**КТ № 8 Реализация базы данных. Триггеры.**

**Программное обеспечение: Windows 8 и выше, Microsoft Office Word 2016 и выше, Microsoft SQL Server 2017 и выше, Microsoft SQL Server Management Studio 2017.**

**Цель работы:**

**На основании разработанной структуры базы данных и описания предметной области, разработать триггеры для журнализации событий пользователей.**

**Время на выполнение практической работы:** 4 часа.

**Получение навыков:**

* **В реализации триггеров.**

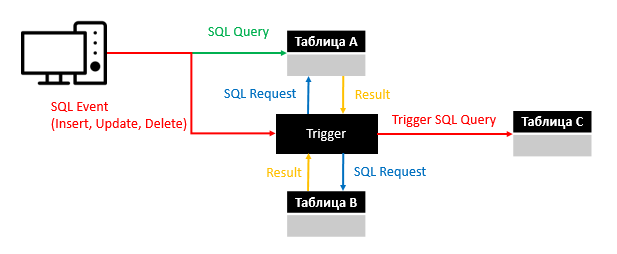
**Теоретические сведения к выполнению практической работы:**

**Триггер** – вид метаданных, является автономной процедурой, который срабатывает в момент манипулирования над данными в таблице или представлении.

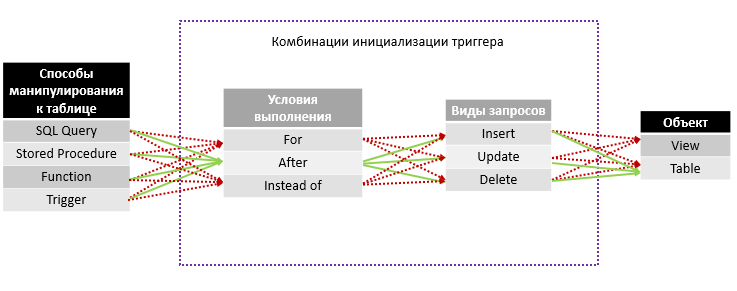
Достоинства триггера:

1. Позволяет автоматизировать большую часть процессов, скрывая её от приложения БД и пользователя;
2. Позволяет обеспечить журнализацию событий;
3. Нет механизмов прямого вызова.

Принцип работы триггера.



Event – инициализация триггера.



Условия выполнения Microsoft SQL Server:

**For, After** – триггер получает и взаимодействует с данными после добавления информации в Table или View. After – работает только с Table;

**Instead of** – триггер выполняет вложенную в него команду, вместо инициализирующего события. Пример: вместо добавления, сделать удаление.

Механизм получения данных из триггерной таблицы.

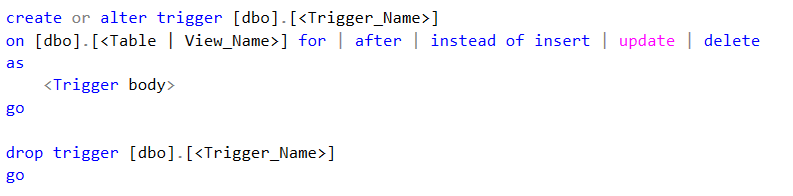
Для обращения к данным, которые учувствовали в работе триггерной таблицы, используются временные, виртуальные таблицы:

**Inserted** – для событий insert, update;

**Deleted** – для события delete.

Данные виртуальные таблицы, не требуют дополнительного объявления и получают только те данные, которые учувствовали в конкретной строке.

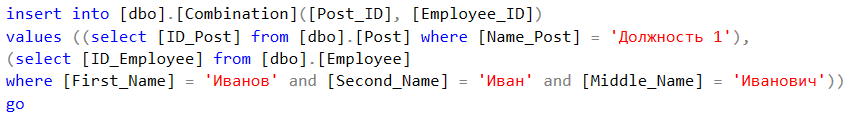
Конструкция триггера.



Вложенный запрос.

Вложенным запросом, называют такой запрос, в котором внутри одно запроса на манипулирование данными, находятся запросы на выборку скалярных значений из прочих объектов базы данных. Является аналогом объявления переменных.

