БАЗЫ ДАННЫХ

Современная жизнь немыслима без эффективного управления. Важной категорией являются системы обработки информации, от которых во многом зависит эффективность работы любого предприятия ли учреждения. Такая система должна:

обеспечивать получение общих и/или детализированных отчетов по итогам работы;

позволять легко определять тенденции изменения важнейших показателей;

обеспечивать получение информации, критической по времени, без существенных задержек;

выполнять точный и полный анализ данных.

Управление информацией всегда было основной сферой применения компьютеров и, надо думать, будет играть еще большую роль в будущем. Основные идеи современной информационной технологии базируются на концепции баз данных (БД). Согласно данной концепции основой информационной технологии являются данные, организованные в Базах Данных, адекватно отражающие реалии действительности в той или иной предметной области и обеспечивающие пользователя актуальной информацией в соответствующей предметной области. Первые БД появились уже на заре 1-го поколения ЭВМ, представляя собой отдельные файлы данных или их простые совокупности. По мере увеличения объемов и структурной сложности хранимой информации, а также расширения круга потребителей информации определилась необходимость создания удобных эффективных систем интеграции хранимых данных и управления ими. В конце 60-х годов это привело к созданию первых коммерческих систем управления базами данных (СУБД), поддерживающих организацию и ведение БД.

Базы данных составляют в настоящее время основу компьютерного обеспечения информационных процессов, входящих практически во все сферы человеческой деятельности.

Процессы обработки информации имеют общую природу и опираются на описание фрагментов реальности, выраженное в виде совокупности взаимосвязанных данных. Фрагмент реальности называется *Предметной областью*, и описывается или моделируется с помощью БД и ее приложений. В предметной области выделяются информационные объекты – идентифицируемые объекты реального мира, процессы, системы, понятия и т.д., сведения о которых хранятся в БД.

Базы данных являются эффективным средством представления структур данных и манипулирования ими. Концепция баз данных предполагает использование интегрированных средств хранения информации, позволяющих обеспечить централизованное управление данными и обслуживание ими многих пользователей. При этом существенным является

постоянное повышение объемов информации, хранимой в БД, что влечет за собой требование увеличения производительности таких систем. Резко возрастает также в разнообразных применениях спрос на интеллектуальный доступ к информации. Это особенно проявляется при организации логической обработки информации в системах баз знаний, на основе которых создаются современные экспертные системы. При этом БД должна поддерживаться в среде ЭВМ единым программным обеспечением, называемым *системой управления базами данных* (СУБД). СУБД вместе с прикладными программами называют *банком данных*.

Одно из основных назначений СУБД – поддержка программными средствами представления, соответствующего реальности.

Быстрое развитие потребностей применений БД постоянно выдвигает новые требования к СУБД:

поддержка широкого спектра типов представляемых данных и операций над ними (включая фактографические, документальные, картинно-графические данные);

естественные и эффективные представления в БД разнообразных отношений между объектами предметных областей (например, пространственно-временных с обеспечением визуализации данных);

поддержка непротиворечивости данных и реализация дедуктивных БД;

обеспечение целостности БД в широком диапазоне разнообразных предметных областей и операционных обстановок;

управление распределенными БД, интеграция неоднородных баз данных;

существенное повышение надежности функционирования БД.

Вместе с тем традиционная программная реализация многочисленных функций современных СУБД на ЭВМ общего назначения приводит к громоздким и непроизводительным системам с недостаточно высокой надежностью.

Основные функции СУБД:

1. Непосредственное управление данными во внешней памяти. Эта функция включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы).

2. Управление буферами оперативной памяти. СУБД обычно работают с БД значительного размера; по крайней мере, этот размер обычно существенно больше доступного объема оперативной памяти. 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. Поэтому в развитых СУБД

поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

3. Управление транзакциями. Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.

4. Журнализация. 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: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и 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. Журнал - это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД. Во всех случаях придерживаются стратегии "упреждающей" записи в журнал (так называемого протокола Write Ahead Log - WAL). Самая простая ситуация восстановления - индивидуальный откат транзакции.

5. Поддержка языков БД. Для работы с базами данных используются специальные языки, в целом 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).

Основные понятия, с которыми придется иметь дело:

- *база данных* (БД) – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области;

- *система управления базами данных* (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного применения БД многими пользователями;

- *банк данных* (БнД) – основанная на технологии БД система программных, языковых, организационных и технических средств, предназначенных для централизованного накопления и коллективного использования данных;

- *информационная система* (ИС) – система, реализующая автоматизированный сбор, обработку и манипулирование данными и включающая технические средства обработки данных, программное обеспечение и соответствующий персонал;

- *объект (Сущность)* – элемент какой-либо системы, информация о котором сохраняется. Объект может быть как реальным (например, человек), так и абстрактным (например, событие – поступление человека в стационар).

- *таблица* – упорядоченная структура, состоящая из конечного набора однотипных записей.

- *атрибут* – информационное отображение свойств объекта. Каждый объект характеризуется набором атрибутов.

- *первичный ключ* – атрибут (или группа атрибутов), позволяющий однозначным образом определить каждую строку в таблице.

Функционально полная СУБД должна включать в свой состав средства, обеспечивающие потребности пользователей различных категорий на всех этапах жизненного цикла систем БД**:** проектирования, создания, эксплуатации.

Типовая организация современной СУБД:

Логически в современной реляционной СУБД можно выделить наиболее внутреннюю часть - ядро СУБД (часто его называют Data Base Engine), компилятор языка БД (обычно SQL), подсистему поддержки времени выполнения, набор утилит. В некоторых системах эти части выделяются явно, в других - нет, но логически такое разделение можно провести во всех СУБД.

Ядро СУБД - отвечает за управление данными во внешней памяти, управление буферами оперативной памяти, управление транзакциями и журнализацию. Соответственно, можно выделить такие компоненты ядра (по крайней мере, логически, хотя в некоторых системах эти компоненты выделяются явно), как менеджер данных, менеджер буферов, менеджер транзакций и менеджер журнала. Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую и используемым в программах, производимых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах БД. Ядро СУБД является основной резидентной частью СУБД. При использовании архитектуры "клиент-сервер" ядро является основной составляющей серверной части системы.

Основной функцией компилятора языка БД является компиляция операторов языка БД в некоторую выполняемую программу.

В отдельные утилиты БД обычно выделяют такие процедуры, которые слишком накладно выполнять с использованием языка БД, например, загрузка и выгрузка БД, сбор

статистики, глобальная проверка целостности БД и т.д. Утилиты программируются с использованием интерфейса ядра СУБД, а иногда даже с проникновением внутрь ядра.

Некоторые сведения о типах данных

СУБД используют несколько моделей данных: иерархическую и сетевую (с 60-х годов) и реляционную (с 70-х). Основное различие данных моделей в представлении взаимосвязей между объектами.

