

Unit-1

Introduction to SQLite

Introduction to SQLite

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. Ordinary disk files can be easily read and write by SQLite because it does not have any separate server like SQL. The SQLite database file format is cross-platform so that anyone can easily copy a database between 32-bit and 64-bit systems. Due to all these features, it is a popular choice as an Application File Format.

History

It was designed by D. Richard Hipp for the purpose of no administration required for operating a program. in August 2000. As it is very lightweight compared to others like MySql and Oracle, it is called SQLite. Different versions of SQLite are released since 2000.

Differences between SQL and SQLite:

SQL	SQLite
SQL is Structured Query Language used to query Relational Database System. It is written in C language.	SQLite is an Relational Database Management System which is written in ANSI-C.
SQL is standard which specifies how relational schema is created, data is inserted or updated in relations, transactions are started and stopped, etc.	SQLite is file-based. It is different from other SQL databases because unlike most other SQL databases, SQLite does not have separate server process.

Unit- 1: Introduction to SQLite

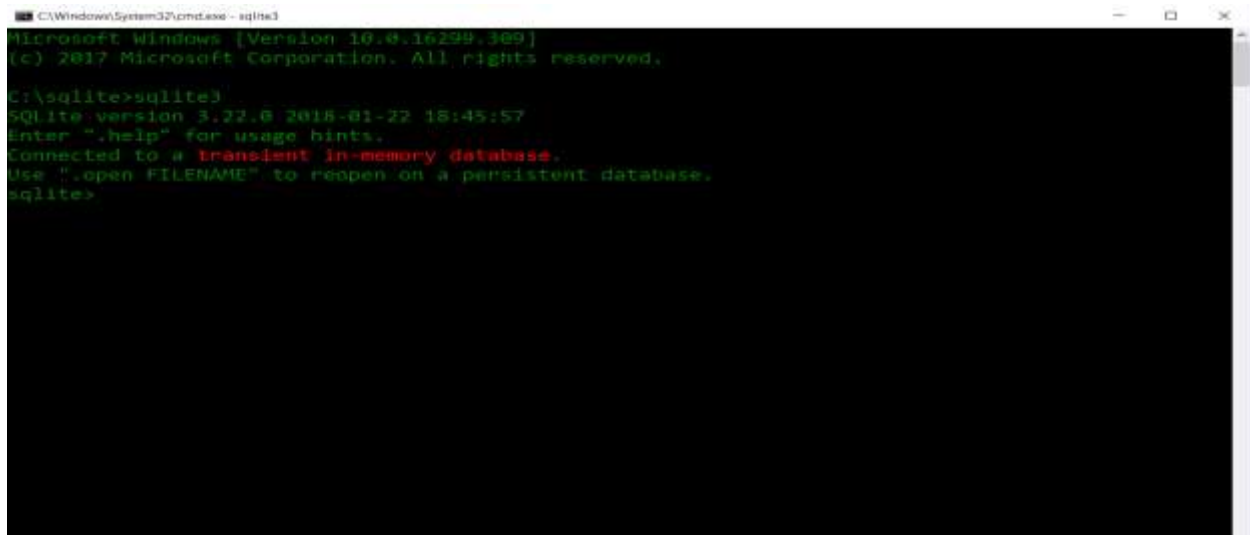
SQL	SQLite
Main components of SQL are Data Definition Language(DDL), Data Manipulation Language(DML), Data Control Language(DCL).	SQLite supports many features of SQL and has high performance but does not support stored procedures.
SQL is Structured Query Language which is used with databases like MySQL, Oracle, Microsoft SQL Server, IBM DB2, etc.	SQLite is portable database resource. It could get an extension in whatever programming language used to access that database.
A conventional SQL database needs to be running as service like Oracle DB to connect to and provide lot of functionalities.	SQLite database system does not provide such functionalities.
SQL is query language which is used by other SQL databases. It is not database itself.	SQLite is relational database management system itself which uses SQL.

Installation on Windows

Installation on Windows:

1. Visit the official website of [SQLite](https://www.sqlite.org/) for downloading the zip file.
2. Download that zip file.
3. Create a folder in C or D (wherever you want) for storing SQLite by expanding zip file.
4. Open the command prompt and set the path for the location of SQLite folder given in the previous step. After that write "sqlite3" and press enter.

