SQLite Code Factory

User's guide

Table of Contents

VI Data management 58

VII Database Tools 93

1 Welcome to SQLite Code Factory!

SQLite Code Factory is a premier SQLite GUI tool aimed at the SQL queries and scripts development. It is a good choice for everyone who need build SQL statements and edit SQL scripts with a convenient easy-to-use interface. The software provides you with a convenient easy-to-use interface, so it really does not require a deep knowledge of SQLite from its users.

Key features include:

- **Visual Query Builder**: SQLite Code Factory provides you with the powerful tool intended for designing queries as visual diagrams. This tool does not require any knowledge of the SELECT statement syntax, it will form a query automatically, you just need to mark what information you want to retrieve;
- **Handy SQL Editor** with code folding and syntax highlighting to prevent mistakes in syntax at once. Also it is possible to separate SQL scripts into regions that can be individually collapsed or expanded;
- **Simultaneous executing** of several queries with multi-threading in order to continue your work with the software while the query is executing;
- **Advanced data management:** viewing, editing, grouping, sorting and filtering abilities to analyze the data in the most convenient way;
- **Data export** to as many as 14 file formats including Excel, RTF and HTML;
- **Data import** from Excel, CSV, text files and more;
- **Powerful BLOB Viewer/Editor:** with SQLite Code Factory you can view or edit BLOB data in the following ways: hexadecimal dump, plain text, graphical image or HTML page. A graphical representation of BLOB data supports five image formats: BMP, Windows metafile, JPEG, GIF and PNG.

The application also provides you with a powerful set of tools to edit and execute SQL scripts, build visual diagrams for numeric data, customize user interface according to your needs and much more.

With all these features our software will be an everyday assistant in your work with SQLite database server!

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

- SQLite 2.8/3.x.

1.2 Installation

To install **SQLite Code Factory** on your PC:

- download the SQLite Code Factory 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 SQLite Code Factory shortcut in the corresponding program group of the Windows Start menu after the installation is completed.

1.3 How can I purchase SQLite Code Factory?

Thank you for your interest in purchasing **SQLite Code Factory**!

You can select licensing options and register SQLite Code Factory 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.

SQLite Code Factory 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 SQLite 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 SQLite .

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 SQLite Code Factory news at http://www.sqlmaestro.com/products/sqlite/codefactory/news/

2 Getting Started

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

How to get started:

- Connect to a database with SQLite Code Factory 12
- Explaining user interface 15
- How SQLite Code Factory looks when you start it for the first time 16
- Shortcut keys 20

Learning more:

- See **Database Tools** 93 section for instructions on more advanced procedures!
- Find out more about **Working with Data in SQLite Code Factory** 58.
- Customize the way SQLite Code Factory works, see **Program Options** 109 for full details.

2.1 Connect to a database

To manage an existing database with SQLite Code Factory, you have to create the according database profile 23 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 25).

When the profile is created you can connect to the database. To do so, select the database in the Explorer tree, 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 13

2.2 Connection parameters

As SQLite is implemented as an embedded database engine contained in a single DLL, SQLite databases usually are stored locally or in the shared folders. To connect to such database, you should provide only a full database file name (e.g. C:\Data\SQLite\MyDatabase.db3) and a password (only for encrypted databases).

To read and write encrypted databases, SQLite Code Factory uses the free wxSQLite3 library that is included into the installation package. This means it can operate only with encrypted databases created by itself or by any other tool that uses the same library. Unfortunately, our software cannot connect to databases encrypted by any other library because different SQLite security extensions use different algorithms, which are not compatible with each other.

SQLite engine does not support network connections, however SQLite Code Factory allows you to manage remote SQLite databases using the HTTP tunneling technique. For this purpose, you need to have a webserver running on a computer that stores the database file. Of course this webserver should be accessible from your workstation and you should be able to upload files there.

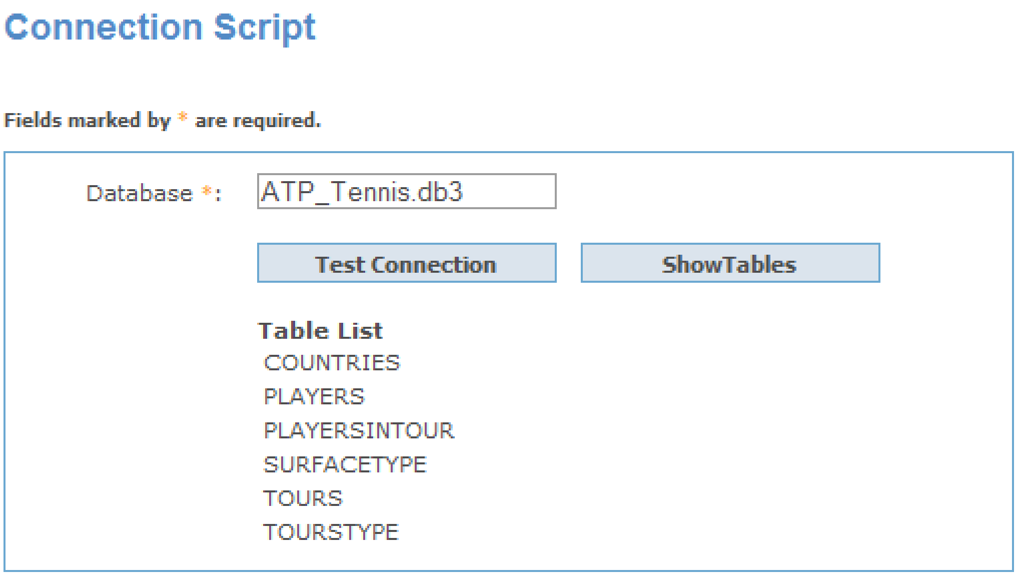
⊟ More about connection via HTTP tunnel

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

1. Upload the connection PHP script to your website. The scripts are named *sqlite_tunnel.php* and *sqlite3_tunnel.php* for SQLite databases versions 2 and 3 accordingly and can be found under the installation folder, usually *C:\Program Files\SQL Maestro Group\SQLite Code Factory*.

2. Turn ON the I have to use HTTP tunneling checkbox.

3. Enter the connection PHP script URL, e.g. *www.yoursite.com/files/sqlite_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 use the Test connection button.



Connection Script

Fields marked by * are required.

Database *: ATP_Tennis.db3

Table List
COUNTRIES
PLAYERS
PLAYERSINTOUR
SURFACETYPE
TOURS
TOURSTYPE

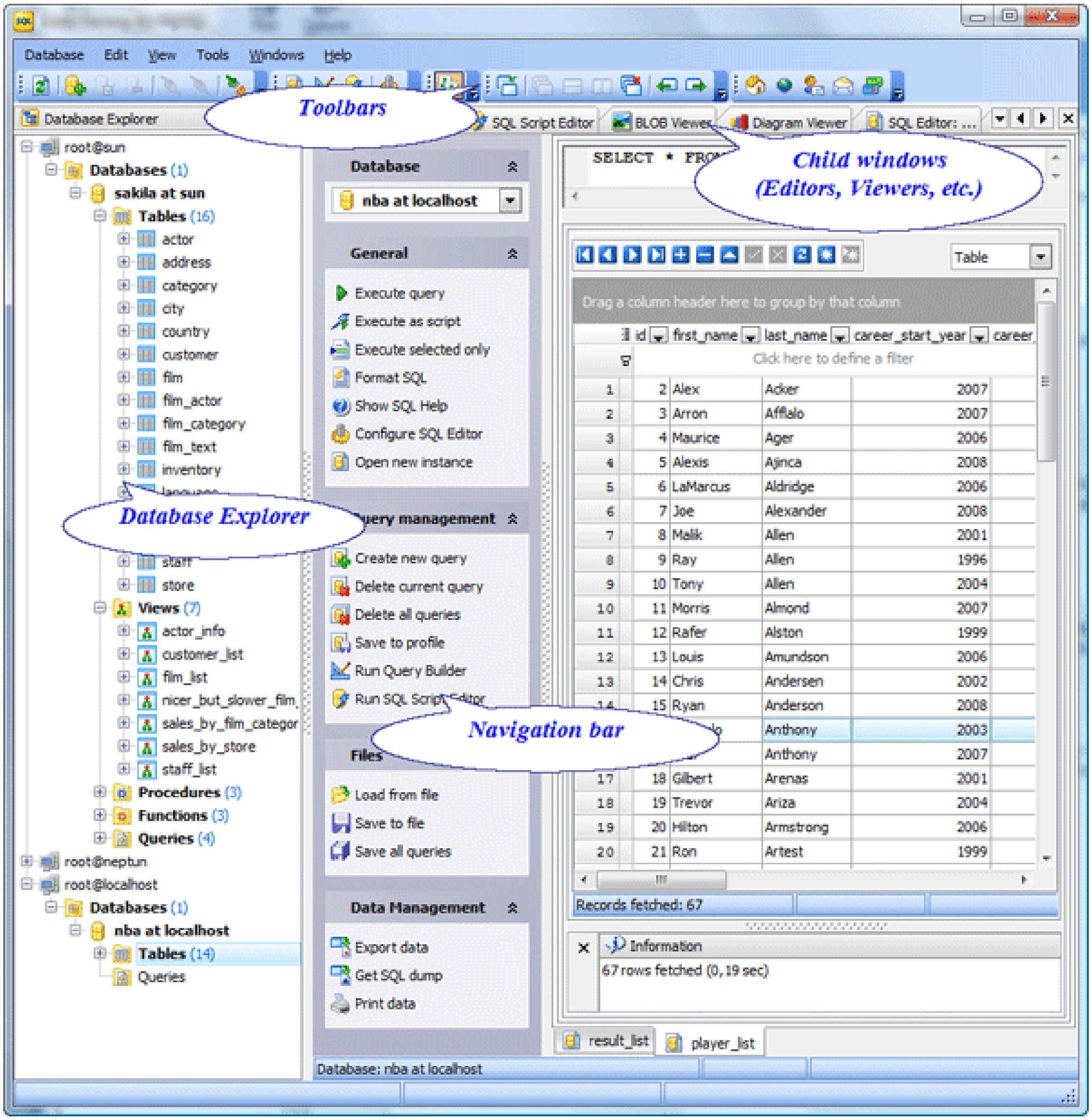
4. In case using of a proxy server use Configure tunnelling options to open the HTTP tunnelling options window and specify your proxy server connection parameters and HTTP authentication.

Note 1. Do not forget to enable read/write permissions for a database file and read/write/execute permissions for the directory where the database file is stored.

Note 2 (only for SQLite 3 databases). The webserver PDO_SQLite library must be compatible (not earlier in the most cases) with the library the database was created with. If they are not compatible, you will get an error message "Could not retrieve table list from _database_name_ ... " on getting a table list at the connection script. If you've got the message, check the PDO_SQLite library version using, for example, the *phpinfo()* function, download a compatible library from the SQLite official website, get an SQL dump of the database and create a new one from the dump file with this library.

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.



```
SELECT * FRO
```

	id	first_name	last_name	career_start_year	career
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	
			Anthony	2003	
			Anthony	2007	
17	18	Gilbert	Arenas	2001	
18	19	Trevor	Ariza	2004	
19	20	Hilton	Armstrong	2006	
20	21	Ron	Artest	1999	

Records fetched: 67

67 rows fetched (0,19 sec)

Database: nba at localhost

Database Explorer

The Database Explorer 34 occupies the left side of SQLite Code Factory main window. It represents all objects of the connected database including system objects 26.

The explorer provides the fastest way to reach object SQL definitions.

See also: Filtering explorer content 36

Editors and Viewers

According to the MDI style implementation the SQLite Code Factory 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 18, Tabbed MDI 17

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.

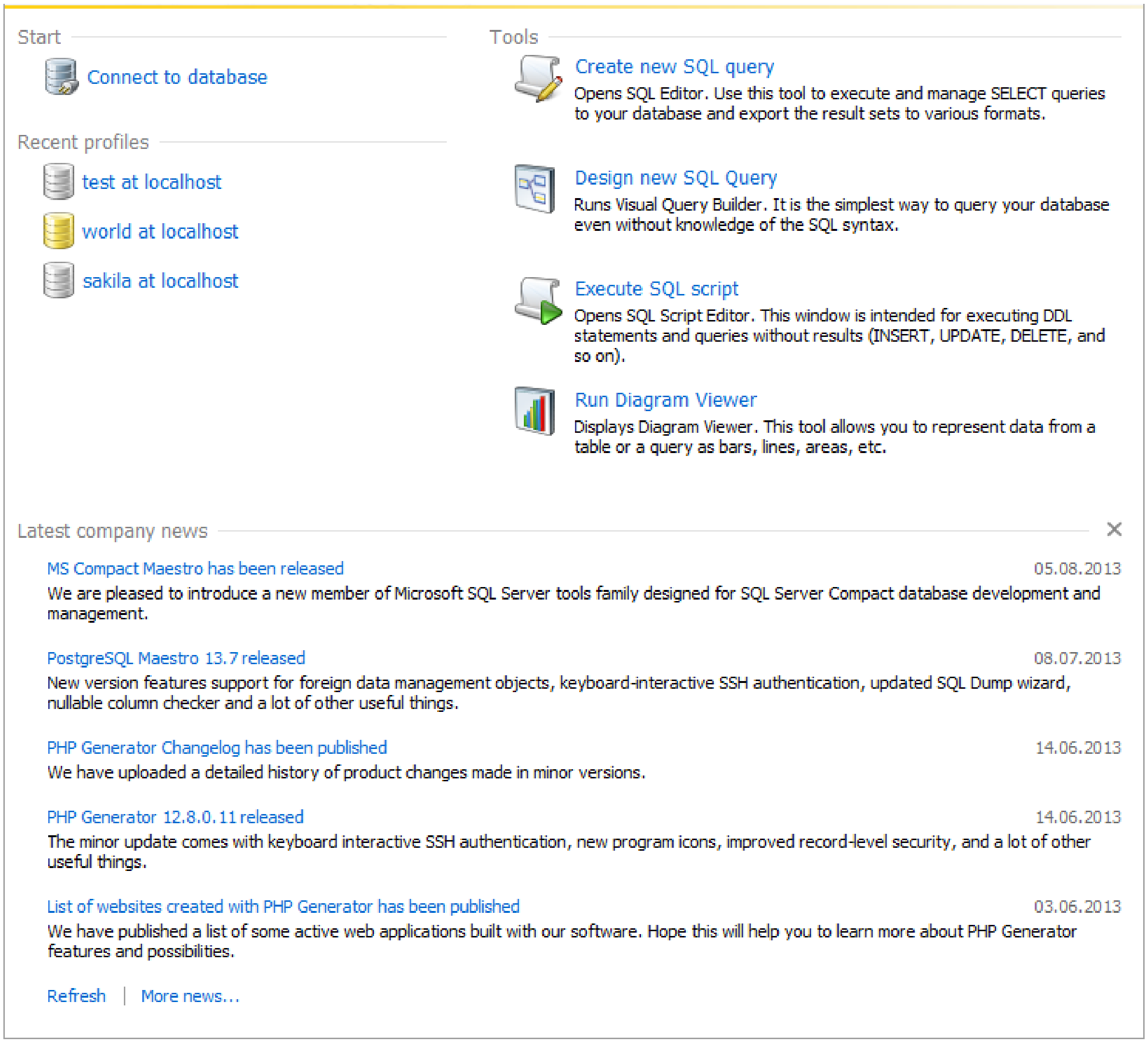
See also: Shortcut keys 20

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 SQLite Code Factory looks when you run it for the first time. The Connect to database link allows you to start working with existing databases. Follow the link to open Create database profile 23



Start

Connect to database

Recent profiles

test at localhost

world at localhost

sakila at localhost

Tools

Create new SQL query
Opens SQL Editor. Use this tool to execute and manage SELECT queries to your database and export the result sets to various formats.

Design new SQL Query
Runs Visual Query Builder. It is the simplest way to query your database even without knowledge of the SQL syntax.

Execute SQL script
Opens SQL Script Editor. This window is intended for executing DDL statements and queries without results (INSERT, UPDATE, DELETE, and so on).

Run Diagram Viewer
Displays Diagram Viewer. This tool allows you to represent data from a table or a query as bars, lines, areas, etc.

Latest company news

MS Compact Maestro has been released 05.08.2013
We are pleased to introduce a new member of Microsoft SQL Server tools family designed for SQL Server Compact database development and management.

PostgreSQL Maestro 13.7 released 08.07.2013
New version features support for foreign data management objects, keyboard-interactive SSH authentication, updated SQL Dump wizard, nullable column checker and a lot of other useful things.

PHP Generator Changelog has been published 14.06.2013
We have uploaded a detailed history of product changes made in minor versions.

PHP Generator 12.8.0.11 released 14.06.2013
The minor update comes with keyboard interactive SSH authentication, new program icons, improved record-level security, and a lot of other useful things.

List of websites created with PHP Generator has been published 03.06.2013
We have published a list of some active web applications built with our software. Hope this will help you to learn more about PHP Generator features and possibilities.

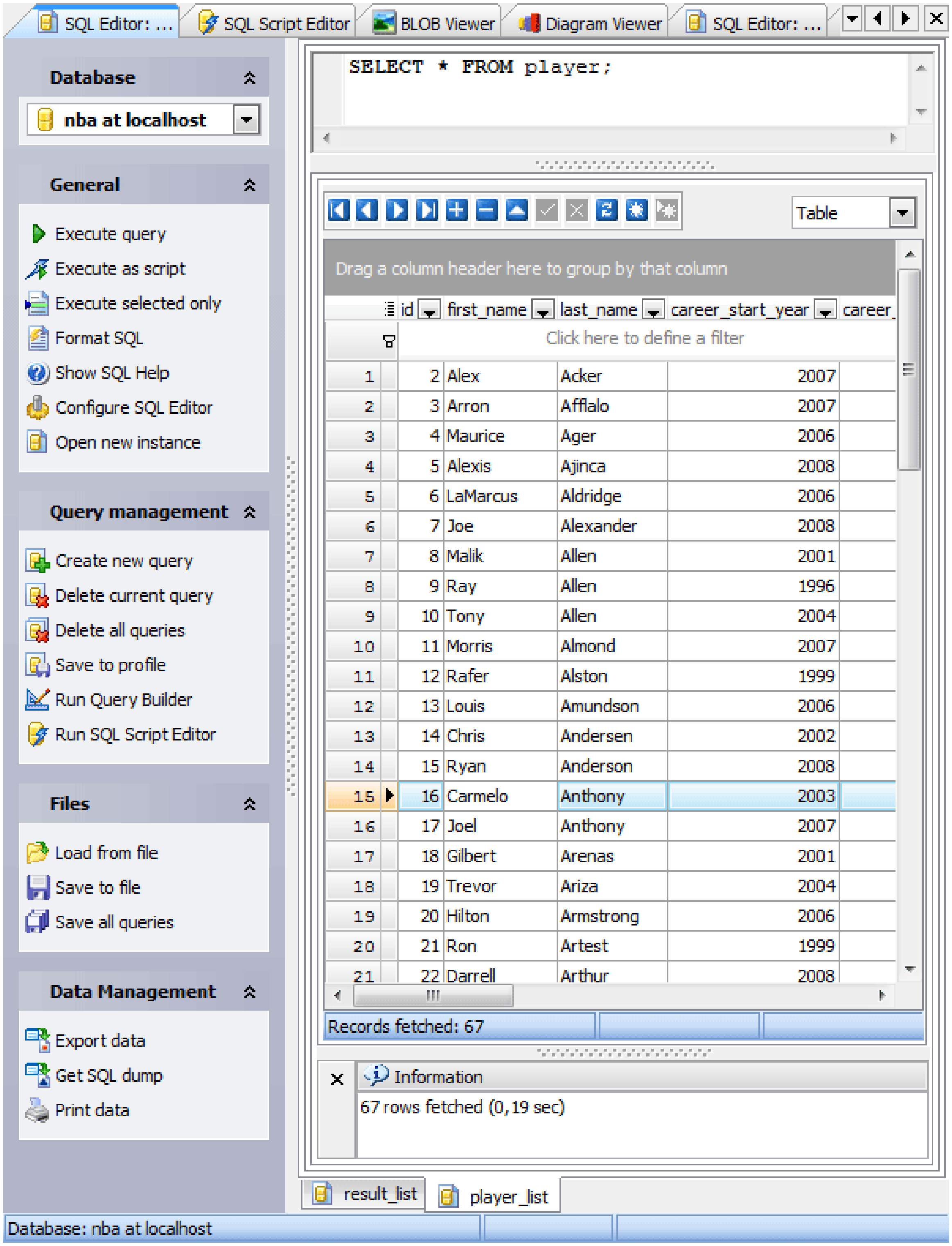
Refresh | More news...

2.3.2 Tabbed MDI

SQLite Code Factory 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.



```
SELECT * FROM player;
```

	id	first_name	last_name	career_start_year	career_
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	

Records fetched: 67

67 rows fetched (0,19 sec)

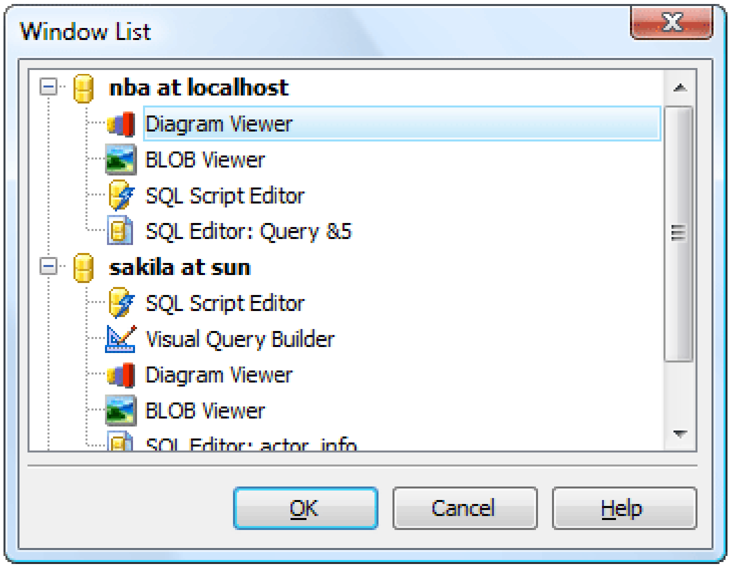
Database: nba at localhost

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+0 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 SQLite Code Factory.

Interface	
Window list	**Alt+0**
Previous Window	**F6**
Next Window	**Ctrl+F6**
Show Database Explorer	**F11**
Refresh	**F5**
Exit	**Alt+F4**
SQLite Code Factory 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	
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	
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

SQLite Code Factory 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 SQLite Code Factory startup, login prompt before connection, etc.). To start working with databases in SQLite Code Factory, you should create database profile(s) first.

Use the following links for details:

How can I create new database profiles?

In SQLite Code Factory database profiles are created within Create Database Profiles Wizard 23. 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 25 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

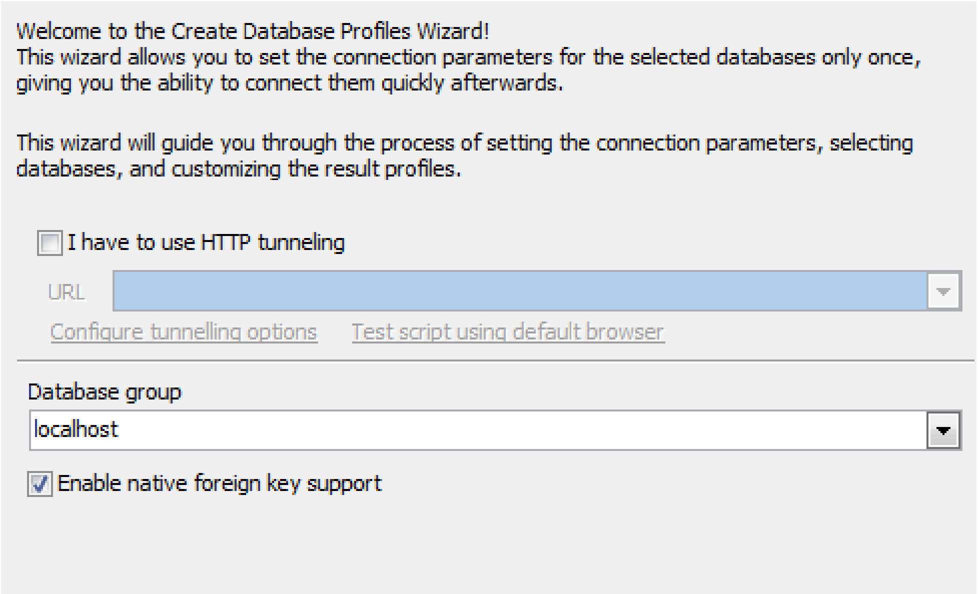
Create Database Profiles Wizard allows you to create a single database profile or several profiles from one host. To run the wizard, select the Database | Create Database Profiles... main menu item, or press the **Shift+Ctrl+P** hot keys combination. You can also use the Create Database Profiles button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

- Set connection properties 23
- Specify database profile options 23

See also: Edit Database Profile Dialog 25

3.1.1 Setting connection properties

Specify SQLite connection properties to be used on further connections.



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 have to use HTTP tunneling

URL

Configure tunnelling options
Test script using default browser

Database group
localhost

☑ Enable native foreign key support

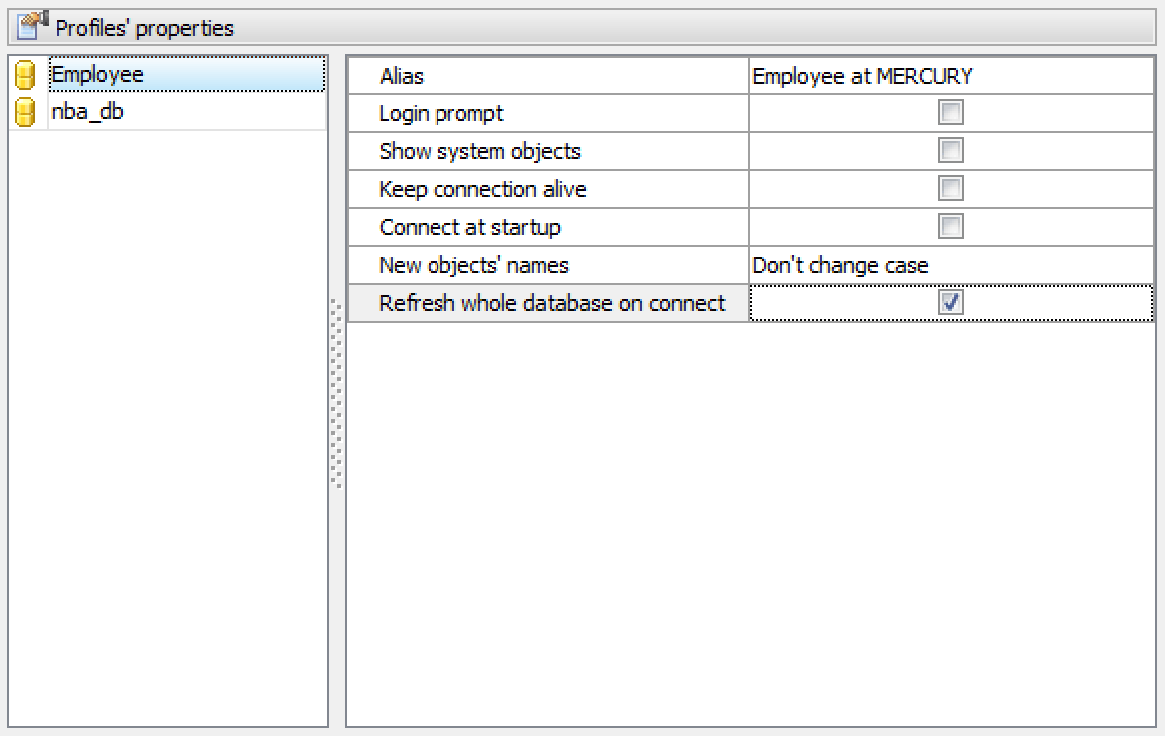
☑ Check the Create a single profile option to set the database name manually and create a single profile for this database.

☑ Hide already registered databases
Check the box to shorten the databases list on the next wizard step.

☑ Enable native foreign key support
This option affects the automatic creation of referential integrity triggers: they are NOT created if it is turned ON and vice versa. Foreign keys are supported by SQLite 3.6.19 and higher.

3.1.2 Setting profile options

To create a new profile, specify a path to the database (for local and shared files) or specify a path relative to the directory storing the *sqlite_tunnel.php* script (for HTTP tunnel connections).



Alias	Employee at MERCURY
Login prompt	☐
Show system objects	☐
Keep connection alive	☐
Connect at startup	☐
New objects' names	Don't change case
Refresh whole database on connect	☑

☑ Show system objects
Check the box to make system objects visible. For example, use this option to see the "shadow" FTS tables.

☑ 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 SQLite Code Factory.

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 25
- Settings database options 26
- Setting default directories for database tools 28
- Editing obligatory scripts to execute 28
- Setting log options and file names 30

See also: Create Database Profile Wizard 23

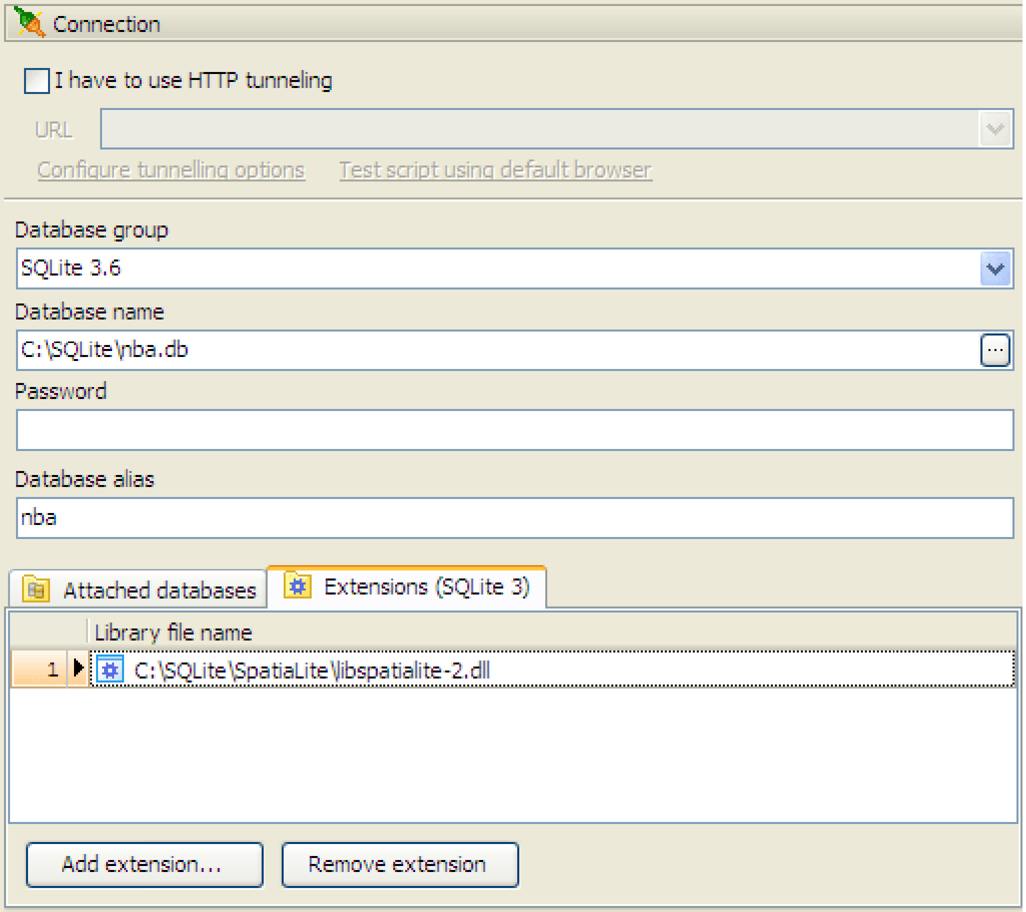
3.2.1 Editing connection properties

The tab allows you to change connection properties 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.

☑ Enable native foreign key support
This option affects the automatic creation of referential integrity triggers: they are NOT created if it is turned ON and vice versa. Foreign keys are supported by SQLite 3.6.19 and higher.

Attached databases
The list of database files to be automatically added to each connection established by the software. Find out more on attached databases at SQLite official documentation.

Extensions (SQLite 3)
The list of extensions to be loaded automatically for each connection established by the software.


Connection
☐ I have to use HTTP tunneling
URL
Configure tunnelling options
Test script using default browser

Database group
SQLite 3.6

Database name
C:\SQLite\nba.db

Password

Database alias
nba

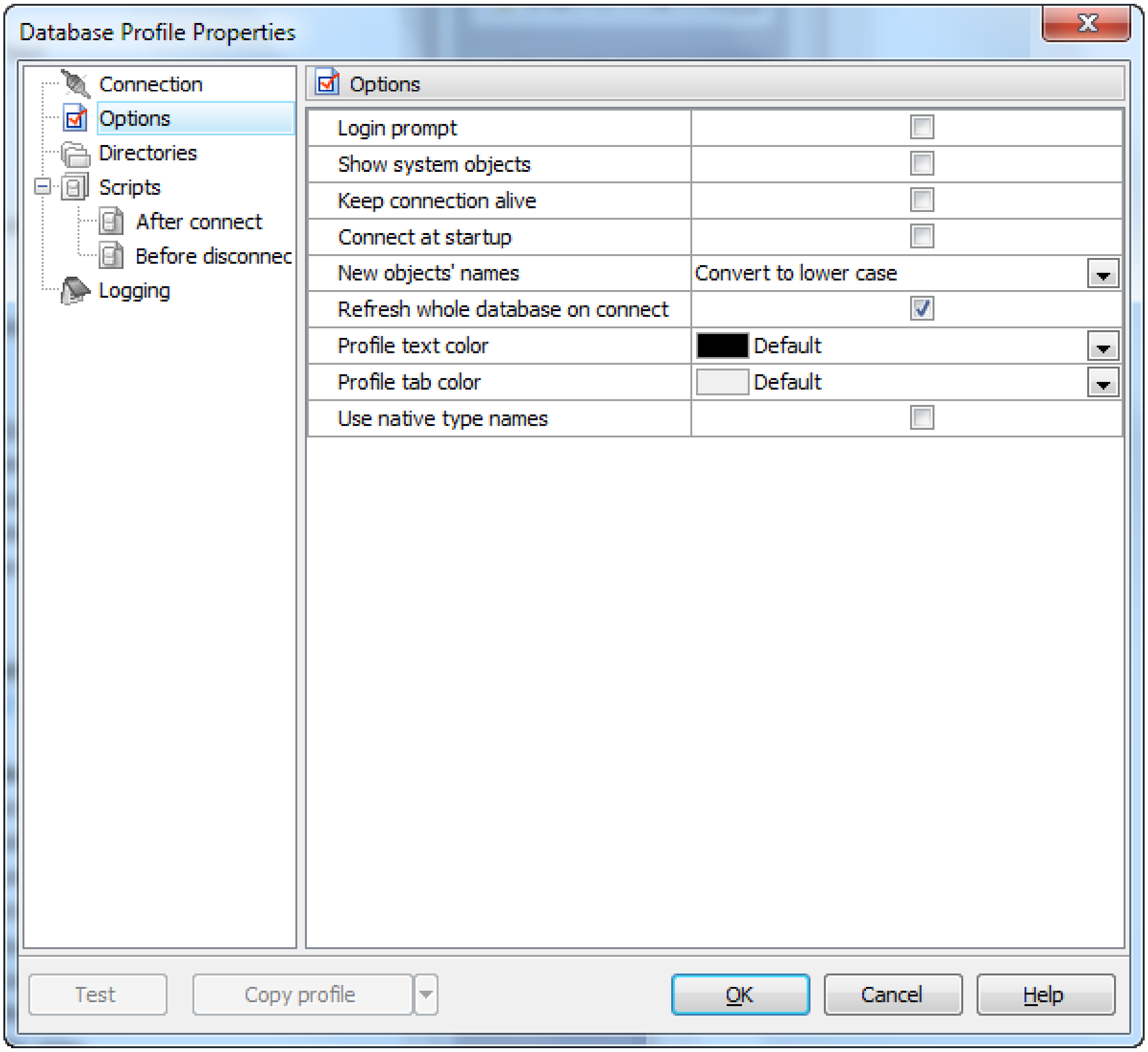
Attached databases
Extensions (SQLite 3)

	Library file name
1	C:\SQLite\SpatiaLite\libspatialite-2.dll

Add extension...
Remove extension

3.2.2 Setting profile options

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



Login prompt	☐
Show system objects	☐
Keep connection alive	☐
Connect at startup	☐
New objects' names	Convert to lower case
Refresh whole database on connect	☑
Profile text color	■ Default
Profile tab color	□ Default
Use native type names	☐

☑ Show system objects

Check the option to make system objects visible.

☑ 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.

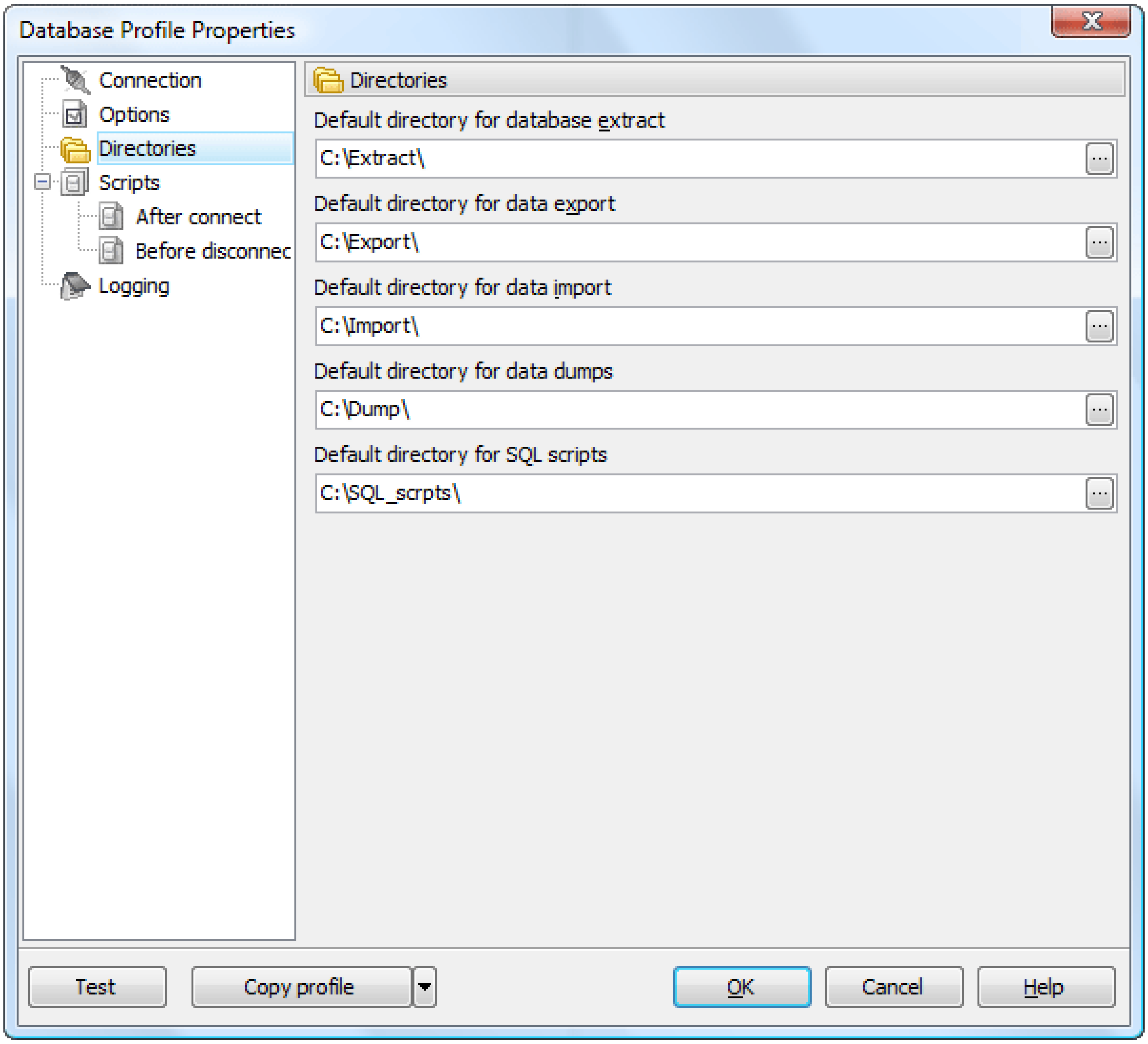
☑ Refresh whole database on connect

Use the option along with the Show empty schemas 113 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.

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.



Default directory for database extract
C:\Extract\

Default directory for data export
C:\Export\

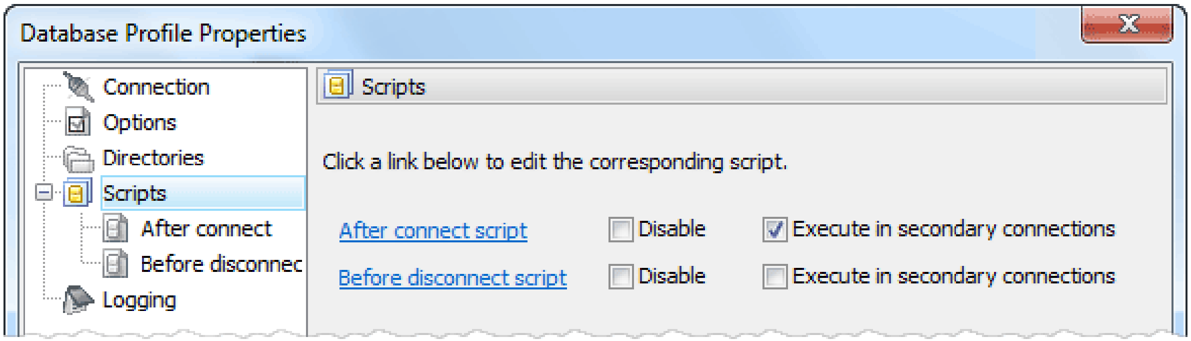
Default directory for data import
C:\Import\

Default directory for data dumps
C:\Dump\

Default directory for SQL scripts
C:\SQL_scrpts\

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.

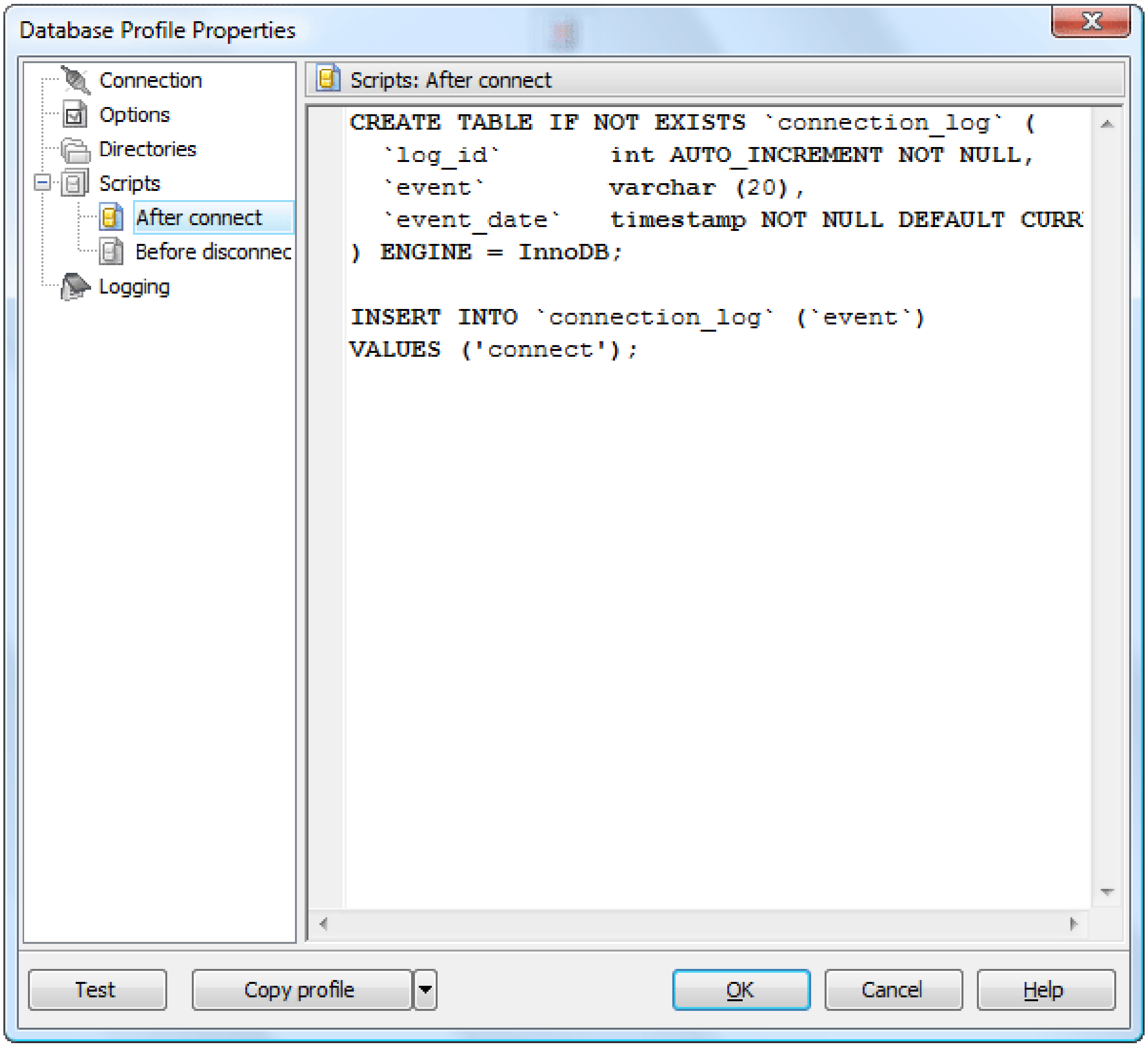


Click a link below to edit the corresponding script.

After connect script ☐ Disable ☑ Execute in secondary connections

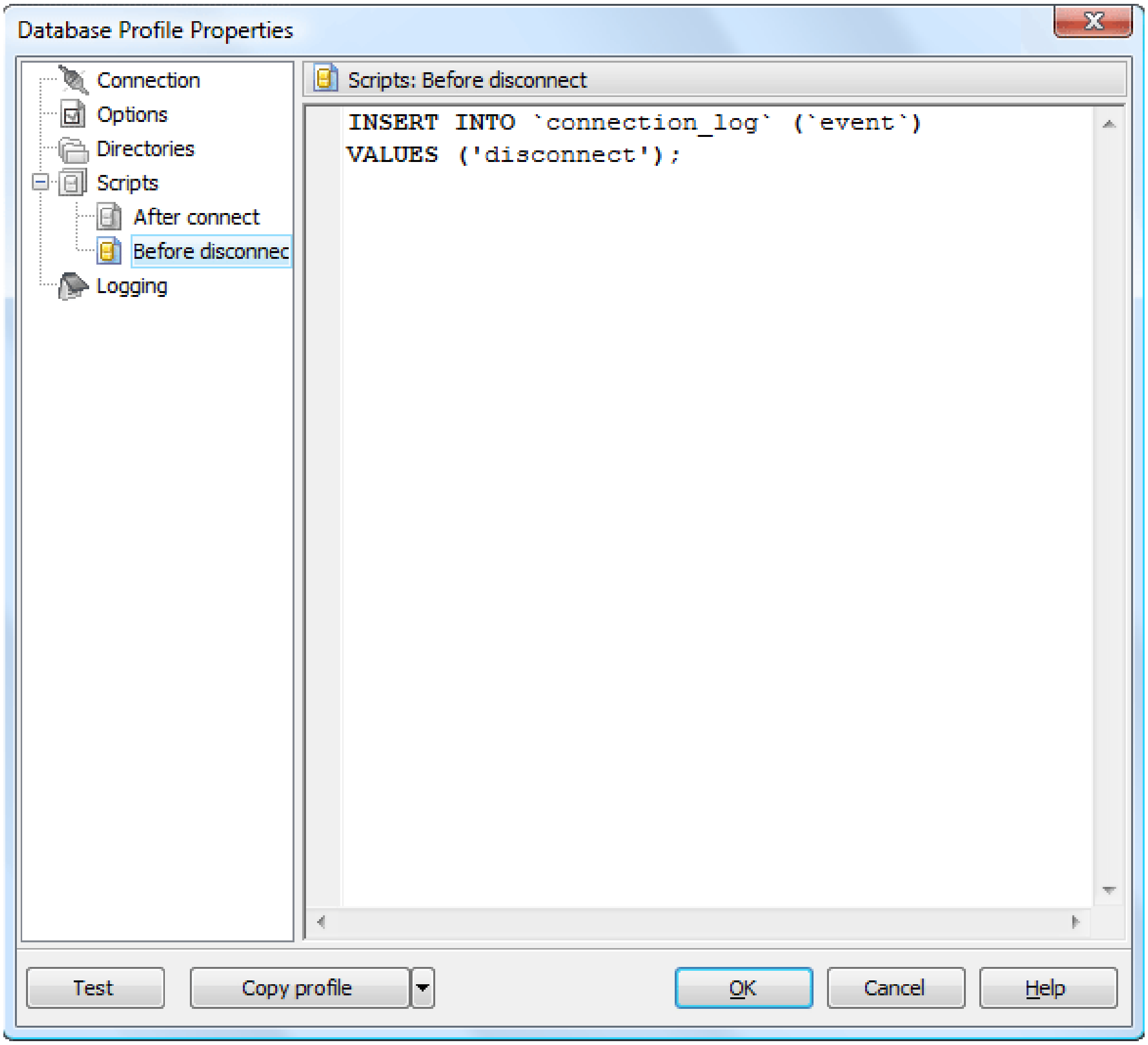
Before disconnect script ☐ Disable ☐ Execute in secondary connections

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



```
CREATE TABLE IF NOT EXISTS `connection_log` (
  `log_id`       int AUTO_INCREMENT NOT NULL,
  `event`        varchar (20),
  `event_date`   timestamp NOT NULL DEFAULT CURR
) ENGINE = InnoDB;

INSERT INTO `connection_log` (`event`)
VALUES ('connect');
```

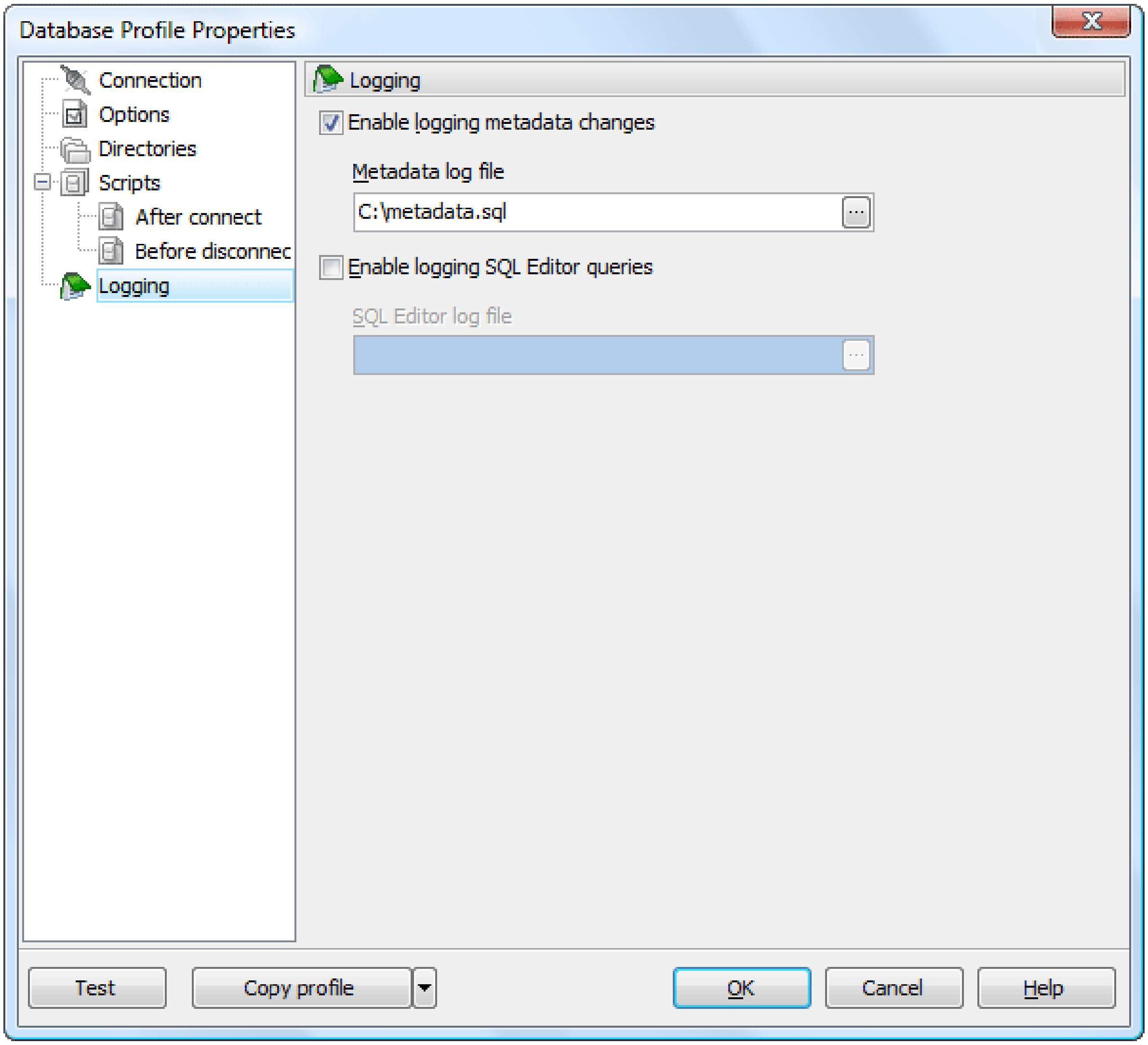
The next screen represents the example of an obligatory script to execute before SQLite Code Factory will disconnect from the database. The script writes a disconnect time to the log table.



```
INSERT INTO `connection_log` (`event`)
VALUES ('disconnect');
```

3.2.5 Setting log options

Enable/disable metadata changes logging and SQL query logging and specify the corresponding log file names if necessary.



