ТРАНЗАКЦИИ

Что такое транзакция?

- **Транзакция –** это один или несколько операторов SQL, выполняемых как единая логическая единица работы (задача)
  - Промежуточные состояния данных, изменяемых операторами внутри транзакции, не должны быть видны другим транзакциям

- Транзакции являются одним из средств обеспечения целостности данных
  - Если какой-либо оператор внутри транзакции не может быть успешно завершен, изменения других операторов не должны сохранится в базе данных

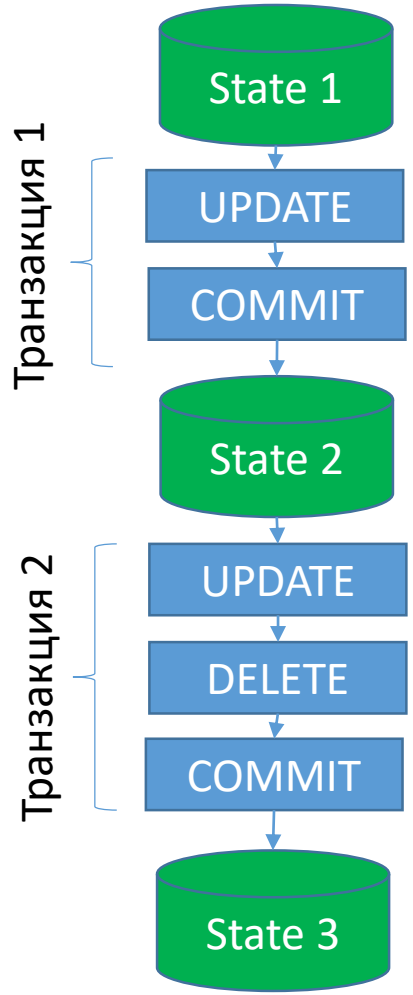
Соответствие требованиям ACID

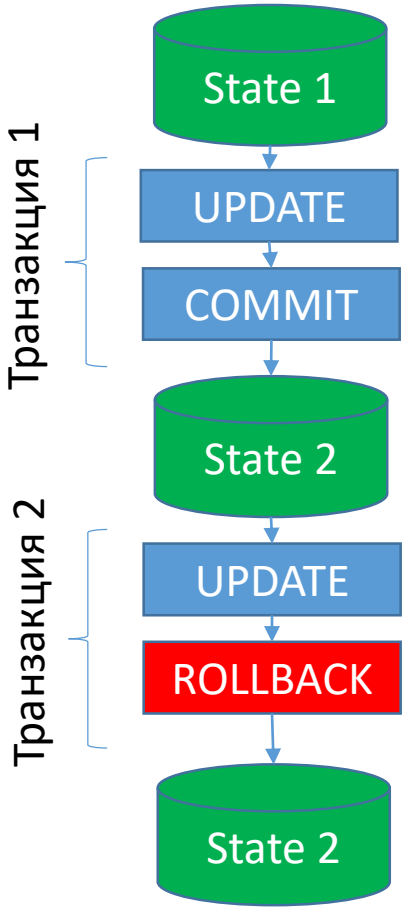
- Для соблюдения целостности данных транзакции должны соответствовать требованиям **ACID**:
  - **Атомарность** (**A**tomicity)
    - это условие, при котором либо транзакция успешно выполняется целиком, либо, если какая-либо из ее частей не выполняется, вся транзакция отменяется
  - **Согласованность** (**C**onsistency)
    - это условие, при котором данные, записываемые в базу данных в рамках транзакции, должны соответствовать всем правилам и ограничениям, включая ограничения целостности
  - **Изоляция** (**I**solation)
    - необходима для контроля над согласованностью и гарантирует базовую независимость каждой транзакции
  - **Надежность** (**D**urability)
    - подразумевает, что все внесенные в базу данных изменения на момент успешного завершения транзакции считаются постоянными

