Проектирование Баз Данных

Дисклемер:

Данное решение билетов было сгенерированно ChatGPT на основе лекций по ПБД, в них могут быть ошибки. Если вы нашли ошибку в данном файле, пожалуйста исправьте её и отправьте pull-request.

Теоретические вопросы

1. Проблемы хранения данных в плоских файлах

Подход на основе плоских файлов (flat files) предполагает хранение информации в обычных текстовых файлах, часто с фиксированной или разделённой (например, CSV) структурой.

Основные проблемы:

- Избыточность данных: одни и те же данные приходится дублировать в разных местах.
- Аномалии обновления: изменение одной записи может потребовать правок во множестве файлов.
- Нет обеспечения целостности данных: нельзя установить ограничения на допустимые значения.
- Нет поддержки транзакций: невозможно обеспечить согласованность данных при сбоях.
- Сложности с доступом и поиском: без индексов поиск занимает много времени.
- Отсутствие независимости данных: изменение структуры требует изменения всех программ, которые используют файл.

2. Системы классификации современных БД и реляционные БД

Классификация БД по структуре хранения:

- Реляционные (PostgreSQL, MySQL, Oracle) – данные в виде таблиц, сильные ограничения на целостность.
- Документо-ориентированные (MongoDB, YTsaurus) – иерархические структуры, гибкая схема.

- Колоночные (wide column) – хранение по столбцам, подходит для аналитики (ClickHouse, Cassandra).
- Распределённые – данные хранятся на разных серверах.
- Хранилища данных (data warehouses) – оптимизированы под аналитические запросы, могут использовать колоночное хранение.

Плюсы реляционных БД:

- Стандартизированный язык SQL.
- Высокая степень нормализации данных.
- Поддержка целостности и транзакций (ACID).
- Удобная модель для большинства бизнес-задач.

Минусы:

- Сложность масштабирования.
- Не всегда удобны для хранения вложенных/иерархических данных.
- Не оптимальны для Big Data и потоков событий.

3. Данные и модели данных. Классификация моделей

Данные — это значения, которые описывают объекты, события или понятия.

Модель данных — это способ описания структуры данных и связей между ними.

Классификация моделей данных:

1. Иерархическая — древовидная структура.
2. Сетевая — граф с произвольными связями.
3. Реляционная — таблицы, где строки — кортежи, а столбцы — атрибуты.
4. Объектная — основана на ООП.
5. ER-модель (сущность-связь) — используется для проектирования.

4. Основные архитектурные компоненты (модули) СУБД

СУБД включает в себя следующие модули:

1. Модуль описания схемы — определяет структуру БД.
2. Модуль интерпретации SQL — обрабатывает запросы пользователей.
3. Оптимизатор запросов — выбирает эффективный способ выполнения.

4. Модуль хранения данных — управляет физическим размещением.
5. Буферный менеджер — отвечает за кэширование.
6. Менеджер транзакций — обеспечивает ACID-свойства.
7. Механизм контроля целостности — следит за ограничениями.
8. Средства резервного копирования и восстановления.
9. Управление пользователями и доступом.

5. Реляционная модель и понятие "отношение"

Реляционная модель данных — способ организации данных в виде отношений (таблиц), предложенный Эдгаром Коддом.

Отношение — это таблица, состоящая из строк (кортежей) и столбцов (атрибутов).

Ключевые свойства:

- Уникальность строк (отсутствие дубликатов).
- Отсутствие упорядоченности строк и столбцов.
- Атомарность значений.
- Все значения — из одного домена.
- Наличие первичного ключа.

6. Операции реляционной алгебры (Кодд)

Кодд предложил 8 операций над отношениями:

1. SELECT (σ) — выборка строк.
2. PROJECT (π) — выборка столбцов.
3. UNION (∪) — объединение.
4. SET DIFFERENCE (−) — разность.
5. CARTESIAN PRODUCT (×) — декартово произведение.
6. RENAME (ρ) — переименование.
7. JOIN (⋈) — соединение.
8. DIVISION (÷) — деление (для обработки запросов вида "все такие, у кого есть все").

7. Операторы манипулирования и управления данными (SQL)

Операторы манипулирования данными (DML):

- SELECT — выборка.
- INSERT — вставка.

- UPDATE — изменение.
- DELETE — удаление.

