11. Введение в NoSQL

к.т.н., доцент кафедры ИиСП

Лучинин
Захар Сергеевич

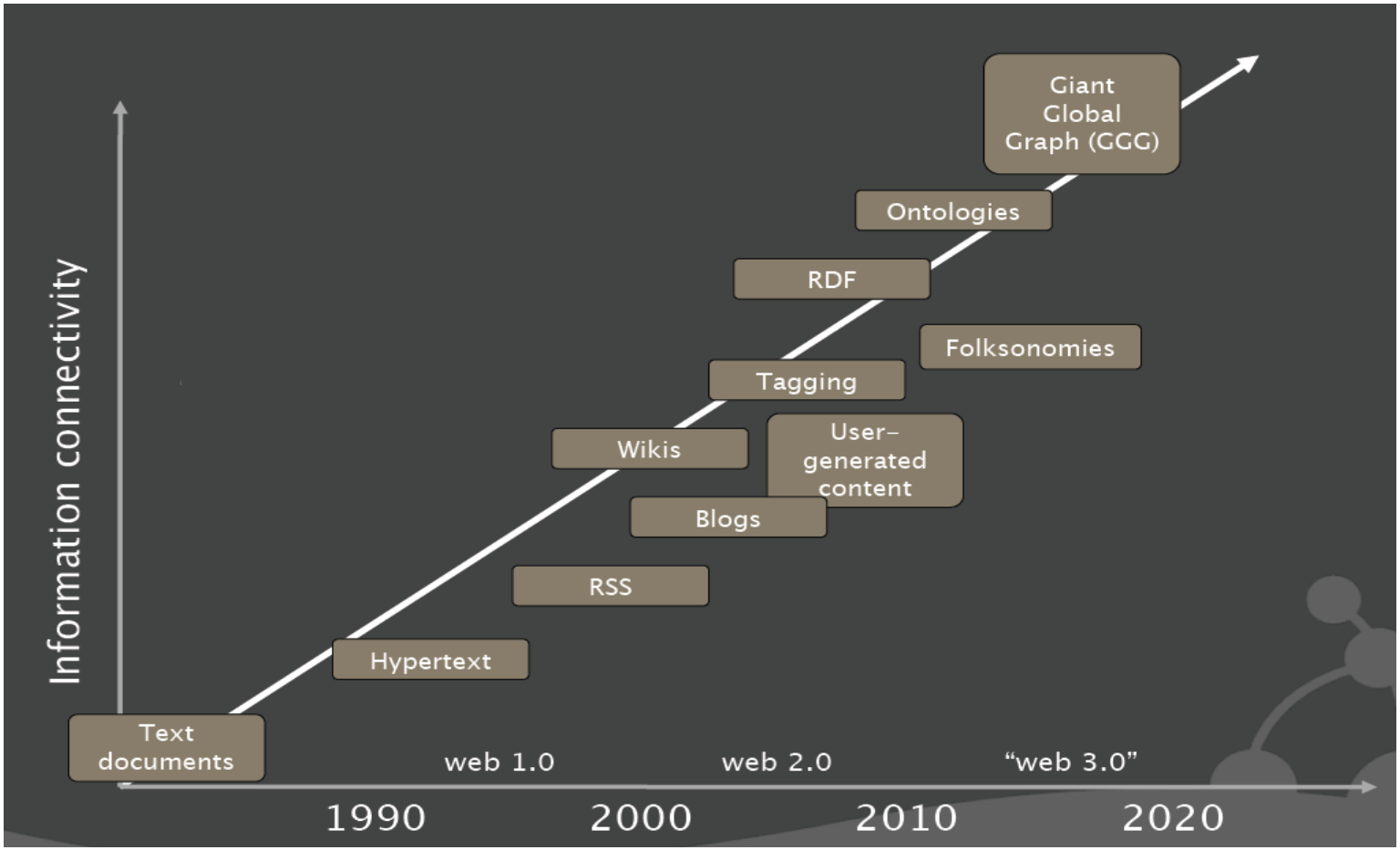
Почему РСУБД - SQL

- Более 30 лет успешного существования на рынке
- Контроль избыточности данных
- Непротиворечивость данных
- Транзакции
- Совместное использование данных
- Поддержка целостности данных

