

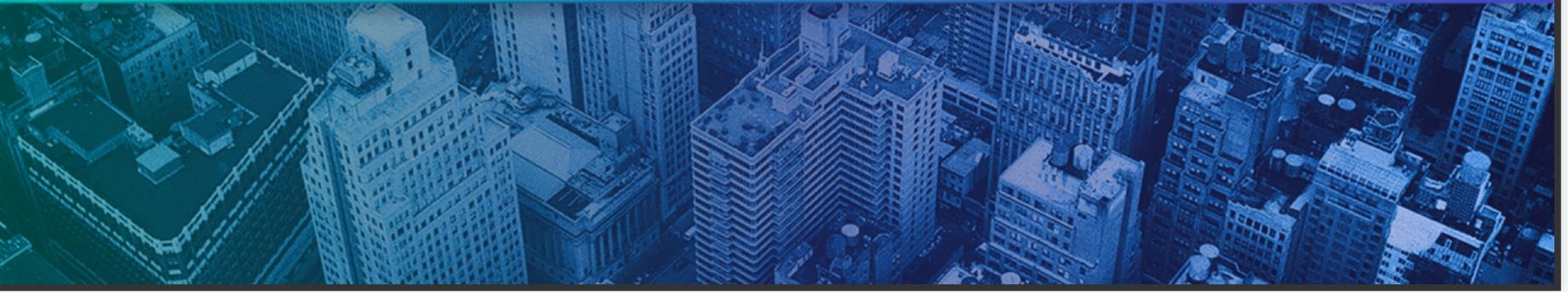


# Онлайн-образование



# Меня хорошо видно && слышно?

Ставьте  , если все хорошо  
Напишите в чат, если есть проблемы



НЕ ЗАБЫТЬ ВКЛЮЧИТЬ  
ЗАПИСЬ!!!

# Postgres. Транзакции и блокировки.

# Сегодня мы разберем...

- Уровни изоляции транзакций в postgres
- Реализация MVCC в постгрес
-

# Уровни изоляции в Postgres

- Read Committed
  - неповторяющееся чтение
  - фантомное чтение
- Repeatable Read
  - фантомное чтение (по стандарту)
- Serializable

# MVCC. Атрибуты записи.

- Записи не меняются и не удаляются
- У записи есть атрибуты- xmin, xmax, cmin, cmax
  - Xmin – ид транзакции, создавшей запись
  - Xmax – ид транзакции, удалившей запись
  - Cmin - порядковый номер команды в транзакции, добавившей запись
  - Cmax - номер команды в транзакции, удалившей запись

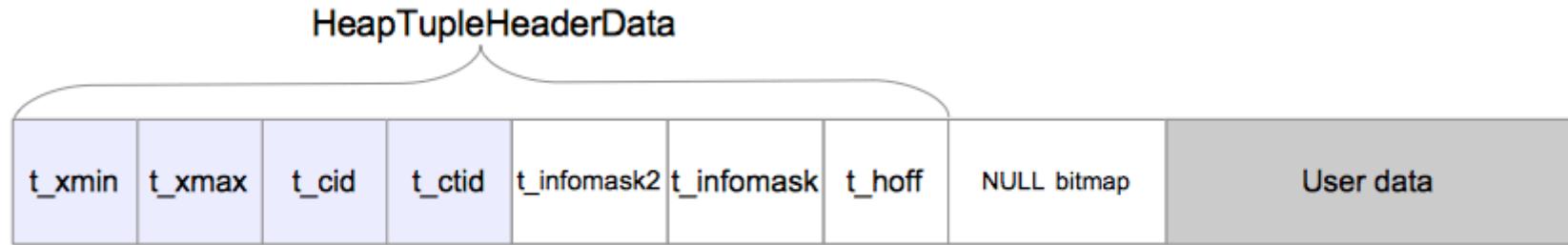
# Дополнительные атрибуты строки

- infomask содержит ряд битов, определяющих свойства данной версии.
  - xmin\_committed, xmin\_aborted
  - xmax\_committed, xmax\_aborted
- ctid является ссылкой на следующую, более новую, версию той же строки. У самой новой, актуальной, версии строки ctid ссылается на саму эту версию.
  - Номера ctid имеют вид (x,y): здесь x — номер страницы, у — порядковый номер указателя в массиве.
- `select xmin, xmax, cmin, cmax, ctid from tbl;`