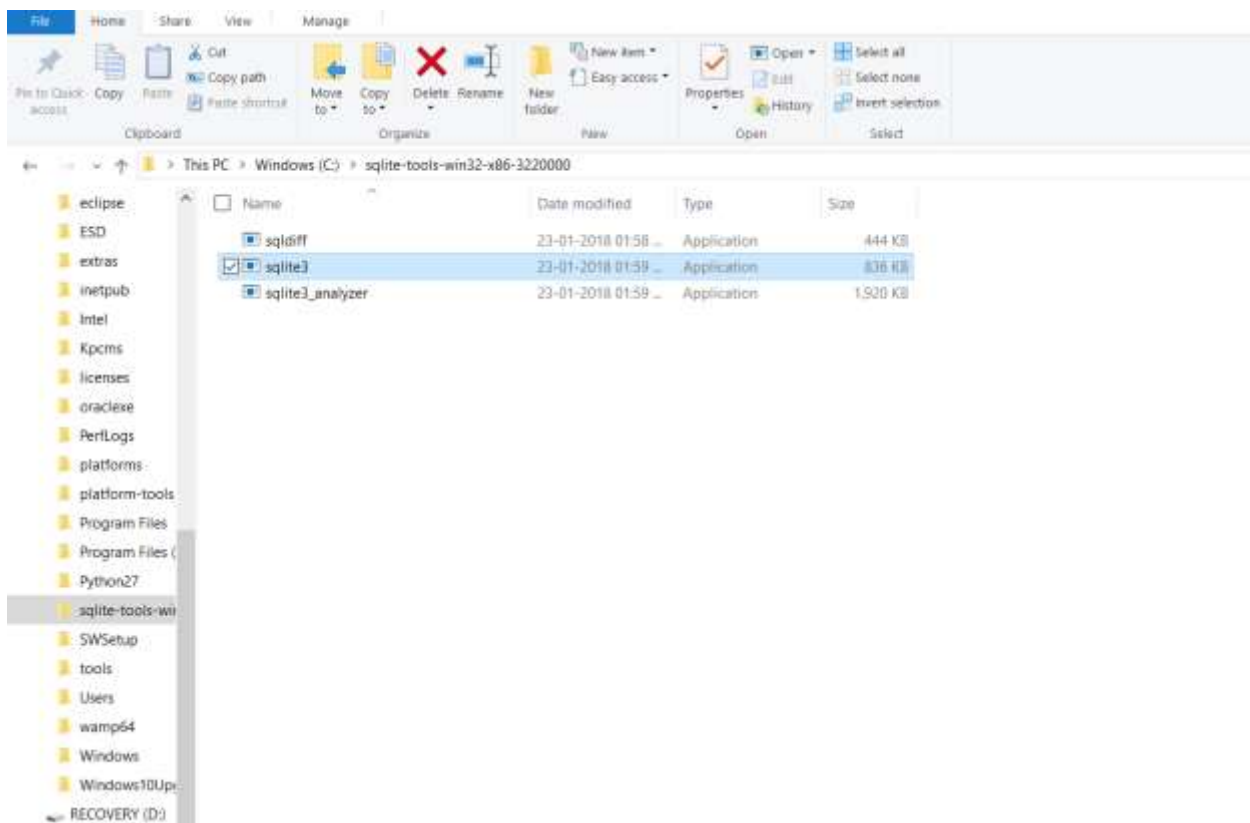
Unit- 1: Introduction to SQLite



```
C:\Windows\System32\cmd.exe - sqlite3
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

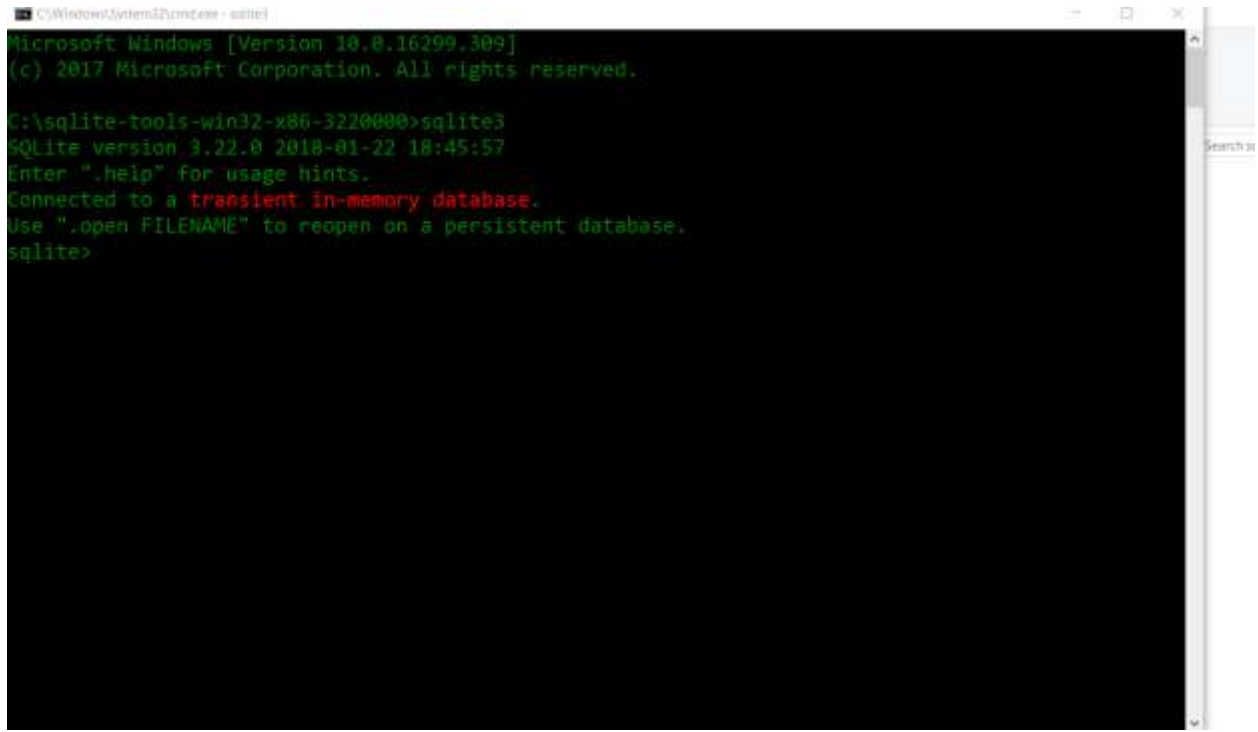
C:\sqlite>sqlite3
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

You can also directly open the .exe file from the folder where you have stored the SQLite whole thing.



Unit- 1: Introduction to SQLite

After clicking on the selected .exe file it will open SQLite application

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\System32\cmd.exe - sqlite'. The window content displays the following text: 'Microsoft Windows [Version 10.0.16299.309] (c) 2017 Microsoft Corporation. All rights reserved. C:\sqlite-tools-win32-x86-3220080>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>'.

