|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе №3 «Триггеры» по дисциплине «Базы данных и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-6  Шукшина М.И.  «13» февраля 2018 | | |  | Проверил ассистент кафедры МОВС  Костарева Т.А.  «13» февраля 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

ПОСТАНОВКА ЗАДАЧИ

Решить проблему рассогласованности спец. таблицы и отслеживаемой таблицы в случае, если в последнюю были внесены изменения в обход разработанной ранее хранимой процедуры (ХП). Для этого автоматически создавать триггер для таблицы, имя которой передаётся в ХП в качестве параметра для каждой новой уникальной пары 'имя таблицы' + 'имя столбца’. Имя триггера, помимо осмысленной части, должно содержать GUID.

Провести тестирование корректности работы программы.

РЕАЛИЗАЦИЯ ТРИГГЕРА

При вызове ХП с новыми параметрами для таблицы, передаваемой одним из параметров, создается триггер. Он, при необходимости, обновляет tek\_max в строке спец. таблицы, соответствующей, переданным параметрам в ХП. То есть, если в колонке таблицы, которая находится в спец. таблице произойдут изменения (обновление данных, добавление данных, удаление данных), то у нас должны обновиться данные в спец. таблице, иначе она будет хранить ложную информацию. Триггер реагирует на вышеперечисленные изменения, и, если они произошли, находит новое максимальное значение в измененной таблице и обновляет данные в спец. таблице.

ТЕСТИРОВАНИЕ КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММЫ

1. Автоматическое создание триггера для таблицы, имя которой передаётся в ХП в качестве параметра для каждой новой уникальной пары 'имя таблицы' + 'имя столбца’. Имя триггера, помимо осмысленной части, должно содержать GUID.

CREATE PROCEDURE [dbo].[Store\_p]

@name\_t VARCHAR(50), @name\_c VARCHAR(50), @res int OUTPUT

AS

BEGIN

DECLARE @tek int

SELECT @tek = tek\_max FROM Stores\_pr WHERE name\_table = @name\_t AND name\_column = @name\_c

IF (@tek is not null)

begin

SET @res = @tek + 1

--обновляем данные во вспомогательной таблице

UPDATE Stores\_pr SET tek\_max = @res WHERE name\_table = @name\_t AND name\_column = @name\_c

end

ELSE

BEGIN

--находим максиальное значение в нужном столбце нужной таблицы

DECLARE @tek1 nvarchar(max)

SET @tek1 = 'SELECT @RES = MAX(' + QUOTENAME(@name\_c) + ') FROM ' + QUOTENAME(@name\_t)

EXECUTE sp\_executesql @tek1, N'@RES INT OUTPUT', @RES = @res OUTPUT

DECLARE @trig nvarchar (max)

SET @trig = '

CREATE TRIGGER' + QUOTENAME((@name\_t) + CAST(NEWID() AS nvarchar(128))) +

'ON ' + QUOTENAME(@name\_t) + 'AFTER INSERT, UPDATE, DELETE

AS

BEGIN

--находим новое макимальное знчение в таблице @name\_t в столбце @name\_c

DECLARE @max int, @max1 int

SELECT @max = MAX(' + QUOTENAME(@name\_c) + ') FROM' + QUOTENAME(@name\_t) + '

--если максимальное значение в таблице @name\_t в столбце @name\_c изменилось,

--то мы должны обновить данные в таблице Stores\_pr

UPDATE Stores\_pr SET tek\_max = @max WHERE name\_table = ' + QUOTENAME(@name\_t, '''') + 'AND name\_column = ' + QUOTENAME(@name\_c, '''') +

'END'

EXEC(@trig)

--находим максимальное значение во вспомогательной таблице

DECLARE @result1 int

EXECUTE [dbo].[Store\_p] 'Stores\_pr', 'id', @result1 OUTPUT

DECLARE @inf int

SELECT @inf = count(\*) FROM Stores\_pr WHERE name\_table = @name\_t AND name\_column = @name\_c

SELECT @inf

if (@RES is not null)

begin

SET @res = @RES + 1

--записываем результат во вспомогательную таблицу

INSERT INTO Stores\_pr VALUES (@result1, @name\_t, @name\_c, @res)

end

else

begin

if (@inf > 0)

begin

--обновляем данные во вспомогательной таблице

UPDATE Stores\_pr SET tek\_max = 1 WHERE name\_table = @name\_t AND name\_column = @name\_c

