Архитектура PostgreSQL

Темы

Структура памяти, буферный кэш

Серверные процессы

Организация данных в файловой системе

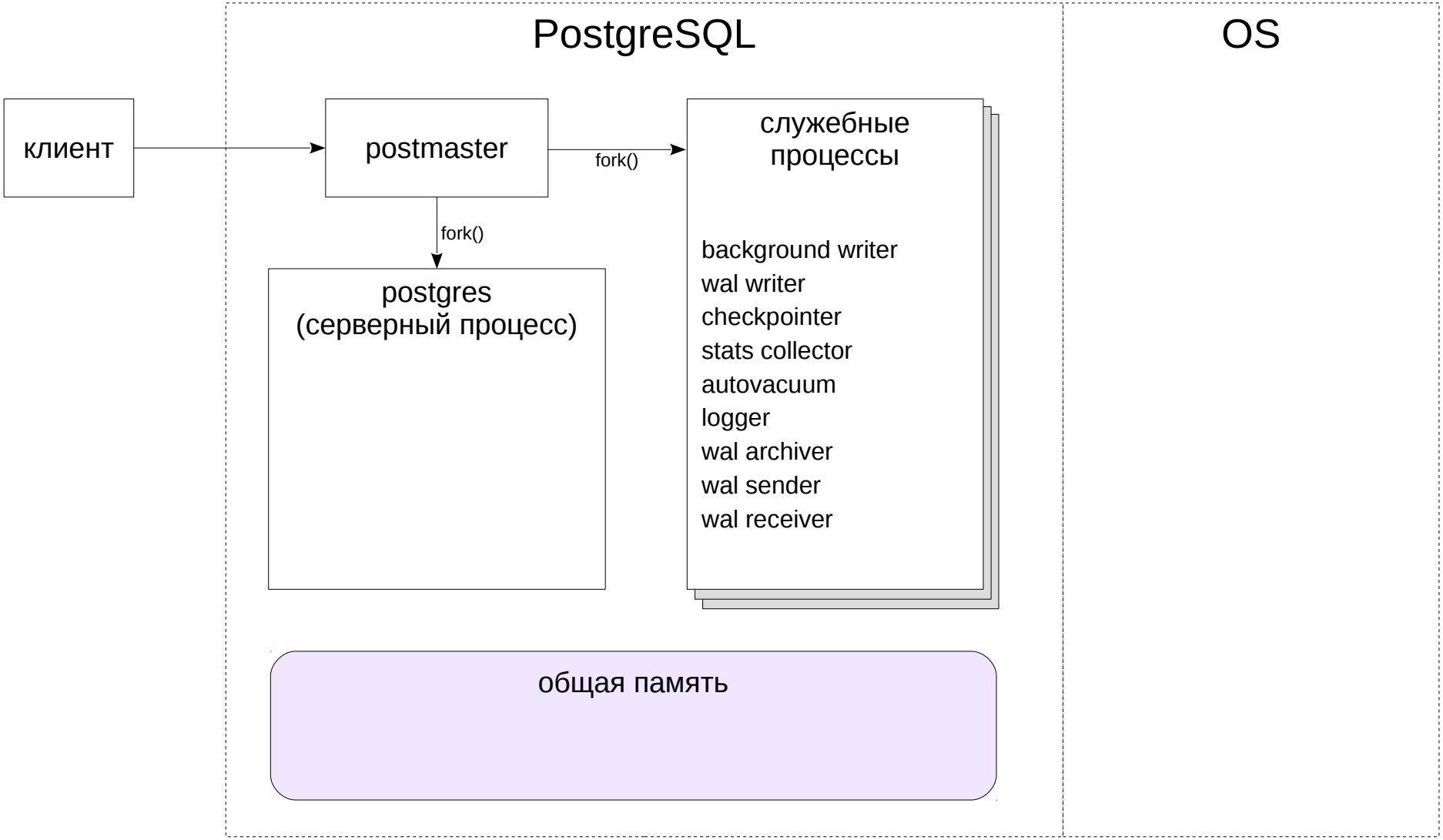
Журнал упреждающей записи

Транзакции и многоверсионность

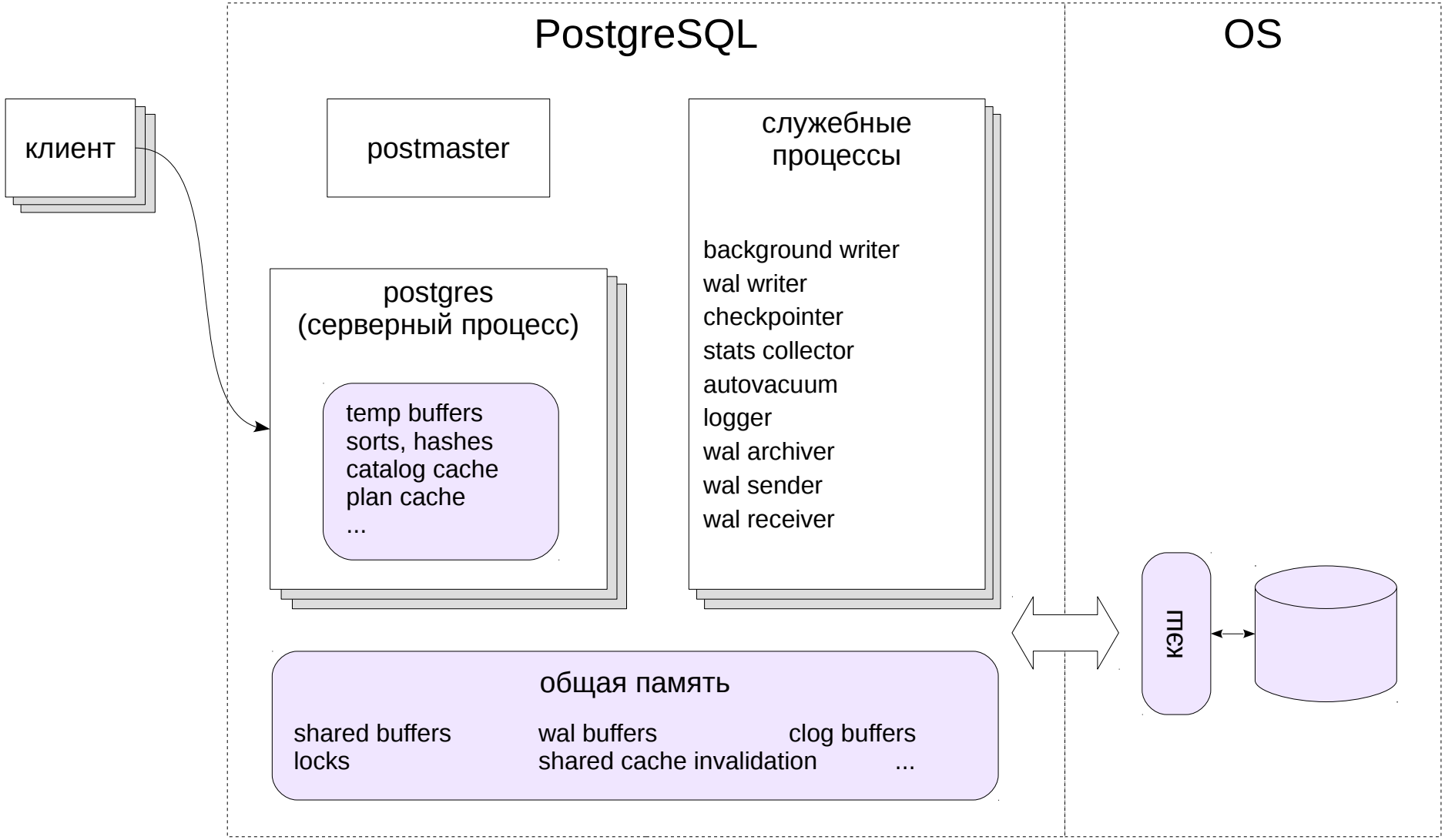
Схема обработки запросов

Расширяемость системы

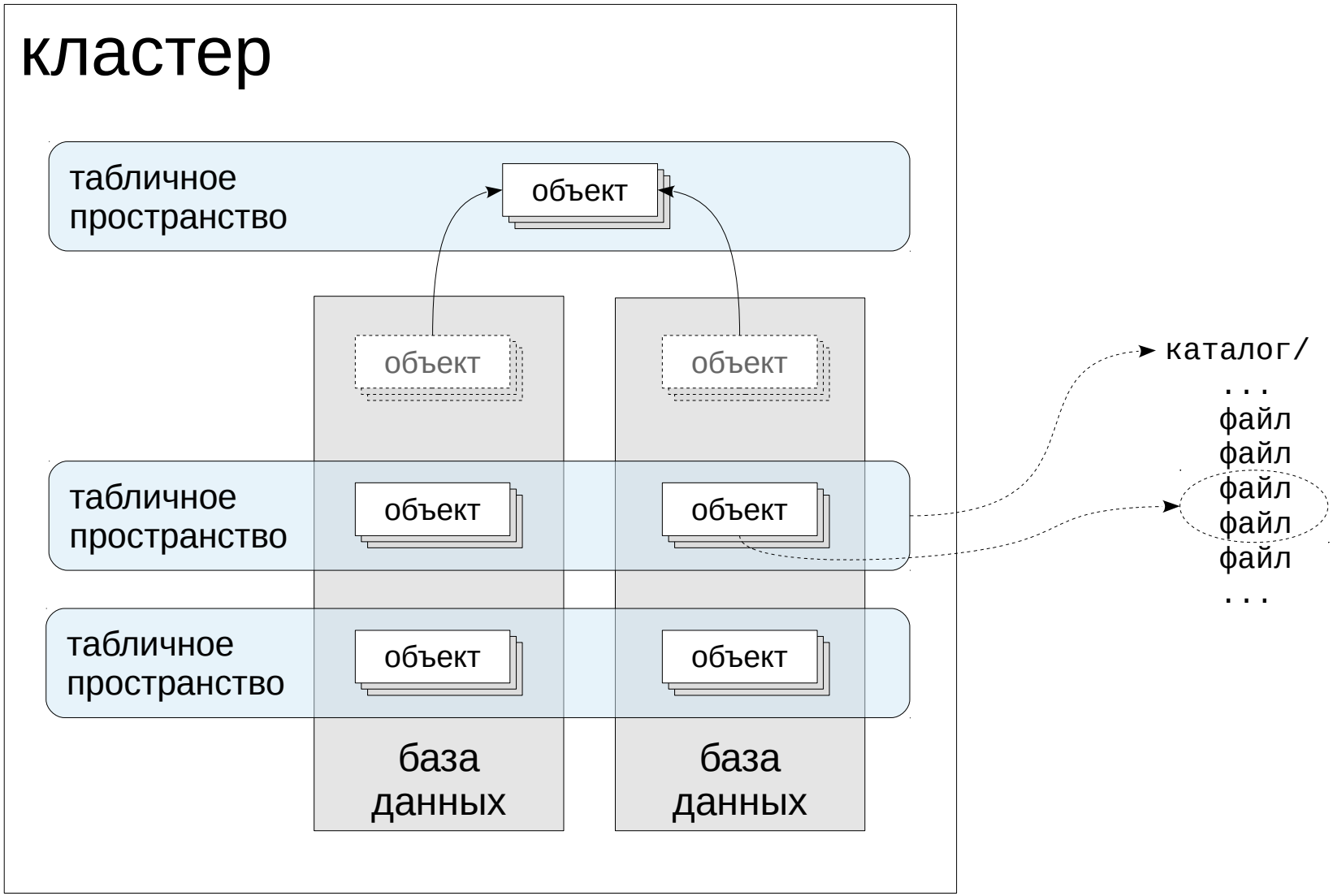
Процессы и память



Процессы и память



Организация данных



Буферный кэш

Массив буферов:

- страницы (обычно 8 КБ)
- дополнительная информация

Как и другие структуры в памяти, защищен блокировками

Вытеснение

- часто используемые страницы остаются
- редко используемые заменяются

«Грязные» буферы постепенно записываются на диск

- процесс background writer
  (но иногда и серверный процесс)

