1. Функционал Файловых систем: Дайте определение файловой системы с точки зрения программиста приложений и перечислите ключевые задачи, которые ФС берет на себя (например, распределение памяти, отображение имен).

Ответ: Файловая система — это программно-аппаратный уровень абстракции операционной системы, который предоставляет прикладным программам удобный способ хранить, организовывать, находить и манипулировать данными в долговременной памяти, скрывая детали физических носителей.
Для приложения ФС выглядит как иерархия каталогов и файлов с именами, атрибутами и операциями таких как создание, открытие, чтение, запись, удаление.

2. Логическое и физическое представление файлов: Опишите базовое представление файла как набора последовательно нумеруемых блоков. Объясните два основных подхода к логическому представлению файла, используемых в разных операционных системах (последовательность записей, как в Open VMS, и непрерывная последовательность байтов, как в UNIX).

Ответ: Файл как набор последовательно нумеруемых блоков (базовая физическая модель)

На уровне хранения любая файловая система представляет файл как последовательность блоков фиксированного размера, например 4 KB.
Каждому блоку соответствует его номер (номер физического блока) на диске или логическом устройстве.

Базовые свойства физического представления:

- Данные файла занимают несколько блоков; последний блок обычно частично заполнен.
- ФС ведёт таблицу или структуру, указывающую, какие именно блоки принадлежат файлу (inode, FAT, extent list и др.).
- Очерёдность блоков в логическом файле определяется логическим порядком, а не физической смежностью на диске.

3. Индексы и доступ к данным: Какие механизмы используются в некоторых файловых системах для обеспечения выборки записей по ключу (например, индексно-последовательные файлы)? На какой технике основана организация индексов?

Ответ: Файловые системы, поддерживающие выборку записей по ключу (индексно-последовательные файлы), используют индексы, которые хранят пары:

ключ → указатель на блок (или запись).

Основные механизмы организации таких индексов:

1. B-деревья / B+-деревья — самый распространённый и универсальный способ.
2. ISAM (Indexed Sequential Access Method) — статический индексный файл рядом с основным.
3. Хеш-индексы — для быстрого доступа без упорядочивания.

Техника основывается на древовидных структурах поиска и хеш-таблицах, оптимизированных под блочную организацию дисков.

4. Схемы именования файлов: Объясните, как многоуровневая схема именования файлов (через каталоги) обеспечивает логическую классификацию файлов и предотвращает коллизии имен.

Ответы: Многоуровневая схема именования использует дерево каталогов. Каждый каталог требует уникальности имён только внутри себя, поэтому одинаковые имена могут существовать в разных каталогах. Это позволяет логически группировать файлы по разделам или проектам и одновременно предотвращает глобальные конфликты имён, поскольку полный путь однозначно определяет конкретный файл.

5. Сравнение изолированных и централизованных ФС: Сравните подходы к организации файловых систем: изолированные ФС (например, Windows/IBM) и централизованные ФС (например, Multics). В чем заключается компромиссное решение, реализованное в ОС UNIX, и как оно использует механизм монтирования ?

Ответы: Изолированные ФС (Windows, IBM):
Каждый физический носитель имеет собственную корневую файловую систему (например, диски C:, D:). Файловые пространства разделены, и пути не пересекаются. Это упрощает структуру, но делает пространство имён фрагментированным.

Централизованные ФС (Multics):
Вся система использует единое древовидное пространство имен, в которое включаются все устройства. Пользователь видит одну общую иерархию, что удобно, но требует сложной организации и управления.

Компромисс UNIX:
UNIX также предоставляет единое дерево каталогов, но каждую отдельную файловую систему подключает к нему с помощью *монтирования*.
Монтирование позволяет «встраивать» разные устройства и их собственные ФС в общую иерархию, назначая им точки подключения (mount points). Так сочетается удобство единой структуры с гибкостью подключения независимых файловых систем.

6. Авторизация доступа: Объясните, в чем заключается дискреционный подход к авторизации доступа к файлам, традиционно поддерживаемый в ОС UNIX. Какие три действия контролируются для каждого файла в этом подходе?

Ответы: Дискреционный подход в UNIX означает, что владелец файла сам определяет, кому и какие права предоставлять. Доступ регулируется через набор разрешений для владельца, группы и остальных пользователей.

Контролируются три действия:

1. Чтение файла
2. Запись (изменение) файла
3. Выполнение файла как программы (для каталогов — право входа и поиска)

7. Синхронизация многопользовательского доступа: Как файловые системы обеспечивают взаимную синхронизацию при многопользовательском доступе, если один из процессов пытается изменить файл?