```
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\sqlite-tools-win32-x86-3220080>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>
```

Features Of SQLite

- **SQLite is totally free:** SQLite is open-source. So, no license is required to work with it.
- **SQLite is serverless:** SQLite doesn't require a different server process or system to operate.
- **SQLite is very flexible:** It facilitates you to work on multiple databases on the same session on the same time.
- **Configuration Not Required:** SQLite doesn't require configuration. No setup or administration required.
- **SQLite is a cross-platform DBMS:** You don't need a large range of different platforms like Windows, Mac OS, Linux, and Unix. It can also be

Unit- 1: Introduction to SQLite

used on a lot of embedded operating systems like Symbian, and Windows CE.

- **Storing data is easy:** SQLite provides an efficient way to store data.
- **Variable length of columns:** The length of the columns is variable and is not fixed. It facilitates you to allocate only the space a field needs. For example, if you have a varchar (200) column, and you put a 10 characters' length value on it, then SQLite will allocate only 10 characters' space for that value not the whole 200 space.
- **Provide large number of API's:** SQLite provides API for a large range of programming languages. **For example:** .Net languages (Visual Basic, C#), PHP, Java, Objective C, Python and a lot of other programming language.
- **SQLite** is written in **ANSI-C** and provides simple and easy-to-use API.
- **SQLite** is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

SQLite Advantages

SQLite is a very popular database which has been successfully used with on disk file format for desktop applications like version control systems, financial analysis tools, media cataloging and editing suites, CAD packages, record keeping programs etc.

There are a lot of advantages to use SQLite as an application file format:

1) Lightweight

- SQLite is a very light weighted database so, it is easy to use it as an embedded software with devices like televisions, Mobile phones, cameras, home electronic devices, etc.

2) Better Performance

Unit- 1: Introduction to SQLite

- Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system.
- It only loads the data which is needed, rather than reading the entire file and hold it in memory.
- If you edit small parts, it only overwrite the parts of the file which was changed.

3) No Installation Needed

- SQLite is very easy to learn. You don't need to install and configure it. Just download SQLite libraries in your computer and it is ready for creating the database.

4) Reliable

- It updates your content continuously so, little or no work is lost in a case of power failure or crash.
- SQLite is less bugs prone rather than custom written file I/O codes.
- SQLite queries are smaller than equivalent procedural codes so, chances of bugs are minimal.

5) Portable

- SQLite is portable across all 32-bit and 64-bit operating systems and big- and little-endian architectures.
- Multiple processes can be attached with same application file and can read and write without interfering each other.
- It can be used with all programming languages without any compatibility issue.

6) Accessible

- SQLite database is accessible through a wide variety of third-party tools.

Unit- 1: Introduction to SQLite

- SQLite database's content is more likely to be recoverable if it has been lost. Data lives longer than code.

7) Reduce Cost and Complexity

- It reduces application cost because content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural queries.
- SQLite can be easily extended in future releases just by adding new tables and/or columns. It also preserve the backwards compatibility.

SQLite Disadvantages

- SQLite is used to handle low to medium traffic HTTP requests.
- Database size is restricted to 2GB in most cases.

SQLite Commands

SQLite commands are similar to SQL commands. There are three types of SQLite commands:

- **DDL** : Data Definition Language
- **DML** : Data Manipulation Language
- **DQL** : Data Query Language

Data Definition Language

There are three commands in this group:

CREATE: This command is used to create a table, a view of a table or other object in the database.

ALTER: It is used to modify an existing database object like a table.

Unit- 1: Introduction to SQLite

DROP: The DROP command is used to delete an entire table, a view of a table or other object in the database.

Data Manipulation language

There are three commands in data manipulation language group:

INSERT : This command is used to create a record.

UPDATE : It is used to modify the records.

DELETE : It is used to delete records.

Data Query Language

SELECT: This command is used to retrieve certain records from one or more table.

SQLite Data Types

Most SQL database engines (every SQL database engine other than SQLite, as far as we know) uses static, rigid typing. With static typing, the datatype of a value is determined by its container - the particular column in which the value is stored.

SQLite uses a more general dynamic type system. In SQLite, the datatype of a value is associated with the value itself, not with its container. The dynamic type system of SQLite is backwards compatible with the more common static type systems of other database engines in the sense that SQL statements that work on statically typed databases should work the same way in SQLite. However, the dynamic typing in SQLite allows it to do things which are not possible in traditional rigidly typed databases.

SQLite Storage Classes

Unit- 1: Introduction to SQLite

NULL	It specifies that the value is a null value.
INTEGER	It specifies the value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
REAL	It specifies the value is a floating point value, stored as an 8-byte IEEE floating point number.
text	It specifies the value is a text string, stored using the database encoding (utf-8, utf-16be or utf-16le)
BLOB	It specifies the value is a blob of data, stored exactly as it was input.

SQLite Affinity Types

SQLite supports type affinity for columns. Any column can still store any type of data but the preferred storage class for a column is called its affinity.

TEXT	This column is used to store all data using storage classes NULL, TEXT or BLOB.
NUMERIC	This column may contain values using all five storage classes.
INTEGER	It behaves the same as a column with numeric affinity with an exception in a cast expression.
REAL	It behaves like a column with numeric affinity except that it forces integer values into floating point representation
NONE	A column with affinity NONE does not prefer one storage class over another and don't persuade data from one storage class into another.

SQLite Transaction

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

Unit- 1: Introduction to SQLite

A transaction is the propagation of one or more changes to the database. For example, if you are creating, updating, or deleting a record from the table, then you are performing transaction on the table. It is important to control transactions to ensure data integrity and to handle database errors.

Practically, you will club many SQLite queries into a group and you will execute all of them together as part of a transaction.

Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym ACID.

- **Atomicity** – Ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state.
- **Consistency** – Ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation** – Enables transactions to operate independently of and transparent to each other.
- **Durability** – Ensures that the result or effect of a committed transaction persists in case of a system failure.

Following are the following commands used to control transactions:

- **BEGIN TRANSACTION** – To start a transaction.
- **COMMIT** – To save the changes, alternatively you can use **END TRANSACTION** command.
- **ROLLBACK** – To rollback the changes.

Transactional control commands are only used with DML commands INSERT, UPDATE, and DELETE. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

BEGIN TRANSACTION Command

Transactions can be started using BEGIN TRANSACTION or simply BEGIN command. Such transactions usually persist until the next COMMIT or ROLLBACK command is encountered. However, a transaction will also ROLLBACK if the database is closed or if an error occurs.

Following is the simple syntax to start a transaction.