Транзакции

Последовательность операций, составляющая логическую единицу работы

Атомарность – все или ничего

- при фиксации выполняются все операции,
  при откате не выполняется ни одна

Согласованность – целостность данных

- система переходит из одного согласованного состояния в другое

Изоляция – от других транзакций

- на результат не должны оказывать влияние
  другие параллельно работающие транзакции

Долговечность – даже после сбоя

- зафиксированные изменения никогда не теряются

Журнал (WAL)

В журнал упреждающей записи (write-ahead log) записывается информация, достаточная для повторного выполнения всех действий при восстановлении

Обязан попасть на диск раньше, чем измененные страницы

- процесс wal writer

Записывается в момент фиксации, либо асинхронно

Сохраняется в файлы (по 16 МБ), старые могут архивироваться

- процесс wal archiver

Контрольная точка – принудительный сброс на диск всех грязных буферов, чтобы не хранить много файлов

- процесс checkpointer

Многоверсионность (MVCC)

Postgres PROFESSIONAL

Каждая транзакция работает со снимком

- согласованные данные на определенный момент времени

В страницах хранятся версии строк

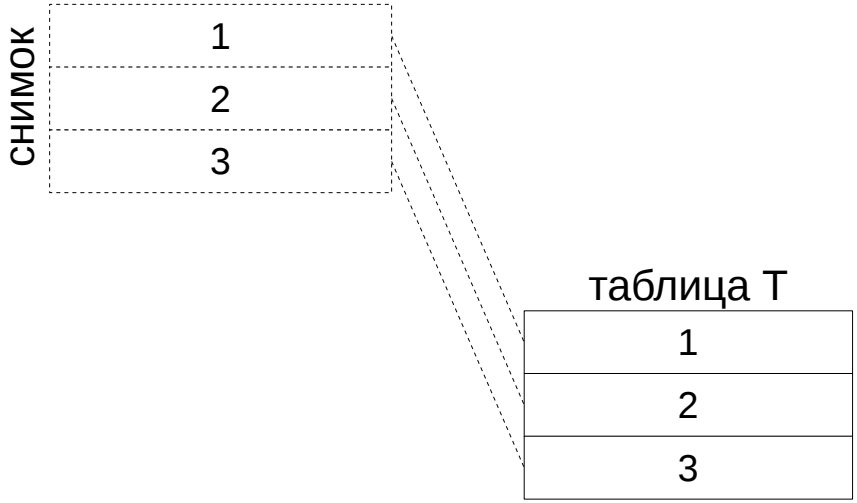
- при удалении старая версия строки остается в странице
- при изменении остается старая версия и появляется новая

Периодически происходит очистка версий,
не доступных больше ни в одном снимке

- процесс autovacuum launcher/workers (или вручную)

