

# Реляционные базы данных: Масштабирование баз данных



Сергей  
Андрюнин



**Сергей Андрюнин**

DevOps инженер, RTLabs

---

# Предисловие

Сегодня мы:

- поговорим о **масштабировании** баз данных;
- разберем, какие методы и технологии **партиционирования и шардинга** бывают и как они работают;
- поговорим о методах репликации сторонними продуктами и технологиями.





# План занятия

1. [Масштабирование](#)
2. [Горизонтальное масштабирование \(scaling out\)](#)
3. [Шардинг \(shard – сегментировать\)](#)
4. [Вертикальный шардинг](#)
5. [Горизонтальный шардинг](#)
6. [SAN-кластер](#)
7. [Итоги](#)
8. [Домашнее задание](#)



# Масштабирование

---

# Масштабирование

**Масштабирование** — это процесс разделения данных на группы и выделение их на отдельные сервера.

**Основные виды масштабирования:**

- репликация (Master-slave, master-master),
- партиционирование,
- шардинг.


**Кластеризация** — это как **репликация** с каким-либо методом масштабирования.

---

# Масштабирование

Типы масштабирования:

- **scaling up** — масштабирование вверх,
- **scaling out** — масштабирование горизонтальное,
- **scaling back** — часть данных открыты, часть данных заархивированы,
- **federation** — доступ к удаленным данным.



# Горизонтальное масштабирование (scaling out)





## Scaling out

При **масштабировании** по горизонтали данные распределяются по нескольким серверам с помощью репликации, а далее используются slave-серверы на чтение. При этом происходит разделение (partition) данных по нескольким **нодам**.

**Нода** – функциональный блок в MySQL, это может быть отдельный сервер.

# Scaling out

**Ноды бывают четырех типов:**

- активный master-сервер и пассивный репликационный slave-сервер (настраивали),
- master-сервер и несколько slave-серверов (настраивали),
- активный сервер со специальным механизмом репликации – distributed replicated block device (DRBD),
- SAN-кластер.



## Scaling out

**Функциональное разделение** — данные разбиваются на таблицы так, что они никогда не соединяются между собой (**портирование**).

**Data sharding** — механическое разделение огромных объемов однотипных данных на несколько частей (**shard — шардинг**). С точки зрения реализации, это наиболее трудоемкий вариант.



**Шардинг (shard —  
сегментировать)**

# Шардинг

**Шардинг (иногда шардирование)** — это техника масштабирования при работе с данными. Его суть в разделении (партиционирование) базы данных на отдельные части так, чтобы каждую из них можно было вынести на отдельный сервер.

Существует два вида шардинга:

- вертикальный;
- горизонтальный.

