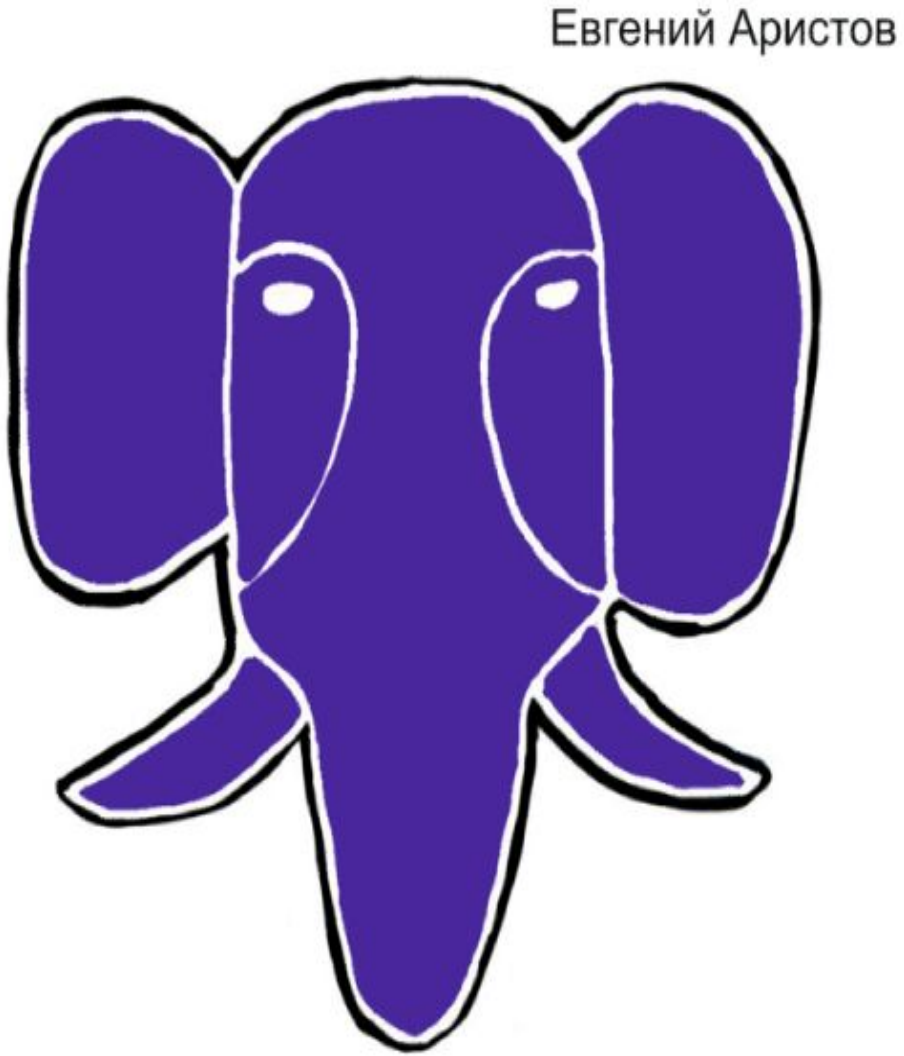
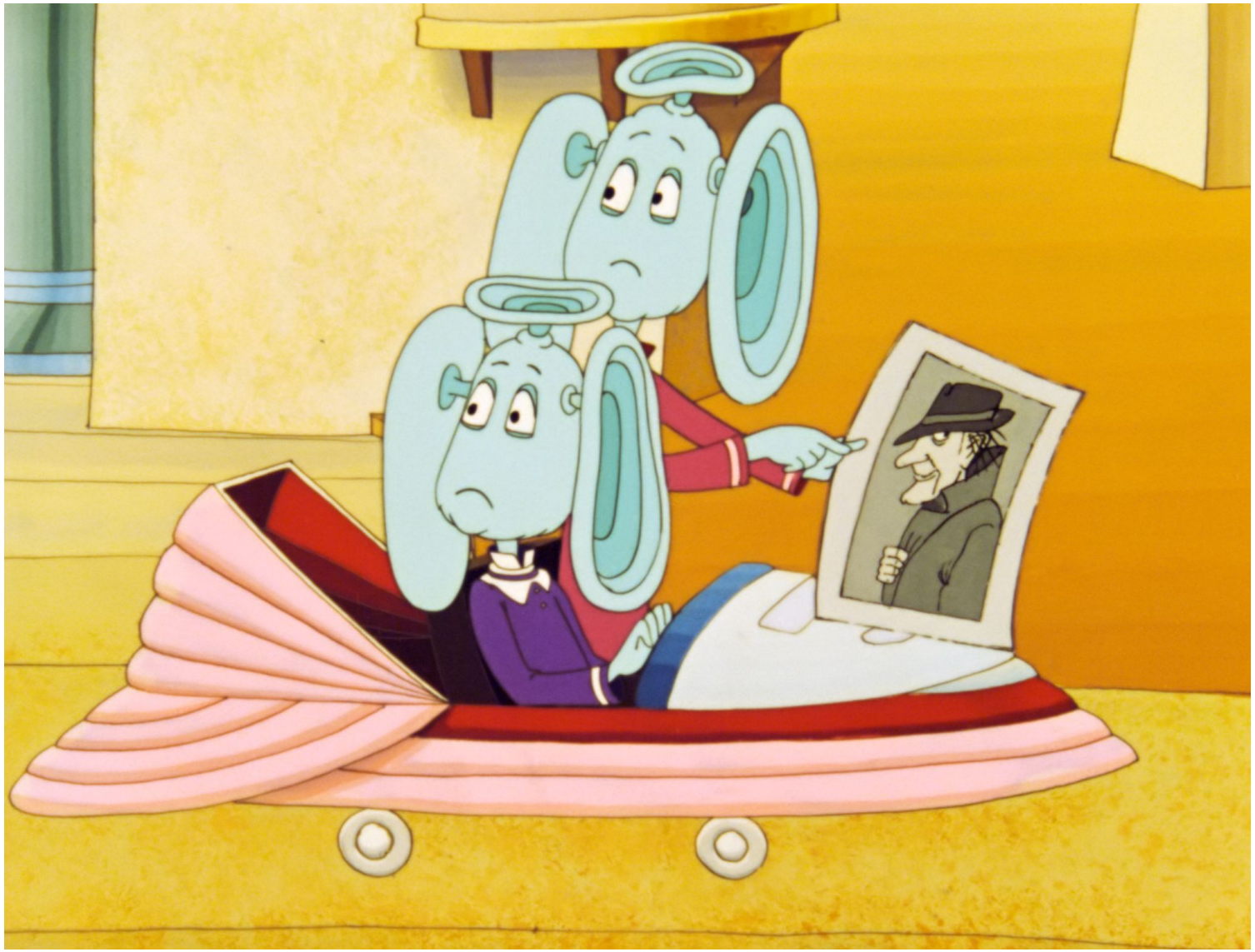
MVCC





Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно их задать на следующем занятии

Маршрут вебинара

- MVCC везде
- Особенности MVCC PostgreSQL
- Практика

https://aristov.tech

Множественная параллельная нагрузка

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?

- 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

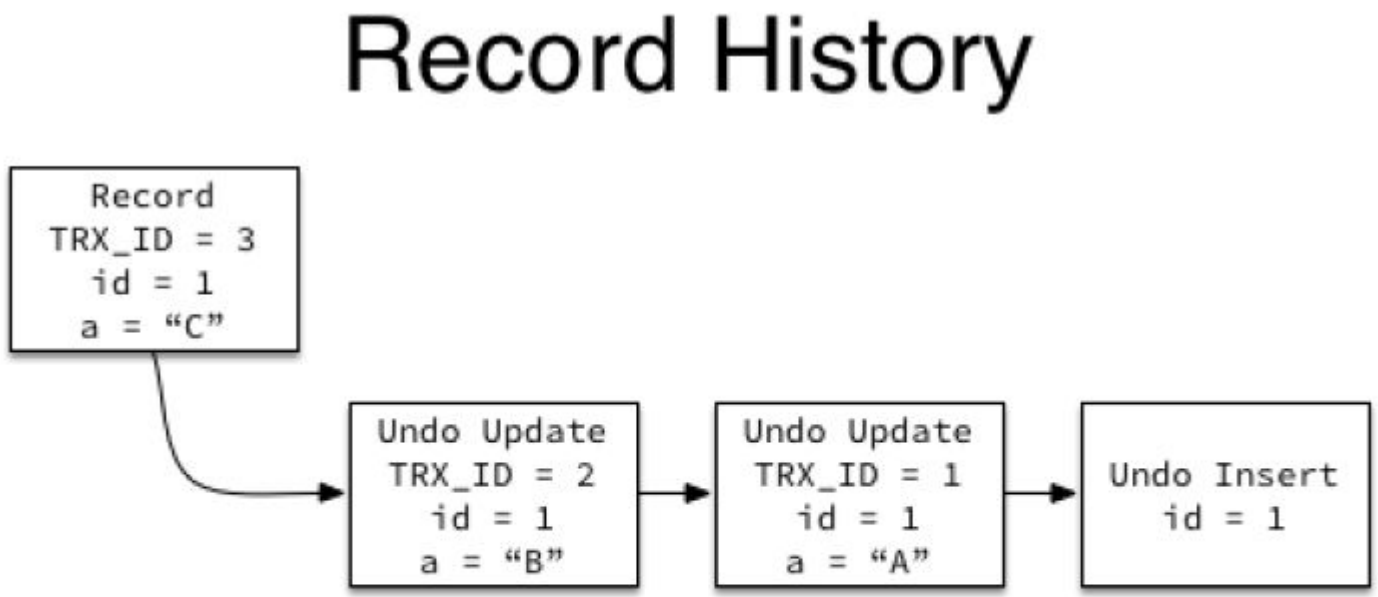
ALL RDBMS MVCC

Стандартный механизм для Oracle Database, MySQL InndoDB 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/

Undo



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;
```

https://aristov.tech

Postgres MVCC

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

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, удаливший запись

PostgreSQL MVCC DML

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

MVCC

Дополнительные атрибуты строки:

- **infomask**
  - содержит ряд битов, определяющих свойства данной версии.
- **xmin_commited, xmin_aborted, xmax_commited, 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

MVCC

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

https://aristov.tech

Итоги

https://aristov.tech

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

https://aristov.tech

Спасибо за внимание!

Когда дальше и куда?
В чате напишу
материалы для бесплатного доступа будут появляться на ютубе

Аристов Евгений