The Java SQLite3 Database Editor

Sidney Marshall

October 17, 2014 (updated November 22, 2024)

Contents

Abstract

1 Introduction - The SQLite3 Database Editor

The Java Database editor is a Java jar file program that together with the sqlite-jdbc jar file edits SQLite3 databases. It runs under windows, linux, and macs. Originally the code was based on some work of RIT students but the current code has diverged from this original code base.

The current editor can run with any SQLite3 version (after 3.7.9) by pointing the class path to the desired jdbc version. Currently, the class path is set to `sqlite-jdbc-3.27.2.jar` although any recent SQLite3 jdbc version can be used. There are some new SQL commands available in the later versions that are not available in the earlier versions.

All of the DatabaseEditor source code is available in the `dbe.jar` file in the file `DatabaseEditor.java`. The interface source code, class files, and native libraries for various operating systems are in the `sqlite-jdbc-xxx.jar` file. You need both the `dbe.jar` and `sqlite-jdbc-xxx.jar` files to run the editor.

2 Tools Needed To Run the Database Editor

You need a Java from `java.sun.com` or `www.oracle.com/technetwork/java`. I use Java8. If you do not want to recompile the editor you only need the JRE. If you do want to recompile the editor you will need the SDK.

3 Using the Database Editor

The editor can be run from the command line or by double-clicking the `dbe.jar` icon. The command line invocation is:

```
java -jar dbe.jar [database [database char-set [file char-set]]
```

If the optional `database` argument is not given or if the editor is run by clicking on its icon a file browser will appear to choose the database to open. If the `database` is given on the command line then the named database is opened. If the database name given does not exist then an empty database will be created.

All internal strings use the Java String character set. When reading or writing the database these strings are converted to or from byte strings using the `database char-set`. When reading or writing files these strings are converted to or from file bytes using the `file char-set`.

If the optional `char-set` or the optional `file char-set` argument is not given then the default UTF-8 character set is used. Otherwise an attempt will be made to use the specified character set encoding. Lines are always terminated with a new-line byte and not a platform-specific end-of-line character.

The windows available in the database editor are:

Database Chooser	Chooses which database to operate on
Database Window	Displays the tables and views in the database
Table Window	Displays the rows and columns of a table or view
Result Table Window	Displays the rows and columns of a result set
Edit Window	An editable text window for editing the contents of a table cell
Command Window	An editable text window allowing the execution of SQL commands
Find-Replace Window	A control window for text searching and replacing in an Edit Window or Command Window
Column Search Window	A control window for searching a table column

The contents of all editable text windows can be copied or pasted to or from the clipboard. The control characters for all editable windows are:

cntrl-X	Cuts the selected text and copies it to the clipboard
cntrl-C	Copies the selected text to the clipboard
cntrl-V	Pastes the text from the clipboard replacing the selected text
cntrl-A	Selects all of the text
DEL	Deletes the selected text or the character following the cursor if nothing is selected

In addition, multiline edit windows support:

cntrl-Z	Undoes the last action
cntrl-Y	Redoes the last action

Multiple redos or undoes can be done.

When a window is closed all dependent windows are also closed. In particular, when a Database Window is closed all windows associated with this database connection are closed. Here is a list of what gets closed:

Database Window	All windows associated with this database connection
Table Window	All Edit Windows editing a cell of the table or view and all Column Search windows

Result Table Window	All Column Search windows
Edit Window	Find Replace windows on the Edit window
Command Window	Find Replace windows on the Command window
Find-Replace Window	Nothing else gets closed
Column Search Window	Nothing else gets closed

3.1 The Database Chooser

Chooses the database file to operate on. If several databases are opened there is no connection between them and SQL statements cannot reference tables from multiple databases. You can `attach` a database (using a Command Window) and SQL statements can then be used to reference multiple databases.

