RELATIONAL DATABASES

FROM CLAUSE

OBJECTIVES

- Relational databases consist of multiple related tables linked together using common columns known as foreign keys.
- In order to view data from multiple tables we must construct and execute joins between tables.
- Cross join
- Inner join
- Left join
- Right join

ALIASES

- When you join tables you need to deal with columns from different tables.
- What happens if both tables have a column with the same name?
- We use aliases to make differentiating the columns so we know which table they come from.

```
SELECT dept.department_id, employees.fname, employees.lname
FROM departments AS dept
INNER JOIN
     employees
ON dept.department_id = employees.department_id;
```

CROSS JOIN

- A CROSS JOIN makes a Cartesian product of rows from multiple tables.
- If you want to join table1 and table2 using the CROSS JOIN, the result will include the combinations of rows from table1 with the rows in table2.

```
SELECT table1.id, table2.id
FROM table1
CROSS JOIN table2;
```

The output would be the product of the rows in table1 with rows in table2.

INNER JOIN

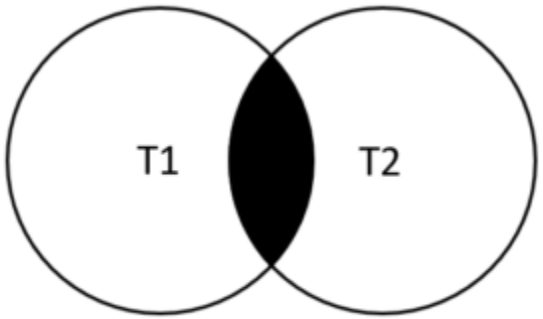
- The inner join allows you to match rows in one table with rows in other tables.
- You must specify the following:
  - The main table that appears in the FROM clause.
  - The table you want to join with the main table, which appears in the INNER JOIN clause.
  - The join condition which appears after the ON keyword of the INNER JOIN clause.

INNER JOIN

```
SELECT column_list
FROM table1
INNER JOIN table2 ON join_condition1
INNER JOIN table3 ON join_condition2
…
WHERE where_conditions;
```

INNER JOIN

- Each row in main table is compares to each row in the join table(s) to see if they satisfy the join_condition.
- The rows in the result set must appear in both tables.
- If the tables have the same column name use a table qualifier or Alias.



MySQL INNER JOIN Venn Diagram

INNER JOIN

```
SELECT t1.productCode,
       t1.productName,
       t2.textDescription
FROM products t1
       INNER JOIN
     productlines t2 ON t1.productLine = t2.productLine;
```

productCode	productName	textDescription
S10_1949	1952 Alpine Renault 1300	Attention car enthusiasts: Make your wildest car ownership dreams come true.
S10_4757	1972 Alfa Romeo GTA	Attention car enthusiasts: Make your wildest car ownership dreams come true.
S10_4962	1962 LanciaA Delta 16V	Attention car enthusiasts: Make your wildest car ownership dreams come true.
S12_1099	1968 Ford Mustang	Attention car enthusiasts: Make your wildest car ownership dreams come true.
S12_1108	2001 Ferrari Enzo	Attention car enthusiasts: Make your wildest car ownership dreams come true.

INNER JOIN

- If the tables being joined have the same name for the join column then you can use this syntax:

```
SELECT t1.productCode,
       t1.productName,
       t2.textDescription
FROM products t1
       INNER JOIN
     productlines t2 USING (productLine);
```

INNER JOIN

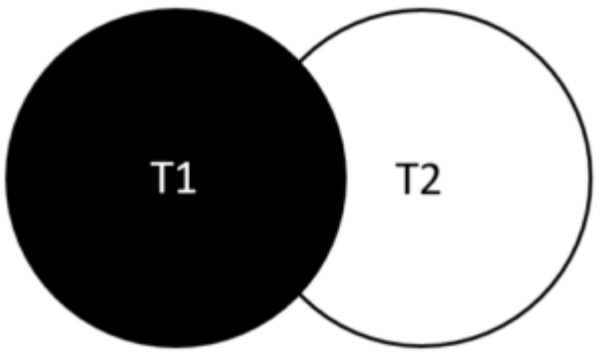
- You can use other operators such as >, <, and <> to join.

```
SELECT
        o.orderNumber,
        p.productName,
        p.productCode,
        p.msrp,
        o.priceEach
FROM products p INNER JOIN orderdetails o
ON p.productCode = o.productCode
WHERE p.productCode = "S10_1678“
AND p.msrp > o.priceEach;
```

LEFT JOIN

- LEFT JOIN clause allows you to query data from two or more database tables.
- This join allows you to select all the rows from the left table that match the join condition with the right table and all those rows that do not match.
- The intersection between two circles are rows that match

The join condition, the remaining part of the T1 circle are those that do not match the join condition but are contained in T1 table.



MySQL LEFT JOIN – Venn Diagram

LEFT JOIN

- Customers and Orders tables:
- Each order in the orders table must belong to a customer in the customers table.
- Each customer in the customers table can have many orders in the orders table.
- To find all orders for each customer we can use the left join. (customers is the left table)

```
SELECT
        c.customerNumber,
        c.customerName,
        o.orderNumber,
        o.status
FROM customers c
LEFT JOIN orders o ON c.customerNumber = o.customerNumber;
```

LEFT JOIN

Result Grid | Filter Rows: | Export: | Wrap Cell Con

customerNumber	customerName	orderNumber	status
124	Mini Gifts Distributors Ltd.	10390	Shipped
124	Mini Gifts Distributors Ltd.	10396	Shipped
124	Mini Gifts Distributors Ltd.	10421	In Process
125	Havel & Zbvszek Co	NULL	NULL
128	Blauer See Auto. Co.	10101	Shipped
128	Blauer See Auto. Co.	10230	Shipped
128	Blauer See Auto. Co.	10300	Shipped
128	Blauer See Auto. Co.	10323	Shipped
129	Mini Wheels Co.	10111	Shipped

The left table is customers, therefore, all customers are included in the result set.
There are customers that have no orders associated as the relationship is optional.

LEFT JOIN

- The left join is very useful to find all those rows in the left table that do not match rows in the right.
- For example if we wanted to only have a list of customers that have no orders associated.

```
SELECT
        c.customerNumber,
        c.customerName,
        o.orderNumber,
        o.status
FROM customers c
LEFT JOIN orders o ON c.customerNumber = o.customerNumber
WHERE orderNumber IS NULL;
```