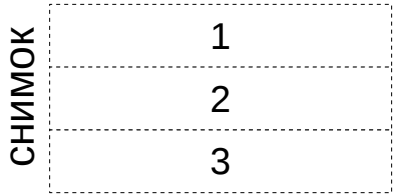
Пример

A. select * from T;

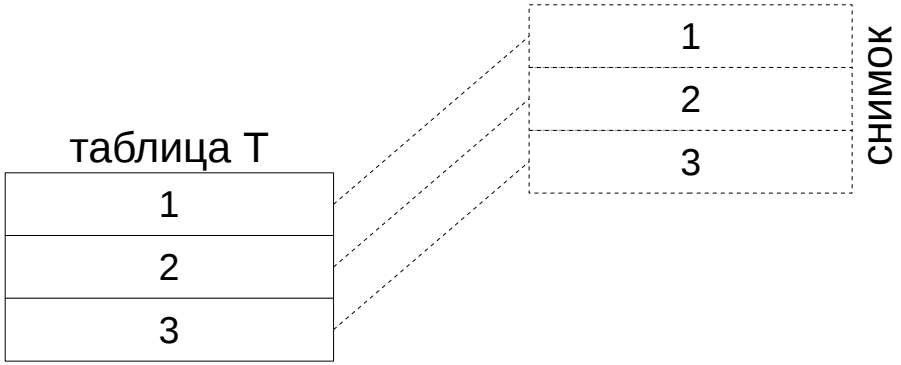


Пример

A. select * from T;

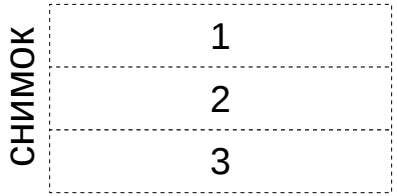


B. update T set ...;

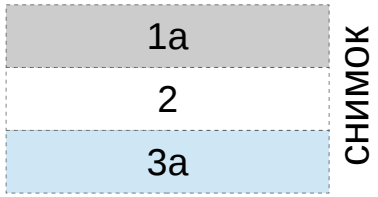


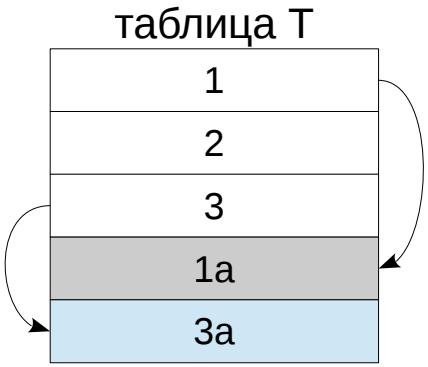
Пример

A. select * from T;



B. update T set ...;





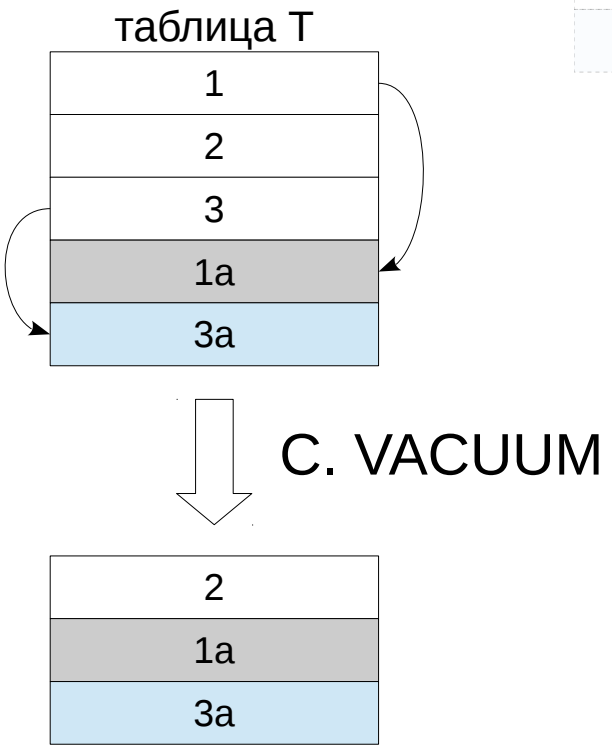
Пример

A. select * from T;

снимок
1
2
3

B. update T set ...;

	снимок
1a	
2	
3a	



Блокировки

Блокировки на уровне строки

Благодаря многоверсионности:

- чтение не блокирует чтение
- чтение не блокирует изменение
- изменение не блокирует чтение

Единственная блокировка:

- изменение блокирует изменение

Обработка запросов

текст запроса

Parser – разбор запроса ← системный каталог

дерево запроса

Rewriter – переписывание запроса ← правила

измененное дерево

Planner – оптимизация запроса ← статистика

план запроса

Executor – выполнение запроса ← данные

результат

Расширяемость

Расширяемость заложена в архитектуре системы

Языки программирования

- SQL, PL/pgSQL, PL/Perl, PL/Python, PL/Tcl, PL/R, PL/Java, PL/V8...

Функции

Типы данных и операторы

Типы индексов и методы доступа

- GiST, GIN, SP-GiST...

