ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.07.02 Базы данных

Специальность 38.05.01 Экономическая безопасность

Специализация Экономико-правовое обеспечение экономической безопасности

Форма обучения заочная

1. КОНСПЕКТ ЛЕКЦИЙ

1. 1 Лекция №1 (1 час).
Тема: «История создания дисциплины»

1.1.1 Вопросы лекции:

1. Краткая характеристика дисциплины
2. История создания дисциплины

1.1.2 Краткое содержание вопросов:

1. Краткая характеристика дисциплины.

Цель преподавания дисциплины - формирование представления о роли и месте знаний по базам данных при практическом использовании их в своей профессиональной деятельности.

Дисциплина «Базы данных» относится к профессиональному циклу, вариативная части.

Компетенции обучающегося, формируемые в результате освоения дисциплины

1. Способность применять математический инструментарий для решения экономических задач (ОК-15).

2. Способность выбирать инструментальные средства для обработки финансовой, бухгалтерской и иной экономической информации и обосновывать свой выбор (ПК-32).

В результате освоения дисциплины обучающийся должен:

Знать:

- базы данных и системы управления базами данных для информационных систем различного назначения.
- основные положений концепции баз данных и принципов построения баз данных;
- классификацию и модели данных;
- современные системы управления базами данных и их место в системах обработки данных.

Уметь:

- разрабатывать инфологические и датологические схемы баз данных;
- создавать базы данных;
- реализовывать простые информационные технологии в экранном интерфейсе современных систем управления базами данных;
- применять методики проектирования баз данных для конкретных предметных областей;
- эффективно работать индивидуально при разработке баз данных;
- эффективно работать в качестве члена команды по разработке программных средств.

Владеть:

- методами описания схем баз данных;
- навыками работы в качестве члена группы при разработке баз данных;
- способностью брать на себя ответственность за результаты работы по разработке программных средств.

Взаимосвязь планируемых результатов обучения по дисциплине (знания, умения, навыки и (или) опыт деятельности) и планируемых результатов освоения образовательной программы (компетенций обучающегося):

- ОК-15: способность применять математический инструментарий для решения экономических задач:

-Знания:

Этап 1. Классификацию и модели данных.

Этап 2. Базы данных и системы управления базами данных для информационных систем различного назначения.

- Умения:
Этап 1: Применять методики проектирования баз данных для конкретных предметных областей.
Этап 2: Реализовывать простые информационные технологии в экранном интерфейсе современных систем управления базами данных; эффективно работать индивидуально при разработке баз данных.
- Навыки и (или) опыт деятельности:
Этап 1: Навыками самостоятельного овладения новыми знаниями.
Этап 2: Методами описания схем баз данных.

- ПК-32: способность выбирать инструментальные средства для обработки финансовой, бухгалтерской и иной экономической информации и обосновывать свой выбор
-Знания:
Этап 1. Основные положений концепции баз данных и принципов построения баз данных.
Этап 2: Современные системы управления базами данных и их место в системах обработки данных.
- Умения:
Этап 1: Создавать базы данных.
Этап 2: Разрабатывать инфологические и датологические схемы баз данных; эффективно работать в качестве члена команды по разработке программных средств.
- Навыки и (или) опыт деятельности:
Этап 1: Навыками работы в качестве члена группы при разработке баз данных.
Этап 2: Способностью брать на себя ответственность за результаты работы по разработке программных средств

Общая трудоемкость дисциплины «Базы данных» составляет 3 ЗЕ (108 часов.

2. История создания дисциплины

В истории вычислительной техники можно проследить развитие двух основных областей ее использования. Первая область — применение вычислительной техники для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную. Развитие этой области способствовало интенсификации методов численного решения сложных математических задач, появлению языков программирования, ориентированных на удобную запись численных алгоритмов, становлению обратной связи с разработчиками новых архитектур ЭВМ. Характерной особенностью данной области применения вычислительной техники является наличие сложных алгоритмов обработки, которые применяются к простым по структуре данным, объем которых сравнительно невелик.

Вторая область, которая непосредственно относится к нашей теме, — это использование средств вычислительной техники в автоматических или автоматизированных информационных системах. Информационная система представляет собой программно-аппаратный комплекс, обеспечивающий выполнение следующих функций:

· надежное хранение информации в памяти компьютера;

· выполнение специфических для данного приложения преобразований информации и вычислений;

· предоставление пользователям удобного и легко осваиваемого интерфейса.

Обычно такие системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

Вторая область использования вычислительной техники возникла несколько позже первой. Это связано с тем, что на заре вычислительной техники возможности компьютеров по хранению информации были очень ограниченными. Говорить о надежном и долговременном

хранении информации можно только при наличии запоминающих устройств, сохраняющих информацию после выключения электрического питания. Оперативная (основная) память компьютеров этим свойством обычно не обладает. В первых компьютерах использовались два вида устройств внешней памяти — магнитные ленты и барабаны. Емкость магнитных лент была достаточно велика, но по своей физической природе они обеспечивали последовательный доступ к данным. Магнитные же барабаны (они ближе всего к современным магнитным дискам с фиксированными головками) давали возможность произвольного доступа к данным, но имели ограниченный объем хранимой информации.

Эти ограничения не являлись слишком существенными для чисто численных расчетов. Даже если программа должна обработать (или произвести) большой объем информации, при программировании можно продумать расположение этой информации во внешней памяти (например, на последовательной магнитной ленте), обеспечивающее эффективное выполнение этой программы. Однако в информационных системах совокупность взаимосвязанных информационных объектов фактически отражает модель объектов реального мира. А потребность пользователей в информации, адекватно отражающей состояние реальных объектов, требует сравнительно быстрой реакции системы на их запросы. И в этом случае наличие сравнительно медленных устройств хранения данных, к которым относятся магнитные ленты и барабаны, было недостаточным.

Можно предположить, что именно требования нечисловых приложений вызвали появление съемных магнитных дисков с подвижными головками, что явилось революцией в истории вычислительной техники. Эти устройства внешней памяти обладали существенно большей емкостью, чем магнитные барабаны, обеспечивали удовлетворительную скорость доступа к данным в режиме произвольной выборки, а возможность смены дискового пакета на устройстве позволяла иметь практически неограниченный архив данных.

С появлением магнитных дисков началась история систем управления данными во внешней памяти. До этого каждая прикладная программа, которой требовалось хранить данные во внешней памяти, сама определяла расположение каждой порции данных на магнитной ленте или барабане и выполняла обмены между оперативной памятью и устройствами внешней памяти с помощью программно-аппаратных средств низкого уровня (машинных команд или вызовов соответствующих программ операционной системы). Такой режим работы не позволяет или очень затрудняет поддержание на одном внешнем носителе нескольких архивов долговременно хранимой информации. Кроме того, каждой прикладной программе приходилось решать проблемы именования частей данных и структуризации данных во внешней памяти.

1. 2 Лекция №2 (1 час).
Тема: «Система баз данных»

1.2.1 Вопросы лекции:
1. Определение и назначение баз данных
2. Файловые системы хранения данных
3. Области применения баз данных

1.2.2 Краткое содержание вопросов:
1. Определение и назначение баз данных.

С самого начала развития вычислительной техники образовались два основных направления ее использования.

Первое направление — применение вычислительной техники для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную.

Второе направление — это использование средств вычислительной техники в автоматических или автоматизированных информационных системах. В самом широком смысле информационная система представляет собой программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера,

выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. Обычно объемы информации, с которыми приходится иметь дело таким системам, достаточно велики, а сама информация имеет достаточно сложную структуру. Классическими примерами информационных систем являются банковские системы, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

Второе направление возникло несколько позже первого. Это связано с тем, что на заре вычислительной техники компьютеры обладали ограниченными возможностями. Надежное и долговременное хранение информации возможно только при наличии запоминающих устройств, сохраняющих информацию после выключения электрического питания. Оперативная память этим свойством обычно не обладает. Используемые в ранних ЭВМ два вида устройств внешней памяти, магнитные ленты и барабаны были несовершенными. Емкость магнитных лент была достаточно велика, но по своей физической природе они обеспечивали последовательный доступ к данным. Магнитные барабаны давали возможность произвольного доступа к данным, но были ограниченного размера. Появление соответствующих носителей данных, в первую очередь, жестких дисков, дало толчок к работам по созданию информационных компьютерных систем.

Основу любой информационной системы составляет база данных — это набор данных, которые организованы специальным образом.

В настоящее время действует Закон «О правовой охране программ для электронных вычислительных машин и баз данных» № 3523-1 от 23.09.92. В этом законе дается следующее определение базы данных: «**База данных** — это объективная форма представления и организации совокупности данных (например, статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ».

В основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются информационные системы (ИС). Автоматизированными называют ИС, в которых применяют технические средства, в частности ЭВМ. Большинство существующих ИС являются автоматизированными, поэтому для краткости просто будем называть их ИС.

В широком понимании под определение ИС подпадает любая система обработки информации. По области применения ИС можно разделить на системы, используемые в производстве, образовании, здравоохранении, науке, военном деле, социальной сфере, торговле и других отраслях. По целевой функции ИС можно условно разделить на следующие основные категории: управляющие, информационно-справочные, поддержки принятия решений.

Заметим, что иногда используется более узкая трактовка понятия ИС как совокупности аппаратно-программных средств, задействованных для решения некоторой прикладной задачи. В организации, например, могут существовать информационные системы, на которых соответственно возложены следующие задачи: учет кадров и материально-технических средств, расчет с поставщиками и заказчиками, бухгалтерский учет и т. п.

Банк данных является разновидностью ИС, в которой реализованы функции централизованного хранения и накопления обрабатываемой информации, организованной в одну или несколько баз данных. Банк данных (БнД) в общем случае состоит из следующих компонентов: базы (нескольких баз) данных, системы управления базами данных, словаря данных, администратора, вычислительной системы и обслуживающего персонала. Вкратце рассмотрим названные компоненты и некоторые связанные с ними важные понятия.

База данных (БД) представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Логическую структуру хранимых в базе данных называют моделью представления данных. К основным моделям представления данных (моделям данных) относятся следующие: иерархическая, сетевая, реляционная, постреляционная, многомерная и объектно-ориентированная (см. раздел 2).

Система управления базами данных (СУБД) - это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования

БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Одними из первых СУБД являются следующие системы: IMS (IBM, 1968 г.), IDMS (Cullinet, 1971 г.), ADABAS (Software AG, 1969 г.) и ИНЭС (ВНИИСИ АН СССР, 1976 г.). Количество современных систем управления базами данных исчисляется тысячами.

Приложение представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи. Нами рассматриваются приложения, использующие БД. Приложения могут создаваться в среде или вне среды СУБД - с помощью системы программирования, использующей средства доступа к БД, к примеру, Delphi или C++ Builder. Приложения, разработанные в среде СУБД, часто называют приложениями СУБД, а приложения, разработанные вне СУБД, - внешними приложениями.

Для работы с базой данных зачастую достаточно средств СУБД и не нужно использовать приложения, создание которых требует программирования. Приложения разрабатывают главным образом в случаях, когда требуется обеспечить удобство работы с БД неквалифицированным пользователям или интерфейс СУБД не устраивает пользователей.

Словарь данных (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т.п. Функционально СД присутствует во всех БнД, но не всегда выполняющий эти функции компонент имеет именно такое название. Чаще всего функции СД выполняются СУБД и вызываются из основного меню системы или реализуются с помощью ее утилит.

Администратор базы данных (АБД) есть лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение. В процессе эксплуатации АБД обычно следит за функционированием информационной системы, обеспечивает защиту от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой в БД информации. Для однопользовательских информационных систем функции АБД обычно возлагаются на лиц, непосредственно работающих с приложением БД.

Вычислительной сети АБД, как правило, взаимодействует с администратором сети. В обязанности последнего входят контроль за функционированием аппаратно-программных средств сети, реконфигурация сети, восстановление программного обеспечения после сбоев и отказов оборудования, профилактические мероприятия и обеспечение разграничения доступа.

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

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

2. Файловые системы хранения данных.

Фа́йловая систе́ма (англ. file system) — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах, а также в другом электронном оборудовании: цифровых фотоаппаратах, мобильных телефонах и т. п. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов и (каталогов), максимальный возможный размер файла и

раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

Файловая система связывает носитель информации с одной стороны и API для доступа к файлам — с другой. Когда прикладная программа обращается к файлу, она не имеет никакого представления о том, каким образом расположена информация в конкретном файле, так же, как и на каком физическом типе носителя (CD, жёстком диске, магнитной ленте, блоке флеш-памяти или другом) он записан. Всё, что знает программа — это имя файла, его размер и атрибуты. Эти данные она получает от драйвера файловой системы. Именно файловая система устанавливает, где и как будет записан файл на физическом носителе (например, жёстком диске).

С точки зрения операционной системы (ОС), весь диск представляет собой набор кластеров (как правило, размером 512 байт и больше). Драйверы файловой системы организуют кластеры в файлы и каталоги (реально являющиеся файлами, содержащими список файлов в этом каталоге). Эти же драйверы отслеживают, какие из кластеров в настоящее время используются, какие свободны, какие помечены как неисправные.

Однако файловая система не обязательно напрямую связана с физическим носителем информации. Существуют виртуальные файловые системы, а также сетевые файловые системы, которые являются лишь способом доступа к файлам, находящимся на удалённом компьютере.

Практически всегда файлы на дисках объединяются в каталоги.

В простейшем случае все файлы на данном диске хранятся в одном каталоге. Такая одноуровневая схема использовалась в CP/M и в первой версии MS-DOS 1.0. Иерархическая файловая система со вложенными друг в друга каталогами впервые появилась в Multics, затем в UNIX.

Wiki.txt
Tornado.jpg
Notepad.exe
(Одноуровневая файловая система)

Каталоги на разных дисках могут образовывать несколько отдельных деревьев, как в DOS/Windows, или же объединяться в одно дерево, общее для всех дисков, как в UNIX-подобных системах.

C:
 \Program files
  \CDEx
   \CDEx.exe
   \CDEx.hlp
   \mppenc.exe
 \Мои документы
  \Wiki.txt
  \Tornado.jpg
D:
 \Music
  \ABBA
   \1974 Waterloo
   \1976 Arrival
    \Money, Money, Money.ogg
   \1977 The Album
(Иерархическая файловая система Windows/DOS)

В UNIX существует только один корневой каталог, а все остальные файлы и каталоги вложены в него. Чтобы получить доступ к файлам и каталогам на каком-нибудь диске, необходимо смонтировать этот диск командой mount. Например, чтобы открыть файлы на CD, нужно, говоря простым языком, сказать операционной системе: «возьми файловую систему на этом компакт-диске и покажи её в каталоге /mnt/cdrom». Все файлы и каталоги, находящиеся на CD, появятся в этом каталоге /mnt/cdrom, который называется точкой

монтирования (англ. mount point). В большинстве UNIX-подобных систем съёмные диски (дискеты и CD), флеш-накопители и другие внешние устройства хранения данных монтируют в каталог /mnt, /mount или /media. Unix и UNIX-подобные операционные системы также позволяют автоматически монтировать диски при загрузке операционной системы.

```
/
  /usr
    /bin
      /arch
      /ls
      /raw
    /lib
      /libhistory.so.5.2
      /libgpm.so.1
  /home
    /lost+found
      /host.sh
    /guest
      /Pictures
        /example.png
      /Video
        /matrix.avi
        /news
          /lost_ship.mpeg
```

(Иерархическая файловая система в Unix и UNIX-подобных операционных системах)

Обратите внимание на использование слешей в файловых системах Windows, UNIX и UNIX-подобных операционных системах (В Windows используется обратный слеш «\», а в UNIX и UNIX-подобных операционных системах простой слеш «/»)

Кроме того, следует отметить, что вышеописанная система позволяет монтировать не только файловые системы физических устройств, но и отдельные каталоги (параметр --bind) или, например, образ ISO (опция loop). Такие надстройки, как FUSE, позволяют также монтировать, например, целый каталог на FTP и ещё очень большое количество различных ресурсов.

Ещё более сложная структура применяется в NTFS и HFS. В этих файловых системах каждый файл представляет собой набор атрибутов. Атрибутами считаются не только традиционные только для чтения, системный, но и имя файла, размер и даже содержимое. Таким образом, для NTFS и HFS то, что хранится в файле, — это всего лишь один из его атрибутов.

Если следовать этой логике, один файл может содержать несколько вариантов содержимого. Таким образом, в одном файле можно хранить несколько версий одного документа, а также дополнительные данные (значок файла, связанная с файлом программа). Такая организация типична для HFS на Macintosh.

По предназначению файловые системы можно классифицировать на нижеследующие категории:

- Для носителей с произвольным доступом (например, жёсткий диск): FAT32, HPFS, ext2 и др. Поскольку доступ к дискам в несколько раз медленнее, чем доступ к оперативной памяти, для прироста производительности во многих файловых системах применяется асинхронная запись изменений на диск. Для этого применяется либо журналирование, например в ext3, ReiserFS, JFS, NTFS, XFS, либо механизм soft updates и др. Журналирование широко распространено в Linux, применяется в NTFS. Soft updates — в BSD системах.
- Для носителей с последовательным доступом (например, магнитные ленты): QIC и др.
- Для оптических носителей — CD и DVD: ISO9660, HFS, UDF и др.
- Виртуальные файловые системы: AEFS и др.

- Сетевые файловые системы: NFS, CIFS, SSHFS, GmailFS и др.

- Для флэш-памяти: YAFFS, ExtremeFFS, exFAT.

Немного выпадают из общей классификации специализированные файловые системы: ZFS (собственно файловой системой является только часть ZFS), VMFS (т. н. кластерная файловая система, которая предназначена для хранения других файловых систем) и др.

Основные функции любой файловой системы нацелены на решение следующих задач:

- именование файлов;

- программный интерфейс работы с файлами для приложений;

- отображения логической модели файловой системы на физическую организацию хранилища данных;

- организация устойчивости файловой системы к сбоям питания, ошибкам аппаратных и программных средств;

- содержание параметров файла, необходимых для правильного его взаимодействия с другими объектами системы (ядро, приложения и пр.).

В многопользовательских системах появляется ещё одна задача: защита файлов одного пользователя от несанкционированного доступа другого пользователя, а также обеспечение совместной работы с файлами, к примеру, при открытии файла одним из пользователей, для других этот же файл временно будет доступен в режиме «только чтение».

3. Области применения баз данных.

Автоматизированные информационные системы (АИС), основу которых составляют базы данных, появились в 60-х годах XX века в военной промышленности и бизнесе — там, где были накоплены значительные объемы полезных данных. Первоначально АИС были ориентированы лишь на работу с информацией фактического характера — числовыми или текстовыми характеристиками объектов. Затем по мере развития техники появилась возможность обработки текстовой информации на естественном языке.

Принципы хранения разных видов информации в АИС аналогичны, но алгоритмы ее обработки определяются характером ин формационных ресурсов. Соответственно различают два класса АИС: документальные и фактографические.

Документальные АИС служат для работы с документами на естественном языке. Наиболее распространенный тип документальных АИС — информационно-поисковые системы, предназначенные для накопления и подбора документов, удовлетворяющих заданным критериям. Эти системы могут выполнять просмотр и подборку монографий, публикаций в периодике, сообщений пресс- агентств, текстов законодательных актов и т.д.

Фактографические АИС оперируют фактическими сведениями, представленными в формализованном виде, и используются для решения задач обработки данных.

Обработка данных — специальный класс решаемых на ЭВМ задач, связанных с вводом, хранением, сортировкой, отбором и группировкой записей данных однородной структуры. К задачам этого класса относятся: учет товаров в магазинах и на складах; начисление зарплаты; управление производством, финансами, телекоммуникациями и т. п.

Различают фактографические АИС оперативной обработки данных, подразумевающие быстрое обслуживание относительно простых запросов от большого числа пользователей, и фактографические АИС аналитической обработки, ориентированные на выполнение сложных запросов, требующих проведения статистической обработки исторических (накопленных за некоторый промежуток времени) данных, моделирования процессов предметной области и прогнозирования развития этих процессов.

Таким образом, АИС применяются в следующих областях:

• организация хранилищ данных;

• системы анализа данных;

• системы принятия решений;

• мобильные и персональные базы данных;

• географические базы данных;

• мультимедиа базы данных;

• распределенные информационные системы.

Каждая информационная система в зависимости от назначения имеет дело с той или иной частью конкретного мира, которую принято называть ее предметной областью. Анализ предмет ной области является необходимым начальным этапом разработки любой информационной системы. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных. Предметная область конкретной информационной системы рассматривается, прежде все го, как некоторая совокупность реальных объектов, которые представляют интерес для ее пользователей. Примерами объектов предметной области могут служить персональные ЭВМ, программные продукты и их пользователи. Каждый из этих объектов обладает определенным набором свойств (атрибутов). Так, например, компьютер характеризуется названием фирмы-производителя, идёнтификатором модели, типом микропроцессора, объемом оперативной и внешней памяти, типом графической карты и т. д.

Информационный объект это описание некоторой сущности предметной области, т. е. реального объекта, процесса, явления или события. Информационный объект (сущность) образуется совокупностью логически взаимосвязанных атрибутов (свойств), представляющих собой качественные и количественные характеристики объекта (сущности).

Между объектами предметной области могут существовать связи, имеющие различный содержательный смысл. Эти связи могут быть обязательными или факультативными (необязательными).

Если вновь порожденный объект оказывается по необходимости связанным с каким-либо объектом предметной области, то между этими двумя объектами существует обязательная связь. В противном случае связь является факультативной. Например, обязательная связь Замещает существует между двумя объектами СОТРУДНИК и ДОЛЖНОСТЬ в предметной области кадровой информационной системы, т. е. каждый принимаемый в организацию сотрудник зачисляется на какую-либо должность и не может быть сотрудника, не замещающего какой-либо должности. В то же время связь Замещает между типами объектов СОТРУДНИК и ДОЛЖНОСТЬ является факультативной, поскольку могут существовать вакантные должности. Совокупность объектов предметной области и связей между ними характеризует структуру предметной области. Множество объектов предметной области, значения атрибутов объектов и связи между ними могут изменяться во времени. Изменения могут сводиться к появлению новых или исключению из рассмотрения некоторых существующих объектов в предметной области, установлению новых или разрушению существующих связей между ними. Следовательно, с каждым моментом времени можно сопоставить некоторое состояние предметной области.

Информационно-логическая модель (ИЛМ) — это совокупность информационных объектов (сущностей) предметной области и связей между ними. Процесс создания информационной модели начинается с определения концептуальных требований будущих пользователей БД. Требования отдельных пользователей интегрируются в едином обобщенном представлении, которое называют концептуальной моделью данной предметной области.

Такая модель отображает предметную область в виде взаимосвязанных объектов без указания способов их физического хранения.

Концептуальная модель представляет собой интегрированные концептуальные требования всех пользователей к базе данных данной предметной области. При этом усилия разработчика должны быть направлены в основном на структуризацию данных, принадлежащих будущим пользователям БД и выявление взаимосвязей между ними.

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБд, что потребует ее изменения. Вер сия концептуальной модели, которая может быть реализована конкретной СУБд, называется логической моделью.

Логическая модель, отражающая логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения, может быть реляционной, иерархической или сете вой. Таким образом, логическая модель отображает логические связи между информационными данными в данной концептуальной модели.

Различным пользователям в информационной модели соответствуют различные подмножества ее логической модели, которые называются внешними моделями пользователей. Таким образом, внешняя модель пользователя представляет собой отображение его концептуальных требований в логической модели и соответствует тем представлениям, которые этот пользователь получает о предметной области на основе логической модели. Следовательно, насколько хорошо спроектирована внешняя модель, настолько полно и точно информационная модель отображает предметную область и настолько полно и точно работает автоматизированная система управления этой предмет ной областью.

Логическая модель отображается в физическую память, которая может быть построена на электронных, магнитных, оптических, биологических или других принципах.

Внутренняя модель предметной области определяет размещение данных, методы доступа к ним и технику индексирования в данной логической модели и иначе называется физической моделью. Информационные данные любого пользователя в БД должны быть независимы от всех других пользователей, т. е. не должны оказывать влияния на существующие внешние модели. Это положение отражает первый уровень независимости данных. С другой стороны, внешние модели пользователей никак не связаны с типом физической памяти, в которой будут храниться данные, и с физическими методами доступа к этим данным. Это положение отражает второй уровень независимости данных.