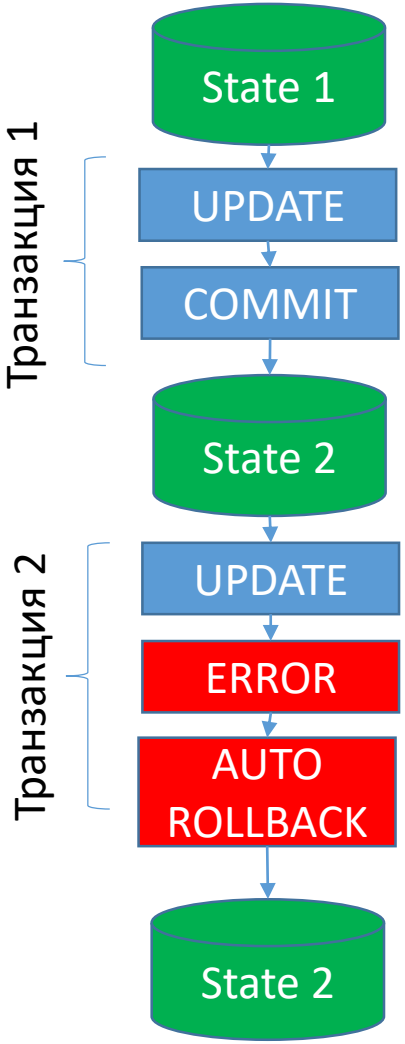
Определение

- Транзакцией называется множество операций, выполняемое приложением, которое переводит базу данных из одного корректного состояния в другое корректное состояние (**согласованность**) при условии, что транзакция выполнена полностью (**атомарность**) и без помех со стороны других транзакций (**изоляция**)

