Лекция 8.

Базы данных NoSQL

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).

Firebird

PostgreSQL

Microsoft SQL Server

MySQL

ORACLE DATABASE

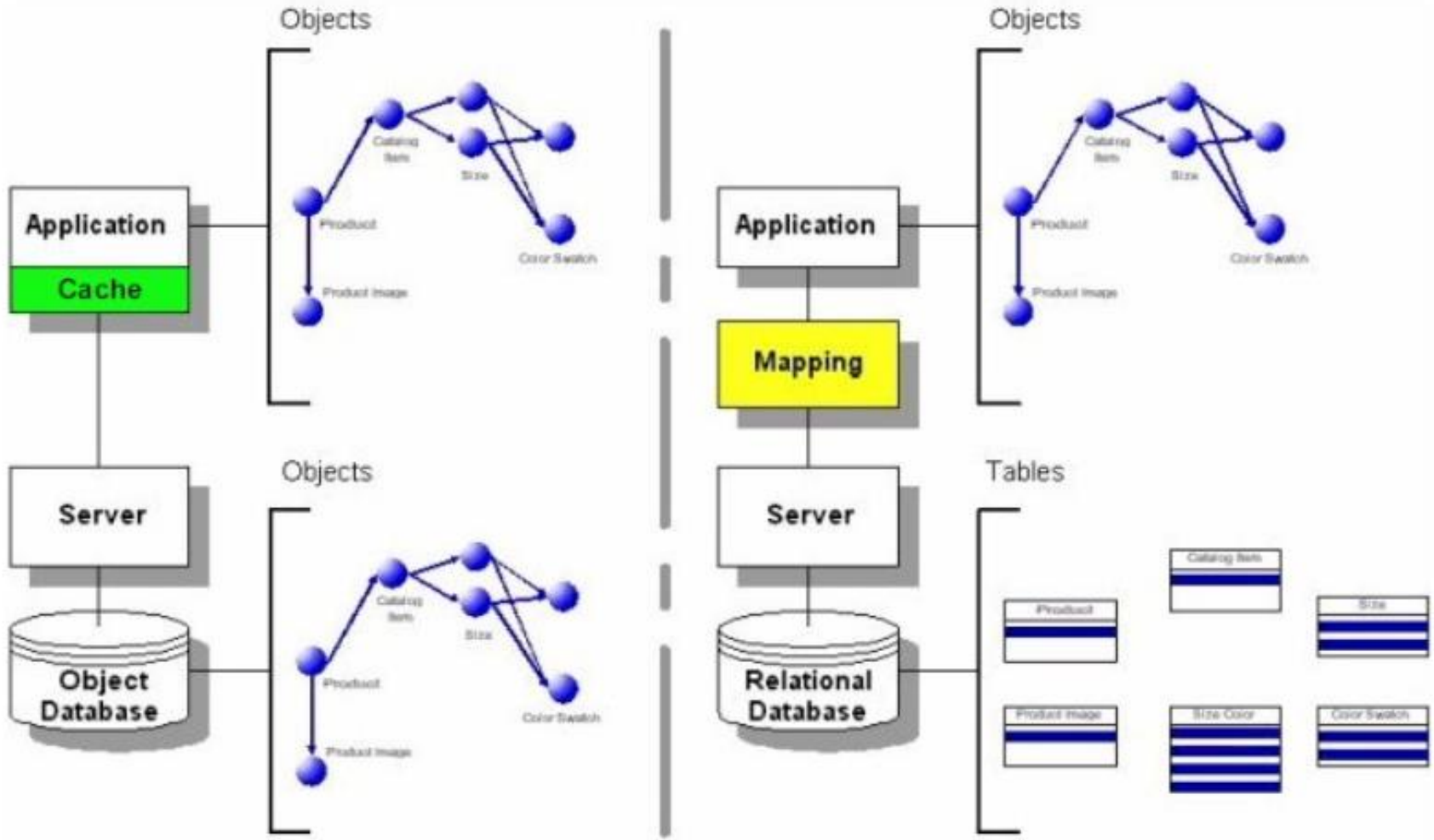
Реляционные СУБД

Чем могут не устраивать?

