Карпухина Анна

Группа М60-307Б-18

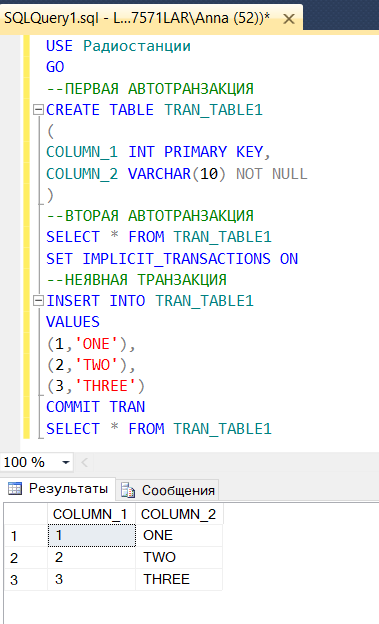
**Оглавление**

[**Задание 1** 2](#_Toc72493572)

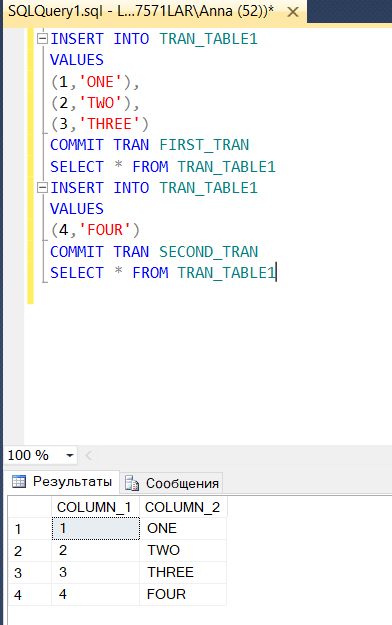
[**Задание 2** 4](#_Toc72493573)

# **Задание 1**

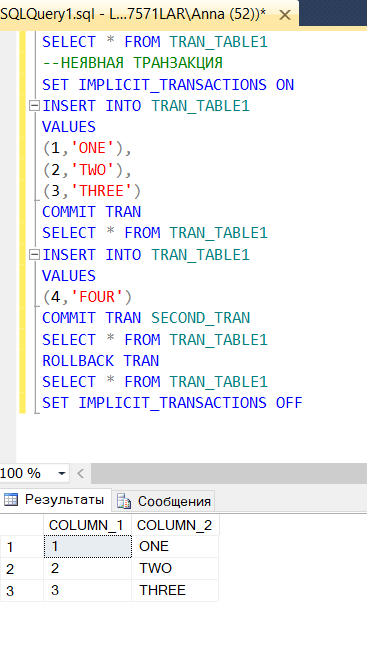
*Неявная транзакция с фиксированием*



*Добавление данных и фиксация:*



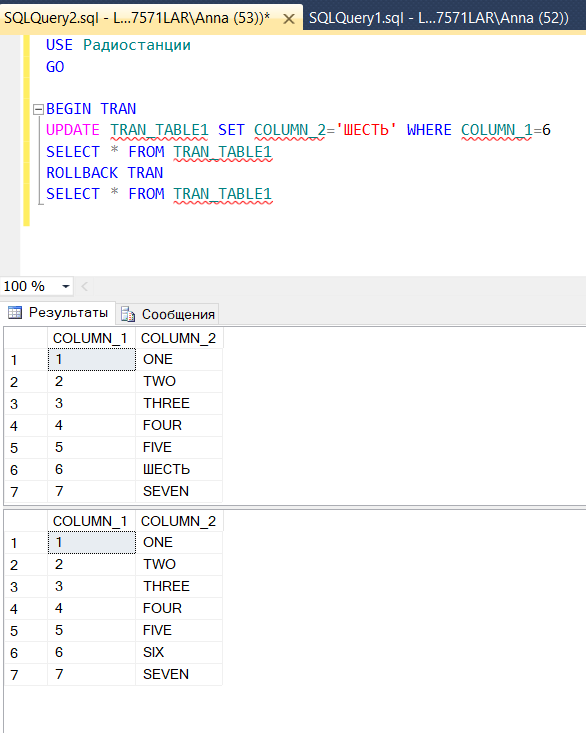
*Произведение отката:*



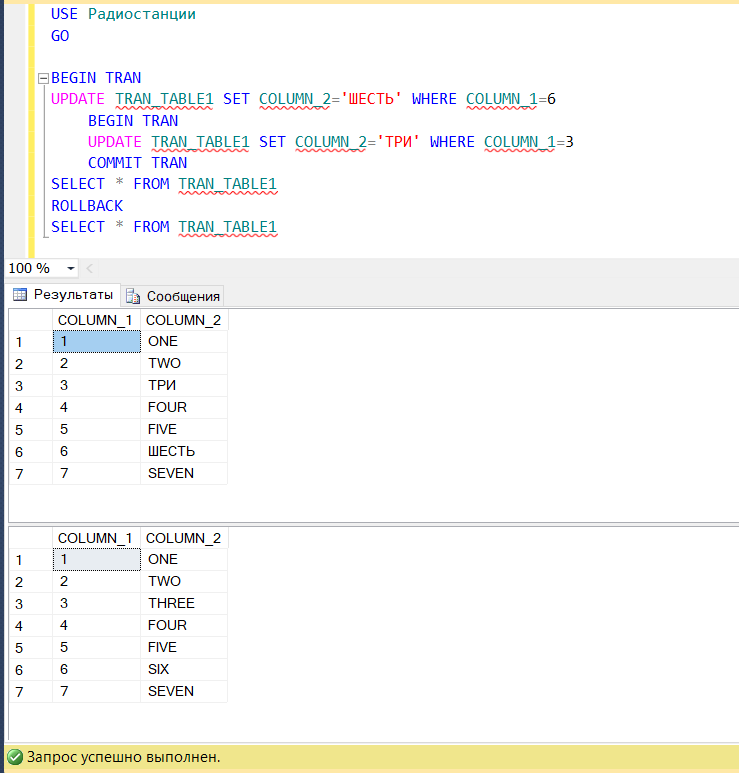
# **Задание 2**

*Явная транзакция с фиксированием*

*(с выборкой до и после)*



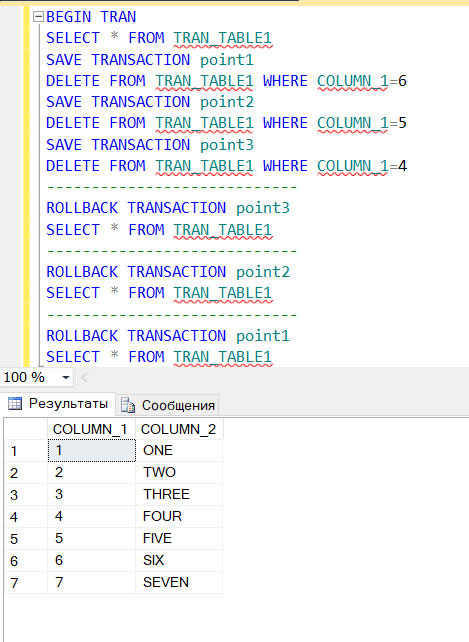
*Откат с фиксацией:*



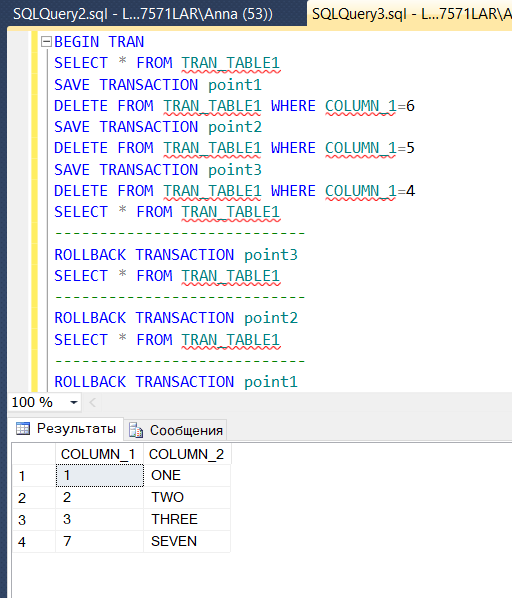
*Откатаны были оба изменения UPDATE так как ROLLBACK общий.*

