

Лекция 4.

**Файловые системы и СУБД.  
Классификация и выбор СУБД**

## Фундаментальная (и сложная) задача

**Как надежно, эффективно и удобно хранить данные на внешних носителях?**

# Как надёжно, эффективно и удобно хранить данные на внешних носителях?

1. **USB-флешки** — используют **NAND-флеш-память** (полупроводниковую энергонезависимую память), где данные хранятся в транзисторах с плавающим затвором.
2. **Внешние SSD (твёрдотельные накопители)** — также основаны на **NAND-флеш-памяти**, но с более мощным контроллером, обеспечивающим высокую скорость, износостойкость и коррекцию ошибок.
3. **Внешние HDD (жёсткие диски)** — используют **магнитную запись** на вращающихся металлических или стеклянных пластинах, где данные кодируются изменением направления намагниченности участков поверхности.
4. **Оптические диски (CD, DVD, Blu-ray, M-DISC)** — применяют **оптическую запись**: данные записываются и считываются с помощью лазера, изменяющего или распознающего отражающие свойства поверхности диска.
5. **Магнитные ленты (например, LTO)** — используют **последовательную магнитную запись** на гибкую пластиковую ленту с магнитным покрытием, аналогично старым кассетам, но с гораздо большей плотностью и надёжностью.

# Абстракция для борьбы со сложностью

Сложность возникает, когда система содержит слишком много деталей, взаимосвязей или уровней взаимодействия, чтобы человек мог эффективно с ней работать. Абстракция помогает спрятать ненужные детали и сосредоточиться только на том, что важно на данном уровне рассмотрения.

---

## Как это работает?

Абстракция разделяет "что" и "как":

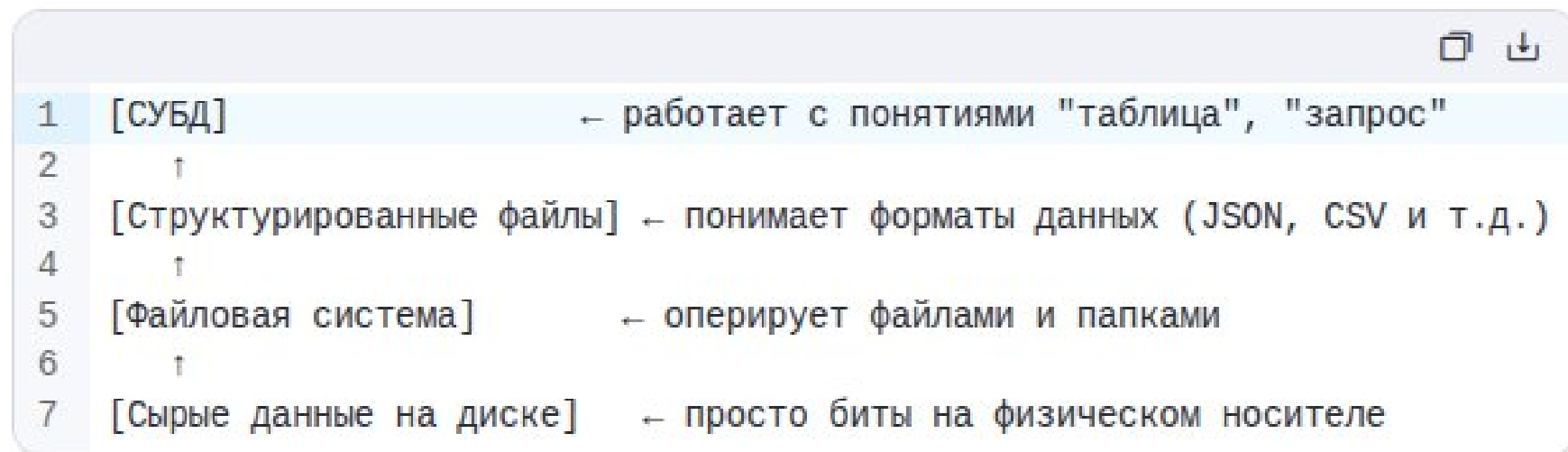
- "Что" — это интерфейс, поведение, функциональность (например, "прочитать файл").
- "Как" — это реализация, внутренние механизмы (например, как именно ОС обращается к секторам диска).

Пользователь или разработчик работает с "что", не заботясь о "как".

# Пирамида абстрагирования при работе с данными

Иерархическая модель, показывающая, как данные проходят путь от физического хранения до высокоуровневого логического представления, удобного для пользователя или приложения.

Каждый уровень скрывает сложность нижележащего и предоставляет более удобный интерфейс:



Эта иерархия позволяет разработчикам и пользователям работать с данными на высоком уровне, не задумываясь о том, как именно они записаны на диск.

# Пирамида абстрагирования

- Упрощение разработки: программист не должен управлять секторами диска.
- Повышение надёжности: ошибки на одном уровне не обязательно влияют на другие.
- Гибкость: можно менять физическое хранение (например, перейти с HDD на SSD), не меняя приложения.
- Масштабируемость: СУБД может распределять данные по множеству файлов или серверов, оставаясь прозрачной для пользователя.

