Laboratory Exercise №6

Understanding Views in Databases

Introduction to the problem

In the context of databases, a **view is a virtual table that is based on the result of a SQL query**. It does not store data physically but provides a way to look at data from one or more tables in a structured, query-driven format. Views are used to simplify complex queries, provide security, and offer an abstraction layer for easier data manipulation and retrieval.

One of the key characteristics of views is their virtual nature. They do not occupy physical storage in the database, as they are essentially saved SQL queries. Every time a user queries a view, the database executes its underlying query to produce the results. This dynamic behavior ensures that the data presented by the view is always up to date with the underlying tables. Because views are defined by queries, they can be as simple as retrieving specific columns from a single table or as complex as joining multiple tables and applying aggregations.

Views serve several important purposes. They simplify complex queries by allowing users to encapsulate them into reusable objects. For instance, instead of writing a long query with multiple joins every time, a user can define a view and retrieve the desired data with a single command. Views also enhance security by restricting access to specific data. Instead of granting direct access to a table, a database administrator can create a view that exposes only non-sensitive columns, effectively hiding sensitive information from certain users.

Another significant benefit of views is their ability to abstract the underlying database schema. When a database schema evolves—for example, splitting a table into multiple tables to normalize the design—views can help maintain compatibility with existing applications by presenting the same structure as before. Views can also be used to present aggregated or preformatted data, making them ideal for generating reports or summaries. For instance, a view can provide the total sales grouped by region, saving users the effort of writing the aggregation logic themselves.

While views offer many advantages, **they also have limitations**. Their performance can suffer when dealing with complex queries, especially on large datasets. Additionally, not all views are updatable; views based on multiple tables, aggregations, or complex calculations are typically read-only. Views are also dependent on their underlying tables, meaning any changes to the tables' structure—such as renaming or deleting columns—can render a view invalid.

Views are incredibly versatile and can be used in a variety of scenarios to simplify database management, enhance security, and improve performance. Below are several practical use cases:

1. Simplifying Complex Queries

Scenario: A business analyst frequently needs to query sales data, including customer details, product information, and total sales amount for each transaction. Writing the complex `JOIN` query every time is error-prone and inefficient.

Solution: Create a view that consolidates the required data from multiple tables.

```
CREATE VIEW salesoverview AS
SELECT o.orderid, o.orderdate, c.customername, p.productname, o.quantity * p.price as
totalsales
FROM orders o
```

```
JOIN customers c ON o.customerid = c.customerid join products p on o.productid =
p.productid;
```

Benefit: Users can now retrieve the data with a simple query:

```
SELECT * FROM SalesOverview WHERE OrderDate >= '2024-01-01';
```

2. Restricting Sensitive Data

Scenario: A company's HR database contains sensitive information such as salaries and social security numbers. Junior HR staff should only access employee names, departments, and positions but not salary details.

Solution: Create a view that exposes only the non-sensitive columns.

```
CREATE VIEW publicemployeedata AS
SELECT employeeid, name, department, position
FROM employees;
```

Benefits: Junior staff access the view without risking exposure to sensitive data; Administrators retain access to the full dataset in the underlying table.

3. Providing Aggregated Data

Scenario: A manager wants monthly sales summaries, including total sales and the number of transactions, to track business performance.

Solution: Create a view with aggregated data.

```
CREATE VIEW monthlysalessummary AS
SELECT
    YEAR(orderdate) AS salesyear,
    MONTH(orderdate) AS salesmonth,
    COUNT(*) AS transactions,
    SUM(totalamount) AS totalsales
FROM orders
GROUP BY YEAR(orderdate), MONTH(orderdate);
```

Benefit: The manager can easily query the view for the required data:

```
SELECT * FROM monthlysalessummary WHERE salesyear = 2024;
```

4. Improving Query Performance with Predefined Results

Scenario: An e-commerce website frequently displays a list of active products with their categories and stock availability. Executing the query repeatedly affects performance.
Solution: Create a view that predefines the active product list.

```
CREATE VIEW ActiveProducts AS
SELECT
    p.ProductID,
    p.ProductName,
    c.CategoryName,
    p.StockQuantity
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID
WHERE p.Status = 'Active';
```

Benefit: The application queries the view instead of repeatedly executing the same logic.

5. Enforcing Business Rules

Scenario: A retail company allows only "active" customers to place orders. To enforce this rule, the `Orders` table should accept only data related to active customers.

