

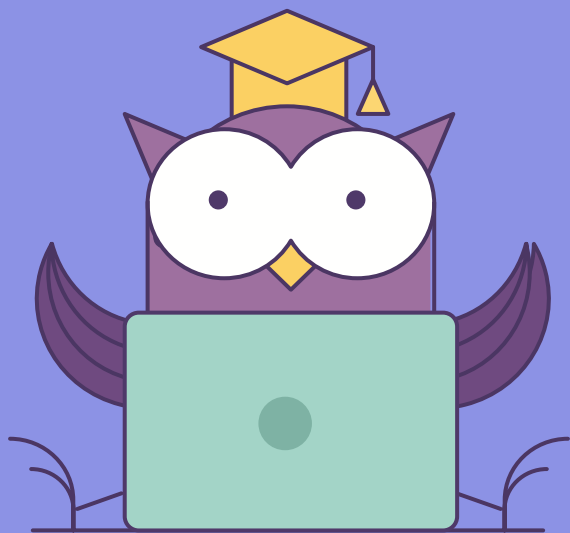


ОНЛАЙН-ОБРАЗОВАНИЕ


Не забыть включить
запись!



Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте  если все хорошо
Или напишите, какие есть проблемы

Уровни изоляции транзакций

Курс “MS SQL Server разработчик”
Группа 2022-02



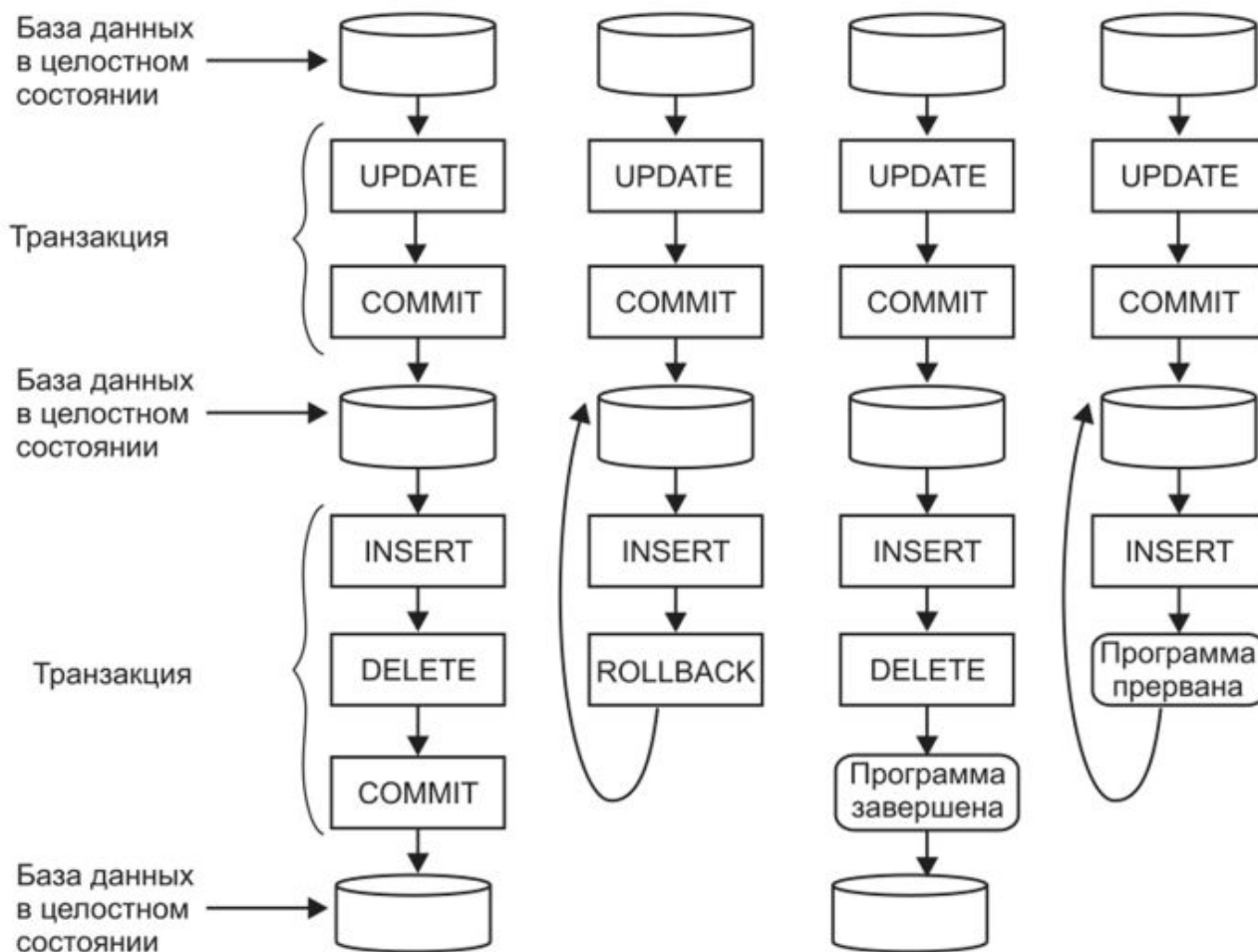
- Что такое "транзакция"
- Свойства ACID
- Уровни изоляции транзакций
- Взаимные блокировки

Что такое транзакции и зачем они нужны?
Ваши варианты.



Транзакция

Транзакция — логическая единица работы, которая переводит БД из одного согласованного состояния в другое.



