

Лабораторная работа 9

Хранимые процедуры и определяемые пользователем функции, триггеры

1.1. Теория (Хранимые процедуры)

При программировании в SQL Server введенный код сначала компилируется, потом запускается. На языке Transact-SQL также есть возможность написанный блок кода сохранить и заранее скомпилировать.

Процедуры аналогичны конструкциям в других языках программирования и выполняют следующие задачи:

- обрабатывают входные параметры и возвращают значения в виде выходных параметров;
- содержат инструкции, которые выполняют операции в базе данных, в отличие от пользовательских функций;
- возвращают сведения об успешном или неуспешном завершении.

Хранимая процедура создается с помощью команды **CREATE PROCEDURE** или **CREATE PROC**, которая имеет следующий упрощенный вид:

```
CREATE {PROC | PROCEDURE} <название>  
[/<@параметр> <тип> [= <значение по умолчанию>] [OUT |  
OUTPUT]] AS  
[BEGIN]  
<команды>  
[END]
```

При создании процедуры после команды **CREATE** указывается тип создаваемого объекта с помощью ключевого слова **PROCEDURE** или его сокращенного варианта **PROC**.

Названия процедур должны соответствовать требованиям, предъявляемым к идентификаторам, и должны быть уникальными в базе данных. **При этом не следует пользоваться префиксом «sp_»**. Этим префиксом в SQL Server обозначаются системные процедуры.

В хранимую процедуру можно передать до 2100 параметров. При выполнении процедуры значение каждого из объявленных параметров должно быть указано пользователем, если для параметра не определено значение по умолчанию.

Ключевое слово **OUT** (можно использовать и **OUTPUT**) показывает, что параметр процедуры является выходным.

Для выполнения хранимой процедуры используется ключевое слово **EXECUTE** (или **EXEC**). Процедуру также можно вызывать и выполнять без ключевого слова, если она является первой инструкцией. Синтаксис команды **EXECUTE** имеет следующий вид:

```
EXECUTE [/<@статус возврата>=] <название процедуры>  
[/<@параметр>=] <значение>| <@переменная> [OUTPUT] | [DEFAULT]
```

В отличие от вызова функций, при вызове хранимых процедур с указанием названия параметра ([<@параметр>=] <значение>), последовательность параметров можно не соблюдать.

Для выходных параметров при вызове указывается ключевое слово **OUTPUT**.

Если для параметра указано значение по умолчанию, можно его использовать с помощью ключевого слова **DEFAULT**.

Для удаления хранимых процедур используется команда DROP PROCEDURE. Упрощенный синтаксис имеет следующий вид:

DROP PROC | PROCEDURE [IF EXISTS] <название хранимой процедуры>

Ключевые слова **IF EXISTS** удаляют хранимую процедуру только в том случае, если она уже существует.

1.2. Практика (Хранимые процедуры) -аудиторная работа

1) Создать Хранимую процедуру для вывода информации о сервере, о базе данных и о текущем пользователе, и вызовите

```
1  --хранимую процедуру для вывода информации о сервере, о базе данных
2  --и о текущем пользователе, и вызовите ее
3
4  CREATE PROC PROC1
5  AS
6  BEGIN
7
8      SELECT
9          @@Servername AS Сервер, @@Version AS ВерсияСУБД
10         , Db_Name() AS БазаДанных
11         , User AS ПользовательБД
12         , System_User AS СистемныйПользователь
13     END
```

Сообщения
Выполнение команд успешно завершено.

```
13  --вызов процедуры
14
15  EXEC PROC1
16
```

152 %

	Сервер	ВерсияСУБД	БазаДанных	ПользовательБД	СистемныйПользователь
1	LAPTOP-HALRD7V5	Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X...	TestDatabas	dbo	sa