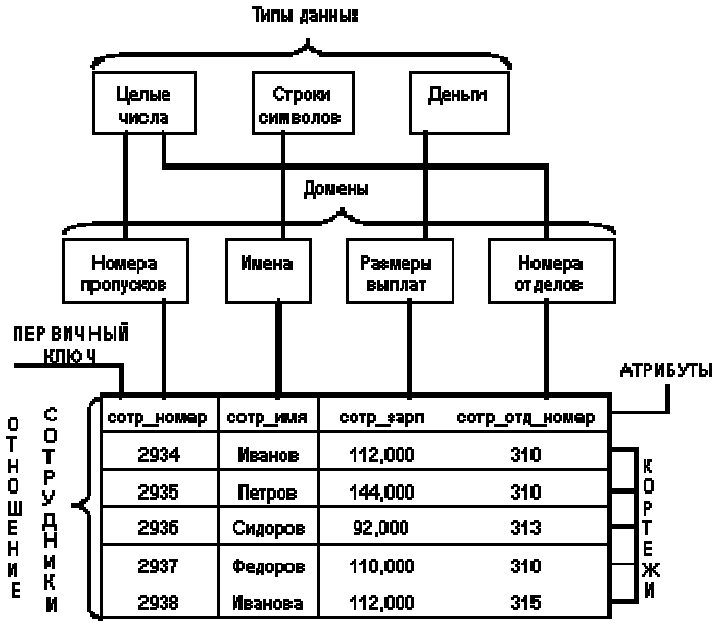
1. 3 Лекция №3 (2 часа).
Тема: «Введение в реляционную модель данных»

1.3.1 Вопросы лекции:
1. Основные понятия реляционных баз данных
2. Фундаментальные свойства отношений
3. Реляционная модель данных

1.3.2 Краткое содержание вопросов:
1. Основные понятия реляционных баз данных.

Основными понятиями реляционных баз данных являются тип данных, домен, атрибут, кортеж, первичный ключ и отношение.

Для начала покажем смысл этих понятий на примере отношения СОТРУДНИКИ, содержащего информацию о сотрудниках некоторой организации:



Понятие **тип данных** в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как "деньги"), а также специальных "темпоральных" данных (дата, время, временной интервал). Достаточно активно развивается подход к расширению возможностей реляционных систем абстрактными типами данных (соответствующими возможностями обладают, например, системы семейства Ingres/Postgres). В нашем примере мы имеем дело с данными трех типов: строки символов, целые числа и "деньги".

Понятие **домена** более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, домен "Имена" в нашем примере определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, такие строки не могут начинаться с мягкого знака).

Следует отметить также семантическую нагрузку понятия домена: данные считаются сравнимыми только в том случае, когда они относятся к одному домену. В нашем примере значения доменов "Номера пропусков" и "Номера групп" относятся к типу целых чисел, но не являются сравнимыми. Заметим, что в большинстве реляционных СУБД понятие домена не используется, хотя в Oracle V.7 оно уже поддерживается.

Схема отношения - это именованное множество пар {имя атрибута, имя домена (или типа, если понятие домена не поддерживается)}. Степень или "арность" схемы отношения - мощность этого множества. Степень отношения СОТРУДНИКИ равна четырем, то есть оно является 4-арным. Если все атрибуты одного отношения определены на разных доменах, осмысленно использовать для именования атрибутов имена соответствующих доменов (не забывая, конечно, о том, что это является всего лишь удобным способом именования и не устраняет различия между понятиями домена и атрибута).

Схема БД (в структурном смысле) - это набор именованных схем отношений.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута,

принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

Однако в реляционных базах данных это не принято. Имя схемы отношения в таких базах данных всегда совпадает с именем соответствующего отношения-экземпляра. В классических реляционных базах данных после определения схемы базы данных изменяются только отношения-экземпляры. В них могут появляться новые и удаляться или модифицироваться существующие кортежи. Однако во многих реализациях допускается и изменение схемы базы данных: определение новых и изменение существующих схем отношения. Это принято называть эволюцией схемы базы данных.

Обычным житейским представлением отношения является таблица, заголовком которой является схема отношения, а строками - кортежи отношения-экземпляра; в этом случае имена атрибутов именуют столбцы этой таблицы. Поэтому иногда говорят "столбец таблицы", имея в виду "атрибут отношения". Когда мы перейдем к рассмотрению практических вопросов организации реляционных баз данных и средств управления, мы будем использовать эту житейскую терминологию. Этой терминологии придерживаются в большинстве коммерческих реляционных СУБД.

Реляционная база данных - это набор отношений, имена которых совпадают с именами схем отношений в схеме БД.

Как видно, основные структурные понятия реляционной модели данных (если не считать понятия домена) имеют очень простую интуитивную интерпретацию, хотя в теории реляционных БД все они определяются абсолютно формально и точно.

2. Фундаментальные свойства отношений.

Остановимся теперь на некоторых важных свойствах отношений, которые следуют из приведенных ранее определений:

Отсутствие кортежей-дубликатов

То свойство, что отношения не содержат кортежей-дубликатов, следует из определения отношения как множества кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов.

Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа - набора атрибутов, значения которых однозначно определяют кортеж отношения. Для каждого отношения по крайней мере полный набор его атрибутов обладает этим свойством. Однако при формальном определении первичного ключа требуется обеспечение его "минимальности", т.е. в набор атрибутов первичного ключа не должны входить такие атрибуты, которые можно отбросить без ущерба для основного свойства - однозначно определять кортеж. Понятие первичного ключа является исключительно важным в связи с понятием целостности баз данных. Во многих практических реализациях РСУБД допускается нарушение свойства уникальности кортежей для промежуточных отношений, порождаемых неявно при выполнении запросов. Такие отношения являются не множествами, а мультимножествами, что в ряде случаев позволяет добиться определенных преимуществ, но иногда приводит к серьезным проблемам.

Отсутствие упорядоченности кортежей

Свойство отсутствия упорядоченности кортежей отношения также является следствием определения отношения-экземпляра как множества кортежей. Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных. Это не противоречит тому, что при формулировании запроса к БД, например, на языке SQL можно потребовать сортировки результирующей таблицы в соответствии со значениями некоторых столбцов. Такой результат, вообще говоря, не отношение, а некоторый упорядоченный список кортежей.

Отсутствие упорядоченности атрибутов

Атрибуты отношений не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута, имя домена}. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута. Это свойство теоретически позволяет, например, модифицировать схемы существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов. Однако в большинстве существующих систем такая возможность не допускается, и хотя упорядоченность набора атрибутов отношения явно не требуется, часто в качестве неявного порядка атрибутов используется их порядок в линейной форме определения схемы отношения.

Атомарность значений атрибутов

Значения всех атрибутов являются атомарными. Это следует из определения домена как потенциального множества значений простого типа данных, т.е. среди значений домена не могут содержаться множества значений (отношения). Принято говорить, что в реляционных базах данных допускаются только нормализованные отношения или отношения, представленные в первой нормальной форме. Потенциальным примером ненормализованного отношения является следующее:

НОМЕР_ОТДЕЛА	ОТДЕЛ		
	СОТР_НОМЕР	СОТР_ИМЯ	СОТР_ЗАРП
310	2934	Иванов	112,000
	2935	Петров	112,500
313	2937	Федоров	110,000
315	2938	Иванова	112,000

Можно сказать, что здесь мы имеем бинарное отношение, значениями атрибута ОТДЕЛЫ которого являются отношения. Заметим, что исходное отношение СОТРУДНИКИ является нормализованным вариантом отношения ОТДЕЛЫ:

СОТР_НОМЕР	СОТР_ИМЯ	СОТР_ЗАРП	СОТР_ОТД_НОМЕР
2934	Иванов	112,000	310
2935	Петров	144,000	310
2936	Сидоров	92,000	313
2937	Федоров	110,000	310
2938	Иванова	112,000	315

Нормализованные отношения составляют основу классического реляционного подхода к организации баз данных. Они обладают некоторыми ограничениями (не любую информацию удобно представлять в виде плоских таблиц), но существенно упрощают манипулирование данными. Рассмотрим, например, два идентичных оператора занесения кортежа:

Зачислить сотрудника Кузнецова (пропуск номер 3000, зарплата 115,000) в отдел номер 320 и
Зачислить сотрудника Кузнецова (пропуск номер 3000, зарплата 115,000) в отдел номер 310.

Если информация о сотрудниках представлена в виде отношения СОТРУДНИКИ, оба оператора будут выполняться одинаково (вставить кортеж в отношение СОТРУДНИКИ). Если же работать с ненормализованным отношением ОТДЕЛЫ, то первый оператор выразится в занесение кортежа, а второй - в добавление информации о Кузнецове в множественное значение атрибута ОТДЕЛ кортежа с первичным ключом 310.

3. Реляционная модель данных.

Когда в предыдущих разделах мы говорили об основных понятиях реляционных баз данных, мы не опирались на какую-либо конкретную реализацию. Эти рассуждения в равной степени относились к любой системе, при построении которой использовался реляционный подход.Другими словами, мы использовали понятия так называемой реляционной модели данных. Модель данных описывает некоторый набор родовых понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими базы данных, если они основываются на этой модели. Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык. Хотя понятие модели данных является общим, и можно говорить о иерархической, сетевой, некоторой семантической и т.д. моделях данных, нужно отметить, что это понятие было введено в обиход применительно к реляционным системам и наиболее эффективно используется именно в этом контексте. Попытки прямолинейного применения аналогичных моделей к дореляционным организациям показывают, что реляционная модель слишком "велика" для них, а для постреляционных организаций она оказывается "мала".

Наиболее распространенная трактовка реляционной модели данных, по-видимому, принадлежит Дейту, который воспроизводит ее (с различными уточнениями) практически во всех своих книгах. Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода: структурной части, манипуляционной части и целостной части. В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных БД, является нормализованное n-арное отношение. По сути дела, в предыдущих двух разделах этой лекции мы рассматривали именно понятия и свойства структурной составляющей реляционной модели. В манипуляционной части модели утверждаются два фундаментальных механизма манипулирования реляционными БД - реляционная алгебра и реляционное исчисление. Первый механизм базируется в основном на классической теории множеств (с некоторыми уточнениями), а второй - на классическом логическом аппарате исчисления предикатов первого порядка. Мы рассмотрим эти механизмы более подробно на следующей лекции, а пока лишь заметим, что основной функцией манипуляционной части реляционной модели является обеспечение меры реляционности любого конкретного языка реляционных БД: язык называется реляционным, если он обладает не меньшей выразительностью и мощностью, чем реляционная алгебра или реляционное исчисление.

Наконец, в целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД. Первое требование называется требованием целостности сущностей. Объекту или сущности реального мира в реляционных БД соответствуют кортежи отношений. Конкретно требование состоит в том, что любой кортеж любого отношения отличим от любого другого кортежа этого отношения, т.е. другими словами, любое отношение должно обладать первичным ключом. Как мы видели в предыдущем разделе, это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Второе требование называется требованием целостности по ссылкам и является несколько более сложным. Очевидно, что при соблюдении нормализованности отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений. Например, представим, что нам требуется представить в реляционной базе данных сущность ОТДЕЛ с атрибутами ОТД_НОМЕР (номер отдела), ОТД_КОЛ (количество сотрудников) и ОТД_СОТР (набор сотрудников отдела). Для каждого сотрудника нужно хранить СОТР_НОМЕР (номер сотрудника), СОТР_ИМЯ (имя

сотрудника) и СОТР_ЗАРП (заработная плата сотрудника). Как мы вскоре увидим, при правильном проектировании соответствующей БД в ней появятся два отношения: ОТДЕЛЫ (ОТД_НОМЕР, ОТД_КОЛ) (первичный ключ - ОТД_НОМЕР) и СОТРУДНИКИ (СОТР_НОМЕР, СОТР_ИМЯ, СОТР_ЗАРП, СОТР_ОТД_НОМ) (первичный ключ - СОТР_НОМЕР).

Как видно, атрибут СОТР_ОТД_НОМ появляется в отношении СОТРУДНИКИ не потому, что номер отдела является собственным свойством сотрудника, а лишь для того, чтобы иметь возможность восстановить при необходимости полную сущность ОТДЕЛ. Значение атрибута СОТР_ОТД_НОМ в любом кортеже отношения СОТРУДНИКИ должно соответствовать значению атрибута ОТД_НОМ в некотором кортеже отношения ОТДЕЛЫ. Атрибут такого рода называется внешним ключом, поскольку его значения однозначно характеризуют сущности, представленные кортежами некоторого другого отношения (т.е. задают значения их первичного ключа). Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом. Требование целостности по ссылкам, или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать). Для нашего примера это означает, что если для сотрудника указан номер отдела, то этот отдел должен существовать.

Ограничения целостности сущности и по ссылкам должны поддерживаться СУБД. Для соблюдения целостности сущности достаточно гарантировать отсутствие в любом отношении кортежей с одним и тем же значением первичного ключа. С целостностью по ссылкам дела обстоят несколько более сложно. Понятно, что при обновлении ссылающегося отношения (вставке новых кортежей или модификации значения внешнего ключа в существующих кортежах) достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа. Но как быть при удалении кортежа из отношения, на которое ведет ссылка? Здесь существуют три подхода, каждый из которых поддерживает целостность по ссылкам. Первый подход заключается в том, что запрещается производить удаление кортежа, на который существуют ссылки (т.е. сначала нужно либо удалить ссылающиеся кортежи, либо соответствующим образом изменить значения их внешнего ключа). При втором подходе при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным. Наконец, третий подход (каскадное удаление) состоит в том, что при удалении кортежа из отношения, на которое ведет ссылка, из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи.

В развитых реляционных СУБД обычно можно выбрать способ поддержания целостности по ссылкам для каждой отдельной ситуации определения внешнего ключа. Конечно, для принятия такого решения необходимо анализировать требования конкретной прикладной области.

1. 4 Лекция №4 (1 час).
Тема: «Системы управления базами данных»

1.4.1 Вопросы лекции:
1. Общие свойства СУБД
2. Обобщенная схема обмена данных с использованием СУБД
3. Типовые информационные процедуры, реализуемые СУБД
4. Общие сведения о СУБД

1.4.2 Краткое содержание вопросов:

1. Общие свойства СУБД.

БД — это единое хранилище данных, которое однократно определяется, а после этого многократно используется разными пользователями для удовлетворения возникающих многообразных потребностей в информации.

ИС может быть

- одно- или
- многопользовательской.

Данные могут быть организованы в одну или несколько БД, данные в БД могут быть интегрированными (объединение нескольких отдельных файлов данных, полностью или частично не перекрывающихся) и общими (использование отдельных областей данных несколькими пользователями для разных целей).

Информация должна быть организована так, чтобы обеспечить минимальную избыточность. БД содержит информацию о данных, принятую называть «метаданными». В совокупности описания всех данных образуют словарь данных, обеспечивающий независимость данных от приложений. Данные должны быть логически связаны, для чего определяют объекты (то, о чем необходимо хранить информацию) и их свойства (простые и сложные), а затем выявляют связи между ними (то, что объединяет два или более объектов, они также являются частью данных и хранятся в БД).

Основные свойства БД:

- Целостность (В каждый момент времени существования БД сведения, содержащиеся в ней, должны быть непротиворечивы),
- Восстанавливаемость (Данное свойство предполагает возможность восстановления БД после сбоя системы или отдельных видов порчи системы),
- Безопасность (предполагает защиту данных от преднамеренного и непреднамеренного доступа, модификации или разрушения),
- Эффективность (сочетание минимального времени реакции на запрос пользователя и минимальной потребности в памяти),
- Предельные размеры и эксплуатационные ограничения.

К основным свойствам СУБД и базы данных можно отнести:

- отсутствие дублирования данных в различных объектах модели, обеспéчивающее однократный ввод данных и простоту их корректировки;
- непротиворечивость данных;
- целостность БД;
- возможность многоаспéктного доступа;
- всевозможные выборки данных и их использование различными задачами и приложениями пользователя;
- защита и восстановление данных при аварийных ситуациях, аппаратных и программных сбоях, ошибках пользователя;
- защита данных от несанкционированного доступа средствами разграничения доступа для различных пользователей;
- возможность модификации структуры базы данных без повторной загрузки данных;
- обеспéчение независимости программ от данных, позволяющей сохранить программы при модификации структуры базы данных;
- реорганизация размещения данных базы на машинном носителе для улучшения объемно-временных характеристик БД;
- наличие языка запросов высокого уровня, ориентированного на конечного пользователя, который обеспéчивает вывод информации из базы данных по любому запросу и предоставление ее в виде соответствующих отчетных форм, удобных для пользователя.

СУБД является основой создания практических приложений пользователя для различных предметных областей.

Критерии выбора СУБД пользователем. Выбор СУБД для практических приложений пользователем определяется многими факторами, к которым относятся:

- имеющееся техническое и базовое программное обеспéчение, их конфигурация, опéративная и дисковая память;
- потребности разрабатываемых приложений пользователя;

- тип поддерживаемой модели данных, спêцифика предметной области, топология информационно-логической модели;

- требования к производительности при обработке данных;

- наличие в СУБД необходимых функциональных средств;

- наличие русифицированной версии СУБД;

- уровень квалификации пользователей и наличие в СУБД диалоговых средств разработки и работ с БД.

СУБД — это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

Различаются два класса СУБД — системы общего назначения (не ориентированы на какую-либо конкретную предметную область, обладают средствами настройки на работу с конкретной БД в условиях конкретного применения, реализуются как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной обстановке) и специализированные системы (весьма трудоемкий процесс, к которому прибегают, в исключительных ситуациях).

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.

СУБД включает язык определения данных, с помощью которого можно определить базу данных, ее структуру, типы данных, а также средства задания ограничений для хранимой информации.

СУБД позволяет вставлять, удалять, обновлять и извлекать информацию из базы данных посредством языка управления данными (язык запросов).

Большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами.

Многопользовательские СУБД имеют достаточно развитые средства администрирования БД.

СУБД предоставляет контролируемый доступ к базе данных с помощью системы обеспечения безопасности, системы поддержки целостности базы данных, системы управления параллельной работой приложений, системы восстановления, доступного пользователям каталога, содержащего описание хранимой в базе данных информации.

Компоненты СУБД - это данные (наиболее важный компонентом среды СУБД, ради общения с ними на должном уровне требуется наличие остальных компонентов, они включают в себя имена, типы и размеры элементов данных, имена связей, ограничения целостности данных, имена зарегистрированных пользователей и их права по доступу к данным, используемые индексы и структуры хранения), пользователи, аппаратное обеспечение (это набор физических устройств, на которых существуют база данных, СУБД и другие компоненты ИС, оно должно соответствовать требованиям использующей его организации, СУБД, БД. Это может быть один персональный компьютер или сеть), программное обеспечение (ОС, программное обеспечение самой СУБД, прикладные программы), процедуры.

Среди пользователей БД можно выделить четыре категории лиц: администраторы данных (отвечают за концептуальное и логическое проектирование базы данных, управление данными, разработку и сопровождение стандартов, бизнес-правил и деловых процедур) и баз данных (отвечают за физическое проектирование и физическую реализацию базы данных, обеспечение безопасности и целостности данных, обеспечение максимальной производительности приложений), разработчики баз данных (группа пользователей, которая функционирует во время проектирования, создания и реорганизации базы данных и результатом деятельности которой является хорошо спроектированная БД, они делятся на разработчиков логической и физической БД, первые занимаются выявлением объектов, их свойств и связей между ними, вторые – разбираются в функциональных возможностях

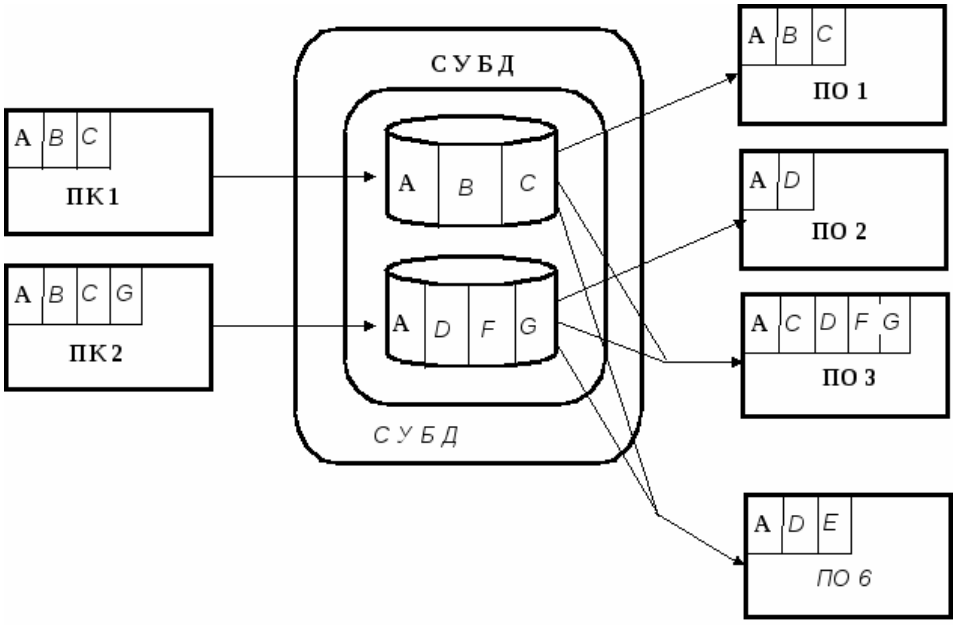
СУБД), прикладные программисты (разрабатывают приложения, предоставляющие пользователям необходимые им функциональные возможности), конечные пользователи (те, для кого проектируется, создается и поддерживается БД).

2. Обобщенная схема обмена данных с использованием СУБД.

СУБД — специализированные программные средства, предназначенные для организации и ведения баз данных.

СУБД используется для обеспечения эффективного доступа к базе данных со стороны программ (предоставление только необходимой информации, обеспечение независимости от возможных изменение в структуре той части базы данных, которую не обрабатывает программа), по существу она исполняет функции операционной системы по управлению данными. Схема иллюстрирует это положение.

В зависимости от способа взаимодействия СУБД и программы обработки либо в программу передается очередная запись требуемой структуры, не зависимо от того, где физически в БД расположены данные, составляющие требуемую структуру, либо для программы создается временный файл требуемой структуры.



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.

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 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;

системы поддержки целостности базы данных;

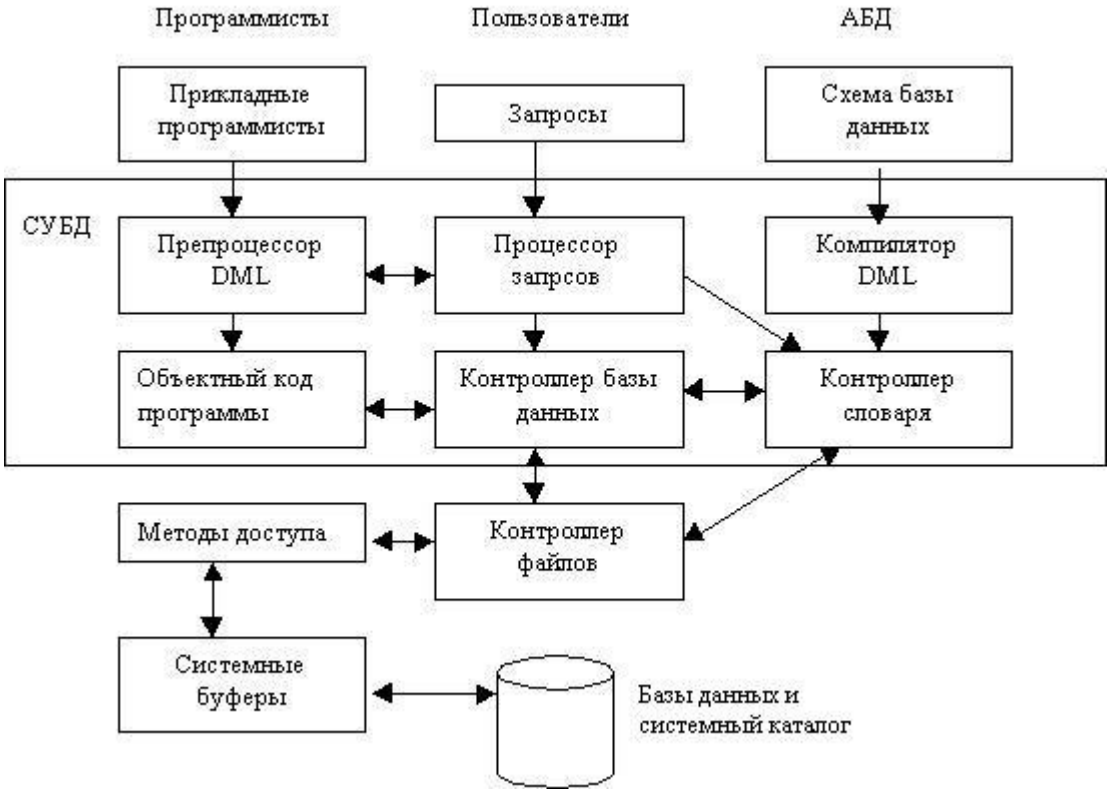
системы управления параллельной работой приложений;

системы восстановления.

