ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

КемТИПП

КЕМЕРОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ
ИНСТИТУТ ПИЩЕВОЙ ПРОМЫШЛЕННОСТИ

М.В. Баканов, В.В. Романова, Т.П. Крюкова

БАЗЫ ДАННЫХ.
СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Учебное пособие

Для студентов вузов

Кемерово 2010

УДК
ББК
Г

Рецензенты:
А.И. Колокольникова доцент, канд. техн. наук;
С.К. Акулов, зам. нач. отдела программного обеспечения отрасли и технического обслуживания департамента социальной защиты населения Кемеровской области

Рекомендовано редакционно-издательским советом Кемеровского технологического института пищевой промышленности

Баканов М.В., Романова В.В., Крюкова Т.П.
Базы данных. Системы управления базами данных: учебное пособие / М.В. Баканов, В.В. Романова, Т.П. Крюкова; Кемеровский технологический институт пищевой промышленности. - Кемерово, 2010. - 166 с.
ISBN

В учебном пособии изложены принципы построения реляционных баз данных. Приведен пример построения реляционной модели в СУБД Microsoft Access. Рассмотрены основные команды и операторы языка программирования, генераторы приложений, создание и использование справочников в среде СУБД FoxPro.

Предназначено для изучения дисциплин - «Базы данных», «Информационные технологии в сервисе. Оргтехника» и «Информатика», студентами всех форм обучения и подготовки по специальностям: 100101 – Сервис, 140504 - Холодильная, криогенная техника и кондиционирование, 220301 - Автоматизация технологических процессов и производств, 280102 - Безопасность технологических процессов и производств, 280104 - Пожарная безопасность.

УДК
ББК

ISBN

ВВЕДЕНИЕ

В настоящее время трудно найти сферу человеческой деятельности, в которой не использовалось понятие баз данных. Возросший поток информации приводит к необходимости разработки новых приемов ее осмысления и обработки. Накопленный опыт и развитие программного обеспечения позволяет переосмыслить такую традиционную область обработки информации как хранение и управление данными. Новый подход к организации процесса обработки информации приводит к понятию *база данных* и методам работы с ней.

Эффективность работы базы данных в большой степени зависит от грамотного проекта базы данных, построить который помогут основы теории реляционных баз данных, рассматриваемые в трех главах настоящего учебного пособия.

В первой главе пособия приведены основные понятия баз данных и систем управления базами данных, дается классификация моделей данных.

Во второй главе показаны основные этапы проектирования баз данных, дается описание процесса реализации реляционной модели в программе Microsoft Access. На конкретном примере рассматривается табличный язык запросов, а также стандартный язык запросов SQL.

В третьей главе на множестве примеров рассмотрены команды и операторы языка программирования среды FoxPro.

Учебное пособие может быть использовано для самостоятельного изучения баз данных и системам управления базами данных (Microsoft Access и среды FoxPro), а так же в процессе обучения студентов по направлениям подготовки: 100101 – Сервис, 140504 - Холодильная, криогенная техника и кондиционирование, 220301 - Автоматизация технологических процессов и производств, 280102 - Безопасность технологических процессов и производств, 280104 - Пожарная безопасность.

ГЛАВА 1. БАЗЫ ДАННЫХ

Использование баз данных становится неотъемлемой составляющей деловой деятельности современного человека. В связи с этим большую актуальность приобретает освоение принципов построения и эффективного применения соответствующих технологий и программных продуктов - систем управления базами данных.

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.

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

СУБД - средство, которое выполняет основные функции:

– описывает структуру информации;
– определяет правила доступа к информации;
– вводит, хранит и проверяет информацию;
– производит поиск информации по запросу,
– выполняет различные виды сортировки;
– формирует выходные формы.

В процессе исследований, посвященных тому, как именно должна быть устроена СУБД, американским комитетом по стандартизации ANSI (American National Standards Institute) сформулирована трехуровневая система организации БД (рис. 1.1).

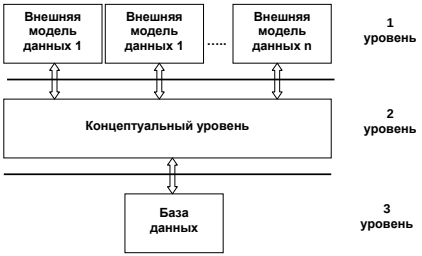


Рис. 1.1. Трехуровневая модель системы управления БД

1. **Уровень внешних моделей** – самый верхний уровень, где каждая модель имеет свое «видение» данных. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению.

2. **Концептуальный уровень** – центральное управляющее звено, здесь БД представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной БД. Фактически концептуальный уровень отражает обобщенную модель предметной области, для которой создавалась БД.

3. **Физический уровень** – собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации. Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными. Первая предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной БД.

1.1 Классификация моделей данных

Данные не обладают определенной структурой, данные становятся информацией тогда, когда пользователь задает им определенную структуру, то есть осознает их смысловое содержание. Поэтому центральным понятием в концепции БД является понятие модели. **Модель данных** - это некоторая абстракция, которая, будучи применима к конкретным данным, позволяет трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними. На рис. 1.2 представлена классификация моделей данных.

В соответствии с трехуровневой архитектурой мы сталкиваемся с понятием модели данных по отношению к каждому уровню. Физическая модель данных оперирует категориями,

касающимися организации внешней памяти и структур хранения, используемых в данной операционной среде. В настоящий момент в качестве физических моделей используются различные методы размещения данных, основанные на файловых структурах: это организация файлов прямого и последовательного доступа, индексных файлов и инвертированных файлов, файлов, использующих различные методы хеширования, взаимосвязанных файлов. Кроме того, современные СУБД широко используют страничную организацию данных.

Рис. 1.2. Классификация моделей данных

Наибольший интерес вызывают модели данных, используемые на концептуальном уровне. По отношению к ним внешние модели называются подсхемами и используют те же абстрактные категории, что и концептуальные модели данных.

Модели концептуального уровня должны выражать информацию о предметной области в виде, который не зависит от используемой СУБД. Эти модели называются **инфологическими**, или **семантическими**, и отражают в естественной и удобной для разработчиков и других пользователей форме с фиксацией и описанием объектов предметной области, их свойств и взаимосвязей.

Инфологические модели данных используются на ранних стадиях проектирования для описания структур данных в процессе разработки приложения, а **даталогические** модели уже поддерживаются конкретной СУБД. **Фактографические** модели данных соответствуют представлению информации в виде определенных структур данных (п. 1.3). **Документальные** модели данных соответствуют представлению о слабоструктурированной информации, ориентированной в основном на свободные форматы документов, текстов на естественном языке.

Модели, основанные на языках разметки документов, связаны, прежде всего, со стандартным общим языком разметки - SGML (Standart Generalised Markup Language). Этот язык предназначен для создания других языков разметки, он определяет допустимый набор тегов (ссылок), их атрибуты и внутреннюю структуру документа. Но ввиду некоторой своей сложности SGML использовался в основном для описания синтаксиса других языков (наиболее известным из которых является HTML), и немногие приложения работали с SGML-документами напрямую.

1.2 Жизненный цикл БД

Под **жизненным циклом** БД понимаются этапы развития БД, начиная от анализа предметной области, и заканчивая ее эксплуатацией. Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить четыре этапа проектирования:

1**. Системный анализ** и словесное описание информационных объектов предметной области. Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описа-

нием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД.

2. **Проектирование инфологической модели** предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели где показан состав и взаимосвязи хранимых данных и которая может дополняться новыми данными.

3. **Даталогическое** (логическое) **проектирование** БД - преобразование требований к данным в структуры данных. На выходе получаем СУБД - ориентированную структуру БД и спецификации прикладных программ. Для реляционной БД на этом этапе производиться описание структуры каждой таблицы и их взаимосвязей.

4. **Физическое проектирование** БД - определение особенностей хранения данных (определение способов и мест размещения), методов доступа, оценку ее объема и других параметров.

Однако на практике между вторым и третьим этапами возникает необходимость выбора конкретной СУБД для реализации проекта, поэтому в процессе проектирования БД следует выделить еще одним этап – **выбор СУБД**. Тогда на этапе даталогического проектирования могут моделировать БД применительно к различным СУБД и проводит сравнительный анализ моделей.

1.3 Классификация СУБД

Важнейшим достоинством применения БД является обеспечение независимости данных от прикладных программ. Это даст возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещения данных в памяти, методов доступа к ним и т.д. Такая независимость достигается поддерживаемым СУБД многоуровневым представлением данных в БД на логическом (пользовательском) и физическом уровнях. Благодаря СУБД и наличию логического уров-

ня представления данных обеспечивается отделение концептуальной (понятийной) модели БД от ее физического представления в памяти ЭВМ.

В качестве основных классификационных признаков можно использовать следующие: модель данных, вид программы, характер использования, средства работы с данными. Названные признаки существенно влияют на целевой выбор СУБД и эффективность использования разрабатываемой информационной системы.

Классификация СУБД по структуре данных.

Логическую структуру хранимых в базе данных называют моделью представления данных. К основным моделям представления данных относятся следующие: реляционная, иерархическая, сетевая, объектно-ориентированная. Некоторые СУБД могут одновременно поддерживать несколько моделей данных.

Реляционная модель является простейшей и наиболее привычной формой представления данных в виде таблицы. В теории множеств для нее имеется развитый математический аппарат - реляционное исчисление и реляционная алгебра. Достоинством реляционной модели является сравнительная простота инструментальных средств ее поддержки, недостатком - жесткость структуры данных (невозможность, например, задания строк таблицы произвольной длины) и зависимость скорости ее работы от размера базы данных. Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение (relation).

Отношение представляет собой множество элементов, называемых кортежами. Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица. Таблица имеет строки (записи) и столбцы (поля). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам - атрибуты отношения.

С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно деление одного объекта (явления, сущности, системы и проч.), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый

из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют связывание таблиц. Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы эффективности обработки данных этого типа также оказались технически вполне разрешимыми.

Основными недостатками реляционной модели является отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля - поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

Для описания структуры иерархической модели используется тип данных - «дерево». Тип «дерево» является составным и включает в себя подтипы («поддеревья»), каждый из которых, в свою очередь, является типом «дерево». Каждый из типов «де-

рево» состоит из одного «корневого» типа и упорядоченного набора подчиненных типов (рис. 1.3). Корневым называется тип, который имеет подчиненные типы и сам не является подтипом. Подчиненный тип (подтип) является потомком по отношению к типу, который выступает для него в роли предка (родителя). Потомки одного и того же типа являются близнецами по отношению друг к другу. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание БД.

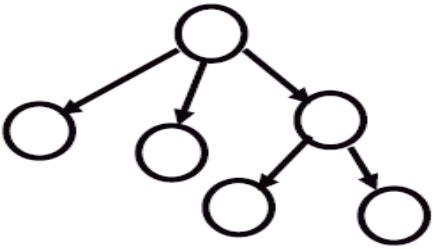


Рис. 1.3. Иерархическая модель

Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо. Иерархическая модель предполагает наличие связей между данными, имеющими какой-либо общий признак. В иерархической модели такие связи могут быть отражены в виде дерева-графа, где возможны только односторонние связи от старших вершин к младшим. Это облегчает доступ к необходимой информации, но только если все возможные запросы отражены в структуре дерева. Никакие иные запросы удовлетворены быть не могут. К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией. Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

Сетевая модель (рис. 1.4) данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель. Для описания схемы сетевой БД используется две группы типов: «запись» и «связь». Тип «связь» определяется для двух типов «запись»: предка и потомка. Переменные типа «связь» являются экземплярами связей.

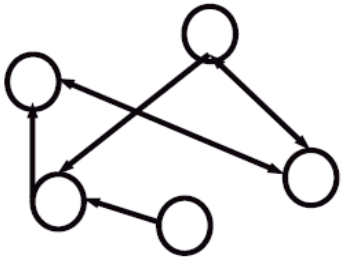


Рис. 1.4. Сетевая модель

Сетевая БД состоит из набора записей и набора соответствующих связей. На формирование связи особых ограничений не накладывается. Если в иерархических структурах запись-потомок могла иметь только одну запись-предка, то в сетевой модели данных запись-потомок может иметь произвольное число записей-предков (сводных родителей).

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей. Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные за-

писи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, механизмы инкапсуляции, наследования и полиморфизма. Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено. Смысл такого свойства будет определяться тем объектом, в который оно инкапсулировано. Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта. Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками (например, между двумя потомками одного родителя), то в их общем предке определяется абстрактное свойство. Полиморфизм означает способность одного и того же программного кода работать с разнотипными данными. Другими словами, он означает допустимость в объектах разных типов иметь методы с одинаковыми именами.

Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента. Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки. Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

Классификация СУБД по видам программ.

Полнофункциональные СУБД имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД: создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчеты, выводить их на печать. К ним относятся, например, такие пакеты, как Clarion Database Developer, DataEase, Dataplex, dBase IV, Microsoft Access, Microsoft FoxPro и Paradox. Для создания за-

просов и отчетов удобно пользоваться языком QBE (Query By Example - формулировки запросов но образцу). Многие СУБД включают средства программирования для профессиональных разработчиков. Часто требуется обеспечения доступа к данным, хранящимся в других форматах и в различных местах. Для обеспечения доступа к другим БД имеются внешние модули.

Для организации центров обработки данных в сетях ЭВМ предназначены серверы БД. Сервером в компьютерной сети называется компьютер (программа), управляющий ресурсом, клиентом - компьютер (программа), использующий этот ресурс. В качестве ресурса компьютерной сети могут выступать, к примеру, базы данных, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является база данных, то соответствующий сервер называется сервером базы данных. Серверы баз данных реализуют функции управления базами данных, запрашиваемые другими программами обычно с помощью операторов SQL. Примерами серверов БД являются следующие программы: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress), Oracle Server.

В роли клиентских программ для серверов БД в общем случае могут использоваться различные программы: СУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т. д. При этом элементы пары «клиент - сервер» могут принадлежать одному или разным производителям программного обеспечения.

К средствам разработки пользовательских приложений относятся система программирования Clipper, инструментальные системы: Delphi и Power Builder (Borland), Visual Basic (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Designor (SDP и Powersoft) и ERwin (LogicWorks), мониторы транзакций.

Транзакция – взаимодействие с пользователем по принципам независимость и неделимости взаимодействия на отдельные операции. Если по каким-либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении, включая приложение) транзакция остается незавершенной, то она отменяется.

Транзакции присущи три основных свойства:

– атомарность (выполняются все входящие в транзакцию операции или ни одна);

– сериализуемость (отсутствует взаимное влияние выполняемых в одно и то же время транзакций);

– долговечность (даже крах системы не приводит к утрате результатов зафиксированной транзакции).

Контроль транзакций важен в однопользовательских и в многопользовательских СУБД, где транзакции могут быть запущены параллельно. В последнем случае говорят о сериализуемости транзакций. Под сериализацией параллельно выполняемых транзакций понимается составление такого плана их выполнения (сериального плана), при котором суммарный эффект реализации транзакций эквивалентен эффекту их последовательного выполнения.

При параллельном выполнении смеси транзакций возможно возникновение конфликтов (блокировок), разрешение которых является функцией СУБД. При обнаружении таких случаев обычно производится «откат» путем отмены изменений, произведенных одной или несколькими транзакциями.

Классификация СУБД по характеру использования.

Персональные СУБД обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения зачастую могут выступать в роли клиентской части многопользовательской СУБД. К персональным СУБД, например, относятся FoxPro 2.6, Visual FoxPro, Paradox, Clipper, dBase, Access и др.

Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle и Informix.

Классификация СУБД по средствам работы с данными.

Для работы с хранящейся в базе данных информацией СУБД предоставляет программам и пользователям следующие два типа языков:

- язык описания данных - высокоуровневый непроцедурный язык декларативного типа, предназначенный для описания логической структуры данных;

- язык манипулирования данными - совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными; ввод, модификацию и выборку данных по запросам. Названные языки в различных СУБД могут иметь отличия. Наибольшее распространение получили два стандартизованных языка: QBE (Query By Example) - язык запросов по образцу и SQL (Structured Query Language) - структурированный язык запросов. QBE в основном обладает свойствами языка манипулирования данными, SQL сочетает в себе свойства языков обоих типов - описания и манипулирования данными.

ГЛАВА 2. СУБД MICROSOFT ACCESS

Microsoft Access – программа для создания и редактирования баз данных, которая входит в состав пакета офисных программ Microsoft Office.

Каждая конкретная СУБД имеет свои особенности, которые необходимо учитывать. Однако, имея представление о функциональных возможностях любой СУБД, можно представить обобщенную схему работы пользователя в этой среде.

В качестве основных этапов работы с СУБД можно выделить следующие:

- создание структуры таблиц базы данных;
- создание схемы данных;
- ввод и редактирование данных в таблицах;
- обработка данных, содержащихся в таблицах;
- вывод информации из базы данных.

Microsoft Access объединяет сведения из разных источников в одной реляционной базе данных. Создаваемые формы, запросы и отчеты позволяют быстро и эффективно обновлять данные, получать ответы на вопросы, осуществлять поиск нужных данных, анализировать данные, печатать отчеты, диаграммы и почтовые наклейки. Все составляющие базы данных (таблицы, отчеты, запросы, формы и объекты) хранятся в едином дисковом файле, имеющем расширение **mdb**.

Основным структурным компонентом базы данных является таблица, в которой хранятся вводимые данные. Таблица состоит из столбцов, называемых **полями**, и строк, называемых **записями**. Каждая запись таблицы содержит всю необходимую информацию об отдельном элементе базы данных.

При разработке структуры таблицы необходимо, прежде всего, определить названия полей, из которых она должна состоять, типы полей и их размеры. Каждому полю таблицы присваивается уникальное имя, которое не может содержать более 64 символов. Далее в режиме конструктора каждому полю присваивается один из типов данных (таблица 2.1).

Таблица 2.1

Типы данных в Microsoft Access

Тип данных	Использование	Размер	Примечание
Текстовый	Алфавитно-цифровые данные: текст или числа, не требующие проведения расчетов	до 255 байтов (символов)	Используется обычно для символьных данных
Поле MEMO	Алфавитно-цифровые данные: предложения, абзацы, тексты	до 64000 байтов	Используется только в тех случаях, когда размер текста > 255 и < 65535 символов
Числовой	Числовые данные	1, 2, 4 или 8 байтов	Точность значений данных зависит от значения свойства Размер поля
Денежный	Данные о денежных суммах	8 байтов	Используется для хранения данных о денежных суммах. Точность – до 15 знаков в целой и до 4 знаков в дробной части.
Дата/время	Даты и время	8 байтов	Используется для хранения календарных дат или значений времени
Счетчик	Уникальное целое, генерируемое Access при создании каждой новой записи	4 байта	Разновидность числового типа, значения в котором последовательно возрастающие (на 1) или случайные числа, автоматически вводящиеся при добавлении новой записи
Логический	Логические данные	1 бит	Используется для хранения значений Истина или Ложь
Объект OLE	Объект, связанный или внедренный в таблицу Microsoft Access	До 1 Гбайта	Картинки, диаграммы и другие объекты из приложений Windows

При работе с данными из нескольких таблиц устанавливаются связи между таблицами. Для поиска и отбора данных, удовлетворяющих определенным условиям, создаются запросы. Они позволяют обновить или удалить несколько записей, выполнить встроенные или специальные вычисления и многое другое. Для просмотра, ввода или изменения данных прямо в таблице применяются формы. Форма позволяет отобрать данные из одной или нескольких таблиц и вывести их на экран, используя стандартный или созданный пользователем макет. Для анализа данных или распечатки их определенным образом используется отчет. Например, можно создать и напечатать отчет, группирующий данные и вычисляющий итоги, или отчет для распечатки почтовых наклеек.

2.1 Интерфейс Microsoft Access

Запуск Microsoft Access производим через ярлык  на рабочем столе или меню Пуск. После загрузки на экране появится главное окно, в котором размещается окно базы данных. При первом запуске Access в главном окне выводится область задач в режиме **Приступая к работе**, с помощью которой можно открыть существующие БД и Создать файл. При выборе команды **Создать файл** в области задач изменится режим на **Создание файла**.

При выборе команды **Новая база данных** откроется окно диалога **Файл новой базы данных**, в котором необходимо выбрать имя диска и папки для хранения БД, а также задать имя БД (например, Институт) и щелкнуть на кнопке **Создать**, будет сохранен файл с расширением институт.mdb (рис. 2.1).

После выполнения этих шагов появится окно БД - **Институт: база данных** (рис. 2.2).

Главное окно приложения Microsoft Access состоит из следующих областей (сверху вниз):

- строка заголовка;
- строка меню;
- панель инструментов;
- окно базы данных;
- строка состояния.

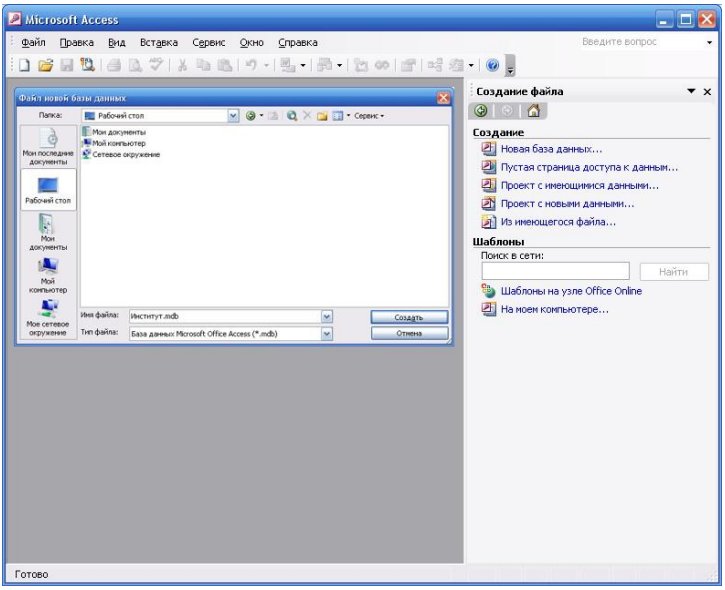


Рис. 2.1 Создание новой базы данных

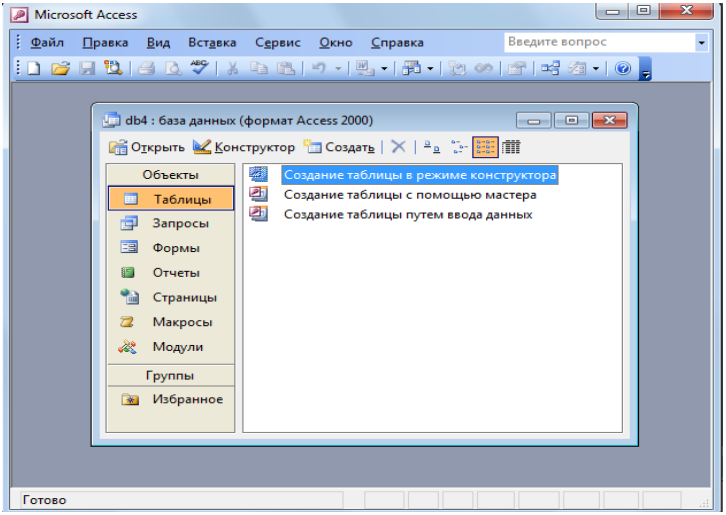


Рис. 2.2. Окно базы данных Институт

В **строке заголовка** находится системное меню в виде пиктограммы, расположенной слева от названия главного окна: «Microsoft Access».

Строка меню содержит группы команд объединенные по функциональному признаку: Файл, Правка, Вид, Вставка, Сервис, Окно, Справка. Команды, содержащиеся в меню аналогичны командам в приложениях семейства Office.

Панель инструментов. При запуске Access по умолчанию активизируется одна панель инструментов. На панели инструментов расположены наиболее часто используемые команды. Перед созданием БД необходимо ознакомиться с главным меню и панелью инструментов.

Окно **базы данных** имеет:

– строку заголовка;

– панель инструментов, на которой расположены следующие кнопки: Открыть; Конструктор; Создать; Удалить; Крупные значки; Мелкие значки; Список; Таблица;

– панель Объекты: таблицы, запросы, формы, отчеты, страницы, макросы и модули;

– область окна со списком возможных режимов создания новых объектов или просмотра и редактирования существующих объектов (в этой области также отображаются списки имеющихся в этой базе таблиц, форм, запросов и т.д.);

– Строка состояния находится внизу главного окна и предназначена для вывода краткой информации о текущем режиме работы.

Команды **панели инструментов** окна БД:

Открыть – открытие выделенного объекта (таблицы, запроса, формы и т.д.) в режиме страницы; **Конструктор** - открытие выделенного объекта в режиме конструктора; **Создать** – создание объекта базы данных; **Удалить** – Удаление выделенного объекта; **Крупные значки; Мелкие значки; Список; Таблица** – представление объектов базы данных в окне базы данных в соответствующем виде.

Панель **Объекты**:

Таблица – двумерные таблицы, которые используется для хранения данных в реляционных базах данных. Данные хранятся в

записях, которые состоят из отдельных полей. Каждая таблица содержит информацию определенного типа (например, студентах).

Запрос - средство для отбора данных, удовлетворяющих определенным условиям. С помощью запросов можно выбрать из БД только необходимую информацию

Форма – средство, которое позволяет упростить процесс ввода или изменения данных в таблицах БД, что обеспечивает ввод данных, просмотр, корректировка и удаления записей.

Отчет - средство, которое позволяет извлечь из базы нужную информацию и представить ее в виде, удобном для восприятия, а также подготовить для распечатки отчет, который оформлен соответствующим образом.

Страницы - страницы доступа к данным представляют собой специальную Web-страницу, предназначенную для просмотра и работы через Интернет или интрасеть с данными, которые хранятся в базах данных Microsoft Access или MS SQL Server.

Макрос - набор макрокоманд, создаваемый пользователем для автоматизации выполнения конкретных операций.

Модуль - объект, содержащий программы на языке Visual Basic, применяемые в некоторых случаях для обработки данных.

Область со списком возможных режимов создания объектов. В этой области кроме списка режимов создания объектов отображаются созданные объекты (например, таблицы, формы и т.д.), которые можно просматривать или редактировать.

2.2 Проектирование базы данных

Методически правильно начинать работу с карандашом и листом бумаги в руках, не используя компьютер. Неоптимальные решения и прямые ошибки, заложенные на этапе проектирования, впоследствии очень трудно устраняются, поэтому этот этап является основополагающим. Получив задание на проектирование базы, составляют:

– Список исходных данных, с которыми будем работать;

– Список выходных данных, которые необходимы для управления структурой БД;

– Список выходных данных.

2.2.1 Разработка структуры базы данных

Спроектировав основную часть данных, которые потребуются, можно приступать к созданию структуры БД, то есть структуры ее основных таблиц.

1. Работа начинается с составления генерального списка полей – он может насчитывать несколько позиций.

2. В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.

3. Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель – обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше – на одном рабочем месте.

4. В каждой из таблиц намечают **ключевое поле** – поле с уникальным значением однозначно определяющем запись. Например, для таблицы данных о студентах, таким полем может служить индивидуальный код студента. Если в таблице нет полей, которые можно было бы использовать, как ключевые, всегда можно ввести дополнительное поле типа **Счетчик** – оно не может содержать повторяющихся данных по определению.

5. С помощью карандаша и бумаги расчерчивают связи между таблицами. Такой чертеж называется **схемой данных**. Существует несколько типов возможных связей между таблицами. Связь между таблицами организуется на основе общего поля, причем в одной из таблиц оно обязательно должно быть ключевым, то есть на стороне «один» должно выступать ключевое поле, содержащее уникальные, неповторяющиеся значения. Значения на стороне «многие» могут повторяться.

6. Разработкой схемы данных заканчивается «бумажный» этап работы. Если схема данных составлена правильно, подключать к базе новые таблицы нетрудно. Если структура базы нерациональна, разработчик может испытать серьезные трудности.

На этом этапе завершается предварительное проектирование базы данных, и на следующем этапе начинается ее непосредственная разработка. С этого момента следует начать работу с СУБД.

2.2.2 Постановка задачи

В качестве примера разработки баз данных в среде СУБД Access выберем систему под условным названием **Деканат**, позволяющую:

– хранить основные сведения о студентах учебных групп, изучаемых ими дисциплинах и оценках, сведенья о предоставлении общежития;

– производить ввод новых данных в систему и редактирования существующих;

– находить сведения о конкретных студентах, преподавателях, дисциплинах, учебных группах;

– составлять различного рода формы, таблицы, отчеты.

На этапе проектирования принято решение о вхождении в БД «Деканат» шести таблиц: **Студенты, Преподаватели, Дисциплины, Оценки, Общежитие и Стипендия** (рис. 2.3).

Таблица Студенты:

Код студента
Фамилия
Имя
Отчество
Номер группы
Адрес
Телефон
Дата рождения
Пол
Фото

Таблица Стипендия:

Код студента
Месяц

Таблица Оценки:

Код студента
Код дисциплины
Номер семестра
Оценка

Таблица Преподаватели:

Код преподавателя
Фамилия
Имя
Отчество
Дата рождения
Должность
Стаж
Телефон

Таблица Общежитие:

Код студента
Общежитие
Оплата

Таблица Дисциплины:

Код дисциплины
Название дисциплины
Код преподавателя
Номер семестра
Экзамен

Рис. 2.3. Структура таблиц БД «Деканат»

2.3 Создание базы данных

Можно воспользоваться мастером баз данных для создания всех необходимых таблиц, форм и отчетов для базы данных выбранного типа - это простейший способ начального создания базы данных. Однако мастер предлагает ограниченный набор параметров для настройки выбранной базы данных (например, с помощью мастера нельзя добавлять новые таблицы, формы и отчеты в существующую БД). Для вызова мастера нажмите кнопку **Создать** на панели инструментов.

– В **области задач** создание файла в группе **Шаблоны** выберите **На моем компьютере**.

– Выберите значок подходящего шаблона базы данных на вкладке **Базы данных** и нажмите кнопку **Ok** (рис 2.4).

– В диалоговом окне **Файл новой базы данных** введите имя базы данных и укажите ее расположение, а затем нажмите кнопку **Создать**.

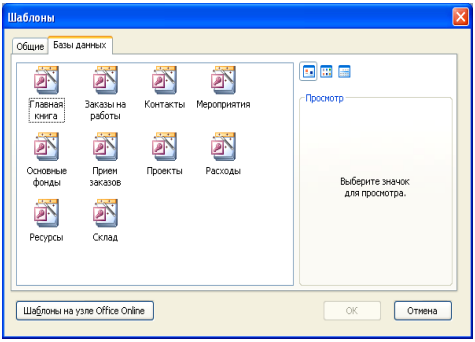


Рис. 2.4 Диалоговое окно Шаблоны

2.4 Создание таблиц

Создание базы данных начинается с создания таблиц, в которых и хранится информация о предметной области. База данных обычно включает несколько взаимосвязанных таблиц. Для создания новой таблицы в окне БД надо выбрать объект

Таблица и нажать кнопку **Создать**, в результате чего появится окно **Новая таблица.**

В Access существует четыре способа создания пустой таблицы:

– Использование мастера БД для создания всей базы данных, содержащей все требуемые отчеты, таблицы и формы, за одну операцию.

– Использование мастера таблиц для выбора полей таблицы из множества определенных ранее таблиц.

– Ввод данных непосредственно в пустую таблицу в режиме таблицы. При сохранении таблицы данные анализируются и каждому полю присваивается необходимый тип данных и формат.

– Определение всех параметров макета таблицы в режиме конструктора.

Для выбора необходимого режима создания таблиц можно дважды щелкнуть на один из них в списке режимов, откроется требуемый режим. Кроме того, можно щелкнуть на пиктограмме **Создать** в окне БД, откроется окно диалога **Новая таблица**, и в нем выбрать требуемый режим создания таблицы (рис. 2.5).

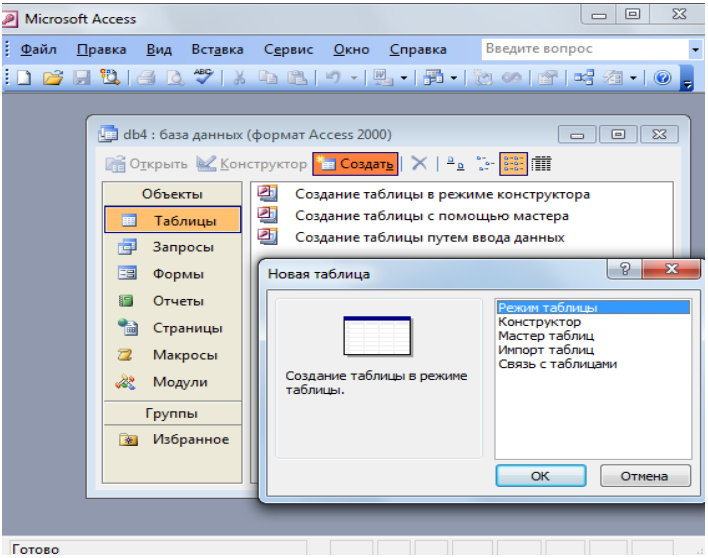


Рис. 2.5 Диалоговое окно Новая таблица

При выборе режима **Мастер таблиц** откроется окно Создание таблиц, в котором с помощью образцов таблиц и полей легко сформировать поля новой таблицы (рис 2.6).

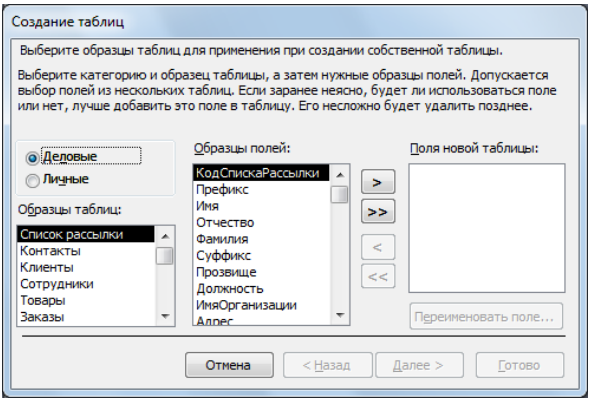


Рис. 2.6 Диалоговое окно Мастера таблиц

Если вы затрудняетесь сразу определить структуру таблицы, Microsoft Access позволяет создать таблицу путем ввода данных в окно с традиционной табличной формой (рис. 2.7).

