Моделирование высоконагруженной системы

Студент: Соколов Е.М., ИУ7-63Б
Научный руководитель: Гаврилова Ю.М.

Москва, 2021г.

Цель работы и постановка задачи

Цель работы

Моделирование высоконагруженной системы с применением механизмов и алгоритмом масштабирования базы данных.

Постановка задачи

- провести анализ существующих механизмов масштабирования БД в высоконагруженных системах;
- реализовать выбранные алгоритмы и механизмы в рамках системы, агрегирующей данные об авиаперелетах;
- предусмотреть возможность только суперпользователем вносить изменения в данные;
- провести нагрузочное тестирование разработанной системы.

Классификация СУБД

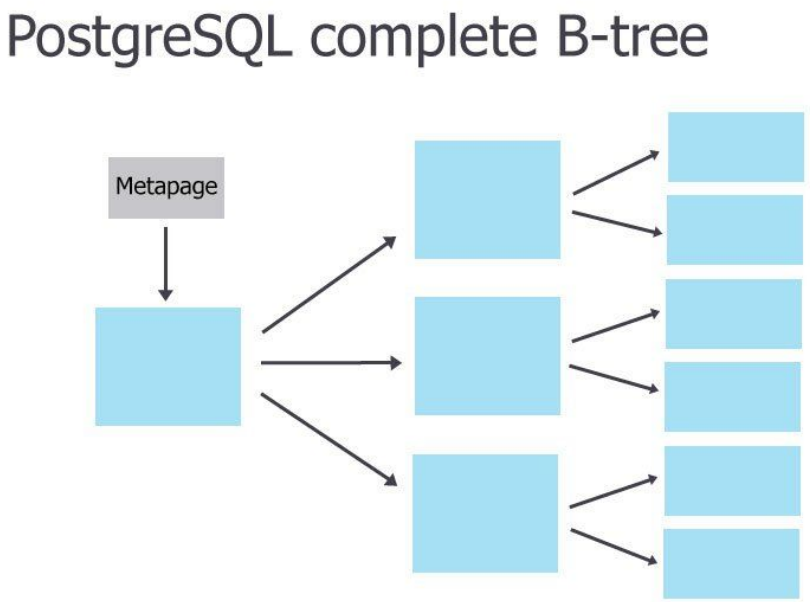
- по способу хранения:
  - построчно;
  - поколоночно;

- по модели данных:
  - реляционные (SQL) - PostgreSQL, Oracle Database, MS SQL Server;
  - нереляционные (NoSQL) - Redis, MongoDB, Giraph.

Механизмы высоконагруженной системы

- Индексирование атрибутов;
- кэширование запросов;
- репликация (Master-Slave, Master-Master);
- шардирование данных.

Индексирование



Дерево поиска PostgreSQL

```
CREATE INDEX IF NOT EXISTS idx_ac_tailno
ON aircrafts(
    tail_no
);
```

Пример создания индекса в СУБД PostgreSQL

Кэширование

Алгоритмы кэширования:

- LRU;
- 2Q.