# Шардинг

**Вертикальный шардинг** — это выделение таблицы или группы таблиц на отдельный сервер.

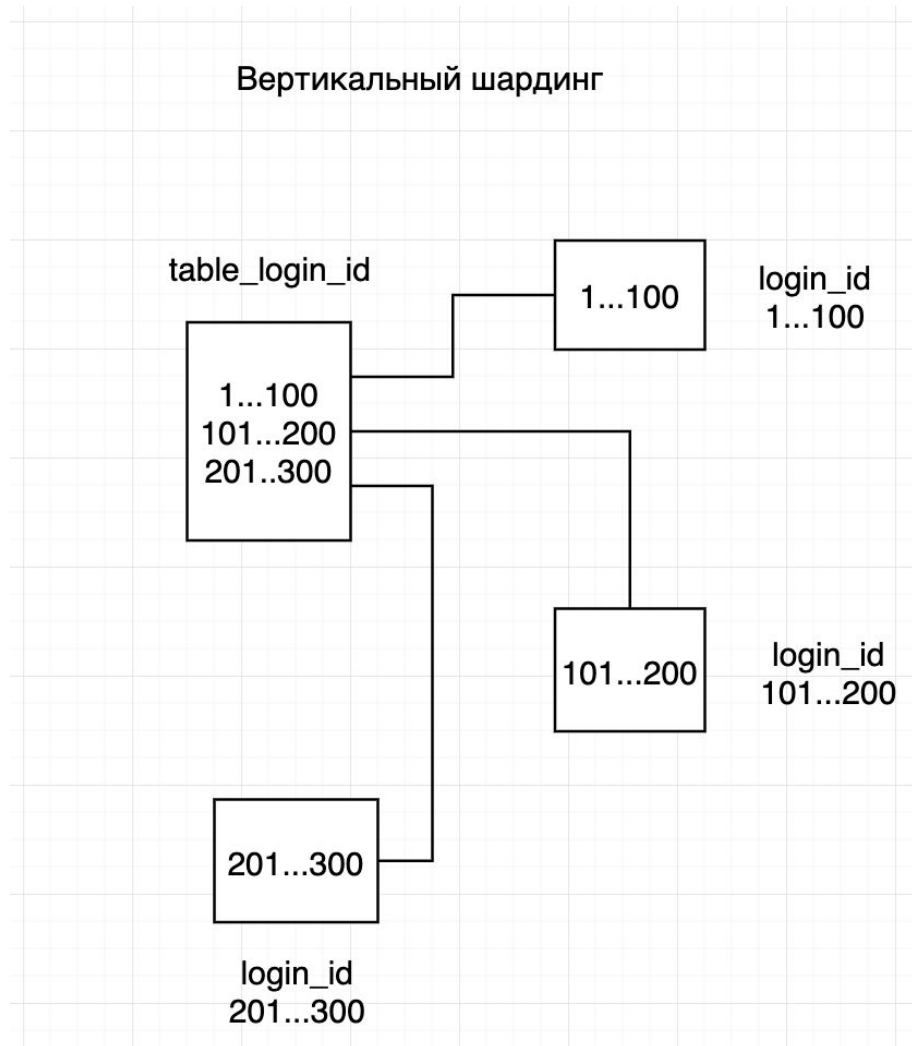
Например, существует две таблицы **users** и **password**. Таблицу **users** оставляем на одном сервере, а таблицу **password** переводим на другой.

Пример:

```
$users_connection = mysql_connect('name_host_node1', 'root', 'pwd');  
$password_connection = mysql_connect('name_host_node2', 'root', 'pwd');
```

Для каждой таблицы или группы таблиц будет отдельное соединение.

# Шардинг



# Шардинг

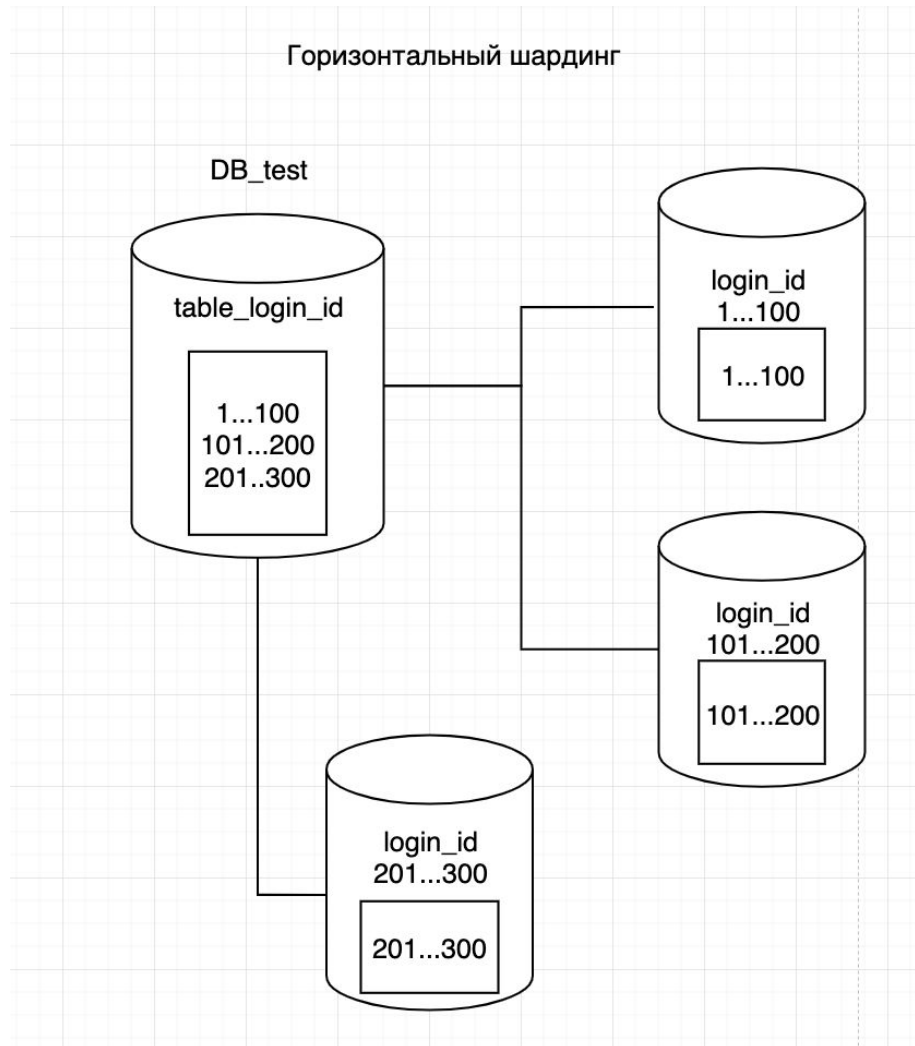
**Горизонтальный шардинг** — это разделение одной таблицы на разные сервера. Это необходимо использовать для огромных таблиц, которые не уместятся на одном сервере.