Ответы: Файловые системы защищаются от конфликтов при одновременной работе нескольких процессов с помощью блокировок.

Если один процесс начинает изменять файл, он ставит эксклюзивную блокировку, и другие процессы либо ждут, либо получают отказ в доступе.
Благодаря этому несколько процессов не могут менять один и тот же файл одновременно.

8. иповые области применения ФС: Для каких типов данных и приложений файловые системы обычно подходят (где структура данных является "аморфной" и дальнейшая структуризация отдается прикладным программам)?

Ответы: Файловые системы обычно подходят для данных без заранее заданной структуры то есть аморфных данных

Примеры
- текстовые документы изображения видео аудио
- архивы и контейнеры
- программы и исполняемые файлы
- данные где структура формируется и интерпретируется приложением например базы данных JSON XML

Файловая система хранит данные как последовательность байтов а логическую организацию обеспечивает прикладная программа

9. Проблема избыточности данных: На примере информационной системы учета служащих (при использовании одного файла СЛУЖАЩИЕ) объясните, как возникает существенная избыточность данных (дублирование информации) и к каким неудобствам это приводит

Ответы: В информационной системе учета служащих если хранить всю информацию в одном файле СЛУЖАЩИЕ, то для каждого служащего приходится повторять одинаковые данные.

Например у нескольких сотрудников может быть один и тот же отдел, должность или проект. Если эти данные хранятся в каждой записи полностью, они дублируются многократно.

Это приводит к неудобствам
- данные занимают больше памяти
- сложно вносить изменения при изменении общей информации например название отдела нужно исправлять во всех записях
- повышается риск ошибок и неконсистентности данных
- труднее выполнять анализ и формировать отчеты из-за повторяющейся информации

10. Проблема целостности (согласованности) данных: При переходе к модели с двумя файлами (СЛУЖАЩИЕ и ОТДЕЛЫ) возникает необходимость в поддержке целостности данных. Приведите примеры двух правил целостности, связанных со ссылками между файлами, и двух правил, связанных с вычисляемыми значениями.

Ответы: Правила целостности при файлах СЛУЖАЩИЕ и ОТДЕЛЫ

Правила связанные со ссылками между файлами
- каждый служащий должен ссылаться на существующий отдел
- нельзя удалить отдел пока в нем есть сотрудники

Правила связанные с вычисляемыми значениями
- зарплата сотрудника не может быть меньше минимальной для его уровня или должности

общая сумма зарплат по отделу должна совпадать с суммой зарплат всех сотрудников этого отдела

11. Роль метаданных: Почему для обеспечения согласованности данных в нескольких файлах информационная система не может обойтись простой библиотекой функций? Введите понятие метаданных.

Ответы: Метаданные — это данные о данных, которые описывают их структуру, формат, связи и свойства.

Простая библиотека функций для работы с файлами не может обеспечить согласованность данных, потому что она не знает:

- структуру данных в каждом файле,
- связи между файлами,
- правила целостности данных.

Метаданные позволяют централизованно хранить эту информацию, автоматически проверять правильность и согласованность данных, облегчая поддержку и расширение системы.

12. Языки запросов: Какое ключевое преимущество дает использование языка запросов (например, SQL) по сравнению с прямым управлением данными через файловую систему?. Объясните, как СУБД использует информацию о ключевых полях (содержащуюся в метаданных) для выполнения запроса, позволяя пользователю не задумываться о процедуре выполнения.

Ответы: Ключевое преимущество использования языка запросов (например, SQL) заключается в том, что пользователь может обращаться к данным на высоком уровне, не заботясь о том, как именно эти данные хранятся, структурированы или извлекаются из файлов. SQL позволяет формулировать «что получить», а не «как это получить».

Как СУБД использует метаданные для выполнения запроса:

1. В метаданных хранятся сведения о структуре таблиц:
    - имена столбцов, типы данных, ключевые поля (первичные и внешние ключи), индексы.

2. Когда пользователь выполняет запрос, СУБД анализирует его и опирается на эти метаданные, чтобы:
    - определить порядок доступа к данным,
    - использовать индексы для быстрого поиска записей по ключам,
    - проверять целостность ссылок между таблицами (через внешние ключи).
3. Благодаря метаданным СУБД сама решает, как выполнить запрос наиболее эффективно, освобождая пользователя от необходимости писать сложные алгоритмы обхода файлов или проверки связей.

