SHUBH PAREEK , 112001039

Assignment 1 : Views and Transactions

(a) Consider the following relation :
Sales (id : integer, region : string, product : string, amount : real)

(i) Suppose you have a view called "HighValueSales" defined as follows :

```
CREATE VIEW HighValueSales AS
SELECT region, product, amount
FROM sales
WHERE amount > 1000 ;
```

Explain what the system will do to process the following query :

```
SELECT H.region, SUM(H.amount) AS total_sales
FROM HighValueSales H
WHERE H.region = ’East’ ;
```

System will throw error ,because h.region is not in any group by clause so aggregate function can’t be used on that.
If only sum was called then this query would have worked.

(ii) Suppose you have a view called "avg_sales_by_region" that shows the average sales by region, based on the "Sales" table. The view has the following definition :

```
CREATE VIEW avg_sales_by_region AS
SELECT region, AVG(amount) AS avg_sales
FROM sales
GROUP BY region ;
```

Explain what happens when you try to execute the following update query on the "avg_sales_by_region"
view :

```
UPDATE avg_sales_by_region
SET avg_sales = avg_sales + 100
WHERE region = ‘North’ ;
```

System will throw error because UPDATE can’t be used on views which have group by clause or aggregate functions.

b) Suppose a user is updating their email address in the application and the application needs to ensure that the new email address is unique before committing the changes to the database, how can transactions be useful ?

Transactions can be useful in this scenario because they provide a way to ensure that the process of updating the email address is completed successfully and consistently, even if multiple operations are involved.

For instance, to ensure the new email address is unique before making any changes to the database, transactions can be used in the following way:

> Initiate a transaction: Begin a transaction to ensure that all database operations related to updating the email address are atomic and are either completed successfully or not at all.
>
> To verify if the new email address is unique: Query the database to check if the new email address is already associated with another user. If the email address is already in use, then the transaction can be rolled back and an error message can be displayed to the user.
>
> Update the email address: If the new email address is unique, update the user's email address in the database.
>
> Commit the transaction: If all database operations related to updating the email address are successful, commit the transaction to make the changes permanent in the database.

Using a transaction ensures that the process of updating the email address is completed consistently and reliably. If any operation fails, the transaction can be rolled back, ensuring that the database is not left in an inconsistent state.

Assignment 2: Roles and Authorizations
(a) Suppose there is a database with the following relations :
employees(id : integer , name : string, salary : real , age : integer)
salaries (id : integer, employee_id : integer, salary : real)
i) The following command is executed :
GRANT INSERT (name, age)
ON employees TO John ;
if John inserts a new row with the following command :
INSERT
INTO employees (name, age)
VALUES (’Smith’, 30) ;

Does the insertion is successful or not ? Justify your answer.

If id is not a primary key, then this insert will work fine , because John is only inserting on attributes which it has insert permission for (name,age).

ii) The authorization to access the relations has been granted to different users :
User "U1" has been granted authorization to access both the "employees" and "salaries" relations.
User "U4" has been granted authorization by "U1" to access the "salaries" relation.
User "U5" has been granted authorization by both "U1" and "U2" to access the "employees" relation.
Suppose the database administrator decides to revoke the authorization of user "U1" to access the "employees" relation.

Which authorizations would be revoked as a result of this action ?

Only “U1” ‘s authorization will be removed on employees. And there will be no effect on authorization for salaries . also since “U5” was provided access by both “U1” and “U2” it will still have authorization.

After the revocation, the access control list for the relations will be as follows:

employees:

- User "U5" has authorization to access the relation.

salaries:

- User "U1" has authorization to access the relation.
- User "U4" has authorization to access the relation.

(b) Suppose we have a database that holds the information of all products in a manufacturing company, and there exist different sets of users like engineers and production managers. The engineers need access to product design specifications and testing data, while the production managers need access to inventory levels and production schedules. How can the database handle this scenario ?

Role based access control can solve this problem .
In this approach, users are assigned roles based on their job responsibilities, and the roles determine the level of access to the data. For example, engineers could be assigned the role of "Design Team" and given access to design specifications and testing data, while production managers could be assigned the role of "Production Team" and given access to inventory levels and production schedules.

Assignment 3: Triggers
(a) Consider the student relation containing the attributes ’ID’, ’Dept_name’, ’Credits’. Write a trigger statement in sql that doesn’t allow inserting students data having ’Credits’ more than 10.0

```
CREATE OR REPLACE FUNCTION validate_credits()
 RETURNS TRIGGER AS $$
BEGIN
   IF NEW.Credits > 10.0 THEN
      RAISE EXCEPTION 'Credits cannot be more than 10.0';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_validate_credits
BEFORE INSERT ON student
FOR EACH ROW
EXECUTE FUNCTION validate_credits();
```

I created a trigger which on each insert calls function validate_credits() which does the required checks asked in the question.

(b) What kind of trigger will you use in the following scenarios and why ? Please mention the trigger structure, no need to write the entire trigger statement.
(i) You don’t want to allow inserting students data who were born before 2000.

We can use a BEFORE INSERT trigger. This is because the trigger will be executed before the insertion of the row, allowing you to check if the birth year of the student is before 2000 and prevent the insertion if it is

(ii) You want to count the number of rows deleted using the DELETE command.

We can use the **after delete trigger** which will get called after the delete statement is called and increase the count of deletes , now to implement a global variable we can use a table and keep updating it after delete has been called to keep count of the deletes .

(iii) You want to move all deleted rows from a table to another table for tracking historical data.
We will use an after delete trigger for each row , which when called will store the deleted row in a table which we have created for tracking historical data.

(c) Which type of trigger is used if we want the triggering statement to complete before executing the trigger action ?

AFTER trigger is useful for this situation.An AFTER trigger is executed after the triggering statement completes, which means that the triggering statement can complete successfully without waiting for the trigger action to complete. This is useful when we want to perform additional actions after the triggering statement has completed, such as auditing, logging, or data validation.

(d) How many times a before update row level trigger will execute if the update statement is affecting 10 rows ?

In general, a row-level trigger will execute once for each row that is affected by the triggering statement, so it will execute 10 times.