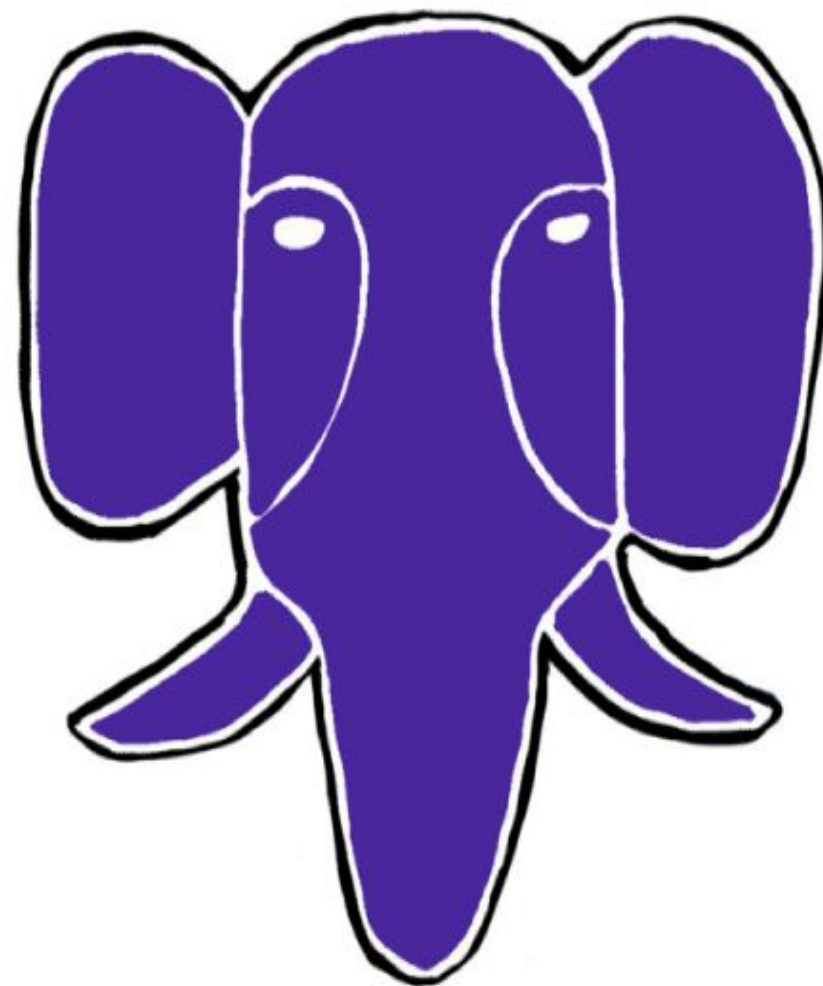


Евгений Аристов

Открытый урок Особенности MVCC PostgreSQL

<https://aristov.tech>





Правила вебинара

<https://aristov.tech>

Задаем вопрос в чат

Можно общаться голосом, для этого поставьте знак ? в чат

Вопросы вижу, отвечу в момент логической паузы

Если остались вопросы, можно написать мне через сайт aristov.tech

<https://aristov.tech>

Маршрут вебинара

<https://aristov.tech>

- ❖ MVCC везде
- ❖ Особенности MVCC PostgreSQL
- ❖ практика
- ❖ Коротко о проекте aristov.tech
- ❖ Розыгрыш скидки 30% на курс
- ❖ Ответ на вопросы

<https://aristov.tech>

Множественная параллельная нагрузка

<https://aristov.tech>

Реляционная теория и SQL позволяет абстрагироваться от конкретной СУБД, но есть одна непростая проблема:

- ❖ как обеспечить **параллельную** работу множества сессий (**concurrency**)
- ❖ которые **модифицируют** данные
- ❖ так чтобы они **не мешали** друг другу
- ❖ ни с точки зрения **чтения** ни с точки зрения **записи**
- ❖ и обеспечивали **целостность** данных - т.н. **consistency**
- ❖ и их **надежность** - т.н. **durability**
- ❖ **помним про ACID**

Почему конкурентный доступ это плохо?