Операторы управления данными (DCL):

- GRANT — выдача прав.
- REVOKE — отзыв прав.

8. Операторы языка определения данных (DDL)

DDL — это часть SQL, отвечающая за структуру базы данных.

Основные операторы:

- CREATE TABLE — создание таблицы.
- ALTER TABLE — изменение структуры.
- DROP TABLE — удаление таблицы.
- CREATE INDEX — создание индексов.
- CREATE VIEW — создание представлений.
-

9. Метод проектирования базы данных «сверху-вниз»

Нисходящий (top-down) подход начинается с построения высокоуровневых моделей данных: определяются главные сущности, связи между ними, затем они уточняются на следующих уровнях, добавляются атрибуты и ограничения.

Особенности:

- Основа — анализ бизнес-процессов и требований пользователей.
- Начинается с концептуального уровня (ER-модель).
- Позволяет задать цельную архитектуру БД с самого начала.

Когда использовать:

- Когда требования чётко определены.
- В системах, где важна семантическая целостность (например, CRM, ERP).

10. Сущности сильного и слабого типов. Примеры

- Сильная сущность (Strong entity) — существует независимо (например, Client, Product).

- Слабая сущность (Weak entity) — не может быть идентифицирована без связи с сильной сущностью.

Пример:

- Client — сильная сущность.
- Preference (предпочтения клиента) — слабая, так как её нельзя однозначно идентифицировать без ClientNo.

Слабые сущности часто используют составной первичный ключ (включающий внешний ключ на родительскую сущность).

11. Шаги концептуального и логического проектирования

Концептуальное проектирование:

1. Сбор требований пользователей.
2. Выявление сущностей и связей (ER-модель).
3. Построение диаграммы сущность-связь.
4. Проверка корректности и согласование с пользователями.

Логическое проектирование:

1. Преобразование ER-модели в логическую (реляционную) схему.
2. Определение отношений и ключей.
3. Удаление избыточности через нормализацию.
4. Создание словаря данных и реляционной схемы.

12. Шаги логического и физического проектирования

Логическое:

- Построение реляционных таблиц на основе ER-модели.
- Нормализация (1NF, 2NF, 3NF).
- Определение первичных и внешних ключей.

Физическое:

- Определение структуры хранения (файлы, индексы).
- Разработка ограничений и триггеров.
- Оптимизация доступа к данным.
- Проектирование пользовательских представлений.
- Реализация в конкретной СУБД.

13. Избыточность данных. Пример

Избыточность — повторное хранение одной и той же информации в разных частях базы.

Пример: В таблице мебели у каждого предмета хранится строка «Массив дерева». При замене на «Натуральное дерево» нужно править все строки. Это неэффективно и ведёт к аномалиям.

14. Аномалия обновления. Пример

Аномалия обновления возникает, когда изменение одного значения требует обновления в нескольких местах. Если это не сделать — возникает противоречие.

Пример: Если материал «Массив дерева» переименовали не во всех строках — в БД останутся несовместимые записи: «Массив дерева», «Натуральное дерево», «Дерево».

15. Кортеж, атрибут, домен. Свойства отношений

- Кортеж — строка в таблице (экземпляр сущности).
- Атрибут — столбец (свойство сущности).
- Домен — допустимые значения атрибута.

Свойства отношений:

1. Нет повторяющихся кортежей.
2. Нет порядка кортежей.
3. Нет порядка атрибутов.
4. Атомарность значений.
5. Атрибуты из одного домена.

16. Процесс нормализации. 1NF, 2NF, 3NF

Нормализация — процесс устранения избыточности и аномалий через разбиение таблиц.

Ненормализованная форма (UNF):

- Содержит повторяющиеся группы, вложенные данные.

1NF (первая нормальная форма):

- Все значения — атомарны.
- Удаляются повторяющиеся группы.

2NF:

- 1NF + отсутствие частичных зависимостей от составного ключа.

3NF:

- 2NF + отсутствие транзитивных зависимостей.

Цель нормализации — корректная структура хранения данных, минимизация избыточности и логических ошибок при обновлениях.

17. Метод проектирования «сверху-вниз». Когда предпочтителен

Суть метода «сверху-вниз»:

- Начинается с обобщённой, концептуальной модели предметной области.
- Затем идёт детализация: определяются атрибуты, связи, ограничения, представления.