СУБД состоит из нескольких программных компонентов (модулей), каждый из которых предназначен для выполнения определенной операции. На схеме также показано, как СУБД взаимодействует с другими программными компонентами, например с такими, как

пользовательские запросы и методы доступа (т.е. методы управления файлами, используемые при сохранении и извлечении записей с данными).

Процессор запросов. Это основной компонент СУБД, который преобразует запросы в последовательность низкоуровневых инструкций для контроллера базы данных.



Основные компоненты типичной системы управления базами данных

Контроллер базы данных. Этот компонент взаимодействует с запущенными пользователями прикладными программами и запросами. Контроллер базы данных принимает запросы и проверяет внешние и концептуальные схемы для определения тех концептуальных записей, которые необходимы для удовлетворения требований запроса. Затем контроллер базы данных вызывает контроллер файлов для выполнения поступившего запроса.

Контроллер файлов манипулирует предназначенными для хранения данных файлами и отвечает за распределение доступного дискового пространства. Он создает и поддерживает список структур и индексов, определенных во внутренней схеме. Если используются хешированные файлы, то в его обязанности входит и вызов функций хеширования для генерации адресов записей. Однако контроллер файлов не управляет физическим вводом и выводом данных непосредственно, а лишь передает запросы соответствующим методам доступа, которые считывают данные в системные буферы или записывают их оттуда на диск.

Препроцессор языка DML. Этот модуль преобразует внедренные в прикладные программы DML-операторы в вызовы стандартных функций базового языка. Для генерации соответствующего кода препроцессор языка DML должен взаимодействовать с процессором запросов.

Компилятор языка DDL. Компилятор языка DDL преобразует DDL-команды в набор таблиц, содержащих метаданные. Затем эти таблицы сохраняются в системном каталоге, а управляющая информация - в заголовках файлов с данными.

Контроллер словаря. Контроллер словаря управляет доступом к системному каталогу и обеспечивает работу с ним. Системный каталог доступен большинству компонентов СУБД.

Ниже перечислены основные программные компоненты, входящие в состав контроллера базы данных.

Контроль прав доступа. Этот модуль проверяет наличие у данного пользователя полномочий для выполнения затребованной операции.

Процессор команд. После проверки полномочий пользователя для выполнения затребованной операции управление передается процессору команд.

Средства контроля целостности. В случае операций, которые изменяют содержимое базы данных, средства контроля целостности выполняют проверку того, удовлетворяет ли затребованная операция всем установленным ограничениям поддержки целостности данных (например, требованиям, установленным для ключей).

Оптимизатор запросов. Этот модуль определяет оптимальную стратегию выполнения запроса.

Контроллер транзакций. Этот модуль осуществляет требуемую обработку операций, поступающих в процессе выполнения транзакций.

Планировщик. Этот модуль отвечает за бесконфликтное выполнение параллельных операций с базой данных. Он управляет относительным порядком выполнения операций, затребованных в отдельных транзакциях.

Контроллер восстановления. Этот модуль гарантирует восстановление базы данных до непротиворечивого состояния при возникновении сбоев. В частности, он отвечает за фиксацию и отмену результатов выполнения транзакций.

Контроллер буферов. Этот модуль отвечает за перенос данных между оперативной памятью и вторичным запоминающим устройством - например, жестким диском или магнитной лентой. Контроллер восстановления и контроллер буферов иногда (в совокупности) называют контроллером данных.

Для воплощения базы данных на физическом уровне помимо перечисленных выше модулей нужны некоторые другие структуры данных. К ним относятся файлы данных и индексов, а также системный каталог. Группой DAFTG (Data-base Architecture Framework Task Group) была предпринята попытка стандартизации СУБД, и в 1986 году ею была предложена некоторая эталонная модель. Назначение эталонной модели заключается в определении концептуальных рамок для разделения предпринимаемых попыток стандартизации на более управляемые части и указания взаимосвязей между ними на очень широком уровне.

3. Типовые информационные процедуры, реализуемые СУБД.

Типовые информационные процедуры, реализуемые СУБД:

1. Определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки. Данная процедура позволяет описать и создать в памяти структуру таблицы, провести начальную загрузку данных в таблицы.

2. Предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация).Такие возможности в СУБД представляются либо на основе использования специального языка программирования, входящего в состав СУБД, либо с помощью графического интерфейса.

3. Обеспечение независимости прикладных программ и данных (логической и физической независимости).Обеспечение логической независимости данных предоставляет возможность изменения логического представления базы данных без необходимости изменения физических структур хранения данных. Обеспечение физической независимости данных предоставляет возможность изменять способы организации базы данных в памяти ЭВМ не вызывая необходимости изменения "логического" представления данных.

4. Защита логической целостности базы данных.

Основной целью реализации этой функции является повышение достоверности данных в базе данных. Достоверность данных может быть нарушена при их вводе в БД или

при неправомерных действиях процедур обработки данных, получающих и заносящих в БД неправильные данные.

5. Защита физической целостности.

Развитые СУБД имеют средства восстановления базы данных, ведение журнала транзакций.

6. Управление полномочиями пользователей на доступ к базе данных.

7. Синхронизация работы нескольких пользователей.

Коллизии(одновременной обращение к одним данным) могут привести к нарушению логической целостности данных, поэтому система должна предусматривать меры, не допускающие обновление данных другим пользователям, пока работающий с этими данными пользователь полностью не закончит с ними работать.

8. Управление ресурсами среды хранения.

9. Поддержка деятельности системного персонала.

При эксплуатации базы данных может возникать необходимость изменения параметров СУБД, выбора новых методов доступа, изменения (в определенных пределах) структуры хранимых данных, а также выполнения ряда других общесистемных действий.

4. Общие сведения о СУБД.

Первые СУБД появились в конце 60-х годов, расцвет их применения приходится на 70-е начало 80-х годов. В то время на первый план выступала способность СУБД хранить данные сложной структуры и значительного объема и использовать установленные связи между информационными элементами при проектировании приложений. Другое важное достоинство - это обеспечение относительной независимости программ от структур хранения. Первые СУБД были ориентированы на программистов. Интерфейс с такими СУБД осуществлялся путем обращения к программе – СУБД из программы приложения, написанной на одном из базовых языков программирования и такие системы стали называть системами с базовым языком. При этом СУБД выполняла лишь простые операции выборки записей, удовлетворяющих определенным условиям и в определенной последовательности, а также включения, замены и удаления записей. Но все эти операции осуществлялись с учетом зафиксированной структуры БД, что существенно сокращало алгоритмическую часть программы, касающуюся согласованной выборки связанных записей, и снижало риск нарушения структурной целостности БД.

IMS (Information Management System) являлись весьма распространенными СУБД, обеспечивающими хранение и доступ к БД иерархической структуры.

Элементом структуры является сегмент, который может состоять из одного или нескольких полей. Поле может иметь тип: символьный или числовой. Сегменты одного типа имеют единую для этого типа внутреннюю структуру и размер, в том числе фиксированные размеры и типы одноименных полей в различных реализациях сегментов.

БД сложной логической структуры может быть разнесена в 10 физических наборов данных (файлов). IMS поддерживает 4 способа физической организации БД: иерархический последовательный метод доступа; иерархический индексно-последовательный метод доступа; иерархический индексно-прямой метод доступа и иерархический прямой.

Система IMS(ОКА) является системой с базовым языком.

Единицей обмена между прикладной программой и БД является реализация сегмента определенного типа (с подсоединенными ключами старших сегментов) или совокупности реализаций сегментов, расположенных в одной иерархической ветви.

Система обеспечивает выполнение всех типичных для СУБД функций: вставку, замену, удаление и чтение реализаций сегментов.

Система ID

СУБД IDS обеспечивают хранение и доступ к БД с сетевой структурой.

Основной структурной единицей, обеспечивающей в конечном счете сетевую структуру любой сложности, является цепь. В цепь объединяются информационно зависимые логические записи.

Запись представляет собой основную единицу информации и состоит из полей данных и служебных полей.

Служебные поля предназначены для идентификации записи и организации цепей с помощью адресных ссылок.

Информационные поля могут содержать числовые и символьные данные и имеют имена.

Записи одного типа содержат одноименные данные и имеют фиксированную длину, равно как и одноименные поля. Записям одного типа присваиваются уникальное имя и тип. Записи адресуются в БД с помощью адресных ссылок, состоящих из номера «страницы» БД и номера байта начала записи на «странице». Цепи образуются с помощью адресных ссылок, хранящихся в записи и указывающих на следующую запись в этой цепи. Каждый цепи присваивается уникальное имя. Каждая запись может входить в любое количество цепей.

Система также обеспечивает типовые операции доступа к данным – запоминание, выборку, модификацию и удаление записей из программ, написанных на классических языках программирования.

Система ADABAS

ADABAS сочетает в себе возможности СУБД с базовым языком и системы замкнутого типа и обеспечивает поддержание ограниченных сетевой и иерархической структур. БД в ADABAS является совокупностью связанных данных, организованных в виде файлов. Реализованное в системе динамическое установление связей между записями различных файлов позволяет создавать межзаписные иерархические и сетевые структуры. ADABAS осуществляет поиск по запросам пользователей записей в БД. Доступ к хранимым данным, обработку данных и выдачу результатов в виде справок и сводок заданной формы. Запросы вводятся в диалоговом режиме с видеотерминалов и в пакетном режиме в образе перфокарт. Языки запросов высокого уровня ориентированы на пользователей-непрограммистов. Обеспечивается и мультидоступ к БД.

ADABAS дает возможность обращаться к БД и из прикладных программ, написанных на АССЕМБЛЕРЕ, КОБОЛе, ФОРТРАНе и PL/1, в том числе под управлением телемонитора.

Система FoxPro

СУБД FoxPro обеспечивает возможность работы в трех режимах:

• выполнение откомпилированных программ, написанных на языке СУБД (xBASE-язык);

• выполнение команд языка xBASE или команд SQL в режиме интер-претации;

• обработка баз данных в экранном интерфейсе (без предварительного программирования и без знания команд языка xBASE).

База данных FoxPro состоит из совокупности взаимосвязанных файлов, каждый из которых может быть представлен в виде таблицы. Запись файла соответствует строке таблицы, поле – столбцу таблицы. Файл и поля имеют имена – идентификаторы назначаемые при первоначальном создании файла.

Все записи одного файла однотипны, имеют фиксированную длину, потому как одноименные поля во всех записях имеют один и тот же размер и тип значения. В одной записи (строке таблицы) у одного поля может быть только единственное значение. В файле (таблице) не может быть двух одинаковых записей. Одно либо несколько полей являются ключом записи. Значение ключа уникально.

Файлы могут быть связаны друг с другом:

• Возможные виды связей 1 : 1, 1 : M, M : 1, M : N:

Система Access

СУБД Access является СУБД реляционного типа, в которой разумно сбалансированы все средства и возможности, типичные для современных СУБД. В отличии от FoxPro, в Access все данные вместе с запросами и формами физически хранятся в одном файле.

База данных Microsoft Access состоит из следующих объектов:

- таблицы,
- запросы,
- формы,
- отчеты,
- макросы,
- модули.

Поскольку, как правило, система управления базами данных обрабатывает одновременно несколько таблиц, то существует возможность установления реляционных связей между таблицами.

Клиент-сервер

При использовании клиент-серверной технологии, на самом сервере, содержащим базу данных, функционирует некоторое программное обеспечение, которое называется "Сервером баз данных" или "Сервером БД". Клиент-серверная СУБД позволяет обмениваться клиенту и серверу минимально необходимыми объёмами информации. При этом основная вычислительная нагрузка ложится на сервер. Клиент может выполнять функции предварительной обработки перед передачей информации серверу, но в основном его функции за-ключаются в организации доступа пользователя к серверу. Таким образом, архитектура клиент-сервер адаптирована для работы с большими объемами данных - сеть нагружается меньше, требования к пользовательским компьютерам, с точки зрения производительности, минимизируется. Однако возрастают требования к серверу, содержащему базу данных, поскольку теперь он один тянет нагрузку всех пользователей. Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу. Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высо-кая надёжность, высокая доступность и высокая безопасность.

Примеры: Oracle, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Cache.

1. 5 Лекция №5 (1 час).

Тема: «Информационные системы, основанные на БД и СУБД»

1.5.1 Вопросы лекции:

1. Обобщенная схема информационной системы, основанной на БД и СУБД
2. Состав и функции средств актуализации БД, средств обработки БД в интересах пользователей, средств администрирования БД
3. Технологии файл-сервер и клиент-сервер

1.5.2 Краткое содержание вопросов:

1. Обобщенная схема информационной системы, основанной на БД и СУБД.

Информационная система – это любая система, реализующая или поддерживающая информационный процесс.

К информационным можно относить любые системы, включающие в себя работу с информацией. В настоящее время основным помощником человека при работе с информацией является компьютер, поэтому именно его мы и будем рассматривать в качестве

источника, способа изменения и хранения информационных систем. А в качестве информационных систем будем рассматривать программное обеспечение компьютера.

В зависимости от предметной области информационные системы могут весьма значительно различаться по своим функциям, архитектуре, реализации. Однако можно выделить ряд свойств, которые являются общими.

Информационные системы предназначены организации и поддержке информационного процесса, поэтому в основе любой из них лежит среда хранения и доступа к информации.

Информационные системы ориентированы на конечного пользователя, не обладающего высокой квалификацией в области вычислительной техники. Поэтому клиентские приложения информационной системы должны обладать простым, удобным, легко осваиваемым интерфейсом.

Таким образом, при разработке информационной системы приходится решать две основные задачи:

- разработка базы данных, предназначенной для хранения информации;
- разработка графического интерфейса пользователя клиентских приложений.

Подавляющее большинство информационных систем работает в режиме диалога с пользователем.

В наиболее общем случае типовые программные компоненты, входящие в состав информационной системы, реализуют:

- диалоговый ввод-вывод;
- логику диалога;
- прикладную логику обработки данных;
- логику управления данными;
- операции манипулирования файлами и (или) базами данных.

Классификация информационных систем

Информационные системы классифицируются по разным признакам:

1. Классификация по масштабу (рис1)

По масштабу информационные системы подразделяются на следующие группы:

- одиночные;
- групповые;
- корпоративные.

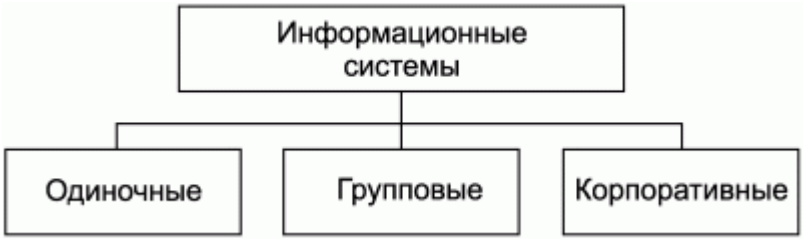


Рис. 1. Деление информационных систем по масштабу.

Одиночные информационные системы реализуются, как правило, на автономном персональном компьютере (сеть не используется). Такая система может содержать несколько простых приложений, связанных общим информационным фондом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место. Подобные приложения создаются с помощью так называемых настольных, или локальных, систем управления базами данных (СУБД). Среди локальных СУБД наиболее известными являются Clarion, Clipper, FoxPro, Paradox, dBase и Microsoft Access.

Групповые информационные системы ориентированы на коллективное использование информации членами рабочей группы и чаще всего строятся на базе локальной вычислительной сети. При разработке таких приложений используют-ся серверы баз данных (называемые также SQL (Structured Query Language – структурированный язык запросов)-серверами) для рабочих групп. Существует довольно большое количество различных SQL-

серверов как коммерческих, так и свободно распространяемых. Среди них наиболее известны такие серверы баз данных, как Oracle, DB2, Microsoft SQL Server, InterBase, Sybase, Informix.

Корпоративные информационные системы являются развитием систем для рабочих групп, они ориентированы на крупные компании и могут поддерживать тер-риториально разнесенные узлы или сети. В основном они имеют иерархическую структуру из нескольких уровней. Для таких систем характерна архитектура клиент-сервер со специализацией серверов или же многоуровневая архитектура. При разработке таких систем могут использоваться те же серверы баз данных, что и при разработке групповых информационных систем. Однако в крупных информационных системах наибольшее распространение получили серверы Oracle, DB2 и Microsoft SQL Server.

2. Классификация по сфере применения

По сфере применения информационные системы обычно подразделяются на четыре группы (рис. 2):

- системы обработки транзакций (протоколов);
- системы поддержки принятия решений;
- информационно-справочные системы;
- офисные информационные системы.

Рис. 2. Деление информационных систем по сфере применения.

Системы обработки транзакций, в свою очередь, по оперативности обработки данных разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций (OnLine Transaction Pro-cessing, OLTP) для отражения актуального состояния предметной области в любой момент времени, а пакетная обработка занимает весьма ограниченную часть. Для систем OLTP характерен регулярный (возможно, интенсивный) поток довольно простых транзакций, играющих роль заказов, платежей, запросов и т.п. Важными требованиями для них являются:

- высокая производительность обработки транзакций;
- гарантированная доставка информации при удаленном доступе к БД по телекоммуникациям.

Системы поддержки принятия решений (Decision Support System, DSS) представляют собой другой тип информационных систем, в которых с помощью довольно сложных

запросов производится отбор и анализ данных в различных разрезах: временных, географических, по другим показателям.

Обширный класс информационно-справочных систем основан на гипертекстовых документах и мультимедиа. Наибольшее развитие такие информационные системы получили в Интернете.

Класс офисных информационных систем нацелен на перевод бумажных документов в электронный вид, автоматизацию делопроизводства и управление документооборотом.

3. Классификация по способу организации

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы (рис. 3):

- системы на основе архитектуры файл-сервер;
- системы на основе архитектуры клиент-сервер;
- системы на основе многоуровневой архитектуры;
- системы на основе Интернет/интранет-технологий.

Рис. 3. Деление информационных систем по способу организации.

В любой информационной системе можно выделить необходимые функциональные компоненты (табл. 1), которые помогают понять ограничения различных архитектур информационных систем. Рассмотрим более подробно особенности вариантов построения информационных приложений.

Таблица 1.1. Типовые функциональные компоненты информационной системы

Обозначение	Наименование	Характеристика
PL	Presentation Logic (логика представления)	Управляет взаимодействием между пользователем и ЭВМ. Обрабатывает действия пользователя при выборе команды в меню, щелчке на кнопке или выборе пункта в списке
BL	Business Logic (прикладная логика)	Набор правил для принятия решений, вычислений и операций, которые должно выполнить приложение
DL	Data Logic (логика управления данными)	Операции с базой данных (реализуемые SQL-операторами), которые нужно выполнить для реализации прикладной логики управления данными
DS	Data Services (операции с базой данных)	Действия СУБД, реализующие логику управления данными, такие как манипулирование данными, определение данных, фиксация или откат транзакций и т. п. СУБД обычно компилирует SQL-предложения
FS	File Services (файловые операции)	Дисковые операции чтения и записи данных для СУБД и других компонентов. Обычно являются функциями операционной системы (ОС)

2. Состав и функции средств актуализации БД, средств обработки БД в интересах пользователей, средств администрирования БД.

К основным функциям ИС относятся функции сбора и регистрации информационных ресурсов, их хранение, обработка, актуализация, а так же обработка запросов пользователя.

Сбор и регистрация обеспечивает фиксирование информации о состоянии предметной области. Работы выполняется как до основного программно-аппаратного комплекса, так в его среде. Реализация функций зависит от источника информации, в качестве которого могут выступать бумажные носители, электронные, автоматизированные технические системы.

Сбор и регистрация могут осуществляться:

- путем измерений (наблюдений) фактов в реальном мире и ввода данных в систему с помощью клавиатуры или каких-либо манипуляторов;
- полуавтоматически путем ввода в компьютер с некоторых носителей и в случае необходимости их перекодировать (например, при использовании текстов на бумажных носителях или аналоговых аудизаписей);
- автоматический с помощью различного рода датчиков или обмена данными с другими автоматизированными системами.

С этими функциями связана необходимость обеспечения контроля, сжатие, конвертирование информации.

Обеспечение контроля информации – необходимая стадия предварительной обработки данных и подготовки их загрузки в систему, особенно в случаях, когда используются несколько источников данных. Обычно она включает процедуры фильтрации данных, верификации, обеспечение логической целостности, устранение несогласованности, избыточности и различных ошибок, восполнения пропусков, а также другие процедуры направленные на улучшение качества информации.

В результате фильтрации производится отбор нужных данных из множества имеющихся в распоряжении. Верификации призвана обеспечивать достоверность и логическую целостность информации. При выполнении данной функции устанавливается, адекватна ли или информация предметной области.

На разных операциях могут применяться различные методы контроля, существуют методы, применимые ко многим операциям, наиболее применимые:

- подсчета контрольных сумм;
- повторное выполнение операций другим оператором с дублированием действий и последующим их сличением;
- контроль набора на клавиатуре;
- контроль информации в соответствие с ее свойствами, структурой и на соответствие значениям.

Способами реализации могут быть:

- ручной (без использования технических средств);
- визуальный (с использованием технических средств и без них);
- аппаратный (технический);
- программный;
- организационные мероприятия.

Выбор конкретных обеспечения верификации зависит от характера, качества, источников данных, видов ограничения целостности.

В некоторых ИС информация хранится в сжатом виде. Сжатие информации минимизирует потребность во внешней памяти, нужной для хранения, а также снижает затраты на передачу данных по каналам связи.

Конвертирование данных при вводе в систему используется для преобразования одного формата данных в другой, допускающий автоматизированный импорт их в ИС. Конвертирование данных необходимо в случаях, когда источником данных является некоторая другая система. Для конвертирования используются специальные программы конверторы.

Хранение и накопление информации вызвано необходимостью многократного использования одни и те же данные при решении задач. Для хранения и поиска информации используются технологии баз данных.

Актуализация информационных ресурсов. Для того, чтобы информация была практически полезной, необходимо своевременно и адекватно отображать в ней изменения состояния предметной области. Актуализация информации в реляционных СУБД сводится к включению и/или удалению строк в таблицах баз данных, обновлению значений некоторых реквизитов. В случаях изменения структуры предметной области системы, актуализация информации заключается в изменении схемы базы данных – добавлении или удалении существующих столбцов таблиц, в создании новых таблиц и удалении существующих таблиц.

В информационно-справочные системах актуализация информации осуществляется путем ввода в систему новых документов, реже удалением существующих.

Актуализация информации в ИС производится дискретно, через определенные интервалы времени. Актуализация информации, т.о., обеспечивается с некоторым отставание во времени. Это отставание в различных ИС изменяется в широком диапазоне и зависит от назначения системы и особенностей ее предметной области. В информационных системах управления сложными техническими объектами, например в системе управления космическими полетами, временной лаг измеряется в миллисекундах. В корпоративных ИС может составлять от нескольких минут до нескольких часов.

Для того чтобы ИС соответствовала своему назначению необходимо соблюдать установленный для нее регламент актуализации.

Предоставление информационных ресурсов пользователю. Все выше описанные операции необходимы для удовлетворения информационных потребностей пользователей.

Существует две технологии предоставления информации пользователю: pull-технология и/ или push-технология.

В случае pull-технологии – инициатором предоставления информации выступает пользователь, а push-технология сама система, в соответствие с регламентом и для определенного круга пользователей.

Для предоставления информации по pull-технологии в ИС предусматриваются пользовательские интерфейсы. Пользовательские интерфейсы – средства взаимодействия пользователя с системой.

При этом пользователь может влиять на последовательность применения тех или иных технологий. С точки зрения влияния пользователя на последовательность операций в процессе функционирования ИС, интерфейсы могут быть разделены на пакетные и диалоговые.

Экономические задачи, решаемые в пакетном режиме, характеризуются следующими свойствами:

- алгоритм решения задачи формализован, процесс ее решения не требует вмешательства человека;
- имеется большой объем входных и выходных данных, значительная часть которых хранится на магнитных дисках;
- расчет выполняется для большинства записей входных файлов;
- большое время решения задачи обусловлено большими объемами данных;
- регламентность, т.е. задачи решаются с заданной периодичностью.

Диалоговый режим не является альтернативой пакетному режиму, а его развитием. Если применение пакетного режима позволяет уменьшить вмешательство пользователя в процесс решения задачи, то диалоговый режим предполагает отсутствие жестко закрепленной последовательности операций обработки данных.

Примером push-технологии может служить рассылка информации среди пользователей Интернет.

