нетология

PostgreSQL

Роман
Гордиенко

Роман Гордиенко

Backend Developer, Factory5

Роман Гордиенко

План занятия

1. Историческая справка
2. Текущая версия
3. CAP-теорема
4. Введение в архитектуру
5. Транзакции в PostgreSQL
6. WAL - журнал упреждающей записи
7. Многоверсионность
8. Типы данных
9. Индексация данных
10. Расширяемость
11. Производительность
12. Безопасность
13. Масштабирование
14. Итоги
15. Домашнее задание

Историческая справка

Историческая справка

Объектно-реляционная система управления базами данных, именуемая сегодня PostgreSQL, произошла от **пакета POSTGRES**, написанного в Беркли, Калифорнийском университете.

После двух десятилетий разработки **PostgreSQL стал самой развитой СУБД** с открытым исходным кодом.

Проект POSTGRES, возглавляемый профессором Майклом Стоунбрейкером, спонсировали агентство DARPA при Минобороны США, Управление военных исследований (ARO), Национальный Научный Фонд (NSF) и компания ESL, Inc. Реализация POSTGRES началась в 1986 г.

Историческая справка

В 1994 г. Эндри Ю и Джолли Чен добавили в POSTGRES интерпретатор языка SQL.

Уже с новым именем Postgres95 был опубликован в Интернете и начал свой путь как потомок разработанного в Беркли POSTGRES, с открытым исходным кодом.

В 1996 г. стало понятно, что имя «Postgres95» не выдержит испытание временем. Было выбрано новое имя, PostgreSQL, отражающее связь между оригинальным POSTGRES и более поздними версиями с поддержкой SQL.

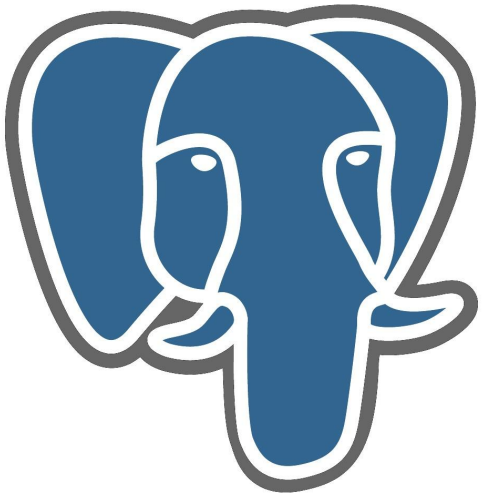
В то же время, продолжена нумерация версий с 6.0, вернувшись к последовательности, начатой в проекте Беркли POSTGRES.

Текущая версия

Текущая версия

Текущая версия **PostgreSQL 13.2** была выпущена **2021-02-11**.

