**Лабораторна робота № 11. Тригери**

## Визначення тригерів

Тригери представляють спеціальний тип збереженої процедури, яка викликається автоматично при виконанні певної дії над таблицею або поданням, зокрема, при додаванні, зміну або видалення даних, тобто при виконанні команд INSERT, UPDATE, DELETE.

Формальне визначення тригера:

|  |
| --- |
| CREATE TRIGGER ім’я\_тригера  ON {Ім'я\_таблиці | ім’я\_представлення}  {AFTER | INSTEAD OF} [INSERT | UPDATE | DELETE]  AS вираз\_sql |

Для створення тригера застосовується вираз CREATE TRIGGER, після якого йде ім'я тригера. Як правило, ім'я тригера відображає тип операцій і ім'я таблиці, над якою проводиться операція.

Кожен тригер асоціюється з певною таблицею або поданням, ім'я яких вказується після слова ON.

Потім встановлюється тип тригера. Ми можемо використовувати один з двох типів:

* **AFTER**: Виконується після виконання дії. Визначається тільки для таблиць.
* **INSTEAD OF**: Виконується замість дії (тобто по суті дію - додавання, зміна або видалення - взагалі не виконується). Визначається для таблиць і уявлень

Після типу тригера йде вказівка операції, для якої визначається тригер: INSERT, UPDATE або DELETE.

Для тригера AFTER можна застосовувати відразу для декількох дій, наприклад, UPDATE і INSERT. У цьому випадку операції вказуються через кому. Для тригера INSTEAD OF можна визначити тільки одну дію.

І потім після слова AS йде набір виразів SQL, які власне і складають тіло тригера.

Створимо тригер. Припустимо, у нас є база даних productsdb з наступним визначенням:

|  |
| --- |
| CREATE DATABASE productdb;  GO  USE productdb;  CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  ); |

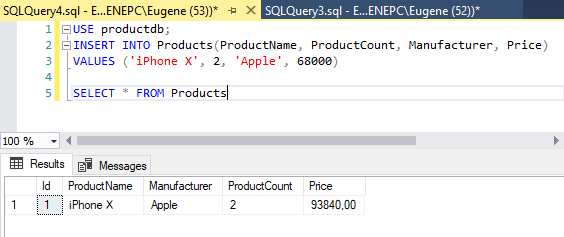
Визначимо тригер, який буде спрацьовувати при додаванні і оновленні даних:

|  |
| --- |
| USE productdb;  GO  CREATE TRIGGER Products\_INSERT\_UPDATE  ON Products  AFTER INSERT, UPDATE  AS  UPDATE Products  SET Price = Price + Price \* 0.38  WHERE Id = (SELECT Id FROM inserted) |

Припустимо, в таблиці Products зберігаються дані про товари. Але ціна товару нерідко містить різні надбавки типу податку на додану вартість, податку на додану корупцію і так далі. Людина, який додає дані, може не знати всі ці тонкощі з податковою базою, і він визначає чисту ціну. За допомогою тригера ми можемо поправити ціну товару на деяку величину.

Таким чином, тригер буде спрацьовувати при будь-якої операції INSERT або UPDATE над таблицею Products. Сам тригер буде змінювати ціну товару, а для отримання того товару, який був доданий або змінений, знаходимо цей товар по Id. Але яке значення має мати Id такий товар? Справа в тому, що при додаванні або зміні дані зберігаються в проміжну таблицю inserted. Вона створюється автоматично. І з неї ми можемо отримати дані про доданих / змінених товарах.