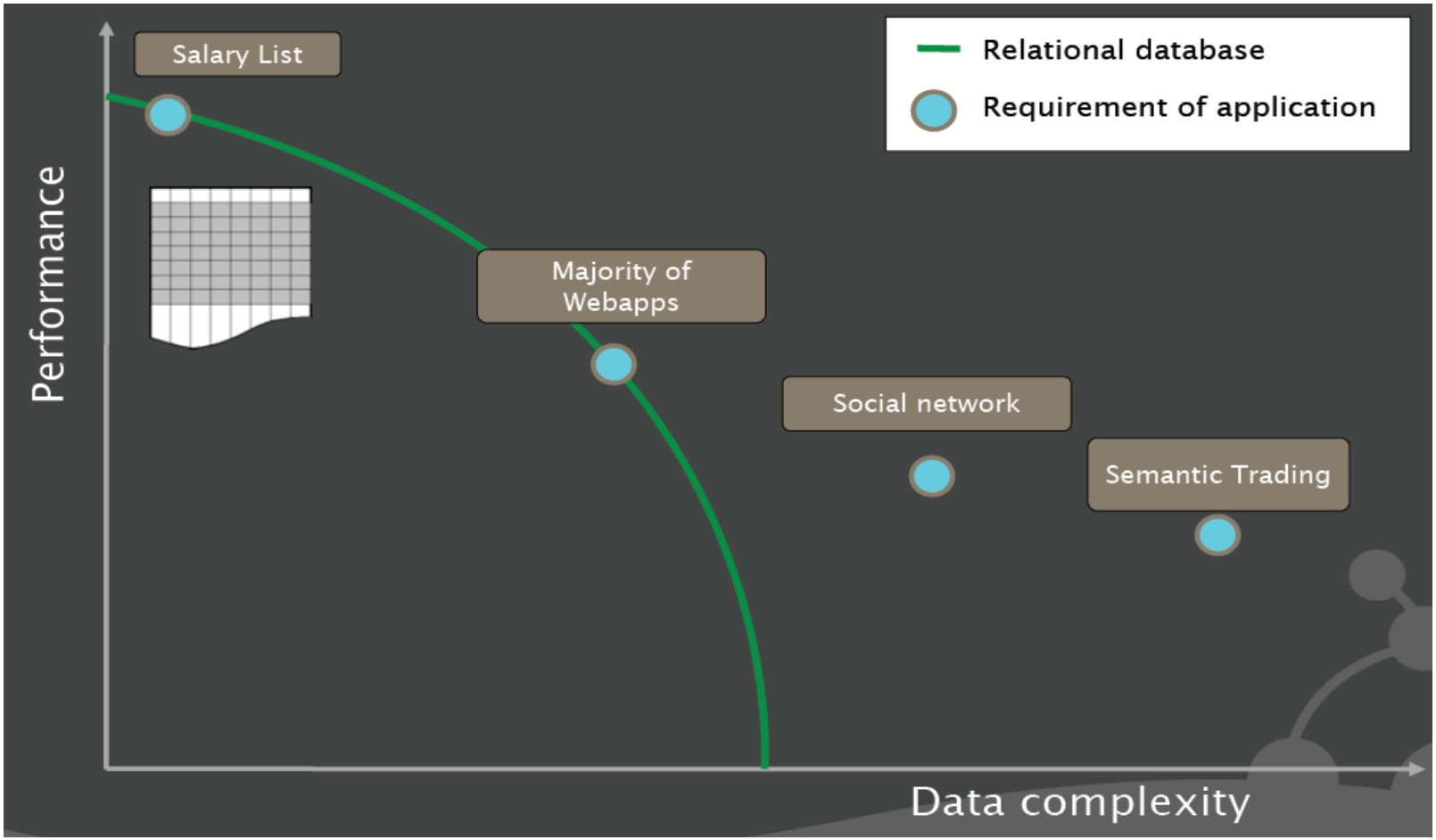
Почему НЕ РСУБД - SQL

- Производительность
- Избыточная сложность
- Затраты на преобразование объектов БД в объекты ПО
- Стоимость коммерческого использования

Данные стали сложнее



Производительность РСУБД



NoSQL

- NoSQL (от англ. **not only SQL** — не только SQL) — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных

Причины возникновения NoSQL

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

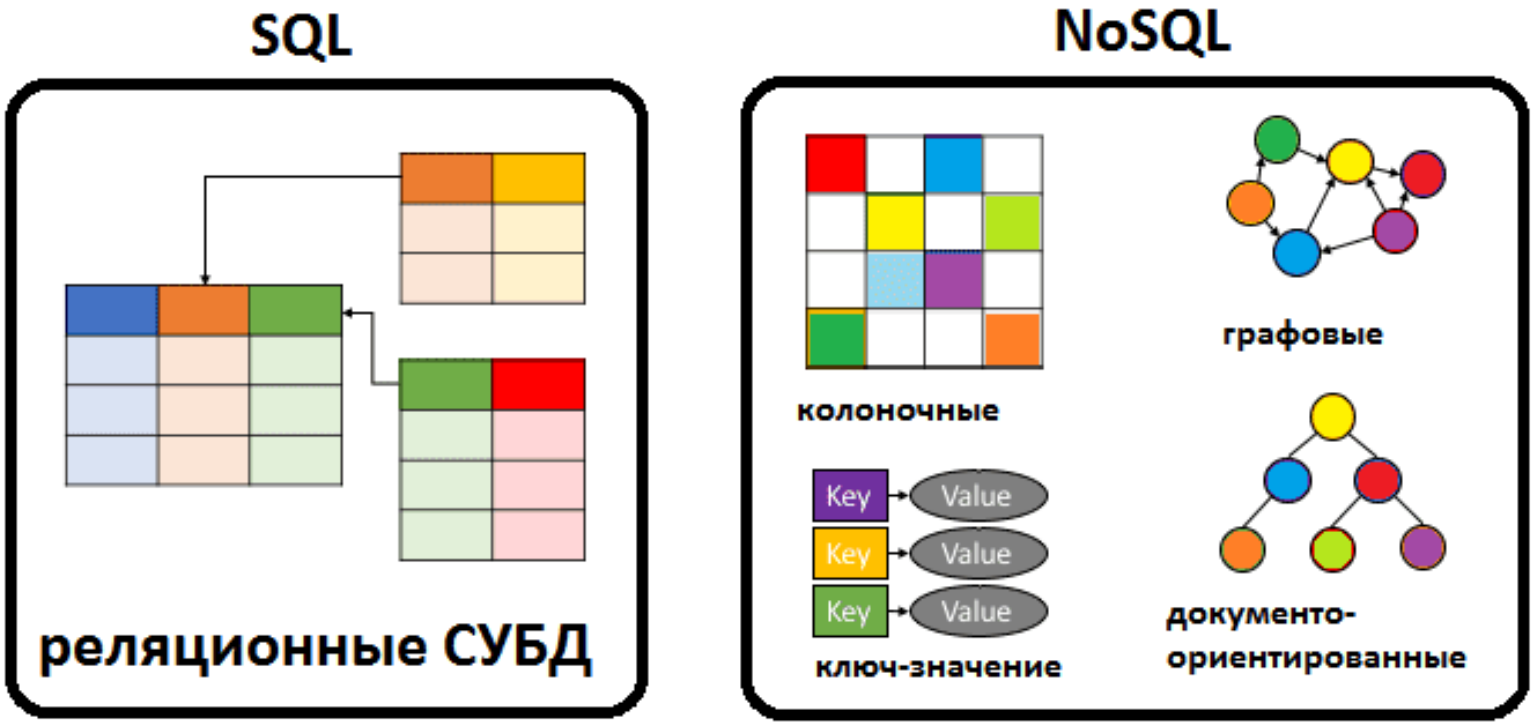
Особенности NoSQL

- Объект данных - более сложная структура, чем просто у строки в таблице
- Без строго определенной схемы
- Без операций соединения JOIN
- Хорошо масштабируется
- Попытки повторить SQL

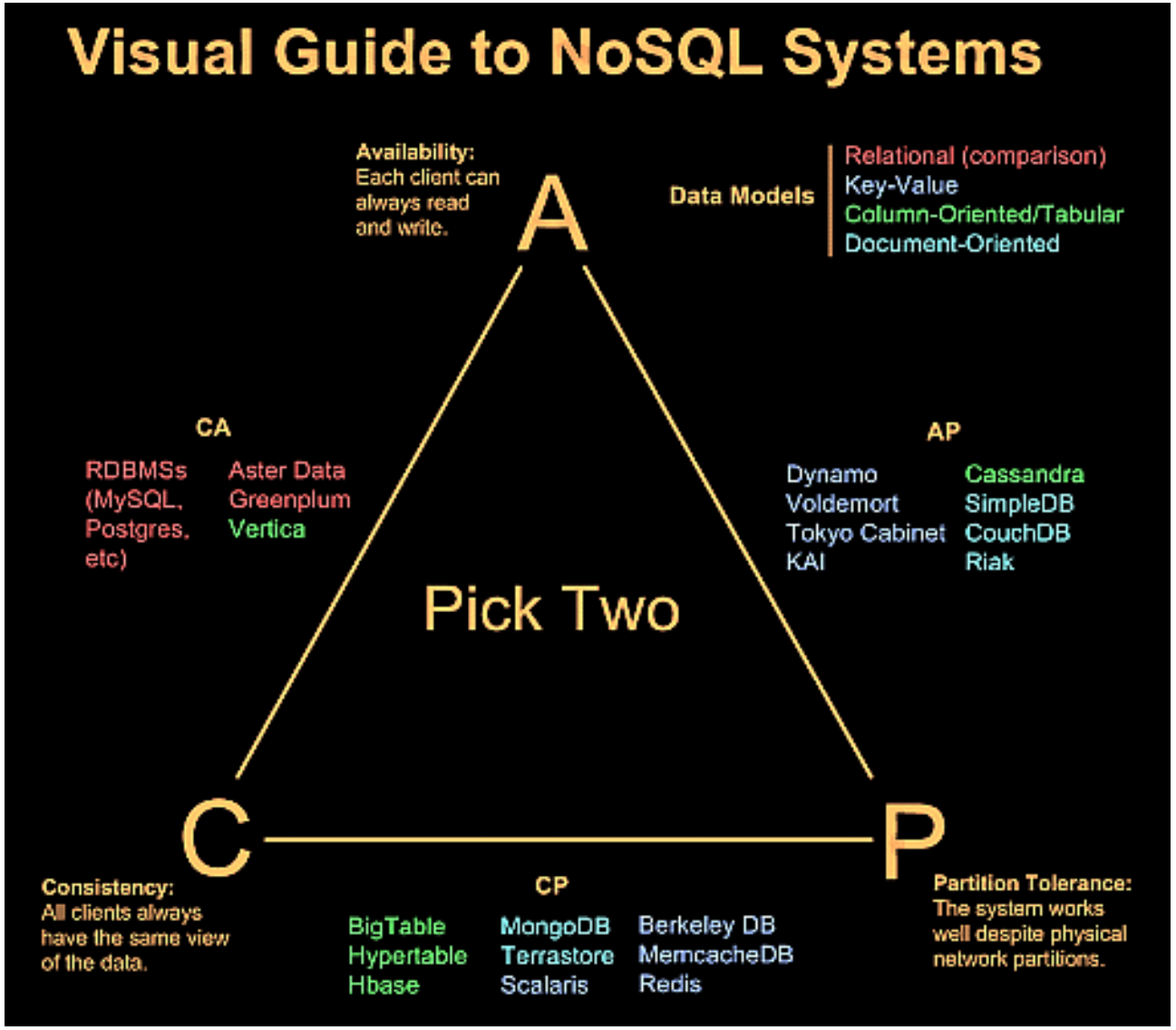
Принцип NoSQL СУБД - BASE

- **Базовая доступность** (basic availability) — каждый запрос гарантированно завершается (успешно или безуспешно).
- **Гибкое состояние** (soft state) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
- **Согласованность в конечном счёте** (*eventual consistency*) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Карта СУБД