Обычно различают три класса СУБД, обеспечивающих работу иерархических, сетевых и реляционных моделей. Однако различия между этими классами постепенно стираются, причем, видимо, будут появляться другие классы, что вызывается, прежде всего, интенсивными работами в области баз знаний (БЗ) и объектно-ориентированной инфотехнологией. Поэтому традиционной классификацией пользуются все реже. Каждая из указанных моделей обладает характеристиками, делающими ее наиболее удобной для конкретных приложений. Одно из основных различий этих моделей состоит в том, что для иерархических и сетевых СУБД их структура часто не может быть изменена после ввода данных, тогда как для реляционных СУБД структура может изменяться в любое время. С другой стороны, для больших БД, структура которых остается длительное время неизменной, и постоянно работающих с ними приложений с интенсивными потоками запросов на БД-обслуживание, именно иерархические и сетевые СУБД могут оказаться наиболее эффективными решениями, ибо они могут обеспечивать более быстрый доступ к информации БД, чем реляционные СУБД.

Этапам реализации баз данных соответствуют уровни описания предметной области:

реальность в том виде, как она существует;

концептуальное описание реальности;

представление описания в виде формального текста;

физическая реализация БД на машинных носителях.

Для ввода в ПК полученное описание должно быть представлено в терминах специального языка описания данных, который входит в комплекс средств СУБД.

Процесс построения концептуального описания с учетом всех необходимых факторов называется *процессом проектирования* БД.

Простое (элементарное) данное – это наименьшая семантически значимая поименованная единица данных (например, ФИО, должность, адрес и т.д.). Значения простого данного описывает представленную им характеристику объекта для каждого экземпляра объекта. Имена простых данных хранятся в описании БД, в то время как их значения запоминаются в самой БД.

Составное данное - совокупность простых данных, объединить в которое можно двумя способами.

Во-первых, можно соединить несколько разнотипных данных. Например, данное АНКЕТА состоит из данных ТАБЕЛЬНЫЙ НОМЕР, ФИО, ГОД РОЖДЕНИЯ, ПОЛ, ДОЛЖНОСТЬ, ЗАРПЛАТА. По этому принципу образуется *структурное данное* или данное типа *структура*. Описание структуры состоит из перечисления ее составных частей, значение – из значений составляющих ее данных.

Во-вторых, *составное данное* может объединять совокупность однотипных данных (список сотрудников, послужной список сотрудника и т.п.). *Составное данное* этого типа называется *массивом*. В описании массива достаточно указать описание одного элемента, значение массива представляется однородным списком значений его элементов.

В общем случае составные данные представляют собой объединенную под одним именем совокупность данных любых типов, в том числе структур и массивов, с произвольной глубиной вложенности составных данных.

Элементы массива могут идентифицироваться ключом - данным, значения которого взаимно однозначно определяют экземпляры элементов.

Составные типы данных представляют *иерархические связи* значений данных в БД. Представление предметной области в виде структур данных с иерархическими связями носит название *иерархической модели данных*. В общем случае в базе могут быть определены также сетевые связи, позволяющие описать сеть – ориентированный граф произвольного вида. Представление предметной области в виде сетевых структур данных общего вида называется *сетевой моделью данных*. Сетевые связи реализуются путем отождествления отдельных данных БД.

Окружающий мир переполнен иерархическими данными. В это широкое понятие в компании, состоящие из дочерних компаний, филиалов, отделов и рабочих групп; детали из кот собираются узлы, входящие затем в механизмы; специальности, специализации и рабочие на начальники и подчинённые и т. д. Любая группа объектов, в которой один объект может быть «родите для произвольного числа других объектов, организована в виде иерархического дерева. Иерархич модель данных строится по принципу иерархии объектов, то есть один тип объекта является главны нижележащие – подчиненными. Устанавливается связь «один ко многим», то есть для некоторого гла типа существует несколько подчиненных типов объектов. Иначе, главный тип именуется исходным т а подчиненные – порожденными. У подчиненных типов могут быть в свою очередь подчиненные Наивысший в иерархии узел (совокупность атрибутов) называют корневым.

Иерархические модели СУБД имеют древовидную структуру. Дерево представляет собой иерархию элементов, называемую узлами. На самом верхнем уровне иерархии имеется только один узел - корень. Каждый узел, кроме корня, связан с одним узлом на более высоком уровне, называемым исходным узлом для данного узла. Ни один элемент не имеет более одного исходного. Каждый элемент может быть связан с одним или нескольким элементами на более низком уровне. Они называются порожденными. Иерархические структуры относительно просто создаются и поддерживаются. Это важно для ряда приложений, однако множество данных по своей природе не связаны в древовидные структуры. Во многих структурах данных одна запись требует более одного представления, поэтому приходится разрабатывать способы объединения данных, которые по-разному представляются различным пользователям, в одну общую схему БД. В результате получаются обычно более сложные структуры по сравнению с древовидными. Поэтому программное обеспечение, сконструированное только для работы с древовидными структурами, имеет ограниченное применение и не редко сильно влияет на возможности увеличения объема и развития БД.

Принципиальным для иерархического представления данных является то, что каждый экземпляр записи приобретает свой смысл только тогда, когда он рассматривается в своем контексте; подчиненный экземпляр записи не может существовать без своего предшественника по иерархии (несимметричность или асимметрия). Асимметрия - основной недостаток иерархического подхода, поскольку она затрудняет работу пользователя. В частности, пользователь вынужден тратить время и усилия на решение проблем, связанных со спецификой модели и никак не следующих из характера задаваемых вопросов. Очевидно, что такие проблемы усугубляются по мере увеличения числа типов записей, представленных в структуре, и по мере роста сложности иерархии. Кроме того, иерархическая модель обладает еще некоторыми нежелательными свойствами аномалии, которые ярко проявляются в связи с выполнением каждой из основных операций запоминания (добавление, удаление, модификация).

Длительный опыт использования иерархических систем показал, что они весьма эффективны лишь для достаточно простых задач, но они практически не пригодны для использования в сложных системах с оперативной обработкой транзакций и распределенной архитектурой. Иерархическая организация не может обеспечить быстродействие, необходимое для работы в условиях одновременного модифицирования файлов несколькими прикладными подсистемами. В настоящее время не разрабатываются СУБД, поддерживающие на концептуальном уровне только иерархические модели. Как правило, использующие иерархический подход системы допускают связывание древовидных структур между собой и/или установление связей внутри них. Это приводит к сетевым моделям СУБД. К основным недостаткам иерархических моделей следует отнести: неэффективность реализации отношений, медленный доступ к узлам данных нижних уровней иерархии, четкая ориентация на определенные типы запросов и др. В связи с этими недостатками ранее созданные иерархические СУБД подвергаются существенным модификациям, позволяющим поддерживать более сложные типы структур и, в первую очередь, сетевые и их модификации. Сетевая модель данных строится по принципу «главный и подчиненный тип одновременно», то есть любой тип данных одновременно может одновременно порождать несколько подчиненных типов (быть владельцем набора) и быть подчиненным для нескольких главных (быть членом набора).

