Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Распределенные системы

**Лабораторная работа 4.1**

**Сравнение подходов хранения больших данных**

Выполнил(а): Шведова С.С., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2024

**Цель работы:** сравнить производительность и эффективность различных подходов к хранению и обработке больших данных на примере реляционной базы данных PostgreSQL и документоориентированной базы данных MongoDB.

**Оборудование и программное обеспечение**

- Компьютер с операционной системой Ubuntu.

- PostgreSQL. - MongoDB. - Python 3.x.

- Библиотеки: psycopg2, pymongo, pandas, matplotlib

**Ход работы:**

Для удобства работы сначала нужно подключиться через удаленный рабочий стол к виртуальной машине. Для этого на рисунке 1 показано управление службой удаленного рабочего стола в системе Ubuntu.

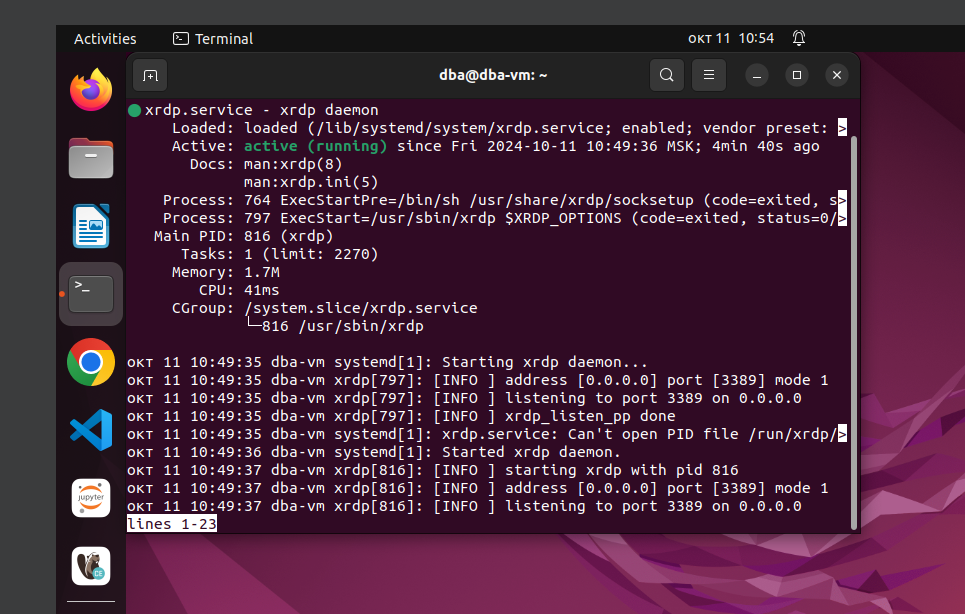


Рисунок 1. Управление службой удаленного рабочего стола в системе Ubuntu

Затем нужно зайти в настройки сети и узнать IP Adress, в данном случае он 172.20.10.2 (рисунок 2)

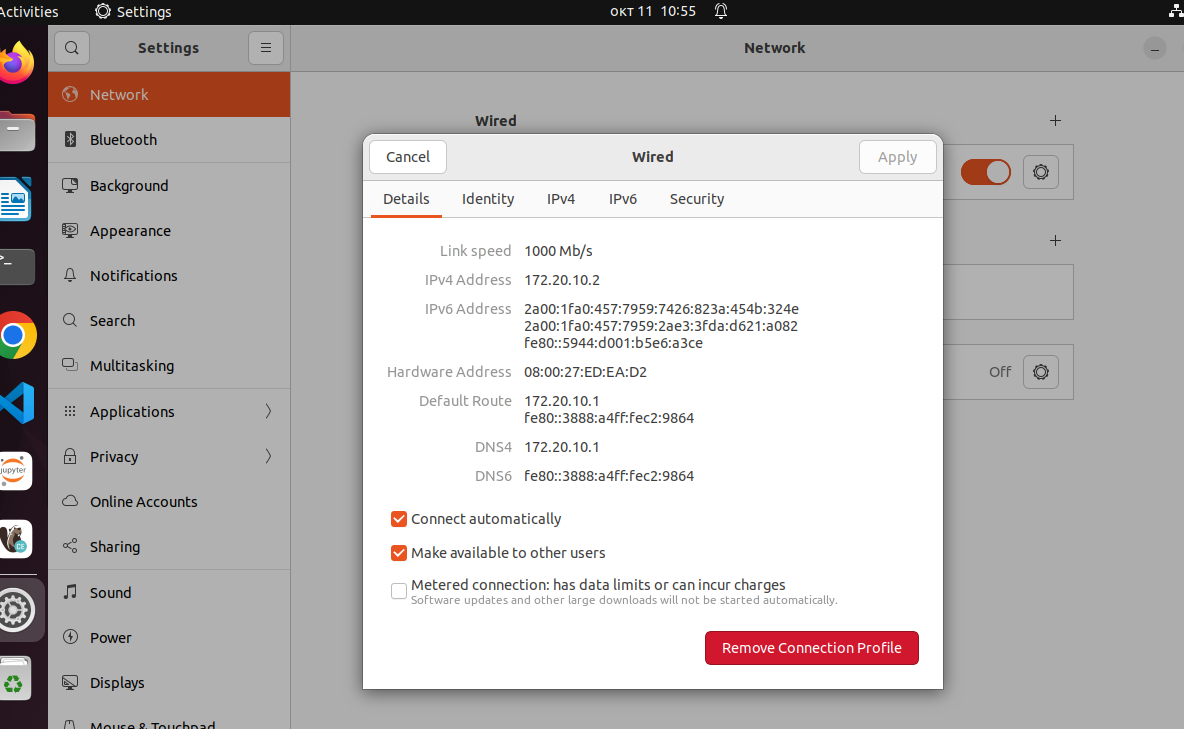


Рисунок 2. IP Adress

Далее нужно сверить с IP-адресом на компьютере, как можно увидеть, они совпадают (рисунок 3).

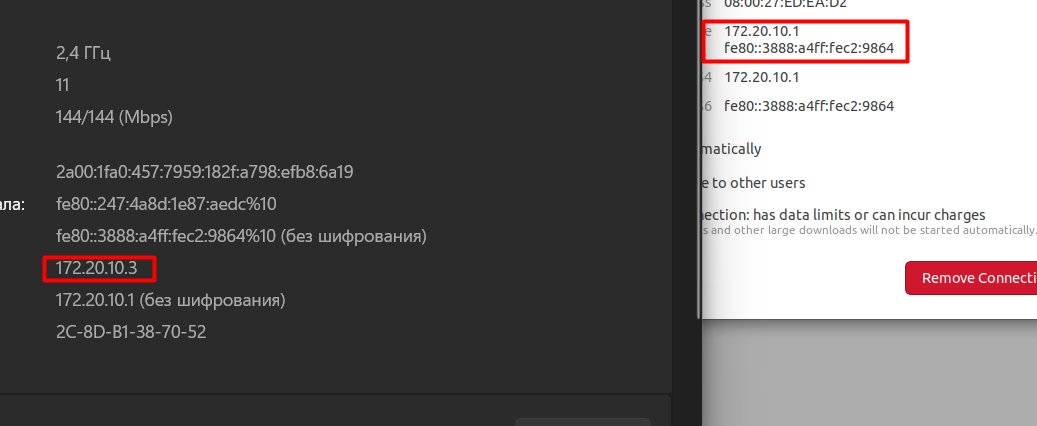


Рисунок 3. IP-адрес на компьютере

После того, как успешно подключились к удаленному рабочему столу к виртуальной машине, надо запустить контейнеры на основе настроек, определенных в файле docker-compose.ym (рисунок 4).

