HAUFE.Gruppe

MultiSQLite
C# Edition

User's Guide

Version 1.0.0.0

HAUFE.Group

uwe.stahlschmidt@haufe-lexware.com

HAUFE.Gruppe

Table of Contents

Introduction

Haufe MultiSQLite is an external tool that can be used independent of the on-premises applications of the Haufe Group in order to evaluate the feasibility of using SQLite in the different situations arising from the environments surrounding the products. This is done by simulating the existing cases which are dominant throughout the development teams:

- Accessing the same SQLite database from different threads within the same application
- Using one SQLite database from different applications that are developed in the same language (C# or C++)
- Using SQLite for joint products that are using both C++ and C# with different connectors for each
- Having different applications that are developed with different development tools access that same database.

MultiSQLite is available as an open source GitHub project, so that developers can look up how the access to the database is done and hence can expect similar results if the connection is done in a similar way in the corresponding applications.

As of this moment, the tool is available in two different editions:

In addition to this edition, which is implemented in C#, a very similar version is also available for C++. The different editions are designed to work along with each other, so that analysis can also be undertaken in mixed environments or for products, that are developed using a combination of C# and C++. For that

purpose, the different editions can be started separately. Since it is insured that the database is located in a predefined directory, analysis in mixed C++ / C# environments if possible without any further ado, however it must be ensured that compatible editions of MultiSQLite are used in a mixed setup.

Please note that the editions are developed based on the needs at hands and hence are not necessarily in sync. For that reason, the functionality and user interface of the different editions may vary depending the current state of development.

Installation

As was already mentioned above, Haufe MultiSQLite is maintained as a GitHub-Project in order to allow further development by different developers.

The GitHub-Project is hosted under the following link:

https://github.com/ushaufe/Sqlite4CS.git

The executable version of the product as well as the source-files are available in the subfolders of this file structure.

Note:
Currently (Version 1.0.0.0) of MultiSQLite is not bundled with an installer. For that reason the installation must be done by downloading and extracting the corresponding files manually as explained in the following sections.

Step 1: Open GitHub-Project Site

As a first step, the MuliSQLite archive must be downloaded from the GitHub-Site. In a webbrowser of your choice navigate to the following link hosting the project: https://github.com/ushaufe/Sqlite4CS.git

HAUFE.Gruppe

This site lists the directory structure including all files of the project. The file structure can be browsed to have access to single files.

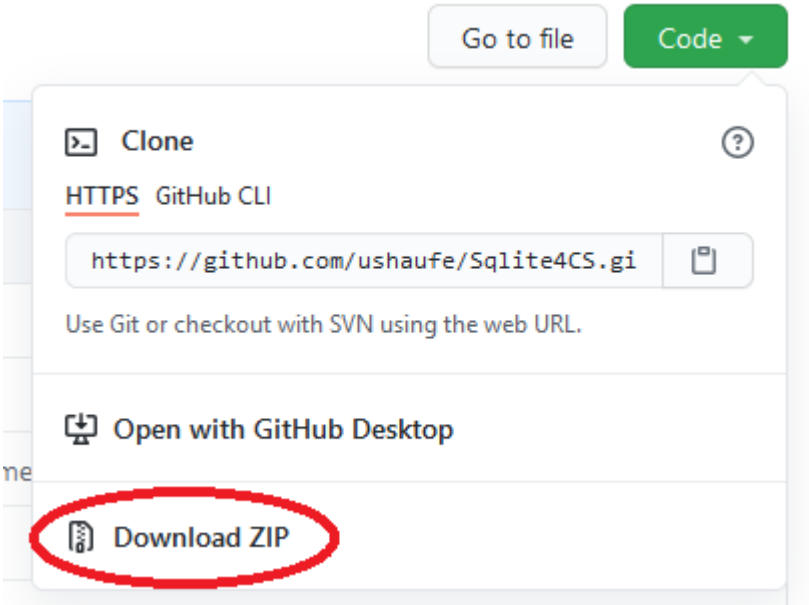
Step 2: Download the project archive

In order to have a locally executable version of MultiSQlite, the whole project must be downloaded as an archive.