Далее, воспользуемся таблицей *Tabl_Kontinent\$* (см. Лабораторная 6.3 Агрегатные функции. Группировка данных. Фильтрация групп)

```

1  /***** Скрипт для команды SelectTopNRows из среды S
2  SELECT TOP (1000) [Nazvanie]
3      ,[Stolica]
4      ,[PL]
5      ,[KolNas]
6      ,[Kontinent]
7  FROM [TestDatabas].[dbo].[Tabl_Kontinent$]

```

152 %

Результаты Сообщения

	Nazvanie	Stolica	PL	KolNas	Kontinent
1	Австрия	Вена	83858	8741753	Европа
2	Азербайджан	Баку	86600	9705600	Азия
3	Албания	Тирана	28748	2866026	Европа
4	Алжир	Алжир	2381740	39813722	Африка
5	Ангولا	Луанда	1246700	25831000	Африка
6	Аргентина	Буэнос-Айрес	2766890	43847000	Южная Америка
7	Афганистан	Кабул	647500	29822848	Азия
8	Бангладеш	Дакка	144000	160221000	Азия
9	Бахрейн	Манама	701	1397000	Азия
10	Белиз	Бельмопан	22966	377968	Северная Америка
11	Белоруссия	Минск	207595	9498400	Европа
12	Бельгия	Брюссель	30528	11250585	Европа
13	Бенин	Порто-Ново	112620	11167000	Африка
14	Болгария	София	110910	7153784	Европа
15	Боливия	Сукре	1098580	10985059	Южная Америка
16	Ботсвана	Габороне	600370	2209208	Африка
17	Бразилия	Бразилиа	8511965	206081432	Южная Америка
18	Буркина-Фасо	Уагадугу	274200	19034397	Африка
19	Бутан	Тхимпху	47000	784000	Азия
20	Великобритания	Лондон	244820	65341183	Европа
21	Венгрия	Будапешт	93030	9830485	Европа
22	Венесуэла	Каракас	912050	31028637	Южная Америка
23	Восточный Тим...	Дили	14874	1167242	Азия
24	Вьетнам	Ханой	329560	91713300	Азия

2) Напишите хранимую процедуру, которая возвращает количество стран, содержащих в названии заданную букву, и вызовите ее

```

18  --хранимая процедура, которая возвращает количество стран
19  --содержащих в названии заданную букву,
20  Go
21  CREATE PROC PROC2
22      @Буква AS CHAR(1),
23      @Количество AS INT OUTPUT
24  AS
25  BEGIN
26      SELECT
27          @Количество = COUNT(*)
28      FROM
29          Tabl_Kontinent$
30      WHERE
31          CHARINDEX(@Буква, Nazvanie) > 0
32  END
33

```

Сообщения

Выполнение команд успешно завершено.

```

35
36 --вызов процедуры PROC2
37 DECLARE @K AS INT
38 DECLARE @Б AS CHAR(1)
39 SET @Б = 'y'
40 EXECUTE PROC2 @Б, @K OUTPUT
41 SELECT
42     @K AS [Количество стран]
43
44

```

Количество стран
4

3) Напишите хранимую процедуру для вывода трех стран с наименьшей площадью в заданной части света= Европа, вызовите ее.

```

44 --Напишите хранимую процедуру для вывода трех стран
45 --с наименьшей площадью в заданной части света= Европа
46 GO
47 CREATE PROC PROC3
48     @Конт AS VARCHAR(50) = 'Европа'
49 AS
50 BEGIN
51     SELECT TOP 3
52         Nazvanie
53         ,Stolica
54         ,PL
55         ,KolNas
56         ,Kontinent
57 FROM
58     Tabl_Kontinent$
59 WHERE
60     Kontinent = @Конт
61 ORDER BY
62     PL
63 END

```

Сообщения

Выполнение команд успешно завершено.

