Лаб 2

1) SET XACT\_ABORT используется для автоматического отката текущей транзакции.

Если выполнена инструкция SET XACT\_ABORT ON и вызывается ошибка, вся транзакция завершается и выполняется ее откат.

* SET XACT\_ABORT OFF

SET XACT\_ABORT OFF

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Anton'

WHERE [ID сотрудника] = 4

UPDATE Сотрудник

SET Имя = 'Anya', [Номер телефона] = 'a'

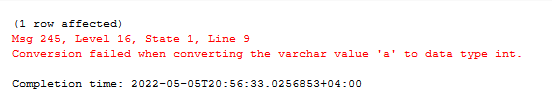
WHERE [ID сотрудника] = 5

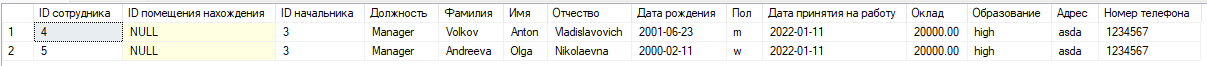
COMMIT

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] IN (4, 5)

**COMMIT**  
Если транзакция выполнена успешно, ее следует зафиксировать. Инструкция COMMIT гарантирует, что все изменения в пределах данной транзакции стали постоянной частью базы данных. Инструкция COMMIT также освобождает используемые транзакцией ресурсы, такие как блокировки.





* SET XACT\_ABORT ON

SET XACT\_ABORT ON

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Anton'

WHERE [ID сотрудника] = 4

UPDATE Сотрудник

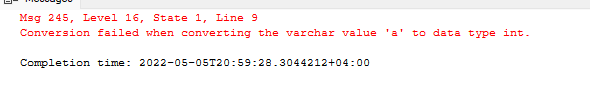
SET Имя = 'Anya', [Номер телефона] = 'a'

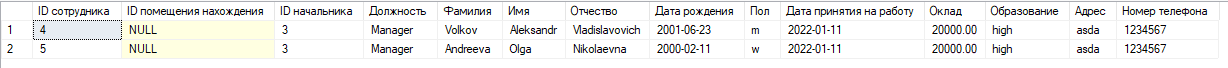
WHERE [ID сотрудника] = 5

COMMIT

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] IN (4, 5)





Автоматический откат транзакции и всех изменений

2) Если задано значение OFF, система находится в *явном* режиме транзакции. Явные транзакции требуют, чтобы пользователь указал начало и конец транзакции.

* SET IMPLICIT\_TRANSACTIONS OFF

SET IMPLICIT\_TRANSACTIONS OFF

UPDATE Сотрудник

SET Имя = 'Anton'

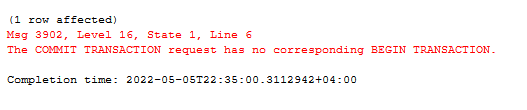
WHERE [ID сотрудника] = 4

COMMIT

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4

SELECT @@TRANCOUNT



Если установлено значение ON, система находится в *неявном* режиме транзакции. Это эквивалентно выполнению невидимой инструкции BEGIN TRANSACTION. В неявном режиме транзакция автоматически начинается при использовании определенных операторов и продолжается, пока не появится оператор явного окончания COMMIT или ROLLBACK

* SET IMPLICIT\_TRANSACTIONS ON

SET IMPLICIT\_TRANSACTIONS ON

UPDATE Сотрудник

SET Имя = 'Anton'

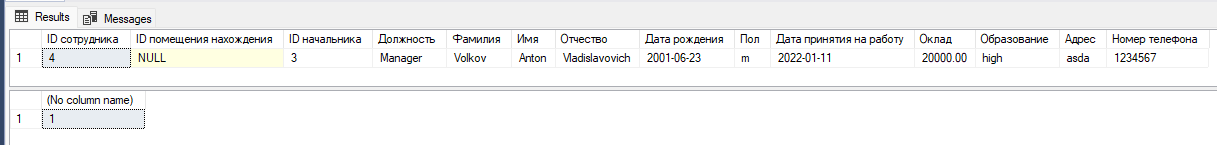
WHERE [ID сотрудника] = 4

COMMIT

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4

SELECT @@TRANCOUNT



@@TRANCOUNT возвращает число инструкций BEGIN TRANSACTION, выполненных в текущем соединении.

Инструкция BEGIN TRANSACTION увеличивает значение @@TRANCOUNT на 1.

Инструкции COMMIT TRANSACTION уменьшает значение @@TRANCOUNT на 1.