Обертки сторонних данных (FDW)

Итоги

Познакомились со структурой памяти, серверными процессами и хранением данных в файловой системе

Рассмотрели поддержку транзакций с помощью журнала упреждающей записи
и механизма многоверсионности

Познакомились со схемой обработки запросов

Узнали про возможности расширения системы

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

Темы

Структура памяти, буферный кэш

Серверные процессы

Организация данных в файловой системе

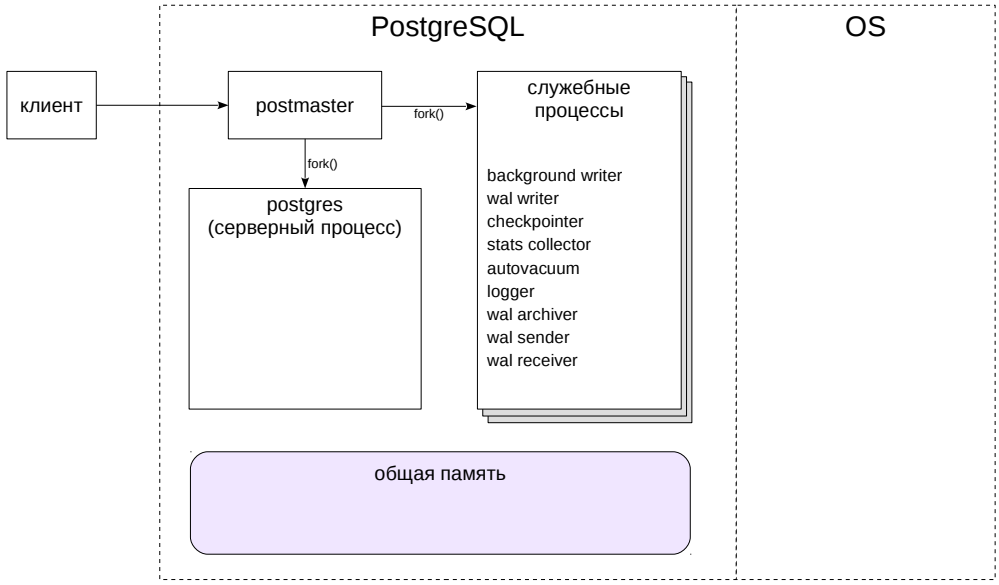
Журнал упреждающей записи

Транзакции и многоверсионность

Схема обработки запросов

Расширяемость системы

Процессы и память



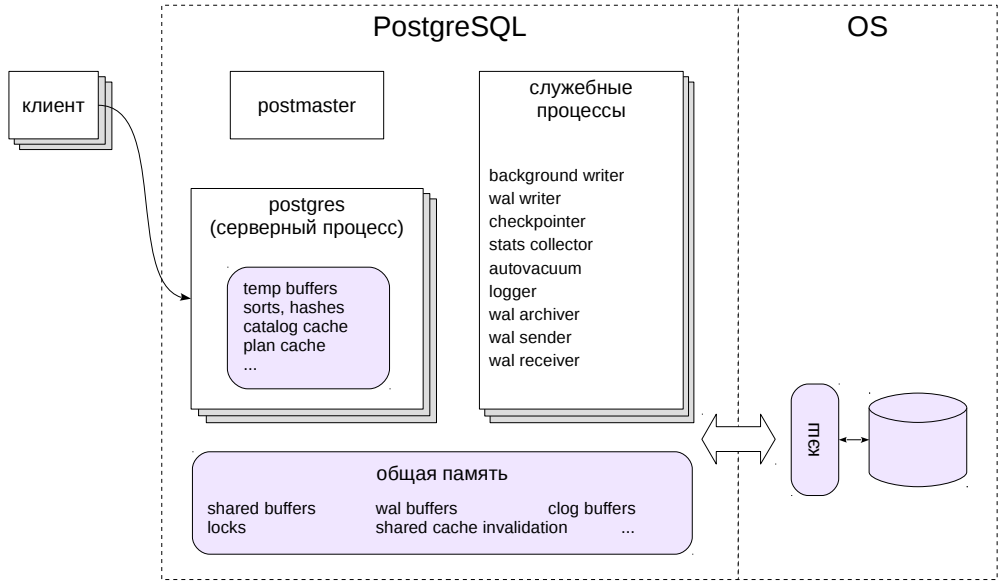
При старте сервера в первую очередь запускается процесс, традиционно называемый postmaster.

Postmaster:

- создает структуры общей памяти
- порождает ряд служебных процессов
- следит за состоянием порожденных процессов
- слушает входящие соединения

Клиент соединяется с процессом postmaster. Postmaster порождает серверный процесс и дальше клиент работает уже с ним. Поэтому при большом числе соединений следует использовать пул.

