Lab 5.
Limbajul Transact SQL

1. Obiective

- Variabile locale. Instructiunile T-SQL
- Interogari parametrizate
- Cursoare. Proceduri stocate. Vederi

2. Mersul lucrarii

2.1. Comenzi SQL cu parametri

Se creează și se execută un script cu următorul conținut:

```
USE Firma
GO

DECLARE @n int;
DECLARE @SQLString nvarchar(200), @ParmDefinition nvarchar(200);
SET @SQLString = 'SELECT IdAngajat, Nume, Prenume
FROM Angajati WHERE IdSectie = @SectieID';
SET @ParmDefinition = '@SectieID int';

-- Executie cu o prima valoare a parametrului
SET @n = 1;
EXECUTE sp_executesql @SQLString, @ParmDefinition, @SectieId = @n;

-- Executie cu o alta valoare a parametrului
SET @n = 2;
EXECUTE sp_executesql @SQLString, @ParmDefinition, @SectieId = @n;
```

Se creează și se execută un script cu următorul conținut:

```
USE Firma
GO

DECLARE @den varchar(24);
DECLARE @SQLString nvarchar(200), @ParmDefinition nvarchar(200);
SET @SQLString = 'SELECT a.nume, a.prenume FROM Angajati a, Sectii s
WHERE a.IdSectie=s.IdSectie AND s.Denumire = @sectie';
SET @ParmDefinition = '@Sectie varchar(24)';
SET @den = 'TESA';

--Executie folosind sp_executesql
EXECUTE sp_executesql @SQLString, @ParmDefinition, @Sectie = @den;
--Executie folosind sp_executesql
EXECUTE sp_executesql @SQLString, @ParmDefinition, @Sectie = 'Productie';
```

Scrieți un script pentru a obține angajații ce au o anumită funcție dată ca parametru.
Scrieți un script pentru a obține produsele vândute de un angajat pentru care se dau numele și prenumele, folosind *sp_executesql*.

2.2. Lucrul cu cursoare

Se creaza si se executa un script cu urmatorul continut:

```
/* Cursor ce afişează denumirea produselor și prețul mediu ponderat */
```

```
USE Firma
GO

DECLARE @id_produs int, @produs char(20), @pmp decimal(7,2)

DECLARE cursor_produse CURSOR FOR
	select p.idprodus, p.denumire produs
	from produse p join vanzari v on v.IDProdus=p.IdProdus
	group by p.idprodus, p.denumire
	having count(v.idprodus) > 1
OPEN cursor_produse

FETCH cursor_produse INTO @id_produs, @produs
WHILE (@@FETCH_STATUS = 0)
BEGIN
	select @pmp = CAST(sum(v.NrProduse*v.PretVanz) as decimal) / sum(v.NrProduse)
		from vanzari v join produse p on v.IDProdus=p.IdProdus
		where v.idprodus = @id_produs
	PRINT @produs + '   ' + CAST(@pmp as varchar)
	FETCH cursor_produse INTO @id_produs, @produs
END

CLOSE cursor_produse
DEALLOCATE cursor_produse
GO
```

Scrieti un program T-SQL care:

- creaza un cursor ce contine produsele vandute (denumire, numar, ...)
- pentru fiecare produs din cursor afiseaza angajatii care au facut vanzarea

2.3. Se creaza si testeaza proceduri stocate folosind urmatoarele modele

2.3.1. Procedura stocata pentru inserare date in tabela Sectii

```
/*	Procedura:	usp_Insert_Sectii
	Sintaxa:	exec usp_Insert_Sectii @Denumire=Denumire */
USE Firma
GO
DROP PROCEDURE IF EXISTS usp_Insert_Sectii
GO

CREATE procedure usp_Insert_Sectii (
	@Denumire varchar(20)
)
as
	set quoted_identifier off
	set NOCOUNT ON

	begin try
		INSERT INTO [Sectii] (
			[Denumire]
		)
		VALUES (
			@Denumire
```

```
        )
    end try
    begin catch
        DECLARE @ErrorMessage NVARCHAR(1000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;
        SELECT @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState );
    end catch
GO

-- Apel procedura (in alta fereastra Query)
exec usp_Insert_Sectii @Denumire='Productie - 2'
exec usp_Insert_Sectii @Denumire='Proiectare - 3'
```

