# Steps to install sqlite on linux system:

## Download source code from the following location

https://github.com/sqlite/sqlite

**:~$ wget https://github.com/sqlite/sqlite/archive/master.zip**

--2024-11-12 14:49:11-- https://github.com/sqlite/sqlite/archive/master.zip

Resolving github.com (github.com)... 20.207.73.82

Connecting to github.com (github.com)|20.207.73.82|:443... connected.

HTTP request sent, awaiting response... 302 Found

Location: https://codeload.github.com/sqlite/sqlite/zip/refs/heads/master [following]

--2024-11-12 14:49:12-- https://codeload.github.com/sqlite/sqlite/zip/refs/heads/master

Resolving codeload.github.com (codeload.github.com)... 20.207.73.88

Connecting to codeload.github.com (codeload.github.com)|20.207.73.88|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: unspecified [application/zip]

Saving to: ‘master.zip’

master.zip [ <=> ] 13.54M 4.07MB/s in 3.4s

2024-11-12 14:49:15 (3.96 MB/s) - ‘master.zip’ saved [14199031]

**Unzip the file:**

~$ unzip master.zip

**~$ ls sqlite-master/**

art auto.def configure doc LICENSE.md main.mk Makefile.linux-generic manifest mptest sqlite3.1 sqlite\_cfg.h.in src tool

autoconf autosetup contrib ext magic.txt Makefile.in Makefile.msc manifest.uuid README.md sqlite3.pc.in sqlite.pc.in test VERSION

## Configure and Compile for Intel x86\_64 and arm64 platform

./configure --host=x86\_64-linux-gnu

./configure --host=aarch64-linux-gnu

## Configure Logs for Intelx86\_64 platform

**:~/sqlite-master$ ./configure --host=x86\_64-linux-gnu**

Host System...x86\_64-pc-linux-gnu

Build System...x86\_64-pc-linux-gnu

C compiler... x86\_64-linux-gnu-gcc

C++ compiler... x86\_64-linux-gnu-g++

Build C compiler...cc

Checking for stdlib.h...ok

Source dir = /home/ubuntu-pc/sqlite-master

Build dir = /home/ubuntu-pc/sqlite-master

Configuring SQLite version 3.48.0

Looking for install ... /usr/bin/install

Checking for sys/types.h...ok

Checking if -D\_FILE\_OFFSET\_BITS=64 is needed...no

Checking for int8\_t...ok

Checking for int16\_t...ok

Checking for int32\_t...ok

Checking for int64\_t...ok

Checking for intptr\_t...ok

Checking for uint8\_t...ok

Checking for uint16\_t...ok

Checking for uint32\_t...ok

Checking for uint64\_t...ok

Checking for uintptr\_t...ok

Checking for gmtime\_r...ok

Checking for isnan...ok

Checking for localtime\_r...ok

Checking for localtime\_s...not found

Checking for malloc\_usable\_size...ok

Checking for strchrnul...ok

Checking for usleep...ok

Checking for utime...ok

Checking for pread...ok

Checking for pread64...ok

Checking for pwrite...ok

Checking for pwrite64...ok

Checking libs for fdatasync...none needed

Checking for sys/types.h...(cached) ok

Checking for sys/stat.h...ok

Checking for dlfcn.h...ok

Checking for unistd.h...ok

Checking for stdlib.h...ok

Checking for malloc.h...ok

Checking for memory.h...ok

Checking for string.h...ok

Checking for strings.h...ok

Checking for inttypes.h...ok

Checking for zlib.h...ok

Checking libs for deflate...-lz

Checking whether the C compiler accepts -rpath /usr/local/lib...no

Checking whether the C compiler accepts -Wl,-rpath,/usr/local/lib...yes

Build shared library? yes

Build static library? yes

Use amalgamation for builds? yes

Use gcov? no

test-runner flags: no

Use #line macros in the amalgamation: no

SQLITE\_DEBUG build? no

Checking for a suitable tcl...

Checking for tclsh9.0...no

Checking for tclsh8.6.../usr/bin/tclsh8.6

WARNING: /usr/bin/tclsh8.6 is unable to recommend a tclConfig.sh

Using tclsh: /usr/bin/tclsh8.6

WARNING: Found tclsh but no tclConfig.sh.