Этапы:

1. Выделение ключевых сущностей.
2. Определение их связей и структурных ограничений.
3. Уточнение атрибутов.
4. Преобразование в ER-диаграмму и логическую схему.
5. Преобразование в физическую схему.

Когда предпочтителен:

- При разработке новой БД с нуля.
- Когда известна предметная область, но нет готовых данных.

- При участии аналитиков и бизнес-заказчиков: позволяет сразу отразить бизнес-логику.

Преимущества:

- Гибкость и масштабируемость.
- Ясная логическая структура.
- Удобство для нормализации.

18. Шардирование и партицирование. Способы

Шардирование (sharding) — горизонтальное деление данных на фрагменты (шарды), хранящиеся на разных серверах.

Основные способы шардирования:

1. Диапазонное (Range-Based) — данные делятся по диапазонам значений (например, ID от 1 до 1000 — шард 1).
2. Хешированное (Key-Based) — применяется хеш-функция к ключу (например, shard_id = hash(user_id) % num_shards).
3. Географическое — шарды выделяются по региональному признаку.
4. Смешанное — комбинация подходов.

Партицирование — деление таблицы внутри одного шарда (в пределах одного физического узла).

Преимущества: масштабируемость, распределённая нагрузка.

Недостатки: сложность администрирования, проблемы с кросс-шардовыми транзакциями, необходимость балансировки и resharding'a.

19. Операции реляционной алгебры: выборка, проекция, соединение, деление

1. Выборка (Selection) — извлекает строки, удовлетворяющие предикату:

Пример: R WHERE курс = 1.

2. Проекция (Projection) — извлекает подмножество столбцов:

Пример: π(name, course)(exams) — выведет только имя и курс.

3. Соединение (Join) — объединяет таблицы по совпадающим значениям.

Пример: соединение таблиц EMP и DEPT по DEPT#.

4. Деление (Division) — находит строки, связанные со всеми значениями другого отношения.

Пример: найти студентов, сдавших все экзамены из заданного списка.

20. Язык SQL. Примеры

Создание таблицы:

```
CREATE TABLE Clubs (
  Club VARCHAR(50),
  Country VARCHAR(50),
  Champs INT,
  Cups INT,
  Eurocups INT
);
```

Вставка данных:

```
INSERT INTO Clubs (Club, Country, Champs, Cups, Eurocups)
VALUES (‘Barcelona’, ‘Spain’, 26, 31, 18);
```

Выборка:

```
SELECT Club, Eurocups FROM Clubs WHERE Country = 'Spain';
```

Фильтрация: WHERE, AND, OR, LIKE, BETWEEN

Агрегация: GROUP BY, HAVING, AVG, MAX, COUNT, SUM и др.

21. Подход на основе данных: БД, описание данных, СУБД

- База данных (БД) — организованная совокупность данных.
- Описание данных — метаинформация, включающая схему, типы данных и ограничения.
- СУБД — программный комплекс для управления БД (добавление, изменение, защита, резервное копирование).
- Система баз данных = БД + СУБД + прикладное ПО.

22. Преимущества и недостатки баз данных

Преимущества:

- Компактность хранения.
- Быстрый доступ к данным.
- Многократное использование информации.
- Ускорение бизнес-процессов.
- Поддержка транзакций и безопасности.

Недостатки:

- Сложность в проектировании и сопровождении.
- Высокие требования к вычислительным ресурсам.
- Зависимость от инфраструктуры СУБД.

23. Чем NULL отличается от пустой строки или нуля в SQL?

- NULL — отсутствие значения. Это не то же самое, что пустая строка ('') или ноль (0).
- NULL ≠ '' и NULL ≠ 0
- При использовании NULL логические выражения с ним возвращают UNKNOWN, а не TRUE/FALSE.

Пример:

```
SELECT * FROM users WHERE email IS NULL;
```

24. Архитектура ANSI/SPARC. Уровни

Модель ANSI/SPARC включает 3 уровня представления данных:

1. Физический уровень — как данные хранятся (индексы, блоки и т.д.).
2. Концептуальный уровень (схема) — логическая структура (таблицы, связи).
3. Внешний уровень (подсхемы) — представление данных для отдельных пользователей/приложений.

Преимущества: изоляция уровней, независимость, безопасность.

25. Принцип «Независимость от данных»

Независимость от данных — это возможность модифицировать схему базы данных на одном уровне (физическом или логическом), не затрагивая схемы и приложения на других уровнях.