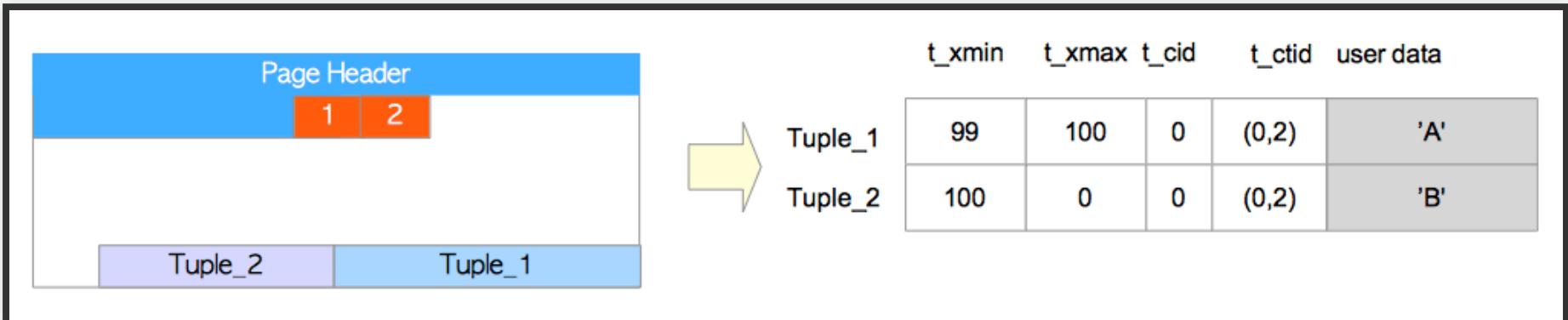
# Заморозка и удаление старых записей

- Для удаления записей существует **autovacuum**
- Все транзакции в системе имеют последовательные номера **xid** (32 бита)
- Счетчик транзакций лимитирован, при достижении лимита начинает с 0
  - **vacuum** "замораживает" старые транзакции, что решает проблему обнуления счетчика

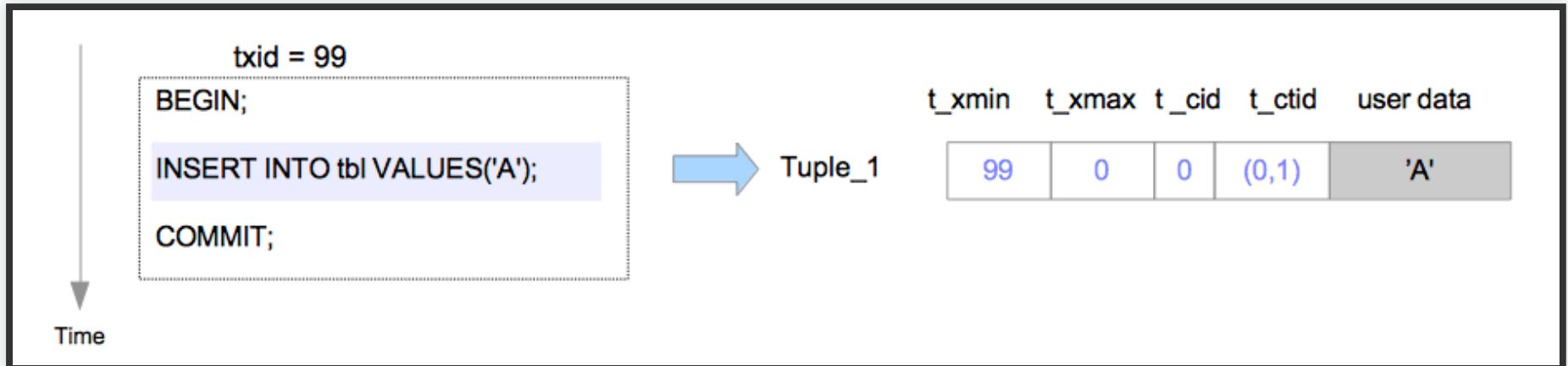
# Формат записм (tuple)



