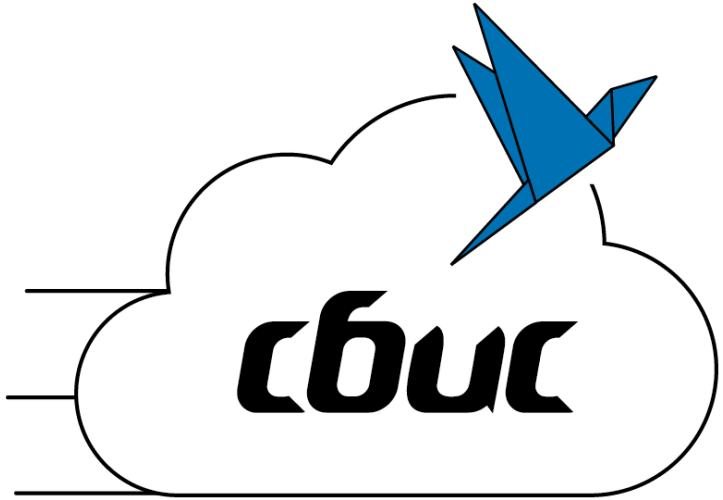


PostgreSQL для начинающих

#6: транзакции

Кирилл Боровиков / Компания «Тензор», технический директор / explain.tensor.ru, sbis.ru

Транзакции

Транза́кция (англ. *transaction*) — **группа последовательных операций** с базой данных, которая представляет собой логическую единицу работы с данными.

Транзакция может быть выполнена либо **целиком и успешно**, соблюдая целостность данных и независимо от параллельно идущих других транзакций, **либо не выполнена вообще**, и тогда она не должна произвести никакого эффекта.

Транзакции обрабатываются транзакционными системами, в процессе работы которых создаётся история транзакций.

https://ru.wikipedia.org/wiki/Транзакция_(информатика)

