RKIT Software Private Limited

Gupta Heer Dipak
Full Stack Developer

Pre-Joining Topics

Week 2: MYSQL ADVANCED

UNIONS

UNION Operator

The **UNION** operator in MySQL is used to combine the results of two or more SELECT statements into a single result set. Each SELECT statement within the UNION must have the same number of columns in the result sets with similar data types. The columns in the result set are named after the columns in the first SELECT statement.

Syntax:

The syntax for using **UNION** Operator in **MySQL** is as follows:

SELECT column1, column2, ...

FROM table1

WHERE condition

UNION

SELECT column1, column2, ...

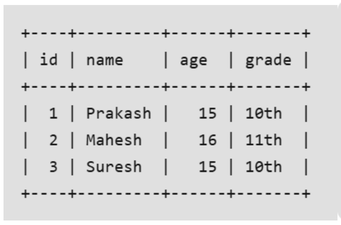
FROM table2

WHERE condition;

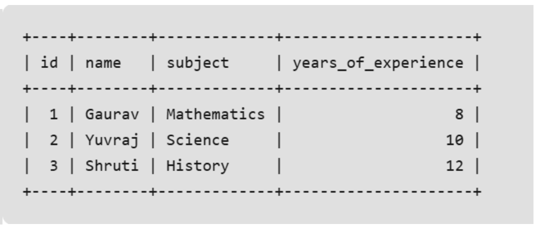
Parameters

- **column1, column2, ...:** Columns selected from each SELECT statement must be the same in number and compatible in data type.
- **table1, table2, ...:** The tables or views from which data is selected.
- **condition:** Optional conditions to filter rows in each SELECT statement.
- The UNION operator combines the results of the two SELECT statements and removes duplicates by default.

Examples:



```
+----+---------+------+-------+
| id | name    | age  | grade |
+----+---------+------+-------+
|  1 | Prakash |   15 | 10th  |
|  2 | Mahesh  |   16 | 11th  |
|  3 | Suresh  |   15 | 10th  |
+----+---------+------+-------+
```

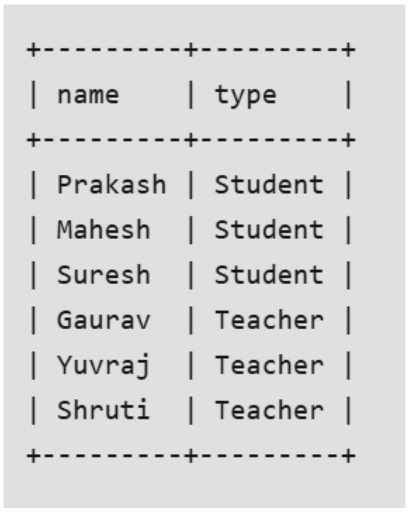


```
+----+--------+-------------+---------------------+
| id | name   | subject     | years_of_experience |
+----+--------+-------------+---------------------+
|  1 | Gaurav | Mathematics |                   8 |
|  2 | Yuvraj | Science     |                  10 |
|  3 | Shruti | History     |                  12 |
+----+--------+-------------+---------------------+
```

Example 1: Combing Names from Students and Teachers Table

In this example, we are using the UNION operator to combine the name column from the **students** table labeled as '**Student**' and the name column from the **teachers** table labeled as '**Teacher**'.

SELECT name, 'Student' AS type
FROM students
UNION
SELECT name, 'Teacher' AS type
FROM teachers;



```
+---------+---------+
| name    | type    |
+---------+---------+
| Prakash | Student |
| Mahesh  | Student |
| Suresh  | Student |
| Gaurav  | Teacher |
| Yuvraj  | Teacher |
| Shruti  | Teacher |
+---------+---------+
```

Example 2: Combining Names with Conditions

In this example, we are using the UNION operator to combine names from the **students** and **teachers** tables based on specific conditions: selecting students with age greater than 15 labeled as '**Student**', and selecting **teachers** with years of experience greater than 8 labeled as '**Teacher**'.

