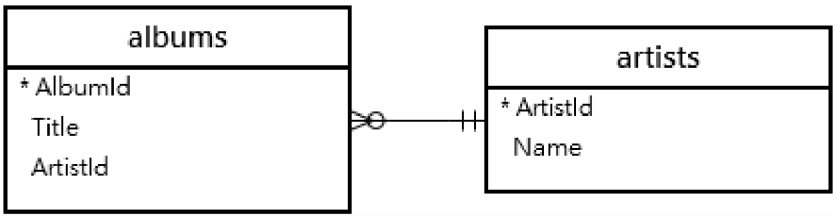
SQLite Join

Summary: in this tutorial, you will learn about various kinds of SQLite joins to query data from two or more tables.

For the demonstration, we will use the `artists` and `albums` tables from the sample database (https://www.sqlitetutorial.net/sqlite-sample-database/) .



An artist can have zero or many albums while an album belongs to one artist.

To query data from both `artists` and `albums` tables, you use can use an `INNER JOIN` (https://www.sqlitetutorial.net/sqlite-inner-join/) , `LEFT JOIN` (https://www.sqlitetutorial.net/sqlite-left-join/) , or `CROSS JOIN` (https://www.sqlitetutorial.net/sqlite-cross-join/) clause. Each join clause determines how SQLite uses data from one table to match with rows in another table.

Note that SQLite doesn't directly support the `RIGHT JOIN` and `FULL OUTER JOIN` (https://www.sqlitetutorial.net/sqlite-full-outer-join/) .

SQLite INNER JOIN

The following statement returns the album titles and their artist names:

```
SELECT
    Title,
    Name
FROM
    albums
INNER JOIN artists
    ON artists.ArtistId = albums.ArtistId;
```

Here is the partial output:

Title	Name
For Those About To Rock We Salute You	AC/DC
Balls to the Wall	Accept
Restless and Wild	Accept
Let There Be Rock	AC/DC
Big Ones	Aerosmith
Jagged Little Pill	Alanis Morissette
Facelift	Alice In Chains
Warner 25 Anos	Antônio Carlos Jobim
Plays Metallica By Four Cellos	Apocalyptica
Audioslave	Audioslave
Out Of Exile	Audioslave
BackBeat Soundtrack	BackBeat
The Best Of Billy Cobham	Billy Cobham
Alcohol Fueled Brewtality Live! [Disc 1]	Black Label Society
Alcohol Fueled Brewtality Live! [Disc 2]	Black Label Society
Black Sabbath	Black Sabbath
Black Sabbath Vol. 4 (Remaster)	Black Sabbath

In this example, the `INNER JOIN` clause matches each row from the `albums` table with every row from the `artists` table based on the join condition (`artists.ArtistId = albums.ArtistId`) specified after the `ON` keyword.

If the join condition evaluates to true (or 1), the columns of rows from both `albums` and `artists` tables are included in the result set.

This query uses table aliases (`l` for the `albums` table and `r` for `artists` table) to shorten the query:

```
SELECT
    l.Title,
    r.Name
FROM
    albums l
INNER JOIN artists r ON
    r.ArtistId = l.ArtistId;
```

In case the column names of joined tables are the same e.g., `ArtistId` , you can use the `USING` syntax as follows:

```
SELECT
   Title,
```

```
    Name
FROM
    albums
INNER JOIN artists USING(ArtistId);
```

The clause `USING(ArtistId)` is equipvalent to the clause `ON artists.ArtistId = albums.ArtistId` .

SQLite LEFT JOIN

This statement selects the artist names and album titles from the `artists` and `albums` tables using the `LEFT JOIN` clause:

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

Here is the output:

Name	Title
A Cor Do Som	[NULL]
AC/DC	For Those About To Rock We Salute You
AC/DC	Let There Be Rock
Aaron Copland & London Symphony Orchestra	A Copland Celebration, Vol. I
Aaron Goldberg	Worlds
Academy of St. Martin in the Fields & Sir Neville Marriner	The World of Classical Favourites
Academy of St. Martin in the Fields Chamber Ensemble &	Sir Neville Marriner: A Celebration
Academy of St. Martin in the Fields, John Birch, Sir Neville	Fauré: Requiem, Ravel: Pavane & Others
Academy of St. Martin in the Fields, Sir Neville Marriner &	Bach: Orchestral Suites Nos. 1 - 4
Academy of St. Martin in the Fields, Sir Neville Marriner &	[NULL]
Accept	Balls to the Wall
Accept	Restless and Wild
Adrian Leaper & Doreen de Feis	Górecki: Symphony No. 3
Aerosmith	Big Ones
Aerosmith & Sierra Leone's Refugee Allstars	[NULL]
Aisha Duo	Quiet Songs
Alanis Morissette	Jagged Little Pill

The `LEFT JOIN` clause selects data starting from the left table (`artists`) and matching rows in the right table (`albums`) based on the join condition (`artists.ArtistId = albums.ArtistId`) .

