**Базы данных лекции, конспект**

**Понятие базы данных**

**1. Основные понятия базы данных**

Явления реального мира зачастую могут быть описаны с помощью структурных взаимосвязей между совокупнос­тями фактов. Для представления информации подобного рода в явлениях может быть использована структурная модель данных.

**База данных (БД)** — это поименованная совокупность структурированных данных о конкретных объектах реаль­ного мира в какой-либо предметной области.

База данных (БД) - это один или несколько файлов данных, предназначенных для хранения, изменения и обработки больших объемов взаимосвязанной информации.

Структури­рование данных предполагает существование (или уста­новление) между ними каких-то отношений (связей). В зависимости от характера этих отношений можно выде­лить несколько ***классификационных признаков структур данных.***

**2.****Классификация структур данных по различным признакам**

1. **По отношению порядка** структуры данных делятся на упорядоченные и неупорядоченные.

В ***упорядоченных структурах*** элементы размещаются по порядку в соответствии со значением некоторого при­знака. Наиболее простым признаком является порядко­вый номер элемента; установление порядка в соответст­вии с номером называется *нумерацией.* При этом если весь набор имеет один общий идентификатор (например, М), то отдельным данным присваиваются собственные идентификаторы — индексы (например, М5 или Mb). Ча­ще всего индекс задается целым числом, хотя это необя­зательно (в качестве индекса может выступать любой знак из конечного алфавита). Лексикографический порядок индексов определяет отношение следования между эле­ментами структуры, т. е. элемент Мб следует за элемен­том М5, а элемент Ма располагается перед элементом Mb. Примером структур, в которых упорядочение произ­водится по номеру элемента, являются *массивы.* Порядко­вый номер элемента можно считать внешним признаком, который может присваиваться элементу независимо от

его значения. Например, регистрационный номер доку­мента определяется только временем его поступления в учреждение, а не его содержанием. Помимо нумерации, в структурах данных используется упорядочение по зна­чению некоторого внутреннего признака (например, раз­мещение фамилий в алфавитном порядке или группы предприятий — в порядке убывания их рентабельности); такое упорядочение называется ***ранжированием.***

Примером ***неупорядоченных структур*** являются *мно­жества* — в них не определен порядок элементов; един­ственное, что можно установить для каких-то конкретных данных, так это их принадлежность (или непринадлеж­ность) выбранному множеству.

2. **По характеру отношений между элементами** структу-  
ры данных подразделяют на линейные и нелинейные.

В ***линейных структурах*** все элементы равноправны. К ним относят *массив, множество, стек, очередь.*

В ***нелинейных структурах*** между элементами сущест­вуют отношения подчиненности или они могут быть свя­заны логическими условиями. К ним относят *деревья, гра­фы, фреймы.*

1. **По однородности** структуры данных делят на одно­родные и неоднородные. К ***однородным*** относят структу­ры, содержащие элементарные данные только одного ти­па. Примерами однородных структур являются *массивы, множества, стеки.* Неоднородные структуры объединяют данные разных типов. К ***неоднородным*** структурам отно­сят *записи.*
2. **По технологии обработки данных** БД подразделяют на централизованные и распределенные.

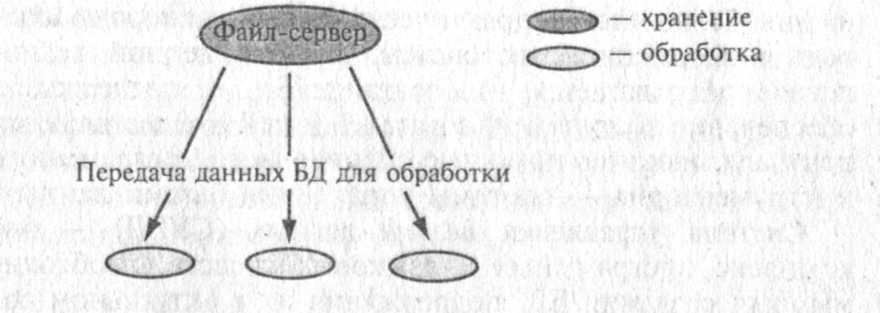
***Централизованная БД*** хранится в памяти одной вы­числительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределен­ный доступ к такой базе. Такой способ использования баз данных часто применяют в локальных сетях ПК.

