

Частное учреждение образования  
«Колледж бизнеса и права»

УТВЕРЖДАЮ  
Заместитель директора  
по учебной работе  
\_\_\_\_\_ Малафеев И.В.  
«\_\_\_» \_\_\_\_\_ 2016

Специальность 2-40 01 01: «Программное обеспечение ин- формационных технологий»	Дисциплина: «Базы данных и системы управления базами данных»
Составлена на основании учебной программы, утвержденной директором Колледжа бизнеса и права 25.11.2011	

### Лабораторная работа № 14

#### Инструкционно-технологическая карта

Тема: Создание и модификация триггеров и процедур в СУБД

Цель работы:

- изучить понятие хранимой процедуры, их типы;
- получить навыки работы с основными инструкциями создания, удаления и изменения хранимых процедур: CREATE PROCEDURE, ALTER PROCEDURE, DROP PROCEDURE.

Время выполнения: 2 часа

#### Краткие теоретические сведения

**Хранимые процедуры** представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемых в базе данных в откомпилированном виде.

Основное различие между пакетом и хранимой процедурой состоит в том, что последняя сохраняется в виде объекта базы данных, т.е. сохраняется на стороне сервера.

#### Типы хранимых процедур

SQL Server поддерживает *пользовательские хранимые процедуры* и *системные процедуры*. Хранимые процедуры создаются таким же образом, как и все другие объекты базы данных, т.е. при помощи языка DDL. Системные процедуры представляются SQL Server.

*Системные хранимые процедуры* предназначены для выполнения различных административных действий. Практически все действия по администрированию сервера выполняются с их помощью. Можно сказать, что системные хранимые процедуры являются интерфейсом, обеспечивающим работу с системными табли-

цами. Системные хранимые процедуры имеют префикс **sp\_**, в системной базе данных и могут быть вызваны в контексте любой другой базы данных.

*Пользовательские хранимые процедуры* реализуют те или иные действия. Хранимые процедуры – полноценный объект базы данных. Хранимая процедура предварительно компилируется перед тем, как она сохраняется в виде объекта базы данных. Предварительно скомпилированная форма процедуры сохраняется в базе данных и используется при каждом ее вызове. Хранимые процедуры создаются таким же образом, как и все другие объекты базы данных, т.е. при помощи языка DDL.

При создании хранимой процедуры можно определить необязательный список параметров. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Хранимые процедуры могут возвращать значение, содержащее определенную пользователем информацию или, в случае ошибки, соответствующее сообщение об ошибке.

*Временные хранимые процедуры* существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Они делятся на *локальные* и *глобальные*. Локальные временные хранимые процедуры могут быть вызваны только из одного соединения, в котором созданы. При создании такой процедуры ей необходимо дать имя, начинающееся из одного символа #. Как и все временные объекты, хранимые процедуры этого типа удаляются при отключении пользователя, перезапуске или остановке сервера. Глобальные временные хранимые процедуры доступны для любых соединений сервера, на котором имеется такая же хранимая процедура. Для ее определения достаточно ей дать имя, начинающееся с символа ##. Удаляются эти процедуры при перезапуске или остановке сервера, а также при закрытии соединения, в контексте которого они были созданы.

Хранимые процедуры можно также использовать для следующих целей:

- ✓ управления авторизацией доступа;
- ✓ создания аудиторского следа действий с таблицами баз данных.

### Создание и изменение хранимых процедур

Синтаксис оператора создания новой или изменения имеющейся хранимой процедуры в обозначениях MS SQL Server:

```
CREATE PROC[EDURE] [schema_name.]proc_name
[({ @param1 } type1[VARYING] [= default1] [OUTPUT])] {, ...}
[WITH {RECOMPILE | ENCRYPTION | EXECUTE AS 'user_name'}]
[FOR REPLICATION]
AS batch| EXTERNAL NAME method_name
```

Параметр	Значение параметра
schema_name.	имя схемы, которая назначается владельцем созданной хранимой процедуры
proc_name	имя хранимой процедуры
@param1	параметр процедуры (формальный аргумент) – значения, которые передаются вызывающим объектом процедуре для использования в ней. Параметра процедуры являются локальными в пределах процедуры
type1	тип данных параметра процедуры @param1. Для определения типа данных параметров хранимой процедуры подходят любые типы данных SQL, включая определенные пользователем

default1	определяет факультативное значение по умолчанию для соответствующего параметра процедуры
OUTPUT	указывает, что параметр процедуры является возвращаемым, и с его помощью можно вернуть значение из хранимой процедуры вызывающей процедуру или системе
VARYING	применяется совместно с параметром OUTPUT, имеющим тип CURSOR, и определяет, что выходным параметром будет результирующее множество
RECOMPILE	предписывает системе создавать план выполнения хранимой процедуры при каждом ее вызове в случае необходимости. См. замечание 1.
FOR REPLICATION	необходим при репликации данных
ENCRYPTION	выполнение шифрования кода хранимой процедуры для обеспечения защиты от использования авторских алгоритмов

**Замечание 1.** Использование опции WITH RECOMPILE сводит на нет одно из наиболее важных преимуществ хранимых процедур: улучшение производительности благодаря одной компиляции.