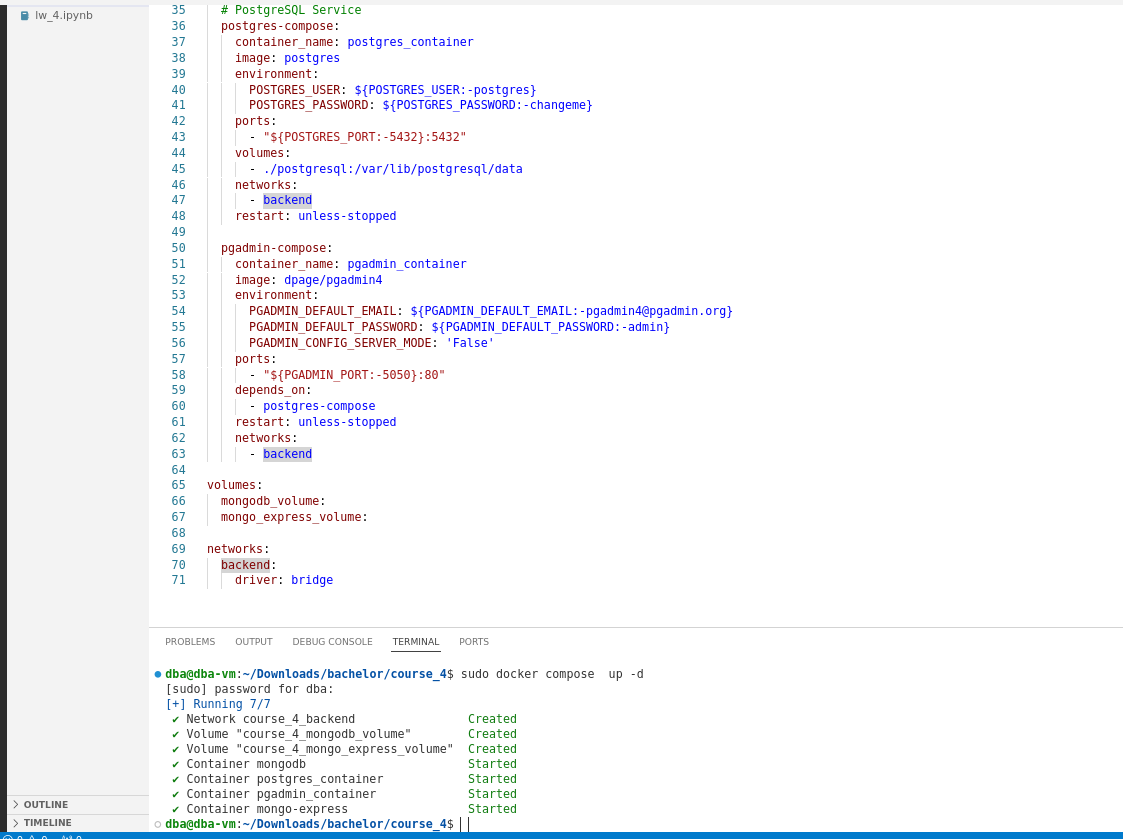


Рисунок 4. Запуск контейнеров

Теперь с помощью команды sudo docker inspect id\_conteiner надо узнать IP address сервера PgAdmin (рисунок 5).