Процессы и память



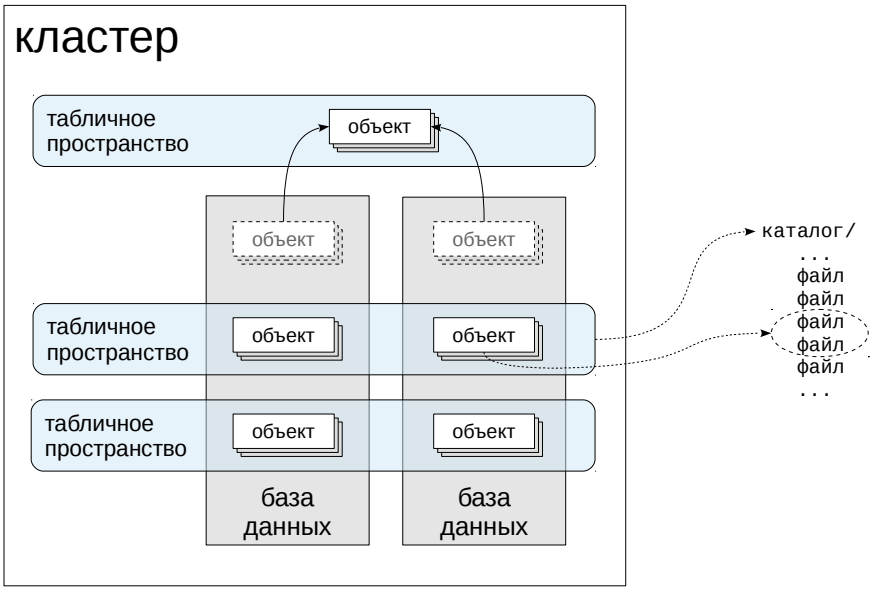
4

Большую часть общей для всех серверных процессов памяти занимает буферный кэш (shared buffers), необходимый для ускорения работы с данными на диске. Обращение к дискам происходит через операционную систему (которая тоже кэширует данные в оперативной памяти). PostgreSQL полагается на то, что вызов fsync() гарантирует попадание данных из памяти на диск.

Кроме буферного кэша в общей памяти находится информация о блокировках и другое; через нее же серверные процессы общаются друг с другом.

У каждого серверного процесса есть своя локальная память. В ней находится кэш каталога (часто используемая информация о базе данных), планы запросов, рабочее пространство для выполнения запросов и другое.

Организация данных



Экземпляр СУБД работает с несколькими базами данных (с так называемым кластером; это слово имеет здесь не тот смысл, который в него обычно вкладывается).

Хранение данных на диске организовано с помощью табличных пространств. Табличное пространство указывает расположение данных (каталог на файловой системе). Оно может использоваться несколькими базами данных.

Объекты (таблицы и индексы) хранятся в файлах; каждый объект занимает один или несколько (2-3) файлов внутри каталога табличного пространства. Кроме того, файлы разбиваются на части по 1 ГБ. Необходимо учитывать влияние потенциально большого количества файлов на используемую файловую систему.

http://www.postgresql.org/docs/9.4/static/storage-file-layout.html

Буферный кэш

Массив буферов:

- страницы (обычно 8 КБ)
- дополнительная информация

Как и другие структуры в памяти, защищен блокировками

Вытеснение

- часто используемые страницы остаются
- редко используемые заменяются

«Грязные» буферы постепенно записываются на диск

- процесс background writer
  (но иногда и серверный процесс)

Буферный кэш используется для сглаживания скорости работы памяти и дисков. Состоит из массива буферов, которые содержат страницы (блоки) данных и дополнительную информацию об этих страницах.

Размер страницы обычно составляет 8 КБ, хотя может устанавливаться при сборке.

Буферный кэш, как и другие структуры в памяти, защищен блокировками от одновременного доступа. Блокировки организованы достаточно эффективно, чтобы не создавать большой конкуренции.

Любая страница, с которой работает СУБД, попадает в кэш. Часто используемые страницы остаются в кэше надолго; редко используемые — вытесняются и заменяются другими страницами.

Буфер, содержащий измененную страницу, называется «грязным». Процесс background writer постепенно записывает их на диск в фоновом режиме — это позволяет снизить нагрузку на диски и увеличить производительность.

Транзакции

Последовательность операций, составляющая логическую единицу работы

Атомарность – все или ничего

- при фиксации выполняются все операции, при откате не выполняется ни одна

Согласованность – целостность данных

- система переходит из одного согласованного состояния в другое

Изоляция – от других транзакций

- на результат не должны оказывать влияние другие параллельно работающие транзакции

Долговечность – даже после сбоя

- зафиксированные изменения никогда не теряются

Транзакции представляют собой последовательности операций, которые должны удовлетворять требованиям ACID (atomicity, consistency, isolation, durability).

Атомарность подразумевает возможность зафиксировать или откатить изменения, сделанные в транзакции.

Согласованность обеспечивается проверкой правил, обеспечивающих целостность данных.

Изоляция может иметь несколько различных уровней, но так или иначе она должна защищать транзакции от воздействия работающих одновременно с ними других транзакций.

Эти три свойства достаточно тесно связаны друг с другом. Один из основных механизмов, обеспечивающих их эффективную реализацию, является многоверсионность.

Долговечность подразумевает возможность восстановить базу данных после сбоя в согласованном состоянии. Она обеспечивается журналом упреждающей записи.

Оба этих механизма рассматриваются ниже.

Журнал (WAL)

В журнал упреждающей записи (write-ahead log) записывается информация, достаточная для повторного выполнения всех действий при восстановлении

Обязан попасть на диск раньше, чем измененные страницы

процесс wal writer

Записывается в момент фиксации, либо асинхронно

Сохраняется в файлы (по 16 МБ), старые могут архивироваться

процесс wal archiver

Контрольная точка – принудительный сброс на диск всех грязных буферов, чтобы не хранить много файлов

процесс checkpointer

В журнал упреждающей записи (Write Ahead Log, WAL) записывается информация, достаточная для повторного выполнения всех действия с базой данных.

