Привет! Разберем каждый из твоих вопросов по порядку с подробными пояснениями.

**1. История возникновения БД**

Базы данных (БД) появились в 1960-х годах, когда начали развиваться системы обработки данных для бизнеса и научных учреждений. В то время данные часто хранились в виде файлов в системе управления файлами. Но возникла необходимость в более эффективной организации данных, их поиске, обновлении и сохранении целостности.

Сначала использовались иерархические (например, IBM's IMS) и сетевые модели данных (например, CODASYL), но эти модели имели ограничения по гибкости. В 1970 году Эдгар Код предложил реляционную модель данных, которая стала основой для большинства современных СУБД.

**2. Классификация БД**

Существует несколько типов баз данных:

* **Реляционные БД (RDBMS)** — хранят данные в виде таблиц, где строки представляют записи, а столбцы — атрибуты.
* **Объектно-ориентированные БД** — хранят данные как объекты, как в объектно-ориентированном программировании.
* **Документо-ориентированные БД** — хранят данные в виде документов (например, JSON, XML).
* **Графовые БД** — ориентированы на представление и обработку данных в виде графов с вершинами и рёбрами (например, Neo4j).
* **Ключ-значение БД** — хранят данные в виде пар "ключ-значение" (например, Redis).
* **Колонковые БД** — оптимизированы для чтения больших объемов данных и анализа (например, Apache HBase).

**3. Архитектура БД. Модели БД**

Архитектура базы данных делится на три уровня:

* **Внешний уровень** — представление данных для пользователя (например, через интерфейс).
* **Концептуальный уровень** — описание структуры данных без учета физических аспектов.
* **Внутренний уровень** — как данные физически хранятся на носителе (например, в виде файлов или индексированных таблиц).

Модели БД описывают, как данные организуются:

* **Реляционная модель** — данные организованы в таблицы.
* **Сетевой и иерархический модели** — данные представляют собой сети или иерархии.
* **Объектно-ориентированная модель** — данные представлены как объекты.

**4. Реляционная модель. История возникновения. Основные понятия.**

Реляционная модель была предложена Эдгаром Кодом в 1970 году. Основные концепции:

* **Отношение** — таблица данных.
* **Кортеж** — строка в таблице (запись).
* **Атрибут** — столбец в таблице (свойство).
* **Ключ** — уникальный идентификатор записи.
* **Реляционная алгебра** — набор операций для работы с таблицами.

**5. Отношение, схема отношения, схема БД. Свойства отношений.**

* **Отношение** — таблица, состоящая из строк и столбцов.
* **Схема отношения** — описание структуры отношения, включает имя таблицы и атрибуты.
* **Схема БД** — совокупность всех отношений, хранящихся в базе данных.
* Свойства отношений:
  + Каждое отношение имеет уникальное имя.
  + Каждое отношение состоит из уникальных строк (кортежей).
  + Все атрибуты атомарны (то есть не делятся на более мелкие части).

**6. Типы связи между объектами. Примеры различной связи между объектами (сущностями).**

Связи между объектами могут быть:

* **Один к одному (1:1)** — одна запись в таблице А связана с одной записью в таблице Б.
* **Один ко многим (1:N)** — одна запись в таблице А может быть связана с несколькими записями в таблице Б.
* **Многие ко многим (M:N)** — несколько записей в таблице А могут быть связаны с несколькими записями в таблице Б.

Пример:

* В университете связь "студент — группа" может быть 1:1, если каждый студент записан только в одну группу.
* Связь "автор — книга" может быть 1:N, где один автор написал много книг.

**7. Виды реляционных отношений. Связывание таблиц. Целостность данных.**

* **Виды реляционных отношений** — как уже упоминалось, это "один к одному", "один ко многим" и "многие ко многим".
* **Связывание таблиц** — обычно выполняется через внешний ключ, который ссылается на первичный ключ другой таблицы.
* **Целостность данных** — поддержание согласованности данных через ограничения, такие как:
  + **Целостность сущности** (каждая строка должна быть уникальной).
  + **Целостность ссылок** (каждая ссылка должна указывать на существующую запись).

**8. Основы реляционной алгебры.**

Реляционная алгебра — это набор операций, которые позволяют манипулировать отношениями. Основные операции:

* **Селекция (σ)** — фильтрация строк по заданному условию.
* **Проекция (π)** — выбор определенных столбцов.
* **Объединение (∪)** — объединение двух отношений.
* **Пересечение (∩)** — пересечение двух отношений.
* **Вычитание (−)** — вычитание строк одного отношения из другого.
* **Декартово произведение (×)** — комбинирование каждой строки одного отношения с каждой строкой другого.
* **Соединение (⋈)** — соединение двух отношений по общему атрибуту.

**9. Операции реляционной алгебры. Операции объединения и проекции.**

* **Объединение (∪)** — объединяет два отношения, сохраняя уникальные строки.
* **Проекция (π)** — выбирает определенные столбцы (атрибуты) из отношения.

**10. Операции реляционной алгебры. Операции пересечения и вычитания.**

* **Пересечение (∩)** — возвращает строки, которые присутствуют в обоих отношениях.
* **Вычитание (−)** — возвращает строки, присутствующие в одном отношении, но отсутствующие в другом.

**11. Операции реляционной алгебры. Операции декартова произведения и выборки.**

* **Декартово произведение (×)** — создает новое отношение, состоящее из всех возможных пар строк двух отношений.
* **Выборка (σ)** — извлечение строк, которые удовлетворяют заданному условию.