# **Задание 3**

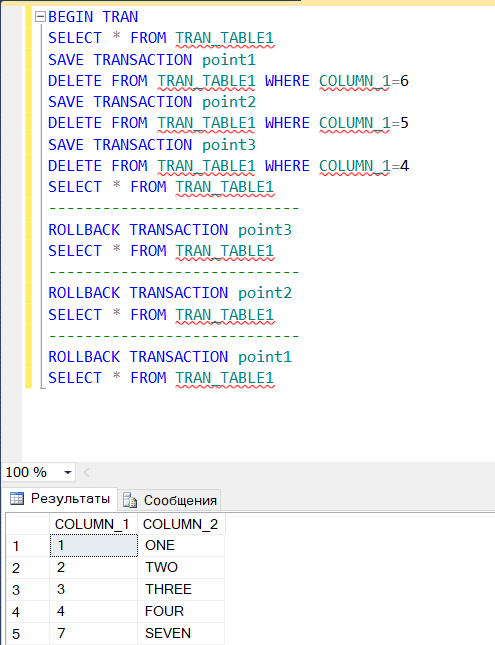
Результат работы первой выборки (то есть сама таблица)



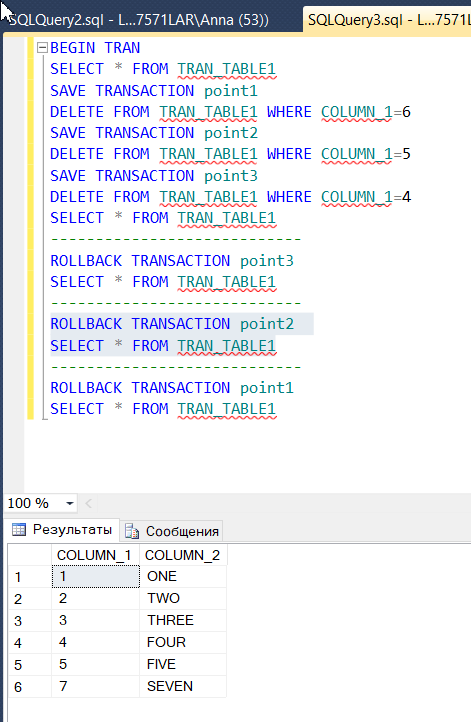
*Результат после второй выборки:*



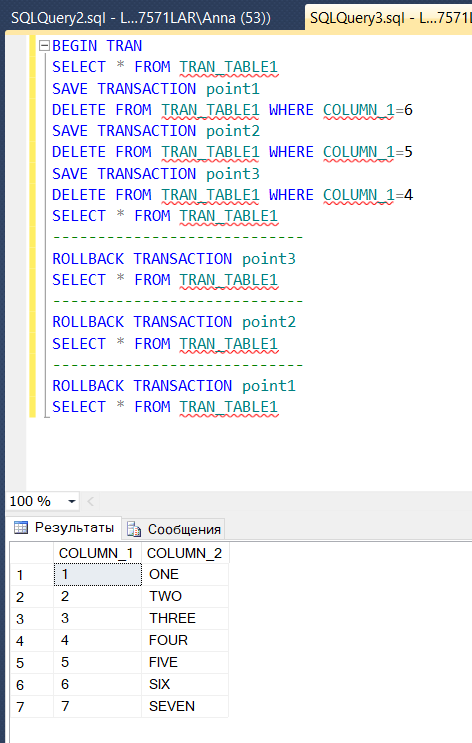
*Результат после возврата к третьей точке сохранения:*



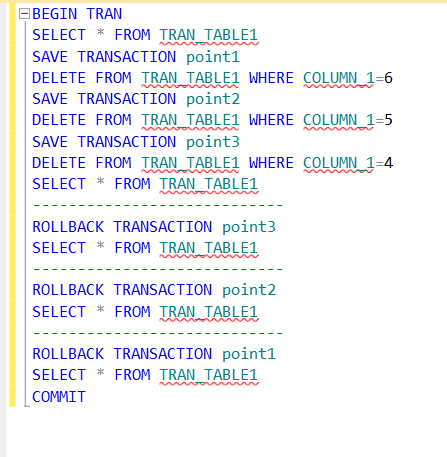
*Результат после возврата ко второй точке сохранения:*



*Результат возврата к первой точке сохранения:*

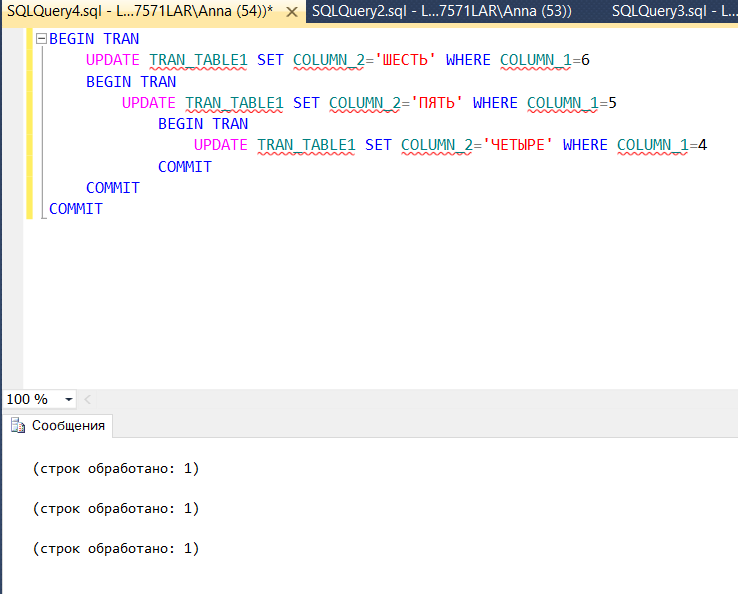


*Зафиксируем транзакции:*



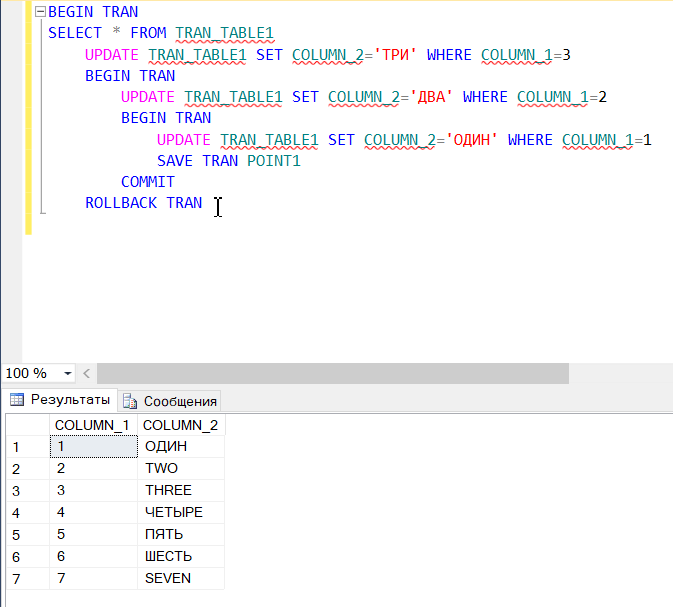
# **Задание 4**

*Создадим транзакцию с тремя уровнями вложенности и зафиксирует транзакции на каждом уровне.*

