

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Рязанский государственный радиотехнический
университет имени В.Ф. Уткина»
Рязанский станкостроительный колледж

Отчёт о практической работе №18
Работа с триггерами.
«Основы проектирования баз данных»

Выполнил:
студент группы ИСП-22
Маркина Н.А
Проверил:
Родин Е.Н

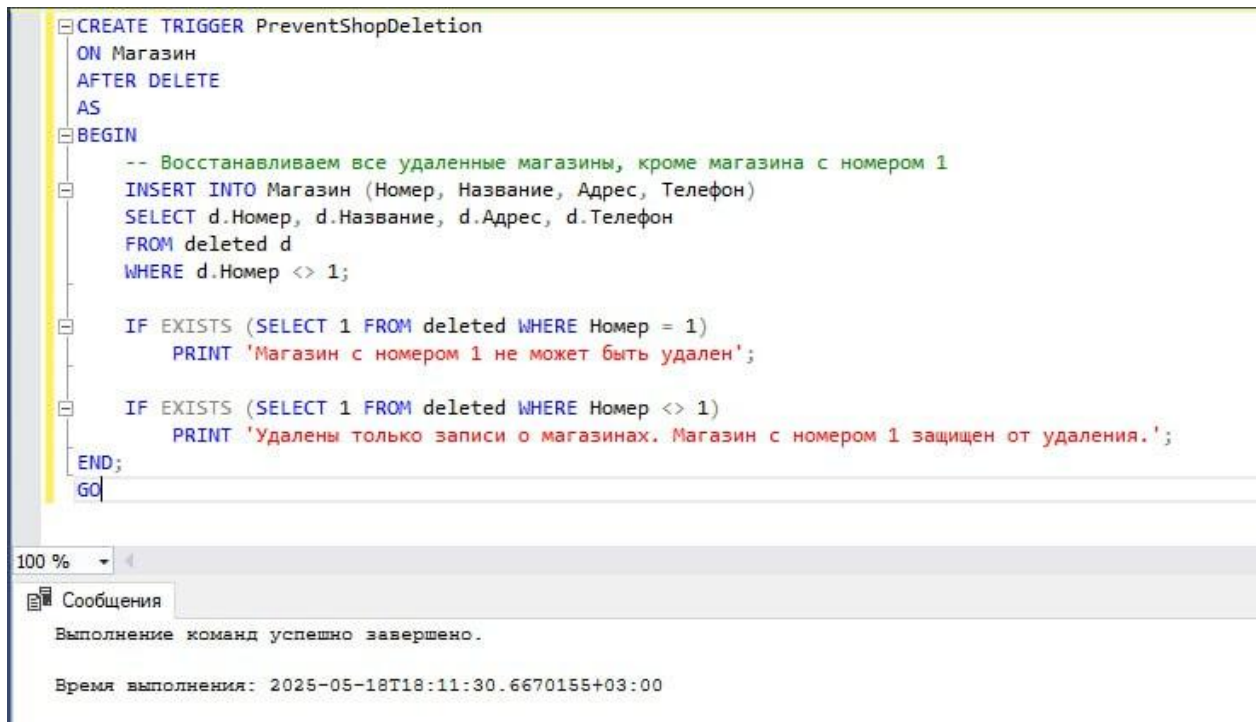
Рязань 2025

Цель работы:

Научиться разрабатывать триггеры.

Ход работы:

Изучение оператора CREATE TRIGGER для создания триггеров к базовым таблицам БД.



```
CREATE TRIGGER PreventShopDeletion
ON Магазин
AFTER DELETE
AS
BEGIN
    -- Восстанавливаем все удаленные магазины, кроме магазина с номером 1
    INSERT INTO Магазин (Номер, Название, Адрес, Телефон)
    SELECT d.Номер, d.Название, d.Адрес, d.Телефон
    FROM deleted d
    WHERE d.Номер <> 1;

    IF EXISTS (SELECT 1 FROM deleted WHERE Номер = 1)
        PRINT 'Магазин с номером 1 не может быть удален';

    IF EXISTS (SELECT 1 FROM deleted WHERE Номер <> 1)
        PRINT 'Удалены только записи о магазинах. Магазин с номером 1 защищен от удаления.';
END;
GO
```

100 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-18T18:11:30.6670155+03:00