Модель транзакции

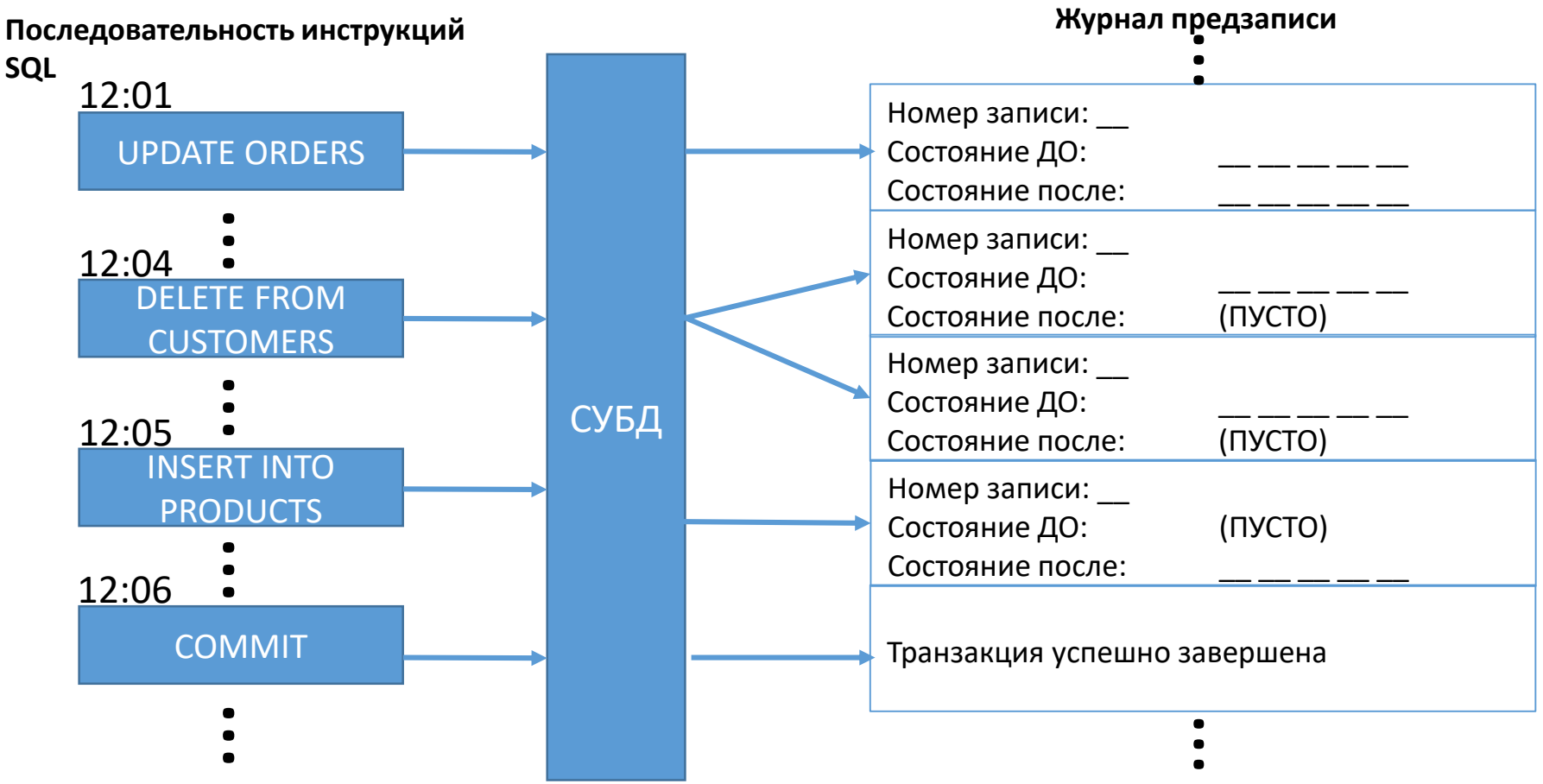






Журнал предварительной записи (WAL)

WAL – механизм, который позволяет восстановить согласованность данных после сбоя



Режимы транзакций

Типы транзакций

- AUTOCOMMIT
- Явные транзакции

Режим транзакций: AUTOCOMMIT

```
CREATE TABLE
"TestBatch" (cola INT PRIMARY KEY, colb CHAR(3)); --1 транзакция

INSERT INTO "TestBatch" VALUES (1, 'aaa'); --2 транзакция
INSERT INTO "TestBatch" VALUES (2, 'bbb'); --3 транзакция
INSERT INTO "TestBatch" VALUSE (3, 'ccc'); --4 транзакция
                                    --Синтаксическая ошибка

SELECT *
FROM "TestBatch" ; --5 транзакция
```

	cola	colb
1	1	aaa
2	2	bbb

Явные транзакции

- BEGIN
- COMMIT
- ROLLBACK
- SAVEPOINT

Откат транзакции

	ClientID	Lname	Fname	City	RegDate
1	6	Sergeev	Sven	London	2022-05-12

	ClientID	Lname	Fname	City	RegDate
1	6	Sergeev	Anton	London	2022-05-12

	ClientID	Lname	Fname	City	RegDate
1	6	Sergeev	Sven	London	2022-05-12

```
begin;
    SELECT "ClientID","Lname","Fname","City","RegDate"
    FROM public.client
    WHERE "ClientID"=6;

    UPDATE public.client
    SET "Fname" = 'Anton'
    WHERE "ClientID"=6;
--Сделали проверку и поняли, что ошиблись
    SELECT "ClientID","Lname","Fname","City","RegDate"
    FROM public.client
    WHERE "ClientID"=6;
rollback;

SELECT "ClientID","Lname","Fname","City","RegDate"
FROM public.client
WHERE "ClientID"=6;
```

Фиксация транзакции

	orderid	productid	unitprice	qty	discount
1	10 875	19	$9.20	25	0
2	10 875	47	$9.50	21	0,1
3	10 875	49	$20.00	15	0
4	10 924	10	$31.00	20	0,1
5	10 924	28	$45.60	30	0,1
6	10 924	75	$7.75	6	0

У нас есть 2 заказа: 10875 и 10924
Клиент попросил перенести одну из позиций заказа 10875 в заказ 10924.