Logging

☑ Enable logging metadata changes

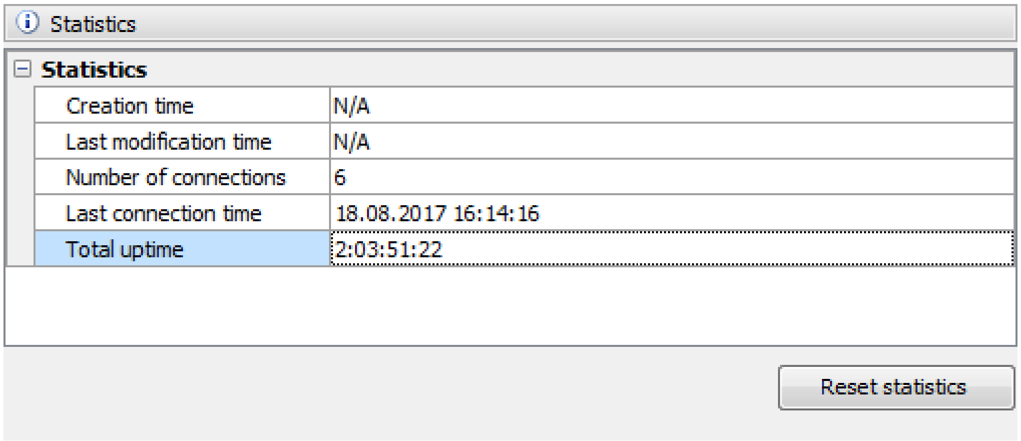
Metadata log file: C:\metadata.sql

☐ Enable logging SQL Editor queries

SQL Editor log file

3.2.6 Statisitcs

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



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

4 Browse Objects

SQLite Code Factory allows to browse objects stored in a Remote Server database within Database Explorer. It represents objects grouped by kind and listed under the according SQLite database node, provided with subobjects if exist. It's possible to reduce 36 the number of represented objects in the explorer tree and also to hide/display 113 table subobjects, represent system objects in different color, etc.

4.1 Database Explorer

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

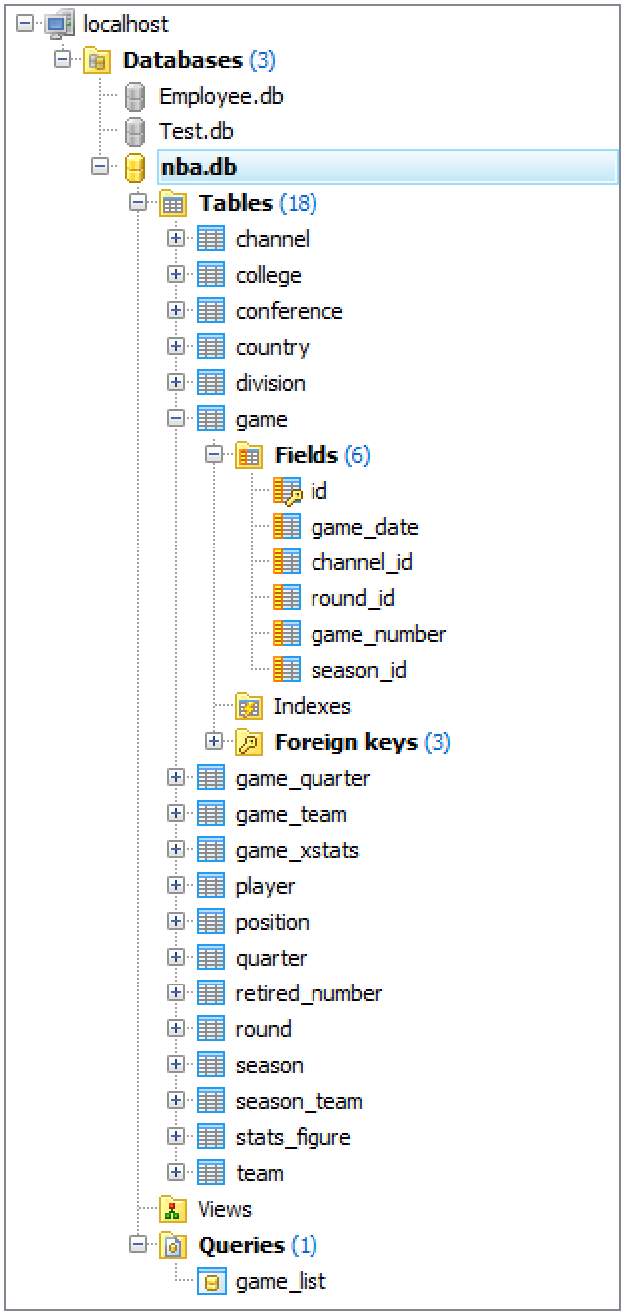
To start working with a database you need to create its profile first. The conception of database profiles gives you an opportunity to connect to databases in one touch and work with the selected databases only.

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 113 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? 35
- How can I connect to a database? 34
- How can I disconnect from a database? 36



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 23 (the Create Database Profiles... item);
- rename currently selected database profile (the Rename Database Profile... item);
- edit currently selected database profile 25 (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).

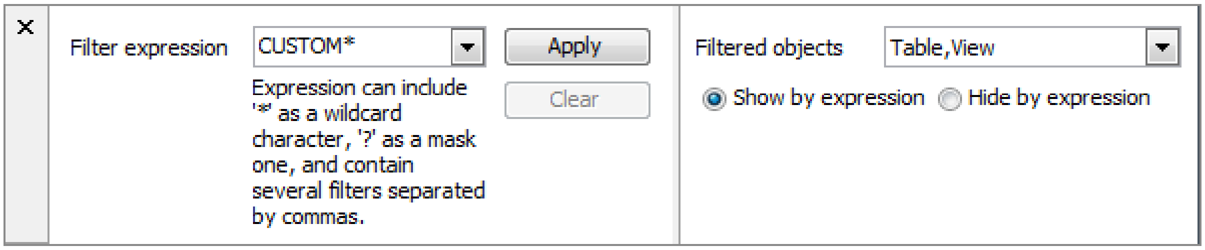
Can I copy a database object from one database to another?
Database Explorer provides you with an ability of copying database objects from one database to another. To perform this operation, you should connect to both the source and the destination databases first. After the connection is established, 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).

4.1.1 Filtering explorer content

SQLite Code Factory 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.



Expression can include '*' as a wildcard character, '?' as a mask one, and contain several filters separated by commas.

- 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.

4.2 Filter Builder Dialog

Filter Builder Dialog 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 Queries and Scripts

SQLite Code Factory provides several tools for working with SQL queries and scripts:

- SQL Editor 41 for editing the query text directly and executing SELECT queries;
- Visual Query Builder 46 for building SELECT, INSERT, UPDATE and DELETE queries visually.

Both SQL Editor and Visual Query Builder supports parameters in queries 45

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, 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.

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.

5.1 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 39.

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 been 114 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 43.

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 42 is also at your disposal.

SQL Editor allows you to use Visual Query Builder 46 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

SQLite Code Factory 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 131.

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 TRANSACTION* 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).

PRAGMA command executing

Using SQL Editor you can execute the PRAGMA command, that is not a part of SQL standard. The PRAGMA command is a special command used to modify the operation of the SQLite library or to query the library for internal (non-table) data. Using the PRAGMA command you can set the auto-vacuum flag in the database, change the maximum number of database disk pages that SQLite will hold in memory simultaneously, change the count-changes flag, etc.

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.

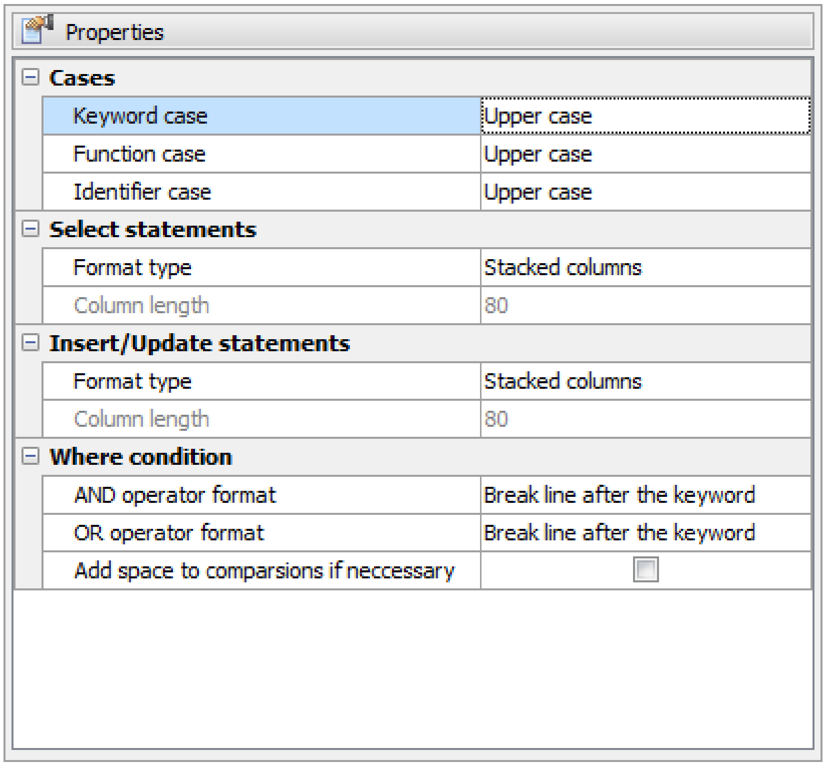
See also: Visual Query Builder 46, SQL Script Editor, SQL Editor Options 114

5.1.1 SQL Formatter

SQLite Code Factory 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.



Cases	
Keyword case	Upper case
Function case	Upper case
Identifier case	Upper case
Select statements	
Format type	Stacked columns
Column length	80
Insert/Update statements	
Format type	Stacked columns
Column length	80
Where condition	
AND operator format	Break line after the keyword
OR operator format	Break line after the keyword
Add space to comparsions if neccessary	☐

5.1.2 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.

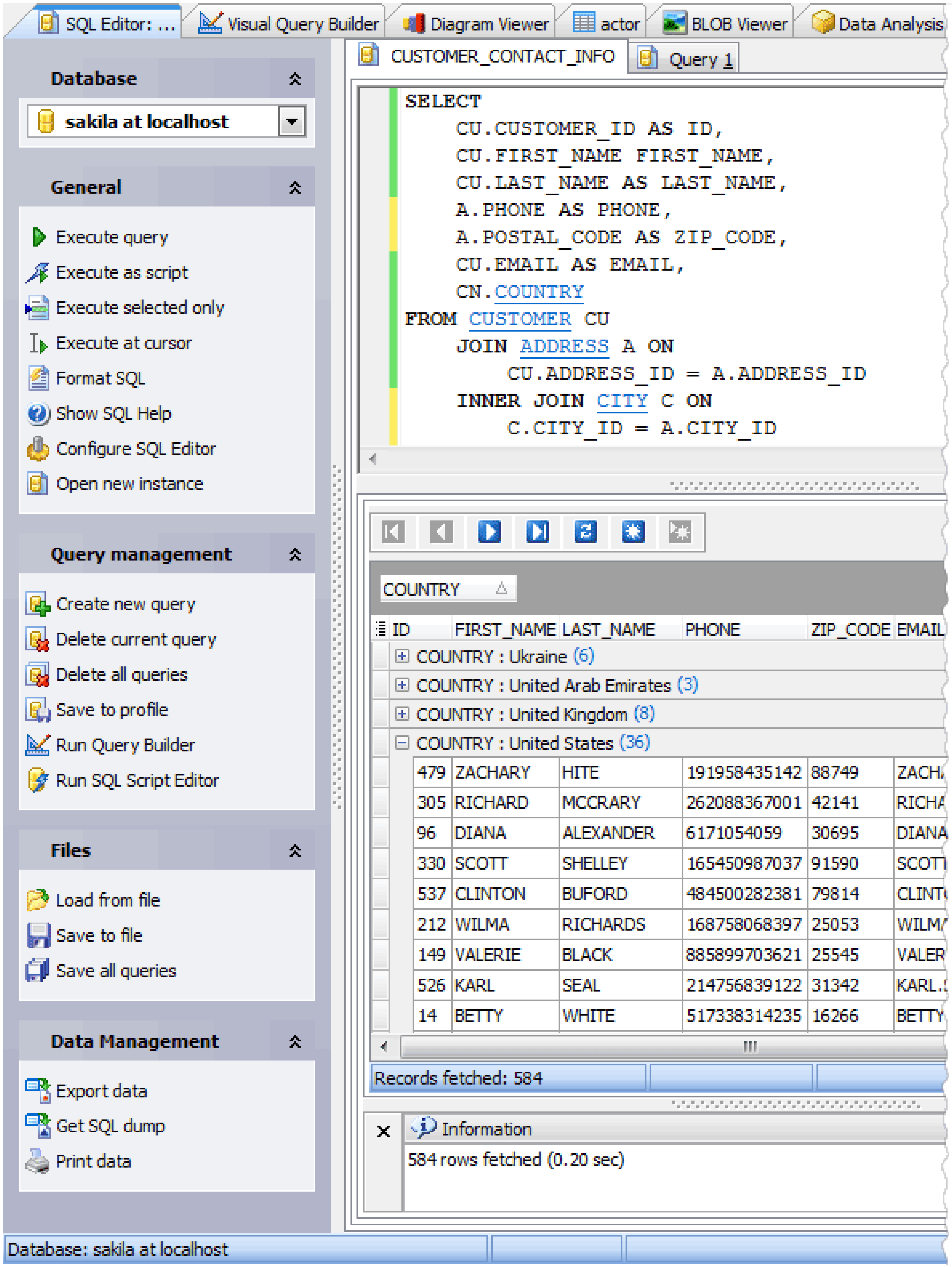
- You can also execute query as script (**Shift+F5**, **Shift+F8**, **Shift+F9**). In this case the query does not return data.

- 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 58 section.



```
SELECT
    CU.CUSTOMER_ID AS ID,
    CU.FIRST_NAME FIRST_NAME,
    CU.LAST_NAME AS LAST_NAME,
    A.PHONE AS PHONE,
    A.POSTAL_CODE AS ZIP_CODE,
    CU.EMAIL AS EMAIL,
    CN.COUNTRY
FROM CUSTOMER CU
    JOIN ADDRESS A ON
        CU.ADDRESS_ID = A.ADDRESS_ID
    INNER JOIN CITY C ON
        C.CITY_ID = A.CITY_ID
```

ID	FIRST_NAME	LAST_NAME	PHONE	ZIP_CODE	EMAIL
⊞ COUNTRY : Ukraine (6)					
⊞ COUNTRY : United Arab Emirates (3)					
⊞ COUNTRY : United Kingdom (8)					
⊟ COUNTRY : United States (36)					
479	ZACHARY	HITE	191958435142	88749	ZACH
305	RICHARD	MCCRARY	262088367001	42141	RICHA
96	DIANA	ALEXANDER	6171054059	30695	DIANA
330	SCOTT	SHELLEY	165450987037	91590	SCOT
537	CLINTON	BUFORD	484500282381	79814	CLINT
212	WILMA	RICHARDS	168758068397	25053	WILM
149	VALERIE	BLACK	885899703621	25545	VALER
526	KARL	SEAL	214756839122	31342	KARL.
14	BETTY	WHITE	517338314235	16266	BETTY

Records fetched: 584

Information

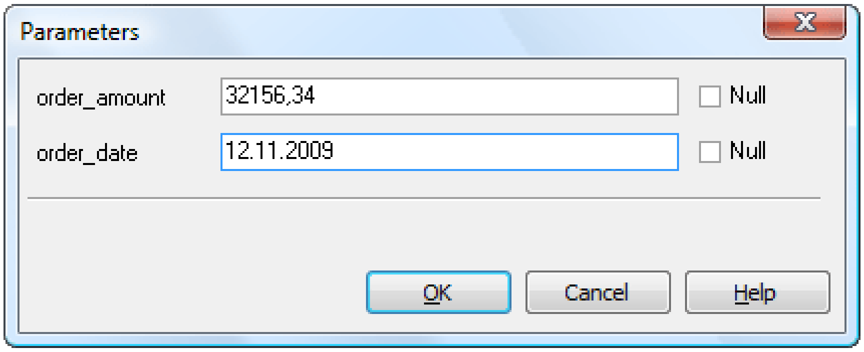
584 rows fetched (0.20 sec)

Database: sakila at localhost

5.1.3 Query Parameters

Both SQL Editor 41 and Visual Query Builder 46 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.



order_amount 32156,34 ☐ Null

order_date 12.11.2009 ☐ Null

Note: To allow use parameters in query text, check the corresponding option at the Tools 112 tab of SQLite Code Factory Options.

5.2 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.

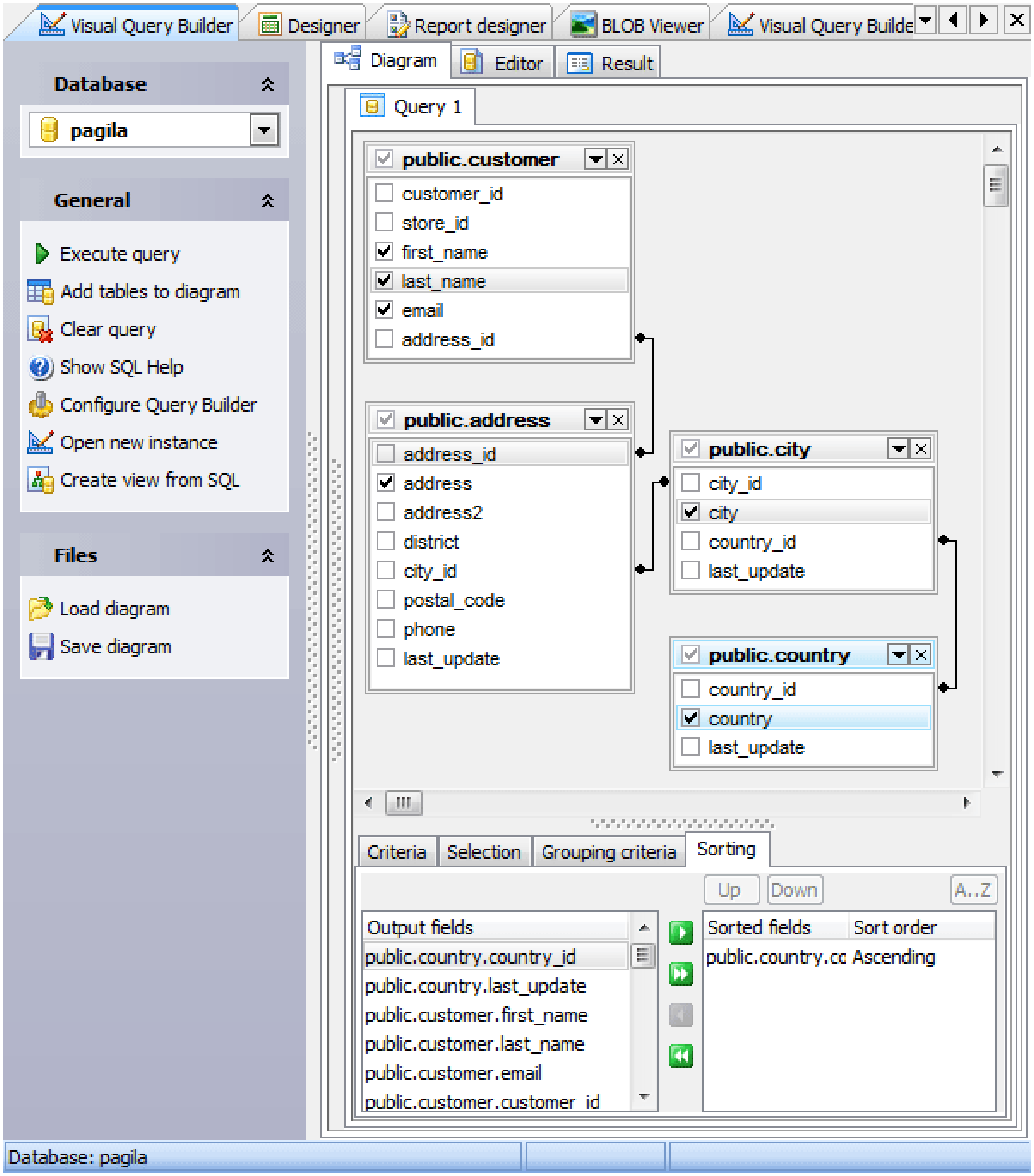
The most popular query management actions (creating, editing, deleting) are covered by the corresponding topic 39.

Builder consists of 3 tabs:

- Diagram 47 - to create a query from a graphical interface,
- Editor 52 – to modify the query text before its executing,
- Result 53 (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.

See also: SQL Editor 41, Visual Query Builder Options 116, Query Parameters 45



5.2.1 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*).

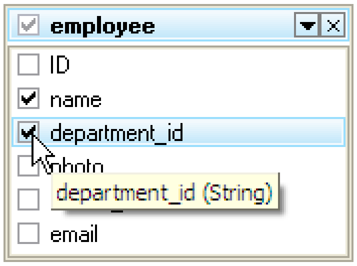
Add required tables to the Diagram area.

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

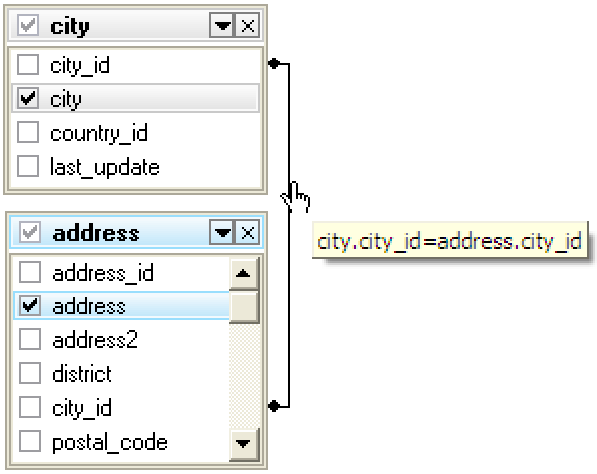
To include a table field to the query, tick off the option box to the left of the field name in the list or double-click it to see the blue icon next to the field name.



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.

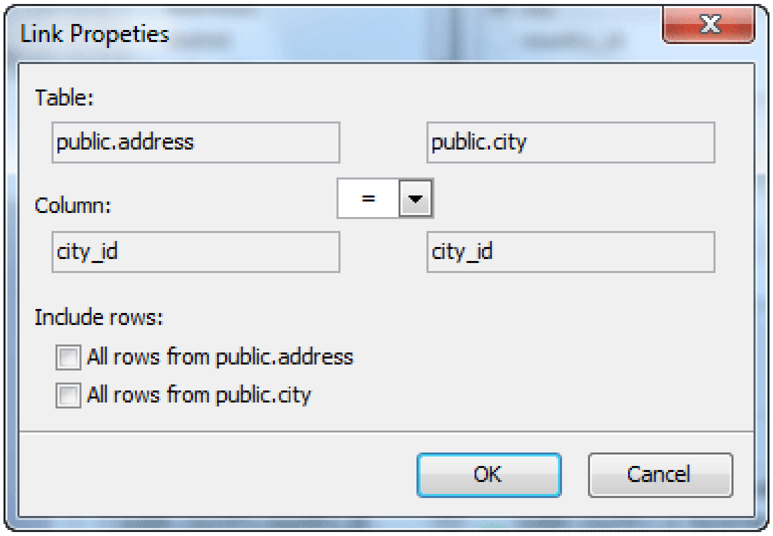


Table:
public.address
public.city
Column: =
city_id
city_id
Include rows:
☐ All rows from public.address
☐ All rows from public.city

To remove a link between objects, select the Delete Link item from the popup menu.

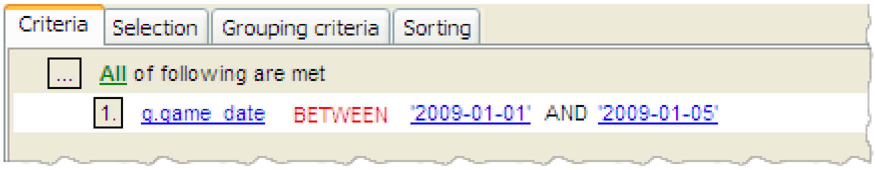
To delete all the links of an object, click the '-' button next to the object alias. To insert a point to the link line, select the Insert Point item from the popup menu, and the new point will appear. Using the point you can move the link line. It does not cause any changes in the query but makes the 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.



... All of following are met

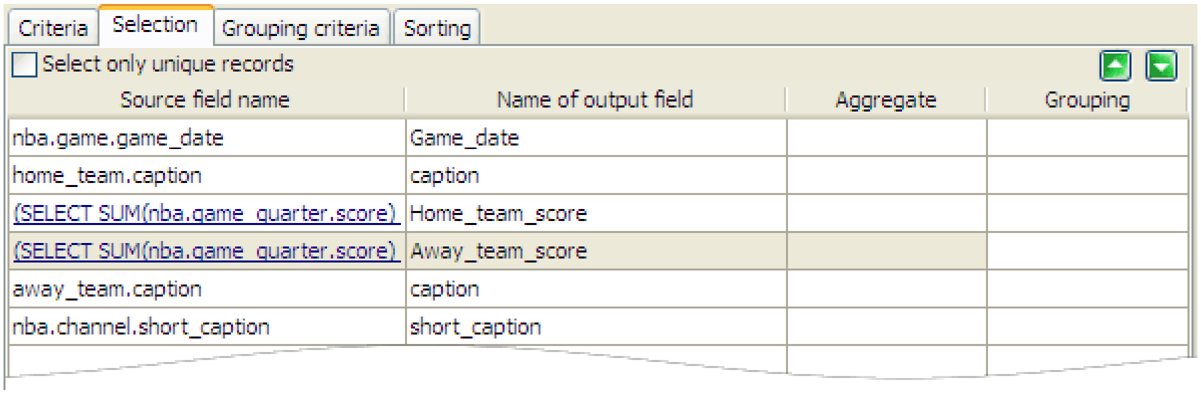
1. g.game_date BETWEEN '2009-01-01' AND '2009-01-05'

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.



☐ 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.

AVG	Returns the average of the values in a group
BIT_AND	Returns the bitwise AND of all bits in the expression.
BIT_OR	Returns the bitwise OR of all bits in the expression.
COUNT	Returns the total number of items in a column. This function does not ignore NULL values when calculating results.
GROUP_CONCAT	Returns a string result with the concatenated non-NULL values from a group.
MAX	Returns the maximum value for the column.
MIN	Returns the minimum value for the column.
STD	Returns the population standard deviation of the expression.
STDDEV	Returns the sample standard deviation of a numeric expression evaluated over a set.
SUM	Returns the sum of all the values in the expression.
VARIANCE	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 filed in the right list and switch the direction (*Ascending, Descending*) using the A..Z/Z..A button.

Create UNIONs

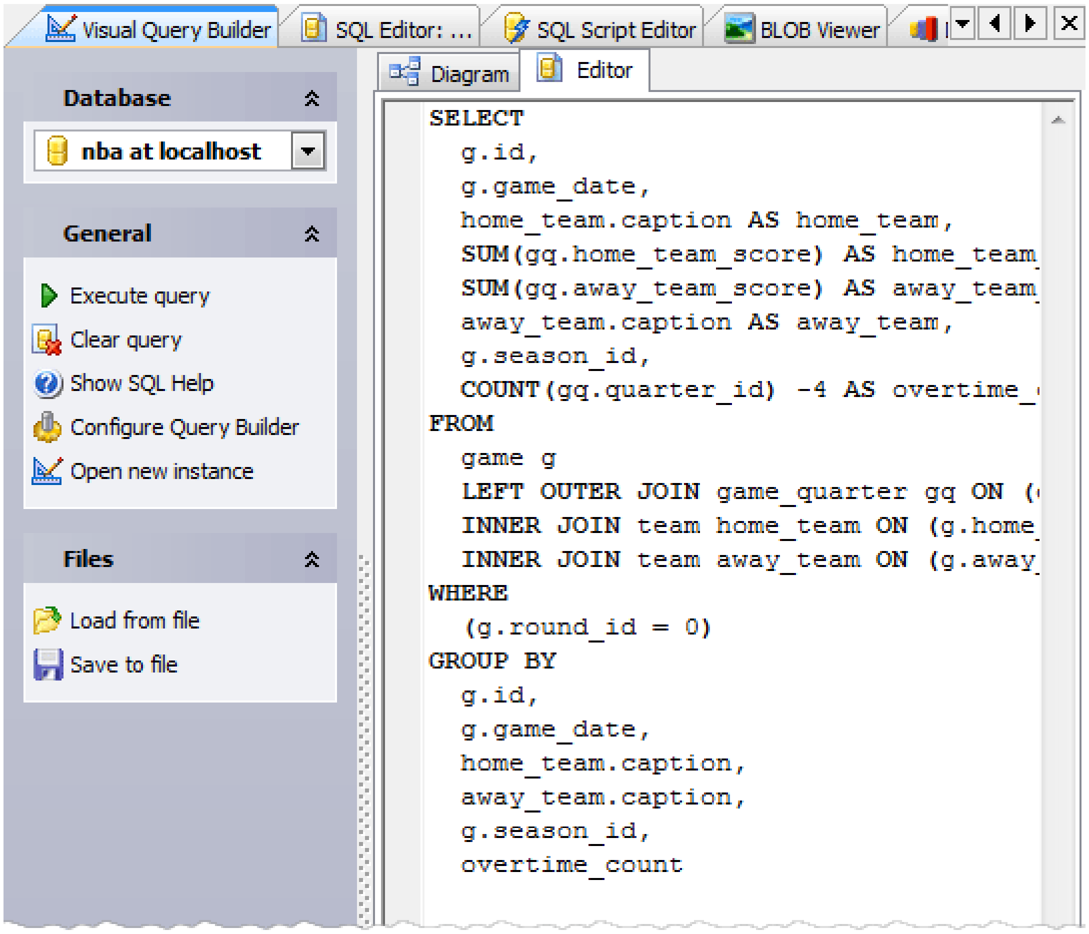
To combine the result from multiple SELECT statements into a single result set, use the Add union link of the Query Explorer popup menu.

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.

5.2.2 Working with editor area

In the Editor area the query text is automatically generated while you are building the query.

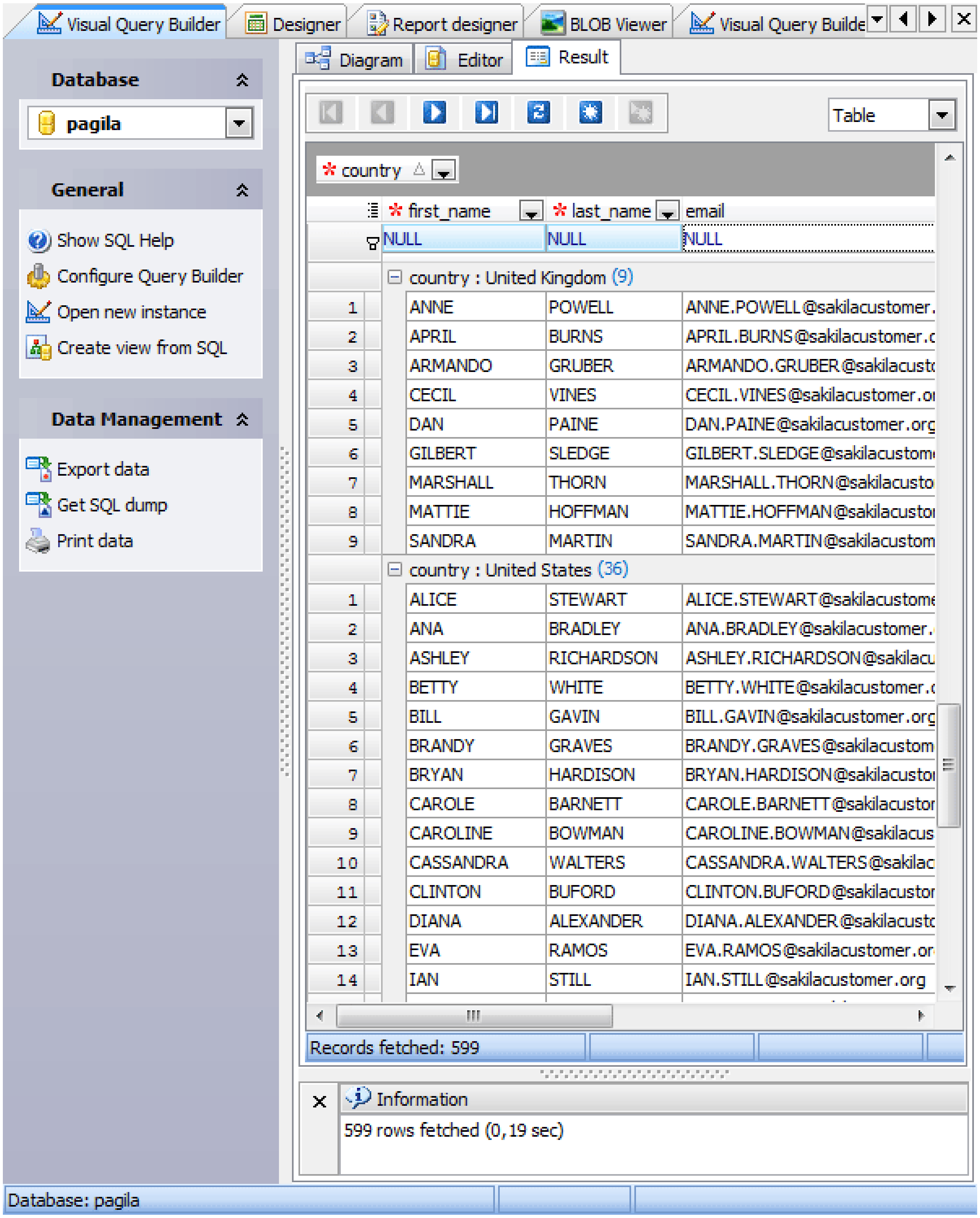
You can edit this text according to the rules of SQL, and all the changes will be displayed on the Diagram page of Visual Query Builder.



```
SELECT
  g.id,
  g.game_date,
  home_team.caption AS home_team,
  SUM(gq.home_team_score) AS home_team_
  SUM(gq.away_team_score) AS away_team_
  away_team.caption AS away_team,
  g.season_id,
  COUNT(gq.quarter_id) -4 AS overtime_
FROM
  game g
  LEFT OUTER JOIN game_quarter gq ON (
  INNER JOIN team home_team ON (g.home_
  INNER JOIN team away_team ON (g.away_
WHERE
  (g.round_id = 0)
GROUP BY
  g.id,
  g.game_date,
  home_team.caption,
  away_team.caption,
  g.season_id,
  overtime_count
```

5.2.3 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.



	first_name	last_name	email
	NULL	NULL	NULL
	country : United Kingdom (9)		
1	ANNE	POWELL	ANNE.POWELL@sakilacustomer.
2	APRIL	BURNS	APRIL.BURNS@sakilacustomer.c
3	ARMANDO	GRUBER	ARMANDO.GRUBER@sakilacust
4	CECIL	VINES	CECIL.VINES@sakilacustomer.o
5	DAN	PAINE	DAN.PAINE@sakilacustomer.org
6	GILBERT	SLEDGE	GILBERT.SLEDGE@sakilacustom
7	MARSHALL	THORN	MARSHALL.THORN@sakilacusto
8	MATTIE	HOFFMAN	MATTIE.HOFFMAN@sakilacusto
9	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustom
	country : United States (36)		
1	ALICE	STEWART	ALICE.STEWART@sakilacustome
2	ANA	BRADLEY	ANA.BRADLEY@sakilacustomer.
3	ASHLEY	RICHARDSON	ASHLEY.RICHARDSON@sakilacu
4	BETTY	WHITE	BETTY.WHITE@sakilacustomer.
5	BILL	GAVIN	BILL.GAVIN@sakilacustomer.org
6	BRANDY	GRAVES	BRANDY.GRAVES@sakilacustom
7	BRYAN	HARDISON	BRYAN.HARDISON@sakilacusto
8	CAROLE	BARNETT	CAROLE.BARNETT@sakilacusto
9	CAROLINE	BOWMAN	CAROLINE.BOWMAN@sakilacus
10	CASSANDRA	WALTERS	CASSANDRA.WALTERS@sakilac
11	CLINTON	BUFORD	CLINTON.BUFORD@sakilacusto
12	DIANA	ALEXANDER	DIANA.ALEXANDER@sakilacusto
13	EVA	RAMOS	EVA.RAMOS@sakilacustomer.or
14	IAN	STILL	IAN.STILL@sakilacustomer.org

Records fetched: 599

599 rows fetched (0,19 sec)

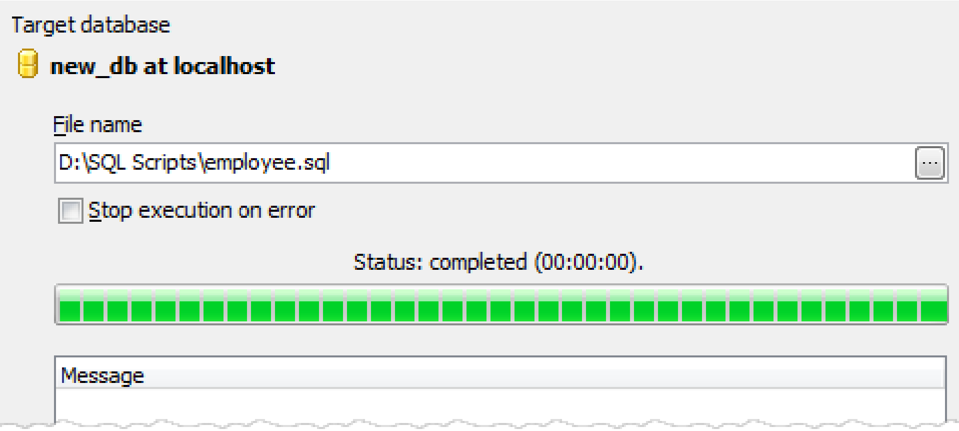
Database: pagila

5.3 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 56.

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.



Target database

new_db at localhost

File name

D:\SQL Scripts\employee.sql

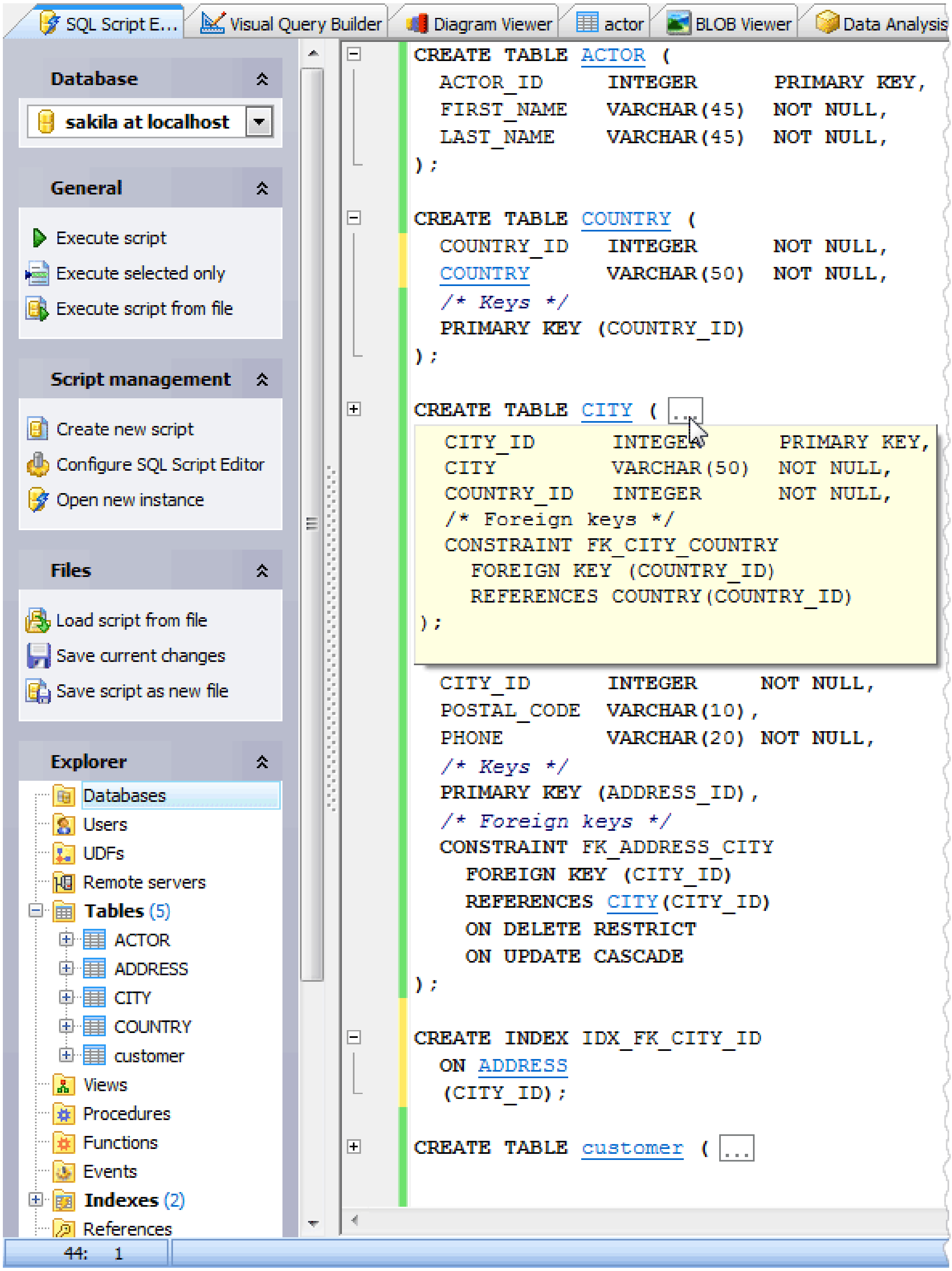
☐ Stop execution on error

Status: completed (00:00:00).

5.4 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 41. 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.



```
CREATE TABLE ACTOR (
  ACTOR_ID     INTEGER      PRIMARY KEY,
  FIRST_NAME   VARCHAR(45)  NOT NULL,
  LAST_NAME    VARCHAR(45)  NOT NULL,
);

CREATE TABLE COUNTRY (
  COUNTRY_ID   INTEGER      NOT NULL,
  COUNTRY      VARCHAR(50)  NOT NULL,
  /* Keys */
  PRIMARY KEY (COUNTRY_ID)
);

CREATE TABLE CITY ( ...
   CITY_ID       INTEGER      PRIMARY KEY,
   CITY          VARCHAR(50)  NOT NULL,
   COUNTRY_ID    INTEGER      NOT NULL,
   /* Foreign keys */
   CONSTRAINT FK_CITY_COUNTRY
     FOREIGN KEY (COUNTRY_ID)
     REFERENCES COUNTRY(COUNTRY_ID)
 );

  CITY_ID      INTEGER     NOT NULL,
  POSTAL_CODE  VARCHAR(10),
  PHONE        VARCHAR(20) NOT NULL,
  /* Keys */
  PRIMARY KEY (ADDRESS_ID),
  /* Foreign keys */
  CONSTRAINT FK_ADDRESS_CITY
    FOREIGN KEY (CITY_ID)
    REFERENCES CITY(CITY_ID)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);

CREATE INDEX IDX_FK_CITY_ID
  ON ADDRESS
  (CITY_ID);

CREATE TABLE customer ( ...
```

6 Data management

Query results are displayed on the Result tabs of SQL Editor 41 or Visual Query Builder 46 .

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 59 for details.

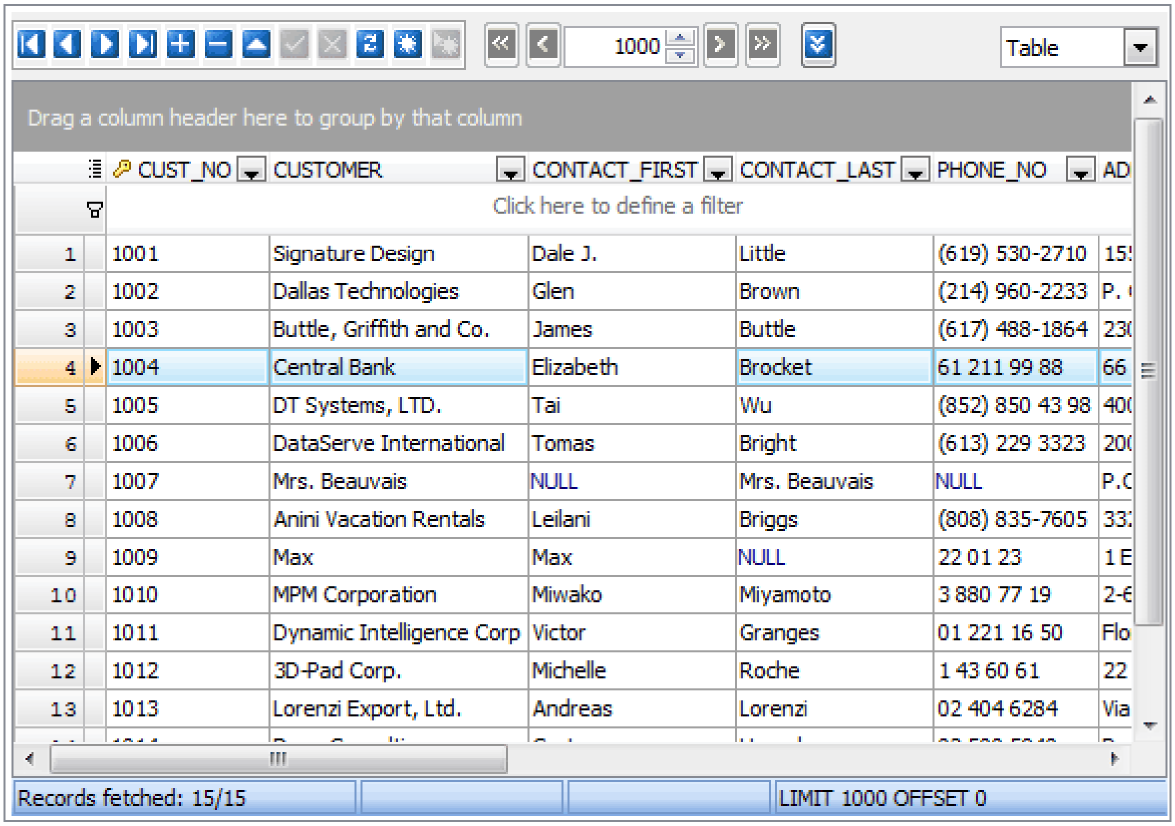
Navigation bars of these tabs as well as popup menus of their working areas places at your disposal the following functions for managing data:

- Export Data 75 allows you to export data to various formats, including MS Excel, MS Access, RTF, HTML, PDF and more.
- Get SQL Dump 82 exports data to the SQL script as a number of INSERT statements.
- Import Data 85 provides you with possibility to import data from MS Excel, MS Access, DBF, XML, TXT, and CSV.
- Edit BLOB 70 allows you to view and edit the content of BLOB and TEXT fields.

