# SQL Joins

# SQL Joins

- A join is a method of linking data between one or more tables based on values of the common column between the tables.
- MySQL supports the following types of joins:
  - Self Join
  - Inner join
  - Left outer join
  - Right outer join
  - Full outer join
  - Cross join

# SELF JOIN

- A self JOIN is a regular join, but the table is joined with itself.
- Self JOIN Syntax

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

# SELF JOIN

**The following SQL statement matches customers that are from the same city:**

```
select A.CustomerName as CN1, B.CustomerName as CN2, A.City
from Customers A, Customers B
where A.CustomerID <> B.CustomerID and
A.City = B.City limit 5;
```

| CN1 | CN2 | City |
|---|---|---|
| Antonio Moreno Taquería | Ana Trujillo Emparedados y helados | México D.F. |
| Centro comercial Moctezuma | Ana Trujillo Emparedados y helados | México D.F. |
| Pericles Comidas clásicas | Ana Trujillo Emparedados y helados | México D.F. |
| Tortuga Restaurante | Ana Trujillo Emparedados y helados | México D.F. |
| Ana Trujillo Emparedados y helados | Antonio Moreno Taquería | México D.F. |

# SAMPLE DATABASE

```
CREATE TABLE members (
member_id INT AUTO_INCREMENT,
name VARCHAR(100),
PRIMARY KEY (member_id) );
```

```
CREATE TABLE committees (
committee_id INT AUTO_INCREMENT,
name VARCHAR(100),
PRIMARY KEY (committee_id) );
```

```
Select * from members;
Select * from committee;
```

```
INSERT INTO members(name)
VALUES('John'),('Jane'),('Mary'),('David'),
('Amelia');
```

```
INSERT INTO committees(name)
VALUES('John'),('Mary'),('Amelia'),('Joe');
```

| member_id | name |
|---|---|
| 1 | John |
| 2 | Jane |
| 3 | Mary |
| 4 | David |
| 5 | Amelia |

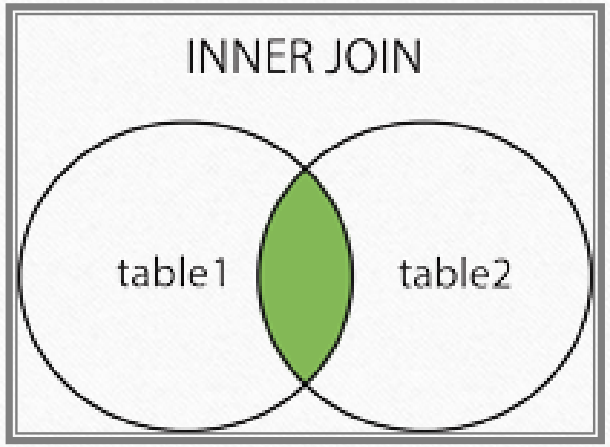
| committee_id | name |
|---|---|
| 1 | John |
| 2 | Mary |
| 3 | Amelia |
| 4 | Joe |

# INNER JOIN

- The INNER JOIN keyword selects records that have matching values in both tables.

INNER JOIN Syntax

- SELECT *column_name(s)*
  FROM *table1*
  INNER JOIN *table2*
  ON *table1.column_name* = *table2.column_name*;

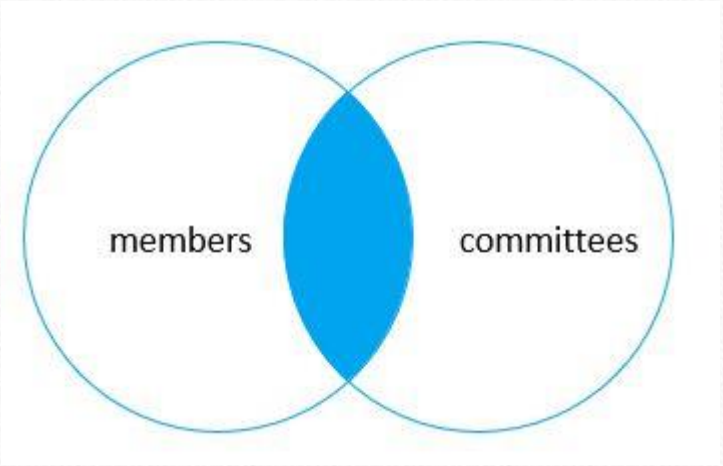


# INNER JOIN

```
SELECT
m.member_id,
m.name members1,
c.committee_id,
c.name committee
FROM members m
INNER JOIN committees c ON
c.name = m.name;
```

| member_id | members1 | committee_id | committee |
|---|---|---|---|
| 1 | John | 1 | John |
| 3 | Mary | 2 | Mary |
| 5 | Amelia | 3 | Amelia |

```
SELECT
m.member_id,
m.name members1,
c.committee_id,
c.name committee
FROM members m
INNER JOIN committees c
using (name);
```



