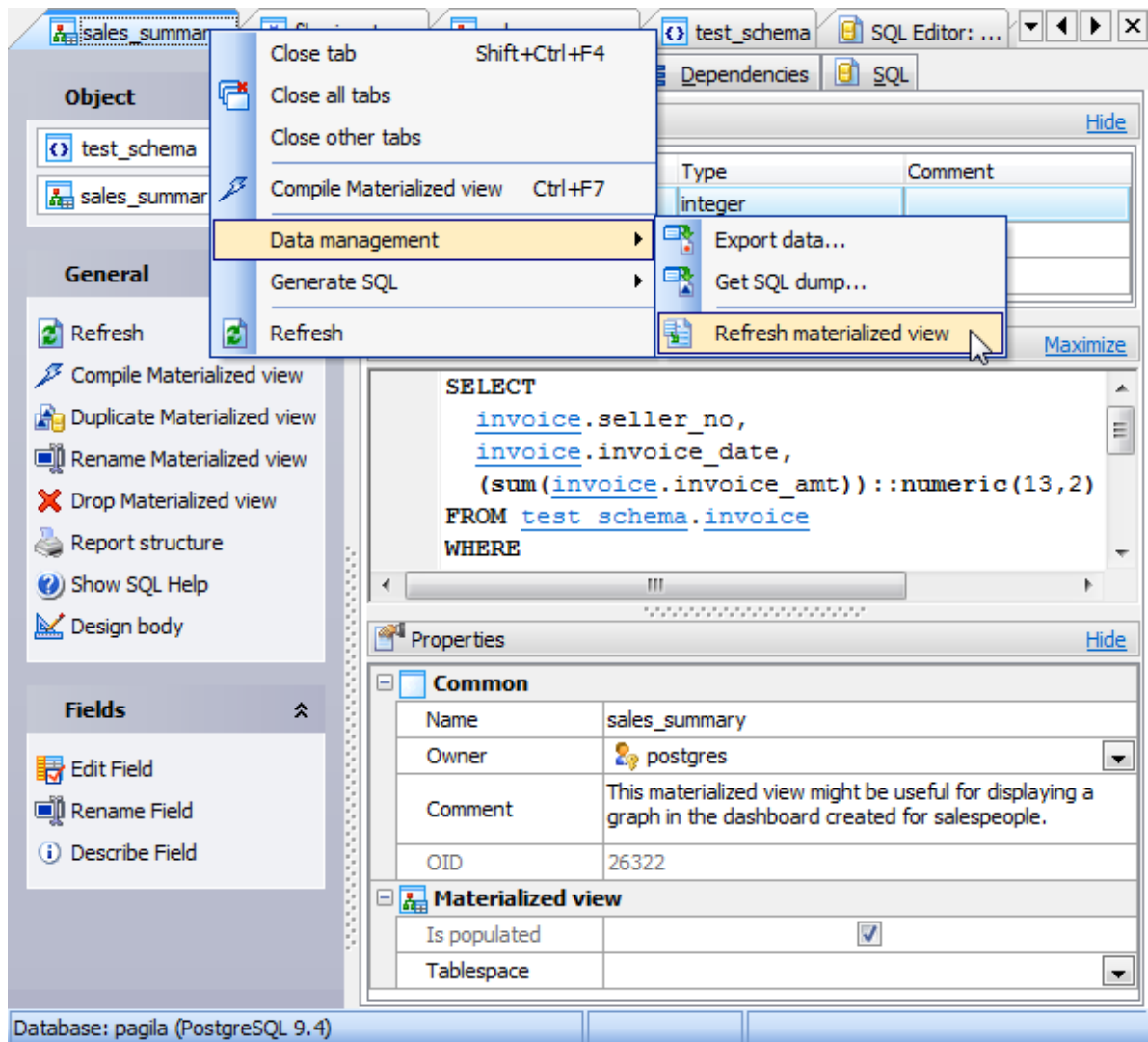
The **Data** tab displays current view data represented as a grid (see [Data View](#)^[229] for details). The popup menu of this tab and the Data Management navigation bar allow you to export data, get SQL dump, set the value of the selected record to *Null* or to *Now* (for **Date** values). In tables with BLOB fields you can also call [BLOB Editor](#)^[240] to view and edit the BLOB fields.

Refreshing materialized views

To completely replace the contents of a materialized view,

- open the popup menu of the editor tab or the view node at the Explorer tree,
- use the Refresh materialized view command of the Data Management options group

- check the [Leave in unscannable state](#) option in the Confirmation dialog to empty the materialized view and left in an unscannable state, or leave it unchecked to enforce the backing query execution to provide the view with fresh data.



See also: [Create Materialized View Wizard](#) ¹¹⁹

5.5 Functions

PostgreSQL provides four kinds of **functions**: *query language functions* (functions written in SQL), *procedural language functions* (functions written in, for example, PL/pgSQL or PL/Tcl), *internal functions*, and *C-language functions*. Every kind of function can take base types, composite types, or combinations of these as arguments (parameters). In addition, every kind of function can return a base type or a composite type. Functions may also be defined to return sets of base or composite values.

Many kinds of functions can take or return certain pseudo-types (such as polymorphic types), but the available facilities vary. SQL Anywhere allows to define user-specific database functions. In an SQL statement, you can then use these user-defined database functions in the same way as any other predefined functions.

■ How can I create a new Function?

New Functions are created within [Create Function Wizard](#)^[126]. In order to run the wizard you should either

- select the **Object | Create Database Object...** main menu item;
 - select the **Function** icon in the **Create Database Object** dialog
- or
- select the **Functions** list or any object from that list in the explorer tree;
 - select the **Create New Function...** item from the popup menu
- or
- open **Schema Editor** and the **Functions** tab there;
 - press the **Insert** key or select the **Create New Function** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

To create a new Function with the same properties as one of the existing Functions has:

- select the **Object | Duplicate Database Object...** main menu item;
- follow the instructions of **Duplicate Object Wizard**.

■ How can I edit an existing Function definition?

Functions can be edited within [FunctionEditor](#)^[128]. In order to open the editor you should either

- select the Function for editing in the explorer tree (type the first letters of the Function name for quick search);
 - select the **Edit Function** item from the popup menu
- or
- open **Schema Editor** and the **Functions** tab there;
 - select the Function to edit;
 - press the **Enter** key or select the **Edit Function** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

You can change the name of the Function using the **Rename**

Function dialog:

- select the Function to rename in the explorer tree;
- select the [Rename Function](#) item from the popup menu.

■ **How can I execute a Function?**

To execute the Function:

- select the Function in the explorer tree (type the first letters of the Function name for quick search);
- select the [Edit Function...](#) item from the popup menu;
- execute the Function using the [Execute](#) link of the [Navigation Bar](#)

or

- open [Schema Editor](#) and the [Functions](#) tab there;
- select the Function to execute;
- press the **Enter** key or select the [Edit Function](#) item from the popup menu, or use the corresponding link of the [Navigation Bar](#);
- execute the Function using the [Execute](#) link of the [Navigation bar](#).

■ **How can I drop a Function?**

To drop a Function:

- select the Function to drop in the explorer tree;
- select the [Drop Function](#) item from the popup menu

or

- open [Schema Editor](#) and the [Functions](#) tab there;
- select the Function to drop;
- press the **Delete** key or select the [Drop Function](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.5.1 Create Function Wizard

[Create Function Wizard](#) guides you through the process of creating a new Function. See [How To create Function](#)^[125] for instructions on running this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Specify the properties for the new Function according to your needs. The detailed description is given below.

Specifying Function properties

Name

Specify a name for the function.

Owner

Select the owner of the Function from the drop-down list. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

Supply a comment to the Function if necessary.

Check the **Returns set** option to indicate that the function will return a set of items rather than a single item.

Return type

Defines the data type of the function result.

Language

The field stores the name of the language the function is implemented in. Select either of the available languages: *SQL*, *C*, *internal*, or the name of a user-defined procedural language. For backward compatibility, the name may be enclosed by single quotes.

Check the **Strict** option to indicate that the function always returns NULL whenever any

of its arguments are null. If this option is specified, the function is not executed when there are null arguments; a null result is assumed automatically instead. Uncheck the [Strict](#) option to indicate that the function will be called normally when some of its arguments are null. It is then the function author's responsibility to check for null values if necessary and to respond in the appropriate way.

Execution Privileges

Select *Invoker* to indicate that the function is to be executed with the privileges of the user that calls it (the default value).

Select *Definer* to specify that the function is to be executed with the privileges of the user that created it.

Stability

Set the attribute to inform the system whether it is safe to replace multiple evaluations of the function with a single evaluation, for run-time optimization. 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; that is, it does not do database lookups or otherwise use information not directly present in its argument list. If this option is given, any call of the function with all-constant arguments can be immediately replaced with the function value.

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. This is the appropriate selection for the functions in which the results depend on database lookups, parameter variables (such as the current time zone), etc. Also note that the *current_timestamp* family of functions qualify as stable, since their values do not change within a transaction.

Volatile indicates that the function value can change even within a single table scan, so no optimizations can be made. Relatively few database functions are volatile in this sense; some examples are *random()*, *currval()*, *timeofday()*. Note that any function that has side-effects must be classified volatile (even if its result is quite predictable) to prevent calls from being optimized away; an example is *setval()*.

Note: You can also add the function definition within the [Properties](#)^[129] tab of [Function Editor](#)^[128].

Managing parameters of a new Function

Use popup menu [Add New Parameter...](#) item to add a new parameter and set its properties in [Parameter Editor](#)^[30]. Use the [Edit](#) and [Delete](#) items to manage Function parameters.

Specifying Function Definition

At this step you can specify the SQL definition for the new function. The step is optional: you can do it later using a non-modal editor.

5.5.2 Function Editor

[Function Editor](#) allows you to execute the existing Function, and edit its definition (Function name, parameter list, etc.). In order to open the editor you should either

- select the Function for editing in the explorer tree (type the first letters of the Function name for quick search);
- select the [Edit Function](#) item from the popup menu

or

- open [Schema Editor](#) and the [Functions](#) tab there;
- select the Function to edit;
- press the **Enter** key or select the [Edit Function](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

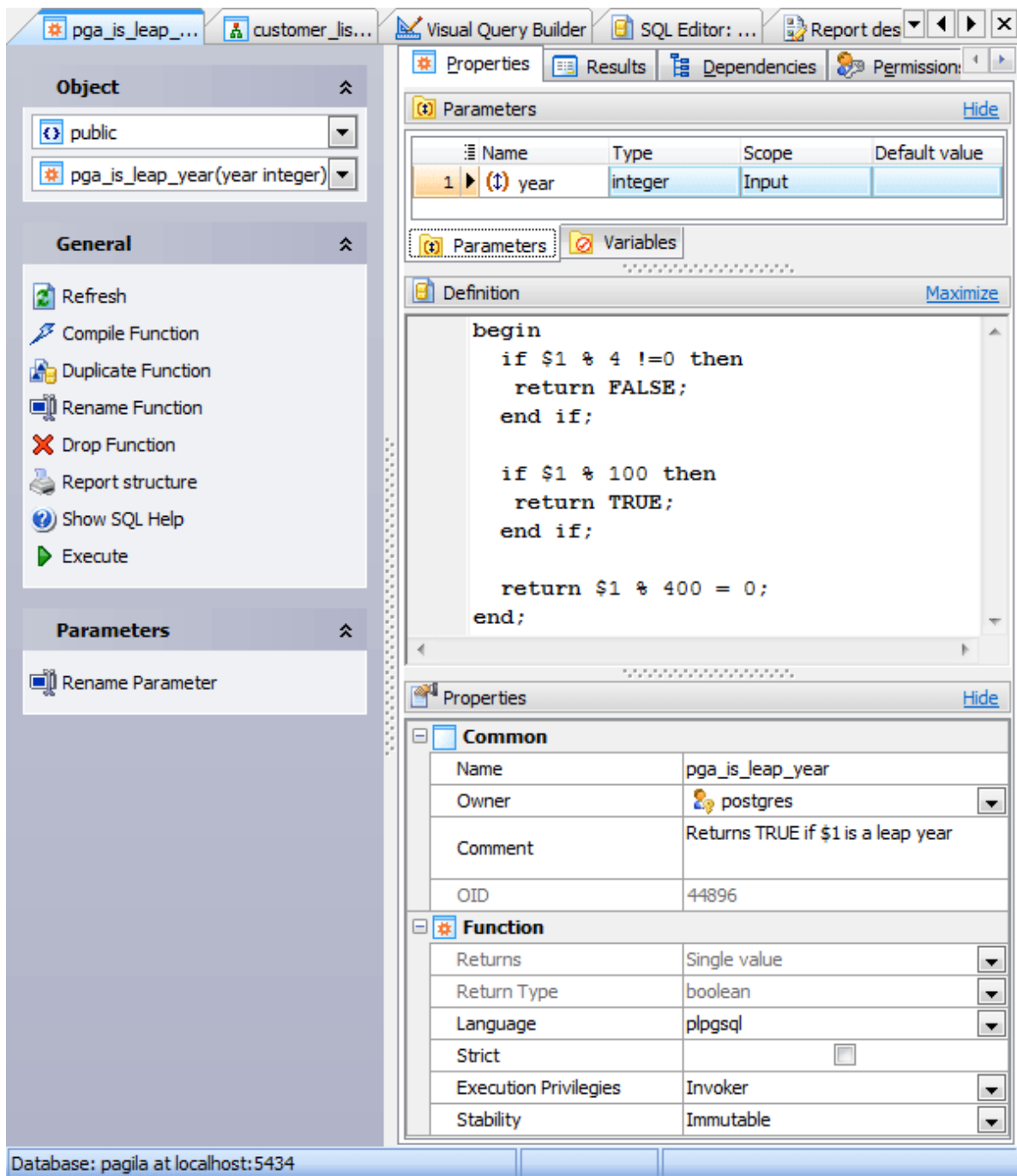
- [Editing Function properties](#)^[129]
- [Viewing Function results](#)^[131]

5.5.2.1 Editing properties

The [Parameters](#) tab contains the list of the current Function parameters with its options. Here you can view the [Name](#) and the [Type](#) of each Function parameter and also view its [Scope](#) and supply a [Comment](#) for the parameter.

Parameters can be edited within the [Parameter Editor](#) dialog window. In order to open the dialog you should

- open the object in its editor and the [Parameters](#) tab there;
 - select the parameter to edit;
 - press the **Enter** key or select the [Edit Parameter](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).
- The [Definition](#) field contains the definition of the Function. Specify a string constant defining the Function here; the meaning depends on the language. It may be an internal Function name, the path to an object file, an SQL command or text in a procedural language.



Name

You can edit the Function name here. The name of the Function must be unique among all the Function names in the database.

Owner

The field contains the owner of the Function. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

Specify a comment to the Function if necessary.

OID

The field stores the OID (object identifier) of the function. An OID can be defined as a serial number that is automatically added by PostgreSQL to all functions.

Parameters

Here you can find the list of the function parameters.

The [Returns set](#) option is checked to indicate that the function will return a set of items rather than a single item.

Return type

The field defines the data type of the function result.

Language

Specify the name of the language the function is implemented in.

Check the [Strict](#) option to indicate that the function always returns NULL whenever any of its arguments are null. If this option is specified, the function is not executed when there are null arguments; a null result is assumed automatically instead. Uncheck the [Strict](#) option to indicate that the function will be called normally when some of its arguments are null. It is then the function author's responsibility to check for null values if necessary and to respond in the appropriate way.

Execution Privileges (*Invoker, Definer*)

Invoker indicates that the function is to be executed with the privileges of the user that calls it.

Definer specifies that the function is to be executed with the privileges of the user that created it.

Stability (*Immutable, Stable, Volatile*)

Set the attribute informing the system whether it is safe to replace multiple evaluations of the function with a single evaluation, for run-time optimization.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.5.2.2 Viewing Function results

[Function Editor](#) provides an opportunity to execute current routine by opening the [Results](#) tab, by clicking the [Execute](#) item of the [Navigation Bar](#), or by pressing the **F9** key.

If the Function has parameters, PostgreSQL Maestro will ask you to specify the values for these parameters in the [Input parameters](#) dialog which appears before the procedure execution. (for PostgreSQL server 8.1+, if the function has parameters marked as IN or INOUT) [Input parameters](#) dialog allows you to specify the values for all input parameters. After changes are made, click the [OK](#) button to execute the Function, or the [Cancel](#) button to abort the execution.

Parameter Values	
(↕) @conference_id int	2
(↕) @division_id int	2
(↕) @team_id int	1

PostgreSQL Maestro supports [Parameter History](#). Values that have been set previously as the routine parameters are represented in the [History](#) tab of the [Input Parameter](#) dialog with a date and time of their last using. Double click a necessary set of values to set them as the routine parameters. You can manage the [Parameter History](#) with [Delete history item](#) and [Clear history](#) links.

The result of the successfully executed routine can be found within the [Results](#) tab of [Function Editor](#).

Note: If any unsaved changes are applied to the routine being currently edited, the execution of the routine is impossible until changes are saved by the [Compile](#) procedure item of the [Navigation Bar](#).

5.6 Domains

Data domains are useful for abstracting common table fields into a single location for maintenance. For example, *an email address* column may be used in several tables, all with the same properties. Define a domain and use that instead of setting up each table constraint individually.

■ How can I create a new Domain?

New Domains are created within Create Domain Wizard. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Domain](#) icon in the [Create Database Object](#) dialog
- or
- select the [Domains](#) list or any object from that list in the explorer tree;
 - select the [Create New Domain...](#) item from the popup menu
- or
- open [Schema Editor](#) and the [Domains](#) tab there;
 - press the **Insert** key or select the [Create New Domain](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new Domain with the same properties as one of the existing Domains has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing Domain?

Domains can be edited within [Domain Editor](#)^[136]. In order to run the editor you should either

- select the Domain for editing in the explorer tree (type the first letters of the Domain name for quick search);
 - select the [Edit Domain...](#) item from the popup menu
- or
- open [Schema Editor](#) and the [Domains](#) tab there;
 - select the Domain to edit;
 - press the **Enter** key or select the [Edit Domain](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a Domain?

To drop a Domain:

- select the Domain to drop in the explorer tree;

- select the [Drop Domain](#) item from the popup menu
- or
- open [Schema Editor](#) and the [Domains](#) tab there;
 - select the Domain to drop;
 - press the **Delete** key or select the [Drop Domain](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.6.1 Create Domain Wizard

New Domains are created within Create Domain Wizard. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Domain](#) icon in the [Create Database Object](#) dialog
- or
- select the [Domains](#) list or any object from that list in the explorer tree;
 - select the [Create New Domain...](#) item from the popup menu
- or
- open [Schema Editor](#) and the [Domains](#) tab there;
 - press the **Insert** key or select the [Create New Domain](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new Domain with the same properties as one of the existing Domains has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Specify the properties for the new Domain according to your needs. The detailed description is given below.

Create domain wizard

Create New Domain

Edit domain options

Properties

Common

Name	us_postal_code
Owner	
Comment	A regular expression test is used to verify that the value looks like a valid US postal code.

Domain

Data type	text
Not null	<input checked="" type="checkbox"/>
Default value	
Collation	

Help < Back Next > Cancel

Name

Specify a name for the domain.

Owner

Select the owner of the Domain from the drop-down list. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

Supply a comment to the Domain if necessary.

Data Type

Select the underlying data type for the Domain. This may include array specifiers.

☒ Not Null

The checkbox indicates that the values of the Domain are not allowed to be null.

Collation

Use the field to specify the sort order and character classification behavior of data. If no collation is specified, the underlying data type's default collation is used.

Default Value

The field is used to specify a default value for the columns of the domain data type. The field value may contain any variable-free expression (however, subqueries are not allowed). The data type of the default expression must match the data type of the

domain. If no default value is specified then the default value is the null value.

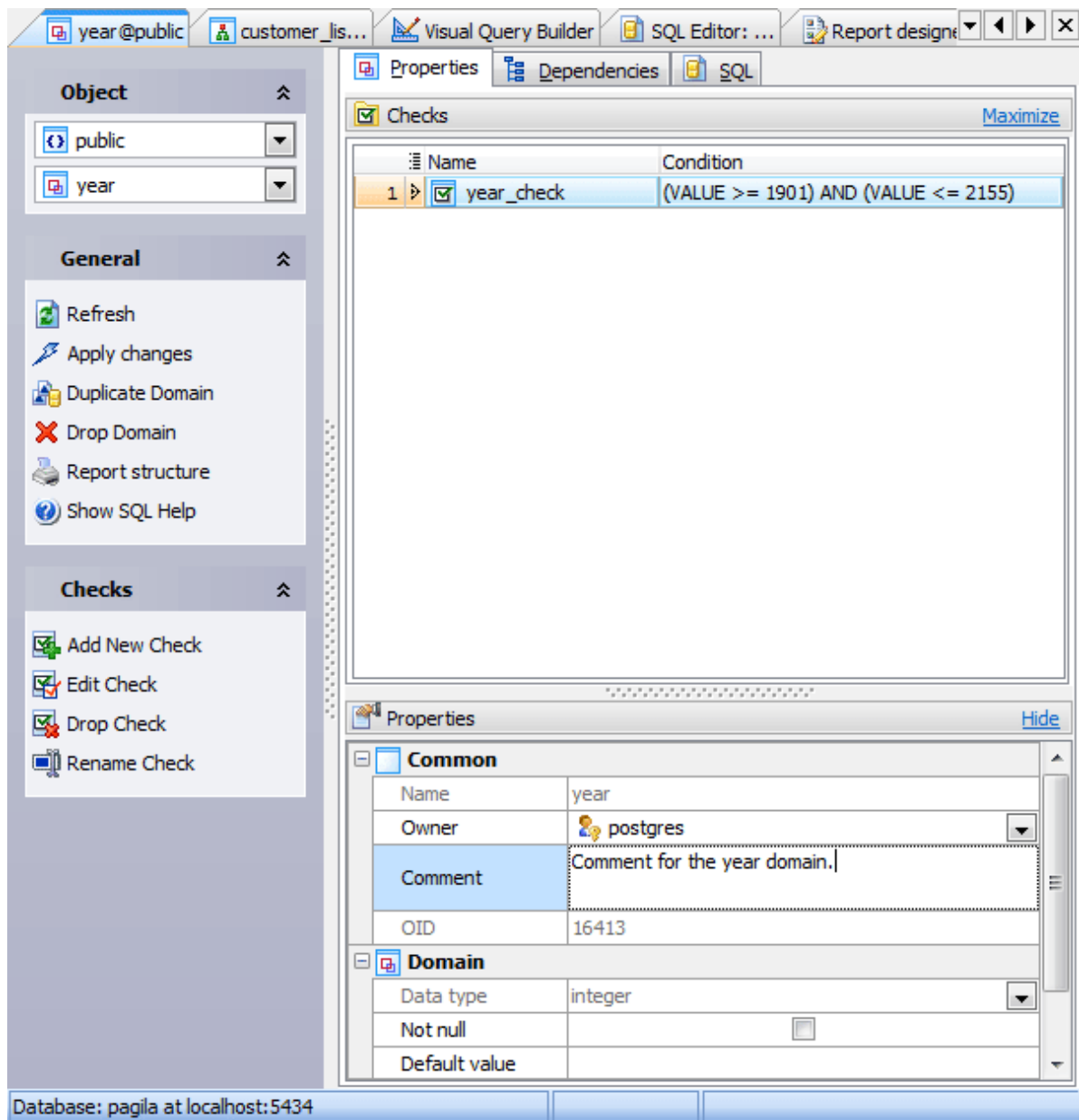
Use popup menu to add a new check and set its properties in [Check Properties](#)^[91] modal window. You can also use the popup menu items to edit or delete the checks of domain being created.

5.6.2 Domain Editor

[Domain Editor](#) is opened automatically after a new Domain is created and is available on editing the existing one (see [Edit Domain](#)^[133] for details).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

[Domain Editor](#) provides you with an ability to edit Domain properties. The [Properties](#) tab allows you to change the Domain name, the Domain owner, etc. You can also find the OID there.



The **Checks** tab of the **Domain Editor** represents the domain check list. **Check** clauses specify integrity constraints or tests to be satisfied by values of the domain. Each constraint must be an expression producing a Boolean result. It should use the name **VALUE** to refer to the value being tested.

The tab includes three columns: **Check name**, **Condition** and **Comment**.

Check name

Here you can specify the domain check name.

Condition

The column stores the condition which must be complied with by a new or updated

values of the domain for an insert or update operations to succeed.

[New comment](#)

This field stores a comment to the domain check.

You can also use Navigation bar to manage [Domain checks](#).

[Name](#)

Here you can view the Domain name. The name of the Domain must be unique among all the Domain names in the schema.

[Owner](#)

By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

[Comment](#)

This field stores a comment to the Domain.

[OID](#)

The field stores the OID (object identifier) of the domain. OID can be defined as a serial number that is automatically added by PostgreSQL to all domains.

The [Base Type](#) represents the SQL Server supplied data type on which the alias data type is based.

☒ [Not Null](#)

The checkbox indicates that the values of the Domain are not allowed to be null.

[Default Value](#)

The field is meant to specify a default value for the domain data type.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.7 Aggregates

Like most other relational database products, PostgreSQL supports aggregate functions. An aggregate function computes a single result from multiple input rows. Some basic and commonly-used aggregate functions are included with the distribution. If one defines new types or needs an aggregate function not already provided, then an [Aggregate](#) can be used to provide the stated features.

■ How can I create a new aggregate?

New aggregates are created within [Create Aggregate Wizard](#)^[140]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Aggregate](#) icon in the Create Database Object dialog
- or
- select the [Aggregates](#) list or any object from that list in the explorer tree;
 - select the [Create New Aggregate...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
 - press the **Insert** key or select the [Create New Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new aggregate with the same properties as one of the existing aggregates has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing aggregate?

Aggregates can be edited within [Aggregate Editor](#)^[141]. In order to run the editor you should either

- select the aggregate for editing in the explorer tree (type the first letters of the aggregate name for quick search);
 - select the [Edit Aggregate ...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
 - select the aggregate to edit;
 - press the **Enter** key or select the [Edit Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the aggregate using the [Rename Aggregate](#) dialog. To open the dialog you should either

- select the aggregate to rename in the explorer tree;
 - select the [Rename Aggregate](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
 - select the aggregate to rename;
 - select the [Rename Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop an aggregate?

To drop an aggregate:

- select the aggregate to drop in the explorer tree;
 - select the [Drop Aggregate](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
 - select the aggregate to drop;
 - press the **Delete** key or select the [Drop Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.7.1 Create Aggregate Wizard

New aggregates are created within [Create Aggregate Wizard](#)^[140]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Aggregate](#) icon in the Create Database Object dialog
- or
- select the [Aggregates](#) list or any object from that list in the explorer tree;
 - select the [Create New Aggregate...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
 - press the **Insert** key or select the [Create New Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new aggregate with the same properties as one of the existing aggregates has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Several options can be specified for the aggregate function that govern the expected

behaviour of the function. The detailed description is given below.

Owner

Defines the owner of the new aggregate. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

The box allows you to set optional text to describe the new aggregate.

Base Type

The field is used for setting the input data type on which this aggregate function operates. This can be specified as "ANY" for an aggregate that does not examine its input values (an example is `count(*)`).

State Type

The data type for the state value of the aggregate.

State Function

The name of the state transition function to be called for each input data value. This is normally a function of two arguments, the first being of type state type and the second of type base type. Alternatively, for an aggregate that does not examine its input values, the function takes just one argument of type state type. In either case the function must return a value of type state type. This function takes the current state value and the current input data item, and returns the next state value.

Final Function

The name of the final function called to compute the aggregate's result after all input data has been traversed. The function must take a single argument of type state type. The return data type of the aggregate is defined as the return type of this function. If final function is not specified, then the ending state value is used as the aggregate result, and the return type is state type.

Initial Condition

The initial setting for the state value. This must be a string constant in the form accepted for the data type state type. If not specified, the state value starts out null.

See also: [Aggregate Editor](#)¹⁴¹

5.7.2 Aggregate Editor

Aggregates can be edited within [Aggregate Editor](#)¹⁴¹. In order to run the editor you should either

- select the aggregate for editing in the explorer tree (type the first letters of the aggregate name for quick search);
- select the [Edit Aggregate ...](#) item from the popup menu

or

- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
- select the aggregate to edit;
- press the **Enter** key or select the [Edit Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the aggregate using the [Rename Aggregate](#) dialog. To open the dialog you should either

- select the aggregate to rename in the explorer tree;
- select the [Rename Aggregate](#) item from the popup menu

or

- open the schema in [Schema Editor](#) and the [Aggregates](#) tab there;
- select the aggregate to rename;
- select the [Rename Aggregate](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

- [Editing aggregate properties](#)^[142]

See also: [Create Aggregate Wizard](#)^[140]

5.7.2.1 Editing aggregate properties

[Aggregate Editor](#) provides you with an ability to edit aggregate properties. The [Properties](#) tab allows you to change the aggregate name, the aggregate owner and the comment for the aggregate. You can also find the OID there.

[Name](#)

Here you can change the aggregate name.

[Owner](#)

Set the owner for the aggregate. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

[Comment](#)

This field stores a comment to the aggregate.

[OID](#)

This field stores the aggregate OID (object identifier). The latter can be defined as a serial number that is automatically added by PostgreSQL to all aggregates.

[Base Type](#)

Here is the input data type on which this aggregate function operates.

[State Type](#)

The field contains the data type for the state value of the aggregate.

The [State Function](#) stores the name of the state transition function to be called for each input data value.

[Final Type](#)

The return data type of the aggregate is supplied here.

The [Final Function](#) field represents the name of the final function called to compute the aggregate result after all input data has been traversed.

[Initial Condition](#)

It is possible to edit the initial setting for the state value here.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.8 Sequences

A [Sequence](#) in PostgreSQL is a new sequence number generator. This involves creating and initializing a new special single-row table. The generator will be owned by the user issuing the command.

■ How can I create a new sequence?

New sequences are created within [Create Sequence Wizard](#)¹⁴⁵. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Sequence](#) icon in the Create Database Object dialog
- or
- select the [Sequences](#) list or any object from that list in the explorer tree;
 - select the [Create New Sequence...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Sequences](#) tab there;
 - press the **Insert** key or select the [Create New Sequence](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new sequence with the same properties as one of the existing sequences has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing sequence?

Sequences can be edited within [Sequence Editor](#)¹⁴⁶. In order to run the editor you should either

- select the sequence for editing in the explorer tree (type the first letters of the sequence name for quick search);
 - select the [Edit Sequence ...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Sequences](#) tab there;
 - select the sequence to edit;
 - press the **Enter** key or select the [Edit Sequence](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a sequence

To drop a sequence:

- select the sequence to drop in the explorer tree;

- select the [Drop Sequence](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Sequences](#) tab there;
 - select the sequence to drop;
 - press the **Delete** key or select the [Drop Sequence](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.8.1 Create Sequence Wizard

Sequences may be created with [Create Sequence Wizard](#). Just specify the wizard options according to your needs.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Name

The field allows you to specify the new sequence name being set on the previous wizard step.

Owner

Defines the owner of the new sequence.

Increment

Specify 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. The default value is 1.

Max Value

Determine the maximum value for the sequence. If this clause is not supplied or NO MAXVALUE is specified, then default values will be used. The defaults are $2^{63}-1$ and -1 for ascending and descending sequences, respectively.

Min Value

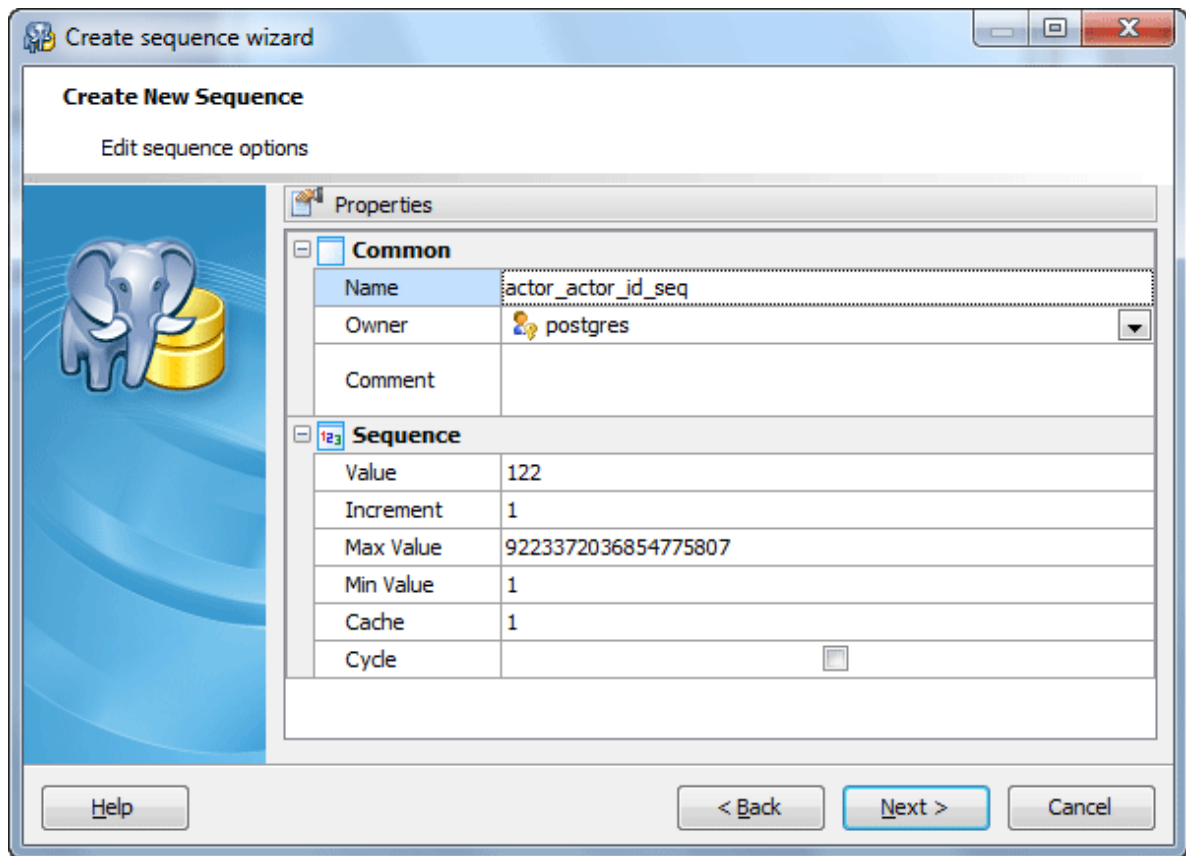
Determine the minimum value a sequence can generate. If this clause is not supplied or NO MINVALUE is specified, then defaults will be used. The defaults are 1 and $-2^{63}-1$ for ascending and descending sequences, respectively.

Cashe

Specify 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

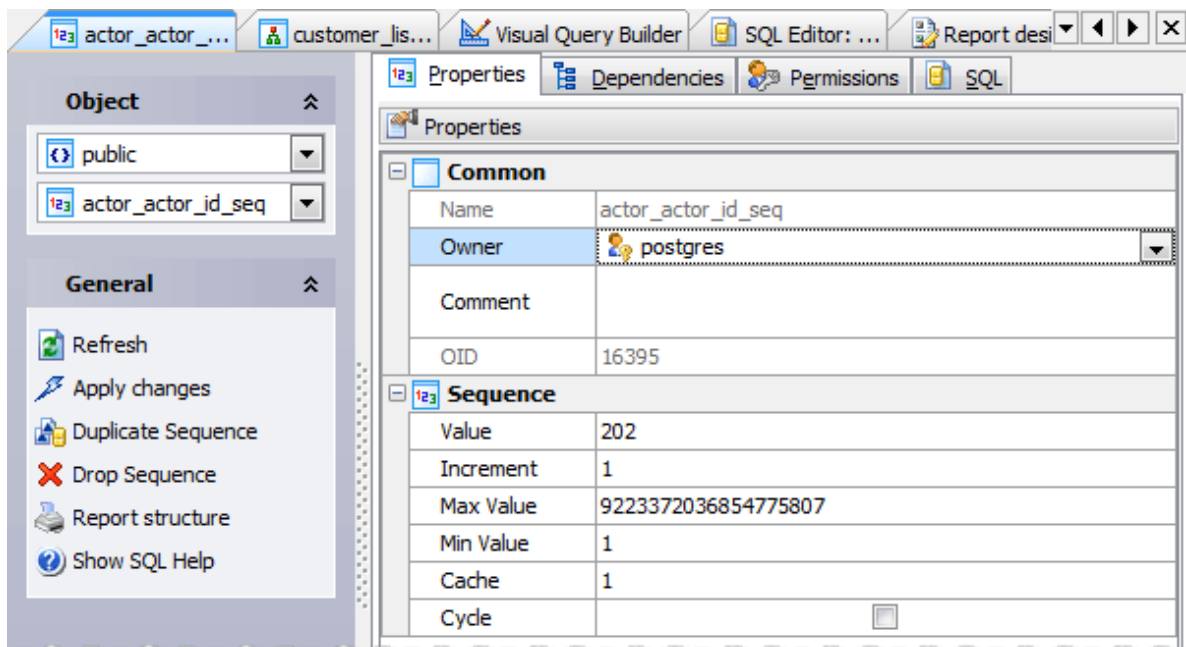
The CYCLE option allows the sequence to wrap around when the *maxvalue* or *minvalue* has been reached by an ascending or descending sequence respectively. If the limit is reached, the next number generated will be the minvalue or maxvalue, respectively.



5.8.2 Sequence Editor

Use [Sequence Editor](#) to change properties of existing sequences. The editor can be opened automatically after [the sequence is created](#)^[145] or from the [Explorer Tree](#) and [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.



Name

Here you can rename the sequence.

Owner

Shows the owner of the sequence. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Increment

Here you can edit the value which is added to the current sequence value to create a new value.

The **Max Value** and the **Min Value** contain the maximum and the minimum values for the sequence.

Cache

Specify how many sequence numbers are to be preallocated and stored in memory for faster access.

☒ Cycle

The checkbox represents whether the sequence is cycle.

To apply the changes, select the **Apply Changes** item in the **Navigation bar** or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the **Object Properties** item of the popup menu of the selected object from the explorer tree.

5.9 Types

PostgreSQL Maestro supports PostgreSQL user-defined types - data types for use in the current database.

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.

■ How can I create a new type?

New types are created within [Create Type Wizard](#)^[149]. In order to run the wizard you should either

- select the **Object | Create Database Object...** main menu item;
 - select the **Type** icon in the **Create Database Object** dialog
- or
- select the **Types** list or any object from that list in the explorer tree;
 - select the **Create New Type...** item from the popup menu
- or
- open the schema in **Schema Editor** and the **Types** tab there;
 - press the **Insert** key or select the **Create New Type** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

To create a new type with the same properties as one of the existing types has:

- select the **Object | Duplicate Database Object...** main menu item;
- follow the instructions of **Duplicate Object Wizard**.

■ How can I edit an existing type?

Types can be edited within [Type Editor](#)^[151]. In order to run the editor you should either

- select the type for editing in the explorer tree (type the first letters of the type name for quick search);
 - select the **Edit Type ...** item from the popup menu
- or
- open the schema in [Schema Editor](#)^[70] and the **Types** tab there;
 - select the type to edit;
 - press the **Enter** key or select the **Edit Type** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

■ How can I drop a type?

To drop a type:

- select the type to drop in the explorer tree;
- select the [Drop Type](#) item from the popup menu

or

- open the schema in [Schema Editor](#) and the [Types](#) tab there;
- select the type to drop;
- press the **Delete** key or select the [Drop Type](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.9.1 Create Type Wizard

[Create Type Wizard](#) guides you through the process of creating a new schema type.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)⁴³. Below you will find a description of wizard steps that are unique for the current object.

Type properties

At the top of the Properties tab define an [Owner](#), and a [Comment](#) of the new type. The type name is represented here as it was set at the previous step.

Input Function

Enter the name of a function that converts data from the type external textual form to its internal form.

Output Function

Specify the name of a function that converts data from the type internal form to its external textual form.

Receive Function

The field contains the name of a function that converts data from the type external binary form to its internal form.

Send Function

The name of a function that converts data from the type internal form to its external binary form.

Analyze Function

Specify the name of a function that performs statistical analysis for the data type.

Internal Length

A numeric constant that specifies the length in bytes of the new type internal representation. The default assumption is that it is *variable-length*.

Default

The default value for the data type. If omitted, the default is null.

Element

The type being created is an array; specify the type of the array elements here.

Delimiter

The delimiter character to be used between values in arrays made of this type.

Alignment (*char, int2, int4, or double*)

Specify the storage alignment requirement of the data type. The default is *int4*.

Passed By Value

The optional flag indicates that values of this data type are passed by *value* rather than by *reference*.

Storage (*Plain, External, Extended, or Main*)

Specify the storage strategy for the data type. The default is *Plain*.

Create type wizard

Create New Type

Edit type options

Properties

Common

Name	citext
Owner	postgres
Comment	

Type

Input Function	public.citextin
Output Function	public.citextout
Receive Function	public.citextrecv
Send Function	public.citextsend
Analyze Function	
Internal Length	-1
Default	
Element	
Delimiter	
Alignment	int4
Passed By Value	<input type="checkbox"/>
Storage	MAIN

Help < Back Next > Cancel

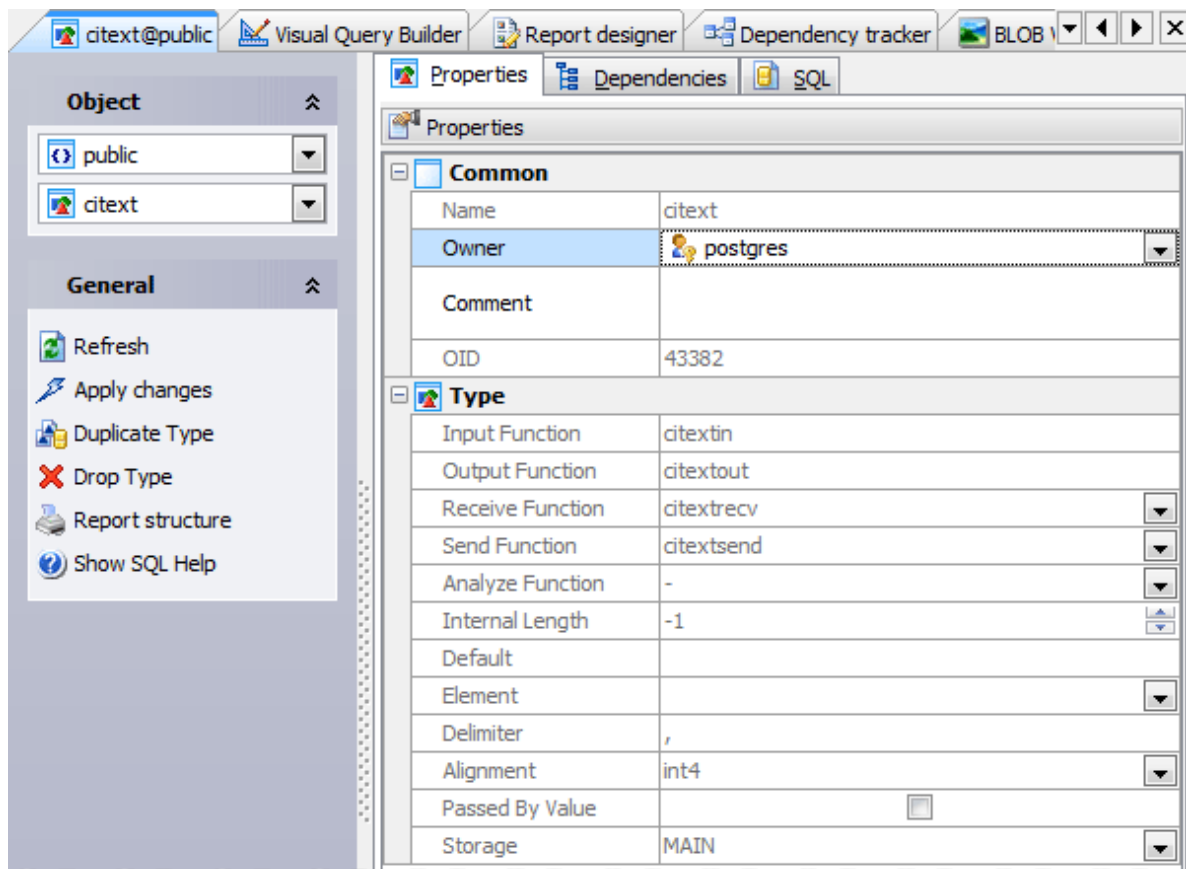
5.9.2 Type Editor

Type Editor is the basic PostgreSQL Maestro tool for working with existing types. You can open a type in **Type Editor** from the **Explorer Tree** or **Object Manager**.

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

The description of type options you can find at the [corresponding topic](#)^[15]

See also: [Create Type Wizard](#)^[149]



5.9.2.1 Editing type options

Type Editor provides you with an ability to edit type properties. The **Properties** tab allows you to change the type name, the type owner; input, output, receive, send and analyze functions, the comment for the type and other type properties. You can also find the OID there.

Name

Here you can change the type name.

Owner

The field contains the owner of the type. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it,

privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

This field stores a comment to the type.

OID

The type OID (object identifier) is stored in this field. This is a serial number that is automatically added by PostgreSQL to all types.

Input Function

You can edit the name of a function that converts data from the type external textual form to its internal form here.

Output Function

Specify the name of a function that converts data from the type internal form to its external textual form.

Receive Function

It is the name of a function that converts data from the type external binary form to its internal form.

Send Function

The name of a function that converts data from the type internal form to its external binary form.

Analyze Function

There is the name of a function that performs statistical analysis for the data type.

Internal Length

It is a numeric constant that specifies the length in bytes of the new type internal representation. The default assumption is that it is variable-length.

Default

The default value for the data type. If this is omitted, the default is null.

Element

The type being created is an array; specify the type of the array elements here.

Delimiter

The delimiter character to be used between values in arrays made of this type.

Alignment

I'll find the storage alignment requirement of the data type here.

Passed By Value

It is the optional flag indicating that values of this data type are passed by value, rather than by reference.

Storage

Define the storage strategy for the data type.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.10 Composite and Enum Types

PostgreSQL Maestro represents composite types and enum types as different objects, but work with them is similar. So, we describe them in one manual section.

A [composite type](#) describes the structure of a row or record; it is in essence just a list of field names and their data types. PostgreSQL allows values of composite types to be used in many of the same ways that simple types can be used. For example, a *column of a table* can be declared to be of a composite type.

[Enumerated \(enum\) types](#) are data types that comprise a static, ordered set of values. They are equivalent to the enum types supported in a number of programming languages. An example of an enum type might be the days of the week, or a set of status values for a piece of data.

■ How can I create a new type?

New composite and enum types are created within [Create Composite Type Wizard](#)^[155] and [Create Enum Type Wizard](#)^[155] respectively. In order to run these wizards you should either

- select the [Object | Create Database Object...](#) main menu item;
- select the [Composite Type](#) or [Enum Type](#) icon in the Create Database Object dialog

or

- select the [Composite \(Enum\) Types](#) list or any object from that list in the explorer tree;
- select the [Create New Composite \(Enum\) Type...](#) item from the popup menu

or

- open the schema in [Schema Editor](#) and the [Composite Types \(Enum Types\)](#) tab there;
- press the **Insert** key or select the [Create New Composite \(Enum\) Type](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new composite type with the same properties as one of the existing composite type has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing type?

Composite Types and Enum Typed are edited within [Composite Type Editor](#)^[157] and [Enum Type Editor](#)^[157] respectively. In order to run these editors you should either

- select the composite type (enum type) for editing in the explorer tree (type the first letters of the type name for quick search);
- select the [Edit Composite \(Enum\) Type ...](#) item from the popup

menu

or

- open the schema in [Schema Editor](#) and the [Composite \(Enum\) Types](#) tab there;
- select the type to edit;
- press the **Enter** key or select the corresponding item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a type?

To drop a composite or enum type:

- select the type to drop in the explorer tree;
- select the corresponding item from the popup menu

or

- open the schema in [Schema Editor](#) and the [Composite Types \(Enum Types\)](#) tab there;
- select the type to drop;
- press the **Delete** key or select the corresponding item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.10.1 Create Type Wizard (Composite and Enum Types)

[Create Composite Type Wizard](#) and [Create Enum Type Wizard](#) allows you to add a new schema composite type or enum type respectively.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Type properties

Owner

Defines the owner of the new composite type. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

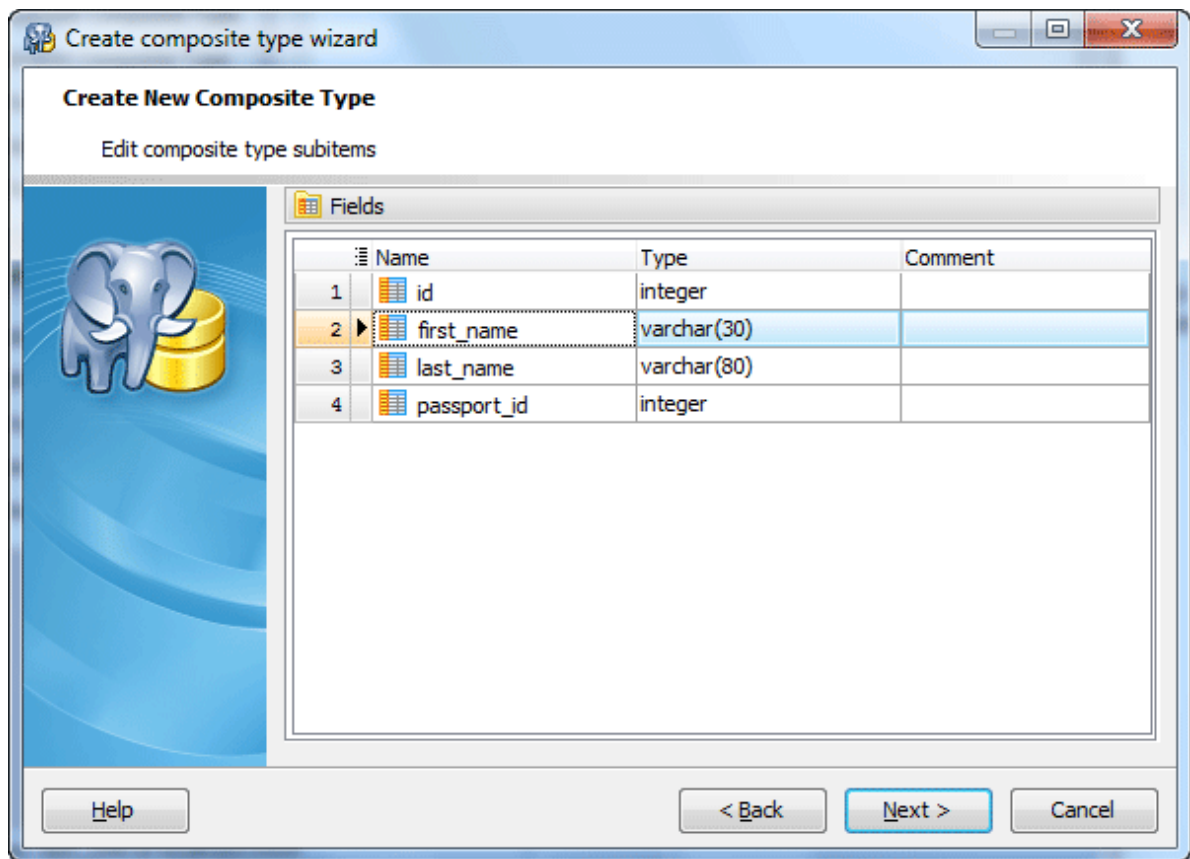
The box allows you to set optional text to describe the new composite type.

Type subitems

On the next step the wizards allow you to set fields of composite types and values of enum types accordingly:

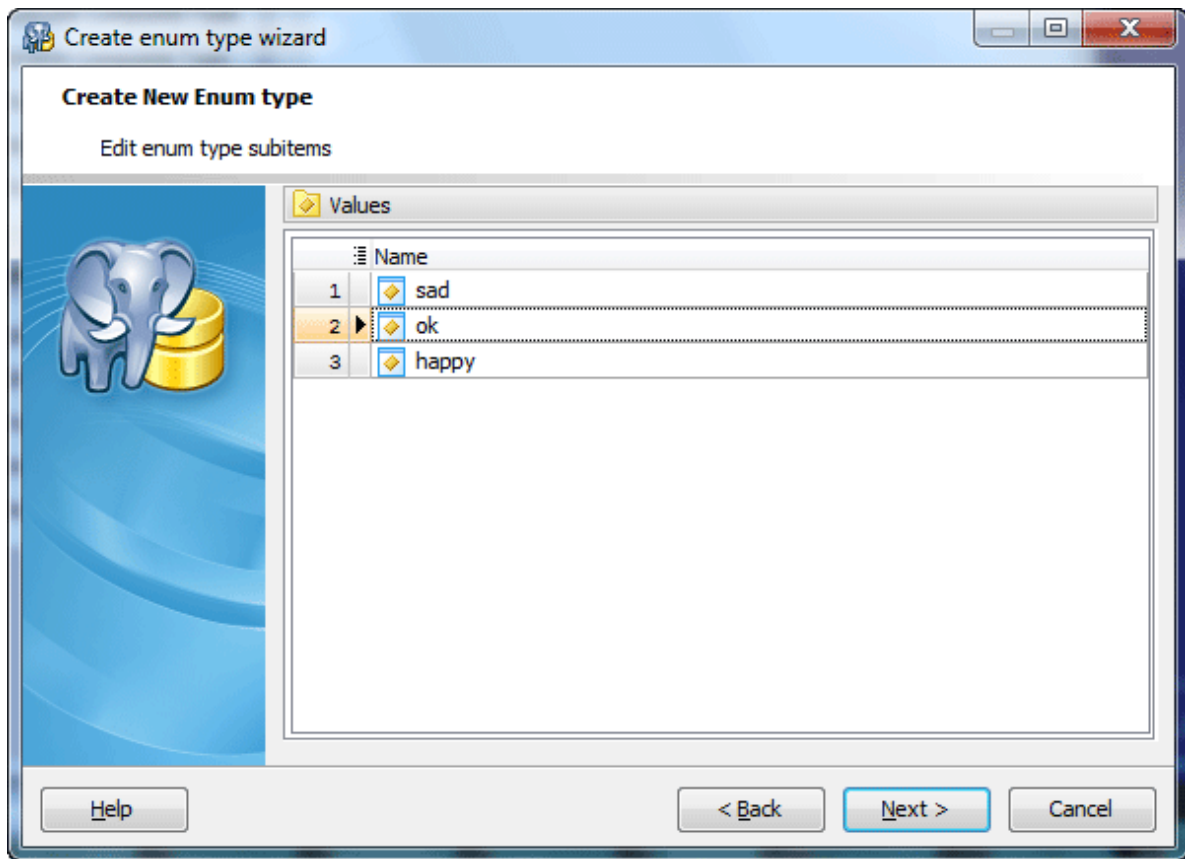
Composite type field list

Specify options for fields of the new composite type. Click the [Add](#) button at the tab area to add a new field and set its properties in [Field Editor](#)^[44]. The fields are available for editing and reordering with the corresponding popup menu items.



Enum type value list

Add values to the new enum type. Click the [Add](#) button at the tab area to add a new value and enter it to the corresponding window. Each of which must be less than 64 in a standard PostgreSQL build.



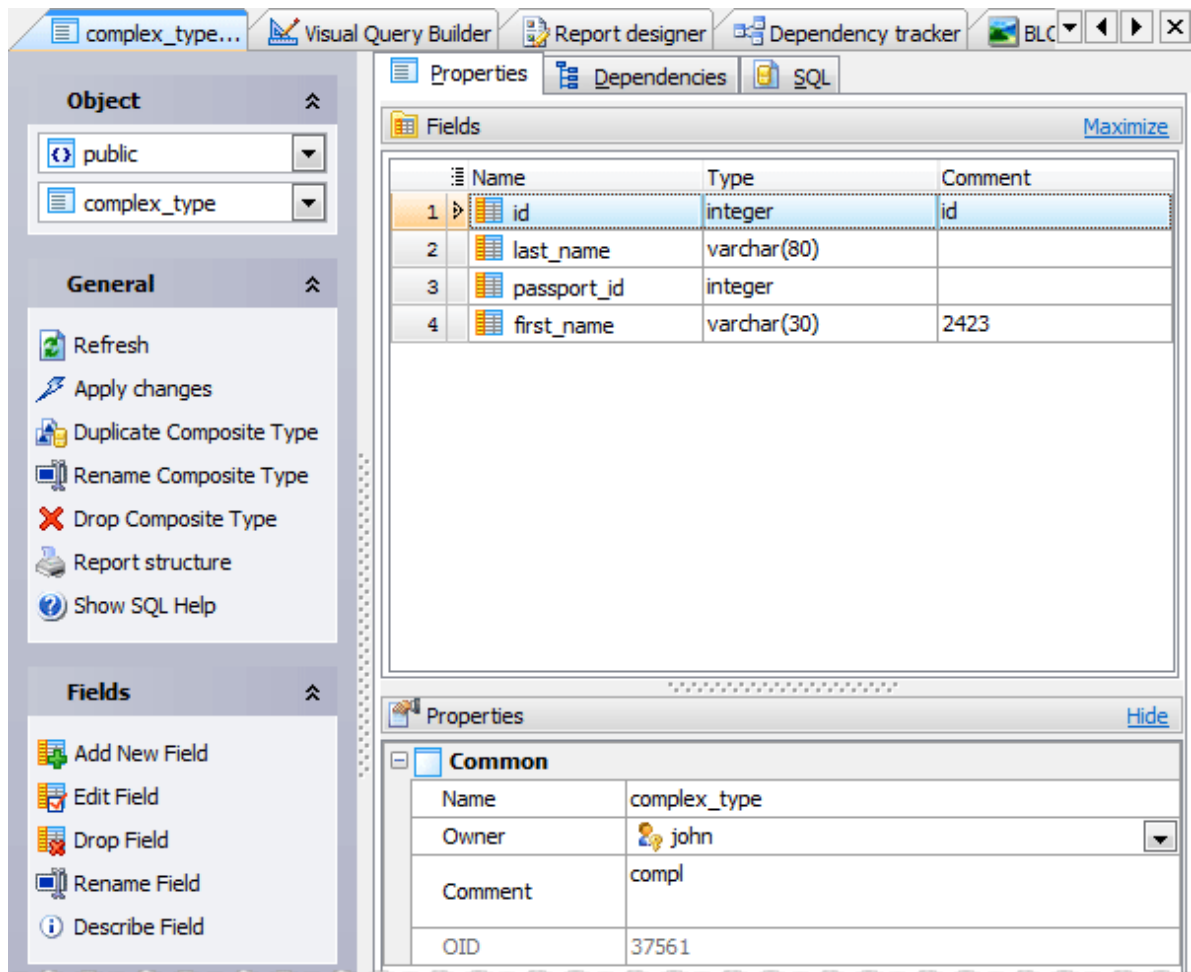
Click [Next](#) when you have added all the type options.

5.10.2 Type Editor (Composite and Enum Types)

[Composite Type Editor](#) and [Enum Type Editor](#) allow to work with existing types. The editors can be opened automatically after the types are created and also from the [Explorer Tree](#) and [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)⁴⁶. Below you will find a description of editor tabs that are unique for the current object.

[Composite and Enum type properties](#)¹⁵⁸




5.10.2.1 Composite and Enum type properties

The [Properties](#) tab allows you to change the type name, owner and comment. You can also find the OID there.

Composite Type contains a list of fields and Enum type - a list of values. The Composite Type Editor provides you with an ability to modify fields. Values of Enum types can not be edited.

Composite type field list

The [Fields](#) tab is intended for managing composite type fields. The tab popup menu allows you to create new, edit, rename or drop a selected field. You can also create a copy of the field with the corresponding item of popup menu. Fields are edited with [the corresponding editor](#) .

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.11 Range Types

Range types are data types representing a range of values of some element type (called the range's subtype). For instance, ranges of timestamp might be used to represent the ranges of time that a meeting room is reserved. In this case the data type is `tsrange` (short for "timestamp range"), and timestamp is the subtype. The subtype must have a total order so that it is well-defined whether element values are within, before, or after a range of values.

Range types are useful because they represent many element values in a single range value, and because concepts such as overlapping ranges can be expressed clearly. The use of time and date ranges for scheduling purposes is the clearest example; but price ranges, measurement ranges from an instrument, and so forth can also be useful.

■ How can I create a new range type?

New range types are created within [Create Range Type Wizard](#)^[160]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the Range Type icon in the [Create Database Object](#) dialog
- or
- select the [Range Types](#) list or any object from that list in the explorer tree;
 - select the [Create New Range Type...](#) item from the popup menu

To create a new range type with the same properties as one of the existing one has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing range type?

Range data types can be edited within [Range Type Editor](#)^[161]. In order to run the editor you should either

- select the [range type](#) for editing in the explorer tree (type the first letters of the type name for quick search);
- select the [Edit Range Type...](#) item from the popup menu

■ How can I drop a range type?

To drop a range data type:

- select the type to drop in the explorer tree;
- select the [Drop Range Type](#) item from the popup menu

and confirm dropping in the dialog window.

5.11.1 Create Range Type Wizard

[Create Range Type Wizard](#) guides you through the process of creating a new type.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)⁴³. Below you will find a description of wizard steps that are unique for the current object.

To create a range type, specify the [Name](#) of the new type and the [Subtype](#) that the range type will represent ranges of, select the type's [Owner](#) and enter an optional text to describe the type as [Comment](#).

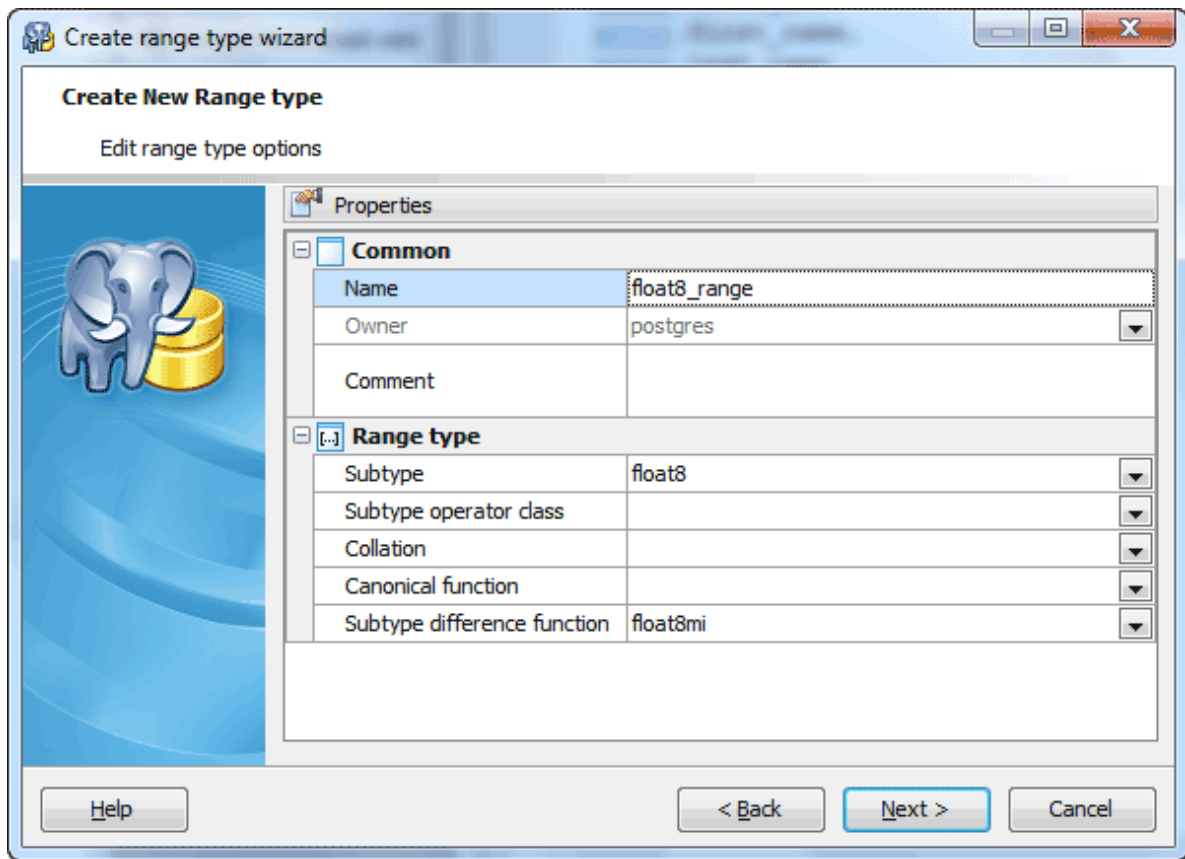
The range type's subtype can be any type with an associated b-tree operator class (to determine the ordering of values for the range type). Normally the subtype's default b-tree operator class is used to determine ordering.

To use a non-default operator class, specify its name with [Subtype operator class](#).

If the subtype is collatable, and you want to use a non-default collation in the range's ordering, specify the desired collation with the [Collation](#) option.

The optional [Canonical function](#) must take one argument of the range type being defined, and return a value of the same type. This is used to convert range values to a canonical form, when applicable. It must be defined before the range type can be declared.

The optional [Subtype difference function](#) function must take two values of the subtype type as argument, and return a double precision value representing the difference between the two given values. While this is optional, providing it allows much greater efficiency of GiST indexes on columns of the range type.



5.11.2 Range Type Editor

[Range Type Editor](#) is the basic PostgreSQL Maestro tool for working with existing range types. You can open the editor from the [Explorer Tree](#) or [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)⁴⁶⁾. Below you will find a description of editor tabs that are unique for the current object.

The editor allows to rename the range type and to browse type options. See the description of [range type creation](#)¹⁶⁰⁾ to find out what the options mean.

5.12 Operators

Each operator is "syntactic sugar" for a call to an underlying function that does the real work; so you must first create the underlying function before you can create the operator. After that, you can create and edit the new operator using the appropriate tools.

■ How can I create a new operator?

New operators are created within [Create Operator Wizard](#)^[163]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Operator](#) icon in the Create Database Object dialog
- or
- select the [Operators](#) list or any object from that list in the explorer tree;
 - select the [Create New Operator...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Operators](#) tab there;
 - press the **Insert** key or select the [Create New Operator...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new operator with the same properties as one of the existing operators has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing operator?

Operators can be edited within [Operator Editor](#)^[164]. In order to run the editor you should either

- select the operator for editing in the explorer tree (type the first letters of the operator name for quick search);
 - select the [Edit Operator ...](#) item from the popup menu
- or
- open the schema in [Schema Editor](#) and the [Operators](#) tab there;
 - select the operator to edit;
 - press the **Enter** key or select the [Edit Operator](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop an operator?

To drop an operator:

- select the operator to drop in the explorer tree;
 - select the [Drop Operator](#) item from the popup menu
- or

- open the schema in [Schema Editor](#) and the [Operators](#) tab there;
- select the operator to drop;
- press the **Delete** key or select the [Drop Operator](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window (here you can also specify whether to drop also the dependent objects or not).

5.12.1 Create Operator Wizard

[Create Operator Wizard](#) guides you through the process of creating a new schema operator.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Create operator wizard

Create New Operator

Edit operator options

Properties

Common

Name	<>
Owner	postgres
Comment	

Operator

Function	public.citext_ne
Left Operand Type	public.citext
Right Operand Type	public.citext
Commutator	public.<>
Negator	public.=
Restrict Function	pg_catalog.neqsel
Join Function	pg_catalog.neqjoinsel
Left Sort Operator	
Right Sort Operator	
Less Than Operator	
Greater Than Operator	
Supports Hash Joins	<input type="checkbox"/>
Supports Merge Joins	<input type="checkbox"/>

Buttons: Help, < Back, Next >, Cancel

Owner

Defines the owner of the new operator.

Comment

The box allows you to set optional text to describe the new operator.

Function

Specify the function used to implement this operator.

Left operand type

Set the data type of the operator left operand, if any. This option can be omitted for a left-unary operator.

Right operand type

Set the data type of the operator right operand, if any. This option can be omitted for a right-unary operator.

Commutator

Specify the commutator of the operator.

Negator

Set the negator of the operator.

Restrict Function

Specify the restriction selectivity estimator function for the operator.

Join Function

Define the join selectivity estimator function for the operator.

Left Sort Operator

If the operator can support a merge join, the less-than operator that sorts the left-hand data type of the operator.

Right Sort Operator

If the operator can support a merge join, the less-than operator that sorts the right-hand data type of the operator.

Less Than Operator

If the operator can support a merge join, the less-than operator that compares the input data types of the operator.

Greater Than Operator

If the operator can support a merge join, the greater-than operator that compares the input data types of the operator.

The [Supports Hash Joins](#) and [Supports Merge Joins](#) checkboxes indicate that this operator can support a hash join and a merge join respectively.

