Stored Procedures

Stored procedures

- Store procedures are **SQL codes** consisting of **declarative** and **procedural** SQL statements stored in the **catalog of a database** that can be **activated** by calling it from a program, a trigger or another stored procedure.
- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- stored procedures are used to group one or more Transact-SQL statements into logical units.
- When you call a stored procedure for the first time, SQL Server creates an execution plan and stores it in the cache.
- In the subsequent executions of the stored procedure, SQL Server reuses the plan to execute the stored procedure very fast with reliable performance.
- SQL Server compiles each Stored Procedure once and then reutilizes the execution plan
- Some of the **benefits** of using stored procedures are:
  - They offer improved performance because of compiled code.(*Precompiled execution*)
  - Code reuse and abstraction.(*Efficient reuse of code and programming abstraction*)
  - Permissions can be granted for stored procedures while being restricted for the underlying tables.(*Enhanced security controls*)
  - This allows the DBA to provide a method for SQL programmers and report writers to access and/or manipulate data in a safe way.

Cont….

- Since database operations can be performed inside the stored procedures, they provide a **strong level of security**.

- Instead of access being granted to the underlying object, permission can be granted only to the stored procedure.

- Essentially, stored procedures **create a level of abstraction for permissions**—instead of the user being granted SELECT, INSERT, UPDATE, or DELETE rights, the user can be granted EXECUTE rights to a stored procedure

Stored Procedures v$_s$ Functions:

User Defined Function	Stored Procedure
Function must return a value.	Stored Procedure may or not return values.
Will allow only Select statements, it will not allow us to use DML statements.	Can have select statements as well as DML statements such as insert, update, delete and so on
It will allow only input parameters, doesn't support output parameters.	It can have both input and output parameters.
It will not allow us to use try-catch blocks.	For exception handling we can use try catch blocks.
Transactions are not allowed within functions.	Can use transactions within Stored Procedures.
We can use only table variables, it will not allow using temporary tables.	Can use both table variables as well as temporary table in it.
Stored Procedures can't be called from a function.	Stored Procedures can call functions.
Functions can be called from a select statement.	Procedures can't be called from Select/Where/Having and so on statements. Execute/Exec statement can be used to call/execute Store Procedure.
A UDF can be used in join clause as a result set.	Procedures can't be used in Join clause

Syntax:

```
CREATE PROCEDURE procedure_name
        (@param1 data_type,@param2 data_type @param3 data_type……)
         AS
        -- SQL statements
```

Creating a simple Stored_ Procedure

```
create procedure spcustomer----creating procedure or proc
as
begin
    select *
    from customer where Sex='f'
end

exec spcustomer---executing procedure
execute spcustomer---or
spcustomer---we can also execute using procedure name
```

Stored Procedure With Multiple Parameters

```
create proc spbyparameter
    @address varchar(50),
    @sex char(1)
as
begin
    select * from customer
    where [Address]=@address and Sex=@sex
end

exec spbyparameter 'bahir dar','m'------or
exec spbyparameter @Address='bahir dar',@sex='m'
sp_helptext spbyparameter--to view the definition of
the stored procedure
```

Using output parameters in stored_porcedure

```
create procedure spcustomerbygender
    @sex char(1),
    @count int output ----Stored procedures with output parameters
as
begin
    select @count=COUNT(Customer_ID) from customer
    where Sex=@sex
end

declare @customer_total int
execute spcustomerbygender 'm',@customer_total output
---or out,if there is no output keyword the output
will be null
print @customer_total
```

Stored procedure output parameters or return values

```
create proc sptotal
@totalcount int out
as
begin
select @totalcount=COUNT(customer_id) from customer
end

declare @totalcustomer int
execute sptotal @totalcustomer output
select @totalcustomer as [total customer]
```

Stored procedure output parameters or return values

```
create proc spreturn
as
begin
return (select COUNT(customer_id) from customer)
end

declare @totalcustomer int
execute @totalcustomer = spreturn

select @totalcustomer as [total customer]
```