Пример применения вложенного запроса:



Дополнительные команды.

**Uniqueidentifier** – 16-битовый уникальный идентификатор, инициализируется с помощью функции NEWID().

Сравнение uniqueidentifier с identity(n,m).

|  |  |  |
| --- | --- | --- |
|  | **identity(n,m)** | **Uniqueidentifier** |
| Достоинства | * Простота в применении, при разработке БД и заполнении данных в дочерних таблицах; * Суррогатный ключ не учувствует в запросе на добавление данных. | * Высокая степень защиты значения, суррогатного ключа; * Уникальные значения для всех суррогатных ключей, всех таблиц БД. |
| Недостатки | * Низкая степень защиты суррогатного ключа; * Имеются повторения значений суррогатных ключей в таблицах БД. | * Сложность в применении, при разработке БД и заполнении данных в дочерних таблицах; * Суррогатный ключ учувствует в запросе на выборку данных. |

**Этапы выполнения:**

*Для выполнения данной работы, создайте новый файл в Microsoft SQL Server Management Studio, для работы со сценариями триггера.*

1. Из описания предметной области, определить фрагмент(ы) описания вывода истории для того или иного пользователя;

Продуктовый магазин осуществляет деятельность, по отпуску различных товаров, покупателю за эквивалентную сумму денег. Покупатель может ознакомиться с ассортиментом отпускаемой продукции: осуществлять поиск данных по названию продукции, производителю, может ознакомиться с составом продукции, сроком годности, страной и фирмой производителя, осуществлять поиск по ценовой категории, может ознакомиться с количеством имеющейся продукции в рамках нескольких торговых точек. В момент окончания количества того или иного продукта, формируется товарная смета на будущую поставку товара в ту или иную торговую точку. Для каждого кассира и клиента, доступна персональная история продаж и покупок.

1. Реализовать, не связанную с общей структурой, таблицу, для хранения истории. Таблица может быть выполнена не по правилам нормализации;

|  |  |
| --- | --- |
| Скрипт DDL | Результат |
| create table [dbo].[Sale\_History]  (  [ID\_Sale\_History] [uniqueidentifier] not null,  [Sale\_Check\_Number] [varchar] (13) not null,  [Sale\_Check\_Date\_Time\_Create] [varchar] (max) not null,  [Buer\_Info] [varchar] (150) not null,  [Employee\_Info] [varchar] (200) not null,  [Sale\_Check\_Info] [varchar] (max) not null,  [Total\_Coast\_Info] [varchar] (max),  [Create\_Record] [datetime] null default(getdate()),  [Change\_Record] [datetime] null default(getdate())  constraint [PK\_Sale\_History] primary key clustered  ([ID\_Sale\_History] ASC) on [PRIMARY]  )  go |  |

1. Реализовать триггеры для 3-ёх ситуаций: после добавления, изменения и удаления данных;

