

Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заведующий
методическим кабинетом
_____ Е.В. Фалей
«__» _____ 2017

Специальность: «Программное обеспечение информационных технологий»	Дисциплина: «Базы данных и системы управления базами данных»
Составлена на основании учебной программы, утвержденной директором Колледжа бизнеса и права 30.12.2016	

Лабораторная работа № 19
Инструкционно-технологическая карта

Тема: Работа с транзакциями в различных режимах.

Цель работы: Научиться работать в различных режимах с транзакциями (вложенными транзакциями): режим автофиксации, явный режим, неявный режим.

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить раздел лекционного курса по теме «Транзакции. Механизм транзакций».
2. Получить у преподавателя индивидуальное задание и выполнить лабораторную работу в соответствии с вариантом задания согласно описанной в разделе «Пример выполнения работы» методике настоящей инструкционно-технологической карты.
3. Ответить на контрольные вопросы.

1. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

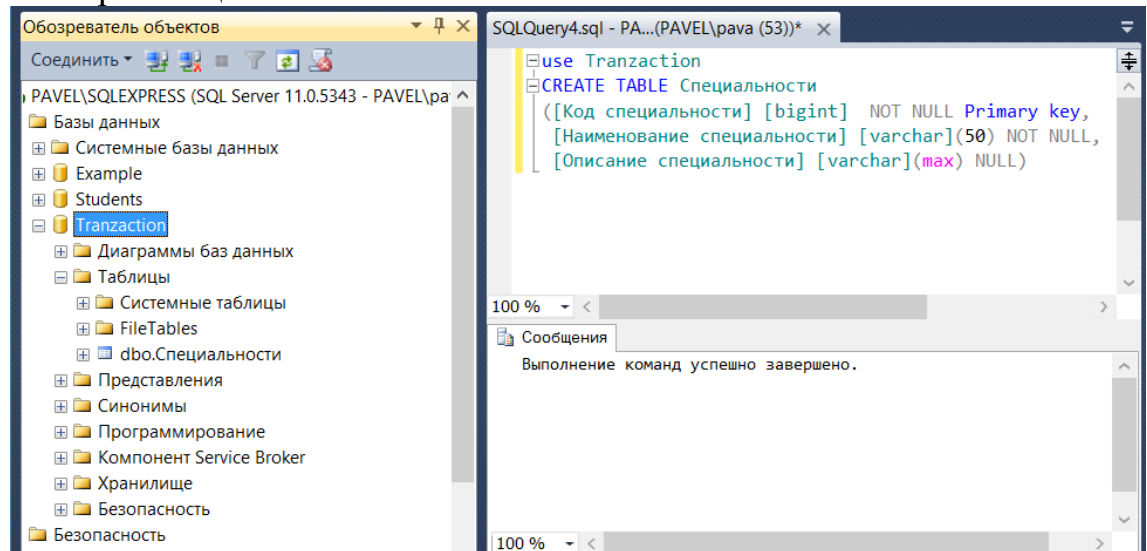
SQL Server предоставляет различные способы обработки транзакций, которые можно определить для каждого соединения с базой данных. Любое соединение может использовать тот режим, который необходим для выполнения специфических для этого соединения требований. Вот эти режимы:

- ✓ автофиксация транзакций;
- ✓ явные транзакции;
- ✓ неявные транзакции.

