SQLiteManager

User's Manual

©2003-2010 SQLabs. All rights reserved.

Introduction 4

Overview 4

Registering 4

SQLite2 and SQLite3 4

Encodings 5

Special Notes for Windows Users 5

Startup Wizard 6

Design Panel 7

Design Panel 7

Creating/Altering Tables 8

Creating Views 9

Creating Indexes 10

Creating Triggers 11

Creating Reports Templates 12

Creating Notes 12

Main Panels 14

Manage Panel 14

SQL Panel 16

Analyze Panel 17

Verify Panel 18

Optimize Panel 19

Chart Panel 20

Vacuum and Settings 21

Vacuum 21

Settings 21

Import and Export 22

Exporting Data 22

Importing Data 23

Features 24

Converting Databases 24

REAL Server 25

Preferences 26

In Memory Databases 27

Bug Reporter 28

Script Manager 29

Appendix A 30

Contact Information 30

Copyright 30

Legal Stuff 30

Introduction

Overview

SQLiteManager is a powerful GUI database manager for sqlite databases, it combines an incredible easy to use interface with blazing speed and advanced features.

SQLiteManager allows you to open and work with sqlite 2, sqlite 3, in memory databases, AES 128 encrypted databases and with REAL Server databases. It allows you to create and browse tables, views, triggers and indexes. It enables you to insert, delete and updates records in a very intuitive way and it supports you arbitrary SQL commands. SQLiteManager also has a flexible report generator whereby you can create virtually any text report (including HTML and XML) using a powerful template language.

Each database that you open with SQLiteManager is presented in one main window with a toolbar with nine different panels: Design, Manage, SQL, Analyze, Verify, Optimize, Chart, Vacuum and Settings. This user's manual will cover each panel in detail.

In addition to the main window, SQLiteManager provides a number of functions that you access through its menus and buttons. This user's manual will cover every one of these functions in detail, but you might want to take a minute to browse all SQLiteManager's menus to familiarize yourself with everything it can do.

Registering

The unregistered version of SQLiteManager runs with certain limitations. Any query will return no more than 20 rows and you will not be able to export and import any data into a database. You will also not be able to convert, dump and generate reports on any database until you have registered. If you have a serial number for SQLiteManager, enter it into the preferences dialog, which you can reach through the Preferences menu item.

To became a registered user and receive technical support and updates you must register SQLiteManager via web at the address: http://www.sqlabs.com/store.php

SQLite2 and SQLite3

SQLiteManager can open and create both SQLite 2 and SQLite 3 databases. When you open a SQLite database, SQLiteManager will automatically determine whether the database is SQLite 2 or SQLite 3 and open it accordingly. The database version will be displayed in the window's title bar.
To create a new database, choose either SQLite 2 or SQLite 3 from the New menu (under File). SQLite 3 is the default if you are creating a database using the keyboard shortcut for new documents. SQLiteManager also has the ability to convert SQLite 2 databases to SQLite 3. Please see the section on converting databases for more information.

Encodings

By default, SQLiteManager displays all query results in both the Manage and SQL panels as UTF8. To change SQLiteManager's text encoding, choose a new encoding from the Database menu (under the Encoding sub-menu) The new encoding only affects the database in that window. Databases in other windows are unaffected.

When you change the text encoding with the encodings menu, the new text encoding takes effect immediately for data in the Manage and SQL panels. Also, if you add or edit records in the Manage tab, your data will be inserted into the database with the encoding in effect at that time.

Special Notes for Windows Users

In order to be able to correctly display Asian characters like Chinese or Japanese for example, under Windows XP you need to install a special Asian Language Pack.

1. Open the Windows Control Panel and (If the Control Panel is in "Category view") Select "Date, Time, Language and Regional options."

2. Open the "Regional and Language Options" icon.

3. Choose the "Languages" tab, and ensure the "Install files for East Asian languages" is checked.

4. Click the "Details" button to open the "Text services and input languages" dialog.

5. If "Chinese" is not listed in the "Installed Services" box, click "Add".

6. In the "Input Language" list, choose "Chinese (PRC)". The "Keyboard layout" should default to "Chinese (Simplified) - Microsoft Pinyin IME 3.0". Click "Ok" on both dialogs to return to the "Regional and Language Options".

7. Click "Ok", you will probably need to insert your Windows CD for the files to be installed.

8. The language bar should now have appeared in the bottom right of the taskbar. It should default to English - "EN".

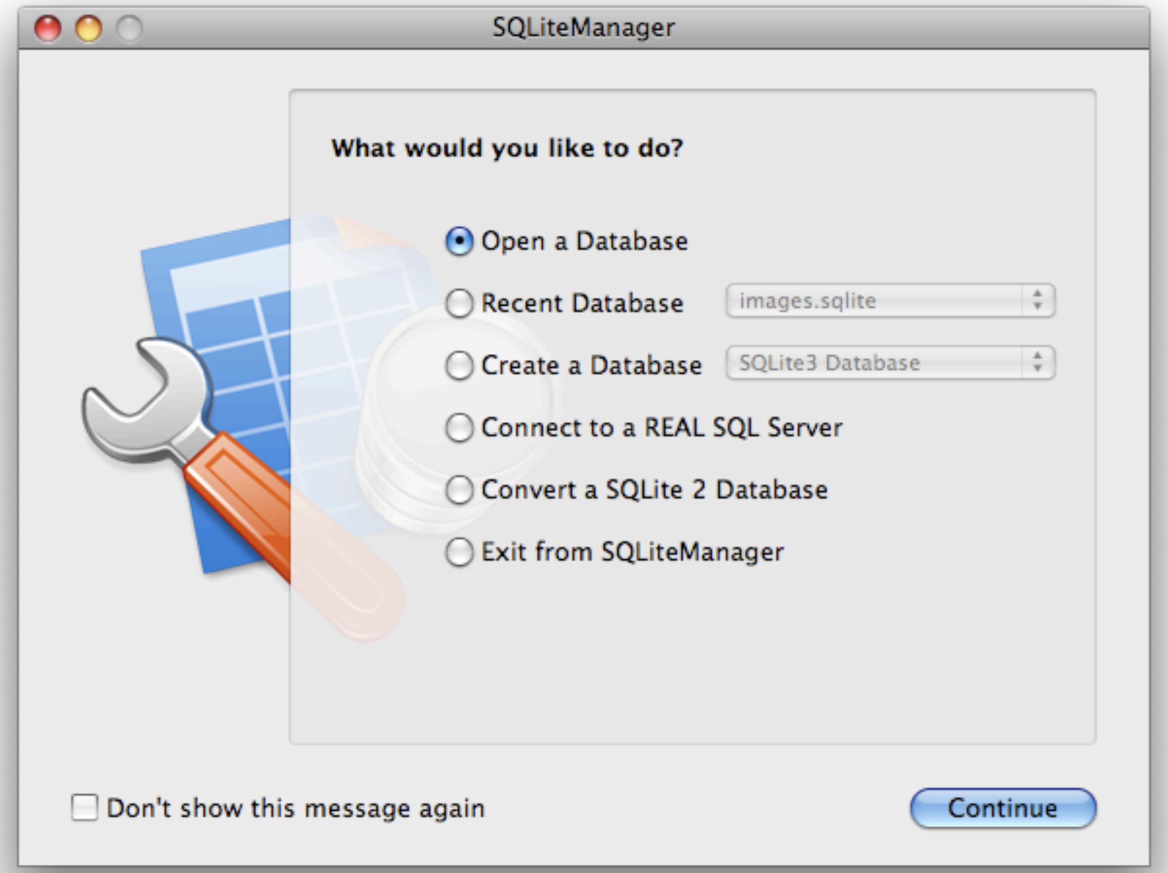
9. Click on the "EN" button to show the available languages.

10. By changing the language to "CH", you can now type in Pinyin. You can change the current language by pressing "ALT" + "SHIFT" on your keyboard.

More information can be found at: http://www.li-ming.org/Form/XPSetup.pdf

Startup Wizard

At startup SQLiteManager allows you to perform frequently used operations:


SQLiteManager

What would you like to do?

[✓ Open a Database]
Recent Database images.sqlite
Create a Database SQLite3 Database
Connect to a REAL SQL Server
Convert a SQLite 2 Database
Exit from SQLiteManager

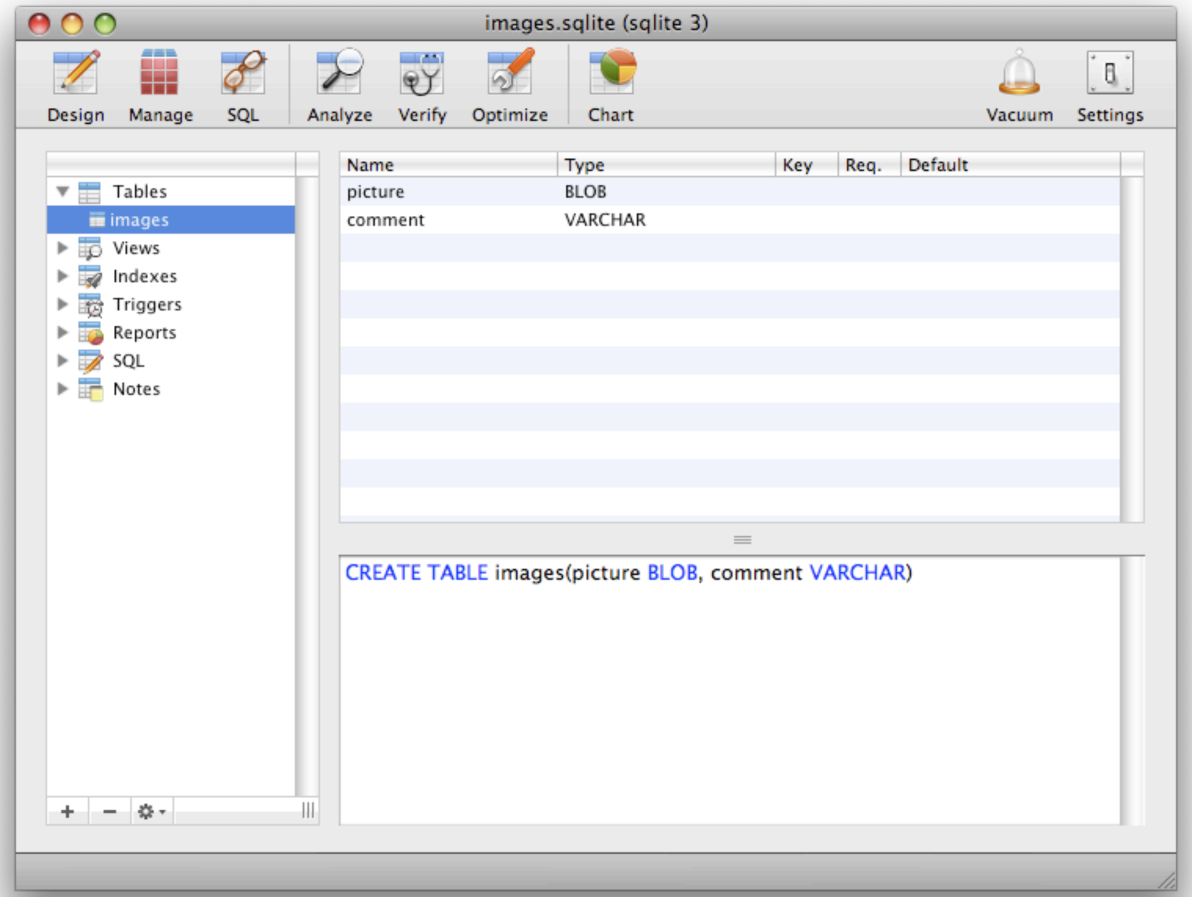
Don't show this message again

Within the Startup Wizard you can:

- Open an existing database
- Open a recently used database
- Create a new database (sqlite2, sqlite3, encrypted database or in-memory)
- Connect to a REAL Server
- Convert a local sqlite 2 database
- Exit from SQLiteManager

On MacOS X you can hide the wizard at each startup.

Design Panel



Name	Type	Key	Req.	Default
picture	BLOB			
comment	VARCHAR			

```
CREATE TABLE images(picture BLOB, comment VARCHAR)
```

The Design panel is where you create, inspect and update tables, views, indexes, triggers, report templates, SQL and Notes in a database. The panel is divided into two basic areas: an object browser on the left, and a space on the right where information about selected objects is displayed. The object browser is presented as a hierarchical list divided into Tables, Views, Indexes, Triggers, Reports, SQL and Notes. Opening any one of those categories displays the objects of that type that are in the database. If a category doesn't have any objects, then no objects will be displayed. Selecting an item in a category reveals details about that item in the space to the right of the object browser.

For example, selecting a table displays the schema for that table in a read-only edit field, and a table with more details about that table's fields and the properties of those fields. Field properties include the field's name and type, whether the field is a primary key, whether the field is required (can't be NULL), and the default value for the field. Selecting a view displays information about that view that is very much like the information that is displayed for a table. Note that the schema field for tables, views, and indexes is read-only, but you can copy the text in the field and paste it elsewhere.

We will have more to say about report templates and saved SQL later, but it is important to realize that in order to implement those two features, it is necessary for SQLiteManager to save information in a special table in your database file. The table is called "sqlitemanager_extras" and it contains all of the extra information that SQLiteManager may need to keep about a particular SQLite database.

SQLiteManager endeavors to filter that table name out of the places where table names are displayed. For example, you will not find a table by that name in the list of tables in the Design tab but it is still possible to interact with that table directly by typing SQL queries and commands into the SQL tab.