2.3.2. Procedura stocata pentru actualizare date in tabela Functii

```
/*  Procedura:  usp_Update_Functii
    Sintaxa: exec usp_Update_Functii @IdFunctie=IdFunctie,
        @Denumire=Denumire, @Salariu=Salariu*/
USE Firma
GO
DROP PROCEDURE IF EXISTS usp_Update_Functii
go

CREATE procedure usp_Update_Functii(
    @IdFunctie int, @Denumire varchar(50) = NULL, @Salariu int = NULL
)
as
    set quoted_identifier off
    set NOCOUNT ON

    begin try
        UPDATE [Functii] SET
            [Denumire] = COALESCE(@Denumire, [Denumire]),
            [Salariu] = COALESCE(@Salariu, [Salariu])
        WHERE [IdFunctie] = @IdFunctie
    end try
    begin catch
        DECLARE @ErrorMessage NVARCHAR(1000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;
        SELECT @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState );
    end catch
go

-- Apel procedura (in alta fereastra Query)
exec usp_Update_Functiii @IdFunctie=3, @Salariu = 2000
exec usp_Update_Functii @IdFunctie=1, @Denumire='Super Manager'
```

- Sa se scrie si testeze proceduri stocate similare pentru inserare si actualizare in tabelele Functii si Angajati (usp_Insert_Functii, usp_Insert_Angajati, usp_Update_Angajati).
- Sa se scrie si testeze proceduri stocate pentru a actualiza tabelele pe baza a altor criterii (exemplu: modificare date despre o sectie pe baza denumirii, modificare date despre o functie pe baza denumirii, etc.)

2.3.3. Procedura stocata pentru inserare date in tabela Angajati folosind denumirea unei sectii si denumirea unei functii

```
-- Procedura se bazeaza pe existenta PS usp_Insert_Angajati, ce insereaza un
-- angajat avand toate datele necesare.
-- Daca nu exista, trebuie creata in  prealabil !

/*      Procedura:      usp_Insert_Angajati_Sectie_Functie
        Sintaxa:        exec usp_Insert_Angajati_Sectie_Functie
                                @Sectie=Sectie,
                                @Functie=Functie,
                                @Nume=Nume,
                                @Prenume=Prenume,
                                @Marca=Marca,
                                @DataNasterii=DataNasterii,
                                @DataAngajarii=DataAngajarii,
                                @Adresa_jud=Adresa_jud    */
USE Firma
GO
DROP PROCEDURE IF EXISTS usp_Insert_Angajati_Sectie_Functie
Go

CREATE procedure usp_Insert_Angajati_Sectie_Functie (
        @Sectie varchar (20),
        @Functie varchar (20),
        @Nume varchar(20),
        @Prenume varchar(20),
        @Marca int,
        @DataNasterii date = NULL,
        @DataAngajarii date = NULL,
        @Adresa_jud varchar(50) = NULL
)
AS
        set quoted_identifier off
        set NOCOUNT ON

        declare @SectieId int, @FunctieId int

        begin try
                select @SectieId=IdSectie from Sectii where Denumire=@Sectie
                -- testati PS in cele doua variante: "= NULL" si "IS NULL"
                -- IF @SectieID = NULL
                IF @SectieID IS NULL
                BEGIN
                        RAISERROR ('Sectie inexistenta', 11, 1)
                        RETURN
                END
```

```
        select @FunctieId=IdFunctie from Functii where Denumire =@Functie
        -- IF @FunctieID = NULL
        IF @FunctieID IS NULL
        BEGIN
            RAISERROR ('Functie inexistenta', 11, 1)
            RETURN
        END

        -- apeleaza o alta PS existenta !
        exec usp_Insert_Angajati
            @Nume = @Nume,
            @Prenume = @Prenume,
            @DataNasterii = @DataNasterii,
            @DataAngajarii = @DataAngajarii,
            @Adresa_jud = @Adresa_jud,
            @Marca = @Marca,
            @IdSectie = @SectieId,
            @IdFunctie = @FunctieId
    end try
    begin catch
        DECLARE @ErrorMessage NVARCHAR(1000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;
        SELECT @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState );
    end catch
go

-- Apel procedura:
exec usp_Insert_Angajati_Sectie_Functie
    @Sectie = 'Productie',
    @Functie = 'Inginer',
    @Nume = 'protopopescu',
    @Prenume = 'izbavitu',
    @Marca = 1001
```