- 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 (горизонтальная масштабируемость).
- Падение производительности в больших распределенных системах из-за необходимости поддерживать согласованность данных + проблемы с устойчивостью к физическому разделению узлов.

Объектные БД и объектно-реляционные проекции



Объектно-реляционные проекции (ORM)

SQL-запрос в программе

```
Query1.Open("SELECT * FROM task_queue WHERE id_task IN (2, 3,
15) AND id task origin = 10");
foreach (DataRow row in Query1.Rows)
{
  ...
}
```

Запрос через ORM

```
IList<TaskQueue> queues = session.CreateCriteria()
  .Add(Expression.In("Task.Id", someTasks.ToArray()))
  .Add(Expression.Eq("TaskOrigin.Id", 10))
  .List<TaskQueue>();
foreach (TaskQueue queue in queues)
{
  ...
}
```

Для простых запросов (CRUD) ORM подходит хорошо:

- Более строгая типизация результатов.
- Простота работы с одиночными объектами вместо строк набора данных.
- Простая навигация по связям.

```
String cityName = order.Customer.Address.City.Name;
```

Более сложные запросы – код резко усложняется

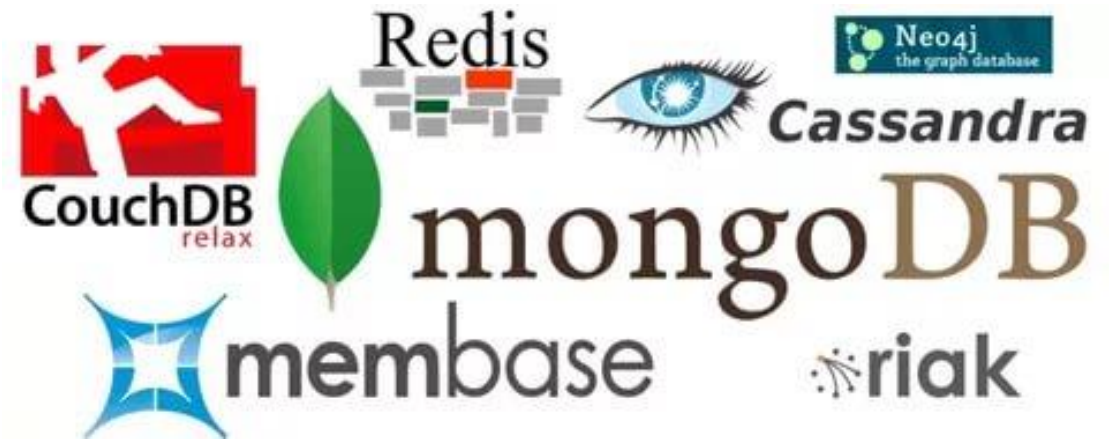
NoSQL СУБД

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



Отсутствие жесткой структуры объектов БД

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

Вместо строго структурированных таблиц имеем, например, набор разнородных документов.

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, ... }
  ]
}
```

Отсутствие жесткой структуры объектов БД

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.

Агрегирование данных (= денормализация)

Relational model

OrderItem

order_id	product_id	quantity
100	55	2
100	56	1

Product

id	name
55	Iphone 5
56	Ipad 3

Order

id	date
100	01.05.2012

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
{...}
```