3.2 The Database Window

When a database is opened a Database Window will appear listing all of the tables and views in all of the databases currently attached. Views have a background of yellow. If the name of a table appears in multiple databases then the name of the datamase (`main`, `temp`, or attached name) is displayed before the table name for disambiguation. Hovering the mouse over an entry will display the full name, i.e., database name and table/view name.

Left clicking on the name of a table or view will bring up a Table Window displaying the contents of this table or view. All of the tables and views in the `temp` database and attached databases are also listed.

Right clicking on the name of a table or view additionally brings up a Table Window displaying the contents of the table or view. It also brings up a Table Search Window that can search the contents of a table column.

In most cases there will be only one Table Window opened on a particular table or view. If an attempt to open another Table Window on the same table or view the previously opened Table Window will be selected and brought to the front.

Below the list of tables and views are four buttons:

Refresh	Refreshes the list of tables and views in case any have been added or removed
Show Hidden Tables	Displays all tables and views in the database with database names including internal SQLite tables
Tables	Bring up a table of database tables with columns ‘name’, and ‘sql’

Triggers	Bring up a table of database triggers with columns 'tbl_name', 'name', and 'sql'
Open Command Window	Opens an empty Command Window which allows executing SQL commands
Open Database	Opens the same or different database in a separate process
Read Database	Reads and executes SQL commands from a file
Write Database	Writes the entire database as SQL statements

Note that if you create or drop tables or views the list of tables and views will not be updated until you click `Refresh`.

If you click `Show Hidden Tables` then all tables are displayed including tables that begin with `sqlite_` used by the SQLite database itself. Do not edit these tables unless you *really* know what you are doing.

If you open the same or another database using the `Open Database` button the two database connections do not share any state, in particular, `temp` tables will not be shared between database connections. If you `attach` a database in a Command Window then you can access both databases simultaneously, e.g., in an SQL statement.

The `Read Database` button will bring up a File Chooser. Selecting a file will read and execute SQL statements from the selected file. If the database is empty and the file was created with the `Write Database` button then the database will be restored. This button will also read arbitrary SQL statements from a file and execute them. For all statements that generate a table (e.g., `select` statements) a result table is displayed. This can be used to open several Table Windows and do other database maintenance automatically. **Warning: If the commands read from the file open a transaction and an error occurs, the transaction is still pending; you will lose all subsequent work unless you close the pending transaction.**

The `Write Database` button will output the entire database to a file as SQL statements that, when read into an empty database, will restore the database. You can also use an editor on this file to extract pieces of the database.

3.3 The Table Window

The Table Window displays the contents of a table, view, or the result of a `select` statement or executed command that returns a result set. The labels at the top of the table indicate the database column names. You can resize columns by dragging the boundaries of the labels. You can move columns around by dragging a label to where you want the column. Neither of these operations has any effect on the actual database itself.

All database tables are assumed to contain strings (or `NULL`). SQLite does not enforce strict typing and will convert other types to strings. `BLOB`s containing

binary information are not specifically handled by this editor although they can be created by executing SQL statements. Read the section on type affinities in the sqlite documentation for more information (sqlite.org ⇒ SQL Syntax ⇒ expression ⇒ scroll to CAST expressions ⇒ rules for determining column affinity).

When the Table Window is first initialized it reads the contents of the table from the database and caches this contents in a "shadow" table in memory. It also initializes another "backing" table with the same contents. Whenever the Table Window is refreshed it uses the backing table. Table entries that differ between the shadow and backing tables are colored red to indicate a changed table entry. Editing a table entry changes the backing table but does not change the shadow table. Only the buttons "Delete Selected Rows" and "Update Table" refresh the shadow copy from the database after performing their other operations. The button "Delete Selected Rows" also forgets any pending updates to the table (this is probably a bug but somewhat hard to fix).

The contents of a table entry may be edited by selecting the entry and editing it in place. You can drag table entries to other entries and the dragged contents will replace the previous contents of the entry. You can also drag file names from file browser windows although only a single file can be dragged at a time.