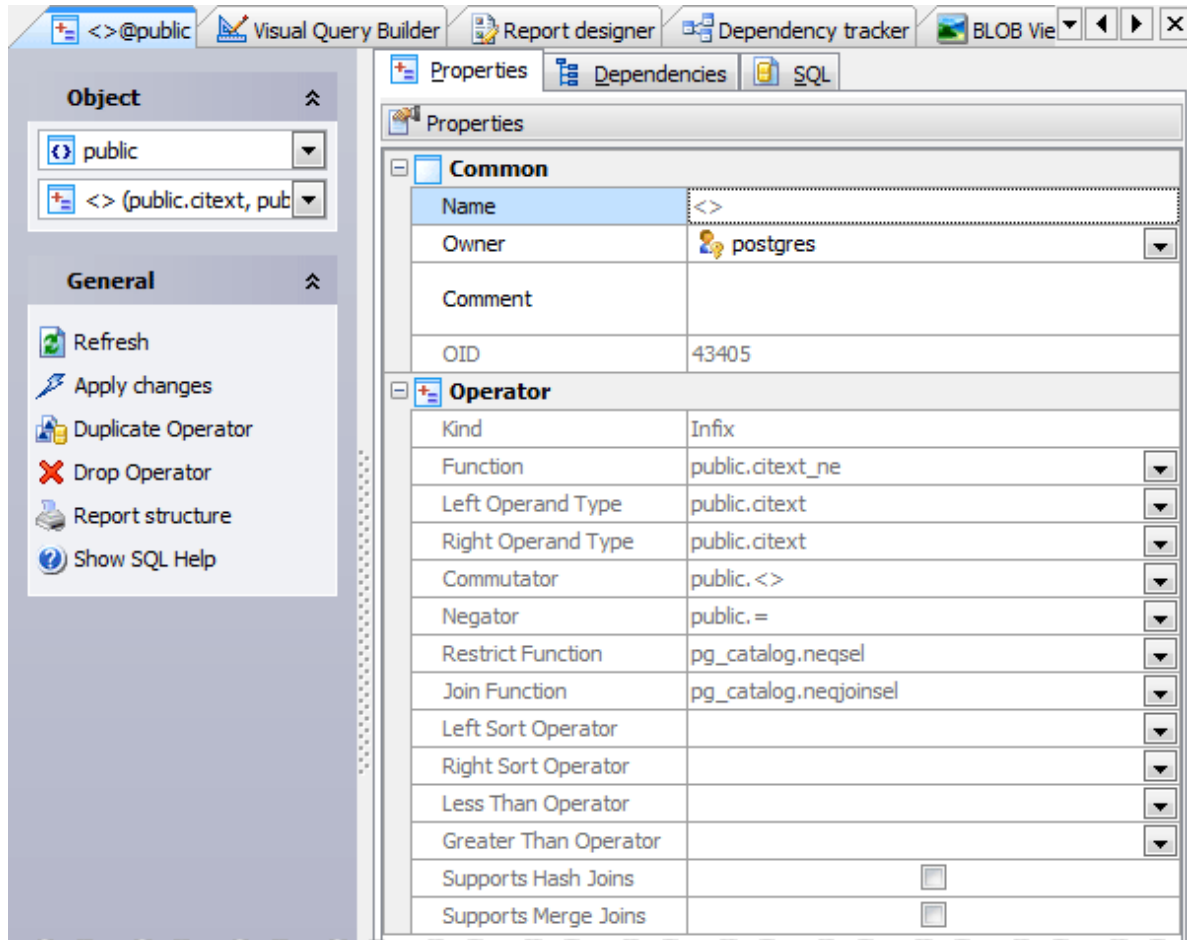
5.12.2 Operator Editor

[Operator Editor](#) is the basic PostgreSQL Maestro tool for working with existing operators. It can be open automatically after the operator is created and also from the [Explorer Tree](#) or [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate](#)

[topic](#)⁴⁶. Below you will find a description of editor tabs that are unique for the current object.

[Operator properties](#)¹⁶⁵



5.12.2.1 Editing operator properties

Operator Editor provides you with an ability to edit operator properties. The **Properties** tab allows you to change the operator name, the operator owner and the comment for the operator. You can also find the OID there.

Name

Here you can change the operator name.

Owner

Here you can change the operator owner. By default, only the owner of an object can perform various operations with the object. In order to allow other users to operate it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Comment

This field stores a comment to the operator.

OID

The field stores the operator OID (object identifier).

Function

The function used to implement this operator is supplied in this box.

Left operand type

Edit the data type of the operator left operand, if any. This option can be omitted for a left-unary operator.

Right operand type

Edit the data type of the operator right operand, if any. This option can be omitted for a right-unary operator.

Commutator

Specify the commutator of the operator.

Negator

Set the negator of the operator.

Restrict Function

Specify the restriction selectivity estimator function for the operator.

Join Function

The join selectivity estimator function for this operator.

Left Sort Operator

If the operator can support a merge join, the less-than operator that sorts the left-hand data type of this operator.

Right Sort Operator

If the operator can support a merge join, the less-than operator that sorts the right-hand data type of this operator.

Less Than Operator

If the operator can support a merge join, the less-than operator that compares the input data types of this operator.

Greater Than Operator

If the operator can support a merge join, the greater-than operator that compares the input data types of this operator.

The [Supports Hash Joins](#) and [Supports Merge Joins](#) checkboxes are meant to indicate that this operator can support a hash join and a merge join respectively.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.13 Collations

The [collation feature](#) allows specifying the sort order and character classification behavior of data per-column, or even per-operation. PostgreSQL allows to define a new collation using the specified operating system locale settings, or by copying an existing collation. To be able to create a collation, you must have CREATE privilege on the destination schema.

To [create a new](#)^[46] collation, set the specified operating system locale for the LC_COLLATE locale category, to be used to control the sort order. But it is rarely necessary in practice to have an LC_CTYPE locale category setting that is different from LC_COLLATE, so it is more convenient to collect these under one concept than to create another infrastructure for setting LC_CTYPE per expression. A collation is tied to a character set encoding. The same collation name may exist for different encodings.

To use a collation, specify it at the [Field Editor](#)^[82] or on a [domain creation](#)^[134].

5.14 Languages

A PostgreSQL user can register a new procedural [language](#) with a PostgreSQL database. Subsequently, functions and trigger procedures can be defined in this new language. The user must have the PostgreSQL superuser privilege to register a new language.

■ How can I create a new language?

New languages are created within [Create Language Wizard](#)¹⁶⁹. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Language](#) icon in the [Create Database Object](#) dialog
- or
- select the [Languages](#) list or any object from that list in the explorer tree;
 - select the [Create New Language...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Languages](#) tab there;
 - press the **Insert** key or select the [Create New Language...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new language with the same properties as one of the existing languages has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing language?

Languages can be edited within [Language Editor](#)¹⁷¹. In order to run the editor you should either

- select the language for editing in the explorer tree (type the first letters of the language name for quick search);
 - select the [Edit Language ...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Languages](#) tab there;
 - select the language to edit;
 - press the **Enter** key or select the [Edit Language](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the language using the [Rename Language](#) dialog. To open the dialog you should either

- select the language to rename in the explorer tree;
- select the [Rename Language](#) item from the popup menu

or

- open the database in [Database Editor](#) and the [Languages](#) tab there;
- select the language to rename;
- select the [Rename Language](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a language?

To drop a language:

- select the language to drop in the explorer tree;
- select the [Drop Language](#) item from the popup menu

or

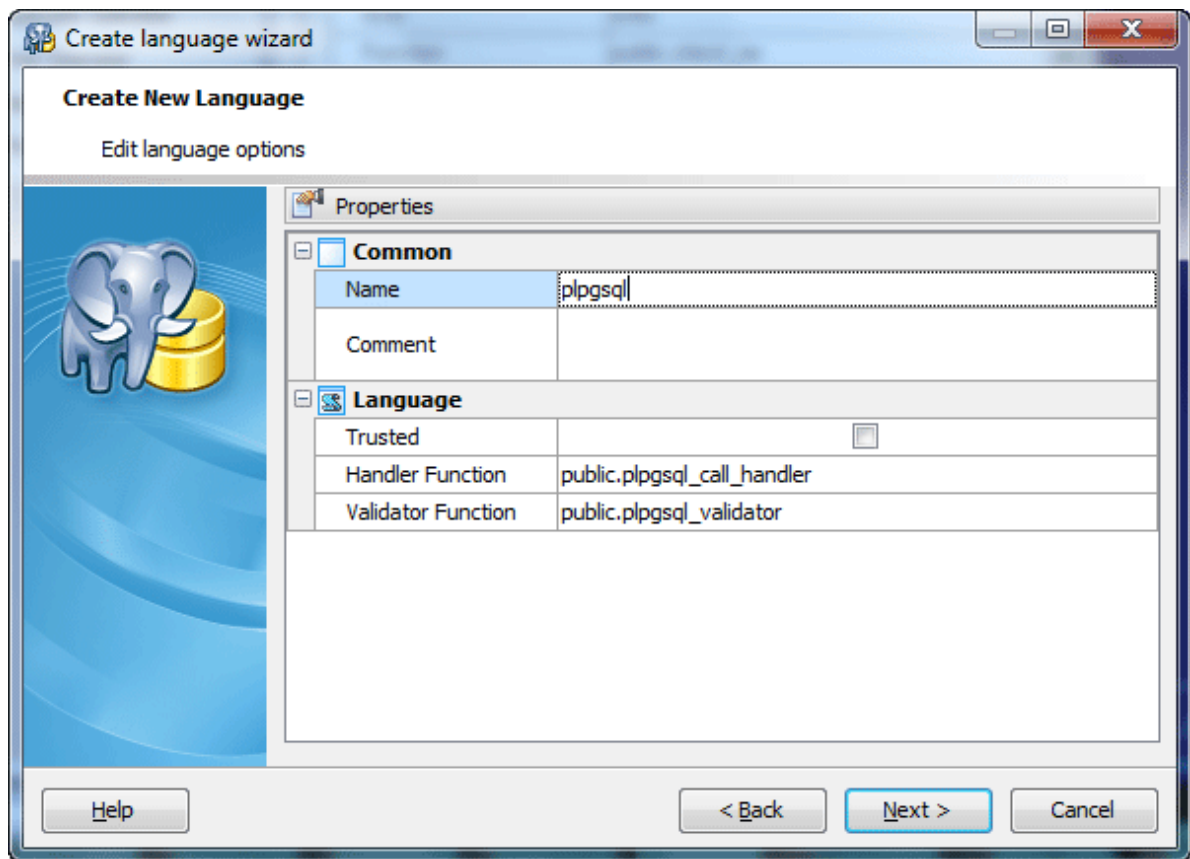
- open the database in [Database Editor](#) and the [Languages](#) tab there;
- select the language to drop;
- press the **Delete** key or select the [Drop Language](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.14.1 Create Language Wizard

[Create Language Wizard](#) guides you through the process of creating a new language. See [How To Create Language](#)^[168] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.



Comment

The box allows you to set optional text to describe the new language.

Trusted specifies that the call handler for the language is safe, that is, it does not offer an unprivileged user any functionality to bypass access restrictions. If this key word is omitted when registering the language, only users with the PostgreSQL superuser privilege can use this language to create new functions.

Handler Function

Specify the name of a previously registered function that will be called to execute the procedural language functions. The call handler for a procedural language must be written in a compiled language such as C with version 1 call convention and registered with PostgreSQL as a function taking no arguments and returning the language_handler type, a placeholder type that is simply used to identify the function as a call handler.

Validator Function

Set the name of a previously registered function that will be called when a new function in the language is created, to validate the new function. If no validator function is specified, then a new function will not be checked when it is created. The validator function must take one argument of type oid, which will be the OID of the to-be-created function, and will typically return void.

A validator function would typically inspect the function body for syntactical correctness, but it can also look at other properties of the function, for example if the language cannot handle certain argument types. To signal an error, the validator function should use the *ereport()* function. The return value of the function is ignored.

5.14.2 Language Editor

[Language Editor](#) is the basic PostgreSQL Maestro tool for working with existing languages. It can be opened automatically after the language is created and also from the [Explorer Tree](#) or [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

[Language properties](#)^[17]

See also: [Create Language Wizard](#)^[169]

5.14.2.1 Editing language properties

[Language Editor](#) provides you with an ability to edit language properties. The [Properties](#) tab allows you to change the language name, the language handler and validator functions and the comment for the language. You can also find the OID there.

Name

Here you can change the language name.

Comment

This field stores a comment to the language.

OID

The field contains the language OID (object identifier).

[Trusted](#) specifies that the call handler for the language is safe, that is, it does not offer an unprivileged user any functionality to by pass access restrictions.

Handler Function

Specify the name of a previously registered function that will be called to execute the procedural language functions.

Validator Function

Edit the name of a previously registered function that will be called when a new function in the language is created, to validate the new function.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.15 Casts

A [cast](#) specifies how a conversion between two data types is to be performed. To be able to create a cast, you must own the source or the target data type. To create a binary-compatible cast, you must be a superuser. (This restriction is made because an erroneous binary-compatible cast conversion can easily crash the server.)

■ How can I create a new cast?

New casts are created within [Create Cast Wizard](#)^[173]. In order to run the wizard you should either

- select the [Object | Create Database Object...](#) main menu item;
 - select the [Cast](#) icon in the [Create Database Object](#) dialog
- or
- select the [Casts](#) list or any object from that list in the explorer tree;
 - select the [Create New Cast...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Casts](#) tab there;
 - press the **Insert** key or select the [Create New Cast...](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

To create a new cast with the same properties as one of the existing casts has:

- select the [Object | Duplicate Database Object...](#) main menu item;
- follow the instructions of [Duplicate Object Wizard](#).

■ How can I edit an existing cast?

Casts can be edited within [Cast Editor](#)^[175]. In order to run the editor you should either

- select the cast for editing in the explorer tree (type the first letters of the cast name for quick search);
 - select the [Edit Cast ...](#) item from the popup menu
- or
- open the database in [Database Editor](#) and the [Casts](#) tab there;
 - select the cast to edit;
 - press the **Enter** key or select the [Edit Cast](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

You can change the name of the cast using the [Rename Cast](#) dialog. To open the dialog you should either

- select the cast to rename in the explorer tree;
 - select the [Rename Cast](#) item from the popup menu
- or

- open the database in [Database Editor](#) and the [Casts](#) tab there;
- select the cast to rename;
- select the [Rename Cast](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a cast?

To drop a cast:

- select the cast to drop in the explorer tree;
- select the [Drop Cast](#) item from the popup menu

or

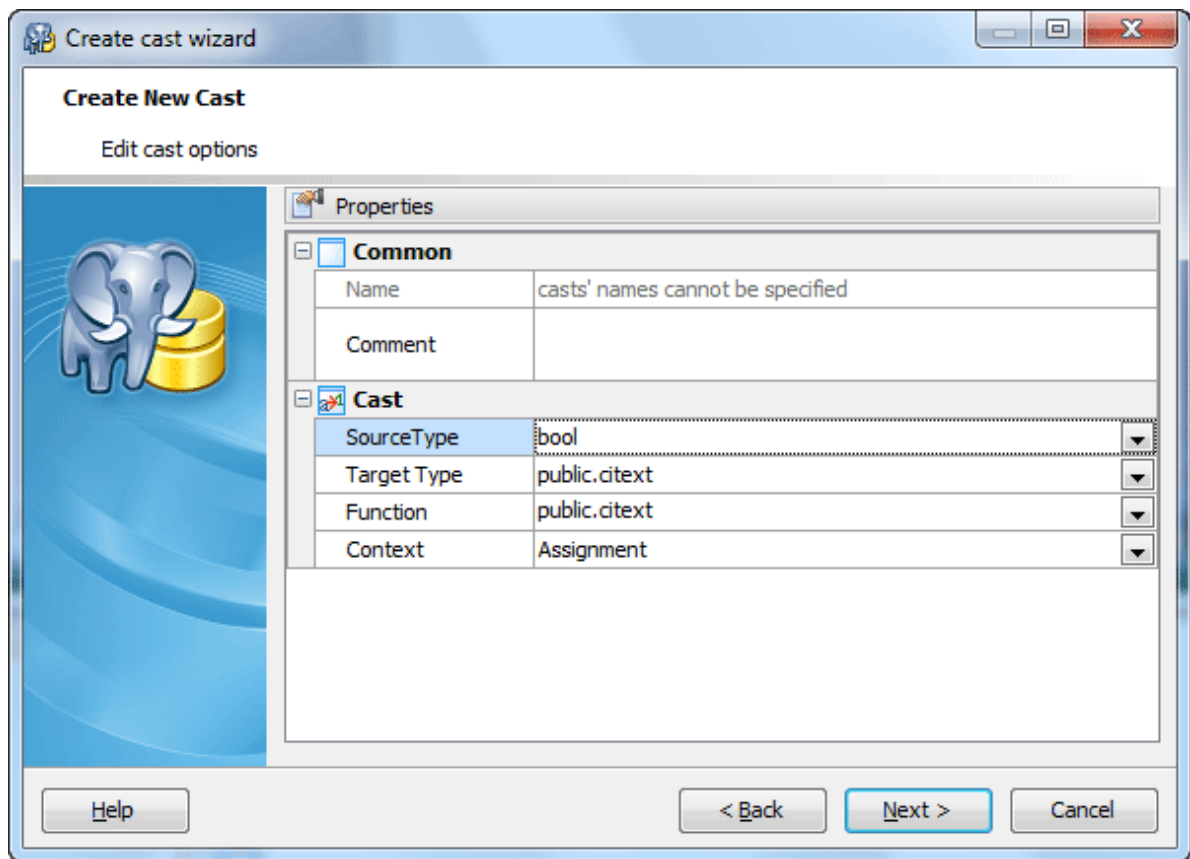
- open the database in [Database Editor](#) and the [Casts](#) tab there;
- select the cast to drop;
- press the **Delete** key or select the [Drop Cast](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.15.1 Create Cast Wizard

[Create Cast Wizard](#) guides you through the process of creating a new cast language. See [How To Create a New Cast](#)^[172] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.



Comment

The box allows you to set optional text to describe the new cast.

Source Type

Specify the name of the source data type of the cast.

Target Type

Set the name of the target data type of the cast.

Function

The function used to perform the cast. The function name may be schema-qualified. If it is not, the function will be looked up in the schema search path. The function result data type must match the target type of the cast. Specify no function to indicate that no function is required to perform the cast.

Context (*Assignment, Implicit, Explicit*)

Assignment indicates that the cast may be invoked implicitly in assignment contexts. *Implicit* indicates that the cast may be invoked implicitly in any context. By default, a cast can be invoked only by an *Explicit* cast.

Note: Remember that if you want to be able to convert types both ways you need to declare casts both ways explicitly.

5.15.2 Cast Editor

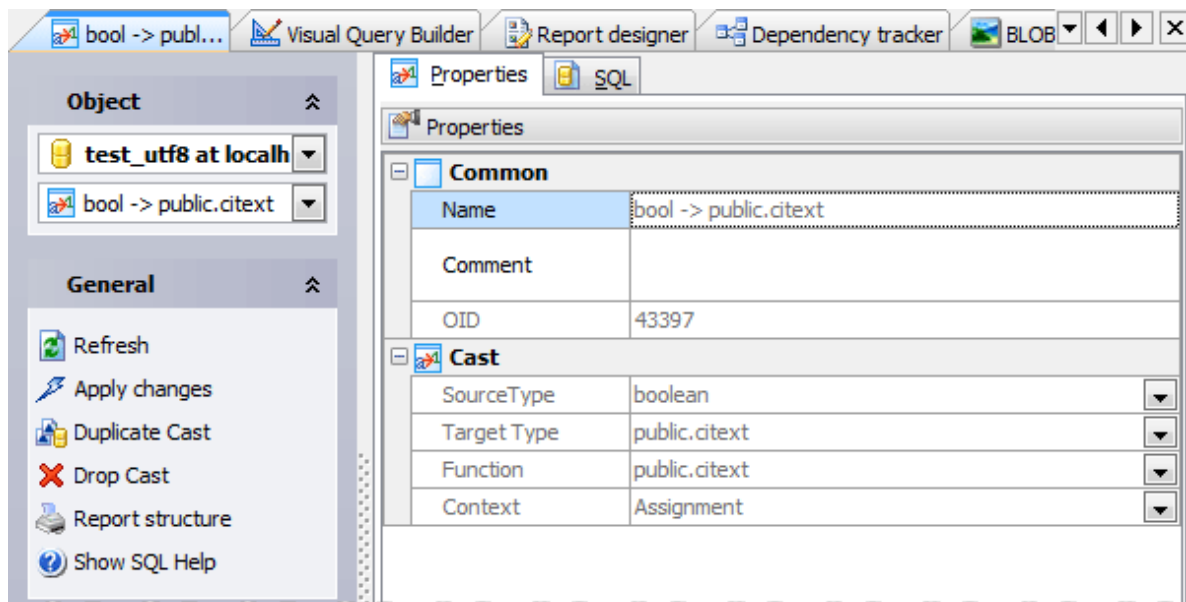
Cast Editor is the basic PostgreSQL Maestro tool for working with existing casts. It can be opened automatically after the cast is created and is available on editing the cast (see [How to edit cast](#)^[172] for details).

You can open a cast in **Cast Editor** from the **Explorer Tree** or **Object Manager**.

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

[Cast properties](#)^[175]

See also: [Create Cast Wizard](#)^[173].



5.15.2.1 Editing cast properties

Cast Editor provides you with an ability to edit cast properties. The **Properties** tab allows you to change the cast name, the cast source and target types and the comment for the cast. You can also find the OID there.

Name

Here you can change the cast name.

Comment

This field stores a comment to the cast.

OID

The field stores the cast OID (object identifier).

Source Type

Edit the name of the source data type of the cast.

Target Type

Set the name of the target data type of the cast.

Function

The function used to perform the cast.

Context (*Assignment, Implicit, Explicit*)

Edit the way the cast may be invoked.

To apply the changes, select the **Apply Changes** item in the **Navigation bar** or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the **Object Properties** item of the popup menu of the selected object from the explorer tree.

5.16 Database Variables

There are a lot of configuration parameters that affect the behavior of the database system. All parameter (server **variables**) names are case-insensitive. Every such variable takes a value of one of four types: *boolean*, *integer*, *floating point* or *string*. For a PostgreSQL database you can specify values of the variables (different from values of the server variables).

■ How can I set a database variable value?

To set a new database variable value use [Database_variable properties](#)^[178] window. In order to open the window you should either

- select the **Object | Create Database Object...** main menu item;
 - select the **Variable** icon in the **Create Database Object** dialog
- or
- select the **Variables** list or any object from that list in the explorer tree;
 - select the **Add New Variable...** item from the popup menu
- or
- open the database in **Database Editor** and the **Variables** tab there;
 - press the **Insert** key or select the **Add New Variable...** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

■ How can I edit a database variable value?

Variables can be edited within [Database_Variable_Properties](#)^[178] window. In order to run the editor you should either

- select the variable for editing in the explorer tree (type the first letters of the variable name for quick search);
 - select the **Edit Variable ...** item from the popup menu
- or
- open the database in **Database Editor** and the **Variables** tab there;
 - select the variable to edit;
 - press the **Enter** key or select the **Edit Variable** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**).

■ How can I reset a database variable?

To reset a database variable:

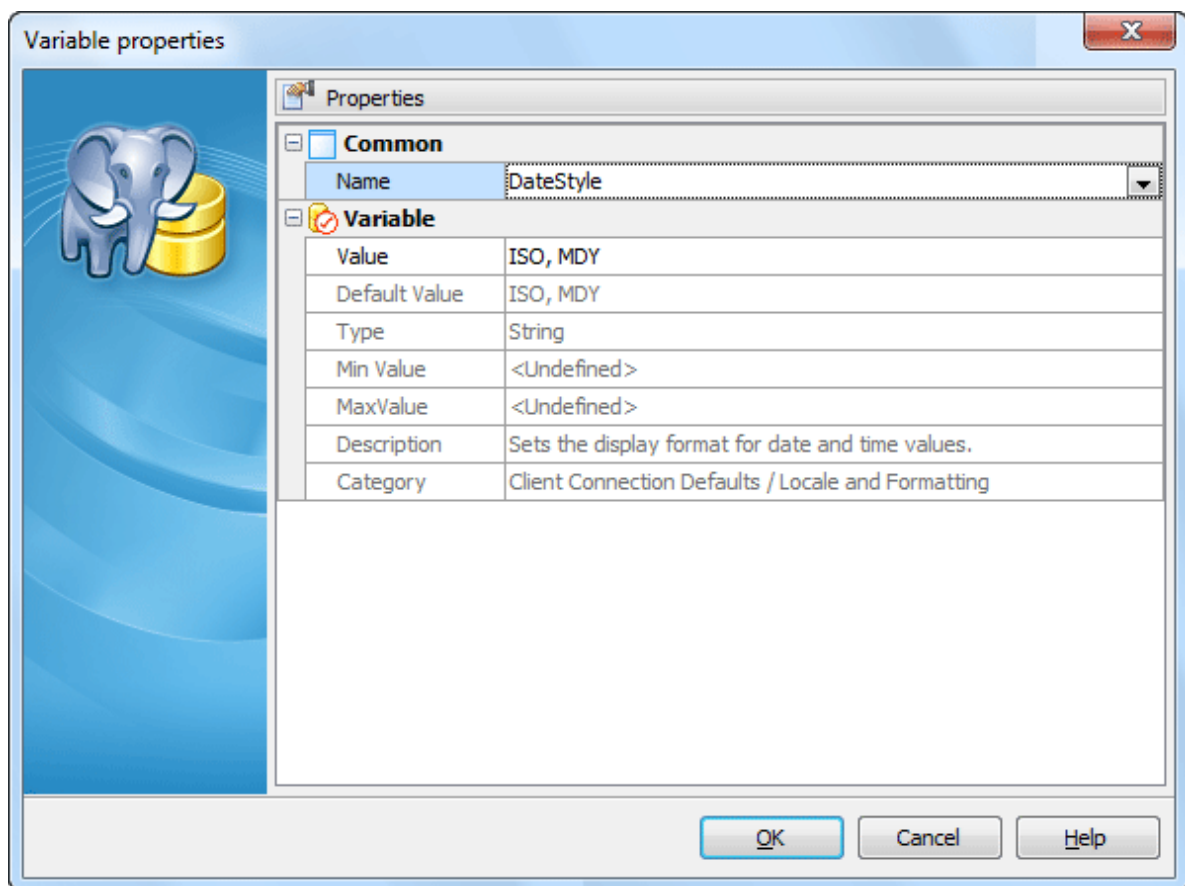
- select the variable to drop in the explorer tree;
 - select the **Drop Variable** item from the popup menu
- or
- open the database in **Database Editor** and the **Variables** tab there;

- select the variable to drop;
- press the **Delete** key or select the [Drop Variable](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

5.16.1 Database variable properties

[Database Variable Editor](#) is the basic PostgreSQL Maestro tool for creation a new database variable or working with existing one. It allows to specify the value of the variable. Several options can be specified for the database variable that govern the expected behavior of the database system.



The [Properties](#) tab allows you to view variable parameters and to change the database variable value.

[Name](#)

Displays the database variable name.

The [Value](#) represents the current value of the variable.

[Default Value](#)

The [Default](#) clause specifies a default value for columns of the variable data type.

The [Type](#) represents the underlying data type for the variable.

The [Max Value](#) and the [Min Value](#) contains the maximum and the minimum value for the variable.

[Description](#) contains a comment to the variable.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

5.17 Extensions

The Extensions node of the Explorer tree shows extensions loaded into the current database. Loading an extension essentially amounts to running the extension's script file. The script will typically create new SQL objects such as functions, data types, operators and index support methods. The process of extension creating additionally records the identities of all the created objects, so that they can be dropped when the extension will be dropped. The extensions currently available for loading can be identified from the *pg_available_extensions* or *pg_available_extension_versions* system views.

To load a new extension into the current database,

- select the [Extensions](#) list or any object from that list in the explorer tree;
- use the [Create New Extension ...](#) item from the popup menu or press **Ins**;
- specify options of the extension to be loaded in the [Create Extension Wizard](#).

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)⁴³. Below you will find a description of wizard steps that are unique for the current object.

Name

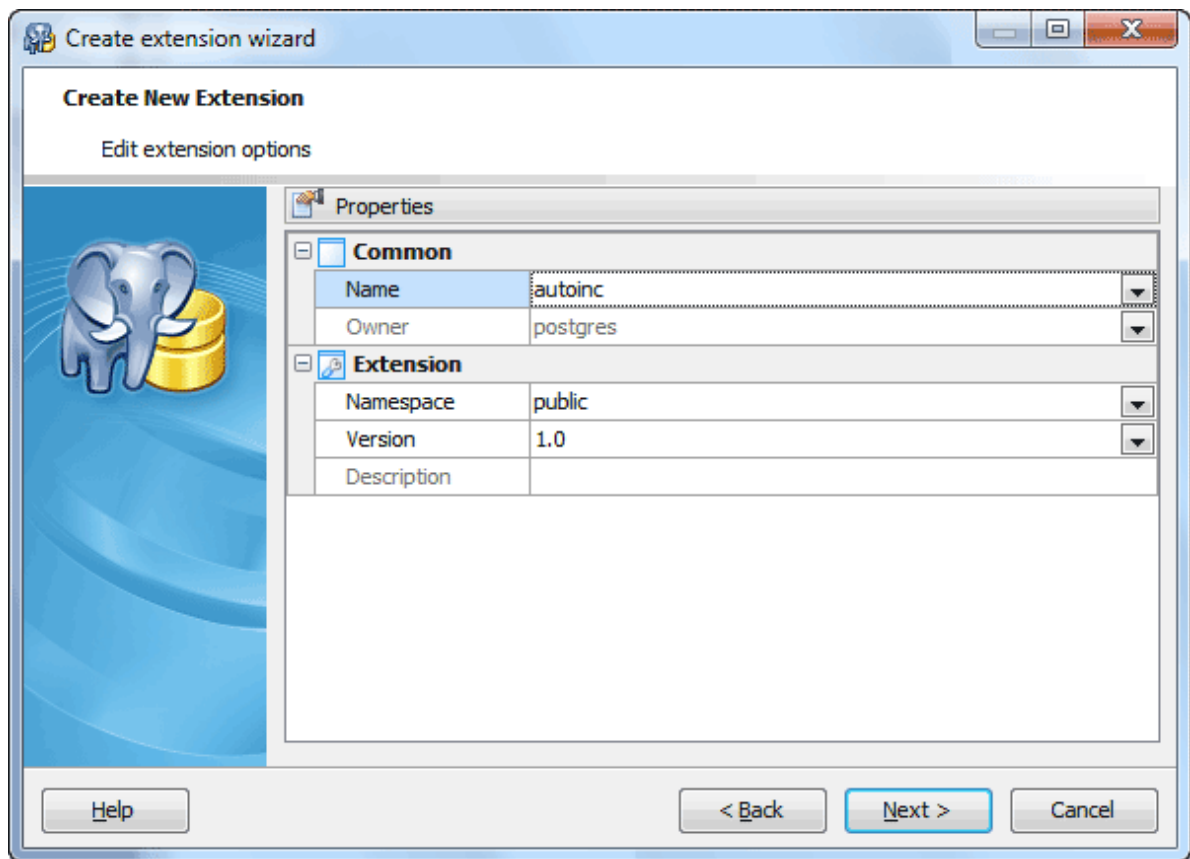
The name of the extension to be installed. PostgreSQL will create the extension using details from the file `SHAREDIR/extension/extension_name.control`.

Namespace

The name of the schema in which to install the extension's objects, given that the extension allows its contents to be relocated.

Version

The version of the extension to install. This can be written as either an identifier or a string literal. The default version is whatever is specified in the extension's control file.



To see details of loaded extensions, use the corresponding editor.

5.18 Foreign Tables

PostgreSQL allows to access data stored on an external PostgreSQL server using regular SQL queries. Such data is referred as foreign data. To access foreign data, you need to create a foreign [server object](#)^[185], which defines how to connect to a particular external data source. Then you need to create one or more **foreign tables**, which define the structure of the remote data. A foreign table can be used in queries just like a normal table, but a foreign table has no storage in the PostgreSQL server.

To get data of table stored on another PostgreSQL server available for querying in the current database profile, in other words to add a new foreign table,

- select the [Foreign tables](#) list or any object from that list in the explorer tree;
- use the [Create New Foreign Table ...](#) item from the popup menu or press **Ins**;
- specify options of this table in the [Create Foreign Table Wizard](#).

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

To change an existing foreign table, open the corresponding editor, modify necessary properties and click [Apply changes](#).

Foreign table properties

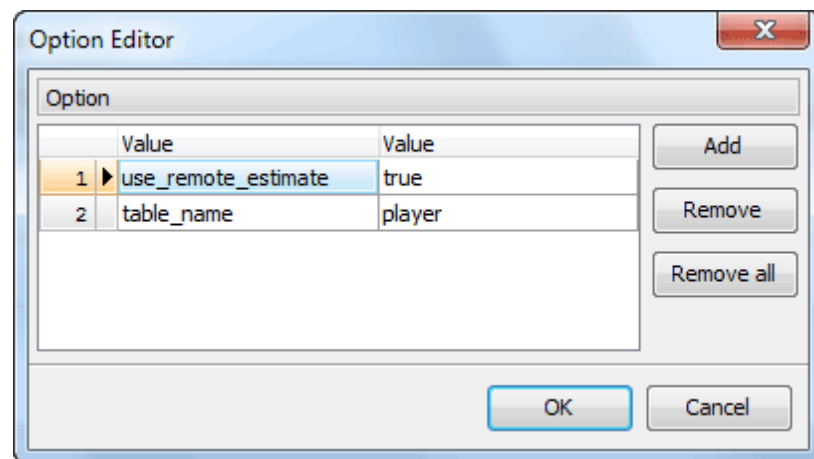
The [Name](#) of the foreign table must be distinct from the name of any other foreign table, table, sequence, index, or view in the same schema.

The screenshot shows the 'Properties' dialog box for a 'Foreign table'. It is divided into two sections: 'Common' and 'Foreign table'.

Common	
Name	nba_player
Owner	postgres
Comment	The Player table stored in nba database on test_server.

Foreign table	
Options	table_name=player,use_remote_estimate=true
Server name	foreign_server_nba

The [Options](#) dialog allows to set options to be associated with the new foreign table or one of its columns. The allowed option names and values are specific to each foreign data wrapper and are validated using the foreign-data wrapper's validator function. Duplicate option names are not allowed (although it's OK for a table option and a column option to have the same name).



Server name shows the name of an existing foreign server that is used for the foreign table.

Foreign table fields list contains fields of table stored in external database which data are available in the current one.

Properties Data Dependencies SQL					
Fields Maximize					
	Name	Type	Not Null	Default	Comment
1	id	integer	<input checked="" type="checkbox"/>		
2	last_name	varchar(30)	<input type="checkbox"/>		
3	first_name	varchar(40)	<input type="checkbox"/>		
4	career_start_year	integer	<input type="checkbox"/>		
5	photo_medium_quality	bytea	<input type="checkbox"/>		
6	birthday	timestamp	<input type="checkbox"/>		




The **Name** of a foreign table column usually must be the same as the underlying remote table column name. The **data type** of the column can include array specifiers. The column is not allowed to contain null values are marked as **Not Null**. The **Default** expression will be used in any insert operation that does not specify a value for the column. If there is no default for a column, then the default is null.

The **Data** tab displays the foreign table data as a grid or as info cards (see [Data View](#)^[229] for details). To edit/add a table record, use **Data Input Form** or type the new data directly in the grid (card). To export/import/get SQL dump of the foreign table data, invoke corresponding modules from the grid's popup menu. To view and edit the content of BLOB columns, run [BLOB Editor](#)^[240].

Properties Data Dependencies SQL

Table

Drag a column header here to group by that column

	id	last_name	first_name	career	photo_medium_quality	birthday
281	407	Webster	Martell	2005		04.12.1986
282	418	Williams	Deron	2005		26.06.1984
283	419	Williams	Louis	2005		27.10.1986

Records fetched: 436/436

5.19 Foreign Servers

A [Foreign Server](#) typically encapsulates connection information that a foreign-data wrapper uses to access an external data resource. Additional user-specific connection information may be specified by means of [user mappings](#)^[186]. PostgreSQL Maestro supports only the [postgres_fdw](#) foreign data wrapper. Creating a server requires USAGE privilege on the foreign-data wrapper being used.

To add a new foreign server,

- select the [Foreign servers](#) list or any object from that list in the explorer tree;
- use the [Create New Foreign Server ...](#) item from the popup menu or press **Ins**;
- specify options of this table in the [Create Foreign Server Wizard](#).

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

To change an existing foreign server, open the corresponding editor, modify necessary properties and click [Apply changes](#).

Name

The server name must be unique within the database.

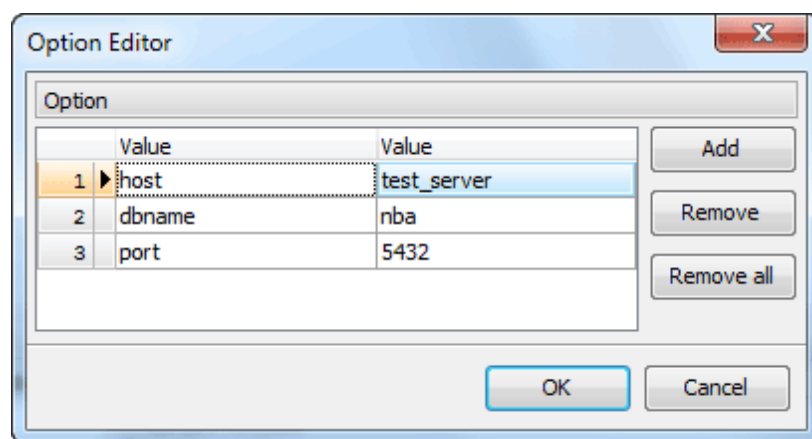
Owner

The owner of a foreign server can create user mappings for that server for any user. Also, a user can create a user mapping for his own user name if USAGE privilege on the server has been granted to the user.

Options

This clause specifies the options for the server. The options typically define the connection details of the server, but the actual names and values are dependent on the server's foreign-data wrapper. *Postgres_fdw* foreign data wrapper supports [libpq options](#), except the following ones:

- *user* and *password* (specify these for a user mapping, instead)
- *client_encoding* (this is automatically set from the local server encoding)
- *fallback_application_name* (always set to *postgres_fdw*)



Type

Optional server type, potentially useful to foreign-data wrappers.

Version

Optional server version, potentially useful to foreign-data wrappers.

5.19.1 User Mappings

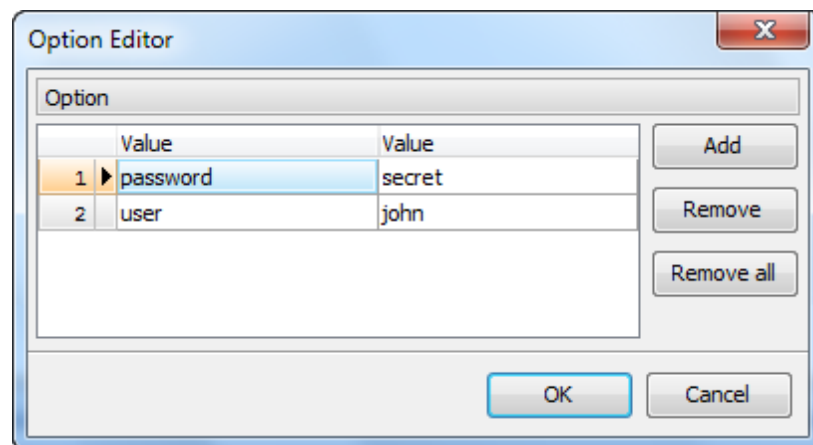
A user mapping typically encapsulates connection information that a foreign-data wrapper uses together with the information encapsulated by a foreign server to access an external data resource. The owner of a foreign server can create user mappings for that server for any user. Also, a user can create a user mapping for his own user name if USAGE privilege on the server has been granted to the user.

Name

The name of an existing user that is mapped to foreign server. CURRENT_USER and USER match the name of the current user. When PUBLIC is specified, a so-called public mapping is created that is used when no user-specific mapping is applicable.

Options

This clause specifies the options of the user mapping. The options typically define the actual user name and password of the mapping. Option names must be unique. The allowed option names and values are specific to the server's foreign-data wrapper. Note that only superusers may connect to foreign servers without password authentication, so always specify the password option for user mappings belonging to non-superusers. The screenshot below shows an example of user mapping options.



6 Server Objects

With PostgreSQL Maestro you can accomplish all the popular actions of server administration. Note that before working with server objects you should connect to any server database first (see [Database Management](#)^[24]).

The following list contains the most common server objects supported by PostgreSQL Maestro.

- [Databases](#)^[190];
- [User](#)^[194];
- [Roles](#)^[200] and [Groups](#)^[198];
- [Tablespaces](#)^[208];
- [Processes](#)^[307];
- [Server Variables](#)^[192].

Below you can find some common ways of server object management.

■ Creating of a new server object

New server objects are created within the appropriate [Create Object Wizard](#). In order to run the wizard you should either

- select the corresponding object list (such as [Users](#)) or any object from that list and then use the [Create New...](#) item from the popup menu

or

- open the server in [Server Editor](#)^[189] and the necessary objects' tab there and press **Insert** or select the [Create New...](#) item from the popup menu (Alternatively, use the corresponding link of the [Navigation Bar](#)).

■ Editing of an existing server object

Server objects are edited within the corresponding [Object Editor](#). In order to open the editor you should either

- select the server object for editing in the explorer tree (type the first letters of the object name for quick search);
- select the [Edit Object](#) item from the popup menu

or

- open the server in [Server Editor](#) and the corresponding objects' tab there;
- select the server object to edit;
- press the **Enter** key or select the [Edit Object](#) item from the popup menu (alternatively, you can use the corresponding link of the [Navigation Bar](#)).

■ Dropping of a server object

To drop the existing server object:

- select the server object to drop in the explorer tree;
- select the [Drop Object](#) item from the popup menu

or

- open the server in [Server Editor](#) and the appropriate objects' tab there;
- select the server object to drop;
- press the **Delete** key or select the [Drop Object](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

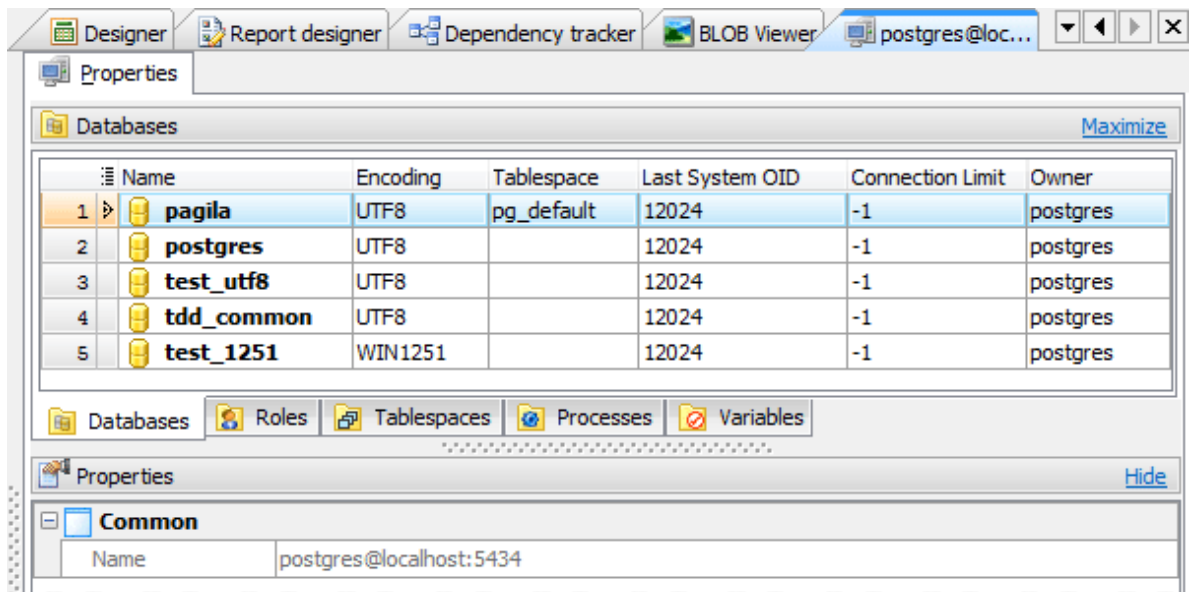
6.1 Server Editor

Server Editor allows you to look through all the server objects. Only connected databases are available for examination.

To run [Server Editor](#), select the essential server from the [Explorer Tree](#) and press **Enter**, or use [popup menu](#).

Server Editor consists of several tabs according different kinds of server-scope objects. Every tab is intended for managing server objects (e.g. *databases*, *users*, *groups*, *tablespaces*, *variables*, etc.). Any object can be opened in its editor. The popup menu allows you to create new, edit or drop the selected server object. Using the popup menu you can also create a copy of the object.

You can operate on several objects at a time. For this you have to select database objects with the **Shift** or the **Ctrl** key pressed. After the group of objects is selected, you can operate on it, e.g. *delete several objects* at once, as it were a single object.



6.2 Databases

PostgreSQL Maestro allows you to manipulate databases by means of database profiles. Profile contains database connection settings and a set of options to automatize common manipulations with databases (a possibility to connect to the database at PostgreSQL Maestro startup, login prompt before connection, etc.). To start working with databases in PostgreSQL Maestro, you should create database profile(s) first.

Use the following links for details:

■ **How can I create a new database?**

Use for this purpose [Create Database Wizard](#)^[37]. In order to run the wizard you should either

- select the [Database | Create New Database...](#) main menu item
- or
- use the [Create New Database...](#) item of the popup menu.

Using [Create Database Wizard](#) set the [Create profile after creating the database](#) option to create a new profile and open the [Database Profile Properties](#) dialog after the database is created.

■ **How can I change attributes of an existing database?**

To edit a database:

- select the database to edit in the explorer tree;
- edit database properties within the appropriate tabs of [Database Editor](#)^[40].

■ **How can I drop an existing database?**

In order to drop a database you should first select the database to drop in the explorer tree and establish connection (if you are not connected to the database yet), then either

- select the [Database | Drop Database](#) main menu item
- or
- use the [Drop Database](#) item of the popup menu

and confirm dropping in the dialog window to complete the operation.

■ **How can I create new database profiles?**

In PostgreSQL Maestro database profiles are created within [Create Database Profiles Wizard](#)^[26]. In order to run the wizard you should either

- select the [Database | Create Database Profiles...](#) main menu item
- or
- use the [Create Database Profiles...](#) item of the popup menu.

Using [Create Database Profiles Wizard](#) set the necessary connection and authorization options and click the [Ready](#) button to complete the operation.

How can I edit existing database profile options?

Database connection properties and profile options are edited within the [Database Profile Properties](#)^[30] dialog window. In order to open the dialog for the selected database profile you should either

- select the [Database | Edit Database Profile...](#) main menu item
- or
- use the [Edit Database Profile...](#) item of the popup menu.

How can I remove database profiles?

In order to remove a database profile you should first select the database profile in the explorer tree, then either select the [Database | Remove Database Profile](#) main menu item, or use the [Remove Database Profile](#) item of the popup menu and confirm removing profile in the dialog window to complete the operation.

How can I connect to a database?

In order to connect to a database you should first select the database in the explorer tree, then either

- select the [Database | Connect to Database](#) main menu item
- or
- use the [Connect to Database](#) item of the popup menu.

How can I disconnect from a database?

In order to disconnect from a database you should first select the database in the explorer tree, then either

- select the [Database | Disconnect from Database](#) main menu item
- or
- use the [Disconnect from Database](#) item of the popup menu.

6.3 Server Variables

There are a lot of configuration parameters that affect the behavior of the database system. All parameter (server variables) names are case-insensitive. These variables can be used to select default connection parameter values, to avoid hard-coding database connection information into simple client applications, for example. They also can be used to specify default behavior for PostgreSQL session. Every variable of this type takes a value of one of four types: *Boolean*, *Integer*, *Floating Point* or *String*.

Server variables are edited within [Server Variable Editor](#)¹⁹². In order to open the editor you should either

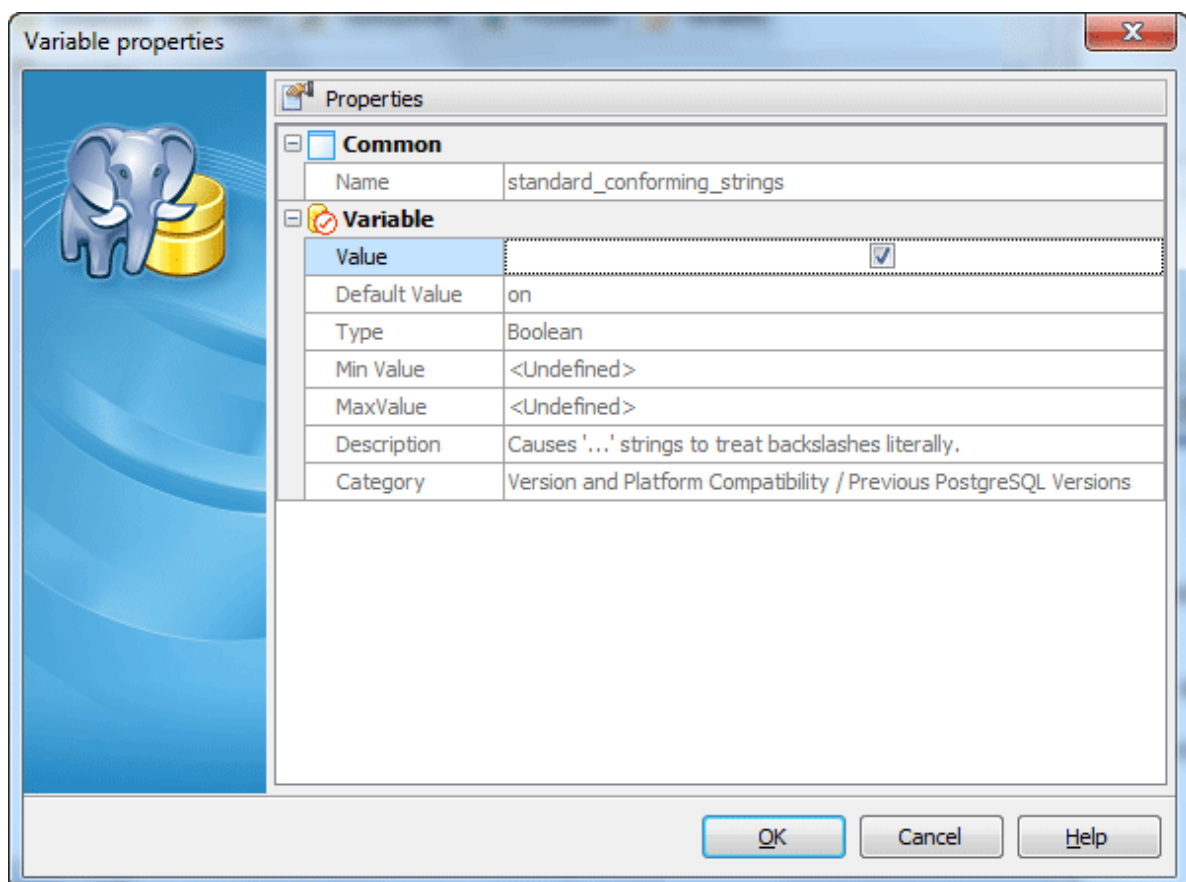
- select the server variables for editing in the explorer tree (type the first letters of the server variable name for quick search);
- select the [Edit Server Variable](#) item from the popup menu

or

- open the server in [Server Editor](#) and the [Server Variables](#) tab there;
- select the server variables to edit;
- press the **Enter** key or select the [Edit Server Variable](#) item from the popup menu (alternatively, you can use the corresponding link of the [Navigation Bar](#)).

6.3.1 Variable Editor

You can change the [Value](#) of server (environment) variable.



Note: The value will be changed for current PostgreSQL session only. To make it a rule for all sessions, you need to change *postgresql.conf* manually.

Here you can also edit the variable [Name](#), the current [Value](#) of the configuration option, [Min Value](#) and [Max Value](#), [Default Value](#), [Category](#) of the variable, the [Type](#) of the underlying data for the variable, and a short [Description](#) to the variable.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

6.4 Users

Every database cluster contains a set of users. Those users are separate from the users managed by the operating system on which the server runs. Users own database objects (for example, tables) and can assign privileges on those objects to other users to control who has access to which object. Only *superuser* can manage users.

Note: Since PostgreSQL 8.1 a role system was implemented to replace users and groups. For more information see [Roles](#)^[200].

■ How can I create a new user?

New users are created within [Create User Wizard](#)^[195]. In order to run the wizard you should either

- select the [Users](#) list or any object from that list in the explorer tree and then use the [Create New User](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Users](#) tab there and press **Insert** or select the [Create New User](#) item from the popup menu (Alternatively, use the corresponding link of the [Navigation Bar](#)).

■ How can I edit an existing user?

Users are edited within [User Editor](#)^[196]. In order to run the editor you should either

- select the user for editing in the explorer tree (type the first letters of the user name for quick search);
 - select the [Edit User](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Users](#) tab there;
 - select the user to edit;
 - press the **Enter** key or select the [Edit User](#) item from the popup menu (alternatively, you can use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a user?

To drop the existing user:

- select the user to drop in the explorer tree;
 - select the [Drop User](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Users](#) tab there;
 - select the user to drop;
 - press the **Delete** key or select the [Drop User](#) item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

6.4.1 Create User Wizard

[Create User Wizard](#) guides you through the process of creating a new user. See [How To Create User](#)^[194] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Use [System ID](#) to specify the PostgreSQL user ID of the new user. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned user ID plus one (with a minimum of 100) will be used as default.

Password

Sets the user's password. If you do not plan to use password authentication you can omit this option, but then the user will not be able to connect if you decide to switch to password authentication. The password can be set or modified later within the user editor.

Password Encrypted

This Checkbox allows you to control whether the password is stored encrypted in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password_encryption*.) If the presented password string is already in MD5-encrypted format, then it is stored encrypted as-is, regardless of whether checked or unchecked is specified (since the system cannot decrypt the specified encrypted password string). This allows reloading of encrypted passwords during dump/restore.

Note that older clients may lack support for the MD5 authentication mechanism that is needed to work with passwords that are stored encrypted.

Can create databases

These clauses define a user's ability to create databases. If it is specified, the user being defined will be allowed to create his own databases. Set false to deny a user the ability to create databases.

Can Create Users

These clauses determine whether a user will be permitted to create new users himself. Check to make the user a *superuser* who can override all access restrictions.

Valid until

Set an absolute time after which the user's password is no longer valid. If this clause is omitted the password will be valid for all time.

Groups

The name of an existing group into which the user is to be included as a new member. Multiple group names may be listed.

User variables

This step suggests you to specify the user variable that govern the expected behavior of the database system during the user session.

6.4.2 User Editor

[User Editor](#) is the basic PostgreSQL Maestro tool for working with existing users. It can be opened automatically after the user is created and is available on editing the user (see [How to edit user](#)^[194] for details).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

The [Properties](#) tab of this editor allows you to change the user password, the user's opportunity to create databases and users, the group list the user belongs to, the user grants.

The [Variables](#) tab is intended for managing user variables. Use [Variable Editor](#)^[178] for editing variables.

Use grid's popup menu to create new, edit or drop the selected variable. Using the popup menu you can also create a *copy* of the variable.

Name

Here you can view and change the user name.

System ID

The field stores the user System ID

Password

Here you can change the user's password.

Password Encrypted

The Checkbox allows you to control whether the password is stored encrypted in the system catalogs.

Can Create Databases

These clauses define a user's ability to create databases.

Can Create Users

These clauses determine whether a user will be permitted to create new users himself.

Valid until

Edit the absolute time after which the user's password is no longer valid. If this clause is omitted the password will be valid for all time.

Groups

The field stands for the server group list. The checkbox represents whether the user belongs to a group or not.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

User objects

Objects that are having this user as an owner are managed within the [Objects](#) tab of User Editor. Double click a row to open the necessary object editor. Use grid's popup menu items to edit, drop or rename object.

6.5 Groups

Groups are implemented for providing a way of grouping users logically to facilitate management of privileges: privileges can be granted to, or revoked from, a group as a whole.

Note: Since PostgreSQL 8.1 a role system was implemented to replace users and groups. For more information see [Roles](#)^[200].

■ How can I create a new group?

New groups are created within [Create Group Wizard](#)^[199]. In order to run this wizard you should either

- select the [Groups](#) list or any object from that list and then use the [Create New Group](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Groups](#) tab there and press **Insert** or select the [Create New Group](#) item from the popup menu (Alternatively, use the corresponding link of the [Navigation Bar](#)).

■ How can I edit an existing group?

Groups are edited within [Group Editor](#)^[199]. In order to open the editor you should either

- select the group for editing in the explorer tree (type the first letters of the group name for quick search);
 - select the [Edit Group](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Groups](#) tab there;
 - select the group to edit;
 - press the **Enter** key or select the [Edit Group](#) item from the popup menu (alternatively, you can use the corresponding link of the [Navigation Bar](#)).

■ How can I drop a group?

To drop the existing group:

- select the group to drop in the explorer tree;
 - select the [Drop Group](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Groups](#) tab there;
 - select the group to drop;
 - press the **Delete** key or select the [Drop Group](#) item from the popup menu
- (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

6.5.1 Create Group Wizard

[Create Group Wizard](#) guides you through the process of creating a new database group. See [How To Create Group](#)^[198] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Use [System ID](#) to specify the PostgreSQL group ID of the new group. This is normally not necessary, but may be useful if you need to recreate a group referenced in the permissions of some object. If this is not specified, the highest assigned group ID plus one (with a minimum of 100) will be used as default.

Users

Set a list of users to include in the group here. The users must already exist in the server.

See also: [Group Editor](#)^[199]

6.5.2 Group Editor

[Group Editor](#) is the basic PostgreSQL Maestro tool for working with existing groups. It can be open automatically after the group is created and is available on editing the group (see [How to edit group](#)^[198] for details).

You can open a group in [Group Editor](#) from the [Explorer Tree](#) or [Object Manager](#).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

Name

Here you can change the group name.

System ID

The field stores the group System ID

Users

Here is a list of users. Check (uncheck) the corresponding checkbox to include (exclude) a user in the group.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the [Object Properties](#) item of the popup menu of the selected object from the explorer tree.

6.6 Roles

Since version 8.1, PostgreSQL manages database access permissions using the concept of roles.

As per the server documentation a Role can be thought of as either a database User, or a Group of database users, depending on how the role is set up. Roles can own database objects (for example, tables) and can assign privileges on those objects to other roles to control who has access to which objects. Furthermore, it is possible to grant membership in a role to another role, thus allowing the member role use of privileges assigned to the role it is a member of.

■ How can I create a new role?

New roles are created within [Create Role Wizard](#)^[204]. In order to run the wizard you should either

- select the [Roles](#) list or any object from that list in the explorer tree and then use the [Create New Role](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Roles](#) tab there and press **Insert** or select the [Create New Role](#) item from the popup menu (Alternatively, use the corresponding link of the Navigation Bar).

■ How can I edit an existing role?

Roles are edited within [Role Editor](#)^[204]. In order to open the editor you should either

- select the role for editing in the explorer tree (type the first letters of the role name for quick search);
 - select the [Edit Role](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Roles](#) tab there;
 - select the role to edit;
 - press the **Enter** key or select the **Edit Role** item from the popup menu (alternatively, you can use the corresponding link of the **Navigation Bar**).

■ How can I drop a role?

To drop the existing role:

- select the role to drop in the explorer tree;
 - select the [Drop Role](#) item from the popup menu
- or
- open the server in [Server Editor](#) and the [Roles](#) tab there;
 - select the role to drop;

- press the **Delete** key or select the **Drop Role** item from the popup menu (alternatively, you may use the corresponding link of the [Navigation Bar](#))

and confirm dropping in the dialog window.

6.6.1 Create Role Wizard

[Create Role Wizard](#) guides you through the process of creating a new role. See [How To Create Role](#)^[200] to learn how to run this wizard.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)^[43]. Below you will find a description of wizard steps that are unique for the current object.

Create role wizard

Create New Role
Edit role options

Properties

- Common**

Name	admin
Comment	
- Role**

Can login	<input type="checkbox"/>
Inherits rights from parent roles	<input checked="" type="checkbox"/>
Connection limit (-1 for no limits)	-1
Valid until	
- Authentication**

Password	*****
Password (again)	*****
Password encrypted	<input checked="" type="checkbox"/>
- Privileges**

Can create users (superuser)	<input checked="" type="checkbox"/>
Can create databases	<input checked="" type="checkbox"/>
Can create roles	<input checked="" type="checkbox"/>
Can update system catalog directly	<input checked="" type="checkbox"/>

Buttons: **Help**, **< Back**, **Next >**, **Cancel**

[Specifying role properties](#)^[202]

[Managing role membership and environment](#)^[203]

6.6.1.1 Specifying role properties

Specify role options according to your needs. The detailed description is given below.

Name

The field contains the new role name as it was set on the previous wizard step. Use it to edit the name of role being made.

Can Login

Use the checkbox to determine whether a role is allowed to log in; that is, whether the role can be given as the initial session authorization name during client connection. A role having with the option checked can be thought of as a user. Roles without this attribute are useful for managing database privileges, but are not users in the usual sense of the word.

Inherits rights from parent roles

The clause determines whether a role "inherits" the privileges of roles it is a member of. A role with the option checked can automatically use whatever database privileges have been granted to all roles it is directly or indirectly a member of. Otherwise, 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.

Connection limit (-1 for no limit)

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

Valid until

Set an absolute time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.

Can Create Users (Superuser)

This clause determines whether the new role is a "superuser", who can override all access restrictions within the database. Superuser status is dangerous and should be used only when really needed. You must yourself be a superuser to create a new superuser.

Can create databases

The option defines a role's ability to create databases. If it is specified, the role being defined will be allowed to create his own databases. Set false to deny a role the ability to create databases.

Can create roles

Use the checkbox to define a role's ability to create databases. If the option is specified, the role being defined will be allowed to create new databases. Specifying the opposite will deny a role the ability to create databases.

Can update system catalog directly

Use **System ID** to specify the PostgreSQL role ID of the new role. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned role ID plus one (with a minimum of 100) will be used as default.

Password

Sets the role's password. If you do not plan to use password authentication you can omit this option, but then the role will not be able to connect if you decide to switch to password authentication. The password can be set or modified later within the role editor.

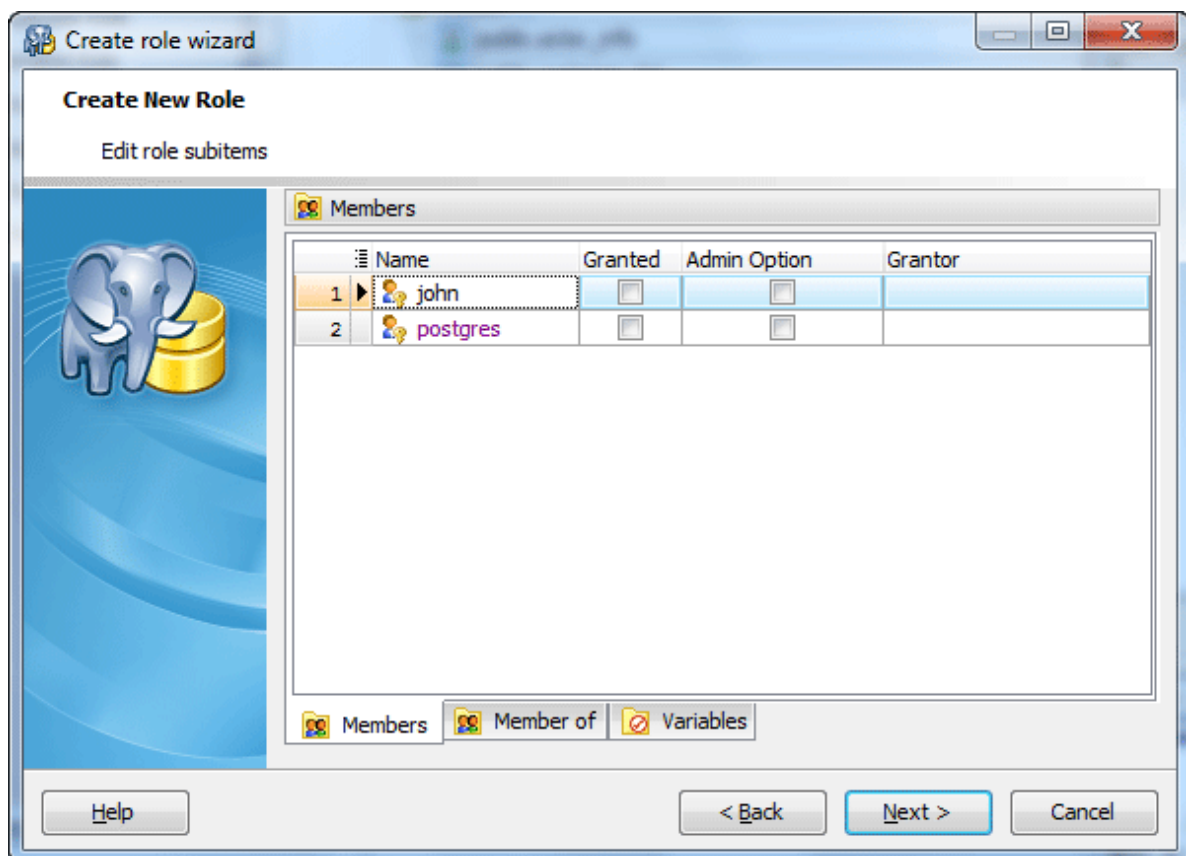
Password Encrypted

This Checkbox allows you to control whether the password is stored encrypted in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password_encryption*.) If the presented password string is already in MD5-encrypted format, then it is stored encrypted as-is, regardless of whether checked or unchecked is specified (since the system cannot decrypt the specified encrypted password string). This allows reloading of encrypted passwords during dump/restore.

Note that older clients may lack support for the MD5 authentication mechanism that is needed to work with passwords that are stored encrypted.

6.6.1.2 Membership and environment

The wizard step allows you to define the new role members, to specify roles the role being made is a member of and customize the role environment variables.



Members

Specify here one or more existing roles which are automatically added as members of the new role. **Admin option** gives to the role the right to grant membership in this role to others.

Member of

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

Variables

Set this role's session defaults for the specified configuration parameters to the given value. To add a new variable, right-click the grid and choose the appropriate item from the popup menu.

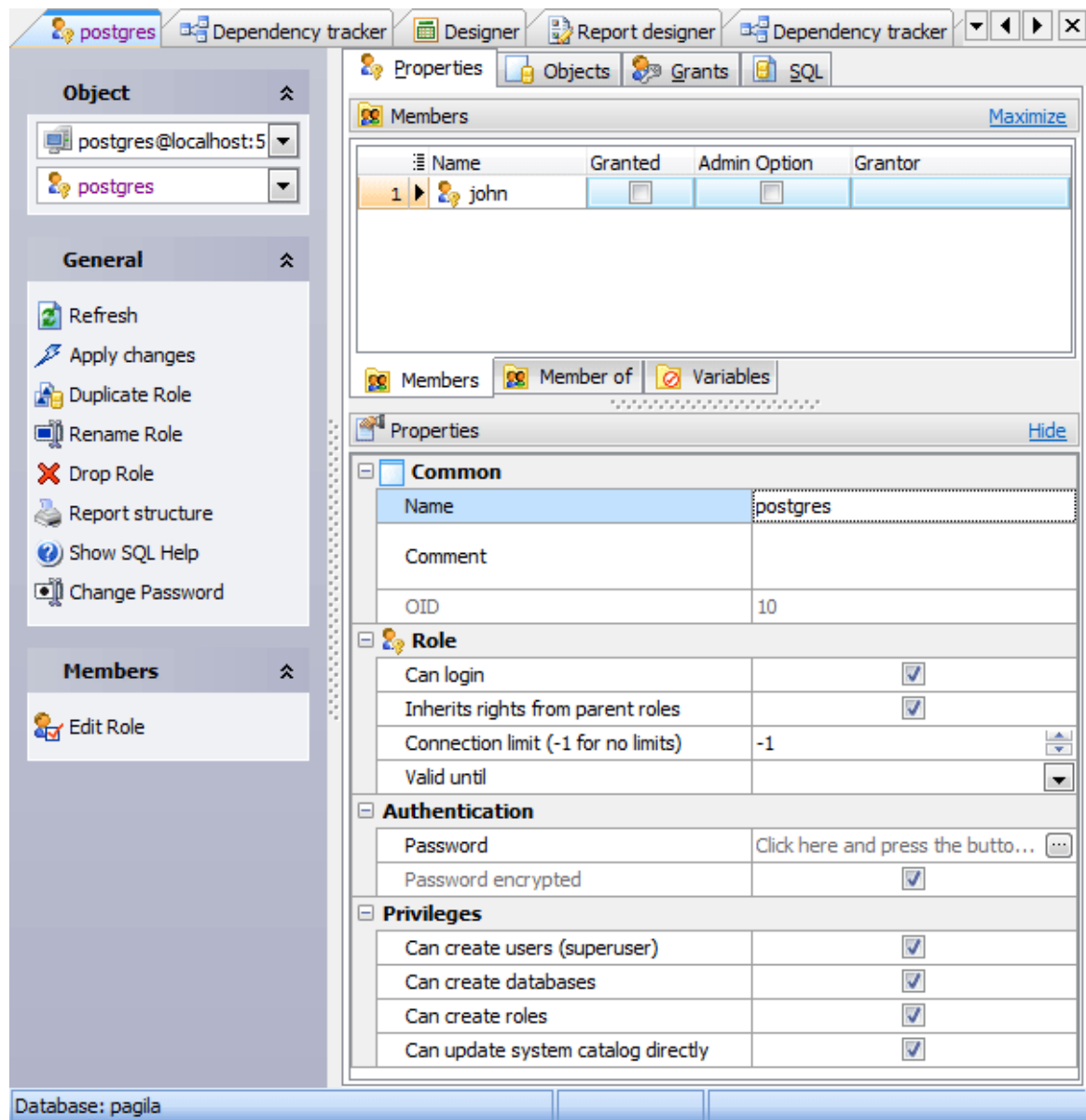
6.6.2 Role Editor

[Role Editor](#) is the basic PostgreSQL Maestro tool for working with existing roles. It can be opened automatically after the role is created and is available on editing the role (see [How to edit role](#)^[200] for details).

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

[Editing role properties](#)^[205]

[Viewing role objects](#)^[207]



6.6.2.1 Editing role properties

Role Editor provides you with an ability to edit role properties and membership. The detailed description is given below.

Members

Specify here one or more existing roles which are automatically added as members of the new role. **Admin option** gives to the role the right to grant membership in this role to others.

Member of

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

Variables

The tab is implemented to view and edit this role's session defaults for the specified configuration parameters to the given value.

[Name](#)

Edit the role name within the field. To do it more quickly use the [Rename Role](#) link of pop-up menu in the explorer tree.

[Can Login](#)

The checkbox determines whether a role is allowed to log in; that is, whether the role can be given as the initial session authorization name during client connection. A role having with the option checked can be thought of as a user. Roles without this attribute are useful for managing database privileges, but are not users in the usual sense of the word.

[Inherits rights from parent roles](#)

The clause determines whether a role "inherits" the privileges of roles it is a member of. A role with the option checked can automatically use whatever database privileges have been granted to all roles it is directly or indirectly a member of. Otherwise, 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.

[Connection limit \(-1 for no limit\)](#)

The field value specifies how many concurrent connections the role can make. -1 (the default) means no limit.

[Valid until](#)

The field contains an absolute time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.

[Can Create Users \(Superuser\)](#)

This clause determines whether the new role is a "superuser", who can override all access restrictions within the database.

[Can create databases](#)

The option defines a user's ability to create databases. If it is specified, the role being defined will be allowed to create his own databases. Set false to deny the role the ability to create databases.