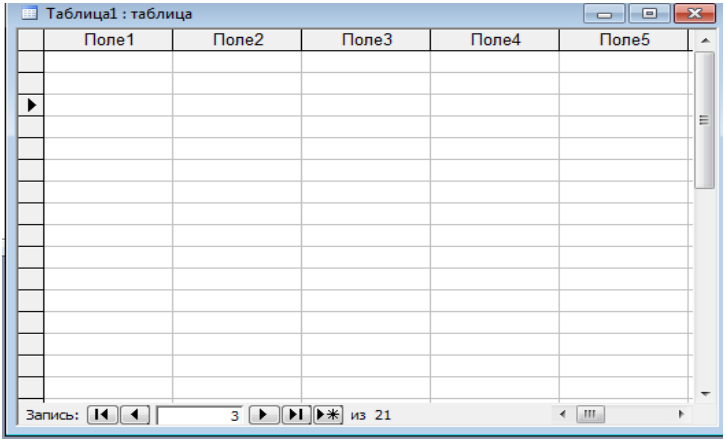


Рис. 2.7 Создание таблицы путем ввода данных

Если в окне **Создание таблиц** нет требуемого образца таблицы, то необходимо выбрать **режим Конструктора**. Создание таблицы в окне конструктора предоставляет более широкие возможности по определению параметров создаваемой таблицы. **Конструктор** позволяет задавать значения всех свойств параметров, характеризующих поле.

Активизировать **Конструктор** для создания новой таблицы можно либо непосредственно из окна диалога **Новая таблица** (рис. 2.5), выбрав из списка вариантов значение **Конструктор** и нажав кнопку **Ок,** либо дважды щелкнув кнопкой мыши на ярлыке **Создание таблицы с помощью Конструктора**. В результате выполнения этих действий откроется окно **Конструктора** таблиц, в котором можно выделить три основные области: панель инструментов конструктора таблиц, область ввода имен и типов полей, область для задания свойств полей.

Область ввода имен и типов полей представляет собой таблицу, которая содержит следующие атрибуты создаваемой таблицы: **наименование поля, тип данных и описание** (рис.2.8)**.**

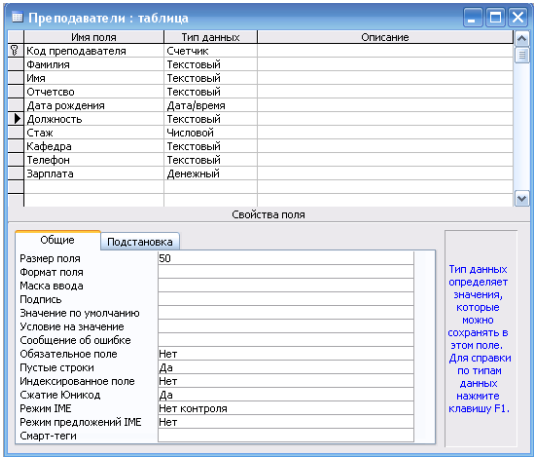


Рис. 2.8. Таблица Преподаватели в режиме Конструктора

Панель инструментов конструктора таблиц (рис.2.9) содержит кнопки для активации команд, используемых при создании структуры таблицы: **Вид** - осуществляет переход из режима конструктора в режим таблицы; **Ключевое поле** - позволяет устанавливать или отключать ключевое поле; **Свойства** - открывает окно свойств выделенного объекта; **Построить** - позволяет создавать выражения и маски ввода данных; **Окно БД** - отображает главное окно БД; **Новый объект** - отображает раскрывающийся список новых объектов, которые можно создать, например таблица, отчет, форма и другие кнопки.

Рис. 2.9 Панель инструментов конструктора таблиц

В окне **Конструктора таблиц** в столбце **Имя поля** вводятся имена полей создаваемой таблицы. В столбце **Тип данных** вводится или выбирается из раскрывающегося списка тип данных, которые будут содержаться в поле, для каждого поля таблицы. В столбце **Описание** можно ввести описание данного поля, хотя это необязательно, однако при работе приложения эта информация выводится в строке состояния окна приложения и может служить справкой пользователю. В нижней части **Конструктора таблиц** на вкладках **Общие и Подстановка** можно ввести свойства каждого поля или оставить значения свойств по умолчанию.

После описания всех полей будущей таблицы необходимо нажать на кнопку **Закрыть.** На вопрос **Сохранить изменения макета или структуры таблицы** надо нажать на кнопку **Да**.

В окне **Сохранить как** в поле **Имя таблицы** ввести имя создаваемой таблицы и нажать кнопку **Ок.** В ответ на сообщение **Ключевые поля не заданы** и вопрос **Создать Ключевое поле сейчас?** нажмем **Да**, если ключевое поле необходимо, и **Нет** в

противном случае. После указанных действий в списке таблиц в окне базы данных появится имя и значок новой таблицы.

Независимо от метода, применяемого для создания таблицы, всегда имеется возможность использовать режим конструктора для дальнейшего изменения макета таблицы, например для добавления новых полей, установки значений по умолчанию или для создания масок ввода. Однако только метод в **режиме конструктора** позволяет сразу задать требуемую структуру таблицы, поэтому далее будем рассматривать именно этот метод.

Ввод данных в созданную таблицу можно, осуществить открыв таблицу в **Режиме таблицы**. При вводе данных следует помнить, что при нажатии клавиши Del ячейка очищается; если ввод данных в ячейку прервать, нажав клавишу Esc, то восстановится старое значение, а если нажать клавиши Enter или Tab, то в ячейку заносится новое значение. Для редактирования текущего значения необходимо дважды щелкнуть мышью или нажать клавишу F2.

Для редактирования данных курсор переводится в нужную ячейку, старые данные удаляются и вводятся новые данные. Если таблица большая, то для поиска можно использовать команду **Правка**, **Найти**. Для замены большого количества одинаковых данных используется команда **Правка**, **Заменить**.

При работе с таблицей можно выполнять переключение между двумя режимами – **режимом конструктора** и **режимом таблицы.** Для переключения используется кнопка на панели инструментов или соответствующие опции в разделе меню **Вид** (рис.2.10). В режиме конструктора выполняется просмотр и редактирование структуры таблицы (названий и типов ее полей), а в режиме таблицы – просмотр и редактирование данных, хранящихся в этой таблице.

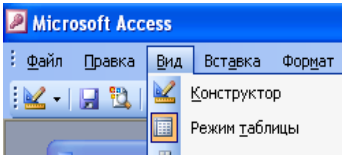


Рис. 2.10. Режимы работы с таблицей

2.4.1 Импорт объектов в Microsoft Access

Для импорта объекта из другой базы данных созданной в Access, выберите команду **Файл**, **Внешние данные**, **Импорт**. В окне диалога Импорт выберите нужный файл базы данных. После щелчка на кнопке Импорт Access откроет окно диалога Импорт объектов (рис. 2.10). В этом окне выделите имя или имена импортируемых объектов.

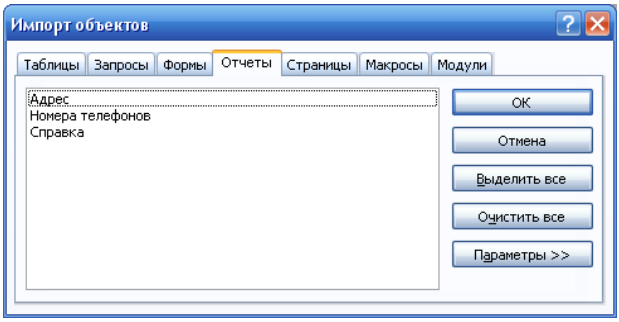


Рис. 2.10. Диалоговое окно **Импорт объектов**

Access позволяет также импортировать данные из файлов электронных таблиц, созданных в Microsoft Excel. Можно импортировать всю электронную таблицу или только ее часть, как в новую, так и в существующую базу данных Access. Если первая строка электронной таблицы содержит заголовки столбцов, можно использовать их в качестве имен полей новой таблицы. Access определяет типы данных для полей новой таблицы, анализируя значения в первых импортируемых строках.

Для импорта таблицы Excel в базу данных Access выберите команду **Файл**, **Внешние данные**, **Импорт**. В раскрывающемся списке **Тип файлов** выберите **Excel**, затем найдите нужный файл, содержащий импортируемую электронную таблицу.

После щелчка на кнопке **Импорт** Access откроет окно мастера импорта электронных таблиц (рис. 2.12), следуйте его указаниям. В одном из окон появится предложение определить первичный ключ таблицы Access. Можно создать поле таблицы с

типом данных Счетчик. Или, если Вы планируете включить в первичный ключ несколько полей, установите флажок «**Не создавать ключ**». Позднее можно открыть таблицу в режиме конструктора и определите нужный первичный ключ.

Отредактируйте созданную таблицу в режиме конструктора. В качестве ключевого задайте поле **Код Студента**, для чего выделите поле и выполните команду меню **Правка, Ключевое Поле,** или щелкните на соответствующей пиктограмме панели инструментов (рис 2.9).

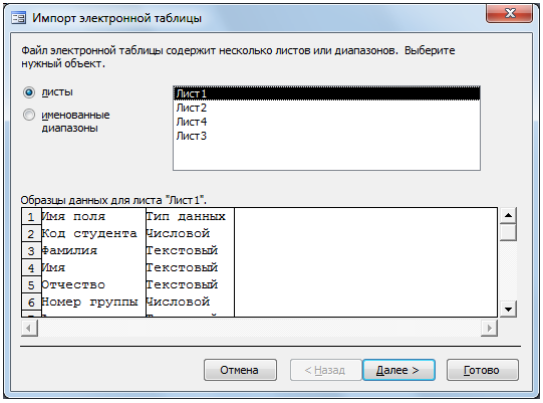


Рис. 2.12 Диалоговое окно Мастера импорта электронной таблицы

2.4.2 Маска ввода

Маску ввода обычно используют при вводе дат, телефонов и других значений, имеющих определенный формат. Оно состоит из двух частей разделенных символом точка с запятой.

При определении поля **Дата Рождения** используем маску для удобного ввода даты (т.е. в датах точки будут вводится автоматически). Для этого в **Свойства полей** на вкладке **Общие** установите курсор на поле маска, справа появится кнопка **с тремя точками** – нажмите на нее. В появившемся окне создания масок выбирайте **Краткий Формат Даты** (рис. 2.12).

В первом окне **Мастера** можно выбрать нужную маску из списка, которую он может создать. Если щелкнуть мышью по полю **Проба** в нижней части окна и ввести пробные данные, то можно увидеть, как будут формироваться вводимые символы.

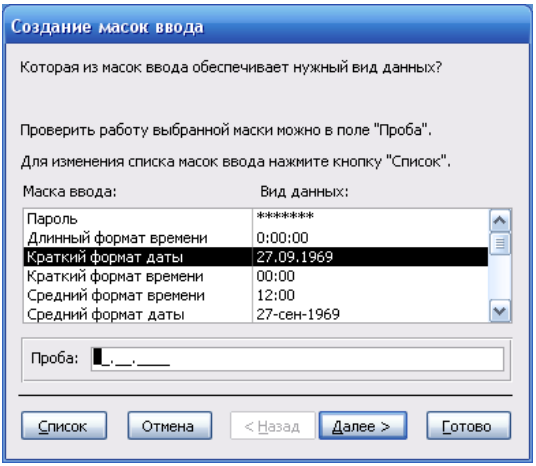


Рис. 2.12. Диалоговое окно Мастера масок ввода

Во втором диалоговом окне, после нажатия на кнопку **Далее**, увидим название маски, предлагаемую для нее символьную строку, поле со списком **Заполнитель,** в котором можно выбрать символ-указатель заполняемых при вводе позиций, и поле проверки работы маски работы маски. Здесь же в поле **Маска ввода** можно с помощью приведенных в таблице символов маски выполнить корректировку маски ввода.

В последнем диалоговом окне выбрать способ сохранения данных: вместе со знаками маски или без знаков маски, и нажать кнопку **Готово**. Если в первом диалоговом окне нажать кнопку **Список**, то появится диалоговое окно **Настройка масок ввода**. В нем надо ввести имя **Маски ввода**. Затем в поле **Маска ввода** сформировать с помощью символов, приведенных в таблице, саму маску ввода и выбрать символ-заполнитель при отображении маски.

В поле **Проба** можно выполнить проверку работы созданной **Маски ввода**. После нажатия кнопки **Закрыть** созданная маска ввода появится под заданным именем в списке масок первого диалогового окна **Мастера создания масок ввода**.

Для поля **Телефон** (рис. 2.8) можно использовать маску ввода 99-99-99, которая позволит не набирать тире в номере телефона при вводе значений в поле.

2.4.3 Мастер подстановок

При его использовании можно создать поле, содержание которого формируется путем выбора значений из списка, содержащего набор постоянных значений или значений из другой таблицы/запроса. Если источником для подстановки выбран столбец другой таблицы, то тип и длина поля, созданного таким способом, будет определяться типом и длиной элементов, служащих источником для подстановки значений.

Обратим внимание на поле **Должность** (рис 2.8). Для выбранной категории сотрудников имеется четыре возможные должности: ассистент, старший преподаватель, доцент и профессор. Заменим ввод этих значений выбором их из списка. В этом поле с помощью **Мастера подстановок** создается поле со списком, содержащее набор или постоянных значений, значений из другой таблицы или запроса.

Выделим поле **Должность** и выберем из списка в столбце **Тип данных** значение **Мастер подстановок** вместо текстового типа данных (рис. 2.8). В появившемся диалоговом окне **Создание подстановки** выбираем **Фиксированный Набор значений**. На втором шаге Мастера подстановок создадим столбец подстановки с набором значений:

- *Ассистент*
- *Старший преподаватель*
- *Доцент*
- *Профессор*

Закончив создание списка в режиме конструктора, на вкладке **Подстановка**, посмотрите появившиеся изменения после работы мастера (рис. 2.14).

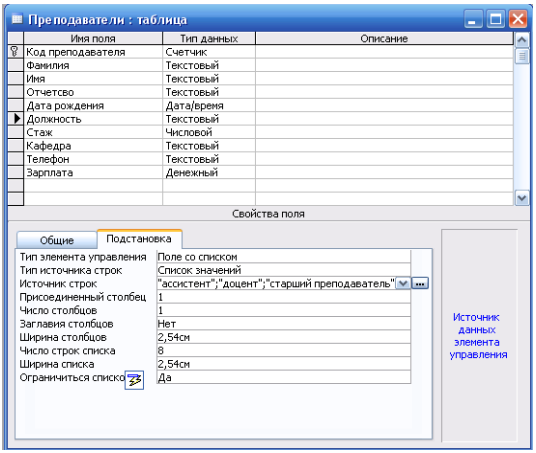


Рис. 2.14. Результат работы Мастера подстановок

При вводе данных в таблицу значения полей подстановки можно не вводить с клавиатуры, а выбирать из заданного списка. Чтобы нельзя было ввести значения, отсутствующие в списке, надо в свойствах поля на вкладке **Подстановка** в позиции **Ограничиться списком** задать значение **Да** (рис. 2.14). В этом случае использование поля подстановки обеспечит не только более эффективный ввод данных, но и более жесткий контроль целостности базы данных.

Применять операцию подстановки можно только к полям, содержащим текстовые и числовые данные, а также к логическим полям. Другие типы полей не могут использовать подстановку.

2.4.4 Вставка графических объектов. Поле объекта OLE

В поле объекта **OLE** Access дает возможность хранить и редактировать документы Word, электронные таблицы Excel, слайды презентаций PowerPoint, звуковые файлы (.wav), видеофайлы (.avi) или рисунки, созданные в приложениях Paint или Draw. Основными операциями над значениями этого поля

являются следующие: просмотр, модификация, создание и удаление.

В импортированную таблицу **Студенты** в режиме конструктора добавим, например, новое поле **Фото** с типом данных **- Поле объекта OLE**. Открыв таблицы **Студенты** и выделив соответствующую запись добавленного поля **Фото** выполним команду меню **Вставка, Объект**.

В появившемся диалоговом окне предлагается выбрать тип объекта, а также способ его определения: создать объект с помощью соответствующей программы или вставить его готовым из файла. Независимо от способа определения объекта, существуют два варианта включения объекта в поле записи базы (задается с помощью флажка «**Связь**»):

- путем внедрения исходного объекта в базу данных;
- путем связывания (флажок «**Связь**» включают), когда устанавливается связь между отдельно хранящимся файлом объекта и записью базы данных.

Включаемые объекты могут отображаться при просмотре, либо отображаться **в виде значков** или **двойным щелчком мыши** разворачиваться полностью.

Для модификации изображений, хранящихся в поле OLE объекта необходимо выбрать нужное поле и дважды щелкнуть по нему кнопкой мыши. В результате будет вызван подходящий графический редактор для редактирования изображения.

2.4.5 Ключевое поле

Ключ (ключевое поле) – это столбец (может быть несколько столбцов), добавляемый к таблице и позволяющий установить связь с записями в другой таблице. Если для таблицы определены ключевые поля, то Microsoft Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

Существуют ключи двух типов:

- **Первичный ключ** – это одно или несколько полей (столбцов), комбинация значений которых однозначно определяет каждую запись в таблице. Первичный ключ не допускает значений Null и всегда должен иметь уникальный индекс. Пер-

вичный ключ используется для связывания таблицы с внешними ключами в других таблицах.

- **Внешний (вторичный) ключ** - это одно или несколько полей (столбцов) в таблице, содержащих ссылку на поле или поля первичного ключа в другой таблице. Внешний ключ определяет способ объединения таблиц.

Из двух логически связанных таблиц одну называют таблицей первичного ключа или главной таблицей, а другую таблицей вторичного (внешнего) ключа или подчиненной таблицей. СУБД позволяют сопоставить родственные записи из обеих таблиц и совместно вывести их в форме, отчете или запросе.

Выделяют три типа первичных ключей:

- **Ключевое поле счетчика (счетчик)**. Тип данных поля в базе данных, в котором для каждой добавляемой в таблицу записи в поле автоматически заносится уникальное числовое значение.

- **Простой ключ**. Если поле содержит уникальные значения, такие как коды или инвентарные номера, то это поле можно определить как первичный ключ. В качестве ключа можно определить любое поле, содержащее данные, если это поле не содержит повторяющиеся значения или значения Null.

- **Составной ключ.** В случаях, когда невозможно гарантировать уникальность значений каждого поля, существует возможность создать ключ, состоящий из нескольких полей. Чаще всего такая ситуация возникает для таблицы, используемой для связывания двух таблиц многие - ко - многим.

Если возникают затруднения с выбором подходящего типа первичного ключа, то в качестве ключа целесообразно выбрать поле счетчика.

В Microsoft Access можно выделить три типа ключевых полей: **счетчик**, **простой ключ** и **составной ключ**. Для задания первого, в режиме Конструктора таблиц, достаточно указать для выбранного поля тип данных – **Счетчик**. Если же до сохранения созданной таблицы ключевые поля не были определены, то при сохранении будет выдано сообщение о создании ключевого поля. При нажатии кнопки **Да** будет автоматически добавлено создано ключевое поле **Код** с типом данных **Счетчик**.

Для создания **простого ключа** достаточно иметь поле, которое содержит уникальные значения (например, коды или но-

мера). **Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое**. В таких случаях следует, либо добавить в таблицу поле счетчика и сделать его ключевым или определить составной ключ.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей (рис.2.15). Для определения составного ключа необходимо открыть таблицу в режиме Конструктора и выделив при помощи мышки, удерживая нажатой клавишу **Ctrl**, необходимые поля щелкнуть кнопку Ключевое поле на панели инструментов (рис 2.9).

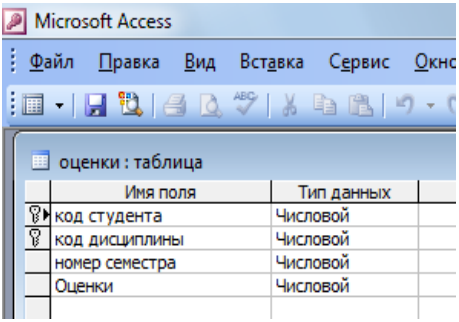


Рис. 2.15. Таблица Оценки в режиме конструктора

2.4.6 Связь между таблицами и целостность данных

Между одноименными полями (имена могут отличаться, но типы данных нет, исключением является счетчик) двух таблиц в Access можно устанавливать связь. Это означает, что при формировании запроса к этой паре таблиц Access сможет объединить строки таблиц, в которых значения поля совпадают. В общем случае допускается связь по двум, трем и более одноименным полям (кроме того, Access позволяет вручную установить связь между таблицами по разноименным полям).

При создании серьезных баз данных придется позаботиться о дополнительных средствах контроля связанных данных, вводимых в разные таблицы. Механизм, который обеспечивает

согласованность данных между двумя связанными таблицами, называется **поддержка целостности данных.**

После того, как таблицы созданы, можно задать их связанность. Для этого надо выбрать команду меню **Сервис, Схема данных** (либо нажать соответствующую кнопку на панели инструментов) и последовательно добавить те таблицы, между которыми будет определяться связь с помощью диалогового окна **Добавление таблицы** (рис. 2.16). Если окно **Добавление таблицы** неактивизировано, выполнить команду меню **Связи, Добавить таблицу** или нажать кнопку **Добавить таблицу** на панели инструментов.

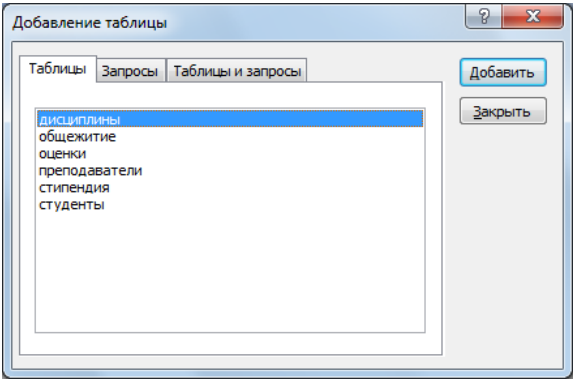


Рис. 2.16. Окно диалога Добавление таблицы

Для связывания таблиц (если ранее использовался **Мастер Подстановок** то связь между таблицами будет отображена сразу) следует выбрать поле для связи, в первой связываемой таблице, и переместить его с помощью мыши на соответствующее поле второй таблицы. На экране откроется диалоговое окно **Изменение связей** (рис. 2.17)

В большинстве случаев ключевое поле (представленное в списке полей полужирным шрифтом) одной таблицы связывают с соответствующим ему полем внешнего ключа (часто имеющим то же имя) во второй таблице. Связанные поля необязательно должны иметь одинаковые имена, однако, они должны иметь

одинаковые типы данных (из этого правила существуют два исключения). Кроме того, связываемые поля типа **Числовой** должны иметь одинаковые значения свойства **Размер поля**. Исключениями из этого правила являются поля **Счётчика** с последовательной нумерацией, которые могут связываться с числовыми полями размера **Длинное целое**, а также поля счётчика с размером **Код репликации**, связываемые с полями типа **Числовой**.

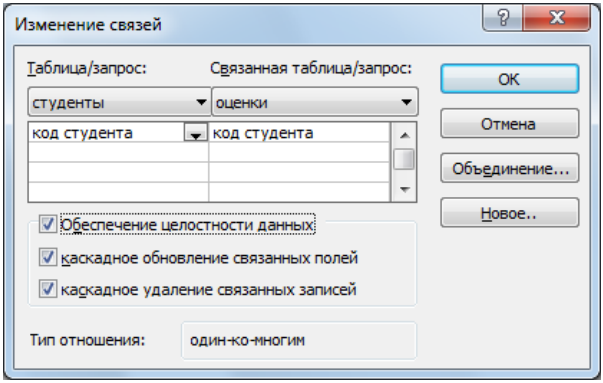


Рис. 2.17. Диалоговое окно Изменение связей

В диалоговом окне **Изменение связей** при необходимости можно установить флажок - **Обеспечение целостности данных**, которое означает, что:

– в связанное поле подчиненной таблицы можно вводить только те значения, которые имеются в связанном поле главной таблицы;

– из главной таблицы нельзя удалить запись, у которой значение связанного поля совпадает хотя бы с одним значением того же поля в подчиненной таблице.

При попытке нарушить эти запреты Access выдает сообщение об ошибке.

Каскадное обновление и удаление записей. Включив механизм поддержки целостности данных, вы можете (но не обязаны) потребовать, чтобы при модификации данных система запускала следующие процессы:

– каскадное обновление связанных полей;
– каскадное удаление связанных записей.

Каскадное обновление означает, что изменение значения связанного поля в главной таблице автоматически будет отражено в связанных записях подчиненной таблицы.

Каскадное удаление означает, что при удалении записи из главной таблицы, из подчиненной таблицы будут удалены все записи, у которых значение связанного поля совпадает с удаляемым значением.

Для завершения процесса создания связи щелкните на кнопке **ОК** и закройте окно **Схема данных**. Связь отображается в виде линии, соединяющей две таблицы. Любую связь можно выделить и удалить нажатием клавиши **Delete.** Кроме того, можно щелкнуть на линии правой кнопкой мыши, чтобы раскрыть контекстное меню, а затем выбрать команду **Изменить связь**, чтобы открыть диалоговое окно **Связи** (рис. 2.17).

Из **Схемы данных** (рис. 2.18) видно, что одному значению поля **Код студента** таблицы **Студенты** соответствует одно значение в таблицах **Стипендия** и **Общежитие** (связь Один-к-Одному) и несколько значений в таблице **Оценки** (связь Один-ко-Многим). Один студент может получать одну стипендию в месяц, проживая в общежитие в одной комнате, но сдать несколько зачетов и экзаменов.

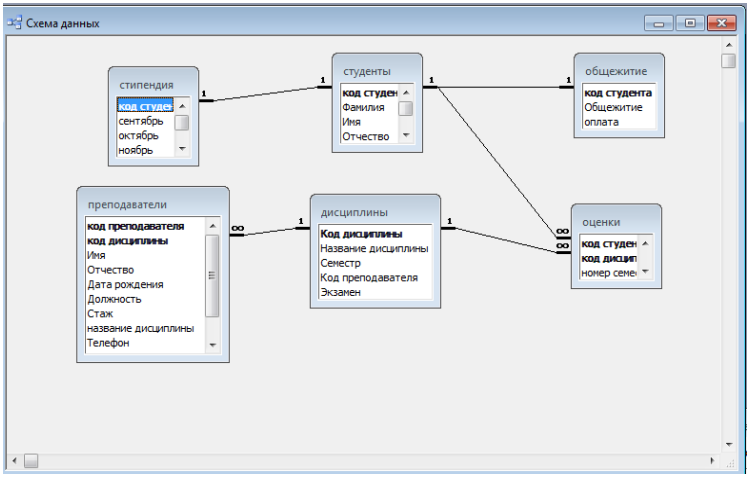


Рис. 2.18 Диалоговое окно Схема данных

Отношения, которые могут существовать между записями двух таблиц:

один – к - одному, каждой записи из одной таблицы соответствует одна запись в другой таблице;

один – ко - многим, каждой записи из одной таблицы соответствует несколько записей другой таблице;

многие – к - одному, множеству записей из одной таблице соответствует одна запись в другой таблице;

многие – ко - многим, множеству записей из одной таблицы соответствует несколько записей в другой таблице.

Тип отношения в создаваемой связи зависит от способа определения связываемых полей:

Отношение «один-ко-многим» создается в том случае, когда только одно из полей является полем первичного ключа или уникального индекса.

Отношение «один-к-одному» создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы.

Отношение «многие-ко-многим» фактически является двумя отношениями «один-ко-многим» с третьей таблицей, первичный ключ которой состоит из полей внешнего ключа двух других таблиц.

2.4.7 Ввод записей

Ввод записей выполняется в режиме работы с таблицами. Переход к табличному представлению БД осуществляется с помощью кнопки **Режим Таблицы** панели инструментов или с помощью меню **Вид - Режим таблицы.** Для копирования данных из аналогичного поля предыдущей записи в текущую надо нажать <**Ctrl**>+<«> (кавычки). Для экономии времени при вводе данных также можно пользоваться инструментами редактирования: вырезанием (<**Ctrl**>+<**X**>), копированием (<**Ctrl**>+<**C**>) и вставкой (<**Ctrl**>+<**V**>) в буфер.

Для перехода между столбцами и к следующей записи используется клавиша **Tab** или комбинация клавиш **Shift+Tab.**

Для перехода между записями также служат кнопки переходов в нижнем левом углу окна, где также отображается общее количество записей и номер текущей записи. Для перехода к конкретной записи вместо номера текущей записи нужно ввести требуемый номер и нажать клавишу **Enter.** Переход к другой записи также может быть осуществлен с помощью меню **Правка**, **Перейти**.

Редактирование данных - курсор переводится в нужную ячейку, старые данные удаляются (клавишами <**Del**>, <**Backspase**> или выберите команду **Удалить записи** меню **Правка**) и вводятся новые данные. Удалять можно не только данные в ячейках, но и целиком строки, предварительно их выделив. Но если таблица большая, то редактируемые данные надо сначала найти. Выберите команду **Правка–Найти** или щелкните на панели инструментов кнопку **Найти.** В появившемся окне вводят образец искомых данных и щелкают по кнопке **Найти**. Если значение найдено, курсор перейдет в эту ячейку. Иногда требуется большое количество одинаковых данных заменить на другое значение. Для этого надо открыть пункт меню **Правка** и выполнить команду **Заменить**. В появившемся окне ввести образцы того, что надо найти и на что заменить.

2.4.8 Связанные таблицы

Access автоматически добавляет к таблице кнопки развертывания. В режиме таблица слева от каждой записи находится маркер развертывания «+». Эти значки позволяют развернуть подтаблицу, в которой отображаются соответствующие записи из связанной таблицы. Чтобы развернуть подтаблицу для определенной записи, надо щелкнуть по кнопке со значком «+». Откроется подтаблица с записями из связанной таблицы, соответствующими текущей записи главной таблицы (рис. 2.19).

В подтаблице можно редактировать данные, а также добавлять, удалять, сортировать записи и устанавливать фильтры. Чтобы по окончанию работы закрыть подтаблицу, надо щелкнуть на кнопке свертывания (значок «–»). Чтобы свернуть все раскрытые подтаблицы разом, необходимо щелкнуть по кнопке свертывания верхнего уровня. Существует возможность не

только закрыть подтаблицу, но и удалить ее. Выберите команду **Формат, Подтаблица, Удалить.** После удаления подтаблицы ее можно снова восстановить. Для восстановления удаленной таблицы при закрытии таблицы можно отказаться от внесенных изменений. Если между двумя таблицами не определено никакой связи, к таблице можно добавить подтаблицу, если эти таблицы имеют общее поле. Последовательность действий при добавлении таблицы следующая: открыть необходимую таблицу в режиме **Таблицы**, выбрать команду **Вставка, Подтаблица.**

преподаватели : таблица

код преподава	Фамилия	Имя	Отчество	Дата рождения	Должность
1	Боровиков	Виктор	Сергеевич	23.11.1964	доцент
2	Митина	Светлана	Викторовна	22.10.1972	старший преподаватель
3	Смирнов	Павел	Юрьевич	05.03.1940	профессор
4	Жбанова	Ольга	Ивановна	09.07.1980	ассистент
5	Ипатова	Татьяна	Павловна	30.05.1966	старший преподаватель
6	Емец	Татьяна	Ивановна	09.12.1956	доцент
7	Коробков	Валентин	Андреевич	21.01.1976	ассистент

Код дисциплин	Название дисциплины	Семестр	Экзамен
7	Информатика	1	зачет

код студента	номер семестр	Оценки
1	1	4
0	0	

Код дисциплин	Название дисциплины	Семестр	Экзамен
8	информационные техно	2	экзамен

код студента	номер семестр	Оценки
2	3	4
0	0	

(Счетчик)

Рис. 2.19 Диалоговое окно таблицы Преподаватели
с подтаблицами Дисциплины, Оценки.

2.4.9 Сортировка данных

Для удобства просмотра можно сортировать записи в таблице в определенной последовательности, например, в таблице **Преподаватели** записи можно отсортировать в порядке убывания стажа преподавателей. Кнопки сортировки на панели инструментов (или команды меню **Записи, Сортировка, Сортировка по возрастанию (сортировка по убыванию))** позволяют сортировать столбцы по возрастанию или по убыванию.

Прежде чем щелкнуть по кнопке сортировки, следует выбрать поля, используемые для сортировки. Для выбора поля достаточно поместить курсор в любую его запись. После этого щелкните по кнопке сортировки - и данные отобразятся в отсортированном порядке, В режиме таблицы можно выделить сразу два или несколько соседних столбцов, а затем выполнить по ним сортировку. По умолчанию в Access сортировка записей начинается с крайнего левого выделенного столбца. Для восстановления порядка отображения записей используется команда **Записи**, **Удалить фильтр**. Если требуется осуществить сортировку по нескольким полям, причем по одному из них – по возрастанию, а по другому – по убыванию, придется воспользоваться расширенным фильтром.

2.4.10 Фильтрация данных

Фильтрация дает возможность ограничить множество просматриваемых записей только теми, которые удовлетворяют некоторому условию. Для управления фильтрацией используются кнопки на панели инструментов:

- Фильтр по выделенному – производит отбор записей в текущем поле, соответствующих записи выбранной ячейки;

- Изменить фильтр – открывает окно фильтра, в котором выводится пустая форма или таблица для определения параметров последующей фильтрации;

- Применение фильтра (Удалить фильтр) – применяет или отменяет назначенную фильтрацию данных таблицы или формы.

Существует четыре вида фильтров:

1. **Фильтр по выделенному**: определяет какие записи выводятся на экран путем выделения данных в таблице в Режиме Таблицы (в том случае, если выделенный фрагмент отсутствует, по умолчанию в качестве условия воспринимается значение той ячейки, в которой стоял курсор). Для фильтрации данных по полям Должность и Дисциплина (рис. 2.20): щелкните по записи Доцент поля Должность и выполните команду **Записи, Фильтр, Фильтр по выделенному**. В таблице останутся только записи о преподавателях доцентах, для отмены

фильтрации выполните команду **Записи, Удалить фильтр**. В таблице появятся все данные.

преподаватели : таблица

	код преподава	Фамилия	Имя	Отчество	Дата рождения	Должность	Стаж	Кафедра	
+	1	Боровиков	Виктор	Сергеевич	23.11.1964	доцент	17	Информатики	77-
	(Счетчик)						0		

Запись: 2 из 2 (Фильтр)

Рис. 2.20. Диалоговое окно Фильтр по выделенному

2. **Обычный фильтр**: по команде **Изменить фильтр** определяет, какие записи выводятся на экран путем выбора в качестве условия значения из списка значений каждого поля.