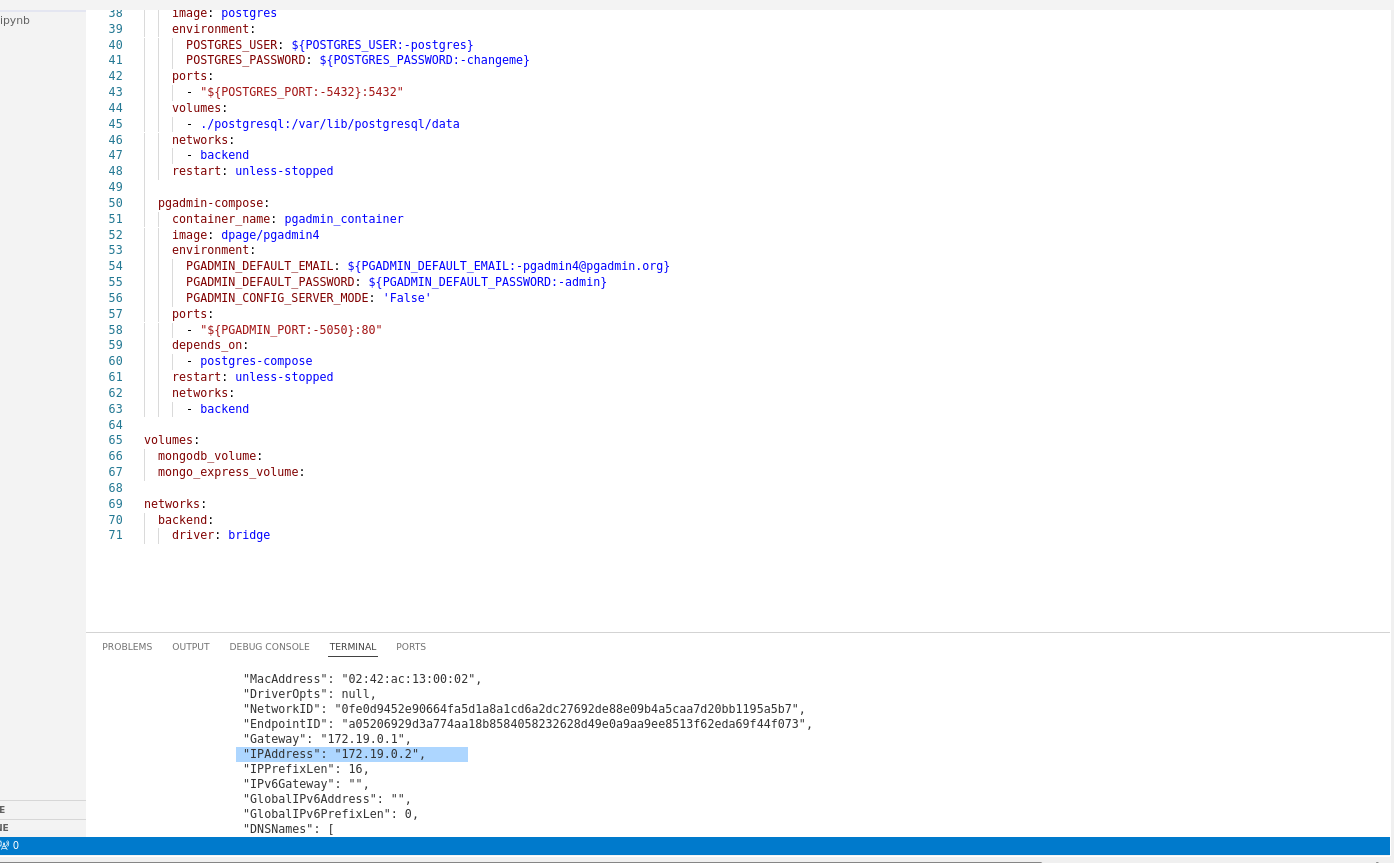


Рисунок 5. IP address сервера PgAdmin

После подключения к PgAdmin надо загрузить в pgAdmin 4 таблицу sales. Как можно увидеть, подключение прошло успешно (рисунок 6-7).