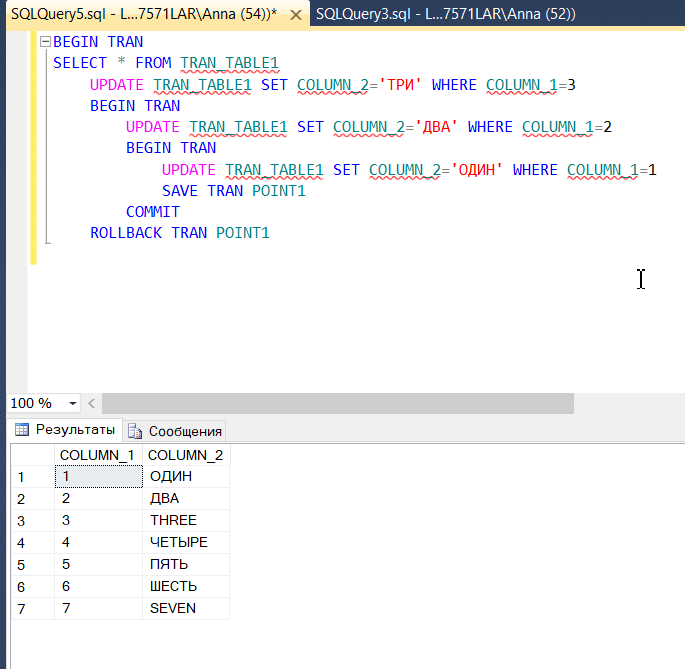


# **Задание 5**

*Изменения после третьей вложенной транзакции*

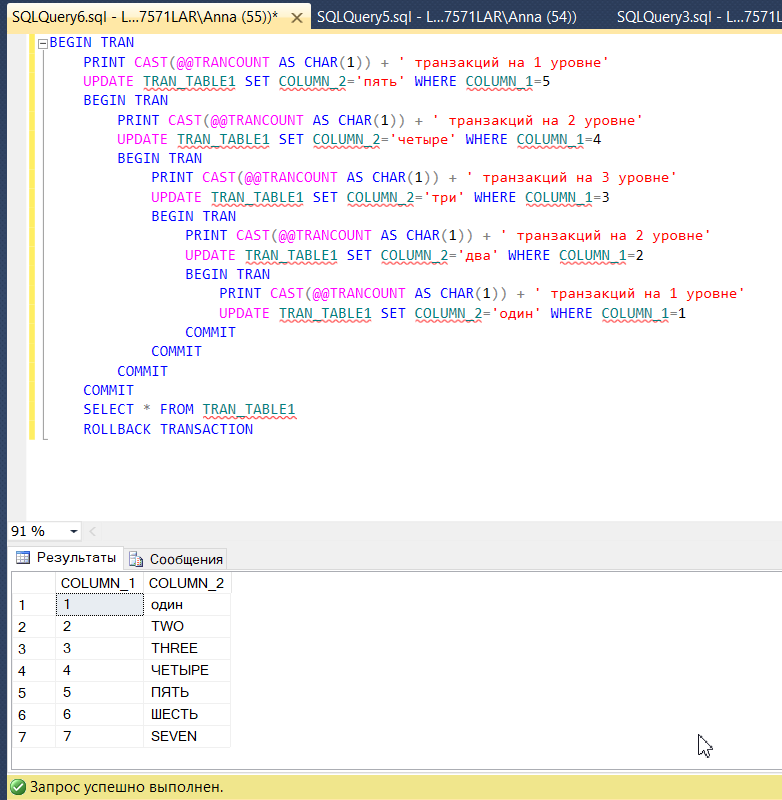


*Выполнение второго уровня транзакций:*

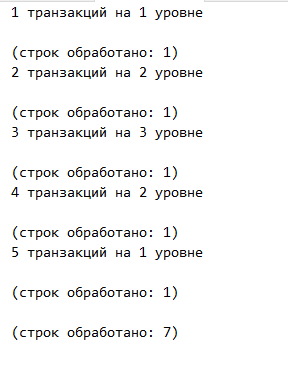


# **Задание 6**

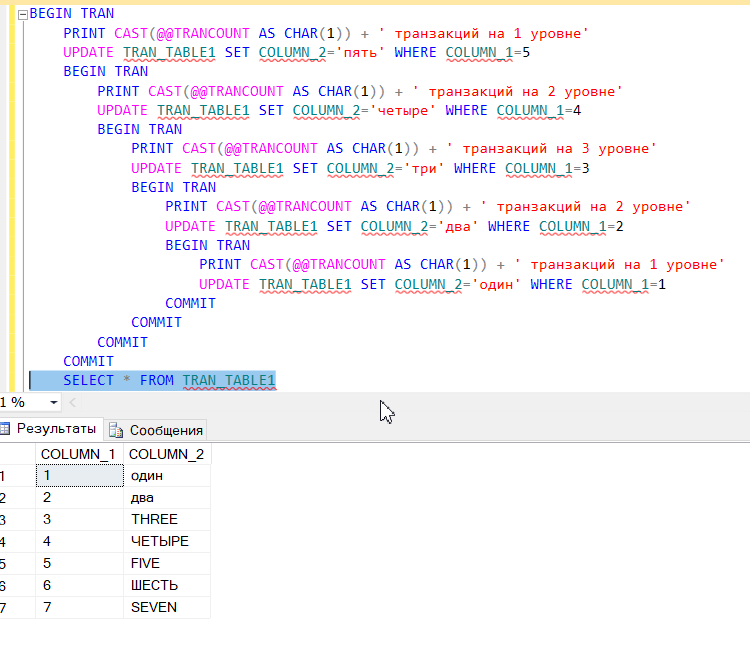
*Реализация внутренней 5 транзакции:*



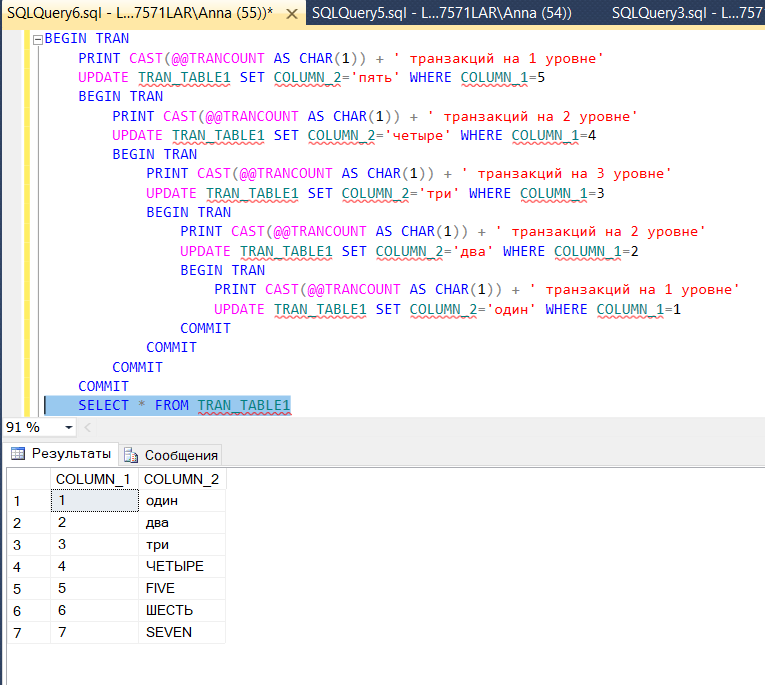
*Количество активных транзакций:*



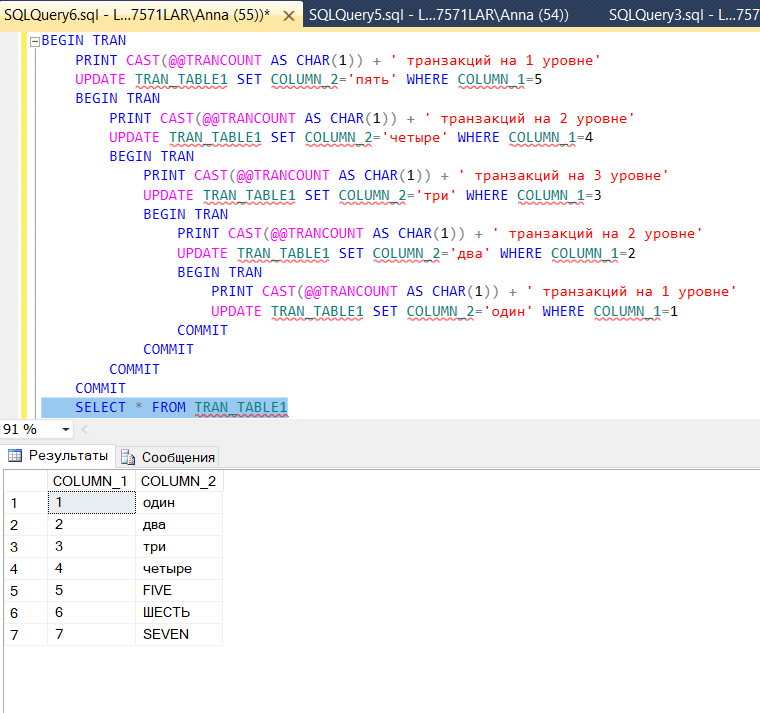
*Реализация внутренней 4 транзакции:*



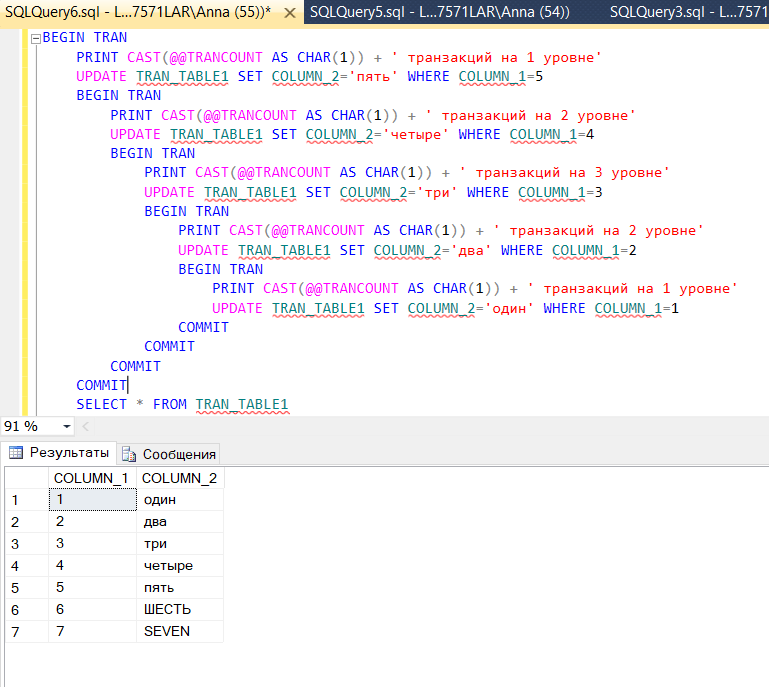
*Реализация внутренней 3 транзакции:*



