Архитектура

Изоляция и многоверсионность

Темы

Многоверсионность

Снимок данных

Уровни изоляции

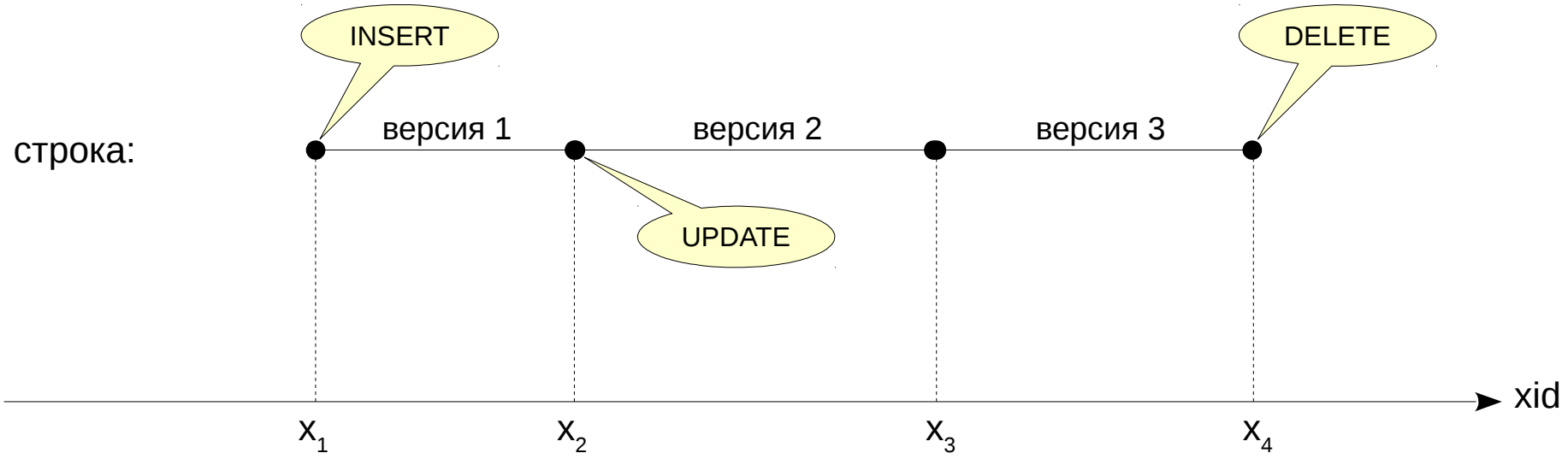
Блокировки

Очистка страниц

Многоверсионность

Наличие нескольких версий одной и той же строки

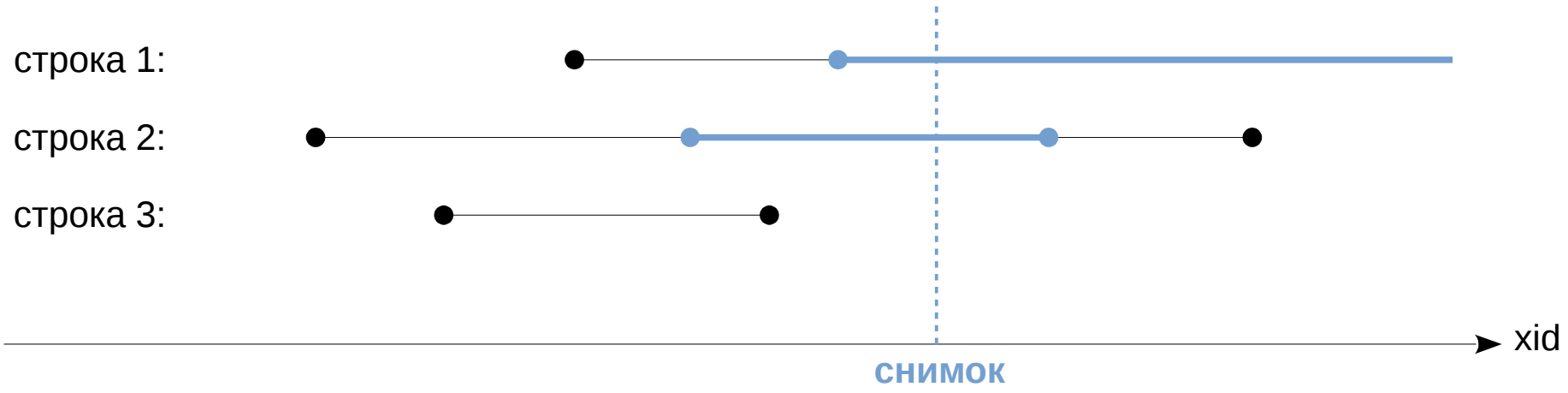
- версии различаются временем действия
- время = номер транзакции (номера выдаются по возрастанию)



