МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна Навчально-науковий інститут комп’ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

ЛАБОРАТОРНА РОБОТА № 7

на тему: «MS SQL»

ВАРИАНТ № 26

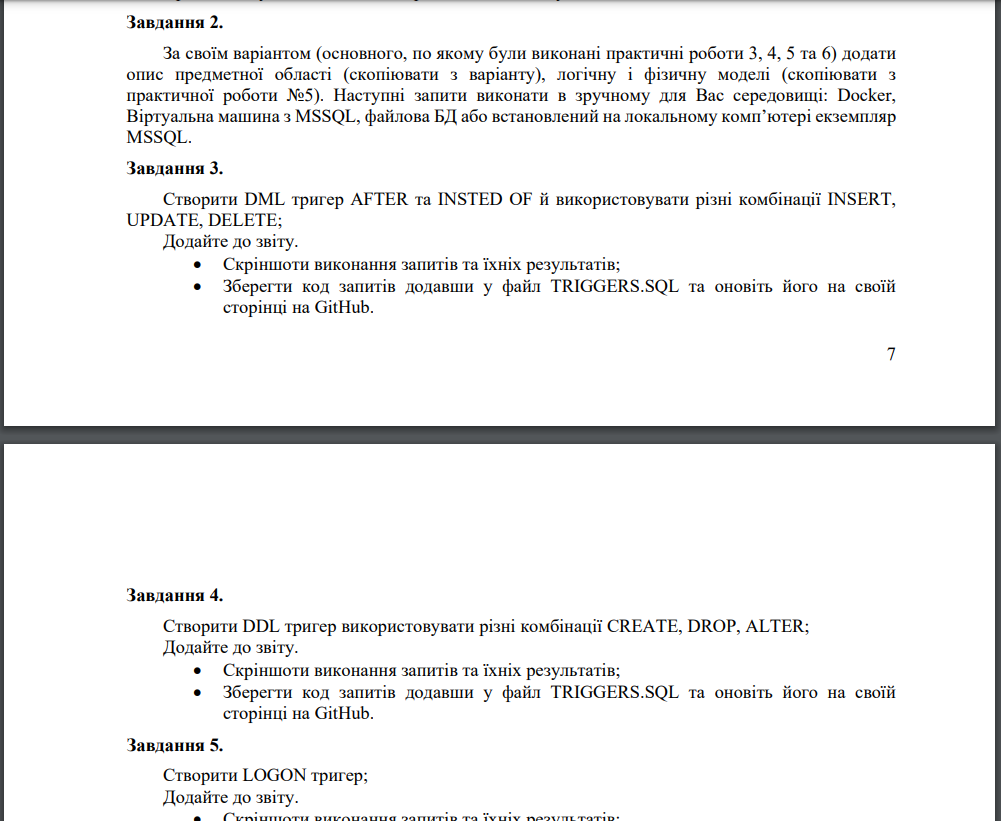
Виконав: студент 3 курсу групи КС-32 Спеціальності 122 «Комп’ютерні науки» Травченко Сергій Миколайович

