доц. д-р Цветанка Георгиева-Трифонова

ТРИГЕРИ

СЪДЪРЖАНИЕ

- Създаване и използване на тригери
- Каскадни и рекурсивни тригери
- Използване на тригери за изпълняване на действия за запазване на целостността на данните

- *Тригерът*
  - специален вид съхранена процедура, която се стартира автоматично от СУБД при опит за модифициране на данните, с които тригерът е свързан;
  - може да бъде настроен да се стартира, когато данните бъдат променяни по някакъв начин, т.е. чрез конструкцията INSERT, UPDATE и/или DELETE;
  - изпълнява се по веднъж за всяка конструкция INSERT, UPDATE или DELETE независимо от броя на редовете, които засяга тя (0, 1 или повече).

ТРИГЕРИ

- *Предназначение*
  - осигуряване на начин за ограничаване на стойностите на данните;
  - реализиране на актуализации, включващи промени в няколко таблици;
  - проследяване на промените в данните в таблиците на базата от данни.

СЪЗДАВАНЕ НА ТРИГЕРИ

```
CREATE TRIGGER trigger_name
 ON {table_name | view_name}
{FOR | AFTER | INSTEAD OF}
[DELETE] [,] [INSERT] [,] [UPDATE]
 AS
   sql_queries
```

СЪЗДАВАНЕ НА ТРИГЕРИ

- Два типа тригери:
  - INSTEAD OF – изпълняват се вместо модификациите, указани в тяхната дефиниция. За всяко действие може да се дефинира най-много един тригер INSTEAD OF за дадена таблица или изглед.
  - AFTER (или FOR) – задействат се, след като данните бъдат модифицирани, т.е. добавени, изтрити или променени. За всяко действие могат да се дефинират повече от един тригери AFTER за дадена таблица. Този вид тригери не могат да се създават за изгледи.

СЪЗДАВАНЕ НА ТРИГЕРИ

- Тригерът AFTER и ROLLBACK TRANSACTION
  - стартира се, след като конструкцията за модифициране на данни приключи работата си, но преди тази модификация да бъде потвърдена в базата данни;
  - както конструкцията, така и всякакви модификации, извършени в тригера, се третират неявно като транзакция;
  - поради това тригерът може да превърти назад модификацията на данните, предизвикали стартирането му чрез ROLLBACK TRANSACTION.

СЪЗДАВАНЕ НА ТРИГЕРИ

- Псевдотаблици *inserted* и *deleted:*
  - имат същия набор **колони** като основната таблица, върху която се извършват промените и толкова **редове**, колкото са засегнати от съответната конструкция за вмъкване, изриване или променяне на данни;
  - таблицата *inserted* съдържа редовете, повлияни от модификацията, т.е. вмъкнати или променени и то техния краен образ;
  - таблицата *deleted* съдържа редовете, засегнати от съответната конструкция за променяне или изтриване и то техния първоначален образ.

СЪЗДАВАНЕ НА ТРИГЕРИ

- Дефиниране на първи и последен тригер AFTER:

```
sp_settriggerorder
    [@triggername = ] 'trigger_name',
    [@order = ] 'value',
    [@stmttype = ] 'statement_type'
```

- @triggername определя името на тригера AFTER;
- @order може да бъде: *First*, *Last*, *None* и указва кой тригер AFTER да се стартира първи и кой последен от всички тригери от този вид за дадена таблица. Останалите тригери AFTER в таблицата се изпълняват в произволен ред.
- @stmttype определя коя SQL конструкция задейства тригера. Възможните стойности са DELETE, INSERT или UPDATE.

ЗАДАВАНЕ НА ПЪРВИ ТРИГЕР AFTER – ПРИМЕР

```
EXEC sp_settriggerorder
     @triggername = 'MyTrigger',
     @order = 'First',
     @stmttype = 'UPDATE'
```

- Забраняване и разрешаване на тригери

```
ALTER TABLE table_name
{DISABLE | ENABLE} TRIGGER
    {ALL | trigger_name [,…]}
```

- Изтриване на тригер

```
DROP TRIGGER trigger_name
```

- Променяне на тригер

```
ALTER TRIGGER trigger_name
   ON {table_name | view_name}
 {FOR | AFTER | INSTEAD OF}
[DELETE] [,] [INSERT] [,] [UPDATE]
 AS
   sql_queries
```