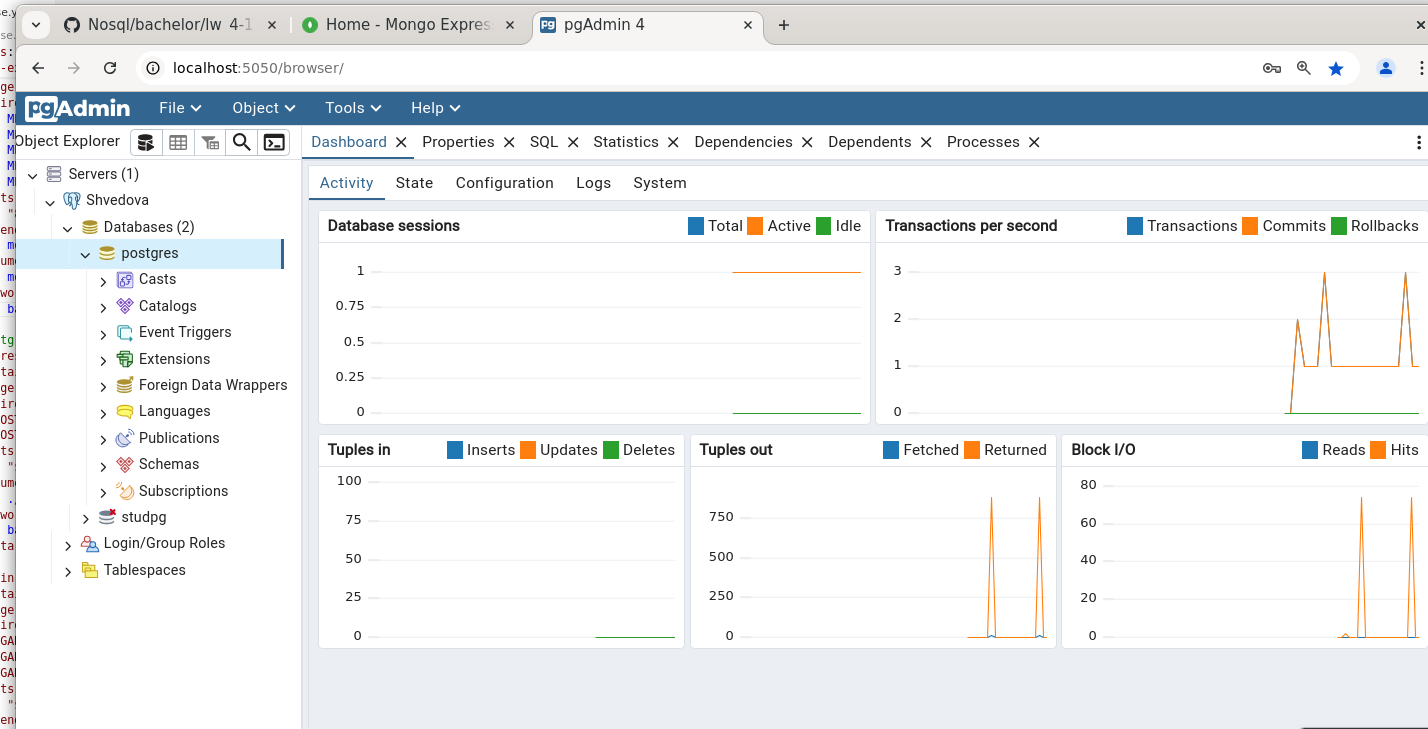


Рисунок 6. Успешное подключение к PgAdmin 4

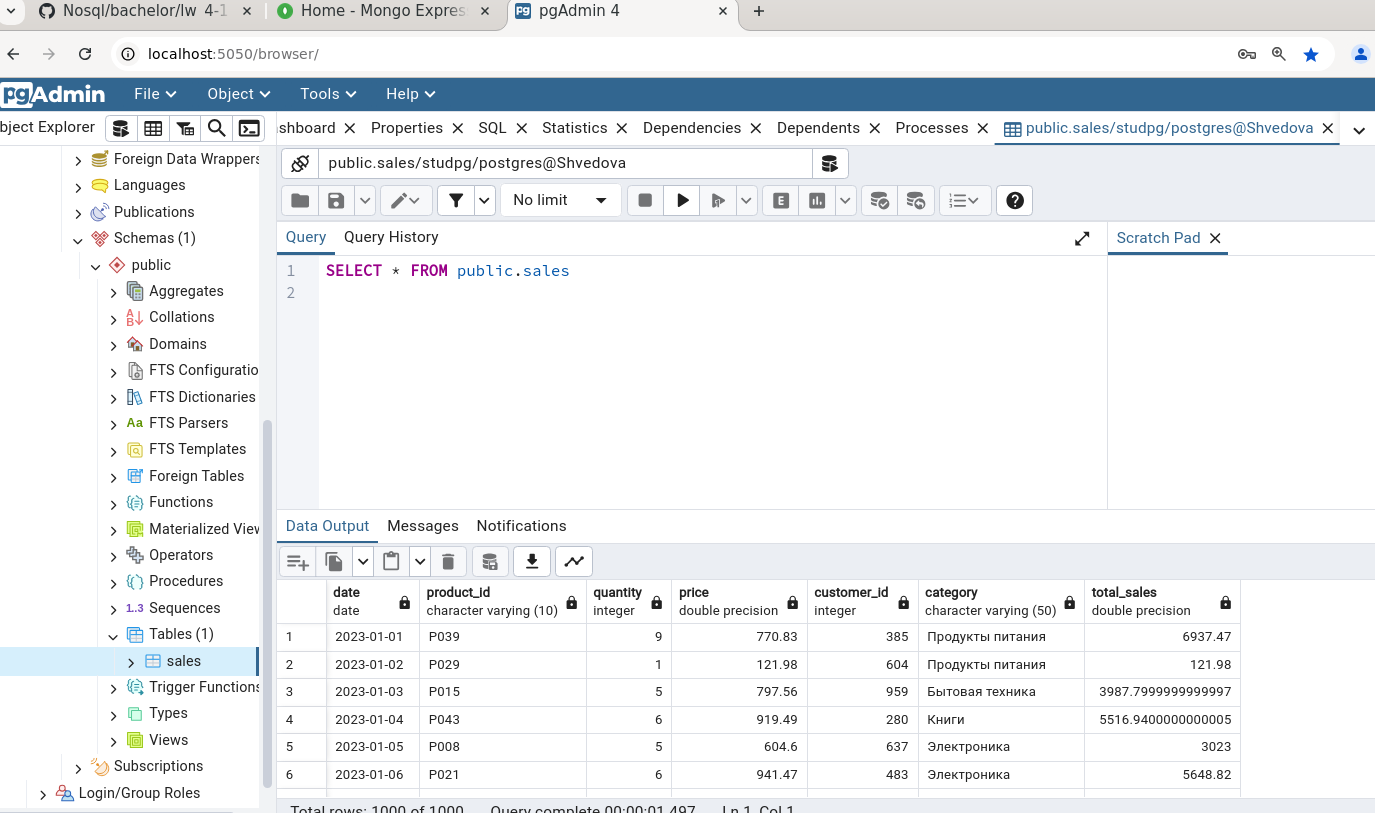


Рисунок 7. Добавление таблицы sales

**Индивидуальное задание**

**Вариант 13.** Исследовать возможности и эффективность шардинга в PostgreSQL и MongoDB на примере системы сбора и анализа данных с IoT-устройств.

Шардирование — это принцип проектирования базы данных, при котором данные разбиваются на части и размещаются на разных шардах. Каждый шард представляет собой отдельный узел внутри кластера, который может состоять из одной или нескольких реплик. Реплики — это серверы, на которых дублируются данные в рамках шарда.

Сначала необходимо сгенерировать данные с помощью представленного кода (рисунок 8).

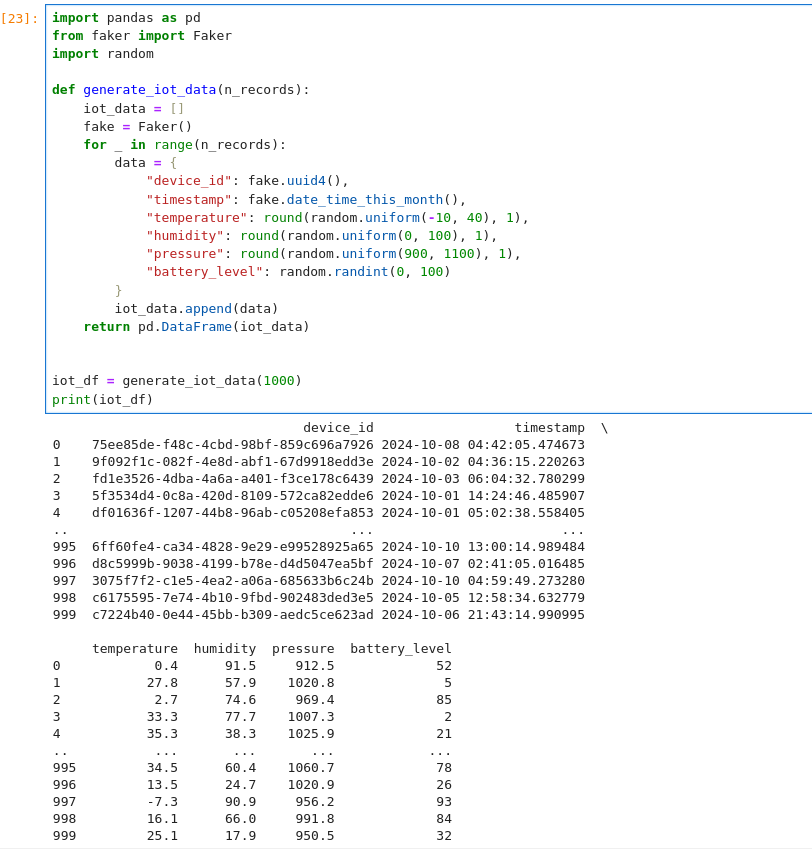


Рисунок 8. Сгенерированные данные

Затем необходимо загрузить данные в PgAdmin 4 (рисунок 9).

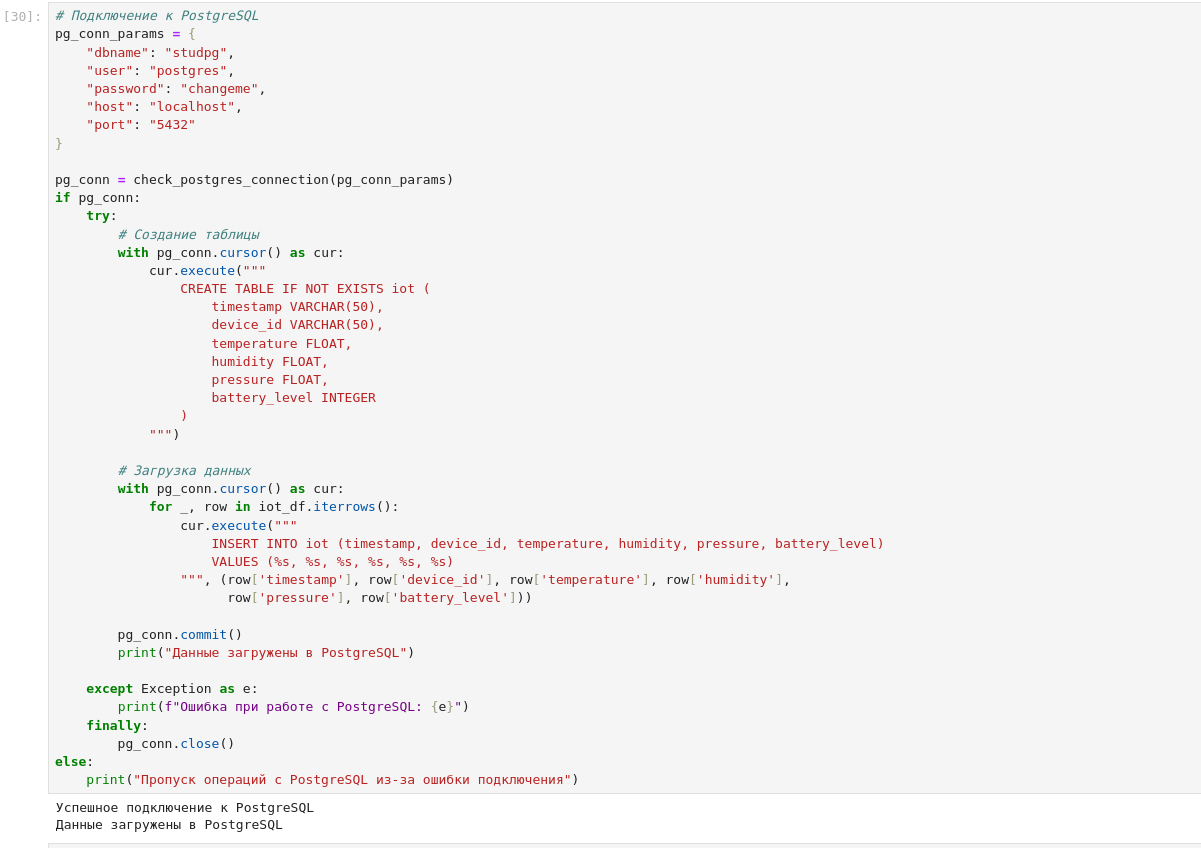


Рисунок 9. Загрузка данных в PgAdmin 4

Как можно заметить, таблица успешно загрузилась (рисунок 10).