Ключевое слово AS размещается в начале собственно тела хранимой процедуры. В тело процедуры могут применяться практически все команды SQL, объявляться транзакции, устанавливаться блокировки и вызываться другие хранимые процедуры. Выход из хранимой процедуры можно осуществить посредством команды RETURN.

**Замечание 2.** Для разделения двух пакетов используется инструкция GO. Инструкцию CREATE PROCEDURE нельзя объединять с другими инструкциями Transact-SQL в одном пакете.

### Удаление и изменение хранимой процедуры

SQL Server поддерживает инструкцию ALTER PROCEDURE для модификации структуры хранимых процедур. Инструкция ALTER PROCEDURE обычно применяется для изменения инструкций Transact-SQL внутри процедуры. Все параметры инструкции ALTER PROCEDURE имеют такое же значение, как и одноименные параметры инструкции CREATE PROCEDURE. Основной целью использования этой инструкции является избежание переопределения существующих прав хранимой процедуры.

Для удаления одной или группы хранимых процедур используется инструкция DROP PROCEDURE. Удалить хранимую процедуру может только ее владелец или члены предопределенных ролей db\_owner и sysadmin.

### Выполнение хранимой процедуры

Жизненный цикл хранимой процедуры состоит из двух этапов: ее создания и ее выполнения. Каждая процедура создается один раз, а выполняется многократно. Хранимая процедура выполняется посредством инструкции EXECUTE пользователем. Инструкция имеет следующий:

```
[[EXEC[UTE]] [@return_status=] {proc_name|@proc_name_var}
{[[@parameter1 =] value|[@parameter1=] @variable[OUTPUT]]|DEFAULT}..
[WITH RECOMPILE]
```

Параметр	Значение параметра
return_status	определяет факультативную целочисленную переменную, в кото-

	рой сохраняется состояние возврата процедуры
DEFAULT	Предоставляет значения по умолчанию для параметра процедуры, которое было указано в определении процедуры
@param1	параметр процедуры (формальный аргумент) – значения, которые передаются вызывающим объектом процедуре для использования в ней. Параметра процедуры являются локальными в пределах процедуры

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

**Замечание 3.** Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении, нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

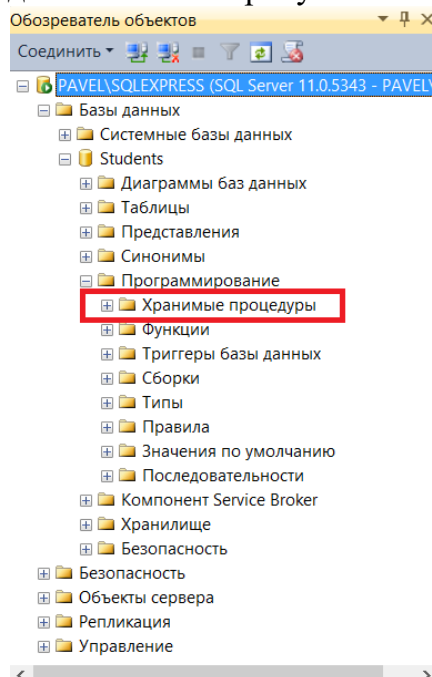
Важно, что при вызове процедуры указываются либо имена параметров со значениями, либо только значения без имени параметра. Их комбинирование не допускается.

### Использование предложения WITH RESULTS SETS инструкции EXECUTE

В SQL Server 2012 для инструкции EXECUTE вводится предложение **WITH RESULT SETS**, посредством которого при выполнении определенных условий можно изменять форму результирующего набора хранимой процедуры.

### Работа с хранимыми процедурами в среде SQL Server Management Studio

В среде SQL Server Management Studio все хранимые процедуры находятся в папке «Хранимые процедуры», расположенной в папке «Программирование» в обозревателе объектов, представленной на рисунке ниже.