Подробно ознакомиться можно тут:
postgresql.org/docs/release/13.2/



Взято с сайта: medium.com

CAP-теорема

CAP-теорема

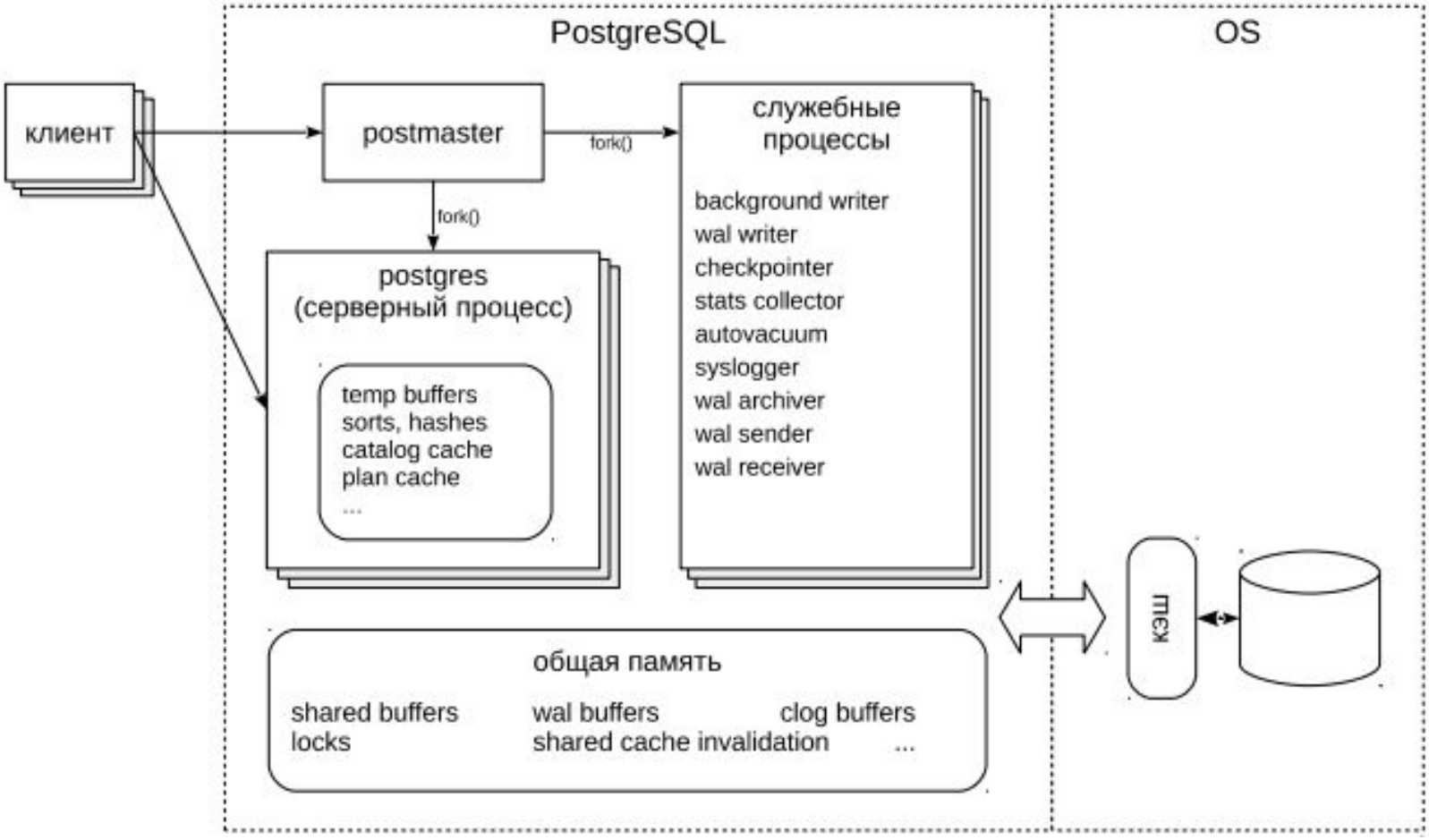
БД Postgresql можно охарактеризовать следующим образом:

- имеется репликация Master-Slave;
- работа с Master в асинхронном или синхронном режиме;
- используется двухфазный коммит транзакций для обеспечения согласованности;
- в случае разделения - взаимодействие с системой нарушается.

Система не может продолжать работу в случае разделения.
Но обеспечивает согласованность и доступность. Это CA система.