SQL
реляционные СУБД
NoSQL
колоночные
графовые
Key
Value
Key
Value
Key
Value
ключ-значение
документо-
ориентированные

Теоретические основы NoSQL



Теорема CAP

- **Согласованность данных (англ. consistency)** — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;
- **Доступность (англ. availability)** — любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;
- **Устойчивость к разделению (англ. partition tolerance)** — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

Виды сервисов согласно CAP теореме

1. **Доступность сервиса**

AP (Availability + Partition Tolerance - best effort availability) системы – отвечаем на запросы, однако возвращенные данные не всегда могут быть актуальными – например это DNS

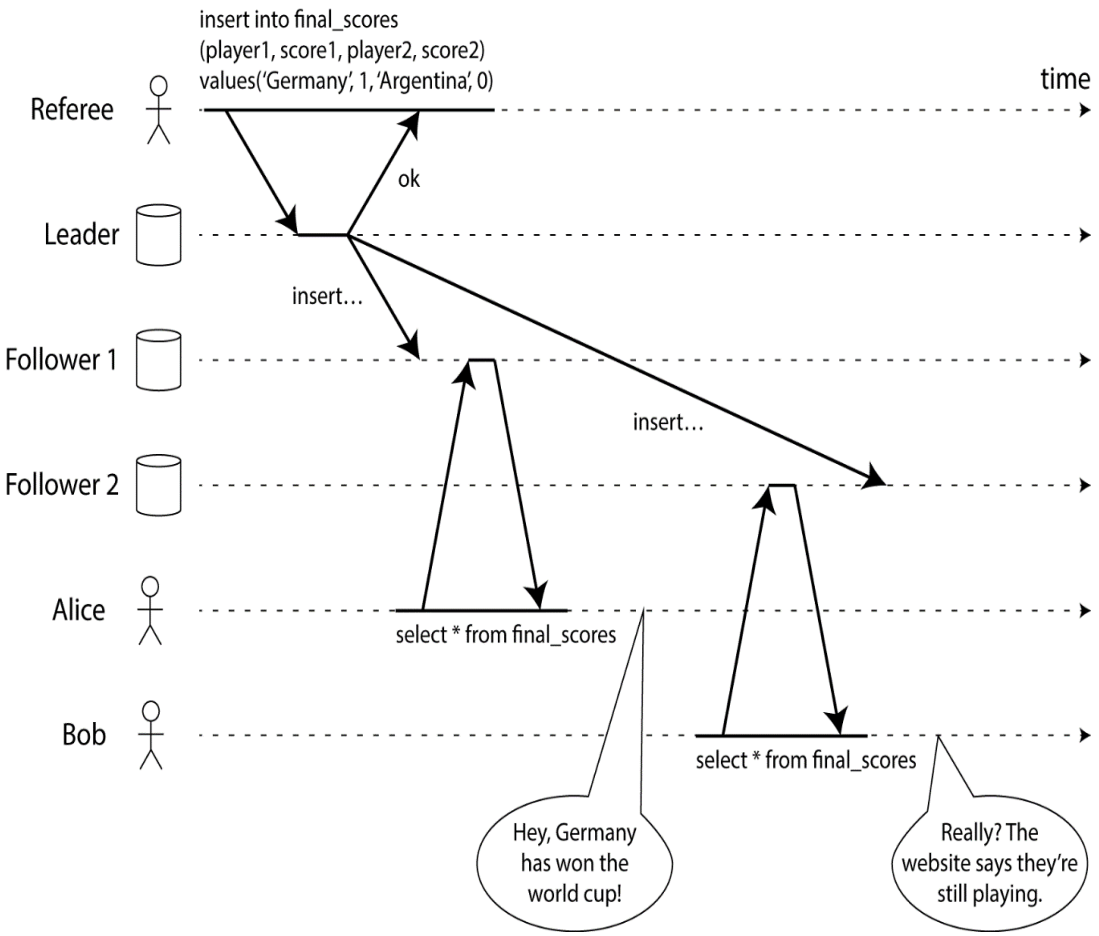
2. **Целостность данных при разделении**

CP (Consistency + Partition tolerance) системы – отвечают на запросы всегда с корректными данными, но при проблемах с сетью, узлы могут не отвечать

3. **Целостность и доступность**

AC (Availability + Consistency) – возвращают всегда корректные данные, однако не имеют возможности разделения данных по сети при этом

На самом деле все сложнее



IT теория

- Сетевые соединения надежны
- Пропускная способность сети бесконечна
- Сетевое соединение безопасно

IT реальность

- Сетевые соединения не надежны и часто падают
- Пропускная способность сети конечна
- Сетевое соединение не безопасно