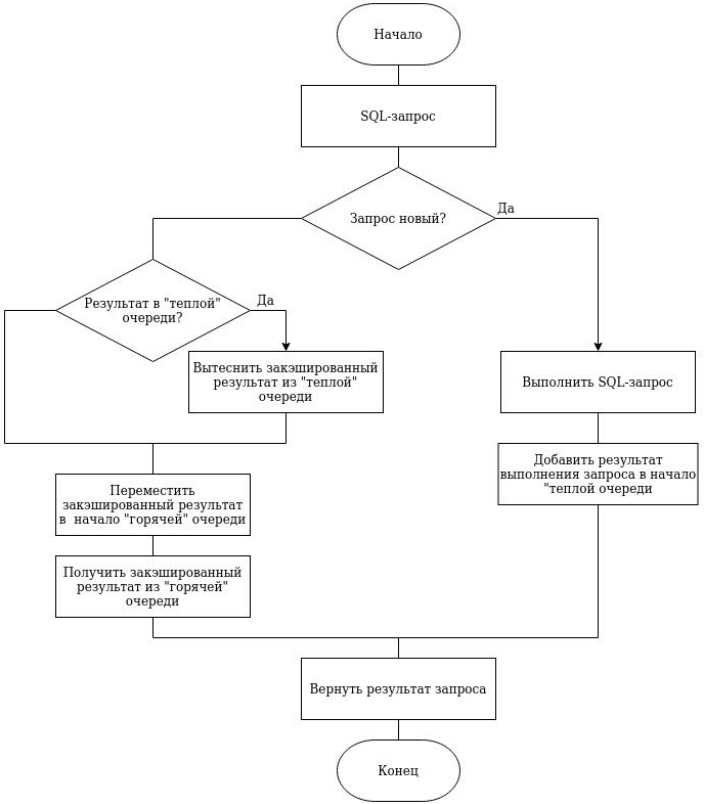


Схема алгоритма 2Q

Master-Slave репликация

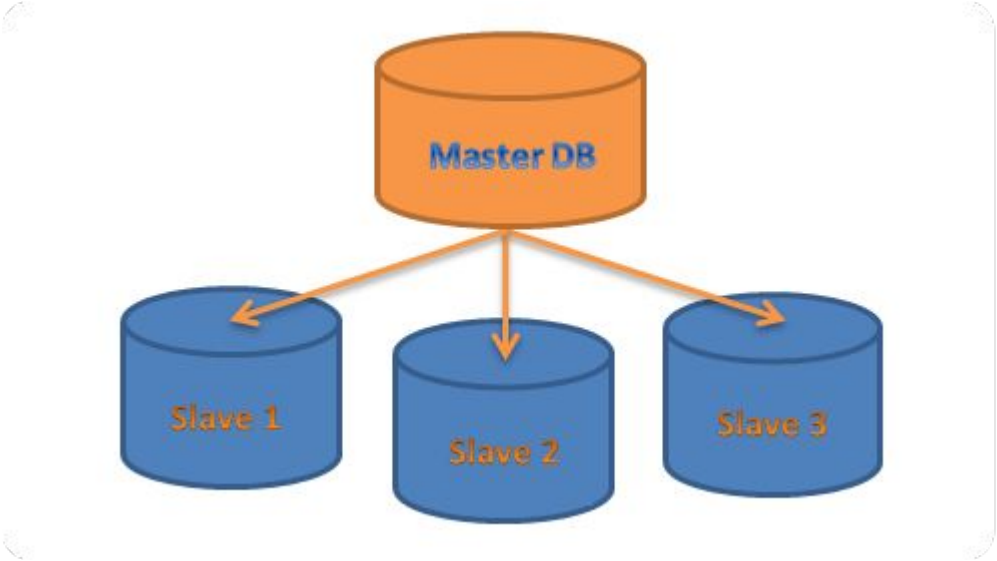


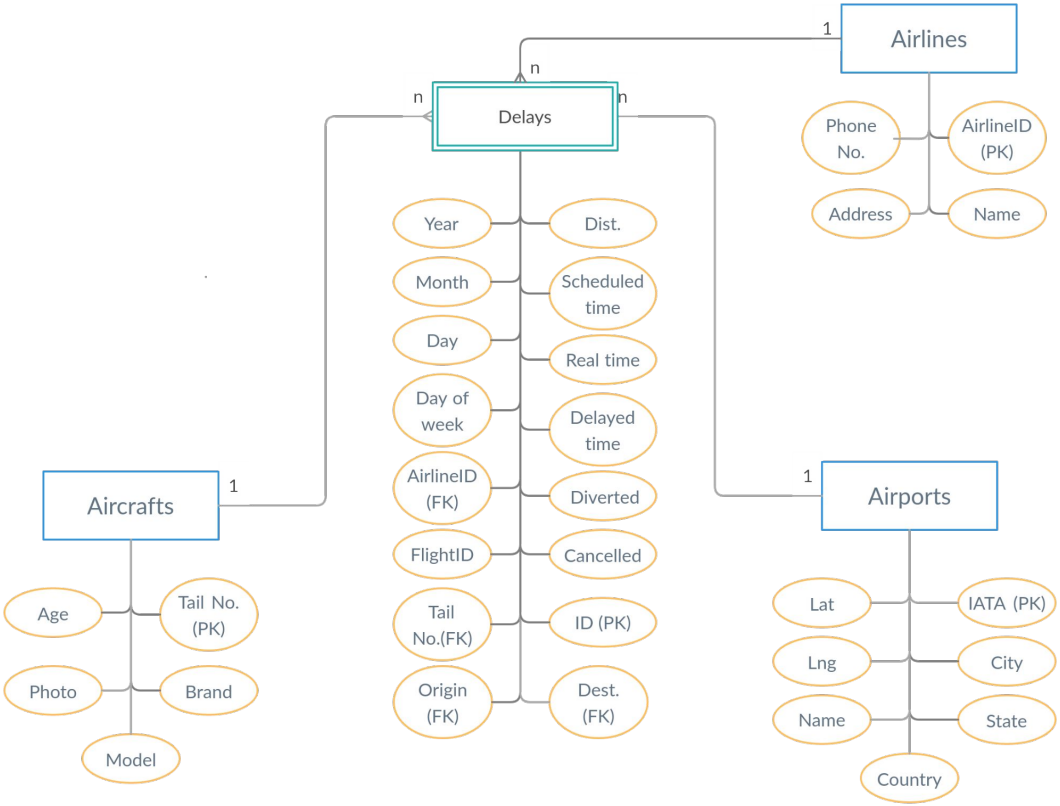
Схема Master-Slave репликации

- 1 Master-реплика, N Slave-реплик;
- схема эффективна, когда число запросов на чтение значительно выше числа запросов на изменение данных.

Выбор технических средств

- Язык программирования: Python 3.