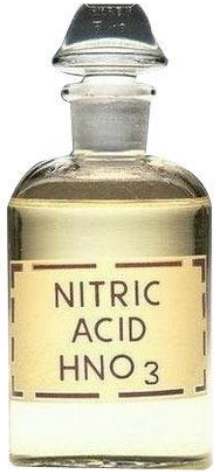
ACID

Atomicity – атомарность

Consistency – согласованность

Isolation – изоляция

Durability – устойчивость

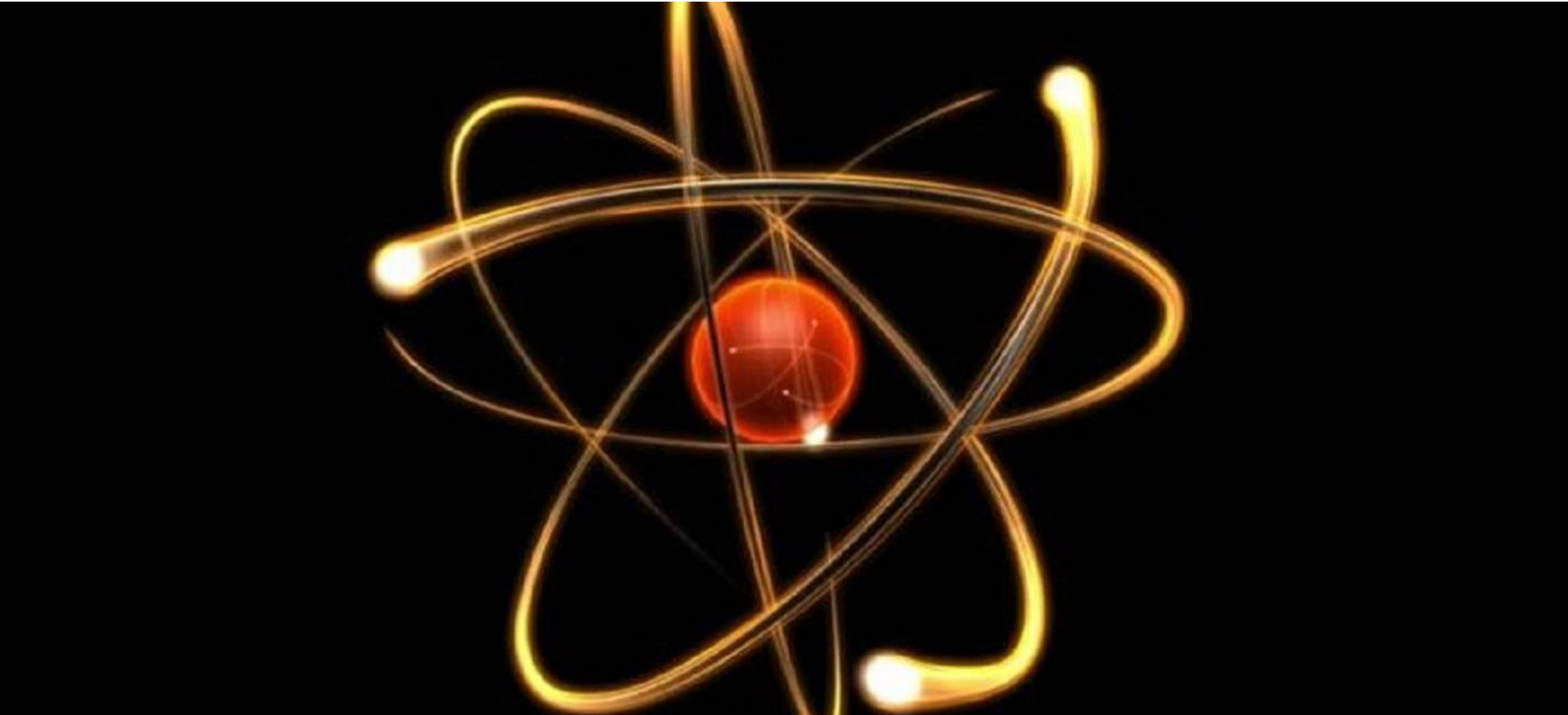


https://ru.wikipedia.org/wiki/ACID

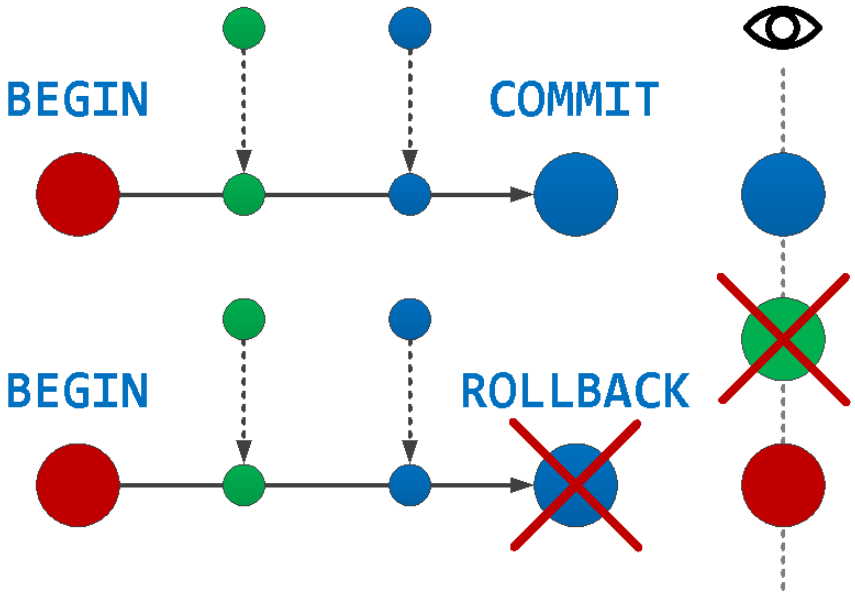
* ключевые требования к **транзакционной** системе

Atomicity

атомарность



Atomicity



Atomicity

BEGIN – начать транзакцию

https://postgrespro.ru/docs/postgresql/15/sql-begin

COMMIT – успешное завершение транзакции

https://postgrespro.ru/docs/postgresql/15/sql-commit

ROLLBACK – откат транзакции

https://postgrespro.ru/docs/postgresql/15/sql-rollback

Atomicity

```
BEGIN;
CREATE TABLE x AS SELECT 1 i;
ROLLBACK;


BEGIN;
CREATE TABLE x AS SELECT 1 i;
COMMIT;
```

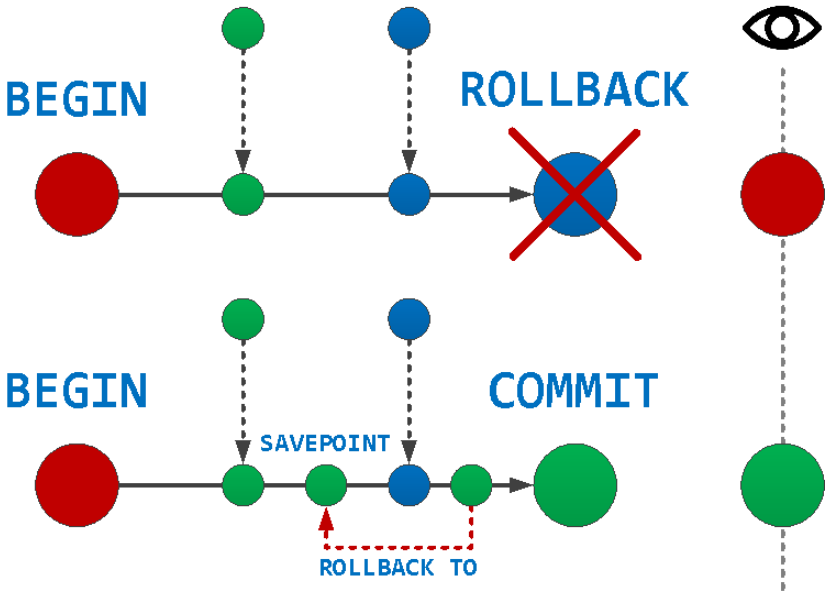
```
TABLE x;
ERROR:  relation "x" does not exist

TABLE x;
ERROR:  relation "x" does not exist

TABLE x;
-- i : 1
```

Atomicity

```
-- autocommit = ON ->
CREATE TABLE x AS SELECT 1 i;
```

```
BEGIN;
CREATE TABLE x AS SELECT 1 i;
COMMIT;
```

