

# *Интеллектуальные информационные системы*

## *NoSQL базы данных*

Кафедра управления и интеллектуальных технологий НИУ «МЭИ»  
2023 г.

# Типы баз данных

## Реляционные

- **Postgres**
- **MySQL**
- **SQLite**
- ...

## NoSQL базы данных в Hadoop:

- **HBase**
- **Cassandra**
- ...

# *Типы баз данных*

**Как хранить данные, чтобы взаимодействие с ними было максимально эффективным?**

- **Память**
- **Файлы**
- **Базы Данных**

# Память

- Высокая скорость
  - Быстрый доступ к данным – чтение, изменение, дополнение
  - Широкий выбор стандартных типов данных и возможность создавать свои типы
- 
- Ограниченность размером оперативной памяти
  - Данные существуют, пока существует процесс
  - Как реализовать общий доступ к данным?

## Примеры:

- Старые игровые приставки
  - Компиляторы
  - Текстовые редакторы
- 
- **Высокая скорость / Низкая надежность**

# Файлы

- Большой объем информации
  - Свобода в выборе структуры данных
  - Стандартные механизмы доступа к файлам
  - Доступ к данным напрямую, без «посредника»
- 
- Сложно вносить изменения в произвольное место файла
  - Тяжело организовать совместный доступ разными процессами
  - Отсутствие контроля целостности данных
- 
- Игры на ПК – при этом нет одновременного доступа, нет конфликтов одновременной записи
  - Поисковый индекс – только чтение данных

# Реляционные БД

- Целостность данных (ACID – atomicity, consistency, isolation, durability)
- Структурный аспект
- Стандартизованная обработка данных (SQL, реляционная алгебра)



# ACID

**Atomicity** – Атомарность

**Consistency** - Согласованность

**Isolation** - Изолированность

**Durability** – Надежность

# Реляционные БД

Реляционные БД имеют широкую область применения.

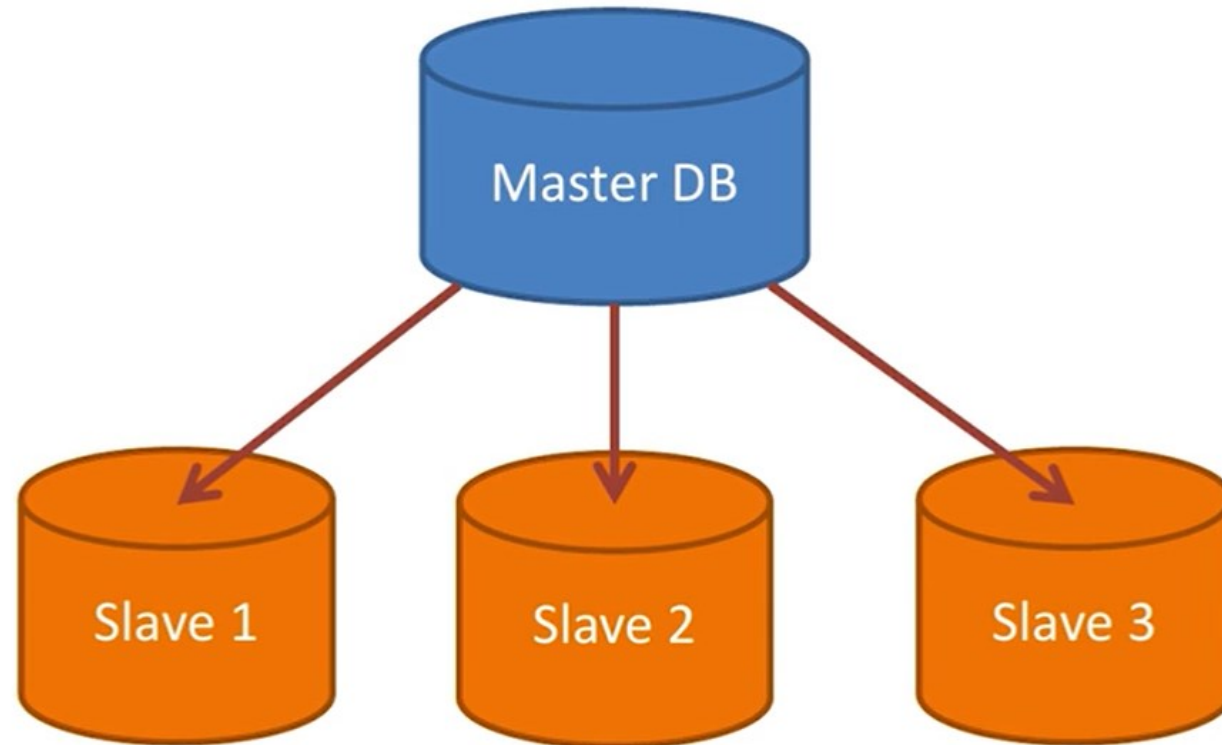
Плюсы:

