Joining Data in SQL

Cheat Sheet

Learn SQL online at www.DataCamp.com

> Definitions used throughout this cheat sheet

Primary key:
A primary key is a field in a table that uniquely identifies each record in the table. In relational databases, primary keys can be used as fields to join tables on.

Foreign key:
A foreign key is a field in a table which references the primary key of another table. In a relational database, one way to join two tables is by connecting the foreign key from one table to the primary key of another.

One-to-one relationship:
Database relationships describe the relationships between records in different tables. When a one-to-one relationship exists between two tables, a given record in one table is uniquely related to exactly one record in the other table.

One-to-many relationship:
In a one-to-many relationship, a record in one table can be related to one or more records in a second table. However, a given record in the second table will only be related to one record in the first table.

Many-to-many relationship:
In a many-to-many relationship, records in a given table 'A' can be related to one or more records in another table 'B', and records in table B can also be related to many records in table A.

> Sample Data

Artist Table

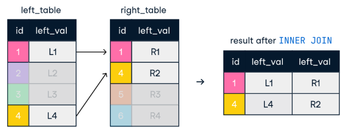
artist_id	name
1	AC/DC
2	Aerosmith
3	Alanis Morissette

Album Table

album_id	title	artist_id
1	For those who rock	1
2	Dream on	2
3	Restless and wild	2
4	Let there be rock	1
5	Rumours	6

INNER JOIN

An inner join between two tables will return only records where a joining field, such as a key, finds a match in both tables.



INNER JOIN join ON one field

```
SELECT *
FROM artist AS art
INNER JOIN album AS alb
ON art.artist_id = alb.artist_id;
```

Result after INNER JOIN:

album_id	name	artist_id
1	AC/DC	1
1	AC/DC	4
2	Aerosmith	2
2	Aerosmith	3

INNER JOIN with USING

```
SELECT *
FROM artist AS art
INNER JOIN album AS alb
USING (artist_id);
```

SELF JOIN

Self-joins are used to compare values in a table to other values of the same table by joining different parts of a table together.

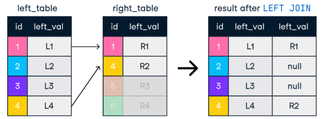
```
SELECT
  art1.artist_id,
  art1.title AS art1_title,
  art2.title AS art2_title
FROM artist as art1
INNER JOIN artist as art2
ON art1.artist_id = art2.album_id;
```

Result after Self join:

artist_id	art1_title	art2_title
1	For those who rock	For those who rock
2	Dream on	Dream on
2	Restless and wild	Dream on
1	Let there be rock	For those who rock

LEFT JOIN

A left join keeps all of the original records in the left table and returns missing values for any columns from the right table where the joining field did not find a match.



LEFT JOIN on one field

```
SELECT *
FROM artist AS art
LEFT JOIN album AS alb
ON art.artist_id = alb.album_id;
```

Result after LEFT JOIN:

artist_id	name	album_id	title	name
1	AC/DC	1	For those who rock	1
1	AC/DC	4	Let there be rock	1
2	Aerosmith	2	Dream on	2
2	Aerosmith	3	Restless and wild	2
3	Alanis Morissette	null	null	null

RIGHT JOIN

A right join keeps all of the original records in the right table and returns missing values for any columns from the left table where the joining field did not find a match. Right joins are far less common than left joins, because right joins can always be re-written as left joins.

RIGHT JOIN on one field

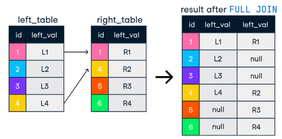
```
SELECT*
FROM artist as art
RIGHT JOIN album AS alb
ON art.artist_id = alb.album_id;
```

Result after RIGHT JOIN:

artist_id	name	album_id	title	name
1	AC/DC	1	For those who rock	1
1	Aerosmith	2	Dream on	2
2	Aerosmith	3	Restless and wild	2
2	AC/DC	4	Let there be rock	1
3	null	5	Rumours	6

FULL JOIN

A full join combines a left join and right join. A full join will return all records from a table, irrespective of whether there is a match on the joining field in the other table, returning null values accordingly.



