Практическая работа 18
Создание триггеров в базах данных

Цель занятия: Получить практический опыт создания триггеров изменения и модификации в базах данных

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
Microsoft SQL Server Management Studio

Краткие теоретические сведения

Триггеры — это методы, с помощью которых разработчик приложений для MS SQL Server может обеспечить целостность базы данных. Это тип хранимой процедуры, которая активизируется при попытке изменения данных в таблице, для которой определен триггер. SQL Server выполняет эту процедуру при операциях добавления, обновления и удаления (INSERT, UPDATE, DELETE) в данной таблице. Поскольку триггер применяется после выполнения операции, он представляет собой последнее слово в модификации. Если триггер вызывает ошибку в запросе, SQL Server отказывается от обновления информации и возвращает приложению, выполняющему это действие, сообщение об ошибке. Если для таблицы определен триггер, то при выполнении соответствующей операции обойти его нельзя.

Создание триггера похоже на объявление хранимой процедуры и имеет следующий синтаксис:

```
CREATE TRIGGER имя_триггера
ON таблица
[WITH ENCRYPTION]
{
{FOR | AFTER | INSTEAD OF} {[DELETE] [,] [INSERT] [,] [UPDATE]}
[WITH APPEND]
[NOT FOR REPLICATION]
AS
{IF UPDATE (столбец_i)
[{AND | OR} UPDATE (столбец_j)]
[… n]
| IF (COLUMNS_UPDATED() {побитовый_оператор} битовая_маска)
{оператор_сравнения} битовая_маска_столбца [… n]
}
инструкции_SQL [… n]
```

}
}

имя_триггера — должно соответствовать стандартным соглашениям об именах объектов SQL Server и быть уникальным в базе данных;

таблица — название таблицы, для которой создается триггер;

WITH ENCRYPTION — эта опция дает разработчикам возможность запретить пользователям читать текст триггера после его загрузки на сервер. Опять же, отметим, что для того чтобы сделать текст триггера действительно невосстановимым, следует после шифрования удалить соответствующие ему строки из таблицы syscomments;

FOR DELETE, INSERT, UPDATE — ключевые слова, определяющие операцию модификации таблицы, при выполнении которой будет активизирован триггер;

WITH APPEND — эта опция необходима, только если установленный уровень совместимости не превышает 65 и используется для создания дополнительных триггеров;

NOT FOR REPLICATION — показывает, что триггер не активизируется при модификации таблицы в процессе репликации;

AS — ключевое слово, задающее начало определения триггера;

инструкции_SQL — в T-SQL триггер может содержать любое количество инструкций SQL, если они заключены в операторные скобки BEGIN ... END;

IF UPDATE (столбец) — для операций добавления и обновления данных можно определить дополнительные условия на конкретный столбец таблицы; при указании нескольких столбцов они разделяются логическими операторами;

IF (COLUMNS_UPDATED()…) — выше было показано, как можно с помощью конструкции IF UPDATE (столбец) определять, какие столбцы затрагиваются изменениями. Если необходимо проверять, изменяется ли какой-то один конкретный столбец, эта конструкция очень удобна. Однако при построении сложного условия, включающего много столбцов, данная конструкция получается слишком громоздкой. Для таких случаев предназначена конструкция IF (COLUMNS_UPDATED()…). Результатом функции COLUMNS_UPDATED() является набор битов, каждый из которых отвечает за один столбец таблицы; младший бит соответствует первому столбцу, старший — последнему. Если в операции, вызвавшей срабатывание триггера, была попытка изменить некоторый столбец, то соответствующий бит будет установлен в 1;

побитовый_оператор — побитовый оператор, определяющий операцию выделения нужных битов, полученных с помощью COLUMNS_UPDATED(). Обычно используется оператор &;

битовая_маска — в сочетании с побитовым оператором битовая маска позволяет выделить интересующие разработчика биты, то есть определить, изменялись ли в операции, вызвавшей срабатывание триггера, интересующие его столбцы;

оператор_сравнения и битовая_маска_столбца — функция COLUMNS_UPDATED() дает набор битов, соответствующий изменяемым

столбцам. С помощью битовой маски и побитового оператора над этим набором битов производится преобразование и получается некий промежуточный результат. С помощью оператора сравнения этот промежуточный результат сравнивается с битовой маской столбца. Если результат сравнения — истина, то набор инструкций SQL, составляющий тело триггера, будет выполнен, иначе — нет.

