MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-03**

PART: A) ASSIGNMENT NO: 03

AIM: **SQL Queries – all types of Join, Sub-Query and View:**

Write at least10 SQL queries for suitable database application using SQL DML statements.

Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join, Sub-Query and View.

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To develop basic Database administration skill.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative.
- Front End: Java/PHP/Python.
- Back End: MySQL/ Oracle Database.

THEORY:

(A). MySQL: Joins

MySQL **JOINS** are used to retrieve data from multiple tables. A MySQL JOIN is performed whenever two or more tables are joined in a SQL statement.

The purpose of a join concept is to combine data spread across tables. A join is actually performed by the “where‟ clause which combines specified rows of tables.

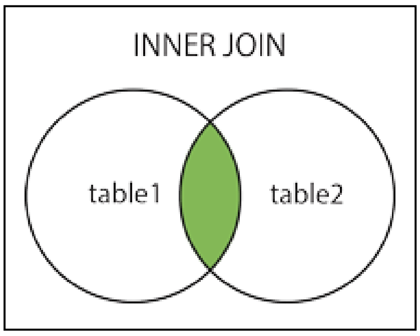
There are different types of MySQL joins:

- ✓ MySQL INNER JOIN (or sometimes called simple join)
- ✓ MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- ✓ MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)
- ✓ MYSQL FULL JOIN (Combine Left & Right Join)

1. MySQL Inner Join

- The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.
- **Note:** The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns.

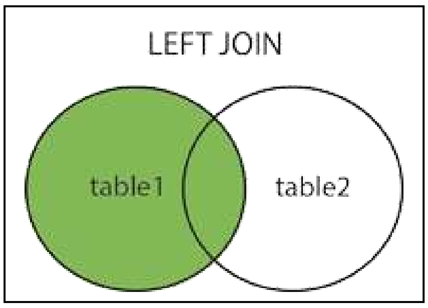
  #SELECT *column_name(s)*
  FROM *table1*
  INNER JOIN *table2*
  ON *table1.column_name=table2.column_name*;
  OR
  SELECT *column_list*
  FROM table1
  INNER JOIN *table2 ON join_condition1*
  INNER JOIN *table3 ON join_condition2*
  ...
  WHERE *where_conditions;*



2. MySQL Left Join

- The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2).
- The result is NULL in the right side when there is no match.
- **Note:** The LEFT JOIN keyword returns all the rows from the left table , even if there are no matches in the right table.

  #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*;

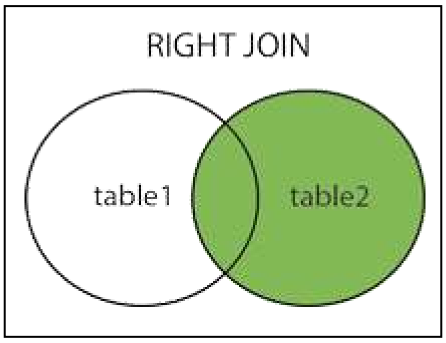


3. MySQL Right Join

- The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1).
- The result is NULL in the left side when there is no match.

- **Note:** The RIGHT JOIN keyword returns all the rows from the right table, even if there are no matches in the left table.

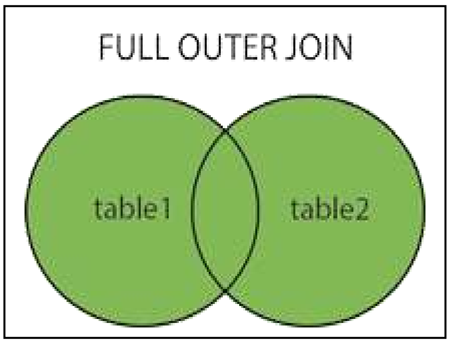
  #SELECT *column_name(s)*
  FROM *table1*
  RIGHT JOIN *table2*
  ON *table1.column_name=table2.column_name*;
  OR
  #SELECT *column_name(s)*
  FROM *table1*
  RIGHT OUTER JOIN *table2*
  ON *table1.column_name=table2.column_name*;



4. MySQL Full Join

- The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).
- The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.
- UINON Keyword can be used to combine result.
- **Note:** The FULL OUTER JOIN keyword returns all the rows from the left table (Table1), and all the rows from the right table (Table2). If there are rows in “Table1" that do not have matches in " Table2", or if there are rows in " Table2" that do not have matches in " Table1", those rows will be listed as well.