We recommend that you do not delete the "sqlitemanager_extras" table, as you may lose functionality by doing so. On the other hand, deleting that table should not interfere with the rest of your SQLite database objects in any way. While in the Design tab, you may wish to drop tables, indexes, or views. To do that, select the object you wish to drop from the object browser and then choose "Drop" from the Edit menu or just push the "Delete" button at the bottom of the Object browser table.

Creating/Altering Tables

To create a new table in SQLiteManager, press the "Add Table" shortcut button at the bottom of the object browser. You will be presented with a new table dialog, pictured below. Within the new table dialog, you can name your table, and then add columns to it. You may also set the properties of each field, such as whether a field is a Primary Key, not null, Unique (every row in the table must have a different value for the field) or Autoincrement.

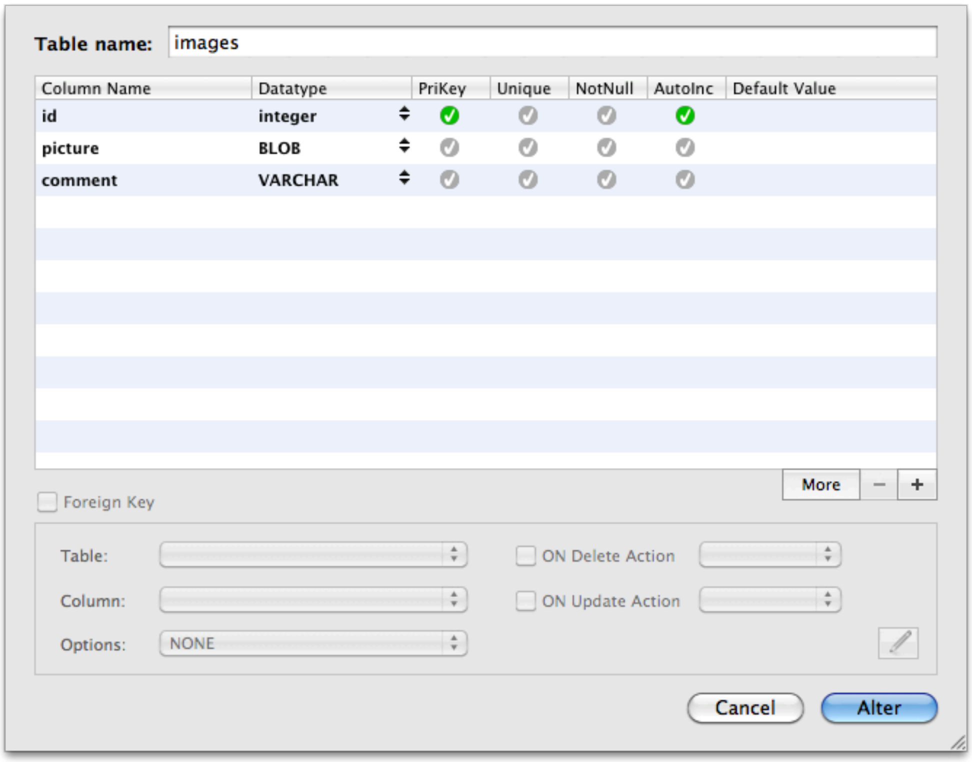


Table name: images

Column Name	Datatype	PriKey	Unique	NotNull	AutoInc	Default Value
id	integer	✓			✓	
picture	BLOB					
comment	VARCHAR					

Foreign Key

Table:

Column:

Options: NONE

ON Delete Action

ON Update Action

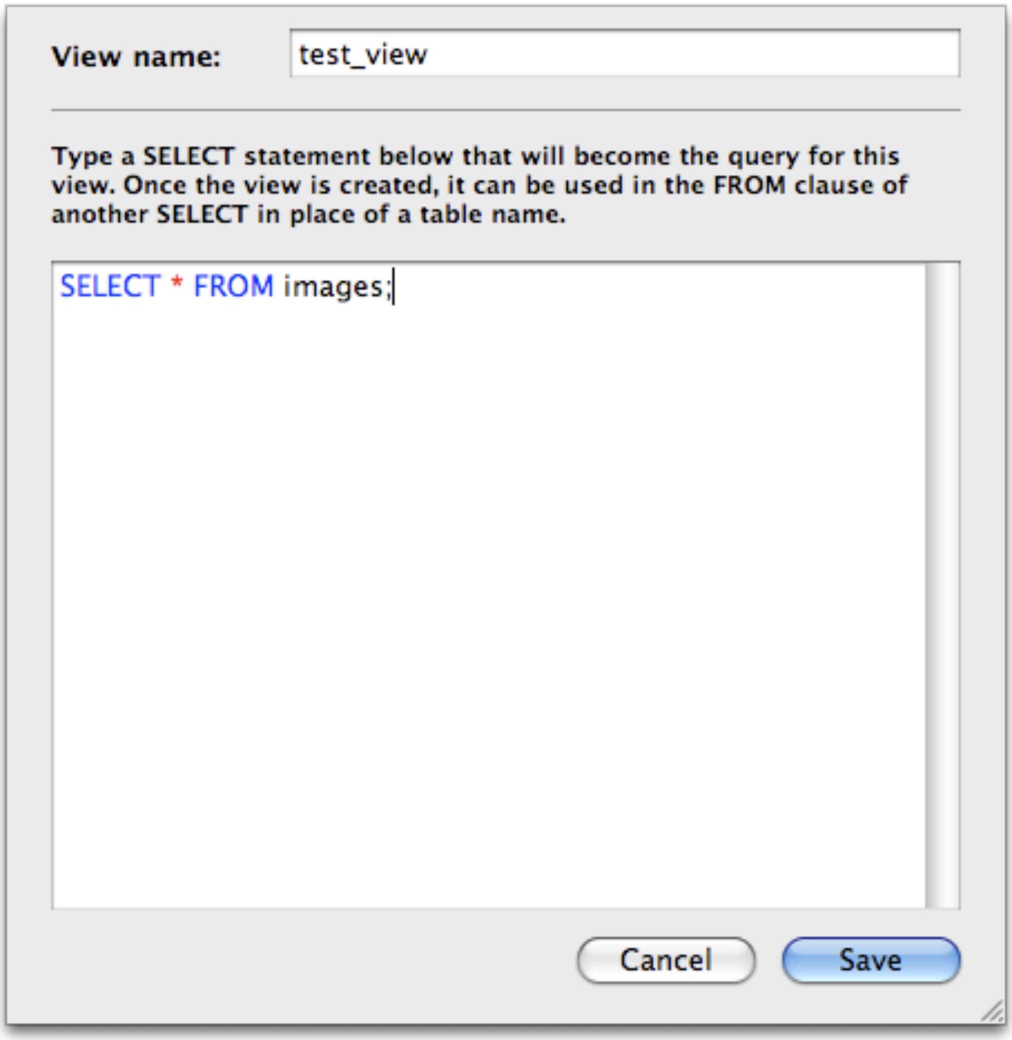
You may also specify a default value for the field so if a field has a default value and you don’t specify a value for that field when you insert a new record into the table, the field will be set to the default value automatically. Field types may either be typed in or set with the popup. The popup has the field types that SQLite and REALbasic understand but because SQLite is fundamentally type-less, you are free to type in just about any type you'd like. If you need to add a check constraint or a collate sequence just press the More button under the listbox and the appropriate columns will be added to the list.

Starting from SQLite 3.6.19 foreign keys are fully supported and SQLiteManager is now able (starting from version 3.5) to add foreign key to tables. Please note that SQLiteManager can add only Column Constraint foreign key definitions. If you need to add Table Constraint foreign key definitions you have to manually type the right sql command in the SQL panel.

The same dialog in used when you need to alter a table (not available for SQLite 2 databases).

Creating Views

To create a new view in SQLiteManager, choose Create View from the Action button (at the bottom of the object browser). You will be presented with a new view dialog, pictured below. A view is basically a saved query (SELECT SQL statement, in other words).


View name: test_view

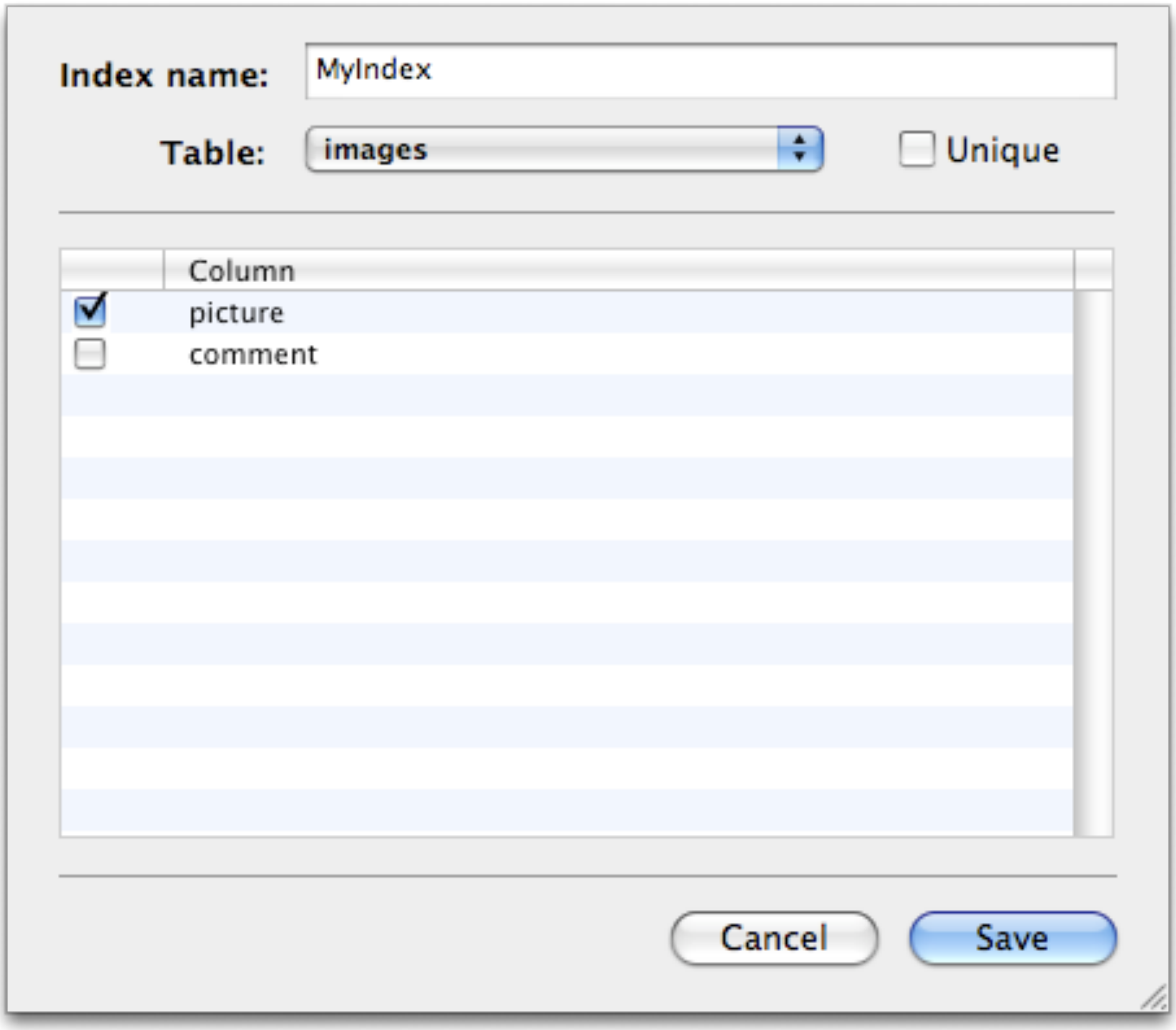
Type a SELECT statement below that will become the query for this view. Once the view is created, it can be used in the FROM clause of another SELECT in place of a table name.

```
SELECT * FROM images;
```

SQLite treats views very much like read-only tables. You can query them as you would a table, and you can include them in other queries as well. You are free to name your view anything you would like. You may then type any arbitrary SELECT statement into the query field and that SELECT statement will become the query for the view.

Creating Indexes

SQLite supports multiple indexes on a table and multiple fields in an index. To create a new index, choose Create Index from the Action button (at the bottom of the object browser). You will be presented with a new index dialog, pictured below.