3. В **Поле Фильтр для:** условие задается непосредственно в контекстном меню для того поля, в котором это меню вызывалось.

4. **Расширенный фильтр:** позволяет проводить не только фильтрацию, но одновременно и сортировку по возрастанию или убыванию по нескольким полям одновременно (**Записи, Фильтр, Расширенный фильтр**), а также поиск записей, удовлетворяющих одному или нескольким условиям, ввод выражений в качестве условий (рис. 2.21).

Поле:	Название дисципл	Семестр
Сортировка:	по возрастанию	
Условие отбора:	Like "инф*"	>1
или:		

Рис. 2.21. Диалоговое окно Расширенный фильтр

Фильтры строятся на основе задания условий, используя операторы:

- **арифметические**: + - * / \ MOD(остаток от деления)
- **отношений**: < <= > >= =
- **логические**: AND OR NOT XOR EQV IMP
- **специальные**: IN, BETWEEN, LIKE.

- & - конкатенация (соединение строк) - *соединение частей текста &,* например, =[Фамилия] & " " & [Имя] ;

Логические операторы (AND, OR, IS, NOT, BETWEEN И LIKE) возвращают в качестве результата одно из значений "Истина" (True), "Ложь" (False) или пустое значение (Null), если результат вычислить невозможно. Логические операторы используются для создания сложных условий.

OR - ИЛИ, используемый в логических выражениях для отбора записей удовлетворяющих ИЛИ одному ИЛИ другому условию (должно выполняться хотя бы одно из условий). Чтобы скомбинировать выражения с условием И, используется оператор **AND**, в этом случае данные должны удовлетворять всем условиям выборки одновременно (должны выполняться все условия).

Not — логического отрицания;

Like – оператор, для работы в текстовых полях. В образец поиска можно включать символы:?, *,[ ], #, !.

Примеры:

Like “А” - строка начинается с “А”.*

*Like “*ова” - строка заканчивается на “ова”.*

Like “[А-Д]” – строка начинается с одной из букв от А до Д.*

*Like “*пр*” - внутри строки сочетание букв пр.*

Like “?[а-д]р[0-9]” – наличие произвольного символа (?) в первой позиции, а-д во второй, р в третьей, цифра в четвертой.*

Like “[!0-9]к#” – в первой позиции любой символ, за исключением цифр от 0 до 9, к во второй, третьей позиции должна быть цифра.*

Between – для выбора значений из определенного интервала. **Between...And** (от... до..), который выполняет выборку в период от одного значения до другого включительно. При задании Условия отбора можно воспользоваться только одним оператором **AND** (*И*).

Пример:

>=200 AND <=300

Between 200 and 300

IN – оператор проверяет, совпадает ли значение выражения с одним из элементов указанного списка, который задается в круглых скобках.

Пример:

In ("Кемерово"; "Томск") любое значение из списка городов;
In (1; 3; 5; 11; 17) любое значение из списка чисел.

2.5 Создание запросов

Запросы являются мощным средством обработки данных, хранимых в таблицах Access. С помощью запросов можно просматривать, анализировать и изменять данные из одной или нескольких таблиц. Они также используются в качестве источника данных для форм и отчетов. Запросы позволяют вычислять итоговые значения, а также выполнять вычисления над группами записей. Запросы можно создавать самостоятельно и с помощью мастеров. Мастера запросов автоматически выполняют основные действия в зависимости от ответов пользователя на поставленные вопросы. Запросы могут быть однотабличными, либо многотабличными.

В Access можно создавать следующие типы запросов:

- **запрос на выборку** (критерий отбора в полях бланка);
- **запрос с параметрами** (критерий отбора задает пользователь, введя нужный параметр при вызове запроса);
- **перекрестный запрос** (позволяет создавать результирующие таблицы на основе результатов расчетов, полученных при анализе группы таблиц);
- **запрос на изменение** (удаление, обновление и добавление) записей (позволяет автоматизировать заполнение полей таблиц);
- **запросы SQL** (на объединение, к серверу, управляющие, подчиненные), написанные на языке запросов SQL.

Работа с запросами в Access может выполняться в трех режимах:

1 Режим **Конструктора** (основной) используется при создании нового запроса или при изменении структуры уже созданного запроса.

2 **Режим SQL** используется для просмотра уже созданного запроса, а также для создания нового запроса или при изменении структуры уже созданного в стиле QBE-запроса.

3 **Режим таблицы** используется для просмотра результатов запроса.

Переход из одного режима в другой может выполняться:

1 По командам **Вид-Конструктор**, **Вид-Режим SQL** и **Вид-Режим таблицы** главного меню;

2 Нажатием экранных кнопок панели инструментов конструктора запросов - **Режим конструктора**, **Режим таблицы** и **Режим SQL** (рис. 2.22).

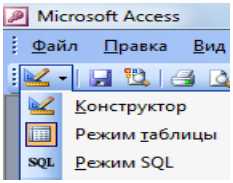


Рис. 2.22. Экранные кнопки конструктора запросов

Инструменты создания запросов. Для создания запросов в Access используются мастер или конструктор. Для создания нового запроса надо в окне базы данных выбрать вкладку Запросы, щелкнуть по кнопке Создать и в открывшемся окне выбрать один из пяти пунктов: Конструктор, Простой запрос, Перекрестный запрос, Повторяющиеся записи, Записи без подчиненных. Конструктор позволяет самостоятельно создать любой тип запроса, но этот режим рекомендуется пользователям, уже имеющим некоторый опыт создания запросов. Простой запрос позволяет создать с помощью Мастера запрос на выборку из определенных полей таблиц или других запросов (наилучший способ создания запроса для начинающих пользователей).

Выбираем **Объекты, Запросы, Создать, Конструктор**, в окне нового запроса появиться окно «Добавление таблицы», позволяющее выбрать таблицы-запросы, являющиеся источником данных для создаваемого запроса (рис 2.23). Для того чтобы указать, на чем будет базироваться создаваемый запрос (таблице, запросе или том и другом одновременно), надо просто выбрать соответствующую закладку. Установив в появившемся списке доступных таблиц/запросов указатель на имя добавляемой таблицы или запроса, или выполнить двойной щелчок “мышью” или нажать клавишу **Ввод**. Допускается одновременное добавление в запрос нескольких таблиц или запросов. Для этого следует, удерживая нажатой клавишу **Ctrl**, выбрать имена

добавляемых таблиц или запросов и нажать кнопку – **Добавить,** через команду меню **Запрос, Отобразить таблицу**.

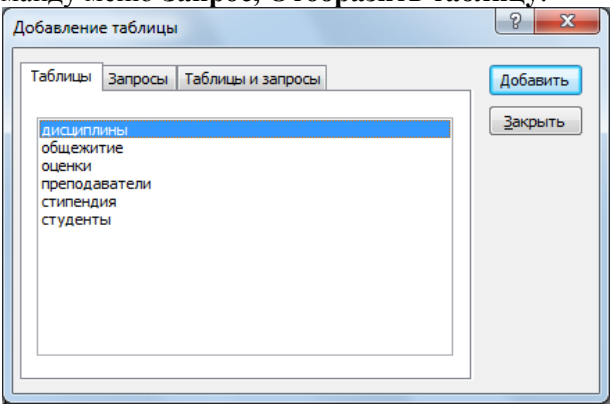


Рис. 2.23. Выбор источника запроса

Если какую-то из таблиц в запрос, или по каким-либо иным причинам надо удалить, выбрать таблицу и нажать клавишу **Del** или выбрать в меню **Запрос, Удалить таблицу**.

После того, как определили исходные таблицы, необходимо выбрать поля, используемые в создаваемом запросе. Для переноса поля в бланк запроса: с помощью мыши, двойной щелчок мышью на имени соответствующего поля в списке полей.

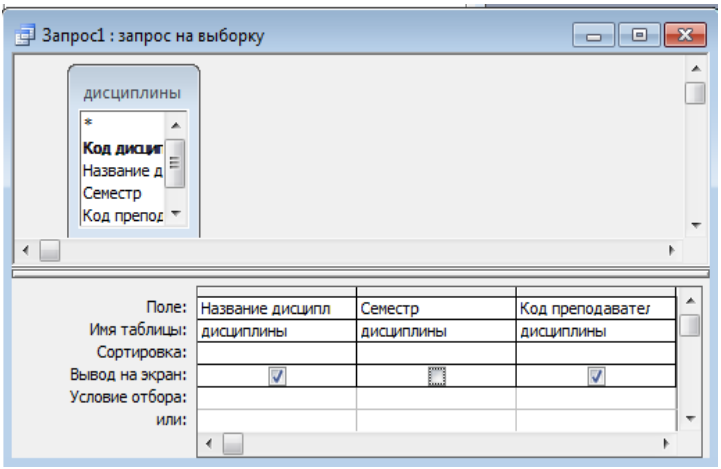


Рис. 2.24 Заполненный бланк запроса по образцу.

Поля, выводимые в ответ, указываются в строке конструктора запроса **Вывод на экран** (“**v**” - галочка). Можно перенести в бланк запроса одновременно все поля. Для этого надо установить указатель на заголовок списка полей и дважды щелкнуть кнопкой мыши или установить указатель на символ звездочки (*****) и нажать кнопку мыши.

После формирования запроса нажмите или на кнопку быстрого доступа – **Запуск** или выберите из меню **Запрос, Запуск**. После просмотра результатов можно переключится в режим **Конструктора**, нажав на кнопку быстрого доступа **Вид**.

2.5.1 Запросы на выборку

Запрос на выборку является самым распространенным типом запроса. Данный запрос определяет, какие записи или поля из одной или нескольких таблиц будут отображены при его выполнении.

Формирование запросов на выборку:

1. На основе таблицы **Преподаватели** создайте простой запрос на выборку, в котором должны отображаться Код преподавателей, Фамилии преподавателей и их Должность.

2. Данные запроса отсортируем по должностям и сохраним запрос.

Для создания простого запроса: в окне базы данных откройте вкладку **Запросы**, в открывшемся окне щелкните по кнопке **Создать**, из появившихся пунктов окна **Новый запрос** выберите **Простой запрос** и щелкните по кнопке **Ок**. В появившемся окне в строке **Таблицы, Запросы** выберите таблицу **Преподаватели** и переместите поля Код преподавателя, Фамилия, Должность, Стаж, Кафедра и нажмите кнопку **Далее** (рис 2.25-2.26).

Поле:	код преподавателя	Фамилия	Должность	Стаж	Кафедра
Имя таблицы:	преподаватели	преподаватели	преподаватели	преподаватели	преподаватели
Сортировка:					
Вывод на экран:	☑	☑	☑	☑	☑
Условие отбора:					
или:					

Рис. 2.25. Структура запроса в режиме конструктора

Запрос1 : запрос на выборку

код преподава	Фамилия	Должность	Стаж	Кафедра
7	Коробков	ассистент	5	Информатики
6	Емец	доцент	12	Математика
5	Ипатова	старший преподаватель	10	Химии
4	Жбанова	ассистент	4	Иностранного языка
3	Смирнов	профессор	27	физика
2	Митина	старший преподаватель	8	Экономики
1	Боровиков	доцент	17	Информатики
(Счетчик)			0	

Рис. 2.26. Результат работы запроса

Для сортировки данных щелкните в любой строке поля **Должность**, отсортируйте данные по убыванию, сохраним запрос и закройте окно запроса.

Создадим запрос на выборку, на основе таблиц **Преподаватели** и **Дисциплины** используя Like – для поиска образцов в текстовых полях. Например, Like «ст*» используем для выбора преподавателей занимающих должность, которая начинается с букв «ст» и стажем работы свыше 5 лет (рис. 2.27-2.28).

Поле:	Название дисципл	код преподавателя	Фамилия	Должность	Стаж
Имя таблицы:	дисциплины	преподаватели	преподаватели	преподаватели	преподаватели
Сортировка:					
Вывод на экран:	☑	☑	☑	☑	☑
Условие отбора:				Like "ст*"	>5
или:					

Рис. 2.27. Структура запроса в режиме конструктора

Запрос2 : запрос на выборку

Название дисциплины	код преподава	Фамилия	Должность	Стаж
Экономика	2	Митина	старший преподаватель	8
Химии	5	Ипатова	старший преподаватель	10
	(Счетчик)			

Запись: 2 из 2

Рис.2.28. Результат работы запроса

Создадим запрос, отображающий дату рождения студента в определенном числовом диапазоне (Between #01.01.1992# And #01.01.1993# Or Between #01.01.1994# And #01.01.1995#), а также выводим поля Фамилию, Код студента (рис. 2.29-2.30).

Поле:	код студента	Фамилия	дата рождения
Имя таблицы:	студенты	студенты	студенты
Сортировка:			
Вывод на экран:	☑	☑	☑
Условие отбора:			Between #01.01.1992# And #01.01.1993#
или:			Between #01.01.1994# And #01.01.1995#

Рис. 2.29. Структура запроса в режиме конструктора

Запрос1 : запрос на выборку

код студента	Фамилия	дата рождения
5	Крылова	23.08.1994
6	Шевченко	03.09.1992
9	Киршин	06.07.1992
10	Петров	23.01.1992
11	Перлов	07.11.1992
12	Соколова	02.05.1992
13	Степанская	02.04.1992
0		

Рис. 2.30. Результат работы запроса

В бланке запроса можно задавать несколько условий. При этом условия, размещенные в разных столбцах одной строки объединяются операцией «**И**» (одновременно должны выполняться условия, заданные в каждом из столбцов). Условия, размещенные в разных строках, объединяются операцией «**ИЛИ**» (должны соблюдаться условия хотя бы в одной из строк).

2.5.2 Запрос с параметром

Как правило, запросы с параметром (параметрический запрос) создаются в тех случаях, когда предполагается выполнять этот запрос многократно, изменяя лишь условия отбора. В отли-

чие от запроса на выборку, где для каждого условия отбора создается свой запрос, и все эти запросы хранятся в БД, параметрический запрос позволяет создать и хранить один единственный запрос и вводить условие отбора (значение параметра) при запуске этого запроса, каждый раз получая новый результат.

В качестве параметра может быть любой текст, смысл которого определяет значение данных, которые будут выведены в запросе. Значение параметра задается в специальном диалоговом окне. В случае, когда значение выводимых данных должно быть больше или меньше указываемого значения параметра, в поле "**Условие отбора**" бланка запроса перед параметром, заключенным в квадратные скобки ставится соответствующий знак. Можно также создавать запрос с несколькими параметрами, которые связываются друг с другом логическими операциями "**И**" и "**ИЛИ**". В момент запуска запроса на выполнение Access отобразит на экране диалоговое окно для каждого из параметров.

Создаем запрос на выборку с параметром, в котором должны отображаться фамилии преподавателей и преподаваемые ими дисциплины. В качестве параметра задаем фамилию преподавателя и выполним этот запрос для преподавателя Смирнов. В качестве условия введите параметр, заключенный в квадратные скобки (например, [**введите фамилию**]). После выполнения запроса, на экране появится таблица с данными о преподавателе Смирнове и преподаваемая дисциплина.

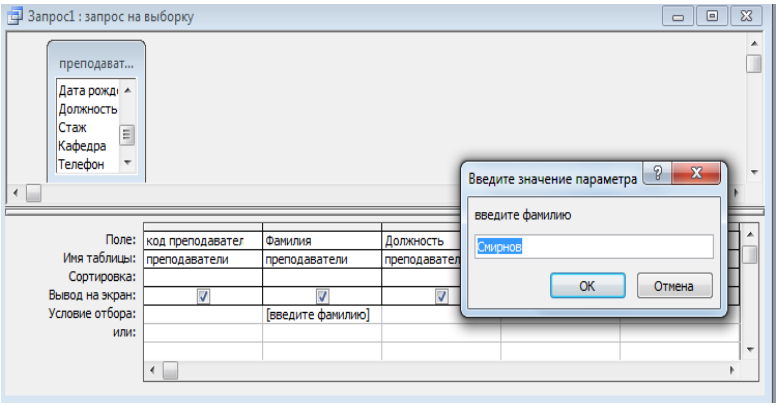


Рис. 2.31 Структура запроса в режиме конструктора

Запрос1 : запрос на выборку

	код преподава	Фамилия	Должность	Стаж	Кафедра
▶	3	Смирнов	профессор	27	физика
*	(Счетчик)			0	

Рис. 2.32. Результат работы запроса с параметром

2.5.3 Вычисления в запросах

Запрос можно использовать для выполнения расчетов и подведения итогов из исходных таблиц. Для создания вычисляемых полей используются математические и строковые операторы. При этом Access проверяет синтаксис выражения и автоматически вставляет следующие символы:

- квадратные скобки (**[]**), в них заключаются имена элементов управления;

- знаки номеров (#), в них заключаются распознанные даты;

- кавычки (" "), в них заключается текст, не содержащий пробелов или знаков пунктуации.

Чаще в идентификаторах встречается оператор "!". Он используется для ссылок на объекты. При ссылке на поле таблицы он служит для отделения имени поля от имени таблицы. Сами имена заключаются в квадратные скобки, и ссылка имеет следующий вид:[<имя таблицы>]![<имя поля>].

Ниже перечислены некоторые функции, которые могут быть полезны при построении запросов. Следует иметь в виду, что при использовании функций в конструкторе запросов их аргументы отделяются не запятой, а точкой с запятой.

Функции обработки текста:

– **Left**(строка, n) – возвращает **n** левых символов строки.

– **Right**(строка, n) – возвращает **n** правых символов строки.

– **Mid**(строка, n1, n2) – возвращает **n2** символов строки, начиная с позиции **n1**.

– **InStr**(строка1, строка2) – номер позиции, с которой **строка2** входит в **строка1**.

– **Ltrim**(строка), **Rtrim**(строка) – удаляют пробелы из начала и конца строки соответственно.

– **Trim**(строка) – удаляет пробелы из начала и конца строки.

Функции обработки даты и времени:

– **Date**() – возвращает текущую дату.

– **Now**() – возвращает текущую дату и время.

– **DateDiff**(интервал, дата1, дата2) – определяет разницу между датами.

– **DateAdd**(интервал, число, дата) – будущая дата, отстоящая от указанной на заданное число интервалов.

– Функции **Year**(дата), **Month**(дата), **Day**(дата) – возвращают число - значение года, месяца и дня для указанной даты.

Функции преобразования типов:

– **Str**(аргумент) – преобразует значение аргумента в текстовую строку

– **Val**(строка) – преобразует строку в число

– **Int**(число) – возвращает целую часть числа

Условная функция: IIf(выражение, если истинно, если ложно) – вычисляет значение аргумента **выражение**. Если значение истинно, возвращает значение второго аргумента, если ложно – значение третьего аргумента.

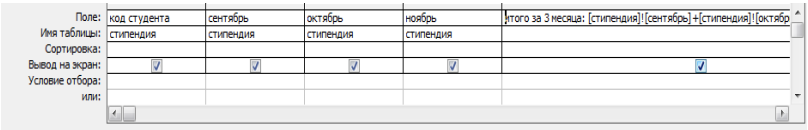
Поле, содержимое которого является результатом расчета по содержимому других полей, называется **вычисляемым полем. Вычисляемое поле** существует только в результирующей таблице. Общий формат вычисляемого поля выглядит так:

Имя вычисляемого поля: Выражение.

Создадим запрос с вычисляемыми полями на основе таблицы **Стипендия**. Необходимо посчитать выплаченную стипендию студентам за квартал (рис. 2.33-2.34).

Пример:

итого за 3 месяца: [стипендия]![сентябрь]+[стипендия]![октябрь] +[стипендия]![ноябрь]



Поле:	код студента	сентябрь	октябрь	ноябрь	Итого за 3 месяца: [стипендия]![сентябрь]+[стипендия]![октябр
Имя таблицы:	стипендия	стипендия	стипендия	стипендия	
Сортировка:					
Вывод на экран:	☑	☑	☑	☑	☑
Условие отбора:					
или:					

Рис. 2.33. Структура запроса в режиме конструктора

Запрос2 : запрос на выборку

код студента	сентябрь	октябрь	ноябрь	итого за 3 месяца
1	1 000,00р.	1 000,00р.	1 000,00р.	3 000,00р.
2	1 500,00р.	1 000,00р.	1 000,00р.	3 500,00р.
3	0,00р.	0,00р.	0,00р.	0,00р.

Запись: 4 из 20

Рис. 2.34. Результат работы запроса с вычисляемым полем

Для того чтобы ввести сложные вычисления используйте окно **Построитель Выражений**, которое вызывается нажатием кнопки **Построить** на панели инструментов, либо соответствующей командой контекстного меню (рис. 2.35). Построитель выражений облегчает создание выражений, позволяя выбирать его составляющие элементы (арифметические операции, встроенные функции, названия полей имеющихся в БД таблиц и запросов и т.п.) при помощи кнопок и списков. В верхней части окна расположена пустая область ввода, предназначенная для создания выражения. В нижней – находятся три списка, предназначенные для поиска необходимых полей и функций. Построитель поможет правильно построить выражение. Щелкните на кнопке Ок, и введенное выражение будет перенесено в бланк запроса.

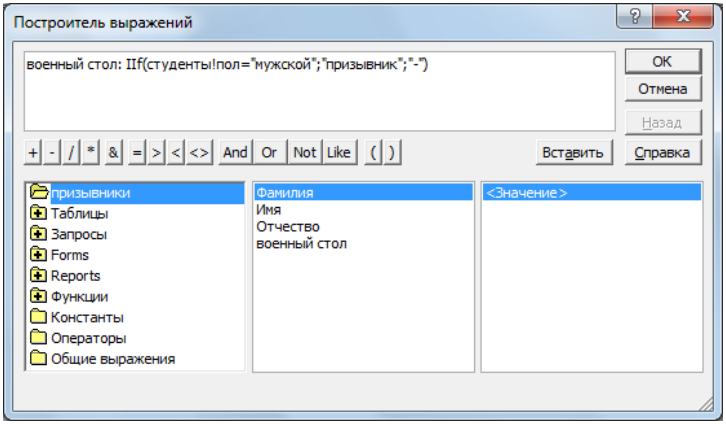


Рис. 2.35. Окно Построителя

Создадим запрос с вычисляемыми полями на основе таблицы **Студенты,** используя условную функцию **IIf ()**. Необходимо получить таблицу студентов мужского пола подлежащих оформлению в «военном столе» (рис. 2.36-2.37).

Пример:

военный стол: IIf(студенты!пол="мужской";"призывник";"-")

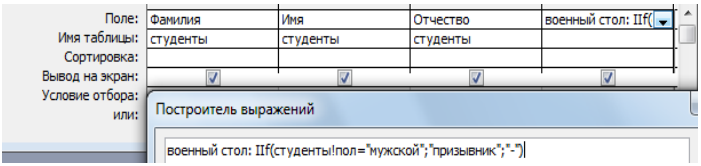
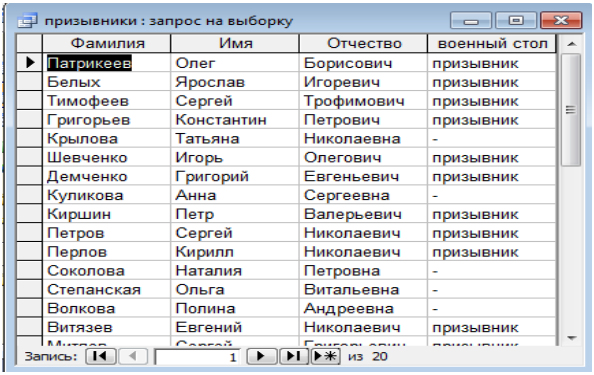


Рис. 2.36. Структура запроса в режиме конструктора



призывники : запрос на выборку

Фамилия	Имя	Отчество	военный стол
Патрикеев	Олег	Борисович	призывник
Белых	Ярослав	Игоревич	призывник
Тимофеев	Сергей	Трофимович	призывник
Григорьев	Константин	Петрович	призывник
Крылова	Татьяна	Николаевна	-
Шевченко	Игорь	Олегович	призывник
Демченко	Григорий	Евгеньевич	призывник
Куликова	Анна	Сергеевна	-
Киршин	Петр	Валерьевич	призывник
Петров	Сергей	Николаевич	призывник
Перлов	Кирилл	Николаевич	призывник
Соколова	Наталия	Петровна	-
Степанская	Ольга	Витальевна	-
Волкова	Полина	Андреевна	-
Витязев	Евгений	Николаевич	призывник

Запись: 1 из 20

Рис. 2.37. Результат работы запроса с вычисляемым полем

2.5.4 Итоговые запросы

Для получения итоговых значений по группам данных используются **итоговые запросы**. Для задания вычислений итоговых значений щелкните на кнопке Групповые операции (∑) на панели инструментов конструктора запросов, чтобы в бланке запроса появилась строка «**Групповая операция**». Тогда записи по каждому полю будут группироваться. Для вычисления ито-

гов замените значение **Группировки** в строке «Групповая операция» на конкретную итоговую функцию.

Функция Описание:

– **Sum** Суммирование значений определенного поля.

– **Avg** Вычисление среднего значения данных определенного поля.

– **Min** Вычисление минимального значения поля.

– **Max** Вычисление максимального значения поля.

– **Count** Вычисление количества записей, отобранных запросом по условию.

– **First** Определяется первое значение в указанном поле записей, отобранных запросом.

– **Last** Определяется последнее значение в указанном поле записей, отобранных запросом.

Например, необходимо посчитать итоговую успеваемость студентов, по разным предметам. Так как один и тот же студент сдает несколько экзаменов и за определенный семестр, используем группировку по полям [код студента] и [Номер семестра].

Поле:	код студента	номер семестра	Sum - Оценки: Оце	Avg - Оценки: Оце	Min - Оценки: Оцен	Max - Оценки: Оце	Count - оценки: Co
Имя таблицы:	оценки	оценки	оценки	оценки	оценки	оценки	
Групповая операция:	Группировка	Группировка	Sum	Avg	Min	Max	Выражение
Сортировка:							
Вывод на экран:	☑	☑	☑	☑	☑	☑	☑
Условие отбора:		1					
или:							

Рис. 2.38. Структура запроса в режиме конструктора

оценки Запрос2 : запрос на выборку

код студента	номер семестр	Sum - Оценки	Avg - Оценки	Min - Оценки	Max - Оценки	Count - оценки
1	1	21	4,2	4	5	5
2	1	18	4,5	4	5	4
3	1	15	3,75	3	5	4
4	1	17	4,25	4	5	4
5	1	20	5	5	5	4
6	1	18	4,5	4	5	4
7	1	14	3,5	3	4	4
8	1	17	4,25	3	5	4
9	1	16	4	4	4	4
10	1	19	4,75	4	5	4
11	1	20	5	5	5	4
12	1	12	3	2	4	4
13	1	20	5	5	5	4
14	1	19	4,75	4	5	4

Запись: 1 из 20

Рис. 2.39. Результат работы запроса на группировку

2.5.5 Перекрестный запрос

В перекрестном запросе отображаются результаты статистических расчетов (суммы, количество записей, средние значения), выполненных по данным из одного поля таблицы. Для построения перекрестного запроса, данные предварительно должны быть приведены к определенному виду: следует сформировать три колонки данных. Перекрестный запрос будет использовать данные из одной колонки в качестве названий строк таблицы, другой – в качестве названий столбцов, а данные из третьей колонки будут размещены в таблице на пересечении соответствующих строк и столбцов.

Для создания перекрестного запроса:

– На вкладке Запросы щелкните по кнопке **Создать**, выберите **Перекрестный запрос** и щелкните по кнопке **ОК**;

– Щелкните по ячейке Запросы, выберите таблицу **Оценки** и щелкните по кнопке **Далее**;

– Выберите поле код студента и щелкните по кнопке **Далее**;

– Выберите поле код дисциплины и щелкните по кнопке **Далее**;

– Выберите функцию и щелкните по кнопке **Далее**;

– Выберите название запроса Средние оценки и щелкните по кнопке **Готово**.

Далее перейдите в режим конструктора и сделайте ручную корректировку запроса добавив таблицы **Студенты** и **Оценки** (для замены полей Код студента – Фамилию из таблицы **Студенты** и т.д. и добавления столбцов с другими функциями).

Результат работы перекрестного запроса приведен на рис. 2.40-2.41.

Поле:	Фамилия	Название дисципл	Оценки	Итоговое значени	Оценки
Имя таблицы:	студенты	дисциплины	оценки	оценки	оценки
Групповая операция:	Группировка	Группировка	Sum	Sum	Avg
Перекрестная таблица:	Заголовки строк	Заголовки столбц	Значение	Заголовки строк	Заголовки строк
Сортировка:					
Условие отбора:					

Рис. 2.40. Структура запроса в режиме конструктора

оценки_перекрестный2 : перекрестный запрос

Фамилия	Итоговое значе	Avg-Оценки	Иностранного языка	Информатика	информационн	Математика	Физика	Экономика
Арбузов	17	4,25		4		4	4	5
Белых	22	4,4	4	5	4	4		5
Витязев	15	3,75		4		3	4	4
Волкова	19	4,75		4		5	5	5
Григорьев	17	4,25	5	4		4		4
Демченко	14	3,5	4	4		3		3

Рис. 2.41. Результат работы перекрестного запроса

2.5.6 Запросы на изменение

Запрос на изменение - это запрос, который за одну операцию вносит изменения в несколько записей. Существует четыре типа запросов на изменение: на удаление, обновление и добавление записей, а также на создание таблицы.

2.5.7 Запрос на обновление записей

Вносит общие изменения в группу записей одной или нескольких таблиц. Например, на 10 процентов увеличивается зарплата преподавателей. Запрос на обновление записей позволяет изменять данные в существующих таблицах.

Создайте новый запрос на выборку и проверьте его корректность, перейдя в режим Таблица. Преобразуйте запрос на выборку в запрос на обновление. Для этого, вернувшись в режим Конструктора, выберите команду **Обновление (**меню **Запрос)**. В появившейся в бланке запроса строке **Обновление** в соответствующих столбцах задайте новые значения полей таблицы. В качестве таких, могут выступать и вычисляемые значения. В случае необходимости воспользуйтесь Построителем выражений.

В строке конструктора запроса (рис. 2.42) Обновление в поле Зарплата введите: **[Зарплата]*1,1**;

Поле:	Должность	Зарплата
Имя таблицы:	преподаватели	преподаватели
Обновление:		[преподаватели]![Зарплата]*1,1
Условие отбора:		

Рис. 2.42. Запрос на обновление записей

2.5.8 Запрос на добавление записей

Запрос на добавление добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц.

Для задания запроса такого типа надо сначала создать запрос, содержащий таблицу, записи из которой необходимо добавить в другую таблицу. Затем в режиме конструктора запроса надо нажать стрелку рядом с кнопкой **Тип запроса** на панели инструментов и выбрать команду **Добавление** (либо выбрать соответствующую позицию в меню **Запросы**). На экране появится диалоговое окно **Добавление** (рис 2.43). В поле **Имя таблицы** надо ввести имя таблицы, в которую необходимо добавить записи.

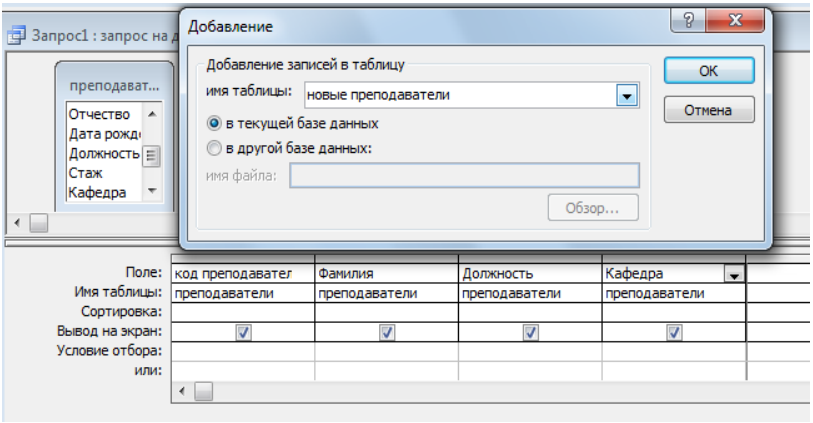


Рис. 2.43 Создание запроса на добавления записей

2.5.9 Запрос на удаление записей

Запрос на удаление удаляет группу записей, удовлетворяющих заданным условиям, из одной или нескольких таблиц, причем можно удалять только всю запись, а не отдельные поля внутри ее. Запрос на обновление записей вносит общие измене-

ния в группу записей одной или нескольких таблиц. Запрос на добавление добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц. Запрос на создание таблицы создает новую таблицу на основе всех или части данных из одной или нескольких таблиц.

Создайте новый запрос на выборку на основе таблицы Студенты - выберите поля Фамилия. Преобразуйте запрос на выборку в запрос на удаление записей. Для этого, вернувшись в **режим Конструктора**, выберите команду **Удалить (**меню **Запрос)**. В появившейся строке «**Удалить**» установите критерии отбора. Выполните запрос (рис 2.44).

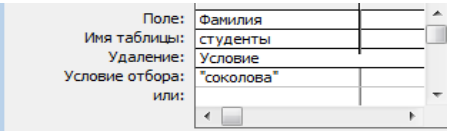


Рис. 2.44. Структура запроса в режиме конструктора

2.5.10 Запрос на создание таблицы

БД на физическом уровне хранит только таблицы. Набор записей запросов физически не существует в БД. Access создает его из данных таблиц только во время выполнения запроса. Иногда возникает необходимость сохранить извлекаемые с помощью запроса на выборку данные в новой таблице (рис. 2.45-2.46):

1. Создайте новый запрос на выборку тех записей, из которых должна состоять создаваемая с помощью запроса таблица.

2. Проверьте правильность отбора записей, перейдя **в режим Таблица**.

3. Преобразуйте запрос на выборку в запрос на создание новой таблицы. Для этого, вернувшись в **режим Конструктора**, выберите **Создание таблицы (**меню **Тип запроса)**.

4. В появившемся окне введите имя новой таблицы.

5. Выполните запрос (кнопка на панели инструментов).

