

CL-2005: Database Systems

Lab # 11: Intro to Stored Procedures, Triggers, Functions and Report Generation in SQL Server

Objective:

To make user defined stored procedures and functions in SQL Server.

Scope:

The student shall know the following:

- SQL Commands.
- SQL Queries
- Hands-on experience with the above-mentioned concepts.

Discussion:

First, we will study what are Stored procedures and how to create user defined procedures in SQL server. Let's first create Database and Tables.

```
CREATE TABLE tblEmployee (  
  id INTEGER NOT NULL,  
  firstName VARCHAR(255) NOT NULL,  
  lastName VARCHAR(255) NOT NULL,  
  salary INTEGER NOT NULL,  
  departID INTEGER NULL  
);  
  
CREATE TABLE tblDepartment (  
  id INTEGER NOT NULL,  
  name VARCHAR(255) NOT NULL,  
  location VARCHAR(255) NOT NULL  
);
```

Now add at least 10 rows in each table above.

Stored Procedures

A SQL stored procedure (SP) is a collection SQL statements and sql command logic, which is compiled and stored on the database. Stored procedures in SQL allows us to create SQL queries to be stored and executed on the server. Stored procedures can also be cached and reused. The main purpose of stored procedures to hide direct SQL queries from the code and improve performance of database operations such as select, update, and delete data.

User Defined Stored Procedures

User defined stored procedures are created by database developers or database administrators. These SPs contains one or more SQL statements to select, update, or delete records from database tables. User defined stored procedure can take input parameters and return output parameters. User defined stored procedure is mixture of DDL (Data Definition Language) and DML (Data Manipulation Language) commands.

Now lets create simple procedure name “ShowEmployeeTable” which will display all the values in the table.

```
Create PROCEDURE ShowEmployeeTable
AS
BEGIN
    select*from tblEmployee;
END
```

Noe lets execute this procedure by using keyword “EXEC” as shown below

```
EXEC ShowEmployeeTable;
```

90 %

Results Messages

	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1
2	2	Hamid	Khan	38000	2
3	3	Aftab	Khan	39000	3
4	4	Abhishek	Raj	32000	NULL
5	5	Haji	Ullah	37000	1

Stored Procedures with One Parameter

Now lets create modify above procedure which will take on parameter and will display values where salary will be equal to the value passed as parameter.

```

- Create PROCEDURE ShowEmployeeTableWithParameter @Salary integer
AS
- BEGIN
- select*from tblEmployee
  where salary =@Salary;
- END

```

Now above Procedure takes parameter declared “@” as prefix. This symbol is used to identify variables in procedures. Now query will display those employees whose salary is equal to value passed as parameter by user as shown below.

The screenshot shows a SQL Server interface. At the top, a command window displays the execution of a stored procedure: `EXEC ShowEmployeeTableWithParameter @Salary= 37000;`. Below this, a results pane shows a table with 6 columns: `id`, `firstName`, `lastName`, `salary`, and `departID`. The table contains two rows of data. The first row has `id` 1, `firstName` Sami, `lastName` Ullah, `salary` 37000, and `departID` 1. The second row has `id` 5, `firstName` Haji, `lastName` Ullah, `salary` 37000, and `departID` 1. The interface also shows a zoom level of 90% and tabs for 'Results' and 'Messages'.

	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1
2	5	Haji	Ullah	37000	1

To execute procedure with parameter, as shown above you have to pass value when executing Procedure.

Note: You can delete procedure by DROP keyword.

Stored Procedures with Multiple Parameter

You can pass multiple parameters with comma separation as shown below

```

Create PROCEDURE ShowEmployeeTableWithMultipleParameter @Salary integer, @name varchar(30)
AS
BEGIN
select*from tblEmployee
where salary =@Salary AND firstName=@name;
END

```

Result is shown below

EXEC ShowEmployeeTableWithMultipleParameter @Salary= 37000, @name='Sami';					
90 %					
Results Messages					
	id	firstName	lastName	salary	departID
1	1	Sami	Ullah	37000	1

Triggers In SQL

SQL Server triggers are special [stored procedures](#) that are executed automatically in response to the database object, database, and server events.