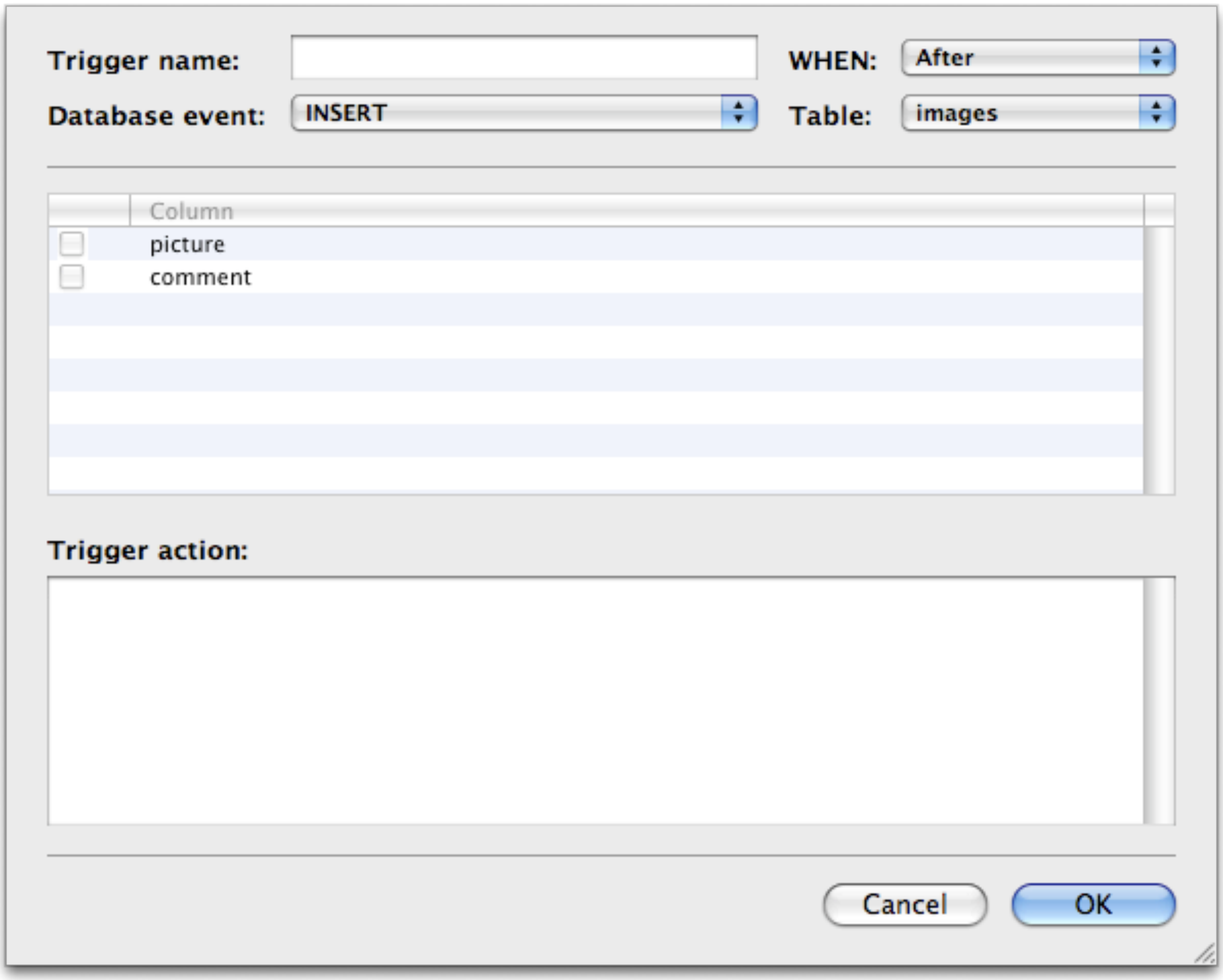

Index name: MyIndex
Table: images
Unique [ ]

	Column
✓	picture
☐	comment

The dialog allows you to name the index and choose the table for which the index will be created. Once you have chosen a table from the table popup menu, the ListBox will display the fields for that table. Check all fields that you wish to include in the index. You may also indicate whether or not the index is unique by checking the Unique checkbox.

Creating Triggers

SQLite fully supports triggers. To create a new trigger, choose Create Trigger from the Action button (at the bottom of the object browser) and you will be presented with a new trigger dialog, pictured below.


Trigger name:
WHEN: After
Database event: INSERT
Table: images

	Column
☐	picture
☐	comment

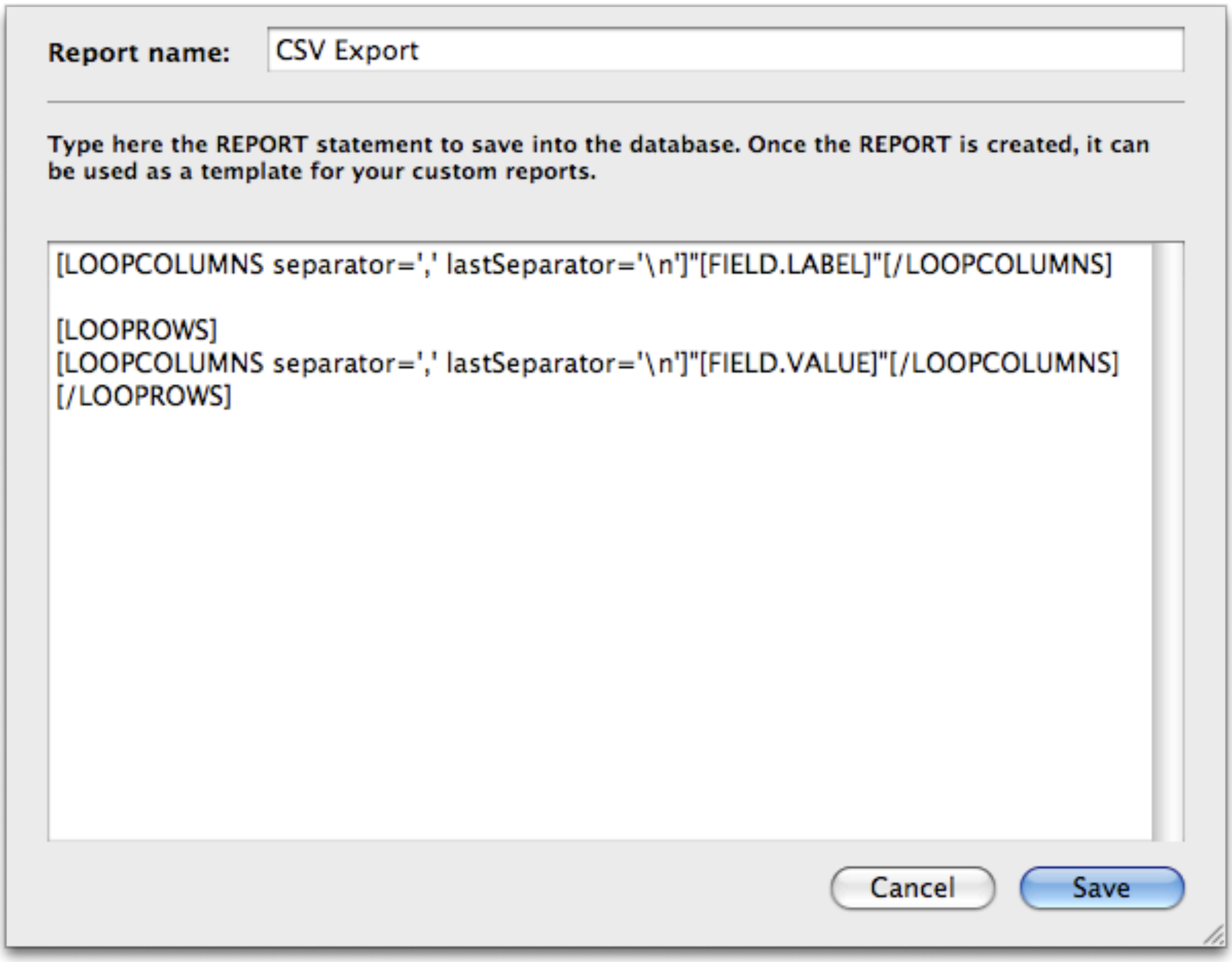
Trigger action:

Triggers are database operations (the trigger-action) that are automatically performed when a specified database event (the database-event) occurs. A trigger may be specified to fire whenever a DELETE, INSERT or UPDATE of a particular database table occurs, or whenever an UPDATE of one or more specified columns of a table are updated. At this time SQLite supports only FOR EACH ROW triggers, not FOR EACH STATEMENT triggers.

Hence explicitly specifying FOR EACH ROW is optional. FOR EACH ROW implies that the SQL statements specified as trigger steps may be executed (depending on the WHEN clause) for each database row being inserted, updated or deleted by the statement causing the trigger to fire.

Creating Reports Templates

We will talk more about generating reports in a later section, but before you can generate any reports, you have to create a report template. To create a report template, choose Create Report from the Action button (at the bottom of the object browser). You will be presented with a new report dialog, pictured below.



Report name: CSV Export

Type here the REPORT statement to save into the database. Once the REPORT is created, it can be used as a template for your custom reports.

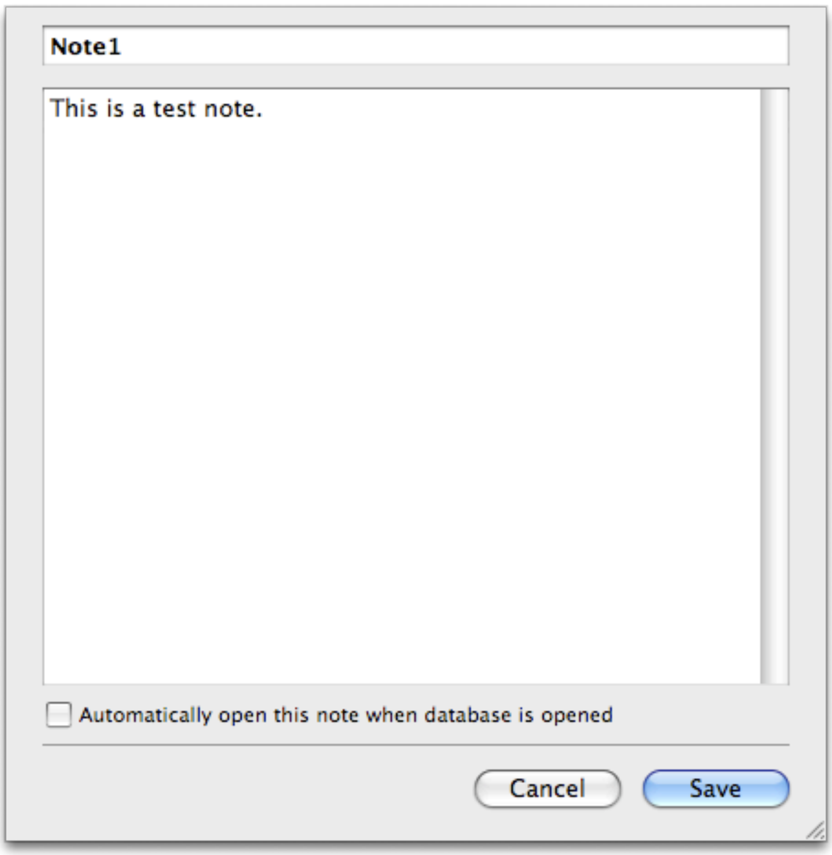
```
[LOOPCOLUMNS separator=',' lastSeparator='\n']"[FIELD.LABEL]"[/LOOPCOLUMNS]

[LOOPROWS]
[LOOPCOLUMNS separator=',' lastSeparator='\n']"[FIELD.VALUE]"[/LOOPCOLUMNS]
[/LOOPROWS]
```

The new report dialog allows you to enter a name for the report template, and then to enter the text of the template itself. You can edit the report template after you have created it by double clicking the template in the object browser (in the Design tab). More information about reports can be found in a separate Reports.rtf file (inside the Docs folder).

Creating Notes

To create a new note in SQLiteManager, choose Create Note from the Action button (at the bottom of the object browser). You will be presented with a new view dialog, pictured below.

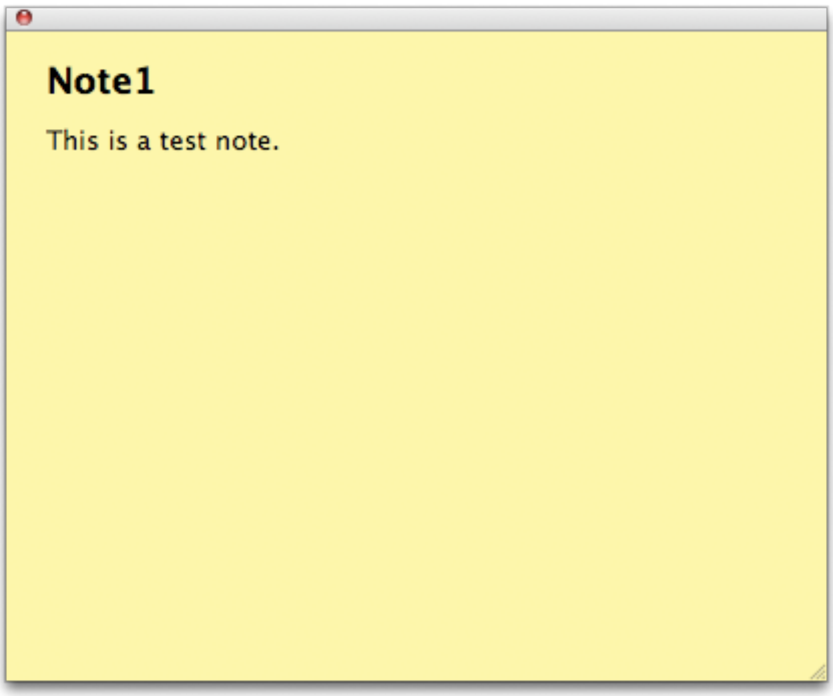
A note is basically a way to add comments and information to a database. You can optionally decide to automatically display the new note each time the database is opened by SQLiteManager.


Note1

This is a test note.