Команда	Описание
BEGIN TRANSACTION BEGIN TRAN	Начало транзакции
COMMIT TRANSACTION COMMIT TRAN COMMIT WORK COMMIT	Подтверждение транзакции
ROLLBACK TRANSACTION ROLLBACK WORK ROLLBACK	Откат транзакции
SAVE TRANSACTION SAVE TRAN SAVE	Создание точки сохранения транзакции
@@TRANCOUNT	Количество транзакций
XACT_STATE()	Статус текущей транзакции

ACID

ACID:

- **Atomicity** — атомарность
Выполняется все или ничего
- **Consistency** — согласованность
Данные остаются в согласованном состоянии
- **Isolation** — изолированность
Транзакции не видят друг друга
- **Durability** — долговечность
Ничего не потеряется после фиксации транзакции

A

Atomicity

Атомарность

«Все или ничего»

Либо все изменения транзакции фиксируются (commit),
либо все откатываются (rollback)



Вы начали покупку билета, билет заблокировался, но деньги с
вашего счета не списались, администратор «убил» транзакцию.

Билет останется заблокированным?



C

Consistency

Согласованность

Соблюдение целостности данных – никакая транзакция не может примениться (завершиться), если нарушена целостность

ID фильма	Название	Год выпуска	ID жанра
1	Звездные войны	1977	1
2	Один дома	1990	2
3	Пила-99	2017	3
4	Странная история про странного мальчика	2009	9

ID жанра	Жанр
1	Приключения
2	Комедия
3	Ужасы
4	Артхаус



Если у вас есть Foreign Key, Constraints, которые выдали ошибку – значит есть нарушение целостности.

Пример – перевод денег со счета на счет.

- **Автоматический**

Одиночные INSERT, UPDATE, DELETE

- **Ручной**

Явные BEGIN / COMMIT / ROLLBACK

- **Неявные транзакции**

Без BEGIN, только COMMIT / ROLLBACK

SET IMPLICIT_TRANSACTIONS ON

ДЕМО

Транзакции



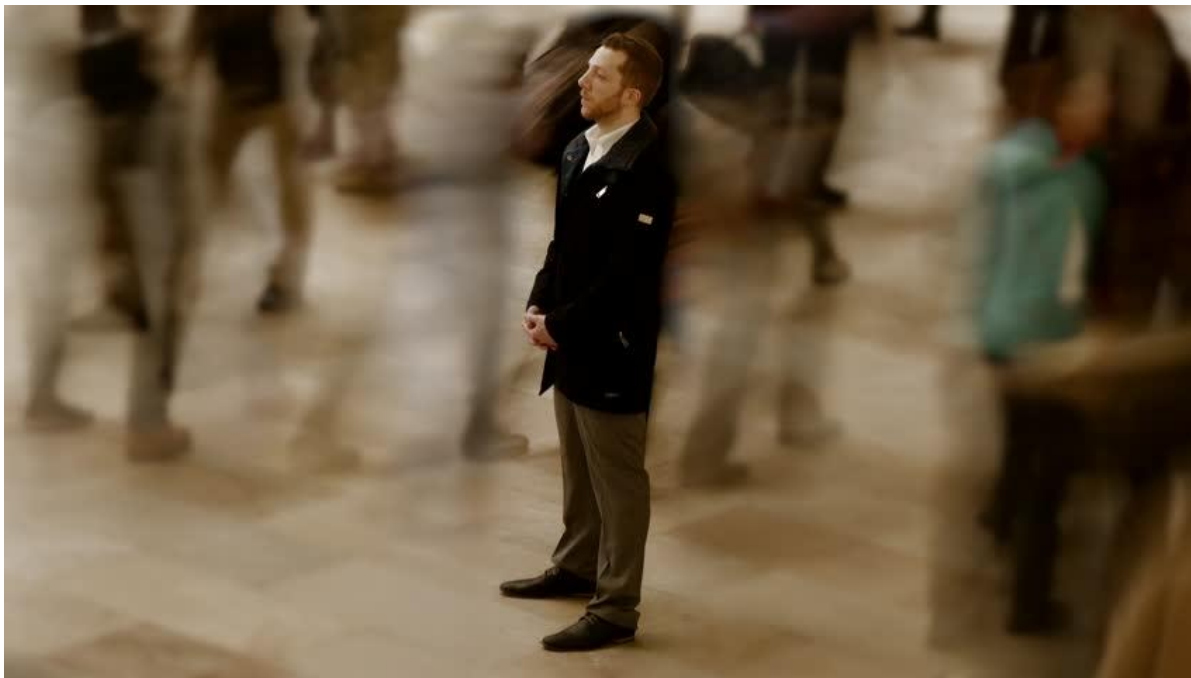


Isolation

Изолированность



Все СУБД многопользовательские, и изоляция – это создание иллюзии того, что пользователь работает в одиночку и не видит незафиксированных транзакций



- **Dirty read** (грязное чтение)
Ситуация когда одна транзакция может прочитать данные измененные другой транзакцией, но не записанные (**uncommitted**).
- **Non repeatable read** (неповторяющееся чтение)
Когда в течении транзакции при повторном чтении одних и тех же строк читаются разные данные.
Был проведен **update** другой транзакцией.
- **Phantom Read** (фантомное чтение)
Когда в течении транзакции при повторном выполнении запроса с одними и теми же условиями результат возвращает разное количество строк.
Был проведен **insert/delete** другой транзакцией.