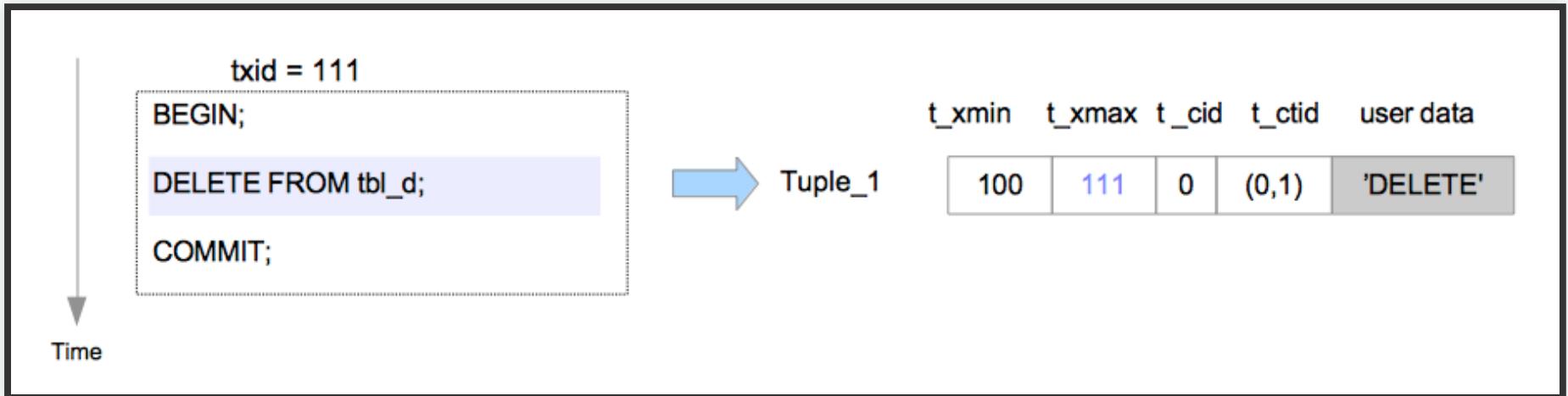
# Хранение записей



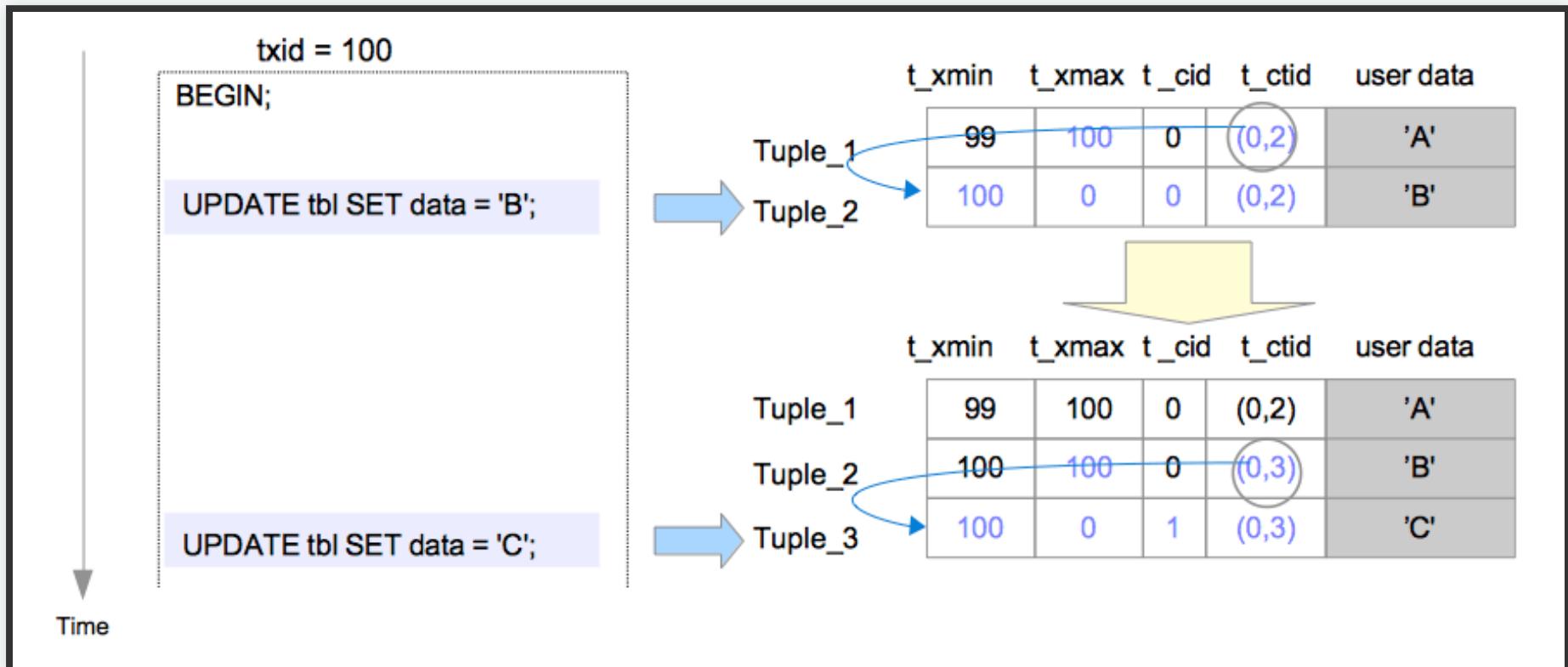
# Insert



# Delete



# Update



# Заморозка

- Замороженная версия строки считается старше любых обычных данных
- всегда видна во всех снимках данных.
- не требуется смотреть на номер транзакции xmin
- Полная заморозка
  - VACUUM FREEZE
  - VACUUM FULL
  - vacuumdb --all --freeze

# Очистка

```
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ имя_таблицы ]
```

```
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ имя_таблицы ]
```

- `SELECT * FROM pg_stat_progress_vacuum \gx`

# Задачи VACUUM

- Для высвобождения или повторного использования дискового пространства, занятого изменёнными или удалёнными строками.
- Для обновления статистики по данным, используемой планировщиком запросов Postgres Pro.
- Для обновления карты видимости, которая ускоряет сканирование только индекса.
- Для предотвращения потери очень старых данных из-за зацикливания идентификаторов транзакций или мультитранзакций.

# Автоочистка

- автоматически запускает VACUUM и ANALYZE
- очистка требуется, если
