EXPERIMENT NO.3

Problem Statement:

Study of SQLite: What is SQLite? Uses of Sqlite. Building and installing SQLite.

Objective:

To gain a comprehensive understanding of SQLite, its uses, and the process of building and installing SQLite.

Prerequisites:

1. Basic knowledge of databases and SQL.
2. Access to a computer where SQLite can be installed.

Equipment and Software:

1. Lab machines with administrative access.
2. SQLite command-line interface.

Theory:

Introduction to SQLite

What is SQLite?

SQLite is a self-contained, serverless, and zero-configuration relational database management system (RDBMS). It is a software library that provides a lightweight, disk-based database that doesn't require a separate server process and allows direct access to the database using a nonstandard variant of the SQL query language. SQLite is known for its simplicity, efficiency, and ease of integration into various applications.

Uses of SQLite

1. **Embedded Systems:** SQLite is commonly used in embedded systems and IoT devices due to its lightweight nature and minimal resource requirements.
2. **Mobile Applications:** Many mobile applications, including Android and iOS, use SQLite as their local database for storing app-related data.
3. **Desktop Software:** SQLite is suitable for desktop applications where a lightweight, file-based database is needed without the complexity of a full-fledged database server.

4. **Web Browsers:** Several web browsers use SQLite for storing bookmarks, history, and other data.
5. **Small to Medium-sized Websites:** SQLite is a good choice for small to medium-sized websites where a lightweight database is sufficient for managing data.
6. **Educational Purposes:** Due to its simplicity and ease of use, SQLite is often used in educational settings for teaching database concepts.

