# Семинар 13 – Программирование на Transact-SQL

## Обзор

На этом семинаре Вы будете использовать DML-триггеры, написанные на языке Transact-SQL для работы с данными в базе данных **AdventureWorksLT**.

## Что необходимо для выполнения

- Доступ к облачному сервису Microsoft Azure SQL Database с БД **AdventureWorksLT**.

или

- Установленный Microsoft SQL Server с SQL Server Management Studio и БД **AdventureWorksLT**.

## Задача 1: Ограничения на цены товаров

Руководство компании приняло решение о том, что товары, продаваемые в рамках одной категории, должны быть сопоставимы по ценам, а именно – отпускная цена товаров (ListPrice) не может отличаться более чем в 20 раз в рамках одной категории.

### 1. Напишите код проверки выполнения правила ограничения цен.

Поскольку перед вводом ограничения в БД необходимо убедиться, что текущие данные удовлетворяют введенному правилу Вам необходимо написать код, который бы возвращал список товаров (все столбцы из таблицы **Product**), которые нарушают указанное правило. Если такие товары обнаружатся – необходимо вывести надпись 'Правило 20-кратной разницы в цене нарушено у <*количество*> товаров', в противном случае вывести 'Правило 20-кратной разницы в цене соблюдено'.

**Подсказка**: посмотрите документацию по триггерам CREATE TRIGGER в справочнике по Transact-SQL.

### 2. Создайте триггер для обеспечения правила 20-кратной разницы в отпускной цене

Вам поручили написать триггер с именем **SalesLT.TriggerProductListPriceRules**, который должен поддерживать в БД правило 20-кратной разницы в отпускной цене товаров из одной рубрики. Основное назначение триггера – отменять изменения данных, которые нарушают правило 20-кратной разницы в цене, оповещая при этом пользователя выбрасыванием ошибки с номером 50001 и сообщением '*Вносимые изменения нарушают правило 20-кратной разницы в цене товаров из одной рубрики (слишком дешево)*' или '*Вносимые изменения нарушают правило 20-кратной разницы в цене товаров из одной рубрики (слишком дорого)*'.

### 3. Протестируйте созданный триггер

После создания триггера необходимо его протестировать. Вы решаете использовать для этого попытку добавления слишком дорогого, а затем слишком дешевого товара в категорию 'Mountain Bikes' (**ProductCategoryID** = 5). Напишите скрипт, который вычисляет минимальную и максимальную отпускные цены товаров из категории 'Mountain Bikes' и выводит их на экран в виде сообщения '*Минимальная цена: <**минимум**>, максимальная цена: <**максимум**>*'. А затем осуществляет попытку добавления (в блоках с наблюдением за ошибками) слишком дорогого и слишком дешевого товаров.

В результате вы должны показать, что выводятся оба сообщения об ошибке, а ошибочные данные в таблицу товаров не попадают.

## Задача 2: Поддержка ссылочной целостности

Вы решаете попрактиковаться в создании триггеров и решаете временно заменить ограничение ссылочной целостности **FK_Product_ProductCategory_ProductCategoryID**, установленное между таблицами **Product** и **ProductCategory** на ту же логику, но на основе триггеров.

### 1. Создание триггеров

Вам необходимо создать два AFTER-триггера **TriggerProduct** и **TriggerProductCategory**: по одному для каждой из таблиц **Product** и **ProductCategory** для поддержания ссылочной целостности между этими таблицами по полю **ProductCategoryID**. Соответственно, триггеры должны выбрасывать ошибку 50002 с описанием 'Ошибка: попытка нарушения ссылочной целостности между таблицами Product и ProductCategory, транзакция отменена' (в **TriggerProduct** состояние 0, в **TriggerProductCategory** – состояние 1).

### 2. Тестирование триггеров

Для тестирования триггеров Вы временно отключаете проверку ограничения на основе внешнего ключа (после обновления диаграммы связь между таблицами становится пунктирной):

```
ALTER TABLE SalesLT.Product NOCHECK CONSTRAINT FK_Product_ProductCategory_ProductCategoryID;
```

Необходимо попытаться добавить запись о любом новом товаре с ProductCategoryID=-1 в таблицу **Product** и убедиться, что Вы получите сообщение об ошибке из триггера. Затем Вам нужно попытаться удалить запись о категории с ProductCategoryID=5 из таблицы **ProductCategory** и убедиться, что Вы также получите сообщение об ошибке из триггера.

### 3. Восстановление внешнего ключа и отключение триггеров

Прочитав в документации, что поддержание ссылочной целостности на основе триггеров работает медленнее, вы решаете восстановить проверку внешнего ключа и отключить созданные триггеры (сделать их неактивными):

```
-- включение проверки внешнего ключа
ALTER TABLE SalesLT.Product CHECK CONSTRAINT FK_Product_ProductCategory_ProductCategoryID;
-- отключение триггеров
DISABLE TRIGGER SalesLT.TriggerProduct ON SalesLT.Product;
DISABLE TRIGGER SalesLT.TriggerProductCategory ON SalesLT.ProductCategory;
```

После это еще раз попытайтесь выполнить добавление товара и удаление категории, как в предыдущем задании и убедитесь, что внешний ключ поддерживает целостность.

P.S.

Если у Вас осталось время после выполнения задания:

- протестируйте триггеры на сценариях с изменением данных в столбцах **ProductCategoryID** таблиц. Убедитесь, что триггеры работают корректно и в этом случае;
- модифицируйте триггер на таблице **ProductCategory**, чтобы реализовать сценарий каскадного удаления (при удалении категории – все товары из нее тоже удаляются). Протестируйте сценарий. Какие сложности при удалении товара могут возникнуть? Как их можно решить?