6.1 Data View

SQLite Code Factory represents all data (stored in tables and views, results of queries and procedures) in grid 60 or in info cards 65. 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.

> **Note:** For databases in UTF-8 encoding it is necessary to specify, which string fields are used to store Unicode data (available options are "Only nvarchar(xx) fields" and "All the string fields"). For databases in UTF-16 encoding no such actions are required.


1000
Table
Drag a column header here to group by that column
Click here to define a filter

	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	23
4	1004	Central Bank	Elizabeth	Brocket	61 211 99 88	66
5	1005	DT Systems, LTD.	Tai	Wu	(852) 850 43 98	40
6	1006	DataServe International	Tomas	Bright	(613) 229 3323	20
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 66 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 120.

Navigation bar
The Data management group of the Navigation bar allows to invoke Data Input Form 66, SQL Editor 41 with SELECT query, Data Export 75, and Data Import 85 modules using corresponding links, also get SQL dump 82 of the current data set and print current data with enabled preview in WYSIWYG mode.

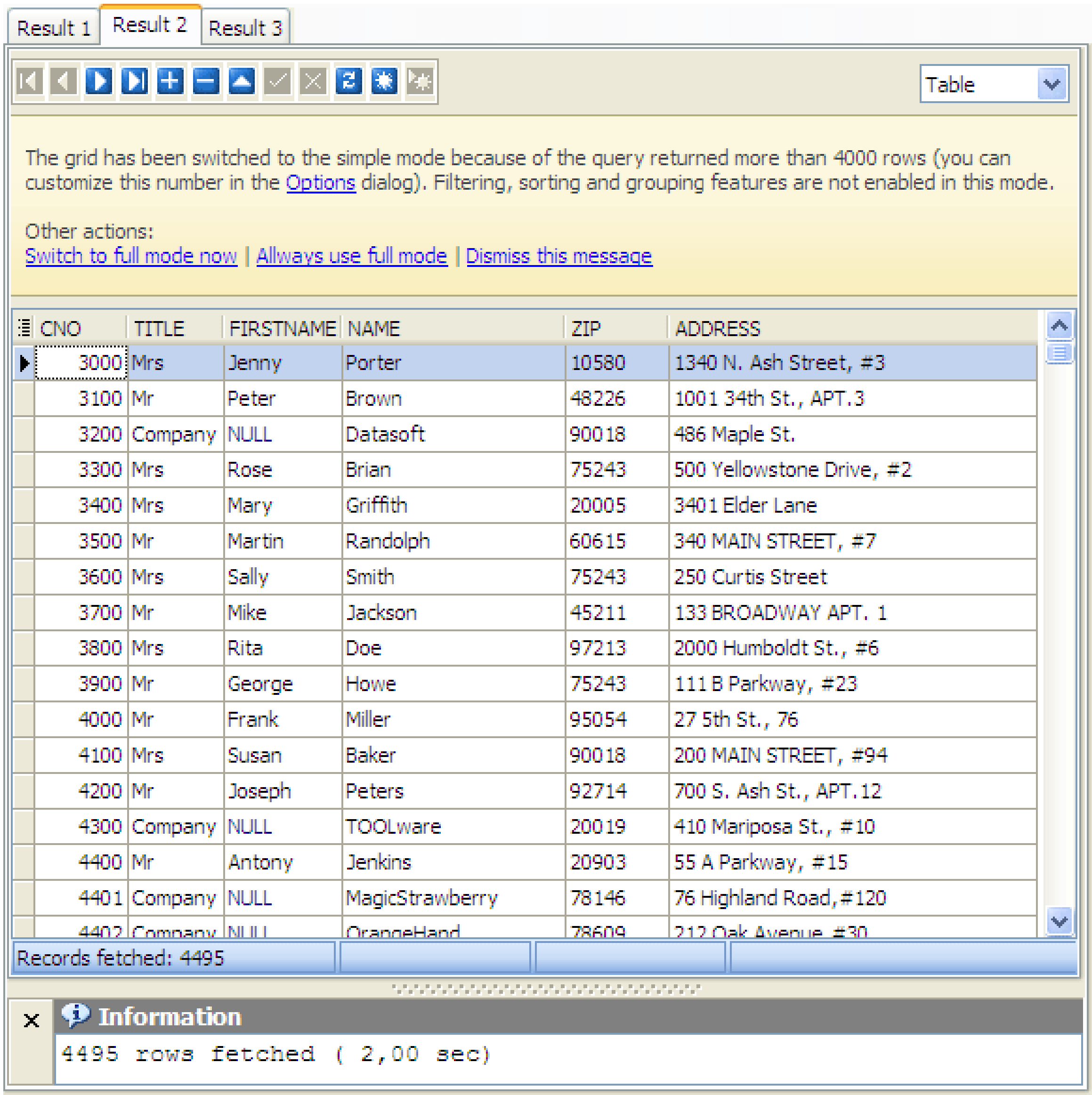
See also: SQL Editor 41, Visual Query Builder 46

6.1.1 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.



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 | Allways 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)
```

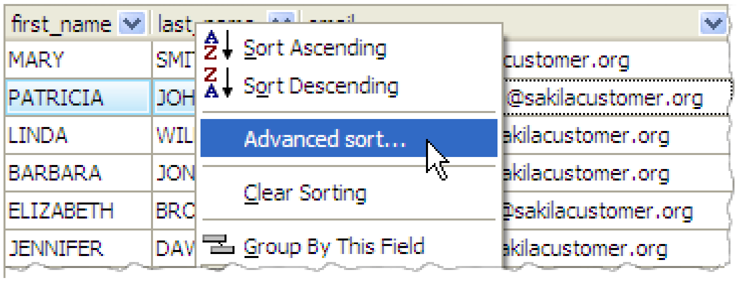
By default, the grid automatically switches to the simple mode for queries returning more than 5000 records (the number can be customized in the Options 120 dialog).

The following abilities are not available in the simple grid mode:

Sorting data (only in the full grid mode)

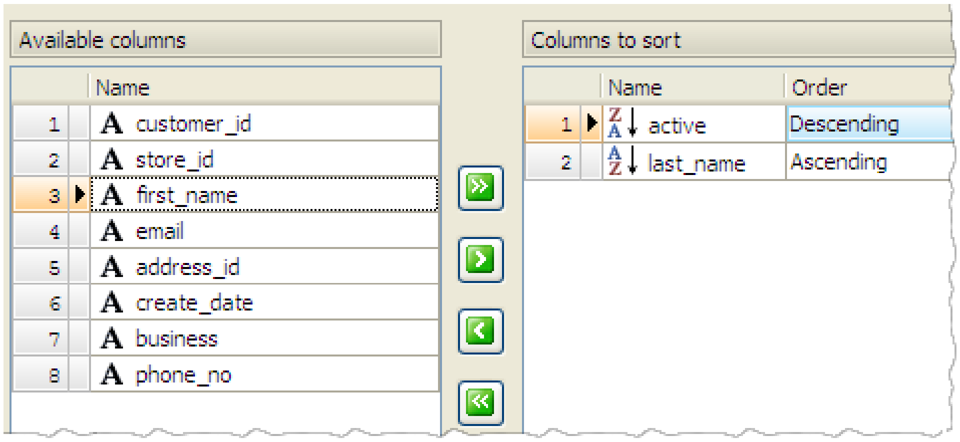
Click the column caption to sort data by the values of this column. To select sort order (ascending or descending), use popup menu of the column caption.

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.



first_name	last_name	email
MARY	SMI	customer.org
PATRICIA	JOH	@sakilacustomer.org
LINDA	WIL	akilacustomer.org
BARBARA	JON	akilacustomer.org
ELIZABETH	BRO	@sakilacustomer.org
JENNIFER	DAV	akilacustomer.org

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.



Available columns

	Name
1	customer_id
2	store_id
3	first_name
4	email
5	address_id
6	create_date
7	business
8	phone_no

Columns to sort

	Name	Order
1	active	Descending
2	last_name	Ascending

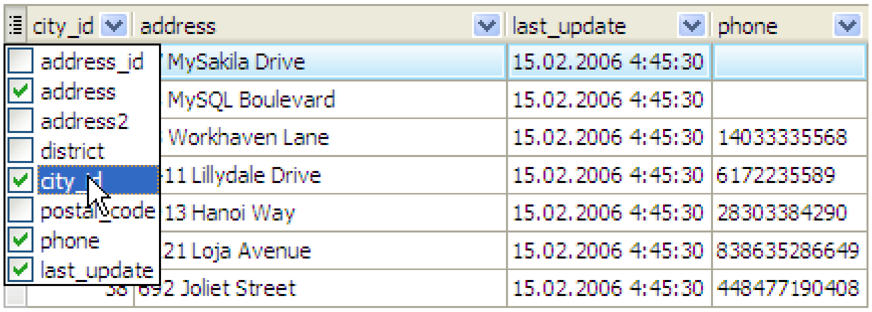
To cancel the sorting order, press **Ctrl** and click on the sorted column caption.

Filtering represented records (only in the full grid mode)

There are several ways to filter data represented in grid. See the corresponding topic 67 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.



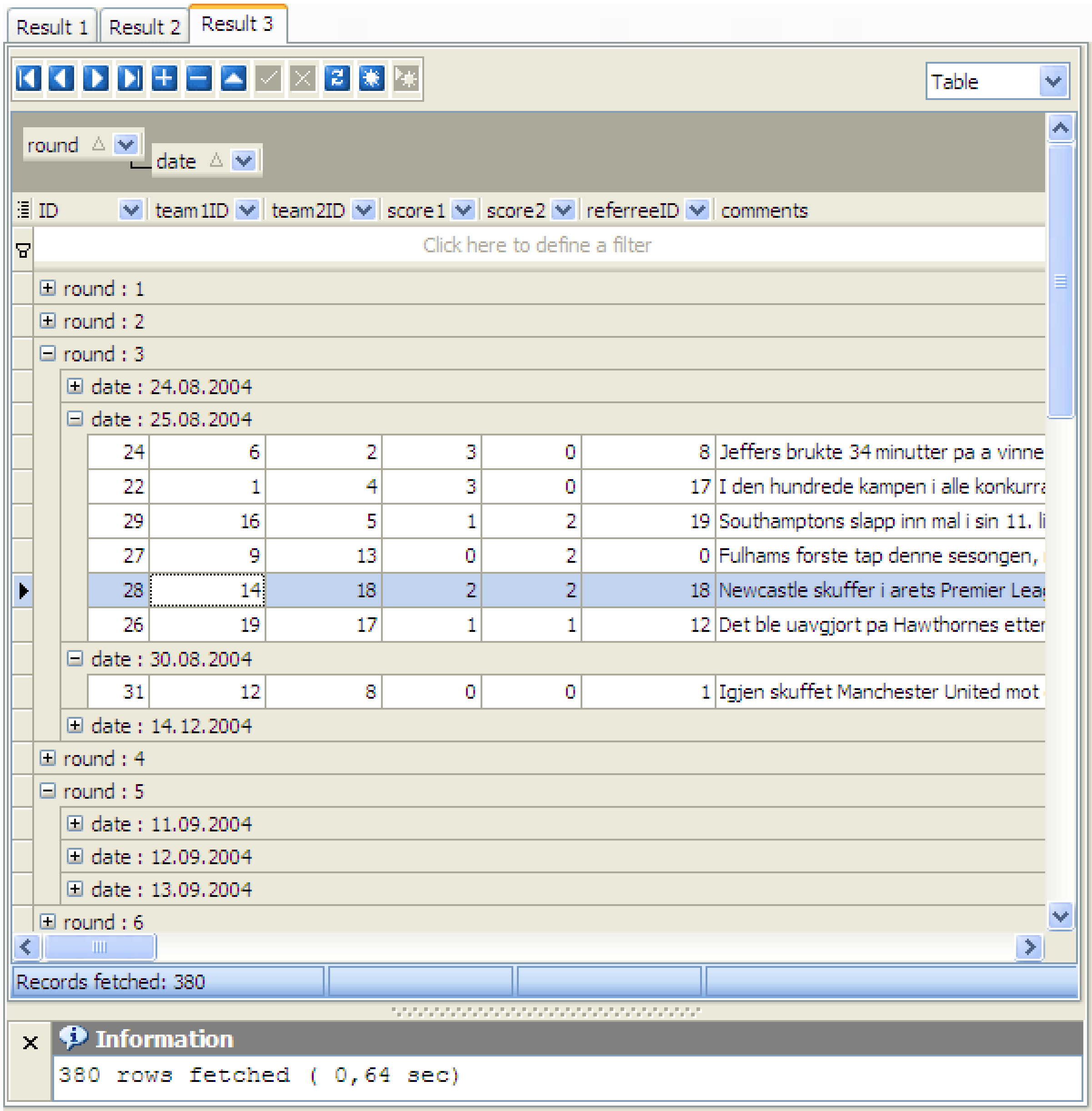
address	last_update	phone
MySakila Drive	15.02.2006 4:45:30	
MySQL Boulevard	15.02.2006 4:45:30	
Workhaven Lane	15.02.2006 4:45:30	14033335568
11 Lillydale Drive	15.02.2006 4:45:30	6172235589
13 Hanoi Way	15.02.2006 4:45:30	28303384290
21 Loja Avenue	15.02.2006 4:45:30	838635286649
692 Joliet Street	15.02.2006 4:45:30	448477190408

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.



ID	team1ID	team2ID	score1	score2	referreeID	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 pa a vinne
22	1	4	3	0	17	I den hundrede kampen i alle konkurra
29	16	5	1	2	19	Southamptons slapp inn mal i sin 11. li
27	9	13	0	2	0	Fulhams forste tap denne sesongen,
28	14	18	2	2	18	Newcastle skuffer i arets Premier Lea
26	19	17	1	1	12	Det ble uavgjort pa 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						

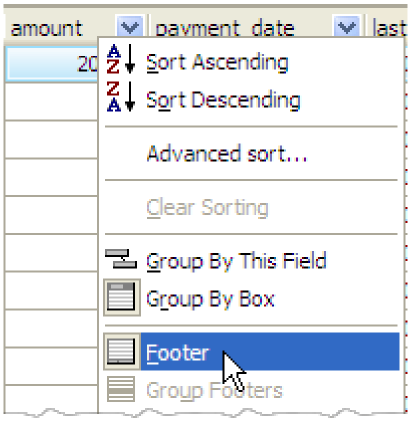
Records fetched: 380

Information

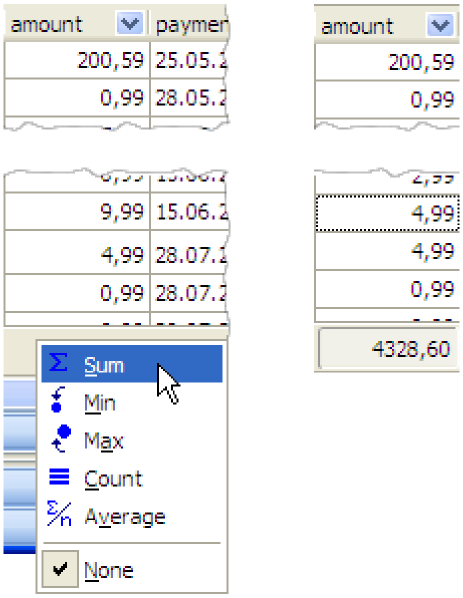
```
380 rows fetched ( 0,64 sec)
```

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.



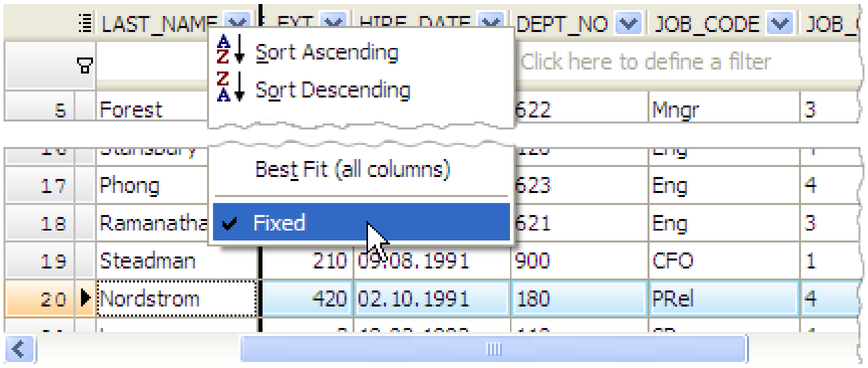
For grouped data use Group Footers.

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.



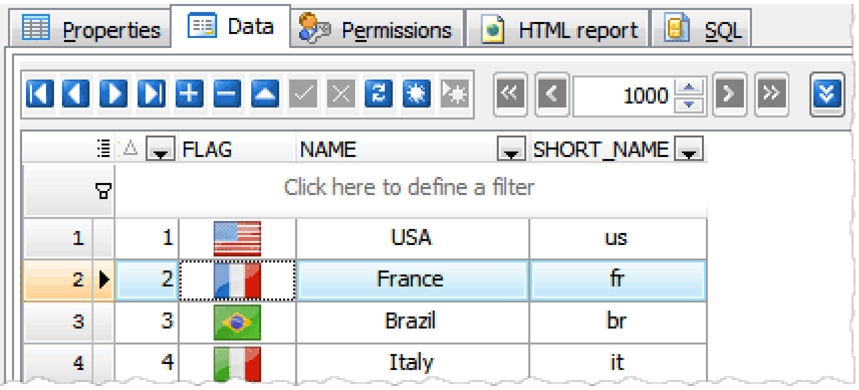
	LAST_NAME	EXT	HIRE_DATE	DEPT_NO	JOB_CODE	JOB_
5	Forest			622	Mngr	3
17	Phong			623	Eng	4
18	Ramanatha			621	Eng	3
19	Steadman	210	09.08.1991	900	CFO	1
20	Nordstrom	420	02.10.1991	180	PRel	4

Row numbering

There is also a possibility to display row numbers in grids. You can adjust 121 the corresponding column to yours liking.

Inline images

It is possible to display images directly in the grid as on the picture below.



		FLAG	NAME	SHORT_NAME
1	1		USA	us
2	2		France	fr
3	3		Brazil	br
4	4		Italy	it

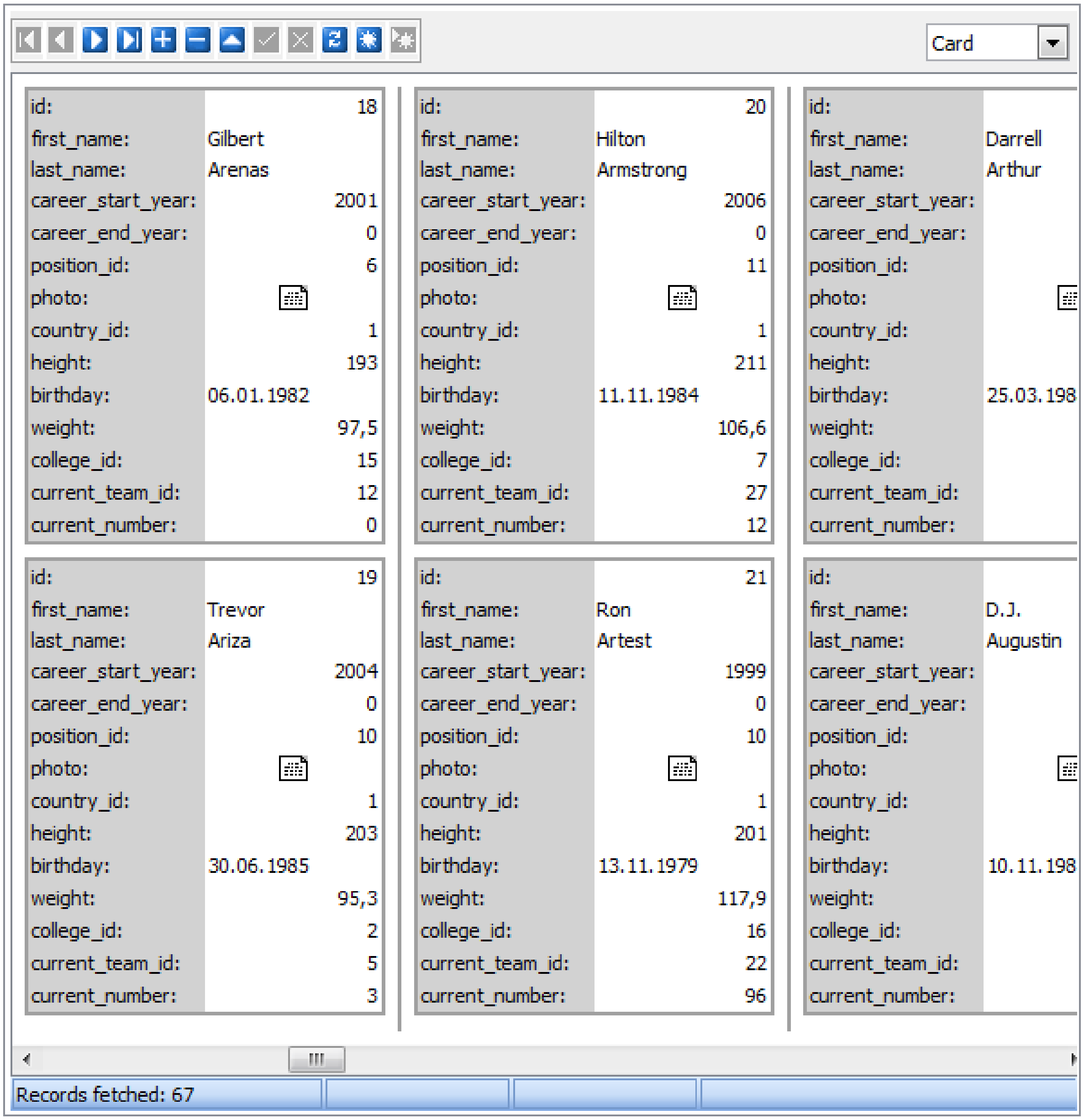
To enable/disable this view mode, open the *Enable inline images* window using the *Manage inline images* item of the column popup menu. The window options allow to set or change the image fitting and specify the row height. To add new images or change existing ones, use BLOB Editor 70 (see below).

Working with BLOBs

To edit a BLOB field 70, 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).

6.1.2 Working with info cards

Info cards correspond to the records. You can filter records by custom conditions 67 and edit data directly in info cards or with Data Input Form 66.



Card

id: 18
first_name: Gilbert
last_name: Arenas
career_start_year: 2001
career_end_year: 0
position_id: 6
photo:
country_id: 1
height: 193
birthday: 06.01.1982
weight: 97,5
college_id: 15
current_team_id: 12
current_number: 0

id: 20
first_name: Hilton
last_name: Armstrong
career_start_year: 2006
career_end_year: 0
position_id: 11
photo:
country_id: 1
height: 211
birthday: 11.11.1984
weight: 106,6
college_id: 7
current_team_id: 27
current_number: 12

id:
first_name: Darrell
last_name: Arthur
career_start_year:
career_end_year:
position_id:
photo:
country_id:
height:
birthday: 25.03.198
weight:
college_id:
current_team_id:
current_number:

id: 19
first_name: Trevor
last_name: Ariza
career_start_year: 2004
career_end_year: 0
position_id: 10
photo:
country_id: 1
height: 203
birthday: 30.06.1985
weight: 95,3
college_id: 2
current_team_id: 5
current_number: 3

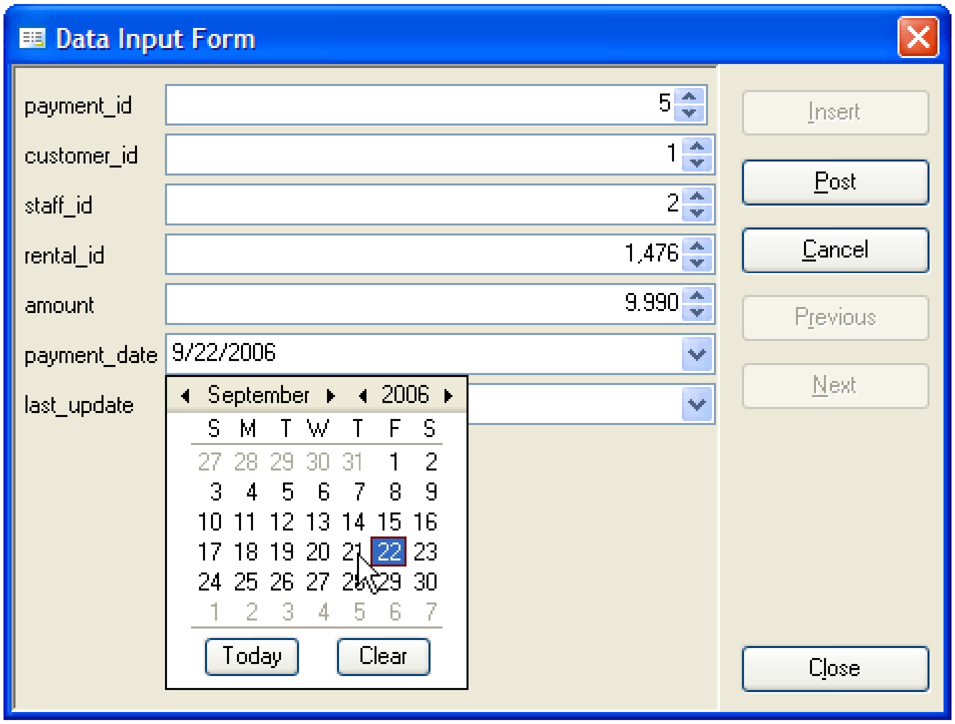
id: 21
first_name: Ron
last_name: Artest
career_start_year: 1999
career_end_year: 0
position_id: 10
photo:
country_id: 1
height: 201
birthday: 13.11.1979
weight: 117,9
college_id: 16
current_team_id: 22
current_number: 96

id:
first_name: D.J.
last_name: Augustin
career_start_year:
career_end_year:
position_id:
photo:
country_id:
height:
birthday: 10.11.198
weight:
college_id:
current_team_id:
current_number:

Records fetched: 67

6.1.3 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.



payment_id 5
customer_id 1
staff_id 2
rental_id 1,476
amount 9.990
payment_date 9/22/2006
last_update

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.

6.1.4 Data filtering

SQLite Code Factory 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**

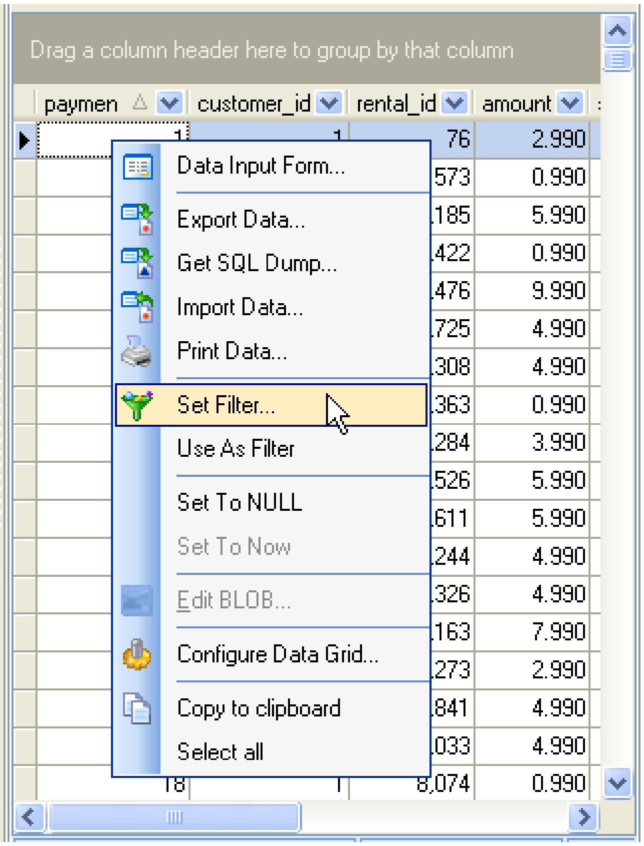
  Use the drop-down button in the column caption area to filter records by the selected column value(s).

- **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.

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 41 and save as a query. You can also use Generate query link on the Navigation bar.

See also: Data View 59, SQL Editor 41, Visual Query Builder 46

6.2 BLOB Editor

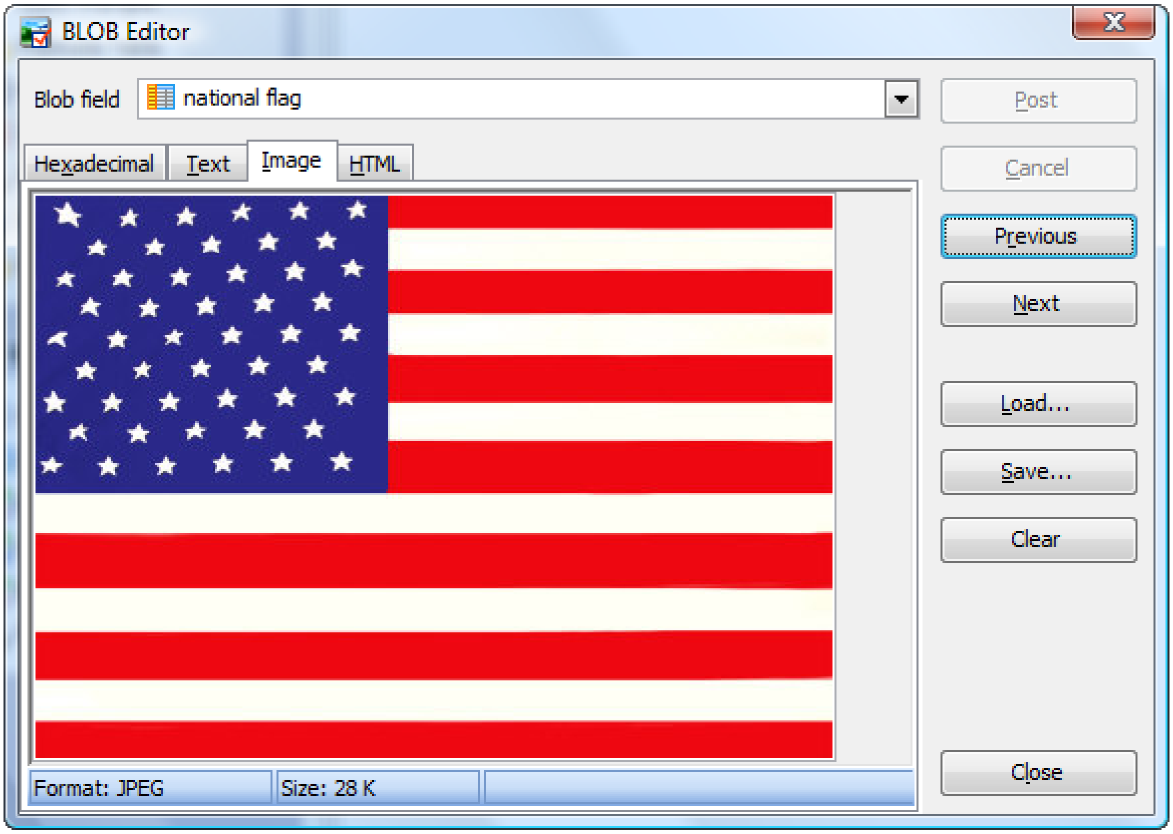
BLOB Editor is a tool to view and edit BLOB data in the following ways: hexadecimal dump 71, plain text 71, graphical image 70, HTML page 72, or PDF document 73. BLOB Editor is invoked from the result tab of SQL Editor 41 and Visual Query Builder 46 by double clicking of the BLOB field to be edited or with the Edit BLOB link of the field's popup menu. The editor also can be called from BLOB Viewer 94 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.

6.2.1 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.



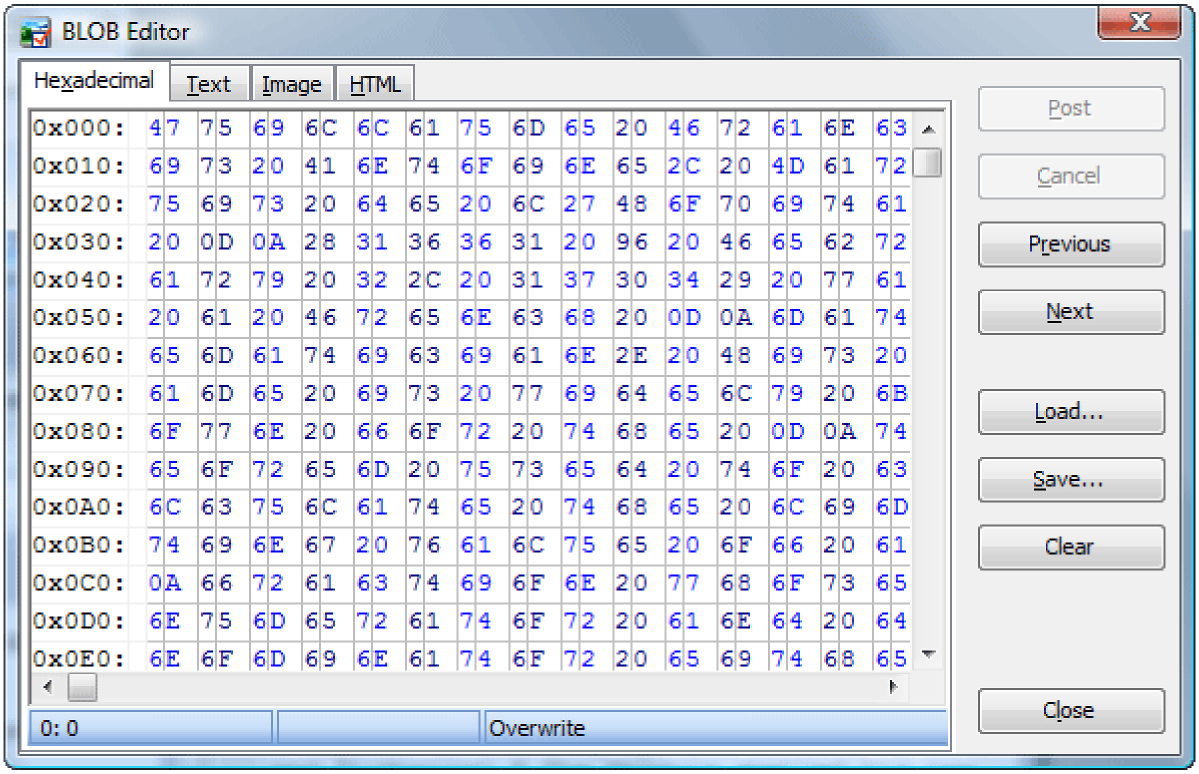
Blob field: national flag

Format: JPEG

Size: 28 K

6.2.2 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.

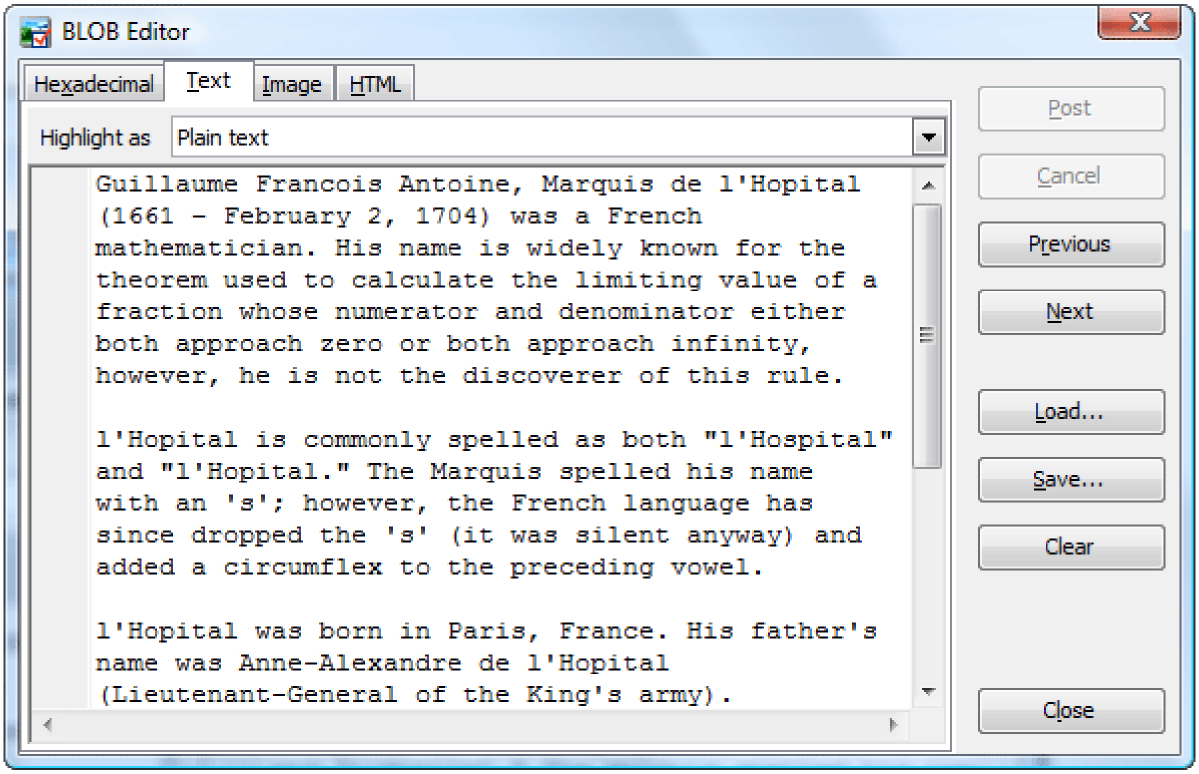


```
0x000: 47 75 69 6C 6C 61 75 6D 65 20 46 72 61 6E 63
0x010: 69 73 20 41 6E 74 6F 69 6E 65 2C 20 4D 61 72
0x020: 75 69 73 20 64 65 20 6C 27 48 6F 70 69 74 61
0x030: 20 0D 0A 28 31 36 36 31 20 96 20 46 65 62 72
0x040: 61 72 79 20 32 2C 20 31 37 30 34 29 20 77 61
0x050: 20 61 20 46 72 65 6E 63 68 20 0D 0A 6D 61 74
0x060: 65 6D 61 74 69 63 69 61 6E 2E 20 48 69 73 20
0x070: 61 6D 65 20 69 73 20 77 69 64 65 6C 79 20 6B
0x080: 6F 77 6E 20 66 6F 72 20 74 68 65 20 0D 0A 74
0x090: 65 6F 72 65 6D 20 75 73 65 64 20 74 6F 20 63
0x0A0: 6C 63 75 6C 61 74 65 20 74 68 65 20 6C 69 6D
0x0B0: 74 69 6E 67 20 76 61 6C 75 65 20 6F 66 20 61
0x0C0: 0A 66 72 61 63 74 69 6F 6E 20 77 68 6F 73 65
0x0D0: 6E 75 6D 65 72 61 74 6F 72 20 61 6E 64 20 64
0x0E0: 6E 6F 6D 69 6E 61 74 6F 72 20 65 69 74 68 65
```

```
0: 0 | | Overwrite
```

6.2.3 Editing as plain text

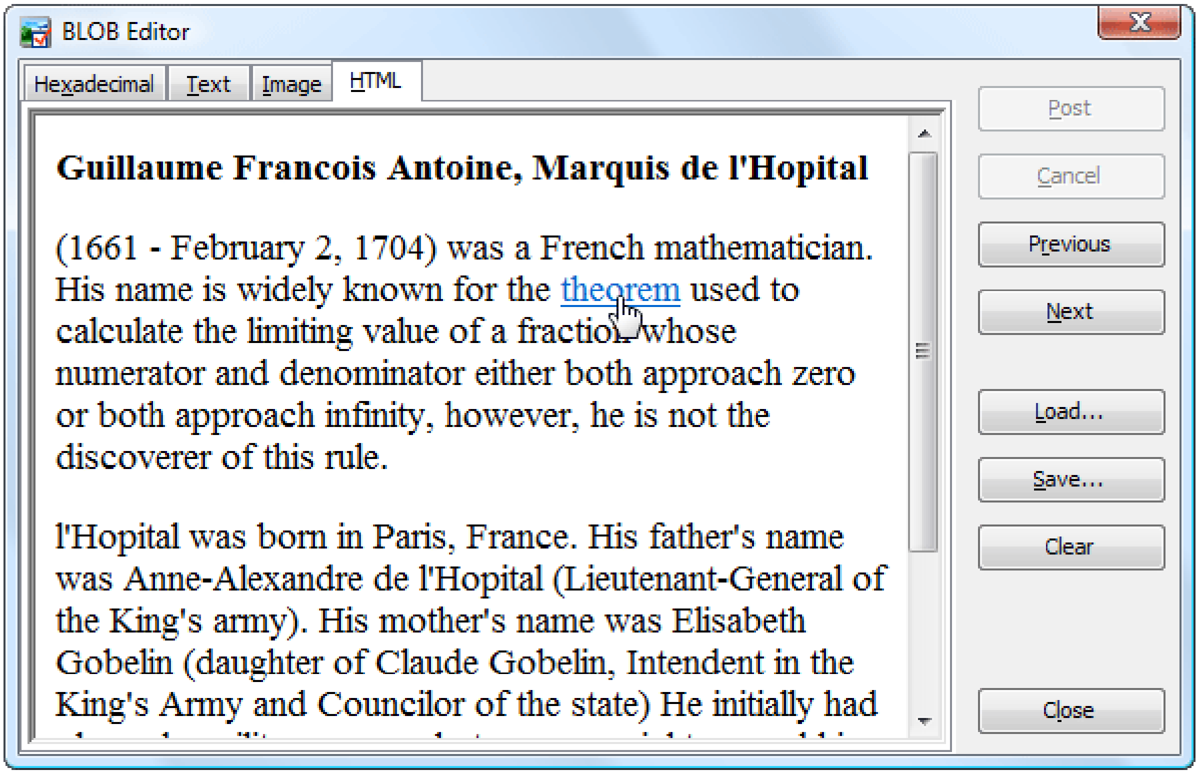
The Text panel allows you to edit data as a simple text. Several types of text highlighting are available (*Plain text, SQL, XML, Java, VBScript, JScript, Cmd batch, PHP, CSS, UnixShell Script, INI,* and *HTML*). The popup menu of the panel allows you to invoke Find Text, Replace Text and Go to line dialogs.



```
Guillaume Francois Antoine, Marquis de l'Hopital
(1661 - February 2, 1704) was a French
mathematician. His name is widely known for the
theorem used to calculate the limiting value of a
fraction whose numerator and denominator either
both approach zero or both approach infinity,
however, he is not the discoverer of this rule.

l'Hopital is commonly spelled as both "l'Hospital"
and "l'Hopital." The Marquis spelled his name
with an 's'; however, the French language has
since dropped the 's' (it was silent anyway) and
added a circumflex to the preceding vowel.

