

Лекция 14.

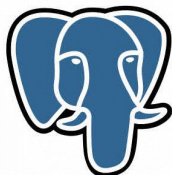
Нереляционные (NoSQL) базы данных

Реляционные СУБД

- Жесткая структура данных в таблицах.
- Нормализация данных (разделение по нескольким таблицам).
- Поддержка ACID-транзакций (Atomicity, Consistency, Isolation, Durability).
- Стандартный язык SQL для выборки и манипуляции с данными.
- Вертикальная масштабируемость (увеличение производительности сервера).



PostgreSQL

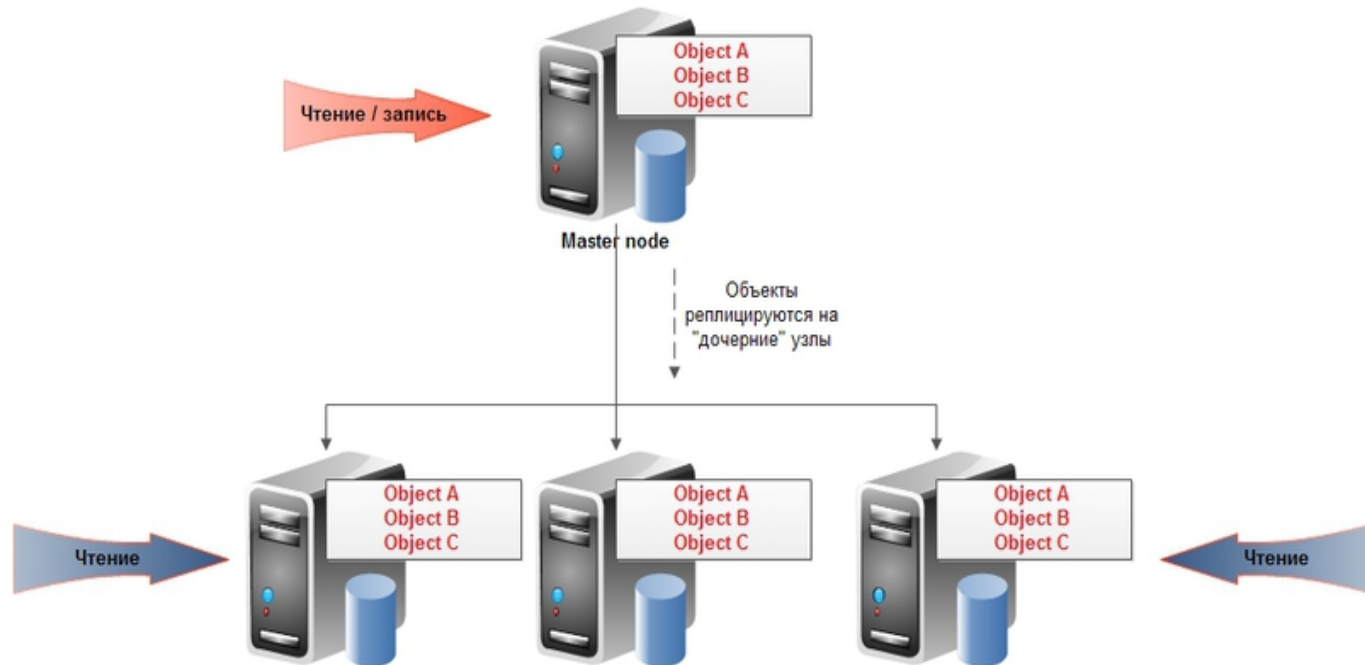


ORACLE®
DATABASE

Борьба с большими объемами данных

Масштабирование БД для ускорения чтения + повышение надежности системы.