**Разделение таблицы на части делается по следующему принципу:**

- на нескольких серверах создается одна и та же таблица (только структура, без данных);
- в приложении выбирается условие, по которому будет определяться нужное соединение (например, четные на один сервер, а нечетные — на другой);
- перед каждым обращением к таблице происходит выбор нужного соединения.



# Шардинг





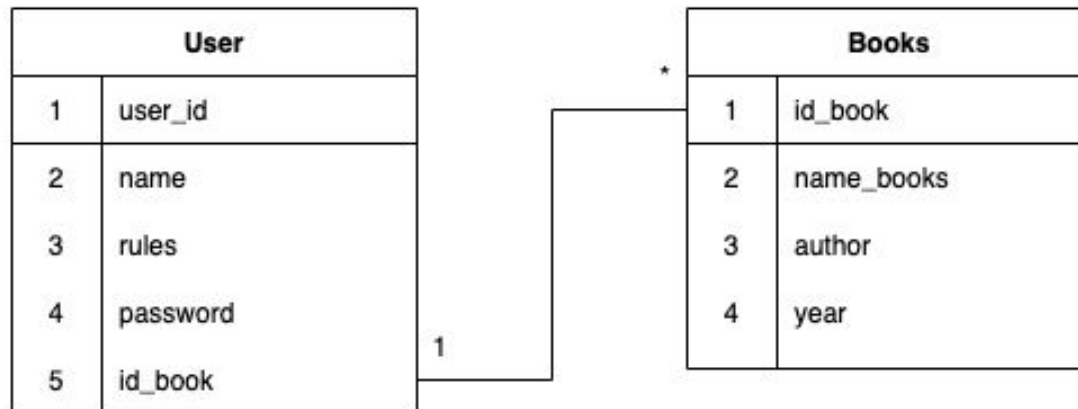
# Вертикальный шардинг

# Вертикальный шардинг

Для тестовой модели представим, что существует база данных из следующих таблиц:

**user** ( user\_id, name, rules, password, id\_books)

**books** (id\_books, name\_books, author, year)



# Вертикальный шардинг

Создание базы данных можно выполнить:

```
CREATE DATABASE library;
```

Создание таблиц user:

```
CREATE TABLE USER (  
  id INT(6) PRIMARY KEY,  
  username VARCHAR(30) NOT NULL,  
  password VARCHAR(30) NOT NULL,  
  book_id INT(6)  
);
```