Пример. Триггеры вставки и обновления. Эти триггеры удобны, поскольку они могут поддерживать условия ссылочной целостности и обеспечивать правильность данных перед вводом в таблицу. Обычно триггеры применяются для обновления столбцов отсчета времени или для проверки данных в определенных столбцах на соответствие требуемому критерию.

В приведенном ниже примере триггер выполняется всегда, когда в таблицу Sales вставляется строка или выполняется ее модификация. Если дата заказа не находится в пределах первых 15 дней месяца, строка в таблицу не вводится.

```
CREATE TRIGGER Tri_Ins_Sales
ON Sales
FOR INSERT, UPDATE
AS
/* Объявить необходимые локальные переменные */
DECLARE @nDayOfMonth TINYINT
/* Найти информацию о добавленной записи */
SELECT @nDayOfMonth = DatePart (day, i.ord_date)
FROM Sales s, Inserted i
WHERE s.stor_id = i.stor_id
AND s.ord_num = i.ord_num
AND s.title_id = i.title_id
/* Проверить критерий отказа и в случае необходимости
послать сообщение об ошибке */
IF @nDayOfMonth > 15
BEGIN
/* Примечание: всегда сначала производите откат. Вы можете не знать, какого рода ошибка обработки произошла, что может вызвать неоправданно продолжительное время блокировки */
ROLLBACK TRAN
RAISERROR('Выполняются только заказы, поданные в первые
15 дней месяца', 16, 10 )
END
```

Если теперь попытаться вставить или обновить запись в таблице, при несоблюдении заданного условия получим соответствующее сообщение об ошибке.

Пример. Триггеры удаления. Триггеры удаления (*delete triggers*) обычно применяются в двух случаях: предотвращение удаления строк, которое может вызвать проблемы с целостностью данных, например строки, используемой в

качестве внешнего ключа к другим таблицам, и выполнение каскадных операций удаления дочерних (*children*) строк главной (*master*) строки. Такой триггер можно использовать для удаления всей информации о заказах из главной строки продаж.
Триггеры учитывают общую сумму всех строк, на которые действует запрошенная операция. Таким образом, они должны иметь возможность работать с различными комбинациями информации в таблице и возвращать необходимые данные. Например, при выполнении инструкции DELETE FROM Authors, триггер должен учесть, что инструкция уничтожит все строки из таблицы.
В следующем примере использование переменной @@ROWCOUNT позволяет предотвратить удаление более одной строки. Этот триггер выполняется всегда, когда пользователь пытается удалить строку из таблицы Stores. Если информация касается продаж, то триггер препятствует выполнению этого запроса.

```
CREATE TRIGGER Tri_Del_Stores
ON Stores
FOR DELETE
AS
/* Проверка количества модифицируемых строк и запрещение удаления более
одной строки за один раз */
IF @@ ROWCOUNT > 1
BEGIN
ROLLBACK TRAN
RAISERROR ('За один раз можно удалить только одну строку.', 16, 10 )
END
/* Объявление временной переменной для сохранения уничтожаемой информации
*/
DECLARE @ StorID char (4)
/* Получение значения удаляемой строки */
SELECT @StorID = d.stor_id
FROM Stores s, Deleted d
WHERE s.stor_id *= d.stor_id
IF EXISTS (SELECT *
FROM Sales
WHERE stor_id = @storID)
BEGIN
ROLLBACK TRAN
RAISERROR ('Эта информация не может быть удалена, поскольку имеется
соответствующая запись в таблице Sales.', 16, 10)
END
```

Примечание: Применение RAISERROR — это самый простой способ посылки вызывающему процессу или пользователю подробной и конкретной информации об ошибке. RAISERROR дает возможность указать текст сообщения, уровень опасности, состояние информации и скомбинировать все это для

пользователя в описательное сообщение. Эта инструкция также облегчает написание общих блоков обработки ошибок в приложениях-клиентах.

Пример. Триггеры удаления

```
CREATE TRIGGER primer2
  ON Sotrydnik
  AFTER DELETE
AS
BEGIN
   SET NOCOUNT ON;
if (select oklad from deleted)>='50000'
Print 'Нельзя удалять запись о сотрудниках, оклад которых более 50000'
rollback
END
GO
```

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания разработать триггеры для отслеживания изменений в своей базе данных.

