Нереляционные СУБД

7 модуль

Причины возникновения NoSQL

- Потребность в распределенных СУБД
- Потребность в быстрой работе с данными
- Некоторые часто встречаемые задачи можно моделировать проще

Особенности NoSQL

- Объект данных - более сложная структура, чем просто у строки в таблице
- Без строго определенной схемы
- Без операций соединения JOIN
- Масштабируется
- Нет SQL, но бывает что-то похожее

CAP-теорема

- Consistency - информация на разных узлах согласована
- Availability - система отвечает на запросы
- Partition tolerance - связи между узлами могут обрываться

Теорема: для распределенных систем можно выбрать только 2 свойства

BASE-архитектура

- Basically Available - сбой узла приводит к отказу только для части пользователей
- Soft-state - система может находиться в неусточивом состоянии
- Eventual Consistency - когда-нибудь согласуется

Классы NoSQL

- **Ключ-значение** - быстрый доступ к данным по ключу
- **Документоориентированные** - удобство моделирования вложенных структур без жесткой схемы
- **Семейство столбцов** - таблицы с множеством незаполненных атрибутов (разреженная матрица)

Классы могут пересекаться.

Ключ-значение

Используется для:

- Работы данными в реальном времени
- Кэширования результатов долгих операций
- Файловые системы на основе “ключ-значение”

Примеры СУБД: Berkeley DB, Redis, Memcache

Документоориентированные

Используются для:

- Хранения документов и поиска по ним
- Информационные сервисы
- Данные с нечеткой схемой

Примеры: MongoDB, CouchDB

Семейство столбцов

Используется для:

- Хранение данных, собранных с веб-страниц
- Хранение и поиск по большому объему данных

Примеры: Cassandra, HBase

Redis

- 100k операций в секунду для одного узла
- Всего больше сотни команд
- Все команды работы с данными по крайне мере содержат ключ
- Для каждого типа данных свой набор CRUD-команд, различаются префиксами, например: LPOP, SPOP

Типы данных Redis

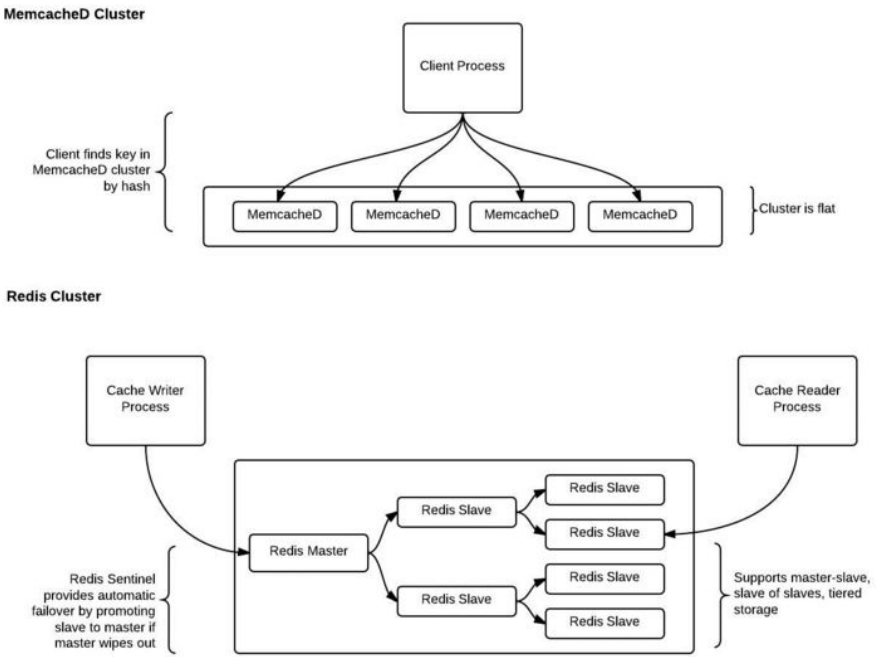
Ключи: уникальная строка для одного набора данных

Значения:

- Строки - скалярные значения, для разных подтипов есть разные операции
- Хэши - для хэш-таблиц внутри значений
- Списки - как list в Python с теми же операциями
- Множества - как set в Python с теми же операциями
- Упорядоченные множества

