**Databases Laboratory Work Nr 6**

**Title:** Tables and indices creation

**Prerequisites:** SQL Server 2019 and SSMS

**Objectives:** Get practical experience in creation of tables and indices.

**Tasks:**

1. Sa se scrie o instructiune T-SQL, care ar popula co Joana *Adresa* \_ *Postala* \_ *Profesor* din tabelul

*profesori* cu valoarea *'mun. Chisinau',* unde adresa este necunoscuta.

2. Sa se modifice schema tabelului *grupe,* ca sa corespunda urmatoarelor cerinte:

a) Campul *Cod\_ Grupa* sa accepte numai valorile unice ~i sa nu accepte valori necunoscute.

b) Sa se tina cont ca cheie primarii, deja, este definitii asupra coloanei *Id\_ Grupa.*

3. La tabelul *grupe,* sa se adauge 2 coloane noi *Sef\_grupa* ~i *Prof\_Indrumator,* ambele de tip

*INT.* Sii se populeze campurile nou-create cu cele mai potrivite candidaturi ill baza criteriilor

de maijos:

a) *$eful grupei* trebuie sa aiba cea mai buna reu~itii (medie) din grupa la toate formele de

evaluare ~i la toate disciplinele. Un student nu poate fi ~ef de grupa la mai multe grupe.

b) *Profesorul fndrumator* trebuie sa predea un numiir maximal posibil de discipline la grupa

data. Daca nu existii o singurii candidaturii, care corespunde primei cerinte, atunci este

ales din grupul de candidati acel cu identificatorul *(Id\_Profesor)* minimal. Un profesor nu

poate fi illdrumator la mai multe grupe.

c) Sii se scrie instructiunile *ALTER, SELECT, UPDATE* necesare pentru crearea coloanelor

in tabelul *grupe,* pentru selectarea candidatilor ~i inserarea datelor . .

4. Sa se scrie o instructiune T-SQL, care ar mari toate notele de evaluare ~efilor de grupe cu un

punct. Nota maximala (10) nu poate fi miirita.

5. Sa se creeze un tabel *profesori\_new,* care include urmatoarele coloane: *Id\_Profesor,*

*Nume* \_ *Profesor, Prenume* \_ *Profesor, Localitate, Adresa* \_ *1, Adresa* \_ *2.*

a) Coloana *Id\_Profesor* trebuie sa fie definita drept cheie primara ~i, ill baza ei, sa fie

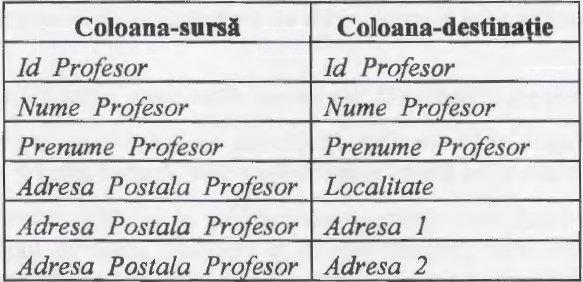
construit un index *CLUSTERED.*

b) Campul *Localitate* trebuie sa posede proprietatea *DEF A ULT= 'mun. Chisinau'.*

c) Sa se insereze toate datele din tabelul *profesori* ill tabelul *profesori\_new.* Sa se scrie, cu

acest scop, un numiir potrivit de instructiuni T-SQL. Datele trebuie sa fie transferate in

felul urmator:



in coloana *Localitate* sii fie inserata doar informatia despre denumirea localitiitii din

coloana-sursa *Adresa\_Postala\_Profesor.* in coloana *Adresa\_l,* doar denumirea striizii. in

*coloanaAdresa\_2,* sii se piistreze numiirul casei ~i (posibil) a apartamentului.

6. Sa se insereze datele in tabelul *orarul* pentru *Grupa= 'CIBJ 71' (Id\_ Grupa=* 1) pentru ziua de

luni. Toate lectiile vor avea loc ill blocul de studii *'B'.* Mai jos, sunt prezentate detaliile de

inserare:

*(ld\_Disciplina* = *107, Id\_Profesor= 101, Ora ='08:00', Auditoriu* = *202);*

*(Id\_Disciplina* = *108, Id\_Profesor= 101, Ora ='11:30', Auditoriu* = *501);*

*(ld\_Disciplina* = *119, Id\_Profesor= 117, Ora ='13:00', Auditoriu* = *501);*

7. Sii se scrie expresiile T-SQL necesare pentru a popula tabelul *orarul* pentru grupa INFl 71 ,

ziua de luni. Datele necesare pentru inserare trebuie sa fie colectate cu ajutorul instructiunii/instructiunilor