<https://aristov.tech>

- ❖ Достаточно сложные алгоритмы реализации
- ❖ Дорого с точки зрения затрат на ресурсы - процессоры, память, диски

Требования к конкурентному доступу к данным:

- ❖ гарантированная сохранность данных
 - данные всех завершенных транзакций будут сохранены
 - данные всех незавершенных транзакций будут отменены
- ❖ необходимо корректно работать как в нормальном режим, так и в случае аварии

Решение этой задачи:

- ❖ **мультиверсионность** или **MVCC** - multiversioning concurrency control

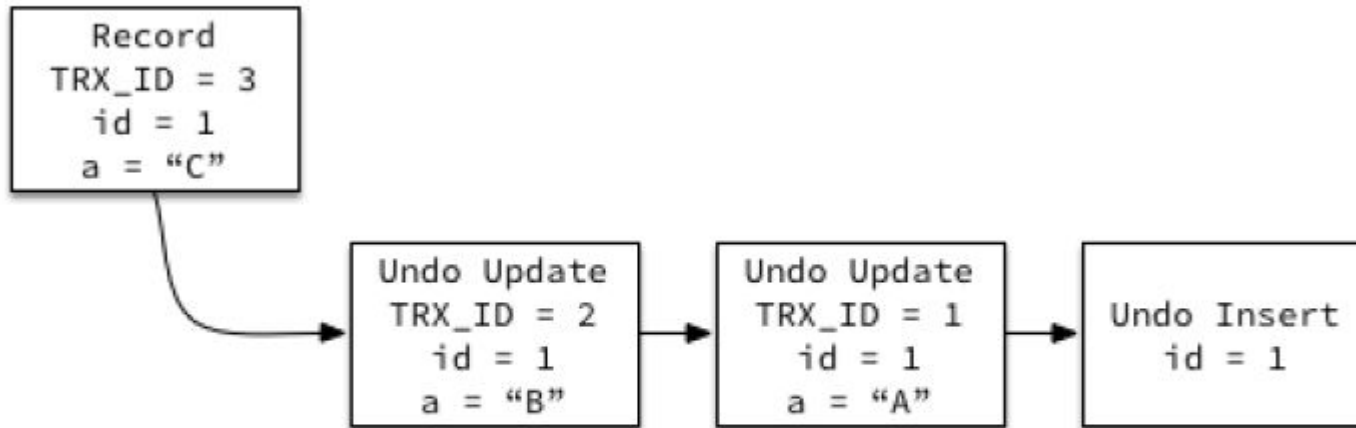
Стандартный механизм для Oracle Database, MySQL InnoDB и др - использование Undo (rollback) segment (сегмент отката).

Особенности реализации:

- ❖ туда вносятся «противодействия». То есть, если в таблицу был внесен insert, то в сегмент отката вносится delete, delete – insert, update – вносится предыдущее значение строки. Блоки (и соответствующие данные undo) помечаются как грязные и переходят в redo log buffer (буфер донакатки изменений после сбоя). В redo журнал записываются не только инструкции, какие изменения стоит внести (redo), но и какие у них противодействия (undo).
- ❖ информация завершенных транзакций потихоньку оттуда вытесняется
- ❖ если rollback, нужно вернуть данные обратно в таблицу - много лишней работы
- ❖ если слишком много данных для вставки, удаления или обновления сегмент может переполниться и произойдет откат транзакции
- ❖ длинные транзакции могут стать проблемой

<https://habr.com/ru/articles/72122/>

Record History



History above represents the following SQL statements:

```
INSERT INTO t (id, a) VALUES (1, "A");  
UPDATE t SET a="B" WHERE id = 1;  
UPDATE t SET a="C" WHERE id = 1;
```


Tuple multiversioning - многоверсионность.

Особенности работы:

- ❖ данные не удаляются в процессе обработки транзакций
- ❖ создаются новые версии записей для операции update
- ❖ что делать со старыми версиями записей?
- ❖ ответ - вакуум, который очищает это место и туда записываются уже следующие транзакции
- ❖ отсюда следует, что нет кластерного индекса (только в виде одноразовой операции)

PostgreSQL MVCC, xmin & xmax, cmin & cmax

<https://aristov.tech>

В каждой таблице есть скрытые колонки

- ❖ **xmin**
 - идентификатор транзакции которая создала данную версию записи
- ❖ **xmax**
 - идентификатор транзакции которая удалила данную версию записи
- ❖ **cmin**
 - порядковый номер команды в транзакции, добавившей запись
- ❖ **cmax**
 - номер команды в транзакции, удалившей запись