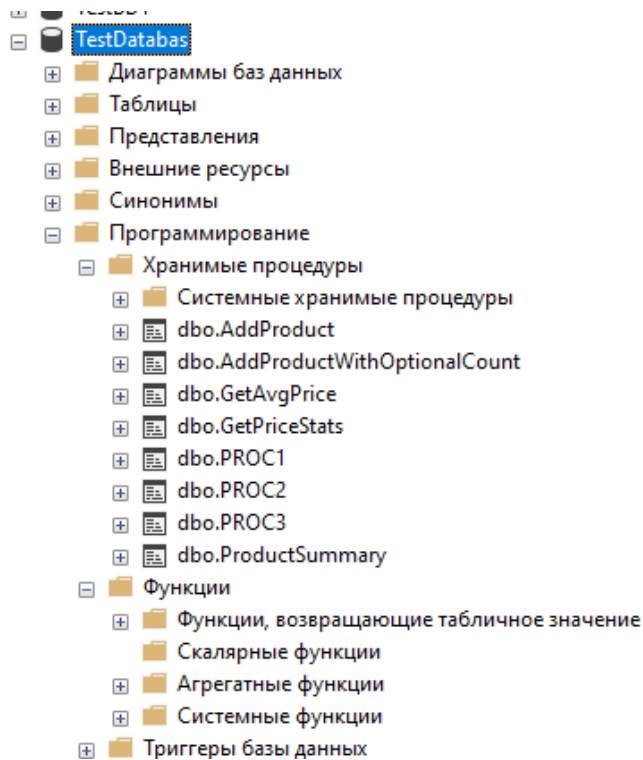
64

65

66

```
--вызов процедуры PROC3  
EXECUTE PROC3 DEFAULT
```

! %				
Результаты				
Сообщения				
Nazvanie	Stolica	PL	KolNas	Kontinent
Албания	Тирана	28748	2866026	Европа
Бельгия	Брюссель	30528	11250585	Европа
Австрия	Вена	83858	8741753	Европа



1.3 Практика (хранимые процедуры) -самостоятельная работа

- 1) Напишите хранимую процедуру, которая выводит данные всех стран
- 2) Напишите хранимую процедуру, которая принимает число, и возвращает количество цифр в нем через параметр OUTPUT
- 3) Напишите хранимую процедуру, которая создает таблицу «TestTabl», заполняет ее странами, названия которых начинаются с первой буквой вашей фамилии

2.2 Теория: Определяемые пользователем функции

Transact-SQL предоставляет возможность создать собственную функцию для решения задач.

Пользовательская скалярная функция возвращает в качестве ответа единственное значение.

Пользовательские функции могут быть вложенными, то есть из одной функции может быть вызвана другая. Вложенность функций не может превышать 32 уровней.

Упрощенный синтаксис создания пользовательской скалярной функции имеет следующий вид:

```
CREATE FUNCTION <название>  
(  
  [{<@параметр> [AS] <тип> [= <значение по умолчанию>]}]  
)  
RETURNS <тип возврата>  
[AS]  
BEGIN  
  <команды>  
RETURN  
  <значение> END
```

Значение, переменная или выражение после ключевого слова RETURN имеет такой же тип, который указан после ключевого слова RETURNS.

Созданная функция может быть вызвана, как и обычная встроенная функция, но при этом должны вызываться с помощью имени владельца. Простой вызов имеет следующий вид:

```
SELECT <владелец>.<функция>(<параметры>)
```

Для пользовательских функций допускается не более 2100 параметров.

При выполнении функции значение каждого из объявленных параметров должно быть указано пользователем, если для них не определены значения по умолчанию. Имя параметра, как и имя переменных, использует знак @ как первый символ.

Для INLINE функций ключевого слова RETURNS указывается тип TABLE без указания списка столбцов. Тело такой функции представляет собой *единственный оператор SELECT*, который начинается сразу после ключевого слова RETURN. Упрощенный синтаксис создания пользовательской функции INLINE имеет следующий вид:

```

CREATE FUNCTION <название>
(
  [{<@параметр> [AS] <тип> [= <значение по умолчанию>]}]
)
RETURNS TABLE
AS
RETURN
(
  SELECT
  <список столбцов>
  FROM
  <таблица>
  WHERE
  <условие>
)

```

В MULTI-STATEMENT функциях после ключевого слова RETURNS указывается тип TABLE с определением столбцов и их типов данных.

