Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут прикладної математики та фундаментальних наук   
Кафедра прикладної математики

Звіт

про виконання лабораторної роботи №3

“Підпрограми СУБД Oracle”

з курсу “Організація баз даних і знань”

Виконала:

Студентка групи ПМ-22

Кудрявцева О.М

Перевірив

Пабирівський В.В.

Львів-2024р.

**Завдання:**

1.У всіх таблицях створити поля UCR, DCR, ULC, DLC. Написати тригери які будуть заповнювати дані поля наступним чином: UCR – ім’я користувача, що створив даний запис; DCR – дата та час створення даного запису; ULC – ім’я користувача, що останнім змінив даний запис; DLC – дата та час останньої модифікації даного запису.

2.Створити сурогатний ключ для деякої таблиці, та написати тригер для обов’язкового заповнення цього поля послідовними значеннями.

3.Написати тригери для перевірки наступних обмежень цілісності:

Пацієнту не можна призначати лікарства, які є протипоказані йому, або містять протипоказані складові. Лікар не може одночасно мати більше 10 пацієнтів.

**Хід роботи:**

Створила поля UCR, DCR, ULC, DLC та створила відповідні тригери для заповнення цих полів:

ALTER TABLE HistoryOfIllness

ADD UCR NVARCHAR(50),

DCR DATETIME,

ULC NVARCHAR(50),

DLC DATETIME;

ALTER TABLE Patient

ADD UCR NVARCHAR(50),

DCR DATETIME,

ULC NVARCHAR(50),

DLC DATETIME;

ALTER TABLE Doctor

ADD UCR NVARCHAR(50),

DCR DATETIME,

ULC NVARCHAR(50),

DLC DATETIME;

ALTER TABLE PrescribedMedication

ADD UCR NVARCHAR(50),

DCR DATETIME,

ULC NVARCHAR(50),

DLC DATETIME;

ALTER TABLE Medicines

ADD UCR NVARCHAR(50),

DCR DATETIME,

ULC NVARCHAR(50),

DLC DATETIME;

CREATE TRIGGER HistoryOfIllness\_Trigger

ON HistoryOfIllness

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @HistoryID INT;

SELECT @HistoryID = HistoryID FROM INSERTED;

UPDATE HistoryOfIllness

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE HistoryID = @HistoryID;

END

SELECT @HistoryID = HistoryID FROM INSERTED;

UPDATE HistoryOfIllness

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE HistoryID = @HistoryID;

END;

CREATE TRIGGER Patient\_Trigger

ON Patient

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @ID INT;

SELECT @ID = ID FROM INSERTED;

UPDATE Patient

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE ID = @ID;

END

SELECT @ID = ID FROM INSERTED;

UPDATE Patient

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE ID = @ID;

END;

CREATE TRIGGER Doctor\_Trigger

ON Doctor

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @ID INT;

SELECT @ID = ID FROM INSERTED;

UPDATE Doctor

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE ID = @ID;

END

SELECT @ID = ID FROM INSERTED;

UPDATE Doctor

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE ID = @ID;

END;

CREATE TRIGGER Medicines\_Trigger

ON Medicines

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @NameOfMedicines NVARCHAR(100);

SELECT @NameOfMedicines = NameOfMedicines FROM INSERTED;

UPDATE Medicines

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE NameOfMedicines = @NameOfMedicines;

END

SELECT @NameOfMedicines = NameOfMedicines FROM INSERTED;

UPDATE Medicines

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE NameOfMedicines = @NameOfMedicines;

END;

CREATE TRIGGER Medicines\_Trigger

ON Medicines

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @NameOfMedicines NVARCHAR(100);

SELECT @NameOfMedicines = NameOfMedicines FROM INSERTED;

UPDATE Medicines

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE NameOfMedicines = @NameOfMedicines;

END

SELECT @NameOfMedicines = NameOfMedicines FROM INSERTED;

UPDATE Medicines

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE NameOfMedicines = @NameOfMedicines;

END;

CREATE TRIGGER PrescribedMedication\_Trigger

ON PrescribedMedication

AFTER INSERT, UPDATE

AS

BEGIN

IF NOT UPDATE(UCR) AND NOT UPDATE(DCR)

BEGIN

DECLARE @UniqueNumber INT;

SELECT @UniqueNumber = UniqueNumber FROM INSERTED;

UPDATE PrescribedMedication

SET ULC = USER\_NAME(),

DLC = GETDATE()

WHERE UniqueNumber = @UniqueNumber;

