МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ», ВГТУ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

КАФЕДРА КОМПЬЮТЕРНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ ПРОЕКТИРОВАНИЯ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

По дисциплине: Системы хранения и обработки данных

Тема: Разработка логической структуры базы данных

Выполнил работу студент группы мИИВТ-231 Котельников В.В.
(подпись) Фамилия, инициалы

Принял Короленко В.В
(подпись) Фамилия, инициалы

Защищена_____ Оценка_____

Воронеж 2023

Цель лабораторной работы: изучить основы логического проектирования базы данных, освоить процесс разработки логической структуры базы данных и построения диаграммы «сущность - связь».

Основные задачи:

- определение сущности для проекта в соответствие с индивидуальным заданием и их атрибуты;
- выделение ключевых атрибутов;
- определение связей между сущностями и типов связей;
- построение диаграммы сущность-связь для отображения логической структуры базы данных.

1. Определение сущности для проекта в соответствии с индивидуальным заданием и их атрибуты.

Индивидуальное задание.

Предлагаемый набор базовых таблиц:

1. Врачи
2. Пациенты
3. Прием пациентов

Минимальный набор полей базовых таблиц:

1. ФИО врача
2. Специальность врача
3. Стоимость приема
4. Процент отчисления на зарплату
5. Фамилия пациента
6. Имя пациента
7. Отчество пациента
8. Дата рождения пациента
9. Адрес пациента

10.Дата приема

Схема сущности базы данных представлена на рисунке 1

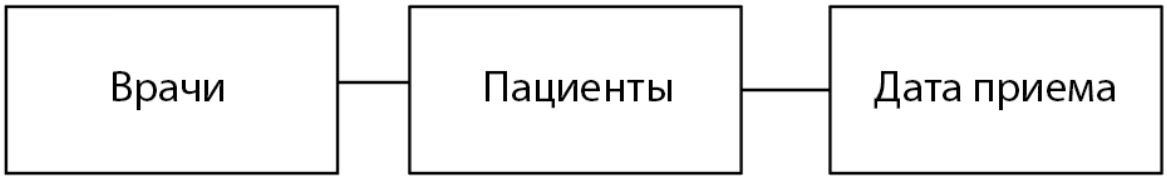

Рисунок 1 – сущности базы данных

2. Выделение ключевых атрибутов.

В таблице «врачи» ключевым атрибутом будет «ФИО врача», а дополнительным ключом «Специальность». В таблице «пациенты» ключевым атрибутом будет ФИО пациента, а дополнительным ключом будет «Специальность врача». В таблице «дата приема» ключевым атрибутом будет «дата», а дополнительными связями будет ФИО врача и ФИО пациента.

3. **Определение связей между сущностями и типов этих связей.**

Определенные связи между таблицами показаны на рисунке 2.