Stored procedure output parameters

```
create proc spnamebyid
@id varchar(10),
@name varchar(50) output
as
begin
select @name=first_name from customer where
customer_id=@id
end
declare @custname varchar(50)
exec spnamebyid 'c-110',@custname output
print 'first name is : '+@custname
```

Example (cont....)

```
CREATE PROCEDURE spFindMax
 @var1 int,
 @var2 int,
 @max int output
AS
BEGIN
 IF @var1>@var2
SET @max=@var1
 ELSE
 SET @max=@var2
END

declare @result int
exec spFindMax 30,20,@result output
select @result as [your result is]
```

Example (cont....)

```
create proc sppurchaseleve
as
begin
select product_name,purchase_price,purchase_level=
case
    when purchase_price>=5000 then 'very hight price'
    when purchase_price>=3000 and purchase_price<5000
then 'medium price'
    when purchase_price<3000 then 'low price'
    else 'price unknown'
end
from product
end

execute sppurchaseleve
```

Example (cont....)

```
create proc spdeletebyid
@custid varchar(10)
as
begin
delete from customer where customer_id=@custid
end

exec spdeletebyid 'c-100'
```

Default parameter in procedure

```
create proc spdefaultparameter
@sex char(1)='m'
as
begin
select * from customer where sex=@sex
end

exec spdefaultparameter
```

```
create proc spdefaultparameter
@sex char(1)='m'
as
begin
select * from customer where sex=@sex
end

exec spdefaultparameter 'f'
```

Default parameter in procedure(cont.…)

```
create proc spdefault
@val int =20
as
begin
select @val as 'result'
end

exec spdefault

exec spdefault 30
```

Stored procedure with function

```
create proc sppricelevel
@productid varchar(10)
as
begin
select product_name,purchase_price,dbo.myfun(product_id)as 'price level'
from product
where @productid=product_id
end

--function goes here
create function myfun( @proid varchar(10))
returns varchar(50)
as
begin
declare @pricelevel varchar(20)
declare @price money
set @price=(select purchase_price from product where product_id=@proid)
if @price>=5000
      set @pricelevel='very high price'
else if @price>=3000 and @price<5000
      set @pricelevel='medium price'
else
      set @pricelevel='low'
return @pricelevel
end

exec sppricelevel 'p-110'
```

Example: Student table

	fname	Lname	StudId	sex	mark
1	Aster	Alemu	100	Female	56
2	Chala	Yihun	101	Male	65
3	Tigst	Belachew	102	Female	80

- Assume that we have the above records
  - When **StudId** is given as an argument, suppose that we want to see the **Fname**, **Lname** and **mark** of a student
    - Write a stored procedure to achieve this purpose.

Example

```
create procedure stmark (@id varchar(20))
as
select fname, lname,dbo.stgrade(StudId)
from student
where  @id=StudId
```

Function

```
create function fngrade( @id varchar(20)) returns varchar
as
begin
    declare @grade varchar(20)
    declare @m int
    set @m=(select mark from student where StudId=@id)
    if @m>80
        set @grade='A'
    else if @m>60
        set @grade='B'
    else
        set @grade='C'
    return @grade
end
```

Execute procedure

```
execute  prfrade '100'
```

Exercise

- Create a procedure that adds(update) a student mark(if mark is greater than 80 add 10 and if the mark

Function

```
create function Ufunction(@mark int)
returns int
as
begin
    declare @m int
    set @m=(select mark from student where @mark=mark)
    return @m
end
```

Procedure

```
create procedure Addmark
as
begin
    update student set mark=mark+10 where dbo.Ufunction1(mark)>80
    update student set mark=mark+100 where dbo.Ufunction(mark)>50 and
    dbo.Ufunction(mark)<80
end
```

- Execute the procedure

execute addmark