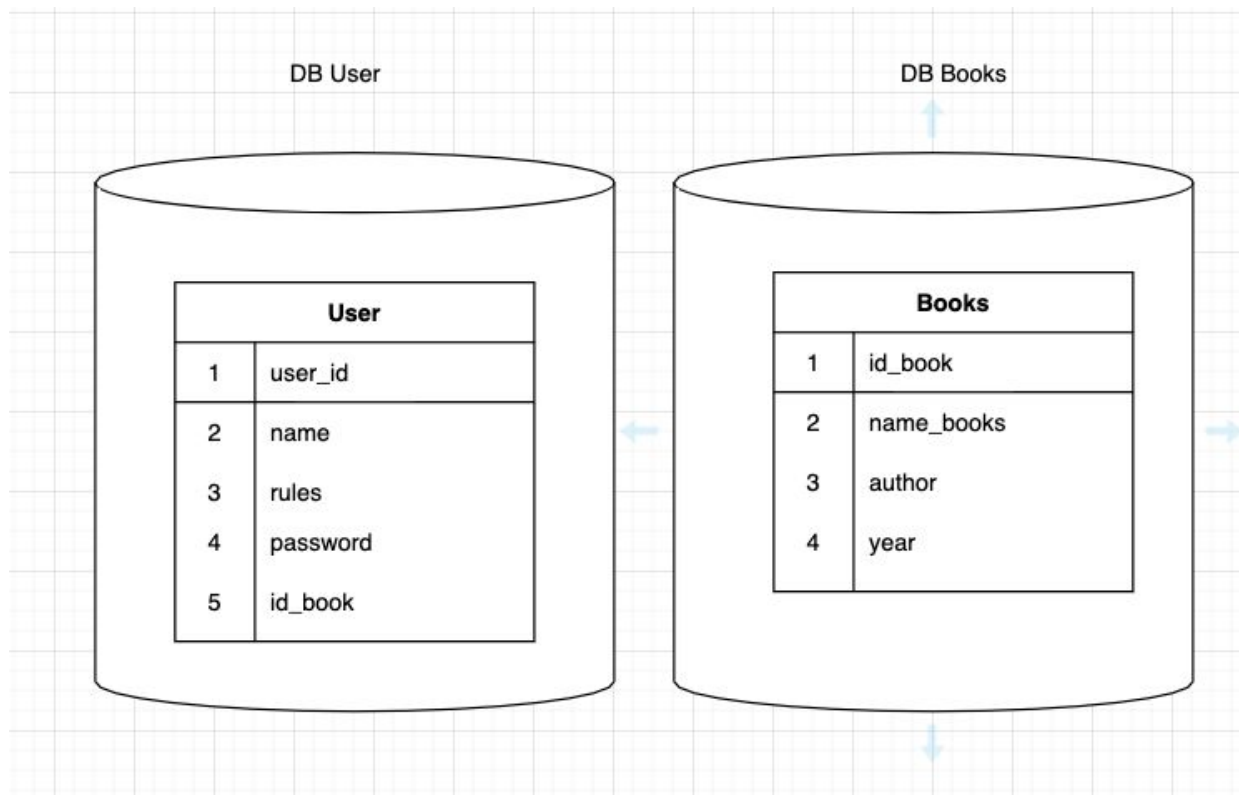
# Вертикальный шардинг

Создание таблиц books:

```
CREATE TABLE BOOKS (  
  book_id INT(6) PRIMARY KEY,  
  name VARCHAR(30) NOT NULL,  
  author VARCHAR(30) NOT NULL,  
  year INT(4)  
);
```

# Вертикальный шардинг

Выполним логическое разбиение одной базы данных на две:



# Вертикальный шардинг

Запустим два контейнера при помощи docker-compose, для этого создадим docker-compose.yml следующего содержания:

```
version: "3"
services:
  db_one:
    image: mysql
    container_name: db_one
    ports:
      - 3001:3306
    environment:
      - MYSQL_ROOT_PASSWORD=pass
    networks:
      db_network:
        ipv4_address: 172.20.0.10
    restart: always
  db_two:
    image: mysql
    container_name: db_two
    ports:
      - 3002:3306
    environment:
      - MYSQL_ROOT_PASSWORD=pass
    networks:
      db_network:
        ipv4_address: 172.20.0.11
    restart: always
networks:
  db_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/24
```

## Вертикальный шардинг

Создадим базу данных на каждом из экземпляров. Чтобы зайти в экземпляр достаточно выполнить команду для первого машины и второй:

```
#Для первой
```

```
docker-compose exec db_one mysql -u'root' -p'pass'
```

```
#Для второй
```

```
docker-compose exec db_one mysql -u'root' -p'pass'
```





## Вертикальный шардинг

Создаем таблицы USER для db\_one и BOOKS для db\_two.