SELECT name, 'Student' AS type
FROM students
WHERE age > 15
UNION
SELECT name, 'Teacher' AS type

FROM teachers
WHERE years_of_experience > 8;

Output:

```
+--------+---------+
| name   | type    |
+--------+---------+
| Mahesh | Student |
| Yuvraj | Teacher |
| Shruti | Teacher |
+--------+---------+
```

UNION ALL Operator

The **UNION ALL operator** in **MySQL** combines the result sets of multiple **SELECT** statements by retaining all duplicate rows for improved performance and efficiency. It is particularly useful when complete data inclusion, including duplicates is required.

In this article, We will learn about the **MySQL UNION ALL Operator** by understanding various examples and so on.

What is UNION ALL?

- The **UNION ALL** operator in MySQL is used to combine the result sets of two or more SELECT statements.
- Unlike the UNION operator, UNION ALL does not remove **duplicate rows** from the result sets.
- This makes UNION ALL faster and more efficient than UNION when we do not need to eliminate duplicates.

Syntax

The basic syntax for the UNION ALL operator is as follows:

SELECT column1, column2, ...

FROM table1

UNION ALL

SELECT column1, column2, ...

FROM table2;

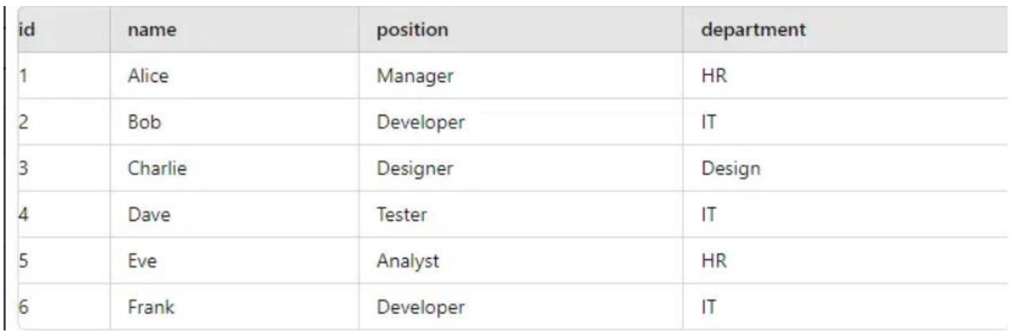
Here, each **SELECT** statement fetches data from different tables or the same table based on certain conditions, and **UNION ALL** combines these results.

How Does UNION ALL Operator Work in MySQL?

- The **UNION ALL** statement combines the records returned by two or more **SELECT** statements and forms a larger dataset by appending the set of values.
- It joins all the result set of rows from each and every **SELECT** query.
- This is done without applying any specific type of condition on the rows.
- To filter through the rows and therefore includes all the data which includes **repetition** of data.

Examples of MySQL UNION ALL Operator

To understand **MySQL UNION ALL Operator** we need a table on which we will perform various operations and queries. Here we will consider a table called **employees** as shown below:



id	name	position	department
1	Alice	Manager	HR
2	Bob	Developer	IT
3	Charlie	Designer	Design
4	Dave	Tester	IT
5	Eve	Analyst	HR
6	Frank	Developer	IT

Example 1: Return Single Field using UNION ALL Operator

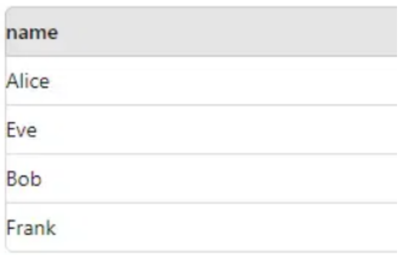
Let's Retrieve a combined list of employee names who either work in the HR department or hold the position of Developer, including duplicates.

```
SELECT name FROM employees WHERE department = 'HR'

UNION ALL

SELECT name FROM employees WHERE position = 'Developer';
```

Output:



name
Alice
Eve
Bob
Frank

Example 2: UNION ALL Operator with ORDER BY Clause & WHERE Option

Let's Suppose we need to retrieve a combined list of names and positions of employees who either work in the HR department or hold the position of Developer and then sort this list by position.