Automatically open this note when database is opened

Note will be displayed with a sticky like style:

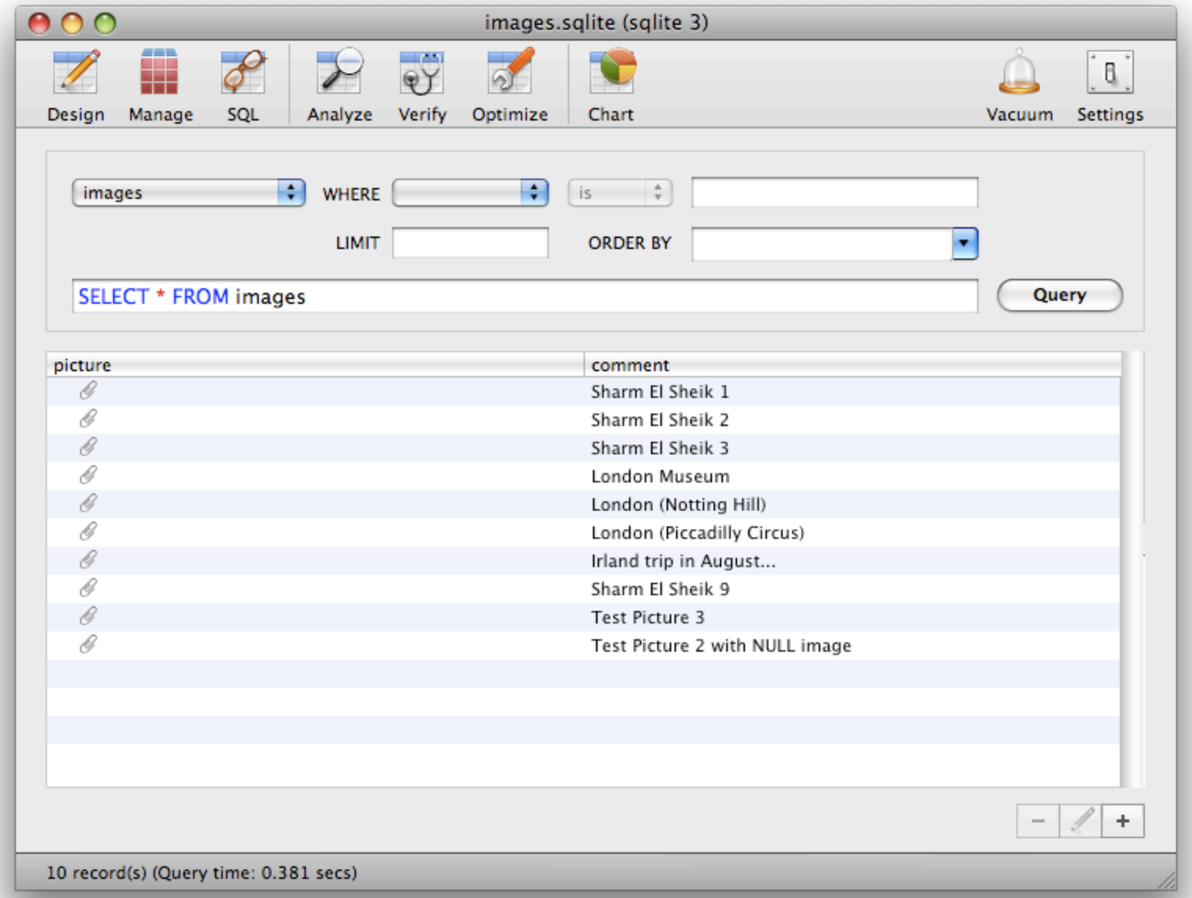

Note1

This is a test note.

Main Panels

Manage Panel

You will visit the Manage panel when you want to insert, remove, or edit records in a table. The tab is divided into three main areas. At the top of the tab are the popup menus and edit fields for building a query. Below that is a read-only edit field where the SQL for the query is displayed. You are free to copy this SQL and paste it elsewhere, but you cannot type arbitrary SQL into that field.



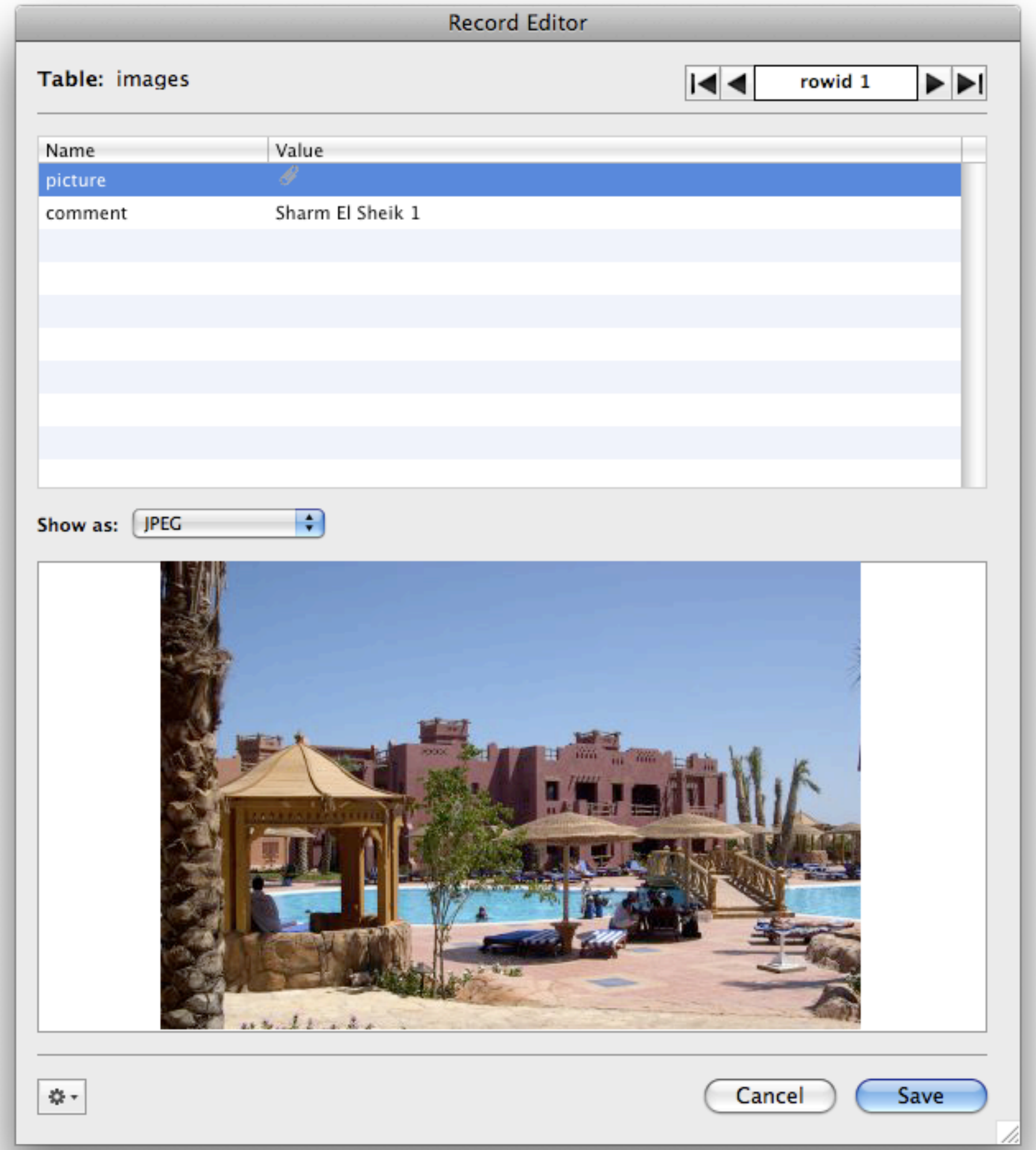
```
SELECT * FROM images
```

picture	comment
	Sharm El Sheik 1
	Sharm El Sheik 2
	Sharm El Sheik 3
	London Museum
	London (Notting Hill)
	London (Piccadilly Circus)
	Irland trip in August...
	Sharm El Sheik 9
	Test Picture 3
	Test Picture 2 with NULL image

10 record(s) (Query time: 0.381 secs)

Below the edit field is a list box containing the results of a query. To build a query, simply construct a SELECT statement out of the popups and edit fields and click the Query button. The query results will be displayed in the list box. To remove a record from the queried table, select it and click the Remove button. To edit a record, double-click it in the list box and the Record Editor dialog will appear (if inline editing has been disabled in the Preferences).

When you open a record to edit it (just double click on it), the fields of the record will be presented in a list box, with a text box below to edit the values of those fields. Make any changes you would like and click the Save button. The changes will be saved in the database immediately. Or, you can click Cancel and the changes will be discarded. This powerful dialog gives you the option to display current fields as TEXT, various graphical formats like JPEG, TIFF, BMP and so on (it depends on the platform) and to show it as a raw BinHex image.



Name	Value
picture	
comment	Sharm El Sheik 1

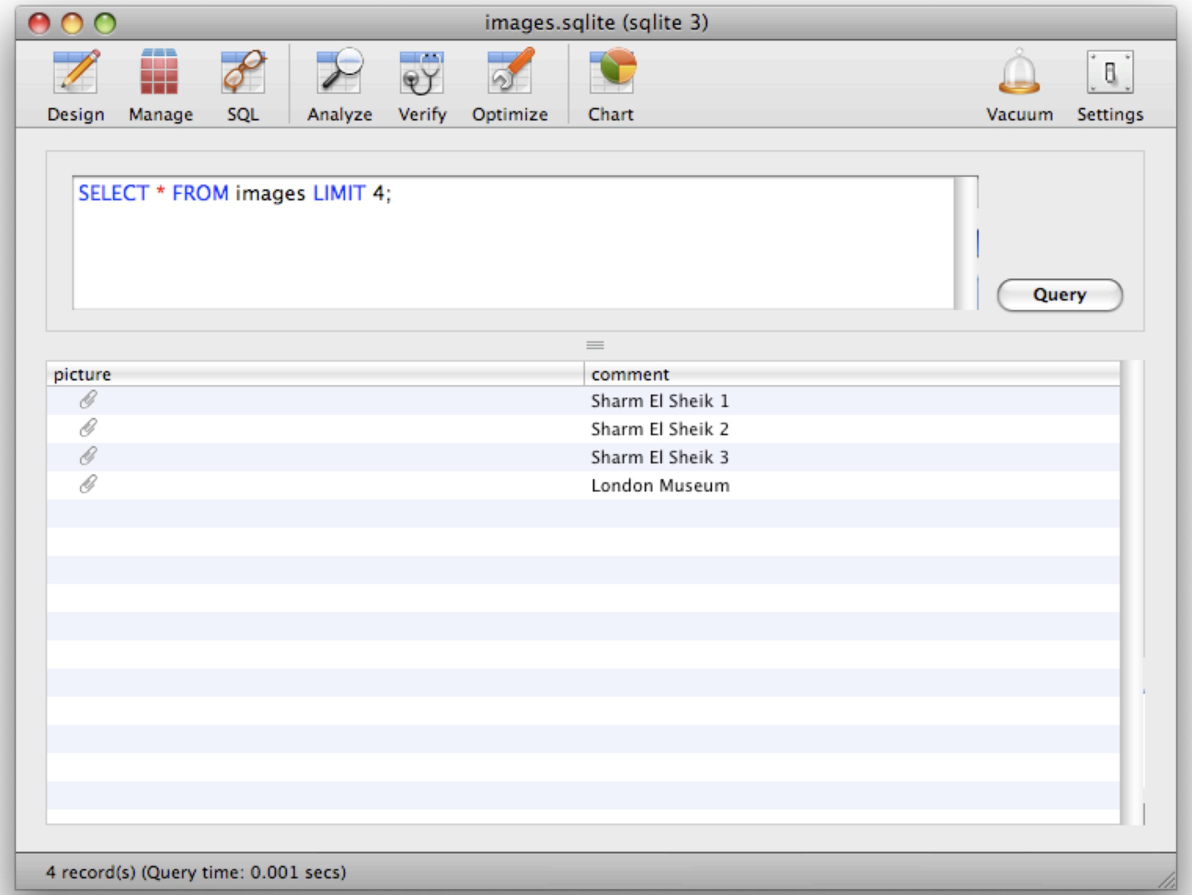
Show as: JPEG

To insert a new record into a table in the database, click the Insert button in the Manage tab of SQLiteManager’s main window. The dialog used for inserting records is exactly the same as that used for editing records.

SQL Panel

You will visit the SQL panel when you need to type arbitrary SQL to perform complex queries and commands on an SQLite database. The tab is divided into two main areas: a large edit field for typing SQL at the top of the tab and a list box for displaying results at the bottom of the tab. To use the SQL tab, just type any SQL into the edit field and click either the Execute/Query button.

If you want to display the results from such a command you must click the Select button. If you type an SQL statement that you'd like to save to use another time, just click the left mouse button and a popup menu will appear allowing you to save the SQL in the database or retrieving a previously saved one .



