SQLite DB on Raspberry Pi

https://iotbytes.wordpress.com/sqlite-db-on-raspberry-pi/

SQLite is an embedded SQL database engine that provides a lightweight disk-based database. It doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language.

SQLite suits well for IoT Devices as it is self-contained, serverless, and requires no configuration. On top of this, it is free for use for any purpose.

So let's explore how to install and use SQLite on Raspberry Pi –

1. Update and Upgrade Raspberry Pi packages:

Update Raspberry Pi packages –

```
sudo apt-get update
```

Upgrade Raspberry Pi Packages –

```
sudo apt-get upgrade
```

2. Install SQLite:

Install SQLite3 using following command –

```
sudo apt-get install sqlite3
```

With this, you have SQLite installed on your Raspberry Pi and its ready for use.

3. Play with SQLite CLI Shell:

Create SQLite DB –

You can create SQLite DB from Raspbian CLI interface using following command –

```
sqlite3 <DB File Name>
```

Example:

```
pi@raspberrypi:~ $ sqlite3 my_DB
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite>
```

Create Table –

You can create DB Table using following command –

```
CREATE TABLE <Table Name> (<Field 1 Name> <Data Type>, ........ <Field n Name>
<Data Type>);
```

Example:

```
sqlite> create table sensors (sensor_ID integer, sensor_Name text);
sqlite>
```

Insert Data in Table –

You can insert Data into your table using following command –

```
INSERT INTO <Table Name> (<Field 1>, <Field 2>, ......... <Field n>) VALUES
(<Value 1>, <Value 2>, .......... <Value n>);
```

Example:

```
sqlite> insert into sensors (sensor_ID, sensor_Name) values (1,"Light Sensor");
sqlite> insert into sensors (sensor_ID, sensor_Name) values (2,"Temperature");
sqlite>
```

Read Data from Table –

You can list the Table data using following query –

```
SELECT * FROM <Table Name>;
```

Example:

```
sqlite> select * from sensors;
1|Light Sensor
2|Temperature
sqlite>
```

Delete Table –

```
DROP TABLE IF EXISTS <Table Name>;
```

Example:

```
sqlite> drop table if exists sensors;
sqlite>
```

Terminate SQLite CLI Shell –

You can terminate the sqlite3 program by typing your system End-Of-File character (usually a Control-D).

SQLite shell can also be terminated using ".exit" command –

```
sqlite>
sqlite> .exit
pi@raspberrypi:~ $
```

4. Use Python to Initialise SQLite DB:

You can use Python Script to initialise your SQLite DB. This method can be very useful if you need to change your DB Schema frequently (for ex during initial development phase) or want to create the same DB on multiple systems (for ex during code distribution or testing).

Create SQL File with Table Definition and Base Data –

Following is an example of SQL file. Save this file as "*Table_Schema.sql*" on your Raspberry Pi.

```
drop table if exists StdSensorTypes;

create table StdSensorTypes (
 SensorCode text,
 SensorType text,
 SensorImage text
);

INSERT INTO StdSensorTypes (SensorCode, SensorType) VALUES ("s0", "Temperature
and Humidity Sensor: DHT22");
INSERT INTO StdSensorTypes (SensorCode, SensorType) VALUES ("s1", "Pressure
Sensor: BMP180");
INSERT INTO StdSensorTypes (SensorCode, SensorType) VALUES ("s2", "Light Sensor:
LDR");
INSERT INTO StdSensorTypes (SensorCode, SensorType) VALUES ("s3", "Door-Windows
Sensor");
```

Create Python Script to Initialise SQLite DB –

Once you have your SQL file ready, create a Python Script with following code. As per your requirements, you can change the "*DB_NAME*" variable in this script. Save this file as "*DB_Ini.py*". Make sure both these files ("*DB_Ini.py*" and "*Table_Schema.sql*") are in same directory on Raspberry Pi.

```
import sqlite3

############### Settings ####################
#DB Name
DB_NAME = "My_DB"

#SQL File with Table Schema and Initialization Data
SQL_File_Name = "Table_Schema.sql"
#############################################

#Read Table Schema into a Variable and remove all New Line Chars
TableSchema=""
with open(SQL_File_Name, 'r') as SchemaFile:
 TableSchema=SchemaFile.read().replace('\n')

#Connect or Create DB File
conn = sqlite3.connect(DB_NAME)
curs = conn.cursor()

#Create Tables
sqlite3.complete_statement(TableSchema)
curs.executescript(TableSchema)

#Close DB
curs.close()
conn.close()
```

Execute Python Script –

To create and initialise the SQLite DB with tables and data simply execute the python script using following command –

```
python DB_Ini.py
```

Validate DB and Tables –

Following is the console output from Raspberry Pi. It includes script execution and DB/Table verification commands –

```
pi@raspberrypi:~/pradeep $ ls
DB_Ini.py Table_Schema.sql
pi@raspberrypi:~/pradeep $ python DB_Ini.py
pi@raspberrypi:~/pradeep $ ls
DB_Ini.py My_DB Table_Schema.sql
pi@raspberrypi:~/pradeep $ sqlite3 My_DB
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .tables
StdSensorTypes
sqlite> select * from StdSensorTypes;
s0|Temperature and Humidity Sensor: DHT22|
s1|Pressure Sensor: BMP180|
s2|Light Sensor: LDR|
s3|Door-Windows Sensor|
sqlite>
pi@raspberrypi:~/pradeep $
```