Solution: Create a view for active customers and use it as a reference in the order entry process.

```
CREATE VIEW ActiveCustomers AS
SELECT CustomerID, Name
FROM Customers
WHERE Status = 'Active'
WITH CHECK OPTION;
```

Benefit: The `WITH CHECK OPTION` ensures that only active customers are used in operations involving this view:

```
INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)
SELECT CustomerID, NOW(), 100.00 FROM ActiveCustomers WHERE CustomerID = 101;
```

6. Supporting Role-Based Data Access

Scenario: In a multi-department system, different roles (e.g., sales, HR, marketing) require access to specific portions of data.

Solution: Create views tailored to each role.

For the Sales team

```
CREATE VIEW SalesTeamView AS
SELECT OrderID, CustomerID, TotalAmount
FROM Orders;
```

For the HR team:

```
CREATE VIEW HRView AS
SELECT EmployeeID, Name, Department
FROM Employees;
```

Benefit: Role-based views ensure that users see only the data relevant to their department.

In summary, views are powerful and versatile tools in database management. They simplify data retrieval, enhance security, and offer a layer of abstraction that protects applications from schema changes. By understanding their features, benefits, and limitations, database users and administrators can effectively leverage views to meet a variety of business and technical needs.

You may find helpful the following online sources of information about SQL syntax related to data types in MySQL, :

- MySQL Views, https://www.w3schools.com/mysql/mysql_view.asp

Case Study „Census Study”

Introduction. You are working for United Nations Statistics Division. The Department Demographic and Social Statistics ask your department for support according to research related to World Population and Housing Census Program. As a database specialist you have an assignment to create several views in their Nepali’s database to facilitate preparation of Regional annual report.

In nepali.sql you’ll find a single table, census. In the census table, you’ll find the following columns:

- `id`, which uniquely identifies each census record
- `district`, which is the name of the census record’s district
- `locality`, which is the name of the census record’s locality within the district
- `families`, which is the number of families associated with the census record
- `households`, which is the total number of households associated with the census record (multiple families may live in the same household)
- `population`, which is the population associated with the census record
- `male`, which is the number of people associated with the census record who have identified as male
- `female`, which is the number of people associated with the census record who have identified as female

Task 1. Write a SQL statement to create a view named `rural`. This view should contain all census records relating to a rural municipality (identified by including “rural” in their name). Ensure the view contains all of the columns from the `census` table.

Task 2. Write a SQL statement to create a view named `total`. This view should contain the sums for each numeric column in `census`, across all districts and localities. Ensure the view contains each of the following columns:

- `families`, which is the sum of families from every locality in Nepal.
- `households`, which is the sum of households from every locality in Nepal.
- `population`, which is the sum of the population from every locality in Nepal.
- `male`, which is the sum of people identifying as male from every locality in Nepal.
- `female`, which is the sum of people identifying as female from every locality in Nepal.

Task 3. Write a SQL statement to create a view named `by_district`. This view should contain the sums for each numeric column in `census`, grouped by district. Ensure the view contains each of the following columns:

- `district`, which is the name of the district.
- `families`, which is the total number of families in the district.
- `households`, which is the total number of households in the district.
- `population`, which is the total population of the district.
- `male`, which is the total number of people identifying as male in the district.
- `female`, which is the total number of people identifying as female in the district.

Task 4. write a SQL statement to create a view named `most_populated`. This view should contain, in order from greatest to least, the most populated districts in Nepal. Ensure the view contains each of the following columns:

- `district`, which is the name of the district.
- `families`, which is the total number of families in the district.
- `households`, which is the total number of households in the district.
- `population`, which is the total population of the district.
- `male`, which is the total number of people identifying as male in the district.
- `female`, which is the total number of people identifying as female in the district.

Task 5. Using the created views, try to answer the following questions:

- How many rural districts are there? How many families live in rural districts?
- How many households are in Nepal?
- Which district has the second lowest number of families? And how many does it have?
- Which district has the highest population? And how many households are in that district?

Case Study “AirBnB”

Introduction. A Bed and Breakfast (“BnB” for short!) is a short-term place one might stay and pay the owner for the service, similar to a hotel. Over the past few years, AirBnB has allowed most anyone to rent out their place, whether it’s a home, a cute cottage, or even a treehouse. You’re a data analyst in Branch Office in Sofia. Your company received database from HQ and your job now is to discover how the rise of AirBnB has changed the local tourist scene. The database schema contains three tables: listing, reviews and availabilities.