Упрощенный синтаксис создания пользовательской MULTI-STATEMENT функции имеет следующий вид:

```

CREATE FUNCTION <название>
(
  [{<@параметр> [AS] <тип> [= <значение по умолчанию>]}]
)
RETURNS <@таблица> TABLE (<определение таблицы>)
AS
BEGIN
  <команды>
RETURN
END

```

Для MULTI-STATEMENT функций оператор RETURN не имеет аргумента. Значение возвращаемой переменной функции возвращается как значение функции.

Для удаления пользовательских функций используется команда **DROP FUNCTION**.

Упрощенный синтаксис имеет следующий вид:

```

DROP FUNCTION [IF EXISTS] <название функции>

```

2.3 Практика: Определяемые пользователем функции (аудиторная работа)

1) Напишите функцию для вывода списка стран с площадью в интервале заданных значений, и вызовите ее

```
1  --Use TestDatabas
2  --функция для вывода списка стран с площадью в интервале заданных значений
3  Go
4  Create Function Fun1
5  (
6      @A1 AS FLOAT,
7      @B1 AS FLOAT
8  )
9  RETURNS TABLE
10 AS
11 RETURN (
12     SELECT
13         Nazvanie
14         ,Stolica
15         ,PL
16         ,KolNas
17         ,Kontinent
18 FROM
19     Tabl_Kontinent$
20 WHERE
21     PL BETWEEN @A1 AND @B1
22 )
23 Go
```

2 %

Сообщения

Выполнение команд успешно завершено.

Представления

Внешние ресурсы

Синонимы

Программирование

Хранимые процедуры

Функции

Функции, возвращающие табличное значение

dbo.Fun1

Скалярные функции

Агрегатные функции

Системные функции

Триггеры базы данных

Сборки

Типы

Правила

```
24
25 --вызов функции Fun1
26 SELECT *
27 FROM   dbo.Fun1(100, 1000)
28
```

%

Результаты

Сообщения

Nazvanie	Stolica	PL	KolNas	Kontinent
Бахрейн	Манама	701	1397000	Азия

2) Напишите функцию для вывода столицы данной страны, и вызовите ее

```
29 --Напишите функцию для вывода столицы данной страны, и вызовите ее
30 Go
31 CREATE FUNCTION Fun2
32 (
33     @Страна AS VARCHAR(50)
34 )
35 RETURNS VARCHAR(50)
36 AS
37 BEGIN
38     DECLARE @S AS VARCHAR(50)
39     SELECT
40         @S = Stolica
41     FROM
42         Tabl_Kontinent$
43     WHERE
44         Nazvanie = @Страна
45     RETURN @S
46 END
47 Go
48
49
```

152 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-13T20:45:25.5518068+03:00

Функции

- Функции, возвращающие табличное значение
 - dbo.Fun1
- Скалярные функции
 - dbo.Fun2
- Агрегатные функции
- Системные функции
- Триггеры базы данных
- Сборки

```
48
49 --вызов функции Fun2
50 SELECT dbo.Fun2('Австрия')
51
```