Существует два вида:

1. Физическая независимость — можно менять способ хранения данных (индексы, блоки и т.д.), не изменяя логическую структуру.
2. Логическая независимость — можно модифицировать логическую схему (добавлять атрибуты), не влияя на внешние представления и прикладное ПО.

Это отражено в правилах Кодда 8 и 9:

- Изменение порядка или добавление атрибута не должно затрагивать приложения.
- Доступ к данным должен осуществляться по имени столбца, а не по его позиции.

26. Функции системы управления базой данных (СУБД)

СУБД выполняет следующие ключевые функции:

- Создание и изменение структуры данных
- Обеспечение целостности данных (первичные и внешние ключи, ограничения)
- Поддержка транзакций (ACID)
- Контроль доступа и безопасность
- Резервное копирование и восстановление
- Оптимизация и выполнение запросов
- Мониторинг и управление производительностью
- Интерфейс для прикладных программ и пользователей.

27. Цели реляционной модели данных. Три условия

Цели:

- Обеспечение структурированной, гибкой и математически обоснованной модели данных.
- Минимизация избыточности и аномалий.
- Гарантия целостности и логической согласованности данных.

Три основных условия (по Кодду):

1. Каждое значение атрибута атомарно
2. Отношение имеет уникальные кортежи
3. Каждое отношение должно иметь первичный ключ

28. Фундаментальные свойства отношений

1. Отсутствие дубликатов кортежей — каждое отношение — множество.
2. Отсутствие порядка строк (кортежей) — строки не упорядочены.
3. Отсутствие порядка столбцов (атрибутов) — порядок не важен.
4. Атомарность значений — каждая ячейка содержит одно значение.
5. Однозначная идентификация — с помощью первичного ключа.

29. Ключи и ссылочная целостность. Примеры

- Первичный ключ — уникально идентифицирует кортеж (например, student_id в таблице «Студенты»).
- Внешний ключ — атрибут, ссылающийся на первичный ключ другой таблицы (например, student_id в таблице «Успеваемость»).

Ссылочная целостность:

- Каждое значение внешнего ключа должно существовать в целевой (родительской) таблице.
- Пример: если есть оценка в «Успеваемости» для студента, студент должен существовать в «Студенты».

Стратегии при удалении/обновлении родительской записи:

- CASCADE, SET NULL, SET DEFAULT, NO ACTION.

30. Реляционное исчисление. Пример

Реляционное исчисление — это декларативный способ выражения запросов на основе логики предикатов.

Пример на кортежной форме:

$$\{ t \mid t \in Students \land t.age > 18 \}$$

Этот запрос возвращает все кортежи t из отношения Students, где возраст больше 18.

31. Схема отношения и экземпляр отношения

- Схема отношения — описание структуры: имя отношения, имена атрибутов, их домены.

Пример: Students(student_id: INT, name: VARCHAR, age: INT)

- Экземпляр отношения — конкретное множество строк (кортежей) в таблице в конкретный момент времени.

Это как описание таблицы (схема) и её содержимое (экземпляр).

32. Фазы проектирования базы данных. Первичный ключ отношений

Фазы проектирования БД:

1. Сбор и анализ требований

   → Определение сущностей, атрибутов, ограничений.

2. Концептуальное проектирование

   → ER-модель: сущности, связи, кратности, типы атрибутов.

3. Логическое проектирование

   → Преобразование ER-модели в логическую (табличную) схему.

   → Определение первичных и внешних ключей.

   → Проверка нормальных форм (нормализация).

4. Физическое проектирование

   → Настройка индексов, стратегий хранения, секционирование, триггеры, процедуры.

5. Имплементация и тестирование

   → Реализация в СУБД, наполнение, оптимизация запросов.

Первичный ключ:

- Атрибут (или группа), уникально идентифицирующий каждый кортеж (строку).
- Не может быть NULL.
- Может состоять из нескольких атрибутов (составной ключ).

Примеры:

- student_id в таблице студентов.

- {student_id, course_id} — в таблице "Оценки".

33. Сбор и анализ требований. Спецификация пользовательских представлений. Цикл работы СУБД

Сбор и анализ требований включает:

- Идентификацию пользовательских ролей (например, клиент, сотрудник, арендодатель).
- Формирование данных, которые необходимо хранить (имя, адрес, телефон и пр.).
- Определение операций над данными: ввод, удаление, обновление, поиск.
- Выявление ограничений и бизнес-правил (например, сотрудник управляет не более чем 100 объектами).

