xeus-sqlite

Mariana Meireles

May 13, 2020

INSTALLATION

`xeus-sqlite` is a Jupyter kernel for SQLite based on the native implementation of the Jupyter protocol xeus.

CHAPTER

ONE

LICENSING

We use a shared copyright model that enables all contributors to maintain the copyright on their contributions.

This software is licensed under the BSD-3-Clause license. See the LICENSE file for details.

1.1 Installation

1.1.1 With Conda

xeus-sqlite has been packaged for the conda package manager.

To ensure that the installation works, it is preferable to install *xeus-sqlite* in a fresh conda environment. It is also needed to use a miniconda installation because with the full anaconda you may have a conflict with the *zeromq* library which is already installed in the anaconda distribution.

The safest usage is to create an environment named *xeus-sqlite* with your miniconda installation

```
conda create -n xeus-sqlite
conda activate xeus-sqlite # Or `source activate xeus-sqlite` for conda < 4.6
```

Then you can install in this freshly created environment *xeus-sqlite* and its dependencies

```
conda install xeus-sqlite notebook -c conda-forge
```

or, if you prefer to use JupyterLab

```
conda install xeus-sqlite jupyterlab -c conda-forge
```

1.1.2 From Source

You can install `xeus-sqlite` from source with cmake. This requires that you have all the dependencies installed in the same prefix.

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
make install
```

On Windows platforms, from the source directory:

```
mkdir build
cd build
cmake -G "NMake Makefiles" -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
nmake
nmake install
```

1.2 Usage

Launch the Jupyter notebook with *jupyter notebook* or Jupyter lab with *jupyter lab* and launch a new SQLite notebook by selecting the **xsqlite** kernel.

1.3 API

1.3.1 Magic

Magics here will allow you to make operations on your Database.

`%LOAD <"path-to-db/yourdatabase.db"> [R | RW]`

Loads a database.

Receives two arguments, the path to the database location as a string and an option to open the database either as read and write "RW" or read only mode "R". If the optional argument is not set it will default to read and write mode.

`%CREATE <"path-to-db/yourdatabase.db">`

Creates a database in read and write mode.

Receives one argument, a string that's the path for the database.

`%DELETE`

Deletes the database that's currently loaded in the Jupyter application.

Warning: this will delete the file permanently.

`%TABLE_EXISTS "table_name"`

Checks if a table exists.

`%GET_INFO`

Get the following information about the loaded database: * Magic header string * Page size bytes * File format write version * File format read version * Reserved space bytes * Max embedded payload fraction * Min embedded payload fraction * Leaf payload fraction * File change counter * Database size pages * First freelist trunk page * Total freelist trunk pages * Schema cookie * Schema format number * Default page cache size bytes * Largest B tree page number * Database text encoding * User version * Incremental vaccum mode * Application ID * Version valid for * SQLite version

`%IS_UNENCRYPTED`

Test if a file contains an unencrypted database.

`%LOAD_EXTENSION <"extension">`

Load a module into the current sqlite database instance.

Receives the name of the shared library containing the extension.

`%REKEY <"key">`

Reset the key for the current sqlite database instance. This is the equivalent of the sqlite3_rekey call and should thus be called after the database has been opened with a valid key. To decrypt a database, call this method with an empty string.

Receives one argument which is the key you want to reset.

`%SET_KEY <"key">`

Set the key for the current sqlite database instance. This is the equivalent of the sqlite3_key call and should thus be called directly after opening the database.

Receives one argument which is the key you want to reset.

`%BACKUP <0, 1>`

Load the contents of a database file on disk into the “main” database of open database connection, or to save the current contents of the database into a database file on disk.

Receives one argument which is an int that can either be 0 for saving and 1 for loading.