WARNING: Cannot find a usable tclConfig.sh file. Use

--with-tcl=DIR to specify a directory where tclConfig.sh can be

found. SQLite does not use TCL internally, but some optional

components require TCL, including tests and sqlite3\_analyzer.

Checking for TCL to use for code generation...

Checking for dirent.h...ok

Checking for sys/time.h...ok

Checking for realpath...ok

TCL for code generation: jimsh

Support threadsafe operation? yes

Checking libs for pthread\_create...none needed

Checking libs for pthread\_mutexattr\_init...none needed

Use an in-RAM database for temporary tables? no

Checking for line-editing capability...

Line-editing support for the sqlite3 shell: none

Checking libs for dlopen...none needed

Checking libs for ceil...-lm

Enabling math SQL functions -lm

ICU support is disabled.

Emscripten SDK? not found

Feature flags...

- fts4

- fts5

- geopoly

- rtree

- session

- update-limit

- memsys5

- memsys3

+ json

Library feature flags: -DSQLITE\_ENABLE\_MATH\_FUNCTIONS -DSQLITE\_THREADSAFE=1

Shell options: -DSQLITE\_HAVE\_ZLIB=1

Created Makefile from Makefile.in

Created sqlite3.pc from sqlite3.pc.in

Created sqlite\_cfg.h

**~/sqlite-master$** make

~/sqlite-master$ sudo make install

[sudo] password for ubuntu-pc:

/usr/bin/install -d "/usr/local/lib/pkgconfig"

/usr/bin/install -m 0644 sqlite3.pc "/usr/local/lib/pkgconfig"

/usr/bin/install libsqlite3.so "/usr/local/lib"

Setting up SO symlinks...

lrwxrwxrwx 1 root root 15 Nov 12 15:02 libsqlite3.so -> libsqlite3.so.3

lrwxrwxrwx 1 root root 20 Nov 12 15:02 libsqlite3.so.3 -> libsqlite3.so.3.48.0

-rwxr-xr-x 1 root root 4915888 Nov 12 15:02 libsqlite3.so.3.48.0

/usr/bin/install -m 0644 libsqlite3.a "/usr/local/lib"

/usr/bin/install -m 0644 sqlite3.h "/home/ubuntu-pc/sqlite-master/src/sqlite3ext.h" "/usr/local/include"

TCL support disabled, so not installing libtclsqlite3.so

/usr/bin/install -s sqlite3 "/usr/local/bin"

/usr/bin/install -m 0644 "/home/ubuntu-pc/sqlite-master/sqlite3.1" "/usr/local/share/man/man1"

# Validating sqlite

## Sqlite commands

**Creating a Database**

sqlite3 mydatabase.db

**Creating a Table**

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

age INTEGER

);

**Inserting Data**

INSERT INTO users (name, age) VALUES ('Alice', 30), ('Bob', 25);

**Querying Data**

SELECT \* FROM users;

**Updating Data**

UPDATE users SET age = 31 WHERE name = 'Alice';

**Querying Data**

SELECT \* FROM users;

**Deleting Data**

DELETE FROM users WHERE id = 1;

**Querying Data**

SELECT \* FROM users;

**Exiting SQLite**

.quit

## Logs from using sqlite commands

**:~/sqlite-master$ sqlite3 --version**

3.48.0 2024-11-11 21:11:02 5c5982e3937acdcda43d6c5b46a95b82bc1839c3558a4b9ae9022384e0f13f04 (64-bit)

**ubuntu-pc@aayudhinapapa:~/sqlite-master$ sqlite3 mydatabase.db**

SQLite version 3.48.0 2024-11-11 21:11:02

Enter ".help" for usage hints.

sqlite> CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

age INTEGER

);

(x1...> (x1...> (x1...> (x1...> sqlite>

sqlite> INSERT INTO users (name, age) VALUES ('Alice', 30), ('Bob', 25);

sqlite> SELECT \* FROM users;

1|Alice|30

2|Bob|25

sqlite>

sqlite>

sqlite>

sqlite> UPDATE users SET age = 31 WHERE name = 'Alice';

sqlite> SELECT \* FROM users;