Atomicity



Atomicity

SAVEPOINT ***имя*** – создать точку сохранения

https://postgrespro.ru/docs/postgresql/15/sql-savepoint

ROLLBACK TO ***имя*** – откатиться до точки сохранения

https://postgrespro.ru/docs/postgresql/15/sql-rollback-to

RELEASE ***имя*** – удалить точку сохранения

https://postgrespro.ru/docs/postgresql/15/sql-release-savepoint

Atomicity

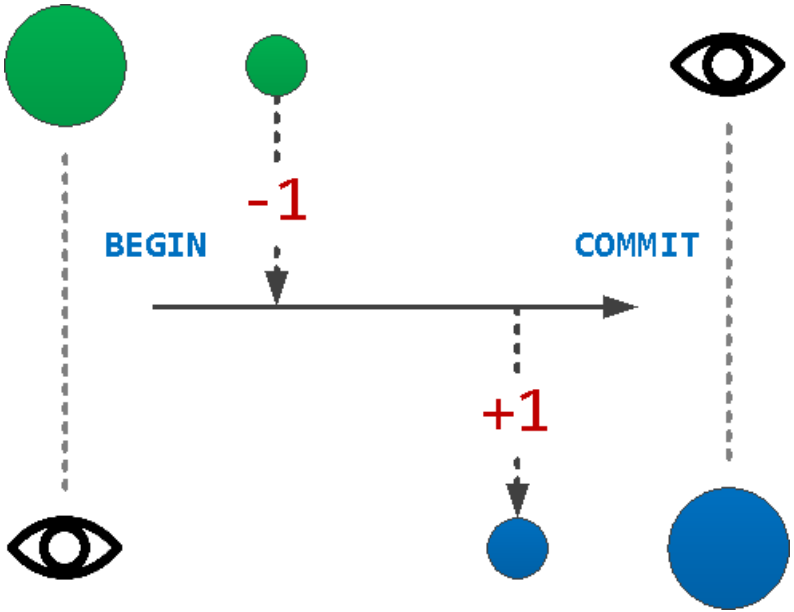
```
BEGIN;
CREATE TABLE x AS SELECT 1 i;
SAVEPOINT sp;    -- сохранились
INSERT INTO x VALUES(2);
ROLLBACK TO sp; -- восстановились
COMMIT;
```

```
TABLE x;
-- i : 1
```

Consistency

согласованность

Consistency



Consistency

```
BEGIN;
UPDATE account SET balance = balance - 1000
  WHERE owner = 'Alice';

-- ... никого, кроме самой транзакции, не должно интересовать состояние внутри

UPDATE account SET balance = balance + 1000
  WHERE owner = 'Bob';
COMMIT;
```

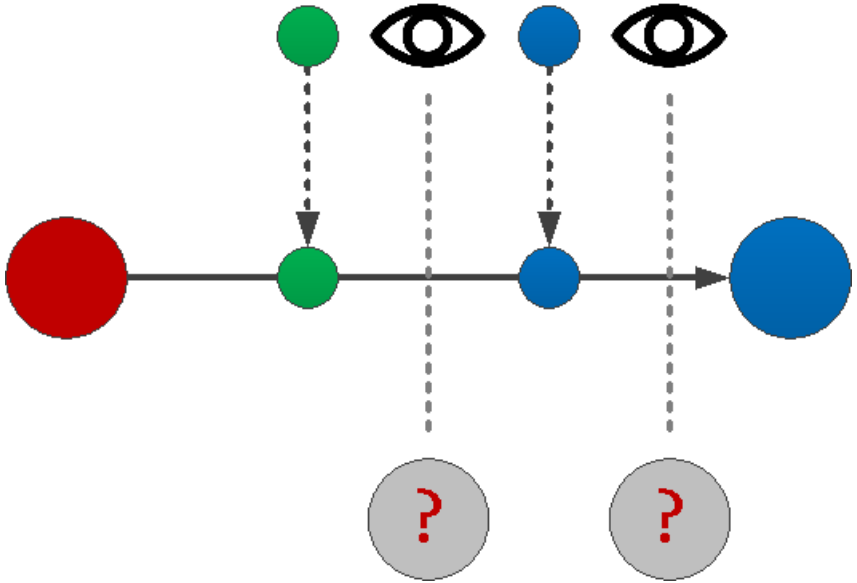
Consistency

```
BEGIN;
BEGIN READ WRITE; -- для опытных гуру

BEGIN READ ONLY;  -- для безопасности базы
CREATE TABLE x AS SELECT 1 i;
ERROR:  cannot execute CREATE TABLE AS in a read-only transaction
```

Isolation

ИЗОЛЯЦИЯ

Isolation



Isolation

MVCC

Multiversion Concurrency Control

«писатели» не блокируют «читателей» и наоборот

у каждой транзакции есть «номер» **xid** (32-bit)

```
SELECT txid_current();
-- 29793191790
```

Isolation

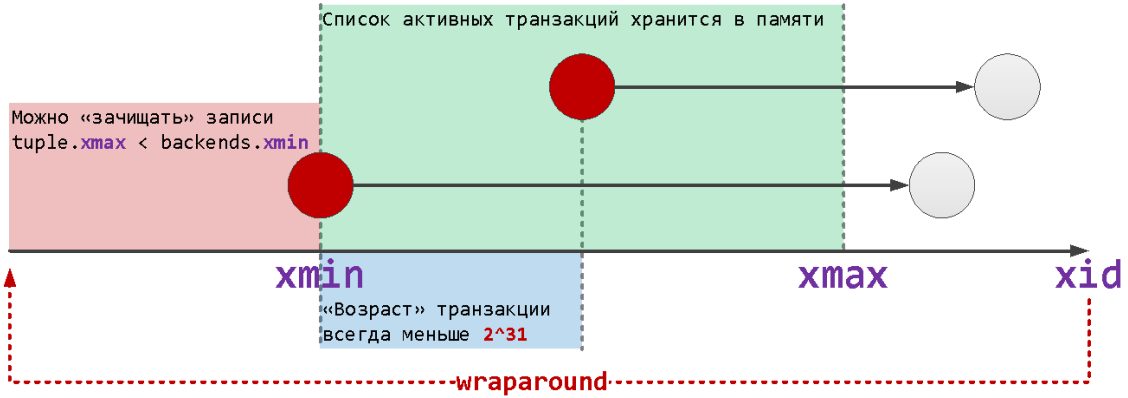
MVCC

Multiversion Concurrency Control

поля **xmin**/**xmax** в заголовке каждой версии каждой записи

ID транзакции, с/до которой версия «видна» другим

https://postgrespro.ru/docs/postgresql/15/storage-page-layout#STORAGE-TUPLE-LAYOUT



Isolation

MVCC

Данные необходимо «подчищать»

10 **UPDATE** – 10 версий записи

VACUUM / **autovacuum**

длительная транзакция – зло!