***Распределенная БД*** состоит из нескольких (возможно, пересекающихся или даже дублирующих друг друга) час­тей, хранимых в различных ЭВМ вычислительной -сети.

Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

5. **По способу доступа** к данным различают БД с ло­кальным и удаленным (сетевым) доступом. Системы централизованных БД с сетевым доступом предполага­ют различные архитектуры подобных систем, такие, как файл-сервер и клиент-сервер.

***Файл-сервер.*** Архитектура систем БД с сетевым досту­пом предполагает выделение одной из машин сети в каче­стве центральной (сервер файлов). На такой машине хра­нится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих стан­ций, с помощью которых поддерживается доступ пользо­вательской системы к централизованной БД. Файлы БД в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производят обра­ботку. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать на рабочих » станциях локальные БД — для своих нужд. Концепция «файл-сервер» условно отображена на рис. 1.



**Рис. 1.** Схема обработки информации в БД по принципу «файл-сервер»

***Клиент-сервер.*** В соответствии с этой концепцией подразумевается, что, помимо хранения централизо­ванной БД, центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема об­работки данных. Запрос на данные, выдаваемый клиен­

том (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Специ­фикой архитектуры «клиент-сервер» является использова­ние *языка запросов SQL.* Концепция «клиент-сервер» ус­ловно изображена на рис. 2.



Рис. **2.** Схема обработки информации в БД по принципу «клиент-сервер»

**База данных, банк данных, система управления базой данных, администратор базы данных**

БД (в том или ином виде) — неотъемлемая часть ре­шения большинства практических задач, в первую оче­редь информационных систем. В современной техно­логии предполагается, что создание БД, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария — системы управления базами данных.

**Операции в** БД:

♦ ввод основных и дополнительных данных;

* изменение и корректировка уже имеющейся информации;
* создание новых структур хранения данных;
* организация взаимодействия с другими БД и прикладными программами;
* длительное хранение данных;

♦ передача информации (межкомпьютерный обмен);

♦ защита от несанкционированного доступа;

Средства защиты данных можно разделить на две группы: 1. различные способы запрещения доступа к данным; 2.средства скрытия источников данных;

♦ резервное копирование данных;

* сортировка информации (по алфавиту) обеспечивает универсальность организации операций над данными и удобный пользовательский интерфейс;
* просмотр данных;

♦ автоматическое обновление данных, т.е. достаточно один раз внести изменение, и оно автоматически будет отражено везде, где это необходимо.

**3. Система управления базами данных (СУБД)** — это комплекс программных и языковых средств, необходи­мых для создания БД, поддержания их в актуальном со­стоянии и организации поиска в них необходимой ин­формации.

СУБД обладает следующими рабочими характеристиками:

* **полнота.** Чем больше информации, тем более вероятность нахождения полезных данных;
* **структурность.** Чем лучше структурирована информация, тем она полезнее и ценнее;
* **удобство.** Достигается выполнением первых трех рабочих характеристик. Для работы с СУБД необходимо несколько типов языков:
* язык для написания программы;
* язык манипулирования данными, обеспечивающий интерфейс между программой и системой управления БД;
* язык описания данных.

Централизованный характер управления данными в БД предполагает необходимость существования некото­рого липа (группы лиц), на которое возложены функции администрирования данными, хранимыми в базе. Систе­мы управления базами данных выполняют две основные функции:

* хранение и ведение представления структурной ин­формации (данных);
* преобразование по некоторому запросу хранимого представления в структурную информацию.

***Процедура хранения информации*** в ПК состоит в том, чтобы сформировать и поддерживать структуру хранения данных в памяти компьютера.

***Процедура актуализации данных*** позволяет изменить значения данных, записанных в базе, либо дополнить оп­ределенный раздел, группу данных. Устаревшие данные могут быть удалены с помощью соответствующей опера­ции.

***Процедура извлечения данных*** необходима для пересыл­ки из БД необходимых сведений дтя преобразования, отображения или передачи по вычислительной сети.

Современные структуры хранения данных должны быть независимыми от программ, использующих эти дан­ные, и реализовывать принципы полноты и минимальной избыточности. Такие структуры и получили название БД. Процедуры создания структуры хранения (БД), актуали­зации, извлечения и удаления данных осуществляют с по­мощью специальных программ, называемых **системами управления базами данных.**

Хранение и передача данных тестю связаны между со­бой, для выполнения этих процедур используют сетевые информационные технологии. Программы, предназна­ченные только для хранения и передачи данных, носят название **информационных хранилищ** и представляют со­бой компьютеризованные архивы.