SET @res = 1

end

else

begin --записываем результат во вспомогательную таблицу

INSERT INTO Stores\_pr VALUES (@result1, @name\_t, @name\_c, 1)

SET @res = 1

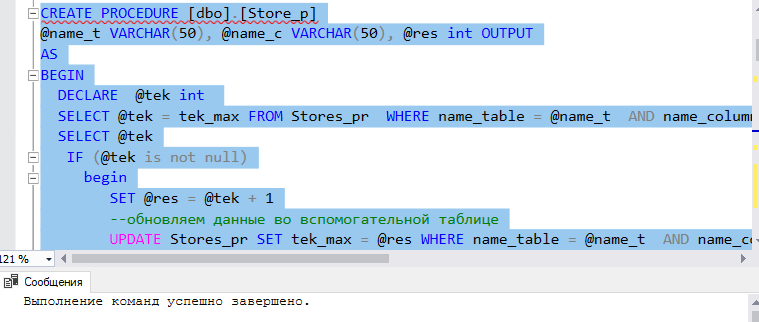
end

end

END

END

GO



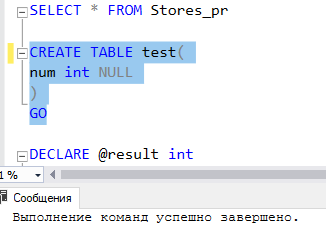
1. Создание таблицы test с полем *num*

CREATE TABLE test(

num int NULL

)

GO



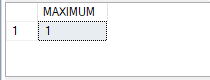
1. Вызов ХП с параметрами *‘test’, ‘num’*

DECLARE @result int

EXECUTE [dbo].[Store\_p] 'test', 'num', @result OUTPUT

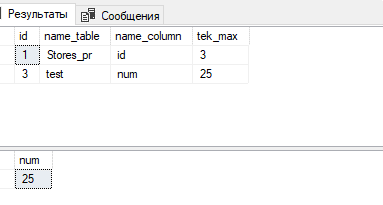
SELECT @result AS MAXIMUM

GO



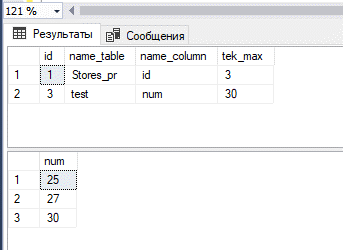
1. Добавим в таблицу *test* в колонку *num* значение 25

INSERT INTO test VALUES (25)



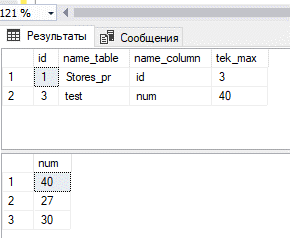
1. Добавим в таблицу *test*  в колонку *num* два значения 27 и 30

INSERT INTO test VALUES (27),(30)



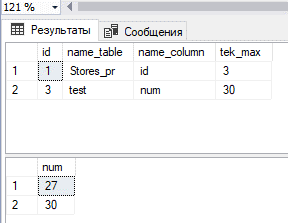
1. Изменим значение поля 25 в таблице *test* в колонке *num* на 40