- **Репликация.**



Синхронное обновление – увеличивается время ответа системы

Асинхронное обновление – есть время, когда реплики несогласованы

Борьба с большими объемами данных

Масштабирование БД для ускорения записи

- Шардинг



Возникают задачи:

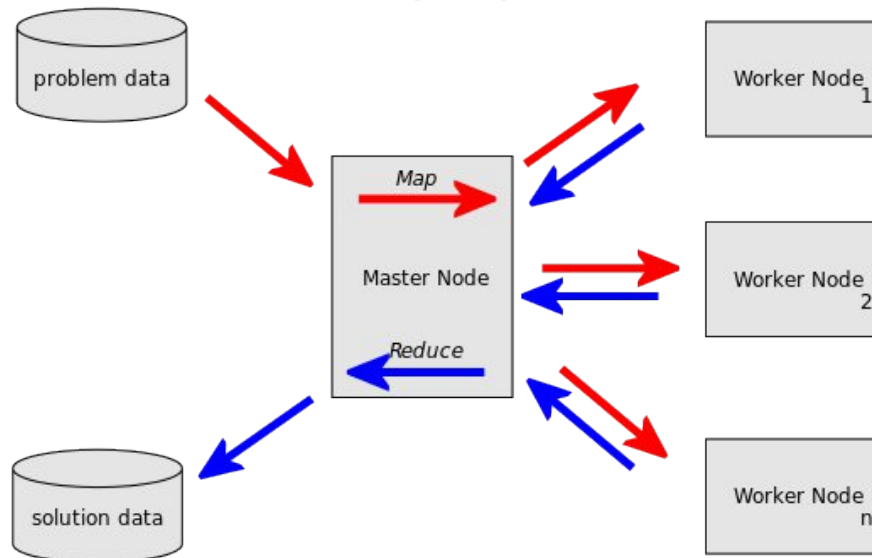
- Распределение данных по узлам.
- Балансировка нагрузки.
- Оптимизация загрузки сети.

MapReduce

Механизм параллельной обработки больших данных на кластерах.

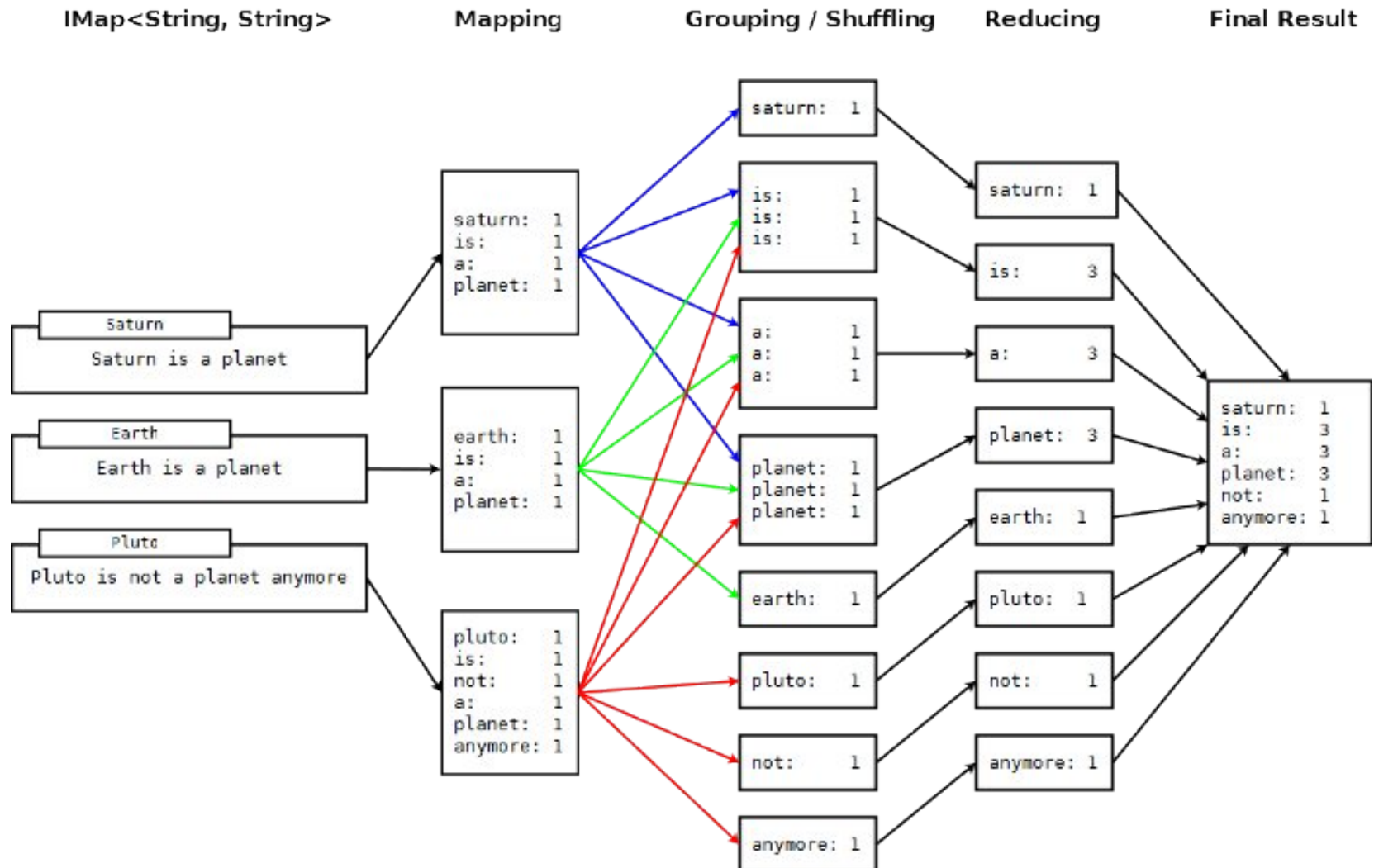
Map: Главный узел (master node) делит задачу на части, распределяя их по остальным машинам (worker nodes).