1|Alice|31

2|Bob|25

sqlite> DELETE FROM users WHERE id = 1;

sqlite> SELECT \* FROM users;

2|Bob|25

sqlite> .quit

## Doing the above using python code to query the sqlite DB

**:~$ cat sql\_python.py**

import sqlite3

# Connect to the database

conn = sqlite3.connect('mydatabase.db')

# Create a cursor object

cursor = conn.cursor()

# Drop the 'users' table if it exists and create it again

print("Dropping and creating the 'users' table...")

cursor.execute("DROP TABLE IF EXISTS users")

cursor.execute('''

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

age INTEGER

)

''')

# Insert data

print("Inserting data into the 'users' table...")

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Alice', 30))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Bob', 25))

# Commit changes

conn.commit()

# Query data

print("Querying data from the 'users' table...")

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

print(row)

# Update data

print("Updating 'Alice's age to 31...")

cursor.execute("UPDATE users SET age = 31 WHERE name = 'Alice'")

conn.commit()

# Query data

print("Querying data from the 'users' table after update...")

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

print(row)

# Delete data

print("Deleting user with id = 1...")

cursor.execute("DELETE FROM users WHERE id = 1")

conn.commit()

# Query data

print("Querying data from the 'users' table after deletion...")

cursor.execute("SELECT \* FROM users")

rows = cursor.fetchall()

for row in rows:

print(row)

# Close the connection

print("Closing the database connection...")

conn.close()

**Logs:**

**~$ python3 sql\_python.py**

Dropping and creating the 'users' table...

Inserting data into the 'users' table...

Querying data from the 'users' table...

(1, 'Alice', 30)

(2, 'Bob', 25)

Updating 'Alice's age to 31...

Querying data from the 'users' table after update...

(1, 'Alice', 31)

(2, 'Bob', 25)

Deleting user with id = 1...

Querying data from the 'users' table after deletion...

(2, 'Bob', 25)

Closing the database connection...

**Python3 already has sqlite3 library installed (atleast on ubuntu), which has API’s to query and update the DB**

>>> import sqlite3

>>>

## **Sample code to get a feel of how the qwrap client interface DB is going to look like**

:~$ cat ping\_sqlite.py

import sqlite3

import random

# Connect to the database

conn = sqlite3.connect('network\_devices.db')

# Create a cursor object

cursor = conn.cursor()

# Drop the 'clients' table if it exists, to start fresh

print("Dropping and creating the 'clients' table...")

cursor.execute("DROP TABLE IF EXISTS clients")

cursor.execute('''

CREATE TABLE clients (

client\_id INTEGER PRIMARY KEY AUTOINCREMENT,

mac\_address TEXT,

ath\_interface TEXT,

veth\_interface TEXT,

veth\_bridge\_interface TEXT,

radio\_interface TEXT,

ip\_address TEXT

)

''')

# Helper functions to generate random data

def random\_mac():

return ':'.join(f"{random.randint(0, 255):02x}" for \_ in range(6))

def random\_ath\_interface():

return f"ath{random.choice([0, 1, 2])}"

def random\_veth\_interface():

return f"veth\_in\_{random.randint(1, 10)}"

def random\_veth\_bridge\_interface():

return f"veth\_br\_{random.randint(1, 10)}"

def random\_radio\_interface():

return random.choice(["2G", "5G", "6G"])

def random\_ip\_address():

return f"192.168.1.{random.randint(0, 255)}"

# Insert 10 sample entries

print("Inserting sample data into the 'clients' table...")

for \_ in range(10):

cursor.execute('''

INSERT INTO clients (mac\_address, ath\_interface, veth\_interface, veth\_bridge\_interface, radio\_interface, ip\_address)

VALUES (?, ?, ?, ?, ?, ?)