PostgreSQL MVCC DML

<https://aristov.tech>

Механизм работы. При CRUD операциях происходит следующее:

- ❖ **Insert**

- добавляется новая запись с **xmin**=txid_current() и **xmax**=0

- ❖ **Delete**

- в старой версии записи **xmax**=txid_current()

- ❖ **Update**

- в старой версии записи **xmax**=txid_current(), то есть делается delete

- добавляется новая запись с **xmin**=txid_current() и **xmax**=0, то есть делается insert

- ❖ Если происходит отмена транзакции

- xmax так и остается с номером пытавшейся изменить его транзакции

- проставляется бит отмены транзакции **xmax_aborted**

Подробнее про версии записей можно почитать тут:

[The Internals of PostgreSQL : Chapter 5 Concurrency Control](#)

[MVCC-3. Версии строк](#)

<https://aristov.tech>

Дополнительные атрибуты строки:

- ❖ **infomask**

- содержит ряд битов, определяющих свойства данной версии.

- ❖ **xmin_committed, xmin_aborted, xmax_committed, xmax_aborted**

- биты отвечающие за коммит или роллбек транзакции и также участвуют при заморозке (freeze)

- ❖ **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, y — порядковый номер в странице

Исходя из архитектуры, отмена транзакции выполняется так же быстро, как и фиксация (просто проставляется бит фиксации или отмены).

Хоть команда и называется ROLLBACK, отката изменений не происходит: все, что транзакция успела изменить в страницах данных, остается без изменений (кроме этих бит).

При обращении к странице будет проверен статус и в версию строки будет установлен бит подсказки **xmax_aborted**. Сам номер **xmax** при этом остается в странице, но смотреть на него уже никто не будет.

Практика

Проект aristov.tech

aristov.tech

Книга по 14 Постгресу (скоро по оптимизации 16 Постгреса) <https://aristov.tech/#orderbook>

Эксклюзивный **курс** по Оптимизации Постгреса 3 поток

<https://aristov.tech/blog/kurs-po-optimizaczii-postgresql-2-0/>

Со всеми **отзывами** без цензуры можно ознакомиться по ссылке

<https://aristov.tech/blog/otzivi-kurs/>

Блог с популярными темами <https://aristov.tech/blog>

ТГ **канал** с новостями блога и проекта https://t.me/aristov_tech

Ютуб канал с интересными видео <https://www.youtube.com/@aristovtech>

Моя **группа** ДБА <https://t.me/dbaristov>

Также за прошедшее время был запущен проект **менторинга** в котором уже участвует 15 лучших экспертов в отрасли ДБА/DevOps и разработки.

Ознакомится с преподавателями и проектом <https://aristov.tech/mentorship>

В направлении крутых **вакансий** уже 8 предложений от партнёров

<https://aristov.tech/blog/vakansii-ot-partnerov-aristov-tech/>

Всегда можно со мной связаться через сайт для консультации, аудита вашего проекта, менторинга, обучения и многого другого

PostgreSQL 16: оптимизируй это

Оглавление

Об авторе	4
1. ПостгресQL 16. Настройка VM, ОС и БД	6
2. Коннектинг к ПостгресQL. Права пользователя	40
3. Настройка файловой системы.....	63
4. Настройка бэкапов и репликации	83
5. Мониторинг, профилирование и логирование	113
6. Тюнинг WAL & shared_buffers	140
7. Vacuum, background writer, work_mem, statistic collector, locks	164
8. Схема данных.....	188
9. Оптимизация запросов	222
10. Обслуживание СУБД. Работа на стенде.....	250
Заключение	274

Розыгрыш книги и скидки на курс
регистрируемся по форме:

<https://forms.gle/9i8eRcSHGLnMXi3B7>

Сам розыгрыш в db-fiddle

<https://www.db-fiddle.com/f/43wbuUzv7YUAcS3aypkvvC/1>

ДЗ

ДЗ

1. Подписаться на канал
2. Подписаться на Ютуб
3. Вступить в мою группу ДБА
4. Посещать открытые уроки
5. Прийти на курс %)
6. Прокачаться с помощью менторинга
7. Устроиться на одну из вакансий 450+

Спасибо за внимание!

Следующий открытый урок 27.04 в 10:00
тему и ссылки для подключения опубликую на сайте в ближайшее время
на ютуб загружу после обработки и опубликую в своём новостном канале

Аристов Евгений