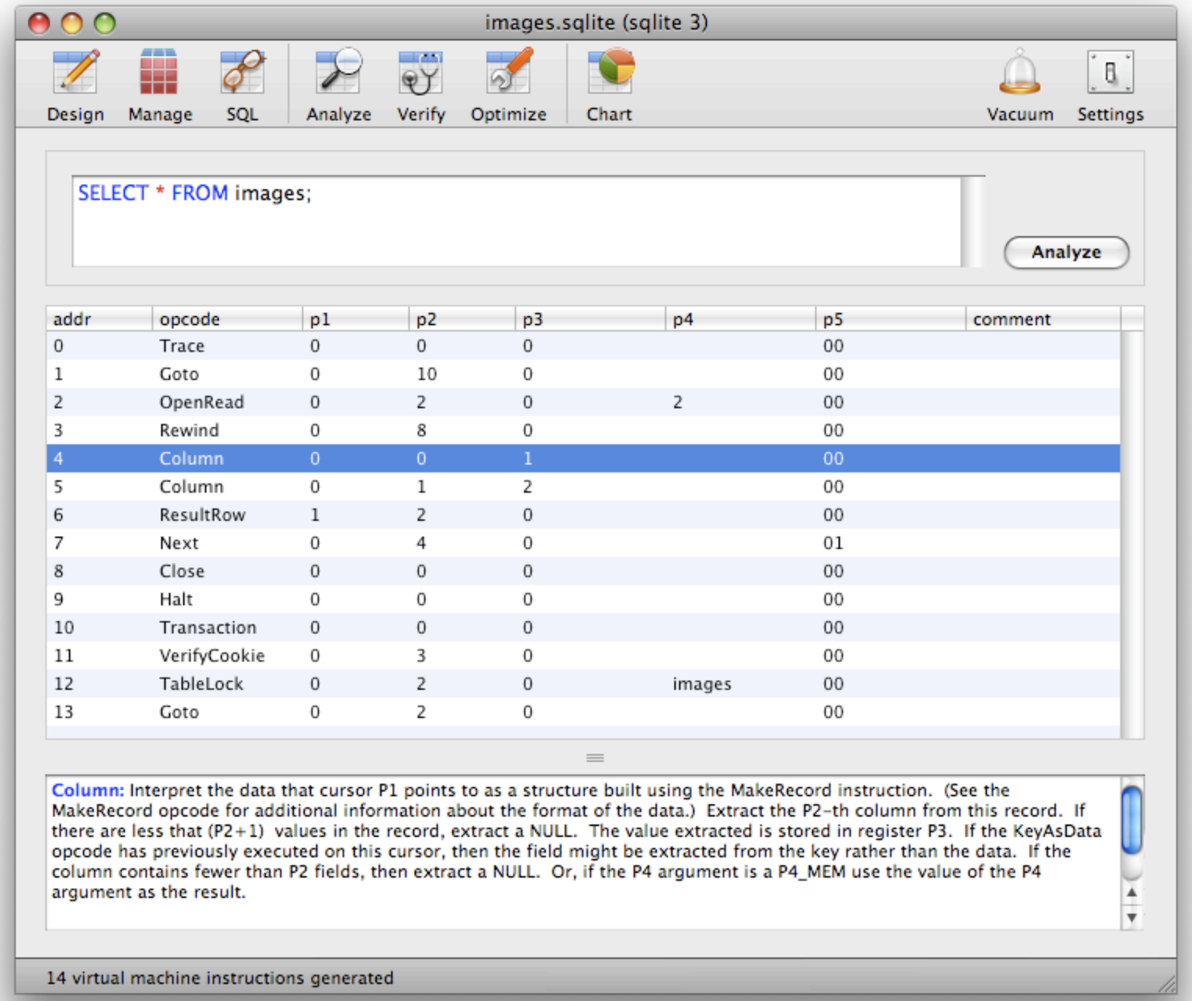
```
SELECT * FROM images LIMIT 4;
```

picture	comment
	Sharm El Sheik 1
	Sharm El Sheik 2
	Sharm El Sheik 3
	London Museum

4 record(s) (Query time: 0.001 secs)

Analyze Panel

You will visit the Analyze panel when you want to examine and disassemble SQL commands into low level virtual machine instructions. This is the most powerful way to check if a query or a SQL statement is efficient or if it can be rewritten in a better way. Each virtual machine opcode has a very detailed description at the bottom of the window.



```
SELECT * FROM images;
```

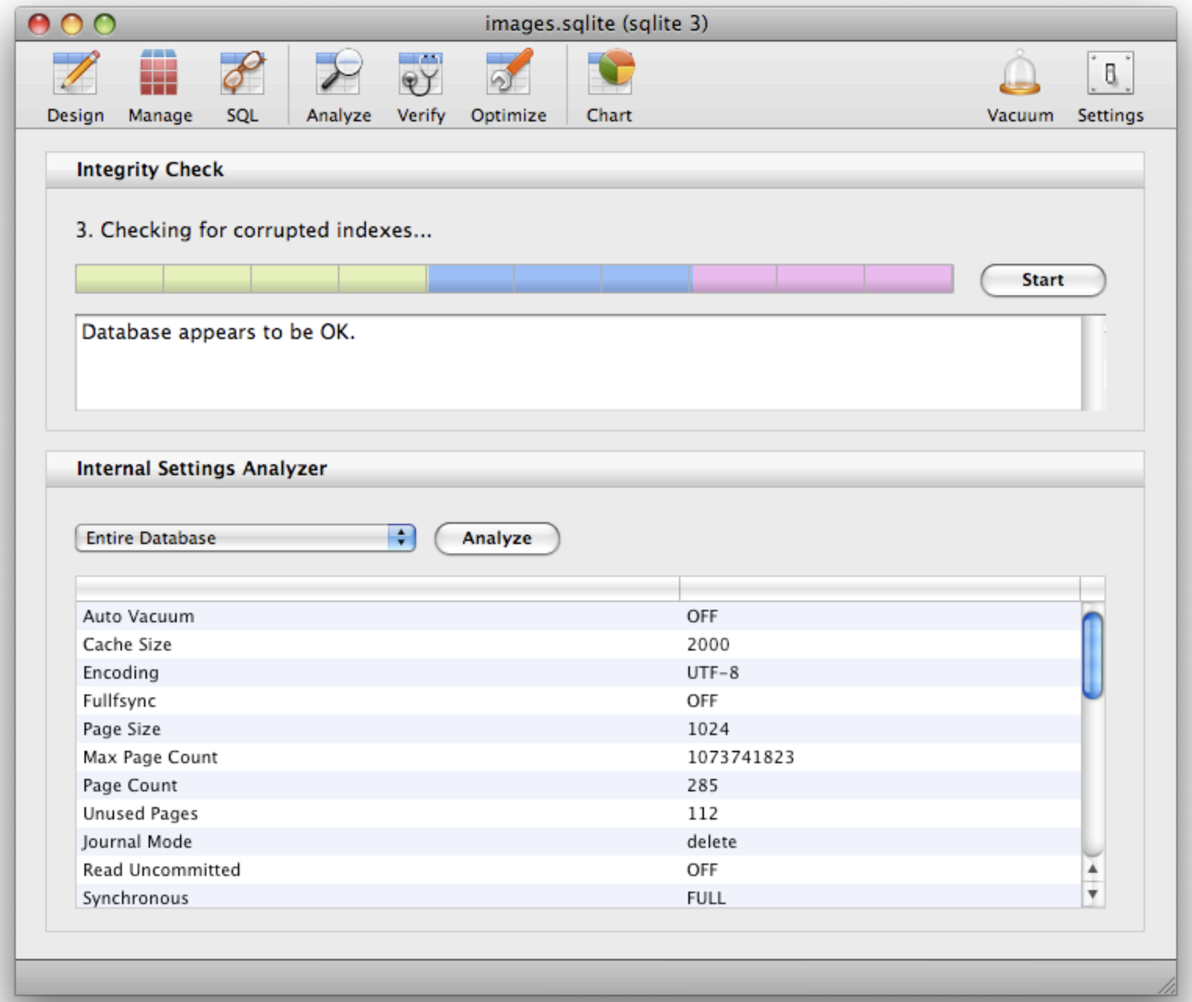
addr	opcode	p1	p2	p3	p4	p5	comment
0	Trace	0	0	0		00	
1	Goto	0	10	0		00	
2	OpenRead	0	2	0	2	00	
3	Rewind	0	8	0		00	
4	Column	0	0	1		00	
5	Column	0	1	2		00	
6	ResultRow	1	2	0		00	
7	Next	0	4	0		01	
8	Close	0	0	0		00	
9	Halt	0	0	0		00	
10	Transaction	0	0	0		00	
11	VerifyCookie	0	3	0		00	
12	TableLock	0	2	0	images	00	
13	Goto	0	2	0		00	

Column: Interpret the data that cursor P1 points to as a structure built using the MakeRecord instruction. (See the MakeRecord opcode for additional information about the format of the data.) Extract the P2-th column from this record. If there are less that (P2+1) values in the record, extract a NULL. The value extracted is stored in register P3. If the KeyAsData opcode has previously executed on this cursor, then the field might be extracted from the key rather than the data. If the column contains fewer than P2 fields, then extract a NULL. Or, if the P4 argument is a P4_MEM use the value of the P4 argument as the result.

14 virtual machine instructions generated

Verify Panel

You will visit the Verify panel when you want to perform a sanity check over your entire database or when you want to examine all the most important internal database settings in a very easy way. The Integrity Checker checks for out of orders records, for missing pages, for malformed records and for corrupted indexes. A detailed reports is shown if some problems are found inside the database.



3. Checking for corrupted indexes...

Database appears to be OK.

Auto Vacuum	OFF
Cache Size	2000
Encoding	UTF-8
Fullfsync	OFF
Page Size	1024
Max Page Count	1073741823
Page Count	285
Unused Pages	112
Journal Mode	delete
Read Uncommitted	OFF
Synchronous	FULL

Optimize Panel

You will visit the Optimize panel when you want to find out a way to optimize your database or your queries. The panel is divided in two main areas: The Index Optimizer that gathers statistics about indices and stores them in a special tables in the database where the query optimizer can use them to help make better index choices.

The Query Optimizer that enables you to analyze which index is used when you perform a query and its order if more than one index is used. Its output tells you how sqlite is scanning the tables and indexes to implement your query.