Экономическая информационная система по своему составу напоминает предприятие по переработке данных и производству выходной информации. Методы и способы реализации функции ИС (сбора, накопления, хранения, поиска и обработки информации на

основе применения средств вычислительной техники) называются информационной технологией.

Информационные технологии должны быть выстроены в последовательность действий, позволяющую из исходной информации получить результат с заданной достоверностью и безопасностью.

Упорядоченная последовательность взаимосвязанных действий, выполняющихся с момента возникновения информации до получения результата, называется технологическим процессом.

Понятие информационной технологии, таким образом, неотделимо от той специфической среды, в которой она реализована, т.е. от технической и программной Среды.

Администрирование базы данных – это функция управления базой данных (БД). Лицо ответственное за администрирование БД называется “Администратор базы данных” (АБД) или “Database Administrator” (DBA).

Функция “администрирования данных” стала активно рассматриваться и определяться как вполне самостоятельная с конца 60-х годов. Практическое значение это имело для предприятий, использующих вычислительную технику в системах информационного обеспечения для своей ежедневной деятельности. Специализация этой функции с течением времени совершенствовалась, но качественные изменения в этой области стали происходить с началом использования так называемых интегрированных баз данных. Одна такая база данных могла использоваться для решения многих задач.

Таким образом, сформировалось определение БД как общего информационного ресурса предприятия, которое должно находиться всегда в работоспособном состоянии. И как для каждого общего ресурса значительной важности, БД стала требовать отдельного управления. Во многих случаях это было необходимо для обеспечения её повседневной эксплуатации, её развития в соответствии с растущими потребностями предприятия. К тому же БД и технология её разработки постоянно совершенствовались и уже требовались специальные знания высокого уровня для довольно сложного объекта, которым стала база данных. Отсюда функция управления базой данных и получила название “Администрирование базы данных”, а лицо ею управляющее стали называть “Администратор баз данных”.

Администратор базы данных (DBA)

DBA Администратор базы данных (АБД) или Database Administrator (DBA) – это лицо, отвечающее за выработку требований к базе данных, её проектирование, реализацию, эффективное использование и сопровождение, включая управление учётными записями пользователей БД и защиту от несанкционированного доступа. Не менее важной функцией администратора БД является поддержка целостности базы данных.

АБД имеет код специальности по общероссийскому классификатору профессий рабочих, должностей служащих и тарифных разрядов (ОКПДТР) — 40064 и код 2139 по Общероссийскому классификатору занятий (ОКЗ). Код 2139 ОКЗ расшифровывается следующим образом: 2 - СПЕЦИАЛИСТЫ ВЫСШЕГО УРОВНЯ КВАЛИФИКАЦИИ, 21 - Специалисты в области естественных* и инженерных наук, 213 - Специалисты по компьютерам, 2139 - Специалисты по компьютерам, не вошедшие в другие группы.

Классические подходы к наполнению содержанием понятия "АБД" стали формироваться после издания рабочего отчета группы по базам данных Американского Национального Института Стандартов ANSI/X3/SPARC в 1975 года. В этом отчете была описана трехуровневая архитектура СУБД, в которой выделялся уровень внешних схем данных, уровень концептуальной схемы данных и уровень схемы физического хранения данных. В соответствии с этой архитектурой определялись три роли АБД: администратор концептуальной схемы, администратор внешних схем и администратор хранения данных. Эти роли в случае очень маленькой системы мог играть один человек, в большой системе для выполнения каждой роли могла назначаться группа людей. Каждой роли соответствовал набор функций, а все эти функции вместе составляли функции АБД.

В 1980 - 1981 г. в американской литературе стало принятым включать в функции АБД:

- организационное и техническое планирование БД,
- проектирование БД,
- обеспечение поддержки разработок прикладных программ,
- управление эксплуатацией БД.

В нашей стране в это же время первое определение АБД в ГОСТ-ах задало слишком узкий состав функций АБД:

- подготовка вычислительного комплекса к установке СУБД, участие в установке и приемке СУБД и самой БД с комплексом прикладных программ
- управление эксплуатацией БД
- подготовка словарей и другой НСИ - нормативно-справочной информации - к моменту начала испытания БД
- Предполагалось, что функции АБД будут ориентированы только на эксплуатацию БД, а её разработка будет вестись силами специализированной организации.

К середине 90-х годов сложились еще не завершенные, но уже достаточно устойчивые и полные методологии разработки систем с базами данных. Основная работа по планированию информационных потребностей предприятия, проектированию концептуальной и логической схемы БД, внешних схем, используемых в отдельных процессах обработки информации, ложится теперь на группу проектирования Автоматизированной Системы (АС). Становиться и более определённым объем функций АБД. Это обеспечение надежной и эффективной работы пользователей и программ с БД, поддержка разработчиков в их доступе к БД и средствам разработки.

Как таковой официальной версии должностной инструкции администратора базы данных не существует. Имеется несколько документов, различающихся в основном оформлением и содержанием некоторых пунктов. Естественно, ни о какой подробной расшифровке задач администратора базы данных в этих документах речи не идёт. Должностная инструкция — это, прежде всего документ, регламентирующий производственные полномочия и обязанности работника. И хотя в некоторых организациях изменяют текст должностной инструкции в соответствии с условиями специфики работы, не стоит ожидать в ней прямых указаний на то, что надо делать администратору. В большинстве случаев такие детальные директивы работы регламентируются другими документами (например, инструкция по резервному копированию, инструкция по обеспечению информационной безопасности при работе с базами данных).

3. Технологии файл-сервер и клиент-сервер.

Архитектура файл‑сервер.

В архитектуре файл‑сервер сетевое разделение компонентов диалога PS и PL отсутствует, а компьютер используется для функций отображения, что облегчает построение графического интерфейса. Файл‑сервер только извлекает данные из файлов, так что дополнительные пользователи и приложения лишь незначительно увеличивают нагрузку на центральный процессор. Каждый новый клиент добавляет вычислительную мощность к сети.

Объектами разработки в файл‑серверном приложении являются компоненты приложения, определяющие логику диалога PL, а также логику обработки BL и управления данными DL. Разработанное приложение реализуется либо в виде законченного загрузочного модуля, либо в виде специального кода для интерпретации.

Однако такая архитектура имеет существенный недостаток: при выполнении некоторых запросов к базе данных клиенту могут передаваться большие объемы данных, загружая сеть и приводя к непредсказуемости времени реакции. Значительный сетевой трафик особенно сказывается при организации удаленного доступа к базам данных на файл‑сервере через низкоскоростные каналы связи. Одним из вариантов устранения данного недостатка является удаленное управление файл‑серверным приложением в сети. При этом в

локальной сети размещается сервер приложений, совмещенный с телекоммуникационным сервером (обычно называемым сервером доступа), в среде которого выполняются обычные файл-серверные приложения.

Одним из традиционных средств, на основе которых создаются файл-серверные системы, являются локальные СУБД. Однако такие системы, как правило, не отвечают требованиям обеспечения целостности данных (в частности, они не поддерживают транзакции). Поэтому при их использовании задача обеспечения целостности данных возлагается на программы клиентов, что приводит к усложнению клиентских приложений. Тем не менее, эти инструменты привлекают своей простотой, удобством применения и доступностью. Поэтому файл-серверные информационные системы до сих пор представляют интерес для малых рабочих групп и, более того, нередко используются в качестве информационных систем масштаба предприятия.

Архитектура клиент-сервер.

Архитектура клиент-сервер предназначена для разрешения проблем файл-серверных приложений путем разделения компонентов приложения и размещения их там, где они будут функционировать наиболее эффективно. Особенностью архитектуры клиент-сервер является наличие выделенных серверов баз данных, понимающих запросы на языке структурированных запросов (Structured Query Language, SQL) и выполняющих поиск, сортировку и агрегирование информации.

Отличительная черта серверов БД – наличие справочника данных, в котором записана структура БД, ограничения целостности данных, форматы и даже серверные процедуры обработки данных по вызову или по событиям в программе. Объектами разработки в таких приложениях помимо диалога и логики обработки являются, прежде всего, реляционная модель данных и связанный с ней набор SQL-операторов для типовых запросов к базе данных.

Большинство конфигураций клиент-сервер использует двухуровневую модель, в которой клиент обращается к услугам сервера. Предполагается, что диалоговые компоненты PS и PL размещаются на клиенте, что позволяет реализовать графический интерфейс. Компоненты управления данными DS и FS размещаются на сервере, а диалог (PS, PL) и логика (BL, DL) – на клиенте. В двухуровневом определении архитектуры клиент-сервер используется именно этот вариант: приложение работает на клиенте, СУБД – на сервере (рис. 4).

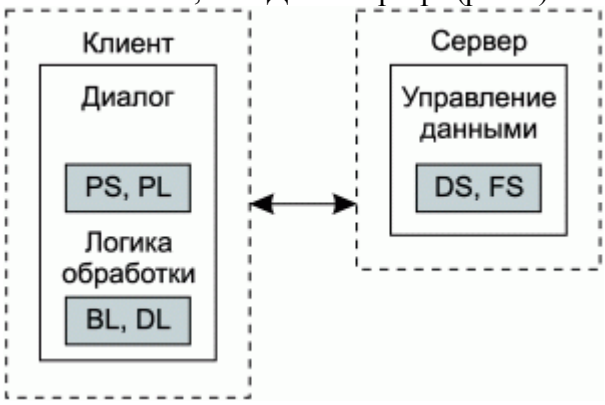


Рис. 4. Классический вариант клиент-серверной системы.

Поскольку эта схема предъявляет наименьшие требования к серверу, она обладает наилучшей масштабируемостью. Однако сложные приложения, активно взаимодействующие с БД, могут жестко загрузить как клиента, так и сеть. Результаты SQL-запроса должны вернуться клиенту для обработки, потому что там реализована логика принятия решения. Такая схема приводит к дополнительному усложнению администрирования приложений, разбросанных по различным клиентским узлам.

Для сокращения нагрузки на сеть и упрощения администрирования приложений компонент BL можно разместить на сервере. При этом вся логика принятия решений оформляется в виде хранимых процедур и выполняется на сервере БД.

Хранимая процедура – процедура с SQL-операторами для доступа к БД, вызываемая по имени с передачей требуемых параметров и выполняемая на сервере БД. Хранимые процедуры могут компилироваться, что повышает скорость их выполнения и сокращает нагрузку на сервер.

Хранимые процедуры улучшают целостность приложений и БД, гарантируют актуальность коллективных операций и вычислений. Улучшается сопровождение таких процедур, а также безопасность (нет прямого доступа к данным).

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1 Лабораторная работа №1 (1 час).

Тема: «Персональные базы данных»

Занятие посвящается проверке полученных навыков по изучаемой теме. Студентам предлагаются для решения индивидуальные варианты заданий, а также, разбившись на группы (не более 3-х человек) они должны предложить конкретную экономическую задачу и решить ее с применением полученных навыков по методам решения и с использованием компьютерной техники. Далее представитель каждой группы представляет решенную ими задачу, защищает ее, отвечая на вопросы студентов других групп (в дискуссии также принимают участие и остальные члены группы), чем осуществляется реализация такой формы, как **разбор конкретных ситуаций**. Оценка выставляется по совокупности индивидуальных заданий и разбора реальных предложенных к рассмотрению задач.

2.1.1 Цель работы: изучить и проанализировать создание персональных баз данных

2.1.2 Задачи работы:

1. Изучение создания персональных баз данных
2. Анализ создания персональных баз данных

2.1.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер

2.1.4 Описание (ход) работы:

Персональные базы данных.

Иерархическая модель данных имеет много общих черт с сетевой моделью данных, хронологически она появилась даже раньше, чем сетевая. Допустимыми информационными конструкциями в иерархической модели данных являются отношение, веерное отношение и иерархическая база данных. В отличие от рассмотренных ранее моделей данных, где предполагалось, что информационным отображением одной предметной области является одна база данных, в иерархической модели данных допускается отображение одной предметной области в нескольких иерархических базах данных.

Понятия «отношение» и «веерное отношение» в иерархической модели данных не изменяются.

Иерархической базой данных называется множество отношений и веерных отношений, для которых соблюдаются два ограничения.

1. Существует единственное отношение, называемое корневым, которое не является зависимым ни в одном веерном отношении.
2. Все остальные отношения (за исключением корневого) являются зависимыми отношениями только в одном веерном отношении.

Схема иерархической базы данных по составу компонент идентична сетевой базе данных. Перечисленные выше ограничения поддерживаются иерархическими СУБД.

На рис. 1 изображена структура иерархической базы данных, представляющая студентов и преподавателей вуза, и удовлетворяющая всем ограничениям, указанным в определении.

В графических иллюстрациях структуры иерархической базы данных приводятся названия соответствующих отношений.

Ограничение, которое поддерживается в иерархической модели данных, состоит в невозможности нарушения требований, фигурирующих в определении иерархической базы данных. Это ограничение обеспечивается специальной укладкой значений отношений в памяти компьютера. Ниже будет рассмотрена одна из простейших реализаций укладки иерархической базы данных.

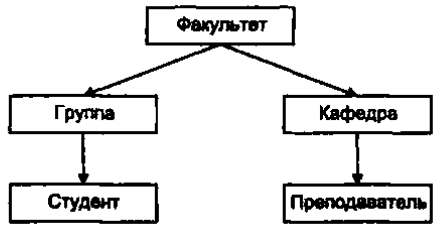


Рис. 1 – Иерархическая база данных «Вуз-Кафедра»

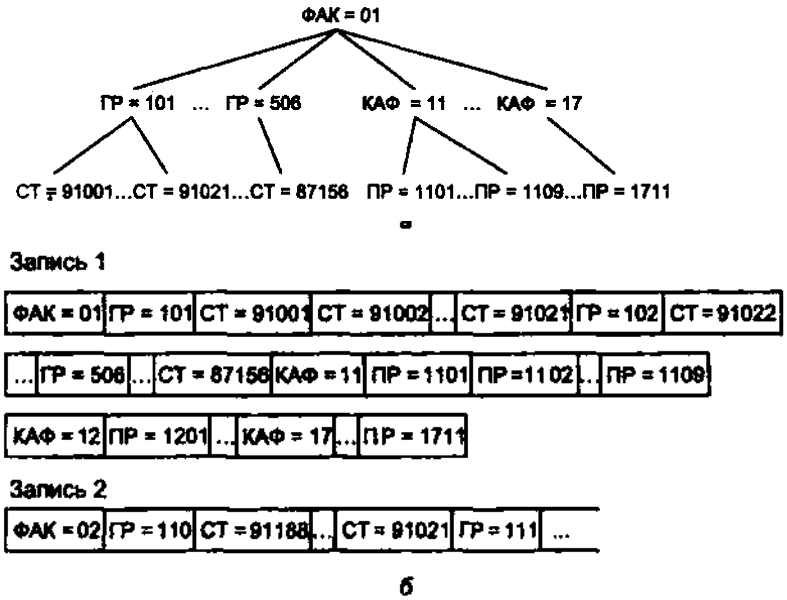


Рис. 2 – Экземпляр иерархической базы данных «Вуз-Кафедра»

На рис. 2 *а* приведена связь значений отношений из иерархической базы данных, структура которой показана на рис. 2.6. Каждое значение представляется соответствующей величиной первичного ключа. Использованы следующие сокращения: ФАК -

факультет, ГР - группа, КАФ - кафедра, СТ - студент, ПР - преподаватель.

Далее эта информация организуется в линейную последовательность значений (рис. 2 *б).*

Необходимо отметить, что известны различные возможности прохождения иерархически организованных значений в линейной последовательности. Принцип, применяемый для иерархических баз данных, называется концевым прохождением.

Правила концевого прохождения следующие:

Шаг 1. Начиная с первого значения корневого отношения перечисляются первые значения соответствующих отношений на каждом уровне вплоть до последнего.

Ш а г 2. Перечисляются все значения в том веерном отношении, на котором остановился шаг 1.

Ш а г 3. Перечисляются значения всех вееров этого веерного отношения.

Шаг 4. От достигнутого уровня происходит подъем на предыдущий уровень, и если возможно применить шаг 1, то процесс повторяется.

Записью иерархической базы данных называется множество значений, содержащих одно значение корневого отношения и все вееры, доступные от этого значения в соответствии со структурой иерархической базы данных. В нашем примере одну запись образуют данные, относящиеся к одному факультету.

Для веерных отношений в составе иерархической базы данных справедлива уже известная закономерность: если существует веерное отношение, то ключ зависимого отношения функционально определяет ключ основного отношения, и наоборот, если ключ одного отношения функционально определяет ключ второго отношения, то первое отношение может быть зависимым, а второе - основным в некотором веерном отношении. Кроме того, ограничение на наличие единственного корневого отношения в иерархической базе данных трансформируется в требование: первичный ключ каждого некорневого отношения должен функционально определять первичный ключ корневого отношения.

Рассмотрим алгоритм формирования иерархической БД на основе известного множества реквизитов и функциональных зависимостей. Исходное множество функциональных зависимостей и реквизиты первичного ключа получаются так же, как при формировании множества отношений в ЗНФ. Алгоритм иллюстрируется тем же примером, что и для двухуровневой сети.

Алгоритм получения структуры иерархической БД

Исходные данные - список реквизитов и функциональных зависимостей в базе данных.

Шаг 1. Для каждой функциональной зависимости вида *A —> B* создать отношение Si(A,B). Каждый блок взаимно-однозначных соответствий также порождает отношение с ключом, равным старшему по объему понятия реквизиту.

В нашем примере будут созданы следующие отношения (ключи помечены знаком #):

S1 (НИИ #, Директор, Адрес),

S2(Отдел #, НИИ, Ксотр),

S3(Тема #, Датанач, Датакон, Приор),

S4(ФИО #, Отдел),

S5(Тема #, Работа #, ФИО #, Прод),

S6(Тема #, Заказ #, Обфин).

Ш а г 2. Разделить отношения на группы по признаку: два отношения находятся в общей группе, если их ключи функционально определяют хотя бы один общий реквизит.

Для отношений S1 - S6 получаем две группы:

• S1, S2,S4, S5 (все ключи функционально определяют реквизит НИИ);

• S3, S6, S5 (все ключи функционально определяют реквизит ТЕМА).

Далее шаги 3, 4, 5 выполняются раздельно для каждой группы. Количество групп определяет количество иерархических БД.

Шаг 3. У всех пар отношений группы проверить условие для ключей отношений *Ki —> Kj* Если оно соблюдается, то из соответствующих отношений создается веерное отношение Wij(Si,Sj).

Ш а г 4. Найти в группе цепи веерных отношений и образовать из них дерево. Элемент цепи образуется по условию Wij -Wik

В нашем примере получим:

группа 1) -цепь W12(S1,S2), W24(S2,S4), W45(S4,S5) образует дерево;

группа 2) - цепей нет, но W35(S3,S5) и W36(S3,S6) образуют дерево.

Ш а г 5. Реквизиты, оставшиеся вне цепей на шаге 4, добавить в структуру тех отношений, где они будут неключевыми, либо в структуру отношений, соответствующих висячим вершинам дерева.

Ш а г 6. Если группы, полученные на шаге 2, содержат общие отношения, то необходимо решить вопрос о целесообразности установления логических связей между иерархическими БД.

Ш а г 7. Сократить список реквизитов в сегментах за счет удаления реквизитов зависимого отношения, общих в паре «основной - зависимый».

Итоговая иерархическая структура для рассматриваемого примера показана на рис 3. Она содержит две иерархические базы данных. В некоторых иерархических СУБД не допускается логическая связь баз данных, так как формально это является нарушением ограничения иерархической модели данных.

Рис. 2.8. Иерархическая база данных НИИ

Рис. 3 – Иерархическая база данных НИИ

Манипулирование иерархической базой данных происходит с применением включающего языка. Подпрограмма обращения к базе данных содержит ряд параметров, из которых мы будем использовать лишь код требуемой операции и одно или несколько условий выбора. По причинам, которые уже были перечислены при рассмотрении сетевой модели данных, для иерархической модели данных необходимы только операции выборки.

Минимальное множество вариантов выборки соответствует трем операциям:

1. GU - получить уникальную запись по известным значениям первичного ключа на каждом уровне дерева иерархической базы данных.

2. GN - получить следующую запись на том уровне дерева, где находится текущая запись после выполнения оператора СШ.

3. GNP - получить следующую запись на расположенном непосредственно ниже уровне дерева, относительно позиции, где находится текущая запись после выполнения оператора СШ или ОК.

Например, для запроса «получить информацию о преподавателе с кодом 1103 кафедры 11 факультета 01» потребуется оператор

```
GU Факультет (Факультет = "01")
   Кафедра (Кафедра = "11")
   Преподаватель (Преподаватель = "1103")
   print Преподаватель.
```

В этом примере названия отношений совпадают с названиями соответствующих первичных ключей.

Запрос «получить список всех студентов группы 10» реализуется следующей последовательностью операторов

```
GU Факультет (Факультет = "01")
```

Группа (Группа="10")
Студент M1: GN Студент
print Студент
goto M1.

Поскольку в операторе GU не указано условие выборки в отношении Студент, текущей записью станет первая запись этого отношения и далее циклическое повторение оператора (GN обеспечит требуемое извлечение всех записей о студентах группы 10. Выход из цикла произойдет в результате получения кода возврата «конец отношения» и эта ситуация должна проверяться средствами включающего языка.

ЗАДАНИЯ

Задание 1. Детализируйте приведенную ниже иерархическую структуру (рис. 4). Необходимые имена реквизитов выбрать самостоятельно. Проверьте соблюдение всех требований алгоритма получения структуры иерархической базы данных.

Рис. 4 – Структура ВУЗа

Задание 2. Для приведенной ниже иерархической структуры базы данных укажите минимально возможный набор реквизитов в отношениях.

Реквизиты: Музей, Город, Экспонат, Год, Выставка, ФИО реставратора.

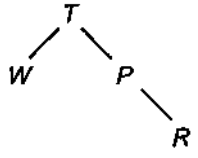
Отношения: W(Музей, Город), C(Экспонат, Год поступления), T(Экспонат, Год реставрации, ФИО), S(Выставка, Экспонат, Год выставки).

Веерные отношения: *(W,C), (C, S),* (C, *T).*

Названия музеев и выставок не повторяются.

Задание 3. Реализуйте иерархическую структуру. Разрешается добавлять или исключать имена реквизитов в отношениях. Разработайте пример записи иерархической базы данных.

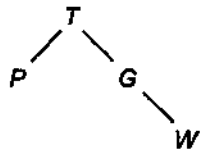
Реквизиты	**Отношения**
Автор	WАвтор, Год аттестации, Число статей)
Журнал	T(Автор, Ученая степень)
Статья	P(Автор, Журнал, Статья)
Год аттестации	R{(Журнал, Страна)
Число статей	
СтранаУченая степень	



Названия журналов не повторяются.

Задание 4. Реализуйте иерархическую структуру. Разрешается добавлять или исключать имена реквизитов в отношениях. Разработайте пример записи иерархической базы данных.

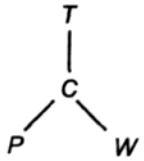
Реквизиты	**Отношения**
Учреждение	T(Учреждение, Отдел)
Отдел	G(Отдел, Тема)
Тема	P(Фио, Отдел)
Код оборудования	W(Тема, Код оборудования, Продолжительность)
Фио сотрудника	
Продолжительность работы	



Каждая тема выполняется в единственном отделе.

Задание 5. Реализуйте иерархическую структуру. Разрешается добавлять или исключать имена реквизитов в отношениях. Разработайте пример записи иерархической базы данных.

Реквизиты	**Отношения**
Город	T(Город, Страна)
Страна	C(Команда, Город)
Команда	P(Фио, Команда, Год рождения)
Фио игрока	W(Дата, Команда-соперник, Счет)
Год рождения	
Дата	
Команда-соперник Счет	



Названия команд не повторяются.

2.2 Лабораторная работа № 2 (1 час).

Тема: «Основы проектирования баз данных»

На занятии происходит привитие навыков решения задач с применением специальных методов и алгоритмов на основе специально подобранных примеров нарастающей сложности, выстроенных по типу **практикума** (особый вид проведения учебных занятий, покрывающий определенную тему образовательной области, имеющий целью практическое усвоение основных положений предмета), а также применяется метод ПОПС-формулы. На занятии студенты высказывают свою точку зрения, отношение к предложенной проблеме. ПОПС-формула применяется для закрепления изученного материала на лекционном занятии. Каждый студент выбирает одну из предложенных задач или создает свою. В случае необходимости студент может усложнить выбранную задачу, повторно решив ее, используя компьютерную симуляцию. После решения задачи студент защищает решение по методу ПОПС-формулы, высказывая: П-позицию (объясняет, в чем заключена точка зрения студента); О-обоснование (не просто объясняет свою позицию, но и доказывает); П-пример (при разъяснении сути позиции пользуется конкретными примерами); С-следствие (делает вывод в результате обсуждения определенной проблемы).