**12. Операции реляционной алгебры. Операции соединения и деления.**

* **Соединение (⋈)** — комбинирует две таблицы по общему атрибуту.
* **Деление (÷)** — используется для извлечения тех строк из одного отношения, для которых существует пара в другом отношении, которая соответствует всем строкам.

**13. Жизненный цикл БД.**

Жизненный цикл базы данных включает:

1. **Проектирование БД** — определение структуры данных и их взаимосвязей.
2. **Реализация БД** — создание таблиц и связей в СУБД.
3. **Использование БД** — внесение, изменение и извлечение данных.
4. **Поддержка и обновление** — исправление ошибок, обновление структуры данных.

**14. Цикл проектирования БД.**

Проектирование БД включает несколько этапов:

* **Анализ требований**.
* **Концептуальное проектирование** — создание ER-диаграмм.
* **Логическое проектирование** — создание схемы БД с учетом выбранной модели.
* **Физическое проектирование** — оптимизация структуры для хранения данных.

**15. Концептуальное проектирование БД.**

Концептуальное проектирование включает создание абстрактной модели данных, которая описывает сущности и их взаимосвязи без учета технических подробностей. Это обычно выполняется с помощью ER-диаграмм.

**16. Логическое проектирование БД.**

На этом этапе концептуальная модель данных преобразуется в логическую модель, которая соответствует выбранной СУБД. Например, для реляционной модели это будет создание таблиц и определение связей между ними.

**17. Физическое проектирование БД.**

Физическое проектирование включает выбор способа хранения данных, индексацию, распределение данных по дискам и другие аспекты, связанные с производительностью.

**18. Типы сущностей. Иерархия наследования.**

Сущности — это объекты, которые содержат информацию, например, "Клиент" или "Заказ". В иерархии наследования одни сущности могут быть подтипами

других. Например, "Человек" может быть родительской сущностью для "Студент" и "Преподаватель".

**19. Типы связей. Описание связей. Примеры.**

Связи могут быть:

* **Один к одному** — например, каждый студент имеет только одну карточку учета.
* **Один ко многим** — например, один автор может написать несколько книг.
* **Многие ко многим** — например, студенты могут записываться на несколько курсов.

**20. Ключи сущности.**

Ключ сущности — это атрибут (или набор атрибутов), который уникально идентифицирует каждую запись в таблице. Это может быть первичный ключ (PK) или внешний ключ (FK).

**21. Визуальные нотации ER(D)-диаграмм.**

ER-диаграммы (диаграммы "сущность-связь") используют прямоугольники для сущностей, ромбы для связей и овал для атрибутов. Связи между сущностями показываются линиями, а множества связей — с помощью маркировок на этих линиях.

**22. Правила преобразования ER-диаграмм в реляционные таблицы.**

Для преобразования ER-диаграмм в реляционные таблицы выполняются следующие шаги:

1. Каждая сущность становится таблицей.
2. Атрибуты сущности становятся столбцами таблицы.
3. Каждая связь превращается в внешний ключ в одной из таблиц или создается новая таблица для связи.

**23. Функциональные зависимости.**

Функциональная зависимость — это зависимость атрибутов в таблице, где значение одного атрибута однозначно определяет значение другого. Например, если "номер телефона" зависит от "имени", то для одного имени может быть только один номер телефона.

**24. Нормальные формы схем отношений. 1, 2, 3 НФ.**

Нормальные формы — это набор правил для упорядочивания таблиц с целью устранения аномалий:

* **1 НФ** — все атрибуты атомарны.
* **2 НФ** — таблица находится в 1 НФ, и все неключевые атрибуты полностью функционально зависят от первичного ключа.
* **3 НФ** — таблица находится в 2 НФ, и все атрибуты зависят только от ключа.

**25. Нормальные формы схем отношений. 4 НФ, БКНФ**

* **4 НФ** — устранение многозначных зависимостей.
* **БКНФ (BCNF)** — таблица находится в 3 НФ, и каждый детерминант является кандидатом в ключ.

**26. Денормализация.**

Денормализация — это процесс добавления избыточности в базу данных для улучшения производительности. Это может включать дублирование данных или добавление дополнительных индексов.

**27. Язык запросов SQL. Основные понятия.**

SQL (Structured Query Language) — это язык для работы с реляционными базами данных. Основные команды:

* **SELECT** — выборка данных.
* **INSERT** — добавление данных.
* **UPDATE** — обновление данных.
* **DELETE** — удаление данных.

**28. Язык запросов SQL. Команды выборки.**

Команда **SELECT** используется для извлечения данных из таблиц. В SQL можно указать столбцы, условия (WHERE), сортировку (ORDER BY) и ограничение количества строк (LIMIT).

**29. Язык запросов SQL. Внутреннее соединение.**

**INNER JOIN** соединяет строки из двух таблиц, только если они соответствуют определенному условию. Например, соединить таблицы "Сотрудники" и "Отделы" по полю "отдел\_id".

**30. Язык запросов SQL. Внешнее соединение**

**LEFT JOIN**, **RIGHT JOIN** и **FULL JOIN** — это виды внешних соединений, которые включают все строки из одной из таблиц, даже если нет соответствующих строк в другой таблице.

**31. Язык запросов SQL. Перекрестные запросы.**

**CROSS JOIN** возвращает декартово произведение двух таблиц, то есть все возможные комбинации строк из обеих таблиц.

**32. Язык запросов SQL. Подчиненные запросы.**

Подчиненный запрос (или подзапрос) — это запрос, вложенный в другой запрос. Он может быть использован в WHERE, SELECT или FROM для выполнения более сложных операций.

Надеюсь, это поможет тебе в изучении! Если нужно больше подробностей по любому из вопросов, не стесняйся спрашивать!