ТЕОРЕТИЧЕСКИЕ ОСНОВЫ БАЗ ДАННЫХ

Учебно-методическое пособие

Оглавление

1. Введение в базы данных

1.1 Основные понятия и определения

Одним из важнейших понятий в теории баз данных является понятие *информации.* Под *информацией* понимаются 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. 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.

Под *задачами обработки данных* обычно понимается 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 Модели представления данных

Модель – способ структурирования данных, описания взаимосвязей между данными.

Очевидные требования к модели:

• Модель должна быть достаточно универсальной, позволяя работать с данными различной структуры и сложности.

• Модель должна допускать автоматическую обработку данных, т.е. должна быть реализуема программными средствами.

• Модель должна быть наглядной, «прозрачной». Поскольку задача описания структуры данных средствами выбранной модели возлагается на разработчика (человека), чем сложнее модель – тем труднее избежать ошибок при проектировании.

Ниже перечислены основные разновидности моделей представления данных, используемых или использовавшихся в прошлом.

Иерархическая модель. Модель (как видно из названия) представляет данные в виде иерархии (рис. 1.1). Модель ориентирована на описание объектов, находящихся между собой в отношении подчинения. Например, структура кадров некоторой организации. Организация состоит из отделов, каждый отдел имеет руководителя и сотрудников. Другой пример: объект «колесо» является составной частью объекта «автомобиль». Между автомобилем и колесом имеется связь, смысл которой можно озвучить следующим образом: объект «автомобиль» включает в себя несколько объектов «колесо».

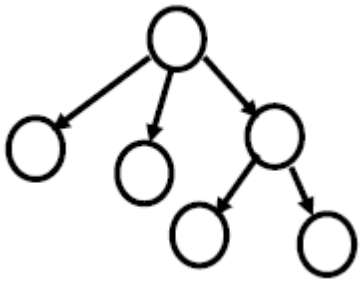


Рисунок 1.1 – Иерархическая модель

Сетевая модель. Сетевая модель (рис. 1.2) представляет собой развитие иерархической. Модель позволяет описывать более сложные виды взаимоотношений между данными. Однако расширение возможностей достигается за счет большей сложности реализации самой модели и трудности манипулирования данными.

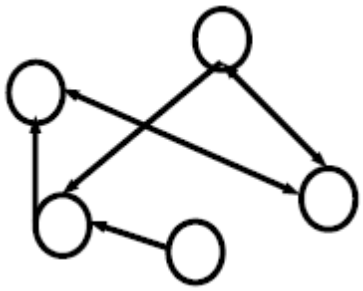


Рисунок 1.2 – Сетевая модель

Реляционная модель. В реляционной модели (рис. 1.3) данные представляются в виде таблиц, состоящих из строк и столбцов. Каждая строка таблицы – информация об одном конкретном объекте, столбцы содержат свойства этого объекта. Взаимоотношения между объектами задаются с помощью связей между столбцами таблиц. Реляционная модель на сегодняшний день наиболее распространена. Она достаточно универсальна и проста в проектировании.

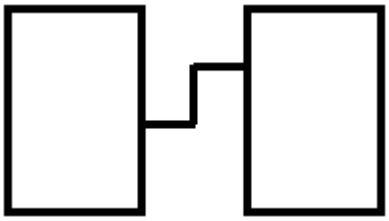


Рисунок 1.3 – Реляционная модель

Объектная модель. В этой модели данные представляются в форме объектов. Объект имеет набор свойств, называемых атрибутами, и может включать в себя также процедуры для обработки данных, которые называют методами.

Объекты, имеющие одинаковые наборы атрибутов и различающиеся только их значениями, образуют некоторый класс объектов. Например, класс «клиент» может иметь следующие атрибуты: «фамилия», «имя», «отчество», «номер кредитной карты». Для каждого объекта из этого класса определены конкретные значения перечисленных атрибутов. Говорят, что объект является экземпляром класса. На основе существующего класса могут создаваться новые, наследующие свойства исходного. При этом исходный класс именуется родителем нового класса. Производный класс называют потомком исходного. При этом объекты – экземпляры класса-потомка принадлежат также и родительскому классу, поскольку обладают всеми его атрибутами. Пример: на основе класса «клиент» может быть определен класс «постоянный клиент». Новый класс наследует атрибуты родительского и при этом добавляется новый атрибут – «процент скидки». Все постоянные клиенты продолжают оставаться клиентами в прежнем понимании, но о них имеется еще и дополнительная информация. В настоящее время не существует единого подхода к реализации объектных баз данных.

Объектный подход – набор общих принципов, которые могут применяться при проектировании различных приложений.

Гибридные модели. В некоторых приложениях предпринимаются попытки смешения различных моделей представления данных. Пример такого смешения – объектно-реляционная модель. В ней использовано некоторое сходство между реляционной и объектной идеологией. Строки таблиц реляционной модели соответствуют объектам объектной модели, столбцы таблиц – атрибутам объектов. Таблицы в целом являются аналогом классов. Отсюда вытекает возможность введения наследования при определении таблиц – таблица-потомок содержит те же столбцы, что и родительская, и, кроме того – дополнительные, определенные при наследовании. По идее создателей, объектно-реляционная модель должна унаследовать от реляционной легкость описания и манипулирования данными, а от объектной – возможность определения более сложных взаимоотношений между объектами.

1.3 Реляционные базы данных

Базы данных, хранение информации в которых основано на реляционной модели, называют **реляционными базами данных**. Как было сказано ранее, реляционная модель предполагает организацию данных в виде таблиц. Строки таблиц называют **записями или кортежами**, столбцы – **полями или атрибутами.**

Клиент

Id кл	Фамилия	Имя	Отчество
15	Иванов	Иван	Иванович
16	Петров	Петр	Петрович
17	Николаев	Николай	Николаевич

Товар

Id тов	Название
1	Шкаф
2	Стул
3	Стол

Заказ

Id зак	Клиент	Товар	Дата	Количество
1	15	1	15.09.2003	1
2	17	1	17.09.2003	2
3	15	2	20.09.2003	12

Рис. 1.4. Таблицы в реляционной модели

В примере (рис. 1.4) показано, как могут быть представлены данные и связи между ними в БД некоторой торговой организации. Таблица «Клиент» содержит данные о каждом клиенте – фамилию, имя и отчество. В таблице «Товар» хранятся сведения об имеющихся товарах. Когда клиент вступает во взаимоотношения с некоторым товаром (делает заказ), этот факт фиксируется в таблице «Заказ». Для того чтобы можно было сослаться на

отдельную запись (строку) в некоторой таблице, каждая запись этой таблицы должна содержать уникальный идентификатор. В данном примере роль таких идентификаторов выполняют поля «Id _кл», «Id_тов» и «Id_зак». Поле таблицы, значения которого гарантированно уникальны для каждой записи этой таблицы, называют **ключевым полем** или **ключом**. Ключ не обязательно должен быть числовым. Иногда уникальным идентификатором может служить не одно поле, а комбинация полей. При этом сочетание значений этих полей должно быть уникальным. Такие поля образуют **составной ключ** таблицы.

Вернемся к примеру. Обратим внимание, что поля «Клиент» и «Товар» содержат значения, совпадающие со значениями ключей таблиц «Клиент» и «Товар» (совпадение имен полей и названий таблиц не является обязательным).

Каждая запись таблицы «Заказ» содержит информацию о заказе клиентом некоторого товара – дату заказа и количество единиц товара. В заказе также содержатся идентификаторы клиента и заказа, что позволяет однозначно определить – кем и на какой товар сделан заказ. Так, клиент Иванов заказал один шкаф, а позднее – двенадцать стульев. Клиент Николаев заказал только один шкаф, а Петров пока не заказал ничего.

Для изображения таблиц и связей между ними обычно применяется более компактная форма, в которой указываются имена полей, без данных. На рис. 1.5 приведена структура рассмотренной ранее базы данных.

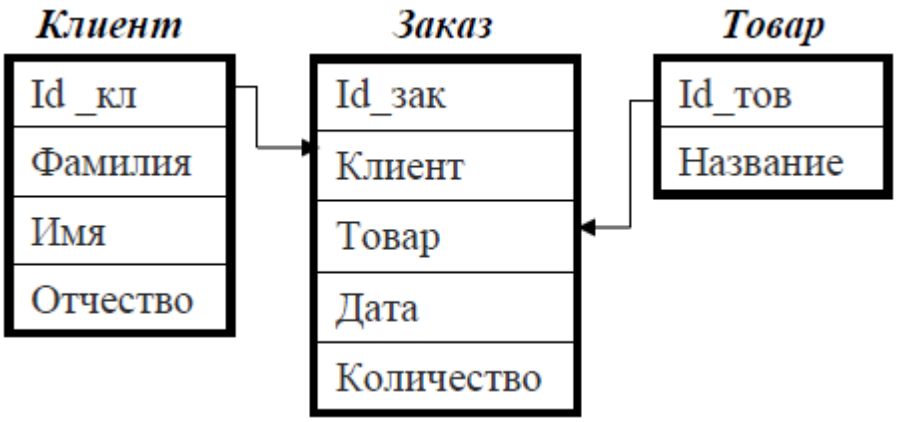


Рис. 1.5. Изображение таблиц и связей

Стрелки указывают «направление», в котором используются ключевые поля. Значения из поля «Id _кл» помещаются в поле «Клиент», а значения из поля «Id_тов» - в поле «Товар». Очевидно, что одному клиенту может соответствовать несколько заказов, но каждому заказу соответствует только один клиент. С товарами – аналогичная ситуация. Один товар может входить в несколько заказов, но в каждый заказ входит только один товар.

Вероятно, оформление заказов было бы более удобным, если бы в одном заказе можно было указывать сразу несколько товаров. Для этого структуру данных следует изменить.

На рис. 1.6 приведена новая структура БД. Таблица «Заказ» содержит для каждого заказа его уникальный идентификатор, идентификатор клиента, сделавшего этот заказ и дату. Перечень и количество товаров, входящих в заказ, хранится в таблице «Состав заказа». Одной записи в таблице «Заказ» может соответствовать несколько записей в таблице «Состав заказа».