- Универсальный язык работы с данными
- Контроль целостности (ACID)
- Одновременная работы с данными
- Безопасность работы с данными

Минусы:

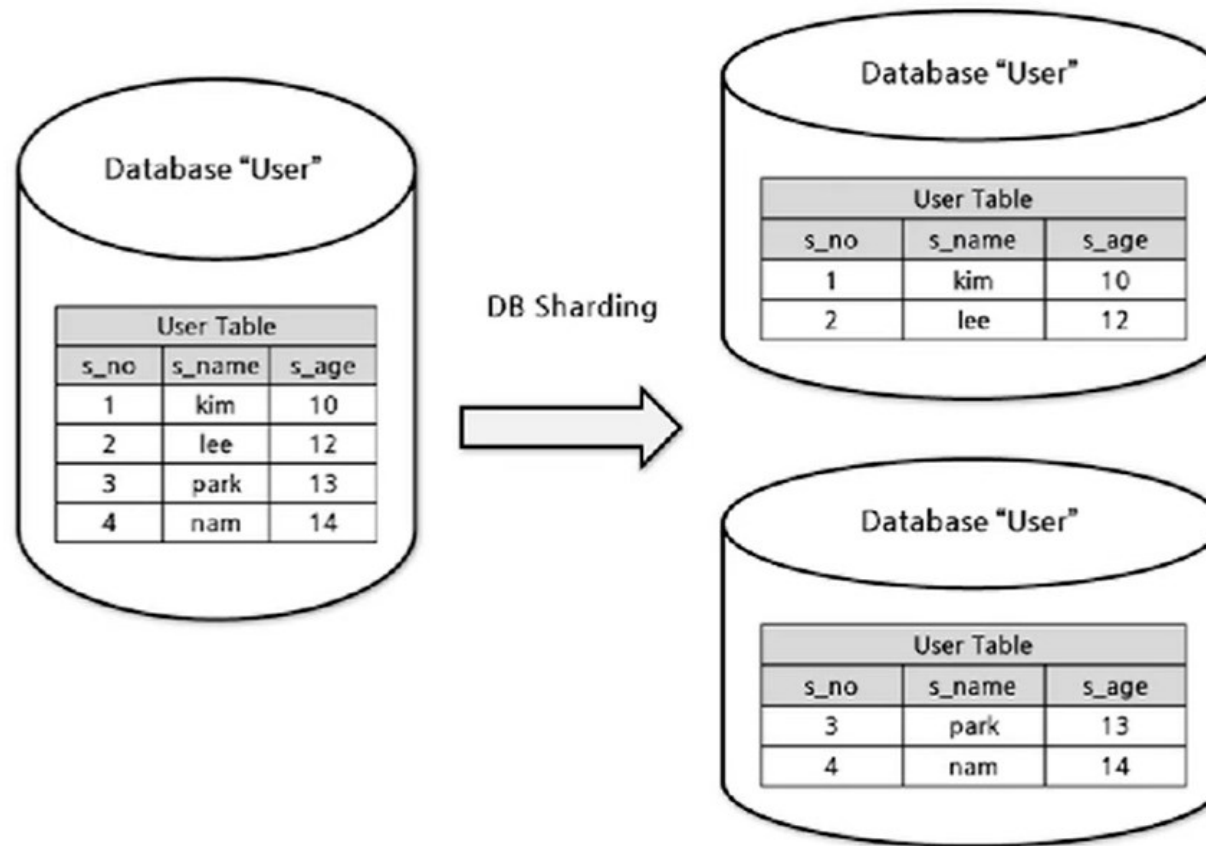
- Проблемы с масштабируемостью
- Подходят не для всех данных (только для реляционных)

# Масштабируемость РБД



Существует синхронная и асинхронная репликация.  
Синхронная надежнее, асинхронная - быстрее

# Масштабируемость РБД - шардинг



Разделение данных на сегменты по некоторому правилу\функции: по ключу, по значению, по хэшу,...

# *NoSQL базы данных*

Реляционные СУБД хранят отношения между объектами и обеспечивают целостность, но они требуют четкой структуры. Кроме того РСУБД имеют проблемы масштабируемости.

Не реляционные СУБД (NoSQL - Not Only SQL), это семейство баз данных, которые не являются такими жестко структурированными и табличными, как реляционные БД, т.е. у них модель хранения, оптимизированная под конкретные типы хранимых данных.

NoSQL – это технология, а не конкретная СУБД.

