

SQLite Expert 5.5

Table of contents

Contents	4
Introduction	4
What's new?	5
What is SQLite?	9
What is SQLite Expert?	10
Feature comparison	11
Terms and Conditions	14
ICU License	16
Ordering	17
System requirements, installation and registration	17
Credits	18
Using SQLite3 databases	18
Creating databases	19
Opening an existing database	19
Closing a database	20
Renaming database alias	20
Displaying database information and internal parameters	20
Performing maintenance operations	22
Using encrypted databases	23
Attaching and Detaching databases	24
Loading and Unloading SQLite extensions	25
Comparing the DDL or Schema of two databases	26
Performing an online backup of a database	28
Editing tables	29
Creating a table	30
Editing a table	31
Renaming a table	32
Deleting a table or a group of tables	32
Editing columns	32
Editing indexes	35
Editing primary keys	38
Editing foreign keys	39
Editing constraints	41
Editing table triggers	45
Copying fields, indexes, constraints and triggers using the clipboard	47
Applying the changes after editing a table	47
Limitations	48
Editing virtual tables	49
Creating a virtual table	49
Editing a virtual table	51
Renaming a virtual table	53
Deleting a virtual table	53
Editing views	53
Creating a view	53
Editing a view	54
Deleting a view or a group of views	55
Using SQL scripts	56

Executing SQL scripts	56
Cancelling the execution of a long-running SQL script	57
Saving and loading SQL scripts	57
Creating a View from an SQL script	57
Using parameters in SQL scripts	58
Code Completion	59
Viewing and editing data	63
Viewing table data	63
Editing table data	64
Editing live queries	67
Copying data between tables using the clipboard	68
Using the SQL Query Builder	69
The Query Builder Layout	69
Building SQL queries using the Query Builder	71
Generating Views using the Query Builder	78
Importing and Exporting Data	78
Exporting data to Excel	79
Exporting data to XML	79
Exporting data to JSON	79
Exporting data to HTML	79
Exporting data to Open Document Spreadsheet (ODS)	79
Exporting data to SQL	79
Exporting data to a text file	79
Importing data from a text file	80
Using the Data Transfer Wizard	81
Importing data from another SQLite database	82
Importing data from a script	91
Importing data from an ADO data source	92
Exporting data to another SQLite database	95
Exporting data to a script	98
Copying tables between databases using the clipboard	102
Using Transactions	103
Starting a transaction	103
Committing a transaction	103
Rolling back a transaction	103
Creating a Savepoint	104
Releasing a Savepoint	104
Rolling back to a Savepoint	104
Extending SQLite Expert using Scripting	104
Scripting examples	105
Lua scripting API	107
Pascal scripting API	112
Data types	117
Changing options	119
Getting Help	122
Integrated Help	122
SQLite Expert on the Web	122
SQLite on the Web	123
Sending feedback	123
Known Issues	123

Contents

SQLite Expert version 5.5

Copyright © 2024 Coral Creek Software.

- » [Introduction](#)
- » [Using SQLite3 databases](#)
- » [Editing tables](#)
- » [Editing views](#)
- » [Viewing and editing data](#)
- » [Using SQL scripts](#)
- » [Using the SQL Query Builder](#)
- » [Using the Data Transfer Wizard](#)
- » [Using Transactions](#)
- » [Data types](#)
- » [Extending SQLite Expert using Scripting](#)
- » [Changing options](#)
- » [Getting Help](#)
- » [Known Issues](#)

Introduction

- » [What's new?](#)
- » [What is SQLite?](#)
- » [What is SQLite Expert?](#)
- » [Feature comparison](#)
- » [Terms and Conditions](#)
- » [Ordering](#)
- » [System requirements, installation and registration](#)

» [Credits](#)

What's new?

New in version 5.0

New user interface

The user interface has been completely redesigned using Developer Express VCL Components.

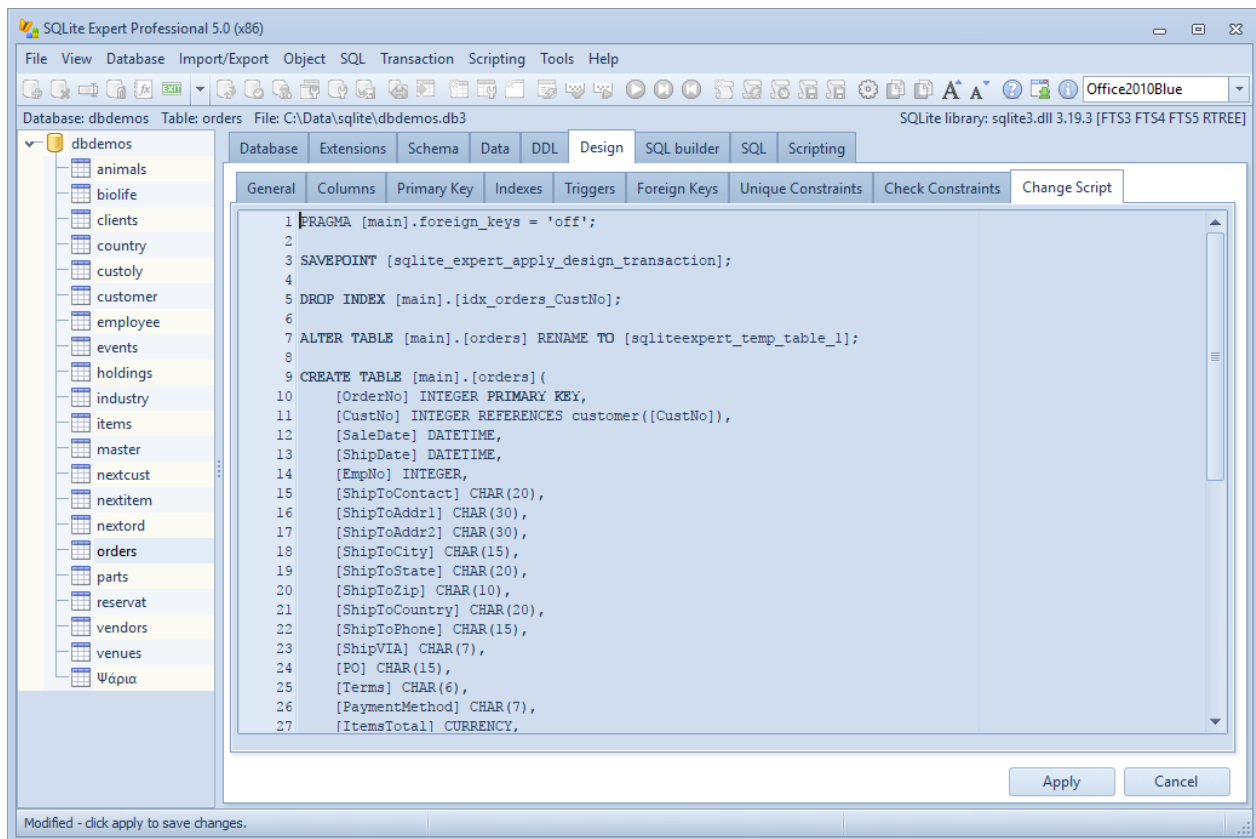
New in version 4.0

New editors

The table/view/virtual table editors have been completely redesigned using a new recursive-descent parser based on COCO/R.

SQL Change script

The SQL change script is now available before applying changes to a database object:



New Dataset Export options

Export dataset to Open Document Spreadsheet (ODS).

Export dataset to SQL script.

64-bit version

Starting with version 4.0, both 32-bit and 64-bit versions of SQLite Expert are now available for both the Personal and Professional Editions.

Other features and changes

Experimental SQL formatter on the SQL and DDL tabs.

Database schema compare.

Settings are now stored in a JSON file.

Statically linked SQLite library is no longer included in version 4.0.

New in version 3.5

Improved license registration process

Added registration dialog in order to make the registration process more intuitive.

The license key is now stored in the common data folder as well, for better Windows 8 compatibility.

New license key format.

Documentation

Documentation is now included in both CHM and PDF format.

Other features:

Support for tables without rowid.

Support for partial indexes.

New in version 3.3

Support for virtual tables

Starting with version 3.3 SQLite Expert supports visual editing of virtual tables, including FTS3 and FTS4.

New in version 3.0

Printing support

Send the contents of any grid or editor to printer.

customer

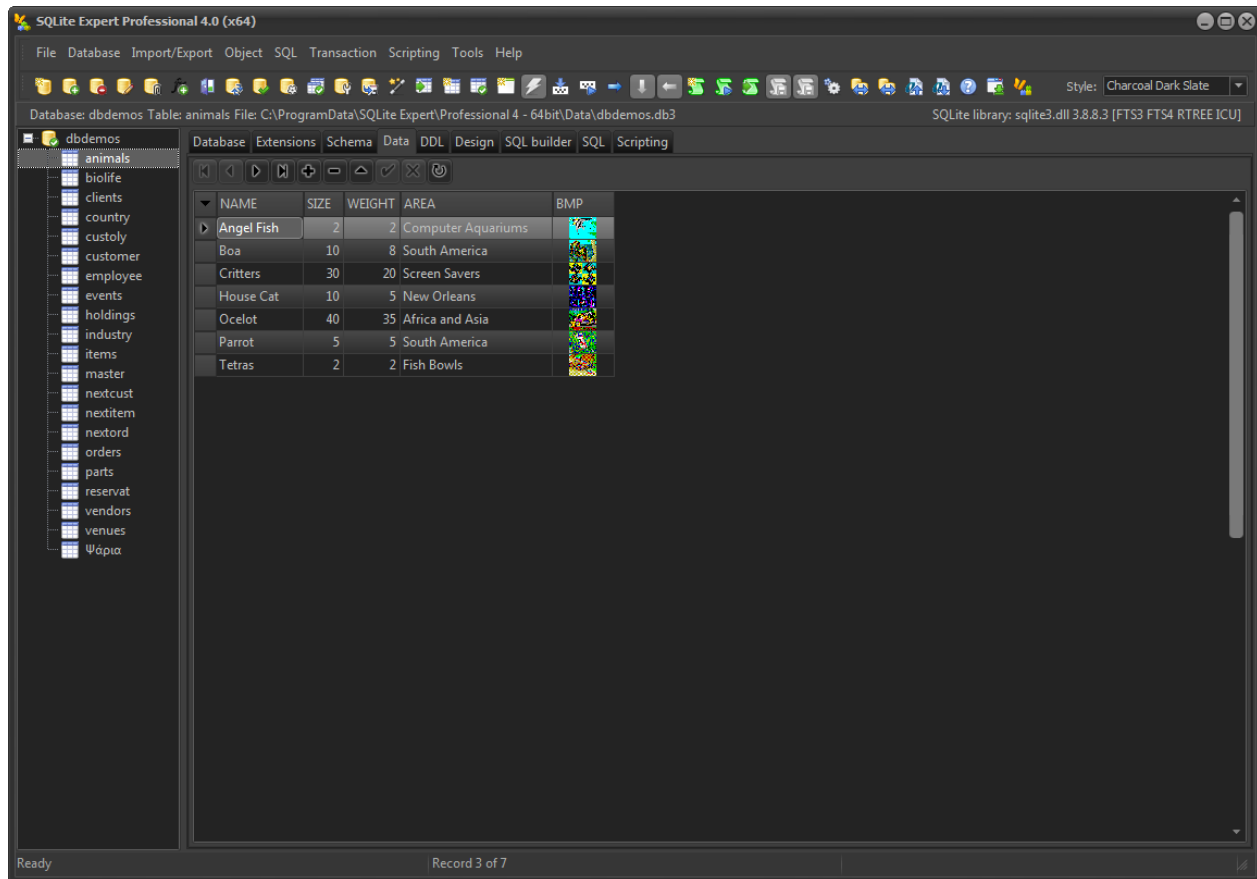
CustNo	Company	Addr1	Addr2	City	State	Zip
1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	Suite 103	Kapaa Kauai	HI	94766-1234
1231	Unisco	PO Box Z-547		Freeport		
1351	Sight Diver	1 Neptune Lane		Kato Paphos		
1354	Cayman Divers World Unlimited	PO Box 541		Grand Cayman		
1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj		Christiansted	St. Croix	00820
1380	Blue Jack Aqua Center	23-738 Paddington Lane	Suite 310	Waipahu	HI	99776
1384	VIP Divers Club	32 Main St.		Christiansted	St. Croix	02800
1510	Ocean Paradise	PO Box 8745		Kailua-Kona	HI	94756
1513	Fantastique Aquatica	Z32 999 #12A-77 A.A.		Bogota		
1551	Marmot Divers Club	872 Queen St.		Kitchener	Ontario	G3N 2E1
1560	The Depth Charge	15243 Underwater Fwy.		Marathon	FL	35003
1563	Blue Sports	203 12th Ave. Box 746		Giribaldi	OR	91187
1624	Makai SCUBA Club	PO Box 8534		Kailua-Kona	HI	94756
1645	Action Club	PO Box 5451-F		Sarasota	FL	32274
1651	Jamaica SCUBA Centre	PO Box 68		Negril	Jamaica	
1680	Island Finders	6133 1/3 Stone Avenue		St Simons Isle	GA	32521
1984	Adventure Undersea	PO Box 744		Belize City		
2118	Blue Sports Club	63365 Nez Perce Street		Largo	FL	34684
2135	Frank's Divers Supply	1455 North 44th St.		Eugene	OR	90427
2156	Davy Jones' Locker	246 South 16th Place		Vancouver	BC	K8V 9P1

Page 1 of 4

Ready record 1 of 33

Skins library

Customize the look and feel of the application using skins. In version 4.0, skins have been replaced with VCL Styles:



New in version 2.0

Improved Unicode support

At first glance, version 2.0 is not much different from the latest version 1.7. However, version 2.0 brings major internal improvements with respect to Unicode support. SQLite Expert is written in Delphi, and Delphi versions older than 2009 did not include native Unicode support. SQLite Expert versions 1.x included Unicode support by using a freeware library - UTF8VCL available from SourceForge. This project was in alpha stage of development and seems to be currently unmaintained.

The latest version of Delphi (Delphi 2009) finally added complete support for the Unicode character set, so it was the logical choice to upgrade the source code of SQLite Expert to Delphi 2009. This process took a few months because of the major internal differences between the two compilers. During this migration, the source code was gradually modified to compile with Delphi 2009 while still being compatible with Delphi 2007. As a result of this process, versions 1.7 and 2.0 of SQLite Expert look very similar but are actually much different internally. The native string type in Delphi 2007 was the AnsiString while in Delphi 2009 it is the UnicodeString (using UTF-16 encoding).

The following differences between versions 1.7 and 2.0 are related to the improved Unicode support:

1. The default type mapping for CHAR and TEXT is now WideString instead of String. That means, all the text is now converted to UTF-16 when it is loaded from the database. In version 1.x it was converted to UTF-8. It is recommended not to change this setting.

2. If you wish to change the default type mappings for other declared types so they are handled internally as strings, you should map them to WideString, not to String as in version 1.x.
3. More encoding options on import/export from/to text files. An encoding detection when importing a text file has been included that seems to work reasonably well in most cases.
4. The Encoding options have been renamed to "Unicode (UTF-8, UTF-16)" and "Default ANSI code page".

Statically linked sqlite3 library

Starting with version 2.0, SQLite Expert includes a statically linked sqlite3 library, so it can be used without an external sqlite3.dll. However, it is still possible to use an external library, which might be a useful feature in case you want to use a more recent sqlite library, or one that supports encryption (note that encryption support is only available in the Professional Edition).

No bug fixes between 1.7 and 2.0

During the migration from Delphi 2007 to Delphi 2009, all the bug fixes have been implemented in both versions 1.7 and 2.0, so if a bug exists in the latest 1.7 version and is not Unicode related, there is a good chance it is still present in version 2.0. However, at the time of the 2.0 released, all the reported bugs have been fixed. Also, after the release of 2.0 no more bug fixes will be implemented in 1.7, so it is recommended to upgrade to version 2.0 to keep up with the latest updates.

What is SQLite?

What is SQLite?

SQLite (www.sqlite.org) is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine. Features include:

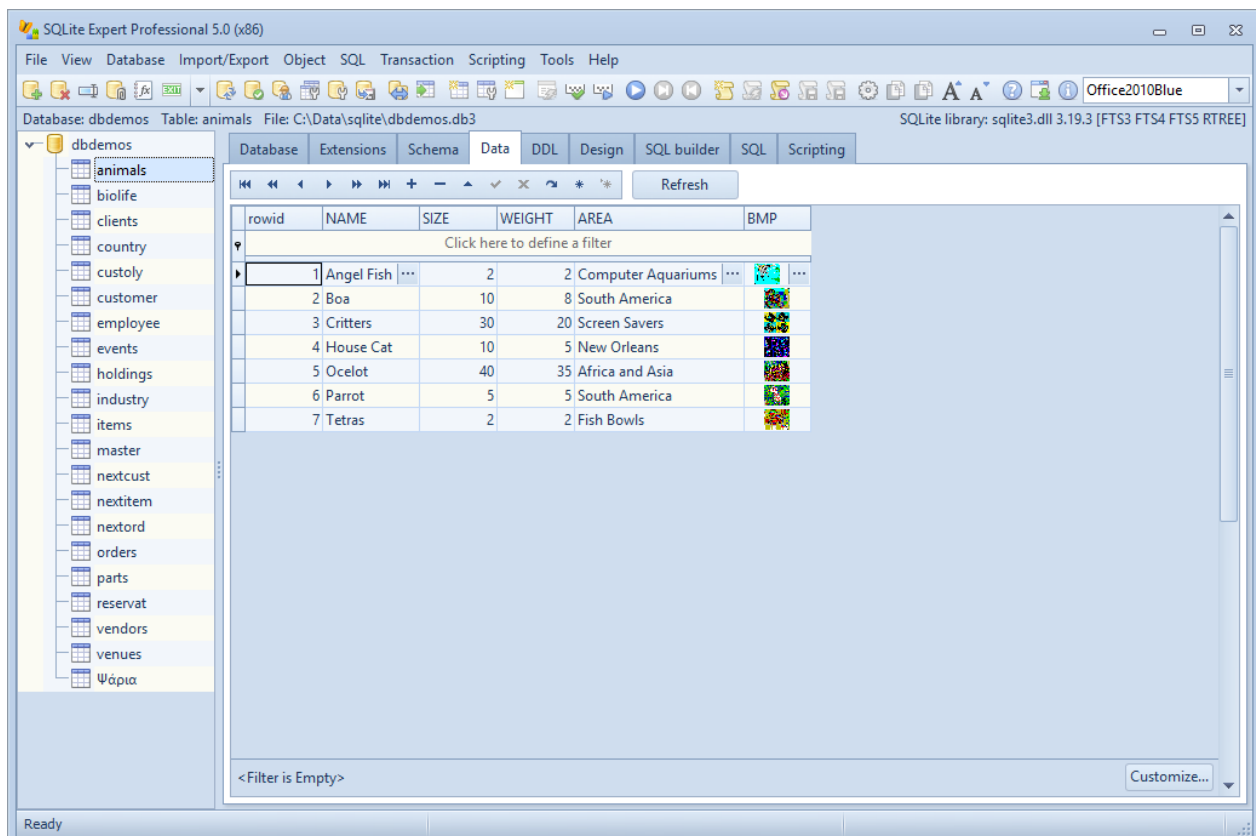
- < Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- < Zero-configuration - no setup or administration needed.
- < Implements most of SQL92.
- < A complete database is stored in a single disk file.
- < Database files can be freely shared between machines with different byte orders.
- < Supports databases up to 2 terabytes (241 bytes) in size.
- < Sizes of strings and BLOBs limited only by available memory.
- < Small code footprint: less than 250 kB fully configured or less than 150 kB with optional features omitted.

- ⟨ Faster than popular client/server database engines for most common operations.
- ⟨ Simple, easy to use API.
- ⟨ TCL bindings included. Bindings for many other languages available separately.
- ⟨ Well-commented source code with over 95% test coverage.
- ⟨ Self-contained: no external dependencies.
- ⟨ Sources are in the public domain. Use for any purpose.

What is SQLite Expert?

SQLite Expert version 5.0

SQLite Expert is a powerful visual tool that enables you to easily administer your [SQLite 3](#) databases and gain significantly better visibility into how your databases are operating. SQLite Expert integrates database management and maintenance into a single, seamless environment, with a clear and intuitive graphical user interface.



Features include:

- ⟨ [Edit tables](#) and [views](#) visually, mostly without writing a line of SQL. Easily restructure fields,

foreign keys, indexes, constraints, triggers without losing data already existing in the tables. [Copy](#) fields, foreign keys, indexes, constraints and triggers between tables using the clipboard.

⟨ Build SQL scripts and generate views visually using the integrated [Query Builder](#).

⟨ [Access encrypted databases](#) (Professional Edition).

⟨ [Create SQLite3 databases](#), view and change database parameters, check database integrity and vacuum (compact) database.

⟨ [Import or Export data](#) from/to SQL script, another SQLite database, ADO data sources, text files, or Excel.

⟨ [Display and edit data](#) in the grid, including BLOB and image fields. Currently supports BMP, JPG, PNG, GIF and ICO image formats. BLOB fields can be edited with the integrated hex editor. Copy records between tables using the clipboard. Copy tables and views between databases using the clipboard or using drag and drop operations.

⟨ [Execute SQL queries](#). Supports multiple SQL statements in the same query.

⟨ [Attach and detach databases](#).

⟨ Cancel the execution of long-running SQL scripts.

⟨ In-grid editing of live queries.

⟨ Save/Load SQL scripts to/from file.












⟨ [Transaction support](#). SQLite supports nested transactions starting with version 3.6.8.

⟨ [Data mapping](#). SQLite Expert supports custom data types, and 40 predefined data types.

⟨ Extend SQLite Expert capabilities using [Lua and Pascal scripting](#).

Starting with version 1.1.4, SQLite Expert is available in two flavors: **Personal Edition** (freeware) and **Professional Edition** (shareware). **The content of this help file applies to the Professional Edition.** For a comparison between the Personal and Professional editions please see the [Feature Matrix](#) available on the SQLite Expert homepage.

Feature comparison

	Personal	Professional
General		
Customize the appearance of the application using skins		
Send the contents of any grid to printer or export it to a PDF file		
Managing SQLite3 databases		
Create database		
View and change database parameters		
Register frequently used databases for using in SQLite Expert		
Check integrity		

	Personal	Professional
Vacuum		
Reindex all tables		
Transaction support		
Supports attached databases		
Supports encrypted databases		
Supports SQLite extensions		
Supports SQLite auto extensions		
Includes ICU extension		
View database schema report		
Export to Excel, ODS		
Export to XML, HTML, JSON		
Import and export from and to text files (CSV, TSV)		
Import data from ADO data source		
Transfer data between SQLite databases		
Export data to SQL script		
Database DDL compare tool		
Online backup of a database		
Managing tables		
Create tables		
Delete tables		
Rename tables		
Reindex tables		
Restructure tables without losing existing data		
View, add, delete, modify, reorder fields		
Copy/Paste fields via clipboard		
View, add, delete, modify foreign keys		
Copy/Paste foreign keys via clipboard		
View, add, delete, modify indexes		
Copy/Paste indexes via clipboard		
View constraints		
Add, delete, modify constraints		
Copy/Paste constraints via clipboard		
View triggers		
Add, delete, modify triggers		
Copy/Paste triggers via clipboard		
Supports temporary tables		
Supports virtual tables including RTREE, FTS3 and FTS4		
Managing views		
Create views		
Delete views		

	Personal	Professional
Rename views		
View triggers		
Add, delete, modify triggers		
Copy/Paste triggers via clipboard		
Supports temporary views		
Executing SQL scripts		
Execute SQL scripts		
Stop long running queries		
SQL code formatting (experimental)		
SQL syntax highlighting		
SQL Code Completion		
Save and load SQL scripts		
Supports parameters in SQL scripts		
Query Builder		
Visual building of complex SQL queries		
User-friendly interface with drag & drop support		
Visual building of sub-queries and unions		
Visual criteria constructing tool for Where and Having clauses		
Save and load SQL scripts		
Displaying and editing data		
Display and edit data in the grid		
Display data as text		
Data filtering		
Image editor (supports BMP, JPEG, PNG, GIF and ICO)		
BLOB editor		
Text editor		
Copy/Paste records between tables via clipboard		
Copy/Paste tables between databases via clipboard		
Copy tables between databases using drag and drop		
Generate HTML Reports		
Extending SQLite Expert using scripting		
Lua scripting support		
Pascal scripting support		
Lua and Pascal example scripts		
Data types		
40 predefined data types		
Custom data types		

Terms and Conditions

SQLite Expert Terms and Conditions

Last Updated: Oct 30, 2024

Thank you for choosing SQLite Expert. By downloading, installing, and using SQLite Expert (“Software”), you agree to be bound by the following Terms and Conditions. Please read them carefully.

1. Acceptance of Terms

By accessing or using SQLite Expert, you confirm that you have read, understood, and agree to abide by these Terms and Conditions. If you do not agree, you may not use the Software.

2. License - Personal Edition

This Software is being distributed as Freeware. It may be freely used, copied and distributed as long as it is not sold, and all original files are included, including this license. You are NOT allowed to make a charge for distributing this Software (either for profit or merely to recover your media and distribution costs) whether as a stand-alone product, or as part of a compilation or anthology, without explicit prior written permission.

3. License - Professional Edition

3.1. License Grant

Subject to the terms and conditions outlined herein, Coral Creek Software grants you a non-exclusive, non-transferable license to use SQLite Expert solely for your internal purposes. You may install and use the Software according to the licensing terms purchased.

3.2. License Restrictions

You shall not:

Copy, modify, distribute, or create derivative works based on the Software.

Reverse engineer, decompile, disassemble, or otherwise attempt to derive source code from the Software.

Rent, lease, lend, or sublicense the Software.

4. Subscription and Fees

SQLite Expert is offered on a subscription basis. By subscribing, you agree to pay the subscription fees as outlined at the time of purchase. Subscription fees may vary based on license type and additional features.

4.1. Automatic Renewal

Your subscription may automatically renew at the end of each term unless canceled or switched to

manual renewal. You may cancel your subscription anytime by contacting our support team or managing your account through the provided portal.

4.2. Refund Policy

Our 30-day refund policy allows customers to request a full refund within 30 days of purchase if they are not fully satisfied with our product. To qualify, customers must provide a valid purchase receipt and describe any issues they encountered. Refunds will be issued to the original payment method within 7-10 business days of approval. After 30 days, refunds are not available.

5. Intellectual Property

SQLite Expert, including but not limited to all software, documentation, trademarks, and other proprietary materials, is the exclusive property of Coral Creek Software. These Terms and Conditions do not grant you any ownership rights, and you agree not to use or reproduce any Coral Creek Software trademarks without permission.

6. Privacy

Your privacy is important to us. Please refer to our Privacy Policy for information on how we collect, use, and disclose your data.

7. Updates and Support

7.1. Software Updates

We may, at our discretion, provide updates or upgrades to the Software to improve functionality, security, or performance. You agree to install such updates to ensure optimal use of SQLite Expert.

7.2. Support

Technical support is provided to users with an active subscription. Support requests can be submitted through support@sqliteexpert.com.

8. Disclaimer of Warranties

The Software is provided “as is,” and Coral Creek Software makes no warranties, express or implied, regarding the Software. This includes but is not limited to implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

9. Limitation of Liability

To the fullest extent permitted by applicable law, Coral Creek Software shall not be liable for any damages, including but not limited to direct, indirect, incidental, consequential, or punitive damages arising from the use of or inability to use the Software, even if advised of the possibility of such damages.

10. Termination

This license is effective until terminated. You may terminate this agreement at any time by ceasing all use of and deleting all copies of the Software. Coral Creek Software reserves the right to terminate this license if you breach any terms herein.

11. Governing Law

These Terms and Conditions are governed by and construed in accordance with the laws of Florida, United States. Any disputes arising under these Terms and Conditions shall be subject to the exclusive jurisdiction of the courts located in Escambia County, Florida.

12. Changes to Terms and Conditions

Coral Creek Software reserves the right to update or modify these Terms and Conditions at any time without prior notice. Continued use of the Software after modifications are posted constitutes acceptance of the revised Terms and Conditions.

If you have any questions or concerns regarding these Terms and Conditions, please contact us at support@sqliteexpert.com.

ICU License

ICU License - ICU 1.8.1 and later

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2014 International Business Machines Corporation and others

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

Ordering

Register online

You may use SQLite Expert free of charge while evaluating it. The evaluation period is limited to 30 days. After the expiration of the evaluation period you must either buy this software or remove it from your computer.

Software registrations are currently handled by [FastSpring](#), a company that specializes in software registrations. All ordering pages are on secure SSL servers, ensuring safety of your confidential information. All major credit cards are accepted.

To order select **Help » Buy Online** or visit www.sqliteexpert.com/order.html and follow the instructions on the page.

After your order has been processed you will receive a license key file by email. To complete the registration process, copy the license file in the installation directory.

System requirements, installation and registration

System Requirements

Operating System: Windows 2000/XP/2003/Vista/7/8/10/11;

Installation

In order to install **SQLite Expert**, download and run SQLiteExpertSetup.exe and follow the on-screen instructions. The setup program will ask you to select options and where to install **SQLite Expert**. If you are unsure, you should accept the defaults.

Once installed, you can start **SQLite Expert** from the **Start** Menu.

Note:

You need to have administrator rights to install SQLite Expert.

To see if you are logged on to Windows as an administrator, please follow the steps below:

1. From your Windows taskbar, click "Start" > "Run."
2. Type [control userpasswords2] into the text box and click "OK."
3. A User Accounts dialog box will open. If the dialog box has two tabs, you are logged in as an administrator. If the dialog box doesn't have any tabs and prompts you for a password, you are not logged in as an administrator.

If you do not have administrator rights on your computer, please contact your system administrator.

Registration

After purchasing and receiving the license key, register by selecting the **Help » Register** option from the main menu and following the instructions on the screen.

Credits

Credits

SQLite Expert is Copyright © 2024 Coral Creek Software.

This program contains or uses the following software components:

- ⟨ DevExpress VCL Components (<http://www.devexpress.com/>)
- ⟨ Virtual Treeview by Mike Lischke (<http://www.soft-gems.net/>)
- ⟨ SynEdit 2.0.1 beta by Michael Hieke (<http://synedit.sourceforge.net>)
- ⟨ TMPHexEditor © Markus Stephany (<http://www.mirkes.de/>)
- ⟨ Inno Setup © 1997-2005 Jordan Russell (<http://www.jrsoftware.org/>)
- ⟨ The FastCode Project (<http://fastcode.sourceforge.net/>)
- ⟨ FastMM by Pierre le Riche (<http://sourceforge.net/projects/fastmm/>)
- ⟨ Vcl-Styles-Utils by Rodrigo Ruz (<https://code.google.com/p/vcl-styles-utils/>)

Using SQLite3 databases

Creating and editing SQLite3 databases, and other commands for working with databases.

- » [Creating databases](#)
- » [Opening an existing database](#)
- » [Closing a database](#)
- » [Editing database registration](#)

- » [Displaying database information and internal parameters](#)
- » [Performing maintenance operations](#)
- » [Using encrypted databases](#)
- » [Attaching and Detaching databases](#)
- » [Loading and Unloading SQLite extensions](#)
- » [Comparing the DDL of two databases](#)
- » [Performing an online backup of a database](#)

Creating databases

To create a new SQLite3 database, select

File » New Database

You will be prompted to select the file name and alias for the new database. Additional parameters can be set on the Database tab.

SQLite Expert will choose a default alias for the database, but you can change it according to your preferences. You can enter any name for the alias, as long as it is not already used. After entering the desired parameters, the database will be created in the specified directory and it will show in the tree panel.

Tips

After immediately creating a SQLite database and before any tables are created, a SQLite database is a single file having a size of 0 bytes. At this point, no information has been written on disk. If you close and re-open SQLite Expert, the program will ask which is the desired encoding for the database. After creating the first table, the encoding setting is persisted into the database and cannot be changed.

Opening an existing database

To open a SQLite3 database, select

File » Open Database

or

File » Open Recent Database

from the main menu. You will be prompted to select the file name for new database. After selecting a file name, the database will show in the left tree panel. SQLite Expert will choose a default alias for the database, but you can change it according to your preferences.

SQLite Expert will automatically connect to the database so you will see all the tables and views in the left tree panel.

You can also drop SQLite database files (with .db3 or another registered extension) from the Windows environment in the left tree panel.

Closing a database

To close and unregister a SQLite3 database, select

File » Close Database

This will close the connection to the database and will remove it from the left tree panel.

Renaming database alias

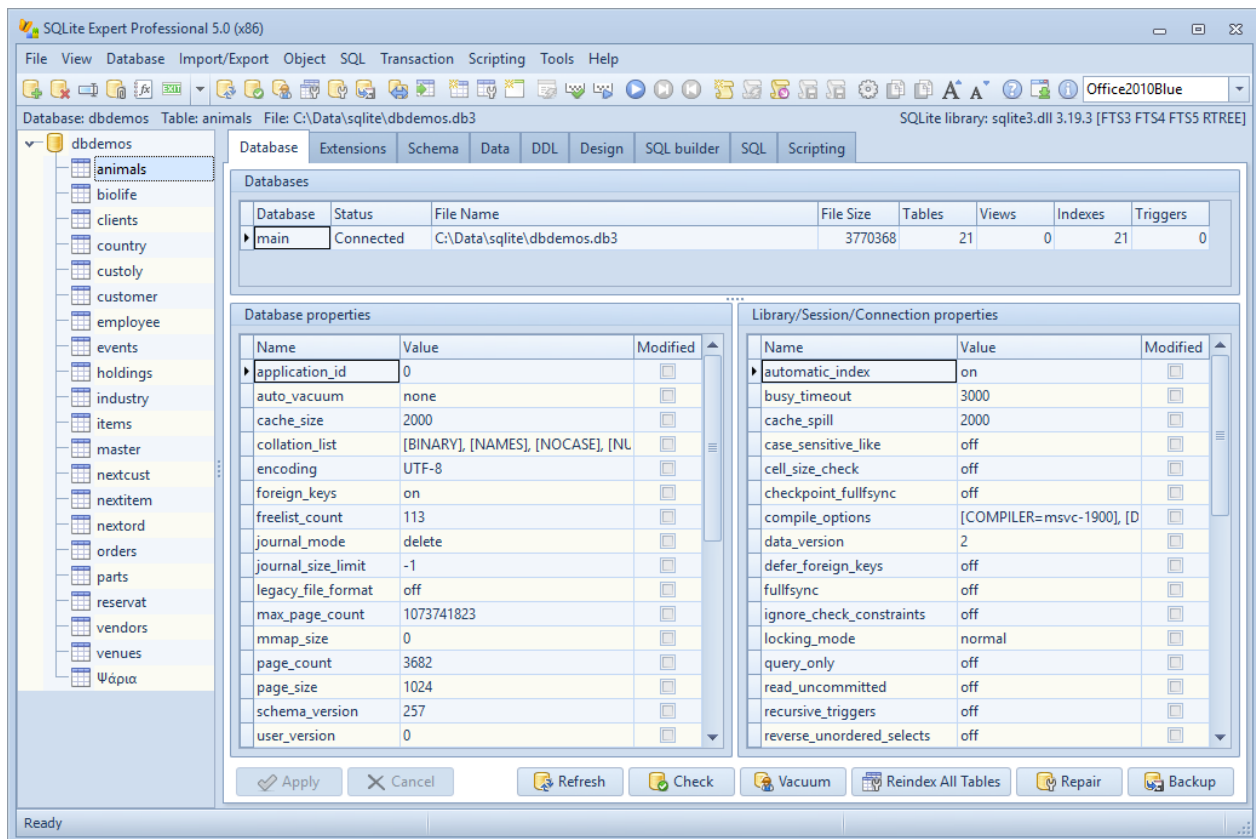
To change the alias for a SQLite3 database, select it in the tree panel and then select

File » Rename Database Alias

You will be prompted with a dialog that will allow you to change the database alias.

Displaying database information and internal parameters

In order to view or change the properties of the selected database, click on the Database tab.



The Databases grid displays information about each of the internal databases ([main], [temp] and

the attached databases if any).

The following information is available:

- ⟨ File name for each database except [temp]
- ⟨ File size in bytes for each database except [temp] or :memory: databases
- ⟨ Number of tables in the database
- ⟨ Number of views in the database
- ⟨ Number of indexes in the database
- ⟨ Number of triggers in the database
- ⟨ Status for the [main] database (connected, disconnected, encrypted, unavailable)

The grid in the lower left part of the screen displays the properties of the selected database in the Databases grid, in a master-detail relationship. These properties are persistent in the database.

The grid in the lower right part of the screen displays the library/session/connection properties of the selected database in the Databases grid, in a master-detail relationship. These properties are not persistent in the database.

These properties can be queried or set using [PRAGMA](#) commands. To modify the value of a property, press ENTER or double-click on the record, enter the new value and then click the **Apply** button.

Additionally, the following commands are available:

- ⟨ Integrity check: performs a "pragma integrity_check" in the selected database.
- ⟨ Vacuum: performs a "vacuum" in the [main] database.
- ⟨ Reindex: reindexes all tables in the selected database.
- ⟨ Repair: repair a corrupted database.
- ⟨ Backup the selected database.

Note: not all properties can be set or queried. For example, the encoding can only be set immediately after the creation of a database, and before any tables have been created. For more information see the [Pragma commands](#) in the SQLite documentation.

Performing maintenance operations

Checking database integrity

To perform an integrity check for a SQLite3 database, select it in the tree panel and then select

Database » Check

This will execute an internal **pragma integrity_check** command. According to the SQLite documentation, this command does an integrity check of the entire database. It looks for out-of-order records, missing pages, malformed records, and corrupt indices. 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.

Performing a database Vacuum

To perform a database vacuum, select

Database » Vacuum

This will execute an internal **vacuum** command. According to the SQLite documentation, this command cleans the main database by copying its contents to a temporary database file and reloading the original database file from the copy. This eliminates free pages, aligns table data to be contiguous, and otherwise cleans up the database file structure.

For a detailed description of the vacuum command, please see the SQLite3 online documentation.

Reindexing all the tables in the database

To reindex all the tables in the database, select

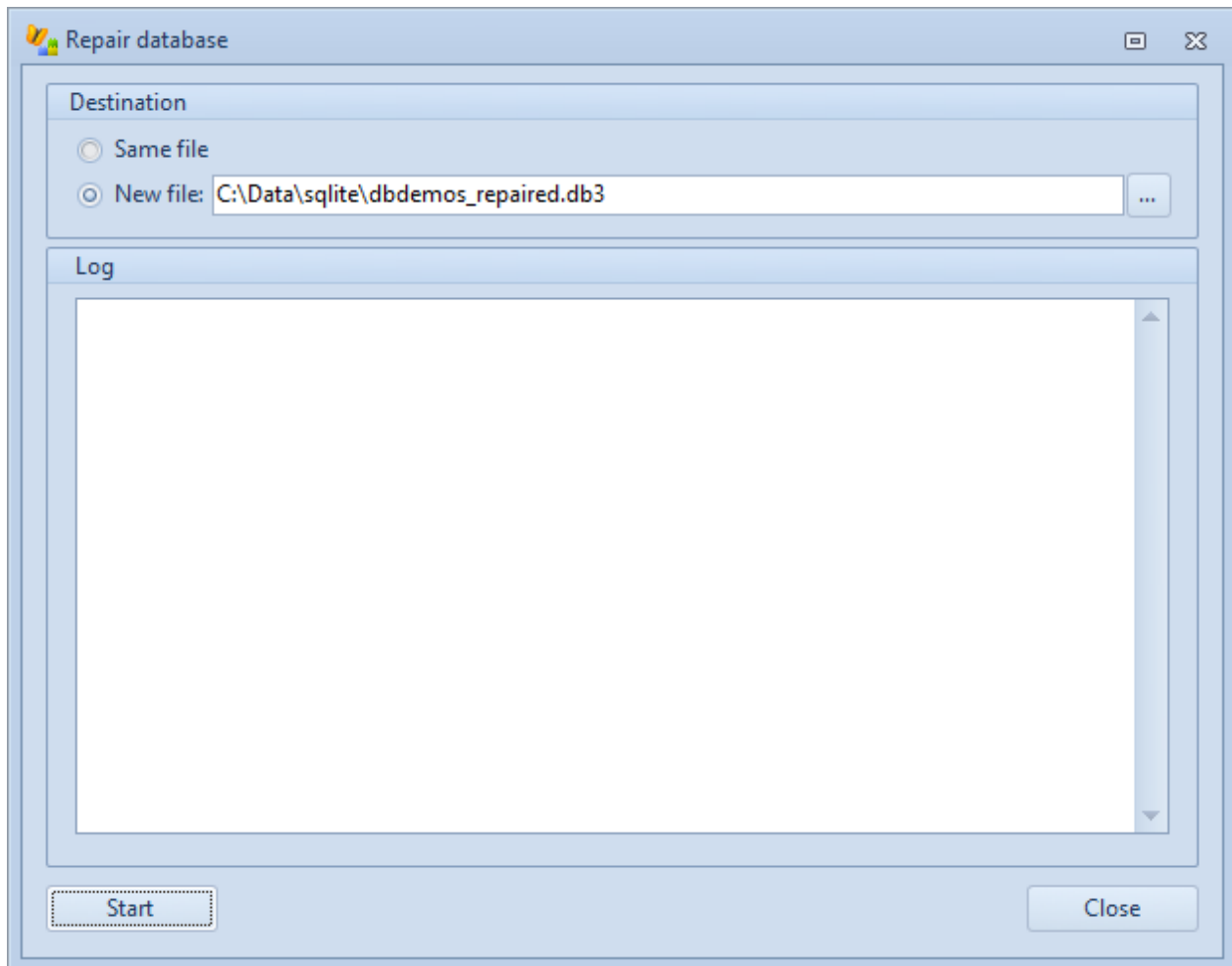
Database » Reindex All Tables

Repairing a database

To repair a damaged database, select

Database » Repair

This will pop up the database repair dialog:



It is recommended to choose the option to repair the database in a new file in order to preserve the old database. Click the Start button to start the repair process. SQLite Expert will create a new database and will try to copy as much data as possible in the new database. There is no guarantee that all the data will be recovered.

