# **SQLite**

SQLite is a RDBMS contained in C library implemented for small, fast, self-contained, high-reliability, full-featured, SQL database engine.

SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers

The lite in **SQLite** means light weight in terms of setup, database administration, and required resource.

Limitations: -

* **Maximum Database Size**

SQLite can have a maximum database size of 140 terabytes (TB). A SQLite database is a set of one more pages where every page is the same size. Maximum size of a page cannot exceed 65536 bytes. The maximum size of a database file is 2147483646 pages.

65536 bytes x 2147483646 pages = 140 TB (140,000 GB)

* **Maximum Number Of Attached Databases**

SQLite allows you to attach multiple databases with a database connection. The maximum number of database attachment limit is 125. The default limit is 10.

* **Maximum Number Of Pages In A Database File**

The SQLITE\_MAX\_PAGE\_COUNT parameter, which is normally set to 1073741823, is the maximum number of pages allowed in a single database file. But the maximum number of pages a SQLite database can be increased to 2147483646 pages. The maximum page size is 65536 bytes.

* **Maximum Number Of Tables In A Schema**

Each table in a SQLite database requires one or more pages. A SQLite database can have maximum 2147483646 pages. Hence the maximum number of tables in a schema cannot reach more than 2147483646.

* **Maximum Number Of Rows In A Table**

The maximum number of rows in a table is 264.

* **Maximum Number Of Columns**

The maximum number of columns is 32767 in a table. The default setting for SQLITE\_MAX\_COLUMN is 2000.

* **Maximum Length Of An SQL Statement**

The maximum number of bytes in the text of an SQL statement is limited to SQLITE\_MAX\_SQL\_LENGTH which defaults to 1000000. You can redefine this limit to be as large as the smaller of SQLITE\_MAX\_LENGTH and 1073741824.

* **Maximum Number Of Tables In A Join**

SQLite does not support joins containing more than 64 tables.

* **Maximum Number Of Arguments On A Function**

The maximum number of arguments in a function cannot be more than 127. The default value of this limit is 100.

* **Maximum length of a string or BLOB**

The maximum number of bytes in a string or BLOB in SQLite is defined by the preprocessor macro SQLITE\_MAX\_LENGTH. The default value of this macro is 1 billion. The current implementation will only support a string or BLOB length up to 231-1 or 2147483647.

* **Maximum Depth Of An Expression Tree**

SQLite parses expressions into a tree for processing. The default value of expression tree is 1000.

* **Maximum Depth Of Trigger Recursion**

The default maximum trigger recursion depth is 1000.

* **Maximum Length Of A LIKE Or GLOB Pattern**

The default maximum length of A LIKE Or GLOB pattern limit is 50000 in a SQLite database.

* **Zero-Configuration**

SQLite does not need to be "installed" before it is used. There is no "setup" procedure. There is no server process that needs to be started, stopped, or configured. There is no need for an administrator to create a new database instance or assign access permissions to users. SQLite uses no configuration files. Nothing needs to be done to tell the system that SQLite is running. No actions are required to recover after a system crash or power failure. There is nothing to troubleshoot.

SQLite just works.

Other more familiar database engines run great once you get them going. But doing the initial installation and configuration can be intimidatingly complex.

* **Stable Cross-Platform Database File**

The SQLite file format is cross-platform. A database file written on one machine can be copied to and used on a different machine with a different architecture. Big-endian or little-endian, 32-bit or 64-bit does not matter. All machines use the same file format. Furthermore, the developers have pledged to keep the file format stable and backwards compatible, so newer versions of SQLite can read and write older database files.

Most other SQL database engines require you to dump and restore the database when moving from one platform to another and often when upgrading to a newer version of the software.

* **Compact**

When optimized for size, the whole SQLite library with everything enabled is [less than 500KiB in size](https://www.sqlite.org/footprint.html) (as measured on an ix86 using the "size" utility from the GNU compiler suite.) Unneeded features can be disabled at compile-time to further reduce the size of the library to under 300KiB if desired.

Most other SQL database engines are much larger than this. IBM boasts that its recently released CloudScape database engine is "only" a 2MiB jar file - an order of magnitude larger than SQLite even after it is compressed! Firebird boasts that its client-side library is only 350KiB. That's as big as SQLite and does not even contain the database engine. The Berkeley DB library from Oracle is 450KiB and it omits SQL support, providing the programmer with only simple key/value pairs.

**Installing SQLite:**

1. type in the following command –

*$ sudo apt-get install sqlite3 libsqlite3-dev*

1. After installation check installation, sqlite terminal will give you a prompt and version info –

*$ sqlite3*

The output will be as follows:

*SQLite version 3.8.2 2013-12-06 14:53:30*

1. Enter &quot;.help&quot; for instructions
2. Enter SQL statements terminated with a &quot;;&quot;

*sqlite&gt;*

1. To quit –

*sqlite&gt; .quit*

4. Go to desired folder and create database –

*$ sqlite3 database\_name.db*

It’ll create database\_name.db in the folder you’ve given the command.

## **Some more limitations**

There are few unsupported features of SQL92 in SQLite which are listed in the following table.

|  |  |
| --- | --- |
| **Sr.No.** | **Feature & Description** |
| 1 | **RIGHT OUTER JOIN**  Only LEFT OUTER JOIN is implemented. |
| 2 | **FULL OUTER JOIN**  Only LEFT OUTER JOIN is implemented. |
| 3 | **ALTER TABLE**  The RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported. The DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT are not supported. |
| 4 | **Trigger support**  FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers. |
| 5 | **VIEWs**  VIEWs in SQLite are read-only. You may not execute a DELETE, INSERT, or UPDATE statement on a view. |
| 6 | **GRANT and REVOKE**  The only access permissions that can be applied are the normal file access permissions of the underlying operating system. |

## **SQLite Commands**

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their operational nature −

## **DDL - Data Definition Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **CREATE**  Creates a new table, a view of a table, or other object in database. |
| 2 | **ALTER**  Modifies an existing database object, such as a table. |
| 3 | **DROP**  Deletes an entire table, a view of a table or other object in the database. |

## **DML - Data Manipulation Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **INSERT**  Creates a record |
| 2 | **UPDATE**  Modifies records |
| 3 | **DELETE**  Deletes records |

## **DQL - Data Query Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **SELECT**  Retrieves certain records from one or more tables |

## **SQLite dot Command**

Following is a list of SQLite dot commands. These commands are not terminated by a semicolon (;).

**.help command:**

Check the list of dot commands by using the ".help" at anytime.