  - $\text{pg\_stat\_all\_tables.n\_dead\_tup} \geq \text{autovacuum\_vacuum\_threshold} + \text{autovacuum\_vacuum\_scale\_factor} * \text{pg\_class.reltuples}$
- autovacuum\_enabled
  - `ALTER TABLE tbl SET (autovacuum_enabled = off);`

# Настройки автоочистки

```
track_counts = on
autovacuum_enabled = on

log_autovacuum_min_duration = 0
autovacuum_max_workers = 10
autovacuum_naptime = 15s
autovacuum_vacuum_threshold = 25
autovacuum_vacuum_scale_factor = 0.1
autovacuum_vacuum_cost_delay = 10
autovacuum_vacuum_cost_limit = 1000

autovacuum_analyze_scale_factor = 0.02; -- 2%
autovacuum_analyze_threshold = 0;
```

# Параметры управления заморозкой

- подробнее
- SHOW vacuum\_freeze\_min\_age;
  - определяет минимальный возраст транзакции xmin, разрешенной для заморозки строки
  - пропускает страницы вне таблицы видимости
- vacuum\_freeze\_table\_age
  - pg\_class.relfrozenxid < vacuum\_freeze\_table\_age
  - проходит по всем страницам таблица для заморозки
- autovacuum\_freeze\_max\_age

# Мониторинг

- pg\_stat\_activity
- pg\_database
- pg\_class
- CREATE EXTENSION pg\_visibility;
  - pg\_visibility\_map
- SELECT txid\_current();