Рис.1

Создала триггер PreventShopDeletion на таблице Магазин, который активируется после операций удаления. Триггер реализует защиту данных: автоматически восстанавливает все удалённые магазины (кроме магазина с номером 1) и выводит соответствующие предупреждения. Для магазина с номером 1 предусмотрена особая защита — его удаление полностью блокируется с выводом сообщения.

```
CREATE TRIGGER InsteadOfDeleteShop
ON Магазин
INSTEAD OF DELETE
AS
BEGIN
    -- Удаляем только магазины с номером не равным 1
    DELETE FROM Магазин
    WHERE Номер IN (SELECT Номер FROM deleted WHERE Номер <> 1);

    IF EXISTS (SELECT 1 FROM deleted WHERE Номер = 1)
        PRINT 'Магазин с номером 1 не может быть удален';

    PRINT 'INSTEAD OF TRIGGER: Удалены только разрешенные магазины';
END;
GO
```

100 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-18T18:11:32.6543008+03:00

Рис.2

Создала триггер InsteadOfDeleteShop типа INSTEAD OF DELETE для таблицы Магазин, который перехватывает операции удаления и реализует условное удаление данных. Триггер разрешает удаление только магазинов с номерами, отличными от 1, а для магазина с номером 1 выводит защитное сообщение. Дополнительно генерируется информационное сообщение о результате работы триггера.

```
CREATE TRIGGER CheckPriceDecrease
ON Комплектующее
AFTER UPDATE
AS
BEGIN
    IF UPDATE(Цена)
    BEGIN
        -- Проверяем, есть ли записи с увеличенной ценой
        IF EXISTS (
            SELECT 1
            FROM inserted i
            JOIN deleted d ON i.Модель = d.Модель
            WHERE i.Цена > d.Цена
        )
        BEGIN
            -- Откатываем изменения
            ROLLBACK TRANSACTION;
            RAISERROR('Ошибка: Цена комплектующего может быть только уменьшена', 16, 1);
            RETURN;
        END
        PRINT 'Цены успешно обновлены (только уменьшение)';
    END
END;
GO
```

100 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-18T18:11:34.9015354+03:00

Рис.3

Создала триггер CheckPriceDecrease типа AFTER UPDATE для таблицы Комплектующее, который контролирует изменение цен. Триггер проверяет, чтобы обновление приводило только к уменьшению цены. При попытке увеличения цены триггер откатывает транзакцию и генерирует ошибку с сообщением. В случае успешного уменьшения цены выводится подтверждающее сообщение.

```
CREATE VIEW НаличиеКомплектующих
AS
SELECT
    м.Номер AS [Номер магазина],
    м.Название AS [Название магазина],
    к.Модель AS [Модель комплектующей],
    к.Наименование AS [Название комплектующей],
    н.Количество,
    н.Год
FROM
    Наличие н
JOIN Магазин м ON н.Магазин = м.Номер
JOIN Комплектующее к ON н.Модель = к.Модель;
GO
```

100 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-18T18:11:36.6870189+03:00

Рис.4

Создала представление **НаличиеКомплектующих**, которое агрегирует данные из трёх таблиц (**Наличие**, **Магазин**, **Комплектующее**) для отображения детализированной информации о наличии товаров в магазинах. Представление включает номер и название магазина, модель и наименование комплектующей, количество и год учёта.

```
SQLQuery25.sql - D...Пользователь (55))* -p X
```