не дает вычистить устаревшие версии

https://postgrespro.ru/docs/postgresql/15/routine-vacuuming

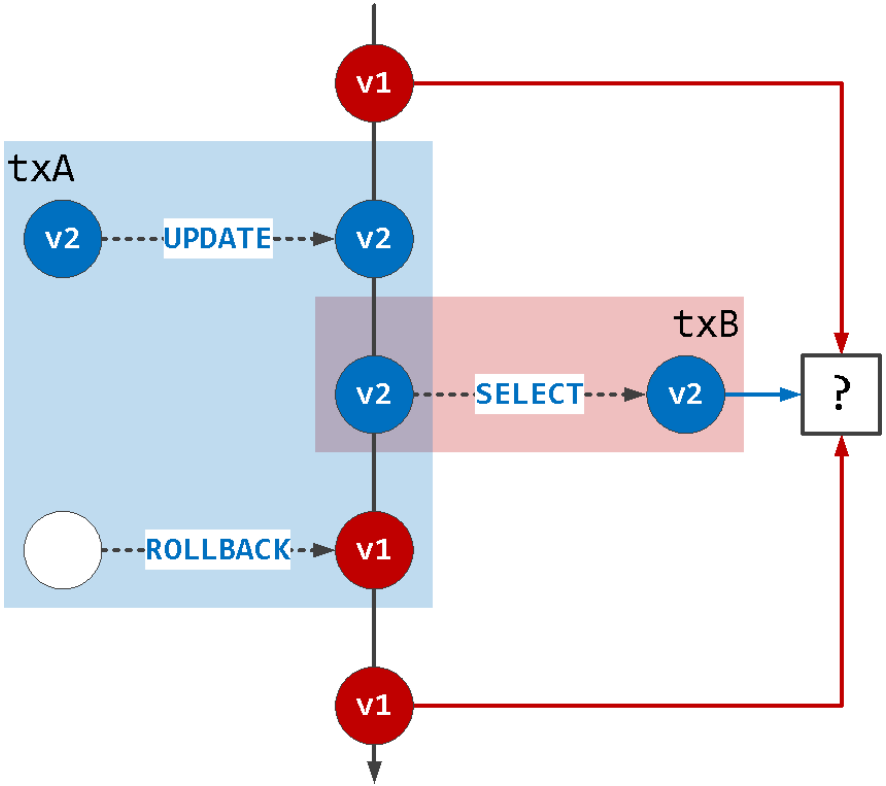
Isolation уровни изоляции

Уровень изоляции / Феномен чтения	«Грязное» чтение (dirty read)	Неповторяемое чтение (nonrepeatable read)	Фантомное чтение (phantom read)	Аномалия сериализации (serialization anomaly)
READ UNCOMMITTED [-] (чтение незафиксированных данных)	возможно но не в PG	невозможно	невозможно	невозможно
READ COMMITTED [d] (чтение зафиксированных данных)	невозможно	невозможно	невозможно	невозможно
REPEATABLE READ (повторяемое чтение)	невозможно	невозможно	возможно но не в PG	невозможно
SERIALIZABLE (сериализуемость)	невозможно	невозможно	невозможно	невозможно

https://postgrespro.ru/docs/postgresql/15/transaction-iso

* феномен чтения – отличия при параллельном и изолированном выполнении транзакций

Isolation dirty read



Isolation

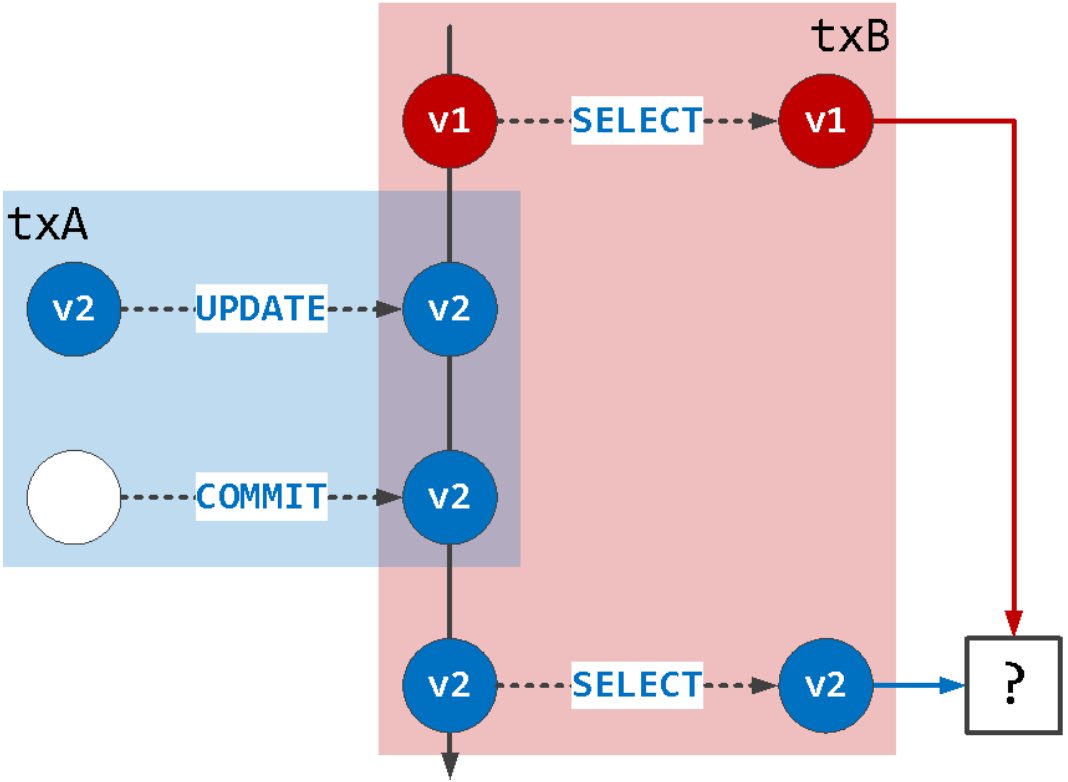
dirty read

```
INSERT INTO x(pk, val) VALUES(1, 1);
BEGIN;
UPDATE x SET val = val + 1 WHERE pk = 1;
...
...
ROLLBACK; -- транзакция отменена
```

```
SELECT val FROM x WHERE pk = 1;
-- val : 2
-- полученное значение отсутствует в БД
```

* прочитали запись а в базе такой «не было и нет»!

