Лекция 1.

Классификация приложений и СУБД. Задачи курса и л/р 1.

Практические задачи курса

- Закрепить навыки работы с Git и GitHub
- Поработать с базам данных разных типов
- Разработать приложения разных типов, используя современные технологии и инструменты

  1. Shell-скрипт с локальной реляционной БД (SQLite)
  2. Консольное (PHP) с локальной реляционной БД (SQLite)
  3. Браузерное (JavaScript) с локальной нереляционной БД (IndexedDB)
  4. Веб-приложение (микрофреймворк PHP) с реляционной БД

Лабораторные работы

1. Установить и настроить Git, SQLite, PHP. Написать shell-скрипт (cmd/bash), работающий с SQLite
2. Установить Composer. Написать консольное приложение на PHP, зарегистрировать его как пакет на Packagist
3. Написать консольный вариант игры на PHP
4. Подключить PHP-приложение к СУБД SQLite
5. Перевести PHP-приложение на ORM RedBeanPHP
6. Написать браузерный вариант игры на языке JavaScript
7. Подключить к игре СУБД IndexedDB
8. Написать игру в виде веб-приложения: бэкенд на PHP с СУБД Sqlite

Классификация приложений

1. Десктопные (Windows, MacOS, Linux):
   - CLI
     - Shell-скрипты
     - Универсальные языки
   - GUI
2. Веб-приложения (в браузере):
   - Только клиентская часть (JavaScript)
   - Клиент (JavaScript) + Сервер
3. Мобильные приложения

Классификация приложений

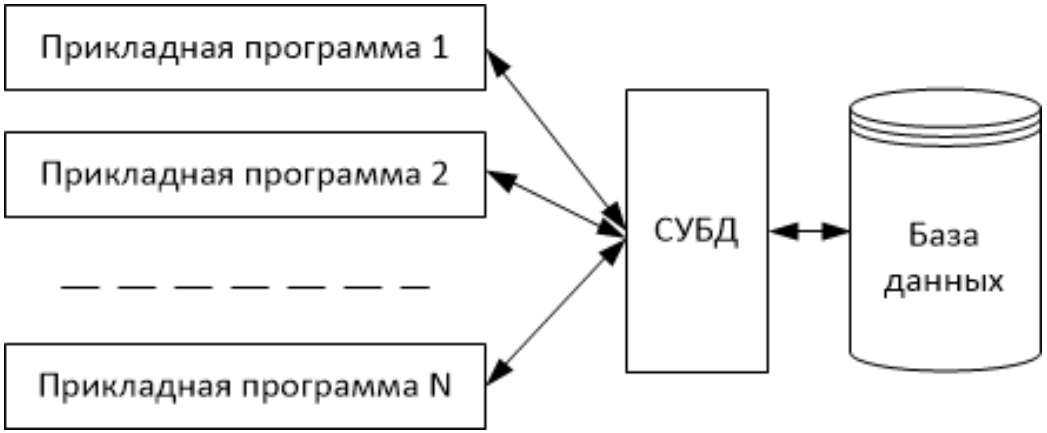
1. Десктопные (Windows, MacOS, Linux):
   - CLI
     - Shell-скрипты
     - Универсальные языки
   - GUI
2. Веб-приложения (в браузере):
   - Только клиентская часть (JavaScript)
   - Клиент (JavaScript) + Сервер
3. Мобильные приложения

Повторяем термины баз данных

База данных - совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причем такое собрание данных, которое поддерживает одну или более областей применения.

Отделение данных от программ

СУБД – программный комплекс поддержки общедоступных данных, предназначенный для создания, ведения и использования базы данных многими пользователями (прикладными программами).