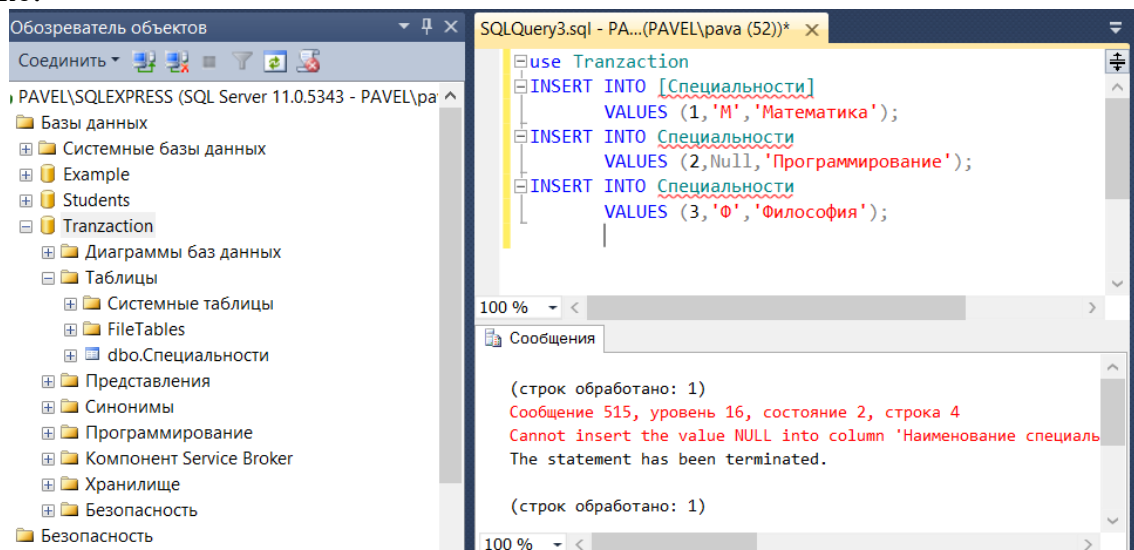
1. Работа в режиме автофиксации транзакций

SQL Server обрабатывает все изменения как транзакции. Никакое изменение данных не может произойти иначе, как в процессе транзакции. Следовательно, SQL Server приходится самому определять транзакцию, если она не определена разработчиком. Транзакции, определяемые SQL Server, называются также транзакцией с автофиксацией. Режим автофиксации используется SQL Server по умолчанию.

Для изучения работы с транзакциями в режиме автофиксации необходимо запустить SQL Server Management Studio и в любой БД, например, Transaction на сервере создать таблицу, как показано на рисунке ниже, которую необходимо использовать в следующем опыте для изучения поведения транзакций:



Теперь необходимо вставить в таблицу Специальности три новых строки. Для этого необходимо ввести следующие инструкции в окне запроса и выполнить все три инструкции вместе, как показано на рисунке ниже.

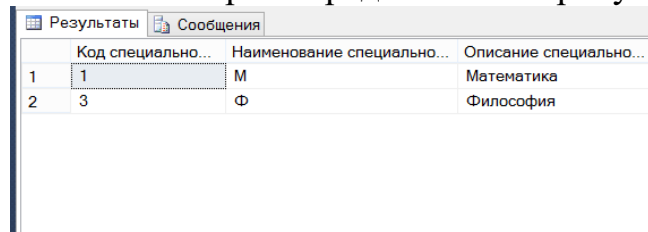


В результате получено сообщение, которое информирует о том, что SQL Server не разрешает вставку значения NULL в столбец Наименование специальности, потому что для этого столбца задано условие NOT NULL.

Чтобы проверить, были ли записи успешно вставлены в таблицу, необходимо ввести и выполнить следующую инструкцию SELECT, которая представлена ниже.

```
select [Код специальности], [Наименование специальности],  
[Описание специальности]  
from Специальности
```

Результат выполнения запроса представлен на рисунке ниже.



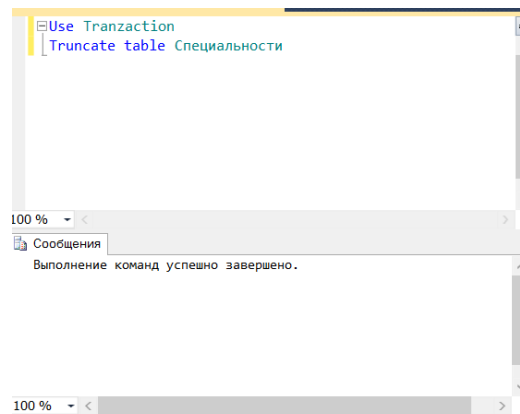
	Код специальн...	Наименование специальн...	Описание специальн...
1	1	М	Математика
2	3	Ф	Философия