Цикл работы СУБД:

- Пользователь или приложение формирует запрос (SQL).
- СУБД интерпретирует запрос.
- Оптимизатор определяет лучший план выполнения.
- План выполняется.
- Возвращаются результаты.

34. Модель данных «сущность–связь». Основные концепции ER-модели

ER-модель описывает структуру предметной области в виде сущностей, связей и атрибутов:

- Сущности (Entities) — объекты реального мира (например, Клиент, Объект недвижимости).
- Связи (Relationships) — логические связи между сущностями (например, Арендует).
- Атрибуты (Attributes) — свойства сущностей и связей (например, имя клиента, арендная плата).

Используется ER-диаграмма, в которой:

- Сущности — прямоугольники.
- Связи — ромбы.
- Атрибуты — эллипсы.

35. Основные элементы ER-модели

Тип сущности — множество объектов с одинаковыми характеристиками.

Экземпляр сущности — конкретный объект (строка в таблице).

Связь — взаимодействие между сущностями (один-к-одному, один-ко-многим, многие-ко-многим).

Атрибуты:

- Простые и составные.
- Однозначные и многозначные.
- Производные (вычисляемые).

Подтипы (subtypes) — специализация сущностей (например, Клиент → Частное лицо и Компания).

36. Атрибуты связей. Структурные ограничения. Кратность связей

Атрибуты связей отображаются как эллипсы, соединённые штриховой линией с ромбом связи.

Кратность — сколько экземпляров сущности может участвовать в связи (1:1, 1:*, :).

Кардинальность — максимальное количество экземпляров, участвующих в связи.

Степень участия — указывает, все ли экземпляры обязаны участвовать в связи:

Полное участие — обязательно (значение кратности ≥ 1).

Частичное участие — необязательно (значение кратности от 0).

37. Нотация «вороньи лапки» и нотация П. Чена

- Нотация П. Чена (1976):

    Сущности — прямоугольники.

    Связи — ромбы.

    Атрибуты — овалы.

    Специализация/обобщение — кружки с буквами d (разделение) и o (пересечение).

Линии могут быть одиночными или двойными (обязательное/необязательное участие).

- Нотация «вороньи лапки» (Crow’s Foot):

  Используется в UML.

  Визуально показывает кратность:

  - |—| : один
  - |< : много
  - 0..1, 1.., 0.. — степени участия

38. Типы данных СУБД PostgresPRO

Основные типы данных в PostgreSQL (и PostgresPRO):

Числовые:

- INTEGER, SMALLINT, BIGINT
- DECIMAL(p,s), NUMERIC, REAL, DOUBLE PRECISION

Строковые:

- CHAR(n), VARCHAR(n), TEXT

Дата и время:

- DATE, TIME, TIMESTAMP, INTERVAL

Булевы:

- BOOLEAN

Массивы:

- TEXT[], INTEGER[] и др.

UUID, JSON, JSONB, XML, BYTEA — для хранения структурированных или бинарных данных.

PostgresPRO полностью совместим с PostgreSQL, включая расширенные типы и механизмы оптимизации.

39. Метод проектирования «снизу-вверх»

Снизу-вверх (Bottom-up) — подход, при котором сначала анализируются существующие данные (например, таблицы, документы), а затем строится общая модель.

Применим, когда:

- Есть уже сформированные данные или файлы.
- Требуется интеграция различных источников.

Недостатки:

- Сложнее выявить глобальные связи.
- Может привести к избыточности.

Используется в аналитике и миграции данных.

40. Аномалии вставки и модификации. Примеры

Аномалия вставки: невозможно добавить данные без наличия другой информации.

- Пример: нельзя добавить нового клиента без привязки к арендованному объекту.

Аномалия модификации: изменение одного значения требует изменений в нескольких строках.

- Пример: изменение адреса клиента в одном месте и забывка в другом → несогласованность.

Решение: нормализация.

41. Функциональные зависимости. Полная функциональная зависимость

Функциональная зависимость (FD): если атрибут B функционально зависит от A (A → B), то каждому значению A соответствует строго одно значение B.

- Полная функциональная зависимость: если атрибут B зависит от всего набора A, но не от его подмножества.