END

SELECT @UniqueNumber = UniqueNumber FROM INSERTED;

UPDATE PrescribedMedication

SET UCR = USER\_NAME(),

DCR = GETDATE()

WHERE UniqueNumber = @UniqueNumber;

END;

Потім створила сурогатний ключ та тригер для обов’язкового заповнення цього поля:

ALTER TABLE PrescribedMedication

ADD MedID INT IDENTITY(1,1) PRIMARY KEY;

CREATE TRIGGER FillMedID

ON PrescribedMedication

INSTEAD OF INSERT

AS

BEGIN

SET IDENTITY\_INSERT PrescribedMedication ON;

DECLARE @NextMedID INT;

SET @NextMedID = COALESCE((SELECT MAX(MedID) FROM PrescribedMedication), 0) + 1;

INSERT INTO PrescribedMedication (MedID, NameOfMedicines, UniqueNumber, HistoryID, UCR, DCR, ULC, DLC)

SELECT @NextMedID, NameOfMedicines, UniqueNumber, HistoryID, UCR, DCR, ULC, DLC

FROM INSERTED;

SET IDENTITY\_INSERT PrescribedMedication OFF;

END;

**Тригер для перевірки, що пацієнту не можна призначати лікарства, які є протипоказані йому, або містять протипоказані складові:**  
CREATE TRIGGER CheckPrescribedMedication

ON PrescribedMedication

FOR INSERT

AS

BEGIN

DECLARE @MedicineName nvarchar(100)

DECLARE @PatientID int

DECLARE @ContrindicatedMedications nvarchar(50)

SELECT @MedicineName = i.NameOfMedicines, @PatientID = h.PatientID

FROM inserted i

INNER JOIN HistoryOfIllness h ON i.HistoryID = h.HistoryID

SELECT @ContrindicatedMedications = p.ContraindicatedMedications

FROM Patient p

INNER JOIN HistoryOfIllness h ON p.ID = h.PatientID

WHERE p.ID = @PatientID

IF (@ContrindicatedMedications IS NOT NULL AND

(CHARINDEX(@MedicineName, @ContrindicatedMedications) > 0 OR

EXISTS (SELECT 1 FROM Medicines WHERE NameOfMedicines = @MedicineName AND CompoundOfMedicines LIKE '%' + @ContrindicatedMedications + '%')))

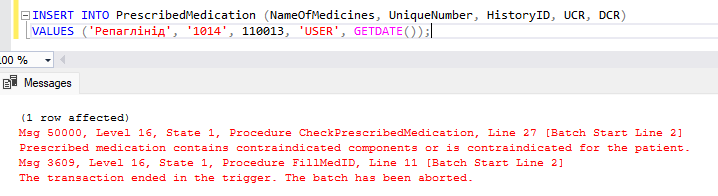
BEGIN

RAISERROR ('Prescribed medication contains contraindicated components or is contraindicated for the patient.', 16, 1)

ROLLBACK TRANSACTION

END

END

**Перевірка роботи тригера:**  


**Тригер для перевірки, що, лікар не може одночасно мати більше 10 пацієнтів**:  
CREATE TRIGGER CheckPatientsLimit

ON HistoryOfIllness

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @DoctorID INT;

DECLARE @MaxPatientsPerDoctor INT = 10;

SELECT @DoctorID = DoctorID

FROM INSERTED

GROUP BY DoctorID;

IF (SELECT COUNT(\*) FROM HistoryOfIllness WHERE DoctorID = @DoctorID) > @MaxPatientsPerDoctor

BEGIN

RAISERROR ('Лікар не може мати більше %d пацієнтів одночасно', 16, 1, @MaxPatientsPerDoctor);

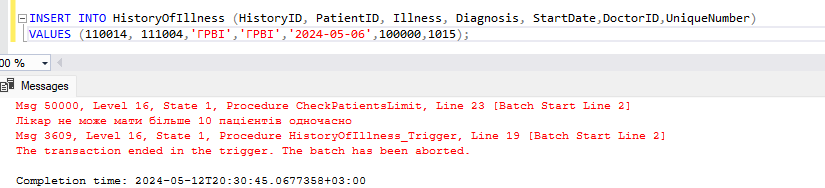
ROLLBACK TRANSACTION;

RETURN;

END;

END;

**Перевірка роботу тригера:**



**Посилання на репозиторій GitHub:**

<https://github.com/Oleksandra-Kudriavtseva/DBSQLKudriavtseva.git>

**Висновок:** Написала тригери відповідно до завдань лабораторної роботи та перевірила їх роботу.