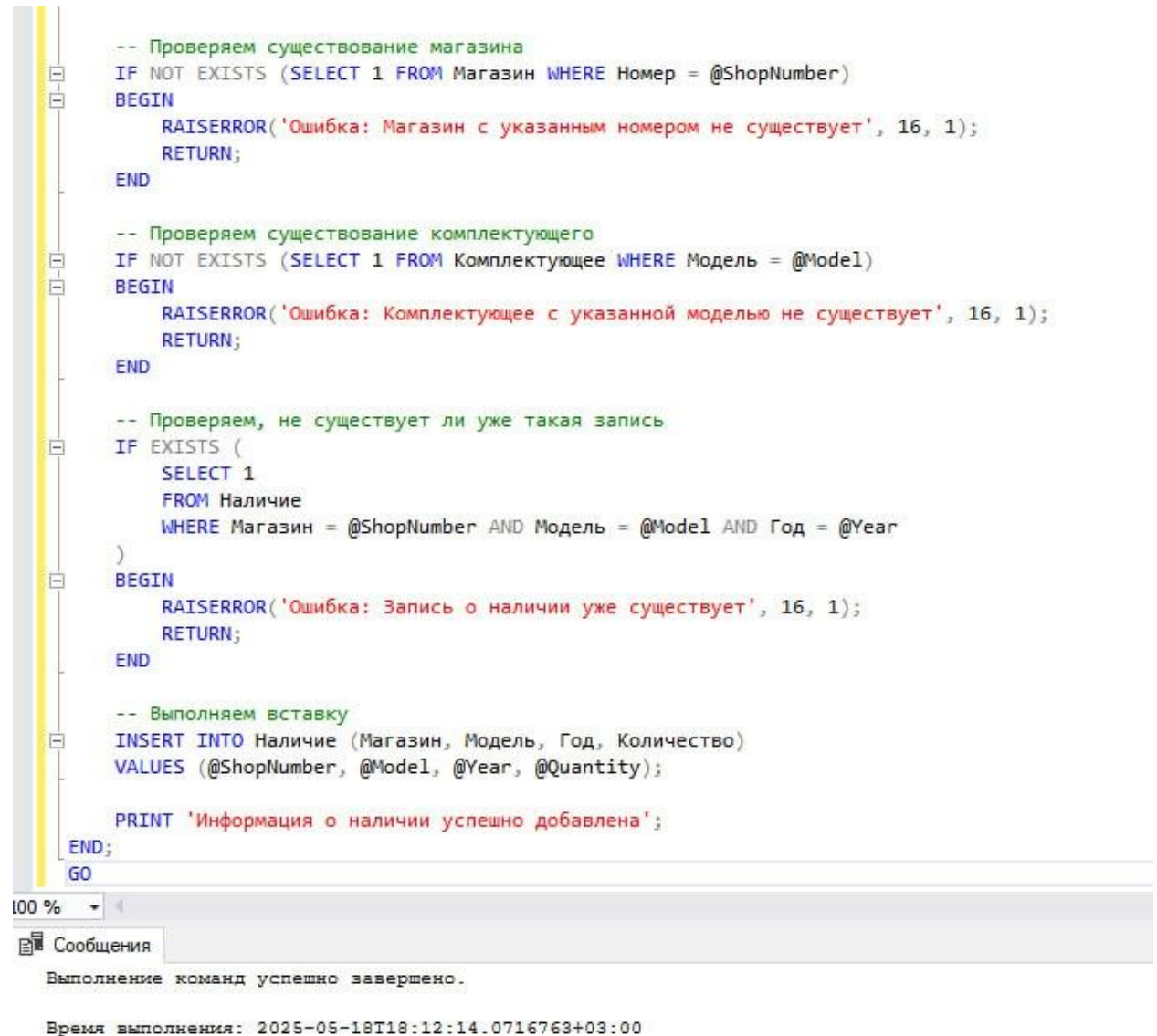
```
CREATE TRIGGER InsertInventoryThroughView
ON НаличиеКомплектующих
INSTEAD OF INSERT
AS
BEGIN
    -- Проверяем, что вставка только одной записи
    IF (SELECT COUNT(*) FROM inserted) > 1
    BEGIN
        RAISERROR('Ошибка: Можно вставлять только одну запись за раз', 16, 1);
        RETURN;
    END

    DECLARE @ShopNumber INT, @Model NVARCHAR(50), @Year INT, @Quantity INT;

    -- Получаем данные из вставки
    SELECT
        @ShopNumber = [Номер магазина],
        @Model = [Модель комплектующей],
        @Year = Год,
        @Quantity = Количество
    FROM inserted;
```

Рис.5

Разработала триггер InsertInventoryThroughView типа INSTEAD OF INSERT для представления !gqw:weKowzuekryhwwx, который реализует контролируемую вставку данных. Триггер ограничивает операции вставки одной записью за раз и извлекает параметры для последующей обработки. При попытке массовой вставки генерируется ошибка с пояснением.



```
-- Проверяем существование магазина
IF NOT EXISTS (SELECT 1 FROM Магазин WHERE Номер = @ShopNumber)
BEGIN
    RAISERROR('Ошибка: Магазин с указанным номером не существует', 16, 1);
    RETURN;
END

-- Проверяем существование комплектующего
IF NOT EXISTS (SELECT 1 FROM Комплектующее WHERE Модель = @Model)
BEGIN
    RAISERROR('Ошибка: Комплектующее с указанной моделью не существует', 16, 1);
    RETURN;
END

-- Проверяем, не существует ли уже такая запись
IF EXISTS (
    SELECT 1
    FROM Наличие
    WHERE Магазин = @ShopNumber AND Модель = @Model AND Год = @Year
)
BEGIN
    RAISERROR('Ошибка: Запись о наличии уже существует', 16, 1);
    RETURN;
END

-- Выполняем вставку
INSERT INTO Наличие (Магазин, Модель, Год, Количество)
VALUES (@ShopNumber, @Model, @Year, @Quantity);

PRINT 'Информация о наличии успешно добавлена';
END;
GO
```

100 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2025-05-18T18:12:14.0716763+03:00

Рис.6

Завершила разработку триггера InsertInventoryThroughView, реализующего безопасную вставку данных о наличии товаров через представление. Триггер выполняет многоуровневую проверку: существование магазина и комплектующего, отсутствие дубликатов записи. При успешной валидации данные добавляются в таблицу Наличие с подтверждающим сообщением."

Заключение:

В ходе выполненной работы я научилась создавать триггеры.