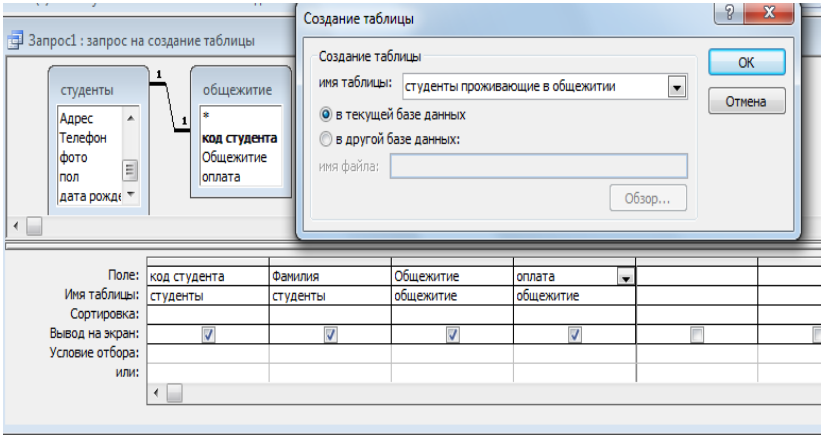
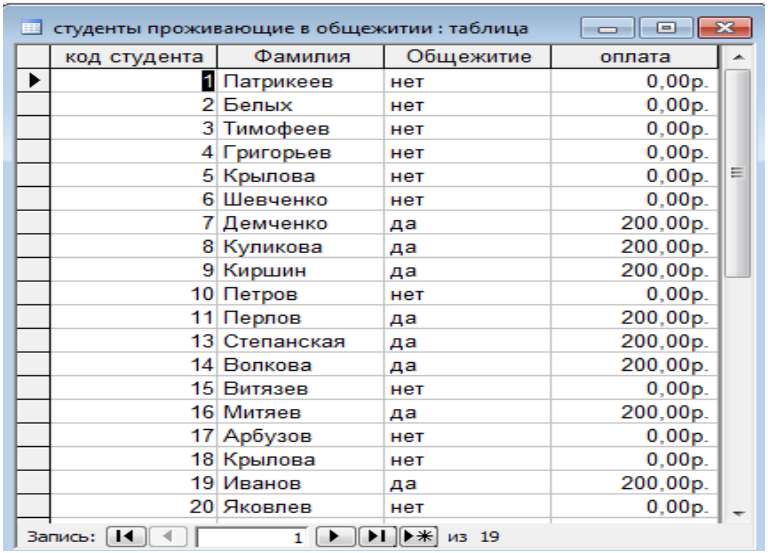


Рис.2.45. Запрос на создание новой таблицы



студенты проживающие в общежитии : таблица

код студента	Фамилия	Общежитие	оплата
1	Патрикеев	нет	0,00р.
2	Белых	нет	0,00р.
3	Тимофеев	нет	0,00р.
4	Григорьев	нет	0,00р.
5	Крылова	нет	0,00р.
6	Шевченко	нет	0,00р.
7	Демченко	да	200,00р.
8	Куликова	да	200,00р.
9	Киршин	да	200,00р.
10	Петров	нет	0,00р.
11	Перлов	да	200,00р.
13	Степанская	да	200,00р.
14	Волкова	да	200,00р.
15	Витязев	нет	0,00р.
16	Митяев	да	200,00р.
17	Арбузов	нет	0,00р.
18	Крылова	нет	0,00р.
19	Иванов	да	200,00р.
20	Яковлев	нет	0,00р.

Запись: 1 из 19

Рис. 2.46. Результат по созданию новой таблицы

2.6 Создание форм

При вводе данных можно не только помещать вычисляемые поля в форму, но и добавлять расширенные правила проверки корректности ввода и элементы управления (например, переключатели, флажки, раскрывающиеся списки). Линии, рамки, цвета и фоновые изображения улучшают внешний вид данных, облегчают восприятие формы и повышают продуктивность работы. В дополнение к этому OLE-объекты (такие, как рисунки и графики) можно увидеть только в форме или в отчете.

Создать форму можно несколькими способами, которые можно увидеть, если в режиме базы данных открыть вкладку **Формы** и щелкнуть по кнопке **Создать**, то откроется окно, в котором указаны способы создания формы (рис. 2.47).

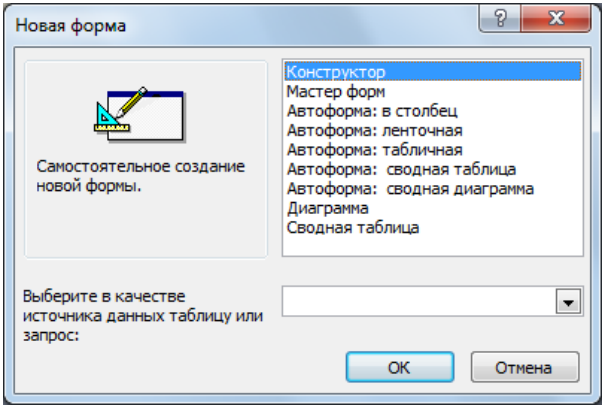


Рис. 2.47. Окно построения формы

Наиболее удобным и гибким способом создания форм является Мастер форм. В этом режиме можно выбрать поля таблицы для отображения в форме, стиль и цвет оформления фона и ячеек, а также вид формы. Мастер форм предлагает четыре вида формы представления данных: в один столбец, ленточная, табличная и выровненная.

Конструктор позволяет вручную создавать и редактировать любые формы. Иногда удобно сначала создать форму в ви-

де автоформы или с помощью мастера, а затем использовать конструктор.

Конструктор форм. Конструктор форм включает в себя следующие средства:

Бланк формы (рис. 2.48). В нем располагаются элементы, предназначенные для отображения в форме. В основном, именно в бланке формируется внешний вид формы. Форма в **режиме Конструктора** всегда имеет раздел **Область Данных.** Раздел **Заголовок Формы** и **Примечание Формы** могут быть отображены по команде **Вид, Заголовок, Примечание Формы**.

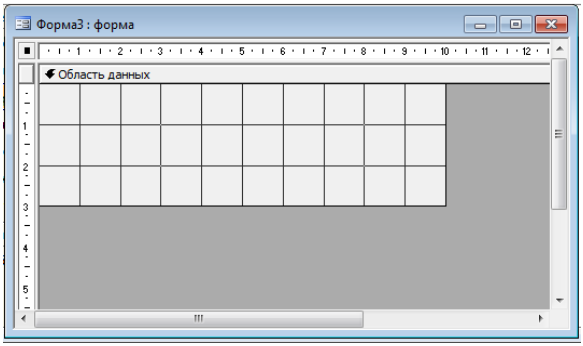


Рис. 2.48. Бланк формы

Панель элементов. Содержит стандартный набор визуальных элементов, которые могут быть помещены в бланк формы.

Выбор объектов - позволяет изменить указатель курсора на инструмент выбора объекта.

Мастера элементов - позволяет включать и отключать мастера по созданию элементов управления.

Надпись - предназначена для вывода на экран не изменяющегося текста, например, заголовков, подписей или пояснений. Надпись относится к **свободным** элементам управления, в которые нельзя вводить данные.

Поле - позволяет создать область для отображения, ввода или изменения данных. В поле можно использовать данные любого типа: текст, числа, дата/время, логические величины и

МЕМО. Поля могут быть как присоединенными, так и свободными. В них можно использовать поля из таблиц или запросов, а также вычисляемые выражения, поэтому такие элементы управления называют **связанными полями**. При создании связанного поля вместе с ним одновременно образуется еще один элемент управления - **присоединенная надпись.**

Группа параметров - позволяет создать область настраиваемого размера для размещения набора флажков, переключателей или выключателей, представляющих набор альтернативных значений.

Выключатель - позволяет создать кнопку, связанную с логическим полем. Элемент может находиться в двух состояниях: **Истина** - кнопка **нажата**, **Ложь** - кнопка **отжата.**

Переключатель - предназначен для создания кнопки. Ее функции аналогичны функциям выключателя. Элемент находится в двух состояниях: **Истина** - кружок **с точкой**, **Ложь** - **пустой** кружок. С кнопкой можно связать команды, например, выполняющие фильтрацию.

Флажок - предназначен для создания флажка связанного с логическим полем. Действуют аналогично переключателям, но в отличие от них, допускают множественный выбор. Элемент может находится в двух состояниях: **Истина** - квадрат **с галочкой**, **Ложь** - **пустой** квадрат**.**

Поле со списком - позволяет создать составной элемент управления, объединяющий поле и раскрывающийся список значений. Для ввода значения, можно ввести значение в поле или выбрать значение в списке.

Список - позволяет создать список, допускающий прокрутку, и предназначенный для выбора значения. Позволяет отображать список значений в форме или отчете. В списках можно также отображать заголовки столбцов.

Кнопка - позволяет создать кнопку, используемую для выполнения набора макрокоманд Access или процедур VBA.

Рисунок - позволяет создать рамку, в которой в форме или отчете выводится неизменяемый рисунок. Поскольку рисунок не является объектом OLE, то после помещения рисунка в форму или отчет не допускается его изменение из Microsoft Access.

Свободная рамка объекта - позволяет создать рамку для отображения в форме или отчете объектов OLE, как правило, набор иллюстраций или диаграмму. Рамка не связана ни с каким полем таблиц базы данных.

Присоединенная рамка объекта - для отображения в форме или отчете объектов OLE, таких как набор иллюстраций или диаграммы. С присоединенной рамкой связано одно из полей таблиц. При переходе от записи к записи в форме или отчете выводятся разные объекты.

Конец страницы - позволяет создать элемент управления, указывающий принтеру начало новой страницы в печатной форме или новой страницы в отчете. Этот элемент управления не появляется в форме или запросе в режиме формы.

Вкладка - позволяет вставить элемент управления **Вкладка** для создания вложенных форм. Страницы элемента управления **Вкладка** могут содержать другие элементы управления.

Подчиненная форма/отчет - предназначена для добавления в основную форму или основной отчет подчиненной формы или подчиненного отчета соответственно. Добавляемые подчиненная форма или подчиненный отчет должны существовать.

Линия - позволяет создать прямую линию, которую можно перемещать и размеры которой можно изменять. Цвет и толщину линии можно изменить с помощью кнопок панели инструментов **Панель форматирования** или окна свойств. Используется для разделения элементов формы или отчета.

Прямоугольник - позволяет создать прямоугольник, который можно перемещать и размеры которого можно изменять. Используется для выделения элементов формы.

Для создания элемента управления: текста, поля, линии, прямоугольника (рамки), кнопки и др.:

1. Щелкните на соответствующей пиктограмме.
2. Укажите курсором мыши (крест с уменьшенным изображением создаваемого элемента) место для создаваемого элемента.

Редактор свойств. Позволяет задавать свойства формы и размещенных в ней элементов (объектов). Для формы и всех элементов определены наборы свойств. Свойства имеют четыре категории Макет, Данные, События, Другие. Мы будем менять только свойства в категории Макет (рис. 2.49).

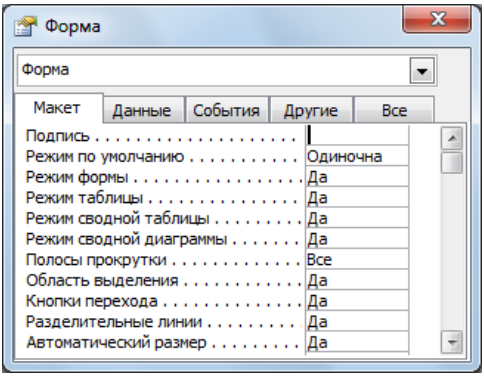


Рис. 2.49. Редактор свойств

Для задания фона, установите текущий объект – **Область данных** и задайте в категории **Макет**, свойство – **Фон**. Для работы с формой перейдите в режим Форма. Для придания окончательного вида, перейдите в режим конструктора и добавьте элемент управления Рисунок. Программа запросит вас указать путь к файлу, где лежит рисунок. Свойство этого элемента управления на вкладке Макет Установка размеров должно быть **Вписать в рамку** (рис. 2.50).

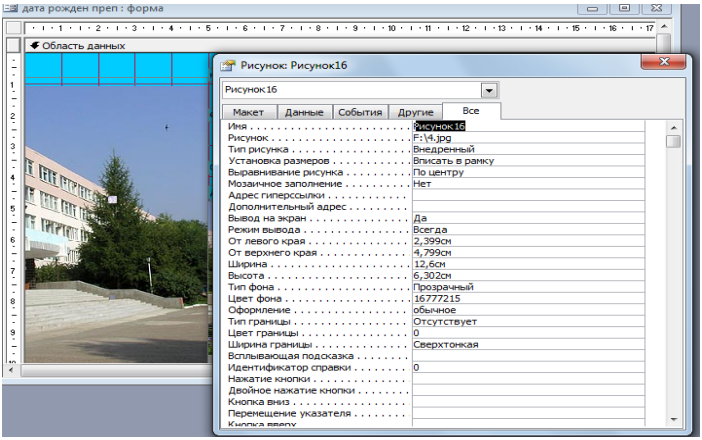


Рис. 2.50. Добавление элемента управления Рисунок

Список полей. Если для формы указан источник данных (таблица или запрос), в данном списке отображаются поля источника, которые можно перенести в бланк формы.

2.6.1 Порядок разработки простых форм

1. В окне базы данных выбрать вкладку **Форма** и нажать кнопку **Создать**.

2. Выбрать в поле **Источник данных** имя таблицы или запроса, на котором будет базироваться форма, а также способ создания формы - мастер форм.

3. В окне **Создание форм** переведите поля, размещаемые на форме из области **доступные поля** в область **выбранные поля** и нажмите **Далее**.

4. Выберите тип формы – в один столбец и нажмите **Далее**.

5. Выберите стиль (фон) формы и нажмите **Далее**. Для формы, выводимой на печать, желательно не задавать темный фон.

6. Задайте имя формы (в соответствие с базовой таблицей или запросом) и нажмите **Готово**.

7. Откройте созданную форму, перейдите в режим конструктора (рис. 2.51) и вручную сделайте изменения - поместить календарь, выберем пункт меню **Вставка, Элемент**. В окне **Вставка** специальных элементов управления в списке Выбор специального элемента – **Элемент календарь**.

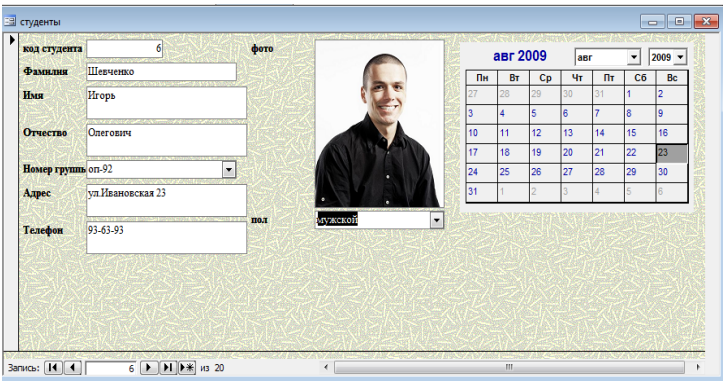


Рис. 2.51 Простая форма в один столбец

2.6.2 Создание сложных форм

Разработаем сложную форму, в которой с таблицей преподаватели была бы связана подчиненная форма Дисциплины.

Для создания сложной формы:

- На вкладке **Формы** щелкните по кнопке **Создать**, выберите Мастер форм и, не выбирая таблицу или запрос, щелкните по кнопке **Ок**;

- В таблице **Преподаватели** выберите необходимые поля. В таблице **Дисциплины** выберите поля Название дисциплины, Семестр, Экзамен; в появившемся окне оставьте предлагаемый вариант формы и щелкните по кнопке **Далее**;

- Оставьте табличный вариант подчиненной формы и щелкните по кнопке **Далее**;

- Выберите стиль оформления формы и щелкните по кнопке **Далее**;

- Введите название формы Преподаватели, щелкните по кнопке **Готово** и просмотрите полученную форму (рис. 2.52).

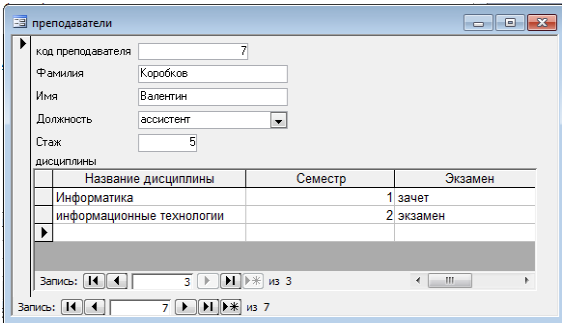


Рис. 2.52. Подчиненная форма

Также подчиненную форму можно создать, используя конструктор и с помощью панели инструментов добавим в бланк формы элемент **Подчиненная форма**. Появившуюся при этом над прямоугольником подчиненной формы надпись можно удалить или сделать невидимой. Теперь зададим свойства подчиненной формы, которые свяжут ее с главной формой и с таблицей – источником данных.

2.6.3 Вставка графических объектов

Для того чтобы вставить в форму диаграмму количества студентов в группах, необходимо:

- Переключиться в режим конструктора, выполнить команду **Вид - Панель элементов**, на этой панели щелкнуть по кнопке <**Аа**>;

- Создать прямоугольник для заголовка диаграммы, ввести надпись Диаграмма оценок и выполнить команду **Вставка - Диаграмма**;

- На свободном месте формы создать прямоугольник для диаграммы, выбрать таблицу **Студенты** и щелкнуть по кнопке **Далее**;

- Выбрать поле **Номер Группы** и щелкнуть по кнопке **Далее**;

- Выбрать вид диаграммы **Гистограмма** (по умолчанию он стоит) и щелкнуть по кнопке **Далее**;

- Вновь щелкнуть по кнопке **Далее**, так как в строках **Поля формы** и **Поля диаграммы** по умолчанию находится **Номер группы**; по необходимости отредактируйте вид осей диаграммы (рис. 2.53).

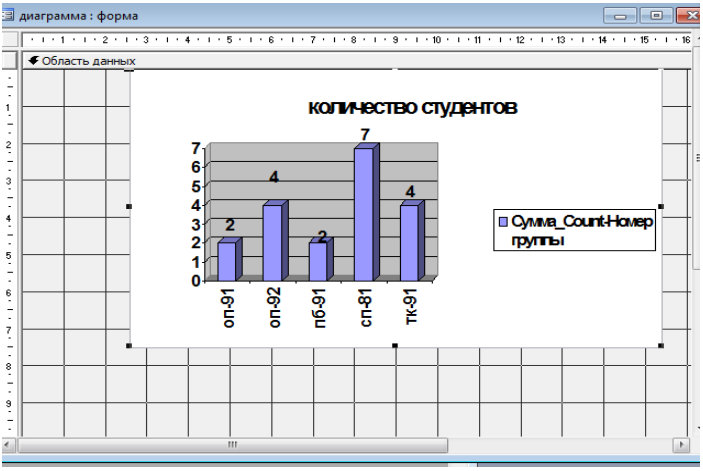


Рис. 2.53. Режим конструктора - Диаграмма

2.6.4 Вычисления в форме

Чтобы произвести вычисления на основе данных в каждой записи, форме необходимо добавить в форму свободное поле (не связанное ни с каким полем базы данных) и в качестве источника данных задать выражении.

В вычисляемом поле можно отобразить результат вычисления всех правильно заданных выражений Access (рис. 2.54). Выражение должно начинаться со "=" (знака равенства) и может использовать функции Access. Для создания вычисляемого поля, отображающего текущие дату и время: Для отображения в поле текущих даты и времени из часов компьютера введите в поле: **=Now()**. В режиме Конструктора в этом поле выводится вычисляемая формула; дата и время отображаются в этом поле только в режиме формы. Для вычисления даты рождения (поле Возраст) создать формулу:

=Year(Date())-Year([Дата рождения])

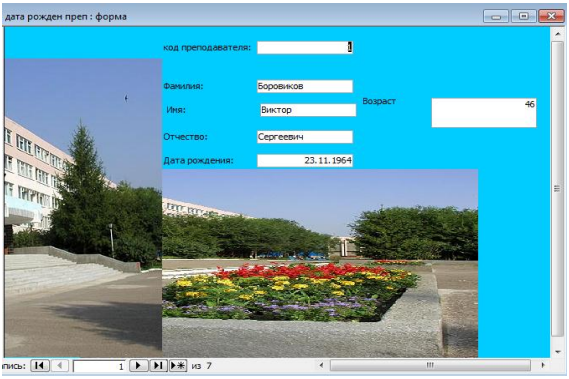


Рис. 2.54. Режим формы - Вычисления в форме

2.6.5 Использование элементов управления

Элемент управления **Кнопка** является очень важным элементом формы, поскольку именно с кнопками связаны различные действия, выполняемые пользователем в приложении (сохранение введенных данных, вызов другой формы, вывод на

печать документа и т. д.). Создать кнопку намного удобнее с помощью **Мастера кнопок**. Например, рассмотрим процесс создания кнопки, которая будет открывать новую форму:

Создайте в режиме Конструктора пустую форму. Выберите на панели элементов элемент управления Кнопка. В появившемся диалоговом окне вы увидите два списка: левый список содержит категории действий, а правый - сами действия. Выберите в **списке Категории** значение **Работа с формой** (рис. 2.55).

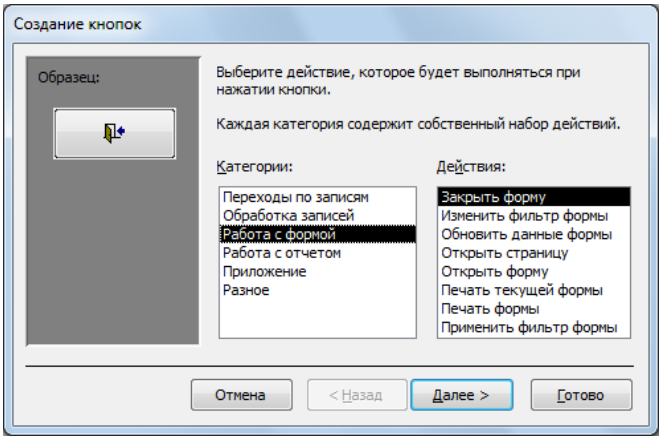


Рис. 2.55 Диалоговое окно Мастера кнопок

В поле Действия при этом появится список действий, относящихся к этой категории. Выберите значение **Открыть форму** и нажмите кнопку Далее. В следующем окне мастера требуется выбрать форму, которую нужно открывать с помощью кнопки. Выберите одну из форм, например **Студенты**.

В следующем окне предлагается выбрать рисунок, который вы хотите поместить на кнопку, или задать надпись. В последнем окне **Мастера кнопок** требуется ввести имя кнопки (оно может не совпадать с надписью на кнопке). Нажмите кнопку **Готово**.

Можно создавать любые кнопки - перехода, добавления записи, выход из приложения, печать (рис. 2.56).

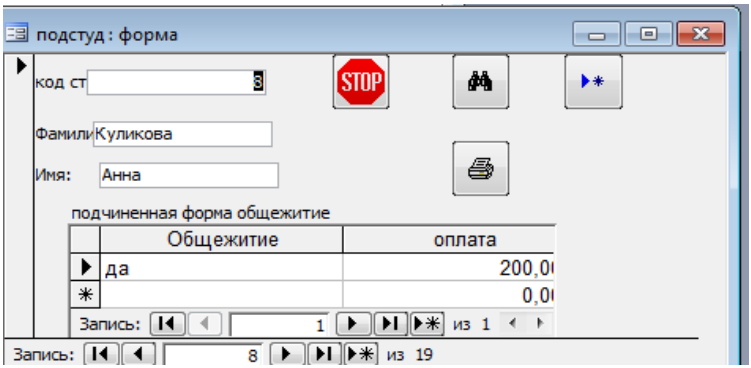


Рис. 2.56 Создание формы с различными кнопками

2.6.6 Создание кнопочных форм

Для создания, изменения и удаления кнопочных форм служит диспетчер кнопочных форм.

В меню **Сервис** выберите команду **Служебные программы**, а затем **Диспетчер кнопочных форм**.

1. Если выводится запрос на подтверждение создания кнопочной формы, нажмите кнопку **Да**.
2. Нажмите кнопку **Создать**.
3. Введите имя новой кнопочной формы и нажмите кнопку **Ok**.

Имя новой кнопочной формы добавляется в поле **Страницы кнопочной формы**.

4. Выберите имя новой кнопочной формы и нажмите кнопку **Изменить**.
5. Нажмите кнопку **Создать**.
6. В поле **Текст** введите текст для первой кнопки кнопочной формы, а затем выберите для нее команду в поле **Команда**. Например, введите текст **Формы**, а затем выберите в поле **Команда** команду **Открыть форму** для изменения (рис. 2.57).

Примечание. Для создания кнопочной формы, которая открывает другие кнопочные формы, выберите в поле ***Команда*** *команду* ***Перейти*** *к кнопочной форме, а затем укажите кнопочную форму, к которой надо перейти.*

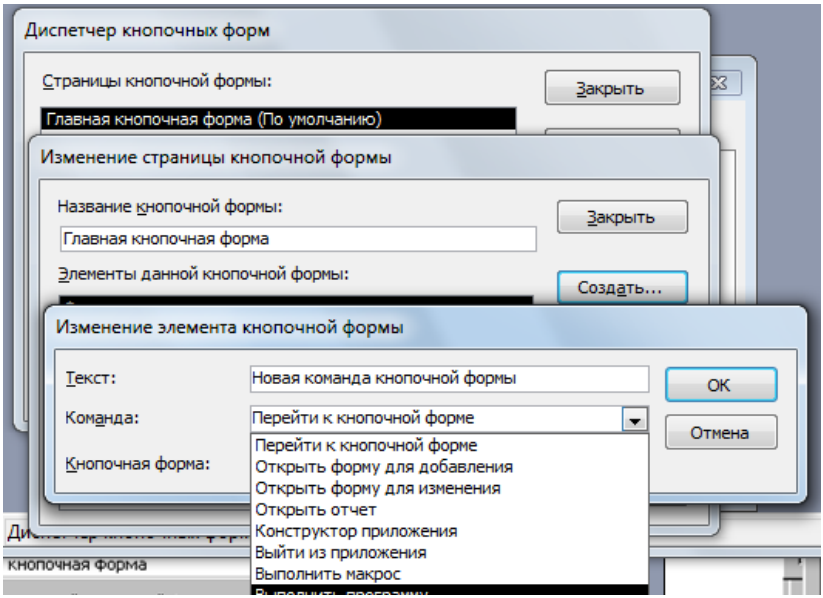


Рис. 2.57. Создание нового элемента кнопочной формы

Примечание. Чтобы изменить или удалить какую-либо из созданных кнопок, выберите ее имя в списке ***Элементы*** *данной кнопочной формы и нажмите кнопку* ***Изменить*** *или* ***Удалить****. Если требуется изменить порядок элементов кнопочной формы, выберите элемент в списке и воспользуйтесь кнопками* ***Вверх*** *или* ***Вниз****.*

Для большинства выбранных команд под **полем Команда** открывается новое поле со списком. При необходимости, выберите нужный элемент в этом поле. Например, если была выбрана команда **Открыть форму** для изменения, выберите в поле **Форма имя** нужной формы, например **Студенты**, и нажмите кнопку **Ok**.

7. Закончив создание кнопочной формы, нажмите кнопку **Закрыть**.

8. Для открытия запроса, необходимо предварительно создать макрос.

После того, как построены все основные части приложения, можно задать режим автоматического запуска базы данных. Удобно для этих целей использовать параметры запуска и задать начальную форму приложения. Для этого необходимо переключиться в окно базы данных и выбрать **команду Сервис - Параметры запуска.** В окне диалога **Параметры запуска Вывод формы/страницы** позволяет выбрать форму, которая будет выводиться на экран при открытии базы данных.

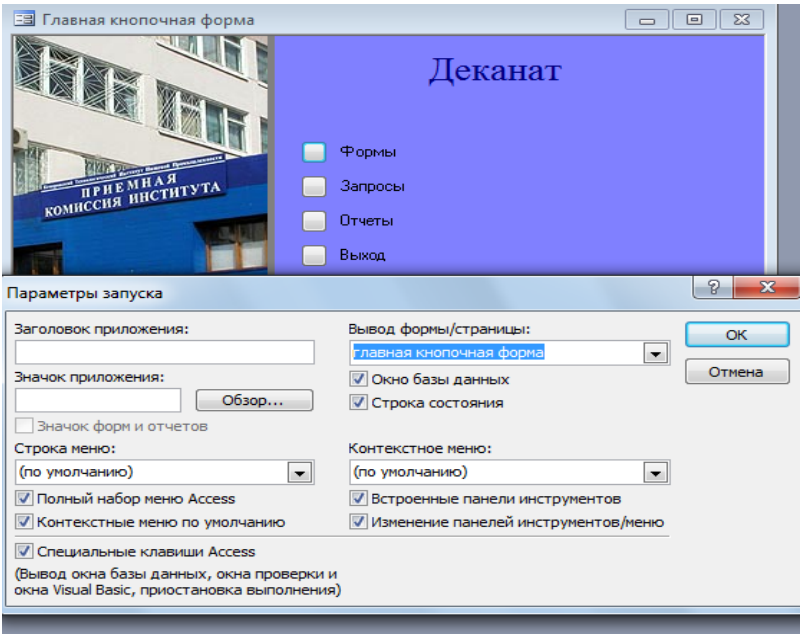


Рис. 2.58. Создание параметров запуска

2.7 Отчеты

Отчет - это гибкое и эффективное средство для организации просмотра и распечатки итоговой информации. В отчете можно получить результаты сложных расчетов, статистических сравнений, а также поместить в него рисунки и диаграммы.

Способы построения отчетов и форм очень близки. Различия между ними связаны с тем, что формы предназначены для просмотра и ввода данных, а отчеты – для просмотра и вывода на печать. В отчетах присутствуют некоторые дополнительные возможности по форматированию данных в документах. Рассмотрим процесс построения отчета с помощью конструктора на примере: составим отчет, выводящий список преподавателей в алфавитном порядке.

Запустим конструктор отчетов для создания нового отчета. В бланке отчета присутствуют области, которых не было в формах: верхний и нижний колонтитул. Колонтитул выводится на каждой печатной странице вверху или внизу листа. Колонтитулы можно разрешить или запретить с помощью контекстного меню бланка отчета или в меню **Вид**. Также можно разрешить или запретить использования заголовка и примечания отчета, выводимых в начале и в конце отчета соответственно.

Процедура формирования **Отчета** с помощью **Мастера отчетов** очень похожа на создание **Формы** с помощью **Мастера форм**, которую вы изучили.

Для создания **Отчета** выполните следующие действия:

1. Выберите закладку **Отчет** в окне базы данных. Нажмите кнопку **Создать.**

2. В окне **Новый отчет** выберите режим создания отчета – **Мастер отчетов**, в качестве источника данных выберите из списка таблицу **Преподаватели,** чтоб получить ведомость на выдачу зарплаты, нажмите кнопку **OK** для перехода к следующему шагу создания отчета.

3. После определения полей, включаемых в отчет, система предлагает выбрать уровни группировки. В общем случае, для выделения уровней группировки надо позиционироваться на соответствующее поле, которое будет являться полем, по которому производиться группировка, и нажать на кнопку со стрел-

кой. Если предполагается несколько уровней группировки, то поля должны выбираться в порядке старшинства группировки (рис. 2.59).

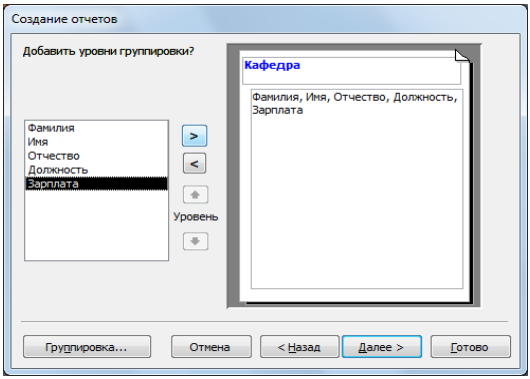


Рис. 2.59. Окно создание отчетов – определение уровней группировки

Следующий экран позволяет задать порядок сортировки и, если необходимо, вычисление итогов (рис. 2.60). Мы выбрали сортировку по трем полям (**Фамилия, Имя, Отчество**) и получение суммарных итогов по полю **Зарплата**.

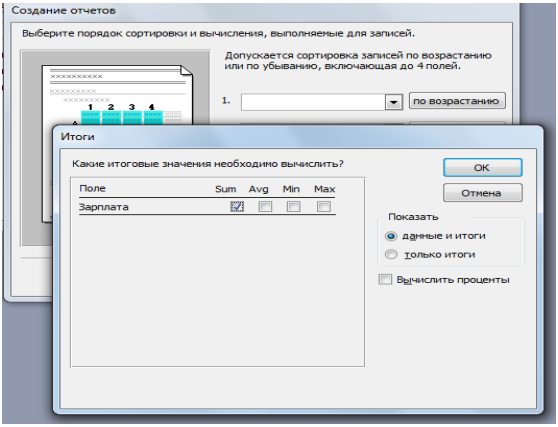


Рис. 2.60. Создание итогов

В результате выполненных действий будет сформирован отчет. Он будет выведен на экран в режиме предварительного просмотра (рис 2.61).

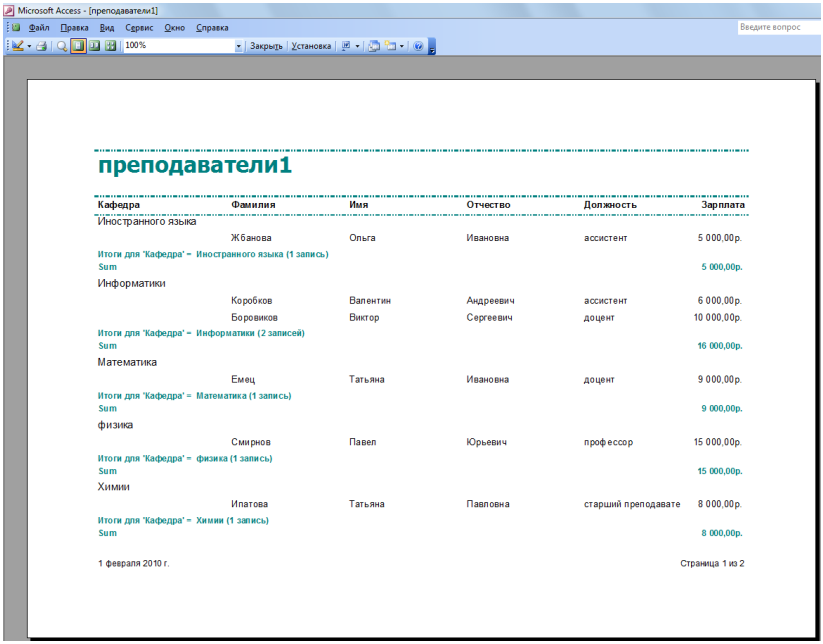


Рис. 2.61. Вид полученного отчета

Если требуется изменить свойства нескольких полей, то их выделяют при нажатой левой кнопке мыши, а затем указывают необходимые свойства.

