УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 4

по предмету

«Базы данных»

Выполнил:
Дубровинский А. В.
группа 751001

Проверил:
Салей О. А.

Минск 2020

Вариант 2

Задание №1

Работа с представлением VIEW. Изменение данных в таблице через представление. Создание AFTER DML триггера для таблицы. Логгирование изменений в history таблицу.

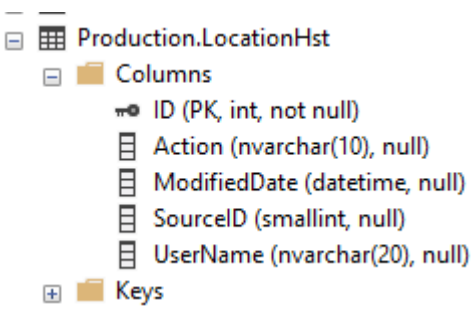
а) Создайте таблицу Production.LocationHst, которая будет хранить информацию об изменениях в таблице Production.Location.

Обязательные поля, которые должны присутствовать в таблице: ID — первичный ключ IDENTITY(1,1); Action — совершенное действие (insert, update или delete); ModifiedDate — дата и время, когда была совершена операция; SourceID — первичный ключ исходной таблицы; UserName — имя пользователя, совершившего операцию. Создайте другие поля, если считаете их нужными.

SQL запрос:

```
--a) Создайте таблицу Production.LocationHst, которая будет хранить информацию об изменениях в таблице Production.Location.
-- Обязательные поля, которые должны присутствовать в таблице: ID – первичный ключ IDENTITY(1,1);
-- Action – совершенное действие (insert, update или delete); ModifiedDate – дата и время, когда была совершена операция;
-- SourceID – первичный ключ исходной таблицы; UserName – имя пользователя, совершившего операцию.
-- Создайте другие поля, если считаете их нужными.

CREATE TABLE Production.LocationHst (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Action NVARCHAR(10),
    ModifiedDate DATETIME,
    SourceID SMALLINT,
    UserName NVARCHAR(20)
);
GO
```

Результат выполнения:

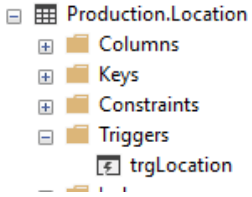


b) Создайте один AFTER триггер для трех операций INSERT, UPDATE, DELETE для таблицы Production.Location. Триггер должен заполнять таблицу Production.LocationHst с указанием типа операции в поле Action в зависимости от оператора, вызвавшего триггер.

SQL запрос:

```sql
-- b) Создайте один AFTER триггер для трех операций INSERT, UPDATE, DELETE для таблицы Production.Location.
-- Триггер должен заполнять таблицу Production.LocationHst с указанием типа операции в поле Action
-- в зависимости от оператора, вызвавшего триггер.

CREATE TRIGGER Production.trgLocation
ON Production.Location
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    IF EXISTS (SELECT * FROM inserted)
        AND EXISTS (SELECT * FROM deleted)
        INSERT INTO Production.LocationHst (
            Action,
            ModifiedDate,
            SourceID,
            UserName
        ) SELECT
            'update',
            CURRENT_TIMESTAMP,
            LocationID,
            CURRENT_USER
        FROM inserted

    ELSE IF EXISTS (SELECT * FROM inserted)
        INSERT INTO Production.LocationHst (
            Action,
            ModifiedDate,
            SourceID,
            UserName
        ) SELECT
            'insert',
            CURRENT_TIMESTAMP,
            LocationID,
            CURRENT_USER
        FROM inserted

    ELSE IF EXISTS (SELECT * FROM deleted)
        INSERT INTO Production.LocationHst (
            Action,
            ModifiedDate,
            SourceID,
            UserName
        ) SELECT
            'delete',
            CURRENT_TIMESTAMP,
            LocationID,
            CURRENT_USER
        FROM deleted;
END
GO
```

Результат выполнения:

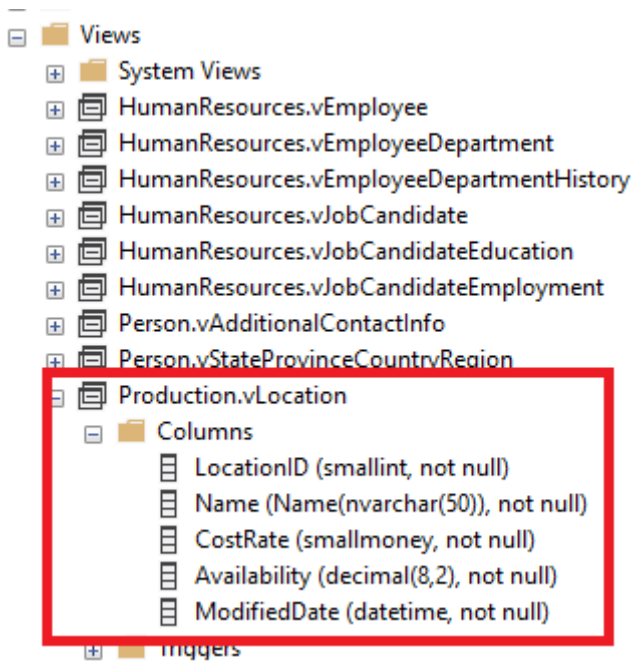


c) Создайте представление VIEW, отображающее все поля таблицы Production.Location.