[СУБД] ← работает с понятиями "таблица", "запрос"

↑

[Структурированные файлы] ← понимает форматы данных (JSON, CSV и т.д.)

↑

[Файловая система] ← оперирует файлами и папками

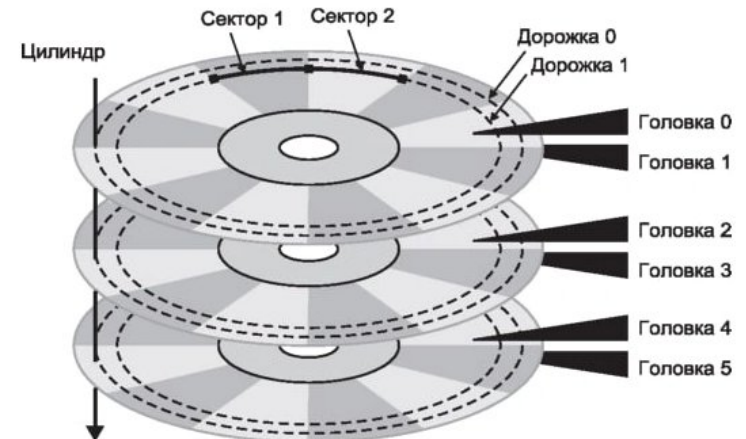
↑

[Сырые данные на диске] ← просто биты на физическом носителе

# 1. Сырые данные на внешнем носителе

Уровень: Физический (аппаратный)

- **Что это:** На самом низком уровне данные представлены в виде последовательностей битов, записанных на физическом носителе — жёстком диске (HDD), SSD, флешке и т.д.
- **Особенности:**
  - Нет никакой структуры: просто последовательность 0 и 1.
  - Доступ осуществляется через низкоуровневые команды контроллера диска.
  - Операционная система (ОС) не "понимает", что это за данные — они просто хранятся в определённых секторах.



[СУБД] ← работает с понятиями "таблица", "запрос"

↑

[Структурированные файлы] ← понимает форматы данных (JSON, CSV и т.д.)

↑

[Файловая система] ← оперирует файлами и папками

↑

[Сырые данные на диске] ← просто биты на физическом носителе

# Файловая система (система управления файлами)

Уровень: Логический (ОС-уровень)

- **Что это:** Операционная система организует сырые данные в виде файлов и папок, используя файловую систему (например, NTFS, ext4, FAT32).
- **Абстракция:**
  - Скрыта физическая адресация (сектора, блоки).
  - Появляются понятия **файла**, каталога, имени, размера, прав доступа.
  - Файлы могут быть организованы в древовидную структуру.
- **Преимущество:** пользователь и программы работают не с секторами, а с именованными объектами.



# Функции файловой системы

---

1. **Хранение неструктурированных данных**
  - Файл — это поток байтов без внутренней семантики.
2. **Организация данных в иерархию каталогов**
  - Дерево папок и файлов с именами.
3. **Управление дисковым пространством**
  - Выделение/освобождение блоков (кластеров), отслеживание свободного места.
4. **Метаданные файла**
  - Имя, размер, дата создания/изменения, права доступа, владелец.
5. **Надёжность и восстановление**
  - Журналирование (ext4, NTFS), проверка целостности (fsck, chkdsk).

# Функции файловой системы

---

6. Безопасность на уровне файлов/каталогов
  - Права чтения/записи/выполнения (Unix: rwx; Windows: ACL).
7. Произвольный и последовательный доступ
  - Чтение/запись по смещению (offset).
8. Кэширование и буферизация
  - Ускорение доступа через RAM-кэш.
9. Абстракция от физического устройства
  - Программы не знают, где именно на диске лежат данные.
10. Поддержка монтирования и различных носителей
  - HDD, SSD, сетевые диски, образы и т.д.

[СУБД] ← работает с понятиями "таблица", "запрос"

↑

[Структурированные файлы] ← понимает форматы данных (JSON, CSV и т.д.)

↑

[Файловая система] ← оперирует файлами и папками

↑

[Сырые данные на диске] ← просто биты на физическом носителе

# Структурированные файлы

---

Уровень: Прикладной (формат данных)

- **Что это:** Файлы содержат данные, организованные по определённому формату, понятному программам.
- **Форматы:** CSV, JSON, XML, Parquet, Avro и др.
- **Абстракция:**
  - Данные уже имеют **логическую структуру**: поля, записи, таблицы, иерархии.
  - Программы могут парсить такие файлы и извлекать смысл.
- **Преимущество:** данные становятся **интерпретируемыми**, а не просто байтами.

[СУБД] ← работает с понятиями "таблица", "запрос"

↑

[Структурированные файлы] ← понимает форматы данных (JSON, CSV и т.д.)