Снимок данных

Согласованный срез на определенный момент времени

- номер транзакции — определяет момент времени
- список активных транзакций — чтобы не смотреть на еще не зафиксированные изменения



Блокировки

Блокировки строк

- чтение никогда не блокирует строк
- изменение строки блокирует ее для изменений, но не для чтений

Блокировки таблиц

- запрещают изменение или удаление таблицы, пока с ней идет работа
- запрещают чтение таблицы при перестроении или перемещении
- и т. п.

Время жизни блокировок

- устанавливаются по мере необходимости или вручную
- снимаются автоматически при завершении транзакции

Уровни изоляции

Read uncommitted — *не поддерживается PostgreSQL*

- позволяет читать не зафиксированные данные

Read committed — *используется по умолчанию*

- снимок строится на момент начала оператора
- одинаковые запросы могут каждый раз получать разные данные

Repeatable read

- снимок строится на момент начала транзакции
- транзакция может завершиться ошибкой сериализации

Serializable

- полная изоляция, но дополнительные накладные расходы
- транзакция может завершиться ошибкой сериализации

Очистка

Postgres PROFESSIONAL

Старые версии строк хранятся вместе с актуальными

- со временем размер таблиц и индексов увеличивается

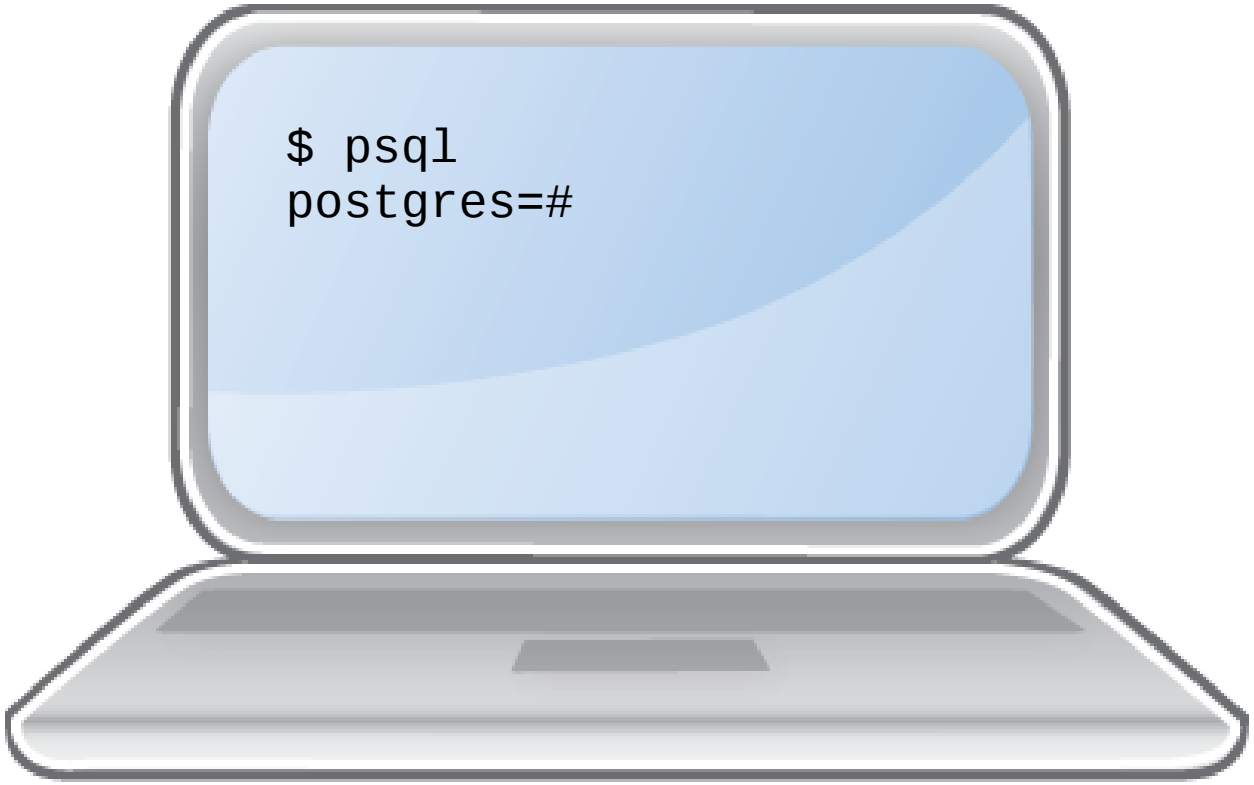
Процесс очистки (vacuum)