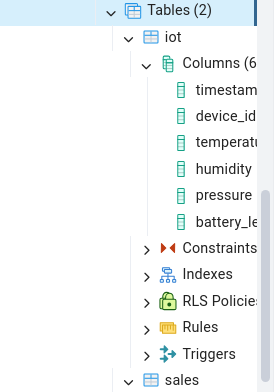


Рисунок 10. Таблица успешно загрузилась

Затем нужно разделить данные по device\_id на 10 таблиц, используя device\_id как shard\_key (рисунок 11).

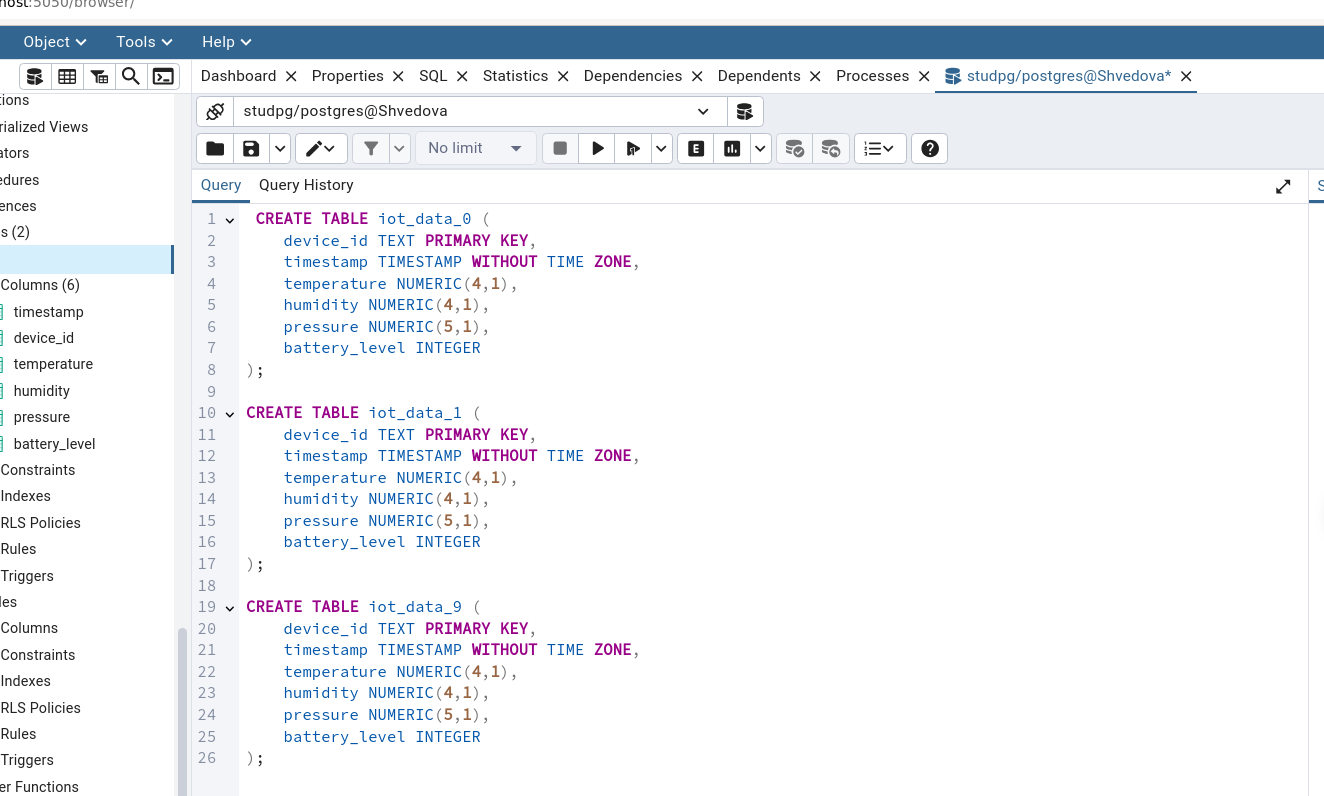


Рисунок 11. Создание таблиц-шардов

Далее нужно воспользоваться функцией для определения номера шара (эта функция предназначена для вычисления идентификатора шарда (shard ID) на основе значения, переданного в качестве аргумента device\_id (рисунок 12).

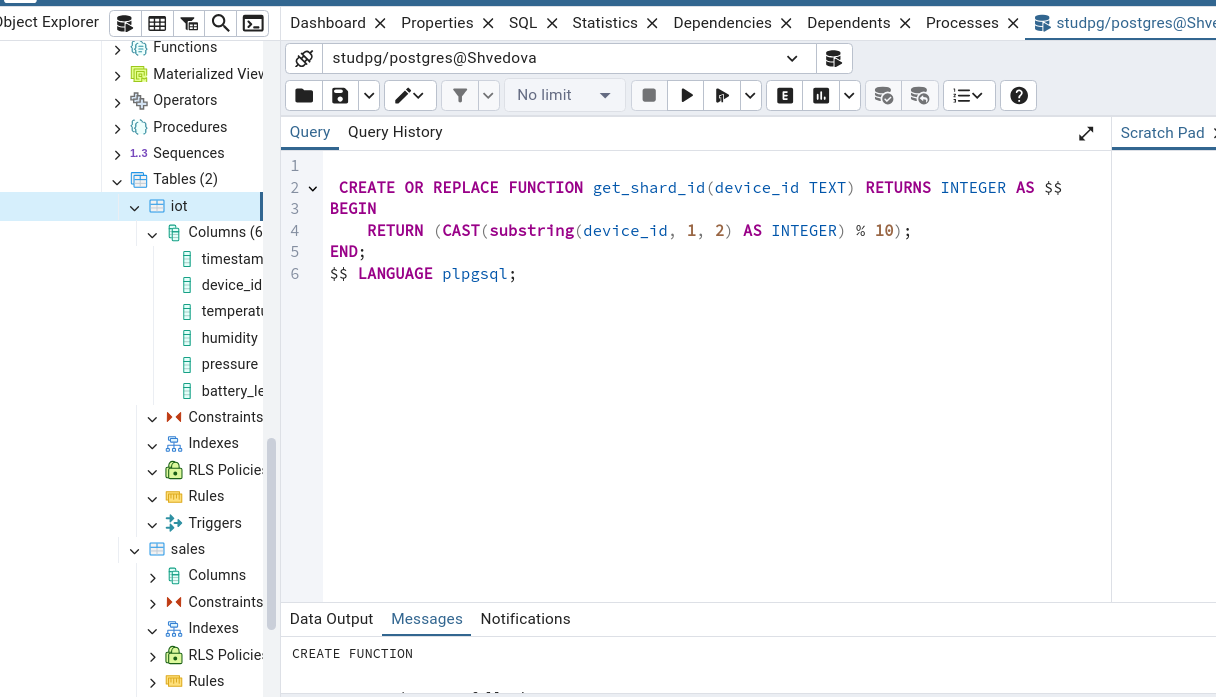


Рисунок 12. Функция для определения номера шарда

Пример использования функции показан на рисунке 13, который возвращает номер шарда по device\_id (в данном случае 5).