Reduce: Master node получает предварительные результаты, и формирует из них конечный результат.



MapReduce

Пример: Подсчет количества слов во входном тексте



Чем могут не устраивать реляционные СУБД?

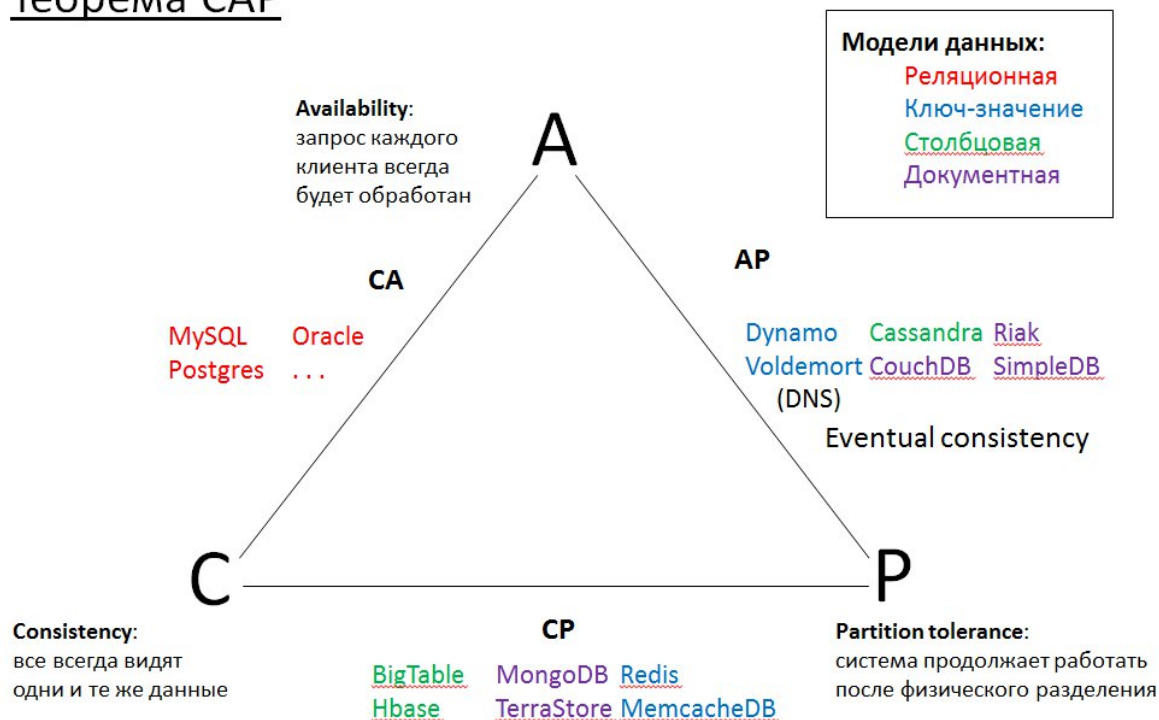
- Необходимо поддерживать две разнородные модели: реляционную в базе и объектную в коде.
- Любые изменения в схеме сущностей нужно отражать в структуре таблиц + менять SQL-запросы и проекции таблиц на объекты.
- При больших объемах данных возникает порог вертикального масштабирования. Необходима параллельная обработка в кластере серверов .
- Падение производительности в больших распределенных системах из-за необходимости поддерживать согласованность данных + проблемы с устойчивостью к физическому разделению узлов.

