SQLite Where

Summary: in this tutorial, you will learn how to use SQLite `WHERE` clause to specify the search condition for rows returned by the query.

Introduction to SQLite WHERE clause

The `WHERE` clause is an optional clause of the `SELECT` (https://www.sqlitetutorial.net/sqlite-select/) statement. It appears after the `FROM` clause as the following statement:

```
SELECT
        column_list
FROM
        table
WHERE
        search_condition;
```

In this example, you add a `WHERE` clause to the `SELECT` statement to filter rows returned by the query. When evaluating a `SELECT` statement with a `WHERE` clause, SQLite uses the following steps:

1. First, check the table in the `FROM` clause.
2. Second, evaluate the conditions in the `WHERE` clause to get the rows that met these conditions.
3. Third, make the final result set based on the rows in the previous step with columns in the `SELECT` clause.

The search condition in the `WHERE` has the following form:

```
left_expression COMPARISON_OPERATOR right_expression
```

For example, you can form a search condition as follows:

```
WHERE column_1 = 100;

WHERE column_2 IN (1,2,3);

WHERE column_3 LIKE 'An%';

WHERE column_4 BETWEEN 10 AND 20;
```

Besides the `SELECT` statement, you can use the `WHERE` clause in the `UPDATE` (https://www.sqlitetutorial.net/sqlite-update/) and `DELETE` (https://www.sqlitetutorial.net/sqlite-delete/) statements.

SQLite comparison operators

A comparison operator tests if two expressions are the same. The following table illustrates the comparison operators that you can use to construct expressions:

Operator	**Meaning**
=	Equal to
<> or !=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

SQLite logical operators

Logical operators allow you to test the truth of some expressions. A logical operator returns 1, 0, or a NULL value.

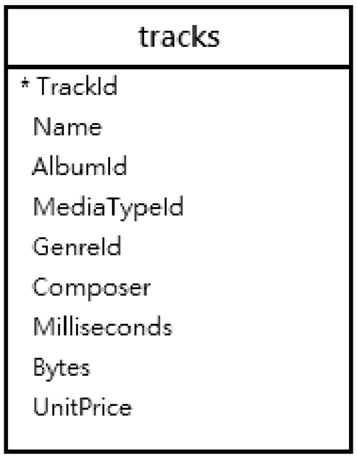
Notice that SQLite does not provide Boolean data type therefore 1 means TRUE, and 0 means FALSE.

The following table illustrates the SQLite logical operators:

Operator	Meaning
ALL	returns 1 if all expressions are 1.
AND	returns 1 if both expressions are 1, and 0 if one of the expressions is 0.
ANY	returns 1 if any one of a set of comparisons is 1.
BETWEEN (https://www.sqlitetutorial.net/sqlite-between/)	returns 1 if a value is within a range.
EXISTS (https://www.sqlitetutorial.net/sqlite-exists/)	returns 1 if a subquery contains any rows.
IN (https://www.sqlitetutorial.net/sqlite-in/)	returns 1 if a value is in a list of values.
LIKE (https://www.sqlitetutorial.net/sqlite-like/)	returns 1 if a value matches a pattern
NOT	reverses the value of other operators such as NOT EXISTS, NOT IN, NOT BETWEEN, etc.
OR	returns true if either expression is 1