1. **Read Uncommitted** (NOLOCK)

Чтение незавершенных транзакций

2. **Read Committed** (по умолчанию)

Чтение подтвержденных данных (завершенных транзакций)

3. **Repeatable read**

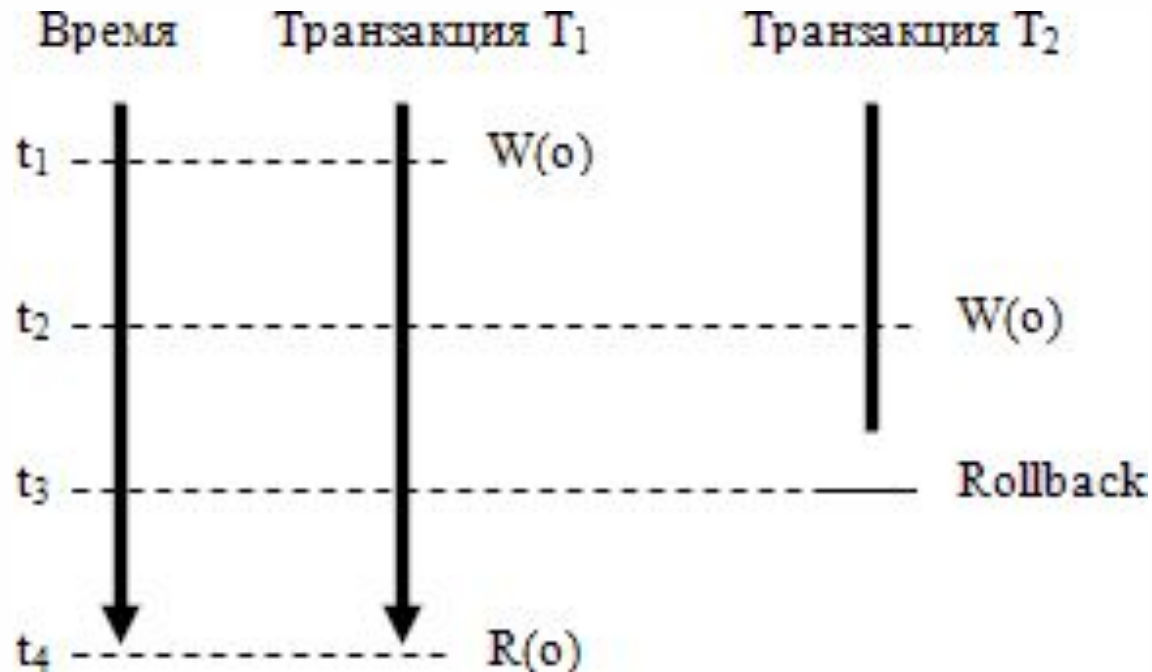
4. **Serializable**

5. **Snapshot**

При разрешенном режиме версионности



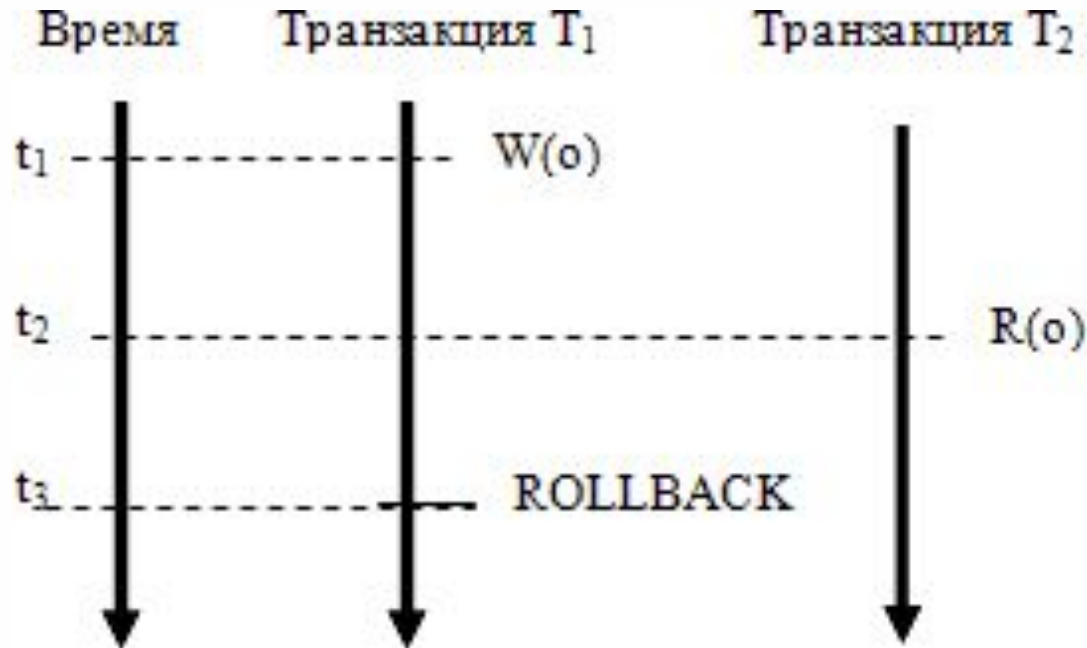
Потерянные обновления (lost update)



В момент времени t_1 транзакция T_1 изменяет объект базы данных o (выполняет операцию $W(o)$). До завершения транзакции T_1 в момент времени $t_2 > t_1$ транзакция T_2 также изменяет объект o . В момент времени $t_3 > t_2$ транзакция T_2 завершается оператором ROLLBACK (например, по причине нарушения ограничений целостности).

Тогда при повторном чтении объекта o (выполнении операции $R(o)$) в момент времени $t_4 > t_3$ транзакция T_1 не видит своих изменений этого объекта, произведенных ранее (в частности, из-за этого может не удастся фиксация этой транзакции, что, возможно, повлечет потерю изменений у еще одной транзакции и т.д.).