Для создания новой хранимой процедуры в обозревателе объектов необходимо щёлкнуть правой кнопкой мыши по папке «Хранимые процедуры» и в появившемся меню выбрать пункт «Создать хранимую процедуру...». Появится окно новой хранимой процедуры, представленное на рисунке ниже.

```
-- =====
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- =====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
-- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

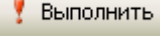
Хранимая процедура состоит из следующих разделов:

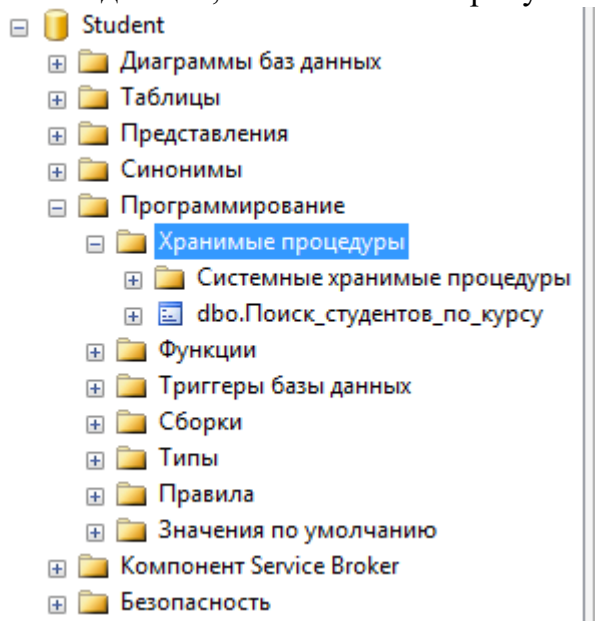
1. область определения имени процедуры;
2. параметры, передаваемые в процедуру (@param1, @param2);
3. тело самой хранимой процедуры состоит из команды SELECT языка T-SQL.

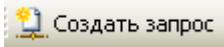
Для того, чтобы создать хранимую процедуру с входными параметрами на поиск студентов заданного курса обучения, в окне новой хранимой процедуры необходимо набрать следующий код, представленный на рисунке ниже

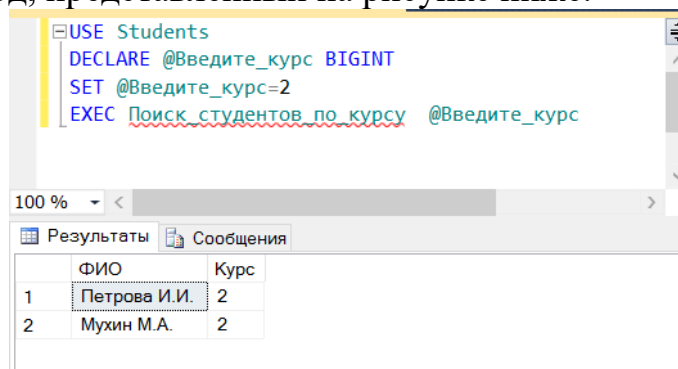
```
CREATE PROCEDURE Поиск_студентов_по_курсу
    @Введите_курс bigint
AS
BEGIN
    SET NOCOUNT ON;
    SELECT ФИО, Курс
    from Студенты
    where Курс=@Введите_курс
END
```

Сообщения  
Выполнение команд успешно завершено.

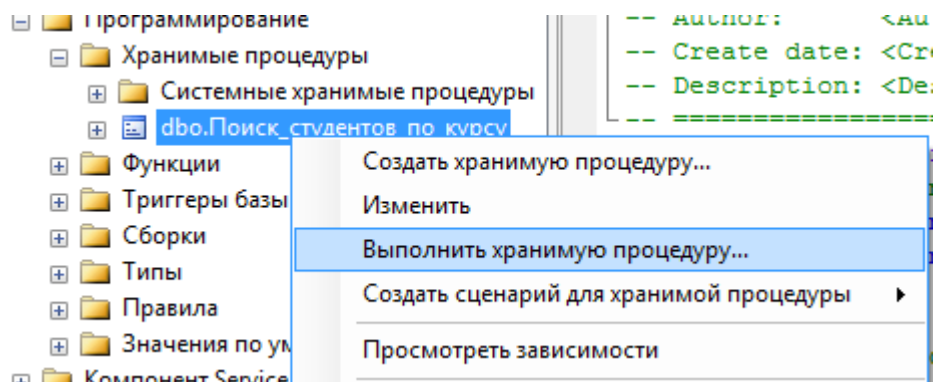