Isolation nonrepeatable read



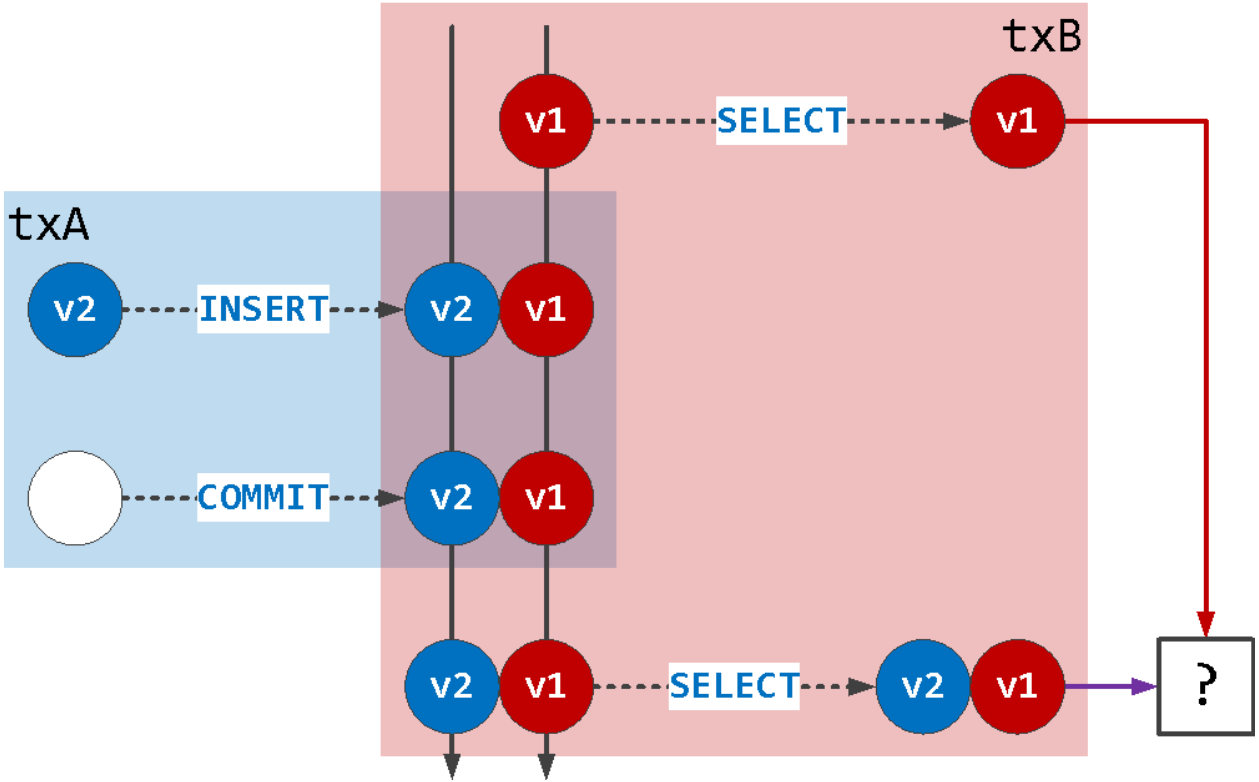
Isolation nonrepeatable read

```
INSERT INTO x(pk, val) VALUES(1, 1);
BEGIN;
...
...
UPDATE x SET val = val + 1 WHERE pk = 1;
COMMIT;
```

```
SELECT val FROM x WHERE pk = 1;
-- val : 1


SELECT val FROM x WHERE pk = 1;
-- val : 2
-- полученное значение не совпадает
```

* перечитали «ту же» запись – а она другая!

Isolation phantom read



Isolation

phantom read

```
INSERT INTO x(pk, val) VALUES(1, 1);
BEGIN;
...
...
INSERT INTO x(pk, val) VALUES(2, 2);
COMMIT;
```

```
SELECT count(*) FROM x;
-- count : 1


SELECT count(*) FROM x;
-- count : 2
-- полученный набор не совпадает
```

* перечитали «тот же» запрос – а набор записей другой!

Isolation

READ UNCOMMITTED – отсутствует в PostgreSQL

READ COMMITTED ↑ лучше производительность

REPEATABLE READ

SERIALIZABLE ↓ «корректнее» данные

Isolation

```
BEGIN [ WORK | TRANSACTION ] [ режим_транзакции [, ...] ]

Где режим_транзакции может быть следующим:

    ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
    READ WRITE | READ ONLY
    [ NOT ] DEFERRABLE
```

https://postgrespro.ru/docs/postgresql/15/sql-begin

```
SET TRANSACTION режим_транзакции [, ...]
  -- сменить для текущей транзакции
SET SESSION CHARACTERISTICS AS TRANSACTION режим_транзакции [, ...]
  -- задать по умолчанию для текущей сессии
```

