**What is SQLite?**

**Learn about the SQLite database engine and how to install it on your computer.**

In this article, we will be exploring the extremely prevalent database engine called [SQLite](https://www.sqlite.org/index.html). We will describe what it does, its main uses, and then explain how to set it up and use it on your own computer.

**What is SQLite?**

SQLite is a database engine. It is software that allows users to interact with a relational database. In SQLite, a database is stored in a single file — a trait that distinguishes it from other database engines. This fact allows for a great deal of accessibility: copying a database is no more complicated than copying the file that stores the data, sharing a database can mean sending an email attachment.

**Drawbacks to SQLite**

SQLite’s signature portability unfortunately makes it a poor choice when many different users are updating the table at the same time (to maintain integrity of data, only one user can write to the file at a time). It also may require some more work to ensure the security of private data due to the same features that make SQLite accessible. Furthermore, SQLite does not offer the same exact functionality as many other database systems, limiting some advanced features other relational database systems offer. Lastly, SQLite does not validate data types. Where many other database software would reject data that does not conform to a table’s schema, SQLite allows users to store data of any type into any column.

SQLite creates schemas, which constrain the type of data in each column, but it does not enforce them. The example below shows that the id column expects to store integers, the name column expects to store text, and the age column expects to store integers:

CREATE TABLE celebs (  
   id INTEGER,   
   name TEXT,   
   age INTEGER  
);

However, SQLite will not reject values of the wrong type. We could accidentally insert the wrong data types in the columns. Storing different data types in the same column is a bad habit that can lead to errors that are difficult to fix, so it’s important to be strict about your schema even though SQLite will not enforce it.

**Uses for SQLite**

Even considering the drawbacks, the benefits of being able to access and manipulate a database without involving a server application are huge. SQLite is used worldwide for testing, development, and in any other scenario where it makes sense for the database to be on the same disk as the application code. SQLite’s maintainers consider it to be among the [most replicated pieces of software in the world](https://www.sqlite.org/mostdeployed.html).

**Setting Up SQLite**

Binaries for SQLite can be installed at the [SQLite Download](https://www.sqlite.org/download.html) page.

**Windows**

For Windows machines:

1. Download the **sqlite-tools-win32-x86-3200100.zip** file and unzip it.
2. From your git-bash terminal, open the directory of the unzipped folder with cd ~/Downloads/sqlite-tools-win32-x86-3200100/sqlite-tools-win32-x86-3200100/.
3. Try running sqlite with the command winpty ./sqlite3.exe. If that command opens a sqlite> prompt, congratulations! You’ve installed SQLite.

We want to be able to access this command quickly from elsewhere, so we’re going to create an alias to the command. Exit the sqlite> prompt by typing in Ctrl + C, and in the same git-bash terminal without changing folders, run these commands:

echo "alias sqlite3=\"winpty ${PWD}/sqlite3.exe\"" >> ~/.bashrc

and

source ~/.bashrc

The first command will create the alias sqlite3 that you can use to open a database. The second command will refresh your terminal so that you can start using this command. Try typing in the command sqlite3 newdb.sqlite. If you’re presented with a sqlite> prompt, you’ve successfully created the sqlite3 command for your terminal. Enter Ctrl + C to quit. You can also exit by typing .exit in the prompt and pressing Enter.

**Video Tutorial:** [Setting Up SQLite Locally (Windows)](https://youtu.be/dcfh5iQ_-3s)

**Mac OS X**

For Macs, use the Mac OS X (x86) sqlite-tools package:

1. Install it, and unzip it.
2. In your terminal, navigate to the directory of the unzipped folder using cd.
3. Run the command mv sqlite3 /usr/local/bin/. This will add the command sqlite3 to your terminal path, allowing you to use the command from anywhere.
4. Try typing sqlite3 newdb.sqlite. If you’re presented with a sqlite> prompt, you’ve installed SQLite! Enter control + d to quit. You can also exit by typing .exit in the prompt and pressing return.

**Video Tutorial:** [Setting Up SQLite Locally (Mac)](https://youtu.be/4MJSZi4qvIE)

**Linux**

In Ubuntu or similar distributions:

1. Open your terminal and run sudo apt-get install sqlite3. Otherwise, use your distribution’s package managers.
2. Try typing in the command sqlite3 newdb.sqlite. If you’re presented with a sqlite> prompt, you’ve successfully created the sqlite3 command for your terminal. You can exit by typing .exit in the prompt and pressing enter.

**Conclusion**

You’ve installed database software and opened a connection to a database. Now you have the full power of SQL at your fingertips. You’ll be able to manage all the data for any application you can dream of writing. Congratulations! For more info on other Relational Database Management Systems, check out [this](https://www.codecademy.com/content-items/8f0c407f439676d91cd3733a394c720e) article.