l'Hopital was born in Paris, France. His father's
name was Anne-Alexandre de l'Hopital
(Lieutenant-General of the King's army).
```

6.2.4 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.



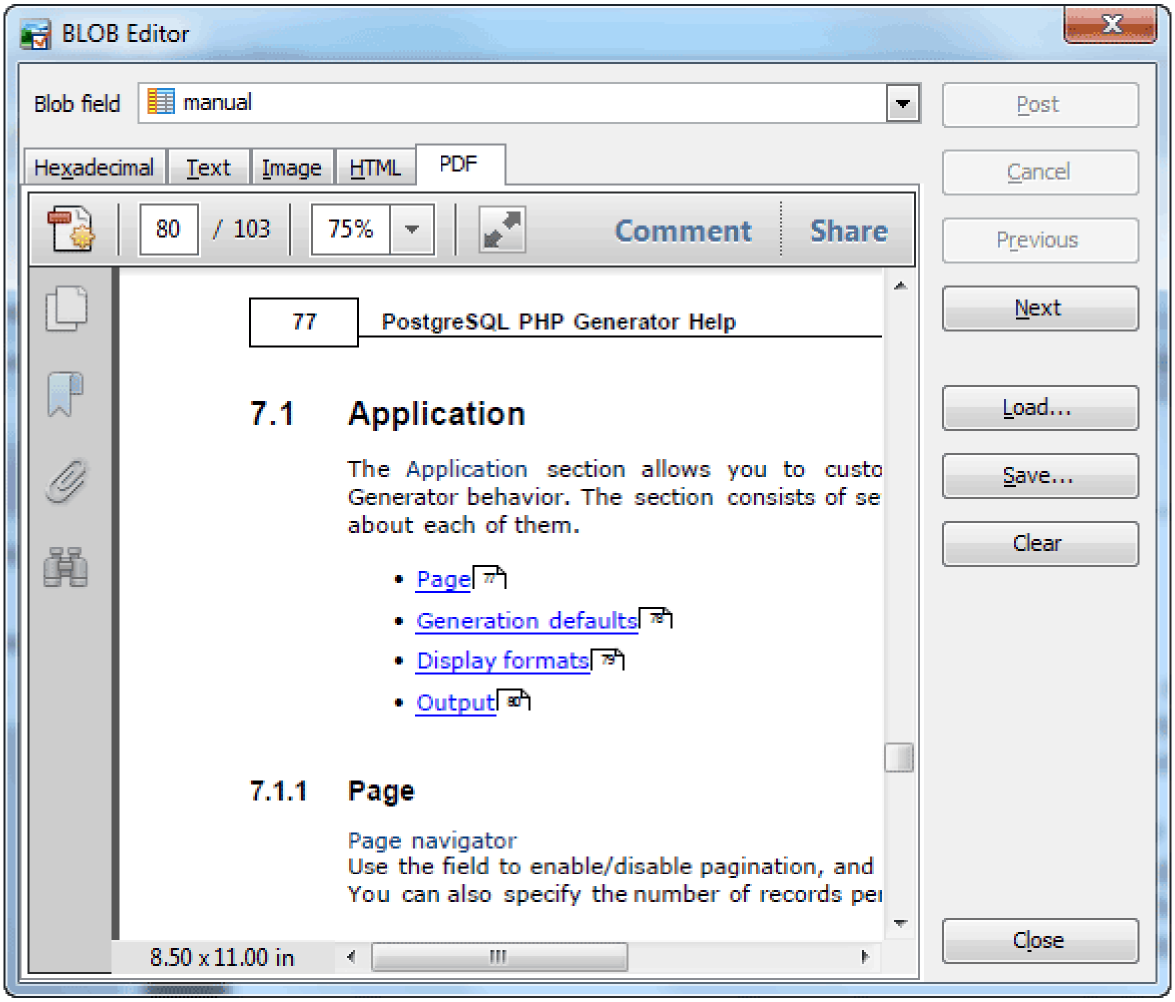
Guillaume Francois Antoine, Marquis de l'Hopital

(1661 - February 2, 1704) was a French mathematician. His name is widely known for the theorem used to calculate the limiting value of a fraction whose numerator and denominator either both approach zero or both approach infinity, however, he is not the discoverer of this rule.

l'Hopital was born in Paris, France. His father's name was Anne-Alexandre de l'Hopital (Lieutenant-General of the King's army). His mother's name was Elisabeth Gobelin (daughter of Claude Gobelin, Intendent in the King's Army and Councilor of the state) He initially had

6.2.5 Editing as PDF document

The PDF panel presents field data as PDF document. To accomplish common operations with data, use the Adobe Reader toolbar.



77 PostgreSQL PHP Generator Help

7.1 Application

The Application section allows you to custo
Generator behavior. The section consists of se
about each of them.

- Page 77
- Generation defaults 78
- Display formats 79
- Output 80

7.1.1 Page

Page navigator
Use the field to enable/disable pagination, and
You can also specify the number of records pe

6.3 Export Data Wizard

Data Export wizard is a tool to save data from SQLite 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

- 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,

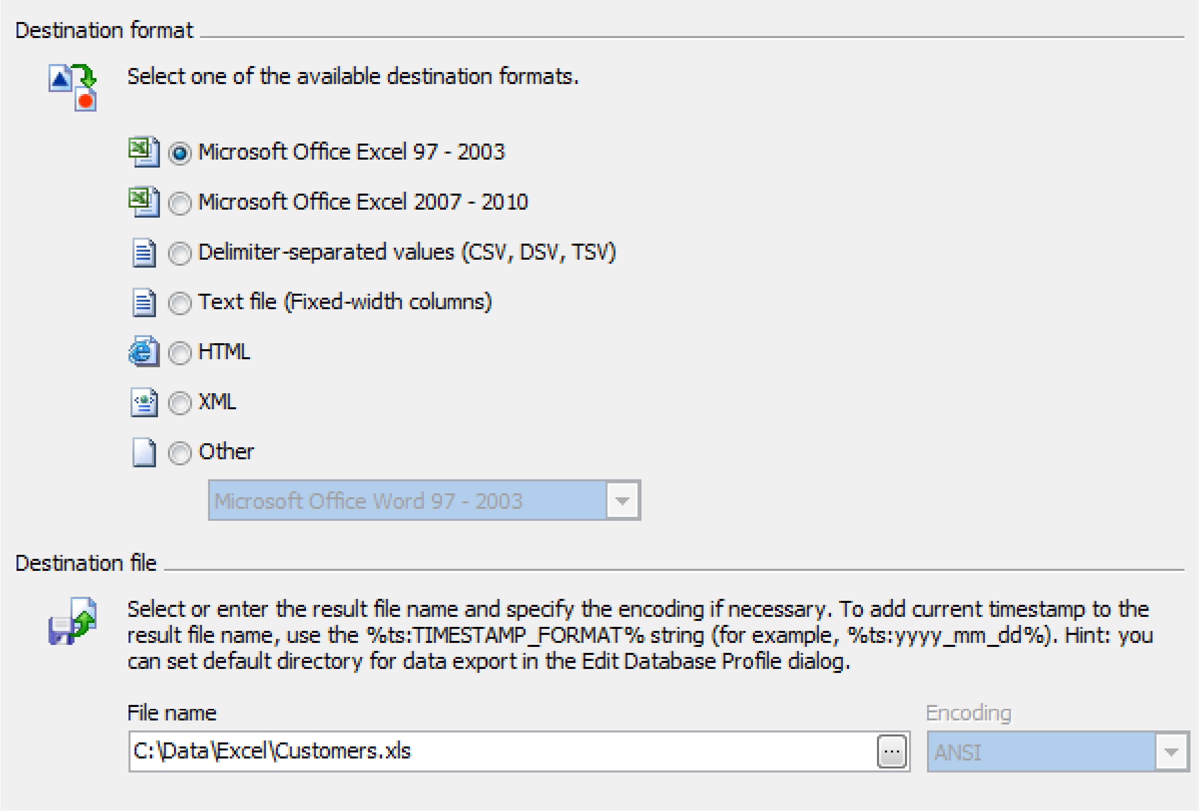
- Set the format and the name 75 of the destination file;
- Specify such additional options of the result file as header and footer 76, formats applied to exported data 77 and some format-specific options 78;
- Select columns 77 you want to include into result files;
- Specify other export options 81.

See also: Get SQL Dump, Import Data Wizard 85

6.3.1 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

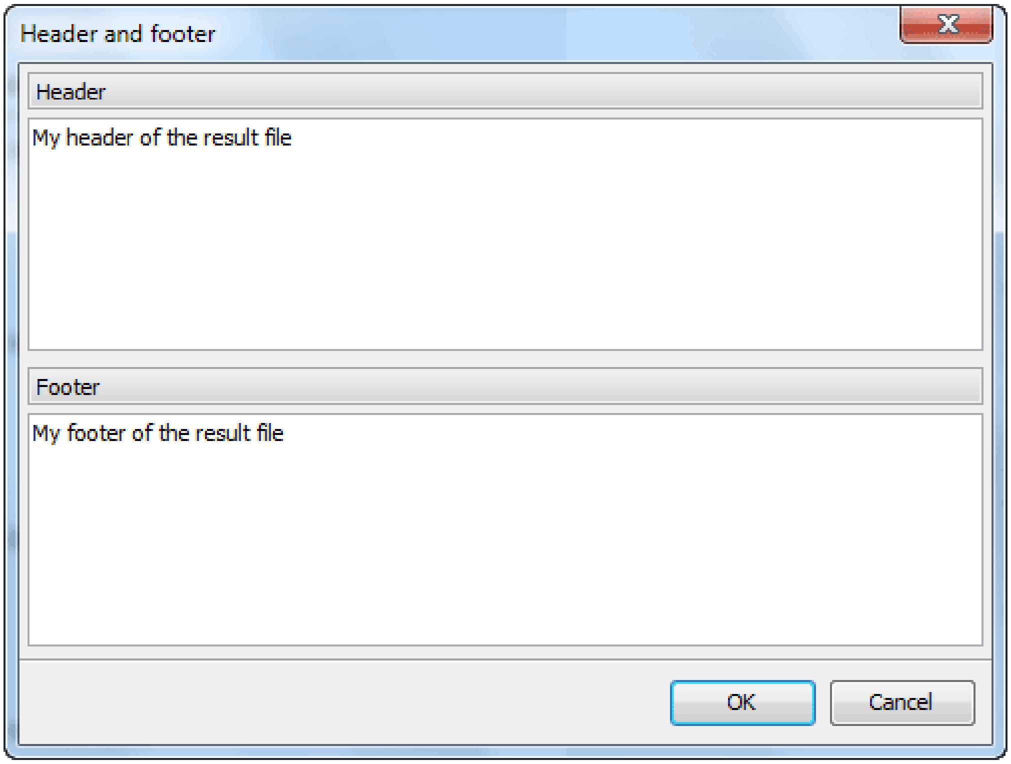
Select or enter the result file name and specify the encoding if necessary. To add current timestamp to the result file name, use the %ts:TIMESTAMP_FORMAT% string (for example, %ts:yyyy_mm_dd%). Hint: you can set default directory for data export in the Edit Database Profile dialog.

File name: C:\Data\Excel\Customers.xls

Encoding: ANSI

6.3.2 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.



Header

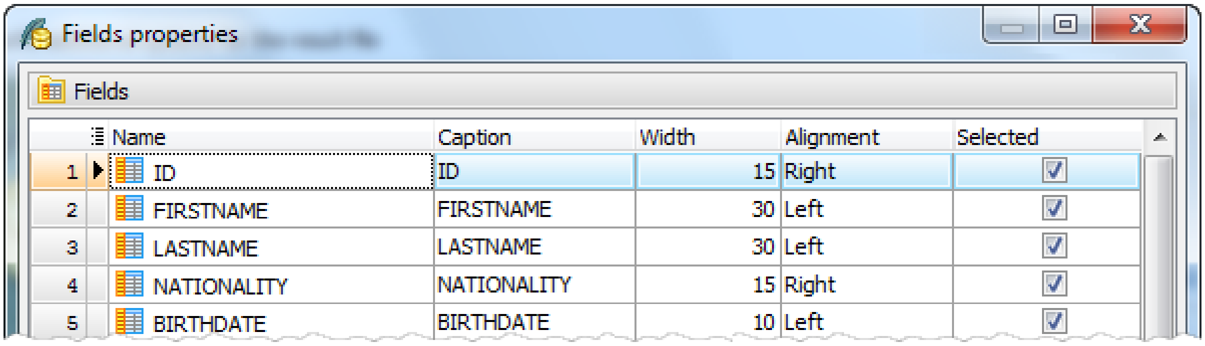
My header of the result file

Footer

My footer of the result file

6.3.3 Selecting fields for export

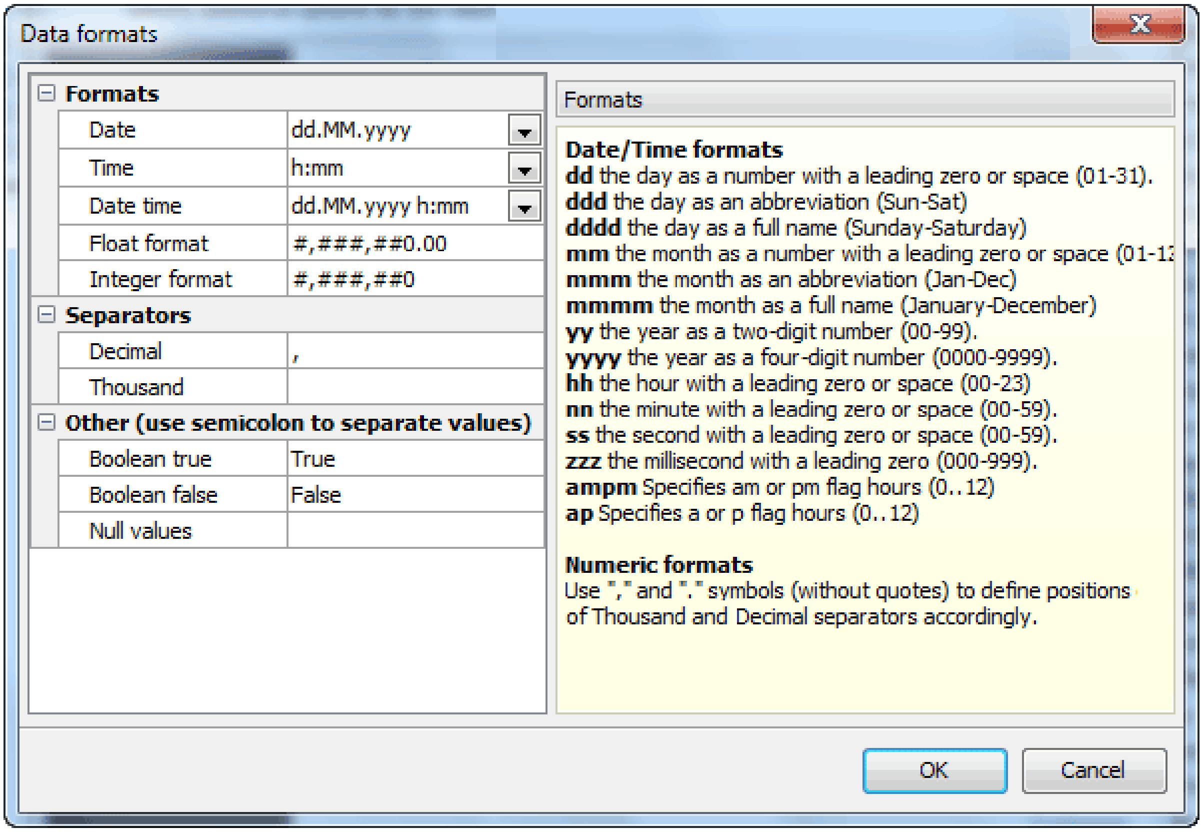
Uncheck the Selected box to exclude the corresponding field from the export, specify a Caption to be used for the result column, and also width, and alignment for output columns (when applicable).



	Name	Caption	Width	Alignment	Selected
1	ID	ID	15	Right	☑
2	FIRSTNAME	FIRSTNAME	30	Left	☑
3	LASTNAME	LASTNAME	30	Left	☑
4	NATIONALITY	NATIONALITY	15	Right	☑
5	BIRTHDATE	BIRTHDATE	10	Left	☑

6.3.4 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.



Formats	
Date	dd.MM.yyyy
Time	h:mm
Date time	dd.MM.yyyy h:mm
Float format	#,###,##0.00
Integer format	#,###,##0
Separators	
Decimal	,
Thousand	
Other (use semicolon to separate values)	
Boolean true	True
Boolean false	False
Null values	

Formats

Date/Time formats
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 (01-1
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)

Numeric formats
Use "," and "." symbols (without quotes) to define positions of Thousand and Decimal separators accordingly.

6.3.5 Setting format-specific options

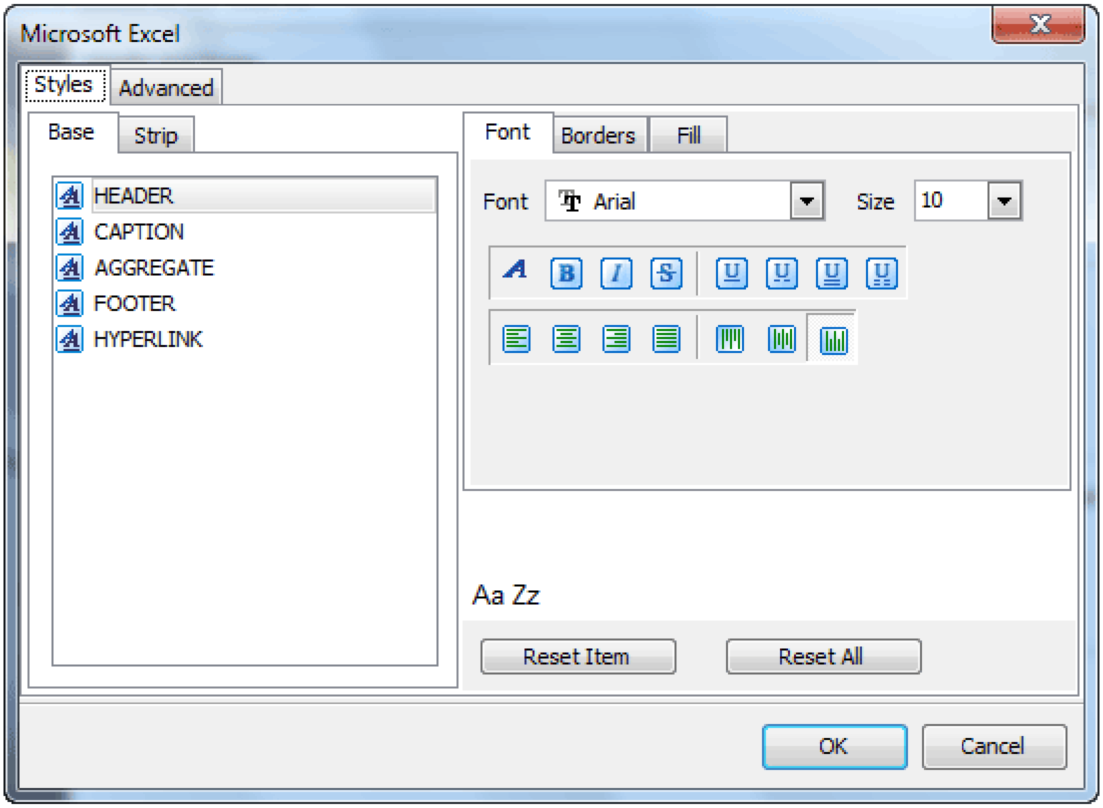
Each format supposes corresponding additional export options. Use the wizard option to adjust export properties depending on the target file format you have selected earlier. The following formats are at your disposal: Microsoft Excel 78, Microsoft Excel 2007, CSV 80, Text 80, HTML 79, XML 80, Microsoft 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.



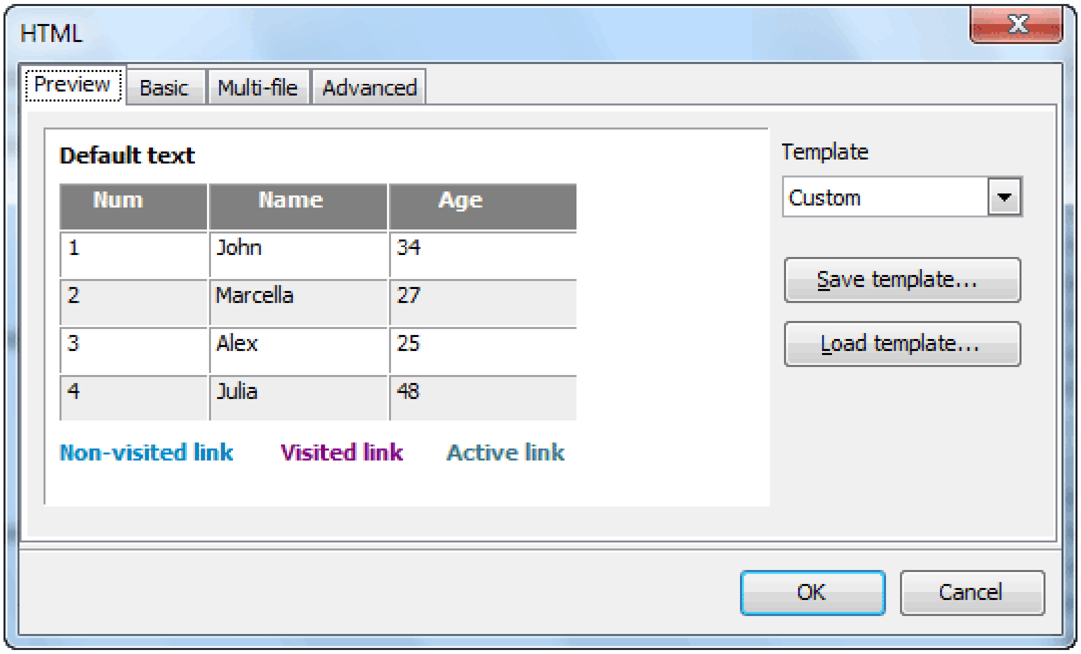
HTML

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.



Default text

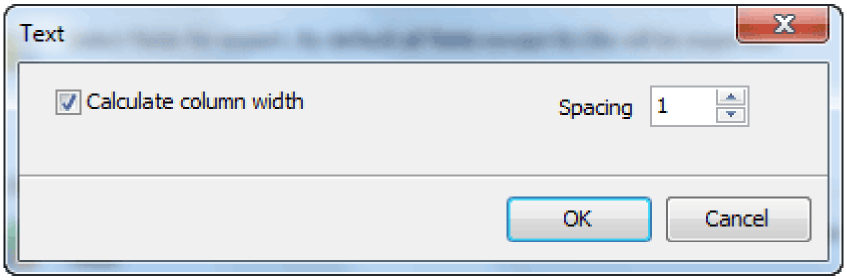
Num	Name	Age
1	John	34
2	Marcella	27
3	Alex	25
4	Julia	48

Non-visited link Visited link Active link

Template: Custom

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.

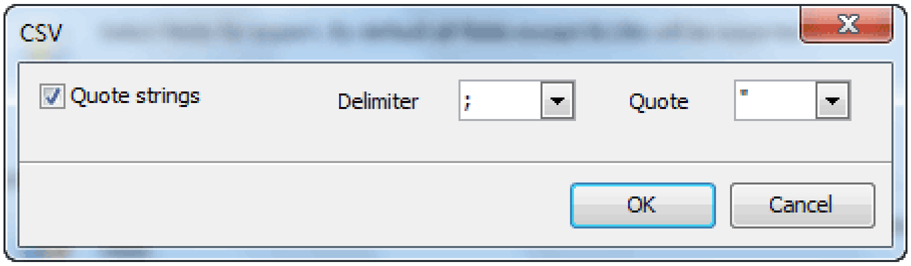


☑ Calculate column width

Spacing: 1

CSV files

You can specify column separator and optional values quote character for the target file on this step.



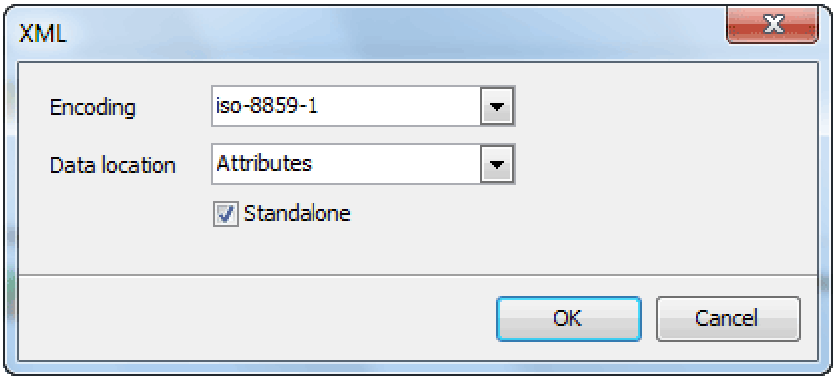
☑ Quote strings

Delimiter: ;

Quote: "

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.



Encoding: iso-8859-1

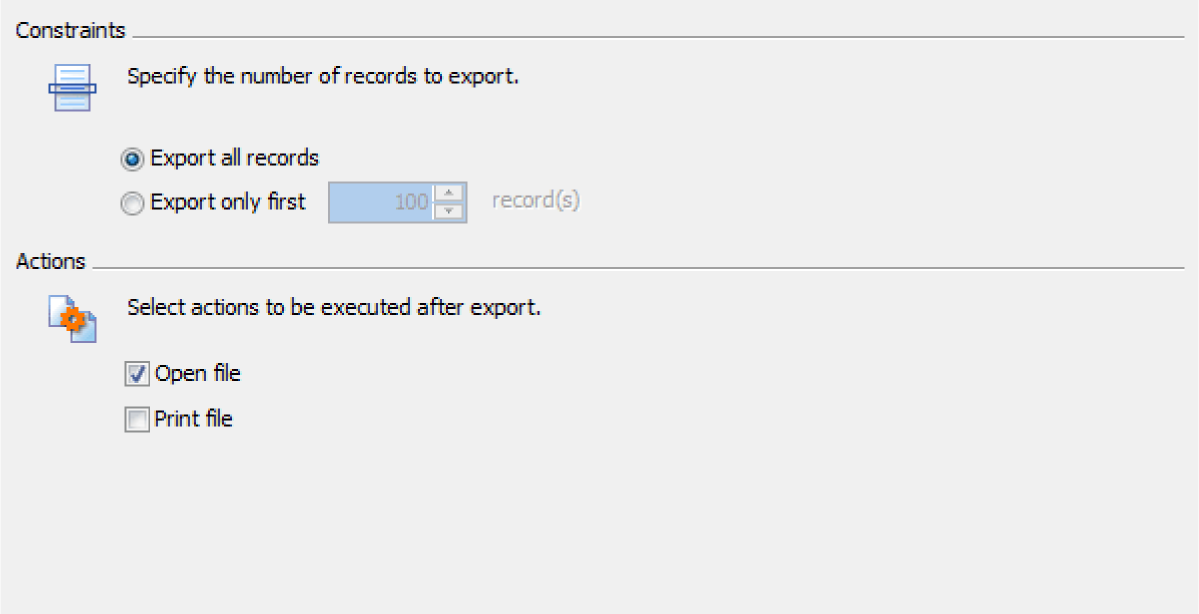
Data location: Attributes

☑ Standalone

6.3.6 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.



Specify the number of records to export.

● Export all records

○ Export only first 100 record(s)

Select actions to be executed after export.

☑ Open file

☐ Print file

6.4 Get SQL Dump

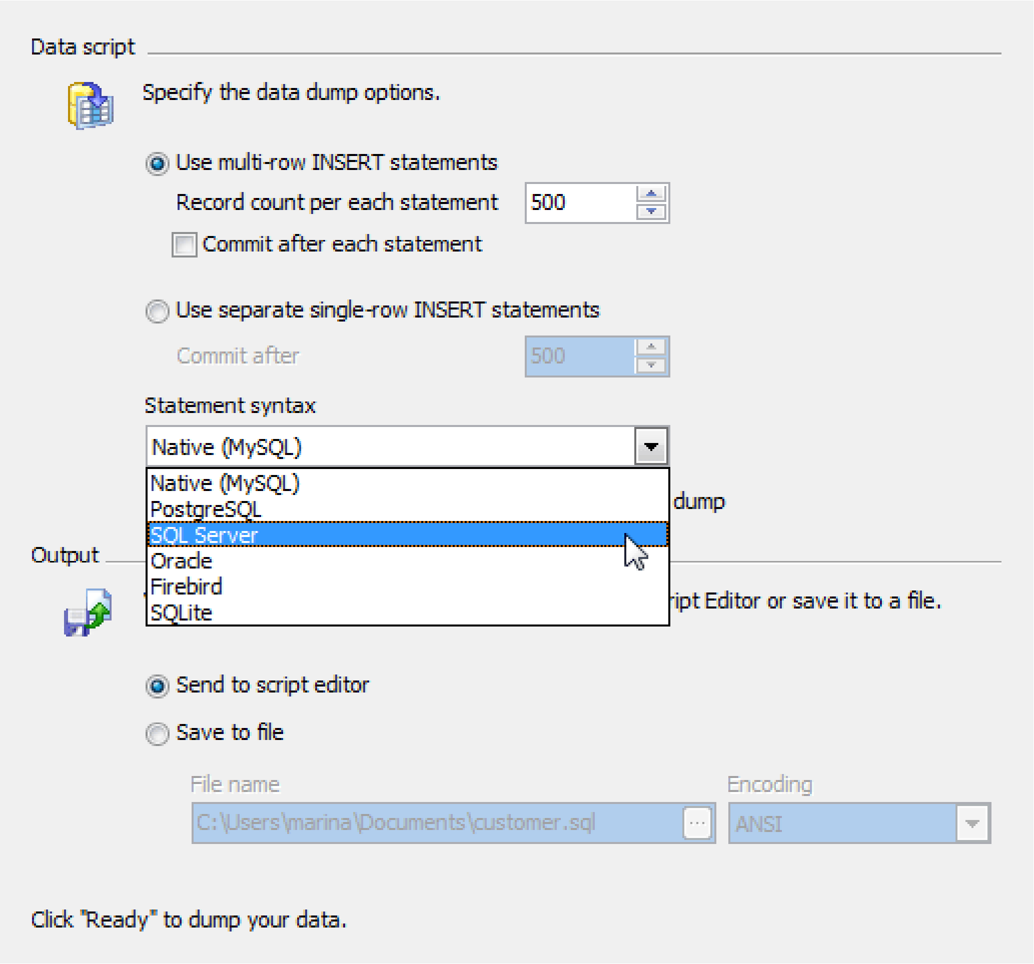
Get SQL Dump Wizard allows you to export data from a table or a query result to the SQL script as a number of INSERT statements.

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 82
- Specifying dump options 83

See also: Export Data Wizard 75, SQL Script Editor 56



Data script

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

Statement syntax

Native (MySQL)

● Send to script editor

○ Save to file

File name C:\Users\marina\Documents\customer.sql

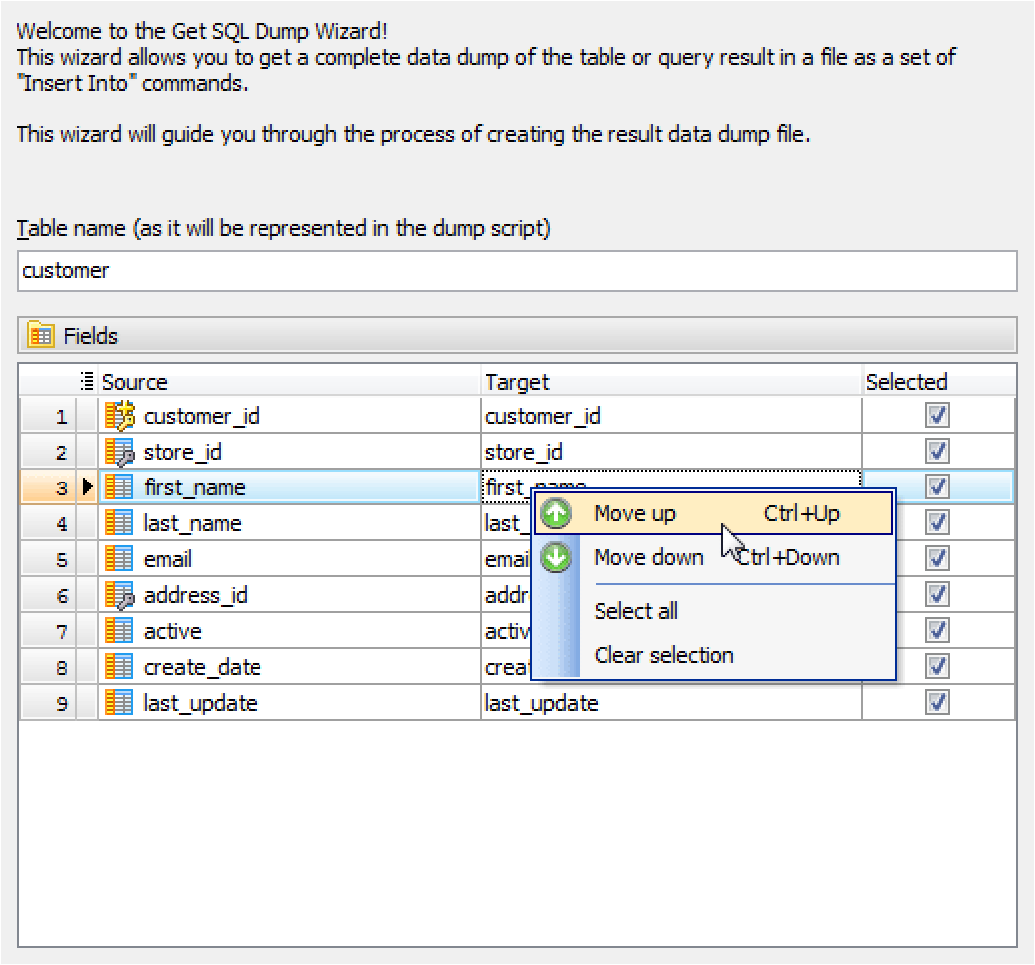
Encoding ANSI

Click "Ready" to dump your data.

6.4.1 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 26.



Welcome to the Get SQL Dump Wizard!
This wizard allows you to get a complete data dump of the table or query result in a file as a set of "Insert Into" commands.

This wizard will guide you through the process of creating the result data dump file.

Table name (as it will be represented in the dump script)

customer

Fields

	Source	Target	Selected
1	customer_id	customer_id	☑
2	store_id	store_id	☑
3	first_name	first_name	☑
4	last_name	last_	☑
5	email	emai	☑
6	address_id	addr	☑
7	active	activ	☑
8	create_date	creat	☑
9	last_update	last_update	☑

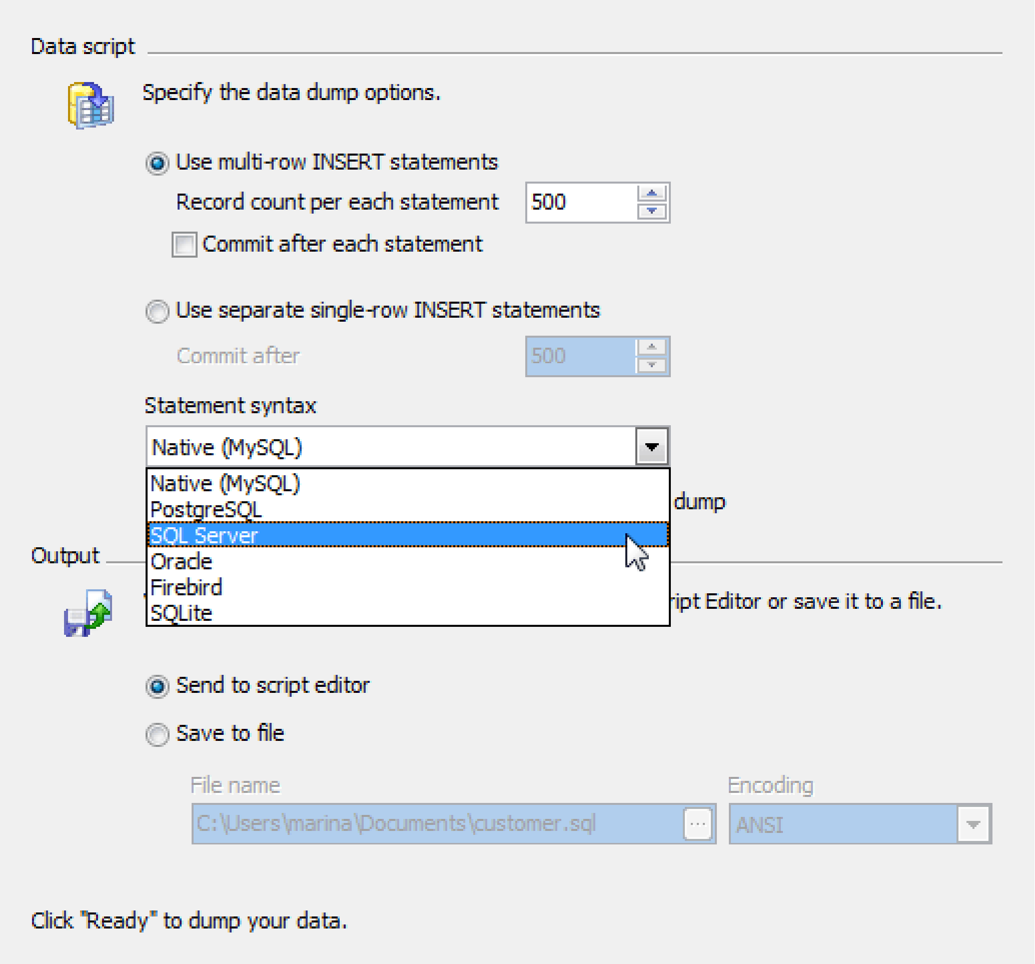
6.4.2 Specifying dump options

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 56 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 500

☐ Commit after each statement

○ Use separate single-row INSERT statements

Commit after 500

Statement syntax

Native (MySQL)

dump

Output

ipt Editor or save it to a file.

● Send to script editor

○ Save to file

File name C:\Users\marina\Documents\customer.sql

Encoding ANSI

Click "Ready" to dump your data.

6.5 Import Data Wizard

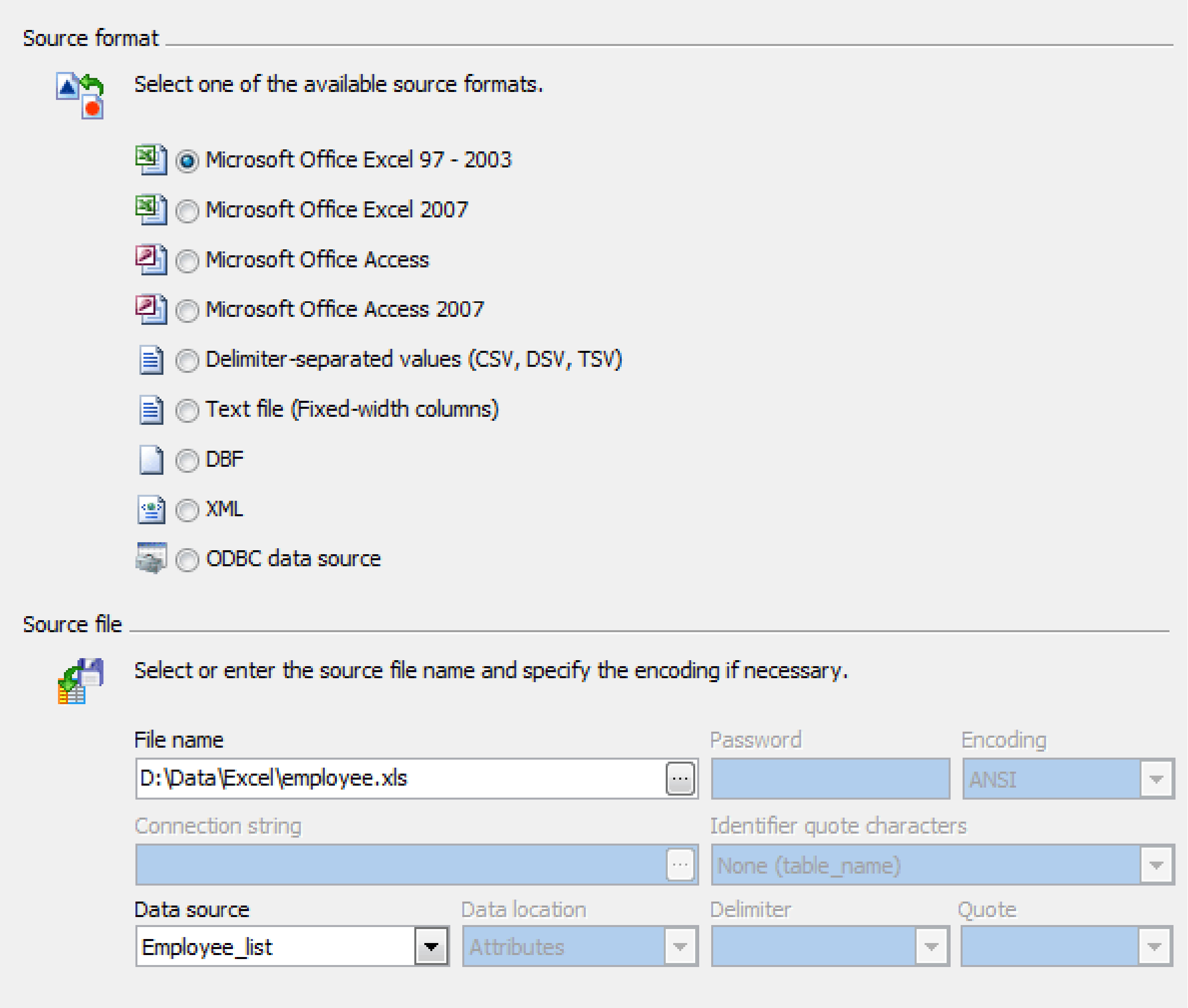
Import Data Wizard provides you with a graphical user interface to import data from the most popular files formats into existing SQLite tables. It allows you to adjust data formats, empty target tables, execute custom SQL scripts, etc.

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

- select the node of the table for importing at the Explorer tree;
- select the Data Management group of the node's popup menu;
- use the Import Data item.

To import data,
- Set the format 86 of the input data and the source file name;
- Map source file columns and target table fields 88;
- Specify other import options 91.



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: D:\Data\Excel\employee.xls

Password:

Encoding: ANSI

Connection string:

Identifier quote characters: None (table_name)

Data source: Employee_list

Data location: Attributes

Delimiter:

Quote:

See also: Export Data Wizard 75

6.5.1 Setting source file name and format

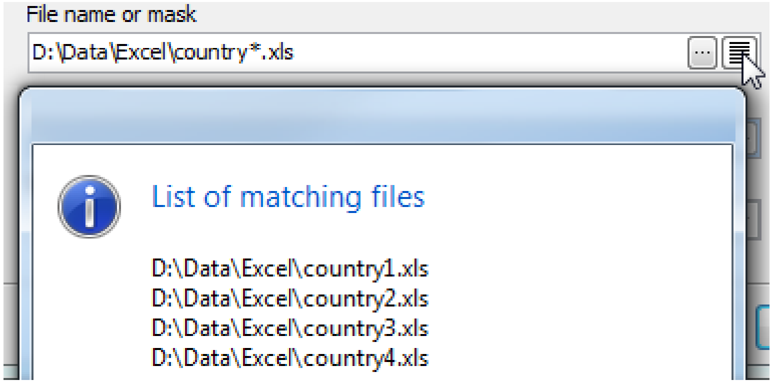
1. Select the format of the source file.

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*.



D:\Data\Excel\country*.xls

List of matching files

D:\Data\Excel\country1.xls
D:\Data\Excel\country2.xls
D:\Data\Excel\country3.xls
D:\Data\Excel\country4.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"?>
```

```
<DATAPACKET Version="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>
```

6.5.2 Setting the accordance between source and target columns

The wizard provides you with several ways to map input data to the target table columns.

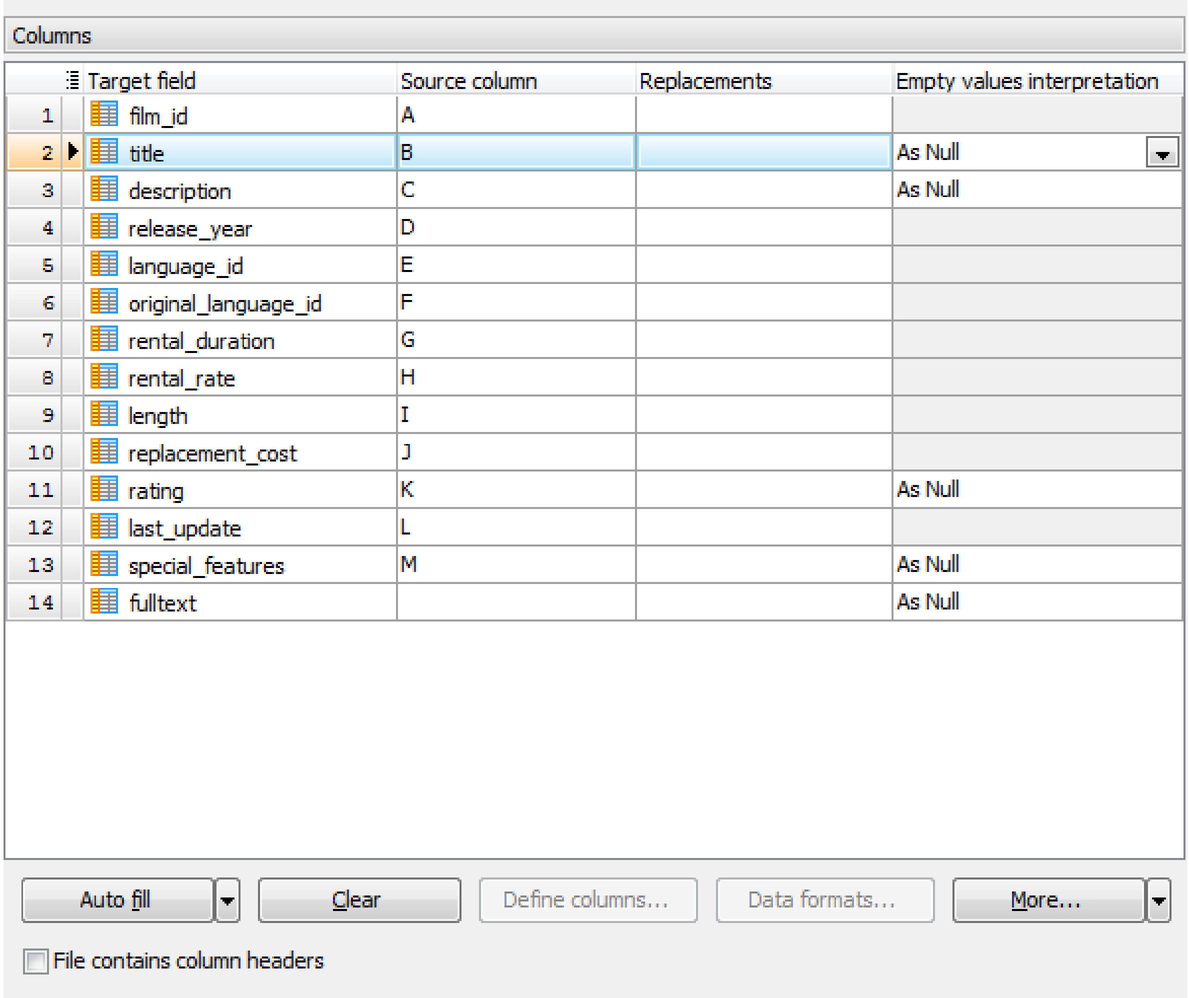
- You can map columns automatically by order with the Auto Fill and Auto fill all maps buttons.
- You can do it manually using the drop-down list of Source column fields.
- To map columns visually, open Map builder 89 with the Build map link.

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 90 used for the input file.

To exclude the first file row, use the File contains column header checkbox.

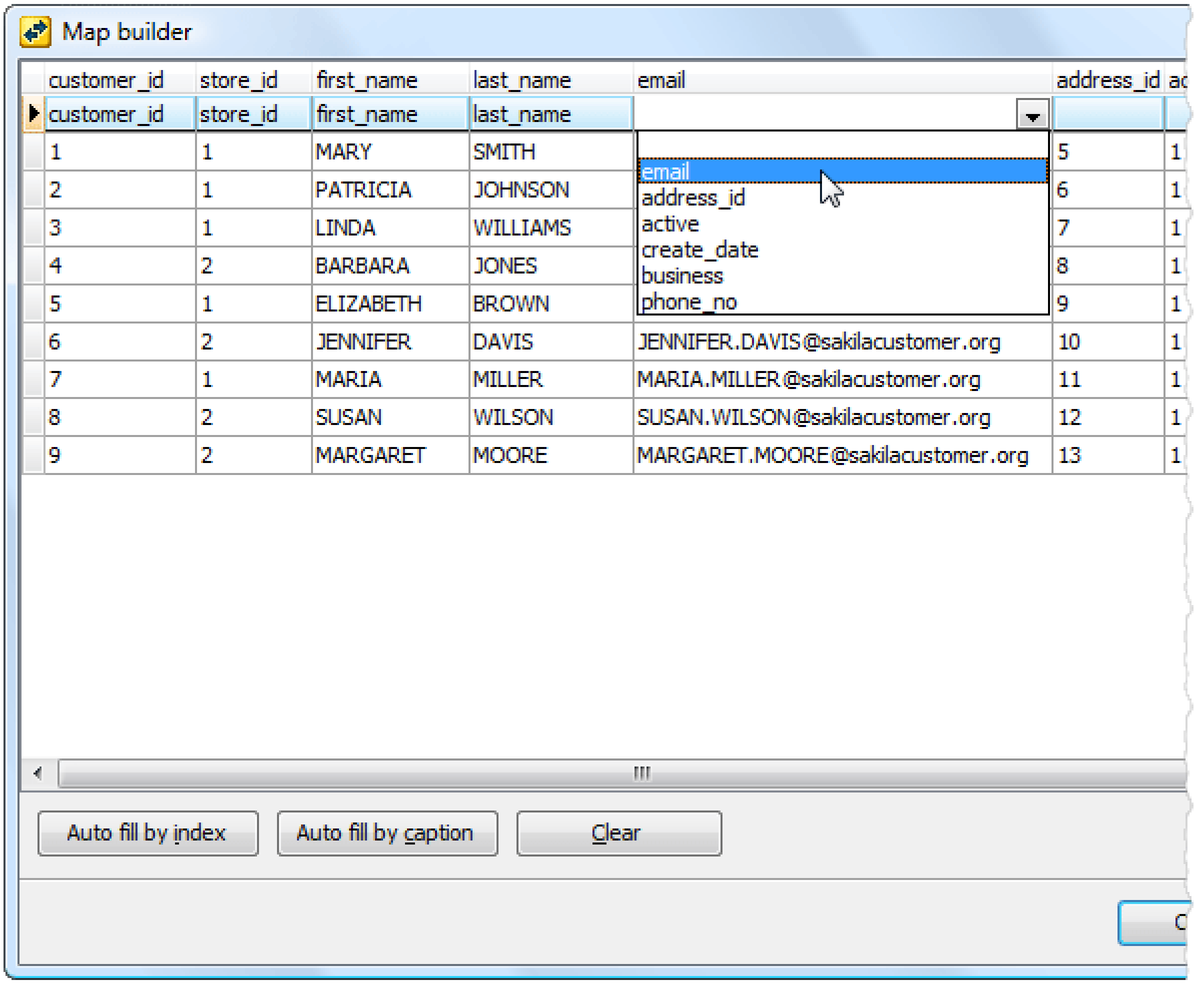


	Target field	Source column	Replacements	Empty values interpretation
1	film_id	A		
2	title	B		As Null
3	description	C		As Null
4	release_year	D		
5	language_id	E		
6	original_language_id	F		
7	rental_duration	G		
8	rental_rate	H		
9	length	I		
10	replacement_cost	J		
11	rating	K		As Null
12	last_update	L		
13	special_features	M		As Null
14	fulltext			As Null

☐ File contains column headers

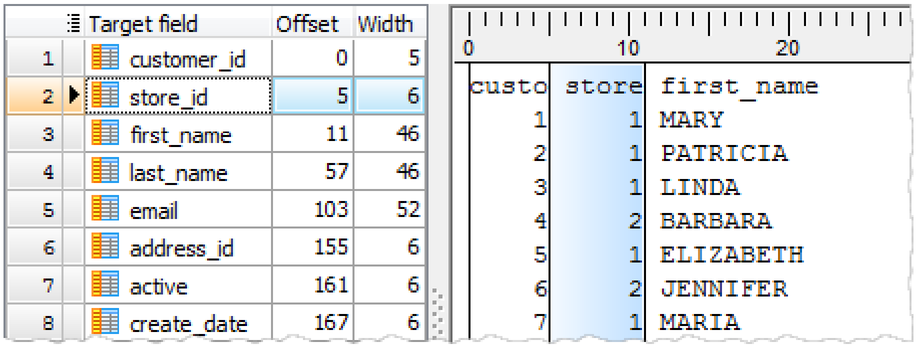
6.5.2.1 Map builder

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.



customer_id	store_id	first_name	last_name	email	address_id	ac
customer_id	store_id	first_name	last_name			
1	1	MARY	SMITH		5	1
2	1	PATRICIA	JOHNSON		6	1
3	1	LINDA	WILLIAMS		7	1
4	2	BARBARA	JONES		8	1
5	1	ELIZABETH	BROWN		9	1
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1

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

```
custo store first_name
    1     1 MARY
    2     1 PATRICIA
    3     1 LINDA
    4     2 BARBARA
    5     1 ELIZABETH
    6     2 JENNIFER
    7     1 MARIA
```

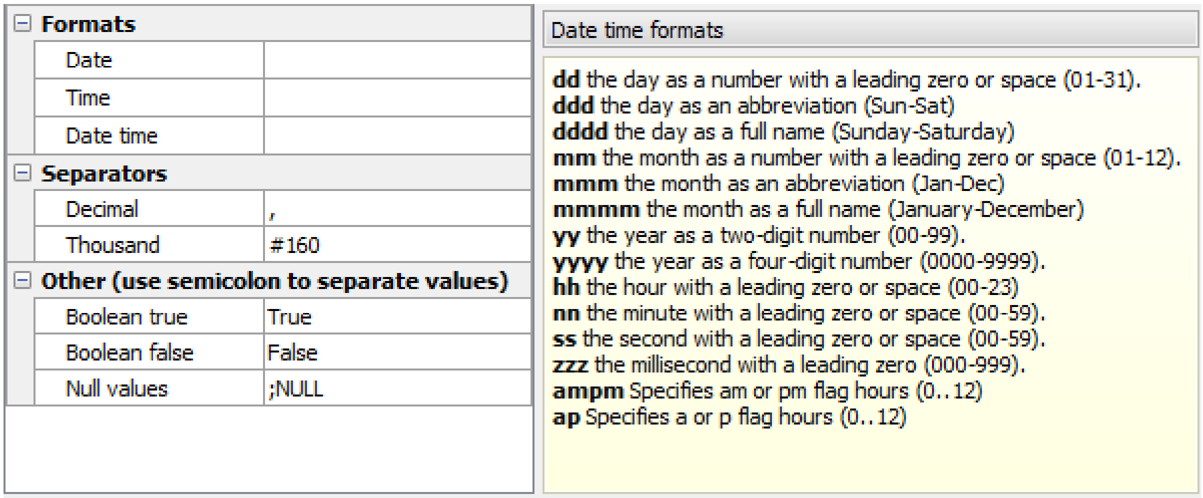
6.5.2.2 Data formats

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	mmmm dd
Jun 29, 2009	mmmmm dd, yyyy
Tue Jun 14 16:50:49	ddd mmm 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.



Formats

Date	
Time	
Date time	

Separators

Decimal	,
Thousand	#160

Other (use semicolon to separate values)

Boolean true	True
Boolean false	False
Null values	;NULL

Date time formats

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)

6.5.3 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 neseccary 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.

7 Database Tools

SQLite Code Factory provides a number of powerful tools for working with databases.

The following tools are available:

- **SQL Editor** 41
  Creates and executes SQL queries.

- **Visual Query Builder** 46
  Builds queries visually.

- **SQL Script Editor** 56
  Executes SQL scripts to the database.

- **BLOB Viewer** 94
  Displays a content of BLOB fields in different representations.

- **Diagram Viewer** 100
  Represents data from a table or a query as a diagram in various ways.

- **SQL Generator** 104
  Provides you with a set of simple SQL statements.

7.1 BLOB Viewer

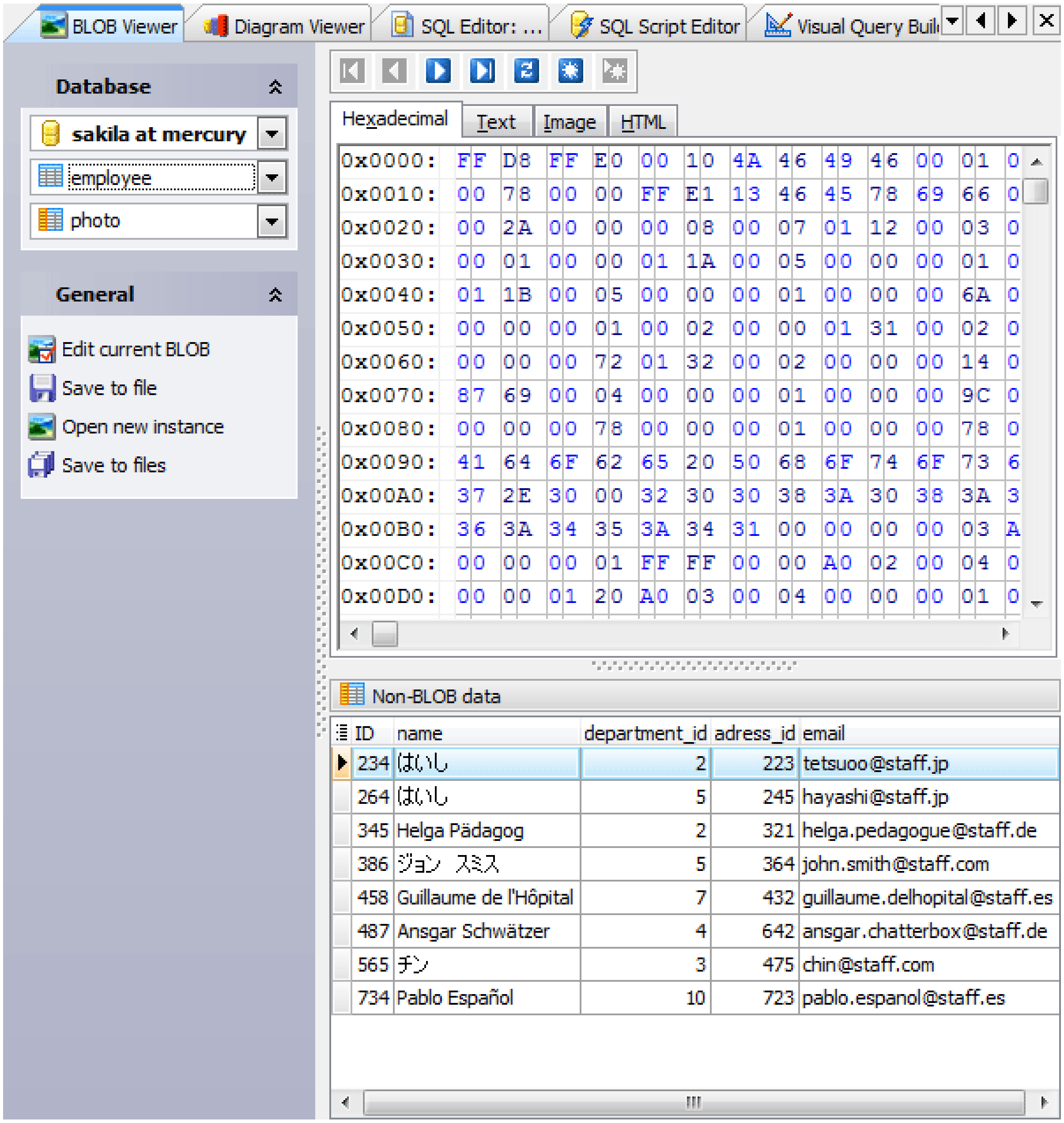
BLOB Viewer allows you to view the content of the BLOB fields in various representations.

