

## VJEŽBA 8 – KORISNIČKE FUNKCIJE I POHRANJENE PROCEDURE

### KORISNIČKE FUNKCIJE

Korisničke funkcije predstavljaju Transact-SQL izraz koji prima parametre (argumente), temeljem njih izvršava akcije, poput kompleksnih izračuna, te vraća rezultat akcije kao vrijednost.

Vrijednost funkcije može biti skalarna (jedinstvena) vrijednost ili tablica.

### ARGUMENTI FUNKCIJE

Argumenti se proslijeđuju funkciji navođenjem neposredno nakon imena funkcije. Argument je definiran nazivom, te tipom podatka. Argument funkcije se **OBAVEZNO** navodi s znakom @ kao prefiksom, te tipom podatka u nastavku. Vrijednost SVIH deklariranih argumenata mora biti dana prilikom poziva funkcije, ukoliko nije zadana pred definirana vrijednost u samoj deklaraciji funkcije. Uz deklaraciju argumenata može se dodati i njihova default vrijednost koja će biti upisana ukoliko se vrijednost argumenta ne proslijedi.

### SINTAKSA:

```
CREATE FUNCTION NazivFunkcije
(@parametar1 data_type = default_vrijednost,
@parametar2 data_type = default_vrijednost,
@parametar3 data_type = default_vrijednost)
RETURNS @varijabla
AS
BEGIN
    ...
    RETURN
END
```

### SKALARNA FUNKCIJE

Skalarne korisničke funkcije vraćaju jednu vrijednost skalarnog podatkovnog tipa poput integer, varchar(n), char(n), money, datetime, bit itd. Skalarne korisničke funkcije mogu također vraćati korisnički definirane podatkovne tipove ukoliko su oni bazirani na skalarnom podatkovnom tipu. Međutim, skalarne funkcije ne mogu vratiti sve tipove podataka, npr.: text, ntext, image, cursor, ili timestamp.

```
/* Kreirai korisničku funkciju koja određuje je li potrebno ponovno
naručiti proizvod. */

CREATE FUNCTION fnNeedToReorder (@nReorderLevel INT, @nUnitsInStock INT,
                                @nUnitsOnOrder INT)
    RETURNS VARCHAR(3)
AS
BEGIN
    DECLARE @sReturnValue VARCHAR(3)

    IF ((@nUnitsInStock + @nUnitsOnOrder) - @nReorderLevel) < 0
```

```

        SET @sReturnValue = 'Yes'
    ELSE
        SET @sReturnValue = 'No'

    RETURN @sReturnValue
END
GO

SELECT ProductID,
       ReorderLevel,
       UnitsInStock,
       UnitsOnOrder,
       dbo.fnNeedToReorder(ReorderLevel, UnitsInStock, UnitsOnOrder) AS
       needToReorder
FROM Products

```

Uočimo da se korisnički definirana skalarna funkcija poziva korištenjem imena koje se sastoji iz naziva vlasnika objekta(object owner) i naziva samog objekta.

#### FUNKCIJE ČIJA JE VRIJEDNOST TABLICA

Postoje dvije skupine funkcija čija je vrijednost tablica :

- inline funkcije
- one koje sadrže više t-sql izraza (multistatement funkcije)

Korisničke inline funkcije vraćaju skup zapisa kroz podatkovni tip TABLE. One se definiraju jednim SELECT izrazom koji tvori tijelo funkcije. Takve funkcije ne mogu sadržavati dodatnu T-SQL logiku van SELECT izraza koji definira vrijednost funkcije. Kod takvih funkcija nije potrebno definirati strukturu tablice koju funkcija vraća.

