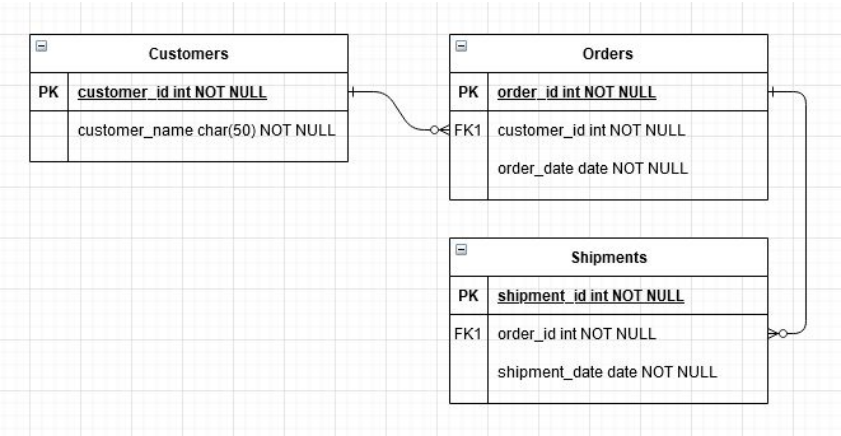
Joins

SQL

Why a Join?

Joins are a way for us to gather related information from multiple tables into one coherent result. This is an especially important feature of relational database systems like MariaDB, as we can leverage relationships between tables to great effect.

Remember the Draw.IO example from earlier?



Note the use of customer_id in the orders table, and order_id in the shipments table. There are relationships here that can provide useful insight when learning about any particular record.

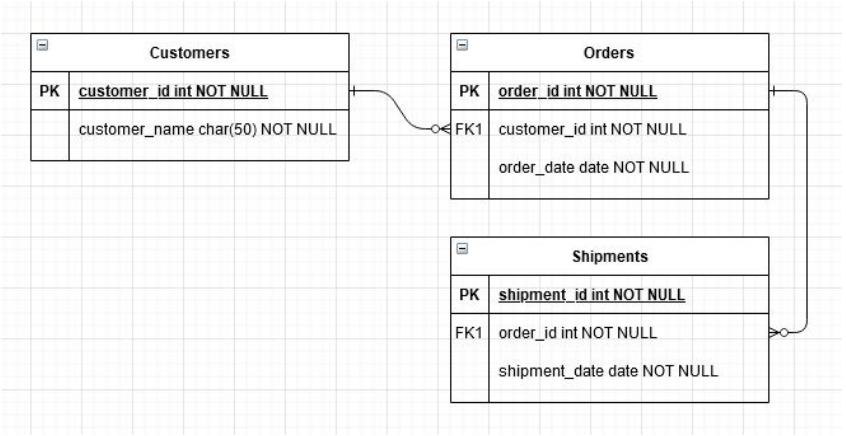
Using joins we can more easily capture groupings of related information than we might otherwise.

Preparing our Database for Joins Practice

In order for us to practice this topic properly, we'll have to add a couple more tables and add some data. Let's go through our steps:

```
-- Create orders table.
CREATE TABLE Orders (
  order_id INT PRIMARY KEY AUTO_INCREMENT,
  customer_id INT NOT NULL,
  order_date DATE NOT NULL
);

-- Create shipments table.
CREATE TABLE Shipments (
  shipment_id INT PRIMARY KEY AUTO_INCREMENT,
  order_id INT NOT NULL,
  shipment_date DATE NOT NULL
);
```



Populate our Database for Joins Practice

Let’s add some orders:

```
-- Add example orders.
INSERT INTO Orders ( `customer_id`, `order_date` )
  VALUES
    (1, ‘2020-11-08’),
    (1, ‘2020-12-15’),
    (3, ‘2021-01-23’),
    (5, ‘2021-03-14’),
    (6, ‘2021-07-03’);
```