Как видно, вставка второй строки не выполнена, но первая и третья строки успешно вставлены. Когда SQL Server использует транзакции с автофиксацией, каждая инструкция рассматривается как транзакция. Если одна инструкция генерирует ошибку, соответствующая ей транзакция автоматически подвергается откату. Если инструкция успешно и без ошибок выполняется, то транзакция автоматически фиксируется. Следовательно, инструкции 1 и 3 были зафиксированы, а инструкция 2, вызвавшая ошибку, была отменена. Важно, что такое поведение имеет место даже в том случае, если три инструкции передаются вместе в виде пакета. Пакетное выполнение не определяет, следует ли обрабатывать все инструкции в пакете как единую транзакцию.

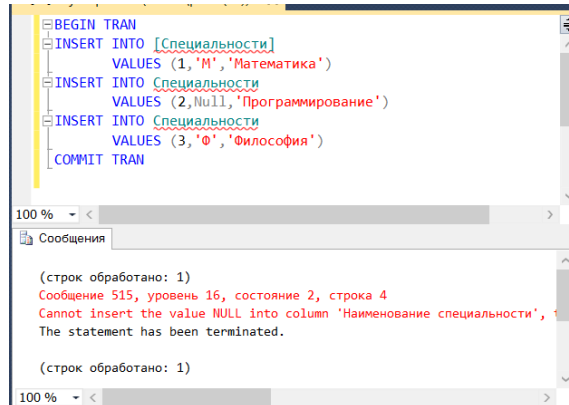
2. Работа в режиме явной транзакции

Для явной транзакции разработчик определяет начало транзакции и момент, в который она должна быть зафиксирована или подвергнута откату. Это достигается при помощи инструкций T-SQL BEGIN TRANSACTION, COMMIT TRANSACTION и ROLLBACK TRANSACTION. Явные транзакции независимы от пакета. Явная транзакция может объединять несколько пакетов; в одном пакете может быть задано несколько явных транзакций.

Для изучения работы с транзакциями в явном режиме необходимо сначала произвести полное удаление строк таблицы Специальности с помощью кода, представленного ниже



Далее необходимо вставить те же три записи в таблицу Специальности. На этот раз группируя инструкции в явную транзакцию, поскольку необходимо, чтобы в таблицу были вставлены все записи, или не было вставлено ни одной из них. В окне запроса необходимо ввести следующие инструкции и выполнить все эти инструкции как одну, как показано на рисунке ниже.



Как видно, получено такое же сообщение, как и раньше, в котором сообщается, что SQL Server не разрешает вставку значения NULL в столбец Наименование специальности, потому что для этого столбца задано условие NOT NULL.

Чтобы проверить, были ли записи успешно вставлены в таблицу, необходимо ввести и выполнить следующую инструкцию SELECT, которая представлена ниже.

```

select [Код специальности], [Наименование специальности],
[Описание специальности]
from Специальности
  
```

Результат выполнения запроса представлен на рисунке ниже.

Результаты			
	Код специальн...	Наименование специальн...	Описание специальн...
1	1	М	Математика
2	3	Ф	Философия