Сетевая модель СУБД во многом подобна иерархической: если в иерархической модели для каждого узла записи допускается только один входной узел при N выходных, то в сетевой модели для узлов допускается несколько входных узлов наряду с возможностью наличия узлов без входов с точки зрения иерархической структуры. Узлы данных в сетевых БД могут иметь множественные связи с узлами старшего уровня. При этом направление и характер связи в сетевых БД не являются столь очевидными, как в случае иерархических БД. Поэтому имена и направление связей должны идентифицироваться при создании БД языка описания данных.

Таким образом, под сетевой СУБД понимается система, поддерживающая сетевую организацию: любая запись, называемая записью старшего уровня, может содержать данные, которые относятся к набору других записей, называемых записями подчиненного уровня. Возможно обращение ко всем записям в наборе, начиная с записи старшего уровня. Обращение к набору записей реализуется по указателям. В рамках сетевых СУБД легко реализуются и иерархические модели. Сетевые СУБД поддерживают сложные соотношения между типами данных, что делает их пригодными во многих различных приложениях. Однако пользователи таких СУБД ограничены связями, определенными для них разработчиками БД-приложений. Более того, подобно иерархическим сетевые СУБД предполагают разработку БД приложений опытными программистами и системными аналитиками.

Если в отношении между данными порожденный элемент имеет более одного исходного элемента, то это отношение уже нельзя описать как древовидную или иерархическую структуру. Его описывают в виде сетевой структуры. Любая сетевая структура может быть приведена к более простому виду введением избыточности. «БД постоянно грозит опасность стать громоздкими, застывшими и слишком сложными системами. Новые приложения порождают новые виды запросов пользователей к базе, что увеличивает набор логических связей между ее элементами. В итоге многие системы БД оказываются очень сложными в построении и эксплуатации. Если разработчики не придумают ясные и простые схемы организации, эти системы будут подобны паутине» [К.Дейт.].

Сетевая модель более симметрична, чем иерархическая модель. Однако процедуры обновления значительно сложнее. Проблема состоит в следующем: всегда имеются две стратегии для определения места одного экземпляра записи, первая начинается с "владельца" и просмотра его цепочки для выбора звена, а другая начинается с "подчиненного звена" и просмотра его цепочки для выбора "владельца".

Как в иерархических, так и сетевых СУБД при описании данных обычно указываются характеристики записей каждого типа, способствующие более эффективному размещению данных во внешней памяти и более быстрому доступу к ним. К таким характеристикам относятся: размеры полей записи (минимальные, средние, максимальные), состав ключа, допустимый набор символов, интервалы значений и т.д.

Иерархические и сетевые базы данных часто называют базами данных с навигацией. Это название отражает технологию доступа к данным, используемую при написании обрабатывающих программ на языке манипулирования данными. При этом очевидно, что доступ к данным по путям, не предусмотренным при создании базы данных, может потребовать неразумно большого времени. Повышая эффективность доступа к данным и сокращая, таким образом, время ответа на запрос, принцип навигации вместе с этим повышает и степень зависимости программ и данных. Обрабатывающие программы оказываются жестко привязанными к текущему состоянию структуры базы данных и должны быть переписаны при ее изменениях. Операции модификации и удаления данных требует переустановки указателей, а манипулирование данными остается *записи-ориентированным*. Кроме того, принцип навигации не позволяет существенно повышать уровень языка манипулирования данными, чтобы сделать его доступным пользователю-непрограммисту, или даже программисту-непрофессионалу. Для поиска записи-цели в иерархической или сетевой структуре программист должен вначале определить путь доступа, а затем просмотреть все записи, лежащие на этом пути, - шаг за шагом.

Насколько запутанной являются схемы представления иерархических и сетевых баз данных, настолько и трудоемким является проектирование конкретных прикладных систем на

их основе. Как показывает, опыт длительные сроки разработки прикладных систем нередко приводят к тому, что они постоянно находятся в стадии разработки и доработки.

Указанные и некоторые другие проблемы, с которыми столкнулись разработчики и пользователи иерархических и сетевых систем послужили стимулом к созданию реляционной модели данных и реляционных СУБД.

Реляционные базы

Реляционный подход стал широко известен благодаря первым работам Е.Кодда, которые появились около 1970г. В течение долгого времени реляционный подход рассматривался как удобный формальный аппарат анализа баз данных, не имеющий практических перспектив, так как его реализация требовала слишком больших машинных ресурсов. Только с появлением персональных ЭВМ реляционные и близкие к ним системы неожиданно стали распространяться, практически не оставив места другим моделям. Один из самых естественных способов представления данных для пользователей - это двумерная таблица. Она привычна для пользователя, понятна и обозрима, ее легко запомнить. Поскольку любая сетевая структура может быть разложена в совокупность древовидных структур, то и любое представление данных может быть сведено к двумерным плоским файлам. Связи между данными могут быть представлены в форме двумерных таблиц.

Таблица обладает следующими свойствами:

Каждый элемент таблицы представляет собой один элемент данных. Повторяющиеся группы отсутствуют.

Все столбцы в таблице однородные. Это означает, что элементы столбца имеют одинаковую природу.

Столбцам присвоены уникальные имена.

В таблице нет двух одинаковых строк.

Порядок расположения строк и столбцов в таблице безразличен. Таблица такого рода называется отношением. База данных, построенная с помощью отношений, называется реляционной базой данных.

Чем же принципиально отличаются реляционные модели от сетевых и иерархических? Вкратце на это можно ответить следующим образом: иерархические и сетевые модели данных - имеют связь по структуре, а реляционные - имеют связь по значению. Проектирование баз данных традиционно считалось очень трудной задачей. Реляционная технология значительно упрощает эту задачу. Реляционная модель данных объекты и связи между ними представляются в виде таблиц, при этом связи тоже рассматриваются как объекты. Все строки, составляющие таблицу в реляционной базе данных должны иметь первичный ключ. Все современные средства СУБД поддерживают реляционную модель данных. В реляционных базах данных (Relational Database System, RDBS) все данные отображаются в двумерных таблицах. База данных, таким образом, это ни что иное, как набор таблиц.

Приложения, работающие с базами данных, ориентируются на один из двух основных их видов: плоские таблицы (flat files) или реляционные базы. Хотя плоские таблицы были стандартом в течение многих лет, в настоящее время они используются только в приложениях типа Microsoft Works или Microsoft Excel. В плоской таблице вся взаимосвязанная информация

должна находиться в одной таблице. Это означает, что любые данные, повторяющиеся в нескольких записях, должны присутствовать в каждой из этих записей.

По мере развития и усложнения баз данных стало ясно, что такой способ неэффективен при хранении больших объемов информации - в результате возникла идея реляционных баз данных. В реляционной базе используется несколько разных таблиц, между которыми устанавливаются связи (relations), Они позволяют ввести информацию в одну таблицу и связать ее с записями другой таблицы через специальный идентификатор.

Таким образом, в реляционной базе данных повторяющаяся в обычной плоской таблице информация вводится один раз в отдельную таблицу.