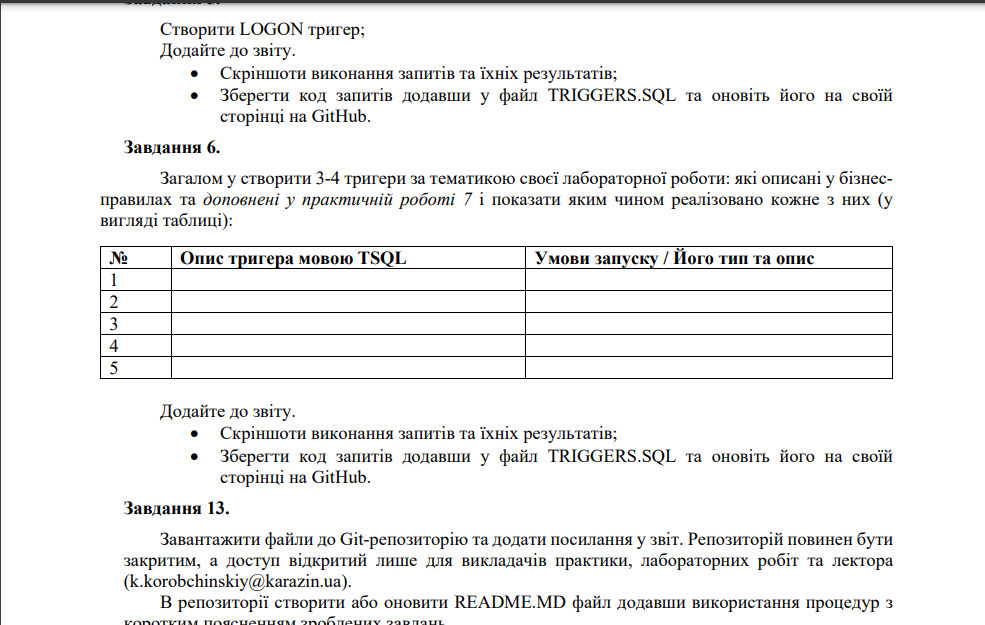
Прийняв: доцент ЗВО каф. ММ та АД

Маций О.Б.

Харків – 2025

**Мета роботи:** Навчиться створювати, змінювати та видаляти таблиці в БД, заносити, оновлювати та видаляти інформацію з цих таблиць. Навчиться створювати найпростіші запити до бази даних.





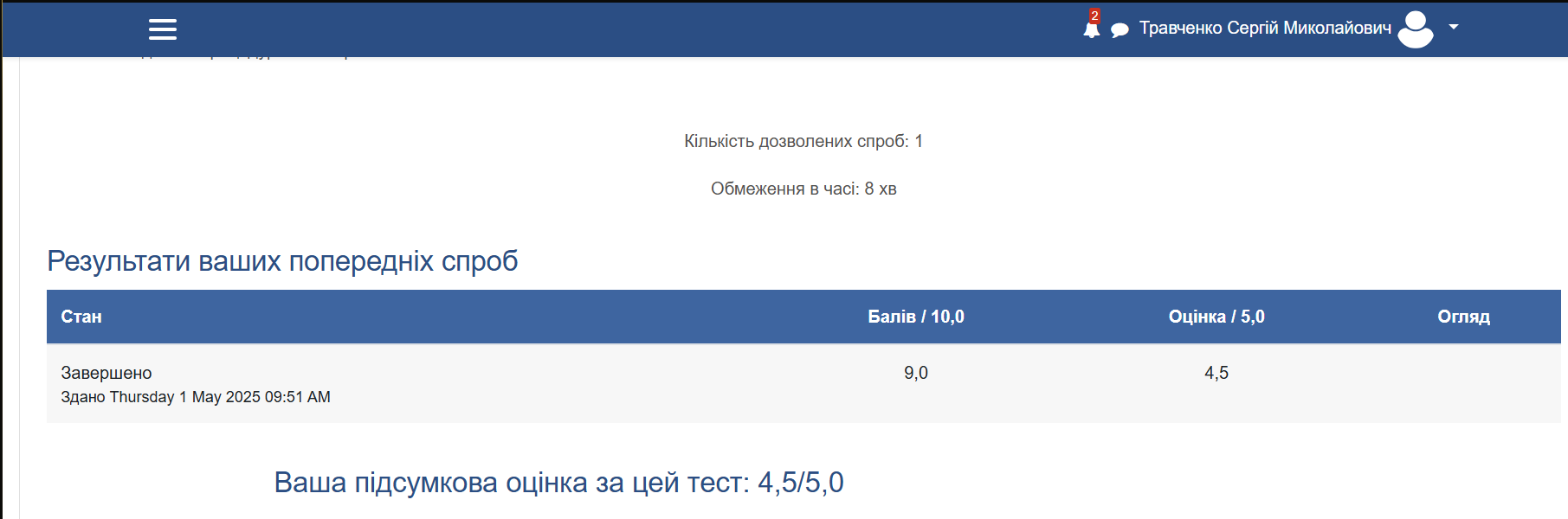


Рисунок 1 – виконання тестування.