*Реализация внутренней 2 транзакции:*



*Реализация внешней первой транзакции:*

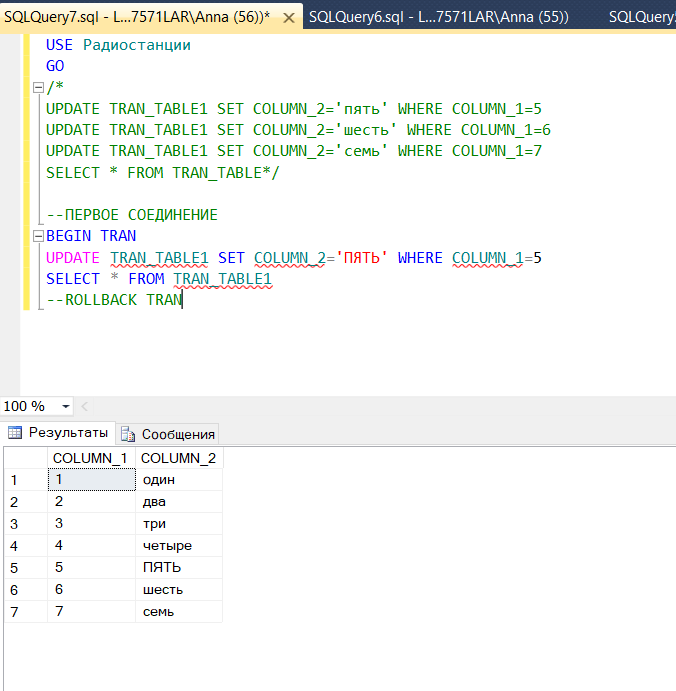


# **Задание 7**

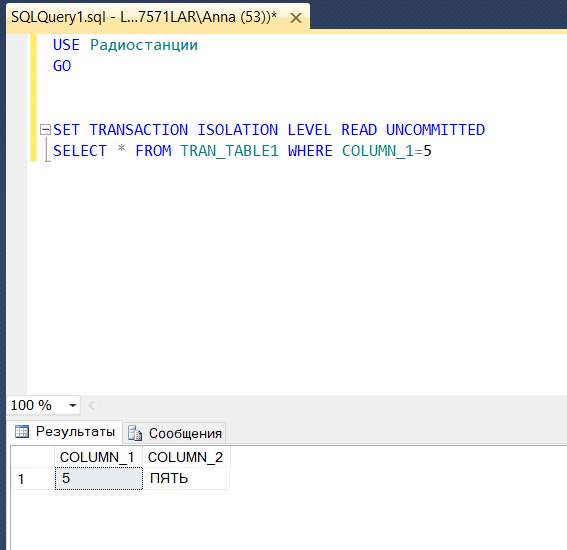
Низший доступный уровень изоляции - READ UNCOMMITTED, когда читающий процесс не запрашивает совместную блокировку и возможно незафиксированное чтение (грязное чтение).

Процесс, не запрашивающий совместную блокировку, никогда не будет конфликтовать с пишущим процессом, удерживающим монопольную блокировку. Это означает, что читающий процесс может прочесть незафиксированные изменения.

