МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
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

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

По дисциплине: «Системы хранения и обработки данных»

Тема: «Разработка логической структуры базы данных»

Выполнил работу студент группы змИИВТ-241: Фролов Д.С.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Лабораторная работа №7

«Разработка логической структуры базы данных»

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

2. Задачи лабораторной работы:

- определение сущности для проекта в соответствии 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.

3. Результаты лабораторной работы

Для базы данных «Платный прием в поликлинике» можно выделить следующие сущности:

* Врачи;
* Специальность;
* Приемы;
* Пациенты;
* Квитанция об оплате;
* Зарплата врача.

«Врачи» – сущность, представляющая медицинских сотрудников, которые осуществляют прием пациентов. Эта сущность необходима для хранения информации о врачах.

«Пациенты» – сущность, представляющая лиц, обращающихся за медицинской помощью. Эта сущность необходима для хранения информации о пациентах.

«Специальность» – врачи имеют определенные специальности (хирург, терапевт и т.д.). Выделение специальностей в отдельную сущность позволяет более гибко управлять категориями врачей и расширять систему при необходимости.

«Приемы» – сущность, отражающая факт оказания медицинской услуги пациенту врачом в определенную дату. Это ключевая сущность, связывающая пациентов с врачами и фиксирующая детали приема.

«Квитанция об оплате» – квитанции фиксируют финансовые транзакции, связанные с оплатой приемов.

«Зарплата врача» – сущность, представляющая информацию о выплате заработной платы медицинским сотрудникам за каждый месяц.

В таблице 1 определены атрибуты сущностей.

Таблица 1 – Атрибуты сущностей

|  |  |
| --- | --- |
| Сущность | Атрибуты |
| Врачи | ID\_врач, Фамилия, Имя, Отчество, Контактная информация (рабочая почта), Стоимость приема, Процент отчисления на зарплату |
| Пациенты | ID\_пациент, Фамилия, Имя, Отчество, Контактная информация (телефон), Дата рождения, Адрес пациента |
| Приемы | ID\_прием, Дата приема |
| Специальность | ID\_специальность, Название специальности |
| Квитанция об оплате | ID\_квитанция, Способ оплаты |
| Зарплата врача | ID\_зарплата, Год, Месяц, Зарплата за месяц |

Выделим ключевые атрибуты сущностей (первичные и внешние ключи):

* Первичным ключом для сущности «Врачи» является «ID\_врач». «ID\_врач» уникально идентифицирует каждого врача в базе данных;
* Первичным ключом для сущности «Пациенты» является «ID\_пациент». «ID\_пациент» уникально идентифицирует каждого пациента в базе данных;
* Первичным ключом для сущности «Специальность» является «ID\_специальность». «ID\_специальность» уникально идентифицирует каждую медицинскую специальность;
* Первичным ключом для сущности «Приемы» является «ID\_прием». «ID\_прием» уникально идентифицирует каждый прием пациента в медицинское учреждение. При этом атрибуты «ID\_врач» и «ID\_пациент» являются внешними ключами. Данные внешние ключи связывают прием с конкретным врачом и пациентом;
* Первичным ключом сущности «Квитанция об оплате» является ID\_квитанция. «ID\_квитанция» уникально идентифицирует каждую квитанцию об оплате. При этом атрибут «ID\_прием» является внешним ключом. Данный внешний ключ связывает квитанцию с конкретным приемом;
* Первичным ключом сущности «Зарплата врача» является «ID\_зарплата». «ID\_зарплата» уникально идентифицирует зарплату врача за месяц. При этом атрибут «ID\_врач» является внешним ключом, так как запись о зарплате привязана к конкретному врачу.

Определим связи между сущностями:

* Сущность «Врачи» связана с сущностью «Специальность» связью «Многие ко многим» (один врач может иметь несколько специальностей, одна специальность может быть у нескольких врачей);
* Сущность «Врачи» и «Зарплата врача» находятся в отношении «Один ко многим» (один врач может иметь несколько записей о зарплате за разные месяцы);
* Сущность «Врачи» и «Приемы» находятся в отношении «Один ко многим» (один врач может провести много приемов);
* Сущность «Пациенты» и «Приемы» находятся в отношении «Один ко многим» (один пациент может посетить много приемов);
* Сущность «Приемы» и «Квитанция об оплате» находятся в отношении «Один к одному» (каждому приему соответствует одна квитанция).

Избавимся от связи «Многие ко многим» между сущностями «Врачи» и «Специальность». Для этого необходимо ввести промежуточную сущность «Специализация врача». Данная сущность включает первичный ключ «ID\_Врач\_Спец» и внешние ключи «ID\_врач» (ссылка на сущность «Врачи») и «ID\_Специальность» (ссылка на сущность «Специальность»). Теперь связь «Многие ко многим» разбивается на две связи «Один ко многим». Сущность «Врачи» и «Специализация врача» находятся в отношении «Один ко многим». Сущность «Специальность» и «Специализация врача» находятся в отношении «Один ко многим».

Построим диаграмму сущность-связь для отображения логической структуры проектируемой базы данных с использованием средства для разработки draw.io. На диаграмме выделим ключевые атрибуты (PK, FK), обозначим связи. Результат построения диаграммы представлен на рисунке 1.

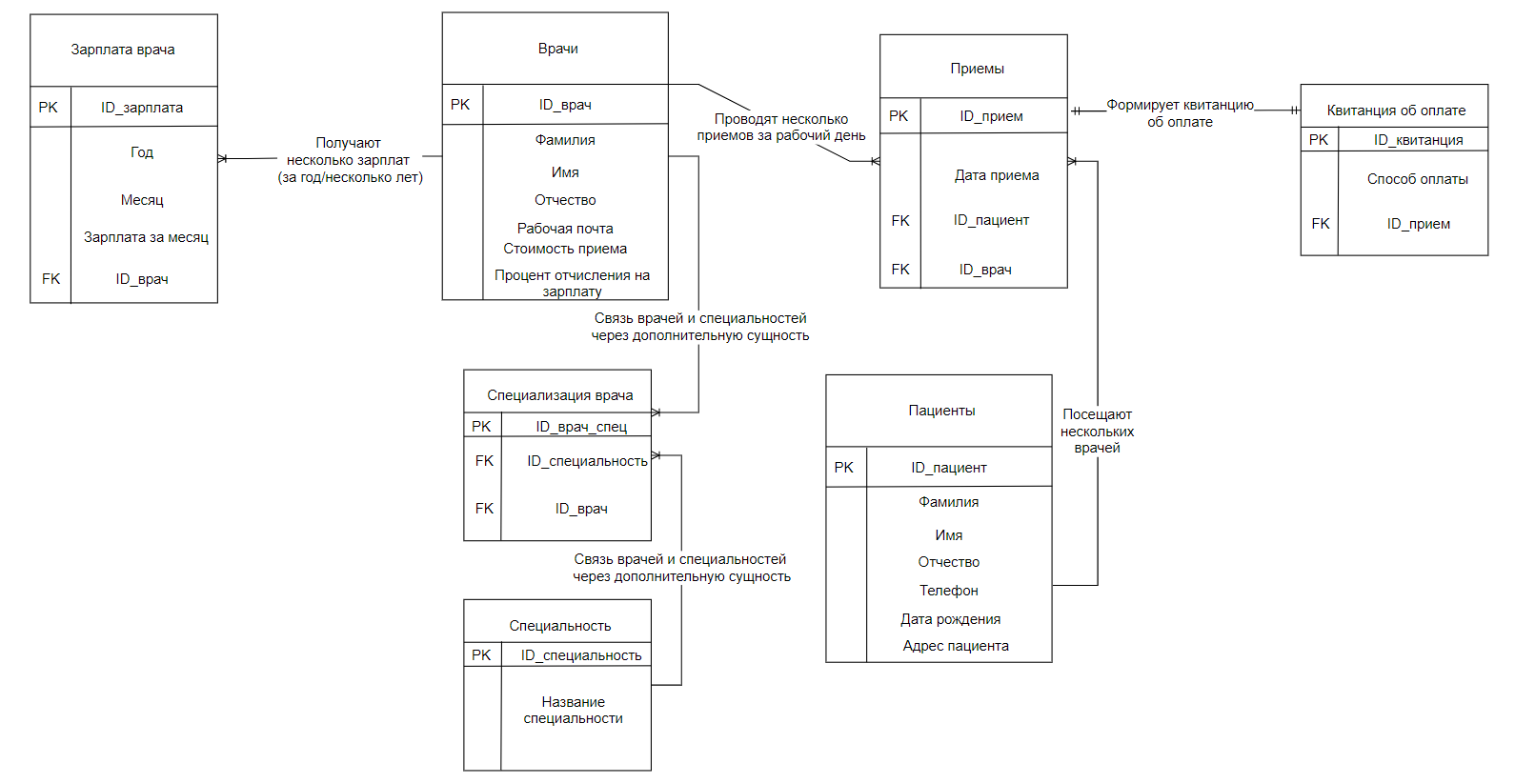


Рисунок 1 – Логическая структура базы данных «Платный прием в поликлинике»

Диаграмма отражает логическую структуру базы данных «Платный прием в поликлинике» и включает в себя следующие основные сущности: «Врачи», «Пациенты», «Приемы», «Специальность», «Квитанция об оплате», «Зарплата врача» и дополнительную сущность «Специализация врача» для реализации связи между врачами и их специальностями.

Сущность «Врачи» содержит данные о врачах, включая стоимость приема и процент отчисления на зарплату, что помогает рассчитывать их доход. Атрибут «ID\_врач» является первичным ключом, который уникально идентифицирует врача. Атрибуты «Фамилия», «Имя» и «Отчество» являются личными данными врача для идентификации. Атрибут «Рабочая почта» предназначен для связи внутри учреждения и с пациентами в рамках профессиональной деятельности. Атрибут «Стоимость приема» предназначен для хранения информации о стоимости приема, которая устанавливается персонально для каждого врача. Атрибут «Процент отчисления на зарплату» используется для расчета заработной платы.

Сущность «Пациенты» хранит данные о пациентах, включая контактные данные и дату рождения. Атрибут «ID\_пациент» является первичным ключом, который уникально идентифицирует пациента. Атрибуты «Фамилия», «Имя» и «Отчество» являются личными данными пациента для идентификации. Атрибут «Телефон» предназначен для связи и оповещений, связанных с приемами. Атрибут «Дата рождения» позволяет учитывать возрастные особенности при лечении. Атрибут «Адрес пациента» предназначен для возможной отправки документов по почте.

Сущность «Приемы» хранит данные о визитах: кто, кто и в какую дату был на приеме. Атрибут «ID\_прием» является первичным ключом, который уникально идентифицирует прием. Атрибут «Дата приема» хранит информацию о дате и времени посещения пациента. Атрибуты «ID\_врач» и «ID\_пациент» являются внешними ключами. Данные внешние ключи связывают прием с конкретным врачом и пациентом.

Сущность «Специальность» позволяет учитывать медицинские специализации врачей. Атрибут «ID\_специальность» является первичным ключом, который уникально идентифицирует специальность. Атрибут «Название специальности» хранит информацию о наименовании медицинской специализации.

Сущность «Квитанция об оплате» предназначена для учета оплаты за прием и привязана к конкретному визиту пациента. Атрибут «ID\_квитанция» является первичным ключом, который уникально идентифицирует квитанцию. Атрибут «Способ оплаты» хранит информацию о способе оплаты за прием (наличные, карта и т.п.). Атрибут «ID\_прием» является внешним ключом. Данный внешний ключ связывает квитанцию с конкретным приемом.

Сущность «Зарплата врача» хранит информацию о заработной плате врача по месяцам. Атрибут «ID\_зарплата» является первичным ключом, который уникально идентифицирует записи о зарплате. Атрибуты «Год» и «Месяц» предназначены для хранения информации о периоде, за который начислена зарплата. Атрибут «ID\_врач» является внешним ключом, так как запись о зарплате привязана к конкретному врачу.

Сущность «Специализация врача» является промежуточной сущностью для реализации связи «Многие ко многим». Атрибут «ID\_врач\_спец» является первичным ключом, который уникально идентифицирует запись. Данная сущность содержит внешние ключи «ID\_врач» (ссылка на сущность «Врачи») и «ID\_Специальность» (ссылка на сущность «Специальность)».

Сущность «Врачи» связана с сущностью «Специализация врача» связью «Один ко многим» – один врач может иметь несколько специализаций. Сущность «Специальность» связана с сущностью «Специализация врача» связью «Один ко многим» – одна специальность может быть у нескольких врачей. Введение сущности «Специализация врача» позволяет устранить связь «Многие ко многим» между врачами и специальностями, нормализуя базу данных.

Сущность «Врачи» связана с сущностью «Приемы» связью «Один ко многим» – один врач может проводить несколько приемов.

Сущность «Пациенты» связана с сущностью «Приемы» связью «Один ко многим» – один пациент может посетить множество приемов.

Сущность «Приемы» связана с сущностью «Квитанция об оплате» связью «Один к одному» – каждому приему соответствует одна квитанция об оплате.

Сущность «Врачи» связана с сущностью «Зарплата врача» связью «Один ко многим» – один врач может иметь множество записей о зарплате за разные месяцы.

Первичные ключи обеспечивают уникальность записей в каждой таблице:

* «ID\_врач» – первичный ключ для сущности «Врачи»;
* «ID\_пациент» – первичный ключ для сущности «Пациенты»;
* «ID\_прием» – первичный ключ для сущности «Приемы»;
* «ID\_специальность» – первичный ключ для сущности «Специальность»;
* «ID\_квитанция» – первичный ключ для сущности «Квитанция об оплате»;
* «ID\_зарплата» – первичный ключ для сущности «Зарплата врача»;
* «ID\_врач\_спец» – первичный ключ для сущности «Специализация врача».

Внешние ключи устанавливают отношения между таблицами, позволяя выполнять запросы, объединять данные и поддерживать ссылочную целостность:

* В сущности «Приемы» – «ID\_врач» (ссылка на таблицу «Врачи») и «ID\_пациент» (ссылка на таблицу «Пациенты»);
* В сущности «Квитанция об оплате» – «ID\_прием» (ссылка на таблицу «Приемы»);
* В сущности «Зарплата Врача» – «ID\_врач» (ссылка на таблицу «Врачи»);
* В сущности «Специализация врача» – «ID\_врач» (ссылка на таблицу «Врачи») и «ID\_специальность» – ссылка на таблицу «Специальности».

Присвоение ключей соответствует нормализации базы данных, избегая избыточности и аномалий при обновлении данных. Избавление от связи «Многие ко многим» между «Врачи» и «Специальность» посредством введения сущности «Специализация врача» соответствует принципам реляционных баз данных. Это позволяет хранить информацию о том, какие врачи обладают какими специальностями, без дублирования данных.

4. Выводы по результатам лабораторной работы

В ходе выполнения данной лабораторной работы были изучены основы логического проектирования базы данных, освоен процесс разработки логической структуры базы данных и построена диаграмма «сущность-связь».

5. Ответы на контрольные вопросы

1. Данные – представление объектов реального мира и их свойств в формализованном виде, пригодном для хранения, передачи, интерпретации или обработки.

База данных (БД) – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ.

Система управления базами данных (СУБД) – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами.

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

2. Данные определяются как неструктурированная информация, такая как тест, наблюдения, изображения, символы и описания. Другими словами, данные не выполняют никакой конкретной функции и сами по себе не имеют значения.

Информация относится к обработанным, организованным и структурированным данным. Она дает контекст фактам и облегчает принятие решений. Другими словами, информация – это обработанные данные, которые имеют смысл для нас.

Данные не зависят от информации, в то время как информация опирается на данные.

Данные напрямую не помогают в принятии решений. Информация напрямую помогает в принятии решений.

3. База данных – это сама коллекция организованных данных.

Банк данных – это, как правило, более масштабное хранилище, состоящее из одной или нескольких баз данных, предназначенное для комплексного управления и анализа больших объемов информации.

СУБД – это программное обеспечение, которое используется для управления базами данных или банками данных, обеспечивая инструменты для их создания, модификации, администрирования и доступа.

4. Банк данных (БнД) в общем случае состоит из следующих компонентов: базы данных (или нескольких баз данных), системы управления базами данных (СУБД), словаря данных, администратора базы данных, вычислительной системы и обслуживающего персонала.

База данных (БД) является ядром банка данных и представляет совокупность взаимосвязанных и вместе хранящихся данных из определенной предметной области, организованных специальным образом и хранимых во внешней памяти (файлах базы данных). В компьютерных базах данных может содержаться любая информация, то простого текста (например, фамилия, имя и адрес) до сложной структуры, включая рисунки, звуки и изображения. Хранение данных в заранее известном формате позволяет извлекать данные в желаемом формате благодаря использованию разных методов обработки.

Система управления базами данных (СУБД) – это комплекс языковых, математических и программных средств, предназначенных для централизованного создания, ведения и совместного использования БД многими пользователями. К функциям СУБД относится:

- перевод схемы, определяющей структуру данных и записанной на языке определения данных в некоторое внутреннее представление, используемой системой при дальнейшей работе с данными;

- создание БД (загрузка данных в БД);

- реализация запросов пользователей (формулируемых на специальном языке, принятом в данной СУБД) на сортировку и отбор по заданным критериям, а также извлечение некоторой части БД, что может сопровождаться редактированием и обработкой информации;

- обновление некоторой части БД без изменения структуры данных;

- обеспечение защиты данных и приоритетов в их использовании.

Можно сказать, что основная функция СУБД – это предоставление пользователю БД возможности работы с ней, не вникая в детали на уровне аппаратного обеспечения. То есть все запросы пользователя к БД, добавление и удаление данных, выборки, обновление данных – это все обеспечивает СУБД.

Словарь данных (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов базы данных друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т.д.

Функционально словарь данных присутствует во всех банках данных, но не всегда выполняющий эти функции компонент имеет именно такое название. Чаще всего функции словаря данных выполняются СУБД и вызываются из основного меню системы или реализуются с помощью ее утилит.

Администратор базы данных (АБД) – это лицо или группа лиц, отвечающих за выработку требований к базе данных, проектирование базы данных, ее создание, эффективное использование и сопровождение.

В процессе проектирования базы данных АБД осуществляет анализ предметной области, проектирование структуры БД, организационно-методическую работу по проектированию БД, задание ограничений целостности при описании структуры БД и процедур обработки БД.

В процессе создания базы данных АБД осуществляет первоначальную загрузку и ведение БД.

В процессе эксплуатации АБД обычно следит за функционированием информационной системы, обеспечивает защиту данных от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой в базе данных информации, обеспечивает восстановление БД, анализирует обращения пользователей, анализирует эффективность функционирования БД, работает с конечными пользователями, занимается модернизацией программных средств и развитием БД.

Вычислительная система (ВС) представляет собой совокупность взаимосвязанных и согласованно действующих ЭВМ или процессоров и других устройств, обеспечивающих автоматизацию процессов приема, обработки и выдачи информации потребителям. Поскольку основными функциями банка данных являются хранение и обработка данных, то используемая вычислительная система, наряду с приемлемой мощностью центральных процессоров (ЦП) должна иметь достаточный объем оперативной и внешней памяти прямого доступа.

Обслуживающий персонал выполняет функции поддержания технических и программных средств в работоспособном состоянии. Он проводит профилактические, регламентные, восстановительные и другие работы по планам, а также по мере необходимости.

5. По типу хранимых данных все ИС подразделяются на фактографические и документальные. Фактографические ИС предназначены для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции. В документальных ИС информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.

6. Трехуровневая архитектура ANSI/SPARC представляет собой абстрактный стандарт проектирования для системы управления базами данных (СУБД).

Цель трехуровневой архитектуры заключается в отделении пользовательского представления БД от ее физического представления. Причины, по которым необходимо выполнить такое разделение:

- каждый пользователь должен иметь возможности обращаться к одним и тем же данным, используя свое собственное представление о них. Каждый пользователь должен иметь возможность изменять свое представление о данных, причем это изменение не должно оказывать влияния на других пользователей;

- взаимодействие пользователя с БД не должно зависеть от особенностей хранения в ней данных;

- администратор БД должен иметь возможность изменять структуру хранения данных в БД, не оказывая влияния на пользовательские представления;

- логическая структура БД не должна зависеть от таких изменений физических аспектов информации, как переключение на новое устройство хранения;

- администратор БД должен иметь возможность изменять концептуальную или логическую структуру БД без какого-либо влияния на всех пользователей.

Таким образом, в модели ANSI/SPARC отражение предметной области представлено моделями данных трех архитектурных уровней: внешнего, концептуального, внутреннего (рисунок 2).

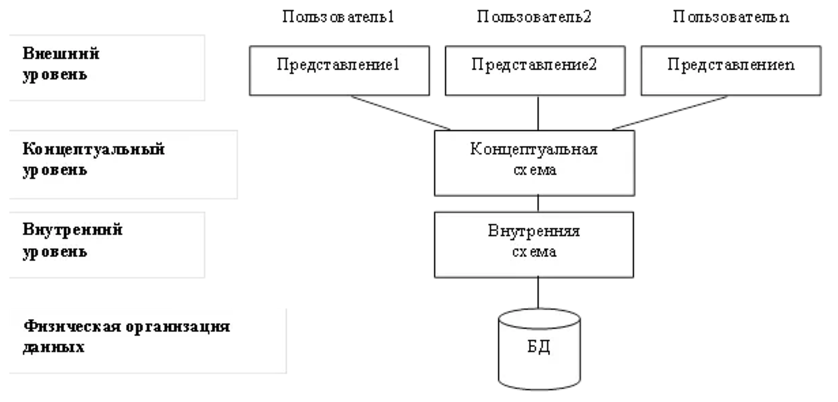


Рисунок 2 – Трехуровневая архитектура БД

Внешний уровень описывает ту часть БД, которая относится к каждому пользователю. Внешний уровень состоит из нескольких различных внешний представлений БД. Каждый пользователь имеет дело с представлением реального мира, выраженным в наиболее удобной для него форме. Внешнее представление содержит только те сущности, атрибуты и связи реального мира, которые интересны пользователю. Другие сущности, атрибуты и связи, которые ему неинтересны, также могут быть представлены в БД, но пользователь может даже не подозревать об их существовании. На внешнем уровне имеется несколько внешних схем или подсистем, которые соответствуют разным представлениям данных.

Концептуальный уровень представляет обобщающее представление БД. Этот уровень описывает то, какие данные хранятся в БД, а также связи, существующие между ними. Концептуальный уровень является промежуточным. Этот уровень содержит логическую структуру всей БД. Однако этот уровень не содержит никаких сведений о методах хранения данных.

Внутренний уровень содержит физическое представление БД в компьютере. Этот уровень описывает, как информация хранится в БД. Внутренний уровень описывает физическую реализацию БД и предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового пространства.

Ниже внутреннего уровня находится физический уровень, который контролируется ОС, но под руководством СУБД.

7. Внешняя схема представляет собой описание того, как отдельные пользователи или группы пользователей видят данные в базе данных. Она определяется пользовательские представления данных, ограничивая доступ к определенным объектам и полям.

Концептуальная схема представляет собой глобальное логическое представление всей базы данных. Она описывает все основные сущности, их свойства, взаимосвязи и ограничения без учета физических особенностей хранения данных.

Внутренняя схема описывает физическое хранение данных в базе данных. Она включает детали о том, как данные организованы на диске, используемые индексы, методы доступа и оптимизации хранения.

8. Модель данных – совокупность правил порождения структур данных в базе данных, операций над ними, а также ограничений целостности, определяющих допустимые связи и значения данных, последовательность их изменения.

Иерархическая модель данных – одна из первых моделей баз данных, разработанная в 1960-х годах. Она представляет данные с использованием древовидных структур, где каждый узел содержит один родительский и несколько дочерних узлов. Эта модель хорошо подходит для отношений «один ко многим», когда родительский объект связан с несколькими дочерними объектами. Иерархическая модель характеризуется простотой и легкостью реализации. Тем не менее, это накладывает некоторые ограничения при работе со сложными отношениями и избыточностью данных. Ключевая особенность данной модели:

- древовидная структура;

- один родительский узел может иметь несколько дочерних узлов, но дочерний узел может иметь только один родительский узел;

- отношения родитель-потомок представлены через родительские указатели или вложенные наборы;

- оптимизирована для перехода от родительского узла к дочернему, а не наоборот;

- лучше всего подходит для отношений «один ко многим».

9. Сетевая модель данных расширяет иерархическую модель, позволяя узлу иметь несколько родительских и дочерних узлов. Эта гибкость позволяет сетевой модели данных представлять отношения «многие ко многим», что делает ее подходящей для более сложных структур данных. Расширение возможностей и гибкости моделирования достигается за счет сложности и производительности. Ключевая особенность:

- графоподобная структура;

- каждый узел может иметь несколько родительских и дочерних узлов;

- отношения представляются с помощью указателей записей, которые напрямую связывают связанные записи;

- идеально подходит для отношений «многие ко многим».

10. Информация в многомерной модели представляется в виде многомерных массивов, называемых гиперкубами. В одной базе данных, построенной на многомерной модели, может храниться множество таких кубов, на основе которых можно проводить совместный анализ показателей. Конечный пользователь в качестве внешней модели данных получает для анализа определенные срезы или проекции кубов, представляемые в виде обычных двумерных таблиц или графиков.

Многомерные базы данных хорошо обслуживают аналитическую обработку данных и обычно являются узкоспециализированными. Они обеспечивают более быстрый поиск и чтение данных по сравнению с реляционными моделями, а также избавляют от необходимости многократного связывания таблиц. Среднее время ответа у них на сложный вопрос в десятки раз меньше, чем при использовании реляционной модели. Основными понятиями для многомерной модели являются: агрегируемость, историчность, прогнозируемость.

Агрегируемость данных означает рассмотрение и возможность анализа данных на разных уровнях обобщения: для пользователя, аналитика, руководителя. Историчность данных обозначает привязку их ко времени и высокий уровень неизменности (статичности) данных и их взаимосвязей. Временная привязка позволяет выполнять запросы, имеющие значения даты и времени. А статичность – использовать специализированные методы загрузки, хранения, выборки. Прогнозируемость данных предполагает задание функций, прогнозирования и применение их к различным временным интервалам.

Многомерность модели данных – это, прежде всего, многомерное логическое представление структуры данных при их описании и в операциях манипулирования ими, а не многомерность их визуализации. По сравнению с реляционной моделью, многомерная организация данных обладает более высокой информативностью.

Для многомерной модели применяются специальные операции: срез, сечение, вращение, агрегация, детализация.

Срез – это подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений.

Сечение – это операция, позволяющая выбрать подмножество данных многомерного куба путем задания конкретных значений или диапазонов для нескольких измерений одновременно.

Вращение изменяет порядок измерений при визуальном представлении данных. Вращение применяется обычно при двумерном представлении данных.

Агрегация – это процесс обобщения данных по измерениям и уровням детализации. Это ключевая особенность многомерных моделей данных, поскольку позволяет пользователям быстро анализировать данные на разных уровнях детализации.

Детализация – это процесс перехода от сводки данных более высокого уровня к более низкому уровню детализации.

11. Постреляционная модель является расширением реляционной модели. Она снимает ограничение неделимости данных, допуская многозначные поля, значения которых состоят из подзначений, и набор значений воспринимается как самостоятельная таблица, встроенная в главную таблицу.

12. Неструктурированные данные – это информация, которая не организована в предопределенную модель или структуру, что затрудняет ее обработку и анализ с помощью традиционных баз данных. К неструктурированным данным относятся:

- текстовые документы: электронные письма, отчеты, статьи;

- мультимедийные файлы: изображения, аудио- и видеофайлы;

- веб-контент: веб-страницы (HTML-код, блоги, форумы), контент сайтов (статьи, новости);

- данные с сенсоров: данные с датчиков движения, температуры, влажности;

- файлы журналов (логов), генерируемые различными системами или приложениями;

- документы с произвольным форматированием: pdf-файлы, презентации (PowerPoint), электронные книги;

- мессенджеры и чаты: сообщения в мессенджерах (WhatsApp, Telegram), чаты на рабочих платформах (Microsoft Teams).

13. Преимущества использования колоночной СУБД по сравнению с реляционной заключается в следующем:

- повышенная производительность для аналитических запросов. Колоночные СУБД оптимизированы для выполнения аналитических и агрегирующих запросов, поскольку позволяют считывать только необходимые столбцы, а не всю строку;

- эффективное сжатие данных. Поскольку данные в одном столбце обычно однородны, их легче и эффективнее сжимать, что снижает объем хранимой информации и улучшает производительность при передаче данных. Меньший размер данных позволяет экономить на ресурсах хранения и ускоряет операции ввода-вывода;

- в колоночных СУБД зачастую требуется меньше индексов, так как доступ к данным осуществляется по столбцам, что упрощает управление базой данных.

14. Связь между таблицами в реляционной СУБД осуществляется через внешний ключ. Он связывает поле (значение) исходной таблицы с первичным ключом внешней таблицы. Выделяют три разновидности связи между таблицами базы данных:

- отношение «один-ко-многим». Одной записи родительской таблицы может соответствовать несколько записей дочерней;

- отношение «один-к-одному». Одной записи в родительской таблице соответствует одна запись в дочерней;

- отношение «многие-ко-многим». Одной записи в родительской таблице соответствует более одной записи в дочерней, и наоборот.

15. На этапе проектирования проблемы дублирующих записей в таблице решаются следующим образом:

- нормализация базы данных. Нормализация – это процесс организации данных в базе данных с целью минимизации избыточности и обеспечения целостности данных. Существует несколько нормальных форм, каждая из которых накладывает определенные ограничения на структуру таблиц. Первая нормальная форма (1NF) – таблица должна содержать только атомарные (неделимые) значения, каждое поле содержит только одно значение, все записи должны быть уникальными. Вторая нормальная форма (2NF) – таблица должна быть в 1NF, все неключевые атрибуты должны полностью зависеть от первичного ключа. Третья нормальная форма (3NF) – таблица должна быть во 2NF, отсутствуют транзитивные зависимости между неключевыми атрибутами;

- использование первичных ключей (Primary Keys). Первичный ключ обеспечивает уникальность каждой записи в таблице. При его правильном выборе каждая строка будет однозначно идентифицирована, что предотвращает появление дубликатов;

- введение уникальных ограничений (Unique Constraints). Уникальные ограничения гарантируют, что значения в определенных столбцах являются уникальными во всей таблице. В отличие от первичного ключа, уникальные ограничения можно применять к нескольким столбцам;

- использование индексов. Создают уникальные индексы, которые не позволяют вставлять повторяющиеся значения в указанные столбцы;

- триггеры и хранимые процедуры могут использоваться для дополнительной проверки данных перед вставкой или обновлением записей, обеспечивая отсутствие дубликатов.