↑

[Файловая система] ← оперирует файлами и папками

↑

[Сырые данные на диске] ← просто биты на физическом носителе

# Система управления базами данных

Уровень: Логический/концептуальный (базы данных)

- **Что это:** СУБД (например, PostgreSQL, MySQL, MongoDB) управляет структурированными данными, обеспечивая их хранение, запросы, целостность, безопасность и т.д.
- **Абстракция:**
  - Данные организованы в **таблицы, индексы, представления, сущности**.
  - Используется **декларативный язык запросов** (например, SQL), где пользователь описывает *что* нужно, а не *как* это получить.
  - СУБД сама решает, как физически хранить и извлекать данные.
- **Преимущества:**
  - Поддержка транзакций, ACID-свойств.
  - Многопользовательский доступ.
  - Оптимизация запросов.
  - Независимость от физического хранения.

# Функции СУБД

- 1. Хранение структурированных данных**
  - Данные организованы в таблицы, документы, графы и т.д. с чёткой схемой (типы полей, связи).
- 2. Управление данными через декларативные запросы**
  - Поддержка языков запросов (например, SQL) для поиска, фильтрации, агрегации и модификации данных без указания *как* это делать.
- 3. Обеспечение целостности данных**
  - Ограничения (constraints): уникальность, NOT NULL, внешние ключи (foreign keys), проверки (CHECK).
  - Поддержка ссылочной и доменной целостности.

# Функции СУБД

## 4. Транзакционность (ACID)

— **Atomicity** (атомарность), **Consistency** (согласованность), **Isolation** (изоляция), **Durability** (долговечность).

— Гарантия корректности даже при сбоях или параллельном доступе.

## 5. Многопользовательский доступ и управление параллелизмом

— Блокировки, уровни изоляции транзакций, управление конфликтами при одновременной работе многих пользователей.

## 6. Безопасность и управление доступом

— Гранулярные права: на уровне базы, таблицы, столбца, строки.

— Аутентификация, авторизация, аудит.

# Функции СУБД

7. **Оптимизация запросов**
  - Планировщик запросов выбирает наиболее эффективный способ выполнения (используя индексы, статистику и т.д.).
8. **Резервное копирование и восстановление**
  - Логическое и физическое резервное копирование, point-in-time recovery, журналы транзакций (WAL).
9. **Поддержка масштабируемости и отказоустойчивости**
  - Репликация, шардирование, кластеризация, автоматическое переключение при отказе.
10. **Абстракция от физического хранения**
  - Пользователь работает с логической моделью; СУБД сама решает, как и где хранить данные (страницы, индексы, кэши).

# Эволюция абстракций

1. Сырые сектора диска →

2. Файловая система:

- объединяет сектора в файлы,
- даёт имена, каталоги, права,
- но не знает содержимого файла.

3. СУБД:

- интерпретирует содержимое,
- вводит **логическую модель данных** (таблицы, отношения),
- скрывает не только **физическое расположение**, но и **формат хранения** (индексы, страницы, журналы),
- предоставляет **декларативный язык запросов (SQL)**.

“💡 Таким образом, СУБД строится поверх файловой системы (обычно хранит свои данные в файлах), но добавляет новый уровень абстракции — логическую модель данных.”

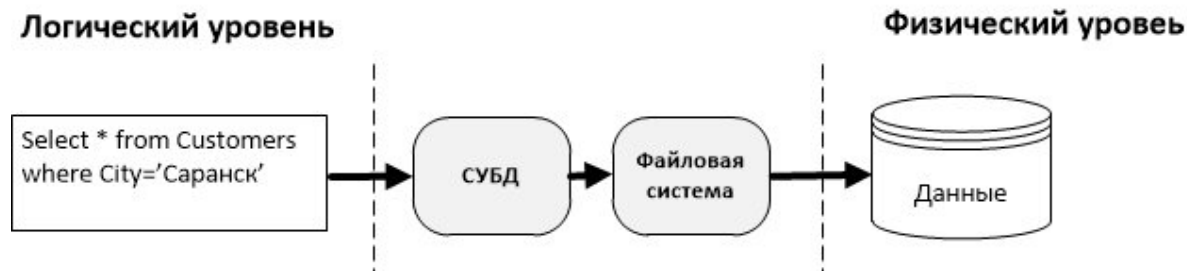


- Файловая система на логическом уровне унифицирует запись и считывание данных.
- СУБД на логическом уровне унифицирует операции с данными, учитывающие их структуру и содержание данных.

## Работа с файлом



## Работа с базой данных



# Различия файловых систем и СУБД

СУБД — не просто «продвинутая файловая система», потому что:

- Файловые системы оптимизированы под последовательный/произвольный доступ к большим блокам.
- СУБД оптимизированы под множество мелких операций, сложные связи, транзакции, аналитику.
- СУБД решают задачи, выходящие за рамки хранения:
  - оптимизация запросов,
  - репликация,
  - резервное копирование на логическом уровне,
  - безопасность на уровне строк/столбцов.

