Stored Procedure

A stored procedure is group of T-SQL (Transact SQL) statements. If we have a situation where we need to run some queries over and over again, then we can save those specific queries as stored procedure and call it just by its name.

Program and queries are not saved in database. We can save them permanently into a database as procedures and we can execute them any number or times by their names only and no need to write queries/program again and again.

**Note:**

Never use sp\_ prefix for naming user defined stored procedure names.

because:

1. System stored procedure are sp\_ prefixed. So, we can’t distinguish the system stored procedure & user defined stored procedures.
2. Ambiguity problem as it may be that user has given same name to procedure which conflict with already defined stored procedure.

Create Stored Procedure

1. Non-Parameterized stored procedure or Parameter less stored procedure

Create procedure | Proc <ProcedureName>

<parameter/variable declerations>

As

Begin

//code starts here

Select \* from Employee

End

Executing stored procedures

1. Exec | Execute <Stored Procedure Name>
2. <Stored Procedure Name> > select it and press F5
3. Right click on <Stored procedure name> in object explorer > Execute as Stored Procedure.
4. Parameterized Stored Procedure

Parameter defined with keyword “input” are input parameter. If we don’t specify anything then parameter will be considered input parameter.

We must have to pass value for input parameter while calling stored procedure otherwise we will get error in calling stored procedure (i.e., there are the mandatory parameter to pass).

Create proc | procedure <procedure name>

@gender nvarchar(20),

@dptId int input

As

Begin

//Code starts here

Select \* from Employee where @gender = Gender and @dptId = DptId

End

Executing stored procedures

1. Exec | Execute <stored procedure name> ‘Male’, 1212
2. Exec | Execute <stored procedure name> 1212, ‘Male’

We can’t change the order of parameter if we don’t specify keywords while passing values.

1. Exec | Execute <stored procedure name> @<parameterName1>=’Value1’, @<parameterName2>=’Value2’

here, we can change order of passing parameters as parameter name are mentioned.

Stored Procedures with output parameter

Use keyword Out | Output for output parameter.

The stored procedure always returns only single integer either zero or non-zero.

In stored procedure, to return value from stored procedure we use return keyword.

Create Proc | Procedure <procedure name>

@<parameterName1> datatype,

@<outputParameterName2> datatype out|output

As

Begin

//Code or logic goes here

Select @<parameterName2>=Count(\*) from Employee where @parameterName1=’Value1’

End

Executing stored procedures

1. Exec | Execute <stored procedure name> ‘Value1’, @<outputVariableName> Out|Output

Print(@<outputVariableName>)

1. Declare @<outputVariable> datatype

Exec | Execute <stored procedure name> @outputVariable = @outputParameterName2 Out|Output, @<parameterName1>=’Value1’

**Note 🡺 If we don’t specify ‘Out|Output’ keyword while executing stored procedure then variable will be NULL**

Stored Procedure with Output Parameter and return value:

Whenever we execute a stored procedure, it returns an integer status variable.

Status values returns by stored procedure meanings:

1. Return value 0 🡪 indicates 🡪 successful run
2. Return value Non – zero 🡪 indicates 🡪 failure / error in stored procedure

Create proc | procedure EmployeeStoredProc

@totalEmp datatype Out|Output

As

Begin

//Code goes here

Select @<outputVariable>=count(id) from Employee

Return (select sum(salary) from Employee)

End

Executing stored procedures

Declare @totalEmp int

Declare @totalSalary int

Execute @totalSalary=EmployeeStoredProc @totalEmp=@totalEmp out

Print(@totalEmp)

Print(@totalSalary)

**When return value of stored procedure will not work, only output parameter will work?**

Stored procedure only returns integral values so, if we want to get non integral value from stored procedure like nvarchar then we can’t use return to get that value, we must have to use output parameter to get the non-integral value from stored procedure.

Create proc getNameById

@id int

As

Begin

return (select name from Employee where @id=id)

End

Declare @empName nvarchar(20)

Exec @empName=getNameById 1

Print(@empName)

Error: conversion failed converting nvarchar value to data type int.

Solution:

Create proc getNameById

@id int,

@name nvarchar(20) output

As

Begin

select @name=name from Employee where @id=id)

End

Declare @empName nvarchar(20)

Exec getNameById 1, @empName output

Print(@empName)

Optional Parameter in stored procedure

We need to supply values for all parameters of stored procedure. But we can make parameter optional and used default value for them.

We can make stored procedure parameter optional by specify default value for those parameters. So that if we don’t supply values for these parameters then procedure will run with their default assigned values.

//In below stored procedure all the parameter are now optional as all the parameter has default value assigned.

Create proc searchEmployee

@name nvarchar(30)=NULL

@email nvarchar(50)=’abc@mail.com’,

@gender nvarchar(10)=’Male’,

@age int=18

As

Begin

Select \* from Employee where @gender=Gender and @age=Age and @email=email and @name=name

End

Execute searchEmployee @gender=’Male’, @age=29

Execute searchEmployee

This concept is used for search and get records in web forms etc.

Return Status Value VS Output Parameter

|  |  |
| --- | --- |
| **Return Status Value** | **Output Parameter** |
| Only integer datatype | Any data type |
| Only single value | Can be more than one value |
| Use to convey success / failure of stored procedure run. | Use to get value from stored procedure. |

Getting text of stored procedure:

To get the definition of stored procedure we use following ways :

1. Using system stored procedure sp\_helpText

Sp\_helpText ‘<procedure name> 🡪 select it and press F5

1. Select \* from sys.sql\_modules where objectid=object\_id(‘<procedure name>’)

How to encrypt text of stored procedure:

If we encrypt stored procedure then nobody can look into it.

We can encrypt stored procedure by using ‘With Encryption’ statement while creating/altering stored procedure.

We use ‘With Encryption’ statement before ‘As’

Create Proc|procedure GetEmpCountByGenderAndDpt

@Gender nvarchar(10),

@DeptId int

With Encryption

As

Begin

Select \* from Employee where @Gender=Gender and @DeptId=DptId

End

Change/Alter stored procedure:

Alter proc|procedure <procedureName>

As

Begin

//New definition

End

Delete Stored Procedure

Drop proc|procedure <procedure name>

Advantages of Stored procedure over Queries (Inline SQL):

1. Execution plan retention and reusability.

Execution plan: whenever we submit query to sql server. 3 process takes place:

1. Check syntax of query
2. Compiles query
3. Generates execution plan ( It is best possible way to generate the data).

Stored procedures are saved in DB as pre compiled objects. Before saving into db, it is first compiled and if it has no errors (or compiled successfully) then only, it get saved in db (i.e., procedure is precompiled db object).

Procedure gets compiled only once and never compile again until it is altered.

We can call procedures from another procedures, functions or programs but we can call procedures inside DML queries.

In stored procedure, reuse the already generated execution plan again and again until we change stored procedure.

Now days, queries also reuse the same execution plan but very slight change to it like change parameter or add space then it will not reuse the already generated execution plan. It will regenerate new execution plan.

1. Reduce network traffic between client tool and dbms.
2. Code reusability and maintainability increases.
3. Better security (Views are also used for security)

Useful system stored procedures

1. Sp\_help
2. Sp\_helpText
3. Sp\_depends