LAB: 07

SQL VIEWS

In SQL, a view is a virtual table or logical table based on the result-set of an SQL statement. Since a database view is similar to a database table, which consists of rows and columns, we can query data against it. The database system stores database views as a SQL SELECT statement. When the actual data of the tables changes, the view mirrors that change as well.

Advantages of SQL Views

Simplicity – A database view allows us to simplify complex queries. Through a database view, we only have to use simple SQL statements instead of complex ones with many joins. We can use database view to hide the complexity of underlying tables to the end-users and external applications.

Security – Views can be made accessible to users while the underlying tables are not directly accessible. This allows the DBA to give users only the data they need, while protecting other data in the same table.

Flexibility – Queries of views may not change when underlying tables change.

Disadvantages of SQL Views

Performance – Querying from database view takes more time than directly querying from the table especially if the view is created based on other views.

Table dependency – We create a view based on the underlying tables of the database. Whenever we change the structure of those tables that view is associated with, we have to change the view as well.

Difference between Temporary Tables and SQL Views

While at first glance, temporary tables and views seem to be similar, there is one key difference between the two.

Temporary tables are base tables that exist only while the session remains active i.e. they are not stored in the database. They need to be re-created in each session and repopulated with data using SELECT/INSERT commands. However, once populated, they no longer have any connection back to the original table(s) from which they were created and, hence, if the data in any of the original table changes, that change is not reflected in the temporary table or vice versa.

SQL views, too, are not stored with data. However, views exist only for a single query and each time a view is generated, it is recreated from the current data in the original table(s).

Therefore, the data in a view is dynamically generated and reflects any changes/updates in the original table(s).

CREATING VIEWS

To create a view in a database, we use the **CREATE VIEW** statement. Views can be created from a single table, multiple tables, or from another view.

- **CREATE VIEW** view_name **AS**
  SELECT col_name(s) **FROM** tb_name
  [*ADDITONAL CLAUSES*];

- **CREATE VIEW** view_name **AS**
  SELECT col_name(s) **FROM** another_view_name
  WHERE *condition(s)*;

WITH CHECK OPTION

The **WITH CHECK OPTION** clause is an optional part of the CREATE VIEW statement. This clause prevents any insertions or updating of rows that are not visible through the view. In other words, whenever we insert or update a row of the base table through a view, MySQL ensures that the insert or update operation is conformed to the definition of the view.

- **CREATE VIEW** view_name **AS**
  SELECT col_name(s) **FROM** tb_name
  [*ADDITIONAL CLAUSES*]
  WITH CHECK OPTION;

For example:

CREATE VIEW hods **AS**
SELECT CONCAT(fname, " ",lname**) AS** Name,
Designation, Email, Extension **FROM** faculty
WHERE designation **LIKE** "%HOD%"
WITH CHECK OPTION;

In the above example, if we wish to enter a new record in the view *hods* where designation does not match "HOD", then MySQL will reject the insert.

TASK

- Create a view StudentView having three columns: student's full name, department name, and GPA. Show the structure of the view.
- Display a list of all the students having GPA greater than 3.0.
- Insert a new record in the Student table. Also, ensure that the student has a GPA.
- Display all the records in Student, StudentInfo, and StudentView.

MODIFYING VIEWS

We can use the following commands to modify an existing view:

- **ALTER VIEW** view_name **AS**
  SELECT col_name(s) **FROM** tb_name
  [*ADDITONAL CLAUSES*];

- **ALTER VIEW** view_name **AS**
  SELECT col_name(s) **FROM** another_view_name
  WHERE *condition(s)*;

- **CREATE OR REPLACE VIEW** view_name **AS**
  SELECT col_name(s) **FROM** tb_name
  [*ADDITONAL CLAUSES*];

- **CREATE OR REPLACE VIEW** view_name **AS**
  SELECT col_name(s) **FROM** another_view_name
  WHERE *condition(s)*;

When using CREATE OR REPLACE VIEW, a view is modified if it already exists. If it does not exist, a new view is created based on the definition provided.

REMOVING VIEWS

Much a table, a view can also be deleted. The command to delete a view is:

- **DROP VIEW (IF EXISTS)** view_name;

TASK

- Modify the view StudentView to have a student's first name and department ID.
- Display all the data of the view.
- Insert a new record in the student table.
- Display all the records in Student, StudentInfo, and StudentView.
- Delete StudentView.

UPDATABLE VIEWS

In MySQL, views can also updatable provided they adhere to certain conditions. In updatable views, we can use INSERT and/or UPDATE statements to insert or update rows of the base table through the updatable view. Similarly, we can also use the DELETE statement to remove rows of the underlying table through the view.

To create an updatable view, the SELECT statement that defines the view must ***not*** contain any of the following elements:

- Aggregate functions such as MIN, MAX, SUM, AVG, and COUNT
- DISTINCT
- GROUP BY clause
- HAVING clause
- UNION or UNION ALL clause
- OUTER join
- Subquery in the SELECT or WHERE clause that refers to the table appeared in the FROM clause
- Reference to non-updatable view in the FROM clause
- Multiple references to any column of the base table.

Additionally, all NOT NULL columns of the base table must be included in the view in order for the INSERT query to function.

Updatable views, however, will allow us to make changes to the base table even if the view itself does not reflect those changes as long as the WITH CHECK OPTION is not used. To illustrate this, consider the following example:

```
CREATE OR REPLACE VIEW hods AS
        SELECT Fname,  Lname, Designation, Email, Extension
                FROM faculty WHERE designation LIKE "%HOD%";

INSERT INTO hods (Fname, Lname, Designation)
        VALUES ("John", "Smith", "Assistant Professor");

SELECT * FROM hods; SELECT * FROM faculty;
```

In the first SELECT statement, the newly entered record (John Smith) will not show because the *hods* view was created to include only those records that have HOD in designation. In the second SELECT statement, however, the record for John Smith would show because the new record was successfully inserted into the base table *faculty* through the view *hods*. This was possible because the WITH CHECK OPTION was not used which allowed the view to accept the new data and make the relevant changes. If, however, the WITH CHECK OPTION was used, an error would be generated and the new record would not be inserted in the *faculty* table either.

To check if a view is updatable, we can use the following command:

```
SELECT Table_Name, Is_Updatable
        FROM Information_Schema.Views
        WHERE Table_Schema = "tb_name";
```

TASK

- Create a view StudentView to have a student's Roll No., first name, and last name.
- Insert a new record in the view.
- Display all the records in Student, StudentInfo, and StudentView.
- Delete a record from StudentView.
- Display all the records in Student, StudentInfo, and StudentView.
- Alter StudentView to restrict first names to have characters "%AE%" in them.
- Using StudentView, add a new student "Michael Leigh".
- Display all the records in Student, StudentInfo, and StudentView.
- Using StudentView, add a new student "Aldina Jenkins".
- Display all the records in Student, StudentInfo, and StudentView.

LAB ASSIGNMENT

1. Create a view EmployeeView having two columns: First name and Date of Hiring. Show the structure of the view.
2. Insert a new record in the Employees table. Display all the records in Employees, EmployeeInfo, and EmployeeView.
3. Alter EmployeeView to restrict it to show only the employees hired in 2020.
4. Using EmployeeView, add a new employee with hiring date "19-12-2012". Display all the records in Employees, EmployeeInfo, and EmployeeView.
5. Using EmployeeView, add a new employee with hiring date "16-08-2020". Display all the records in Employees, EmployeeInfo, and EmployeeView.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. '**cs181xxx_Lab07'**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.