52 %

Результаты Сообщения

	(Отсутствует имя столбца)
1	Вена

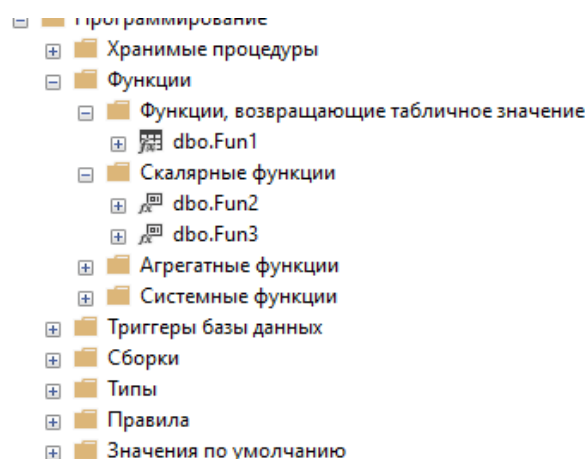
3) Напишите функцию для вычисления плотности населения, и вызовите ее

```
52 --Напишите функцию для вычисления плотности населения
53 Go
54 CREATE FUNCTION Fun3
55 (
56     @Население AS INT,
57     @Площадь AS FLOAT
58 )
59 RETURNS FLOAT
60 AS
61 BEGIN
62     DECLARE @P AS FLOAT
63     SET @P = ROUND(CAST(@Население AS FLOAT) / @Площадь, 2)
64     RETURN @P
```

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T13:24:01.6500040+03:00



```
68
69 --вызов функции Fun3
70
71 SELECT Nazvanie,
72        Stolica,
73        PL,
74        KolNas,
75        Kontinent,
76        dbo.Fun3 (KolNas, PL) AS Плотность
77 FROM
78     Tabl_Kontinent$
79
```

152 %

Результаты

	Nazvanie	Stolica	PL	KolNas	Kontinent	Плотность
1	Австрия	Вена	83858	8741753	Европа	104,24
2	Азербайджан	Баку	86600	9705600	Азия	112,07
3	Албания	Тирана	28748	2866026	Европа	99,69
4	Алжир	Алжир	2381740	39813722	Африка	16,72
5	Ангولا	Луанда	1246700	25831000	Африка	20,72
6	Аргентина	Буэнос-Айрес	2766890	43847000	Южная Америка	15,85
7	Афганистан	Кабул	647500	29822848	Азия	46,06
8	Бангладеш	Дакха	144000	160221000	Азия	1112,65
9	Бахрейн	Манама	701	1397000	Азия	1992,87
10	Белиз	Бельмопан	22966	377968	Северная Америка	16,46
11	Белоруссия	Минск	207595	9498400	Европа	45,75
12	Бельгия	Брюссель	30528	11250585	Европа	368,53
13	Бенин	Порто-Ново	112620	11167000	Африка	99,16
14	Болгария	София	110910	7153784	Европа	64,5
15	Боливия	Сукре	1098580	10985059	Южная Америка	10
16	Ботсвана	Габороне	600370	2209208	Африка	3,68
17	Бразилия	Бразилиа	8511965	206081432	Южная Америка	24,21
18	Буркина-Фасо	Уагадугу	274200	19034397	Африка	69,42
19	Бутан	Тхимпху	47000	784000	Азия	16,68
20	Великобритания	Лондон	244820	65341183	Европа	266,89
21	Венгрия	Будапешт	93030	9830485	Европа	105,67
22	Венесуэла	Каракас	912050	31028637	Южная Америка	34,02

2.4 Практика: Определяемые пользователем функции. Самостоятельная работа.

- 1) Напишите функцию для возврата списка стран с площадью меньше заданного числа и вызовите ее.
- 2) Напишите функцию для возврата таблицы с названием страны и плотностью населения, и вызовите ее

3. Теория. Триггеры

Триггер – это вид хранимой процедуры, который вызывается автоматически при определенных событиях. *Часто триггеры применяются для автоматической поддержки целостности и защиты БД.*

В MS SQL Server существует *три вида триггеров*, которые отличаются по функциям и по синтаксису создания и изменения:

Триггеры DML вызываются при выполнении команд INSERT, UPDATE или DELETE. Можно создать триггер, реагирующий на две или на все три команды.

Триггеры DDL реагируют на события изменения структуры БД: создание, изменение или удаление отдельных объектов БД.