Порядок выполнения работы

Обычно триггеры применяются для обновления столбцов отсчета времени или для проверки данных в определенных столбцах на соответствие требуемому критерию. Рассмотрим другие случаи использования триггеров. В некоторых базах данных при изменении данных в одной из таблиц автоматически должны корректироваться данные в других таблицах. Например, пусть имеется база данных библиотеки некоторого учебного заведения, в котором студенты читают книги. И в этой базе данных есть таблицы «Издания» (с полями инвентарный номер, автор, название, тип, год издания, количество страниц, количество экземпляров в наличии) и «Выдача» (код выдачи, инвентарный номер, код читателя, дата выдачи, дата возврата, количество). При оформлении выдачи книги

– необходимо проверить, а есть ли эта книга в наличии;

– в таблице «Издания» количество экземпляров в наличии должно уменьшиться на количество выданных книг;

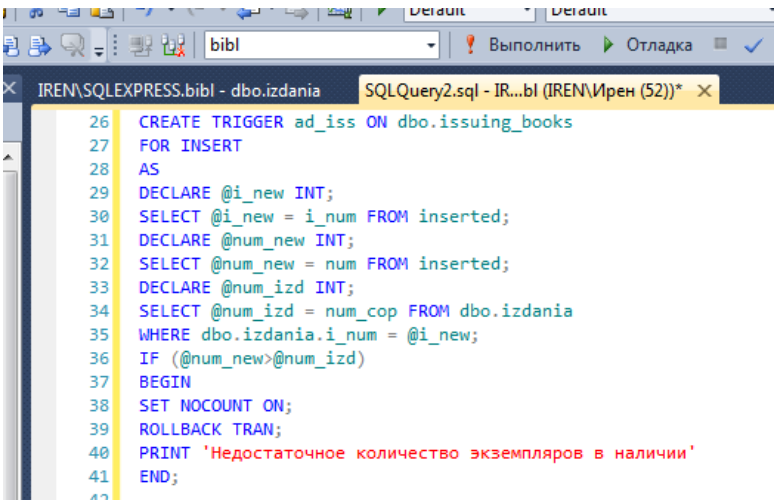
– в случае возврата книг в таблице «Издания» количество экземпляров в наличии должно увеличиться на количество возвращенных книг.

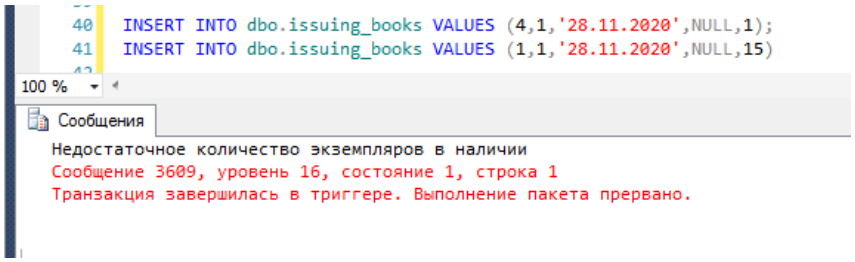
Создадим триггеры, выполняющие проверку наличия книг и обновление таблицы «Издания» при добавлении записи в таблицу «Выдача», а также триггер,

выполняющий увеличение количества книг в наличии в таблице «Издания» при обновлении таблицы «Выдача» (добавлении даты возврата).

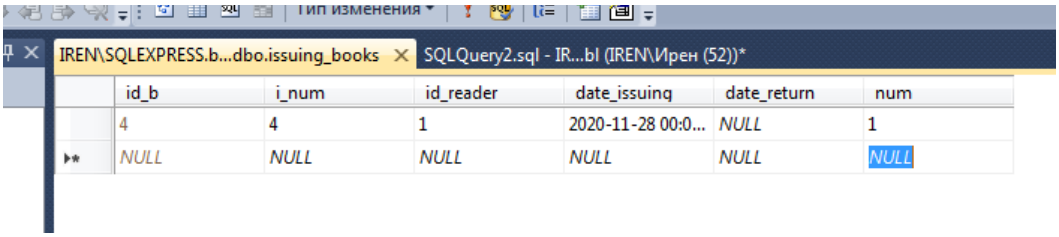
1 Триггер для проверки соответствия данных в связанных таблицах

Создадим триггер для проверки наличия книг в библиотеке. Если количество экземпляров нужной книги в таблице Издания меньше количества в заявке, то ввод данных не выполняется, при этом отображается сообщение о проблеме:



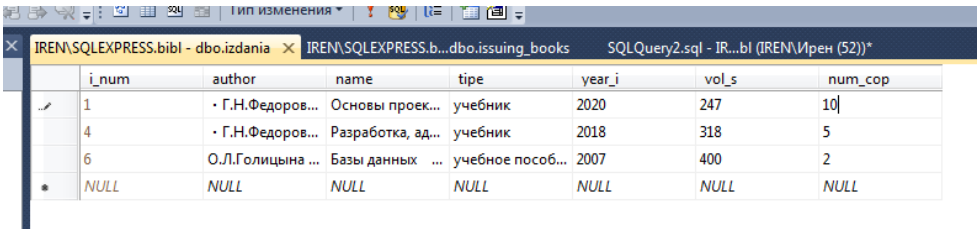


Если количество экземпляров в таблице Издания не меньше количества в заявке, то выполняется ввод данных в таблицу issuing_books:



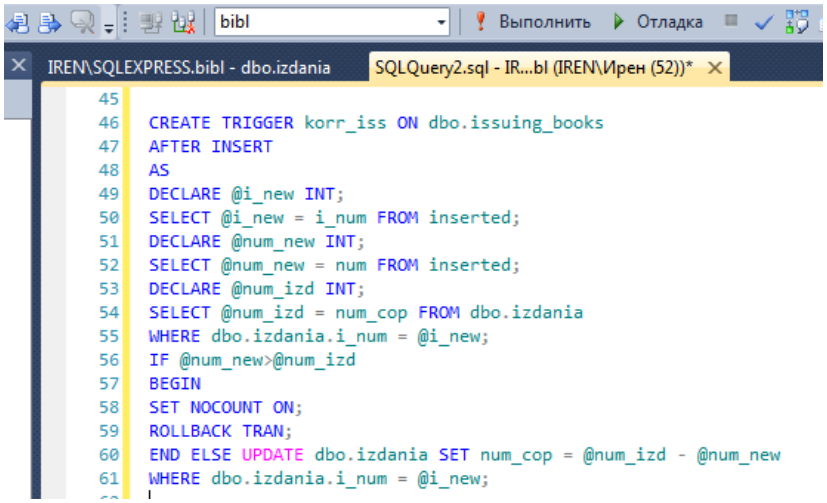
2 Корректировка данных в связанных таблицах

Организуем корректировку количества экземпляров при выполнении выдачи книг читателям. Пусть первоначально в таблице изданий имеется следующее количество книг:



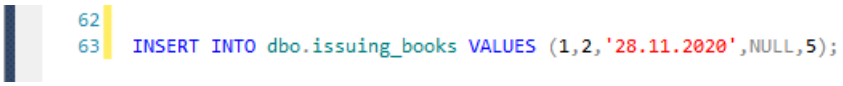
i_num	author	name	tipe	year_i	vol_s	num_cop
1	· Г.Н.Федоров...	Основы проек...	учебник	2020	247	10
4	· Г.Н.Федоров...	Разработка, ад...	учебник	2018	318	5
6	О.Л.Голицына ...	Базы данных ...	учебное пособ...	2007	400	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Создадим триггер для обновления таблицы «Издания». Если количество экземпляров нужной книги в таблице Издания меньше количества в заявке, то ввод данных не выполняется, иначе происходит обновление данных

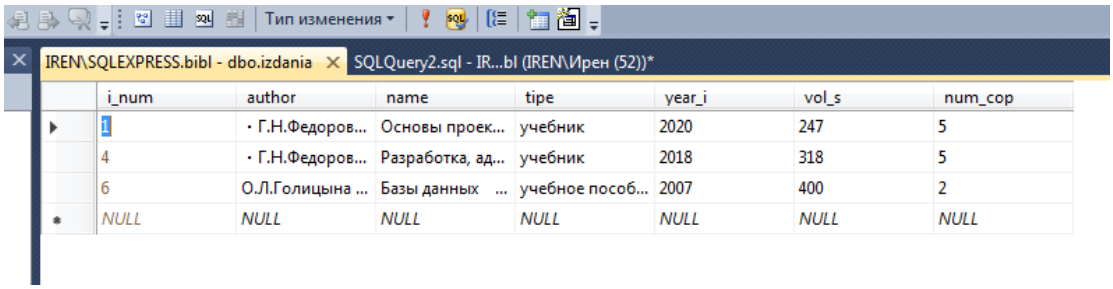


```
45
46  CREATE TRIGGER korr_iss ON dbo.issuing_books
47  AFTER INSERT
48  AS
49  DECLARE @i_new INT;
50  SELECT @i_new = i_num FROM inserted;
51  DECLARE @num_new INT;
52  SELECT @num_new = num FROM inserted;
53  DECLARE @num_izd INT;
54  SELECT @num_izd = num_cop FROM dbo.izdania
55  WHERE dbo.izdania.i_num = @i_new;
56  IF @num_new>@num_izd
57  BEGIN
58  SET NOCOUNT ON;
59  ROLLBACK TRAN;
60  END ELSE UPDATE dbo.izdania SET num_cop = @num_izd - @num_new
61  WHERE dbo.izdania.i_num = @i_new;
```

Составим запрос на добавление данных в таблицу «Выдача» и посмотрим на данные таблицы «Издания».