The left join returns all rows from the `artists` table (or left table) and the matching rows from the `albums` table (or right table).

If a row from the left table doesn't have a matching row in the right table, SQLite includes columns of the rows in the left table and `NULL` for the columns of the right table.

Similar to the `INNER JOIN` clause, you can use the `USING` syntax for the join condition as follows:

```
SELECT
   Name,
   Title
FROM
   artists
LEFT JOIN albums USING (ArtistId)
ORDER BY
   Name;
```

If you want to find artists who don't have any albums, you can add a `WHERE` (https://www.sqlitetutorial.net/sqlite-where/) clause as shown in the following query:

```
SELECT
    Name,
    Title
FROM
    artists
LEFT JOIN albums ON
    artists.ArtistId = albums.ArtistId
WHERE Title IS NULL
ORDER BY Name;
```

This picture shows the partial output:

Name	Title
A Cor Do Som	[NULL]
Academy of St. Martin in the Fields, Sir Neville Marriner & William Bennett	[NULL]
Aerosmith & Sierra Leone's Refugee Allstars	[NULL]
Avril Lavigne	[NULL]
Azymuth	[NULL]
Baby Consuelo	[NULL]
Banda Black Rio	[NULL]
Barão Vermelho	[NULL]
Bebel Gilberto	[NULL]
Ben Harper	[NULL]
Big & Rich	[NULL]
Black Eyed Peas	[NULL]
Charlie Brown Jr.	[NULL]
Christina Aguilera featuring BigElf	[NULL]
Corinne Bailey Rae	[NULL]
DJ Dolores & Orchestra Santa Massa	[NULL]

Generally, this type of query allows you to find rows that are available in the left table but don't have corresponding rows in the right table.

Note that `LEFT JOIN` and `LEFT OUTER JOIN` are synonyms.

SQLite CROSS JOIN

The `CROSS JOIN` clause creates a Cartesian product (https://en.wikipedia.org/wiki/Cartesian_product) of rows from the joined tables.

Unlike the `INNER JOIN` and `LEFT JOIN` clauses, a `CROSS JOIN` doesn't have a join condition. Here is the basic syntax of the `CROSS JOIN` clause:

```
SELECT
    select_list
FROM table1
CROSS JOIN table2;
```

The `CROSS JOIN` combines every row from the first table (`table1`) with every row from the second table (`table2`) to form the result set.

If the first table has `N` rows, the second table has `M` rows, the final result will have `NxM` rows.

A practical example of the `CROSS JOIN` clause is to combine two sets of data for forming an initial data set for further processing. For example, you have a list of products and months, and you want to make a plan when you can sell which products.

The following script creates the `products` and `calendars` tables:

```
CREATE TABLE products(
    product text NOT null
);

INSERT INTO products(product)
VALUES('P1'),('P2'),('P3');



CREATE TABLE calendars(
    y int NOT NULL,
    m int NOT NULL
);

INSERT INTO calendars(y,m)
VALUES
    (2019,1),
    (2019,2),
    (2019,3),
    (2019,4),
    (2019,5),
    (2019,6),
    (2019,7),
    (2019,8),
    (2019,9),
    (2019,10),
    (2019,11),
    (2019,12);
```

This query uses the `CROSS JOIN` clause to combine the products with the months:

```
SELECT *
FROM products
CROSS JOIN calendars;
```

Here is the output:

product	y	m
A	2,019	1
A	2,019	2
A	2,019	3
A	2,019	4
A	2,019	5
A	2,019	6
A	2,019	7
A	2,019	8
A	2,019	9
A	2,019	10
A	2,019	11
A	2,019	12
B	2,019	1
B	2,019	2
B	2,019	3
B	2,019	4
B	2,019	5
B	2,019	6
B	2,019	7
B	2,019	8
B	2,019	9
B	2,019	10
B	2,019	11
B	2,019	12
C	2,019	1
C	2,019	2
C	2,019	3
C	2,019	4
C	2,019	5
C	2,019	6
C	2,019	7
C	2,019	8
C	2,019	9
C	2,019	10
C	2,019	11
C	2,019	12

In this tutorial, you have learned various kind of SQLite joins that allow you to query from multiple tables.