Right clicking on any cell of the table will bring up an Edit Window initialized with the contents of the cell. This is useful if the text is long or contains multiple lines. The in-place editor does not handle multiple lines of text. Right clicking on a cell in a newly created table row will bring up an editor on the cell but the update button is missing in the editor (as the cell does not have a real row in the table until that row is saved to the table as the editor would be confused as to which cell it was editing).

Right clicking on a column title will bring up a Table Search Window that can search the contents of an entire table column.

Drag-and-drop works between table cells of the same or different tables. You can also drag-and-drop single files from a file window to a cell and the file name (without directory information) will be copied to the cell. Copy and paste also work.

Updates to a view or result set (created by executing a select statement in a command window) are not permitted (since views and result sets cannot be modified) unless `instead of` triggers are defined on a view. Also **********

There are two varieties of Table Windows: those on tables and views and those on a result set generated by executing a `select` statement.

3.3.1 The Table Window on Tables and Views

Editing a table entry does not affect the database table until one of the update buttons is clicked. Entries that differ from the value fetched from the table are colored red to indicate that the entry has not been updated. The buttons at the bottom of the Table Window on a table or view are:

Delete Selected Rows Deletes the selected row

Make NULL	Makes the selected cell `NULL`
Revert Selected	Reverts all entries in the row with the selection to their initial values
Revert Table	Reverts all entries in the table to their initial values
Add Row	Adds a new row to the table with all entries `NULL`
Fill Column	See below
Update Selected	Updates the database with the changed entries in the row with the selection
Update Table	Updates the database with all changed entries in the entire table and then refreshes the table from the database

The `Fill Column` button fills contiguous `NULL` cells in a column. The filling will proceed until a non-`NULL` cell is encountered or the end of the column is reached.

If the selected cell contains a `NULL` and the cell above it is non-`NULL` or the selected cell is non-`NULL` then the `NULL` cells are filled with a copy of the non-`NULL` cell.

If the selected cell is non-`NULL` and the followung cell is non-`NULL` and the second following cell is `NULL` and the two non-`NULL` cells contain strings of the same length and differ in only one character then the following `NULL` cells are filled with strings in an arithmetical progression for the differing character in the strings.

In the following examples the selected cell is in brackets. The examples are layed out horizontaly but they represent a vertical set of cells.

Clicking on `Fill Column` in the following examples gives the indicated results.

[xab], null, null, null, abc ⇒ xab, xab, xab, xab, abc
xab, [null], null, null, abc ⇒ xab, xab, xab, xab, abc
[xab], xbb, null, null, abc ⇒ xab, xbb, xcb, xdb, abc

3.3.2 The Table Window on Result Sets

Result sets are not editable (although the display of a result set can be edited). The bottom of a Table Window on a result set resulting from executing a select statement in a Command Window contains:

Write File	Writes a file containing an image of the result set.
Text line	The name of the file to write defaults to `data.txt`.
Browse button	Brings up a file browser to select the file to be written.
Escape Strings	Check to escape cell strings.

Column Separator	Text separating cells in a row — defaults to a comma character.
Row End	Text terminating rows — defaults to the new-line character.
null	Text for outputting nulls — defaults to `null`.

Clicking on the `Write File` button writes the named file with the contents of the result set. Cells in a row are separated by the `Row Separator` field and each row is terminated by the contents of the `ColunmEnd` field. Cells with `null` are output with the contents of the `null` field.

The `Browse` button brings up a file browser to select the file to be written. The `set` button sets the file to be written but does not actually write any files.

If the `Escape Strings` box is checked then all of the cells are output with quotes (′) around them and any quote characters are replaced with double quotes. This does not affect `null` fields.

