

Лабораторная работа №4

Задание №1

Вариант 2

- 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(8) NOT NULL,  
    ModifiedDate DATETIME NOT NULL,  
    SourceID INT NOT NULL,  
    UserName NVARCHAR(25) NOT NULL  
);  
GO
```

Production.Instruction
Production.Location
Production.LocationHst
Production.Product

- 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_Location_TR  
ON Production.Location  
AFTER INSERT, UPDATE, DELETE  
AS  
IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)  
    INSERT INTO Production.LocationHst  
    SELECT  
        'update',  
        CURRENT_TIMESTAMP,  
        LocationID,  
        CURRENT_USER  
    FROM inserted  
ELSE IF EXISTS (SELECT * FROM inserted)  
    INSERT INTO Production.LocationHst  
    SELECT  
        'insert',  
        CURRENT_TIMESTAMP,  
        LocationID,  
        CURRENT_USER  
    FROM inserted  
ELSE IF EXISTS (SELECT * FROM deleted)  
    INSERT INTO Production.LocationHst  
    SELECT  
        'delete',  
        CURRENT_TIMESTAMP,  
        LocationID,  
        CURRENT_USER  
    FROM deleted;  
GO
```

- [-] Production.Location
 - [+] Columns
 - [+] Keys
 - [+] Constraints
 - [+] Triggers
 - [-] Production_Location_TR

```

INSERT INTO Production.Location (
    LocationID,
    Name,
    CostRate,
    Availability,
    ModifiedDate
) VALUES (
    666,
    'DB',
    12,
    0.6,
    CURRENT_TIMESTAMP
);
GO

```

Messages (1 row affected)

(1 row affected)

```

SELECT * FROM Production.LocationHst
WHERE SourceID = 666;
GO

```

Results		Messages			
	ID	Action	ModifiedDate	SourceID	UserName
1	2	insert	2020-09-18 13:54:14.493	666	dbo

```

UPDATE Production.Location
SET Name = 'db'
WHERE LocationID = 666;
GO

```

Messages (1 row affected)

(1 row affected)

```

SELECT * FROM Production.LocationHst
WHERE SourceID = 666;
GO

```

Results		Messages			
	ID	Action	ModifiedDate	SourceID	UserName
1	2	insert	2020-09-18 13:54:14.493	666	dbo
2	4	update	2020-09-18 14:08:58.807	666	dbo

```

DELETE FROM Production.Location
WHERE LocationID = 666;
GO

```

Messages (1 row affected)

(1 row affected)

```

SELECT * FROM Production.LocationHst
WHERE SourceID = 666;
GO

```

Results		Messages			
	ID	Action	ModifiedDate	SourceID	UserName
1	2	insert	2020-09-18 13:54:14.493	666	dbo
2	4	update	2020-09-18 14:08:58.807	666	dbo
3	5	delete	2020-09-18 14:11:00.240	666	dbo

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

```

CREATE VIEW LocationView AS
SELECT * FROM Production.Location;
GO

```

Views

- [+] System Views
- [+] dbo.LocationView

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 LocationView (
    LocationID,
    Name,
    CostRate,
    Availability,
    ModifiedDate
) VALUES (
    666,
    'DBo',
    12,
    0.6,
    CURRENT_TIMESTAMP
);
GO
```

Messages

(1 row affected)

```
SET IDENTITY_INSERT Production.Location OFF;
GO
```

(1 row affected)

```
SELECT * FROM LocationView
WHERE LocationID = 666;
GO
```

Results		Messages			
	LocationID	Name	CostRate	Availability	ModifiedDate
1	666	DBo	12.00	0.60	2020-09-18 14:33:25.440

```
UPDATE LocationView
SET Name = 'dbo'
WHERE LocationID = 666;
GO
```

Messages

(1 row affected)

(1 row affected)

Messages

```
DELETE FROM LocationView
WHERE LocationID = 666;
GO
```

(1 row affected)

(1 row affected)