БД и СУБД являются частью *Банков данных*, которые хранят сведения из Самых различных областей человеческой деятельности: это библиотечное и банковское дело, образование и медицина, управление предприятием и государством, право, экология, транспорт, туризм и многое другое.

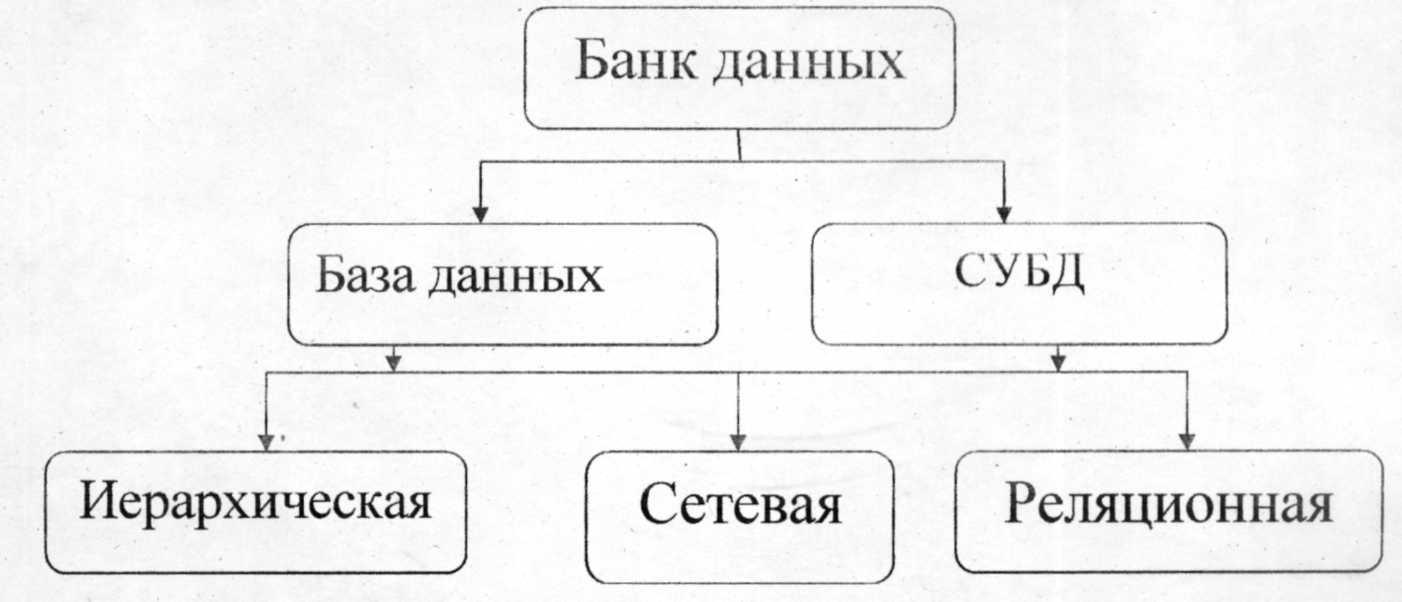


Рис.3. Структура банка данных

**Cреда базы данных**

Основная цель системы управления базами данных заключается в том, чтобы предложить пользователю абстрактное представление данных, скрыв конкретные особенности хранения и управления ими. Следовательно, отправной точкой при проектировании БД должно быть общее описание информационных потребностей пользователей, которые должны найти свое отражение в создаваемой базе данных.

Более того, поскольку база данных является общим ресурсом, то каждому пользователю может потребоваться свое, отличное от других представление о характеристиках информации, сохраняемой в базе данных. Для удовлетворения этих потребностей архитектура большинства современных СУБД в той или иной степени строится на базе так называемой трёхуровневой архитектуры.

Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления. Ниже перечислено несколько причин, по которым желательно выполнять такое разделение.

* Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое собственное представление о них. Каждый пользователь должен иметь возможность изменять свое представление о данных, причем это изменение не должно оказывать влияния на других пользователей.
* Пользователи не должны непосредственно иметь дело с подробностями физического хранения данных в базе
* Администратор базы данных должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские представления.
* Внутренняя структура базы данных не должна зависеть от таких изменений физических аспектов хранения информации, как переход на новое устройство хранения.
* АБД должен иметь возможность изменять концептуальную или глобальную структуру базы данных без какого-либо влияния на всех пользователей.

Уровень, на котором воспринимают данные пользователи, называется внешним уровнем (external level), тогда как СУБД и операционная система воспринимают данные на внутреннем уровне (internal level). Именно на внутреннем уровне данные реально сохраняются с использованием всех тех структур и файловой организации. Концептуальный уровень (conceptual level) представления данных предназначен для отображения внешнего уровня на внутренний и обеспечения необходимой независимости друг от друга.

  
  
**Рис. 5.6 Трехуровневая архитектура ANSI-SPARC**

**Внешний уровень.** Внешний уровень состоит из нескольких различных внешних представлений БД. Каждый пользователь имеет дело с представлением предметной области, выраженным в наиболее удобной для него форме. Внешнее представление содержит только те сущности, атрибуты и связи предметной области, которые интересны пользователю.

Помимо этого, различные представления могут поразному отображать одни и те же данные. Например, один пользователь может просматривать даты в формате (день, месяц, год), а другой - в формате (год, месяц, день). Некоторые представления могут включать производные или вычисляемые данные, которые не хранятся в базе данных как таковые, а создаются по мере надобности. Представления могут также включать комбинированные или производные данные из нескольких объектов.

**Концептуальный уровень.** Промежуточным уровнем в трехуровневой архитектуре является концептуальный уровень. Этот уровень содержит логическую структуру всей базы данных (с точки зрения АБД). Фактически, это полное представление требований к данным, которое не зависит от любых соображений относительно способа их хранения. На концептуальном уровне представлены следующие компоненты: все сущности, их атрибуты и связи; накладываемые на данные ограничения; семантическая информация о данных; информация о мерах обеспечения безопасности и поддержки целостности данных.

Концептуальный уровень поддерживает каждое внешнее представление, в том смысле, что любые доступные пользователю данные должны содержаться (или могут быть вычислены) на этом уровне. Однако этот уровень не содержит никаких сведений о методах хранения данных.

**Внутренний уровень.** Внутренний уровень описывает физическую реализацию базы данных и предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. Он содержит описание структур данных и организации отдельных файлов, используемых для хранения данных в запоминающих устройствах. На этом уровне осуществляется взаимодействие СУБД с методами доступа операционной системы с целью размещения данных на запоминающих устройствах, создания индексов, извлечения данных и т.д. На внутреннем уровне хранится следующая информация: сведения о распределении 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.

**Схемы, отображения и экземпляры.** Общее описание базы данных называется схемой базы данных. Существует три различных типа схем базы данных, которые определяются в соответствии с уровнями абстракции трехуровневой архитектуры, показанной на рис. 5.6. 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.

Концептуальная схема описывает все элементы данных и связи между ними, с указанием необходимых ограничений поддержки целостности данных. Для каждой базы данных имеется только одна концептуальная схема. На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных.

Она содержит определения хранимых записей, методы представления, описания полей данных, сведения об индексах и выбранных схемах хеширования. Для каждой базы данных существует только одна внутренняя схема.

СУБД отвечает за установление соответствия между этими тремя типами схем, а также за проверку их непротиворечивости. Иначе говоря, СУБД должна убедиться в том, что каждую внешнюю схему можно вывести на основе концептуальной схемы. Для установления соответствия между любыми внешней и внутренней схемами СУБД должна использовать информацию из концептуальной схемы. Концептуальная схема связана с внутренней схемой посредством отображения "концептуальный - внутренний". Оно позволяет СУБД найти фактическую запись или набор записей на физическом устройстве хранения, которые образуют логическую запись в концептуальной схеме, с учетом любых ограничений, установленных для выполняемых над данной логической записью операций. Оно также позволяет обнаружить любые различия в именах объектов, именах атрибутов, порядке следования атрибутов, их типах данных и т.д. Наконец, каждая внешняя схема связана с концептуальной схемой с помощью отображения "внешний - концептуальный". С его помощью СУБД может отображать имена пользовательского представления на соответствующую часть концептуальной схемы.