NoSQL СУБД

Основные причины появления – распределенность систем:

- Потребность в горизонтальной масштабируемости БД (путем добавления новых узлов).
- Сложность эффективной реализации транзакций в распределенной среде.

Теорема CAP



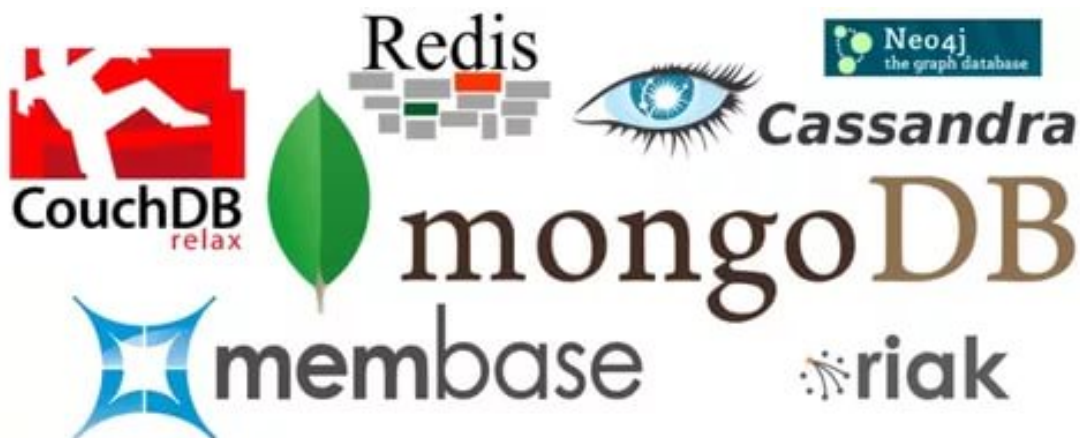
NoSQL СУБД

Основные особенности:

- Отказ от реляционной модели и языка SQL
- Использование распределенной архитектуры
- Отсутствие полноценной поддержки ACID-транзакций

Преимущества:

- Большая производительность
- Хорошая масштабируемость при возрастающих нагрузках и огромных объемах данных



Реляционные БД vs NoSQL

Реляционные БД: есть четко определенные схемы таблиц

NoSql: Schema-less данные – в NoSQL БД структура данных заранее не регламентируется.

Реляционные БД vs NoSQL

Реляционные БД: есть четко определенные схемы таблиц

NoSql: Schema-less данные – в NoSQL БД структура данных заранее не регламентируется.

Плюсы: структура каждого документа может быть уникальной (он сам себя описывает).

Минусы: поддержка структуры данных переносится с сервера в логику приложений.

Реляционные БД vs NoSQL

Реляционные БД: есть связи между таблицами. Данные в одной таблице являются ссылкой на данные в другой (принцип нормализации). Есть встроенные механизмы поддержки целостности данных.

NoSql: Каждый документ – изолированная информационная единица, хранит в себе все данные. Нет встроенной поддержки целостности данных. Важна денормализация данных.