  Пример:

  A = {staffNo, sName} → branchNo — полная, если нельзя удалить sName и сохранить зависимость.

- Частичная зависимость — когда зависимость сохраняется от части ключа.

Это понятие критично для нормализации, особенно при переходе от 1NF ко 2NF.

42. Ненормализованная форма. Пример. Денормализация

Ненормализованная форма (UNF / ННФ):

- Содержит повторяющиеся группы данных, списки, массивы, вложенные структуры.

  Пример: в одной строке указано несколько номеров телефонов через запятую.

Преобразование:

- Удаление повторяющихся групп.
- Разделение на несколько таблиц.

Денормализация — это преднамеренный отказ от нормализации с целью повышения производительности (например, чтобы избежать JOIN'ов при чтении данных).

43. Нормальная форма Бойса–Кодда (НФБК/BCNF). Пример

BCNF — усиленная 3NF. Требования:

1. Таблица уже в 3NF.
2. Любая детерминанта — это суперключ.

Пример:

- У нас есть таблица Проект + Направление → Куратор.
- Куратор однозначно определяет направление → зависимость Куратор → Направление нарушает правило.

Решение: декомпозировать таблицу на:

- Куратор → Направление
- Проект + Направление → Куратор

Таким образом устраняются зависимости, не идущие от ключа.

44. Оптимизация баз данных. Способы

Основные подходы:

1. Индексация — ускоряет поиск, сортировку, фильтрацию.
2. Оптимизация SQL-запросов — переписывание запросов с учётом индексов, фильтров, условий.
3. Нормализация — устранение избыточности, поддержка целостности.
4. Денормализация — для ускорения операций чтения.
5. Секционирование (partitioning) — деление больших таблиц.
6. Кэширование — использование временных структур для ускоренного доступа.
7. Мониторинг и анализ — отслеживание производительности и выявление узких мест.
8. Репликация и резервное копирование — для отказоустойчивости и масштабирования.

45. Правила преобразования ER-диаграммы в схему БД. Связи между сущностями

Основные правила:

1. 1:1 обязательная с обеих сторон — можно объединить в одно отношение.
2. 1:1 обязательная с одной стороны — внешний ключ добавляется к таблице с обязательным участием.
3. 1:N — внешний ключ от "1" в таблицу "N".
4. M:N (многие ко многим) — создаётся промежуточная таблица с внешними ключами обеих сущностей.

Пример:

Если связь между «Студент» и «Курс» — M:N → создаётся таблица Запись(студент_id, курс_id).

46. Типы архитектуры СУБД: 1-, 2-, 3-уровневая

- 1-уровневая архитектура:
  - Всё на одном уровне (данные, логика, интерфейс).
  - Простой, но не масштабируемый.
- 2-уровневая архитектура (клиент–сервер):
  - Клиент ↔ Сервер.
  - Данные и логика — на сервере, интерфейс — на клиенте.
  - Упрощённое администрирование, но требуется сетевое соединение.
- 3-уровневая архитектура (ANSI/SPARC):

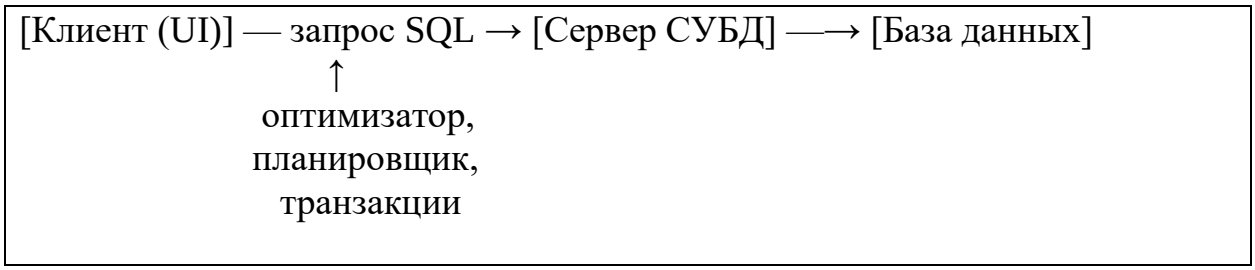
- Уровень хранения (физический).
- Концептуальный уровень (схема).
- Внешний уровень (представления).
- лучшее разделение обязанностей, поддержка независимости данных.