UNION ALL Query with ORDER BY

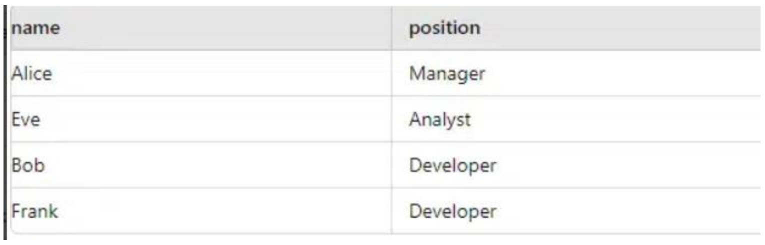
SELECT name, position FROM employees WHERE department = 'HR'

UNION ALL

SELECT name, position FROM employees WHERE position = 'Developer'

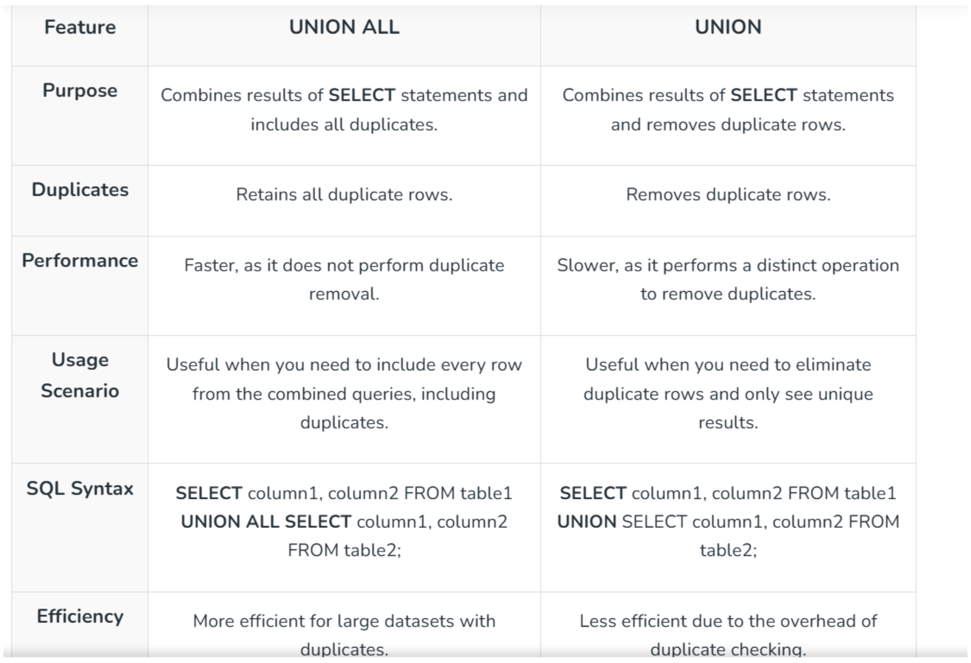
ORDER BY position;

Output:



name	position
Alice	Manager
Eve	Analyst
Bob	Developer
Frank	Developer

UNION ALL Operator vs UNION Operator



Feature	UNION ALL	UNION
Purpose	Combines results of **SELECT** statements and includes all duplicates.	Combines results of **SELECT** statements and removes duplicate rows.
Duplicates	Retains all duplicate rows.	Removes duplicate rows.
Performance	Faster, as it does not perform duplicate removal.	Slower, as it performs a distinct operation to remove duplicates.
Usage Scenario	Useful when you need to include every row from the combined queries, including duplicates.	Useful when you need to eliminate duplicate rows and only see unique results.
SQL Syntax	**SELECT** column1, column2 FROM table1 **UNION ALL SELECT** column1, column2 FROM table2;	**SELECT** column1, column2 FROM table1 **UNION** SELECT column1, column2 FROM table2;
Efficiency	More efficient for large datasets with duplicates.	Less efficient due to the overhead of duplicate checking.