SQL запрос:

```
-- c) Создайте представление VIEW, отображающее все поля таблицы Production.Location.

CREATE VIEW Production.vLocation AS
    SELECT * FROM Production.Location;
GO
```

Результат выполнения:



d) Вставьте новую строку в Production.Location через представление. Обновите вставленную строку. Удалите вставленную строку. Убедитесь, что все три операции отображены в Production.LocationHst.

SQL запрос:

```
-- d) Вставьте новую строку в Production.Location через представление. Обновите вставленную строку.
-- Удалите вставленную строку. Убедитесь, что все три операции отображены в Production.LocationHst.

SET IDENTITY_INSERT Production.Location ON;
GO

INSERT INTO Production.vLocation (
    LocationID,
    Name,
    CostRate,
    Availability,
    ModifiedDate
) VALUES (
    100,
    'SomeName',
    10.00,
    100.00,
    GETDATE()
);
GO

UPDATE Production.vLocation
    SET Name = 'OtherName'
    WHERE LocationID = 100;
GO

DELETE FROM Production.vLocation
    WHERE LocationID = 100;
GO

SELECT * FROM Production.LocationHst;
GO
```

Результат выполнения:

	ID	Action	ModifiedDate	SourceID	UserName
1	1	insert	2020-09-15 23:29:16.180	100	dbo
2	2	update	2020-09-15 23:29:16.190	100	dbo
3	3	delete	2020-09-15 23:29:16.240	100	dbo

Задание №2

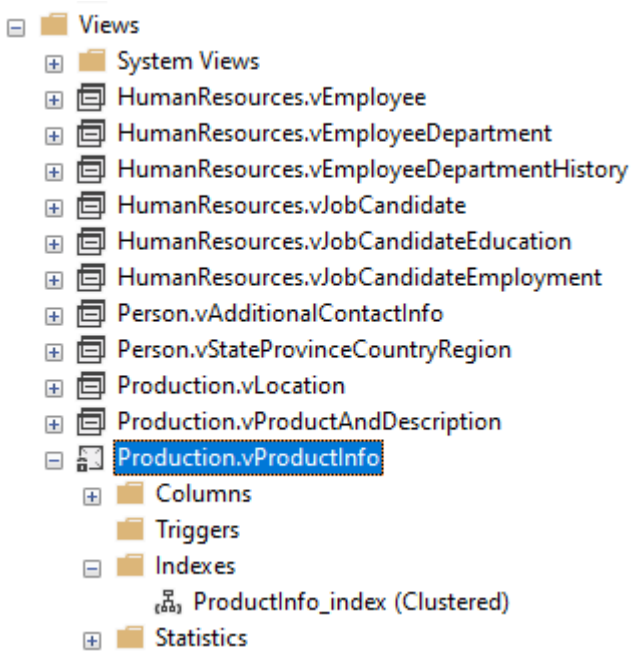
Индексированное представление. Создание AFTER DML триггера для представления.

а) Создайте представление VIEW, отображающее данные из таблиц Production.Location и Production.ProductInventory, а также Name из таблицы Production.Product. Сделайте невозможным просмотр исходного кода представления. Создайте уникальный кластерный индекс в представлении по полям LocationID,ProductID.

SQL запрос:

```
-- а) Создайте представление VIEW, отображающее данные из таблиц Production.Location и
-- Production.ProductInventory, а также Name из таблицы Production.Product.
-- Сделайте невозможным просмотр исходного кода представления.
-- Создайте уникальный кластерный индекс в представлении по полям LocationID, ProductID.

DROP VIEW IF EXISTS Production.vProductInfo;
GO

CREATE VIEW Production.vProductInfo
WITH SCHEMABINDING, ENCRYPTION AS
SELECT
    PrInv.ProductID,
    PrInv.LocationID,
    PrInv.Shelf,
    PrInv.Bin,
    PrInv.Quantity,
    PrInv.rowguid,
    PrInv.ModifiedDate AS ProductInventoryDate,
    Loc.Name AS LocationName,
    Loc.CostRate,
    Loc.Availability,
    Loc.ModifiedDate AS LocationDate,
    Pr.Name AS ProductName

    FROM Production.ProductInventory AS PrInv

    INNER JOIN Production.Location AS Loc
        ON PrInv.LocationID = Loc.LocationID

    INNER JOIN Production.Product AS Pr
        ON PrInv.ProductID = Pr.ProductID;
GO

CREATE UNIQUE CLUSTERED INDEX ProductInfo_index
    ON Production.vProductInfo(LocationID, ProductID);
GO
```