13. **Транзакционное управление и восстановление: Определите понятие транзакции. Объясните, почему транзакционное управление является обязательным условием для обеспечения логической целостности БД, особенно в случае аварийного сбоя (например, выключения питания).**

Ответы: Транзакция — это логически завершённая последовательность операций над базой данных, которая выполняется полностью или не выполняется вовсе. Другими словами, транзакция — это «единица работы» с базой данных.

Почему транзакционное управление важно для логической целостности БД:

1. Целостность данных
    - Транзакция гарантирует, что все изменения данных будут выполнены полностью или не будут выполнены вовсе.
    - Например, при переводе денег с одного счёта на другой: списание с одного и зачисление на другой должны произойти одновременно.
2. Защита от сбоев
    - Если во время выполнения транзакции произойдёт аварийный сбой (выключение питания, ошибка системы), СУБД сможет откатить (undo) все неполные операции, чтобы база не оказалась в неконсистентном состоянии.
3. Механизмы СУБД

- Для этого используются журналы транзакций и механизмы восстановления, которые фиксируют все изменения перед их фактическим применением.

- После сбоя система читает журнал и завершает или отменяет транзакции, обеспечивая согласованность данных.

14. Журнализация (Journaling): Что такое журнал изменений БД и почему он необходим для обеспечения надежности хранения данных?. Опишите стратегию Write Ahead Log (WAL) и ее назначение.

Ответы: 1. Журнал изменений БД

Журнал изменений (Transaction Log, Journaling) — это специальный файл или структура данных, в котором фиксируются все изменения, выполняемые транзакциями в базе данных, до их окончательного применения.

Зачем нужен журнал:

- Обеспечивает надёжность и восстановимость данных.

- При сбое (например, выключении питания) СУБД может использовать журнал, чтобы дополнить или отменить незавершённые операции, восстанавливая базу в консистентное состояние.

2. Стратегия Write Ahead Log (WAL)

Write Ahead Log (WAL) — это принцип работы с журналом, при котором:

1. Все изменения записываются в журнал до того, как они будут записаны в основную базу данных.

2. После успешной записи в журнал транзакция считается «зафиксированной» (committed), и только затем данные могут быть внесены в таблицы.

Назначение WAL:

- Обеспечивает безопасность данных при сбоях: если система падает, СУБД использует журнал для восстановления всех изменений, которые были зафиксированы, но ещё не записаны на диск.

- Позволяет откатить транзакции, которые не были завершены до сбоя.

Вывод: журнал изменений и стратегия WAL — ключевые механизмы обеспечения надежности и консистентности базы данных, позволяющие безопасно выполнять транзакции даже в случае аварий.

15. Автономность СУБД (Архитектура «клиент-сервер»): Объясните, почему СУБД выделилась в независимый системный компонент (сервер). Как архитектура «клиент-сервер» позволяет нескольким информационным системам совместно использовать одну базу данных без потери согласованности?

Ответы:

Почему СУБД стала независимым компонентом (сервером):

- Чтобы централизованно хранить и управлять данными.
- Чтобы гарантировать целостность и согласованность данных.
- Чтобы клиентские программы не зависели от деталей хранения данных.

Как архитектура «клиент-сервер» обеспечивает совместное использование базы:

- Сервер хранит все данные и выполняет запросы от разных клиентов.
- Сервер управляет транзакциями и предотвращает конфликты при одновременном доступе.
- Все клиенты работают с одной базой, данные остаются согласованными.

16. Функции языков БД: Перечислите основные функции, которые выполняют Язык определения данных (DDL) и Язык манипулирования данными (DML).

Ответы:

1. Язык определения данных (DDL, Data Definition Language):

- Создание таблиц, представлений и других объектов базы данных.
- Изменение структуры таблиц (добавление/удаление столбцов).

- Удаление таблиц или других объектов.
- Определение ограничений целостности (например, первичных и внешних ключей).

2. Язык манипулирования данными (DML, Data Manipulation Language):

- Вставка новых данных в таблицы.
- Изменение существующих данных.
- Удаление данных.
- Чтение и выборка данных (запросы SELECT).

17. Типовая организация СУБД: Какие основные логические части можно выделить в современной СУБД (например, ядро, компилятор)?. Какова основная роль ядра СУБД (Data Base Engine)?