# Файловые системы и СУБД

АСПЕКТ	ФАЙЛОВАЯ СИСТЕМА	СУБД
Единица данных	Файл (поток байтов)	Запись в таблице (структурированный объект)
Структура	Иерархия каталогов	Реляционная / документная / графовая модель
Семантика	Нет — данные «непрозрачны»	Да — данные имеют тип, связи, ограничения
Запросы	Чтение/запись по имени	Высокоуровневые запросы (SQL, XPath и др.)
Целостность	Минимальная	Строгая (ограничения, транзакции, внешние ключи)
Параллелизм	Ограниченный	Полноценная поддержка (блокировки, изоляция)

# Файловые системы и СУБД

- **Файловая система** отвечает на вопрос:

“«Как сохранить и найти набор байтов по имени?» ”

- **СУБД** отвечает на вопрос:

“«Как хранить, связывать, запрашивать и защищать смысловые данные с гарантией корректности при сложных операциях?» ”



## **Важно помнить:**

- **СУБД** использует файловую систему как основу для хранения своих файлов (данных, журналов, индексов).
- **Файловая система** не может заменить **СУБД** при работе со сложными, связанными, часто изменяемыми данными.
- **СУБД** не нужна, если данные просты, редко меняются и не требуют сложных запросов (например, лог-файлы, изображения, архивы).

# Файловые системы и СУБД

## ✓ Заключение

СУБД и файловая система — дополняющие друг друга уровни абстракции в стеке хранения данных:

- **Файловая система** — фундамент: управляет *местом и именами*.
- **СУБД** — надстройка: управляет *смыслом, связями и логикой*.

Их совместная работа позволяет современным системам эффективно обрабатывать всё — от простых текстовых файлов до глобальных банковских транзакций.

# Классификация СУБД

## 1. По архитектуре:

- Централизованная (встроенная)
- Файл-сервер
- Клиент-сервер

## 2. По масштабу:

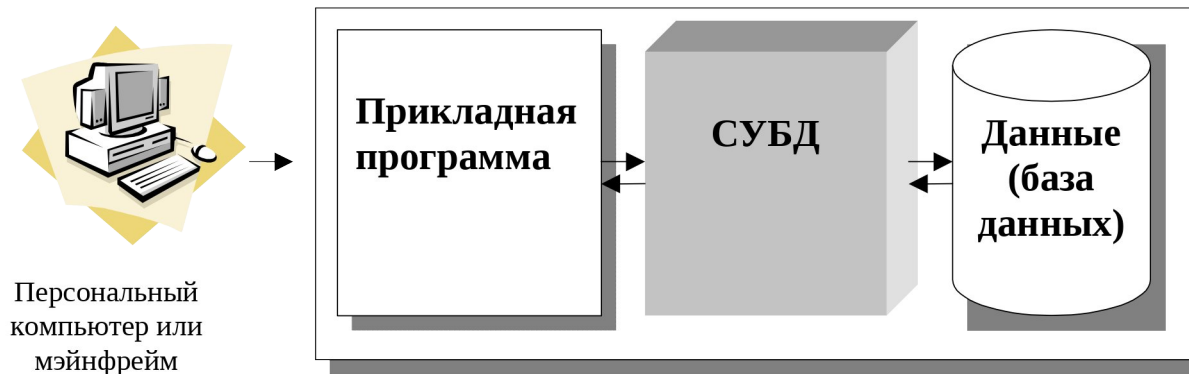
- Персональные (десктопные и облачные)
- Серверные

## 3. По способу структурирования данных:

- Реляционные
- Хранилища «ключ-значение»
- Столбцовые
- Документарные
- Графовые

# Централизованная архитектура. Встраиваемые СУБД

- База данных – набор файлов, находящихся на жестком диске компьютера.
- СУБД и приложение для работы с базой данных установлены на той же машине.
- Простейший случай (встроенная СУБД): БД — один файл, СУБД — библиотека, подгружаемая к прикладной программе.



Ранние СУБД для мэйнфреймов (IDMS, ...) и персоналок (DBase, ...).