```
SELECT * FROM Production.LocationHst
WHERE SourceID = 666;
GO
```

Results		Messages			
	ID	Action	ModifiedDate	SourceID	UserName
1	2	insert	2020-09-18 13:54:14.493	666	dbo
2	4	update	2020-09-18 14:08:58.807	666	dbo
3	5	delete	2020-09-18 14:11:00.240	666	dbo
4	9	insert	2020-09-18 14:33:25.447	666	dbo
5	10	update	2020-09-18 14:35:42.200	666	dbo
6	11	delete	2020-09-18 14:36:15.650	666	dbo

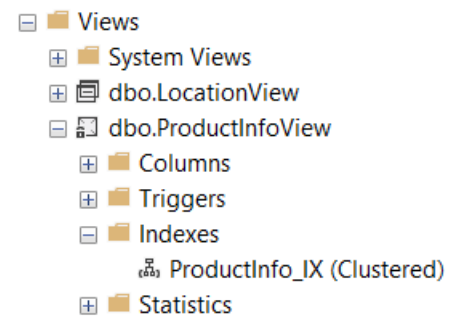
Задание №2

Вариант 2

- a) 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.

```
CREATE VIEW ProductInfoView
WITH SCHEMABINDING, ENCRYPTION AS
SELECT
    l.LocationID,
    l.Name AS LocationName,
    l.CostRate,
    l.Availability,
    l.ModifiedDate AS LocationModifiedDate,
    ppi.ProductID,
    ppi.Shelf,
    ppi.Bin,
    ppi.Quantity,
    ppi.rowguid,
    ppi.ModifiedDate AS ProductInventoryModifiedDate,
    p.Name
FROM Production.Location AS l
INNER JOIN Production.ProductInventory AS ppi
ON l.LocationID = ppi.LocationID
INNER JOIN Production.Product AS p
ON ppi.ProductID = p.ProductID;
GO

CREATE UNIQUE CLUSTERED INDEX ProductInfo_IX
ON ProductInfoView(LocationID, ProductID);
GO
```



- 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.

```

CREATE TRIGGER ProductInfo_Ins_TR
ON ProductInfoView
INSTEAD OF INSERT AS
BEGIN
    INSERT INTO Production.Location
    SELECT
        LocationName,
        CostRate,
        Availability,
        LocationModifiedDate
    FROM inserted
    INNER JOIN Production.Product AS p
    ON inserted.Name = p.Name;
    INSERT INTO Production.ProductInventory
    SELECT
        p.ProductID,
        l.LocationID,
        Shelf,
        Bin,
        Quantity,
        inserted.rowguid,
        ProductInventoryModifiedDate
    FROM inserted
    INNER JOIN Production.Product AS p
    ON inserted.Name = p.Name
    INNER JOIN Production.Location AS l
    ON inserted.LocationName = l.Name;
END;
CREATE TRIGGER ProductInfo_Upd_TR
ON ProductInfoView
INSTEAD OF UPDATE AS
BEGIN
    IF UPDATE(LocationID) OR UPDATE(ProductID)
    BEGIN
        RAISERROR ('UPDATE of Primary Key through ProductInfoView is prohibited.', 16, 1);
        ROLLBACK;
    END
    ELSE
    BEGIN
        UPDATE Production.Location
        SET
            Name = inserted.LocationName,
            CostRate = inserted.CostRate,
            Availability = inserted.Availability,
            ModifiedDate = inserted.LocationModifiedDate
        FROM Production.Location AS l
        INNER JOIN inserted
        ON inserted.LocationID = l.LocationID;
        UPDATE Production.ProductInventory
        SET
            Shelf = inserted.Shelf,
            Bin = inserted.Bin,
            Quantity = inserted.Quantity,
            rowguid = inserted.rowguid,
            ModifiedDate = inserted.ProductInventoryModifiedDate
        FROM Production.ProductInventory AS ppi
        INNER JOIN inserted
        ON ppi.ProductID = inserted.ProductID;
    END;
END;
GO

```