Ответы:

Основные логические части современной СУБД:

1. Ядро СУБД (Database Engine) — выполняет операции с данными, управляет хранением и доступом.
2. Компилятор/интерпретатор запросов — анализирует и преобразует SQL-запросы в команды для ядра.
3. Менеджер транзакций — обеспечивает целостность данных, контроль блокировок и откат транзакций при сбоях.
4. Менеджер хранения/дисковый менеджер — управляет физическим хранением данных на диске.
5. Системные каталоги/метаданные — хранят информацию о структуре базы данных, таблицах, ключах и связях.
6. Интерфейсы для клиентов — позволяют приложениям и пользователям отправлять запросы к базе данных.

ЗАДАНИЕ 2.2.9

Accounts

acctNo	type	balance
12345	savings	12000
23456	checking	1000
34567	savings	25

Customers

idNo	firstName	lastName	account
901-222	Robbie	Banks	12345
805-333	Lena	Hand	12345
805-333-B	Lena	Hand	23456

Orders

orderId	customerId	acctNo	orderDate	amount
1	901-222	12345	2025-01-02	150
2	805-333	12345	2025-01-10	75
3	805-333-B	23456	2025-02-01	20

Shippings

shippingId	orderId	customerId	address	shippedAt
1	1	901-222	Boston, Main St. 12	2025-01-05
2	2	805-333	Chicago, Lake Ave. 21	2025-01-12

КОД:

```
PRAGMA foreign_keys = ON;


BEGIN TRANSACTION;


-- Удаляем дочерние таблицы сначала, чтобы не было ошибок при DROP
```

```
DROP TABLE IF EXISTS Shippings;
DROP TABLE IF EXISTS Orders;

-- Затем удаляем Customers и Accounts
DROP TABLE IF EXISTS Customers;
DROP TABLE IF EXISTS Accounts;

-- ----------------------
-- Создаём Accounts
-- ----------------------
CREATE TABLE Accounts (
    acctNo   INTEGER PRIMARY KEY,
    type     TEXT NOT NULL,
    balance  INTEGER NOT NULL
);

-- ----------------------
-- Создаём Customers
-- idNo используется как PRIMARY KEY — все FK будут ссылаться на него
-- ----------------------
CREATE TABLE Customers (
    idNo       TEXT PRIMARY KEY,
    firstName  TEXT NOT NULL,
    lastName   TEXT NOT NULL,
    account    INTEGER NOT NULL,
```

```
    FOREIGN KEY (account) REFERENCES Accounts(acctNo)
);


-- ----------------------
-- Создаём Orders (дочерняя таблица Customers, ссылается на Accounts и Customers)
-- ----------------------
CREATE TABLE Orders (
    orderId    INTEGER PRIMARY KEY AUTOINCREMENT,
    customerId TEXT NOT NULL,
    acctNo     INTEGER NOT NULL,
    orderDate  TEXT,
    amount     INTEGER,
    FOREIGN KEY (customerId) REFERENCES Customers(idNo),
    FOREIGN KEY (acctNo) REFERENCES Accounts(acctNo)
);


-- ----------------------
-- Создаём Shippings (может ссылаться на Orders и Customers)
-- ----------------------
CREATE TABLE Shippings (
    shippingId INTEGER PRIMARY KEY AUTOINCREMENT,
    orderId    INTEGER NOT NULL,
    customerId TEXT NOT NULL,
    address    TEXT NOT NULL,
```

```
    shippedAt  TEXT,
    FOREIGN KEY (orderId) REFERENCES Orders(orderId),
    FOREIGN KEY (customerId) REFERENCES Customers(idNo)
);

-- ----------------------
-- Вставляем данные в Accounts
-- ----------------------
INSERT INTO Accounts (acctNo, type, balance) VALUES
(12345, 'savings', 12000),
(23456, 'checking', 1000),
(34567, 'savings', 25);

-- ----------------------
-- Вставляем данные в Customers
-- (idNo — уникальный, PRIMARY KEY)
-- ----------------------
INSERT INTO Customers (idNo, firstName, lastName, account) VALUES
('901-222', 'Robbie', 'Banks', 12345),
('805-333', 'Lena',   'Hand',  12345),
('805-333-B','Lena',  'Hand',  23456);

-- ----------------------
-- Вставляем данные в Orders (ссылаются на существующих customers и accounts)
```

-- ----------------------

INSERT INTO Orders (customerId, acctNo, orderDate, amount) VALUES

('901-222', 12345, '2025-01-02', 150),

('805-333', 12345, '2025-01-10', 75),

('805-333-B',23456, '2025-02-01', 20);