Let’s add some shipments:

```
-- Add example shipments.
INSERT INTO Shipments ( `order_id`, `shipment_date` )
  VALUES
    (1, ‘2020-12-05’),
    (2, ‘2021-01-02’),
    (3, ‘2021-02-14’);
```

```
MariaDB [store]> -- Create orders table.
MariaDB [store]> CREATE TABLE Orders (
    ->   order_id INT PRIMARY KEY AUTO_INCREMENT,
    ->   customer_id INT NOT NULL,
    ->   order_date DATE NOT NULL
    -> );
Query OK, 0 rows affected (0.013 sec)

MariaDB [store]> -- Create shipments table.
MariaDB [store]> CREATE TABLE Shipments (
    ->   shipment_id INT PRIMARY KEY AUTO_INCREMENT,
    ->   order_id INT NOT NULL,
    ->   shipment_date DATE NOT NULL
    -> );
Query OK, 0 rows affected (0.008 sec)

MariaDB [store]> -- Add example orders.
MariaDB [store]> INSERT INTO Orders ( `customer_id`, `order_date` )
    ->   VALUES
    ->     (1, ‘2020-11-08’),
    ->     (1, ‘2020-12-15’),
    ->     (3, ‘2021-01-23’),
    ->     (5, ‘2021-03-14’),
    ->     (6, ‘2021-07-03’);
Query OK, 5 rows affected (0.011 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [store]> -- Add example shipments.
MariaDB [store]> INSERT INTO Shipments ( `order_id`, `shipment_date` )
    ->   VALUES
    ->     (1, ‘2020-12-05’),
    ->     (2, ‘2021-01-02’),
    ->     (3, ‘2021-02-14’);
Query OK, 3 rows affected (0.011 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

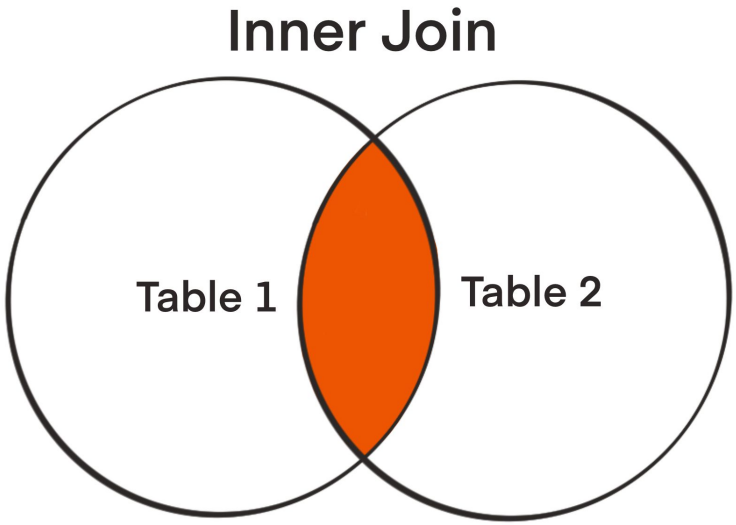
About Joins

- Using SELECT * is generally a bad idea when using a JOIN; too much of a chance of there being ambiguous column names. Name the columns you want to select
- Joining is done on related columns between tables
- Four types of joins, INNER, LEFT, RIGHT, FULL they differ in how ‘incomplete’ matches are handled
    - INNER - both ends match
    - LEFT - all rows in left table are represented
    - RIGHT - all rows in right table are represented
    - FULL - all rows in left and right table are represented ← usually used for auditing purposes

INNER JOIN

Includes only rows whose values match.

- Use INNER JOIN when you need all of the rows in the result set to be complete
- If left and right tables are equally valuable/important
- Only rows that have ‘both ends’ of the match will be selected (i.e. only gives you results that exist in both tables)
- Does NOT allow incomplete rows

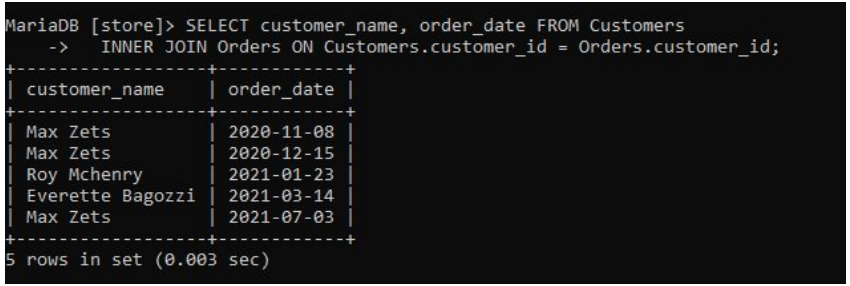


INNER JOIN Practice

Let’s see when orders were placed, and by whom. For this, we will need to join the “Customers” table and the “Orders” table.