# Централизованная архитектура. Встраиваемые СУБД

Microsoft Access,  
LibreOffice Base, Firebird,  
SQLite



## Преимущества:

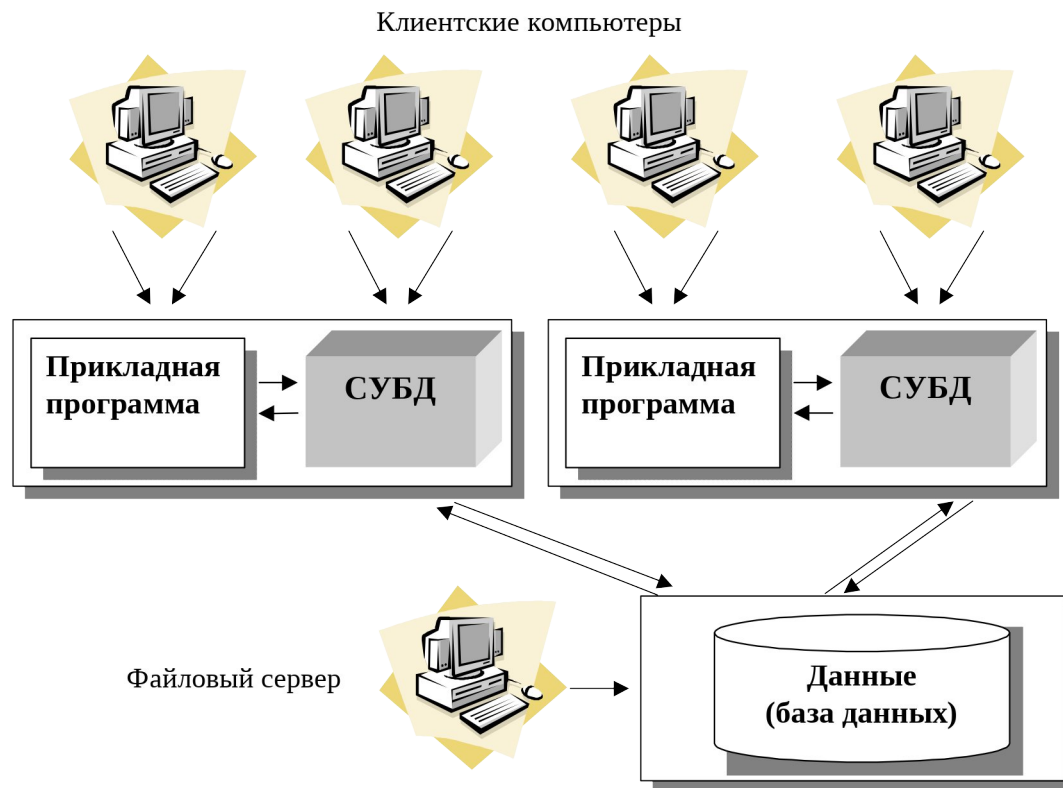
- Простота разработки и развертывания приложений.
- Простота обслуживания локальной БД.
- Высокое быстродействие на простых операциях считывания и модификации одиночных записей.

## Недостатки:

- Высокий риск потери или повреждения данных.
- Невозможность распределения вычислительной нагрузки.

# Архитектура «файл-сервер»

- База данных – набор файлов на выделенном сервере.
- СУБД устанавливается на клиентах.
- Центральный сервер выполняет в основном только роль хранилища файлов, не участвуя в обработке самих данных.



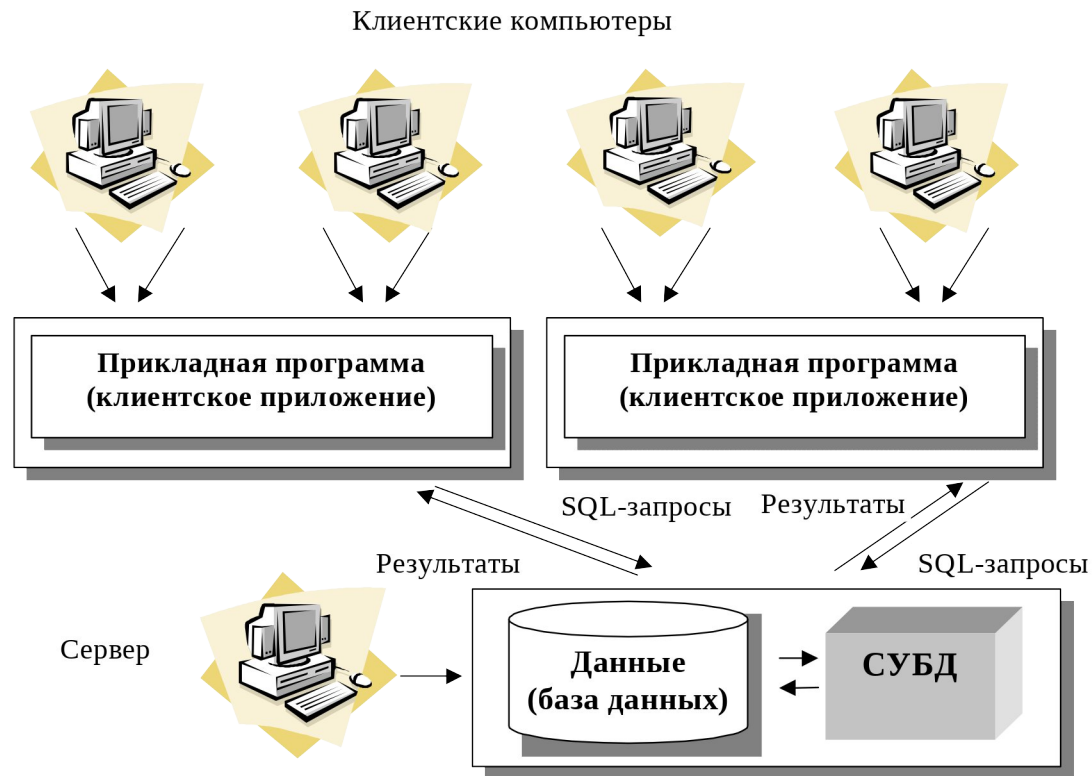
# Архитектура «файл-сервер»

## Недостатки

- Резкое падение производительности при обращении многих пользователей к одним и тем же данным (блокировка файла на время выполнения медленных сетевых операций).
- Данные обрабатываются на клиентских машинах – большой сетевой трафик (файлы с данными полностью копируются на клиентов) и загрузка мощностей клиентов.
- Низкий уровень безопасности (только на уровне файловой системы на сервере). *Несанкционированный доступ, внесение ошибочных изменений.*

# Архитектура «клиент-сервер»

- База данных и СУБД (интеллектуальная) на выделенном сервере в сети или облачном ресурсе (DBaaS).
- На клиентах стоят приложения. К серверу посылаются только текст запросов на специальном языке.