2.7.1 Создание сложных отчетов

Создать отчет, включающий подчиненные, можно выполнив следующую последовательность действий:

1. Откройте отчет, который должен быть главным отчетом, в режиме конструктора.

2. Убедитесь, что кнопка **Мастера** на панели элементов нажата.

3. Нажмите кнопку **Подчиненная форма/отчет** на панели элементов.

4. Установите указатель в отчете на том месте, куда требуется поместить подчиненный отчет, и нажмите кнопку мыши. В результате появиться диалоговое окно **Мастер подчиненных отчетов**.

5. В появившемся окне выберите таблицу/запрос, являющуюся источником для подчиненного отчета или заранее созданный отчет, который должен быть включен в создаваемый отчет в качестве подчиненного. После нажатия кнопки **Готово** элемент управления **Подчиненная форма/отчет** будет вставлен в главный отчет (рис. 2.62).

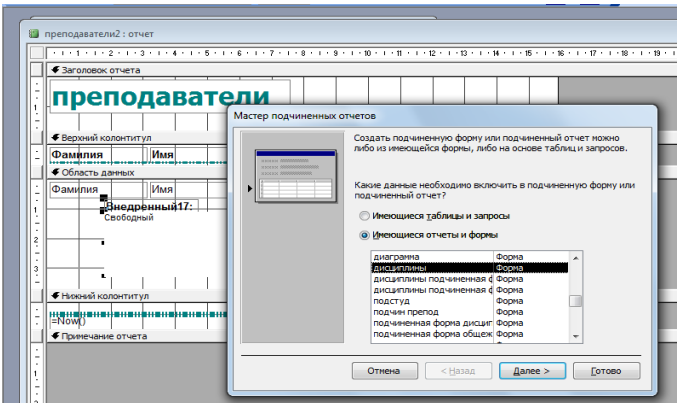


Рис. 2.62. Создание подчиненного отчета

2.7.2 Вычисления в отчете

В отчетах (также как и в запросах, формах) можно использовать вычисляемые поля.

Так как вычисляемое поле, как следует из его названия, является полем, то для его создания в отчет следует включить элемент-поле (при этом в области данных создается поле, внутри которого вместо имени поля таблицы указано слово **Свободный**). Чтобы ввести выражение для вычисления значения поля

следует создать и выделить это поле, щелчком правой кнопки мыши вывести на экран контекстно-зависимое меню, выбрать в нем строку **Свойства**, в появившемся окне свойств поля перейти на позицию **Данные** и в данной строке ввести требуемое выражение. Оно может вводиться вручную либо строиться с использованием построителя выражений.

Предположим, что мы хотим в нашу ведомость ввести графу «Подоходный налог» (для простоты будем считать, что все преподаватели платят налог в размере 13%). Выражение, записанное в строку **Данные** окна **Свойства полей**, будет иметь следующий вид:

*= [Зарплата] * 0,13.*

Для задания сортировки и группировки следует воспользоваться соответствующей кнопкой .

Зададим свойство объекта отчета: Данные - **Источник записей** = **Преподаватели**. Разместим в области данных элемент **Поле** и зададим для него следующее свойство:

поле **Данные** = «=[Фамилия]+" "+[Имя]+" "+[Отчество]». Сгруппируем преподавателей по первой букве фамилии и перед каждой группой укажем эту букву и выполним сортировку по фамилии.

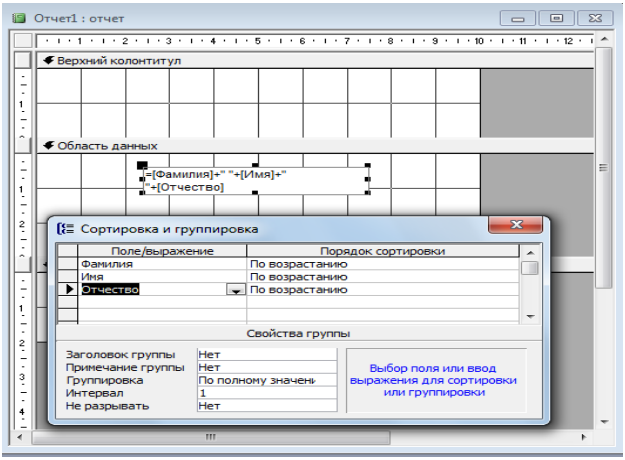


Рис. 2.63. Окно конструктора Отчетов с группировкой

2.7.3 Параметрические отчеты

Иногда в документы необходимо вставлять данные, которые отсутствуют в таблицах БД и не могут быть вычислены. Одним из способов осуществления этого является использование «параметрического» отчета. Примером такой ситуации является необходимость печатать в заголовке Ведомости на выдачу зарплаты название месяца. Это можно сделать следующим способом: вставить в соответствующее место отчета новое поле, после этого войти в свойства этого элемента и в строке данные ввести слово «месяц» (рис. 2.64), а в строке подпись – «**За**».

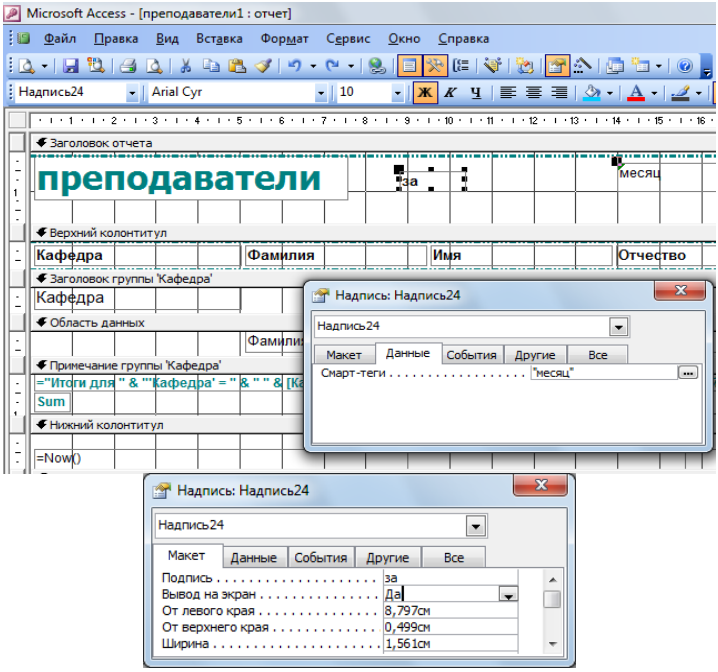


Рис. 2.64. Задание подписи поля в свойствах элемента

Кроме отчетов табличной формы, о которых шла речь выше, в практике используются и другие формы отчетов, и Access позволяет легко их создавать.

2.8 Макросы

Макрос представляет собой совокупность определенных команд управления (макрокоманд), используемых для автоматизации часто повторяющихся действий.

В окне **База Данных** щелкните на вкладке **Макрос**. В открывшемся окне щелкните на кнопке **Создать**. Появится окно **Макрос**. В столбце **Макрокоманда** щелкните на кнопке выпадающего списка и выберите из него действие, которое должно быть автоматизировано (например, открытие запросов).

В области **Аргументы макрокоманд** укажите необходимые для этой макрокоманды аргументы (например, имя запроса, отображаемый режим и т.п.). Аргументы макрокоманд указывают, к какому объекту применить макрокоманду, и задают условия выполнения действий. Большинство может быть также выбрано из выпадающего списка. Если макрос будет содержать более чем одну макрокоманду, то в столбце **Макрокоманда** укажите эти действия в необходимом порядке. В столбце **Примечание** можно описать назначение каждой макрокоманды. Сохраните макрос.

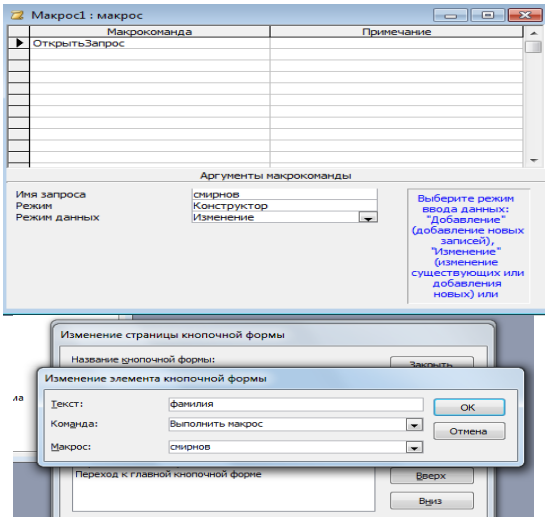


Рис. 2.65. Создание макроса на открытие запроса

Существует несколько методов запуска макросов на выполнение:

- **Из окна База Данных**: дважды щелкните на имени макроса или выберите кнопку **Запуск**;

- **Из меню**: в меню **Сервис** выберите команду **Макрос / Запуск макроса** и из раскрывающегося списка выберите нужный макрос (или введите его имя);

- **Из окна макрос**: щелкните на пиктограмме с восклицательным знаком (**Запуск**) или в меню **Макрос** выберите команду **Запуск**.

Существует несколько методов запуска макросов на выполнение:

- **Из окна База Данных**: дважды щелкните на имени макроса или выберите кнопку **Запуск**;

-**Из меню**: в меню **Сервис** выберите команду **Макрос, Запуск макроса** и из раскрывающегося списка выберите нужный макрос (или введите его имя);

- **Из окна макрос**: щелкните на пиктограмме с восклицательным знаком (**Запуск**) или в меню **Макрос** выберите команду **Запуск**.

2.9 Защита автоматизированной системы

Простейшим способом защиты является установка пароля для открытия базы данных (.mdb). После установки пароля при каждом открытии базы данных будет появляться диалоговое окно, в которое требуется ввести пароль. Только те пользователи, которые введут правильный пароль, смогут открыть базу данных. Этот способ достаточно надежен (Microsoft Access шифрует пароль, поэтому к нему нет доступа при непосредственном чтении файла базы данных), но он действует только при открытии базы данных.

Для установки пароля необходимо выполнить следующие действия:

1. В меню **Сервис** выбрать **Защита, Задать пароль базы данных**;

2. В открывшемся диалоговом окне **Задание пароля базы данных** задать **Пароль** и **Подтверждение.**

2.10 Язык запросов SQL

Аббревиатура SQL означает Structured Query Language (структурированный язык запросов). Синтаксис SQL разрабатывался для удобства формирования запросов «близко к естественному английскому». Предполагалось, что его смогут использовать рядовые пользователи баз данных.

Составленный запрос на языке SQL передается клиентом серверу, на сервере производится его выполнение, после чего клиенту передается сформированный результирующий набор. В системах «клиент-сервер» SQL чрезвычайно удобен, поскольку позволяет выполнить достаточно сложную обработку на сервере, без передачи промежуточных данных клиенту.

В Access при обращении к БД также применяется язык. Любой запрос, построенный с помощью мастера или конструктора, имеет соответствующее представление на языке SQL. Конструктор – лишь визуальное средство для создания запросов. В Access имеется возможность редактировать запросы непосредственно в режиме SQL. Для переключения режимов отображения запросов используется кнопка «**Вид**» панели инструментов.

Чаще всего возникает задача построения запросов на извлечение данных. Для этих целей используется SQL-оператор SELECT.

Общая форма оператора SELECT

SELECT [**DISTINCT**] список_выражений|*
[**INTO** новая_таблица]
[**FROM** объединение_источников]
[**WHERE** условие]
[**GROUP BY** список_столбцов]
[**HAVING** условие]
[**UNION** [**ALL**] **SELECT** …]
[**ORDER BY** список_столбцов]

Простейшая форма оператора SELECT

SELECT <список полей>
FROM <список источников>**;**

Пример:

SELECT *Фамилия, Имя, Отчество*
FROM *Преподаватели****;***

Результат работы этого запроса (рис. 2.66-2.67).

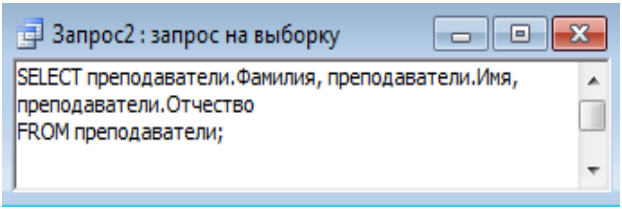


Рис.2.66. Окно конструктора SQL

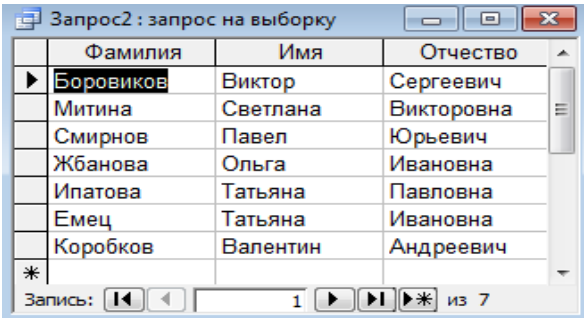


Рис. 2.67. Окно таблицы

Например, чтоб узнать дату рождения студента Иванова, необходимо задать запрос (рис. 2.68):

SELECT студенты.Фамилия, студенты.[дата рождения]
FROM студенты
WHERE (((студенты.Фамилия)="иванов"));

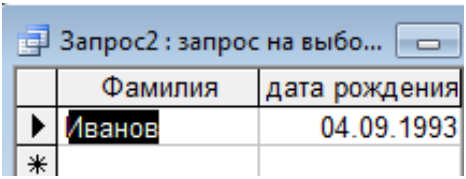


Рис.2.68. Окно таблицы Студенты
для запроса по дате рождения

Чтоб получить информацию о всех девушках группы Пб-91, сформируем запрос (рис. 2.69):

*SELECT **

FROM студенты

WHERE (((студенты.пол) Like "ж") AND ((студенты.[Номер группы]) Like "Пб*"));*

Запрос3 : запрос на выборку

код студента	Фамилия	Имя	Отчество	Номер группы	Адрес	Телефон	фото	пол	дата рождения
8	Куликова	Анна	Сергеевна	пб-91	ул.Ворошилова 25-604	90-24-45		женский	24.03.1993
0									

Запись: 1 из 1

Рис. 2.69. Окно таблицы Студенты о всех девушках группы Пб-91

Параметрические запросы

В параметрических запросах всегда использую значение параметра, которое запрашивается во время ввода (рис. 2.70).

*SELECT **

FROM студенты

WHERE (((студенты.[Номер группы])=[укажите номер группы]));

Запрос3 : запрос на выборку

код студента	Фамилия	Имя	Отчество	Номер группы	Адрес
1	Патрикеев	Олег	Борисович	оп-91	пр.Ленина 1-3
2	Белых	Ярослав	Игоревич	оп-91	ул.Савхозная 4
0					

Запись: 1 из 2

Рис. 2.70. Окно таблиц параметрического запроса

Контрольные вопросы

1. Что в Access называется базой данных?
2. К какому классу относится СУБД Access?
3. Каковы особенности реляционной модели данных?
4. Как создать новую базу данных в Access?
5. Как добавить новый объект в существующую базу данных?
6. Какие способы создания таблиц Вы знаете?
7. Какие типы полей допустимы в Access? Каковы особенности работы с полями каждого из этих типов?
8. Какие преимущества дает использование полей подстановки?
9. Что называется ключом таблицы? Какие разновидности ключей вы знаете?
10. Какими способами можно создать ключ?
11. Каким образом можно создавать запросы на языке в Access?
12. Какие еще языки запросов можно использовать в Access?
13. Что может служить источником данных для запроса?
14. Какие разновидности запросов Вы знаете?
15. Какие запросы называются «сложными»?
16. Как задаются условия «И» и «ИЛИ» в запросах?
17. Что собой представляют перекрестные запросы? Как и для чего они создаются?
18. Что собой представляют параметрические запросы? Как и для чего они создаются?
19. Как вводятся в запрос вычисляемые поля?
20. Как можно получать итоговые значения в запросах?
21. Какие разновидности корректирующих запросов Вы знаете? Как их задать?
22. Какие разновидности экранных форм вы знаете?
23. Каково назначение экранных форм?
24. В каких режимах можно работать с экранной формой? Каково назначение каждого из этих режимов?
25. Какими способами можно создавать экранную форму?

26. Каким образом можно менять размещение элементов на экране?
27. Как можно менять размер элемента управления?
28. Что может являться источником данных для экранной формы?
29. Каковы особенности использования запроса со «*» в качестве источника для формы?
30. Как можно включить в отчет вычисляемое поле?
31. Как можно включить в отчет рисунок?
32. Как можно запретить корректировку таблицы при ее просмотре с помощью формы?
33. Как можно создать форму для ввода данных?
34. Каково назначение отчетов?
35. Каким способом могут создаваться новые отчеты в Access?
36. Какие разновидности отчетов Вы знаете?
37. Как вводятся в отчеты вычисляемые поля?
38. Что может являться источником данных для отчетов?
39. Что такое «параметрический отчет»? Как его можно создать?
40. Как можно скорректировать существующий отчет?
41. Как можно задавать группировку данных в отчете?
42. Сколько уровней группировки позволяет создать Access в отчете?
43. Как можно сортировать данные в отчете?
44. Как можно обеспечить нумерацию строк в отчете?
45. Что такое сложные отчеты и как их можно создавать?

ГЛАВА 3. СУБД FOXPRO

Существующие СУБД поддерживают три технологии разработки программ:

– ручное кодирование программ (Clipper, FoxPro, Paradox);

– создание текстов приложений с помощью генераторов (FoxApp в FoxPro, Personal Programmer в Paradox);

– автоматическая генерация готового приложения методами визуального программирования (Delphi, Access, Paradox for Windows).

При ручном кодировании программисты вручную набирают программы, после чего выполняют их отладку. Использование генераторов упрощает разработку приложений, поскольку при этом можно получать программный код без ручного набора. Генераторы приложений облегчают разработку основных элементов приложений (меню, экранных форм, запросов) но зачастую не могут полностью исключить ручное кодирование.

Средства визуального программирования приложений являются дальнейшим развитием идеи использования генераторов приложений. Приложение при этом строится из готовых «строительных блоков» с помощью удобной интегрированной среды. При необходимости разработчик легко может вставить в приложение свой код. Интегрированная среда, как правило, предоставляет мощные средства создания, отладки и модификации приложений. Использование средств визуального программирования позволяет в кратчайшие сроки создавать более надежные, привлекательные и эффективные приложения по сравнению с приложениями, полученными первыми двумя способами.

Разработанное приложение обычно состоит из одного или нескольких файлов операционной системы. Если основным файлом приложения является исполняемый файл (например, exe-файл), то это приложение, скорее всего, является независимым приложением, которое выполняется автономно от среды СУБД. Получение независимого приложения на практике осуществляется путем компиляции исходных текстов программ, полученных различными способами: путем набора текста вруч-

ную, а также полученных с помощью генератора приложения или среды визуального программирования. Независимые приложения позволяют получать, например, СУБД FoxPro и система визуального программирования Delphi.

Во многих случаях приложение не может исполняться без среды СУБД. Выполнение приложения состоит в том, что СУБД анализирует содержимое файлов приложения и автоматически строит необходимые исполняемые машинные команды. Другими словами, приложение выполняется методом интерпретации. Режим интерпретации реализован во многих современных СУБД, например FoxPro 2.6, Access, Visual FoxPro и Paradox. Важным достоинством применения интерпретируемых приложений является легкость их модификации. Если готовая программа подвергается частым изменениям, то для их внесения нужна инструментальная система, то есть СУБД или аналогичная среда. Для интерпретируемых приложений такой инструмент всегда под рукой, что очень удобно.

При выборе средств для разработки приложения следует учитывать три основных фактора: ресурсы компьютера, особенности приложения (потребность в модификации функций программы, время на разработку, необходимость контроля доступа и поддержание целостности информации) и цель разработки (отчуждаемый программный продукт или система автоматизации своей повседневной деятельности).

При использовании компьютера со слабыми характеристиками лучше остановить свой выбор на системе со средствами разработки независимых приложений.

Для пользователя, имеющего современный компьютер и планирующего создать несложное приложение, по всей видимости, больше подойдет система интерпретирующего типа - СУБД FoxPro - достаточно мощный интерпретатор, который имеет высокоуровневые средства, удобен для разработки и отладки, позволяет быстро выполнить разработку и обеспечивает удобное сопровождение и модификацию приложения.

Кроме того, разработанное приложение может быть конвертировано на платформу Windows. Встроенный в среду СУБД FoxPro язык SQL запросов может служить стандартом для изучения и работы с ним в любых базах данных.

3.1 Характеристики FoxPro 2.6

Полнофункциональная реляционная многопользовательская СУБД с удобным пользовательским интерфейсом имеет мощный язык управления запросами SQL, поддерживает разнообразные всплывающие и многоуровневые меню, работу с окнами и мышью имеет тип транслятора – интерпретатор, но в то же время имеет возможность формирования загрузочных exe-модулей, которые далее могут запускаться самостоятельно без поддержки их "родительской" СУБД. В ней реализованы функции низкоуровневого доступа к файлам, управления цветами, настройками принтера. Данные могут быть представлены в виде, похожем на электронные таблицы. Система также обладает средствами быстрой генерации экранов, отчетов, меню и запросов SQL, устойчиво работает в сети, обладает исключительно высокими скоростными характеристиками. FoxPro реализован в средах Macintoch, Unix и MS Windows, сохранив полную преемственность по языку и формату данных, так что программы могут быть легко перенесены на другую платформу. Информация представляется в виде таблиц. Каждая конкретная информация называется поле. Совокупность полей, относящихся к одному объекту, называется запись. Совокупность записей объединяется в файл. Максимальное число записей в файле-1 млрд. Максимальная величина записи - 4000 байт. Максимальное число полей в файле базы данных - 255.

3.2 Команды FoxPro 2.6

3.2.1 Обозначения и структура команд

Существуют 2 вида команд:

- Команды установки
- Команды работы с данными

Команды установки, как правило, не влекут каких-то немедленных действий, а определяют условия работы других команд, т.е. устанавливают параметры операционной среды FoxPro. Установки выполняются командами SET.

SET<параметр команды> TO <значение параметра>

или

SET<параметр команды> OFF/ON

ON - включен *OFF –выключен*

Структура команды работы с данными:

НАЗВАНИЕ [<границы>] [<список выражений>]
[FOR < условие>][WHILE<условие>]

Опции:

НАЗВАНИЕ - имя команды, возможно сокращение до 4 букв.

<границы> область действия команды, возможно одно из следующих значений:

ALL - все записи,
REST - от текущей записи до конца БД,
NEXT <N.> - от текущей записи следующие N,
RECORD <N> - запись с номером N,

<список выражений> - перечень полей, арифметических действий и функций.

FOR< условие> - выполнение команды только для записей, отвечающих условию.

WHILE <условие> - выполнение команды до тех пор, пока не перестанет выполняться условие. При встрече первой записи, в которой WHILE - условие не удовлетворяется, выполнение команды прекращается, несмотря на то, что нижняя граница еще не достигнута и далее, возможно, имеются записи с нужными свойствами. Поэтому область применения WHILE-условия - это файлы, упорядоченные по полю, в котором анализируется условие. Если ранее каким-либо образом была найдена первая запись, удовлетворяющая условию, то все остальные такие записи находятся ниже рядом. В этом случае использование команд с WHILE-условием предпочтительнее, так как по достижении последней нужной записи выполнение команды прекращается, а в случае FOR-условия поиск записей был бы бессмысленно продолжен до достижения нижней границы действия команды.

НАЗВАНИЕ всегда первое, порядок появления остальных - произвольный, Имя команды можно записывать как прописными, так и строчными буквами, названия полей должны быть тождественны названиям полей в БД. Команда может иметь длину до 2048 символов, если команда не помещается в одной сроке, то ее можно перенести на другую строку, в конце первой строки надо поставить точку с запятой.

Константы различных типов данных отображаются в командах следующим образом:

Символьные константы выделяются апострофами, кавычками или квадратными скобками, например, 'Петров А.'. «Петров» ‘ Петров‘ [Петров]. Чаще всего пользуются апострофами. В случае, если сами символы-ограничители являются элементами данных, вся строка должна быть заключена в другие разрешенные ограничители, например [Кинотеатр "Ударник"].
Дата заключается в фигурные скобки. {01.01.2007}
Логические данные обрамляются точками или пробелами. .T., .t., .F., .f.

Знаки операций:

1. Математические (перечислены в порядке их приоритетов):
   - ** или ^ -возведение в степень
   - * -умножение, /-деление, %- остаток от деления
   - +- сложение, - -вычитание
2. Логические (перечислены в порядке их приоритетов):
   - NOT не
   - AND и
   - OR или
3. Отношения
   - > - больше
   - < - меньше
   - = - равно
   - # - не равно
   - <= - не больше
   - >= - не меньше

Знаки отношения применимы к числам, датам, символьным, причем, если сравниваются символьные разной длины, то сравнение выполняется по длине второго выражения.
Таким образом, считаются истинными (.T.) отношения вида

'ПЕТРОВ А.'='ПЕТРОВ А.' и 'ПЕТРОВ А.'='ПЕТ',

но ложными (.F.) 'ПЕТ'='ПЕТРОВ А.' и 'ПЕТРОВ А.'='ПЕТ

Кроме того, есть операции отношения только для символьных переменных: $ - операция истинна, когда первое выражение входит во второе, == - операция истинна, когда первое выражение полностью тождественно второму.

4. Операции сцепления
   - "+" - соединение двух или более строк в одну.

– "-" - то же, но пробелы в конце строки, предшествующей знаку "-", помещаются в конец итоговой строки.

В выражениях все операции выполняются слева направо с учетом их приоритетов и круглых скобок.

3.2.2 Команды установки

SET BELL ON / OFF

SET BELL TO *[<частота>, <длительность>]*

Два варианта команды. Первый определяет включение предупреждающего звукового сигнала при вводе неверного значения поля или при достижении конца области ввода. Второй устанавливает параметры этого сигнала. Диапазон частоты от 19 до 10000 герц, диапазон длительности от 2 до 19 сек, значения по умолчанию - 512 герц, 2 сек. Соответственно.

SET DEFAULT TO *[<вырC>]*

Перенаправление вывода на другое устройство или каталог.

Пример:

set default to a - все файлы будут записываться на диск A.

SET TALK ON/OFF

Определяет вывод/подавление сообщений о результатах выполнения работы. По умолчанию установлено ON.

SET STATUS ON/OFF

Устанавливает на экране статус-строку. В этой строке будет содержаться имя выполняемой программы, активный диск, имя открытой базы, номер текущей записи, общее число записей в базе данных (эти параметры отображаются в виде дроби как числитель и знаменатель), признак пометки текущей записи к удалению (слово Del), положение клавиш Num Lock и Caps Lock (Num и Caps)

SET HEADING ON/OFF

Определяет вывод/подавление вывода заголовков колонок над каждым из полей в командах. Если установлено ON - заголовки полей выводятся, в противном случае – нет. По умолчанию установлено ON.

SET CONFIRM ON/OFF

Включение / отключение автоматического перехода курсора на следующее поле. По умолчанию установлено ON

SET FILTER TO [<выpL>]

Позволяет выводить и обрабатывать только те записи файла базы данных, которые удовлетворяют заданному условию L.

SET CENTURY ON / OFF

Позволяет определить формат вывода года в датах. Если установлено ON, год при выводе даты отображается четырьмя цифрами, дата выводится длиной 10 символов. Если установлено OFF, год отображается двумя цифрами, дата выводится длиной 8 символов. При этом подразумевается год 20-го века. По умолчанию OFF.

SET DATE *AMERICAN*
BRITISH
FRENCH
GERMAN
ITALIAN

По умолчанию SET DATE TO AMERICAN.

Тип даты	Формат
American	мм/дд/гг
British/French	дд/мм/гг
German	дд.мм.гг
Italian	дд-мм-гг

3.2.3 Команды работы с данными

Создание файла базы данных включает 2 этапа:

– создание структуры
– заполнение его данными.

3.2.3.1 Создание структуры

CREATE *<имя файла>* - создание структуры файла, имя файла базы данных задается самостоятельно, длиной не более 8 символов, расширение DBF набирать не обязательно. По этой команде предоставляется форма, в которой надо указать имя поля (длина имени не более 10 символов, состоит из латинских букв, цифр и знака подчеркивания, начинается с буквы, пробелов быть не должно), тип поля (выбирается нажатием первой буквы типа (C, N, F, D, L, M) или клавишей Space), длину поля (для числовых указать 2 параметра – общую длину поля и число дробных позиций). Возможно удаление, дополнение, изменение порядка расположения полей с помощью кнопок Insert и Delete. Файл автоматически сохраняется в активной директории комбинацией клавиш CTRL +W.

Типы полей:

Символьные поля - это буквы, цифры и другие знаки, длина поля не более 254 б, выравниваются по правой границе.

Числовые поля - это знаки «+» , «-» , цифры и десятичная точка, длина поля не более 19 б, что позволяет обеспечивать 15 значащих цифр, выравниваются по левой границе. Для числовых полей необходимо задавать 2 размера - общее число позиций, отводимых под число и число цифр после точки.

Поля дат имеют стандартную длину, допускают только цифры, причем только 8. Разрешенные даты в диапазоне от 1 янв 100 года до 12 дек 9999. При вводе дат осуществляется контроль ввода, если дата бессмысленная - компьютер подаёт звуковой сигнал. FOXPRO даты может сравнивать между собой, потому что введенную дату он переводит в число, означающее- сколько дней прошло с 1 янв 0001 года.

Логические поля могут иметь только 2 значения TRUE или FALSE, длина поля 1 б.

Текстовые поля или их еще называют поля MEMO- это любые символы, длина поля не ограничена, но они хранятся в отдельном файле, имеющем то же самое имя, но расширение FPT, а в вашем файле будет храниться только адрес, по которому его можно найти. MEMO поля удобно использовать для информации, имеющей непредсказуемую длину. При заполнении MEMO поля загру-

жается редактор текстов комбинацией клавиш CTRL+HOME, набирается текст, сохраняется комбинацией клавиш CTRL+W.

Действие команды рассмотрим на примере: Создадим файл базы данных, который содержит следующие данные

Табельный номер работника	TAB
Фамилия	FAM
Пол	POL
Дата рождения	DTR
Наличие жилья	HOME
Зарплата	ZARP
Биография	BIOGR

Выберем для его полей типы и размеры:

TAB	Числовой тип длиной 4 разряда целых
FAM	Символьный тип длиной 10 символов
POL	Символьный тип длиной 1 символ (М или Ж)
DTR	Тип дата длиной 8
HOME	Тип логический (наличие жилья - .T. отсутствие - .F.)
ZARP	Числовой тип длиной 10, из которых 2 дробных
BIOGR	Тип текстовый выбран ввиду непредсказуемости длины

Для создания базы данных в командном окне наберём команду *create primer*. На экране появится форма для ввода структуры базы данных, которую необходимо заполнить. Результат заполнения представлен на рис. 3.1.

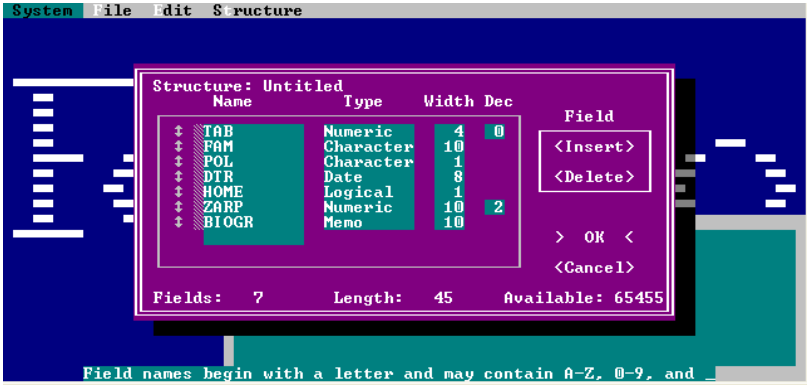


Рис.3.1. Создание структуры базы данных.

После окончания формирования структуры файла она должна быть записана на диске. Это осуществляется комбинацией клавиш Ctrl-W. Нажатие клавиши Escape вызовет отказ от сохранения структуры. Затем FoxPro запросит ввод данных.

Input data records now?	
< Yes >	>No<

Данные можно ввести, ответив YES. Сохранение осуществляется комбинацией клавиш CTRL+W.

MODIFY STRUCTURE - модификация структуры файла базы данных. Команда отображает на экране структуру файла базы данных и позволяет ее редактирование, включая добавление и удаление полей записи, изменение имен, длин и типов полей. После окончания исправления измененную структуру надо запомнить на диске CTRL+W. При модификации базы данных старые структуры сохраняются на диске с расширениями имен bak для dbf-файлов и tbk для fpt-файлов.

DISPLAY STRUCTURE – вывод структуры файла базы данных. После каждого поля выводятся его тип (символьный, числовой, дата, логический или текстовое) и размер. Если поле числовое, выводится количество десятичных знаков. Если база содержит текстовое поле, выводится размер его блока. Отображаются также количество записей в базе, дата последней корректировки, общая длина всех полей.

3.2.3.2 Заполнение базы данных

Файл после создания структуры остается открытым, т.е. доступным для команд ввода, просмотра и изменения. При перезагрузке FOXPRO база данных должна быть вновь открыта командой, то есть, произведено её считывание с винчестера в оперативную память.

USE *< имя файла >* открывает базу данных

USE *< без имени >* закрывает базу данных.

APPEND **–** добавляет новые записи в конец активного файла базы данных, при этом на экран предъявляется окно ввода данных со всеми пустыми полями записи. Все поля базы выделяются контрастным цветом в соответствии, с их длиной. Мемо-поле указывается только словом «memo». После ввода текущей записи автома-

тически происходит доступ к следующей записи и т.д. Для того чтобы войти в мемо-поле, необходимо в него поместить курсор и нажать CTRL+HOME или дважды кнопку мыши. Выход из мемо-поля с сохранением сделанных в нем изменений на диске осуществляется клавишами CTRL+W, без сохранения - ESC**,**

APPEND BLANK **–** добавляет новую пустую запись в конец файла, но на экран не предъявляет и вводить не дает.

Пример: ввода данных представлен на рис. 3.2.