Korisničke inline funkcije mogu se referirati odnosno povezivati poput tablica, jer vraćaju skup zapisa preko podatkovnog tipa TABLE.

```

/* Izraditi funkciju koja vraća sve zaposlenike iz određenog grada */

CREATE FUNCTION fnGetEmployeesByCity (@sCity VARCHAR(30))
    RETURNS TABLE
AS
RETURN
(
    SELECT      FirstName, LastName, Address
    FROM        Employees
    WHERE       City = @sCity
)
GO

SELECT * FROM fnGetEmployeesByCity('seattle')

```

Uočimo da kod poziva inline korisničke funkcije nije potrebno navoditi vlasnika objekta.

Multistatement korisničke funkcije eksplicitno definiraju strukturu tablice koju vraćaju, preko naziva atributa i tipova podataka odmah nakon RETURNS izraza.

```

/* Izraditi funkciju koja vraća sve zaposlenike iz određenog grada.
   Ukoliko za poslenici ne postoje, ispisati: Nije pronađen niti jedan
   zaposlenik u danom gradu.*/

CREATE FUNCTION fnGetEmployeesByCity3 (@sCity VARCHAR(30))
    RETURNS @tblMyEmployees TABLE
    (
        FirstName VARCHAR(20),
        LastName VARCHAR(40),
        Address VARCHAR(120)
    )
AS
BEGIN
    INSERT    @tblMyEmployees
    SELECT    FirstName, LastName, Address
    FROM      Employees
    WHERE     City = @sCity
    ORDER BY LastName

    IF NOT EXISTS (SELECT * FROM @tblMyEmployees)
        INSERT @tblMyEmployees (Address)
            VALUES ('Nije pronađen niti jedan zaposlenik u danom gradu.')

    RETURN
END

SELECT * FROM fnGetEmployeesByCity3('Split')

```

## IZMJENA KORISNIČKE FUNKCIJE

Ukoliko je postojeću funkciju potrebno izmijeniti, jednostavno zamjenjujemo ključnu riječ **CREATE** sa ključnom riječju **ALTER**.

```

ALTER FUNCTION fnNeedToReorder (@nReorderLevel INT, @nUnitsInStock INT,
                                @nUnitsOnOrder INT)
    RETURNS VARCHAR(3)
AS
BEGIN
    DECLARE @sReturnValue VARCHAR(3)

    IF ((@nUnitsInStock + @nUnitsOnOrder) - @nReorderLevel) < 0
        SET @sReturnValue = 'Yes'
    ELSE
        SET @sReturnValue = 'No'

    RETURN @sReturnValue
END
GO

```

## BRISANJE KORISNIČKE FUNKCIJE

Korisnička funkcija se briše izrazom **DROP FUNCTION**:

```
DROP FUNCTION fnNeedToReorder
```

## POHRANJENE PROCEDURE

Pohranjene procedure predstavljaju grupe Transact-SQL izraza kompajliranih u jedinstveni plan izvršavanja. Sastoje od jedne ili više SQL naredbi pohranjenih pod jedinstvenim imenom u bazi podataka.

Pohranjene procedure su osnova svake klijent-server aplikacije, te omogućavaju da se obrada podataka obavi na serveru. Jednom napisanu pohranjenu proceduru može koristiti više aplikacija, čime se omogućuje višestruka upotreba koda. Velika prednost je i to što smanjuju mrežni promet, time što se preko mreže šalje samo ime pohranjene procedure i njeni parametri, a obrada grešaka se odvija na serveru.

### SINTAKSA:

```
CREATE PROCEDURE NazivProcedure
AS
SELECT *
FROM NazivTablice
```

## PARAMETRI POHRANJENE PROCEDURE

Parametri se proslijeđuju pohranjenoj procedure navođenjem neposredno nakon imena procedure. Maksimalni broj parametara koji mogu biti proslijeđeni pohranjenoj procedure je 2100. Parametar je definiran nazivom, te tipom podatka. Parametar pohranjene procedure se **OBAVEZNO** navodi s znakom @ kao prefiksom, te tipom podatka u nastavku. Vrijednost SVIH deklariranih parametara mora biti dana prilikom poziva procedure, ukoliko nije zadana pred definirana vrijednost u samoj deklaraciji procedure.

