Database Systems

Spring 2019

Lab Manual 9

Instructors: Samyan Qayyum Wahla, Sahar Waqar

Graduate Assistants: M. Junaid Zafar, Rao Nasir

Book Reference:

- Chapter 5 T-SQL Fundamentals (This chapter will be fully included in Quiz)

Purpose:

Understanding of

- VIEWS
- STORED PROCEDURE

Instructions:

- Read manually carefully and thoroughly
- Do comment your queries properly
- Only one file will be submitted to eduko named as Lab9.txt will all queries at one plave

Reading Material:

In a database, a **view** is the result set of a *stored* query on the data. A view 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. In other words, In SQL, a view is a virtual table based on the result-set of an SQL statement.

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. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Views, which are a type 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 in such a way 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.

Views can also provide following advantages over tables:

- Views can represent a subset of the data contained in a table. Consequently, a view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.
- Views can join and simplify multiple tables into a single virtual table.
- Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc.) and presents the calculated results as part of the data.

- Views can hide the complexity of data. For example, a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table.
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.
- Depending on the SQL engine used, views can provide extra security.

Note:- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

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.

The basic **CREATE VIEW** syntax is as follows −

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

Note:- You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

DELETING VIEWS

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement.

Syntax:

```
DROP VIEW view_name;
```

view_name: Name of the View which we want to delete.

UPDATE VIEWS

So we have learned about how to create and drop a View, Now time to learn how to **UPDATE** an existing VIEW. But before Learn how to update a view, we should know some more concepts. There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

CREATE OR REPLACE VIEW statement to add or remove fields from a view.

Syntax:

```
CREATE OR REPLACE VIEW view_name AS

SELECT column1,coulmn2,..

FROM table_name

WHERE condition;
```

The WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to a updatable view. If the view is not updatable, then there is no meaning of including this clause in the CREATE VIEW statement.

- The WITH CHECK OPTION clause is used to prevent the insertion of rows in the view where the condition in the WHERE clause in CREATE VIEW statement is not satisfied.
- If we have used the WITH CHECK OPTION clause in the CREATE VIEW statement, and if the UPDATE or INSERT clause does not satisfy the conditions then they will return an error.

Syntax is as follows −

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

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.

Syntax

```
INSERT view_name(column1, column2 , column3,..)

VALUES(value1, value2, value3..);


view_name: Name of the View
```

DELETING 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.

Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.

Syntax:

```
DELETE FROM view_name

WHERE condition;


view_name:Name of view from where we want to delete rows
condition: Condition to select rows
```

STORED PROCEDURE

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs.

Stored procedures can access or modify data in a database, but it is not tied to a specific database or object, which offers a number of advantages.

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So, if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

Note :- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Benefits of using stored procedures:

A stored procedure provides an important layer of security between the user interface and the database. It supports security through data access controls because end users may enter or change data, but do not write procedures. A stored procedure preserves data integrity because information is entered in a consistent manner. It improves productivity because statements in a stored procedure only must be written once.

Stored procedures offer advantages over embedding queries in a graphical user interface (GUI). Since stored procedures are modular, it is easier to troubleshoot when a problem arises in an application. Stored procedures are also tunable, which eliminates the need to modify the GUI source code to improve its performance. It's easier to code stored procedures than to build a query through a GUI.

se of stored procedures can reduce network traffic between clients and servers, because the commands are executed as a single batch of code. This means only the call to execute the procedure is sent over a network, instead of every single sline of code being sent individually.

Stored procedure in SQL:
Stored procedures in SQL Server can accept input parameters and return multiple values of output parameters; in SQL Server, stored procedures program statements to perform operations in the database and return a status value to a calling procedure or batch.

User-defined procedures are created in a user-defined database or in all system databases, except for when a read-only (resource database) is used. They are developed in Transact-SQL (T-SQL) or a reference to Microsoft. Temporary procedures are stored in tempdb, and there are two types of temporary procedures: local and global. Local procedures are only visible to the current user connection, while global procedures are visible to any user after they are created. System procedures arrive with SQL Server and are physically stored in an internal, hidden-resource database. They appear in the SYS schema of each system, as well as in a user-defined database.

Stored Procedure Syntax

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Execute a Stored Procedure

```
EXEC procedure_name;
```

Stored procedure vs. function

Stored procedures and functions can be used to accomplish the same task. Both can be custom-defined as part of any application, but functions are designed to send their output to a query or T-SQL statement. Stored procedures are designed to return outputs to the application, while a user-defined function returns table variables and cannot change the server environment or operating system environment.

Assignment

Script for this homework is available here

1) Write a query to create a view for those salesmen belongs to the city New York.
2) Write a query to create a view for all salesmen with columns salesman_id, name, and city.
3) Write a query to find the salesmen of the city New York who achieved the commission more than 13%.
4) Write a query to create a view to getting a count of how many customers we have at each level of a grade.
5) Write a query to create a stored procedure named **prOrderWithNames** that shows for each order the salesman and customer by name.
6) Write a query to create a stored procedure named **prDailyDetail** to keeping track the number of customers ordering, number of salesmen attached, average amount of orders and the total amount of orders in a day.
7) Write a query to create a stored procedure named **prHieghesScore** that shows the customers who have the highest grade.
8) Write a query to create a stored procedure named **prOrdersAgainestName** that shows the average and total orders for each salesman after his or her name. (Assume all names are unique)
9) Write a query to create a stored procedure named **prMultiCustomers** that shows each salesman with more than one customer.
10) Write a query to create a stored procedure named prOrderOnDate that finds the salesmen who issued orders on October 10th, 2012.