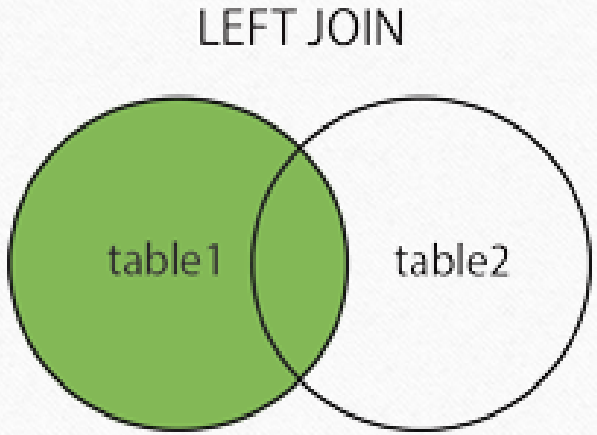
# LEFT JOIN (or LEFT OUTER JOIN)

- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2).
- The result is NULL from the right side, if there is no match.

Left Join Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

In some databases LEFT JOIN is called LEFT OUTER JOIN.

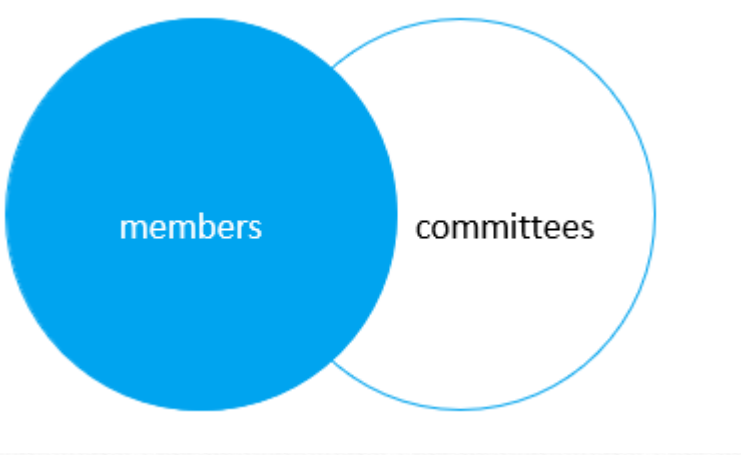


# LEFT JOIN

```
SELECT
m.member_id,
m.name member1,
c.committee_id,
c.name committee
FROM members m
LEFT JOIN committees c USING(name);
```

| | member_id | member 1 | committee_id | committee |
|---|---|---|---|---|
| ▶ | 1 | John | 1 | John |
| | 3 | Mary | 2 | Mary |
| | 5 | Amelia | 3 | Amelia |
| | 2 | Jane | NULL | NULL |
| | 4 | David | NULL | NULL |

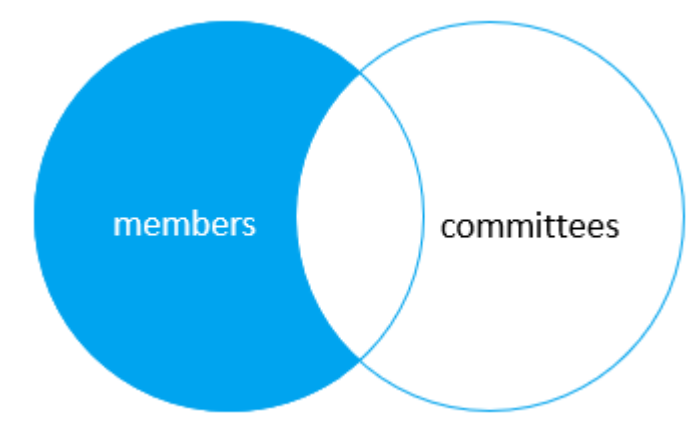
```
SELECT
m.member_id,
m.name member1,
c.committee_id,
c.name committee
FROM members m
LEFT JOIN committees c ON
c.name = m.name;
```



# LEFT JOIN

```
SELECT
m.member_id,
m.name member,
c.committee_id,
c.name committee
FROM members m
LEFT JOIN committees c USING(name)
WHERE c.committee_id IS NULL;
```

| member_id | member | committee_id | committee |
|---|---|---|---|
| 2 | Jane | NULL | NULL |
| 4 | David | NULL | NULL |