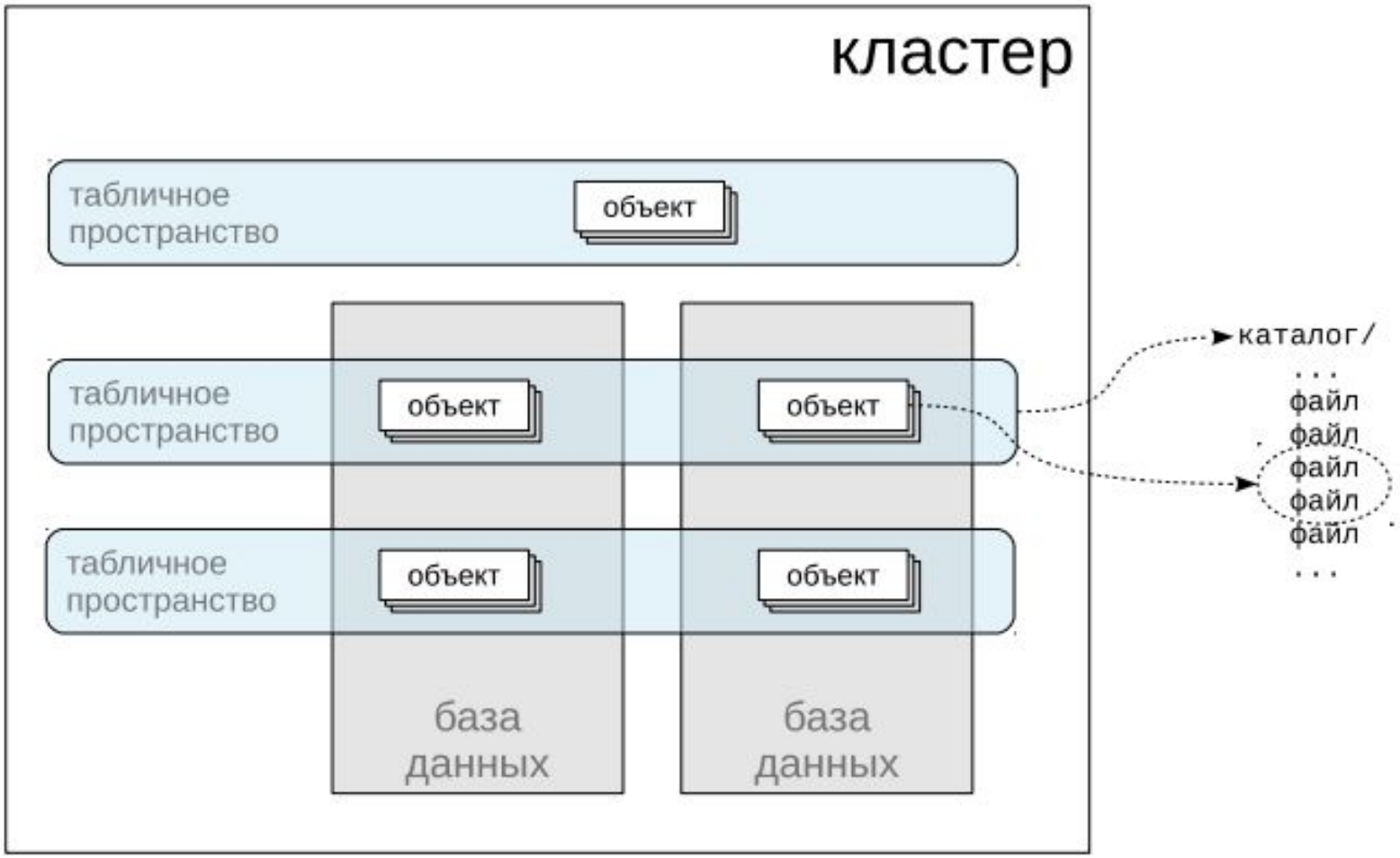
Введение в архитектуру

Введение в архитектуру



Взято с сайта: snakeproject.ru/rubric/files/doc/postgres_arch.pdf

Введение в архитектуру



Взято с сайта: snakeproject.ru/rubric/files/doc/postgres_arch.pdf

Введение в архитектуру

Основные моменты:

- При подключении к серверу клиент соединяется с процессом postmaster.
- Postmaster порождает серверный процесс и дальше клиент работает уже с ним.
- У всех серверных процессов в рамках одного экземпляра Postgresql имеется общая память.
- Обращение к дисковым устройствам происходит средствами ОС (которая тоже кэширует данные в оперативной памяти).
- Экземпляр PostgreSQL работает с несколькими базами данных. Эти базы данных называются кластером.
- Хранение данных на диске организовано с помощью табличных пространств.
- Табличное пространство указывает расположение данных (каталог на файловой системе).

Транзакции в PostgreSQL

Транзакции в PostgreSQL

PostgreSQL соответствует принципу ACID:

- atomicity - атомарность;
- consistency - согласованность;
- isolation - изолированность;
- durability - долговечность.

Механизмом, обеспечивающих эффективную реализацию ACI, является многоверсионность.

Долговечность (D) обеспечивается журналом предварительной записи.

Транзакции в PostgreSQL

Поддерживаемые **уровни изоляции в Postgresql**:

- Read uncommited - чтение незафиксированных данных;
- Read committed - чтение зафиксированных данных;
- Repeatable read - повторяемое чтение;
- Serializable - сериализуемость.