Using encrypted databases

SQLite Expert Professional supports encrypted databases if an SQLite library with encryption support is provided. For legal reasons, an SQLite library with encryption support is currently not included in the distribution.

You can select which SQLite library DLL to use in Tools/Options/SQLite Library. This section displays a list of the SQLite library DLLs found in the installation directory. If you wish to use a SQLite library other than the default one included with the distribution, just copy it in the installation directory and then select it in Tools/Options/SQLite Library. For encryption support, the following libraries are available as freeware or commercial products:

[SQLCipher](#) - open source

[System.Data.SQLite](#) - public domain, includes a SQLite library with encryption support - SQLite.Interop.dll. This library is compatible with the sqlite3crypt.dll included with older versions of SQLite Expert.

[The SQLite Encryption Extension \(SEE\)](#) - commercial

[The Free SQLite Encryption Extension \(FSEE\)](#) - open source

The following operations are available from the database tree popup menu when using a library that supports encryption:

- ⟨ Set an encryption key to a database that was previously unencrypted. This will encrypt the database using the provided encryption key.
- ⟨ Enter the encryption key for accessing an encrypted database. This option allows access to the database, but does not modify its content.
- ⟨ Enter a new key for an encrypted database. This option will re-encrypt the database using the new encryption key. For removing the encryption on a database, just enter an empty encryption key.
- ⟨ Enter default encryption key. This is a key that will be tried first for all the encrypted databases.

If you wish SQLite Expert to remember the encryption keys between sessions (including the default encryption key), check the option Remember encryption keys in Tools/Options/General/Startup. In this case SQLite Expert will store the (encrypted) encryption keys with the settings. This option is less secure and is disabled by default.

Attaching and Detaching databases

Attaching and detaching databases can be performed using the **ATTACH** and **DETACH** SQL commands, or using the user interface commands.

To attach another database to an existing database connection, select

File » Attach Database

You will be prompted to select the file name and alias for the new database. If the attach operation is successful, the tables and views in the attached databases will be displayed in the tree being prefixed with the database alias.

You can also drop SQLite database files (with .db3 or another registered extension) from the Windows environment in the left tree panel while pressing the SHIFT key. The dropped databases will be attached to the current database.

To detach an attached database, select

File » Detach Database

then select the desired database from the list of the existing attached databases.

SQLite Expert remembers attached databases between sessions. At program startup, it will attempt to attach the databases that were attached at the end of the previous session.

Loading and Unloading SQLite extensions

Loading and Unloading SQLite extensions

To load an [SQLite extension](#) in the selected database, select

Load Extension

from the left panel drop-down menu.

You will be prompted to select the file name and the entry point for the new database, and also if the extension should be registered as an Auto-extension. If the operation is successful, the functions defined in the extension will be available for usage in SQL expressions in the selected database.

If the Auto option is selected, the extension will be registered using the [sqlite3_auto_extension\(\) API](#) and will be loaded for each new database connection, but not for the existing connections.

Loading an SQLite extension can also be performed using a **SELECT LOAD_EXTENSION** command. However, when loading an extension using the **Load Extension** menu item, this operation is performed by calling the [sqlite3_load_extension\(\) API](#).

Note: when loading a SQLite extension, make sure it is compiled for the same platform as SQLite Expert, i.e. SQLite Expert x64 will only load 64bit SQLite extensions.

To unload an extension, select

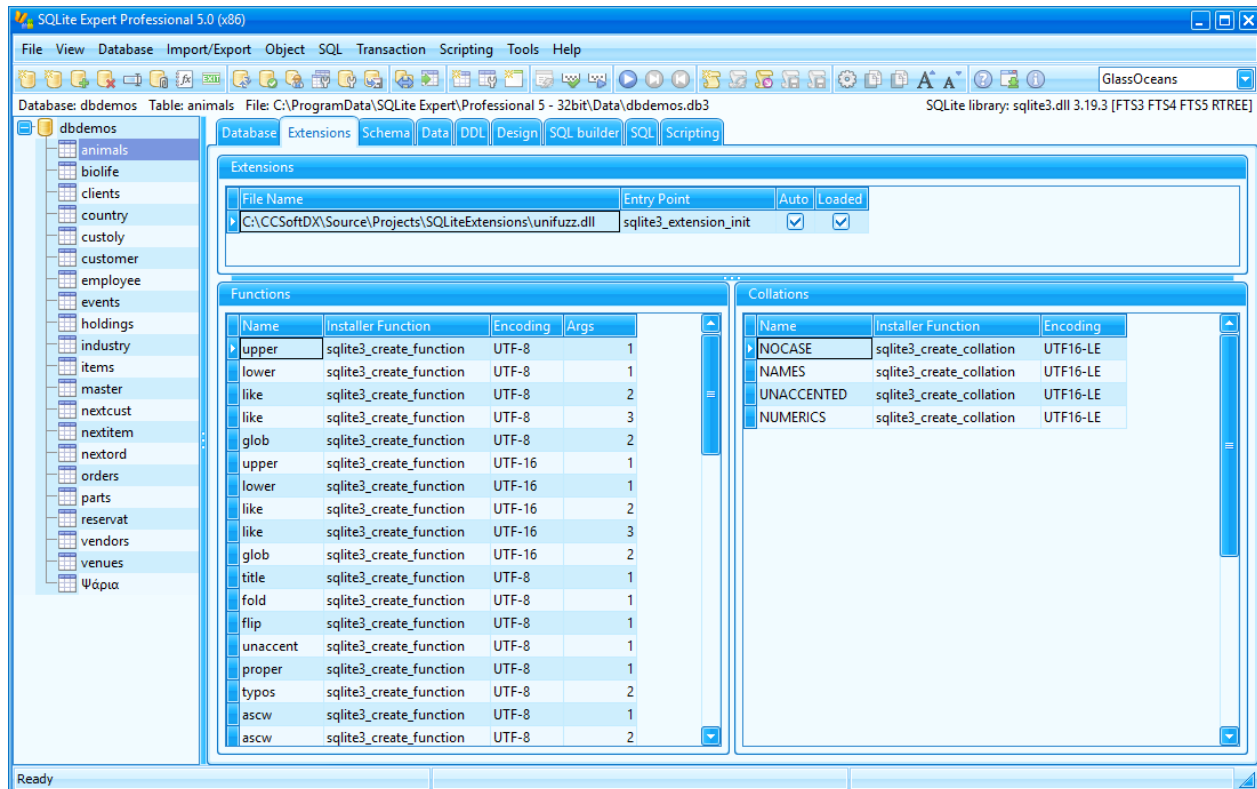
Unload Extension

from the left panel drop-down menu, then select the desired extension from the list of the existing loaded extensions for the selected database. Unlike extension loading, there is no SQL command for unloading an extension. Therefore, in order to unload an extension, SQLite Expert is closing and reopening the selected database.

SQLite Expert remembers loaded extensions between sessions. At program startup, it will attempt to load the extensions that were loaded in the previous session.

Viewing the loaded SQLite extensions

At any moment, a list of the SQLite extensions currently loaded is displayed on the Extensions tab. For each extension, a list of the included functions and collations is also displayed.



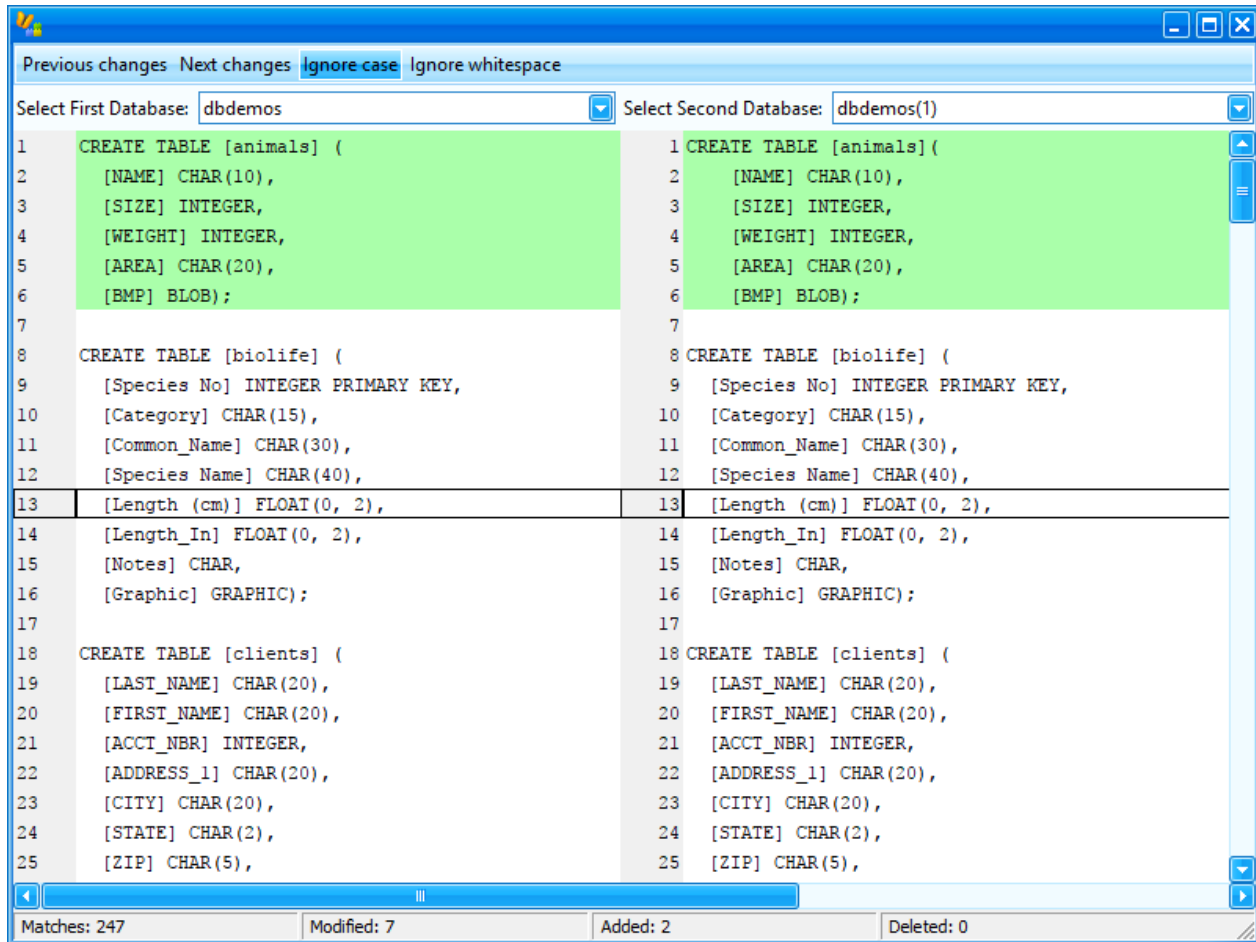
Comparing the DDL or Schema of two databases

To compare the DDL of two databases, follow the steps below:

1. Select **Tools » Database DDL Compare**

from the main menu.

2. Select the first and second database.

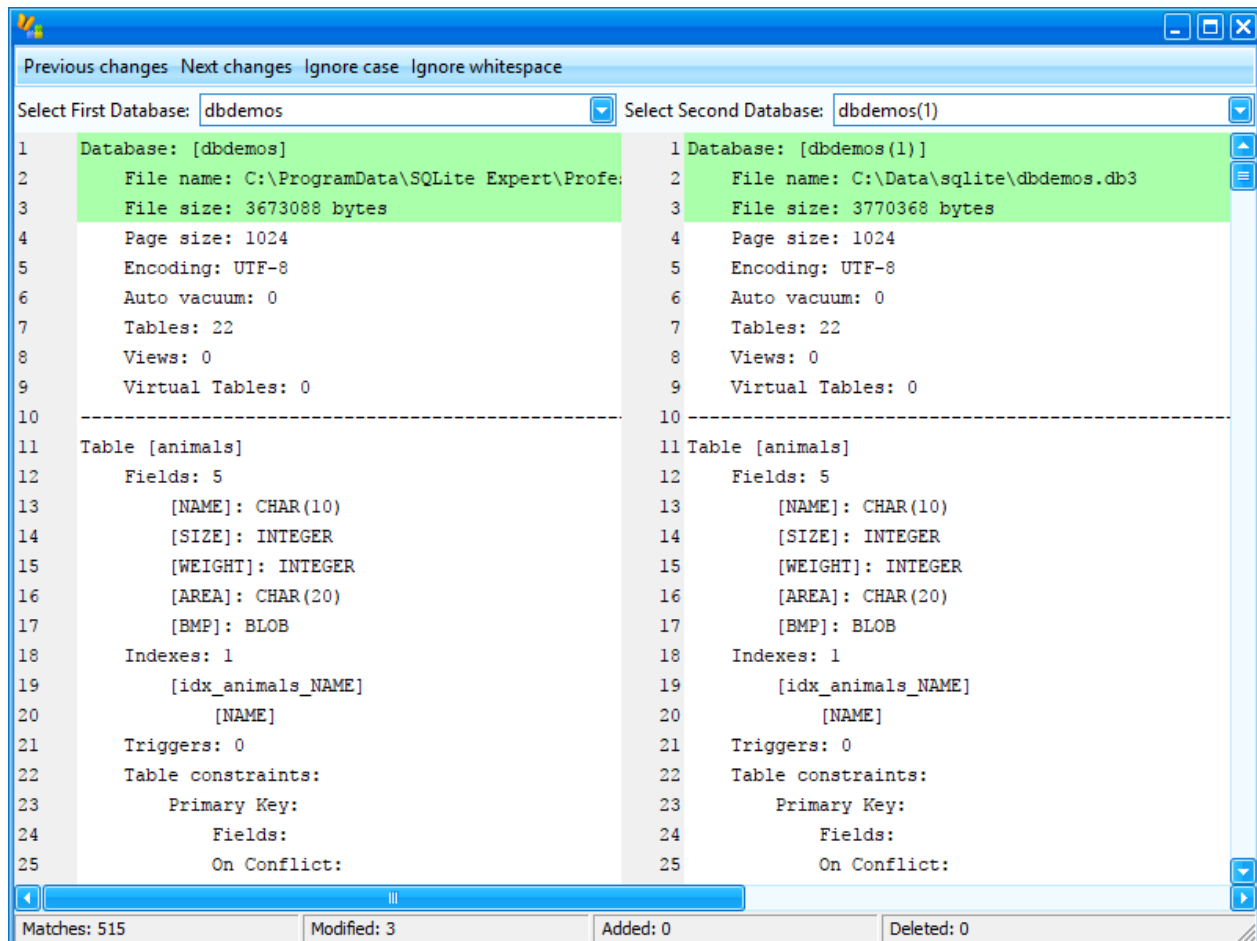


To compare the schema of two databases, follow the steps below:

1. Select **Tools » Database Schema Compare**

from the main menu.

2. Select the first and second database.

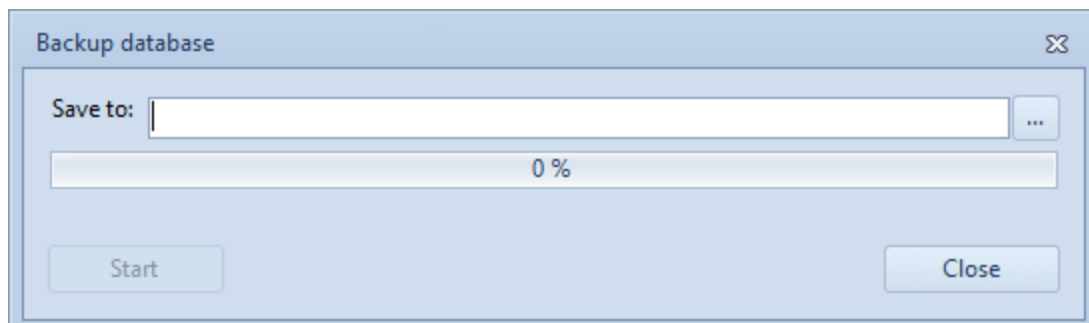


Performing an online backup of a database

To perform an online backup of a database, follow the steps below:

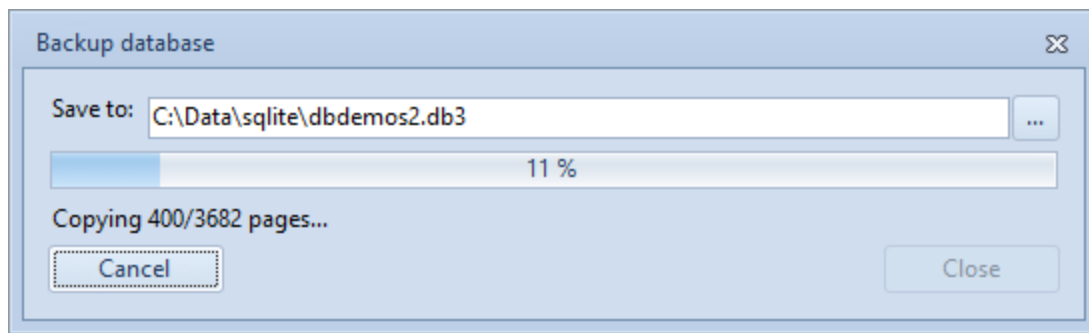
1. Select **Database » Backup**

from the main menu.



2. Enter the filename for the backup database.

3. Click the **Start** button to start the backup process.



The backup process can be canceled by clicking the **Cancel** button.

If the source database is encrypted, the backup database will be encrypted using the same encryption key.

Note: the backup is performed using the [SQLite Online Backup API](#). According to SQLite documentation:

"The online backup API allows the contents of one database to be copied into another database, overwriting the original contents of the target database. The copy operation may be done incrementally, in which case the source database does not need to be locked for the duration of the copy, only for the brief periods of time when it is actually being read from. This allows other database users to continue uninterrupted while a backup of an online database is made."

Editing tables

- » [Creating a table](#)
- » [Editing a table](#)
- » [Renaming a table](#)
- » [Deleting a table](#)
- » [Editing fields](#)
- » [Editing indexes](#)
- » [Editing foreign keys](#)
- » [Editing constraints](#)
- » [Editing table triggers](#)

- » [Copying fields, indexes, constraints and triggers using the clipboard](#)
- » [Applying the changes after editing a table](#)
- » [Limitations](#)

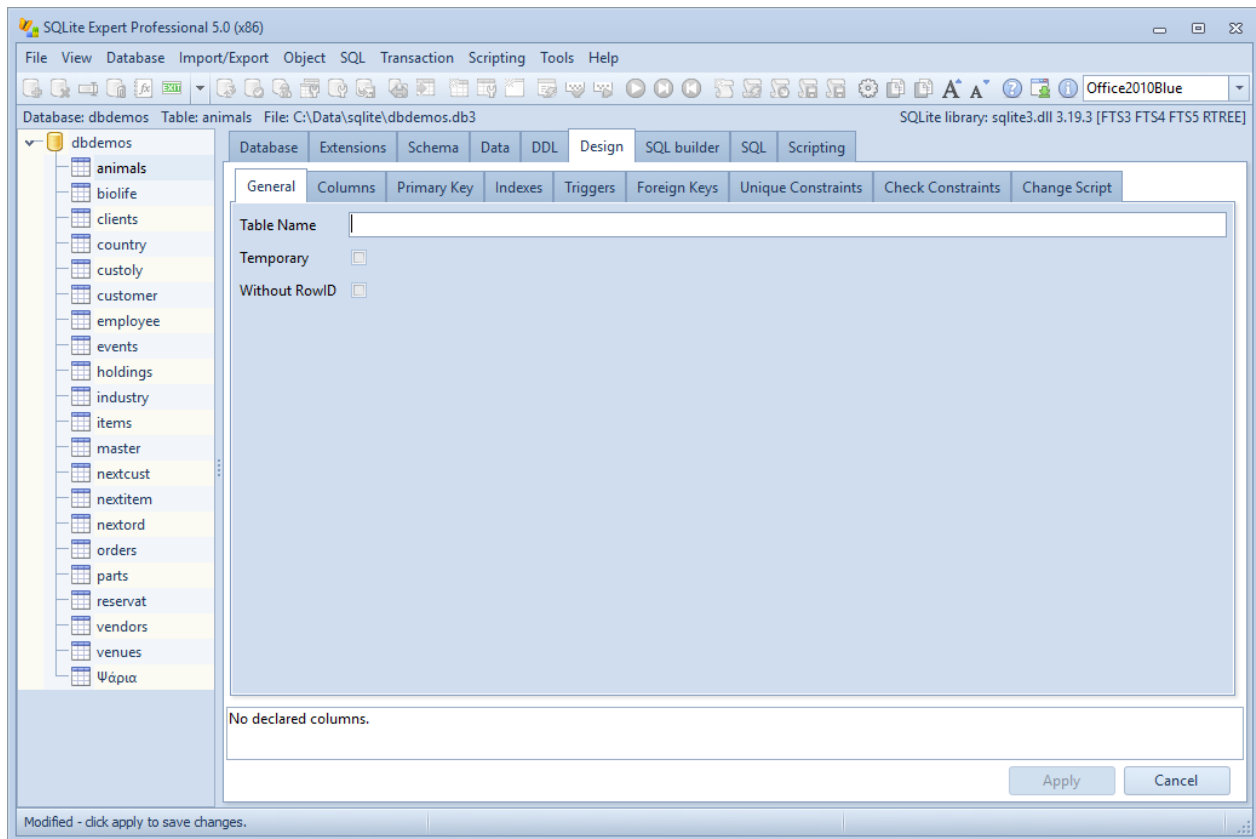
Creating a table

To create a table in the selected database, follow the steps below:

1. Select

Object » New Table

from the main menu, or **New Table** from the left tree panel popup menu. This enters the table editing mode:



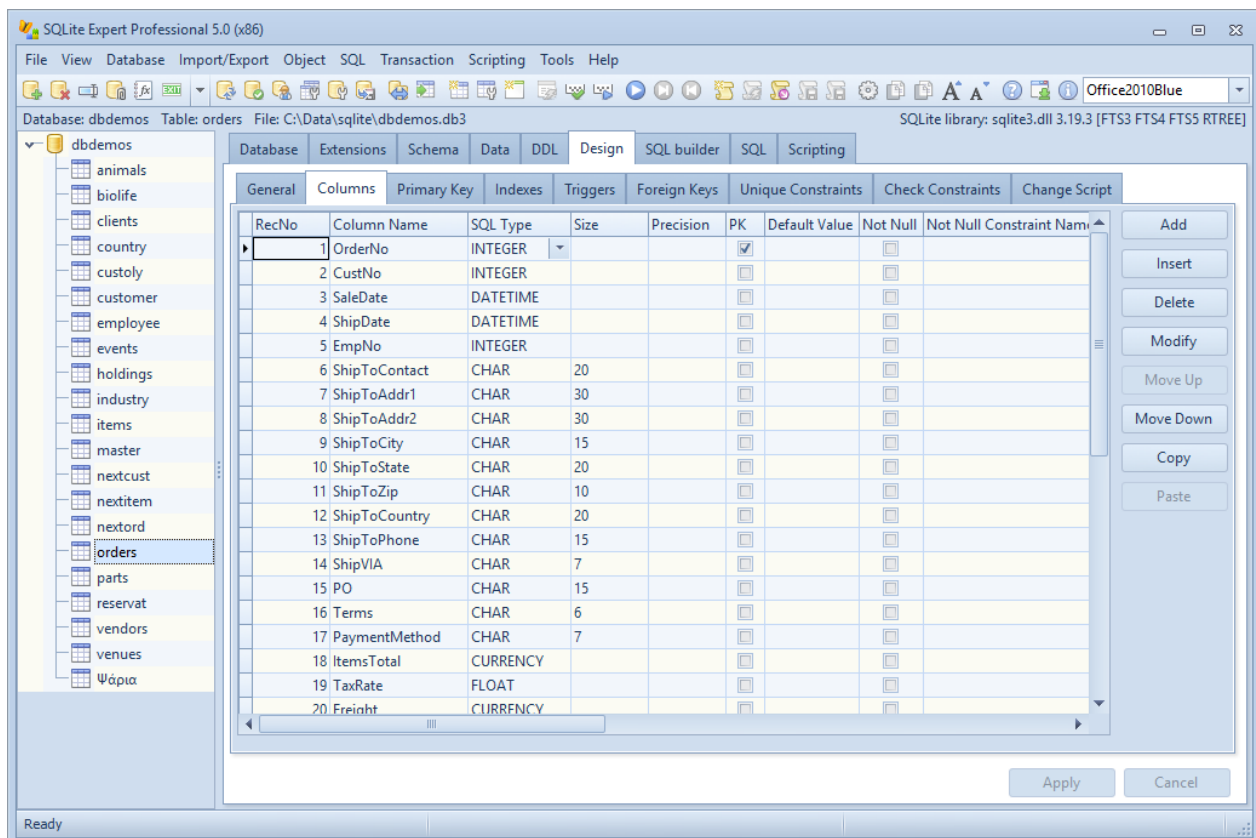
2. If you want to create a temporary table, check the 'Temporary' checkbox.
3. If you want to create a table without rowid, check the 'Without RowID' checkbox.
4. Enter a name for the table.
5. [Add, modify or delete columns.](#)
6. [Add, modify or delete primary key.](#)

7. [Add, modify or delete indexes.](#)
8. [Add, modify or delete foreign keys.](#)
9. [Add, modify or delete constraints.](#)
10. [Add, modify or delete triggers.](#)
11. [Apply the changes.](#)

Editing a table

To edit an existing table, follow the next steps:

1. Select the table in the left tree panel.
2. Go to the Design tab.



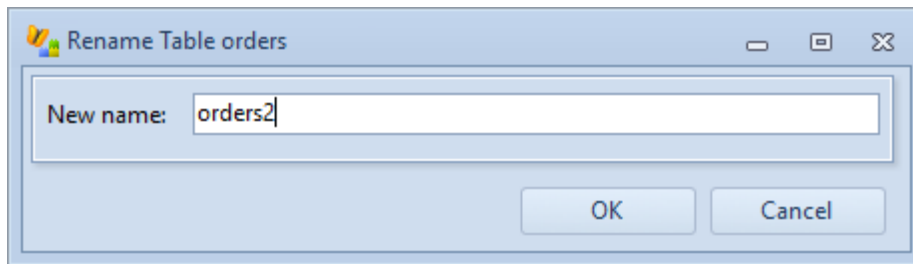
3. [Add, modify or delete columns.](#)
4. [Add, modify or delete primary key.](#)
5. [Add, modify or delete indexes.](#)
6. [Add, modify or delete foreign keys.](#)

7. [Add, modify or delete constraints.](#)
8. [Add, modify or delete triggers.](#)
9. [Apply the changes.](#)

Renaming a table

To rename a table, follow the next steps:

1. Select the table you wish to rename in the left tree panel.
2. Select **Table » Rename Table** from the main menu.



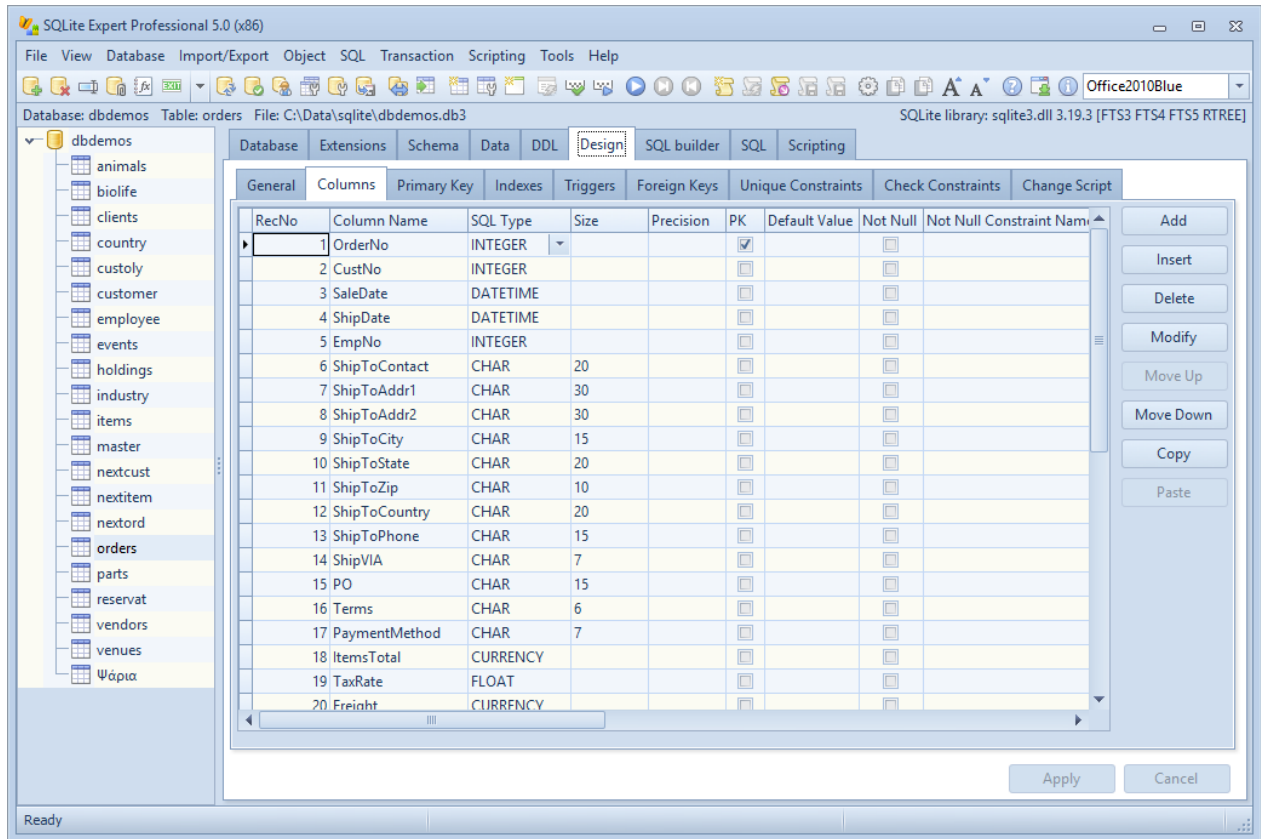
3. Enter the new name for the table in the table rename popup dialog.
4. Click **OK**.

Deleting a table or a group of tables

To delete one or more tables in the selected database, select the tables using the mouse or the keyboard using CTRL or SHIFT operations, then press **DELETE** or select **Delete** from the left tree panel popup menu, then click **OK** to confirm.

Editing columns

To edit the columns of a table, go to the Design tab on the outer tab view, and then go to the Columns tab on the inner tab view:



To add a column:

1. Click the **Add** button. This will pop up the column properties dialog.

Column Editor

Column Name

SQL Type

Size

Precision

Default Constraint Name

Default Value

Not Null Constraint Name

Not Null ☐

Not Null Conflict Clause

Unique Constraint Name

Unique ☐

Unique Conflict Clause

Collate Constraint Name

Collate

OK Cancel

2. Enter the desired properties for the column: name, type, size, precision, default value, not null, unique and collation.

3. Click **OK** to close the dialog.

To delete a column:

1. Select the column in the list.

2. Click the **Delete** button.

3. Click **OK** to confirm.

To modify a column:

1. Select the column in the list.

2. Click the **Modify** button. This will pop up the column properties dialog.

3. Change the desired properties for the column: name, type, size, precision, default value, primary key, collation, not null, unique, collate, check constraints and foreign key references.

4. Click **OK** to close the dialog.

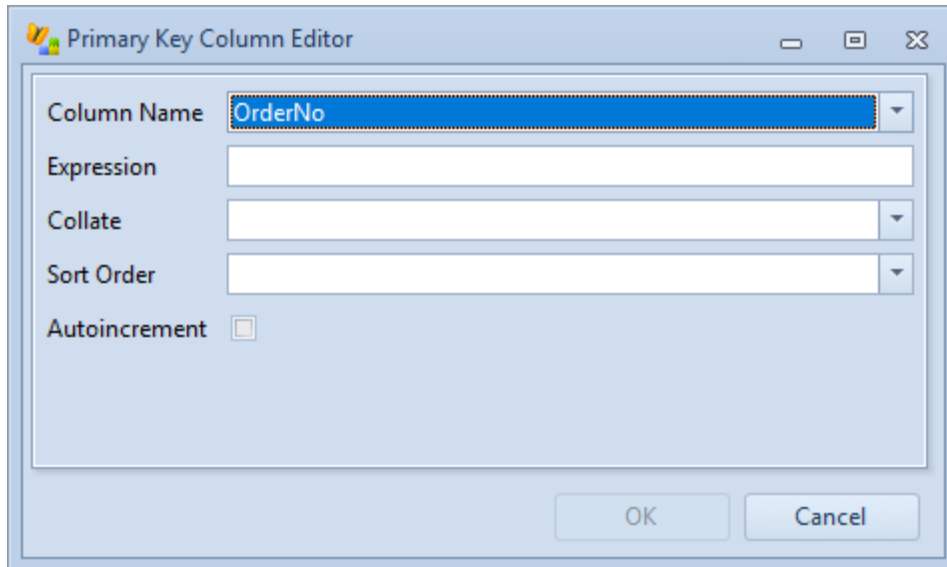
To change the order of the columns in a table:

Use the **Move Up** and **Move Down** buttons.

To create an Autoincrement column:

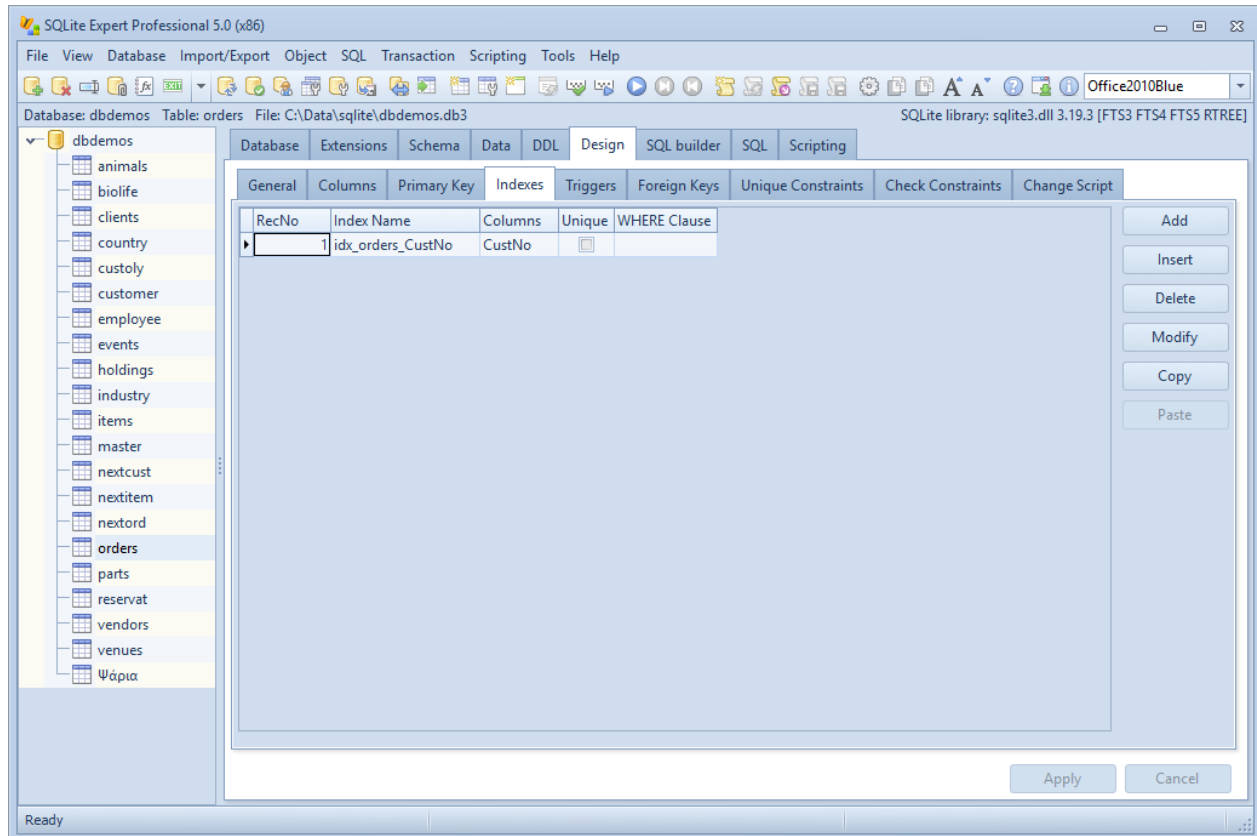
SQLite allows the creation of a single auto-increment column for a table, and only if the field is declared as INTEGER PRIMARY KEY.