СЪЗДАВАНЕ НА ТРИГЕРИ – ПРИМЕР

- Тригер, който забранява изтриването на редове в таблицата, съдържаща данните за клиентите:

```
CREATE TRIGGER No_DeleteCustomers
 ON Customers
 INSTEAD OF DELETE
AS
   IF @@ROWCOUNT = 0 RETURN

   RAISERROR('Не е разрешено изтриване на
              редове в таблицата.', 11, 1)

-- стартиране на тригера:
DELETE FROM Customers
WHERE CustomerID = 1
```

СЪЗДАВАНЕ НА ТРИГЕРИ

- IF UPDATE(*column_name*)
  - изпълняване на определени действия в тригера, ако промените са отразили точно определени колони.
- Пример
  - Тригер, който изчислява отстъпката на дадена по идентификатора си продажба, т.е. изчислява и актуализира стойността на колоната Discount в таблицата за продажбите Sales в зависимост от стойностите на TotalForSale и CurrentBalance:

```
CREATE TRIGGER SalesAffectsDiscount
  ON Sales
  AFTER INSERT, UPDATE
AS
 IF @@ROWCOUNT = 0 RETURN

 IF UPDATE(TotalForSale)
  BEGIN
   UPDATE Sales
      SET Discount =
          CASE
            WHEN i.TotalForSale >= 100
              AND c.CurrentBalance >= 1000 THEN 0.15
            WHEN i.TotalForSale >= 100 THEN 0.10
            WHEN c.CurrentBalance >= 1000 THEN 0.12
            ELSE 0
            -- ако се пропусне, се въвежда стойност NULL
          END
   FROM Sales s
   INNER JOIN inserted i ON s.SaleID = i.SaleID
   INNER JOIN Customers c ON s.CustomerID = c.CustomerID
  END
```

- Стартиране на тригера *SalesAffectsDiscount*:

```
INSERT INTO Sales
  (CustomerID, EmployeeID, SaleDate, TotalForSale)
VALUES (2, 4, GetDate(), 250.30)
```

- Примерен резултат от:

```
SELECT * FROM Sales
```

	SaleID	CustomerID	EmployeeID	SaleDate	TotalForSale	Discount
1	1	1	1	2010-01-30 09:08:00.000	8.75	0.00
2	2	3	1	2010-01-30 10:08:00.000	42.00	0.00
3	3	2	3	2010-01-31 12:08:00.000	1.80	0.00
4	4	2	4	2010-01-31 13:08:00.000	10.10	0.00
5	5	1	2	2009-10-06 00:00:00.000	0.00	0.00
6	6	1	2	2009-10-06 00:00:00.000	0.00	0.00
7	8	1	3	2010-04-16 20:00:34.110	125.50	0.10
8	9	2	4	2010-04-16 20:02:51.153	250.30	0.10

СЪЗДАВАНЕ НА ТРИГЕРИ – ПРИМЕР

- Тригер, който забранява продажбата на продукти, определени в таблицата `Products` като продукти с преустановена продажба:

	ProductID	ProductName	CategoryID	SupplierID	Price	Stock	ReorderLevel	Discontinued
1	1	ябълки	1	1	2.50	120.000	10.000	0
2	2	портокали	1	1	3.50	90.000	10.000	0
3	3	домати	1	1	3.50	100.000	15.000	0
4	4	картофи	1	1	3.00	55.000	20.000	0
5	5	чипс	2	3	1.20	20.000	5.000	1
6	6	царевичен снакс	2	2	1.50	24.000	5.000	0
7	7	макарони	3	2	2.60	10.000	12.000	0
8	8	еклери	5	2	1.20	52.000	5.000	0

	SaleID	ProductID	Quantity	Price	Discount
1	1	1	2.000	3.00	0.00
2	1	3	1.000	2.75	0.00
3	2	2	5.000	4.00	0.10
4	2	4	2.000	3.50	0.00
5	2	9	10.000	2.60	0.00
6	3	5	1.000	1.80	0.00
7	4	7	2.000	2.80	0.00
8	4	8	3.000	1.50	0.00

```
INSERT INTO SaleDetails
       (SaleID, ProductID, Quantity, Price)
VALUES (1, 5, 3, 4.5)
```