The `listings` table contains the following columns:

- `id`, which is the ID of the listing.
- `property_type`, which is the type of the listing (e.g., “Entire rental unit”, “Private room in rental unit”, etc.).
- `host_name`, which is the AirBnB username of the listing’s host.
- `accommodates`, which is the listing’s maximum number of occupants.
- `bedrooms`, which is the listing’s number of bedrooms.
- `description`, which is the description of the listing on AirBnB.

The `reviews` table contains the following columns:

- `id`, which is the ID of the review.
- `listing_id`, which is the ID of the listing which received the review.
- `date`, which is the date the review was posted.
- `reviewer_name`, which is the AirBnB username of the reviewer.
- `comments`, which is the content of the review.

The `availabilities` table contains the following columns:

- `id`, which is the id of the availability.
- `listing_id`, which is the listing ID associated with the availability.

- `date`, which is the date of the availability.
- `available`, which is whether the date is still available to be booked (TRUE or FALSE). In this case FALSE was represented by 0 and TRUE by 1.
- `price`, which is the price of staying on the given date.

Task 1. Write a SQL statement to create a view named `no_descriptions` that includes all of the columns in the listings table **except** for description.

Task 2. Write a SQL statement to create a view named `one_bedrooms`. This view should contain all listings that have exactly one bedroom. Ensure the view contains the following columns:

- `id`, which is the id of the listing from the listings table.
- `property_type`, from the `listings` table.
- `host_name`, from the `listings` table.
- `accommodates`, from the `listings` table.

Task 3. Write a SQL statement to create a view named `available`. This view should contain all dates that are available at all listings. Ensure the view contains the following columns:

- `id`, which is the id of the listing from the `listings` table.
- `property_type`, from the `listings` table.
- `host_name`, from the `listings` table.
- `date`, from the `availabilities` table, which is the date of the availability.

Task 4. Write a SQL statement to create a view named `frequently_reviewed`. This view should contain the 100 most frequently reviewed listings, sorted from most- to least-frequently reviewed. Ensure the view contains the following columns:

- `id`, which is the `id` of the listing from the `listings` table.
- `property_type`, from the `listings` table.
- `host_name`, from the `listings` table.
- `reviews`, which is the number of reviews the listing has received.

If any two listings have the same number of reviews, sort by property_type (in alphabetical order), followed by host_name (in alphabetical order).

Task 5. Write a SQL statement to create a view named `june_vacancies`. This view should contain all listings and the number of days in June of 2023 that they remained vacant. Ensure the view contains the following columns:

- `id`, which is the `id` of the listing from the `listings` table.
- `property_type`, from the `listings` table.
- `host_name`, from the `listings` table.
- `days_vacant`, which is the number of days in June of 2023, that the given listing was marked as available.

Task 6. Using the created views, try to answer the following questions:

- How many listings are there in total?

- How many one-bedroom listings are there? And how many can accommodate at least 4 guests?
- How many listings have availability for December 31st, 2023 (i.e., “2023-12-31”)? How many of those are available on any type of boat?
- How many reviews does the most frequently reviewed property have? And who is the host of that property?
- How many listings were available in June 2023?

Answers Key “Census Study”

1. How many rural districts are there? How many families live in rural districts? Using your `rural` view, you should find there are 461 rural districts with 2,229,834 families.
2. How many households are in Nepal? Using your `total` view, you should find there are 5,642,674.
3. Which district has the second lowest number of families? And how many does it have? Using your `by_district` view, you should find that the Mustang district has only 3,751 families.
4. Which district has the highest population? And how many households are in that district? Using your `most_populated` view, you should find that the most populated is Kathmandu with 275,806 households.

Case Study “AirBnB”

1. How many listings are there in total? Use your no_descriptions view to find that there are 3,973.
2. How many one-bedroom listings are there? And how many can accommodate at least 4 guests? Use your one_bedrooms view to find that of the 1,228 one-bedrooms, 222 of them can accommodate your group of 4.
3. How many listings have availability for December 31st, 2023 (i.e., “2023-12-31”)? Use your available view to find that there are 2,251. How many of those are available on any type of boat? You should find that there are 7. Enjoy your New Year’s Eve afloat!
4. How many reviews does the most frequently reviewed property have? And who is the host of that property? Use your frequently_reviewed view to find that Tiffany’s property has 860 reviews.
5. How many listings were available in June 2023? Use your june_vacancies view to find that there were 1,895 vacancies.