**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr.3

la cursul „Baze de date”

**Tema 3:** „ Proceduri și declanșatoare”

A efectuat : **TI-214 Reguș Ruslan**

A verificat:  **Gabriela Cebotar**

**Chișinău 2024**

# Primele 5 Exercitii practice de la pagina 329:

## Să se creeze o procedură stocată fara parametri. Procedura calculează numărul tuturor profesorilor la facultate.

**CREATE** **PROCEDURE** CalculateTotalProfessors

**AS**

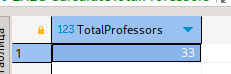
**BEGIN**

**SELECT** **COUNT**(\*) **AS** TotalProfessors

**FROM** profesori;

**END**;

**EXEC** CalculateTotalProfessors



## Să se creeze o procedură stocate, care nu are niciun parametru de intrare si posedă un parametru de iesire. Parametrul de iesire trebuie si returneze numărul de studenti, care nu au sustinut cel putin o form de evaluare (nota mai mica de 5 sau valoare NULL).

**CREATE** **PROCEDURE** NumarStudentiNepromovati

**@NumarStudentiNepromovati** **INT** **OUTPUT**

**AS**

**BEGIN**

**SELECT** **@NumarStudentiNepromovati** = **COUNT**(\*)

**FROM** studenti s

**WHERE** **NOT** **EXISTS** (

**SELECT** 1

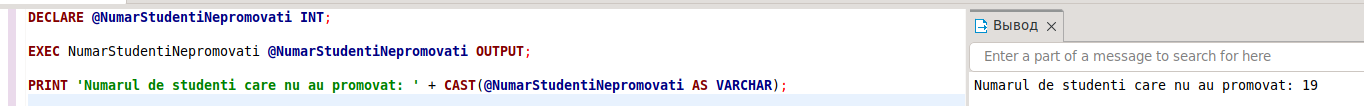
**FROM** studenti\_reusita sr

**WHERE** sr.Id\_Student = s.Id\_Student

**AND** (Nota < 5 **OR** Nota **IS** **NULL**)

);

**END**;



## Să se creeze o procedură stocată cu un parametru de intrare. De exemplu, trebuie numiti (numele, prenumele) profesorii care predau cel putin două discipline. Parametrul @mumar\_discipline specifics un număr de intrare mai mare decat 1.

**CREATE** **PROCEDURE** ProfesoriCuDouaDiscipline

**@NumarDiscipline** **INT**

**AS**

**BEGIN**

**SELECT** **DISTINCT** Nume\_Profesor, Prenume\_Profesor

**FROM** profesori p

**WHERE** Id\_Profesor **IN** (

**SELECT** Id\_Profesor

**FROM** studenti\_reusita sr

**WHERE** sr.Id\_Profesor = p.Id\_Profesor

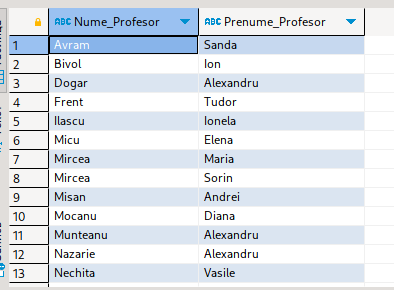
**GROUP** **BY** Id\_Profesor

**HAVING** **COUNT**(\*) >= **@NumarDiscipline**

);

**END**;

**EXEC** ProfesoriCuDouaDiscipline **@NumarDiscipline** = 100;



## Să se elaboreze procedura stocată, care calculează numărul de examene la care s-a prezentat un student al cărui nume e dat ca parametru.

**CREATE** **PROCEDURE** NumarExameneStudent

**@NumeStudent** **NVARCHAR**(50)

**AS**

**BEGIN**

**DECLARE** **@IdStudent** **INT**;

-- Obține ID-ul studentului folosind numele dat ca parametru

**SELECT** **@IdStudent** = Id\_Student

**FROM** studenti

**WHERE** Nume\_Student = **@NumeStudent**;

-- Calculează numărul de examene la care s-a prezentat studentul

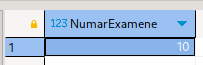
**SELECT** **COUNT**(\*) **AS** NumarExamene

**FROM** studenti\_reusita sr

**WHERE** sr.Id\_Student = **@IdStudent** **AND** sr.Tip\_Evaluare = **'Examen'**;

**END**;

**EXEC** NumarExameneStudent **@NumeStudent** = **'Coste'**;



## Să se creeze o procedură stocata, care ar insera in baza de date informatii despre un student nou. in calitate de parametri de intrare si serveasca datele personale ale studentului nou si Cod\_Grupa. Si se genereze toate intrarile-cheie necesare in tabelul studenti\_reusita. Notele de evaluare sa fie inserate ca NULL.

