1. Определение БД

База данных (БД) — это совокупность специальным образом организованных данных, которые:

- Подлежат долговременному хранению в памяти ЭВМ;
- Содержат информацию о небольшом количестве классов объектов, но количество экземпляров объектов в классе может быть огромным (все классы относятся к одной прикладной области);
- Используются в одном или нескольких приложениях одной прикладной области.

Ключевые свойства:

- Жизненный цикл данных превышает жизненный цикл ПО.
- Данные интегрированы и используются совместно.

---

2. Определение схемы БД

Схема БД — это совокупность схем отношений (таблиц) с установленными связями и ограничениями целостности. Она включает:

1. **Логическую структуру:** заголовки таблиц, атрибуты, типы данных.
2. **Связи между таблицами** (например, по первичным и внешним ключам).

Пример: Схема БД о сотрудниках, оборудовании и рабочих местах включает таблицы "Сотрудники", "Оборудование" и связи между ними.

---

3. Ограничения целостности на данные

Ограничения целостности — это правила, обеспечивающие корректность данных:

- **Ссылочная целостность:** Запрет удаления записи, на которую есть ссылки.
- **Целостность сущностей:** Первичный ключ не может быть пустым.
- **Пользовательские ограничения:** Например, возраст сотрудника > 18.

Пример: Если сотрудник уволен (удалён из таблицы "Сотрудники"), все связанные записи (например, в "Рабочие места") должны быть обработаны (удалены или изменены).

---

4. Неизбыточность и непротиворечивость данных

- **Неизбыточность:** Отсутствие дублирования данных. Например, сведения о сотрудниках не должны храниться в отделе кадров и бухгалтерии в разных файлах.
- **Непротиворечивость:** Данные должны быть согласованы. Например, если сотрудник уволен, он не должен получать зарплату.
- **Управляемая избыточность:** Допускается, если СУБД контролирует её (например, индексы или репликация данных).

---

5. Защита от программных и аппаратных сбоев

Типы сбоев:

1. **Логические:**
    - Ошибка 1-го рода: попытка добавить дубликат (отклоняется СУБД).
    - Ошибка 2-го рода: удаление записи со ссылками (отклоняется).
2. **Физические:** Отключение питания, повреждение файлов.

Методы защиты:

- Журнализация изменений.
- Архивация данных.
- Локальность модификаций (изменения затрагивают минимальную часть данных).

---

6. Принцип независимости данных. Технологическая основа его реализации

Принцип: Прикладные программы не зависят от способа хранения данных и аппаратуры.

Реализация через трехуровневое описание:

1. **Физический уровень:** Способ хранения (файлы, индексы).
2. **Логический уровень:** Схема БД (таблицы, связи).
3. **Внешний уровень:** Представление данных для приложений.

Пример: Изменение структуры хранения (например, переход на SSD) не требует переписывания приложений.

---

7. Защита от несанкционированного доступа

Методы:

- Пароли для доступа к внешним схемам.
- Защита файловой системы ОС.
- Шифрование данных на физическом уровне.

Пример: Разные пользователи имеют разные права (бухгалтерия — доступ к зарплатам, кадры — к личным данным).

---

8. Базисные функции СУБД

- Управление обменом данными между приложениями и БД.
- Преобразование данных согласно внешним схемам.
- Защита данных (целостность, секретность).
- Обеспечение многопользовательского доступа.

Пример: СУБД обрабатывает запросы, проверяет права доступа, оптимизирует выполнение.

---

9. Последовательность обработки запросов к БД

1. Приложение отправляет запрос в СУБД (с указанием схемы и пароля).
2. СУБД проверяет права доступа.
3. Оптимизация запроса (логическая и физическая).
4. Поиск данных через методы доступа (индексы, файлы).
5. Чтение блоков данных в буферы.
6. Преобразование данных для приложения.
7. Передача результата приложению.

Примечание: Для записи последовательность аналогична, но данные передаются в обратном направлении.

---

10. Способы организации ЯОД и ЯМД

Язык описания данных (ЯОД) и **язык манипулирования данными (ЯМД)** реализуются:

1. Как расширение классических языков:
   - Вызов процедур (например, VISTA).
   - Добавление синтаксических конструкций (ADABAS).
2. Встроенные языки СУБД: Например, dBase, FoxPro.
3. Независимые стандартизированные языки: Например, SQL.

Пример: SQL объединяет ЯОД (`CREATE TABLE`) и ЯМД (`SELECT`, `INSERT`).

---

11. Элементы данных. Определение и свойства

Элемент данных (атрибут) — минимальная именованная единица информации с типом и однозначной семантикой.

Свойства:

- Атомарность (неделимость).
- Неизменяемость (например, "Дата рождения", а не "Возраст").

Ошибки:

- Неоднозначность (например, "Цех" вместо "Номер цеха").
- Вычисляемые значения (например, "Средняя зарплата").