```
SELECT customer_name, order_date FROM Customers
  INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

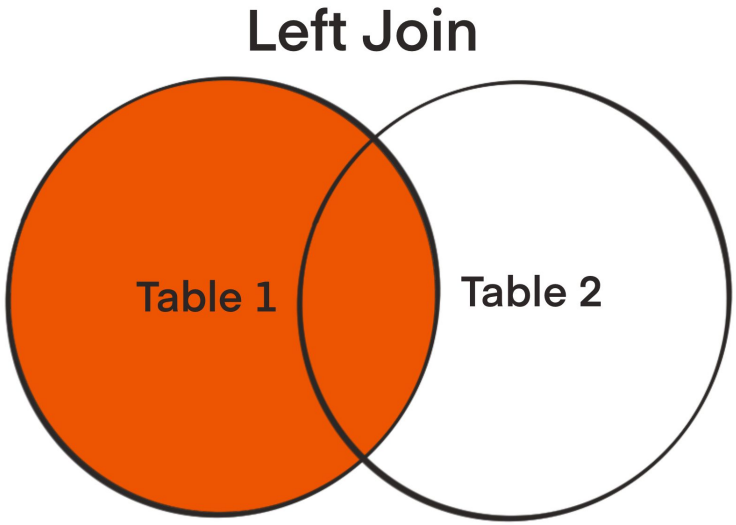
Note that only the intersection—only the matches are displayed in our result. This makes for shorter, concise results with a very particular focus.



LEFT JOIN

All of the rows in the left table are represented.

- Use LEFT JOIN when you have a primary table and use secondary table to resolve i.e. InstructorID → InstructorName
- ALL of the rows in the left table are represented even if they don't match rows in the right table
- Allows for the possibility that one or more rows may be partially empty
- LEFT table is the important one (below, it would be Animal)

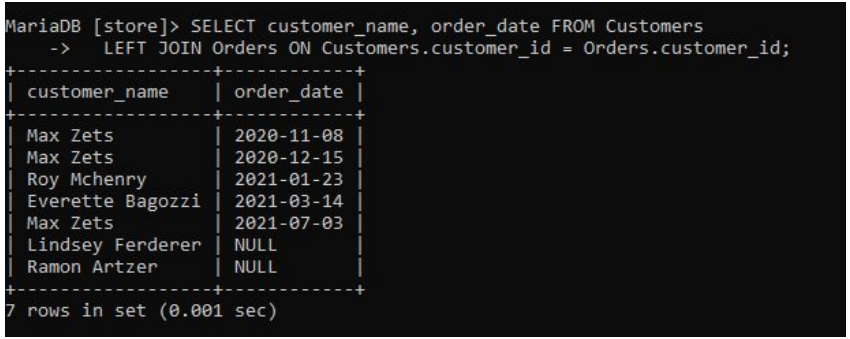


LEFT JOIN Practice

Let's check everyone for orders instead. To see what everyone has ordered, we'll do a LEFT JOIN. This way, we see all customers, and we can determine for ourselves which have, or have not, submitted orders.

```
SELECT customer_name, order_date FROM Customers
  LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

Notice in the result, that even Lindsey and Ramon are shown, despite not having orders (NULL is displayed, representing a lack of available order date.)