Транзакции в PostgreSQL

Для данных уровней изоляции есть особые неподдерживаемые их выполнение условия:

- **«грязное» чтение** (транзакция читает данные, записанные параллельной незавершённой транзакцией).
- **неповторяемое чтение** (при повторном чтении данных в рамках одной транзакции обнаружено, что они были изменены).
- **фантомное чтение** (при выполнении повторного возврата набора строк в рамках одной транзакции обнаружено, что он был изменен).
- **аномалия сериализации** (результат успешной фиксации группы транзакций оказывается несогласованным при всевозможных вариантах исполнения этих транзакций по очереди).

Транзакции в PostgreSQL

Уровень изоляции	“Грязное чтение”	Неповторяемое чтение	Фантомное чтение	Аномалия сериализации
Read uncommited	-	+	+	+
Read commited	-	+	+	+
Repeatable read	-	-	-	+
Serializable	-	-	-	-

WAL - журнал упреждающей записи

WAL - журнал упреждающей записи

Журнал упреждающей записи (Write Ahead Log, WAL) содержит информацию, достаточную **для повторного выполнения всех действий с БД**.

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.

Запись в WAL может происходить в синхронном и асинхронном режиме (процесс записи - WAL Writer).

WAL состоит из нескольких файлов (обычно по 16 мб) с циклической перезаписью или архивированием (процесс архивирование - WAL Archiver).

Для избежания накапливания буфера операций WAL - периодически операции принудительно “сбрасываются на диск” (процесс сбрасывания - Checkpointer).

WAL - журнал упреждающей записи

Журнальные файлы можно посмотреть в файловой системе в каталоге $PGDATA/pg_wal/.

Начиная с PostgreSQL 10, их также можно увидеть специальной функцией:

```
=> SELECT * FROM pg_ls_waldir() WHERE name = '000000010000000000000033';
```

```
            name            |   size   |      modification
----------------------------+----------+------------------------
 000000010000000000000033 | 16777216 | 2019-07-08 20:24:13+03
(1 row)
```

Взято с сайта: habr.com/ru/company/postgrespro/blog/459250/

Многоверсионность

Многоверсионность

Многоверсионность - разделение уровней представления данных.

На нижнем уровне имеем дело со страницами и физическим хранением данных в них.

Хранятся несколько версий строк. Операции со строками помечаются номер транзакции. Обновление реализуется как удаление старой строки и вставка новой. В каждой версии строки хранится информация о начале и конце ее действия.

На верхнем уровне имеем так называемые снимки данных.

Сники данных предоставляют согласованную картинку на момент времени за счет информации о начальном и конечном номере транзакции. Транзакции работаю со снимками данных.

Многоверсионность

Старые версии строк, не видимые ни одной из активных транзакций - физически удаляются.

- **Autovacuum Launcher** - процесс, запускающий очистку.
- **Autovacuum Worker** - процесс, выполняющий очистку.

Типы данных

Типы данных

Поддерживаемые типы данных указаны в документации на PostgreSQL: postgresql.org/docs/12/datatype.html

Стоит отметить, что дополнительно к стандартным типам данных, принятых в SQL поддерживаются:

- **Large Objects** (binary data up to 2Gb)
- **Геометрические типы** (point, line, circle,polygon, box)
- **GIS** (geo-data)
- **Network types** (inet ipv4/ipv6, cidr, macaddr)
- **Composite types** (union of other data types)

Индексация данных

Индексация данных

PostgreSQL поддерживает несколько полезных алгоритмов индексации:

- **Стандартные индексы** - B-tree, hash, R-tree, GiST (обобщенное поисковое дерево)
- **Частичные индексы** (partial indices) - можно создавать индекс по ограниченному подмножеству значений (например для значений в столбце > 0).
- **Функциональные индексы** (expressional indices) позволяют создавать индексы используя значения функции от параметра (например для значений в столбце, длина которых от 1 до 5 символов).

Расширяемость

Расширяемость

PostgreSQL спроектирован с расчетом на расширяемость.

В стандартный состав, помимо SQL и PL/pgSQL, входят три языка программирования (PL/Perl, PL/Python, PL/Tcl).

Возможные расширения PostgreSQL:

- типы данных;
- операции над типами данных;
- индексы;
- обертки для данных вне СУБД.