```
BEGIN;  
or  
BEGIN TRANSACTION;
```

SQLite COMMIT Command

COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

Following is the syntax for COMMIT command.

```
COMMIT;  
or  
END TRANSACTION;
```

ROLLBACK Command

ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

Following is the syntax for ROLLBACK command.

ROLLBACK

Data Filtering and Triggers

Filtering:

Distinct

The Distinct clause is an optional clause of the select statement. The Distinct clause allows you to remove the duplicate rows in the result set

The following statement illustrates the syntax of the Distinct clause

```
SELECT DISTINCT  
Select list;  
FROM table;  
Where:
```

The where clause is an optional clause of the select statement. It appears after the from clause as the following statement:

```
SELECT  
    Column list  
FROM  
    table  
WHERE  
    Search condition;
```

Between

The between operator is a logical operator that tests whether a value is in range of values. If the value is in the specified range, the between operator returns true. The between operator can be used in the where clause of the select delete, update and replace statements.

The following illustrates the syntax of the SQLite between operator:

```
test _expression  
BETWEEN  
Low _expression AND high _expression;
```

In

The SQLite in operator determines whether a value matches any value in a list or a subquery. The syntax of the in operator is as follows:

```
expression [NOT] IN (value list | subquery);
```

like

Sometimes, you don't know exactly the complete keyword that you want to query. For example, you may know

that your most favorite song contains the word , elevtor but you don't know exactly the name.

Unit- 1: Introduction to SQLite

To query data based on partial information, you use the like operator in the where clause of the select statement as follows:

```
SELECT
    Column _list
FROM
    Table _name
WHERE
    column_1 LIKE pattern;
```

Union

Sometimes, you need to combine data from multiple tables into a complete result set. It may be for tables with similar data within the same database or maybe you need to combine similar data from multiple databases.

To combine rows from two or more queries into a single result set, you use SQLite union operator. The following illustrates the basic syntax of the union operator:

```
query_1
UNION [ALL]
query_2
UNION [ALL]
query_3
...;
```

Intersect

SQLite intersect operator compares the result sets of two queries and returns distinct rows that are output by both queries.

The following illustrates the syntax of the intersect operator:

SQLite intersect operator compares the result sets of two queries and returns distinct rows that are output by both queries.

The following illustrates the syntax of the intersect operator:

```
SELECT select_list1
FROM table1
INTERSECT
SELECT select_list2
FROM table2
```

Except

SQLite except operator compares the result sets of two queries and returns distinct rows from the left query that are not output by the right query.

The following shows the syntax of the except operator:

```
SELECT select_list1
FROM table1
EXCEPT
```



```
SELECT select_list2  
FROM table2
```

Limit

The limit clause is an optional part of the select statement. You use the limit clause to constrain the number of rows returned by the query.

For example, a select statement may return one million rows. However, if you just need the first 10 rows in the result set, you can add the limit clause to the select statement to retrieve 10 rows.

The following illustrates the syntax of the limit clause.

```
SELECT  
    Column list  
FROM  
    table  
LIMIT row count;
```

IS NULL

null is special. It indicates that a piece of information is unknown or not applicable.

For example, some songs may not have the songwriter information because we don't know who wrote them.

To store these unknown songwriters along with the songs in a database table, we must use NULL.

NULL is not equal to anything even the number zero, an empty string, and so on.

Especially, NULL is not equal to itself. The following expression returns 0:

```
NULL = NULL
```

Having

SQLite having clause is an optional clause of the having statement. The having clause specifies a search condition for a group.

You often use the having clause with the group by clause. The group by clause groups a set of rows into a set of summary rows or groups. Then the having clause filters groups based on a specified condition.

If you use the having clause, you must include the group by clause; otherwise, you will get the following error:

BY clause is required before HAVING

The following illustrates the syntax of the having clause:

```
SELECT
    column_1,
    column_2,
    aggregate _function (column_3)
FROM
    table
GROUP BY
    column_1,
    column_2
HAVING
    search_condition;
```

Group by

Unit- 1: Introduction to SQLite

The group by clause is an optional clause of the select statement. The group by clause a selected group of rows into summary rows by values of one or more columns.

The group by clause returns one row for each group. For each group, you can apply an aggregate function such as min, max, sum, count, or avg to provide more information about each group.

The following statement illustrates the syntax of the SQLite group by clause.