Оценка результатов практического занятия происходит согласно, предусмотренных баллов в рамках каждого пункта ПОПС-формулы.

2.2.1 Цель работы: изучить и проанализировать основы проектирования баз данных

2.2.2 Задачи работы:

1. Изучение основ проектирования баз данных
2. Анализ персональных баз данных

2.2.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Microsoft Office Access

2.2.4 Описание (ход) работы:

Основы проектирования баз данных.

Краткие сведения о программе

Поскольку база данных - это компьютерный эквивалент организованного списка информации, то ее назначение состоит в возможности быстро получать из нее точную информацию автоматизированным способом. Одним из самых популярных программных продуктов, обеспечивающих функции обработки, просмотра, поиска больших объемов данных является СУБД Access. Приложение Microsoft Access 2003 – система управления реляционными базами данных, предназначенная для работы на автономном ПК или в локальной вычислительной сети под управлением Microsoft Windows. Все преимущества Windows доступны в Access (например, обмен данными с другими приложениями Windows), базовые объекты интерфейса - **меню**, **панели инструментов**, **диалоговые окна** - работают точно так же, как в других продуктах Office или других приложениях для Microsoft Windows. Средства Access 2003 предоставляют пользователю возможность выполнять следующие операции:

- Проектирование базовых объектов ИС – двумерных таблиц с разными типами данных (разработка макетов таблиц).
- Установление связей между таблицами с поддержкой целостности данных.
- Ввод информации в таблицы, хранение, просмотр, корректировка с использованием различных средств контроля информации, индексирования таблиц.
- Создание и использование форм, запросов и отчетов для обработки данных таблиц в соответствии с требованиями пользователя.
- Подготовка собственных приложений (программ) на языке VBA (Visual Basic for Applications).
- Таблицу Access можно связать с данными, находящимися на другом компьютере или сервере, а также использовать таблицу, созданную в СУБД Paradox или Dbase. Данные Access легко комбинируются с данными Excel.

В СУБД Access предусмотрено много дополнительных возможностей. Удобные и гибкие средства визуального проектирования объектов с помощью **Мастеров** позволяют практически неподготовленному пользователю довольно быстро создать полноценную ИС на уровне таблиц, форм, запросов и отчетов. Система Access содержит набор инструментов для управления базами данных, включающий **Конструкторы** таблиц, форм, запросов и отчетов. Использование **Макросов** позволяет автоматизировать многие процессы программирования для разработки приложений пользователя. Для обеспечения взаимодействия Access с другими приложениями используются такие возможности языка программирования Си, как функции и обращения к Windows API (Application Programming Interface – интерфейс прикладных программ Windows). Access также располагает средствами для облегчения работы в Internet и создания приложений для Web.

Объектом обработки MS Access является файл базы данных, имеющий произвольное имя и расширение .mdb. Этот файл содержит основные объекты MS Access:

- **таблицы** - основа БД, столбцы которых называется полями, а строки – записями;
- **формы** – окна, позволяющие более наглядно отобразить информацию, содержащуюся в одной записи БД, позволяют упростить операции ввода и просмотра данных;
- **запросы** - предназначены для поиска и получения информации из БД по различным критериям;

▪ **макросы** – последовательности макрокоманд для расширения возможностей СУБД. С их помощью можно изменять ход выполнения приложения, открывать, фильтровать и изменять данные в формах и отчетах, выполнять запросы и создавать новые таблицы. **Макросы** могут использоваться для того, чтобы сделать часто повторяющиеся действия доступными в виде командных кнопок на формах, которые помогают менее опытным пользователям работать с вашей базой данных;

▪ **Модули** - это программы на Microsoft Visual Basic for Applications (VBA). В то время как макросы могут автоматизировать многие действия, VBA может использоваться для выполнения задач, которые слишком сложны для выполнения с помощью макросов.

▪ **Отчеты** отображают информацию из таблиц на экране компьютера или на бумаге в красиво отформатированном виде. **Отчет** может включать элементы информации, выбранные из нескольких таблиц и запросов, значения, вычисленные на основе информации из базы данных, и форматирующие элементы, такие, как заголовки, колонтитулы, названия и «шапки».

▪ **страницы** показывают все ярлыки страниц доступа к данным Internet и Intranet, дают возможность ввода, редактирования, просмотра и манипулирования данными из сети.

Каждый объект MS Access имеет имя, длина которого – не более 64 произвольных символов (за исключением точки и некоторых служебных символов). Любой объект БД (таблицу, форму, запрос, отчет) можно создать либо вручную в режиме **конструктора**, либо с помощью **мастера**.

В состав системы MS Access разработчики включили несколько готовых БД, которые находятся в папке Programs Files\MS Office\Office\Samples. Некоторые из них, например, Борей.mdb (Access 2000) или Acwzdat.mdb (Access 2003) можно использовать для знакомства с приемами работы с БД.

Основные операции с файлами БД (Создание, открытие, сохранение, закрытие) в Access, также как и в других Windows-приложениях выполняются с помощью стандартных диалоговых окон-файлеров, которые открываются командами меню **Файл**, или соответствующими кнопками. Настройка инструментария системы, выбор режима просмотра данных осуществляется с помощью команд пункта меню **Вид.** Настройка вида экрана, клавиатуры, рабочего каталога, порядка сортировки БД, вызов служебных программ и др. производится командами меню **Сервис**. При возникновении затруднений в работе с программой можно обратиться к справочной системе Access, которая имеет удобные и простые в использовании содержание, предметный указатель, систему поиска, журнал хронологии и закладки. Для получения полной справки используется пункт меню **Справка** или кнопка  на панели инструментов. **Контекстно-зависимая справка** вызывается клавишей **F1**. Access, также использует новое средство – **Помощник**, который отвечает на вопросы, выдает советы и справки об особенностях используемой программы.

Запуск программы. Создание БД

Запуск MS Access осуществляется:

- Через ярлык на рабочем столе;
- Кнопкой **Пуск/Программы/MS Office/MS Access.**

При открытии системы появляется диалоговое окно, в котором предлагается выбор: создать новую БД самостоятельно (пустую неструктурированную БД), использовать мастер или проект для разработки БД (пустую структурированную БД на основе шаблона) или открыть уже существующую ранее созданную БД, выбрав нужный файл из списка. В Access 2003 **область задач** у правой границы окна предлагает приступить к работе в программе в одном из перечисленных ниже режимов. Если программа Access была открыта ранее для работы с другой БД, для создания нового файла БД можно воспользоваться кнопкой  на панели инструментов или командой меню **Файл/Создать,** выбрав на вкладке **Общие** режим

База данных (в Access 2003 командой **Создать Файл** перейти в область задач **Создание файлов**, выбрать команду **Новая база данных**).

Microsoft Office Access 2003 включает набор **мастеров**, которые помогают быстро и просто создавать **базы данных** и другие объекты, такие, как **таблицы**, **запросы**, **формы** и **отчеты**.

Разработка таблицы. Типы данных в Access

Таблицы являются основой БД и состоят из **полей** и **записей**. Каждая запись таблицы содержит всю необходимую информацию об отдельном элементе БД. При разработке структуры таблицы для каждого поля задается уникальное имя длиной не более 64 символов (оно обычно совпадает с названием атрибута, значения которого будут храниться в этом поле), определяется тип данных поля. Значение типа поля может быть задано только в режиме конструктора. Ниже в таблице представлены типы данных Access и их описание.

Таблица 1.2. Типы данных в Access

Тип данных	Описание
Текстовый (Значение по умолчанию)	Текст или числа, не требующие проведения расчетов, например, номера телефонов (до 255 знаков)
Числовой	Числовые данные различных форматов, используемые для проведения расчетов
Дата/время	Для хранения информации о дате и времени с 100 по 9999 год включительно
Денежный	Денежные значения и числовые данные, используемые в математических расчетах, проводящихся с точностью до 15 знаков в целой и до 4 знаков в дробной части
Поле MEMO	Для хранения комментариев; до 65535
Счетчик	Специальное числовое поле, в котором Access автоматически присваивает уникальный порядковый номер каждой записи. Значения полей типа счетчика обновлять нельзя.
Логический	Может иметь только одно из двух возможных значений (Да/Нет, True/False)
Поле объекта OLE	Объект (например, электронная таблица Microsoft Excel, документ Microsoft Word, рисунок, звукозапись или другие данные в двоичном формате), связанный или внедренный в таблицу Access.
Гиперссылка	Строка, состоящая из букв и цифр и представляющая адрес гиперссылки. Адрес гиперссылки может состоять максимум из трех частей: текст, выводимый в поле или в элементе управления; путь к файлу (в формате UNC) или к странице (адрес URL). Чтобы вставить адрес гиперссылки в поле или в элемент управления, выполните команду **Вставка, Гиперссылка**
Мастер подстановок	Создает поле, в котором предлагается выбор значений из списка или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы. Это в действительности не тип поля, Access способ хранения поля

Создать таблицу в Access можно несколькими способами:

- В режиме **конструктора таблиц**, который предусматривает определение всех параметров структуры (макета) таблицы вручную;
- С помощью **мастера таблиц**. Мастер позволяет выбрать поля для данной таблицы из множества определенных ранее таблиц в одной из категорий – Деловые (контакты, клиенты, заказы…) или Личные (личное имущество, гости, рецепты…);

▪ В **режиме таблицы путем ввода данных** непосредственно в пустую таблицу. При сохранении таблицы Access анализирует данные и каждому полю присваивает необходимый тип данных и формат;

▪ С помощью **импортирования** – импорт данных и объектов из внешнего файла в текущую БД;

▪ С помощью **связывания** с другими таблицами – создание таблиц в текущей БД, связанных с таблицами внешнего файла;

▪ Использованием **мастера баз данных** для создания всей БД, содержащей все требуемые отчеты, таблицы, формы за одну операцию. Мастер БД создает новую БД целиком, его нельзя использовать для добавления новых таблиц, форм и отчетов в уже существующую БД.

Независимо от способа, использованного для создания таблицы, пользователь может использовать режим конструктора для изменения макета таблицы – добавления или удаления полей, установления ограничений на ввод значений, для установки значений по умолчанию.

Так как только режим конструктора позволяет пользователю сразу задать ту структуру таблицы, которая ему нужна, для разработки таблиц используется именно этот способ.

Вызов конструктора таблиц можно осуществить:

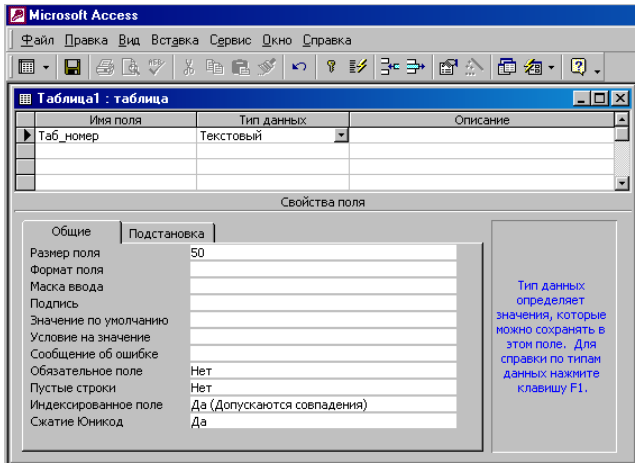
▪ объект **таблицы** (в окне открытой БД) – **Создать таблицу в режиме конструктора;**

▪ кнопка 

(в окне открытой БД) – **Конструктор**;

▪ меню **Вставка/Таблица** – **Конструктор**;

В режиме **конструктора** таблица содержит три колонки: **Имена полей**, **Тип данных**, **Описание. Имена полей** могут содержать русские, латинские буквы, цифры без пробелов. После присвоения имени задается **Тип поля** (по умолчанию программой выбирается Текстовый). **Описание** полей не обязательно, оно добавляет наглядности БД. Перемещение, удаление, добавление полей осуществляется с помощью мыши. Для перемещения поля его следует выделить (щелкнуть левой кнопкой мыши слева от имени перемещаемого поля в области маркировки записи) и «перетащить» в нужное место. Для удаления поля его следует выделить (выделить всю строку, соответствующую этому полю) и удалить клавишей **Delete** (или выполнить команду **Правка/Удалить**). Чтобы выделить несколько полей, следует выделение выполнять совместно с клавишей **Shift** (для смежных полей) или **Ctrl** (если поля расположены не подряд). Для добавления поля используется команда **Вставка/Строки**. Новая строка будет вставлена над строкой, в которой находится курсор. Удаление и вставку полей можно выполнять через контекстное меню, которое открывается правой кнопкой мыши или с помощью кнопок на панели инструментов  - добавить строки,  - удалить строки.



Прежде чем сохранить таблицу в файле БД, следует определить первичный ключ (для таблиц многотабличной БД). Для установки ключевого поля нужно предварительно его выделить и выполнить команду **Правка/Ключевое** поле в главном меню или команду **Ключевое поле** в контекстном меню, можно воспользоваться кнопкой  на панели инструментов. Если поле определено ключевым по ошибке, можно использовать команду **Правка/Отменить** ключевое поле или повторно щелкнуть на кнопке  .

В нижней части окна конструктора указываются **свойства полей**. Для их определения нужно:

- установить курсор в верхней части окна в нужное поле;
- перейти в нижнюю часть (**Свойства поля**) клавишей **F6** или мышью;
- установить курсор в строке нужного параметра;
- ввести с клавиатуры значение этого параметра или выбрать его из раскрывающегося списка. Заполнение некоторых свойств можно выполнить с помощью вспомогательного окна (мастера) **Построителя выражений**, вызываемого кнопкой  , расположенной справа от ячейки соответствующего свойства.

В окне **Свойства поля** определяются следующие параметры:

- **Размер поля** – максимальный размер данных в поле. Для текстовых значений по умолчанию устанавливается – 50, для числовых – Длинное целое (4 байта).

Основные размеры полей для числового типа данных приведены в таблице 1.3.

Таблица 1.3.

Тип	**Размер**
Байт (1 байт)	Целые числа от 0 до 255
Целое (2 байта)	Целые числа от -32768 до +32767
Длинное целое (4 байта)	Целые числа от –2 147 483 648 до +2 147 483 647
Одинарное с плавающей точкой (4 байта)	С точностью до 6 знаков от $-3{,}4 \times 10^{38}$ до $+3{,}4 \times 10^{38}$
Двойное с плавающей точкой (8 байт)	С точностью до 10 знаков от $-1{,}797 \times 10^{308}$ до $+1{,}797 \times 10^{308}$

- **Формат поля** – задает формат представления данных при выводе на экран или печать. Для типов данных **Числовой**, **Денежный** и **Счетчик** существует набор форматов:
  - **Стандартный** – отсутствуют разделители тысяч и знаки денежных единиц, число десятичных знаков зависит от точности данных (устанавливается по умолчанию);
  - **Денежный** или **евро** – символы валют и два знака после запятой;
  - **Фиксированный** – два знака после запятой;
  - **С разделителями разрядов** – разделители тысяч и два знака после запятой;
  - **Процентный** – два знака после запятой и символ % (значение увеличивается в 100 раз);
  - **Экспоненциальный** – запись числа в экспоненциальной форме (например, 3.46E+7 – это экспоненциальная форма числа $3{,}46 \times 10^7$ или $34{,}6 \times 10^6$;

Для типов **Дата/Время** существует набор форматов:

  - Полный формат даты (15.06.99 07:15:12 PM)
  - Длинный формат даты (Вторник, 20 сентября 2005);
  - Средний формат даты (20-сен-05)
  - Краткий формат даты (20.09.05)
  - Длинный формат времени (07:15:12 PM);
  - Средний формат времени (07:15 PM);
  - Краткий формат времени (07:15).

Для **Логического** типа определяются форматы: Да/Нет, Истина/Ложь, Вкл/Выкл.

- **Маска ввода** – автоматически изображает неизменяемые символы поля. При вводе данных в поле, заданное маской, достаточно заполнить пустые позиции. Маску ввода можно ввести с клавиатуры или использовать Мастер масок(). Для ввода маски с клавиатуры используются следующие обязательные символы:
  - 0 – цифра
  - L – буква

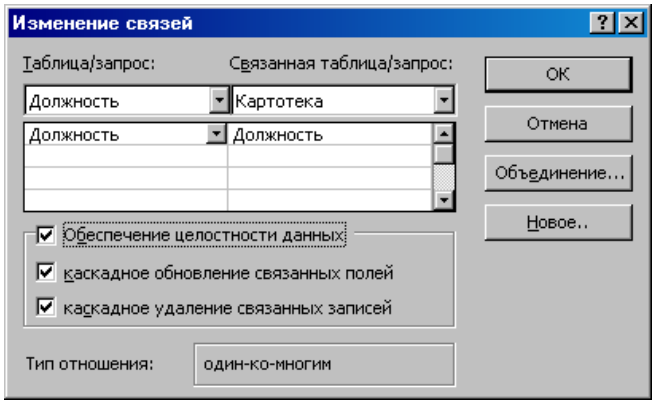
- A – буква или цифра
- @ - любой символ или пробел
- <(>) – преобразует все символы справа к нижнему (верхнему) регистру
- ! – маску следует заполнять справа налево.

- **Подпись поля** – определяет подпись для использования в формах и отчетах, если она отличается от имени поля.
- **Значение по умолчанию** – значение, которым программа автоматически заполняет данное поле во всех новых записях таблицы.
- **Условие на значение** – указывает, каким условиям должны удовлетворять значения, вводимые в данное поле. Оно задается выражением, состоящим из операторов сравнения и значений, используемых для сравнения (операндов). При вводе данных производится автоматическая проверка (контроль) соответствия данных указанным типам и проверка выполнения заданных условий. Условие задается либо с клавиатуры, либо с помощью построителя выражений (...).
- **Сообщение об ошибке** – позволяет задать текст, выводимый на экран, если значение не удовлетворяет **Условию на значение**.
- **Обязательное поле** – определяет, может ли это поле остаться незаполненным при вводе данных.
- **Индексированное поле** – задает построение индекса для полей с типом данных **Текстовый**, **Числовой**, **Денежный**, **Дата/Время** и **Счетчик** для ускорения выполнения запросов, поиска и сортировки.
- **Смарт-тэги.** Когда создается таблица в Access 2003, можно к каждому из полей применить несколько **смарт-тэгов**. Когда информация из такого поля отображается в таблице, форме или запросе, и вы подводите к ней указатель мыши, отображается кнопка со списком  **Действия для смарт-тэгов** (**Smart Tag Action**), и вы можете выполнить некоторые действия, связанные с данным типом информации (аналогично контекстному меню).

Ввод данных в таблицу осуществляется с клавиатуры, либо методом **подстановки** значений из другой таблицы (вкладка **Подстановка** в окне **Свойства поля**). Данные в текущую таблицу можно также **скопировать** из таблицы другого файла БД командами меню **Правка/Копировать** и **Правка/Вставить** или соответствующими кнопками на панели инструментов.

Установление связей между таблицами

Все таблицы БД могут функционировать самостоятельно, т.е. отдельно друг от друга. Но связи между ними помогают работать с БД более эффективно, предоставлять пользователю сведения в **большем** объеме и **более** конкретные. Связь между двумя таблицами устанавливается обычно по одноименным полям, совпадающим по **типу** и **размеру** (совпадение **имен** полей связи необязательно, но желательно). В одной из них поле связи **обязательно** должно быть **ключевым**. **Таблица, в которой поле связи является первичным ключом, называется *главной*, а ее первичный ключ – *уникальным ключом*. Таблица, в которой поле связи не является ключевым, называется *подчиненной*, а поле связи в ней – *внешним ключом*.** Установить, посмотреть, отредактировать Связи между таблицами можно в окне **Схемы данных**, которое открывается командой меню **Сервис/Схема данных** или кнопкой  - **схема данных**. При первоначальной установке связей открывается окно **Добавление Таблицы**, из которого нужные таблицы переносятся в окно схемы данных.



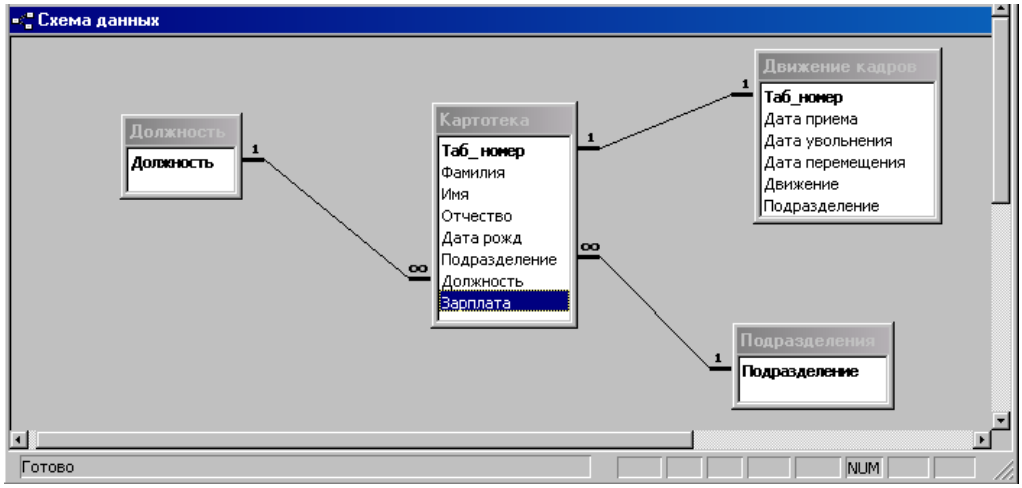
Для установки связи в одной из связываемых таблиц нужно выделить поле связи и совместить (перетащить) его с

соответствующим полем второй таблицы. В открывающемся диалоге **Изменение Связей** (его можно также открыть двойным щелчком на линии связи) нужно установить флажок у режима **Обеспечение целостности данных** и связанных с ним режимов **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей**. **Целостность данных** – это набор правил, защищающих данные от случайных изменений или удалений с помощью механизма поддержки корректности связей между связанными таблицами.

Контролировать целостность данных можно, если выполнены некоторые условия:

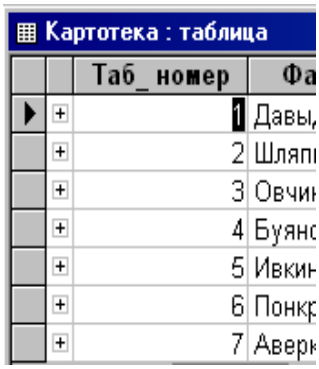
- поле связи является ключевым хотя бы в одной таблице;
- связанные поля имеют одинаковый тип данных. Поле **счетчика** является исключением, оно может быть связано с числовым полем, если для него определен тип **Длинное целое**;
- связываемые таблицы принадлежат одной БД Access.

Система автоматически определяет тип устанавливаемой связи: **один-к-одному**, **один-ко-многим**, **многие-ко-многим** (в нижней части диалога). На линии связи в окне схемы данных появляются значки «**1**» и «∞», обозначающие отношение «**один**» или «**многие**». Пример схемы данных изображен на рисунке.



Тип создаваемой связи зависит от полей, для которых определяется связь:

- связь **Один-ко-многим** создается в том случае, когда только в одной из связываемых таблиц поле связи является ключевым или имеет уникальный индекс, т.е. значения в нем не повторяются;
- связь **Один-к-одному** создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы;
- связь **Многие-ко-многим** фактически представляет две связи типа **один-ко-многим** через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, общих для двух других таблиц.



Если связь между таблицами системой определена, то при просмотре главной таблицы слева от первого поля радом с полосой выделения появится колонка со знаками «+». Щелчок на «+» откроет подчиненную таблицу.