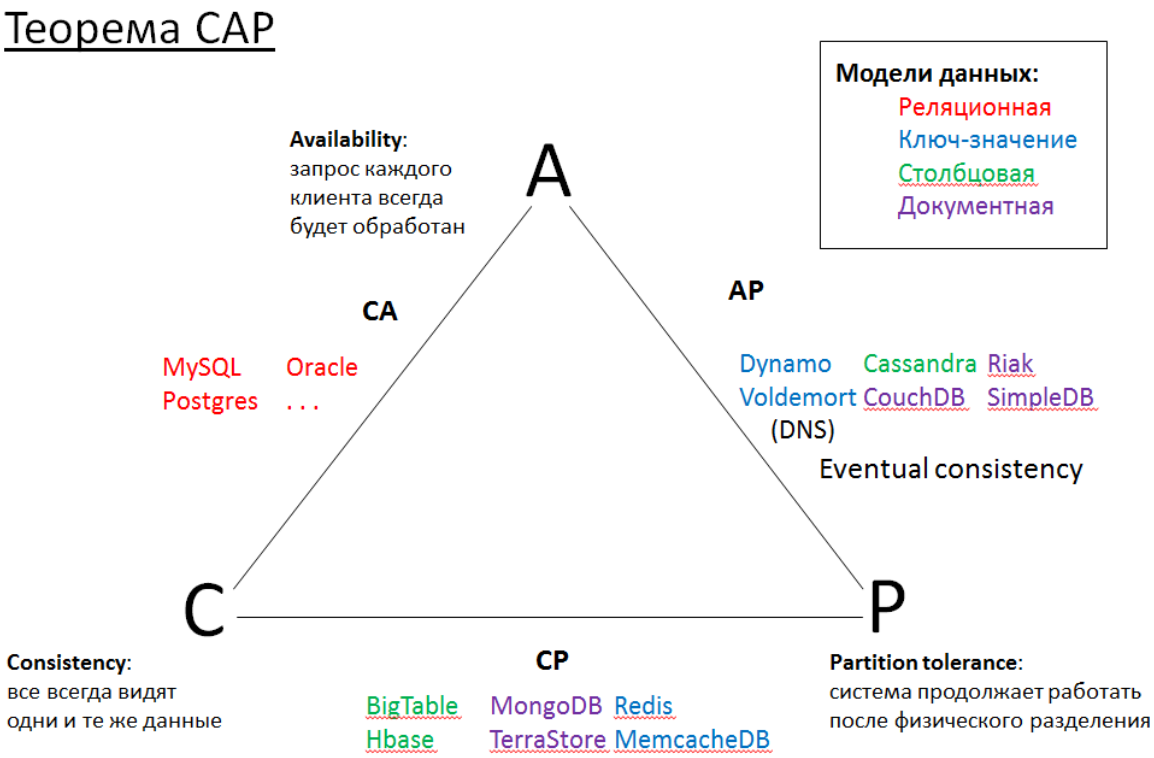
Агрегирование данных (= денормализация)

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

NoSQL СУБД

Причины появления – распределенность систем:

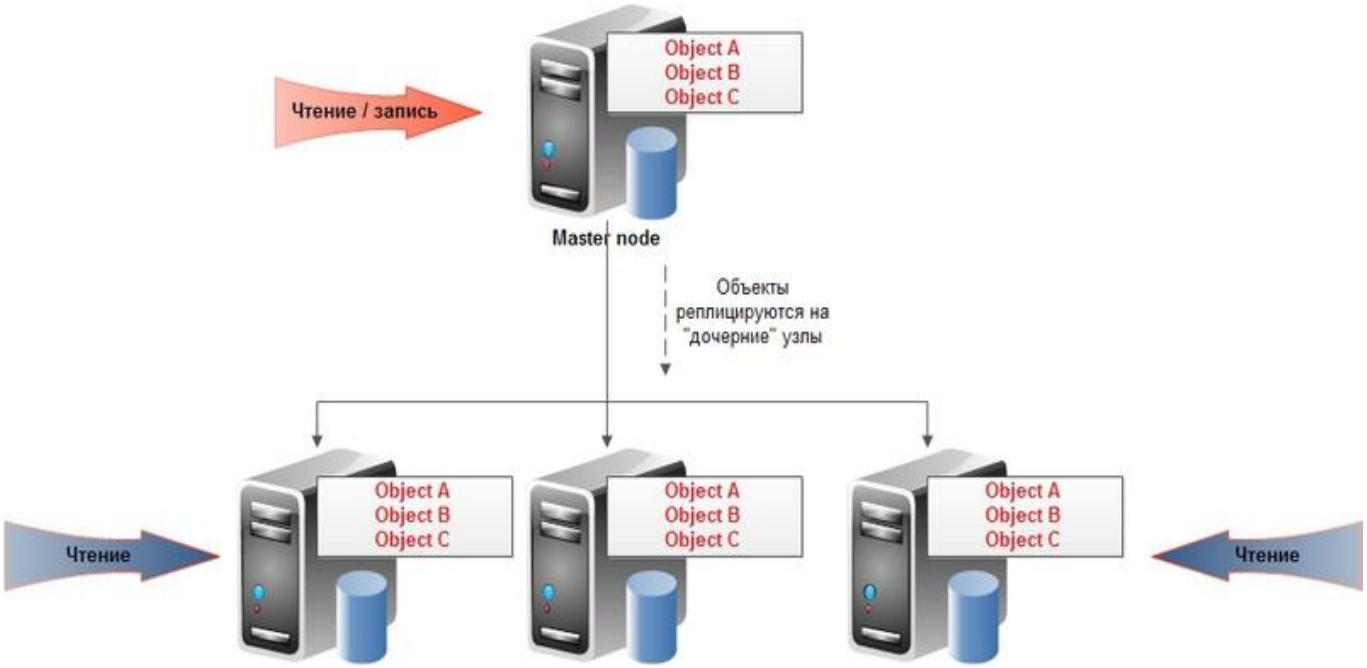
- Потребность в горизонтальной масштабируемости БД (путем добавления новых узлов).
- Сложность эффективной реализации транзакций в распределенной среде.



Борьба с большими объемами данных

Масштабирование БД для ускорения чтения + повышение надежности системы.