```
SELECT
    column_1,
    aggregate _function(column_2)
FROM
    table
GROUP BY
    column_1,
    column_2;
```

Order by

SQLite stores data in the tables in an unspecified order. It means that the rows in the table may or may not be in the order that they were inserted.

If you use the select statement to query data from a table, the order of rows in the result set is unspecified.

To sort the result set, you add the order by clause to the select statement as follows:

```
SELECT
```

```
select_list  
FROM  
table  
ORDER BY  
column_1 ASC,  
column_2 DESC;
```

Conditional Logic (CASE)

The SQLite case expression evaluates a list of conditions and returns an expression based on the result of the evaluation.

The case expression is similar to the IF-THEN-ELSE statement in other programming languages.

You can use the case expression in any clause or statement that accepts a valid expression. For example, you can use the case expression in clauses such as where, order by, having, select and statements such as select, update and delete.

SQLite provides two forms of the case expression: simple case and searched case.

```
CASE case expression  
  WHEN when_expression_1 THEN result_1  
  WHEN when_expression_2 THEN result_2  
  ...  
  [ ELSE result else ]  
END
```

SQLite joins

Inner joins

Unit- 1: Introduction to SQLite

In relational databases, data is often distributed in many related tables. A table is associated with another table using foreign keys.

To query data from multiple tables, you use Inner join clause. The Inner join clause combines columns from correlated tables.

Suppose you have two tables: A and B.

A has a1, a2, and f columns. B has b1, b2, and f column. The A table links to the B table using a foreign key column named f.

The following illustrates the syntax of the inner join clause:

```
SELECT a1, a2, b1, b2
FROM A
INNER JOIN B on B.f = A.f;
```

left join

Similar to the inner join clause, the left join clause is an optional clause of the select statement. You use the left join clause to query data from multiple related tables.

Suppose we have two tables: A and B.

- A has m and f columns.
- B has n and f columns.

To perform join between A and B using left join clause, you use the following statement:

```
SELECT
a,
```

```
        b
FROM
        A
LEFT JOIN B ON A .f = B .f
WHERE search_ condition;
```

Cross join

If you use a left join, inner join, or cross join without the on or using clause, SQLite produces the cartesian product of the involved tables. The number of rows in the Cartesian product is the product of the number of rows in each involved tables.

Suppose, we have two tables A and B. The following statements perform the cross join and produce a cartesian product of the rows from the A and B tables.

```
SELECT *
FROM A
INNER JOIN B;
```

Self join

The self-join is a special kind of joins that allow you to join a table to itself using either left join or inner join clause. You use self-join to create a result set that joins the rows with the other rows within the same table.

Because you cannot refer to the same table more than one in a query, you need to use a table alias to assign the table a different name when you use self-join.

Unit- 1: Introduction to SQLite

The self-join compares values of the same or different columns in the same table. Only one table is involved in the self-join.

You often use self-join to query parents / child relationship stored in a table or to obtain running totals.

Full outer joins

The result of the FULL OUTER JOIN is a combination of a left join and a RIGHT JOIN. The result set of the full outer join has NULL values for every column of the table that does not have a matching row in the other table. For the matching rows, the FULL OUTER JOIN produces a single row with values from columns of the rows in both tables.

SQLite Trigger:

Concepts of Trigger

An SQLite trigger is a named database object that is executed automatically when an insert, update or delete statement is issued against the associated table.

create trigger:

statement as follows:

CREATE TRIGGER [IF NOT EXISTS] trigger name

Unit- 1: Introduction to SQLite

```
[BEFORE|AFTER|INSTEAD OF]
[INSERT|UPDATE|DELETE]
ON table name
[WHEN condition]
BEGIN
    statements;
END;
```

Example of Before and After trigger

BEFORE INSERT

```
CREATE TRIGGER validate_email_before_insert_leads
BEFORE INSERT ON leads
BEGIN
    SELECT
    CASE
        WHEN NEW.email NOT LIKE '%_@__%. __%' THEN
            RAISE (ABORT,' Invalid email address')
    END;
END;
```

- **AFTER UPDATE**

```
CREATE TABLE lead_logs (
    id INTEGER PRIMARY KEY,
    old_id int,
    new_id int,
```


Unit- 1: Introduction to SQLite

```
old _phone text,  
new _phone text,  
old _email text,  
new _email text,  
user _action text,  
created _at text
```

```
);
```

DROP TRIGGER

To drop an existing trigger, you use the DROP TRIGGER statement as follows:

```
DROP TRIGGER [IF EXISTS] trigger name;
```