---

12. Классификация типов связей на схеме БД

Связи отражают соотношение между записями:

- **1:1 (один к одному):** Номер студента → номер читательского билета.
- **M:1 (многие к одному):** Табельный номер → должность.
- **1:M (один ко многим):** Должность → табельные номера сотрудников.
- **M:M (многие ко многим):** Должность → разряды ЕТС.

Обозначение: Одинарная стрелка для 1:1 и M:1, сдвоенная — для 1:M и M:M.

---

13. Избыточные связи на схеме БД

Избыточная связь — дублирующая или не несущая новой информации.

Пример: Если связь между A и B уже определена через промежуточную таблицу, прямая связь A→B избыточна.

Устранение: Проверка замыкания функциональных зависимостей.

---

14. Правило склейки записей

Если между элементами данных A и B связь 1:1 или M:1, то B присоединяется к A, образуя логическую запись. A становится ключевым атрибутом.

Пример: Связь "Сотрудник (табельный номер) → Должность" → запись "Сотрудник" дополняется атрибутом "Должность".

---

15. Зависимость данных от структуры

Данные зависят от структуры, если для их получения необходимо использовать связи.

Пример: В иерархической модели для доступа к данным потомка требуется пройти через предка. В реляционной модели зависимость меньше благодаря независимым таблицам.

Следствие: Изменение структуры (например, добавление связи) может потребовать изменения приложений.

---

16. Преобразование сложных сетевых моделей к простым сетевым

Сложная сетевая схема содержит связь типа M:M. Для преобразования:

1. Создается новое отношение с ключом, объединяющим ключи исходных отношений (например, A+B).
2. Связь M:M заменяется на две связи M:1 от нового отношения к исходным.

Пример:

- Исходно: "Студент" (M) ↔ "Курс" (M).
- После преобразования: "Студент_Курс" (ключ: `ID_студента` + `ID_курса`) с связями M:1 к "Студент" и "Курс".

---

17. Общие данные – определение и правило преобразования

Общие данные — атрибуты, которые могут быть присоединены к нескольким типам записей по правилу склейки.

Правило преобразования: Создается отдельное отношение, содержащее:

- Ключевые атрибуты всех связанных записей.

- Сам общий атрибут.

Пример: Атрибут "Адрес" для "Сотрудник" и "Клиент" → таблица "Адреса" с ключами `ID_сотрудника` и `ID_клиента`.

---

18. Данные пересечения – определение и правило преобразования

Данные пересечения — атрибуты, которые не могут быть присоединены ни к одной записи (к ним ведут только связи M:M), но идентифицируются комбинацией ключей других записей.

Правило преобразования: Создается отношение с:

- Ключами из связанных таблиц.
- Атрибутом пересечения.

Пример: "Оценка" (для связи "Студент" ↔ "Дисциплина") → таблица "Оценки" (`ID_студента`, `ID_дисциплины`, `Оценка`).

---

19. Изолированные данные – определение и правило преобразования

Изолированные данные — атрибуты, не связанные с другими (нет ключей для идентификации).

Правило преобразования:

1. Вводится искусственный ключ (например, `ID`).
2. Создается отдельное отношение с этим ключом и изолированным атрибутом.

Пример: "Комментарий" без привязки → таблица "Комментарии" (`ID_комментария`, `Текст`).

---

20. Определение реляционной модели данных - 1НФ

Отношение находится в **1НФ**, если:

- Нет дублирующих кортежей.
- Все атрибуты атомарны (неделимы).
- Столбцы однородны (одного типа).
- У каждого столбца уникальное имя.
- Порядок строк не имеет значения.

Пример: Таблица "Сотрудники" с колонками `ID`, `ФИО`, `Должность` (каждое поле содержит одно значение).

---

21. Преобразование древовидной и сетевой схемы БД к реляционному виду

- **Древовидная модель:** Каждый узел (кроме корня) имеет одного предка → таблицы с внешними ключами на родителя.
  Пример: "Отдел" (`ID`, `Название`) ↔ "Сотрудник" (`ID`, `ID_отдела`).

- **Сетевая модель:** Связи M:M преобразуются в отдельные таблицы (см. вопрос 16).