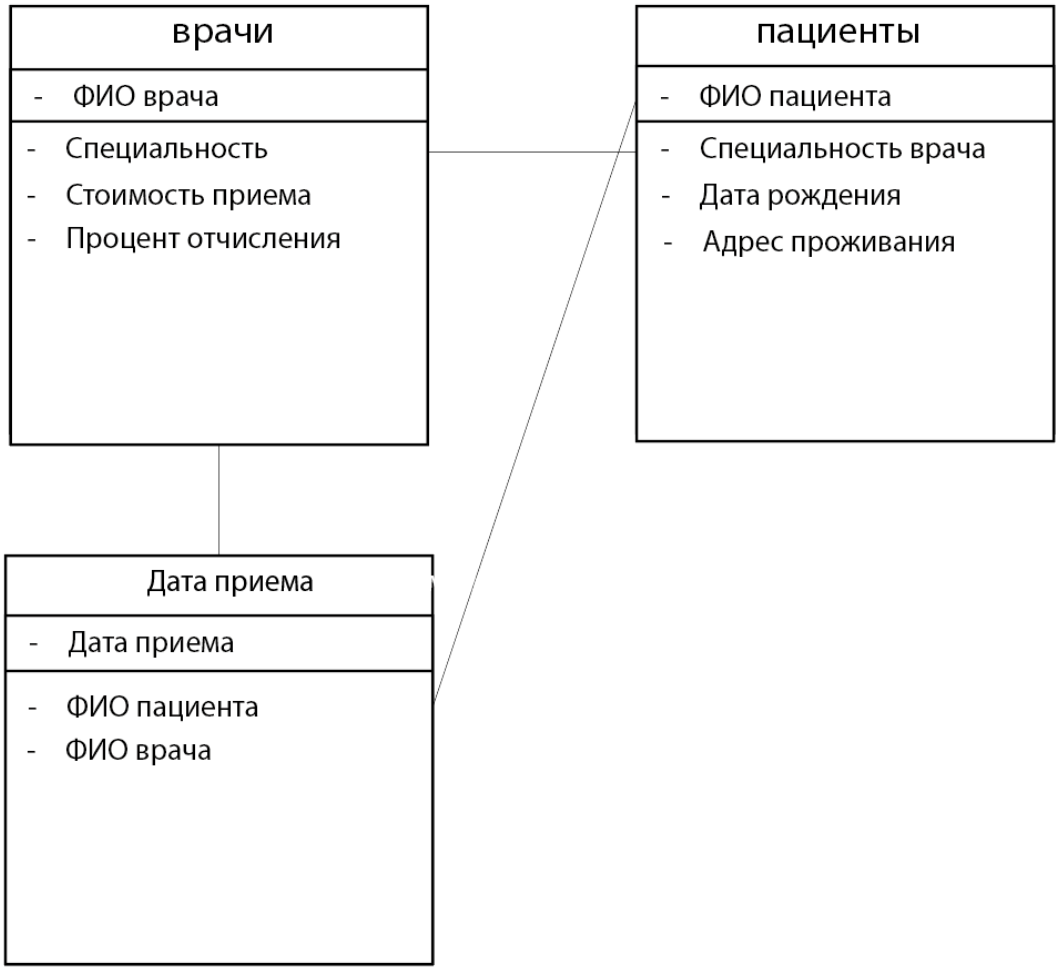

Рисунок 2 – Определенные связи.

4. Построение диаграммы сущность-связь для отображения логической структуры базы данных.

Построенная диаграмма сущность-связь представлена на рисунке 3.