There are two types of Triggers:

1. DDL Trigger
2. DML Trigger

DDL Trigger

The DDL triggers are fired in response to DDL (Data Definition Language) command events that start with Create, Alter and Drop, such as Create_table, Create_view, drop_table, Drop_view and Alter_table.

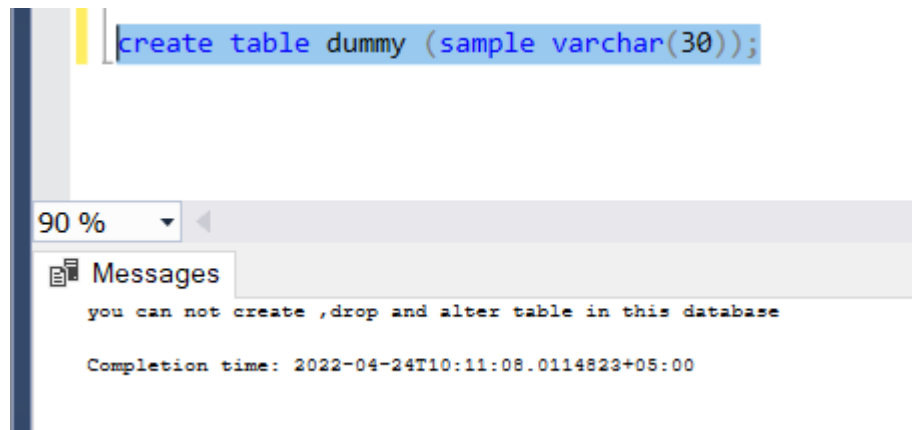
For example we want to se trigger on above mentioned events. We want to make safety trigger such that user can not create, delete or alter any table in the database. Fot that purpose we write a trigger as shown below

```

create trigger saftey
on database
for
CREATE_TABLE,
ALTER_TABLE,
DROP_TABLE
as
BEGIN
print 'you can not create ,drop and alter table in this database'
END

```

Now whenever user tries to create table, user will shown with the error as mentioned in above trigger. Now let's check our trigger



Above screenshot shows us with error when trying to create table in our database.

DML Trigger

The DML triggers are fired in response to DML (Data Manipulation Language) command events that start with Insert, Update, and Delete. Like insert_table, Update_view and Delete_table.

Below is the trigger defined to prevent any operation on Depart table.

```
create trigger Department_safety
on tblDepartment
for
insert,update,delete
as
BEGIN
print'you can not insert,update and delete this table i'
END
```

90 %

Messages

Commands completed successfully.

Completion time: 2022-04-24T10:18:54.1807341+05:00

Now lets test our trigger as shown below

```
create trigger Department_safety
on tblDepartment
for
insert,update,delete
as
BEGIN
print'you can not insert,update and delete this table i'
END

INSERT into tblDepartment VALUES (
10, 'DUMMY', 'FICTION'
);
```

90 %

Messages

you can not insert,update and delete this table i

(1 row affected)

Completion time: 2022-04-24T10:20:57.2679394+05:00

Note as mentioned in definition, it is triggered in response to DML command. That is why command has been also executed and data has been inserted. And at the same time message has been also displayed. To prevent command execution we have further two types of **DML triggers**

AFTER Trigger

AFTER triggers are executed after the action of an INSERT, UPDATE, or DELETE statement.

```
create trigger Department_AFTER
on tblDepartment
AFTER insert
as
BEGIN
print 'Value has been inserted into table'
END
```