---

22. Бинарные базисные операции реляционной алгебры (РА) и их эквиваленты в SQL

Операция (РА)	Обозначение	SQL-эквивалент
Объединение	$R = R_1 \cup R_2$	`R1 UNION R2`
Разность	$R = R_1 \setminus R_2$	`DELETE * FROM R1 WHERE ID IN (SELECT ID FROM R2`
Декартово произведение	$R = R_1 \times R_2$	`SELECT * FROM R1, R2`

---

23. Унарные базисные операции РА и их эквиваленты в SQL

Операция (РА)	Обозначение	SQL-эквивалент
Селекция	$R = \sigma_{условие}(R_1)$	`SELECT * FROM R1 WHERE условие`
Проекция	$R = \pi_{атрибуты}(R_1)$	`SELECT DISTINCTROW атрибуты FROM R1`

---

24. Дополнительный набор операций РА и их выражение через базисный набор

- **Пересечение (∩):**
  $R_1 \cap R_2 = R_1 \setminus (R_1 \setminus R_2)$.
- **Соединение (⋈):**
  $R_1 \bowtie_{условие} R_2 = \sigma_{условие}(R_1 \times R_2)$.
- **Естественное соединение:** Частный случай соединения по одноименным атрибутам.

---

25. Свойство операции селекции

1. **Коммутативность:**
   $\sigma_{F1}(\sigma_{F2}(R)) = \sigma_{F2}(\sigma_{F1}(R))$.
2. **Каскадность:**
   $\sigma_{F1 \wedge F2}(R) = \sigma_{F1}(\sigma_{F2}(R))$.

---

26. Свойство операции проекции

- Уменьшение мощности: Удаление дубликатов.
- Некоммутативность с селекцией: Порядок операций влияет на результат.

---

27. Аномалии при проектировании структуры данных

Тип аномалии	Пример	Решение

Тип аномалии	Пример	Решение
Вставки	Невозможность добавить сотрудника без отдела	Нормализация (2НФ, 3НФ)
Удаления	Потеря сотрудников при удалении отдела	
Обновления	Несогласованность дублируемых данных	

---

28. Определение функциональной зависимости (ФЗ)

ФЗ X → Y означает, что для любых двух кортежей в отношении, если совпадают значения X, то совпадают и значения Y.
Пример: В "Сотрудники" `ID → ФИО` (ID уникально определяет ФИО).

---

29. Аксиомы функциональных зависимостей

1. **Рефлексивность:** Если Y ⊆ X, то X → Y.
2. **Пополнение:** Если X → Y, то XZ → YZ.
3. **Транзитивность:** Если X → Y и Y → Z, то X → Z.

---

30. Правила для ФЗ

- **Разложение:** Если X → YZ, то X → Y и X → Z.
- **Объединение:** Если X → Y и X → Z, то X → YZ.

---

31. Замыкание множества ФЗ. Первичный ключ

- **Замыкание (F^+):** Все ФЗ, выводимые из F по аксиомам.
- **Первичный ключ:** Минимальный набор атрибутов X, такой что X → U (все атрибуты отношения).

---

32. Вторая нормальная форма (2НФ)

Отношение в **2НФ**, если:

1. Находится в 1НФ.
2. Все неключевые атрибуты функционально полно зависят от **всего** первичного ключа (нет частичных зависимостей).

Пример:
В таблице "Заказы" (`ID_заказа`, `ID_товара`, `Количество`, `Название_товара`) → нарушение 2НФ, если "Название_товара" зависит только от `ID_товара`.

---

33. Третья нормальная форма (3НФ)

Отношение в **3НФ**, если:

1. Находится в 2НФ.
2. Нет транзитивных зависимостей (неключевой атрибут не зависит от другого неключевого атрибута).

Пример:
В "Сотрудники" (`ID`, `Отдел`, `Менеджер_отдела`) → нарушение, если "Менеджер_отдела" зависит от "Отдела".

---

34. Замыкание множества атрибутов

Замыкание X^+ — множество атрибутов, функционально зависящих от X.

Алгоритм:

1. Инициализация: $X^+ = X$.
2. Добавлять атрибуты Y, для которых найдется $Z \to Y$ и $Z \subseteq X^+$.
3. Повторять, пока X^+ не перестанет расти.

---

35. Алгоритм построения замыкания множества атрибутов

Пример:
Для $F = \{A \to B, B \to C\}$, $A^+ = \{A, B, C\}$.

---

36. Определение и проверка эквивалентности двух множеств ФЗ

Множества F и G **эквивалентны**, если $F^+ = G^+$.

Проверка:

1. Для каждой ФЗ в F проверить, что она выводится из G.
2. И наоборот.

---

37. Определение и построение минимального покрытия множества ФЗ

Минимальное покрытие:

- Левые части ФЗ несокращаемы (нельзя удалить атрибут).
- Нет избыточных ФЗ.

Алгоритм:

1. Упростить правые части (разложить $X \to YZ$ на $X \to Y$ и $X \to Z$).
2. Удалить избыточные атрибуты слева.
3. Удалить избыточные ФЗ.

---

38. Определение декомпозиции отношений в БД

Декомпозиция — замена одного отношения R на несколько $\{R_1, R_2, ...\}$, таких что $R = R_1 \bowtie R_2 \bowtie ...$
Цель: Устранение аномалий через нормализацию.