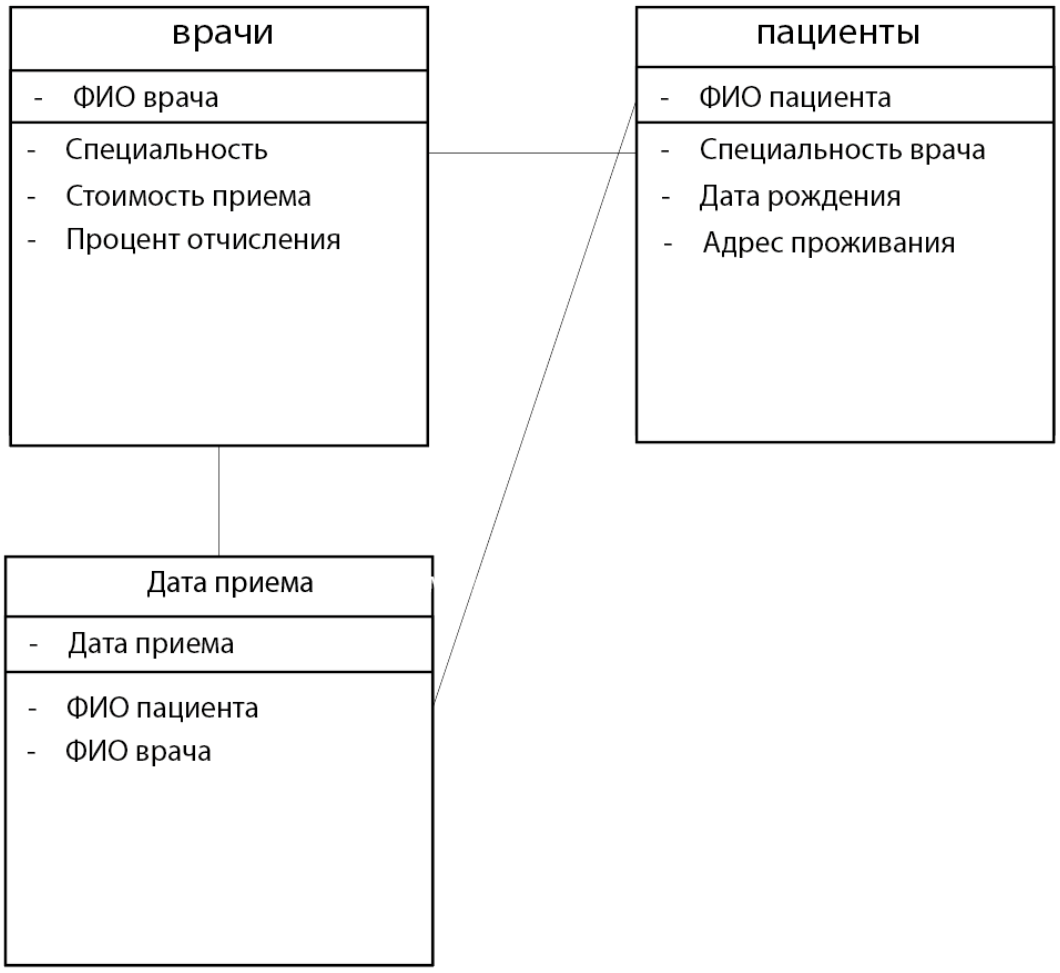

Рисунок 3 – диаграмма сущность-связь

5. **Контрольные вопросы.**

1) Данные – поддающееся многократной интерпретации представление информации в формализованном виде, пригодном для передачи, связи или обработки;

База данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

СУБД – это набор программ, которые управляют структурой БД и контролируют доступ к данным, хранящимся в БД.

Ведение базы данных – деятельность по обновлению, восстановлению и изменению структуры базы данных с целью обеспечения её целостности, сохранности и эффективности использования.

2) Данные становятся информацией при использовании соответствующего метода преобразования данных в известные понятия. Информация не является статичным объектом. Одни и те же данные могут в момент потребления представлять разную информацию.

3) База данных - это хранилище данных, банк данных может представлять собой более крупную систему, объединяющую несколько баз данных, а СУБД - это программное обеспечение для управления и обработки данных в базе данных.

4) Банк данных (или база данных) обычно состоит из нескольких основных компонентов, каждый из которых выполняет определенные функции. Вот основные компоненты банка данных и их назначение:

Таблицы (Tables): Таблицы представляют собой основную структурную единицу базы данных. Они организуют данные в виде строк и столбцов, где каждая строка представляет запись, а каждый столбец - поле данных.

Запросы (Queries): Запросы используются для извлечения данных из таблиц и их обработки. Запросы позволяют выбирать, фильтровать, объединять и сортировать данные в соответствии с определенными критериями.

Формы (Forms): Формы используются для удобного ввода данных в базу данных. Они предоставляют пользовательский интерфейс, который упрощает добавление, редактирование и удаление записей.

Отчеты (Reports): Отчеты предназначены для вывода данных из базы данных в удобном для пользователя виде. Отчеты могут включать в себя таблицы, графику и другие элементы для визуализации информации.

Индексы (Indexes): Индексы создаются для ускорения процесса поиска и сортировки данных. Они позволяют базе данных быстро находить конкретные записи, улучшая производительность.

Ключи (Keys): Ключи используются для уникальной идентификации записей в таблицах. Они помогают обеспечивать целостность данных и связи между различными таблицами.

Триггеры (Triggers): Триггеры представляют собой набор инструкций, которые выполняются автоматически при определенных событиях, таких как вставка, обновление или удаление данных. Они обеспечивают автоматизацию определенных операций.

Схемы (Schemas): Схемы определяют структуру базы данных, включая таблицы, связи между ними, права доступа и другие параметры.

Эти компоненты совместно обеспечивают эффективное хранение, организацию и управление данными в базе данных, делая их доступными и удобными для использования различными приложениями и пользователями.

5) Автоматизированные информационные системы (АИС) можно классифицировать по типу хранимых данных на основе их основной функциональности и предназначения. Вот несколько основных типов АИС в зависимости от характера хранимых данных:

АИС для управления предприятием (ERP - Enterprise Resource Planning):

Хранимые данные: Интегрированные данные о бизнес-процессах, финансах, ресурсах предприятия (кадры, материалы, оборудование), заказах и поставках.