3.4 The Edit Window

This is an editable text window that is tied to a particular cell in a Table Window. You can perform any edits here and copy and paste between windows. The buttons at the bottom of the Edit Window are:

Open in Command Window	Opens a Command Window with the contents of this window
Break Lines in Selection	Replaces selection with lines broken at no more than 70 characters - preserves double newlines
Update	Updates the table cell spawning this window with this window's contents

Note that the database is not updated when the `Update` button is clicked. An update button on the Table Window must also be clicked to update the database.

The `Open in Command Window` button was added to conveniently execute SQL statements that are stored in the database itself.

3.5 The Command Window

The Command Window is an editable text window that allows SQL statements to be executed. There are six buttons at the bottom:

Execute Selection	Executes the selection in the window as an SQL statement
Select Next Command	Selects the next sql statement following the current selection

Break Lines in Selection	Replaces selection with lines broken at no more than 70 characters - preserves double newlines
Print	Send the contents of the window to a printer
Read File	Loads a file into the Command Window for editing
Write File	Saves the contents of the Command Window into a file

If the button is clicked and there is no selection then the entire window contents is executed as an SQL statement.

Because the JDBC interface to SQLite3 only executes a single statement only one statement at a time can be executed.

After the statement is executed any errors (Java exceptions) are noted in the status line at the bottom of the window. If there are no errors then the update count is displayed in the status line if it is not negative. Otherwise, the statement returned a result set and a Table Window is created displaying the contents of this result set. This result set is not editable but its cell contents can be copied or brought up in an Edit Window.

The `Read File` and `Write File` buttons allow the Command Window to be used as a text editor.

3.6 The Find-Replace Window

Right clicking in an Edit Window or a Command Window brings up a Find-Replace Window. There are two editable fields (find and replace) and four buttons in the window:

Find	Finds and selects the next occurrence of the text in the adjacent find field
Find Funny	Finds and selects the next "funny" character - a funny character is any character that is not 7-bit ASCII or a new-line
Replace	Replaces the selected text with the contents of the adjacent replace field
Replace/Find	Does a replace operation followed by a find operation

The `Replace` operations modify the contents of the associated Edit Window but the database will not be affected until the Edit Window update is clicked and an `Update` button on the Table Window is clicked.

If a `Find` or `Find Funny` search does not find any further matches then the selection is set to the beginning of the file and the status line displays `not found`. This means the search will start at the beginning, i.e., the search will start at the beginning the next time a search is attempted.

The status line also contains a check box on the right labelled `Ignore Case`. If checked, the search is done without requiring that the case of the search string maches. If unchecked, the search string must match exactly, including case.

3.7 The Column Search Window

Right clicking on a table in the The Table Search window or right clicking on a table column label brings up a Column Search Window. A Column Search Window enables searching the contents of an entire column of a table or view. There are four buttons in the window:

Find Finds the next cell in the specified column after the selected cell containing the search string and brings up an Edit Window on the cell's contents with an associated Find-Replace Window.

Find Funny Finds the next cell in the specified column after the selected cell with a "funny" character and brings up an Edit Window on the cell's contents with an associated Find-Replace Window - a funny character is any character that is not ASCII or a new-line

To Hex Replaces the contents of the replace field with a hex representation of its contents

From Hex Replaces the contents of the replace field with a string having this hex representation

Before performing a column search the column field must be initialized to a valid column designation. Right clicking on a table header will do this automatically. Right clicking on a table or view in a Database Editor Window initializes this designation to the first column in the table. Using the drop-down menu you can select any column. You can change the column designation at any time.

The table field is for your information only and indicates which table is being searched. Its contents are not editable.

If there is no cell selected the search starts at the first row. Otherwise it starts in the row following the row with the selection.

When a `Find` or `Find Funny` operation finds a matching cell it selects it and brings up an Edit Window on this cell and a Find-Replace window on this Edit Window. Nothing is selected in the Edit Window and the find and replace fields in the Find-Replace window are initialized from the corresponding fields of the Column Search window. You must do your own search or other editing in the cell contents Edit window.