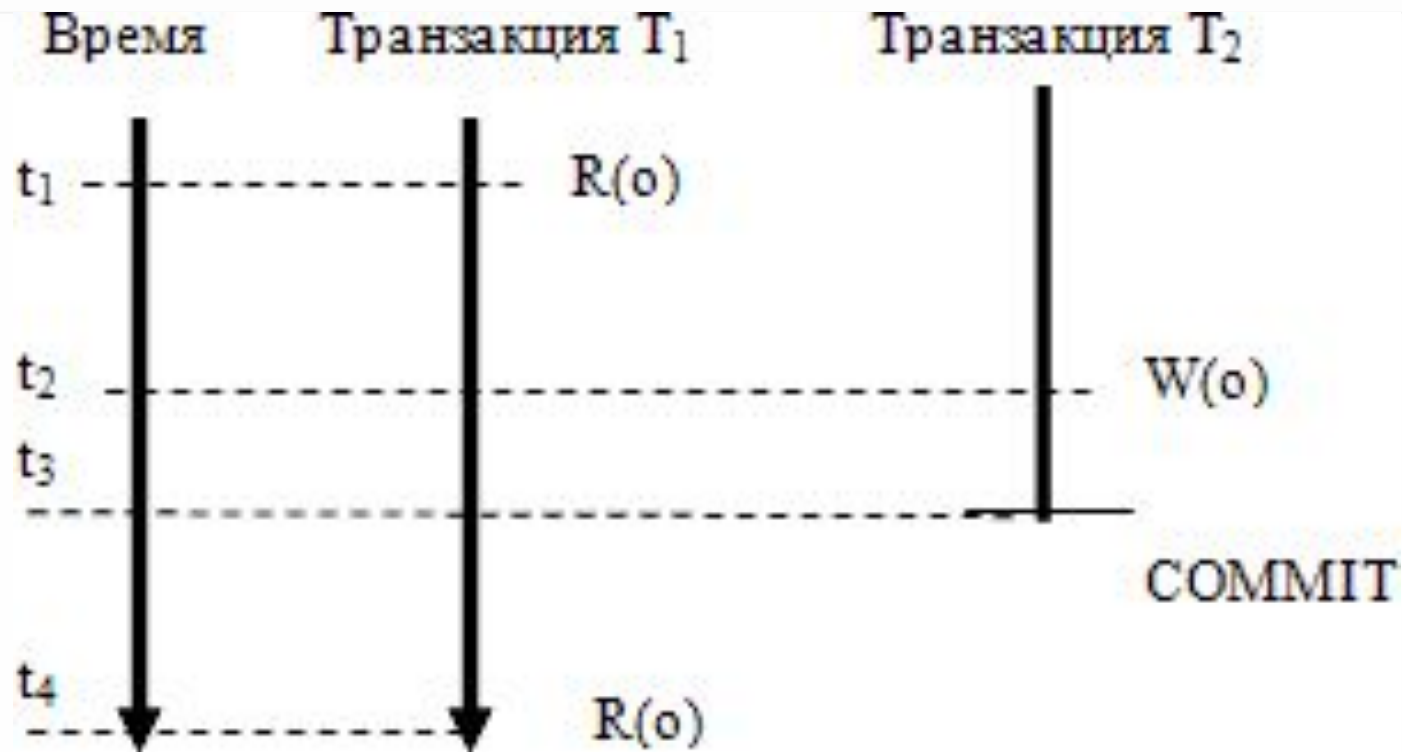
"Грязное чтение" (dirty read)



В момент времени t₁ транзакция T₁ изменяет объект базы данных o (выполняет операцию W(o)). В момент времени t₂ > t₁ транзакция T₂ читает объект o (выполняет операцию R(o)). Поскольку транзакция T₁ еще не завершена, транзакция T₂ видит несогласованные «грязные» данные. В частности, в момент времени t₃ > t₂ транзакция T₁ может завершиться откатом (например, по причине нарушения ограничений целостности).

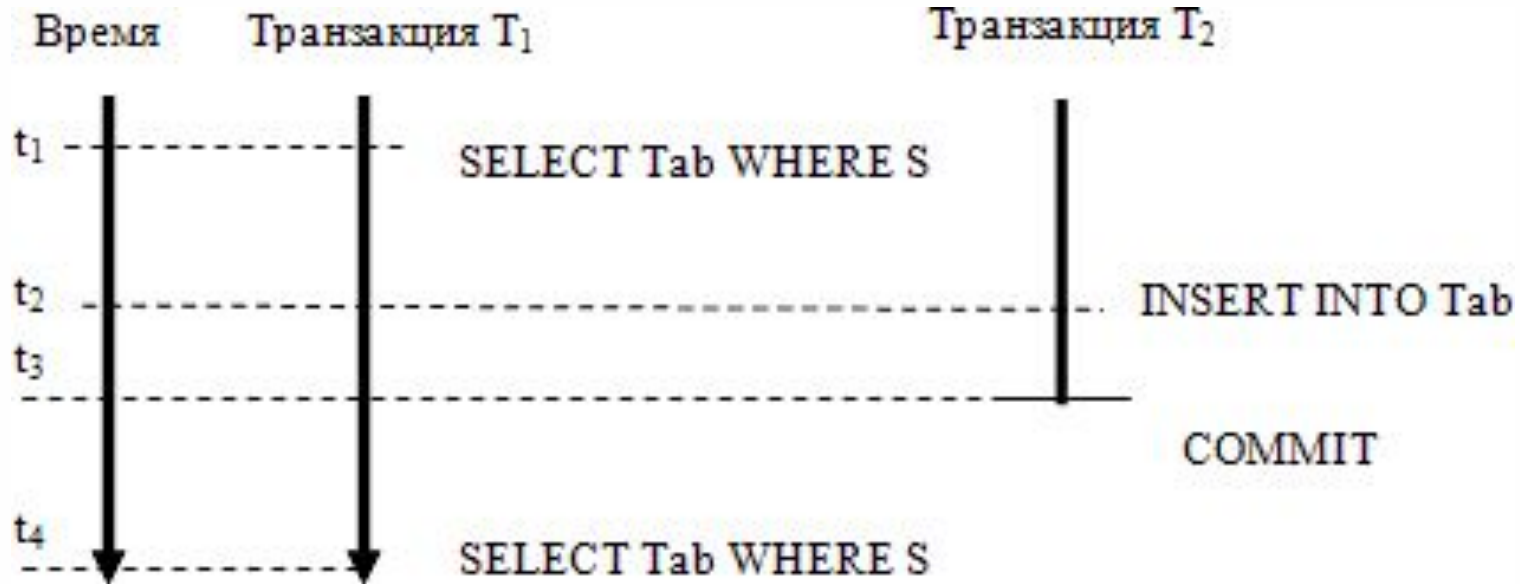
Неповторяющиеся чтения (non repeatable read)

○ ∩ U S



В момент времени t_1 транзакция T_1 читает объект базы данных o (выполняет операцию $R(o)$). До завершения транзакции T_1 в момент времени $t_2 > t_1$ транзакция T_2 изменяет объект o (выполняет операцию $W(o)$) и успешно завершается оператором COMMIT. В момент времени $t_3 > t_2$ транзакция T_1 повторно читает объект o и видит его измененное состояние.