Видно, что результат тот же, что и в режиме автофиксации. Две из трех записей вставлены в таблицу, а одна, нарушающая ограничение NULL, не вставлена. Как отмечалось ранее, обязанностью разработчика является не только определение длины транзакции, но и то, должен ли выполняться откат. Поэтому в транзакцию необходимо добавить обработчик ошибок. Без обработчика ошибок SQL Server после ошибки просто обработает следующую инструкцию, потому что пакет не отменяется. В предыдущем пакете SQL Server просто обрабатывает каждую инструкцию INSERT и после этого обрабатывает инструкцию COMMIT TRAN. Следовательно, получился тот же результат, что и в режиме автофиксации. Чтобы добавить обработчик ошибок, можно использовать TRY и CATCH. Снова необходимо выполнить усечение таблицы, а затем запустить транзакцию с обработчиком ошибок, как показано ниже на рисунке.

```
--Выполнение удаления строк таблицы
Truncate table Специальности
--Транзакция с обработчиком ошибок
BEGIN TRY
    BEGIN TRAN
    INSERT INTO [Специальности]
        VALUES (1,'М','Математика')
    INSERT INTO Специальности
        VALUES (2,Null,'Программирование')
    INSERT INTO Специальности
        VALUES (3,'Ф','Философия')
    COMMIT TRAN
END TRY
BEGIN CATCH
    ROLLBACK TRAN
END CATCH
```

В этом случае сообщение об ошибке не будет получено, поскольку ошибка была захвачена блоком CATCH.

Чтобы проверить, был ли выполнен откат транзакции, необходимо ввести и выполнить следующую инструкцию SELECT, которая представлена ниже.

```
select [Код специальности], [Наименование специальности],
[Описание специальности]
from Специальности
```

Эта инструкция не возвратила ни одной записи. Как видно, произошел откат всей транзакции. Когда во второй инструкции INSERT произошло нарушение, SQL Server перешел к блоку CATCH и выполнил откат транзакции.

Однако, этот код не возвращает каких-либо сообщений, которые информировали бы о том, что произошла ошибка. Это поведение управляется в блоке CATCH, в котором можно использовать особые функции для возвращения ошибок; можно также использовать функцию RAISERROR для задания пользовательского текста сообщения об ошибке. Изменение блока CATCH показано ниже.

```
INSERT INTO [Специальности]
VALUES (1,'М','Математика')
INSERT INTO Специальности
VALUES (2,Null,'Программирование')
INSERT INTO Специальности
VALUES (3,'Ф','Философия')
COMMIT TRAN
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() as ErrorState,
        ERROR_PROCEDURE() as ErrorProcedure,
        ERROR_LINE() as ErrorLine,
        ERROR_MESSAGE() as ErrorMessage;
    RAISERROR('Error in Transaction!',14,1)
    ROLLBACK TRAN
```

100 %

Результаты Сообщения

```
(строк обработано: 1)

(строк обработано: 0)

(строк обработано: 1)
Сообщение 50000, уровень 14, состояние 1, строка 21
Error in Transaction!
```

100 %

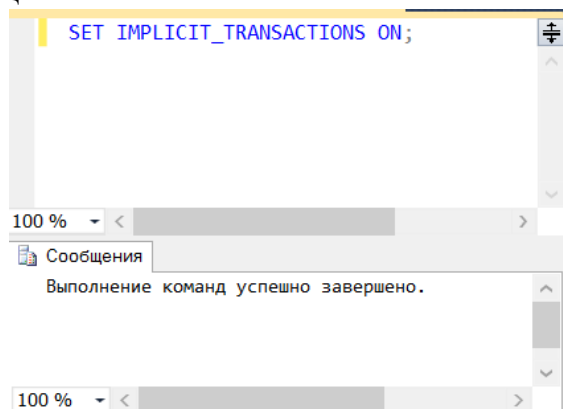
После выполнения транзакции будет возвращена запись со всей информацией, имеющей отношение к ошибке, и пользовательский текст сообщения, в котором говорится о том, что произошла ошибка. Кроме того, в инструкцию RAISERROR. можно также включить реальное сообщение об ошибке.