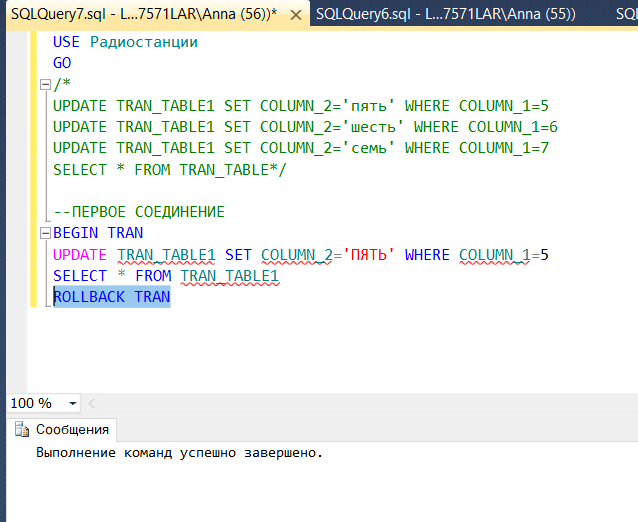
*Итак, организуем соединение 1:*



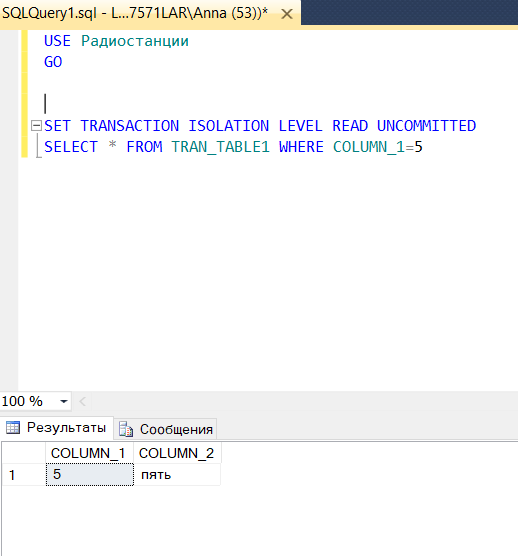
*Выполним соединение 2:*



*Теперь завершим транзакцию в соединении 1, используя ROLLBACK:*



*Выполним «грязное чтение» из второго соединения:*



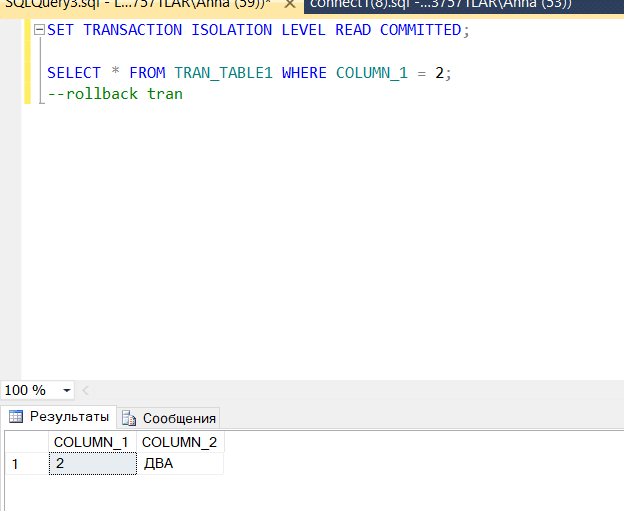
Стоит отметить, что самую высокую скорость выполнения и самую низкую согласованность имеет уровень read uncommitted.

Данный уровень можно использовать, например, для примерных расчетов чего-либо. Результат *COUNT(\*)* или *MAX(\*)* можно использовать в каких-нибудь нестрогих отчетах.  
Другой пример это режим отладки. Когда во время транзакции, вы хотите видеть, что происходит с базой.

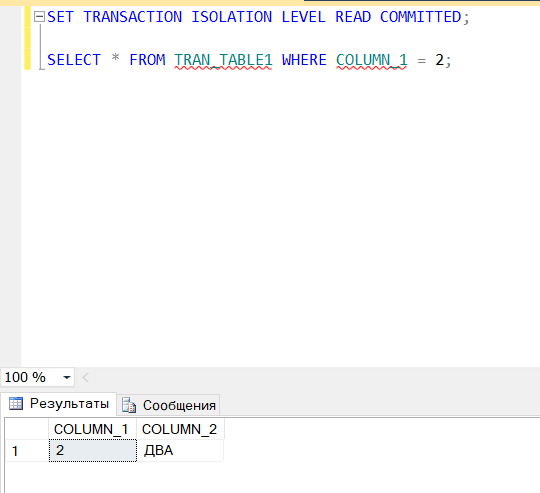
# **Задание 8**

Для этого уровня параллельно исполняющиеся транзакции видят только зафиксированные изменения из других транзакций. Таким образом, данный уровень обеспечивает защиту от «грязного чтения». Читаются только зафиксированные изменений.

*Обеспечим соединение 1:*



*Теперь подключим соединение 2, которое должно обеспечить блокировку на данном этапе, но при откате ROLLBACK запрос выполняется:*



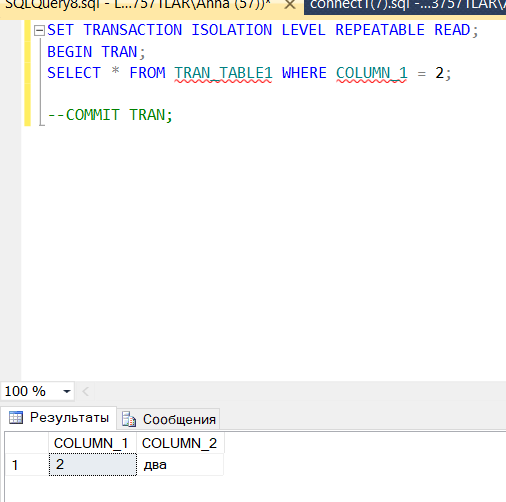
На этом уровне изоляции также гарантируется отсутствие потерянных обновлений, но возможно неповторяющееся чтение данных.

Что касается продолжительности блокировок, на уровне изоляции READ   COMMITTED читающий процесс только удерживает совместную блокировку на время работы с ресурсом. Блокировка не сохраняется до конца транзакции, в действительности она даже не сохраняется до завершения инструкции.

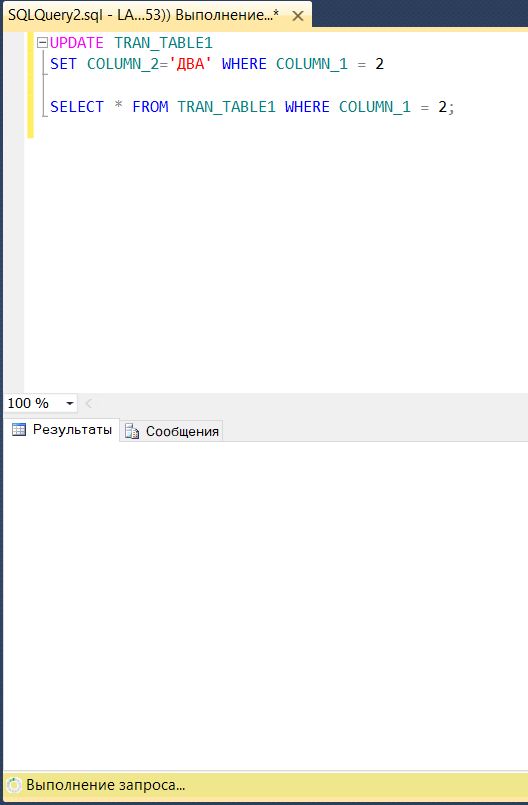
# **Задание 9**

Уровень, позволяющий предотвратить феномен неповторяющегося чтения. Т.е. мы не видим в исполняющейся транзакции измененные и удаленные записи другой транзакцией. Но все еще видим вставленные записи из другой транзакции. Чтение фантомов никуда не уходит.

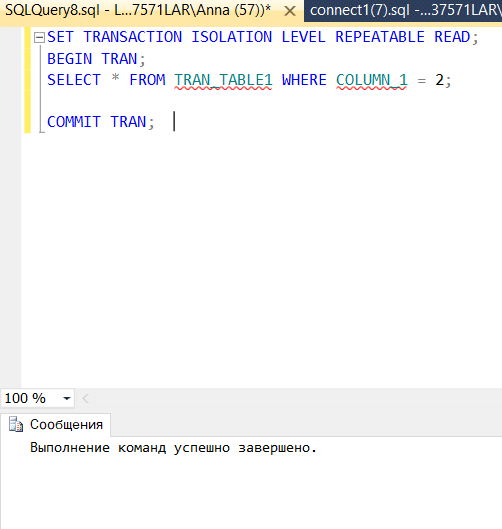
*Соединение 1:*



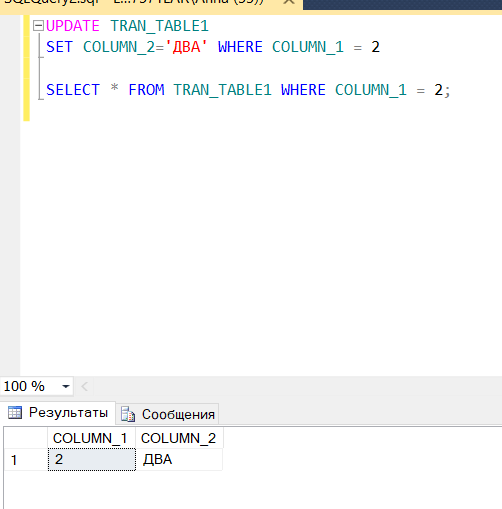
*Результат не выполняется так как срабатывает блокировка:*