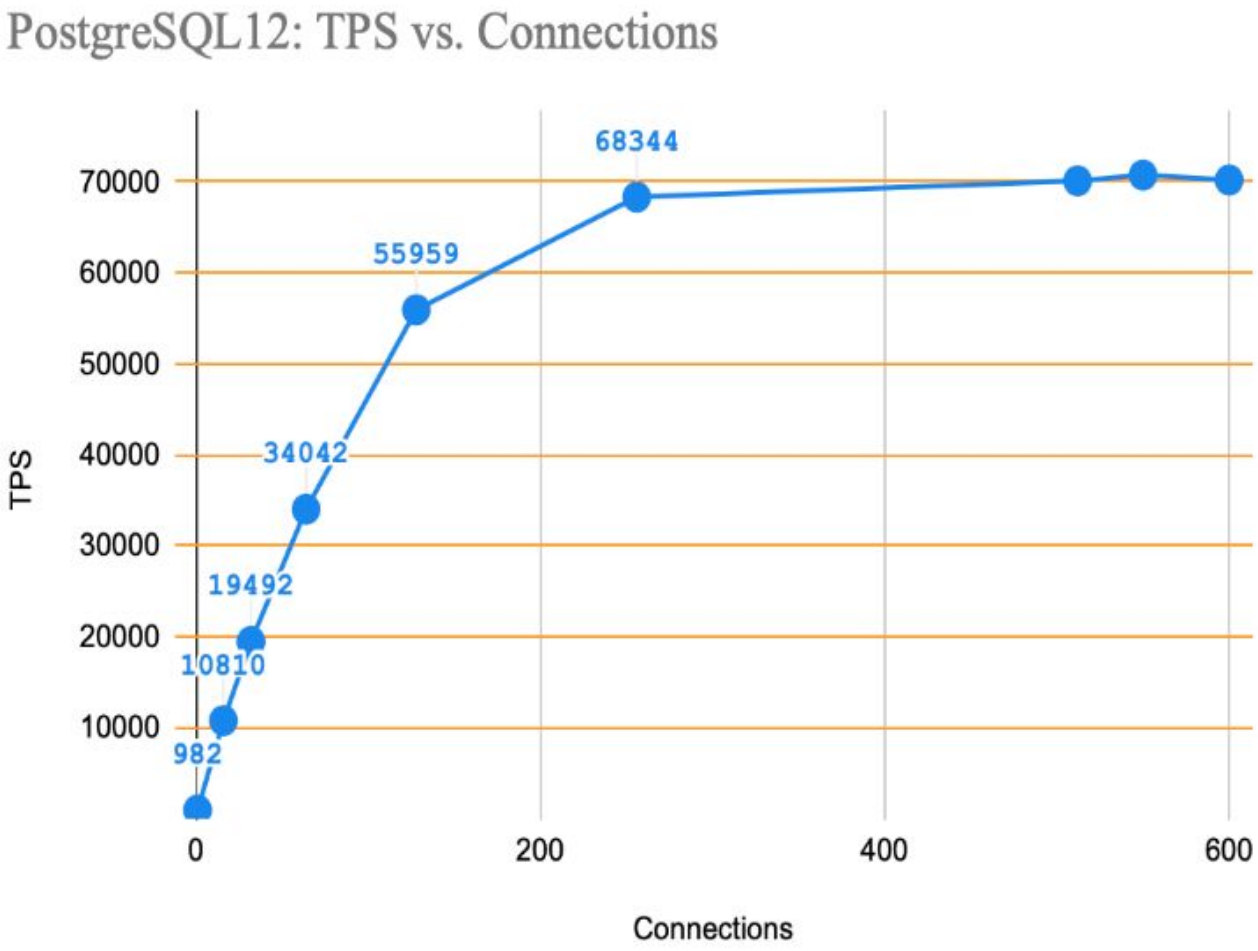
Производительность

Производительность

Производительность PostgreSQL обеспечивают:

- использование индексов;
- планировщик запросов;
- система блокировок;
- система управления буферами памяти и кэширования;
- табличные пространства;
- масштабируемость при конкурентной работе.

