SQLite Left Join

Summary: in this tutorial, you will learn how to use SQLite `LEFT JOIN` clause to query data from multiple tables.

Introduction to SQLite LEFT JOIN clause

Similar to the INNER JOIN (https://www.sqlitetutorial.net/sqlite-inner-join/) clause, the `LEFT JOIN` clause is an optional clause of the SELECT (https://www.sqlitetutorial.net/sqlite-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;
```

The expression `A.f = B.f` is a conditional expression. Besides the equality (=) operator, you can use other comparison operators such as greater than (>), less than (<), etc.

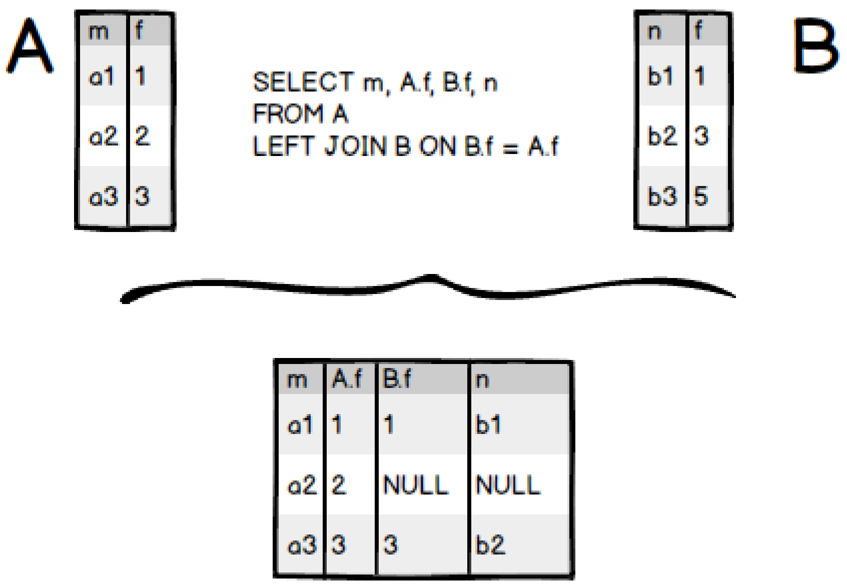
The statement returns a result set that includes:

1. Rows in table A (left table) that have corresponding rows in table B.
2. Rows in the table A table and the rows in the table B filled with NULL values in case the row from table A does not have any corresponding rows in table B.

In other words, all rows in table A are included in the result set whether there are matching rows in table B or not.

In case you have a WHERE (https://www.sqlitetutorial.net/sqlite-where/) clause in the statement, the `search_condition` in the `WHERE` clause is applied after the matching of the `LEFT JOIN` clause completes.

See the following illustration of the `LEFT JOIN` clause between the A and B tables.



m	f
a1	1
a2	2
a3	3

```
SELECT m, A.f, B.f, n
FROM A
LEFT JOIN B ON B.f = A.f
```

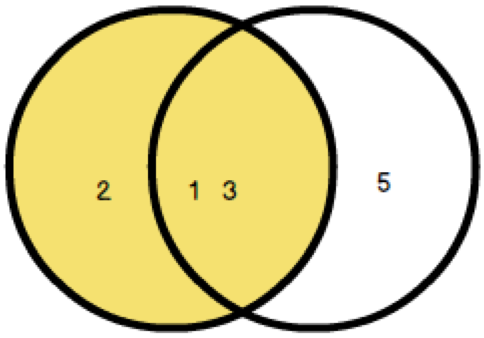
n	f
b1	1
b2	3
b3	5

m	A.f	B.f	n
a1	1	1	b1
a2	2	NULL	NULL
a3	3	3	b2

All rows in the table A are included in the result set.

Because the second row `(a2,2)` does not have a corresponding row in table B, the `LEFT JOIN` clause creates a *fake row* filled with `NULL`.

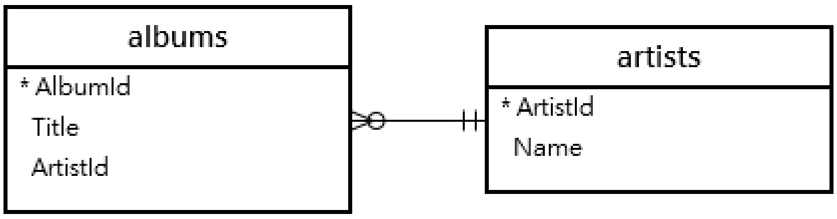
The following Venn Diagram illustrates the `LEFT JOIN` clause.



It is noted that `LEFT OUTER JOIN` is the same as `LEFT JOIN`.

SQLite LEFT JOIN examples

We will use the `artists` and `albums` tables in the sample database (https://www.sqlitetutorial.net/sqlite-sample-database/) for demonstration.



One album belongs to one artist. However, one artist may have zero or more albums.

To find artists who do not have any albums by using the `LEFT JOIN` clause, we select artists and their corresponding albums. If an artist does not have any albums, the value of the `AlbumId` column is `NULL.`

To display the artists who do not have any albums first, we have two choices:

- First, use `ORDER BY` (https://www.sqlitetutorial.net/sqlite-order-by/) clause to list the rows whose `AlbumId` is `NULL` values first.
- Second, use `WHERE` (https://www.sqlitetutorial.net/sqlite-where/) clause and `IS NULL` (https://www.sqlitetutorial.net/sqlite-is-null/) operator to list only artists who do not have any albums.

The following statement uses the `LEFT JOIN` clause with the `ORDER BY` clause.

```
SELECT
   artists.ArtistId,
   AlbumId
FROM
   artists
LEFT JOIN albums ON
   albums.ArtistId = artists.ArtistId
ORDER BY
   AlbumId;
```

The following statement uses the `LEFT JOIN` clause with the WHERE (https://www.sqlitetutorial.net/sqlite-where/) clause.

```
SELECT
   artists.ArtistId
   , AlbumId
FROM
   artists
LEFT JOIN albums ON
   albums.ArtistId = artists.ArtistId
WHERE
   AlbumId IS NULL;
```

In this tutorial, you have learned how to use SQLite `LEFT JOIN` clause to query data from multiple tables.