- Viewing BLOB field as hexadecimal dump 94
- Viewing BLOB field as plain text 95
- Viewing BLOB field as graphical image 96
- Viewing BLOB field as HTML 97
- Viewing BLOB field as PDF 98

See also: BLOB Editor 70

7.1.1 Viewing as hexadecimal dump

The Hexadecimal panel allows you to view data in hexadecimal mode.

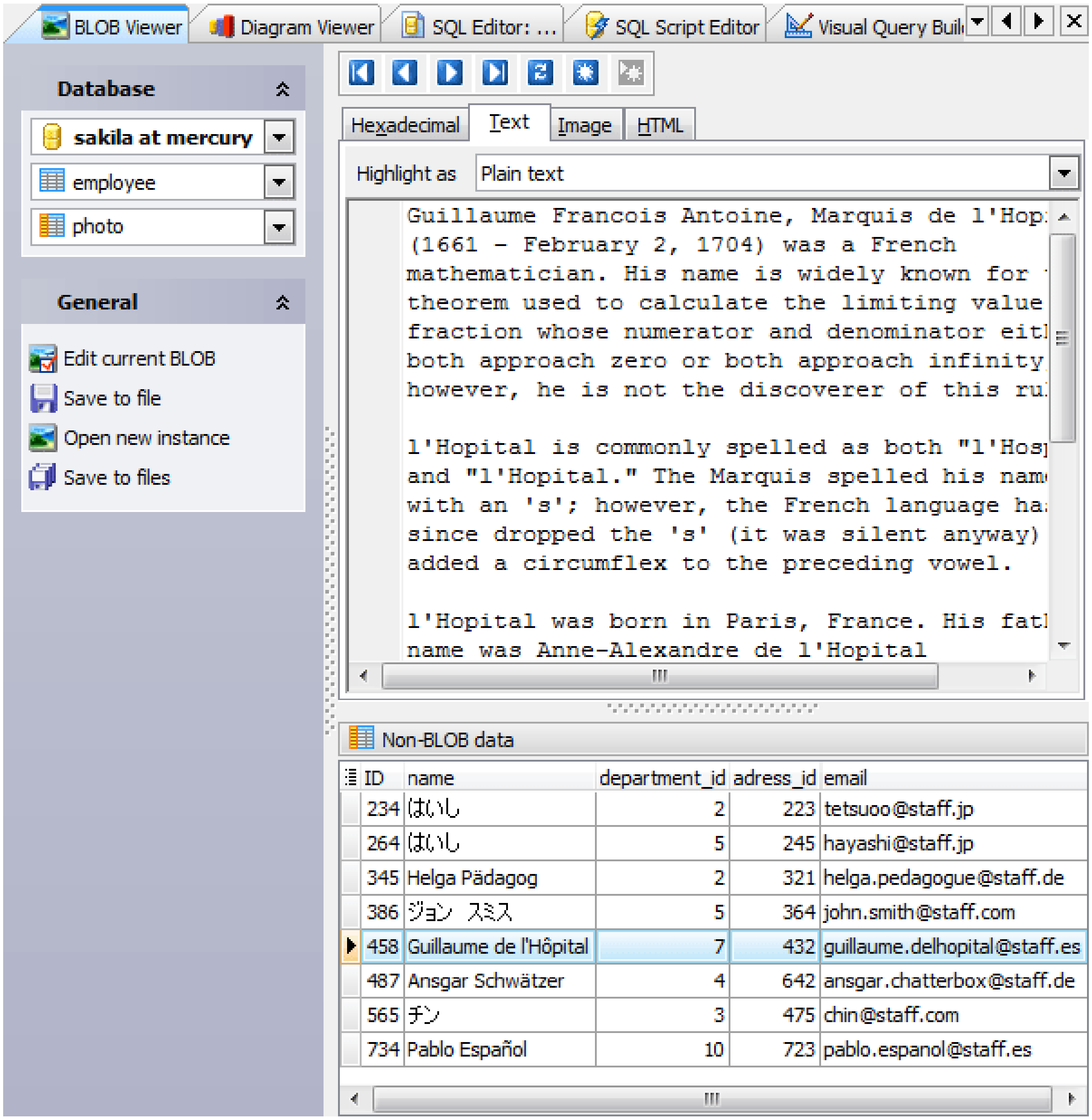


```
0x0000: FF D8 FF E0 00 10 4A 46 49 46 00 01 0
0x0010: 00 78 00 00 FF E1 13 46 45 78 69 66 0
0x0020: 00 2A 00 00 00 08 00 07 01 12 00 03 0
0x0030: 00 01 00 00 01 1A 00 05 00 00 00 01 0
0x0040: 01 1B 00 05 00 00 00 01 00 00 00 6A 0
0x0050: 00 00 00 01 00 02 00 00 01 31 00 02 0
0x0060: 00 00 00 72 01 32 00 02 00 00 00 14 0
0x0070: 87 69 00 04 00 00 00 01 00 00 00 9C 0
0x0080: 00 00 00 78 00 00 00 01 00 00 00 78 0
0x0090: 41 64 6F 62 65 20 50 68 6F 74 6F 73 6
0x00A0: 37 2E 30 00 32 30 30 38 3A 30 38 3A 3
0x00B0: 36 3A 34 35 3A 34 31 00 00 00 00 03 A
0x00C0: 00 00 00 01 FF FF 00 00 A0 02 00 04 0
0x00D0: 00 00 01 20 A0 03 00 04 00 00 00 01 0
```

ID	name	department_id	adress_id	email
234	はいし	2	223	tetsuoo@staff.jp
264	はいし	5	245	hayashi@staff.jp
345	Helga Pädagog	2	321	helga.pedagogue@staff.de
386	ジョン　スミス	5	364	john.smith@staff.com
458	Guillaume de l'Hôpital	7	432	guillaume.delhopital@staff.es
487	Ansgar Schwätzer	4	642	ansgar.chatterbox@staff.de
565	チン	3	475	chin@staff.com
734	Pablo Español	10	723	pablo.espanol@staff.es

7.1.2 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.



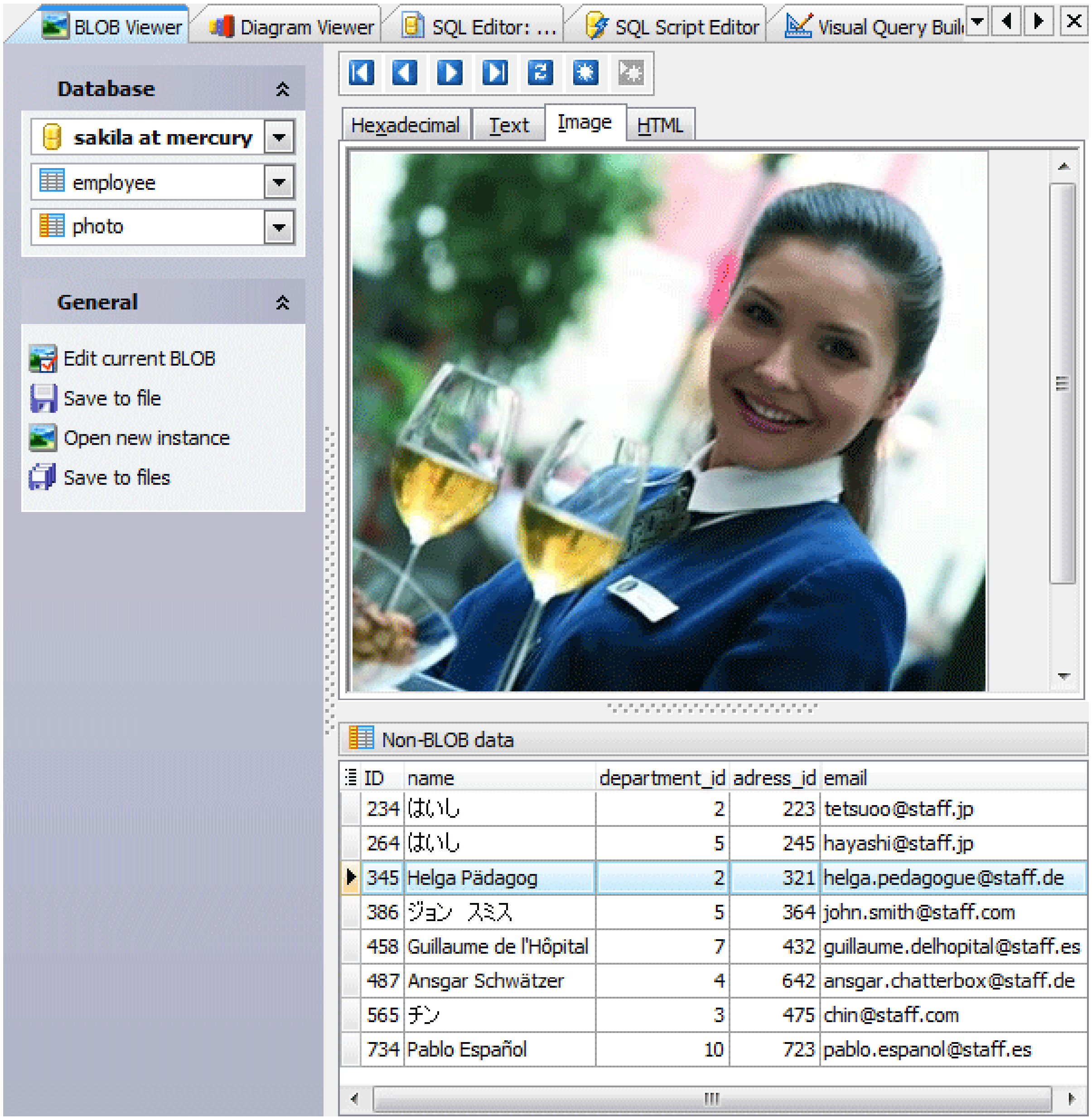
Highlight as: Plain text

```
Guillaume Francois Antoine, Marquis de l'Hop
(1661 - February 2, 1704) was a French
mathematician. His name is widely known for
theorem used to calculate the limiting value
fraction whose numerator and denominator eit
both approach zero or both approach infinity
however, he is not the discoverer of this ru

l'Hopital is commonly spelled as both "l'Hos
and "l'Hopital." The Marquis spelled his nam
with an 's'; however, the French language ha
since dropped the 's' (it was silent anyway)
added a circumflex to the preceding vowel.

l'Hopital was born in Paris, France. His fat
name was Anne-Alexandre de l'Hopital
```

ID	name	department_id	adress_id	email
234	はいし	2	223	tetsuoo@staff.jp
264	はいし	5	245	hayashi@staff.jp
345	Helga Pädagog	2	321	helga.pedagogue@staff.de
386	ジョン スミス	5	364	john.smith@staff.com
458	Guillaume de l'Hôpital	7	432	guillaume.delhopital@staff.es
487	Ansgar Schwätzer	4	642	ansgar.chatterbox@staff.de
565	チン	3	475	chin@staff.com
734	Pablo Español	10	723	pablo.espanol@staff.es

7.1.3 Viewing as image

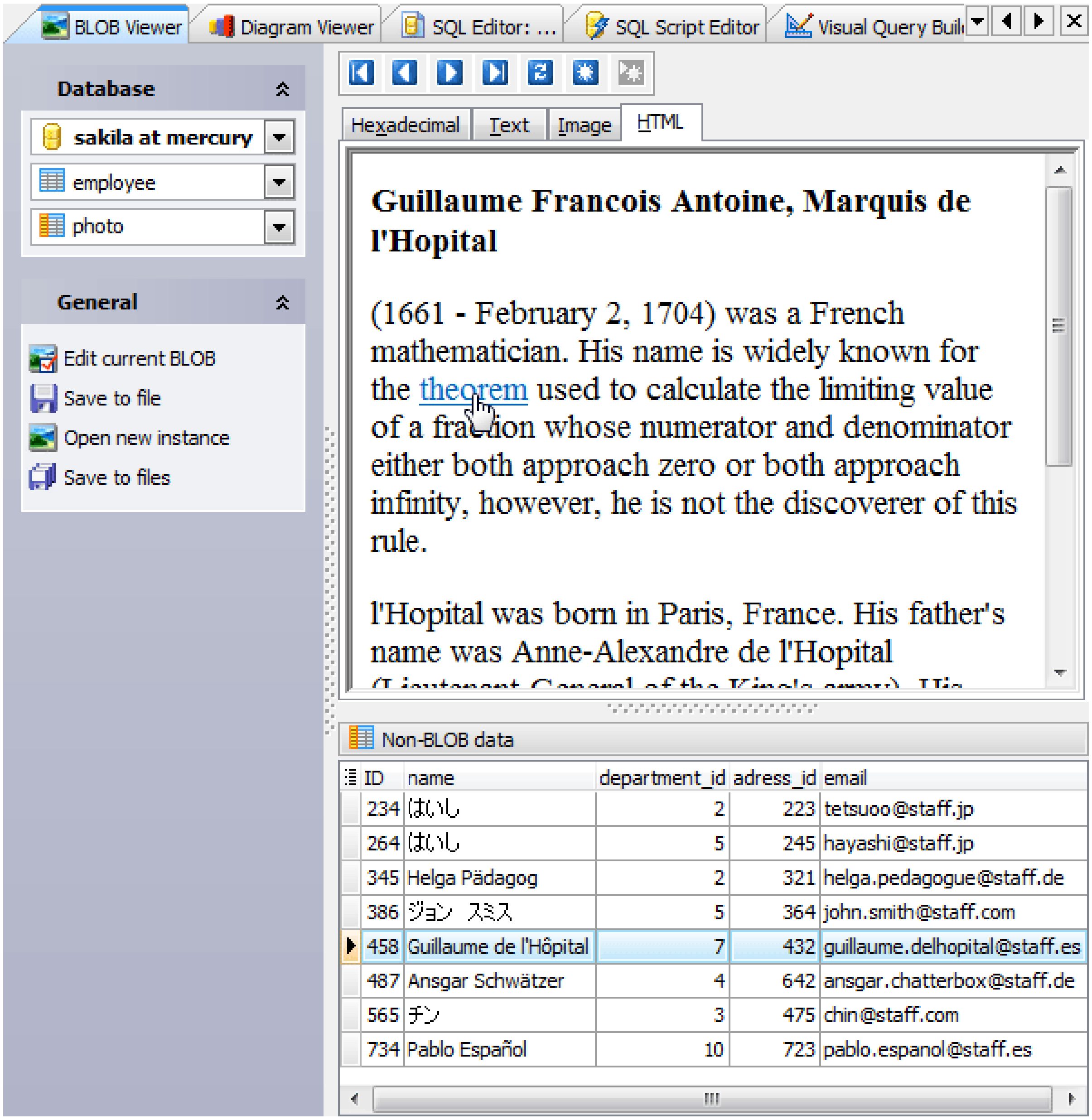
The Image panel displays field data as image.



ID	name	department_id	adress_id	email
234	はいし	2	223	tetsuoo@staff.jp
264	はいし	5	245	hayashi@staff.jp
345	Helga Pädagog	2	321	helga.pedagogue@staff.de
386	ジョン　スミス	5	364	john.smith@staff.com
458	Guillaume de l'Hôpital	7	432	guillaume.delhopital@staff.es
487	Ansgar Schwätzer	4	642	ansgar.chatterbox@staff.de
565	チン	3	475	chin@staff.com
734	Pablo Español	10	723	pablo.espanol@staff.es

7.1.4 Viewing as HTML

The HTML panel displays field data as HTML.



Guillaume Francois Antoine, Marquis de l'Hopital

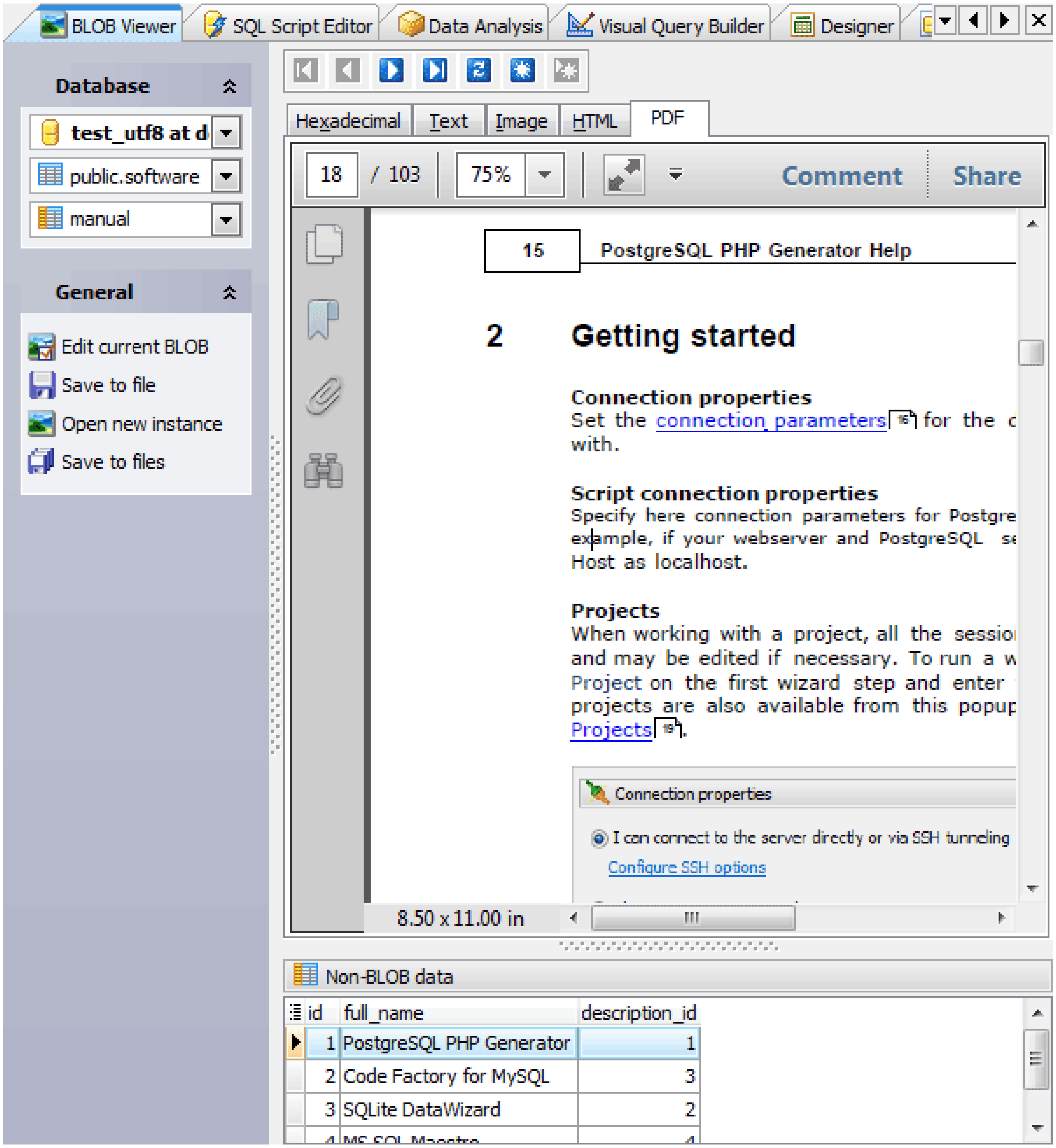
(1661 - February 2, 1704) was a French mathematician. His name is widely known for the theorem used to calculate the limiting value of a fraction whose numerator and denominator either both approach zero or both approach infinity, however, he is not the discoverer of this rule.

l'Hopital was born in Paris, France. His father's name was Anne-Alexandre de l'Hopital

ID	name	department_id	adress_id	email
234	はいし	2	223	tetsuoo@staff.jp
264	はいし	5	245	hayashi@staff.jp
345	Helga Pädagog	2	321	helga.pedagogue@staff.de
386	ジョン スミス	5	364	john.smith@staff.com
458	Guillaume de l'Hôpital	7	432	guillaume.delhopital@staff.es
487	Ansgar Schwätzer	4	642	ansgar.chatterbox@staff.de
565	チン	3	475	chin@staff.com
734	Pablo Español	10	723	pablo.espanol@staff.es

7.1.5 Viewing as PDF

The PDF panel allows you to browse PDF data stored in the database.



15 PostgreSQL PHP Generator Help

2 Getting started

Connection properties
Set the connection parameters for the c
with.

Script connection properties
Specify here connection parameters for Postgre
example, if your webserver and PostgreSQL se
Host as localhost.

Projects
When working with a project, all the sessio
and may be edited if necessary. To run a w
Project on the first wizard step and enter
projects are also available from this popup
Projects.

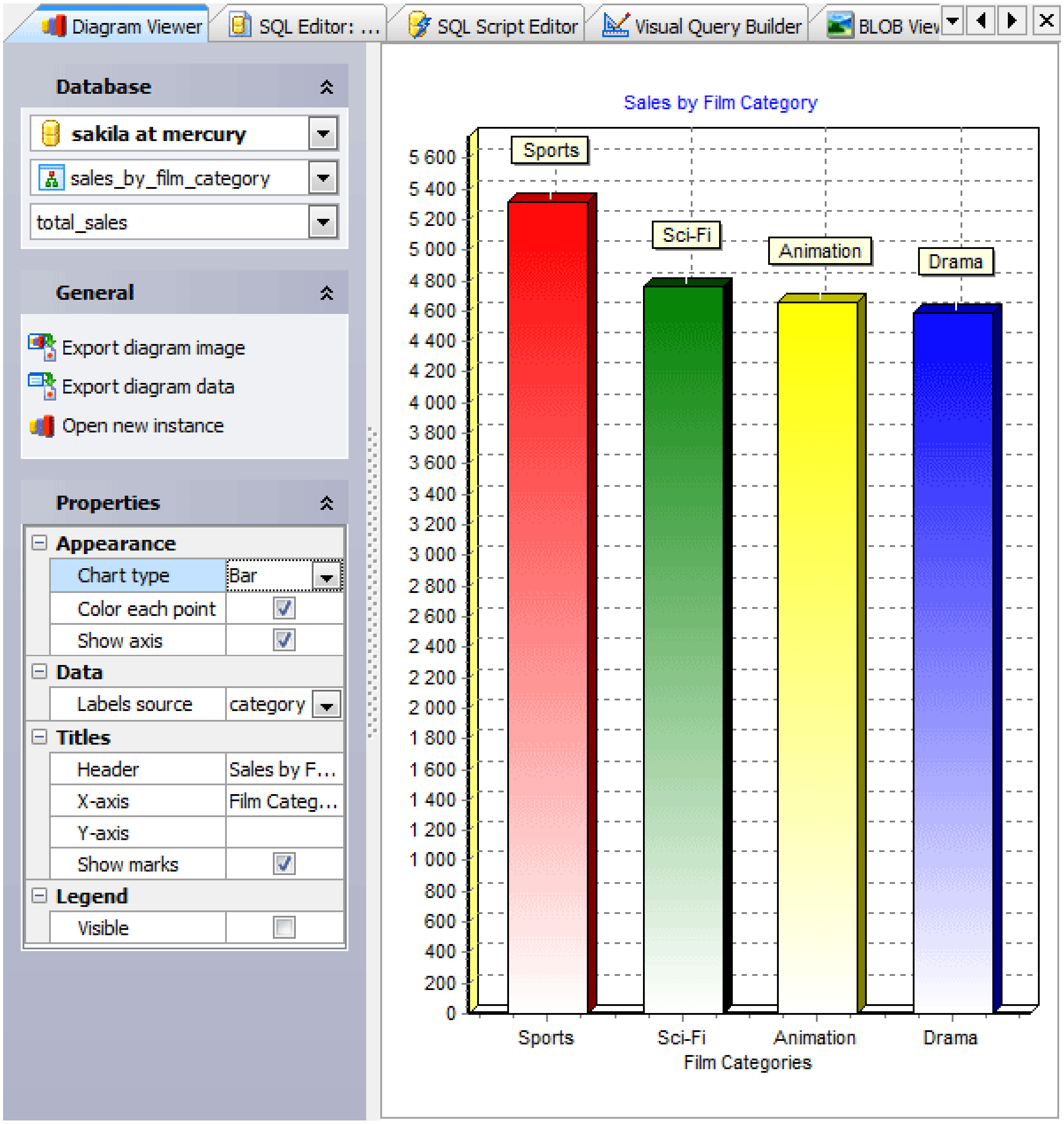
Connection properties
● I can connect to the server directly or via SSH tunneling
Configure SSH options

id	full_name	description_id
1	PostgreSQL PHP Generator	1
2	Code Factory for MySQL	3
3	SQLite DataWizard	2
4	MS SQL Maestro	4

7.2 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 102 and the Export diagram data features implemented, with a lot of formats supported.

- Customizing diagram options 101
- Exporting diagram as a graphical image 102

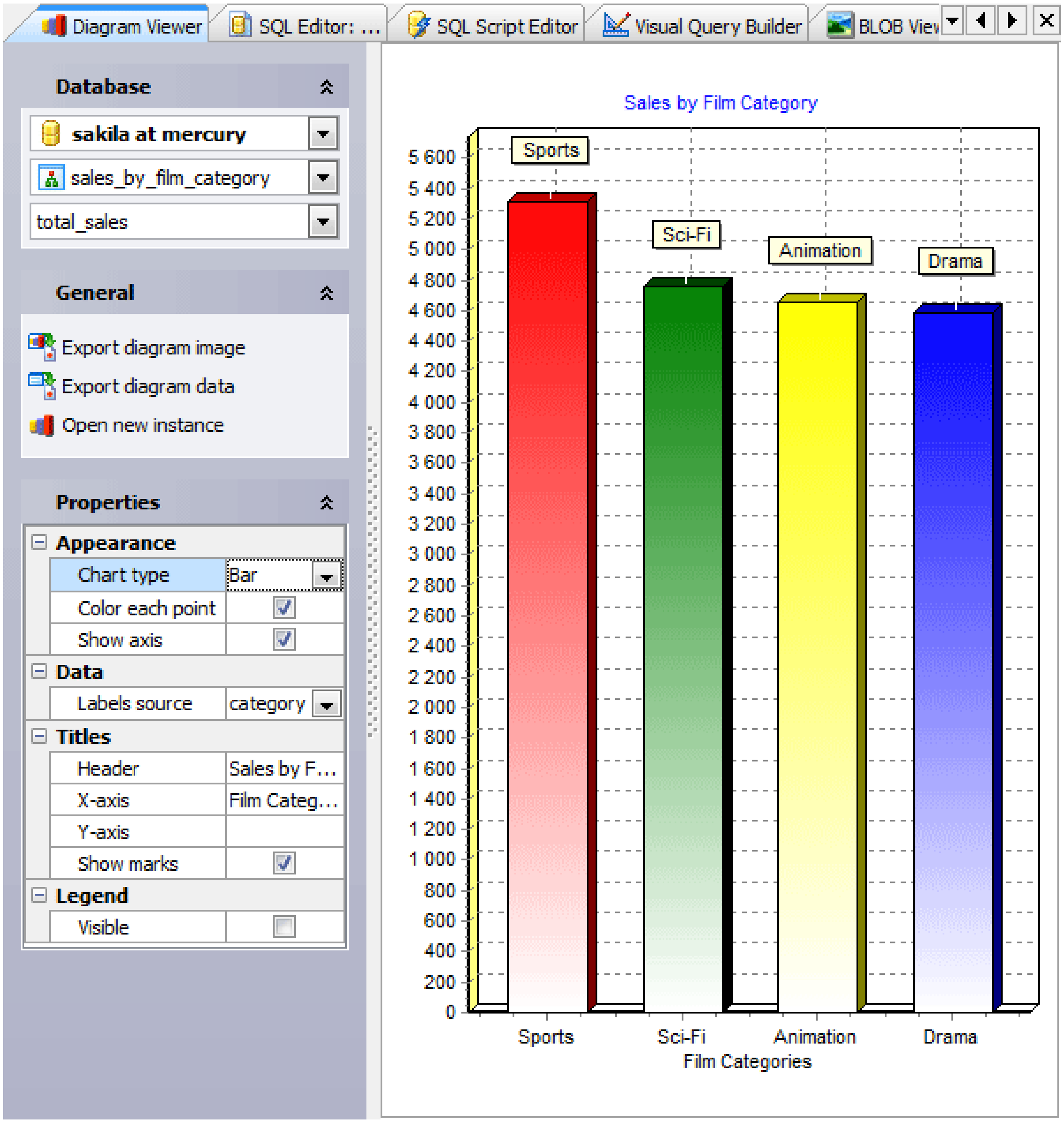


sakila at mercury
sales_by_film_category
total_sales

Appearance
Chart type: Bar
Color each point: ☑
Show axis: ☑
Data
Labels source: category
Titles
Header: Sales by F...
X-axis: Film Categ...
Y-axis:
Show marks: ☑
Legend
Visible: ☐

7.2.1 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.



sakila at mercury
sales_by_film_category
total_sales

Appearance	
Chart type	Bar
Color each point	☑
Show axis	☑
Data	
Labels source	category
Titles	
Header	Sales by F...
X-axis	Film Categ...
Y-axis	
Show marks	☑
Legend	
Visible	☐

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 point marks .

Titles
Contains properties for defining titles for different parts of the diagram:

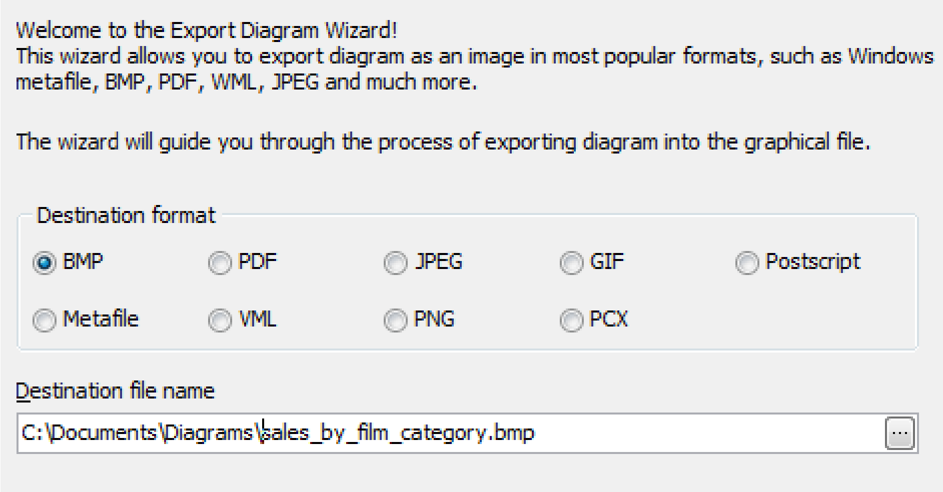
- Header - defines the title appeared on the top of the diagram
- X-axis and Y-axis - define the titles for diagram axis
- Show marks - defines if the diagram point marks are visible or not

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.

7.2.2 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.

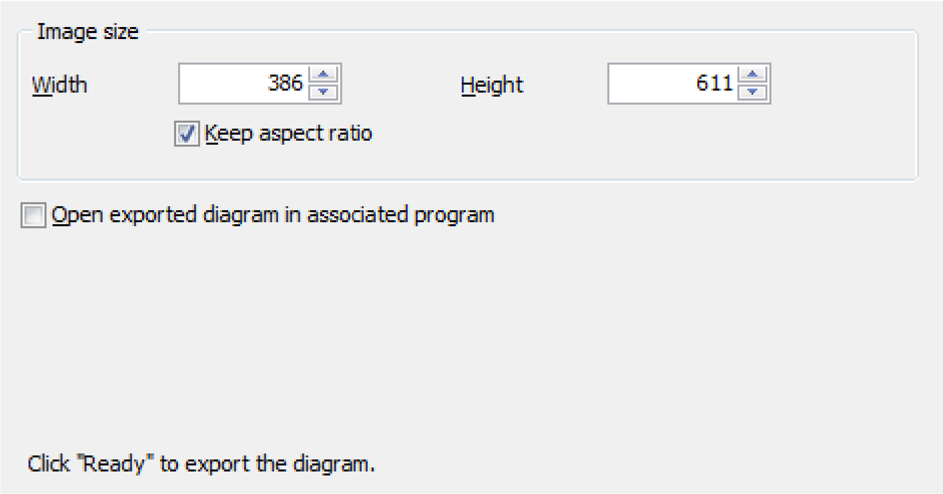


Image size

Width 386

Height 611

☑ Keep aspect ratio

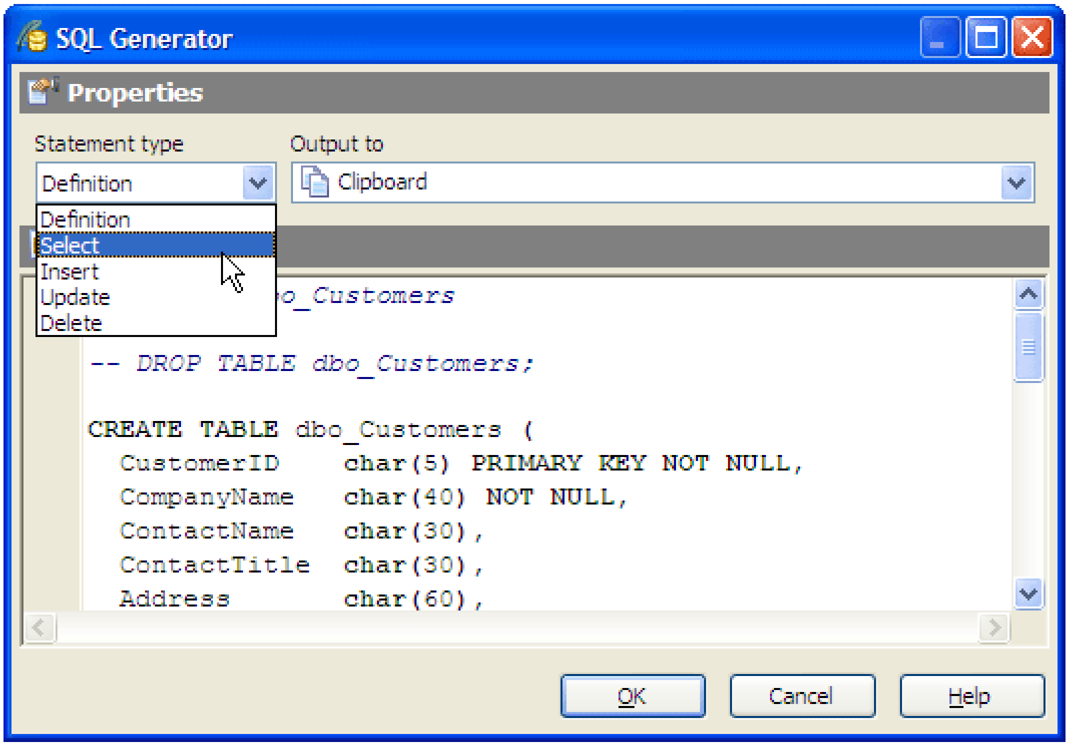
☐ Open exported diagram in associated program

Click "Ready" to export the diagram.

7.3 SQL Generator

Among other features SQLite Code Factory 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.



Statement type: Definition

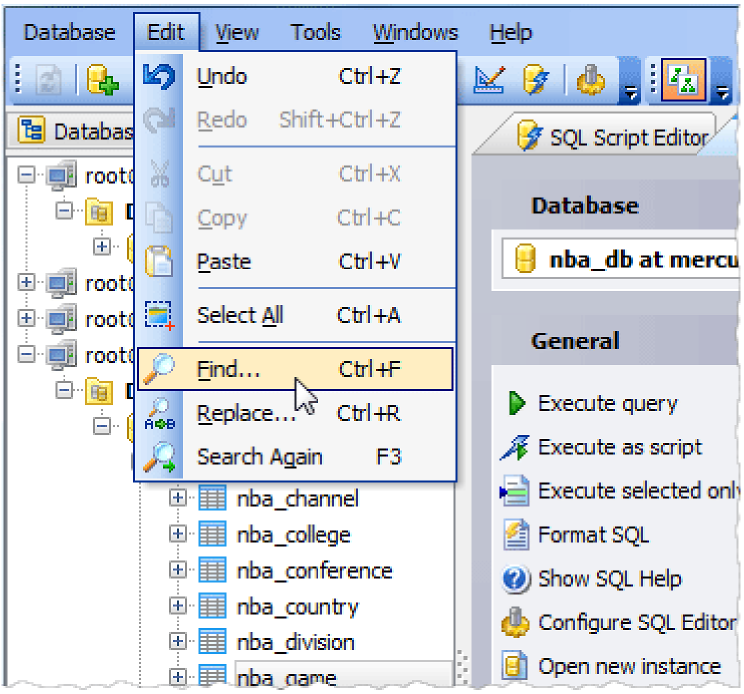
Output to: Clipboard

```
o_Customers

-- DROP TABLE dbo_Customers;

CREATE TABLE dbo_Customers (
  CustomerID    char(5) PRIMARY KEY NOT NULL,
  CompanyName   char(40) NOT NULL,
  ContactName   char(30),
  ContactTitle  char(30),
  Address       char(60),
```

7.4 Dialogs

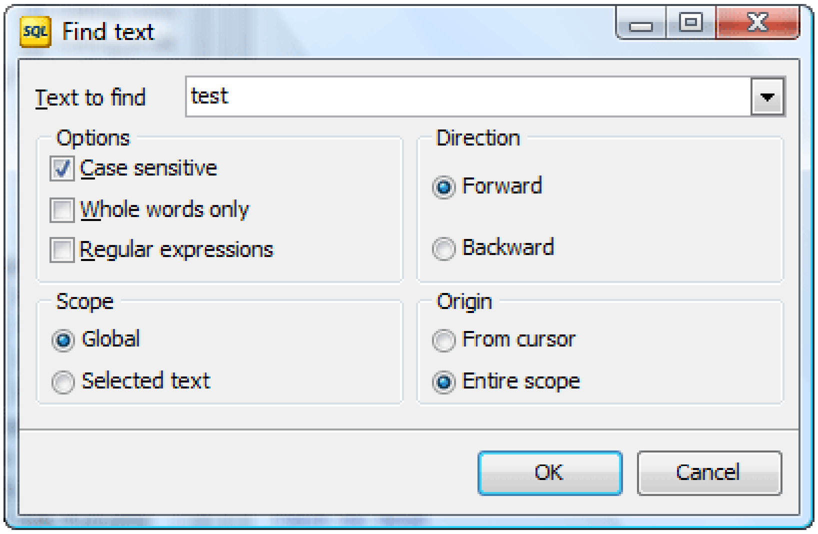
SQLite Code Factory provides two dialogs for searching and replacing text in the editor areas of the database tools. Both of them are available through the popup menu of the editor area.

- Find Text dialog 105
- Replace Text dialog 106



7.4.1 Find Text dialog

The Find Text dialog is provided for quick search for certain text.


Find text

Text to find: test

Options
☑ Case sensitive
☐ Whole words only
☐ Regular expressions

Direction
● Forward
○ Backward

Scope
● Global
○ Selected text

Origin
○ From cursor
● Entire scope

OK Cancel

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.

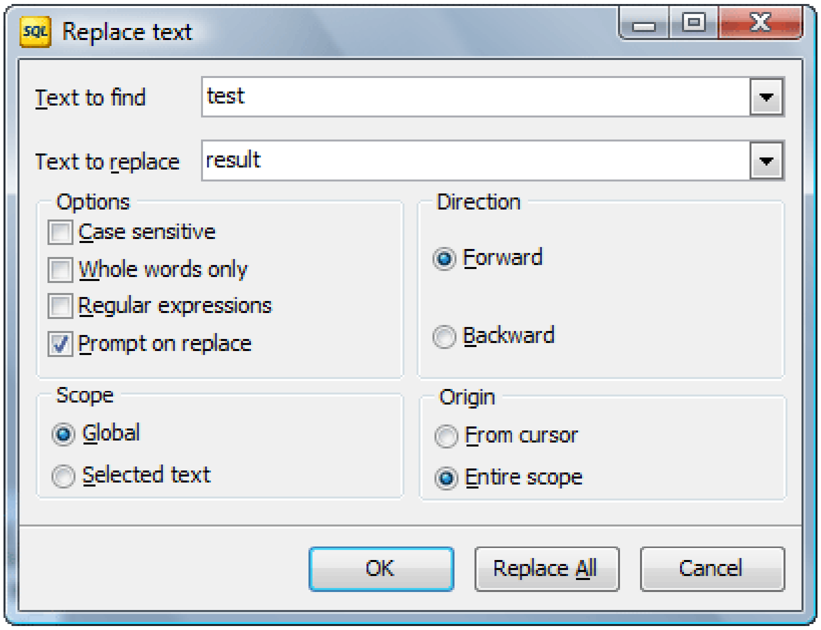
Selected text
Searches within the selected text only, in the direction specified by the Direction setting. You can use the mouse or block commands to select a block of text.

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.

7.4.2 Replace Text dialog

The Replace Text dialog is provided for searching and replacing text in the editor window.



Text to find: test

Text to replace: result

Options
- ☐ Case sensitive
- ☐ Whole words only
- ☐ Regular expressions
- ☑ Prompt on replace

Direction
- ● Forward
- ○ Backward

Scope
- ● Global
- ○ Selected text

Origin
- ○ From cursor
- ● Entire scope

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.

8 Options

SQLite Code Factory 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** 110
  General SQLite Code Factory options: environment style, confirmations, window restrictions, explorer tree, SQL Editor, Visual Query Builder, etc.
- **Editors & Viewers** 127
  Customizing of all the SQL editors - SQL Editor, SQL Script Editor, etc.
- **Appearance** 135
  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 144 for details).

It is a good idea to check through these settings before you start working with SQLite Code Factory. You may be surprised at all the things you can adjust and configure!

8.1 Application

The Application section allows you to customize common rules of SQLite Code Factory behavior. The section consists of several tab; follow the links to find out more about each of them.

- Preferences 110
- Confirmations 111
- Tools 112
  - Explorer 113
  - SQL Editor 114
  - SQL Script Editor 115
  - Query Builder 116
  - BLOB Viewer 118
  - Export data 119
- Data Grid 120
  - Colors 123
  - Formats 123

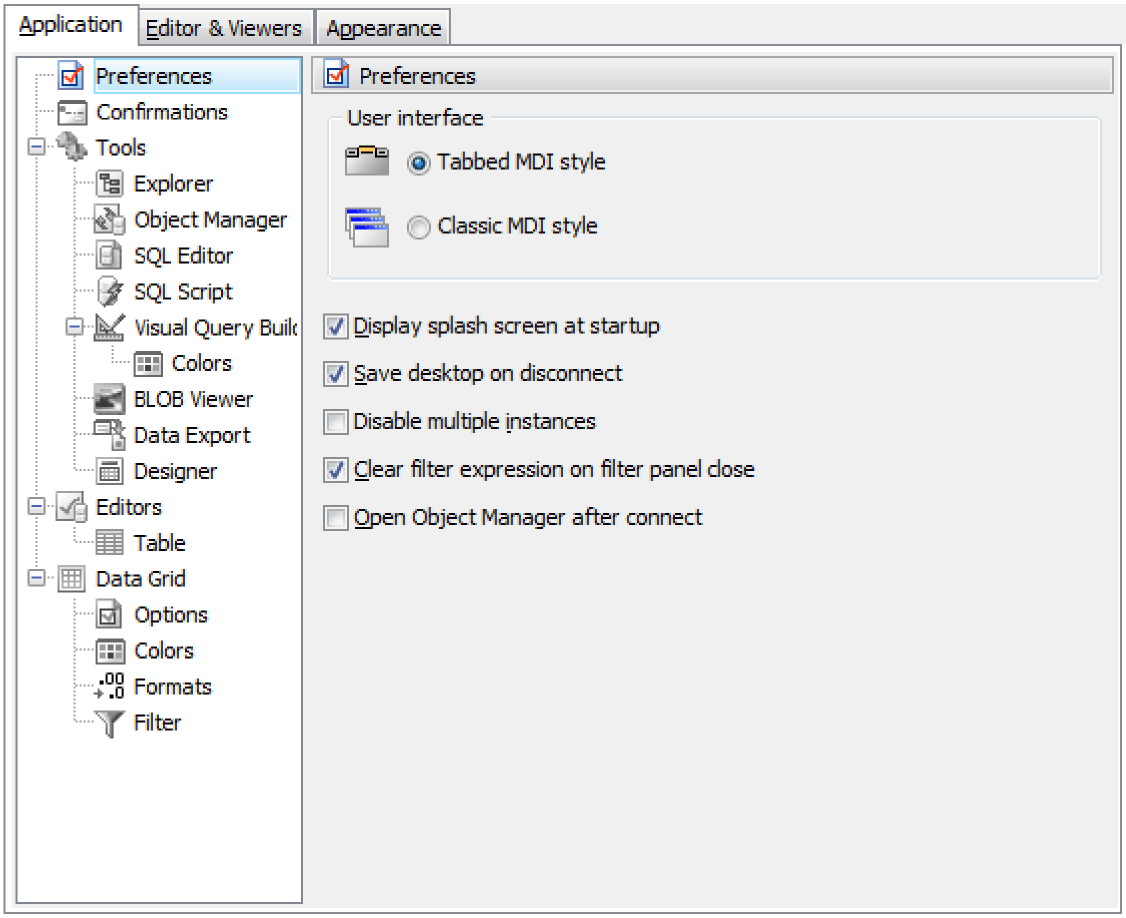
8.1.1 Preferences

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 SQLite Code Factory 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 SQLite Code Factory.



● Tabbed MDI style

○ Classic MDI style

☑ Display splash screen at startup

☑ Save desktop on disconnect

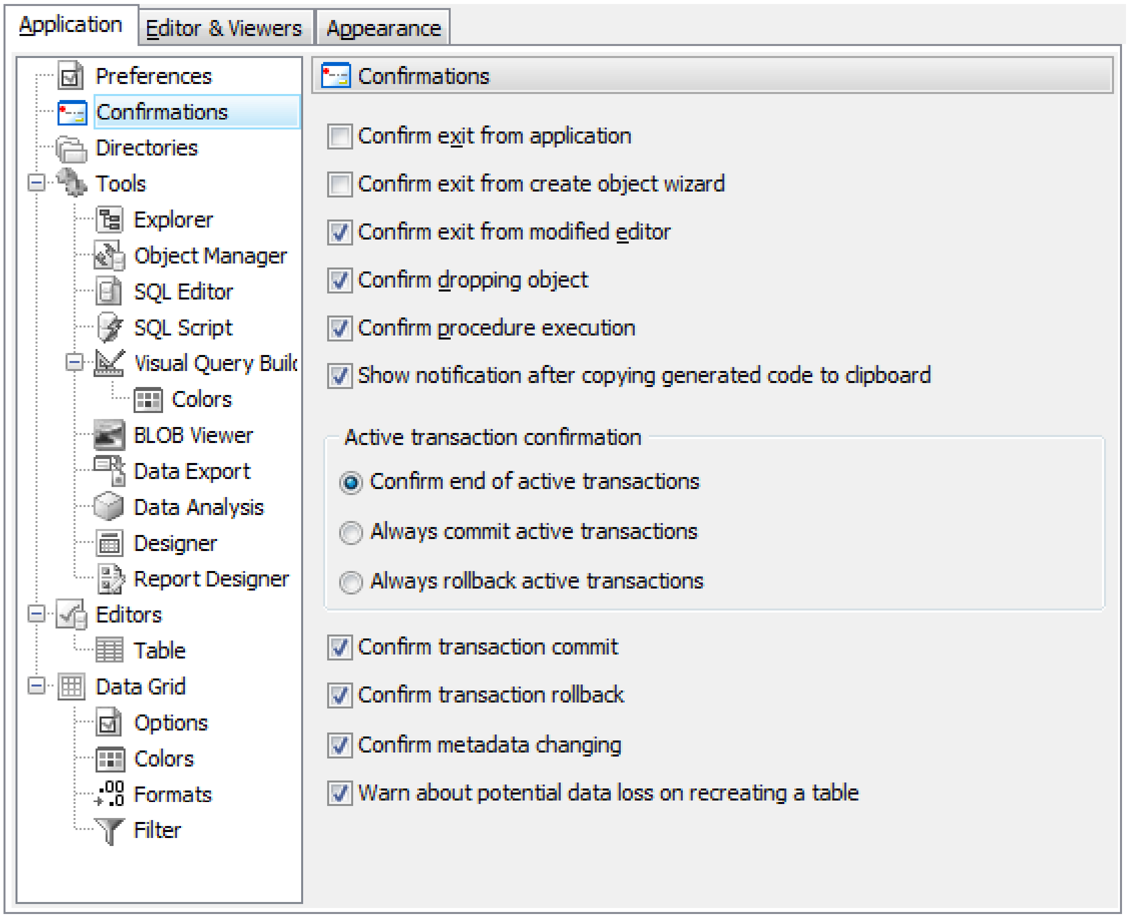
☐ Disable multiple instances

☑ Clear filter expression on filter panel close

☐ Open Object Manager after connect

8.1.2 Confirmations

Use this tab to manage application confirmations.



