LAB: 5

Introduction to Joins & Creating Views

SQL Joins

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

SQL Join Types:

There are different types of joins available in SQL:

- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: returns rows when there is a match in one of the tables.
- SELF JOIN: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN: returns the Cartesian product of the sets of records from the two or more joined tables.

INNER JOIN

The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN.The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

Syntax:

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name=table2.column_name*;

OR

SELECT *column_name(s)*
FROM *table1*
JOIN *table2*
ON *table1.column_name=table2.column_name*;

LEFT JOIN

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

Syntax:

SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name=table2.column_name*;

or:

SELECT *column_name(s)*
FROM *table1*
LEFT OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;

RIGHT JOIN

The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result , but with NULL in each column from left table.This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

Syntax:

SELECT *column_name(s)*
FROM *table1*
RIGHT JOIN *table2*
ON *table1.column_name=table2.column_name*;

or:

SELECT *column_name(s)*
FROM *table1*
RIGHT OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;

FULL JOIN

The SQL FULL JOIN combines the results of both left and right outer joins.

The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

Syntax:

SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;

SELF JOIN

The SQL SELF JOIN is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

Syntax:

SELECT a.column_name, b.column_name...

FROM table1 a, table1 b

WHERE a.common_filed = b.common_field;

SQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

Syntax

SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*;

Note: The UNION operator selects only distinct values by default. To allow duplicate values, use the ALL keyword with UNION.

UNION ALL Syntax

SELECT *column_name(s)* FROM *table1*
UNION ALL
SELECT *column_name(s)* FROM *table2*;

PS: The column names in the result-set of a UNION are usually equal to the column names in the first SELECT statement in the UNION.

There are two other clauses (i.e., operators), which are very similar to UNION clause:

 SQL INTERSECT Clause: is used to combine two SELECT statements, but returns rows only from the first

SELECT statement that are identical to a row in the second SELECT statement.

 SQL EXCEPT Clause : combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.

INTERSECT Clause

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

Just as with the UNION operator, the same rules apply when using the INTERSECT operator. MySQL does not

support INTERSECT operator

Syntax:

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

INTERSECT

SELECT column1 [, column2]

FROM table1 [, table2]

[WHERE condition]

Here given condition could be any given expression based on your requirement.

EXCEPT Clause

The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement. This means EXCEPT returns only rows, which are not available in second SELECT statement.

Just as with the UNION operator, the same rules apply when using the EXCEPT operator. MySQL does not support EXCEPT operator.

Syntax:

SELECT column1 [, column2]
FROM table1 [, table2]
[WHERE condition]
EXCEPT
SELECT column1 [, column2]
FROM table1 [, table2]s
[WHERE condition]
Here given condition could be any given expression based on your requirement.

SQL - Using Views

Aview is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views:

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables, or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in very similar way as you use them in normal SQL SELECT query.

The WITH CHECK OPTION:

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

Example:

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.
Updating a View:

A view can be updated under certain conditions:

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So if a view satisfies all the above mentioned rules then you can update a view.

Following is an example to update the age of Ramesh:

```
SQL > UPDATE CUSTOMERS_VIEW
SET AGE = 35
WHERE name='Ramesh';
```

This would ultimately update the base table CUSTOMERS and same would reflect in the view itself. Now, try to query base table, and SELECT statement would produce the following result:

Inserting Rows into a View:

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we can not insert rows in CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in similar way as you insert them in a table.

Deleting Rows into a View:

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE= 22.

SQL > DELETE FROM CUSTOMERS_VIEW

WHERE age = 22;

This would ultimately delete a row from the base table CUSTOMERS and same would reflect in the view itself.

Dropping Views:

Syntax:

DROP VIEW view_name;

Following is an example to drop CUSTOMERS_VIEW from CUSTOMERS table:

DROP VIEW CUSTOMERS_VIEW;

Lab Task:

1. Perform join operation on teacher and employee table and display the Ename , Faculty and salary.
2. Perform left join on table book list and book table.
3. Perform right join on booklist and book table.
4. Perform Full join on student and issues table.
5. Display those employees name and salary whose name starts with ‘s’ and whose name consist ‘ni’ as sub string.
6. Display name of the employee who is also a teacher.
7. Display all employees name except the name who are teachers.
8. Create a view Employee-view which consist of eid, ename , salary as attributes.
9. Insert an new record in recently created view. And also display the contents of primary table.
10. Delete the information from view where salary are less than 5000.