- **Репликация.**

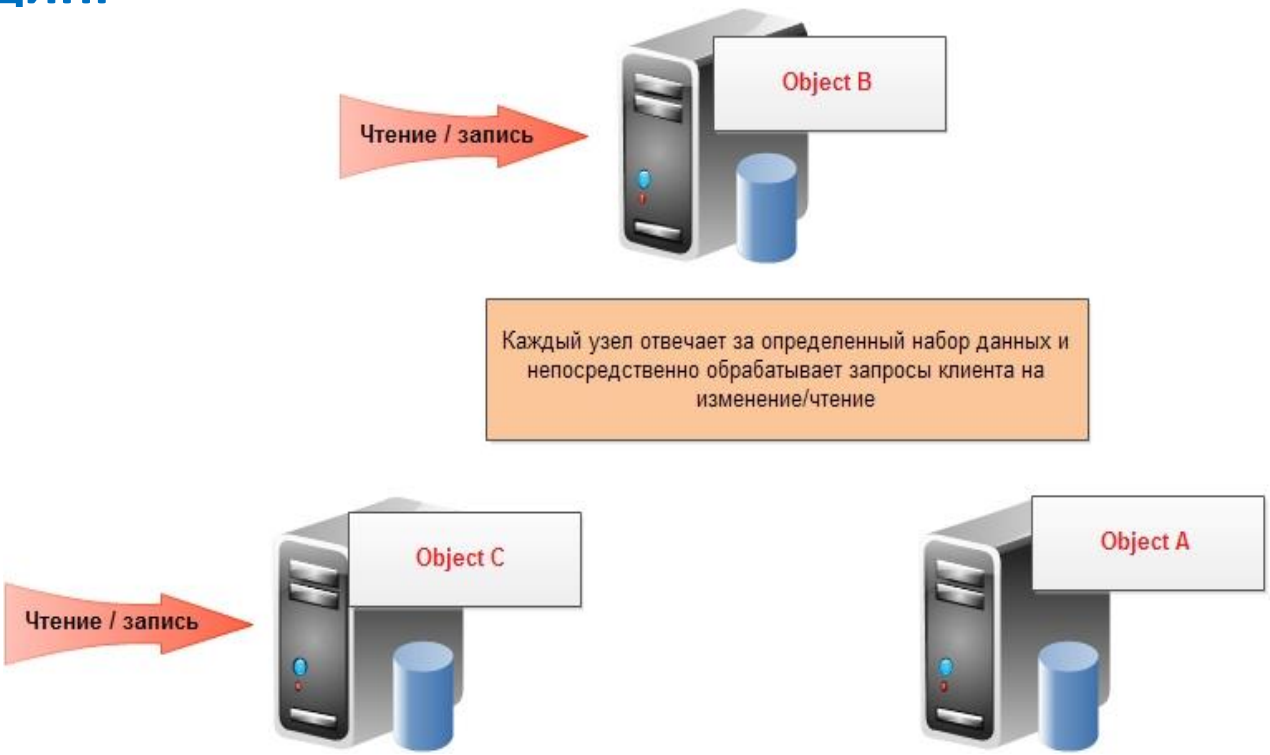


Синхронное обновление – увеличивается время ответа системы
Асинхронное обновление – есть время, когда реплики несогласованы

Борьба с большими объемами данных

Масштабирование БД для ускорения записи

- **Шардинг**



Возникают задачи:

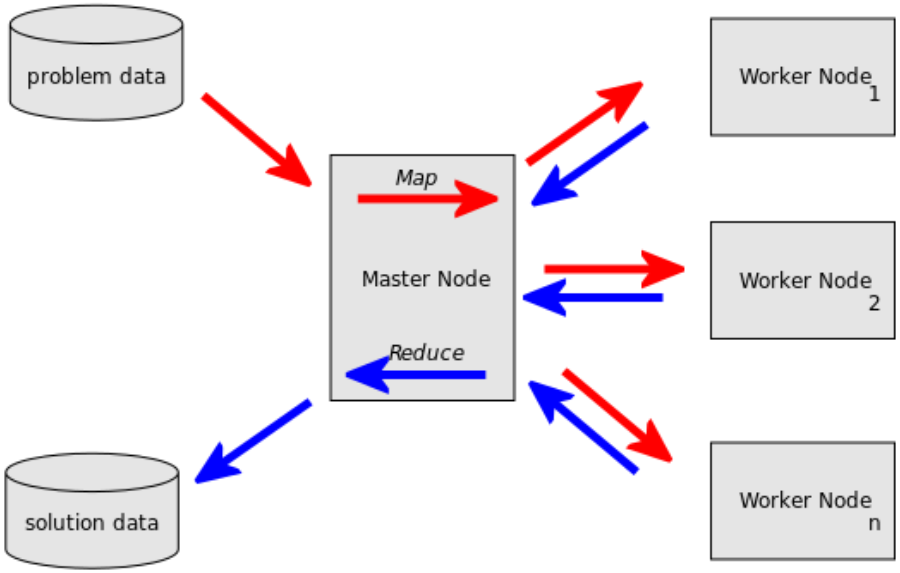
- Распределение данных по узлам.
- Балансировка нагрузки.
- Оптимизация загрузки сети.

MapReduce

Механизм параллельной обработки больших данных на кластерах.