https://postgrespro.ru/docs/postgresql/15/sql-set-transaction

Isolation

Независимо от уровня изоляции

видим изменения, внесенные этой же транзакцией

при изменениях всех поиск записей идет по правилам **SELECT**

UPDATE

DELETE

SELECT FOR SHARE/UPDATE

Isolation READ COMMITTED

Видим

результаты транзакций, завершившихся к началу запроса

Не видим

результаты незавершенных транзакций

записи, измененные во время работы запроса

https://postgrespro.ru/docs/postgresql/15/transaction-iso#XACT-READ-COMMITTED

Isolation READ COMMITTED

Обработка конфликта изменений записи

ждем завершения блокирующей транзакции

при **ROLLBACK** – применяем операцию к исходной версии записи

при **COMMIT** после **DELETE** – пропускаем

при **COMMIT** после **UPDATE**

проверяем **WHERE**-условие для новой версии

применяем операцию к ней, если условие еще выполняется

Isolation

READ COMMITTED

Проблемы

два последовательных **SELECT** могут дать разный результат

видит записи транзакций, закончившихся между стартами первого и второго

UPDATE/**DELETE** видит нецелостное состояние базы

результат конкурирующих **UPDATE** только на изменяемых записях

неатомарная проверка условий на записи

Isolation READ COMMITTED

Проблемы

DELETE/UPDATE «ничего не делает»

```
CREATE TABLE x(pk, val) AS
  VALUES(1, 1), (2, 2);
BEGIN;
UPDATE x SET val = val + 1;
...
...
COMMIT;
```

```
DELETE FROM x WHERE val = 2;
-- ждем заблокированную запись (2, 2)
-- (2, 2) превратилась в (2, 3)
-- теперь условие не выполняется
```

Isolation REPEATABLE READ

Видим

результаты транзакций, завершившихся к началу транзакции

точнее, первой не-управляющей команды внутри нее

Не видим

результаты незавершенных транзакций

записи, измененные во время работы транзакции

https://postgrespro.ru/docs/postgresql/15/transaction-iso#XACT-REPEATABLE-READ

Isolation REPEATABLE READ

Обработка конфликта изменений записи

ждем завершения блокирующей транзакции

при **ROLLBACK** – применяем операцию к исходной версии записи

при **COMMIT** после **DELETE**/**UPDATE** – откатываем с ошибкой

```
ERROR:  could not serialize access due to concurrent update
```

Isolation REPEATABLE READ

Проблемы

надо быть готовым повторить транзакцию

```
CREATE TABLE x(pk, val) AS
  VALUES(1, 1), (2, 2);
BEGIN ISOLATION LEVEL REPEATABLE READ;
UPDATE x SET val = val + 1;
...
COMMIT;
```

```
BEGIN ISOLATION LEVEL REPEATABLE READ;
DELETE FROM x WHERE val = 2;
ERROR:  could not serialize access due to concurrent update
```

Isolation REPEATABLE READ

Проблемы

могут выполниться принципиально несериализуемые транзакции

```
CREATE TABLE x(pk, val) AS
  VALUES(1, 1), (2, 2);
BEGIN ISOLATION LEVEL REPEATABLE READ;
INSERT INTO x(pk, val)
  SELECT 1, sum(val) FROM x WHERE pk = 2;
...
COMMIT;
-- +(1, 2)
```

```
BEGIN ISOLATION LEVEL REPEATABLE READ;
INSERT INTO x(pk, val)
  SELECT 2, sum(val) FROM x WHERE pk = 1;
COMMIT;
-- +(2, 1)
```

Isolation SERIALIZABLE

Все как при **REPEATABLE READ**

\+ контроль сериализуемости

https://postgrespro.ru/docs/postgresql/15/transaction-iso#XACT-SERIALIZABLE

Isolation

SERIALIZABLE

Проблемы

надо быть готовым повторить транзакцию

```
CREATE TABLE x(pk, val) AS
  VALUES(1, 1), (2, 2);
BEGIN ISOLATION LEVEL SERIALIZABLE;
INSERT INTO x(pk, val)
  SELECT 1, sum(val) FROM x WHERE pk = 2;
...
COMMIT;
-- +(1, 2)
```

```
BEGIN ISOLATION LEVEL SERIALIZABLE;
INSERT INTO x(pk, val)
  SELECT 2, sum(val) FROM x WHERE pk = 1;
COMMIT;
ERROR:  could not serialize access due to
read/write dependencies among transactions
```

Isolation

снимки данных

```
BEGIN ISOLATION LEVEL REPEATABLE READ;
SELECT pg_export_snapshot();
-- 00000003-0000001B-1
```

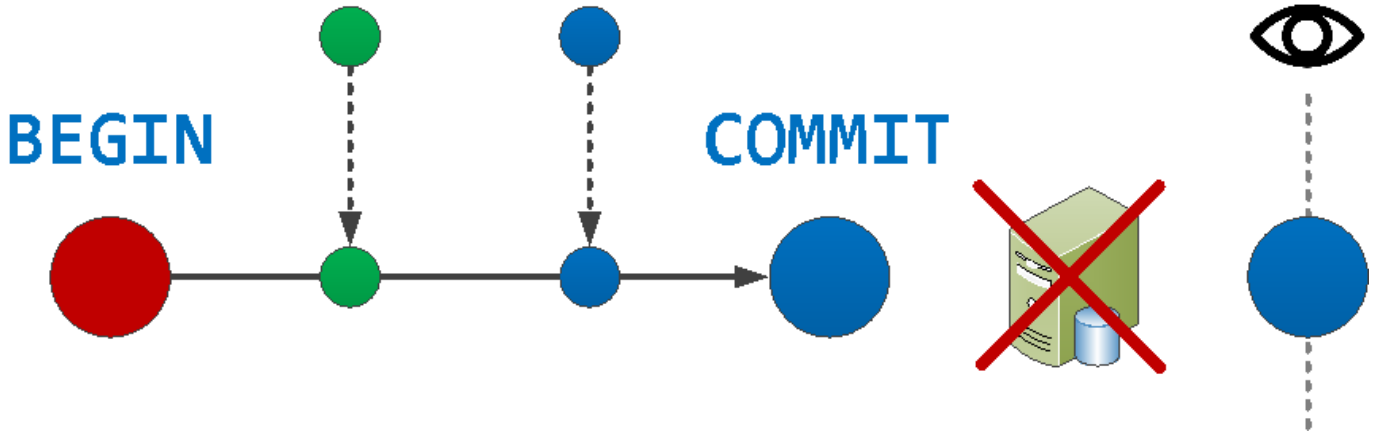
```
BEGIN ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION
 SNAPSHOT '00000003-0000001B-1';
```