Поиск информации в БД

Одной из основных функций СУБД является **поиск** информации в БД и ее предоставление для просмотра или печати – **выборка**. В зависимости от информационных потребностей пользователя можно использовать простые приемы поиска данных или более сложные, конкретизирующие условия отбора данных с помощью непростых выражений. Microsoft Office Access 2003 предоставляет различные инструменты, которые можно

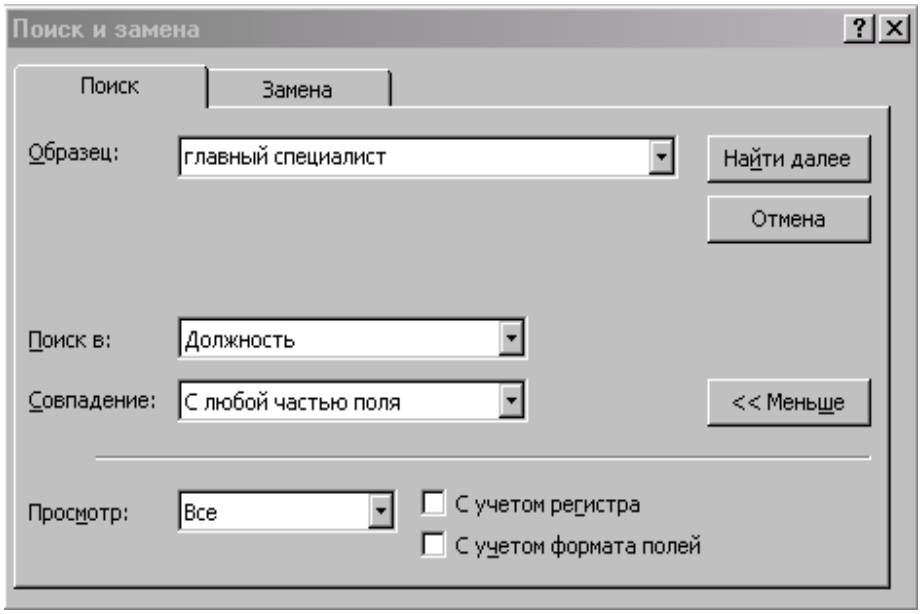
использовать для организации отображения информации из базы данных и для поиска конкретных частей этой информации.

Классификацию средств поиска в Access можно представить в виде таблицы-схемы.

Таблица 1.4.

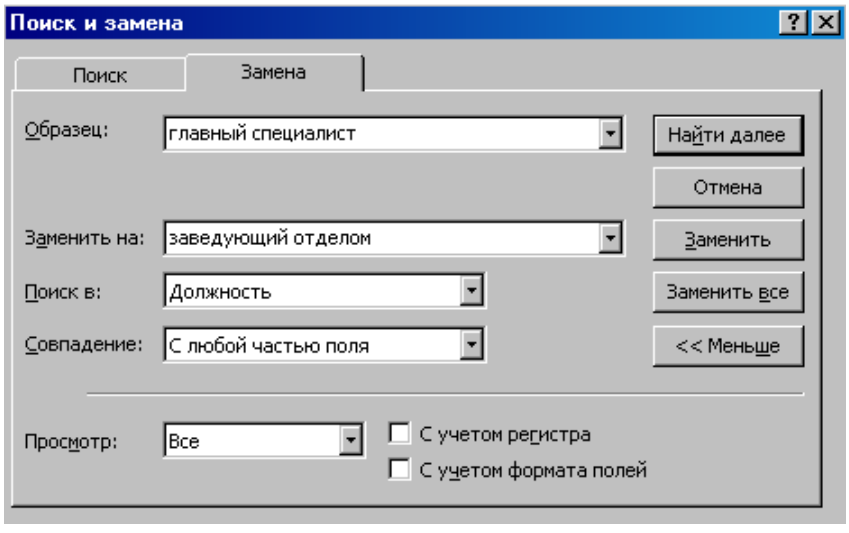
<table>
<tr><td colspan="4">Поиск данных в БД</td></tr>
<tr><td colspan="2">Простые средства поиска</td><td colspan="2">Запросы</td></tr>
<tr><td>Найти
Заменить
Сортировка</td><td>Фильтры</td><td>На выборку</td><td>На изменение</td></tr>
<tr><td rowspan="5"></td><td>по выделенному</td><td>итоговые</td><td>на обновление</td></tr>
<tr><td>по форме</td><td>параметрические</td><td>на удаление</td></tr>
<tr><td>по вводу</td><td rowspan="2">перекрестные</td><td>на добавление</td></tr>
<tr><td rowspan="2">расширенный фильтр</td><td rowspan="2">на создание таблицы</td></tr>
<tr><td></td></tr>
</table>

К простейшим видам поиска относятся операции с использованием команд **Правка/Найти** и **Правка/Заменить**. Диалог **Найти** (**Заменить**) можно открыть также кнопкой  . В появившемся окне нужно ввести образец искомых данных (значение, фраза или выражение), задать режимы поиска, щелкнуть на кнопке Найти далее.



Если значение найдено системой, курсор остановится в таблице в соответствующей ячейке, при перемещении курсора он будет останавливаться в ячейках со значениями, совпадающими с образцом поиска. Если необходимо большое число одинаковых данных заменить новым значением, используется команда **Правка/Заменить**.

В диалоге замены вводится образец значения, которое надо найти и значение, на которое нужно заменить найденное, устанавливаются режимы поиска и замены. Для выполнения поиска используется кнопка Найти далее, для замены кнопка Заменить или Заменить все.



В условиях поиска могут быть использованы операции сравнения (>,<,>=,<=,=,<>), а также подстановочные символы:

* - любая цифра или символ;

? – любой текстовый символ;

[ ] – любой один символ из заключенных в скобки;

! - любой один символ, кроме заключенных в скобки;

- - любой символ из диапазона;

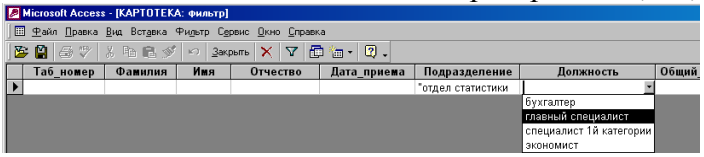
- любая цифра.

Использование фильтров. Сортировка

Фильтр используется для отображения в окне только тех записей БД, которые удовлетворяют условиям пользователя. Фильтры просты в использовании и представляют собой одноразовые запросы без имени. Их можно применять к таблице, запросу или форме. В фильтре всегда отображаются все поля и используются данные только одной таблицы. В СУБД Access используются фильтры четырех видов:

Фильтр по выделенному фрагменту – оставляет в таблице записи, совпадающие по значению какого-либо поля. В таблице предварительно курсором помечается значение отбора (в любой строке).

Фильтр по форме (**изменение фильтра**) – представляет таблицу в виде пустой табличной строки с пиктограммой списка в каждом поле, где можно задавать критерии отбора с использованием логических операторов And, Or, Not.



Фильтр по вводу - устанавливается через контекстное меню нужного поля таблицы. Позволяет найти записи, удовлетворяющие нескольким условиям одновременно.

Расширенный фильтр – позволяет с помощью специального бланка фильтра, в котором задаются условия отбора, в открытой форме или таблице выделять подмножество записей, отвечающих заданным условиям. В бланке фильтра можно задать порядок сортировки для одного или нескольких полей.

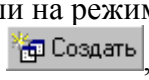
Фильтры устанавливаются командой **Записи/Фильтр…** или с помощью кнопок на панели инструментов: - **фильтр по выделенному**, - **изменить фильтр**, - **удалить фильтр**.

Сортировка – определяет порядок вывода записей в таблице или запросе. Выполнить сортировку данных можно с помощью команды меню **Записи/Сортировка** или кнопками , предварительно установив курсор в поле ключа сортировки.

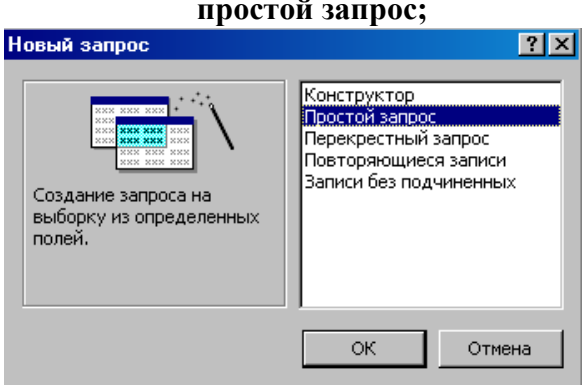
Запросы в Access

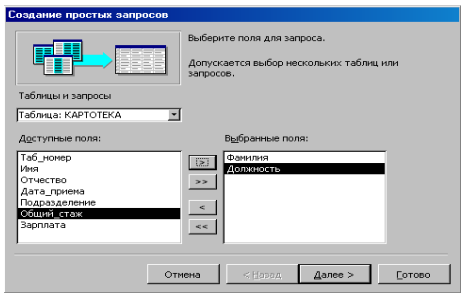
Запрос к БД является мощным средством Access для поиска, выборки, просмотра, анализа и изменения данных одной или нескольких таблиц. Он представляет собой сформулированную информационную потребность. В работе с запросом выделяется два этапа: создание запроса (проектирование) и выполнение. В результате выполнения запроса из БД (из одной или нескольких таблиц) выбирается информация в соответствии с критериями запроса. Запросы можно создавать самостоятельно – в режиме **Конструктора** или – с помощью **Мастера запросов**, который автоматически выполняет основные действия в ответ на задаваемые пользователю вопросы. Непрофессиональному пользователю рекомендуется формировать запрос с помощью **Мастера**, а критерии поиска, выборки, изменения данных добавлять в режиме **Конструктора запросов**.

Мастер запросов можно открыть из окна БД, выделив объект **Запросы**

- двойным щелчком мыши на режиме **Создание запроса с помощью мастера;**
- или щелкнув на кнопке 

, и выбрав из предложенного списка режимов – **простой запрос;**





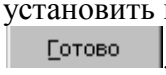
- выполнив команду меню **Вставка/Запрос/Простой запрос**.

В окне мастера запросов нужно:

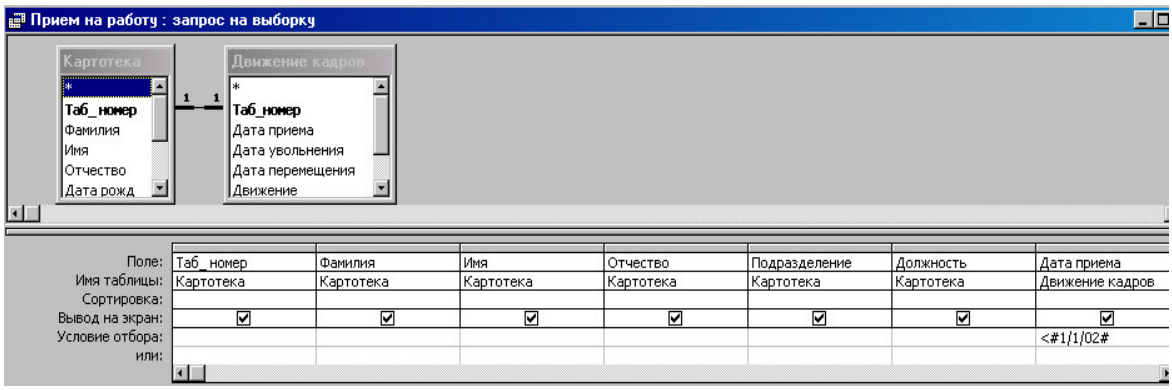
- выбрать объект (таблицу или запрос), на основе которого создается запрос;
- из перечисленного списка полей выбрать те, которые содержат данные для просмотра или изменения (т.е. поля, которые должны выводится на экран в результате выполнения запроса).

Для перехода к каждому следующему окну мастера запросов используется кнопка 

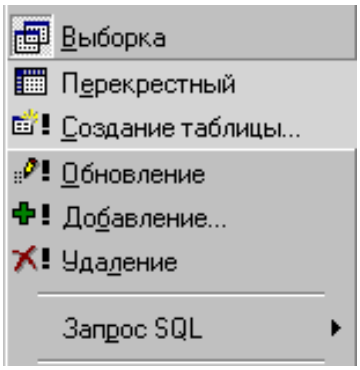
. Для просмотра результатов запроса на экране используется кнопка 

. Для определения условий отбора записей можно перейти в режим Конструктора запроса. Для этого после задания имени запроса, нужно установить переключатель в позицию режима **Изменить макет запроса**, щелкнуть на кнопке 

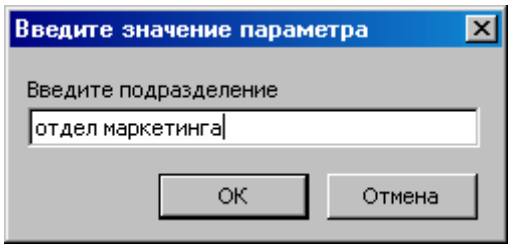
. В конструктор можно перейти из режима просмотра кнопкой  или командой меню **Вид/Конструктор.**



Окно конструктора запроса состоит из двух областей. В верхней области система отображает структуру таблиц (запросов) из которых были выбраны поля в создаваемый запрос, указывая связи между таблицами. В нижней части окна формируется структура текущего запроса: в столбцах отображаются имена выбранных полей (строка **Поле**), имена таблиц, из которых были выбраны поля (строка **Имя таблицы**), условия сортировки записей (строка **Сортировка**), режим отображения полей на экране (строка **Вывод на экран**, помеченные галочкой поля будут выводиться на экран в результате выполнения запроса), критерии поиска и выборки записей (Строки **Условие отбора**, **или**). В качестве **Условия отбора** могут быть выражения, даты, числовые значения, текст, которые либо вводятся с клавиатуры, либо с помощью **Построителя выражений** -  , либо с помощью команды контекстного меню **Построить**. Все критерии отбора, указанные в одной строке должны выполняться совместно (логическое объединение And). Если критерии отбора указываются в разных строках “**Условие отбора”** и “**Или”,** предусматривается выбор между этими условиями (логический выбор Or).



Поскольку в Access существует несколько видов запросов (на выборку, на создание, на обновление, перекрестный, итоговый и др.) при создании запроса необходимо указать его вид. По умолчанию формируется запрос **на выборку**. Тип запроса может быть преобразован в любой другой командами меню **Запрос** или кнопкой . Запрос запускается на выполнение из окна конструктора - кнопкой (команда меню **Запрос/Запуск**), из окна БД - кнопкой Открыть, или двойным щелчком мыши на имени запроса. При выполнении запроса **на выборку** система извлекает записи из таблиц и формирует результирующий набор данных, который выглядит как таблица, но по сути не является ею. Такой набор данных является динамическим и не хранится в БД. При сохранении запроса сохраняется только его структура: перечень таблиц, состав полей, условия отбора, тип запроса. Такой способ организации обеспечивает возможность многократного выполнения запроса с учетом любых изменений в базовых таблицах.

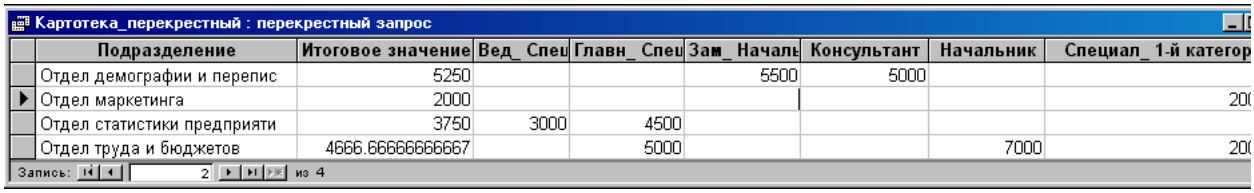


ПАРАМЕТРИЧЕСКИЕ ЗАПРОСЫ . Для того чтобы сделать запрос на выборку более универсальным, можно вместо конкретного значения отбора включить в запрос **параметр**, который обеспечит диалоговый режим выполнения запроса. Для этого в строку **Условие отбора** в конструкторе запроса вводится фраза в квадратных скобках [ ], которая будет выводиться на экран в качестве подсказки пользователю в процессе выполнения запроса, например [Введите фамилию], [Введите должность сотрудника].

Аналогичные параметры могут быть заданы для нескольких полей запроса одновременно, причем имя каждого параметра должно быть уникальным. Для корректировки параметров можно использовать команду меню **Запрос/Параметры**.

ИТОГОВЫЕ ЗАПРОСЫ . Иногда при выполнении поиска и отбора данных необходимо найти какое-либо итоговое значение, например, сумму значений или среднее значение в поле. Запросы, выполняющие вычисления с группой записей, называются **итоговыми**. Для их создания в режиме конструктора нужно добавить новую строку **Групповая операция** командой меню **Вид/Групповые операции** или кнопкой Σ. В этой строке все поля запроса будут заполнены значением **Группировка**. Открыв список группировки в поле, для которого рассчитывается итоговое значение (это должно быть числовое поле), нужно выбрать одну из возможных функций (Sum, Avg, Min, Max, Count и др.).

ПЕРЕКРЕСТНЫЕ ЗАПРОСЫ – особый тип итоговых запросов, представляющих результаты поиска в виде матрицы. **Перекрестный запрос** состоит из трех полей, из которых одно определяет заголовки строк, второе – заголовки столбцов, третье (обязательно должно быть числовым) – определяет значения, по которым рассчитываются итоги. Если в перекрестный запрос выбираются поля из разных таблиц и запросов, то предварительно нужно создать простой запрос на выборку, содержащий данные поля, а затем на его основе создать перекрестный запрос. Пример перекрестного запроса для расчета **средней зарплаты** по всем **подразделениям** организации, для каждой **должности** сотрудников приведен ниже.



Картотека_перекрестный : перекрестный запрос

Подразделение	Итоговое значение	Вед_ Спец	Главн_ Спец	Зам_ Начальн	Консультант	Начальник	Специал_ 1-й категор
Отдел демографии и перепис	5250			5500	5000		
Отдел маркетинга	2000						20(
Отдел статистики предприяти	3750	3000	4500				
Отдел труда и бюджетов	4666.66666666667		5000			7000	20(

Запись: 2 из 4

В перекрестных запросах можно использовать дополнительные условия отбора и сортировку.

ЗАПРОСЫ НА ИЗМЕНЕНИЕ БД – за одну операцию приводят к **необратимым** изменениям данных в нескольких записях. Существует четыре типа запросов на изменение, каждый из которых помечается в окне БД специальным значком:

Запрос на удаление – удаляет группу «старых» или неиспользуемых записей, удовлетворяющих заданным условиям, одной или нескольких таблиц.

Запрос на обновление – изменяет значения в группе записей одной или нескольких таблиц.

Запрос на добавление – пополняет таблицы БД группой новых записей. Данные могут быть добавлены из другой БД.

Запрос на создание таблицы – создает новую таблицу на основе данных одной или нескольких таблиц, которая будет являться подмножеством исходных таблиц.

ЗАПРОСЫ SQL – особый тип запросов, которые создаются при помощи конструкций SQL (Structured Query Language – язык структурного программирования) для обращения к серверам баз данных в сетевой версии СУБД. Этот тип запросов обычно используется опытными пользователями, имеющими навыки программирования, и для начинающих пользователей достаточно сложен.

ЗАПРОСЫ С ВЫЧИСЛЯЕМЫМИ ПОЛЯМИ Можно создать запрос с дополнительным полем, которого нет в исходной таблице. Такое поле Access заполняет автоматически в соответствии с заданным для него выражением и присваивает ему имя **ВыражениеN** (таких полей может быть в запросе несколько). Для построения выражений используется **Построитель выражений**, вызываемый кнопкой или командой **Построить** в контекстном меню.

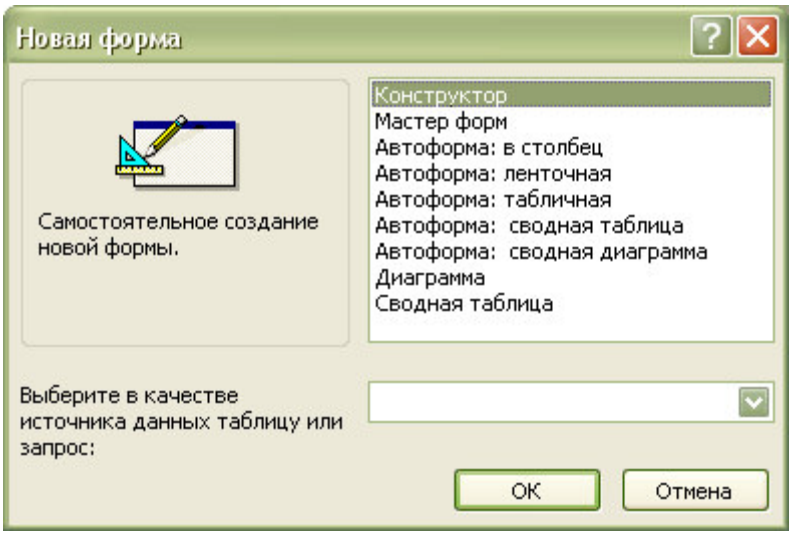
Формы в Access

Формы используются для организации удобного интерфейса БД. Они обеспечивают более гибкий способ ввода, просмотра и редактирования данных, позволяют вывести на экран одну или несколько записей в виде электронного бланка (шаблона). Формы могут использоваться для просмотра данных таблиц, содержащих очень большое количество полей. Такая форма может вмещать несколько десятков полей на одном экране. Кроме того, режим формы позволяет отобразить графические объекты и содержимое полей типа OLE, что невозможно в режиме таблицы. Для ввода новой информации, редактирования, поиска или удаления существующей информации используются **элементы управления** формы. Каждое текстовое поле формы связано с конкретным полем из таблицы. Таблица является источником записей, а поле является источником элемента управления. Каждый элемент управления имеет некоторое количество свойств, таких, как **стиль шрифта**, **размер** и **цвет**, которые вы можете изменять для улучшения внешнего вида формы (в режиме конструктора).

Форму можно создать разными способами: **Автоматически**, с помощью **Мастера форм**, с помощью **Конструктора форм**. Любой из них можно выбрать из списка окна **Новая форма**, которое открывается из окна БД:

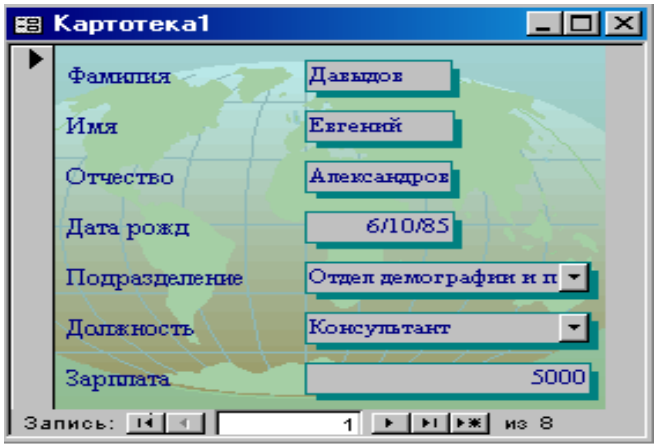
- Щелчком на кнопке Создать, предварительно должен быть выделен объект **Формы**;

- Командой меню **Вставка/Форма**.

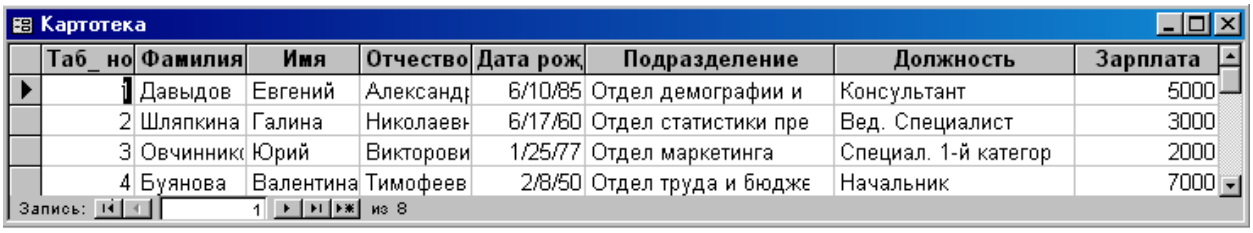


Автоматический режим – **Автоформа** - предусматривает создание формы без участия пользователя, все параметры настройки формы программа задает по умолчанию. Пользователь лишь выбирает таблицу, для которой создается форма, определяет тип формы (в столбец, ленточная или табличная) и задает имя. Автоформы являются частными случаями мастера форм.

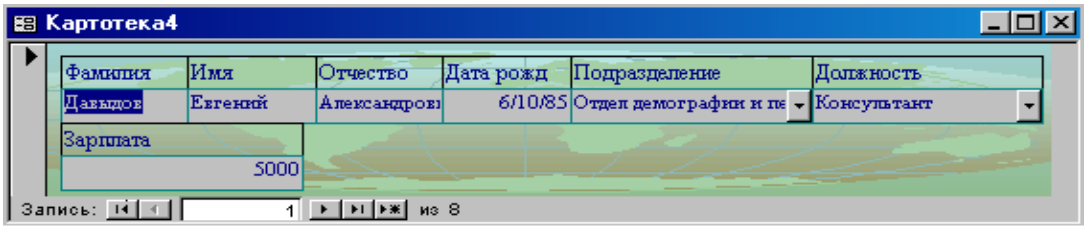
Мастер форм позволяет полуавтоматически создать форму на основе выбранных полей. Выбор полей для формы выполняется аналогично формированию запроса. Мастер форм позволяет настроить цвет и стиль оформления формы, а также ее вид: **в один столбец, ленточный, табличный, выровненный** (основные виды форм представлены на рисунке).