Результат выполнения:



b) Создайте три INSTEAD OF триггера для представления на операции INSERT, UPDATE, DELETE. Каждый триггер должен выполнять соответствующие операции в таблицах Production.Location и Production.ProductInventory для указанного Product Name. Обновление и удаление строк производите только в таблицах Production.Location и Production.ProductInventory, но не в Production.Product.

SQL запрос:

Триггер на вставку:

```
-- b) Создайте три INSTEAD OF триггера для представления на операции INSERT, UPDATE, DELETE.
-- Каждый триггер должен выполнять соответствующие операции в таблицах Production.Location и Production.ProductInventory
-- для указанного Product Name. Обновление и удаление строк производите только в таблицах Production.Location
-- и Production.ProductInventory, но не в Production.Product.

DROP TRIGGER IF EXISTS Production.instead_of_insert_trigger;
GO

CREATE TRIGGER Production.instead_of_insert_trigger
ON Production.vProductInfo
INSTEAD OF INSERT AS
BEGIN
    INSERT INTO Production.Location (
        Name,
        CostRate,
        Availability,
        ModifiedDate
    ) SELECT
        ins.LocationName,
        ins.CostRate,
        ins.Availability,
        ins.LocationDate
    FROM inserted AS ins
    INNER JOIN Production.Product AS Pr
        ON ins.ProductName = Pr.Name;

    INSERT INTO Production.ProductInventory (
        ProductID,
        LocationID,
        Shelf,
        Bin,
        Quantity,
        rowguid,
        ModifiedDate
    ) SELECT
        Pr.ProductID,
        Loc.LocationID,
        ins.Shelf,
        ins.Bin,
        ins.Quantity,
        ins.rowguid,
        ins.ProductInventoryDate
    FROM inserted AS ins

    INNER JOIN Production.Product AS Pr
        ON ins.ProductName = Pr.Name

    INNER JOIN Production.Location AS Loc
        ON ins.LocationName = Loc.Name;
END
GO
```

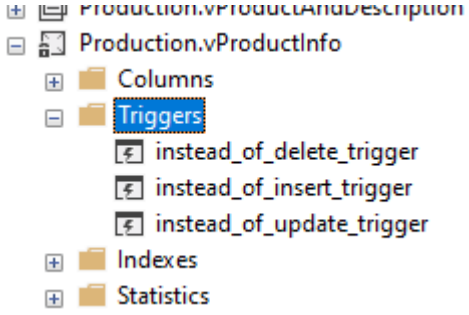
Триггер на обновление:

```
CREATE TRIGGER Production.instead_of_update_trigger
ON Production.vProductInfo
INSTEAD OF UPDATE AS
BEGIN
    UPDATE Production.Location SET
        Name = ins.LocationName,
        CostRate = ins.CostRate,
        Availability = ins.Availability,
        ModifiedDate = ins.LocationDate
    FROM Location AS Loc
    INNER JOIN inserted AS ins
        ON ins.LocationID = Loc.LocationID;

    UPDATE Production.ProductInventory SET
        Shelf = ins.Shelf,
        Bin = ins.Bin,
        Quantity = ins.Quantity,
        rowguid = ins.rowguid,
        ModifiedDate = ins.ProductInventoryDate
    FROM ProductInventory AS PrInv
    INNER JOIN inserted AS ins
        ON ins.ProductID = PrInv.ProductID;
END
GO
```

Триггер на удаление:

```
CREATE TRIGGER Production.instead_of_delete_trigger
ON Production.vProductInfo
INSTEAD OF DELETE AS
BEGIN
    DECLARE @prodID INT;

    SET @prodID = (SELECT ProductID FROM deleted);
    CREATE TABLE #locations (
        LocationID SMALLINT NOT NULL
    );

    INSERT INTO #locations
    SELECT DISTINCT PrInv.LocationID
    FROM Production.ProductInventory AS PrInv

    INNER JOIN deleted
        ON deleted.ProductID = PrInv.ProductID

    WHERE PrInv.LocationID NOT IN (
        SELECT DISTINCT ppi.LocationID
        FROM Production.ProductInventory as ppi
        WHERE ppi.ProductID != @prodID
    );

    DELETE PrInv
    FROM Production.ProductInventory AS PrInv
        WHERE PrInv.ProductID = @prodID;

    DELETE Loc
    FROM Production.Location AS Loc
        WHERE LocationID IN (SELECT * FROM #locations);
END
GO
```

Результат выполнения:



c) Вставьте новую строку в представление, указав новые данные для Location и ProductInventory, но для существующего Product (например для ‘Adjustable Race’). Триггер должен добавить новые строки в таблицы Production.Location и Production.ProductInventory для указанного Product Name. Обновите вставленные строки через представление. Удалите строки.