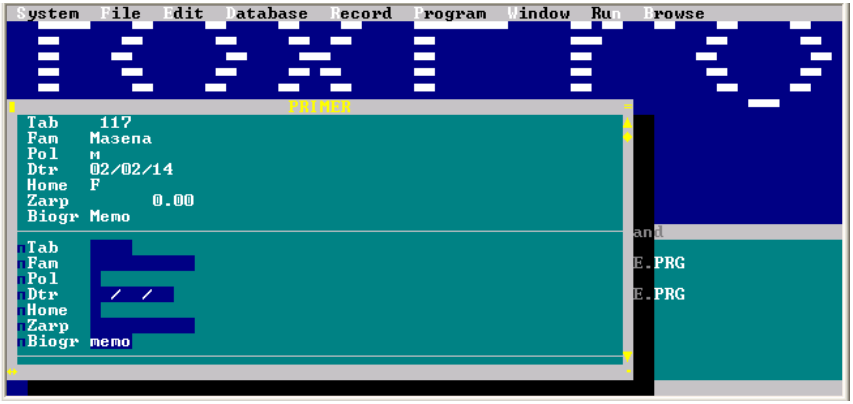


Рис. 3.2. Ввод исходных данных по команде APPEND

3.2.3.3 Редактирование данных

EDIT, CHANGE - команды редактирования введённых записей. Для этих команд окно редактирования будет выглядеть одинаково - все поля базы данных располагаются вертикально. На экране видно столько записей и полей, сколько удается разместить.

Другая команды редактирования - BROWSE - это целая среда доступа и управления данными нескольких баз данных. Записи предъявляются горизонтально на экране. Формат отображения задаётся пользователем. Пометка записей к удалению выполняется клавишами - CTRL+T. Дополнение базы новой записью осуществляется нажатием клавиш CTRL+N. Сохранение в BROWSE - CTRL+W, без сохранения – ESC.. В окне BROWSE можно редак-

тировать и текстовые поля. Для этого курсор нужно установить на текстовое поле и нажать CTRL+PgDn. Если какие-то поля записи не умещаются в строке, с помощью клавиш TAB и мыши возможно перемещение (скроллинг) изображения вправо или влево. По умолчанию окно редактирования может быть видно не целиком. Раскрывается окно во весь экран нажатием клавиш Ctrl-F10. Повторное нажатие клавиш возвращает окно к исходной форме.

Формат команды:

BROWSE [FIELDS *<список полей>] ;*
*[**FOR** <выpL1>] [**PARTITION**<выр N2>] [**NOLINK**];*
*[**TITLE** <выpC2>] [**WINDOW** <имя окна>];*
*[**VALID** <выpL2>] [**COLOR SCHEME** <выpN3>*

Опции:

FIELDS <список полей> - перечень предъявляемых полей, если список не указан, то все поля предъявляются. В список полей могут включаться настоящие поля из базы и создаваться вычисляемые поля, которые являются функциями полей и переменных. Такие поля не редактируются и не запоминаются, но отображаются на экране наравне с существующими полями, что дает возможность пользователю, например, оценивать свои данные по заданному критерию. Вычисляемые поля формируются следующим образом: имя поля задаётся самостоятельно, ставится знак равно, затем арифметическое выражение из существующих полей и функций. Имя каждого поля может сопровождаться ключами, определяющими режим доступа к нему:

[:R]
[:<ширина поля>]
[:V = <выpL1> [:E = <выpC1>]]
[:H = <выpC3>]

R - разрешен только просмотр данных
V - устанавливает контроль ввода данных по выражению L1
E - на ошибку при вводе выдает в правом верхнем углу сообщение- выражение C1
H - указание собственного имени поля - выражение C3

FOR <выр L1> отображаются только те записи, которые удовлетворяют условию L1.

PARTITION <выр N2> делит окно Browse, на 2 части,. Выр N2 указывает позицию разделения, переход по клавише CTRL+H.

NOLINK - несинхронное перемещение записей в разделенных окнах.

TITLE <выр С2> -заголовок «выр С2» всего окна Browse.

VALID <выр L2> контроль ввода данных для всей записи по выражению L2.

WINDOW <имя окна> представляет окно Browse в окне с указанным именем, окно должно быть определено заранее.

COLOR SCHEME <выр N5 > меняет цветовую гамму раскраски окна Browse. По умолчанию - цветовая схема 10.

Пример: Создать окно Browse «Начисление премии» с заголовками полей на русском языке, контролем значения поля табельный номер, выдачей сообщения об ошибке, расчётом премии в вычисляемом поле. Премия составляет 50% от заработной платы.

brow fields tab:h="таб.ном.":v=tab>0:e="неверно",;
fam:h="фамилия":r,zarp:h="зарплата",;
*prem=0.5*zarp:h="премия" title "Начисление премии"*

В команде представлены для редактирования четыре поля, одно из них вычисляемое - prem. Название вычисляемого поля выбирается самостоятельно, после него ставится знак «=» и вводится формула для расчёта, задан ключ h – заголовок поля на русском языке.

– Для поля tab заданы ключи: h – заголовок поля на русском языке, v – проверка табельного номера по условию - табельный номер должен быть положительным числом, e – выдача сообщения об ошибке – «неверно».

– Для поля fam заданы ключи r – запрет на внесение изменений и h – заголовок поля на русском языке.

– Для поля zarp задан ключ h – заголовок поля на русском языке.

Кроме того, задан заголовок всего окна BROWSE - title «Начисление премии».

Результат работы команды представлены на рис. 3.3.

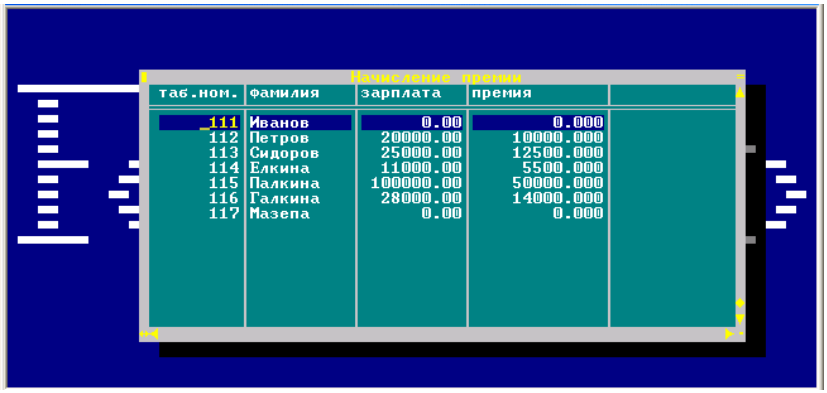


Рис. 3.3. Начисление премии по команде Browse

3.2.3.4 Перемещение в базе данных

При работе с базой данных необходимы средства перемещения внутри нее. Запись, на которой находится указатель записей, является текущей, и только к ней в данный момент возможен непосредственный доступ.

GO – устанавливает указатель записи на соответствующую запись внутри базы данных.

Модификации команды:

– GO <*выр*N>- переход к записи с номером <выр N>

– GO TOP - переход к первой записи.

– GO BOTTOM переход к последней записи

SKIP <выр N> - переход к записи, отстоящей от текущей на N записей, причем N может быть и отрицательным и положительным.

Для контроля положения указателя и наличия записей в файле предусмотрены функции:

RECNO() – возвращает номер записи, на которой стоит указатель.

RECCOUNT() – возвращает общее количество записей файла.

BOF() **–** проверяет выполнение условия «начало файла» для файла базы данных. функция истинна, при попытке установки указателя записи перед первой записью файла базы данных.

EOF() **–** проверяет выполнение условия «конец файла» для файла базы данных Функция возвращает «истина», если указатель записи установлен после последней записи файла.

3.2.3.5 Просмотр данных

LIST **-** просмотр данных всего файла на экране.

DISPLAY–[<*границы****>] [<****поля****>] [OFF] [TO PRINT] [FOR <****условиеL1****>] [WHILE <****условиеL2****>]*** **-** просмотр данных всего файла порциями, поэкранно, при нажатии любой клавиши просмотр может быть продолжен. После выполнения команды указатель записи перемещается на последнюю показанную запись. Записи, помеченные к удалению, команда выдает со звездочкой в первой позиции. Для текстового поля название поля должно быть указано, иначе они выводятся столбцом *memo.*

Опции:

OFF – номера записей не указываются

TO PRINT – результат печатается на принтере

FOR <условиеL1> – только те записи, у которых выполняется условие

WHILE<условиеL2> – только до той записи, у которой выполнится условие

DISPLAY без параметров - осуществляет выдачу всех полей базы данных только одной текущей записи.

Пример: распечатать содержимое базы данных, распечатать данные о женщинах, имеющихся в базе данных.

list

list for pol="ж"

Приведены две команды. Команда list без параметров распечатывает содержимое базы данных. Команда list for pol="ж" распечатывает только те записи, у которых выполняется условие pol="ж", то есть только данные о женщинах, имеющихся в базе, соответствие значений поля в команде и базе должно быть полным, так если в команде значение поля pol было "Ж" , а команда - *list for pol="ж"* то в результате не будет найдено ни одной записи.

Результат работы команды представлены на рис. 3.4.

```
System  File  Edit  Database  Record  Program  Window  Run
Record#   TAB FAM         POL DTR       HOME        ZARP BIOGR
      1   111 Иванов      м   08/08/96 .T.         0.00 Memo
      2   112 Петров      м   09/09/87 .F.     20000.00 Memo
      3   113 Сидоров     м   10/10/70 .T.     25000.00 Memo
      4   114 Елкина      ж   11/11/81 .T.     11000.00 Memo
      5   115 Палкина     ж   12/12/36 .T.    100000.00 Memo
      6   116 Галкина     ж   01/01/55 .F.     28000.00 Memo
      7   117 Мазепа      м   02/02/14 .F.         0.00 Memo

Record#   TAB FAM         POL DTR       HOME        ZARP BIOGR
      4   114 Елкина      ж   11/11/81 .T.     11000.00 Memo
      5   115 Палкина     ж   12/12/36 .T.    100000.00 Memo
      6   116 Галкина     ж   01/01/55 .F.     28000.00 Memo
                                          Command
                                          USE C:\1\PRIMER.DBF
                                          list
                                          list for pol="ж"
```

Рис. 3.4. Просмотр содержимого базы данных и данных о женщинах, имеющихся в базе

3.2.3.6 Удаление данных

ERASE *<файл>* - удаление любого неактивного файла.

ZAP **–** удаление всех записей в активном файле.

DEL *<границы>* ***[FOR****<условиеL1>****] [WHILE****<условиеL2>****]*** - пометка к удалению записей в указанных границах и отвечающих условиям.

DELETE *без параметров* – помечает текущую запись.

PACK**–** физическое удаление помеченных записей и сжатие файла.

RECALL *<границы>* ***[FOR****<условL1>****] [WHILE****<условL2>****]*** - снятие пометок к удалению записей

RECALL *без параметров* действует на текущую запись.

Пример: пометить к удалению потерявших работу, затем отменить пометку к удалению тем, у кого нет жилья.

Текст команды:

delete all for zarp=0
recall all for !home

Первая команда помечает к удалению те записи, у которых условие zarp=0 истинно, вторая отменяет пометку к удалению тем записям, у которых условие home ложно.

Результат работы команды представлены на рис. 3.5.

```
System  File  Edit  Database  Record  Program  Window  Run
        2 records deleted
Record#   TAB FAM        POL DTR      HOME        ZARP BIOGR
      1 * 111 Иванов     м   08/08/96 .T.        0.00 Memo
      2   112 Петров     м   09/09/87 .F.    20000.00 Memo
      3   113 Сидоров    м   10/10/70 .T.    25000.00 Memo
      4   114 Елкина     ж   11/11/81 .T.    11000.00 Memo
      5   115 Палкина    ж   12/12/36 .T.   100000.00 Memo
      6   116 Галкина    ж   01/01/55 .F.    28000.00 Memo
      7 * 117 Мазепа     м   02/02/14 .F.        0.00 Memo

        1 records recalled
Record#   TAB FAM        POL DTR      HOME        ZARP BIOGR
      1 * 111 Иванов     м   08/08/96 .T.        0.00 Memo
      2   112 Петров     м   09/09/87 .F.    20000.00 Memo
      3   113 Сидоров    м   10/10/70 .T.    25000.00 Memo
      4   114 Елкина     ж   11/11/81 .T.    11000.00 Memo
      5   115 Палкина    ж   12/12/36 .T.   100000.00 Memo
      6   116 Галкина    ж   01/01/55
      7   117 Мазепа     м   02/02/14
                                          Command
                                          clea
                                          delete all for zarp=0
                                          LIST
                                          recall all for !home
                                          LIST
```

Рис. 3.5. Пометка к удалению записей и отмена пометки записей, отвечающих условиям

3.2.3.7 Изменение данных

REPLACE<*границы*>[***FOR***<*условL1*][<***WHILE***<*услL2*>] <*поле1*> ***WITH*** <*выражение 1*> <*поле2*>***WITH*** <*выражение 2*> [***ADDITIVE***] – изменение <поле> на <выражение> для записей, соответствующих условию L1.

<*поле*> WITH <*выражение*> может присутствовать несколько раз для разных полей,

ADDITIVE - опция только для текстовых полей, означающая что выражение не заменяет текст, а дописывается в конце него.

Пример: Внести изменения в базу данных человеку по фамилии Мазепа - проставить отметку о том, что он имеет жильё и закончил КемТИПП.

replace all home with .t. biogr with "закончил КемТИПП" additive;
for fam="Мазепа"
list fam,home,biogr

Первая команда изменяет два поля – в логическом поле home изменяет значение ложь на истина, текстовое поле biogr дополняется фразой "закончил КемТИПП" с помощью опции additive. Команда выполняет замену для тех, у кого фамилия "Мазепа". Во второй команде перечислены только три поля, для того чтобы распечатать содержание текстового поля название поля должно быть указано в команде.

Результат работы команды представлены на рис. 3.6.

```
System  File  Edit  Database  Record  Program  Window  Run
 replace all home with .t. biogr with "закончил КемТИПП" additive;
 for fam="Мазепа"
 list fam,home,biogr

Record#  FAM        HOME BIOGR
      1 *Иванов      .T. Ученик школы №3 Мечтает поступить в Кемеровский
                         технологический институт пищевой промышленности
      2  Петров      .F. Начальник транспортного цеха Поступил в
                         Кемеровский технологический институт пищевой
                         промышленности в 2008
      3  Сидоров     .T. Кассир Поступил в Кемеровский технологический
                         институт пищевой промышленности в 2008
      4  Елкина      .T. Лесник Поступила в Кемеровский технологический
                         институт пищевой промышленности в 2008
      5  Палкина     .T. Лыжница Поступила в Кемеровский технологический
                         институт пищевой промышленности в 2008
      6  Галкина     .F. орнитолог
                         Поступил в Кемеровский технологический институт
                         пищевой промышленности в 2008
      7  Мазепа      .T. Гетман Поступил в Кемеровский технологический
                         институт пищевой промышленности в 2008 закончил
                         КемТИПП
```

Рис. 3.6. Замена значений полей у записей, отвечающих условию

3.2.3.8 Фильтрация данных

SET FILTER TO [<условие>] - позволяет установить FOR-условие для всех без исключения команд обработки данных. Здесь <условие> указывает на то, какие именно записи могут быть доступны для обработки.

3.2.3.9 Поиск данных

LOCATE<границы>[**FOR**<усл L1>][**WHILE** <усл L2>] - осуществляет последовательный поиск одной самой первой записи в базе данных, удовлетворяющей заданному условию L1, среди за-

писей, находящихся в заданных границах, и до тех пор, пока соблюдается условие L2

CONTINUE - продолжает поиск записей, начатый ранее командой LOCATE.

При успешном поиске указатель записей устанавливается на найденную запись, функция RECNO( ) равна номеру этой записи, а функция FOUND( ), оценивающая результат поиска, возвращает значение «Истина» (.T.). Если командой LOCATE или CONTINUE не было найдено нужных записей, указатель записей устанавливается на нижнюю границу поиска (если она введена в команде), функция RECNO( ) равна числу записей в базе плюс 1, FOUND( )=.F., а функция достижения конца файла EOF( ) возвращает .T.(Истина).

3.2.3.10 Индексирование баз данных

Самая распространенная операция в системах обработки данных – это поиск. Для реальных БД поиск осуществляется медленно. Для ускорения процесса поиска реализован механизм индексирования БД. Один файл БД может быть проиндексирован по нескольким полям, в которых записи располагаются в хронологическом, алфавитном или числовом порядке.

FoxPro позволяет создавать два типа индексных файлов:

- Индексные файлы с расширением .IDX, содержащие один индексный элемент.
- Мультииндексные файлы с расширением .CDX, содержащие несколько индексных элементов, называемых тегами.

Мультииндексные файлы могут быть 2 видов: структурный файл, с именем, совпадающим с именем БД и обычный с произвольным именем. Структурный автоматически открывается каждый раз, когда открывается база данных.

Создание одноиндексного файла выполняется командой:

INDEX ON *<ключ>* ***TO*** *<имя файла.idx >***[FOR** *<условие L1>]*

Создание мультииндексного структурного файла выполняется следующей командой:

INDEX ON** <ключ> **TAG** <имя тега> **[FOR** <условие L1>] **[DESCENDING] [ADDITIVE]

Создание мультииндексного обычного файла выполняется следующей командой:

INDEX ON** <ключ> **TAG** <имя тега> **[OF** <имя файла.cdx>**]** **[FOR** <условие L1>] **[DESCENDING] [ADDITIVE]

Опции:

Ключ – имя поля, по которому надо упорядочить файл. Ключ может быть составным или функцией нескольких полей или переменных.

TO <имя файла.idx> - задает имя индексному файлу.

TAG <имя тега> [OF <имя файла.cdx>] задает имя тега мультииндексному файлу, для структурного файла опцию OF задавать не надо, имена тегов должны начинаться с буквы или символа подчеркивания и могут содержать любую комбинацию до 10 букв, цифр или символов подчеркивания. Число тегов в файле .CDX ограничено только объемом доступной памяти и пространством на диске.

FOR <условие L1> - отбор записей по условию.

COMPACT – только для индексных файлов. Мультииндексные файлы всегда компактны, поэтому не нужно включать опцию COMPACT при их создании.

DESCENDING - индексирование по убыванию только для мультииндексных файлов. По умолчанию индексные файлы создаются в возрастающем порядке.

ADDITIVE- ранее открытые индексные файлы остаются открытыми.

Если индексный файл был уже создан, то его надо открыть командой:

***USE** <файл> **[INDEX**<список индексных файлов>]*

Или

***SET INDEX TO** [<список индексных файлов>],* если БД уже открыта.

***SET INDEX TO** без параметров* –закрывает индексные файлы.

Для индексных файлов существует специальная команда ускоренного поиска.

SEEK < *выражение*> - используется для поиска в индексированной базе данных первой записи, у которой индексный ключ равен выражению. Соответствие должно быть точным.

SET NEAR ON - позволяет выполнить сравнение приблизительно.

Пример: Упорядочить базу по фамилиям в алфавитном порядке. Упорядочить базу по полу, при этом фамилии должны быть отсортированы в алфавитном порядке

index on fam to fam.idx

index on pol+ fam tag p_fam

Первая команда создаёт одноиндексный файл с именем *fam.idx* , индексирование по ключу fam. Вторая команда создаёт мультииндексный файл по составному ключу *pol+ fam* с именем тега *p_fam.*

Результат работы команды представлены на рис. 3.7.

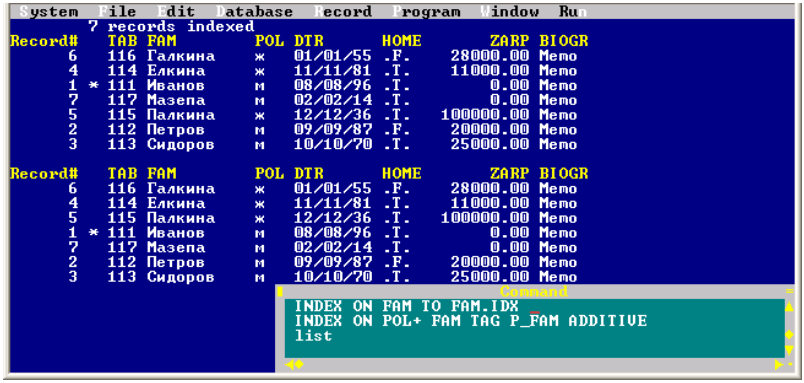


Рис. 3.7. Упорядочивание файла базы данных
с помощью индексирования

3.2.3.11 Сортировка данных

SORT *<границы>* ***[FOR*** *<выpL1>]* ***[WHILE*** *<выpL2>]* ***TO*** *<имя нового файла >* ***ON*** *<поле1>* ***ASCENDING | DESCENDING*** - **с**оздает новый файл базы данных, в котором записи отсортированы в указанном порядке по заданным ключевым полям.

ASCENDING **с**ортировка выполняется по возрастанию значений.

3.2.4 Математическая обработка базы данных

COUNT*<границы> [****FOR****<выpL1>] [****WHILE*** *<выpL2>]* ***TO*** *переменная* -

Выполняет подсчет числа записей базы данных в заданных границах, удовлетворяющих заданным условиям, и заносит в переменную.

SUM *<границы> [****FOR****<выpL1>] [****WHILE*** *<выpL2>] список выражений* ***TO*** *список временных переменных*

Вычисляет суммы числовых полей базы данных в заданных границах, удовлетворяющих заданным условиям, и заносит во временные переменные.

список выражений - одно или несколько полей или выражений для вычисления суммы. Если список выражений не указан, будут просуммированы все поля.

AVERAGE *<границы>* ***[FOR****<выpL1>]* ***[WHILE*** *<выpL2>]* *список выражений* ***TO*** *список переменных*

Подсчитывает средние арифметические значения выражений или полей базы в заданных границах, удовлетворяющих заданным условиям, и заносит во временные переменные.

CALCULATE *<границы>* ***[FOR****<выpL1>]* ***[WHILE*** *<выpL2>] список выражений* ***TO*** *список переменных*

Выполняет финансовые и статистические вычисления над полями базы данных или выражениями, в которых содержатся поля в заданных границах, удовлетворяющих заданным условиям, и записываются во временные переменные.

*список выражений с*одержит любой набор следующих внутренних для этой команды функций:

AVG(выр N) вычисляет среднее арифметическое полей.

CNT() - вычисляет количество записей в базе данных.

MAX(выр) - вычисляет максимальное значение поля и работает с полями числового, символьного типа или типа даты.

MIN(выр) - вычисляет минимальное значение поля и работает с полями числового, символьного типа или типа даты.

STD(выpN) - вычисляет среднеквадратическое отклонение значения поля, работает с полями числового типа. Эта величина характеризует степень отклонения значений полей от среднеарифметического значения поля.

SUM(выpN) - вычисляет сумму значения поля, работает с полями числового типа.

VAR(выpN) вычисляет дисперсию значения поля, работает с полями числового типа.

Пример: Определить количество женщин в базе данных и количество женщин, имеющих жильё. Определить сумму заработной платы всех работников.

count all for pol="ж" to a
count all for pol="ж" and home to b
sum all zarp to c
? "Кол. женщин в базе " +str(a,2)+" чел."
? "Кол. женщин в базе, имеющих жильё " +str(b,2)+" чел."
? "сумма заработной платы " +str(c,10,2)+" руб."

Первая команда подсчитывает количество записей, соответствующих условию pol="ж" и помещает результат в переменную a. Вторая - количество записей, соответствующих двум условиям одновременно pol="ж" and home и помещает результат в переменную b. Третья команда подсчитывает сумму заработной платы всех работников и помещает результат в переменную c. Остальные команды выводят полученные результаты.

Результат работы команд представлен на рис. 3.8.

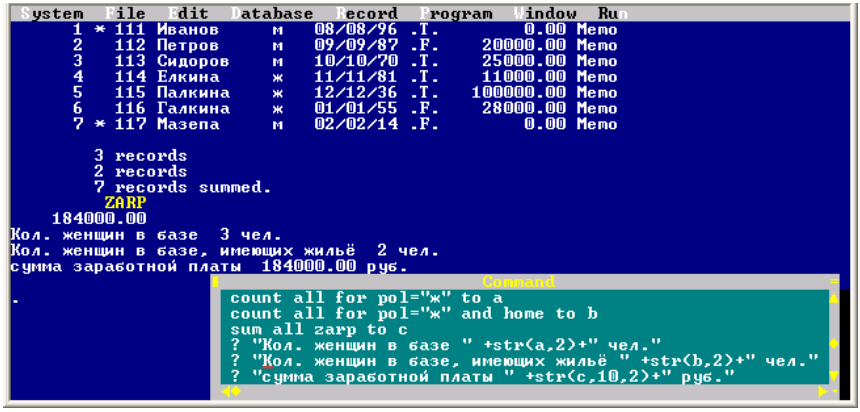


Рис. 3.8. Подсчёт количества и суммы

3.2.5 Работа со связанными базами данных

В FoxPro допускается работа сразу со многими базами данных и при этом возможно установление разнообразных связей между ними. Указатели записей в таких связанных базах будут двигаться синхронно. База, в которой указатель движется произвольно, считается старшей (родительской), а база, в которой указатель следует за указателем старшей базы - младшей (дочерней). В старшей и младших базах должны быть поля, несущие какой-то общий признак, иначе, хотя связь и возможна, она будет бессмысленна. Допускается сцепление одной базы с несколькими другими. Младшие базы, в свою очередь, могут быть связаны с базами следующего уровня и т.д.

Возможно установление двух типов связей.

- Связь типа одна-запись-к-одной перемещает указатель в младшей базе таким образом, что он всегда устанавливается на первую встреченную им запись с совпадающим признаком. Остальные такие записи (если есть) остаются «не замеченными».
- Связь типа одна-запись-ко-многим позволяет обратиться ко всем записям младшей базы с совпадающим признаком.

Оба типа связей могут быть распространены на несколько баз сразу. Каждый файл открывается в своей отдельной рабочей области.

SELECT <*рабочая область*> *п*ереход в область. Первые десять рабочих областей идентифицируются номерами 1 -10 или буквами A — J. Области с 11-й по 25-ю обозначаются номерами или буквенно-цифровыми именами W11 - W25. Если в качестве параметра, указать цифру 0, произойдет переход в первую свободную рабочую области. Область, в которой мы находимся в данный момент, называется активной рабочей областью.

SET RELATION TO <*ключ*> ***INTO*** <*область*> ***[ADDITIVE]*** - команда организации связи вида одна-запись-с-одной. Связывает указатель записей в активной рабочей области с указателями записей из другой рабочей области, имя которой указано после слова

INTO, по заданному общему полю (ключу). Файл, с которым устанавливается связь, должен быть проиндексирован по этому полю.

В FoxPro имеется возможность устанавливать связи с несколькими базами одновременно. Если со старшим файлом, который уже связан с другим, необходимо связать некоторый третий следует во все последующие команды SET RELATION включить слово ADDITIVE, которое обеспечит сохранение связей, установленных ранее.

SET RELATION TO** без параметров**. - команда, которая разрывает связь между всеми файлами

***SET SKIP TO** <область>* - устанавливает связь вида одна-запись-со-многими между двумя или несколькими базами данных. При этом с каждой записью из старшей базы могут быть сцеплены несколько записей из младшей базы. Связь может быть установлена сразу с несколькими младшими базами, находящимися в указанных областях. Прежде чем использовать команду SET SKIP TO. необходимо выполнить начальное сцепление вида одна-запись-с-одной командой SET RELATION.

***SET SKIP TO** без параметров* команда, которая удаляет связи одна-запись-со-многими.

Команда BROWSE позволяет выявлять связи между базами данных. Одно окно BROWSE может включать в себя поля, как из порождающей базы данных, так и из всех ее дочерних баз. Это обеспечивается включением в команду опции FIELDS со списком этих полей.

Пример: Выявить в базе людей, у которых есть дети.

select 2
use deti
index on tab to deti
select 1
use primer
set relation to tab into deti
browse fields tab,fam,deti.tab,deti.fam,deti.dtr

Первая и вторая команды выбирают рабочую область 2 и открывают в ней базу данных детей. Третья команда индексирует базу по общему полю табельный номер. Четвёртая и пятая команды выбирают рабочую область 1 и открывают родительскую базу данных. Шестая команда устанавливает связь одна- -с-одной от

родительской базы к дочерней по общему полю tab. Последняя команда организует окно редактирования для просмотра списка родителей и одного их ребёнка.

Результат работы программы представлен на рис. 3.9.

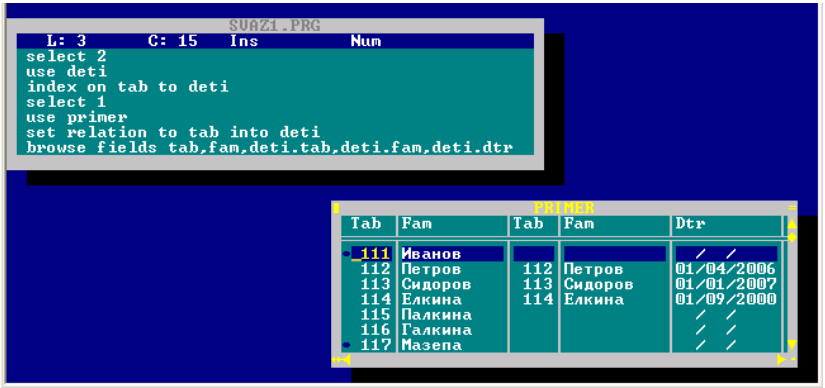


Рис. 3.9. Организация связи вида одна-запись-с-одной между двумя базами данных

Пример: Выявить в базе данных людей, у которых есть дети. Сформировать список людей с указанием наличия детей и их перечня с датой рождения

select 2
use deti
index on tab to deti
select 1
use primer
set relation to tab into deti
set skip to deti
browse fields tab,fam,deti.tab,deti.fam,deti.dtr

Этот программа отличается от предыдущей командой set skip to deti, которая преобразует связь одна- -с-одной в связь одна- -со-многими. Последняя команда организует окно редактирования для просмотра списка родителей и всех их детей с указанием их фамилий и дат рождения.

Результат работы программы представлен на рис. 3.10.

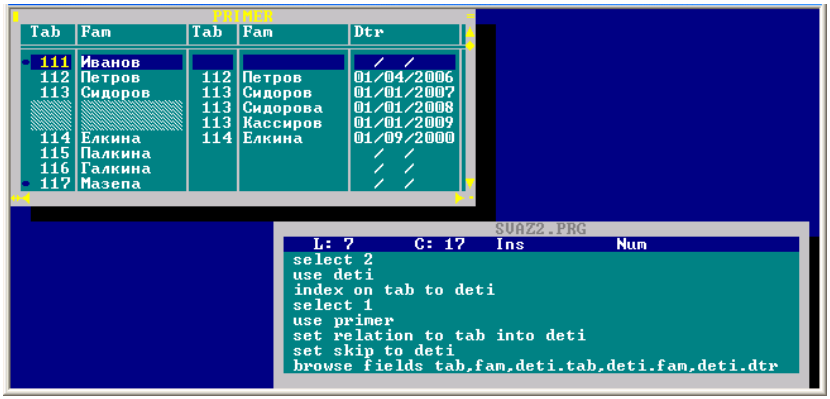


Рис. 3.10. Организация связи вида одна-запись-со-многими между двумя базами данных

3.3 Основы программирования в FoxPro 2.6

MODIFY COMMAND *<имя программы>* - открытие редактора текста для написания программы. Любой фрагмент программы можно выделить, вырезать, скопировать, переместить. Программа сохраняется в активном каталоге в файле с расширением *.prg*. комбинацией клавиш CTRL+W.

DO *<имя программы>* - запуск программы на выполнение.

3.3.1 Оператор присваивания

Оператор присваивания существует в двух вариантах:

переменная = <выражение>

STORE <выражение> ***TO*** <список переменных> / <массив> - помещает данные в переменные и массивы..

Пример*:* В переменную p поместить символьное выражение "Привет" , в первый и пятый элемент массива d, состоящего из 10 элементов занести значение 10.

store "привет" to p
dimension d(10)
d=0
store 10 to d(1),d(5)

3.3.2 Операторы ввода-вывода

CLEAR –оператор очистки экрана.

@ Y, X SAY *<выражение>* - выводит выражение в строке с номером Y, начиная со столбца с номером X. Y меняется в диапазоне от 0 до 24, X – от 0 до 79.

Если выражение символьное, то оно заключается в апострофы, если числа - то нет, если смесь чисел и символов, то числа надо преобразовать в символьную переменную, используя функцию STR(переменная, количество позиций).

? - *<выражение>* ***AT*** *N* – печатает выражение на следующей строке от курсора, начиная со столбца N.

?? *<выражение>* - печатает выражение на месте остановки курсора.

Пример: В 30 колонке 10 строки напечатать текст “программа № 2”

@ 10, 30 say “программа №”+str(2,1)

@ Y, X GET *<переменная > /<поле>* - предъявляет данные в строке с номером Y, начиная со столбца с номером X.

READ - считывает данные, указанные в GET, всегда стоит после GET.

Таким образом, ввод состоит из 2 частей: GET выделяет на экране рамку, READ считывает с клавиатуры.

Пример: В переменную p поместить символьное выражение "Привет", ввод организовать с клавиатуры.

p=” “
@ 10, 30 say “Введите Привет”
@ 10, 50 get p
read

INPUT *<сообщение>* ***TO*** *<числовая переменная>* – команда ввода для чисел, если будете с помощью неё вводить символьную переменную, то она должна быть заключена в апострофы, сообщение – это поясняющая информация к вводу.

ACCEPT *<сообщение>* ***TO*** *<символьная переменная>*, команда ввода для символов, если будете с помощью неё вводить числовую переменную, то она должна быть заключена в апострофы сообщение – это поясняющая информация к вводу.

Пример: В переменную p поместить символьное выражение "Привет", ввод организовать с клавиатуры.
accept “Введите Привет” TO p

3.3.3 Оператор условия

IF *<условие>*
<операторы>
[ELSE
<операторы>]
ENDIF

Если условие истинно, то выполняются все операторы от IF до ELSE, если ложно, то операторы от ELSE до ENDIF. Если необязательная фраза ELSE отсутствует и условие ложно, все внутренние операторы пропускаются и выполняется оператор следующий за ENDIF. В одной строке можно записать только один оператор. Оператор условия записывается столбцом, каждый оператор в отдельной строке.