Важно различать описание базы данных и саму базу данных. Описанием базы данных является схема базы данных. Схема создается в процессе проектирования базы данных, причем предполагается, что она изменяется достаточно редко. Однако содержащаяся в базе данных информация может меняться часто - например, всякий раз при вставке о том или ином объекте. Совокупность информации, хранящейся в базе данных в любой определенный момент времени, называется состоянием базы данных. Следовательно, одной и той же схеме базы данных может соответствовать множество ее различных состояний. Схема базы данных иногда называется содержанием базы данных, а ее состояние - детализацией.

***Независимость от данных.*** Основным назначением трехуровневой архитектуры является обеспечение независимости от данных, которая означает, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

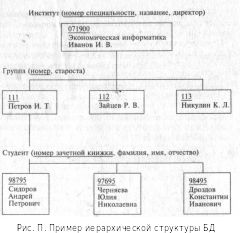
*Логическая независимость* от данных означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему. Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без внесения изменений в уже существующие внешние схемы или переписывания прикладных программ.

*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. Пользователем могут быть замечены изменения только в общей производительности системы. На рис. 5.7 показано место перечисленных выше типов независимости от данных в трехуровневой архитектуре СУБД.

Принятое в архитектуре ANSI-SPARC двухэтапное отображение может сказываться на эффективности работы, но при этом оно обеспечивает более высокую независимость от данных.

  
  
**Рис. 5.7 Реализация независимости от данных в трехуровневой архитектуре ANSI-SPARC**

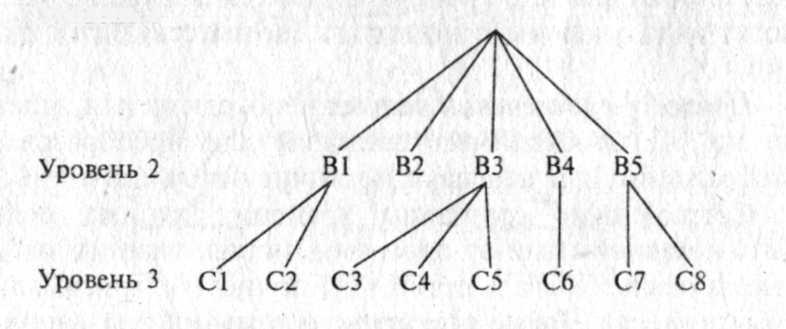
**Уровни представления данных**

Я  
дром любой БД является модель данных - множе­ство структур данных, ограничений целостности и опе­раций манипулирования данными. **Модель данных** — это совокупность структур данных и операций их обработки. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

СУБД основывается на использовании иерархической сетевой или реляционной моделей, на комбинации эти^ моделей или на некотором их подмножестве.

**Иерархическая модель данных.** Иерархическая струкч тура — это совокупность элементов, связанных межд^, собой по определенным правилам. Объекты, связанны^ иерархическими отношениями, образуют ориентирован ный граф (перевернутое дерево), вид которого представ-лен на рис. 10.

Уровень 1 . А



**Рис. 10.** Графическое изображение иерархической структуры БД

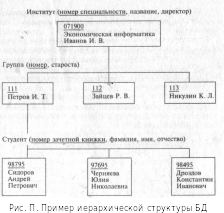
К основным понятиям иерархической структуры отно­сятся: уровень, узел (элемент), связь.

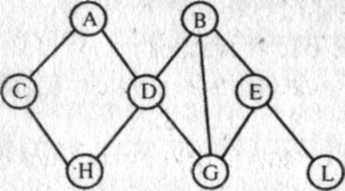
***Узел —*** это совокупность атрибутов данных, описываю-щих некоторый объект. На схеме иерархического дереву узлы представляются вершинами графа. Каждый узел ***щ*** более низком уровне связан только с одним узлом, нахо­дящимся на более высоком уровне. Иерархическое дерево имеет только одну вершину (корень дерева), не подчинен­ную никакой другой вершине и находящуюся на само^ц верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и т. д. уровнях. Количество деревьев в БД определяется числом корневых записей.

К каждой записи БД существует только один (иерархи­ческий) путь от корневой записи. Например, как видно из рис. 10, для записи С4 путь проходит через записи А и ВЗ.