- Редактор кода: VS Code.
- СУБД: PostgreSQL 13, Redis.
- Web-framework: Flask.
- Библиотека для нагрузочного тестирования: Locust.

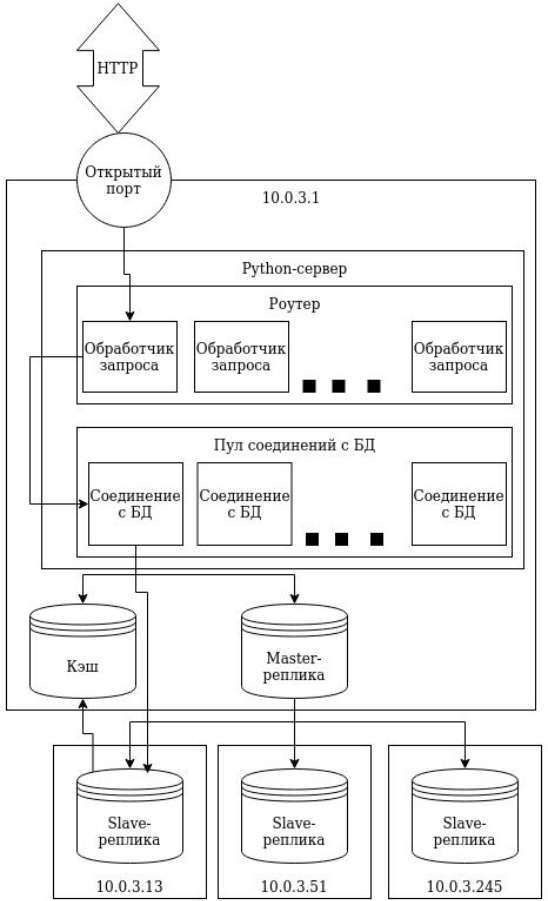
ER-диаграмма



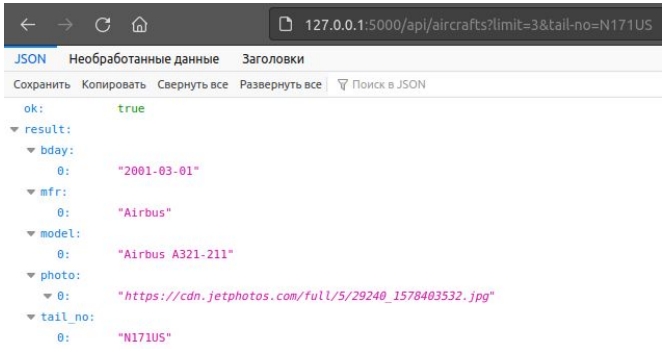
Ролевая модель

- postgres - разрешено чтение, создание, изменение и удаление данных, суперпользователь;
- guest - разрешено только чтение.