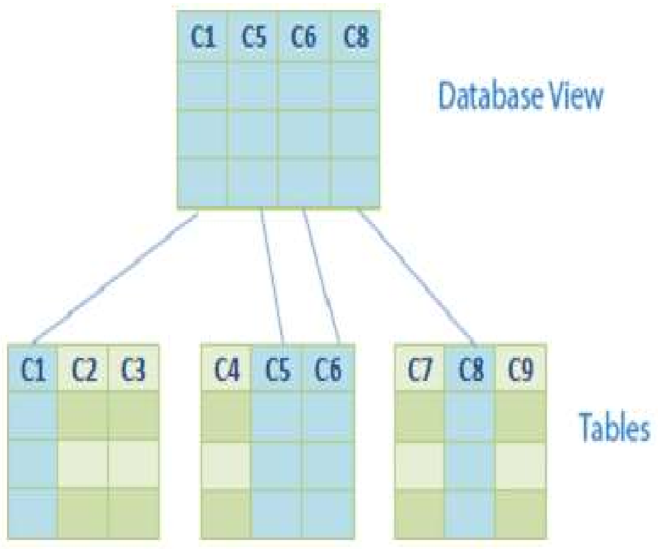
  #SELECT *column_name(s)*
  FROM *table1*
  RIGHT JOIN *table2*
  ON *table1.column_name=table2.column_name*
  UNION
  #SELECT *column_name(s)*
  FROM *table1*
  LEFT JOIN *table2*
  ON *table1.column_name=table2.column_name*;



(B). MySQL-Views

- A database view is a virtual table or logical table which is defined as a SQL SELECT query with joins. Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it. Most database management systems, including MySQL, allow you to update data in the underlying tables through the database view with some prerequisites.

- A database view is dynamic because it is not related to the physical schema. The database system stores database views as a SQL SELECT statement with joins. When the data of the tables changes, the view reflects that changes as well.
- The difference between a view and a table is that views are definitions built on top of other tables (or views).
- A view can be built on top of a single or multiple tables.
- View is a data object which does not contain any data. Contents of the view are the resultant of a base table. They are operated just like base table but they don't contain any data of their own.
- MySQL supports database views or views since version 5.X. In MySQL, almost features of views conform to the SQL: 2003 standard. MySQL process queries to the views in two ways:
  - ✓ MySQL creates a temporary table based on the view definition statement and then executes the incoming query on this temporary table.
  - ✓ First, MySQL combines the incoming query with the query defined the view into one query. Then, MySQL executes the combined query.
- MySQL supports version system for views. Each time when the view is altered or replaced, a copy of the existing view is back up in arc (archive) folder which resides in a specific database folder. The name of back up file is view_name.frm-00001. If you then change the view again, MySQL will create a new backup file named view_name.frm-00002.



1. Creating View

```
#CREATE OR REPLACE
[ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}] VIEW
[database_name].[view_name]
AS [SELECT statement]
```

OR

```
#CREATE VIEW view_name AS
 SELECT column_name(s)
 FROM table_name
WHERE condition
```

2. **Creating Updateable Views**

   In MySQL, views are not only read-only but also updateable. However in order to create an updateable view, the SELECT statement that defines the view has to follow several following rules:

   - ✓ The SELECT statement must only refer to one database table.
   - ✓ The SELECT statement must not use GROUP BY or HAVING clause.
   - ✓ The SELECT statement must not use DISTINCT in the column list of the SELECT clause.
   - ✓ The SELECT statement must not refer to read-only views.
   - ✓ The SELECT statement must not contain any expression (aggregates, functions, computed columns…)

   When you create updateable views, make sure that you follow the rules above.

   #UPDATE *view_name*

   SET *field1=new -valuel, field2=new -value2* [WHERE Clause]

3. **Drop a View**

   A pre-existing view may be deleted from a database using the following statement:

   ```
   #DROP VIEW view_name;
   #DROP VIEW [IF EXISTS] [database_name].[view_name];
   ```

4. **Getting Information About a View**

   All views are the result of an underlying SELECT statement. Sometimes it can be useful to find out what the SELECT statement behind a view looks like. This information can be obtained using the following:

   ```
   #SHOW CREATE VIEW view_name;
   #SHOW CREATE VIEW [database_name].[view_ name];
   ```

5. **Display a View**

We can select the data from the view by executing following query:

```
#SELECT * FROM view_name;
```

6. **Replacing a View**

   An existing view may be replaced with a new view using the same name via the CREATE OR REPLACE VIEW statement:

   ```
   #CREATE OR REPLACE VIEW view name AS select statement
   ```

7. **Create View With Where**

   ```
   #CREATE VIEW view_name AS
     SELECT Column_name FROM Table_name WHERE condition;
   ```

8. **Create View With AND And OR**

   ```
   #CREATE VIEW view_name AS
       SELECT Column_name FROM Table_name WHERE (Condition1 AND Condition2)
            OR (Condition3 AND Condition4);
   ```

