Union, Intersect & Sub-query

Union

- The Union is a binary set operator in DBMS. It is used to combine the result set of two select queries. Thus, it combines two result sets into one.
- In other words, the result set obtained after union operation is the collection of the result set of both the tables.
- But two necessary conditions need to be fulfilled when we use the union command.
  - ✓ Both SELECT statements should have an equal number of fields in the same order.
  - ✓ The data types of these fields should either be the same or compatible with each other.

Syntax:

```
SELECT (column_names) from table1 [WHERE condition]

UNION

SELECT (column_names) from table2 [WHERE condition];
```

Union

- The Union is a binary set operator in DBMS. It is used to combine the result set of two select queries. Thus, it combines two result sets into one.
- In other words, the result set obtained after union operation is the collection of the result set of both the tables.
- But two necessary conditions need to be fulfilled when we use the union command.
  - ✓ Both SELECT statements should have an equal number of fields in the same order.
  - ✓ The data types of these fields should either be the same or compatible with each other.

Syntax:

```
SELECT (column_names) FROM table1 [WHERE condition]

UNION

SELECT (column_names) FROM table2 [WHERE condition];
```

Example: *Get all the employees and all departments*

```
SELECT Emp.ID, Name, Dept_name FROM Emp LEFT JOIN Dept USING(ID)

UNION

SELECT Emp.ID, Name, Dept_name FROM Emp RIGHT JOIN Dept USING(ID)
```

Union All

- This operator returns all rows by combining two or more results from multiple SELECT queries into a single result set. It does not remove the duplicate rows from the result set.

- *The difference between Union and Union All operators is that Union returns all distinct rows (eliminate duplicate rows) from two or more tables into a single output. In contrast, Union All returns all the rows, including duplicates rows.*

Syntax:

```
SELECT (column_names) FROM table1 [WHERE condition]

UNION ALL

SELECT (column_names) FROM table2 [WHERE condition];
```

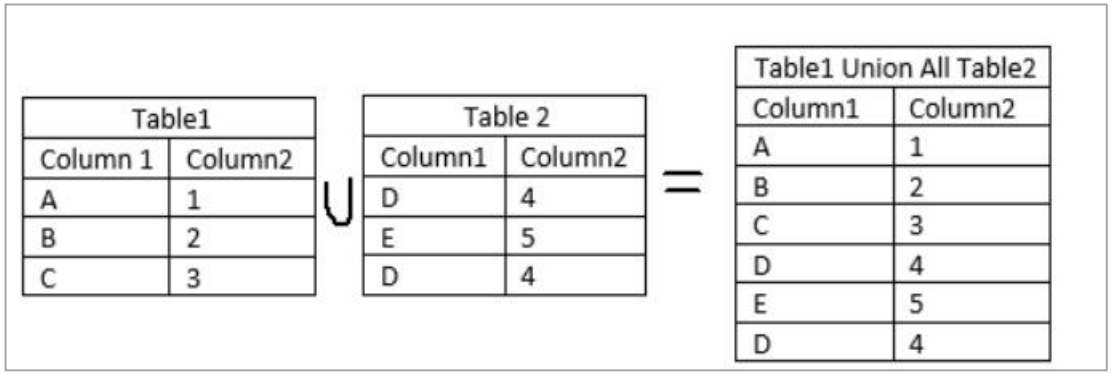


Table1	
Column 1	Column2
A	1
B	2
C	3

∪

Table 2	
Column1	Column2
D	4
E	5
D	4

=

Table1 Union All Table2	
Column1	Column2
A	1
B	2
C	3
D	4
E	5
D	4

Intersect

- The INTERSECT operator returns the distinct (common) elements in two sets or common records from two or more tables.
- In other words, it compares the result obtained by two queries and produces unique rows, which are the result returned by both queries.
- Although there is no INTERSECT operator in MySQL, you can easily simulate this type of query using either the IN clause or the EXISTS clause, depending on the complexity of the INTERSECT query.