SQL запрос (вставка):

```
-- c) Вставьте новую строку в представление, указав новые данные для Location и ProductInventory,
-- но для существующего Product (например для ‘Adjustable Race’). Триггер должен добавить новые строки
-- в таблицы Production.Location и Production.ProductInventory для указанного Product Name.
-- Обновите вставленные строки через представление. Удалите строки.

INSERT INTO Production.vProductInfo
(
    Shelf,
    Bin,
    Quantity,
    rowguid,
    LocationDate,
    ProductInventoryDate,
    LocationName,
    CostRate,
    Availability,
    ProductName
) VALUES (
    'A',
    3,
    200,
    '99944D7D-CAF5-46B3-AB22-5718DCC26B5E',
    GETDATE(),
    GETDATE(),
    'Window',
    '10000',
    1000.00,
    'Adjustable Race'
);
GO

SELECT * FROM Production.Location;
GO

SELECT * FROM Production.ProductInventory;
GO
```

Результаты выполнения:

Production.Location:

Results Messages

	LocationID	Name	CostRate	Availability	ModifiedDate
1	2	Sheet Metal Racks	0,00	0.00	2002-06-01 00:00:00.000
2	3	Paint Shop	0,00	0.00	2002-06-01 00:00:00.000
3	4	Paint Storage	0,00	0.00	2002-06-01 00:00:00.000
4	5	Metal Storage	0,00	0.00	2002-06-01 00:00:00.000
5	6	Miscellaneous St...	0,00	0.00	2002-06-01 00:00:00.000
6	7	Finished Goods S...	0,00	0.00	2002-06-01 00:00:00.000
7	10	Frame Forming	22,50	96.00	2002-06-01 00:00:00.000
8	20	Frame Welding	25,00	108.00	2002-06-01 00:00:00.000
9	30	Debur and Polish	14,50	120.00	2002-06-01 00:00:00.000
10	40	Paint	15,75	120.00	2002-06-01 00:00:00.000
11	45	Specialized Paint	18,00	80.00	2002-06-01 00:00:00.000
12	50	Subassembly	12,25	120.00	2002-06-01 00:00:00.000
13	60	Final Assembly	12,25	120.00	2002-06-01 00:00:00.000
14	1002	Window	10000,00	1000.00	2020-09-23 01:39:29.487

Production.ProductInventory:

Results Messages

	ProductID	LocationID	Shelf	Bin	Quantity	rowguid	ModifiedDate
1	1	1002	A	3	200	99944D7D-CAF5-46B3-AB22-5718DCC26B5E	2020-09-23 01:39:29.487
2	316	5	A	11	532	1EE3DBD3-2A7E-47DC-AF99-1B585575EFB9	2002-05-02 00:00:00.000
3	316	10	B	1	388	CB2A24D7-9B70-4140-8836-9EB7592621D3	2002-05-02 00:00:00.000
4	317	5	A	1	158	83332A73-48A9-401D-95F4-385C944D716F	2008-09-09 00:00:00.000

SQL запрос (обновление):

```
UPDATE Production.vProductInfo
    SET CostRate = '555',
        LocationName = 'NewLocation'

    WHERE ProductName = 'Adjustable Race';
GO

SELECT * FROM Production.Location AS Loc
    INNER JOIN Production.ProductInventory AS PrInv
        ON Loc.LocationID = PrInv.LocationID
    WHERE CostRate = '555';
GO
```

Результат выполнения запроса:

	LocationID	Name	CostRate	Availability	ModifiedDate	ProductID	LocationID	Shelf	Bin	Quantity	rowguid	ModifiedDate
1	1002	NewLocation	555,00	1000.00	2020-09-23 01:39:29.487	1	1002	A	3	200	99944D7D-CAF5-46B3-AB22-5718DCC26B5E	2020-09-23 01:39:29.487

SQL запрос (удаление):

```
DELETE FROM Production.vProductInfo
    WHERE ProductName = 'Adjustable Race';
GO

SELECT * FROM Production.Location L
    INNER JOIN Production.ProductInventory I
        ON L.LocationID = I.LocationID
    INNER JOIN Production.Product P
        ON P.ProductID = I.ProductID
    WHERE P.Name = 'Adjustable Race';
GO
```

Результат выполнения запроса: