**MINISTERUL EDUCAŢIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Programul de studii: Tehnologia informației**

RAPORT

# LUCRARE DE LABORATOR NR. 3

# la Baze de Date

A efectuat:

st. gr. TI-211 Popa Cătălin

A verificat: Olga Grosu

UTM, Chișinău 2023

**Proceduri stocate și funcții definite de utilizator**

Procedura stocată reprezintă un program, care se execută în interiorul unei baze de date și poate lua acțiuni complexe, folosind date definite de utilizator. Deoarece procedurile stocate sunt efectuate direct pe serverul bazei de date, acestea oferă o performanță mai rapidă decât, atunci, când aceleași operații sunt efectuate utilizând instrumentele-client a bazei de date. Procedura stocată combină interogările cu logica procedurală și este stocată într-o bază de date.

--crearea unei proceduri stocate ce returneaza tabela angajati si toate datele care persista in ea

--daca efectuam exec angajati\_procedure, se va returna datele din tabela

USE SistemGestionalBon

GO

DROP PROCEDURE IF EXISTS angajati\_procedure

GO

CREATE PROCEDURE angajati\_procedure

AS

BEGIN

SELECT \* FROM SistemGestionalBon.dbo.Angajati;

END;

go

--executarea procedurii angajati\_procedure

exec angajati\_procedure



FIgura 1 – Procedură stocată

Această procedura va prelua și va realiza interogarea care se află în corpul procedurii. În rezultat, dacă vom chema procedura, se va returna datele din tabela Angajati.

--procedura cu parametru care va afisa produsele ce au pretul mai mic decat parametrul introdus

USE SistemGestionalBon

GO

DROP PROCEDURE IF EXISTS produse\_pret\_redus

GO

CREATE PROCEDURE produse\_pret\_redus

@pret int

AS

BEGIN

SELECT \* FROM SistemGestionalBon.dbo.Produse WHERE PretProdus < @pret;

END;

go

--executarea procedurii date

exec produse\_pret\_redus @pret = 50;

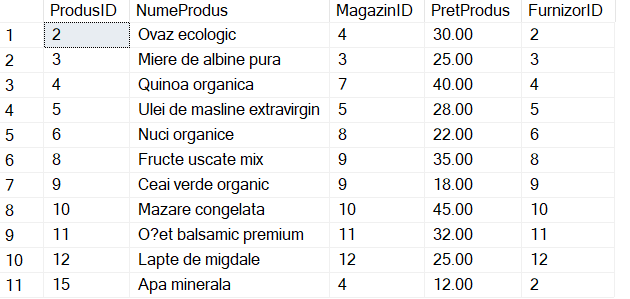


Figura 2 – Procedură cu parametri

Executarea acestei procedura va returna produsele din tabela Produse, care au prețul mai mic decât parametrul introdus. Parametru se introduce atunci când executăm procedura.

**Procedura cu parametri de ieșire**

--procedura data poseda parametri de iesire

--procedura afiseaza soldul final dupa scaderea tututor bonurilor din suma initiala

USE SistemGestionalBon

GO

DROP PROCEDURE IF EXISTS achitari\_bon

GO

CREATE PROCEDURE achitari\_bon

@SoldInitial INT,

@SoldFinal INT OUTPUT

AS

BEGIN

SET @SoldFinal = @SoldInitial - (SELECT SUM(SumaTotala) FROM SistemGestionalBon.dbo.Bon)

END

--pentru a executa procedura, este nevoie de introdus soldul initial si apoi executam pentru a vedea soldul ramas

DECLARE @SoldInitial INT = 10000; -- Valoarea initiala a soldului

DECLARE @SoldFinal INT; -- Variabila pentru rezultatul de iesire

EXEC achitari\_bon @SoldInitial, @SoldFinal OUTPUT;

PRINT 'Soldul final este: ' + CAST(@SoldFinal AS NVARCHAR(10));

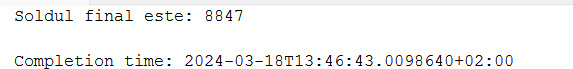


Figura 3 – Calcularea soldului final

Executarea acestei proceduri va returna soldul final, scăzând din soldul inițial, suma toatala de care a fost facute bonurile. Aici au fost folosită o procedură stocată cu parametri de ieșire.

**Instrucțiunea EXECUTE cu clauza SET WITH RESULTS SETS**

--WITH RESULT SETS: Aceasta clauza este utilizata pentru a specifica structura setului de rezultate intors de procedura stocata.

-- In cazul de fata, este utilizata pentru a defini explicit coloanele si

--tipurile de date ale setului de rezultate.

USE SistemGestionalBon

GO

EXEC produse\_pret\_redus @pret = 50

WITH RESULT SETS

(([Id] INT NOT NULL,

[Nume] CHAR(30) NOT NULL,

[Magazin] INT NOT NULL,

[Pret] INT NOT NULL,

[Furnizor] INT NOT NULL))

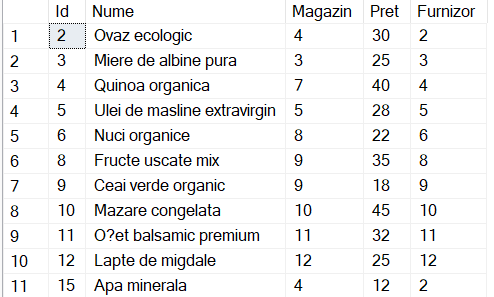


Figura 4 – Set with result

Astfel, se va returna rezultatul executiei procedurii pret\_redus, doar ca se va schimba denumirea coloanelor returnate.

**Functii**

--- TIP Scalar

--aceasta functie de mai jos va parsa datele din tabela Produse si va returna pretul produsului in baza la ID

USE SistemGestionalBon

GO

DROP FUNCTION IF EXISTS GetPrice

GO

CREATE FUNCTION GetPrice (@ProductId INT)

RETURNS INT

AS

BEGIN

DECLARE @Price INT;

SELECT @Price = PretProdus

FROM Produse

WHERE ProdusID = @ProductId;

RETURN @Price;

END;

--executarea functiei

SELECT dbo.GetPrice(5); -- Exemplu: obbine pretul produsului cu ID-ul 5



Figura 5 – Pretul produsului

Aceasta functie poate fi folosita pentru a obtine pretul unui anumit produs.

--- TIP tabel

-- aceasta functie returneaza toate produsele dintr-un magazin anumit

CREATE FUNCTION dbo.GetProductsByStore (@StoreId INT)

RETURNS TABLE

AS

RETURN

(

SELECT ProdusID, NumeProdus, MagazinID, PretProdus, FurnizorID

FROM Produse

WHERE MagazinID = @StoreId

);

--apelarea functiei

SELECT \* FROM dbo.GetProductsByStore(8); -- obtine toate produsele din magazinul cu ID-ul 8

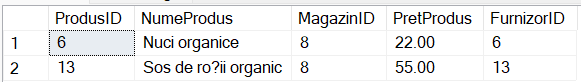


Figura 6 – Produsele dintr-un anumit magazin

Aceasta functie poate fi folosita pentru a obtine toate produsele prezente intr-un anumit magazin cu un id specific lui.

--- TIP tabel multi-instructiune

CREATE FUNCTION GetProductsByStoreAndSort (@StoreId INT)

RETURNS @ProductTable TABLE

(

Id INT,

Nume NVARCHAR(100),

Pret DECIMAL(10, 2)

)

AS

BEGIN

INSERT INTO @ProductTable (Id, Nume, Pret)

SELECT ProdusId,NumeProdus,PretProdus

FROM Produse

WHERE MagazinID = @StoreId

ORDER BY PretProdus;

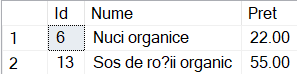
RETURN;

END;

--executaarea functiei

DECLARE @StoreId INT = 8; -- ID-ul magazinului dorit

SELECT \* FROM GetProductsByStoreAndSort(@StoreId);

  
Figura 7 – Functie de tip tabel

Această funcție va returna toate produsele din magazinul cu ID-ul specificat și le va sorta în ordine crescătoare după preț.

**Declanșatoarele DML**

Declanșatoarele DML (Data Manipulation Language) sunt obiecte în bazele de date care sunt activate automat atunci când sunt efectuate anumite operațiuni DML asupra unei tabele. DML-ul include operațiuni de manipulare a datelor, cum ar fi inserarea (INSERT), actualizarea (UPDATE) și ștergerea (DELETE) într-o tabelă.

--in continuare am creat un declansator care afiseaza un mesaj de informare cand se inserare date noi in tabela

USE SistemGestionalBon

GO

DROP TRIGGER IF EXISTS trig\_Exemplu\_Insert

GO

-- Cream declansatorul DML

CREATE TRIGGER trig\_Exemplu\_Insert

ON Furnizori

AFTER INSERT

AS

BEGIN

-- Afisam un mesaj la consola de mesaje

PRINT 'O noua inregistrare a fost adaugata in tabela Furnizori.';

END;

insert into Furnizori(FurnizorID, Denumire, Adresa)

VALUES (16, 'Srl sapr', 'Decebal 1212')

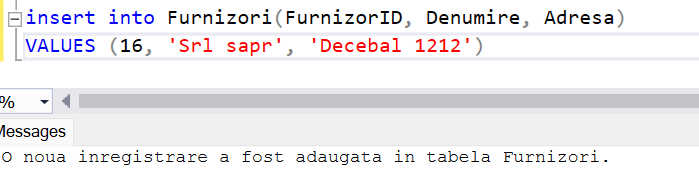


Figura 8 – Afisare mesaj de informare dupa inserarea datelor

Acest declansator va informa mereu utilizatorul atunci cand a inserat date noi in tabela.

**Declanșatoarele DDL**

--Acest declansator va fi declansat atunci când se creeaza sau se elimină o tabelă în baza de date.

--El utilizează funcția EVENTDATA() pentru a accesa informatii despre evenimentul DDL care a declanșat

--declanșatorul, precum tipul evenimentului și numele obiectului afectat. Apoi, un mesaj corespunzător

--este afișat utilizând instrucțiunea PRINT.

USE SistemGestionalBon

GO

DROP TRIGGER IF EXISTS trig\_DDL\_Table;

GO

CREATE TRIGGER trig\_DDL\_Table

ON DATABASE

FOR CREATE\_TABLE, DROP\_TABLE

AS

BEGIN

DECLARE @EventType NVARCHAR(100);

DECLARE @ObjectName NVARCHAR(255);

SET @EventType = EVENTDATA().value('(/EVENT\_INSTANCE/EventType)[1]', 'nvarchar(100)');

SET @ObjectName = EVENTDATA().value('(/EVENT\_INSTANCE/ObjectName)[1]', 'nvarchar(255)');

-- Verificăm dacă evenimentul este pentru crearea unei tabele

IF @EventType = 'CREATE\_TABLE'

BEGIN

PRINT 'O noua tabela a fost creata: ' + @ObjectName;

END

-- Verificăm dacă evenimentul este pentru eliminarea unei tabele

ELSE IF @EventType = 'DROP\_TABLE'

BEGIN

PRINT 'O tabela a fost eliminata: ' + @ObjectName;

END

END;

go

drop table if exists test

go

create table test(

dasa int)

go

drop table test

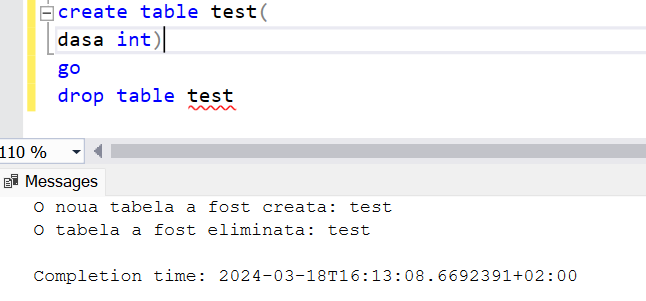


Figura 9 – Declansator DDL

Acest declanșator va fi declanșat atunci când se creează sau se elimină o tabelă în baza de date. El utilizează funcția EVENTDATA() pentru a accesa informații despre evenimentul DDL care a declanșat declanșatorul, precum tipul evenimentului și numele obiectului afectat. Apoi, un mesaj corespunzător este afișat utilizând instrucțiunea PRINT.

**Declanșatoarele LOGON**

----LOGON

-- Cream declanșatorul LOGON

--Acest declanșator este declanșat atunci când un utilizator se autentifică pe serverul SQL

--Server și afișează un mesaj cu numele utilizatorului care s-a autentificat și data și ora autentificării.

USE SistemGestionalBon

GO

DROP TRIGGER IF EXISTS trig\_Logon;

GO

CREATE TRIGGER trig\_Logon

ON ALL SERVER WITH EXECUTE AS 'sa'

FOR LOGON

AS

BEGIN

DECLARE @LoginName NVARCHAR(100);

SET @LoginName = ORIGINAL\_LOGIN();

PRINT 'Utilizatorul ' + @LoginName + ' s-a autentificat pe server la data si ora ' + CONVERT(NVARCHAR(30), GETDATE(), 121);

END;

GO



Figura 10 - Logon

Acest declanșator este declanșat atunci când un utilizator se autentifică pe serverul SQL Server și afișează un mesaj cu numele utilizatorului care s-a autentificat și data și ora autentificării. Este important să acordați atenție faptului că acest declanșator utilizează privilegiile EXECUTE AS 'sa', ceea ce înseamnă că va fi executat cu privilegiile utilizatorului sa. Acest lucru este necesar pentru a obține informațiile despre utilizatorul autentificat folosind funcția ORIGINAL\_LOGIN().

**Concluzie**

Acest laborator prezintă exemple de proceduri stocate, funcții și declanșatoare în SQL Server. Procedurile stocate sunt create pentru a accesa și modifica datele din baza de date, în timp ce funcțiile sunt utilizate pentru a returna valori sau seturi de date filtrate. Declanșatoarele DDL și DML gestionează evenimentele de modificare a structurii obiectelor și a datelor în baza de date. În plus, este prezentat un declanșator de logon pentru monitorizarea autentificărilor utilizatorilor pe serverul SQL. Aceste instrumente oferă flexibilitate și control în gestionarea datelor și a activității în baza de date, contribuind la dezvoltarea și gestionarea eficientă a aplicațiilor bazate pe SQL Server.