Триггеры входа в систему запускаются при соединении пользователя с экземпляром сервера. *Их можно применять для дополнительной проверки полномочий пользователей.*

Триггеры DML можно вызвать после событий (FOR | AFTER), или вместо него (INSTEAD OF).

Триггер AFTER выполняется после успешного завершения вызвавшего его события. Можно определить несколько AFTER-триггеров для каждой операции.

Триггер INSTEAD OF вызывается вместо выполнения команд. Для каждой операции INSERT, UPDATE, DELETE можно определить только один INSTEAD OF-триггер.

Упрощенный синтаксис создания триггера имеет следующий вид:

```
CREATE TRIGGER <название триггера> ON <название таблицы>  
<FOR | AFTER | INSTEAD OF> <INSERT | UPDATE | DELETE>  
AS  
[BEGIN]  
<команды>  
[END]
```

Ключевое слово *FOR* или *AFTER* указывает, что триггер DML срабатывает только после успешного запуска всех операций в инструкции SQL, по которой срабатывает триггер.

Ключевое слово *INSTEAD OF* указывает, что триггер DML выполняется вместо инструкции SQL, по которой он срабатывает, то есть переопределяет действия запускающих инструкций.

В определении триггера ключевые слова INSERT | UPDATE | DELETE определяют инструкции изменения данных, при применении которых к таблице или представлению срабатывает триггер DML. Указание хотя бы одного варианта обязательно. В определении триггера разрешены любые сочетания вариантов в любом порядке.

Если триггер выполняется для события **добавления** данных (команды INSERT), в теле триггера **доступна виртуальная таблица INSERTED**, которая содержит список добавленных данных.

Если триггер выполняется для события **удаления** данных (команды DELETE), в теле триггера **доступна виртуальная таблица DELETED**, которая содержит список удаленных данных.

Если триггер выполняется для события изменения данных (команды UPDATE), в теле триггера доступны **две виртуальные таблицы INSERTED и DELETED**, которые содержат список новых и старых данных, соответственно.

Если при определенных обстоятельствах выполнение триггера нежелательно, то можно его **отключить**.

Для этого используется команда DISABLE TRIGGER, его синтаксис:
DISABLE TRIGGER <название триггера> ON <название таблицы>

А когда триггер снова понадобится, его можно **включить** с помощью команды ENABLE TRIGGER, его синтаксис:

ENABLE TRIGGER <название триггера> ON <название таблицы>

Для **удаления** триггера используется команда DROP TRIGGER, его синтаксис:

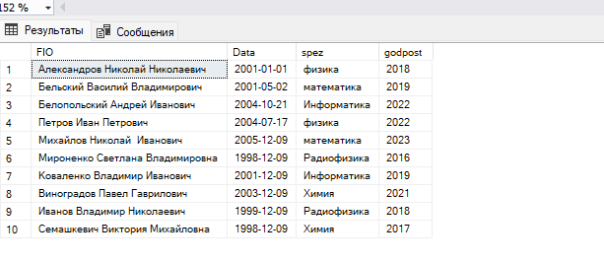
DROP TRIGGER <название триггера>

3.1 Практика. Триггеры. Аудиторная работа.

Воспользуемся данными из ,Лаб. Работы 2, а именно табл. Student

1) Напишите триггер на добавление записи в таблицу «Student». Данный триггер, в случае успешного добавления данных, выводит «Запись добавлена»

```
1  /***** Скрипт для команды SelectTopNRows из sp
2  SELECT TOP (1000) [FIO]
3      ,[Data]
4      ,[spez]
5      ,[godpost]
6  FROM [Ucheb_12_Ivanov].[dbo].[Student]
```