АИС управления отношениями с клиентами (CRM - Customer Relationship Management):

Хранимые данные: Информация о клиентах, контактах, истории взаимодействия с клиентами, продажах и маркетинге.

АИС для обработки транзакций (OLTP - Online Transaction Processing):

Хранимые данные: Транзакционные данные, такие как заказы, платежи, резервирование товаров и другие операции в реальном времени.

АИС для поддержки принятия решений (DSS - Decision Support System):

Хранимые данные: Информация для анализа и принятия стратегических решений, включая статистические данные, отчеты, прогнозы и моделирование.

АИС для обработки данных по научным исследованиям (SDSS - Scientific Data Storage System):

Хранимые данные: Научные данные, результаты экспериментов, измерения, геномные данные, географические данные и другие научные информации.

АИС управления знаниями (KMS - Knowledge Management System):

Хранимые данные: Знания, документация, экспертные знания, базы знаний и другие ресурсы для управления и распределения знаний в организации.

АИС для управления человеческими ресурсами (HRIS - Human Resources Information System):

Хранимые данные: Информация о сотрудниках, кадровые данные, оплата труда, тренинги и развитие персонала.

Эти категории предоставляют общую идею о том, как можно классифицировать АИС в зависимости от хранимых данных и их функциональности. Однако, в реальности, многие системы могут включать в себя элементы различных типов АИС в зависимости от своей комплексности и потребностей организации.

6) **Архитектура ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee)** представляет собой концептуальную модель для организации баз данных. Она определяет структуру и взаимосвязь между различными уровнями базы данных. Эта модель состоит из трех основных уровней:

Внешний уровень (External Level):

Это уровень, который видят конечные пользователи или приложения.

Здесь определяется как информация представляется и видится конкретным пользователем.

Пользователи видят только необходимую им часть данных, что делает систему более удобной в использовании.

Концептуальный уровень (Conceptual Level):

Этот уровень определяет общую структуру и организацию всей базы данных независимо от ее физической реализации.

Он описывает структуру данных, их отношения и ограничения.

Это своего рода промежуточный уровень между внешним и внутренним уровнями, который предоставляет абстрактное представление всей базы данных.

Внутренний уровень (Internal Level):

Этот уровень определяет, как данные хранятся в физическом аспекте.

Здесь учитываются детали реализации базы данных на уровне хранения данных на диске, структуры файлов, методы доступа и оптимизации для обработки запросов.

Такая архитектура позволяет разделять структуру данных на три уровня абстракции, что облегчает разработку, модификацию и поддержку базы данных, поскольку изменения на одном уровне не обязательно должны отражаться на других уровнях.

7) В контексте баз данных (БД) используются три уровня абстракции, представляющие различные виды схем: внешняя схема, концептуальная схема и внутренняя схема.

Внешняя схема (External Schema):

Определение: Внешняя схема представляет собой описание того, как данные видятся конкретному пользователю или группе пользователей. Этот уровень абстракции определяет, какие данные доступны для конкретного пользователя, как они организованы и каким образом пользователи могут выполнять операции с этими данными. Внешняя схема может быть специфичной для приложения или группы пользователей.

Концептуальная схема (Conceptual Schema):

Определение: Концептуальная схема представляет собой общее и независимое от приложения представление данных в базе данных. Она описывает структуру данных, их отношения и основные ограничения. Концептуальная схема используется для определения общего понимания данных в базе данных, и она независима от конкретных технологий или приложений.

Внутренняя схема (Internal Schema):

Определение: Внутренняя схема описывает, как данные физически организованы и хранятся внутри базы данных. Этот уровень абстракции связан с тем, как система управления базами данных (СУБД) реализует концептуальную схему в рамках конкретной архитектуры и хранения данных. Внутренняя схема отражает структуру данных на уровне хранения, индексов, методов доступа и т.д.

Каждая из этих схем играет свою роль в проектировании и использовании баз данных, обеспечивая абстракцию данных на разных уровнях, что упрощает управление и поддержку баз данных.