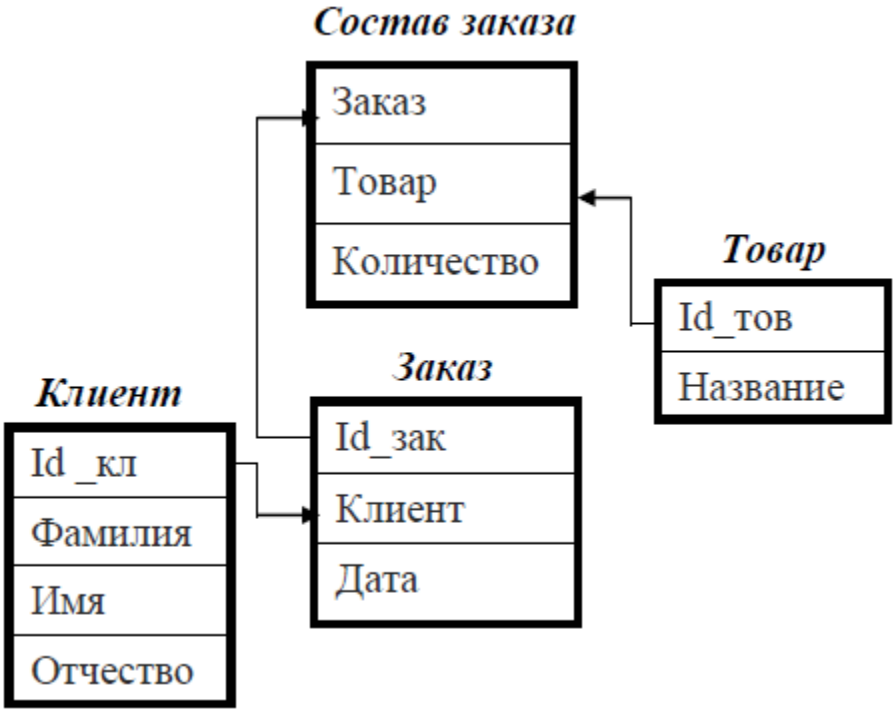


Рисунок 1.6 – Преобразованная структура БД

1.4. 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:

- одну или несколько БД;
- СУБД;
- словарь или каталог данных;
- администратора (АБД);
- вычислительную систему (ВС), включающую аппаратные (АС) и программные (ПС) средства;
- обслуживающий персонал (ОП).

Схематично это выглядит так, как показано на рис. 1.7.

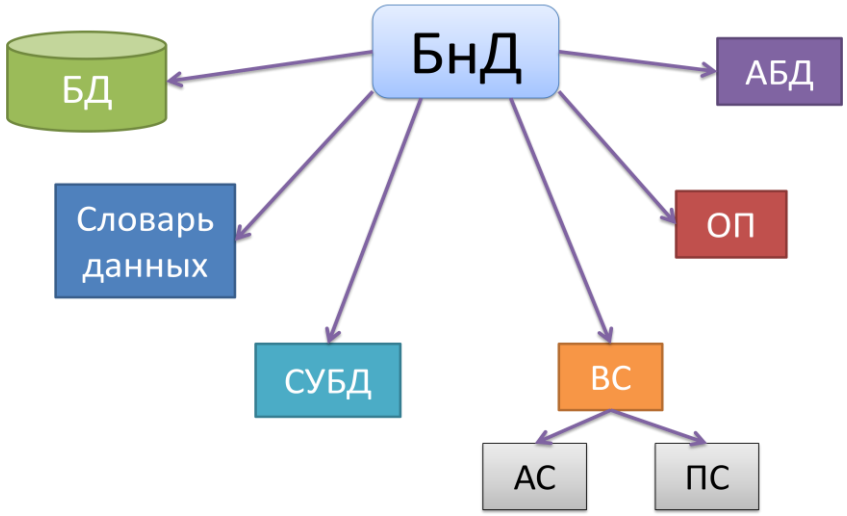


Рис. 1.7 - Банк данных

Дадим краткие определения новым составляющим этой схемы.

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.

Администратор БД (АБД) — человек или группа лиц, которые принимают решения. Основные функции АБД:

- участие в разработке БД;
- контроль правильности функционирования БД.

Вычислительная система (ВС) — включает программные (ПС) и аппаратные средства (АС).

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.

1.5. Базы данных

БД, как правило, создается как общий ресурс всего предприятия, где данные являются *интегрированными и общими.* Под понятием *интегрированные* данные подразумевается возможность представить базу данных как объединение нескольких отдельных файлов данных. Под понятием *общие* данные подразумевается 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 рис. 1.8. На нем показан пример схемы базы данных, которую можно назвать ФАКУЛЬТЕТ.

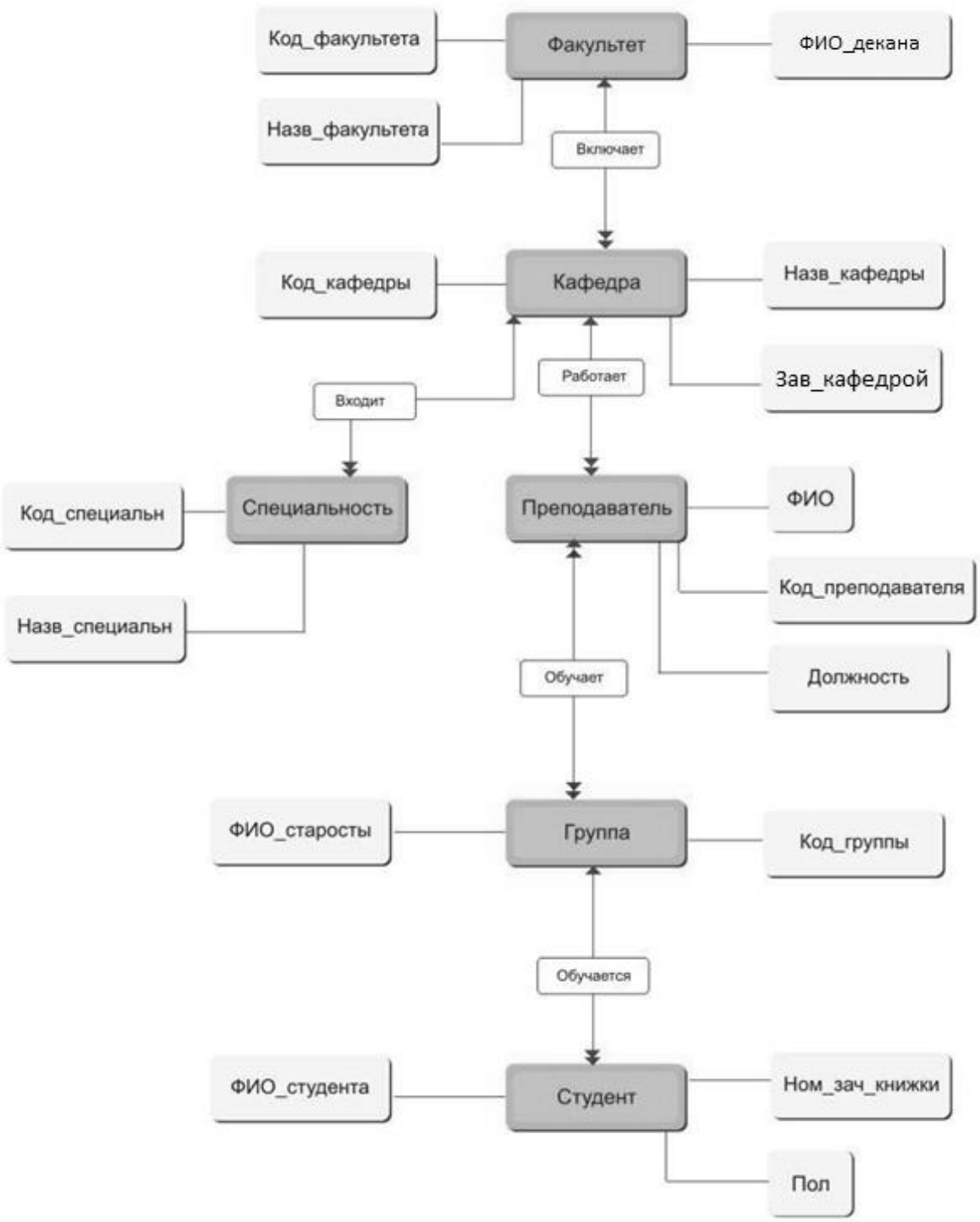


Рис. 1.8 - Пример ER-диаграммы базы данных ФАКУЛЬТЕТ

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

- шести сущностей, которые изображены темными прямоугольниками, каждый из которых имеет свои атрибуты, помещенные в светлые прямоугольники, а в нижеприведенном списке они перечислены в скобках рядом с именем сущностей:

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:

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.

1.6. 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 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.

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 (рис. 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.

Описание структуры данных на любом уровне называется *схемой.* 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.*