Хранение данных в связанных таблицах обладает рядом преимуществ:

- экономия времени, поскольку одни и те же данные не приходится вводить в нескольких таблицах;
- уменьшение размера базы данных (порой весьма значительное по сравнению с размером плоской таблицы), которое экономит дисковое пространство на вашем компьютере и облегчает перенос базы данных;
- существенное сокращение количества ошибок. Отпадает необходимость ввода одних и тех же данных в большое количество записей. Если же повторяющиеся данные хранятся в связанной таблице, то информацию достаточно ввести всего один раз; после этого в исходной таблице для повторяющихся данных вводится только код (обычно короткая цифровая или алфавитно-цифровая последовательность). Можно настроить поле так, чтобы выбрать код из списка и обойтись вообще без набора текста.

Обновление данных является одношаговым процессом.

Реляционные базы данных обладают рядом других преимуществ по скорости и производительности, но для понимания работы с ними вовсе не обязательно разбираться в теории. Необходимо осознать всего несколько положений:

- поле является информационной категорией;
- значением называется информация, содержащаяся в определенном поле одной записи;
- записью называется совокупность взаимосвязанных значений для одного элемента базы данных (заполняющих строку в таблице);
- связи между таблицами создаются с помощью данных, находящихся в специальных полях, и это позволяет обойтись однократным вводом повторяющейся информации.

Базы данных называются *реляционными*, если управление ими основано на математической модели, использующей методы реляционной алгебры и реляционного исчисления.

С. Дейт дает следующее неформальное определение реляционных баз данных:

- *Вся информация в базе данных представлена в виде таблиц.*
- *Поддерживаются три реляционных оператора – выбора, проектирования и объединения, с помощью которых можно получить любые необходимые данные, заложенные в таблицы.*

Доктор И.Ф. Кодд, автор реляционной модели баз данных, разработал целый список критериев, которым должна удовлетворять реляционная модель. Описание этого списка, часто называемого «12 правилами Кодда», требует введения сложной терминологии но, тем не менее, можно назвать некоторые правила Кодда для реляционных систем. Чтобы считаться реляционной по Кодду, система управления базами данных должна:

- *Представлять всю информацию в виде таблиц;*
- *Поддерживать логическую структуру данных, независимо от их физического представления;*
- *Использовать язык высокого уровня для структурирования, выполнения запросов и изменения информации в базах данных;*
- *Поддерживать основные реляционные операции (выбор, проектирование и объединение), а также теоретико-множественные операции, такие как объединение, пересечение и дополнение;*
- *Поддерживать виртуальные таблицы, обеспечивая пользователям альтернативный способ просмотра данных в таблицах;*
- *Различать в таблицах неизвестные значения (nulls), нулевые значения и пропуски в данных;*
- *Обеспечивать механизмы для поддержки целостности, авторизации, транзакций и восстановления данных.*

Первое правило Кодда гласит, что вся информация в реляционных базах данных представляется значениями в таблицах. В реляционных системах таблицы состоят из горизонтальных *строк* и вертикальных *столбцов*. Все данные представляются в табличном формате – другого способа просмотреть информацию в базе данных не существует. Набор связанных таблиц образует *базу данных*. Таблицы в реляционной базе разделены, но полностью равноправны. Между ними не существует никакой иерархии.

Каждая таблица состоит из строк и столбцов. Каждая строка описывает отдельный *объект* или *сущность* – ученика, предмет, день недели или что-нибудь другое. Каждый столбец описывает одну характеристику объекта – имя или фамилию ученика, его адрес, оценку, дату.

Каждый элемент данных, или *значение*, определяется пересечением строки и столбца. Чтобы найти требуемый элемент данных, необходимо знать имя содержащей его таблицы, столбец и значение его *первичного ключа*, или уникального идентификатора.

В реляционной базе данных существует два типа таблиц – пользовательские таблицы и системные таблицы. Пользовательские таблицы содержат информацию, для поддержки которой собственно и создавались реляционные базы данных. Системные таблицы обычно поддерживаются самой СУБД, однако доступ к ним можно получить так же, как и к любым другим таблицам. Возможность получения доступа к системным таблицам, по аналогии с любыми другими таблицами, составляет основу другого правила Кодда для реляционных систем.

Реляционная модель обеспечивает *независимость данных* на двух уровнях – физическом и логическом. *Физическая независимость* данных означает с точки зрения пользователя, что представление данных абсолютно не зависит от способа их физического хранения. Как следствие этого, физическое перемещение данных никоим образом не может повлиять на логическую структуру базы данных. Другой тип независимости, обеспечиваемый реляционными системами - *логическая независимость* – означает, что изменение взаимосвязей между таблицами и строками не влияет на правильное функционирование программных приложений и текущих запросов.

В определении системы управления реляционными базами данных упоминаются три операции по выборке данных – *проектирование, выбор и объединение*, которые позволяют строго указать системе, какие данные необходимо показать. Операция проектирования выбирает столбцы, операция выбора – строки, а операция объединения собирает вместе данные из связанных таблиц.

Виртуальные таблицы можно рассматривать как некоторую перемещаемую по таблицам рамку, через которую можно увидеть только необходимую часть информации. Виртуальные таблицы можно получить из одной или нескольких таблиц базы данных (включая и другие виртуальные таблицы), используя любые операции выбора, проектирования и объединения. Виртуальные таблицы, в отличие от «настоящих», или базовых таблиц, физически не хранятся в базе данных. В то же время необходимо осознавать, что виртуальные таблицы это не копия некоторых данных, помещаемая в другую таблицу. Когда вы изменяете, данные в виртуальной таблице, то тем самым изменяете данные в базовых таблицах. В идеальной реляционной системе с виртуальными таблицами можно оперировать, как и с любыми другими таблицами. В реальном мире на виртуальные таблицы накладываются определенные ограничения, в частности на обновление. Одно из правил Кодда гласит, что в истинно реляционной системе над виртуальными таблицами можно выполнять все

«теоретически» возможные операции. Большинство современных систем управления реляционными базами данных не удовлетворяют этому правилу полностью.

В реальном мире управления информацией данные часто являются неизвестными или неполными: неизвестен телефонный номер, не захотели указать возраст. Такие пропуски информации создают «дыры» в таблицах. Проблема, конечно, состоит не в простой неприглядности подобных дыр. Опасность состоит в том, что из-за них база данных может стать противоречивой. Чтобы сохранить целостность данных в реляционной модели, так же, как и в правилах Кодда, для обработки пропущенной информации используется понятие нуля.

«Нуль» не означает пустое поле или обычный математический нуль. Он отображает тот факт, что значение неизвестно, недоступно или неприменимо. Существенно, что использование нулей инициирует переход с двухзначной логики (да/нет) на трехзначную (да/нет/может быть). С точки зрения другого эксперта по реляционным системам, Дейта, нули не являются полноценным решением проблемы пропусков информации. Тем не менее, они являются составной частью большинства официальных стандартов различных реляционных СУБД.

Целостность – очень сложный и серьезный вопрос при управлении реляционными базами данных. Несогласованность между данными может возникать по целому ряду причин. Несогласованность или противоречивость данных может возникать вследствие сбоя системы – проблемы с аппаратным обеспечением, ошибки в программном обеспечении или логической ошибки в приложениях. Реляционные системы управления базами данных защищают данные от такого типа несогласованности, гарантируя, что команда либо будет исполнена до конца, либо будет полностью отменена. Этот процесс обычно называют *управлением транзакциями*.