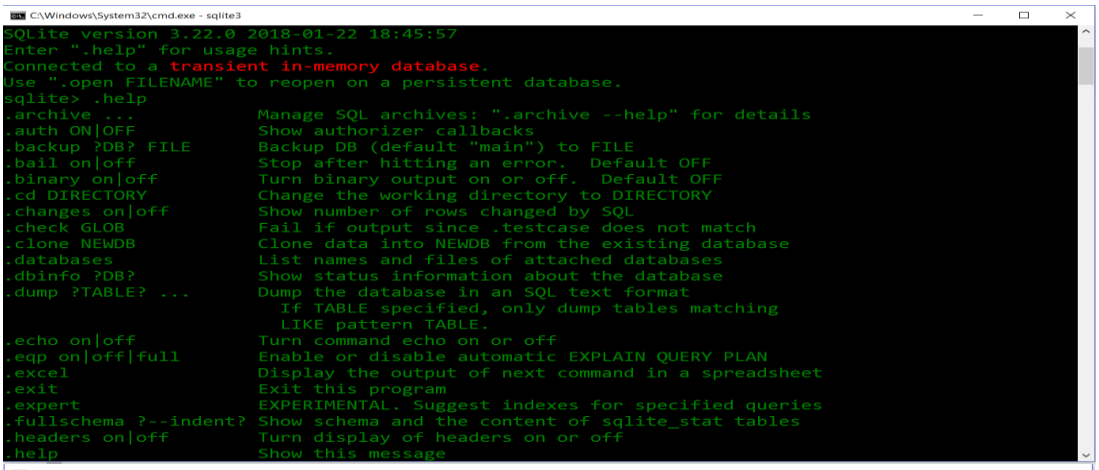
Features of SQLite

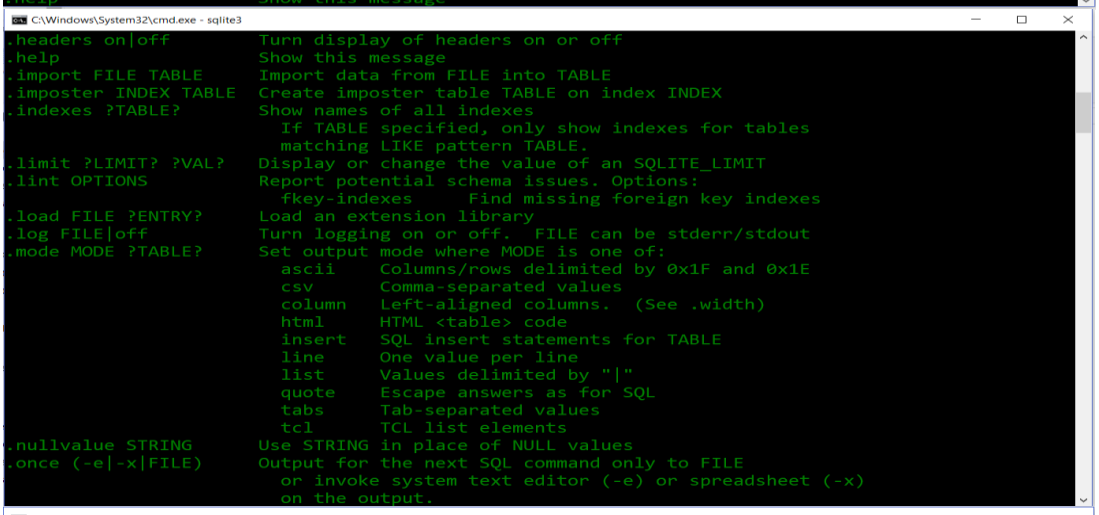
1. The transactions follow ACID properties i.e. atomicity, consistency, isolation, and durability even after system crashes and power failures.
2. The configuration process is very easy, no setup or administration is needed.
3. All the features of SQL are implemented in it with some additional features like partial indexes, indexes on expressions, JSON, and common table expressions.
4. Sometimes it is faster than the direct file system I/O.
5. It supports terabyte-sized databases and gigabyte-sized strings and blobs.
6. Almost all OS supports SQLite like Android, BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, etc. It is very much easy to port to other systems.
7. A complete database can be stored in a single cross-platform disk file.

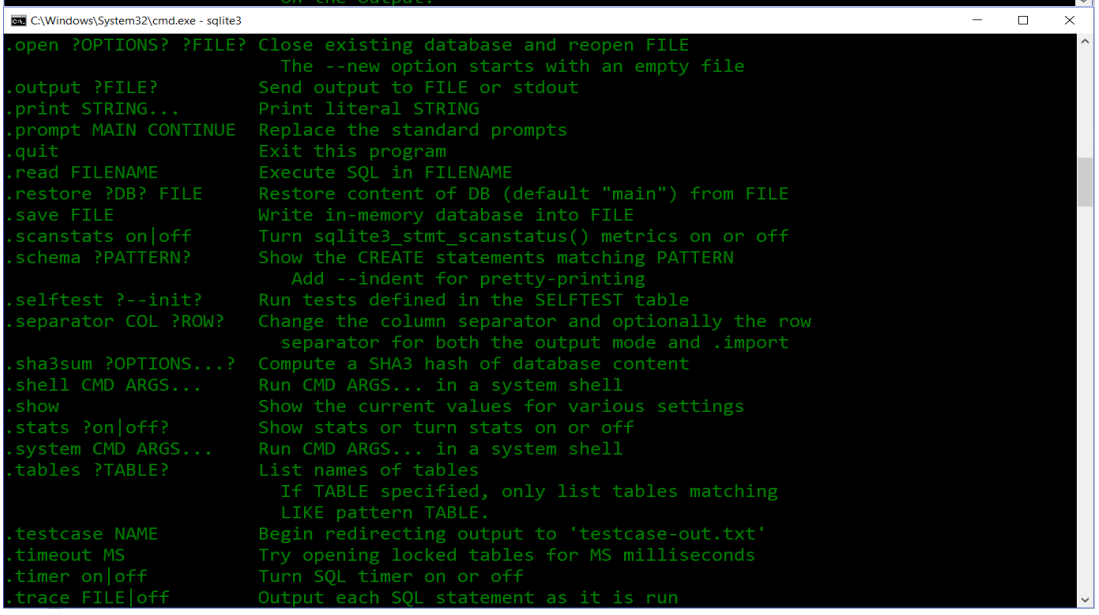
Applications of SQLite

1. Due to its small code print and efficient usage of memory, it is the popular choice for the database engine in cell phones, PDAs, MP3 players, set-top boxes, and other electronic gadgets.
2. It is used as an alternative for open to writing XML, JSON, CSV, or some proprietary format into disk files used by the application.
3. As it has no complication for configuration and easily stores file in an ordinary disk file, so it can be used as a database for small to medium sized websites.
4. It is faster and accessible through a wide variety of third-party tools, so it has great applications in different software platforms.