- удаляет версии строк, которые уже не нужны
  (то есть не видны ни в одном снимке данных)
- работает параллельно с остальными процессами
- удаленные версии отставляют в файлах данных «дыры»,
  которые затем используются для новых версий строк
- процесс обычно работает автоматически (autovacuum)

Полная очистка

- полностью перестраивает файлы данных, делая их компактными
- блокирует таблицу на время работы

Демонстрация



Итоги

В файлах данных могут храниться несколько версий каждой строки

Транзакции работают со снимком данных — согласованным срезом на определенный момент времени

Писатели не блокируют читателей,
читатели не блокируют никого

Время создания снимка влияет на уровень изоляции

Версии строк накапливаются, поэтому нужна периодическая очистка

Практика

1. Создайте таблицу с одной строкой.
2. Начните первую транзакцию и выполните запрос к таблице.
3. Во втором сеансе удалите строку и зафиксируйте изменения.
4. Сколько строк увидит первая транзакция, выполнив тот же запрос повторно? Проверьте.
5. Завершите первую транзакцию.
6. Повторите все то же самое, но пусть теперь транзакция работает на уровне изоляции repeatable read:

   ```
   BEGIN ISOLATION LEVEL REPEATABLE READ;
   ```

   Объясните отличия.

Авторские права
© Postgres Professional, 2017 год.
Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса
Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Отказ от ответственности
Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или непрямым, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Темы

Многоверсионность

Снимок данных

Уровни изоляции

Блокировки

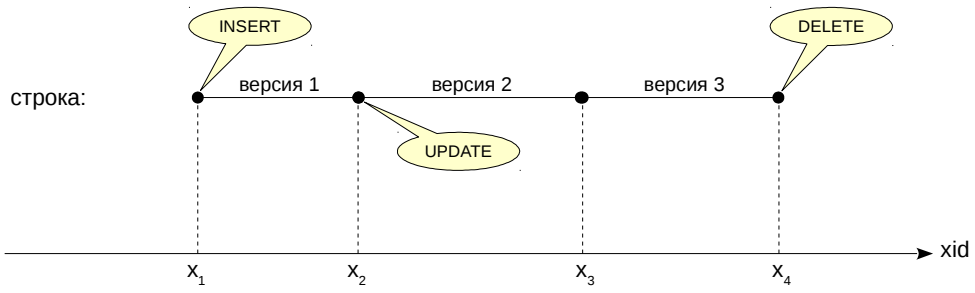
Очистка страниц

Многоверсионность

Наличие нескольких версий одной и той же строки

версии различаются временем действия

время = номер транзакции (номера выдаются по возрастанию)



При одновременной работе нескольких сеансов возникает задача: что делать, если несколько две транзакции одновременно обращаются к одной и той же строке? Если обе транзакции читающие, сложностей нет. Если обе пишущие — тоже (в этом случае они выстраиваются в очередь и выполняют изменения друг за другом). Самый интересный вариант — как взаимодействуют пишущая и читающая транзакции.

Простых пути два. Такие транзакции могут блокировать друг друга — но тогда страдает производительность. Либо же читающая транзакция сразу видит изменения, сделанные пишущей транзакцией, даже если они не зафиксированы — но это очень плохо, ведь изменения могут быть отменены (это называется «грязным чтением»).

PostgreSQL идет сложным путем и использует *многоверсионность* — хранит несколько версий одной и той же строки. При этом пишущая транзакция работает со своей версией, а читающая видит свою.

Чтобы отличать версии друг от друга, они помечаются двумя отметками, определяющими «время» действия данной версии.
В качестве времени используются всегда возрастающие номера транзакций (в действительности все немного сложнее, но это детали).
Когда строка создается, она помечается номером транзакции, выполнившей команду INSERT. Когда удаляется — версия помечается номером транзакции, выполнившей DELETE (но физически не удаляется). UPDATE состоит из двух операций DELETE и INSERT.