Фантомное чтение (phantom read)



В момент времени t_1 транзакция T_1 выполняет оператор выборки кортежей таблицы Tab с условием выборки S (т.е. выбирается часть кортежей таблицы Tab, удовлетворяющих условию S). До завершения транзакции T_1 в момент времени $t_2 > t_1$ транзакция T_2 вставляет в таблицу Tab новый кортеж r , удовлетворяющий условию S, и успешно завершается. В момент времени $t_3 > t_2$ транзакция T_1 повторно выполняет тот же оператор выборки, и в результате появляется кортеж, который отсутствовал при первом выполнении оператора.

Аномалии и уровни изоляции транзакций

	uncommitted	committed	
	update, insert, delete	update	insert, delete
Аномалия Уровень изоляции	Dirty read	Non repeatable read	Phantom read
Read Uncommitted	Возможно	Возможно	Возможно
Read Committed (по умолчанию)	Не возможно	Возможно	Возможно
Repeatable read	Не возможно	Не возможно	Возможно
Serializable	Не возможно	Не возможно	Не возможно
Snapshot	Не возможно	Не возможно	Не возможно

Не возможно — аномалия не проявляется

Возможно — аномалия проявляется

* Есть еще lost update (потерянное обновление), но в SQL Server не проявляется.

1. Соединение (сессия)

```
SET TRANSACTION ISOLATION LEVEL
```

```
{ READ UNCOMMITTED
```

```
| READ COMMITTED
```

```
| REPEATABLE READ
```

```
| SNAPSHOT
```

```
| SERIALIZABLE
```

```
}
```

2. Запрос (NOLOCK, READUNCOMMITTED)

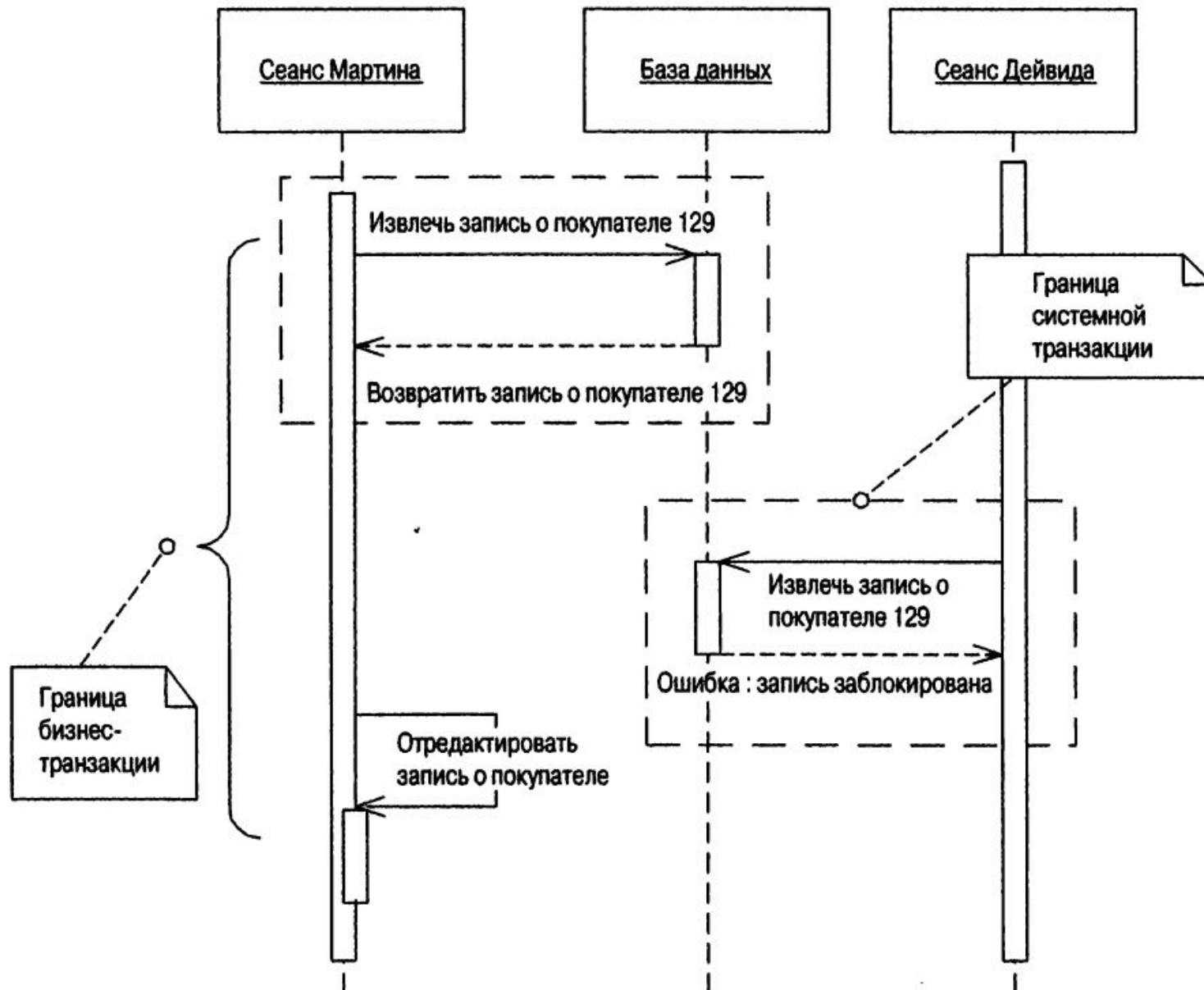
```
SELECT * FROM Sales.Orders WITH (NOLOCK)
```

```
SELECT * FROM Sales.Orders (NOLOCK)
```