*SELECT* ~i introduse in tabelul-destinatie, ~tiind ca:

lectie #1 *(Ora* ='08:00', *Disciplina* = *'Structuri de date* si *algoritmi', Profesor ='Bivol Ion')*

lectie #2 *(Ora* ='11 :30', *Disciplina* = *'Programe aplicative', Profesor ='Mircea Sorin')*

lectie #3 *(Ora* ='13:00', *Disciplina ='Baze de date', Profesor* = *'Micu Elena')*

8. Sa se scrie interogarile de creare a indecilor asupra tabelelor din baza de date *universitatea*

pentru a asigura o performanta sporita la executarea interogarilor *SELECT* din Lucrarea

practica 4. Rezultatele optimizarii sa fie analizate in baza planurilor de executie, pana la ~i

dupa crearea indecilor.

Indecii nou-creati sa fie plasati fizic in grupul de fiiere *userdatafgroupl (Crearea $i*

*intrefinerea bazei de date* - sectiunea 2.2.2)

**Implementation**

USE universitatea;

/\* Task 1 \*/

BEGIN

UPDATE profesori

SET Adresa\_Postala\_Profesor = 'mun. Chisinau'

WHERE Adresa\_Postala\_Profesor IS NULL

SELECT \* FROM profesori

END

/\* Task 2 \*/

BEGIN

ALTER TABLE grupe

ALTER COLUMN Cod\_Grupa VARCHAR(255) NOT NULL

ALTER TABLE grupe

ADD CONSTRAINT UQ\_Cod\_Grupa UNIQUE (Cod\_Grupa)

ALTER TABLE grupe

ADD CONSTRAINT PK\_grupe PRIMARY KEY (Id\_Grupa)

SELECT \* FROM grupe

END

/\* Task 3 \*/

BEGIN

ALTER TABLE grupe

ADD Sef\_grupa INT

ALTER TABLE grupe

ADD CONSTRAINT UQ\_Sef\_Grupa UNIQUE (Sef\_Grupa)

ALTER TABLE grupe

ADD Prof\_Indrumator INT

UPDATE grupe

SET Sef\_grupa = (SELECT TOP(1) Id\_Student, Id\_Grupa FROM studenti\_reusita

WHERE studenti\_reusita.Id\_Grupa = grupe.Id\_Grupa

GROUP BY Id\_Student

ORDER BY SUM(Nota) DESC)

UPDATE grupe

SET Prof\_Indrumator = (SELECT TOP(1) Id\_Profesor FROM (SELECT TOP(1000) Id\_Profesor FROM studenti\_reusita

WHERE studenti\_reusita.Id\_Grupa = grupe.Id\_Grupa

GROUP BY Id\_Profesor, Id\_Grupa

ORDER BY COUNT(DISTINCT Id\_Disciplina) DESC, Id\_Profesor DESC) AS temp)

SELECT \* FROM grupe

END

/\* Task 4 \*/

BEGIN

UPDATE studenti\_reusita

SET Nota = Nota + 1

WHERE Nota < 10 AND Id\_Student IN (SELECT Sef\_grupa FROM grupe)

END

/\* Task 5 \*/

GO

CREATE FUNCTION dbo.GetEnd (@STRING AS VARCHAR(255), @SEGMENT AS INT)

RETURNS INT

BEGIN

DECLARE @END INT

IF @SEGMENT = 1

BEGIN

IF (CHARINDEX(', ', @STRING) != 0)

SET @END = CHARINDEX(', ', @STRING)

ELSE

SET @END = LEN(@STRING) + 1

END

ELSE IF @SEGMENT = 2

BEGIN

IF (CHARINDEX(', ', @STRING) != 0)

SET @END = CHARINDEX(', ', @STRING, CHARINDEX(', ', @STRING) + 2) - CHARINDEX(', ', @STRING) - 2

ELSE

SET @END = 0

END

ELSE IF @SEGMENT = 3

BEGIN

IF (CHARINDEX(', ', @STRING) != 0)

SET @END = LEN(@STRING) - CHARINDEX(', ', @STRING, CHARINDEX(', ', @STRING) + 2) - 1

ELSE

SET @END = 0

END

RETURN @END;

END

GO

CREATE FUNCTION dbo.GetStart (@STRING AS VARCHAR(255), @SEGMENT AS INT)