To create an auto-increment column, go to the Primary Key tab and declare a primary key on a single INTEGER column, then select the Autoincrement option:



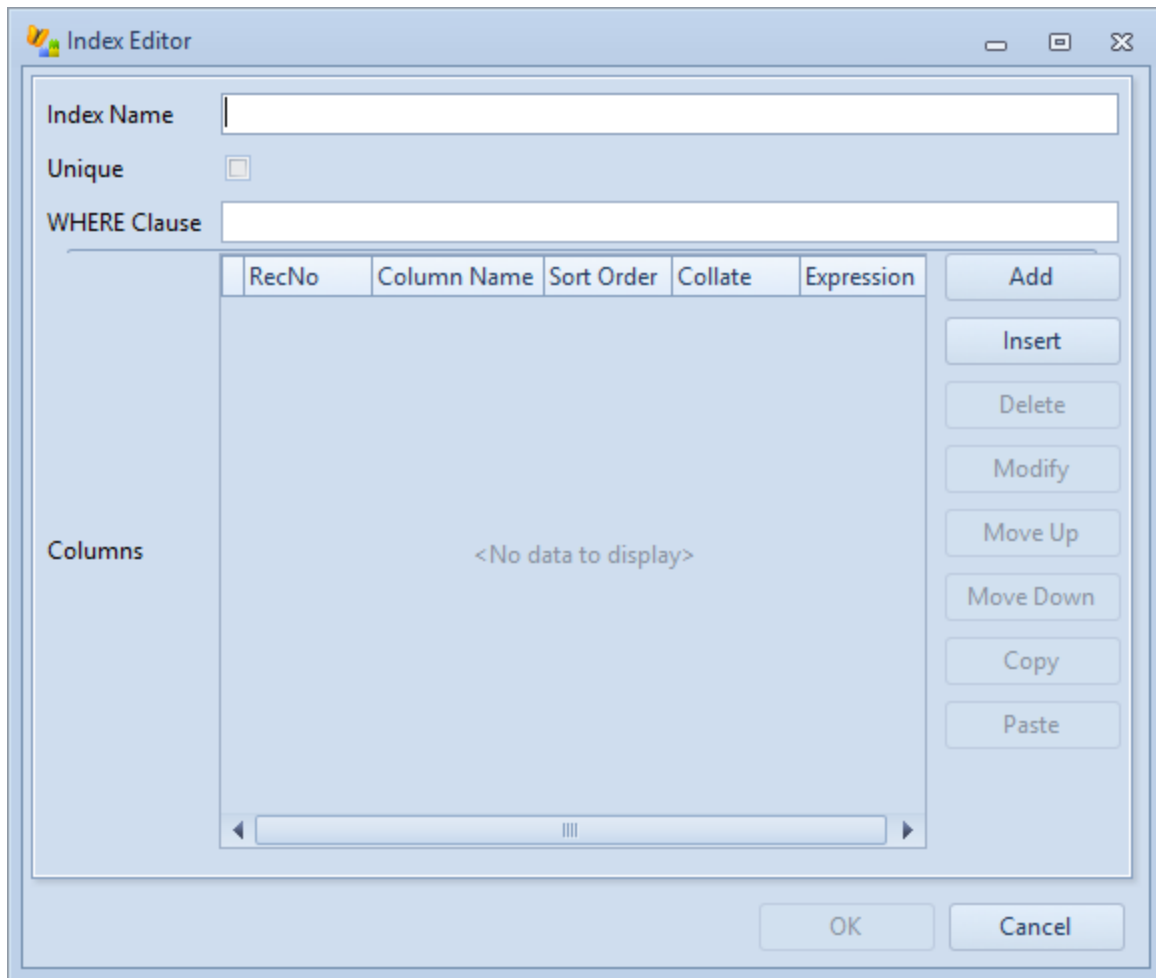
Editing indexes

To edit the indexes of a table, go to the Design tab on the outer tab view, and then go to the Indexes tab on the inner tab view:



To add an index:

1. Click the **Add** button. This will pop up the index properties dialog:



2. Enter the desired properties for the index: name, unique, WHERE clause and columns.
3. To add a column to the index, click the **Add** button.
4. To remove a column from an index, select it in the list and click the **Delete** button.
5. To change the order of the fields in an index, use the **Move Up** and **Move Down** buttons.
6. Click **OK** to close the dialog.

To delete an index:

1. Select the index in the list.
2. Click the **Delete** button.
3. Click **OK** to confirm.

To modify an index:

1. Select the index in the list.
2. Click the **Modify** button. This will pop up the index properties dialog.
3. Change the desired properties for the index: name, unique, WHERE clause and columns.

4. To add a column to the index, select it in the left and click the **Add** button.
5. To remove a column from an index, select it in the list and click the **Delete** button.
6. To change the order of the fields in an index, use the **Move Up** and **Move Down** buttons.
7. Click **OK** to close the dialog.

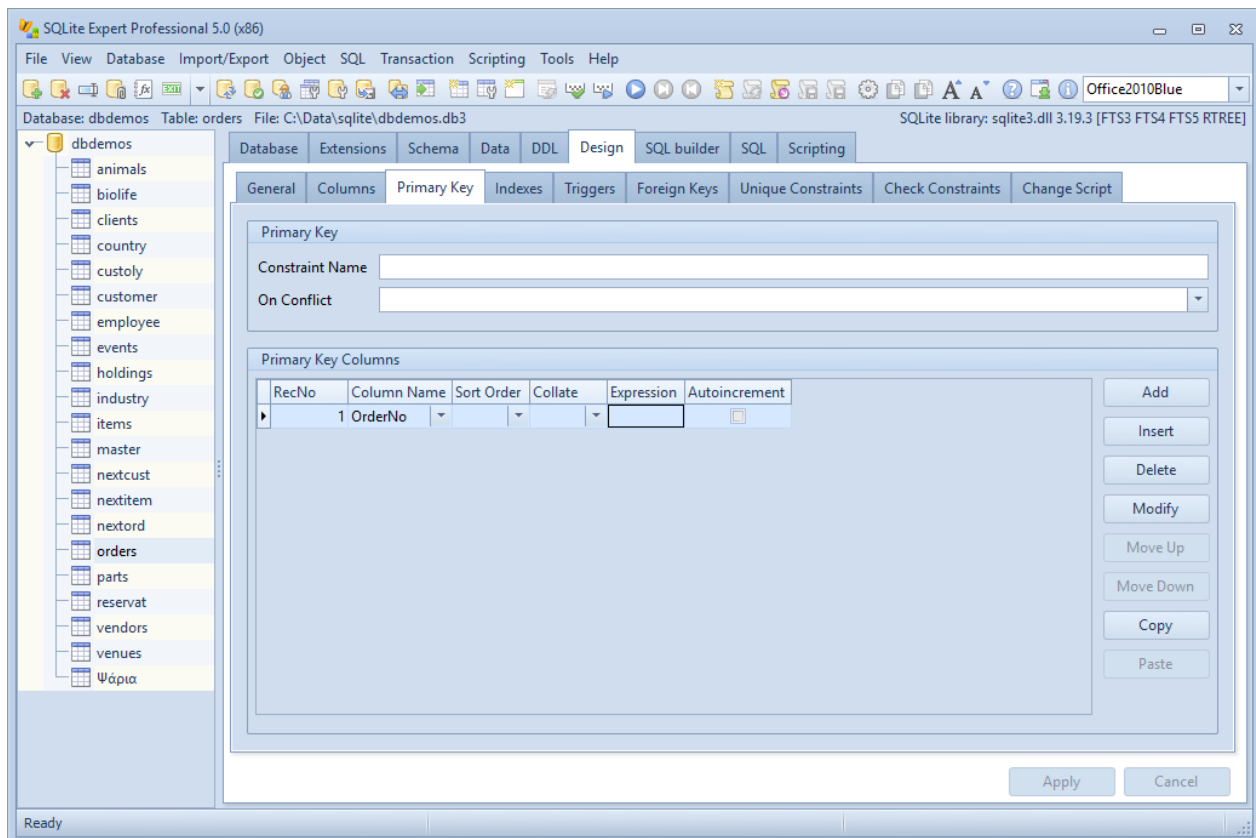
You need to add at least one column to a table before adding any indexes.

Indexes associated with unique constraints cannot be deleted or modified. They are created automatically by the SQLite engine and their name follows this pattern: 'sqlite_autoindex_<tablename>_<n>' where <tablename> is the name of the table, and <n> is a number starting from 2. To delete or modify an index associated with a unique constraint, delete or modify its associated constraint.

Note: You can find more information about some issues with primary keys in SQLite [here](#).

Editing primary keys

To create a primary key constraint, open the table editor and go to the Primary key tab:



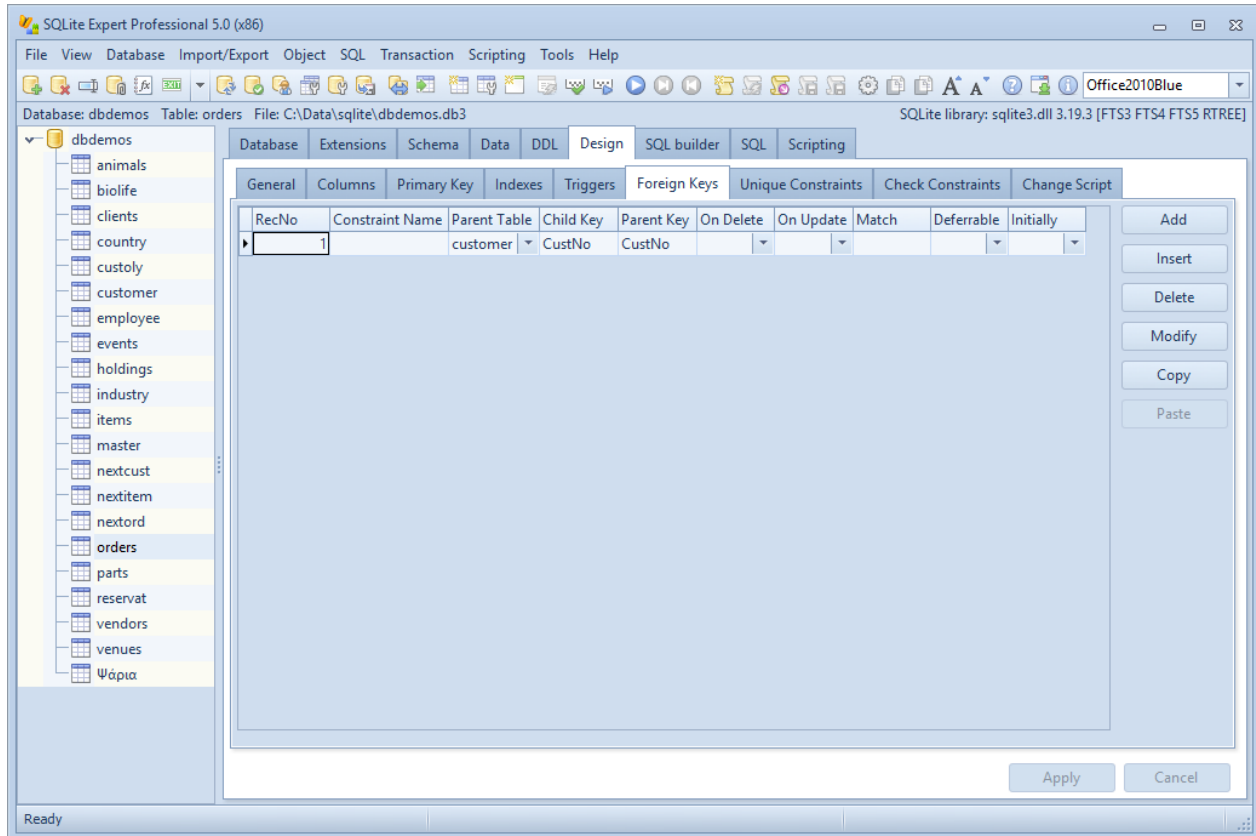
You can modify the following properties for the table primary key: constraint name, conflict clause and columns.

Note: you can only declare a single primary key for a table. If the primary key has a single column then it will be created as a column constraint, otherwise it will be created as a table constraint.

Editing foreign keys

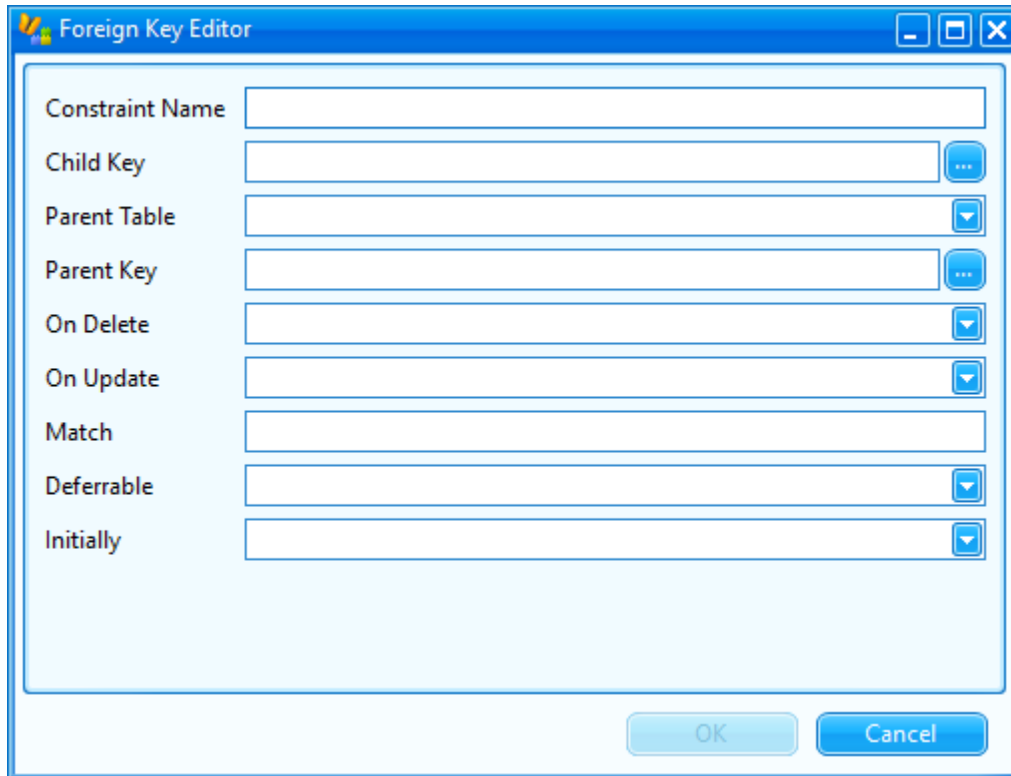
SQLite added full [support for foreign keys](#) in [version 3.6.19](#) of the library. SQLite Expert added support for foreign keys in version 2.2.0.

To create a foreign key, go to the Design tab and then go to the Foreign Keys sub-tab:

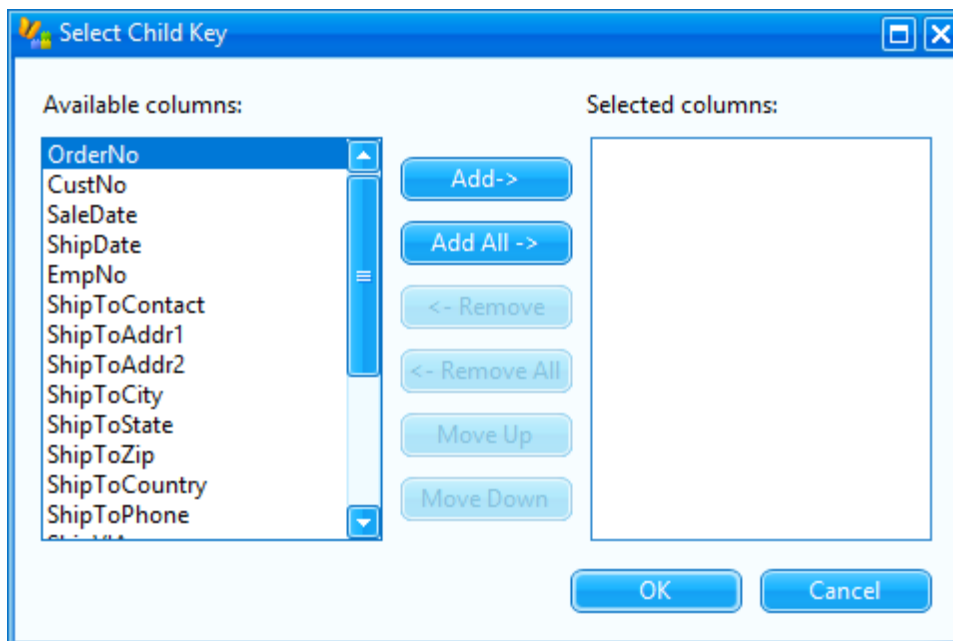


To add a foreign key:

1. Click the **Add** button. This will pop up the table Foreign Key Editor:



2. Enter the desired properties for the foreign key: constraint name, parent table, child key, parent key, and ON DELETE, ON UPDATE, MATCH, DEFERRABLE and INITIALLY clauses.
3. To select the child or parent keys, click on the corresponding '...' button. This pops up the column selection dialog:



4. Select the desired columns, then click **OK** to close the column selection dialog.
5. Click **OK** to close the foreign key dialog.

To delete a foreign key:

1. Select the foreign key in the list.
2. Click the **Delete** button.
3. Click **OK** to confirm.

To modify a foreign key:

1. Select the foreign key in the list.
2. Click the **Modify** button. This will pop up the foreign key properties dialog.
3. Change the desired properties for the foreign key: name, parent table, child key, parent key, and ON DELETE, ON UPDATE, MATCH, DEFERRABLE and INITIALLY clauses.
4. Click **OK** to close the dialog.

Limitations:

It is not possible to restructure a table that is referenced by one or more foreign keys. When restructuring a table, SQLite Expert renames the table, creates a new table with the original name, copies the data from the temporary table, then deletes the temporary table. A foreign key referencing a table prevents it from being restructured.

Notes:

The foreign keys are enabled by default in SQLite Expert when opening a database.

If the foreign key has a single column then it will be created as a column constraint, otherwise it will be created as a table constraint.

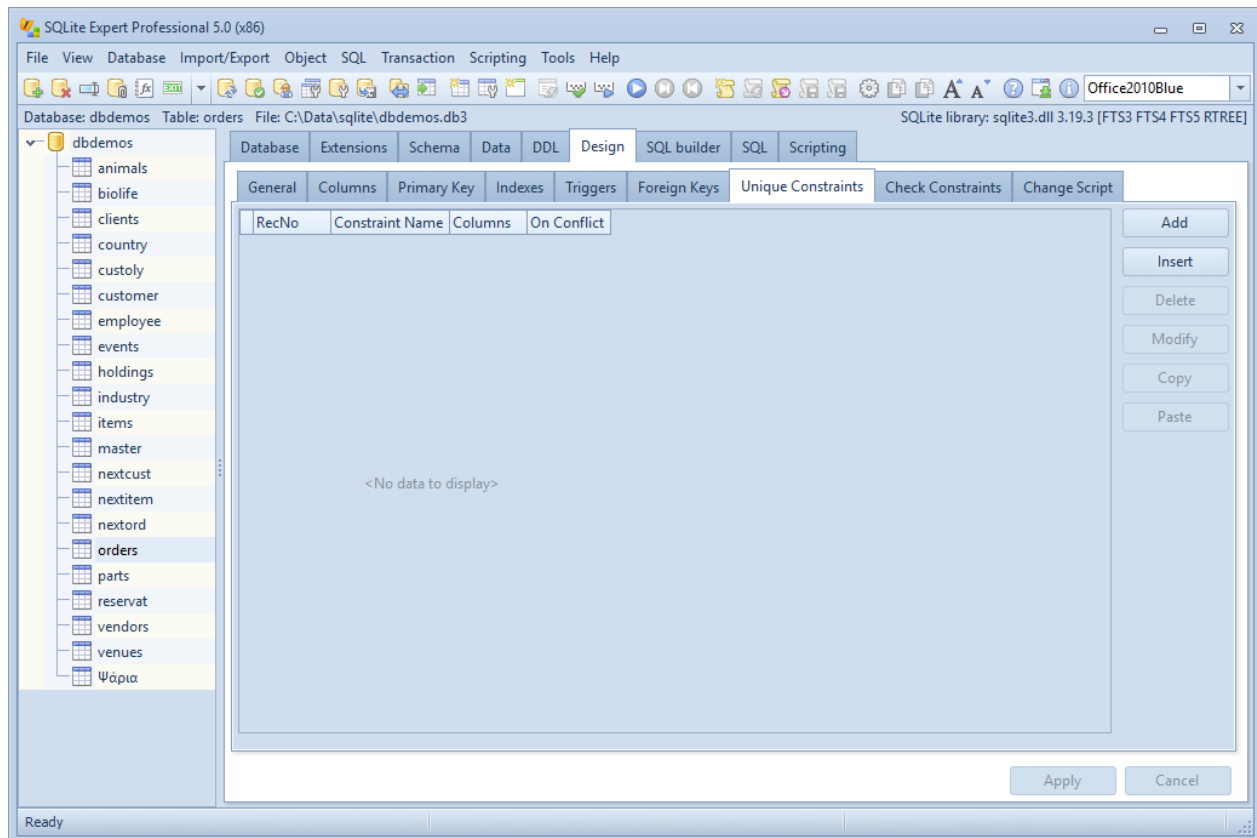
Editing constraints

To edit column constraints, see [Editing columns](#).

Unique Constraints

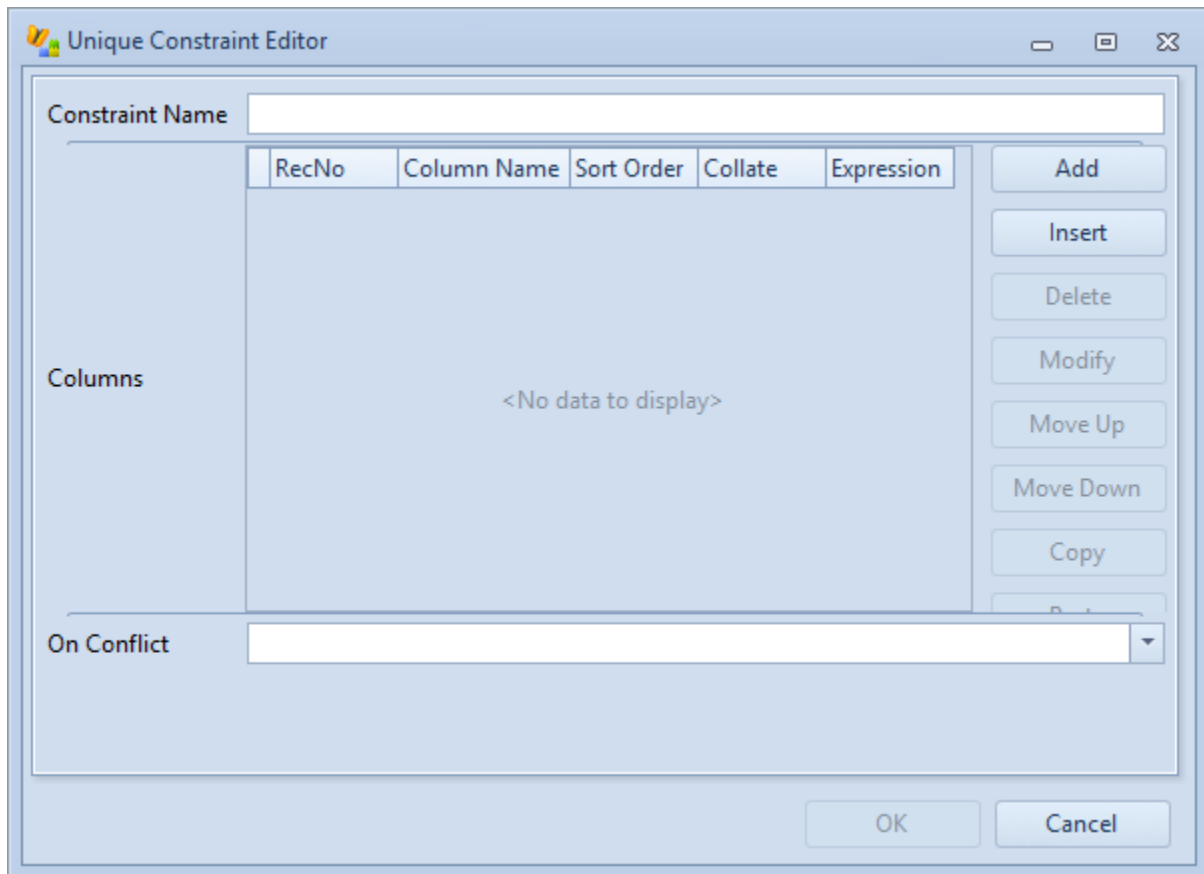
To edit the unique constraints of a table, go to the Design tab on the outer tab view, and then go to

the Unique Constraints tab on the inner tab view:



To add a Unique constraint:

1. Click the **Add** in the Unique Constraints group box. This will pop up the unique constraint properties dialog:



2. Enter the desired properties for the unique constraint: name, columns (each with optional collate and sort order) and conflict clause.
3. Click **OK** to close the dialog.

To delete a Unique constraint:

1. Select the unique constraint in the list.
2. Click the **Delete** button.
3. Click **OK** to confirm.

To modify a Unique constraint:

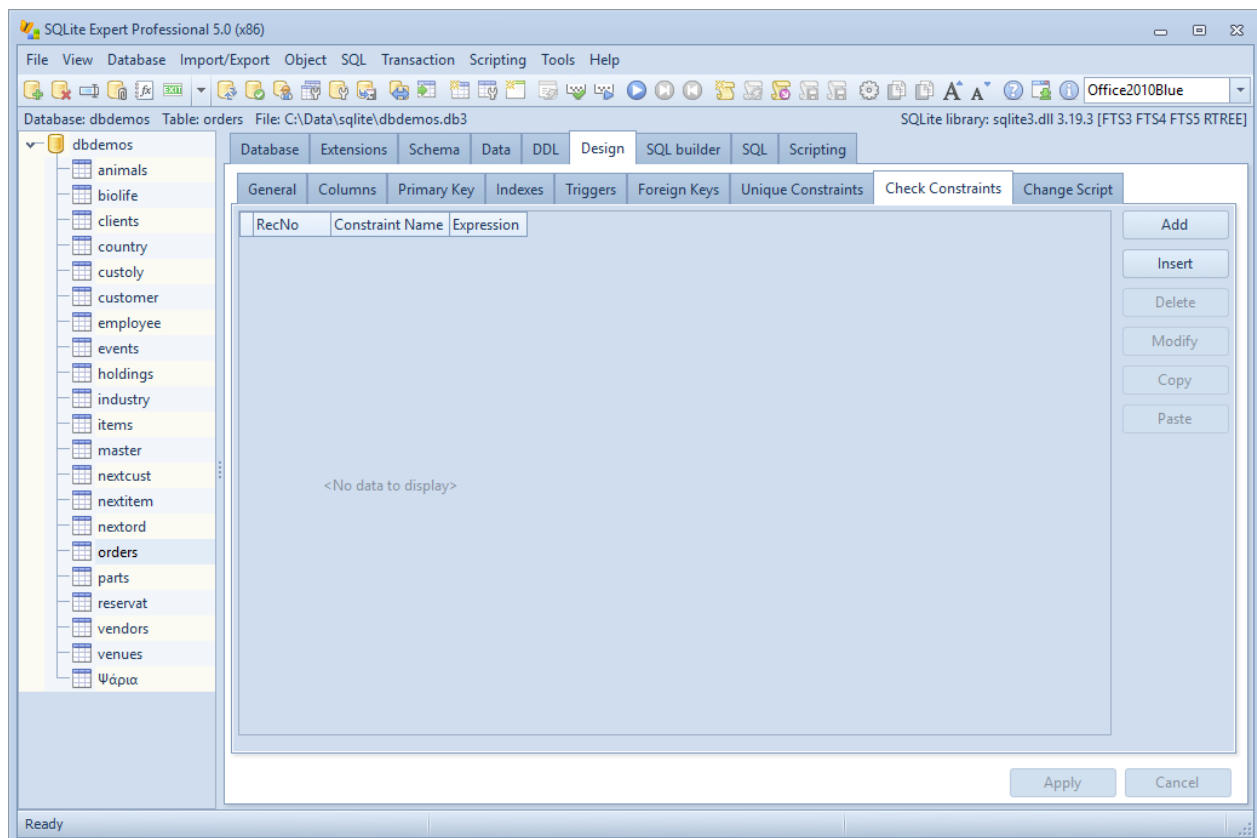
1. Select the constraint in the list.
2. Click the **Modify** button. This will pop up the unique constraint properties dialog.
3. Change the desired properties for the unique constraint: name, columns (each with optional collate and sort order) and conflict clause.

4. Click **OK** to close the dialog.

Note: You need to add at least one column to a table before adding any unique constraints.

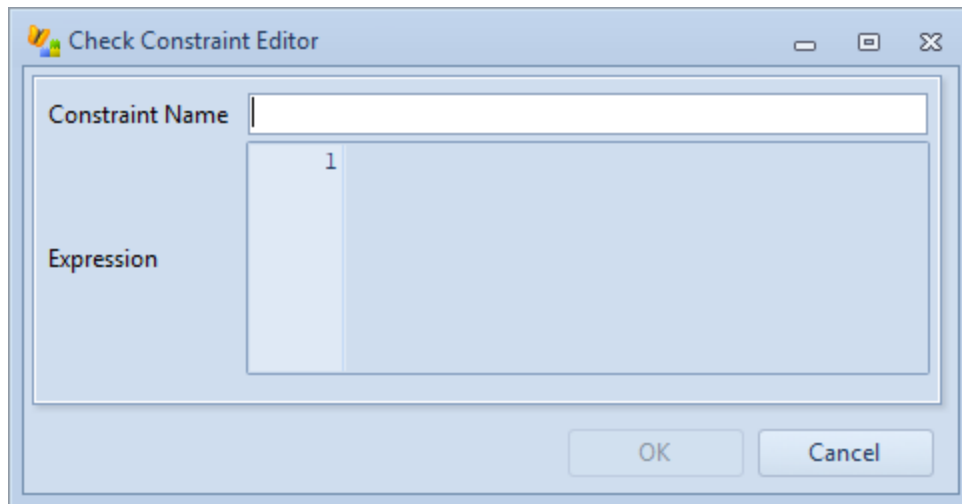
Check Constraints

To edit the check constraints of a table, go to the Design tab on the outer tab view, and then go to the Check Constraints tab on the inner tab view:



To add a Check constraint:

1. Click the **Add** in the Check Constraints group box. This will pop up the check constraint properties dialog:



2. Enter the desired properties for the check constraint: name and expression.
3. Click **OK** to close the dialog.

To delete a Check constraint:

1. Select the constraint in the list.
2. Click the **Delete** button.
3. Click **OK** to confirm.

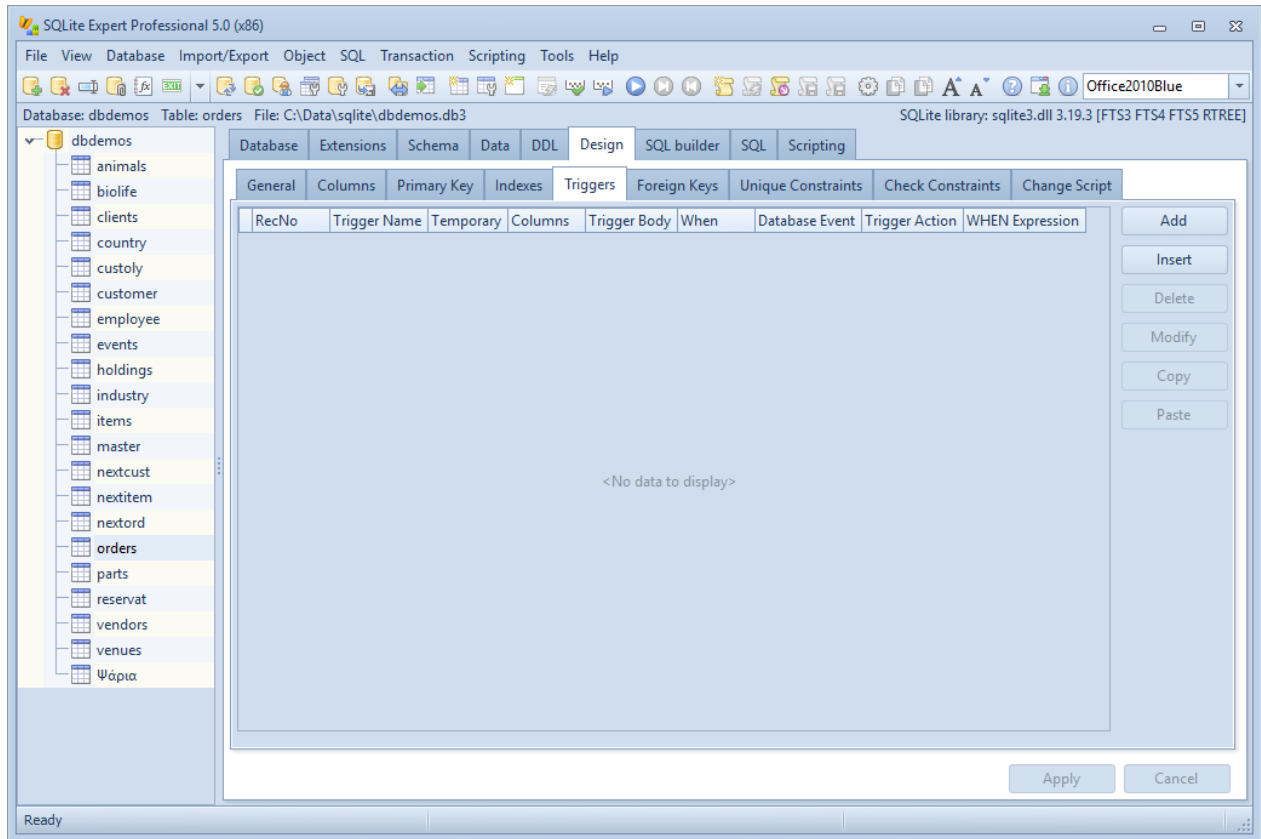
To modify a Check constraint:

1. Select the constraint in the list.
2. Click the **Modify** button. This will pop up the check constraint properties dialog.
3. Change the desired properties for the constraint: name and expression.
4. Click **OK** to close the dialog.

Note: You need to add at least one column to a table before adding any check constraints.

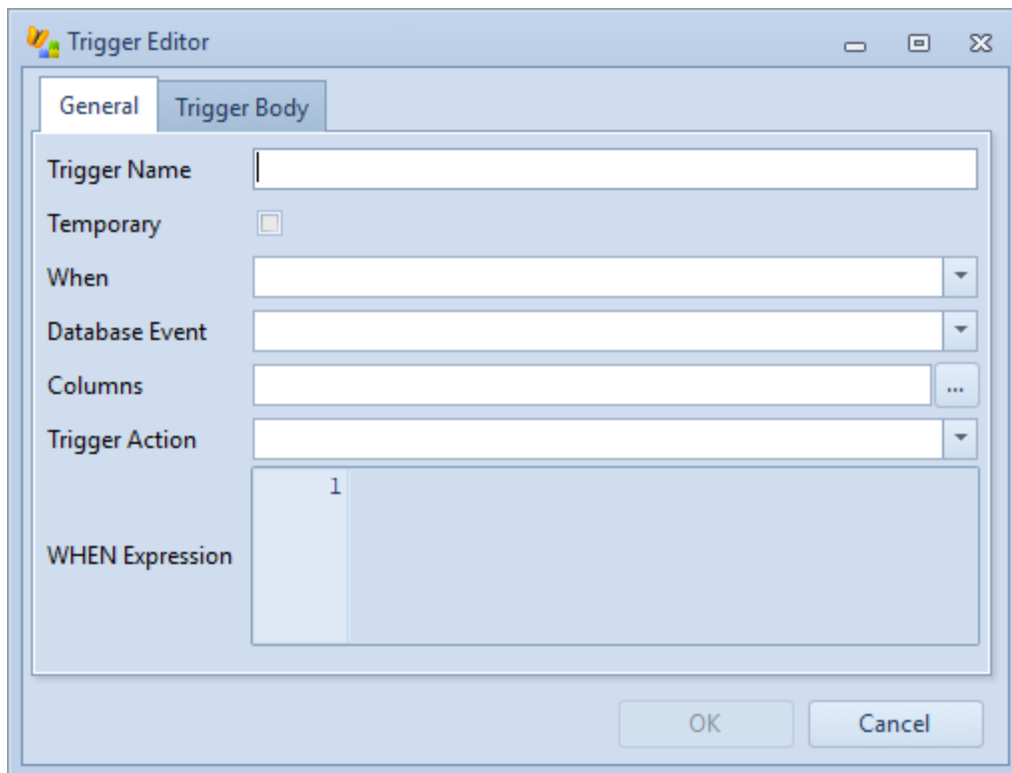
Editing table triggers

To edit the triggers on a table, go to the Design tab on the outer tab view, and then go to the Triggers tab on the inner tab view:



To add a trigger:

1. Click the **Add** button. This will pop up the trigger dialog:



2. Use the trigger dialog to visually design the CREATE TRIGGER script.

3. Click **OK** to close the dialog.

To delete a trigger:

1. Select the trigger in the list.
2. Click the **Delete** button.
3. Click **OK** to confirm.

To modify a trigger:

1. Select the trigger in the list.
2. Click the **Modify** button. This will pop up the trigger dialog:
3. Use the trigger dialog to visually design the CREATE TRIGGER script.
4. Click **OK** to close the dialog.

Copying fields, indexes, constraints and triggers using the clipboard

While editing tables, SQLite Expert allows copying columns, indexes, constraints and triggers between tables using the clipboard.

To copy a column or a group of columns to clipboard, go to the Columns tab, select the desired fields and then press the **Copy** button.

To paste one or more fields from the clipboard, go to the Columns tab and then press the **Paste** button.

Indexes, constraints and triggers can be copied using the clipboard using a similar procedure.

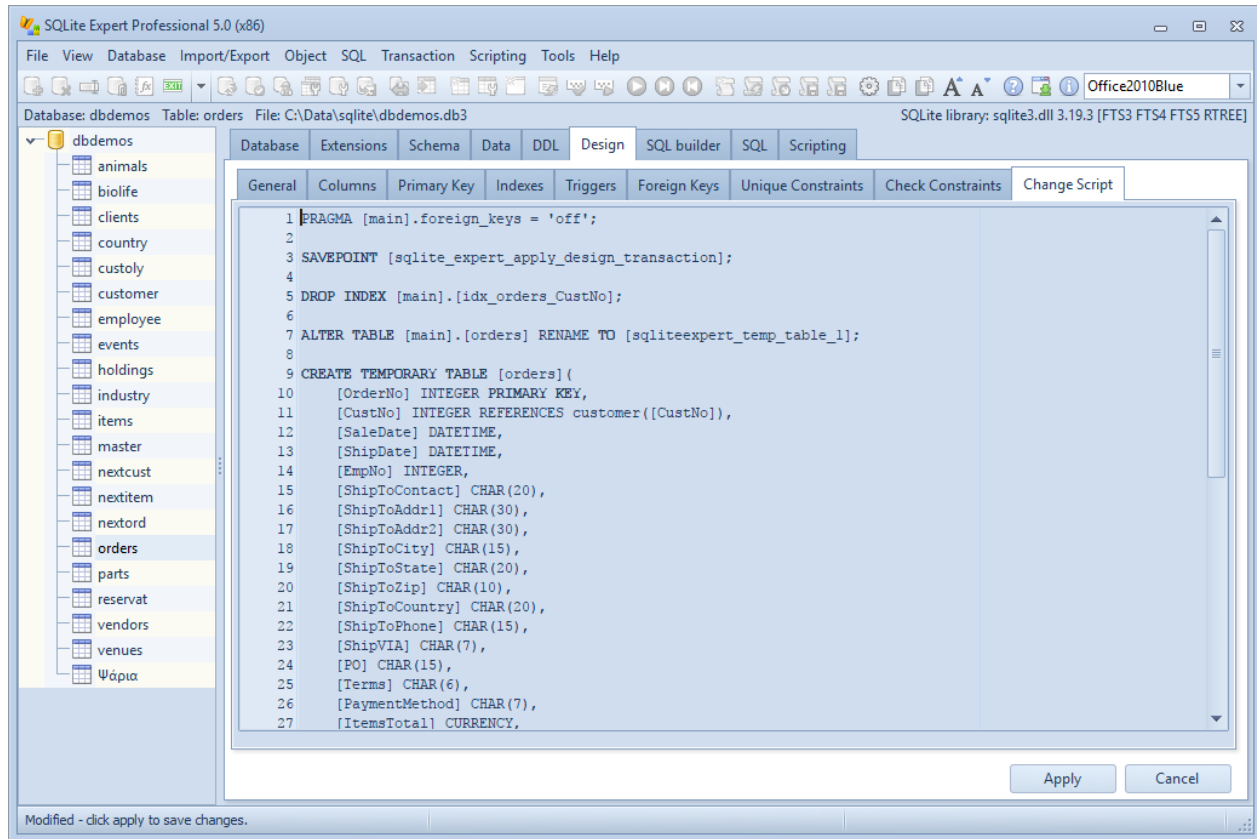
Applying the changes after editing a table

After editing a table, you can apply all the changes by clicking on the **Apply** button, or you can cancel the changes by clicking on the **Cancel** button.

If you choose to apply the changes, SQLite Expert will create a new table with the new structure, copy all the existing data from the old table, and then delete the old table. If any columns have been deleted from the table, the data in these columns will be lost.

Note: SQLite Expert will automatically start a nested transaction before attempting to apply changes. If any errors occur, the transaction will be rolled back and you will be given the chance to correct the error before another attempt to apply the changes.

Before applying the changes, you can see the generated script on the **Change Script** tab.