После этого необходимо нажать на кнопку , при этом созданная хранимая процедура появится в базе данных, как показано на рисунке ниже:



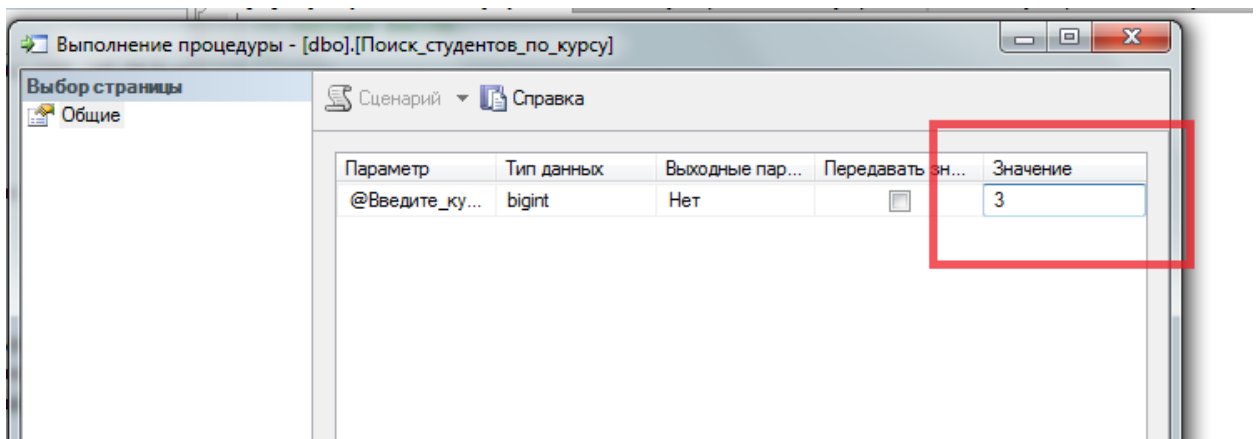
Для вызова хранимой процедуры Поиск\_студентов\_по\_курсу необходимо создать новый запрос с помощью кнопки  и в открывшемся окне нового запроса ввести код, представленный на рисунке ниже.



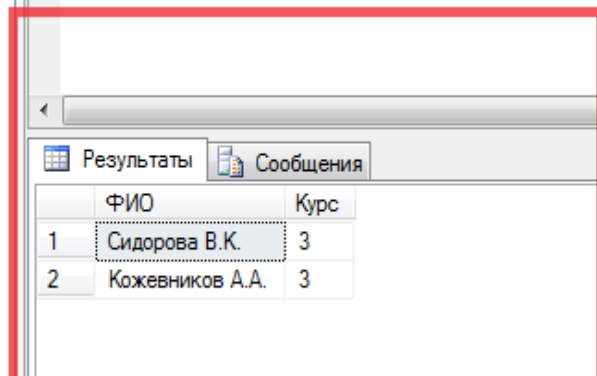
Выполнить хранимую процедуру можно и визуальными средствами среды SQL Server Management Studio. Для выполнения хранимой процедуры необходимо нажать правой кнопкой мыши по названию хранимой процедуры и выбрать «**Выполнить хранимую процедуру...**», как показано на рисунке ниже.



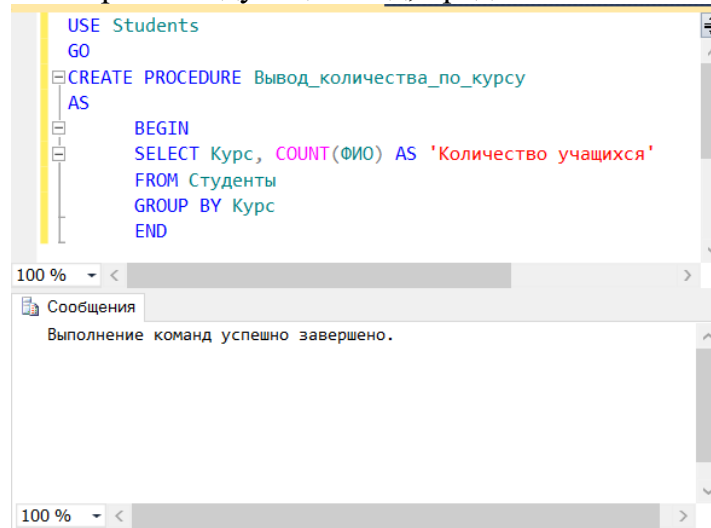
В появившемся окне необходимо ввести значение параметра для поиска и нажать кнопку «ОК», как показано на рисунке ниже.



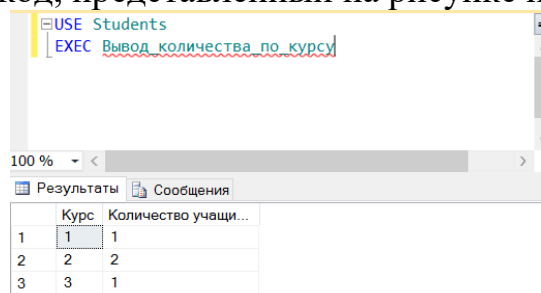
Результат выполнения хранимой процедуры показан ниже.



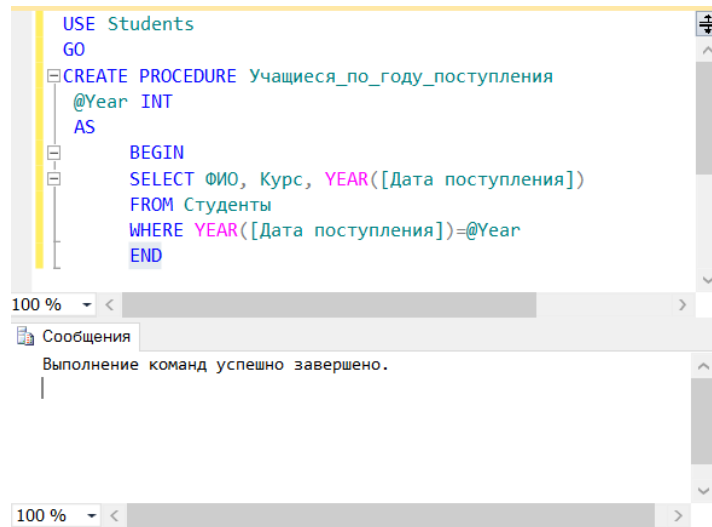
Для того, чтобы создать хранимую процедуру без входных параметров для вывода количества учащихся каждого курса обучения, в окне новой хранимой процедуры необходимо набрать следующий код, представленный на рисунке ниже.



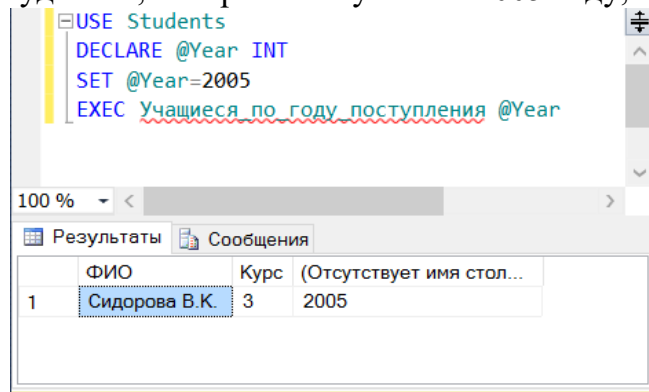
Для вызова хранимой процедуры Вывод количества\_по\_курсу необходимо создать новый запрос с помощью кнопки и в открывшемся окне нового запроса ввести код, представленный на рисунке ниже.