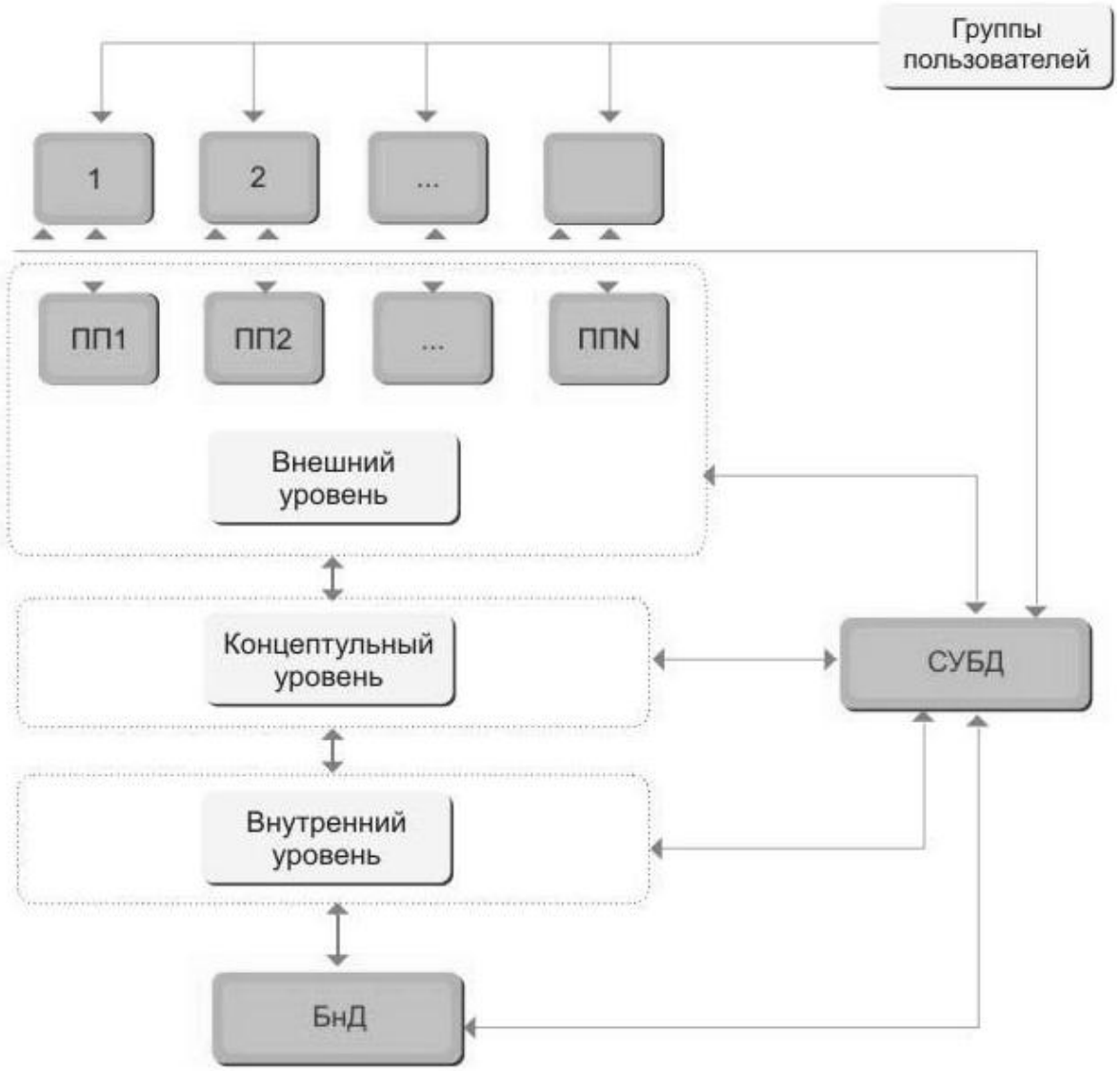


Рис. 2.1. Трехуровневая архитектура СУБД

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. Таким образом, *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.

Внутренняя схема описывает физическую реализацию базы данных и предназначена для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. Для каждой базы данных существует только одна внутренняя схема.

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

1) 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.

2) 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.

3) 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.

4) 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.

5) 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.

6) 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.

7) 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.

8) 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.

9) 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).

2.3.1 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.

2.3.2 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

Часть непроцедурного языка ЯМД, которая отвечает за извлечение данных, называется *языком запросов* Язык запросов можно определить как высокоуровневый узкоспециализированный язык, предназначенный для удовлетворения различных требований по выборке информации из базы данных.

2.4. Архитектура многопользовательских СУБД

2.4.1. 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 (рис. 2.2).

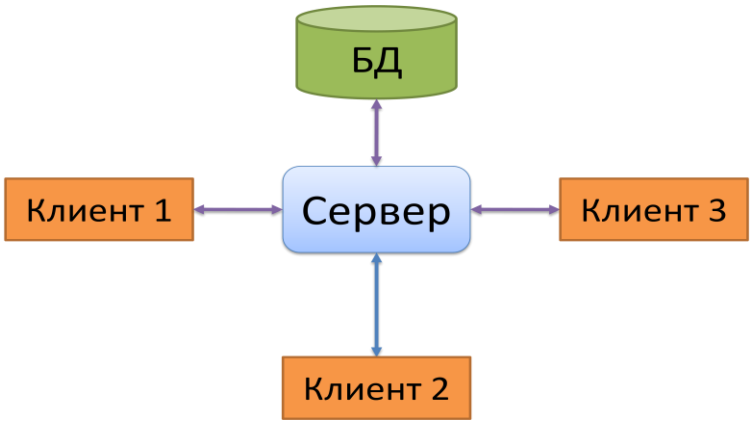


Рис. 2.2. Структура системы БД с выделением клиентов и сервера

Сервер в простейшем случае — 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.

Если все пять компонентов приложения распределяются только между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере, то такая модель называется *двухуровневой.* Она имеет несколько основных разновидностей. Рассмотрим их.

Файловый сервер

Модель файлового сервера называется моделью *удаленного управления данными.* 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 (рис. 2.3).

Файл-сервер

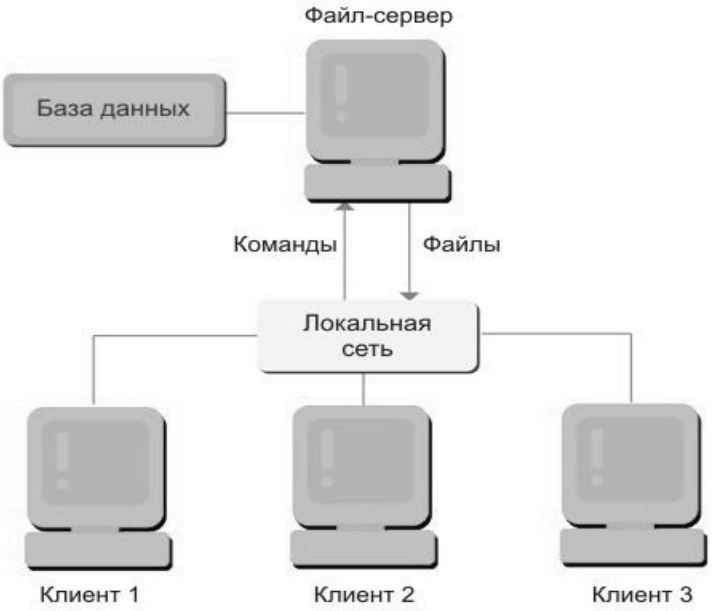


Рис. 2.3. Модель файлового сервера

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 рис. 2.4.

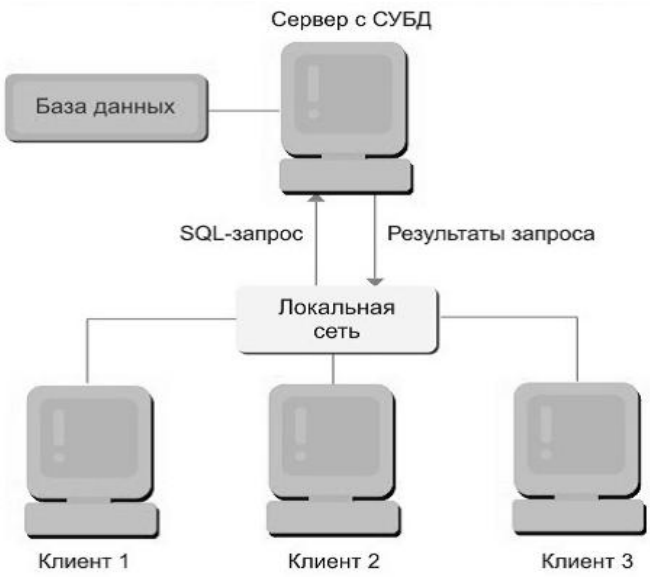


Рис. 2.4. Модель удаленного доступа

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 (рис. 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.

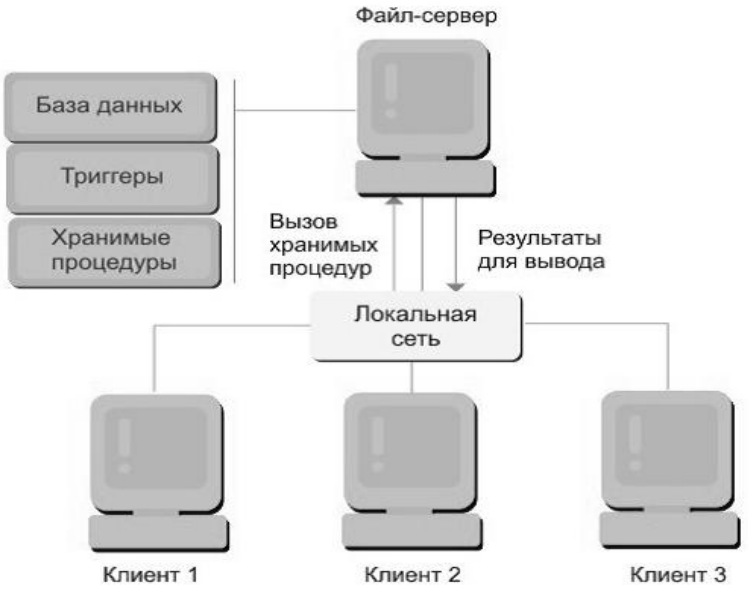


Рис. 2.5. Модель сервера БД

2.4.2. 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 рис. 2.6.

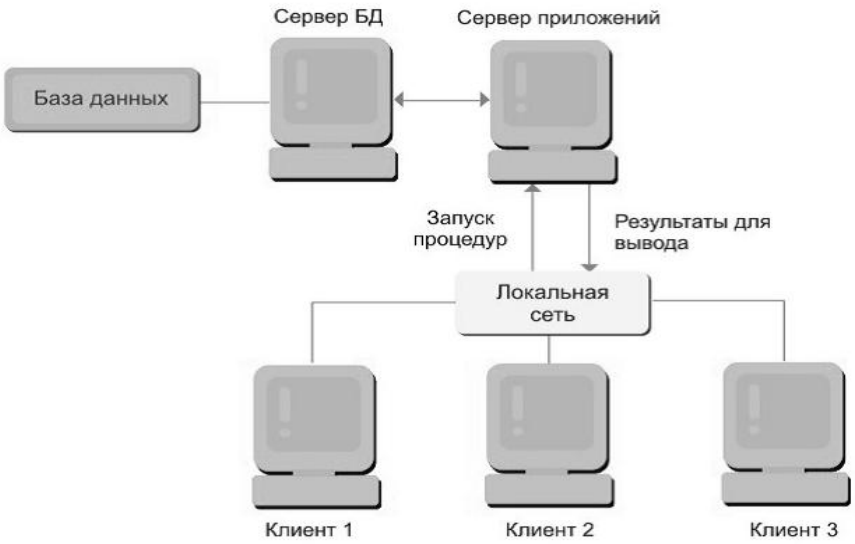


Рис. 2.6. Архитектура трехуровневой модели

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.