RETURNS INT

BEGIN

DECLARE @START INT

IF @SEGMENT = 1

SET @START = 0

ELSE IF @SEGMENT = 2

BEGIN

IF (CHARINDEX(', ', @STRING) != 0)

SET @START = CHARINDEX(', ', @STRING) + 2

ELSE

SET @START = 0

END

ELSE IF @SEGMENT = 3

BEGIN

IF (CHARINDEX(', ', @STRING) != 0)

SET @START = CHARINDEX(', ', @STRING, CHARINDEX(', ', @STRING) + 1) + 2

ELSE

SET @START = 0

END

RETURN @START;

END

GO

-- delete function

DROP FUNCTION dbo.GetEnd;

DROP FUNCTION dbo.GetStart;

GO

BEGIN

CREATE TABLE profesori\_new (

Id\_Profesor INT,

Nume\_Profesor VARCHAR,

Prenume\_Profesor VARCHAR,

Localitate VARCHAR DEFAULT('Mun. Chisinau'),

Adresa\_1 VARCHAR,

Adresa\_2 VARCHAR,

)

CREATE CLUSTERED INDEX profesori\_new\_PK\_index

ON profesori\_new(Id\_Profesor)

SELECT

Id\_Profesor,

Nume\_Profesor,

Prenume\_Profesor,

SUBSTRING(Adresa\_Postala\_Profesor, dbo.GetStart(Adresa\_Postala\_Profesor, 1), dbo.GetEnd(Adresa\_Postala\_Profesor, 1)) as Localitate,

SUBSTRING(Adresa\_Postala\_Profesor, dbo.GetStart(Adresa\_Postala\_Profesor, 2), dbo.GetEnd(Adresa\_Postala\_Profesor, 2)) as Adresa\_1,

SUBSTRING(Adresa\_Postala\_Profesor, dbo.GetStart(Adresa\_Postala\_Profesor, 3), dbo.GetEnd(Adresa\_Postala\_Profesor, 3)) as Adresa\_2

INTO profesori\_new

FROM profesori

select \* from profesori\_new

END

/\* Task 6 \*/

GO

DROP TABLE IF EXISTS orarul;

CREATE TABLE orarul (

Id\_Disciplina INT,

Id\_Profesor INT,

Id\_Grupa INT,

Zi VARCHAR(10),

Ora TIME,

Auditoriu INT,

Bloc VARCHAR)

GO

INSERT INTO orarul VALUES

(107, 101, 1, 'Luni', '08:00', 202, 'B'),

(108, 101, 1, 'Luni', '11:30', 501, 'B'),

(119, 117, 1, 'Luni', '13:00', 501, 'B')

SELECT \* FROM orarul

GO

/\* Task 7 \*/

INSERT INTO orarul

SELECT Id\_Disciplina,

(SELECT Id\_Profesor FROM profesori WHERE Nume\_Profesor = 'Bivol' AND Prenume\_Profesor = 'Ion') AS Id\_Profesor,

(SELECT TOP(1) Id\_Grupa FROM grupe WHERE Cod\_Grupa = 'INF171') AS Id\_Grupa,

'Luni' as Zi, '08:00' as Ora, 202 as Auditoriu, 'B' as Bloc

FROM discipline WHERE Disciplina = 'Structuri de date si algoritmi';

INSERT INTO orarul

SELECT Id\_Disciplina,

(SELECT Id\_Profesor FROM profesori WHERE Nume\_Profesor = 'Mircea' AND Prenume\_Profesor = 'Sorin') AS Id\_Profesor,

(SELECT TOP(1) Id\_Grupa FROM grupe WHERE Cod\_Grupa = 'INF171') AS Id\_Grupa,

'Luni' as Zi, '11:30' as Ora, 202 as Auditoriu, 'B' as Bloc

FROM discipline WHERE Disciplina = 'Programe aplicative';

INSERT INTO orarul

SELECT Id\_Disciplina,

(SELECT Id\_Profesor FROM profesori WHERE Nume\_Profesor = 'Micu' AND Prenume\_Profesor = 'Elena') AS Id\_Profesor,

(SELECT TOP(1) Id\_Grupa FROM grupe WHERE Cod\_Grupa = 'INF171') AS Id\_Grupa,

'Luni' as Zi, '13:00' as Ora, 202 as Auditoriu, 'B' as Bloc

FROM discipline WHERE Disciplina = 'Baze de date';