3.3.4 Оператор выбора

DO CASE
CASE *<условие 1>*
<операторы 1>
CASE *<условие 2>*
<операторы 2>
CASE *<условие 3>*
<операторы 3>
OTHERWISE
<операторы 4>
ENDCASE

Оператор используется для множественного выбора или организации меню.

3.3.5 Организация циклов

1 вид:
DO WHILE *<условие>*
<операторы 1>

[LOOP]

<операторы 2>

[EXIT]

ENDDO

Оператор цикла, обеспечивающий повторение последовательности операторов, заключенной в конструкции DO WHILE ENDDO до тех пор, пока заданное условие "истинно". ENDDO и LOOP возвращают управление оператору DO WHILE для следующей оценки "истинности" условия. EXIT передает управление следующему за ENDDO оператору. Если условие оценивается как "ложное" система FoxPro пропускает все операторы, заключенные между DO WHILE и ENDDO и переходит на выполнение следующего за ENDDO оператора. ENDDO завершает структуру DO WHILE.

2 вид:

FOR *<перем> = <выpN1>* ***TO*** *<выpN2>* ***[STEP*** *<выpN3>****]***

<операторы>

[EXIT]

[LOOP]

ENDFOR

Выполняет операторы в цикле указанное число раз. Переменная <перем> используется как счетчик, определяющий число выполнений. Числовое выражение <выpN1> задает начальное значение счетчика, а числовое выражение <выpN2> - окончательное значение. Включенные в конструкцию операторы выполняются последовательно, пока не встретится ENDFOR. При этом значение <перем> увеличивается на значение, задаваемое фразой STEP <выpN3>. Если эта фраза опущена, счетчик увеличивается на единицу. Затем значение счетчика сравнивается с <выpN2>, если счетчик меньше или равен значению <выpN2>, цикл выполняется еще раз. В противном случае управление передается оператору, следующему за ENDFOR. Если значение STEP <выpN3> отрицательное, то значение счетчика всякий раз уменьшается на эту величину, и исходное значение <выpN1> должно быть больше значения <выpN2>. Фраза EXIT прерывает выполнение цикла с передачей управления оператору, следующему за ENDFOR. Она может располагаться в любом месте между FOR и ENDFOR. Фраза LOOP также может быть помещена в любом месте между FOR и

ENDFOR. Она передает управление команде FOR, не дожидаясь команды ENDFOR. Конструкция FOR ENDFOR может быть вложенной.

3 вид:

SCAN** [<сфера>] **[FOR** <условие 1>**] [WHILE** <условие 2>**]
[<операторы>]
[LOOP]
[EXIT]
ENDSCAN

Просматривает всю базу данных и для каждой записи, отвечающей заданным условиям, выполняет операторы, заключенные в программную конструкцию. SCAN автоматически наращивает границы. Перед использованием команды SCAN база данных должна быть открыта. По умолчанию значение <сферы> - ALL. Фраза LOOP может помещаться в любом месте между SCAN и ENDSCAN и вызывает немедленный возврат к оператору SCAN в начало цикла. Фраза EXIT вызывает прекращение выполнения цикла и переход к оператору, непосредственно следующему за ENDSCAN.

3.3.6 Функции СУБД

Функции в FoxPro используются для анализа или преобразования данных. Синтаксическая особенность функций - обязательное наличие скобок (кроме функции &). Функции разбиты на следующие группы:

– **математические функции**

ABS(N) – модуль числа N

BETWEEN(*<выр>, <выр1>,<выр2>****)*** **–** функция истинна, когда *выр1<=выр<=выр2*

Пример: Распечатать фамилии девушек, с заработной платой в пределах от 10000 до 20000 рублей.

list fam for between(zarp,10000,20000) and pol='ж'

INT(N) –целая часть числа N

MAX(*<выр>, <выр1>, <выр2>...)* - максимальное значение среди перечисленных выражений

MIN(*<выр>, <выр1>, <выр2>...)* - минимальное значение среди перечисленных выражений

MOD(N,N1) –целочисленный остаток от деления числа N на N1

ROUND(N,N1) –округление числа N до N1 знаков после запятой.

RAND() – выдача случайного числа в диапазоне от 0 до 1 по равномерному закону распределения.

Пример: Выдать целое случайное число на интервале от A до B

*? int((b-a+1)*rand()+a)*

SIGN(n)- знак числа, если число n положительное, то значение =1, иначе = -1

EXP(N) **-** экспонента числа N

LOG(N) –натуральный логарифм числа N

LOG10(N) – десятичный логарифм числа N

SQRT(N) – корень квадратный из числа N

SIN(N) – синус числа N

COS(N) – косинус числа N

TAN(N) - тангенс числа N

ATAN(N)- арктангенс числа N

ASIN(N) косинус числа N

PI() – число π

RTOD(угол в радианах) – перевод угла из радианной меры в градусную.

DTOR(угол в градусах) - перевод угла из градусной .меры в радианную.

– **строковые функции**

<выр1>$<выр2> – функция истинна, когда *выр1* содержится *в выр2*

INLIST(*<выр>, <список>****)*** – функция истинна, когда *выр1* содержится в *списке.*

IIF(*<условие>, <выр1>,<выр2>****)*** – выдает значение *выр1* ,если *условие* истинно, и *выр2* - если ложно, допускается вложенность функции.

Пример: Выдать список фамилий, перед фамилией поместить обращение

list iif (pol="ж", "Уважаемая ", " Уважаемый ")+fam

LIKE(*<выр1>, <выр2>****)*** – функция истинна, когда *выр1* содержится в *выр2,* для *выр1* допускается использование шаблонов –

(* - любое количество символов и ? – любой одиночный символ), используется для поиска по неполному ключу.

Пример: Выдать список фамилий, содержащих не менее двух букв «а».

*list fam for like(”*a * a*” , fam)*

LEN(выр)- число символов в выражении**.**

SUBSTR(*выр,N, N1****)-*** выделение подстроки из выражения, начиная с позиции N длиной N1.

Пример: Выдать список фамилий и первые 10 символов из их биографии.

list fam.substr(biogr,1,10)

LEFT(*<выр>, N****) –*** выделение из *выр* слева *N* символов.

RIGHT(*<выр>, N****) –*** выделение из *выр* справа *N* символов.

Пример: Выдать список фамилий и последние 10 символов из их биографии

list fam. right (biogr,10)

– **функции преобразования данных**

LTRIM(*<выр>****)*** –удаление ведущих пробелов в *выр*.

TRIM(*<выр>****)*** –удаление завершающих пробелов в *выр*.

Пример: Распечатать список тех, у кого фамилия заканчивается на ‘ ин‘

list fam for substr(fam,len(trim(fam))-1,2)=’ин’

ALLTRIM(*<выр>****)*** –удаление в *выр* ведущих и завершающих пробелов.

REPLICATE(*<выр> ,N****) –*** *выр* повторяется *N* раз.

Пример: Напечатать строку подчеркивания

? replicate(“-“,30)

STUFF(*<выр1>,N,N1,<выр2>****) –*** в выр1 вместо старого текста помещается новый, начиная с позиции N длиной N1, замещая его.

Пример: Распечатать список фамилий, заменив слово «товарищ» на слово «господин»

list stuff(‘товарищ ’+fam,1,8,’господин’)

SPACE(*N****) –*** формирование строки пробелов длиной *N*.

LOWER(*<выр>****)*** **-** прописные преобразуются в строчные.

UPPER*(<выр>)* - строчные преобразуются в прописные.

– **функции работы с датами:**

CDOW(<дата>) –возвращает имя дня недели на английском языке из даты.

CMONTH(<дата>) –возвращает имя месяца недели на английском языке из даты.

DOW(<дата>) –возвращает номер дня недели из даты.

MONTH(<дата>) –возвращает номер месяца из даты.

DAY(<дата>) –возвращает номер дня месяца из даты.

MONTH(<дата>) –возвращает номер месяца из даты.

YEAR(<дата>) –возвращает год из даты.

DATE() –возвращает системную дату**.**

Пример: Распечатать сегодняшний день недели и месяца.

? cdow(date()),day(date())

Распечатать список фамилий и возраст каждого человека.

list fam,year(date())-year(dtr)

– **функции преобразования типов данных:**

CTOD(<символьная>) –преобразует дату из символьной формы в формат даты.

DTOC(<дата>) –преобразует дату из формата даты в символьный формат.

STR(*N,N1,N2****)*** – преобразует число *N* в символьную форму, где *N1* –общая длина числа, *N*2 – длина дробной части.

VAL(*<символьная >****)*** – преобразует символьные данные в число. Преобразование начинается с самого левого символа и продолжается до тех пор, пока не встретится нецифровой символ или не закончится строка символов. Ведущие пробелы игнорируются. Если выражение не число, возвращается 0.

ASC*(выр*C) - возвращает код ASCII первого символа строки.

CHR*(выр*N) – возвращает символ, у которого код ASCII соответствует этому числу. Функция может использоваться для засылки в принтер управляющих кодов.

Пример: Сформировать длительный звуковой сигнал.

? replicate(chr(7),2000)

– **функции проверки позиционирования курсора**

BOF() – проверяет выполнение условия "начало файла" для базы данных. функция истинна, при попытке установки указателя записи перед первой записью файла базы данных.

EOF() **–** проверяет выполнение условия "конец файла" для файла базы данных Функция истина, если указатель записи установлен после последней записи файла.

DELETED() **–** функция истинна, когда указатель стоит на записи, помеченной к удалению.

Пример: Подсчитать количество записей, помеченных к удалению.

calculate cnt() for deleted() to d

RECNO() **–** возвращает номер записи, на которой стоит указатель.

RECCOUNT() **–** возвращает общее количество записей файла.

Пример: Выбрать из базы фамилию случайным образом.

*a= int((reccount()-1+1)*rand()+1)*

list fam for recno()=a

– **функции позиционирования выдачи данных:**

COL() - возвращает номер колонки в строке экрана, на которой расположен курсор. Функция особенно полезна при управлении курсором из программы.

– **функции подстановки:**

& <*временная переменная*> Функция & указывает системе на необходимость выполнения макроподстановки. Значение временной переменной замещает символ и следующее за ним имя. Затем команда выполняется обычным образом, как если бы она была задана с явным включением символов временной переменной.

Пример: Открытие базы данных. Имя базы вводится с клавиатуры.

input 'введите имя базы данных' to basa

use &basa

– **функция вызова функции:**

= *функция* - функция FoxPro должна выполнить действие, но нет необходимости присваивать возвращаемое функцией значение какой-либо переменной.

Пример: Задержка работы на 120 секунд.

= inkey(120)

Программный переход на верхний регистр набора букв.

= capslock(.t.)

3.4 Генераторы приложений

Генераторы приложений облегчают разработку основных элементов приложений: отчётов, меню, экранных форм, запросов и проектов, позволяя получать программный код без ручного набора, но оставляя только ручное кодирование в специальном окне-планшете.

3.4.1 Генератор отчёта

Генератор отчёта позволяет программисту без рутинного труда создавать отчёт. Формирование отчёта сводится к физическому размещению элементов в специальном окне-планшете. В планшете создаётся образ того отчёта, который необходим.

3.4.1.1 Приёмы работы в планшете

Объект – это любая созданная рамка, надпись, поле.

Создание объекта – установить курсор и выбрать тип объекта из меню Report или комбинацией клавиш:

Объект	Пункт меню	Комбинация клавиш
Рамка	Box	CTRL+B
Поле	Fields	CTRL+F
Текст	Text	CTRL+T

Перемещение среди объектов - позиционирование курсора мыши на объекте, или клавишей TAB, или SHIFT+TAB.

Выбор объекта - нажмите левую кнопку мыши на объекте, или позиционируйте курсор на объекте и нажмите клавишу (пробел).

Отмена выбора объекта - SHIFT+кнопка мыши или SHIFT+пробел на объекте.

Перемещение объекта - нажмите левую кнопку мыши на объекте и переместите его, или выберите объект, переместите курсор с помощью клавиш направления, нажмите ENTER..

Изменение длины объекта - нажмите CTRL+кнопку мыши и переместите указатель, или нажмите CTRL+пробел, переопределите размер с помощью клавиш направления, нажмите ENTER

Модифицировать объект - нажмите дважды кнопку мыши на объекте, или нажмите ENTER 2 раза. В диалоговом окне выбрать одну из возможностей модификации.

Удаление объекта - объект выделить и удалить клавишей DELETE.

Копировать объект - объект выделить и комбинацией клавиш CTRL+C отправить в буфер, установить курсор на нужное место и комбинацией клавиш CTRL+V скопировать из буфера.

3.4.1.2 Создание отчёта

CREATE REPORT *<имя отчёта >* -вызывает планшет для создание отчёта из командного окна.

Базы данных должны быть предварительно открыты. При работе создаётся 2 типа файлов (с расширением .FRX и .FRT), которые хранят образ отчёта и параметры среды. В системном меню появляется новый пункт меню REPORT.

MODIFY REPORT *<имя отчёта >* - вызывает планшет для внесения исправления в отчёт.

При переходе в генератор отчёта на экране появится специальное окно, в котором находится условное отображение будущего отчёта – планшет, который представлен на рис. 3.11.

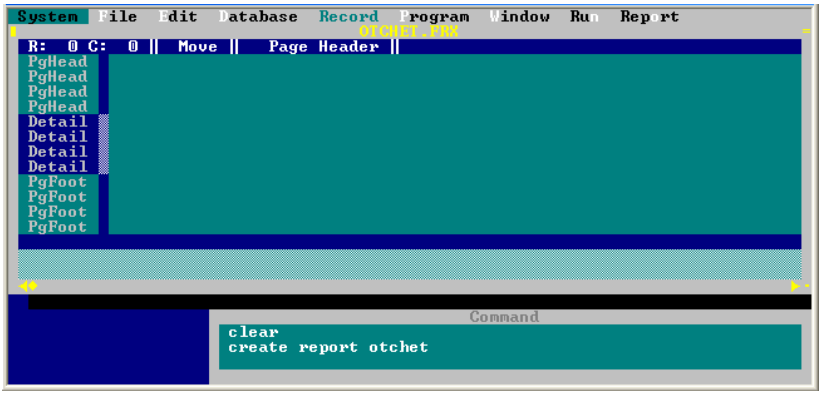


Рис. 3.11. Планшет генератора отчёта

Планшет находится в системном окне, положением и размещением которого можно управлять. Отчет шире размера экрана. Самая верхняя строка (статус-строка) планшета включает указание в нем на координаты курсора, режим курсора и положение курсора в области планшета. Курсор имеет следующие режимы:

- Move – режим перемещения, в котором курсор может свободно перемещаться на экране (действует по умолчанию),
- Text - режим ввода текста. В режим Text можно перейти, нажав любую содержательную клавишу, CTRL-T, либо выбрав пункт TEXT в REPORT-меню. В этом режиме можно вводить любой текст на экране и он воспроизведется в отчете.
- Field - режим работы с полями базы, В режим FIELD можно перейти с помощью меню или CTRL-F.
- Box - режим работы с рамками. В режим Box можно перейти с помощью меню или нажав клавиши CTRL-B

Аналогично возврат в режим Move осуществляется нажатием клавиши Enter или Escape. В последнем случае изменения, сделанные в объекте, не сохраняются.

Планшет может иметь следующие области, которые указаны надписями в левой части экрана:

Title	заголовок всего отчета;
Page Header	заголовок каждой страницы отчета;
Detail	содержание отчета;
Page Footer	подвал каждой станицы;
Summary	подвал всего отчета;

Строки могут быть дополнены через пункт ADD LINE (CTRL-N) или удалены через пункт REPORT-меню REMOVE LINE (клавиши CTRL-0}. В подвале страницы (левый нижний угол) выводится текущая дата (DATEO), номер страницы (правый нижний угол) _PAGENO.

Контрольный просмотр отчета может быть выполнен нажатием клавиш CTRL-I или выбором пункта PAGE PREVIEW в REPORT-меню.

Формирование линий подчёркивания осуществляется следующим образом: Установить курсор в начале линии подчёркивания, нажать комбинацию клавиш CTRL+B, протащить до конца линии,

нажать стрелку движения курсора вверх. Чтобы сделать её двойной, нужно выделить её. В окне диалога выбрать DOUBLE.

Формирование выражений осуществляется следующим образом: дважды нажмём кнопку мыши, или CLRL-F. В ответ будет предъявлено окно REPORT EXPRESSION.. Перейдём в окно построителя выражений, выбрав кнопку <Expr...>

Здесь в области ввода REPORT <Expr> формируется желаемое выражение. В верхней части экрана изображены четыре POPUP-меню, через которые доступны все функции FoxPro, разбитые на группы, — математические, строковые, логические функции и функции работы с датами (MATH, STRING, LOGICAL, DATE). Кроме этого можно выбрать ширину вывода (WEIGHT), различного рода вычисления по вертикали (CALCULATE).

Рассмотрим пункт системного меню Report, представленный на рис. 3.12.

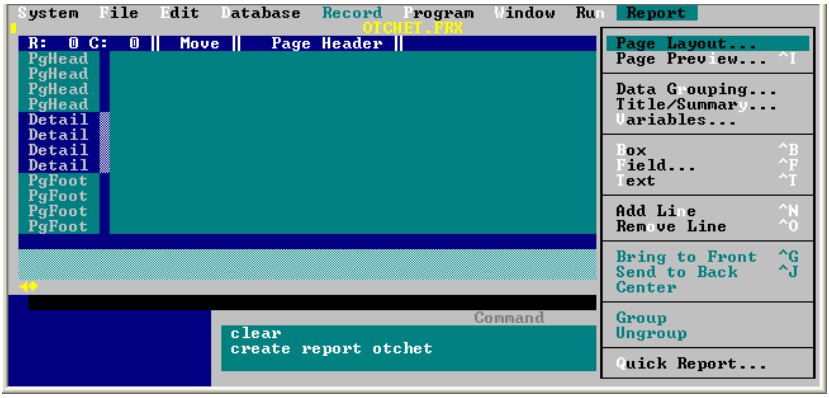


Рис. 3.12. Пункты меню работы с отчётом

PAGE LAYOUT - диалоговое окно форматирования страницы, позволяет сохранить информацию о состоянии операционной среды. Состоит из подпунктов:

– ***Page length (rows)*** - вводится количество строк для каждой страницы отчета.

– ***Top margin (rows)*** **-** вводится количество пробельных строк, которые появятся в верхней части каждой страницы отчета.

– ***Bottom margin (rows)*** **-** вводится количество пробельных строк, которые появятся в нижней части каждой страницы отчета.

– ***Printer indent (columns)*** **-** вводится количество знаков для отступа от левого края печатаемой страницы (отступ слева)

– ***Right margin column*** **-** вводится число символов, печатаемых в строке (ширина печати)

– ***Environment*** **–** сохранение параметров среды, то есть команд конфигурации и открытие баз данных.

– ***Options*** содержит подпункты: ***Plain page*** **–** не выводить заголовки полей на английском языке; ***Summary report*** **-** не выводить данные, а только итоги; ***Suppress blank lines*** **–** не выводить пустые строки из базы данных; ***Add alias*** **–** к именам полей добавить алиас (псевдоним), если в отчёт выводятся данные из нескольких баз.

PAGE PREVIEW выводит отчет для предварительного контроля внешнего вида перед печатью. В нижней части диалогового окна предварительного просмотра страницы Page Preview появляются два индикатора: ***More*** *(Далее).* позволяет прокрутить вперед отчет; ***Done*** *(Выбор)* вызывает возврат в окно компоновки отчета.

DATA GROUPING **-** группирует данные по признакам

TITLE/SUMMARY **-** выделяет строки для заголовка и подвала.

VARIABLES **–** вызов диалога создания переменных.

BOX **–** создание линий и рамок.

FIELDS – вызов диалога построения выражений.

TEXT – ввод текста в отчёт.

ADD LINE **-** выделяет строки в отчёте

REMOVE LINE**-** убирает строки в отчёте.

BRING TO FRONT **–** прямой порядок обхода полей.

SEND TO BACK **–** обратный порядок обхода полей.

CENTER **–** центрирования объекта на странице.

QUICK REPORT **-** генератор быстрого отчёта, содержит следующие опции: ***Column Layout*****-** вывод отчёта столбцами; ***Form Layout*****-** вывод отчёта строками; ***Titles*** **-** выдача заголовков полей на английском языке; ***Fields*****-** осуществляет выбор полей для отчета. По умолчанию выбираются все поля. В диалоговом окне два спис-

ка: список полей БД и список выбранных полей. Поля выбираются кнопкой MOVE, возвращаются кнопкой REMOVE.

Пример: Создать отчёт с заголовком «Список сотрудников», сгруппировать их по полу, фамилии должны быть указаны в алфавитном порядке. Произвести подсчёт количества мужчин и женщин и сумм их заработных плат, общее количество сотрудников, В списке указать количество полных лет каждого сотрудника. Отчет сформировать на текущую дату. Указать текущее время выдачи отчёта. Страницы отчёта пронумеровать.

Порядок работы:

Перед созданием отчёта базу необходимо открыть, упорядочить по полу и по фамилиям командой *index on pol+ fam tag p_fam.* Предварительные команды и параметры операционной среды могут быть сохранены в пункте меню ENVIRONMENT.

Создать отчёт с именем OTCHET командой *create report OTCHET.*

Приступить к формированию отчёта удобно с пункта меню QUICK REPORT. Выбрать «формирование колонками, заголовки полей не выдавать, выбрать поля из базы данных «фамилия», «дата рождения», «пол», «заработная плата».

Сгруппировать данные по полу с помощью пункта меню DATA GROUPING. ADD, GROUP, выбрать поле *pol*. В планшете появится новая область 1-pol, состоящая из двух строк: верхняя строка для заголовка группы, нижняя - для итогов группы.

Добавить области заголовка и подвала отчёта с помощью пункта меню TITLE/SUMMARY. В область TITLE добавить три строки, во вторую строку внести заголовок «Список сотрудников на», переместить из левого нижнего угла планшета поле с функцией системной даты DATE() справа от заголовка. Установить курсор на первой позиции первой строки, сформировать линию подчеркивания до конца заголовка (CTRL+B), опустить вниз на две строки. Выделить образовавшуюся рамку двойным щелчком левой кнопки мыши, в диалоговом окне выбрать двойную линию.

В области PGHEAD построить линию подчеркивания заголовка колонок отчёта. Набрать заголовок «N Фамилия Пол Возраст Зарплата» и снова построить линию подчеркивания.

В области DETAIL сформировать новое поле для порядкового номера внутри группы, для этого вызываем построитель выра-

жений комбинацией клавиш CTRL+F, в диалоговом окне выбираем EXPRESSION, выбираем опцию CALCULATE, COUNT, EXPRESSION, поле *fam.* Для поля *pol* сформировать выбор слова «мужской» или «женский» в зависимости от пола. Среди функций LOGICAL выбрать функцию IIF() и сформировать её следующим образом *iif(pol='м','мужской', 'женский'),* установить ширину поля =8. Для поля *dtr* среди функций DATA выбрать функцию YEAR(*dtr*).

Создать заголовок групп «мужчины» или «женщины», для этого сформировать функцию *iif(pol='м','мужчины', 'женщины')* в области заголовка группы установить ширину созданного поля 20.

Создать подвал групп «мужчины» или «женщины», для этого сформировать функцию *'Итого по '+iif(primer.pol='м','мужчинам','женщинам'),* в области подвала группы установить ширину созданного поля 26.

Для подсчёта сумм заработных плат по группам вызываем построитель выражений комбинацией клавиш CTRL+F, в диалоговом окне выбираем EXPRESSION, выбираем опцию CALCULATE, SUM, EXPRESSION, поле *zarp,* выбираем в опции RESET *pol.*

В области PGFOOT формируем подвал страницы, для этого набираем текст «Страница», перемещаем системную функцию _PAGEN() из правого нижнего угла планшета.

В области SUMMARY формируем подвал отчёта, для этого набираем текст «Итого», для подсчёта сумм заработных плат по отчёту вызываем построитель выражений комбинацией клавиш CTRL+F, в диалоговом окне выбираем EXPRESSION, выбираем опцию CALCULATE, SUM, EXPRESSION, поле *zarp,* выбираем в опции RESET параметр END OF REPORT.

Сформированный планшет представлен на рис. 3.13.

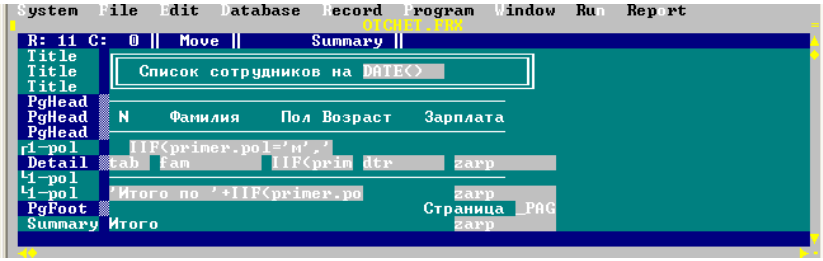


Рис.3.13. Планшет отчёта «Список сотрудников»

Для включения отчёта в программу необходимо набрать команду *report format otchet*.

Результаты работы генератора отчёта в режиме предварительного просмотра представлены на рис. 3.14.

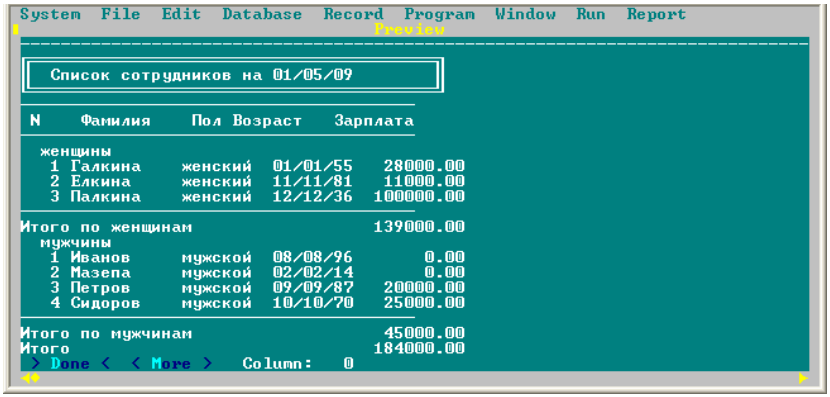


Рис.3.14. Отчёт «Список сотрудников»

3.4.2 Генератор меню

3.4.2.1 Принципы организации меню

Меню в FoxPro можно реализовать в 4 модификациях.

– световое

– клавишное
– кнопочное
– с помощью генератора меню

В FoxPro имеются 2 технологии построения меню:

– Fox - меню
– Dbase- меню

Fox – меню является частью программы. Внутри программы на экране появляется список из нескольких пунктов с предложением выбрать нужный пункт, введя его номер.

Dbase- меню – независимая часть программы, существующая всё время работы программы, состоит из элементов:

– определение меню
– активация меню
– деактивация меню (удаляется с экрана, но остаётся в памяти)
– удаление меню

Dbase- меню имеет 2 типа элементарных меню:

– вертикальное POPUP –меню
– горизонтальное BAR – меню

Определение меню состоит из 3 частей:

– описание меню
– описание пунктов меню
– описание реакций на выбор пунктов меню

Структура описания меню:

DEFINE MENU *<имя меню>*
AT LINE *<номер строки>*
IN WINDOW *<имя окна>* ***/IN SCREEN***
MESSAGE *<сообщение>*
KEY *<имя клавиши>*
MARK *<символ>*
COLOR SCHEME *<номер цветовой схемы>*

AT LINE <номер строки> – указывает номер строки, в которой появится меню, по умолчанию 0 (всего 25)

IN WINDOW <имя окна> /IN SCREEN – меню можно выводить в окно, которое должно быть заранее определено, либо на весь экран (по умолчанию на весь экран)

KEY<имя клавиши> – задаётся клавиша для вызова меню.

MARK <символ> – задаётся символ, появляющийся слева от выбранного пункта (по умолчанию ромб).

MESSAGE <сообщение> – выводится сообщение на русском языке, появляющееся посередине нижней строки.

COLOR SCHEME <номер схемы> по умолчанию 2

Структура описания пунктов меню

DEFINE PAD *<имя пункта>* ***OF*** *<имя меню>*
PROMPT *<назван пункта>* ***AT*** *< номер строки, номер столбца>*
KEY *<имя клавиши>*
MARK *<символ>*
COLOR SCHEME *<номер цветовой схемы>*
SKIP FOR *<условие>*
MESSAGE *<сообщение>*

PROMPT <назван пункта> AT <номер строки, номер столбца> – задаётся название пункта на русском языке, которое появится на экране в указанных позициях.

SKIP FOR <условие> – если условие истинно, то пункт доступен, если ложно, то пункт выдаётся приглушенным светом, и не доступен

Структура описания реакции на выбор пункта меню

ON SELECTION PAD *<имя пункта>* ***OF*** *<имя меню>* ***DO*** *<имя процедуры>*

Основная программа состоит из команд:

ACTIVATE MENU *<имя>* - активирует горизонтальное меню.
DEACTIVATE MENU – удаляет меню с экрана, но не из памяти.

3.4.2.2 Создание меню

CREATE MENU *<имя меню>*. - создание меню. Меню строится на базе системного меню _MSYSMENY и получает все его свойства. Базы данных должны быть предварительно открыты. При работе формируются четыре типа файлов. MNX и .MNT хранят образ меню, .MPR – текст программы, .MPX – откомпилированный файл.

На экране появится специальное окно, в котором находится отображение меню – планшет, представленный на рис. 3.15.

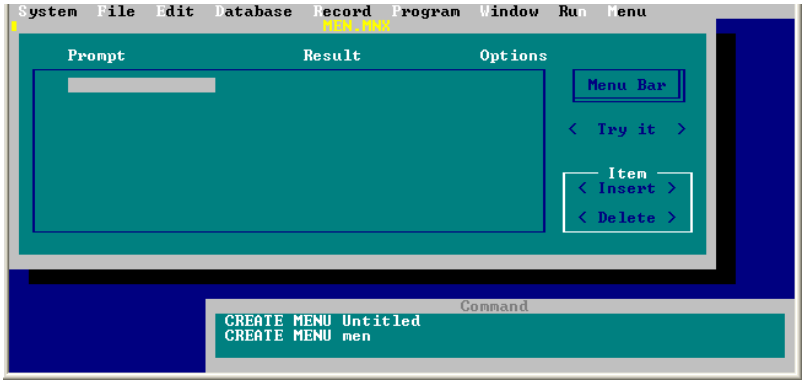


Рис. 3.15. Планшет генератора меню

Планшет состоит из четырёх областей:

Prompt для ввода собственного названия пункта

Result для определения действия при выборе пункта. Виды действий:

- Command - служит для определения команды, относящейся к опции горизонтального или вертикального меню. Появляется область ввода текста. В эту область вводится команда.
- Submenu - служит для создания подменю, присоединяемого к опции горизонтального или вертикального меню. Выполняется с помощью индикатора <Create>.
- Padname – используется пункт системного меню, которому можно дать собственное имя.
- Proc – за пунктом меню закрепляется процедура. Её текст вводится в окне редактора, с помощью индикатора < Create >

Options

- Shotcut - выбор горячей клавиши для вызова пункта и поясняющего текста
- Mark - символ пометки пункта,
- Skip for - условие, когда пункт недоступен. Условие вводится в окне построителя выражений.

4 бласть выдаёт текущий уровень меню, по которому можно подняться и опуститься, удерживая кнопку мыши.

- Try it – просмотр структуры меню

- Insert – вставка нового пункта
- Delete – удаление пункта

Рассмотрим пункт системного меню MENU, представленный на рис. 3.16.

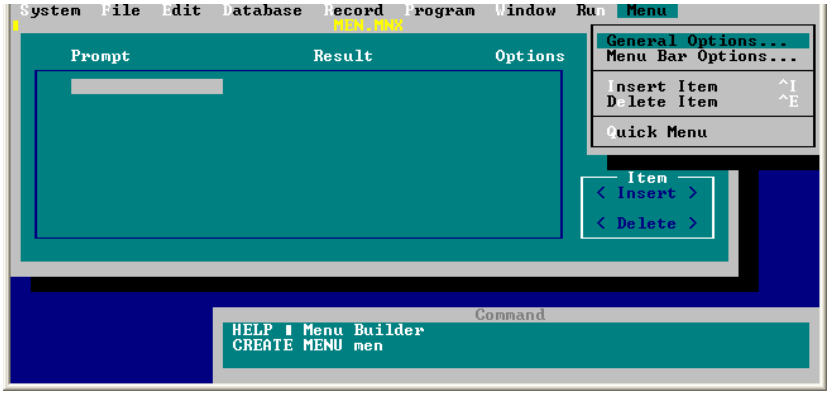


Рис. 3.16. Пункты меню работы с меню

GENERAL OPTIONS **-** активизирует диалоговое окно общих установок создания меню. Состоит из подпунктов**:**

- ***Locations*** **:** расположение меню: ***Replace*** **–** разместить вместо системного меню***; Append*** – разместить справа от системного меню; ***Before*** – разместить перед пунктом; ***After*** – разместить после пункта.

- ***Menu Code*** создание окон редактирования для набора команд: ***Setup*** –предваряющие запуск меню; ***Cleanup*** –завершающие меню.

MENU BAR OPTIONS. - определяет вертикальные меню.

INSERT/DELETE ITEM - вставка/удален. пунктов меню.

QUICK MENU **-** создаёт экспресс-меню, базирующееся на системе меню FoxPro, которое можно отредактировать, поменять названия, добавить или удалить пункты, изменить горячие клавиши.

Пример: Создать меню, замещающее системное меню и восстанавливающее его после окончания работы. Меню должно состоять из трёх пунктов:

Отчёт *Корректировка* *Выход*

Просмотр *Выход в FoxPro*

Редактирование *Выход в ОС*

Печать

Первый и третий пункты состоят из ниспадающих подменю. В меню предусмотреть открытие базы данных перед началом работы. Для защиты персональных данных от несанкционированного изменения предусмотреть отключение доступа к пункту <Корректировка> по выходным дням.