Limitations

If the current table is referenced in foreign keys, the following restructure operations are disabled:

- Delete or rename fields referenced in foreign keys

If the current table is referenced in foreign keys and a transaction is in progress, the following restructure operations are disabled:

- Add, delete or modify fields
- Add, delete or modify constraints
- Add, delete or modify foreign keys
- Add, delete or modify primary keys
- Paste primary keys from clipboard

Editing virtual tables

- » [Creating a virtual table](#)
- » [Editing a virtual table](#)
- » [Renaming a virtual table](#)
- » [Deleting a virtual table](#)

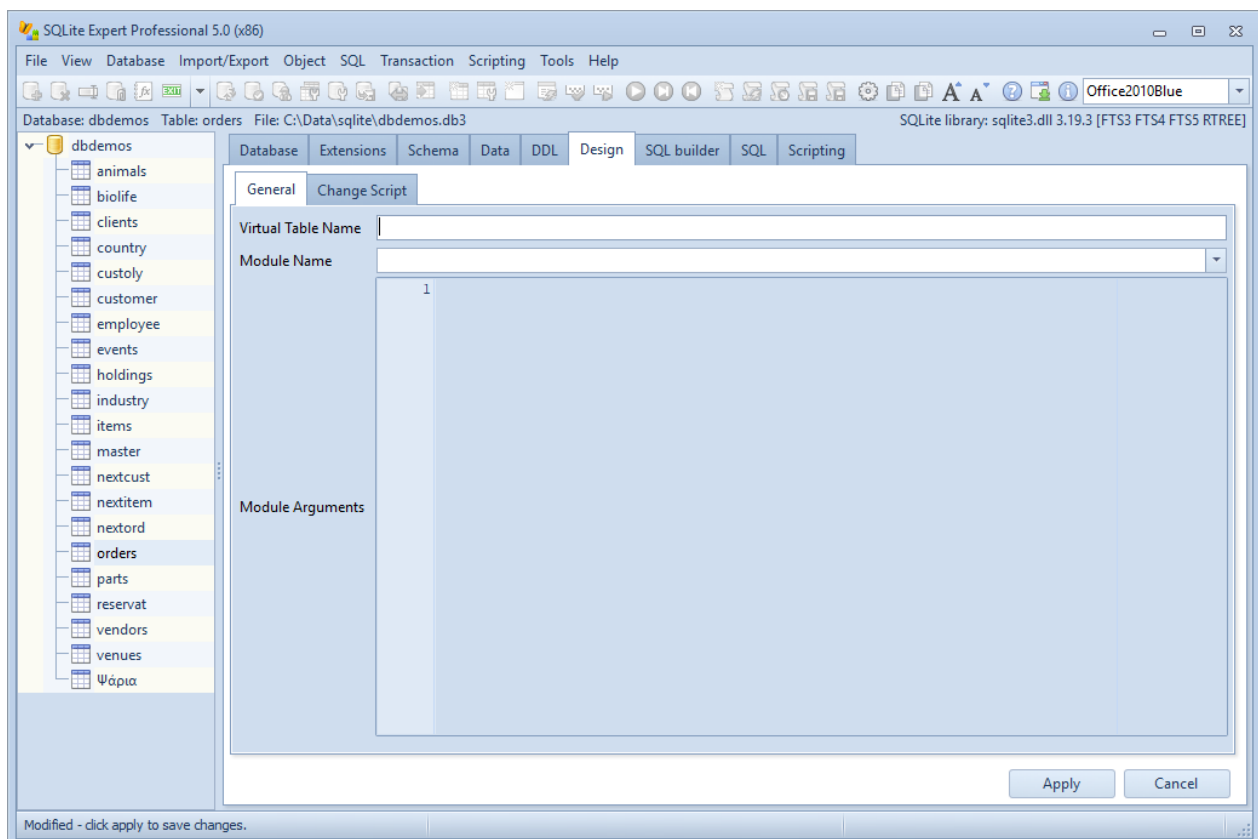
Creating a virtual table

To create a virtual table in the selected database, follow the steps below:

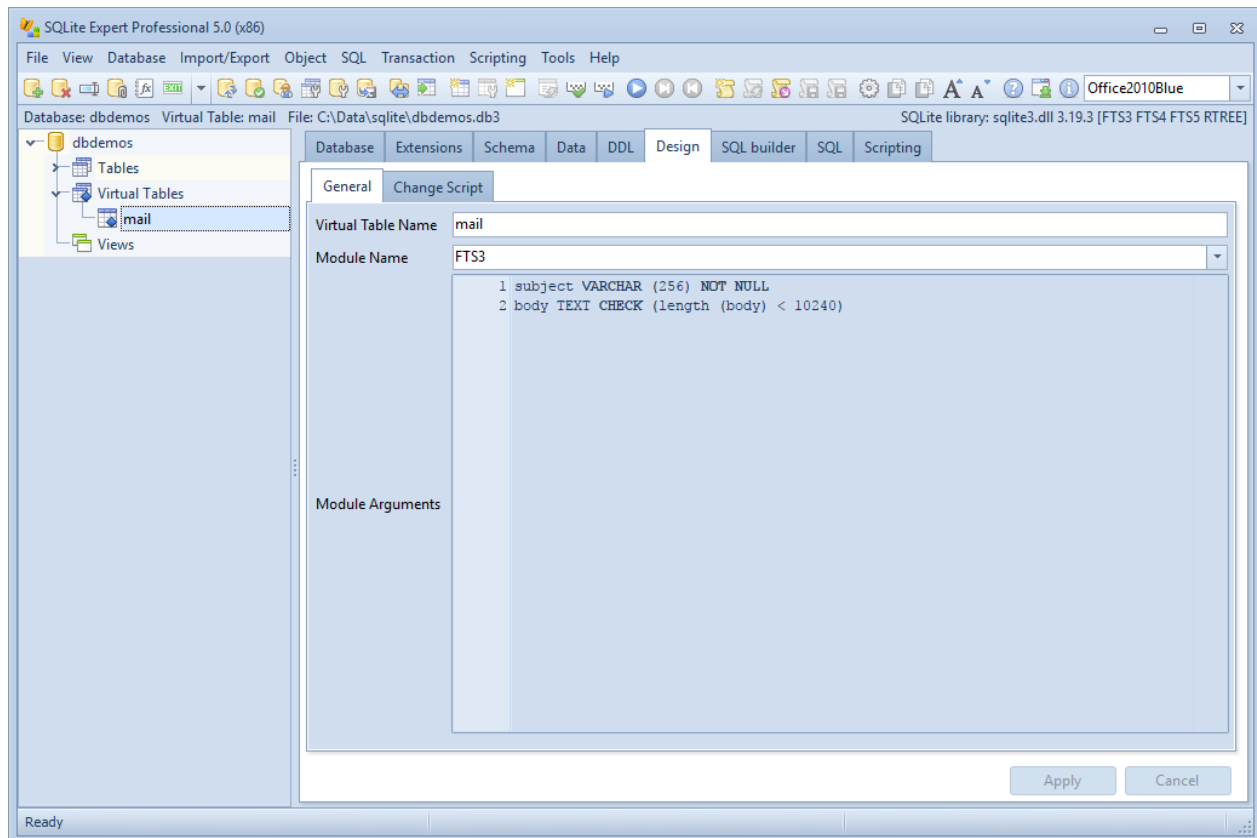
1. Select

Object » New Virtual Table

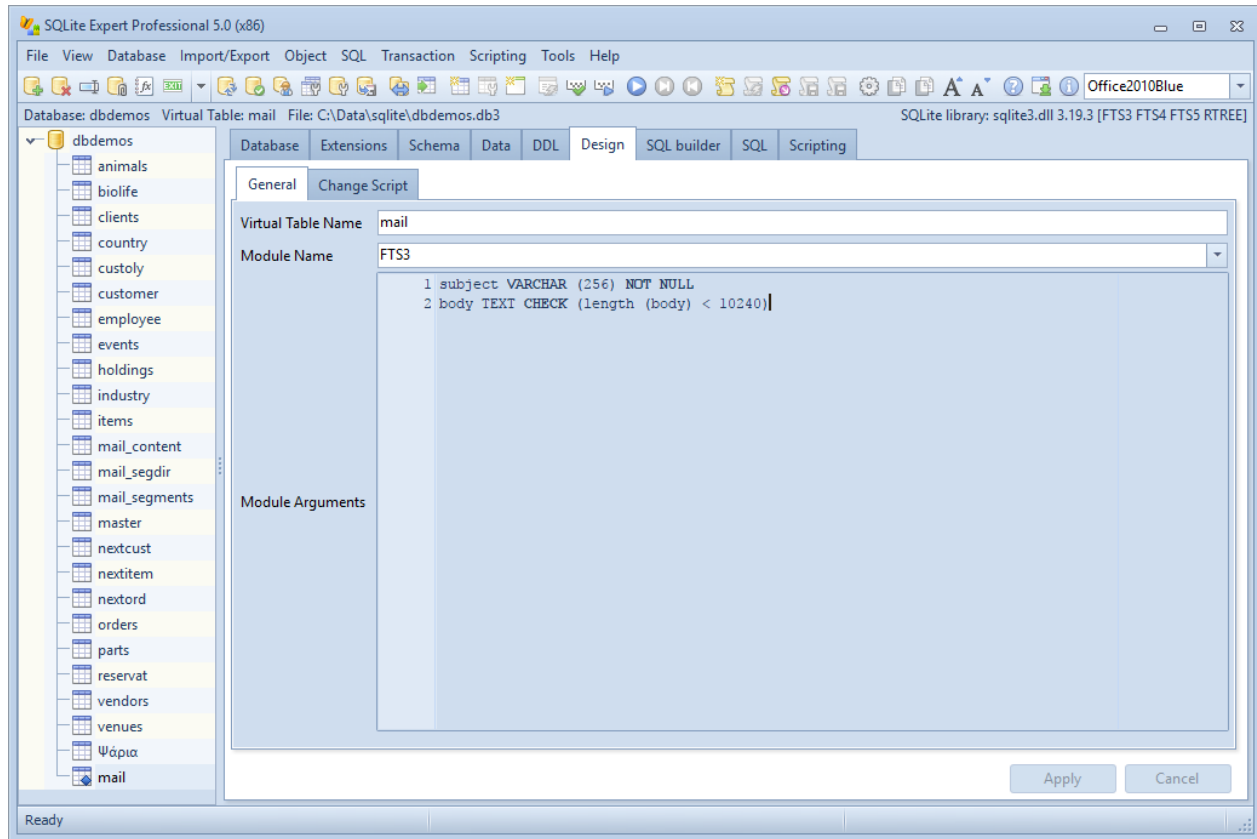
from the main menu, or **New Virtual Table** from the left tree panel popup menu. This enters the virtual table editing mode:



2. In the virtual table editor, enter the name of the virtual table, the module name and the arguments.



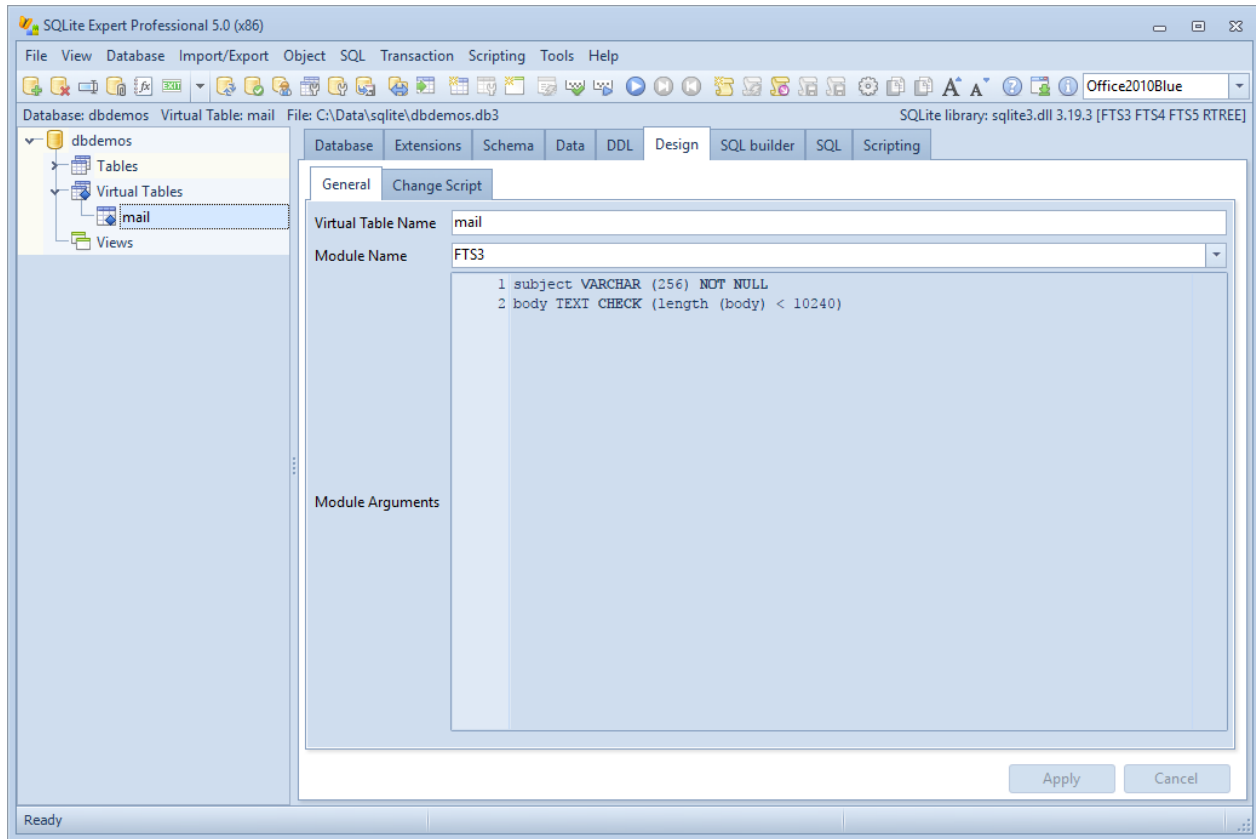
3. Click **Apply**. The virtual table is created and the creation SQL can be inspected on the DDL tab:



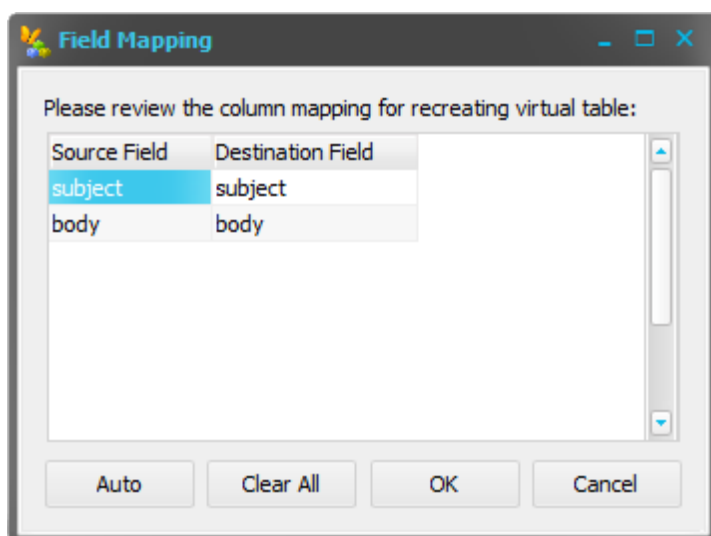
Editing a virtual table

To edit an existing virtual table, follow the steps below:

1. Select the virtual table in the left tree panel.
2. Go to the Design tab.



3. Perform the desired changes, then click **Apply**. The new table is created. If the old table contains any records then the data is copied from the old table to the new table in which case you will be prompted to review the column mapping for the data copy operation:



4. Click **OK** to copy the data from the old table to the new table.

Renaming a virtual table

Renaming a virtual table is similar to renaming a table or a view:

1. Select the virtual table you wish to rename in the left tree panel.
2. Select **Object » Rename Table** from the main menu.
3. Enter the new name for the virtual table in the table rename popup dialog.
4. Click **OK**.

Deleting a virtual table

To delete one or more virtual tables in the selected database, select the tables using the mouse or the keyboard using CTRL or SHIFT operations, then press **DELETE** or select **Delete** from the left tree panel popup menu, then click **OK** to confirm.

Editing views

» [Creating a view](#)

» [Editing a view](#)

» [Deleting a view](#)

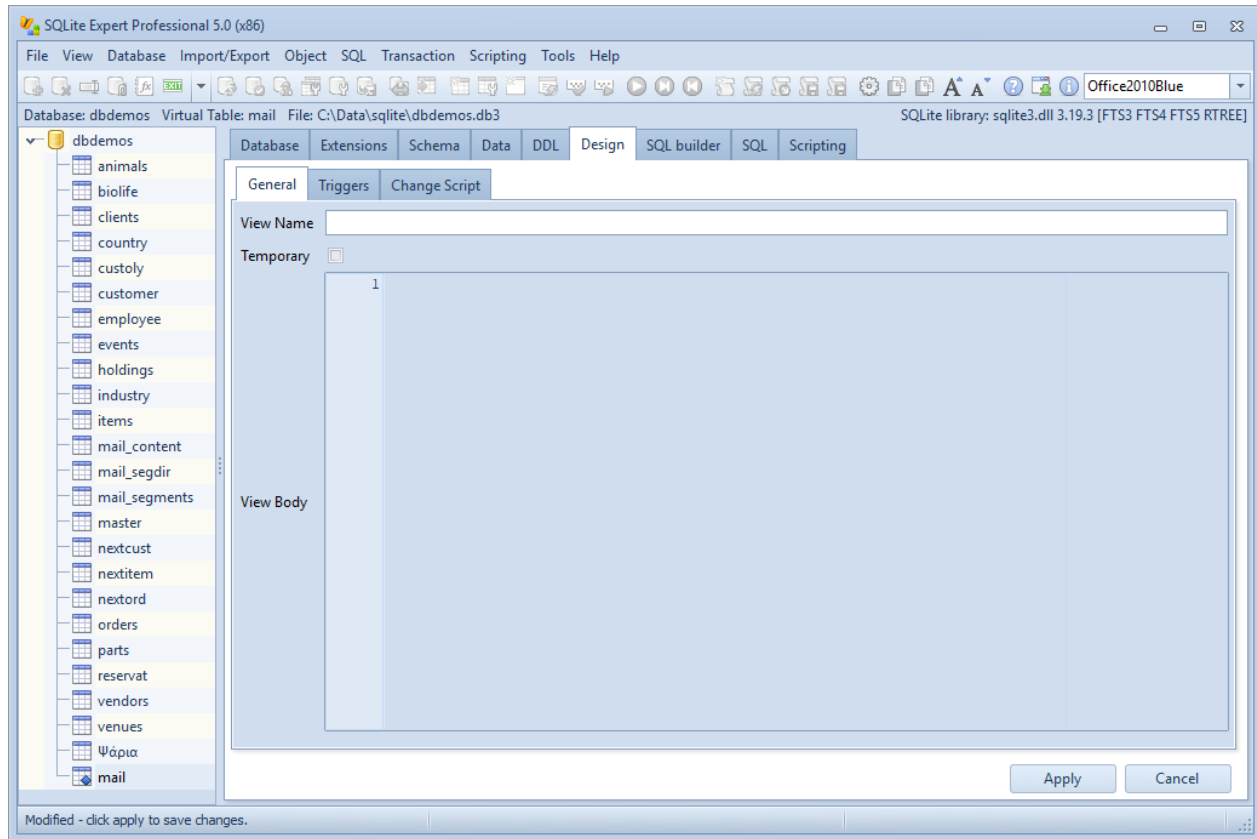
Creating a view

To create a view, follow the next steps:

1. Select

Object » New View

from the main menu, or **New View** from the left tree panel popup menu. This enters the view editing mode:



2. Enter a name for the view, and the SQL text for the view body (the AS clause).
3. If you want to create a temporary view, check the 'Temporary view' checkbox.
4. Click the **Apply** button.

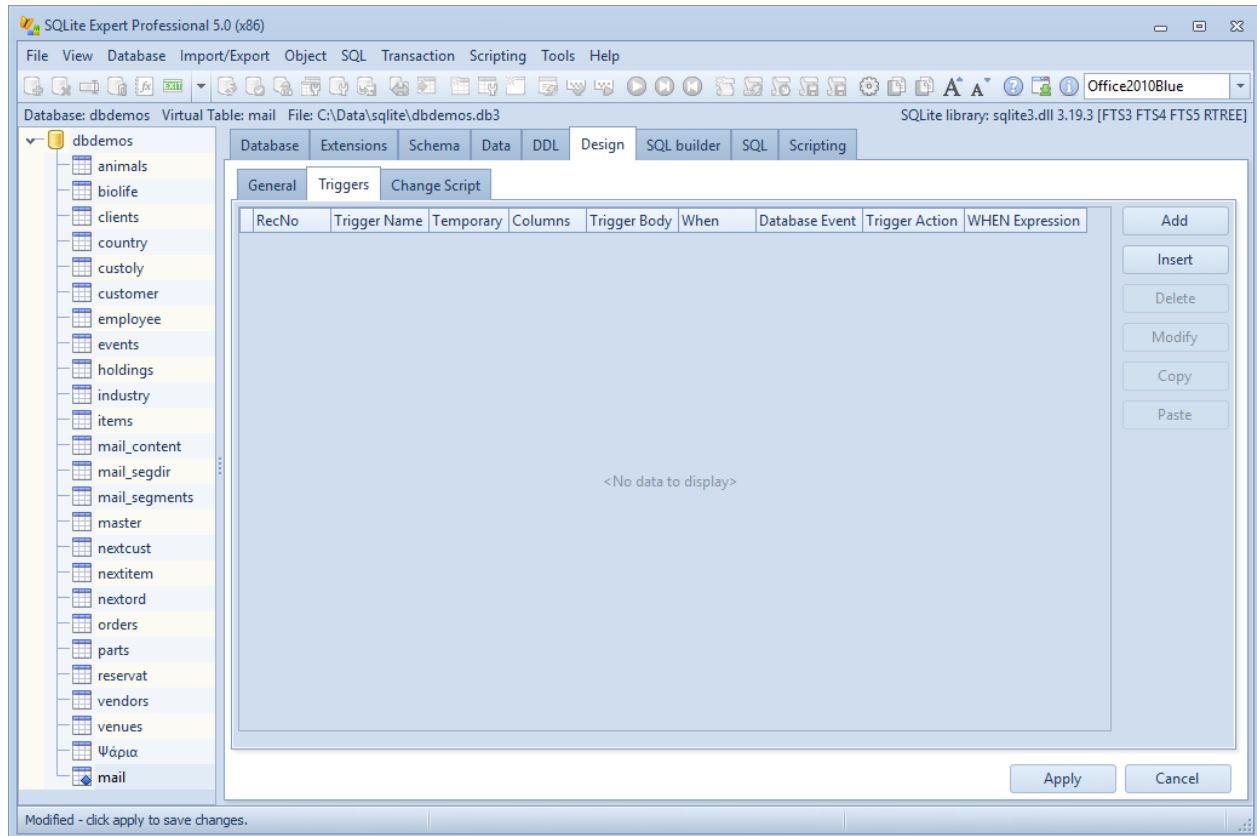
Note: there is a known issue in SQLite regarding invalid field names in views with no field aliases. You can find more information about this subject [here](#).

Editing a view

To edit an existing view, select it in the left tree panel and go to the Design tab. Here you can change the name of the view and the SQL text of the AS clause.

Editing view triggers

To add, delete or modify triggers for a view, go to the Triggers tab on the inner tab view.



To add a trigger, click the **Add** button. This command pops up the trigger editing window where you can enter the name and all the parameters for the trigger. When you are finished with the changes, click the **OK** button. To cancel the changes, click the **Cancel** button.

To delete a trigger, select it in the list and click the **Delete** button.

To modify a trigger, select it in the list and click the **Modify** button. This command pops up the trigger editing window where you can enter the name and all the parameters for the trigger. When you are finished with the changes, click the **OK** button. To cancel the changes, click the **Cancel** button.

Applying changes

When you are done with the changes, click the **Apply** button. This command will delete the old view and create a new one with the selected structure.

To cancel view editing before applying changes, click the **Cancel** button.

Note: SQLite Expert will automatically start a nested transaction before attempting to apply changes. If any errors occur, the transaction will be rolled back and you will be given the chance to correct the error before another attempt to apply the changes.

Deleting a view or a group of views

To delete one or more views in the selected database, select the views using the mouse or the keyboard using CTRL or SHIFT operations, then press **DELETE** or select **Delete** from the left tree panel popup menu, then click **OK** to confirm.

Using SQL scripts

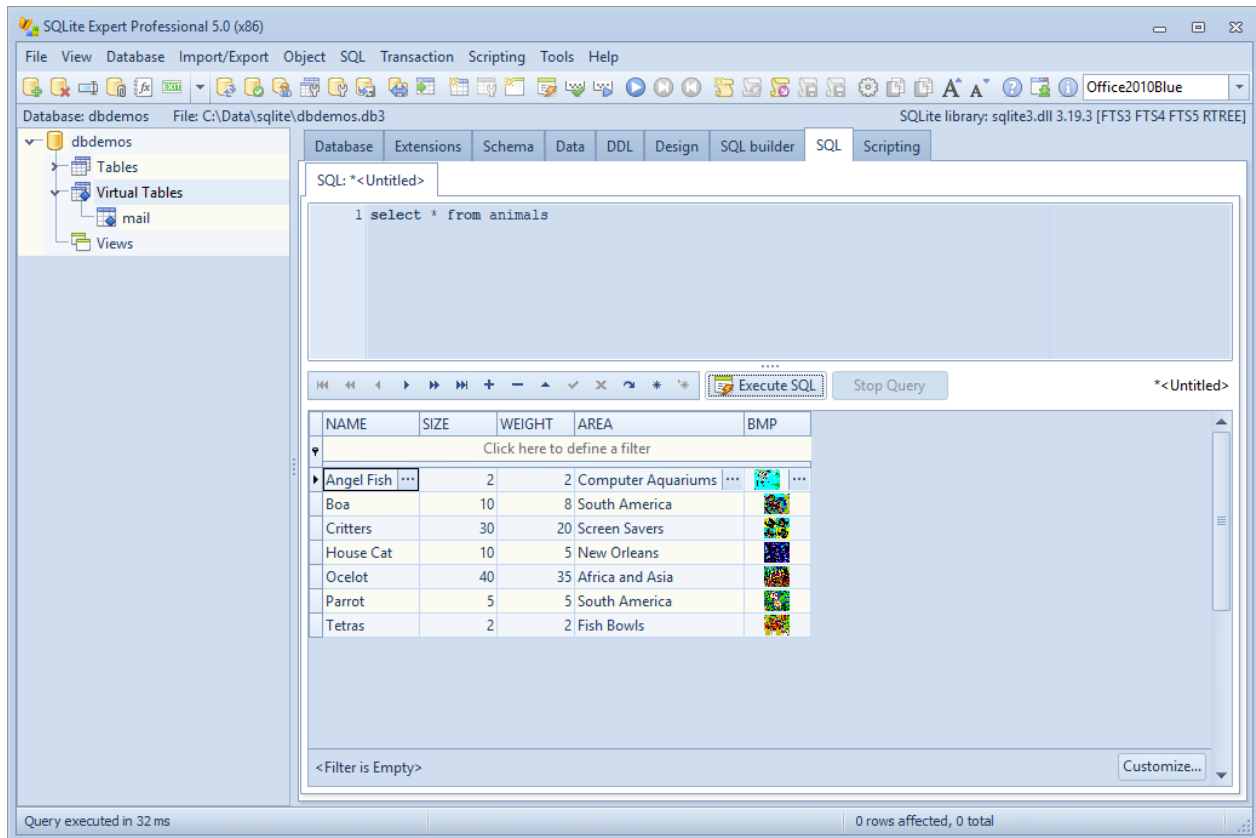
- » [Executing SQL scripts](#)
- » [Saving and loading SQL scripts](#)
- » [Creating a View from an SQL script](#)
- » [Using parameters in SQL scripts](#)

Executing SQL scripts

To execute an SQL script in the selected database, go to the SQL tab, type the SQL in the upper edit window and press F5 or select

SQL » Execute

If the SQL script is a SELECT statement the result set will be displayed in the grid or as text, depending on the current settings.



If the SQL script contains multiple statements separated by semicolon, they will all be executed. However, only the results for the last statement (if it is a SELECT statement) will be displayed.

You can also execute only a part of the SQL script by selecting the text to execute and pressing F5.

Cancelling the execution of a long-running SQL script

To cancel the execution of an SQL script, press ESC or select

SQL » Stop Query

You can also cancel the execution of a script by clicking the Stop Query button (only visible when data is displayed in a grid).

Saving and loading SQL scripts

Saving an SQL script to a file

To save an SQL script to a file, select

SQL » Save SQL Script

from the main menu, or **Save SQL Script** from the SQL window popup menu. You will be prompted to select a name for the script file.

Loading an SQL script from a file

To load an SQL script from a file, select

SQL » Open SQL Script

from the main menu, or **Open SQL Script** from the SQL window popup menu. A file dialog will allow you to choose the script file which will be loaded in the SQL window.

You can also drop script files (with .sql extension) from the Windows environment in the SQL window.

To load a recent SQL script from a file, select

SQL » Open Recent SQL Script

from the main menu, or **Open Recent SQL Script** from the SQL window popup menu. A sub-menu will allow you to choose from the most recent 10 SQL script files.

Creating a View from an SQL script

After executing an SQL script that returns a cursor (a SELECT query), you can create a view from this script by following the next steps:

1. Copy the whole script to the clipboard using CTRL+A and CTRL+C.
2. Create a new view by selecting

Object » New View

from the main menu, or **New View** from the left tree panel popup menu.

3. Paste the script into the View Body edit box.

4. Enter a name for the view.

5. If you want to create a temporary view, check the 'Temporary view' checkbox.

6. Click the **Apply** button.

Using parameters in SQL scripts

In the SQL statements accepted by SQLite Expert, literals may be replaced by a parameter in one of these forms:

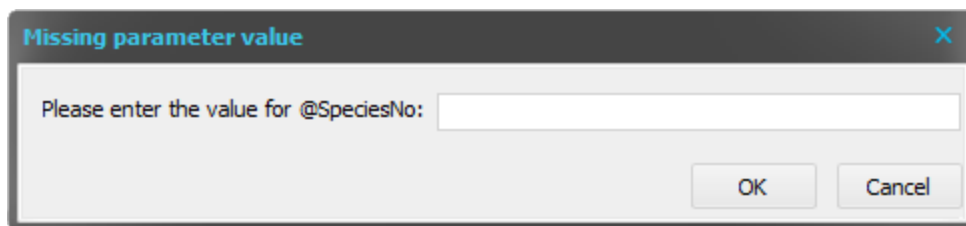
```
?
?NNN
:AAAA
@AAAA
$AAAA
```

The syntax is as described in the [SQLite documentation](#), with one limitation: SQLite Expert currently does not support "::" occurrences or suffixes enclosed in "(...)" in the \$AAAA format.

When a parameter is encountered in an SQL statement, SQLite Expert will prompt for the value of the parameter unless it was already defined. For example, in the included DBDEMOS database, executing the following statement:

```
select * from biolife where [Species No] = @SpeciesNo;
```

will prompt for the value of @SpeciesNo:



Another way to use parameters is to enter them in the script, on the SQL tab. Most of the time, SQLite Expert just reads lines of input from the SQL tab and passes them on to the SQLite library for execution. But if an input line begins with a dot ("."), then that line is intercepted and interpreted by the SQLite Expert program itself. Currently the "dot commands" are only used for setting the values of parameters used in the queries, but more commands may be added in the future.

The following "dot commands" are used with parameters:

.SET PARAM param_name = literal_value

This command sets the value of a parameter to the specified literal value. The literal value can be a string, an integer number, a floating point number, a BLOB literal, or the value **NULL**. BLOB literals are string literals containing hexadecimal data and preceded by a single "x" or "X" character.

.RESET PARAMS

This command clears the values of all the parameters.

Note: all parameters are cleared after the execution of the last statement in the list.

Example 1:

The following script will not prompt for the value of the @SpeciesNo parameter:

```
.set param @SpeciesNo = 90020;  
select * from biolife where [Species No] = @SpeciesNo;
```

Example 2:

The following script will prompt for the value of the @SpeciesNo parameter:

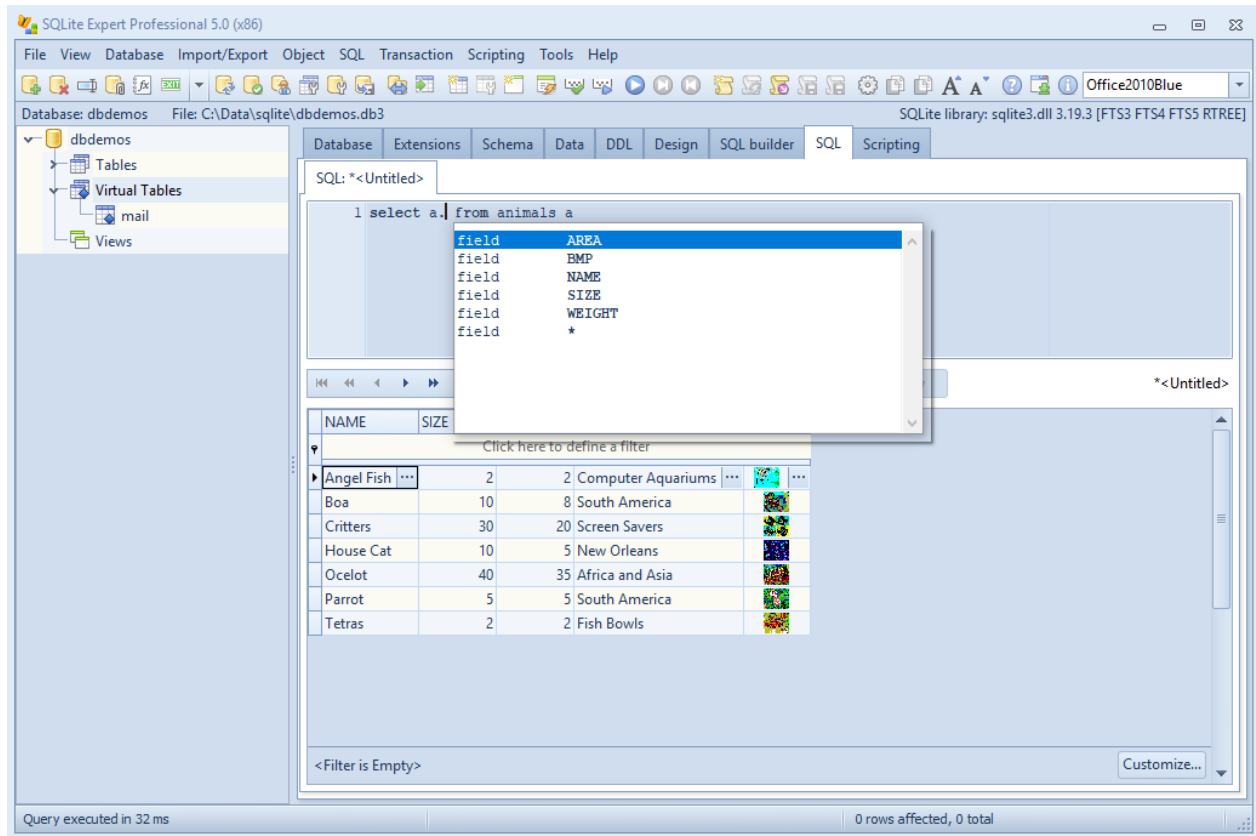
```
.set param @SpeciesNo = 90020;  
.reset params;  
select * from biolife where [Species No] = @SpeciesNo;
```

Note: the support for parameters in SQL statements using dot commands is experimental and its syntax may change without notice.

Code Completion

Code Completion

Code Completion (Ctrl+Space) is a feature available in the **SQL Editor**. Code Completion displays a resizable "hint" window that lists valid elements that you can select to add to your code.



Automatic code completion is on by default, and options for enabling and disabling **Code Completion** are located on the **Tools > Options > Code Completion** dialog box.

- ⟨ When **Auto Invoke** is enabled for Code Completion, typing a dot (.) invokes Code Completion for the SQL editor. The **Delay** sets the duration of the pause before a **Code Completion** window displays. Select a value between 0 and 10000.
- ⟨ However, you can use **Ctrl+Space** to invoke Code Completion even if the **Auto Invoke** option is disabled.