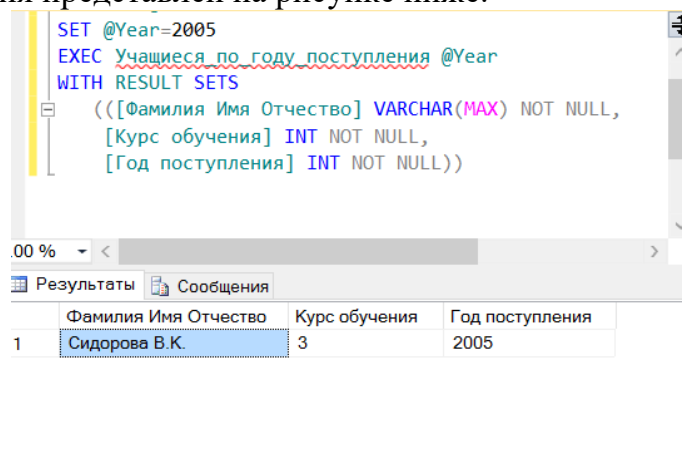
Для того, чтобы создать хранимую процедуру с входными параметрами для вывода информации об учащихся, поступивших в заданном году, в окне новой хранимой процедуры необходимо набрать следующий код, представленный на рисунке ниже.



Результатом поиска студентов, которые поступили в 2005 году, будет



Выполнение этой процедуры выводит таблицу с двумя столбцами, заглавия которых совпадают с наименованиями соответствующих столбцов таблицы базы данных, и один столбец без имени: ФИО, Курс, Отсутствует имя столбца. Чтобы изменить заглавия столбцов (а также их тип данных), в SQL Server 2012 применяется новое предложение WITH RESULT SETS. Предыдущий пример с применением этого предложения представлен на рисунке ниже.





## Триггеры

*Триггер* – это механизм, который вызывается, когда в указанной таблице происходит определенное действие. Каждый триггер имеет следующие основные составляющие:

- ✓ имя, которое может содержать максимум 128 символов;
- ✓ действие, которым может быть или инструкция DML (INSERT, DELETE или UPDATE), или инструкция DDL. Таким образом, существует два типа триггеров: триггеры DML и триггеры DDL;
- ✓ исполнение. Исполнительная часть триггера обычно состоит из хранимой процедуры или пакета.

### Создание триггера DML

Триггер создается с помощью инструкции CREATE TRIGGER, которая имеет следующий синтаксис:

```
CREATE TRIGGER [schema_name.] trigger_name
ON {table_name| view_name}
[WITH dml_trigger_option[,...]]
{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
[WITH APPEND] {AS sql_statement| EXTERNAL NAME method_name}
```

Параметр	Значение параметра
schema_name.	имя схемы, к которой принадлежит триггер
trigger_name	имя триггера
table_name	имя таблицы, для которой создается триггер
select_statement	необходимо задать инструкцию SELECT, которая извлекает строки и столбцы из таблиц (или других представлений). См. замечание 2
FOR   AFTER   INSTEAD OF	указывают тип триггера: триггер AFTER (FOR является синонимом параметра AFTER) или триггер INSTEAD OF. См. замечание 1.
INSERT, UPDATE, DELETE	действие триггера. Под действием триггера имеется в виду инструкция T-SQL, которая запускает триггер. Допускается любая комбинация этих трех инструкций.
sql_statement	действие или действия

**Замечание 1.** Триггеры типа AFTER вызываются после выполнения действия, запускающего триггер, а триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггер. Триггеры AFTER можно создавать только для таблиц, а триггеры INSTEAD — как для таблиц, так и для представлений.

**Замечание 2.** SQL Server позволяет создавать несколько триггеров для каждой таблицы и для каждого действия (INSERT, UPDATE и DELETE). По умолчанию определенного порядка исполнения нескольких триггеров для данного модифицирующего действия не имеется.

### Изменение структуры триггера

Язык Transact-SQL также поддерживает инструкцию **ALTER TRIGGER**, которая модифицирует структуру триггера. Эта инструкция обычно применяется для изменения тела триггера. Все предложения и параметры инструкции ALTER TRIGGER имеют такое же значение, как и одноименные предложения и параметры инструкции CREATE TRIGGER.

Для удаления триггеров в текущей базе данных применяется инструкция **DROP TRIGGER**.

### Использование виртуальных таблиц **deleted** и **inserted**

При создании действия триггера обычно требуется указать, ссылается ли он на значение столбца до или после его изменения действием, запускающим триггер. По этой причине, для тестирования следствия инструкции, запускающей триггер, используются две специально именованные виртуальные таблицы:

- ✓ **deleted** — содержит копии строк, удаленных из таблицы;
- ✓ **inserted** — содержит копии строк, вставленных в таблицу.

Структура этих таблиц эквивалентна структуре таблицы, для которой определен триггер.

Таблица **deleted** используется в том случае, если в инструкции **CREATE TRIGGER** указывается предложение **DELETE** или **UPDATE**, а если в этой инструкции указывается предложение **INSERT** или **UPDATE**, то используется таблица **inserted**. Это означает, что для каждой инструкции **DELETE**, выполненной в действии триггера, создается таблица **deleted**. Подобным образом для каждой инструкции **INSERT**, выполненной в действии триггера, создается таблица **inserted**.

Инструкция **UPDATE** рассматривается, как инструкция **DELETE**, за которой следует инструкция **INSERT**. Поэтому для каждой инструкции **UPDATE**, выполненной в действии триггера, создается как таблица **deleted**, так и таблица **inserted** (в указанной последовательности).

### Триггеры **AFTER**

Триггеры **AFTER** вызываются после того, как выполняется действие, запускающее триггер. Триггер **AFTER** задается с помощью ключевого слова **AFTER** или **FOR**. Триггеры **AFTER** можно создавать только для базовых таблиц.

### Триггеры **INSTEAD OF**

Триггер с предложением **INSTEAD OF** заменяет соответствующее действие, которое запустило его. Этот триггер выполняется после создания соответствующих таблиц **inserted** и **deleted**, но перед выполнением проверки ограничений целостности или каких-либо других действий.

Триггеры **INSTEAD OF** можно создавать как для таблиц, так и для представлений. Когда инструкция **Transact-SQL** ссылается на представление, для которого определен триггер **INSTEAD OF**, система баз данных выполняет этот триггер вместо выполнения любых действий с любой таблицей. Данный тип триггера всегда использует информацию в таблицах **inserted** и **deleted**, созданных для представления, чтобы создать любые инструкции, требуемые для создания запрошенного события.

Значения столбцов, предоставляемые триггером **INSTEAD OF**, должны удовлетворять определенным требованиям:

- ✓ значения не могут задаваться для вычисляемых столбцов;
- ✓ значения не могут задаваться для столбцов с типом данных **TIMESTAMP**;
- ✓ значения не могут задаваться для столбцов со свойством **IDENTITY**, если только параметру **IDENTITY\_INSERT** не присвоено значение **ON**.

Эти требования действительны только для инструкций **INSERT** и **UPDATE**, которые ссылаются на базовые таблицы.

### Триггеры **DDL** и области их применения

SQL Server позволяет определять триггеры для инструкций **DDL**, таких как **CREATE DATABASE**, **DROP TABLE** и **ALTER TABLE**.

Триггеры для инструкций **DDL** имеют следующий синтаксис:

**CREATE TRIGGER** [schema\_name.]trigger\_name