3. Неявные транзакции

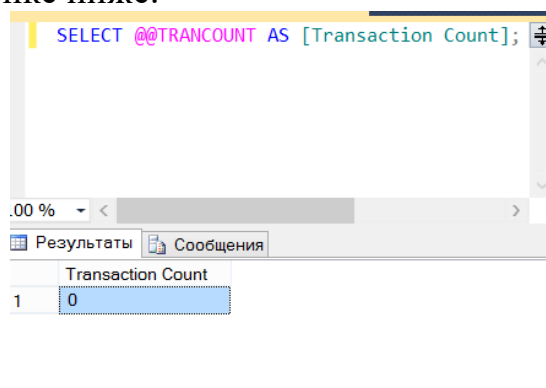
Третий режим получил название неявной транзакции, поскольку в этом режиме SQL Server запускает транзакцию, если не задано ни одной транзакции, но не выполняет инструкции COMMIT или ROLLBACK автоматически, как это происходит в режиме автофиксации. Транзакции нуждаются в явном завершении. Следующие инструкции запускают транзакции неявным образом при отсутствии явных транзакций:

ALTER TABLE	FETCH	REVOKE
CREATE TABLE	GRANT	SELECT
DELET	INSERT	TRUNCATE
		TABLE
DROP TABLE	OPEN	UPDATE

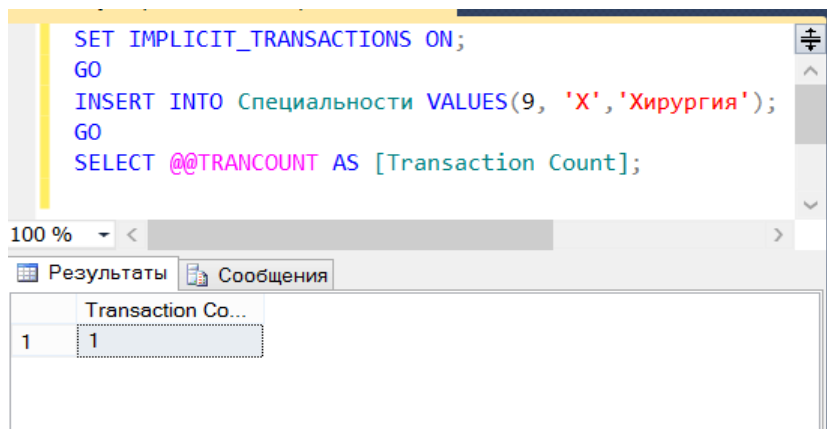
Для изучения работы с транзакциями в неявном режиме необходимо сначала задать для установленного соединения неявный режим, выполнив следующую инструкцию:



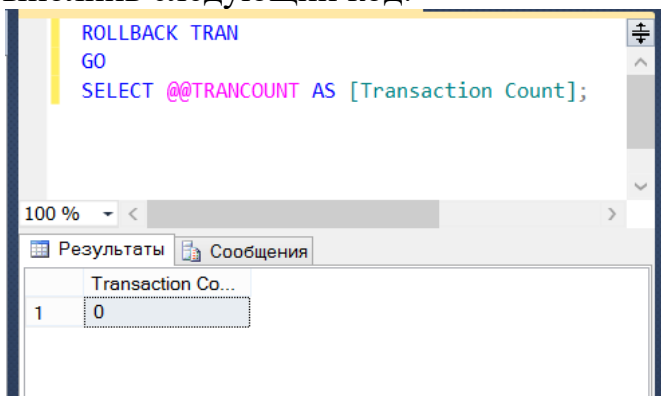
Чтобы проверить, открыта ли транзакция, используется функция @@TRANCOUNT. Результатом работы данной функции могут быть три значения: 1 – означает, что соединение имеет открытую транзакцию; 0 – означает, что в данный момент не открыто ни одной транзакции; число больше 1 – имеют место вложенные транзакции. Работа с данной функцией представлена на рисунке ниже:



Далее необходимо вставить в таблицу Специальности запись и снова проверить результат работы функции @@TRANCOUNT, как представлено ниже на рисунке:

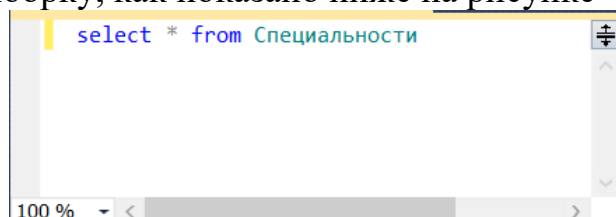


Значение функции @@TRANCOUNT равно 1. SQL Server запустил новую транзакцию. С помощью функции ROLLBACK TRAN необходимо выполнить откат транзакции и снова проверить значение функции @@TRANCOUNT, выполнив следующий код.