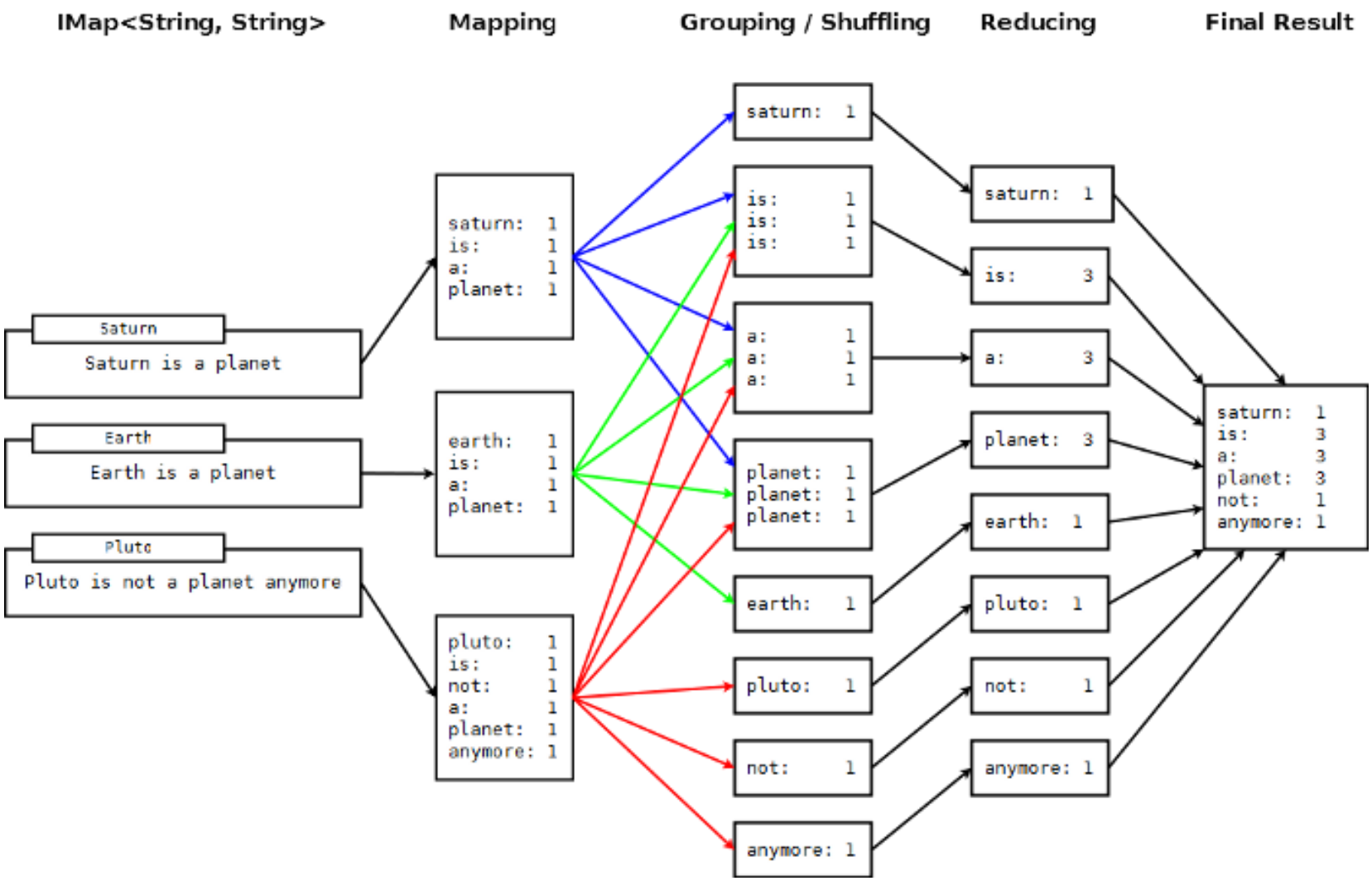
Map: Главный узел (master node) делит задачу на части, распределяя их по остальным машинам (worker nodes).

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



MapReduce

Пример: Подсчет количества слов во входном тексте



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).

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

redis

riak

FOUNDATIONDB

Модели данных NoSQL

Документ-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.