[Can create roles](#)

Use the checkbox to define a role's ability to create databases. If the option is specified, the role being defined will be allowed to create new databases. Specifying the opposite will deny a role the ability to create databases.

[Can update system catalog directly](#)

Use [System ID](#) to specify the PostgreSQL role ID of the new role. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned role ID plus one (with a minimum of 100) will be used as default.

[Password](#)

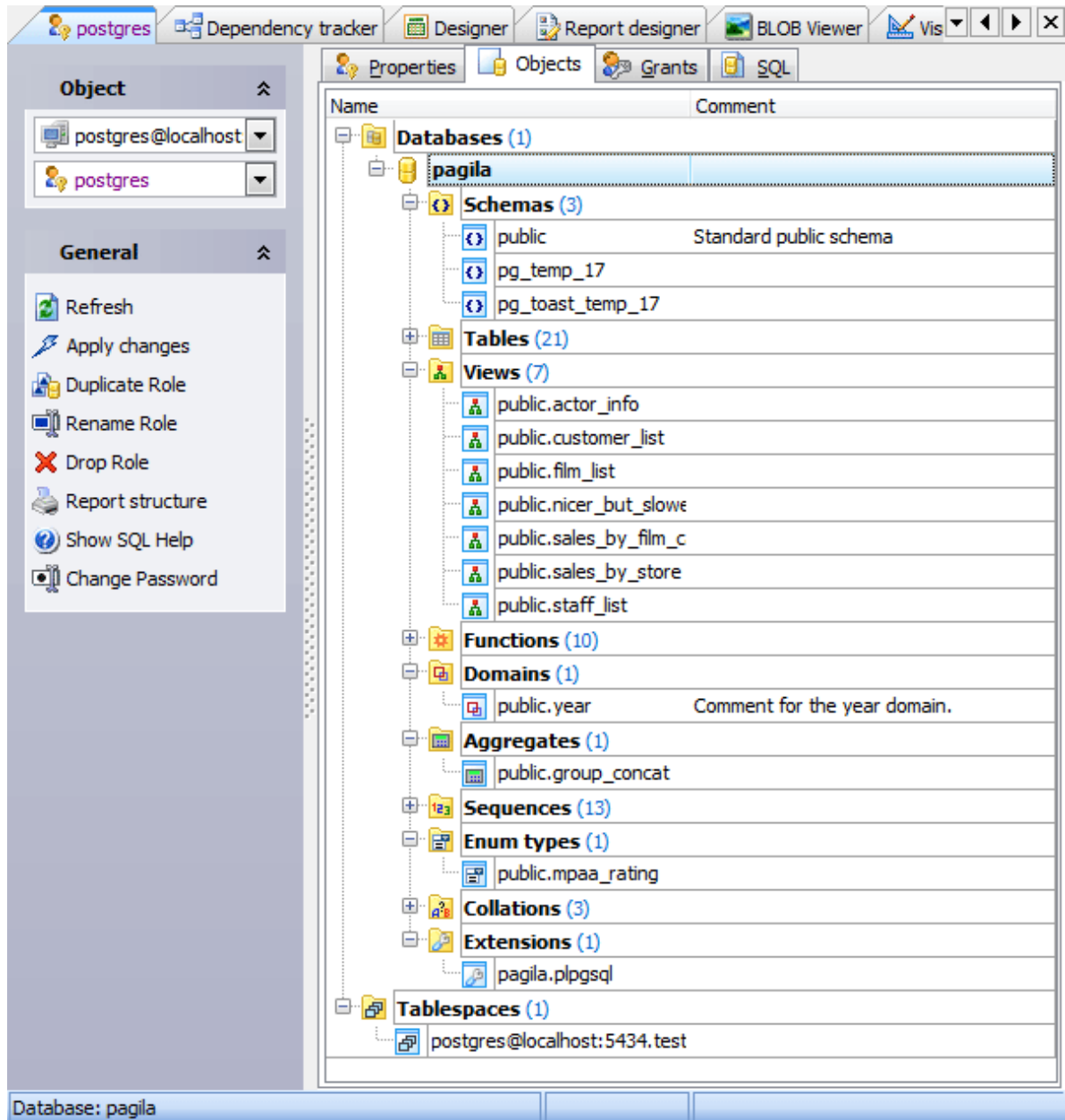
Use the field to edit the role's password. If you do not plan to use password authentication you can omit this option, but then the role will not be able to connect if you decide to switch to password authentication.

Password Encrypted

This Checkbox allows you to control whether the password is stored encrypted in the system catalogs.

6.6.2.2 Viewing role objects

Objects that are having this role as an owner are managed within the [Objects](#) tab of [Role Editor](#)^[204]. Double click a row to open the necessary object editor. Use grid's popup menu items to Edit, Drop or Rename Object.



6.7 Tablespaces

Tablespaces are being implemented in PostgreSQL Server 8.0 and higher. They 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.

■ How can I create a new tablespace?

New tablespaces are created within [Create Tablespace Wizard](#)^[209]. In order to run this wizard you should either

- select the **Tablespaces** list or any object from that list and then use the **Create New Tablespace** item from the popup menu
- or
- open the server in **Server Editor** and the **Tablespaces** tab there and press **Insert** or select the **Create New Tablespace** item from the popup menu (Alternatively, use the corresponding link of the **Navigation Bar**).

■ How can I edit an existing tablespace?

Tablespaces are edited within [Tablespace Editor](#)^[210]. In order to open the editor you should either

- select the tablespace for editing in the explorer tree (type the first letters of the tablespace name for quick search);
 - select the **Edit Tablespace** item from the popup menu
- or
- open the server in **Server Editor** and the **Tablespaces** tab there;
 - select the tablespace to edit;
 - press the **Enter** key or select the **Edit Tablespace** item from the popup menu (alternatively, you can use the corresponding link of the **Navigation Bar**).

■ How can I drop a tablespace?

To drop the existing tablespace:

- select the tablespace to drop in the explorer tree;
 - select the **Drop Tablespace** item from the popup menu
- or
- open the server in **Server Editor** and the **Tablespaces** tab there;
 - select the tablespace to drop;
 - press the **Delete** key or select the **Drop Tablespace** item from the popup menu (alternatively, you may use the corresponding link of the **Navigation Bar**)

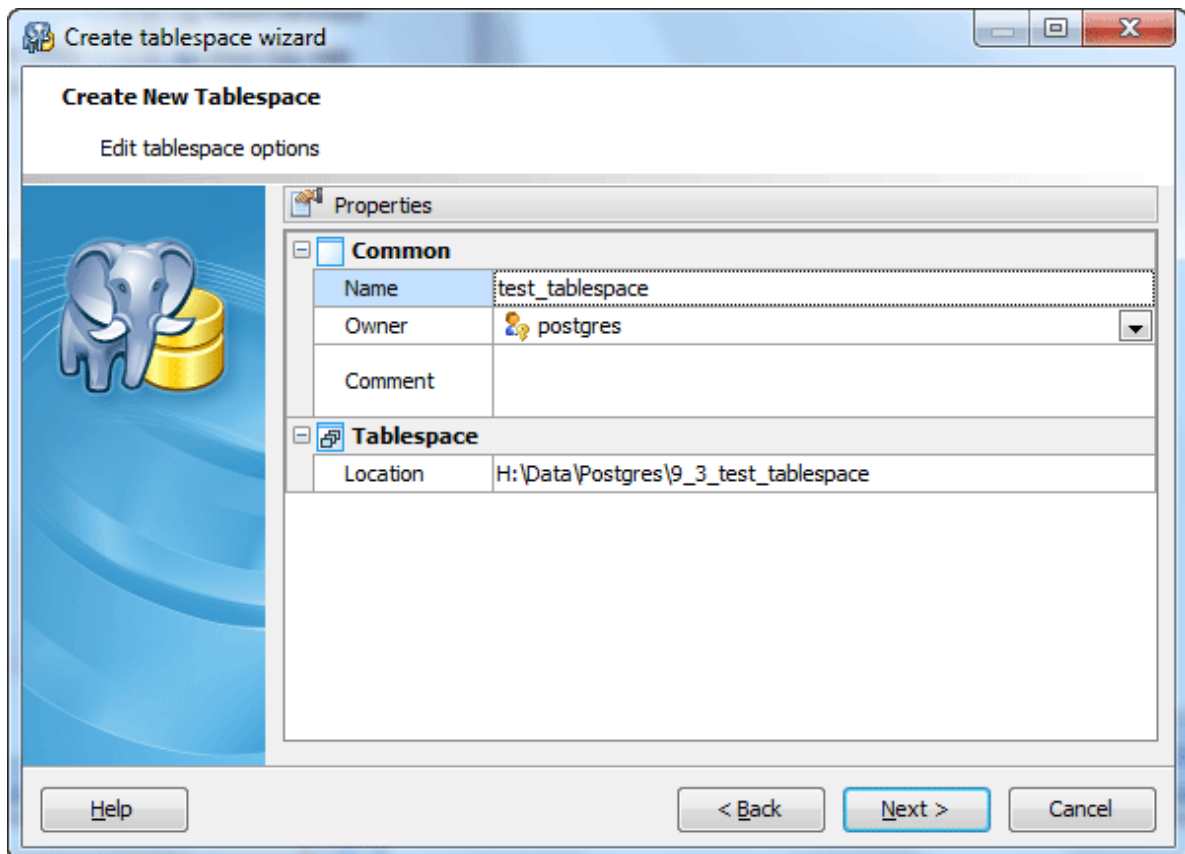
and confirm dropping in the dialog window.

6.7.1 Create Tablespace Wizard

[Create Tablespace Wizard](#) guides you through the process of creating a new server tablespace.

The basic principles of Create Object Wizards in PostgreSQL Maestro are explained in a [separate topic](#)⁴³. Below you will find a description of wizard steps that are unique for the current object.

Specify tablespace options according to your needs. The detailed description is given below.



Owner

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

Location

Set 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.

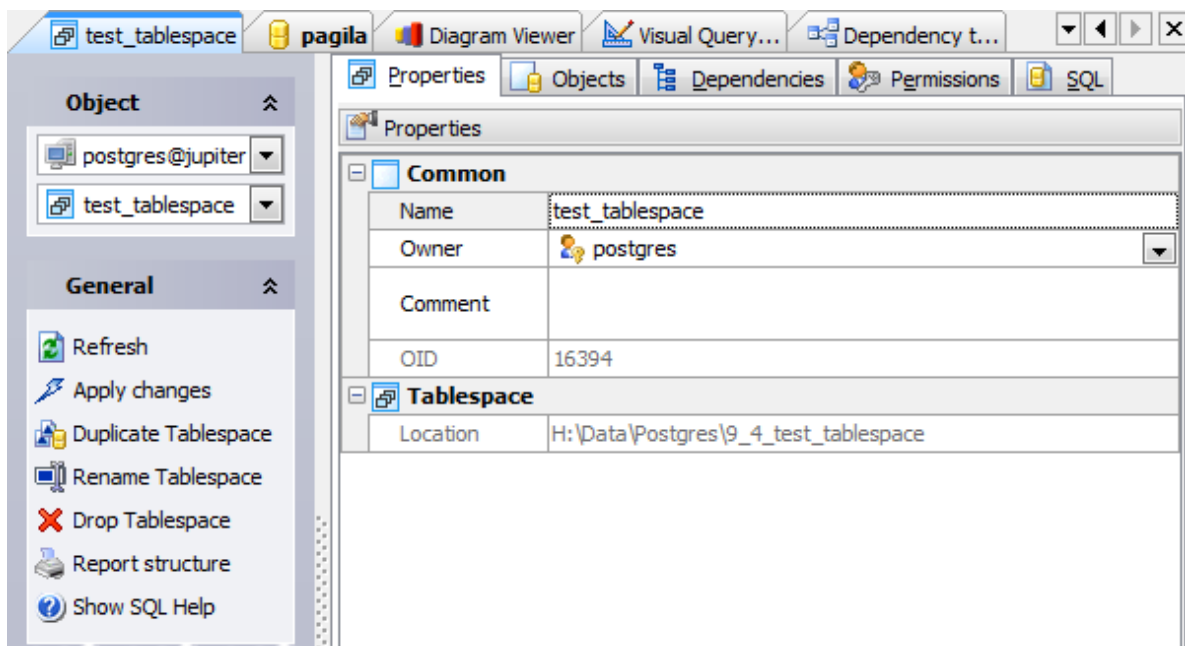
6.7.2 Tablespace Editor

Tablespace Editor allows you to browse options of existing tablespaces and change some of them. The editor can be open automatically after the tablespace is created and is available on editing the tablespace (see [How to edit tablespace](#)^[208] for details) or from the corresponding node of Explorer Tree or Object Manager.

The basic principles of Object Editors in PostgreSQL Maestro are explained in a [separate topic](#)^[46]. Below you will find a description of editor tabs that are unique for the current object.

[Tablespace properties](#)^[210]

See also: [Create Tablespace Wizard](#)^[209]



6.7.2.1 Editing tablespace properties

The **Properties** tab allows you to rename the tablespace, change its owner and learn the OID.

The tablespace OID (object identifier) is represented in the OID field. It is a serial number automatically added by PostgreSQL to all tablespaces. The OID is *read-only*.

Location

This field represents indicates the directory where the tablespace is located.

The **Objects** tab displays all the tables, indexes, and materialized views that belong to the tablespace.

To apply the changes, select the [Apply Changes](#) item in the [Navigation bar](#) or use **Ctrl+F9** or **Ctrl+F7** shortcut keys.

It is also possible to modify object properties without opening the object editor: use the

[Object Properties](#) item of the popup menu of the selected object from the explorer tree.

7 Queries

PostgreSQL Maestro provides several tools for working with SQL queries:

- [SQL Editor](#)^[214] for editing the query text directly and executing SELECT queries;
- [Visual Query Builder](#)^[219] for building SELECT, INSERT, UPDATE and DELETE queries visually;
- [SQL Script Editor](#)^[266] for executing SQL scripts.

Both SQL Editor and Visual Query Builder supports [parameters in queries](#)^[218]

Save frequently used queries to profiles and manage them in the same way as if they were database objects. This means that you can view queries in the explorer tree, in [Object Manager](#) and [Object Browser](#), use them in [BLOB Viewer](#) and [Diagram Viewer](#), perform drag-and-drop operation upon them, and copy them to clipboard like you copy an object.

How can I create a new SQL query?

New queries can be created either in [SQL Editor](#) or in [Visual Query Builder](#).

To create a new query in [SQL Editor](#):

- select the [Tools | SQL Editor](#) main menu item;
- select the [Create New Query](#) item from the navigation bar;
- edit the query text on the [Editor](#) tab of [SQL Editor](#).

To create a new query in [Query Builder](#):

- select the [Tools | Visual Query Builder](#) main menu item;
- build the query on the [Diagram](#) tab of [Visual Query Builder](#).

PostgreSQL Maestro also provides you with SQL Generator, a tool to create simple SQL statements.

How can I save a query to a file/profile?

To save an existing query from the editor:

- to save the query to profile, use the [Save to profile](#) link from the [Navigation bar](#).
- to save the current query to an *.sql file, select the [Save to file](#) item from the [Navigation bar](#);
- to save all the opened queries to one file, select the [Save all queries](#) item from the [Navigation bar](#);
- to save the designed diagram, select the [Save diagram](#) item from the [Navigation bar](#) of the [Diagram](#) tab of [Visual Query Builder](#).

■ How can I edit an existing SQL query?

Queries can be opened either in [SQL Editor](#) or in [Visual Query Builder](#).

You can open the query directly from the Explorer tree with a double click or using popup menu. By default it will be opened in [SQL Editor](#).

To edit a query from file, open [SQL Editor](#) (the [Tools | SQL Editor](#) main menu item) and use [Load From File](#) from the [Navigation Bar](#) of [SQL Editor](#) to load a query from an `*.sql` file.

To edit a query in [Query Builder](#), open the builder (the [Tools | Visual Query Builder](#) main menu item) and then perform one of the following operations:

- to edit a query from a profile, drag it from the [Explorer](#) and drop on the [Editor](#) tab;
- to load a previously saved diagram, use the [Load Diagram](#) item from the [Navigation Bar](#);
- to load a query from an `*.sql` file, open the [Editor](#) tab and select the [Load query](#) item from the [Navigation Bar](#).

On the [Query Builder](#) opening the [Diagram](#) tab contains the last edited query.

■ How can I execute an SQL query?

To execute a query:

- create a new query or open the existing one;
- select the [Execute Query](#) item from the navigation bar of [SQL Editor](#) or [Visual Query Builder](#) respectively;
- view/edit the returned data on the [Result](#) tab.

7.1 SQL Editor

[SQL Editor](#) is a tool for creating and executing SELECT queries. It allows you to create and edit SQL text for the query, prepare and execute queries, and view the results of execution. To open [SQL Editor](#), select the [Tools | SQL Editor](#) main menu item. The most popular query management actions (creating, editing, deleting) are covered by the corresponding [topic](#)^[212].

To use the editor for working with several queries, open new query tab with the [Create new query](#) link on the Navigation bar. With the tabs' popup menu you can create a new query, close existing one, save the query to profile, etc even if editor's navigation bar is closed. Queries' tabs [can be](#)^[328] displayed at the all sides of the editor (bottom, top, left or right).

For more information about query executing and working with query result see the [corresponding topic](#)^[216].

Working with query text

The [popup menu](#) of the editing area provides you with standard operations for working with text such as *Cut* (**Ctrl+X**), *Copy* (**Ctrl+C**), *Paste* (**Ctrl+V**), *Undo* (**Ctrl+Z**), *Redo* (**Shift+Ctrl+Z**) along with a possibility to convert selected text to different cases (*lower*, *UPPER*, and *NameCase*).

You can also comment/uncomment selected text (**Shift+Ctrl+.** and **Shift+Ctrl+,** shortcuts respectively). If no text is selected, the whole line will be commented. By the way, it is not necessary to select commented text to uncomment it, just press **Shift+Ctrl+.** having the cursor inside the commented text. Both kinds of comments (single-line and multi-line) are supported. [SQL Formatter](#)^[215] is also at your disposal.

SQL Editor allows you to use [Visual Query Builder](#)^[219] modal instance to design query visually and load the result query text directly in the editor area. For this purpose use the [Design query](#) link of the editor area's popup menu.

Code completion

PostgreSQL Maestro provides you with code completion (as on the screen below) to select from a list of tables, columns, views, or other objects without having to manually enter the object's name in the editor. You can activate the completion list by pressing the **Ctrl+Space** key combination.

Syntax highlighting

Database objects are highlighted in the text. You can open the proper object editor by clicking the object name in the text with the **Ctrl** key pressed or with the [Find Object](#) link on the [Navigation bar](#). To adjust the highlighting settings, use [SQL highlight options](#)^[348].

Line modification markers

Lines of code that have been edited during the current session are indicated with a yellow line in the left margin of the editor. When you save the file, the yellow markers turn green. Thus at any time, yellow markers show changed but unsaved

lines of code, and green markers show changes in this session that have been saved.

Find and replace text

Use find and replace to search for, and optionally, replace text in the [SQL Editor](#). To open [Find text/Replace text](#) window, use [Edit | Find/Replace](#) main menu item, corresponding link of popup menu, or **Ctrl+F/Ctrl+H** shortcut. You can also use the [Search again](#) link to apply recent Find text dialog.

Transaction management

SQL Editor supports the explicit transaction management. You can execute queries either in [autocommit mode](#) (default behavior) or [manage transactions manually](#). In the second case you have to issue the *BEGIN* statement to start a transaction and explicitly end the transaction by *COMMIT* or *ROLLBACK* statements (it is also possible to use the corresponding links at the editor's navigation bar).

Managing the query text

To load query from .sql file, use the corresponding link on the Navigation bar. You can also find there links allowing you to save query text to file, export the contents of the editor to RTF and HTML formats (to file or to clipboard), copy the selected text from to clipboard as a ready-to-use string written in one of the following programming languages: C#, C++, Delphi (Object Pascal), and Java, and also print/preview the contents of the editor.

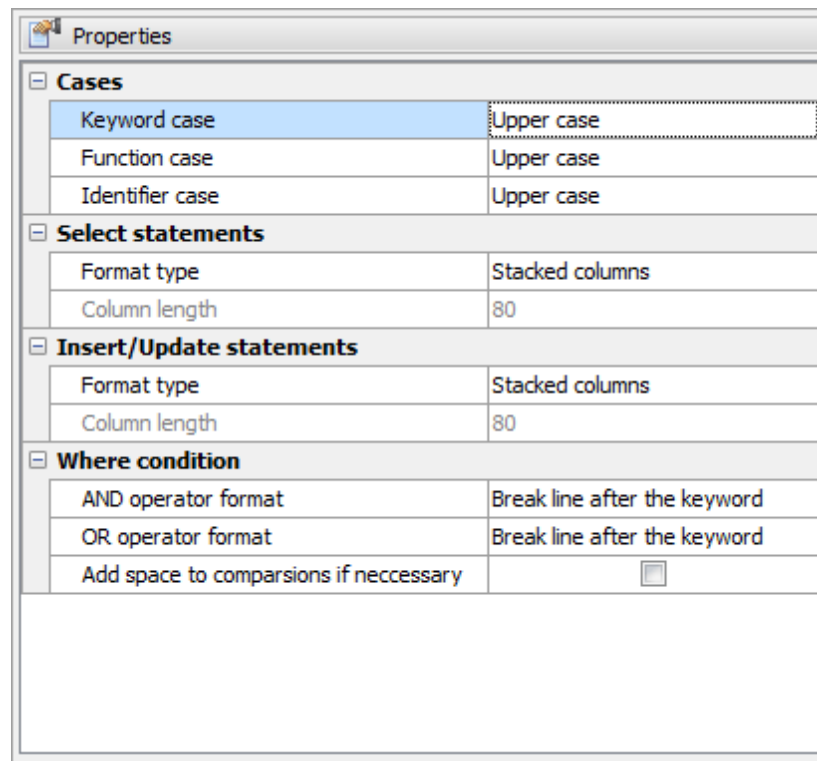
See also: [Visual Query Builder](#)^[219], SQL Script Editor, [SQL Editor Options](#)^[328]

7.1.1 SQL Formatter

PostgreSQL Maestro provides you with [SQL Formatter](#) for DML statements (*SELECT*, *INSERT*, *UPDATE* and *DELETE*). It can be invoked through the [Format SQL](#) link on the [SQL Editor](#)'s navigation bar (**Ctrl+Alt+D** shortcut).

The following options allows you to tune up SQL scripts according to your preferences.

- Cases (for keywords, functions, and identifiers);
- Format type and column length for *INSERT/UPDATE*, and *SELECT* statements;
- *AND* and *OR* operators format.



7.1.2 Executing query

SQL Editor provides you with several variants of the query executing.

- To execute all statements of the text area with result data, click the [Execute query](#) item of the Navigation bar or use **F5**, **F8**, or **F9** shortcuts. Statements of each tab of SQL Editor are executed together in a separate thread in order to continue your work with the software while the query is executing.
- You can also [execute query as script](#) (**Shift+F5**, **Shift+F8**, **Shift+F9**). In this case the query does not return data.
- To execute only a selected part of the query text, use [Execute selected only](#) or the **Alt+F5**, **Alt+F8**, **Alt+F9** shortcuts.
- There is also a possibility to execute a statement at the cursor position. For this purpose, use the [Execute at cursor](#) link at the Navigation bar or use the **Ctrl+F5**, **Ctrl+F8**, or **Ctrl+F9** shortcuts.

If the query text is correct, the query is executed, and if the query statement is supposed to return data (e.g. SELECT statement), the [Result](#) tab opens with the data returned by the query. If an error occurs while executing the query, execution stop is stopped and the appropriate error message is displayed in the Information tab.

The [Result](#) area displays the result data in grid. All principles of working with data you can find in [Data Management](#) ²²⁸ section.

SQL Editor: ... BLOB Viewer Dependency tracker Visual Query Builder GAME@

Database

sdb_demo at sun

General

- Execute query
- Execute as script
- Execute selected only
- Format SQL
- Show SQL Help
- Configure SQL Editor
- Open new instance

Query management

- Create new query
- Delete current query
- Delete all queries
- Save to profile
- Run Query Builder
- Run SQL Script Editor

Files

- Load from file
- Save to file
- Save all queries

Data management

- Export data
- Get SQL dump
- Print data

```

SELECT
  NBA.GAME.GAME_DATE,
  HOME_TEAM.CAPTION AS HOME_TEAM,
  (SELECT
    SUM(NBA.GAME_QUARTER.SCORE) AS FIELD_1
  FROM NBA.GAME_QUARTER
  WHERE
    (NBA.GAME_QUARTER.GAME_ID = NBA.GAME.ID)
    (NBA.GAME_QUARTER.TEAM_ID = HOME_TEAM.ID)
  (SELECT
    SUM(NBA.GAME_QUARTER.SCORE) AS FIELD_2
  FROM NBA.GAME_QUARTER
  WHERE
    (NBA.GAME_QUARTER.GAME_ID = NBA.GAME.ID)
    (NBA.GAME_QUARTER.TEAM_ID = HOME_TEAM.ID)
  )

```

Result 1 Result 2

Table

Drag a column header here to group by that column

	GAME_DATE	HOME_TEAM	HOME_TEAM_SCORE
	Click here to define a filter		
1	28.10.2008	Boston Celtics	
2	28.10.2008	Chicago Bulls	1
3	28.10.2008	Los Angeles Lakers	
4	29.10.2008	Philadelphia 76ers	
5	29.10.2008	Orlando Magic	
6	29.10.2008	Washington Wizards	
7	29.10.2008	New York Knicks	1
8	29.10.2008	Detroit Pistons	1
9	29.10.2008	Minnesota Timberwolves	
10	29.10.2008	Oklahoma City Thunder	
11	29.10.2008	San Antonio Spurs	

Records fetched: 1315

Information
1315 rows fetched (0,56 sec)

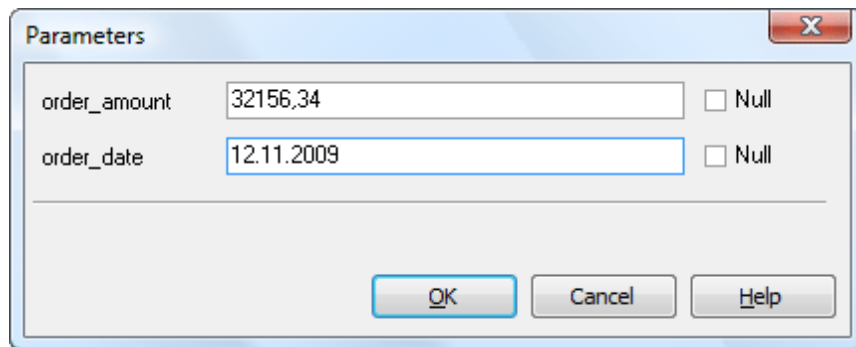
RESULT_LIST Query 5 Query 1

Database: sdb_demo at sun

7.1.3 Query Parameters

Both [SQL Editor](#)^[214] and [Visual Query Builder](#)^[219] admit to using parameters inside the query text. A parameter is a kind of variable. Its value can be specified just before the query execution in the [Parameters](#) window. In the query text the parameter should appear as an identifier with a colon (':') at its beginning, e.g. `:param1`.

The [Parameters](#) dialog is used to specify the query parameters as well as values of the input parameters of procedures or functions before the execution. Enter parameter values and click the **OK** button to apply the values and execute the query or use the **Cancel** button to abort the execution.



Note: To allow use parameters in query text, check the corresponding option at the [Tools](#)^[325] tab of PostgreSQL Maestro Options.

7.2 Visual Query Builder

[Visual Query Builder](#) is provided for building data manipulation statements visually. It allows you to create and edit queries without knowledge of SQL, prepare and execute queries, and view the results of the execution. Builder can produce *INSERT*, *UPDATE* and *DELETE* statements as well as the *SELECT* statements containing subqueries and/or *UNIONS*. One instance of the builder can be used only for one query at a time. To open [Visual Query Builder](#), select the [Tools | Query Builder](#) main menu item.

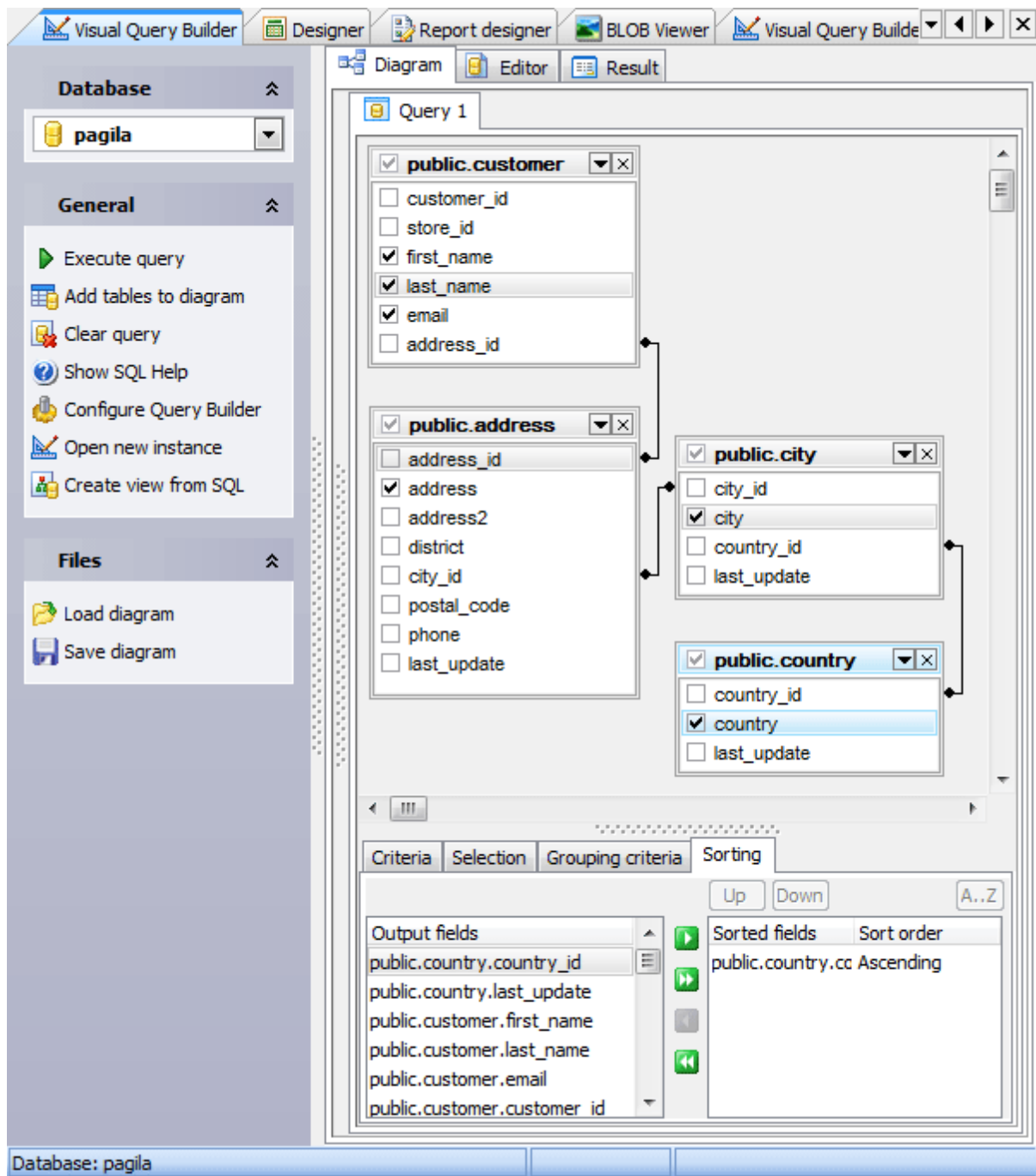
The most popular query management actions (creating, editing, deleting) are covered by the corresponding [topic](#)^[212].

Builder consists of 3 tabs:

- [Diagram](#)^[220] - to create a query from a graphical interface,
- [Editor](#)^[225] - to modify the query text before its executing,
- [Result](#)^[226] (appears after the query executing) - for working with data the query returns.

The builder also allows you to create a view based on the prepared query. For this purpose after the query creating and possibly testing use the Create view from SQL link at the Navigation bar to invoke the corresponding window, and specify [view properties](#)^[109].

See also: [SQL Editor](#)^[214], [Visual Query Builder Options](#)^[330], [Query Parameters](#)^[218]



7.2.1 Creating query diagram

The **Diagram** tab is the main area of Visual Query Builder. Using its graphical interface you can select tables and views, join or select columns, and add conditions to the statement.

The **Query Explorer** field occupies the left side of Visual Query Builder main window. All the queries included in the result query (unions, subqueries) are represented at the Query Explorer for prompt access. They are grouped by kind and listed under the according node.

Below step-by-step description of query diagram creating.

- **Select the statement type** from the drop-down list at the top of the [Diagram](#) tab (*SELECT, INSERT, UPDATE, DELETE*).

■ **Add required tables to the Diagram area.**

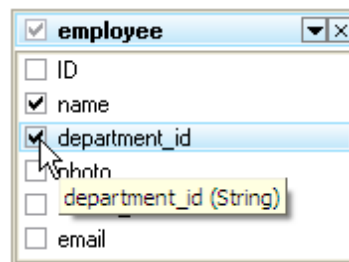
Use the [Add Table\(s\)](#) link of the area popup menu and select tables from the opened window (Use **Ctrl** or **Shift** pressed to select several tables).

To add only one table, simply drag it from the [Database Explorer](#) or from [Object Manager/Browser](#) to the [Diagram](#) area.

To remove the object, close its window or select the object and press the **Delete** key.

■ **Pick up columns with data to output**

To include a table field to the query, tick off the option box to the left of the field name in the list or double-click it to see the blue icon next to the field name.

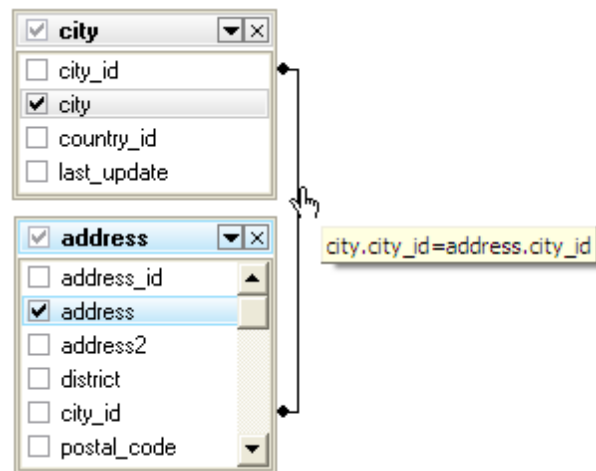


To include all the table fields, tick off the option box to the left of the table caption. In case none fields is included, the SQL statement is generated as `SELECT * FROM <Table_Name>`, i.e. all the fields are selected.

To remove the fields from the query, uncheck the corresponding boxes.

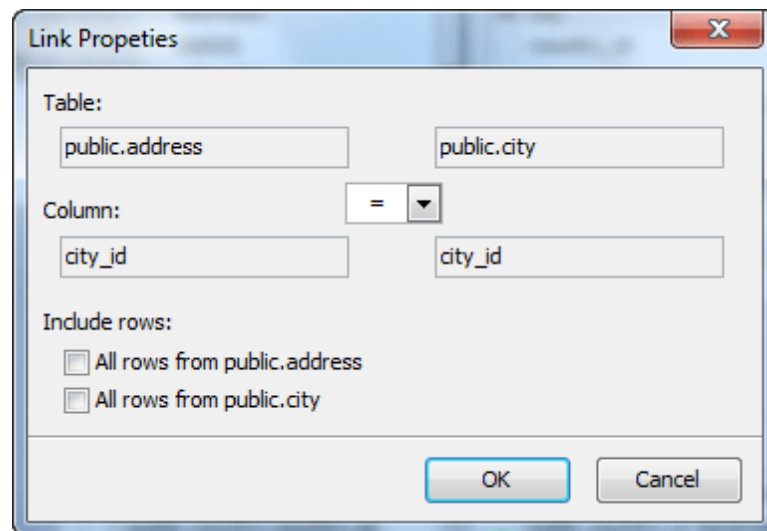
■ **Join tables if necessary**

Visual Query Builder supports *INNER JOIN*, *LEFT OUTER JOIN*, and *RIGHT OUTER JOIN*. To associate database objects by two fields, drag a field from the first object's field list to a field from another object's field list. This will set a link between these objects by the selected fields. After you finish dragging, a line will appear between the linked fields. By default *INNER JOIN* syntax will be used.



You can view the properties of the object association from the query tab directly. Just set the cursor to the link line. A hint containing the association condition will appear.

To edit the properties, select the [Properties](#) item from the popup menu. A dialog window will appear, there you can change the association condition by selecting it from the list (`=`, `>`, `<`, `>=`, `<=`, `<>`). To create *LEFT OUTER JOIN* / *RIGHT OUTER JOIN* statements, check [All rows from first_table](#)/[All rows from second_table](#) from the window.



To remove a link between objects, select the [Delete Link](#) item from the popup menu.

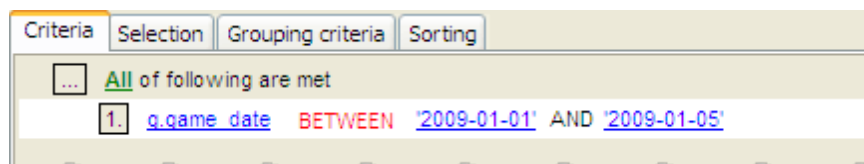
To delete all the links of an object, click the '-' button next to the object alias. To insert a point to the link line, select the [Insert Point](#) item from the popup menu, and the new point will appear. Using the point you can move the link line. It does not cause any changes in the query but makes the diagram performance vivid and the visual building more obvious.

Specify WHERE condition

Criteria tab allows you to set the selection conditions. To add a condition, click the button on the left and select the [Add condition](#) item in the popup menu. Edit the condition by clicking its parts and setting their values. Clicking the button to the left of the condition string activates the popup menu which allows you to add a new condition of the same enclosure level, add a new enclosure level, delete the current condition, open or close the condition if it is composite.

A simple condition string contains three fields: an argument, a condition and a second argument (if required for the condition). Clicking each field allows you to set its value. Clicking the argument field make it possible to edit the argument as a text field. You can set a table name or a definite value in this field. The popup menu of the field in the editing mode which contains the [Insert Field](#) function (also called by the **Shift+Enter** hot keys combination).

This function allows you to choose a field from the list of all the table fields available in the query. The popup menu of the condition field allows you to specify the condition you need. The way of proceeding the condition is set in the upper string of the area (*All, Any, None, or Not all* of the following are possible variants). Click the underlined word to modify it.



Create subquery if necessary

You can add one or more subqueries to further limit the tables and records returned from a *SELECT* statement when setting a *WHERE* condition in the query builder. To add subquery:

- open [Criteria](#) tab;
 - click the button on the left and select the [Add condition](#) item in the popup menu;
 - right click on an argument field and use the [Insert query](#) link of the popup menu;
 - build the subquery in the new query tab that have appeared in the [Diagram](#) area,
- or
- open [Selection](#) tab;
 - use the [Insert query](#) link of the popup menu;
 - build the subquery in the new query tab that have appeared in the [Diagram](#) area.

Use column aliases

You can set/edit the object alias directly from the query tab by double-clicking the object caption.

Criteria	Selection	Grouping criteria	Sorting
<input type="checkbox"/> Select only unique records			
Source field name	Name of output field	Aggregate	Grouping
nba.game.game_date	Game_date		
home_team.caption	caption		
(SELECT SUM(nba.game_quarter.score)	Home_team_score		
(SELECT SUM(nba.game_quarter.score)	Away_team_score		
away_team.caption	caption		
nba.channel.short_caption	short_caption		

In case the alias is used as the expression's column name use the [Selection](#) tab displays the output fields of the query. It allows you to edit the names of the query or CASE output fields, set their displaying order and set the aggregate functions (*SUM*, *MIN*, *MAX*, *AVG*, etc.) for each field.

<i>AVG</i>	Returns the average of the values in a group
<i>BIT_AND</i>	Returns the bitwise AND of all bits in the expression.
<i>BIT_OR</i>	Returns the bitwise OR of all bits in the expression.
<i>COUNT</i>	Returns the total number of items in a column. This function does not ignore NULL values when calculating results.
<i>GROUP_CONCAT</i>	Returns a string result with the concatenated non-NULL values from a group.
<i>MAX</i>	Returns the maximum value for the column.
<i>MIN</i>	Returns the minimum value for the column.
<i>STD</i>	Returns the population standard deviation of the expression.
<i>STDDEV</i>	Returns the sample standard deviation of a numeric expression evaluated over a set.
<i>SUM</i>	Returns the sum of all the values in the expression.
<i>VARIANCE</i>	Returns the population standard variance of the expression.

To remove the field from the list, select the [Delete current row](#) item from the popup menu of the field row.

To modify the input query field, double-click it and then type the field name or select one from the drop-down list.

To modify the output query field name, double-click it and enter the field name.

DISTINCT option

To specify removal of duplicate rows from the result set, open the [Selection](#) tab and check the [Select only unique records](#) box.

Add HAVING statement

Set the conditions to be included into the HAVING statement within the [Grouping Criteria](#) tab. They are set in the same way as the *WHERE* conditions. To set the aggregate function for the field, double-click the field row in the [Aggregate](#) column and then type the function name or select one from the drop-down list.

ORDER BY clause

Set the way of sorting the query records within the [Sorting](#) tab. The field list on the left represents all the output query fields; the list on the right contains fields by which the query records will be sorted. To move the field from one list to another, drag the selected field or use the [Add](#) and [Remove](#) buttons. To change the sorting order, select a field in the right list and move it using the [Up](#) and [Down](#) buttons.

To change the sorting direction, select a field in the right list and switch the direction (*Ascending, Descending*) using the [A..Z/Z..A](#) button.

■ Create UNIONS

To combine the result from multiple `SELECT` statements into a single result set, use the [Add union](#) link of the Query Explorer popup menu.

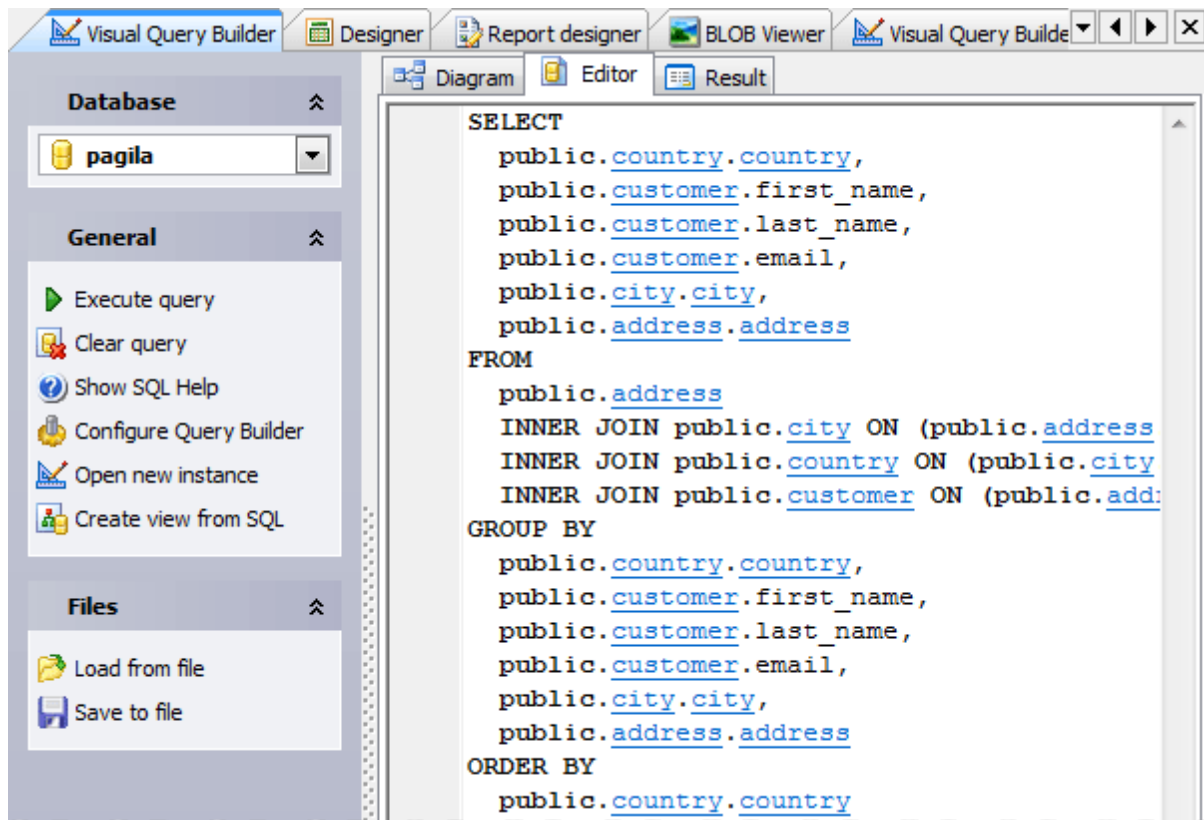
Note: The column names from the first `SELECT` statement are used as the column names for the results returned.

Selected columns listed in corresponding positions of each `SELECT` statement should have the same data type.

7.2.2 Working with editor area

In the [Editor](#) area the query text is automatically generated while you are building the query.

You can edit this text according to the rules of SQL, and all the changes will be displayed on the [Diagram](#) page of [Visual Query Builder](#).



7.2.3 Executing query

To execute the query select the [Execute](#) item in the navigation bar. After that the [Result](#) tab is displayed. This page contains the result data returned by the query, as a grid (see [Data View](#) for details). The popup menu of this tab and the items of the navigation bar allow you to export data and get SQL dump.

Visual Query Builder Designer Report designer BLOB Viewer Visual Query Builder

Database: pagila

General

- Show SQL Help
- Configure Query Builder
- Open new instance
- Create view from SQL

Data Management

- Export data
- Get SQL dump
- Print data

Diagram Editor Result

Table

* country

	* first_name	* last_name	email
	NULL	NULL	NULL
country : United Kingdom (9)			
1	ANNE	POWELL	ANNE.POWELL@sakilacustomer.org
2	APRIL	BURNS	APRIL.BURNS@sakilacustomer.org
3	ARMANDO	GRUBER	ARMANDO.GRUBER@sakilacustomer.org
4	CECIL	VINES	CECIL.VINES@sakilacustomer.org
5	DAN	PAINE	DAN.PAINE@sakilacustomer.org
6	GILBERT	SLEDGE	GILBERT.SLEDGE@sakilacustomer.org
7	MARSHALL	THORN	MARSHALL.THORN@sakilacustomer.org
8	MATTIE	HOFFMAN	MATTIE.HOFFMAN@sakilacustomer.org
9	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustomer.org
country : United States (36)			
1	ALICE	STEWART	ALICE.STEWART@sakilacustomer.org
2	ANA	BRADLEY	ANA.BRADLEY@sakilacustomer.org
3	ASHLEY	RICHARDSON	ASHLEY.RICHARDSON@sakilacustomer.org
4	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org
5	BILL	GAVIN	BILL.GAVIN@sakilacustomer.org
6	BRANDY	GRAVES	BRANDY.GRAVES@sakilacustomer.org
7	BRYAN	HARDISON	BRYAN.HARDISON@sakilacustomer.org
8	CAROLE	BARNETT	CAROLE.BARNETT@sakilacustomer.org
9	CAROLINE	BOWMAN	CAROLINE.BOWMAN@sakilacustomer.org
10	CASSANDRA	WALTERS	CASSANDRA.WALTERS@sakilacustomer.org
11	CLINTON	BUFORD	CLINTON.BUFORD@sakilacustomer.org
12	DIANA	ALEXANDER	DIANA.ALEXANDER@sakilacustomer.org
13	EVA	RAMOS	EVA.RAMOS@sakilacustomer.org
14	IAN	STILL	IAN.STILL@sakilacustomer.org

Records fetched: 599

Information

599 rows fetched (0,19 sec)

Database: pagila

8 Data Management

Query results and table data are displayed on the [Data](#)^[79] or [Result](#)^[216] tabs of [Table Editor](#)^[77], [SQL Editor](#)^[214] or [Visual Query Builder](#)^[219].

Data are displayed as a grid (or as info cards) which provide a lot of useful features such as editing, grouping, sorting, filtering, etc. See [Data View](#)^[229] for details.

Navigation bars of these tabs as well as popup menus of their working areas places at your disposal the following functions for managing data:

- [Export Data](#)^[245] allows you to export data to various formats, including MS Excel, MS Access, RTF, HTML, PDF and more.
- [Get SQL Dump](#)^[252] exports data to the SQL script as a number of INSERT statements.
- [Import Data](#)^[255] provides you with possibility to import data from MS Excel, MS Access, DBF, XML, TXT, and CSV.
- [Edit BLOB](#)^[240] allows you to view and edit the content of BLOB and TEXT fields.

8.1 Data View

PostgreSQL Maestro represents all data (stored in tables and views, results of queries and procedures) in [grid](#)^[230] or in [info_cards](#)^[236]. By default, data is displayed in a grid - tabular view of data. To change the type of the data representation, use the drop-down list at the top of the tab. Both of the data representations support UNICODE/UTF-8 data. The status bar displays the number of records in the current data set. To reset grid to default settings, open the Data tab when holding the **Ctrl** key.

Drag a column header here to group by that column

	CUST_NO	CUSTOMER	CONTACT_FIRST	CONTACT_LAST	PHONE_NO	AD
1	1001	Signature Design	Dale J.	Little	(619) 530-2710	15
2	1002	Dallas Technologies	Glen	Brown	(214) 960-2233	P.
3	1003	Buttle, Griffith and Co.	James	Buttle	(617) 488-1864	230
4	1004	Central Bank	Elizabeth	Brocket	61 211 99 88	66
5	1005	DT Systems, LTD.	Tai	Wu	(852) 850 43 98	400
6	1006	DataServe International	Tomas	Bright	(613) 229 3323	200
7	1007	Mrs. Beauvais	NULL	Mrs. Beauvais	NULL	P.C
8	1008	Anini Vacation Rentals	Leilani	Briggs	(808) 835-7605	33
9	1009	Max	Max	NULL	22 01 23	1 E
10	1010	MPM Corporation	Miwako	Miyamoto	3 880 77 19	2-6
11	1011	Dynamic Intelligence Corp	Victor	Granges	01 221 16 50	Flo
12	1012	3D-Pad Corp.	Michelle	Roche	1 43 60 61	22
13	1013	Lorenzi Export, Ltd.	Andreas	Lorenzi	02 404 6284	Via

Records fetched: 15/15 LIMIT 1000 OFFSET 0

Navigation buttons

Both data representations are equipped with navigation buttons. They are represented at the top of the data tab and allow you to navigate between records and to accomplish common operations:

- To add a new record, use the *Plus* button or the **Insert** shortcut.
- To delete a new record, use the *Minus* button or the **Delete** shortcut.
- To edit an existing record, push the corresponding button or invoke the [Data Input Form](#)^[236] using popup menu of the necessary record, with **Ctrl+Alt+D** shortcut, or with the corresponding link at the Navigation bar. To edit a field value, click it and enter the new one inline.

The pagination option allows you to limit the number of browsed records. By default, the number of records represented in grid at once is 1000. To change the number of records represented in the current grid, enter the necessary value in the pagination bar. To specify the default one or to disable pagination, use the [data grid option](#)^[337].

Navigation bar

The Data management group of the Navigation bar allows to invoke [Data Input Form](#)^[236], [SQL Editor](#)^[214] with SELECT query, [Data Export](#)^[245], and [Data Import](#)^[255] modules using corresponding links, also get [SQLdump](#)^[252] of the current data set and print current data with enabled preview in WYSIWYG mode.

See also: [Table Editor](#)^[77], [SQL Editor](#)^[214], [Visual Query Builder](#)^[219]

8.1.1 Working with data grid

Our software offers two grid modes:

- the full grid mode is a fully functional data representation equipped with abilities to filter and to sort data;
- the simple grid mode is provided for working with large number of records. For speed-up data fetching, filtering and sorting abilities are not enabled in this mode. The notification bar at the top of the grid (see the picture below) announces that the grid has been switched to the simple mode.

Result 1 Result 2 Result 3

Table

The grid has been switched to the simple mode because of the query returned more than 4000 rows (you can customize this number in the [Options](#) dialog). Filtering, sorting and grouping features are not enabled in this mode.

Other actions:
[Switch to full mode now](#) | [Always use full mode](#) | [Dismiss this message](#)

CNO	TITLE	FIRSTNAME	NAME	ZIP	ADDRESS
3000	Mrs	Jenny	Porter	10580	1340 N. Ash Street, #3
3100	Mr	Peter	Brown	48226	1001 34th St., APT.3
3200	Company	NULL	Datasoft	90018	486 Maple St.
3300	Mrs	Rose	Brian	75243	500 Yellowstone Drive, #2
3400	Mrs	Mary	Griffith	20005	3401 Elder Lane
3500	Mr	Martin	Randolph	60615	340 MAIN STREET, #7
3600	Mrs	Sally	Smith	75243	250 Curtis Street
3700	Mr	Mike	Jackson	45211	133 BROADWAY APT. 1
3800	Mrs	Rita	Doe	97213	2000 Humboldt St., #6
3900	Mr	George	Howe	75243	111 B Parkway, #23
4000	Mr	Frank	Miller	95054	27 5th St., 76
4100	Mrs	Susan	Baker	90018	200 MAIN STREET, #94
4200	Mr	Joseph	Peters	92714	700 S. Ash St., APT.12
4300	Company	NULL	TOOLware	20019	410 Mariposa St., #10
4400	Mr	Antony	Jenkins	20903	55 A Parkway, #15
4401	Company	NULL	MagicStrawberry	78146	76 Highland Road, #120
4402	Company	NULL	OrangeHand	78609	212 Oak Avenue, #30

Records fetched: 4495

Information
 4495 rows fetched (2,00 sec)

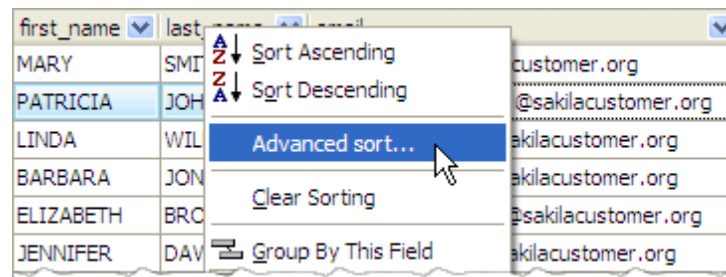
By default, the grid automatically switches to the simple mode for queries returning more than 5000 records (the number can be customized in the [Options](#) ³³⁷ dialog).

The following abilities are not available in the simple grid mode:

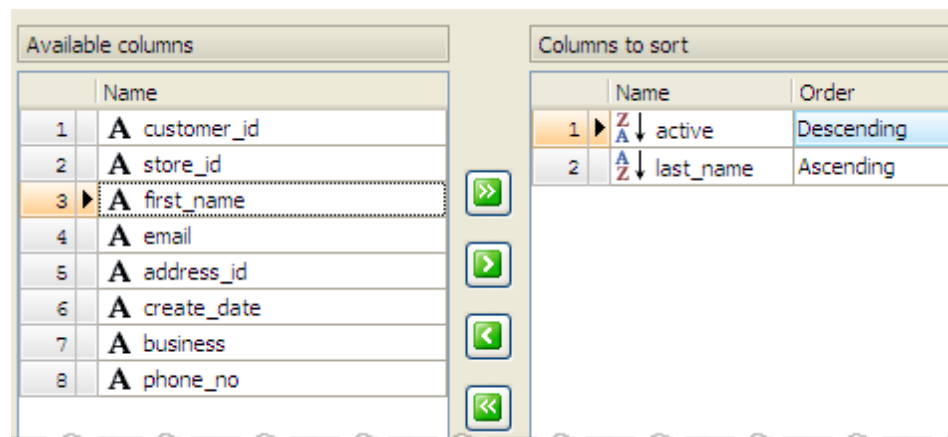
■ **Sorting data (only in the full grid mode)**

Click the column caption to sort data by the values of this column. To select sort order (ascending or descending), use popup menu of the column caption.

To sort data on a combination of grid columns, use the [Advanced sort...](#) link of the popup menu of the grid's header. The [Advanced sorting](#) window will be shown.



Select there the columns you want to sort from the [Available columns](#) list in the order of priority. Specify the sort order if necessary and click OK.



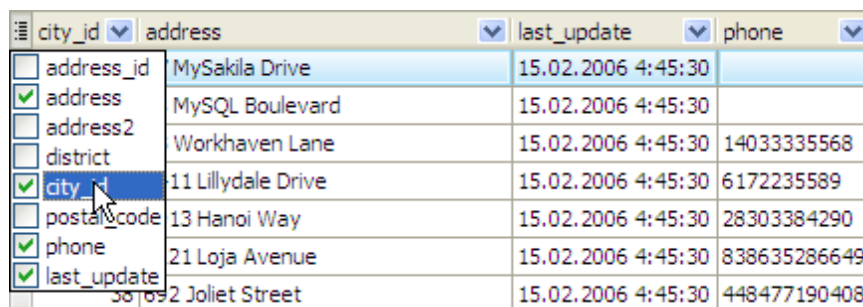
To cancel the sorting order, press **Ctrl** and click on the sorted column caption.

■ Filtering represented records (only in the full grid mode)

There are several ways to filter data represented in grid. See [the corresponding topic](#)^[237] to find out their descriptions.

■ Hiding selected columns

You can show/hide columns using a button in the left top corner of the grid. Just check/uncheck the column in the drop-down list.



■ Columns reordering

To reorder columns, use drag-n-drop.

■ Grouping records

You can group grid data by any of the columns by dragging the column header to the destination area. Now all the records are displayed as subnodes to the grouping row value as shown in the picture. To reverse grouping, just drag the column name from the upper area back.

The screenshot shows a data grid application with a toolbar at the top containing navigation and action icons. Below the toolbar, there are tabs for 'Result 1', 'Result 2', and 'Result 3'. The main area displays a table with columns: ID, team1ID, team2ID, score1, score2, refereeID, and comments. The data is grouped by 'round' and 'date'. The 'round' column has values 1, 2, 3, 4, 5, and 6. The 'date' column has values 24.08.2004, 25.08.2004, 30.08.2004, 14.12.2004, 11.09.2004, 12.09.2004, and 13.09.2004. The table shows records for each date, with some records highlighted in blue. At the bottom, a status bar indicates 'Records fetched: 380'. An 'Information' popup at the bottom left shows '380 rows fetched (0,64 sec)'.

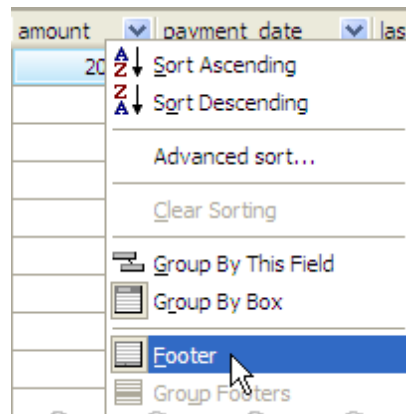
ID	team1ID	team2ID	score1	score2	refereeID	comments
round : 1						
round : 2						
round : 3						
date : 24.08.2004						
date : 25.08.2004						
24	6	2	3	0	8	Jeffers brukte 34 minutter på å vinne
22	1	4	3	0	17	I den hundrede kampen i alle konkurranse
29	16	5	1	2	19	Southamptons slapp inn mål i sin 11. li
27	9	13	0	2	0	Fulhams første tap denne sesongen,
28	14	18	2	2	18	Newcastle skuffer i årets Premier League
26	19	17	1	1	12	Det ble uavgjort på Hawthornes etter
date : 30.08.2004						
31	12	8	0	0	1	Igjen skuffet Manchester United mot
date : 14.12.2004						
round : 4						
round : 5						
date : 11.09.2004						
date : 12.09.2004						
date : 13.09.2004						
round : 6						

Records fetched: 380

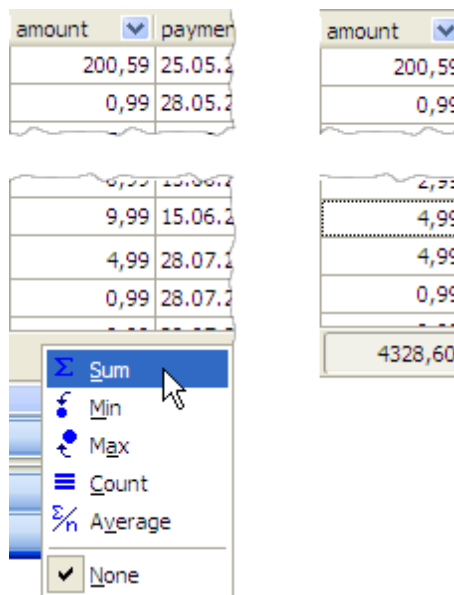
Information
380 rows fetched (0,64 sec)

■ Using aggregate functions

To get a sum of column values, a min or a max value, an average column value or an amount of records, use **Data Grid Footer**. Select the **Footer** item at the grid caption's popup menu.



It will be shown at the bottom of the grid. The popup menu of the footer allows you to get an aggregate function result calculated with the corresponding column values.



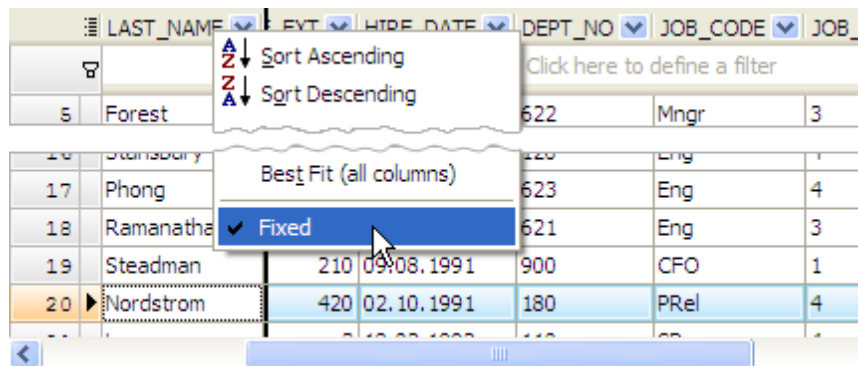
For grouped data use [Group Footers](#).

■ Data alignment

The grid's header popup menu allows to align column data. Use the [Alignment](#) link and select the alignment type.

■ Fixing columns

You can fix grid columns to view them permanently when working with other grid data. To fix a column, choose the corresponding item from the grid's header popup menu.

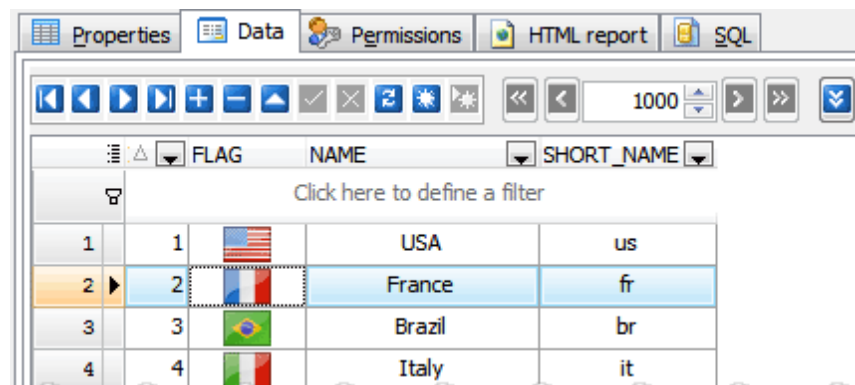


■ Row numbering

There is also a possibility to display row numbers in grids. You can [adjust](#)^[339] the corresponding column to yours liking.

■ Inline images

It is possible to display images directly in the grid as on the picture below.



To enable/disable this view mode, open the *Enable inline images* window using the *Manage inline images* item of the column popup menu. The window options allow to set or change the image fitting and specify the row height. To add new images or change existing ones, use [BLOB Editor](#)^[240] (see below).

■ Working with BLOBs

To [edit a BLOB field](#)^[240], double click the field, or use the corresponding popup menu item. There are also possibilities to export all BLOBs stored in the table column to files and import BLOBs from a directory to the table columns. In this case you need to set the Target directory, specify the template to be used for file names and the column BLOBs to be exported from (imported to).

8.1.2 Working with info cards

Info cards correspond to the records. You can [filter records by custom conditions](#)^[237] and edit data directly in info cards or with [Data Input Form](#)^[236].

The screenshot displays the PostgreSQL Maestro Data Input Form, which is a window for entering or editing data. The window has a toolbar at the top with various icons for navigation and editing. The main area is divided into six panels, each representing a record. Each panel contains a list of attributes and their corresponding values. The attributes are: id, first_name, last_name, career_start_year, career_end_year, position_id, photo, country_id, height, birthday, weight, college_id, current_team_id, and current_number. The records are for players with IDs 18, 20, 21, 19, 21, and 21. The bottom status bar indicates that 67 records were fetched.

id:	first_name:	last_name:	career_start_year:	career_end_year:	position_id:	photo:	country_id:	height:	birthday:	weight:	college_id:	current_team_id:	current_number:
18	Gilbert	Arenas	2001	0	6		1	193	06.01.1982	97,5	15	12	0
20	Hilton	Armstrong	2006	0	11		1	211	11.11.1984	106,6	7	27	12
	Darrell	Arthur							25.03.198				
19	Trevor	Ariza	2004	0	10		1	203	30.06.1985	95,3	2	5	3
21	Ron	Artest	1999	0	10		1	201	13.11.1979	117,9	16	22	96
	D.J.	Augustin							10.11.198				

Records fetched: 67

8.1.3 Data input form

Use [Data Input Form](#) to add new records or edit existing ones. To invoke the dialog, use the corresponding link from the popup menu or **Ctrl+Alt+D** shortcut.

language_id	* name	* last_update
1	English	15.02.2006 10
2	Italian	15.02.2006 10
3	Japanese	15.02.2006 10
4	Mandarin	15.02.2006 10
5	French	15.02.2006 10
6	German	15.02.2006 10

The dialog's fields contain the values of the current grid row. Use the **Insert** button to enter values of a new record and the **Post** button to update the current row. The **Cancel** button reverts all the field values within a form to their initial values (or to the last posted values). The **Previous** and **Next** buttons allow you to switch between grid records without closing the dialog.

Controls containing values of primary and foreign key columns are marked with the 'gold key' and 'silver key' images accordingly. Controls containing values of required (NOT NULL) columns are marked with a red asterisk.

There are possibilities to use lookup editors on working with columns linked with foreign keys, a calendar for *timestamp* columns and a calculator for *decimal* ones.

8.1.4 Data filtering

PostgreSQL Maestro support filtering records by the following methods:

- **Filter by a column value**

Select the **Use as Filter** item from the field popup menu to filter records by the current column value.

- **Filter by several column values**

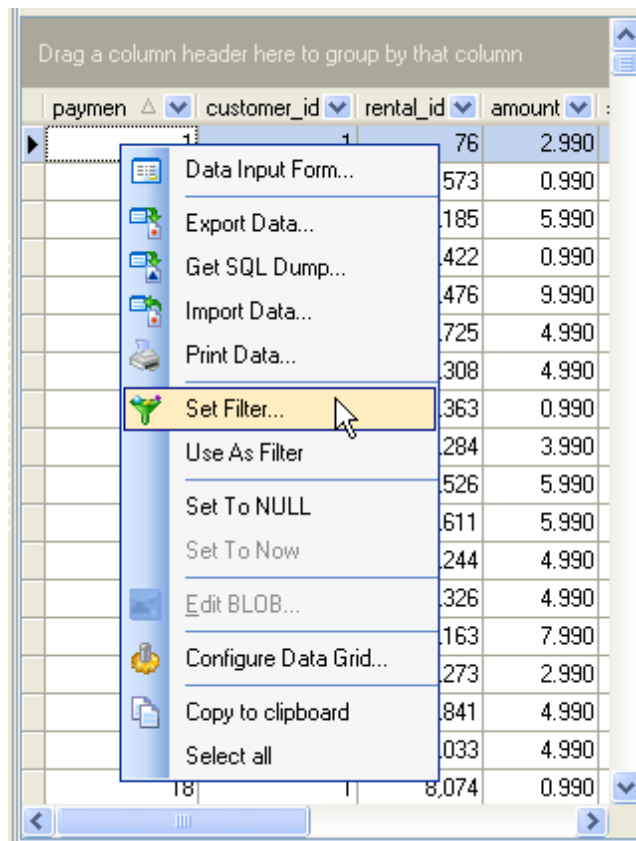
Use the drop-down button in the column caption area to filter records by the selected column value(s) or enter the filter condition directly in [the filter row](#)³⁴².

Filter by two operators

Invoke simple filter dialog using the [Custom](#) item of the column caption area drop-down list. Select a logical operator for checking the column values (like "is less than", "is greater than", etc.) and set the value to be checked by this operator in the next box; then set the second condition if necessary in the following way and set the relation between these two conditions, whether both of them should be matched or just one of them; use the '_' character to represent any single symbol in the condition and the '%' character to represent any series of symbols in the condition.

Filter by any custom criteria

To filter data according to more difficult custom conditions, use the Filter Builder dialog. To invoke the dialog, use the [Set Filter](#) link of popup menu or click the [Customize](#) button on the [Filter](#) panel. This panel is visible if any filtering is already applied to the grid (you can use column header menu or grid menu for quick filtering).



The dialog also allows to save filter criteria to an external file for future use.

After you set a filter, the filtering panel becomes visible at the top/bottom of the grid

where you can see the active filtering condition and easily enable or disable it by clicking the check box on the left. To customize the filtering process, use [filter options](#)^[342].

The [Copy current filter as SQL condition to clipboard](#) feature is useful in case the same compound filter is applied several times. Just once apply the filter, copy to clipboard as SQL condition, paste to [SQL Editor](#)^[214] and save as a query. You can also use [Generate query](#) link on the Navigation bar.