```
MariaDB [store]> SELECT customer_name, order_date FROM Customers
    ->   LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
+------------------+------------+
| customer_name    | order_date |
+------------------+------------+
| Max Zets         | 2020-11-08 |
| Max Zets         | 2020-12-15 |
| Roy Mchenry      | 2021-01-23 |
| Everette Bagozzi | 2021-03-14 |
| Max Zets         | 2021-07-03 |
| Lindsey Ferderer | NULL       |
| Ramon Artzer     | NULL       |
+------------------+------------+
7 rows in set (0.001 sec)
```

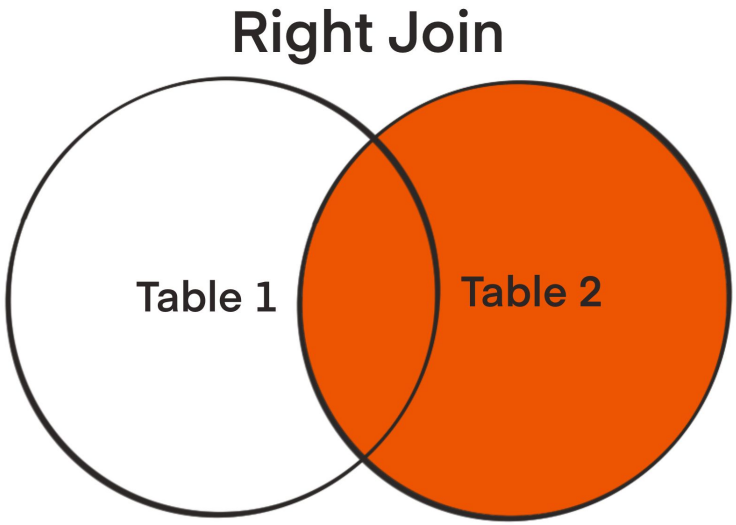
You can modify the selection to include more information that might be useful.

```
SELECT customer_id, customer_name, order_id,  order_date FROM Customers
  LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

RIGHT JOIN

All of the rows in the right table are represented.

- ALL of the rows in the right table are represented even if they don't match rows in the left table
- Allows for the possibility that one or more rows may be partially empty
- RIGHT table is the important one (below, it would be Staff)
- You can reverse the tables in the query, use LEFT JOIN, and get the same thing



RIGHT JOIN Practice

Perhaps we only want to see orders with shipment dates. Let's try checking this using a RIGHT JOIN:

```
SELECT order_id, order_date, shipment_date FROM Orders
  RIGHT JOIN Shipments ON Orders.order_id = Shipments.order_id;
```

```
MariaDB [store]> SELECT order_id, order_date, shipment_date FROM Orders
    ->   RIGHT JOIN Shipments ON Orders.order_id = Shipments.order_id;
ERROR 1052 (23000): Column 'order_id' in field list is ambiguous
```

Oh no! This won't work, because we have a shared column name (order_id) in both tables. We'll have to use an alias… for this we use the AS keyword:

```
SELECT Orders.order_id as order_id, order_date, shipment_date
  FROM Orders
  RIGHT JOIN Shipments ON Orders.order_id = Shipments.order_id;
```

```
MariaDB [store]> SELECT Orders.order_id as order_id, order_date, shipment_date
    ->   FROM Orders
    ->   RIGHT JOIN Shipments ON Orders.order_id = Shipments.order_id;
+----------+------------+---------------+
| order_id | order_date | shipment_date |
+----------+------------+---------------+
|        1 | 2020-11-08 | 2020-12-05    |
|        2 | 2020-12-15 | 2021-01-02    |
|        3 | 2021-01-23 | 2021-02-14    |
+----------+------------+---------------+
3 rows in set (0.007 sec)
```

Recommended Reading

It’s a good idea to read up more on what we’ve covered today. Have a look at the following:

- Beaulieu, A. (March 2020). Learning SQL, 3rd Edition. O’Reilly Media, Inc.
  - Chapter 5. Querying Multiple Tables