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 (рис. 3.1):

1. планирование разработки базы данных;
2. определение требований к системе;
3. сбор и анализ требований пользователей;
4. проектирование базы данных:
   - системный анализ предметной области;
   - инфологическое проектирование;
   - выбор СУБД;
   - даталогическое проектирование;
   - физическое проектирование.
5. разработка приложений:
   - проектирование транзакций;
   - проектирование пользовательского интерфейса;
6. реализация;
7. загрузка данных;
8. тестирование;
9. эксплуатация и сопровождение:

3.2 Системный анализ предметной области

С точки зрения проектирования БД в рамках системного анализа, необходимо осуществить первый этап, то есть провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Желательно, чтобы данное описание позволяло корректно определить все взаимосвязи между объектами предметной области.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

Функциональный подход - он реализует принцип движения «от задач» и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае мы можем четко выделить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

Предметный подход - когда информационные потребности будущих пользователей БД жестко не фиксируются. Они могут быть многоаспектными и весьма динамичными. Мы не можем точно выделить минимальный набор объектов предметной области, которые необходимо описывать. В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее. БД, конструируемая при этом, называется предметной, то есть она может быть использована при решении множества разнообразных, заранее не определенных задач. Конструирование предметной БД в некотором смысле кажется гораздо более заманчивым, однако трудность всеобщего охвата предметной области с невозможностью конкретизации потребностей пользователей может привести к избыточно сложной схеме БД, которая для конкретных задач будет неэффективной.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД.

3.3 Инфологическое (семантическое) моделирование предметной области

Инфологическое моделирование (иногда используется термин семантическое моделирование) применяется на втором этапе проектирования БД, то есть после *системного анализа предметной области.* На этапе системного анализа были сформированы понятия о предметах, фактах и событиях, которыми будет оперировать БД. Инфологическое проектирование связано с представлением семантики предметной области в модели БД, т.е. моделирование структур данных, опираясь на смысл этих данных. Инфологическое моделирование было предметом исследований в конце

1970-х и начале 1980-х годов. Было предложено несколько моделей данных, названных семантическими моделями. Наибольшее распространение получила модель "сущность-связь" (entity-relationship model, ER-модель), предложенная в 1976 г. Питером Пин-Шэн Ченом.

Модель «сущность-связь» является *концептуальной моделью,* т.е. не учитывает особенности конкретной СУБД. Из модели "сущность-связь" могут быть получены все основные *фактографические модели данных* (иерархическая, сетевая, реляционная, объектно-ориентированная).

Модели "сущность-связь" удобны тем, что процесс создания модели является итерационным. Разработав первый приближенный вариант модели, можно уточнять ее, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, является сама модель "сущность-связь".

3.3.1 Модель «сущность-связь»

Основными понятиями модели "сущность-связь" являются: сущность, связь и атрибут. Любой фрагмент предметной области может быть представлен как *множество сущностей,* между которыми существует некоторое *множество связей.*

Сущность - это реальный или представляемый объект, информация о котором должна сохраняться в проектируемой системе.

Сущность имеет имя, уникальное в пределах системы. Сущность соответствует некоторому классу однотипных объектов, то есть в системе существует множество экземпляров данной сущности.

Примеры сущностей: люди, продукты, студенты и т.д.

Примеры экземпляров сущностей: конкретный человек, конкретный продукт, конкретный студент и т. д.

Сущности не обязательно должны быть непересекающимися. Например, экземпляр сущности СТУДЕНТ, также принадлежит сущности ЛЮДИ.

Объект, которому соответствует понятие сущности, имеет свой набор ***атрибутов*** -характеристик, определяющих свойства данного объекта. Атрибут должен иметь имя, уникальное в пределах данной сущности.

Пример:

Рассмотрим множество пищевых продуктов, имеющихся в магазине. Каждый продукт можно представить следующими характеристиками: Код продукта, Продукт, Срок хранения, Условия хранения.

В дальнейшем для определения сущности и ее атрибутов будем использовать обозначение вида:

Продукты(КодПродукта, Продукт, ЕдиницаИзмерения, СрокХранения, УсловияХранения).

Например, поставщиков, поставляющих продукты в магазин, можно описать как:

Поставщики(КодПоставщика, Поставщик, Адрес)

Множество допустимых значений (область определения) атрибута называется ***доменом.***

Например, атрибут *СрокХранения* хранит информацию о количестве дней, в течение которых продукт годен к употреблению. То есть этот атрибут принадлежит домену *КоличествоДней,* который задается интервалом целых чисел больших нуля, поскольку количество дней отрицательным быть не может.

Набор атрибутов сущности должен быть таким, чтобы можно было однозначно найти требуемый экземпляр сущности.

Например, сущность *Продукты* однозначно определяется атрибутом *КодПродукта,* поскольку все коды продуктов различны.

Отсюда определяется ***ключ сущности*** - это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся.

Пример. Сущность *Продажа* с атрибутами *ДатаПродажи, КодПродукта, Количество, Цена* содержит информацию о продаже продуктов за конкретный день. Для этой сущности ключом будут атрибуты *ДатаПродажи* и *КодПродукта,* поскольку за день могут быть проданы несколько продуктов, а конкретный продукт может быть продан в разные дни. Исключение любого атрибута из ключа не позволит однозначно найти требуемый экземпляр сущности, т.е. будет нарушено условие минимальности. Обозначим эту сущность как:

Продажа(ДатаПродажи. КодПродукта. Количество, Цена)

Ключевой атрибут выделен подчеркиванием

Между сущностями могут быть установлены связи.

Связь - это ассоциация. установленная между несколькими сущностями. и показывающая как взаимодействуют сущности между собой.

Примеры:

- в магазине происходит продажа продуктов. т.е. между сущностями *Продукты* и *Продажа* существует связь *«происходит»* или *Продукты-Продажа* (обычно связь обозначается глаголом или двойным названием сущностей. между которыми установлена эта связь)
- так как продукты в магазин поставляют поставщики, то между сущностями *Продуты* и *Поставщики* существует связь *«поставка»* или *Продукты-Поставщики*
- могут существовать и связи между экземплярами одной и той же сущности (рекурсивная связь). например связь *Родитель-Потомок* между экземплярами сущности *Человек*

Связь также может иметь атрибуты. Например. для связи *Продукты-Поставщики* можно задать атрибуты *ДатаПоставки. Цена* и т.д.

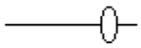
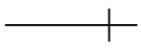
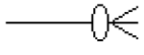
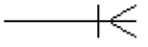
Связь. существующая между двумя сущностями. называется ***бинарной связью.*** Связь. существующая между n сущностями. называется ***n-арной связью. Рекурсивная связь*** - это связь между экземплярами одной сущности.

Доказано. что любую n-арную связь всегда можно заменить множеством бинарных. однако первые лучше отображают семантику предметной области.

То число экземпляров сущностей. которое может быть ассоциировано через связь с экземплярами другой сущности. называют ***степенью связи.*** Рассмотрение степеней особенно полезно для бинарных связей.

Очень важным свойством модели "сущность-связь" является то. что она может быть представлена в виде графической схемы (диаграммы). Это значительно облегчает анализ предметной области. Существует несколько вариантов обозначения элементов диаграммы "сущность-связь" (нотаций). Для обозначения сущностей. связей и атрибутов будем использовать нотацию Чена. а для обозначения степеней и кардинальностей связей - нотацию Мартина (понятие кардинальности связи поясним чуть позже). В Таблице 1 приводится список используемых здесь обозначений.

Таблица 1

Обозначение	Пояснение
Имя сущности	Независимая сущность
Имя сущности	Зависимая сущность
имя атрибута	Атрибут
имя атрибута	Ключевой атрибут
Имя связи	Связь
	Связь степени 1, необязательный класс принадлежности
	Связь степени 1, обязательный класс принадлежности
	Связь степени N, необязательный класс принадлежности
	Связь степени N, обязательный класс принадлежности
	Связь от зависимой к независимой сущности

3.3.2 Степени бинарных связей

• **один-к-одному** (обозначается 1:1). Это означает, что в такой связи в каждый момент времени каждому экземпляру сущности A соответствует 1 или 0 экземпляров сущности B. Данный факт представлен на Рис. 3.1, где прямоугольники обозначают сущности, а ромб -связь. Так как степень связи для каждой сущности равна 1, то они соединяются одной линией.

Рисунок 3.1 – Связь один-к-одному

один-ко-многим (1:N). Одному экземпляру сущности A соответствуют 0, 1 или N экземпляров сущности B. Графически степень связи N отображается "древообразной" линией, так это сделано на следующем рисунке (Рис. 3.2).

Рисунок 3.2 – Связь один-ко-многим

многие-к-одному (N:1). Эта связь аналогична отображению 1:N. Одному экземпляру сущности B соответствуют 0, 1 или N экземпляров сущности A (Рис. 3.3).

Рисунок 3.3 – Связь многие-к-одному

многие-ко-многим (M:N). В этом случае одному экземпляру сущности A соответствуют 0, 1 или N экземпляров сущности B, и наоборот, одному экземпляру сущности B соответствуют 0, 1 или N экземпляров сущности A (Рис. 3.4).

Рисунок 3.4 – Связь многие-ко-многим

Другой важной характеристикой связи помимо ее степени является ***класс принадлежности*** входящих в нее сущностей. Существует *обязательный* и *необязательный* классы принадлежности.

Рассмотрим *примеры* степени связи один-к-одному:

1. Если рассматривать **традиционный брак,** то степень связи между сущностями *Мужчина* и *Женщина* будет 1:1. Кроме того, каждый мужчина, состоящий в браке, обязательно ДОЛЖЕН иметь жену. Таким образом, говорят, что сущность *Женщина* имеет *обязательный класс принадлежности.* И, наоборот, каждая женщина, состоящая в браке, ДОЛЖНА иметь мужа. То есть, сущность *Мужчина* также имеет *обязательный класс принадлежности* (Рис. 3.5).

Рисунок 3.5 – Связь 1:1, обязательные классы принадлежности

2. **Сотрудник руководит отделом** (Рис. 3.6). Поскольку сотрудник может руководить только ОДНИМ отделом, а в отделе может быть только ОДИН руководитель, то степень связи в этом примере 1:1. Кроме того, в каждом отделе ДОЛЖЕН быть руководитель, т.е. каждому экземпляру сущности *Отдел* ДОЛЖЕН соответствовать экземпляр сущности *Сотрудник* (сущность *Сотрудник* имеет обязательный класс принадлежности). С другой стороны, далеко не все сотрудники должны быть руководителями, т.е. сотрудник МОЖЕТ (но не должен) быть руководителем. Таким образом, есть экземпляры сущности *Сотрудник,* которым не соответствует ни один экземпляр сущности *Отдел* (необязательный класс принадлежности).