Характеристики C, A, P

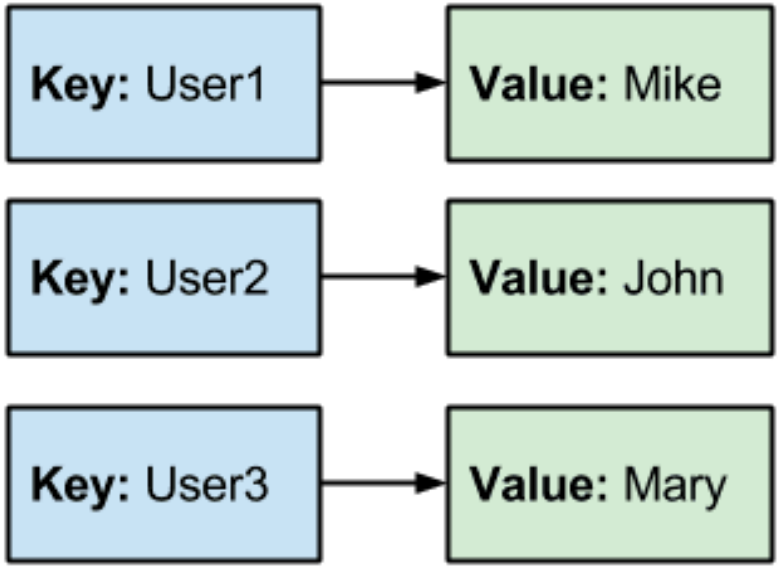
- Доступность, очевидно, может измеряться по времени прошедшему между посылкой запроса и получением ответа.
- Какие-то системы могут позволить себе долго отвечать на запрос, другие должны сделать за определенный промежуток времени.
- Свойство устойчивости к разделению напрямую зависит от того, каким образом классифицировать сложившуюся ситуацию как разделение.
- У согласованности данных тоже могут быть разные степени **(Согласованность в конечном счете)**

Типы хранилищ данных

- Хранилище «ключ-значение»
- Хранилище семейств колонок
- Документо-ориентированная СУБД
- Базы данных на основе графов

Хранилище «ключ-значение»

- Модель данных: множество пар ключ-значение
- Для БД содержание значение черный ящик



Хранилище «ключ-значение»

- In-memory: Redis, Aerospike, Tarantool
- Persistent first: Riak, Dynamo, Oracle NoSQL Database

Применение:

- Хранение кеша
- Хранение пользовательских сессий

Взаимодействие с СУБД

- **GET (key)** – получение значения по ключу
- **SET (key , value, ttl)** – установка значения по ключу с TTL
- **DEL (key)** – удаление ключа

Обработка ближе к данным

https://gist.github.com/jboner/2841832

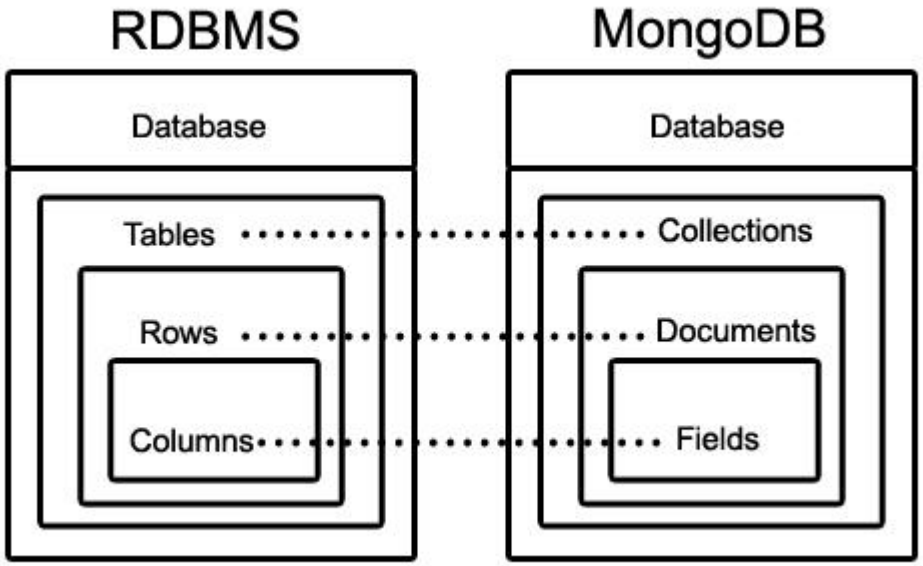
Документо-ориентированная СУБД

Особенности:

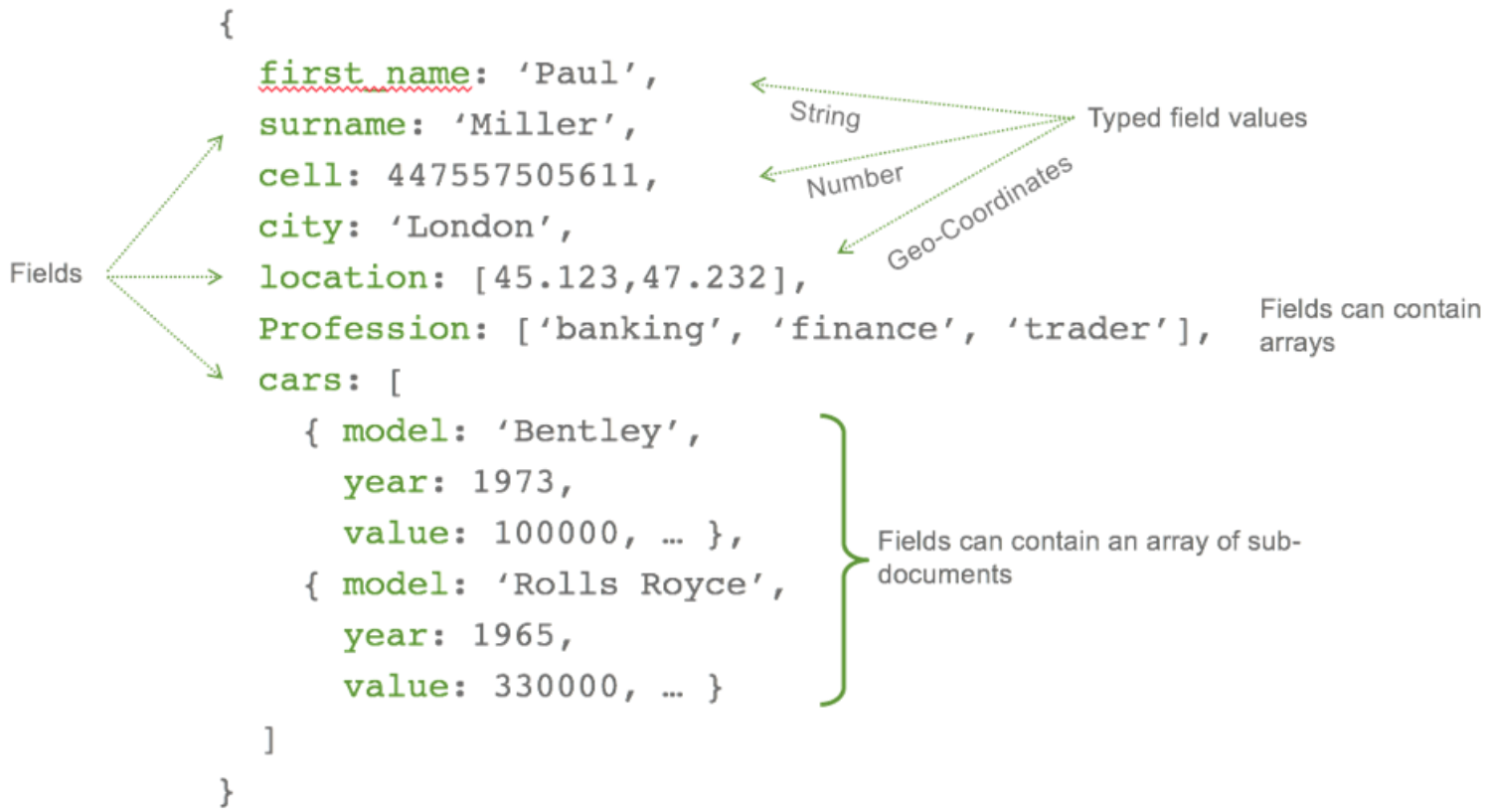
- Модель данных: множество множеств ключ-значение
- Для БД содержание значение прозрачно

Применение:

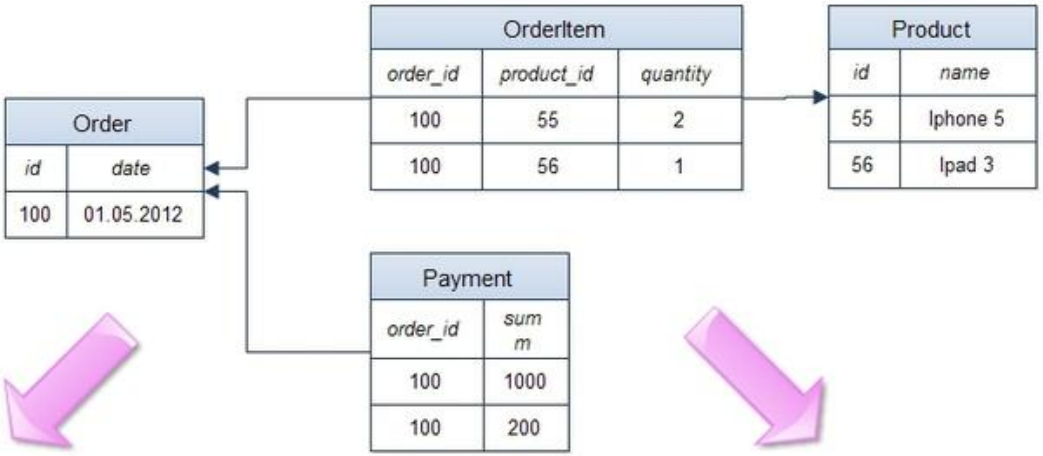
- Хранение слабоструктурированных данных
- Замена РСУБД



Пример документа



Relational model



Order

id	date
100	01.05.2012

OrderItem

order_id	product_id	quantity
100	55	2
100	56	1

Product

id	name
55	Iphone 5
56	Ipad 3

Payment

order_id	sum m
100	1000
100	200

Aggregate model 1

```
// Order document
{
"id": 100,
"customer_id": 1000,
"date": 01.05.2012,
"order_items": [
        {
        "product_id": 55,
        "product_name": Iphone5,
        "quantity": 2
        },
        {
        "product_id": 56,
        "product_name": Ipad3
        "quantity": 1
        }
],
"payments":[
        {
        "sum": 1000,
        "date": 03.05.2012
        }
]
}
// Product document here
{...}
```