І після додавання товару в таблицю Products в реальності товар буде мати трохи більшу ціну, ніж та, яка була визначена при додаванні:



**Видалення тригера**

Для видалення тригера необхідно застосувати команду DROP TRIGGER:

|  |
| --- |
| DROP TRIGGER Products\_INSERT\_UPDATE |

**Відключення тригера**

Буває, що ми хочемо зупинити дію тригера, але видаляти його повністю не хочемо. В цьому випадку його можна тимчасово відключити за допомогою команди DISABLE TRIGGER:

|  |
| --- |
| DISABLE TRIGGER Products\_INSERT\_UPDATE ON Products |

А коли тригер знадобиться, його можна включити за допомогою команди ENABLE TRIGGER:

|  |
| --- |
| ENABLE TRIGGER Products\_INSERT\_UPDATE ON Products |

## Тригери для операцій INSERT, UPDATE, DELETE

Для розгляду операцій з тригерами визначимо наступну базу даних productsdb:

|  |
| --- |
| CREATE DATABASE productsdb;  GO  USE productsdb;  CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE History  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL,  Operation NVARCHAR (200) NOT NULL,  CreateAt DATETIME NOT NULL DEFAULT GETDATE (),  ); |

Тут визначені дві таблиці: Products - для зберігання товарів і History - для зберігання історії операцій з товарами.

**Додавання**

При додаванні даних (при виконанні команди INSERT) в тригері ми можемо отримати додані дані з віртуальної таблиці INSERTED.

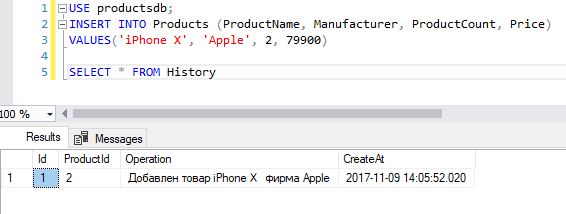
Визначимо тригер, який буде спрацьовувати після додавання:

|  |
| --- |
| USE productsdb  GO  CREATE TRIGGER Products\_INSERT  ON Products  AFTER INSERT  AS  INSERT INTO History (ProductId, Operation)  SELECT Id, 'долучення товар' + ProductName + 'фірма' + Manufacturer  FROM INSERTED |

Цей тригер буде додавати в таблицю History дані про додавання товару, які беруться з віртуальної таблиці INSERTED.

Виконаємо додавання даних в Products і отримаємо дані з таблиці History:

|  |
| --- |
| USE productsdb;  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES ( 'iPhone X', 'Apple', 2, 79900)  SELECT \* FROM History |



**Видалення даних**

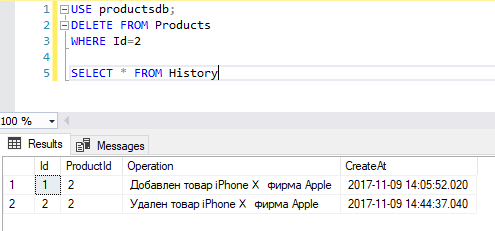
При видаленні всі видалені дані містяться в віртуальну таблицю DELETED:

|  |
| --- |
| USE productsdb  GO  CREATE TRIGGER Products\_DELETE  ON Products  AFTER DELETE  AS  INSERT INTO History (ProductId, Operation)  SELECT Id, 'Вилучений товар' + ProductName + 'фірма' + Manufacturer  FROM DELETED |

Тут, як і в випадку з попереднім тригером, поміщаємо інформацію про віддалених товари в таблицю History.

Виконаємо команду на видалення:

|  |
| --- |
| USE productsdb;  DELETE FROM Products  WHERE Id = 2  SELECT \* FROM History |



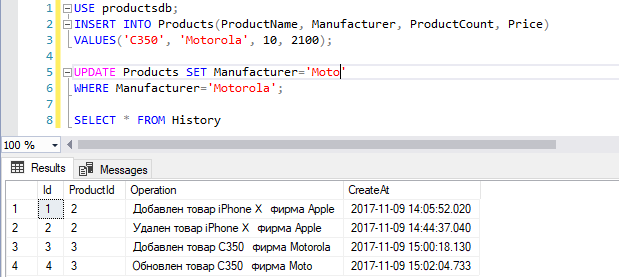
**Зміна даних**

Тригер поновлення даних спрацьовує при виконанні операції UPDATE. І в такому тригері ми можемо використовувати дві віртуальних таблиці. Таблиця INSERTED зберігає значення рядків після поновлення, а таблиця DELETED зберігає ті ж рядки, але до оновлення.