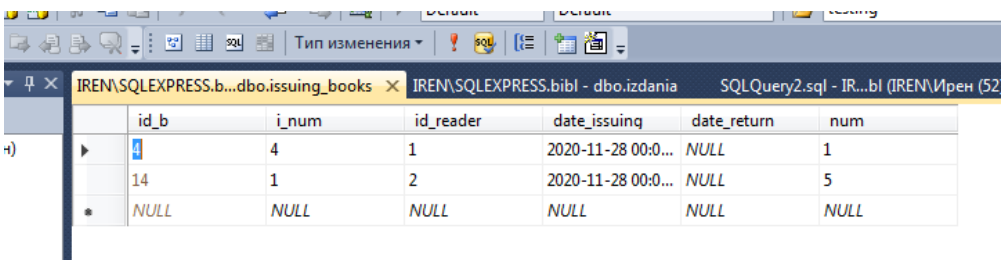


```
62
63  INSERT INTO dbo.issuing_books VALUES (1,2,'28.11.2020',NULL,5);
```



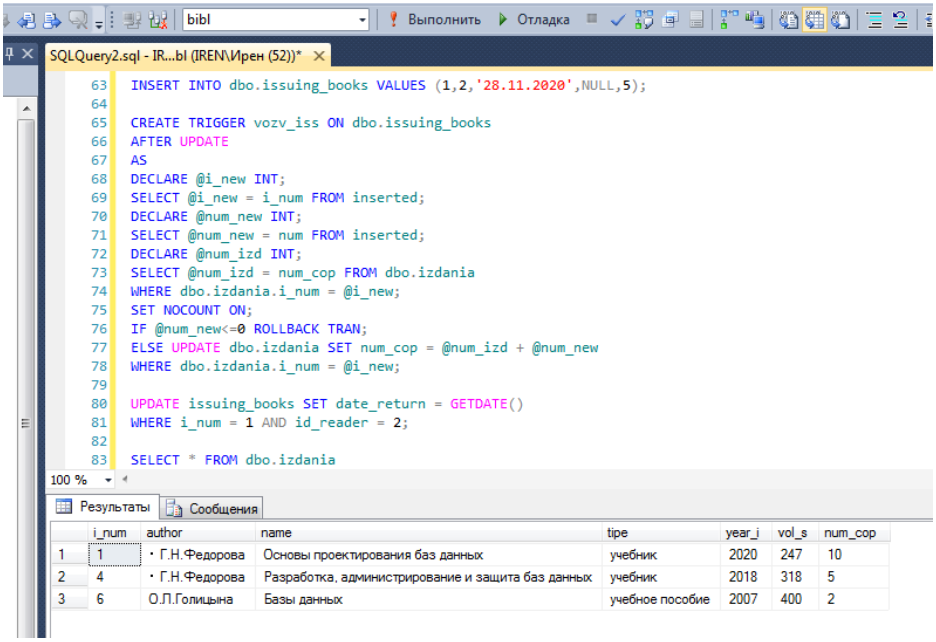
i_num	author	name	tipe	year_i	vol_s	num_cop
1	· Г.Н.Федоров...	Основы проек...	учебник	2020	247	5
4	· Г.Н.Федоров...	Разработка, ад...	учебник	2018	318	5
6	О.Л.Голицына ...	Базы данных ...	учебное пособ...	2007	400	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

В таблице выдачи появилась новая запись. Как можно заметить, в поле даты возврата значение NULL.