Code Template Completion

Automatically adds a code template when you type a token that starts a template and press **Ctrl+J**. The default value is On (checked). For example: type `ssf` in the SQL window, then press **Ctrl+J**. The code is replaced by the "select * from..." template:

```
SELECT * FROM
```

The following predefined templates are available:

- atr = ALTER TABLE RENAME
- ata = ALTER TABLE ADD COLUMN
- an = ANALYZE
- ad = ATTACH DATABASE
- bt = BEGIN TRANSACTION
- ct = COMMIT TRANSACTION
- ci = CREATE INDEX
- cui = CREATE UNIQUE INDEX
- ct = CREATE TABLE
- cta = CREATE TABLE AS
- ctt = CREATE TEMPORARY TABLE
- ctta = CREATE TEMPORARY TABLE AS
- ctbd = CREATE TRIGGER BEFORE DELETE
- ctbi = CREATE TRIGGER BEFORE INSERT
- ctbu = CREATE TRIGGER BEFORE UPDATE OF
- ctad = CREATE TRIGGER AFTER DELETE
- ctai = CREATE TRIGGER AFTER INSERT
- ctau = CREATE TRIGGER AFTER UPDATE OF
- ctid = CREATE TRIGGER INSTEAD OF DELETE
- ctii = CREATE TRIGGER INSTEAD OF INSERT
- ctIU = CREATE TRIGGER INSTEAD OF UPDATE OF
- fer = FOR EACH ROW
- cv = CREATE VIEW AS

- cvt = CREATE VIRTUAL TABLE
- rt = ROLLBACK TRANSACTION
- df = DELETE FROM
- dd = DETACH DATABASE
- dt = DROP TABLE
- di = DROP INDEX
- dtr = DROP TRIGGER
- dv = DROP VIEW
- et = END TRANSACTION
- ex = EXPLAIN
- i = INSERT
- ir = INSERT OR REPLACE
- ie = IF EXISTS
- ine = IF NOT EXISTS
- pg = PRAGMA
- re = REINDEX
- rs = RELEASE SAVEPOINT
- rts = ROLLBACK TO SAVEPOINT
- scf = SELECT COUNT FROM
- ssf = SELECT * FROM
- sv = SAVEPOINT
- ut = UPDATE TABLE
- vc = VACUUM

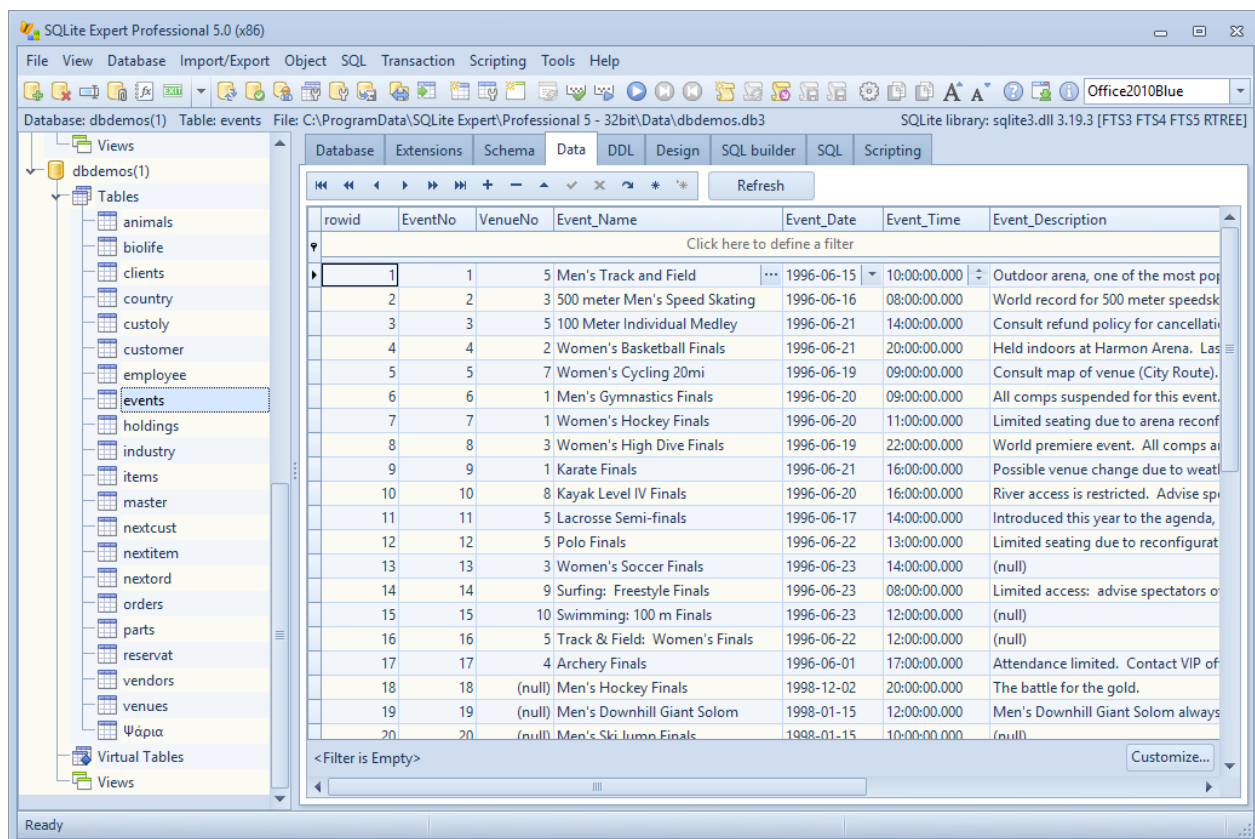
Viewing and editing data

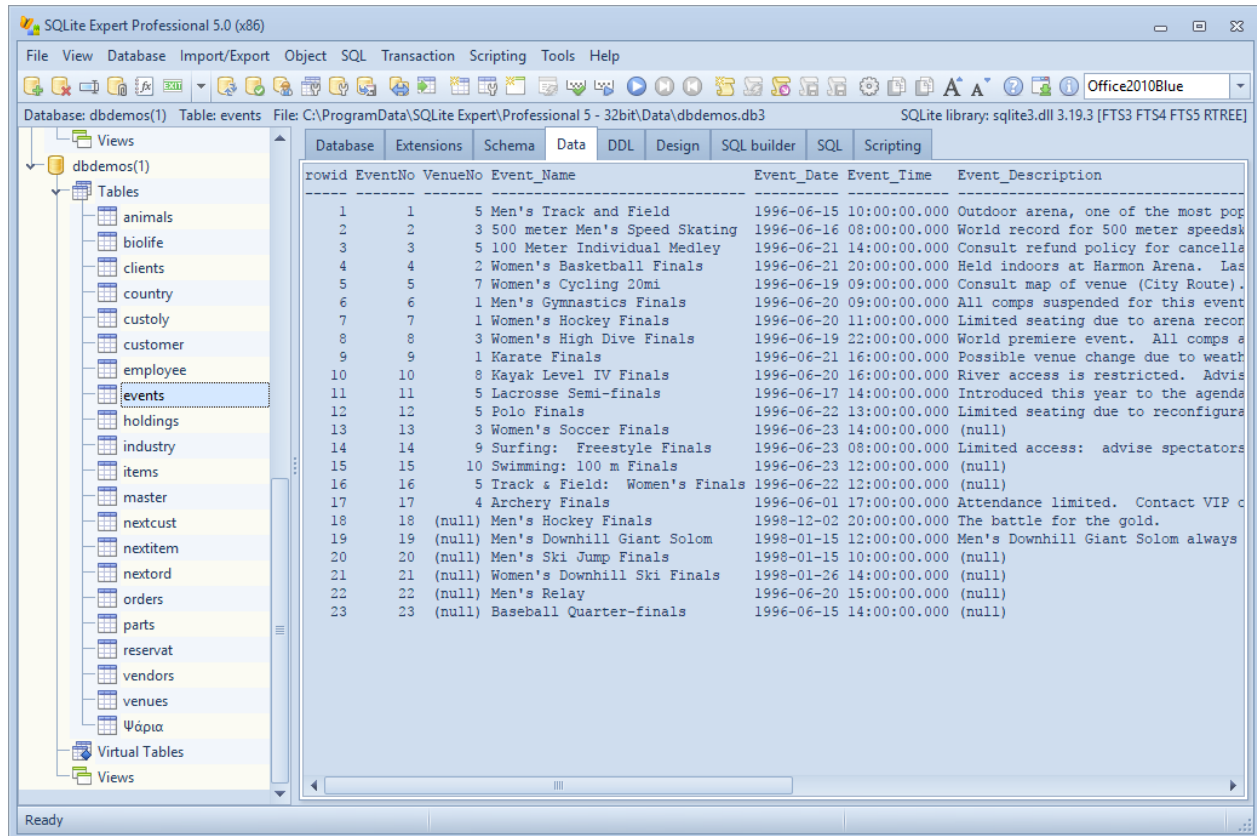
- » [Viewing table data](#)
- » [Editing table data](#)
- » [Editing live queries](#)
- » [Copying data between tables using the clipboard](#)

Viewing table data

To view the data in the selected table, go to the **Data** tab. The data will be reloaded automatically when you select a table. You can select the columns to be displayed using the 'Select columns' dialog available from the grid popup menu.

The data is displayed in a grid or as text, depending on your [Options](#).

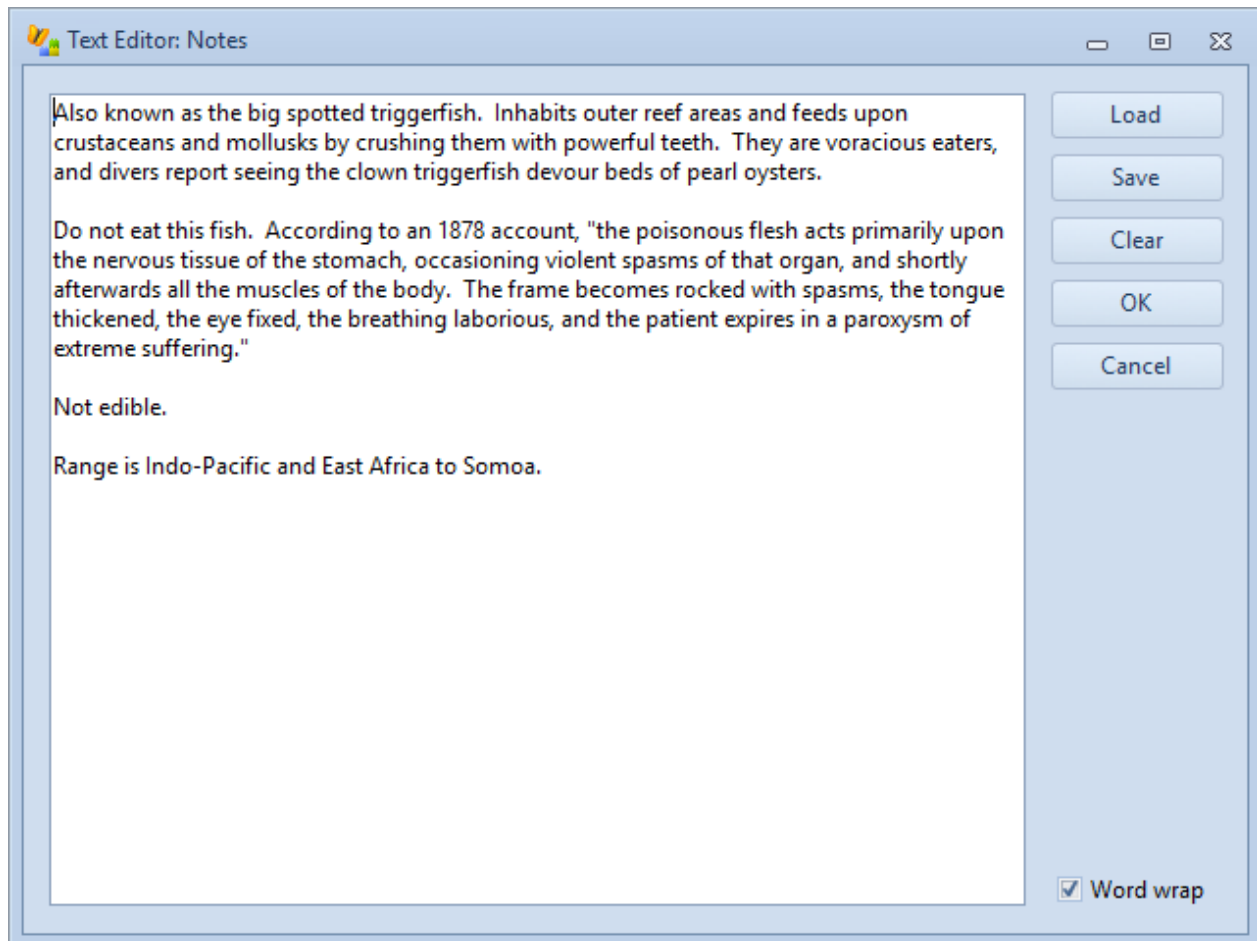




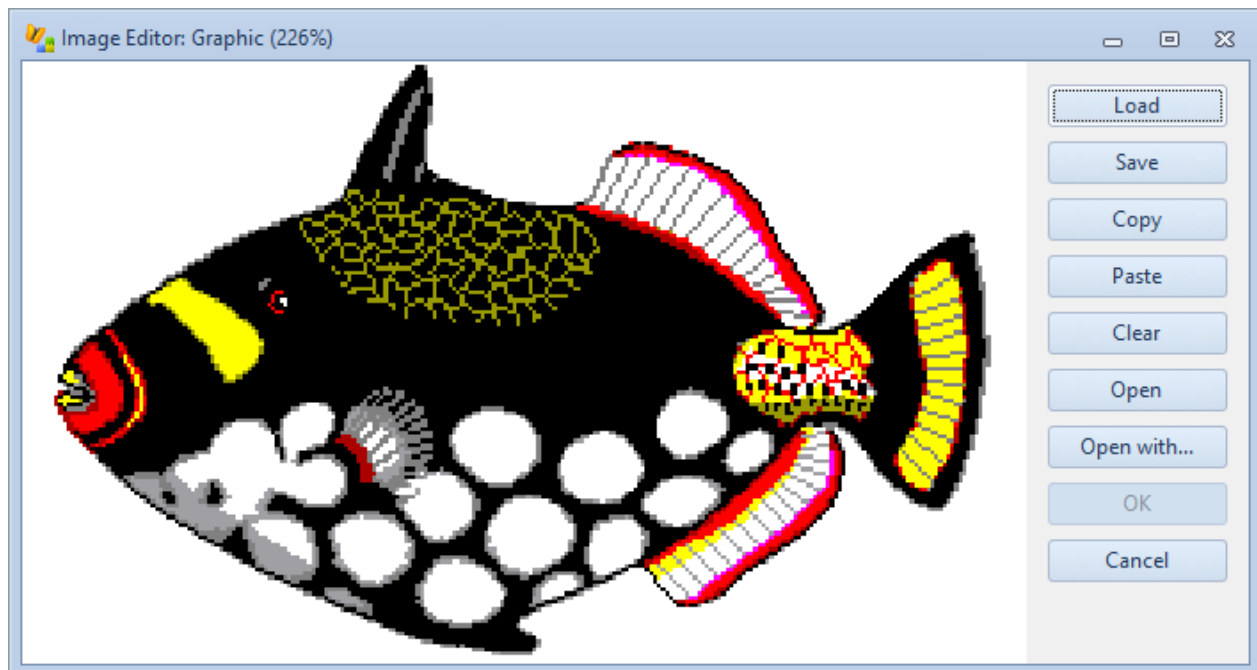
Editing table data

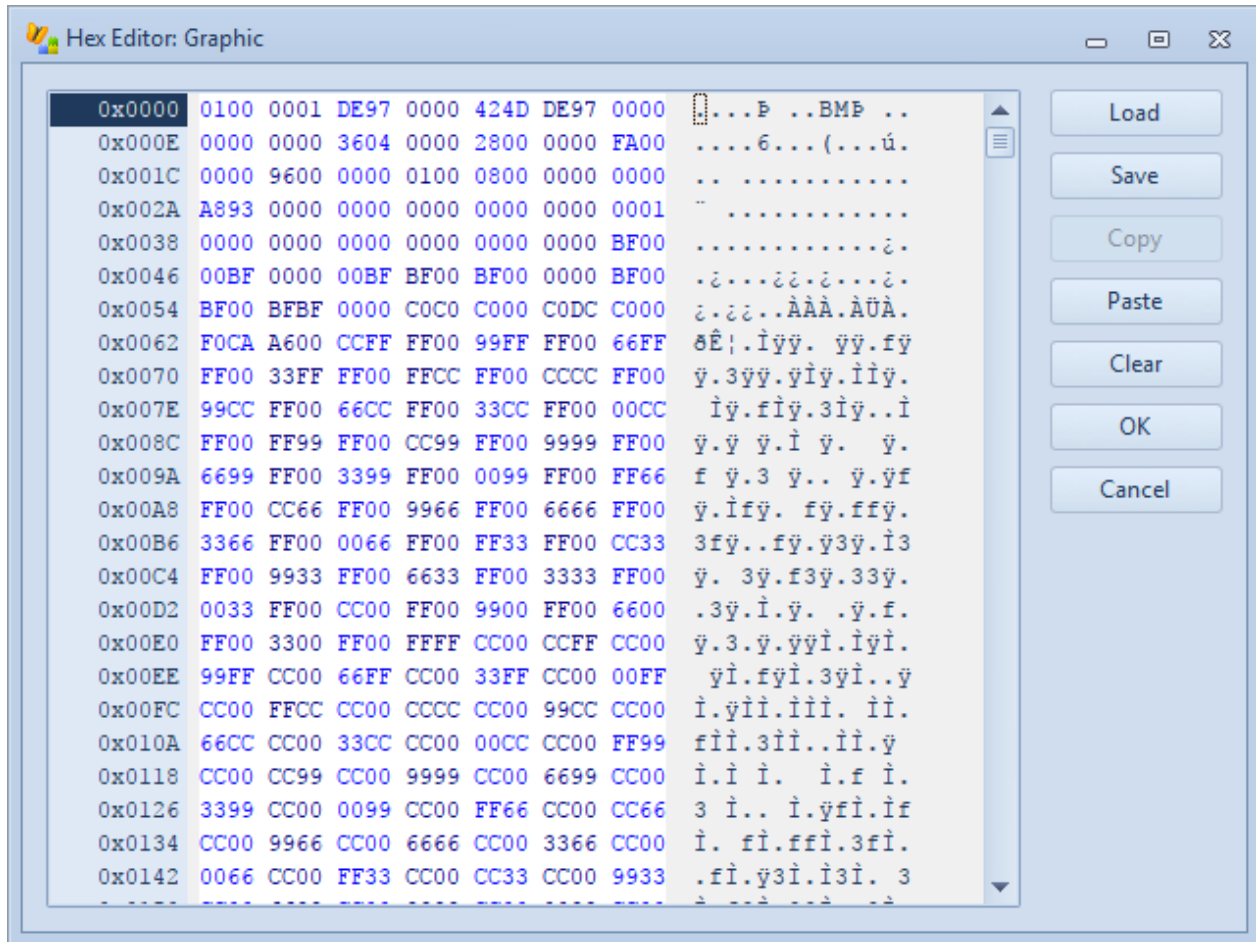
If the data is displayed in a grid, you can edit the records in the grid or using the appropriate editor. SQLite Expert provides three type of field editors:

◁ The **Text Editor** for memo fields (text fields such as CHAR, VARCHAR with no specified size)



◁ The **Image Editor** and the **Hex Editor** for the other BLOB fields (GRAPHIC, BLOB)





The appropriate editors are available by right-clicking on the field to be edited. Non-text BLOB fields can be edited using either the Hex or Image editor.

Double-clicking on a record will bring up the record editor, if this option is enabled:

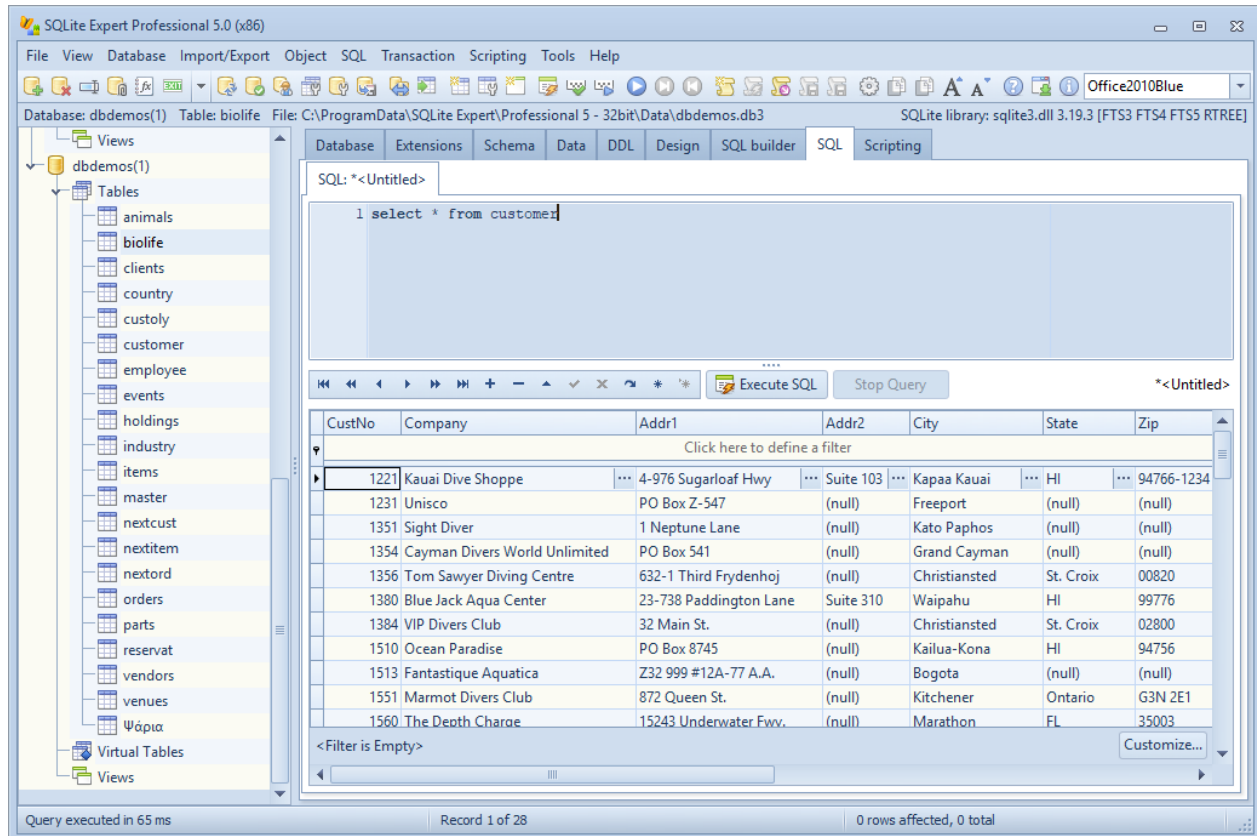
The Record Editor window displays the following data for Record 1:

Field	Value
rowid	90020
Species No	90020
Category	Triggerfish
Common_Name	Clown Triggerfish
Species Name	Ballistoides conspicillum
Length (cm)	50
Length_In	19.6850393700787

The window includes a toolbar with navigation and editing icons, and OK/Cancel buttons at the bottom right.

Editing live queries

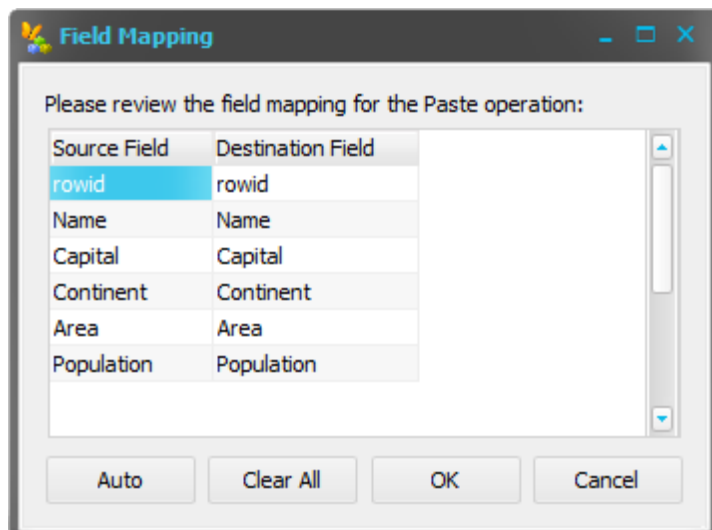
The result set of a SELECT SQL query is editable if the query contains data from a single table. Editing the result set of a query is similar to [editing the data in a table](#), the only difference being that the grid is displayed on the SQL tab.



Copying data between tables using the clipboard

To copy data between tables using the clipboard, follow the next steps:

1. Select the source table and the Data tab.
2. Select the records in the source table using the mouse or the keyboard and press **CTRL-C**.
3. Select the destination table and go to the Data tab, then press **CTRL-V**.
4. The Field Mapping dialog pops up:



5. Map the source fields to the destination fields, or press the **Auto** button to match the source fields to destination fields by name.
6. Press the **OK** button.
7. SQLite Expert is using a transaction during the Paste operation. If any errors are encountered, the transaction is rolled back.

Using the SQL Query Builder

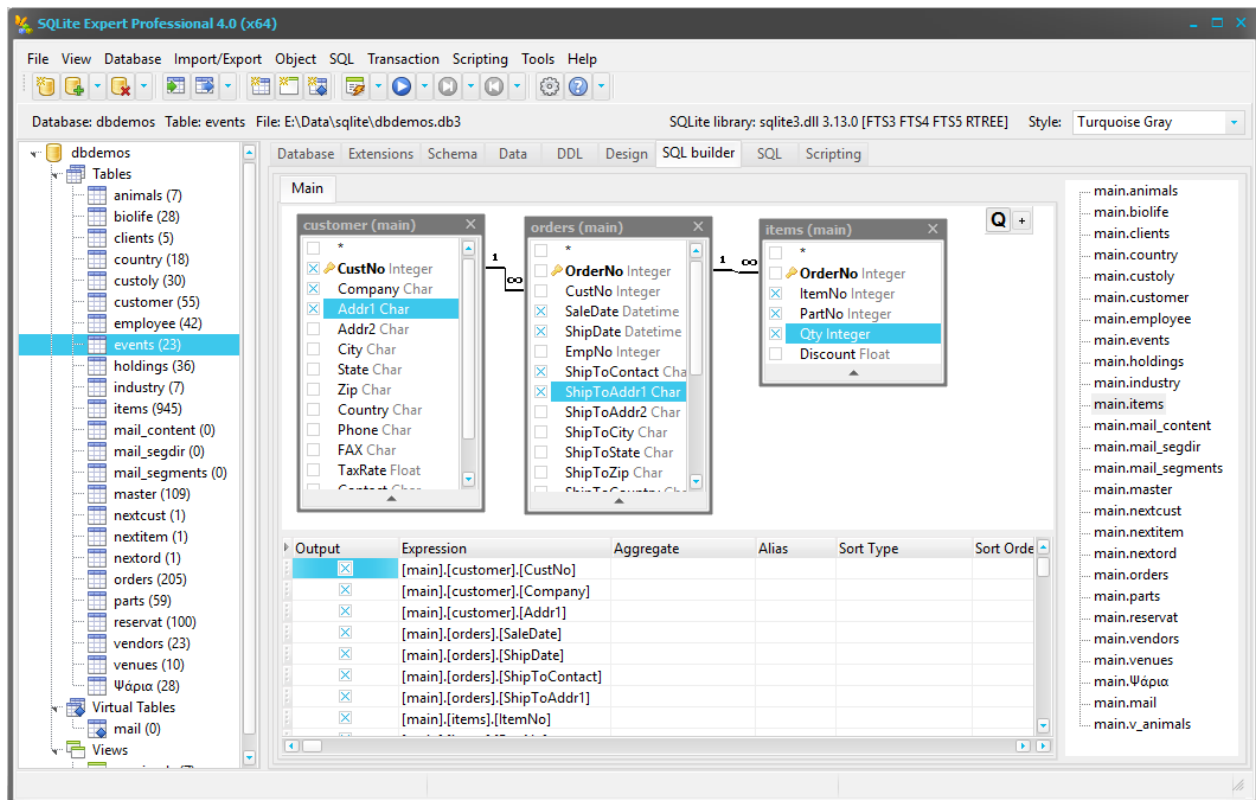
The integrated SQL Query Builder is a visual query builder component that allows you to build complex SQL queries via an intuitive visual query building interface.

To work with the Query Builder you need basic knowledge of SQL concepts. The Query Builder will help you to write correct SQL code hiding technical details, but only understanding of the SQL principles will make possible to achieve desired results.

- » [The Query Builder Layout](#)
- » [Building SQL queries using the Query Builder](#)
- » [Generating Views using the Query Builder](#)

The Query Builder Layout

To access the Query Builder, go to the SQL Builder page. The Query Builder is divided into three sections: the Table List, the Query Building Area and the Columns pane. For a better usage of the available screen space, the generated SQL script is sent to the [SQL page](#) where it can be executed like a script entered manually.



Object List

The Object List is located on the right side of the page. It contains a list of all objects (tables and views) in the database. You can drag and drop tables from the Object List to the Query Building Area.

Query Building Area

The Query Building Area is the main area where the visual representation of query will appear. This area allows you to define source database objects and derived tables, set links between them and set properties of tables and links.

Columns Pane

The Columns Pane is below the query building area. It is intended for performing all required operations with query output columns and expressions. Here you can define field aliases, sort and group out and build criteria.

The page control above the query building area will allow you to switch between the main query and sub-queries.

The small area in the corner of the query building area with letter "Q" is the union sub-query handling control. Here you may add new union sub-queries and perform all necessary operations on them.

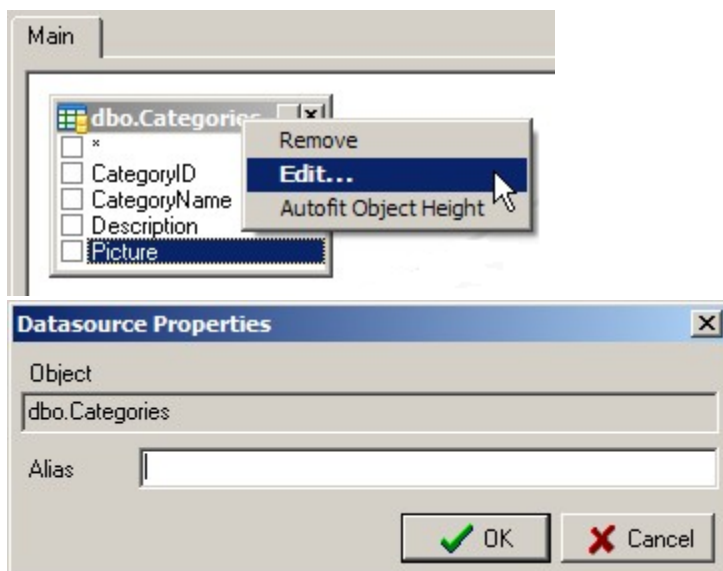
Building SQL queries using the Query Builder

To build an SQL query, follow the next steps:

1. Go to the SQL Builder page.
2. Drag and drop objects from the Object list in the Query Building Area.
3. Create joins by dragging a field from an object to another field, then set their properties by selecting **Properties** from the join drop-down menu.
4. Specify which fields you want in your query by selecting them with a checkbox next to field name.
5. Use the Columns pane to specify sort types, grouping, function or select criteria.
6. To build the query, select **Build SQL** from the Query Building Area popup menu, or press F4. The generated query will be displayed on the SQL page.
7. To build and execute the query, select **Build and Execute SQL** from the Query Building Area popup menu, or press F6. The generated query and the result set will be displayed on the SQL page.

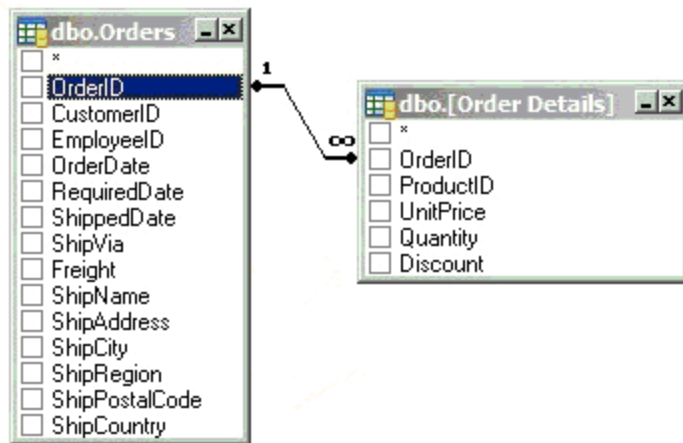
Editing object properties

You may change the properties of each object added to the query by right clicking the object and selecting the Edit... item from the drop-down menu or simply by double-clicking the object's header.

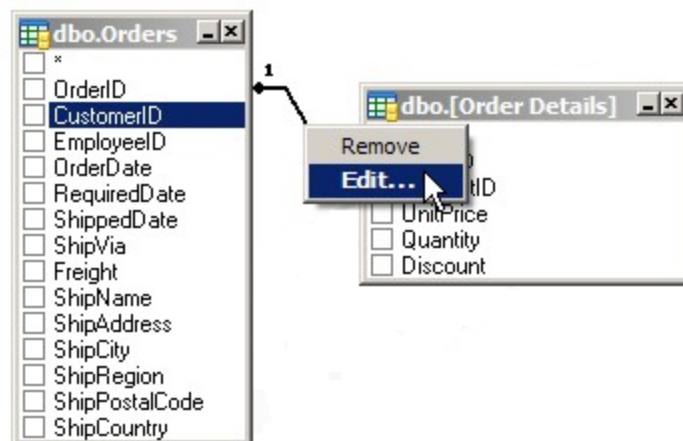


Joining tables

To create a link between two objects (i.e. join them) you should select the field by which you want to link an object with another and drag it to the corresponding field of another object. After you finish dragging, a line will appear between the linked fields.



The join type created by default is INNER JOIN, i.e. only matching records of both tables will be included in resulting dataset. To define other types of joins you should right click the link and select the Edit... item from the drop down menu or simply double-click it to open the Link Properties dialog. This dialog allows you to define join type and other link properties.



To remove a link between objects, right-click the link line and select the Remove item from the drop-down menu.

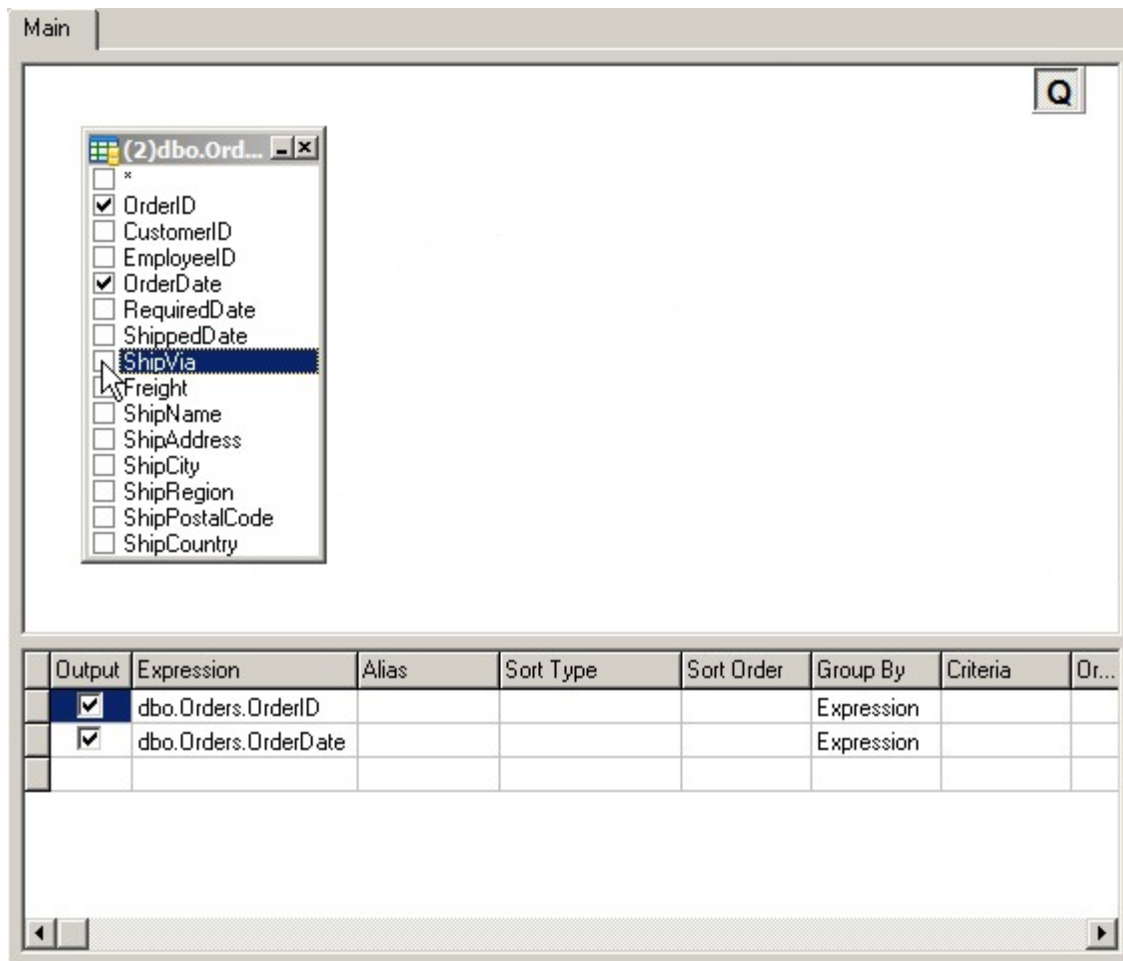
Selecting output fields

The easiest way to add a field to the list of query output fields is to check the checkbox at the left of field name in the Query Building Area. To include all the fields of an object you should click the checkbox at the left of the asterisk item of an object.

Another way is to select a field name from the drop-down list of the Expression column in the Columns Pane. And the most common way is to write any valid expression to the Expression column in the Columns Pane.

To remove a field from the list of query output fields you should uncheck the checkbox at the left of field name in the Query Building Area or you may remove it by unchecking the Output column checkbox.

Such operations as removing lines from the Columns Pane or re-ordering output fields are available by right clicking on the left-most gray column via the drop-down menu.



Output field's aliases may be defined in the Alias column of the Columns Pane:

	Output	Expression	Alias	Sort Type	Sort Order	Group By	Criteria	Or...
	<input checked="" type="checkbox"/>	dbo.Orders.OrderID	OID			Expression		
	<input checked="" type="checkbox"/>	dbo.Orders.OrderDate	ODate			Expression		

Sorting output fields

To define the sorting of output query fields you should use the Sort Type and Sort Order columns of

the Columns Pane.

The Sort Type column allows you to specify how the fields will be sorted - in Ascending or Descending order.

The Sort Order column allows you to setup the order in which fields will be sorted, if there are more than one field to sort specified.

To cancel sorting by some field you should clear the Sort Type column for this field.

	Output	Expression	Alias	Sort Type	Sort Order	Group By	Criteria	Or...
	<input checked="" type="checkbox"/>	dbo.Orders.OrderID	OID			Count		
	<input checked="" type="checkbox"/>	dbo.Orders.OrderDate	ODate			Group By		
						Expression		
						Where		
						Group By		
						Having		
						Count		

Defining criteria

To define criteria for the expression listed in the Columns Pane you must use the Criteria column.

Here you should write the criteria omitting the expression itself. For get the following criteria in your query

WHERE (field >= 10) AND (field <= 20)

you should write

>= 10 AND <= 20

in the Criteria column.

You may specify several criteria for one expression using the Or... columns. These criteria will be concatenated in the query with the OR operator.

Grouping output fields

To setup grouping by some of the fields and/or to define aggregate functions on grouped rows you may use the Group by column.

You may select one of the following values for this column from the drop-down list:

- "Expression" and "Where": These values are used when no grouping is specified. The "Expression" value is set when this expression is used as output expression in the SELECT clause and nothing else. The "Where" value is set automatically when you define a criteria to this expression that results in including this expression to the WHERE clause. Normally you shouldn't care about value of the Group By column when you don't want to define grouping.

- "Group by" and "Having": These values are similar to the previous two, but used when you want to define grouping in your query. In this case you should set the "Group by" value for all columns you want to group by. Specifying criteria for the grouped columns will include these criteria in the HAVING clause. If you want to include an expression ONLY in the HAVING clause you should set

the "Having" value in the Group By column for this expression.

· Aggregate functions (Count, Sum, etc): By selecting one of these values you will create an aggregate expression for the value indicated in the Expression column.

Output	Expression	Alias	Sort Type	Sort Order	Group By	Criteria	Or...
<input checked="" type="checkbox"/>	dbo.Orders.OrderID	OID			Count		
<input checked="" type="checkbox"/>	dbo.Orders.OrderDate	ODate			Group By		
					Expression		
					Where		
					Group By		
					Having		
					Count		

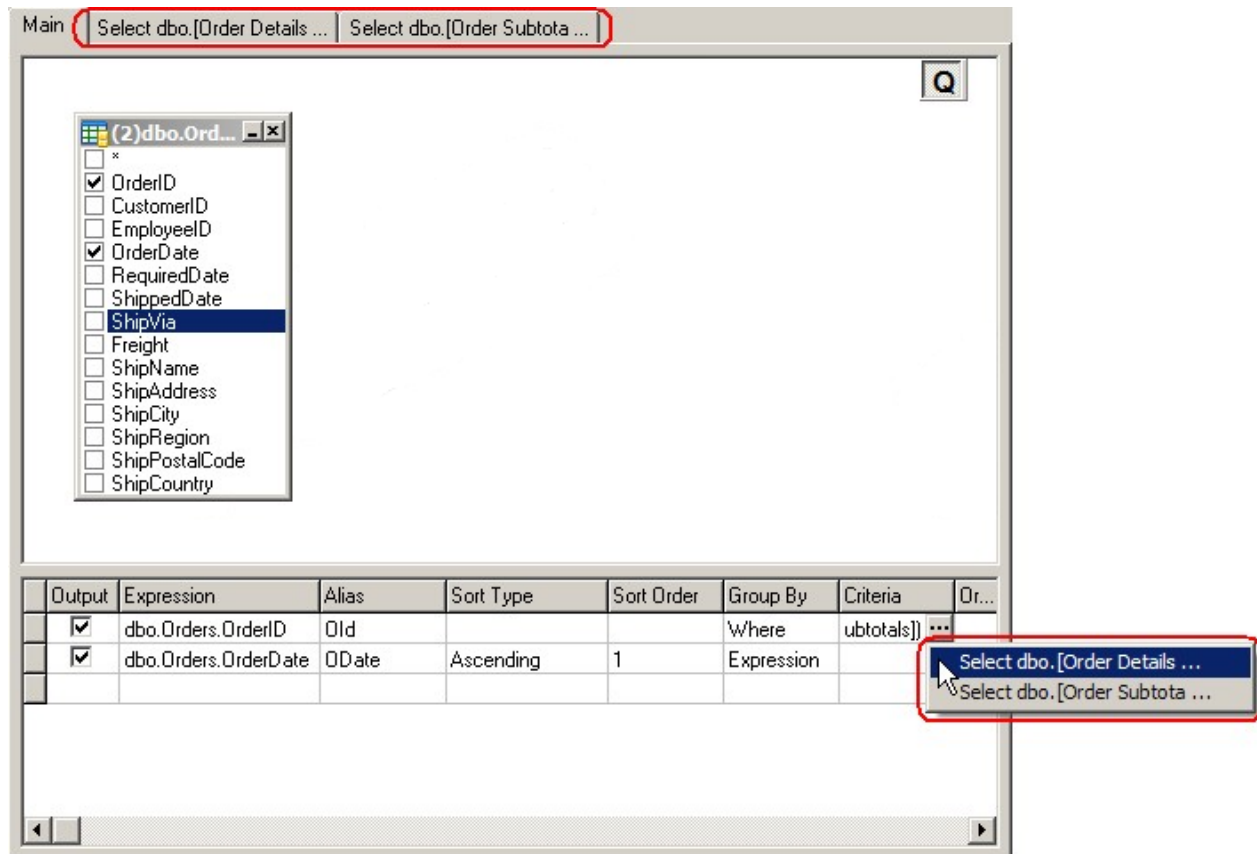
Adding a sub-query

You may add a sub-query as a part of expression or criteria using the Columns Pane.