-- ----------------------

-- Вставляем данные в Shippings (ссылаются на существующие orders/customers)

-- ----------------------

INSERT INTO Shippings (orderId, customerId, address, shippedAt) VALUES

(1, '901-222', 'Boston, Main St. 12', '2025-01-05'),

(2, '805-333', 'Chicago, Lake Ave. 21', '2025-01-12');

COMMIT;

ЗАДАНИЕ 2.3.7

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

Laptop

model	speed	ram	hd	screen	price	od
empty						

Orders

order_id	item	amount	customer_id
1	Keyboard	400	4
2	Mouse	300	4
3	Monitor	12000	3
4	Keyboard	400	1
5	Mousepad	250	2

PC

model	speed	ram	hd	price
empty				

Printer

model	type	price
empty		

Product

maker	model	type
empty		

Shippings

shipping_id	status	customer
1	Pending	2
2	Pending	4
3	Delivered	3
4	Pending	5
5	Delivered	1

ВЕСЬ КОД SQL:

```
DROP TABLE IF EXISTS Printer;

DROP TABLE IF EXISTS Laptop;

DROP TABLE IF EXISTS PC;

DROP TABLE IF EXISTS Product;


----------------------------------

-- a) Schema for Product

----------------------------------

CREATE TABLE Product (

    maker   TEXT,

    model   INTEGER PRIMARY KEY,

    type    TEXT

);
```

```
-----------------------------------
-- b) Schema for PC
-----------------------------------
CREATE TABLE PC (
   model   INTEGER PRIMARY KEY,
   speed   REAL,
   ram     INTEGER,
   hd      INTEGER,
   price   REAL,
   FOREIGN KEY(model) REFERENCES Product(model)
);

-----------------------------------
-- c) Schema for Laptop
-----------------------------------
CREATE TABLE Laptop (
   model   INTEGER PRIMARY KEY,
   speed   REAL,
   ram     INTEGER,
   hd      INTEGER,
   screen  REAL,
   price   REAL,
   FOREIGN KEY(model) REFERENCES Product(model)
);
```

```
-----------------------------------
-- d) Schema for Printer
-----------------------------------
CREATE TABLE Printer (
   model   INTEGER PRIMARY KEY,
   color   TEXT,
   type    TEXT,
   price   REAL,
   FOREIGN KEY(model) REFERENCES Product(model)
);

-----------------------------------
-- e) ALTER Printer: delete attribute COLOR
-----------------------------------
ALTER TABLE Printer RENAME TO Printer_old;

CREATE TABLE Printer (
   model   INTEGER PRIMARY KEY,
   type    TEXT,
   price   REAL,
   FOREIGN KEY(model) REFERENCES Product(model)
);

INSERT INTO Printer(model, type, price)
```

SELECT model, type, price

FROM Printer_old;

DROP TABLE Printer_old;

-- f) ALTER Laptop: add attribute OD with default 'none'

ALTER TABLE Laptop ADD COLUMN od TEXT DEFAULT 'none';

ЗАДАНИЕ 2.4.14

Обозначение таблиц:

```
Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)
```

Какие модели PC имеют скорость >= 3.00?

$$\pi_{model}\left(\sigma_{speed \geq 3.00}(PC)\right)$$

b) Какие производители (maker) делают ноутбуки с жёстким диском **>= 100GB**?

$$\pi_{maker}\left(\sigma_{hd \geq 100}(Product \bowtie_{Product.model=Laptop.model} Laptop)\right)$$

На SQL пример:

```sql
SELECT M.model, M.price
FROM Product P
JOIN (
    SELECT model, price FROM PC
    UNION
    SELECT model, price FROM Laptop
    UNION
    SELECT model, price FROM Printer
) AS M ON P.model = M.model
WHERE P.maker = 'B';
```

d) Найти номера моделей всех цветных лазерных принтеров.

$$\pi_{model}\left(\sigma_{color=true \wedge type='laser'}(Printer)\right)$$

e) Найти производителей, которые продают Laptop, но не продают PC.

$$\left(\pi_{maker}(Product \bowtie Laptop)\right) - \left(\pi_{maker}(Product \bowtie PC)\right)$$

f) Найти размеры жёстких дисков (hd), которые встречаются в двух или более разных PC.

$$\pi_{P1.hd}\left(\sigma_{P1.hd=P2.hd \wedge P1.model \neq P2.model}(\rho_{P1}(PC) \times \rho_{P2}(PC))\right)$$