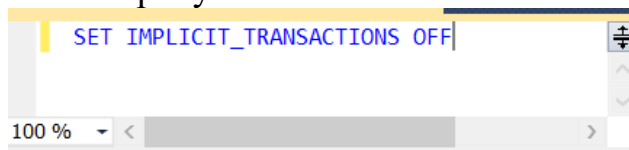
Видно, что значение @@TRANCOUNT равно 0 после выполнения инструкции ROLLBACK TRAN.

Если сделать выборку, как показано ниже на рисунке



то SQL Server не выдаст ни одной записи, так как неявная транзакция была запущена при помощи инструкции INSERT, а инструкция ROLLBACK TRAN отменила результаты работы, выполненной первой инструкцией.

Код представленный на рисунке ниже отключает неявный режим.



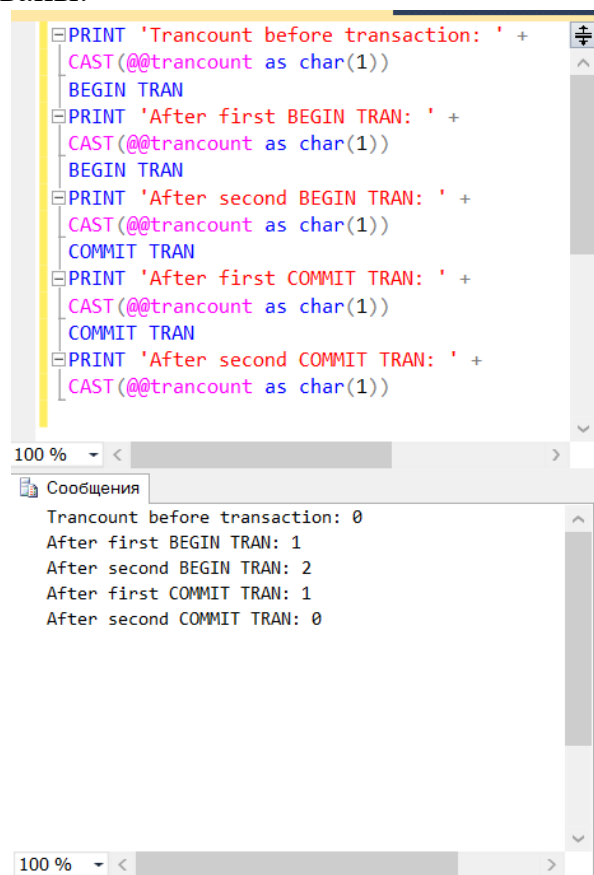
Замечание. Нужно быть особенно внимательны с неявными транзакциями. Важно не забывать выполнять фиксацию или откат сделанных изменений. Поскольку здесь не используется явная инструкция BEGIN

TRANSACTION, об этом легко забыть, что может вызвать длительно работающие транзакции, нежелательные откаты при закрытии соединений и проблемы с блокировками для других соединений.

4. Вложенные транзакции

Явные транзакции могут быть вложенными; это означает, что можно запускать явные транзакции в других явных транзакциях. Одна из основных причин поддержки этого механизма – это разрешение транзакций внутри хранимых процедур, независимо от того, была ли сама процедура вызвана из транзакции.