In order to achive that, locate the button “Code” in your GitHub-project and click on it in order to expand it.

Once the section grouped under the icon “Code” is expanded, the user is given a choice of options.

In that expanded dialog click on “Download ZIP” and save the zip folder on your local harddrive.

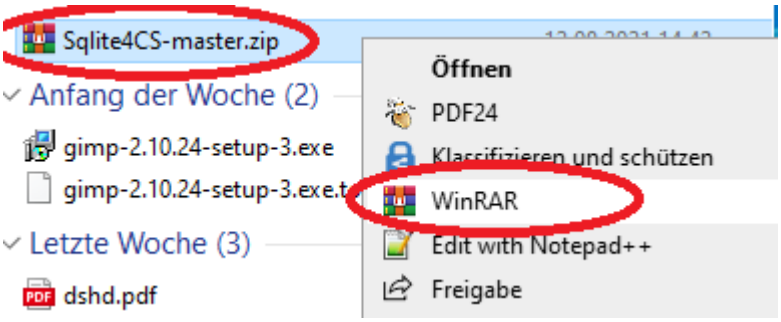


This file contains the source-code as well as the executable application in a compressed form.

Note:
If you are a developer that is already actively using Git, you might have tools with a graphical user interface installed (i.e. Tortoise-Git). In that case you can also use those tools to download the project. This will enable you to commit changes.

Step 3: Extracting the Archive

HAUFE.Gruppe



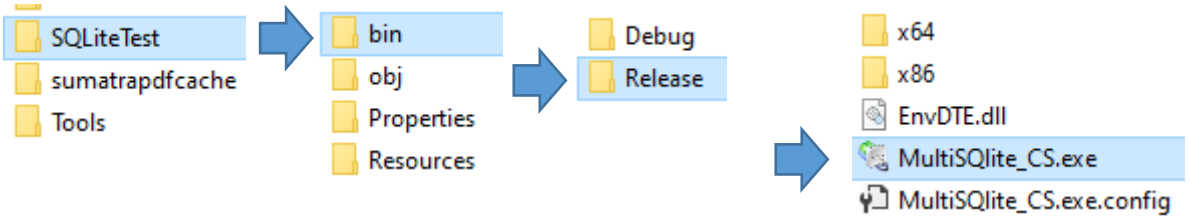
Once downloaded, extract the file "SQlite4CS-master.zip" to a folder of your choosing.

This will create a file-structure in the underlying directory that looks like this

Name	Größe	Gepackt	Typ	Geändert	CRC32
..			Dateiordner		
.vs	6.025.892	504.925	Dateiordner	11.08.2021 18:25	
packages	32.493.968	19.140.079	Dateiordner	11.08.2021 18:25	
res	47.389	40.376	Dateiordner	11.08.2021 18:25	
SQLiteTest	22.182.230	11.778.787	Dateiordner	11.08.2021 18:25	
sumatrapdfcache	7.919	7.919	Dateiordner	11.08.2021 18:25	
Tools	6.434.995	3.711.202	Dateiordner	11.08.2021 18:25	
1920px-SQLite370.svg...	76.725	71.987	IrfanView PNG File	11.08.2021 18:25	982C7660
Connection2.jpg	170.781	170.077	IrfanView JPG File	11.08.2021 18:25	205CC02A
Connection2_Light1.jpg	397.543	392.883	IrfanView JPG File	11.08.2021 18:25	597A2FA0
Connection2_Light2.jpg	325.204	321.708	IrfanView JPG File	11.08.2021 18:25	A1C9DD29
ConnectionIcon.ico	67.646	21.321	IrfanView ICO File	11.08.2021 18:25	138A1ADE
ConnectionIcon.png	19.192	19.192	IrfanView PNG File	11.08.2021 18:25	25D1A485
Connections.ico	67.646	16.969	IrfanView ICO File	11.08.2021 18:25	961B7F45
Connections.png	15.524	15.524	IrfanView PNG File	11.08.2021 18:25	205C75D4
Connections2.jpg	376.609	358.847	IrfanView JPG File	11.08.2021 18:25	D90EFA9D
Connections2.png	2.105.669	2.105.669	IrfanView PNG File	11.08.2021 18:25	D506A41B
Connections2_Light.jpg	256.566	252.316	IrfanView JPG File	11.08.2021 18:25	DCF0585C