Из приведенного на рис. 11 примера очевидно, что иерархическая структура правомерна, так как каждый сту­дент учится в определенной (только одной) группе, кото­рая относится к определенному (только одному) институту.

**Сетевая модель данных.** В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый эле­мент может быть связан с любым другим элементом.

П  
римером сложной сетевой структуры может служить структура БД, содержащей сведения о работах, студентов учебных заведений. Возможно участие одного студента в нескольких разработках, а также участие нескольких сту­дентов в разработке одной темы. Графическое изображе­ние примера, состоящего только из двух типов записей, показано на рис. 12 и рис. 13. Единственное отношение представляет собой сложную связь между записями в обо­их направлениях.

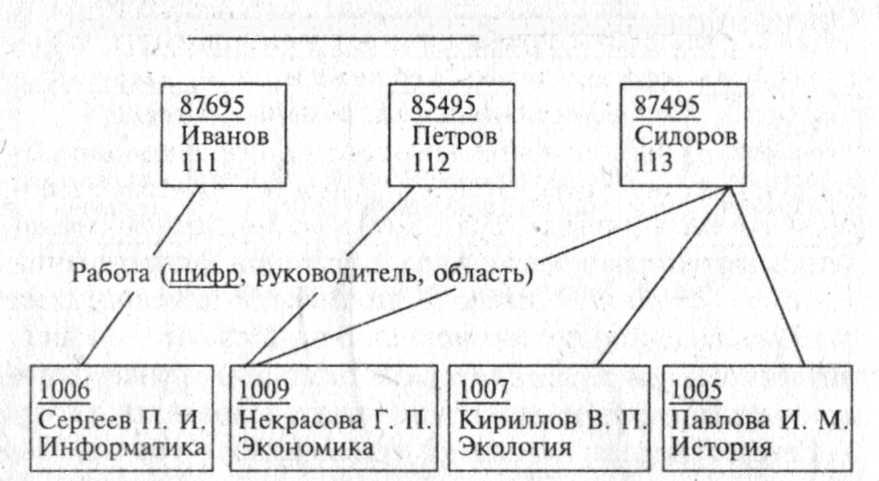


**Реляционная модель данных.**

Понятие «реляционный» (от англ. *relation* — отношение) связано с разработками извест­ного американского специалис­та в области систем баз данных **Рис. 12.** Графическое Е. Кодда. Реляционные модели изображение сетевой характеризуются простотой стру-структуры в виде графа ктуры данных, удобным для

пользователя табличным пред­ставлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Студент (номер зачетной книжки, фамилия, группа)



**Рис. 13.** Пример сетевой структуры БД

Реляционная модель ориентирована на организацию данных в виде двухмерных таблиц. Каждая ***реляционная таблица*** представляет собой двухмерный массив и обла­дает следующими свойствами:

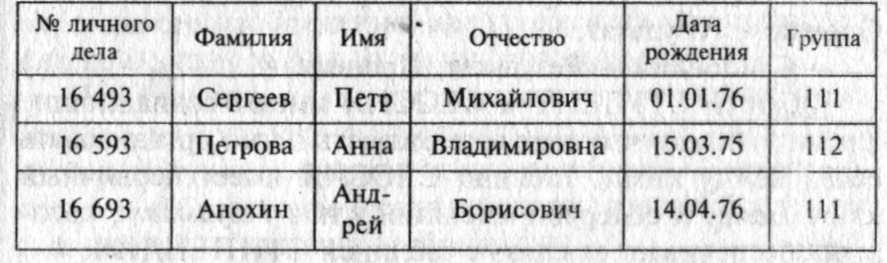
— каждый элемент таблицы — один элемент данных;

— все столбцы в таблице однородны, т. е. все элемен­ты в столбце имеют одинаковые тип (числовой, символь­ный и т. д.) и длину;

* каждый столбец имеет уникальное имя;
* одинаковые строки в таблице отсутствуют;

— порядок следования строк и столбцов может быть произвольным.

В качестве примера можно представить информапию о студентах вуза в виде *реляционной таблицы:*

**

Здесь отношения представлены в виде таблицы, стро­ки которой соответствуют ***кортежам,*** или ***записям,*** а столбцы — атрибутам отношений, доменам, ***полям.***

Поле, каждое значение которого однозначно опреде­ляет соответствующую запись, называется ***простым клю­чом*** (ключевым полем). Если записи однозначно опреде­ляются значениями нескольких полей, то такая таблица БД имеет ***составной ключ.*** В примере, показанном на рис. 14, ключевым полем таблицы является «Номер лич­ного дела».