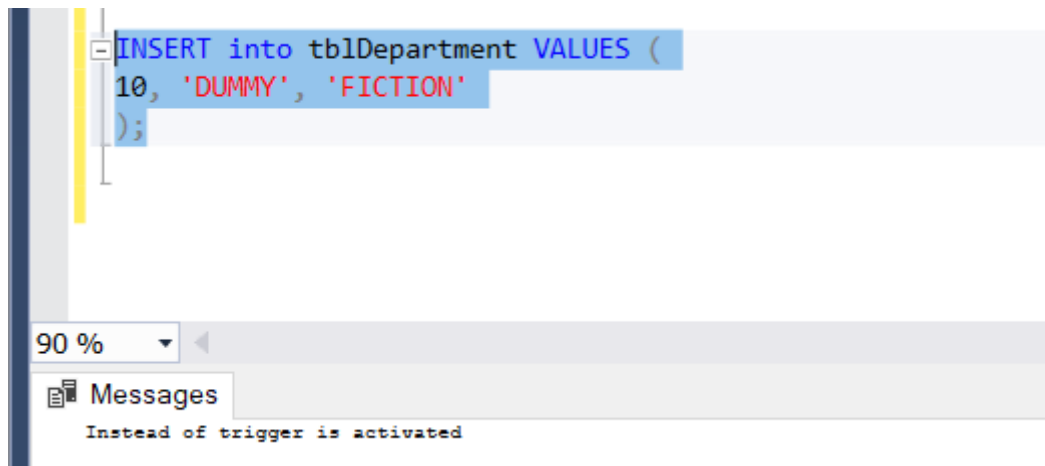
Above trigger is executed after insertion query.

INSTEAD of Trigger

It will tell the database engine to execute the trigger instead of executing the statement. For example below we have defined instead of trigger.

```
create trigger Department_INSTEAD
on tblDepartment
Instead of insert
as
BEGIN
print 'Instead of trigger is activated'
END
```

Now let's test our trigger



CREATING FUNCTIONS IN SQL

Lets create simple function that takes integer value as argument and returns string.

Below is the example of function that takes salary as argument and returns the name of the person who has salary equal to the passed value.

```
create function getNamee (@salary integer)
RETURNS VARCHAR(30)
as
BEGIN

RETURN (select firstName from tblEmployee where salary = @salary);

END
```

Now let's execute the function to test the result.

```
create function getNamee (@salary integer)
RETURNS VARCHAR(30)
as
BEGIN

RETURN (select firstName from tblEmployee where salary = @salary);

END

select dbo.getNamee(32000) as name
```

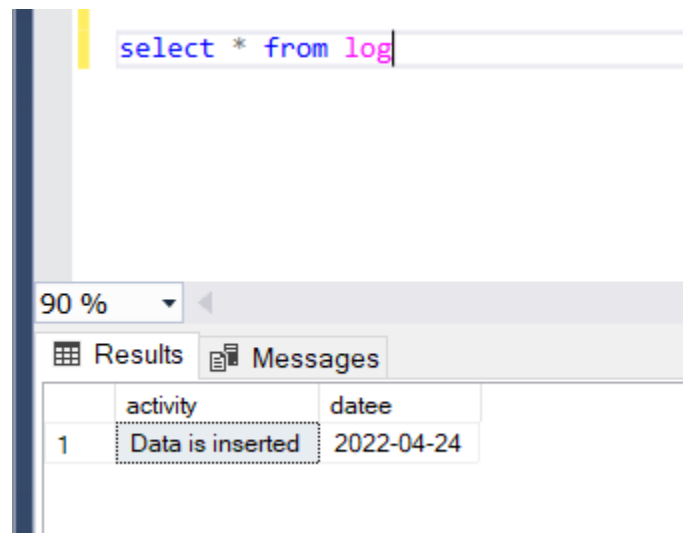
90 %

Results Messages

	name
1	Abhishek

TASK

- Create a Procedure which displays name of the department who has minimum average salary. (HINT : Average salary should be calculated for all respective departments first.)
- Create a Procedure that takes salary and department name as argument. Then it displays employees of that department having salary less than specified salary.
- Create DDL trigger such that when new table is created, display a message that table has been created successfully.
- Create DML after insert trigger that will maintain log. (First create log table with two columns (Activity varchar(40), date). Now whenever data is inserted, log table should be populated with activity and date on insertion as example shown below)



The screenshot shows a database management tool interface. At the top, a SQL query is entered in a text box: `select * from log`. Below the query box, there is a tabbed interface with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'activity' and 'date'. The table contains one row of data: 'Data is inserted' and '2022-04-24'.

	activity	date
1	Data is inserted	2022-04-24

HINT. 'getDate()' function adds data automatically.

- Create Function that returns the sum of all the salaries in department name passed as argument.