''', (

random\_mac(),

random\_ath\_interface(),

random\_veth\_interface(),

random\_veth\_bridge\_interface(),

random\_radio\_interface(),

random\_ip\_address()

))

# Commit changes

conn.commit()

# Query all data

print("\nQuerying all data from the 'clients' table...")

cursor.execute("SELECT \* FROM clients")

rows = cursor.fetchall()

for row in rows:

print(row)

# Query by MAC address

sample\_mac = rows[0][1]

print(f"\nQuerying data based on MAC address '{sample\_mac}'...")

cursor.execute("SELECT \* FROM clients WHERE mac\_address = ?", (sample\_mac,))

print(cursor.fetchall())

# Query by IP address

sample\_ip = rows[1][6]

print(f"\nQuerying data based on IP address '{sample\_ip}'...")

cursor.execute("SELECT \* FROM clients WHERE ip\_address = ?", (sample\_ip,))

print(cursor.fetchall())

# Query by ath interface

sample\_ath = rows[2][2]

print(f"\nQuerying data based on ath interface '{sample\_ath}'...")

cursor.execute("SELECT \* FROM clients WHERE ath\_interface = ?", (sample\_ath,))

print(cursor.fetchall())

# Query by Radio interface

sample\_radio = rows[3][5]

print(f"\nQuerying data based on Radio interface '{sample\_radio}'...")

cursor.execute("SELECT \* FROM clients WHERE radio\_interface = ?", (sample\_radio,))

print(cursor.fetchall())

# Close the connection

print("\nClosing the database connection...")

conn.close()

## **Sample output for the above code**

:~$ python3 ping\_sqlite.py

Dropping and creating the 'clients' table...

Inserting sample data into the 'clients' table...

Querying all data from the 'clients' table...

(1, '1e:27:7c:fe:c7:fa', 'ath1', 'veth\_in\_1', 'veth\_br\_8', '5G', '192.168.1.151')

(2, 'b6:ab:18:32:3b:91', 'ath0', 'veth\_in\_4', 'veth\_br\_3', '6G', '192.168.1.187')

(3, 'db:2b:26:78:79:d3', 'ath2', 'veth\_in\_10', 'veth\_br\_2', '6G', '192.168.1.102')

(4, 'f6:37:d5:bc:0b:85', 'ath1', 'veth\_in\_4', 'veth\_br\_9', '2G', '192.168.1.250')

(5, 'cd:fb:57:56:36:51', 'ath0', 'veth\_in\_2', 'veth\_br\_4', '6G', '192.168.1.122')

(6, '40:cc:22:3d:26:08', 'ath2', 'veth\_in\_1', 'veth\_br\_3', '5G', '192.168.1.163')

(7, '73:85:b3:85:be:09', 'ath0', 'veth\_in\_1', 'veth\_br\_3', '2G', '192.168.1.97')

(8, '5a:83:7d:c3:54:0e', 'ath1', 'veth\_in\_2', 'veth\_br\_10', '6G', '192.168.1.143')

(9, 'a2:3d:a5:2d:62:e6', 'ath0', 'veth\_in\_6', 'veth\_br\_5', '6G', '192.168.1.185')

(10, '43:1a:ed:6e:bb:19', 'ath1', 'veth\_in\_9', 'veth\_br\_5', '5G', '192.168.1.96')

Querying data based on MAC address '1e:27:7c:fe:c7:fa'...

[(1, '1e:27:7c:fe:c7:fa', 'ath1', 'veth\_in\_1', 'veth\_br\_8', '5G', '192.168.1.151')]

Querying data based on IP address '192.168.1.187'...

[(2, 'b6:ab:18:32:3b:91', 'ath0', 'veth\_in\_4', 'veth\_br\_3', '6G', '192.168.1.187')]

Querying data based on ath interface 'ath2'...

[(3, 'db:2b:26:78:79:d3', 'ath2', 'veth\_in\_10', 'veth\_br\_2', '6G', '192.168.1.102'), (6, '40:cc:22:3d:26:08', 'ath2', 'veth\_in\_1', 'veth\_br\_3', '5G', '192.168.1.163')]

Querying data based on Radio interface '2G'...

[(4, 'f6:37:d5:bc:0b:85', 'ath1', 'veth\_in\_4', 'veth\_br\_9', '2G', '192.168.1.250'), (7, '73:85:b3:85:be:09', 'ath0', 'veth\_in\_1', 'veth\_br\_3', '2G', '192.168.1.97')]

Closing the database connection...