Aggregate model 2

```
// Order document
{
"id": 100,
"customer_id": 1000,
"date": 01.05.2012,
"order_items": [
        {
        "product_id": 55,
        "product_name": Iphone5,
        "quantity": 2
        },
        {
        "product_id": 56,
        "product_name": Ipad3
        "quantity": 1
        }
]
}
// Payment document
{
"order_id": 100,
"sum": 1000,
"date": 03.05.2012
}
// Product document here
{...}
```

Агрегаты

Нормализация данных	Данные в виде агрегатов
• Целостность информации при обновлении (меняем запись в одной таблице, а не в нескольких) • Ориентированность на широкий спектр запросов к данных	• Оптимизация только под определенный вид запросов • Сложности при обновлении денормализованных данных
• Неэффективна в распределенной среде • Низкая скорость чтения при использовании объединений (joins) • Несоответствие объектной модели приложения физической структуре данных (impedance mismatch, решается с помощью Hibernate etc.)	• Лучший способ добиться большой скорости на чтение в распределенной среде • Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании) • Родная (native) поддержка атомарности на уровне записей

Документо-ориентированная СУБД

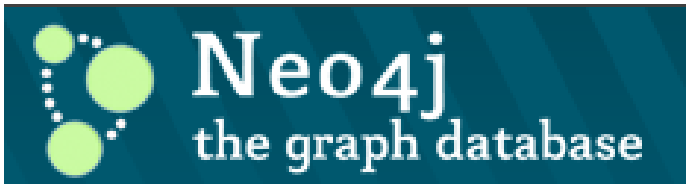
MongoDB

CouchDB

Графовые базы данных

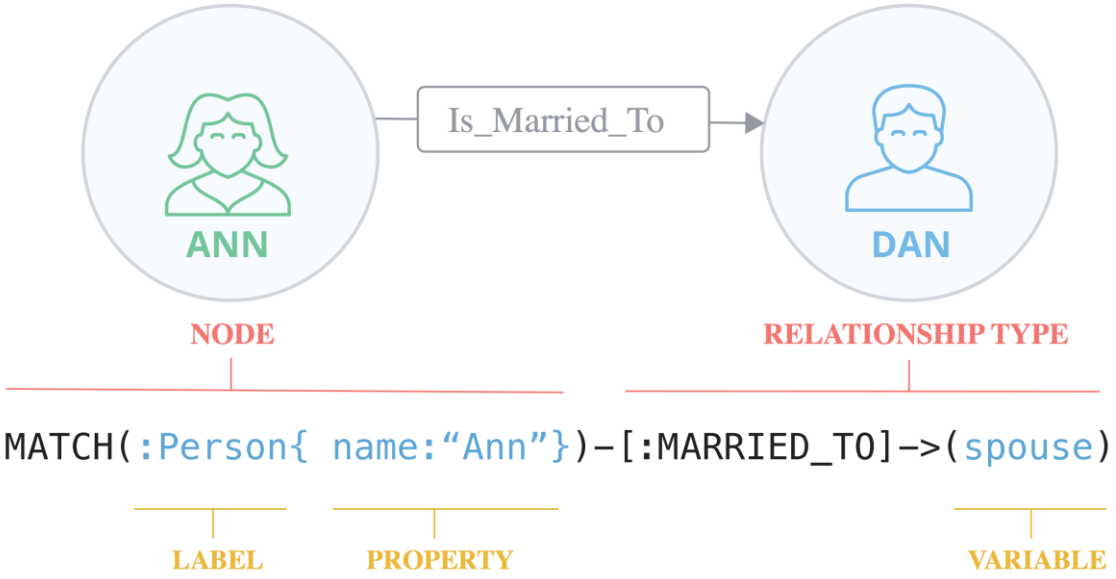
- Навеяны теорией графов G=(V, E) от математиков 18го века
- Хорошо моделируют сложные данные
- Модель данных: узлы, ребра и их атрибуты

- Пример: Neo4j



Применение графовых баз данных

- Используют сервисы вроде социальных приложений, рекомендательные сервисы, обнаружение мошенничества, графики знаний, биологические исследования и ИТ/сеть.



Колоночные СУБД

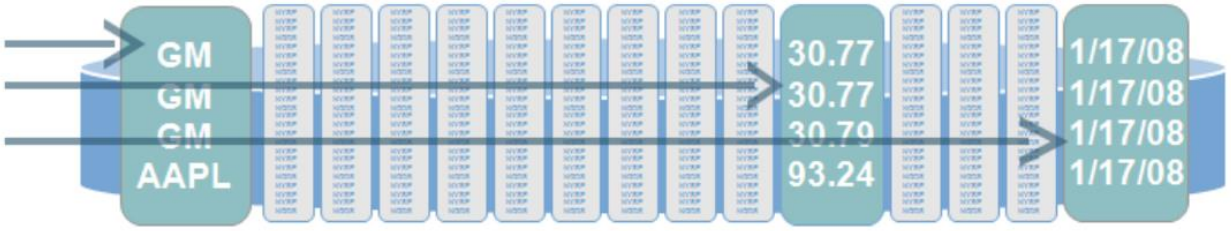
- Призваны решить проблему неэффективной работы традиционных СУБД в аналитических системах и системах в подавляющим большинством операций типа «чтение».