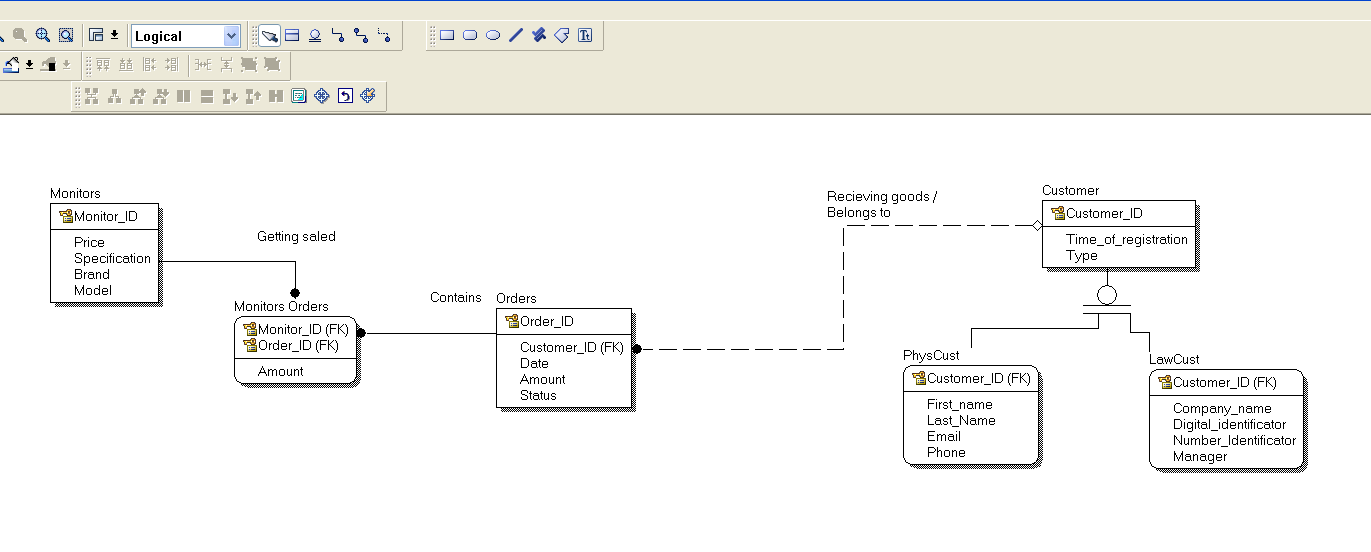


Рисунок 2 – ER-діаграма(логічна) з власної лабораторної роботи №5



Рисунок 3 – ER-діаграма(фізична) з власної лабораторної роботи №5

**Опис предметної області**

Магазин моніторів **GlobeXMonitors** спеціалізується на продажу різноманітних моніторів для комп'ютерів та інших пристроїв. Компанія пропонує широкий асортимент продукції від різних брендів, охоплюючи різні категорії моніторів, такі як **LED**, **LCD**, **OLED** та інші типи екранних технологій. Метою магазину є надання своїм клієнтам високоякісних моніторів, що відповідають різним вимогам щодо функціональності та дизайну. Магазин **GlobeXMonitors** є прикладом ефективної організації бізнес-процесів у сфері роздрібної торгівлі електронікою, де важливу роль відіграє класифікація товарів, управління клієнтською базою, ефективне стеження за замовленнями та інтерфейс для обробки даних. Створення структурованої бази даних забезпечує правильну обробку інформації, зручність управління та високий рівень обслуговування клієнтів.

**Скріни працездатності**

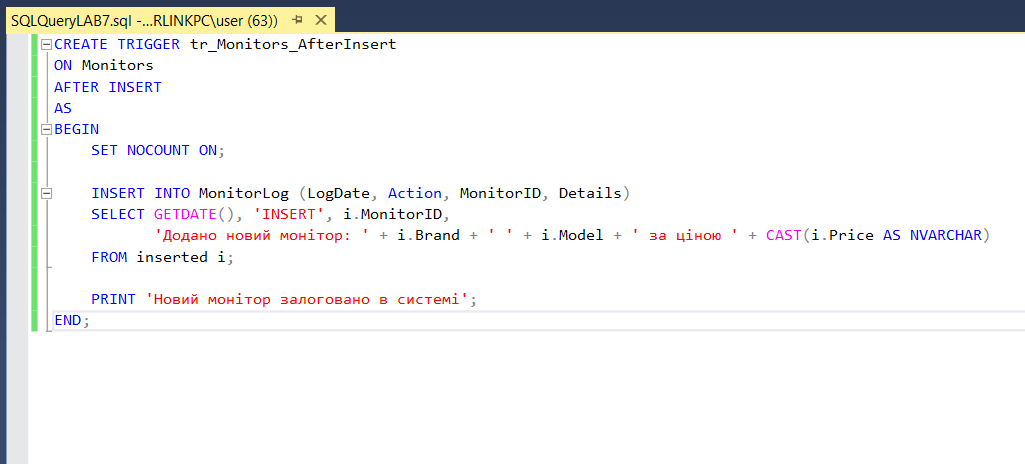


Рисунок 4 – приклад вдалого працювання скрипту

**Створення скриптів T-SQL**

**SETUP.SQL**

CREATE TABLE ProductCategories (

CategoryID INT PRIMARY KEY IDENTITY(1,1),

CategoryName NVARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE Monitors (