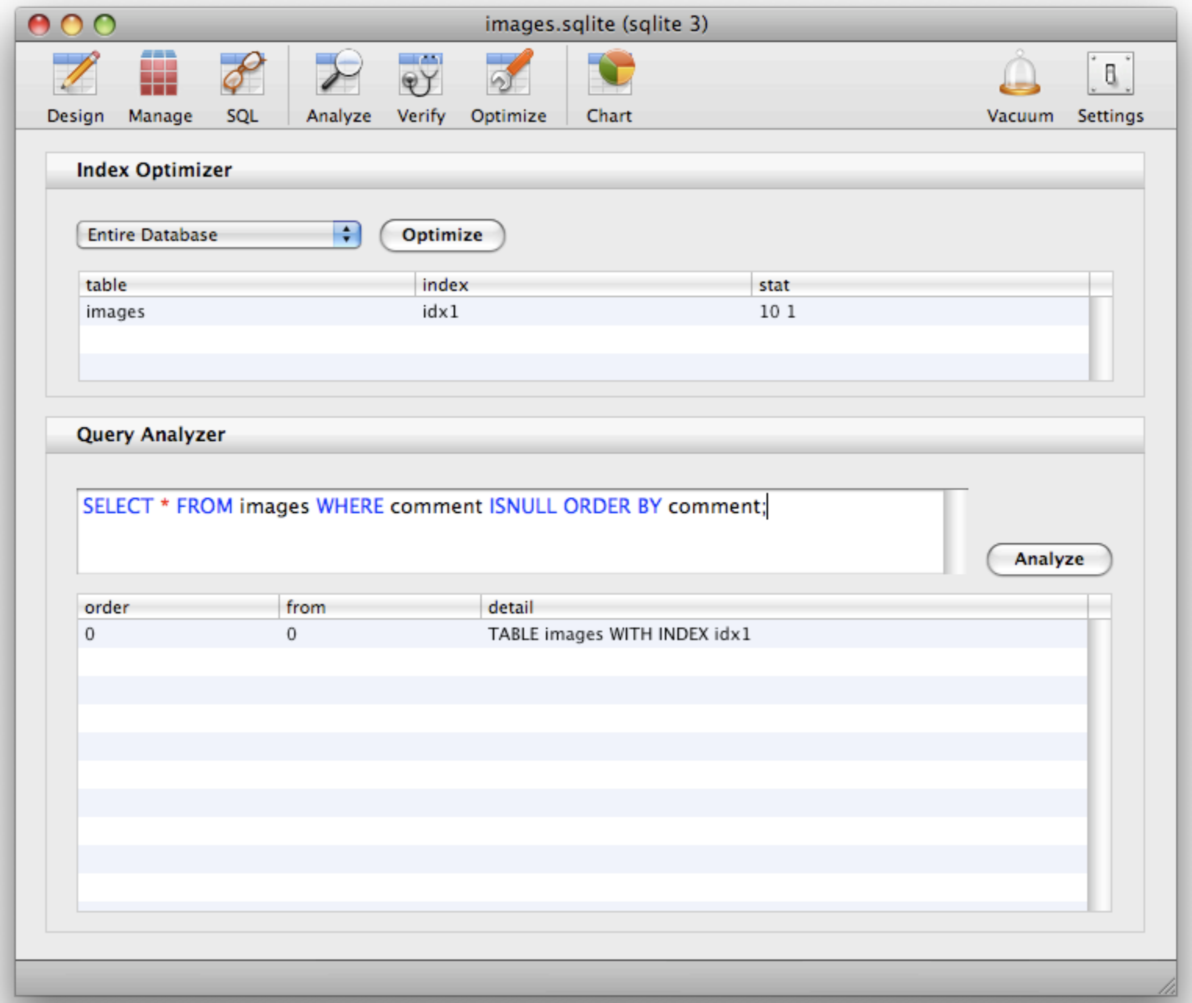


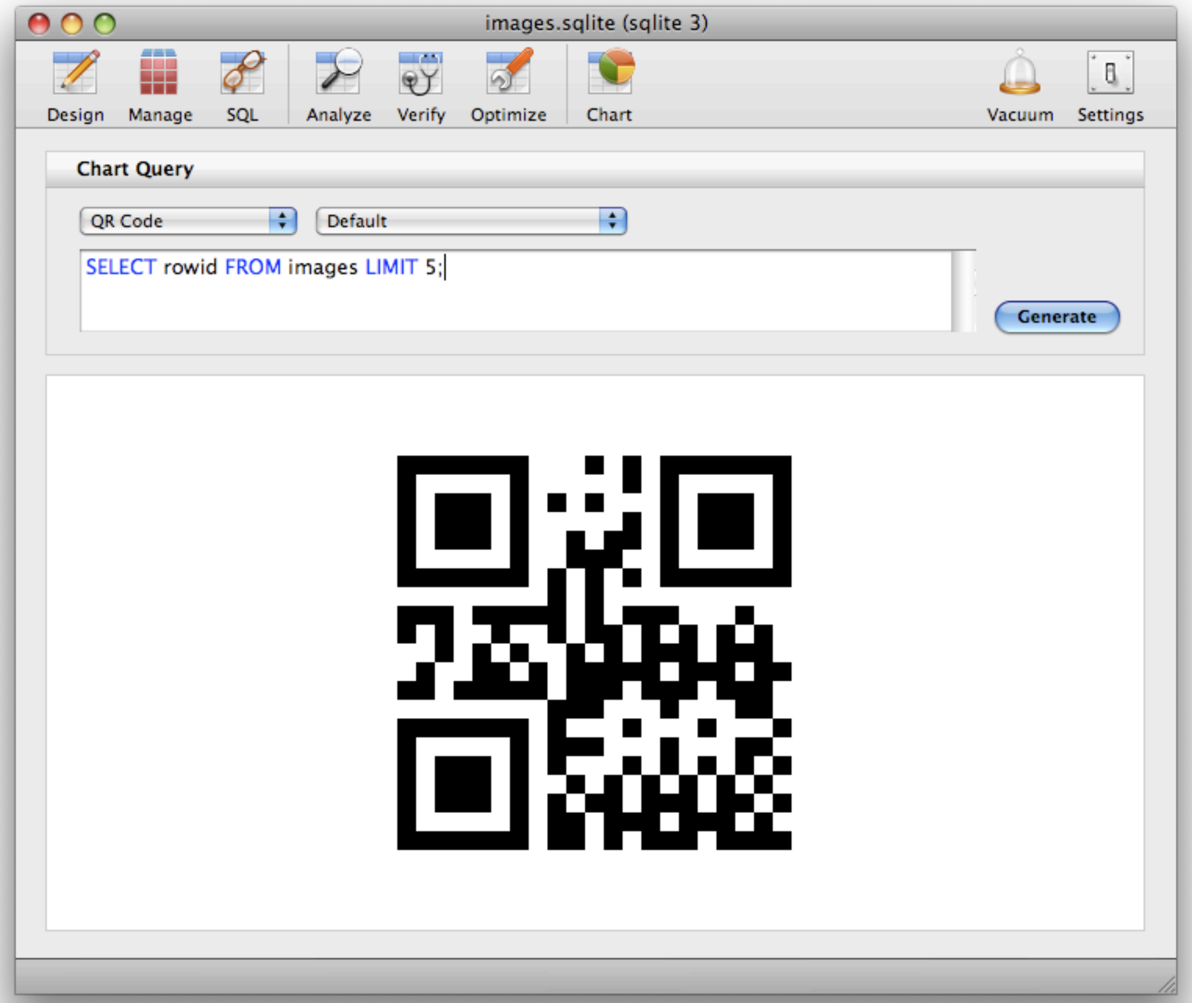
table	index	stat
images	idx1	10 1

```
SELECT * FROM images WHERE comment ISNULL ORDER BY comment;
```

order	from	detail
0	0	TABLE images WITH INDEX idx1

Chart Panel

The new Chart Panel allows you to easily visualize your data with Line Chart, Bar Chart, Pie Chart, Venn Chart, Scatter, Radar, Map and even QR Code. Just type a simple sql query and you can plot your data in 2D and 3D!



```
SELECT rowid FROM images LIMIT 5;
```

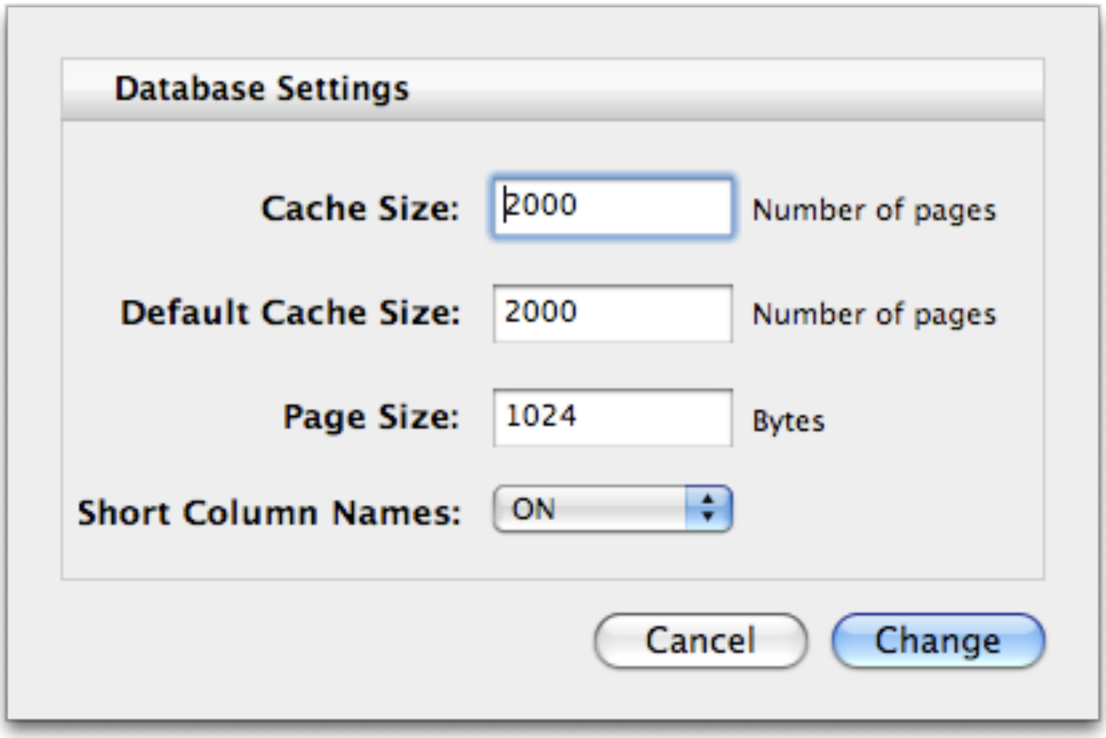
Vacuum and Settings

Vacuum

When an object (table, index, or trigger) is dropped from the database, it leaves behind empty space. This makes the database file larger than it needs to be, but can speed up inserts. In time inserts and deletes can leave the database file structure fragmented, which slows down disk access to the database contents. The VACUUM 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.

Settings

You will visit the Setting panel when you want to inspect or change the most important internal database settings. Please make sure to completely understand what are you trying to change because these settings can really affect your database performance.


Database Settings

Cache Size: 2000 Number of pages

Default Cache Size: 2000 Number of pages

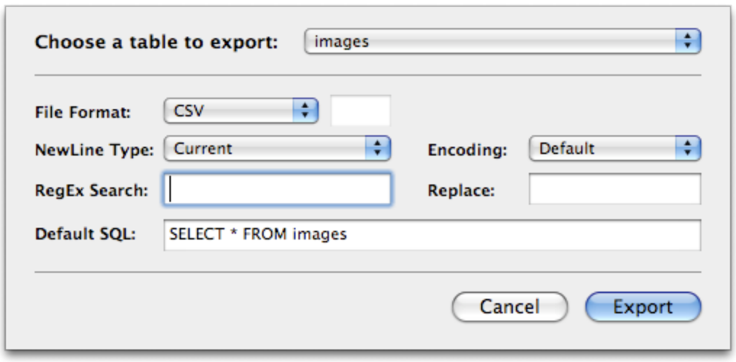
Page Size: 1024 Bytes

Short Column Names: ON

Import and Export

Exporting Data

SQLiteManager can export data in several common formats. The currently supported formats are CSV, SQL, tab-delimited and custom character delimited. To export data from an open database, choose an export format from the Export hierarchical menu in the File menu.



Choose a table to export: images

File Format: CSV

NewLine Type: Current

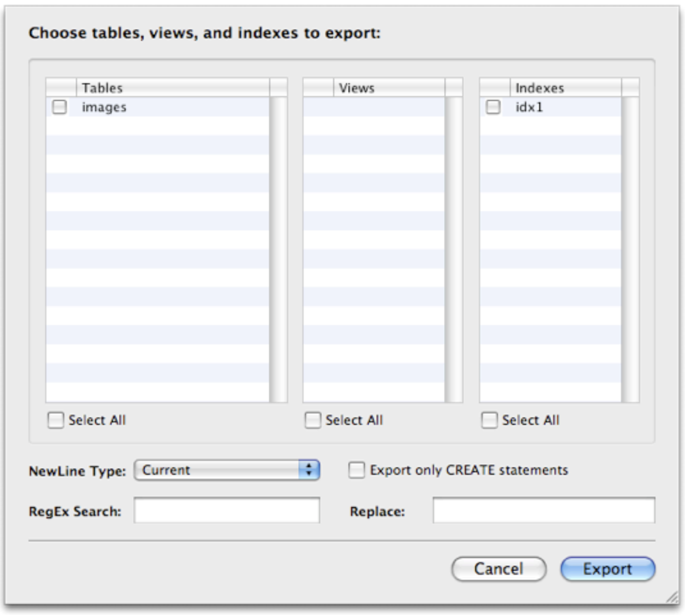
Encoding: Default

RegEx Search:

Replace:

Default SQL: SELECT * FROM images

Choosing “Others” exports displays a dialog asking which table in the database to export, which format to use, which encodings and so on. Every types of exports include the table headers as the first row of the exported data. Choosing a SQL export will display the dialog pictured below. Select which tables, views, and indexes you wish to export. You can choose to export only the CREATE statements by unchecking the “Export data” checkbox.



Choose tables, views, and indexes to export:

Tables	Views	Indexes
☐ images		☐ idx1

☐ Select All ☐ Select All ☐ Select All

NewLine Type: Current

☐ Export only CREATE statements

RegEx Search:

Replace:

You can also select the new line type (MacOS, Windows, Linux or Current) and apply a special RegEx search/replace pattern to “transform” your data before exporting. Although SQLiteManager can export in CSV, SQL, and tab-delimited formats automa-tically for you, you have virtually unlimited exporting options when you use a custom report template. See the section on report templates for more information.

Importing Data

To import data to an open database, choose an import format from the Import hierarchical menu in the File menu. If you choose "Others" you will be prompted both for a file of data of the appropriate format and a table name into which to import the data.

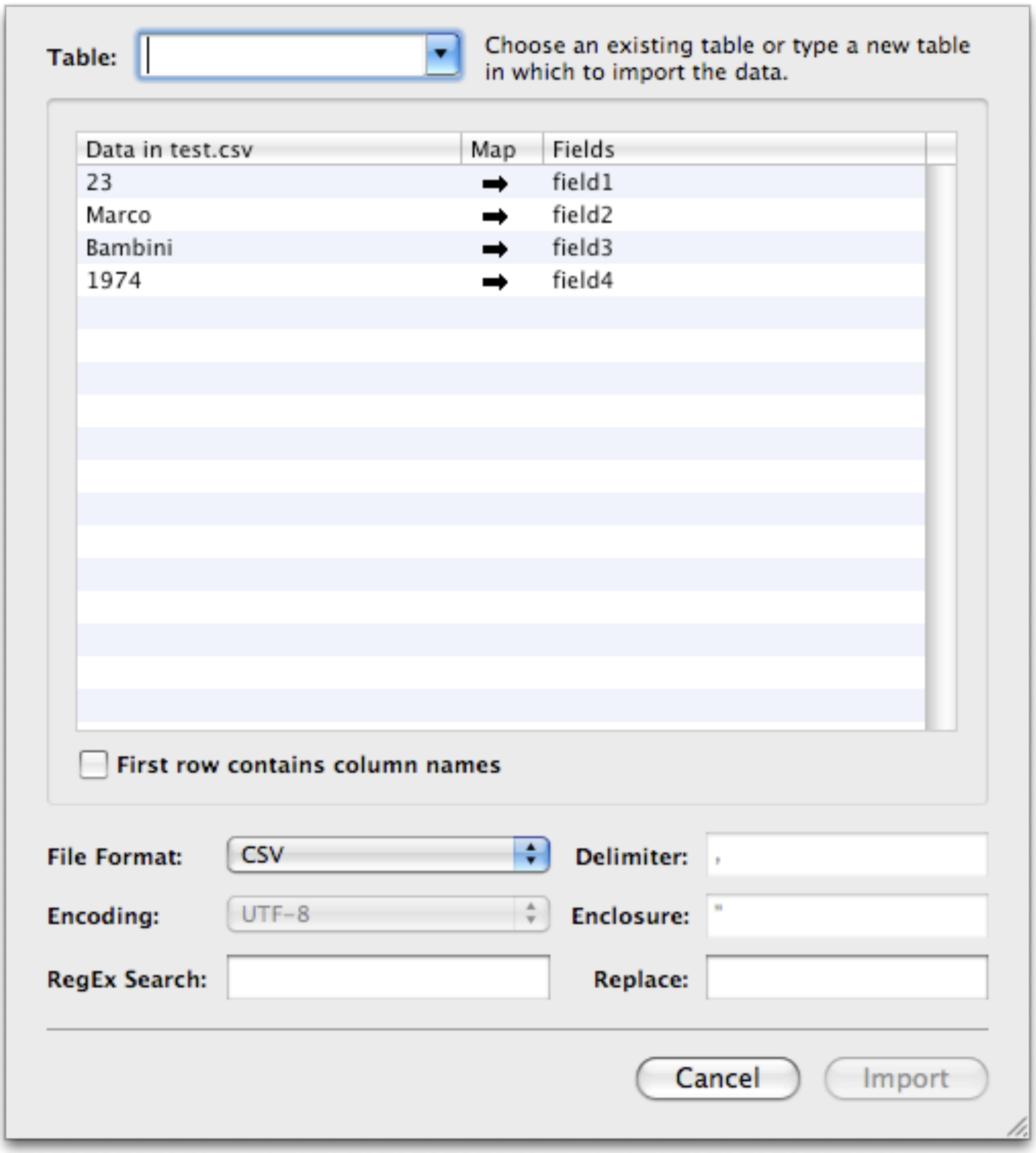


Table: Choose an existing table or type a new table in which to import the data.