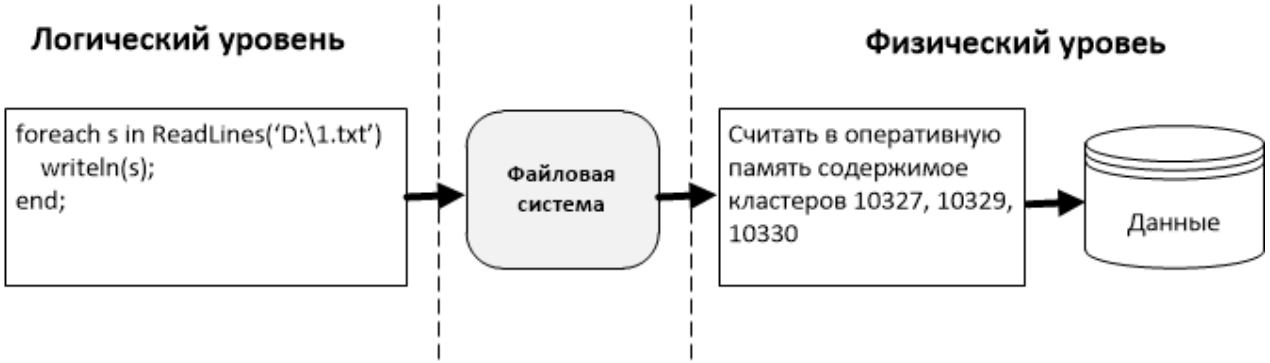
Логическая независимость данных – общая структура данных может быть изменена без изменения прикладных программ.

Физическая независимость данных – изменение способов организации базы данных на внешнем носителе не должно влиять на логическое представление данных и прикладные программы.

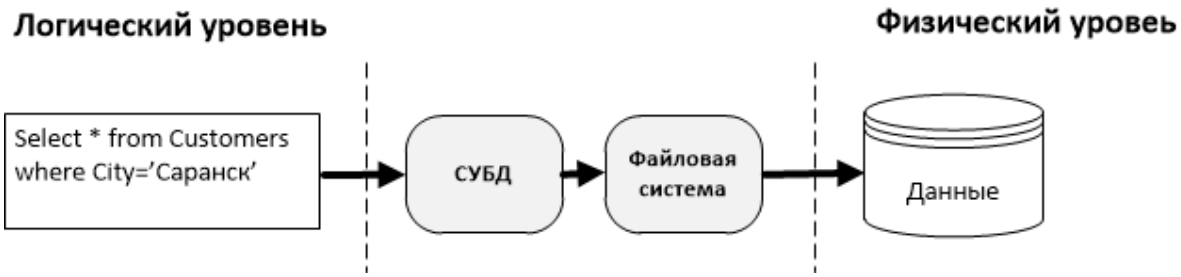
- Файловая система на логическом уровне унифицирует запись и считывание данных.

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

Работа с файлом



Работа с базой данных



Основные функции СУБД

1. Обеспечение логической и физической независимости прикладных программ и данных.

Основные функции СУБД

2. Определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки.

```
CREATE TABLE `vendor` (
  `id` INTEGER PRIMARY KEY,
  `name` VARCHAR(30) NOT NULL,
  `city` VARCHAR(30) NOT NULL,
  `percent` INTEGER NOT NULL CHECK (`percent`>0)
);
```

Основные функции СУБД

3. Обеспечение возможности манипулирования данными (выборка данных, вычисления, разработка интерфейса ввода/вывода, визуализация).
Специальный язык, CLI или GUI.

SQL

```
INSERT INTO `vendor` (`id`, `name`, `city`, `percent`) VALUES
(1001, 'Иванов', 'Саранск', 12),
(1002, 'Петров', 'Москва', 13),
(1003, 'Андреев', 'Кострома', 10),
(1004, 'Сидоров', 'Саранск', 10)

SELECT * FROM vendor WHERE name LIKE "Пе%"
```

NoSQL (MongoDB)

```
db.vendor.insert({name: 'Потапкин', city: 'Саранск', percent: '14'})
db.vendor.insert({name: 'Петров', city: 'Москва', percent: '15'})
db.vendor.insert({name: 'Андреев', city: 'Кострома', percent: '12'})
db.vendor.insert({name: 'Сидоров', city: 'Саранск', percent: '12'})
db.vendor.insert({name: 'Козлов', city: 'Саранск', percent: '26'})

db.vendor.find({city: 'Саранск', percent: {$gt: '14'}}).sort({name: 1})
```