	FIO	Data	spez	godpost
1	Александров Николай Николаевич	2001-01-01	физика	2018
2	Бельский Василий Владимирович	2001-05-02	математика	2019
3	Белопольский Андрей Иванович	2004-10-21	информатика	2022
4	Петров Иван Петрович	2004-07-17	физика	2022
5	Михайлов Николай Иванович	2005-12-09	математика	2023
6	Мироненко Светлана Владимировна	1998-12-09	Радиофизика	2016
7	Коваленко Владимир Иванович	2001-12-09	информатика	2019
8	Виноградов Павел Гаврилович	2003-12-09	Химия	2021
9	Иванов Владимир Николаевич	1999-12-09	Радиофизика	2018
10	Семашкин Виктор Михайлович	1998-12-09	Химия	2017

```
1 USE Ucheb_12_Ivanov
2 -- триггер выводит «Запись добавлена»
3 GO
4 CREATE TRIGGER trig1 ON Student
5 FOR INSERT
6 AS
7 BEGIN
8     PRINT 'Запись добавлена'
9 END
10
```

2 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T15:51:54.7047273+03:00

- 2) **Самостоятельно** проверьте работу триггера Trig1, для этого добавьте запись в табл. Student
- 3) Напишите триггер на удаление записи из таблицы «Student». Данный триггер, при попытке удаления данных, выводит «Нельзя удалить данные»

```
10
11 --триггер на удаление записи из таблицы «Student
12 --Данный триггер, при попытке удаления данных,
13 --выводит «Нельзя удалить данные»
14 GO
15 CREATE TRIGGER trig2
16 ON Student
17 INSTEAD OF DELETE
18 AS
19 BEGIN
20     PRINT 'Нельзя удалить данные'
21 END
22
```

152 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T16:47:19.7414340+03:00

- 4) **Самостоятельно** проверьте работу триггера Trig2
- 5) **Создать** таблицу «Студент_Архив», которая будет содержать все данные об удаленных Студентах и даты их удаления.
- Написать** триггер, который будет фиксировать в таблице «Студент_Архив1» данные студента, удаленного из таблицы «Студенты»

```
27 CREATE TABLE Студент_Архив1
28 (
29     Fio NVARCHAR(40) NULL,
30     Data Date NULL,
31     spez NVARCHAR(20) NULL,
32     godpost int NULL,
33     Удалено DATETIME NOT NULL
34 )
```

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T17:17:33.6555077+03:00

```
36
37 --триггер для фиксации в таблице «Студент_Архив» данные студента,
38 --удаленного из таблицы «Студенты»
39
40 GO
41 CREATE TRIGGER trig33
42 ON Student
43 FOR DELETE
44 AS
45 BEGIN
46     INSERT
47         Студент_Архив1
48     SELECT
49         Fio,
50         Data,
51         spez,
52         godpost,
53         GETDATE () AS Удалено
54     FROM
55         DELETED
56
57 END
58
```

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T17:24:32.3267247+03:00

```

36
37 --триггер для фиксации в таблице «Студент_Архив» данные студента,
38 --удаленного из таблицы «Студенты»
39
40 GO
41 CREATE TRIGGER trig31
42 ON Student
43 FOR DELETE
44 AS
45 BEGIN
46     INSERT
47         Студент_Архив
48     SELECT
49         Fio,
50         Data,
51         spez,
52         godpost,
53         GETDATE () AS Удалено
54     FROM
55         DELETED
56
57 END
58

```

6

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T17:28:20.6672919+03:00

5) Самостоятельно проверьте работу триггера Trig31

```
39 [
40 GO
41 - CREATE TRIGGER trig33
42 ON Student
43 FOR DELETE
44 AS
45 - BEGIN
46 - INSERT
47 Студент_Архив1
48
49 SELECT
50 Fio,
51 Data,
52 spez,
53 godpost,
54 GETDATE () AS Удалено
55 FROM
56 DELETED
57 END
58
```

152 %

Сообщения

Выполнение команд успешно завершено.

Время выполнения: 2023-05-14T17:24:32.3267247+03:00