```

ON {ALL SERVER | DATABASE}
[WITH {ENCRYPTION | EXECUTE AS clause_name}
{FOR | AFTER} {event_group | event_type | LOGON}
AS {batch | EXTERNAL NAME method_name}

```

Триггеры DDL создаются таким же способом, как и триггеры DML. А для изменения и удаления этих триггеров используются те же инструкции **ALTER TRIGGER** и **DROP TRIGGER**, что и для триггеров DML.

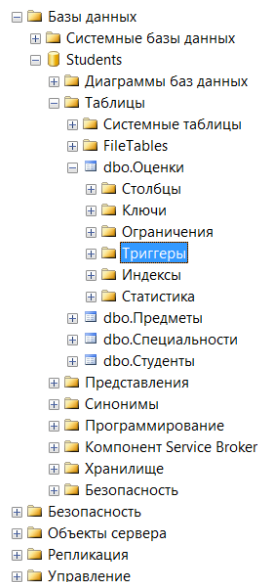
Первым делом при определении триггера DDL нужно указать его область действия. Предложение **DATABASE** указывает в качестве области действия триггера DDL текущую базу данных, а предложение **ALL SERVER** — текущий сервер.

После указания области действия триггера DDL нужно в ответ на выполнение одной или нескольких инструкций DDL указать способ запуска триггера. В параметре **event\_type** указывается инструкция DDL, выполнение которой запускает триггер, а в альтернативном параметре **event\_group** указывается группа событий языка Transact-SQL. Триггер DDL запускается после выполнения любого события языка Transact-SQL, указанного в параметре **event\_group**.

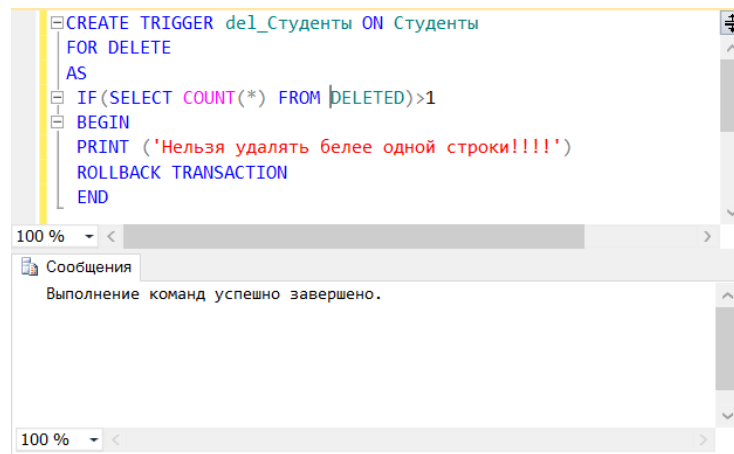
Кроме сходства триггеров DML и DDL, между ними также есть несколько различий. Основным различием между этими двумя видами триггеров является то, что для триггера DDL можно задать в качестве его области действия всю базу данных или даже весь сервер, а не всего лишь отдельный объект. Кроме этого, триггеры DDL не поддерживают триггеров **INSTEAD OF**. Кроме того, для триггеров DDL не требуются таблицы **inserted** и **deleted**, поскольку эти триггеры не изменяют содержимого таблиц.

### Работа с триггерами в среде SQL Server Management Studio

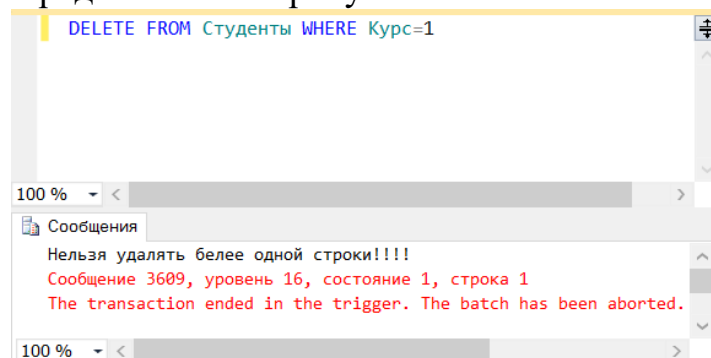
Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке «Триггеры» соответствующей таблицы, которая представлена на рисунке ниже.



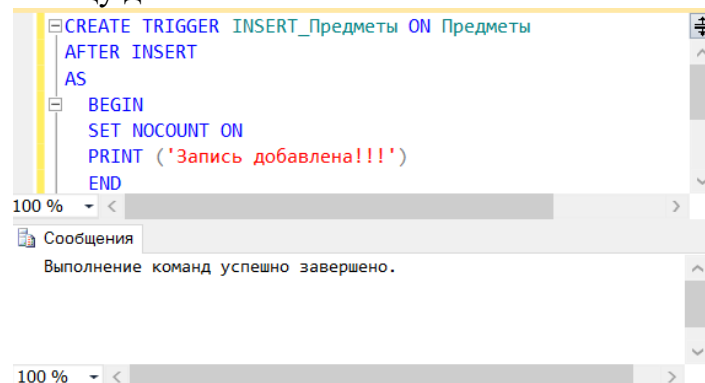
В следующем примере показано создание триггера **del\_Студенты**, который запрещает удалять более одной строки из таблицы **Студенты**.



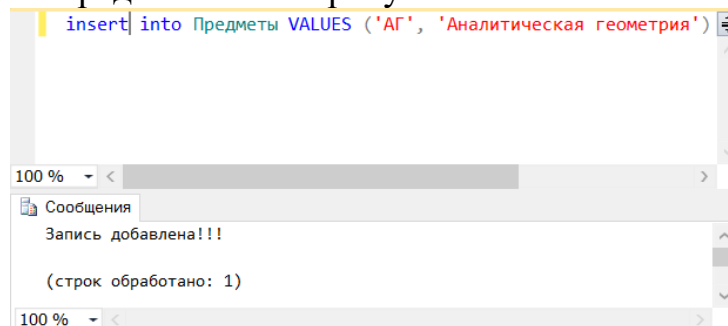
Для того, чтобы проверить, как работает новый триггер необходимо создать новый пустой запрос и в нём выполним удаление данных из таблицы «Студенты», как представлено на рисунке ниже.



