CREATE TABLE EmployeeUK1

(

EmployeeId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUK1 VALUES(1, 'Harish', 'Mishra', 'Male','IT');

INSERT INTO EmployeeUK1 VALUES(2, 'Ashrulochan', 'Sahoo', 'Male','IT');

INSERT INTO EmployeeUK1 VALUES(3, 'Sai', 'Sravanthi', 'Female','HR');

INSERT INTO EmployeeUK1 VALUES(4, 'Wasim', 'Akram', 'Male','HR');

INSERT INTO EmployeeUK1 VALUES(5, 'Shaheed', 'Mushreef', 'Male','IT');

INSERT INTO EmployeeUK1 VALUES(6, 'Bharathi', 'Shanmugam', 'Male','HR');

INSERT INTO EmployeeUK1 VALUES(7, 'Hina', 'Sharma', 'Female','IT');

CREATE TABLE EmployeeUSA1

(

EmployeeId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUSA1 VALUES(1, 'James', 'Pattrick', 'Male','IT');

INSERT INTO EmployeeUSA1 VALUES(2, 'Ashrulochan', 'Sahoo', 'Male','IT');

INSERT INTO EmployeeUSA1 VALUES(3, 'Sara', 'Taylor', 'Female','HR');

INSERT INTO EmployeeUSA1 VALUES(4, 'Wasim', 'Akram', 'Male','HR');

INSERT INTO EmployeeUSA1 VALUES(5, 'Sushanta', 'Jena', 'Male','HR');

INSERT INTO EmployeeUSA1 VALUES(6, 'Mahesh', 'Sindhey', 'Female','HR');

INSERT INTO EmployeeUSA1 VALUES(7, 'Hina', 'Sharma', 'Female','IT');

UNION Operator in MySQL

• The UNION operator is used to combine the result set of two or more SELECT statements into a single result set by removing the duplicate records.

• That means the UNION Operator selects only the distinct values. Following is the Syntax to use UNION Operator in MySQL.

The following query combines two select statements using the UNION operator. In our example, both the EmployeeUK and EmployeeUSA tables having seven records.

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK1

UNION

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA1;

• Once you execute the above query, you will get the following result set. Please observe, here we don’t have any duplicate data.

• Here, in the result set, we got a total of 11 rows out of 14 rows. This is because 3 rows are present in both the result set.

UNION ALL Operator in MySQL

The UNION ALL operator is used to combine the result set of two or more SELECT statements into a single result including the duplicate values. Following is the Syntax to use UNION ALL Operator in MySQL.

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK1

UNION ALL

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA1;

• From the output, it is very clear to us that UNION Operator removes duplicate rows whereas UNION ALL operator does not remove the duplicate rows.

• When we use a UNION operator, in order to remove the duplicate rows from the result set, it must do a distinct operation which is time-consuming.

• For this reason, UNION ALL is much faster than UNION Operator in MySQL.

UNION/UNION ALL with ORDER BY Clause in MySQL

• The UNION/UNION ALL Operator can be used with the ORDER BY clause to sort the result returned from the query.

• Suppose we want to sort the employees by First Name column values. ORDER BY clause should be part of the last select statement.

• The SQL statement will be:

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK1

UNION

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA1

ORDER BY FirstName;

MySQL EXCEPT Operator:

• The EXCEPT operator is used to combine two tables or two result sets and will return rows from the first select statement that are not present in the second select statement.

• Following is the syntax of EXCEPT Operator.

But, the EXCEPT Operator is not supported by MYSQL. We can achieve the EXCEPT Operator functionality in MySQL using the following ways.

Using NOT IN Operator to achieve EXCEPT functionality:

Here, we are checking the FirstName column value only.

Following is the SQL Query using the NOT IN Operator which returns the employees from the first EmployeeUK table that are not present in the EmployeeUSA table.

SELECT \* FROM EmployeeUK1

WHERE FirstName NOT IN (SELECT FirstName FROM EmployeeUSA1);

Using Join to achieve EXCEPT functionality in MySQL:

• We can use LEFT JOIN to achieve the functionality of EXCEPT Operator. Here, the join clause needs to contain all 4 columns FirstName, LastName, Gender, and Department.

• Where clause picks null values in EmployeeId in EmployeeUSA, which limits to rows that exist in EmployeeUK only.

SELECT t1.\* FROM EmployeeUK1 AS t1

LEFT JOIN EmployeeUSA1 AS t2 ON

t1.FirstName=t2.FirstName

AND t1.LastName=t2.LastName

AND t1.Gender=t2.Gender

AND t1.Department=t2.Department

WHERE t2.EmployeeId IS NULL;

INTERSECT Operator in MySQL

• The INTERSECT operator is used to combine two result sets and returns the data which are common in both the result set.

• Following is the syntax of INTERSECT operator.

But the INTERSECT Operator is not supported by MYSQL.

We can achieve the INTERSECT Operator functionality in MySQL using the following ways.

Using IN Operator to achieve INTERSECT functionality:

• Here, we are checking the FirstName column value only.

• Following is the SQL Query using the IN Operator which returns the common employees i.e. the employees which are present in both t EmployeeUK and EmployeeUSA tables.

• Here, we are checking common based on the First Name column value.

SELECT \* FROM EmployeeUK1

WHERE FirstName IN (SELECT FirstName FROM EmployeeUSA1);

Using Join to achieve INTERSECT functionality in MySQL:

We can use INNER JOIN to achieve the functionality of INTERSECT Operator. Here, the join clause needs to contain all 4 columns FirstName, LastName, Gender, and Department.

SELECT t1.\* FROM EmployeeUK AS t1

INNER JOIN EmployeeUSA AS t2 ON

t1.FirstName=t2.FirstName

AND t1.LastName=t2.LastName

AND t1.Gender=t2.Gender

AND t1.Department=t2.Department;

Transaction Management in DBMS: What are ACID Properties?

• Database Transaction is a logical unit of processing in a DBMS which entails one or more database access operation.

• In a nutshell, database transactions represent real-world events of any enterprise.

• All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction in DBMS.

• During the transaction the database is inconsistent.

• Only once the database is committed the state is changed from one consistent state to another.