Другой тип целостности, называемый *объектной целостностью,* связан с корректным проектированием базы данных. Объектная целостность требует, чтобы ни один первичный ключ не имел нулевого значения.

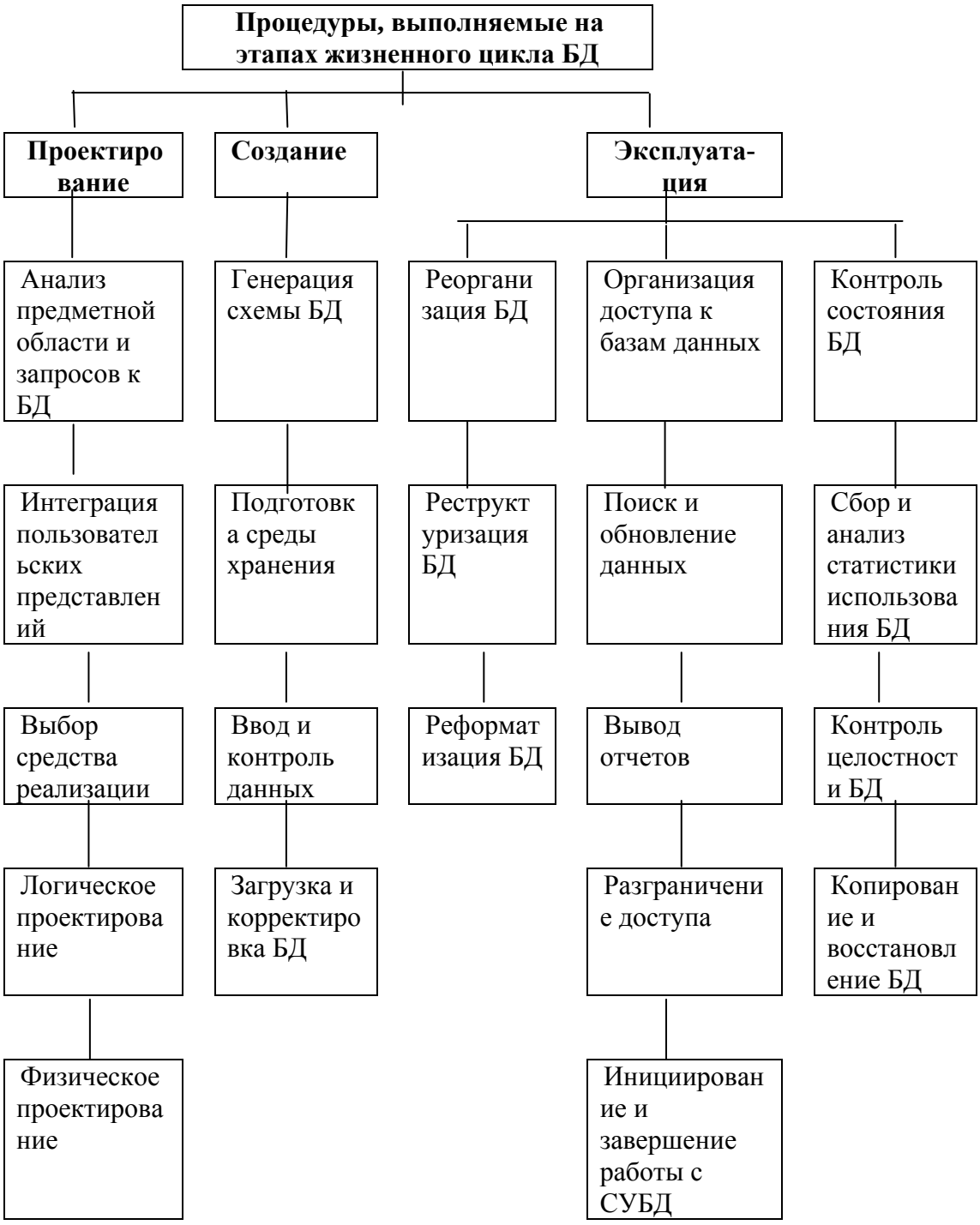
Третий тип целостности, называемой *ссылочной целостностью,* означает непротиворечивость между частями информации, повторяющимися в разных таблицах. Например, если вы изменяете неправильно введенный номер карточки страхового полиса в одной таблице, другие таблицы, содержащие эту же информацию, продолжают ссылаться на старый номер, поэтому необходимо обновить и эти таблицы. *Чрезвычайно важно, чтобы при изменении информации в одном месте, она соответственно изменялась и во всех других местах*. Кроме того, по определению Кодда, ограничения на целостность должны:

- Определяться на языке высокого уровня, используемом системой для всех других целей;
- Храниться в словаре данных, а не в программных приложениях.

Эти возможности в том или ином виде реализованы в большинстве систем.

Проектирование баз данных

Процесс, в ходе которого решается, какой вид будет у вновь создаваемой БД, называется проектированием базы данных. На этапе проектирования необходимо предусмотреть все возможные действия, которые могут возникнуть на различных этапах жизненного цикла базы данных БД.