Рис. 3.6 - Связь 1:1, обязательный и необязательный классы принадлежности

3. **Человек читает книгу** (Рис. 3.7). Человек может читать сразу только ОДНУ книгу, а конкретная книга может быть читаема только ОДНИМ человеком, следовательно, степень связи 1:1. Человек МОЖЕТ читать книгу, а может ничего не читать. С другой стороны не каждая книга должна читаться, некоторые стоят на полке. Таким образом, обе сущности имеют необязательный класс принадлежности.

Рис. 3.7 - Связь 1:1, необязательные классы принадлежности

Рассмотрим *примеры* степени связи один-ко-многим (многие-к-одному):

4. В процессе обучения студенты объединены в группы. Каждая группа может содержать множество студентов, а каждый студент может входить только в одну группу, т.е. степень связи 1:N (Рис. 3.8). Каждая группа ДОЛЖНА содержать студентов, а каждый студент ДОЛЖЕН быть зачислен в конкретную группу, т.е. обе сущности имеют обязательные классы принадлежности

Рис. 3.8 - Связь 1:N, обязательные классы принадлежности

5. Поставщики продуктов имеют один юридический адрес, следовательно, ДОЛЖНЫ находиться в одном конкретном городе. А в одном городе МОГУТ находиться один, несколько или ни одного поставщика. Т.е. связь будет N:1, сущность Города будет иметь обязательный, а сущность Поставщики - необязательный классы принадлежности (Рис. 3.9).

Рис. 3.9 - Связь 1:N, обязательный и необязательный классы принадлежности

Если существование сущности *x* зависит от существования сущности *y,* то *x* называется **зависимой сущностью** (иногда сущность *x* называют "слабой", а "сущность" *y - сильной).*

Степень связи для сильной сущности всегда будет 1 и обязательный класс принадлежности. Класс принадлежности и степень связи для зависимой сущности могут быть любыми.

6. В магазине происходит продажа продуктов. Продукт не может быть продан, если его нет в магазине. Поэтому сущность *Продажи* является зависимой от сущности *Продукты* (Рис. 3.10). Продукт МОЖЕТ быть продан в разные дни (а может быть вообще не продан), конкретная продажа связана только с одним продуктом. Таким образом, степень связи N:1, сущность *Продажи* имеет необязательный, а сущность *Продукты* - обязательный классы принадлежности (в самом деле, продажа без продукта теряет смысл)

Рис. 3.10 – Зависимая и независимая сущности

Рассмотрим *пример* степени связи многие-ко-многим:

7. Продукты в магазин поставляются поставщиками. Каждый продукт, имеющийся в магазине, ДОЛЖЕН быть поставлен одним или несколькими поставщиками, а каждый из поставщиков МОЖЕТ поставлять один или несколько продуктов или не поставлять ни одного. Т.е. степень связи M:N (Рис. 3.11), а класс принадлежности для *Поставщиков* - обязательный, для *Продуктов* - необязательный.

Рис. 3.11 - Связь M:N, обязательный и необязательный классы принадлежности

8. Между одними и теми же сущностями могут существовать несколько связей, например: С одной стороны продукты в магазин поставляются заказчиками, с другой стороны, чтобы продукты были поставлены в магазин, необходимо заказать поставщикам необходимые продукты. Таким образом, между сущностями *Продукты* и *Поставщики* существуют связи *«Поставляют»* и *«Заказаны»* (Рис. 3.12). Связь *«Поставляют»* рассмотрена в предыдущем примере. Рассмотрим подробнее связь *«Заказаны».* Каждый продукт ДОЛЖЕН быть заказан одному или нескольким поставщикам, каждый поставщик МОЖЕТ получить заказ на один или несколько продуктов или вообще не получить заказ.

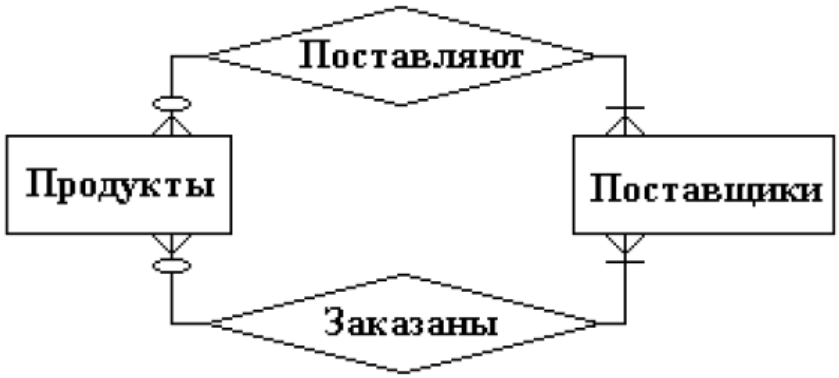


Рис. 3.12 – Несколько связей между двумя сущностями

9. Рассмотрим сущности *Врач* и *Пациент.* Пациент ДОЖЕН иметь одного лечащего врача, а врач МОЖЕТ лечить несколько пациентов. Кроме того, пациент МОЖЕТ иметь нескольких врачей-консультантов, а врач МОЖЕТ консультировать нескольких пациентов (Рис.3.13).

Рис. 3.13 – Несколько связей между двумя сущностями

3.3.3 Пример построения модели «сущность-связь»

В процессе построения диаграммы "сущность-связь" можно выделить несколько этапов:

- Определение списка сущностей выбранной предметной области
- Определение списка атрибутов сущностей
- Описание связей между сущностями (степени, классы принадлежности связей, а также атрибуты связей, если они необходимы)
- Организация данных в виде диаграммы "сущность-связь"

В качестве примера построим диаграмму, отображающую связь данных для информационной системы учета продажи продуктов в магазине. БД

должна хранить информацию о продуктах, поставляемых в магазин, их ежедневной продаже, заказах на поставку продуктов, а также о поставщиках продуктов.

Составим список сущностей, необходимых для реализации поставленной задачи:

1. Продукты

Для этой сущности необходимы следующие атрибуты:

- *Код продукта* - уникальный идентификатор, ключевой атрибут
- *Продукт* - название продукта
- *Единица измерения* - литры, килограммы, штуки и т.п.
- *Срок хранения в днях* - для определения даты окончания срока годности продукта
- *Условия хранения* - температура, влажность и т.п.

2.Поставщики

- *Код поставщика* - уникальный идентификатор, ключевой атрибут
- *Поставщик* - название организации или ФИО физического лица
- *Код города* - выделим отдельно город, где находится поставщик, для удобства дальнейшей
- работы (например, для поиска)
- *Адрес* - поскольку город выделен в отдельный атрибут, то в адресе остается улица и дом (а
- также квартира - для физического лица)
- ФИО директора
- Телефон
- Факс

3.Продажи

- Дата продажи
- Код продукта - какой именно продукт был продан
- Количество - сколько продано этого продукта в тех единицах измерения, которые указаны
- для этого продукта в сущности Продукт
- Цена продажи - цена при продаже за единицу продукта

4. Города - поскольку мы выделили отдельно город из адреса поставщика, то возникает необходимость в этой сущности

- *Код города* - уникальный идентификатор, ключевой атрибут
- *Город*

Сократив для удобства названия атрибутов, получим список сущностей:

- Продукты(КодПрод, Продукт, ЕдИзм, СрокХран(дней), УсловияХран)
- Поставщики(КодПост, Поставщик, КодГорода, Адрес, ФИОдиректора, Телефон, Факс)
- Продажи(ДатаПродажи, КодПрод, Количество, ЦенаПродажи),

обратите внимание, что в этой сущности ключ составной, поскольку каждый день продается множество продуктов, и конкретный продукт может быть продан в разные дни

- Города(КодГорода, Город)

Рассмотрим связи, существующие между описанными выше сущностями:

1. Продукты в магазин поставляются поставщиками, т.е. существует связь M:N **«Поставляют»** между сущностями Продукты и Поставщики (Рис. 3.11). Эта связь имеет следующие атрибуты:

- Дата поставки
- Код поставщика - какой поставщик поставил этот продукт
- Код продукта - какой именно продукт был поставлен
- КоличествоП - сколько поставлено этого продукта в тех единицах измерения, которые
- указаны для этого продукта в сущности Продукт
- Цена поставки - цена при поставке за единицу продукта
- Дата изготовления - дата изготовления продукта

Ключом будет составной атрибут: *Дата поставки, Код поставщика, Код продукта*.

2. Продукты должны быть заказаны поставщикам, т.е. существует связь M:N «Заказаны» между сущностями Продукты и Поставщики (Рис. 3.12). Эта связь имеет следующие атрибуты:

- Дата заказа
- Код поставщика - какому поставщику заказан этот продукт
- Код продукта - какой именно продукт был заказан
- КоличествоЗ - сколько поставлено этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности Продукт.

Ключом будет составной атрибут: *Дата заказа, Код поставщика, Код продукта*

3. В магазине происходит продажа продуктов, т.е. существует связь N:1 **«Происходит»** между сущностями Продажи и Продукты (Рис. 3.10)

4. Поставщики находятся в определенном городе, т.е. существует связь N:1 «Находятся» между сущностями Поставщики и Города (Рис. 3.9)

После объединения всех фрагментов в общую модель и добавления атрибутов, получится диаграмма "сущность-связь", приведенная на Рис. 3.14.