47. Клиент–серверная модель. Схема и описание

Описание:

- Клиент — пользовательское приложение, формирующее запросы.
- Сервер СУБД — принимает запрос, обрабатывает его и возвращает результат.

Схема:



Плюсы:

- Централизованное управление.
- Масштабируемость.
- Разделение функций.

Минусы:

- Зависимость от сети.
- Загрузка сервера при большом числе клиентов.

48. Корпоративная архитектура СУБД. Место БД в архитектуре ИС и приложений

Корпоративная архитектура включает:

- Клиентский уровень (интерфейсы пользователя),
- Уровень приложений (бизнес-логика),
- Уровень данных (СУБД и хранилище).

СУБД в этой структуре выполняет ключевые функции:

- Хранение и обработка данных;
- Интеграция с приложениями;
- Реализация ограничений безопасности;
- Поддержка целостности и транзакций.

СУБД — это фундамент информационной системы, обеспечивающий устойчивость, масштабируемость и надёжность всего программного комплекса.

49. Роль и задачи администратора баз данных (DBA)

Администратор БД (DBA) — специалист, ответственный за:

- Определение и изменение схемы данных;
- Управление подсхемами;
- Назначение прав доступа и контроль безопасности;
- Мониторинг целостности данных;
- Организация резервного копирования и восстановления;
- Регистрация пользователей и настройка прав.

Это ключевая роль в обеспечении стабильной и безопасной работы БД.

50. Принцип независимости представлений данных

Существует два типа независимости:

1. Независимость внешнего представления от концептуального — можно менять схему без изменения представлений (подсхем).
2. Независимость концептуального уровня от физического — можно менять способ хранения данных (например, перестроить индекс) без изменения логической схемы.

Это позволяет:

- Обновлять архитектуру без переписывания приложений;
- Упрощать сопровождение;
- Улучшать производительность без риска для логики обработки.

51. Транзакция в СУБД. Пример

Транзакция — это логически связанная последовательность операций над данными, которая выполняется как единое целое.

- ACID-свойства транзакций:
- Atomicity — атомарность;
- Consistency — согласованность;
- Isolation — изолированность;
- Durability — долговечность.

```
BEGIN;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
COMMIT;
```

Если одна операция провалится — будет выполнен ROLLBACK.

52. Реляционная алгебра: ассоциативность и коммутативность

Ассоциативность:

Операции объединения, пересечения и соединения можно выполнять в любом порядке:

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

Коммутативность:

Некоторые операции возвращают одинаковый результат при перестановке аргументов:

$$R \bowtie S = S \bowtie R$$
$$R \cup S = S \cup R$$

Это позволяет оптимизатору запросов СУБД перестраивать планы выполнения без изменения результата.

53. Проблемы распределённых БД

В распределённых СУБД данные хранятся на разных узлах.

Основные проблемы:

- Согласованность данных: при дублировании копий может возникнуть расхождение.
- Сложность транзакций: трудности с обеспечением ACID в распределённой среде.
- Сетевые задержки и сбои: влияние на доступность и отклик.
- Сложность синхронизации и репликации.
- Безопасность: необходимо защищать данные на разных узлах.

54. Переход от логической к физической модели данных

Переход включает:

1. Определение физических типов данных (INT, VARCHAR и пр.).
2. Проектирование хранения (индексы, секционирование).
3. Оптимизация запросов с учётом плана выполнения.
4. Разработка систем резервного копирования, журналирования.
5. Определение стратегий размещения таблиц, буферизации.

Цель — эффективное хранение и доступ к данным при соблюдении ограничений логической схемы.

55. Свойства отношений и отличие от плоских файлов

Свойства отношений:

1. Атомарность — каждое значение в ячейке — неделимое.
2. Отсутствие дубликатов кортежей.
3. Нет порядка строк (кортежей).
4. Нет порядка столбцов (атрибутов).
5. Все значения — из доменов с определённым типом.

Отличия от плоских файлов:

- В плоских файлах часто нет уникальности строк.
- Плоские файлы не поддерживают типизацию и ограничений (например, внешние ключи).
- Нет встроенного механизма целостности и транзакционности.

56. Почему отношение "многие-ко-многим" реализуется через промежуточную таблицу?

Причина:

Прямая реализация приведёт к:

- Дублированию данных,
- Хранению списков значений в ячейках (нарушение 1НФ),
- Потере возможности задавать уникальные ключи.