SQLite Commands In SQLite, there are several dot commands which do not end with a semicolon(;). Here are all commands and their description:


C:\Windows\System32\cmd.exe - sqlite3
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .help
.archive ... Manage SQL archives: ".archive --help" for details
.auth ON|OFF Show authorizer callbacks
.backup ?DB? FILE Backup DB (default "main") to FILE
.bail on|off Stop after hitting an error. Default OFF
.binary on|off Turn binary output on or off. Default OFF
.cd DIRECTORY Change the working directory to DIRECTORY
.changes on|off Show number of rows changed by SQL
.check GLOB Fail if output since .testcase does not match
.clone NEWDB Clone data into NEWDB from the existing database
.databases List names and files of attached databases
.dbinfo ?DB? Show status information about the database
.dump ?TABLE? ... Dump the database in an SQL text format
If TABLE specified, only dump tables matching
LIKE pattern TABLE.
.echo on|off Turn command echo on or off
.eqp on|off|full Enable or disable automatic EXPLAIN QUERY PLAN
.excel Display the output of next command in a spreadsheet
.exit Exit this program
.expert EXPERIMENTAL. Suggest indexes for specified queries
.fullschema ?--indent? Show schema and the content of sqlite_stat tables
.headers on|off Turn display of headers on or off
.help Show this message


C:\Windows\System32\cmd.exe - sqlite3
.headers on|off Turn display of headers on or off
.help Show this message
.import FILE TABLE Import data from FILE into TABLE
.imposter INDEX TABLE Create imposter table TABLE on index INDEX
.indexes ?TABLE? Show names of all indexes
If TABLE specified, only show indexes for tables
matching LIKE pattern TABLE.
.limit ?LIMIT? ?VAL? Display or change the value of an SQLITE_LIMIT
.lint OPTIONS Report potential schema issues. Options:
fkey-indexes Find missing foreign key indexes
.load FILE ?ENTRY? Load an extension library
.log FILE|off Turn logging on or off. FILE can be stderr/stdout
.mode MODE ?TABLE? Set output mode where MODE is one of:
ascii Columns/rows delimited by 0x1F and 0x1E
csv Comma-separated values
column Left-aligned columns. (See .width)
html HTML <table> code
insert SQL insert statements for TABLE
line One value per line
list Values delimited by "|"
quote Escape answers as for SQL
tabs Tab-separated values
tcl TCL list elements
.nullvalue STRING Use STRING in place of NULL values
.once (-e|-x|FILE) Output for the next SQL command only to FILE
or invoke system text editor (-e) or spreadsheet (-x)
on the output.


C:\Windows\System32\cmd.exe - sqlite3
.open ?OPTIONS? ?FILE? Close existing database and reopen FILE
The --new option starts with an empty file
.output ?FILE? Send output to FILE or stdout
.print STRING... Print literal STRING
.prompt MAIN CONTINUE Replace the standard prompts
.quit Exit this program
.read FILENAME Execute SQL in FILENAME
.restore ?DB? FILE Restore content of DB (default "main") from FILE
.save FILE Write in-memory database into FILE
.scanstats on|off Turn sqlite3_stmt_scanstatus() metrics on or off
.schema ?PATTERN? Show the CREATE statements matching PATTERN
Add --indent for pretty-printing
.selftest ?--init? Run tests defined in the SELFTEST table
.separator COL ?ROW? Change the column separator and optionally the row
separator for both the output mode and .import
.sha3sum ?OPTIONS...? Compute a SHA3 hash of database content
.shell CMD ARGS... Run CMD ARGS... in a system shell
.show Show the current values for various settings
.stats ?on|off? Show stats or turn stats on or off
.system CMD ARGS... Run CMD ARGS... in a system shell
.tables ?TABLE? List names of tables
If TABLE specified, only list tables matching
LIKE pattern TABLE.
.testcase NAME Begin redirecting output to 'testcase-out.txt'
.timeout MS Try opening locked tables for MS milliseconds
.timer on|off Turn SQL timer on or off
.trace FILE|off Output each SQL statement as it is run

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

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.

Building and Installing SQLite

Source Code Download

1. Visit the official SQLite website at https://www.sqlite.org/.
2. Navigate to the "Download" section and choose the source code package compatible with your system.

Building SQLite

1. Extract the downloaded source code package to a directory of your choice.
2. Open a terminal or command prompt and navigate to the extracted directory.
3. Run the following commands to configure, build, and install SQLite:

./configure make sudo make install

Replace **sudo** with the appropriate command for your system if you don't have superuser privileges.

Verification

1. After installation, run the following command to check the SQLite version:

sqlite3 --version

This should display the installed SQLite version.

2. Launch the SQLite command-line interface:

sqlite3

You should now be in the SQLite prompt.

3. Exit the SQLite prompt:

.exit

This ensures that you can enter and exit the SQLite environment successfully.

Conclusion

In this introduction, we've covered the basics of SQLite, its common uses, and the process of building and installing SQLite from source code. Understanding these fundamentals is crucial for further exploration and utilization of SQLite in various applications.