# Snapshots

- одновременно существуют несколько версий отной записи
- каждая транзакция видит только **одну** или не одной
- `SELECT txid_current_snapshot();`
  - `snapshot.xmin`, `snapshot xmax` и `snapshot.xip`
- возможен **экспорт** текущего снэпшот

```
SELECT pg_export_snapshot();
BEGIN ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION SNAPSHOT '00000004-00000E7B-1';
```

# XACT (CLOG)

- это файлы в каталоге PGDATA/pg\_xact
- кэшируется в памяти
- массив транзакций с битами состояний
  - committed
  - aborted

# Вложенные транзакции

- реализация SAVEPOINT
- Собственный номер и статус в XACT
- каталог \$PGDATA/pg\_subtrans/
- кэшируется в памяти
- обработка исключений в PL/pgSQL (EXCEPTION)
- режим psql ON\_ERROR\_ROLLBACK = on/interactive

# Индексы

- при обычном обновлении в индексе создаются ссылки на все версии строки, присутствующие в табличной странице.
- При любом обновлении строки надо изменять индекс
- В индексе накапливаются ссылки на неактуальные версии
- Все сложности умножаются на количество индексов, построенных по таблице
- Heap-Only Tuple Update (HOT)
  - обновление записи, если изменяется столбец, который не входит в индекс

# Мониторинг блокировок

- представление pg\_locks:
  - locktype – тип блокируемого ресурса
  - mode – режим блокировки
- Вывод сообщений в журнал сервера
  - параметр log\_lock\_waits:
  - выводит сообщение об ожидании дольше deadlock\_timeout
- функция pg\_blocking\_pids + представление pg\_stat\_activity

# Журнал упреждающей записи (WAL)

- Основная задача
  - возможность восстановления согласованности данных после сбоя
- Механизм
  - при изменении данных действие записывается в журнал
  - журнальная запись попадает на диск раньше измененных данных
  - восстановление после сбоя — повторное выполнение потерянных операций с помощью журнальных записей

# Что фиксируется в журнале

- Изменение любых страниц (таблиц и индексов) в буферном кэше
  - в том числе страницы таблиц и индексов
  - кроме нежурналируемых и временных таблиц
- Фиксация и отмена транзакций
  - буферы XACT
- Файловые операции
  - создание и удаление файлов создание и удаление каталогов

# Устройство журнала

- в памяти
  - кольцевой буферный кэш
  - wal\_buffers = -1 (1/32 буферного кэша)
- на диске
  - \$PGDATA/pg\_wal
  - файлы по 16 мб

# ПОЛЕЗНЫЕ ФУНКЦИИ

- `SELECT pg_current_wal_insert_lsn();`
  - Позиция для записи
- `SELECT pg_current_wal_lsn()`
  - последняя записанная на диск позиция
- `SELECT pg_walfile_name(pg_current_wal_insert_lsn());`
- `SELECT * FROM pg_ls_waldir() LIMIT 10;`
- `pg_waldump`

# Настройка журнала

- wal\_level = replica
- minimal
  - восстановление после сбоя
- replica
  - восстановления из бэкапа, репликация
  - операции массовой обработки данных
- logical
  - логическая репликация

# Синхронизация журнала

- Задача: данные должны дойти до энергонезависимого хранилища через многочисленные кэши
- `fsync = on`
- `wal_sync_method`
  - утилита `pg_test_fsync` помогает выбрать оптимальный способ

# Проверки на повреждения данных

- data\_checksums
- ignore\_checksum\_failure = off
- wal\_log\_hints = off (неявно on при контрольных суммах страниц)
- wal\_compression = off

# Синхронная запись

```
synchronous_commit =on  
-- можно увеличить задержку комитта при большом потоке коротких транзакций  
commit_delay = 0  
commit_siblings = 5
```

# Асинхронная запись

- можно потерять зафиксированные изменения

```
# можно менять на уровне транзакции
synchronous_commit = off
wal_writer_delay = 200ms
```

# Ваши вопросы