See also: [Data View](#)^[229], [Table Editor](#)^[77], [SQL Editor](#)^[214], [Visual Query Builder](#)^[219]

8.2 BLOB Editor

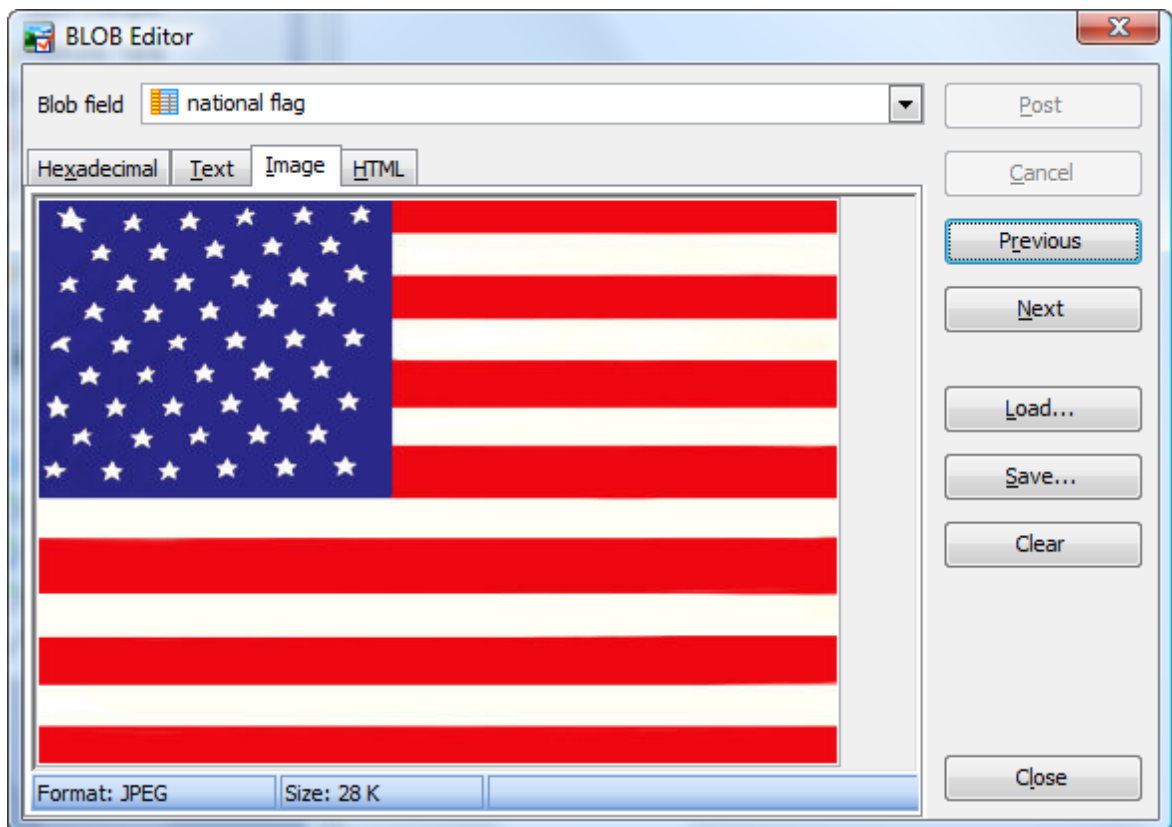
BLOB Editor is a tool to view and edit BLOB data in the following ways: [hexadecimal dump](#)^[241], [plain text](#)^[241], [graphical image](#)^[240], [HTML page](#)^[242], or [PDF document](#)^[243]. BLOB Editor is invoked from [data grid](#)^[229] of any [table editor](#)^[77] or the result tab of [SQL Editor](#)^[214] and [Visual Query Builder](#)^[219] by double clicking of the BLOB field to be edited or with the Edit BLOB link of the field's popup menu. The editor also can be called from [BLOB Viewer](#)^[276] with the Edit current BLOB button.

With BLOB Editor you can work with all BLOB columns of the grid. To switch between columns, select the necessary one from the [BLOB field](#) list.

BLOB Editor allows you to navigate between the grid records using the [Previous](#) and [Next](#) buttons. You can load the new BLOB content and save or clear it using corresponding buttons. After changes are made, click the Post button to apply the changes or the [Cancel](#) button to discard them.

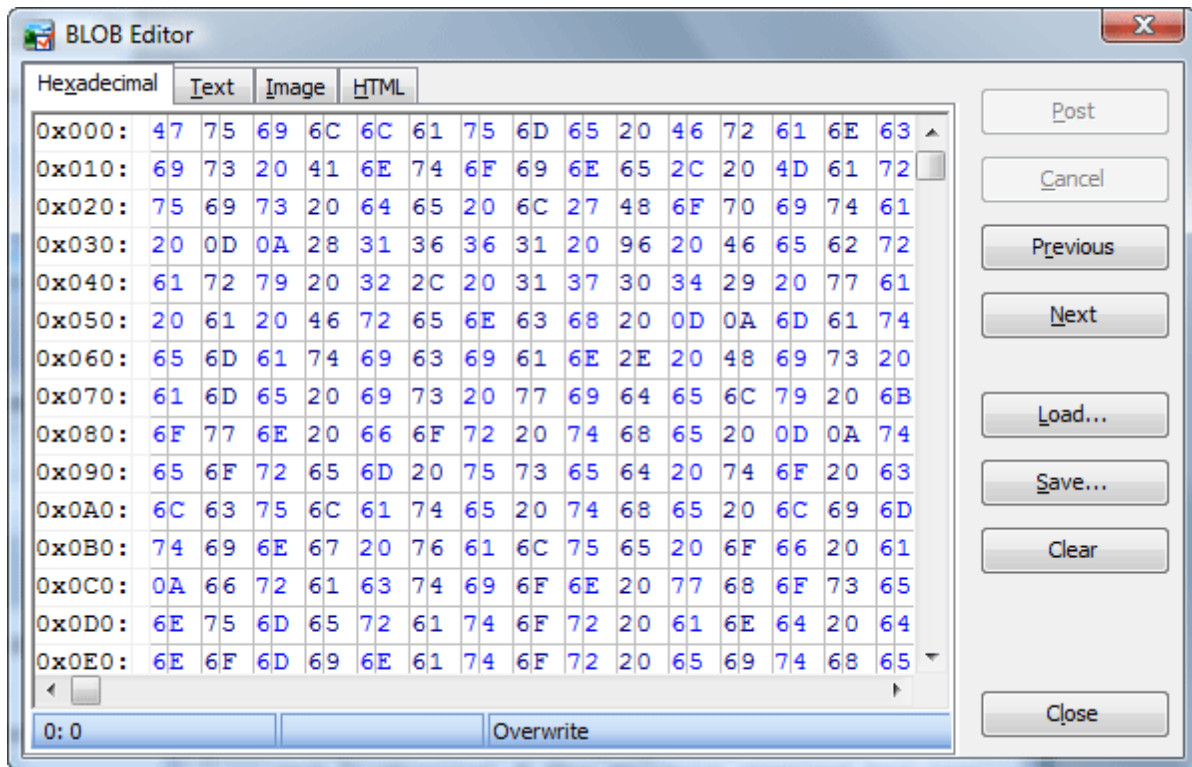
8.2.1 Editing as image

The [Image](#) panel of BLOB Editor displays field data as graphical image. Use the Save and Load buttons to save the image to a file or load an image from a file. A graphical representation of BLOB data supports five image formats: BMP, Windows metafile, JPEG, GIF and PNG.



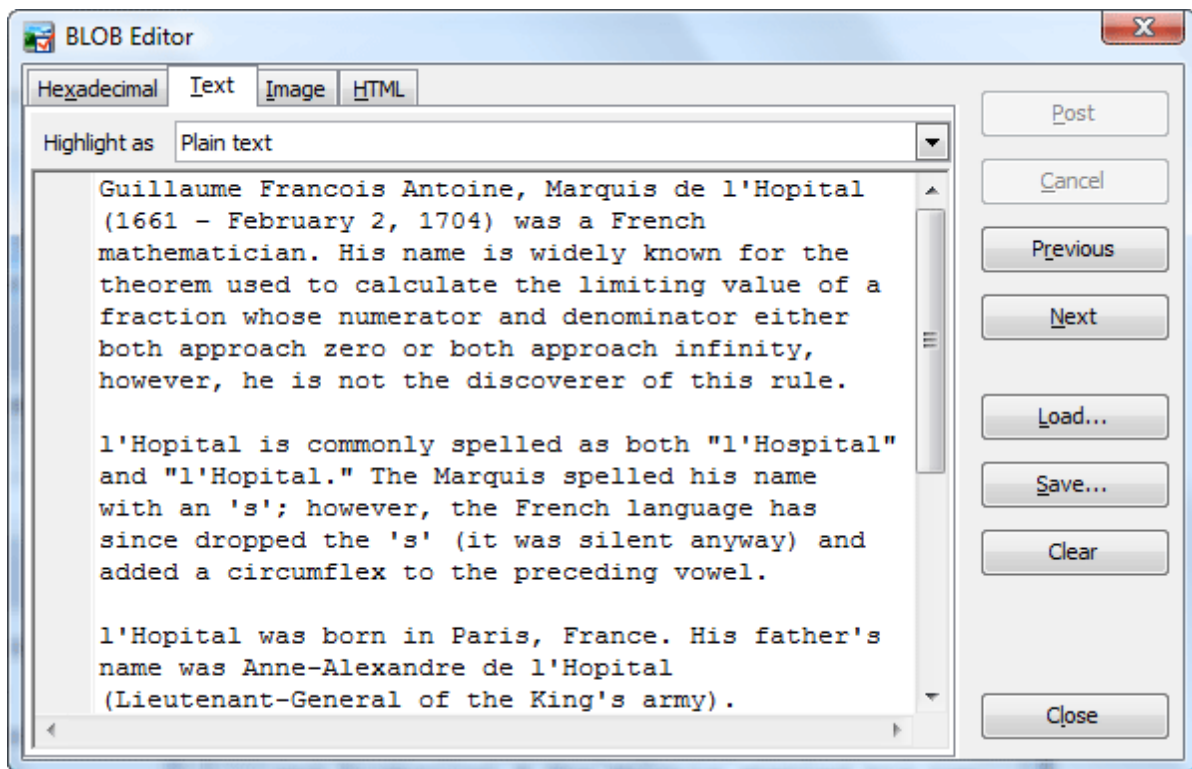
8.2.2 Editing as hexadecimal dump

The [Hexadecimal](#) panel allows you to edit data in hexadecimal mode. To load/save a hexadecimal dump from/to a file, use the corresponding buttons. Use the Insert key to switch between Insert and Overwrite modes.



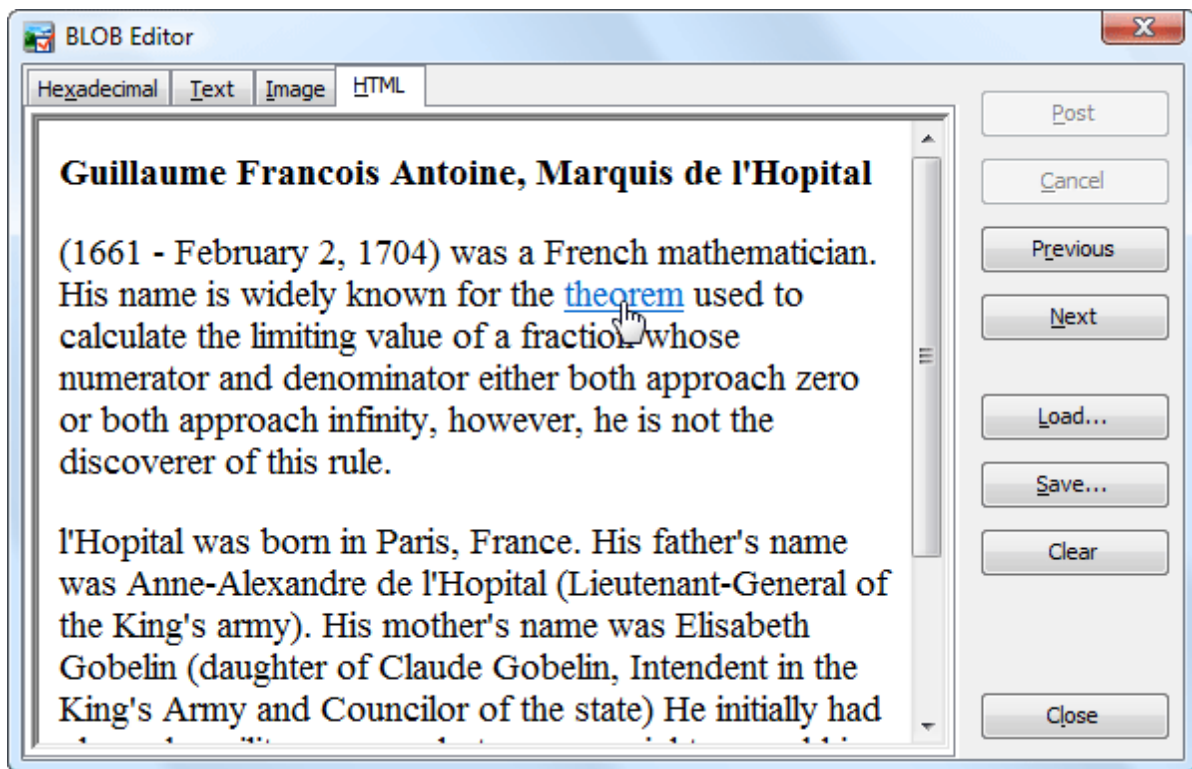
8.2.3 Editing as plain text

The [Text](#) panel allows you to edit data as a simple text. Several types of text highlighting are available (*Plain text*, *SQL*, *XML*, *Java*, *VBScript*, *JScript*, *Cmd batch*, *PHP*, *CSS*, *UnixShell Script*, *INI*, and *HTML*). The popup menu of the panel allows you to invoke [Find Text](#), [Replace Text](#) and [Go to line](#) dialogs.



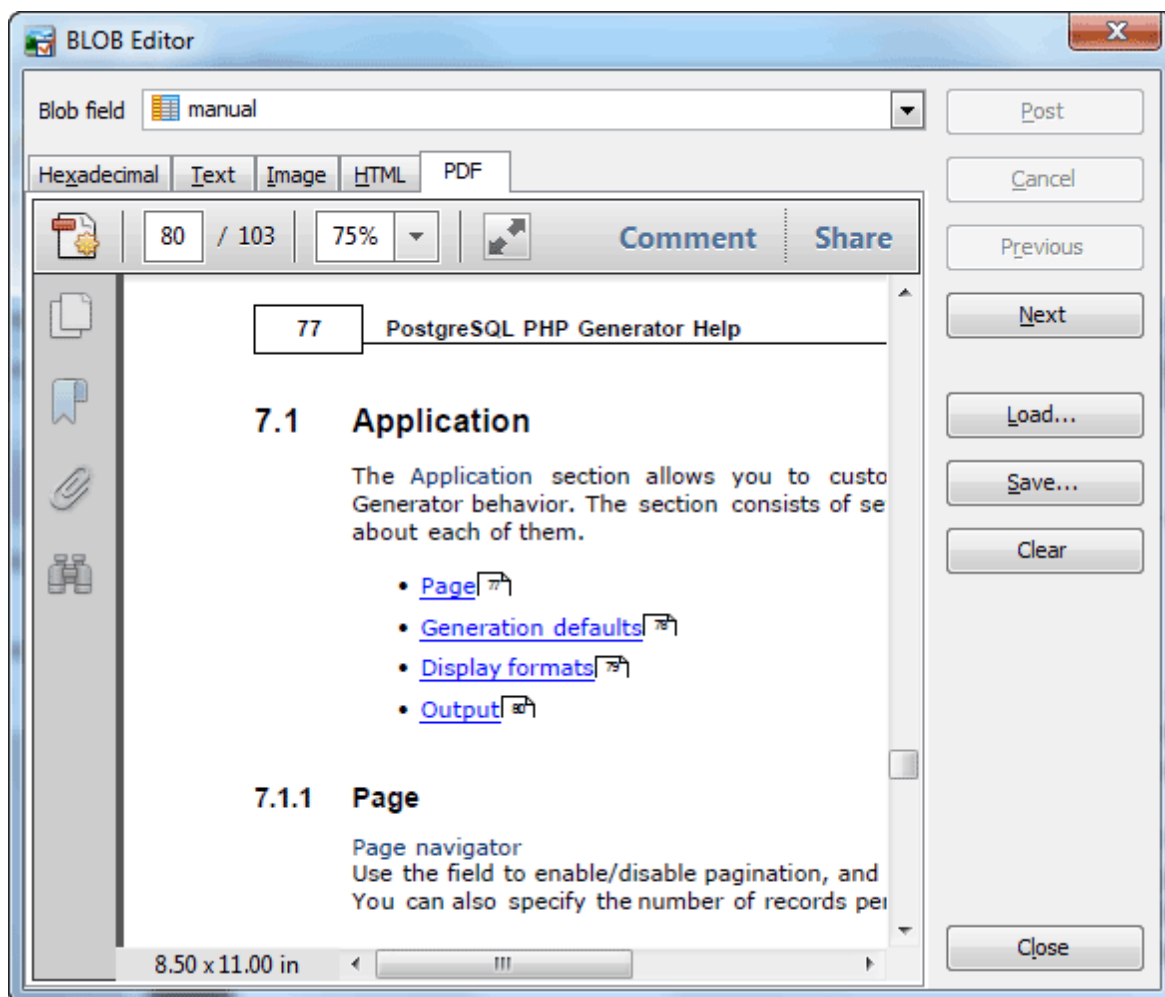
8.2.4 Editing as HTML

The [HTML](#) panel presents field data as HTML. You can load a new content of the current field from a .html file or type it manually within the [Text](#) tab of the editor.



8.2.5 Editing as PDF document

The PDF panel presents field data as PDF document. To accomplish common operations with data, use the Adobe Reader toolbar.



8.3 Export Data Wizard

Data Export wizard is a tool to save data from PostgreSQL tables, views, and queries to the most popular formats. It allows you to fully customize output files including header and footer, fonts, colors, and data formats.

Export Data tool supports:

- Microsoft Office Excel 97-2003, 2007
- CSV
- HTML
- XML
- Text
- Microsoft Office Word 97-2003, 2007
- Microsoft Office Access
- OpenDocument Spreadsheet
- OpenDocument Text
- DBF
- PDF
- RTF
- DIF
- SYLK
- LaTeX.

In order to run the wizard you should either

- open the table in [Table Editor](#);
- go on to the [Data](#) tab

or

- open and execute the query in [SQL Editor](#) or [Query Builder](#);
- proceed to the [Result](#) tab

and select the [Export Data](#) item from the [Navigation Bar](#).

To export your data,

- [Set the format and the name](#)^[245] of the destination file;
- Specify such additional options of the result file as [header and footer](#)^[246], [formats applied to exported data](#)^[247] and [some format-specific options](#)^[248];
- [Select columns](#)^[247] you want to include into result files;
- [Specify other export options](#)^[251].

See also: Get SQL Dump, [Import Data Wizard](#)^[255]

8.3.1 Setting destination file name and format

Select one of the available destination formats and set the name for the result file. The file name extension in the [Destination file name](#) box varies according to the selected export type.

The file name may contain current timestamp with the %ts:TIMESTAMP_FORMAT% string. Examples of valid log file names:

dbname_export_%ts:yyyy_mm_dd%.log
export_%ts:yyyy_mm_dd_hh_mm%.log
%ts:yyyy_mm_dd_hh_mm_ss%.log

Destination format

Select one of the available destination formats.

☒ Microsoft Office Excel 97 - 2003

☐ Microsoft Office Excel 2007 - 2010

☐ Delimiter-separated values (CSV, DSV, TSV)

☐ Text file (Fixed-width columns)

☐ HTML

☐ XML

☐ Other

Microsoft Office Word 97 - 2003

Destination file

Select or enter the result file name and specify the encoding if necessary. To add current timestamp to the result file name, use the %ts:TIMESTAMP_FORMAT% string (for example, %ts:yyyy_mm_dd%). Hint: you can set default directory for data export in the Edit Database Profile dialog.

File name

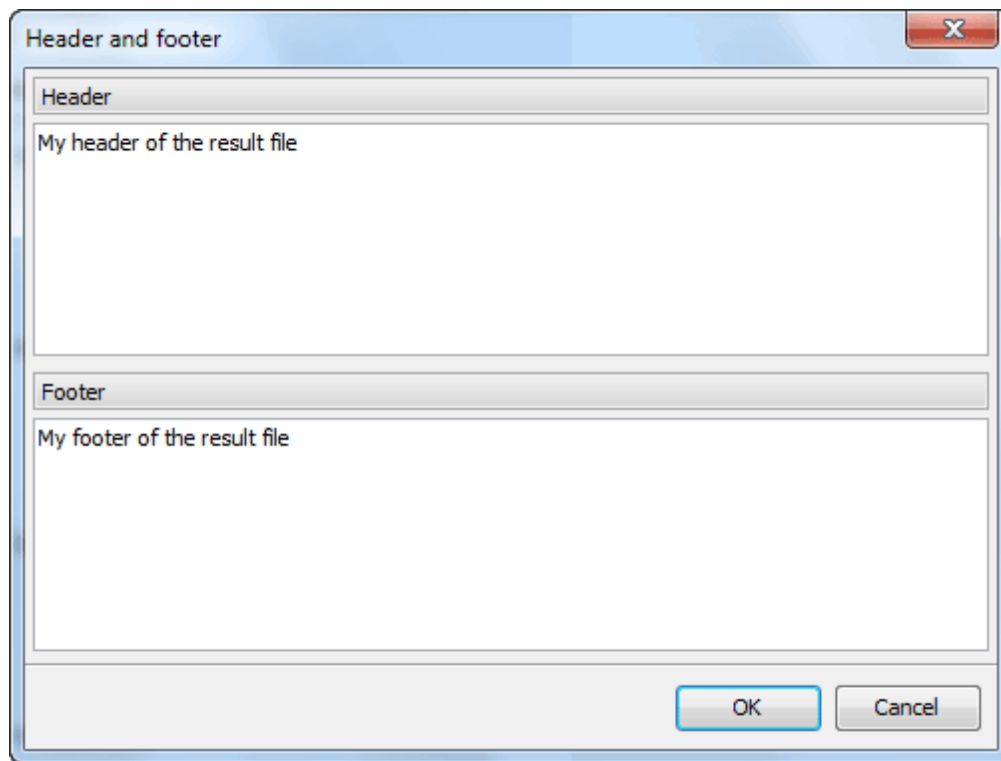
C:\Data\Excel\Customers.xls

Encoding

ANSI

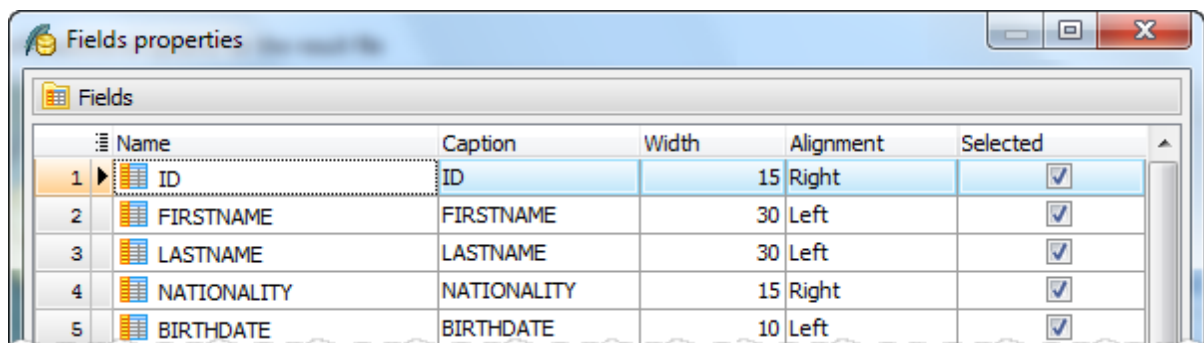
8.3.2 Setting header and footer

To specify the result file's header and footer, double click the corresponding button and complete fields of the [Header and Footer](#) window.



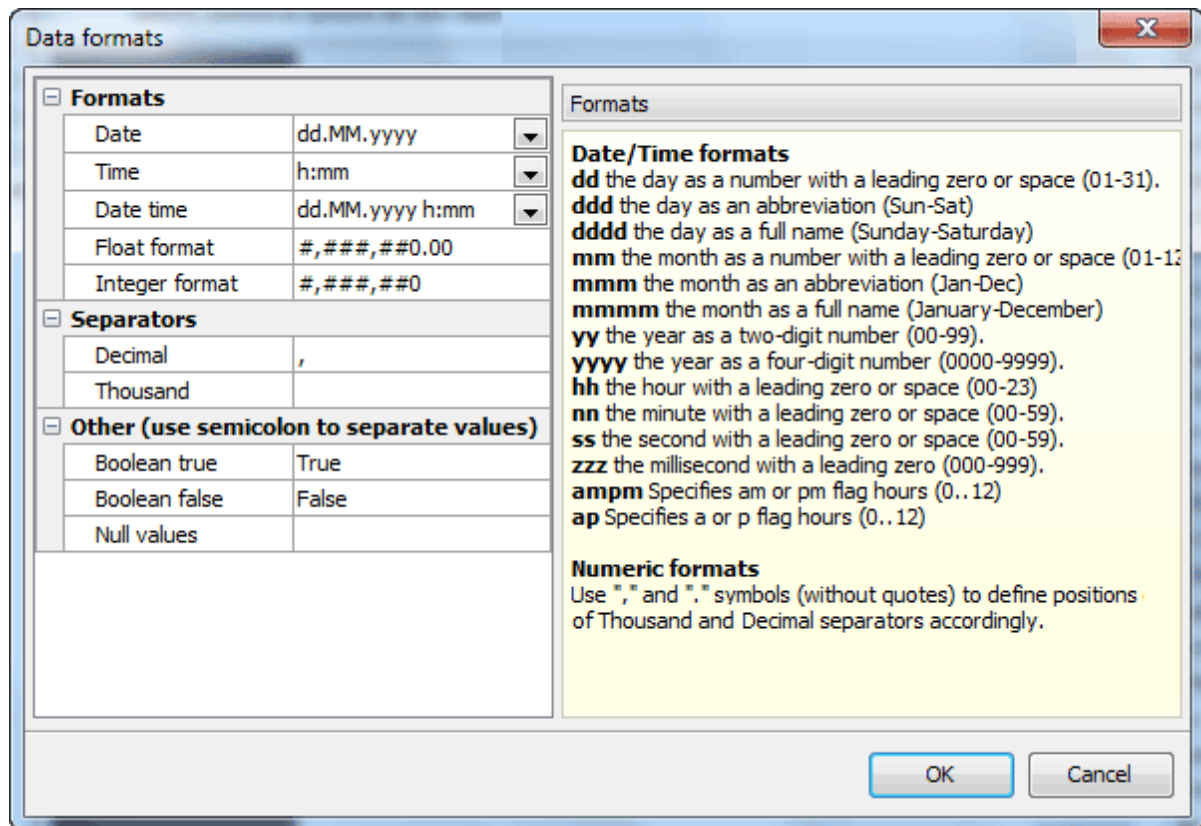
8.3.3 Selecting fields for export

Uncheck the Selected box to exclude the corresponding field from the export, specify a Caption to be used for the result column, and also width, and alignment for output columns (when applicable).



8.3.4 Adjusting data formats

This step allows you to customize formats applied to exported data. Edit the format masks to adjust the result format in the way you need.



8.3.5 Setting format-specific options

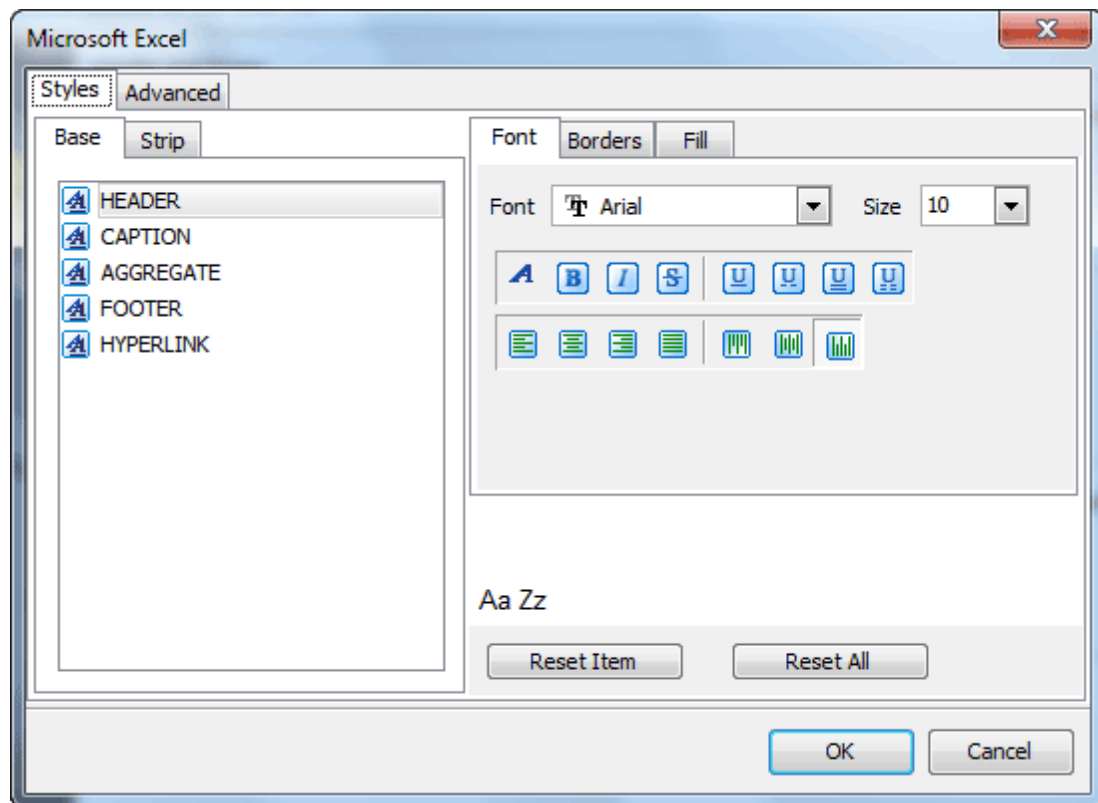
Each format supposes corresponding additional export options. Use the wizard option to adjust export properties depending on the target file format you have selected earlier. The following formats are at your disposal: [Microsoft Excel](#)^[248], Microsoft Excel 2007, [CSV](#)^[250], [Text](#)^[250], [HTML](#)^[249], [XML](#)^[250], Microsoft Word, Microsoft Word 2007, Microsoft Access, OpenDocument Spreadsheet, OpenDocument Text, DBF, PDF, RTF, DIF, SYLK, and LaTeX.

Microsoft Excel

The [Data Format](#) tab contains general options, which allow you to adjust the format for each kind of Excel cells. This means that you can specify such parameters as font, borders, filling color and method, etc. for each entity (such as data field, header, footer, caption, data, hyperlink and so on) separately. Also it is possible to create styles to make target Excel file be striped by columns or rows (the [Styles](#) tab).

The [Extensions](#) tab provides a possibility to add hyperlinks and notes to any cell of target file. Click the [Plus](#) button to add a new hyperlink or note to target Excel sheet and adjust its parameters. Click the [Minus](#) button to delete added hyperlink or note.

The [Advanced](#) tab allows you to define page header, page footer and title for target Excel sheet.



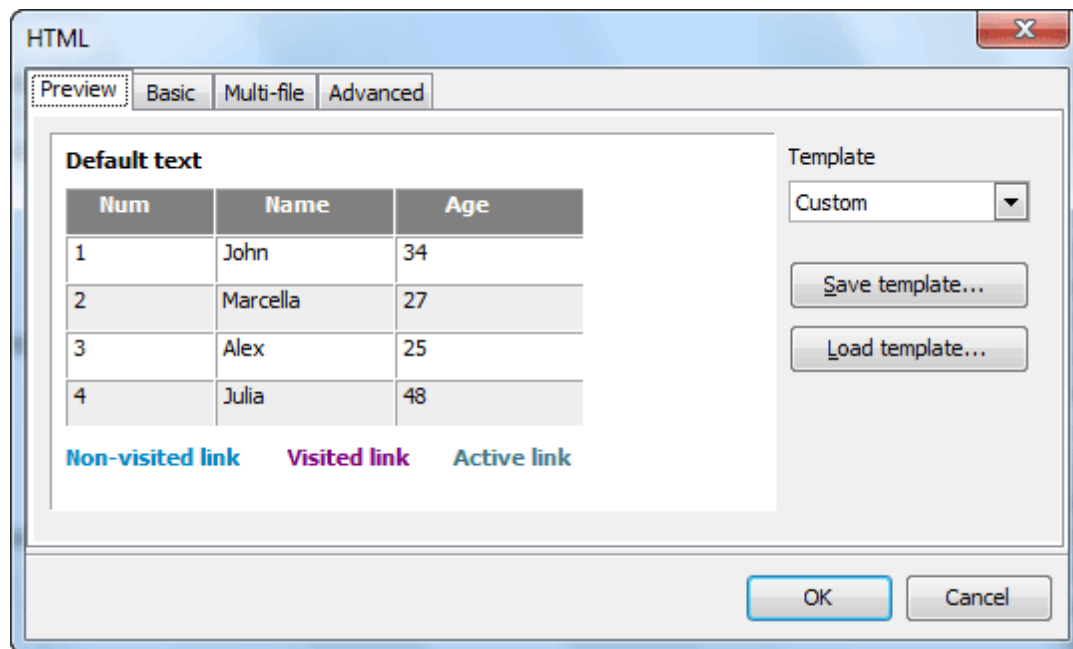
HTML

The [Preview](#) tab allows you to select the style of HTML file from a number of built-in templates provided by the [Templates](#) combo box. You can choose any of these templates, customize it by clicking on objects in the preview panel, and save it as a custom template using the [Save template](#) button. Use the [Load template](#) button to load previously saved custom templates from hard disk.

The [Basic](#) tab allows you to specify basic parameters of target HTML file, such as its title, cascade style sheet options, etc.

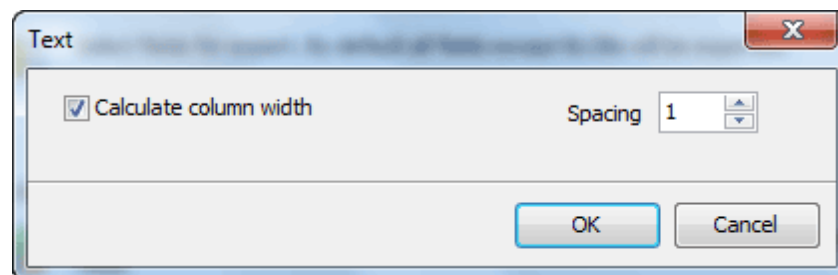
The [Multi-file](#) tab provides you with a possibility to split target HTML file into several separated files. This tab allows you to specify the record count for a single file, set an option to generate an index HTML file, and add an ability of navigation between each other to each of exported files.

The [Advanced](#) tab contains such HTML options as default font, background, cell padding and spacing, etc.



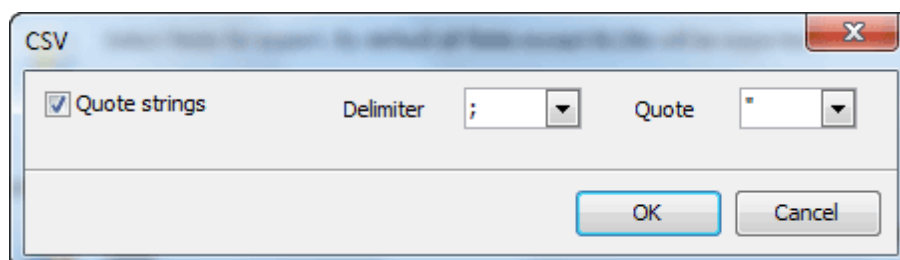
Text files

Set the [Calculate](#) column width options on if you want each column of target file to be adjusted to the maximum number of characters in it. The [Spacing](#) option specifies the number of spaces between columns in the target file.



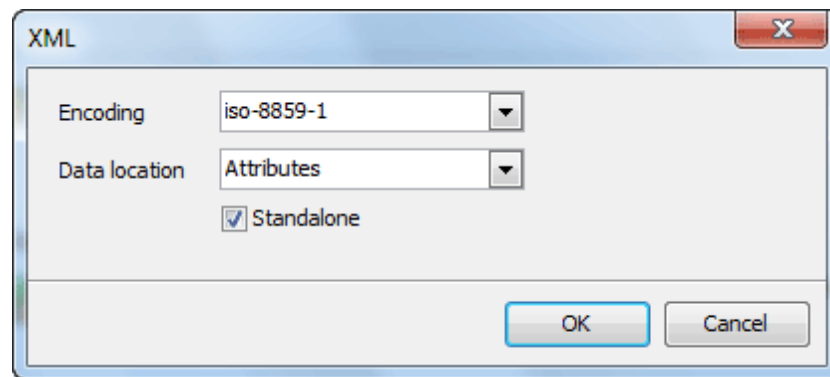
CSV files

You can specify column separator and optional values quote character for the target file on this step.



XML documents

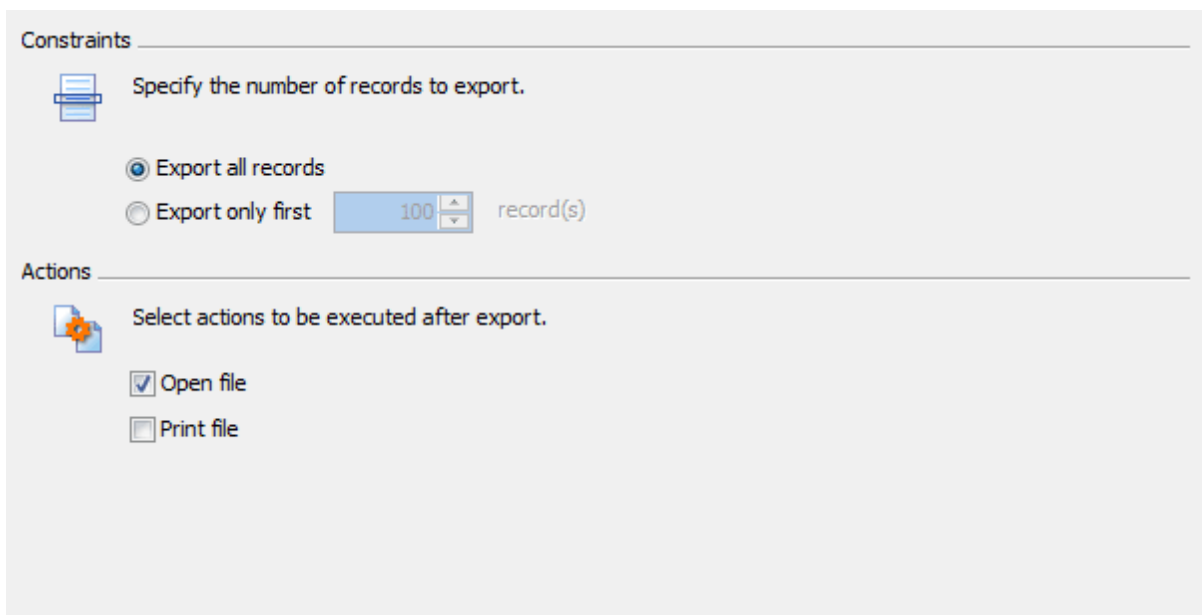
Specify XML document encoding in the [Encoding](#) edit box and set the Standalone option on if you wish the target document to be standalone.



8.3.6 Setting common export options

Use this step to specify options to be applied to all exported data:

- Select the number of records to be exported from each table: a fixed number or all records.
- Specify actions to be executed after the export. To open the result files in the associated program (MS Excel, Notepad, default browser, etc), check the [Open file](#) box. To send the result files to the default printer, use the [Print file](#) checkbox.



8.4 Get SQL Dump

[Get SQL Dump Wizard](#) allows you to export data from a table or a query result to the SQL script as a number of INSERT statements.

In order to get a SQL dump from a table or a query:

- open the table in [Table Editor](#) or open and execute query in [SQL Editor](#) or [Query Builder](#);
- open the [Data](#) tab or the [Result](#) tab respectively;
- use the [Get SQL Dump](#) item of the [Navigation Bar](#).

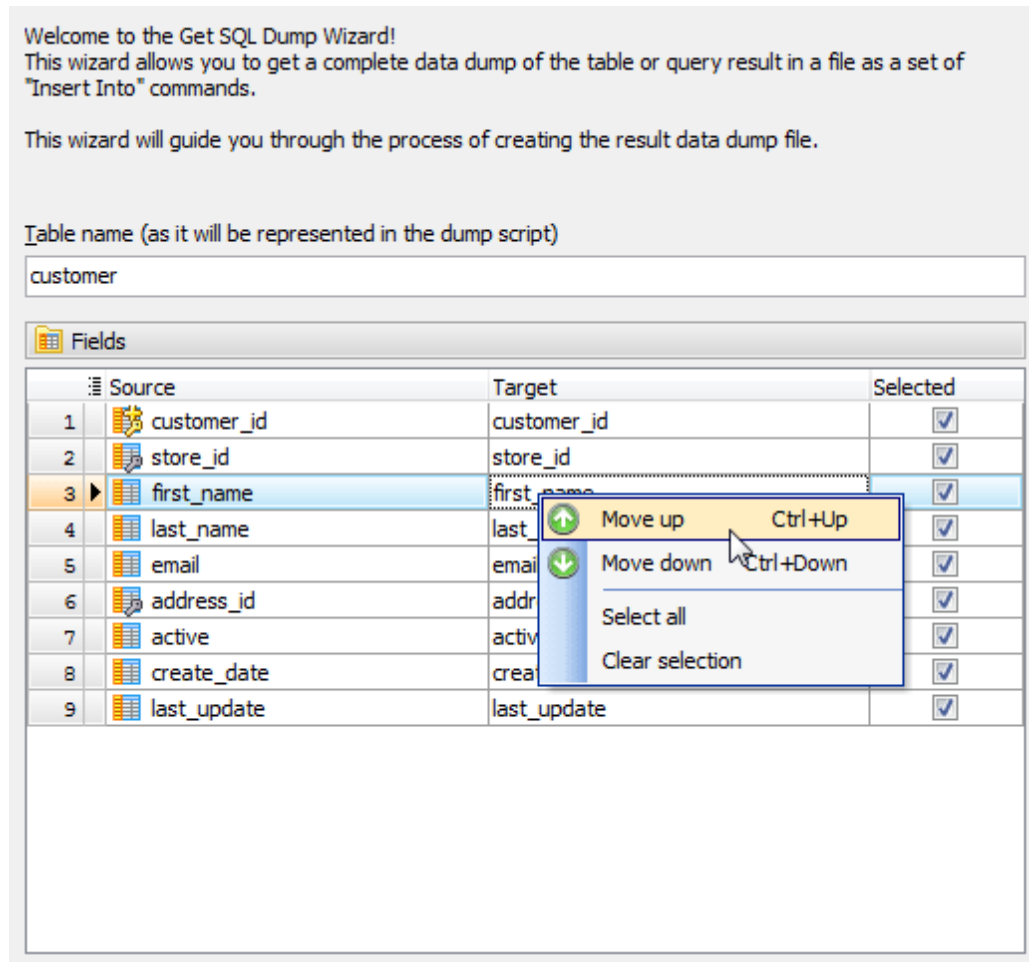
- [Selecting fields to include in the result INSERT statement](#) ^[252]
- [Specifying dump options](#) ^[253]

See also: [Export Data Wizard](#) ^[245], [SQL Script Editor](#) ^[266]

8.4.1 Selecting fields

The first wizard step allows you to specify the table name as it will be included in the result script.

You can also select the fields to be included in the result *INSERT* statement. All the table fields are included into the [Selected fields](#) list by default. If you do not want some fields to be exported, move them back to the [Available fields](#) list. *Text*, *GUID*, *Date*, *Time*, and *DateTime* columns are included in the result *INSERT* statements according to the [Storage Options](#) of the [Database Profile](#)^[31].



8.4.2 Specifying dump options


Select the data dump mode to be used ([Multi-row INSERT statements](#) or [separate single-row INSERT statements](#)) and specify commits' frequency.

To add the "CREATE TABLE" to the top of the dump, check the corresponding box.

Get SQL Dump Wizard allows you to send the result script to [SQL Script Editor](#)^[266] or to save it to a specified file. Select the [Send to script editor](#) option to load the result to the editor. To save the result to the file, enter the script file name (*.sql).

Click the [Ready](#) button to start the process.

Data script

 Specify the data dump options.

☒ Use multi-row INSERT statements






























Record count per each statement

☐ Commit after each statement

☐ Use separate single-row INSERT statements

Commit after

Statement syntax

8.5 Import Data Wizard

[Import Data Wizard](#) provides you with a graphical user interface to import data from the most popular files formats into existing PostgreSQL tables. It allows you to adjust data formats, empty target tables, execute custom SQL scripts, etc.

Import Data tool supports:

- Microsoft Office Excel 95-2003
- Microsoft Office Excel 2007
- Microsoft Office Access
- Microsoft Office Access 2007
- Delimiter-separated values (CSV, DSV, TSV)
- DBF
- Text files
- XML
- ODBC data sources (any database accessible via an ODBC driver or OLE DB provider, such as SQL Server, MySQL, Oracle, MS Access, Sybase, DB2, PostgreSQL, etc.)

In order to run the wizard you should

- open the table in [Table Editor](#);
- go on to the [Data](#) tab;
- select the [Import Data](#) item from the [Navigation Bar](#).

To import data,

- [Set the format](#) ^[256] of the input data and the source file name;
- [Map source file columns and target table fields](#) ^[258];
- [Specify other import options](#) ^[261].

Source format

Select one of the available source formats.

- ☒ Microsoft Office Excel 97 - 2003
- ☐ Microsoft Office Excel 2007
- ☐ Microsoft Office Access
- ☐ Microsoft Office Access 2007
- ☐ Delimiter-separated values (CSV, DSV, TSV)
- ☐ Text file (Fixed-width columns)
- ☐ DBF
- ☐ XML
- ☐ ODBC data source

Source file

Select or enter the source file name and specify the encoding if necessary.

File name	Password	Encoding
D:\Data\Excel\employee.xls		ANSI
Connection string	Identifier quote characters	
	None (table_name)	
Data source	Data location	Delimiter
Employee_list	Attributes	
		Quote

See also: [Export Data Wizard](#)

8.5.1 Setting source file name and format

1. Select the format of the source file.
2. Specify the file you want to import. The file name extension in the **File name** box varies according to the selected import type. The wizard allows you to import data from several files at a time.

To import data from multiple files with the same structure, set the mask of the file names to the corresponding field. To see the list of matching files, use with the button on the right.

Example 1:

Suppose, you need to import data from the following tables:

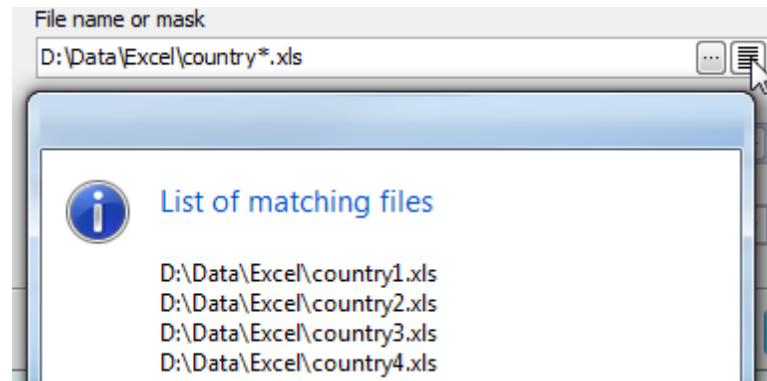
D:\Data\Excel\country1.xls

D:\Data\Excel\country2.xls

D:\Data\Excel\country3.xls

D:\Data\Excel\country4.xls

The mask for these file names is *D:\Data\Excel\country*.xls*.



3. For ODBC data sources specify the [connection string](#) to be used to connect to the data source.
4. Select the data source to import: a table of MS Access database or a spreadsheet of MS Excel.
5. Enter the password to the database (MS Access).
6. For CSV file set the delimiter and quote characters.
7. Select source file [Encoding](#).
8. For .XML files, define the [XPath](#) to the data to be imported to the selected table and select whether data is stored in Attributes or in Subnodes.

Example 2:

To import data from the following .xml file, use XPath=/Employees/Employee and Data location=Subnodes

```
<?xml version="1.0" encoding="utf-8"?>
<Employees>
  <Employee>
    <ID>1</ID>
    <FirstName>Klaus</FirstName>
    <LastName>Salchner</LastName>
    <PhoneNumber>410-727-5112</PhoneNumber>
  </Employee>
  <Employee>
    <ID>2</ID>
    <FirstName>Peter</FirstName>
    <LastName>Pan</LastName>
    <PhoneNumber>604-111-1111</PhoneNumber>
  </Employee>
</Employees>
```

Example 3:

To import data from the .xml file below, use XPath=DATAPACKET/Data/Item and Data location=Attributes

```
<?xml version="1.0"?>
```

```
<DATAPACKETVersion="2.0">
<Data>
  <Item ID="1" FirstName="Klaus" LastName="Salchner" PhoneNumber="410-727-
5112" />
  <Item ID="2" FirstName="Peter" LastName="Pan" PhoneNumber="604-111-1111" />
</Data>
</DATAPACKET>
```

8.5.2 Setting the accordance between source and target columns

The wizard provides you with several ways to map input data to the target table columns.

- You can map columns automatically by order with the [Auto Fill](#) and [Auto fill all maps](#) buttons.
- You can do it manually using the drop-down list of [Source column](#) fields.
- To map columns visually, open [Map builder](#)^[259] with the [Build map](#) link.

It's useful to save a specified map to a file for further using it in the next wizard sessions. To save a map, use the [More...](#) button and follow the [Save map](#) link.

To see the 100 first rows of input file or output table, use the [More...](#) button and follow the [View source data](#) or [Preview results](#) links respectively.

You can also specify [Replacements](#) to be applied to the selected column before the import and [data format masks](#)^[260] used for the input file.

To exclude the first file row, use the [File contains column header](#) checkbox.

Columns				
	Target field	Source column	Replacements	Empty values interpretation
1	film_id	A		
2	title	B		As Null
3	description	C		As Null
4	release_year	D		
5	language_id	E		
6	original_language_id	F		
7	rental_duration	G		
8	rental_rate	H		
9	length	I		
10	replacement_cost	J		
11	rating	K		As Null
12	last_update	L		
13	special_features	M		As Null
14	fulltext			As Null

Auto fill

Clear

Define columns...

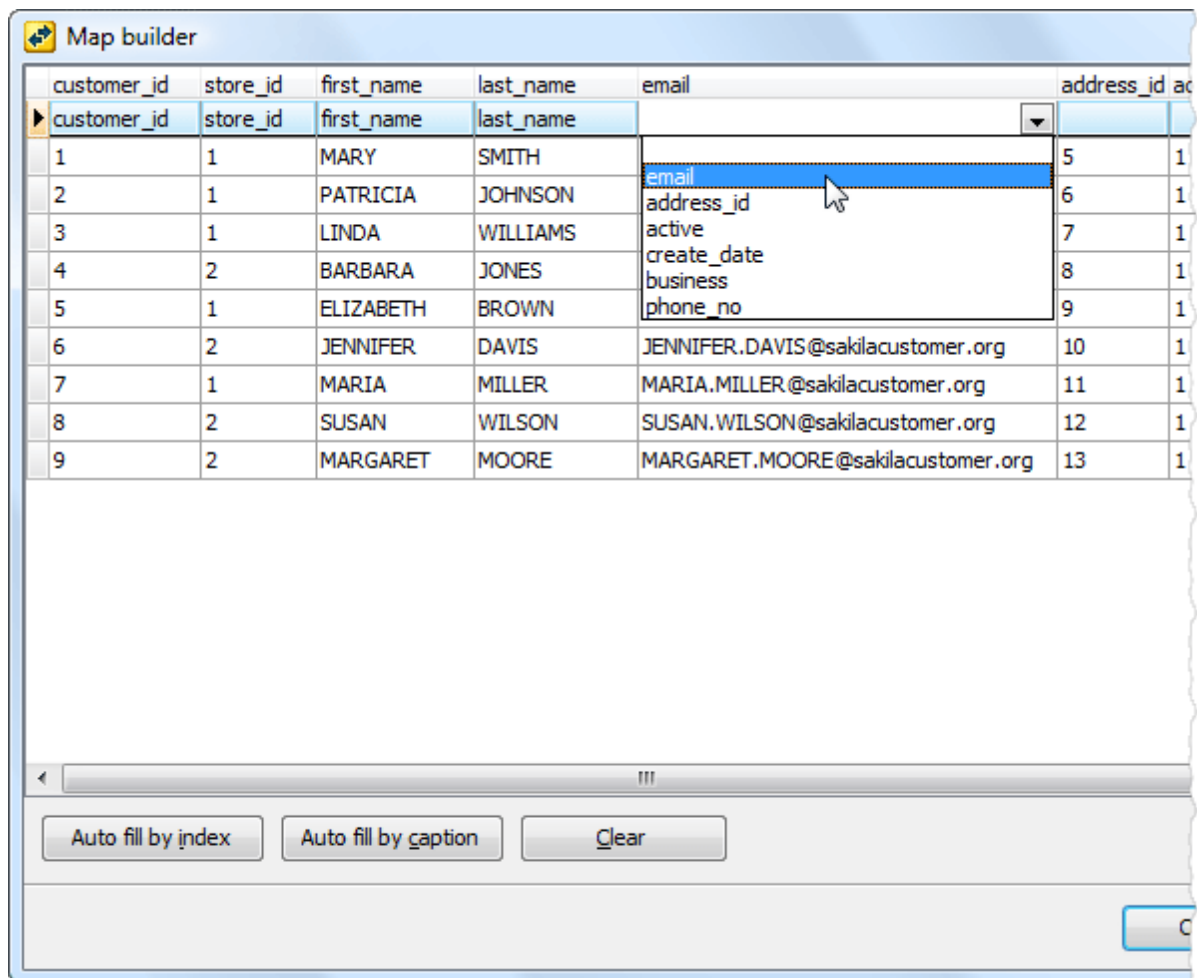
Data formats...

More...

☐ File contains column headers

8.5.2.1 Map builder

To specify the accordance between source and target columns visually, use popup menu of the upper row to map source file columns to target table fields.



For text files define columns bounds first. To add a bound, double-click near the column data in the builder area. To map a column to a target table field, select the field in the **Target field** list and then click between the bounds.

	Target field	Offset	Width
1	customer_id	0	5
2	store_id	5	6
3	first_name	11	46
4	last_name	57	46
5	email	103	52
6	address_id	155	6
7	active	161	6
8	create_date	167	6

	0	10	20
	custo	store	first_name
1	1	1	MARY
2	1	1	PATRICIA
3	1	1	LINDA
4	2	1	BARBARA
5	1	1	ELIZABETH
6	2	1	JENNIFER
7	1	1	MARIA

8.5.2.2 Data formats

Use the window fields to indicate format masks of the source data imported to the table. It allows the application to import data correctly.

The components of the date time format mask are represented at the window. Compose

your date, time, and date time format mask of this components and separators. The following table contains some types of input fields and suggests masks to import them.

To import these input data correctly	Use these format masks
June 29	mmm dd
Jun 29, 2009	mmmm dd, yyyy
Tue Jun 14 16:50:49	dddmmm dd hh:nn:ss
01/15/09 08:26 AM	mm/dd/yy h:nn ampm

You can also set decimal and thousand separators, and custom NULL,TRUE and FALSE values. If you have several values to be imported to NULL(TRUE, FALSE) value, use semicolons to separate them.

<div> <div>Formats</div> <table> <tr><td>Date</td><td></td></tr> <tr><td>Time</td><td></td></tr> <tr><td>Date time</td><td></td></tr> </table> </div>		Date		Time		Date time		<div>Date time formats</div> <p> dd the day as a number with a leading zero or space (01-31). ddd the day as an abbreviation (Sun-Sat) dddd the day as a full name (Sunday-Saturday) mm the month as a number with a leading zero or space (01-12). mmm the month as an abbreviation (Jan-Dec) mmmm the month as a full name (January-December) yy the year as a two-digit number (00-99). yyyy the year as a four-digit number (0000-9999). hh the hour with a leading zero or space (00-23) nn the minute with a leading zero or space (00-59). ss the second with a leading zero or space (00-59). zzz the millisecond with a leading zero (000-999). ampm Specifies am or pm flag hours (0..12) ap Specifies a or p flag hours (0..12) </p>
Date								
Time								
Date time								
<div> <div>Separators</div> <table> <tr><td>Decimal</td><td>,</td></tr> <tr><td>Thousand</td><td>#160</td></tr> </table> </div>		Decimal	,	Thousand	#160			
Decimal	,							
Thousand	#160							
<div> <div>Other (use semicolon to separate values)</div> <table> <tr><td>Boolean true</td><td>True</td></tr> <tr><td>Boolean false</td><td>False</td></tr> <tr><td>Null values</td><td>;NULL</td></tr> </table> </div>		Boolean true	True	Boolean false	False	Null values	;NULL	
Boolean true	True							
Boolean false	False							
Null values	;NULL							

8.5.3 Customizing common options

On the wizard step you can set the number of records to import, whether the tool import all table records or only the specified number. In the second case you can set the number of records to skip.

Logging

This options group let you to manage logging of the import process.

Scripts

There are many cases where the import process is necessary to correct with additional scripts. So to disable table indexes before the importing, specify the corresponding scripts to be executed before and after the process.

The typical example of usage of the [Before each table](#) and [After each table](#) scripts is the import data to autoincrement columns of several tables. In this case it's necessary to set the corresponding scripts:

```
SET IDENTITY_INSERT %table_name% ON
```

and

```
SET IDENTITY_INSERT %table_name% OFF
```

to be executed before and after import data to each table correspondingly.

Import mode

If the [Update existing records](#) option is turned ON, the records will be either updated or inserted: an UPDATE will be performed when a target row exists in the table and an INSERT is performed when the target row does not exist.

Import Data Wizard supports the [COPY FROM](#) command to insert data to the table. This feature can speed up the import process up to 10 times so it is recommended to use it always if possible. Uncheck this option to use INSERT statements instead.

9 Database Tools

PostgreSQL Maestro provides a number of powerful tools for working with databases.

The following tools are available:

- **[SQL Editor](#)**^[214]
Creates and executes SQL queries.
- **[Visual Query Builder](#)**^[219]
Builds queries visually.
- **[Script Runner](#)**^[265]
Executes SQL scripts to the database.
- **[SQL Script Editor](#)**^[266]
Allows to edit and execute SQL scripts.
- **[Extract Database Wizard](#)**^[269]
Extracts the database objects and data to the SQL script, which can be executed later to reserve the database structure and data.
- **[Generate Database Report Wizard](#)**^[274]
Generates the database HTML or PDF report for structure of selected object in a whole or partially.
- **[BLOB Viewer](#)**^[276]
Displays a content of BLOB fields in different representations.
- **[Diagram Viewer](#)**^[282]
Represents data from a table or a query as a diagram in various ways.
- **[Data Analysis](#)**^[286]
Allows to slice and dice information efficiently according your business rules.
- **[Report Designer](#)**^[291]
Prepares data for reading, viewing, and printing in a polished look.
- **[Schema Designer](#)**^[298]
Allows to represent database tables and relationships as ER diagrams.
- **[PL/SQL Debugger](#)**^[303]
An excellent tool to debug PL/SQL code such as procedures and functions (both stand-alone and packaged) using traditional debugging features.
- **[Process Browser](#)**^[307]
A very useful feature for DBAs to monitor the users' activity.
- **[SQL Generator](#)**^[309]
Provides you with a set of simple SQL statements.
- Simple tools for **[DML procedures](#)**^[310] and **[Updatable views](#)**^[312] generation allow to

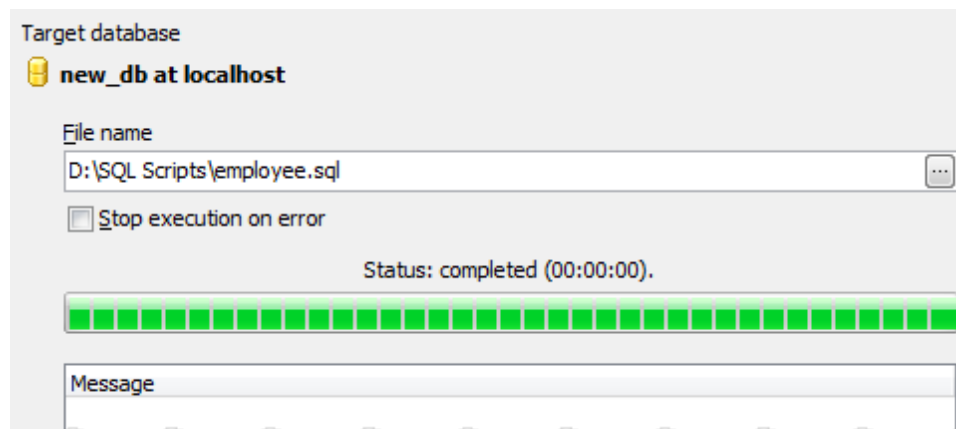
create a bunch of CRUD procedures automatically.

9.1 Script Runner

Script Runner is designed for executing of SQL scripts that don't require modifications. The window can be invoked from the **Tools** menu or with the **Execute script from file** link of **SQL Script Editor**^[266].

Script Runner allows to execute .sql files as well as archived scripts directly from .zip files. In case archived files this tool unpacks zip archives to temporary files by itself for further executing. The tool neither starts any implicit transactions before executing the script nor issues COMMIT or ROLLBACK commands after the executing.

To execute a script with Script Runner, set the file name and the **Stop execution on error** option value. This option allows to view all the execution errors (OFF). The specified script will be executed immediately on the database which name is represented at the top of the window.

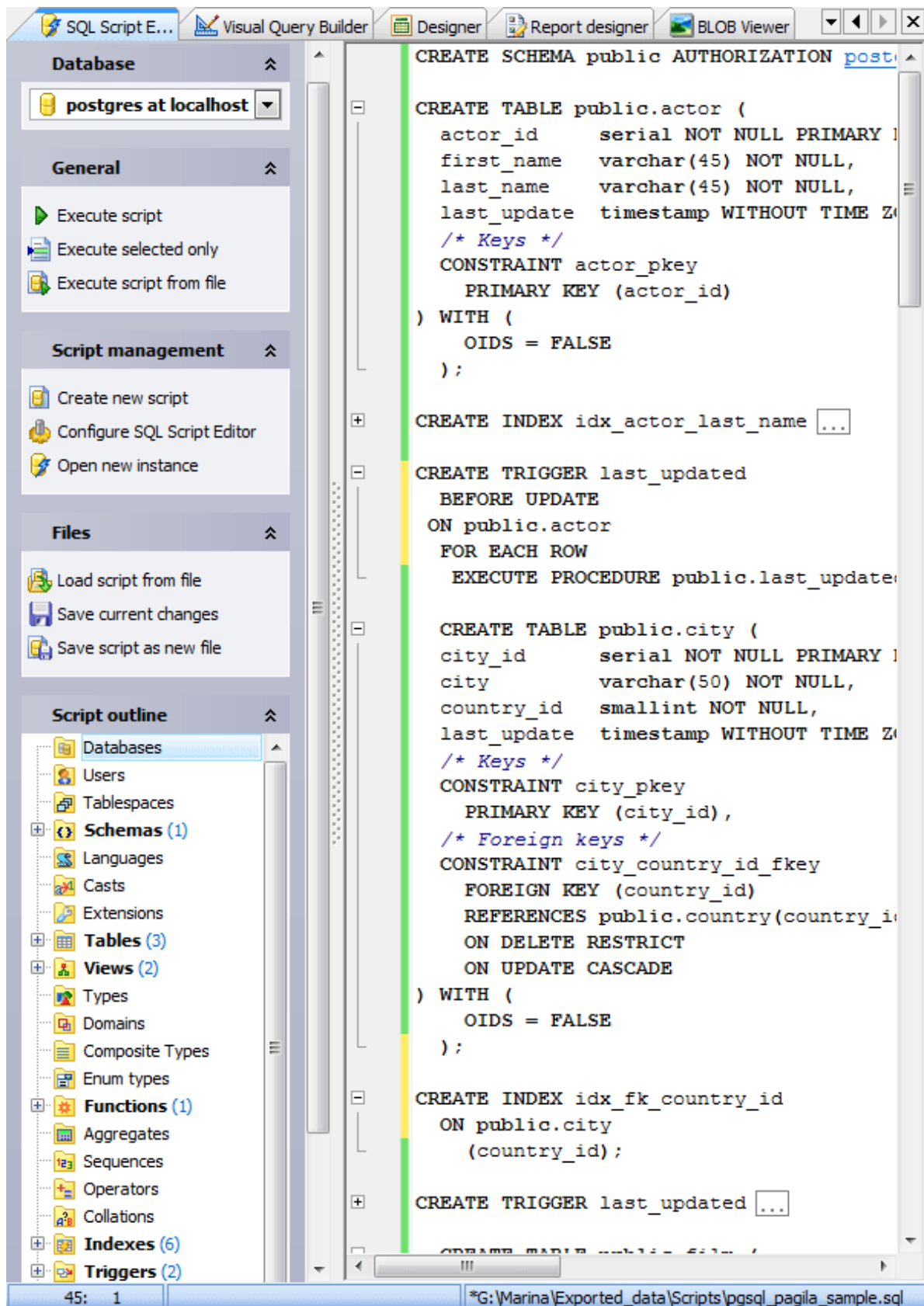


9.2 SQL Script Editor

[SQL Script Editor](#) is designed for SQL scripts editing and executing. The editor does not display results of SELECT queries. To work with such queries' data, use [SQL Editor](#)^[214]. If you have a script that is ready to use, execute it with [Script Runner](#)^[265]. To open [SQL Script Editor](#), select the [Tools | SQL Script Editor](#) main menu item.

To work with a script within SQL Script Editor, load it from an `.sql` file or type it in the editor area directly. To prevent mistakes in SQL syntax, the editor supports syntax highlighting, code completion and divides the script text into logical parts that can be individually collapsed or expanded (code folding). All the logical parts are represented at the [Explorer](#) at the [Navigation bar](#). It allows you to transfer to the proper script fragment quickly by clicking the corresponding node in the tree.

SQL Script Editor allows you to execute the whole SQL script or only its selected part. To make the executing of a large script much faster, execute the script directly from a file with [Script Runner](#)^[265]. By default, if a user opens a file larger than 100K, SQL Script Editor will suggest him to execute the script file without opening it in the editor. This file size may be changed at the editor's [options](#)^[329] tab.



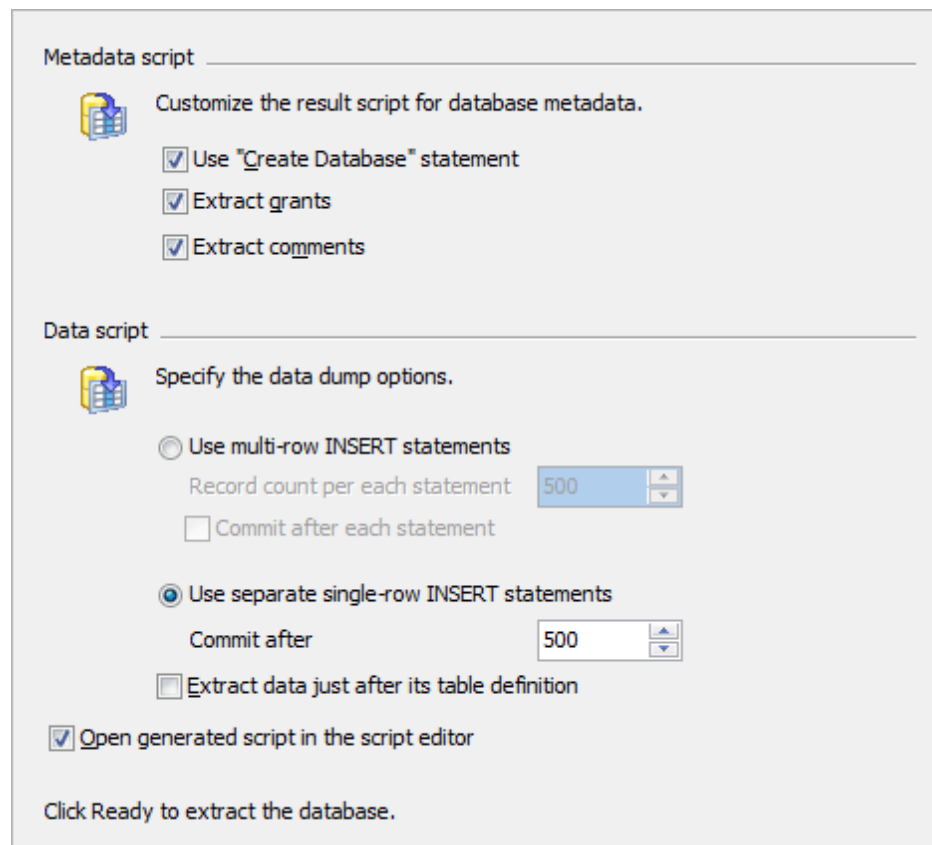
9.3 Extract Database

Using [Extract Database Wizard](#) you can extract database objects and data to a SQL script, e.g. for backup purposes. To run this wizard select the [Tools | Extract Database](#) main menu item.

Use the [More...](#) button to save the extract configuration for future use or to load the previously saved configuration for faster extract.

- [Selecting database to extract and the target script file name](#)^[269]
- [Selecting database objects to extract their structure](#)^[270]
- [Selecting database objects to extract their data](#)^[271]
- [Customizing script options](#)^[272]

See also: [Get SQL Dump Wizard](#)^[252]



The screenshot shows the 'Data script' tab of the 'Extract Database Wizard' configuration window. The window is titled 'Metadata script' at the top, but the 'Data script' tab is selected. The 'Data script' section contains the following options:

- Customize the result script for database metadata.**
 - ☒ Use "Create Database" statement
 - ☒ Extract grants
 - ☒ Extract comments
- Specify the data dump options.**
 - ☐ Use multi-row INSERT statements
 - Record count per each statement: 500
 - ☐ Commit after each statement
 - ☒ Use separate single-row INSERT statements
 - Commit after: 500
 - ☐ Extract data just after its table definition
 - ☒ Open generated script in the script editor

Click Ready to extract the database.

9.3.1 Selecting the database and the target file name


Select the [source database](#) to extract and set the target script [file name](#).

Select the components to be extracted: object definitions, table data or both.

Welcome to the Extract Database Wizard!
This wizard allows you to extract the database structure and table data into the SQL script.

This wizard will guide you through the process of selecting schema objects and data tables and setting other options for generating the result script.