The `To Hex` and `From Hex` buttons are for manipulating text by their numerical value. The results can be copied and pasted to other windows. The replace field should be set to a replacement string if a replace operation is desired.

There is also a check box labelled `Ignore Case` on the status line. If checked, the search is done without requiring that the case of the search string matches. If unchecked, the search string must match exactly, including case.

4 Compiling The Database Editor

To recompile the `dbe.jar` file you need the following files:

DatabaseEditor.java	The source for the Database editor
manifest.txt	A file with a manifest with a class path pointing to the current jdbc database `.jar` file
sqlite-jdbc-XXX.jar	The jdbc file implementing the Java interface

You can make the `manifest.txt` file with a single line using any text editor with the contents (for example):

```
Class-Path: sqlite-jdbc-3.16.1.jar
```

Then do the following:

```
rm DatabaseEditor.class DatabaseEditor\$*.class
javac -target 1.8 -source 1.8 DatabaseEditor.java
jar cfme dbe.jar manifest.txt DatabaseEditor DatabaseEditor*.class DatabaseEditor.java
rm DatabaseEditor.class DatabaseEditor\$*.class
```

Here is my Makefile section:

```
dbe.jar: DatabaseEditor.java manifest.txt Makefile
touch DatabaseEditor.class
-rm DatabaseEditor*.class
javac -target 1.8 -source 1.8 DatabaseEditor.java
jar cfme dbe.jar manifest.txt DatabaseEditor DatabaseEditor*.class DatabaseEditor.java
rm DatabaseEditor*.class
```

4.1 SQLite3

SQLite3 is a free relational database system. All of the tables of a database are stored in a single file making backup as simple as copying this file. It can be downloaded from www.sqlite.org by following directions there. I am using version 3.16.1 although most versions are compatible with each other. I compiled the sqlite-amalgamation version which also contains a shell driver for executing SQL commands. The database editor can also be made with SQLite3 version 3.7.9 with reduced functionality.

You do not need to compile SQLite3 to maintain a database. Command Windows in the Java Database editor are all you need. But if you want a `.c` program or want to use `C` programs to interact with a database you can do the following.

To compile the SQLite3 program use the command

```
gcc shell.c sqlite3.c -o sqlite3
```

This compiles the files `shell.c`, `sqlite3.c`, `sqlite3.h`, and `sqlite3ext.h` and makes an `sqlite3` executable. Any ANSI C compiler can be used.

You can also use the shell driver `sqlite3` for examining or modifying databases. You run it (after appropriately compiling it) by typing

```
./sqlite3 xxx.db
```

4.2 Java / sqlite-jdbc

The sqlitejdbc project has free sqlite3 interfaces for Java. I used this interface to write Java programs to display and update SQLite3 tables and views. This is the program I use to enter and update data for the web site. Programs can execute raw SQL commands for changing the structure of the database or you can use other tools (like the SQLite shell) for this. The jar file `dbe.jar` is a self-executing file and can be run (in some environments) by double-clicking on it or running the command `java -jar dbe.jar`. You can also specify a database to be opened with the command `java -jar dbe.jar squaredance.db`. You must have a `sqlite-jdbc-XXX.jar` file in the same directory as this is the interface between Java and SQLite3. Various versions of the `sqlite-jdbc-XXX.jar` file are available from `https://bitbucket.org/xerial/sqlite-jdbc/downloads`

If the appropriate triggers are written you can also update views. Check the corresponding trigger to see what you should enter. It may not be what you expect.

4.3 Java

The tools work under Java8. A `sqlite-jdbc-XXX.jar` file is needed to supply the interface between Java and sqlite3. Included in this .jar file are native libraries that speed up the interface. The supplied `Makefile` has a target for `dbe.jar` that will compile `DatabaseEditor.java` and create a database editor `.jar` file with the correct manifest.

5 Bugs or Things To Fix

Sometimes edit windows created off-screen.
Break lines needs work.