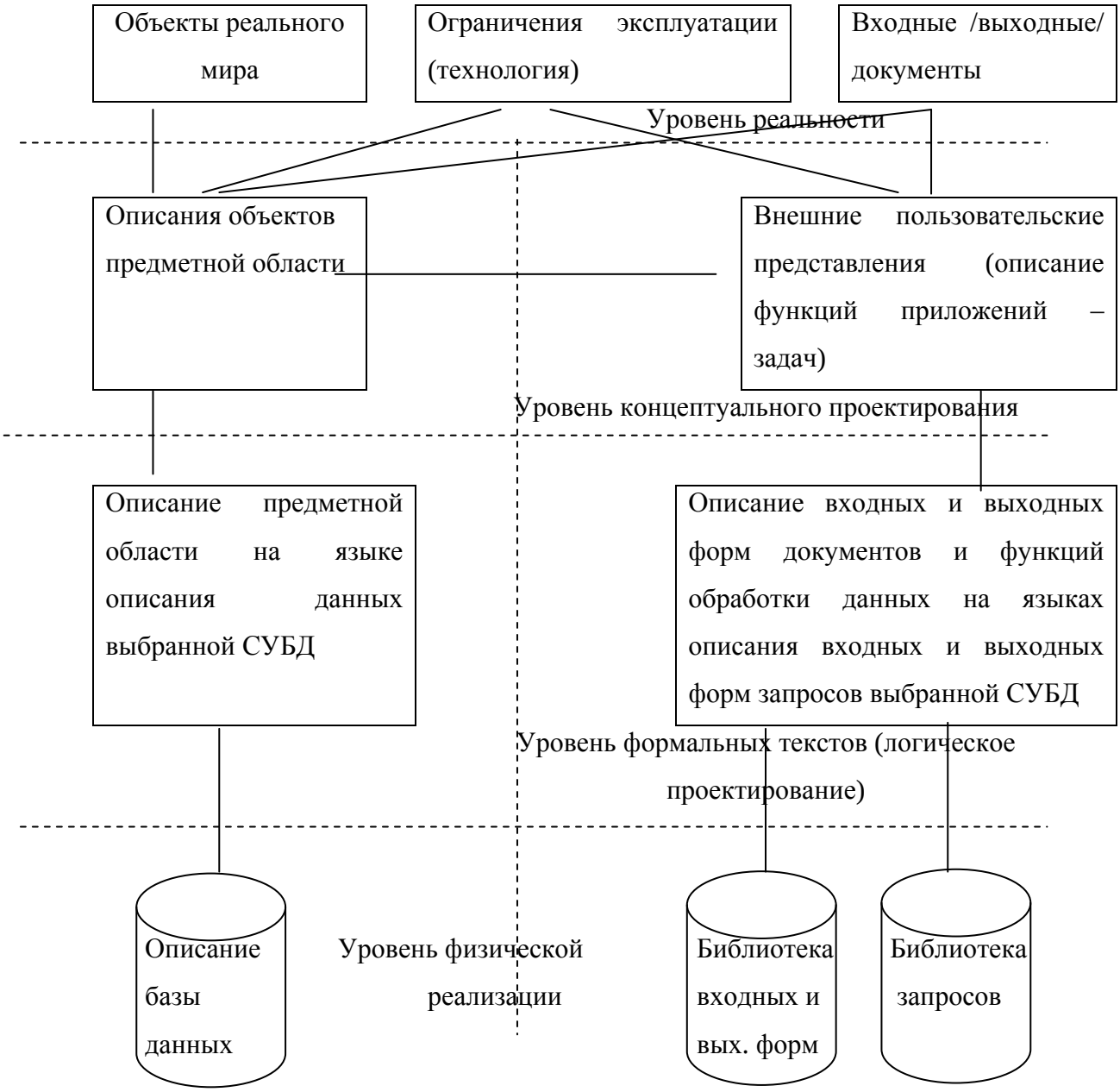


Анализ предметной области и запросов к БД.

На данном этапе необходимо проанализировать запросы пользователей, выбрать информационные объекты и их характеристики и на основе анализа структурировать предметную область.

Анализ предметной области целесообразно разбить на три фазы:

- Анализ концептуальных требований и информационных потребностей;
- Выявление информационных объектов и связей между ними;
- Построение концептуальной модели предметной области и проектирование концептуальной схемы БД



Анализ концептуальных требований

На этапе анализа концептуальных требований и информационных потребностей необходимо решить следующие задачи:

- Анализ требований пользователей к БД (концептуальных требований);
- Выявление имеющихся задач по обработке информации, которая должна быть представлена в БД (анализ приложений);
- Выявление перспективных задач (перспективных приложений);
- Документирование результатов анализа.

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики получают в диалоге с будущими пользователями БД. Здесь же выясняются требования к вводу, обновлению и корректировке информации. Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных приложений.

Например, в случае разработки БД для ведения электронной документации учебного заведения необходимо получить ответы на вопросы:

1. Сколько студентов учится в учебном заведении?
2. Сколько смен и групп в учебном заведении?
3. Как распределены студенты по группам и сменам?
4. Сколько предметов дается по каждой параллели и в каких объемах?
5. Сколько имеется учебных аудиторий?
6. Сколько преподавателей в учебном заведении, их специализация и классность?
7. Как часто обновляется информация в БД?
8. Какие существуют виды отчетов, справок и диаграмм?

Необходимо решить задачи:

1. Ведения личных дел студентов
2. Ведения журналов групп
3. Составление расписания занятий
4. Ведения табеля рабочего времени преподавателей

На основе информации хранящейся в БД необходимо выдавать следующие отчеты:

1. Карточки успеваемости студентов
2. Ведомость успеваемости и посещаемости группы
3. Динамика роста успеваемости по группам и учебном заведении
4. Отчет по успеваемости за год
5. Таблица мониторинга учебного процесса

6. Статистические данные по количеству студентов
7. Результаты тестирования
8. Результаты работы преподавателей
9. Результаты экзаменов по семестрам
10. Качество знаний студентов
11. Отчеты по предметам
12. Акт о несчастном случае
13. Протокол государственного экзамена
14. Сведения о травматизме за учебный год/семестр
15. Список выбывших студентов
16. Движение за год
17. График результатов успеваемости по семестрам
18. График итогов успеваемости за год.

Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ним, характеристики информационных объектов.

При выборе информационных объектов необходимо ответить на ряд вопросов:

1. На какие таблицы можно разбить данные, подлежащие хранению в БД?
2. Какое имя можно присвоить каждой таблице?
3. Какие наиболее интересные характеристики (с точки зрения пользователя) можно выделить?
4. Какие имена можно присвоить выбранным характеристикам?

В нашем случае предполагается завести следующие таблицы:

Учебное заведение	**Группа**	**Предметы**	**Студенты**	**Преподаватель**	**Оценки**
Номер	Группа	Предмет	Группа	Фамилия	Группа
Телефон	Смена		Фамилия	Имя Отчест	Предмет
Директор			Имя	Предмет	Фамилия
					Имя
					Дата
					Оценка

Выделяются связи между информационными объектами, в процессе которого необходимо ответить на следующие вопросы:

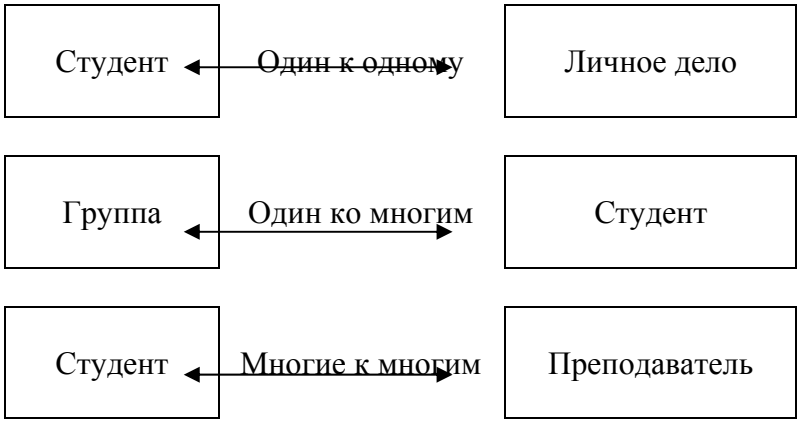
1. Какие типы связей между информационными объектами?
2. Какое имя можно присвоить каждому типу связей?
3. Каковы возможные типы связей, которые могут быть использованы впоследствии?

При задании ограничений на объекты, их характеристики и связи необходимо ответить на вопросы:

1. Какова область значений для числовых характеристик?
2. Каковы функциональные зависимости между характеристиками одного информационного объекта?
3. Какой тип отображения соответствует каждому типу связей?

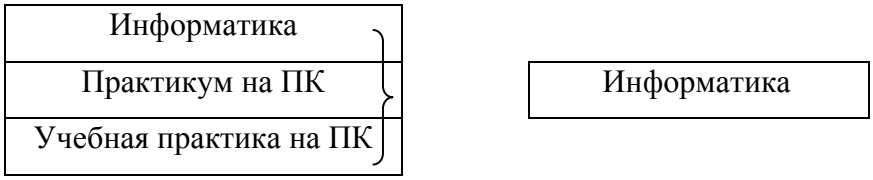
Для объектов баз данных существуют взаимосвязи между информационными объектами трех типов: «один к одному», «один ко многим», «многие ко многим».

Например:



Построение концептуальной модели

В простых случаях для построения концептуальной схемы используют традиционные методы агрегации и обобщения. При агрегации объединяются информационные объекты (элементы данных) в один в соответствии с семантическими связями между объектами. Например, занятие по информатике группы ЭФК-111 проводится в ауд.3108, начало в 8-30. Методом агрегации создаем информационный объект (сущность) РАСПИСАНИЕ со следующими атрибутами: «Группа», «предмет», «аудитория», «время». При обобщении информационные объекты (элементы данных) объединяются в родовой объект:



Выбор модели диктуется, прежде всего, характером предметной области и требованиями к БД. Немаловажным обстоятельством является независимость концептуальной модели от СУБД, которая должна быть выбрана после построения концептуальной схемы.

Модели «сущность-связь», дающие возможность представлять структуру и ограничения реального мира, а затем трансформировать их в соответствии с возможностями промышленных СУБД, являются весьма распространенными.

Под сущностью понимают основное содержание того явления, процесса или объекта, о котором собирают информацию для БД. В качестве сущности могут выступать место, вещь, личность, явление и т.д. При этом различают тип сущности и экземпляр сущности. Под типом сущности обычно понимают набор однородных объектов, выступающих как целое. Понятие «экземпляр сущности» относится к конкретному предмету. Например:

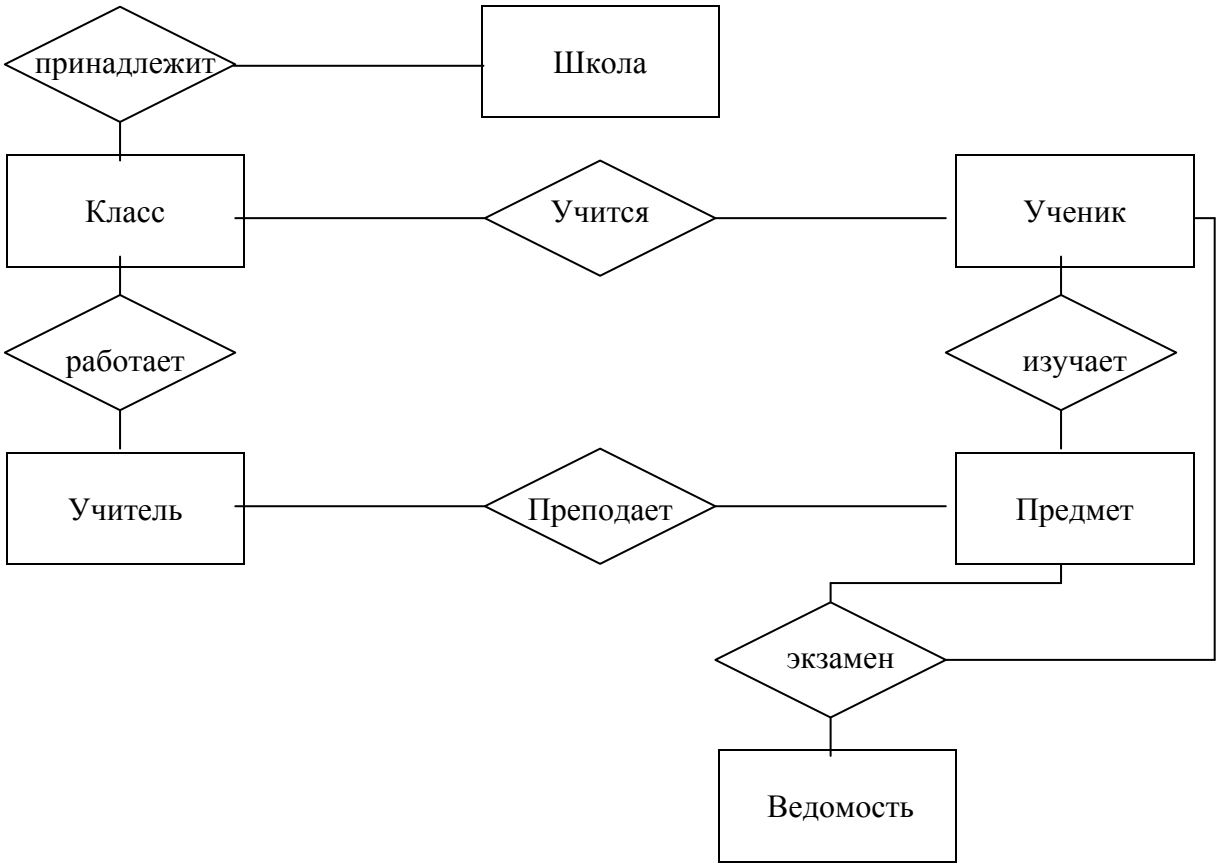
Тип сущности - студент

Экземпляр сущности - Иванов, Петров, Сидоров и др.

В нашем примере Учебное заведение, Группа, Предметы, Студенты, Преподаватели, Оценки – сущности. Проанализируем связи между сущностями.

Название связи	**Между сущностями**	
Учится	Студенты	Группа
Изучает	Студенты	Предмет
Имеет	Школа	Группа
Преподает	Преподаватели	Предмет
Работает	Преподаватели	Предмет

Теперь можно перейти к проектированию информационной (концептуальной) схемы БД (атрибуты сущностей не показаны).



Логическое проектирование

Логическое проектирование представляет собой необходимый этап при создании БД. Основной задачей логического проектирования является разработка логической схемы, ориентированной на выбранную систему управления базами данных. Процесс логического проектирования состоит из следующих этапов:

1. Выбор конкретной СУБД;
2. Отображение концептуальной схемы на логическую схему;
3. Выбор языка манипулирования данными.

Выбор конкретной СУБД

Одним из основных критериев выбора СУБД является оценка того, насколько эффективно внутренняя модель данных, поддерживаемая системой, способна описать концептуальную схему. Системы управления базами данных, ориентированные на персональные компьютеры, как правило, поддерживают реляционную или сетевую модель данных. Подавляющее большинство современных СУБД – реляционные.

Конструирование баз данных на основе реляционной модели имеет ряд важных преимуществ перед другими моделями:

- Независимость логической структуры от физического и пользовательского представления.
- Гибкость структуры базы данных – конструктивные решения не ограничивают возможности разработчика БД выполнять в будущем самые разнообразные запросы.

Так как реляционная модель не требует описания всех возможных связей между данными, впоследствии разработчик может задавать запросы о любых логических взаимосвязях, содержащихся в базе, а не только о тех, которые планировались первоначально.

Отображение концептуальной схемы на логическую схему

При отображении информационной схемы, каждый прямоугольник схемы отображается в таблицу, которая является одним отношением. При этом следует учитывать ограничения на размер таблиц, которые накладывает конкретная СУБД.