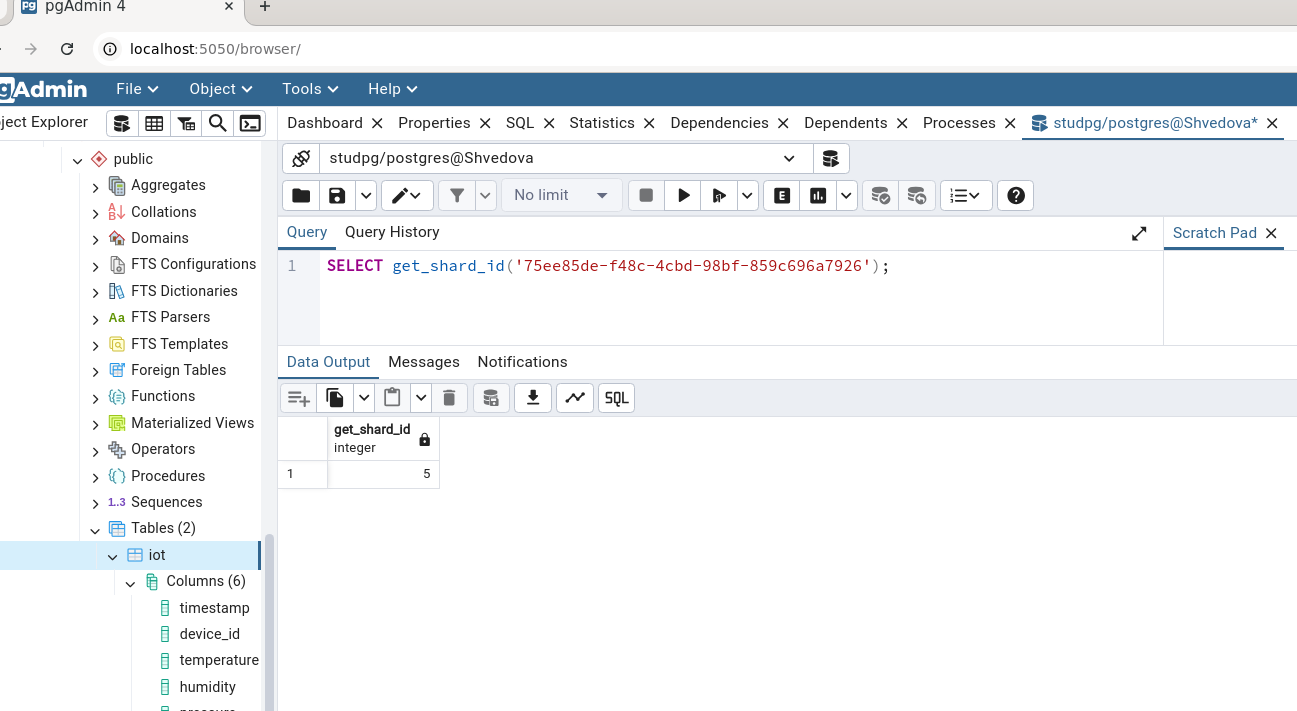


Рисунок 13. Пример использования функции

Вывод самих таблиц-шардов показано на рисунке 14.

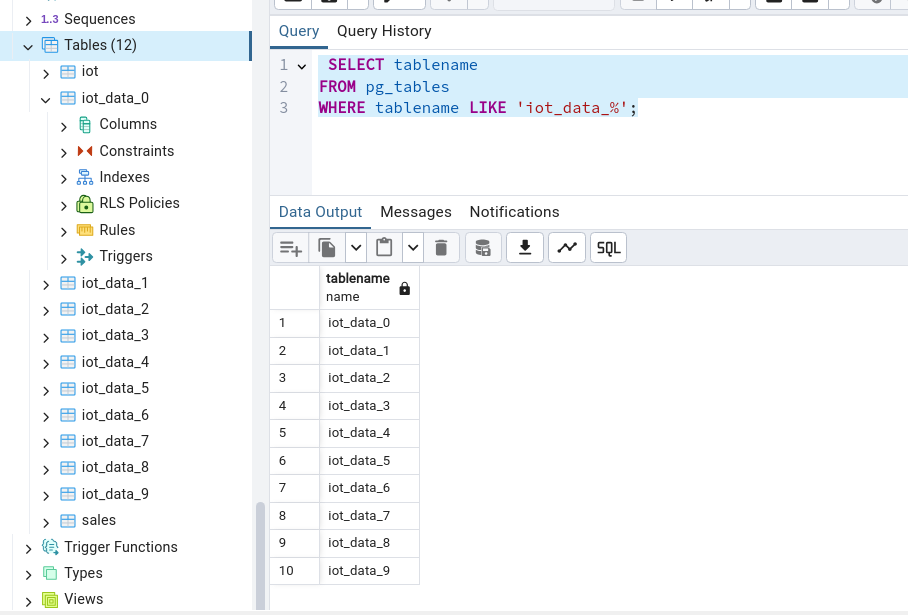


Рисунок 14. Таблицы-шарды

**Выводы:**

Преимущества шардирования:

1. Масштабируемость: шардирование позволяет эффективно обрабатывать большие объемы данных. Разделение данных на несколько таблиц-шардов снижает нагрузку на базу данных и повышает скорость запросов.

2. Улучшение производительности: запросы к конкретному шарду выполняются быстрее, чем запросы к большой таблице.

3. Упрощение операций: разделение данных на меньшие части делает операции с данными более простыми и эффективными.

Недостатки шардирования:

1. Сложность реализации: шардирование требует дополнительных усилий при разработке и поддержании системы.

2. Сложность запросов: необходимо учитывать шардирование при написании запросов, чтобы обращаться к правильным таблицам-шарам.

Теперь нужно провести шардирование в MongoDB. Для этого надо подключиться к MongoDB и загрузить туда данные (рисунок 15).