Основные функции СУБД

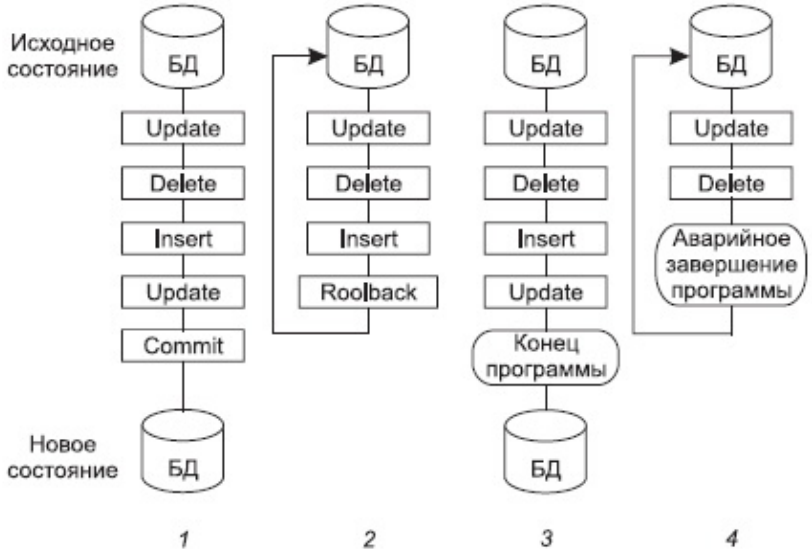
4. Защита логической целостности базы данных.

```
CREATE TABLE `vendor` (
  `id` INTEGER PRIMARY KEY,
  `name` VARCHAR(30) NOT NULL,
  `city` VARCHAR(30) NOT NULL,
  `percent` INTEGER NOT NULL CHECK (`percent`>0)
)

CREATE TABLE `customer` (
  `id` INTEGER PRIMARY KEY,
  `name` VARCHAR(30) NOT NULL,
  `city` VARCHAR(30) NOT NULL,
  `rating` INTEGER NOT NULL CHECK (`rating`>0),
  `vendor_id` INTEGER NULL,
  FOREIGN KEY (`vendor_id`) REFERENCES `vendor` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

Основные функции СУБД

5. Защита физической целостности. Поддержка транзакций и блокировок.



Основные функции СУБД

6. Управление полномочиями пользователей на доступ к базе данных.

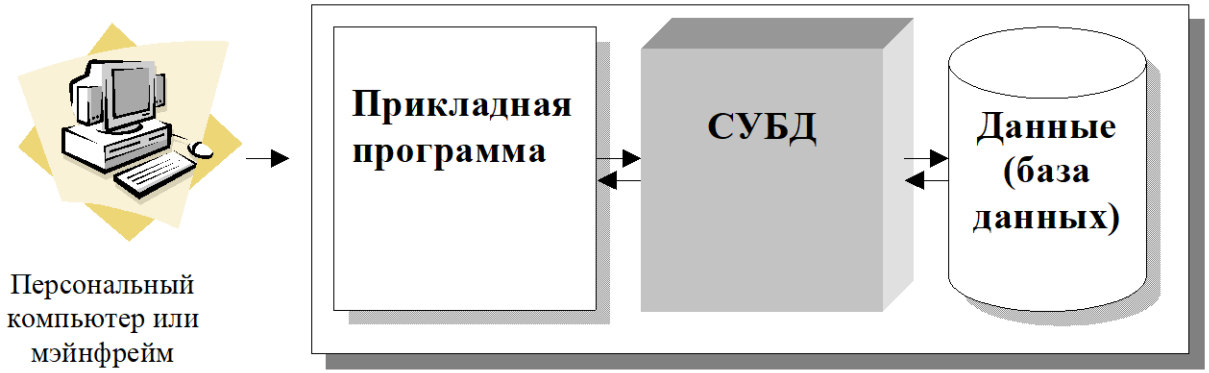
7. Синхронизация работы нескольких пользователей.

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

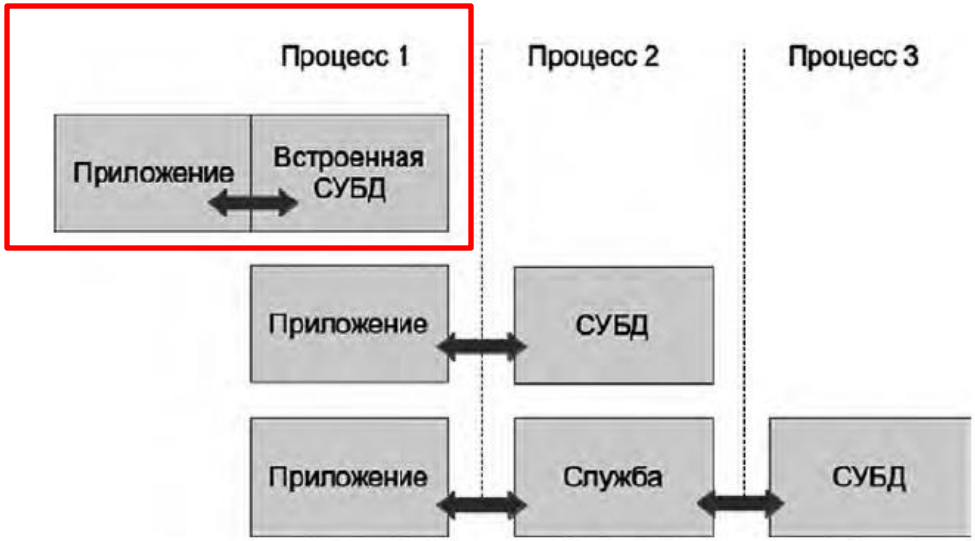
- База данных – набор файлов, находящихся на жестком диске компьютера.