MySQL, Microsoft SQL Server, PostgreSQL, MongoDB ...

# Разделение функций

## Клиент

- Формирование и посылка запросов серверу (на специальном языке или через API).
- Интерпретация результатов запросов, полученных от сервера (*зависит от языка программирования*).
- Пользовательский интерфейс.

## Сервер

- Прием запросов от клиентов, их интерпретация, оптимизация и выполнение.
- Отправка результатов приложению-клиенту.
- Обеспечение системы безопасности и разграничение доступа.
- Управление целостностью БД.
- Реализация стабильности многопользовательского режима работы

# Преимущества «клиент-серверной» архитектуры

1. Уменьшается сетевой трафик.
2. Повышается целостность и безопасность БД.
3. Уменьшается сложность клиентских приложений.

## Преимущества «клиент-серверной» архитектуры

1. Уменьшается сетевой трафик.
2. Повышается целостность и безопасность БД.
3. Уменьшается сложность клиентских приложений.

**Проблема:** Могут быть трудности с обновлением ПО на клиентах.

# Преимущества «клиент-серверной» архитектуры

1. Уменьшается сетевой трафик.
2. Повышается целостность и безопасность БД.
3. Уменьшается сложность клиентских приложений.

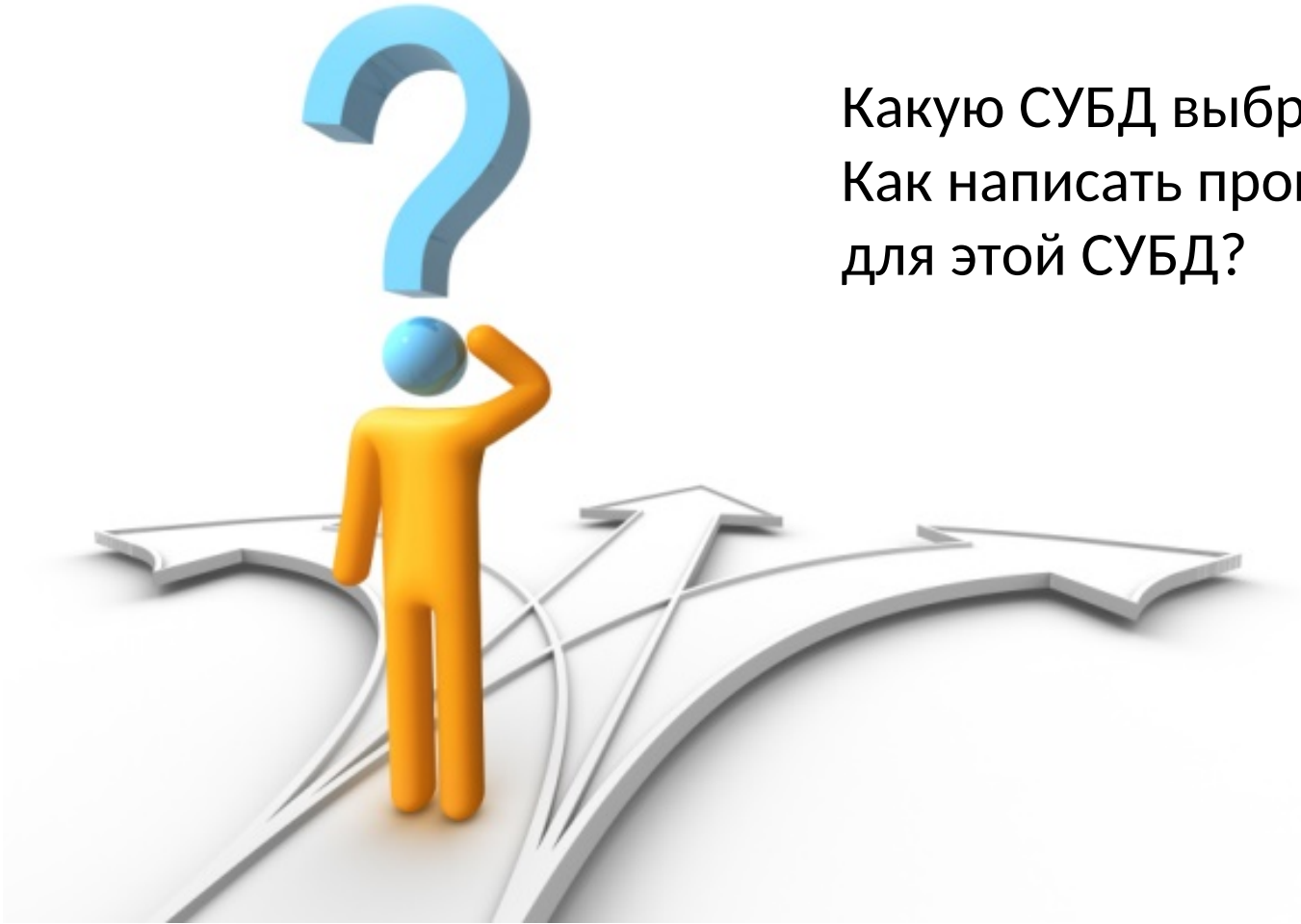
**Проблема:** Могут быть трудности с обновлением ПО на клиентах.