Инструкция ROLLBACK TRANSACTION уменьшает значение @@TRANCOUNT до 0.

3)

* **READ UNCOMMITTED**

**Read Uncommitted** – самый низкий уровень изоляции. Позволяет читать "грязные" данные незафиксированых транзакций, отсюда и название феномена – «грязное чтение» (Dirty Read). Суть феномена в том, что если первая транзакция запишет какие-то данные, вторая их прочитает, а потом первая транзакция будет отменена, то получится, что вторая транзакция прочитала данные, которые никогда не существовали.

* Если в первой транзакции поменять данные, вторая транзакция не будет ждать окончания первой и вернет записанные данные незакомиченной транзакции.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

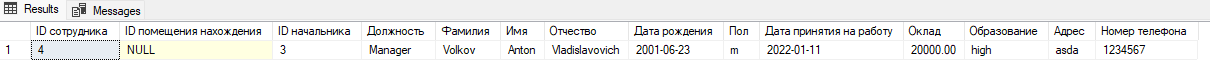
UPDATE Сотрудник

SET Имя = 'Anton'

WHERE [ID сотрудника] = 4

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4



Другая транзакция может прочитать данные первой незакоммиченной транзакции

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

BEGIN TRANSACTION

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4

COMMIT



* Если в одной транзакции считать данные — другая транзакции не будет ждать окончания первой и обновит данные.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Anton'

WHERE [ID сотрудника] = 4

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4

COMMIT



* **READ COMMITTED**

**Read Committed** – при этом уровне изоляции грязное чтение невозможно, то есть второй транзакции не дадут прочитать данные первой до тех пор, пока первая транзакция не зафиксируется. Но при этом уровне изоляции все еще возможна аномалия неповторяющегося чтения. Суть этого феномена в том, что если первая транзакция один раз прочитала данные, а потом вторая их изменила и зафиксировалась, то повторное чтение тех же данных первой транзакцией вернет уже измененные данные.

* Если в одной транзакции поменять данные, то другая транзакция будет ждать окончания первой, чтобы прочитать эти данные.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Kiril'

WHERE [ID сотрудника] = 4

SELECT \* FROM Сотрудник

WHERE [ID сотрудника] = 4



Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION

SELECT \* FROM Сотрудник WHERE [ID сотрудника] = 4

COMMIT



(Выполнение запроса…)

* Если в одной транзакции считать данные — другая транзакции не будет ждать окончания первой и обновит данные.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION

SELECT \* FROM Сотрудник WHERE [ID сотрудника] = 4

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

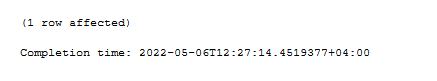
BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Kirilll'

WHERE [ID сотрудника] = 4

COMMIT



* **REPEATABLE READ**

**Repeatable Read** – этот уровень решает предыдущую проблему, но при этом возможно появление фантомов. Изменение однажды прочитанных первой транзакцией данных другими транзакциями (до фиксации первой) невозможно. Однако если первая транзакция сделала выборку по какому-то условию, а потом вторая транзакция добавила новые данные, этому условию удовлетворяющие, и зафиксировалась, то повторная выборка первой транзакцией по тому же условию вернет в том числе и добавленные данные – фантомы.

* Если в одной транзакции считать данные — другая транзакция будет ждать окончания первой, чтобы обновить данные.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRANSACTION

SELECT \* FROM Сотрудник WHERE [ID сотрудника] = 4

Другая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Kiril'

WHERE [ID сотрудника] = 4

COMMIT



* Транзакция не может считывать данные, которые были изменены, но еще не зафиксированы другой транзакцией.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRANSACTION

UPDATE Сотрудник

SET Имя = 'Kirill'

WHERE [ID сотрудника] = 4

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRANSACTION

SELECT \* FROM Сотрудник WHERE [ID сотрудника] = 4

COMMIT

(Выполнение запроса…)

* Если считывать данные в новой транзакции, то уровень изоляции допускает возможность изменения данных другой транзакцией.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRANSACTION

SELECT \* FROM Посетитель

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

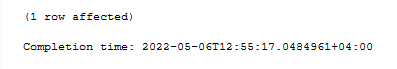
BEGIN TRANSACTION

INSERT INTO Посетитель VALUES(

'Ivanov', 'Alex', 'Ivanovich', NULL)

COMMIT





* **SERIALIZABLE**

**Serializable** – при этом уровне изоляции никакие фантомы невозможны в принципе, равно как и другие феномены, даже такие, которых еще не придумали. Этот уровень изоляции ни на какие феномены не опирается, просто требуется, чтобы результат параллельного выполнения транзакций был таким же, как если бы они выполнялись последовательно.