- СУБД и приложение для работы с базой данных установлены на той же машине.
- Простейший случай (встроенная СУБД): БД — один файл, СУБД — библиотека, подгружаемая к прикладной программе.



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

Централизованная архитектура. Встраиваемые СУБД

Microsoft Access,
LibreOffice Base, Firebird,
SQLite



Преимущества:

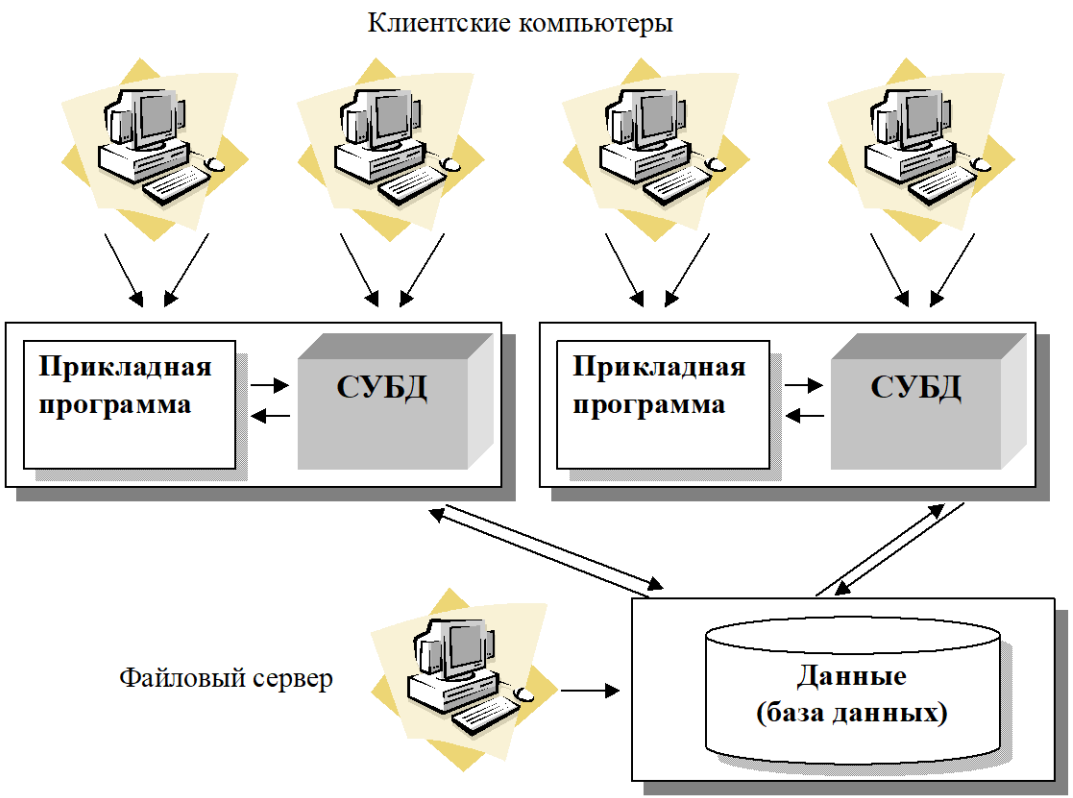
- Простота разработки и развертывания приложений.
- Простота обслуживания локальной БД.
- Высокое быстродействие на простых операциях считывания и модификации одиночных записей.

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.



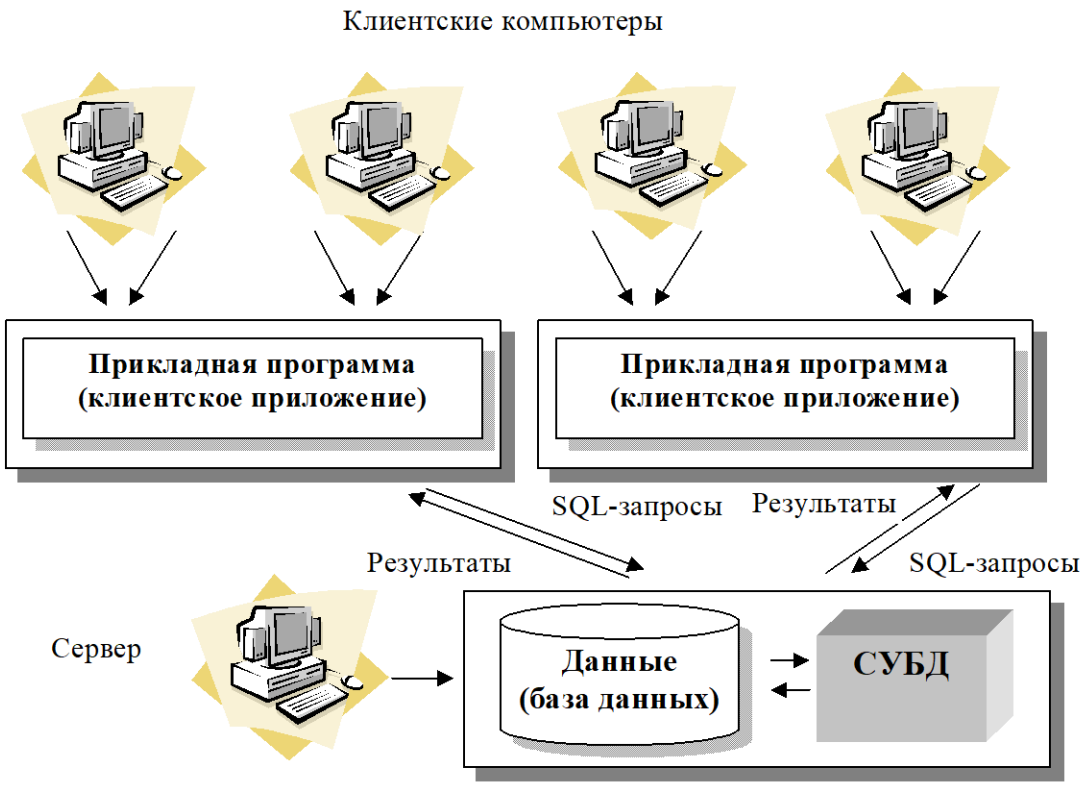
Архитектура «файл-сервер»

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.

Трехуровневая архитектура

Дуэйн Ричард Хипп (D. Richard Hipp), 1961 г.р.

Создал СУБД SQLite, систему контроля версий Fossil, участвовал в разработке языка TCL