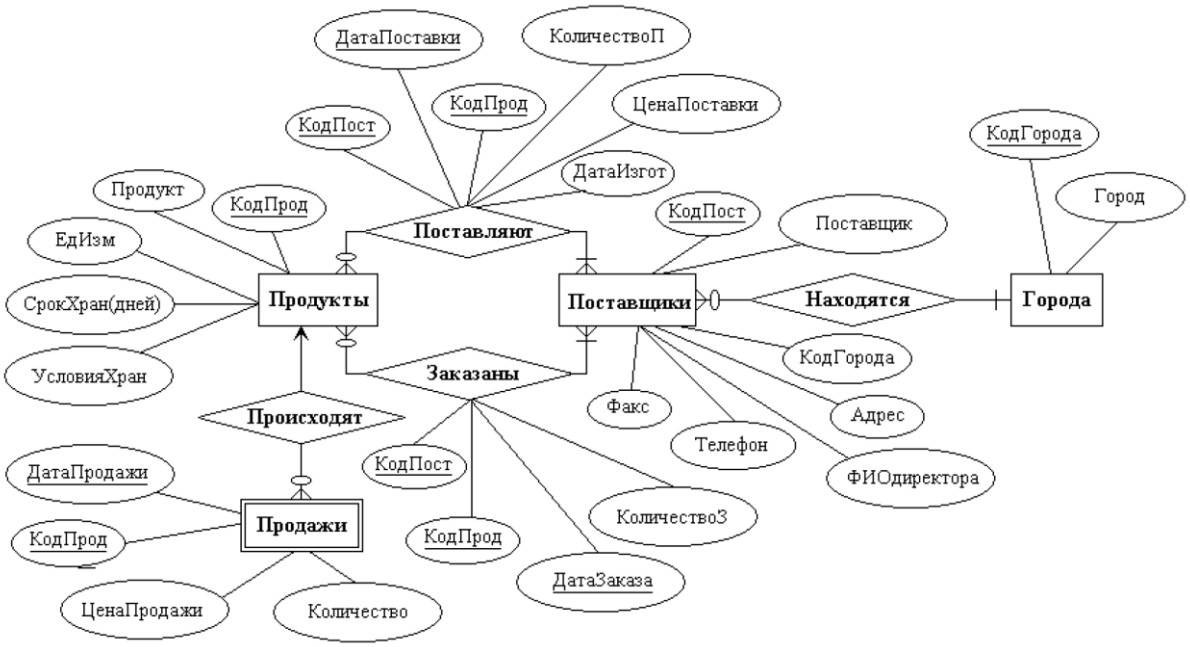


Рис. 3.14. Диаграмма «сущность-связь» учета продажи продуктов в магазине

3.4 Аномалии

3.4.1 Избыточность данных и аномалии обновления

Основная цель проектирования реляционной базы данных заключается в группировании атрибутов в отношения таким образом, чтобы минимизировать избыточность данных и тем самым сократить объем памяти, необходимый для физического хранения отношений, представленных в виде таблиц. Проблемы, связанные с избыточностью данных, можно проиллюстрировать, сравнив отношения Сотрудник и Отдел в таблицах 2 и 3 с отношением СотрудникОтдел таблице 4. Отношение СотрудникОтдел является альтернативной формой представления отношений Сотрудник и Отдел. Упомянутые отношения описываются следующим образом:

- Сотрудник (**СотрудникNo**, СотрИмя, Должность, Зарплата, ОтделNo)
- Отдел (**ОтделNo**, ОтделАдрес)
- СотрудникОтдел (**СотрудникNo**, СотрИмя, Должность, Зарплата, ОтделNo, ОтделАдрес)

Обратите внимание, что здесь первичный ключ каждого отношения выделен жирным начертанием.

Таблица 2

СотрудникNo	**СотрИмя**	**Должность**	**Зарплата**	**ОтделNo**
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Таблица 3

ОтделNo	**ОтделАдрес**
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Таблица 4

СотрудникNo	**СотрИмя**	**Должность**	**Зарплата**	**ОтделNo**	**ОтделАдрес**
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

В отношении СотрудникОтдел содержатся избыточные данные, поскольку сведения об отделении компании повторяются в записях, относящихся к каждому сотруднику данного отделения. В противоположность этому в отношении Отдел сведения об отделении содержатся только в одной строке, а в отношении Сотрудник повторяется только номер отделения компании (ОтделNo), который представляет собой место работы каждого сотрудника. При работе с отношениями, содержащими избыточные данные, могут возникать проблемы, которые называются аномалиями обновления и подразделяются на аномалии вставки, удаления и модификации.

3.4.2 Аномалии вставки

Существуют два основных типа аномалий вставки, которые иллюстрируются с помощью отношения СотрудникОтдел (см, таблицу 4).

- При вставке сведений о новых сотрудниках в отношение Сотрудник Отдел. необходимо указать и сведения об отделении компании, в котором эти сотрудники работают. Например, при вставке сведений о новом сотруднике отделения 'B007' требуется ввести сведения о самом отделении 'B007', которые должны соответствовать сведениям об этом же отделении в других строках отношения СотрудникОтдел. Отношения, показанные в табл. 2 и 3, не подвержены влиянию этой потенциальной несовместимости данных, поскольку для каждого сотрудника в отношение Сотрудник потребуется ввести только соответствующий номер отделения компании. Кроме того,

сведения об отделении компании с номером 'BOO7' заносятся в базу данных однократно, в виде единственной строки отношения Отдел.

• Для вставки сведений о новом отделении компании, которое еще не имеет собственных сотрудников, требуется присвоить значение NULL всем атрибутам описания персонала отношения СотрудникОтдел, включая и табельный номер сотрудника СотрудникNo. Но поскольку атрибут СотрудникNo является первичным ключом отношения СотрудникОтдел, то попытка ввести значение NULL в атрибут СотрудникNo вызовет нарушение целостности сущностей и потому будет отклонена. Следовательно, в отношение СотрудникОтдел невозможно ввести строку о новом отделении компании, содержащую значение NULL в атрибуте СотрудникNo. Структура отношений, представленных в табл. 1 и 2, позволяет избежать возникновения этой проблемы, поскольку сведения об отделениях компании вводятся в отношение Отдел независимости ввода сведений о сотрудниках. Сведения о сотрудниках, которые будут работать в новом отделении компании, могут быть введены в отношение Сотрудник позже.

3.4.3 Аномалии удаления

При удалении из отношения Сотрудник Отдел строки с информацией о последнем сотруднике некоторого отделения компании сведения об этом отделении будут полностью удалены из базы данных. Например, после удаления из отношения Сотрудник Отдел строки для сотрудника 'Mary Howe с табельным номером ' SA9' из базы данных неявно будут удалены все сведения об отделении с номером B007 . Однако структура отношений, показанных в табл. 2 и 3, позволяет избежать возникновения этой проблемы, поскольку строки со сведениями об отделениях компании хранятся отдельно от строк со сведениями о сотрудниках. Связывает эти два отношения только общий атрибут ОтделNo. При удалении из отношения Сотрудник строки с номером сотрудника ' SA9' сведения об отделении 'BO07' в отношении Отдел останутся нетронутыми.

3.4.4 Аномалии обновления

При попытке изменения значения одного из атрибутов для некоторого отделения компании в отношении Сотрудник Отдел (например, адреса отделения 'BOO3') необходимо обновить соответствующие значения в строках для всех сотрудников этого отделения. Если такой модификации будут подвергнуты не все требуемые строки отношения Сотрудник Отдел., база данных будет содержать противоречивые сведения. В частности, в нашем примере для отделения компании с номером 'BOO3' в строках, относящихся к разным сотрудникам, ошибочно могут быть указаны разные значения адреса этого отделения.

Все приведенные выше примеры иллюстрируют то, что представленные в табл. 2 и 3 отношения Сотрудник и Отдел обладают более приемлемыми свойствами, чем отношение Сотрудник Отдел, представленное в табл. 4. Это

доказывает, что отношение Сотрудник Отдел подвержено аномалиям обновления, но этих аномалий можно избежать путем декомпозиции первоначального отношения на отношения Сотрудник и Отдел. С декомпозицией крупного отношения на более мелкие связаны два важных свойства. Во-первых, свойство соединения без потерь гарантирует, что любой экземпляр первоначального отношения может быть определен с помощью соответствующих экземпляров более мелких отношений. Во-вторых, свойство сохранения зависимостей гарантирует, что ограничения на первоначальное отношение можно поддерживать, просто применяя такие же ограничения к каждому из более мелких отношений. Иными словами, для проверки того, не нарушается ли ограничение, которое распространялось на первоначальное отношение, нет необходимости выполнять операции соединения на более мелких отношениях.

3.5 Нормализация и нормальные формы

Нормализация - требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами.

Процесс нормализации был впервые предложен Э. Ф. Коддом. Нормализация часто выполняется в виде последовательности тестов с целью проверки соответствия (или несоответствия) некоторого отношения требованиям заданной нормальной формы. Сначала были предложены только три вида нормальных форм: первая (1НФ), вторая (2НФ) и третья (ЗНФ). Затем Р. Бойсом и Э. Ф. Коддом было сформулировано более строгое определение третьей нормальной формы, которое получило название нормальной формы Бойса-Кодда (НФБК). Все эти нормальные формы основаны на функциональных зависимостях, существующих между атрибутами отношения, Нормальные формы более высокого порядка, которые превосходят НФБК, были введены позднее. К ним относятся четвертая (4НФ) и пятая (5НФ) нормальные формы. Но на практике эти нормальные формы более высоких порядков используются крайне редко.

Отношение было представлено как состоящее из некоторого количества атрибутов, а реляционная схема — из некоторого количества отношений. Атрибуты могут группироваться в отношения с образованием реляционной схемы на основе либо собственного опыта разработчика базы данных, либо посредством вывода реляционной схемы из разработанной ER-диаграммы. При использовании любого из этих двух подходов часто требуется применять определенный формальный метод, способный помочь проектировщику базы данных найти оптимальную группировку атрибутов для каждого отношения в схеме.

Процесс нормализации является формальным методом, позволяющим определять отношения на основе их первичных или потенциальных ключей и функциональных зависимостей, существующих между их атрибутами. Проектировщики баз данных могут использовать нормализацию в виде

наборов тестов, применяемых к отдельным отношениям с целью нормализации реляционной схемы до заданной конкретной формы, что позволит предотвратить возможное возникновение аномалий обновления.

Нормализация — это формальный метод анализа отношений на основе их первичного ключа (или потенциальных ключей) и существующих функциональных зависимостей. Он включает ряд правил, которые могут использоваться для проверки отдельных отношений таким образом, чтобы вся база данных могла быть нормализована до желаемой степени. Если некоторое требование не удовлетворяется, то противоречащее данному требованию отношение должно быть разделено на отношения, каждое из которых (в отдельности) удовлетворяет всем требованиям нормализации.

Чаще всего нормализация осуществляется в виде нескольких последовательно выполняемых этапов, каждый из которых соответствует определенной нормальной форме, обладающей известными свойствами. В ходе нормализации формат отношений становится все более ограниченным (строгим) и менее восприимчивым к аномалиям обновления.