Денормализация (агрегация) данных



Relational data model



Document data model

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

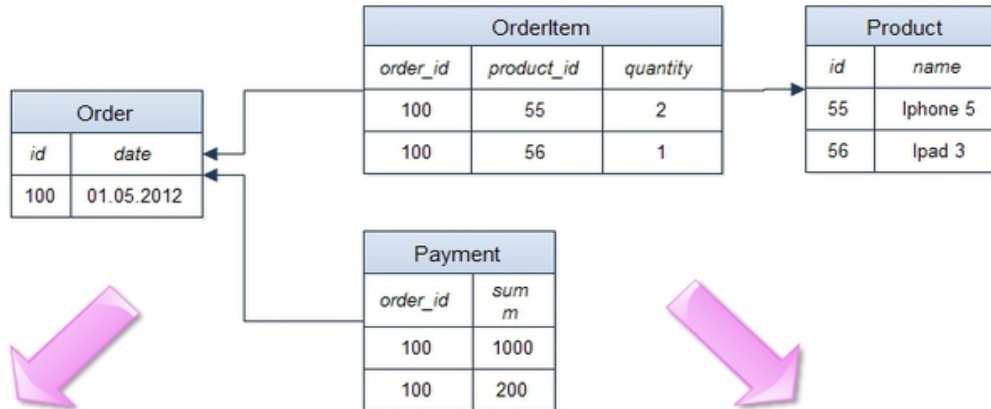


MongoDB Document

```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, ... },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, ... }
  ]
}
```

Денормализация (агрегация) данных

Relational model



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
{...}
```

Денормализация (агрегация) данных

Нормализованные данные	Данные в виде агрегатов
<ul style="list-style-type: none">• Целостность данных при обновлении (меняем только в одной таблице)• Ориентированность на широкий спектр запросов	<ul style="list-style-type: none">• Оптимизация только под определенный тип запросов• Сложности при обновлении денормализованных данных
<ul style="list-style-type: none">• Неэффективность в распределенной среде• Низкая скорость чтения при использовании объединений (join)• Несоответствие объектной модели приложения физической структуре данных	<ul style="list-style-type: none">• Повышение скорости чтения данных.• Можно хранить данные в том виде, в каком они обрабатываются в приложении.

Реляционные БД vs NoSQL

Реляционные БД: поддерживается механизм транзакций (несколько SQL-запросов по принципу «все или ничего»).

NoSql: транзакции поддерживаются ограниченно (например, только внутри одного документа).

Реляционные БД vs NoSQL

NoSQL базы данных проще масштабировать, так как отсутствуют сложные логические связи между документами.

В идеале NoSQL СУБД работает быстрее реляционной СУБД.

Модели данных NoSQL

Системы «ключ-значение»

- CRUD-операции (Create-Retrieve-Update-Delete)
- Хорошая горизонтальная масштабируемость
- Простота и производительность
- Плохо подходят для операций, отличных от CRUD



Модели данных NoSQL

Документо-ориентированные системы

- Единица хранения – документ, т.е. объект, обладающий произвольным набором атрибутов (полей), который может быть представлен, например, в JSON
- Поддерживают поиск по полям документов, индексы, часто допускаются вложенные документы и массивы.



Документо-ориентированные модели

Расширять структуру данных проще, чем в реляционной модели

Коллекция "Товары"

```
{
  Id: 1,
  Артикул: "4678",
  Название: "Стол письменный",
  Исполнения: [
    {Материал: "Сосна",
     Цвет: "Беж"},
    {Материал: "ДСП",
     Цвет: "Белый"}
  ]
}
```

```
{
  Id: 2,
  Артикул: "7875",
  Название: "Плита ДСП",
  Габариты: {
    Ширина: 150,
    Длина: 200,
    Толщина: 1.5
  }
}
```

Товары

Id	Артикул	Название
1	4678	Стол письменный
2	7875	Плита ДСП

Исполнения товара

Id	Материал	Цвет
1	Сосна	Беж
1	ДСП	Белый

Габариты товара

Id	Ширина	Длина	Толщина
2	150	200	1.5

Документо-ориентированные модели



```
SELECT
  Dim1, Dim2,
  SUM(Measure1) AS MSum,
  COUNT(*) AS RecordCount,
  AVG(Measure2) AS MAvg,
  MIN(Measure1) AS MMin
  MAX(CASE
    WHEN Measure2 < 100
    THEN Measure2
  END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A', 'B'))
  AND (Filter2 = 'C')
  AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
```

- ① Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
- ② Measures must be manually aggregated.
- ③ Aggregates depending on record counts must wait until finalization.
- ④ Measures can use procedural logic.
- ⑤ Filters have an ORM/ActiveRecord-looking style.
- ⑥ Aggregate filtering must be applied to the result set, not in the map/reduce.
- ⑦ Ascending 1; Descending: -1