Data in test.csv	Map	Fields
23	➡	field1
Marco	➡	field2
Bambini	➡	field3
1974	➡	field4

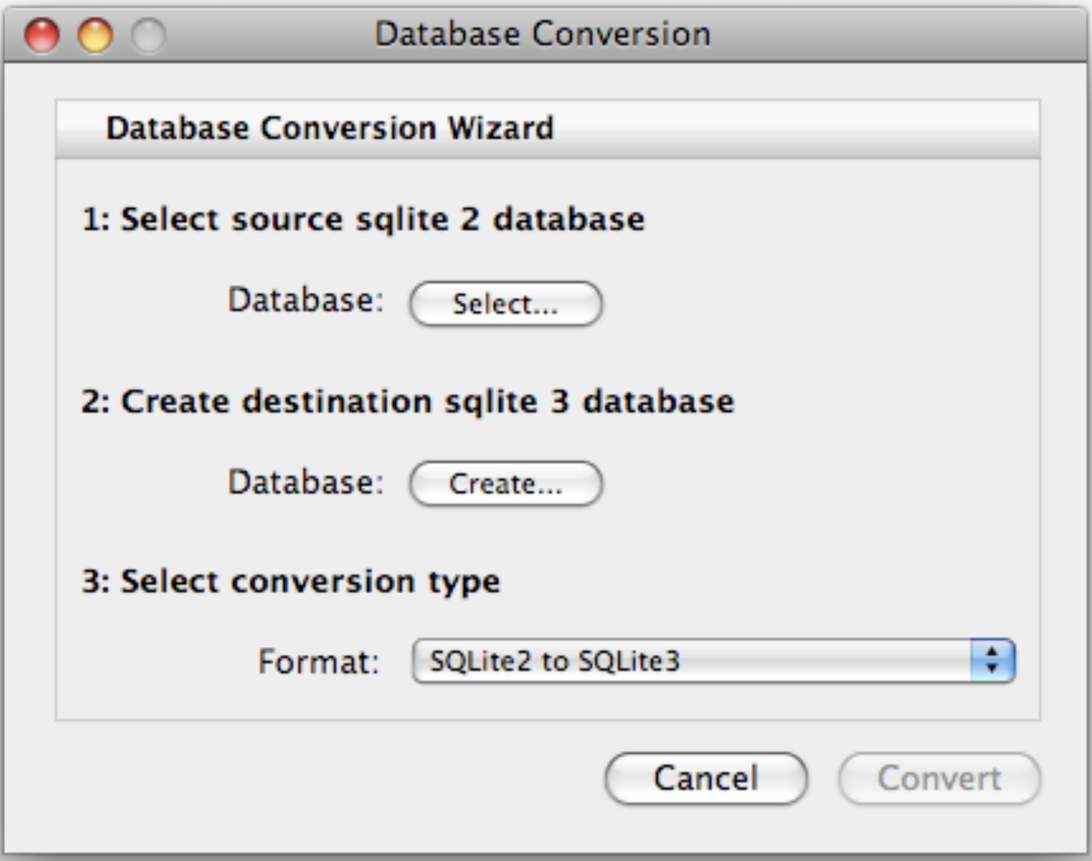
☐ First row contains column names

File Format: CSV
Delimiter: ,
Encoding: UTF-8
Enclosure: "
RegEx Search:
Replace:

If you choose to import SQL, then you will not be prompted for a table in which to import the data, as SQL statements that insert data into tables refer to those tables directly. Any SQL statements can be in the file you import, including statements to create tables and indexes and such. If you wanted to recreate an entire database, you could create a new, empty database in SQLiteManager, and then import the SQL that would create and populate all of the tables in that database.

Converting Databases

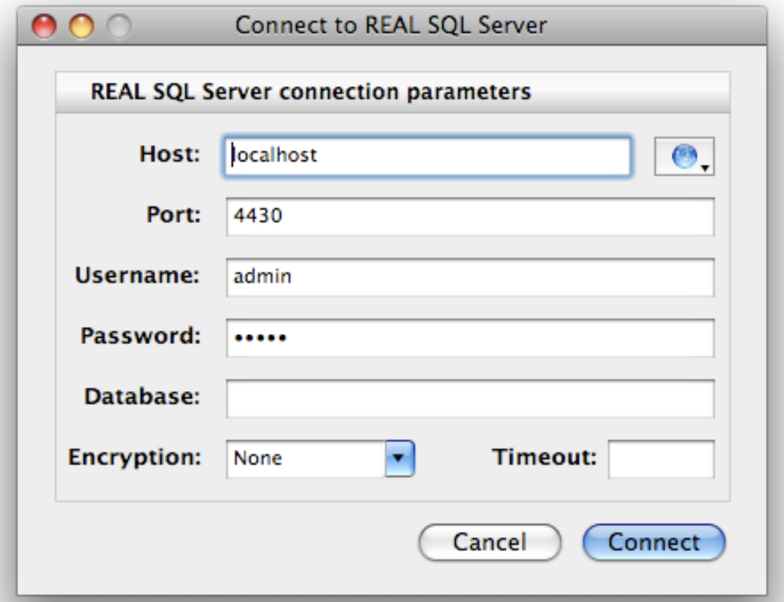
SQLiteManager can convert SQLite 2 databases to SQLite 3 format. Choose "SQLite 2 to SQLite 3" from the Convert menu and you will be prompted for a SQLite 2 database to convert. In addition to tables, SQLiteManager will also convert the views, indexes, and triggers in the SQLite 2 database.


Database Conversion
Database Conversion Wizard
1: Select source sqlite 2 database
Database:
Select...
2: Create destination sqlite 3 database
Database:
Create...
3: Select conversion type
Format:
Cancel
Convert

Format: SQLite2 to SQLite3

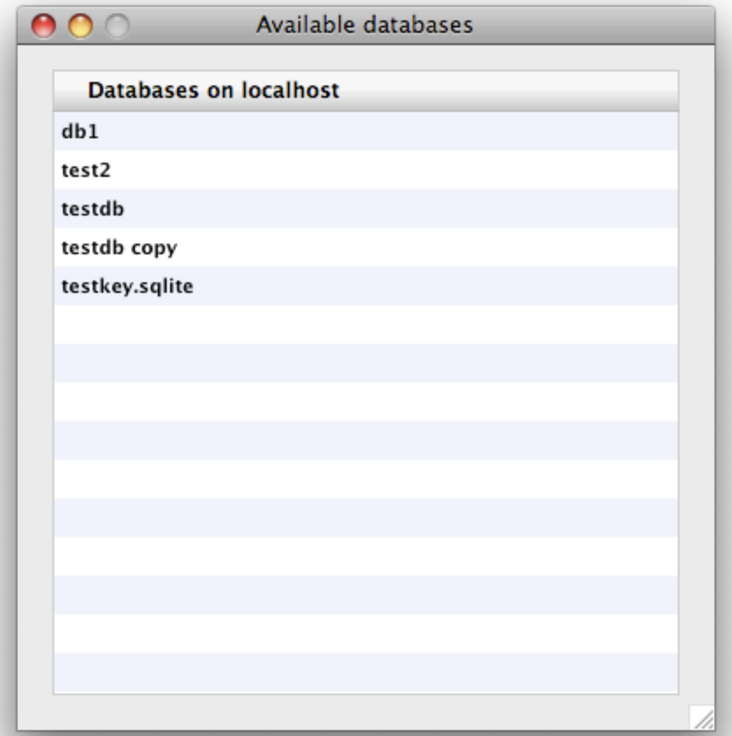
REAL Server

REAL Server is a powerful and fully featured SQL server build upon SQLite 3 databases technology. SQLiteManager can easily connects to a remote server and uses its databases as normal local database files. For a better experience, an admin account is recommended when you try to manage a remote database.


Connect to REAL SQL Server
REAL SQL Server connection parameters
Cancel
Connect

Host:	localhost
Port:	4430
Username:	admin
Password:	•••••
Database:	
Encryption:	None
Timeout:	

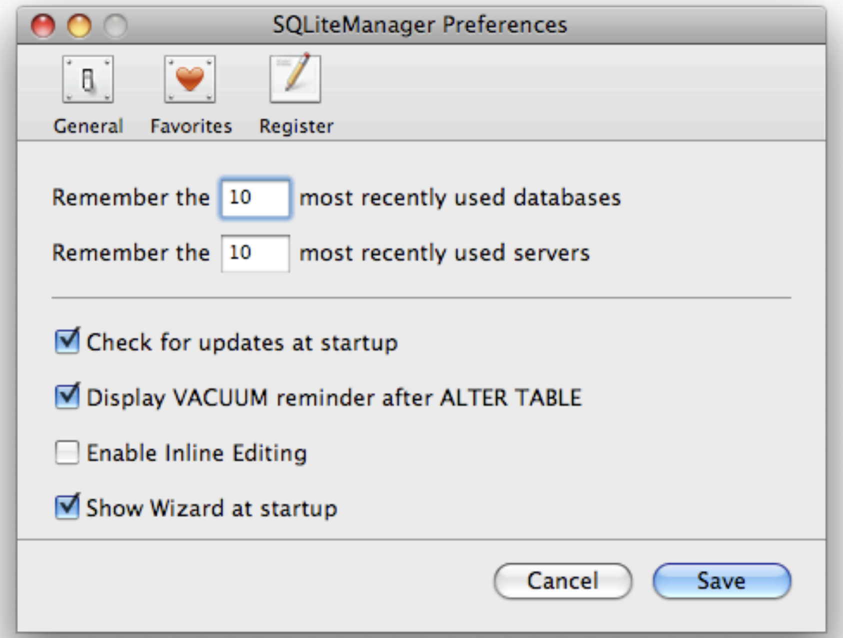
If you leave the Database field blank, SQLiteManager will automatically retrieve all the databases available on the server, then just double click of the one you want to connect to.



Databases on localhost
db1
test2
testdb
testdb copy
testkey.sqlite

Preferences

Use SQLiteManager's preferences dialog, shown above, to adjust the SQLiteManager's behavior. You can set the maximum number of databases and server to remember or you can modify the appearance of the Query and SQL listboxes. Note that the option "Show Wizard at startup" is available only in the MacOS X version.

