LAB 5: Stored Procedures and Triggers
PART 2

1. Triggers and Stored Procedure

Triggers are a set of instructions that execute automatically after a specific action is done on a database. On the other hand, procedures are a sequence of SQL statements and can be used as a form of modular programming. Procedures are called on demand multiple times and can perform more complex actions. A procedure cannot call a trigger, but a trigger can call a procedure.

Advantages of triggers:

- Protection of data, syncing tables, enforces referential integrity.

Disadvantages:

- Complex debugging, performance overhead.

Advantages of procedures:

- Modular programming, reduced network traffic, data security, easy maintenance, easy debugging/testing.

Disadvantages:

- Manual execution, no event handling.

	Trigger	Procedure
How it works	A trigger is executed automatically when a specified event occurs in the database.	A procedure must be called by the user in order to be executed.
Definition	Cannot define another trigger inside of a trigger.	Can define another procedure inside of a procedure.
Calling	Cannot call another trigger inside of a trigger, only achieved if the actions of the trigger causes another trigger to trigger.	Can call another procedure inside of the procedure.
Transaction statements	Not allowed.	Allowed.
Usage	Maintenance of referential integrity.	Perform tasks defined by the user.
Parameters	Does not allow parameters.	Allows passing parameters.
Return value	Cannot return a value.	Can return 0 to n values.

To conclude, triggers are used to keep the referential integrity of the database because of its automated operations that occur when the set trigger is triggered by an event in the database. On the other hand, Stored procedures are used differently, the user has to call stored procedures manually to perform a defined task, but they allow modular programming and can perform more powerful tasks compared to triggers. Despite the large differences, both tools are essential for every database.

2. Stored procedures and functions

Functions in SQL are divided into two types: 1. Built-in functions such as aggregate and math functions. These functions are available in the system by default. 2. User-defined functions, these functions are created by the user have a single return value which can be either scalar or a table. User defined functions have 3 different types:

1. Scalar function: returns only 1 value, allows usage of declare and control flow statements as well as try-catch.

2. Inline table function: returns a single value in form of a table, does not allow usage of declare and control flow statements, and no try-catch.

3. Multi statement table function: returns a table and allows using control flow statements and declare, as well as try-catch.

Functions must always have a return value, but it is optional to pass a parameter to it. Functions cannot modify data in the DB, they can only use its objects to return a result based on the definition of the function, allowing it to be usable in select statements, where clauses and joins.

On the other hand, a procedure can be used to modify the database if the user desires. Stored procedures are a compiled collection of SQL statements that allows the user to manipulate data, process automation and implement logic based on the requirement. Procedures can have input and output parameters, control flow, as well as transactions, and much more.

	Function	Stored Procedure
Parameter	Can have multiple parameters.	Can have multiple parameters.
Return	Must return a value.	May return a value.
Effect on DB	Can only read the database.	Can perform data manipulation on the database.
Transaction	No transaction statements.	Can have transaction statements.
Usage	Can be used in select, where, having, ...	Used to contain multiple SQL statements in one block.
Out Parameters	No out parameters.	Can have multiple out parameters.
Calling another function/procedure	Cannot call a stored procedure.	Can call a function or another stored procedure.

3. Drop and Delete statements

Drop is used to remove a DB object from the DB such as tables and views and it is a Data Definition Language (DDL) command, on the other hand, delete is

used to remove a row from a table and it is a Data Manipulation Language (DML) command.
Examples:

1. Delete:
    a. Delete from [*table_name*] ~ deletes all rows from the table one by one.
    b. Delete from [*table_name*] WHERE *Id = 123* ~ delete a row that matches the condition.
2. Drop:
    a. Drop table [*Table_Name*] ~ removes the table along with its data
    b. Drop database [*Database_Name*] ~ removes the DB with all its data and objects.

4. Select and select into statements

"Select" displays the result from a query given by the user, on the other hand, "Select into" copies the result and inserts it into a new table instead of displaying it.