---

39. Свойство соединения без потерь информации: определение

Декомпозиция обладает **свойством соединения без потерь**, если исходное отношение можно точно восстановить естественным соединением.

Условие:
Для декомпозиции $\{R_1, R_2\}$, $R_1 \cap R_2 \to R_1$ или $R_1 \cap R_2 \to R_2$.

---

40. Свойство соединения без потерь информации: алгоритм

1. Построить таблицу с атрибутами R и строками для $R_1, R_2, ...$
2. Заполнить ячейки символами a (если атрибут входит в отношение) и b (иначе).
3. Применять ФЗ для замены b на a.
4. Если в строке все a — свойство выполнено.

---

41. Сохранение и реализация зависимостей при декомпозиции

Зависимости **сохраняются**, если $F^+ = (F_1 \cup F_2 \cup ...)^+$.
Проблема: Некоторые ФЗ могут требовать соединения таблиц для проверки.

---

42. Многозначные зависимости, признаки их наличия

Многозначная зависимость X →→ Y означает, что для каждого значения X существует множество значений Y, независимых от $Z = R \setminus (X \cup Y)$.
Признак: Наличие избыточности в данных (например, дублирование при соединении).

---

43. Аксиомы многозначных зависимостей

1. **Дополнение:** Если X →→ Y, то X →→ $(R \setminus Y)$.
2. **Транзитивность:** Если X →→ Y и Y →→ Z, то X →→ $(Z \setminus Y)$.

---

44. Синтез схемы БД. Свойства

На основе множества ФЗ строится декомпозиция в **3НФ**.
Свойства:

- Сохранение зависимостей.
- Соединение без потерь.

---

45. Проблемы обобщенного ключа

1. **Неинтерпретируемость:** Невозможность дать осмысленное имя отношению.
2. **Нетехнологичность:** Отсутствие службы для сопровождения данных.
   Решение: Декомпозиция или введение новых атрибутов.

---

46. Факторы, влияющие на выбор физической организации БД

- Скорость поиска/модификации данных.
- Объем БД.
- Ограничения целостности.
- Многопользовательский доступ.

---

47. Классификация методов доступа (МД)

Метод доступа	Применение
Последовательный	Для запросов с просмотром многих записей
Индексный	Для поиска уникальных записей
Хеширование	Для быстрого доступа по ключу

---

48. Последовательный МД, оценка количества прочитанных записей

- Чтение всех записей: N (количество записей в файле).
- Чтение части записей: X% × N.

---

49. Связные списки

Структура: Записи связаны указателями.
Применение: Организация областей переполнения.

---

50. Индексно-последовательный МД, оценка количества прочитанных блоков

- Поиск по индексу: $\log_{\beta}(N)$ (где b — степень ветвления).
- Чтение блока данных: 1.

---

51. Отведенное свободное пространство и область переполнения

- **Свободное пространство:** Резерв в блоках для дополнения записей.
- **Область переполнения:** Хранение записей, не поместившихся в основные блоки.

---

52. Методы поиска в полном индексе

Метод	Описание
Бинарный поиск	Для упорядоченных индексов
Поиск в B-дереве	Для динамических индексов

---

53. Методы хеширования, метод квадратов

Алгоритм:

1. Ключ возводится в квадрат.
2. Выбираются средние разряды результата.
   Пример: Ключ 123 → 123^2 = 15129 → средние разряды "512".

---

54. Методы хеширования, метод деления

Алгоритм:
`Адрес = Ключ mod P`, где P — простое число.
Пример: Ключ 123, P = 101 → 123 mod 101 = 22.

---

55. Методы хеширования, метод замены системы счисления

Алгоритм: Перевод ключа в другую систему счисления и выбор части разрядов.

---

56. Обработка переполнений в методах хеширования

Метод	Описание
Открытая адресация	Поиск следующей свободной ячейки
Метод цепочек	Хранение синонимов в связанном списке

---

57. Определение и свойства B-дерева

Свойства:

- Сбалансированность.
- Каждый узел (кроме корня) содержит от t до 2t ключей.
  Применение: Индексы в СУБД.

---

58. Процедура дополнения записи в B-дерево

1. Поиск листового узла.
2. Вставка ключа с разбиением узла при переполнении.

---

59. Процедура удаления записи в B-дереве

1. Поиск ключа.
2. Удаление с возможным слиянием узлов.

---

60. Мультисписок: определение и поиск записей

Структура: Связный список с несколькими цепочками.
Поиск: Переход по указателям нужной цепочки.

---

61. Инвертированный файл: определение и поиск записей

Структура: Индекс, где для каждого значения атрибута хранится список записей.
Поиск: Чтение списка по значению атрибута.

---

62. Индекс соединения: определение и поиск записей

Структура: Индекс для ускорения соединения таблиц.
Поиск: Совместная обработка индексов связанных таблиц.