**Решение:** Клиент должен быть «тощим» (в идеале – только веб-браузер для отображения интерфейса пользователя). Бизнес-логика приложений выносится на отдельный сервер.



## Есть задача с данными

Какую СУБД выбрать?  
Как написать программу  
для этой СУБД?



## Выбор СУБД по масштабу задачи

### Персональные (desktopные или облачные):

- Для небольших простых задач.

### Серверные:

- Для средних и крупных задач, где важна производительность и надежность (либо кроссплатформенность).

## Персональные СУБД

- Для небольших простых задач.
- Упрощенная архитектура и функциональность.
- Простые и удобные визуальные средства для быстрой разработки приложений (даже без программирования).

# Электронные таблицы с элементами СУБД

- Microsoft Excel (*Windows, платная*)
- Google таблицы (*Облачная, бесплатная*)

*fx* =QUERY(Sales!\$A:\$D, "SELECT A, SUM(C) GROUP BY A PIVOT B ORDER BY A")

	A	B	C	D
1	Sales By Manager Monthly			
2	Manager	201601	201602	201603
3	Greg Lestrage	£4,144.00	£6,397.00	£3,344.00
4	John Watson	£3,500.00	£2,678.00	
5	Mrs Hudson	£6,654.00	£377.00	
6	Mycroft Holmes	£14,210.00	£88.00	
7	Sherlock Holmes	£1,000.00	£5,000.00	

В7 Нектарин

A	B	C
1	Форма ввода данных о пр...	
2		
3	Дата	08.09.2016 11:20
5	Клиент	Дубинин
7	Товар	Нектарин
9	Количество	3
11	Цена	90
13	Стоимость	270
14		
15		
16		
17		
18		

Проверка вводимых значений

Параметры Сообщение для ввода Сообщение об ошибке

Условие проверки

Тип данных:

Игнорировать пустые ячейки

Значение:

Список допустимых значений

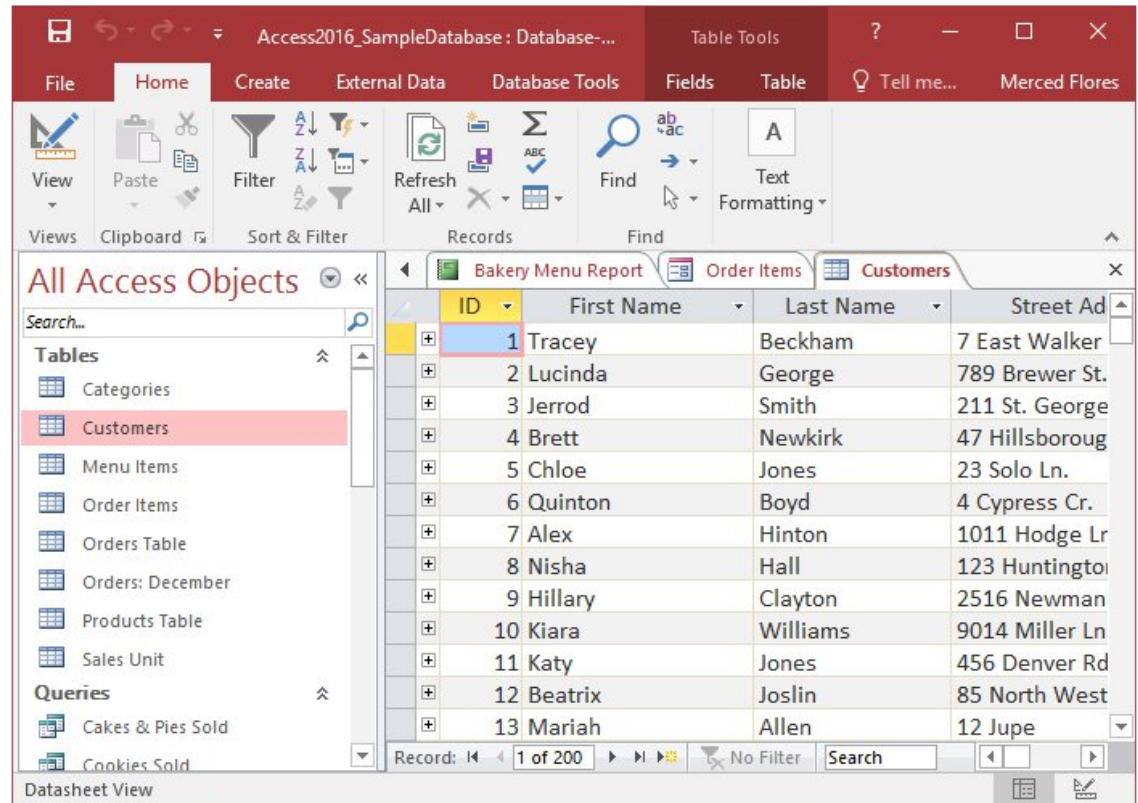
Источник:

Распространить изменения на другие ячейки с тем же условием

Очистить все

# Десктопные СУБД

- Microsoft Access (*Windows, платная*)
- LibreOffice Base (*Windows, Mac, Linux, бесплатная*)
- KEXI (*Windows, Mac OS, Linux, бесплатная*)
- Filemaker (*Mac OS, платная*)



# Облачные СУБД

- NuBuilder Forte
- Memento Database
- *Airtable (гибрид электронной таблицы и СУБД)*

The screenshot shows the Airtable interface for a table named 'Favorite Movies'. The table has the following columns: Name, Photo, Seen?, Actors, Director, Genre, Description, Personal Rating, and Personal Notes. The data is as follows:

	Name	Photo	Seen?	Actors	Director	Genre	Description	Personal Rating	Personal Notes
1	The Godfather		✓	Marlon Brando Al P...	Francis Ford Coppola	Drama	The Godfather is a 1972 American cri...	4: Entertaining 😊	
2	Sister Act			Whoopi Goldberg	Emile Ardolino	Comedy	Sister Act is a 1992 American comed...		
3	Pulp Fiction		✓	Samuel L. Jackson	Quentin Tarantino	Drama	Pulp Fiction is a 1994 American black...	3: Average 😐	
4	Caddyshack			Bill Murray	Zach McBride	Comedy	Caddyshack is a 1980 American sport...		
5	Get Smart		✓	Barbara Feldon Don	Gary Nelson	Comedy	Get Smart is an American comedy ...	2: Mediocre 😞	
6	Forrest Gump		✓	Tom Hanks	Robert Zemeckis	Drama	Forrest Gump is a 1994 American epi...	4: Entertaining 😊	
7	You've Got Mail			Meg Ryan Tom Han	Nora Ephron	Romantic Co...	You've Got Mail is a 1998 American ...		I can't imagine this movie has age
8	Seven Samurai		✓	Takashi Shimura	Akira Kurosawa	Adventure Drar	Seven Samurai is a 1954 Japanese ...	3: Average 😐	
9	Billy Madison			Adam Sandler	Tamra Davis	Romantic Co...	Billy Madison is a 1995 American ...		
	Fight Club			Brad Pitt	David Fincher	Drama	ht Club is a 1999 film based on the...		

A dropdown menu is open for the 'Genre' column of the 'Fight Club' row, showing the following options: Drama, Comedy, Documentary, Romantic Comedy, and Adventure.

## Серверные СУБД

- Для средних и крупных задач, где важна производительность, надежность, кроссплатформенность.
- Для создания приложений требуется более высокая квалификация.

# Стек технологий

Как написать простое CRUD веб-приложение, работающее с SQLite/MySQL/PostgreSQL ?

# Стек технологий

Как написать простое CRUD веб-приложение, работающее с SQLite/MySQL/PostgreSQL ?

## Frontend

HTML, CSS, JavaScript — обязательно.

- Фреймворки типа Vue.js, React — желательно.

## Backend

- SQL — обязательно
- PHP или PHP-фреймворк (Laravel, Symfony)
- Или Django (Python), Node.js (JavaScript), Ruby on Rails (Ruby), ASP.NET (C#) и т.д.

## Deploy

Развертывание окружения на Linux-сервере

Или настройка среды на облачной платформе (Heroku, Google, AWS, Yandex, ...)

# Классификация СУБД по способу структурирования данных

**1. Реляционные**

**2. Нереляционные (NoSQL)**

# Классификация СУБД по способу структурирования данных

## 1. Реляционные

Самые распространенные и развитые. База данных – совокупность связанных друг с другом двумерных таблиц (строки=записи, столбцы=поля). Каждая запись в таблице имеет одинаковую структуру (набор полей). Стандартный язык SQL для работы с данными.

MySQL, PostgreSQL, Microsoft SQL Server

## 2. Нереляционные (NoSQL)

# Классификация СУБД по способу структурирования данных

## 1. Реляционные

Самые распространенные и развитые. База данных – совокупность связанных друг с другом двумерных таблиц (строки=записи, столбцы=поля). Каждая запись в таблице имеет одинаковую структуру (набор полей). Стандартный язык SQL для работы с данными.

MySQL, PostgreSQL, Microsoft SQL Server

## 2. Нереляционные (NoSQL)

- **Хранилища ключей и значений.** Простейшая модель. Данные можно легко распределять в кластере. Для сложных запросов не подходят. Redis, Riak.
- **Столбцовые.** Данные хранятся не по строкам, а по столбцам. Хорошо подходят для BigData. HBase, ClickHouse.
- **Документарные.** Документ – объект, обладающий произвольным набором атрибутов (полей). Коллекция – набор документов (документы могут иметь разную структуру). База данных – совокупность коллекций. MongoDB.
- **Графовые.** Упор на установление произвольных связей между данными. Neo4j.