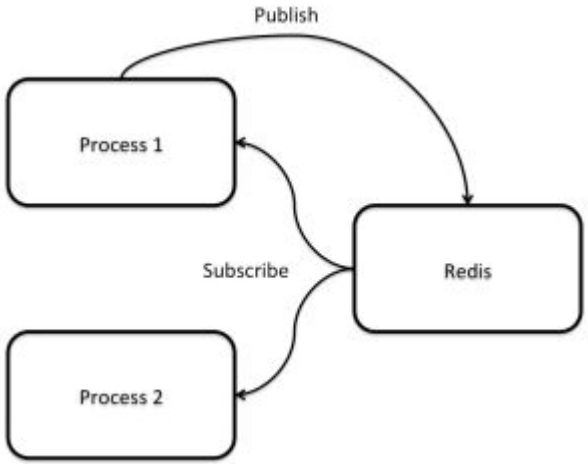
Репликация в Redis

- Обновления поступают не только от мастера, но и от уже обновленных слейвов, децентрализуя операцию
- Изменения всегда просты *(с одним ключом)*, поэтому передаются значения, а не операторы



Подписки (PubSub)

Не нужно каждый раз спрашивать, есть ли новые данные, сервер сам пройдется по активным подписчикам и отправит им изменения, когда они появятся.



Особенности MongoDB

- Самая популярная ⇒ самая стабильная
- Простота хранения данных ⇒ нет проблем с миграциями
- Сложность восприятия запросов, но легко писать интерпретаторы и генераторы ⇒ разнообразие ORM

Пользовательские команды

Все параметры операций - валидный JSON:

- Сложность восприятия: скобки, нет многих символов
- Легко найти драйвер с подходящим синтаксисом в любом языке

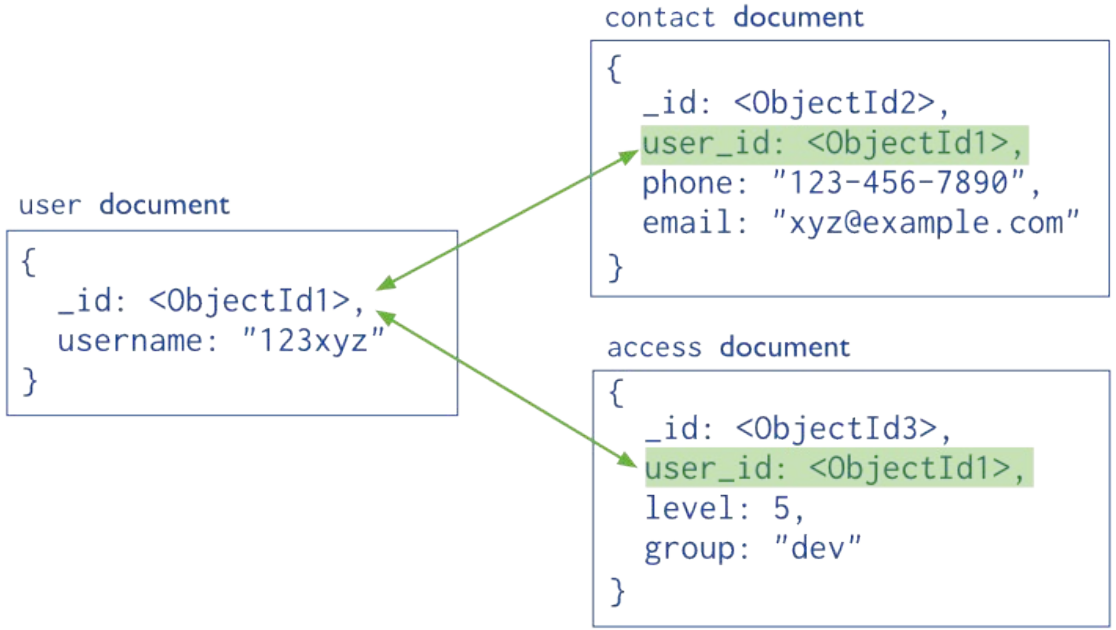
```
db.unicorns.find({gender: 'f', $or: [{loves: 'apple'}, {loves: 'orange'}, {weight: {$lt: 500}}]})
```

Что есть вместо JOIN

- Ссылки и списки ссылок на объекты
- Встроенные документы, денормализация
- Тип DBRef для драйверов

Ссылки на объекты

Но нужно делать
несколько запросов



Встроенные документы

Аномалии обновления, если данные дублируются

DBRef

Только для использования в драйверах и ORM

db.post.find({'author': DBRef("user", ObjectId('...'))})