**Funcții și proceduri stocate**

Funcțiile și procedurile stocate reprezintă mulțimi de instrucțiuni Transact SQL precompilate, memorate în baza de date, ce se execută cu ajutorul comenzilor de apel. Funcțiile și procedurile stocate sunt obiecte ale bazei de date memorate de o manieră durabilă asemănător tabelelor. Funcțiile și procedurile stocate se execută direct pe server, astfel traficul în rețea între client și server este redus la regia lor de apelare.

Funcțiile și procedurile stocate beneficiază de acces direct la baza de date ceea ce permite o prelucrare rapidă a informaţiilor.

Sintaxa ce permite crearea unei proceduri stocate Transact-SQL este următoarea:

CREATE {PROC | PROCEDURE}[schema.]nume\_procedura[; numar]

     [{@parametru tip\_data}[VARYING][=valoare\_implicita]

[OUT|OUTPUT | [READONLY]] [,...n ]

[WITH {[ENCRYPTION][RECOMPILE]}]

[FOR REPLICATION]

AS {[BEGIN]instructiune\_sql [;] [...n] [END]}[;]

Sintaxa ce permite modificarea unei proceduri stocate Transact-SQL este următoarea:

ALTER {PROC | PROCEDURE}[schema.]nume\_procedura[; numar]

     [{@parametru tip\_data}[VARYING][=valoare\_implicita]

[OUT|OUTPUT | [READONLY]] [,...n ]

[WITH {[ENCRYPTION][RECOMPILE]}]

[FOR REPLICATION]

AS {[BEGIN]instructiune\_sql [;] [...n] [END]}[;]

Sintaxa ce permite stergerea unei proceduri stocate Transact-SQL este următoarea:

DROP { PROC | PROCEDURE } { [schema.] nume\_procedura }

O funcţie este un subprogram utilizat, în general, pentru a calcula o valoare. Structura unei funcţii este asemănătoare structurii unei proceduri, singura diferenţă constând în faptul că o funcţie trebuie să conţină o instrucţiune RETURN.

Sintaxa ce permite crearea unei funcţii este următoarea:

CREATE FUNCTION [schema.] nume\_functie

([{ @nume\_parametru [AS] tip\_data [= valoare\_implicita]

[READONLY]} [,...n]])

RETURNS tip\_data\_returnat

 BEGIN

   instructiune\_sql [;] [...n]

END

Sintaxa ce permite modificarea unei funcţii este următoarea:

ALTER FUNCTION [schema.] nume\_functie

([{ @nume\_parametru [AS] tip\_data [= valoare\_implicita]

[READONLY]} [,...n]])

RETURNS tip\_data\_returnat

 BEGIN

   instructiune\_sql [;] [...n]

END

Sintaxa ce permite stergerea unei funcţii este următoarea:

DROP FUNCTION [IF EXISTS] [schema.] function\_name[;]

**schema** este numele schemei căreia îi aparţine procedura/funcţia

**nume\_procedura, nume\_functie** reprezintă numele procedurii stocate respectiv al funcției care se crează. Procedurile stocate( nu și funcțiile) pot fi create în baza de date de sistem *TempDb (baza de date cu obiecte temporare)*. Procedurile temporare locale au numele prefixat cu # iar procedurile temporare globale au numele prefixat cu ##.

**valoare\_implicita** este o valoare implicită pentru parametru. În cazul în care o valoare implicită este definită, funcţia poate fi executată fără a preciza o valoare pentru acest parametru.

**Număr** reprezintă un întreg care este folosit pentru a crea um grup de proceduri cu același nume. Aceste proceduri pot fi sterse cu o singură comandă DROP PROCEDURE având ca parametru numele comum al procedurilor din grup.

**OUT | OUTPUT** stabilește faptul că parametrul corespunzător este parametru de ieșire, astfel putem putem returna valoarea curentă a parametrului către programul care apelează procedura atunci când procedura se termină. Pentru a salva valoarea parametrului într-o variabilă care poate fi folosită de programul apelant, acesta trebuie să folosească cuvântul cheie OUTPUT la executarea procedurii.

Parametrul **READONLY** indică faptul că parametrul nu poate fi actualizat sau modificat în definiţia funcţiei

**RECOMPILE** indică faptul că procedura va fi recompilată inainte de fiecare execuție a sa.

**ENCRYPTION** indică faptul că textul procedurii va fi memorat criptat.

Exemple de funcții utilizator:

1)

/\* Calculul mediei notelor unui student identificat prin codStudent. Sunt luate in calcul doar notele >= 5

\*/

create function DAVG(@CodStudent char(10))

returns decimal(4,2)

as

BEGIN

declare @S float

declare @sql varchar(255)

select @S=Avg(convert(decimal(4,2),Nota))

from tNote where CodStud=@CodStudent and nota>=5;

return round(@S,2,1); -- trunchiere la 2 zecimale

END

--Exemplu de apel

print dbo.DAVG('S3')

2)

/\* Calculul sumei salariilor la nivel de departament \*/

create function SumaSalPeDep(@codDep char(10))

Returns int

AS

BEGIN

declare @SumaSal int

Select @SumaSal=sum(salariu) from tAngajati where codDep=@codDep

return @sumaSal

end

Exemplu apel

Declare @S int

Set @S=dbo.SumaSalPeDep('d1')

if @s is null

print 'Departament necunoscut'

else

print 'Suma sal dep d1 este ' +convert(char(10),@s)