СЪЗДАВАНЕ НА ТРИГЕРИ – ПРИМЕР

- Тригер, който забранява продажбата на продукти, определени в таблицата `Products` като продукти с преустановена продажба:

```
CREATE TRIGGER SaleDetailNotDiscontinued
   ON SaleDetails
   AFTER INSERT, UPDATE
AS
 IF @@ROWCOUNT = 0 RETURN

 IF UPDATE ( ProductID )
  BEGIN
    IF EXISTS
   ( SELECT * FROM Inserted I
     INNER JOIN Products p ON i.ProductID = p.ProductID
     WHERE p.Discontinued = 1 )
      BEGIN
       RAISERROR('Продуктът е с преустановена
            продажба. Транзакцията е отменена.', 16, 1)
       ROLLBACK TRANSACTION
      END
  END
```

КАСКАДНИ И РЕКУРСИВНИ ТРИГЕРИ

- Каскадни тригери
  - `EXEC sp_configure 'nested triggers', 1`
    - ако тригерът модифицира данни в някоя друга таблица и тя има тригер, той ще бъде стартиран;
    - опцията разрешава тригерите да се извикват *каскадно* до максимална последователност от 32;
- Рекурсивни тригери
  - `EXEC sp_dboption 'database_name', 'recursive triggers', TRUE`
    - ако тригерът модифицира данни в същата таблица, в която той съществува, използването на същата операция стартира тригера отново.

ИЗПОЛЗВАНЕ НА ТРИГЕРИ ЗА ИЗПЪЛНЯВАНЕ НА ДЕЙСТВИЯ ЗА ЗАПАЗВАНЕ НА ЦЕЛОСТНОСТТА НА ДАННИТЕ

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name]
    [FOREIGN KEY (col_name1 [, …])]
    REFERENCES ref_table_name (ref_col_name1
  [, …])
    [ON UPDATE {CASCADE | NO ACTION}]
    [ON DELETE {CASCADE | NO ACTION}]
```

- при необходимост от изпълняване на допълнителни действия към стандартните действия, поддържащи целостността на данните, ще трябва да се създаде тригер;
- Например след изтриване на редовете от свързаните таблици да се извърши добавяне на изтритите редове в друга таблица за по-късен преглед.

	SaleID	CustomerID	EmployeeID	SaleDate	TotalForSale	Discount
1	1	1	1	2010-01-30 09:08:00.000	8.75	0.00
2	2	3	1	2010-01-30 10:08:00.000	42.00	0.00
3	3	2	3	2010-01-31 12:08:00.000	1.80	0.00
4	4	2	4	2010-01-31 13:08:00.000	10.10	0.00
5	5	1	2	2009-10-06 00:00:00.000	0.00	0.00
6	6	1	2	2009-10-06 00:00:00.000	0.00	0.00
7	8	1	3	2010-04-16 20:00:34.110	125.50	0.10
8	9	2	4	2010-04-16 20:02:51.153	250.30	0.10

	SaleID	ProductID	Quantity	Price	Discount
1	1	1	2.000	3.00	0.00
2	1	3	1.000	2.75	0.00
3	2	2	5.000	4.00	0.10
4	2	4	2.000	3.50	0.00
5	2	9	10.000	2.60	0.00
6	3	5	1.000	1.80	0.00
7	4	7	2.000	2.80	0.00
8	4	8	3.000	1.50	0.00

```
DELETE FROM Sales
WHERE SaleID = 1
```

```
CREATE TRIGGER DelCascadeSale
       ON Sales
       AFTER DELETE
 AS
   DECLARE @i int, @j int

   SET @i = @@ROWCOUNT

   IF @i = 0 RETURN

   INSERT INTO SalesHistory
    (SaleID, CustomerID, EmployeeID, SaleDate,
     TotalForSale, DiscountFromTotal, ProductID, Price,
     Quantity, DiscountFromPrice)
   SELECT d.SaleID, d.CustomerID, d.EmployeeID,
       d.SaleDate, d.TotalForSale, d.Discount,
       sd.ProductID, sd.Price, sd.Quantity, sd.Discount
   FROM SaleDetails sd
   RIGHT JOIN deleted d ON sd.SaleID = d.SaleID
```

```
DELETE FROM SaleDetails
FROM SaleDetails sd
INNER JOIN deleted d ON sd.SaleID = d.SaleID

