02 - Retrieving Data From Multiple Tables

Saturday, March 30, 2024 5:25 PM

# Inner Join

As you know we make a lot for tables for several reasons like

- Save more space
- Save our Self for pain of maintain redundant data

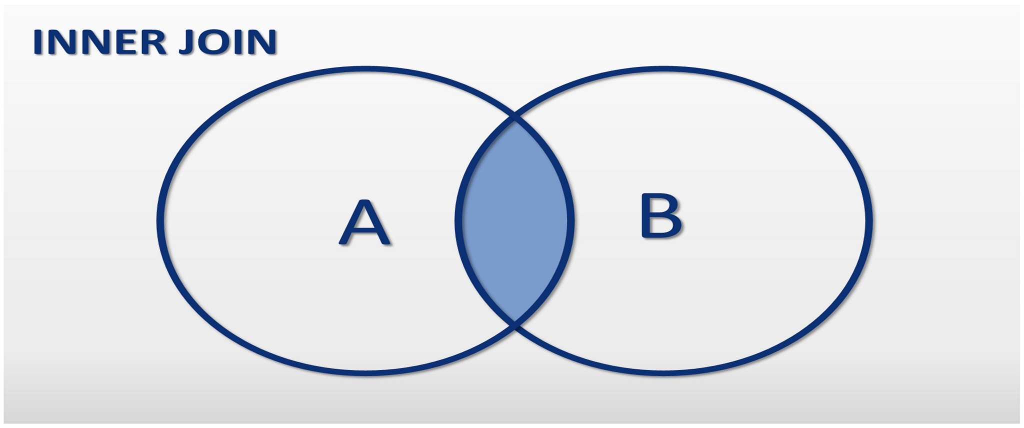
The problem is here now , with data that I have I don't have full picture of information with another Word I don't have meaningful data !

- **JOIN** is here to solve that | BY Default it's **INNER JOIN**

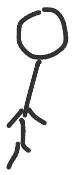
```
SELECT order_id , name , unit_price
FROM orders o
JOIN products p ON o.order_id = p.product_id;
```

Notes :

1 - You Can make Table Allies from making things easier
2 - You Must say table name in select if the column is common
3 - You Join in tables not even from the same database but note that you must tell the database that table info "databaseName.tableName.column"



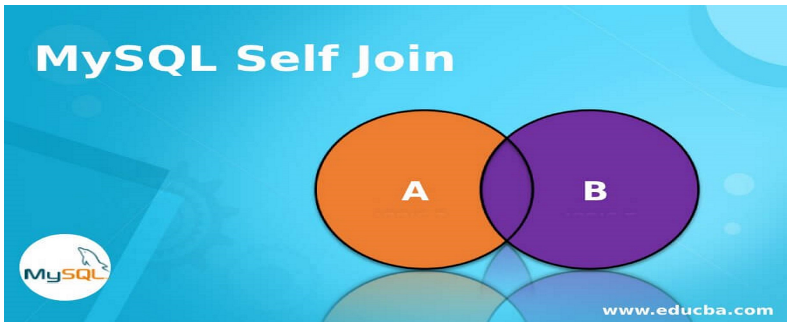
# SELF JOIN



It's Look Not Clear To Me
What Case that I need to do that ?

You Can imagine these scenario
You have table for employee and
There are column where you have
Manager id , But Guess What ?
The Manager is self he is an employee , so do you can imagine how it will
Be pain when you store details about the manager ?
 - what about space
 - what about if we need to modify an employee data and
by chance he is a manager of set of employees ?

```
SELECT e.first_name , m.first_name
FROM employees e
JOIN employees m ON e.reports_to = m.employee_id;
```



# Multiple Tables Joins

You will need in your real daily work to join multiple tables for reposts and so on

```
SELECT order_id , o.order_date , c.first_name , c.last_name , os.name
FROM orders o
JOIN customers c
      ON o.customer_id = c.customer_id
JOIN order_statuses os
      ON o.status = os.order_status_id;
```

```
SELECT p.payment_id , p.date , c.name , pm.name
FROM payments p
JOIN clients c
       ON p.client_id = c.client_id
JOIN payment_methods pm
      ON p.payment_method = pm.payment_method_id ;
```

# Compound Keys and Joining them

sometimes just one column can't make the table row is **UNIQUE** so he must supported with other columns in the table

Like this example

O.id	P.id	order Data	Comments	Shipedate	id
10	6	2018-04-22	2	2018-04-23	2
8	10	2017-07-05	2 Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
8	5	2018-06-08	1 Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.		