To add a sub-query right click on the Expression or Criteria column of the Columns Pane and select the Insert SubQuery item from the drop-down menu. Another way to add a sub-query is to type "(Select)" in the Expression column and "= (Select)" or "In (Select)" or any other valid sub-query expression in the Criteria column.

Output	Expression	Alias	Sort Type	Sort Order	Group By	Criteria	Or...
<input checked="" type="checkbox"/>	dbo.Orders.OrderID	OID			Where		
<input checked="" type="checkbox"/>	dbo.Orders.OrderDate	ODate	Ascending	1	Expression		

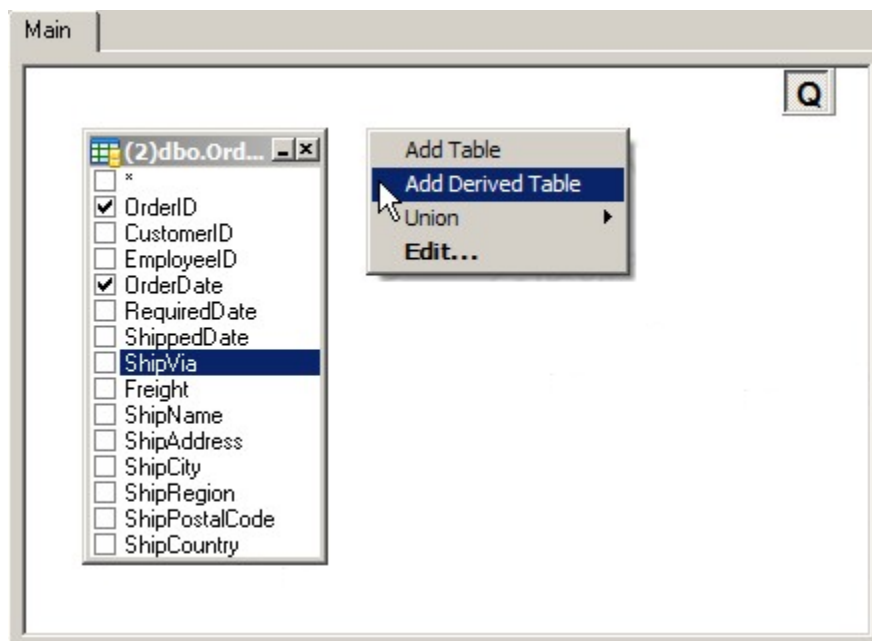
The corresponding tab will be created after adding a sub-query. This tab allows you to edit your sub-query visually in the same way as you edit the main query. Another way to switch to the sub-query tab is to press the ellipsis button of the cell where this sub-query is located. If this cell contains more than one sub-query, the drop-down menu will be shown for you to select the right sub-query.



Adding a derived table

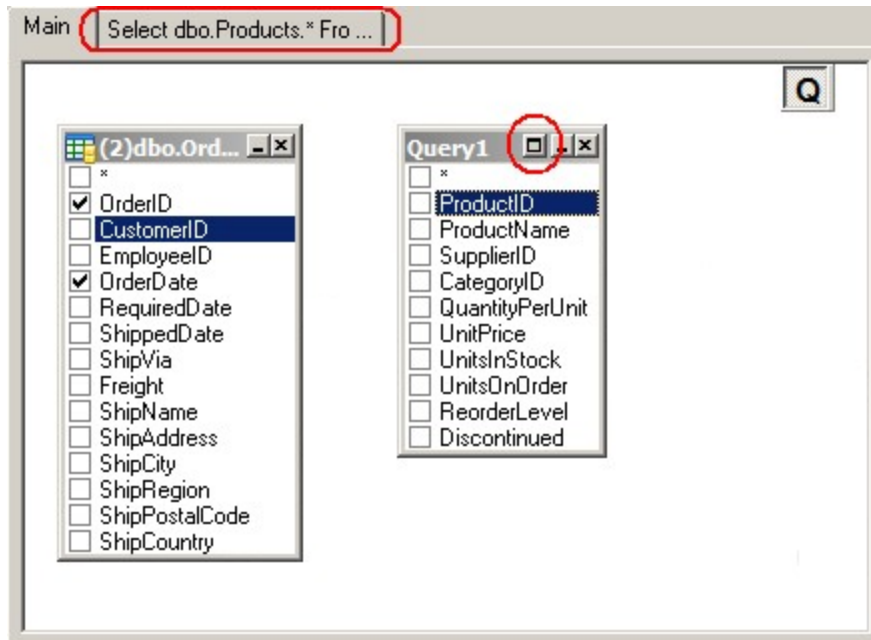
You may add a derived table (a sub-query used in the FROM clause) using the Query Building Area.

To add a sub-query right click on the Query Building Area and select the Add Derived Table item from the drop-down menu.



A new object representing newly created derived table will be added to the query building area of

the main query. Also the corresponding tab will be created after adding a derived table. This tab allows you to edit the sub-query visually in the same way as you edit the main query. Another way to switch to the sub-query tab is to press the maximize button at the header of the object representing this derived table.



Working with unions

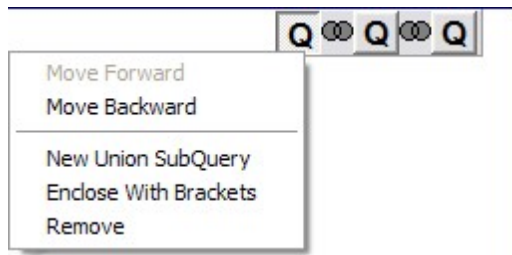
Working with union sub-queries is implemented via the small panel in the top-right corner of the Query Building Area. Initially there is only one union sub-query labeled with letter "Q" in it. All required operations are performed by means of the context drop-down menus.

- To add a new union sub-query select the New Union SubQuery menu item.
- To enclose a sub-query with brackets select the Enclose with Brackets menu item.
- To move a sub-query or a bracket to the top of the query (the topmost sub-query is the left one), select the Move Backward menu item.
- To move a sub-query or a bracket to the bottom of the query, select the Move Forward menu item.
- To remove a sub-query or a bracket select the Remove menu item.
- To change uniting operator select the needed operator from the list of supported operators from the drop-down menu.

Union Sub-Query operations:

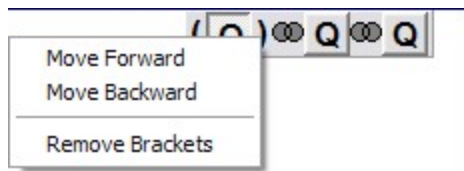
- Move Forward
- Move Backward
- New Union SubQuery
- Enclose With Brackets

- Remove

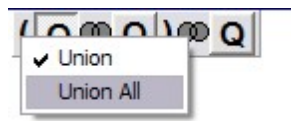


Brackets operations:

- Move Forward
- Move Backward
- Remove Brackets



Selecting uniting operator:



Generating Views using the Query Builder

You can use the Query Builder to generate views visually, without writing a line of SQL.

To create a view using the Query Builder, follow the next steps:

1. [Build an SQL query using the Query Builder](#) as described in the previous chapter.
2. [Create a View from the generated SQL script.](#)

Importing and Exporting Data

SQLite Expert provides tools for importing and exporting data from and to data sources, including Excel, text files (CSV or TSV), SQLite databases, ADO data sources, or SQL scripts. Currently the following operations are supported:

- » [Exporting data to Excel](#)
- » [Exporting data to XML](#)
- » [Exporting data to JSON](#)
- » [Exporting data to a text file](#)

- » [Importing data from a text file](#)
- » [For transferring data between SQLite, ADO and SQL scripts, use the Data Transfer Wizard](#)
- » [Copying tables between databases using the clipboard](#)

Exporting data to Excel

To export data from a grid to Excel, select **Export to Excel** from the grid pop-up menu. You will be prompted to select the destination file name.

Exporting data to XML

To export data from a grid to an XML file, select **Export to XML** from the grid pop-up menu. You will be prompted to select the destination file name.

Exporting data to JSON

To export data from a grid to a JSON file, select **Export to JSON** from the grid pop-up menu. You will be prompted to select the destination file name.

Exporting data to HTML

To export data from a grid to an HTML file, select **Export to HTML** from the grid pop-up menu. You will be prompted to select the destination file name.

Exporting data to Open Document Spreadsheet (ODS)

To export data from a grid to Open Document Spreadsheet (ODS), select **Export to Open Document Spreadsheet (ODS)** from the grid pop-up menu. You will be prompted to select the destination file name.

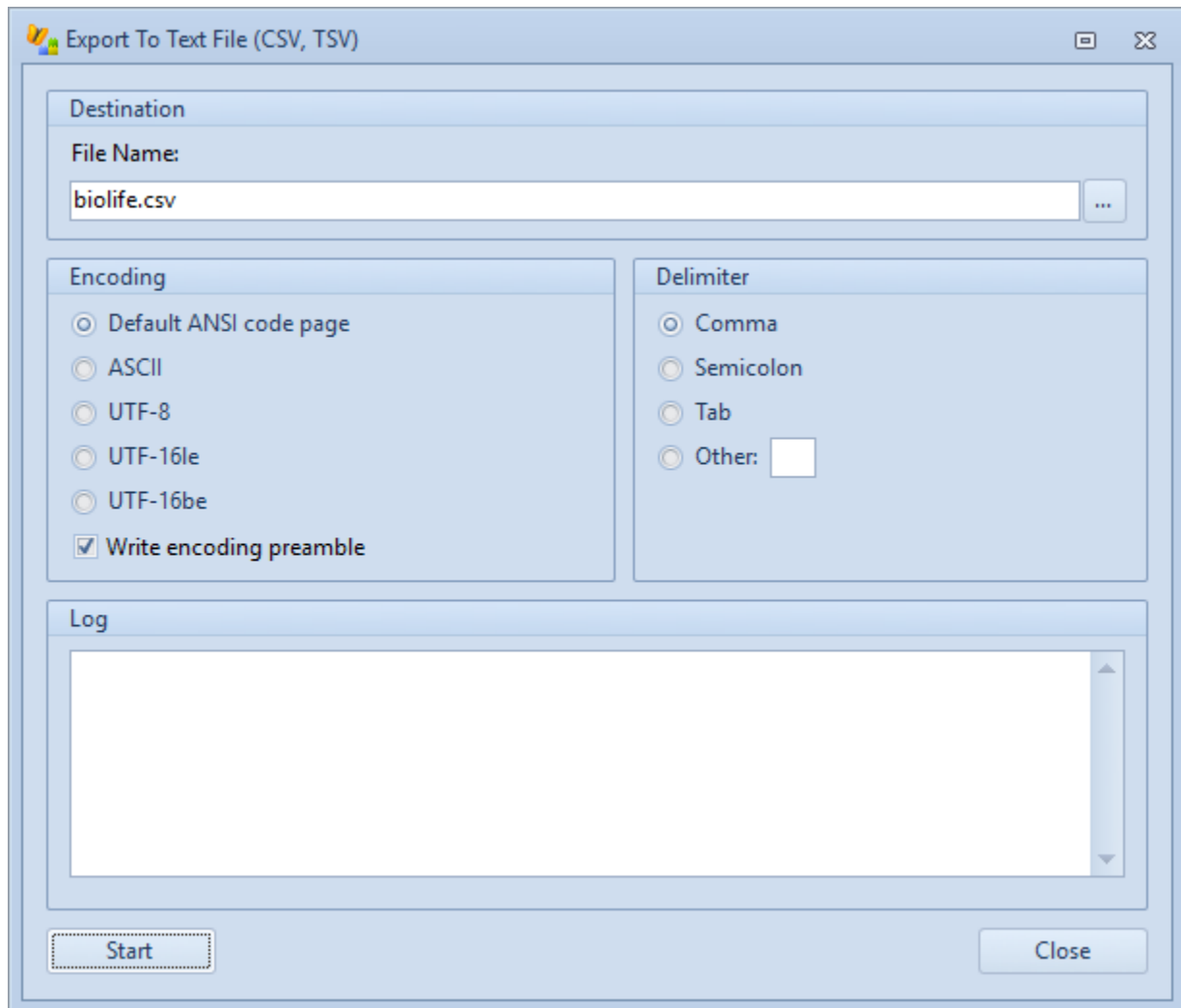
Exporting data to SQL

To export data from a grid to an SQL script, select **Export to SQL** from the grid pop-up menu. You will be prompted to select the destination file name.

Exporting data to a text file

To export data from a grid to a text file, follow the next steps:

1. Select **Export to text file** from the grid pop-up menu.
2. The 'Export text file' dialogs pops up. This allows you to choose the destination file name, encoding and delimiter. The following delimiters are supported: comma, semicolon, tab and custom.

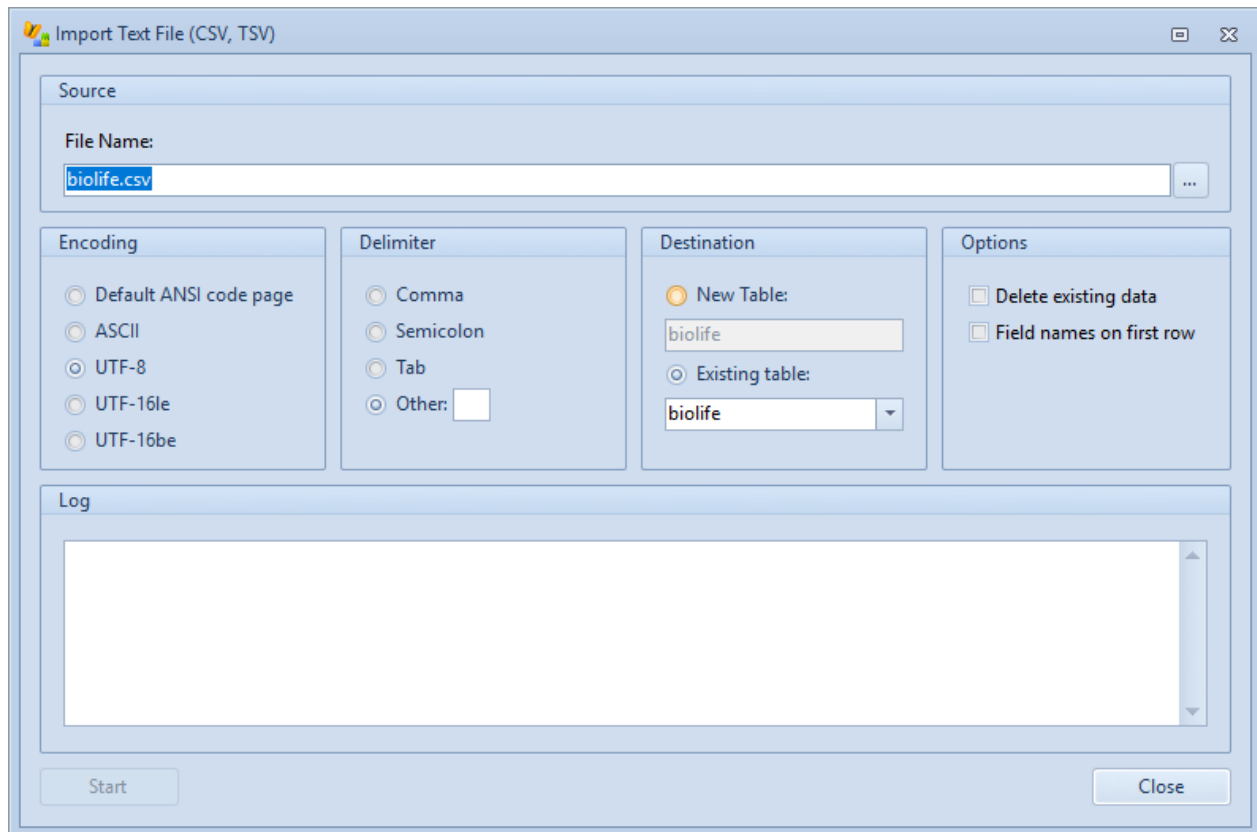


3. Choose the destination file, the encoding and the delimiter.
4. Click the 'Start' button to start the export process.

Importing data from a text file

To import data from a text file to a table in the current database, follow the next steps:

1. Select **Import/Export » Import text file** from the main menu.
2. The 'Import text file' dialog pops up. This allows you to select the source file name, the destination table and a few other options.



3. Select the text file name.
4. SQLite Expert will attempt to detect the delimiter used in the file (comma, semicolon or tab). You may wish to change it to a different value if the auto-detect function is not accurate.
5. Select the destination table. If you wish to import the data in an existing table, check the 'Existing table' and select the destination table from the drop-down list. Otherwise, select 'New table' and enter the name for the new table.
6. Select 'Delete existing data' if you wish to clear the data before importing the text file.
7. If you chose to import the text file into a new table, check 'Auto detect field types' if you want SQLite Expert to try and guess the data type based on the data in the file. This option is slower as it requires two passes through the text file. If you leave this option unchecked all the data will be imported as CHAR.
8. Click the 'Start' button to start the import process.

Using the Data Transfer Wizard

The **Data Transfer Wizard** is a powerful tool to import or export your data quickly, and it allows you to set all the import/export options for different files visually. Currently the following operations are supported:

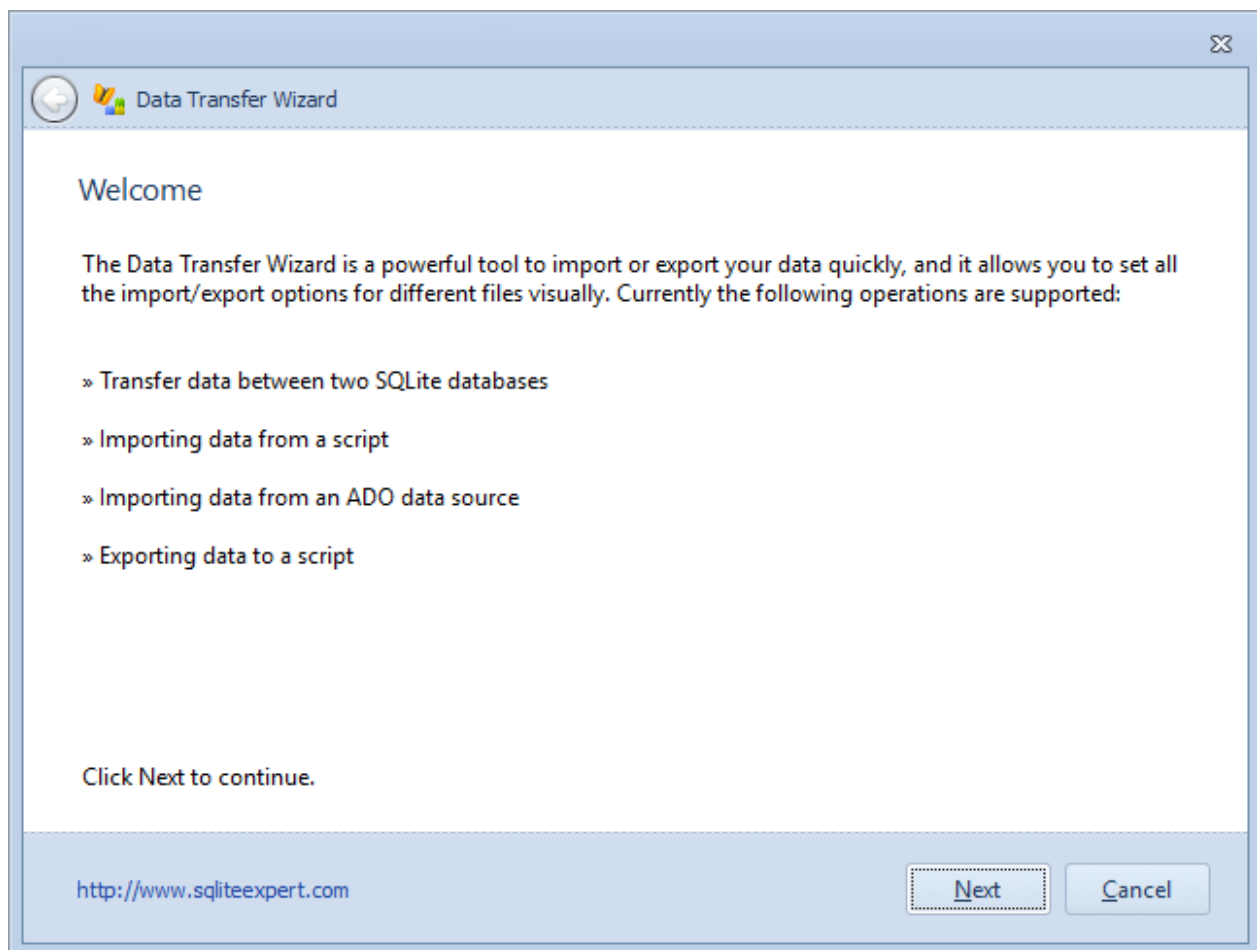
- » [Importing data from another SQLite database](#)
- » [Importing data from a script](#)

- » [Importing data from an ADO data source](#)
- » [Exporting data to another SQLite database](#)
- » [Exporting data to a script](#)

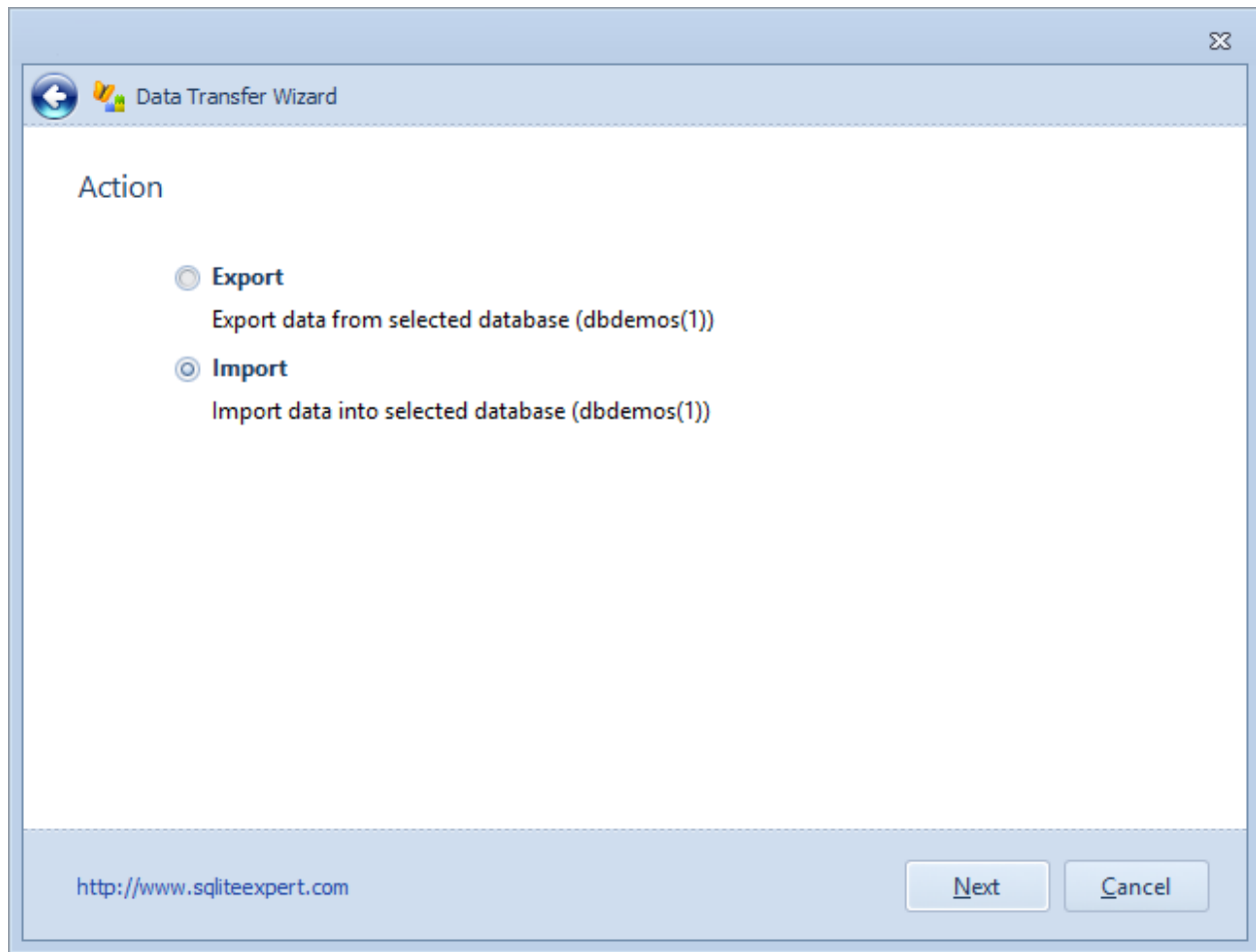
Importing data from another SQLite database

To import data into the selected database from another SQLite database, follow the next steps:

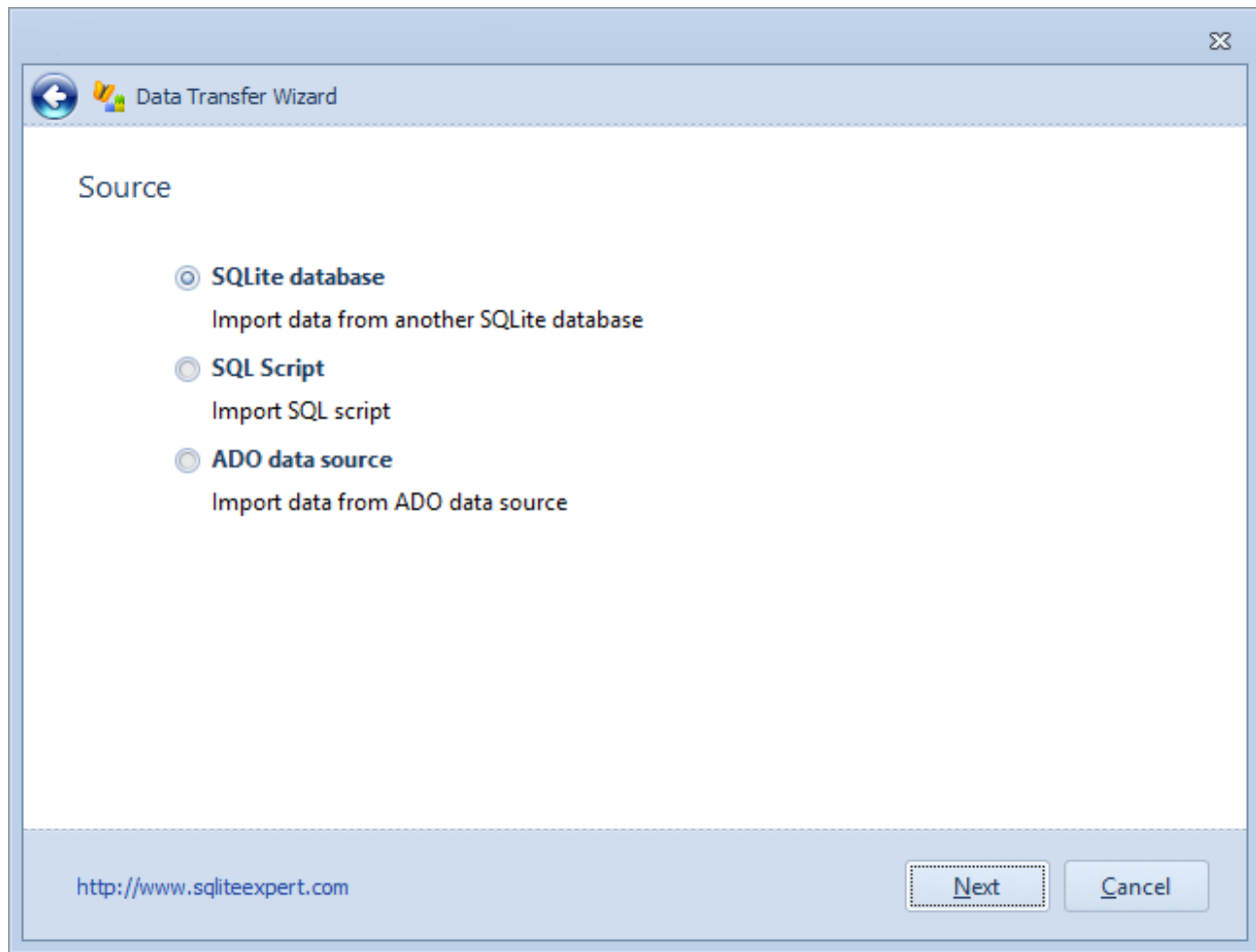
1. Select **Import/Export » Data Transfer Wizard** from the main menu. This pops up the Data Transfer Wizard.



2. Click the **Next** button.

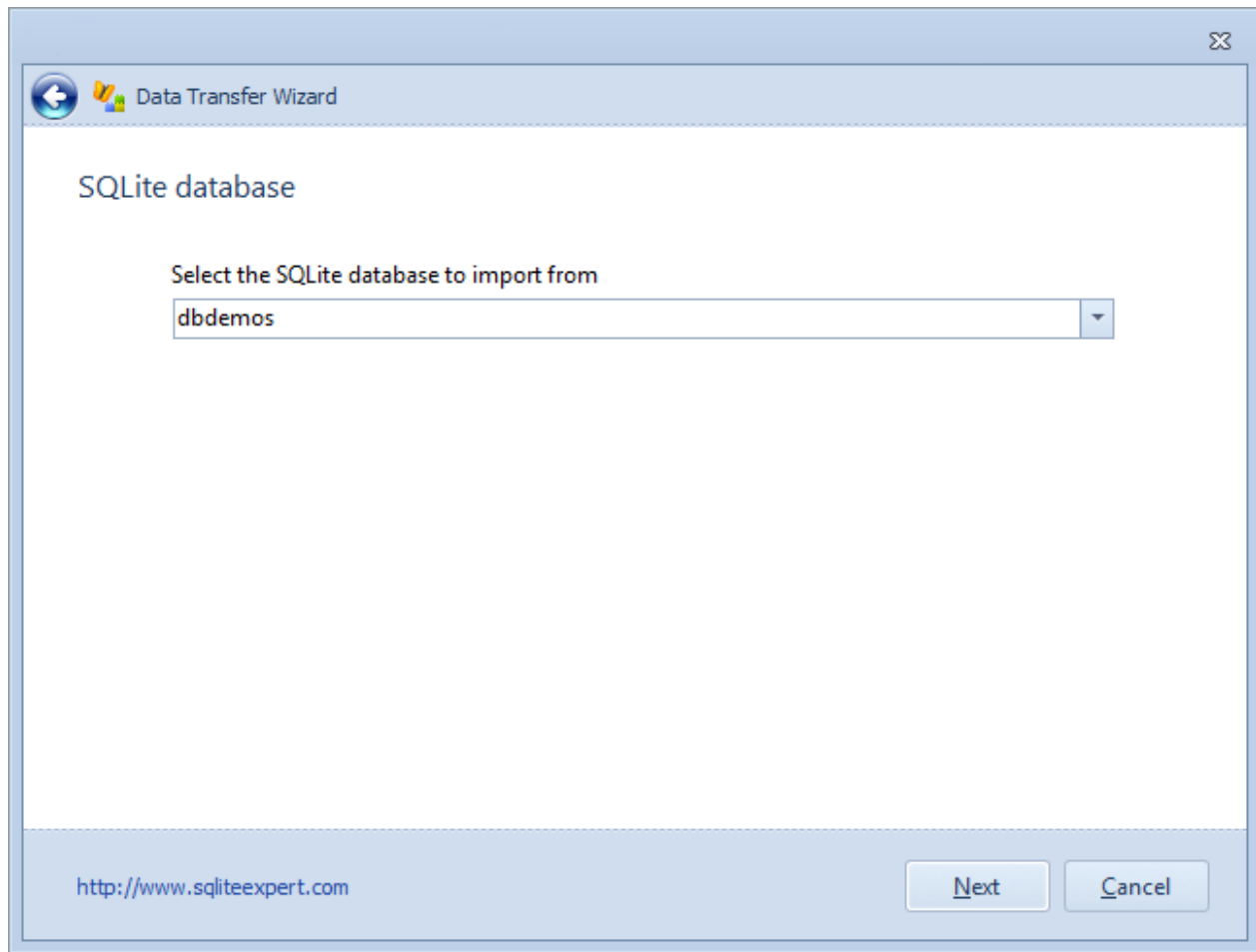


3. Select 'Import data into the selected database'.
4. Click the **Next** button.

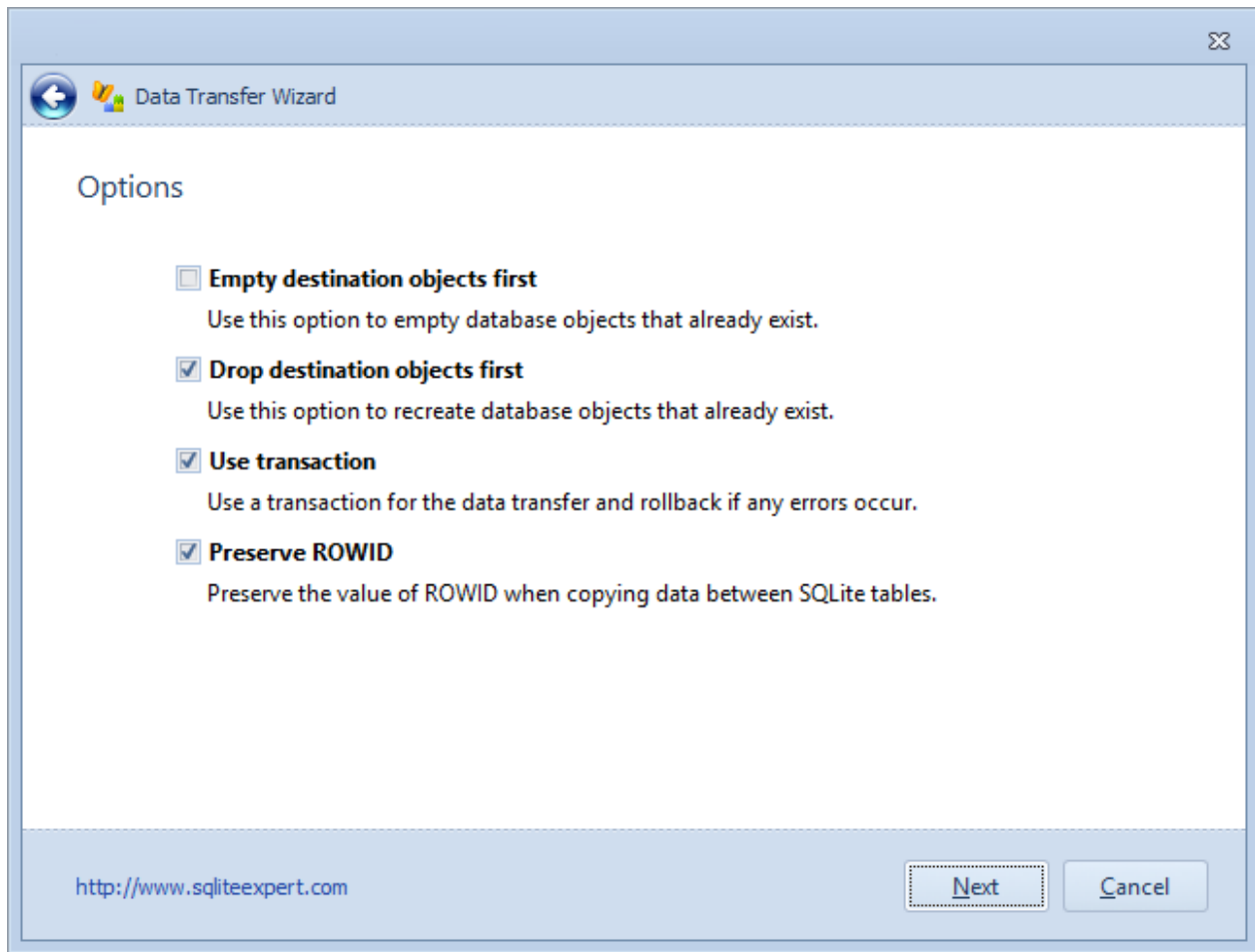


5. In the Source page, select 'SQLite database'.

6. Select the alias of the source database.

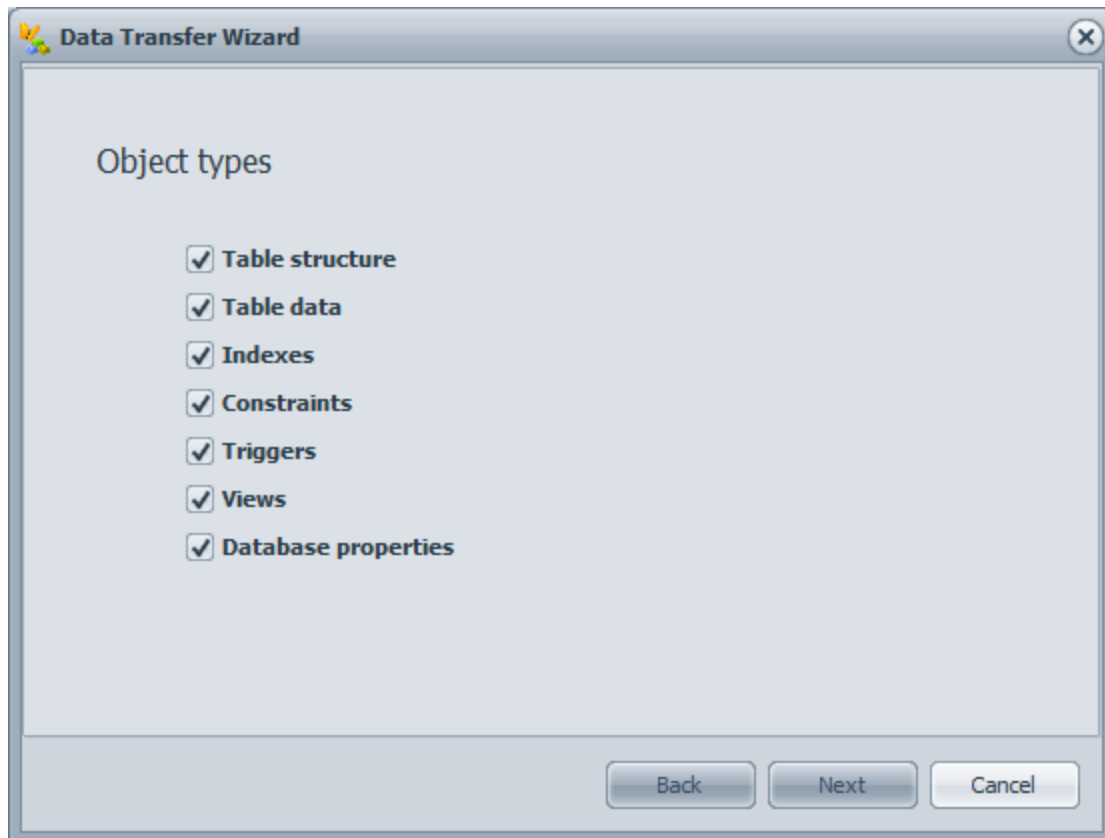


7. Click the **N**ext button.
8. On the next page, select import options:



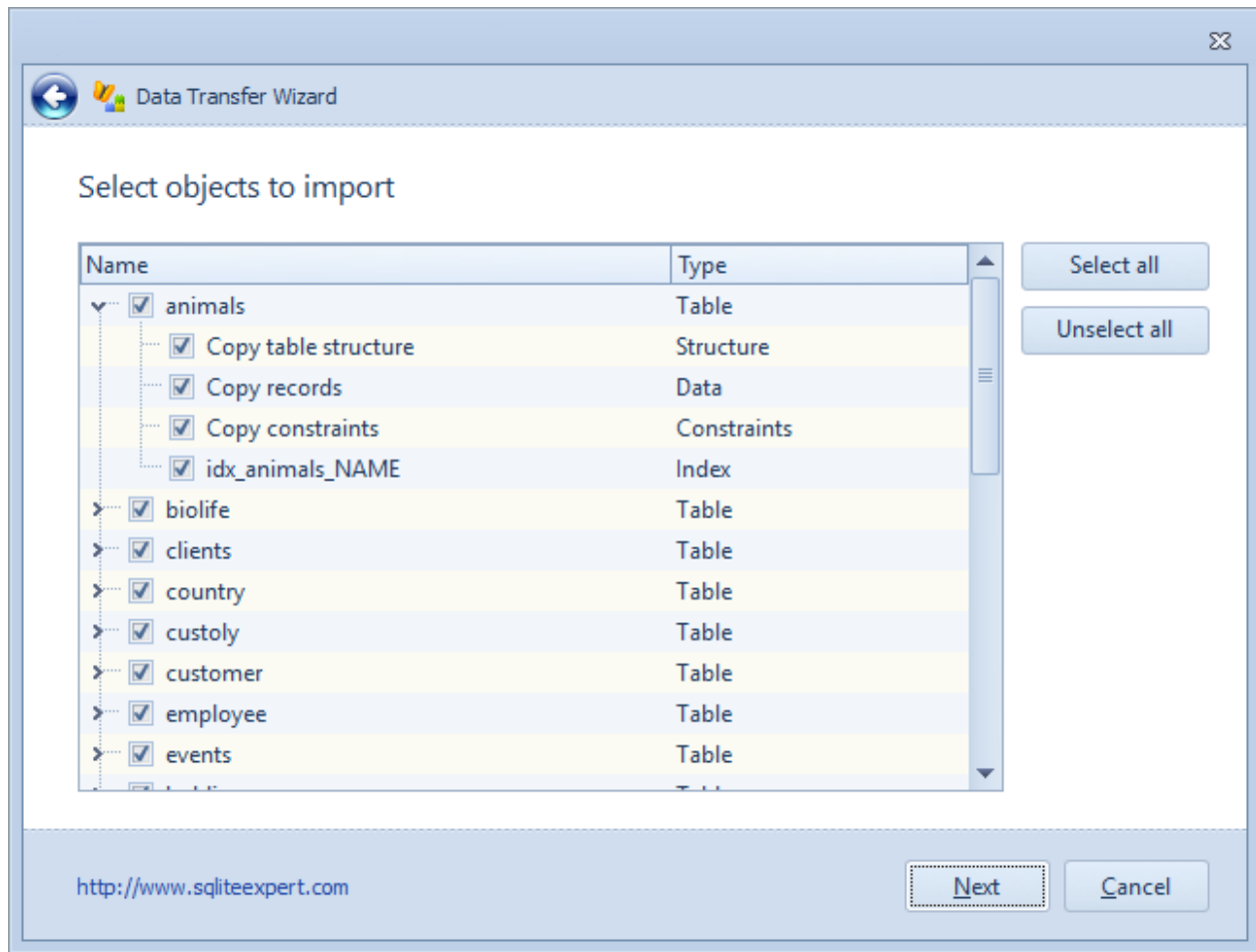
9. Click the **Next** button.