**Рис. 14.** Пример реляционной модели

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй табли­цы (возможно совпадение ключей), в противном случае

нужно ввести в структуру первой таблицы *внешний ключ —* ключ второй таблицы. На рис. 14 показана реляционная модель, построенная на основе отношений «Студент, Сессия, Стипендия», где:

* Студент — Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа;
* Сессия — Номер, Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат;
* Стипендия — Результат, Процент.

Таблицы СТУДЕНТ и СЕССИЯ имеют совпадающие ключи *(Номер),* что дает возможность легко организовать связь между ними. Таблица СЕССИЯ имеет первичный ключ *Номер* и содержит внешний ключ *Результат,* кото­рый обеспечивает ее связь с таблицей СТИПЕНДИЯ.

**База данных** — средство организации хранения и управления большим количеством упорядоченной разнородной информации.

Базу данных можно представить в виде таблицы с конечным числом столбцов и неопределённым числом строк. Примером базы данных может служить классный журнал или итоговая ведомость, куда заносятся оценки за четверть. Количество столбцов ограниченно — их столько, сколько предметов изучается, а количество учащихся может изменяться. Пустая ведомость — это база данных, из которой удалены все записи, но остаются названия полей базы, то есть её структура. При создании базы данных необходимо сначала разработать её структуру, а затем заполнить её информацией — эти функции и выполняются с помощью СУБД. Вторым этапом является ввод и редактирование записей в таблицу. БД считается созданной, даже если она пустая.

**Столбцы** в базе данных называют **полями,** а **строки — записями.** Каждое **поле** имеет своё **имя** и содержит отдельный элемент информации. Для каждого поля необходимо указывать его **имя, тип данных, размер.** От типа и размера поля зависит скорость доступа к БД и объём файла. **Тип данных** поля определяется значениями, которые предполагается вводить в поле.

Основные объекты окна БД имеют следующее назначение:

• **таблица** — основное средство для хранения информации в БД;

• **запрос** — это инструмент для извлечения необходимой информации из исходных таблиц и представления ее в удобной форме;

• **форма** — это основное средство для ввода данных, управления СУБД и вывода результатов на экран монитора;

• **отчёт** — это специальное средство для формирования выходных до­кументов и вывода их на принтер;

• **макросы** в Access представляют собой совокупность внутренних команд, предназначенных для автоматизации работы с БД;

• **модули** являются программами, создаваемыми средствами языка VBA, и похожи на макросы в Word и Excel.

**Работа с СУБД Access.**

СУБД Access создаёт и обрабатывает реляционные базы данных, то есть она позволяет хранить данные не в одной, а в нескольких таблицах и устанавливать связь между ними. Такие таблицы называются связанными, т.е. объединёнными в единую базу. Для задания связи таблицы должны иметь поля с одинаковым типом данных. Связь между таблицами устанавливает отношение между совпадающими значениями в этих полях. Такая организация позволяет уменьшить избыточность хранимых данных, упрощает их ввод, удаление, поиск.

Для установления связи между таблицами необходимо, чтобы:

• связываемые поля имели одинаковый тип данных. Исключение составляет поле-счётчик, так как поле-счётчик может быть связано с числовым полем, имеющим размер «длинное целое».

• таблицы хранились внутри одной БД.

• главная таблица связывалась с подчинённой по ключу.

Для обеспечения целостности данных для связанных таблиц нужно помнить, что:

• в подчиненную таблицу не может быть добавлена запись с несуществующим в главной таблице ключом связи;

• в главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчиненной таблице;

• изменение значений ключа связи главной таблицы должно приводить к изменению соответствующих значений в записях подчиненной таблицы.

Та таблица, от которой идет связь, называется главной, а таблица, к которой эта связь ведёт, — подчинённой.

В каждой таблице должно быть уникальное поле, с помощью которого можно связать таблицы между собой. Такое поле называется полем первичного ключа, или первичным ключом.