**CREATE** **PROCEDURE** InserareStudentNou

**@Nume** **NVARCHAR**(50),

**@Prenume** **NVARCHAR**(50),

**@CodGrupa** **NVARCHAR**(10),

**@AdresaPostalaStudent** **NVARCHAR**(100)

**AS**

**BEGIN**

**DECLARE** **@IdStudent** **INT**;

-- Inserare informatii despre studentul nou in tabela Studenti

**INSERT** **INTO** studenti (Nume\_Student, Prenume\_Student, Adresa\_Postala\_Student)

**VALUES** (**@Nume**, **@Prenume**, **@AdresaPostalaStudent**);

-- Obține ID-ul studentului nou inserat

**SET** **@IdStudent** = SCOPE\_IDENTITY();

-- Genereaza intrările-cheie necesare in tabela studenti\_reusita cu note de evaluare setate ca NULL

**INSERT** **INTO** studenti\_reusita (Id\_Student, Id\_Disciplina, Id\_Profesor, Id\_Grupa, Tip\_Evaluare, Nota)

**SELECT** **@IdStudent**, sd.Id\_Disciplina, g.Id\_Profesor, g.Id\_Grupa, **'Examen'**, **NULL**

**FROM** grupe g

**CROSS** **JOIN** studenti\_discipline sd

**WHERE** g.Cod\_Grupa = **@CodGrupa**;

**END**;

**EXEC** InserareStudentNou

**@Nume** = **'NumeStudent'**,

**@Prenume** = **'PrenumeStudent'**,

**@CodGrupa** = **'CodGrupa'**,

**@AdresaPostalaStudent** = **'AdresaStudent'**;

# Primele 5 exercitii practice de la pagina 344:

## Să se creeze un declansator, cara verifica daca au fost modificate printr-o instructiune UPDATE coloanele 5, si 6 din tabelul studenti\_reusita.

**CREATE** **TRIGGER** VerificareModificareColoane

**ON** studenti\_reusita

**AFTER** **UPDATE**

**AS**

**BEGIN**

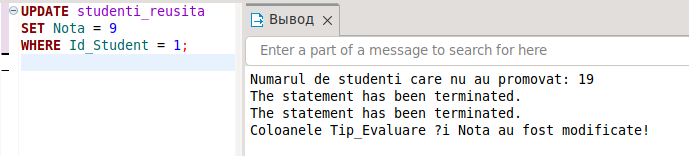
**IF** **UPDATE**(Tip\_Evaluare) **OR** **UPDATE**(Nota)

**BEGIN**

**PRINT** **'Coloanele Tip\_Evaluare și Nota au fost modificate!'**;

**END**

**END**;



## Să se modifice declansatorul tr\_inregistrare\_noua, in asa fel, incat in cazul actualizirii auditoriului si apara mesajul de informare, care, in afara de disciplina si ora, va afisa codul grupei afectate, ziua, blocul, auditoriul vechi si auditoriul nou.

**ALTER** **TRIGGER** tr\_inregistrare\_noua

**ON** Orar

**AFTER** **UPDATE**

**AS**

**BEGIN**

**IF** **UPDATE**(Auditoriu)

**BEGIN**

**DECLARE** **@Disciplina** **NVARCHAR**(50);

**DECLARE** **@Ora** **TIME**;

**DECLARE** **@CodGrupa** **NVARCHAR**(10);

**DECLARE** **@Zi** **NVARCHAR**(20);

**DECLARE** **@Bloc** **NVARCHAR**(20);

**DECLARE** **@AuditoriuVechi** **NVARCHAR**(10);

**DECLARE** **@AuditoriuNou** **NVARCHAR**(10);

**SELECT** **@Disciplina** = INSERTED.Disciplina,

**@Ora** = INSERTED.Ora,

**@CodGrupa** = INSERTED.Cod\_Grupa,

**@Zi** = INSERTED.Zi,

**@Bloc** = INSERTED.Bloc,

**@AuditoriuVechi** = DELETED.Auditoriu,

**@AuditoriuNou** = INSERTED.Auditoriu

**FROM** INSERTED

**INNER** **JOIN** DELETED **ON** INSERTED.Id\_Orar = DELETED.Id\_Orar;

**PRINT** **'Auditoriul pentru disciplina '** + **@Disciplina** + **', ora '** + **CONVERT**(**NVARCHAR**, **@Ora**) +

**', în grupa '** + **@CodGrupa** + **', ziua '** + **@Zi** + **', blocul '** + **@Bloc** +

**', a fost modificat de la auditoriul '** + **@AuditoriuVechi** + **' la auditoriul '** + **@AuditoriuNou** + **'.'**;

**END**

**END**;

## Să se creeze declansatorul, care ar asigura popularea corecta (consecutiva) a tabelelor studenti si studenti\_reusita, si ar permite evitarea erorilor la nivelul cheilor externe.