Source database

 NORTHWIND at MERCURY

Script file name

NORTHWIND.sql

You can select to extract either database structure, or table data only, or both.
Which components would you like to extract?

☒ Extract both of structure and data

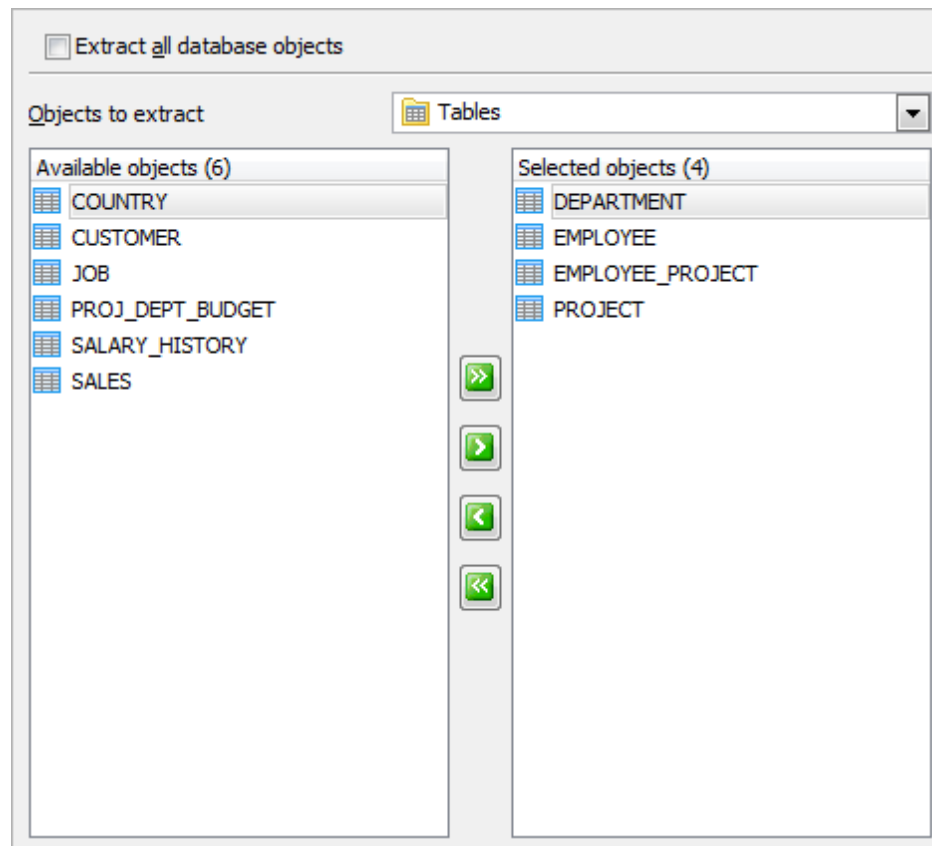
☐ Extract structure only

☐ Extract data only

9.3.2 Selecting objects to extract their structure

Select the database object to be extracted or check the [Extract all database objects](#) option to extract all objects from the database.

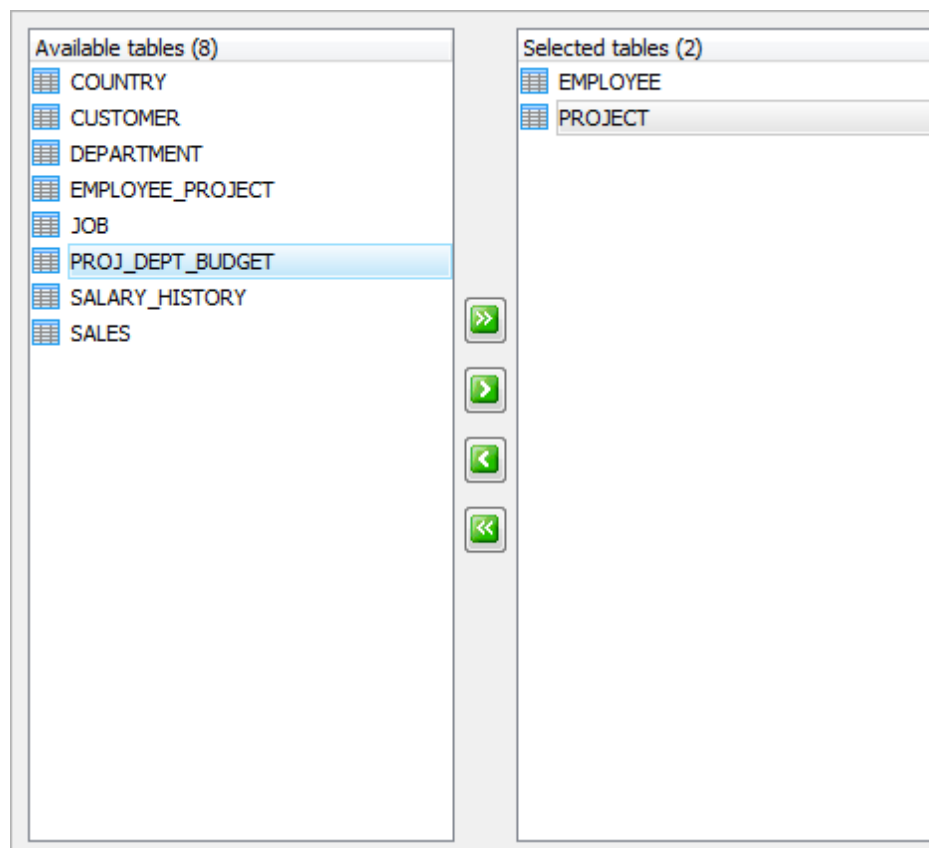
Note: this step is not available if you select the extract data only mode when [selecting components to be extracted](#).



9.3.3 Selecting objects to extract their data

Select the tables to extract their data by moving them from the [Available tables](#) list to the [Selected tables](#) list.

Note: this step is not available if you select the extract structure only mode when [selecting components to be extracted](#).



9.3.4 Customizing script options

Adjust the result script for database metadata according to your needs. To include/exclude the following statements into the script, check the corresponding boxes: *CREATEDATABASE*, *DROPDATABASE IF EXISTS*, *DROP IF EXISTS* statements for database objects, and foreign key checks.


Select the data dump mode to be used ([Multi-row INSERT statements](#) or [separate single-row INSERT statements](#)) and specify commits' frequency.

You can also set extract data after its table definition.

Fill the [Open generated script in the script editor](#) box to load the result script to [SQL Script Editor](#)²⁶⁶.


Click the [Ready](#) button to start the extraction process.

Metadata script _____

 Customize the result script for database metadata.

- ☒ Use "Create Database" statement
- ☒ Extract grants
- ☒ Extract comments

Data script _____

 Specify the data dump options.

☐ Use multi-row INSERT statements

Record count per each statement

☐ Commit after each statement

☒ Use separate single-row INSERT statements

Commit after

☐ Extract data just after its table definition

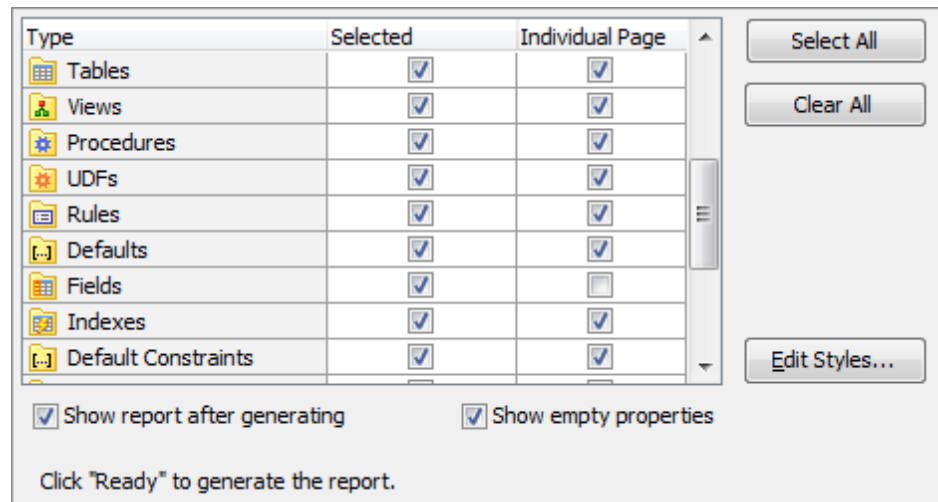
☒ Open generated script in the script editor

Click Ready to extract the database.

9.4 Generate Database Report

With the help of [Generate Database Report Wizard](#) you can create HTML or PDF report for the structure of the selected object in the whole or partially. To run this wizard select the [Tools | Generate Database Report](#) main menu item.

- [Selecting reporting elements and setting other report options](#)^[274]
- [Specifying reporting objects and editing styles](#)^[274]



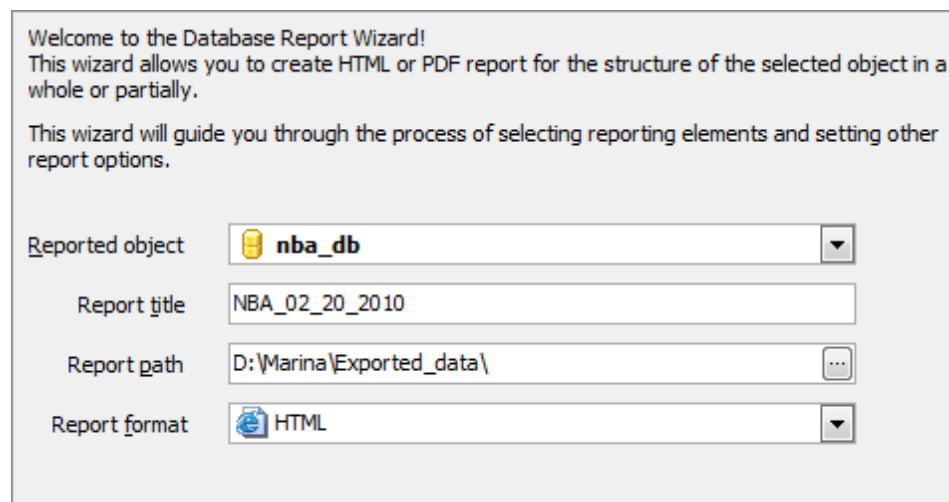
Type	Selected	Individual Page
Tables	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Views	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Procedures	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
UDFs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Rules	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Defaults	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fields	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Indexes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Default Constraints	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

☒ Show report after generating ☒ Show empty properties

Click "Ready" to generate the report.

9.4.1 Selecting reporting elements and setting other report options

Select the [report object](#) and the [report format](#) items, set the [report title](#) and the [report path](#) options in the respective boxes.



Welcome to the Database Report Wizard!
This wizard allows you to create HTML or PDF report for the structure of the selected object in a whole or partially.
This wizard will guide you through the process of selecting reporting elements and setting other report options.

Reported object: nba_db

Report title: NBA_02_20_2010

Report path: D:\Marina\Exported_data\

Report format: HTML

9.4.2 Reporting objects and editing styles options

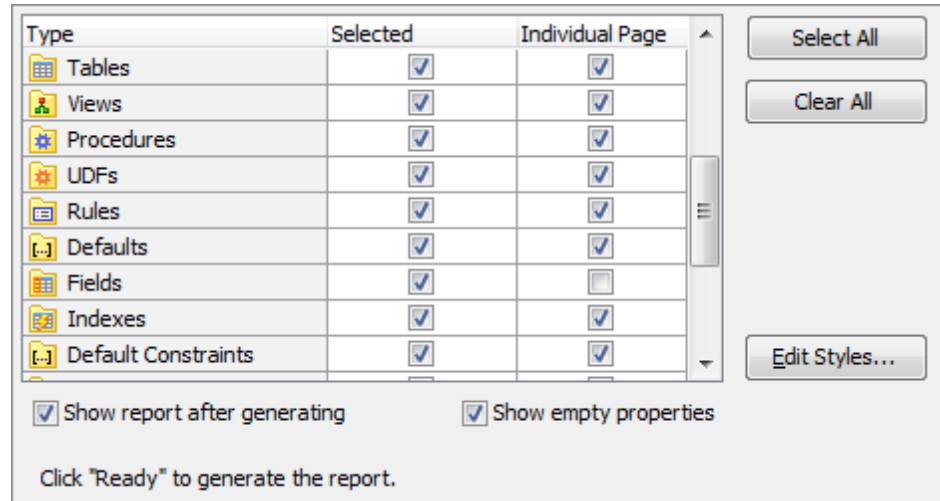
This step allows you to select the essential objects to report and to specify the output format and style using [Report Style Editor](#)^[275].

- ☒ Show report after generating

If checked, opens the result files in the associated program after making the report.

☒ **Show empty properties**

If checked, allows you to report objects even if they are empty.



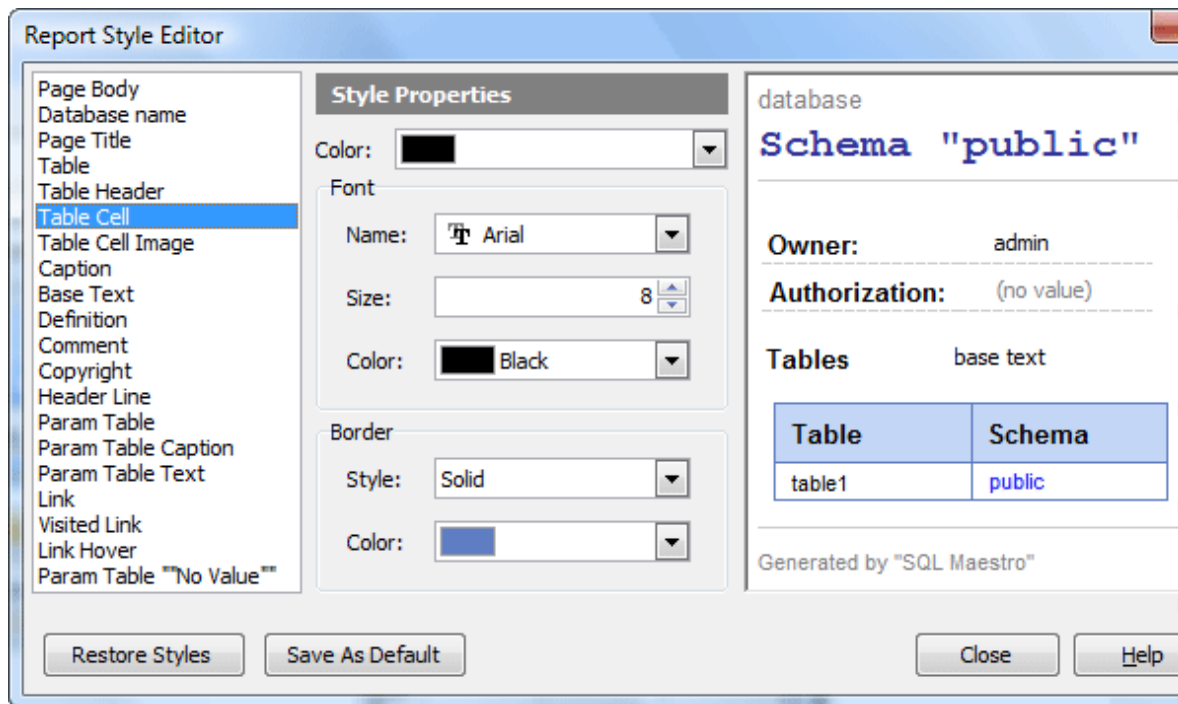
Type	Selected	Individual Page
Tables	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Views	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Procedures	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
UDFs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Rules	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Defaults	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fields	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Indexes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Default Constraints	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

☒ Show report after generating ☒ Show empty properties

Click "Ready" to generate the report.

9.4.3 Editing database report style

Using [Report Style Editor](#) you can specify style properties of a report including font size, color and name for different elements.



Report Style Editor

Page Body
Database name
Page Title
Table
Table Header
Table Cell
Table Cell Image
Caption
Base Text
Definition
Comment
Copyright
Header Line
Param Table
Param Table Caption
Param Table Text
Link
Visited Link
Link Hover
Param Table "No Value"

Style Properties

Color:

Font

Name: A Arial

Size: 8

Color: Black

Border

Style: Solid

Color:

database
Schema "public"

Owner: admin

Authorization: (no value)

Tables base text

Table	Schema
table1	public

Generated by "SQL Maestro"

Restore Styles Save As Default Close Help

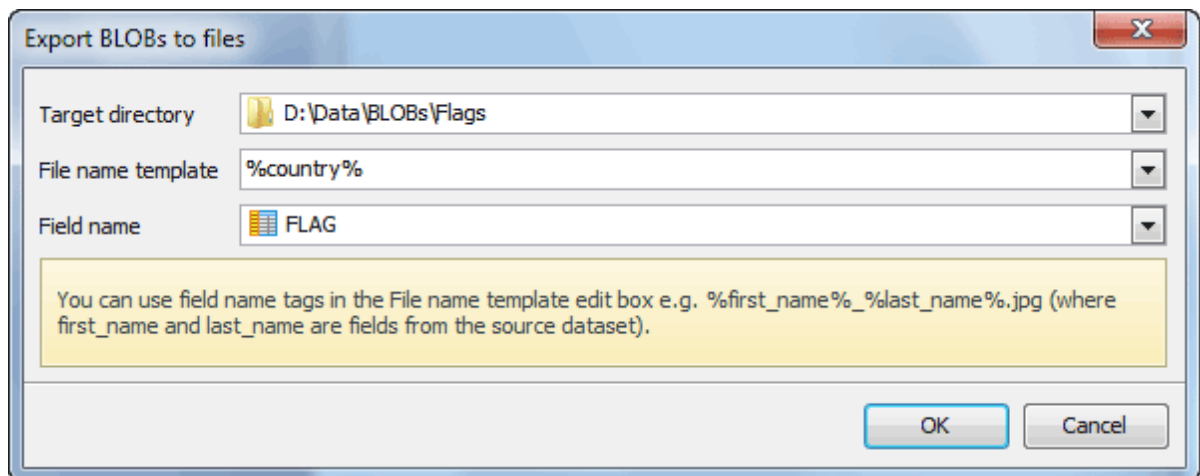
9.5 BLOB Viewer

BLOB Viewer allows you to view the content of the BLOB fields in various representations.

- [Viewing BLOB field as hexadecimal dump](#)^[276]
- [Viewing BLOB field as plain text](#)^[277]
- [Viewing BLOB field as graphical image](#)^[278]
- [Viewing BLOB field as HTML](#)^[279]
- [Viewing BLOB field as PDF](#)^[280]

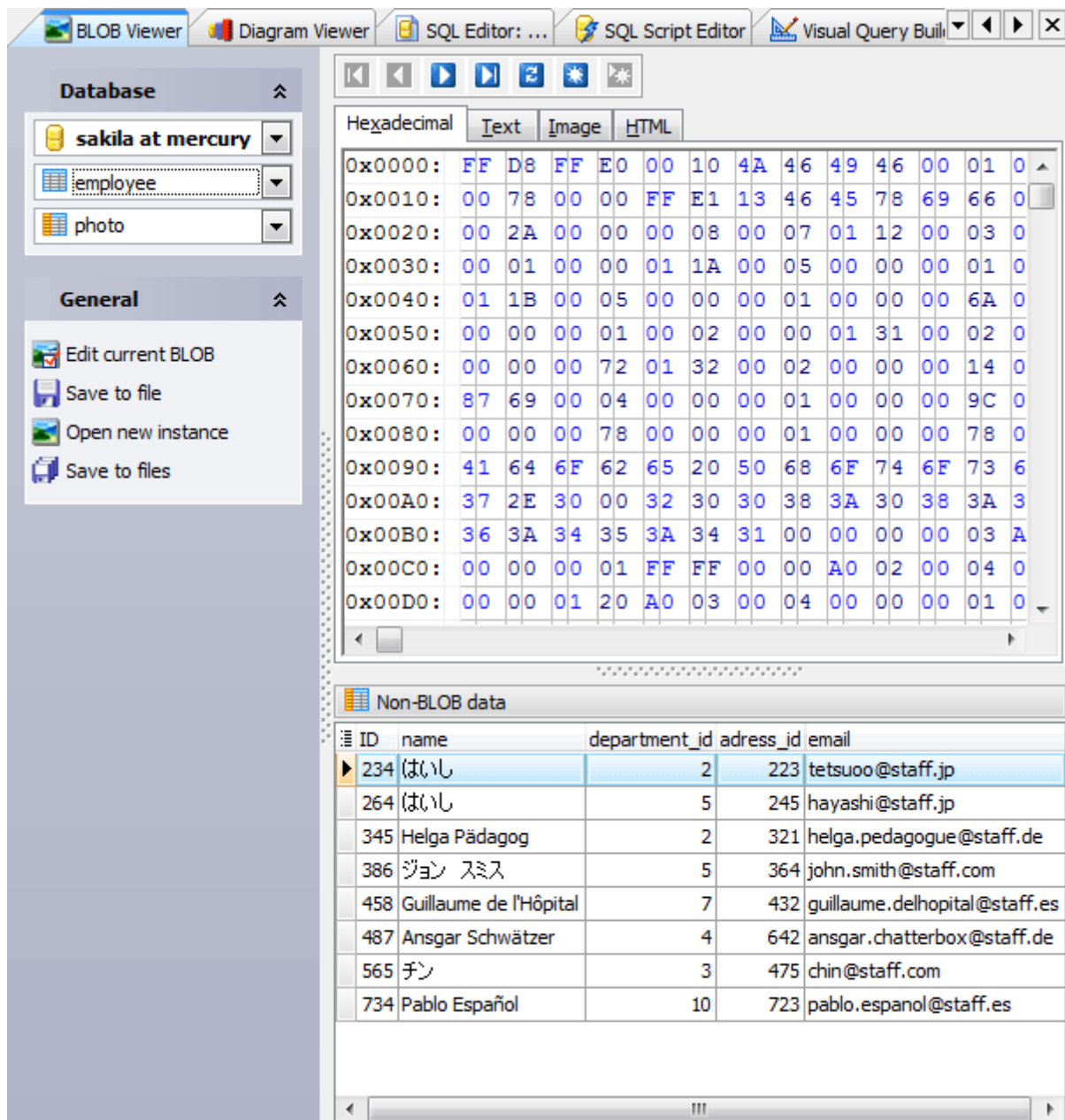
See also: [BLOB Editor](#)^[240]

BLOB Viewer also allows you to save all BLOBs from a table or view to a given directory. Just click [Save to files](#) on the [Navigation bar](#) and fill all fields in the [Export BLOBs](#) window shown below. You can use table columns' names enclosed in % as a file name template.



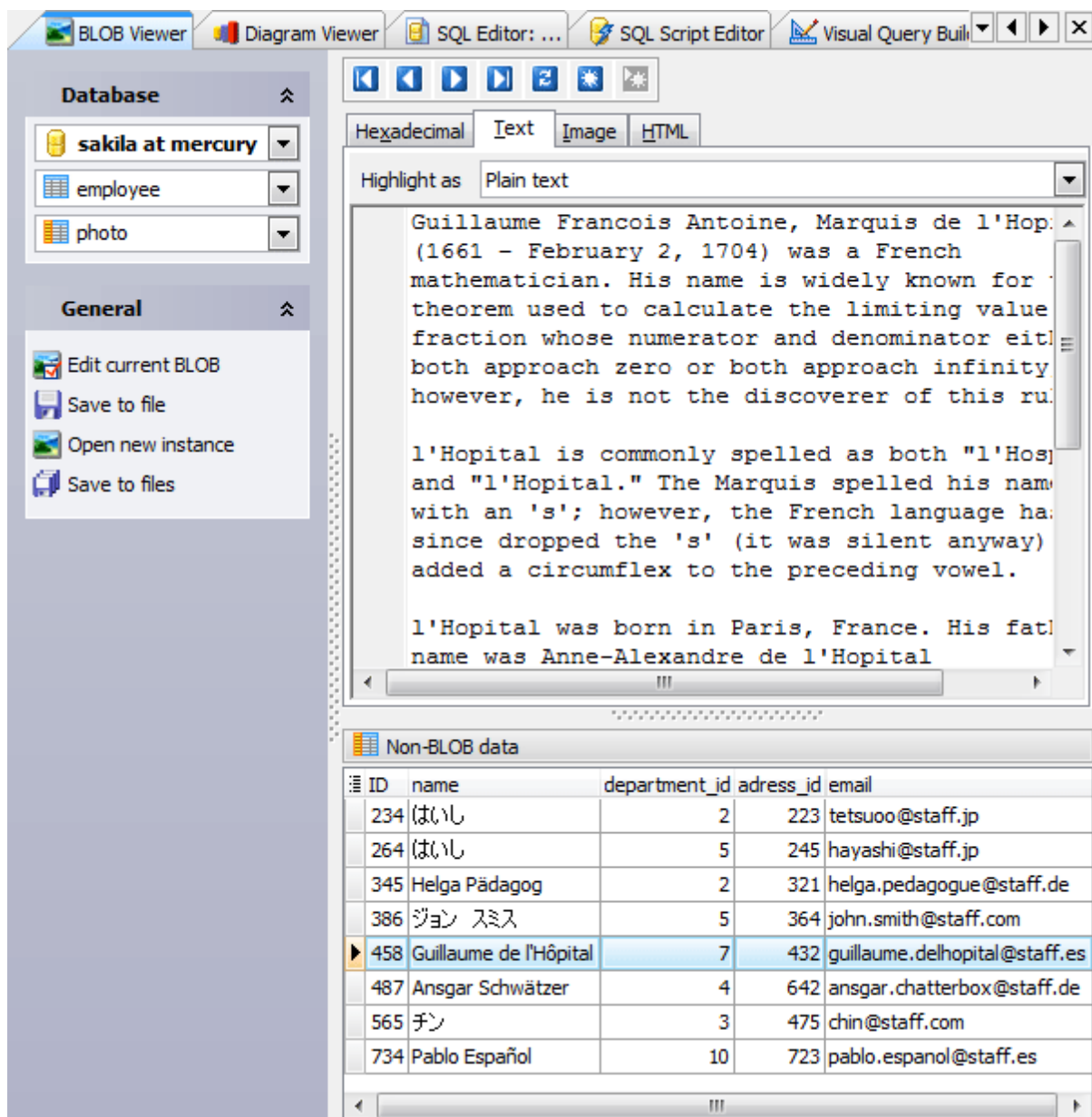
9.5.1 Viewing as hexadecimal dump

The [Hexadecimal](#) panel allows you to view data in hexadecimal mode.



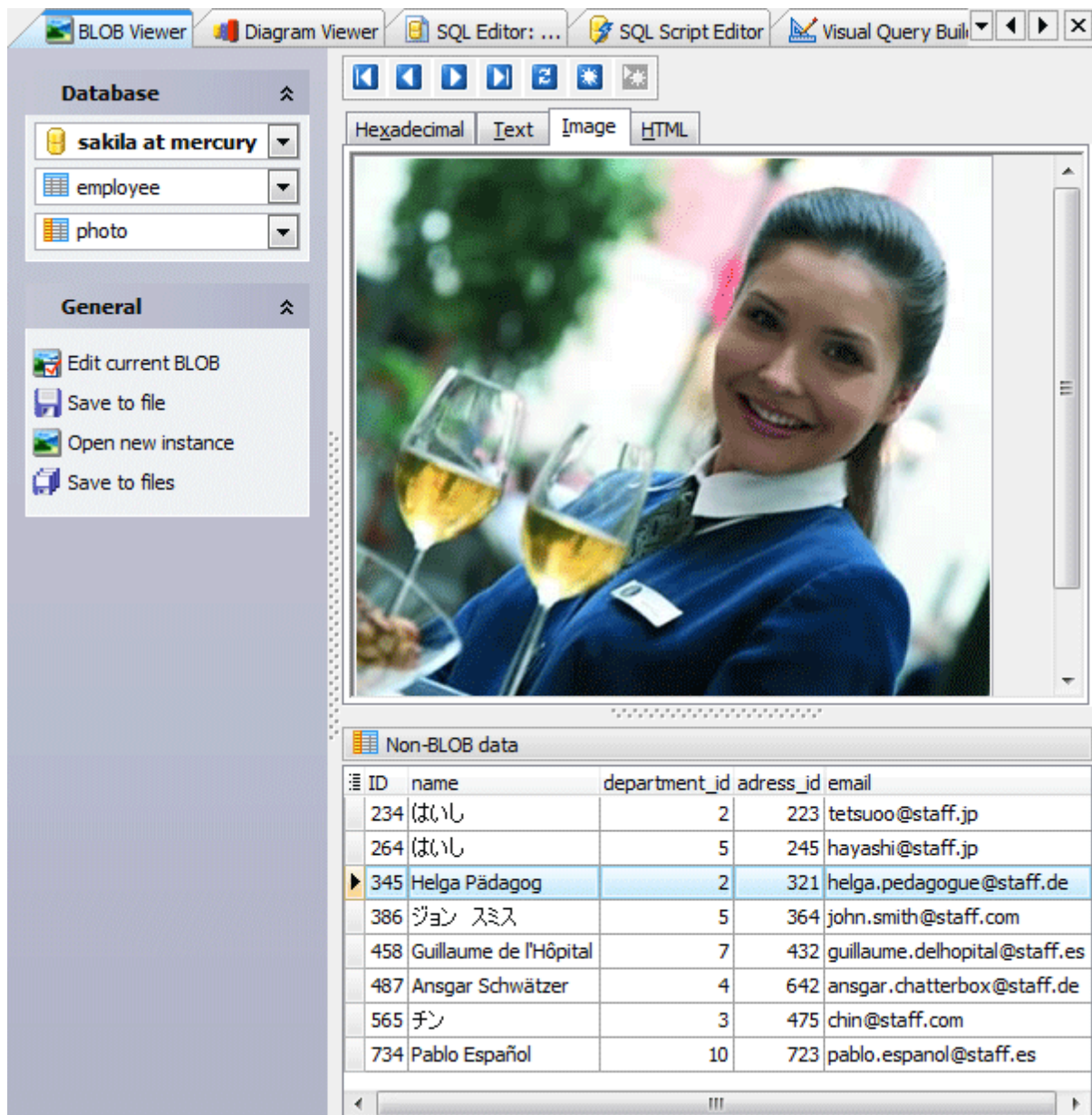
9.5.2 Viewing as plain text

The **Text** panel allows you to view data as simple text. For your convenience several types of text highlighting are available (*Plain text*, *HTML*, *JScript*, *CSS*, *PHP*, *XML*, *SQL*, and *SQLite DDL*). The popup menu of the panel provides you to **Find** or **Replace** a necessary text fragment.



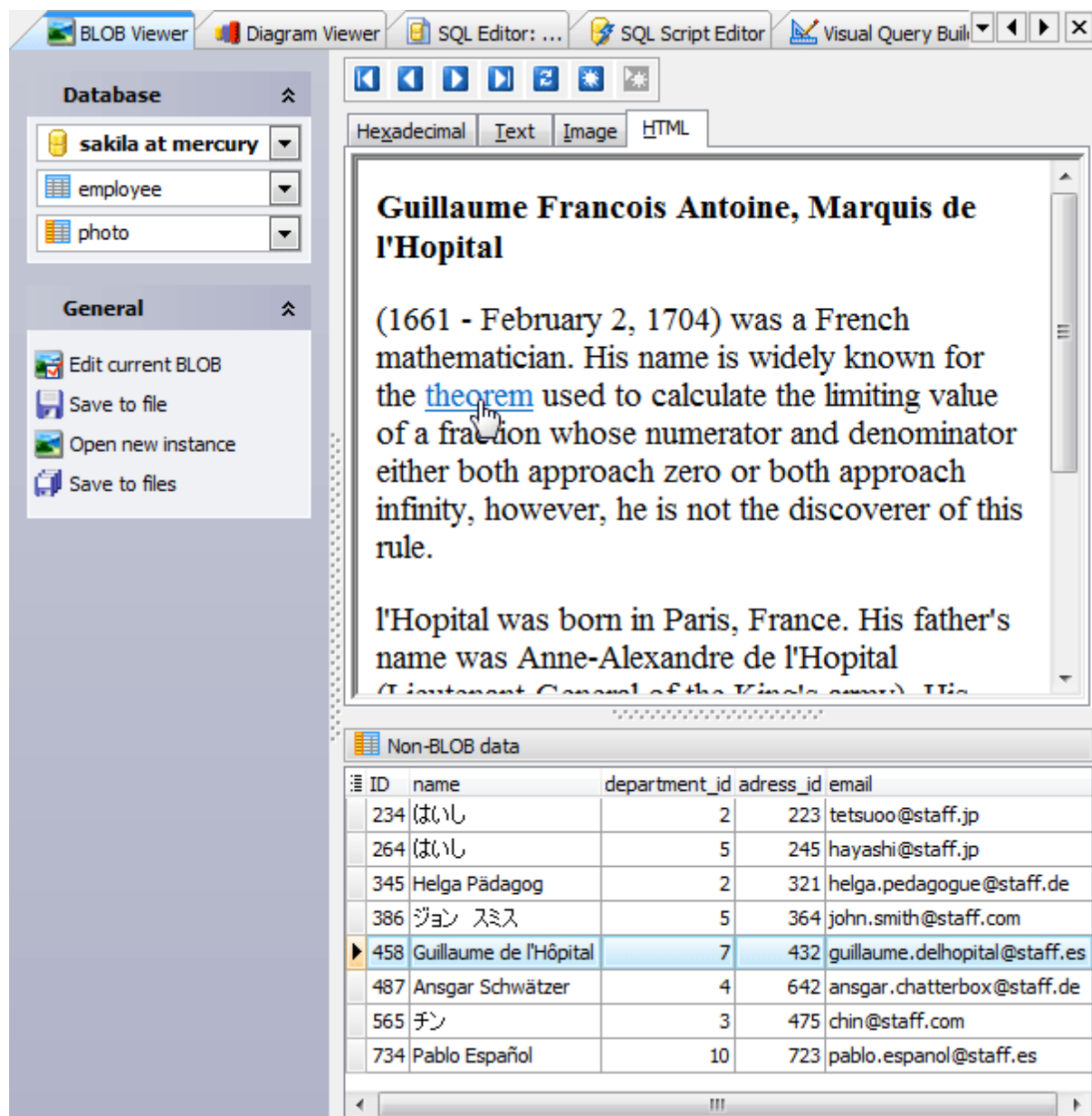
9.5.3 Viewing as image

The [Image](#) panel displays field data as image.



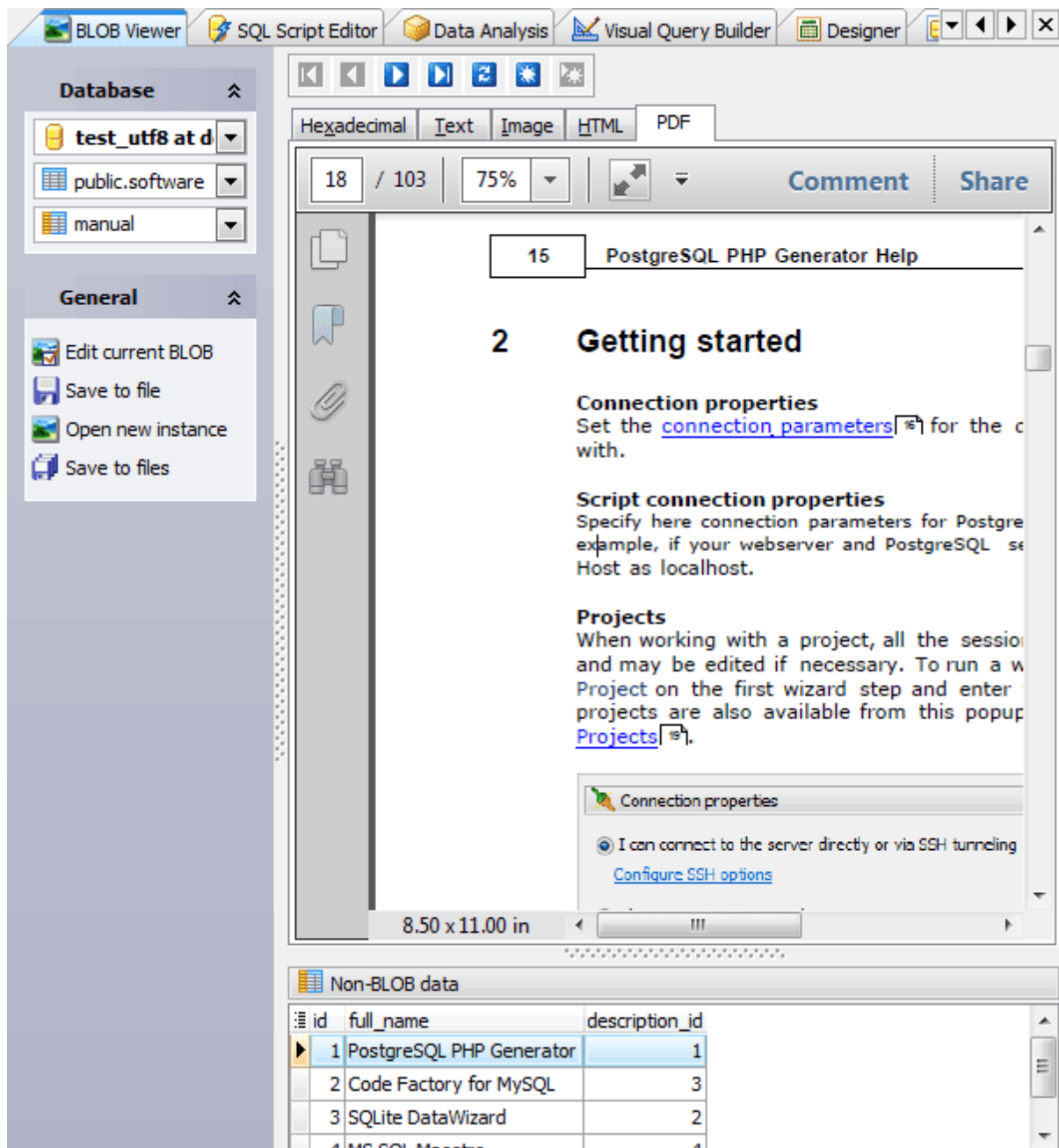
9.5.4 Viewing as HTML

The [HTML](#) panel displays field data as HTML.



9.5.5 Viewing as PDF

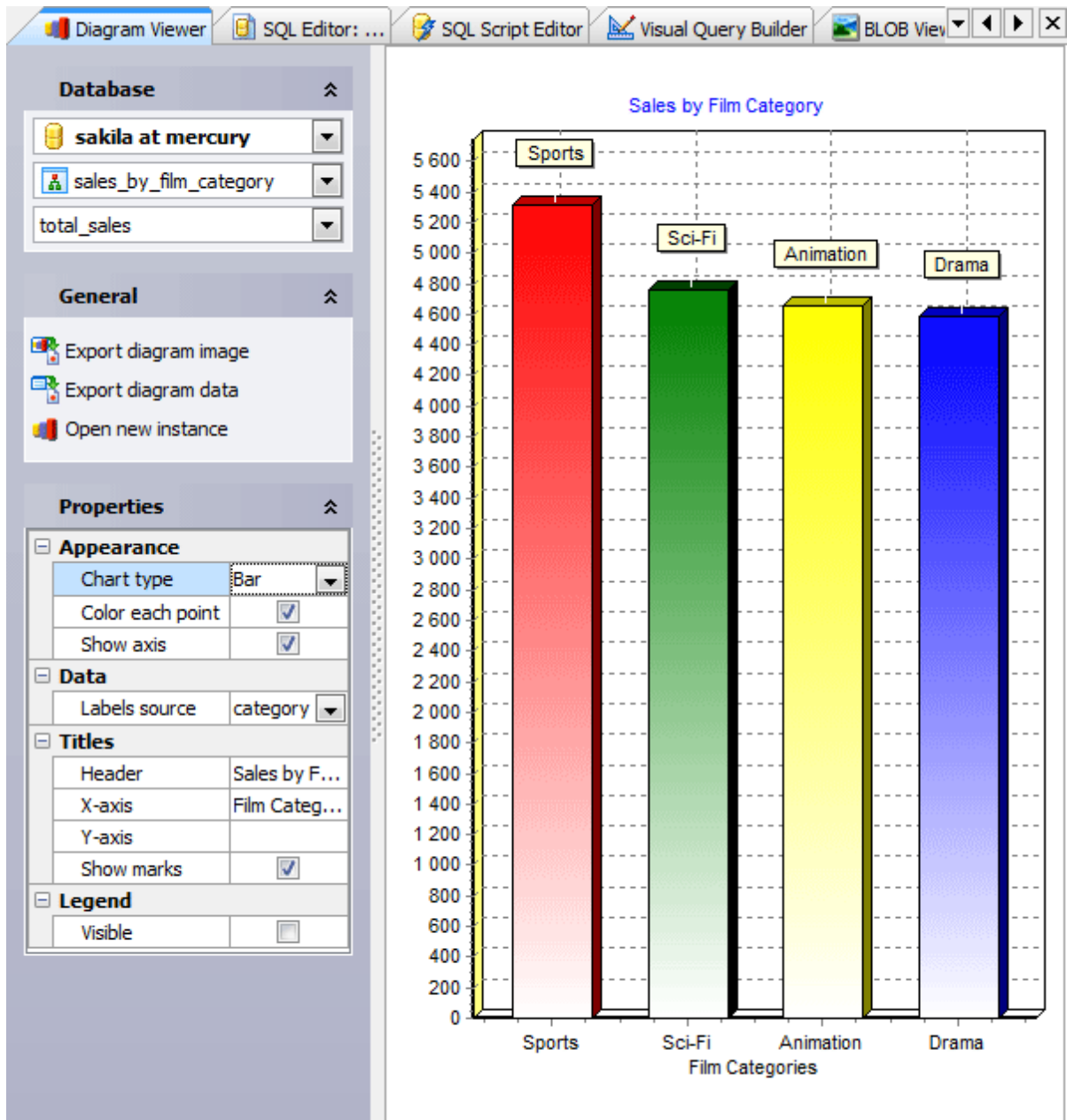
The PDF panel allows you to browse PDF data stored in the database.



9.6 Diagram Viewer

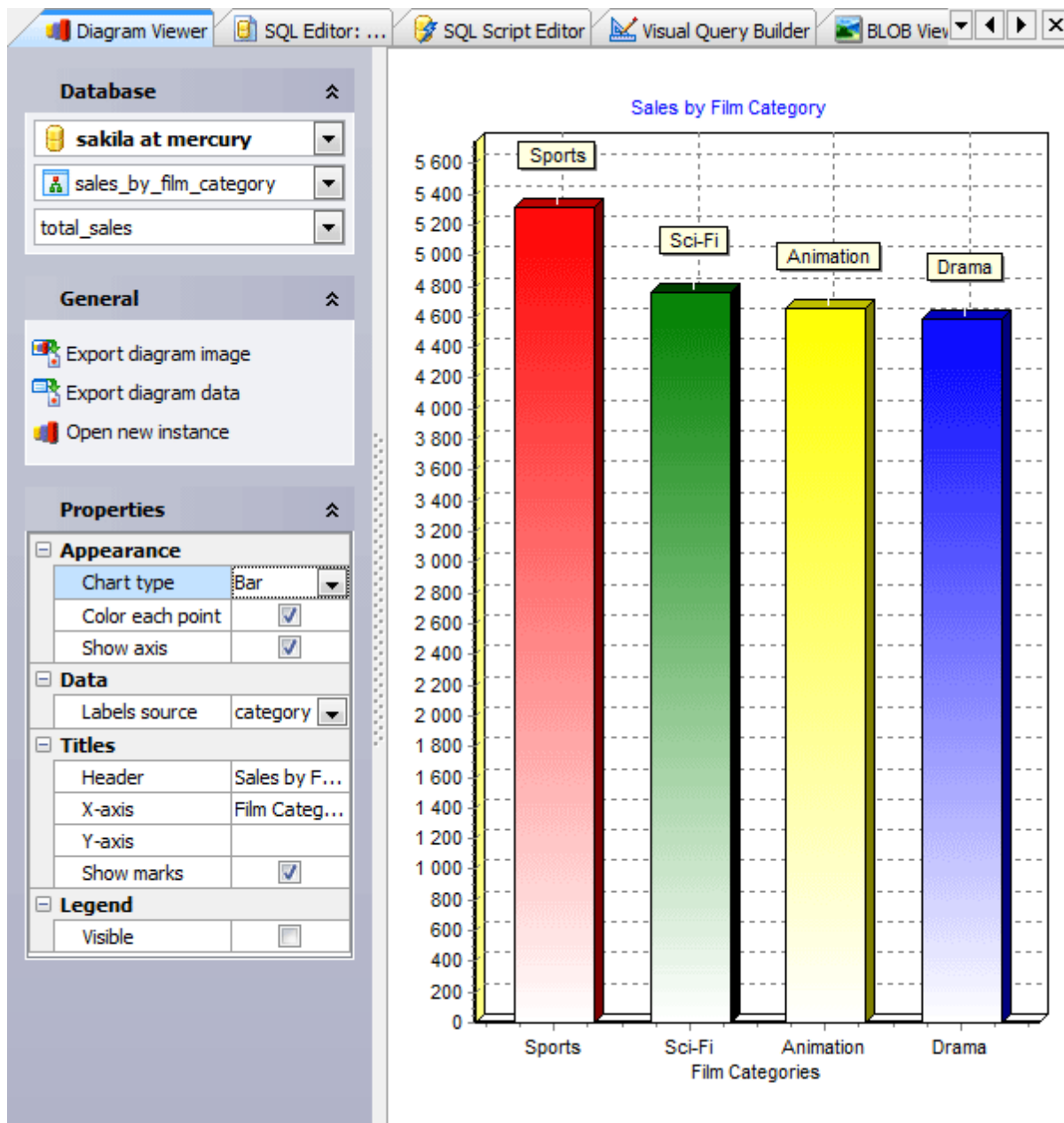
Diagram Viewer is a tool for representing data from a table or a query as a diagram in various ways. This means you can build a diagram represented as bars, lines, areas, points or pies, colored or not, with axis visible or not; specify axis labels source, diagram header and more. The **Diagram Viewer** also has the [Export diagram image](#)^[283] and the [Export diagram data](#) features implemented, with a lot of formats supported.

- [Customizing diagram options](#)^[283]
- [Exporting diagram as a graphical image](#)^[284]



9.6.1 Customizing diagram properties

To build a diagram in [Diagram Viewer](#), you should select the source field(s) to be represented in the diagram first. Only numeric types of fields can be used in the diagram, and each selected field corresponds to a separated diagram series. Fields are selected by checking items in the third combo box from the top in the [Database](#) group of the [Navigation Bar](#). If the combo box is empty then either data source is not yet selected or it contains no numeric fields.



[Diagram Viewer](#) provides a special control for customizing the diagram properties. This control is located in the **Properties** group of the [Navigation Bar](#) and consists of four separate subgroups:

Appearance

Contains properties responsible for major diagram appearance:

- **Chart type** - defines a way of how the diagram will be represented: as bars, lines, areas, points, pies, or fast lines
- **Color each points** - if checked, each bar, point, line or sector of the diagram has an individual color; if not checked, all the points are colored red
- **Show axis** - defines if the diagram has the axis and background grid or not

Data

Contains the **Labels source** property which allows you to specify the field for X-axis labels as well as for diagram pointmarks .

Titles

Contains properties for defining titles for different parts of the diagram:

- **Header** - defines the title appeared on the top of the diagram
- **X-axis** and **Y-axis** - define the titles for diagram axis
- **Show marks** - defines if the diagram point marks are visible or not

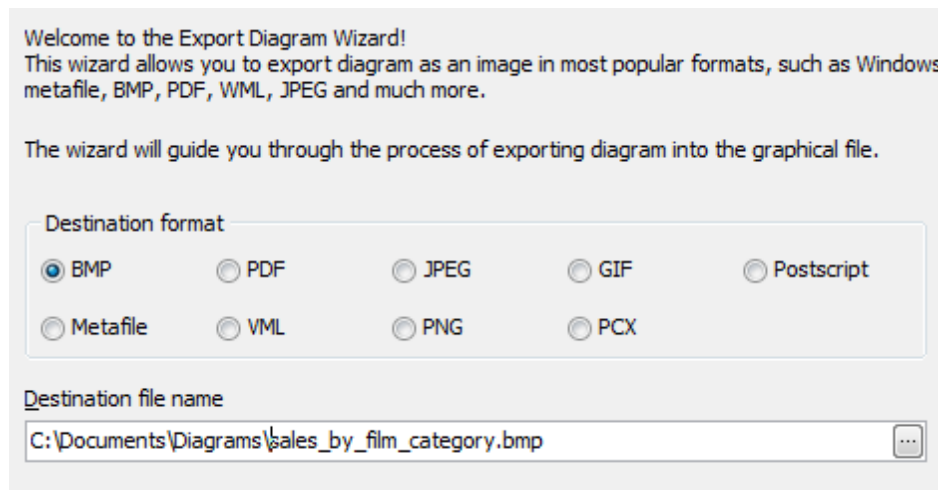
Legend

The only **Visible** property of this subgroup specifies whether the legend rectangle should be represented on the right side of the diagram or not.

9.6.2 Exporting diagram image

Diagram Viewer provides an ability to export current diagram to a file as graphical image. This ability is constituted in **Export Diagram Wizard** which can be invoked by the **Export diagram image** item of the **Navigation Bar**.

Select the desired graphical format in the **Destination format** radio group and specify the file name in the **Destination file name** box.



Welcome to the Export Diagram Wizard!
This wizard allows you to export diagram as an image in most popular formats, such as Windows metafile, BMP, PDF, WML, JPEG and much more.

The wizard will guide you through the process of exporting diagram into the graphical file.

Destination format

☒ BMP
 ☐ PDF
 ☐ JPEG
 ☐ GIF
 ☐ Postscript
 ☐ Metafile
 ☐ VML
 ☐ PNG
 ☐ PCX

Destination file name

C:\Documents\Diagrams\sales_by_film_category.bmp

Set the destination width and height by the corresponding spin edits. Check or uncheck the **Keep aspect ratio** option to keep the image ratio for exported image or not. Check the **Open exported image in associated program** option to view the image after the export is done.

Image size

Width Height

☒ Keep aspect ratio

☐ Open exported diagram in associated program

Click "Ready" to export the diagram.

9.7 Data Analysis

Data Analysis is a tool to define a multidimensional model with analytic calculations to analyze information also called OLAP cube. Such cubes could effectively be re-oriented. So the tool allows you to view data in various ways, such as displaying all the cities down the page and all the products across a page and then immediately view it in another way. Because this re-orientation involves re-summarizing very large amounts of data, this new view of the data has to be generated efficiently to avoid wasting the analyst's time, i.e. within seconds, rather than the hours a relational database and conventional report-writer might have taken. It allows you to focus on business rules rather than creating dozens and dozens of reports. To run Data Analysis, choose [Tools | Data Analysis](#) main menu item.

To get an OLAP cube:

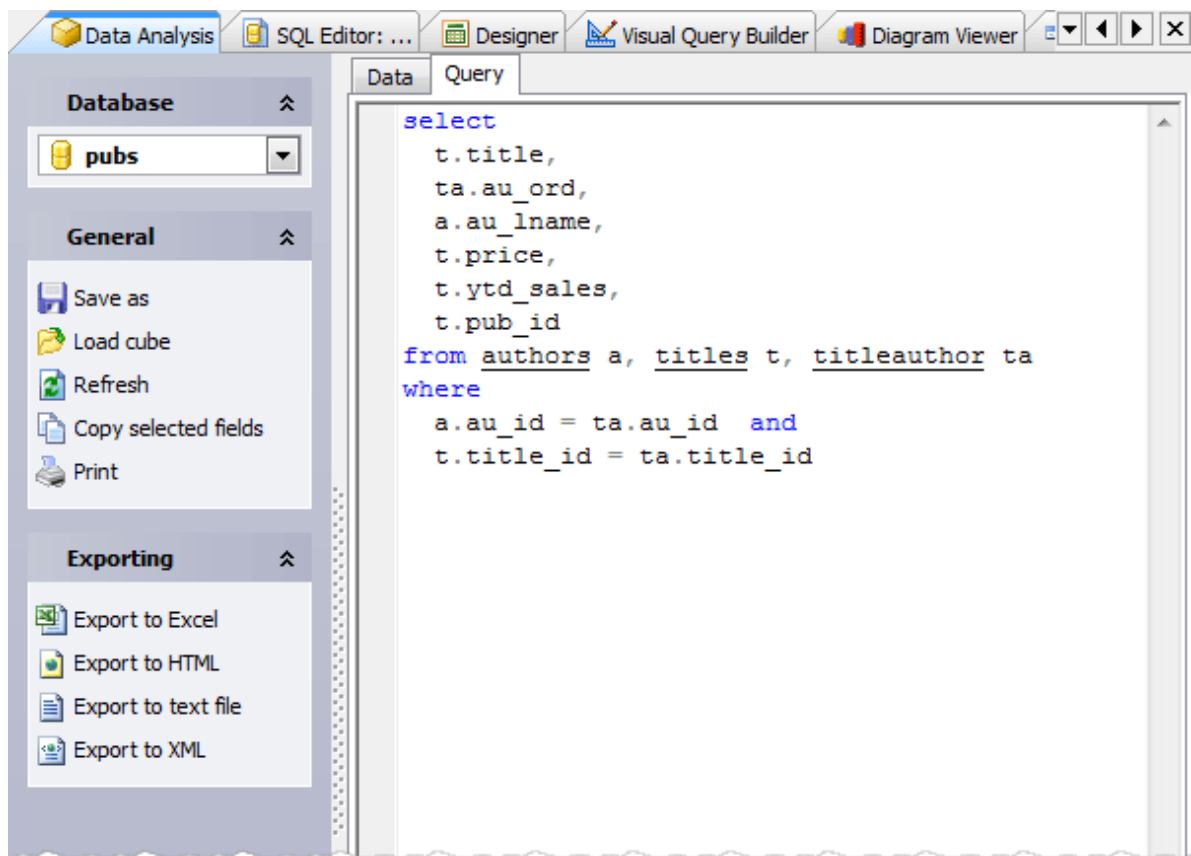
- [Input SELECT query](#)²⁸⁷ in the [Query](#) window or load it from the .cub file.
- [Manage report data](#)²⁸⁸ in the [Data](#) window.

The Data Analysis tool allows you to save the created OLAP cube to .cub file, print it, or export it to Excel, HTML, XML, and text file.

Data		Query			
au ord		vtd sales			
price		pub id			
au lname	title	0736	0877	1389	Grand Total
<input type="checkbox"/> Bennet	The Busv Executive's Database Guide			19,99	19,99
<input type="checkbox"/> Blotchett-Halls	Fiftv Years in Buckingham Palace Kitchens		11,95		11,95
<input type="checkbox"/> Carson	But Is It User Friendly?			22,95	22,95
<input type="checkbox"/> DeFrance	The Gourmet Microwave		2,99		2,99
<input type="checkbox"/> Dull	Secrets of Silicon Valle			20,00	20,00
<input type="checkbox"/> Green	The Busv Executive's Database Guide			19,99	19,99
	You Can Combat Computer Stress!	2,99			2,99
Green Total		2,99		19,99	22,98
<input type="checkbox"/> Grindlesbv	Sushi. Anyvone?		14,99		14,99
<input type="checkbox"/> Hunter	Secrets of Silicon Valle			20,00	20,00
<input type="checkbox"/> Karsen	Computer Phobic AND Non-Phobic Individuals: Beh		21,59		21,59
<input type="checkbox"/> Locksley	Emotional Security: A New Algorith	7,99			7,99
	Net Etiquette				
Lockslev Total		7,99			7,99
<input type="checkbox"/> MacFeather	Computer Phobic AND Non-Phobic Individuals: Beh		21,59		21,59
	Cooking with Computers: Surreptitious Balance Sh			11,95	11,95
MacFeather Total			21,59	11,95	33,54
<input type="checkbox"/> O'Leary	Cooking with Computers: Surreptitious Balance Sh			11,95	11,95
	Sushi. Anyvone?		14,99		14,99
O'Leary Total			14,99	11,95	26,94
<input type="checkbox"/> Pantelev	Onions. Leeks. and Garlic: Cooking Secrets of the		20,95		20,95
<input type="checkbox"/> Ringer	Is Ander the Enemy?	21,90			21,90
	Life Without Fear	7,00			7,00
	The Gourmet Microwave		2,99		2,99
Ringer Total		28,90	2,99		31,89
<input type="checkbox"/> Straight	Straight Talk About Computers			19,99	19,99
<input type="checkbox"/> White	Prolonged Data Deprivation: Four Case Studies	19,99			19,99
<input type="checkbox"/> Yokomoto	Sushi. Anyvone?		14,99		14,99
<input type="checkbox"/> del Castillo	Silicon Valle Gastronomic Treats		19,99		19,99
Grand Total		59,87	147,02	146,82	353,71

9.7.1 Input SELECT query

To get an [OLAP cube](#)²⁸⁸, enter SELECT query as a snowflake schema, represented by centralized fact tables (with numeric data) which are connected to multiple dimensions (the numeric data to be categorized by). Input the query text in the SQL Editor area directly or use "drag-n-drop" operation [SQL Editor](#) or Visual Query Builder areas and the Query tab of Data Analysis.



9.7.2 Managing report data

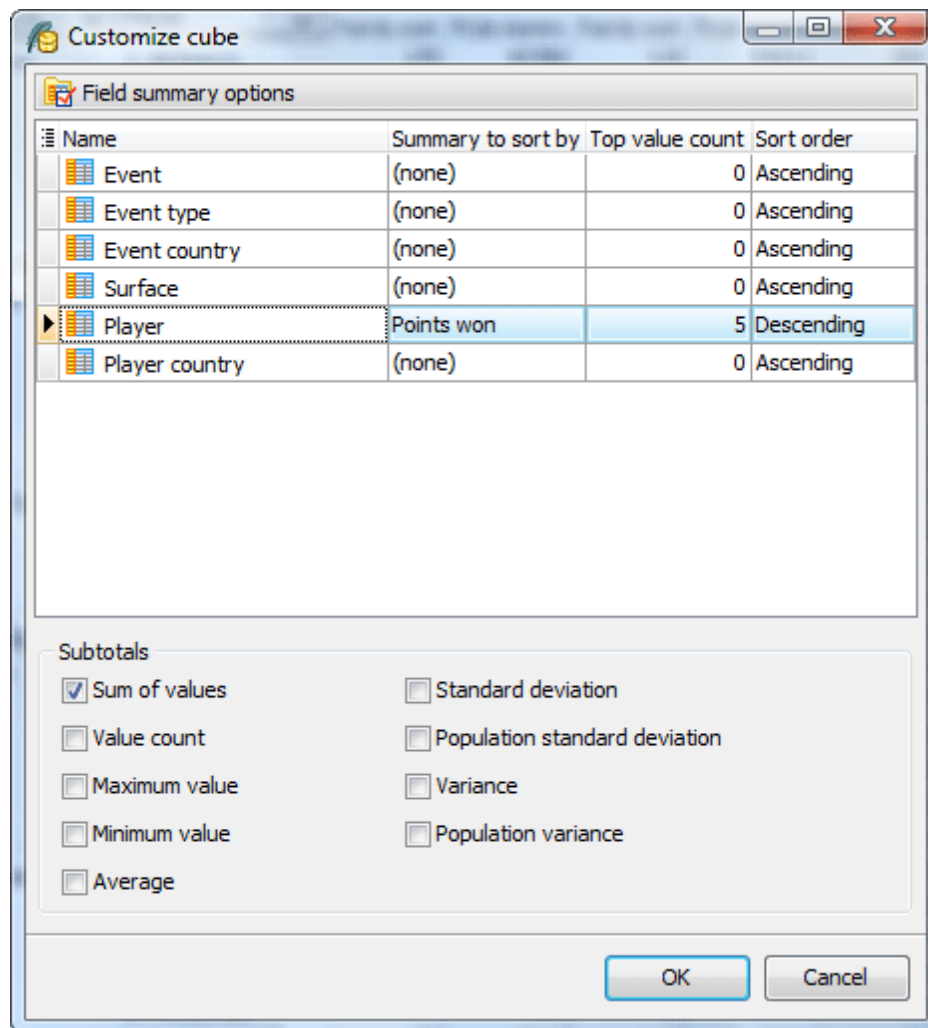
The [Data](#) tab allows you to manipulate the created OLAP cube appearance. At the beginning all the [query](#)^[287] columns are arranged at the top of the tab. Put them according to your business rules: drag numeric columns to be filtered and summarized corresponding to the chosen columns and rows to the [Data Fields](#) area; place necessary columns to [Column Fields](#) / [Row Fields](#) areas respectively.

Note: Use for the Data Fields area only numerical columns.

The screenshot shows a database query tool interface. At the top, there are tabs for 'Data' and 'Query'. Below the tabs, there is a query grid with columns: 'title', 'au', 'ord', 'vtd', 'sales', and 'pub id'. The 'pub id' column is highlighted. Below the query grid, there is a 'Drop Column Fields Here' area. Below this area, there is a table with columns 'au lname' and 'Grand Total'. The table contains 20 rows of data, including a 'Grand Total' row at the bottom. The 'au lname' column lists author names, and the 'Grand Total' column shows the sum of sales for each author.

au lname	Grand Total
Bennet	19,99
Blotch-Halls	11,95
Carson	22,95
DeFrance	2,99
Dull	20,00
Green	22,98
Grindlesbv	14,99
Hunter	20,00
Karsen	21,59
Lockslev	7,99
MacFeather	33,54
O'Learv	26,94
Pantelev	20,95
Ringer	31,89
Straight	19,99
White	19,99
Yokomoto	14,99
del Castillo	19,99
Grand Total	353,71

To set the aggregates calculated on the numeric columns, use the [Customize cub](#) window opened with the corresponding link at the Navigation bar. The window provides you also with an ability to specify columns the summary to be sorted by, the sort order and the max number of records represented in grid.



9.8 Report Designer

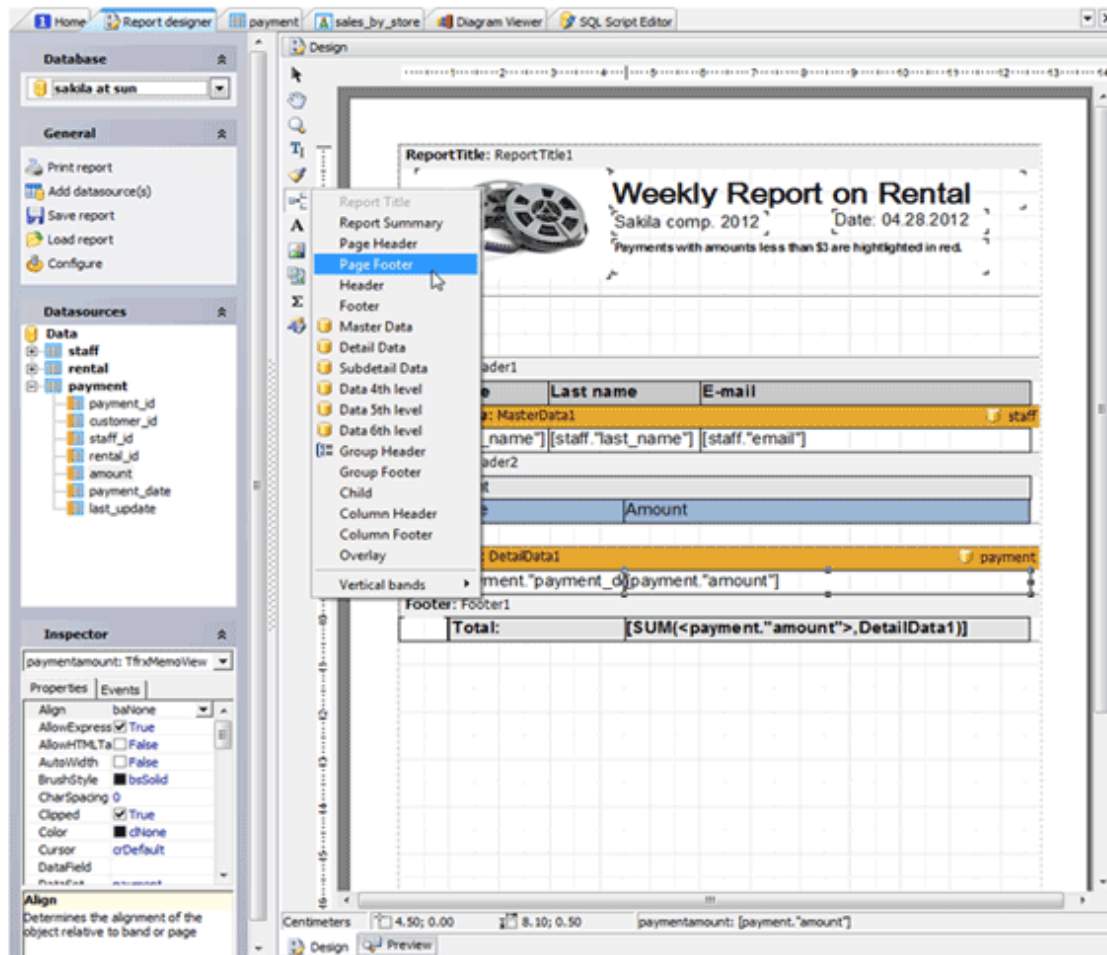
Report Designer allows you to create database reports, define reports appearance in your own style, equip it with master-detail data views, aggregate functions, and images and control the result with the ability of simultaneous previewing. To run Report Designer, choose [Tools | Report Designer](#) main menu item.

To create a report, you need to:

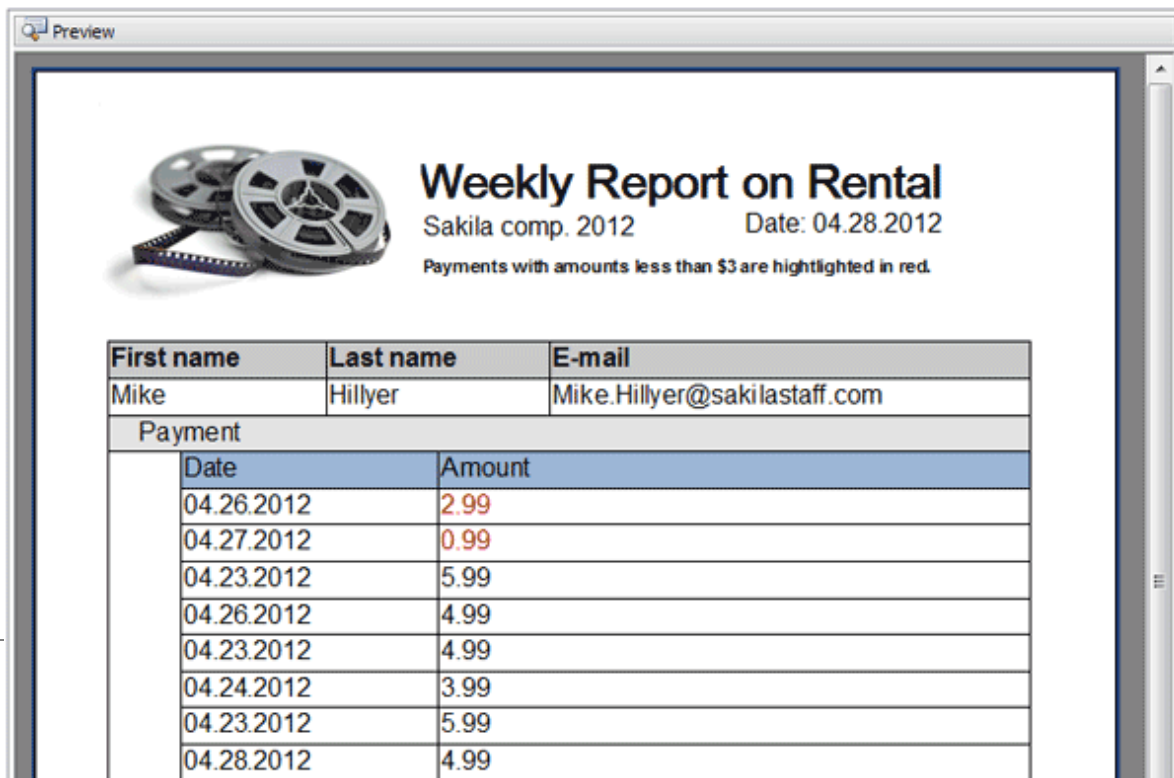
- [specify data sources](#)^[293] for the data to be used in the report;
- [add all necessary objects](#)^[293] to the report template;
- set the objects' format within the [Inspector window](#)^[295].

The prepared report pages are available immediately at the [Preview](#) window where you can browse it, save to it an .smr file, or print.

Report Designer in Action



Report Preview



9.8.1 Designer Tools and Objects

A blank report is presented as a paper page. At any place on the page, a user is able to add objects, which can display different information (such as text and/or graphics), as well as to define report's appearance. There is a possibility to use rulers and a grid with a specified size in the Design tab. To enable/disable these options, follow the [Configure](#) link at the [Navigation Bar](#) and check the corresponding boxes.

Datasources

To use content of a table (view) column data in a report,

- check the necessary database is selected as Database at the Navigation Bar;
- drag the table which data to be used in the report to the Datasources pane at the Navigation Bar;
- drag the necessary column from this pane and drop it to the necessary location on a report page.

Designer tools:

Select tool

The standard tool to select objects, modify their sizes, etc.

Hand tool

The tool allows dragging a report page.

Zoom tool

When the button is pressed, clicking on the left button doubles the zoom (adds 100%), while clicking the right one zooms out by 100%. When holding the left mouse button while dragging, the selected area would be zoomed.

Edit text tool

Clicking on the text object allows editing its contents right on the report page. If you hold the left mouse button when moving the cursor, the text object appears in the selected place, and then its editor launches.

Copy format tool

The button becomes enabled when the text object is selected. When clicking on the text object with the left button, it copies formatting, which has the previously selected text object, into the object.

Available objects:

[Band objects](#) allow to specify where, when, and how to display data and information in reports. Bands are used for logically placing the objects it contains at a location on the output page. Insert Band adds an area with definite behaviour according to its type such basic bands as Header, Footer, Title, and Summary, and databands whose allow to print data from database tables such as Master Data, Detail Data, etc.

[Text object](#) displays one or several text lines within the rectangular area.

[Picture object](#) displays a graphic file in BMP, JPEG, ICO, WMF, or EMF format.