*Блокировка снимается после использования COMMIT*



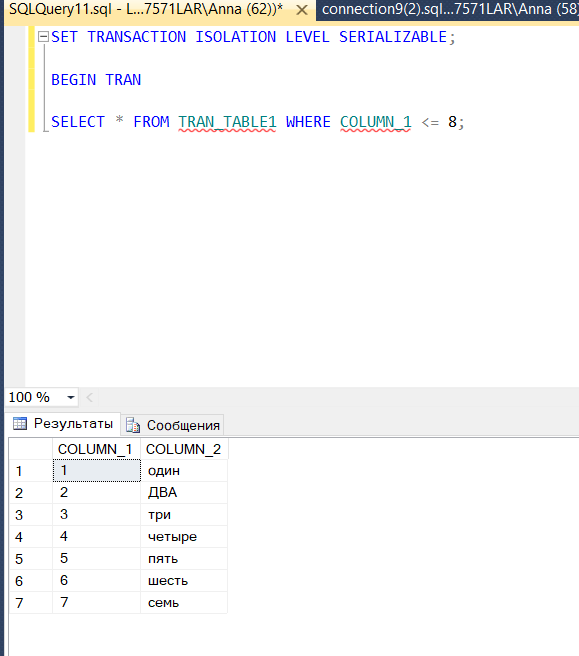
*Теперь соединение 2 разблокировано и выдает результат:*



# **Задание 10**

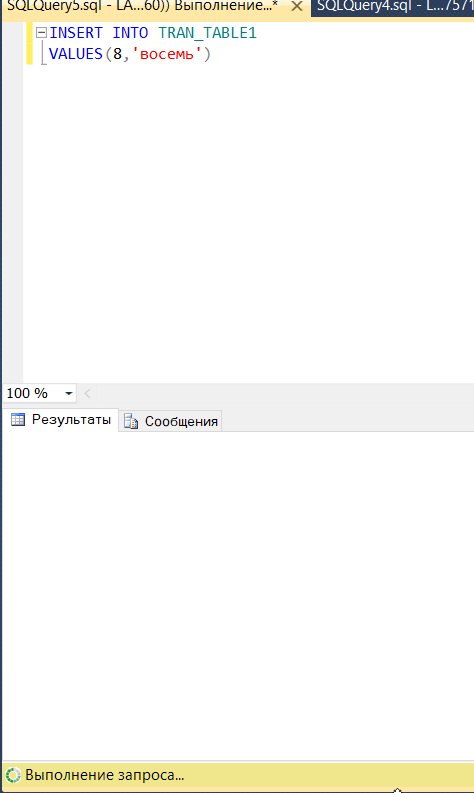
Уровень, при котором транзакции ведут себя как будто ничего более не существует, никакого влияния друг на друга нет. В классическом представлении этот уровень избавляет от эффекта чтения фантомов.

*Соединение1:*



*Соединение 2:*

*Добавим строку, соответствующую условию выборки выше и попытаемся выполнить запрос:*



*Запрос завершился с блокировкой*

Уровень изоляции SERIALIZABLE ведет себя так же, как и REPEATABLE READ, а именно, для чтения он требует от читающего процесса получения совместной блокировки и сохранения ее до завершения транзакции. Но уровень изоляции SERIALIZABLE добавляет еще один аспект — логически этот уровень изоляции заставляет Читающий процесс блокировать целый диапазон ключей (строк), которые удовлетворяют условию фильтрации запроса читающего процесса.

Это означает, что процесс блокирует не только существующие строки, удовлетворяющие условию фильтра запроса, но и будущие строки. Или более точно, он блокирует попытки других транзакций добавить строки, удовлетворяющие условию фильтра запроса читающей стороны.

# **Задание 11**

Сравнение REPEATABLE READ и SERIALIZABLE

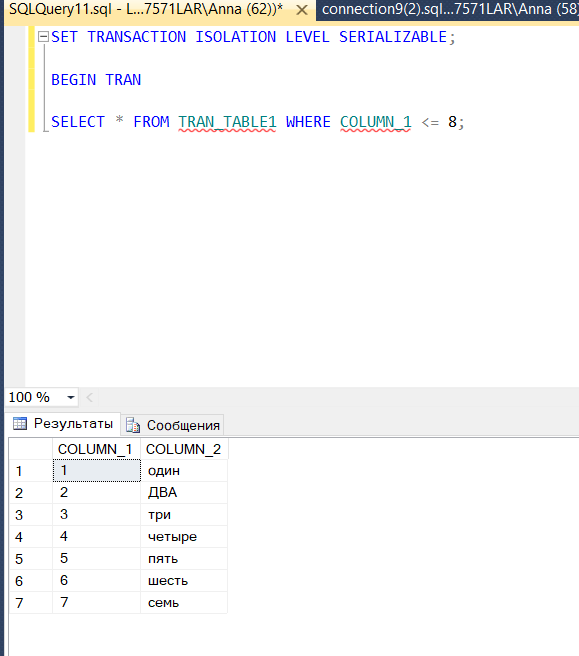
REPEATABLE READ – повторяющееся чтение. Повторное чтение строки возвратит первоначально считанные данные, несмотря на любые обновления, произведенные другими пользователями до завершения транзакции. Тем не менее на этом уровне изоляции возможно возникновение фантомов.

SERIALIZABLE – сериализуемость. Чтение запрещено до завершения транзакции. Это максимальный уровень изоляции, который обеспечивает полную изоляцию транзакций друг от друга.

*Попробуем добавить данные при этих двух уровнях:*

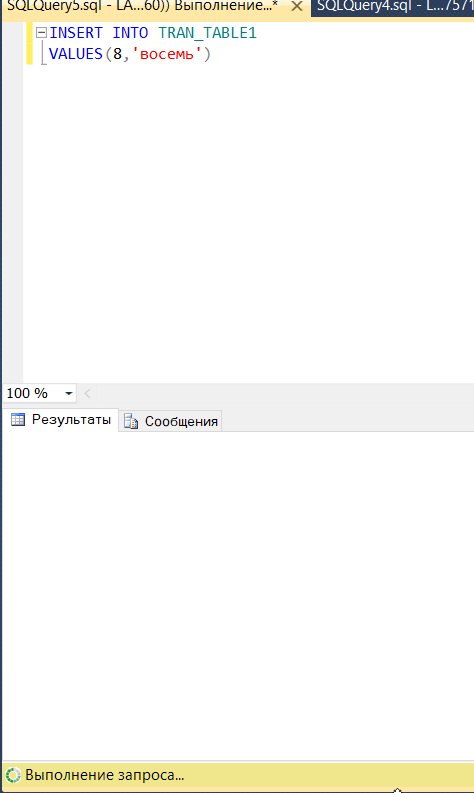
*Добавление при уровне SERIALIZABLE*

*Соединение1:*



*Соединение 2:*

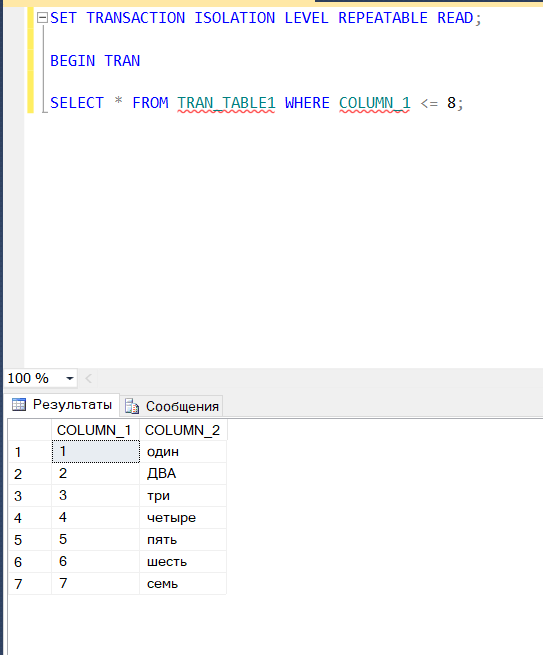
*Добавим строку, соответствующую условию выборки выше и попытаемся выполнить запрос:*