Команды создания приведены на слайде 19-20.

В итоге у нас получилось создать вертикальный шардинг с разделением данных по различным источникам.

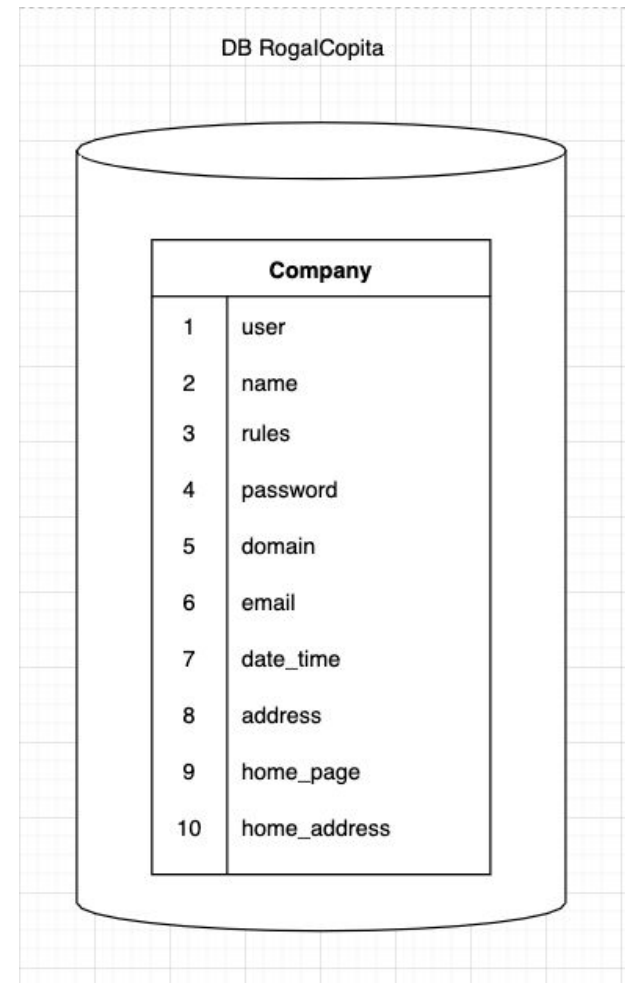


# Горизонтальный шардинг

# Горизонтальный шардинг

Для тестовой модели представим, что существует база данных из следующей таблицы:

company (user, name, rules, password, domain, email, date\_time, address, home\_page, home\_address)



# Горизонтальный шардинг

Создадим базу данных:

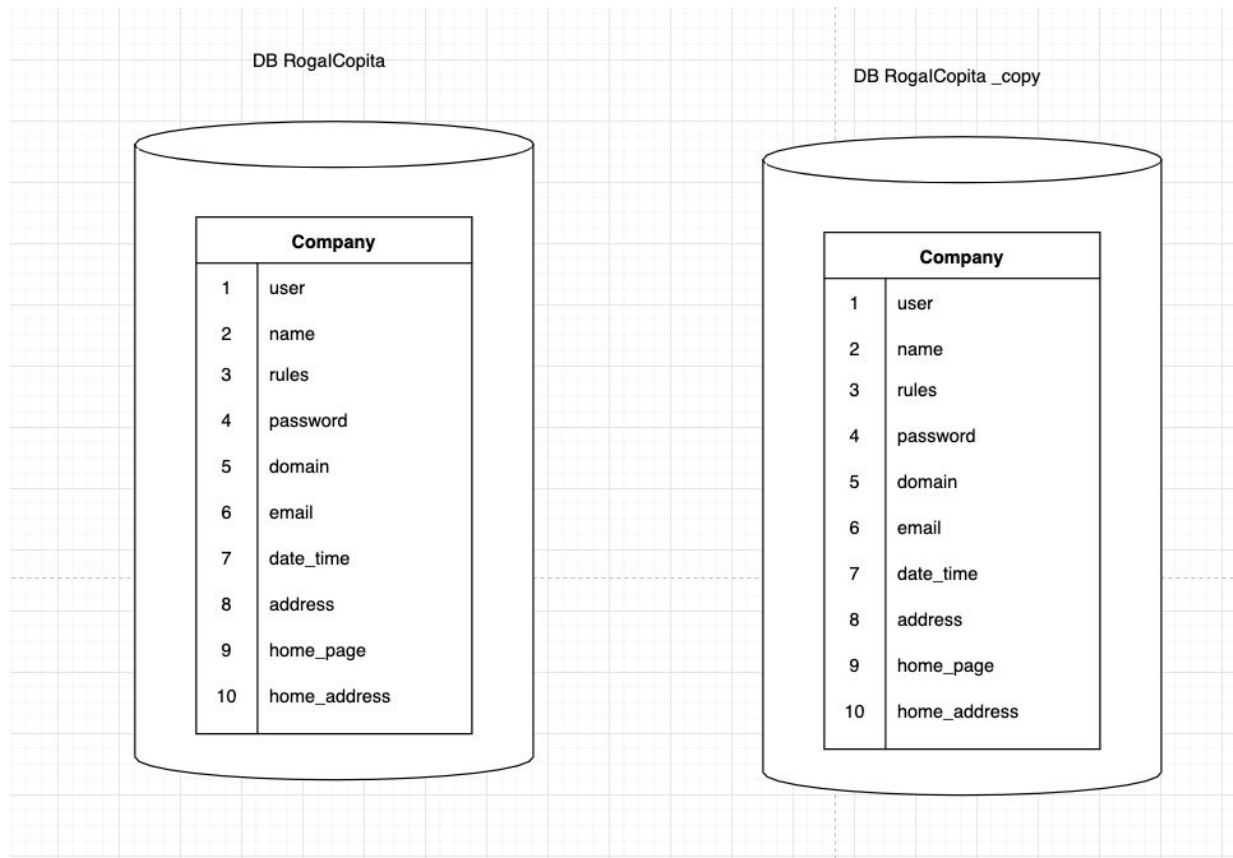
```
CREATE DATABASE DB RogaICopita;
```

Создадим таблицу user:

```
CREATE TABLE COMPANY (  
  user VARCHAR(30) NOT NULL,  
  name VARCHAR(60) NOT NULL,  
  rules VARCHAR(30),  
  password VARCHAR(30) NOT NULL,  
  domain VARCHAR(30),  
  email VARCHAR(30),  
  date_time DATE,  
  address VARCHAR(30),  
  home_page VARCHAR(30),  
  home_address VARCHAR(30))  
;
```