Step 4: Finding and starting the application

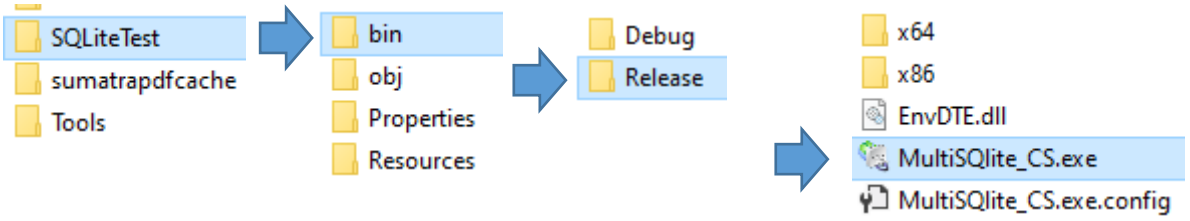
The file structure that was extracted contains the executable application in addition to the source files. In order to start the application, move to the following subfolder and start the following application:



HAUFE.Gruppe

Starting the Application

As mentioned in the above chapter, your application has been extracted to the following subfolder of the file structure that you have created.



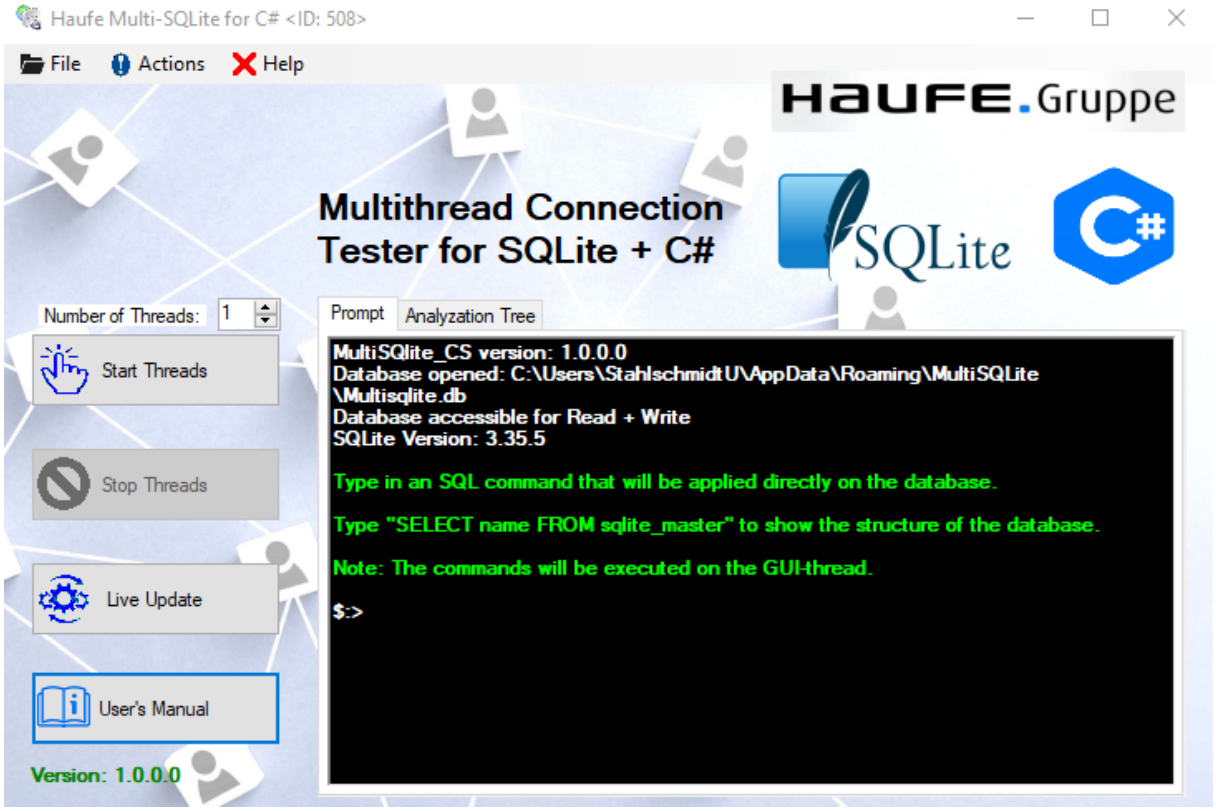
The application can be started directly by launching the executable “MultiSQlite_CS.exe” in the “Release”-subfolder. If the Debug-folder also contains an executable version of that file, it is preferrable to start it from the “Release”-subfolder because this should contain the version with a better performance.