Структура системы



Примеры работы ПО




```
❯ curl -X DELETE -H "Content-Type: application/json" \
    -d '{ "id": 1 }' \
    http://127.0.0.1:5000/api/flights
{"ok":false,"result":"401 - unauthorized"}
❯ curl -X DELETE -H "Content-Type: application/json" \
    -d '{"token": "wrong-token",  "id": 1}' \
    http://127.0.0.1:5000/api/flights
{"ok":false,"result":"403 - forbidden"}
```

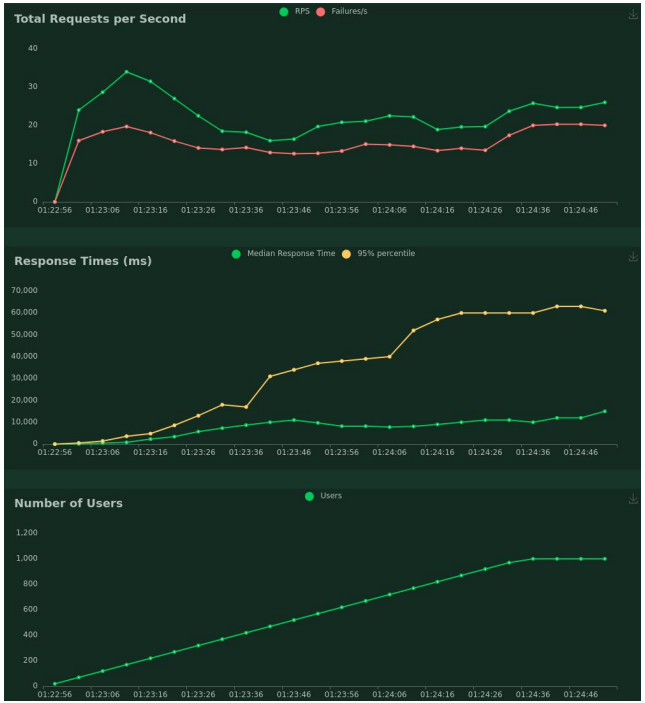
```
❯ curl -X PUT -H "Content-Type: application/json" \
    -d '{"token": "@topsecrettoken@",  "tail_no": "N171US", "photo": "https://example.com/photos/newphoto.jpg" }' \
    http://127.0.0.1:5000/api/aircrafts
{"ok":true,"result":null}
```

Исследование эффективности разработанной системы

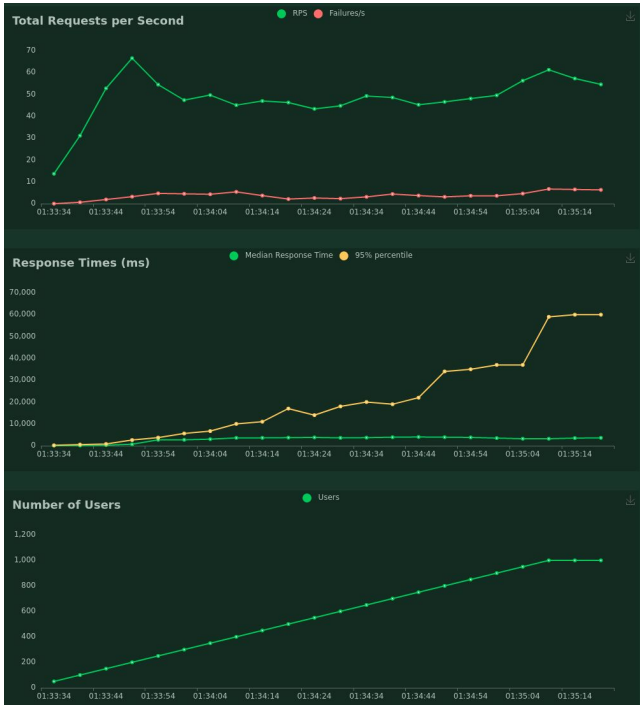
	Всего запросов отправлено	Число ошибок	Среднее время отклика (мс)	RPS	Failures/s
Базовая система	2728	1902	14055	23.4	16.3
DIA система	5670	463	5713	49.9	4.1

	Время отклика системы (мс)				
	50% процентиль	80% процентиль	90% процентиль	95% процентиль	100% процентиль
Базовая система	8300	16000	50000	60000	75000
DIA система	3000	58000	10000	21000	71000

Исследование эффективности разработанной системы



Базовая система



DIA-система

Заключение

- Произведен анализ способов масштабирования БД в высоконагруженных сервисах.
- Создано программное обеспечение, агрегирующее данные об авиаперелетах, в котором реализованы рассмотренные методы масштабирования.

- Исследована эффективность разработанной системы с применением механизмов DIA в сравнении с базовой системой.

Спасибо за внимание!