Uz deklaraciju parametara može se dodati i njihova default vrijednost koja će biti upisana ukoliko se vrijednost parametra ne proslijedi.

```
CREATE PROCEDURE NazivProcedure
(@parametar1 data_type = default_vrijednost,
@parametar2 data_type = default_vrijednost,
@parametar3 data_type = default_vrijednost)
AS
-- SQL procedure
```

```
CREATE PROCEDURE [FindEmployee] ( @name VARCHAR(32) )
AS
SELECT *
```

```
FROM Employees
WHERE FirstName like '%' + @name + '%' OR LastName like '%' + @name + '%'
```

Procedure mogu primiti i više parametara:

```
CREATE PROCEDURE InsertEmployee
(@fname NVARCHAR(10), @lname NVARCHAR(20), @title NVARCHAR(30))
AS
INSERT INTO Employees (FirstName, LastName, Title)
VALUES (@fname, @lname, @title)
RETURN @@IDENTITY
```

### IZMJENA POHRANJENE PROCEDURE

Ukoliko je postojeću pohranjenu procedure potrebno izmijeniti, jednostavno zamjenjujemo ključnu riječ **CREATE** sa ključnom riječju **ALTER**.

```
ALTER PROCEDURE InsertEmployee
(@fname NVARCHAR(10), @lname NVARCHAR(20), @title NVARCHAR(30), @country
NVARCHAR(15))
AS
INSERT INTO Employees (FirstName, LastName, Title, COUNTRY)
VALUES (@fname, @lname, @title, @country)
RETURN @@IDENTITY
```

### POKRETANJE POHRANJENE PROCEDURE

Pohranjena procedura se pokreće korištenjem EXECUTE or EXEC ključne riječi.

```
EXEC NazivProcedure
```

Ukoliko pohranjena procedura ima razmak u nazivu, naziv zatvaramo u navodnike.

```
EXEC "Naziv Procedure"
```

Ako pohranjena procedura prima parametre, njihove vrijednosti navodimo nakon naziva procedure:

```
EXEC InsertEmployee @fname='Ann', @lname='Scott', @title='Sales manager'

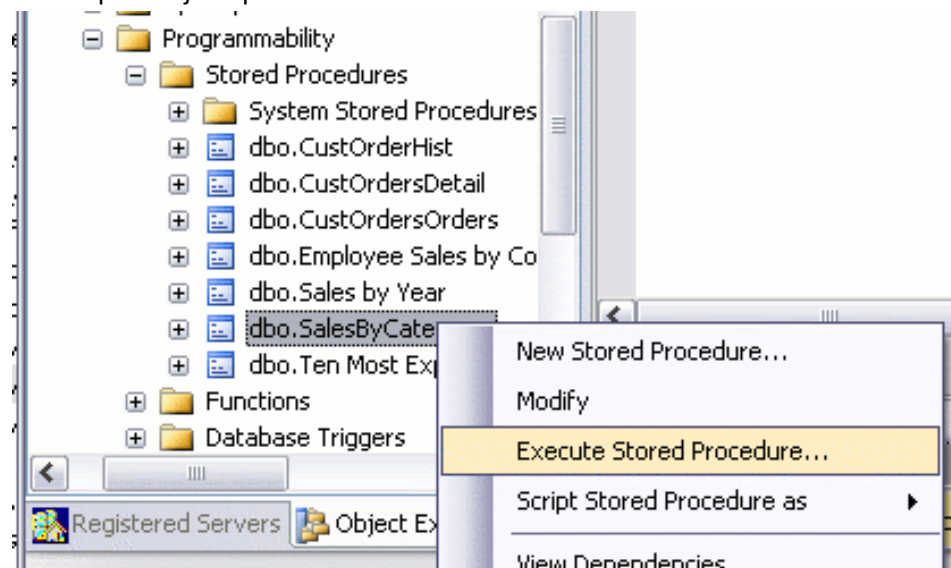
ILI

EXEC InsertEmployee 'Ann', 'Scott', 'Sales manager'
```

Pohranjene procedure se mogu pokretati i kroz GUI Microsoft SQL Server Management Studio.