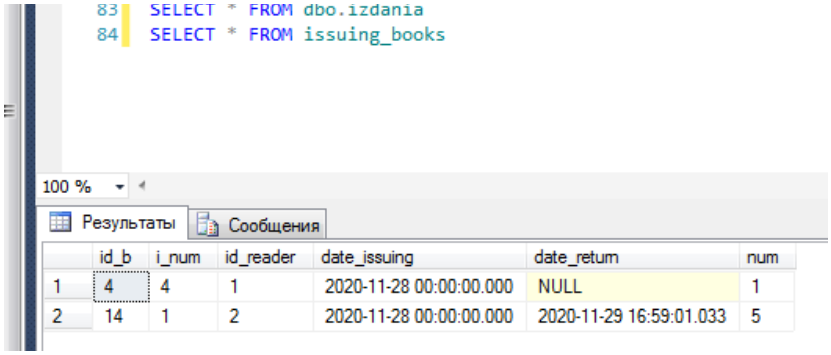


id_b	i_num	id_reader	date_issuing	date_return	num
4	4	1	2020-11-28 00:0...	NULL	1
14	1	2	2020-11-28 00:0...	NULL	5
NULL	NULL	NULL	NULL	NULL	NULL

Это значит, что читатель книги не вернул, при их возврате необходимо обновить таблицу «Выдача», при этом также обновить таблицу «Издания», где в поле наличия экземпляров добавить значения. Создадим триггер для обновления данных таблицы «Издания», запрос на обновление таблицы выдачи. В результате можно видеть, что количество книг в таблице изданий увеличилось.



А в таблице выдачи появилась дата возврата



3 Использование триггеров при работе с данными в хранимых процедурах

Если читатель вернул не все взятые книги, а только часть, можно оформить выдачу следующим образом: закрыть уже созданный запрос и сразу создать новый на еще не сданные книги. Оформим эти запросы в виде хранимой процедуры.

SQLQuery2.sql - IR...bl (IREN\Ирен (52))*

```
86  CREATE PROC c_vozv
87  @i_r INT,
88  @i_b INT,
89  @kol INT
90  AS
91  BEGIN
92    DECLARE @num_new INT;
93    SELECT @num_new = num FROM issuing_books
94    WHERE i_num = @i_b AND id_reader = @i_r;
95    UPDATE issuing_books SET date_return = GETDATE()
96    WHERE i_num = @i_b AND id_reader = @i_r;
97    INSERT INTO dbo.issuing_books VALUES (@i_b,@i_r,GETDATE(),NULL,@num_new-@kol)
98  END;
```

Оформим выдачу для читателя с кодом 5 6 учебников:

```
100  INSERT INTO dbo.issuing_books VALUES (1,5,GETDATE(),NULL,6);
101  SELECT * FROM dbo.izdania;
102  SELECT * FROM issuing_books;
```

100 %

Результаты | Сообщения

	i_num	author	name	tipe	year_i	vol_s	num_cop
1	1	· Г.Н.Федорова	Основы проектирования баз данных	учебник	2020	247	4
2	4	· Г.Н.Федорова	Разработка, администрирование и защита баз данных	учебник	2018	318	5
3	6	О.Л.Голицына	Базы данных	учебное пособие	2007	400	2

	id_b	i_num	id_reader	date_issuing	date_return	num
1	4	4	1	2020-11-28 00:00:00.000	NULL	1
2	14	1	2	2020-11-28 00:00:00.000	2020-11-29 16:59:01.033	5
3	15	1	5	2020-11-29 18:00:12.040	NULL	6

После запуска процедуры таблицы имеют вид:

```
99
100  INSERT INTO dbo.issuing_books VALUES (1,5,GETDATE(),NULL,6);
101  SELECT * FROM dbo.izdania;
102  SELECT * FROM issuing_books;
103
```

100 %

Результаты | Сообщения

	i_num	author	name	tipe	year_i	vol_s	num_cop
1	1	· Г.Н.Федорова	Основы проектирования баз данных	учебник	2020	247	8
2	4	· Г.Н.Федорова	Разработка, администрирование и защита баз данных	учебник	2018	318	5
3	6	О.Л.Голицына	Базы данных	учебное пособие	2007	400	2

	id_b	i_num	id_reader	date_issuing	date_return	num
1	4	4	1	2020-11-28 00:00:00.000	NULL	1
2	14	1	2	2020-11-28 00:00:00.000	2020-11-29 16:59:01.033	5
3	15	1	5	2020-11-29 18:00:12.040	2020-11-29 18:06:20.667	6
4	16	1	5	2020-11-29 18:06:20.750	NULL	2

Можно заметить, что хранимая процедура обновила данные в таблице «Выдача», а триггеры – записи в таблице «Издания».

Из вышеизложенного следует, что использование триггеров позволяет …

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 13.

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018
– https://metanit.com/sql/sqlserver/12.1.php
– https://info-comp.ru/obucheniest/361-trigger-in-transact-sql.html