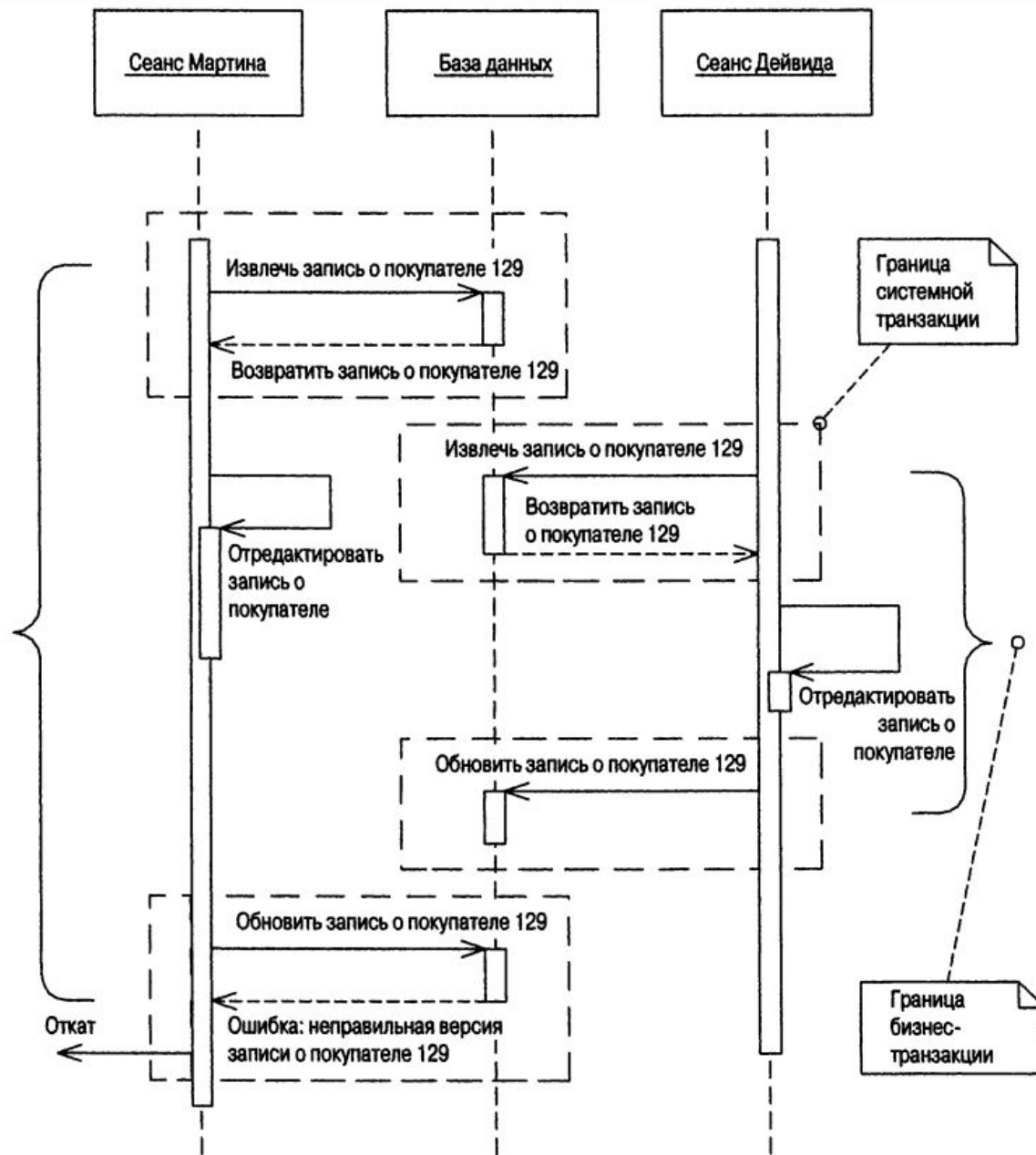
```
SELECT * FROM Sales.Orders WITH (READUNCOMMITTED)
```

```
SELECT * FROM Sales.Orders (READUNCOMMITTED)
```

Что такое пессимистическая блокировка



Что такое оптимистическая блокировка



БД в SQL Server может работать в двух режимах:

- LOCK (блокировки) — пессимистичный
- SNAPSHOT (версионность строк) — оптимистичный

В других СУБД это называется MVCC (multiversion concurrency control — управление параллельным доступом посредством многоверсионности)

- C SQL Server 2005
- Используется tempdb (с 2019 Accelerated Database Recovery Persistent Version Store, ADR PVS в пользовательской БД)

- **READ COMMITED SNAPSHOT**

ALTER DATABASE

SET READ_COMMITTED_SNAPSHOT ON;

- **SNAPSHOT**

ALTER DATABASE

SET ALLOW_SNAPSHOT_ISOLATION ON;

<http://sqlcom.ru/dba-tools/sql-server-and-snapshot-isolation-level/>

<https://infostart.ru/1c/articles/708848/>

<https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/sql/snapshot-isolation-in-sql-server>

ДЕМО

Изоляция транзакций



- `SELECT * FROM Table WITH (NOLOCK)`
- Блокировка Sch-S (Schema stability lock)
подробнее про NOLOCK можно почитать [здесь](#)
- Запросы для мониторинга SQL Server
- Запросы для просмотра прогресса выполнения
длительного запроса
- Не использовать в нормальных “бизнес-ситуациях”

Если мы делаем UPDATE в одной транзакции, то увидим ли мы изменения в другой транзакции при уровне изоляции **READ COMMITTED**?