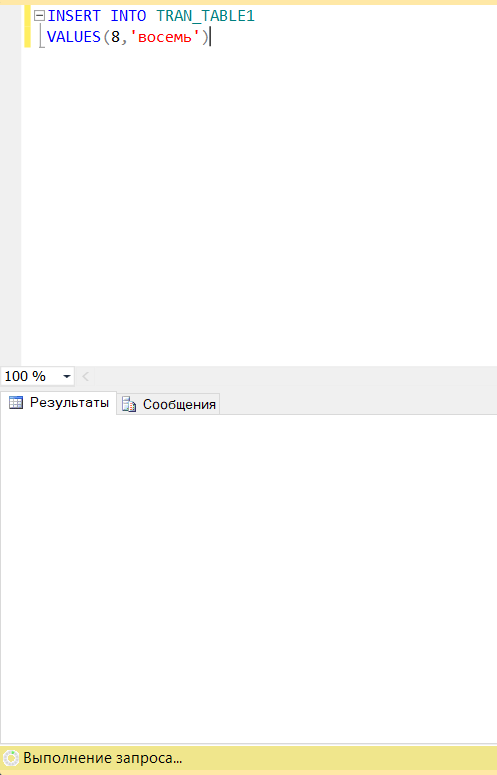
*Запрос завершился с блокировкой.*

*Добавление при уровне REPEATABLE READ*:

*Выводим запрос:*



*Попытка добавления строки в таблицу также привела к блокировке:*



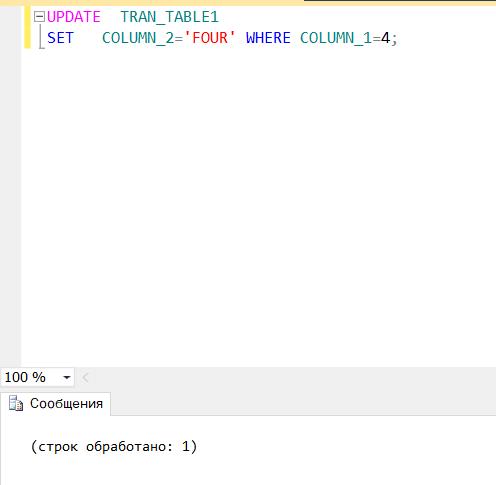
# **Задание 12**

ALLOW\_SNAPSHOT\_ISOLATION (использование мгновенного снимка) 

Моментальный снимок базы данных является статичным, доступным только для чтения представлением базы данных SQL Server. Моментальный снимок базы данных согласуется на уровне транзакций с базой данных-источником в момент создания моментального снимка.

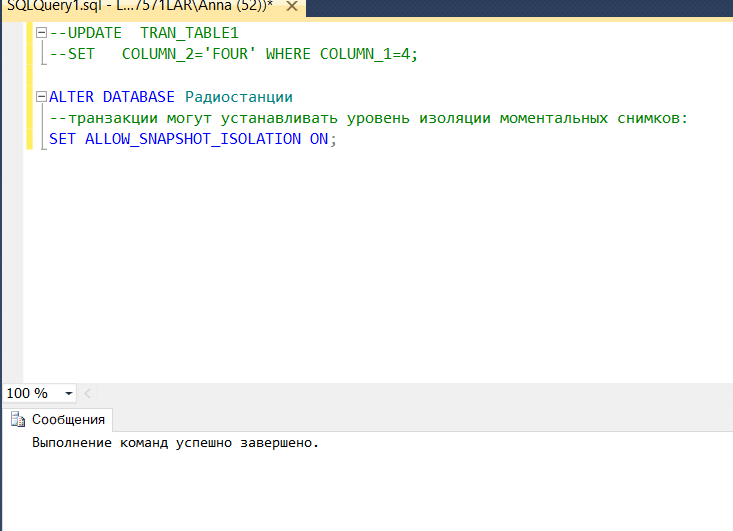
*При Соединении 1  в таблице TRAN\_TABLE1 для строки с COLUMN\_1 = 4 значение поля COLUMN\_2=’FOUR’.*

*Определяем обновление:*

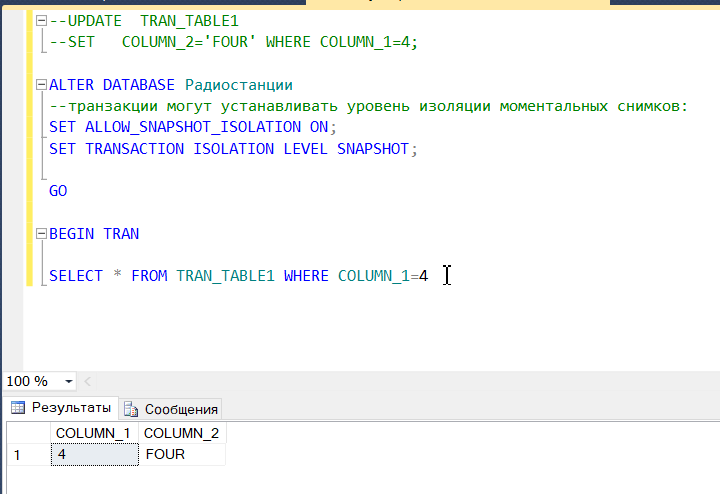


*Устанавливаем Соединение 1:*

*Определяем использование мгновенного снимка:*

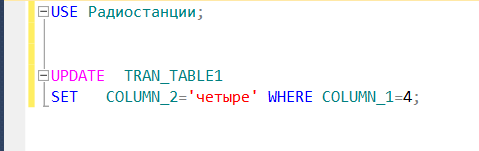


*Зададим уровень изоляции, выполним выборку:*

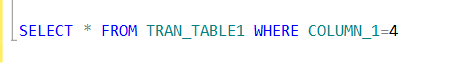


*Установим соединение 2:*

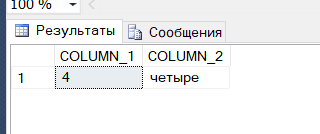
*При попытке изменения данных в первоначальное состояние, обновление не блокируется, так как использование данного уровня изоляции позволяет не блокировать используемые записи.*



*Если попытаемся вывести значение:*



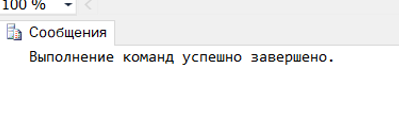
*То выведется неизменное значение:*



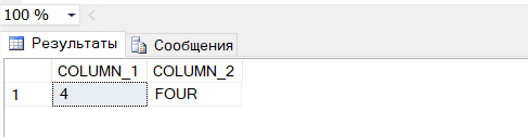
*Завершим транзакцию в соединении 2:*



*Транзакция успешно завершается:*



*При обращении к оператору SELECT в соединении 2 и выборке значения, происходит выборка следующего значения:*



*Поскольку данные были изменены вне транзакции с уровнем изоляции мгновенного (моментального) снимка, любые дальнейшие изменения данных в Соединении 1.*

*Вызывается сообщение, что транзакция не завершилась, генерируется ошибка 3960 и транзакция завершается.*

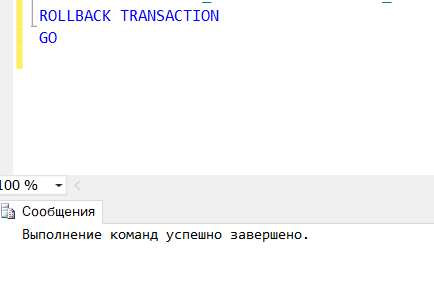
*Ошибка:*

*Сообщение 3960, уровень 16, состояние 2, строка 1*

*Транзакция в режиме изоляции моментального снимка прервана из-за конфликта обновлений. Невозможно использовать режим изоляции моментального снимка для прямого или косвенного доступа к таблице TRAN\_TABLE\_1 в базе данных Радиостанции для обновления, удаления или вставки строки, которая изменена или удалена другой транзакцией. Повторите транзакцию или измените уровень изоляции для инструкции обновления или удаления.*