O.id it self can't alone make the row unique So He must take help from other columns like P.id
That's what called **compound keys**

- How To Make Join with this table ? **AND**

```
SELECT *
FROM order_items oi
JOIN order_item_notes oin
   ON oi.order_id = oin.order_Id
   AND oi.product_id = oin.product_id;
```

# Implicit Join

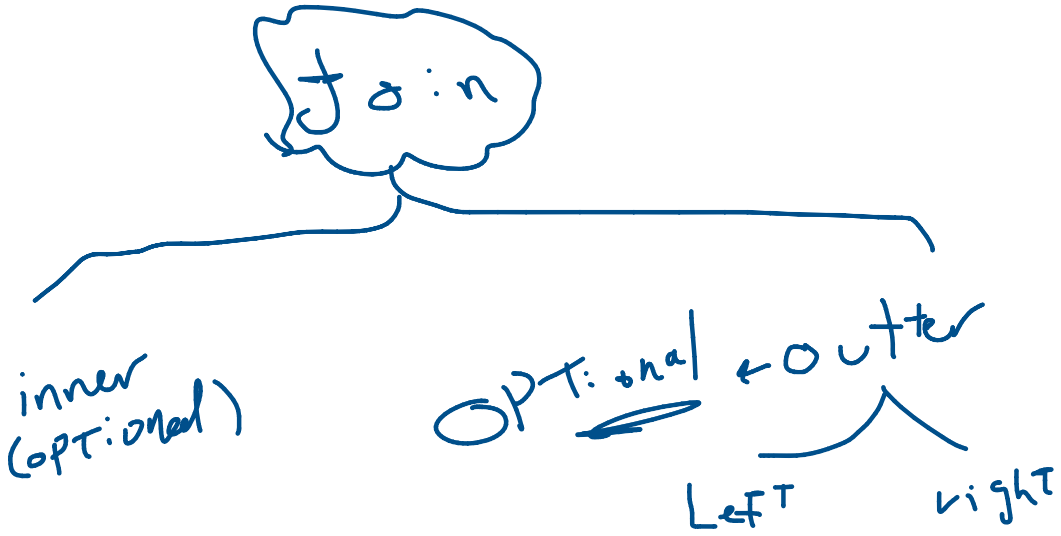
Simply it's about the Join without using JOIN Syntax it's like the most normal syntax

```
SELECT *
FROM orders o
JOIN customers c
ON o.customer_id = c.customer_id;
```

```
SELECT *
FROM orders , customers
WHERE orders.customer_id = customers.customer_id;
```

- The 2 Queries make the same job but not that

NOT Recommended to write the join with that as if you forget the where condition the CROSS JOIN will applied



# OUTTER JOIN

What is the problem that he try to solve ?

The problem with inner join Is that he just get result based on the **JOIN CONDITION**

Like inner JOIN of order and customer

Result will appear just if we have match between customer id in CUSTOMERS and customer id in Orders
But what if we need a report for all customer and if they have order mention that order with details ?

Inner join here can't do something like that so **OUTER JOIN** is the Solution

NOTE :

- **OUTTER JUST USED WHEN YOU NEED TOTAL Data AND IF THERE ARE MATCH BRING HIM**
- **Mixing between using LEFT , RIGHT SO Annoying best practice use LEFT**

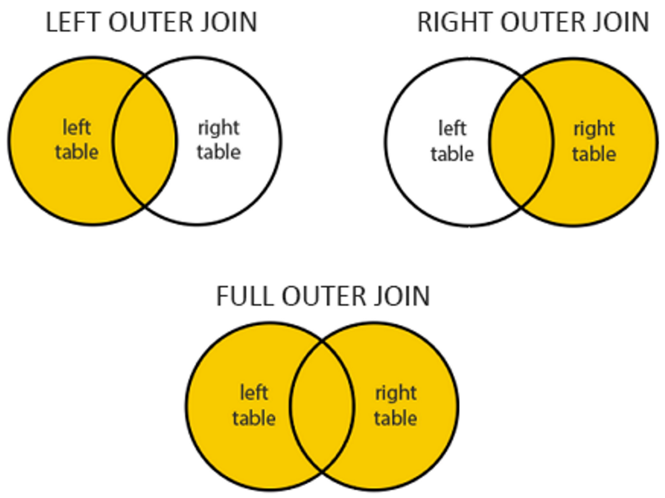
```
SELECT o.order_id , c.customer_id , c.first_name
FROM orders o
RIGHT JOIN customers c ON o.customer_id = c.customer_id;
```

```
SELECT c.customer_id , c.first_name,o.order_id
FROM customers c
LEFT JOIN orders o ON o.customer_id = c.customer_id;
```