Чтобы реализовать данный перенос необходимо:

1) Добавить в заказ 10924 детали выбранной позиции
2) Удалить выбранную позицию из заказа 10875
3) Необходимо быть внимательными и не удалить, например, все детали 10875 заказа!

```
BEGIN;
    insert into "Sales"."OrderDetails"
    select 10924, productid, unitprice, qty, discount
    from "Sales"."OrderDetails"
    where orderid = 10875 and productid = 19;
SAVEPOINT my_savepoint;
    delete  from "Sales"."OrderDetails"
    where orderid = 10875;
-- Ошибочное действие... Его нужно забыть!
-- Необходимо удалить только одну запись
ROLLBACK TO my_savepoint;
    delete  from "Sales"."OrderDetails"
    where orderid = 10875 and productid = 19;
COMMIT;
SELECT *
FROM "Sales"."OrderDetails"
WHERE orderid IN (10875, 10924);
```

Особенности

- Поведение некоторых функций и типов данных PostgreSQL в транзакциях подчиняется особым правилам

- **Например**, изменения последовательностей (и следовательно, счётчика в столбце, объявленном как **serial**)
  - немедленно видны во всех остальных транзакциях
  - не откатываются назад, даже если выполнившая их транзакция прерывается – возможны «дыры»

Проблемы конкурентного доступа к данным

Одновременный доступ к данным

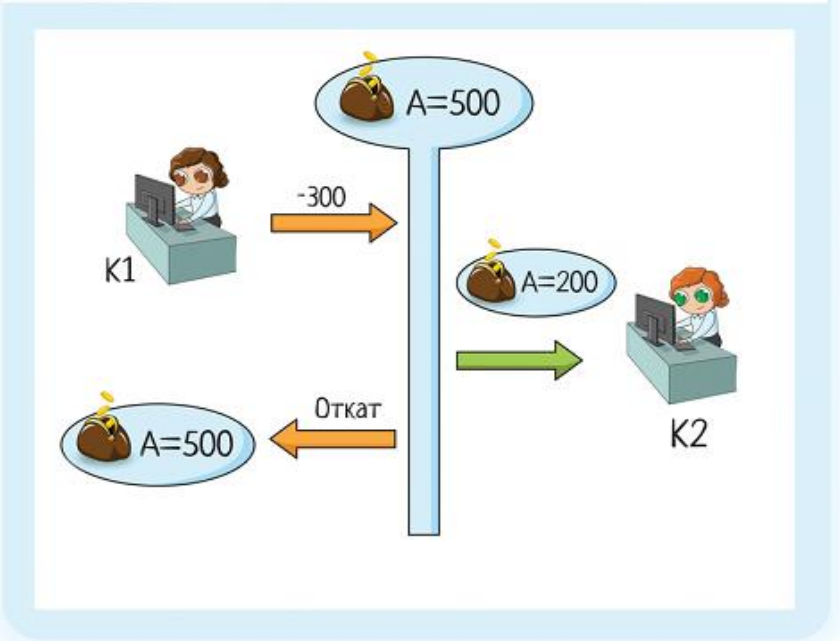
- В один момент времени к одним и тем же данным могут обращаться различные сеансы
- Возможны следующие ситуации:
  - Две транзакции обращаются к одним и тем же данным с запросом на чтение
  - Одна транзакция читает данные, а вторая в это время запрашивает модификацию этих данных
  - Одна транзакция изменяет данные, а вторая в это время запрашивает эти данные на чтение
  - Две транзакции обращаются к одним и тем же данным с запросом на модификацию этих данных
- ***Если две и более транзакций обращаются к одним и тем же данным с запросом на чтение проблем не возникает!***

Проблемы конкурентного доступа

- Грязное чтение (Dirty read)
- Потерянное обновление (Lost update)
- Неповторяемое чтение (Non-repeatable read)
- Фантомное чтение (Phantom read)
- Двойное чтение (Double read)