SELECT \* FROM orarul

GO

/\* Task 8 \*/

CREATE NONCLUSTERED COLUMNSTORE INDEX non\_clust\_discpline

ON studenti(Id\_Student, Nume\_Student, Prenume\_Student)

WITH (DATA\_COMPRESSION=COLUMNSTORE) ON userdatafgroup01

CREATE NONCLUSTERED COLUMNSTORE INDEX non\_clust\_discpline

ON studenti\_reusita(Id\_Student, Nota)

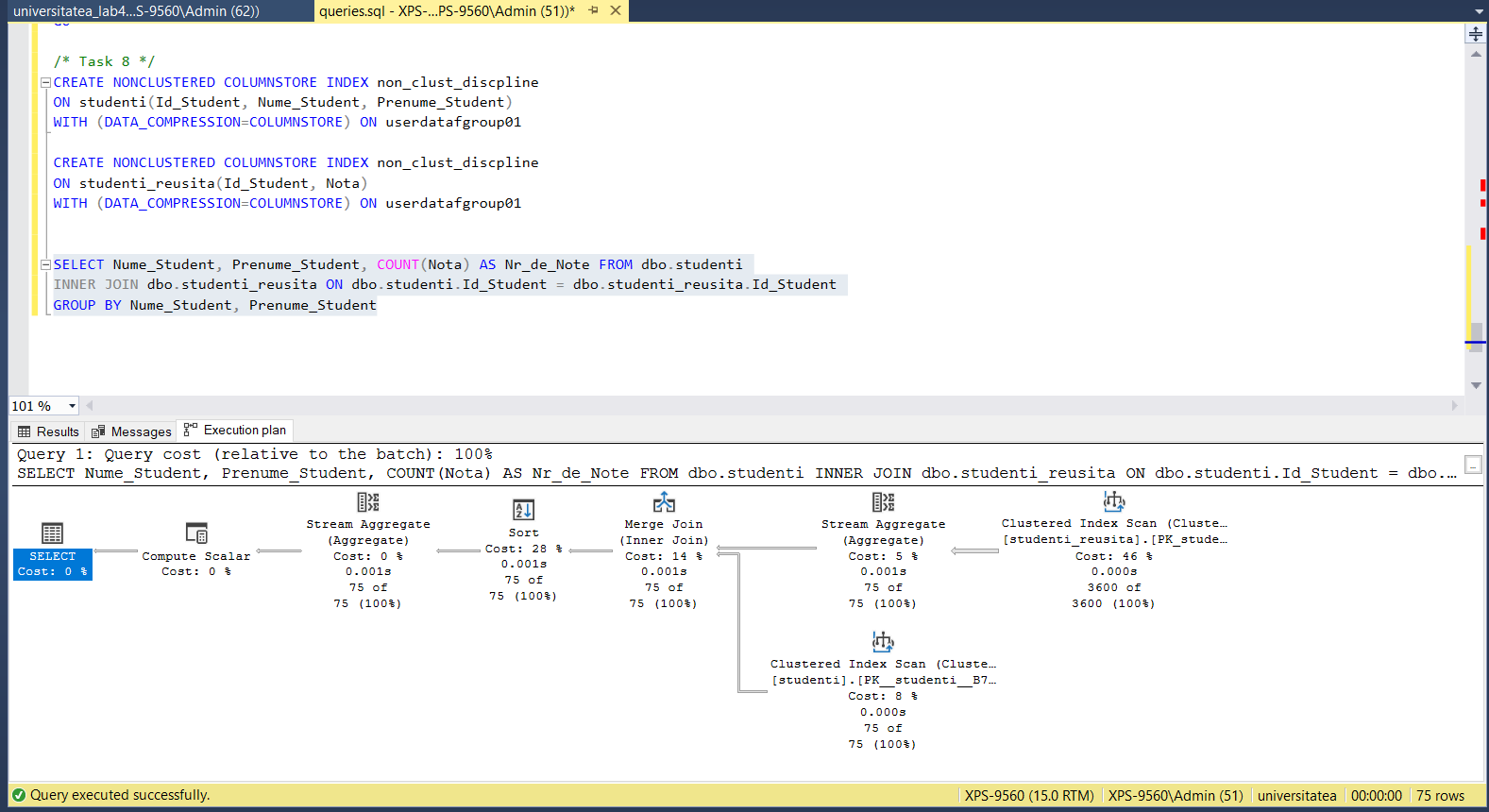
WITH (DATA\_COMPRESSION=COLUMNSTORE) ON userdatafgroup01