Sa se faca validari in procedura stocata anterioara pentru DataAngajarii (de exemplu sa fie ulterioara datei de nastere).
Sa se creeze proceduri stocate pentru inserarea de date in alte tabele pe baza unor denumiri sau nume.
Sa se creeze o procedura stocata pentru actualizarea tabelei Angajati pe baza numelui si prenumelui respectiv pe baza marcii.

2.4. Se creaza si testeaza vederi (views) folosind urmatoarele modele

2.4.1. Simplificarea unor interogari prin folosirea unor vederi

Vedere care contine clientii ce au cumparat produse:

```
USE Firma
GO
```

```
CREATE VIEW vClientiProduse AS
    SELECT DISTINCT
        c.denumire den_client,
        p.denumire produs,
        v.datavanz,
        v.pretvanz,
        v.nrproduse
    FROM vanzari v, clienti c, produse p
    WHERE (v.idprodus=p.idprodus) AND (v.idclient=c.idclient)
GO
```

Vedere ce contine angajatii si totalul produselor de acelasi tip vandute de acestia:

```
USE Firma
GO

CREATE VIEW vAngajatiProduse AS
    SELECT DISTINCT
        a.idangajat,
        a.nume,
        a.prenume,
        p.denumire produs,
        SUM(v.nrproduse) NrBucati
    FROM angajati a, vanzari v, produse p
    WHERE (v.idprodus=p.idprodus) AND (a.idangajat=v.idvanzator)
    GROUP BY a.idangajat, a.nume, a.prenume, p.denumire
GO
```

Folosind aceste vederi ca sursa de date, sa se raspunda (cu cate o interogare) la urmatoarele intrebari (daca este cazul, introduceti date suplimentare in tabelele de baza pentru a avea rezultate):

- Care sunt clientii ce au cumparat cel putin un produs ?
- Care sunt clientii ce au cumparat mai mult de un produs de acelasi tip ?
- Care este media vanzarilor pe o anumita perioada de timp ?
- Care este cea mai mare vanzare pe o anumita perioada de timp ?
- Care este numarul total de produse vandute pe o anumita perioada de timp ?
- Care sunt angajatii care au vandut cele mai multe produse ?
- Care sunt angajatii care au vandut cele mai putine produse ?
- Care sunt angajatii care au vandut mai mult de un produs ?
- Care sunt angajatii care nu au vandut niciun produs ?

In toate cazurile interogarea folosita are clauza *FROM vClientiProduse* sau *FROM vAngajatiProduse.*

2.4.2. Formatarea datelor

```
USE Firma
GO

CREATE VIEW vAngajati AS
    SELECT
        s.Denumire Sectie, RTRIM(a.nume + ' ' + a.prenume) Angajat,
        f.Denumire Functie,
```

```
        DATEDIFF(year, a.datanasterii, GETDATE()) Varsta,
        DATEDIFF(year, a.dataangajarii, GETDATE()) Vechime
    FROM angajati a, sectii s, functii f
    WHERE a.IdSectie = s.IdSectie AND a.IdFunctie = f.IdFunctie
GO
```

Folosind aceasta vedere ca sursa de date, sa se raspunda (cu cate o iterogare) la intrebarile urmatoare:

- Cati angajati au depasit varsta de 50 de ani, in firma si apoi intr-o anumita sectie (data prin denumire) ?
- Care este media de varsta a angajatilor pe sectii ?
- Cati angajati au depasit vechimea de 10 de ani, in firma si apoi intr-o anumita sectie (data prin denumire) ?
- Care este media vechimii angajatilor pe sectii ?