8) Иерархическая модель организации данных была одной из первых моделей баз данных, разработанной в 1960-х годах. Её основные особенности включают:

- Иерархическая структура: Данные организованы в виде иерархической структуры, напоминающей дерево. Каждый элемент данных связан с одним родительским элементом и может иметь несколько дочерних элементов.
- Поведение "родитель-потомок": В этой модели доступ к данным осуществляется с помощью операции "родитель-потомок". Это означает, что каждый элемент данных имеет только одного родителя, что упрощает доступ к информации.
- Физическая реализация: Часто реализуется в виде иерархической базы данных, где данные хранятся с использованием связей "один-ко-многим".
- Ограничения в сложности запросов: Поскольку доступ к данным осуществляется через их иерархическую структуру, выполнение сложных запросов может быть затруднительным. Запросы могут быть ограничены простыми операциями родитель-потомок.

- Применение: Использовалась в ранних информационных системах, таких как IMS (Information Management System), особенно в коммерческих и банковских приложениях.

- Склонность к дублированию данных: Поскольку каждый элемент связан только с одним родителем, для доступа к тому же элементу из другого контекста может потребоваться дублирование данных, что может привести к избыточности и несогласованности.

- Ограничения в изменяемости: Изменение структуры данных (например, добавление новых типов данных) может быть сложным и требовать изменения всей структуры.

Хотя иерархическая модель организации данных имела свои преимущества и использовалась в прошлом, она имеет свои ограничения по сравнению с более современными моделями баз данных, такими как реляционные или объектно-ориентированные модели.

9) Сетевая модель организации данных представляет собой структуру данных, которая описывает взаимосвязи между различными типами данных в сетевой базе данных. Эта модель была разработана в 1960-х годах и является одним из предшественников реляционной модели данных. Вот несколько особенностей сетевой модели:

Сущности и связи:

- В сетевой модели данные организованы в виде записей (сущностей), которые могут быть связаны друг с другом.
- Сущности связаны отношениями, которые могут иметь различные типы, такие как "родитель-ребенок" или "сосед-сосед".

Структура графа:

- Данные представлены в виде графа, где узлы представляют сущности, а ребра - связи между ними.
- Это отличается от реляционной модели, где данные представлены в виде таблиц.

Повторное использование данных:

- Сетевая модель позволяет повторное использование данных путем создания множественных связей между сущностями.
- Это обеспечивает более гибкую структуру данных в сравнении с реляционной моделью.

Комплексные запросы:

- Запросы в сетевой модели могут быть более сложными, поскольку они могут охватывать различные пути в графе данных.
- Однако это также может сделать запросы более сложными для формулирования.

Примеры сетевых баз данных:

- Примерами сетевых баз данных являются CODASYL (Conference on Data Systems Languages) DBTG (DataBase Task Group) и IDMS (Integrated Database Management System).

Отсутствие явного разделения данных и метаданных:

В сетевой модели метаданные (информация об описании структуры данных) и фактические данные часто хранятся вместе.

Преимущества и недостатки:

- Сетевая модель обеспечивает хорошую производительность при доступе к связанным данным, но ее сложность и трудность сопровождения стали основной причиной перехода к реляционным базам данных.

- В современных системах сетевая модель уступила место более распространенным и понятным моделям, таким как реляционная или объектно-ориентированная модели данных.

10) Многомерная модель организации данных используется для представления информации в виде многомерных кубов, что обеспечивает более эффективный и удобный способ анализа данных. Вот некоторые особенности многомерной модели организации данных:

1. Многомерные кубы (OLAP-кубы): Данные организуются в виде кубов, где каждая ось представляет собой измерение, такое как время, продукт, регион и т.д. Это обеспечивает легкий доступ к данным для анализа в различных измерениях.

2. Измерения: Многомерные модели поддерживают различные измерения данных, что позволяет анализировать информацию с разных точек зрения. Например, если у вас есть данные о продажах, измерения могут включать продукты, регионы, временные периоды и др.

3. Факты: Данные, хранящиеся в многомерной модели, представляют собой факты, связанные с конкретными комбинациями измерений. Факты могут быть числовыми значениями, такими как продажи, прибыль, количество и т.д.