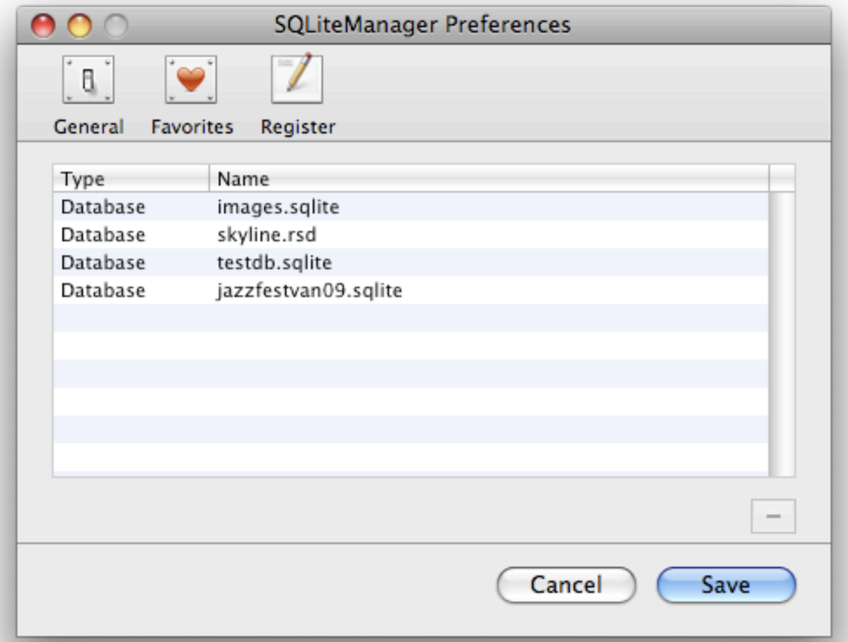


Remember the 10 most recently used databases

Remember the 10 most recently used servers

- [✓] Check for updates at startup
- [✓] Display VACUUM reminder after ALTER TABLE
- [ ] Enable Inline Editing
- [✓] Show Wizard at startup

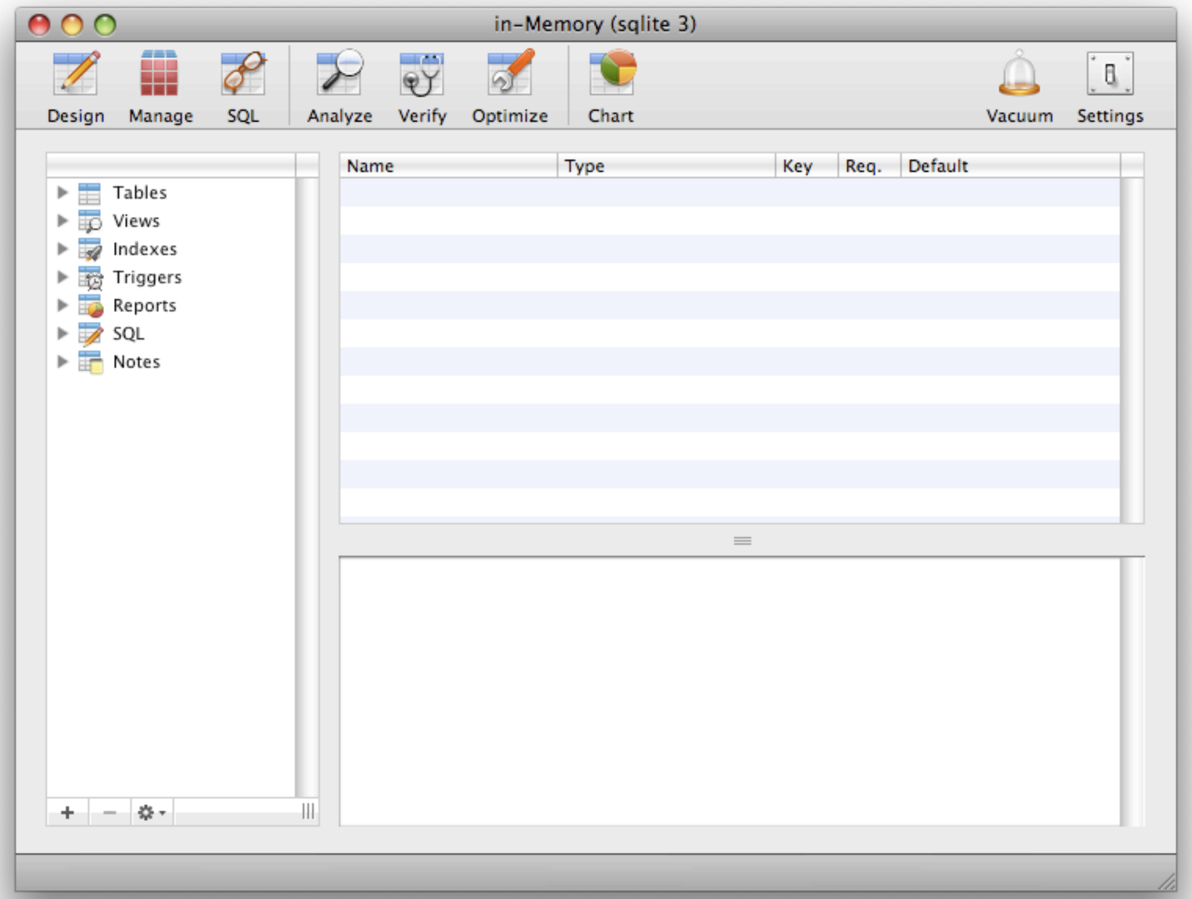
Select the "Register" icon, shown above, to register your copy of SQLiteManager. Until you register, SQLiteManager runs with a lot of limitations. To register, enter your serial number in the "Serial Number" field. The Company and Name fields are optional. Press the button "Check SN" if you want to check if your SN is valid and if it can unlocks all the SQLiteManager's features.



Type	Name
Database	images.sqlite
Database	skyline.rsd
Database	testdb.sqlite
Database	jazzfestvan09.sqlite

In Memory Databases

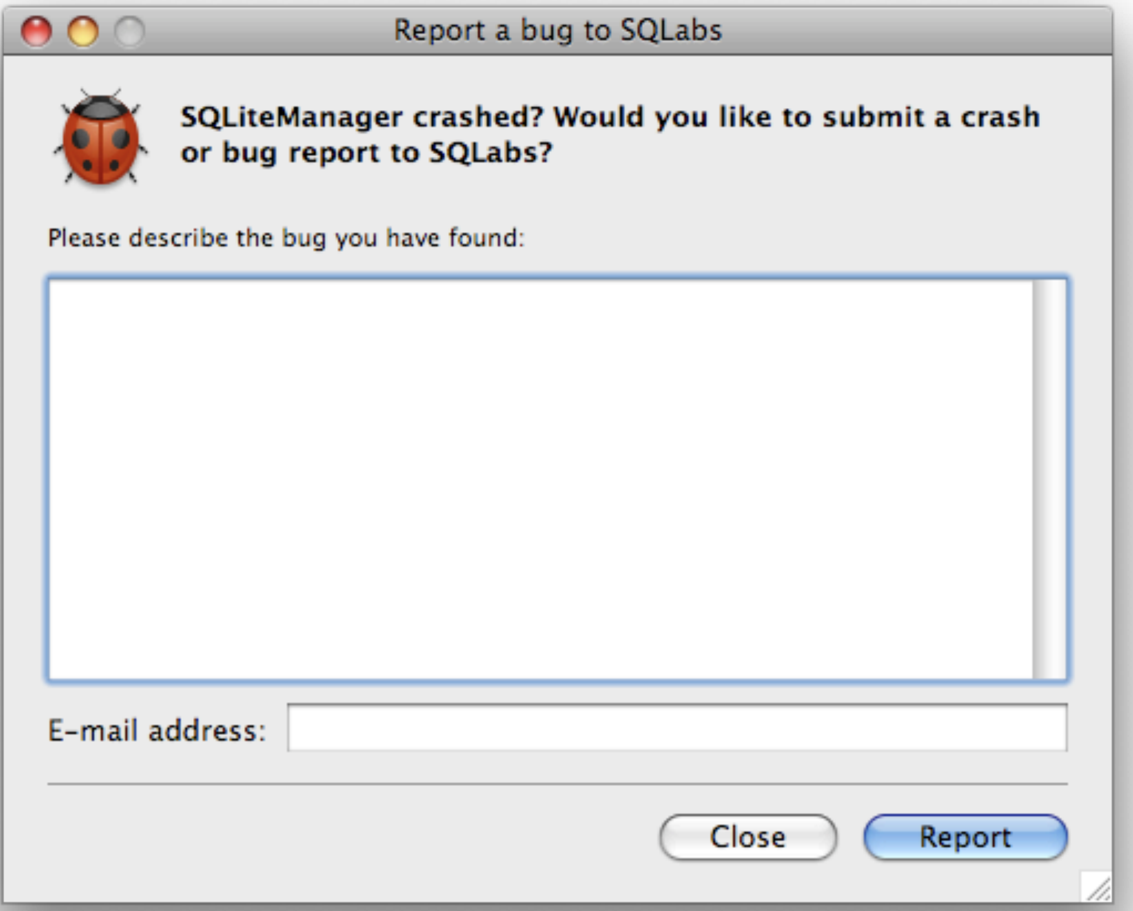
SQLiteManager supports creation of In-Memory database (from the startup Wizard or from the New submenu in the File menu). In-Memory databases can be very useful if you need to test your ideas, your SQL or just to have the speed of RAM based databases.



Once you finished working with In-Memory databases you can choose to discard it or you can choose to dump the entire database (with its structure and data) to file. To do this just select “Dump database” from the File menu.

Bug Reporter

SQLiteManager has a built-in bug reporter, just select “Report a bug to SQLabs” from the Help menu in order to have the Bug Reporter dialog. Note that this dialog will automatically appears each time a crash bug will occurs in the application.


Report a bug to SQLabs

SQLiteManager crashed? Would you like to submit a crash or bug report to SQLabs?

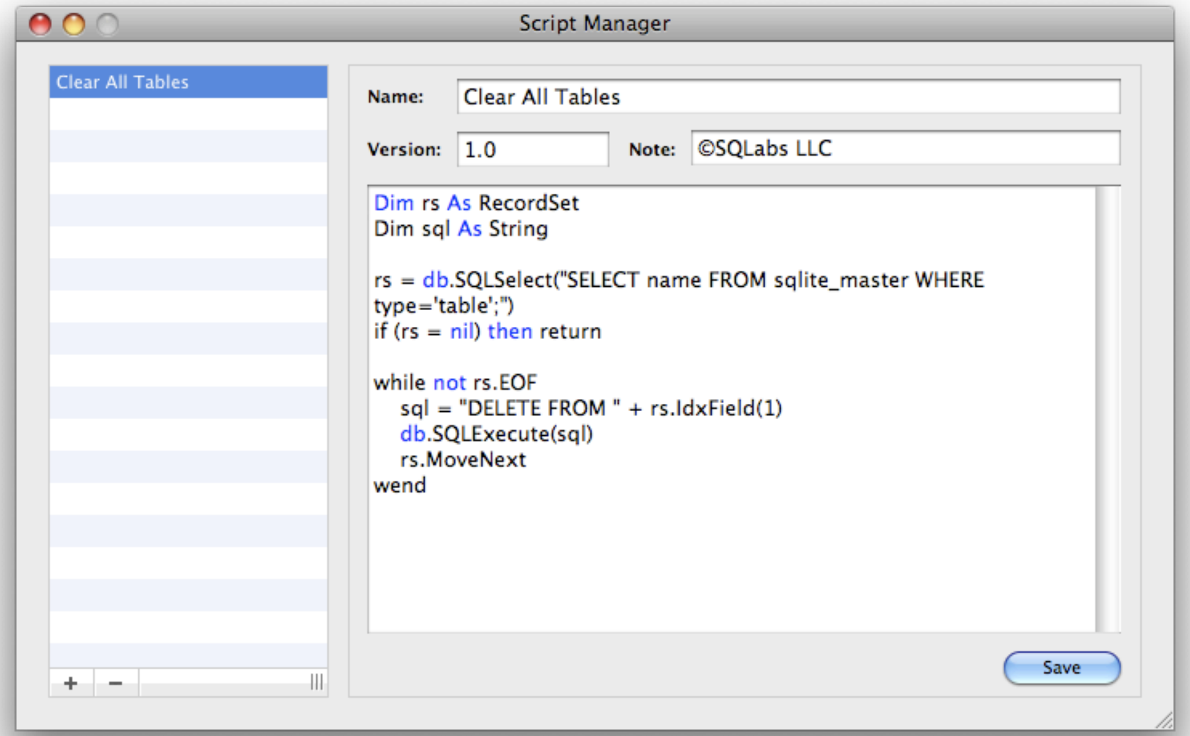
Please describe the bug you have found:

E-mail address:

The bug report will be automatically filled with important information necessary to SQLabs to better track down the bug. No sensible information like your name or you serial number will be sent with the bug/crash report.

Script Manager

A built-in scripting language based on RBScript (a Basic like scripting language) enables you to write plugins and to automate repetitive tasks inside SQLiteManager. You can also share your plugins with others SQLiteManager's users. Please note that it also supports dynamic loading of native sqlite extensions.



Name: Clear All Tables

Version: 1.0

Note: ©SQLabs LLC

```
Dim rs As RecordSet
Dim sql As String

rs = db.SQLSelect("SELECT name FROM sqlite_master WHERE
type='table';")
if (rs = nil) then return

while not rs.EOF
   sql = "DELETE FROM " + rs.IdxField(1)
   db.SQLExecute(sql)
   rs.MoveNext
wend
```

Appendix A

Contact Information

SQLabs LLC
marco@sqlabs.com
Web: http://www.sqlabs.com
Email: info@sqlabs.com

Copyright

All materials are copyright 2003-2010 by SQLabs.

All Rights Reserved. SQLiteManager may be freely distributed, so long as it is not sold for profit, and registration serial numbers are not distributed. Express permission is granted to online services and other shareware/public domain distribution avenues to carry SQLiteManager.
Express permission is further granted to include SQLiteManager on CD-ROMs or floppy disks accompanying books, or on shareware collections, provided that no more than a nominal compilation and/or media fee is charged for these collections.

Legal Stuff

Unregistered copies may be used and evaluated in demonstration mode. Copies are registered per individual developer and may be used by that developer, royalty-free, in an unlimited number of applications, commercial or otherwise. Once obtained, licenses may not be transferred to other individuals or organizations.
SQLabs reserves the right to revoke the license of anyone who ignores or violates these restrictions. See our web site at http://www.sqlabs.com/ for registration information. Company licenses are available.
SQLiteManager is distributed AS IS. There is no warranty of any kind as to its fitness for any purpose. The risks associated with the use of this product are borne by the user in their entirety. In other words, the SQLiteManager, although it is in no way designed to do so, could be capable of ruining your software, crashing your computer and erasing your hard drive. Marco Bambini and SQLabs take no responsibility for these or other consequences.
REALbasic® it is a registered trademark of REAL Software, Inc. See their web site at http://www.realsoftware.com for more details. SQLabs is not way affiliated with REAL Software. All questions regarding REALbasic should be directed to REAL Software, Inc.