You can make multiple OUTTER JOIN like multiple inner to solve same problem

```
SELECT o.order_date ,o.order_id , c.first_name , sh.name
FROM orders o
LEFT JOIN customers c
ON o.customer_id = c.customer_id

LEFT JOIN shippers sh
ON o.shipper_id = sh.shipper_id;
```



# OUTER SELF JOIN

Like the same thing that solved with inner self join but
In the above example the CEO who has no manager don't appear in the query set why ?
Clearly is because of the inner JOIN

```
SELECT e.employee_id , e.first_name , m.first_name AS 'manager'
FROM employees e
LEFT JOIN employees m
ON e.reports_to = m.employee_id;
```

# USING Clause

AS your query become complex and do multiple joins things gonna be not clean and not clear later
So mysql provide good solution for that to make life easier

USING CLASUE

It's used when the matching columns are literally the same like x.id , y.id
Here you can just use the USING **KEYWORD**

```
SELECT *
FROM orders o
JOIN customers c
USING (customer_id)


SELECT p.date , c.name , p.amount , pm.name
```

```
FROM payments p
JOIN clients c
USING (client_id)

JOIN payment_methods pm
ON p.payment_method = pm.payment_method_id;
```

# Natural JOIN

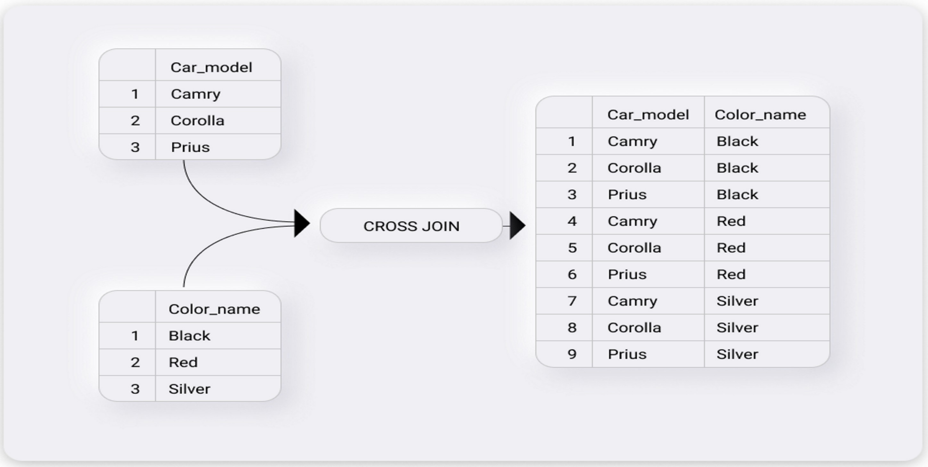
Type of join but not recommended because of that

The database make join based on the common columns between the two tables

```
SELECT o.order_id , c.first_name , c.last_name
FROM orders o
NATURAL JOIN customers c;
```

# Cross Join

Literally generate all combination between the two tables



	Car_model
1	Camry
2	Corolla
3	Prius

	Color_name
1	Black
2	Red
3	Silver

	Car_model	Color_name
1	Camry	Black
2	Corolla	Black
3	Prius	Black
4	Camry	Red
5	Corolla	Red
6	Prius	Red
7	Camry	Silver
8	Corolla	Silver
9	Prius	Silver

```
SELECT *
FROM orders
CROSS JOIN  customers ;
```

The above form the called explicit form you can simplify it to implicit form like that

```
SELECT *
FROM orders , customers
```

# Union

Suppose this case when you asked to make report about classify all orders that done this year and make it active and make other that Done in the previous years as achieve

```
SELECT order_id , order_date , 'Active' AS 'Status'
FROM orders
WHERE order_date >= '2019-01-01';

SELECT order_id , order_date , 'Archive' AS 'Status'
FROM orders
WHERE order_date < '2019-01-01';
```

The problem is that the second query overlap the first one result and the second the only one who appear

UNION

Union here to solve that it's about collecting the results

```
SELECT order_id , order_date , 'Active' AS 'Status'
FROM orders
WHERE order_date >= '2019-01-01'
UNION
SELECT order_id , order_date , 'Archive' AS 'Status'
FROM orders
WHERE order_date < '2019-01-01';
```

union
جب لم كل ال Queries دي ل
البوكس بينى