Решение — создать отдельную таблицу-связь (junction table), содержащую внешние ключи на обе таблицы, участвующие в связи.

57. Потенциальный, первичный и внешний ключ. Сравнение

- Потенциальный ключ — любой уникальный набор атрибутов, способный идентифицировать кортеж.
- Первичный ключ — один из потенциальных ключей, выбранный как основной.
- Внешний ключ — атрибут, ссылающийся на первичный ключ другой таблицы.

Сравнение:

- Потенциальных может быть несколько, первичный — один.
- Внешний не обязан быть уникальным, но обязан ссылаться на существующий первичный.

58. Операции реляционной алгебры: объединение, пересечение, разность, декартово произведение

- Объединение (UNION): ∪ — возвращает кортежи, присутствующие хотя бы в одном из отношений.
- Пересечение (INTERSECT): ∩ — только общие кортежи.
- Разность (DIFFERENCE): − — кортежи, присутствующие в первом, но не во втором.
- Декартово произведение (×): каждая строка первого отношения объединяется с каждой строкой второго.

59. Приведение отношений к 3НФ и НФБК

- 3НФ — устраняются транзитивные зависимости (A → B → C).

Пример: если студент → группа, а группа → факультет, нужно отделить факультет в отдельную таблицу.

- НФБК (BCNF) — каждая детерминанта должна быть суперключом.

Более строгая форма 3НФ, устраняет и нестандартные зависимости.

Процесс:

1. Анализ функциональных зависимостей.
2. Декомпозиция таблиц на основе выявленных нарушений.
3. Проверка потерь и зависимости.

60. Почему не принято хранить несколько значений в одной ячейке

Нарушается атомарность (1НФ).

Затрудняется:

- Поиск,
- Индексация,
- Связи с другими таблицами,
- Обеспечение целостности.

Правильное решение — вынос многозначных атрибутов в отдельные таблицы с внешними ключами.

61. Приведение к 1НФ

Требование 1НФ:

- Все значения в ячейках должны быть атомарными (одно значение, один тип).

Шаги:

1. Выделение повторяющихся групп.
2. Разделение таблицы.
3. Формирование связи "один-ко-многим".

Пример:

Клиент	Телефоны
Иван	123, 456

↓

Клиент	Телефон
Иван	123
Иван	456

62. Приведение к 2НФ

Требование 2НФ:

- Отношение уже в 1НФ.
- Все неключевые атрибуты зависят от всего первичного ключа, а не от части.

Шаги:

1. Найти частичные зависимости.
2. Разбить таблицу.
3. Сформировать связи.

Пример:

| Проект, Участник, Должность |

Если "Должность" зависит только от "Участник", → нарушение 2НФ.
Решение — вынести в отдельную таблицу.

63. Правила приведения к НФБК

Требование НФБК:

- Для всех зависимостей X → Y, X — суперключ.

Правила:

1. Выявить детерминанты, не являющиеся суперключами.
2. Провести декомпозицию:
    - Сохранять зависимость.
    - Не терять данные (lossless decomposition).
3. Проверить обратимость.

64. Правила Кодда: фундаментальные основы

1. Информационное правило — все данные хранятся в виде значений в таблицах.
2. Гарантированный доступ — каждый элемент данных доступен через комбинацию имени таблицы, атрибута и значения ключа.
3. Поддержка NULL — допустимость отсутствующих значений.
4. Реляционная модель — база = множество отношений.

5. Реляционный язык — доступ и манипуляции только через язык высокого уровня.

65. Правила Кодда: обработка, целостность и независимость

1. Обновляемые представления (VIEW).
2. Высокоуровневые операции (групповые изменения).
3. Физическая независимость.
4. Логическая независимость.
5. Целостность через декларативные ограничения (NOT NULL, UNIQUE и т.д.).
6. Каталог как реляционная база (вся метаинформация хранится в виде таблиц).
7. Никаких обходных путей (обходной доступ — запрещён).

66. Приведение ненормализованной таблицы к НФБК

Пример:

| Студент | Курс | Преподаватель |

Если преподаватель зависит от курса, а курс — не ключ, то нужно разделить:

- Таблица 1: Курс → Преподаватель
- Таблица 2: Студент → Курс

Цель — все зависимости должны быть от ключей и только от них.