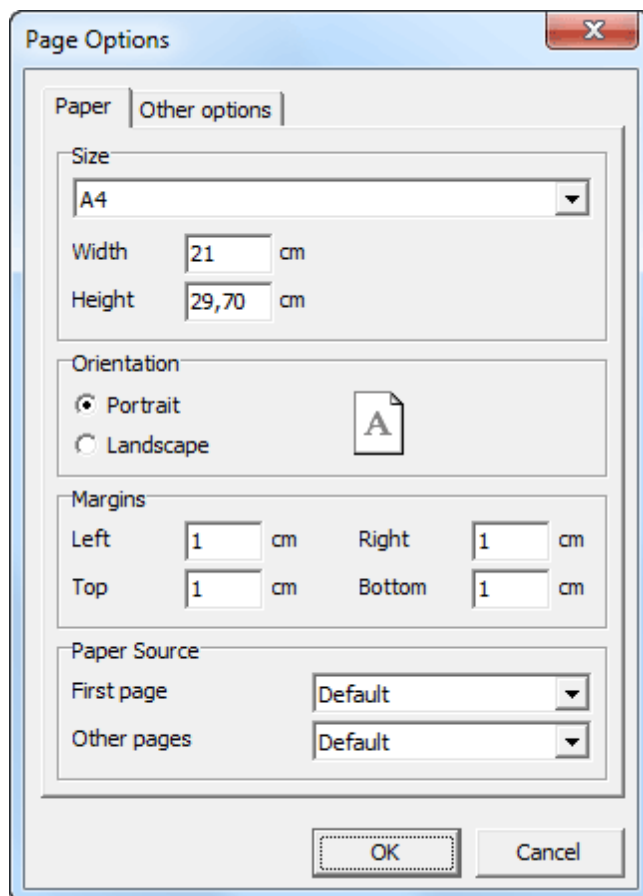
[Subreport object](#) allows inserting an additional report design page inside the basic one.

[System text](#) displays service information (date, time, page number, etc), as well as aggregate values.

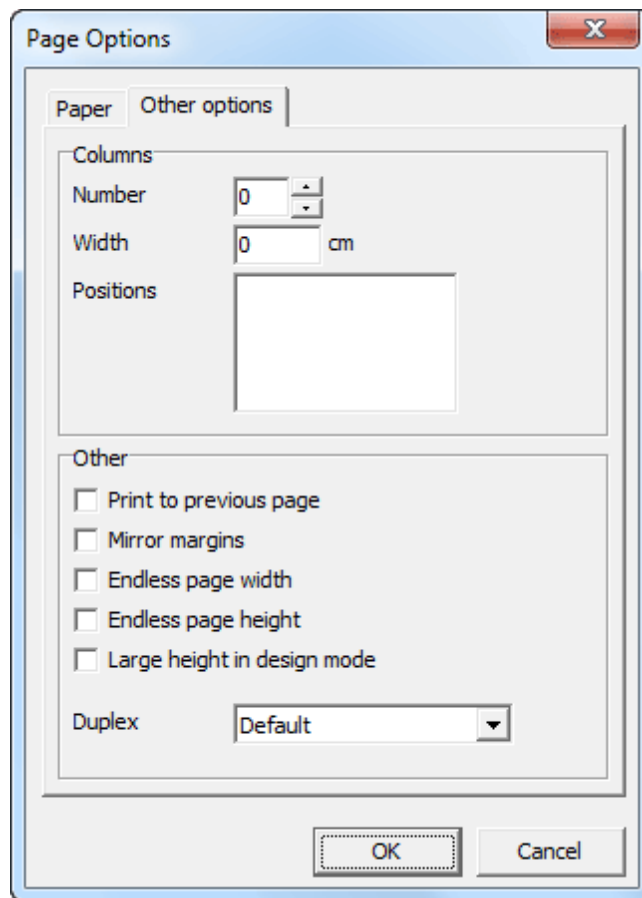
[Draw object](#) represents different geometrical figures (diagonal line, rectangle, rounded rectangle, ellipse, triangle, and diamond).

Page options

This dialogue allows you to set the page settings of the prepared. To invoke the window, use the [Edit...](#) link of the page blank space popup menu. The dialogue has two pages: [Paper](#) and [Other options](#). On the [Paper](#) page, you can select size and alignment of paper, as well as set margins. In [Paper source](#) drop-down lists you can select a printer tray for the first page and the rest of the report pages.



On [Other Options](#) you can set the number of columns for multi-column reports' printing. The current settings are displayed in the designer.



The [Print to previous page](#) flag allows you to print pages, beginning from blank space of the previous page. This option can be used in case when a report template consists of several pages or when printing batch (composite) reports.

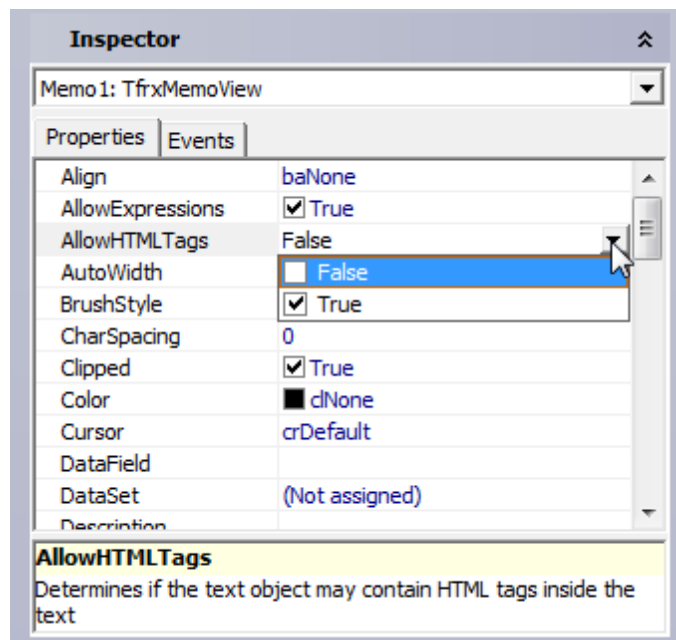
The [Mirror margins](#) option switches right and left margins of page for even pages during previewing or printing a report.

The [Endless page width & height](#) option increases page's sizes depending on number of data records on the page (when running a report). In this case you will see one big page in the preview window instead of several standard size pages.

The [Large height in design mode](#) option increases page's height several times more. This feature can be useful if many bands are located in the page, and must be used when working with the overlay band. This only effects the page height in design mode.

9.8.2 Object Inspector

[Object Inspector](#) pane allows you to specify the appearance of each report object in detail. To setup object properties, select it at the Design area or select it from the popup menu at the top of the pane. Now all the properties of the object are available for editing. The most of properties are provided by a set of available values. The description of the selected option is displayed at the bottom of the pane.



Below you can find a brief description of several options.

Align - set here the align option of the object according to the list.

AllowExpressions - enables the ability to display not only a static text, but expressions as well.

AllowHTMLTags - Enables using some simple HTML tags inside the text of an object. This option is disabled by default. Here is the list of supported tags:

 - bold text;

<i> - text in italic;

<u> - underlined text;

<sub> - subscript;

<sup> - superscript;

 - font color;

<nowrap> - text which does not get broken up when using **WordWrap**, but gets transported wholly.

Font: there are abilities to specify the charset, font color, font name, and font size, and also set the bold, italic, underline, strike out attributes.

Frame: You can set as the color, the style and the shadow for all the frame, as well as for each frame line.

BrushStyle - type of object filling.

CharSpacing - space between symbols in pixels.

GapX, GapY - text indents from object's left and top boundaries (in pixels).

LineSpacing - space between lines (in pixels).

ParagraphGap – the first paragraph line indent (in pixels).

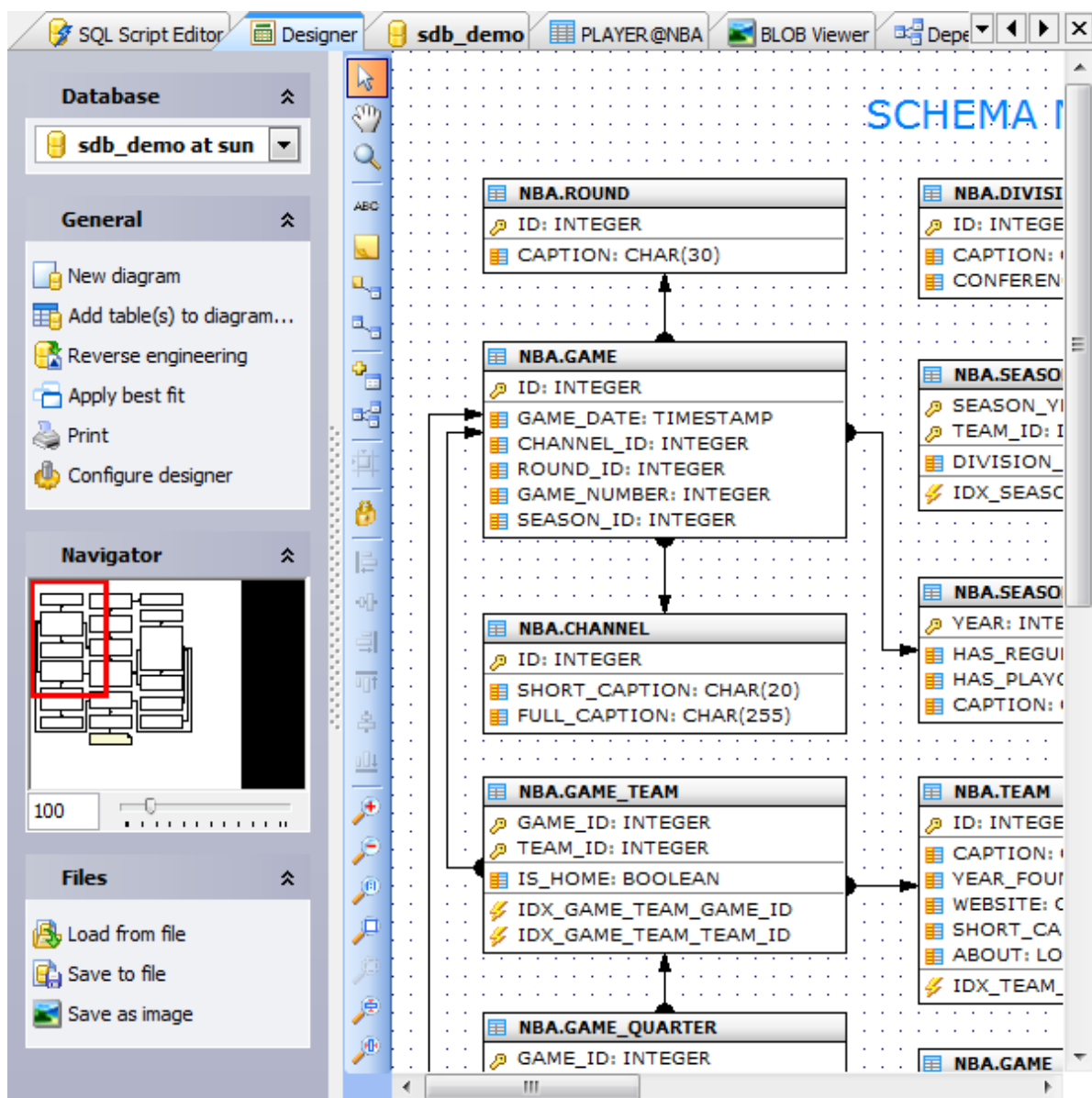
Rotation - specify the angle of the text rotation in the field.

WordWrap - if it is disabled, the long lines will be simply cut off.

9.9 Schema Designer

Schema Designer allows you to create physical Entity Relationship Diagram that will represent objects in your PostgreSQL database. A diagram represents the tables of your database and the relationships between them. The tool is intended for reverse engineering and database modification in an easy and powerful way. It helps you to simplify database maintenance.

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.



Below you can find answers for the following questions:

- [How can I add a table to a diagram?](#)^[299]
- [How can I add a relationship between tables?](#)^[299]
- [How can I delete a diagram object?](#)^[299]
- [How can I work with diagram objects?](#)^[300]

See also: [Designer Navigation bar](#)^[300], [Schema Designer Toolbox](#)^[300].

Adding a table

You can add an existing PostgreSQL table to the diagram using popup menu in the working area, or with the corresponding link on the [Navigation bar](#)^[300].

To create a new table, use the appropriate item of the popup menu in the working area. The table will be created in the current database.

Tables also may be dragged on the diagram from [Explorer](#) and the similar to [Explorer](#) tools like [Object Manager](#) and [Object Browser](#).

Moreover, [Designer](#) provides you with a possibility to represent all the tables and relationships existing in the database automatically (the [Reverse engineering](#) link of the [Navigation bar](#)). At that the database contents will be represented on a diagram in the most compact and vivid manner.

All the diagram objects are available for editing. Just double click the object (table or relationship) to view/edit its properties within the corresponding editor.

Adding a relationship between tables

At adding to a diagram tables that reference on each other, the relationships between tables are represented automatically. The [Schema Designer](#) tool also allows you to add new foreign keys to the diagram tables. Thereto you can do the following.

Select a table (child table)

- Use the [Create new...](#) item of the popup menu to launch [Foreign Key Properties](#)^[88] window.
- Specify there properties of the relationship been created.

Moreover you can add a reference graphically:

- Choose the [Create relation](#) tool on the [Toolbox](#)^[300]. Your mouse cursor will change its appearance.
- Then click on the table (child table) that will have foreign key and then click on the second table (parent table) whose primary key will be referred by the new foreign key.
- Specify properties of the relationship been created in the [Foreign Key Properties](#)^[88] window.

With the [Create new...](#) item of the popup menu you can also add a new field, an index, a trigger, etc. to the selected table. For more information about object properties see: [Field Editor](#)^[82], [Index Editor](#)^[84], [Create Trigger Wizard](#)^[97].

Deleting of the diagram objects

To hide a table (several tables) or a relationship between tables, select the objects and click [Remove selection](#) link of the popup menu or [Navigation bar](#). You can also use the **Del**

key for this purpose.

It's also possible to physically delete a table/foreign key from the database: just select the object to delete and use the appropriate item of the popup menu.

Editing of a diagram appearance

Movement of a table/several tables along the diagram is realized with dragging or pressing **Ctrl**+arrows. You can use **Shift**+arrows to change width/height of table/several tables representation.

[Designer](#) also allows you to edit shape of the line representing foreign key relations/ logical relations. In order to break the line you should

- Select the relationship.
- Press **Ctrl** and click on the necessary line section to create a new node.
- Position the node by dragging.

You can also delete a node on the line. Thereto

- Select the relationship.
- Press **Alt** and click the node to delete. In that case the near nodes will be united by a straight line.

9.9.1 Designer Navigation Bar

The [Navigation Bar](#) of [Schema Designer](#) provides you the following opportunities:

Use the [Database](#) drop-down list to move around your PostgreSQL databases.

There are also links for adding a [New diagram](#) or an existing [table to diagram](#) quickly.

[Reverse engineering](#)

The link provides you to create a new diagram with all the database tables and

[Apply best fit](#)

Use the link to dispose tables on the diagram in the most clear manner.

[Remove selection](#)

The link cancels current object selection.

Use [Print](#) to see the print preview of the diagram.

Certainly, it's possible to customize [Schema Designer](#) with [Configure designer](#). For more information see [Schema Designer Customization](#)^[334].

The [Navigator](#) part allows you to adjust the scale of the diagram and the position of the visible part.

Besides the [Navigation bar](#) allows you to [Load](#) a diagram from file, [Save to file](#), and [Save as image](#) (Bitmap, GIF and JPEG formats are supported).

9.9.2 Schema Designer Toolbox

The toolbox is located on the left side of the [Schema Designer's](#) working area.



Move

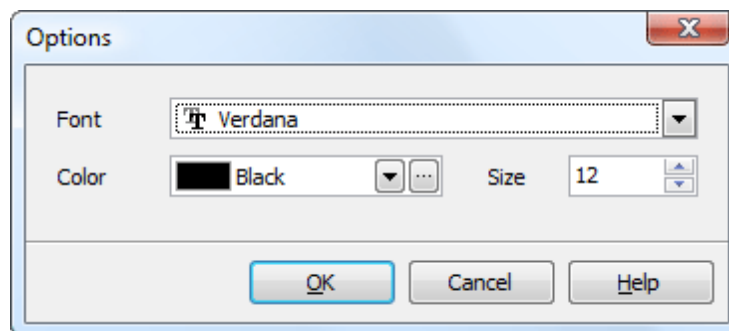
The tool is intended for selection of diagram objects. Use the tool then click anywhere inside of the object. Double click opens the corresponding [Object Editor](#).

To select multiple objects, use the tool then click and drag a selection rubber-band so that the rubber-band box encompasses the objects you want to select, and then release the mouse button.

To add objects to the list of already selected objects again, use the Move tool then click anywhere inside of the object holding the **Shift** button. To quick launch of the tool, use **M** shortcut.



Use [Create text box](#) to add [title and comments](#) on your diagram. Click on the necessary place and double-click on the appeared box to enter a text. You can also tune up the text font, color and size with [Text options](#) of the box popup menu. To quick launch of the tool, use **XX** shortcut.



Moreover you can add notes and also links between them and diagram elements using



[Create note](#) and [Create link to note](#) links. To quick launch of the tools, use **N** and **L** shortcuts accordingly.



Lock



The tool to locking/unlocking diagram objects. This feature prevents your diagram from unforeseen changes: when the diagram is locked, you can neither move/resize/delete existing objects nor add new ones.



[Hand](#) tool moves a diagram within its window. To quick launch of the tool, use **H** shortcut.



[Zoom](#) magnifies and reduces the view of a diagram. To zoom out, hold the Alt key. To quick launch of the tool, use **Z** shortcut.

There are also tools allowing to  [Create table](#) and  [Create relation](#) directly from the [Designer](#). To quick launch of the tool, use **T** and **R** shortcut.

Below you can find toolset for aligning the selected objects by left and right edges, by horizontal and vertical centers, tops and bottoms.

Click the [Zoom in](#) button in the options bar to magnify to the next preset percentage.

When the image has reached its maximum magnification level, the command is dimmed.

Click the [Zoom out](#) button in the options bar to reduce to the previous preset percentage. When the image has reached its maximum reduction level, the command is dimmed.

Click the [Zoom 1:1](#) button to display a diagram at 100%.

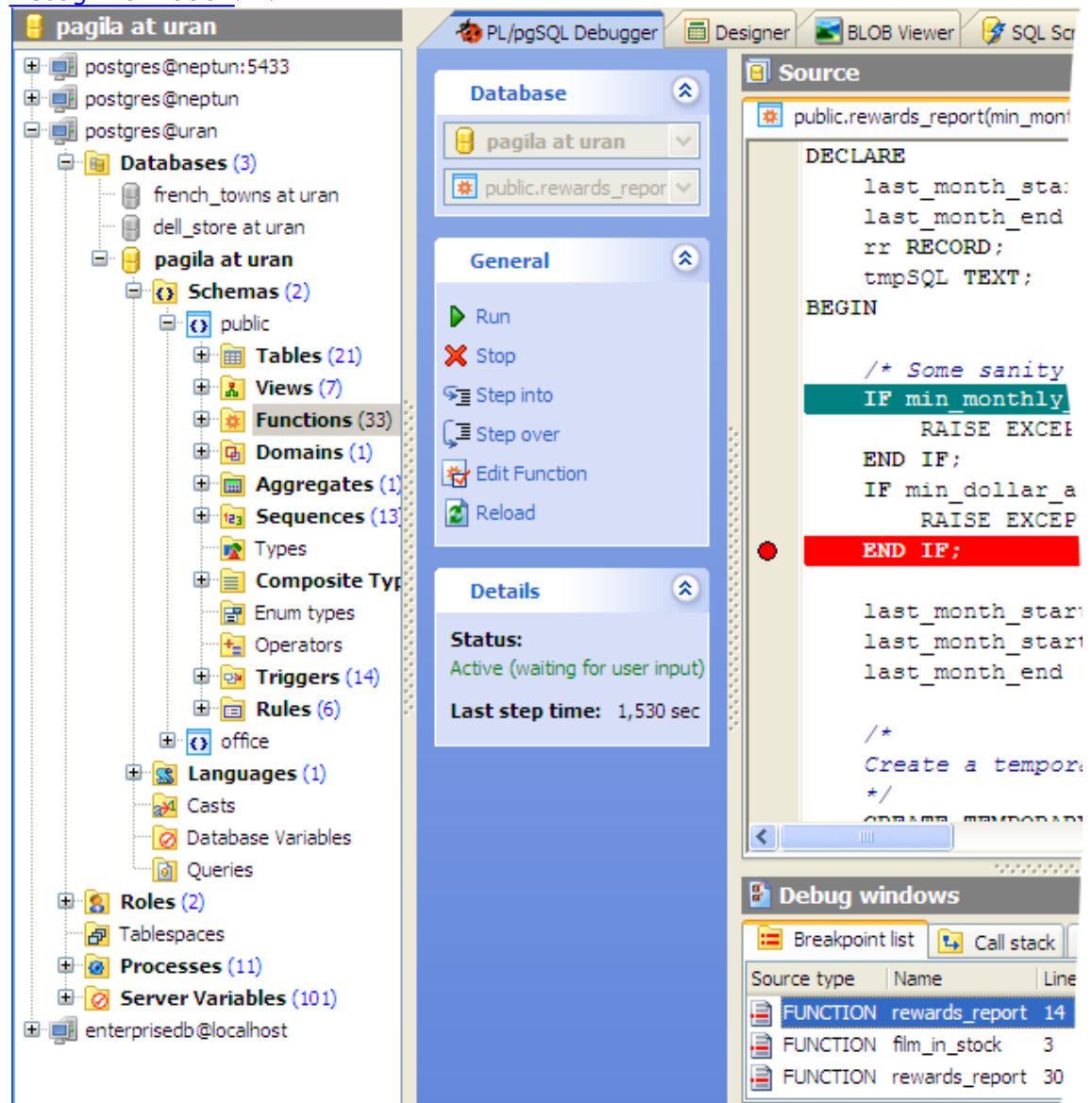
Pay attention to the [Fit diagram](#) function, that pick-up properly scaling factor to display your diagram fully. For your convenience the [Fit selected](#), [Fit height](#), and [Fit width](#) were added.

9.10 PL/pgSQL Debugger

The [PL/pgSQL Debugger](#) tool allows you to debug PL/pgSQL functions written on PL/pgSQL using traditional debugging features such as setting breakpoints, viewing variable values, and examining the call stack.

To start working with the debugger, choose the [Tools | PL/pgSQL Debugger](#) main menu item. You can also use the corresponding link at the [Navigation bar](#) of the [Function Editor](#)^[128] or select the appropriate command from the object pop-up menu in the [Explorer tree](#).

- [Debugging process](#)^[304]
- [Debug information](#)^[306]



9.10.1 Debugging process

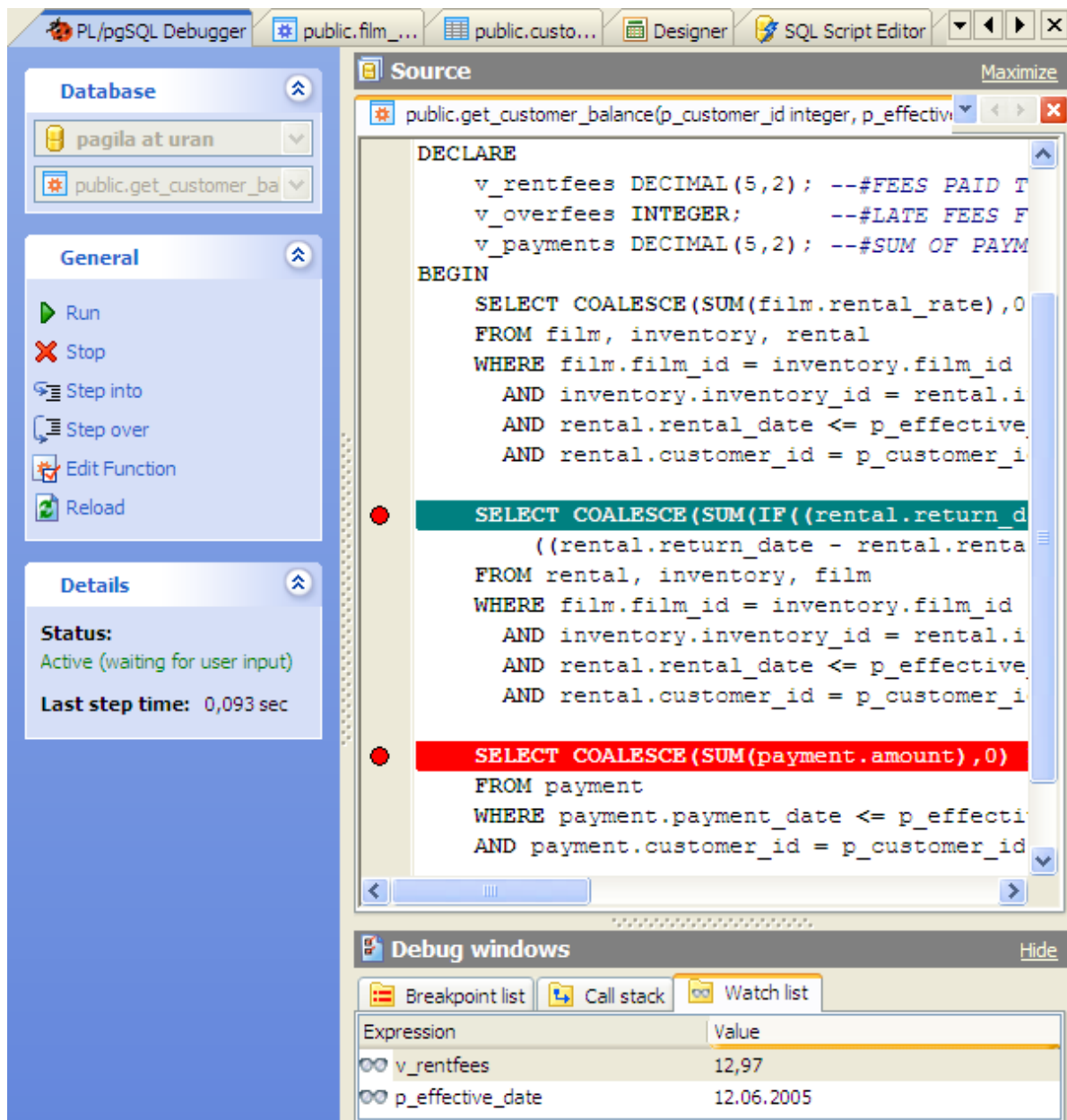
For those programming errors that are really hard to track, PostgreSQL Maestro provides an integrated [PL/pgSQL](#) debugger. The main principles of the debugging process within the [PL/pgSQL Debugger](#) are similar to the same tools methods.

The [PL/pgSQL Debugger](#) window consists of three parts: [Source](#), [Debug window](#), and [Navigation bar](#).

The [Source](#) tab contains script editors for each program unit, so that you can easily switch between them to view the source, set/remove breakpoints, and so on.

The [Debug window](#) is provided to represent the debugging process characteristics such as [Breakpoints](#) information, [Call Stack](#), and [Watch List](#).

- [Controlling execution](#) ³⁰⁵
- [Using breakpoints](#) ³⁰⁶
- [Debug information](#) ³⁰⁶



Controlling execution

You can set the procedure parameters before the debugging process.

To start the debugger, use the **Run (F9)** link at the **Navigation bar**.

After starting the debugger, execution will pause on the first script statement. After this, you can control execution with the buttons in the **Navigation bar**:

Use the **Run (F9)** link at the **Navigation bar** to run the script until completion, a breakpoint or an exception.

To move to the next line, click the **Step Over** icon or use **F8**. The nested procedure will be executed, but you will not step into the source.

[Step into \(F7\)](#) allows you to open the nested procedure as the new [Source](#) instance to debug it line-by-line.

Note: The nested procedure body is available to debug if it was [compiled with debug information](#).

[Stop \(Ctrl + F2\)](#)

Use the link to abort the debugging process. To make changes in the procedure body, use the corresponding [Procedure Editor](#).

Using Breakpoints

Breakpoints can be used to halt program execution on a certain line in your [PL/pgSQL](#) code. When execution halts, you can view variables, step through the code, and so on.

To set a breakpoint, just click on the grey leftmost edge of the code window opposite to a necessary line. To delete a breakpoint, simply click on the breakpoint mark again.

It's also possible to disable/enable a breakpoint by clicking the breakpoint indicator with pressed **Ctrl**.

9.10.2 Debug information

[Breakpoint list](#)

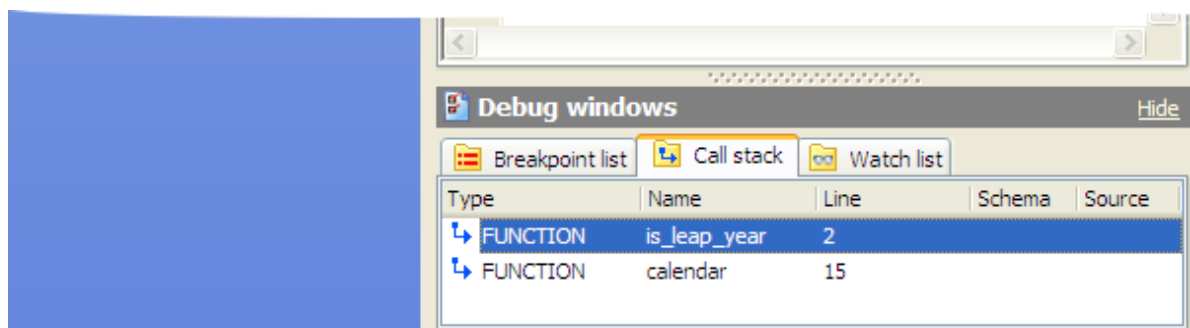
The tab contains all the information about the session breakpoints.

[Viewing variable values](#)

If you want to view variables values, you need to create a watch with the [Watch list](#) tab pop-up menu. Just select [Add Watch](#) and enter the variable name.

[Viewing the call stack](#)

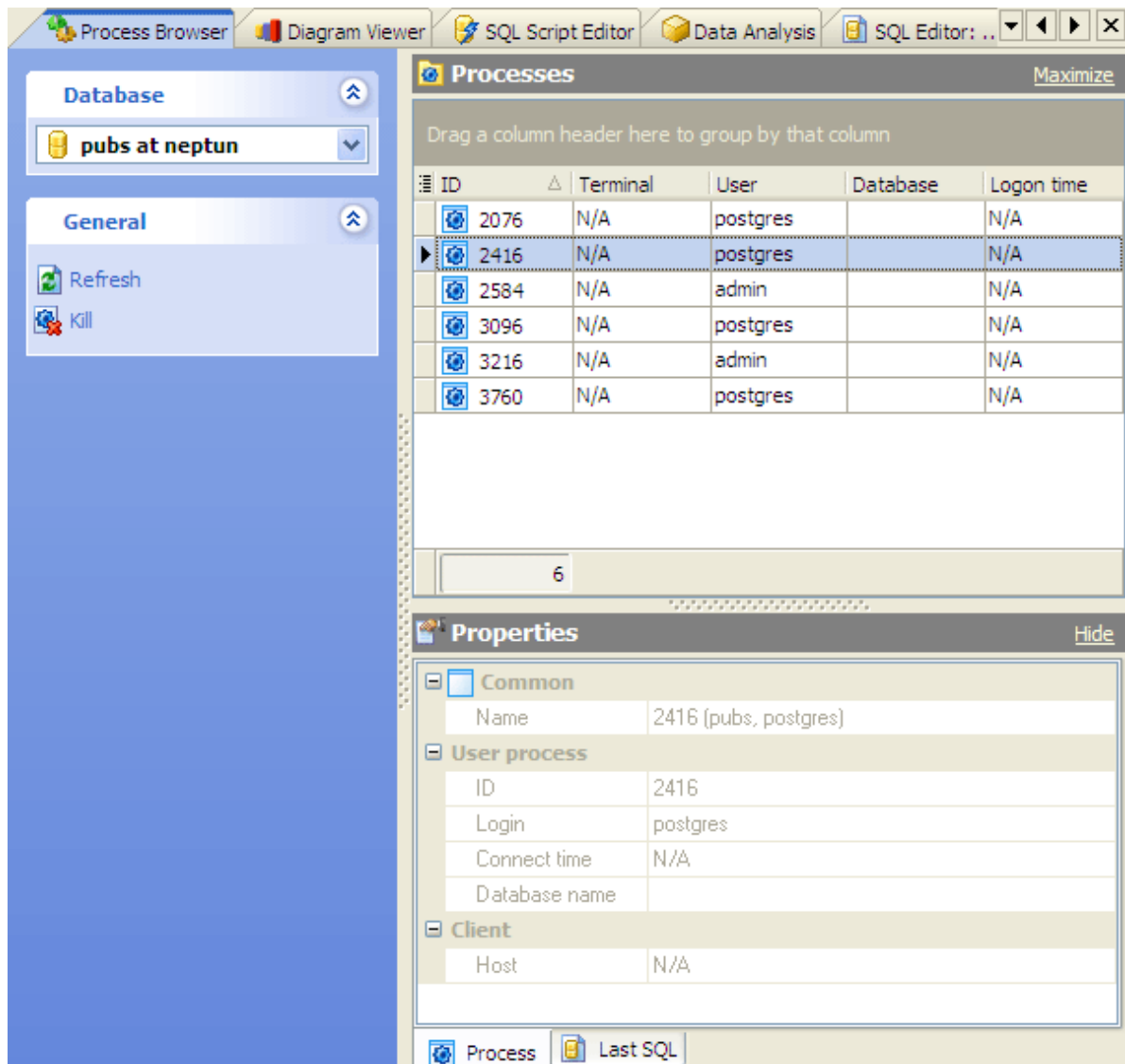
Whenever the debugger is paused, you can use the [Call stack](#) window to see the procedure flow as a stack of method calls that got you to the current line. It is automatically updated after each debug step.



9.11 Process Browser

Process Browser is very useful tool for DBAs who want to monitor the users' activity (in fact, there are potentially thousands of sessions in a database at any one time). You can view details for each session (such as login, connect time, database name, client host, last SQL statement executed and more) as well as group and filter processes by client host, connected user, database, client application, etc.

To access the **Process Browser** window, select the corresponding item from the **Tools** menu.

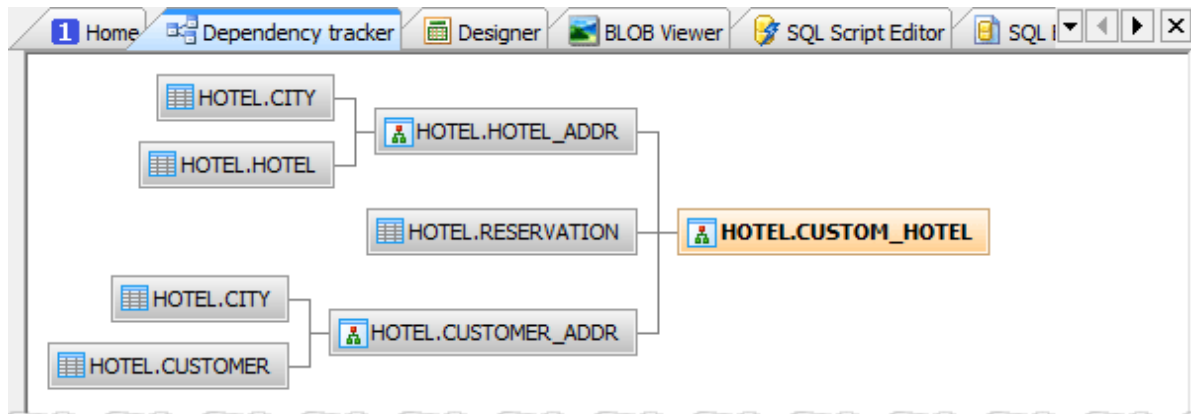


9.12 Dependency tracker

Dependency tracker is a tool to browse all-level dependencies of a schema object (table, view, function, etc). To display dependencies of an object, drag and drop it from the Explorer tree (or Object Manager, Object Browser) to the tracker's working area.

This tool allows you to see the way any database object is involved in the net of scheme dependencies. The selected object is displayed in a highlighted rectangle in the center of the working area. The right side of this area represents objects depending on the selected one. The left side represents objects on which the selected object depends.

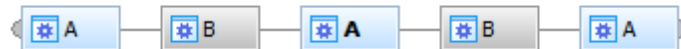
To highlight all occurrences of an object in the diagram, click the appropriate rectangle. Double click an object to change the tracker focus to this object. Right click a rectangle to display a popup menu with common operations related to the appropriate object.



The recursive dependencies are marked with a semicircle. This means marked object depends on itself directly or via other objects.

Example

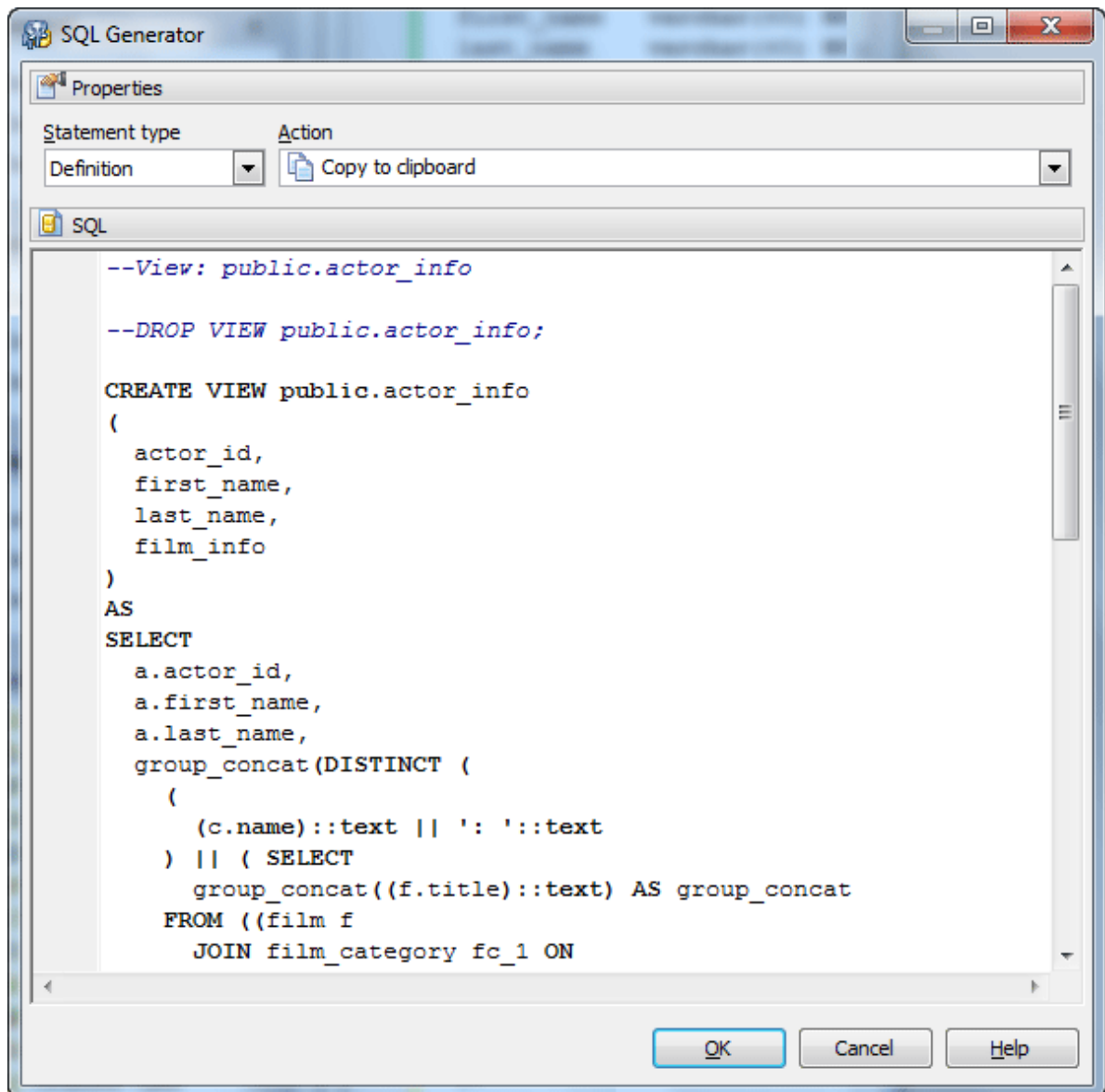
Suppose we have two procedures A and B calling each other. The tracker displays these dependencies in the following way:



9.13 SQL Generator

Among other features PostgreSQL Maestro provides you with SQL Generator, a tool to create simple SQL statements. Just choose a database object, select statement type (Definition, Select, Insert, Update, or Delete) and the destination device (Clipboard, File, SQL Editor, SQL Script Editor).

The [SQL Generator](#) window can be invoked from the Explorer tree.



9.14 DML procedures generation

PostgreSQL Maestro allows you to create DML (also known as CRUD) procedures automatically. CRUD is an acronym for the four essential database operations: Create, Read, Update, and Delete. The application designer has many choices for accomplishing the CRUD operations but the most efficient choice in terms of PostgreSQL performance is to create a set of stored procedures to perform the operations.

The reasons for using DML Procedures instead of allowing ad hoc SQL statements are:

- The best possible performance

After the first use of each stored procedure, the plan for executing the procedure is cached in the server's procedure cache. For subsequent invocations of the stored procedure, the plan is reused. This avoids the parsing and optimization steps with their overhead.

- Removing of the SQL code from the other layers of the application

By removing the SQL statements from the application code, all the SQL can be kept in the database.

- Preventing of SQL injection attacks


Anytime a client application uses string concatenation to create SQL statements, there is a possibility of a SQL injection attack. In short, these attacks involve clever entry of SQL in the data entry fields of an application in such a way that the SQL statements executed are different from the ones intended by the programmer. They require that the application developer is careless about not cleaning any user input to prevent the attack.

- Preventing of casual table browsing and modifications

If an application uses ad hoc SQL statements, the users of the application must have the required permissions on the database tables. Once they are given permission on the tables, they can work with them in any application that can read and manipulate the data such as Excel, Word and various report writers. Casual examination of the data and even updates that bypass the application's business rules become possible. Stored procedures have long been used to prevent casual browsing and updates. This is implemented by granting permission to execute the CRUD stored procedures to the users and revoking permission to access the tables directly.

To generate DML procedure,

- select the [Object | Generate DML procedures...](#) main menu item (to create procedures for several tables) or use the corresponding popup menu item of the table's node at the Explorer tree (to create procedures for one table).
- Specify tables the procedures will be created for (in case of several tables).
- Uncheck the operations the procedures will not be created for. By default the procedures are generated for inserting, reading, updating, and deleting of table data.
- Adjust templates of procedures names.
- Select the action to perform after the generation. The created definitions can be copied to Clipboard, saved to a file, sent to SQL Script Editor or executed immediately.

 Options



Procedures to create

Select procedure	<input checked="" type="checkbox"/>
Update procedure	<input checked="" type="checkbox"/>
Insert procedure	<input checked="" type="checkbox"/>
Delete procedure	<input checked="" type="checkbox"/>

Naming

Select procedure name	sp_sel_%TableName%
Update procedure name	sp_upd_%TableName%
Insert procedure name	sp_ins_%TableName%
Delete procedure name	sp_del_%TableName%
Parameter name	p_%ColumnName%

Action to perform after generation

 Execute immediately 

9.15 Generation of updatable views

To generate updatable view,

- select the **Object | Generate updatable views...** main menu item (to create views for several tables) or use the corresponding popup menu item of the table's node at the Explorer tree (to create a view for one table).
- Specify tables the views will be created for (in case of several tables).
- Specify the abilities to be available on working with the view data. By default the views are generated for inserting, updating, and deleting of table data.
- Adjust the name templates of views and corresponding triggers.
- Select the action to perform after the generation. The created definitions can be copied to Clipboard, saved to a file, sent to SQL Script Editor or executed immediately.

Options

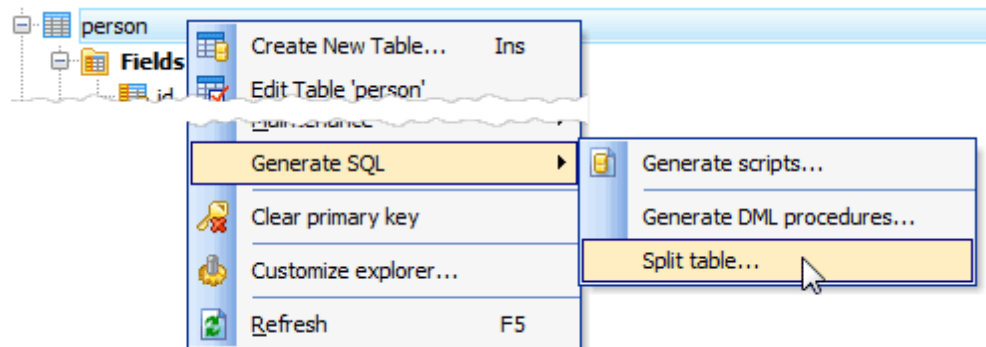
Abilities	
Insert	<input checked="" type="checkbox"/>
Update	<input checked="" type="checkbox"/>
Delete	<input checked="" type="checkbox"/>

Naming	
View name	V_%TableName%
Insert trigger name	TR_BI_%TableName%
Update trigger name	TR_BU_%TableName%
Delete trigger name	TR_BD_%TableName%

Action to perform after generation

9.16 Split table

It's not an uncommon situation when new requirements arise, or when you need to enforce referential integrity on a set of columns, and the best decision is to split a table into two separate tables. PostgreSQL Maestro provides you with [Split Table Wizard](#), a simple tool to generate a bunch of SQL scripts to modify the primary table, to create a secondary table with a primary key, and to transfer data from the primary table to the secondary one without duplicating of data. To invoke the wizard, follow the corresponding link of the [Generate SQL](#) section of popup menu of the selected table at the Explorer tree.



Let's see the wizard in action on the example of a table with the following SQL definition:

```
CREATE TABLE person (
  id          integer NOT NULL,
  city        varchar(30) NOT NULL,
  full_name   varchar(30) NOT NULL,
  /* Keys */
  CONSTRAINT person_pkey
    PRIMARY KEY (id)
);
```

The table stores sample data:

	id	city	full_name
1	1	New York	John Smith
2	2	Boston	Mary Doe
3	3	Boston	Jason Lee
4	4	New York	Deisy O'Connor

To enforce the referential integrity, we specify 'city' as secondary table:

Primary table

public

person

Secondary table

public

city

The primary table must contain now only 'id' and 'full_name' columns. The field 'city_id' will be added to the table automatically.

Primary table fields

Name	
1	id
2	full_name

Secondary table fields

Name	
1	city

>

Now we have to specify what kind of primary key to be created for the secondary table: surrogate or natural. We create the 'city' table with a surrogate primary key.

☒ Use surrogate primary key

Surrogate key field name

id

☐ Use natural primary key

Secondary table key fields

Primary		Name
1	<input type="checkbox"/>	city

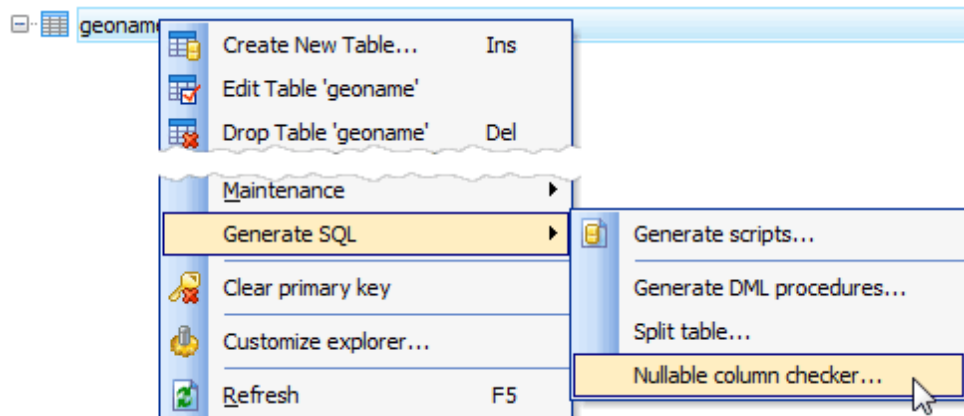
Then click Ready and get the following tables:

	id	full_name	city_id
1	1	John Smith	2
2	2	Mary Doe	1
3	3	Jason Lee	1
4	4	Deisy O'Connor	2

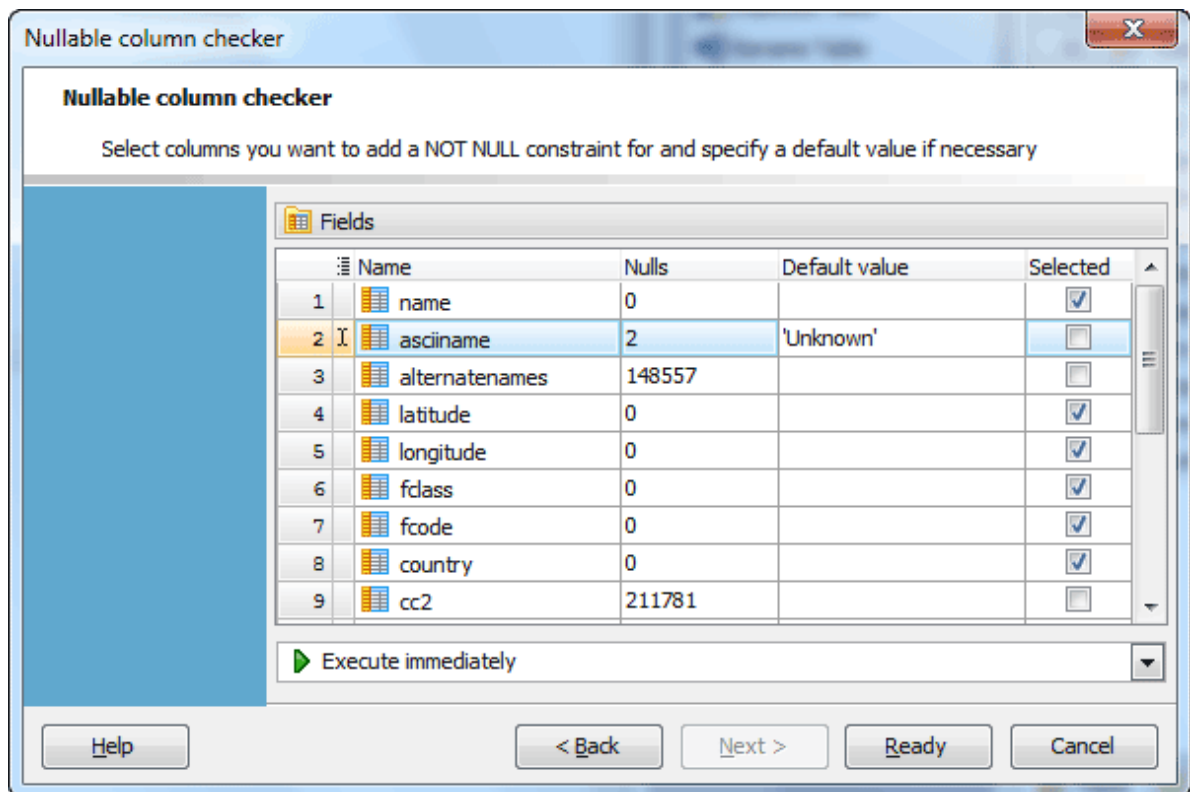
	id	city
1	1	Boston
2	2	New York

9.17 Nullable Column Checker

Nullable Column Checker allows you to refactor your database schema by enforcing NOT NULL constraints to all necessary table columns. It suggests candidates for NOT NULL columns among columns of the selected table and generates SQL script to replace all NULL values of selected columns with specified default values and to add the NOT NULL constraint to these columns. To invoke the wizard, follow the corresponding link of the **Generate SQL** section of popup menu of the selected table at the Explorer tree.



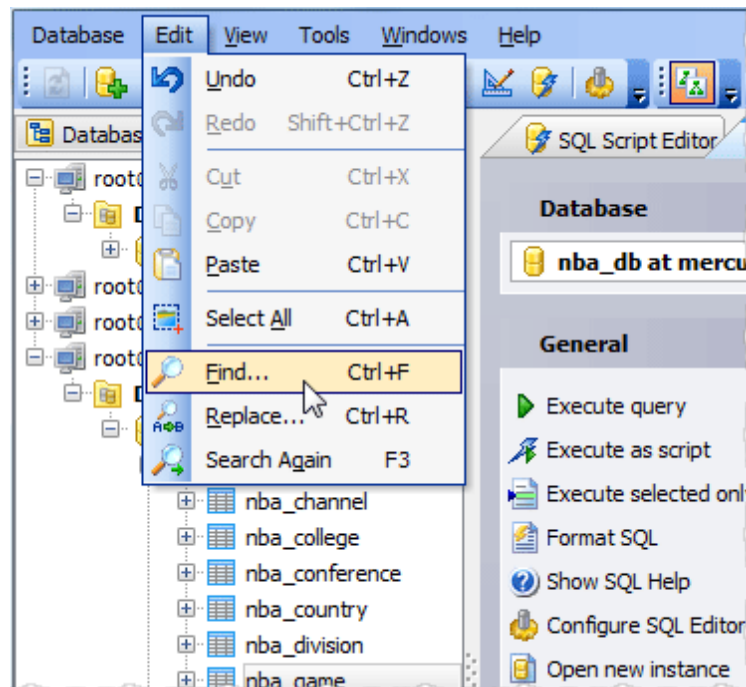
To get SQL scripts enforcing NOT NULL constraints to columns of an existing table, select the necessary columns, specify the default values to be used instead of existing columns NULLs and select the action to perform after the generation. The created scripts can be copied to Clipboard, saved to a file, sent to [SQL_Script_Editor](#)^[266] or executed immediately.



9.18 Dialogs

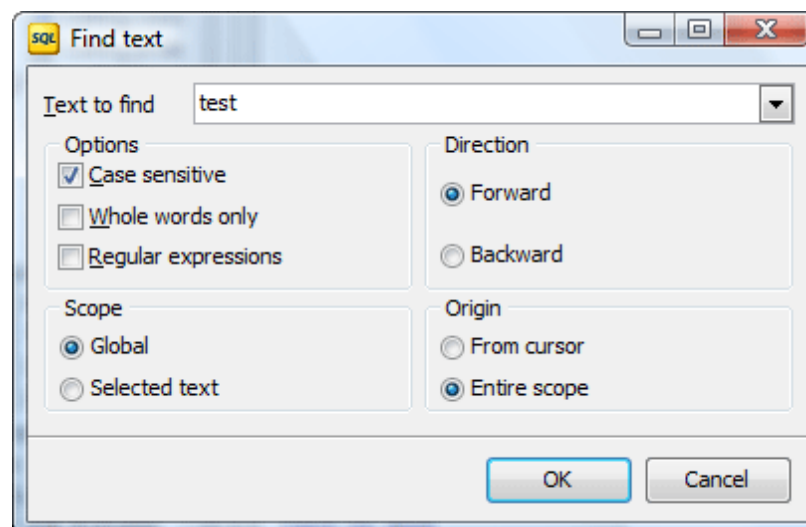
PostgreSQL Maestro provides two dialogs for searching and replacing text in the editor areas of the database tools. Both of them are available through the popup menu of the editor area.

- [Find Text dialog](#) ³¹⁷
- [Replace Text dialog](#) ³¹⁸



9.18.1 Find Text dialog

The Find Text dialog is provided for quick search for certain text.



Text to find

Enter a search string or click the down arrow next to the input box to select from a list of previously entered search strings.

☒ Case sensitive

Differentiates uppercase from lowercase when performing a search.

☒ Whole words only

Searches for words only. (With this option off, the search string might be found within longer words.)

☒ Regular expressions

Recognizes regular expressions in the search string.

Forward

Searches from the current position to the end of the file. [Forward](#) is the default.

Backward

Searches from the current position to the beginning of the file.

Global

Searches the entire file, in the direction specified by the [Direction](#) setting. Global is the default scope.

Selected text

Searches within the selected text only, in the direction specified by the [Direction](#) setting. You can use the mouse or block commands to select a block of text.

From cursor

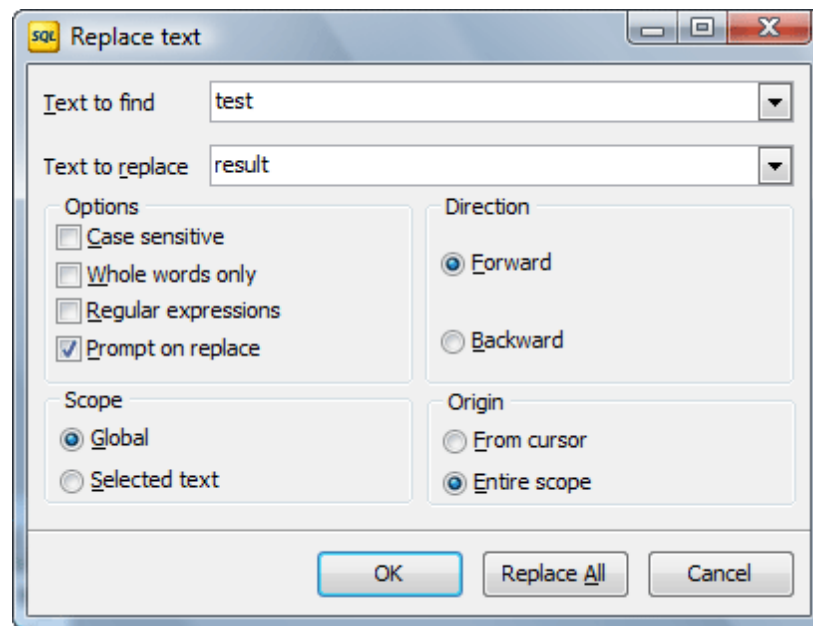
The search starts at the cursor's current position, and then proceeds either forward to the end of the scope, or backward to the beginning of the scope depending on the [Direction](#) setting. [From Cursor](#) is the default setting.

Entire scope

The search covers either the entire block of selected text or the entire file (no matter where the cursor is), depending upon the [Scope](#) options.

9.18.2 Replace Text dialog

The [Replace Text](#) dialog is provided for searching and replacing text in the editor window.



Text to find

Enter a search string. To select from a list of previously entered search strings, click the down arrow next to the input box.

Text to replace

Enter the replacement string. To select from a list of previously entered search strings, click the down arrow next to the input box. To replace the text with nothing, leave this input box blank.

☒ Case sensitive

Differentiates uppercase from lowercase when performing a search.

☒ Whole words only

Searches for words only. (With this option off, the search string might be found within longer words.)

☒ Regular expressions

Recognizes specific regular expressions in the search string.

☒ Prompt on replace

Prompts you before replacing each occurrence of the search string. When Prompt on replace is off, the editor automatically replaces the search string.

Forward

Searches from the current cursor position, to the end of the file. **Forward** is the default Direction setting.

Backward

Searches from the current cursor position, to the beginning of the file.

Global

Searches the entire file, in the direction specified by the Direction setting. **Global** is the

default scope.

[From cursor](#)

The search starts at the cursor's current position, and proceeds either forward to the end of the scope, or backward to the beginning of the scope depending on the Direction setting. [From cursor](#) is the default Origin setting.

[Entire scope](#)

The search covers either the entire block of selected text or the entire file (no matter where the cursor is in the file), depending upon the Scope options.

[Replace All](#)

Click Replace all to replace every occurrence of the search string. If you check [Prompt on replace](#), the [Confirm dialog](#) box appears on each occurrence of the search string.

10 Options

PostgreSQL Maestro allows you to customize the way it works within the [Options](#) dialog. To open the dialog, select the [Tools | Options](#) main menu item.

The window allows you to customize the options grouped by the following sections:

- [Application](#)^[322]
General PostgreSQL Maestro options: environment style, confirmations, window restrictions, explorer tree, [SQL Editor](#), [Visual Query Builder](#), etc.
- [Editors & Viewers](#)^[344]
Customizing of all the SQL editors - [SQL Editor](#), [SQL Script Editor](#), etc.
- [Appearance](#)^[352]
Customizing program interface - bars, trees, menus, etc.

Besides, the [Options](#) dialog allows you to export all program settings to a *.reg file for future use, e.g. on another PC (see [Export Settings](#)^[361] for details).

It is a good idea to check through these settings before you start working with PostgreSQL Maestro. You may be surprised at all the things you can adjust and configure!

10.1 Application

The [Application](#) section allows you to customize common rules of PostgreSQL Maestro behavior. The section consists of several tab; follow the links to find out more about each of them.

- [Preferences](#) ³²²
- [Confirmations](#) ³²³
- [Directories](#) ³²⁵
- [Tools](#) ³²⁵
 - [Explorer](#) ³²⁷
 - [Object Manager](#) ³²⁸
 - [SQL Editor](#) ³²⁸
 - [SQL Script Editor](#) ³²⁹
 - [Query Builder](#) ³³⁰
 - [BLOB Viewer](#) ³³²
 - [Export data](#) ³³³
 - [Database Designer](#) ³³⁴
- [Object Editors](#) ³³⁵
 - [Table](#) ³³⁷
 - [Data Grid](#) ³³⁷
 - [Colors](#) ³⁴⁰
 - [Formats](#) ³⁴¹
 - [Filter](#) ³⁴²

10.1.1 Preferences

User interface area allow you to select your favorite UI style according to your preferences.

☒ [Display splash screen at startup](#)

Displays the splash screen on PostgreSQL Maestro startup.

☒ [Save desktop on disconnect](#)

Saves all the database windows and their positions on disconnecting from the database.

☒ [Disable multiple instances](#)

Prohibits running multiple instances of PostgreSQL Maestro.

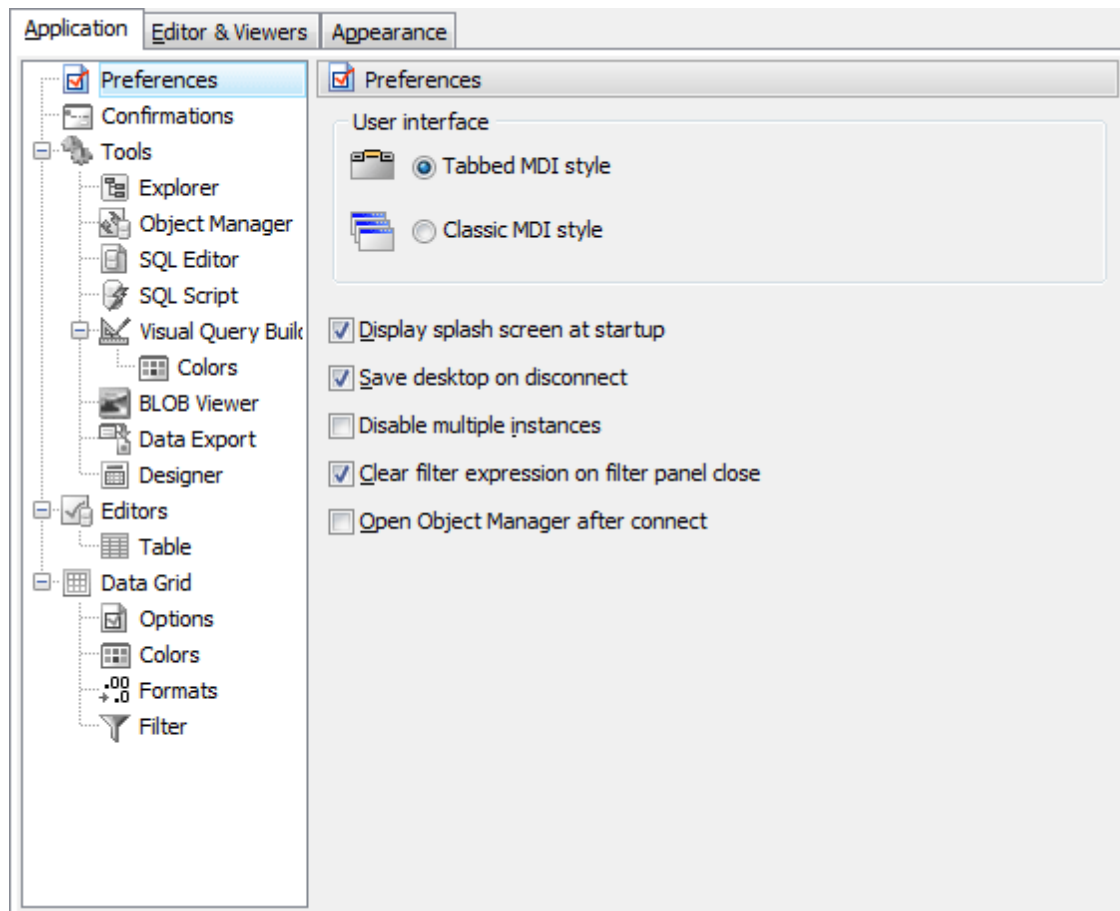
☒ [Open Object Manager after connect](#)

Opens the Object Manager window after connection is established.

☒ [Clear filter expression on filter panel close](#)

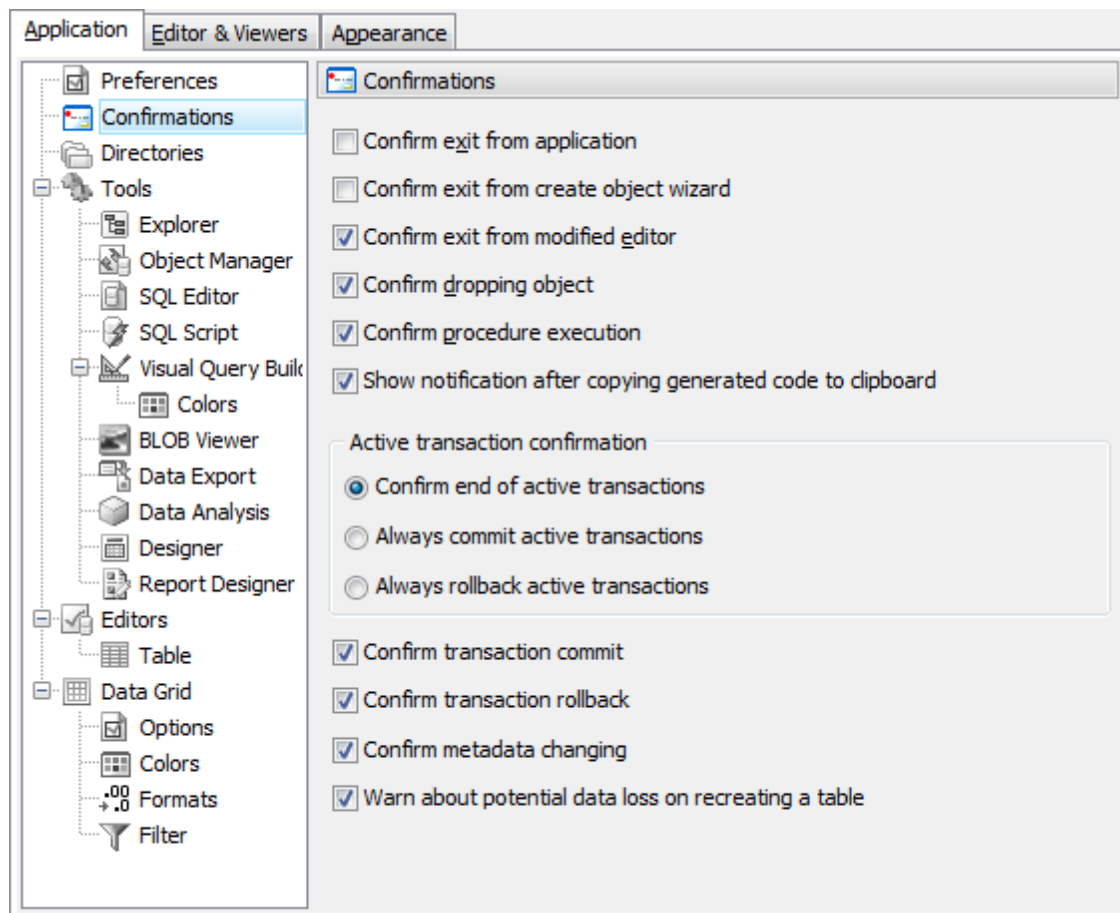
Clears the filter applied to the explorer tree and all the instances of Object Manager

after the filter panel is closed.



10.1.2 Confirmations

Use this tab to manage application confirmations.



☒ **Confirm exit from Create Object Wizard**

If this option is checked, the program requires confirmation each time you want to exit the Create Object Wizard.

☒ **Confirm exit from modified editor**

If this option is checked, the program asks you to confirm exit from the editor, if you have made any changes.

☒ **Confirm dropping object**

If this option is checked, the program requires confirmation for dropping database object.

☒ **Confirm exit from application**

If this option is checked, the program requires confirmation when you want to exit <% PRODUCT_NAME%>.

☒ **Transaction confirmation**

Select whether you will be prompted to commit or rollback active transaction or PostgreSQL Maestro will commit or rollback transactions without asking.

☒ **Confirm metadata changing**

If this option is checked, the program requires confirmation for changing metadata.