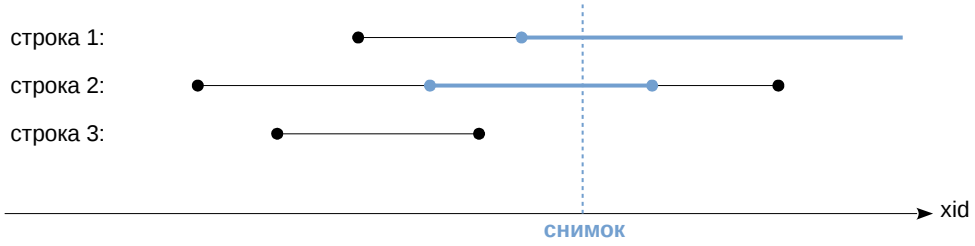
https://postgrespro.ru/docs/postgresql/9.6/mvcc-intro

Снимок данных

Согласованный срез на определенный момент времени

номер транзакции — определяет момент времени

список активных транзакций — чтобы не смотреть на еще не зафиксированные изменения



Но транзакция, работающая с таблицей, должна видеть только одну версию каждой строки (или не видеть ни одной). Для этого транзакция работает со *снимком данных,* созданным в определенный момент времени. Снимок — это не физическая копия данных, а всего несколько чисел:

- номер текущей транзакции на момент создания снимка (он определяет тот самый момент времени),

- и список активных транзакций на этот момент.

Список нужен для того, чтобы рассматривать изменения только тех транзакций, которые были зафиксированы до создания снимка. Нас не интересуют транзакции, которые начались до создания снимка, но еще не были зафиксированы, а также те, которые начались позже создания снимка.

Зная снимок, мы всегда можем сказать, какая из версий строки будет в нем видна. Иногда это будет актуальная (самая последняя) версия, как для строки 1 на иллюстрации. Иногда не самая последняя: строка 2 удалена (и изменение уже зафиксировано), но транзакция еще продолжает видеть эту строку, пока работает со своим снимком. Это правильное поведение — оно дает согласованную картину данных на выбранный момент времени.

Какие-то строки вовсе не попадут в снимок: строка 3 удалена до того, как был построен снимок, поэтому в снимке ее нет.

Блокировки

Postgres PROFESSIONAL

Блокировки строк

- чтение никогда не блокирует строк
- изменение строки блокирует ее для изменений, но не для чтений

Блокировки таблиц

- запрещают изменение или удаление таблицы, пока с ней идет работа
- запрещают чтение таблицы при перестроении или перемещении
- и т. п.

Время жизни блокировок

- устанавливаются по мере необходимости или вручную
- снимаются автоматически при завершении транзакции

Что же дает многоверсионность? Она позволяет обойтись только самым необходимым минимумом блокировок, тем самым увеличивая производительность системы.

Основные блокировки устанавливаются на уровне строк. При этом чтение никогда не блокирует ни читающие, ни пишущие транзакции. Изменение строки не блокирует ее чтение. Единственный случай, когда транзакция будет ждать освобождения блокировки — если она пытается менять строку, которая уже изменена другой, еще не зафиксированной, транзакцией.

Блокировки также устанавливаются на более высоком уровне, в частности на таблицах. Они нужны для того, чтобы никто не смог удалить таблицу, пока другие транзакции читают из нее данные, или чтобы запретить доступ к перестраиваемой таблице. Как правило, такие блокировки не вызывают проблем, поскольку удаление или перестроение таблиц — очень редкая операция.

Все необходимые блокировки устанавливаются автоматически и автоматически же снимаются при завершении транзакции. Можно также установить и дополнительные пользовательские блокировки; необходимость в этом возникает не часто.

https://postgrespro.ru/docs/postgresql/9.6/explicit-locking

Уровни изоляции

Read uncommitted — *не поддерживается PostgreSQL*

- позволяет читать не зафиксированные данные

Read committed — *используется по умолчанию*

- снимок строится на момент начала оператора
- одинаковые запросы могут каждый раз получать разные данные

Repeatable read

- снимок строится на момент начала транзакции
- транзакция может завершиться ошибкой сериализации

Serializable

- полная изоляция, но дополнительные накладные расходы
- транзакция может завершиться ошибкой сериализации

Стандарт SQL определяет четыре уровня изоляции: чем строже уровень, тем меньше влияния оказывают параллельно работающие транзакции друг на друга. Во времена, когда стандарт принимался, считалось, что чем строже уровень, тем сложнее его реализовать и тем сильнее его влияние на производительность (с тех пор эти представления несколько изменились).