10. On the next page, select the types of objects that you want to import:



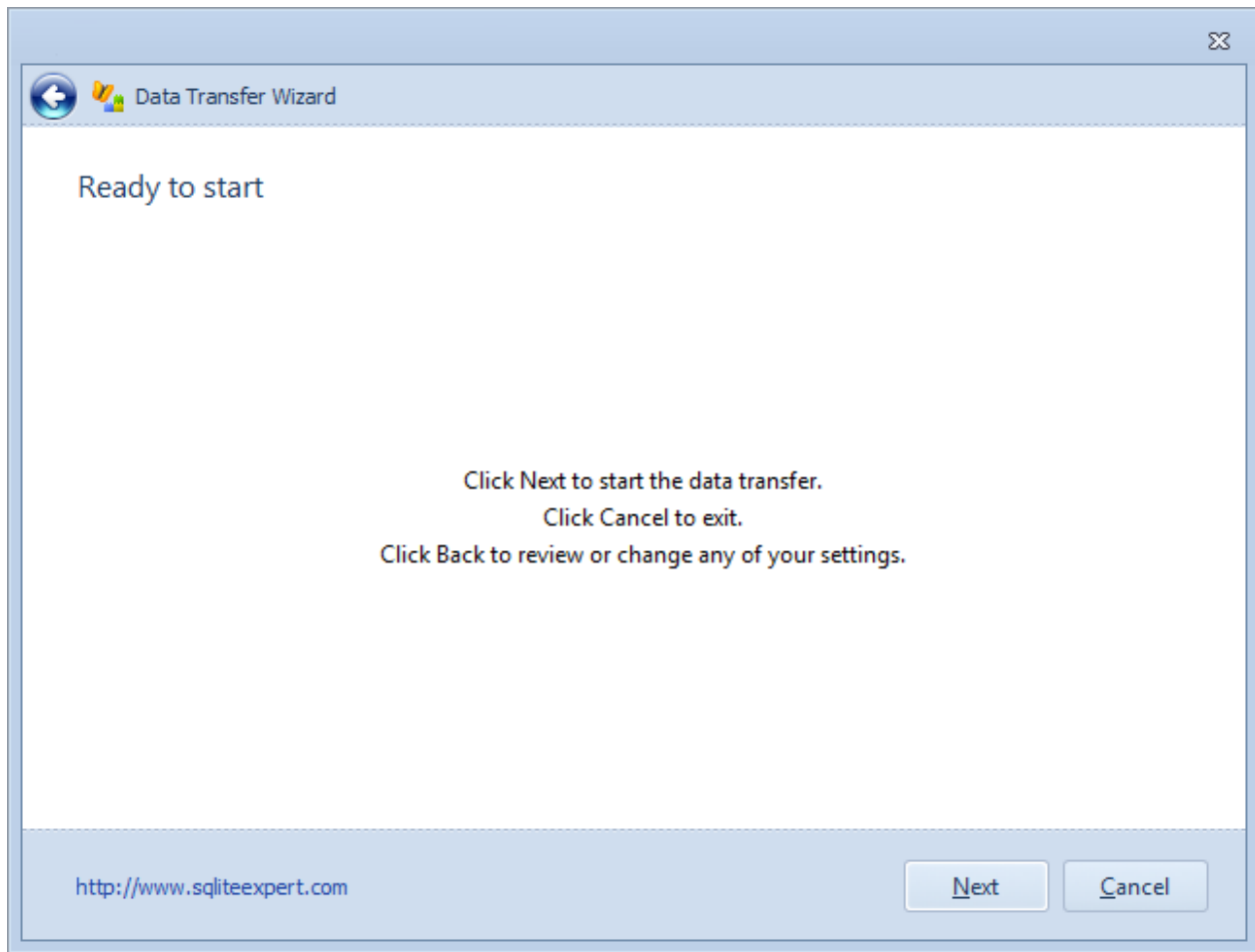
11. Click the **Next** button.

12. On the next page, select the individual objects that you want to import.



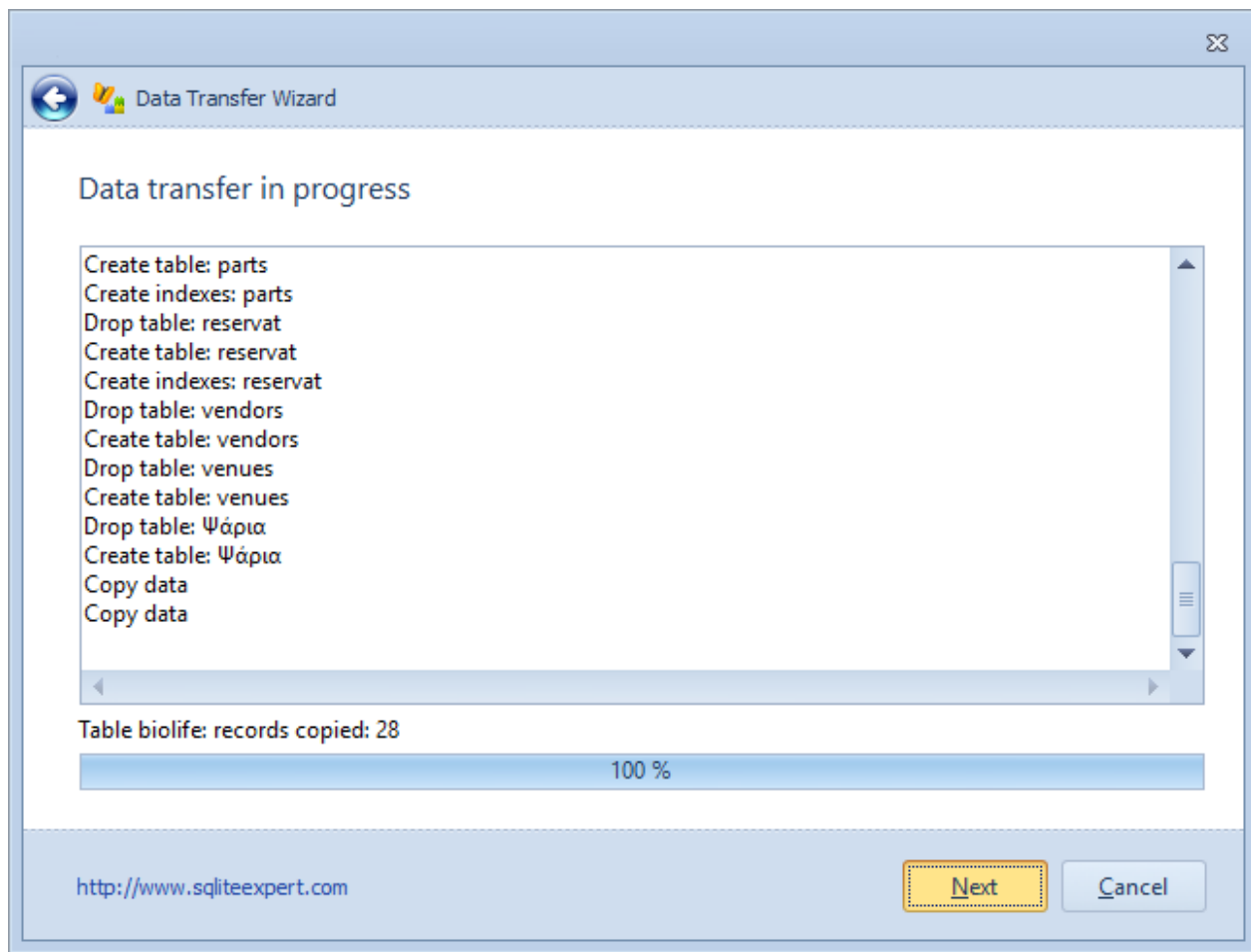
13. Click the **Next** button.

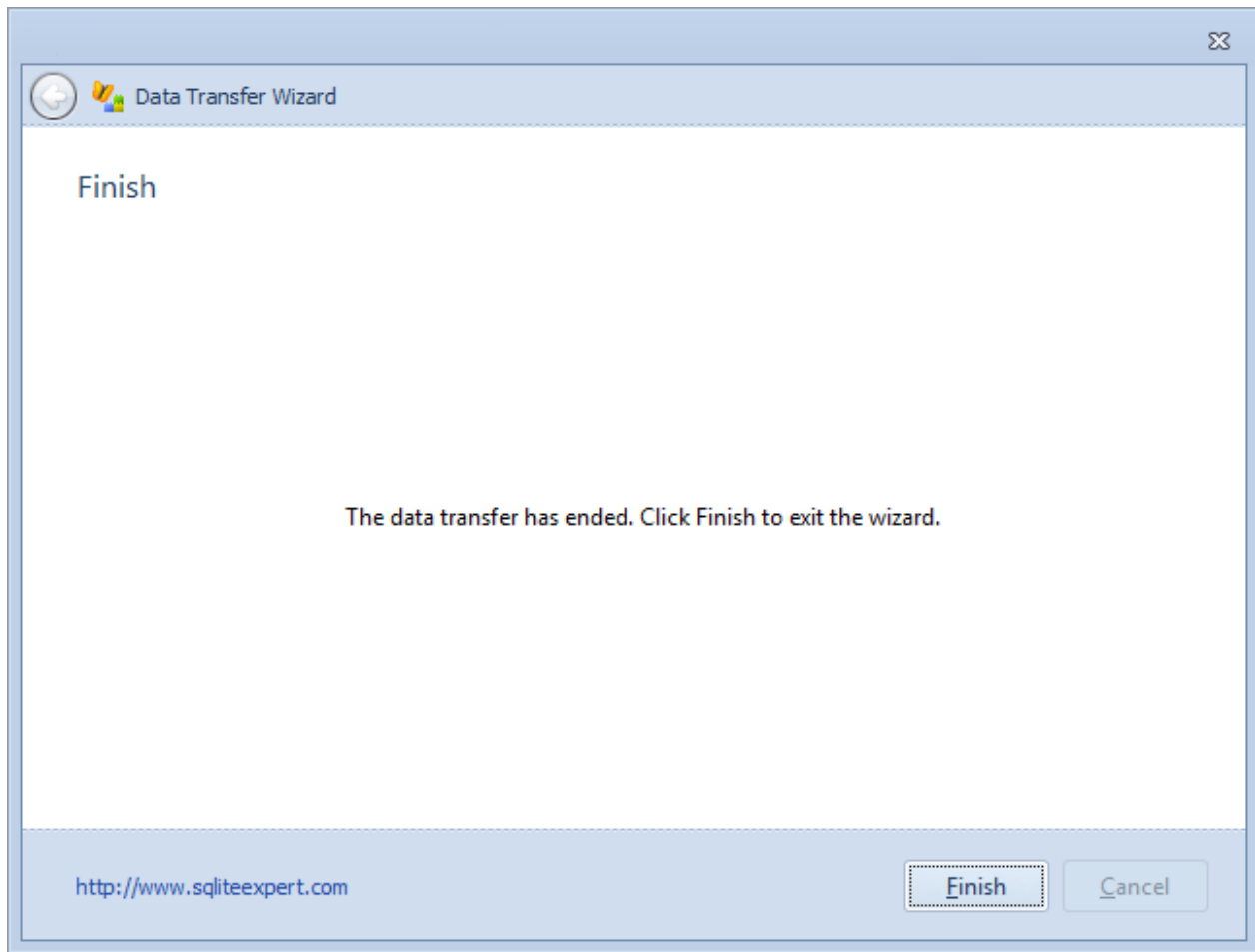
14. The next page displays the **Ready to start** page.



15. If you are satisfied with your choices, click **Next** to start the data transfer. Otherwise, click the **Back** button to change your choices.

16. During the data transfer, you have the option to cancel the operation by clicking the **Cancel** button.





17. Click **Finish** to exit the wizard.

Importing data from a script

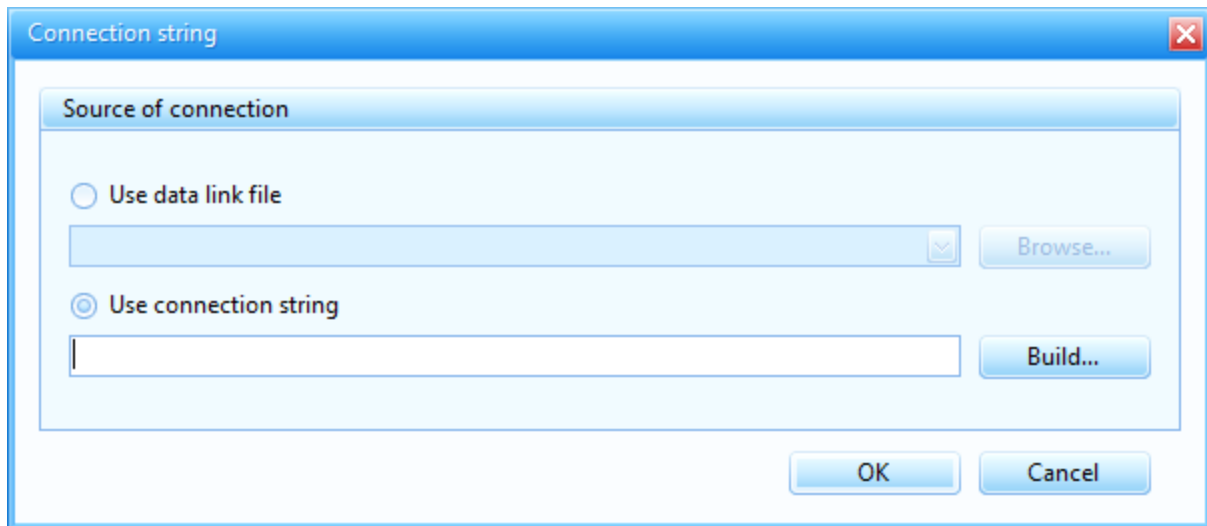
To import data into the selected database from an SQL script, follow the next steps:

1. Select **Import/Export » Data Transfer Wizard** from the main menu.
2. In the Action page, select 'Import data into the selected database'. Click Next.
3. In the Source page, select 'SQL Script'. Click Next.
4. Enter the file name for the script to import or use the file open dialog to select the script file. Click Next;
5. On the encoding page, select the encoding of the source file, if it is different than the detected encoding. Click Next.
6. On the Options page, select to use transaction. Click Next.
7. If you are satisfied with your choices, click **Next** to start the data transfer. Otherwise, click the **Back** button to change your choices.
8. Click **Finish** to exit the wizard.

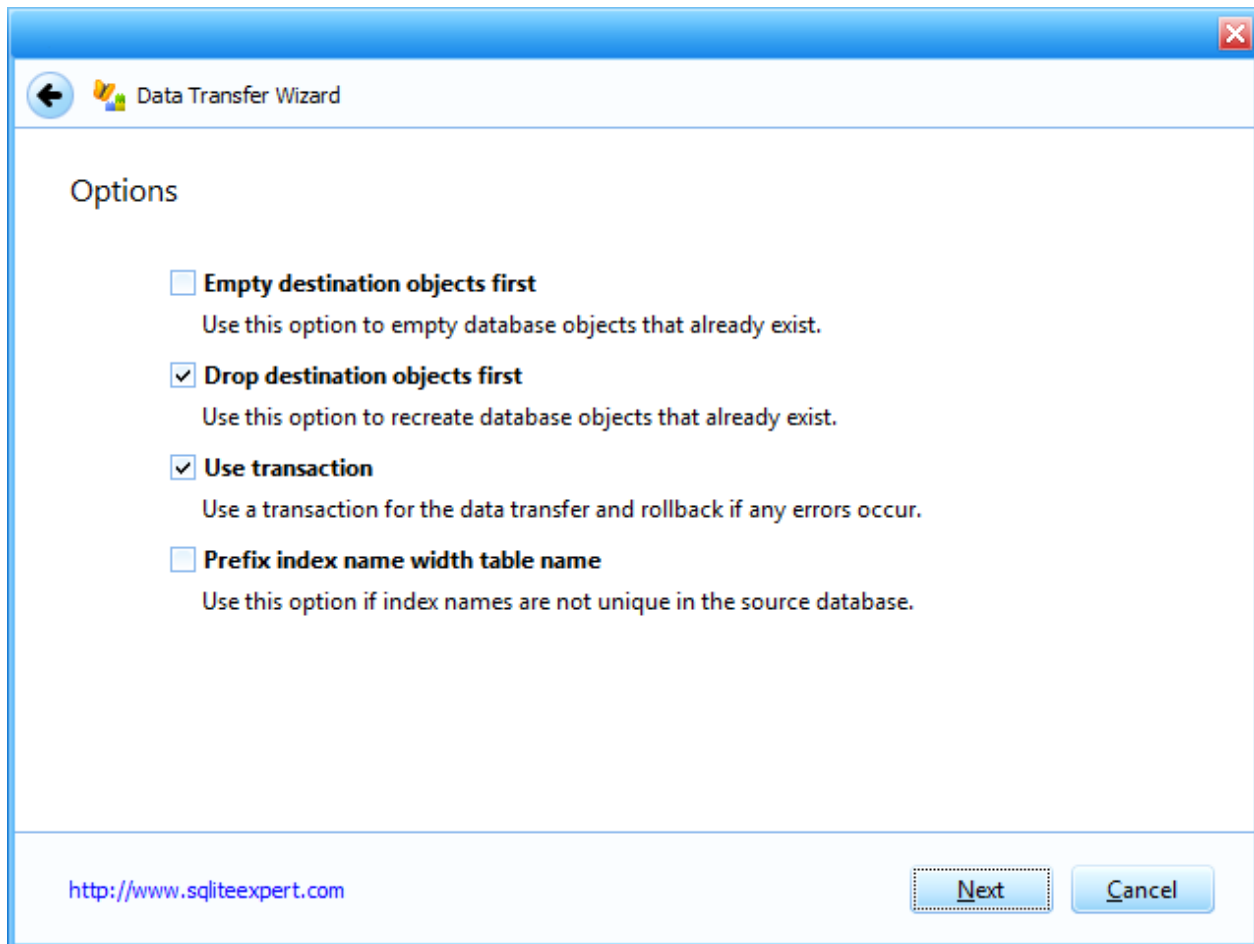
Importing data from an ADO data source

To import data into the selected database from an ADO data source, follow the next steps:

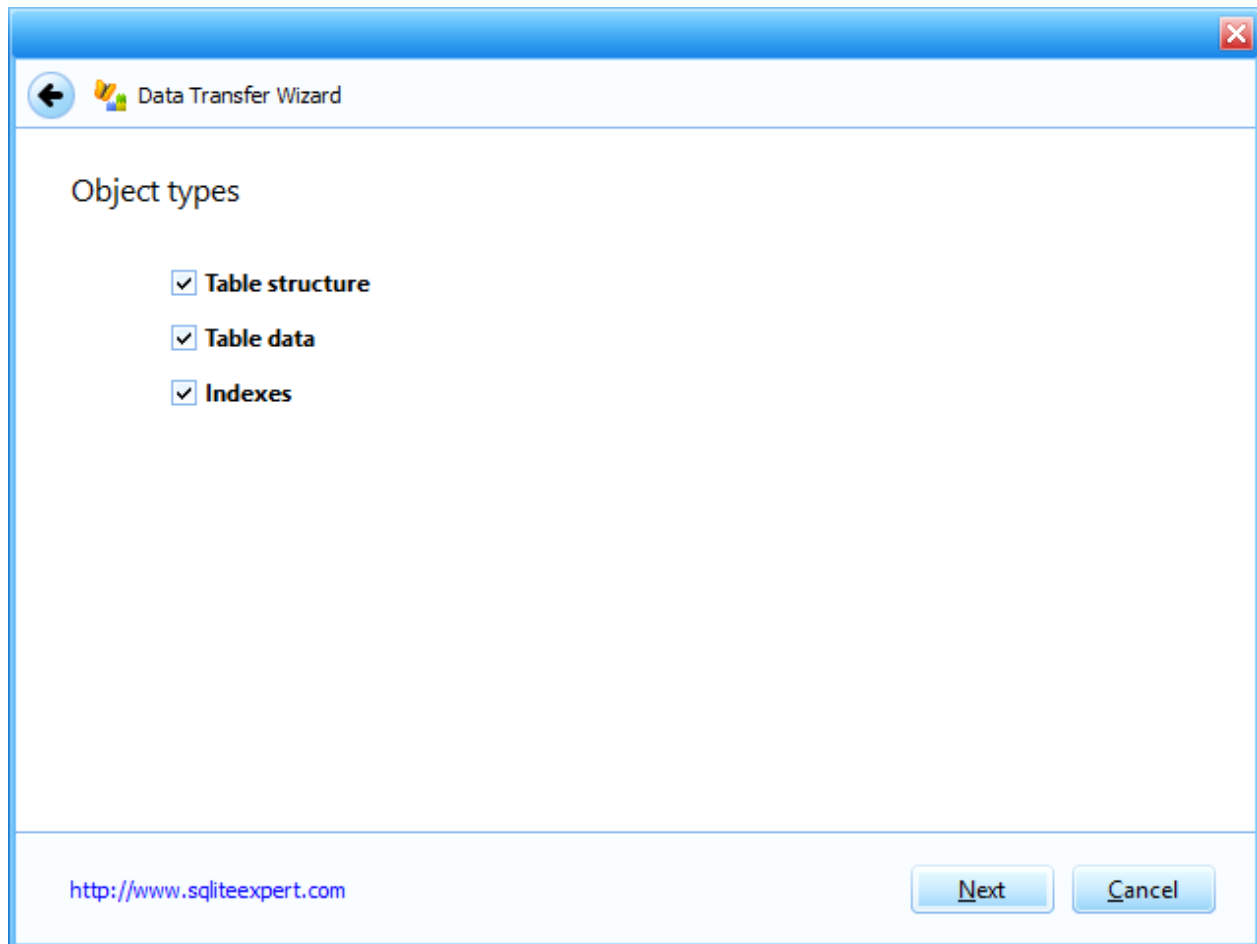
1. Select **Import/Export » Data Transfer Wizard** from the main menu.
2. In the Action page, select 'Import data into the selected database'. Click Next.
3. In the Source page, select 'ADO Data Source'. Click Next.
4. Click the '...' button to open a standard dialog for building the ADO connection string.



5. After building the ADO connection string, click the **Next** button.
6. Select additional options.

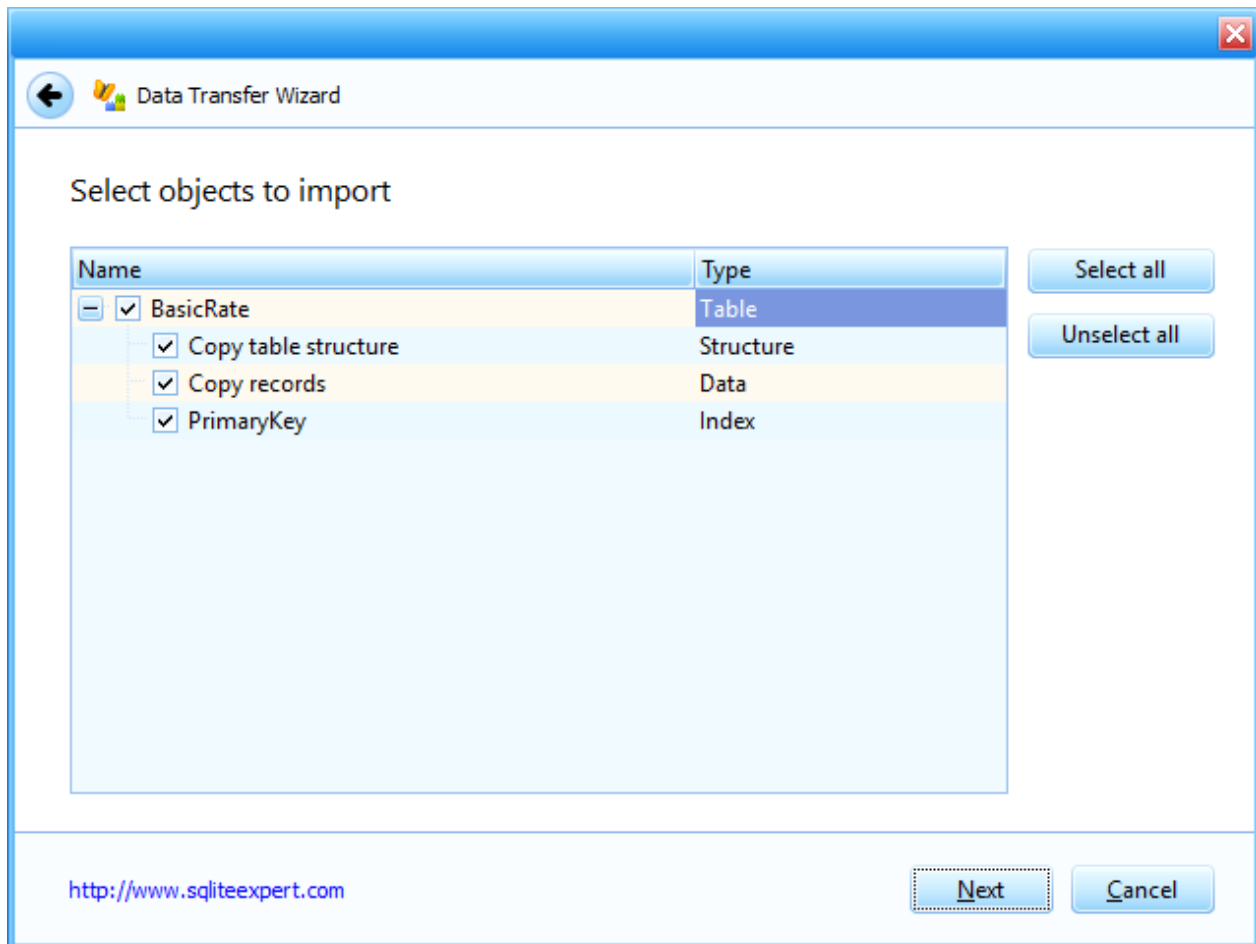


7. On the next page, select the types of objects that you want to import, then click Next.



8. Click the **Next** button.

9. On the next page, select the individual objects that you want to import.



10. Click the **Next** button.

11. If you are satisfied with your choices, click **Next** to start the data transfer. Otherwise, click the **Back** button to change your choices.

12. During the data transfer, you have the option to cancel the operation by clicking the **Cancel** button.

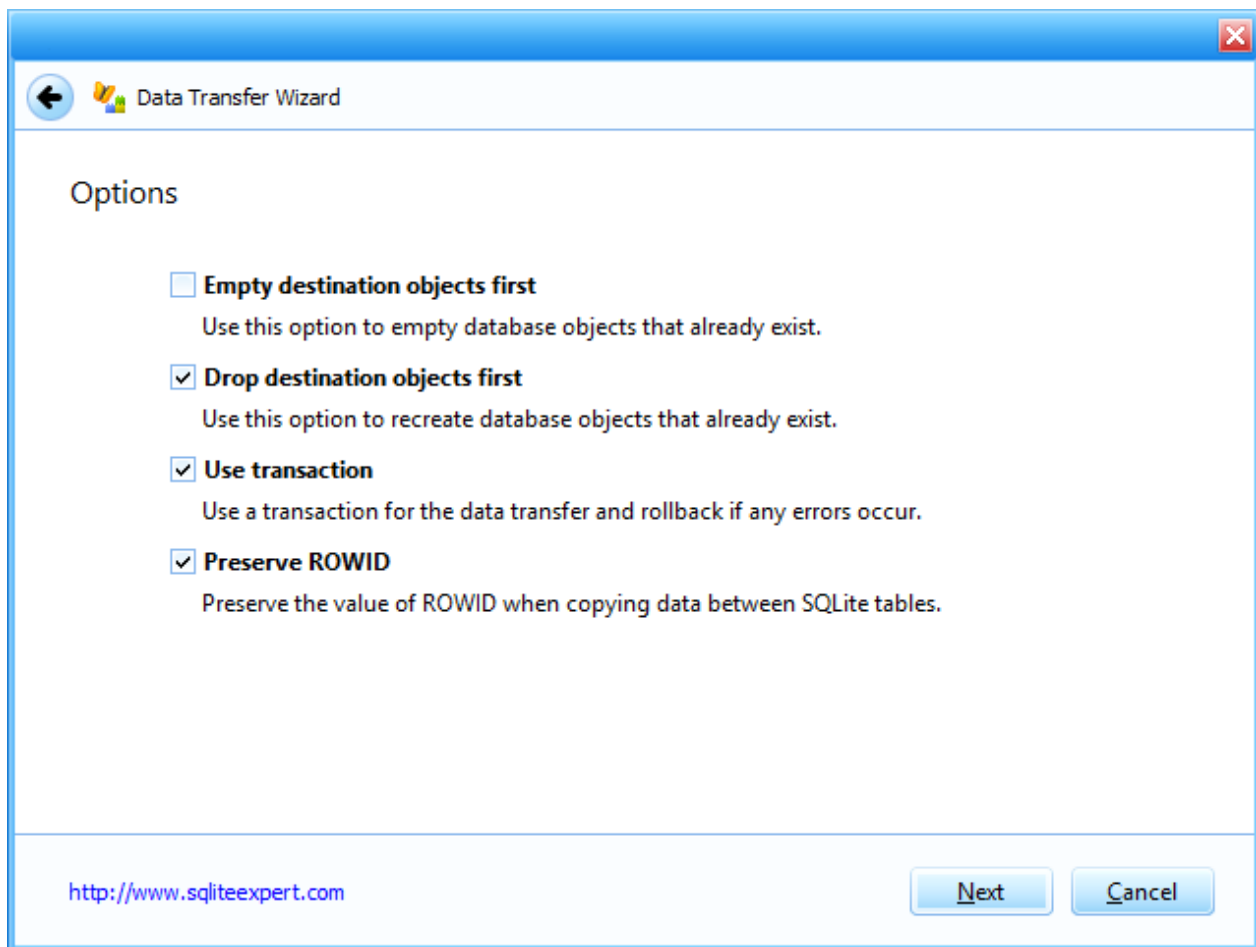
13. Click **Finish** to exit the wizard.

Exporting data to another SQLite database

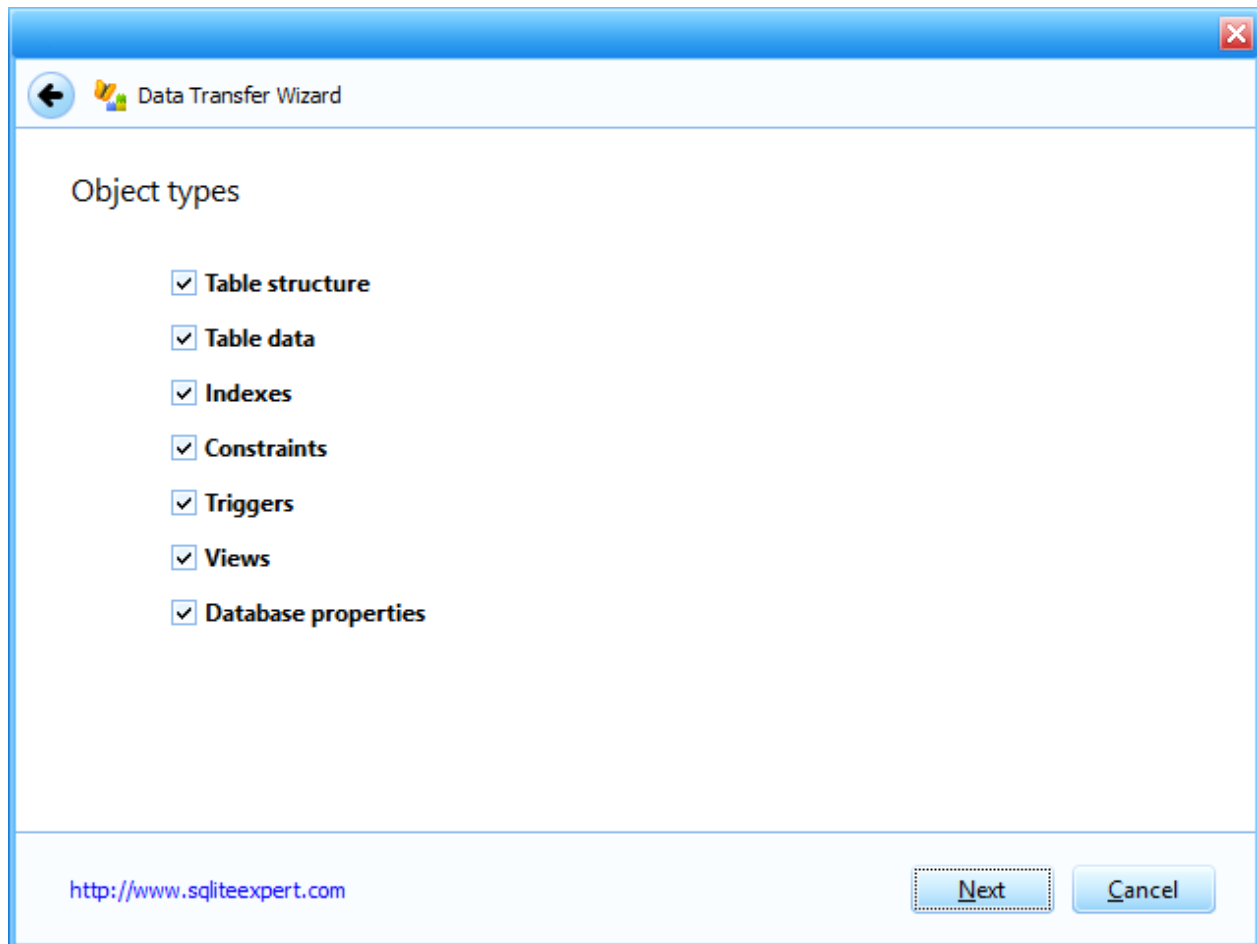
To export data from the selected database to another SQLite database, follow the next steps:

1. Select **Import/Export » Data Transfer Wizard** from the main menu.
2. In the Action page, select 'Export data from the selected database'.
3. In the Destination page, select 'Another SQLite database'.
4. Select the alias of the destination database.
5. Click the **Next** button.

6. Select additional options:

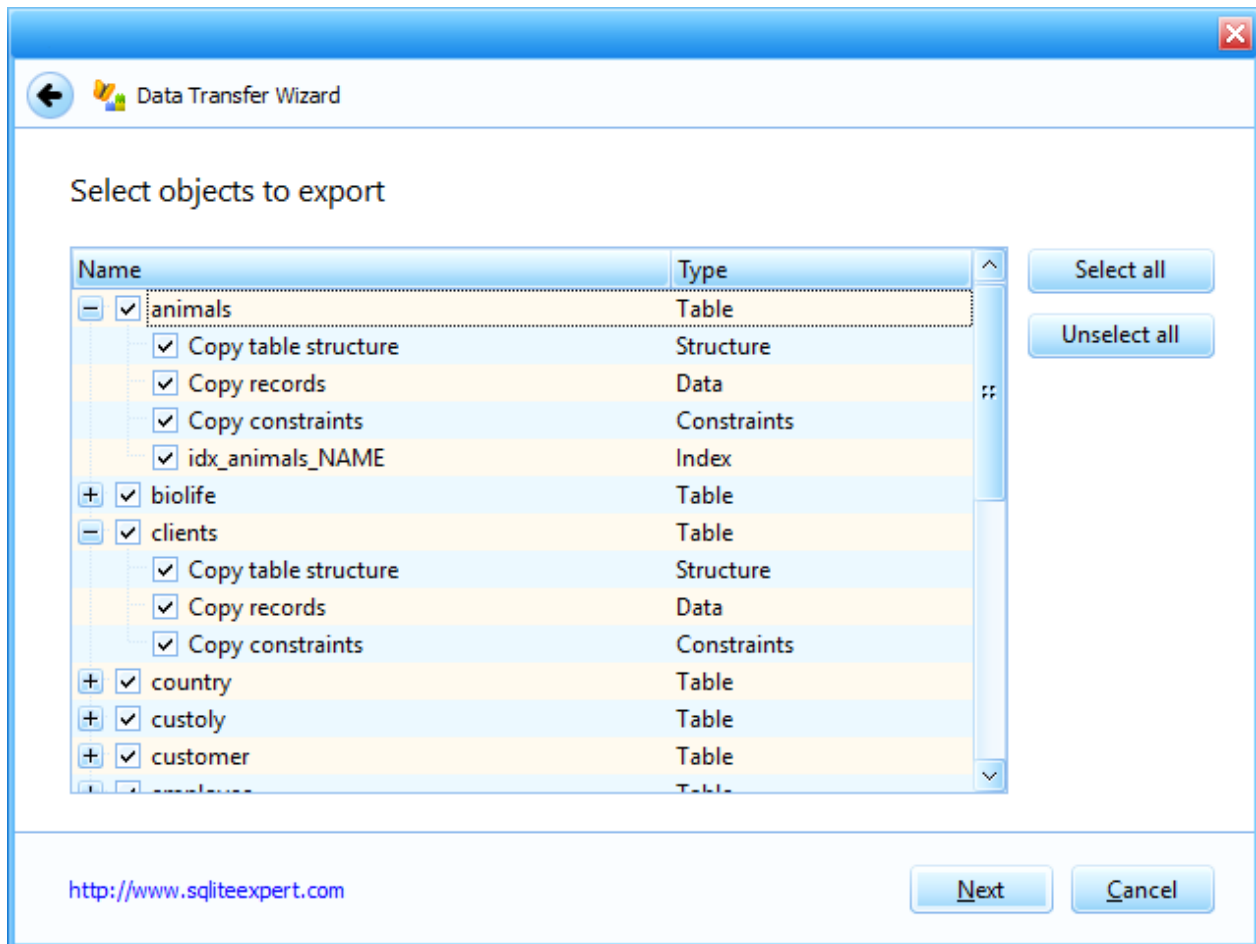


7. On the next page, select the types of objects that you want to export:



8. Click the **Next** button.

9. On the next page, select the individual objects that you want to export.



10. Click the **Next** button.

11. If you are satisfied with your choices, click **Next** to start the data transfer. Otherwise, click the **Back** button to change your choices.