* Другие транзакции не могут вставлять новые строки со значениями ключа, которые входят в диапазон ключей, считываемых инструкциями текущей транзакции, до ее завершения.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

SELECT \* FROM Посетитель

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

INSERT INTO Посетитель VALUES(

'Ivanov', 'Alex', 'Andreevich', NULL)

COMMIT

(Выполнение запроса…)

* Инструкции не могут считывать данные, которые были изменены другими транзакциями, но еще не были зафиксированы.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

UPDATE Посетитель

SET Имя = 'Aleksandr'

WHERE [ID посетителя] = 5

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

SELECT \* FROM Посетитель

COMMIT



* Другие транзакции не могут изменять данные, считываемые текущей транзакцией, до ее завершения.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

SELECT \* FROM Посетитель

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

UPDATE Посетитель

SET Имя = 'Aleksandr'

WHERE [ID посетителя] = 5

COMMIT

(Выполнение запроса…)

* Разделяемая блокировка

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

SELECT \* FROM Посетитель

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRANSACTION

SELECT \* FROM Посетитель WHERE [ID посетителя] = 5

COMMIT



* **SNAPSHOT**

**SNAPSHOT** является отдельным уровнем изоляции. Указывает на то, что данные, считанные любой инструкцией транзакции, будут согласованы на уровне транзакции с версией данных, существовавших в ее начале. Транзакция распознает только те изменения, которые были зафиксированы до ее начала. Инструкции, выполняемые текущей транзакцией, не видят изменений данных, произведенных другими транзакциями после запуска текущей транзакции.

ALTER DATABASE ВожовАВ2

SET ALLOW\_SNAPSHOT\_ISOLATION ON

* Считывание данных транзакциями моментальных снимков не блокирует запись данных другими транзакциями.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SNAPSHOT

BEGIN TRANSACTION

SELECT \* FROM Посетитель

Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SNAPSHOT

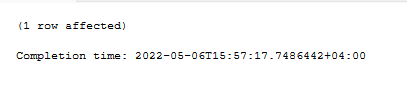
BEGIN TRANSACTION

UPDATE Посетитель

SET Имя = 'Anton'

WHERE [ID посетителя] = 5

COMMIT



* Транзакции, осуществляющие запись данных, не блокируют считывание данных транзакциями моментальных снимков. Инструкции, выполняемые текущей транзакцией, не видят изменений данных, произведенных другими транзакциями после запуска текущей транзакции.

Первая транзакция незакоммичена

SET TRANSACTION ISOLATION LEVEL SNAPSHOT

BEGIN TRANSACTION

UPDATE Посетитель

SET Имя = 'Ivan'

WHERE [ID посетителя] = 5

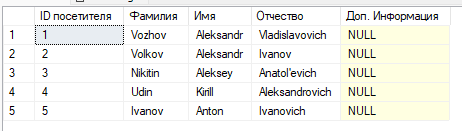
Вторая транзакция закоммичена

SET TRANSACTION ISOLATION LEVEL SNAPSHOT

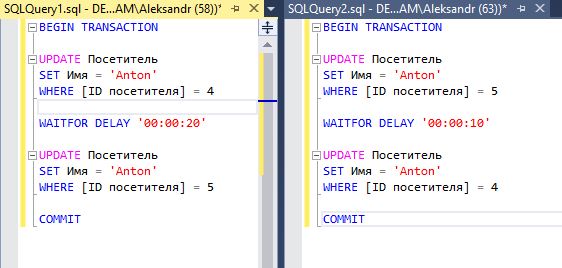
BEGIN TRANSACTION

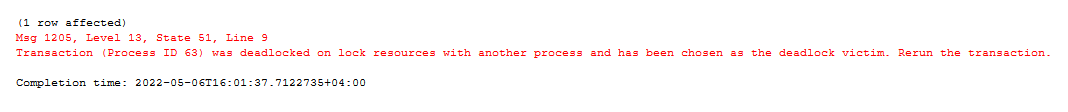
SELECT \* FROM Посетитель

COMMIT



4)





5)

SELECT \* FROM sys.dm\_tran\_locks

Возвращает сведения об активных в данный момент в SQL Server ресурсах диспетчера блокировок.



SELECT \* FROM sys.dm\_tran\_database\_transactions

Возвращает сведения о транзакциях на уровне базы данных.