	uncommitted	committed	
	update, insert, delete	update	insert, delete
Аномалия Уровень изоляции	Dirty read	Non repeatable read	Phantom read
Read Uncommitted	Возможно	Возможно	Возможно
Read Committed (по умолчанию)	Не возможно	Возможно	Возможно
Repeatable read	Не возможно	Не возможно	Возможно
Serializable	Не возможно	Не возможно	Не возможно
Snapshot	Не возможно	Не возможно	Не возможно

Уровни изоляции - READ COMMITTED

Если мы делаем UPDATE в одной транзакции, то увидим ли мы изменения в другой транзакции при уровне изоляции **READ COMMITTED**?

	uncommitted	committed	
	update, insert, delete	update	insert, delete
Аномалия Уровень изоляции	Dirty read	Non repeatable read	Phantom read
Read Uncommitted	Возможно	Возможно	Возможно
Read Committed (по умолчанию)	Не возможно	Возможно	Возможно
Repeatable read	Не возможно	Не возможно	Возможно
Serializable	Не возможно	Не возможно	Не возможно
Snapshot	Не возможно	Не возможно	Не возможно

Если мы делаем UPDATE в одной транзакции, то увидим ли мы изменения в другой транзакции при уровне изоляции **REPEATABLE READ**?

	uncommitted	committed	
	update, insert, delete	update	insert, delete
Аномалия Уровень изоляции	Dirty read	Non repeatable read	Phantom read
Read Uncommitted	Возможно	Возможно	Возможно
Read Committed (по умолчанию)	Не возможно	Возможно	Возможно
Repeatable read	Не возможно	Не возможно	Возможно
Serializable	Не возможно	Не возможно	Не возможно
Snapshot	Не возможно	Не возможно	Не возможно

Уровни изоляции - READ COMMITTED

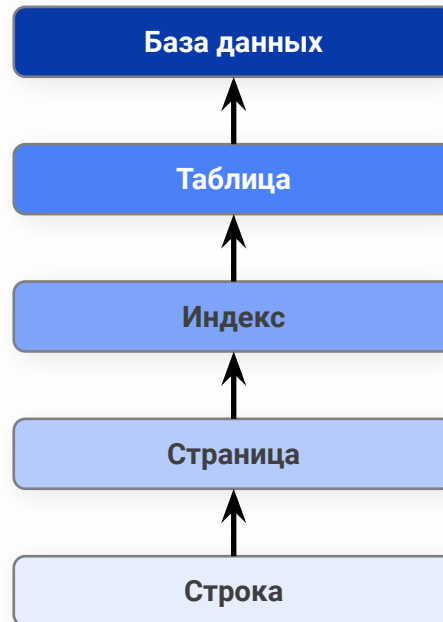
Если мы делаем UPDATE в одной транзакции, то увидим ли мы изменения в другой транзакции при уровне изоляции **REPEATABLE READ**?

	uncommitted	committed	
	update, insert, delete	update	insert, delete
Аномалия Уровень изоляции	Dirty read	Non repeatable read	Phantom read
Read Uncommitted	Возможно	Возможно	Возможно
Read Committed (по умолчанию)	Не возможно	Возможно	Возможно
Repeatable read	Не возможно	Не возможно	Возможно
Serializable	Не возможно	Не возможно	Не возможно
Snapshot	Не возможно	Не возможно	Не возможно

Блокировка строки

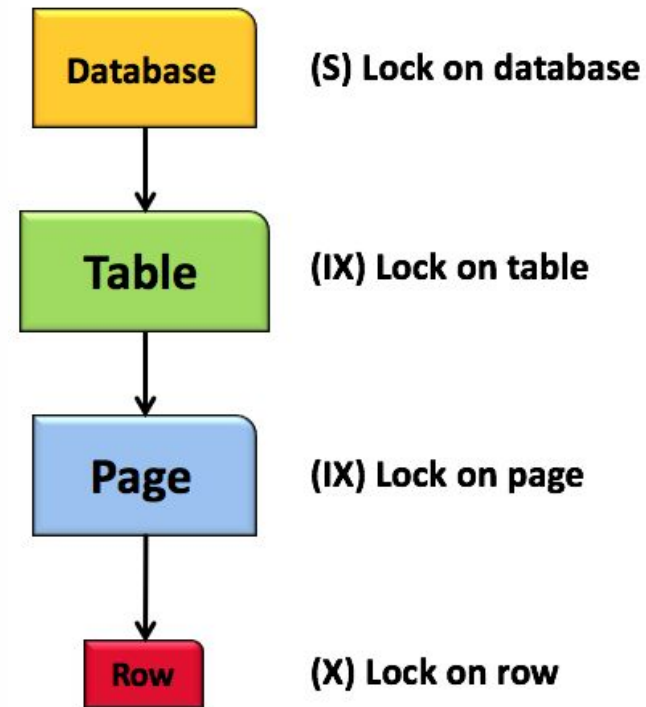
Если сервер понимает, что ему нужно заблокировать много строк – он думает а не заблокировать ли страницы.

А если много страниц – может дешевле заблокировать всю таблицу?



Какие блокировки бывают?