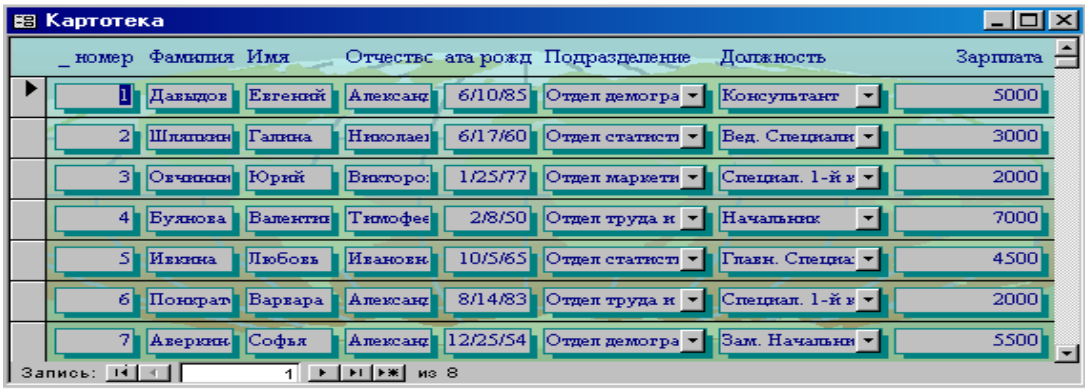
форма в один столбец



табличная форма



выровненный вид формы



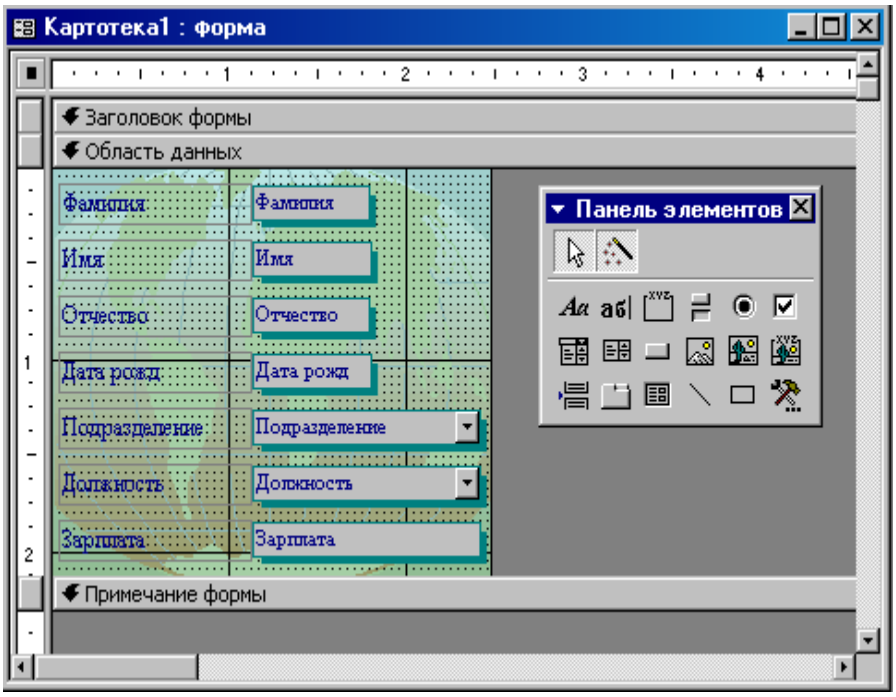
ленточная форма

Способ создания формы с помощью мастера наиболее удобен для начинающих пользователей, так как параметры формы настраиваются в режиме диалога.

Режим **Диаграмма** позволит создать форму со встроенной диаграммой, режим **Сводная таблица** – со сводной таблицей Excel.

Конструктор форм позволит создать форму самостоятельно, но для непрофессиональных пользователей это сложный и долгий процесс. Поэтому создавать форму удобнее с помощью **Мастера**, а дополнять и улучшать ее внешний вид – в режиме **Конструктора**.

Для открытия готовой формы в режиме конструктора можно воспользоваться кнопкой  на панели инструментов или командой меню **Вид/Конструктор**. В конструкторе для добавления в форму элементов (полей ввода, вычисляемых полей, надписей, кнопок и др.) используются кнопки **панели элементов,** а также окно **свойств формы (**см. рис.).



Макет формы состоит из разделов: **область данных** (содержит данные из таблиц), **заголовок формы** (верхняя часть первой страницы), **примечание формы** (нижняя часть последней страницы), **верхний и нижний колонтитулы** (при печати формы). Выделяя мышью отдельный элемент формы можно с помощью команд меню и кнопок панели инструментов изменять настройки данного элемента, а также перемещать элемент в пределах формы. В режиме конструктора вы можете изменять размер трех основных областей формы: **Заголовка формы**, **Области данных** и **Примечания формы**. Можно настраивать любой из этих разделов формы, добавляя и удаляя надписи, перемещая надписи и элементы управления текстом, и добавляя логотипы и другую графику. Наиболее популярные элементы управления находятся в **Панели элементов**.

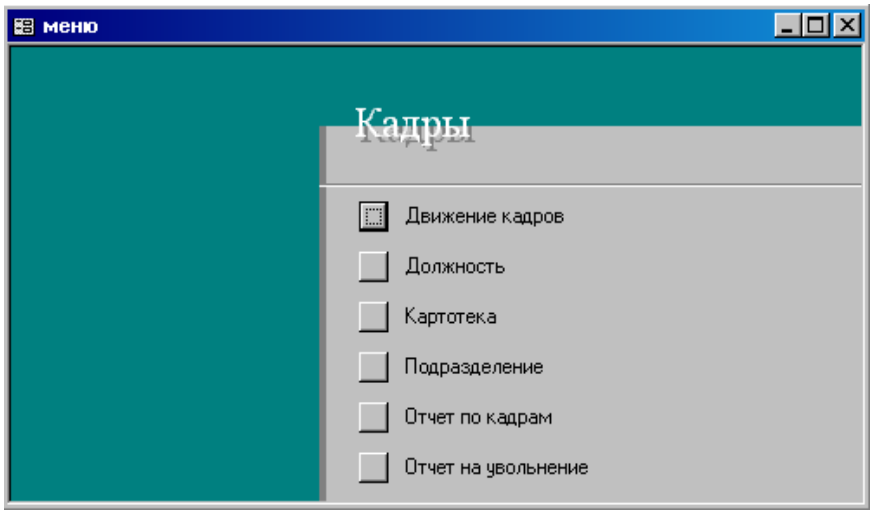
Элементы формы могут выполнять различные действия, определяемые макросом или процедурой VBA, прикрепленных к ним.

Для возврата в режим просмотра формы используется кнопка  или команда меню **Вид/Форма**.

Если разрабатывается форма на основе нескольких таблиц, Access предложит создать либо **подчиненную**, либо **связанную** форму. Такие формы называются **сложными**. **Подчиненная форма** представляет собой форму внутри другой формы. Первичная форма называется **главной**, а внутренняя – **подчиненной** (субформой). Подчиненная форма удобна для просмотра данных таблиц и запросов, связанных по типу **один-ко-многим** или **один-к-одному**. В подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме. Главная форма может отображаться только в виде столбца, подчиненная – в столбцовом, табличном или ленточном виде. Главная форма может иметь несколько подчиненных форм и до семи уровней вложенности.

Для манипулирования записями в режиме формы используются те же способы, что и в режиме таблицы – поиск, замена, сортировка, фильтрация данных. Настройка параметров просмотра и печати форм, так же как и таблиц, производится с помощью команд **Параметры страницы**, **Предварительный просмотр** и **Печать** в меню **Файл**.

Особым типом формы является **Кнопочная форма** (**Кнопочное меню**) – форма с расположенными на ней элементами управления, кнопками с поясняющими надписями (см. рисунок).



Щелчком на командной кнопке можно открыть соответствующую таблицу, форму или отчет. Это удобный инструмент работы с БД. Кнопочное меню можно создать вручную с помощью **конструктора** (однако для этого требуется достаточно много времени) или воспользоваться специальной программой - **Диспетчером кнопочных форм**. Созданную с помощью диспетчера кнопочную форму можно отредактировать или дополнить в режиме конструктора. Диспетчер кнопочных форм открывается командой меню **Сервис/Служебные программы/Диспетчер кнопочных форм**.

Отчеты в Access

Для предоставления конечного документа по работе с БД данные таблиц и запросов могут быть оформлены в виде отчета. **Отчет** – гибкое и эффективное средство для организации просмотра и распечатки итоговой информации, позволяющее представить данные на печать в желаемом формате. В Access 2003 вы можете создать **главный отчет**, который служит как оболочка для одного или нескольких **суботчетов**.

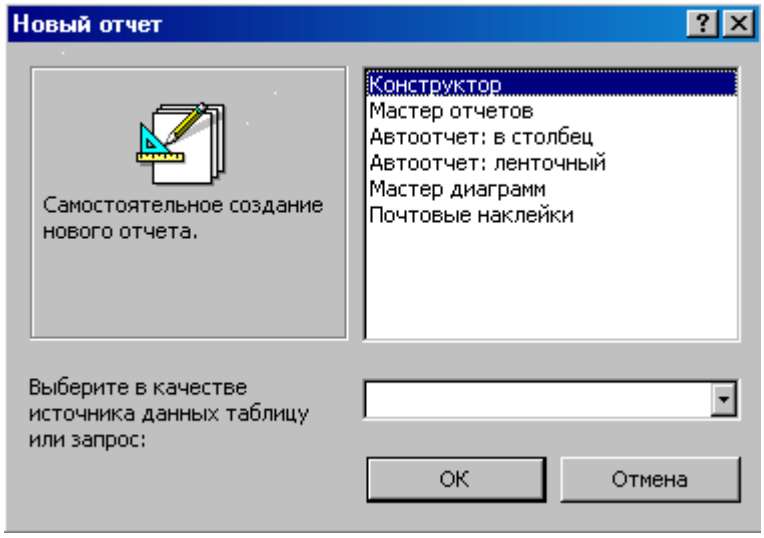
В ваш **отчет** и/или **суботчеты** вы можете добавлять **элементы управления** и можете устанавливать **свойства** этих элементов управления. Вы можете отображать информацию из одной или нескольких записей из одной или нескольких таблиц или запросов, и можете устанавливать несколько наборов заголовков и нижних колонтитулов.

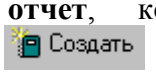
Отчет можно дополнить результатами сложных расчетов и статистической обработки, рисунками и диаграммами. Также как и формы, отчеты могут содержать вычисляемые поля. В формах значения вычисляемых полей рассчитываются на основе полей в текущей записи. В отчетах вычисляемые поля (итоги) формируются на основе группы

записей, страницы записей или всех записей отчета. Вы можете сделать отчет или суботчет более полезным, выполнив вычисления в самом отчете. Для создания выражений, по которым нужно выполнить вычисления, вы можете использовать **Построитель Выражений**.

Предварительный просмотр печати позволяет вам проверять отчеты перед их печатью.

Просмотр **Образец** отображает только ту информацию, которой достаточно для того, чтобы показать все элементы отчета.



Разработать новый отчет, как и любой другой объект БД можно самостоятельно (в режиме **Конструктора**) или с помощью **Мастера отчетов**. Режим создания отчета можно выбрать в окне **Новый отчет**, которое открывается кнопкой 

из окна БД или командой меню **Вставка/Отчет**. Создать отчет можно также двойным щелчком на соответствующем режиме в окне БД: **Создание отчета в режиме конструктора** или **Создание отчета с помощью мастера**. Мастер отчетов является наиболее удобным средством, как для начинающих, так и для опытных пользователей. Мастер отчетов состоит из нескольких диалоговых окон, содержащих приглашения ввести необходимые данные (диалоговый режим), и отчет создается на основании ответов пользователя. Такой способ создания отчета позволяет быстро разработать макет, на основе которого можно с помощью инструментов конструктора разработать высоко профессиональный документ, имеющий более привлекательный вид. Для начинающих пользователей также как и при создании формы, разработку отчета рекомендуется выполнять с помощью мастера, а детальную настройку отдельных элементов и добавление новых элементов – в режиме конструктора.

В зависимости от того, какой отчет создается – простой или сложный (с группировкой данных) – мастер предлагает различные варианты макетов отчета, образец каждого макета приводится в окне мастера.

Простейшие способы создания отчетов – разработка их в автоматическом режиме (**Автоотчет в столбец** и **Автоотчет ленточный**), пользователю достаточно лишь выбрать таблицу или запрос, на основе которого будет создаваться отчет.

Сформированный отчет выводится в окно предварительного просмотра печати.

Мастер диаграмм поможет создать отчет в виде диаграммы.

Мастер Почтовых наклеек создаст отчет, отформатированный для печати почтовых наклеек.

В режиме конструктора бланк отчета разделяется на несколько областей:

- **Область данных** - где размещаются записи из таблицы или запроса;
- **Заголовок отчета** – размещается в начале бланка отчета;
- **Область примечаний** – размещается в конце отчета;
- **Верхний** и **нижний колонтитулы** (в области нижнего колонтитула обычно выводятся номера страниц отчета и текущая дата);
- **Заголовок** и **Примечание группы** – присутствуют в бланке отчета, если задана группировка записей. Количество этих разделов - по числу уровней группировки.

Группа – набор записей, выделяемых по определенному признаку. Она состоит из заголовка, области данных и примечания. Группировка позволяет разбить записи на логические группы и отобразить промежуточную и итоговую информацию для каждой.

Для настройки отдельных элементов отчета, а также для добавления новых используются команды меню **Правка**, **Вид**, **Вставка**, **Формат**, **кнопки панелей инструментов** и **панель элементов**, а также команды контекстного меню для выделенного элемента. Для перехода в режим конструктора используется кнопка  . Возврат в окно просмотра осуществляется кнопкой  .

Обмен информацией с другими программами

Access 2003 облегчает импорт информации из многочисленных форматов, созданных другими программами. Если информация из другой программы все еще активно обновляется, вы можете связать вашу базу данных Access с этой информацией, не извлекая ее из оригинальной программы для дальнейшей обработки.

Вы можете импортировать в новую или существующую таблицу из рабочего листа Excel весь лист или именованный диапазон. Вы также можете импортировать из рабочего листа отдельные поля.

Вы можете использовать для импорта в вашу базу данных Access текстовых файлов с разделителями и с фиксированной шириной данных **Мастер импорта (Import Wizard)**.

Вы легко можете импортировать один или несколько стандартных объектов Access: таблицы, запросы, формы, отчеты, страницы, макросы и модули.

Вы можете импортировать в Access 2003 данные из некоторых версий **dBASE**, **Lotus 1-2-3** и **Paradox**. Можно импортировать информацию в первоначальном виде ,и обработать ее в Access, а можете перенести ее в другую программу, такую, как **Excel** или **Word**, и обработать ее перед импортом в Access.

Можно импортировать в Access 2003 документ, сохраненный с помощью другой программы в формате HTML. При попытке импортировать HTML-документ в Access 2003, он проведет разбор документа и идентифицирует все, что выглядит как структурированные данные. Затем эти данные можно просмотреть, и решить, импортировать ли это или нет.

Вы можете импортировать в Access 2003 файлы **Extensible Markup Language** (**XML**). XML состоит из данных и схемы, которая описывает структуру данных. Программы, которые экспортируют XML, могут объединять данные и схему в одном файле, а могут создавать два отдельных файла. Если программа экспортирует два отдельных файла, для импорта данных и структуры в Access 2003 нужно иметь оба этих файла.

Вы можете экспортировать информацию из вашей базы данных Access в различные другие форматы, набор которых зависит от объекта, который вы пытаетесь экспортировать.

Вы можете оставить данные в другой программе и создать связь с ней.

Вы также можете скопировать данные из вашей базы данных в другую программу через **буфер обмена**. Быстрым способом совместного с Word или Excel использования информации в базе данных Access 2003 является использование кнопки **Связи с Office** (OfficeLinks) на панели инструментов. Вы можете объединить данные из таблицы с документом письма Word, опубликовать таблицу в документе Word или экспортировать таблицу в электронную таблицу Excel.

Доступ к базе данных

- Чтобы облегчить другим пользователям доступ к вашим данным и работу с ними, а также исключить **случайное** изменение или удаление данных, вы можете создать **кнопочную форму** и установить параметры запуска.
- Можно создать **экран заставки**, который появляется при открытии вашей базы данных. Он может содержать рекламу, графический логотип или анимацию, а также может предоставлять пользователям полезные инструкции.
- В Access 2003 имеется несколько утилит, которые вы можете использовать для поддержания хорошей работоспособности вашей базы данных - **Сжать и восстановить базу данных (Compact and Repair Database)**, **Анализ быстродействия (Performance Analyzer),**

Архивариус (Documenter), и Найти и восстановить (Detect and Repair). Вы можете поддерживать работу вашего приложения, используя возможности этих утилит до того, как Access укажет, что в вашей базе данных возникли проблемы.

Организация безопасности информации

Для организации безопасности информации в базе данных в Access 2003 можно использовать различные средства:

Можно зашифровать базу данных, что не защищает ее от открытия и просмотра в Access, но не дает возможности прочитать ее содержимое тем людям, у которых нет установленной копии Access.

Вы можете задать для своей базы данных пароль, предотвратив ее открытие неавторизованными пользователями.

Можно совместно использовать базу данных в рамках локальной сети (LAN) и с помощью сетевой безопасности, которая используете для защиты информации в сети, ограничить действия пользователей.

Чтобы запретить одновременное обновление одной и той же записи несколькими пользователями (при сетевом доступе к БД), можно реализовать **пессимистическое блокирование**, которое блокирует запись на все время ее редактирования, или **оптимистическое блокирование**, которое блокирует запись только на короткое время, пока Access сохраняет изменения.

Вы можете преобразовать вашу базу данных в новую версию (Design Master), а затем создать **реплики** этой главной базы данных для распространения в удаленных местах. Затем эти **реплики** могут синхронизироваться и объединять свои изменения с главной базой данных. После того как все изменения будут записаны, все реплики обновляются в соответствии с текущей информацией из Design Master и отправляются обратно в их расположения.

Можно разделить базу данных на «**табличную**» **часть** базы данных, которая содержит таблицы, и **интерфейсную часть** базы данных, которая содержит все остальные объекты базы данных. Вы можете хранить **табличную часть** базы данных на **сервере**, а **интерфейсную часть** распространить среди других пользователей, которые работают с данными. Они могут использовать все созданные вами объекты (за исключением таблиц, которые недоступны для редактирования) или создавать свои собственные.

Если вы добавили в базу данных процедуры VBA, вы можете защитить ваш код VBA с помощью **пароля**, либо сохранив базу данных как файл Microsoft Database Executable (MDE). Если вы устанавливаете для кода пароль, этот код остается доступным для редактирования всеми, кто знает этот пароль. Если вы сохраняете базу данных как файл MDE, другие пользователи могут запускать код, но не могут его просматривать или редактировать.

2.3 Лабораторная работа №3 (2 часа).

Тема: «Создание однотабличной базы данных»

На занятии происходит привитие навыков решения задач с применением специальных методов и алгоритмов на основе специально подобранных примеров нарастающей сложности, выстроенных по типу **практикума** (особый вид проведения учебных занятий, покрывающий определенную тему образовательной области, имеющий целью практическое усвоение основных положений предмета), а также применяется метод ПОПС-формулы. На занятии студенты высказывают свою точку зрения, отношение к предложенной проблеме. ПОПС-формула применяется для закрепления изученного материала на лекционном занятии. Каждый студент выбирает одну из предложенных задач или создает свою. В случае необходимости студент может усложнить выбранную задачу, повторно решив ее, используя компьютерную симуляцию. После решения задачи студент защищает решение по методу

ПОПС-формулы, высказывая: П-позицию (объясняет, в чем заключена точка зрения студента); О-обоснование (не просто объясняет свою позицию, но и доказывает); П-пример (при разъяснении сути позиции пользуется конкретными примерами); С-следствие (делает вывод в результате обсуждения определенной проблемы).

Оценка результатов практического занятия происходит согласно, предусмотренных баллов в рамках каждого пункта ПОПС-формулы

2.3.1 Цель работы: изучить и проанализировать создание однотабличной базы данных

2.3.2 Задачи работы:

1. Изучение функционирования однотабличной базы данных
2. Разработка базы данных однотабличной базы данных

2.3.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Microsoft Office Access

2.3.4 Описание (ход) работы:

Создание однотабличной базы данных.

Постановка задачи

Предположим, что организация (предприятие) создает информационную систему для автоматизации процессов учета кадров и контроля изменения заработной платы сотрудников. Личная карточка сотрудника содержит анкетные данные, информацию о дате приема на работу, должности, месте работы, общем стаже работы и размере заработной платы.

ИС должна функционировать в двух режимах:

- первичной загрузки данных – ввод информации из личных карточек;
- текущей обработки информации.

Пользователем базы данных может быть сотрудник отдела кадров или бухгалтерии.

ИС должна обеспечивать:

- контроль ввода новой информации;
- получение информации о сотрудниках по различным критериям;
- возможность обновления данных, в частности: перерасчет заработной платы, начисление различных надбавок и премий и т.п;
- возможность добавления в картотеку сведений о новых сотрудниках и удаления устаревшей информации;
- расчет итоговых показателей по различным подразделениям и должностям;
- подготовку кадровых и финансовых отчетов.

Задание на разработку базы данных

1. Разработать базу данных ***КАДРЫ*** с числом сотрудников 15 человек.
2. База данных содержит одну таблицу ***Картотека*** со следующей структурой записи: *Табельный номер, Фамилия, Имя, Отчество, Дата приема на работу, Подразделение, Должность, Общий стаж, Зарплата*.
3. Предусмотреть контроль ввода данных, задав ограничения на ввод данных по полю *Подразделение*, полю *Должность* и полю *Общий стаж*.
4. Ввод данных в таблицу осуществить посредством формы.
5. Создать запрос на выборку с параметром для получения информации о сотрудниках отдельных *подразделений* (отделы статистики, маркетинга, бухгалтерия).
6. Разработать запрос с параметром для получения информации о любом сотруднике организации по его *фамилии*.
7. Разработать запрос с параметром для получения информации о сотрудниках организации по *должности*.
8. Разработать запрос на сотрудников организации, принятых на работу в 2005 году.

9. Создать запрос о начислении *премии* работникам отдела статистики.

10. Рассчитать среднюю зарплату по должностям сотрудников.

11. Разработать запрос на удаление из базы данных записей, соответствующих уволенным сотрудникам.

12. Предусмотреть возможность увеличения заработной платы на 15% тем работникам, чей *общий стаж* работы превысил 20 лет.

13. Разработать формы для просмотра результатов запросов.

14. Создать отчет ***Кадры***, сгруппировав информацию по *подразделениям*, с расчетом средней, минимальной и максимальной зарплаты для сотрудников каждого подразделения.

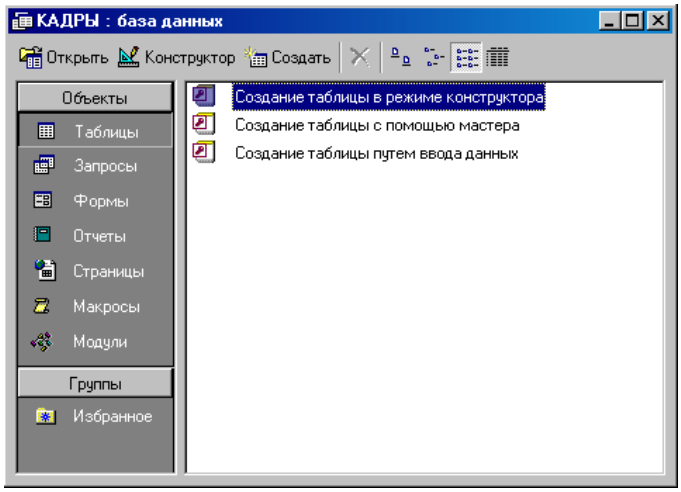
15. Разработать отчет ***Премия*** с группировкой по *должности* и сортировкой записей в алфавитном порядке *фамилий* сотрудников.

16. Разработать кнопочную форму для удобства работы с базой данных.