**CREATE** **TRIGGER** tr\_PopulareCorectaStudenti

**ON** studenti

**AFTER** **INSERT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON**;

**DECLARE** **@UltimulId** **INT**;

**SELECT** **@UltimulId** = **MAX**(Id\_Student) **FROM** studenti;

**INSERT** **INTO** studenti\_reusita (Id\_Student, Id\_Disciplina, Id\_Profesor, Id\_Grupa, Tip\_Evaluare, Nota)

**SELECT** i.Id\_Student, d.Id\_Disciplina, p.Id\_Profesor, g.Id\_Grupa, **'Examen'**, **NULL**

**FROM** inserted i

**CROSS** **JOIN** discipline d

**CROSS** **JOIN** profesori p

**CROSS** **JOIN** grupe g

**WHERE** i.Id\_Student > **@UltimulId**;

**END**;

**INSERT** **INTO** studenti (Id\_Student , Nume\_Student, Prenume\_Student, Adresa\_Postala\_Student)

**VALUES** (55,**'NumeStudent'**, **'PrenumeStudent'**, **''**);

## Să se creeze un declansator, care ar interzice micgorarea notelor in tabelul studenti\_reusita si modificarea valorilor cAmpului Data\_Evaluare, unde valorile acestui camp sunt nenule. Declansatorul trebuie sa se lanseze, numai daci sunt afectate datele studentilor din grupa ,CIB171”. Se va afisa un mesaj de avertizare in cazul tentativei de a inclca constrangerea.

**CREATE** **TRIGGER** tr\_InterzicereModificareNote

**ON** studenti\_reusita

**AFTER** **UPDATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON**;

-- Verificăm dacă sunt afectate datele studenților din grupa "CIB171"

**IF** **EXISTS** (

**SELECT** 1

**FROM** inserted i

**JOIN** deleted d **ON** i.Id\_Student = d.Id\_Student

**JOIN** grupe g **ON** i.Id\_Grupa = g.Id\_Grupa

**WHERE** g.Cod\_Grupa = **'CIB171'**

)

**BEGIN**

-- Verificăm dacă se încearcă micșorarea notelor sau modificarea câmpului Data\_Evaluare

**IF** **EXISTS** (

**SELECT** 1

**FROM** inserted

**JOIN** deleted **ON** inserted.Id\_Student = deleted.Id\_Student

**WHERE** inserted.Nota < deleted.Nota **OR** (inserted.Data\_Evaluare **IS** **NOT** **NULL** **AND** deleted.Data\_Evaluare **IS** **NOT** **NULL**)

)

**BEGIN**

**PRINT** **'ATENȚIE: Nu este permisă micșorarea notelor sau modificarea câmpului Data\_Evaluare pentru studenții din grupa CIB171.'**;

**ROLLBACK**; -- Anulăm modificările

**END**

**END**

**END**;

**UPDATE** studenti\_reusita

**SET** Nota = 7

**WHERE** Id\_Student = 101;



## Să se creeze un declansator DDL care ar interzice modificarea coloanei Id\_Disciplina in tabelele bazei de date universitatea cu afisarea mesajului respectiv.

**CREATE** **TRIGGER** tr\_InterzicereModificareIdDisciplina

**ON** **DATABASE**

**FOR** ALTER\_TABLE

**AS**

**BEGIN**

**SET** NOCOUNT **ON**;

**DECLARE** **@EventData** **XML** = EVENTDATA();

**DECLARE** **@SchemaName** **NVARCHAR**(255) = **@EventData**.value(**'(/EVENT\_INSTANCE/SchemaName)[1]'**, **'NVARCHAR(255)'**);

**DECLARE** **@TableName** **NVARCHAR**(255) = **@EventData**.value(**'(/EVENT\_INSTANCE/ObjectName)[1]'**, **'NVARCHAR(255)'**);

**DECLARE** **@ColumnName** **NVARCHAR**(255) = **@EventData**.value(**'(/EVENT\_INSTANCE/AlterTableActionList/\*/Columns/Name)[1]'**, **'NVARCHAR(255)'**);

**IF** **@ColumnName** = **'Id\_Disciplina'**

**BEGIN**

**PRINT** **'Modificarea coloanei Id\_Disciplina este interzisă!'**;

**ROLLBACK**; -- Anulăm modificarea

**END**

**END**;

# Concluzie

Pe baza sarcinilor realizate, am înțeles importanța utilizării declanșatorilor în bazele de date pentru a controla și a gestiona modificările asupra datelor. Am învățat să folosesc declanșatorii pentru a implementa diverse funcționalități, cum ar fi verificarea modificărilor în anumite coloane, interzicerea unor tipuri specifice de modificări sau realizarea unor acțiuni automatice după anumite evenimente.

Prin crearea de declanșatori am dobândit cunoștințe despre:

1. Utilizarea declanșatorilor AFTER și INSTEAD OF pentru a reacționa la evenimente precum INSERT, UPDATE și DELETE.

2. Modul în care pot folosi declanșatorii pentru a valida și controla modificările făcute asupra datelor în tabele.

3. Implementarea restricțiilor de integritate referențială și a altor reguli de afaceri în baza de date.

4. Importanța gestionării tranzacțiilor pentru a menține consistența datelor în baza de date.

Prin aplicarea acestor concepte în diverse sarcini, am dezvoltat abilități practice în crearea, testarea și gestionarea declanșatorilor în mediul SQL Server. Aceste cunoștințe și competențe sunt esențiale pentru administrarea și dezvoltarea eficientă a bazelor de date în mediul profesional și academic.