1. **Shared (S)** - разделяемая
SELECT
2. **Update (U)** - на изменение записей
UPDATE
3. **Exclusive (X)** – эксклюзивная
UPDATE, DELETE
4. **Intend (I*)** - намерения

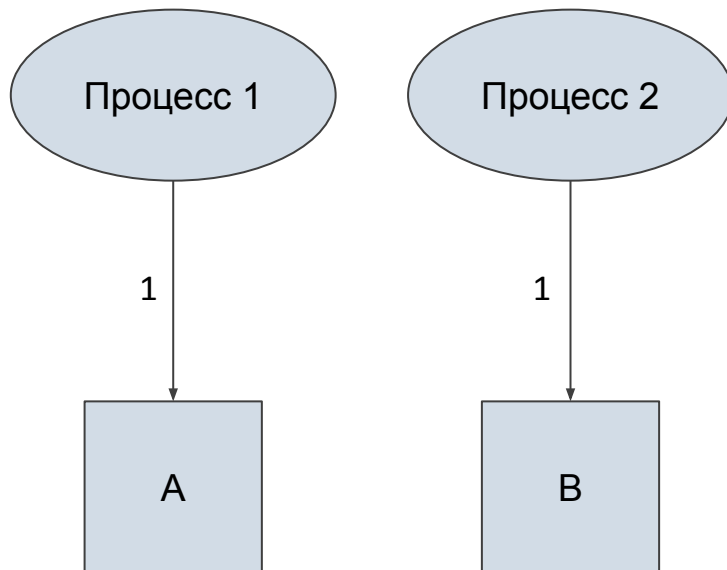


Requested mode	Existing granted mode					
	IS	S	U	IX	SIX	X
Intent shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No
Intent exclusive (IX)	Yes	No	No	Yes	No	No
Shared with intent exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver15#lock_compatibility

Взаимная блокировка

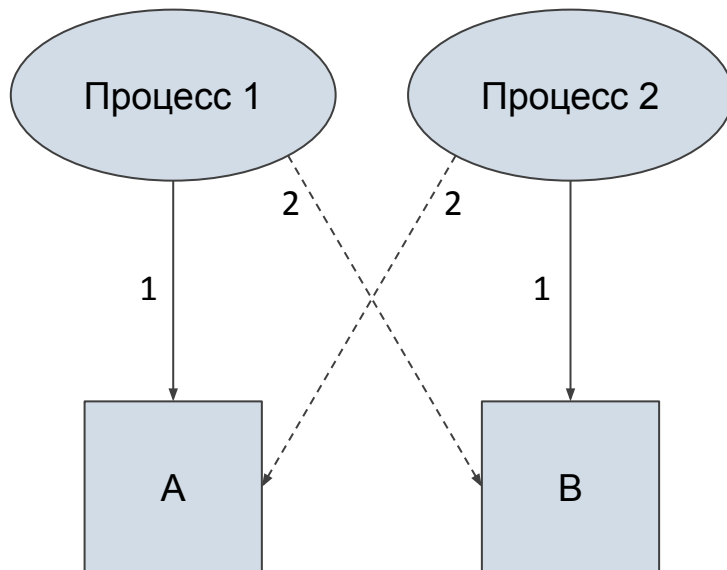
Взаимная блокировка (deadlock) — ситуация в многозадачной среде или СУБД, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжить свое выполнение.



Шаг	Процесс 1	Процесс 2
0	Хочет захватить A и B, начинает с A	Хочет захватить A и B, начинает с B
1	Захватывает ресурс A	Захватывает ресурс B

Взаимная блокировка

Взаимная блокировка (deadlock) — ситуация в многозадачной среде или СУБД, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжить свое выполнение.



Шаг	Процесс 1	Процесс 2
0	Хочет захватить A и B, начинает с A	Хочет захватить A и B, начинает с B
1	Захватывает ресурс A	Захватывает ресурс B
2	Ожидает освобождения ресурса B	Ожидает освобождения ресурса A
3	Взаимная блокировка	

Как вы узнаете, что у вас есть дедлок?

Msg 1205, Level 13, State 51, Line 13

Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.

- Trace Flag (флаги трассировки) – 1204, 1222
 - DBCC TRACEON(1204,-1)
 - DBCC TRACESTATUS(1204)
 - DBCC TRACEOFF(1204,-1)
- Profiler
- Extended Events
- SET DEADLOCK_PRIORITY LOW | NORMAL | HIGH

в качестве жертвы выбирается сеанс с более низким приоритетом

ДЕМО

Deadlock



- При дедлоке можно просто упасть или попытаться рестартовать транзакцию
- Стараться делать транзакции короткими



D

Durability
Долговечность

Все что зафиксировано, должно остаться в БД.

Используется Transaction Log (журнал транзакций)

Write-ahead log или упреждающая журнализация.

Сначала пишется в журнал транзакций, потом в файл данных.

- Транзакции (откат)
- Восстановление после сбоя
- Высокая доступность (Log Shipping, репликация и т.д.)

Что пишется в лог

- изменения данных (insert, update, delete)
- создание, перестроение индексов
- ...

RECOVERY MODEL

- FULL
- BULK_LOGGED
- SIMPLE

- [Ведение журнала и восстановление в SQL Server](#)
- [Чтение журнала транзакций SQL Server](#)
- [Understanding log buffer flushes](#)

Документация SQL Server

- [Журнал транзакций \(SQL Server\) - SQL Server](#)
- [Руководство по архитектуре журналов транзакций и управлению ими - SQL Server](#)

Домашнее задание

“Нормального” ДЗ нет. Есть материал для самостоятельного изучения.

Вложенные транзакции и **SAVEPOINT**:

- Understanding SQL Server Transaction Savepoints

<https://www.mssqltips.com/sqlservertip/5538/understanding-sql-server-transaction-savepoints/>

- Nesting Transactions

[https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189336\(v=sql.105\)](https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189336(v=sql.105))

- SQL Server DBA myth: nested transactions are real

<https://www.sqlskills.com/blogs/paul/a-sql-server-dba-myth-a-day-2630-nested-transactions-are-real/>

Рефлексия

О чем мы говорили сегодня?

- Что такое транзакции?
- Какие уровни изоляции бывают?
- Блокировки, взаимные блокировки?



Рефлексия

Напишите, пожалуйста, свое впечатление о вебинаре.

- Отметьте 3 пункта, которые вам запомнились с вебинара.
- Что вы будете применять в работе из сегодняшнего вебинара?



**Пройдите, пожалуйста
опрос**

**Спасибо
за внимание!**