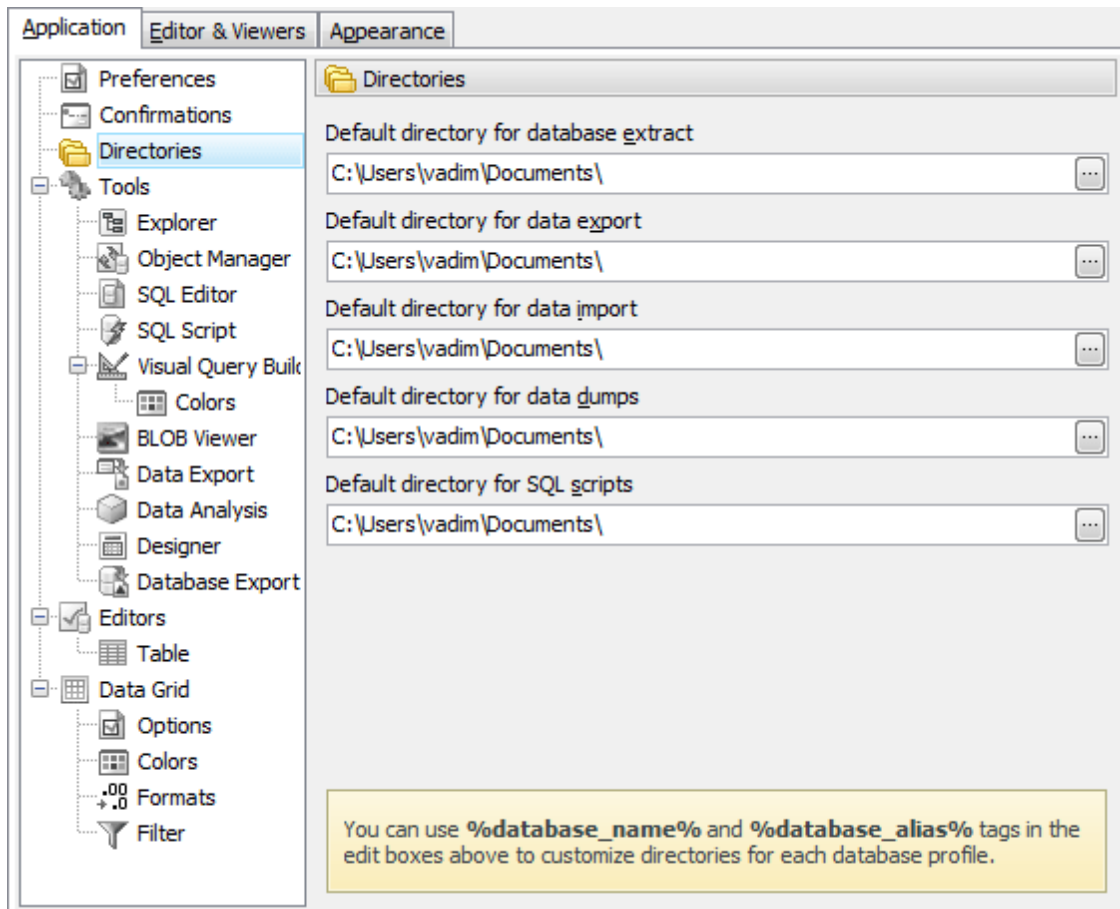
10.1.3 Directories

The tab allows you to specify default directories to be used on database profiles creating. You can use such variables as %database_name%, %database_alias%, and %user_name%.

Example:

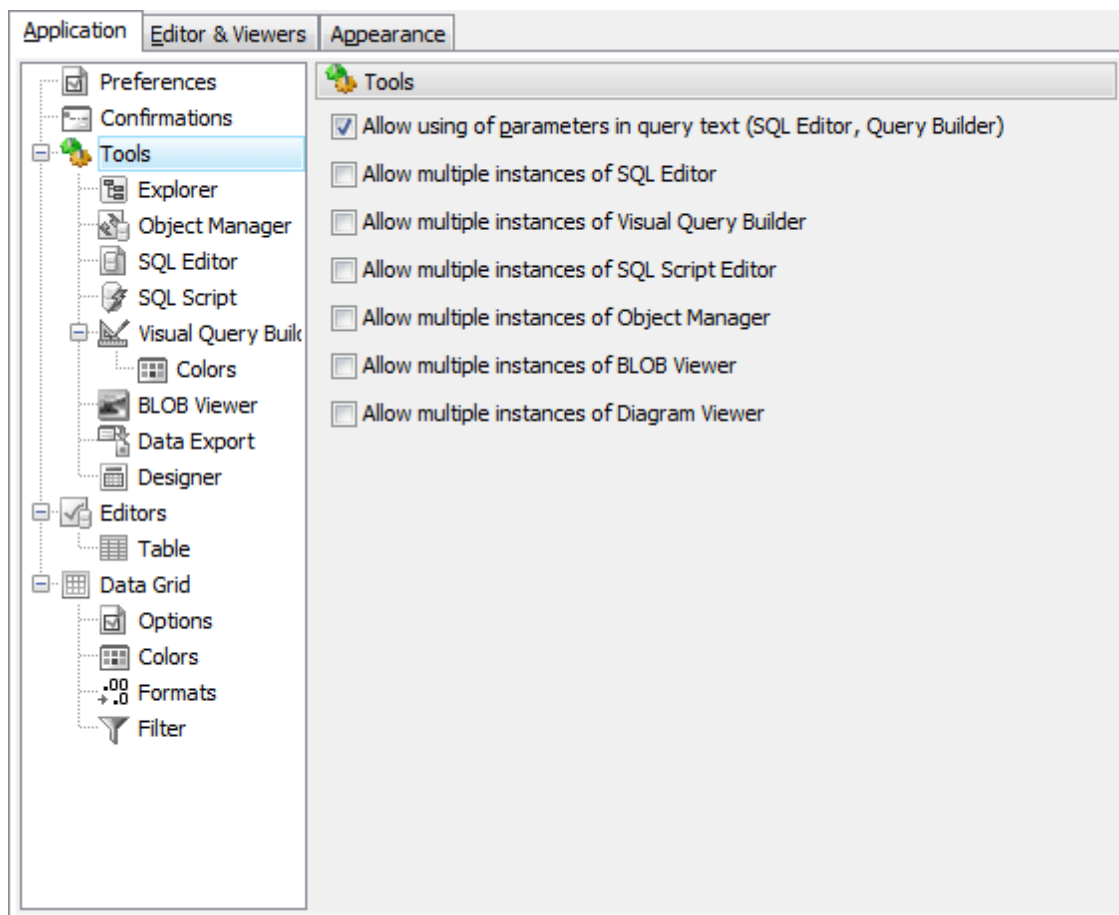
To store all SQL scripts in folders sorted by databases in the "C:\SQL Scripts\" directory, specify the default directory for SQL scripts as follows:

C:\SQL Scripts\%database_name%



10.1.4 Tools

Below you will find a detailed decryption of the following tools options.



☒ Allow using of parameters in query text

Check this option to be able to use query parameters in [SQL Editor](#)^[214] and [Visual Query Builder](#)^[219].

☒ Allow multiple instances of SQL Editor

Check this option to be able to use multiple instances of [SQL Editor](#)^[214] simultaneously.

☒ Allow multiple instances of Visual Query Builder

Check this option to be able to use multiple instances of [Visual Query Builder](#)^[219] simultaneously.

☒ Allow multiple instances of SQL Script Editor

Check this option to be able to use multiple instances of [SQL Script Editor](#)^[266] simultaneously.

☒ Allow multiple instances of Object Manager

Check this option to be able to use multiple instances of Object Manager simultaneously.

☒ Allow multiple instances of BLOB Viewer

Check this option to be able to use multiple instances of [BLOB Viewer](#)^[276] simultaneously.

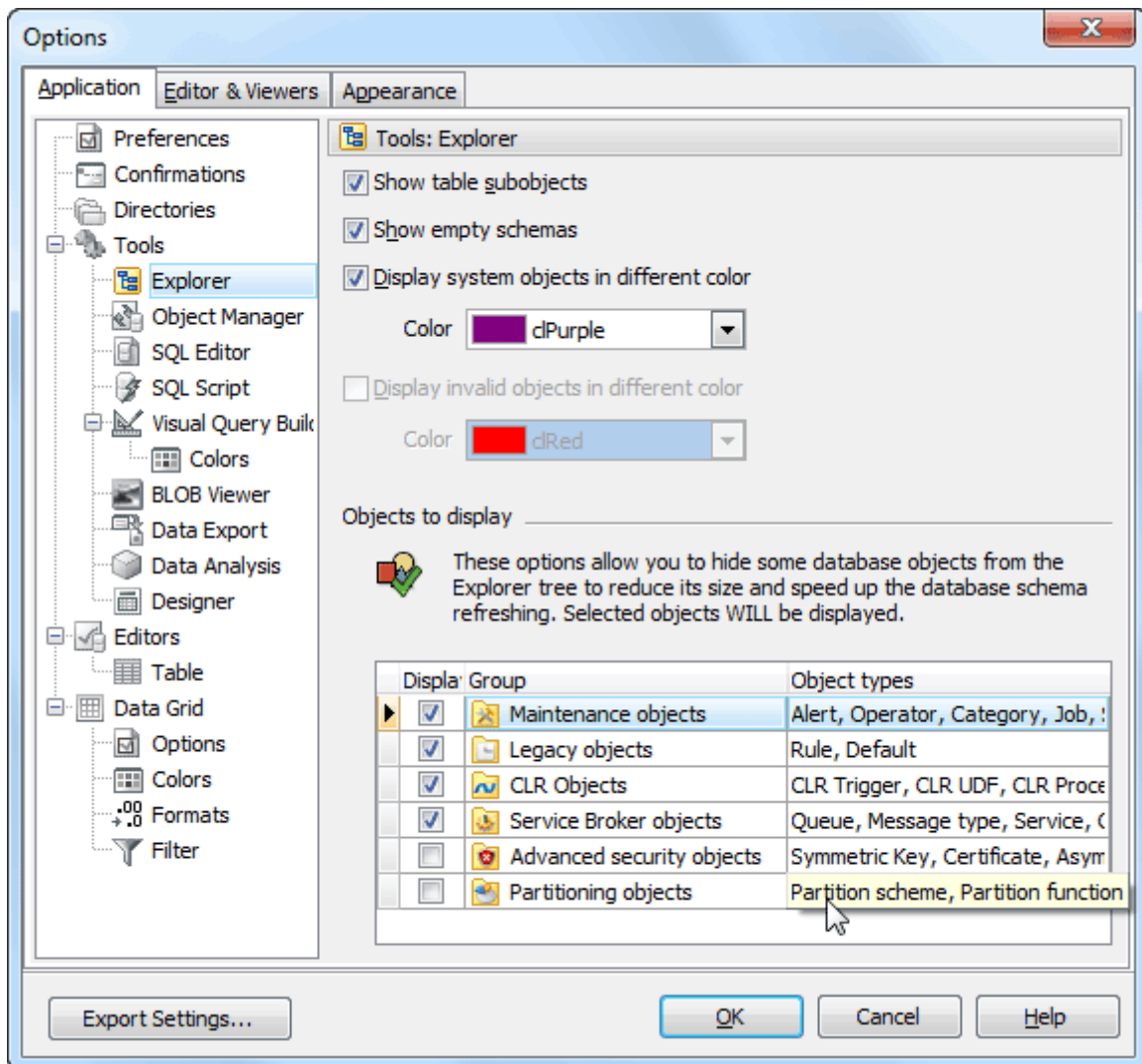
☒ Allow multiple instances of Diagram Viewer

Check this option to be able to use multiple instances of [Diagram Viewer](#)^[282]

simultaneously.

10.1.4.1 Explorer

Below you will find a detailed decryption of the following explorer options.



☒ **Show table subobjects**

Shows/hides table subobjects (fields and indexes) in the explorer tree.

☒ **Sort profiles by aliases**

Sorts profile aliases alphabetically in the explorer tree.

☒ **Expand the "Tables" node after connection**

Shows all database tables in the explorer tree after connecting to the database.

☒ **Expand the "Queries" node after connection**

Shows all database queries in the explorer tree after connecting to the database.

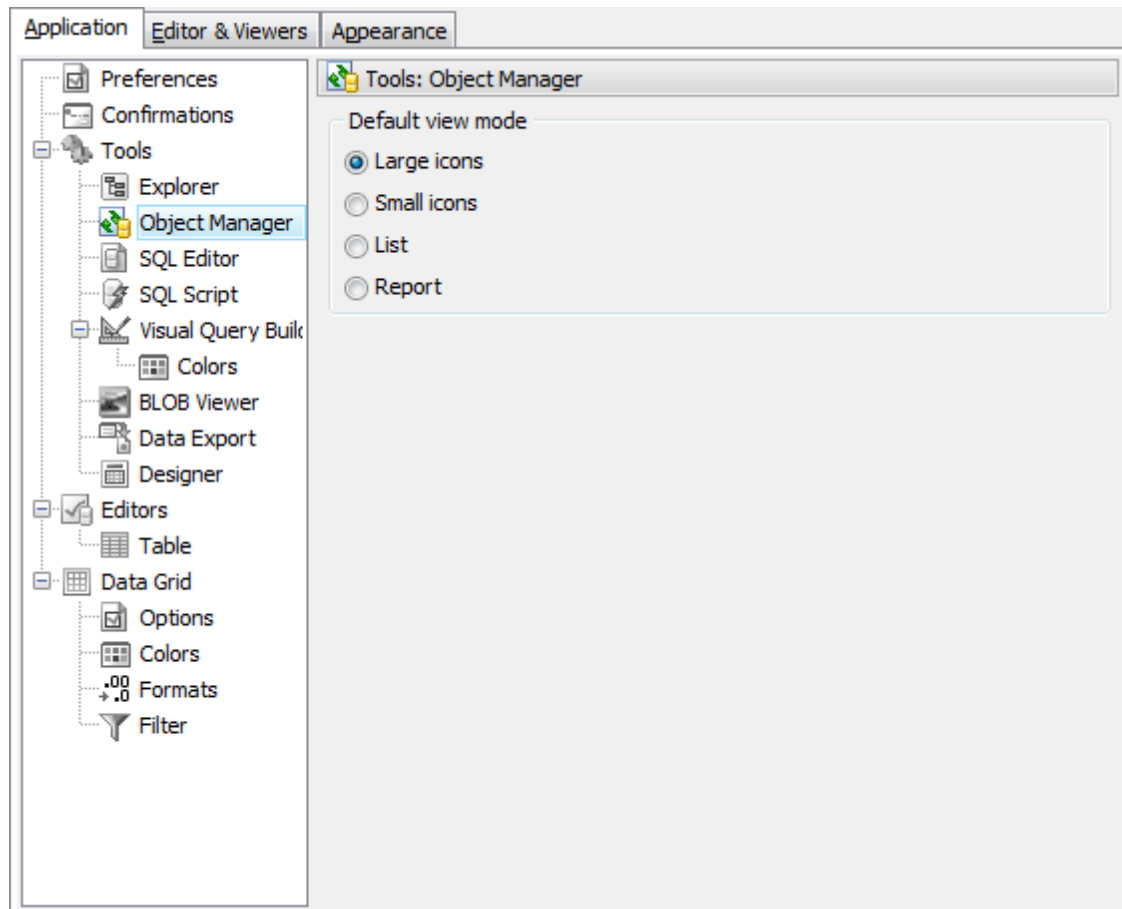
☒ **Display system objects in different color**

Represents all system objects in selected color.

You can also exclude/include rarely used objects from/to the Explorer tree. Manage object groups to be displayed at Explorer with corresponding checkboxes.

10.1.4.2 Object Manager

Below you will find a detailed decryption of the following [Object Manager](#) options.

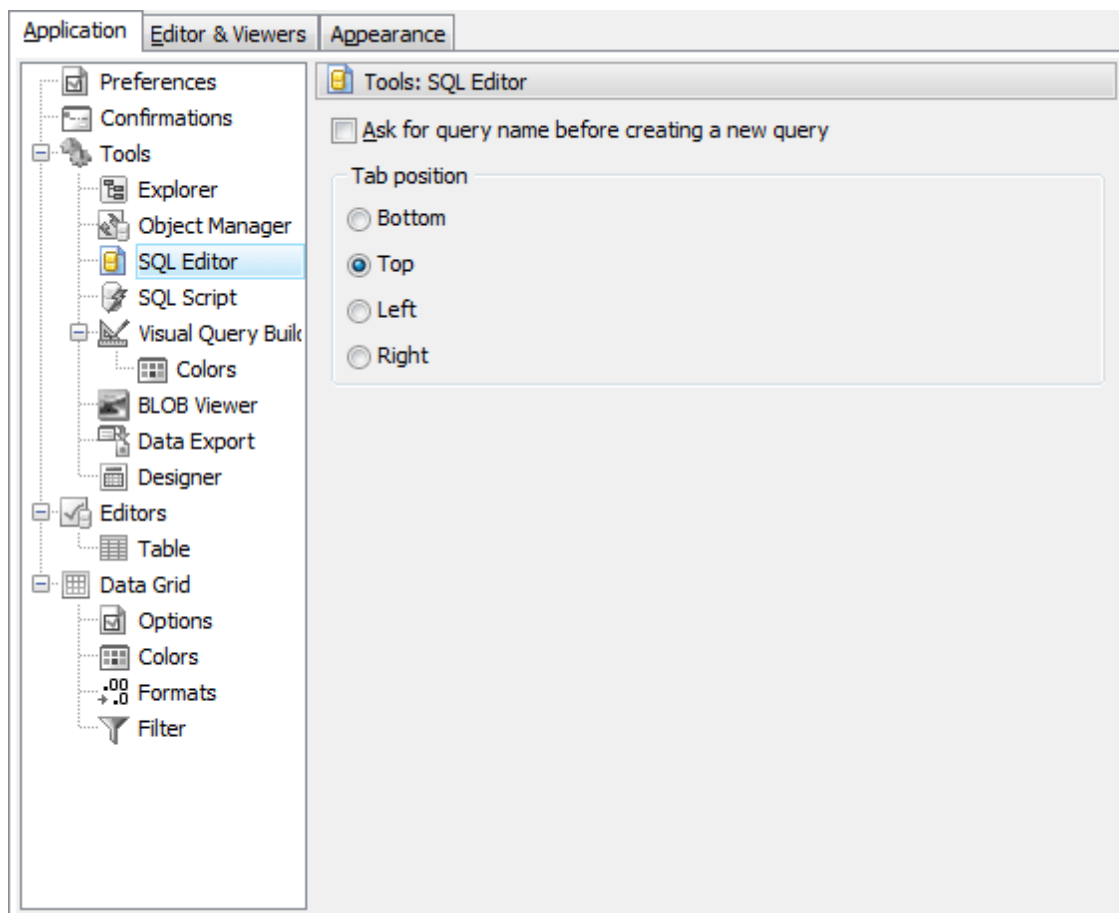


Default view mode

Defines which view mode (*large icons*, *small icons*, *list* or *report*) is applied to Object Manager by default.

10.1.4.3 SQL Editor

Below you will find a detailed decryption of the following [SQL Editor](#) options.



☒ **Ask for query name before creating a new query**

If this option is checked, [SQL Editor](#)²¹⁴ asks for a query name each time you create a new query.

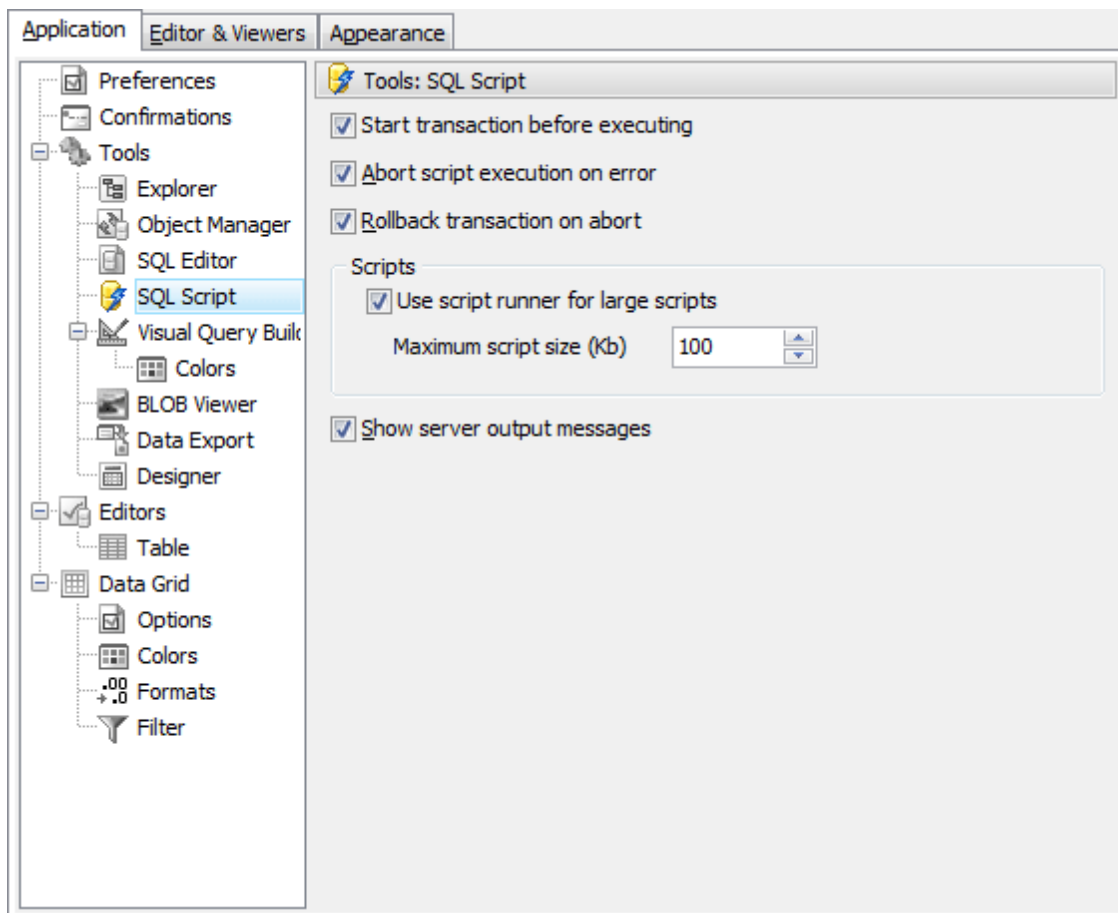
☒ **Auto commit**

Check the option to execute queries in autocommit mode (default value) or leave it blank to manage transactions manually.

You can also select [position](#) of query tabs.

10.1.4.4 SQL Script Editor

Below you will find a detailed decryption of the following [SQL Script Editor](#) options.



☒ **Abort script execution on error**

If this option is checked, script execution aborts when an error occurs.

☒ **Rollback transaction on abort**

This option evokes automatic rollback on script execution abort.

☒ **Use script runner for large scripts**

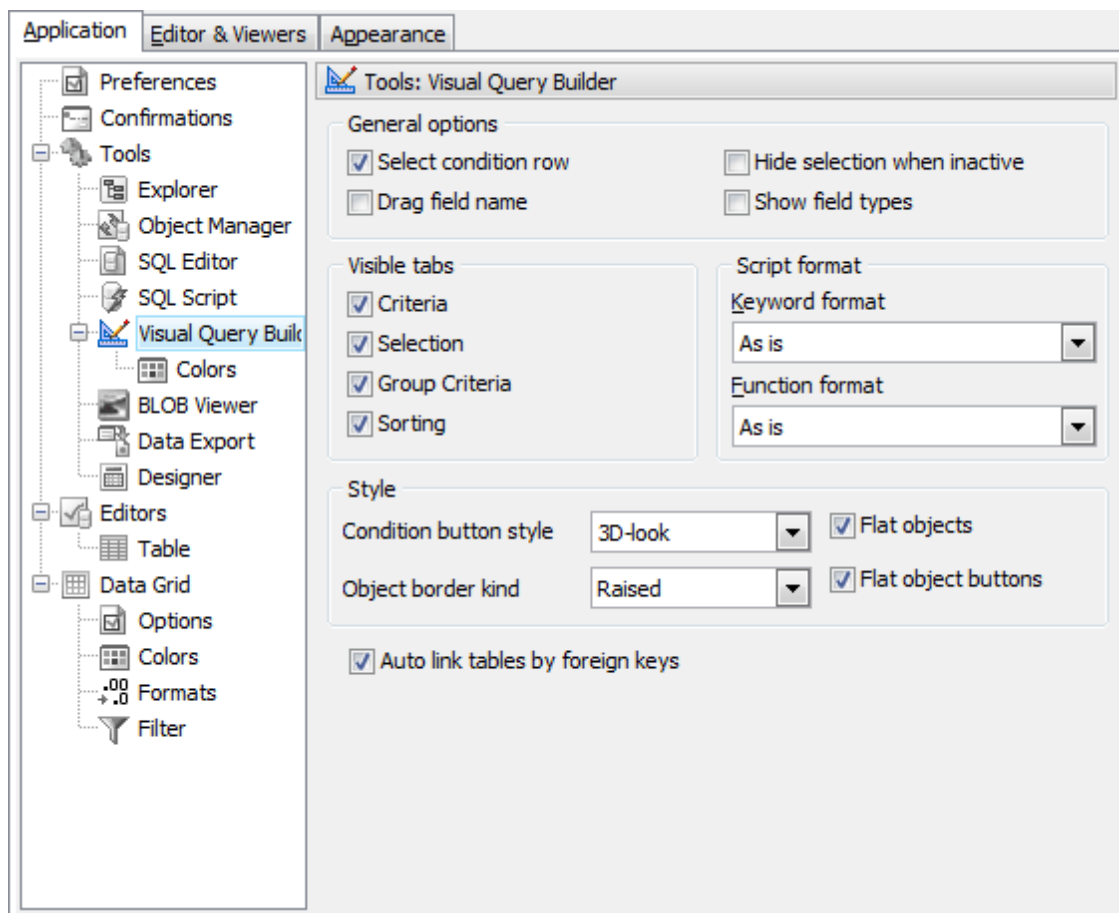
Check the box to execute large script in the fastest way. You can change the maximum size of a script to execute without script runner.

☒ **Show server output messages**

Turn the option ON to see warning messages generated by the server.

10.1.4.5 Query Builder

Below you will find a detailed decryption of the following [Query Builder](#) options.



☒ **Select condition row**

Displays the selected condition in different row on the **Criteria** and **Grouping Criteria** tabs of [Visual Query Builder](#)^[219].

☒ **Drag field name**

Displays the dragged field name in the **Builder** area.

☒ **Hide selection when inactive**

Hides the selection when the query builder is inactive.

☒ **Show field types**

Displays the field type next to the field in the table box.

Visible tabs

These options specify which the query builder tabs are available and which are not. Check them to make the appropriate tabs visible.

Script format

These options specify the case formatting of keywords and functions in query text on the **Edit** tab. **As is** saves the original case, **Uppercase** sets all the keywords/functions to upper case, **Lowercase** sets all the keywords/functions to lower case, and **First upper** sets the first letters of all keywords/functions to upper case.

Style

These options specify how different the [Query Builder](#) objects look like - 3D, flat, etc.

☒ Auto link tables by foreign keys

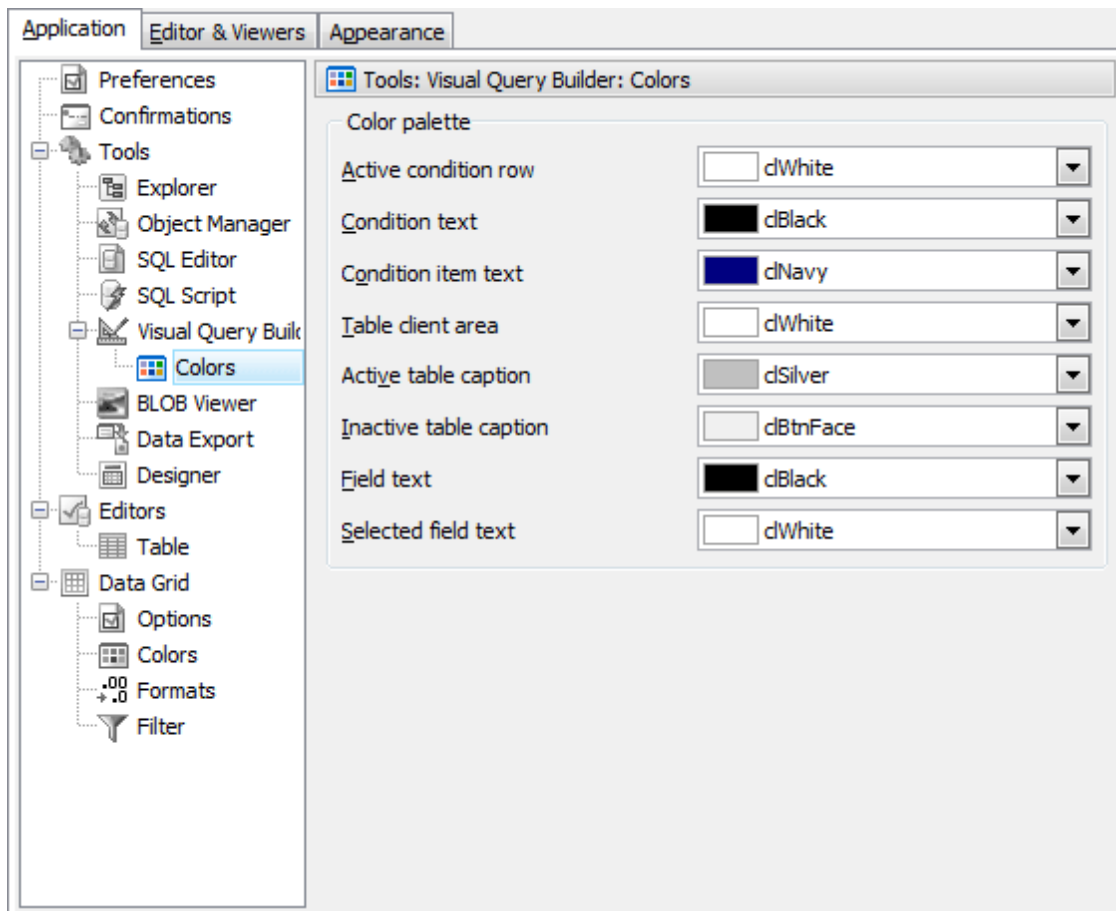
If tables that have foreign key reference are placed in [Query Builder](#), in diagram they will be auto linked.END

Colors

These options define colors of the different [Query Builder](#) elements: condition row, active caption, table client area, etc. Click an item to select a color for the appropriate Query Builder element.

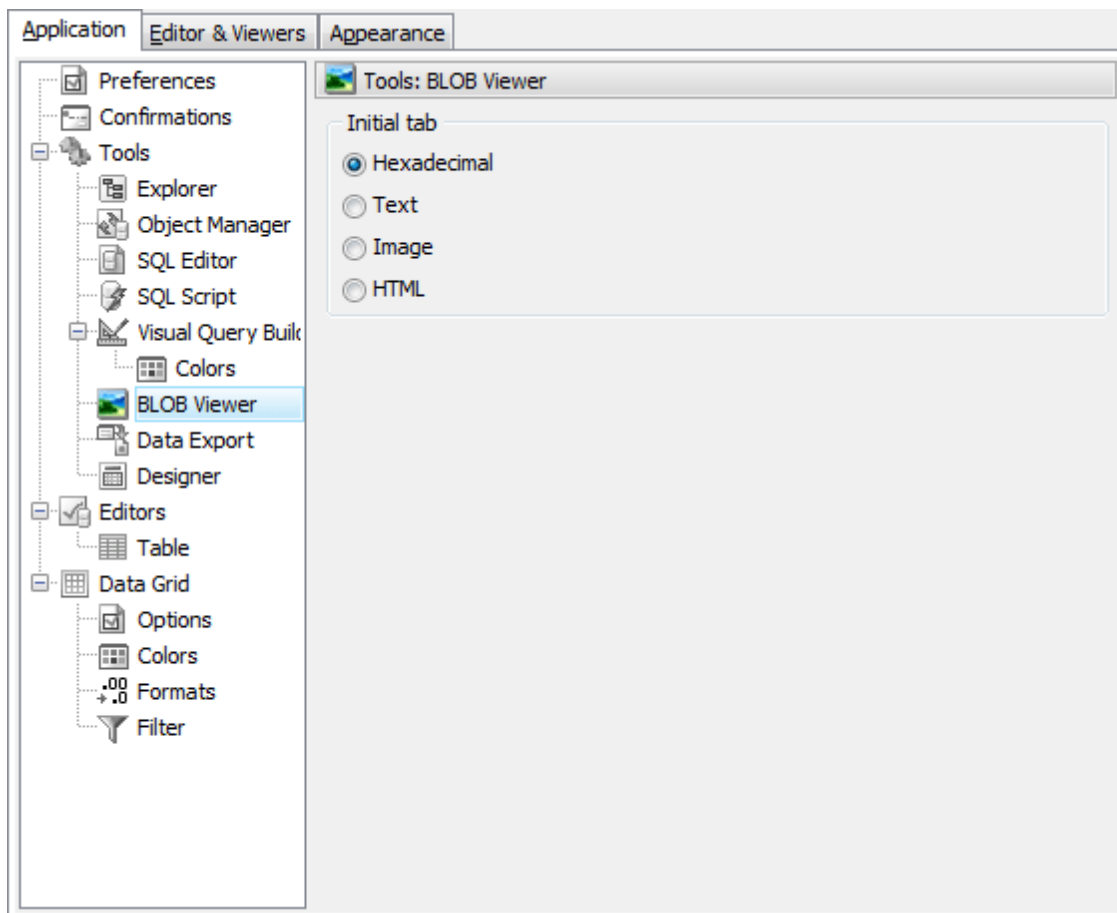
10.1.4.5.1 Colors

The tab is provided to editing of the Query Builder color schema. Customize colors for all editor element according to your preferences.



10.1.4.6 BLOB Viewer

Below you will find a detailed decryption of the following [BLOB Viewer](#)²⁷⁶ options.



Initial tab

Specifies which tab of **BLOB Viewer** should be active when it is initially opened.

10.1.4.7 Data Export

This window allows you to customize formats applied to exported data. Edit the format masks to adjust the result format in the way you need.

In *numeric* formats using digit placeholder (`#` or `0`) you can specify the format of number. For example, integer 1234567890 with `# # # # # 0` integer format is represented like 1 234 567 890. The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

Conversion and their description for *date*, *time* and *date time* format:

dd	day of the month, represented by 1 or 2 symbols. For example, the first day of month is 1
DD	day of the month, represented only by 2 symbols. For example, the first day of month is 01
mm	minutes

MM	month
yy	year, represented by 2 symbols. For example, 2006 year will be 06
yyyy	year, represented by 4 symbols. For example, 2006
h	hour, represented by 1 or 2 symbols. For example, 2
hh	hour, represented only by 2 symbols. For example, 02

".", ",", ":" symbols can be placed as separators in format masks for *integer*, *float*, *date*, *time*, *date time*, *currency* types.

☒ **Auto save format strings**

Auto saves format strings when they are changed (e.g. at the [Adjusting data formats](#) step in [Export Data Wizard](#)^[245]).

Using [Set defaults](#) button, you can set default values of data formats.

10.1.4.8 Database Designer

The following topic provides a shot [Database Designer](#)^[298] retrofitting's description.

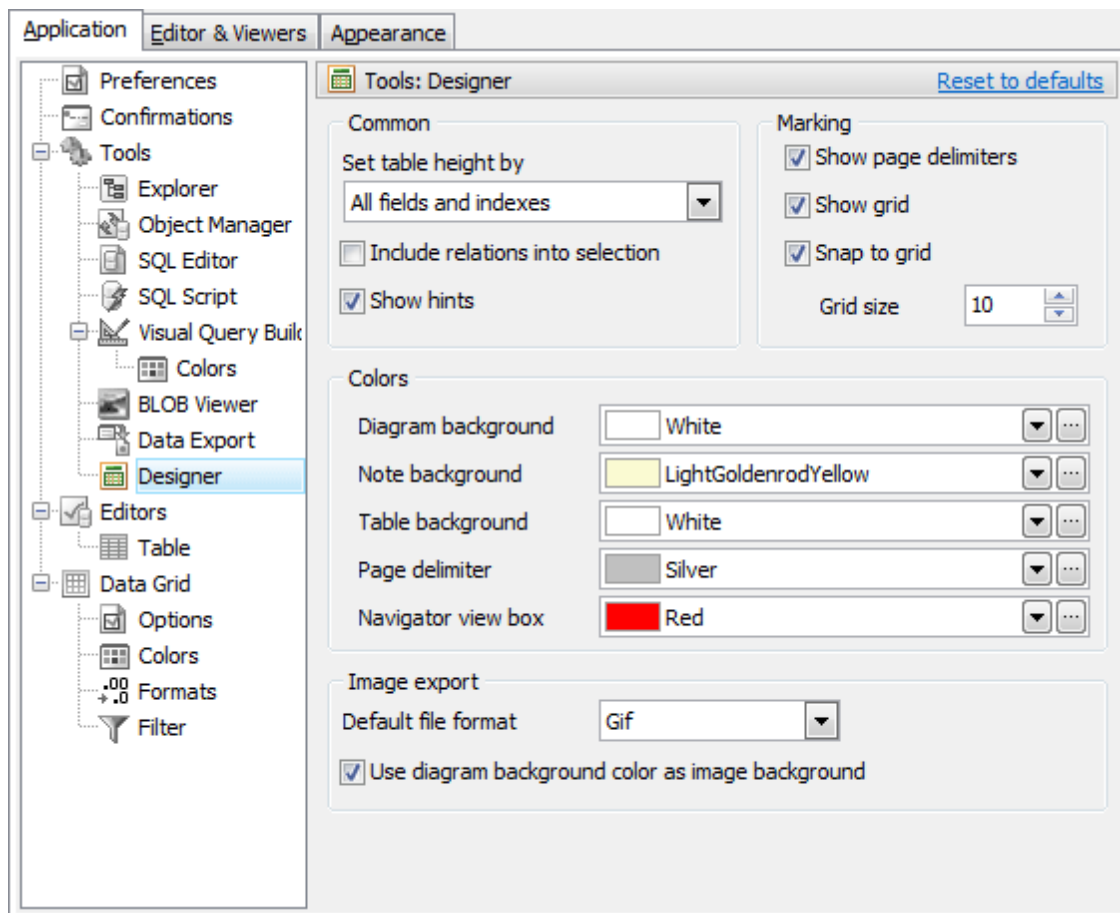
The [Set table height](#) option determines what the table attributes will be represented at the diagram ([Caption only](#), [Key fields](#), [All fields](#) or [All fields and indexes](#)).

Frequently it is necessary to line up some tables and objects on the diagram in a column. One of the ways that you may find convenient to do this is to take advantage of the grid. You can enable showing of the grid by checking the [Show grid](#) clause at the [Marking](#) options. Then diagram will be covered by points disposed on the same interval between.

If you enable snap to grid function by using the [Snap to grid](#) checkbox, when you move the table or other object, its upper left corner "snaps" to the nearest grid point. This feature will assist you to line up objects both horizontally and vertically. Diagram objects moved on a diagram can be automatically snapped to the grid points even if the grid is not displayed on the diagram.

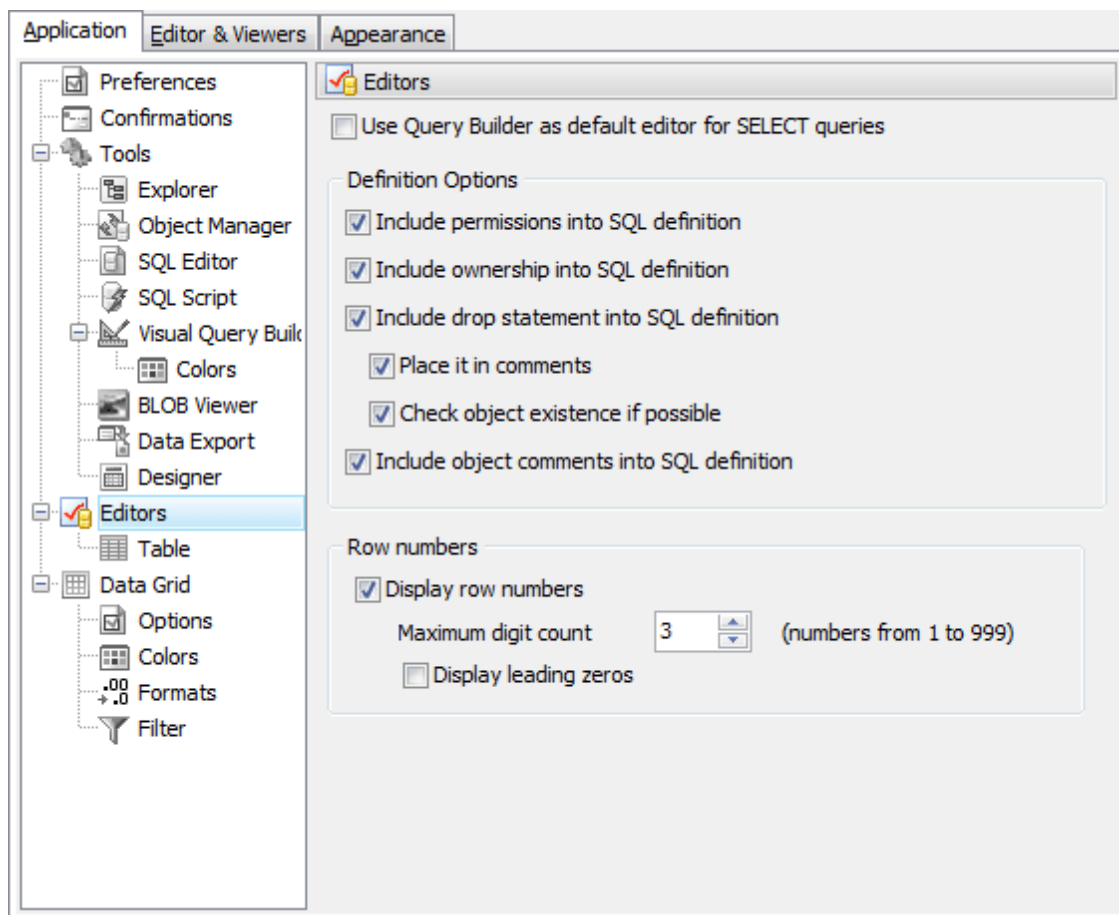
Interval between grid points can be changed as [grid size](#).

The [Colors](#) options are to attune the Designer appearance to your taste. Here you change diagrams and tables background colors, the colors of page delimiter and Navigator view box.



10.1.5 Object Editors

Below you will find a detailed description of the following object editors options.



☒ **Open each object editor in a new window**

With this option checked a new child window opens each time you open an object for editing, otherwise the edited object is being changed in the existing object editor (except the existing object editor is in modified state).

☒ **Use Query Builder as default editor for SELECT queries**

With this option enabled all the SELECT queries will be opened in [Visual Query Builder](#)^[219] instead of [SQL Editor](#)^[214].

☒ **Include permissions**

If checked, the SQL definition includes all the GRANT statements, which are applied to the object. Let's assume that a user named 'john' has rights to read the data from the table 'customers' from the schema 'public', and a user named 'michael' can delete data from that table. In this case the script includes itself the following forms:

```
GRANT SELECT ON public.cutomers to john;
```

```
GRANT DELETE ON public.cutomers to michael;
```

Note: Please take a look at the topic Grant (PostgreSQL Reference) to learn more about security management in PostgreSQL server.

☒ **Include ownership**

If checked, the SQL definition includes the object owner specification. For example, if a user named 'john' is the owner of the table 'customers' from the schema 'public', the script contains the following statement:

```
ALTER TABLE public.customers OWNER TO john;
```

☒ [Include drop statement](#)

If checked, the SQL definition includes the drop statement.

☒ [Place it in comments](#)

With this option drop statement will be placed in comments of the SQL definition.

☒ [Include object comments into SQL definition](#)

With this option enabled comments that are specified for the object and object subitems are placed in SQL definition.

☒ [Show dependencies details](#)

With this option enabled the Dependencies part of the object editors will represent the dependencies details.

☒ [Group types by schemas](#)

It allows you to group types in different ways.

[Row numbers](#)


This options group allows you to manage the row numbering of the subobjects lists such as fields, indexes, parameters and so on.

To enable/disable the numbering, use [Display row numbers](#) checkbox. You can set the number columns width with [Maximum digit count](#). (I.e. for the value '3' the max column number will be 999).

For uniformity you can use the [Display leading zeros](#) option. With this option enabled and maximum digit count '3' your numbering column will be of the form: '001, 002, 003, ...'.

10.1.5.1 Table

[Initial tab](#)

Specifies which tab of [Table Editor](#)  should be active when it is initially opened.

☒ [Retrieve record count before loading data in the data grid](#)

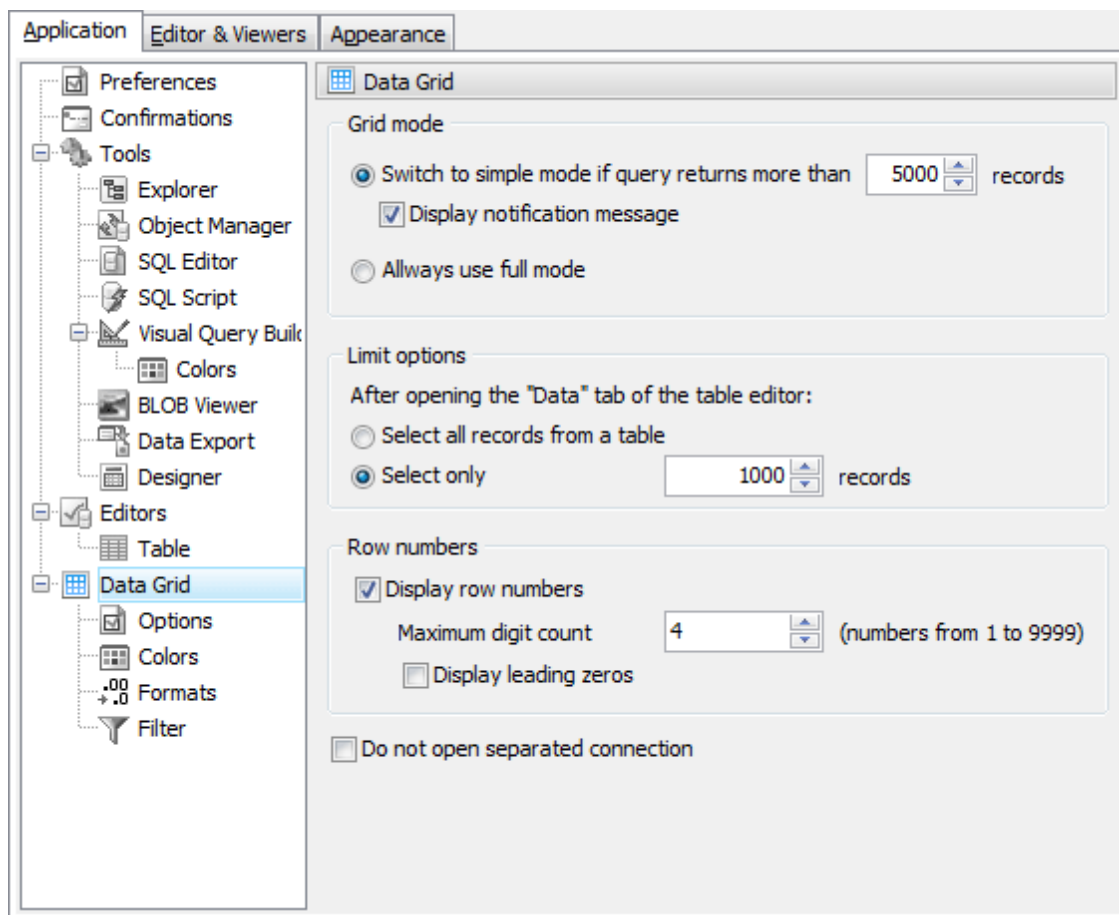
With this option enabled all the SELECT COUNT(*)... query is executed before loading data in the data grid.

[Default field type](#)

Specifies the field type appearing in [Field Editor](#)  by default.

10.1.6 Data Grid

Below you will find a detailed description of the following data grid options.



PostgreSQL Maestro provides you with [two grid modes](#) of viewing data:

- Full grid mode allows you to group, filter and sort data in a usual way.
- Simple mode is provided for working with large records number. For data fetching speed-up, filtering, sorting, and grouping features are not enabled in this mode.

You can use [notification message](#) to indicate simple mode.

Set the number of records to switch to simple mode automatically or select [Always use full mode](#).

Limit options

Allows you either to select all records from table after opening the Data tab, or select only specified number of rows on one page with an ability to rotate pages and view all data.

Row numbers

This options group allows you to manage grid rows numbering.

To enable/disable the numbering, use [Display row numbers](#) checkbox. You can set the number columns width with [Maximum digit count](#). (I.e. for the value '3' the max column number will be 999).

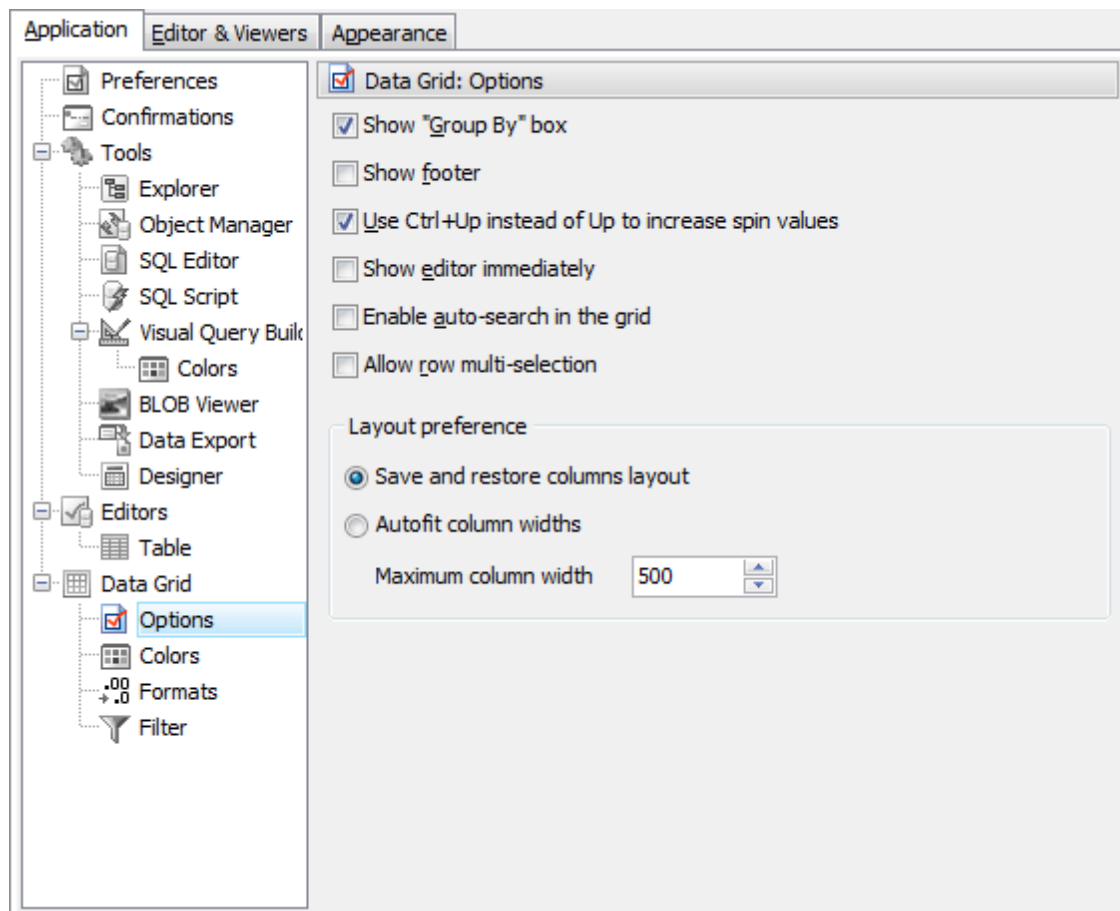
For uniformity you can use the [Display leading zeros](#) option. With this option enabled and maximum digit count '3' your numbering column will be of the form: '001, 002, 003, ...'.

☒ [Do not open separated connection](#)

With this option enabled a new connections for fetching data is not opened. This gives you an ability to work with data a little bit faster, because time for opening a new connections is not demanded.

10.1.6.1 Options

Below you will find a detailed decryption of the data grid options.



☒ [Show "Group By" box](#)

Shows the box on the top of the grid view for grouping data by fields.

☒ [Show footer](#)

Shows the footer on the bottom of the grid view.

☒ [Use Ctrl+Up instead of Up to increase spin values](#)

Allows you to use Ctrl+Up and Ctrl+Down key combinations for editing the spin for numeric fields.

☒ [Show editor immediately](#)

Allows editing the cell value right after the cell is clicked.

☒ [Enable auto-search in the grid](#)

Allows you to search records in the grid by the first letters.

☒ [Allow row multi-selection](#)

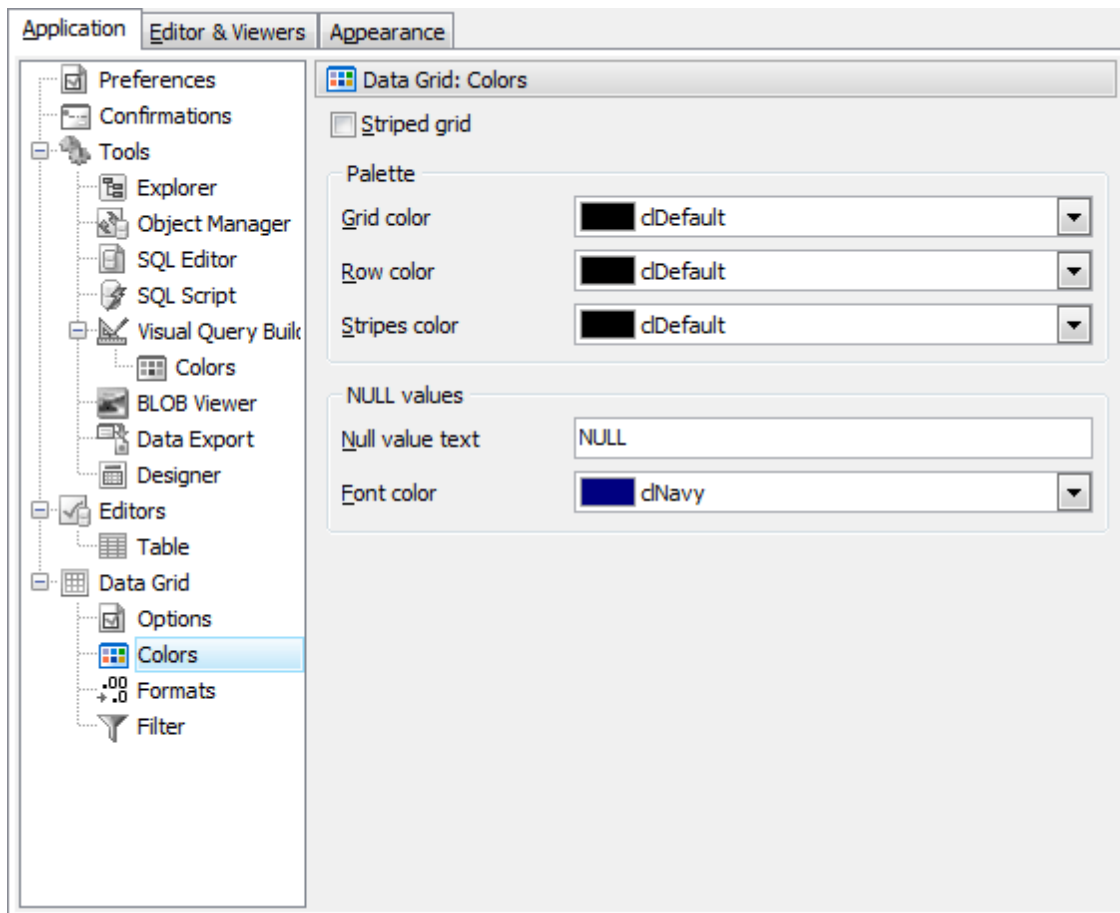
Allows you to select multiple records using the Ctrl and Shift keys.

[Layout preference](#)

Select whether PostgreSQL Maestro should remember the column positions for the grids or fit them automatically.

10.1.6.2 Colors

Below you will find a detailed decryption of the following colors options.



☒ [Striped grid](#)

Displays the odd grid rows in a different color defined by the [Stripes color](#) option.

[Grid color](#)

Defines the background color of the data grid.

[Row color](#)

Defines the color of the selected row in the data grid.

[Stripes color](#)

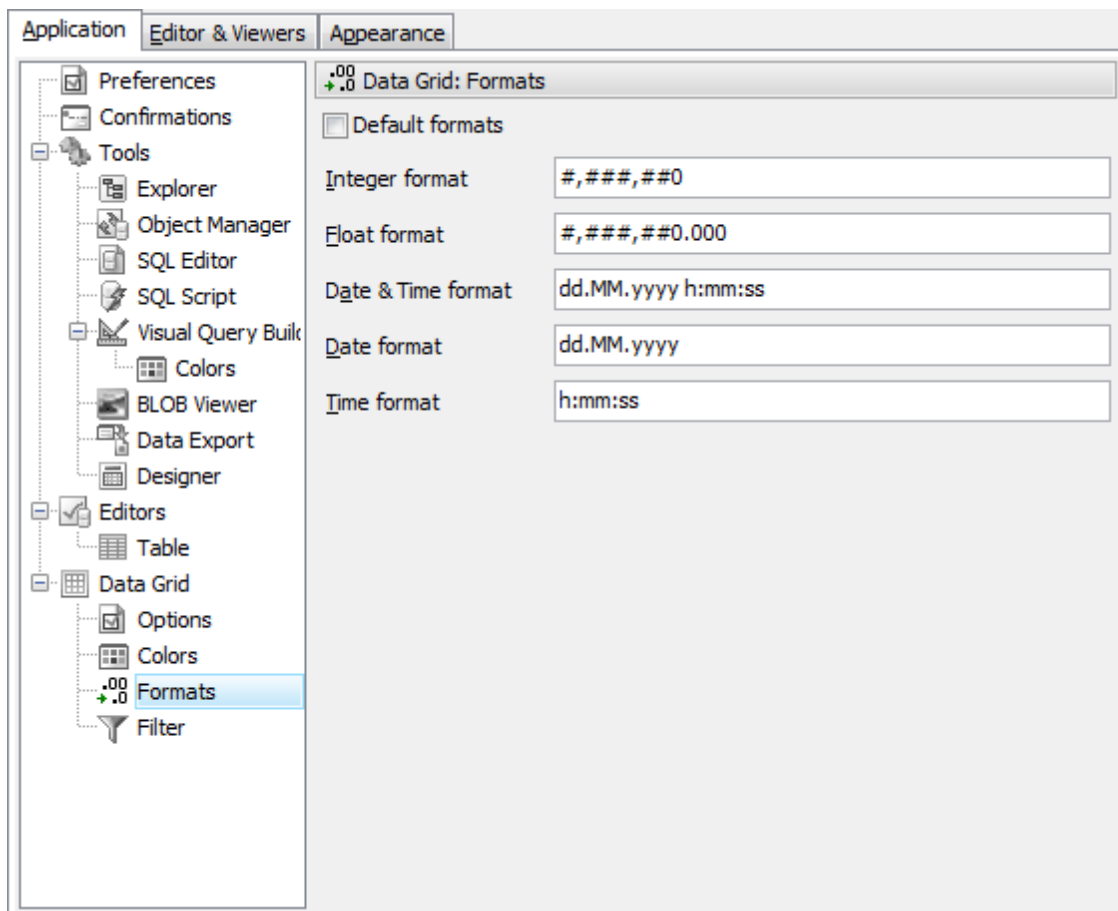
Defines the color of the odd rows if the [Striped Grid](#) option is on.

Null values

Use [Null value text](#) to define the text that stand for the NULL value and use [Font color](#) to set the color for displaying this text.

10.1.6.3 Formats

Below you will find a detailed decryption of the following formats options.



This window allows you to customize formats applied to data in grid. Edit the format masks to adjust the result format in the way you need.

In *numeric* formats using digit placeholder (# or 0) you can specify the format of number. For example, integer 1234567890 with # # # # # # 0 integer format is represented like 1 234 567 890. The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

Conversion and their description for *date*, *time* and *date time* format:

dd	day of the month, represented by 1 or 2 symbols. For example, the first day of month is 1
DD	day of the month, represented only by 2 symbols. For example, the first

",";": symbols can be placed as separators in format masks for *integer*, *float*, *date*, *time*, *date time*, *currency* types.

These options allows you to customize data filtering in grids.



When checked the filter row is always represented in the data grid as an additional row.

[Apply filter row changes](#) and [Apply column popup filter changes](#) allows you to manage the speed of the data filtering. To speed-up the process select [Immediately](#) as the time the filter you are set will be applied.

Change the [Position of filter panel](#) and customize timestamp data filtering: check the [Use relative dates in filters](#) box to include in column popup filter such options as "Yesterday", "Today", "Tomorrow", "Last 30 day", "Last week", "Next week", and others; check the [Ignore time part](#) box to neglect time part of timestamp data under the filtering.

By default filter buttons are shown at all columns header, to [show filter button only in selected column](#), check the corresponding option.

You can specify the case sensitivity of the grid filter with the [Case insensitive](#) checkbox (ON by default).

10.2 Editors & Viewers

The [Editors & Viewers](#) section allows you to set the parameters of viewing and editing the SQL statements within PostgreSQL Maestro.

- [General](#) ^[344]
- [Display](#) ^[345]
- [SQL highlight](#) ^[346]
- [PHP highlight](#) ^[348]
- [XML highlight](#) ^[347]
- [Code Insight](#) ^[349]
- [Code Folding](#) ^[350]

See also: [SQL Editor](#) ^[214], [SQL Script Editor](#) ^[266], [Visual Query Builder](#) ^[219], [Table Editor](#) ^[77].

10.2.1 General

If the [Auto indent](#) option is checked, each new indentation is the same as the previous when editing SQL text.

☒ [Insert mode](#)

If this option is checked, insert symbols mode is default on.

☒ [Use syntax highlight](#)

Enables syntax highlight in the object editor window.

☒ [Always show links](#)

If this option is checked, hyperlinks are displayed in the editor window. To open a link click it with the **Ctrl** button pressed.

☒ [Show line numbers](#)

If this option is checked, line numbers are displayed in the editor window.

☒ [Show special chars](#)

If this option is checked, special chars (like line breaks) are displayed in the editor window.

☒ [Use smart tabs](#)

With this option on the number of tab stops is calculated automatically, depending on the previous line tab.

☒ [Convert tabs to spaces](#)

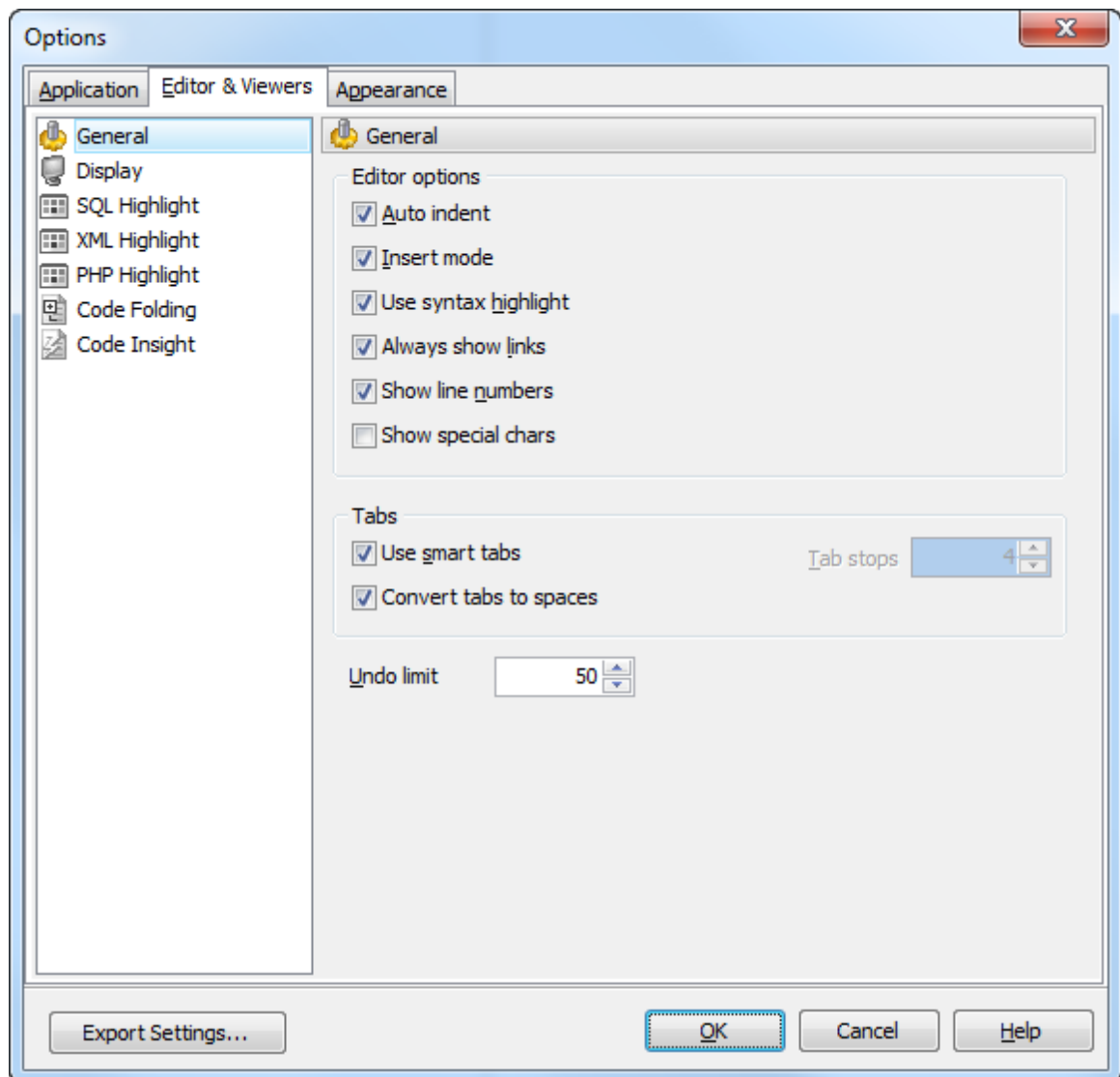
If this option is checked, each time you press the Tab key, the appropriate number of spaces will be added to the edited text.

[Tab Stops](#)

Defines the tab length, used when editing text.

Undo Limit

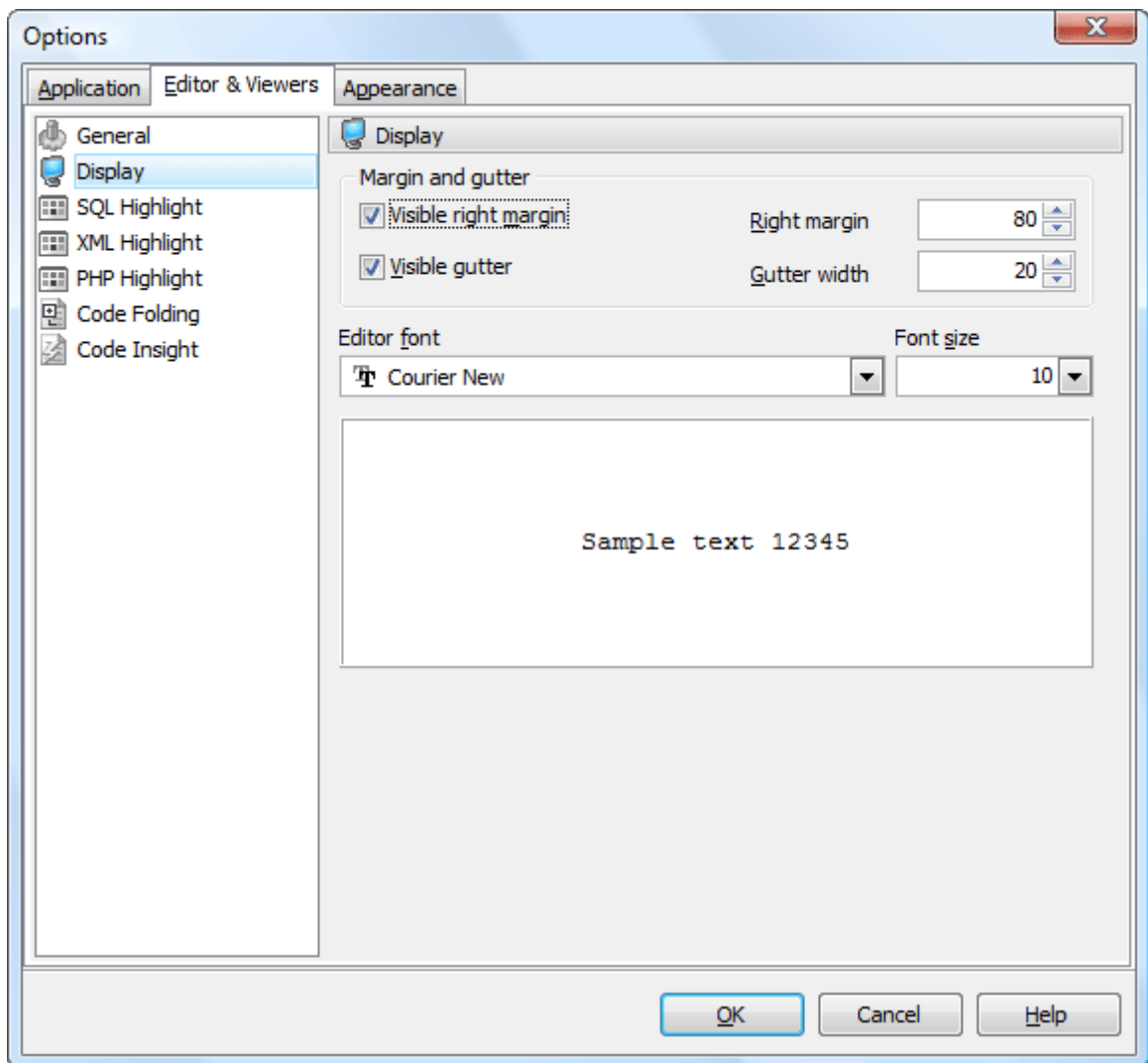
Defines the maximum number of changes possible to be undone.