Kako bi se pokrenula pohranjena procedura:

1. Odabir pohranjene procedure



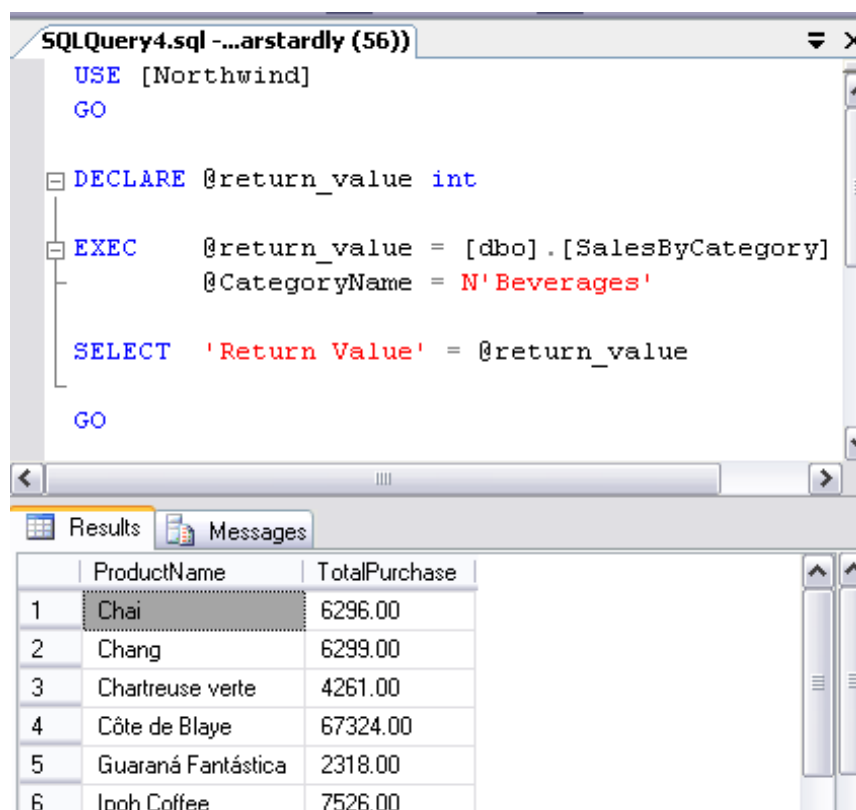
2. Desni klik na pohranjenu procedure i odabir "Execute Stored Procedure...":
3. Unos odabranih vrijednosti parametara:

**SalesByCategory**

Script Help

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@CategoryName	nvarchar(15)	No	<input type="checkbox"/>	Beverages
@OrdYear	nvarchar(4)	No	<input type="checkbox"/>	

4. Klik "OK"
5. SQL Server generira SQL skriptu i izvršava pohranjenu procedure:



The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery4.sql -...arstardly (56))". The query window contains the following T-SQL code:

```
USE [Northwind]
GO

DECLARE @return_value int

EXEC     @return_value = [dbo].[SalesByCategory]
        @CategoryName = N'Beverages'

SELECT  'Return Value' = @return_value
GO
```

Below the query window, the "Results" tab is active, displaying a table with two columns: "ProductName" and "TotalPurchase". The table contains six rows of data:

	ProductName	TotalPurchase
1	Chai	6296.00
2	Chang	6299.00
3	Chartreuse verte	4261.00
4	Côte de Blaye	67324.00
5	Guaraná Fantástica	2318.00
6	Ipoh Coffee	7526.00

## BRISANJE POHRANJENE PROCEDURE

Pohranjena procedura briše se izrazom **DROP PROCEDURE**:

```
DROP PROCEDURE InsertEmployee
```