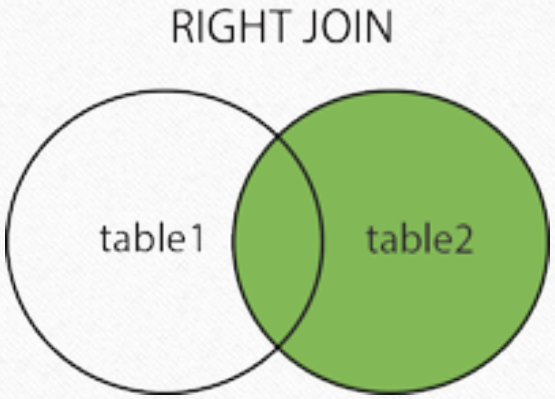


# RIGHT JOIN (or RIGHT OUTER JOIN)

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

**Syntax:**

- SELECT *column_name(s)*
  FROM *table1*
  RIGHT JOIN *table2*
  ON *table1.column_name* = *table2.column_name*;
- It is also called RIGHT OUTER JOIN

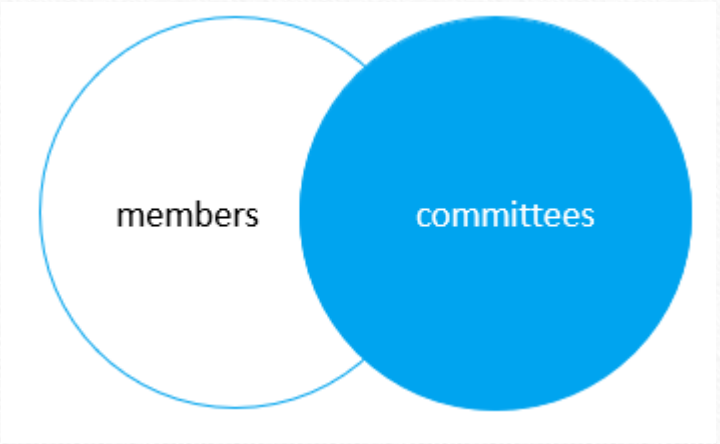


# RIGHT JOIN

```
SELECT
m.member_id,
m.name member,
c.committee_id,
c.name committee
FROM members m
RIGHT JOIN committees c on
c.name = m.name;
```

```
SELECT
m.member_id,
m.name member,
c.committee_id,
c.name committee
FROM members m RIGHT JOIN
committees c USING(name);
```

| member_id | member | committee_id | committee |
|---|---|---|---|
| 1 | John | 1 | John |
| 3 | Mary | 2 | Mary |
| 5 | Amelia | 3 | Amelia |
| NULL | NULL | 4 | Joe |

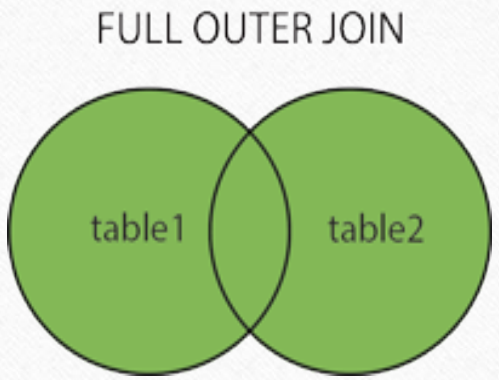


# FULL OUTER JOIN

- It returns all records when there is a match in left (table1) or right (table2) table records.
- **Note:** FULL OUTER JOIN can potentially return very large result-sets!
- **Tip:** FULL OUTER JOIN and FULL JOIN are the same.

**Syntax**

- SELECT *column_name(s)*
  FROM *table1*
  FULL OUTER JOIN *table2*
  ON *table1.column_name* = *table2.column_name*
  WHERE *condition*;



# CROSS JOIN

- The cross join makes a Cartesian product of rows from the joined tables.
- The cross join combines each row from the first table with every row from the right table to make the result set.
- Suppose the first table has **n** rows and the second table has **m** rows.
- The cross join that joins the first with the second table will return **nxm** rows.

**Syntax:**

```
SELECT select_list
FROM table_1
CROSS JOIN
table_2;
```

# CROSS JOIN

```
SELECT
m.member_id,
m.name member,
c.committee_id,
c.name committee
FROM members m
CROSS JOIN committees c;
```

| member_id | member | committee_id | committee |
|---|---|---|---|
| 1 | John | 1 | John |
| 1 | John | 2 | Mary |
| 1 | John | 3 | Amelia |
| 1 | John | 4 | Joe |
| 2 | Jane | 1 | John |
| 2 | Jane | 2 | Mary |
| 2 | Jane | 3 | Amelia |
| 2 | Jane | 4 | Joe |
| 3 | Mary | 1 | John |
| 3 | Mary | 2 | Mary |
| 3 | Mary | 3 | Amelia |
| 3 | Mary | 4 | Joe |
| 4 | David | 1 | John |
| 4 | David | 2 | Mary |
| 4 | David | 3 | Amelia |
| 4 | David | 4 | Joe |
| 5 | Amelia | 1 | John |
| 5 | Amelia | 2 | Mary |
| 5 | Amelia | 3 | Amelia |
| 5 | Amelia | 4 | Joe |