Технология выполнения работы

1. Для открытия файла новой базы данных ***КАДРЫ*** выполните следующее:

- В меню **Пуск** выберите команду **Программы/Microsoft Office/Microsoft Access**, на экране откроется окно программы, предлагающее открыть базу данных в одном из трех режимов.
- Выберите режим **Новая база данных**, <OK>.
- В диалоге **Файл новой базы данных** в списке поля **Папка** выберите папку, в которой будет храниться файл базы данных ***КАДРЫ***;
- В строке **Имя Файла** введите с клавиатуры имя ***КАДРЫ*** вместо заданного по умолчанию имени db1.mdb (каждый следующий новый файл базы данных Access будет именовать соответственно db2, db3 и т.д.);
- В строке **Тип Файла** оставьте заданный программой по умолчанию тип файла **Базы данных Microsoft Access (*.mdb)**, <Создать>. На экране открывается окно базы



данных ***КАДРЫ***.

Для создания структуры таблицы ***Картотека*** воспользуемся режимом **конструктора** таблиц, для этого:

- В окне базы данных слева нужно выбрать объект **Таблицы**, режим **Создание таблицы в режиме конструктора** (можно щелкнуть на кнопке <Создать>, и из предложенного списка способов создания таблицы выбрать конструктор, <OK>);
- В окне конструктора таблиц определите имена полей и их характеристики (тип и размер) в соответствие с Таблицей 2.1:

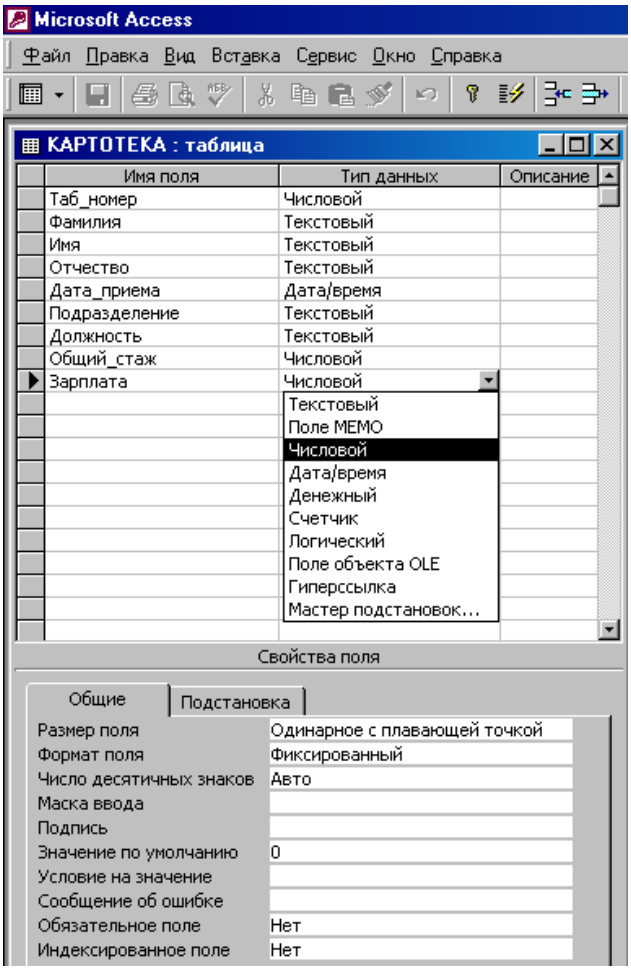
Таблица 2.1.

Имя поля	**Тип поля**	**Размер поля**
Таб_номер	Числовой	Целое
Фамилия	Текстовый	15
Имя	Текстовый	15

Отчество	Текстовый	15
Дата_приема	Дата/Время	Краткий
Подразделение	Текстовый	35
Должность	Текстовый	35
Общий_стаж	Числовой	Целое
Зарплата	Числовой	Одинарное с плавающей точкой

В именах полей, состоящих из двух и более слов, не рекомендуется использовать в качестве разделителя слов знак пробела. Слова в имени поля можно разделять символом «_», или вообще не разделять, записывая каждое следующее слово с заглавной буквы (напр. *ОбщийСтаж*).

- Тип значений поля выбирается из списка в столбце **Тип поля** конструктора таблиц, размер поля устанавливается на вкладке **Общие** в окне **Свойства поля** в нижней части окна конструктора. Установите курсор в поле *Таб_номер*, на вкладке **Общие** в окне **Свойства поля**, в строке параметра **Размер поля** выберите из списка размер числового значения **Целое**;



- Установите курсор в поле *Фамилия*, в окне свойств поля в строке **Размер поля** введите с клавиатуры значение **15** (максимальное количество символов для записи фамилии, обычно задается по количеству символов в самой длинной фамилии списка, по умолчанию программа задает размер текстового поля 50 символов);
- Аналогично задайте размер для текстовых полей *Имя*, *Отчество*, *Подразделение*, *Должность*;
- Установите курсор в поле *Дата_приема*, в окне свойств поля в строке **Формат поля** откройте список и выберите **Краткий формат даты** (дд.мм.гггг);
- Размер поля *Общий_стаж* задайте аналогично полю *Таб_номер*;
- Установите курсор в поле *Зарплата*, в окне свойств поля в строке **Размер поля** выберите из списка **Одинарное с плавающей точкой** (дробное числовое значение с десятичной точкой), в строке **Формат поля** выберите из списка **фиксированный** (с двумя знаками после запятой).

Для этого поля можно также определить тип **Денежный**. Числовые значения данного типа отображаются в таблице в виде вещественных чисел с символом денежной единицы.

- Структура таблицы в режиме конструктора должна иметь следующий вид:

2. Чтобы ИС могла обеспечивать контроль вводимой информации, нужно задать ограничения на ввод значений в поле *Подразделение*, поле *Должность* и поле *Общий_стаж*, для этого выполните следующее:

- Выберите три подразделения, на работников которых создается база данных, например – **отдел статистики, отдел маркетинга, бухгалтерия.**
- Установите курсор в поле *Подразделение*. В окне свойств поля в строке параметра **Условие на значение** щелкните по кнопке ... **Построитель выражений**. В окне Построителя выражений введите значение «**отдел статистики»**, затем щелкните на кнопке

оператора <**Or**> (логический оператор выбора ИЛИ), введите значение «**отдел маркетинга»**, щелкните на кнопке оператора <**Or**>, введите значение «**бухгалтерия»**, вернитесь в окно конструктора таблицы кнопкой <OK>. В строке **Условие на значение** появится выражение:

«отдел статистики» Or «отдел маркетинга» Or «бухгалтерия»

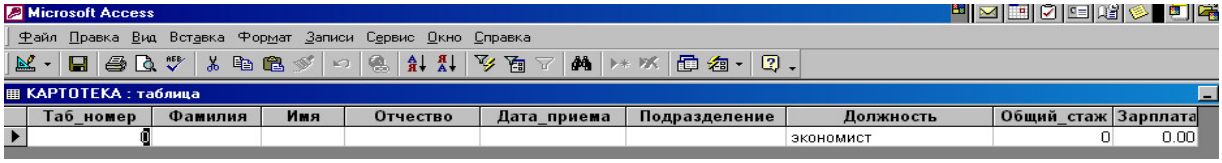
Построитель выражений – специальная программа, помогающая пользователю правильно записать условия и расчетные выражения с использованием синтаксиса допустимых конструкций языка VBA (Visual Basic for Application).

Для удобства ввода значений в поля, на которые наложены ограничения, можно сформировать текст сообщения об ошибке, которое будет выводиться на экран при попытке ввода значения, противоречащего условию. Например, если в поле *Подразделение* ввести значение **отдел кадров**, не предусмотренное условием, Access может вывести на экран следующее сообщение: **«Введено неверное подразделение, повторите ввод!!!»** (текст сообщения может быть произвольным, по усмотрению пользователя).

- Для формирования текста сообщения об ошибке установите курсор в поле *Подразделение*, в окне свойств поля в строке параметра **Сообщение об ошибке**, введите с клавиатуры текст: **ВВЕДЕНО НЕВЕРНОЕ ПОДРАЗДЕЛЕНИЕ, ПОВТОРИТЕ ВВОД!!!**
- Аналогично задайте ограничения на значения поля *Должность*, выбрав должности: **главный специалист, специалист 1й категории, экономист, бухгалтер**. Сформируйте текст сообщения об ошибке для этого поля.

Чтобы облегчить ввод больших объемов информации, для тех полей, в которых часто повторяются значения можно задать значение по умолчанию. Access автоматически будет заполнять указанным значением соответствующее поле таблицы (числовые поля автоматически по умолчанию заполняются нулевыми значениями). Пользователь при необходимости может менять его на другое значение в процессе заполнения таблицы. Например, для поля *Должность* можно определить значение по умолчанию – **экономист**, т.к. оно чаще других встречается в этом поле. Для этого установите курсор в поле *Должность*, в окне свойств поля в строке **Значение по умолчанию** откройте построитель выражений, введите слово **экономист**, вернитесь в конструктор кнопкой <OK>.

- Для задания ограничения на ввод значений в поле *Общий_стаж* в строке параметра **Условие на значение** с помощью построителя выражений запишите условие <**50**, сообщение об ошибке для этого поля сформулируйте аналогично полям *Подразделение* и *Должность*.
- Сохраните созданную структуру таблицы, щелкнув на кнопке , задайте имя таблицы ***Картотека***, в ответ на предложение определить ключевое поле щелкните по кнопке <Нет> (в противном случае программа автоматически создаст дополнительное поле *Код* и определит его как ключевое. В однотабличной базе данных определение ключевого поля - необязательно).
- Перейдите в режим просмотра таблицы, выбрав команду меню **Вид/Режим таблицы.**



3. Для заполнения таблицы создайте форму:

- В окне базы данных выберите объект **Формы**, щелкните на кнопке <Создать>, из предложенного списка режимов создания формы выберите **Мастер форм**, <OK>.
- В окне мастера форм в поле **Таблицы и запросы** выберите таблицу ***Картотека***. В области **Доступные поля** перечислены все поля таблицы.
- Поочередно выделяя каждое поле перенесите его в область **Выбранные поля** кнопкой (для быстрого выполнения этой рации можно воспользоваться кнопкой ,

которая переносит сразу все поля из области **Доступные поля** в область **Выбранные поля**), щелкните на кнопке <Далее>.

- Из предложенного списка видов форм выберите **В один столбец**, <Далее>;
- Выберите подходящий вам стиль оформления формы, <Далее>.
- Задайте имя формы ***Кадровый Состав***, оставьте установленный по умолчанию режим **Открыть форму для просмотра**, щелкните на кнопке <Готово>.

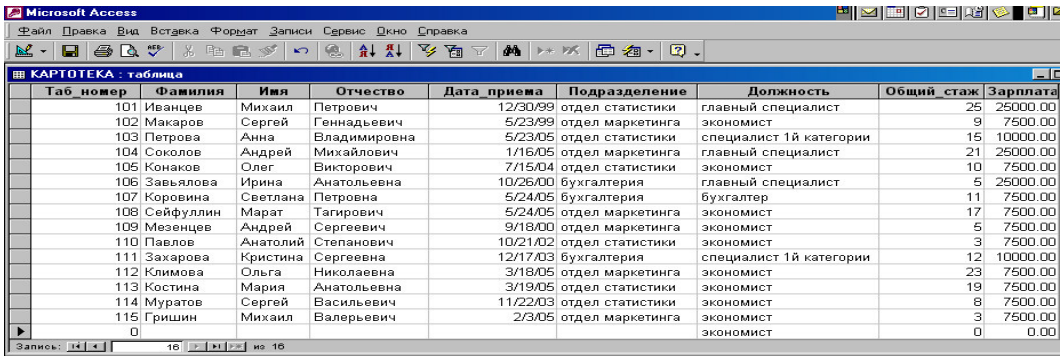
Для более быстрого создания формы можно воспользоваться автоматическим режимом, для этого: из окна базы данных для объекта **Формы** нужно щелкнуть на кнопке <Создать>, выбрать из списка **Автоформа: в один столбец**, внизу в поле списка выбрать таблицу ***Картотека***, <OK>. Форма создается программой без участия пользователя и содержит все поля таблицы, параметры оформления задаются автоматически, пользователь задает лишь имя формы.

- Форму заполните данными на сотрудников организации в соответствии с Таблицей 2.2. Для перемещения от одного поля к другому можно использовать клавишу **Tab** (в обратном направлении **Shift+Tab**), для перехода к следующей записи воспользуйтесь кнопками прокрутки в строке **Запись** в нижней части окна формы: к следующей записи, к предыдущей записи, к первой записи, к последней записи, добавление пустой строки для новой записи. Проверьте, как будет осуществляться контроль вводимой информации в поля *Подразделение*, *Должность*, *Общий_стаж*. Для этого попробуйте ввести в эти поля значения, противоречащие заданным для них условиям в любой записи: в *Подразделение* введите – **отдел кадров**, в *Должность* введите **консультант**, в поле *Общий_стаж* укажите значение **55**. На каждое недопустимое значение программа реагирует сообщением об ошибке ввода данных.

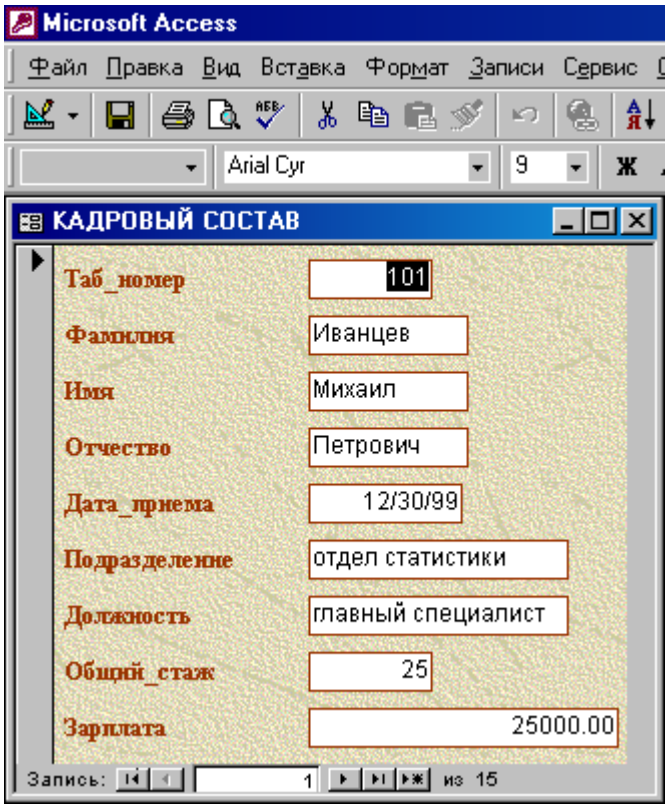
Таблица 2.2.

Таб. номер	**Фамилия**	**Имя**	**Отчество**	**Дата приема**	**Подразде-ление**	**Должность**	**Общий Стаж**	**Зар-плата**
101	Иванцев	Михаил	Петрович	10.12.00	отдел статистики	главный специалист	25	25000
102	Макаров	Сергей	Геннадьевич	23.05.99	отдел маркетинга	экономист	9	7500
103	Петрова	Анна	Владимировна	23.05.05	отдел статистики	специалист 1й категории	15	10000
104	Соколов	Андрей	Михайлович	16.01.05	бухгалтерия	главный специалист	21	25000
105	Конаков	Олег	Викторович	15.07.04	отдел статистики	экономист	10	7500
106	Завьялова	Ирина	Анатольевна	26.10.00	бухгалтерия	специалист 1й категории	5	10000
107	Коровина	Светлана	Петровна	24.05.05	бухгалтерия	бухгалтер	11	7500
108	Сейфуллин	Марат	Тагирович	24.05.05	отдел маркетинга	экономист	17	7500
109	Мезенцев	Андрей	Сергеевич	18.09.00	отдел маркетинга	экономист	5	7000
110	Павлов	Анатолий	Степанович	21.10.02	отдел статистики	экономист	3	7000
111	Захарова	Кристина	Сергеевна	17.12.03	бухгалтерия	специалист 1й категории	12	10000
112	Климова	Ольга	Николаевна	18.03.05	отдел маркетинга	экономист	23	7500
113	Костина	Мария	Анатольевна	19.03.05	отдел статистики	экономист	19	7500
114	Муратов	Сергей	Васильевич	22.11.03	отдел статистики	экономист	8	7500
115	Гришин	Михаил	Валерьвич	03.02.05	отдел маркетинга	экономист	3	7000

- Закройте форму ***Кадровый Состав***, предварительно сохранив ее щелчком на кнопке  на панели инструментов.
- В окне базы данных выберите объект **Таблицы** и откройте таблицу ***Картотека*** для просмотра кнопкой <Открыть>, или двойным щелчком левой кнопки мыши. Заполненная таблица и соответствующая ей форма должны выглядеть следующим образом:



Таб_номер	Фамилия	Имя	Отчество	Дата_приема	Подразделение	Должность	Общий_стаж	Зарплата
101	Иванцев	Михаил	Петрович	12/30/99	отдел статистики	главный специалист	25	25000.00
102	Макаров	Сергей	Геннадьевич	5/23/99	отдел маркетинга	экономист	9	7500.00
103	Петрова	Анна	Владимировна	5/23/05	отдел статистики	специалист 1й категории	15	10000.00
104	Соколов	Андрей	Михайлович	1/16/05	отдел маркетинга	главный специалист	21	25000.00
105	Конаков	Олег	Викторович	7/15/04	отдел статистики	экономист	10	7500.00
106	Завьялова	Ирина	Анатольевна	10/26/00	бухгалтерия	главный специалист	5	25000.00
107	Коровина	Светлана	Петровна	5/24/05	бухгалтерия	бухгалтер	11	7500.00
108	Сейфуллин	Марат	Тагирович	5/24/05	отдел маркетинга	экономист	17	7500.00
109	Мезенцев	Андрей	Сергеевич	9/18/00	отдел маркетинга	экономист	5	7500.00
110	Павлов	Анатолий	Степанович	10/21/02	отдел статистики	экономист	3	7500.00
111	Захарова	Кристина	Сергеевна	12/17/03	бухгалтерия	специалист 1й категории	12	10000.00
112	Климова	Ольга	Николаевна	3/18/05	отдел маркетинга	экономист	23	7500.00
113	Костина	Мария	Анатольевна	3/19/05	отдел статистики	экономист	19	7500.00
114	Муратов	Сергей	Васильевич	11/22/03	отдел статистики	экономист	8	7500.00
115	Гришин	Михаил	Валерьевич	2/3/05	отдел маркетинга	экономист	3	7500.00
0						экономист	0	0.00

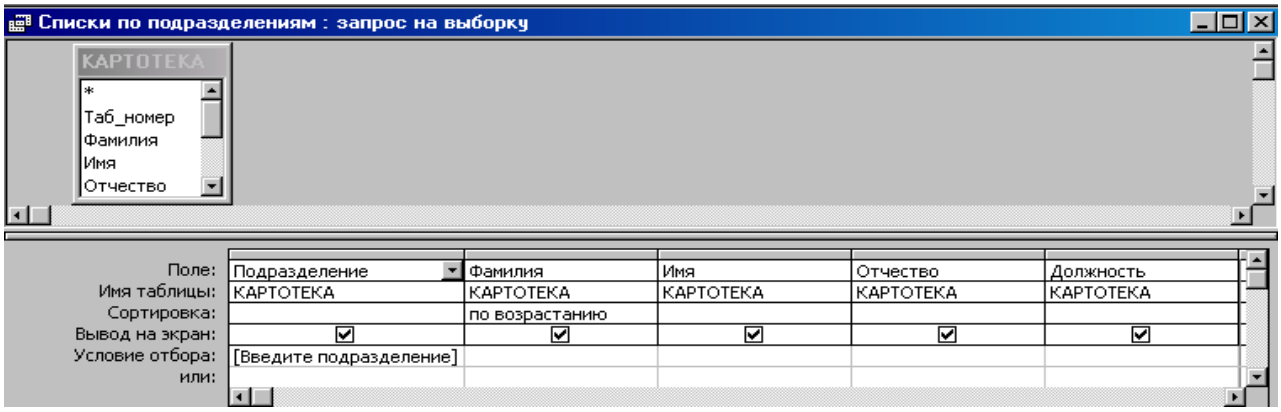


- Откройте вновь форму ***Кадровый Состав***, зарегистрируйте новых сотрудников, используя кнопку со звездочкой в строке **Запись**, введите 1-2 новых записи (данные задайте произвольно).
- Произведите сортировку данных по возрастанию *Даты_приема*: установите курсор в это поле и выполните команду **Записи/Сортировка;**
- Посмотрите в таблице ***Картотека*** новые данные командой меню **Вид/Режим таблицы**, вернитесь в окно формы и закройте ее с сохранением.
- Последовательно создайте три **Автоформы** с различным размещением полей: **ленточная**, **выровненная**, **табличная**. Сохраните одну из них по своему усмотрению под именем ***Кадровый Состав 2***.

3. Разработайте **запрос на выборку с параметром** для формирования списков сотрудников по отдельным подразделениям. Для этого выполните следующие действия:

- В окне базы данных выделите объект **Запросы**, щелкните на кнопке <Создать>, из предложенного списка режимов создания запросов выберите **Простой Запрос** (создание запроса с помощью мастера), <ОК>.
- В окне мастера запросов в поле списка **Таблицы и запросы** выберите таблицу ***Картотека***, из списка полей таблицы в области **Доступные поля** выделите и перенесите в область **Выбранные поля** следующие: *Подразделение*, *Фамилия*, *Имя*, *Отчество*, *Должность*, <Далее>.
- Задайте имя запроса ***Списки По Подразделениям***, щелкните на кнопке <Готово>. На экран выводится таблица с данными обо всех сотрудниках в составе пяти выбранных полей. Для того, чтобы отобрать из этого списка данные о сотрудниках одного какого-либо подразделения перейдите в режим **конструктора запросов** используя кнопку  на панели инструментов или команду меню **Вид/Конструктор** (перейти в режим конструктора

сразу, не открывая запрос в режиме просмотра, можно установив режим **Изменить макет**



запроса до нажатия кнопки <Готово>).

- В поле *Подразделение* в строке **Условие отбора** введите с клавиатуры параметр запроса **[Введите Подразделение]** (это приглашение будет выводиться на экран при каждом запуске запроса на выполнение).
- Определите порядок вывода записей в запросе. Установите курсор в поле *Фамилия* в строку **Сортировка**, откройте список и выберите **по возрастанию**.
- Сохраните запрос с внесенными в его структуру изменениями (кнопка ).
- Запустить запрос на выполнение можно непосредственно в режиме конструктора, выберите в меню команду **Запрос/Запуск** или воспользуйтесь кнопкой  на панели инструментов. В ответ на приглашение запроса **Введите подразделение** с клавиатуры введите слово **бухгалтерия**, Enter. В результате выполнения запроса на экране будет получен список работников бухгалтерии.


Списки по подразделениям : запрос на выборку

Подразделение	Фамилия	Имя	Отчество	Должность
бухгалтерия	Завьялова	Ирина	Анатольевна	главный специалист
бухгалтерия	Захарова	Кристина	Сергеевна	специалист 1й категории
бухгалтерия	Коровина	Светлана	Петровна	бухгалтер
				экономист

Выполните запрос для других подразделений для получения списков сотрудников отделов статистики и маркетинга.

2.4 Лабораторная работа №4 (2 часа).
Тема: «Итоговое обзорное занятие»

На занятии происходит привитие навыков решения задач с применением специальных методов и алгоритмов на основе специально подобранных примеров нарастающей сложности, выстроенных по типу **практикума** (особый вид проведения учебных занятий, покрывающий определенную тему образовательной области, имеющий целью практическое усвоение основных положений предмета), а также применяется метод ПОПС-формулы. На занятии студенты высказывают свою точку зрения, отношение к предложенной проблеме. ПОПС-формула применяется для закрепления изученного материала на лекционном занятии. Каждый студент выбирает одну из предложенных задач или создает свою. В случае необходимости студент может усложнить выбранную задачу, повторно решив ее, используя компьютерную симуляцию. После решения задачи студент защищает решение по методу ПОПС-формулы, высказывая: П-позицию (объясняет, в чем заключена точка зрения студента); О-обоснование (не просто объясняет свою позицию, но и

доказывает); П-пример (при разъяснении сути позиции пользуется конкретными примерами); С-следствие (делает вывод в результате обсуждения определенной проблемы).

Оценка результатов практического занятия происходит согласно, предусмотренных баллов в рамках каждого пункта ПОПС-формулы.

2.4.1 Цель работы: повторение всех вопросов, которые решались на предыдущих занятиях

2.4.2 Задачи работы:

1. Повторить этапы создания базы данных
2. Повторить этапы и способы программирования баз данных
3. Повторить администрирование баз данных

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Microsoft Office Access

2.4.4 Описание (ход) работы:

Студенты на обзорном итоговом занятии повторяют пройденный материал дисциплины, выполняют пропущенные лабораторные работы и иные виды деятельности для увеличения количества полученных за семестр обучения баллов рейтингового контроля.

Разработал(и): _____ Д.А. Андриенко