Note:
The application will only launch and work correctly if in addition to the executable, all additional files have been extracted properly and are included in the “Release”-folder. The executable of the application will not launch properly without the additional files.

Once the application is started, it automatically sets up a database that is used for testing. If the application is restarted from a previous session, the same database is reused.

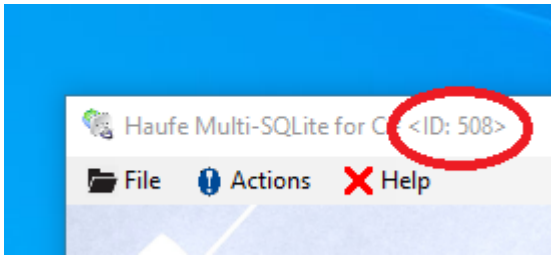
In case multiple instances of the application are started, they all use the same database and simulate the sharing of a database by several applications. No further action has to be taken by the user to indicate the database or other running instances of the application.

Once the application is started the user is presented with the following GUI:



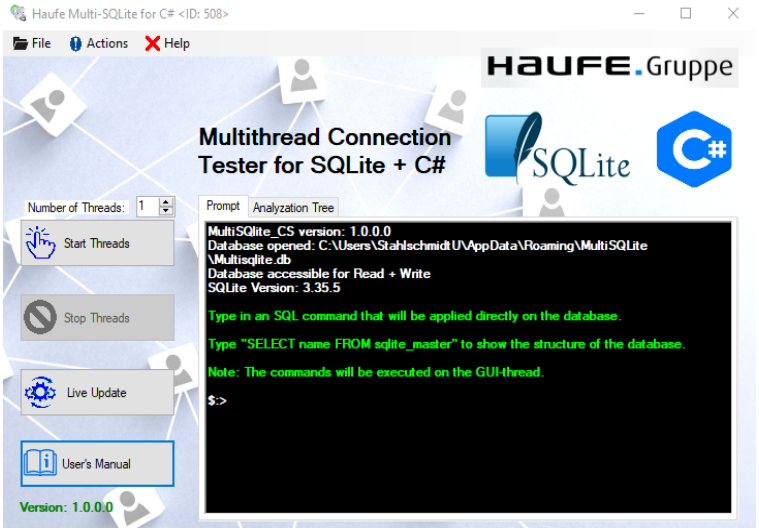
In the prompt window that is shown during startup the user can see if the connection to the database was successful. By default the application opens one connection to the database from now on referred to as the GUI-Connection.

In the title bar of the application the instance of the running application is indicated, so that several instances of the same application can be distinguished.



HAUFE.Gruppe

User Interface



The main screen of the user interface consists of two areas:
A row of buttons on the left side to perform different actions
A tabbed window on the right side allowing the user to interact with the database.

The functionality of each element will be described in the following subsections.

Starting and stopping threads

With the two buttons and the number control on the left side the user can start and stop threads that serve to continually insert datasets into the database.

With this number control the user can set how many threads should insert data in the database simultaneously. Each thread is inserting the data as fast as it can on an individual connection.

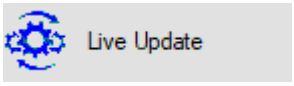
When the threads are started it serves as an indicator showing the number of threads currently running, when the threads are stopped it allows the user to select the number of threads that should be started.

