Тема 1. Введение в базы данных

1.1. Основные понятия и определения

Стержневые идеи современных информационных технологий базируются на концепции баз данных.

Согласно этой концепции, основой информационных технологий являются данные, которые должны быть организованы в базы данных в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Одним из важнейших понятий в теории баз данных является понятие информации.

Информация – это любые сведения о каком-либо событии, процессе, объекте.

Данные – это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные – это информация в дискретном, фиксированном виде, удобная для хранения, обработки на ЭВМ, а также для передачи по каналам связи.

База данных (БД) – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД – это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Автоматизированная информационная система (АИС) – это система, реализующая автоматизированный сбор, обработку, манипулирование данными, функционирующая на основе ЭВМ и других технических средств и включающая соответствующее программное обеспечение (ПО) и персонал. В дальнейшем в этом качестве будет использоваться термин информационная система (ИС), который подразумевает понятие автоматизированная.

Каждая ИС в зависимости от ее назначения имеет дело с той или иной частью реального мира, которую принято называть предметной областью (ПрО) системы. Выявление ПрО – это необходимый начальный этап разработки любой

ИС. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных.

Банк данных (БнД) является разновидностью ИС.

Банк данных (БнД) – это система специальным образом организованных данных: баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

Задачи обработки данных – это специальный класс решаемых на ЭВМ задач, связанных с видом, хранением, сортировкой, отбором по заданному условию и группировкой записей однородной структуры.

Отдельные программы или комплекс программ, реализующие автоматизацию решения прикладных задач обработки данных, называются приложениями. Приложения, созданные средствами СУБД, относят к приложениям СУБД. Приложения, созданные вне среды СУБД с помощью систем программирования, использующих средства доступа к БД, к примеру, Delphi или Visual Studio, называют внешними приложениями.

1.2. Современное состояние технологий баз данных

Кратко сформулируем основные современные принципы организации баз данных.

- Значительная часть современных СУБД способна работать на компьютерах различной архитектуры под управлением разных операционных систем.
- Подавляющее большинство современных СУБД обеспечивают поддержку полной реляционной модели данных, обеспечивая целостность категорий и целостность на уровне ссылок.
- Современные СУБД для определения данных и манипуляции ими опираются на принятые стандарты в области языков, а при обмене данными между различными СУБД базируются на существующих технологиях по обмену информацией.
- Многие существующие СУБД относятся к так называемым сетевым СУБД, которые предназначены для поддержки многопользовательского режима работы с базой данных и поддержки возможности децентрализованного хранения данных.
- Такие СУБД имеют развитые средства администрирования баз данных и средства защиты хранимой в них информации.

- Подобные СУБД имеют средства подключения клиентских приложений.
- Современные СУБД характеризуются опытами применения концепции фундаментальной идеи объектно–ориентированного подхода, способствующей повышению уровня абстракции баз данных, являющейся перспективным этапом на пути развития технологий баз данных.

Информационные системы, созданные средствами технологии баз данных, иногда принято называть банками данных (БнД).

БнД включает в себя:

- технические средства;

Технические средства – это совокупность устройств (изделий), обеспечивающих получение, ввод, подготовку, преобразование, обработку, хранение, регистрацию, вывод, отображение, использование и передачу данных, выработку и реализацию управляющих воздействий.

- одну или несколько БД;

Именованная совокупность данных,отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД – это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

- СУБД;

Совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

- словарь или каталог данных;

Служит для централизованного накопления и описания ресурса данных. Он содержит описание ПрО, сведения о структуре БД, о связях между элементами БД. Словарь данных можно рассматривать как часть самой базы данных.

- администратора;

Администратор БД (АБД) – человек или группа лиц, которые принимают решения. Основные функции АБД: участие в разработке БД и выполнение контроля правильности функционирования БД.

- вычислительную систему;

Включает программные (ПС) и аппаратные средства (АС).

- обслуживающий персонал.

Обслуживающий персонал (ОП) – это лица, прямыми обязанностями которых является создание и поддержание корректного функционирования банка данных.

Они ответственны за работу БнД и прикладного программного обеспечения. К обслуживающему персоналу относятся:

- разработчики и администраторы базы данных;
- аналитики;
- программисты.

Схематично это выглядит так, как показано на рисунке 1.1.

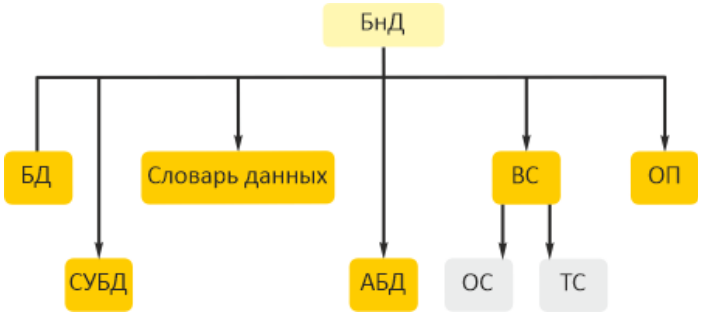


Рисунок 1.1 – Банк данных

1.3. Базы данных

БД, как правило, создается как общий ресурс всего предприятия, где данные являются интегрированными и общими.

Интегрированные данные – это возможность представить базу данных как объединение нескольких отдельных файлов данных.

Общие данные – это возможность использования отдельных областей данных в БД несколькими различными пользователями для разных целей.

В базе данных информация должна быть организована так, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, так как чрезмерная избыточность данных влечет за собой ряд негативных последствий. Главные из них:

- увеличение объема информации, а значит, потребность в дополнительных ресурсах для хранения и обработки дополнительных объемов данных;
- появление ошибок при вводе дублирующей информации, нарушающих целостность базы данных и создающих противоречивые данные.

БД содержит не только данные, всесторонне характеризующие деятельность самой организации, фирмы, процесса или другой предметной области, но и описания этих данных. Информацию о данных принято называть "метаданными", т. е. "данными о данных". В совокупности описания всех данных образуют словарь данных.

В БД должны храниться данные, логически связанные между собой. Для того чтобы данные можно было связать между собой, и связать так, чтобы эти связи соответствовали реально существующим в данной предметной области, последнюю подвергают детальному анализу, выделяя сущности или объекты. Сущность или объект – это то, о чем необходимо хранить информацию. Сущности имеют некоторые характеристики, называемые атрибутами, которые тоже необходимо сохранять в БД. Атрибуты по своей внутренней структуре могут быть простыми, а могут быть сложными. Простые атрибуты могут быть представлены простыми типами данных. Различного рода графические изображения, являющиеся атрибутами сущностей, – это пример сложного атрибута. Определив сущности и их атрибуты, необходимо перейти к выявлению связей, которые могут существовать между некоторыми сущностями. Связь – это то, что объединяет две или более сущностей. Связи между сущностями также являются частью данных, и они также должны храниться в базе данных.

Если все это: сущности, атрибуты сущностей и связи между сущностями определено, то схема базы данных может выглядеть примерно так, как представлено на рисунке 1.2. На нем показан пример схемы базы данных, которую можно назвать ФАКУЛЬТЕТ.

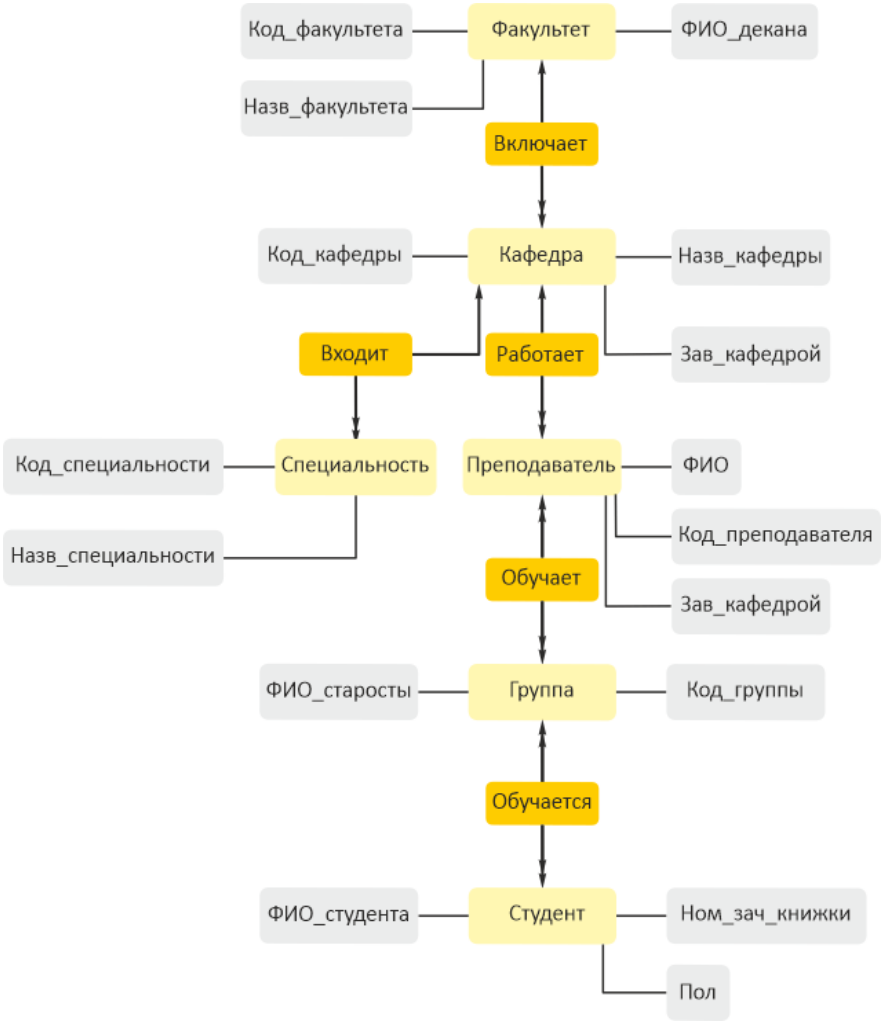


Рисунок 1.2 – Пример ER-диаграммы базы данных ФАКУЛЬТЕТ

Схема, которая называется ER-диаграммой (Entity-Relationship), состоит из следующих компонентов:

- шести сущностей, которые изображены прямоугольниками, каждый из которых имеет свои атрибуты, помещенные в овалы, а в нижеприведенном списке они перечислены в скобках рядом с именем сущностей;
  Сущности и их атрибуты
  ФАКУЛЬТЕТ:(Код_факультета, Назв_факультета, ФИО_декана);
  КАФЕДРА:(Код_кафедры, Назв_кафедры, Зав_кафедрой);
  СПЕЦИАЛЬНОСТЬ:(Код_специальности, Назв_специальности);
  ПРЕПОДАВАТЕЛЬ:(Код_преподавателя, ФИО, Должность);
  ГРУППА:(Код_группы, ФИО_старосты);
  СТУДЕНТ:(Ном_зач_книжки, ФИО, Пол);
- пяти связей, которые обозначены стрелками и связывают те сущности, на которые они направлены.

Связи и сущности

связь **ВКЛЮЧАЕТ** показывает, что на факультете несколько кафедр;

связь **ВХОДИТ** изображает, что одна и та же кафедра готовит специалистов по нескольким специальностям;

связь **РАБОТАЕТ** определяет то, что на кафедре работает ряд преподавателей;

связь **ОБУЧАЕТ** с двойными стрелками в обоих направлениях поясняет тот факт, что один и тот же преподаватель преподает в разных группах, а одна и та же группа занимается с разными преподавателями;

связь **ОБУЧАЕТСЯ** определяет, что каждая группа включает в себя ряд студентов.

Из представленной диаграммы понятно, что данные обладают определенной структурой. Для выявления этой структуры база данных должна пройти процесс проектирования.

Проектируемая БД должна обладать определенными свойствами. Назовем основные свойства БД.

- Целостность: В каждый момент времени существования БД сведения, содержащиеся в ней, должны быть непротиворечивы. Целостность БД достигается вследствие введения ограничений целостности, в частности, к ним относятся ограничения, связанные с нормализацией БД.
- Восстанавливаемость: Данное свойство предполагает возможность восстановления БД после сбоя системы или отдельных видов порчи системы.
- Безопасность: Безопасность БД предполагает защиту данных от преднамеренного и непреднамеренного доступа, модификации или разрушения. Применяется запрещение несанкционированного доступа, защита от копирования и криптографическая защита.
- Эффективность: Свойство эффективности обычно понимается как минимальное время реакции на запрос пользователя, минимальные потребности в памяти, сочетание этих параметров.

1.4. Системы управления базами данных

Итак, как уже не один раз упоминалось, СУБД – это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

По степени универсальности различаются два класса СУБД – системы общего назначения и специализированные системы.

СУБД общего назначения не ориентированы на какую-либо конкретную предметную область или на информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной обстановке. СУБД общего назначения обладает средствами настройки на работу с конкретной БД в условиях конкретного применения.

В некоторых ситуациях СУБД общего назначения не позволяют добиться требуемых проектных и эксплуатационных характеристик (производительность, занимаемый объем памяти и прочее). Тем не менее создание специализированных СУБД весьма трудоемкий процесс и для того, чтобы его реализовать, нужны очень веские основания.