Examples:

1. Select
    a. Select * from [*Table_Name*]

2. Select into
    a. Select * into [*New_Table_Name*] from [*Source_Table_Name*]

5. DDL, DML, DCL, DQL

a. **DDL:**

Stands for "Data Definition Language", contains the SQL commands that are used to define the database schema, such as creating the DB itself, or the objects within the DB.

	Syntax	Description
Create	Create table *table_name* (col1 data_type, col2 data_type, ...)	Create database or database objects (tables, views, triggers, etc...).
Drop	Drop table *table_name*	Removes an object from the DB or the DB itself.
Alter	Alter table *table_name* add column *col_name data_type*	Changes the structure of the database by altering its objects.
Truncate	Truncate table *table_name*	Removes the data of a table.
Rename	Exec sp_rename *'current_table_name', 'new_table_name'*	Changes the name of a table.

b. **DML:**

Stands for "Data Manipulation Language", these are the SQL commands that are responsible for manipulation of data in the database.

	Syntax	Description
Insert	Insert into *table_name* values(...)	Inserts a new row of data into the table.
Update	Update *table_name* Set *col_name* = *value* Where *condition*	Updates a column with a new value, can be used with a condition.
Delete	Delete from *table_name* where *condition*	Removes a record from the table where it matches the condition.

c. **DCL**

Stands for "Data Control Language", it is responsible for rights and permissions of the DB system.

	Syntax	Description
Grant	Grant *privilege_name* on *object_name* to *user_name*	Assigns a new privilege to a user which allows them different levels of access to specific objects.
Revoke	Revoke *privilege_name* on *object_name* from *user_name*	Removes assigned privilege from a user on a DB object or actions.

d. **DQL**

Stands for "Data Query Language", contains the commands to perform queries on the DB data to filter and/or order it.

Command:

Select

Syntax:

select *col1, col2, …* from *table_name* where *condition*

Description:

Used to get the data from the database.

6. Table valued and multi statement function

Multi statement function is one of the two types of table valued functions, the other being inline function. Multi statement function is a user defined function that can return multiple rows and columns defined by the user, it is very similar to inline functions with the difference being that the structure of the multi statement table must be defined by the user.

Syntax of multi statement function:

```
Create function function_name (@parameter_name data_type)
Returns @table_name table (col1 data_type, col2 data_type, …)
As
Begin
        ~Multiple statements in the function body
Return
End
```

Advantages of multi statement function:

1. Reusability
2. Improved performance
3. Data transformation
4. Maintainability
5. Security

Disadvantages:
1. Complexity
2. Performance issues if not optimized
3. Limited compatibility

7. Varchar(50) and varchar(max)

Varchar is one of the many data types in SQL, it can hold characters of different lengths in the form of a string of characters. The user can specify the size of the varchar variable such as varchar(50), which will allocate a space for 50 characters for that variable in the memory. varchar(max) will give the variable the maximum length of characters allowed.

8. SQL and windows authentication

Windows authentication uses the windows security model which allows users to use their windows credentials to authenticate. Windows also allows single sign-on options for repeat connections to the SQL server. Windows authentication is more secure than SQL server authentication because it uses existing windows security features for better and more robust encryption methods and policies.

SQl server authentication is independent of Windows credentials, instead, the users authenticate by using a SQL specific username and password that are stored in the SQL server. For each connection, the users must provide their credentials each time. SQL server authentication is useful for applications that access SQL server from non-Windows platforms where windows authentication is unavailable.

9. Inline function and View

Inline functions and views are similar in many different ways, but there are some clear differences between them. One of the differences is that a view cannot have parameters passed to it unlike an inline function, however, views can have triggers which is not possible for functions to have. Another difference is that the output of an inline function can be used directly in a select clause, unlike the result of a view.

	Views	Inline Functions
Parameters	No	Yes
Triggers	Yes	No
Side-effecting operator	Yes	No
Materialization in advance	Yes (indexed Views)	No

10. Identity and unique constraint

Identity is used to automatically generate a number for each row by specifying a starting number and the increment value for each number generation. However, identity does not guarantee the uniqueness of the numbers, as it may be reset due to unexpected server issues.

On the other hand, unique constraint is used to ensure that each row has a unique number and does not allow repetition.