UPDATE test SET num = 40 WHERE num = 25



1. Удалим значение 40 в таблице *test* в колонке *num*

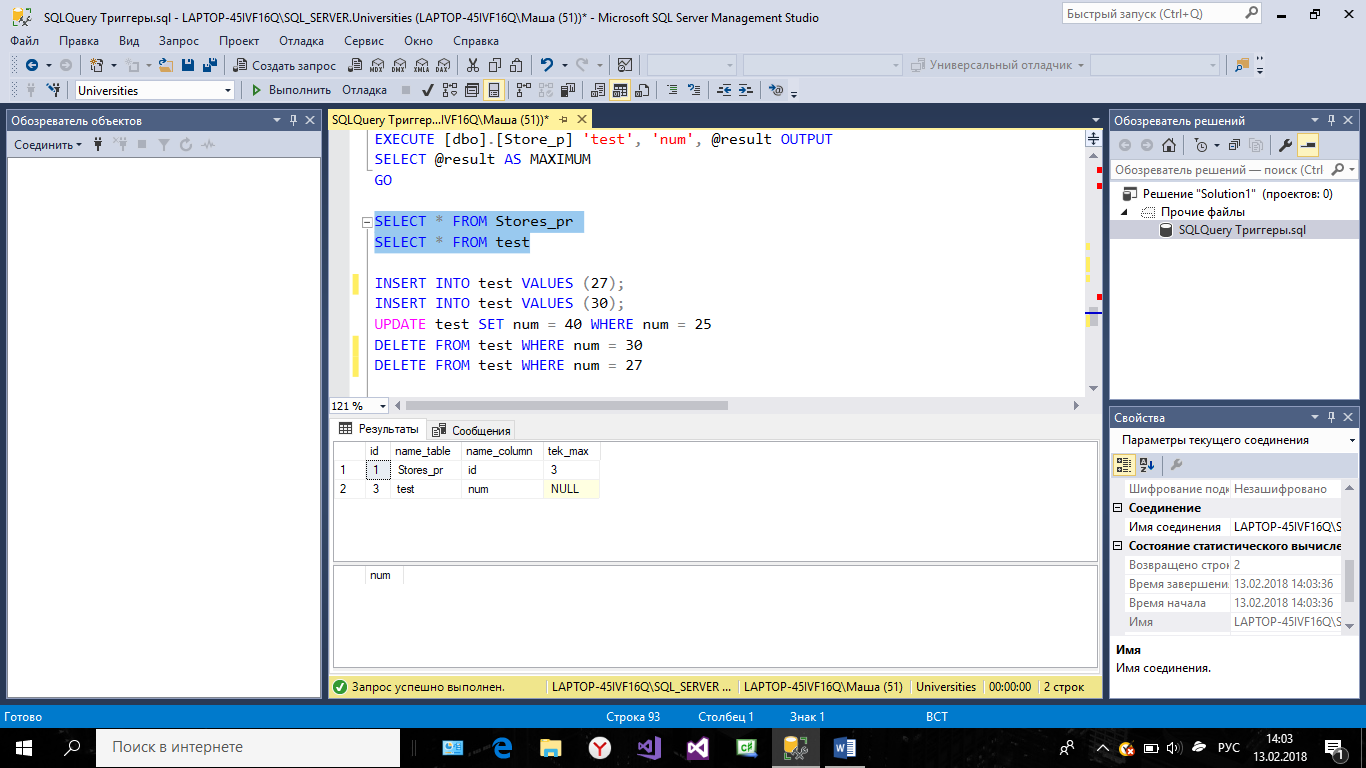
DELETE FROM test WHERE num = 40



1. Удалим все строки в таблице *test* в колонке *num*

DELETE FROM test WHERE num = 30

DELETE FROM test WHERE num = 27



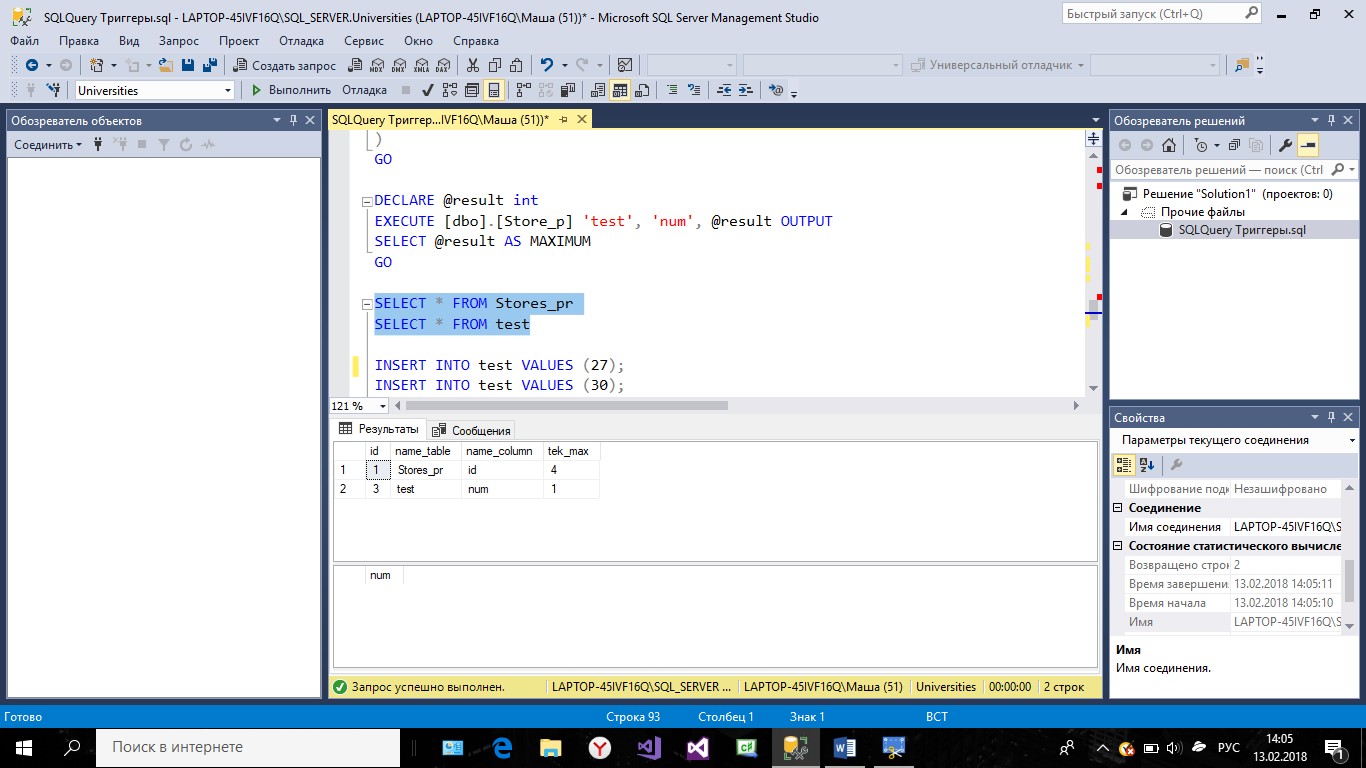
1. Вызов ХП с параметрами *‘test’, ‘num’*

DECLARE @result int

EXECUTE [dbo].[Store\_p] 'test', 'num', @result OUTPUT

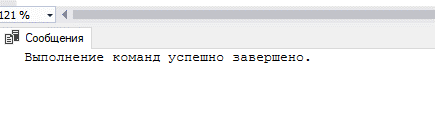
SELECT @result AS MAXIMUM

GO



1. Добавим еще одно поле num2 в таблицу

ALTER TABLE test ADD num2 int NULL

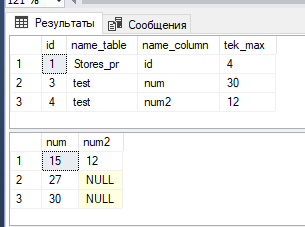


1. Вызов ХП с параметрами *‘test’, ‘num2’*