Confirmations

☐ Confirm exit from application
☐ Confirm exit from create object wizard
☑ Confirm exit from modified editor
☑ Confirm dropping object
☑ Confirm procedure execution
☑ Show notification after copying generated code to clipboard

Active transaction confirmation
● Confirm end of active transactions
○ Always commit active transactions
○ Always rollback active transactions

☑ Confirm transaction commit
☑ Confirm transaction rollback
☑ Confirm metadata changing
☑ Warn about potential data loss on recreating a table

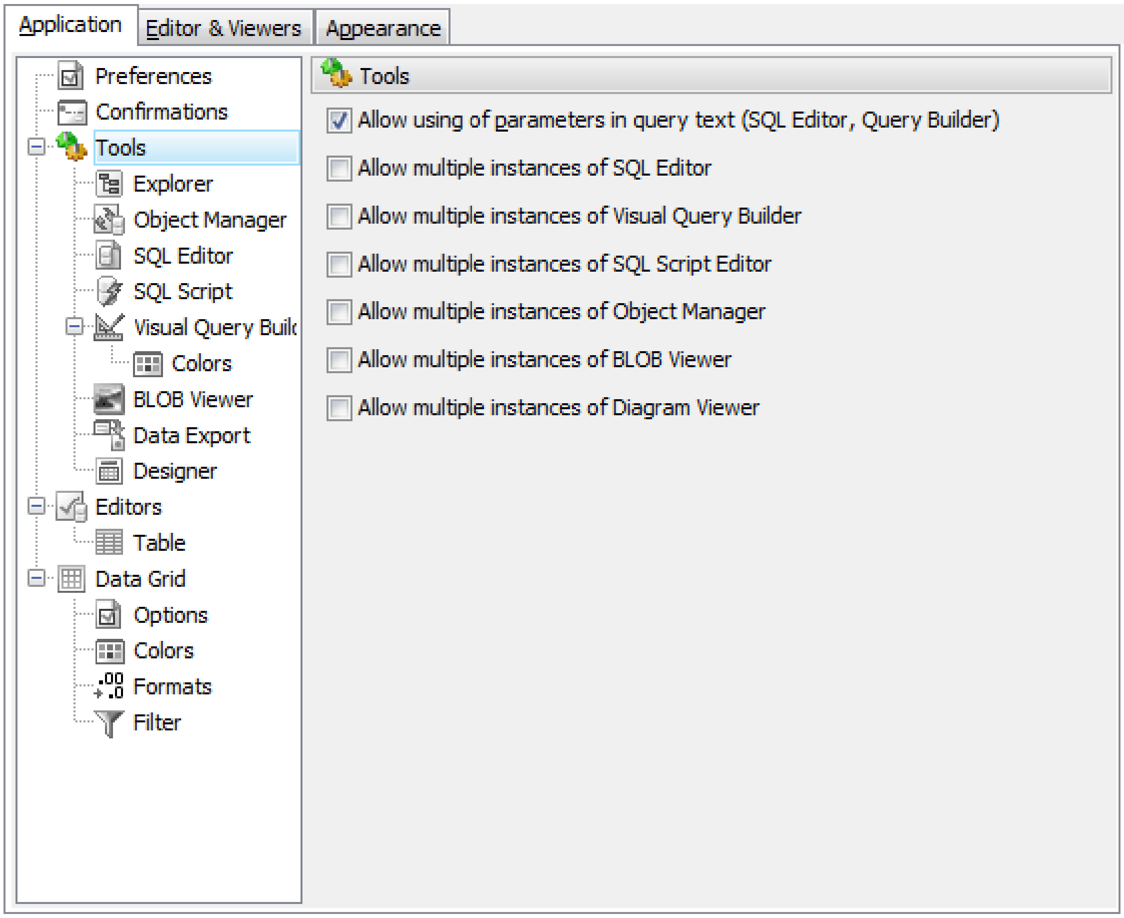
☑ Confirm exit from application
If this option is checked, the program requires confirmation when you want to exit <%PRODUCT_NAME%.

☑ Transaction confirmation
Select whether you will be prompted to commit or rollback active transaction or SQLite Code Factory will commit or rollback transactions without asking.

☑ Confirm metadata changing
If this option is checked, the program requires confirmation for changing metadata.

8.1.3 Tools

Below you will find a detailed decryption of the following tools options.



☑ Allow using of parameters in query text (SQL Editor, Query Builder)
☐ Allow multiple instances of SQL Editor
☐ Allow multiple instances of Visual Query Builder
☐ Allow multiple instances of SQL Script Editor
☐ Allow multiple instances of Object Manager
☐ Allow multiple instances of BLOB Viewer
☐ Allow multiple instances of Diagram Viewer

☑ Allow using of parameters in query text
Check this option to be able to use query parameters in SQL Editor 41 and Visual Query Builder 46.

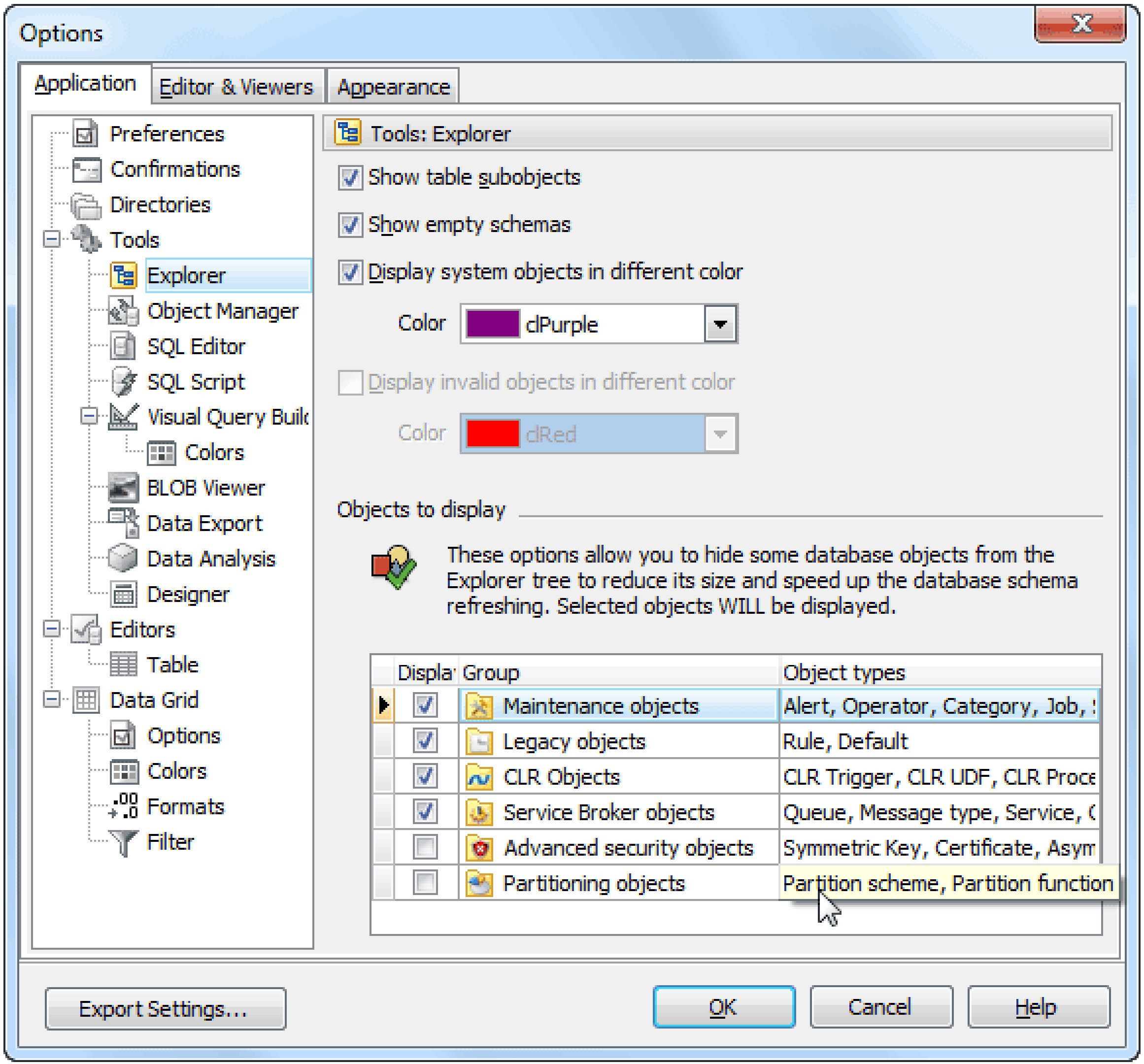
☑ Allow multiple instances of SQL Editor
Check this option to be able to use multiple instances of SQL Editor 41 simultaneously.

☑ Allow multiple instances of Visual Query Builder
Check this option to be able to use multiple instances of Visual Query Builder 46 simultaneously.

☑ Allow multiple instances of SQL Script Editor
Check this option to be able to use multiple instances of SQL Script Editor 56 simultaneously.

8.1.3.1 Explorer

Below you will find a detailed decryption of the following explorer options.



Tools: Explorer

☑ Show table subobjects

☑ Show empty schemas

☑ Display system objects in different color

Color: clPurple

☐ Display invalid objects in different color

Color: clRed

Objects to display

These options allow you to hide some database objects from the Explorer tree to reduce its size and speed up the database schema refreshing. Selected objects WILL be displayed.

Displa	Group	Object types
☑	Maintenance objects	Alert, Operator, Category, Job,
☑	Legacy objects	Rule, Default
☑	CLR Objects	CLR Trigger, CLR UDF, CLR Proce
☑	Service Broker objects	Queue, Message type, Service,
☐	Advanced security objects	Symmetric Key, Certificate, Asym
☐	Partitioning objects	Partition scheme, Partition function

☑ Show table subobjects

Shows/hides table subobjects (fields and indexes) in the explorer tree.

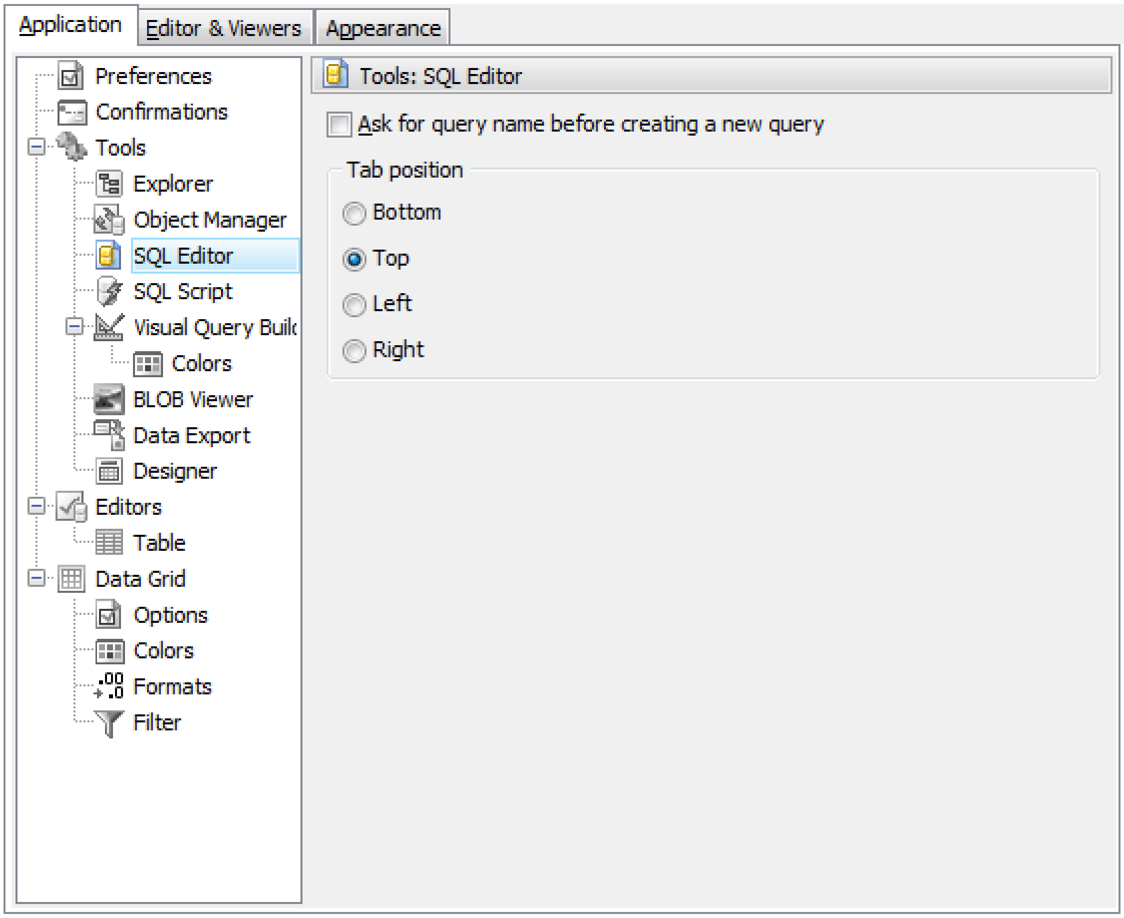
☑ Display system objects in different color

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.

8.1.3.2 SQL Editor

Below you will find a detailed decryption of the following SQL Editor options.



Tools: SQL Editor

☐ Ask for query name before creating a new query

Tab position

○ Bottom
● Top
○ Left
○ Right

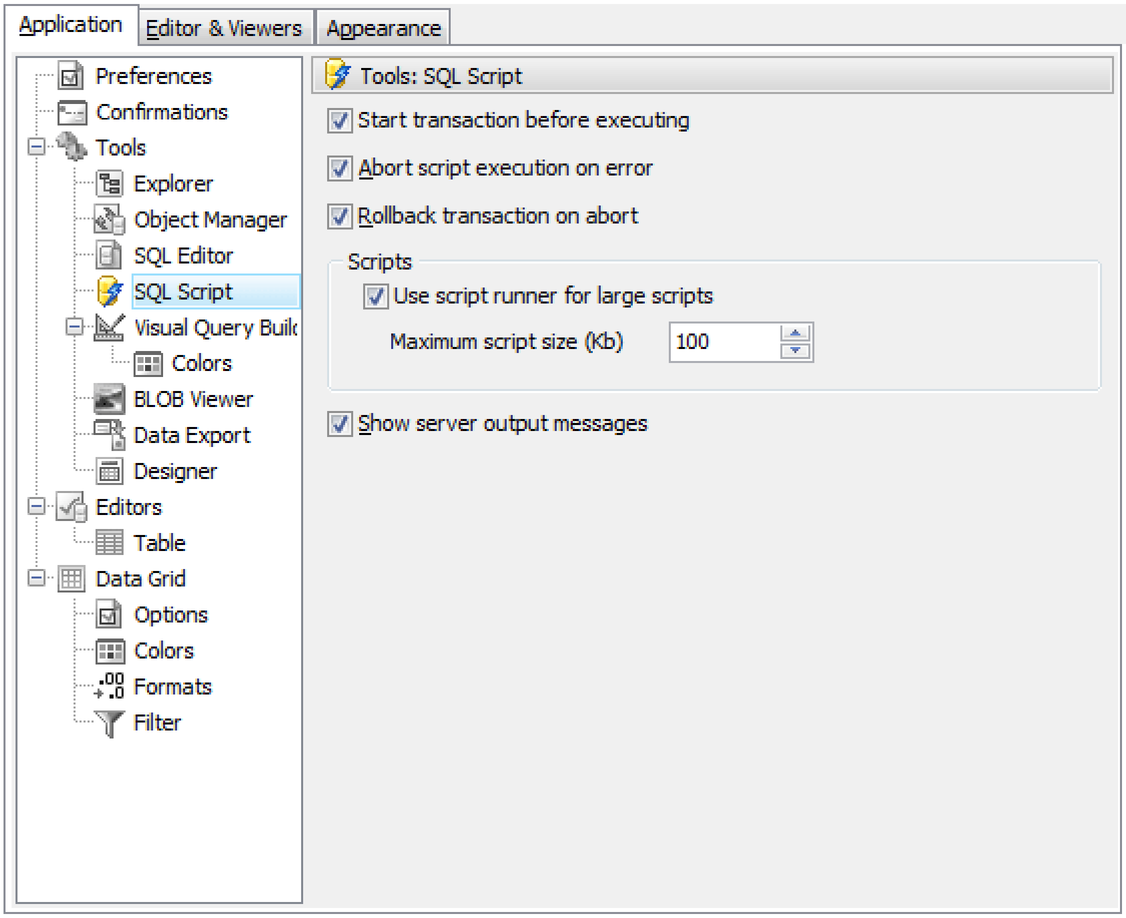
☑ Ask for query name before creating a new query
If this option is checked, SQL Editor 41 asks for a query name each time you create a new query.

☑ 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.

8.1.3.3 SQL Script Editor

Below you will find a detailed decryption of the following SQL Script Editor options.



Tools: SQL Script

☑ Start transaction before executing

☑ Abort script execution on error

☑ Rollback transaction on abort

Scripts

☑ Use script runner for large scripts

Maximum script size (Kb) 100

☑ Show server output messages

☑ Abort script execution on error
If this option is checked, script execution aborts when an error occurs.

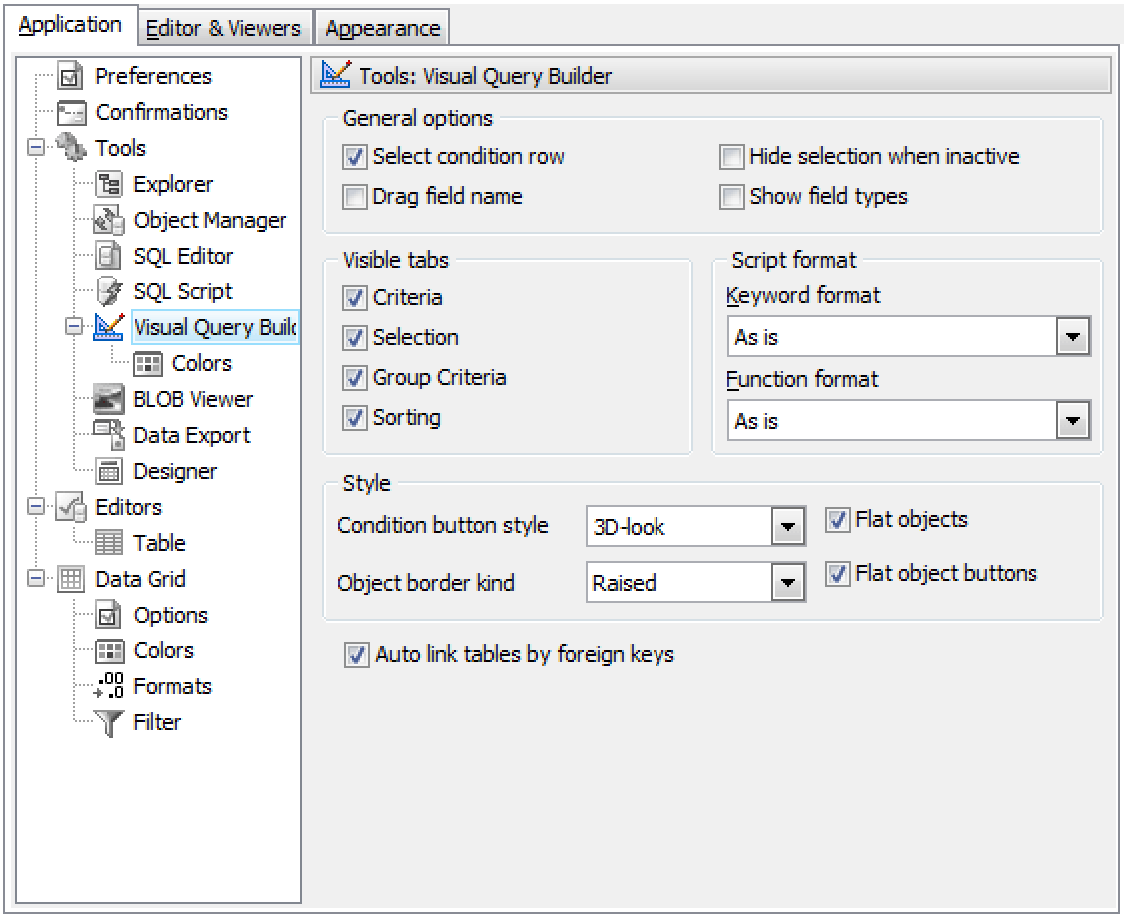
☑ Rollback transaction on abort
This option evokes automatic rollback on script execution abort.

☑ 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.

8.1.3.4 Query Builder

Below you will find a detailed decryption of the following Query Builder options.



☑ Select condition row
☐ Drag field name
☐ Hide selection when inactive
☐ Show field types

☑ Criteria
☑ Selection
☑ Group Criteria
☑ Sorting

Keyword format: As is
Function format: As is

Condition button style: 3D-look
Object border kind: Raised
☑ Flat objects
☑ Flat object buttons

☑ Auto link tables by foreign keys

☑ Select condition row
Displays the selected condition in different row on the Criteria and Grouping Criteria tabs of Visual Query Builder 46.

☑ 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.

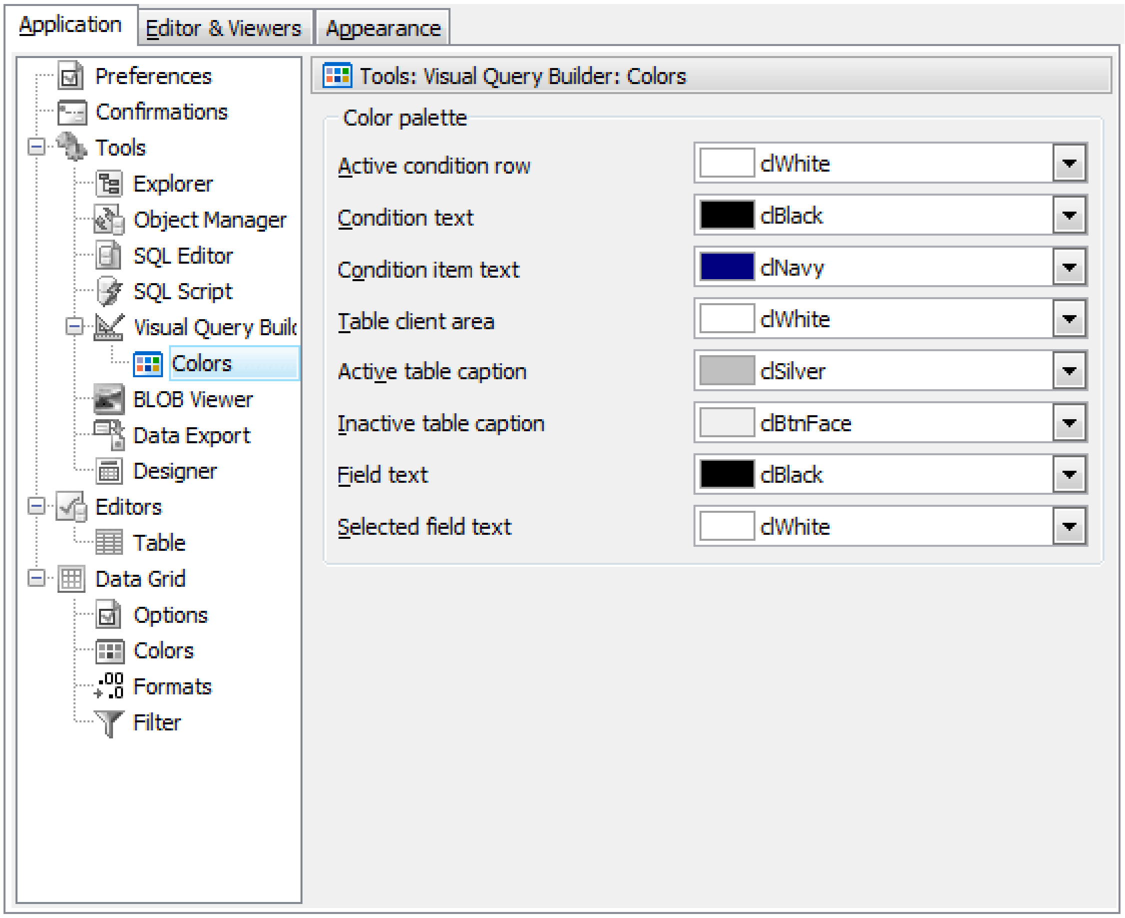
Script format
These options specify the case formatting of keywords and functions in query text on the Edit tab. As is saves the original case, Uppercase sets all the keywords/functions to upper case, Lowercase sets all the keywords/functions to lower case, and First upper sets the first letters of all keywords/functions to upper case.

Style

These options specify how different the Query Builder objects look like - 3D, flat, etc.

8.1.3.4.1 Colors

The tab is provided to editing of the Query Builder color schema. Customize colors for all editor element according to your preferences.



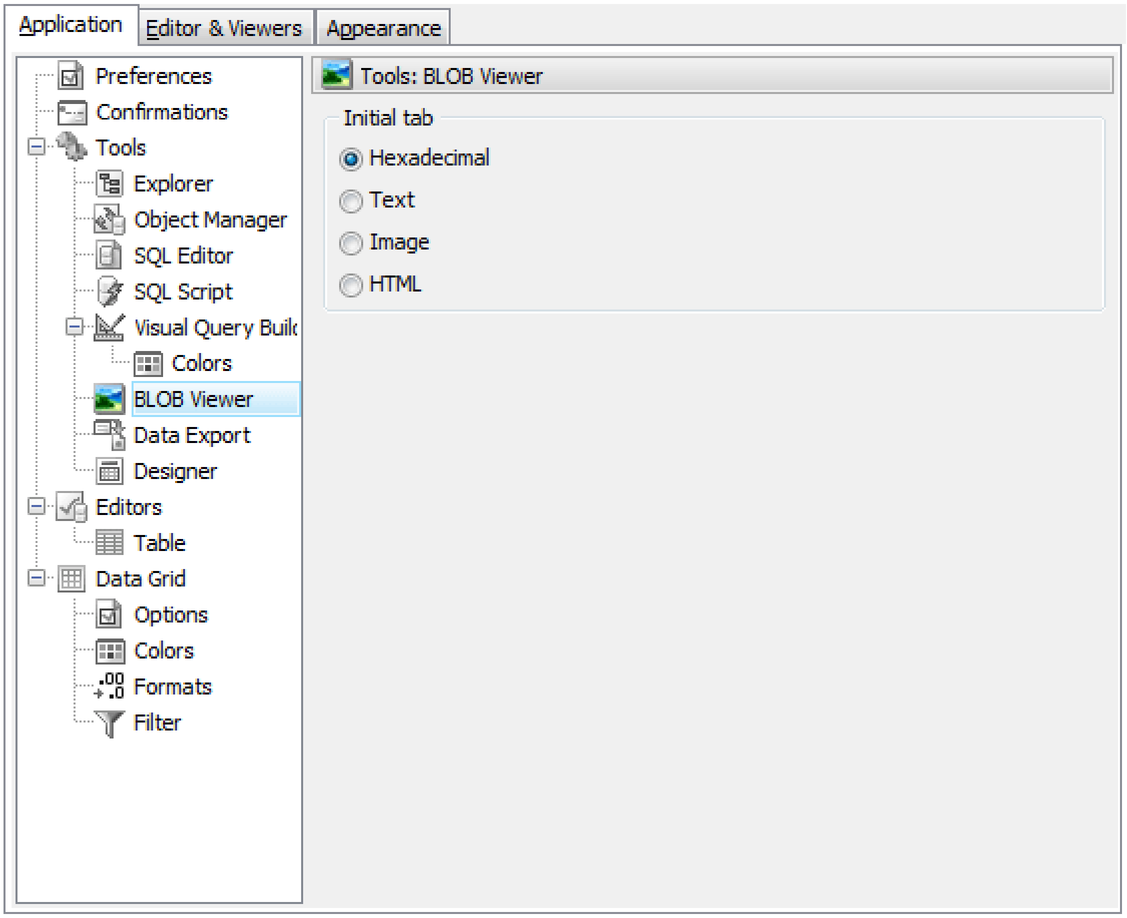
Tools: Visual Query Builder: Colors

Color palette

Active condition row	clWhite
Condition text	clBlack
Condition item text	clNavy
Table client area	clWhite
Active table caption	clSilver
Inactive table caption	clBtnFace
Field text	clBlack
Selected field text	clWhite

8.1.3.5 BLOB Viewer

Below you will find a detailed decryption of the following BLOB Viewer 94 options.



Initial tab

● Hexadecimal

○ Text

○ Image

○ HTML

Initial tab
Specifies which tab of BLOB Viewer should be active when it is initially opened.

8.1.3.6 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

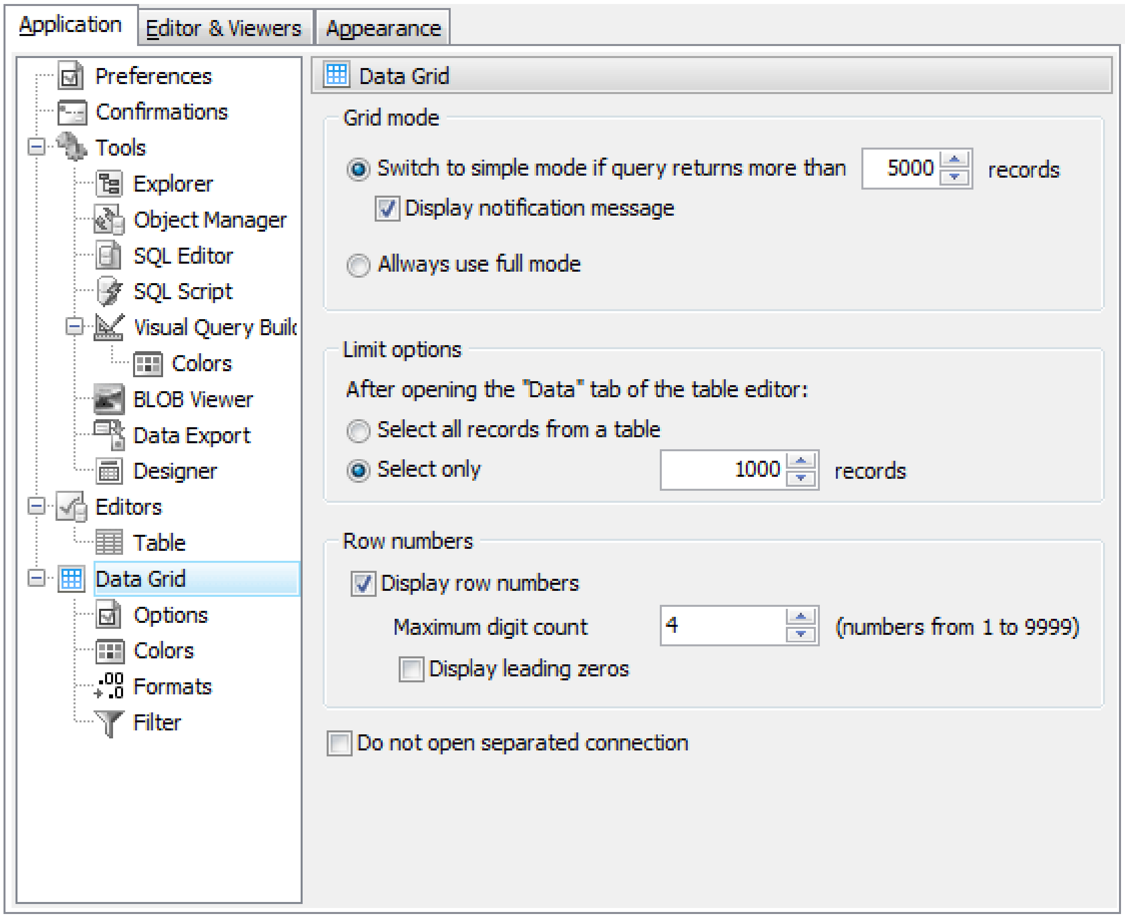
".", **","**, **":"** symbols can be placed as separators in format masks for *integer, float, date, time, date time, currency* types.

☑ Auto save format strings
Auto saves format strings when they are changed (e.g. at the Adjusting data formats step in Export Data Wizard 75).

Using Set defaults button, you can set default values of data formats.

8.1.4 Data Grid

Below you will find a detailed description of the following data grid options.



Grid mode
● Switch to simple mode if query returns more than 5000 records
☑ Display notification message
○ Allways use full mode

Limit options
After opening the "Data" tab of the table editor:
○ Select all records from a table
● Select only 1000 records

Row numbers
☑ Display row numbers
Maximum digit count 4 (numbers from 1 to 9999)
☐ Display leading zeros

☐ Do not open separated connection

SQLite Code Factory provides you with two grid modes 60 of viewing data:
 - Fool 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.

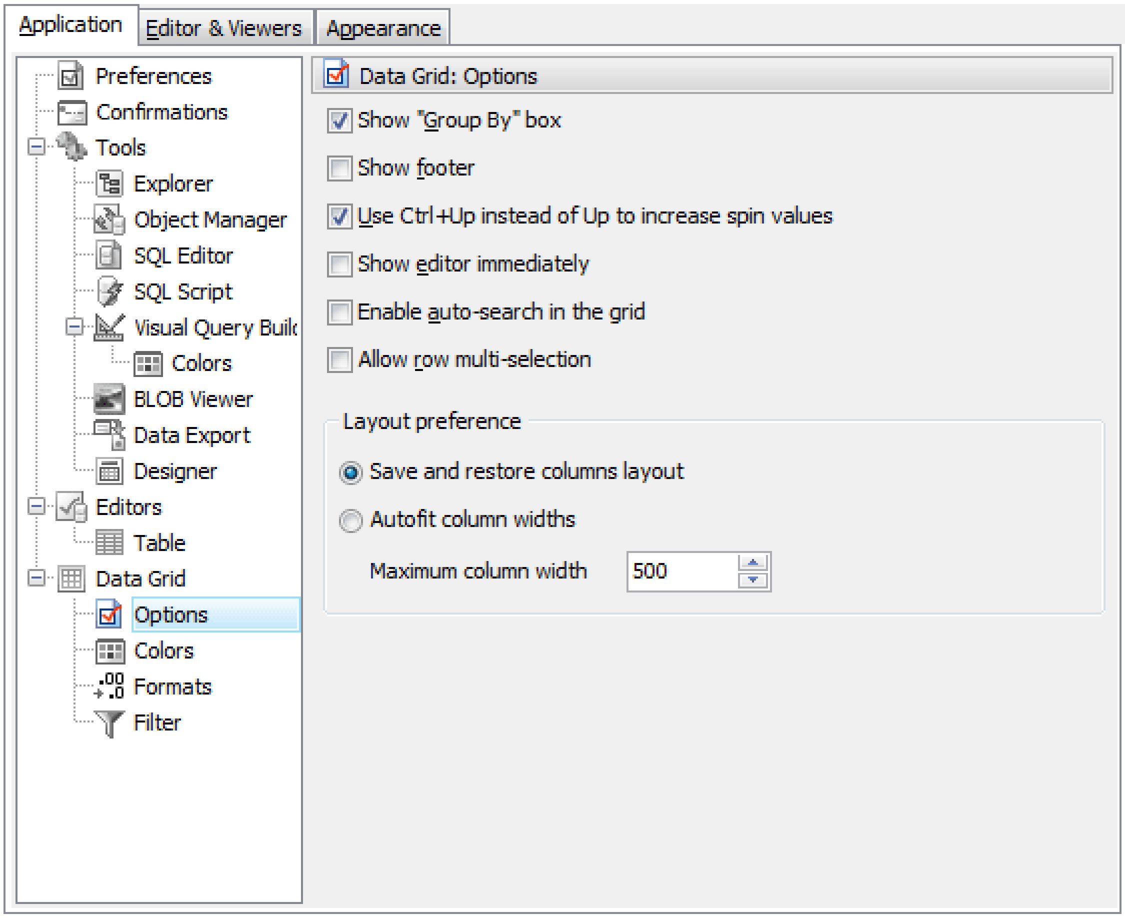
Row numbers
This options group allows you to manage grid rows numbering.
To enable/disable the numbering, use Display row numbers checkbox. You can set the number columns width with Maximum digit count. (I.e. for the value '3' the max column number will be 999).
For uniformity you can use the Display leading zeros option. With this option enabled and maximum digit count '3' you 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.

☑ Display TEXT fields as ordinary strings
Specify the option to view the TEXT fields as ordinary strings.

8.1.4.1 Options

Below you will find a detailed decryption of the data grid options.



Data Grid: Options

☑ Show "Group By" box
☐ Show footer
☑ Use Ctrl+Up instead of Up to increase spin values
☐ Show editor immediately
☐ Enable auto-search in the grid
☐ Allow row multi-selection

Layout preference

● Save and restore columns layout
○ Autofit column widths

Maximum column width 500

☑ **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.

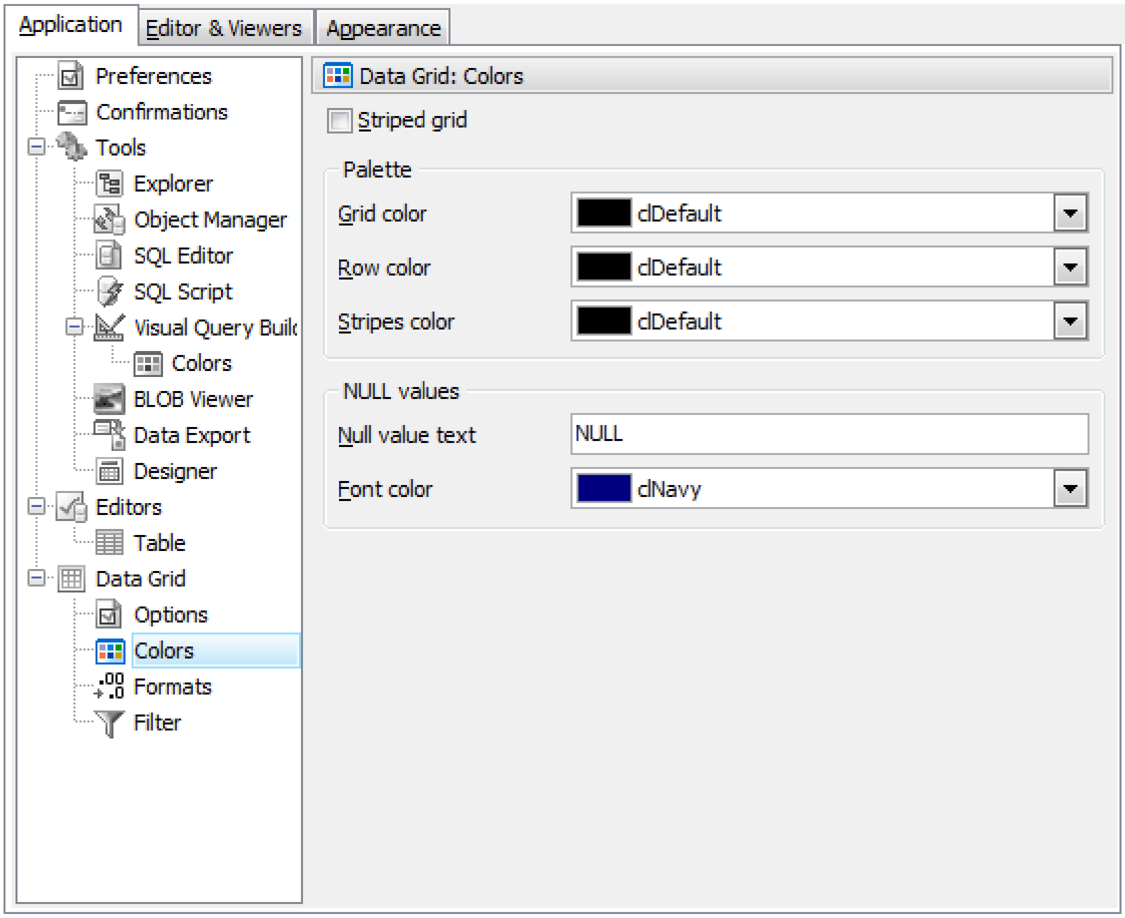
☑ **Enable auto-search in the grid**
Allows you to search records in the grid by the first letters.

☑ **Allow row multi-selection**
Allows you to select multiple records using the Ctrl and Shift keys.

Layout preference
Select whether SQLite Code Factory should remember the column positions for the grids or fit them automatically.

8.1.4.2 Colors

Below you will find a detailed decryption of the following colors options.



Data Grid: Colors

☐ Striped grid

Palette

Grid color: clDefault

Row color: clDefault

Stripes color: clDefault

NULL values

Null value text: NULL

Font color: clNavy

☑ 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.

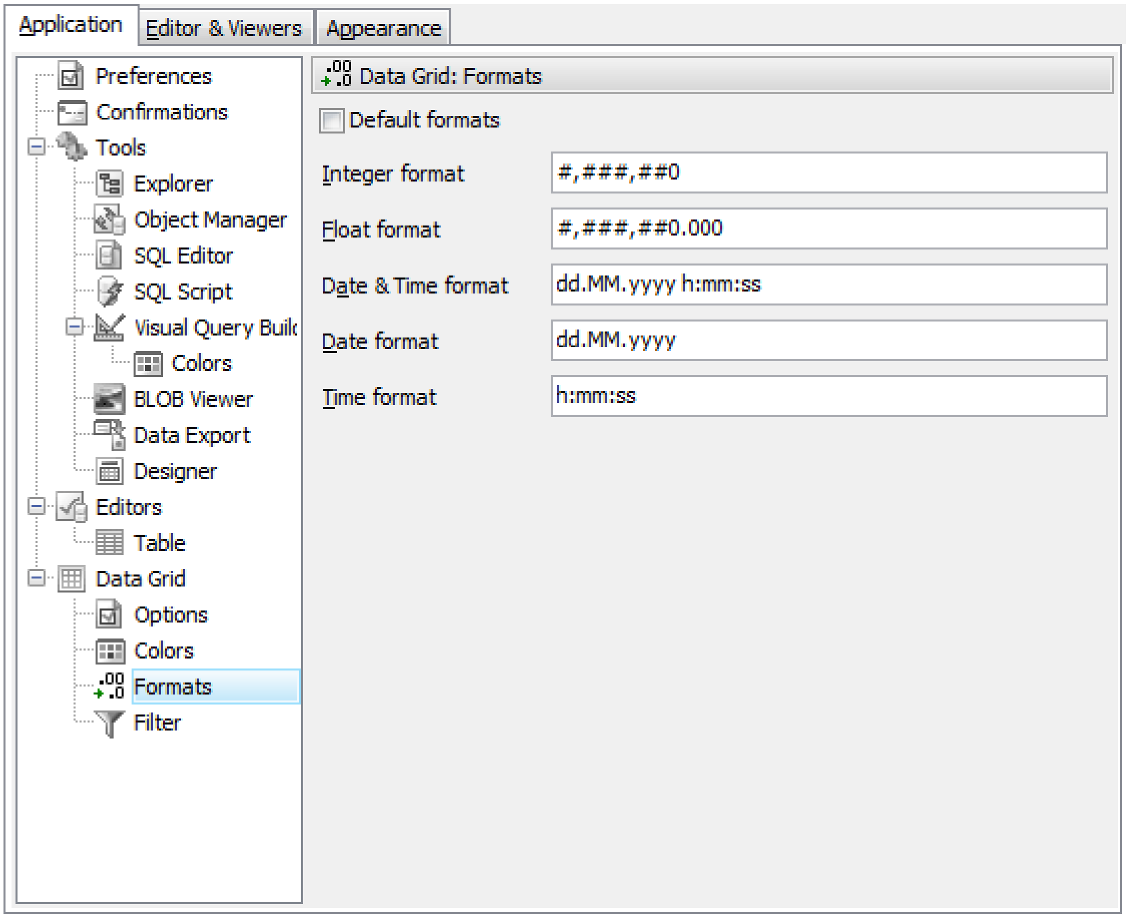
Row color
Defines the color of the selected row in the data grid.

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.

8.1.4.3 Formats

Below you will find a detailed decryption of the following formats options.



☐ Default formats

Integer format: #,###,##0

Float format: #,###,##0.000

Date & Time format: dd.MM.yyyy h:mm:ss

Date format: dd.MM.yyyy

Time format: h:mm:ss

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:

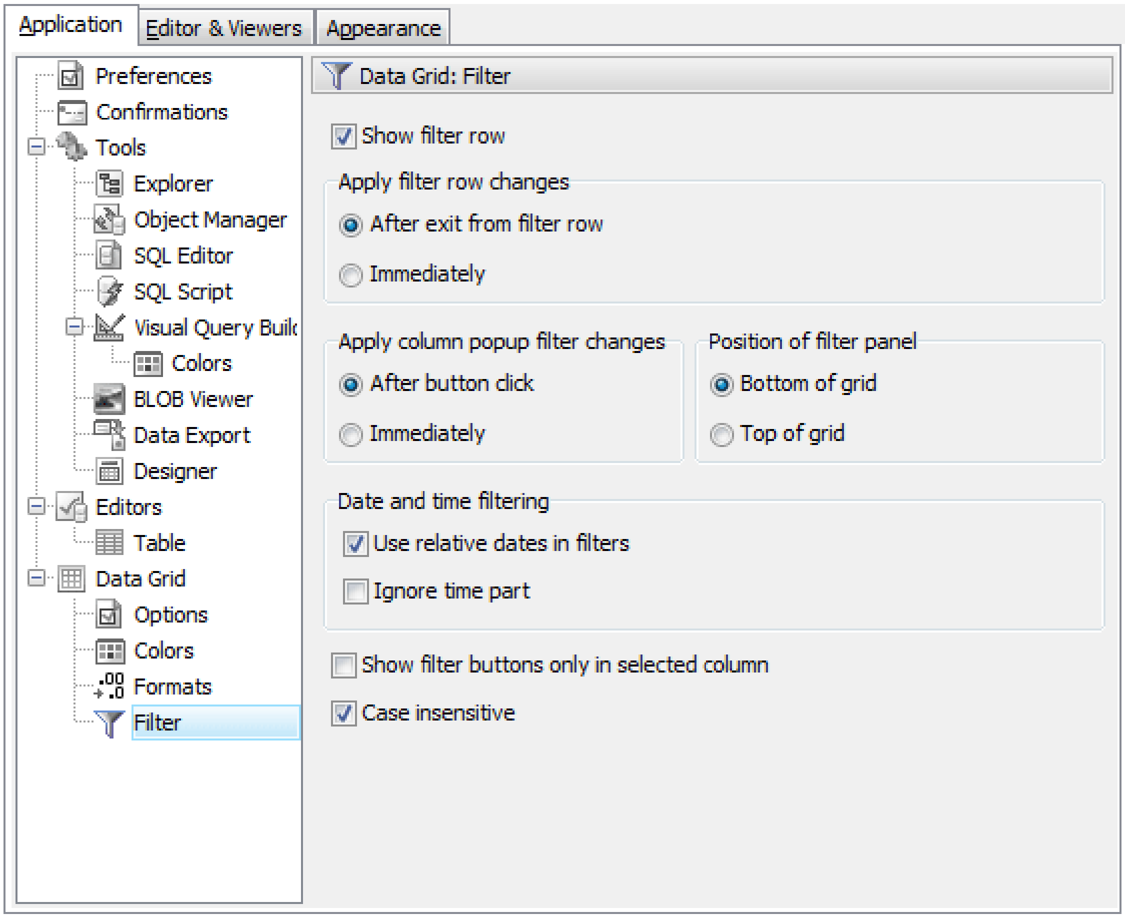
dd	day of the month, represented by 1 or 2 symbols. For example, the first day of month is 1
DD	day of the month, represented only by 2 symbols. For example, the first day of month is 01
mm	minutes
MM	month
yy	year, represented by 2 symbols. For example, 2006 year will be 06

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.

8.1.4.4 Filter

These options allows you to customize data filtering in grids.



Data Grid: Filter

☑ Show filter row

Apply filter row changes
● After exit from filter row
○ Immediately

Apply column popup filter changes
● After button click
○ Immediately

Position of filter panel
● Bottom of grid
○ Top of grid

Date and time filtering
☑ Use relative dates in filters
☐ Ignore time part

☐ Show filter buttons only in selected column
☑ Case insensitive

☑ Show filter row
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).

8.2 Editors & Viewers

The Editors & Viewers section allows you to set the parameters of viewing and editing the SQL statements within SQLite Code Factory.

- General 127
- Display 128
- SQL highlight 129
- PHP highlight 131
- XML highlight 130
- Code Insight 132
- Code Folding 133

See also: SQL Editor 41, SQL Script Editor 56, Visual Query Builder 46.

8.2.1 General

If the Auto indent option is checked, each new indention 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.

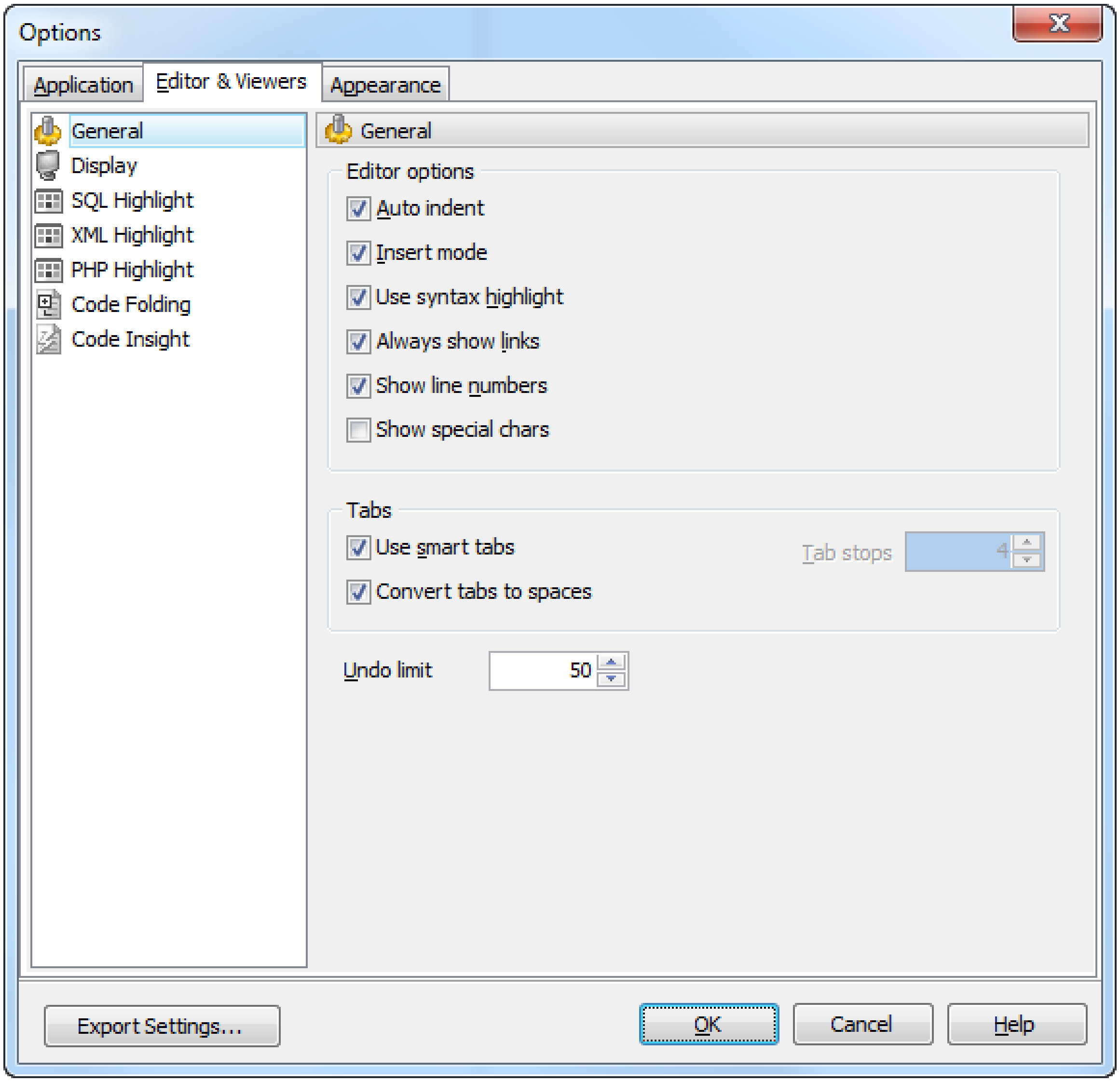
☑ Show special chars
If this option is checked, special chars (like line breaks) are displayed in the editor window.