|  |  |
| --- | --- |
| Скрипт DDL | Результат |
| Триггер на добавление и изменение данных |  |
| create or alter trigger [dbo].[Sale\_History\_Insert\_Update]  on [dbo].[Sale\_Check\_View] after insert, update  as  begin  declare @Sale\_Check\_Number [varchar] (13) = (select distinct([Sale\_Check\_Number]) from [dbo].[Sale\_Check]  inner join [dbo].[Sale\_Check\_View] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [Sale\_Check\_ID] = (select [Sale\_Check\_ID] from [inserted]) and [Product\_ID] = (select [Product\_ID] from [inserted]))  if ((select count(\*) from [dbo].[Sale\_History]  where [Sale\_Check\_Number] = @Sale\_Check\_Number) > 0)  begin  update [dbo].[Sale\_History] set  [Sale\_Check\_Info] = (select STRING\_AGG(CONCAT('Название: ',[Product\_Name],', цена за ед:',CONVERT([varchar](max),[Price\_Product]),' руб., кол-во:',CONVERT([varchar](max),[Sale\_Check\_View\_Ammount]),', сумма: ',CONVERT([varchar](max),[Price\_Product]\*[Sale\_Check\_View\_Ammount]),' руб.'),'; ') from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  [Total\_Coast\_Info] = (select sum([Sale\_Check\_View\_Ammount]\*[Price\_Product]\*0.80) from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  [Change\_Record] = GETDATE()  where  [Sale\_Check\_Number] = @Sale\_Check\_Number  print ('Update Record Complete!')  end  else  begin  insert into [dbo].[Sale\_History] ([ID\_Sale\_History], [Sale\_Check\_Number], [Sale\_Check\_Date\_Time\_Create],  [Buer\_Info], [Employee\_Info], [Sale\_Check\_Info], [Total\_Coast\_Info])  values (NEWID(),  (select [Sale\_Check\_Number] from [dbo].[Sale\_Check] where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  (select concat(CONVERT([varchar] (10),[Date\_Create\_Check]),' ',CONVERT([varchar] (10),[Time\_Create\_Check])) from  [dbo].[Sale\_Check] where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  (select CONCAT('Bank Card # ',[Card\_Number],', Фамилия инициалы: ',[First\_Name\_Buyer],' ',SUBSTRING([Second\_Name\_Buyer],1,1),'.',SUBSTRING([Middle\_Name\_Buyer],1,1),'.') from [dbo].[Buyer]  inner join [dbo].[Buyer\_Card] on [Buyer\_ID] = [ID\_Buyer]  inner join [dbo].[Sale\_Check] on [Buyer\_Card\_ID] = [ID\_Buyer\_Card]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  (select CONCAT('Должность: ',[Name\_Post],', Фамилия инициалы: ',[First\_Name\_Emp],' ',SUBSTRING([Second\_Name\_Emp],1,1),'.',SUBSTRING([Middle\_Name\_Emp],1,1),'.') from [dbo].[Employee]  inner join [dbo].[Combination] on [Employee\_ID] = [ID\_Employee]  inner join [dbo].[Post] on [Post\_ID] = [ID\_Post]  inner join [dbo].[Employee\_Change] on [Combination\_ID] = [ID\_Combination]  inner join [dbo].[Sale\_Check] on [ID\_Employee\_Change] = [Employee\_Change\_ID]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  (select STRING\_AGG(CONCAT('Название: ',[Product\_Name],', цена за ед:',CONVERT([varchar](max),[Price\_Product]),' руб., кол-во:',CONVERT([varchar](max),[Sale\_Check\_View\_Ammount]),', сумма: ',CONVERT([varchar](max),[Price\_Product]\*[Sale\_Check\_View\_Ammount]),' руб.'),'; ') from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])),  (select sum([Sale\_Check\_View\_Ammount]\*[Price\_Product]\*0.80) from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [inserted])))  print('Insert Record Complete!')  end  end  go |
| Триггер на удаление данных |
| create or alter trigger [dbo].[Sale\_History\_Delete]  on [dbo].[Sale\_Check\_View] after delete  as  begin  declare @Sale\_Check\_Number [varchar] (13) = (select distinct([Sale\_Check\_Number]) from [dbo].[Sale\_Check]  inner join [dbo].[Sale\_Check\_View] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [Sale\_Check\_ID] = (select [Sale\_Check\_ID] from [deleted]) and [Product\_ID] = (select [Product\_ID] from [deleted]))  if ((select count(\*) from [dbo].[Sale\_History]  where [Sale\_Check\_Number] = @Sale\_Check\_Number) > 1)  begin  update [dbo].[Sale\_History] set  [Sale\_Check\_Info] = (select STRING\_AGG(CONCAT('Название: ',[Product\_Name],', цена за ед:',CONVERT([varchar](max),[Price\_Product]),' руб., кол-во:',CONVERT([varchar](max),[Sale\_Check\_View\_Ammount]),', сумма: ',CONVERT([varchar](max),[Price\_Product]\*[Sale\_Check\_View\_Ammount]),' руб.'),'; ') from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [deleted])),  [Total\_Coast\_Info] = (select sum([Sale\_Check\_View\_Ammount]\*[Price\_Product]\*0.80) from [dbo].[Product]  inner join [dbo].[Sale\_Check\_View] on [Product\_ID] = [ID\_Product]  inner join [dbo].[Sale\_Check] on [Sale\_Check\_ID] = [ID\_Sale\_Check]  where [ID\_Sale\_Check] = (select [Sale\_Check\_ID] from [deleted])),  [Change\_Record] = GETDATE()  where  [Sale\_Check\_Number] = @Sale\_Check\_Number  print ('Remove Update Record Complete!')  end  else  begin  delete from [dbo].[Sale\_History]  where [Sale\_Check\_Number] = @Sale\_Check\_Number  print ('Delete Record Complete!')  end  end  go |