SET @j = @@ROWCOUNT

IF @i > 0 OR @j > 0
    RAISERROR('%d реда от SaleDetails са
      изтрити като резултат от
      изтриването на %d реда в
      таблицата Sales и са добавени в
      SalesHistory.', 11, 1, @j, @i)
```

- Стартиране на тригера

```
DELETE FROM Sales
WHERE SaleID = 2
```

- Забраняване на всякакви вмъквания и обновявания, ако външният ключ в реферираща таблица няма съответствие в първичния ключ в реферираната таблица:

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	PhoneNumber	FaxNumber	EmailAddress
1	1	Алтметал ЕООД	Иван Иванов	мениджър	ул. Васил Левски, 12	Велико Търново	5000	123456	98765544	ivan@gmail.com
2	2	Деница Стар ЕООД	Петър Иванов	мениджър	бул. България, 34	Велико Търново	5000	565656	3453645765	peter@gmail.com
3	3	МИЛСТОР 90 ЕООД	Петър Петров	мениджър	ул. Страцин, 67	Велико Търново	5000	5465465	456566	petrov@gmail.com
4	4	Мики-92 ЕООД	Христо Димитров	мениджър	ул. Г. Козарев, 7	Велико Търново	5000	454545	6766655	hristo@gmail.com

	SaleID	CustomerID	EmployeeID	SaleDate	TotalForSale	Discount
1	1	1	1	2010-01-30 09:08:00.000	8.75	0.00
2	2	3	1	2010-01-30 10:08:00.000	42.00	0.00
3	3	2	3	2010-01-31 12:08:00.000	1.80	0.00
4	4	2	4	2010-01-31 13:08:00.000	10.10	0.00
5	5	1	2	2009-10-06 00:00:00.000	0.00	0.00
6	6	1	2	2009-10-06 00:00:00.000	0.00	0.00
7	8	1	3	2010-04-16 20:00:34.110	125.50	0.10
8	9	2	4	2010-04-16 20:02:51.153	250.30	0.10

```
INSERT INTO Sales
       (CustomerID, EmployeeID, SaleDate)
VALUES (50, 2, GetDate())
```

- Забраняване на всякакви вмъквания и обновявания, ако външният ключ в реферираща таблица няма съответствие в първичния ключ в реферираната таблица:

```
CREATE TRIGGER SaleHasCustomer
    ON Sales
    FOR INSERT, UPDATE
AS
    IF @@ROWCOUNT = 0 RETURN

    IF EXISTS
       ( SELECT * FROM inserted i
         WHERE i.CustomerID NOT IN
             (SELECT c.CustomerID FROM Customers c)
        )
    BEGIN
       RAISERROR('Продажбата трябва да има валиден
                  CustomerID.', 16, 1)
       ROLLBACK TRANSACTION
    END
```

- Забраняване на изтриване на ред от реферираната таблица, ако той има съответни редове в реферираща таблица:

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	PhoneNumber	FaxNumber	EmailAddress
1	1	Алтметал ЕООД	Иван Иванов	мениджър	ул. Васил Левски, 12	Велико Търново	5000	123456	98765544	ivan@gmail.com
2	2	Деница Стар ЕООД	Петър Иванов	мениджър	бул. България, 34	Велико Търново	5000	565656	3453645765	peter@gmail.com
3	3	МИЛСТОР 90 ЕООД	Петър Петров	мениджър	ул. Страцин, 67	Велико Търново	5000	5465465	456566	petrov@gmail.com
4	4	Мики-92 ЕООД	Христо Димитров	мениджър	ул. Г. Козарев, 7	Велико Търново	5000	454545	6766655	hristo@gmail.com

	SaleID	CustomerID	EmployeeID	SaleDate	TotalForSale	Discount
1	1	1	1	2010-01-30 09:08:00.000	8.75	0.00
2	2	3	1	2010-01-30 10:08:00.000	42.00	0.00
3	3	2	3	2010-01-31 12:08:00.000	1.80	0.00
4	4	2	4	2010-01-31 13:08:00.000	10.10	0.00
5	5	1	2	2009-10-06 00:00:00.000	0.00	0.00
6	6	1	2	2009-10-06 00:00:00.000	0.00	0.00
7	8	1	3	2010-04-16 20:00:34.110	125.50	0.10
8	9	2	4	2010-04-16 20:02:51.153	250.30	0.10