```

CREATE TRIGGER ProductInfo_Del_TR
ON ProductInfoView
INSTEAD OF DELETE AS
BEGIN
    DECLARE @pID INT;
    SELECT @pID = (SELECT ProductID FROM deleted);
    CREATE TABLE #locations (
        LocationID SMALLINT NOT NULL
    );
    INSERT INTO #locations
    SELECT DISTINCT p.LocationID
    FROM Production.ProductInventory AS p
    INNER JOIN deleted
    ON deleted.ProductID = p.ProductID
    WHERE p.LocationID NOT IN (
        SELECT DISTINCT ppi.LocationID
        FROM Production.ProductInventory as ppi
        WHERE ppi.ProductID != @pID
    );
    DELETE p
    FROM Production.ProductInventory AS p
    WHERE p.ProductID = @pID;
    DELETE l
    FROM Production.Location AS l
    WHERE LocationID IN (SELECT * FROM #locations);
END;

```

Views
System Views
dbo.LocationView
dbo.ProductInfoView
Columns
Triggers
ProductInfo_Del_TR
ProductInfo_Ins_TR
ProductInfo_Upd_TR

- 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 ProductInfoView (
    LocationName,
    CostRate,
    Availability,
    LocationModifiedDate,
    Shelf,
    Bin,
    Quantity,
    rowguid,
    ProductInventoryModifiedDate,
    Name
) VALUES (
    'SHELF',
    66.6,
    0.6,
    CURRENT_TIMESTAMP,
    'A',
    6,
    666,
    '47A24246-6C43-48EB-968F-025738A8410',
    CURRENT_TIMESTAMP,
    'Blade'
);
GO

```

```

SELECT * FROM Production.Location
WHERE Name = 'SHELF';
GO

```

Results

Messages

	LocationID	Name	CostRate	Availability	ModifiedDate
1	6671	SHELF	66,60	0.60	2020-09-20 16:51:25.980

```

SELECT TOP 2 * FROM Production.ProductInventory
ORDER BY ModifiedDate DESC;
GO

```

Results

Messages

	ProductID	LocationID	Shelf	Bin	Quantity	rowguid	ModifiedDate
1	316	6671	A	6	666	47A24246-6C43-48EB-968F-025738A8A410	2020-09-20 16:51:25.980
2	325	1	H	1	569	624AC935-868E-40E0-8668-950451746F90	2008-09-12 00:00:00.000

```

UPDATE ProductInfoView
SET LocationID = 3
WHERE Name = 'Blade';

```

Messages	
Msg 50000, Level 16, State 1, Procedure ProductInfo_Upd_TR, Line 7 [Batch Start Line 138] UPDATE of Primary Key through ProductInfoView is prohibited.	
Msg 3609, Level 16, State 1, Line 140 The transaction ended in the trigger. The batch has been aborted.	

```

UPDATE ProductInfoView
SET CostRate = 8.8
WHERE Name = 'Blade';
GO

```

```

SELECT ppi.LocationID, p.Name, CostRate
FROM Production.Location
INNER JOIN Production.ProductInventory AS ppi
ON ppi.LocationID = Production.Location.LocationID
INNER JOIN Production.Product AS p
ON p.ProductID = ppi.ProductID
WHERE p.Name = 'Blade';

```

Results		Messages	
	LocationID	Name	CostRate
1	5	Blade	8.80
2	10	Blade	8.80
3	50	Blade	8.80
4	6671	Blade	8.80

```

UPDATE ProductInfoView
SET Quantity = 8
WHERE Name = 'Blade';
GO

```

```

SELECT Name, Quantity
FROM Production.ProductInventory AS ppi
INNER JOIN Production.Product AS p
ON ppi.ProductID = p.ProductID
WHERE Name = 'Blade';

```

Results		Messages	
	Name	Quantity	
1	Blade	8	
2	Blade	8	
3	Blade	8	
4	Blade	8	

```

DELETE FROM ProductInfoView
WHERE Name = 'Blade';
GO

```

```

SELECT COUNT(*) AS recCount
FROM Production.Location AS l
INNER JOIN Production.ProductInventory AS ppi
ON ppi.LocationID = l.LocationID
INNER JOIN Production.Product AS p
ON p.ProductID = ppi.ProductID
WHERE p.Name = 'Blade';

```

Results		Messages	
	recCount		
1	0		