В процессе реализации своих функций СУБД постоянно взаимодействует с базой данных и с другими прикладными программными продуктами пользователя, предназначенными для работы с данной БД и называемыми приложениями.

Для того чтобы СУБД успешно справлялась со своими задачами, она должна обладать определенными возможностями.

Можно дать следующую обобщенную характеристику возможностям современных СУБД.

1. СУБД включает язык определения данных, с помощью которого можно определить базу данных, ее структуру, типы данных, а также средства задания ограничений для хранимой информации. В многопользовательском варианте СУБД этот язык позволяет формировать представления как некоторое подмножество базы данных, с поддержкой которых пользователь может создавать свой взгляд на хранимые данные, обеспечивать дополнительный уровень безопасности данных и многое другое.
2. СУБД позволяет вставлять, удалять, обновлять и извлекать информацию из базы данных посредством языка управления данными
3. Большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами, причем на 12 работу пользователя при доступе к данным практически тип платформы влияния не оказывает.
4. Многопользовательские СУБД имеют достаточно развитые средства администрирования БД.
5. СУБД предоставляет контролируемый доступ к базе данных с помощью:

- системы обеспечения безопасности, предотвращающей несанкционированный доступ к информации базы данных;
- системы поддержки целостности базы данных, обеспечивающей непротиворечивое состояние хранимых данных;
- системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к базе данных;
- системы восстановления, позволяющей восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате аппаратного или программного обеспечения.

Тема 2. Архитектура СУБД

2.1. Трехуровневая архитектура базы данных

Одним из важнейших аспектов развития СУБД является идея отделения логической структуры БД и манипуляций данными, необходимыми пользователям, от физического представления, требуемого компьютерным оборудованием.

Одна и та же БД в зависимости от точки зрения может иметь различные уровни описания. По числу уровней описания данных, поддерживаемых СУБД, различают одно-, двух- и трехуровневые системы. В настоящее время чаще всего поддерживается трехуровневая архитектура описания БД (см. рисунок 2.1), с тремя уровнями абстракции, на которых можно рассматривать базу данных.

Такая архитектура включает:

- внешний уровень, на котором пользователи воспринимают данные, где отдельные группы пользователей имеют свое представление (ПП) на базу данных;
- внутренний уровень, на котором СУБД и операционная система воспринимают данные;
- концептуальный уровень представления данных, предназначенный для отображения внешнего уровня на внутренний уровень, а также для обеспечения необходимой их независимости друг от друга; он связан с обобщенным представлением пользователей.

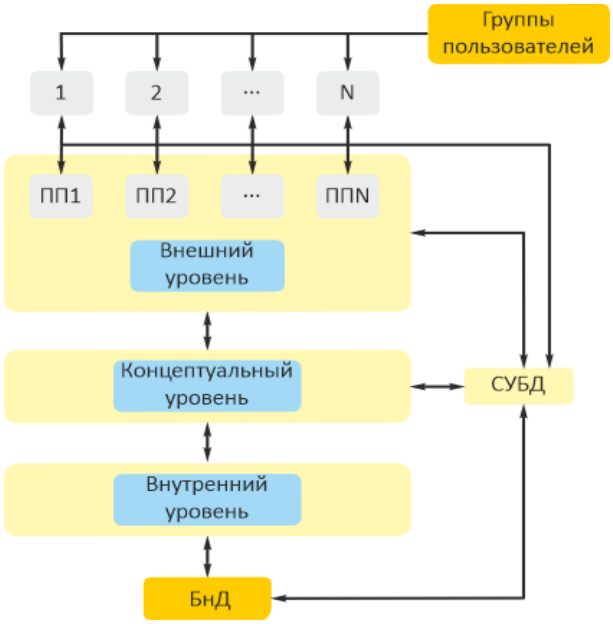


Рисунок 2.1 – Трехуровневая архитектура СУБД

Схема – это описание структуры данных на любом уровне.

Существует три различных типа схем базы данных, которые определяются в соответствии с уровнями абстракции трехуровневой архитектуры. На самом высоком уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных. На концептуальном уровне описание базы данных называют концептуальной схемой, а на самом низком уровне абстракции – внутренней схемой.

Основным назначением трехуровневой архитектуры является обеспечение независимости от данных. Суть этой независимости заключается в том, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

- логическая независимость от данных: Означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему. Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без необходимости внесения изменений в уже существующие внешние схемы для других групп пользователей.
- физическая независимость от данных: Означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему. Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы.

Далее рассмотрим каждый из трех названных уровней

Внешний уровень – это пользовательский уровень. Пользователем может быть программист, или конечный пользователь, или администратор базы данных. Представление базы данных с точки зрения пользователей называется внешним представлением. Каждая группа пользователей выделяет в моделируемой предметной области, общей для всей организации, те сущности, атрибуты и связи, которые ей интересны. Эти частичные или переопределенные описания БД для отдельных групп пользователей или ориентированные на отдельные аспекты предметной области называют подсхемой.

Концептуальный уровень является промежуточным уровнем в трехуровневой архитектуре и обеспечивает представление всей информации базы данных в абстрактной форме. Описание базы данных на этом уровне называется

концептуальной схемой, которая является результатом концептуального проектирования.

Концептуальное проектирование базы данных включает анализ информационных потребностей пользователей и определение нужных им элементов данных.

Концептуальная схема – это единое логическое описание всех элементов данных и отношений между ними, логическая структура всей базы данных. Для каждой базы данных имеется только одна концептуальная схема.

Концептуальная схема должна содержать:

- сущности и их атрибуты;
- связи между сущностями;
- ограничения, накладываемые на данные;
- семантическую информацию о данных;
- обеспечение безопасности и поддержки целостности данных.

Внутренний уровень является третьим уровнем архитектуры БД. Внутреннее представление не связано с физическим уровнем, так как физический уровень хранения информации обладает значительной индивидуальностью для каждой системы.

На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных. Для каждой базы данных существует только одна внутренняя схема.

Внутренняя схема описывает физическую реализацию базы данных и предназначена для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. На внутреннем уровне хранится следующая информация

- распределение дискового пространства для хранения данных и индексов;
- описание подробностей сохранения записей (с указанием реальных размеров сохраняемых элементов данных);
- сведения о размещении записей;
- сведения о сжатии данных и выбранных методах их шифрования.

СУБД отвечает за установление соответствия между всеми тремя типами схем разных уровней, а также за проверку их непротиворечивости.

Ниже внутреннего уровня находится физический уровень, который контролируется операционной системой, но под руководством СУБД. Физический уровень учитывает, каким образом данные будут представлены в

машине. Он обеспечивает физический взгляд на базу данных: дисководы, физические адреса, индексы, указатели и т.д. За этот уровень отвечают проектировщики физической базы данных, которые работают только с известными операционной системе элементами. Область их интересов: указатели, реализация последовательного распределения, способы хранения полей внутренних записей на диске. Однако функции СУБД и операционной системы на физическом уровне не вполне четко разделены и могут варьироваться от системы к системе.

2.2. Функции СУБД

Управление данными во внешней памяти

Данная функция предоставляет пользователям возможности выполнения самых основных операций, которые осуществляются с данными, – это сохранение, извлечение и обновление информации. Она включает в себя обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным.

Управление транзакциями

Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого базы данных.

Примерами простых транзакций может служить добавление, обновление или удаление в базе данных сведений о некоем объекте. Сложная же транзакция образуется в том случае, когда в базу данных требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

Восстановление базы данных

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев:

- мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания);
- жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.

Поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Поддержка языков БД

Для работы с базами данных используются специальные языки, называемые языками баз данных.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language – язык структурированных запросов). Язык SQL позволяет определять схему реляционной БД и манипулировать данными.

Словарь данных

Одной из основополагающих идей рассмотренной выше трехуровневой архитектуры является наличие интегрированного системного каталога с данными о схемах, пользователях, приложениях и т. д. Системный каталог, который еще называют словарем данных, является, таким образом, хранилищем информации, описывающей данные в базе данных. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных: содержится следующая информация:

- имена, типы и размеры элементов данных;
- имена связей;
- накладываемые на данные ограничения поддержки целостности;
- имена пользователей, которым предоставлено право доступа к данным;
- внешняя, концептуальная и внутренняя схемы и отображения между ними;
- статистические данные, например частота транзакций и счетчики обращений к объектам базы данных.

Управление параллельным доступом

Одна из основных целей создания и использования СУБД заключается в том, чтобы множество пользователей могло осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных,

поскольку в этом случае они не могут помешать друг другу. Однако когда два или больше пользователей одновременно получают доступ к базе данных, конфликт с нежелательными последствиями легко может возникнуть, например, если хотя бы один из них попытается обновить данные.

СУБД должна гарантировать, что при одновременном доступе к базе данных многих пользователей подобных конфликтов не произойдет.

Управление буферами оперативной памяти

СУБД обычно работают с БД значительного размера. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Контроль доступа к данным

СУБД должна иметь механизм, гарантирующий возможность доступа к базе данных только санкционированных пользователей и защищающий ее от любого несанкционированного доступа.

В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных: избирательный подход или обязательный подход.

В большинстве современных систем предусматривается избирательный подход, при котором некий пользователь обладает различными правами при работе с разными объектами. Значительно реже применяется альтернативный, обязательный подход, где каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

Поддержка целостности данных

Термин целостность используется для описания корректности и непротиворечивости хранимых в БД данных. Реализация поддержки целостности данных предполагает, что СУБД должна содержать сведения о тех правилах, которые нельзя нарушать при работе с данными, и обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам.

2.3. Языки баз данных

В СУБД поддерживается несколько специализированных по своим функциям подъязыков. Их можно разбить на две категории:

- язык определения данных БД – ЯОД {DDL – Data Definition Language);
- язык манипулирования данными – ЯМД (DML – Data Manipulation , Language).

Язык определения данных

Язык определения данных – описательный язык, с помощью которого описывается предметная область: именуются объекты, определяются их свойства и связи между объектами. Он используется главным образом для определения логической структуры БД.

Схема базы данных, выраженная в терминах специального языка определения данных, состоит из набора определений. Язык ЯОД используется как для определения новой схемы, так и для модификации уже существующей.

Результатом компиляции ЯОД – операторов является набор таблиц, хранимый в системном каталоге, в котором содержатся метаданные – т. е. данные, которые включают определения записей, элементов данных, а также другие объекты, представляющие интерес для пользователей или необходимые для работы СУБД. Перед доступом к реальным данным СУБД обычно обращается к системному каталогу.

Языки манипулирования данными

Язык манипулирования данными содержит набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

Множество операций над данными можно классифицировать следующим образом:

1. операции селекции;
2. действия над данными:

- включение – ввод экземпляра записи в БД с установкой его связей;
- удаление – исключение экземпляра записи из БД с установкой новых связей;
- модификация – изменение содержимого экземпляра записи и коррекция связей при необходимости.

Языки манипулирования данными делятся на два типа. Это разделение обусловлено коренным различием в подходах к работе с данными, а следовательно, различием в базовых конструкциях в работе с данными.

Первый тип – это процедурный ЯМД.

Второй тип – это декларативный (непроцедурный) ЯМД.

К процедурным языкам манипулирования данными относятся и языки, поддерживающие операции реляционной алгебры, которую основоположник теории реляционных баз данных Э. Ф. Кодд ввел для управления реляционной базой данных. Реляционная алгебра – это процедурный язык обработки реляционных таблиц, где в качестве операндов выступают таблицы в целом.

Декларативные языки предоставляют пользователю средства, позволяющие указать лишь то, какие данные требуются. Решение вопроса о том, как их следует извлекать, берет на себя процессор данного языка, работающий с целыми наборами записей.

Реляционные СУБД обычно включают поддержку непроцедурных языков манипулирования данными – чаще всего это бывает язык структурированных запросов SQL или язык запросов по образцу QBE.

В настоящее время нормой является поддержка декларативного языка SQL, в основе которого лежит реляционное исчисление, также введенное Э Коддом. Этот язык стал стандартом для языков реляционных баз данных, что позволяет использовать один и тот же синтаксис и структуру команд при переходе от одной СУБД к другой.

Следует отметить, что язык SQL имеет сразу два компонента: язык DDL (ЯОД) для описания структуры базы данных, и язык DML (ЯМД) для выборки и обновления данных.

Другим широко используемым языком обработки данных является язык QBE, который заслужил репутацию одного из самых простых способов извлечения информации из базы данных. Особенно это ценно для пользователей, не являющихся профессионалами в этой области Язык предоставляет графические средства создания запросов на выборку данных с использованием шаблонов Ответ на запрос также представляет собой графическую информацию

Язык запросов – это часть непроцедурного языка ЯМД, которая отвечает за извлечение данных.

Язык запросов можно определить как высокоуровневый узкоспециализированный язык, предназначенный для удовлетворения различных требований по выборке информации из базы данных.

2.4. Архитектура многопользовательских СУБД

Модели двухуровневой технологии "клиент – сервер"

Систему баз данных можно рассматривать как систему, где осуществлено распределение процесса выполнения по принципу взаимодействия двух программных процессов, один из которых в этой модели называется "клиентом", а другой, обслуживающий клиента, – сервером (машина, хранящая базы данных). Клиентский процесс запрашивает некоторые услуги, а серверный процесс обеспечивает их выполнение. При этом предполагается, что один серверный процесс может обслужить множество клиентских процессов (см. рисунок 2.2).

Рисунок 2.2 – Структура системы БД с выделением клиентов и сервера

Сервер – это собственно СУБД. Он поддерживает все основные функции СУБД и предоставляет полную поддержку на внешнем, концептуальном и внутреннем уровнях.

Клиенты – это различные приложения, которые выполняются над СУБД.

Обычно в приложении выделяются следующие группы функций:

- функции ввода и отображения данных;
- прикладные функции, определяющие основные алгоритмы решения задач приложения;
- функции обработки данных внутри приложения;
- функции управления информационными ресурсами;
- служебные функции, играющие роль связок между функциями первых четырех групп.

Двухуровневая модель – это модель при которой все пять компонентов приложения распределяются только между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере.

Она имеет несколько основных разновидностей.

Файловый сервер

Модель удаленного управления данными – это модель файлового сервера .

Данная модель предполагает следующее распределение функций – на клиенте располагаются почти все части приложения: презентационная часть приложения, прикладные функции, а также функции управления информационными ресурсами. Файловый сервер содержит файлы, необходимые для работы приложений и самой СУБД и поддерживает доступ к файлам (см. рисунок 2.3).

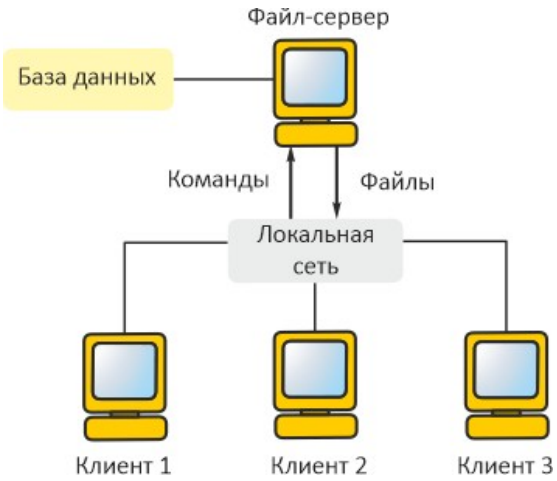


Рисунок 2.3 – Модель файлового сервера

Поскольку передача файлов представляет собой длительную процедуру, такой подход характеризуется значительным сетевым трафиком, что может привести к снижению производительности всей системы в целом.

Помимо этого недостатка использование файлового сервера несет еще и другие:

- на каждой рабочей станции должна находиться полная копия СУБД;
- управление параллельностью, восстановлением и целостностью усложняется, поскольку доступ к одним и тем же файлам могут осуществлять сразу несколько экземпляров СУБД;
- узкий спектр операций манипулирования данными, который определяется только файловыми командами;
- защита данных осуществляется только на уровне файловой системы.

Основное достоинство этой модели, заключается в том, что в ней уже осуществлено разделение монопольного приложения на два

взаимодействующих процесса. При этом сервер может обслуживать множество клиентов, обращающихся к нему с запросами.

Модель удаленного доступа к данным

В модели удаленного доступа база данных также хранится на сервере. На сервере же находится и ядро СУБД. На клиенте располагаются части приложения, поддерживающие функции ввода и отображения данных и прикладные функции.

Клиент обращается к серверу с запросами на языке SQL. Структура модели удаленного доступа приведена на рисунке 2.4.

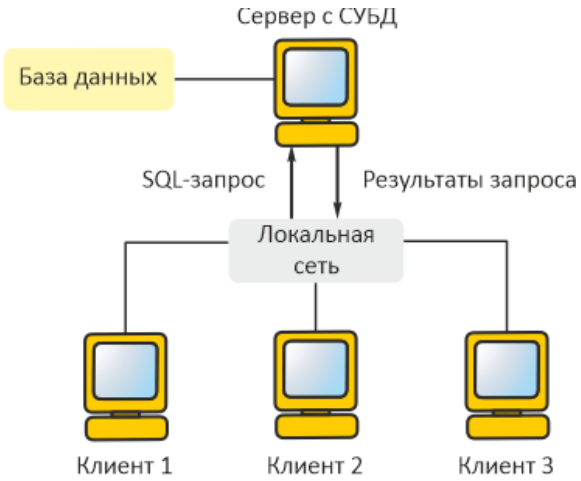


Рисунок 2.4 – Модель удаленного доступа

Сервер принимает и обрабатывает запросы со стороны клиентов, проверяет полномочия пользователей, гарантирует соблюдение ограничений целостности, выполняет обновление данных, выполняет запросы и возвращает результаты клиенту, поддерживает системный каталог, обеспечивает параллельный доступ к базе данных и ее восстановление. К тому же резко уменьшается загрузка сети, так как по ней от клиентов к серверу передаются не файловые команды, а запросы на SQL, и их объем существенно меньше. В ответ на запросы клиент получает только данные, соответствующие запросу, а не блоки файлов, как в модели файлового сервера.

Тем не менее, данная технология обладает и рядом недостатков:

- запросы на языке SQL при интенсивной работе клиентских приложений могут существенно загрузить сеть;
- презентационные и прикладные функции приложения должны быть повторены для каждого клиентского приложения;
- сервер в этой модели играет пассивную роль, поэтому функции управления информационными ресурсами должны выполняться на клиенте

Модель сервера баз данных

Технологию "клиент – сервер" поддерживают большинство современных СУБД: Informix, Ingres, Sybase, Oracle, MS SQL Server. В основу данной модели добавлен механизм хранимых процедур и механизм триггеров.

Механизм хранимых процедур позволяет создавать подпрограммы, работающие на сервере и управляющие его процессами.

Таким образом, размещение на сервере хранимых процедур означает, что прикладные функции приложения разделены между клиентом и сервером. Трафик обмена информацией между клиентом и сервером резко уменьшается.

Централизованный контроль целостности базы данных в модели сервера баз данных выполняется с использованием механизма триггеров. Триггеры также являются частью БД.

Триггер – это особый тип хранимой процедуры, реагирующий на возникновение определенного события в БД.

Он активизируется при попытке изменения данных – при операциях добавления, обновления и удаления. Триггеры определяются для конкретных таблиц БД.

Внедрение триггеров незначительно влияет на производительность сервера и часто используется для усиления приложений, выполняющих многокаскадные операции в БД.

В данной модели (см. рисунок 2.5) сервер является активным, потому что не только клиент, но и сам сервер, используя механизм триггеров, может быть инициатором обработки данных в БД. Поскольку функции клиента облегчены переносом части прикладных функций на сервер, он в этом случае называется "тонким".

При всех положительных качествах данной модели у нее все же есть один недостаток – очень большая загрузка сервера.

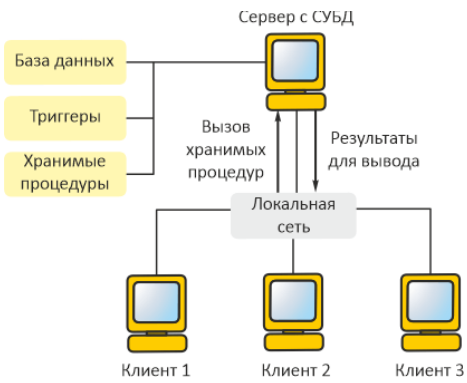


Рисунок 2.5 – Модель сервера БД

Сервер приложений. Трехуровневая модель

Эта модель является расширением двухуровневой модели и в ней вводится дополнительный промежуточный уровень между клиентом и сервером. Архитектура трехуровневой модели приведена на рисунке 2.6.

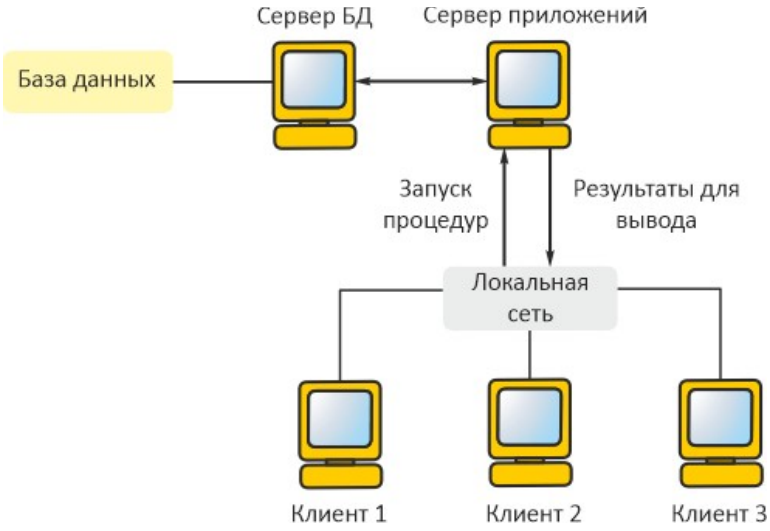


Рисунок 2.6 – Архитектура трехуровневой модели

Такая архитектура предполагает, что на клиенте располагаются: функции ввода и отображения данных, включая графический пользовательский интерфейс, локальные редакторы, коммуникационные функции, которые обеспечивают доступ клиенту в локальную или глобальную сеть.

Серверы баз данных в этой модели занимаются исключительно функциями управления информационными ресурсами БД: обеспечивают функции создания и ведения БД, поддерживают целостность БД, осуществляют функции создания резервных копий БД и восстановления БД после сбоев, управления выполнением транзакций и так далее.

Промежуточному уровню, который может содержать один или несколько серверов приложений, выделяются общие не загружаемые функции для клиентов: наиболее общие прикладные функции клиента, функции, поддерживающие сетевую доменную операционную среду, каталоги с данными, функции, обеспечивающие обмен сообщениями и поддержку запросов.

Преимущества трехуровневой модели наиболее заметны в тех случаях, когда клиенты выполняют сложные аналитические расчеты над базой данных.

Тема 3. Концепции проектирования БД

3.1. Жизненный цикл БД

Как и любой программный продукт, база данных обладает собственным жизненным циклом (ЖЦБД). Главной составляющей в жизненном цикле БД является создание единой базы данных и программ, необходимых для ее работы.

ЖЦБД включает в себя следующие основные этапы (см. рисунок 3.1):

1. планирование разработки базы данных;
   Содержание данного этапа – разработка стратегического плана, в процессе которой осуществляется предварительное планирование конкретной системы управления базами данных.
   Планирование разработки базы данных состоит в определении трех основных компонентов: объема работ, ресурсов и стоимости проекта.

   Важной частью разработки стратегического плана является проверка осуществимости проекта, состоящая из нескольких частей.

   Первая часть – проверка технологической осуществимости. Она состоит в выяснении вопроса, существует ли оборудование и программное обеспечение, удовлетворяющее информационным потребностям фирмы.

   Вторая часть – проверка операционной осуществимости – выяснение наличия экспертов и персонала, необходимых для работы БД.

   Третья часть – проверка экономической целесообразности осуществления проекта. При исследовании этой проблемы весьма важно дать оценку ряду факторов, в том числе и таким:

   - целесообразность совместного использования данных разными отделами;
   - величина риска, связанного с реализацией системы базы данных;
   - ожидаемая выгода от внедрения подлежащих созданию приложений;
   - время окупаемости внедренной БД;
   - влияние системы управления БД на реализацию долговременных планов организации.

2. определение требований к системе;
   На данном этапе необходимо определить диапазон действия приложения базы данных, состав его пользователей и области применения.

Определение требований включает выбор целей БД, выяснение информационных потребностей различных отделов и руководителей фирмы и требований к оборудованию и программному обеспечению.

3. сбор и анализ требований пользователей;
   На данном этапе необходимо создать для себя модель движения важных материальных объектов и уяснить процесс документооборота. По каждому документу необходимо установить периодичность использования, определить данные, необходимые для выполнения выделенных функций (анализируя существующую и планируемую документацию, выясняют, как получается каждый элемент данных, кем получается, где в дальнейшем используется, кем контролируется.
   Собранная информация о каждой важной области применения приложения и пользовательской группе должна включать следующие компоненты: исходную и генерируемую документацию, подробные сведения о выполняемых транзакциях, а также список требований с указанием их приоритетов.

   Формализация собранной на этом этапе информации может быть повышена с помощью методов составления спецификаций требований, к числу которых относятся, например, технология структурного анализа и проектирования, диаграммы потоков данных и графики "вход – процесс – выход".

4. проектирование базы данных;
   Полный цикл разработки базы данных включает концептуальное, логическое и физическое ее проектирование.

   Концептуальное проектирование базы данных

   Первая фаза процесса проектирования базы данных заключается в создании для анализируемой части предприятия концептуальной модели данных.

   Этот подход начинается с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

   Нисходящий подход демонстрируется в концепции модели "сущность – связь" (Entity–Relationship model – ER–модель) – самой популярной технологии высокоуровневого моделирования данных, предложенной П. Ченом.

Модель "сущность – связь" относится к семантическим моделям. Семантическое моделирование данных, связанное со смысловым содержанием данных, независимо от их представления в ЭВМ.

В построении общей концептуальной модели данных выделяют ряд этапов.

- Выделение локальных представлений, соответствующих обычно относительно независимым данным. Каждое такое представление проектируется как подзадача.
- Формулирование сущностей, описывающих локальную предметную область проектируемой БД, и описание атрибутов, составляющих структуру каждой сущности.
- Выделение ключевых атрибутов.
- Спецификация связей между сущностями. Удаление избыточных связей.
- Анализ и добавление неключевых атрибутов.
- Объединение локальных представлений.

Созданная концептуальная модель данных предприятия является источником информации для фазы логического проектирования базы данных.

Логическое проектирование базы данных

Цель второй фазы проектирования базы данных состоит в создании логической модели данных для исследуемой части предприятия.

Глобальная логическая модель данных – это логическая модель, отражающая особенности представления о функционировании предприятия одновременно многих типов пользователей.

Процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД.

Концептуальное и логическое проектирование – это итеративные процессы, которые включают в себя ряд уточнений, продолжающиеся до тех пор, пока не будет получен наиболее соответствующий структуре предприятия продукт.

Физическое проектирование базы данных

Целью проектирования на данном этапе является создание описания СУБД ориентированной модели БД.

Действия, выполняемые на этом этапе, слишком специфичны для различных моделей данных, поэтому их сложно обобщить. Остановимся на реляционной модели данных. В этом случае под физическим проектированием подразумевается:

- создание описания набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;
- определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность системы с базой данных;
- разработка средств зашиты создаваемой системы.

5. разработка приложений;

Разработка приложений

Параллельно с проектированием системы базы данных выполняется разработка приложений. Главные составляющие данного процесса – это проектирование транзакций и пользовательского интерфейса.

Проектирование транзакций

Транзакции представляют некоторые события реального мира.

Транзакция может состоять из нескольких операций, однако с точки зрения пользователя эти операции представляют собой единое целое, переводящее базу данных из одного непротиворечивого состояния в другое. Реализация транзакций опирается на тот факт, что СУБД способна обеспечивать сохранность внесенных во время транзакции изменений в БД и непротиворечивость базы данных даже в случае возникновения сбоя.

Проектирование транзакций заключается в определении:

- данных, которые используются транзакцией;
- функциональных характеристик транзакции;
- выходных данных, формируемых транзакцией;
- степени важности и интенсивности использования транзакции.

Проектирование пользовательского интерфейса

Интерфейс должен быть удобным и обеспечивать все функциональные возможности, предусмотренные в спецификациях требований пользователей.

Специалисты рекомендуют при проектировании пользовательского интерфейса использовать следующие основные элементы и их характеристики:

- содержательное название;
- ясные и понятные инструкции;
- логически обоснованные группировки и последовательности полей;
- визуально привлекательный вид окна формы или поля отчета;
- легко узнаваемые названия полей;
- согласованную терминологию и сокращения;
- согласованное использование цветов;
- визуальное выделение пространства и границ полей ввода данных;
- удобные средства перемещения курсора;
- средства исправления отдельных ошибочных символов и целых полей;
- удобные средства перемещения курсора;
- средства исправления отдельных ошибочных символов и целых полей;
- средства вывода сообщений об ошибках при вводе недопустимых значений;
- особое выделение необязательных для ввода полей;
- средства вывода пояснительных сообщений с описанием полей;
- средства вывода сообщения об окончании заполнения формы.

6. Реализация;

На данном этапе осуществляется физическая реализация базы данных и разработанных приложений, позволяющих пользователю формулировать требуемые запросы к БД и манипулировать данными в БД.

База данных описывается на языке определения данных выбранной СУБД. В результате компиляции его команд и их выполнения создаются схемы и пустые файлы базы данных. На этом же этапе определяются и все специфические пользовательские представления.

Прикладные программы реализуются с помощью языков третьего или четвертого поколений. Кроме того, на этом этапе создаются другие компоненты проекта приложения – например, экраны меню, формы ввода данных и отчеты

Реализация этого, а также и более ранних этапов проектирования БД может осуществляться с помощью инструментов автоматизированного проектирования и создания программ, которые принято называть CASE-инструментами (Computer-Aided Software Engineering).

7. загрузка данных.

   На этом этапе созданные в соответствии со схемой базы данных пустые файлы, предназначенные для хранения информации, должны быть заполнены данными. Наполнение базы данных может протекать поразному, в зависимости от того, создается ли база данных вновь или новая база данных предназначена для замены старой.
8. тестирование.

   Для оценки законченности и корректности выполнения приложения базы данных может использоваться несколько различных стратегий тестирования:

   - нисходящее тестирование;
   - восходящее тестирование;
   - тестирование потоков;
   - интенсивное тестирование.

   Нисходящее тестирование начинается на уровне подсистем с модулями, которые представлены заглушками, т. е. простыми компонентами, имеющими такой же интерфейс, как модуль, но без функционального кода. Каждый модуль низкого уровня представляется заглушкой. Постепенно все программные компоненты заменяются фактическим кодом и после каждой замены снова тестируются.

   Восходящее тестирование выполняется в противоположном направлении по отношению к нисходящему. Оно начинается с тестирования модулей на самых низких уровнях иерархии системы, продолжается на более высоких уровнях и заканчивается на самом высоком уровне.

   Тестирование потоков осуществляется при тестировании работающих в реальном масштабе времени систем, которые обычно состоят из большого количества взаимодействующих процессов, управляемых с помощью прерываний. Стратегия тестирования потоков направлена на слежение за отдельными процессами.

   Стратегия интенсивного тестирования часто включает серию тестов с постепенно возрастающей нагрузкой и продолжается до тех пор, пока система не выйдет из строя.
9. эксплуатация и сопровождение.

   Основные действия, связанные с этим этапом сводятся к наблюдению за созданной системой и поддержке ее нормального функционирования по окончании развертывания.

   Поддержка БД предполагает разрешение проблем, возникающих в процессе эксплуатации БД и связанных как с ошибками реализации БД,

так и с изменениями в самой предметной области, созданием дополнительных программных компонент или модернизацией самой БД.

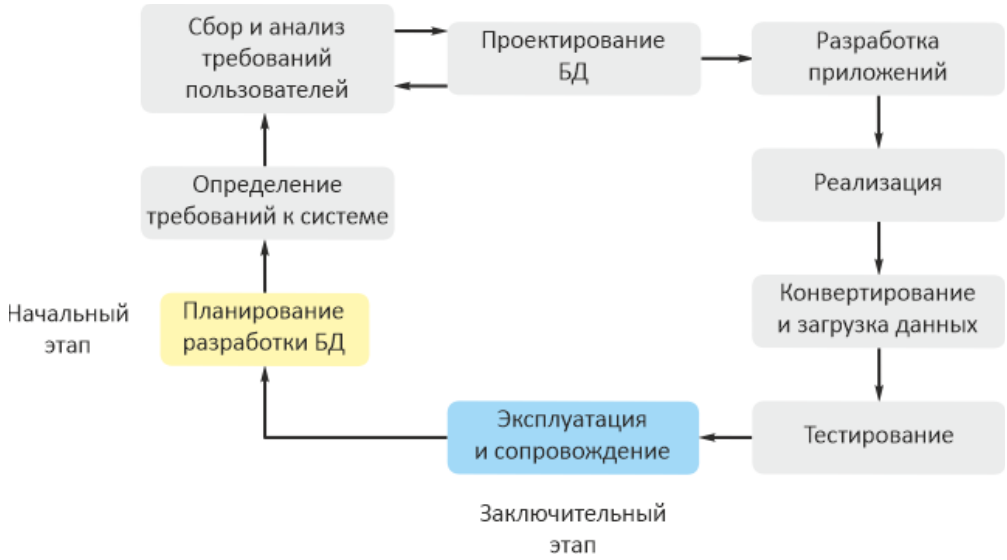


Рисунок 3.1 – Жизненный цикл БД

3.2. Концептуальное проектирование

Фундаментальные понятия

Для нормального функционирования информационной системы необходимо, чтобы концептуальная модель адекватно отображала реалии той предметной области, для которой она разрабатывается. Методологии, позволяющие эффективно отображать существующую смысловую содержательность реальности в конструкции модели, относятся к так называемым семантическим методологиям.

Наиболее популярной семантической моделью стала уже упоминавшаяся раннее модель "сущность – связь" (ER–модель), предложенная П. Ченом в 1976 году, которая с тех пор неоднократно усовершенствовалась самим Ченом и многими другими специалистами.

Главными элементами семантической модели данных являются сущности, их атрибуты и типы связей. Сущности часто представляют в виде существительных, а типы связей – в виде глаголов.

Семантическая модель предметной области изображается в виде диаграммы с учетом принятых обозначений для ее элементов (см. рисунок 3.2).

Сущности

Сущность – это то, о чем накапливается информация в информационной системе и что может быть однозначно идентифицировано.

Сущность – тип (в дальнейшем просто сущность) характеризуется независимым существованием и представляет множество объектов реального мира с одинаковыми свойствами. Отдельные объекты, которые входят в данный тип, называют экземплярами сущности.

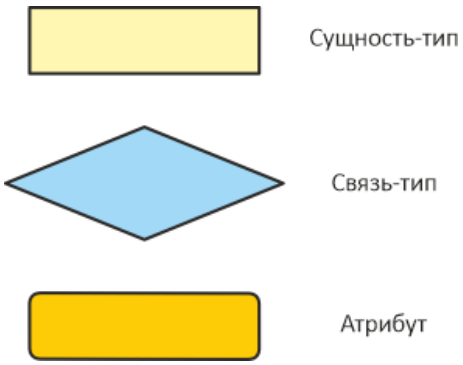
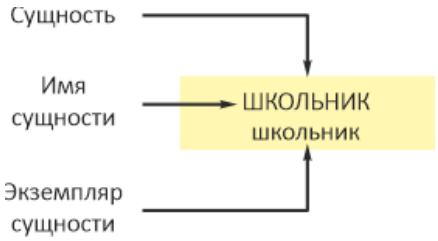


Рисунок 3.2 – Обозначения элементов диаграммы

Каждая сущность имеет имя и изображается на диаграммах в виде прямоугольника, а экземпляр сущности – в виде точки в прямоугольнике данной сущности (см. рисунок 3.3).

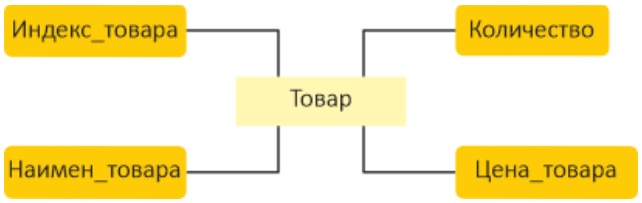


Рисукно 3.3 – Обозначение сущности на ER–диаграмма

Атрибуты

Атрибут – это поименованная характеристика сущности, с помощью которой моделируется ее свойство. Каждой сущности присущи свои атрибуты.

Например, сущность **ТОВАР** должна иметь такие атрибуты: Наименование_товара, Индекс_товара, Цена_товара, Количество. На диаграммах атрибуты сущности соединяются с ней линиями (см. рисунок 3.4).



Рисукно 3.4 – Диаграмма представлений сущности ТОВАР и ее атрибутов

Домен – это множество значений, включающего все потенциальные значения, которые могут быть присвоены атрибуту. Это множество значений называется .

Сущность и экземпляр сущности могут быть определены следующим образом: Сущность: **СТУДЕНТ (ФИО, Группа, Год_рождения)**.

Экземпляр сущности (Петров П.И., 93–ОА–22, 1992).

Значения атрибутов могут часто меняться, в то время как описываемая ими сущность остается той же самой. Так, у экземпляра сущности **СТУДЕНТ** может измениться значение атрибута **ФИО**, но сама сущность останется той же.

Ключи

Ключи – это такие атрибуты, с помощью которых можно идентифицировать экземпляр сущности.

Атрибут или несколько атрибутов, значения которых уникальным образом идентифицируют каждый экземпляр сущности, являются потенциальным ключом данной сущности. Потенциальных ключей может быть несколько. Например, экземпляр сущности **ФАКУЛЬТЕТ (Код_факультета, Название_факультета, ФИО_декана)** может однозначно идентифицироваться любым из первых двух указанных атрибутов.

Один из потенциальных ключей может быть выбран в качестве первичного ключа. Обычно в качестве первичного ключа выбирается тот, который имеет наименьшую длину. Остальные потенциальные ключи называются альтернативными. Тот факт, что атрибут служит первичным ключом, отмечается его подчеркиванием.

Идентификацию некоторых сущностей иногда приходится осуществлять при помощи составных ключей, которые включают несколько атрибутов.

Например, сущность:

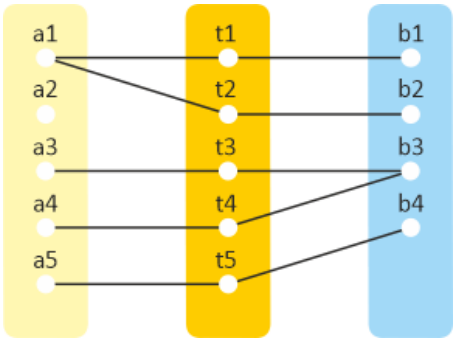
ЛЕЧЕНИЕ (ФИО_врача, ФИО_пациента, Дата_назначения, Лекарство), однозначно идентифицировать можно только составным ключом: **(ФИО_врача, ФИО_пациента, Дата_назначения).**

Связи между сущностями

Две сущности могут быть связаны между собой. Подобная связь осуществляется через связь экземпляров одной сущности с экземплярами другой сущности, образуя набор экземпляров связи между двумя сущностями, который называется типом связи.

Каждому типу связи присваивается имя, которое должно представлять его функцию. Рассмотрим сущности **ПРЕПОДАВАТЕЛЬ** и **КУРС**. Между этими сущностями можно определить связь **ЧИТАЕТ**, сопоставив каждому

преподавателю ту дисциплину, по которой он читает лекции, или, наоборот, каждой дисциплине – преподавателя. Связь **ЧИТАЕТ** составлена из множества пар, в каждой из которых преподаватель – из сущности **ПРЕПОДАВАТЕЛЬ**, а дисциплина – из сущности **КУРС** (см. рисунок 3.5).



Рисукно 3.5 – Экземпляры типа связи ЧИТАЕТ

Полученная структура сама по себе является сущностью, состоящей из пар экземпляров, взятых из двух сущностей, связанных между собой. Сущность **ЧИТАЕТ**, полученная путем связи между сущностями **ПРЕПОДАВАТЕЛЬ** и **КУРС**, называется составной сущностью.

Описанная ситуация на диаграммах имеет свое графическое изображение, где тип связи обозначается в виде ромбика с указанным на нем именем связи, который соединен линиями со связываемыми сущностями (см. рисунок 3.6).

исукно 3.6 – Диаграмма типа связи ЧИТАЕТ

В связи могут участвовать не две, а большее количество сущностей, которые в данном случае являются участниками этой связи.

Степень связи – это количество участников некоторой связи.

В подавляющем числе случаев проектирования БД можно ограничиться рассмотрением бинарных связей.

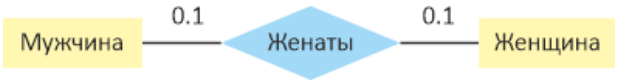
Мощность связи

Мощность – это максимальное количество экземпляров одной сущности, связанных с одним экземпляром другой сущности.

Например, если допустить, что у человека может быть только один супруг, то мощность связи **ЖЕНАТЫ** будет равна одному в каждом направлении (см. рисунок 3.7).

Иногда помимо максимальной мощности полезно определять и минимальную мощность. В рассматриваемом примере не исключаются одинокие мужчины и

женщины, поэтому минимальная мощность равна нулю в каждом направлении. Такая ситуация может быть обозначена следующим образом:



Рисукно 3.7 – Диаграмма связи ЖЕНАТЫ с указание минимальной и максимальной мощности

Некоторые связи не имеют конкретного значения максимальной мощности Например, преподаватель может читать не один курс, а, возможно, больше. Такую мощность обозначают: 1,*, где 1 обозначает минимальную мощность. а * обозначает "много" (существует и другой способ обозначения "много" – вместо * ставится N). С другой стороны, если допустить, что каждый данный курс читается одним и только одним преподавателем, то мощность в обратном направлении будет 1,1.

Показатель кардинальности

Показатель кардинальности – это показатель количество возможных связей для каждого экземпляра участвующего в связи сущностию.

Для бинарных связей показатель кардинальности может иметь следующие значения:

"один к одному" (1 : 1), *"один ко многим"* (1 : N), *"многие ко многим"* (M : N).

Если максимальная мощность связи в обоих направлениях равна одному, мы называем ее связью *"один к одному"* (1 : 1).

Например, на факультете может быть один декан, и обратно, один и тот же декан может руководить только одним факультетом, что может быть обозначено следующим образом:

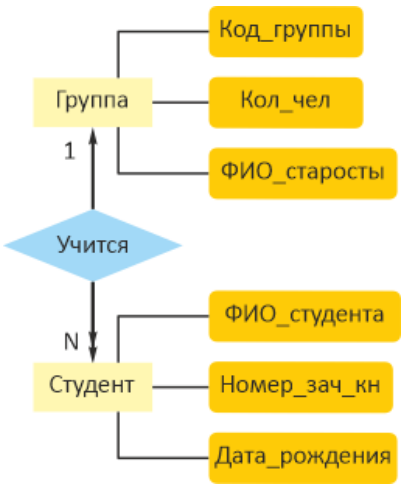
ФАКУЛЬТЕТ <===> ДЕКАН.

Если максимальная мощность в одном направлении равна одному, а в другом – многим, то связь называется *"один ко многим"* (1 : N).

Например, в группе учится много студентов, но каждый студент учится только в одной группе:

ГРУППА <===>> СТУДЕНТ.

Концептуальная модель подобной связи приведена на рисунке 3.8.



Рисукно 3.8 – Диаграмма типа связи УЧИТСЯ с указанием показателя кардинальности

На диаграмме использовано два способа обозначения вида бинарной связи: символическая (со стороны сущности **ГРУППА** выход связи помечен символом 1, а со стороны сущности **СТУДЕНТ** – символом N) и стрелками (в направлении, где максимальная мощность равна многим, проставлена двойная стрелка, а со стороны, где она равна единице – одинарная). Реально при построении диаграмм выбирают один из них.

И наконец, если максимальная мощность в обоих направлениях равна многим, то такая связь относится к типу "многие ко многим" (M : N). Например, преподаватель работает в разных группах, и в одной и той же группе работают различные преподаватели:

ПРЕПОДАВАТЕЛЬ <<===>> ГРУППА.

Связь между сущностями осуществляется посредством атрибутов. Например, рассмотрим две сущности:

Сущность: **СТУДЕНТ**

Атрибуты: **Номер_зачетной_книжки, ФИО студента**

Сущность: **ГРУППА**

Атрибуты: **Код_группы, Количество студентов, ФИО старосты**.

Для их связи в число атрибутов сущности **СТУДЕНТ** необходимо добавить код группы, в которой он учится, и значение которого будет использовано для связи экземпляра одной сущности с экземпляром другой сущности.

Супертип и подтип

Для удовлетворения новых требований, выдвигаемых все более усложняющимися приложениями, в семантическое моделирование были

введены дополнительные концепции, расширяющие его возможности. Дополнительные концепции базируются на таких понятиях, как супертип и подтип, а также используют процесс наследования атрибутов.

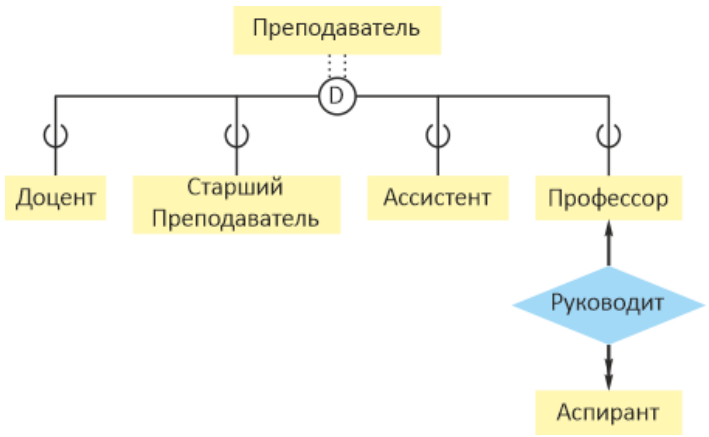
Супертип – это сущность, включающая разные подтипы, которые необходимо представить в модели данных.

Подтип – это сущность, являющаяся членом супертипа, но выполняющая отдельную роль в нем.

Супертип может иметь несколько разных подтипов. Так, например, подтипы: **АССИСТЕНТ, СТАРШИЙ ПРЕПОДАВАТЕЛЬ, ДОЦЕНТ, ПРОФЕССОР** являются членами супертипа **ПРЕПОДАВАТЕЛЬ**. Это означает, что каждый экземпляр подтипа является в то же время и экземпляром супертипа. Связь между супертипом и подтипом относится к типу "один к одному".

Использование понятий супертипа и подтипов позволяет при моделировании выделить для подтипа свои собственные атрибуты и атрибуты, наследуемые им от супертипа.

На диаграмме (см. рисунок 3.9) подтипы соединяются линиями с кружком, который в свою очередь соединяется с супертипом. На каждой линии, идущей от подтипа, располагается U-образный символ, который обозначает направление включения. Верхняя часть **U** "открывается" в сторону супертипа. Внутри кружка располагается буква **D**, если подтипы не пересекаются, и буква **O** – для пересекающихся подтипов. В последнем случае экземпляр супертипа может быть членом сразу нескольких подтипов. Изображенная на диаграмме ситуация исключает пересечение подтипов, поэтому в кружок помещен символ **D**.



Рисукно 3.9 – Диаграмма с использованием понятий супертип и подтип

3.3. Пример моделирования локальной ПРО

С помощью рассмотренных выше понятий могут быть получены ERмодели для большинства схем баз данных в традиционных административно-управленческих приложениях. Если ПрО обширная, то построение ее концептуальной модели будет протекать более успешно, если эту ПрО разбить на несколько локальных предметных областей. Объем локальной ПрО выбирается таким образом, чтобы в нее входило не более 6 – 7 сущностей. Как ранее упоминалось, отправными элементами для построения ER–модели локальной ПрО очень часто являются используемые в организации документы.

Предположим, что определена локальная ПрО: поставка товаров на склад. Пусть используемая форма поставки имеет вид, как на рисунке 3.10.

Покажем, как, используя приведенную форму, можно построить концептуальную модель этой небольшой локальной предметной области.

Рисунок 3.10 – Форма поставки

Итак, анализируемая форма содержит следующую информацию: **Поставщик, Индекс поставщика, Адрес поставщика, Товар. Индекс товара, Цена товара. Количество товара, Поставка, Индекс поставщика, Дата поставки** и **Номер склада.**

Выделим две сущности: **ПОСТАВЩИК** и **ТОВАР** (см. рисунок 3.11).

Оставшиеся атрибуты характеризуют сущность – **ПОСТАВКА**. Сформируем ее и установим определенные типы бинарных связей между тремя сущностями, исходя из следующих рассуждений: один и тот же поставщик может осуществить ряд поставок, но каждая поставка осуществляется только одним поставщиком.

Мощность связи между сущностями **ПОСТАВКА** и **ТОВАР** должна быть установлена M:N, так как каждая поставка может содержать несколько товаров, и один и тот же товар может содержаться в нескольких поставках. Исходя из вышесказанного, диаграмма модели предметной области **ПОСТАВКА** примет такой вид, как на рисунке 3.12.

Рисунок 3.11 – ПрО ПОСТАВКА

Атрибуты **Индекс поставщика, Индекс поставки** и **Индекс товара** были введены дли однозначной идентификации экземпляров рассматриваемых сущностей, так как ни один из остальных атрибутов не подходит на эту роль. Как уже упоминалось, такие идентификационные атрибуты называются первичными ключами.

При построении концептуальной модели следует избегать избыточности информации. После того, как выделены сущности, ключи, определяют и удаляют имеющиеся избыточные связи. Большое внимание уделяется анализу атрибутов. Забегая вперед, следует указать на то, что в хорошо спроектированной БД должно соблюдаться правило: среди атрибутов сущности должна наблюдаться зависимость описательного атрибута от ключевого, но не должна существовать зависимость между описательными атрибутами.

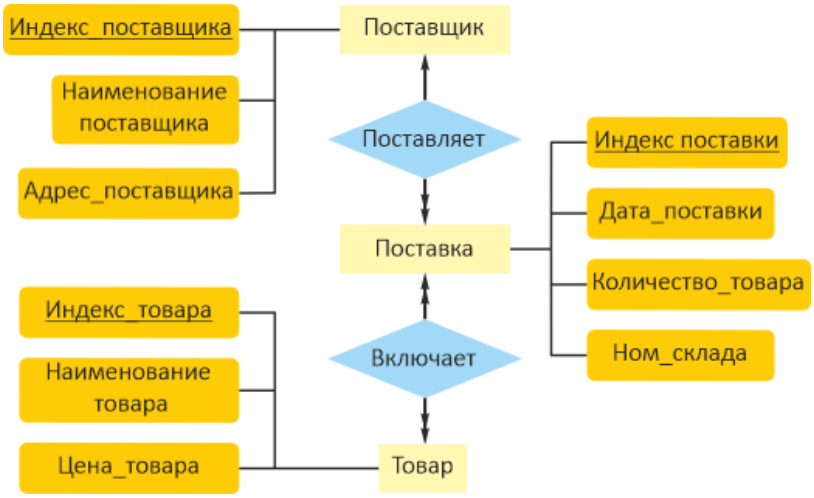


Рисунок 3.12 – Диаграмма модели предметной области ПОСТАВКА

Завершающим этапом построения концептуальной модели исследуемой ПрО является спецификация всех сущностей, входящих в модель. Для данного примера результаты этого шага должны быть сведены к следующему:

- спецификация сущностей;
  ПОСТАВЩИК:
  Индекс_поставщика – идентификационный атрибут
  Адрес_поставщика – описательный атрибут
  Наименование_поставщика – описательный атрибут
  ПОСТАВКА:
  Индекс_поставки – идентификационный атрибут
  Количество_товара – описательный атрибут
  Дата_поставки – описательный атрибут
  Номер_склада – описательный атрибут
  ТОВАР:
  Индекс_товара – идентификационный атрибут
  Наименование_товара – описательный атрибут
  Цена_товара – описательный атрибут
- спецификация типов связей;
  ПОСТАВКА:связь**ПОСТАВЩИК <——> ПОСТАВКА** 1:N
  ВКЛЮЧАЕТ:связь**ПОСТАВКА <——> ТОВАР** M:N
- спецификация атрибутов.
  Индекс_поставщика: символьный, 6 символов
  Адрес_поставщика: символьный, 50 символов
  Цена_товара: денежный

Сформированные спецификации заносятся в словарь данных.

После создания моделей каждой выделенной предметной области производится объединение локальных концептуальных моделей в одну общую, как правило, довольно сложную схему.

Тема 4. Модели данных

4.1. Классификация моделей данных

Модель данных – это формализованное описание структуры единиц информации и операций над ними в информационной системе

Модель данных – это некоторая абстракция, в которой отражаются самые важные аспекты функционирования выделенной предметной области, а второстепенные – игнорируются. Модель данных включает в себя набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные. В модели данных различают три главные составляющие:

- структурную часть, определяющую правила порождения допустимых для данной СУБД видов структур данных;
- управляющую часть, определяющую возможные операции над такими структурами;
- классы ограничений целостности данных, которые могут быть реализованы средствами этой системы.

Каждая СУБД поддерживает ту или иную модель данных.

По существу модель данных, поддерживаемая механизмами СУБД, полностью определяет множество конкретных баз данных, которые могут быть созданы средствами этой системы, а также способы модификации состоянии БД с целью отображения тех изменений, которые происходят в предметной области.

В настоящее время описано много разнообразных моделей, построение которых преследует разные цели. Из множества опубликованных моделей данных можно выделить три категории:

- **объектные модели данных**. Среди объектных моделей следует выделить ER–модель, которая наиболее часто используется в методологии проектирования баз данных, а также объектно-ориентированную модель, последнее время широко используемую в технологиях баз данных. Объектно-ориентированная модель расширяет понятие объекта, включая в него не только атрибуты, характеризующие состояние объекта, но и связанные с ним действия.
- **модели данных на основе записей.** В модели данных на основе записей база данных состоит из нескольких записей фиксированного формата, которые могут иметь разные типы.

В большинстве коммерческих СУБД используются ставшие классическими два типа такого рода моделей данных: теоретико-графовые (ТГ) и теоретико-множественные (ТМ) модели данных.

К теоретико-графовым моделям относятся две разновидности:

- сетевые модели;
- иерархические модели.

В таких моделях данных предусматриваются характерные для подобного рода структур операции навигации и манипулирования данными.

Аппарат навигации в ТГ-моделях служит для установки тех объектов данных, к которым будет применяться очередная операция манипулирования данными.

Теоретико-множественные модели используют математический аппарат, реляционную алгебру (знаковая обработка множеств), реляционное исчисление. К моделям данного типа относятся реляционные модели.

В соответствии с реляционной моделью данных БД представляется в виде совокупности таблиц, над которыми могут выполняться операции, формируемые в терминах реляционной алгебры и реляционного исчисления.

- **физические модели данных**. Физические модели данных описывают то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Наиболее распространены из них следующие: обобщающая модель и модель памяти кадров.

4.2. Сетевая модель

Сети – естественный способ представления отношений между объектами, всевозможных их взаимосвязей. Сетевая модель опирается на математическую структуру, которая называется направленным графом. Направленный граф состоит из узлов, соединенных ребрами. В контексте моделей данных узлы представляют собой объекты в виде типов записей данных, а ребра – связи между объектами со степенью кардинальности "один к одному" или "один ко многим.

Иерархическая модель данных является частным случаем сетевой модели.

Структуры данных сетевой модели

В сетевой модели используются несколько различных типовых структур данных, главными из которых являются типы записей и наборы. Для построения этих структур применяются такие конструктивные элементы, как элемент данных и агрегат (см. рисунок 4.1).

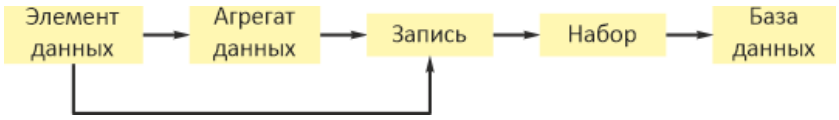


Рисунок 4.1 – Основные структуры сетевой модели данных

Элемент данных – это наименьшая поименованная информационная единица данных, доступная пользователю (аналог – поле в файловой системе). Элемент данных должен иметь свой тип (не структурный, простой).

Агрегат данных соответствует следующему уровню обобщения – поименованная совокупность элементов данных внутри записи или другого агрегата (см. рисунок 4.2).

Рисунок 4.2 – Агрегат Дата

Запись – конечный уровень агрегации. Каждая запись представляет собой именованную структуру, содержащую один или более именованных элементов данных, каждый из которых обладает своим особым форматом.

Агрегат данных **Дата** входит в состав записи Сотрудник (см. рисунок 4.3).

Тип записей – это совокупность логически связанных экземпляров записей. Тип записей моделирует некоторый класс объектов реального мира.

В качестве элемента данных могут быть использованы только простые типы, а в качестве агрегатов могут быть использованы сложные типы: вектор и повторяющаяся группа.

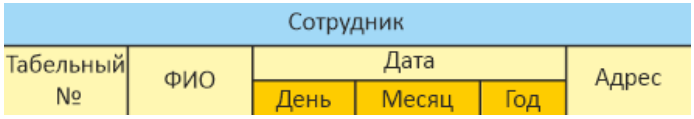


Рисунок 4.3 – Запись Сотрудник

Агрегат типа вектор соответствует линейному набору элементов данных (см. рисунок 4.2). Агрегат типа повторяющаяся группа соответствует совокупности векторов данных. Так, например, в заказе может быть указано несколько видов товаров с числом повторений 10 (см. рисунок 4.4).

Рисунок 4.4 – Запись Заказ

Набор – это поименованная двухуровневая иерархическая структура, которая содержит запись владельца и записи членов.

Наборы выражают связи "один ко многим" или "один к одному" между двумя типами записей. Тип набора поддерживает работу с внутренними структурами типов записей.

Набор, приведенный на рисунке 4.5, определяет тип записи-владельца **Отдел** и тип записи-члена набора **Сотрудник**, а также тип связи между ними "один ко многим" – с именем **Работает**. Имя набора – это метка, присвоенная стрелке. Связь типа "один ко многим" допускает возможность того, что с данным экземпляром записи-владельца может быть связан ноль, один, или несколько экземпляров записи-члена.

Рисунок 4.5 – Диаграмма типа набора Работает

Используя понятия сетевой модели данных, можно получить другое изображение такого набора (см. рисунок 4.6), где представлены логические типы записей **Отдел** и **Сотрудник**, их структура и связь между типами записей **Работает**.

База данных в сетевой модели данных – это поименованная совокупность экземпляров записей различного типа и экземпляров наборов, содержащих связи между ними.

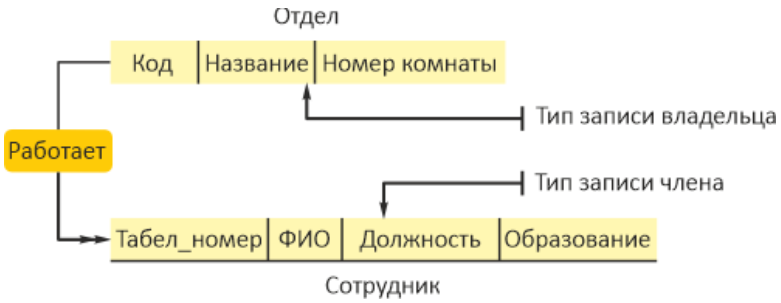


Рисунок 4.6 – Набор Работает между двумя типами записей Отдел и Сотрудник

База данных в сетевой модели данных – это поименованная совокупность экземпляров записей различного типа и экземпляров наборов, содержащих связи между ними.

Преобразование концептуальной модели в сетевую

Сетевая модель данных может быть без осложнений получена из концептуальной модели. Для этого надо предположить, что в последней используются только бинарные связи. Причем они должны принадлежать к типам "один к одному" или "один ко многим". При этом вместо сущностей концептуальной модели необходимо использовать типы записей сетевой модели, где имена сущностей становятся именами типов записей, атрибуты сущностей становятся полями записей, связь между сущностями превращается в связь между типами записей.

Бинарные связи, принадлежащие к типу "один ко многим", переносятся в сетевую модель следующим образом: тип записи со стороны "один" становится владельцем, а тип записи со стороны "много" становится типом записи-члена. Для связи типа "один к одному" выбор типа записи-владельца и типа записи-члена может быть осуществлен произвольно.

Управляющая часть сетевой модели

Для манипулирования данными в сетевой модели данных определен ряд типичных операций, которые можно подразделить на две группы: навигационные операции и операции модификации.

Навигационные операции осуществляют перемещение по БД путем прохождения по связям, определенным в схеме БД. В результате таких операций определяется запись, которая называется текущей. К подобным операциям относятся:

- найти конкретную запись в наборе однотипных записей и сделать ее текущей;
- перейти от записи-владельца к записи-члену в некотором наборе;
- перейти к следующей записи в некоторой связи;
- перейти от записи-члена к владельцу по некоторой связи.

Операции модификации осуществляют как добавление новых экземпляров отдельных типов записей, так и экземпляров новых наборов, удаление экземпляров записей и наборов, модификацию отдельных компонентов самой записи. Для реализации этих операций в системе текущее состояние детализируется путем запоминания трех его составляющих: текущего набора, текущего типа записи, текущего экземпляра типа записи. В такой ситуации возможны следующие операции:

- извлечь текущую запись в буфер прикладной программы для обработки;

- заменить в извлеченной записи значения указанных элементов данных на заданные новые их значения;
- запомнить запись из буфера в БД;
- создать новую запись;
- уничтожить запись;
- включить текущую запись в текущий экземпляр набора;
- исключить текущую запись из текущего экземпляра набора.

4.3. Иерархическая модель данных

Первые системы управления базами данных использовали иерархическую модель данных, и во времени их появление предшествует появлению сетевой модели.

Структурная часть иерархической модели

Основными информационными единицами в иерархической модели данных являются сегмент и поле. Поле данных определяется как наименьшая неделимая единица данных, доступная пользователю. Для сегмента определяются тип сегмента и экземпляр сегмента. Экземпляр сегмента образуется из конкретных значений полей данных. Тип сегмента – это поименованная совокупность входящих в него типов полей данных.

Как и сетевая, иерархическая модель данных базируется на графовой форме построения данных, и на концептуальном уровне она является просто частным случаем сетевой модели данных. В иерархической модели данных вершине графа соответствует тип сегмента или просто сегмент, а дугам – типы связей предок – потомок. В иерархических структурах сегмент – потомок должен иметь в точности одного предка.

Иерархическая модель представляет собой связный неориентированный граф древовидной структуры, объединяющий сегменты. Иерархическая БД состоит из упорядоченного набора деревьев (см. рисунок 4.7).

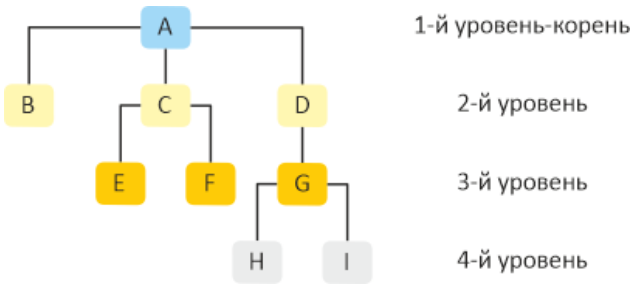


Рисунок 4.7 – Иерархическая модель данных

Преобразование концептуальной модели в иерархическую модель данных

Преобразование концептуальной модели в иерархическую структуру данных во многом схоже с преобразованием ее в сетевую модель, но и имеет некоторые отличия в связи с тем, что иерархическая модель требует организации всех данных в виде дерева.

Преобразование связи типа "один ко многим" между предком и потомком осуществляется практически автоматически в том случае, если потомок имеет одного предка, и происходит это следующим образом. Каждый объект с его атрибутами, участвующий в такой связи, становится логическим сегментом. Между двумя логическими сегментами устанавливается связь типа "один ко многим". Сегмент со стороны "много" становится потомком, а сегмент со стороны "один" становится предком.

Ситуация значительно усложняется, если потомок в связи имеет не одного, а двух и более предков. Так как подобное положение является невозможным для иерархической модели, то отражаемая структура данных нуждается в преобразованиях, которые сводятся к замене одного дерева, например, двумя (если имеется два предка). В результате такого преобразования в базе данных появляется избыточность, так как единственно возможный выход из этой ситуации – дублирование данных.

Управляющая часть иерархической модели

В рамках иерархической модели выделяют языковые средства описания данных (ЯОД) и средства манипулирования данными (ЯМД).

Каждая физическая база описывается набором операторов, обусловливающих как ее логическую структуру, так и структуру хранения БД. При этом способ доступа устанавливает способ организации взаимосвязи физических записей. Определены следующие способы доступа:

- иерархически последовательный;
- иерархически индексно-последовательный;
- иерархически прямой;
- иерархически индексно-прямой;
- индексный.

Помимо задания имени БД и способа доступа описания должны содержать определения типов сегментов, составляющих БД, в соответствии с иерархией, начиная с корневого сегмента. Каждая физическая БД содержит только один корневой сегмент, но в системе может быть несколько физических БД.

Среди операторов манипулирования данными можно выделить операторы поиска данных, операторы поиска данных с возможностью модификации, операторы модификации данных. Набор операций манипулирования данными в иерархической БД невелик, но вполне достаточен.

Примеры типичных операторов поиска данных:

- найти указанное дерево БД;
- перейти от одного дерева к другому;
- найти экземпляр сегмента, удовлетворяющий условию поиска;
- перейти от одного сегмента к другому внутри дерева;
- перейти от одного сегмента к другому в порядке обхода иерархии.

Примеры типичных операторов поиска данных с возможностью модификации:

- найти и удержать для дальнейшей модификации единственный экземпляр сегмента, удовлетворяющий условию поиска;
- найти и удержать для дальнейшей модификации следующий экземпляр сегмента с теми же условиями поиска;
- найти и удержать для дальнейшей модификации следующий экземпляр для того же родителя.

Примеры типичных операторов модификации иерархически организованных данных, которые выполняются после выполнения одного из операторов второй группы (поиска данных с возможностью модификации):

- вставить новый экземпляр сегмента в указанную позицию;
- обновить текущий экземпляр сегмента;
- удалить текущий экземпляр сегмента.

В иерархической модели автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

Тема 5. Реляционная модель данных

5.1. История вопроса

Создателем реляционной модели является сотрудник фирмы IBM доктор Э. Ф. Кодд. Будучи по образованию математиком, Э. Кодд предложил использовать для обработки данных аппарат теории множеств. В статье "A Relational Model of Data for Large Shared Data Banks", вышедшей в свет в 1970 году, он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение (relation).

Положив теорию отношений в основу реляционной модели, Э. Кодд обосновал реляционную замкнутость отношений и ряда некоторых специальных операций, которые применяются сразу ко всему множеству строк отношения, а не к отдельной строке. Указанная реляционная замкнутость означает, что результатом выполнения операций над отношениями является также отношение, над которым в свою очередь можно осуществить некоторую операцию. Из этого следует, что в данной модели можно оперировать реляционными выражениями, а не только отдельными операндами в виде простых имен таблиц.

Одним из основных преимуществ реляционной модели является ее однородность. Все данные рассматриваются как хранимые в таблицах и только в таблицах. Каждая строка такой таблицы имеет один и тот же формат.

К числу достоинств реляционного подхода можно отнести:

- наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;
- наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации БД;
- возможность не навигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

Кодд сформулировал двенадцать приведенных ниже правил, которым должна соответствовать настоящая реляционная база данных.

1. Правило информации. Вся информация в базе данных должна быть представлена исключительно на логическом уровне и только одним способом – в виде значений, содержащихся в таблицах.
2. Правило гарантированного доступа. Логический доступ ко всем и каждому элементу данных (атомарному значению) в реляционной базе данных должен обеспечиваться путем использования комбинации имени таблицы, первичного ключа и имени столбца.
3. Правило поддержки недействительных значений. В настоящей реляционной базе данных должна быть реализована поддержка недействительных значений, которые отличаются от строки символов нулевой длины, строки пробельных символов и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных.
4. Правило динамического каталога, основанного на реляционной модели. Описание базы данных на логическом уровне должно быть представлено в том же виде, что и основные данные, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка, который они применяют для работы с основными данными.
5. Правило исчерпывающего подъязыка данных. Реляционная система может поддерживать различные языки и режимы взаимодействия с пользователем (например, режим вопросов и ответов). Однако должен существовать, по крайней мере, один язык, операторы которого можно представить в виде строк символов в соответствии с некоторым четко определенным синтаксисом и который в полной мере поддерживает следующие элементы:
    1. определение данных;
    2. определение представлений;
    3. обработку данных (интерактивную и программную);
    4. условия целостности;
    5. идентификацию прав доступа;
    6. границы транзакций (начало, завершение и отмена).
6. Правило обновления представлений. Все представления, которые теоретически можно обновить, должны быть доступны для обновления.
7. Правило добавления, обновления и удаления. Возможность работать с отношением как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных.
8. Правило независимости физических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться

нетронутыми при любых изменениях способов хранения данных или методов доступа к ним.

9. Правило независимости логических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при внесении в базовые таблицы любых изменений, которые теоретически позволяют сохранить нетронутыми содержащиеся в этих таблицах данные.
10. Правило независимости условий целостности. Должна существовать возможность определять условия целостности, специфические для конкретной реляционной базы данных, на подъязыке реляционной базы данных и хранить их в каталоге, а не в прикладной программе.
11. Правило независимости распространения. Реляционная СУБД не должна зависеть от потребностей конкретного клиента.
12. Правило единственности. Если в реляционной системе есть низкоуровневый язык (обрабатывающий одну запись за один раз), то должна отсутствовать возможность использования его для того, чтобы обойти правила и условия целостности, выраженные на реляционном языке высокого уровня (обрабатывающем несколько записей за один раз).

5.2. Структурная часть реляционной модели

Отношение

Реляционная база данных – это конечный (ограниченный) набор отношений. Отношения используются для представления сущностей, а также для представления связей между сущностями. Отношение – это двумерная таблица, имеющая уникальное имя и состоящая из строк и столбцов, где строки соответствуют записям, а столбцы – атрибутам. Каждая строка в таблице представляет некоторый объект реального мира или соотношения между объектами.

Атрибут – это поименованный столбец отношения. Свойства сущности, его характеристики определяются значениями атрибутов. Порядок следования атрибутов не влияет на само отношение.

Пусть имеется отношение r. Схемой отношения r называется конечное множество имен атрибутов $R = (A_1, A_2, \ldots A_n)$. Заголовки столбцов отношения содержат имена его атрибутов и, следовательно, все вместе отражают его схему.

Схема отношения **ПРЕПОДАВАТЕЛЬ** может быть представлена следующим образом: (**Таб_ном_преп, Фамилия, Должность**)

Или

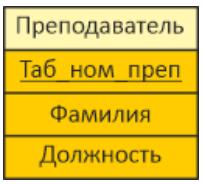


Рисунок 5.1 – Схема отношения ПРЕПОДАВАТЕЛЬ

Тогда заголовок отношения **ПРЕПОДАВАТЕЛЬ** примет вид

Рисунок 5.2 – Заголовок отношения ПРЕПОДАВАТЕЛЬ

Отношение строится с учетом ряда факторов. Каждому имени атрибута A_i, 1 >= i <= n ставится в соответствие множество допустимых для соответствующего столбца значений. Это множество D_i, называется доменом данного имени атрибута.

Каждая строка отношения является множеством значений, взятых по одному из домена каждого имени атрибута. Домены являются произвольными непустыми конечными или счетными множествами и образуют множество:

$$D = D_1 \cup D_2 \ldots \cup D_n$$

Отношение r со схемой R – это конечное множество отображений (t_1, t_2, ... t_p) из R в D. Причем каждое отображение $t \in r$ должно удовлетворять следующему ограничению:

$t(A_i)$ принадлежит D_i, где 1 <= i <= n.

Эти отображения называются кортежами. Каждый кортеж отношения отображает экземпляр сущности, а атрибут отношения отображает атрибут сущности.

Множество кортежей называется телом отношения. Тело отношения отражает состояние сущности, поэтому во времени оно постоянно меняется. Тело отношения характеризуется кардинальным числом, которое равно количеству содержащихся в нем кортежей.

Одной из главных характеристик отношения является его степень. Степень отношения определяется количеством атрибутов, которое в нем присутствует. Эта характеристика отношения имеет еще названия: ранг и арность. Отношение с одним атрибутом называется унарным, с двумя атрибутами – бинарным, с тремя – тернарным, с n атрибутами n-арным. Определение степени отношения осуществляется по заголовку отношения.

Все названные характеристики отношения обозначены на рисунке 5.3.

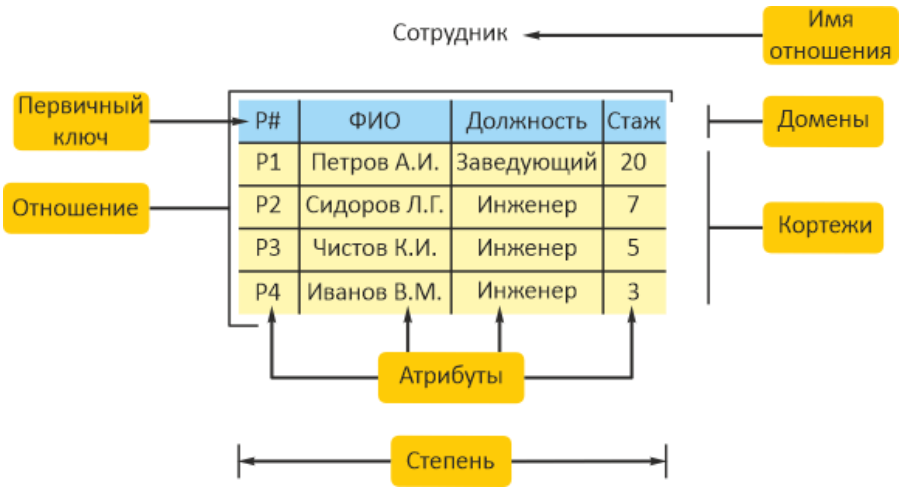


Рисунок 5.3 – Отношение СОТРУДНИК

Свойства и виды отношений

Отношение по структуре подобно таблице, но таблице, обладающей определенными свойствами. Сведем воедино все свойства отношения.

- Отношение имеет имя, которое отличается от имен всех других отношений.
- Отношение представляется в виде табличной структуры.
- Каждый атрибут имеет уникальное имя, его значения берутся из одного и того же домена.
- Каждый компонент кортежа является простым, атомарным значением, не состоящим из группы значений.
- Упорядочение атрибутов теоретически несущественно, однако оно может влиять на эффективность доступа к кортежам.
- Все строки (кортежи) должны быть различны.
- Теоретически порядок следования кортежей не имеет значения.

В реляционной теории встречается несколько видов отношений, но не все они поддерживаются реальными системами. Различают:

- именованное отношение – это переменная отношения, определенная в СУБД посредством специальных операторов;
- базовое отношение – это именованное отношение, являющееся частью базы данных;
- производное отношение – это отношение, определенное посредством реляционного выражения через базовые отношения;
- представление – это именованное виртуальное производное отношение, представленное в системе исключительно через определение в терминах других именованных отношений;

- снимки – это отношения, подобные представлениям, но они сохраняются, доступны для чтения и периодически обновляются;
- результат запроса – это неименованное производное отношение, получаемое в результате запроса, которое для сохранения необходимо преобразовать в именованное отношение;
- хранимое отношение – это отношение, которое поддерживается в физической памяти.

Реляционные ключи

В отношении могут существовать несколько одиночных или составных атрибутов, которые однозначно идентифицируют кортеж отношения. Это – потенциальные ключи.

Говорят, что множество атрибутов K = (A_i, A_j, ..., A_k) отношения r является потенциальным ключом r тогда и только тогда, когда удовлетворяются два независимых от времени условия:

- уникальность: в произвольный заданный момент времени никакие два различных кортежа r не имеют одного и того же значения для A_i, A_j, ..., A_k;
- минимальность: ни один из атрибутов A_i, A_j, ..., A_k не может быть исключен из K без нарушения уникальности.

Отношение может иметь несколько потенциальных ключей. Ключ, содержащий два и более атрибута, называется составным ключом. Каждое отношение обладает хотя бы одним возможным ключом, поскольку в отношении не может быть одинаковых кортежей, а это значит, что, по меньшей мере, комбинация всех его атрибутов удовлетворяет условию уникальности. Потенциальные ключи, позволяя гарантированно выделить точно один кортеж, обеспечивают основной механизм адресации на уровне кортежей реляционной модели.

Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Обычно первичным ключом назначается тот возможный ключ, которым проще всего пользоваться при повседневной работе. Остальные возможные ключи, если они есть, называются альтернативными ключами. Для индикации связи между отношениями используются внешние ключи.

Внешний ключ – это набор атрибутов одного отношения, являющийся потенциальным ключом другого отношения.

Благодаря наличию связок между потенциальными и внешними ключами обеспечивается взаимосвязь кортежей определенных отношений. Отношение,

содержащее внешний ключ, называется дочерним или ссылающимся отношением. А отношение, содержащее связанный с внешним ключом потенциальный ключ, – родительским или целевым отношением.

Отношения не могут рассматриваться как статические объекты, так как они предназначены для отражения некоторой части реального мира, а эта часть реального мира может изменяться во времени. Поэтому и отношения изменяются во времени: кортежи могут добавляться, удаляться или модифицироваться. Тем не менее, предполагается, что сама схема отношения инвариантна во времени. Отношение должно восприниматься как множество возможных состояний, которые может принимать отношение.

Пример

Пусть рассматривается концептуальная модель, приведенная на рисунке 5.4. Пример относится к предметной области, которую можно назвать "Преподавательская деятельность". Данная модель содержит две сущности: **ЛЕКТОР** и **ПРЕДМЕТ**, между которыми установлена связь **ЧИТАЕТ** типа "многие ко многим". Характеристики сущностей представлены изображенными на рисунке атрибутами. Связь **ЧИТАЕТ** не имеет собственных атрибутов. Для преобразования концептуальной модели в реляционную модель разработан ряд технологий, знакомство с которыми состоится несколько позже.

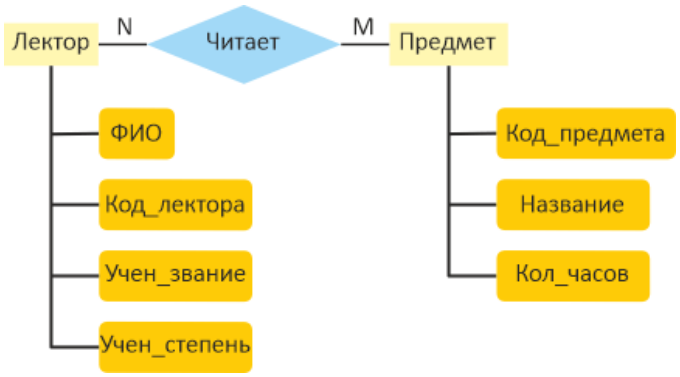


Рисунок 5.4 – Концептуальная модель для примера

В данный момент, не вдаваясь в подробности причин принятого решения, просто приведем реляционную схему, соответствующую указанной концептуальной модели. Она включает в себя три отношения: **ЛЕКТОР**, **ПРЕДМЕТ**, **ЧИТАЕТ**. Схемы отношений и связи между ними изображены на рисунке 5.5.

Рисунок 5.5 – Реляционная схема базы данных для примера

На рисунке 5.6 даны соответствующие отношения, заполненные кортежами. Эти отношения имеют следующие характеристики:

- **ЛЕКТОР** – 4-нарное отношение с первичным ключом Код_лек с кардинальным числом, равным четырем; атрибуты определены на следующих доменах: **Код_лек** – (целые: 1...4), **ФИО** – (возможные фамилии и инициалы), **Уч_степень** – (к.т.н., д.т.н., нет степени), **Уч_звание** – (Доцент, Профессор, Нет_звания);
- **ПРЕДМЕТ** – тернарное отношение с первичным ключом **Код_предм**. с кардинальным числом, равным шести; атрибуты определены на следующих доменах: **Код_предм** – (символьный). **Назв_предм** – (Информатика, Программирование, Физика, ООП, Базы данных, Базы данных), **Кол_во_час** – (целые: 54, 102, 36);
- **ЧИТАЕТ** – бинарное отношение с составным первичным ключом **Код_лек, Код_предм**, с кардинальным числом, равным шести, в котором присутствуют первичные ключи только читающих лекторов и первичные ключи только читаемых предметов.

Лектор

Код_лек	ФИО	Уч_степень	Уч_звание
01	Петров А.И.	К.т.н	Доцент
02	Сидоров Л.Г.	К.т.н	Доцент
03	Чистов К.И.	Д.т.н	Профессор
04	Иванов В.М.	К.т.н	Доцент

Читает

Код_предм	Код_лек
01	02
02	02
03	04
04	01
05	03
06	01

Предмет

Код_предм	Назв_предм	Кол_во_час
01	Информатика	54
02	Базы данных	102
03	ООП	34
04	Программир-е	45
05	Физика	45
06	Структуры	45

Рисунок 5.6 – Реляционные отношения модели для примера

Все приведенные отношения являются нормализованными, поскольку атрибуты всех отношений имеют атомарные значения. Во всех трех отношениях отсутствуют дублирующие кортежи.

5.3. Обновление отношений

Для обновления отношений необходимо иметь возможность выполнять следующие операции;

- добавление кортежа;
- удаление кортежа;
- изменение кортежа.

Операция добавления для отношения r со схемой (A_1, A_2, ..., A_n) имеет вид:

ADD (r: $A_1=d_1$, $A_2=d_2$, ..., $A_n=d_n$).

Если порядок атрибутов фиксирован, возможна более короткая запись:

ADD (r: d_1, d_2, ..., d_n).

Выполнение этой операции может стать невозможным в ряде случаев:

- добавляемый кортеж не соответствует схеме определенного отношения;
- некоторые значения кортежей не принадлежат соответствующим доменам;
- описанный кортеж совпадает по ключу с кортежем, уже находящимся в отношении.

Операция удаления предназначена для удаления кортежей. Она может быть записана следующим образом:

DEL (r: $A_1=d_1$, $A_2=d_2$, ..., $A_n=d_n$).

или для упорядоченных атрибутов:

DEL (r: d_1, d_2, ..., d_n).

Для удаления некоторого кортежа часто достаточно указать значение некоторого ключа:

DEL (r: ключ).

Если указанный кортеж в отношении отсутствует, то отношение остается неизменным.

Операция изменения предназначена для модификации части кортежа. Для отношения r ее можно при $(C_1, C_2, ..., C_p) \in (A_1, A_2, ..., C_n)$ определить так:

CH (r: $A_1=d_1$, $A_2=d_2$, ..., $A_n=d_n$; $C_1=e_1$, $C_2=e_2$, ..., $C_n=e_p$)

Если K = (B_1, B_2, ..., B_k) является ключом, то запись данной операции может быть сокращена:

$$СН\ (r: B_1=d_1, B_2=d_2, ..., B_k=d_k; C_1=e_1, C_2=e_2, ..., C_n=e_p)$$

Возможные ошибки в данном случае те же, что и у предыдущих операций:

- указанный кортеж не существует;
- изменения имеют неправильный формат;
- используемые значения не принадлежат соответствующим доменам.

5.4. Целостность базы данных

Поддержка целостности базы данных реализуется посредством ряда ограничений, накладываемых на данные.

Первый тип ограничений проистекает из того факта, что каждый атрибут определяется на своем домене, или наоборот: домен атрибута задает множество значений, которые может принимать атрибут. Указанное ограничение называется ограничением атрибута.

Важными понятиями в теории реляционных баз данных являются категорная целостность и целостность на уровне ссылок.

Категорная целостность ограничивает набор значений первичных ключей базовых отношений. Такого рода целостность заключается в следующем: кортеж не может записываться в БД до тех пор, пока значения его ключевых атрибутов не будут полностью определены. Иными словами: никакой ключевой атрибут любого кортежа отношения не может содержать отсутствующего значения, обозначаемого определителем NULL.

Целостность на уровне ссылок. При построении отношений для связывания строк одной таблицы со строками другой таблицы используются внешние ключи. База данных, в которой все непустые внешние ключи ссылаются на текущие значения ключей другого отношения, обладает целостностью на уровне ссылок.

Как правило, определение условия целостности данных осуществляется при установлении взаимосвязи между родительским и дочерним отношениями; при этом пользователю часто предоставляется возможность самому решить, каким путем сохранять целостность базы данных и насколько жестко должно соблюдаться условие целостности при различных операциях изменения данных.

Так, например, при изменении значения первичного ключа в родительском отношении часто разрешены следующие варианты действий, из которых два первых – обеспечивают нахождение базы данных в целостном непротиворечивом состоянии.

- При изменении значений полей первичного ключа в родительском отношении автоматически осуществляется каскадное изменение всех соответствующих значений в дочернем отношении.
- Не позволяется изменять значения полей первичного ключа в родительском отношении, если в дочернем отношении имеется хотя бы одна запись, содержащая ссылку на изменяемую запись
- Позволяется изменять значения полей первичного ключа в родительском отношении, независимо от существования связанных записей в дочернем отношении. Целостность данных при этом не поддерживается.

При удалении записи в родительском отношении также возможны несколько вариантов действий.

- При удалении записи в родительском отношении автоматически осуществляется каскадное удаление всех записей из дочернего отношения, связанных с удаляемой записью.
- Не позволяется удалять записи в родительском отношении, если в дочернем отношении имеется хотя бы одна запись, содержащая ссылку на удаляемую запись. При попытке удаления записи возникает ошибка, которую можно обработать программно.
- Позволяется удалять записи в родительском отношении независимо от существования связанных записей в дочернем отношении. Очевидно, что целостность данных при этом не поддерживается.

При добавлении новой записи в дочернее отношение или редактировании в нем существующей записи возможно выбрать один из вариантов:

- не позволяется вводить запись, если значение индексного выражения дочернего отношения не соответствует одной из записей в родительском отношении;
- при вводе данных в дочернее отношение не анализируется значение индексного выражения. Целостность данных при этом не поддерживается.

Еще один вид ограничений, накладываемый на информацию, содержащуюся в отношениях, связан с теми бизнес-правилами, которые существуют в данной организации или в моделируемой предметной области. Это так называемые дополнительные правила поддержки целостности данных, определяемые пользователем или администратором данных, которые называются корпоративными ограничениями целостности. Приведем примеры таких правил:

- студентам нельзя планировать занятия продолжительностью более восьми часов в день;
- студенты весь день должны заниматься в одном корпусе вуза, чтобы не требовалось дополнительное время на переходы из одного здания в другое.

Глоссарий

А

Автоматизированная информационная система (АИС) – это система, реализующая автоматизированный сбор, обработку, манипулирование данными, функционирующая на основе ЭВМ и других технических средств и включающая соответствующее программное обеспечение (ПО) и персонал. В дальнейшем в этом качестве будет использоваться термин информационная система (ИС), который подразумевает понятие автоматизированная.

Администратор БД (АБД) – человек или группа лиц, которые принимают решения.

Атрибут – это поименованная характеристика сущности, с помощью которой моделируется ее свойство. Каждой сущности присущи свои атрибуты.

Б

База данных (БД) – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД – это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

Банк данных (БнД) – это система специальным образом организованных данных: баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

В

Внешний ключ – это набор атрибутов одного отношения, являющийся потенциальным ключом другого отношения.

Г

Глобальная логическая модель данных – это логическая модель, отражающая особенности представления о функционировании предприятия одновременно многих типов пользователей.

Д

Данные – это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные – это информация в дискретном, фиксированном виде, удобная для хранения, обработки на ЭВМ, а также для передачи по каналам связи.

Двухуровневая модель – это модель при которой все пять компонентов приложения распределяются только между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере.

Домен – это множество значений, включающего все потенциальные значения, которые могут быть присвоены атрибуту. Это множество значений называется .

З

Задачи обработки данных – это специальный класс решаемых на ЭВМ задач, связанных с видом, хранением, сортировкой, отбором по заданному условию и группировкой записей однородной структуры.

Запрос – это команда, которую вы даете вашей СУБД и которая сообщает ей чтобы она вывела определенную информацию из таблиц в память.

И

Интегрированные данные – это возможность представить базу данных как объединение нескольких отдельных файлов данных.

Информация – это любые сведения о каком-либо событии, процессе, объекте.

К

Клиенты – это различные приложения, которые выполняются над СУБД.

Ключи – это такие атрибуты, с помощью которых можно идентифицировать экземпляр сущности.

Концептуальная схема – это единое логическое описание всех элементов данных и отношений между ними, логическая структура всей базы данных. Для каждой базы данных имеется только одна концептуальная схема.

М

Модель данных – это формализованное описание структуры единиц информации и операций над ними в информационной системе

Модель удаленного управления данными – это модель файлового сервера .

Мощность – это максимальное количество экземпляров одной сущности, связанных с одним экземпляром другой сущности.

Н

Набор – это поименованная двухуровневая иерархическая структура, которая содержит запись владельца и записи членов.

О

Обслуживающий персонал (ОП) – это лица, прямыми обязанностями которых является создание и поддержание корректного функционирования банка данных.

Общие данные – это возможность использования отдельных областей данных в БД несколькими различными пользователями для разных целей.

П

Подтип – это сущность, являющаяся членом супертипа, но выполняющая отдельную роль в нем.

Показатель кардинальности – это показатель количество возможных связей для каждого экземпляра участвующего в связи сущностию.

С

Сервер – это собственно СУБД. Он поддерживает все основные функции СУБД и предоставляет полную поддержку на внешнем, концептуальном и внутреннем уровнях.

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Степень связи – это количество участников некоторой связи.

Супертип – это сущность, включающая разные подтипы, которые необходимо представить в модели данных.

Сущность – это то, о чем накапливается информация в информационной системе и что может быть однозначно идентифицировано.

Схема – это описание структуры данных на любом уровне.

Т

Технические средства – это совокупность устройств (изделий), обеспечивающих получение, ввод, подготовку, преобразование, обработку, хранение, регистрацию, вывод, отображение, использование и передачу данных, выработку и реализацию управляющих воздействий.

Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого базы данных.

Триггер – это особый тип хранимой процедуры, реагирующий на возникновение определенного события в БД.

Э

Элемент данных – это наименьшая поименованная информационная единица данных, доступная пользователю (аналог – поле в файловой системе). Элемент данных должен иметь свой тип (не структурный, простой).

Я

Язык запросов – это часть непроцедурного языка ЯМД, которая отвечает за извлечение данных.

Язык определения данных – описательный язык, с помощью которого описывается предметная область: именуются объекты, определяются их свойства и связи между объектами. Он используется главным образом для определения логической структуры БД.