SELECT Nume\_Student, Prenume\_Student, COUNT(Nota) AS Nr\_de\_Note FROM dbo.studenti

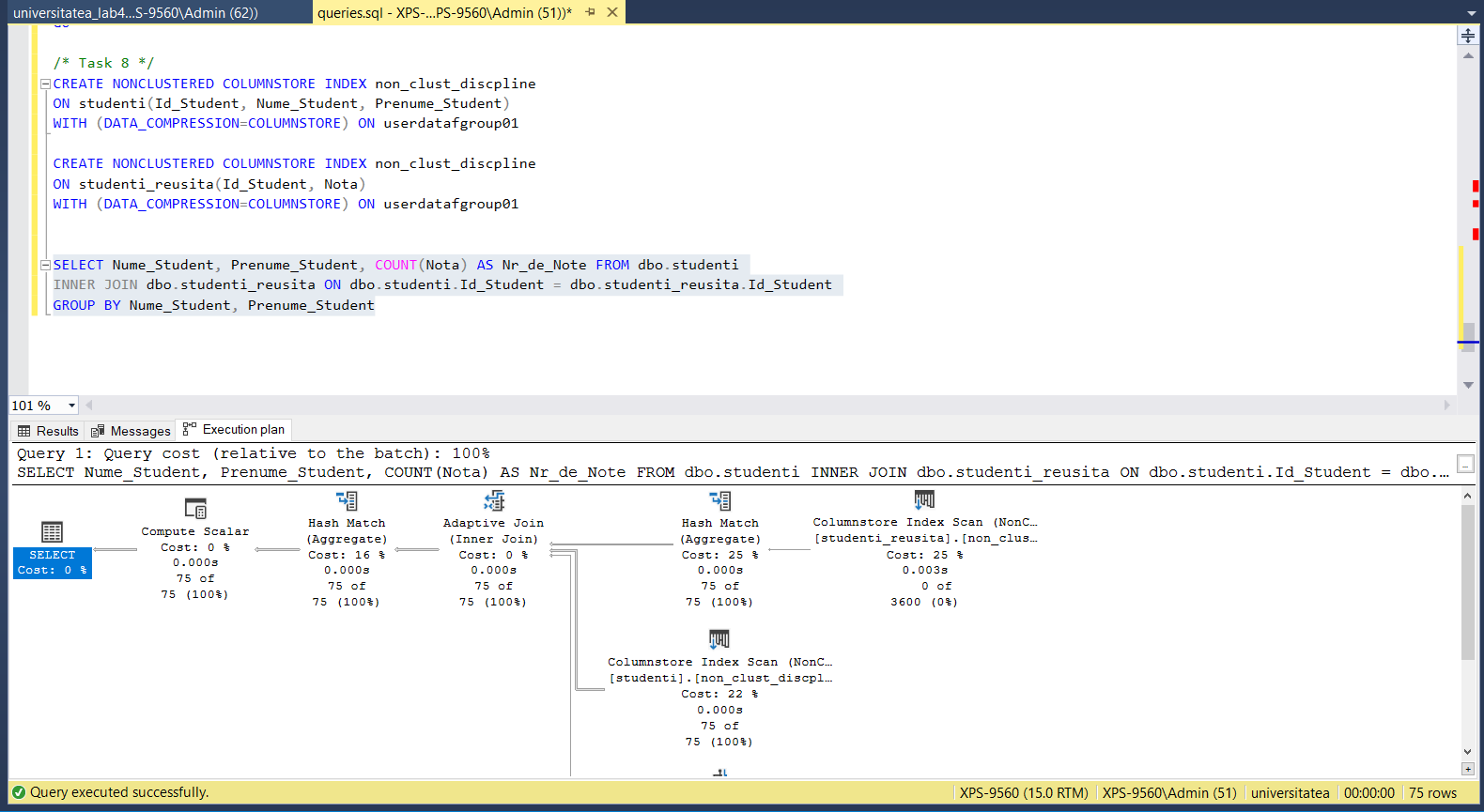
INNER JOIN dbo.studenti\_reusita ON dbo.studenti.Id\_Student = dbo.studenti\_reusita.Id\_Student

GROUP BY Nume\_Student, Prenume\_Student

Before:



After:



**Conclusion:**

In this work I had a possibility to work with CREATE and INSERT commands as well as practice different variations of them, such as INSERT INTO SELECT and SELECT INTO. Also, I’ve got a better understanding of indices and worked more with nested queries.