*Откатим транзакция через ROLLBACK в соединении 1 в случае невыполнения UPDATE.*

*Завершим транзакцию в Соединении 1 (откат транзакции в случае невыполнения операции UPDATE. При появлении ошибки транзакция автоматически закрывается).*



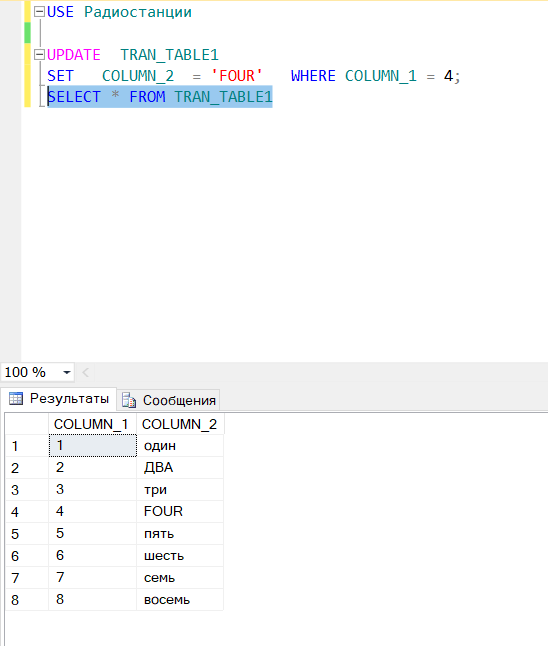
# **Задание 13**

READ\_COMMITTED\_SNAPSHOT (использование версии строк)

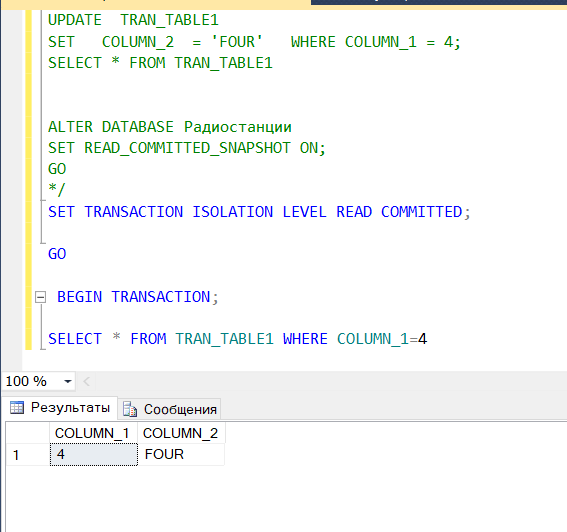
В отличие от транзакции моментального снимка, в этой транзакции:

1. считываются измененные данные после того, как другая транзакция фиксирует изменения;
2. могут обновляться данные, измененные другой транзакцией.

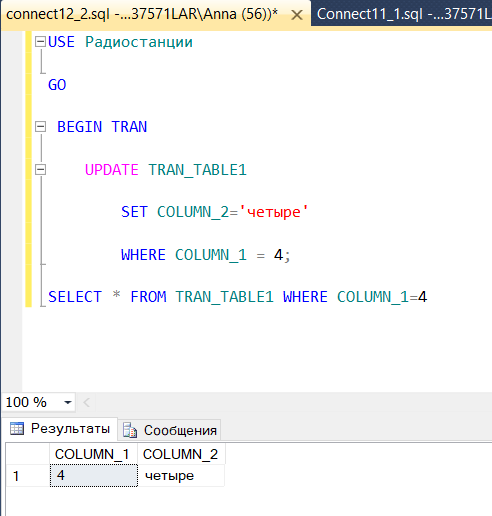
*Заранее изменим значение в базе:*



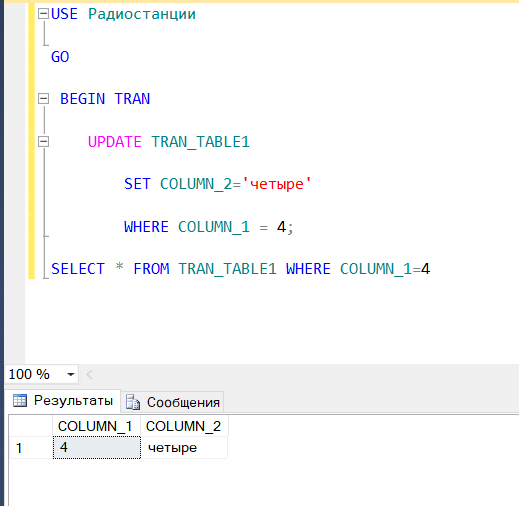
*Соединение 1:*



*Соединение 2:*



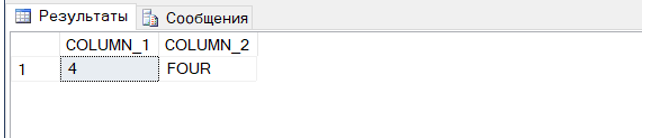
*Вывелось прежнее значение*



*После изменения данных в соединении 2 внутри транзакции выполним еще раз оператор выбора в Соединении 1:*



*Данные не изменились*

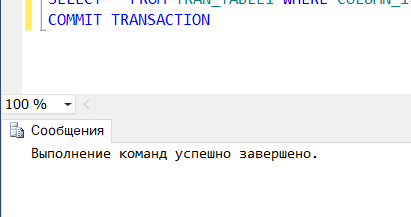


*В этой транзакции оператор выбора считывает данные из версионной строки (версия строк), в которой записано исходное значение. Транзакция в Соединении 2 еще не зафиксировала изменения (транзакция не закрыта).*

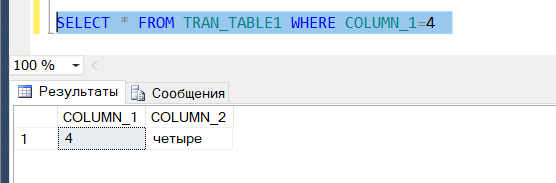
*В Соединении 2 фиксируем транзакцию.*



*Успешное фиксирование:*

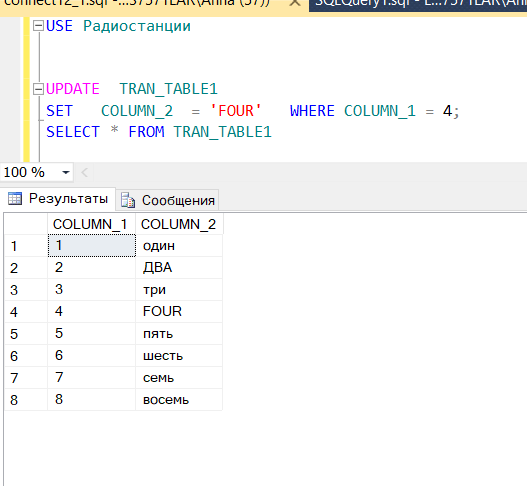


*В Соединении 1 выполним еще раз оператор:*



*Это отличается от рассмотренного выше случая с использованием мгновенного снимка (нельзя было прочитать измененные данные с прекращение работы и выводом ошибки). В данном случае при использовании версии строк считывается измененное значение.*

*Также в этом случае можно выполнить изменение данных в Соединении 1:*



*Ошибки не выдалось, но изменений не произошло.*

*Т.е. не отменяются изменения, которые были сделаны в Соединении 2.*

*Данные в Соединении 2 также не меняются.*

*Закройте транзакцию в Соединении 1 после выполнения примера*

*ROLLBACK TRANSACTION*