Рисунок 3.15 - Связь нормальных форм

При работе с реляционной моделью данных важно понимать, что для создания отношений приемлемого качества обязательно только выполнение требований первой нормальной формы (1НФ). Все остальные формы могут использоваться по желанию проектировщиков. Но для того чтобы избежать аномалий обновления, нормализацию рекомендуется выполнять как минимум до третьей нормальной формы (ЗНФ). На рисунке показана взаимосвязь между различными нормальными формами. Согласно этому рисунку, некоторые отношения в форме 1НФ могут находиться также в форме ЗНФ, отношения 2НФ — в форме ЗНФ и т.д.

Предположим, что задано множество функциональных зависимостей для каждого от этого рисунка. Схема взаимосвязей между отдельными нормальными формами ношения и что в каждом отношении имеется назначенный первичный ключ. Эта информация имеет исключительно важное значение для нормализации и служит для проверки того, находится ли отношение в определенной нормальной форме.

3.5.1 Первая нормальная форма (1НФ)

Ненормализованная форма (ННФ) - таблица, содержащая одну или несколько повторяющихся групп данных.

Первая нормальная форма (1НФ) - отношение, в котором на пересечении каждой строки и каждого столбца содержится одно и только одно значение.

Для преобразования ненормализованной таблицы в первую нормальную форму (1НФ) в исходной таблице следует найти и устранить все повторяющиеся группы данных. Повторяющейся группой называется группа, состоящая из одного или нескольких атрибутов таблицы, в которой возможно наличие нескольких значений для единственного значения ключевого атрибута (атрибутов) таблицы. Существуют два способа исключения повторяющихся групп из ненормализованных таблиц.

1. Повторяющиеся группы устраняются путем ввода соответствующих данных в пустые столбцы строк с повторяющимися данными, т.е. пустые места заполняются дубликатами неповторяющихся данных (этот способ частно называют "выравниванием" таблицы). Полученная таблица, теперь называемая отношением, содержит элементарные (или единственные) значения на пересечении каждой строки с каждым столбцом и поэтому находится в 1НФ. Однако, полученное отношение имеет определенную избыточность данных, устраняемую в ходе дальнейшей нормализации.

2. Один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы изымаются и помещаются в отдельные отношения вместе с копиями ключа исходной таблицы, в новых отношениях устанавливаются свои первичные ключи. При наличии в ненормализованной таблице нескольких повторяющихся групп или повторяющихся групп, содержащихся в других повторяющихся группах, данный прием применяется до тех пор, пока повторяющихся групп совсем не останется. Полученный набор отношений будет находиться в первой нормальной форме только тогда, когда ни в одном из них не будет повторяющихся групп атрибутов. Данный способ находится в 1НФ и обладает меньшей избыточностью данных, чем первый способ.

При использовании первого подхода выровненное отношение 1НФ раскладывается в ходе дальнейшей нормализации на те же отношения, которые могли быть сразу же получены с помощью второго подхода.

Пример приведения таблицы к первой нормальной форме

Исходная, ненормализованная, таблица:

Сотрудник	**Номер телефона**
Иванов И. И.	283-56-82 390-57-34
Петров П. Ю.	708-62-34

Таблица, приведённая к 1НФ:

Сотрудник	**Номер телефона**
Иванов И. И.	283-56-82
Иванов И. И.	390-57-34
Петров П. Ю.	708-62-34

3.5.2 Вторая нормальная форма (2НФ)

Полная функциональная зависимость: если A и B - атрибуты отношения, то атрибут B находится в полной функциональной зависимости от атрибута A, если атрибут B является функционально зависимым от A, но не зависит ни от одного собственного подмножества атрибута A.

Функциональная зависимость A->B является полной функциональной зависимостью, если удаление какого-либо атрибута из A приводит к утрате этой зависимости. Функциональная зависимость A->B называется частичной, если в A есть некий атрибут, при удалении которого эта зависимость сохраняется.

Вторая нормальная форма применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух или нескольких атрибутов. Дело в том, что отношение с первичным ключом на основе единственного атрибута всегда находится, по крайней мере, в форме 2НФ. Отношение, которое не находится в форме 2НФ, может быть подвержено аномалиям обновления.

Вторая нормальная форма (2НФ) - отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

Нормализация отношений 1НФ с приведением к форме 2НФ предусматривает устранение частичных зависимостей. Если в отношении между атрибутами существует частичная зависимость, то функционально-зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Пример приведения таблицы ко второй нормальной форме

Пусть Начальник и Должность вместе образуют первичный ключ в такой таблице:

Начальник	**Должность**	**Зарплата**	**Наличие компьютера**
Гришин	Кладовщик	20000	Нет
Васильев	Программист	40000	Есть
Васильев	Кладовщик	25000	Нет

Зарплату сотруднику каждый начальник устанавливает сам, но её границы зависят от должности. Наличие же компьютера у сотрудника зависит только от должности, то есть зависимость от первичного ключа неполная.

В результате приведения к 2НФ получаются две таблицы:

Начальник	**Должность**	**Зарплата**
Гришин	Кладовщик	20000
Васильев	Программист	40000
Васильев	Кладовщик	25000

Здесь первичный ключ, как и в исходной таблице, составной, но единственный не входящий в него атрибут Зарплата зависит теперь от всего ключа, то есть полно.

Должность	**Наличие компьютера**
Кладовщик	Нет
Программист	Есть

3.5.3 Третья нормальная форма (3НФ)

Транзитивная зависимость: если для атрибутов A, B и C некоторого отношения существуют зависимости вида A->B и B->C, это означает, что атрибут C транзитивно зависит от атрибута A через атрибут B (при условии, что атрибут A функционально не зависит ни от атрибута B, ни от атрибута C).

Транзитивная зависимость является одним из типов функциональной зависимости.

Третья нормальная форма (3НФ) - отношение, которое находится в первой и во второй нормальных формах и не имеет атрибутов, не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависит от этого первичного ключа.

Нормализация отношений 2НФ с образованием отношений 3НФ предусматривает устранение транзитивных зависимостей. Если в отношении существует транзитивная зависимость между атрибутами, то транзитивно зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Определения второй (2НФ) и третьей (3НФ) нормальных форм, приведенные выше, не допускают наличия частичных или транзитивных зависимостей от первичного ключа отношения, поскольку только при этом условии можно избежать аномалий обновления. Но в этих определениях не рассматриваются другие потенциальные ключи отношения, даже если они существуют. Приведем общие определения форм 2НФ и 3НФ, в которых учитываются потенциальные ключи отношения.

Следует отметить, что реализация этого требования не влечет за собой корректировку определения формы 1НФ, поскольку эта нормальная форма не зависит от ключей и функциональных зависимостей. В качестве общих определений укажем, что атрибут первичного ключа входит в состав любого потенциального ключа и что частичные, полные и транзитивные зависимости рассматриваются с учетом всех потенциальных ключей отношения.

Вторая нормальная форма (2НФ) - отношение, находящееся в первой нормальной форме, в котором каждый атрибут, отличный от атрибута первичного ключа, является полностью функционально независимым от любого потенциального ключа.

Третья нормальная форма (3НФ) - отношение, находящееся в первой и второй нормальной форме, в котором ни один атрибут, отличный от атрибута первичного ключа, не является транзитивно зависимым ни от одного потенциального ключа.

При использовании этих общих определений форм 2НФ и 3НФ необходимо убедиться в отсутствии частичных и транзитивных зависимостей от всех потенциальных ключей, а не только от первичного ключа. Такое требование может повлечь за собой усложнение процесса нормализации, но эти общие определения налагают дополнительные ограничения на отношения и могут позволить выявить скрытую избыточность в отношениях, которая в ином случае могла остаться незамеченной.

Необходимо найти компромисс между стремлением к максимальному упрощению процесса нормализации путем исследования зависимостей только от первичных ключей, что позволяет выявить лишь наиболее обременительную и очевидную избыточность в отношениях, и тенденцией к использованию общих определений для повышения вероятности выявления скрытой избыточности. Но на практике чаще всего результаты декомпозиции являются одинаковыми, независимо от того, используются ли определения форм 2НФ и 3НФ, основанные на первичных ключах, или общие определения, приведенные в настоящем разделе.

Пример приведения таблицы к третьей нормальной форме Исходная таблица:

Фамилия	**Отдел**	**Телефон**
Гришин	1	11-22-33
Васильев	1	11-22-33
Петров	2	44-55-66

В результате приведения к 3НФ получаются две таблицы:

Фамилия	**Отдел**
Гришин	1
Васильев	1
Петров	2

Отдел	**Телефон**
1	11-22-33
2	44-55-66

3.5.4 Нормальная форма Бойса-Кодда (НФБК)

Отношения базы данных проектируются таким образом, чтобы можно было исключить в них присутствие частичных или транзитивных зависимостей, поскольку эти зависимости приводят к появлению аномалий обновления. До сих пор мы использовали определения второй и третьей нормальных форм, для получения которых требуется найти и исключить частичные и транзитивные зависимости от первичного ключа. Однако, как описано в разделе 13.8, в этих определениях не рассматриваются такие же зависимости от потенциальных ключей отношения, если таковые имеются, В разделе 13.8 приведены общие определения форм 2НФ и ЗНФ. Применение этих общих определений может позволить выявить дополнительную избыточность, вызванную зависимостями от всех потенциальных ключей. Но даже после ввода этих дополнительных ограничений в отношениях все еще могут существовать зависимости, которые приводят к появлению избыточности в отношениях ЗНФ, С учетом этого недостатка третьей нормальной формы была разработана более строгая нормальная форма, получившая название нормальной формы Бойса-Кодда (НФБК).

Определение нормальной формы Бойса-Кодда

Нормальная форма Бойса-Кодда (НФБК) основана на функциональных зависимостях, в которых учитываются все потенциальные ключи отношения. Тем не менее в форме НФБК предусмотрены более строгие ограничения по сравнению с общим определением формы ЗНФ.

Нормальная форма Бойса-Кодда (НФБК): отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

Для проверки принадлежности отношения к НФБК необходимо найти все его детерминанты и убедиться в том, что они являются потенциальными ключами. Напомним, что детерминантом является один атрибут или группа атрибутов, от которой полностью функционально зависит другой атрибут.