Clicking this button starts the number of threads indicated in the number control. The corresponding number of threads is created, each thread owning its individual connection. Each of the threads grabs as

much capacity as it can and tries to insert as many datasets as possible into the database. The bandwidth is concurrently shared among the threads. As soon as the inserting-process is started, the button is disabled, and the “Stop Threads” button is enabled. By pushing this button, the view on the right side of the window is switched to tree view.

This button terminates the threads currently active and stops the inserting-process. After the running threads have been stopped, only the GUI-process remains open allowing the user to access on a step-by-step base. The button becomes disabled as soon as the threads are terminated and toggles the “Start Threads” Button to enabled.

Continuous Monitoring



By enabling “Live Update” the tree view on the left right side is continually updated and the corresponding elements in the tree are showing live values.

The button toggles to gray when the automatic polling is activated and toggles back into the default state if pressed again, hence indicating that the automatic polling has been disabled.

As with the preceding buttons, the view on the right side of the window is automatically switched to tree view.

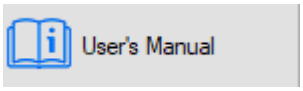
Note
This feature is switched off by default and should be used with caution. Depending on the tree elements that are currently opened on the right side, enabling this feature uses a lot of performance and puts a lot of load on the GUI-thread.

In addition, the button is flickering each time the information in the tree view is updated.

Version Label and Alive-Signal

In addition to showing the current version of the application, the version label serves the purpose of giving an alive signal to the user: The database is polled at given intervals and status information about the access-levels is updated in the database. Every time this happens, the version label flickers in red.

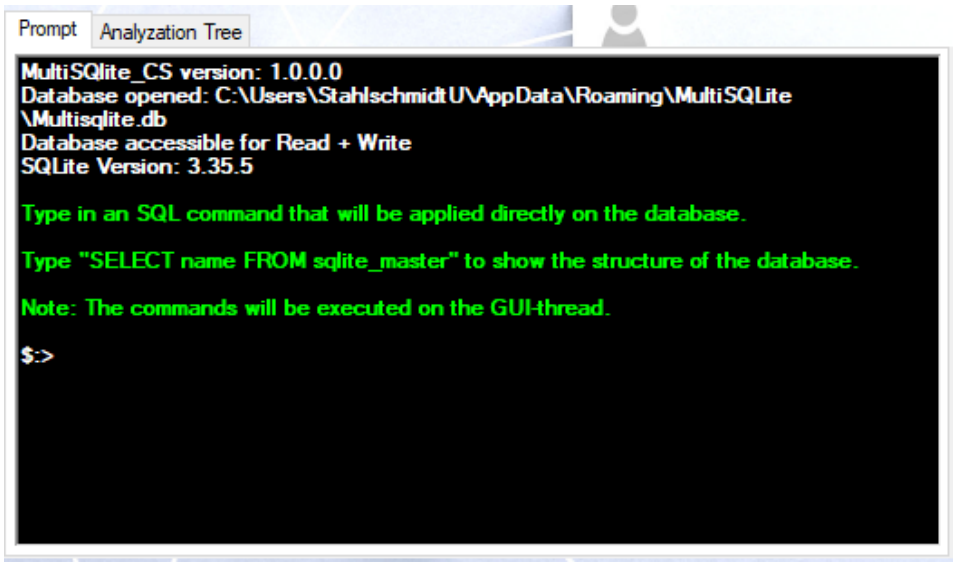
Showing User's manual



Pushing this button brings up this user's manual. Note that that this button will bring up the latest version of the manual, which might not necessarily correspond with the version of the application.

The prompt: Direct access to the database

The right side of the application houses a tabsheet, with two tab-views. The tab "Prompt" gives the user direct access to the database, allowing him to execute SQL-commands in plaintext form.



Note

The commands typed in this prompt are always executed on the GUI-threads, hence not affecting the processes under way in the other threads. Because of that fact, SQL-commands putting heavy load on either the database or the output-shell will freeze the GUI for the time the command is executed.

The Analyzation Tree

If the tab “Analyzation Tree” is opened, the user is shown a tree providing the different applications currently accessing the database along with their corresponding threads and measuring data such as throughput for each of the threads and applications.