Записи этого журнала обязаны попасть на диск раньше, чем изменения в соответствующей странице. Тогда при восстановлении можно прочитать страницу с диска, посмотреть в ней номер последней записи WAL, и применить к странице все записи WAL, которые еще не были применены.

Записью журнала занимается процесс WAL Writer. Можно сохранять записи журнала непосредственно при фиксации изменений. В таком случае гарантируется, что зафиксированные данные не пропадут при сбое. Второй вариант — сохранять записи асинхронно, что более эффективно. В этом случае некоторые зафиксированные данные могут пропасть, но согласованность все равно гарантируется.

Журнал состоит из нескольких файлов (обычно по 16 МБ), которые циклически перезаписываются. Старые файлы могут сохраняться и архивироваться процессом WAL Archiver.

Поскольку страница может долго не попадать на диск, возникает необходимость хранить неопределенно много журнальных записей. Чтобы избежать этого, периодически происходит т. н. контрольная точка, при которой все грязные буферы принудительно сбрасываются на диск. Этим занимается процесс Checkpointer.

Многоверсионность (MVCC)

Каждая транзакция работает со снимком

- согласованные данные на определенный момент времени

В страницах хранятся версии строк

- при удалении старая версия строки остается в странице
- при изменении остается старая версия и появляется новая

Периодически происходит очистка версий,
не доступных больше ни в одном снимке

- процесс autovacuum launcher/workers (или вручную)

Идея многоверсионности (Muliversion Concurrency Control, MVCC) состоит в том, что в страницах таблицы может храниться несколько версий одних и тех же строк. При этом каждая транзакция работает со своим снимком данных, в котором видит только доступные ей данные, согласованные на определенный момент времени.

При удалении строки она не удаляется физически из страницы, а помечается номером удалившей его транзакции.

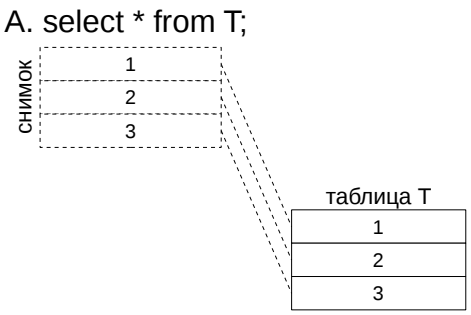
Новая строка помечается номером создавшей его транзакции.

Обновление реализуется как удаление старой строки и вставка новой.

Таким образом, в каждой версии строки хранится информация о начале и конце ее действия. На основании этой информации и строятся снимки для разных транзакций.

Старые версии строк, которые не видны ни одной из активных транзакций, должны быть физически удалены, чтобы освободить занимаемое ими место. Этим занимается процесс autovacuum launcher, запускающий для выполнения работы процессы autovacuum worker. Также очистку можно запустить вручную командой VACUUM.

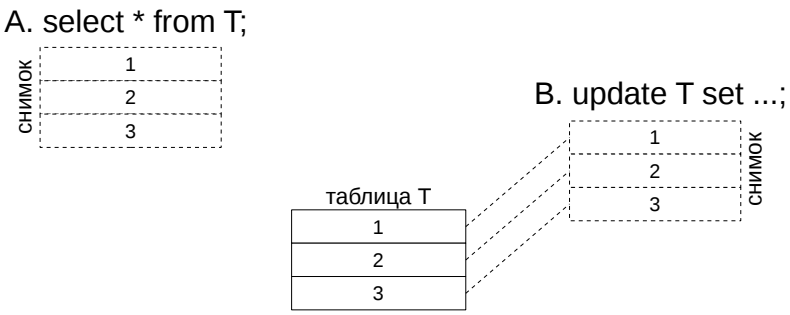
Пример



Изначально таблица T содержит три строки (1, 2, 3).

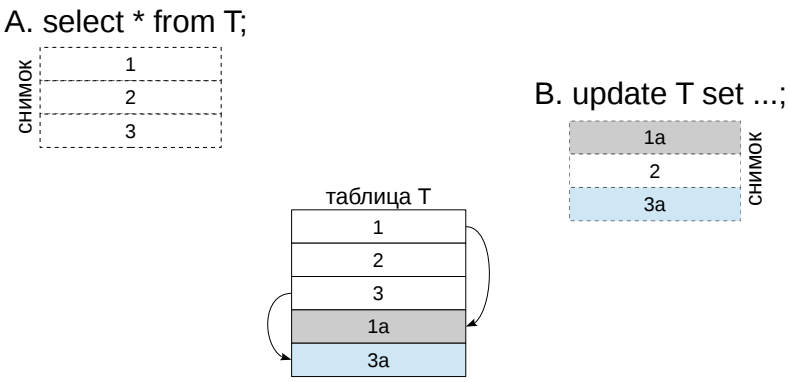
В транзакции A выполняется запрос. Транзакция работает со снимком, в котором видны три исходные строки.