* pg_dump так снимает консистентный бэкап параллельно в нескольких сессиях

Durability

устойчивость

Durability



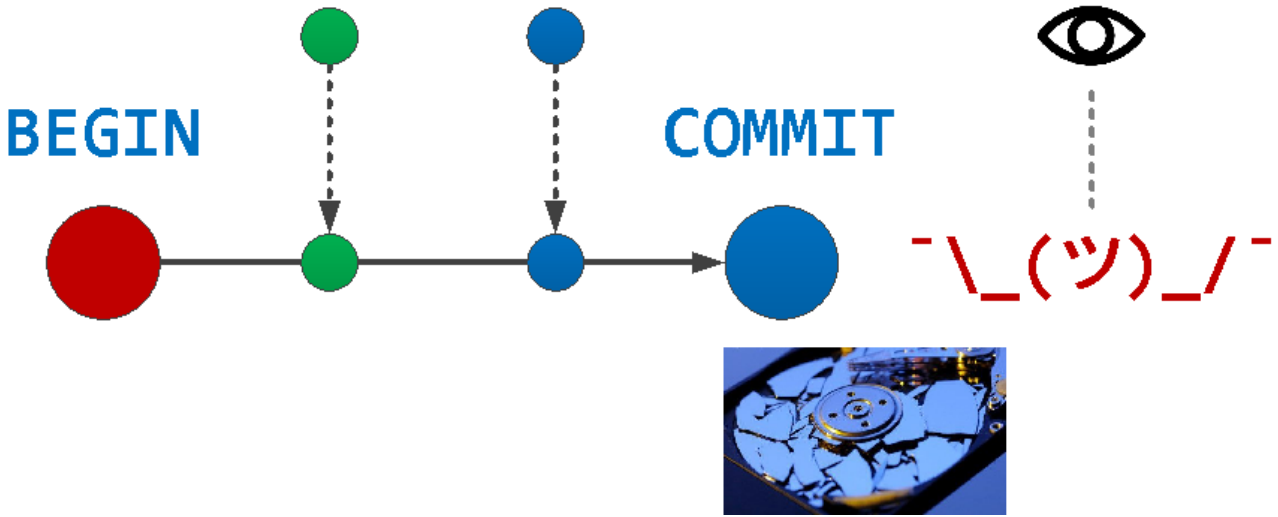
https://postgrespro.ru/docs/postgresql/15/wal

Durability

контрольные суммы

```
SHOW data_checksums; -- default: OFF
```

https://postgrespro.ru/docs/postgresql/15/checksums



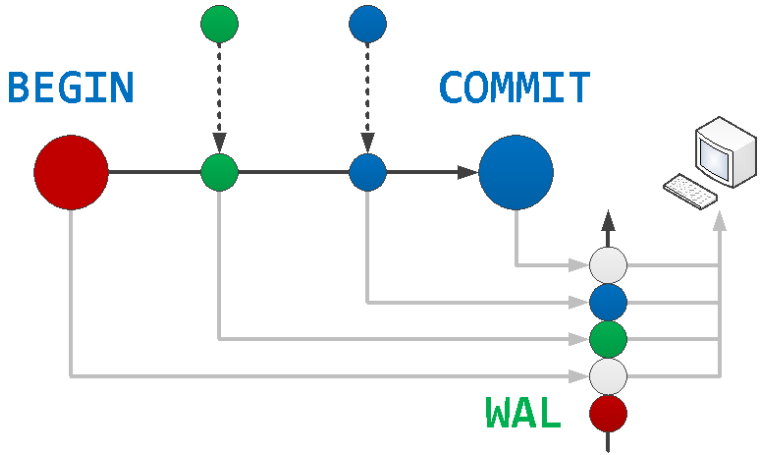
* только для всего сервера целиком, требует перезапуск

Durability журнал предзаписи (WAL)

