Sql > Language used to work with RDBMS

Statements in SQL

DDL > create, alter, drop, truncate table

DML > insert delete update

Why Stored procedure??

1. When you write queries, The queries are always compiled
2. All the queries are sent to server separately, even if we select queries
3. We have to write the queries every time

select \* from employees

insert into employees(name, batch, marks) values ('Deepak','B002',90)

select \* from employees

select \* from employees where id<2

1. We want to send these queries to server together

For this we can create batch , we use Go keyword for that

select \* from employees

insert into employees(name, batch, marks) values ('Deepak','B002',90)

select \* from employees

select \* from employees where id<2

Go

The queries are sent to server together, but they are not saved

1. Stored Procedure > Is a group / block of SQl Statements

1. The statements in SP are compiled for the first time. After that, they are stored in a pre-compiled form, which means after first time, it will take less time to execute statements written in SP

2. Reusability

3. Modularity

Syntax of making procedure :

Create proc/procedure <procedurename> {(parameters)}

AS

Begin

SQl Statements

{return}

END

First procedure

create procedure GetEmpoloyes

As

Begin

select \* from employees

End

How to call or execute procedure

exec GetEmployees

How do we get statements of a stored procedure

**sp\_helptext GetEmployees**

create procedure InsertEmployee

As

Begin

Insert into Employees(name, batch, marks)

values('Harpreet','B001',90)

End

--------------------------------------------------

exec InsertEmployee

alter procedure InsertEmployee (@name varchar(20), @batch varchar(4), @marks int)

As

Begin

Insert into Employees(name, batch, marks)

values(@name,@batch,@marks)

End

exec InsertEmployee 'Manoj','B002',90

OR

declare @name varchar(20)

declare @batch varchar(4)

declare @marks int

set @name='Lalit'

set @batch ='B002'

set @marks=98

exec InsertEmployee @name,@batch,@marks

Update

create proc UpdateEmployee(@id int,

@batch char(4), @marks int)

AS

begin

Update Employees Set batch=@batch, marks = @marks

where id=@id

End

exec UpdateEmployee 2, 'B003',89

Delete

create proc DeleteEmployee @id int

As

Begin

Delete Employees where id=@id

End

exec DeleteEmployee 2

Parameters that we pass to SP are by default Input Parameters

Can Procedures return a value also ???

YES

In two ways, SP can return value

1. Return statement > By using return statement, SP can return only one value and that too is in integer

2. Output Parameters > When you want to multiple values from a procedure

How Procedure can return a value???

Create a SP which returns no. of Employees

create proc GetEmployeeCount

AS

Begin

**declare @count int**

Select @count = count(\*) from employees

**return @count**

End

-- Calling Part

declare @count int

exec @count = GetEmployeeCount

print @count

-- Return 1 when record already exists for that PK , else return 0 and insert record

alter proc InsertStudent (@rn int, @name varchar(20),

@marks int, @batchCode char(4),

@address varchar(50),

@dob datetime,

@course varchar(20))

AS

Begin

if(exists(Select \* from student where rn=@rn))

return 1

else

begin

insert into student (rn, name, marks,batchCode, address, DOB, course)

values (@rn,@name,@marks,@batchCode,@address,@dob,@course)

return 0

end

End

-- Call Para Procedure

declare @flag int

exec @flag =InsertStudent 8,'Preeti', 90,'B002','Delhi', '1/08/2002','Java'

if @flag=1

print 'This rn already exist'

else

print 'record inserted'

We want a SP to return more than 1 value , For this, we will use Output Parameters

We pass id, and give us name, batch & marks for that id

create proc GetEmployeeByID(@id int,

@name varchar(20) out, @batch char(4) out, @marks int out)

AS

Begin

Select @name=name , @batch=batch,

@marks=marks from employees where id=@id

End

-- Calling Part

declare @name varchar(20)

declare @batch char(4)

declare @marks int

exec GetEmployeeByID 1,@name out , @batch out, @marks out

print @name

print @batch

print @marks

Pass ID, return details & if record do not exist, ,return 1

-- Give me ID and return details

-- out para are used

alter proc GetEmpDetail (@id int, @name varchar(20) out

, @dept varchar(20) out , @salary int out)

AS

Begin

if(Not(exists(Select \* from Employee where id=@id)))

return 1

else

begin

Select @name = name, @dept = dept , @salary = salary

from Employee where id=@id

return 0

end

End

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Calling Part

declare @name varchar(20)

declare @dept varchar(20)

declare @salary int

declare @flag int

exec @flag= GetEmpDetail 10 , @name out , @dept out, @salary out

if @flag=0

print @name + ' ' + @dept + ' ' + convert(varchar,@salary)

else

print 'Record with this ID do not exist'

Functions Vs Stored Prcedures

Functions are used to perform some specific task

1. Functions have to always return a value Whereas SP may or may not return a value

2. Functions can only have Select statements Whereas SP can have any Sql Statements

3. Functions are always compiled whenever we call them Whereas SP are stored in a pre-compiled form

4. Functions can be called from anywhere, whereas SP cannot be called from functions

5. Functions can be invoked with Select command Whereas SP are called by exec proc name

6. Functions do not support transactions, Exception Handling Whereas SP supports both

7. Functions have only input parameters, Whereas SP can have input & output parameters both

Functions can be System Function OR User Defined Functions

System Functions

Functions cud be scalar OR Aggregate / Group Functions

Scalar functions take single value , and give single result also

String functions : Len() , left()

Date functions : DatePart()

Numeric functions : max() min()

UDF : User Defined

-- Functions are similar to SP in concept

--create function functionname ()

-- returns type

-- AS

--Begin

-- statements

-- return value

-- end

-------------------------------------------

create function MaxMarks()

returns int

As

Begin

declare @count int

Select @count = max(marks) from Employees

return @count

End

-- Calling Function

select dbo.MaxMarks()

create function GetEmpsCount()

returns int

As

Begin

declare @count int

Select @count = count (\*) from Employee

return @count

End

-- Calling Function

select dbo.GetEmpsCount()

create function CombineStrings(@string1 varchar(20) ,

@string2 varchar(20))

returns varchar(40)

As

Begin

return concat(@string1, ' ', @string2)

End

select dbo.CombineStrings('Ajay','Sood')

select dbo.CombineStrings(name , batch) from Employees

alter function CombineNames(@para1 varchar(20),

@para2 varchar(20))

returns varchar(40)

AS

Begin

declare @temp varchar(40)

--set @temp = @para1 + ' '+ @para2

set @temp= CONCAT(@para1, ' ' , @para2)

return @temp

end

select dbo.CombineNames('Deepak', 'Kumar')

select dbo.combineNames(name,dept) from Employee

User-Defined Functions

SQL Server supports two types of user-defined functions:

1. Table-Valued Functions
2. Scalar Valued Functions : Functions which returns a single value

Table-Valued Functions

In this type of function, we select a table data using a user-created function. A function is created using the Create function SQL command. The following query creates a new user-defined function.

Create function Fun\_EmployeesInformation()

returns table

as

return(select \* from Employees )

--Calling Part

select \* from dbo.Fun\_EmployeesInformation()

-- FUnctions

-- String Functions

-- Numeric functions

-- Date Functions

-- General Functions

select GetDate()

Select Datepart(year, GetDate())

Select Datepart(month, GetDate())

select DateAdd(yyyy,2, GetDate())

select Getdate()

select CONVERT(VARCHAR(19),GETDATE())

select CONVERT(VARCHAR(19),GETDATE(),103)