9. **Create View With Like**

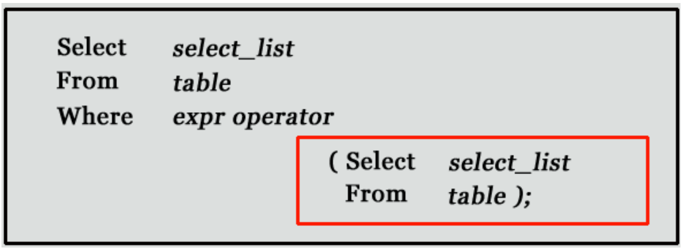
   ```
   #CREATE VIEW view_name AS SELECT Column_name FROM Table_name
           WHERE Column_name NOT LIKE "Pattern%"
           AND Column_name NOT LIKE"% Pattern";
   ```

10. **Create View With Group By**

    ```
    #CREATE VIEW view_name AS SELECT Column_name FROM Table_name
            GROUP BY Column_name;
    ```

(C). Sub Queries:

- A MySQL subquery is a query that is nested inside another query such as SELECT, INSERT, UPDATE or DELETE.
- A MySQL subquery is also can be nested inside another subquery.
- A MySQL subquery is also called an inner query, while the query that contains the subquery is called an outer query.
- **Sub-queries: Guidelines**
  - ✓ A subquery must be enclosed in parentheses.
  - ✓ Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.
  - ✓ If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.



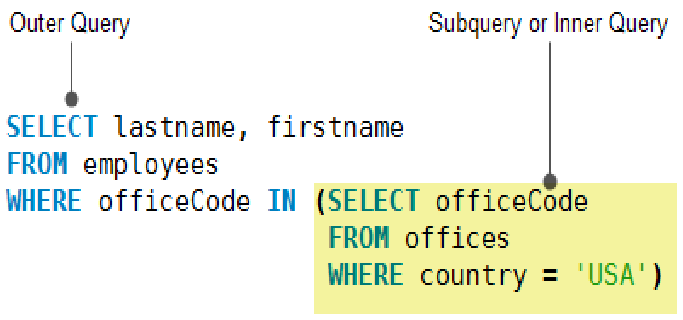
```
Select    select_list
From      table
Where     expr operator
                        ( Select   select_list
                          From     table );
```

- **Example:**

  Let's take a look at the following subquery that returns employees who locate in the offices in the USA.

  ✓ The subquery returns all *offices codes* of the offices that locate in the USA.

  ✓ The outer query selects the last name and first name of employees whose office code is in the result set returned from the subquery.



```
SELECT lastname, firstname
FROM employees
WHERE officeCode IN (SELECT officeCode
                     FROM offices
                     WHERE country = 'USA')
```

IMPLEMENTATION

1. Consider following relation and solve the queries: Create different tables given below with appropriate constraints like primary key, foreign key, check constrains, not null etc.

Account (Acc_no, branch_name,balance)

Branch (branch_name,branch_city,assets)

Customer (cust_name,cust_street,cust_city)

Depositor (cust_name,acc_no)

Loan (loan_no,branch_name,amount)

Borrower (cust_name,loan_no)

1. Create a View1 to display List all customers in alphabetical order who have loan from Pune_Station branch.
2. Create View2 on branch table by selecting any two columns and perform insert update delete operations.
3. Create View3 on borrower and depositor table by selecting any one column from each table perform insert update delete operations.
4. Create Union of left and right joint for all customers who have an account or loan or both at bank

5. Display content of View1,View2,View3
6. Create Simple and Unique index.
7. Display index Information
8. Truncate table Customer.

2. Consider following Relation:

Companies (comp_id, name, cost, year)

C001	ONGC	2000	2010
C002	HPCL	2500	2012
C005	IOCL	1000	2014
C006	BHEL	3000	2015

Orders (comp_id, domain, quantity)

C001	Oil	109
C002	Gas	121
C005	Telecom	115

Create above tables with appropriate constraints execute the following query:

1. Find names, costs, domains and quantities for companies using inner join.
2. Find names, costs, domains and quantities for companies using left outer join.
3. Find names, costs, domains and quantities for companies using right outer join.
4. Find names, costs, domains and quantities for companies using Union operator.
5. Create View View1 by selecting both tables to show company name and quantities.
6. Create View2 on branch table by selecting any two columns and perform insert update delete operations.
7. Display content of View1, View2.

CONCLUSION:

QUESTIONS:

1. What is joins? How to optimize joins in MySQL?
2. Difference between inner join and outer join.
3. How many types of JOIN are supported by MySQL? Which are they? Explain
4. What are advantages and disadvantages of database view?
5. What is use of sub-queries? Explain.