```
DELETE FROM Customers
WHERE CustomerID = 1
```

- Забраняване на изтриване на ред от реферираната таблица, ако той има съответни редове в реферираща таблица:

```
CREATE TRIGGER CustomerHasSales
   ON Customers
   FOR DELETE
AS
 IF @@ROWCOUNT = 0 RETURN

 IF EXISTS
    ( SELECT * FROM Deleted d
      INNER JOIN Sales s
         ON d.CustomerID = s.CustomerID )
  BEGIN
   RAISERROR('Клиентът има осъществени
    продажби. Изтриването не е изпълнено!', 16, 1)
   ROLLBACK TRANSACTION
  END
```

- Каскадно обновяване в реферираната таблица, т.е. обновява всички редове в реферираща таблица със съответните стойности за външния ключ:

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	PhoneNumber	FaxNumber	EmailAddress
1	1	Алтметал ЕООД	Иван Иванов	мениджър	ул. Васил Левски, 12	Велико Търново	5000	123456	98765544	ivan@gmail.com
2	2	Деница Стар ЕООД	Петър Иванов	мениджър	бул. България, 34	Велико Търново	5000	565656	3453645765	peter@gmail.com
3	3	МИЛСТОР 90 ЕООД	Петър Петров	мениджър	ул. Страцин, 67	Велико Търново	5000	5465465	456566	petrov@gmail.com
4	4	Мики-92 ЕООД	Христо Димитров	мениджър	ул. Г. Козарев, 7	Велико Търново	5000	454545	6766655	hristo@gmail.com

	SaleID	CustomerID	EmployeeID	SaleDate	TotalForSale	Discount
1	1	1	1	2010-01-30 09:08:00.000	8.75	0.00
2	2	3	1	2010-01-30 10:08:00.000	42.00	0.00
3	3	2	3	2010-01-31 12:08:00.000	1.80	0.00
4	4	2	4	2010-01-31 13:08:00.000	10.10	0.00
5	5	1	2	2009-10-06 00:00:00.000	0.00	0.00
6	6	1	2	2009-10-06 00:00:00.000	0.00	0.00
7	8	1	3	2010-04-16 20:00:34.110	125.50	0.10
8	9	2	4	2010-04-16 20:02:51.153	250.30	0.10

```
UPDATE Customers
   SET CustomerID = 50
WHERE CustomerID = 1
```

- Каскадно обновяване в реферираната таблица, т.е. обновява всички редове в реферираща таблица със съответните стойности за външния ключ:

```
CREATE TRIGGER UpdateCascadeCustomers
  ON Customers
  FOR UPDATE
 AS
  DECLARE @num_affected int, @customerID int,
          @old_customerID int

  SET @num_affected = @@ROWCOUNT

  IF @num_affected = 0 RETURN

  IF UPDATE ( CustomerID )
     BEGIN
        IF @num_affected = 1
           BEGIN
```

```
SELECT @customerID = CustomerID
FROM inserted

 SELECT @old_customerID = CustomerID
 FROM deleted

 UPDATE Sales
    SET CustomerID = @CustomerID
 WHERE CustomerID = @old_customerID

 SET @num_affected = @@ROWCOUNT

 RAISERROR('Каскадното актуализиране в
           Customers на първичния ключ от %d
           на %d на %d реда в Sales.', 11, 1,
           @old_customerID, @customerID,
           @num_affected)
END
ELSE
```

```
 BEGIN
     RAISERROR('Не може да се променят множество
                стойности на първичния ключ в
                една конструкция, поради
                съществуването на каскадно
                актуализиращ тригер.', 16, 1)
     ROLLBACK TRANSACTION
   END
END
```

- в разгледания тригер е забранено обновяването на първичния ключ да засяга повече от един ред чрез проверка на стойността на функцията `@@ROWCOUNT`;
- обикновено не е добре да се прави масово обновяване на първичния ключ.

- Тригерът `UpdateProductName` се стартира при актуализиране на колона с ограничение за уникалност. Необходимо е при конструкции, обновяващи множество редове, тригерът да игнорира тези стойности, които нарушават ограничението за уникалност, всички останали да се приемат (за действието `INSERT` този резултат може да се постигне чрез дефиниране на уникален индекс с опцията `IGNORE_DUP_KEY`).

- За проследяване на действието на тригера, нека е създадена временна таблица `#MyTemp` със следните примерни данни:

```
SELECT t.ProductID AS MyID,
       t.ProductName AS MyName,
       p.ProductID, p.ProductName
FROM #MyTemp t
INNER JOIN Products p
   ON t.ProductID = p.ProductID
```

нови стойности

MyID	MyName
1	ябълки
2	круши
3	круши
4	портокали
5	мандарини
6	грейпфрути
7	киви

стари стойности

ProductID	ProductName
1	домати
2	краставици
3	зеле
4	моркови
5	лук
6	чесън
7	ябълки

стойности след актуализацията

MyID	MyName
1	**ябълки**
2	круши
3	**круши**
4	портокали
5	мандарини
6	грейпфрути
7	киви

ProductID	ProductName
1	**домати**
2	круши
3	**зеле**
4	портокали
5	мандарини
6	грейпфрути
7	киви

- Нека имената на продуктите се актуализират в съответствие със стойностите във временната таблица по следния начин:

```
UPDATE Products
   SET ProductName = t.ProductName
FROM Products p
INNER JOIN #MyTemp t
   ON t.ProductID  = p.ProductID
```

```
CREATE TRIGGER UpdateProductName
   ON Products
   INSTEAD OF UPDATE
AS
   IF @@ROWCOUNT = 0 RETURN

   IF UPDATE(ProductName)
      BEGIN
         UPDATE Products
            SET ProductName = i.ProductName
         FROM Products p
         INNER JOIN inserted i
            ON p.ProductID = i.ProductID
         WHERE i.ProductName NOT IN
               (SELECT ProductName FROM Products)
           AND i.ProductName NOT IN
               (SELECT ProductName FROM inserted i1
                WHERE i1.ProductID < i.ProductID)
      END
ELSE -- ако модификацията засяга друга колона
```

```
UPDATE Products
      SET CategoryID = i.CategoryID,
          SupplierID = i.SupplierID,
          Price = i.Price,
          Stock = i.Stock,
          ReorderLevel = i.ReorderLevel,
          Discontinued = i.Discontinued
   FROM Products p
   INNER JOIN inserted i
      ON p.ProductID = i.ProductID
```

СЪЗДАВАНЕ НА ТРИГЕРИ

- Проверките за нарушаване на ограниченията се извършват преди активирането на тригерите.
- За по-добра четимост може да се декларира ограничение FOREIGN KEY в скриптовете с CREATE TABLE, а след това да се забрани ограничението външен ключ със следната конструкция:

```
ALTER TABLE table_name
NOCHECK CONSTRAINT {ALL | constraint_name
[,…]}
```

ЗАДАЧИ

- ***Задача 1***. Да се създаде тригер, забраняващ променянето на продажна цена, която да е по-ниска от съответната доставна цена на продукт.
- ***Задача 2***. Да се създаде тригер, забраняващ въвеждането на нов ред в `SaleDetails`, в който продажната цена да е по-ниска от съответната доставна цена на продукт. Ако не се зададе стойност или стойността е по-малка от доставната, да се въвежда стойността на доставната за новия ред.
- ***Задача 3.*** Да се създаде тригер, който актуализира отстъпката от продажната цена за покупка на голямо количество от даден продукт.
- ***Задача 4***. Да се създаде тригер, който актуализира общата сума на продажба.
- ***Задача 5.*** Да се създаде тригер, който актуализира текущия баланс на клиент.
- ***Задача 6***. Да се създаде тригер, който забранява продажбата на продукт в количество, по-голямо от наличното. Тригерът да актуализира наличното количество, като го намалява с продаденото.

ЗАДАЧИ

- *Задача 7*. Да се създаде тригер, който използва таблицата `ProductInformation`, съхраняваща допълнителна информация за продуктите. Нека в таблицата Products е добавена колона `InformationFlag` от тип *bit*.
  - Стойност единица в тази колона показва, че съществува поне един съответен ред за този продукт в таблицата `ProductInformation`;
  - стойност нула означава, че не съществува нито един съответен ред с допълнителна информация за продукта.
- Тригерът да актуализира стойността на колоната `InformationFlag` при добавяне или изтриване на редове в `ProductInformation`. Тази колона да се поддържа актуална и при променяне на идентификатора на продукта, за който се отнася допълнителната информация.