FULL JOIN on one field

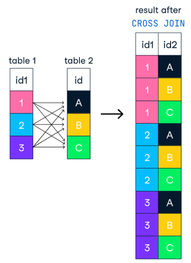
```
SELECT *
FROM artist as art
FULL OUTER JOIN album AS alb
ON art.artist_id = alb.album_id;
```

Result after FULL JOIN:

artist_id	name	album_id	title	name
1	AC/DC	1	For those who rock	1
1	AC/DC	4	Let there be rock	1
2	Aerosmith	2	Balls to the wall	2
2	Aerosmith	3	Restless and wild	2
3	Alanis Morissette	null	null	null
null	null	5	Rumours	6

CROSS JOIN

`CROSS JOIN` creates all possible combinations of two tables. `CROSS JOIN` does not require a field to join `ON`.




```
SELECT name, title
FROM artist
CROSS JOIN album;
```

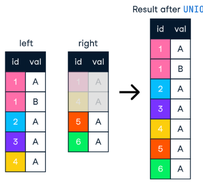
Result after CROSS JOIN:

name	title
AC/DC	For those who rock
AC/DC	Dream on
AC/DC	Restless and wild
AC/DC	Let there be rock
AC/DC	Rumours
Aerosmith	For those who rock
Aerosmith	Dream on
Aerosmith	Restless and wild
Aerosmith	Let there be rock
Aerosmith	Rumours
Alanis Morissette	For those who rock
Alanis Morissette	Dream on
Alanis Morissette	Restless and wild
Alanis Morissette	Let there be rock
Alanis Morissette	Rumours

Set Theory Operators in SQL

UNION

The `UNION` operator is used to vertically combine the results of two `SELECT` statements. For `UNION` to work without errors, all `SELECT` statements must have the same number of columns and corresponding columns must have the same data type. `UNION` does not return duplicates.



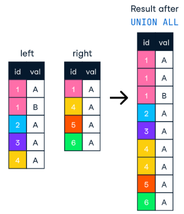

```
SELECT artist_id
FROM artist
UNION
SELECT artist_id
FROM album;
```

Result after UNION:

artist_id
1
2
3
6

UNION ALL

The `UNION ALL` operator works just like `UNION`, but it returns duplicate values. The same restrictions of `UNION` hold true for `UNION ALL`



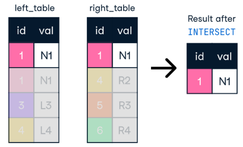

```
SELECT artist_id
FROM artist
UNION ALL
SELECT artist_id
FROM album;
```

Result after UNION ALL:

artist_id
1
2
3
1
2
2
1
6

INTERSECT

The `INTERSECT` operator returns only identical rows from two tables.



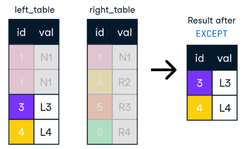

```
SELECT artist_id
FROM artist
INTERSECT
SELECT artist_id
FROM album;
```

Result after INTERSECT:

artist_id
1
2

EXCEPT

The `EXCEPT` operator returns only those rows from the left table that are not present in the right table.



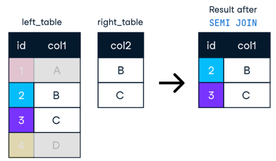

```
SELECT artist_id
FROM artist
EXCEPT
SELECT artist_id
FROM album;
```

Result after EXCEPT:

artist_id
1
2
3

SEMI JOIN

A semi join chooses records in the first table where a condition is met in the second table. A semi join makes use of a `WHERE` clause to use the second table as a filter for the first.



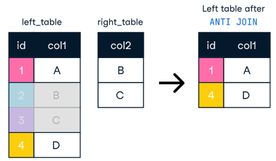

```
SELECT *
FROM album
WHERE artist_id IN
(SELECT artist_id
FROM artist);
```

Result after Semi join:

album_id	title	artist_id
1	For those who rock	1
2	Dream on	2
3	Restless and wild	2

ANTI JOIN

The anti join chooses records in the first table where a condition is `NOT` met in the second table. It makes use of a `WHERE` clause to use exclude values from the second table.




```
SELECT *
FROM album
WHERE artist_id NOT IN
(SELECT artist_id
FROM artist);
```

Result after Anti join:

album_id	title	artist_id
5	Rumours	6