Выбор языка манипулирования данными

Важной составной частью СУБД является язык манипулирования данными, который используется при работе различных приложений с БД. Как правило, язык манипулирования данными встраивается в язык программирования. Кроме того, при выборе СУБД, реализующей конкретную БД, необходимо оценить и техническую сторону дела, которая непосредственно связана с производительностью системы. В целом необходимо оценить семь групп параметров для выбора СУБД:

- Характеристики ПК: тип, модель, фирма производитель, наличие гарантии.
- Управление файлами и поиск: тип связи, модификация нескольких файлов, двунаправленное соединение таблиц, язык манипулирования данными, тип поиска.
- Средства поддержки приложений: каталог данных, генератор приложений, процедурный язык, подпрограммы, макросы, отладчик, система поддержки исполнения, шифровка программ и данных, разграничения доступа, графика, текстовый редактор, статистика.
- Ввод и поддержка целостности: управление с помощью команд, управление с помощью меню, проверка целостности по таблице, проверка уникальности ключа, проверка по дате, независимость данных.
- Отчеты: отчеты по нескольким файлам, сохранение форматов отчетов, выдача отчета на экран, выдача отчета на магнитный носитель, вычисляемые поля, группы, переопределение формата даты, заголовки отчетов, генератор отчетов, итоговые поля, максимальная ширина отчета.

- Операционная среда: тип операционной системы, объем требуемой оперативной памяти, необходимость использования постоянной памяти, объем требуемой постоянной памяти, язык подсистемы.
- Дополнительные сведения: наличие сетевого варианта, стоимость, примечание, источники.

Интерфейс Базы Данных.

Интерфейс определяет переход от представления данных в БД к представлению, принятому среди пользователей, и обратно. В общем случае пользователи представляют данные в виде документов различных видов, от произвольных текстов до справок и таблиц фиксированного формата.

Интерфейс доступа конечного пользователя охватывает комплекс технических, организационных и программных решений, обеспечивающих в итоге унифицированность, хорошую понимаемость и надежность взаимодействия конечного пользователя с различными моделями персональных компьютеров.

Под документом понимается произвольный структурированный текст, который может быть представлен на алфавитно-цифровых печатающих устройствах. При этом под структурой текста понимается структура взаимосвязей данных, составляющих текст.

В процессе проектирования, как правило, возникает необходимость точного учета структур документов. Для полного представления этих структур могут использоваться средства описания данных БД. Тем самым облегчается процесс сопоставления БД и документов при организации интерфейса.

Совместная реализация БД и интерфейса на единой концептуальной основе предполагает сопоставление соответствующих понятий концептуального описания с понятиями пользователей.

Конкретные функциональные требования пользователей и предполагаемое их обеспечение отображаются понятием *пользовательского представления данных*. В общем случае пользовательское представление включает так называемое локальное внешнее представление функций обработки данных, а также определение форматов входных и выходных данных.

ACCESS

СУБД Access является системой управления базами данных реляционного типа. Данные хранятся в такой базе в виде таблиц, строки (записи) которых состоят из наборов полей определенных типов. С каждой таблицей могут быть связаны индексы (ключи), задающие нужные пользователю порядки на множестве строк. Таблицы могут иметь однотипные поля (столбцы), и это позволяет устанавливать между ними связи, выполнять операции реляционной алгебры. Типичными операциями над базами данных являются определение, создание и удаление таблиц, модификация определений (структур, схем) существующих таблиц, поиск данных в таблицах по определенным критериям (выполнение запросов), создание отчетов о содержимом базы данных.

СУБД позволяет задавать типы данных и способы их хранения. Можно также задать критерии (условия), которые СУБД будет в дальнейшем использовать для обеспечения правильности ввода данных. В самом простом случае условие на значение должно гарантировать, что не будет введен случайно в числовое поле буквенный символ. Другие условия могут определять область или диапазоны допустимых значений вводимых данных.

Microsoft Access предоставляет максимальную свободу в задании типа данных (текст, числовые данные, даты, время, денежные значения, рисунки, звук, электронные таблицы). Можно задавать также форматы хранения представления этих данных при выводе на экран или печать. Для уверенности, что в базе хранятся только корректные значения, можно задать условия на значения различной степени сложности.

Так как Microsoft Access является современным приложением Windows, можно использовать в работе все возможности DDE (динамический обмен данными) и OLE (связь и внедрение объектов). DDE позволяет осуществлять обмен данными между Access и любым другим поддерживающим DDE приложением Windows. В Microsoft Access можно при помощи макросов или Visual Basic (VBA) осуществлять динамический обмен данными с другими приложениями.

OLE является более изощренным средством Windows, которое позволяет установить связь с объектами другого приложения или внедрить какие-либо объекты в базу данных Access. Такими объектами могут быть картинки, диаграммы, электронные таблицы или документы из других поддерживающих OLE приложений Windows.

В Microsoft Access для обработки данных базовых таблиц используется мощный язык SQL (структурированный язык запросов). Используя SQL можно выделить из одной или нескольких таблиц необходимую для решения конкретной задачи информацию. Access значительно упрощает задачу обработки данных. Совсем не обязательно знать язык SQL. При любой обработке данных из нескольких таблиц Access использует однажды заданные связи между таблицами.

В Microsoft Access имеется также простое и в то же время богатое возможностями средство графического задания запроса – так называемый «запрос по образцу» (query by example), которое используется для задания данных, необходимых для решения некоторой задачи. Используя для выделения и перемещения элементов на экране стандартные приемы работы с мышью в Windows и несколько клавиш на клавиатуре, можно буквально за секунды построить довольно сложный запрос.

Microsoft Access спроектирован таким образом, что он может быть использован как в качестве самостоятельной СУБД на отдельной рабочей станции, так и в сети – в режиме «клиент-сервер». Поскольку в Microsoft Access к данным могут иметь доступ одновременно несколько пользователей, в нем предусмотрены надежные средства защиты и обеспечения целостности данных. Можно заранее указать, какие пользователи или группы пользователей могут иметь доступ к объектам (таблицам, формам, запросам) базы данных. Microsoft Access автоматически обеспечивает защиту данных от одновременной их корректировки разными пользователями. Access также опознает и учитывает защитные средства других подсоединенных к базе данных структур (таких, как базы данных Paradox, dBASE и SQL).

Практически все существующие СУБД имеют средства разработки приложений, которые могут использованы программистами или квалифицированными пользователями при создании процедур для автоматизации управления и обработки данных.

Microsoft Access предоставляет дополнительные средства разработки приложений, которые могут работать не только с собственными форматами данных, но и с форматами других наиболее распространенных СУБД. Возможно, наиболее сильной стороной Access является его способность обрабатывать данные электронных таблиц, текстовых файлов, файлов dBASE, Paradox, Btrieve, FoxPro и любой другой базы данных SQL, поддерживающей стандарт ODBE. Это означает, что можно использовать Access для создания такого приложения Windows, которое может обрабатывать данные, поступающие с сетевого сервера SQL или базы данных SQL на главной ЭВМ.