MonitorID INT PRIMARY KEY IDENTITY(1,1),

Model NVARCHAR(100) NOT NULL,

Brand NVARCHAR(50) NOT NULL,

Price DECIMAL(10,2) NOT NULL,

Specifications NVARCHAR(500),

CategoryID INT NOT NULL FOREIGN KEY REFERENCES ProductCategories(CategoryID)

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

Email NVARCHAR(100) NOT NULL UNIQUE,

Phone NVARCHAR(15)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY IDENTITY(1,1),

CustomerID INT NOT NULL FOREIGN KEY REFERENCES Customers(CustomerID),

OrderDate DATE NOT NULL,

TotalAmount DECIMAL(10,2) NOT NULL,

OrderStatus NVARCHAR(20) NOT NULL

);

CREATE TABLE Suppliers (

SupplierID INT PRIMARY KEY IDENTITY(1,1),

SupplierName NVARCHAR(100) NOT NULL,

ContactInfo NVARCHAR(200)

);

-- Зміна структури таблиці Monitors: додаємо поле WarrantyPeriod

ALTER TABLE Monitors

ADD WarrantyPeriod INT NOT NULL DEFAULT 12;

**INSERT100.SQL**

-- Заповнення таблиці ProductCategories

DECLARE @i INT = 1;

WHILE @i <= 100

BEGIN

INSERT INTO ProductCategories (CategoryName) VALUES ('Category ' + CAST(@i AS NVARCHAR));

SET @i = @i + 1;

END;

-- Заповнення таблиці Monitors

DECLARE @j INT = 1;

WHILE @j <= 100

BEGIN

INSERT INTO Monitors (Model, Brand, Price, Specifications, CategoryID, WarrantyPeriod)

VALUES ('Model ' + CAST(@j AS NVARCHAR), 'Brand ' + CAST(@j AS NVARCHAR), 100.00 + @j, 'Specifications for Model ' + CAST(@j AS NVARCHAR), @j, 12 + (@j % 24));

SET @j = @j + 1;

END;

-- Заповнення таблиці Customers

DECLARE @k INT = 1;

WHILE @k <= 100

BEGIN

INSERT INTO Customers (FirstName, LastName, Email, Phone)

VALUES ('FirstName' + CAST(@k AS NVARCHAR), 'LastName' + CAST(@k AS NVARCHAR), 'user' + CAST(@k AS NVARCHAR) + '@example.com', '123456789' + CAST(@k AS NVARCHAR));

SET @k = @k + 1;

END;

-- Заповнення таблиці Orders

DECLARE @l INT = 1;

WHILE @l <= 100

BEGIN

INSERT INTO Orders (CustomerID, OrderDate, TotalAmount, OrderStatus)

VALUES (@l, DATEADD(DAY, @l, '2024-01-01'), 100.00 + @l, CASE WHEN @l % 2 = 0 THEN 'Completed' ELSE 'Pending' END);

SET @l = @l + 1;

END;

-- Заповнення таблиці Suppliers

DECLARE @m INT = 1;

WHILE @m <= 100

BEGIN

INSERT INTO Suppliers (SupplierName, ContactInfo)

VALUES ('Supplier ' + CAST(@m AS NVARCHAR), 'contact' + CAST(@m AS NVARCHAR) + '@supplier.com');

SET @m = @m + 1;

END;

**SQLQueryLAB7.SQL**

CREATE TRIGGER tr\_Monitors\_AfterInsert

ON Monitors

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO MonitorLog (LogDate, Action, MonitorID, Details)

SELECT GETDATE(), 'INSERT', i.MonitorID,

'Додано новий монітор: ' + i.Brand + ' ' + i.Model + ' за ціною ' + CAST(i.Price AS NVARCHAR)

FROM inserted i;

PRINT 'Новий монітор залоговано в системі';

END;