SQLite WHERE clause examples

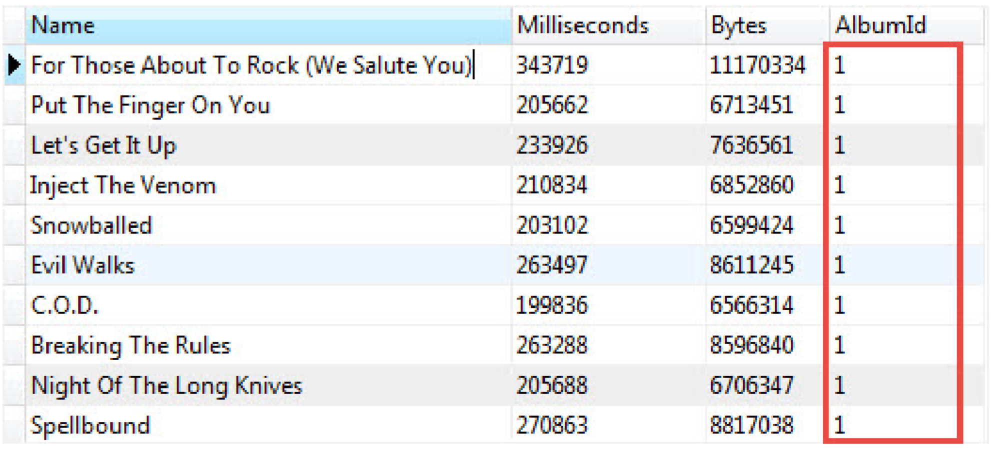
We will use the `tracks` table in the sample database (https://www.sqlitetutorial.net/sqlite-sample-database/) to illustrate how to use the `WHERE` clause.



The equality operator (`=`) is the most commonly used operator. For example, the following query uses the `WHERE` clause the equality operator to find all the tracks in the album id 1:

```
SELECT
    name,
    milliseconds,
    bytes,
    albumid
FROM
    tracks
WHERE
    albumid = 1;
```

Try It ›



Name	Milliseconds	Bytes	AlbumId
For Those About To Rock (We Salute You)	343719	11170334	1
Put The Finger On You	205662	6713451	1
Let's Get It Up	233926	7636561	1
Inject The Venom	210834	6852860	1
Snowballed	203102	6599424	1
Evil Walks	263497	8611245	1
C.O.D.	199836	6566314	1
Breaking The Rules	263288	8596840	1
Night Of The Long Knives	205688	6706347	1
Spellbound	270863	8817038	1

SQLite compares the values stored in the `AlbumId` column with a literal value `1` to test if they are equal. Only the rows that satisfy the condition are returned.

When you compare two values, you must ensure that they are the same data type. You should compare numbers with numbers, string with strings, etc.

In case you compare values in different data types e.g., a string with a number, SQLite has to perform implicit data type conversions, but in general, you should avoid doing this.

You use the logical operator to combine expressions. For example, to get tracks of the album 1 that have the length greater than 200,000 milliseconds, you use the following statement:

```
SELECT
	name,
	milliseconds,
	bytes,
	albumid
FROM
	tracks
WHERE
	albumid = 1
AND milliseconds > 250000;
```

Try It ›

The statement used two expressions `albumid = 1` and `milliseconds > 250000`. It uses the `AND` logical operator to combine these expressions.

(https://www.sqlitetutorial.net/wp-content/uploads/2015/11/SQLite-WHERE-multiple-conditions.jpg)

SQLite WHERE clause with LIKE operator example

Sometimes, you may not remember exactly the data that you want to search. In this case, you perform an inexact search using the `LIKE` (https://www.sqlitetutorial.net/sqlite-like/) operator.

For example, to find which tracks composed by Smith, you use the `LIKE` operator as follows:

```
SELECT
	name,
	albumid,
	composer
FROM
	tracks
WHERE
	composer LIKE '%Smith%'
ORDER BY
	albumid;
```

Try It ›

You get tracks composed by R.A. Smith-Diesel, Adrian Smith, etc.

SQLite WHERE clause with the IN operator example

The `IN` (https://www.sqlitetutorial.net/sqlite-in/) operator allows you to check whether a value is in a list of a comma-separated list of values. For example, to find tracks that have media type id is 2 or 3, you use the `IN` operator as shown in the following statement:

```
SELECT
        name,
        albumid,
        mediatypeid
FROM
        tracks
WHERE
        mediatypeid IN (2, 3);
```

Try It ›

In this tutorial, you have learned how to use the SQLite `WHERE` clause to filter rows in the final result set using comparison and logical operators.