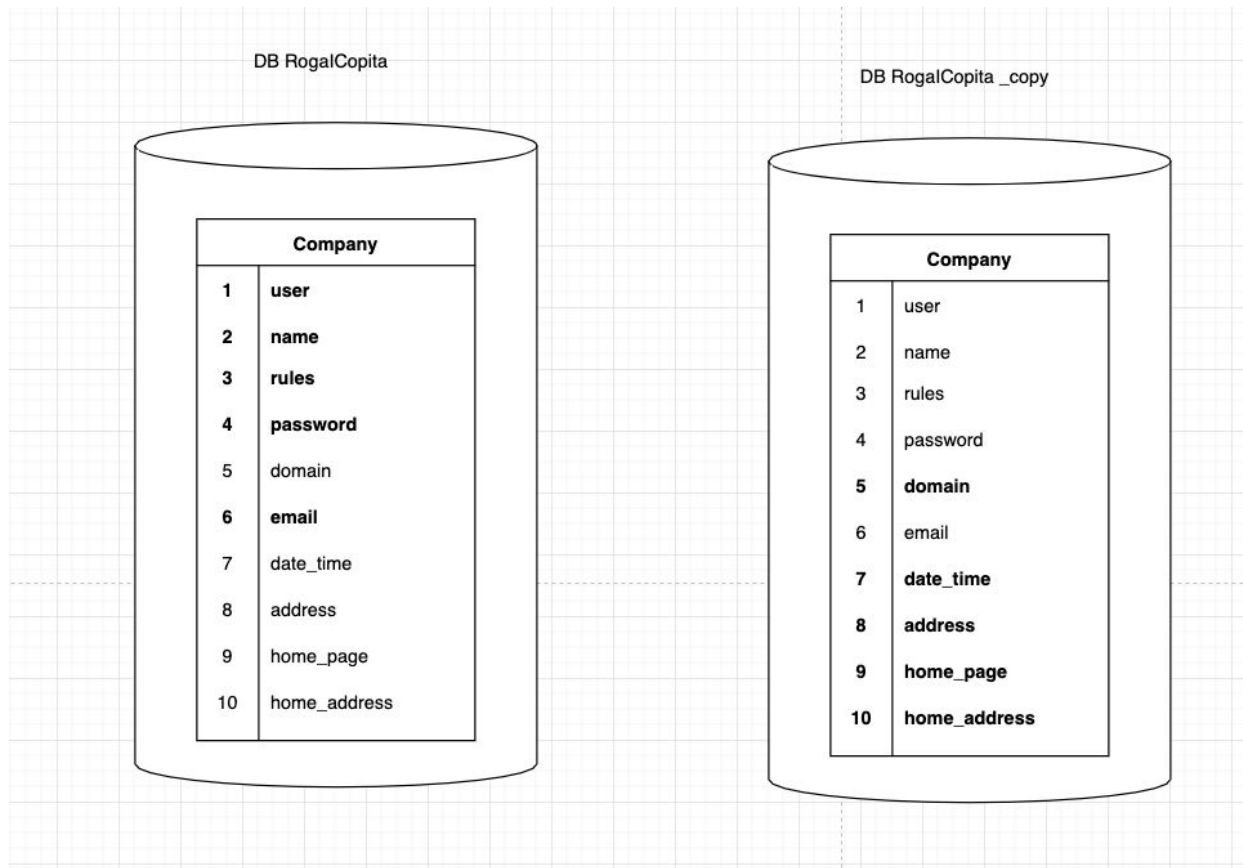
# Горизонтальный шардинг

Для примера использования горизонтального шардинга сделаем копию db:



# Горизонтальный шардинг

Выделим данные которые будут хранится в одной базе данных и в другой базе данных:



# Горизонтальный шардинг

Запустим два контейнера при помощи docker-compose, для этого создадим docker-compose.yml следующего содержания:

```
version: "3"
services:
  db_one:
    image: mysql
    container_name: db_one
    ports:
      - 3003:3306
    environment:
      - MYSQL_ROOT_PASSWORD=pass
    networks:
      db_network:
        ipv4_address: 172.20.0.12
    restart: always
  db_two:
    image: mysql
    container_name: db_two
    ports:
      - 3004:3306
    environment:
      - MYSQL_ROOT_PASSWORD=pass
    networks:
      db_network:
        ipv4_address: 172.20.0.13
    restart: always
networks:
  db_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/24
```

# Горизонтальный шардинг

Создадим базу данных на каждом из экземпляров согласно модели. Чтобы зайти в экземпляр достаточно выполнить команду для первой машины и второй:

```
#Для первой
```

```
docker-compose exec db_one mysql -u'root' -p'pass'
```

```
#Для второй
```

```
docker-compose exec db_one mysql -u'root' -p'pass'
```





# **SAN-кластер**



# Storage Area Network

Сеть хранения данных представляет собой архитектурное решение для подключения внешних устройств хранения данных (дисковые массивы, ленточные библиотеки) к серверам таким образом, чтобы операционная система распознала подключённые ресурсы как локальные.



# Итоги

---

# Итоги

## Сегодня мы:

- рассмотрели принципы масштабирования, шардинга, SAN;
- поговорили о вертикальном и горизонтальном масштабировании;
- настроили масштабирование и шардирование на примере MySQL.



# Домашнее задание



## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

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

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Сергей Андрюнин**