CREATE TRIGGER tr\_Monitors\_InsteadOfUpdate

ON Monitors

INSTEAD OF UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Блокуємо зниження ціни більше ніж на 20%

IF UPDATE(Price)

BEGIN

IF EXISTS (

SELECT 1 FROM inserted i

JOIN deleted d ON i.MonitorID = d.MonitorID

WHERE i.Price < d.Price \* 0.8

)

BEGIN

RAISERROR('Зниження ціни більше ніж на 20% заборонено', 16, 1);

RETURN;

END

END

-- Якщо все в порядку, виконуємо оновлення

UPDATE m

SET

m.Model = i.Model,

m.Brand = i.Brand,

m.Price = i.Price,

m.Specifications = i.Specifications,

m.CategoryID = i.CategoryID,

m.WarrantyPeriod = i.WarrantyPeriod

FROM Monitors m

JOIN inserted i ON m.MonitorID = i.MonitorID;

PRINT 'Дані монітора успішно оновлено';

END;

CREATE TRIGGER tr\_Orders\_AfterDelete

ON Orders

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO OrdersArchive (OrderID, CustomerID, MonitorID, OrderDate, TotalAmount, OrderStatus, ArchiveDate)

SELECT

d.OrderID, d.CustomerID, d.MonitorID, d.OrderDate, d.TotalAmount, d.OrderStatus, GETDATE()

FROM deleted d;

PRINT 'Видалене замовлення архівовано';

END;

CREATE TRIGGER tr\_Database\_TableChanges

ON DATABASE

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @EventData XML = EVENTDATA();

INSERT INTO DatabaseChangeLog (ChangeDate, ChangeType, ObjectName, UserName, SQLCommand)

VALUES (

GETDATE(),

@EventData.value('(/EVENT\_INSTANCE/EventType)[1]', 'nvarchar(100)'),

@EventData.value('(/EVENT\_INSTANCE/ObjectName)[1]', 'nvarchar(100)'),

@EventData.value('(/EVENT\_INSTANCE/UserName)[1]', 'nvarchar(100)'),

@EventData.value('(/EVENT\_INSTANCE/TSQLCommand/CommandText)[1]', 'nvarchar(max)')

);

PRINT 'Зміна структури БД залогована';

END;

CREATE TRIGGER tr\_Reviews\_UpdateRating

ON MonitorReviews

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

SET NOCOUNT ON;

-- Оновлюємо середній рейтинг для всіх моніторів, які були зачіплені

UPDATE m

SET

m.AverageRating = (

SELECT AVG(CAST(r.Rating AS DECIMAL(3,1)))

FROM MonitorReviews r

WHERE r.MonitorID = m.MonitorID

),

m.ReviewCount = (

SELECT COUNT(\*)

FROM MonitorReviews r

WHERE r.MonitorID = m.MonitorID

)

FROM Monitors m

WHERE m.MonitorID IN (

SELECT MonitorID FROM inserted

UNION

SELECT MonitorID FROM deleted

);

END;

CREATE TRIGGER tr\_Monitors\_CheckSerialNumber

ON Monitors

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF UPDATE(SerialNumber)

BEGIN

IF EXISTS (

SELECT 1 FROM inserted i

GROUP BY i.SerialNumber

HAVING COUNT(\*) > 1

)

OR EXISTS (

SELECT 1 FROM inserted i

JOIN Monitors m ON i.SerialNumber = m.SerialNumber

WHERE i.MonitorID <> m.MonitorID

)

BEGIN

RAISERROR('Серійний номер повинен бути унікальним', 16, 1);

ROLLBACK TRANSACTION;

END

END

END;

**Надаю посилання на репозиторій Github з усьома двома запитами.** <https://github.com/SerlinkX/CS32-Travchenko_DB_Pract1>

**Висновки**

Процес розробки та управління базою даних для магазину моніторів є важливим етапом для забезпечення правильної організації даних, збереження цілісності інформації та ефективної роботи з нею. Завдяки використанню стандартів T-SQL, вдалося реалізувати всі основні операції з базою даних, включаючи створення, внесення, оновлення та видалення даних. Важливою частиною є правильне застосування обмежень і ключів для уникнення помилок при введенні даних та підтримки коректних зв'язків між сутностями.

В результаті, база даних для **GlobeXMonitors** готова до подальшого використання і може бути ефективно використовувана для ведення бізнес-процесів у магазині.