Самый нестрогий уровень **read uncommitted** допускает грязные чтения и не поддерживается PostgreSQL — он не представляет практической ценности и не дает выигрыша в производительности.

Уровень **read committed** является уровнем изоляции по умолчанию в PostgreSQL. На этом уровне снимок данных строится в начале выполнения каждого оператора SQL. Таким образом, оператор работает с неизменной и согласованной картиной данных, но два одинаковых запроса, следующих один за другим, могут показать разные данные.

На уровне **repeatable read** снимок строится в начале транзакции — поэтому все запросы в одной транзакции видят одни и те же данные. Этот уровень удобен, например, для отчетов, состоящих из нескольких запросов.

Уровень **serializable** гарантирует полную изоляцию: можно писать операторы так, как будто транзакция работает одна. Плата за удобство — определенная доля транзакций завершается с ошибкой; приложение должно уметь повторять такие транзакции.

https://postgrespro.ru/docs/postgresql/9.6/transaction-iso

Очистка

Postgres PROFESSIONAL

Старые версии строк хранятся вместе с актуальными

- со временем размер таблиц и индексов увеличивается

Процесс очистки (vacuum)

- удаляет версии строк, которые уже не нужны
  (то есть не видны ни в одном снимке данных)
- работает параллельно с остальными процессами
- удаленные версии отставляют в файлах данных «дыры»,
  которые затем используются для новых версий строк
- процесс обычно работает автоматически (autovacuum)

Полная очистка

- полностью перестраивает файлы данных, делая их компактными
- блокирует таблицу на время работы

В PostgreSQL все версии строк — и актуальные, и старые — хранятся вместе в одном и том же файле данных. Понятно, что со временем старые версии строк накапливаются и это приводит к разрастанию таблиц (и индексов) и снижению производительности.

Между тем нет необходимости хранить неактуальные версии строк, которые уже не видны ни в одном снимке данных. Такие версии удаляются специальным процессом *очистки* (vacuum). Очистка физически удаляет ненужные версии из файлов, оставляя «дыры», которые затем используется для размещения новых версий строк.

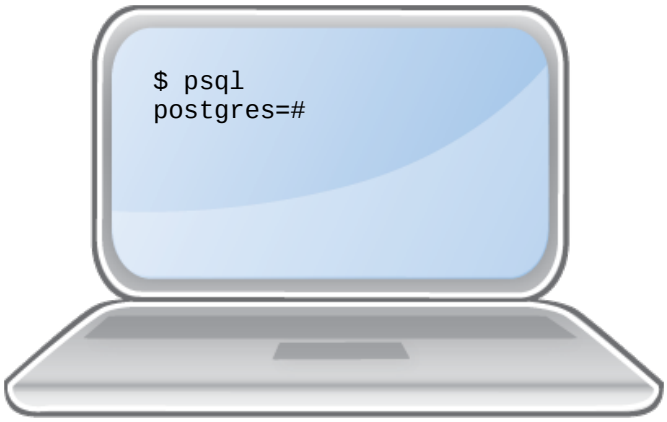
Очистка обычно работает автоматически и настраивается администратором так, чтобы очищать данные вовремя, не допуская большого увеличения размера файлов. Ей соответствует фоновый процесс autovacuum launcher (который запускает рабочие процессы autovacuum worker).

Очистка не блокирует остальные процессы и работает параллельно с ними.

Существует возможность полностью перестроить таблицу и ее индексы, выполнив полную очистку (vacuum full). При этом файлы получаются компактными, но этот процесс полностью блокирует работу с таблицей на время своей работы.

https://postgrespro.ru/docs/postgresql/9.6/routine-vacuuming

Демонстрация



Итоги

В файлах данных могут храниться несколько версий каждой строки

Транзакции работают со снимком данных — согласованным срезом на определенный момент времени

Писатели не блокируют читателей,
читатели не блокируют никого

Время создания снимка влияет на уровень изоляции

Версии строк накапливаются, поэтому нужна периодическая очистка

Практика

1. Создайте таблицу с одной строкой.
2. Начните первую транзакцию и выполните запрос к таблице.
3. Во втором сеансе удалите строку и зафиксируйте изменения.
4. Сколько строк увидит первая транзакция, выполнив тот же запрос повторно? Проверьте.
5. Завершите первую транзакцию.
6. Повторите все то же самое, но пусть теперь транзакция работает на уровне изоляции repeatable read:

   ```
   BEGIN ISOLATION LEVEL REPEATABLE READ;
   ```

   Объясните отличия.