Документ-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

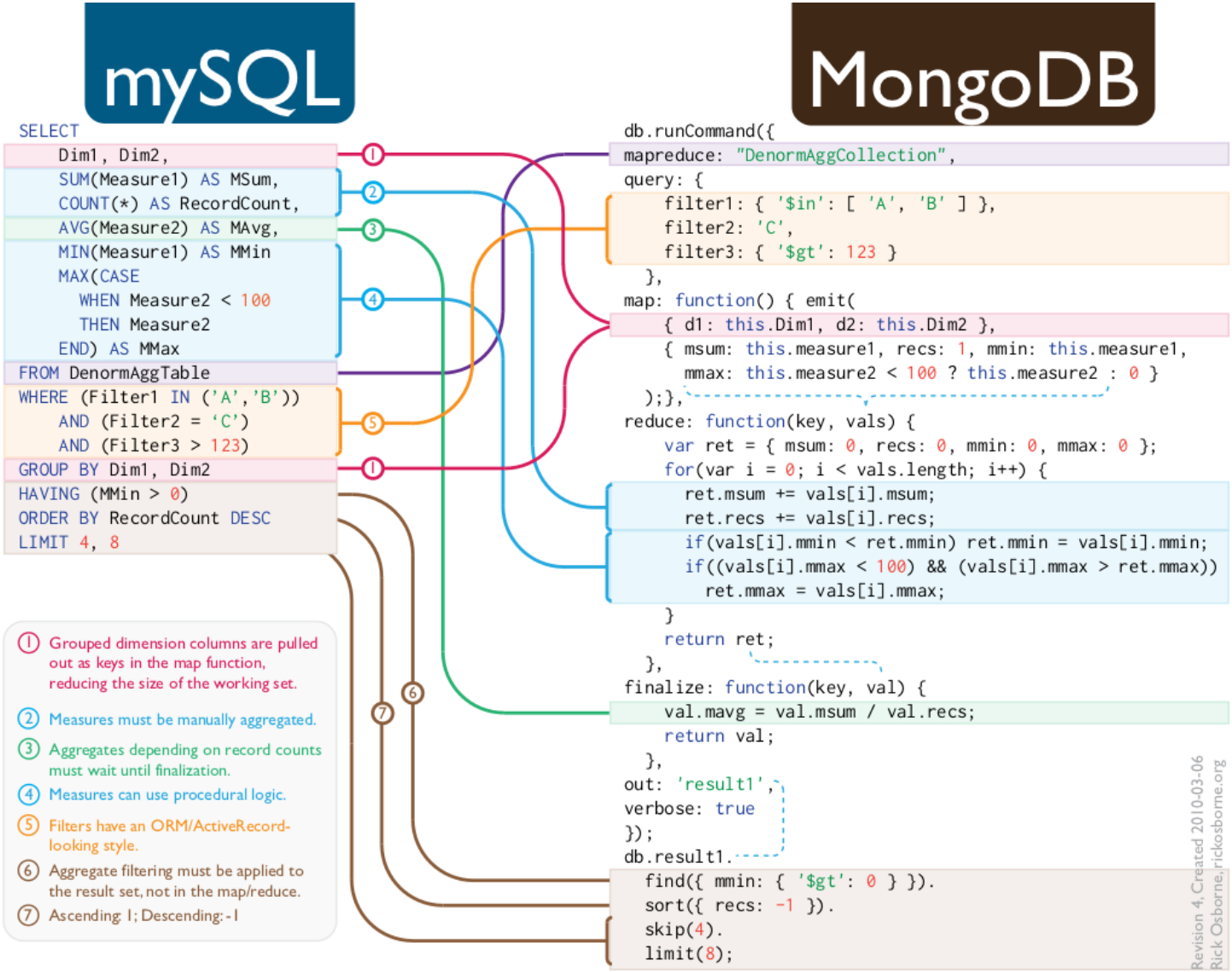
Исполнения товара

Id	Материал	Цвет
1	Сосна	Беж
1	ДСП	Белый

Габариты товара

Id	Ширина	Длина	Толщина
2	150	200	1.5

Документ-ориентированные модели



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.