Створимо тригер поновлення:

|  |
| --- |
| USE productsdb  GO  CREATE TRIGGER Products\_UPDATE  ON Products  AFTER UPDATE  AS  INSERT INTO History (ProductId, Operation)  SELECT Id, 'Оновлений товар' + ProductName + 'фірма' + Manufacturer  FROM INSERTED |

І при оновленні даних спрацює даний тригер:



## Тригер INSTEAD OF

Тригер INSTEAD OF спрацьовує замість операції з даними. Він визначається в принципі також, як тригер AFTER, за тим винятком, що він може визначатися тільки для однієї операції - INSERT, DELETE або UPDATE. І також він може застосовуватися як для таблиць, так і для уявлень (тригер AFTER застосовується тільки для таблиць).

Наприклад, створимо такі базу даних і таблицю:

|  |
| --- |
| CREATE DATABASE prods;  GO  USE prods;  CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  Price MONEY NOT NULL,  IsDeleted BIT NULL); |

Тут таблиця містить стовпець IsDeleted, який вказує, чи повністю запис. Тобто замість жорсткого видалення повністю з бази даних ми хочемо виконати м'яке видалення, при якому запис залишається в базі даних.

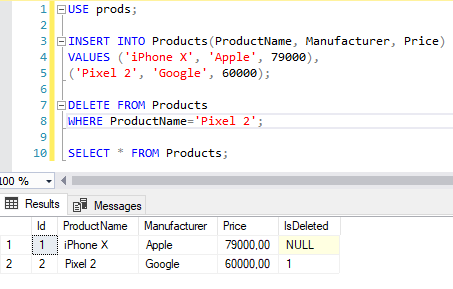
Визначимо тригер для видалення запису:

|  |
| --- |
| USE prods  GO  CREATE TRIGGER products\_delete  ON Products  INSTEAD OF DELETE  AS  UPDATE Products  SET IsDeleted = 1  WHERE ID = (SELECT Id FROM deleted) |

Додамо деякі дані в таблицю і виконаємо видалення з неї:

|  |
| --- |
| USE prods;  INSERT INTO Products (ProductName, Manufacturer, Price)  VALUES ( 'IPhone X', 'Apple', 79000),  ( 'Pixel 2', 'Google', 60000);  DELETE FROM Products  WHERE ProductName = 'Pixel 2';  SELECT \* FROM Products; |

Таким чином, видаляються записи насправді не будуть видалятися, просто у них буде встановлюватися значення для стовпця IsDeleted:



Завдання

1. Проробити всі приклади тригерів
2. Створити тригер на видалення даних з таблиці orders, який буде зберігати дані про номер видаленого замовлення та його дату
3. Створити тригер на видалення даних з таблиці Customers. Який замість видалення запису змінить його статус IsDeleted (НЕОБХІДНО!!! Попередньо створити дане поле у таблиці).
4. Створити тригер на Insert до Customers, який буде вписувати в поле LastName копію введеного поля FirstName
5. Створити тригери для таблиць Customers та Products для збереження в таблиці History (створити, якщо не створили під час аналізу теоретичних відомостей (стор. 3) дані про стан запису перед зміною командами Delete та Update (див. приклад для Orders).

Додаткове завдання

1. \*Створити тригер на Insert до Customers, який на основі функції dbo.want\_name1 (ДИВИСЬ слайд 30 лекція 11) буде додавати лише частину Імені користувача до першого пробілу (треба відключати тригер з теоретичних відомостей, для чистоти спрацювання).
2. \*Реалізувати тригером рознесння Повного імені на складові при введенні даних:
   1. Створити користувацькі функції для отримання з повного імені окремо Ім’я та окремо По батькові. Наприклад, функцію dbo.want\_name2 для імені, dbo.want\_name3 – для По батькові
   2. Модифікувати таблицю Customers і додати поля Name2, Name3 (відповідно для імені і для По батькові)
   3. Модифікувати тригер на Insert до Customers, який на основі користувацьких функцій dbo.want\_name1 буде додавати прізвище в LastName, Ім’я - в Name2, По батькові – в Name3