Для того чтобы понять принципы разработки реляционных баз данных, требуется дать определения различных типов реляционных ключей и таблиц:

• **Базовая таблица.** В реляционной базе данных базовой таблицей называется таблица, которая включает один или несколько столбцов свойств объекта и содержит **первичный ключ,** который однозначно определяет этот объект. Более того, базовая таблица должна содержать первичный ключ. Базовые таблицы часто называют **первичными,** поскольку они имеют **первичный ключ.**

• **Промежуточная таблица.** Таблица, не являющаяся базовой (т. к. она не объединяет свойства объекта или не содержит поле первичного ключа), которая используется для обеспечения связей между другими таблицами, называется **таблицей отношений.** Ключевые поля в таблицах отношений должны быть**внешними ключами,** связанными с первичными ключами базовой таблицы. То есть, таблица отношений состоит только из внешних ключей и не содержит независимых элементов данных.

• **Первичный ключ.** Первичный ключ состоит из набора значений, которые однозначно определяют запись базовой таблицы. Любому значению первичного ключа должна соответствовать одна и только одна строка таблицы. Первичный ключ включает одно поле только в том случае, если это поле не содержит повторяющихся значений.

• **Составные ключи.** Если для выполнения условий, накладываемых на значения первичного ключа, заданный ключ включает несколько полей таблицы, то тогда он называется **составным.**

• **Внешние ключи.** Внешний ключ — это столбец, значения которого соответствуют значениям первичного ключа другой связанной таб­лицы.

В программе предусмотрены пять возможностей создания таблицы:

• **Импорт таблиц** из другой базы. В зависимости от обстоятельств из импортируемой таблицы может поступить структура полей, их на­звания и свойства, а также и содержимое базы. Необходимые прав­ки вносят вручную.

• Режим **Связь с таблицами** применяется в тех случаях, когда речь идет о чужой таблице, которая находится на удаленном сервере и которую нельзя импортировать целиком. Это напоминает подклю­чение к таблице для совместного использования её данных.

• **Мастер таблиц.** Это программа, ускоряющая создание структуры таблицы. Мастер задает ряд вопросов и, руководствуясь получен­ными ответами, создает структуру таблицы автоматически.

• **Режим таблицы** открывает заготовку, в которой все поля имеют формальные имена: Полег, Поле2... и т. д. — и один стандартный текстовый тип. Такую таблицу можно сразу наполнять информацией.

• **Конструктор** предоставляет возможность одновременно задавать поля будущей таблицы и устанавливать свойства этих полей.

Мощным средством обработки данных, хранимых в таблицах Access, являются запросы. С их помощью можно просматривать, анализировать и изменять данные из нескольких таблиц, а также можно использовать запросы в качестве источника данных для форм и отчётов. Запросы позволяют вычислять итоговые значения и выводить их в компактном формате, подобном формату электронной таблицы, а также выполнять вычисления над группами записей.

В Access можно создавать следующие типы запросов:

**Запрос на выборку.** При его выполнении данные, удовлетворяющие условиям отбора, выбираются из одной или из нескольких таблиц и выводятся в определённом порядке. Простые запросы на выборку практически не отличаются от фильтров, которые можно сохранять как запросы. Этот запрос можно использовать, чтобы сгруппировать записи для вычисления сумм, средних значений, пересчёта и других действий.

**Запрос с параметрами.** Это запрос, при выполнении которого в его диалоговом окне пользователю выдается приглашение ввести данные, на основе которых будет выполняться запрос.

**Перекрёстный запрос.** Перекрёстные запросы предназначены для группирования данных и представления их в компактном виде. Позволяют представить большой объем данных в виде, удобном для восприятия,

анализа, сравнения. Могут использоваться в качестве базового при со­здании отчёта. Отчёты позволяют выбирать из базы данных нужную информацию, оформить её в виде документа и распечатать. Источником данных может быть таблица, запрос или несколько взаимосвязанных таблиц. Отчеты и формы похожи, разница в том, что, в отличие от форм, отчёты не предназначены для ввода и корректировки данных. Отчёты состоят из разделов, подобных разделам форм. В процессе конструирования отчета формируется состав и содержимое разделов отчёта, размещение в нём значений, выводимых из полей связанных таблиц баз данных, формируются заголовки, размещаются вычисляемые поля. Отчёт позволяет сгруппировать данные по нескольким уровням, для каждого из которых производится вычисление итогов, определяются заголовки и примечания.