Introduction to SQLite

-

SQLite – Introduction

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a popular choice as an embedded database for local/client storage in application software such as web browsers. It is also used in many other applications that need a lightweight, embedded database.

SQLite is ACID-compliant and implements most of the SQL standards, using a dynamically and weakly typed SQL syntax that does not guarantee domain integrity.

To use SQLite in a C/C++ program, you can use the sqlite3 API, which provides a lightweight, simple, self-contained, high-reliability, full-featured, and SQL database engine. The API is implemented as a library of C functions that can be called from your program. One of the main benefits of using SQLite is that it is very easy to get started with. To create a new database in SQLite, you simply need to create a new file on your filesystem and connect to it using the sqlite3 API. For example, in C:

- C

```
#include <sqlite3.h>



int main(int argc, char **argv) {

  sqlite3 *db;

  int rc;



  rc = sqlite3_open("test.db", &db);

  if (rc != SQLITE_OK) {
    // error handling

  }



  // do something with the database



  sqlite3_close(db);

  return 0;

}
```

SQLite History

SQLite was created in the year 2000 by D. Richard Hipp, who continues to lead the development of the software today. SQLite was designed to be a lightweight and simple database engine that could be easily embedded into other applications. It was created as an alternative to more complex and heavyweight database engines, such as MySQL and PostgreSQL. Over the years, SQLite has gained widespread adoption and is now one of the most widely used database engines in the world. It is used in many applications, including web browsers, mobile phones, and a wide variety of other software.

SQLite is an open-source software project, and the source code is available under the terms of the SQLite license, which is a permissive, public domain-like license. This has contributed to its widespread adoption, as developers are free to use and modify the source code as they see fit.

Why SQLite?

There are several reasons why you might choose to use SQLite in your project:

1. **Ease of use:** SQLite is very easy to get started with, as it requires no setup or configuration. You can simply include the library in your project and start using it.

2. **Embeddability:** SQLite is designed to be embedded into other applications. It is a self-contained, serverless database engine, which means you can include it in your application without the need for a separate database server.
3. **Lightweight:** SQLite is a very lightweight database engine, with a small library size (typically less than 1MB). This makes it well-suited for use in applications where the database is embedded directly into the application binary, such as mobile apps.
4. **Serverless:** As mentioned earlier, SQLite is a serverless database engine, which means there is no need to set up and maintain a separate database server process. This makes it easy to deploy and manage, as there are no additional dependencies to worry about.
5. **Cross-platform:** SQLite is available on many platforms, including Linux, macOS, and Windows, making it a good choice for cross-platform development.
6. **Standalone:** SQLite stores all of the data in a single file on the filesystem, which makes it easy to copy or backup the database.
7. **High reliability:** SQLite has been widely tested and used in production systems for many years, and has a reputation for being a reliable and robust database engine.

SQLite Commands

In SQLite, DDL (Data Definition Language) is used to create and modify database objects such as tables, indices, and views. Some examples of DDL statements in SQLite are:

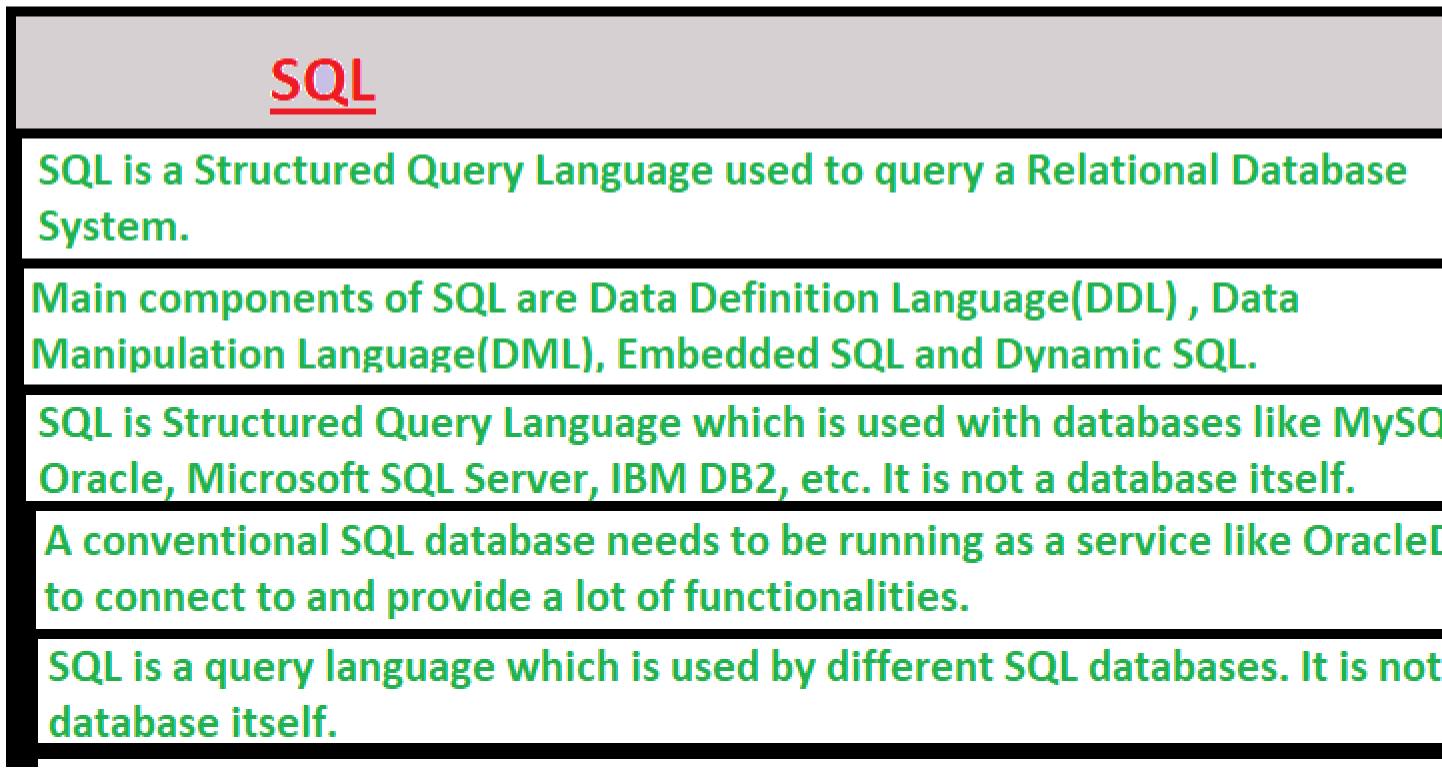
CREATE TABLE: creates a new table in the database
ALTER TABLE: modifies an existing table in the database
DROP TABLE: deletes a table from the database
CREATE INDEX: creates a new index on a table
DROP INDEX: deletes an index from a table

DML (Data Modification Language) is used to modify the data stored in the database. Some examples of DML statements in SQLite are:

INSERT INTO: inserts a new row into a table
UPDATE: updates the data in one or more rows of a table
DELETE FROM: deletes one or more rows from a table

DQL (Data Query Language) is used to retrieve data from the database. Some examples of DQL statements in SQLite are:

SELECT: retrieves data from one or more tables in the database
JOIN: retrieves data from multiple tables based on a common field
GROUP BY: groups the results of a query by one or more fields
HAVING: filters the results of a query based on a condition



SQL
SQL is a Structured Query Language used to query a Relational Database System.
Main components of SQL are Data Definition Language(DDL) , Data Manipulation Language(DML), Embedded SQL and Dynamic SQL.
SQL is Structured Query Language which is used with databases like MySQ Oracle, Microsoft SQL Server, IBM DB2, etc. It is not a database itself.
A conventional SQL database needs to be running as a service like OracleD to connect to and provide a lot of functionalities.
SQL is a query language which is used by different SQL databases. It is not database itself.

SQLite Limitation

Limited concurrency: SQLite uses file-based locking to control access to the database, which can lead to performance issues when multiple clients are trying to read and write to the database simultaneously. This makes it less suitable for use in highly concurrent systems.
No support for stored procedures: SQLite does not support stored procedures, which are pre-compiled SQL statements that can be executed on the server. This means that all SQL code must be sent to the server and compiled at runtime, which can be less efficient than using stored procedures.
No support for triggers: SQLite does not support triggers, which are database actions that are automatically triggered by specified events (such as the insertion of a row into a table). This means that you have to manually implement any logic that needs to be triggered by specific events.
Limited support for data types: SQLite has a relatively small set of data types compared to other database engines. It does not support many of the more

advanced data types, such as arrays and JSON, that are available in other databases.

Limited scalability: SQLite is not designed to be a high-concurrency, high-transaction-rate database engine. It is more suited for use in smaller-scale, low-concurrency systems, and may not be able to scale to handle very large amounts of data or very high levels of concurrency.