```
db.runCommand({
  mapreduce: "DenormAggCollection",
  query: {
    filter1: { '$in': [ 'A', 'B' ] },
    filter2: 'C',
    filter3: { '$gt': 123 }
  },
  map: function() { emit(
    { d1: this.Dim1, d2: this.Dim2 },
    { msum: this.measure1, recs: 1, mmin: this.measure1,
      mmax: this.measure2 < 100 ? this.measure2 : 0 }
  );},
  reduce: function(key, vals) {
    var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
    for(var i = 0; i < vals.length; i++) {
      ret.msum += vals[i].msum;
      ret.recs += vals[i].recs;
      if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
      if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
        ret.mmax = vals[i].mmax;
    }
    return ret;
  },
  finalize: function(key, val) {
    val.mavg = val.msum / val.rec;
    return val;
  },
  out: 'result1',
  verbose: true
});
db.result1.
find({ mmin: { '$gt': 0 } }).
sort({ recs: -1 }).
skip(4).
limit(8);
```

Модели данных NoSQL

Системы типа Google BigTable

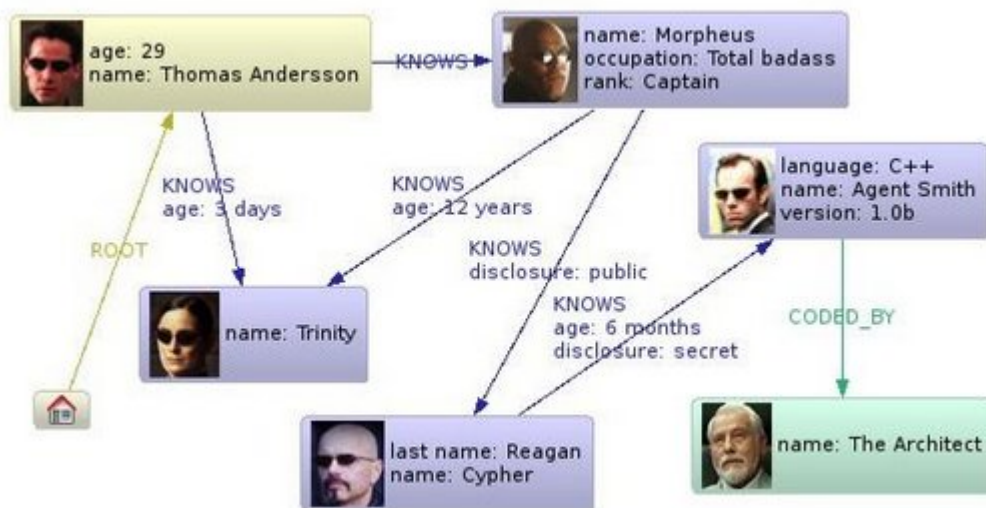
- Данные хранятся в виде строк. Для доступа к данным три ключа: ключ строки (по нему отсортированы строки в базе), ключ столбца, временная метка.
- Связанная информация хранится в одном месте.



Модели данных NoSQL

Графовые системы

- Состоит из узлов и связей между ними.
- Как с узлами, так и со связями можно ассоциировать свойства (пары ключ-значение), в которых хранятся данные.
- Быстрый просмотр узлов и связей для поиска



Реляционные СУБД vs NoSQL

	SQL	NoSQL
Хранилище данных	Таблицы (строки и столбцы). Строка - информация об одной сущности, а столбцы - отдельные значения данных.	Различными модели хранения данных (документ, граф, ключ-значение, столбчатые данные).
Схемы и гибкость	Каждая запись соответствует определенной схеме. Изменение схемы требует изменение всей базы данных.	Схема данных является динамической и может меняться в любой момент времени. Запись может содержать не все столбцы.
Масштабируемость	В основном по вертикали (на одном сервере). Масштабирование на нескольких серверах возможно, но сложно.	По горизонтали (на нескольких серверах). Технологии NoSQL могут автоматически распределять данные по серверам.
Соответствие ACID	Да.	Не полностью (ради производительности и масштабируемости).

Реляционные БД vs NoSQL

Реляционные СУБД подходят для проектов, где данные имеют сложные логические связи друг с другом и целостность данных имеет большое значение.

NoSQL СУБД подходят для проектов с большим объемом данных, который можно легко разделить на отдельные самостоятельные объекты. Высокая скорость и масштабирование.

Применимость NoSQL

