SQLITE TUTORIAL

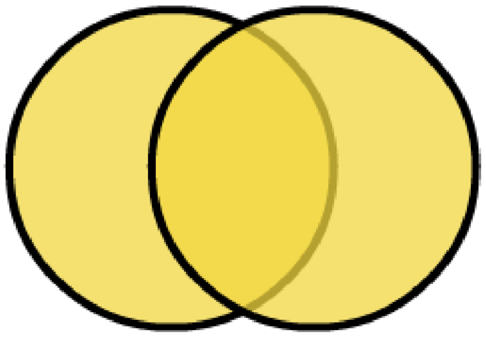
SQLite FULL OUTER JOIN Emulation

Summary: in this tutorial, you will learn how to emulate SQLite full outer join using the `UNION` and `LEFT JOIN` clauses.

Introduction to SQL FULL OUTER JOIN clause

In theory, the result of the `FULL OUTER JOIN` is a combination of a `LEFT JOIN` (https://www.sqlitetutorial.net/sqlite-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.

The following picture illustrates the result of the `FULL OUTER JOIN` clause:



See the following `cats` and `dogs` tables.

```
-- create and insert data into the dogs table
CREATE TABLE dogs (
    type       TEXT,
    color TEXT
);

INSERT INTO dogs(type, color)
VALUES('Hunting','Black'), ('Guard','Brown');

-- create and insert data into the cats table
CREATE TABLE cats (
    type       TEXT,
    color TEXT
);

INSERT INTO cats(type,color)
VALUES('Indoor','White'),
      ('Outdoor','Black');
```

The following statement uses the `FULL OUTER JOIN` clause to query data from the `dogs` and `cats` tables.

```
SELECT *
FROM dogs
FULL OUTER JOIN cats
    ON dogs.color = cats.color;
```

The following shows the result of the statement above:

Type	Color	Type	Color
Hunting	Black	Outdoor	Black

Type	Color	Type	Color
Guard	Brown	NULL	NULL
NULL	NULL	Indoor	White

Unfortunately, SQLite does not support the `RIGHT JOIN` clause and also the `FULL OUTER JOIN` clause. However, you can easily emulate the `FULL OUTER JOIN` by using the `LEFT JOIN` clause.

Emulating SQLite full outer join

The following statement emulates the `FULL OUTER JOIN` clause in SQLite:

```
SELECT d.type,
         d.color,
         c.type,
         c.color
FROM dogs d
LEFT JOIN cats c USING(color)
UNION ALL
SELECT d.type,
         d.color,
         c.type,
         c.color
FROM cats c
LEFT JOIN dogs d USING(color)
WHERE d.color IS NULL;
```

How the query works.

- Because SQLilte does not support the `RIGHT JOIN` clause, we use the `LEFT JOIN` `(https://www.sqlitetutorial.net/sqlite-left-join/)` clause in the second `SELECT` `(https://www.sqlitetutorial.net/sqlite-select/)` statement instead and switch the positions of the `cats` and `dogs` tables.

- The `UNION ALL` (https://www.sqlitetutorial.net/sqlite-union/) clause retains the duplicate rows from the result sets of both queries.
- The `WHERE` clause in the second `SELECT` statement removes rows that already included in the result set of the first `SELECT` statement.

In this tutorial, you have learned how to use the `UNION ALL` and `LEFT JOIN` clauses to emulate the SQLite `FULL OUTER JOIN` clause.