☑ 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.



Editor options
☑ Auto indent
☑ Insert mode
☑ Use syntax highlight
☑ Always show links
☑ Show line numbers
☐ Show special chars

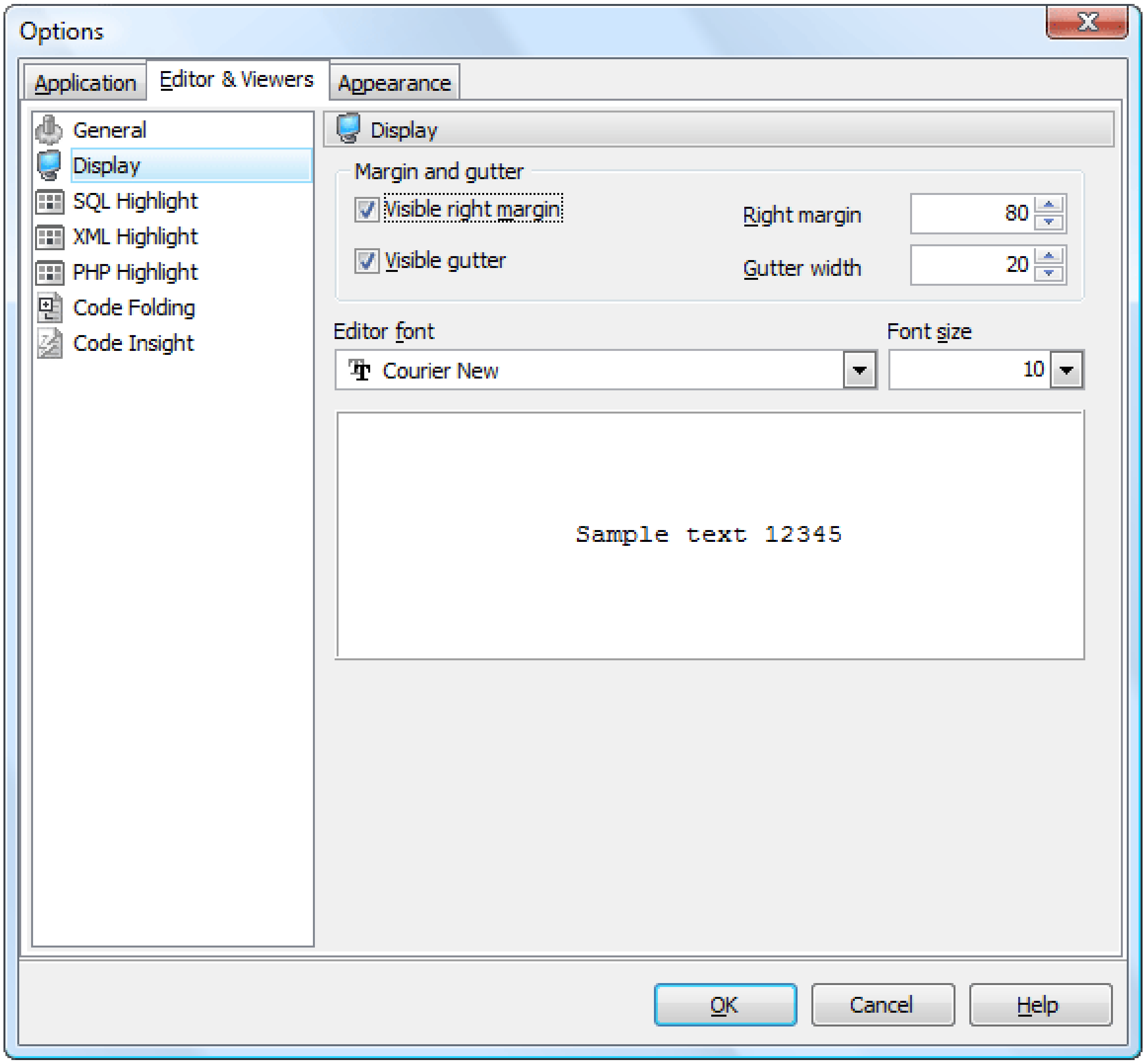
Tabs
☑ Use smart tabs
☑ Convert tabs to spaces
Tab stops 4

Undo limit 50

8.2.2 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.



☑ Visible right margin

Right margin 80

☑ Visible gutter

Gutter width 20

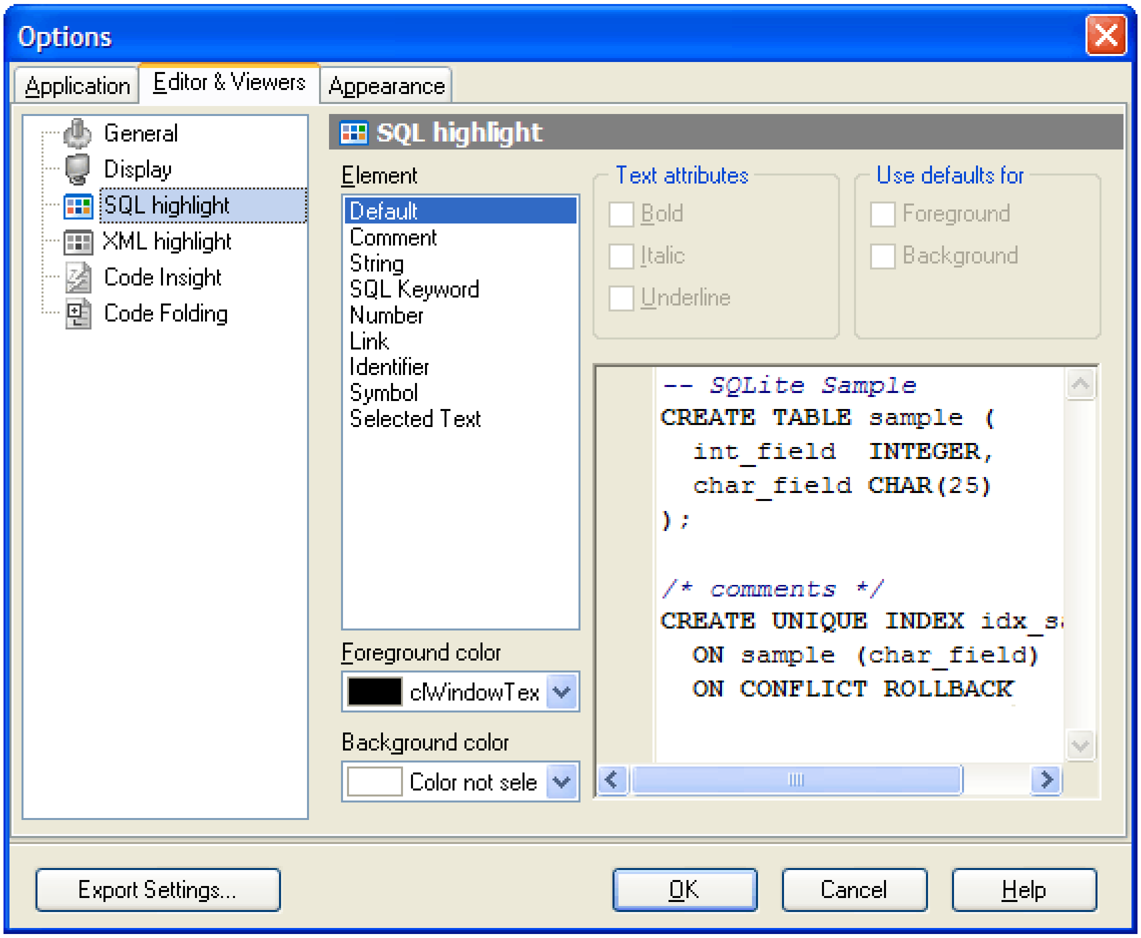
Editor font: Courier New

Font size: 10

```
Sample text 12345
```

8.2.3 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.



Element: Default

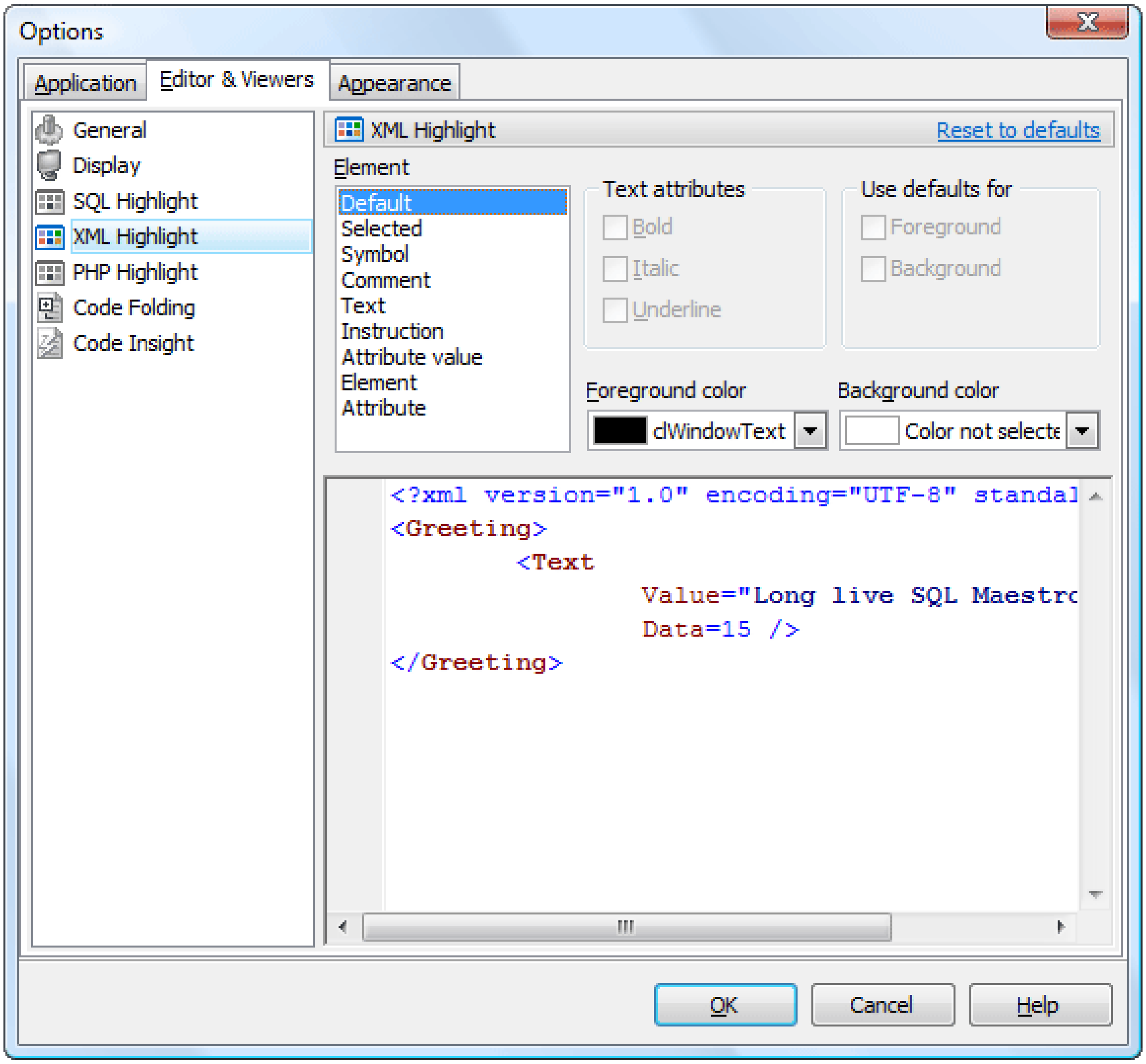
Foreground color: clWindowTex

Background color: Color not sele

```
-- SQLite Sample
CREATE TABLE sample (
  int_field  INTEGER,
  char_field CHAR(25)
);

/* comments */
CREATE UNIQUE INDEX idx_s
  ON sample (char_field)
  ON CONFLICT ROLLBACK
```

8.2.4 XML highlight

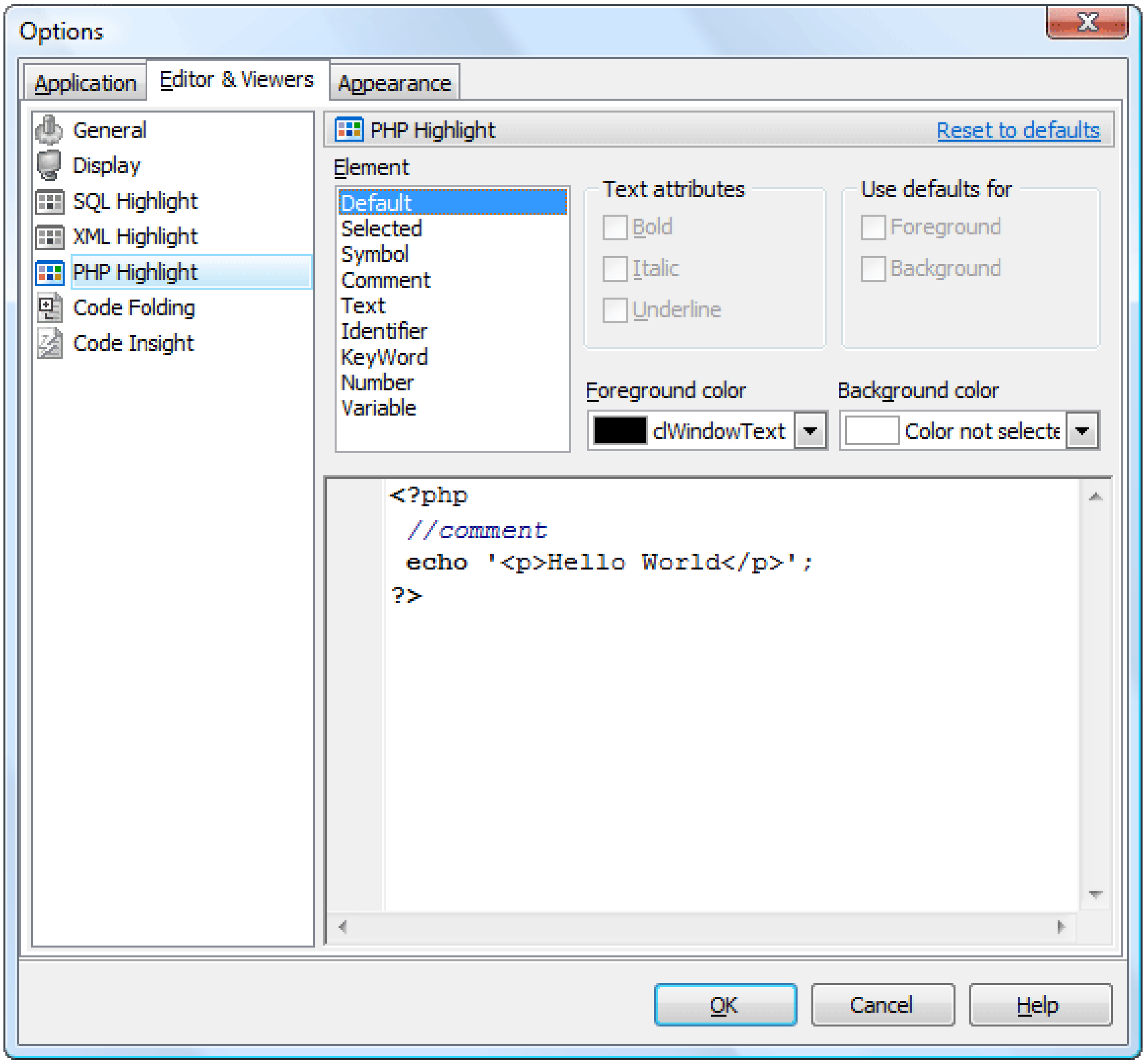
Use the XML highlight item to customize XML syntax highlight for the text representation of BLOBs in BLOB Viewer/Editor. Select the text element from the list, e.g. attribute or attribute value and adjust its foreground color, background color and text attributes according to your wishes.



```
<?xml version="1.0" encoding="UTF-8" standal
<Greeting>
        <Text
                Value="Long live SQL Maestrc
                Data=15 />
</Greeting>
```

8.2.5 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.



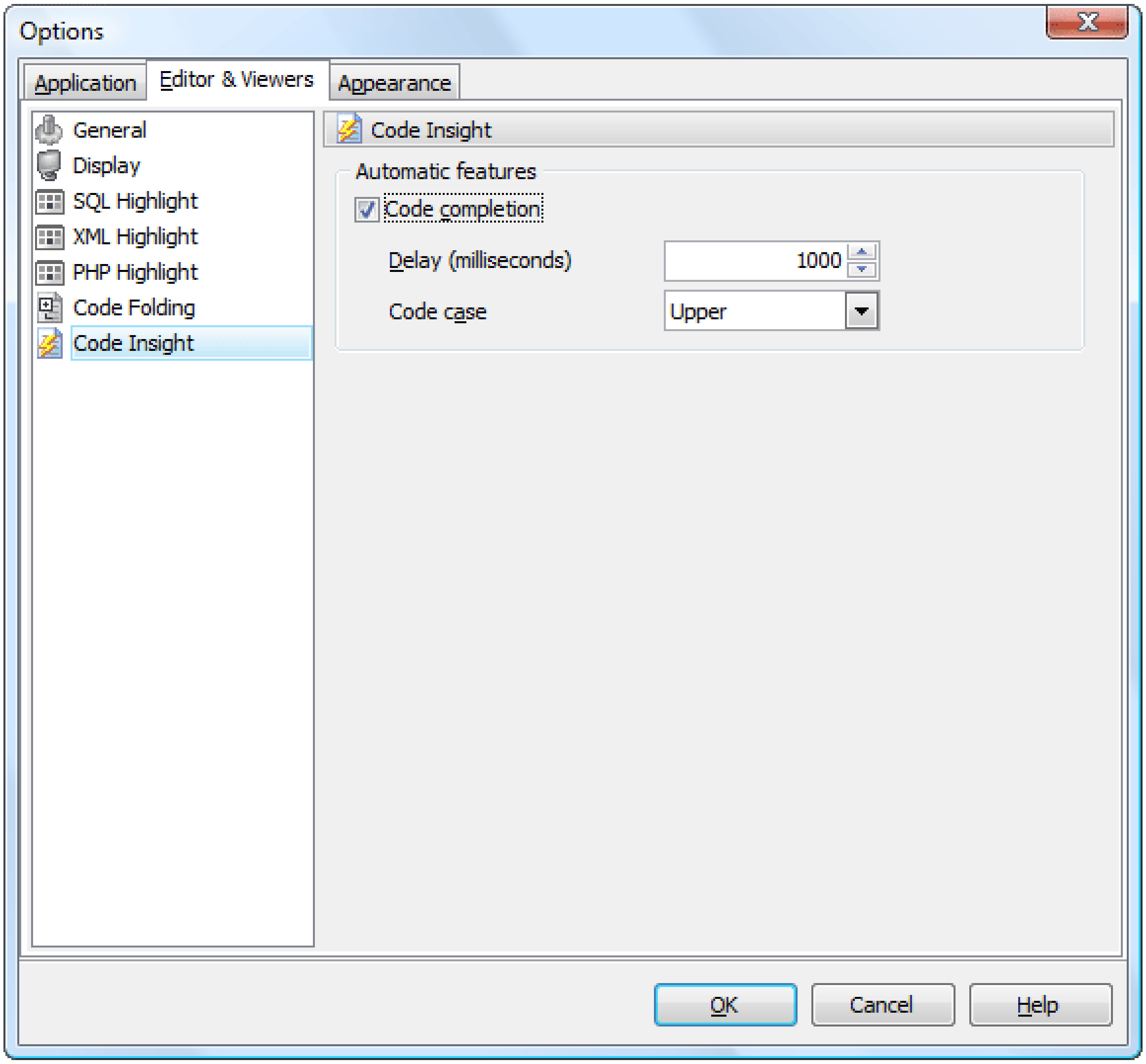
Foreground color: clWindowText

Background color: Color not selecte

```
<?php
 //comment
 echo '<p>Hello World</p>';
?>
```

8.2.6 Code Insight

You can disable/enable the code completion with the corresponding option and also set the time is appears as Delay, and case of words inserted automatically.



Code Insight

Automatic features

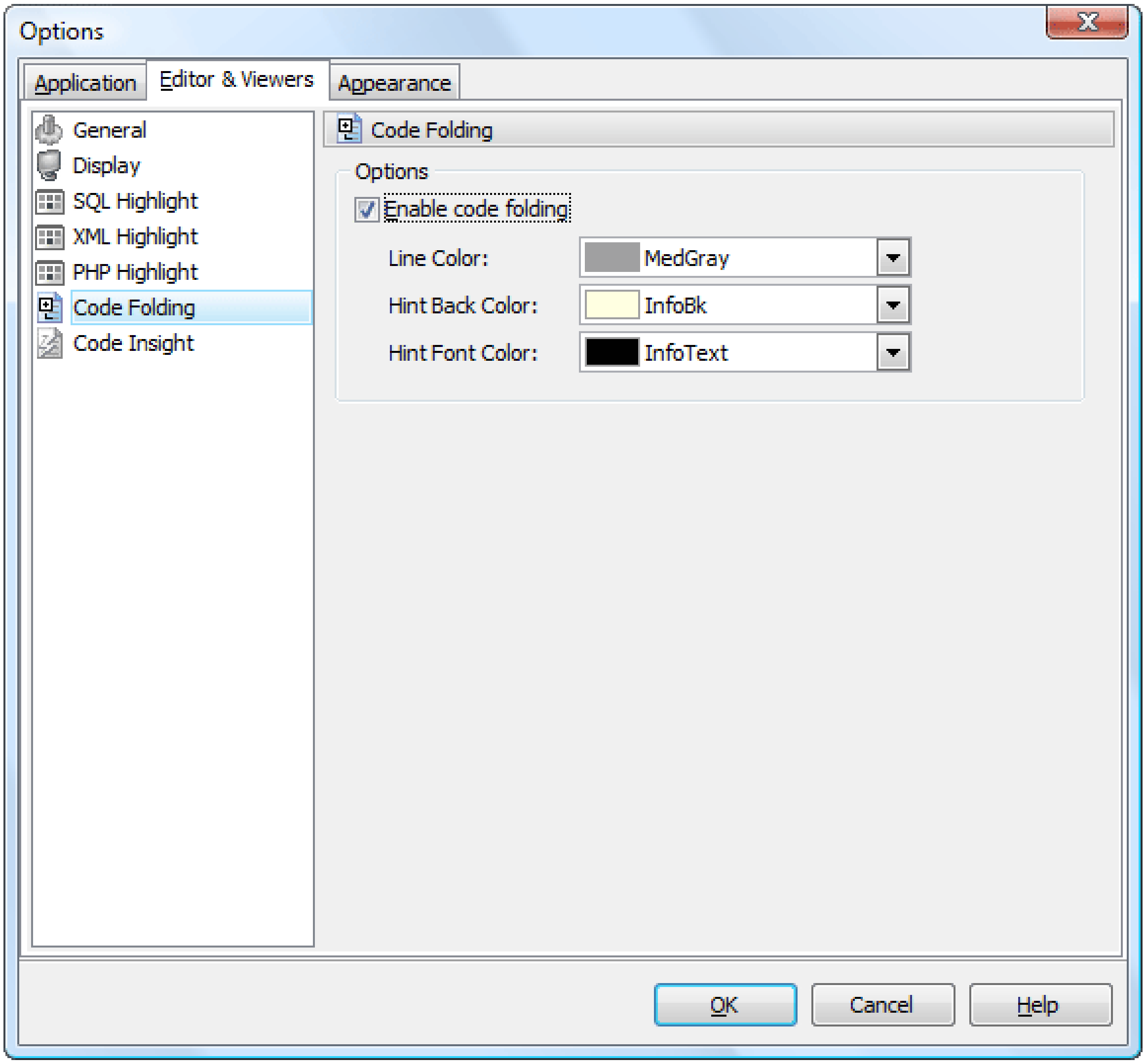
☑ Code completion

Delay (milliseconds) 1000

Code case Upper

8.2.7 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.



☑ Enable code folding

Line Color: MedGray

Hint Back Color: InfoBk

Hint Font Color: InfoText

You can enable/disable code folding in SQL editors and viewers and customize the colors of its items.

8.3 Appearance

The Appearance section allows you to customize the application interface style to your preferences.

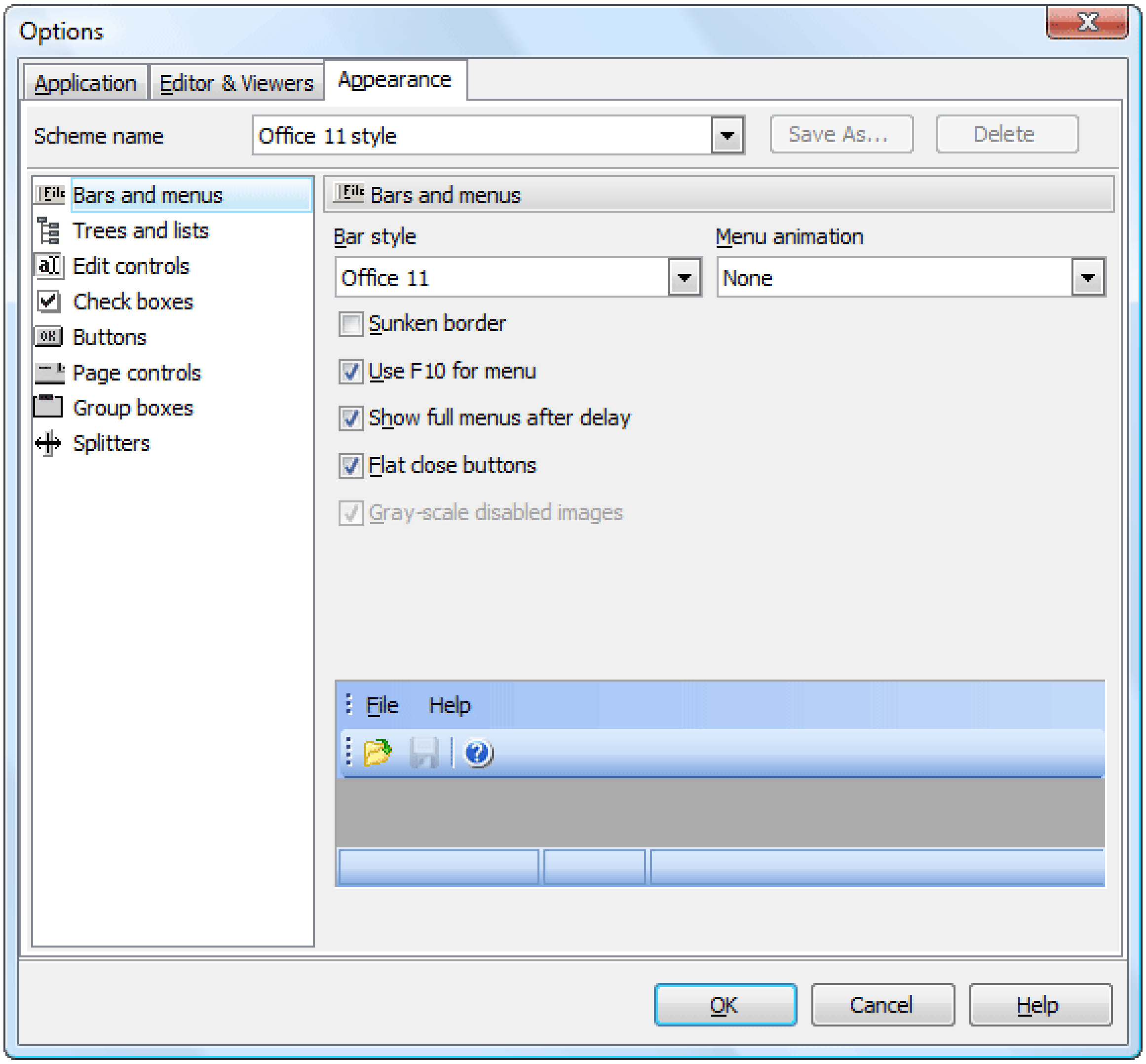
Use the Scheme name box to select the interface scheme you prefer: *Office XP style*, *Windows XP native style*, etc. You can create your own interface schemes by customizing any visual options (Bars and menus, Trees and lists, Edit controls, Check boxes, Buttons, etc.) and clicking the Save As button. All the customized options are displayed on the sample panel.

- Bars and menus 135
- Trees and lists 136
- Edit controls 137
- Check boxes 138
- Buttons 139
- Page controls 140
- Group boxes 141
- Splitters 142

8.3.1 Bars and menus

Use the Bars and menus item to customize SQLite Code Factory 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.



Scheme name: Office 11 style

Bar style: Office 11

Menu animation: None

☐ Sunken border

☑ Use F10 for menu

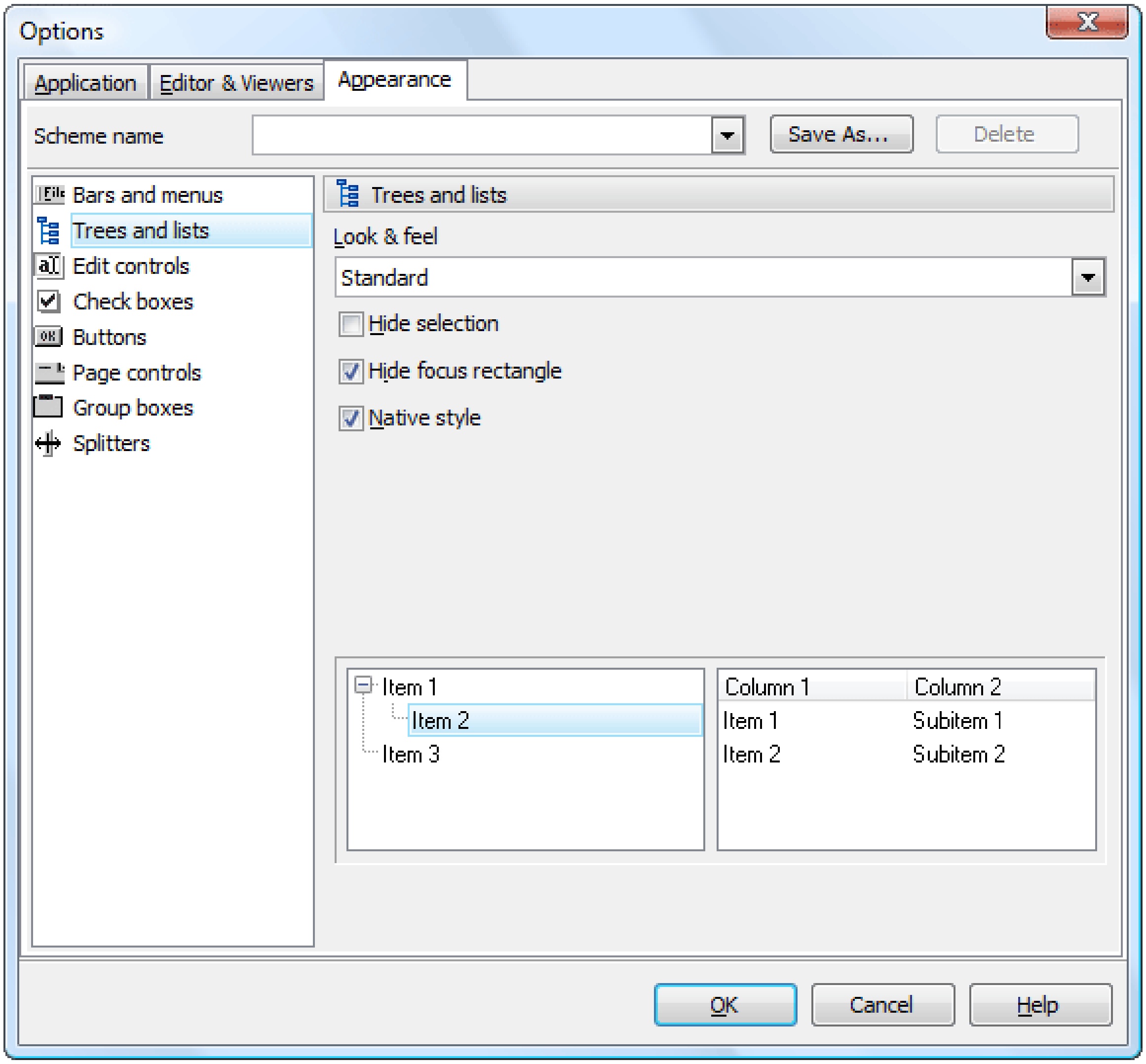
☑ Show full menus after delay

☑ Flat close buttons

☑ Gray-scale disabled images

8.3.2 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.



Look & feel: Standard

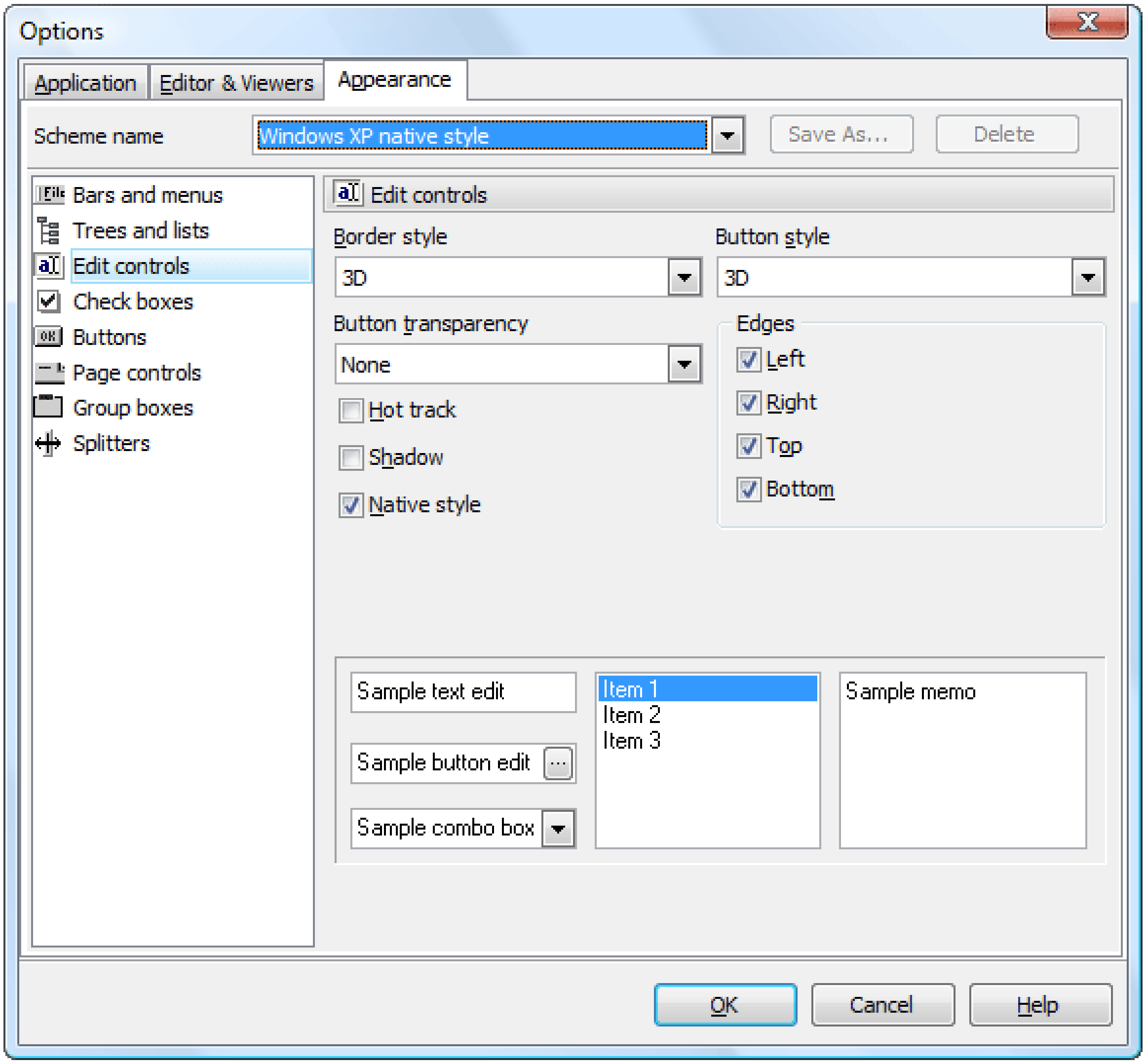
☐ Hide selection

☑ Hide focus rectangle

☑ Native style

8.3.3 Edit controls

Use the Edit controls item to customize the appearance of different SQLite Code Factory 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.



Scheme name: Windows XP native style

Border style: 3D

Button style: 3D

Button transparency: None

☐ Hot track

☐ Shadow

☑ Native style

Edges

☑ Left

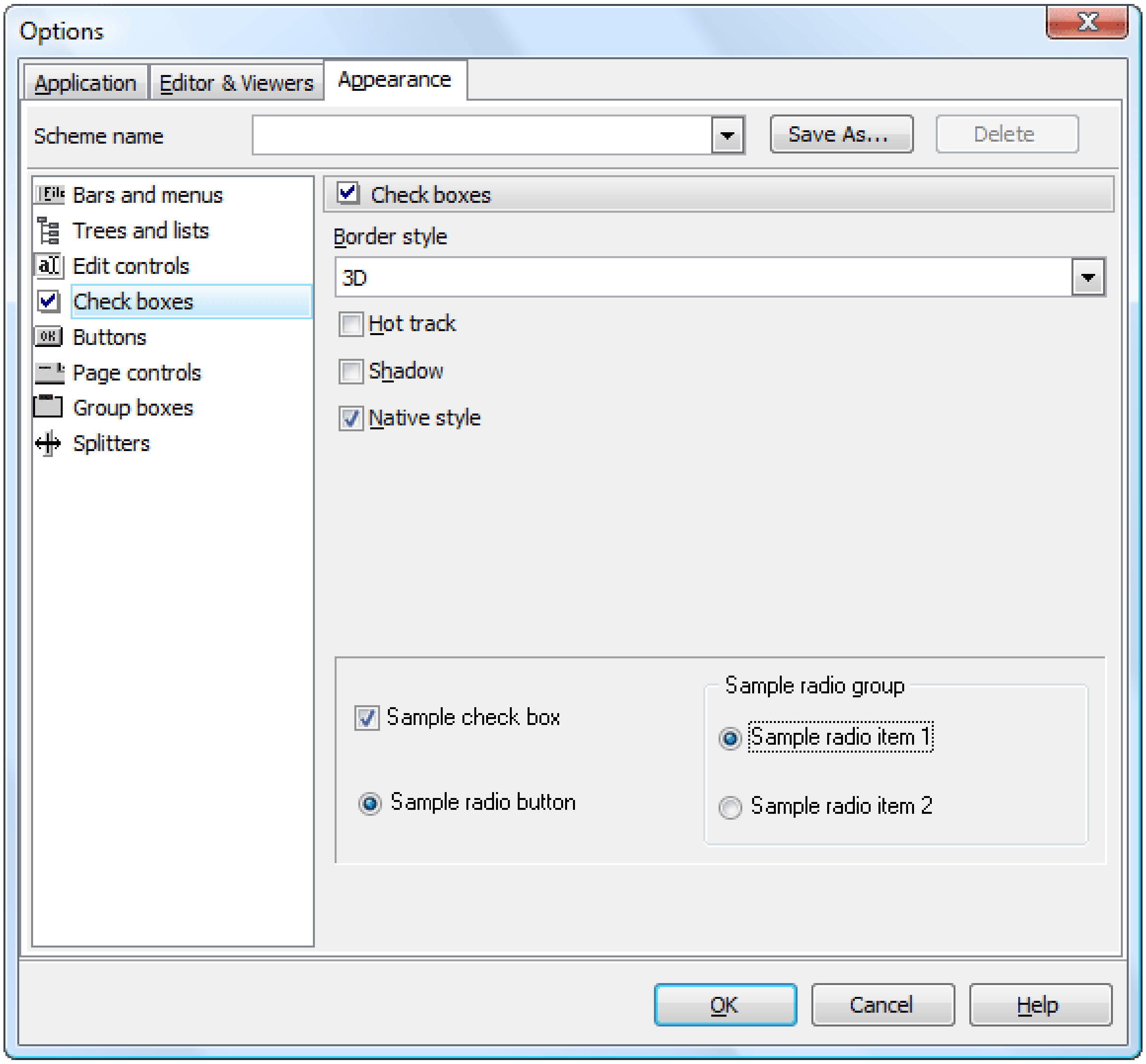
☑ Right

☑ Top

☑ Bottom

8.3.4 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.



☑ Check boxes

Border style: 3D

☐ Hot track

☐ Shadow

☑ Native style

☑ Sample check box

● Sample radio button

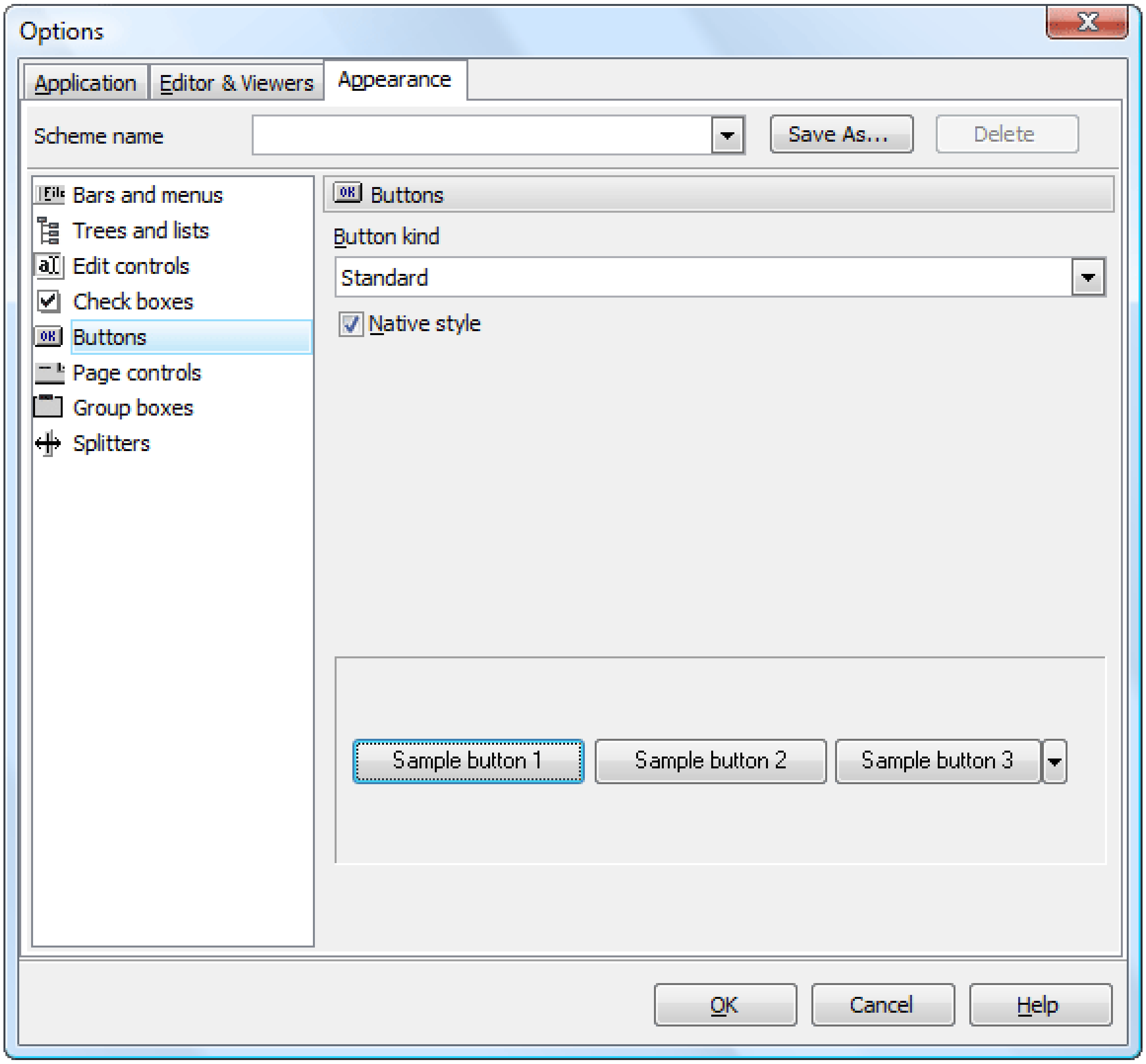
Sample radio group

● Sample radio item 1

○ Sample radio item 2

8.3.5 Buttons

Use the Buttons item to customize SQLite Code Factory buttons. The tab allows you to adjust the appearance of buttons and define sample buttons as well.



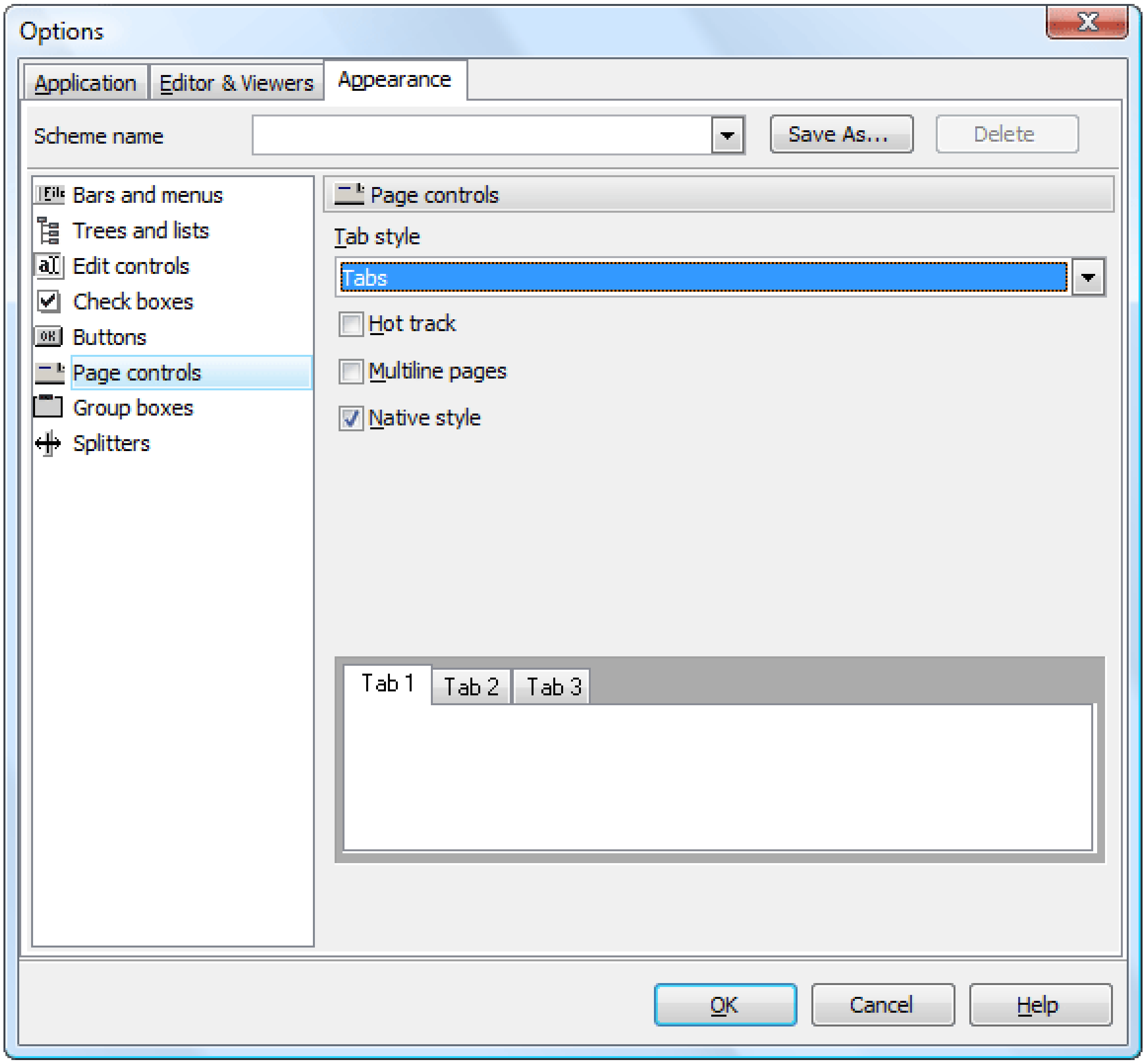
Scheme name

Button kind: Standard

☑ Native style

8.3.6 Page controls

The Page controls item allows you to customize the style of all SQLite Code Factory page controls. The tab allows you to select tab styles, enable/disable hot track, multi-line pages and native style.