3)

/\* Formatare text. Fiecare cuvant din text va fi scris cu prima litera mare si restul cu litere mici

\*/

create function Formatare\_tip\_Nume( @text varchar(256))

returns varchar(256)

AS

begin

declare @s varchar(256)

declare @n int

set @text=rtrim(ltrim(@text)) /\* am eliminat spatiile de la

inceputul si sfarsitul textului\*/

set @s=''

while (len(@text)>0)

begin

set @n=charindex(' ',@text)

if @n=0 set @n=len(@text)

set @s=@s+upper(substring(@text,1,1)) +

lower(substring(@text,2,@n-1))

set @text=ltrim(substring(@text,@n+1,255))

end

return @s

end

-- Exemplu de apel

declare @Nume varchar(50)='POPESCU angela ElENa'

print dbo.Formatare\_tip\_Nume(@Nume)

-- Se va afisa

-- Popescu Angela Elena

4)

Fie tabelul *tZileLibere* care conține zilele, din anul curent, sarbători naționale, declarate zile libere. *tZileLibere* este creat astfel:

CREATE TABLE tZileLibere

( ziLibera smalldatetime NOT NULL )

Funcția **nrZileLucratoare** următoare**,** determină numărul de zile lucrătoare din perioada [data1,data2], adică din numărul total de zile se exclud zilele de sâmbătă, duminică și zilele sărbători naționale.

create function dbo.nrZileLucratoare(@data1 smalldatetime, @data2 smalldatetime)

returns int

AS

BEGIN

declare @d smalldatetime

declare @zi as int

declare @nrZileLucratoare int

set @nrZileLucratoare=0

set @d=@data1

while @d<=@data2

begin

set @zi=datepart(dw,@d)

if @zi!=1 and @zi!=7 and

@d not in (select ziLibera from tZileLibere)

set @nrZileLucratoare=@nrZileLucratoare+1

set @d=@d+1

end

return @nrZileLucratoare

end

APELUL FUNCTIEI

Presupunem că în tabelul *tZileLibere* sunt inserate zilele sărbători naționale libere

print dbo.nrZileLucratoare( '04/01/2020','04/30/2020')

declare @n int

set @n=dbo.nrZileLucratoare( '04/01/2020','04/30/2020')

print @n

select @n=dbo.nrZileLucratoare( '04/01/2020','04/30/2020')

print @n

5)

Urmatoarea functie returneaza data calendaristica transmisa prin parametru, trunchiata prin renuntarea la ora, minut, secunda

create function truncData(@data datetime)

returns datetime

as

begin

return convert(datetime,convert(char(8),getdate(),112),112)

end

Exemplu de apel:

select dbo.truncData(getdate())

Exemple de proceduri stocate

1)

create procedure ps\_ListaStudenti @CodSpec varchar(11)='%',

@Nume varchar(50)='%'

as

begin

select CodSpec,CodStud,Nume,CNP

from tStudenti

where CodSpec like @CodSpec

and Nume like @Nume

end

Exemple de apel:

1. afiseaza toti studentii

execute ps\_ListaStudenti

1. afiseaza studentii de la info

execute ps\_ListaStudenti 'Info'

1. afiseaza studentii de la info al caror nume incepe cu litera A

execute ps\_ListaStudenti 'Info', 'A%'

**Utilizarea codului de retur**

Procedurile stocate pot returna o valoare de tip întreg, denumită cod de retur, folosind comanda

**return [expresie-intreaga]**

La apel, codul de retur poate fi ignorat.

Exemplificare:

Vom modifica procedura stocata anterioara astfel incat sa returneze numarul de randuri afisate

alter procedure ps\_ListaStudenti @CodSpec varchar(11)='%',

@Nume varchar(50)='%'

as

begin

select CodSpec,CodStud,Nume,CNP

from tStudenti

where CodSpec like @CodSpec

and Nume like @Nume

return @@rowcount

end

/\*

Variabila globala @@rowcount returnează numărul de rânduri afectate de ultima comanda SQL, in acest caz comanda select

\*/

Exemple de apel:

1. fara utilizarea codului de retur

execute ps\_ListaStudenti 'Info'

1. cu utilizarea codului de retur

declare @r int

execute @r=ps\_ListaStudenti 'Info'

if @r=0 print 'Nu exista studenti conform solicitarilor'

else print'Am gasit '+str(@r,3) +' studenti'

2)

create procedure ps\_Sinteza\_Note @NrStudRestantieri int out,

@NrStudPromovati int out,

@CodSpec varchar(11)='%',

@CodCurs varchar(11)='%'

as

begin

select @NrStudRestantieri=count(case when nota<5 then 1 end),

@NrStudPromovati=count(case when nota>=5 then 1 end)

from tStudenti as A inner Join tNote as B on

A.CodStud=B.CodStud

where CodSpec like @CodSpec and codCurs like @CodCurs

end

utilizare

declare @NrStudRest int, @NrStudPromovati int

execute ps\_Sinteza\_Note @NRStudRest out, @NrStudPromovati out ,'info'

select @NrStudRest as [Nr stud restantieri], @NrStudPromovati as NrStudPromovati