Различие между ЗНФ и НФБК заключается в том, что функциональная зависимость А—>В допускается в отношении ЗНФ, если атрибут В является первичным ключом, а атрибут А не обязательно является потенциальным ключом. Тогда как в отношении НФБК эта зависимость допускается только тогда, когда атрибут А является потенциальным ключом. Следовательно, нормальная форма Бойса-Кодда является более строгой версией формы ЗНФ, поскольку каждое отношение НФБК является также отношением ЗНФ, но не всякое отношение ЗНФ является отношением НФБК.

Пример приведения таблицы к нормальной форме Бойса—Кодда

Исходная таблица:

Номер клиента	**Дата собеседования**	**Время собеседования**	**Номер комнаты**	**Номер сотрудника**
С345	13.10.03	13.00	103	А138

C355	13.10.03	13.05	103	A136
C368	13.09.03	13.00	102	A154
C366	13.09.03	13.30	105	A207

В результате приведения к форме Бойса—Кодда получаются две таблицы:

Номер клиента	**Дата собеседования**	**Время собеседования**	**Номер Сотрудника**
C345	13.10.03	13.00	A138
C355	13.10.03	13.05	A136
C368	13.09.03	13.00	A154
C366	13.09.03	13.30	A207

Дата собеседования	**Номер сотрудника**	**Номер комнаты**
13.10.03	A138	103
13.10.03	A136	103
13.09.03	A154	102
13.09.03	A207	105

Пояснение терминов по теме НОРМАЛИЗАЦИЯ:

- Тип сущности — это группа объектов с одинаковыми свойствами, которые характеризуются независимым существованием (с точки зрения проектировщика).
- Сущностью называется отдельный экземпляр типа сущности, который может быть однозначно идентифицирован.
- Тип связи — это множество осмысленных ассоциаций между типами сущностей. Экземпляром связи называется однозначно идентифицируемая ассоциация, которая включает по одному экземпляру сущности каждого типа, участвующих в этой связи,
- Степенью связи называется количество сущностей, участвующих в данной связи.
- Рекурсивной связью называется связь, в которой несколько раз участвует одна и та же сущность, но в разных ролях.
- Атрибутом называется свойство типа сущности или типа связи.
- Домен атрибута представляет собой множество допустимых значений, которые могут быть присвоены одному или нескольким атрибутам.
- Простой атрибут состоит из одного компонента, который характеризуется не зависимым существованием.
- Составной атрибут состоит из нескольких компонентов, каждый из которых характеризуется независимым существованием.
- Однозначный атрибут — это атрибут, содержащий по одному значению для каждого экземпляра сущности определенного типа.

- Многозначный атрибут — это атрибут, содержащий несколько значений для каждого экземпляра сущности определенного типа.
- Производным атрибутом называется атрибут, содержащий значение, производное от значения связанного с ним атрибута или множества атрибутов, причем не обязательно из той же сущности.
- Потенциальным ключом называется атрибут или набор атрибутов, которые однозначно идентифицируют отдельные экземпляры типа сущности,
- Первичным ключом называется некоторый выбранный потенциальный ключ сущности, однозначно идентифицирующий каждый экземпляр сущности определенного типа.
- Составным ключом является потенциальный ключ, который состоит из двух или нескольких атрибутов.
- Сильный тип сущности — это сущность, существование которой не зависит ни от какой другой сущности. Слабый тип сущности — это сущность, существование которой зависит от другой сущности.
- Кратностью называется количество (заданное как одно значение или как диапазон значений) возможных экземпляров типа сущности, которые могут быть связаны с одним экземпляром соответствующего типа сущности с помощью определенной связи.
- Кратностью сложной связи называется количество (заданное как одно значение или как диапазон значений) возможных экземпляров типа сущности в n-арной связи, которое регистрируется после фиксации остальных (n-1) значений.
- Кардинальность описывает максимальное количество возможных связей для каждой сущности, участвующей в связи данного типа.
- Степень участия определяет, должны ли участвовать в конкретной связи все или только некоторые экземпляры сущности.
- Дефект типа "разветвление" возникает, если в модели данных представлена некоторая связь между типами сущностей, но путь между некоторыми экземплярами сущности определен неоднозначно.
- Дефект типа "разрыв" возникает, когда в модели предполагается связь между типами сущностей, но не существует пути между некоторыми экземплярами сущностей.

КРАТКИЙ СЛОВАРЬ ТЕРМИНОВ

Администрирование базы данных - сопровождение базы данных в процессе разработки, эксплуатации и добавления приложений; включает также создание и необходимую реорганизацию базы данных, создание резервных копий базы данных, назначение пользователям паролей и санкций доступа к базе данных.

Атрибут - информация, описывающая объект и служащая его идентификатором (поименованная характеристика объекта). В записи данных атрибут представлен типом элемента данных.

База данных - совокупность данных, организованная по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными; не зависит от прикладных программ.

Безопасность данных - защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения данных.

Внешний ключ - одно или несколько полей в таблице, которые соответствуют первичному ключу другой таблицы.

Внешняя схема - представление данных с позиции прикладного программиста, логическая структура базы данных.

Внутренняя схема - физическая структура базы данных; представление данных с позиции вычислительной системы, как они выглядят на запоминающем устройстве.

Данные - информация, представленная в определенной форме, пригодной для последующего хранения и обработки.

Домен - область определения значений одного столбца отношения.

Запрос - операция над отношениями, результатом которой также является отношение.

Инструкция SQL - предложение на языке структурированных запросов (SQL), представляющее собой запрос для выборки или обработки данных.

Информация - сведения об окружающем мире и протекающих в нем процессах.

Ключ - совокупность атрибутов, значения которых однозначно определяют кортеж в отношении.

Концептуальная схема - высокоуровневое представление администратора базы данных и пользователя о данных, обычно в виде объектов и связей.

Модель данных - представление о предметной области в виде данных и связей между ними.

Независимость данных - возможность изменения структуры базы данных без изменения пользующихся ею прикладных программ.

Объект - нечто, о чем хранится информация.

Ограничения целостности - определяемые моделью данных или задаваемые схемой базы данных ограничения, обеспечивающие внутреннюю непротиворечивость хранимой информации.

Отношение - конечное множество кортежей из допустимых значений атрибутов схемы отношения. Отношение ассоциируется с таблицей, имена атрибутов - с именами столбцов таблицы, кортежи - со строками таблицы.

Отображение - реляционная операция языка SQL.

Первичный ключ - один или несколько столбцов (атрибутов), которые однозначно идентифицируют каждую запись в таблице.

Предметная область - часть реального мира, подлежащая изучению с целью организации управления и автоматизации.

Реляционная база данных (РБД) - совокупность отношений конкретной предметной области. Предполагается, что отношения логически связаны между собой.

Реляционная СУБД - СУБД, базирующаяся на реляционной модели данных.

Реляционные операции - набор операций манипулирования данными, операндами которых являются отношения; результатом любой реляционной операции является также отношение.

Связь - ассоциация между экземплярами примитивных или агрегированных объектов (записей) данных (например, 1 : 1, 1 : M, M : N).

Система управления базами данных - комплекс программных и языковых средств, предназначенных для создания, ведения и использования баз данных.

Словарь данных - каталог всех типов элементов в базе данных, включающий для каждого типа его определение, формат, источник и применение. Широко распространены автоматические словари данных.

Структура данных - способ объединения нескольких элементов данных в один.

Структура хранения - описание способа организации физического хранения данных в системе: указатели, представление знаков, плавающая запятая, блокирование, метод доступа т.д.

Схема данных - графическое или формальное описание логической структуры базы данных.

Схема отношения - совокупность имен атрибутов, определяющих объект.

Схема реляционной базы данных - совокупность схем отношений приложения.

Тип данных - множество операций, характеризующих определенное множество значений.

Транзакция - группа операций над данными, логически рассматриваемая как одна операция и сохраняющая целостность базы данных.

Целостность базы данных - свойство базы данных, при выполнении которого база данных содержит полную и непротиворечивую информацию, необходимую и достаточную для корректного функционирования приложений; это свойство сохраняется при всех манипуляциях с данными.

Целостность данных - система правил, используемых для поддержания связей между записями в связанных таблицах.

Язык SQL - язык структурированных запросов для выборки, изменения и удаления данных из таблиц базы данных.

Список рекомендуемой литературы

1. Кузин А.В. Базы данных : учеб. Пособие для студ. высш. учеб. Заведений / А.В. Кузин, С.В. Левонисова. – 2-е изд., стер. – М.: изд. «Академия», 2008. – 320 с.
2. Райордан Р. Основы реляционных баз данных / Пер. с англ. – М.: изд. «Русская Редакция», 2001. – 384 с.
3. Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление. – 5-е изд., перераб. и доп.: Пер. с англ. – СПб.: БХВ-Петербург, 2004. – 1040 с.
4. Дейт, К., Жд. Введение в системы баз данных, 7-е изд.: Пер. с англ. – М.: «Вильямс», 2001. – 1072 с.

Список использованных источников

1. Базы данных. Нормальные формы отношений: учебно-методические указания к лабораторной работе №1 для студентов специальности 230102,230201 / НГТУ; сост.: Балашова Т.И., Н.Новгород 2010. 9с.
2. Введение в системы баз данных: учебное пособие / сост. П.В. Бураков, В.Ю. Петров, Санкт-Петербург, 2010
3. Базы данных / КРАМОРЕНКО Н. В., ДВГУ, г. Владивосток, 2004 – 85с.
4. Бобцов А.А., Шиегин В.В. Банки и базы данных. Основы работы с MS Access. Часть 1 (для пользователей). Учебное пособие. – СПб., 2005. - 93 с.
5. Бобцов А.А., Шиегин В.В. Банки и базы данных. Основы работы с MS Access. Часть 2 (для разработчиков). Учебное пособие. – СПб., 2005. - 57 с.
6. Введение в системы баз данных: методические указания для студентов вечернего и дневного отделения механико-математического факультета / Невская Е.С., Амелина Н.И., Мачулина Л.А., РГУ, г. Ростов-на-Дону, 2003 г.
7. К. Дж. Дейт, "Введение в системы баз данных", стр. 103-128, 163-234, 457-520
8. http://ru.wikipedia.org/wiki/Реляционная база данных
9. http://ru.wikipedia.org/wiki/ Нормальная_форма
10. http://ru.wikipedia.org/wiki/ Первая_нормальная_форма
11. http://ru.wikipedia.org/wiki/ Вторая_нормальная_форма
12. http://ru.wikipedia.org/wiki/ Третья_нормальная_форма
13. http://ru.wikipedia.org/wiki/ Нормальная_форма_Бойса—Кодда