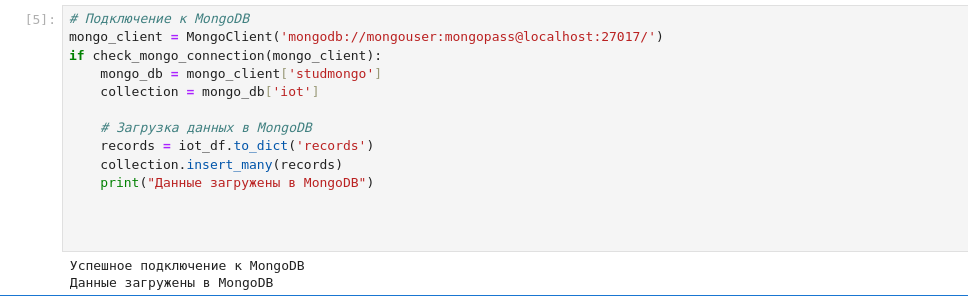


Рисунок 15. Загрузка данных в MongoDB

Как можно увидеть на рисунке 16, таблица успешно загрузилась.

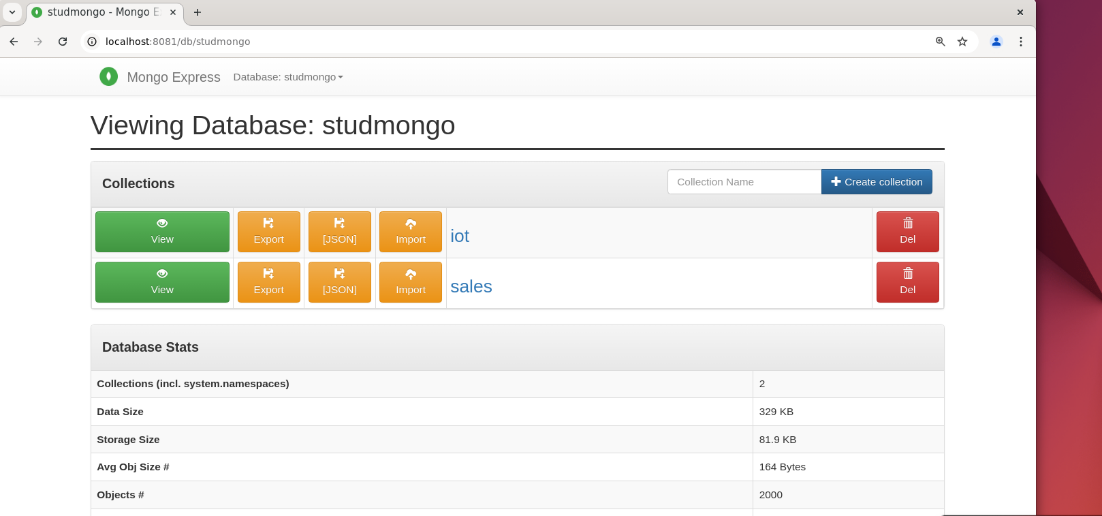


Рисунок 16. Таблица успешно загрузилась

Таблица в MongoDB Compass тоже отображается (рисунок 17).

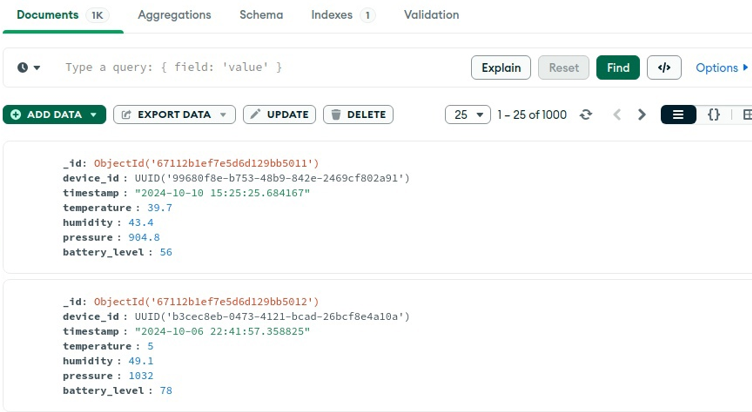


Рисунок 17. Таблица в MongoDB Compass

Шардирование в MongoDB — это стратегия горизонтального масштабирования данных, при которой части коллекций MongoDB размещаются на разных хостах кластера.

Для шардирования в MongoDB нужно создать 5 пустых директорий, три из которых - для конфигурационных серверов, в которых будут храниться данные, а другие два – для шардов (рисунок 18).

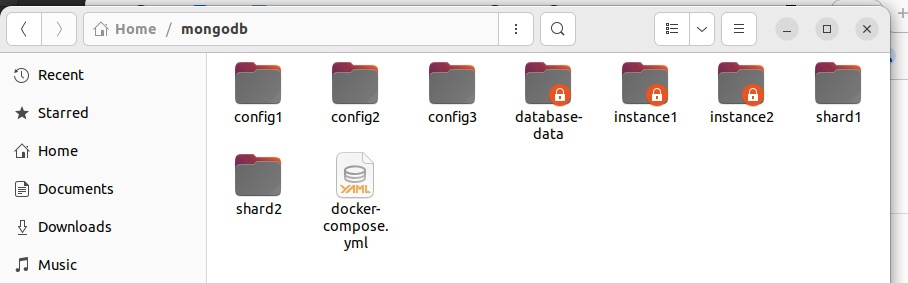


Рисунок 18. Создание пустых директорий

Теперь нужно запустить 3 конфигурационных сервера на разных портах и в разных терминалах (рисунок 19,20,21).

https://sun9-12.userapi.com/impg/0zJYTaSIjhUBaJZxuB9ncXdC-630AuwCiANGTw/He8BMhyH8YQ.jpg?size=757x62&quality=96&sign=8ce095d34b66f052eeca23ee8fb62f5a&type=album

Рисунок 19. Конфигурационный сервер на порту 27022

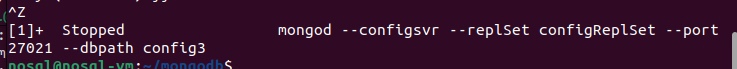


Рисунок 20. Конфигурационный сервер на порту 27021

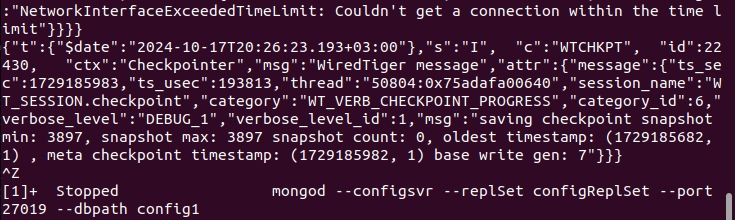


Рисунок 21. Конфигурационный сервер на порту 27019

Далее нужно инициализировать репликацию для конфигурационного сервера с помощью rs.initiate, перед этим написав команду mongosh –port 27019 (рисунок 22, 23).