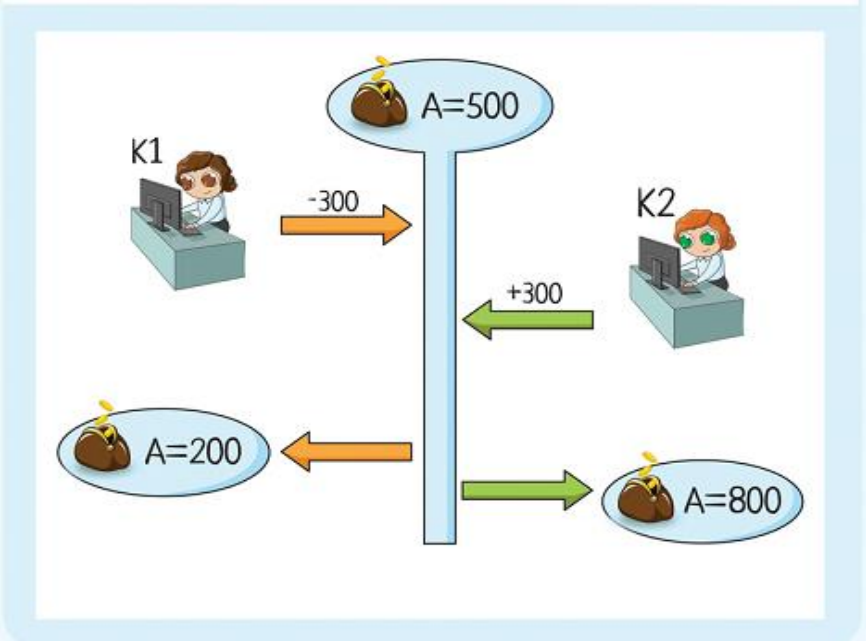
Грязное чтение и Потерянное обновление

Dirty read



Транзакция читает данные, записанные параллельной **<u>незавершённой</u>** транзакцией

Lost update

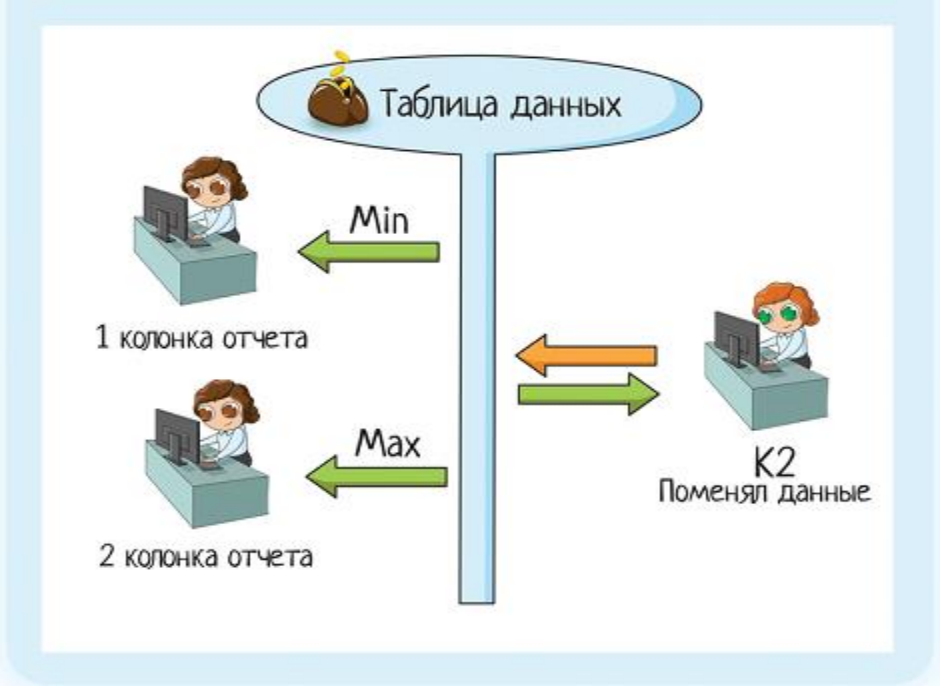


Выполняются два одновременных обновления; результат первого обновления потерян

https://habr.com/ru/post/555920

Неповторяемое чтение и Фантомное чтение

Non-repeatable read



При повторном чтении данных транзакция обнаруживает, что они были изменены и уже зафиксированы другой транзакцией

Phantom read\Double read

Первая транзакция читает данные, удовлетворяющие некоторому условию.
В это время вторая транзакция добавляет/удаляет/изменяет записи, удовлетворяющие этому условию