Применение:

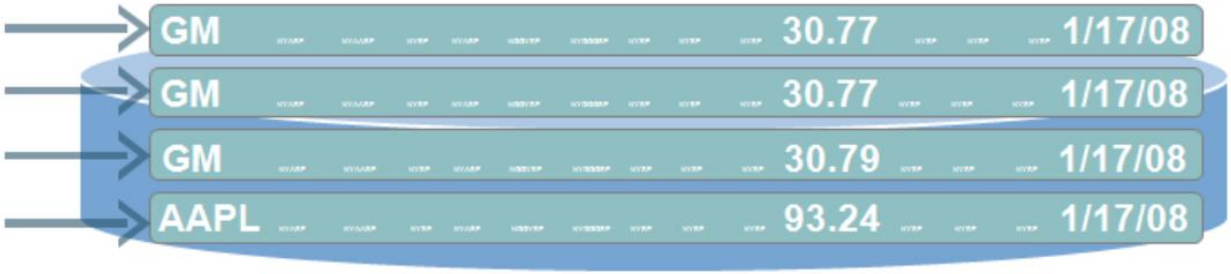
- Аналитические системы

Хранение по строкам и столбцам

Хранение по столбцам
Чтение 3х столбцов



Хранение по строкам
Чтение всех столбцов



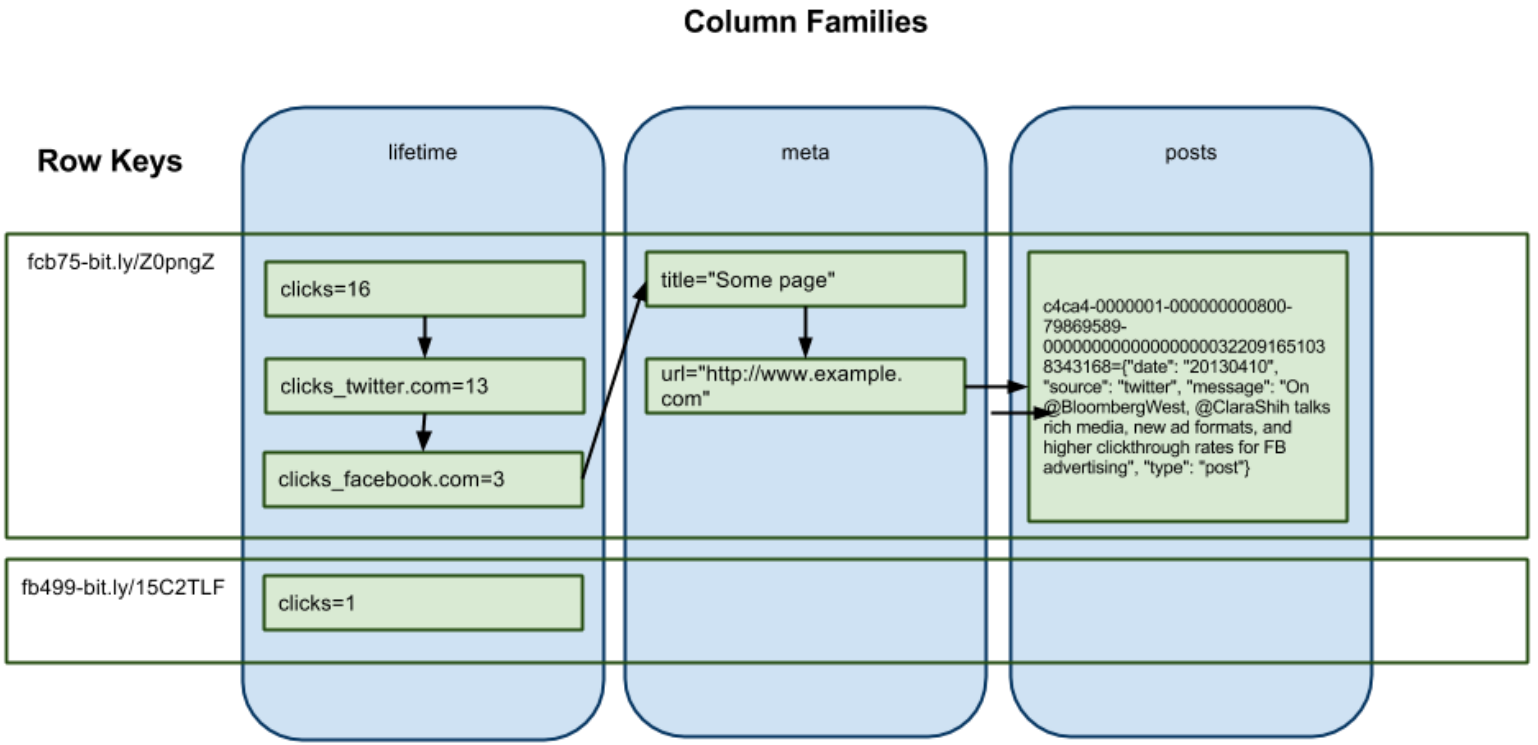
Колоночные СУБД

- HBase

- InfluxDB

Wide-column store

- KV внутри KV, хранение произвольного числа колонок внутри значения
- Подходит для гибкого хранения денормализованных объектов



Колоночные СУБД

- HBase

- Cassandra