Google BigTable

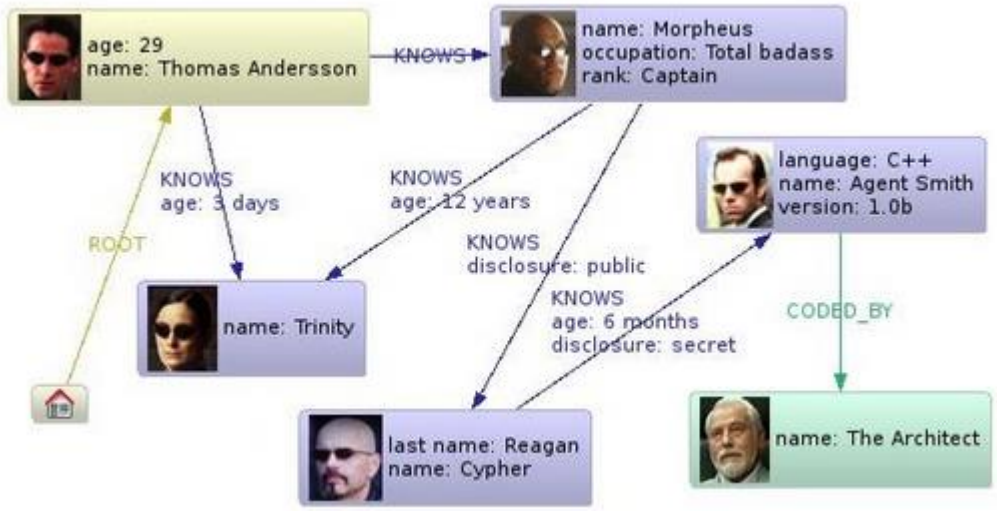
APACHE HBASE

cassandra

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 нужных данных.



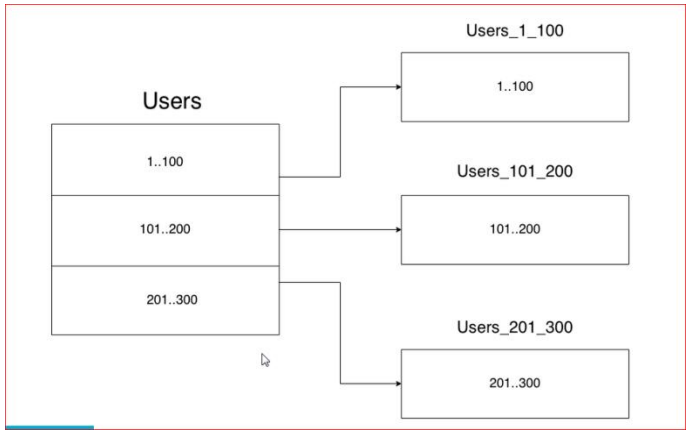
Neo4j

Применимость NoSQL

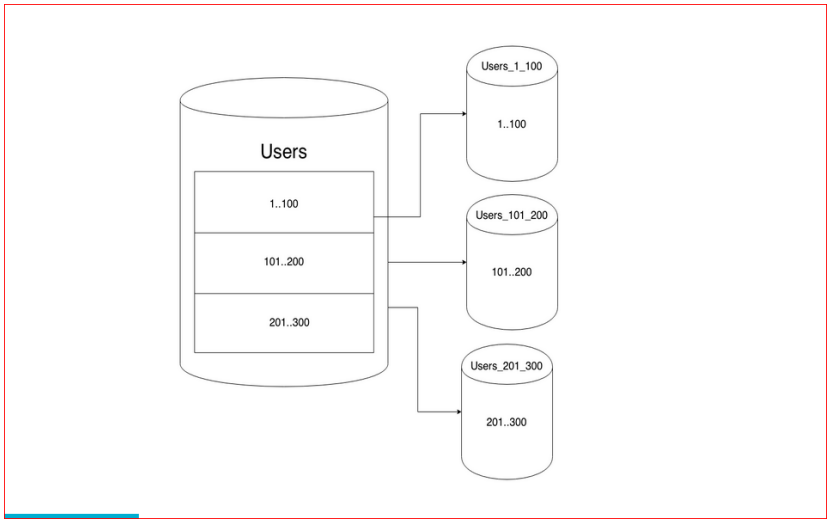
Борьба с большими объемами данных

Масштабирование БД для ускорения записи

- **Партицирование (вертикальный шардинг)**



- **Горизонтальный шардинг**



Возникают задачи:

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