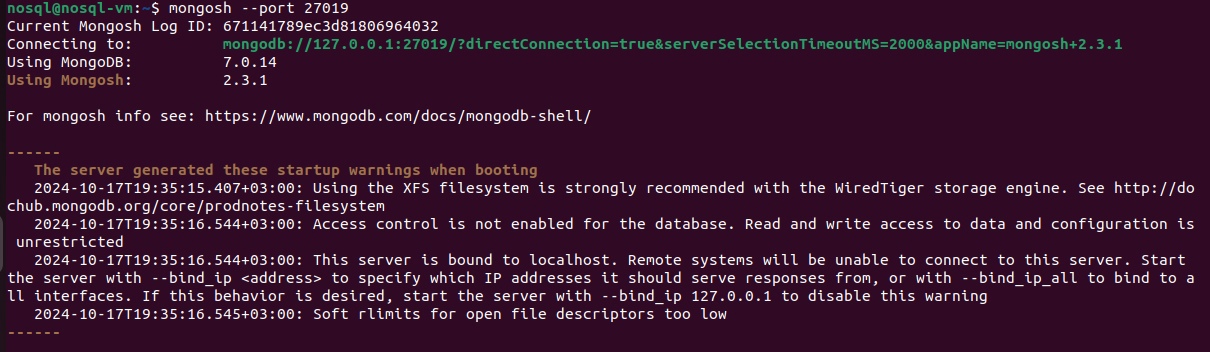


Рисунок 22. Команда mongosh –port 27019

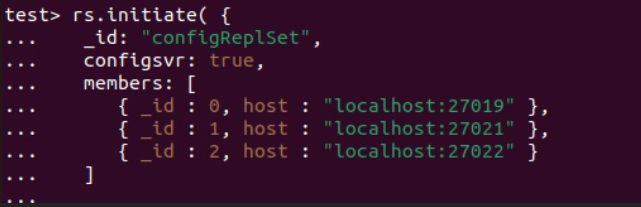


Рисунок 23. Инициализация репликации для конфигурационного сервера

Теперь нужно запустить каждый сервер MongoDB в режиме шарда (рисунок 24, 25).

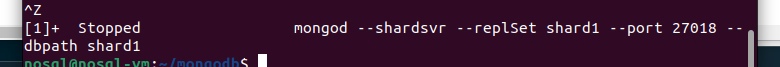


Рисунок 24. Запуск первого сервера в режиме шарда

https://sun9-1.userapi.com/impg/my7Qo-IxLtftKEGB0H8beLmIhPbfa2t6QhewWA/QvkSHF9KsmI.jpg?size=743x55&quality=96&sign=444e3d5b7b5abfbada8460af81cf805d&type=album

Рисунок 25. Запуск второго сервера в режиме шарда

Инициализация двух шардов показана на рисунках 26, 27.

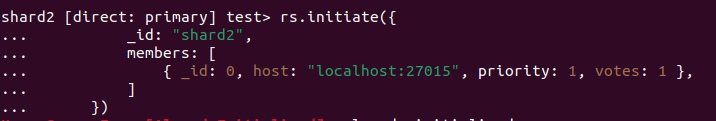


Рисунок 26. Инициализация первого шарда

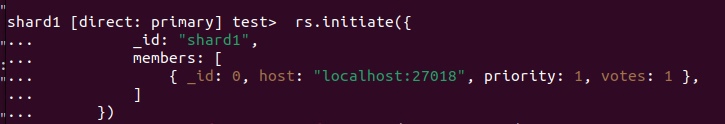


Рисунок 27. Инициализация второго шарда

Для добавления шардов shard1 и shard2 надо выполнить команду sh.addShard, которая продемонстрирована на рисунке 28.

https://sun9-3.userapi.com/impg/SlDDPENU0JrXej6JxDG0AK-kpzlTLNmgvHjvFw/NEUwjdXP7lU.jpg?size=690x22&quality=96&sign=7642b8131629d6aadc25f4a84a73405c&type=album

Рисунок 28. Добавление шардов

Теперь создаются ключи шардирования и применение их к коллекции iot (рисунок 29).

https://sun9-2.userapi.com/impg/HFm-XyqMlOu_QmIj1yfwn6Ufir_nnfDwucxNZw/qWMawckmOqI.jpg?size=880x57&quality=96&sign=6d8a6bec71d63f1922003b0b205df583&type=album

Рисунок 29. Ключи шардирования и применение к коллекции

После проведения шардирования данные сначала записываются в основной шард, а затем распределяются на второй шард, пока количество данных на шардах не станет равным. Диапазоны шардов могут при этом изменяться.

**Вывод:**

1. Горизонтальное масштабирование: шардинг в MongoDB позволяет горизонтально масштабировать базу данных путем распределения данных по нескольким серверам (шардам). Это позволяет обрабатывать большие объемы данных и улучшает производительность системы.

2. Улучшение производительности запросов: шардинг помогает улучшить производительность запросов, так как данные распределены между несколькими шардами, что позволяет параллельно выполнять операции над данными.

3. Отказоустойчивость: при использовании репликации в сочетании с шардингом, MongoDB обеспечивает отказоустойчивость, так как данные могут быть реплицированы и сохранены на нескольких узлах.

Сравнительная характеристика шардинга в PostreSQL и MongoDB представлена в таблице 1.

Таблица 1. Сравнительная характеристика

|  |  |  |
| --- | --- | --- |
|  | **MongoDB** | **PostreSQL** |
| **Преимущества** | 1. Обеспечивает высокую производительность при работе с большими объемами неструктурированных данных, что может быть полезно для хранения данных от IoT-устройств.  2. Возможность гибкого определения ключа шардирования позволяет учитывать специфику данных IoT и обеспечивать эффективное их распределение. | 1. Обладает богатыми возможностями работы с транзакциями, SQL-запросами и поддержкой различных типов данных, что полезно для комплексного анализа данных от IoT-устройств.  2. Горизонтальный шардинг позволяет обрабатывать большие объемы данных и распределять нагрузку на несколько серверов, что существенно улучшает производительность. |
| **Возможности** | 1. Имеет встроенную поддержку горизонтального масштабирования и шардинга. Данные могут быть автоматически разделены между несколькими шардами на основе ключа шардирования.  2. Встроенный механизм балансировки данных позволяет распределять данные и запросы между шардами для оптимальной производительности. | 1. PostgreSQL поддерживает горизонтальный шардинг через расширение, такое как pg\_shard или Citus. Они позволяют распределить данные по разным серверам и объединять результаты запросов.  2. Распределенные таблицы позволяют распределять данные между несколькими узлами.  3.Возможность использовать горизонтальный шардинг в сочетании с репликацией, обеспечивая отказоустойчивость и масштабируемость. |

**Общий вывод**:

Обе базы данных, PostgreSQL и MongoDB, предоставляют возможности для настройки шардинга и горизонтального масштабирования, что делает их подходящими для систем сбора и анализа данных с IoT-устройств, но при этом в PostgreSQL это реализуется легче, так как не надо поднимать сервера и требуется только знание SQL.