10.2.2 Display

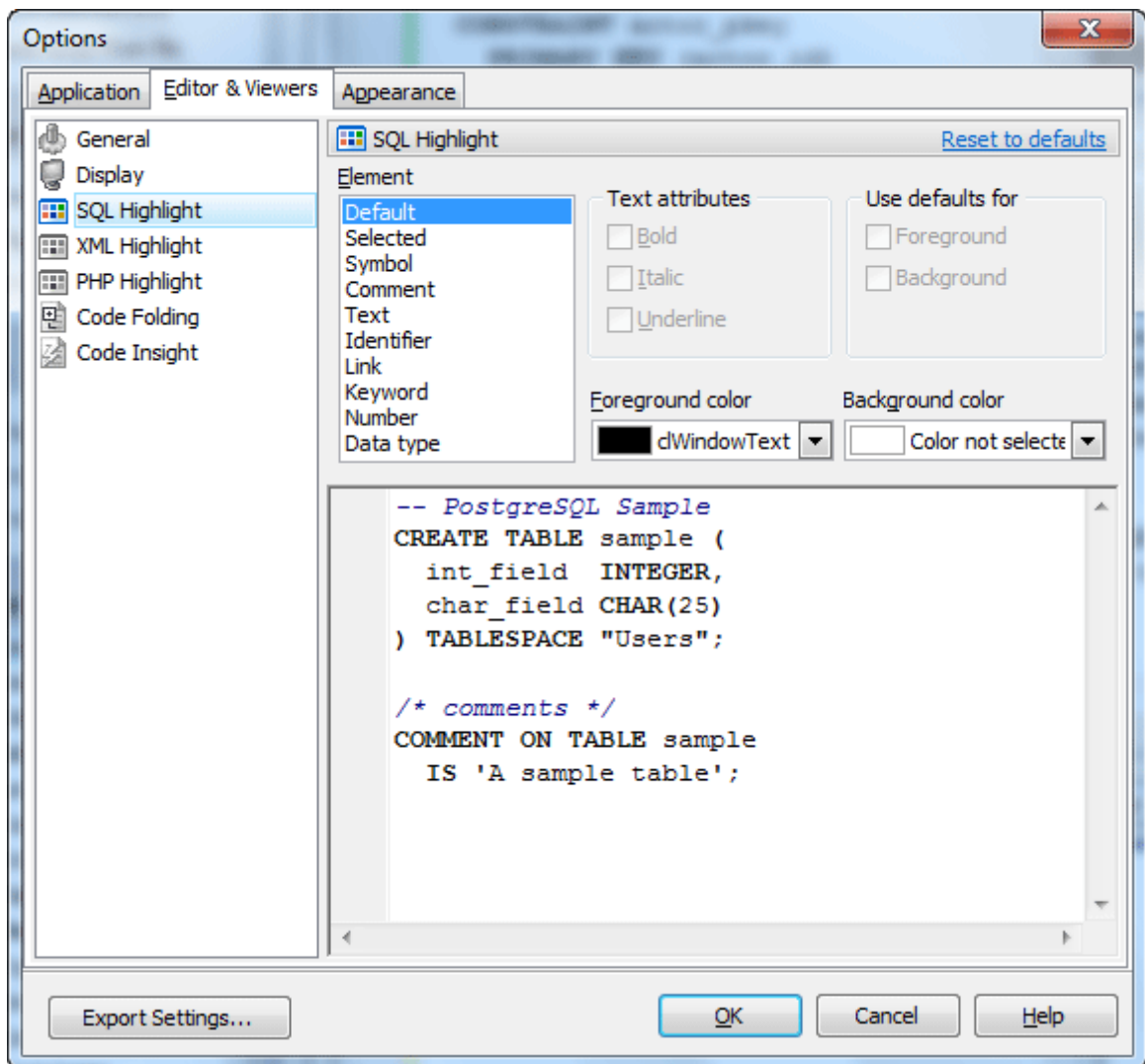
You can disable/enable the right text margin and the gutter of the editor area, set the position of the right text margin as [Right margin](#), and [the Gutter width](#).

Use the [Editor font](#) and [Font size](#) to define the font used in all program editors and viewers. The panel below displays the sample of the selected font.



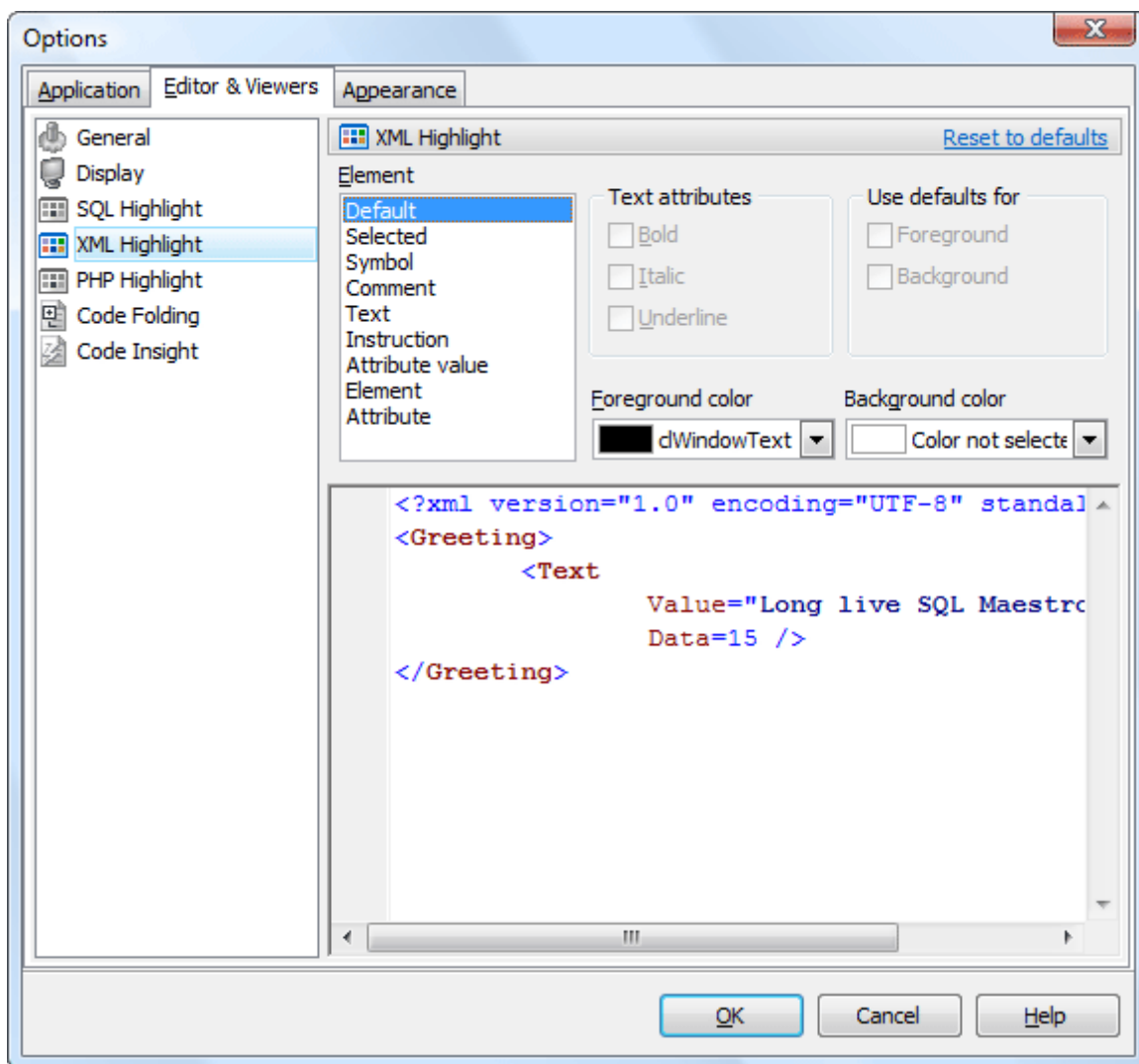
10.2.3 SQL highlight

Use the [SQL highlight](#) item to customize syntax highlight in all SQL editors and viewers, e.g. in [SQL Editor](#), [Query Builder](#), [Table Editor](#) and others. Select the text element from the list, e.g. *comment* or *SQL keyword* and adjust its foreground color, background color and text attributes according to your preferences.



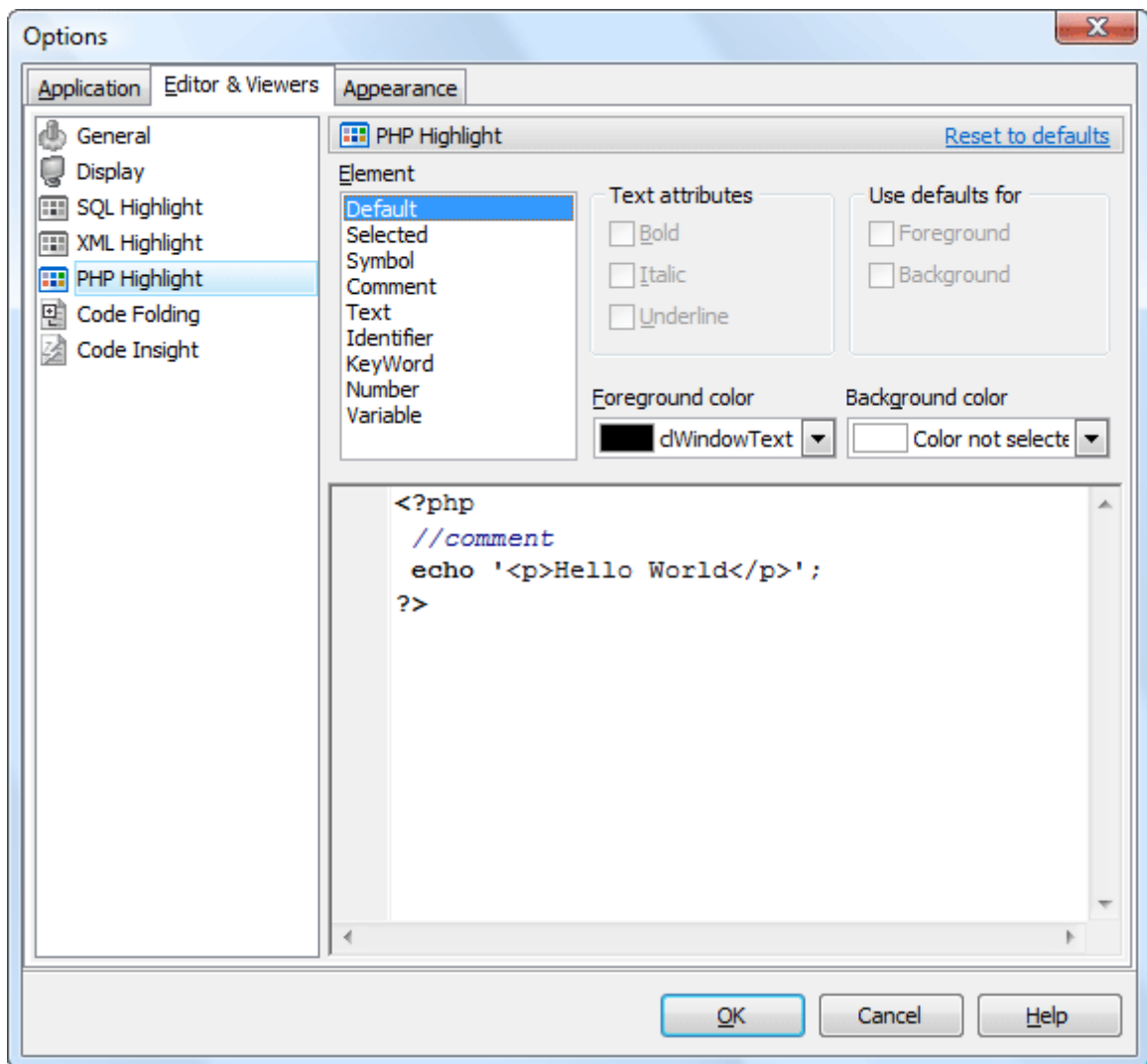
10.2.4 XML highlight

Use the [XML highlight](#) item to customize XML syntax highlight for the text representation of BLOBs in [BLOB Viewer/Editor](#). Select the text element from the list, e.g. attribute or attribute value and adjust its foreground color, background color and text attributes according to your wishes.



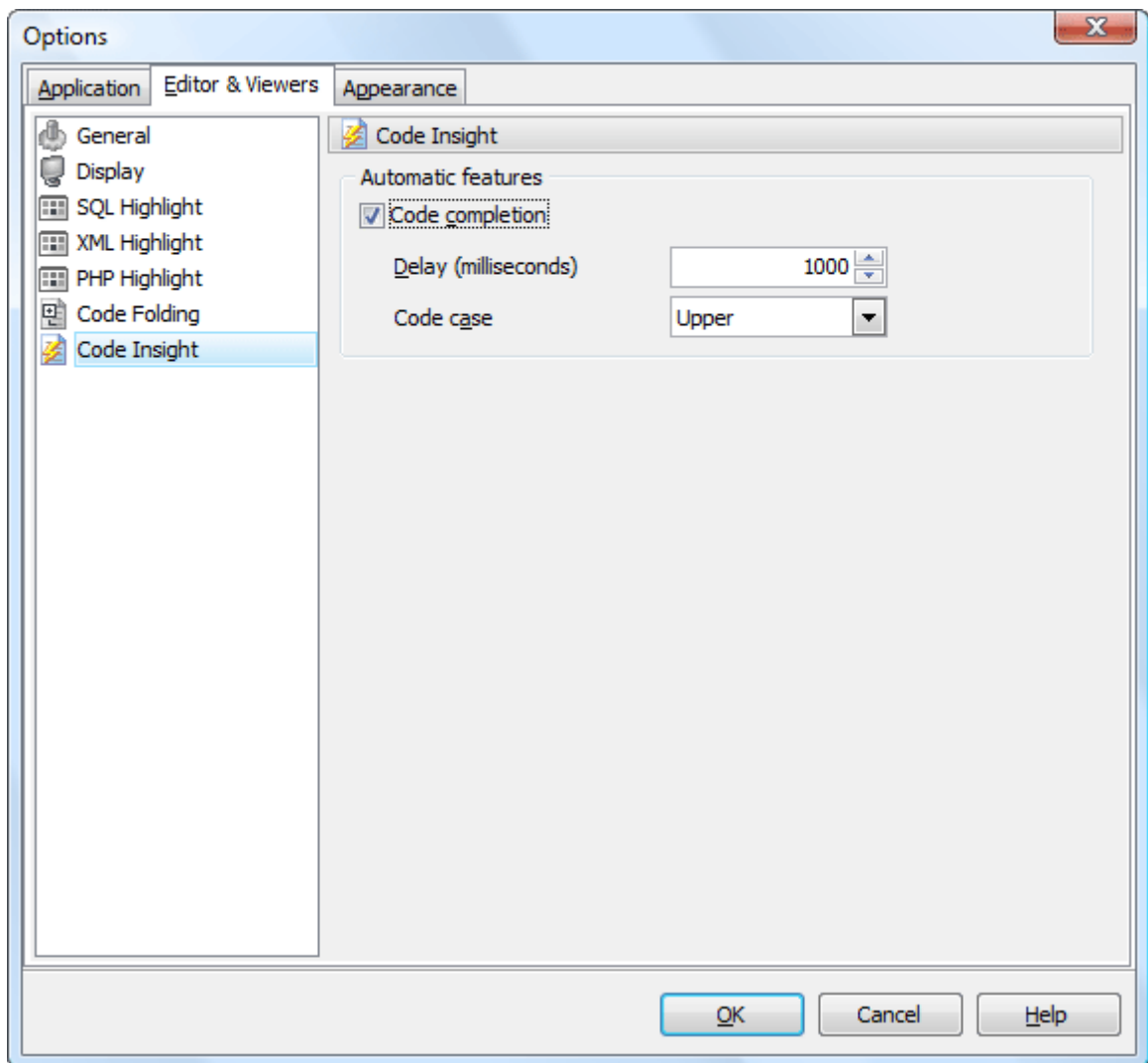
10.2.5 PHP highlight

Use the **PHP highlight** item to customize PHP syntax highlight for the text representation of BLOBs in **BLOB Viewer/Editor**. Select the text element from the list (e.g. Keyword, Comment, Identifier), and adjust its foreground color, background color and text attributes according to your wishes.



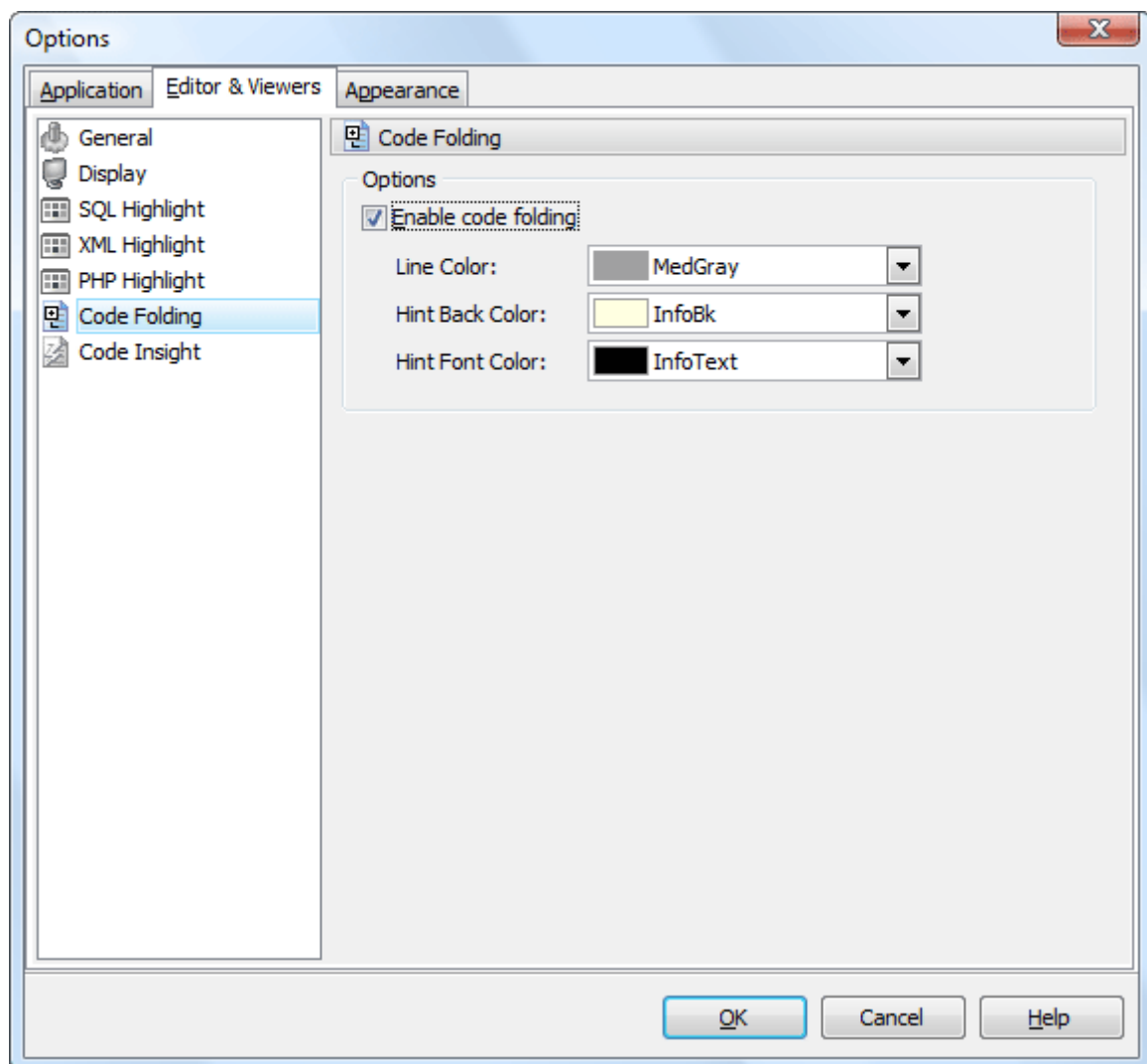
10.2.6 Code Insight

You can disable/enable the code completion with the corresponding option and also set the time it appears as *Delay*, and case of words inserted automatically.



10.2.7 Code Folding

The [Code Folding](#) item group makes it possible both to view the whole text and to divide it into logical parts (regions). Each part can be collapsed and extended. In extended mode the whole text is displayed (set by default), in collapsed mode the text is hidden behind one text line denoting the first line of the collapsed region.



You can enable/disable code folding in SQL editors and viewers and customize the colors of its items.

10.3 Appearance

The [Appearance](#) section allows you to customize the application interface style to your preferences.

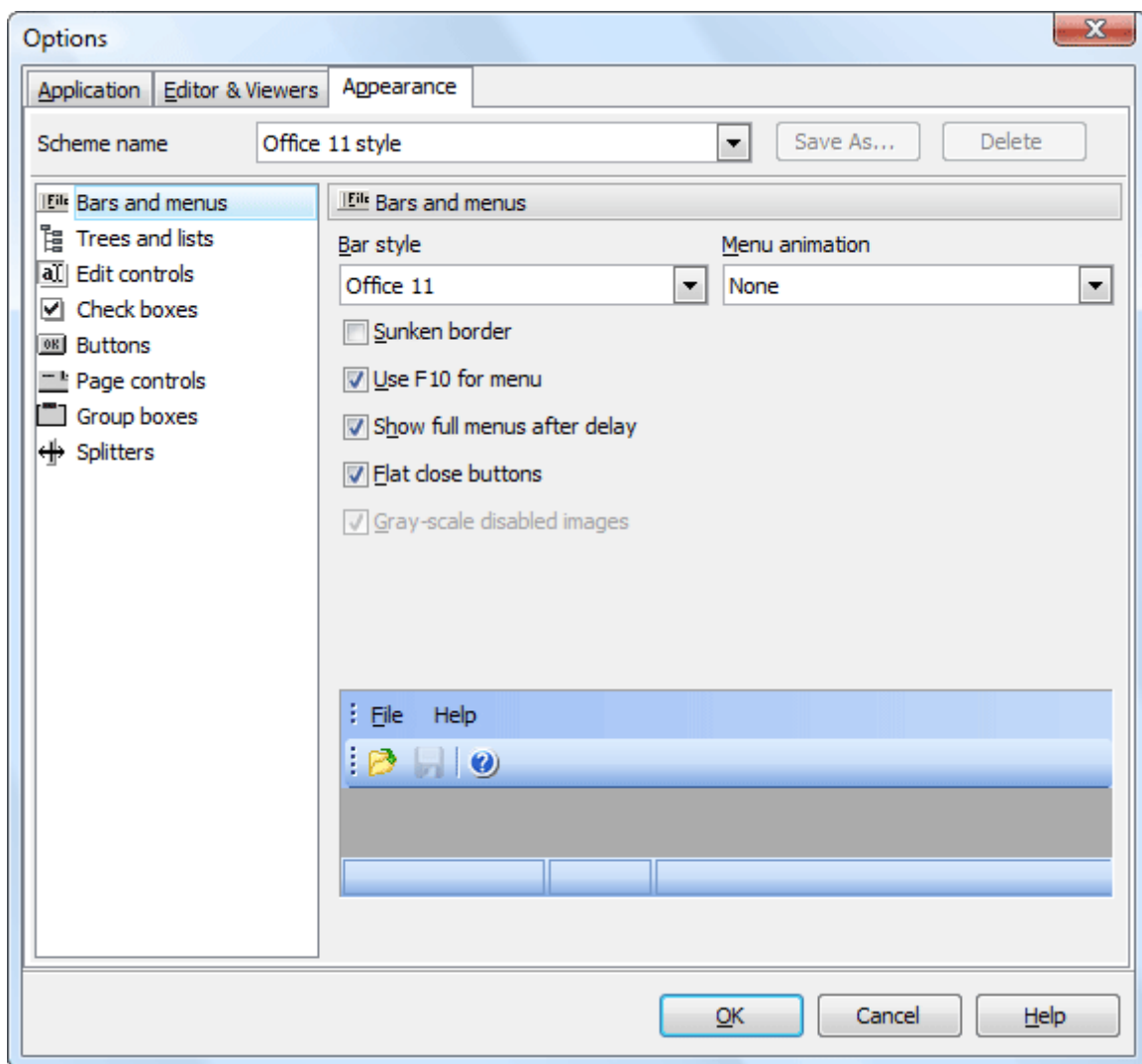
Use the [Scheme name](#) box to select the interface scheme you prefer: *Office XP style*, *Windows XP native style*, etc. You can create your own interface schemes by customizing any visual options ([Bars and menus](#), [Trees and lists](#), [Edit controls](#), [Check boxes](#), [Buttons](#), etc.) and clicking the [Save As](#) button. All the customized options are displayed on the sample panel.

- [Bars and menus](#) ³⁵²
- [Trees and lists](#) ³⁵³
- [Edit controls](#) ³⁵⁴
- [Check boxes](#) ³⁵⁵
- [Buttons](#) ³⁵⁶
- [Page controls](#) ³⁵⁷
- [Group boxes](#) ³⁵⁸
- [Splitters](#) ³⁵⁹

10.3.1 Bars and menus

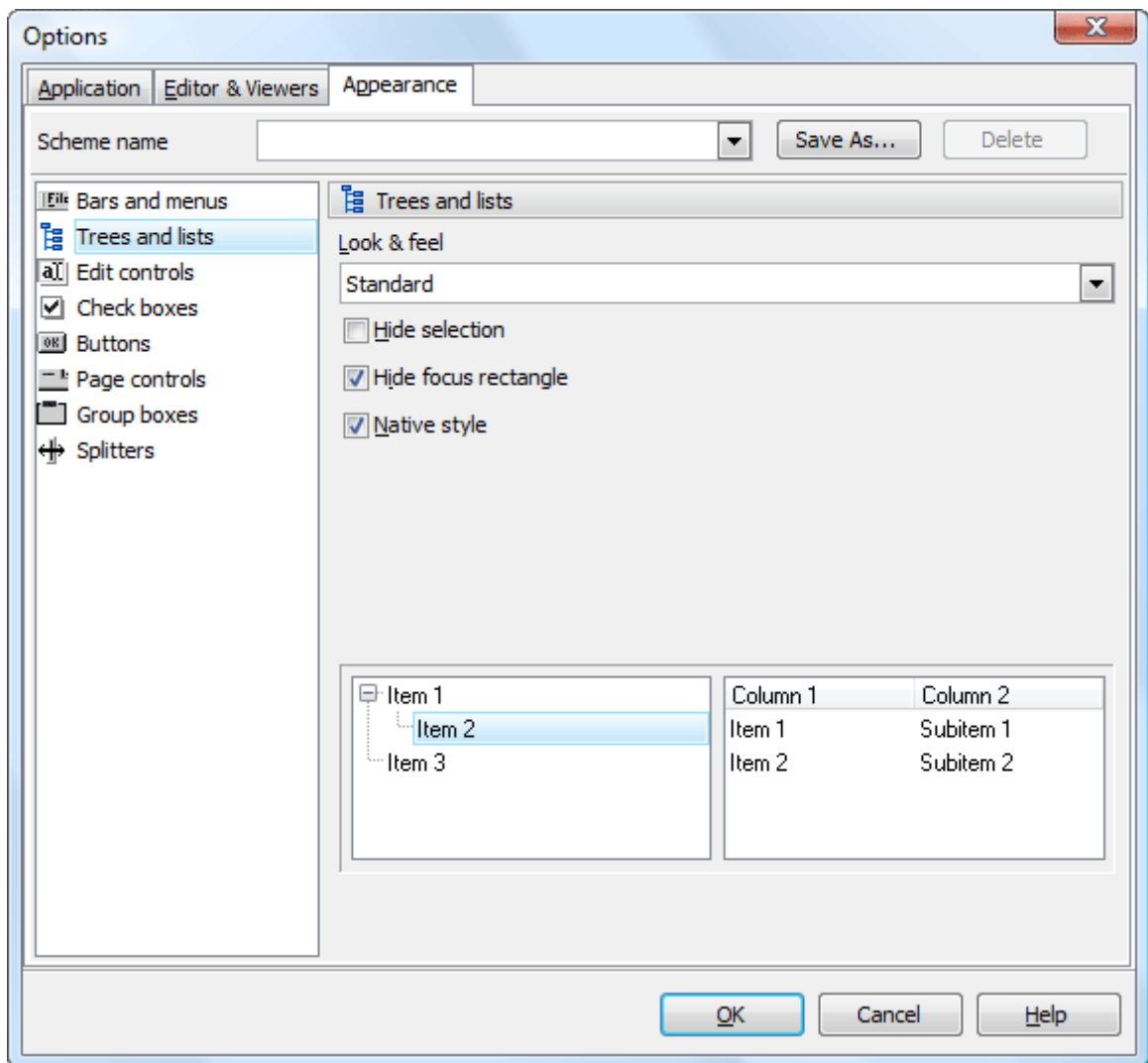
Use the [Bars and menus](#) item to customize PostgreSQL Maestro toolbars style and menus animation.

The item allows you to select Bar style and menu animation from the corresponding drop-down lists and to enable or disable such options as sunken border, F10 key for opening menu, viewing full menus after delay, flat close buttons, gray-scale images.



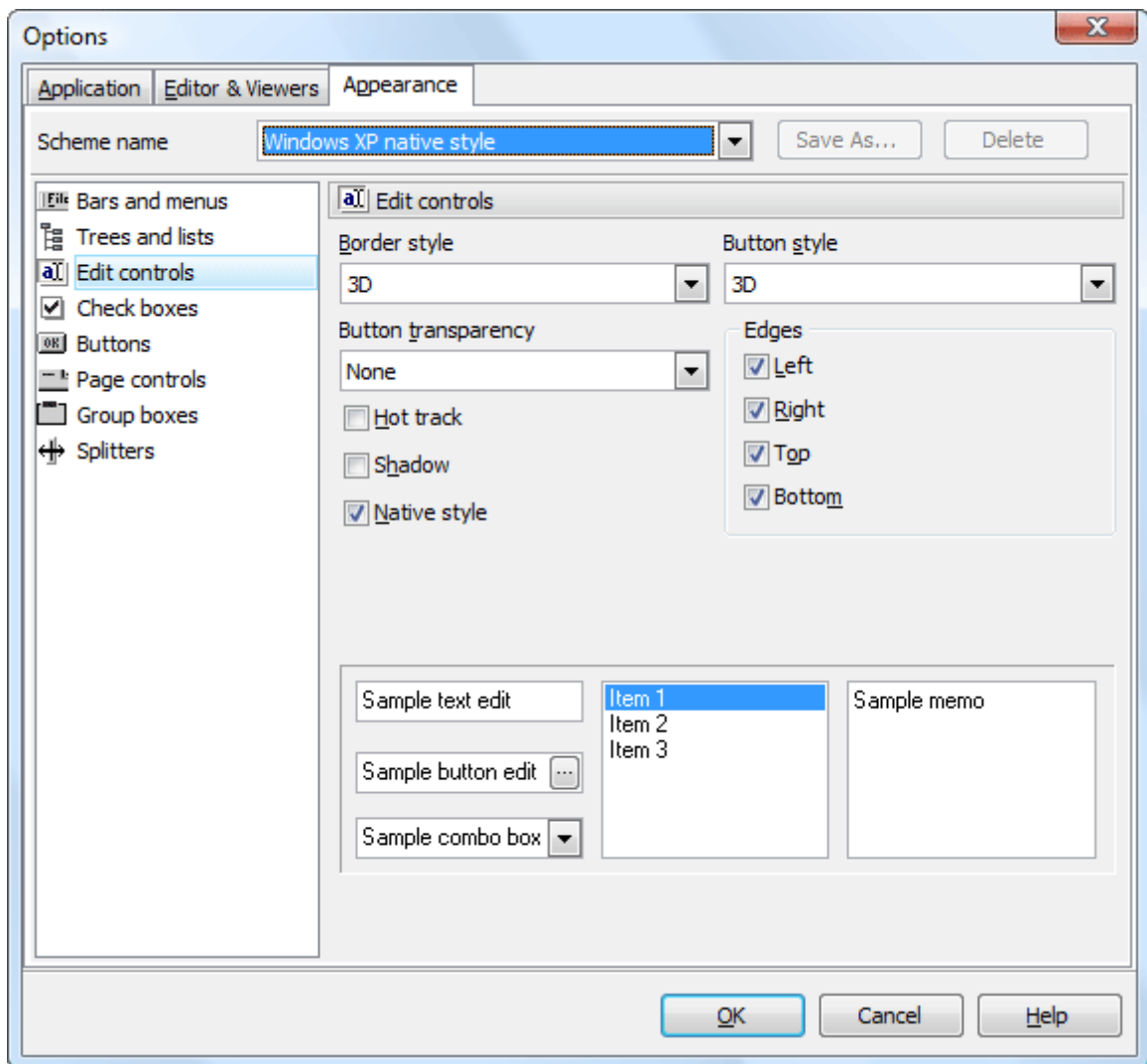
10.3.2 Trees and lists

Use the **Trees and lists** item to select various tree view options. Use the item to select *standard*, *flat* or *ultraflat* styles, check or uncheck the *hide selection*, *hide focus rectangle* and *native style* options.



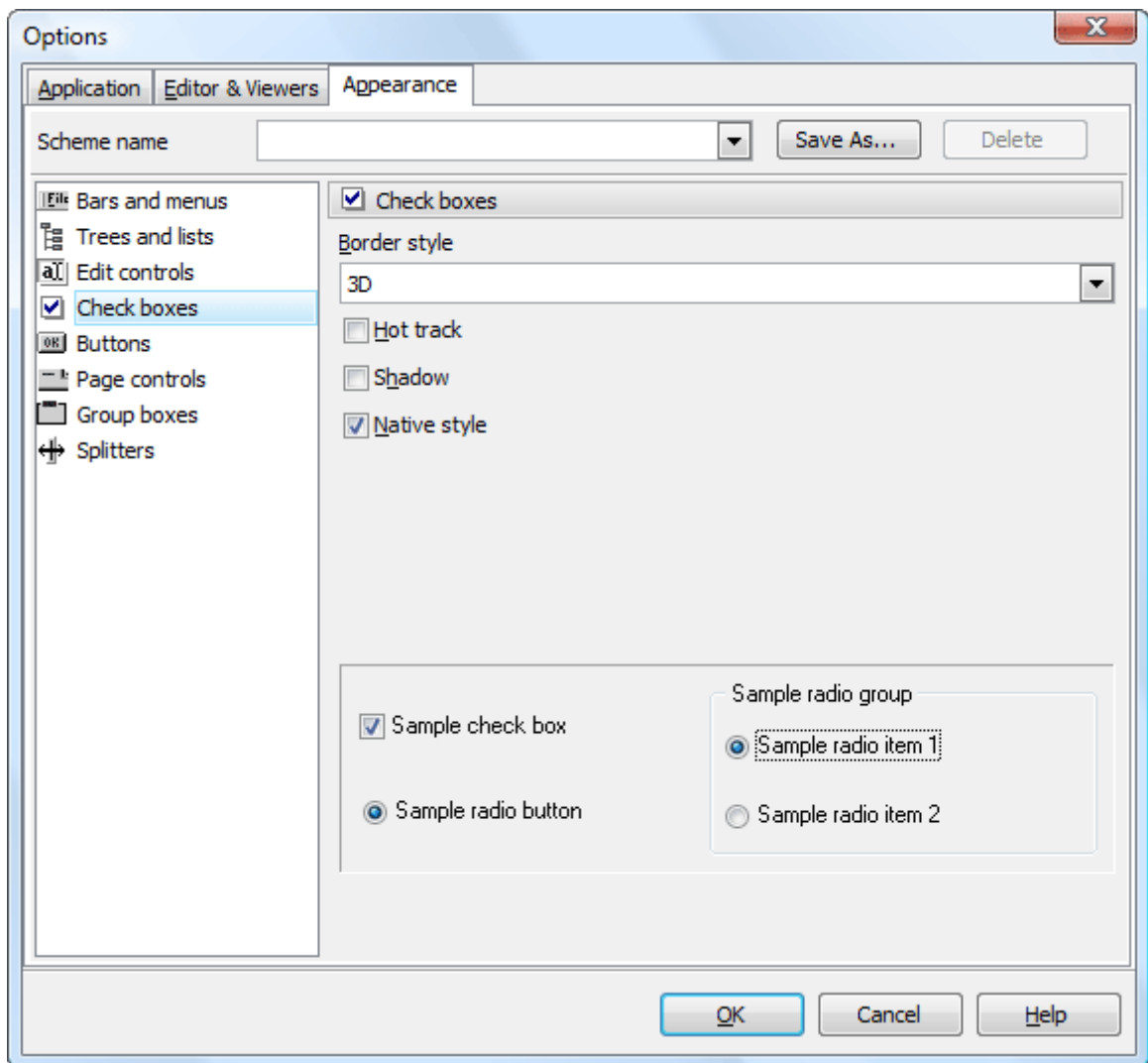
10.3.3 Edit controls

Use the [Edit controls](#) item to customize the appearance of different PostgreSQL Maestro edit controls. The tab allows you to select the edit controls border style, button style and transparency, enable/disable hot tracks, shadows, native style and customize edges. It is also possible to define samples for the text edit, button edit and combo box controls.



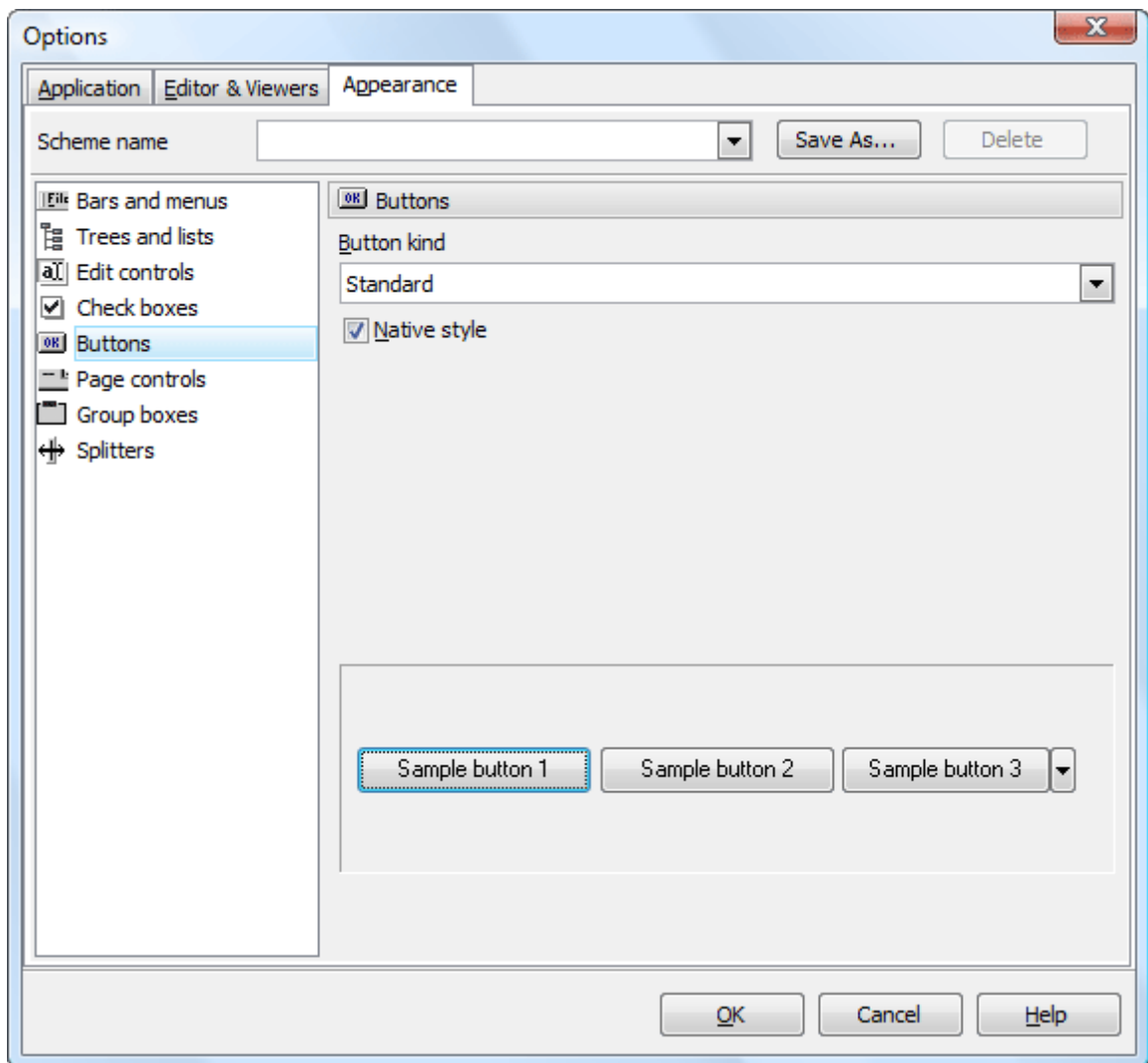
10.3.4 Check boxes

The [Check boxes](#) item allows you to customize the appearance of check boxes and radio buttons. The tab allows you to customize the appearance of check boxes: set border style, enable/disable hot tracks, shadows, native style. It is also possible to define samples for check boxes and radio buttons.



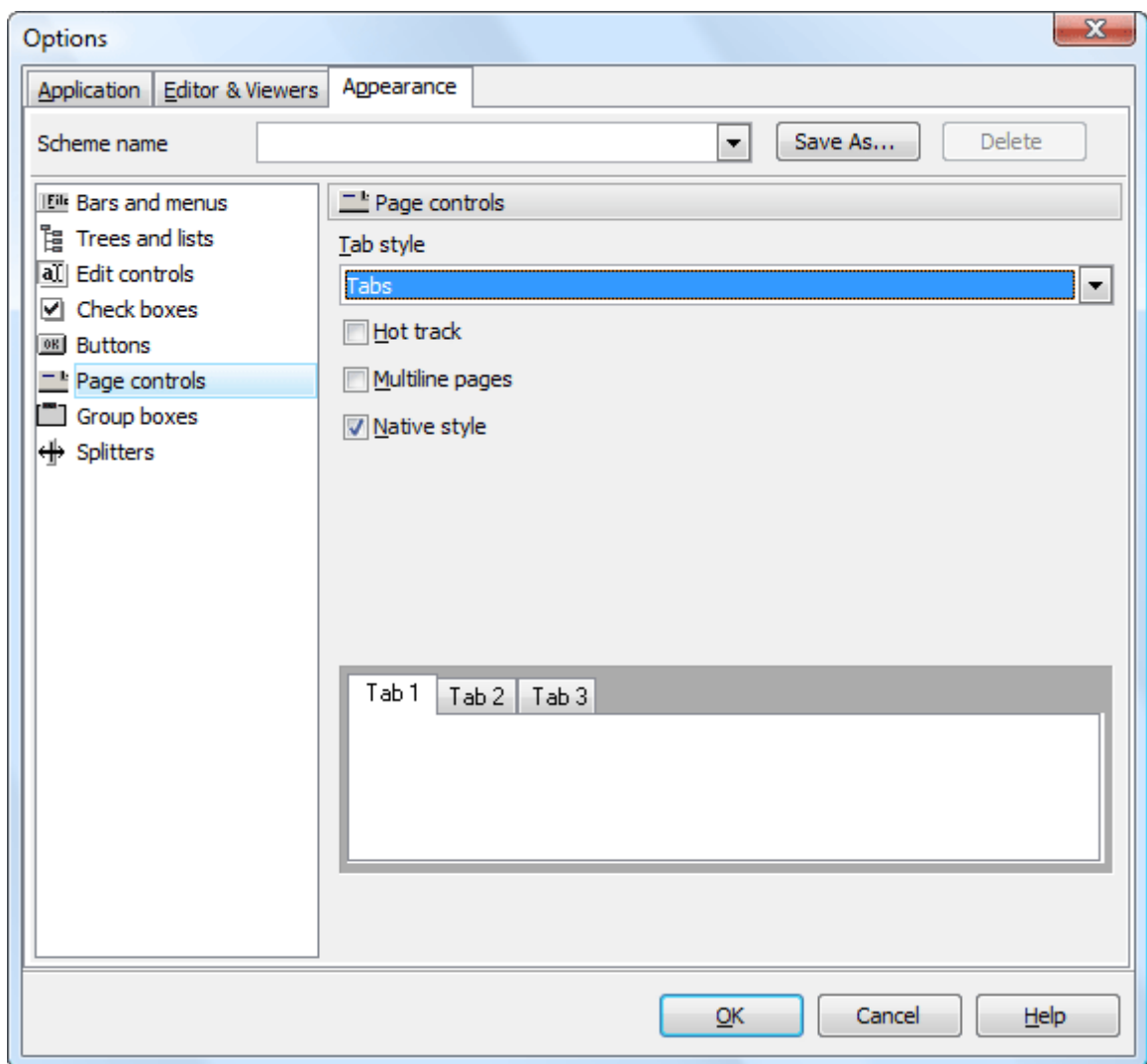
10.3.5 Buttons

Use the [Buttons](#) item to customize PostgreSQL Maestro buttons. The tab allows you to adjust the appearance of buttons and define sample buttons as well.



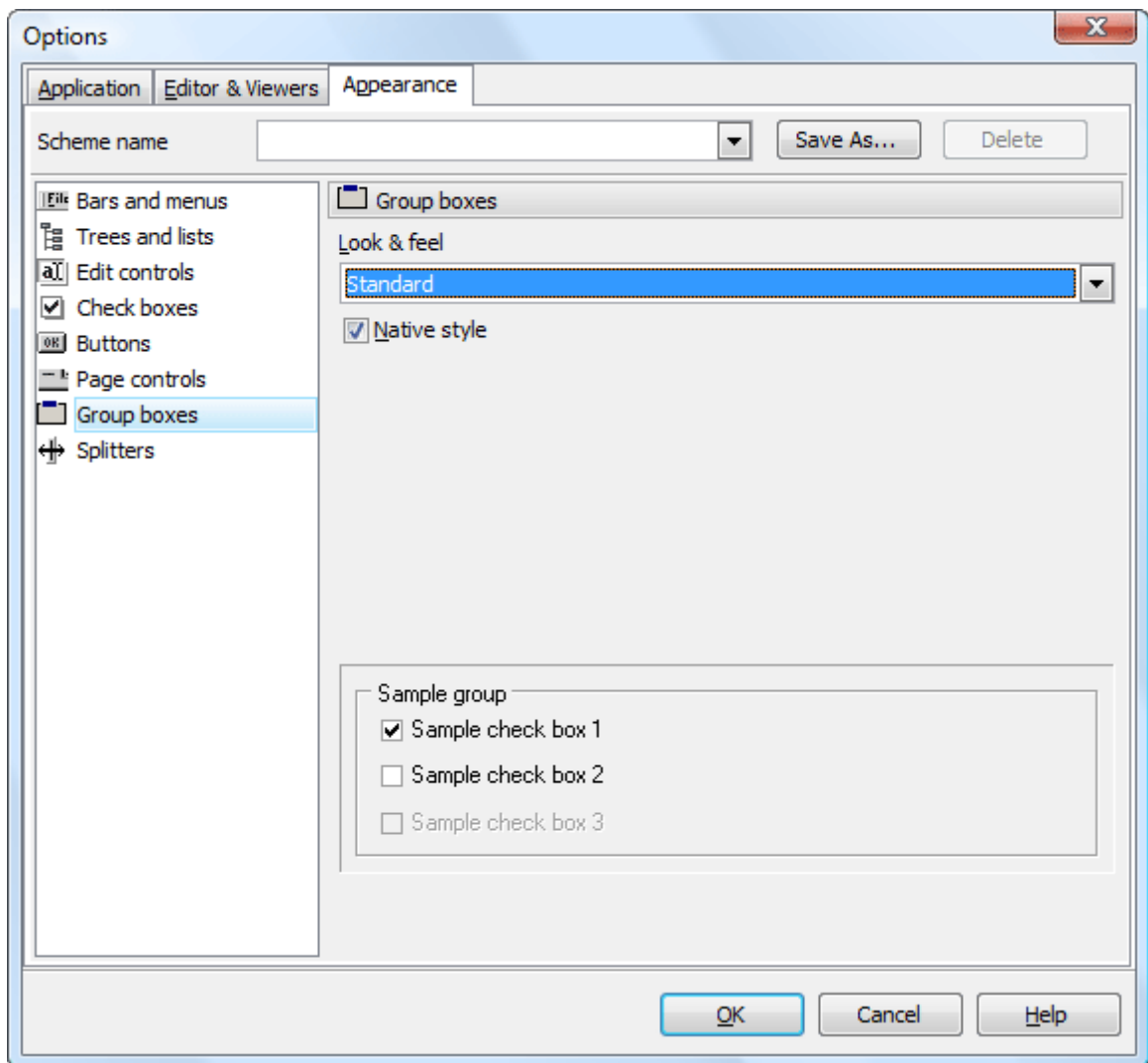
10.3.6 Page controls

The [Page controls](#) item allows you to customize the style of all PostgreSQL Maestro page controls. The tab allows you to select tab styles, enable/disable hot track, multi-line pages and native style.



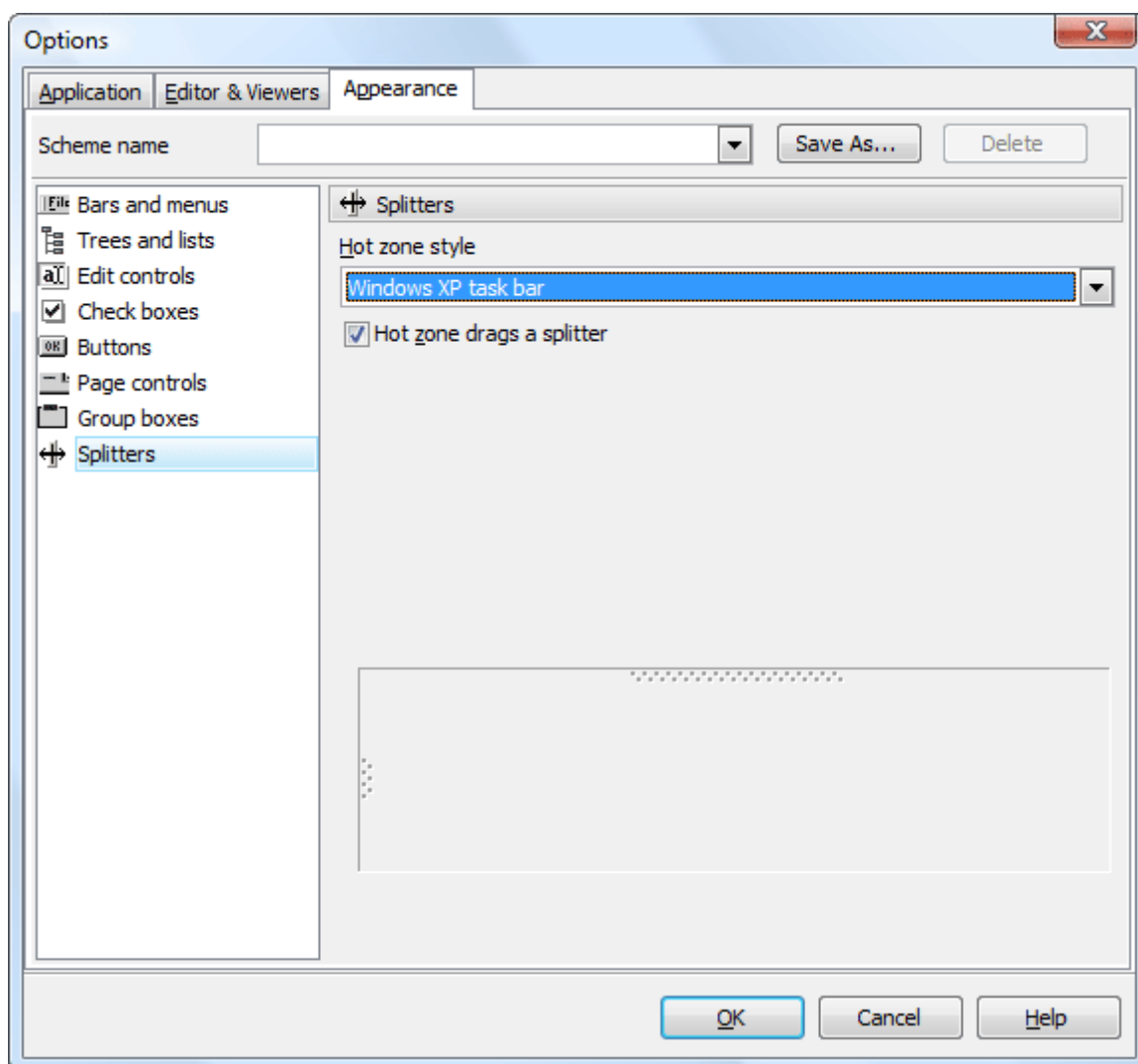
10.3.7 Group boxes

Use the [Group boxes](#) item to customize all PostgreSQL Maestro group boxes according to your preferences. Use tab to apply styles for group boxes, enable/disable native style and define samples.



10.3.8 Splitters

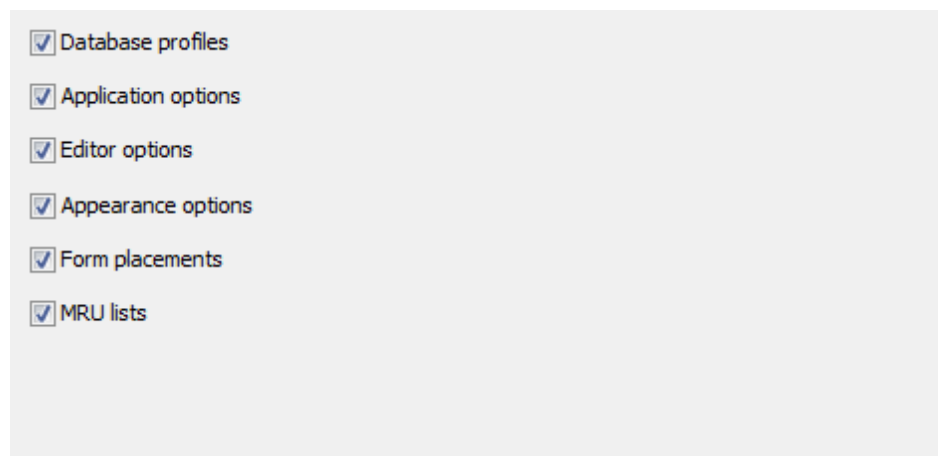
Use the [Splitters](#) item to customize all PostgreSQL Maestro splitters according to your preferences. Use the tab to select hot zone style (*Windows XP task bar*, *Media Player 8*, *Media Player 9*, *Simple* or *none*) and specify the [Hot zone drags a splitter](#) option.



10.4 Export Settings

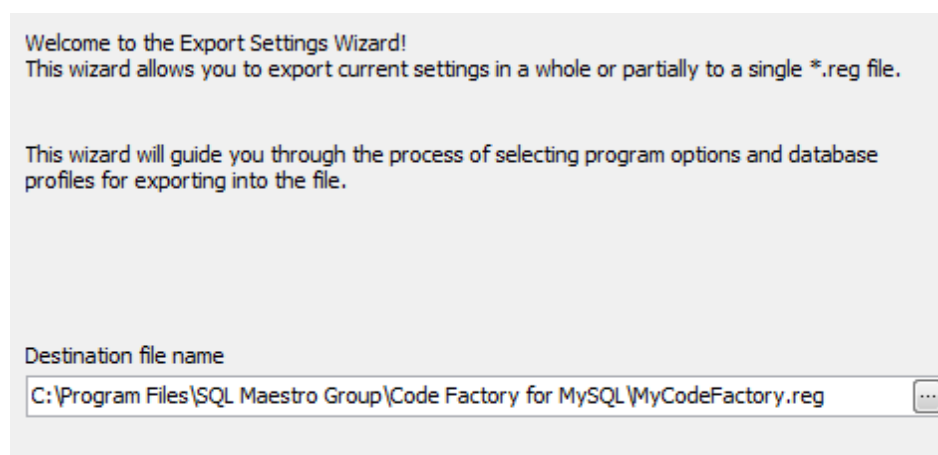
Export Settings Wizard allows you to export all or partial PostgreSQL Maestro settings to single *.reg file which can be applied to the application of PostgreSQL Maestro installed on another machine or used to backup previous settings. To run the wizard, select the Tools | Options main menu item and click Export Settings in the [Options](#)^[361] dialog.

- [Specifying destination file to save settings to](#)^[361]
- [Specifying settings categories to save](#)^[361]
- [Select database profiles to save](#)^[362]
- [Saving settings](#)^[362]



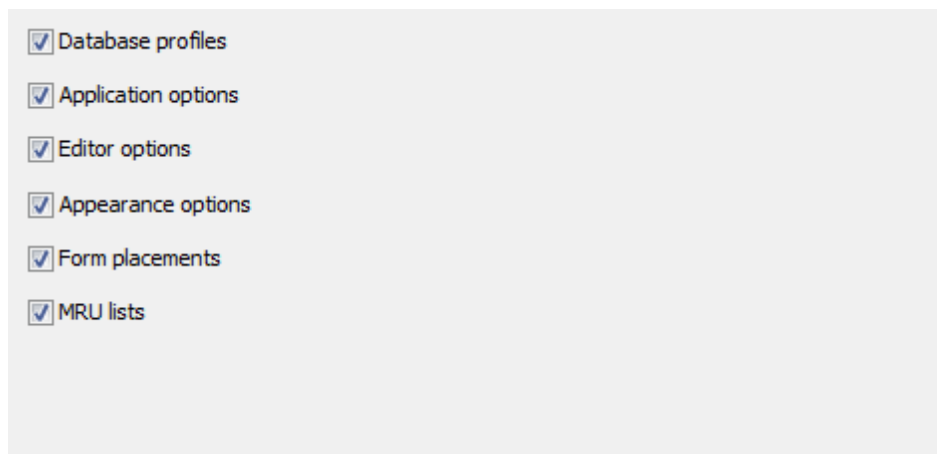
10.4.1 Specifying destination file

Specify a *.reg file to extract PostgreSQL Maestro setting to.



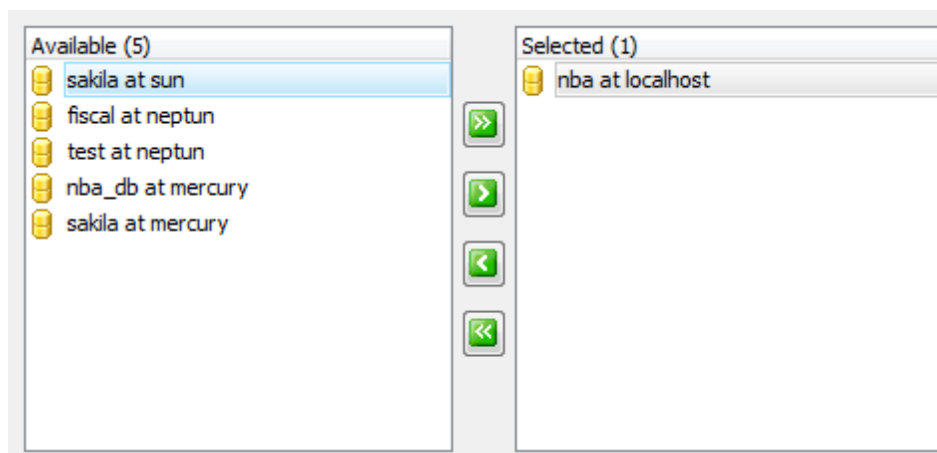
10.4.2 Selecting setting categories

The options of this step specify the information saved to the result file, e.g. Database profiles, [Application options](#)^[322], etc.



10.4.3 Selecting database profiles

Select database profiles to save their settings by moving them from the [Available Databases](#) list to the [Selected Databases](#) one.

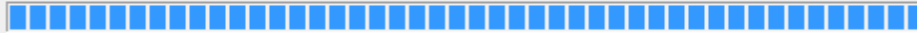


10.4.4 Saving settings

Click the [Ready](#) button to start the extracting. The process log is displayed in the [Export log](#) box.

Export log

The command(s) completed successfully.
Exporting editor options...
The command(s) completed successfully.
Exporting appearance options...
The command(s) completed successfully.
Exporting form placements...
The command(s) completed successfully.
Exporting MRU lists...
The command(s) completed successfully.



Click "Ready" to export settings.

Index

- A -

- Aggregate Editor 141
 - Editing aggregate properties 142
- Aggregates 139
 - Aggregate Editor 141
 - Create Aggregate Wizard 140
- Appearance Options
 - Bar and menus 352
 - Buttons 356
 - Check boxes 355
 - Edit controls 354
 - Group boxes 358
 - Page controls 357
 - Splitters 359
 - Trees and lists 353

- B -

- BLOB Editor 240
 - Hexadecimal mode 241
 - HTML mode 242
 - Image mode 240
 - Navigating withing the BLOB editor 240
 - PDF mode 243
 - Plain text mode 241
- BLOB Viewer 276
 - Hexadecimal mode 276
 - HTML mode 279
 - Image mode 278
 - PDF mode 280
 - Plain text mode 277

- C -

- Cast Editor 175
 - Editing cast properties 175
- Casts 172
 - Cast Editor 175
 - Create Cast Wizard 173
- Checks 91
- Collations 167
- Composite Type Editor 157

- Editing type properties 158
- Composite Types 154
 - Composite Type Editor 157
 - Create Composite Type Wizard 155
- Create Aggregate Wizard 140
- Create Cast Wizard 173
- Create Composite Type Wizard 155
- Create Database Profiles Wizard 26
 - Setting connection properties 26
 - Setting profile options 27
- Create Database Wizard 37
 - Setting connection properties 37
 - Specifying database properties 38
- Create Domain Wizard 134
- Create Enum Type Wizard 155
- Create Function Wizard 126
- Create Group Wizard 199
- Create Language Wizard 169
- Create Materialized View Wizard 119
- Create object wizard - basic principles 43
 - Setting object name 44
 - Viewing common information 45
- Create Operator Wizard 163
- Create Role Wizard 201
 - Managing role content 203
 - Specifying role proprties 202
- Create Rule Wizard 101
- Create Schema Wizard 68
- Create Sequence Wizard 145
- Create Table Wizard 74
- Create Tablespace Wizard 209
- Create Trigger Wizard 97
- Create Type Wizard 149
- Create updatable views 312
- Create User Wizard 195
- Create View Wizard 109
- CRUD Procedures Generation 310

- D -

- Data Analysis 286
 - Data 288
 - Query 287
- Data Management 228
 - BLOB editor 240
 - Data View 229
 - Export Data Wizard 245
 - Get SQL Dump Wizard 252

- Data Management 228
 - Import Data Wizard 255
 - Data View
 - Editing data in dialog 236
 - Lookup editor 236
 - Viewing data 229
 - Working with data grid 230
 - Working with info cards 235
 - Database Editor 40
 - Database Explorer
 - Filtering explorer content 61
 - Database Management 24, 190
 - Creating a database 37
 - Creating a database profile 26
 - Database editor 40
 - Editing a database profile 30
 - Database Object
 - Functions 125
 - Materialized Views 118
 - Schemas 67
 - Tables 73
 - Views 108
 - Database Objects 66
 - Casts 172
 - Changing object properties 52
 - Copy, Paste and Drag-n-Drop features 56
 - Create object wizard 43
 - Creating a database object 43
 - Database Variables 177
 - Describing object 52
 - Duplicate Object wizard 53
 - Duplicating a database object 53
 - Duplicating selected object 55
 - Languages 168
 - Managing database objects 63
 - Object Editor 46
 - Queries 212
 - Viewing database objects 61
 - Database Profile Editor 30
 - Connection properties 30
 - Database options 31
 - Default directories 33
 - Editing obligatory scripts to execute 34
 - Setting log options 34
 - Statistics 35
 - Database Variable Editor 178
 - Database Variables 177
 - Database Variable Editor 178
 - Diagram Viewer 282
 - Exporting diagram image 284
 - Selecting fields 283
 - DML Procedures Generation 310
 - Domain Editor
 - Editing domain properties 136
 - Domains 133
 - Create Domain Wizard 134
 - Duplicate Object Wizard 53
 - Modifying new object's definition 55
 - Selecting object to duplicate 54
 - Selecting source and destination databases 53
 - Duplicating a database object 53
 - Duplicate Object Wizard 53
 - Duplicating selected objects 55
- ## - E -
- Editor & Viewer Options
 - Code Folding 350
 - Code Insight 349
 - Display 345
 - General 344
 - PHP highlight 348
 - SQL highlight 346
 - XML highlight 347
 - Enum Type Editor 157
 - Editing type properties 158
 - Enum Types 154
 - Create Enum Type Wizard 155
 - Enum Type Editor 157
 - EULA 6
 - Export Data Wizard 245
 - Adjusting data formats 247
 - Selecting fields for export 247
 - Setting common export options 251
 - Setting destination file name 245
 - Setting format-specific options 248
 - Setting header and footer 246
 - Export Settings Wizard
 - Saving settings 362
 - Selecting database profiles 362
 - Selecting setting categories 361
 - Specifying destination file 361
 - Extensions 180
 - Extract Database Wizard 269
 - Customizing script options 272
 - Selecting database 269

Extract Database Wizard 269
 Selecting objects to extract their data 271
 Selecting objects to extract their structure 270

- F -

Field Editor 82
 Filter Panel 61
 Find Text Dialog 317
 Foreign Key References 105
 Foreign Keys 88
 Foreign Servers 185
 User Mappings 186
 Foreign Tables 182
 Function Editor 128
 Editing function properties 129
 Viewing function result 131
 Functions 125
 Create Function Wizard 126
 Function Editor 128

- G -

Generate Database Report Wizard 274
 Editing report style 275
 Selecting reporting elements 274
 Setting report paths 274
 Get SQL Dump Wizard 252
 Selecting fields 252
 Specifying dump options 253
 Getting Started 12
 Explaining user interface 18
 First time started 19
 Switching between windows 22
 Working with databases 13
 Group Editor 199
 Groups 198
 Create Group Wizard 199
 Group Editor 199

- I -

Import Data Wizard 255
 Customizing common options 261
 Data Format 260
 Map builder 259
 Setting fields correspondence 258

Setting source file name 256
 Indexes 84
 Installation instructions 4

- L -

Language Editor 171
 Editing language properties 171
 Languages 168
 Create LanguageWizard 169
 Language Editor 171
 License Agreement 6

- M -

Materialized View Editor 123
 Materialized Views 118
 Create Materialized View Wizard 119
 Materialized View Editor 123

- N -

NOT NULL Constraints 93

- O -

Object Browser 61
 Object editor - basic principles 46
 Function executing 51, 131
 Object dependencies 49
 Object grants 48
 Parameter Editor 50
 Permissions of an object 47
 Results tab 51, 131
 SQL definition 50
 Types of dependencies 49
 Object Management 42
 Changing object properties 52
 Copy, Paste and Drag-n-Drop features 56
 Creating a database object 43
 Databases 190
 Describing object 52
 Duplicating a database object 53
 Managing aggregates 139
 Managing casts 172
 Managing composite types 154
 Managing database variables 177

- Object Management 42
 - Managing domains 133
 - Managing enum types 154
 - Managing functions 125
 - Managing groups 198
 - Managing languages 168
 - Managing materialized views 118
 - Managing objects 63
 - Managing operators 162
 - Managing Queries 212
 - Managing range types 159
 - Managing roles 200
 - Managing sequences 144
 - Managing tables 73
 - Managing tablespaces 208
 - Managing types 148
 - Managing users 194
 - Managing views 108
 - Schemas 67
 - Server Variables 192
 - Viewing objects 61
 - Wizards and Editors 46
- Object Manager 63
- Operator Editor 164
 - Editing operator properties 165
- Operators 162
 - Create Operator Wizard 163
 - Operator Editor 164
- Options 321
 - Appearance 352
 - Application 322
 - Application confirmations 323
 - Application preferences 322
 - BLOB Viewer 332
 - Data Grid 337
 - Designer 334
 - Editor & Viewers 344
 - Editors 335
 - Explorer 327
 - Export 333
 - Export Settings 361
 - Object Manager 328
 - Query Builder 330
 - SQL Editor 328
 - SQL Script Editor 329
 - Tools 325

- P -

- PL/SQL Debugger 303
 - Debug information 306
 - Debugging process 304
- PostgreSQL Maestro 1
 - Getting started 12
 - Installation 4
 - License agreement 6
 - Registration 5
 - System requirements 3
- Purchase PostgreSQL Maestro 5

- Q -

- Queries 212
 - Query Parameters 218
 - SQL Editor 214
 - Visual Query Builder 219
- Query Parameters 218

- R -

- Range Types 159
 - Create Range Type Wizard 160
 - Range Type Editor 161
- Registration 5
- Replace Text Dialog 318
- Report Designer 291
 - Object Inspector 295
 - Toolbox 293
- Role Editor 204
 - Editing role properties 205
 - Viewing role objects 207
- Roles 200
 - Create Role Wizard 201
 - Role Editor 204
- Row Security Policies 106
- Rule Editor 103
- Rules 100
 - Create Rule Wizard 101
 - Rule Editor 103

- S -

- Schema Designer 298

- Schema Designer 298
 - Navigation bar 300
 - Toolbox 300
- Schema Editor 70
- Schema Objects
 - Aggregates 139
 - Composite Types 154
 - Domains 133
 - Enum Types 154
 - Operators 162
 - Range Types 159
 - Sequences 144
 - Types 148
- Schemas 67
 - Create Schema Wizard 68
 - Schema editor 70
- Sequence Editor 146
- Sequences 144
 - Create Sequence Wizard 145
 - Sequence Editor 146
- Server Editor 189
- Server Management 189
- Server Objects 187
 - Changing object properties 52
 - Copy, Paste and Drag-n-Drop features 56
 - Create object wizard 43
 - Databases 190
 - Describing object 52
 - Groups 198
 - Object Editor 46
 - Roles 200
 - Server Variables 192
 - Tablespaces 208
 - Users 194
- Server Variables 192
 - Server Variable Editor 192
- Split table 313
- SQL Editor 214
 - Executing query 216
- SQL Script Editor 266
- System requirements 3
- Viewing table data 79
- Tables 73
 - Checks 91
 - Create Table Wizard 74
 - Fields 82
 - Foreign key references 105
 - Foreign keys 88
 - Indexes 84
 - NOT NULL Constraints 93
 - Row Security Policies 106
 - Rules 100
 - Table editor 77
 - Triggers 95
- Tablespace Editor 210
 - Editing tablespace properties 210
- Tablespaces 208
 - Create Tablespace Wizard 209
 - Tablespace Editor 210
- Tools 263
 - BLOB Viewer 276
 - Data Analysis 286
 - Database Designer 298
 - Dependency Tracker 308
 - Diagram Viewer 282
 - Dialogs 317
 - Extract Database Wizard 269
 - Generate Database Report Wizard 274
 - PL/SQL Debugger 303
 - Process Browser 307
 - Report Designer 291
 - Schema Designer 298
 - Script Runner 265
 - SQL Generator 309
 - SQL Script Editor 266
- Trigger Editor 98
- Triggers 95
 - Create Trigger Wizard 97
 - Trigger Editor 98
- Type Editor 151
 - Editing type properties 151
- Types 148
 - Create Type Wizard 149
 - Type Editor 151

- T -

- Tabbed MDI 18
- Table Editor 77
 - Editing table properties 77
 - Master-detail data view 79

- U -

- User Editor 196
- User interface 18

Users 194
 Create User Wizard 195
 User Editor 196

- V -

View Editor 114
 Editing view properties 115
 Viewing data 116
Views 108
 Create View Wizard 109
 View Editor 114
Visual Query Builder 219
 Executing a query 226
 Working with editor area 225

- W -

WHERE condition 109, 119
Window List 22
Wizards and Editors 46
 Create object wizard - basic principles 43
 Object Editor 46