В следующем примере показано создание триггера INSERT\_Предметы, который срабатывает после добавления записей в таблицу Предметы и выводит запись, что записи в таблицу добавлены.



Для того, чтобы проверить, как работает новый триггер необходимо создать новый пустой запрос и в нём выполним добавление данных в таблицу «Предметы», как представлено на рисунке ниже.



### Порядок выполнения работы

1. Изучить теоретический материал.
2. Выполнить все примеры и проверить результаты запросов.
3. Получить у преподавателя задание для индивидуальной работы.
4. Ответить на контрольные вопросы.

### Контрольные вопросы

1. Дайте определение понятию *хранимая процедура*.
2. Назовите основное отличие пакета от хранимых процедур.
3. Какой префикс имеют системные хранимые процедуры?
4. Для чего предназначены системные хранимые процедуры?
5. Назовите типы временных хранимых процедур?
6. С какого символа должно начинаться имя временной хранимой процедуры?
7. Назовите виды хранимых процедур?
8. Какую команду необходимо выполнить для того, чтобы разместить создаваемую хранимую процедуру в конкретной базе данных?
9. С какого символа должны начинаться имена параметров для передачи входных и выходных данных в создаваемой хранимой процедуре?
10. Сколько параметров можно задавать в хранимой процедуре?
11. Какое ключевое слово представляет собой значение, которое будет принимать соответствующий параметр по умолчанию?
12. Какое ключевое слово определяет начало тела хранимой процедуры?
13. Перечислите цели создания хранимых процедур.
14. Дайте определение понятию параметр хранимой процедуры.
15. Назовите этапы жизненного цикла хранимой процедуры.
16. Назовите команды создания, выполнения, удаления и изменения хранимой процедуры.
17. Поясните возможность использования предложения WITH RESULTS SETS инструкции EXECUTE.
18. Дайте определение понятию триггер.
19. Назовите два типа триггеров по типу действия, которое запускает триггер. Поясните.
20. Назовите инструкции создания, модификации и удаления триггера.
21. Поясните суть виртуальных таблиц deleted и inserted.
22. Назовите основные отличия триггеров AFTER и триггеров INSTEAD OF.
23. Назовите основные отличия триггеров DML и DDL.

Преподаватель

С.В. Банцевич

Рассмотрено на заседании цикловой  
комиссии программного обеспечения  
информационных технологий №10  
Протокол № от « » \_\_\_\_\_ 201\_  
Председатель ЦК С.В. Банцевич