# NoSQL базы данных

- Применение различных хранилищ данных
- Нефиксированная схема БД
- Горизонтальная масштабируемость
- Параллельная работа (многопроцессорность)

Google BigTable – 2004 год – проприетарная СУБД на основе GFS.

Apache Hbase – 2006 года – NoSQL СУБД с открытым исходным кодом.

# CAP теорема

- **C (consistency)** — согласованность. Каждое чтение дает самую последнюю запись.
- **A (availability)** — доступность. Каждый узел (не упавший) всегда успешно выполняет запросы (на чтение и запись). На какое-то время доступ к узлу может быть закрыт — узел недоступен
- **P (partition tolerance)** — устойчивость к разделению — можем разделить на части.

В любой распределенной системе могут выполняться только два из трех свойств

# CAP теорема

- **C (consistency)** — согласованность. Каждое чтение дает самую последнюю запись.
- **A (availability)** — доступность. Каждый узел (не упавший) всегда успешно выполняет запросы (на чтение и запись). На какое-то время доступ к узлу может быть закрыт — узел недоступен
- **P (partition tolerance)** — устойчивость к разделению — можем разделить на части.

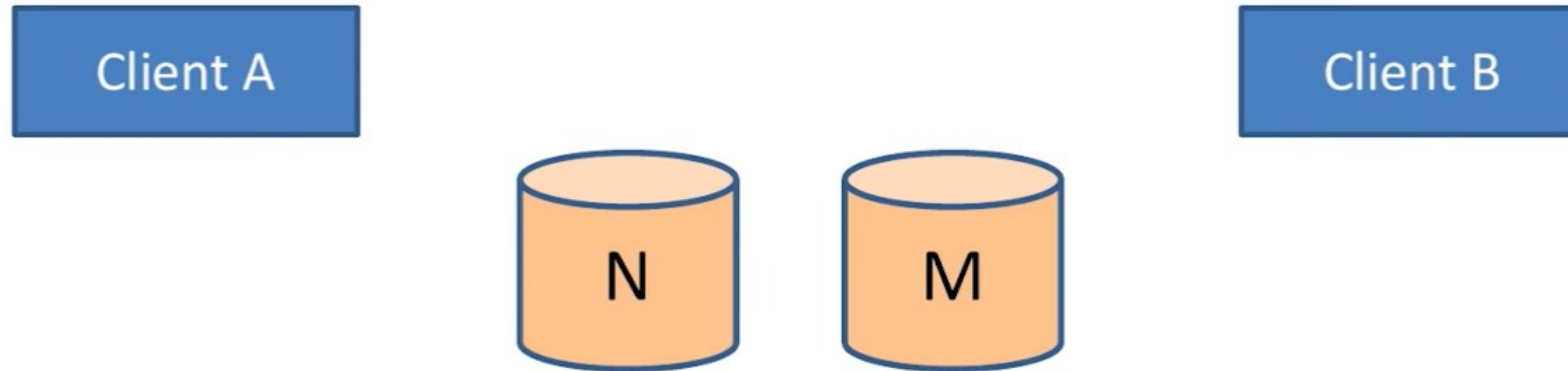
**CP** — закрываем какую-то часть узлов для доступа, чтобы данные пришли в непротиворечивое состояние.

**AP** — Выдвигаем требования к согласованности

# Модели согласованности

- **Строгая (*strict consistency*)**
- Последовательная (*sequential consistency*)
- Причинная (*causal consistency*)
- Процессорная (*processor consistency*)
- Слабая (*weak consistency*)
- **Консистентность в конечном счете (*eventual*)**
- Консистентность по выходу (*release consistency*)
- Консистентность по входу (*entry consistency*)

# Модели согласованности



**Видит ли клиент В записанные данные от А?**

- Strict Consistency: *Да*
- Eventual consistency: *Может быть*

**CAP-теорема: Strict consistency не может быть достигнута одновременно с availability и partition tolerance**

# BASE

## BASE на замену ACID:

**Basic Availability.** Система отвечает на любой запрос, но этот ответ может содержать ошибку или несогласованные данные.

**Soft-state.** Состояние системы может меняться со временем из-за изменений конечной согласованности.

**Eventual consistency** (конечная согласованность). Система, в конечном итоге, станет согласованной. Она будет продолжать принимать данные и не будет проверять каждую транзакцию на согласованность.

# Виды NoSQL СУБД

**Key-value** (модель данных – хэш-таблица):

- **Amazon DynamoDB**
- **Redis**

**Column-based** (модель данных – разреженная матрица) :

- **Hbase**
- **Cassandra**
- **ClickHouse**

# *Виды NoSQL СУБД*

**Document-based** (модель данных – дерево):

- **MongoDB**

**Graph-based**(модель данных – граф) :

- **OrientDB**