Самостоятельная работа 2:

Инвариантная часть:

Задание 2.1: Визуализация примера для моделей и подходов к организации данных

Модель данных — это структурное представление элементов данных, их отношений и ограничений в системе управления базами данных (СУБД).

В зависимости от степени абстракции модели данных можно разделить на следующие категории:

1. Иерархическая модель

Предметная область: Система управления файловой структурой компьютера.
Взаимоотношения:

- Данные организованы в виде дерева с одним корневым узлом.
- Каждая папка (родитель) содержит файлы или подпапки (потомки).
- Пример:

Корень (C:/)

├─── Документы

│ ├─── Работа

│ └─── Учеба

└─── Программы

├─── Браузеры

└─── Офис

Характеристика: Жесткая структура, быстрый доступ к потомкам, но сложность связей «многие-ко-многим».

2. Сетевая модель

Предметная область: Система учета поставщиков и товаров.
Взаимоотношения:

- Объекты связаны через указатели (сети).
- Один товар может поставляться разными поставщиками, а поставщик — работать с несколькими товарами.
- Пример:

Поставщик А → Товар 1

Поставщик Б → Товар 1

Поставщик Б → Товар 2

Характеристика: Гибкость связей, но сложность проектирования.

---

3. Реляционная модель

Предметная область: Интернет-магазин.
Взаимоотношения:

- Данные хранятся в таблицах с связями через ключи.
- Пример таблиц:
  - Покупатели (id, имя)
  - Заказы (id, покупатель_id, дата)
  - Товары (id, название, цена)
  - Состав_заказа (заказ_id, товар_id, количество)
    SQL-запрос:

sql

SELECT Покупатели.имя, Товары.название

FROM Покупатели

JOIN Заказы ON Покупатели.id = Заказы.покупатель_id

JOIN Состав_заказа ON Заказы.id = Состав_заказа.заказ_id

JOIN Товары ON Товары.id = Состав_заказа.товар_id;

Характеристика: Простота, стандартизация (SQL), но проблемы с масштабированием.

---

4. Объектно-ориентированная модель

Предметная область: Система проектирования CAD (чертежи).
Взаимоотношения:

- Данные — объекты с методами (например, «сохранить», «отрендерить»).
- Пример классов:

python

```
class Деталь:
    def __init__(self, название, материал):
        self.название = название
        self.материал = материал
    def отрендерить(self):
        pass


class Сборка:
    def __init__(self, детали):
        self.детали = детали  # Список объектов Деталь
```

Характеристика: Естественность для ООП, но низкая скорость сложных запросов.

---

5. Документо-ориентированная модель (NoSQL)

Предметная область: Блог-платформа.
Взаимоотношения:

- Данные — JSON-документы с вложенными структурами.
- Пример документа в MongoDB:

json

```
{
  "_id": "123",
  "title": "Как работает NoSQL",
  "author": "Иван Петров",
  "comments": [
    {"user": "Анна", "text": "Отличная статья!"},
    {"user": "Петр", "text": "Спасибо!"}
  ]
}
```

Характеристика: Гибкость схемы, быстрое чтение/запись, но нет JOIN.

---

6. Графовая модель

Предметная область: Социальная сеть.
Взаимоотношения:

- Узлы (люди) и ребра (дружба, лайки).
- Запрос в Neo4j (Cypher):

cypher

```
MATCH (user:User)-[:FRIENDS_WITH]->(friend)
WHERE user.name = "Анна"
RETURN friend.name;
```

Характеристика: Идеально для сложных связей, но избыточно для простых данных.

---

7. Колоночная модель (NoSQL)

Предметная область: Аналитика продаж.
Взаимоотношения:

- Данные хранятся по столбцам, а не строкам.
- Пример (Cassandra):

sql

```
CREATE TABLE sales (
    product_id UUID,
    date DATE,
    region TEXT,
    amount DECIMAL,
    PRIMARY KEY (product_id, date)
);
```

Характеристика: Быстрые агрегации (SUM, AVG), но медленные обновления.

Вариативная часть:

Задание 2.2: Заполните таблицу "Преимущества и недостатки моделей данных"

№	Модель данных	Преимущества	Недостатки
1	Иерархическая	Простота, скорость	Жесткость связей
2	Сетевая	Гибкость связей	Сложность проектирования
3	Реляционная	Стандартизация (SQL)	Проблемы с масштабированием
4	Объектно-ориентированная	Естественность для ООП	Медленные запросы
5	Документно-ориентированная	Гибкость схемы	Нет транзакций
6	Графовая	Эффективность связей	Избыточность для простых данных
7	Колоночная	Скорость агрегаций	Сложность обновлений