Scheme name

Tab style: Tabs

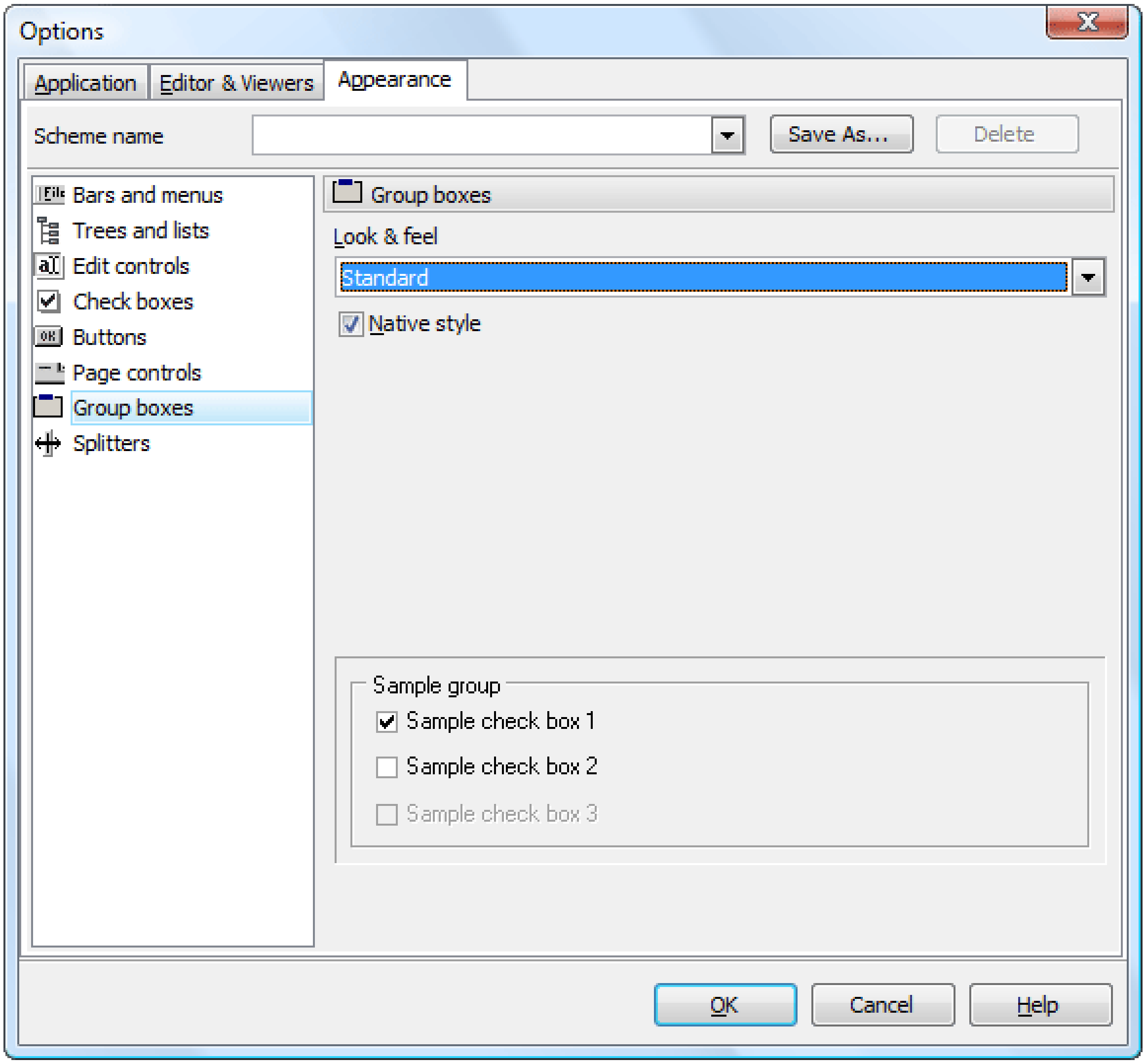
☐ Hot track

☐ Multiline pages

☑ Native style

8.3.7 Group boxes

Use the Group boxes item to customize all SQLite Code Factory group boxes according to your preferences. Use tab to apply styles for group boxes, enable/disable native style and define samples.



Look & feel: Standard

☑ Native style

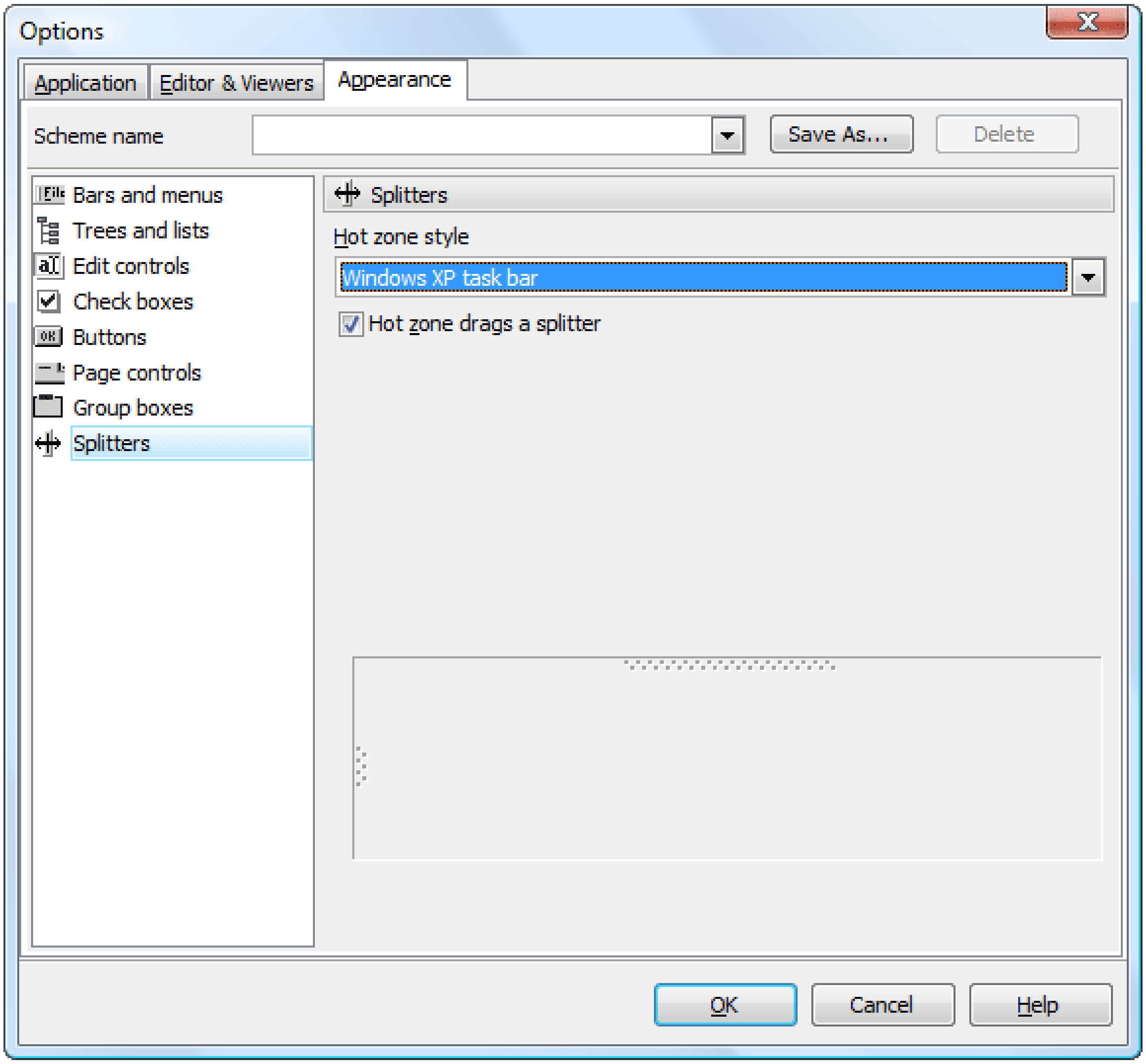
☑ Sample check box 1

☐ Sample check box 2

☐ Sample check box 3

8.3.8 Splitters

Use the Splitters item to customize all SQLite Code Factory 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.



Scheme name

Splitters

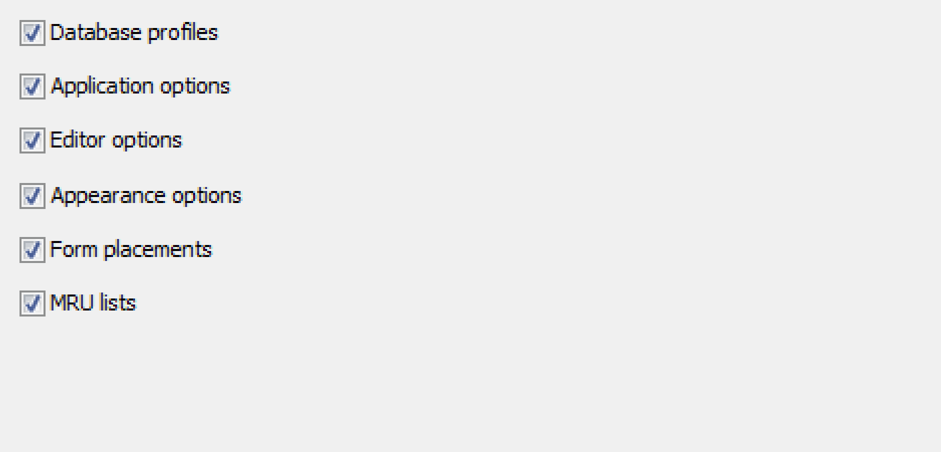
Hot zone style: Windows XP task bar

☑ Hot zone drags a splitter

8.4 Export Settings

Export Settings Wizard allows you to export all or partial SQLite Code Factory settings to single **.reg* file which can be applied to the application of SQLite Code Factory 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 109 dialog.

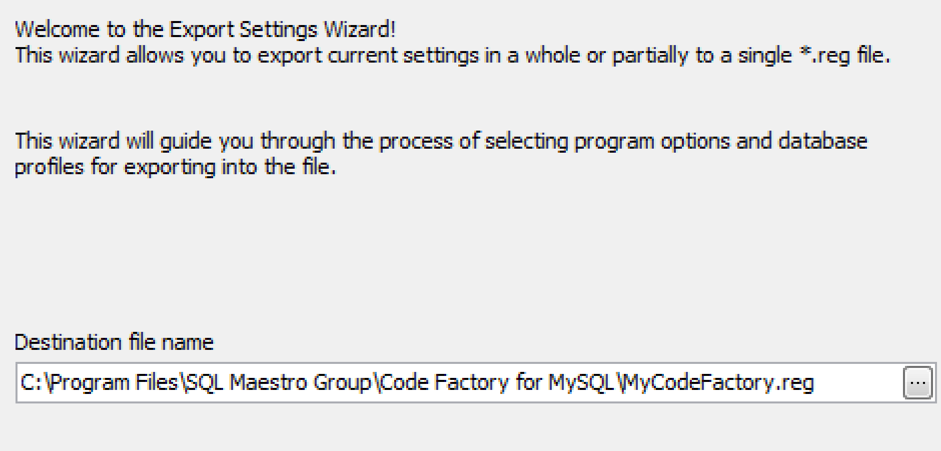
- Specifying destination file to save settings to 144
- Specifying settings categories to save 144
- Select database profiles to save 145
- Saving settings 145



☑ Database profiles
☑ Application options
☑ Editor options
☑ Appearance options
☑ Form placements
☑ MRU lists

8.4.1 Specifying destination file

Specify a **.reg* file to extract SQLite Code Factory setting to.



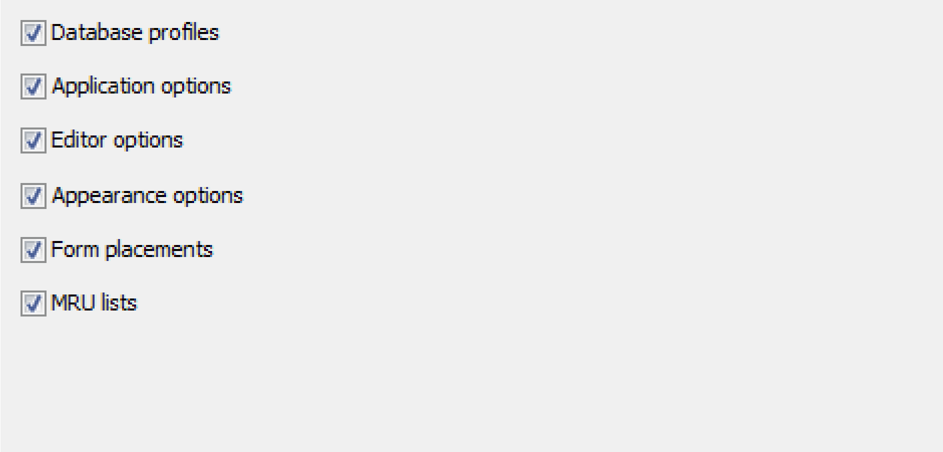
Welcome to the Export Settings Wizard!
This wizard allows you to export current settings in a whole or partially to a single *.reg file.

This wizard will guide you through the process of selecting program options and database profiles for exporting into the file.

Destination file name
C:\Program Files\SQL Maestro Group\Code Factory for MySQL\MyCodeFactory.reg

8.4.2 Selecting setting categories

The options of this step specify the information saved to the result file, e.g. Database profiles, Application options 110, etc.



☑ Database profiles

☑ Application options

☑ Editor options

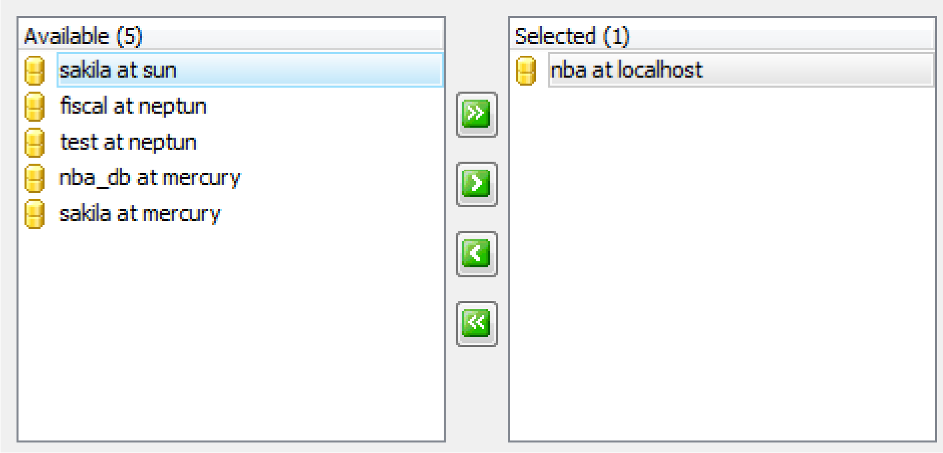
☑ Appearance options

☑ Form placements

☑ MRU lists

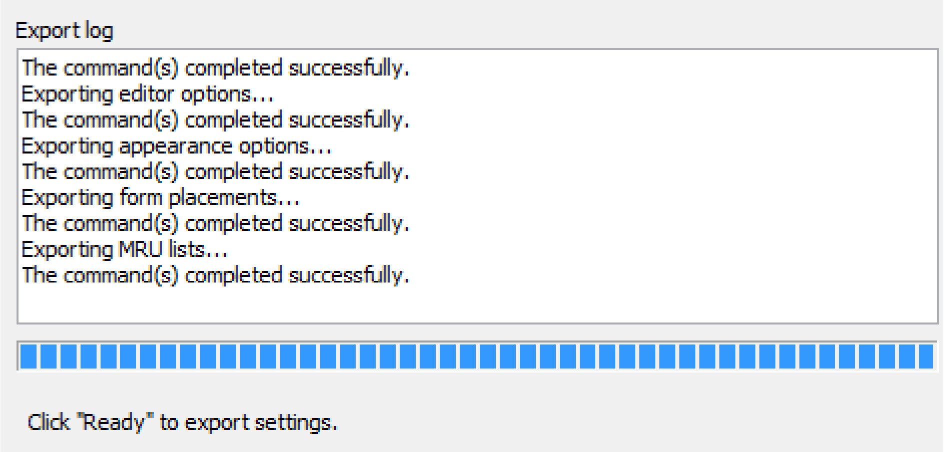
8.4.3 Selecting database profiles

Select database profiles to save their settings by moving them from the Available Databases list to the Selected Databases one.



8.4.4 Saving settings

Click the Ready button to start the extracting. The process log is displayed in the Export log box.



```
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.

9 SQLite references

The SQLite library understands most of the standard SQL language. But it does omit some features while at the same time adding a few features of its own. This document attempts to describe precisely what parts of the SQL language SQLite does and does not support.

In all of the syntax diagrams that follow, literal text is shown in bold blue. Non-terminal symbols are shown in italic red. Operators that are part of the syntactic markup itself are shown in black roman.

This document is just an overview of the SQL syntax implemented by SQLite. Many low-level productions are omitted. For detailed information on the language that SQLite understands, refer to the source code.

SQLite implements the follow syntax:

- BEGIN TRANSACTION 148
- COPY 149
- CREATE INDEX 150
- CREATE TABLE 151
- CREATE TRIGGER 153
- CREATE VIEW 156
- DELETE 157
- DROP INDEX 158
- DROP TABLE 159
- DROP TRIGGER 160
- DROP VIEW 161
- EXPLAIN 162
- expression 163
- INSERT 167
- ON CONFLICT clause 168
- PRAGMA 170
- REPLACE 173
- SELECT 174
- UPDATE 176
- VACUUM 177

Details on the implementation of each command are provided in the sequel.

9.1 BEGIN TRANSACTION

sql-statement ::= **BEGIN** [**TRANSACTION** [*name*]] [**ON CONFLICT** *conflict-algorithm*]

sql-statement ::= **END** [**TRANSACTION** [*name*]]

sql-statement ::= **COMMIT** [**TRANSACTION** [*name*]]

sql-statement ::= **ROLLBACK** [**TRANSACTION** [*name*]]

Beginning in version 2.0, SQLite supports transactions with rollback and atomic commit. No changes can be made to the database except within a transaction. Any command that changes the database (basically, any SQL command other than SELECT) will automatically starts a transaction if one is not already in effect. Automatically started transactions are committed at the conclusion of the command.

Transactions can be started manually using the BEGIN command. Such transactions usually persist until the next COMMIT or ROLLBACK command. But a transaction will also ROLLBACK if the database is closed or if an error occurs and the ROLLBACK conflict resolution algorithm is specified. See the documentation on the ON CONFLICT 168 clause for additional information about the ROLLBACK conflict resolution algorithm.
The optional ON CONFLICT clause at the end of a BEGIN statement can be used to changed the default conflict resolution algorithm. The normal default is ABORT. If an alternative is specified by the ON CONFLICT clause of a BEGIN, then that alternative is used as the default for all commands within the transaction. The default algorithm is overridden by ON CONFLICT clauses on individual constraints within the CREATE TABLE or CREATE INDEX statements and by the OR clauses on COPY, INSERT, and UPDATE commands.

9.2 COPY

sql-statement ::= **COPY** [**OR** *conflict-algorithm*] *table-name* **FROM** *filename*
[**USING DELIMITERS** *delim*]

The COPY command is an extension used to load large amounts of data into a table. It is modeled after a similar command found in PostgreSQL. In fact, the SQLite COPY command is specifically designed to be able to read the output of the PostgreSQL dump utility pg_dump so that data can be easily transferred from PostgreSQL into SQLite.
The table-name is the name of an existing table which is to be filled with data. The filename is a string or identifier that names a file from which data will be read. The filename can be the STDIN to read data from standard input.
Each line of the input file is converted into a single record in the table. Columns are separated by tabs. If a tab occurs as data within a column, then that tab is preceded by a backslash "\" character. A backslash in the data appears as two backslashes in a row. The optional USING DELIMITERS clause can specify a delimiter other than tab.
If a column consists of the character "\N", that column is filled with the value NULL.

The optional conflict-clause allows the specification of an alternative constraint conflict resolution algorithm to use for this one command. See the section titled ON CONFLICT 168 for additional information.
When the input data source is STDIN, the input can be terminated by a line that contains only a backslash and a dot: "\.".

9.3 CREATE INDEX

sql-statement ::= **CREATE** [**UNIQUE**] **INDEX** *index-name*
ON *table-name* **(** *column-name* [**,** *column-name*]* **)**
[**ON CONFLICT** *conflict-algorithm*]

column-name ::= *name* [**ASC** | **DESC**]

The CREATE INDEX command consists of the keywords "CREATE INDEX" followed by the name of the new index, the keyword "ON", the name of a previously created table that is to be indexed, and a parenthesized list of names of columns in the table that are used for the index key. Each column name can be followed by one of the "ASC" or "DESC" keywords to indicate sort order, but the sort order is ignored in the current implementation.
There are no arbitrary limits on the number of indices that can be attached to a single table, nor on the number of columns in an index.
If the UNIQUE keyword appears between CREATE and INDEX then duplicate index entries are not allowed. Any attempt to insert a duplicate entry will result in an error.

The optional conflict-clause allows the specification of an alternative default constraint conflict resolution algorithm for this index. This only makes sense if the UNIQUE keyword is used since otherwise there are not constraints on the index. The default algorithm is ABORT. If a COPY, INSERT, or UPDATE statement specifies a particular conflict resolution algorithm, that algorithm is used in place of the default algorithm specified here. See the section titled ON CONFLICT 168 for additional information.
The exact text of each CREATE INDEX statement is stored in the sqlite_master or sqlite_temp_master table, depending on whether the table being indexed is temporary. Every time the database is opened, all CREATE INDEX statements are read from the sqlite_master table and used to regenerate SQLite's internal representation of the index layout.

9.4 CREATE TABLE

sql-command ::= **CREATE** [**TEMP** | **TEMPORARY**] **TABLE** _table-name_ **(**
    column-def [**,** _column-def_]*
    [**,** _constraint_]*
)

sql-command ::= **CREATE** [**TEMP** | **TEMPORARY**] **TABLE** _table-name_ **AS** _select-statement_

column-def ::= _name_ [_type_] [_column-constraint_]*

type ::= _typename_ |
typename **(** _number_ **)** |
typename **(** _number_ **,** _number_ **)**

column-constraint ::= **NOT NULL** [_conflict-clause_] |
PRIMARY KEY [_sort-order_] [_conflict-clause_] |
UNIQUE [_conflict-clause_] |
CHECK (_expr_ **)** [_conflict-clause_] |
DEFAULT _value_

constraint ::= **PRIMARY KEY (** _name_ [**,** _name_]* **)** [_conflict-clause_]|
UNIQUE (_name_ [**,** _name_]* **)** [_conflict-clause_] |
CHECK (_expr_ **)** [_conflict-clause_]

conflict-clause ::= **ON CONFLICT** _conflict-algorithm_

A CREATE TABLE statement is basically the keywords "CREATE TABLE" followed by the name of a new table and a parenthesized list of column definitions and constraints. The table name can be either an identifier or a string. Tables names that begin with "sqlite_" are reserved for use by the engine.

Each column definition is the name of the column followed by the datatype for that column, then one or more optional column constraints. SQLite is typeless. The datatype for the column does not restrict what data may be put in that column. All information is stored as null-terminated strings. The UNIQUE constraint causes an index to be created on the specified columns. This index must contain unique keys. The DEFAULT constraint specifies a default value to use when doing an INSERT.
Specifying a PRIMARY KEY normally just creates a UNIQUE index on the primary key. However, if primary key is on a single column that has datatype INTEGER, then that

column is used internally as the actual key of the B-Tree for the table. This means that the column may only hold unique integer values. (Except for this one case, SQLite ignores the datatype specification of columns and allows any kind of data to be put in a column regardless of its declared datatype.) If a table does not have an INTEGER PRIMARY KEY column, then the B-Tree key will be a automatically generated integer. The B-Tree key for a row can always be accessed using one of the special names "ROWID", "OID", or "_ROWID_". This is true regardless of whether or not there is an INTEGER PRIMARY KEY.

If the "TEMP" or "TEMPORARY" keyword occurs in between "CREATE" and "TABLE" then the table that is created is only visible to the process that opened the database and is automatically deleted when the database is closed. Any indices created on a temporary table are also temporary. Temporary tables and indices are stored in a separate file distinct from the main database file.

The optional conflict-clause following each constraint allows the specification of an alternative default constraint conflict resolution algorithm for that constraint. The default is abort ABORT. Different constraints within the same table may have different default conflict resolution algorithms. If an COPY, INSERT, or UPDATE command specifies a different conflict resolution algorithm, then that algorithm is used in place of the default algorithm specified in the CREATE TABLE statement. See the section titled ON CONFLICT 168 for additional information.

CHECK constraints are ignored in the current implementation. Support for CHECK constraints may be added in the future. As of version 2.3.0, NOT NULL, PRIMARY KEY, and UNIQUE constraints all work.

There are no arbitrary limits on the number of columns or on the number of constraints in a table. The total amount of data in a single row is limited to about 1 megabytes. (This limit can be increased to 16MB by changing a single #define in the source code and recompiling.)

The CREATE TABLE AS form defines the table to be the result set of a query. The names of the table columns are the names of the columns in the result.

The exact text of each CREATE TABLE statement is stored in the sqlite_master table. Every time the database is opened, all CREATE TABLE statements are read from the sqlite_master table and used to regenerate SQLite's internal representation of the table layout. If the original command was a CREATE TABLE AS then an equivalent CREATE TABLE statement is synthesized and store in sqlite_master in place of the original command. The text of CREATE TEMPORARY TABLE statements are stored in the sqlite_temp_master table.

9.5 CREATE TRIGGER

sql-statement ::= **CREATE TRIGGER** *trigger-name* [**BEFORE** | **AFTER**]
database-event **ON** *table-name*
trigger-action

sql-statement ::= **CREATE TRIGGER** *trigger-name* **INSTEAD OF**
database-event **ON** *view-name*
trigger-action

database-event ::= **DELETE** |
INSERT |
UPDATE |
UPDATE OF *column-list*

trigger-action ::= [**FOR EACH ROW**] [**WHEN** *expression*]
BEGIN
trigger-step **;** [*trigger-step* **;**]*
END

trigger-step ::= *update-statement* | *insert-statement* |
delete-statement | *select-statement*

The CREATE TRIGGER statement is used to add triggers to the database schema. Triggers are database operations (the trigger-action) that are automatically performed when a specified database event (the database-event) occurs.

A trigger may be specified to fire whenever a DELETE, INSERT or UPDATE of a particular database table occurs, or whenever an UPDATE of one or more specified columns of a table are updated.

At this time SQLite supports only FOR EACH ROW triggers, not FOR EACH STATEMENT triggers. Hence explicitly specifying FOR EACH ROW is optional. FOR EACH ROW implies that the SQL statements specified as trigger-steps may be executed (depending on the WHEN clause) for each database row being inserted, updated or deleted by the statement causing the trigger to fire.

Both the WHEN clause and the trigger-steps may access elements of the row being inserted, deleted or updated using references of the form "NEW.column-name" and "OLD.column-name", where column-name is the name of a column from the table that the trigger is associated with. OLD and NEW references may only be used in triggers on trigger-events for which they are relevant, as follows:

INSERT	NEW references are valid
UPDATE	NEW and OLD references are valid

DELETE	OLD references are valid

If a WHEN clause is supplied, the SQL statements specified as trigger-steps are only executed for rows for which the WHEN clause is true. If no WHEN clause is supplied, the SQL statements are executed for all rows.
The specified trigger-time determines when the trigger-steps will be executed relative to the insertion, modification or removal of the associated row.
An ON CONFLICT clause may be specified as part of an UPDATE or INSERT trigger-step. However if an ON CONFLICT clause is specified as part of the statement causing the trigger to fire, then this conflict handling policy is used instead.
Triggers are automatically dropped when the table that they are associated with is dropped.
Triggers may be created on views, as well as ordinary tables, by specifying INSTEAD OF in the CREATE TRIGGER statement. If one or more ON INSERT, ON DELETE or ON UPDATE triggers are defined on a view, then it is not an error to execute an INSERT, DELETE or UPDATE statement on the view, respectively. Thereafter, executing an INSERT, DELETE or UPDATE on the view causes the associated triggers to fire. The real tables underlying the view are not modified (except possibly explicitly, by a trigger program).
Example:
Assuming that customer records are stored in the "customers" table, and that order records are stored in the "orders" table, the following trigger ensures that all associated orders are redirected when a customer changes his or her address:

```
CREATE TRIGGER update_customer_address UPDATE OF address ON customers
  BEGIN
    UPDATE orders SET address = new.address WHERE customer_name = old.name;
  END;
```

With this trigger installed, executing the statement:

```
UPDATE customers SET address = '1 Main St.' WHERE name = 'Jack Jones';
```

causes the following to be automatically executed:

```
UPDATE orders SET address = '1 Main St.' WHERE customer_name = 'Jack Jones';
```

Note that currently, triggers may behave oddly when created on tables with INTEGER PRIMARY KEY fields. If a BEFORE trigger program modifies the INTEGER PRIMARY KEY field of a row that will be subsequently updated by the statement that causes the trigger to fire, then the update may not occur. The workaround is to declare the table with a PRIMARY KEY column instead of an INTEGER PRIMARY KEY column.

A special SQL function RAISE() may be used within a trigger-program, with the following syntax

raise-function ::= **RAISE (ABORT,** *error-message* **)** |
RAISE (FAIL, *error-message* **)** |
RAISE (ROLLBACK, *error-message* **)** |
RAISE (IGNORE)

When one of the first three forms is called during trigger-program execution, the specified ON CONFLICT processing is performed (either ABORT, FAIL or ROLLBACK) and the current query terminates. An error code of SQLITE_CONSTRAINT is returned to the user, along with the specified error message.

When RAISE(IGNORE) is called, the remainder of the current trigger program, the statement that caused the trigger program to execute and any subsequent trigger programs that would of been executed are abandoned. No database changes are rolled back. If the statement that caused the trigger program to execute is itself part of a trigger program, then that trigger program resumes execution at the beginning of the next step.

9.6 CREATE VIEW

sql-command ::= **CREATE VIEW** *view-name* **AS** *select-statement*

The CREATE VIEW command assigns a name to a pre-packaged SELECT statement. Once the view is created, it can be used in the FROM clause of another SELECT in place of a table name.

You cannot COPY, INSERT or UPDATE a view. Views are read-only.

9.7 DELETE

sql-statement ::= **DELETE FROM** *table-name* [**WHERE** *expr*]

The DELETE command is used to remove records from a table. The command consists of the "DELETE FROM" keywords followed by the name of the table from which records are to be removed.

Without a WHERE clause, all rows of the table are removed.

If a WHERE clause is supplied, then only those rows that match the expression are removed.

9.8 DROP INDEX

sql-command ::= **DROP INDEX** *index-name*

The DROP INDEX statement consists of the keywords "DROP INDEX" followed by the name of the index. The index named is completely removed from the disk. The only way to recover the index is to reenter the appropriate CREATE INDEX command.

9.9 DROP TABLE

sql-command ::= **DROP TABLE** *table-name*

The DROP TABLE statement consists of the keywords "DROP TABLE" followed by the name of the table. The table named is completely removed from the disk. The table can not be recovered. All indices associated with the table are also deleted.

9.10 DROP TRIGGER

sql-statement ::= **DROP TRIGGER** *trigger-name*

Used to drop a trigger from the database schema. Note that triggers are automatically dropped when the associated table is dropped.

9.11 DROP VIEW

sql-command ::= **DROP VIEW** *view-name*

The DROP VIEW statement consists of the keywords "DROP VIEW" followed by the name of the view. The view named is removed from the database. But no actual data is modified.

9.12 EXPLAIN

sql-statement::= **EXPLAIN** *sql-statement*

The EXPLAIN command modifier is a non-standard extension. The idea comes from a similar command found in PostgreSQL, but the operation is completely different.

If the EXPLAIN keyword appears before any other SQLite SQL command then instead of actually executing the command, the SQLite library will report back the sequence of virtual machine instructions it would have used to execute the command had the EXPLAIN keyword not been present. For additional information about virtual machine instructions see the architecture description or the documentation on available opcodes for the virtual machine.

9.13 EXPRESSION

expr ::= *expr binary-op expr* |
expr like-op expr |
unary-op expr |
(*expr* **)** |
column-name |
table-name **.** *column-name* |
literal-value |
function-name **(** *expr-list* | ***** **)** |
expr **ISNULL** |
expr **NOTNULL** |
expr [**NOT**] **BETWEEN** *expr* **AND** *expr* |
expr [**NOT**] **IN** **(** *value-list* **)** |
expr [**NOT**] **IN** **(** *select-statement* **)** |
(*select-statement* **)** |
CASE [*expr*] **(** **WHEN** *expr* **THEN** *expr* **)+** [**ELSE** *expr*] **END**

like-op ::= **LIKE** | **GLOB** | **NOT LIKE** | **NOT GLOB**

This section is different from the others. Most other sections of this document talks about a particular SQL command. This section does not talk about a standalone command but about "expressions" which are subcomponent of most other commands. SQLite understands the following binary operators, in order from highest to lowest precedence:

```
||
*     /     %
+     -
<<    >>    &     |
<     <=    >     >=
=     ==    !=    <>    IN
AND
OR
```

Supported unary operators are these:

```
-     +     !     ~
```

Any SQLite value can be used as part of an expression. For arithmetic operations, integers are treated as integers. Strings are first converted to real numbers using atof(). For comparison operators, numbers compare as numbers and strings compare using the strcmp () function. Note that there are two variations of the equals and not equals operators. Equals can be either `=` or `==`. The non-equals operator can be either != or <>. The ||

operator is "concatenate" - it joins together the two strings of its operands.

The LIKE operator does a wildcard comparison. The operand to the right contains the wildcards. A percent symbol % in the right operand matches any sequence of zero or more characters on the left. An underscore _ on the right matches any single character on the left. The LIKE operator is not case sensitive and will match upper case characters on one side against lower case characters on the other. (A bug: SQLite only understands upper/lower case for 7-bit Latin characters. Hence the LIKE operator is case sensitive for 8-bit iso8859 characters or UTF-8 characters. For example, the expression 'a' LIKE 'A' is TRUE but 'æ' LIKE 'Æ' is FALSE.)
The GLOB operator is similar to LIKE but uses the Unix file globing syntax for its wildcards. Also, GLOB is case sensitive, unlike LIKE. Both GLOB and LIKE may be preceded by the NOT keyword to invert the sense of the test.
A column name can be any of the names defined in the CREATE TABLE statement or one of the following special identifiers: "ROWID", "OID", or "_ROWID_". These special identifiers all describe the unique random integer key (the "row key") associated with every row of every table. The special identifiers only refer to the row key if the CREATE TABLE statement does not define a real column with the same name. Row keys act like read-only columns. A row key can be used anywhere a regular column can be used, except that you cannot change the value of a row key in an UPDATE or INSERT statement. "SELECT * ..." does not return the row key.
SELECT statements can appear in expressions as either the right-hand operand of the IN operator or as a scalar quantity. In both cases, the SELECT should have only a single column in its result. Compound SELECTs (connected with keywords like UNION or EXCEPT) are allowed. A SELECT in an expression is evaluated once before any other processing is performed, so none of the expressions within the select itself can refer to quantities in the containing expression.
When a SELECT is the right operand of the IN operator, the IN operator returns TRUE if the result of the left operand is any of the values generated by the select. The IN operator may be preceded by the NOT keyword to invert the sense of the test.
When a SELECT appears within an expression but is not the right operand of an IN operator, then the first row of the result of the SELECT becomes the value used in the expression. If the SELECT yields more than one result row, all rows after the first are ignored. If the SELECT yields no rows, then the value of the SELECT is NULL.
Both simple and aggregate functions are supported. A simple function can be used in any expression. Simple functions return a result immediately based on their inputs. Aggregate functions may only be used in a SELECT statement. Aggregate functions compute their result across all rows of the result set.

The functions shown below are available by default. Additional

abs(X)	Return the absolute value of argument X.
coalesce(X, Y,...)	Return a copy of the first non-NULL argument. If all arguments are NULL then NULL is returned.
glob(X,Y)	This function is used to implement the "Y GLOB X" syntax of SQLite.

last_insert_rowid()	Return the ROWID of the last row insert from this connection to the database. This is the same value that would be returned from the
length(X)	Return the string length of X in characters. If SQLite is configured to support UTF-8, then the number of UTF-8 characters is returned, not the number of bytes.
like(X,Y)	This function is used to implement the "Y LIKE X" syntax of SQL.
lower(X)	Return a copy of string X will all characters converted to lower case. The C library tolower() routine is used for the conversion, which means that this function might not work correctly on UTF-8 characters.
max(X,Y,...)	Return the argument with the maximum value. Arguments may be strings in addition to numbers. The maximum value is determined by the usual sort order. Note that max() is a simple function when it has 2 or more arguments but converts to an aggregate function if given only a single argument.
min(X,Y,...)	Return the argument with the minimum value. Arguments may be strings in addition to numbers. The minimum value is determined by the usual sort order. Note that min() is a simple function when it has 2 or more arguments but converts to an aggregate function if given only a single argument.
random(*)	Return a random integer between -2147483648 and +2147483647.

round(X)
round(X,Y)
	Round off the number X to Y digits to the right of the decimal point. If the Y argument is omitted, 0 is assumed.

substr(X,Y,Z)	Return a substring of input string X that begins with the Y-th character and which is Z characters long. The left-most character of X is number 1. If Y is negative the first character of the substring is found by counting from the right rather than the left. If SQLite is configured to support UTF-8, then characters indices refer to actual UTF-8 characters, not bytes.
upper(X)	Return a copy of input string X converted to all upper-case letters.
avg(X)	Return the average value of all X within a group.
count(X)	The first form return a count of the number of times that X is

count(*)	not NULL in a group. The second form (with no argument) returns the total number of rows in the group.
max(X)	Return the maximum value of all values in the group. The usual sort order is used to determine the maximum.
min(X)	Return the minimum value of all values in the group. The usual sort order is used to determine the minimum.
sum(X)	Return the numeric sum of all values in the group.

9.14 INSERT

sql-statement::= **INSERT** [**OR** *conflict-algorithm*] **INTO** *table-name* [**(** *column-list* **)**] **VALUES(** *value-list* **)** |
INSERT [**OR** *conflict-algorithm*] **INTO** *table-name* [**(** *column-list* **)**] *select-statement*

The INSERT statement comes in two basic forms. The first form (with the "VALUES" keyword) creates a single new row in an existing table. If no column-list is specified then the number of values must be the same as the number of columns in the table. If a column-list is specified, then the number of values must match the number of specified columns. Columns of the table that do not appear in the column list are fill with the default value, or with NULL if not default value is specified.
The second form of the INSERT statement takes it data from a SELECT statement. The number of columns in the result of the SELECT must exactly match the number of columns in the table if no column list is specified, or it must match the number of columns name in the column list. A new entry is made in the table for every row of the SELECT result. The SELECT may be simple or compound. If the SELECT statement has an ORDER BY clause, the ORDER BY is ignored.

The optional conflict-clause allows the specification of an alternative constraint conflict resolution algorithm to use during this one command. See the section titled ON CONFLICT 168 for additional information. For compatibility with MySQL, the parser allows the use of the single keyword "REPLACE" as an alias for "INSERT OR REPLACE".

9.15 ON CONFLICT clause

conflict-clause ::= **ON CONFLICT** *conflict-algorithm*

conflict-algorithm ::= **ROLLBACK** | **ABORT** | **FAIL** | **IGNORE** | **REPLACE**

The ON CONFLICT clause is not a separate SQL command. It is a non-standard clause that can appear in many other SQL commands. It is given its own section in this document because it is not part of standard SQL and therefore might not be familiar.
The syntax for the ON CONFLICT clause is as shown above for the CREATE TABLE, CREATE INDEX, and BEGIN TRANSACTION commands. For the COPY, INSERT, and UPDATE commands, the keywords "ON CONFLICT" are replaced by "OR", to make the syntax seem more natural. But the meaning of the clause is the same either way.
The ON CONFLICT clause specifies an algorithm used to resolve constraint conflicts. There are five choices: ROLLBACK, ABORT, FAIL, IGNORE, and REPLACE. The default algorithm is ABORT. This is what they mean:

ROLLBACK
When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.

ABORT
When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.

FAIL
When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.

IGNORE
When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned.

REPLACE
When a UNIQUE constraint violation occurs, the pre-existing row that is causing the constraint violation is removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally. No error is returned.
If a NOT NULL constraint violation occurs, the NULL value is replaced by the default

value for that column. If the column has no default value, then the ABORT algorithm is used.
The conflict resolution algorithm can be specified in three places, in order from lowest to highest precedence:
1. On a BEGIN TRANSACTION command.
2. On individual constraints within a CREATE TABLE or CREATE INDEX statement.
3. In the OR clause of a COPY, INSERT, or UPDATE command.

The algorithm specified in the OR clause of a COPY, INSERT, or UPDATE overrides any algorithm specified by a CREATE TABLE or CREATE INDEX. The algorithm specified within a CREATE TABLE or CREATE INDEX will, in turn, override the algorithm specified by a BEGIN TRANSACTION command. If no algorithm is specified anywhere, the ABORT algorithm is used.

9.16 PRAGMA

sql-statement ::= **PRAGMA** *name* = *value* |
PRAGMA *function*(*arg*)

The PRAGMA command is used to modify the operation of the SQLite library. The pragma command is experimental and specific pragma statements may be removed or added in future releases of SQLite. Use this command with caution.

The current implementation supports the following pragmas:

- **PRAGMA cache_size;**
  PRAGMA cache_size = *Number-of-pages***;**

Query or change the maximum number of database disk pages that SQLite will hold in memory at once. Each page uses about 1.5K of memory. The default cache size is 2000. If you are doing UPDATEs or DELETEs that change many rows of a database and you do not mind if SQLite uses more memory, you can increase the cache size for a possible speed improvement.
When you change the cache size using the cache_size pragma, the change only endures for the current session. The cache size reverts to the default value when the database is closed and reopened. Use the default_cache_size pragma to check the cache size permanently.

- **PRAGMA count_changes = ON;**
  PRAGMA count_changes = OFF;

When on, the COUNT_CHANGES pragma causes the callback function to be invoked once for each DELETE, INSERT, or UPDATE operation. The argument is the number of rows that were changed.
This pragma may be removed from future versions of SQLite. Consider using the sqlite_changes() API function instead.

- **PRAGMA default_cache_size;**
  PRAGMA default_cache_size = *Number-of-pages***;**

Query or change the maximum number of database disk pages that SQLite will hold in memory at once. Each page uses about 1.5K of memory. This pragma works like the cache_size pragma with the addition feature that it changes the cache size persistently. With this pragma, you can set the cache size once and that setting is retained and reused every time you reopen the database.

- **PRAGMA default_synchronous;**
  PRAGMA default_synchronous = ON;
  PRAGMA default_synchronous = OFF;

Query or change the setting of the "synchronous" flag in the database. When synchronous is on (the default), the SQLite database engine will pause at critical moments to make sure that data has actually be written to the disk surface. (In other words, it invokes the equivalent of the fsync() system call.) In synchronous mode, a SQLite database should be fully recoverable even if the operating system crashes or power is interrupted unexpectedly. The penalty for this assurance is that some database operations take longer because the engine has to wait on the (relatively slow) disk drive. The alternative is to turn synchronous off. With synchronous off, SQLite continues processing as soon as it has handed data off to the operating system. If the application running SQLite crashes, the data will be safe, but the database could (in theory) become corrupted if the operating system crashes or the computer suddenly loses power. On the other hand, some operations are as much as 50 or more times faster with synchronous off.
This pragma changes the synchronous mode persistently. Once changed, the mode stays as set even if the database is closed and reopened. The synchronous pragma does the same thing but only applies the setting to the current session.

- **PRAGMA empty_result_callbacks = ON;**
  PRAGMA empty_result_callbacks = OFF;

When on, the EMPTY_RESULT_CALLBACKS pragma causes the callback function to be invoked once for each query that has an empty result set. The third "argv" parameter to the callback is set to NULL because there is no data to report. But the second "argc" and fourth "columnNames" parameters are valid and can be used to determine the number and names of the columns that would have been in the result set had the set not been empty.

- **PRAGMA full_column_names = ON;**
  PRAGMA full_column_names = OFF;

The column names reported in a SQLite callback are normally just the name of the column itself, except for joins when "TABLE.COLUMN" is used. But when full_column_names is turned on, column names are always reported as "TABLE.COLUMN" even for simple queries.

- **PRAGMA index_info(***index-name***);**

For each column that the named index references, invoke the callback function once with information about that column, including the column name, and the column number.

- **PRAGMA index_list(***table-name***);**

For each index on the named table, invoke the callback function once with information about that index. Arguments include the index name and a flag to indicate whether or not the index must be unique.

- **PRAGMA parser_trace = ON;**

PRAGMA parser_trace = OFF;

Turn tracing of the SQL parser inside of the SQLite library on and off. This is used for debugging. This only works if the library is compiled without the NDEBUG macro.

- **PRAGMA integrity_check;**

The command does an integrity check of the entire database. It looks for out-of-order records, missing pages, and malformed records. If any problems are found, then a single string is returned which is a description of all problems. If everything is in order, "ok" is returned.

- **PRAGMA synchronous;**
  PRAGMA synchronous = ON;
  PRAGMA synchronous = OFF;

Query or change the setting of the "synchronous" flag in the database for the duration of the current database connect. The synchronous flag reverts to its default value when the database is closed and reopened. For additional information on the synchronous flag, see the description of the default_synchronous pragma.

- **PRAGMA table_info(*table-name*);**

For each column in the named table, invoke the callback function once with information about that column, including the column name, data type, whether or not the column can be NULL, and the default value for the column.

- **PRAGMA vdbe_trace = ON;**
  PRAGMA vdbe_trace = OFF;

Turn tracing of the virtual database engine inside of the SQLite library on and off. This is used for debugging.
No error message is generated if an unknown pragma is issued.
Unknown pragmas are ignored.

9.17 REPLACE

sql-statement ::= **REPLACE INTO** *table-name* [**(** *column-list* **)**] **VALUES (** *value-list* **)** |
REPLACE INTO *table-name* [**(** *column-list* **)**] *select-statement*

The REPLACE command is an alias for the "INSERT OR REPLACE" variant of the INSERT command 167. This alias is provided for compatibility with MySQL. See the INSERT command 167 documentation for additional information.

9.18 SELECT

sql-statement ::= **SELECT** [**DISTINCT**] *result* [**FROM** *table-list*]
[**WHERE** *expr*]
[**GROUP BY** *expr-list*]
[**HAVING** *expr*]
[*compound-op select*]*
[**ORDER BY** *sort-expr-list*]
[**LIMIT** *integer* [**OFFSET** *integer*]]

result ::= *result-column* [**,** *result-column*]*

result-column ::= ***** | *table-name* **. *** | *expr* [[**AS**] *string*]

table-list ::= *table* [*join-op table join-args*]*

table ::= *table-name* [**AS** *alias*] |
(*select* **)** [**AS** *alias*]

join-op ::= **,** | [**NATURAL**] [**LEFT** | **RIGHT** | **FULL**] [**OUTER** |
INNER] **JOIN**

join-args ::= [**ON** *expr*] [**USING (** *id-list* **)**]

sort-expr-list ::= *expr* [*sort-order*] [**,** *expr* [*sort-order*]]*

sort-order ::= **ASC** | **DESC**

compound_op ::= **UNION** | **UNION ALL** | **INTERSECT** | **EXCEPT**

The SELECT statement is used to query the database. The result of a SELECT is zero or more rows of data where each row has a fixed number of columns. The number of columns in the result is specified by the expression list in between the SELECT and FROM keywords. Any arbitrary expression can be used as a result. If a result expression is * then all columns of all tables are substituted for that one expression. If the expression is the name of a table followed by .* then the result is all columns in that one table.
The query is executed against one or more tables specified after the FROM keyword. If multiple tables names are separated by commas, then the query is against the cross join of the various tables. The full SQL-92 join syntax can also be used to specify joins. A sub-query in parentheses may be substituted for any table name in the FROM clause. The entire FROM clause may be omitted, in which case the result is a single row

consisting of the values of the expression list.
The WHERE clause can be used to limit the number of rows over which the query operates.
The GROUP BY clauses causes one or more rows of the result to be combined into a single row of output. This is especially useful when the result contains aggregate functions. The expressions in the GROUP BY clause do not have to be expressions that appear in the result. The HAVING clause is similar to WHERE except that HAVING applies after grouping has occurred. The HAVING expression may refer to values, even aggregate functions, that are not in the result.
The ORDER BY clause causes the output rows to be sorted. The argument to ORDER BY is a list of expressions that are used as the key for the sort. The expressions do not have to be part of the result for a simple SELECT, but in a compound SELECT each sort expression must exactly match one of the result columns. Each sort expression may be optionally followed by ASC or DESC to specify the sort order.
The LIMIT clause places an upper bound on the number of rows returned in the result. A LIMIT of 0 indicates no upper bound. The optional OFFSET following LIMIT specifies how many rows to skip at the beginning of the result set.

A compound SELECT is formed from two or more simple SELECTs connected by one of the operators UNION, UNION ALL, INTERSECT, or EXCEPT. In a compound SELECT, all the constituent SELECTs must specify the same number of result columns. There may be only a single ORDER BY clause at the end of the compound SELECT. The UNION and UNION ALL operators combine the results of the SELECTs to the right and left into a single big table. The difference is that in UNION all result rows are distinct where in UNION ALL there may be duplicates. The INTERSECT operator takes the intersection of the results of the left and right SELECTs. EXCEPT takes the result of left SELECT after removing the results of the right SELECT. When three are more SELECTs are connected into a compound, they group from left to right.

9.19 UPDATE

sql-statement ::= **UPDATE** [**OR** *conflict-algorithm*] *table-name*
SET *assignment* [**,** *assignment*]
[**WHERE** *expr*]

assignment ::= *column-name* **=** *expr*

The UPDATE statement is used to change the value of columns in selected rows of a table. Each assignment in an UPDATE specifies a column name to the left of the equals sign and an arbitrary expression to the right. The expressions may use the values of other columns. All expressions are evaluated before any assignments are made. A WHERE clause can be used to restrict which rows are updated.

The optional conflict-clause allows the specification of an alternative constraint conflict resolution algorithm to use during this one command. See the section titled ON CONFLICT 168 for additional information.

9.20 VACUUM

sql-statement::= **VACUUM** [*index-or-table-name*]

The VACUUM command is a SQLite extension modeled after a similar command found in PostgreSQL. If VACUUM is invoked with the name of a table or index then it is suppose to clean up the named table or index. In version 1.0 of SQLite, the VACUUM command would invoke gdbm_reorganize() to clean up the backend database file. Beginning with version 2.0 of SQLite, GDBM is no longer used for the database backend and VACUUM has become a no-op.

Index