Производительность



Взято с сайта: enterprisedb.com

Безопасность

Безопасность

Безопасность обеспечивается **4-мя уровнями**:

- нельзя запустить под привилегированным пользователем;
- SSL/SSH между клиентом и сервером;
- гибкие системы клиентской аутентификации;
- тонко проработанная система ролей и прав ко всем объектам БД.

Масштабирование

Масштабирование

Масштабирование - это распределение данных для увеличения производительности и отказоустойчивости.

Масштабирование делится на:

- **шардирование** - разделение таблиц на куски по какому-либо принципу.
- **репликацию** - дублирование данных на разных экземплярах СУБД и формирование правил выполнения io операций.

Каждый из видов масштабирования также имеет деление на подтипы.

Эффективная работа с БД достигается путем правильного комбинирования типов шардирования и репликации.

Масштабирование

Шардирование можно разделить на:

- **вертикальное** - разделение таблиц на куски по какому-либо условию в рамках одного экземпляра СУБД.
- **горизонтальное** - разделение таблиц на куски по какому-либо условию в рамках нескольких экземпляров СУБД.

Условие шардирования может быть:

- **строгое** - значение строго равно чему-то, например *x=0*
- **список значений** - значение выбирается из списка, например *x in [‘a’, ‘b’, ‘c’]*
- **диапазонное** - значение из диапазона, например *x>0 and x<5*

Масштабирование

Репликацию можно разделить следующим образом:

- на разделяемых дисках;
- на уровне файловой системы;
- трансляция WAL;
- логическая репликация;
- master-slave репликация;
- multi-master репликация.

Масштабирование

Отказоустойчивость на разделяемых дисках позволяет избежать избыточности синхронизации путём непрерывного копирования БД и использования только 1го рабочего экземпляра БД в единицу времени.

Используется единственный дисковый массив для хранения данных, который разделяется между несколькими серверами.

Если основная копия СУБД откажет, запускается резервная копия.

Это обеспечивает быстрое переключение без потери данных.

Масштабирование

Репликация на уровне файловой системы - это зеркальное отражение файловой системы одного сервера на другой.

Ограничение:

Синхронизация должна гарантировать целостность копии файловой системы на резервном сервере.

Пример метода синхронизации файловых систем - DRBD.

Масштабирование

При **трансляции WAL** - данные из лога WAL асинхронно транслируются с ведущего узла на ведомый, где тут же применяются.

В случае остановки ведущего узла мы имеем его копию.

Ограничение:

Во время этого процесса ведомым узлом пользоваться нельзя.

Масштабирование

Логическая репликация позволяет одному серверу БД передавать поток изменений данных на другой сервер.

Поток данных является логическими изменениями обработки WAL.

Возможна настройка логической репликации для избранных таблиц.

Направление передачи данных не имеет значения (отсутствует ролевая модель master-slave).

Масштабирование

Master-slave репликация обеспечивает асинхронную передачу данных на запись с ведущего узла (master) на ведомые узлы (slave).

Ведомые узлы работают в режиме “только для чтения”.

Репликация ведомых узлов может выполняться как в синхронном, так и в асинхронном режиме.

При остановке ведущего узла - происходит его замена на один из ведомых.

При отказе ведущего узла - возможна потеря данных, так как запись происходит в асинхронном режиме.

Масштабирование

Multi-master репликация похожа на master-slave репликацию, за исключением наличия нескольких ведущих узлов.

Каждый ведущий узел обрабатывает входящий запрос, затем производит его синхронизацию на других ведущих серверах.

Недостаток данного вида репликации - возможность возникновения конфликтов между ведущими серверами на уровне транзакций.

В PostgreSQL используется асинхронная multi-master репликация.

Итоги

Итоги

PostgreSQL - одна из современных СУБД, позволяющая решать большой набор задач, при этом оставаясь производительной и отказоустойчивой.

В текущей лекции мы узнали, что PostgreSQL:

- CA система по теореме Брюера
- Соответствует принципу ACID
- Имеет 4 уровня изоляции
- Для восстановления использует WAL
- Позволяет расширять свой функционал
- Обладает достаточно высокой производительностью
- Имеет 4 уровня безопасности
- В обеспечение производительности и отказоустойчивости возможны гибкие настройки шардинга и репликации

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Роман Гордиенко

Роман Гордиенко