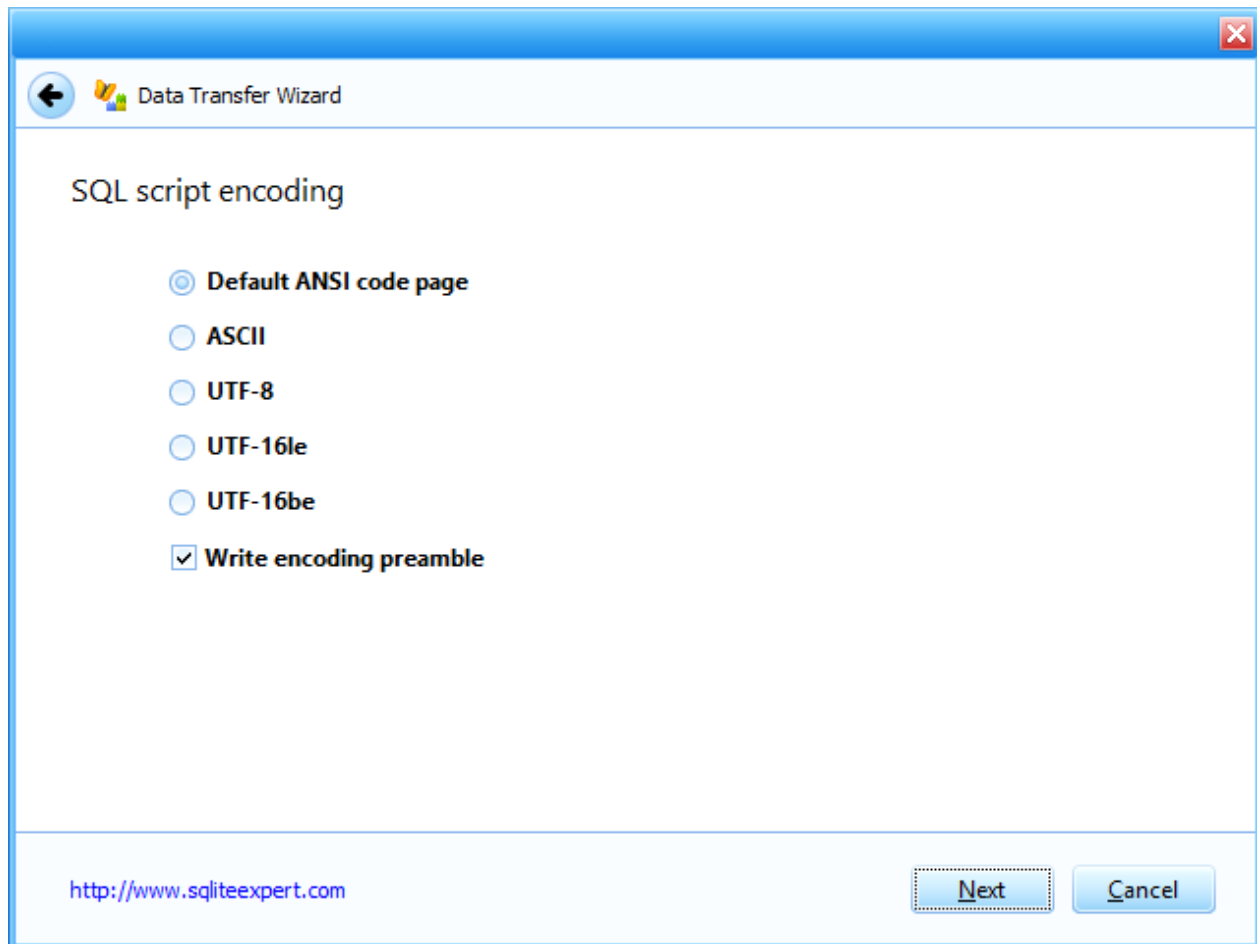
12. During the data transfer, you have the option to cancel the operation by clicking the **Cancel** button.

13. Click **Finish** to exit the wizard.

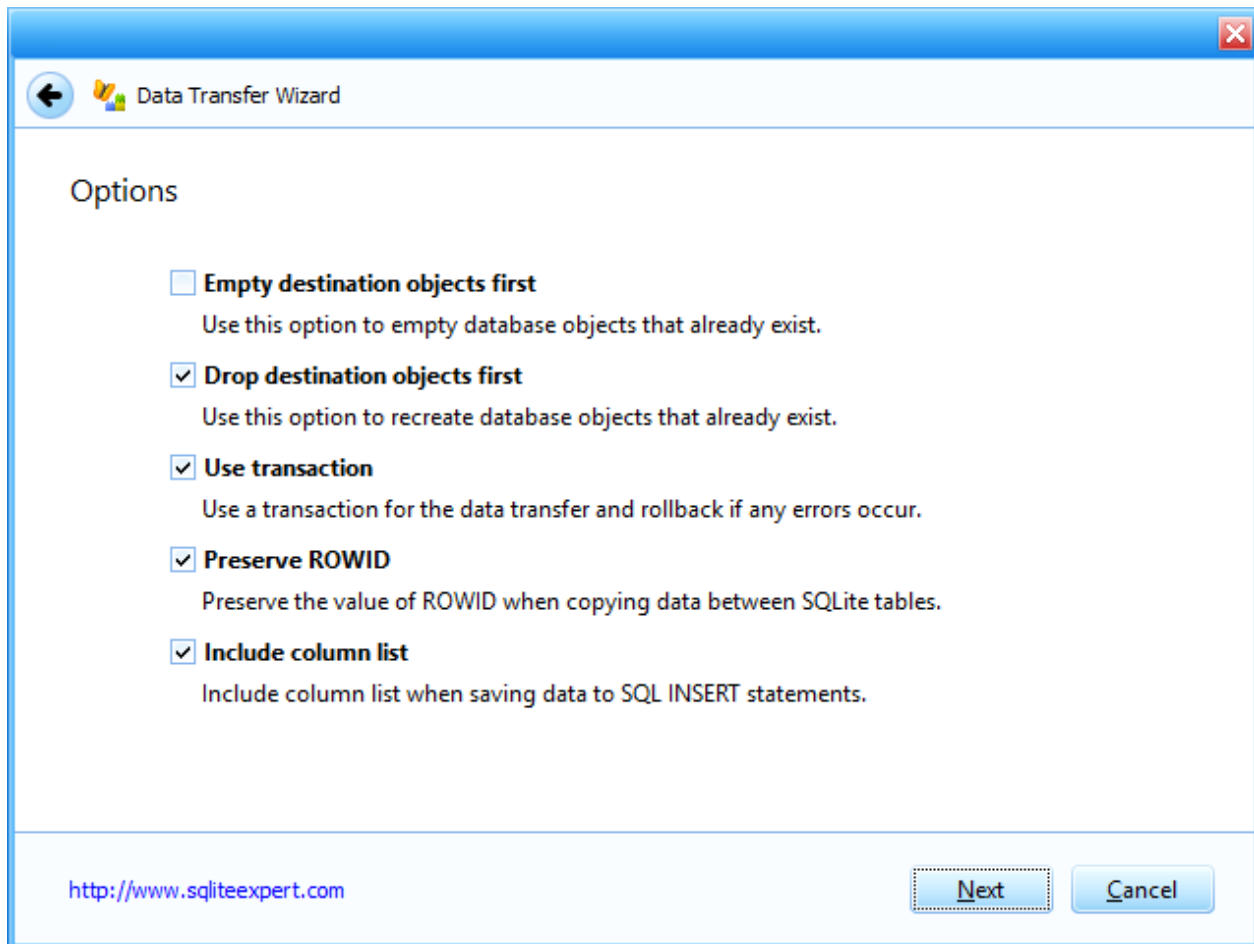
Exporting data to a script

To export data from the selected database to an SQL script, follow the next steps:

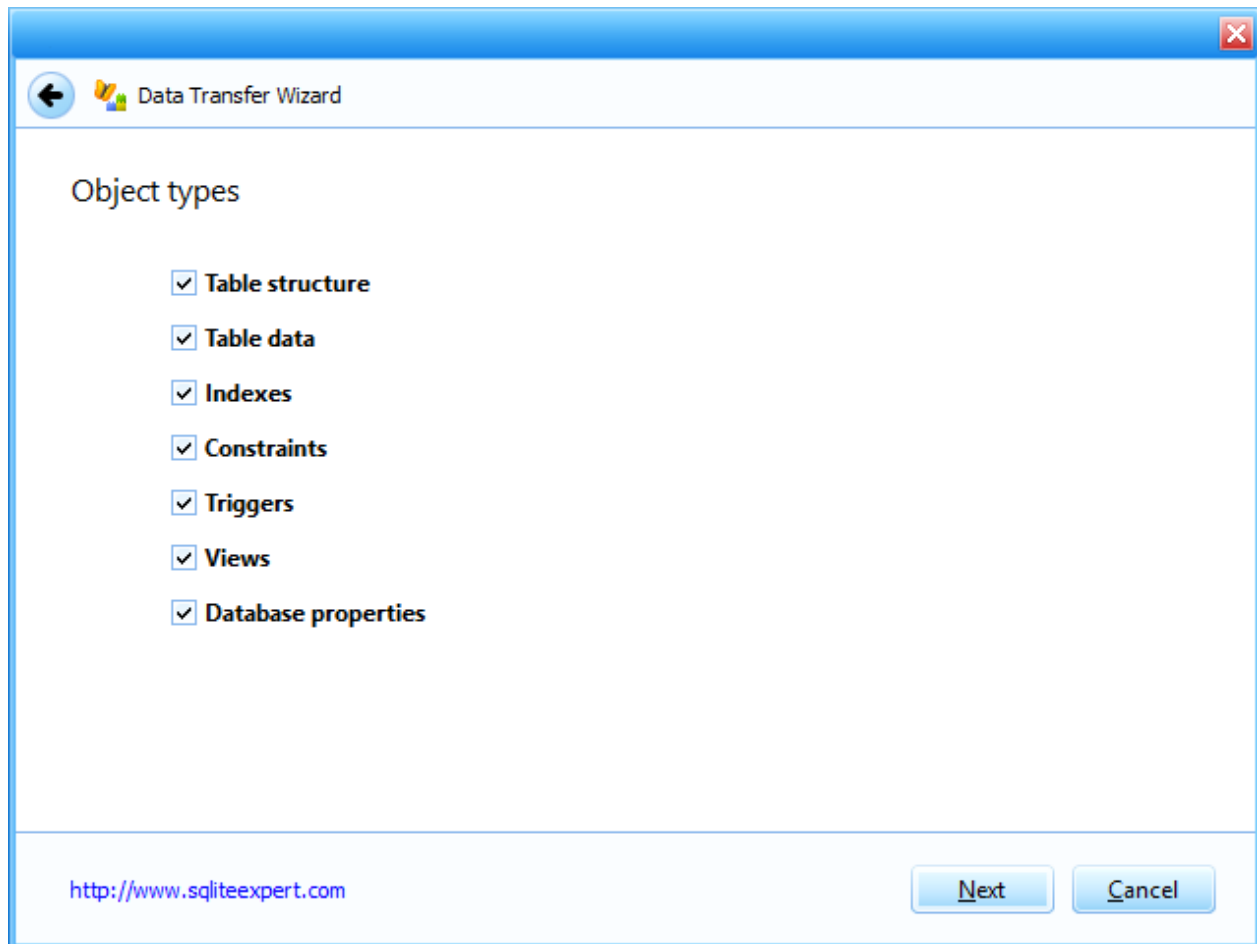
1. Select **Import/Export » Data Transfer Wizard** from the main menu.
2. In the Action page, select 'Export data from selected database'.
3. In the Destination page, select 'SQL Script'.
4. Select the file name for the destination script. Click Next.
5. Select the file encoding, then click Next:



6. Select additional options:

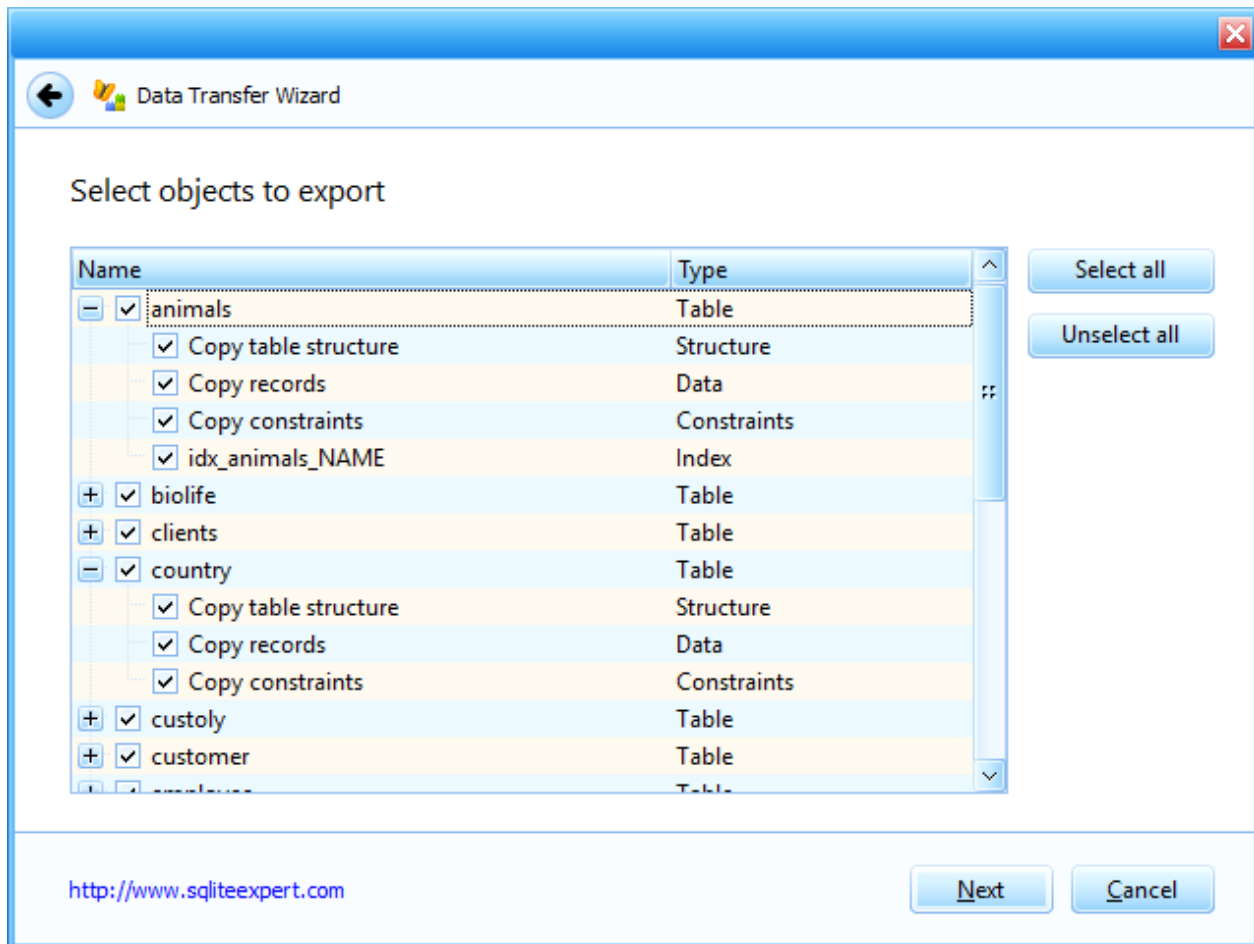


7. On the next page, select the types of objects that you want to export.



8. Click the **Next** button.

9. On the next page, select the individual objects that you want to export.



10. Click the **Next** button.

11. If you are satisfied with your choices, click **Next** to start the data transfer. Otherwise, click the **Back** button to change your choices.

12. During the data transfer, you have the option to cancel the operation by clicking the **Cancel** button.

13. Click **Finish** to exit the wizard.

Copying tables between databases using the clipboard

To copy one or more objects (tables or views) between databases using the clipboard, follow the next steps:

1. Select the source objects using the mouse or keyboard with **CTRL** or **SHIFT** operations, then press **CTRL-C** or select **Copy** from the tree popup menu.
2. Select the destination database, then press **CTRL-V** or select **Paste** from the tree popup menu.
3. The selected objects will be copied in the destination database, using a nested transaction. If any errors are encountered, the transaction is rolled back and no data will be copied.

Another way to copy tables or views between databases is by using drag and drop operations.

Using Transactions

Beginning in version 2.0, SQLite supports transactions with rollback and atomic commit. The optional transaction name is ignored. SQLite supports nested transactions starting with version 3.6.8.

- » [Starting a transaction](#)
- » [Committing a transaction](#)
- » [Rolling back a transaction](#)
- » [Creating a Savepoint](#)
- » [Releasing a Savepoint](#)
- » [Rolling back to a Savepoint](#)

Starting a transaction

To start a transaction in the selected database, select

Transaction » Start Transaction

from the main menu. Transactions can also be started manually by executing a **BEGIN [TRANSACTION]** command in an SQL script.

Committing a transaction

To commit a transaction in the selected database, select

Transaction » Commit

from the main menu. An active transaction can also be committed manually by executing a **COMMIT [TRANSACTION]** command in an SQL script.

Rolling back a transaction

To roll back a transaction for the selected database, select

Transaction » Rollback

from the main menu. An active transaction can also be rolled back manually by executing a **ROLLBACK [TRANSACTION]** command in an SQL script.

Creating a Savepoint

To create a savepoint in the selected database, select

Transaction » Savepoint

from the main menu. You will be prompted to enter the name of the new savepoint. A savepoint can also be created manually by executing a **SAVEPOINT** command in an SQL script.

Releasing a Savepoint

To release a savepoint in the selected database, select

Transaction » Release...

from the main menu and then select the name of the savepoint to be released. A savepoint can also be released manually by executing a **RELEASE [SAVEPOINT]** command in an SQL script.

Rolling back to a Savepoint

To roll back a transaction to an existing savepoint in the selected database, select

Transaction » Rollback to...

from the main menu, then select the name of the savepoint. This operation can also be performed manually by executing a **ROLLBACK [TRANSACTION] TO [SAVEPOINT]** command in an SQL script.

Extending SQLite Expert using Scripting

SQLite Expert provides an API that can be used to extend and enhance the application's functionality with scripts written in Lua or Pascal. Simple scripts can be written without any extra tools in the SQLite Expert user interface. The API allows scripts access to SQLite databases and implement functionality not duplicated in the formal Lua or Pascal API.

To select the scripting language: use the **Language** combo box on the Scripting tab.

To load a script: use the **Load script** command in the scripting popup menu.

To save a script: use the **Save script** command in the scripting popup menu.

To execute a script: use the **Execute** command in the scripting popup menu. Errors encountered while executing scripts are displayed in the Output tab.

» [Scripting examples](#)

» [Lua scripting API](#)

» [Pascal scripting API](#)

Scripting examples

Scripting example 1: display data

The following Lua script runs a query and displays the result on the Data tab:

```
database = sqlitedatabase:new()
database:filename("C:\\Program Files\\SQLite Expert\\Professional\\Data\\
dbdemos.db3")
database:connected(true)
query = sqlitequery:new()
query:database(database)
query:sql("select * from customer")
query:active(true)
displaydata(query)
query:free()
database:free()
```

The same script in Pascal:

```
var
  Database: TSQLiteDatabase;
  Query: TSQLiteQuery;
begin
  Database := TSQLiteDatabase.Create(nil);
  Database.FileName := 'C:\\Program Files\\SQLite
Expert\\Professional\\Data\\dbdemos.db3';
  Database.Connected := True;
  Query := TSQLiteQuery.Create(nil);
  Query.Database := Database;
  Query.SQL := 'select * from customer';
  Query.Open;
  DisplayData(Query);
  Query.Free;
  Database.Free;
end.
```

Scripting example 2: display all table names in a database

The following Lua script displays all the tables in the database dbdemos.db3:

```
database = sqlitedatabase:new()
database:filename("C:\\Program Files\\SQLite Expert\\Professional\\Data\\
dbdemos.db3")
database:open()
query = sqlitequery:new()
query:database(database)
query:sql("select * from sqlite_master where type = \"table\"")
query:open()
query:first()
while (query:eof() == false) do
  print(query:getfield("name"):value(), "\n")
  query:next()
```

```

end
query:free()
database:free()

```

The same example in Pascal:

```

var
    Database: TSQLiteDatabase;
    Query: TSQLiteQuery;
begin
    Database := TSQLiteDatabase.Create(nil);
    Database.FileName := 'C:\Program Files\SQLite
Expert\Professional\Data\dbdemos.db3';
    Database.Connected := True;
    Query := TSQLiteQuery.Create(nil);
    Query.Database := Database;
    Query.SQL := 'select * from sqlite_master where type = "table"';
    Query.Open;
    while not Query.Eof do
    begin
        Writeln(Query.FieldName('name').AsString);
        Query.Next;
    end;
    DisplayData(Query);
    Query.Free;
    Database.Free;
end.

```

Scripting example 3: update field

The following script iterates through the employee table and increases the Salary field by 10% to all employees:

```

database = sqlitedatabase:new()
database:filename("C:\\Program Files\\SQLite Expert\\Professional\\Data\\
dbdemos.db3")
database:connected(true)
query = sqlitequery:new()
query:database(database)
query:sql("select * from employee")
query:active(true)
query:first()
while (query:eof() == false) do
    query:edit()
    query:getfield('Salary'):asfloat(query:getfield('Salary'):asfloat() * 1.1)
    query:post()
    query:next()
end
displaydata(query)
query:free()
database:free()

```

The same example in Pascal:

```

var
    Database: TSQliteDatabase;
    Query: TSQliteQuery;
begin
    Database := TSQliteDatabase.Create(nil);
    Database.FileName := 'C:\Program Files\SQLite
Expert\Professional\Data\dbdemos.db3';
    Database.Connected := True;
    Query := TSQliteQuery.Create(nil);
    Query.Database := Database;
    Query.SQL := 'select * from employee';
    Query.Open;
    while not Query.Eof do
    begin
        Query.Edit;
        Query.FieldName('Salary').AsFloat := Query.FieldName('Salary').AsFloat *
1.1;
        Query.Post;
        Query.Next;
    end;
    DisplayData(Query);
    Query.Free;
    Database.Free;
end.

```

Lua scripting API

The sqlitedatabase class

sqlitedatabase:new()

Creates and returns a new instance of a sqlitedatabase object.

sqlitedatabase:filename([filename])

Sets or retrieves the value of the filename associated with the database object. The filename must point to an existing SQLite3 database, or :memory: for opening an in- memory database.

sqlitedatabase:connected([value])

When called with an argument, connects or disconnects the database to the filename set by sqlitedatabase:filename(). Returns true if connected to the database.

sqlitedatabase:open()

Connects to the database. Similar to sqlitedatabase:connected(true) except it does not return a value.

sqllitedatabase:close()

Similar to sqllitedatabase:connected(false) except it does not return a value.

sqllitedatabase:execute(sql)

Executes an SQL statement in the current database.

sqllitedatabase:beginTransaction()

Starts a transaction.

sqllitedatabase:commit()

Commits the current transaction.

sqllitedatabase:rollback()

Rolls back the current transaction.

sqllitedatabase:intransaction()

Returns true if a transaction is active, and false otherwise.

sqllitedatabase:free()

Destroys the object and frees its associated memory.

The sqlitequery class

sqlitequery represents a dataset with a result set that is based on an SQL statement.

sqlitequery:new()

Creates and returns a new instance of a sqlitequery object.

sqlitequery:database([sqllitedatabase])

When called with an argument, it binds the query with the `sqllitedatabase` object. Returns the associated `sqllitedatabase` object.

`sqlitequery:active([value])`

When called with a boolean argument: opens or closes the query depending on whether **value** is true or false. Returns true if active, false if not.

`sqlitequery:open()`

Opens the query by executing the SQL statement. Similar to **`sqlitequery:active(true)`**. If the SQL statement returns a cursor the following functions can be used to iterate through the result set: **`first`**, **`last`**, **`next`**, **`prior`**, **`eof`**, **`bof`**.

`sqlitequery:close()`

Closes the query.

`sqlitequery:eof()`

Test eof (end of file) to determine if the dataset is positioned at the last record.

`sqlitequery:bof()`

Test bof (beginning of file) to determine if the dataset is positioned at the first record.

`sqlitequery:first()`

Moves to the first record in the dataset.

`sqlitequery:last()`

Moves to the last record in the dataset.

`sqlitequery:next()`

Moves to the next record in the dataset.

sqlitequery:prior()

Moves to the previous record in the dataset.

sqlitequery:edit()

Enables editing of data in the dataset.

sqlitequery:post()

Writes the modified record to the database.

sqlitequery:insert()

Inserts a new, empty record in the dataset.

sqlitequery:delete()

Deletes the active record and positions the dataset on the next record.

sqlitequery:cancel()

Cancels modifications to the active record if those changes are not yet posted.

sqlitequery:recordcount()

Indicates the total number of records associated with the dataset. Note: the value returned by this function is only valid after reaching the last record in the dataset.

sqlitequery:recno()

Indicates the active record in the dataset.

sqlitequery:fieldcount()

Indicates the number of field components associated with the dataset.

sqlitequery:getfield([value])

Finds a field based on its name (if **value** is a string) or its index (if **value** is an integer number). Returns a **sqlitefield** object, or **null** if the field was not found.

sqlitequery:sql([value])

Sets or retrieves the text of the SQL statement to execute for the query.

sqlitequery:free()

Destroys the object and frees its associated memory.

The sqlitefield class**sqlitefield:name()**

Returns the name of the field.

sqlitefield:value([value])

Sets or retrieves the value of the field in the current record, as string. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

sqlitefield:asstring([value])

Same as **sqlitefield:value([value])** : sets or retrieves the value of the field in the current record, as string. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

sqlitefield:asfloat([value])

Sets or retrieves the value of the field in the current record, as a floating point number. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

sqlitefield:asboolean([value])

Sets or retrieves the value of the field in the current record, as a boolean value. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

sqlitefield:fullsqltype()

Returns the full declared type of the field including sqltype, size and precision.

sqlitefield:sqltype()

Returns the field declared type.

sqlitefield:size()

Returns the field size.

sqlitefield:precision()

Returns the field precision.

Pascal scripting API**The TSQLiteDatabase class****constructor Create(AOwner: TComponent)**

Creates and returns a new instance of a TSQLiteDatabase object.

property FileName: string;

Sets or retrieves the value of the filename associated with the database object. The filename must point to an existing SQLite3 database, or :memory: for opening an in- memory database.

property Connected: Boolean;

When called with an argument, connects or disconnects the database to the filename set by the FileName property. Returns True if connected to the database.

procedure Open;

Connects to the database. Similar to calling Connected(True) except it does not return a value.

procedure Close;

Disconnects from database. Similar to calling `Connected(False)` except it does not return a value.

procedure Execute(const SQL: string);

Executes an SQL statement in the current database.

procedure BeginTransaction;

Starts a transaction.

procedure Commit;

Commits the current transaction.

procedure Rollback;

Rolls back the current transaction.

property InTransaction: Boolean;

Returns True if a transaction is active, and False otherwise.

procedure Free;

Destroys the object and frees its associated memory.

The TSQLiteQuery class

TSQLiteQuery represents a dataset with a result set that is based on an SQL statement.

constructor Create(AOwner: TComponent);

Creates and returns a new instance of a sqlitequery object.

property Database: TSQLiteDatabase;

Specifies the database object associated with the query.

property Active: Boolean;

Specifies whether or not the dataset is open.

procedure Open;

Opens the query by executing the SQL statement. Similar to calling

Active := True.

If the SQL statement returns a cursor the following functions and properties can be used to iterate through the result set: **First, Last, Next, Prior, Eof, Bof.**

procedure Close;

Closes the query.

property Eof: Boolean;

Test eof (end of file) to determine if the dataset is positioned at the last record.

property Bof: Boolean;

Test bof (beginning of file) to determine if the dataset is positioned at the first record.

procedure First;

Moves to the first record in the dataset.

procedure Last;

Moves to the last record in the dataset.

procedure Next;

Moves to the next record in the dataset.

procedure Prior;

Moves to the previous record in the dataset.

procedure Edit;

Enables editing of data in the dataset.

procedure Post;

Writes the modified record to the database.

procedure Insert;

Inserts a new, empty record in the dataset.

procedure Delete;

Deletes the active record and positions the dataset on the next record.

procedure Cancel;

Cancels modifications to the active record if those changes are not yet posted.

property RecordCount: Integer;

Indicates the total number of records associated with the dataset. Note: the value returned by this property is only valid after reaching the last record in the dataset.

property RecNo: Integer;

Indicates the active record in the dataset.

property FieldCount: Integer;

Indicates the number of field components associated with the dataset.

```
function FieldByName(const Value: string): TField;
```

Finds a field based on its name. Returns a **TField** object, or **nil** if the field was not found.

```
function Fields(const Index: Integer): TField;
```

Finds a field based on its index. Returns a **TField** object. An 'index out of range' error occurs if the Index is < 0 or >= FieldCount.

```
property SQL: string;
```

Sets or retrieves the text of the SQL statement to execute for the query.

```
procedure Free;
```

Destroys the object and frees its associated memory.

The TField class

```
property FieldName: string;
```

Returns the name of the field.

```
property Value: Variant;
```

Sets or retrieves the value of the field in the current record, as a variant. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

```
property AsString: string;
```

Sets or retrieves the value of the field in the current record, as string. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

```
property AsFloat: Double;
```

Sets or retrieves the value of the field in the current record, as a floating point number. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

property AsBoolean: Boolean;

Sets or retrieves the value of the field in the current record, as a boolean value. To set the value of the field, the dataset must be in the Edit mode, set by a previous call to Insert or Edit.

property Size: Integer;

Indicates the size used in the definition of the physical database field for data types that support different sizes.

Data types

SQLite3 allows the user to declare any data type, but each value stored in an SQLite database (or manipulated by the database engine) has one of the following storage classes:

⟨ **NULL**. The value is a NULL value.

⟨ **INTEGER**. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.

⟨ **REAL**. The value is a floating point value, stored as an 8-byte IEEE floating point number.

⟨ **TEXT**. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16-LE).

⟨ **BLOB**. The value is a blob of data, stored exactly as it was input.

SQLite Expert supports the following predefined data types:

Declared type	Mapped to internal data type	Description
DEC	Float	Floating-point numeric field (8 bytes)
DECIMAL	Float	Floating-point numeric field (8 bytes)
DOUBLE	Float	Floating-point numeric field (8 bytes)
DOUBLE PRECISION	Float	Floating-point numeric field (8 bytes)
FLOAT	Float	Floating-point numeric field (8 bytes)
NUMERIC	Float	Floating-point numeric field (8 bytes)
REAL	Float	Floating-point numeric field (8 bytes)
INT	Largeint	64-bit integer field
INTEGER	Largeint	64-bit integer field
SMALLINT	Largeint	64-bit integer field

TINYINT	Largeint	64-bit integer field
WORD	Largeint	64-bit integer field
AUTOINC	Largeint	64-bit integer field
BIGINT	Largeint	64-bit integer field
LARGEINT	Largeint	64-bit integer field
INT64	Largeint	64-bit integer field
CHAR	WideString	Wide string field
VARCHAR	WideString	Wide string field
VARCHAR2	WideString	Wide string field
TEXT	WideString	Wide string field
DATETEXT	WideString	Wide string field
NCHAR	WideString	Wide string field
NVARCHAR	WideString	Wide string field
NVARCHAR2	WideString	Wide string field
NTEXT	WideString	Wide string field
CURRENCY	Currency	Money field (8 bytes)
MONEY	Currency	Money field (8 bytes)
SMALLMONEY	Currency	Money field (8 bytes)
BLOB	Blob	Binary Large Object field
RAW	Blob	Binary Large Object field
BINARY	Blob	Binary Large Object field
VARBINARY	Blob	Binary Large Object field
CLOB	WideMemo	Text wide memo field
MEMO	WideMemo	Text wide memo field
DATE	Date	Date field (Borland TDateTime)
TIME	Time	Time field (Borland TDateTime)
DATETIME	DateTime	Date and time field (Borland TDateTime)

TIMESTAMP	DateTime	Date and time field (Borland TDateTime)
BOOLEAN	Boolean	Boolean field
GRAPHIC	Graphic	Graphic field

Note: The integral part of a Date, Time or DateTime value is the number of days that have passed since 12/30/1899. The fractional part of the Date, Time or DateTime value is fraction of a 24 hour day that has elapsed. SQLite Expert will store the Date, Time and DateTime fields in the database in the format 'yyyy-mm-dd hh-mm-ss.sss' but will recognize the date and time fields stored in the Borland DateTime format.

In addition to the predefined data types, SQLite Expert allows the user to declare custom data types. That is, you can add your own declared types but they have to map internally to one of the known data types shown in the above table. To change the internal data mapping please see [Changing options](#).

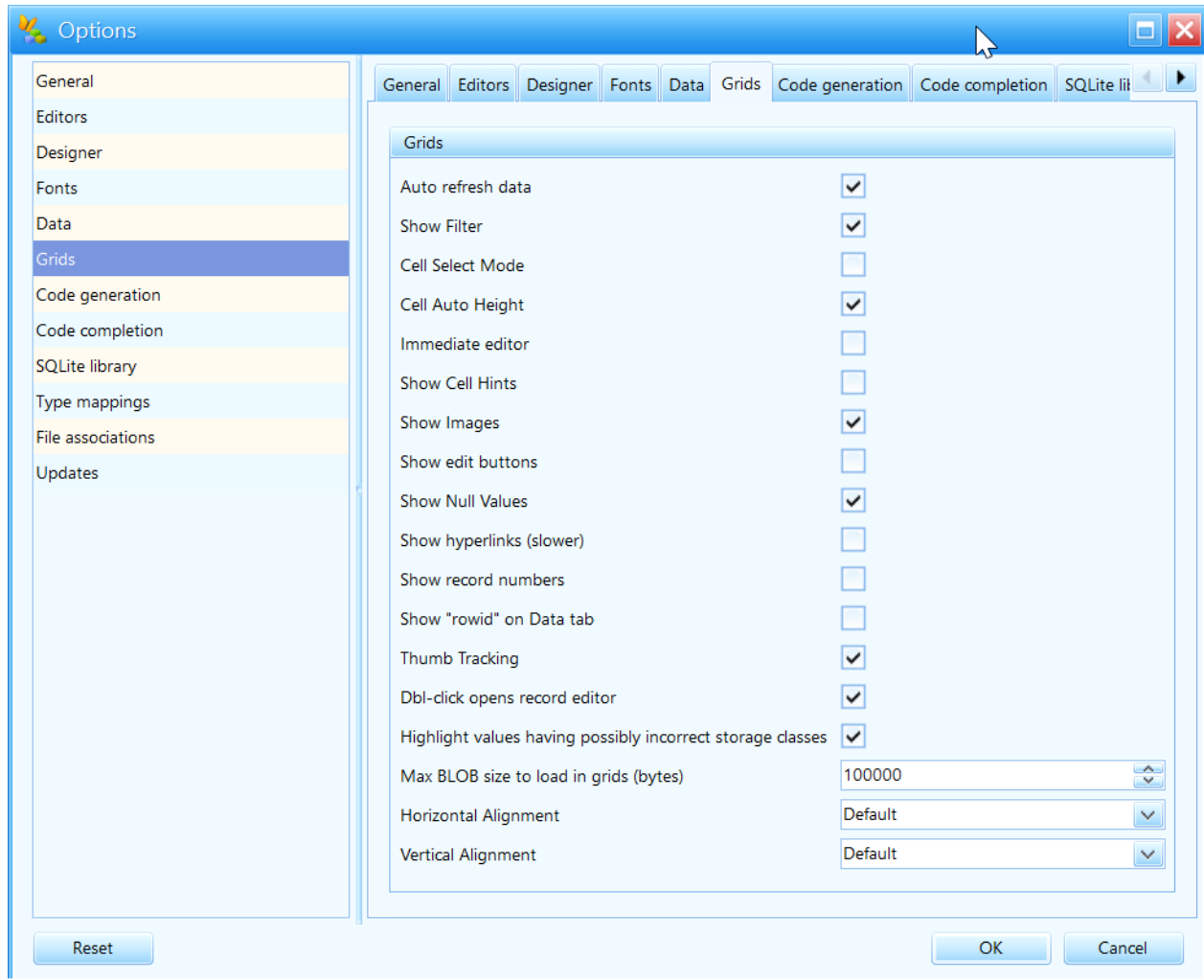
For more information about SQLite3 data types, please consult the [SQLite3 documentation](#).

Changing options

To change some of the program options, choose

Tools » Options

from the main menu.



The following options are configurable:

General options

- the look and feel of the user interface
- toolbar on/off
- tree panel options like object grouping or level of detail
- startup options

Editor options

- auto-save files
- word wrap
- tab width

Fonts

- default font - used everywhere except the code editors

- fixed font - used in the code editors

Data options

- use default alias
- display the results in grid or as text
- maximum text size to be displayed

Grid options

- show filter
- cell select mode
- cell auto height
- show cell hints
- show images
- show edit buttons
- show null values (as <null>)
- show hyperlinks
- show record numbers
- show "rowid" on Data tab
- dbl-click opens record editor
- highlight values having possibly incorrect storage classes
- max BLOB size to load in grids (bytes)
- horizontal and vertical alignment

Code Generation options

- capitalization
- quote characters

Code completion options

- enable code completion
- enable code completion auto-invoke
- auto-invoke delay
- enable code template completion
- code templates

SQLite library

Type mappings

This allows changing the internal mapping of data types. For more information about the type mappings, see [Data types](#).

File associations

This allows changing the file extensions associated with the application. By default the following extensions are associated: .db, .db3, .sqlite, .tooldb.

Updates

This allows changing the number of days between checks for updates.

Getting Help

- » [Integrated Help](#)
- » [SQLite Expert on the Web](#)
- » [SQLite on the Web](#)
- » [Sending feedback](#)

Integrated Help

The help file that comes with SQLite Expert contains complete documentation of all aspects of the program.

To display Help:

Choose **Help » Contents** or press **F1**.

SQLite Expert on the Web

[SQLite Expert web site](#) is a great resource for up-to-date information on SQLite Expert and other products. Through the SQLite Expert web site, you can find:

- < technical support
- < a knowledge base
- < news and tips

To access the SQLite Expert web site, choose

Help » SQLite Expert Home Page

or visit <http://www.sqliteexpert.com>.

SQLite on the Web

[SQLite web site](#) is a frequently updated website with links, tips, news, and downloads for SQLite users around the world. Through the SQLite web site, you can find:

- ⟨ the latest version of the database engine sqlite3.dll
- ⟨ the complete SQLite documentation
- ⟨ technical support on SQLite
- ⟨ a knowledge base
- ⟨ news and tips
- ⟨ a mailing list

To access the SQLite web site, choose

Help » SQLite Home Page or visit www.sqlite.org.

To access the SQLite online documentation, choose

Help » SQLite Online Help.

Sending feedback

To send us your suggestions, ideas, and bug reports about SQLite Expert, choose

Help » Send Feedback

from the main menu. This command launches the default mail program which allows you to send email to support@sqliteexpert.com.

Known Issues

Invalid field names in views

If you create a view from multiple tables using quoted identifiers and do not use field aliases, the generated view has invalid field names. This is a known issue in SQLite, not in SQLite Expert. For more information please consult the SQLite bug list:

<http://www.sqlite.org/src/wiki?name=Bug+Reports>

To work around this issue, use field aliases when generating a view. The integrated Query Builder automatically generates unique field aliases so the views created using the Query Builder are not affected by this problem.

Syntax error in queries using databasename.tablename.*

The following query

```
select databasename.tablename.* from databasename.tablename
```

returns:

```
near ".*": syntax error
```

However, the following query returns no error:

```
select databasename.tablename.fieldname from  
databasename.tablename
```

This is a SQLite error and it has been reported to www.sqlite.org ([ticket 4037](#)).