Пример



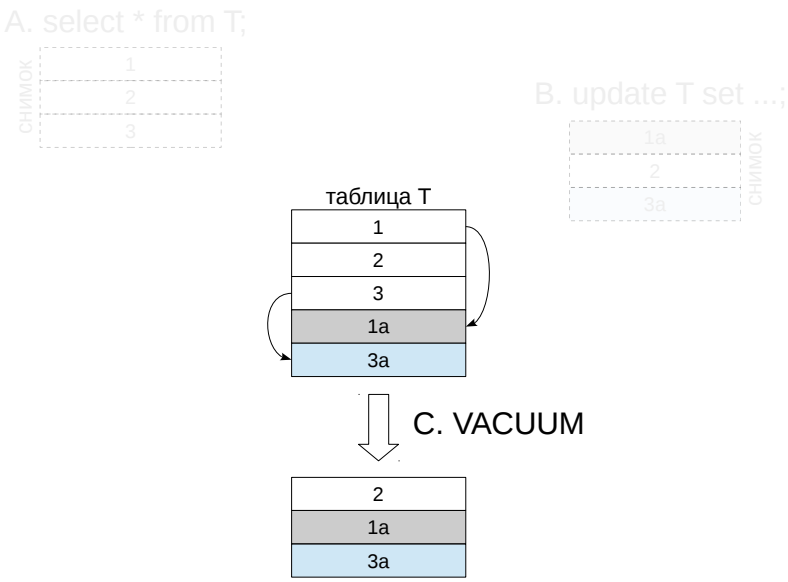
В транзакции B выполняется изменение таблицы. Для нее также строится снимок, содержащий исходные строки.

Пример



В транзакции B выполняется изменение таблицы. При этом в странице остаются старые версии строк (они нужны для снимка транзакции A) и создаются новые версии 4 и 5 (они попадают в снимок транзакции B).

Пример



После того, как транзакции завершены, старые версии строк могут быть удалены, так как не входят ни в один снимок.

Блокировки

Блокировки на уровне строки

Благодаря многоверсионности:

- чтение не блокирует чтение
- чтение не блокирует изменение
- изменение не блокирует чтение

Единственная блокировка:

- изменение блокирует изменение

Преимущество многоверсионности перед реализациями, основанными только на блокировках, состоит в существенно большей эффективности, так как гарантируется, что блокируется только одновременное изменение одних и тех же строк. Но читатели никогда не блокируют писателей, а писатели — читателей.

Обработка запросов

текст запроса

Parser – разбор запроса ← системный каталог

дерево запроса

Rewriter – переписывание запроса ← правила

измененное дерево

Planner – оптимизация запроса ← статистика

план запроса

Executor – выполнение запроса ← данные

результат

Текст запроса проходит несколько этапов обработки.

Анализатор (parser) выполняет разбор текста запроса. Результатом его работы является дерево разбора.

Далее запрос переписывается (rewriter) с помощью системы правил. В частности, на этом этапе происходит замена представлений на текст запроса. Результатом является измененное дерево разбора.

Планировщик (planner) выбирает для запроса «лучший» план выполнения. Критерий оптимизации — минимизация стоимости выполнения. Стоимость вычисляется на основе статистики, которую собирает процесс Stats Collector, а также Vacuum. План определяет, в каком порядке будут соединяться таблицы, какие методы доступа и соединений будут использоваться.

План запроса передается на выполнение (executor). В случае запроса select результатом выполнения является набор строк, удовлетворяющий условиям.

Расширяемость

Расширяемость заложена в архитектуре системы

Языки программирования

- SQL, PL/pgSQL, PL/Perl, PL/Python, PL/Tcl, PL/R, PL/Java, PL/V8...

Функции

Типы данных и операторы

Типы индексов и методы доступа

- GiST, GIN, SP-GiST...

Обертки сторонних данных (FDW)

PostgreSQL спроектирован с расчетом на расширяемость.

В его стандартном составе, помимо SQL и PL/pgSQL, три языка программирования (PL/Perl, PL/Python, PL/Tcl), еще несколько доступно в качестве расширений и есть возможность добавлять новые языки.

http://www.postgresql.org/docs/9.4/static/plhandler.html

Поддерживается множество различных типов данных, включая как стандартные (числа, строки, даты), так и логический тип, перечисления, массивы, типы для работы с геометрическими объектами и др.

http://www.postgresql.org/docs/9.4/static/datatype.html

Если их не достаточно, можно реализовать свой тип и определить для него новые операции. Чтобы индексировать поля с новым типом, можно реализовать свой тип индекса.

http://www.postgresql.org/docs/9.4/static/indexam.html

Для доступа к данным вне СУБД можно написать обертку сторонних данных (foreign data wrapper, FDW).

http://www.postgresql.org/docs/9.4/static/fdwhandler.html

Итоги

Познакомились со структурой памяти, серверными процессами и хранением данных в файловой системе

Рассмотрели поддержку транзакций с помощью журнала упреждающей записи
и механизма многоверсионности

Познакомились со схемой обработки запросов

Узнали про возможности расширения системы