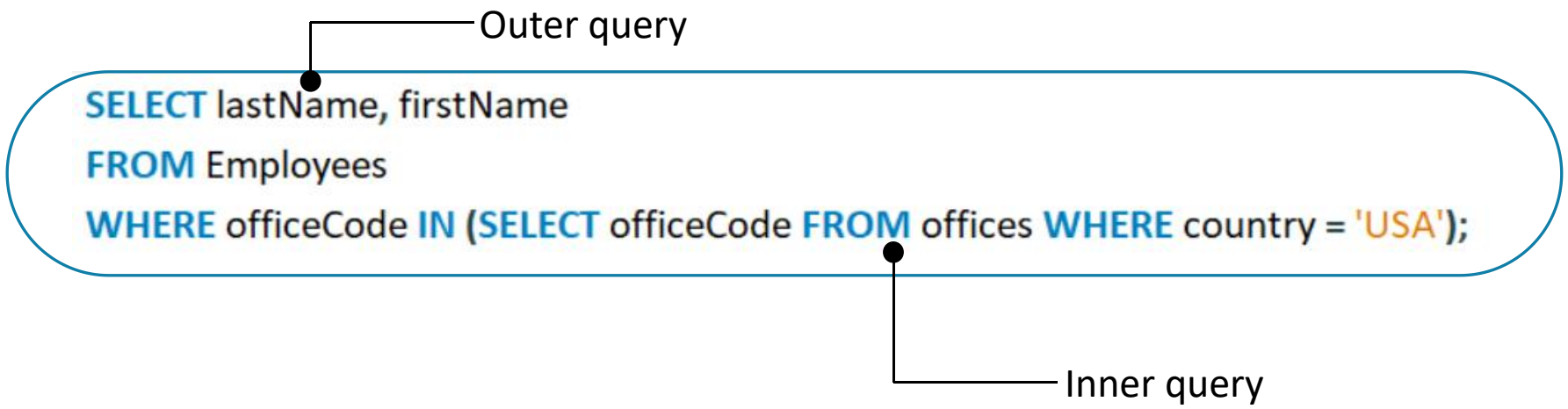
Syntax: *If the database supported the INTERSECT operator (which MySQL does not), this is how we would have used the INTERSECT operator to return the common category_id values between the products and inventory tables.*

```
SELECT products.category_id
FROM products
WHERE products.category_id IN (SELECT inventory.category_id FROM inventory);
```

Subquery

- A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. Also, a subquery can be nested within another subquery.
- A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query.
- A subquery can be used anywhere that expression is used and must be closed in parentheses.

For example, the following query uses a subquery to return the employees who work in the offices located in the USA.



Subquery with WHERE clause

- We can use comparison operators e.g., =, >, < to compare a single value returned by the subquery with the expression in the WHERE clause.

- For example, the following query returns the customer who has the highest payment.

```
SELECT customerNumber, checkNumber, amount
FROM payments
WHERE amount = (SELECT MAX(amount) FROM payments);
```

Rules

- Always use Parentheses.

- If the main query does not have multiple columns for subquery, then a subquery can have only one column in the SELECT command.

- Can use various comparison operators with the subquery, such as >, <, =, IN, ANY, SOME, and ALL

Correlated Subquery

- A correlated subquery is a subquery that uses the data from the outer query.
- In other words, a correlated subquery depends on the outer query. A correlated subquery is evaluated once for each row in the outer query.
- The following example uses a correlated subquery to select products whose buy prices are greater than the average buy price of all products in each product line.

```
SELECT
   productname,
   buyprice
FROM products p1
WHERE buyprice > (SELECT AVG(buyprice) FROM products
                  WHERE productline = p1.productline);
```

Subquery with EXISTS and NOT EXISTS

- When a subquery is used with the EXISTS or NOT EXISTS operator, a subquery returns a Boolean value of TRUE or FALSE. The following query illustrates a subquery used with the EXISTS operator.

```
SELECT * FROM table name WHERE
EXISTS(subquery);
```

```
SELECT customerNumber,  customerName FROM customers
WHERE EXISTS( SELECT
                orderNumber, SUM(priceEach * quantityOrdered)
                FROM orderdetails INNER JOIN orders USING (orderNumber)
                WHERE customerNumber = customers.customerNumber
                GROUP BY orderNumber
                HAVING SUM(priceEach * quantityOrdered) > 60000
              );
```