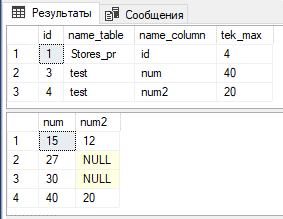
1. Добавим в таблицу *test*  строку 15, 12

INSERT INTO test VALUES (15,12)



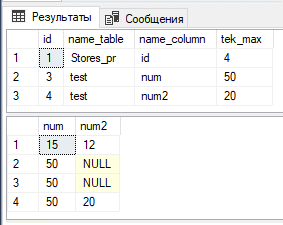
1. Добавим в таблицу *test*  строку 40, 20

INSERT INTO test VALUES (40,20)



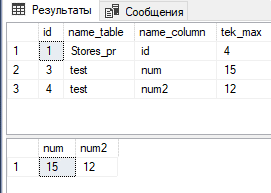
1. Изменим значение все значения которые больше 35 в колонке *num* в таблице *test* на 50

UPDATE test SET num = 50 WHERE num > 15



1. Удалим все строки значение которых равно 50 в таблице *test* в колонке *num*

DELETE FROM test WHERE num = 50



1. Удалим таблицу *test* , вспомогательную таблицу *Srore\_pr* и хранимую процедуру *[dbo].[Store\_p]*

DROP TABLE test

DROP TABLE Stores\_pr

DROP PROC [dbo].[Store\_p]