«Если мы подумаем о каждом SQL-операторе как о программе, то нужно просто взять эту программу и скомпилировать её в своего рода исполняемый код. Я придумал структуру байт-кода, который фактически запускал запрос, а затем написал компилятор, который переводил SQL в этот байт-код. И вуаля: родилась СУБД SQLite»
https://habr.com/ru/company/macloud/blog/566396/
https://habr.com/ru/company/macloud/blog/566540/

Лабораторные работы

1. Установить и настроить Git, SQLite, PHP. Написать shell-скрипт (cmd/bash), работающий с SQLite
2. Установить Composer. Написать консольное приложение на PHP, зарегистрировать его как пакет на Packagist
3. Написать консольный вариант игры на PHP
4. Подключить PHP-приложение к СУБД SQLite
5. Перевести PHP-приложение на ORM RedBeanPHP
6. Написать браузерный вариант игры на языке JavaScript
7. Подключить к игре СУБД IndexedDB
8. Написать игру в виде веб-приложения: бэкенд на PHP с СУБД Sqlite

Задачи для л/р 1

- Установить SQLite
- Установить PHP (CLI)
- Написать shell-скрипт (cmd/bash), работающий с SQLite
- Написать PHP-скрипт
- Настроить Git и GitHub, сделать Pull Request