Write-Ahead Log

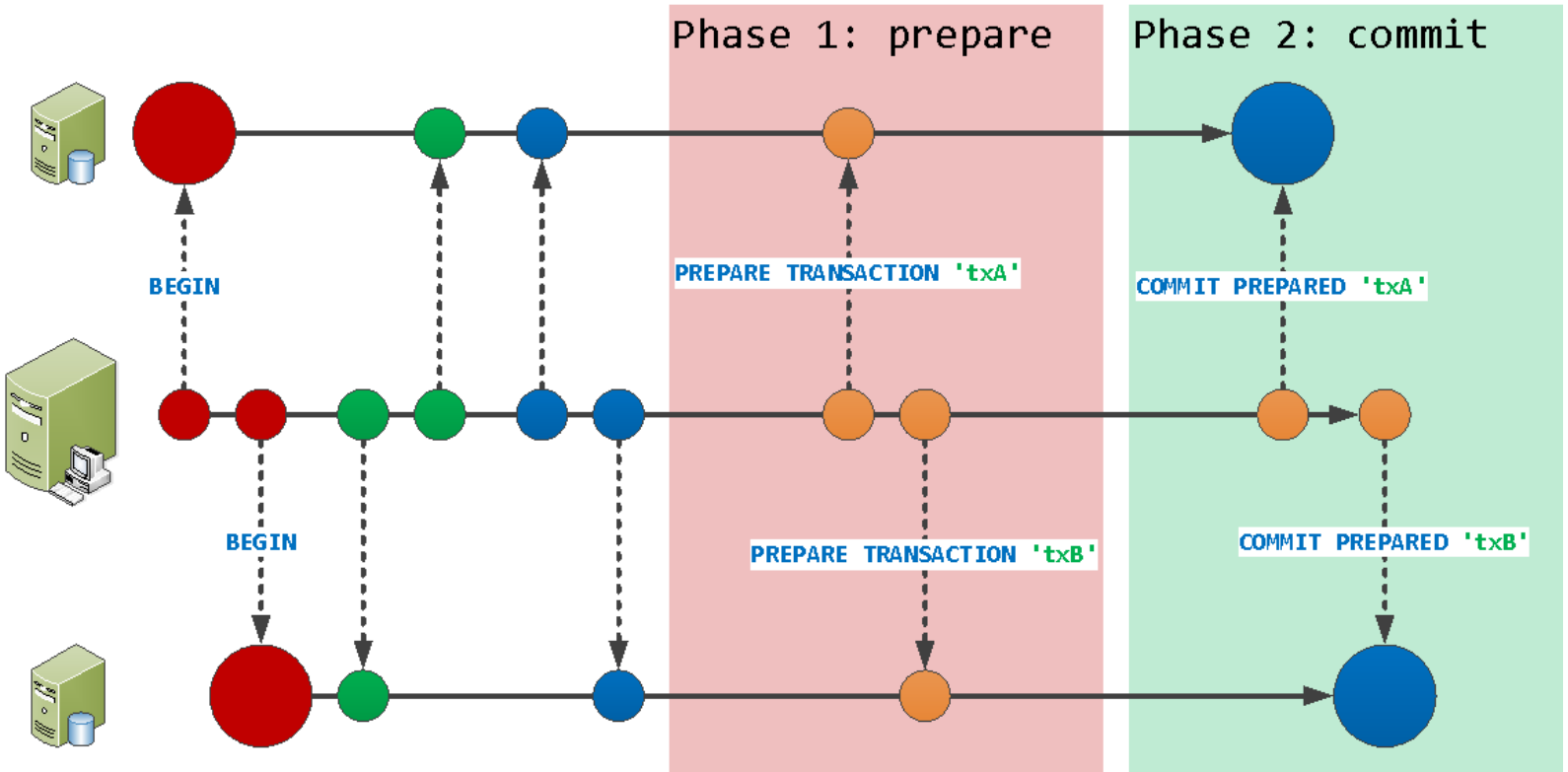
сначала изменения пишем в журнал, потом – в «тело» базы

https://postgrespro.ru/docs/postgresql/15/wal-intro



2PC

двухфазный коммит



2PC

PREPARE TRANSACTION ***id*** – подготовить транзакцию для 2PC

https://postgrespro.ru/docs/postgresql/15/sql-prepare-transaction

COMMIT PREPARED ***id*** – зафиксировать 2PC-транзакцию

https://postgrespro.ru/docs/postgresql/15/sql-commit-prepared

ROLLBACK PREPARED ***id*** – отменить 2PC-транзакцию

https://postgrespro.ru/docs/postgresql/15/sql-rollback-prepared

2PC

```
BEGIN;
...
INSERT INTO x(pk, val) VALUES(1, 1);
...
PREPARE TRANSACTION 'txA';
-- ждем "ok" от всех
COMMIT PREPARED 'txA';
```

```
BEGIN;
...
UPDATE y SET val = val + 1 WHERE pk = 1;
...
PREPARE TRANSACTION 'txB';
-- ждем "ok" от всех
COMMIT PREPARED 'txB';
```

```
-- "зависшие" подготовленные транзакции требуют отслеживания
SELECT * FROM pg_prepared_xacts;
```

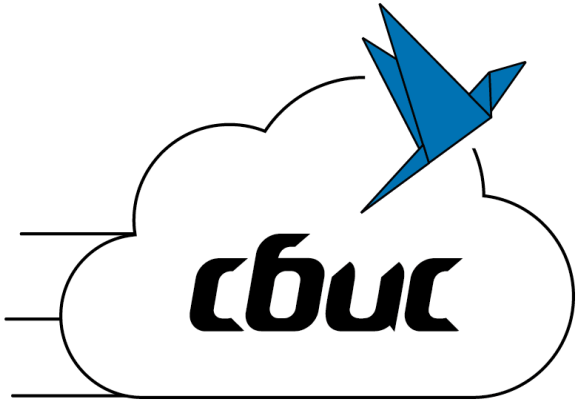
https://postgrespro.ru/docs/postgresql/15/view-pg-prepared-xacts

Егор Рогов, серия «WAL в PostgreSQL»

Буферный кэш	https://habr.com/ru/companies/postgrespro/articles/458186/
Журнал предзаписи	https://habr.com/ru/companies/postgrespro/articles/459250/
Контрольная точка	https://habr.com/ru/companies/postgrespro/articles/460423/
Настройка журнала	https://habr.com/ru/companies/postgrespro/articles/461523/

Егор Рогов, серия «MVCC в PostgreSQL»

Изоляция транзакций	https://habr.com/ru/companies/postgrespro/articles/442804/
Хранение данных	https://habr.com/ru/companies/postgrespro/articles/444536/
Версии строк	https://habr.com/ru/companies/postgrespro/articles/445820/
Снимки данных	https://habr.com/ru/companies/postgrespro/articles/446652/
Очистка на странице (HOT)	https://habr.com/ru/companies/postgrespro/articles/449704/
Очистка таблицы (VACUUM)	https://habr.com/ru/companies/postgrespro/articles/452320/
Автоматическая очистка	https://habr.com/ru/companies/postgrespro/articles/452762/
«Заморозка» данных	https://habr.com/ru/companies/postgrespro/articles/455590/



Спасибо за внимание!

Боровиков Кирилл

kilor@tensor.ru / https://n.sbis.ru/explain

sbis.ru / tensor.ru