4. Агрегация данных: Многомерные модели позволяют легко агрегировать данные на разных уровнях детализации. Например, можно агрегировать продажи на уровне дня, недели, месяца или года.

5. OLAP-операции: Многомерные модели поддерживают OLAP-операции (Online Analytical Processing), такие как свертка (roll-up), дробление (drill-down), срез (slice) и отсечение (dice), что облегчает анализ данных.

6. Поддержка множественных источников данных: Многомерные модели позволяют интегрировать данные из различных источников, что особенно полезно в среде предприятия, где информация может храниться в различных системах.

7. Удобство анализа: Многомерные модели делают анализ данных более интуитивным и понятным, обеспечивая пользователям простой и эффективный способ взаимодействия с данными.

8. Поддержка различных типов данных: Многомерные модели поддерживают работу с различными типами данных, включая числовые, текстовые, даты и другие.

В целом, многомерные модели организации данных являются мощным инструментом для анализа и представления данных в структурированной форме, что облегчает принятие бизнес-решений и выявление закономерностей.

11) Постреляционная модель организации данных представляет собой эволюцию реляционной модели, которая стала доминирующей в мире баз данных. В постреляционных моделях учитываются новые требования и сценарии использования, которые не всегда легко учесть в традиционных реляционных базах данных. Вот некоторые особенности постреляционной модели:

1. Гибкость схемы данных (Schema Flexibility): В постреляционных базах данных схема данных может быть более гибкой. Это позволяет добавлять новые поля или изменять структуру данных без необходимости пересоздания всей базы данных.

2. Поддержка неструктурированных данных: Постреляционные базы данных могут обрабатывать не только традиционные табличные данные, но и неструктурированные данные, такие как JSON, XML, текстовые документы и изображения.

3. Масштабируемость (Scalability): Постреляционные базы данных обеспечивают более легкую горизонтальную масштабируемость, что означает, что их можно легко масштабировать на более мощные серверы или даже распределенные системы для обработки больших объемов данных.

4. Поддержка NoSQL-типов хранилищ данных: Многие постреляционные базы данных предоставляют поддержку NoSQL-подобных моделей данных, таких как ключ-значение, документоориентированные, столбцовые семейства и графовые базы данных.

5. Горизонтальное расширение (Horizontal Partitioning): Данные могут быть разделены горизонтально, что означает, что они могут быть распределены по разным серверам или узлам для улучшения производительности и распределения нагрузки.

6. Автоматическое масштабирование (Auto-Sharding): Некоторые постреляционные базы данных предоставляют возможность

автоматического разделения (sharding) данных, чтобы обеспечить эффективное распределение данных по различным узлам.

7. Поддержка транзакций и целостности данных: Некоторые постреляционные базы данных сохраняют поддержку транзакций и обеспечивают целостность данных, что является важным требованием для многих приложений.

Важно отметить, что постреляционные базы данных разнообразны, и различные системы могут предоставлять разные функции в зависимости от своей спецификации и назначения.

12) Неструктурированные данные - это данные, не организованные в четкую структуру или формат. Они не подчиняются определенным правилам и схемам, что делает их менее удобными для автоматизированной обработки. Примеры включают текстовые документы, изображения, аудио- и видеофайлы, электронные письма, социальные медиа-посты и прочее.

13) Преимущество использования колоночной СУБД по сравнению с реляционной заключается в более эффективном хранении и обработке данных благодаря тому, что они сохраняют информацию по столбцам, а не по строкам. Это обеспечивает лучшую сжимаемость данных, уменьшение накладных расходов при выполнении агрегатных функций и улучшенную производительность при анализе больших объемов информации.

14) Связь между таблицами в реляционной СУБД осуществляется с использованием ключей. Основной механизм - это установка внешних ключей в одной таблице, которые связываются с первичными ключами в другой таблице. Это создает отношение между данными в различных таблицах, обеспечивая целостность и эффективность запросов.

15) Проблема дублирующих записей в таблице решается на этапе проектирования схемы базы данных путем использования нормализации. Нормализация помогает устранить избыточность данных, разделяя их на отдельные таблицы и устанавливая связи между ними. Это повышает эффективность хранения данных и обеспечивает их целостность.