1. Заполнить триггерную таблицу (при необходимости, ряд дочерних таблиц), тестовыми данными не менее 5 записей и вывести результат;

|  |
| --- |
| Скрипт DML |
| exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 1, @Buyer\_Card\_ID = 1  go  exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 1, @Buyer\_Card\_ID = 2  go  exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 2, @Buyer\_Card\_ID = 1  go  exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 3, @Buyer\_Card\_ID = 3  go  exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 4, @Buyer\_Card\_ID = 2  go  exec [dbo].[Sale\_Check\_insert] @Sale\_Check\_Number = ' ', @Deposited\_Ammount = 1, @Whith\_VAT = 1,  @Without\_VAT = 1, @Payment\_Type\_ID = 2, @Employee\_Change\_ID = 4, @Buyer\_Card\_ID = 3  go |
| Результат |
|  |
| Скрипт DML |
| exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 5, @Product\_ID = 1  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 5, @Product\_ID = 2  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 5, @Product\_ID = 1  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 6, @Product\_ID = 3  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 6, @Product\_ID = 2  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 7, @Product\_ID = 1  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 7, @Product\_ID = 1  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 7, @Product\_ID = 2  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 7, @Product\_ID = 4  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 8, @Product\_ID = 5  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 9, @Product\_ID = 4  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 10, @Product\_ID = 3  go  exec [dbo].[Sale\_Check\_View\_Work] @Sale\_Check\_ID = 10, @Product\_ID = 5  go |
| Результат |
|  |

1. Вывести результат из таблицы с историей;

|  |
| --- |
| Скрипт DML |
| select \* from [dbo].[Sale\_History]  go |
| Результат |
|  |

1. Реализовать любым из методов: запросом, представлением или функцией вывод данных для каждого из пользователя.

|  |  |
| --- | --- |
| Роль | Создание объекта и вызов |
| Кассир | create or alter function [dbo].[Employee\_Sale\_History](@Employee\_Info [varchar] (100))  returns table  as  return(select [Sale\_Check\_Number]+' '+[Sale\_Check\_Date\_Time\_Create] as "Основная информация по чеку",  [Sale\_Check\_Info] +', итоговая стоимость: '+CONVERT([varchar](max), [Total\_Coast\_Info])+' руб.' as "Состав и итог" from [dbo].[Sale\_History]  where [Employee\_Info] like '%'+@Employee\_Info+'%')  go  select \* from [dbo].[Employee\_Sale\_History]('Дмитриев Д.Д')  go |
| Результат |
|  |
| Покупатель | Создание объекта и вызов |
| create or alter function [dbo].[Buyer\_Sale\_History](@Buyer\_Info [varchar] (100))  returns table  as  return(select [Sale\_Check\_Number]+' '+[Sale\_Check\_Date\_Time\_Create] as "Основная информация по чеку",  [Sale\_Check\_Info] +', итоговая стоимость: '+CONVERT([varchar](max), [Total\_Coast\_Info])+' руб.' as "Состав и итог",  [Employee\_Info] as "Отвесвенный сотрудник" from [dbo].[Sale\_History]  where [Buer\_Info] like '%'+@Buyer\_Info+'%')  go  select \* from [dbo].[Buyer\_Sale\_History]('Иванов')  go |
| Результат |
|  |