В следующем примере рассмотрены вложенные транзакции. Из результата видно, что каждая инструкция BEGIN TRAN увеличивает значение функции @@TRANCOUNT на 1, а каждая инструкция COMMIT TRAN уменьшает значение на 1. Как уже известно, значение 0 означает, что не открыто ни одной транзакции. Следовательно, транзакция завершается, когда значение функции @@TRANCOUNT уменьшается от 1 до 0, что происходит при фиксации самой внешней транзакции. Таким образом, каждая внутренняя транзакция требует фиксации. Самая внешняя транзакция определяет, будут ли внутренние транзакции полностью фиксироваться, поскольку эта транзакция запускается первой инструкцией BEGIN TRAN и фиксируется только последней инструкцией COMMIT TRAN. Если эта самая внешняя транзакция не зафиксирована, то вложенные в нее транзакции также не будут зафиксированы.



```

PRINT 'Trancount before transaction: ' +
CAST(@@trancount as char(1))
BEGIN TRAN
PRINT 'After first BEGIN TRAN: ' +
CAST(@@trancount as char(1))
BEGIN TRAN
PRINT 'After second BEGIN TRAN: ' +
CAST(@@trancount as char(1))
COMMIT TRAN
PRINT 'After first COMMIT TRAN: ' +
CAST(@@trancount as char(1))
COMMIT TRAN
PRINT 'After second COMMIT TRAN: ' +
CAST(@@trancount as char(1))

```

Сообщения

```

Trancount before transaction: 0
After first BEGIN TRAN: 1
After second BEGIN TRAN: 2
After first COMMIT TRAN: 1
After second COMMIT TRAN: 0

```

Необходимо помнить, что при использовании вложенных транзакций только самая внешняя транзакция определяет, будут ли зафиксированы

внутренние транзакции. Каждая инструкция COMMIT TRAN всегда применяется к инструкции BEGIN TRAN, которая выполнялась последней. Следовательно, чтобы зафиксировать транзакцию, нужно вызывать инструкцию COMMIT TRAN для каждой выполненной инструкции BEGIN TRAN. Инструкция ROLLBACK TRAN всегда принадлежит самой внешней транзакции и поэтому всегда вызывает откат всей транзакции, независимо от того, сколько вложенных транзакций открыто. По этой причине управление вложенными транзакциями может быть непростым. Вложенные транзакции чаще всего случаются во вложенных хранимых процедурах, где каждая процедура сама по себе запускает транзакцию. Вложенных транзакций можно избежать, если перед решением о том, нужно ли запускать транзакцию, проверять значение функции @@TRANCOUNT в начале этой процедуры. Если значение, возвращаемое функцией @@TRANCOUNT, больше 0, то не обязательно запускать новую транзакцию, поскольку процедура уже находится в состоянии транзакции, и вызывающий экземпляр может вызвать откат этой транзакции, если произойдет ошибка.

2. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные режимы работы с транзакциями.
2. Какой режим работы с транзакциями SQL Server использует по умолчанию?
3. Поясните суть вложенных транзакций.
4. Дайте определение понятию «транзакция».
5. Назовите два типа транзакций.
6. Поясните сокращение ACID относительно свойств транзакции.
7. Назовите следующие инструкции для работы с транзакциями: начало явной транзакции, окончание явной транзакции, использование неявных транзакций, установка точки сохранения транзакций.
8. Дайте определение понятию «распределенная транзакция».
9. Дайте определение понятию «точка сохранения транзакции».
10. Назовите параметр и значение, которое ему необходимо присвоить для разрешения неявных транзакций.
11. Назовите глобальную переменную, которая используется для определения числа активных транзакций для активного пользователя.

ДОМАШНЕЕ ЗАДАНИЕ

[1], страницы 371-391

ЛИТЕРАТУРА

1. Петкович, Д. Microsoft SQL Server 2012. Руководство для начинающих: пер. с английского / Д. Петкович. – СПб.: БХВ-Петербург, 2013. – 816 с.: ил.

2. Сеть разработчиков Microsoft [Электронный ресурс]. – Режим доступа: <https://msdn.microsoft.com/ru-ru/library>

Преподаватель

С.В. Банцевич

Рассмотрено на заседании цикловой
комиссии программного обеспечения
информационных технологий №10
Протокол № от « » _____ 2017
Председатель ЦК С.В. Банцевич