Порядок работы:

Создать меню с именем MEN командой create menu MEN. В область PROMPT внести названия пунктов горизонтального меню. Для первого пункт выбрать из области RESULT вариант SUBMENU, так как он состоит из ниспадающих подменю. Для пункта «Просмотр» выбрать из области RESULT варианта Proc, ввести в окно редактирования следующие команды:

clear
repo form otchet to file ot.txt
modi comm ot.txt noedit

Для пункта «Редактирование» - следующие команды:

clear
repo form otchet to file ot.txt
modi comm ot.txt

Для пункта «Печать» - следующие команды:

clear
repo form otchet to prn noconsole

Для второго пункта горизонтального меню «Корректировка» выбрать из области RESULT вариант COMMAND с записью команды в область ввода текста:

brow fields tab:h="таб.ном",fam:h="фамилия":r,zarp :h="зарплат"

Для защиты персональных данных от несанкционированного изменения предусмотреть отключение доступа к пункту <корректировка> по выходным дням, для этого в область OPTIONS, SKIP FOR внести условие

dow(date())=1

Для третьего пункта выбрать из области RESULT вариант SUBMENU, так как он состоит из ниспадающих подменю. Для

пункта «Выход в FoxPro» выбрать из области RESULT вариант COMMAND с записью команды в область ввода текста:
set sysmenu to default

Для пункта «Выход в ОС» выбрать из области RESULT вариант COMMAND с записью команды в область ввода текста:
quit

Для открытие базы данных перед началом работы в пункт меню GENERAL OPTIONS, MENU CODE, SETUP внести команды:

```
clear
If !used('primer')
    Select 0
    Use primer
Endif
```

Создание горизонтальных пунктов меню представлено на рис. 3.17.

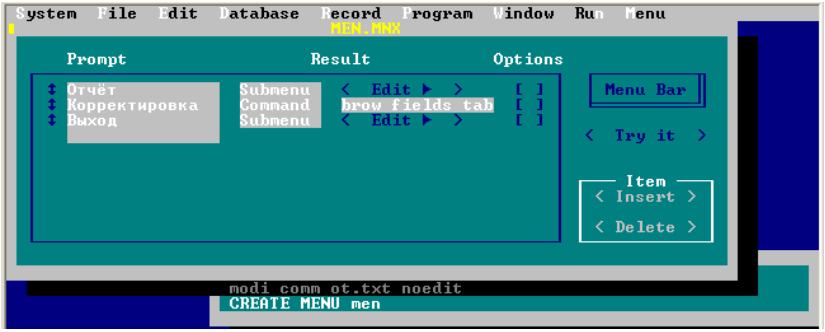


Рис. 3.17. Создание горизонтальных пунктов меню

После создания меню в планшете или его изменения необходимо сгенерировать исполняемую программу. Генерация выполняется через меню PROGRAMM, команда GENERATE. В результате будет сформирован файл с расширением .MPR, который можно выполнить как обычную программу FoxPro оператором *do men.mpr*. Меню активизируется нажатием F10 или ALT Для возврата к системному меню надо ввести команду *set sysmenu to default*.

Удобно использовать эту команду в специальном пункте меню «Выход», там же можно предусмотреть закрытие ненужных окон, отключить реакцию на нажатие клавиш и т.п.

Результаты работы созданного меню представлено на рис. 3.18.

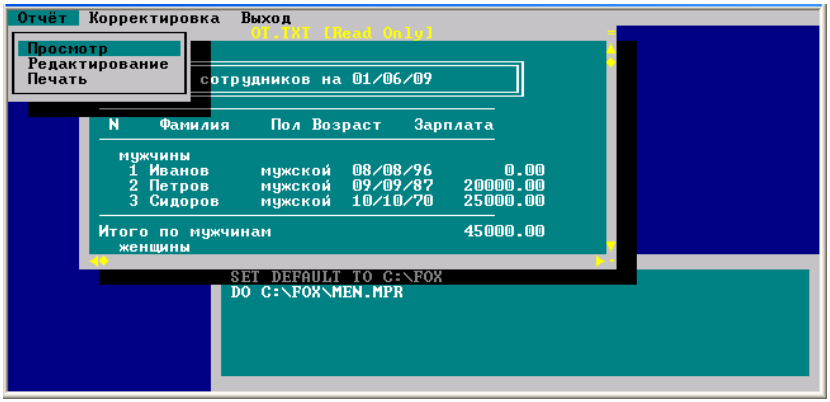


Рис. 3.18. Результаты работы меню

Текст программы, созданный Генератором меню в результате конструирования в планшете представлен листингом:

```
*
*
*  01/06/09                MEN.MPR          00:01:59
*
*
*
*  Author's Name
*
*  Copyright (c) 2009 Company Name
*  Address
*  City,      Zip
*
*  Description:
*  This program was automatically generated by
*                      GENMENU.
*


*
*
```

```
*╟                          Setup Code                          ╢
*╟                                                              ╢
*╚══════════════════════════════════════════════════════════════╝
*
clear
If !used('primer')
      Select 0
      Use c:\fox\primer.dbf
USE PRIMER.DBF
Endif

*╔══════════════════════════════════════════════════════════════╗
*╟                                                              ╢
*╟                        Menu Definition                       ╢
*╟                                                              ╢
*╚══════════════════════════════════════════════════════════════╝

SET SYSMENU TO
SET SYSMENU AUTOMATIC
DEFINE PAD _2jx002k0s OF _MSYSMENU PROMPT "Отчёт";
      COLOR SCHEME 3
DEFINE PAD _2jx002k0t OF _MSYSMENU ;
      PROMPT "Корректировка" COLOR SCHEME 3 ;
      SKIP FOR DOW(DATE())=1
DEFINE PAD _2jx002k0u OF _MSYSMENU PROMPT "Выход" ;
      COLOR SCHEME 3
ON PAD _2jx002k0s OF _MSYSMENU ACTIVATE POPUP отчёт
      ON SELECTION PAD _2jx002k0t OF _MSYSMENU ;
      brow fields tab:h="таб.ном.",;
      fam:h="фамилия":r,zarp:h="зарплата"
ON PAD _2jx002k0u OF _MSYSMENU ACTIVATE POPUP выход

DEFINE POPUP отчёт MARGIN RELATIVE SHADOW ;
      COLOR SCHEME 4
DEFINE BAR 1 OF отчёт PROMPT "Просмотр"
DEFINE BAR 2 OF отчёт PROMPT "Редактирование"
DEFINE BAR 3 OF отчёт PROMPT "Печать"
ON SELECTION BAR 1 OF отчёт ;
      DO _2jx002k0v ;
      IN LOCFILE("\FOX\MEN" ,"MPX;MPR|FXP;PRG" ,;
      "Where is MEN?")
ON SELECTION BAR 2 OF отчёт ;
      DO _2jx002k0w ;
```

```
      IN LOCFILE("\FOX\MEN" ,"MPX;MPR|FXP;PRG" ,;
      "Where is MEN?")
ON SELECTION BAR 3 OF отчёт ;
      DO _2jx002k0z ;
      IN LOCFILE("\FOX\MEN" ,"MPX;MPR|FXP;PRG" ,;
      "Where is MEN?")

DEFINE POPUP выход MARGIN RELATIVE SHADOW ;
      COLOR SCHEME 4
DEFINE BAR 1 OF выход PROMPT "Выход в FoxPro"
DEFINE BAR 2 OF выход PROMPT "Выход в ОС"
ON SELECTION BAR 1 OF выход ;
      SET SYSMENU TO DEFAULT
ON SELECTION BAR 2 OF выход quit
ON SELECTION MENU _MSYSMENU use primer

*
*
*                  Cleanup Code & Procedures
*
*
*close database
*
*
*  _2JX002K0V  ON SELECTION BAR 1 OF POPUP отчёт
*
*  Procedure Origin:
*
*  From Menu:  MEN.MPR,            Record:    5
*  Called By:  ON SELECTION BAR 1 OF POPUP отчёт
*  Prompt:     Просмотр
*  Snippet:    1
*
*
*
PROCEDURE _2jx002k0v
clear
repo form otchet to file ot.txt
modi comm ot.txt noedit
*
*
*  _2JX002K0W  ON SELECTION BAR 2 OF POPUP отчёт
*
*  Procedure Origin:
```

```
*╟──────────────────────────────────────────────────────────────────────╢
*│                                                                      │
*│ From Menu:  MEN.MPR,            Record:    6                         │
*│ Called By:  ON SELECTION BAR 2 OF POPUP отчёт                        │
*│ Prompt:     Редактирование                                           │
*│ Snippet:    2                                                        │
*│                                                                      │
*╙──────────────────────────────────────────────────────────────────────╜
*
PROCEDURE _2jx002k0w
clear
repo form otchet to file ot.txt
modi comm ot.txt
*╔══════════════════════════════════════════════════════════════════════╗
*│                                                                      │
*│ _2JX002K0Z  ON SELECTION BAR 3 OF POPUP отчёт                        │
*│                                                                      │
*│ Procedure Origin:                                                    │
*│                                                                      │
*│ From Menu:  MEN.MPR,            Record:    7                         │
*│ Called By:  ON SELECTION BAR 3 OF POPUP отчёт                        │
*│ Prompt:     Печать                                                   │
*│ Snippet:    3                                                        │
*│                                                                      │
*╚══════════════════════════════════════════════════════════════════════╝
*
PROCEDURE _2jx002k0z
clear
repo form otchet to prn noconsole
```

В листинге представлена полноценная программа меню, которая может быть использована для создания реальных приложений. Генератор автоматически формирует имена пунктов меню и процедур с использованием датчика случайных чисел. Программа осуществляет поиск вызываемых процедур по всему каталогу в одноимённых программах.

3.5 Работа со справочниками

При разработке прикладных систем часто приходится сталкиваться с повторяющимися данными, которые имеют большую длину. С цепью повышения скорости обработки данных такие данные целесообразно хранить в отдельном коротком файле, где каждой строке данных соответствует некоторый назначаемый ей код. Такой файл обычно называется кодификатором, справочником или словарем. Тогда вместо соответствующей строки данных в основной рабочий файл включается небольшое поле кода.

В связи с этим возникает задача организации взаимодействия файла-словаря и основного файла данных.

Пример. Для расчёта заработной платы сотрудников необходимы сведения об их должностных окладах и компенсационных выплатах от предприятия на детей, так как величина компенсационных выплат зависит от количества детей.

Добавим в базу PRIMER.DBF новые сведения о должности сотрудников и о количестве детей, которые находятся у них на иждивении. Название должности сотрудников имеет большую длину и может меняться, ввиду этого в основной базе будем хранить не сами данные, а их числовые коды. Также меняется и количество детей.

Добавим два новых поля KODDOL - числовое поле, размерность поля равна двум, хранит код должности, KOLDET - числовое поле, размерность поля равна двум, хранит количество детей у сотрудника.

Изменённая структура базы данных приведена на рис. 3.19.

Справочники хранятся в двух отдельных базах данных:

- база данных для хранения названий должностей и их окладов включает поля

KOD	Числовой тип длиной 2 разряда целых
DOL	Символьный тип длиной 20 символов
OKLAD	Числовой тип длиной 10 разрядов, из которых 2 дробных

- база данных для компенсационных доплат от предприятия в зависимости от количества детей включает поля

KOL	Числовой тип длиной 2 разряда целых
DOPL	Числовой тип длиной 10 разрядов, из которых 2

дробных

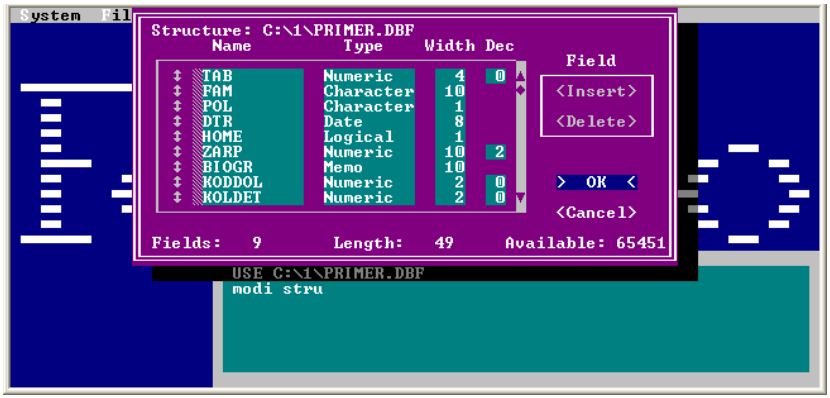


Рис. 19. Добавление полей в базу данных

Структура справочника названий должностей и их окладов представлена на рис. 3.20.

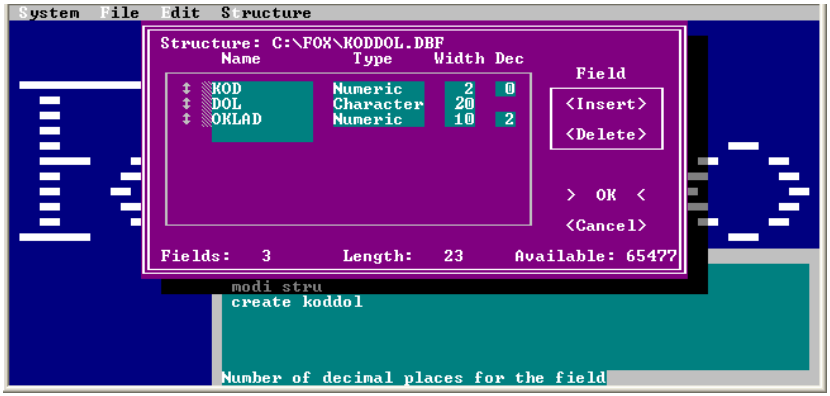


Рис. 3.20. Создание справочника должностей

Структура справочника доплат от предприятия в зависимости от количества детей представлена на рис. 3.21.

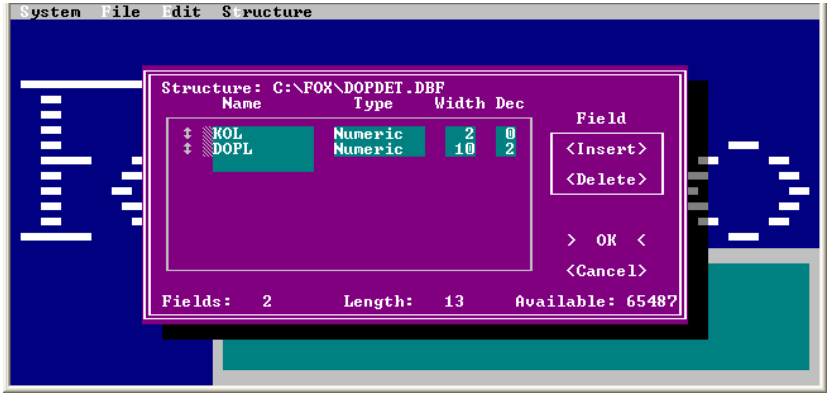


Рис. 3.21. Создание справочника детской компенсации от предприятия

Для решения задачи необходимо создать меню в генераторе меню, которое будет выглядеть следующим образом:

Расчет зарплаты *Справочники* *Выход*

Должности

Детские доплаты

Второй пункт состоит из двух ниспадающих подменю.

Пункт «Должности» предназначен для ввода новых должностей, корректировки наименований должностей и удаления ненужных. Коды должностей формируются автоматически. Они не предъявляются при работе с записями, а заменяются наименованиями должностей при работе. В тексте программы эту работу осуществляют процедура *sp_dol* и функция *kd.*

Пункт «Детские доплаты» предназначен для ввода и изменения суммы компенсационных доплат на разное количество детей. В программе эту работу осуществляет процедура *sp_dop*

Расчет зарплаты выполняется внутри процедуры с именем *rasc*, которая устанавливает связь со справочником должностей по коду должности и со справочником доплат на детей по полю количество детей и рассчитывает зарплату в виде суммы должностного оклада и надбавки на детей. Результаты расчёта выводятся на экран.

Планшет генератора меню заполняется следующим образом:

Command		*do rasc*
Справочники	*Submenu*	*<Create >*
Должности	*Command*	*do sp_dol*
Детские доплаты	*Command*	*do sp_dop*
Выход	*Proc.*	*<Edit >*

Процедура «Выход» набирается в редакторе текстов с использованием виртуальной кнопки <Edit > внутри меню. Её текст приведён ниже:

dropdead = .f.	&& временная переменная dropdead, организующая работу цикла обновления меню, при активации меню её значение равно *.t.*
set sysmenu to default	&& восстановление системного меню
on key label Enter	&& отключить реакцию на нажатие клавиши «Enter»
cancel	&& выход в FoxPro

В диалоговом окне общих установок создания меню «General Options», вызываемом из построителя меню «Menu» по команде «Menu Code» внутри опции < Setup... > составляем программу для определения начального кода запуска меню

close data	&& все посторонние базы должны быть закрыты
set proc to menu zarp.mpr	&& открывает файл процедур для поиска программ, вызываемых командой do и пользовательских функций, если они не найдены в текущей выполняемой программе.
select 0	&& выбор свободной рабочей области
use koddol	&& открытие справочника названий должностей и их окладов
index on kod tag kod	&& индексирование справочника по кодам
select 0	&& выбор свободной рабочей области
use dopdet	&& открытие справочника доплат от предприятия на детей
index on kol tag kol	&& индексирование справочника по кодам
select 0	&& выбор свободной рабочей области
use primer	&& открытие основной базы сотрудников
set relation to koddol	&& установка связи между базами данных

into koddol	
set relation to koldet into dopdet additive	&& установка связи между базами данных
set talk off	&& отключение выдачи подтверждающих сообщений
public dropdead dropdead = .F	&& определение глобальной временной переменной, которая инициализируются значением .f.
push menu _msysmenu	&& помещение системного меню в стек меню. Эта команда позволяет сохранить изменения, а затем восстановить системное меню
set sysmenu automatic	&& поддерживание системного меню в процессе выполнения программы

После составления меню, его необходимо сгенерировать, для этого в меню «Program» выполняется команда «Generate».

Далее надо составить основную программу приложения и сохранить её с именем «MainZarp.prg».

Текст программы:

public Home_path	&& определение пути доступа к процедурам и базам
Home_path= sys(5)+sys(2003)	&& sys(5)- выбор устройства по умолчанию sys(2003)- выбор текущей директории
aa="do "+ (Home_path+ "\menuzarp.mpr")	&& формирование символьной переменной для запуска меню из программы функцией подстановки
READ VALID myhandler()	&& главный цикл программы

Процедуры и функции набираются внутри опции <Edit> в диалоговом окне общих установок создания меню «General Options».

Текст их приведён ниже:

Процедура расчёта заработной платы сотрудников:

```
procedure rasc
 select primer
 brow fields fam:V=v_dol() :f:h="ФИО",;
            dol=koddol.dol :V=v_dol() :f:h="Должность",;
            koldet :h="Кол-во детей",;
```

zarplata =koddol.oklad+dopdet.dopl :h="Зарплата" ;
title "Ведомость (<Enter> - сменить должность)"

В процедуре осуществляется переход в область, где открыта основная база данных primer.dbf. Создаётся browse-окно, в котором предъявлено поле fam из основной базы с возможностью замены фамилии, поле dol из дочерней базы и реализована возможность через использование ключа «: V=v_dol()» при нажатии клавиши «Enter» подключить кодификатор должностей, подставить вместо кода наименование должности или выполнить замену должности на необходимую. Кроме того, предъявлено поле koldet из дочерней базы с возможностью замены количества детей, находящихся на иждивении и вычисляемое поле zarplata. Вычисление зарплаты выполняется суммированием оклада сотрудника, выбранного из кодификатора должностей и компенсационных доплат на указанное количество детей Текст функций «v_dol()» и «vib_dol» приведён ниже:

function v_dol	&& название функции
if lastk()=13	&& проверка нажатия клавиши «Enter»
select koddol	&& переход в область, где открыта дочерняя база
on key label Enter do vib_dol	&& выполнение функции замены должности при нажатии клавиши «Enter»
BROWSE fields ;	&& формирования browse – окна для выбора наименования должности из кодификатора без редактирования их
dol:h='должности';	
title 'должность выбор - <Enter>' noed	
on key label Enter	&& при повторном нажатии клавиши «Enter» закончить проверку
endif	
function vib_dol	&& название функции
replace primer.koddol ;	&& замена кода должности на код выбранной должности
with koddol.kod	
on key label Enter	&& при нажатии клавиши «Enter» закрытие окна
release window	

Процедура работы со справочником должностей:

procedure sp_dol

```
select koddol
BROWSE field dol:h='должность':f :v= kd(),oklad:h='оклад' ;
  title 'должность <CTRL+N> - добавить <CTRL+T> - удалить'
count for deleted() to n
if n>0
  pack
endif
```

В процедуре осуществляется переход в область, где открыта дочерняя база данных koddol.dbf. Создаётся browse-окно, в котором предъявлено поле dol и реализована возможность через использование ключа «:V= kd(),» ввести наименование новой должности в кодификатор, автоматически вычислить следующий код для новой должности или удалить имеющуюся должность и соответственно удалить из кодификатора её код. Кроме того, предъявлено поле oklad с возможностью изменения его.

Текст функции «kd» приведён ниже:

```
func kd                  && название функции
if empty(kod)            && проверка наличие кода
n=recno()                && запоминание номера записи по-
                         последнего существующего кода
calc max(kod) to m       && вычисление и запоминание мак-
                         симального номера кода из уже суще-
                         ствующих
replace record n kod ;   && увеличение номера на единицу для
with m+1                 создания следующего кода
endif                    && конец цикла
```

Процедура работы со справочником компенсационных доплат на указанное количество детей:

```
procedure sp_dop
 select dopdet
 BROWSE field kol:h='количество детей',dopl:h='доплата'  ;
 title 'доплата <CTRL+N> - добавить <CTRL+T> - удалить'
```

В процедуре осуществляется переход в область, где открыта дочерняя база данных dopdet.dbf. Создаётся browse-окно, в котором предъявлено поле kol и поле dopl с возможностью их изменения и добавления.

3.6 Язык запросов SQL

Язык SQL (Structured Query Language) символизирует собой структурированный язык запросов и является стандартным языком, который используется в большом количестве различных видов компьютерных сред. Стандартный язык позволяет пользователям, знающим один набор команд, использовать их для создания, нахождения, изменения и передачи информации - независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ. Стандарт SQL определяется ANSI (Американским национальным институтом стандартов) и в данное время также принимается ISO (Международной организацией по стандартизации).

SQL предназначен для определения структуры баз данных, манипулирования данными в реляционных базах данных, Команды SQL могут непосредственно включаться в программы наряду с собственными командами FoxPro. Язык SQL в FoxPro представлен командами:

– создание баз данных – команда CREATE
– дополнение базы – команда INSERT
– формирование запросов из базы данных – команда SELECT.

3.6.1 Создание баз данных

CREATE DBF *<имя файла>*. ***(****<имя поля> <тип> [(<размер>[, <точность>]) [, <имя поля2> ...]]****)*** - команда создает новую базу данных с указанным именем *<имя файла>*. Для каждого поля задаются его имя, тип (одной из букв C, N, D, M, F, L), длина и число десятичных разрядов. Длина и точность не задаются для типов дата (D), логический (L) и примечаний (M), а точность - для символьного типа (C). Все требования к описанию полей базы данных - стандартные. Созданная база сразу открывается.

Пример. Командой SQL создать базу данных с именем PRIMER.DBF

Табельный номер TAB	Числовой тип длиной 4 разряда целых

работника		
Фамилия	FAM	Символьный тип длиной 10 символов
Пол	POL	Символьный тип длиной 1 символ (М или Ж)
Дата рождения	DTR	Тип дата длиной 8
Наличие жилья	HOME	Тип логический (наличие жилья - .T. отсутствие - .F.)
Зарплата	ZARP	Числовой тип длиной 10 разрядов, из которых 2 дробных
Биография	BIOGR	Тип текстовый выбран ввиду непредсказуемости длины данных

Создание базы выполняется командой:

create dbf primer (tab N(4),fam C(10),pol C(1),dtr D, home L,zarp N(10,2),biogr M,koddol N(2),koldet N(2))

Результаты работы команды представлены на рис. 3.22.

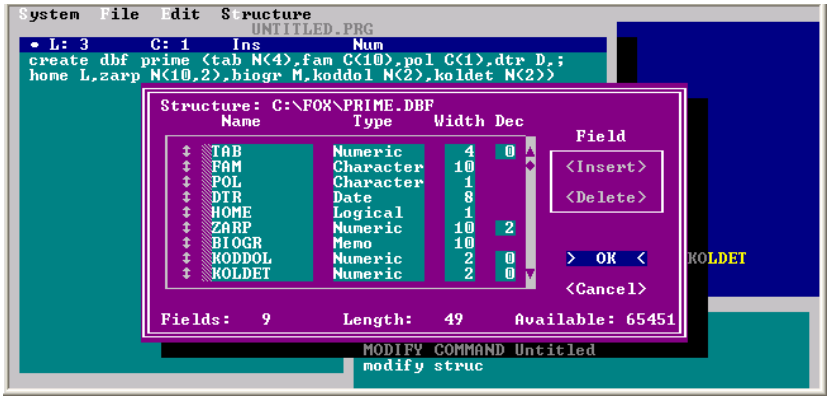


Рис. 3.22. Создание базы данных командой SQL

3.6.2 Дополнение баз данных

INSERT INTO *< имя файла > [(<поле1> [, <поле2> [,'...]])]* ***VALUES*** *(<выр1> [, <выр2> [, ...]])* - заполнение базы данных и дополнение базы новыми записями.

Команда добавляет записи в конец существующего <файла базы данных>, используя <выражения>, перечисленные после сло-

ва VALUES. Если опущены имена полей, <выражения> будут записываться в последовательные поля <базы данных> в соответствии с ее структурой. Дополняемая база данных может быть и не открыта к моменту выполнения команды, однако после этого она останется открытой и активной.

Пример. Заполнить базу данных PRIMER на языке SQL. Результат работы команд представлен на рис. 3.23.

insert into primer (tab, fam, pol, dtr, home,zarp, biogr, koddol,koldet);
values (222,'Красников','м',{01.01.2001},.t.,0,'дитя',0,0)
insert into prime (tab, fam, pol, dtr, home,zarp, biogr, koddol,koldet);
values (333,'Красникова','ж',{01.01.1967},.t.,0,'работает на заводе',10,1)

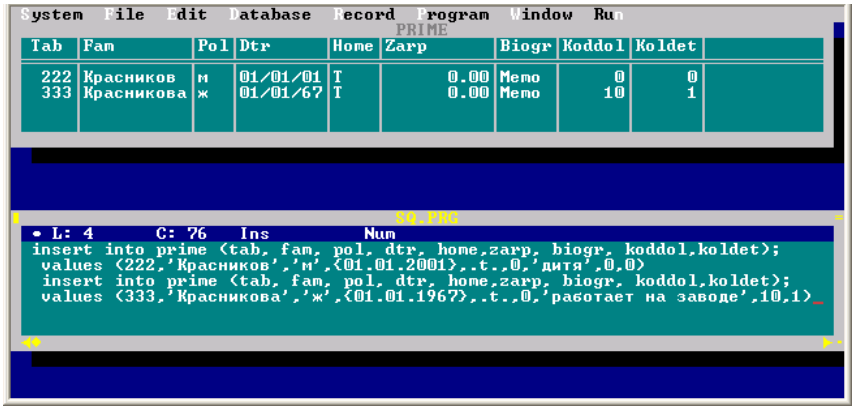


Рис.3.23. Добавление в базу данных двух записей командой SQL

3.6.3 Формирование запросов из базы данных

Запросы формируются с помощью команды SELECT, которая сама открывает нужные базы данных и индексные файлы, группирует данные по различным условиям, упорядочивает желаемым образом, связывает базы между собой, выбирает данные по условиям, представляет в требуемое место и допускает включение в себя других внутренних команд SELECT.

Логическая структура команды:

SELECT *<что выводится>* ***FROM*** *<откуда>* ***INTO*** *<куда>*
WHERE *<каким условиям должно отвечать>*
GROUP BY *<колонки, по которым выполняется группирование>*
HAVING *<условие группирования записей в одну строку>*
ORDER BY *<в каком порядке выводить данные>*

Полная структура команды:

SELECT [ALL / DISTINCT] *<искомый_элемент>* ***[AS*** *<имя столбца>],<искомый_элемент>* ***[AS*** *< имя столбца >]...]* ***FROM*** *<имя файла> [,<имя файла > ...]*
[[INTO *<получатель>] /[****TO FILE*** *< имя файл>* ***[ADDITIVE] /TO PRINTER]] [NOCONSOLE] [PLAIN] [NOWAIT]***
[WHERE *<условие связи> [AND <условие связи>...]*
[AND/OR <условие отбора> [AND/OR <условие отбора>...]]]
[GROUP BY *<столбец> [, <столбец> ...]]*
[HAVING *<условие отбора>]* ***[ORDER BY*** *<столбец>*
[ASC/DESC] [,*<столбец>* ***[ASC/DESC]...]]***

Рассмотрим опции команды.

Опция ALL - по умолчанию выводятся все строки результата запроса. Для исключения дубликатов строк указывается ключевое слово DISTINCT. В команде SELECT можно указывать DISTINCT только однократно.

Каждый «искомый_элемент» может быть полем базы данных, константой или выражением, которое может содержать пользовательскую функцию. Для использования с «искомыми_элементами» доступны функции:

– ***AVG****(<искомый_элемент>)* - среднее значение по столбцу числовых данных.

– ***COUNT(****<искомый_элемент>****)*** **-** число искомых элементов в столбце.

– ***COUNT(********)*** - определяет число строк в выводе запроса**.**

– ***MIN****(<искомый_элемент>)* - определяет наименьшее значение искомого элемента. в столбце.

– ***MAX****(<искомый_элемент>)* - определяет наибольшее значение искомого элемента в столбце.

– ***SUM****(<искомый_элемент>)* - определяет суммарный итог по столбцу числовых данных.

Опция AS <имя_столбца> является необязательной фразой, специфицирует заголовок столбца в выводе результата запроса.

В опции FROM <имя файла> [,<имя файла > ...] - перечисляются все базы, которые содержат данные, извлекаемые из них при поиске.

Опция INTO <получатель> указывает куда надо отправить результаты выборки. Это может быть:

– ***ARRAY*** **-** вновь создаваемый временный двумерный массив

– ***CURSOR*** **-** временная область памяти

– ***DBF*** - новая база данных с указанным именем

Если же фраза INTO не использована, то результат запроса появится на экране.

Опция TO FILE <файл> [ADDITIVE] / TO PRINTER посылает выборку в текстовый файл или на принтер. Если используется опция ADDITIVE, то выборка будет добавлена в конец существующего файла без его перезаписи.

Опция NOCONSOLE указывает, что выборка не выдается на экран.

Опция PLAIN указывает, что заголовки столбцов не выдаются на экран.

Опция NOWAIT указывает, что не делаются паузы при заполнении экрана.

Опции NOCONSOLE, PLAIN, NOWAIT используются только при выводе данных на экран.

Опция WHERE <условие связи> [AND <условие связи>...] [AND/OR <условие отбора> [AND/OR <условие отбора>...]] - представляет собой критерий отбора данных из баз.

<условие связи> - применяется в случае, если выборка делается более чем из одной базы данных, и указывается критерий, которому должны отвечать поля из разных баз. Разрешается использовать знаки отношения =, #, ==, >, >=, <, <=. Допускается задание нескольких критериев, соединенных знаком AND.

<условие отбора> - строится аналогично, но из выражений только для одной базы, и допускается использование логических операторов OR и NOT.

Условия, кроме любых функций FoxPro, могут содержать следующие операторы SQL:

– ***LIKE*** - позволяет построить условие сравнения по шаблону, где символ «_» указывает единичный неопределенный символ в строке, «%»- любое количество неопределенных символов в строке. Эти символы аналогичны символам маски «?» и «*» в описании функции FoxPro. Формат оператора: <выражение> LIKE <шаблон>

– ***BETWEEN*** - проверяет, находится ли выражение в указанном диапазоне. Формат оператора: <выражение> BETWEEN <нижнее знач.> AND <верхнее знач.>

– ***IN*** - проверяет, находится ли выражение, стоящее слева от слова IN, среди перечисленных справа от него (аналогично функции INLIST). Формат оператора: <выражение> IN (<выражение>,<выражение>,...)

Все указанные операторы можно комбинировать с помощью операндов OR, AND, NOT и скобок.

Опция GROUP BY <столбец> [, <столбец> ...] указывает столбцы, по которым производится группирование выходных данных. Все записи базы, для которых значения столбцов совпадают, отображаются в выборке единственной строкой. Группирование удобно для получения некоторых сводных характеристик группы.

Опция HAVING <условие отбора> - задает критерий отбора данных в каждую сформированную в процессе выборки группу. Она используется совместно с GROUP BY, может включать множество условий фильтрации, которые соединяются операндами AND, OR или NOT.

Опция ORDER BY <столбец> [ASC/DESC] сортирует результаты запроса на основании значений данных в столбце. Фраза ASC задаёт упорядочение по заданному столбцу. Фраза DESC вызывает упорядочение по убыванию значения соответствующего столбца. По умолчанию сортировка выполняется по возрастанию.

Пример. Из базы данных PRIMER выбрать сотрудников, зарплата которых находится в диапазоне от 3000 рублей до 60000 рублей. Выдать список на экран в виде двух столбцов: первый столбец содержит фамилию сотрудника, заголовок столбца на русском языке, второй столбец содержит значение зарплаты сотрудника, заголовок столбца на русском языке.

select fam as 'фамилия' ,zarp as 'зарплата' from primer ;
where zarp between 3000 and 60000

Пример. Из базы данных PRIMER выбрать старшего по возрасту сотрудника. Напечатать его фамилию и год рождения.

select fam as 'фамилия',min(year(dtr)) as 'год_рождения'
from primer

Пример. Из базы данных PRIMER распечатать список сотрудников и их должностей, заменив код должности на наименование должности. Для этого задания необходимо связать две базы данных PRIMER и KODDOL. Заголовки столбцов выдать на русском языке.

select fam as 'фамилия', dol as 'должность' from primer, koddol;
where primer. koddol=koddol. kod

Пример. Из базы данных PRIMER распечатать список сотрудников в должности «директор». Для этого задания необходимо связать две базы данных PRIMER и KODDOL. Заголовки столбцов выдать на русском языке.

select fam as 'фамилия', dol as 'должность' from primer, koddol;
where primer. koddol=koddol. kod and dol='директор'

Пример. Из базы данных PRIMER распечатать список сотрудников, и величину их должностного оклада. Для этого задания необходимо связать две базы данных PRIMER и KODDOL, отсортировать список в порядке возрастания их окладов. Заголовки столбцов выдать на русском языке.

select fam as 'фамилия', dol as 'должность, oklad as 'оклад' from
primer, koddol where primer. koddol=koddol. kod order by oklad

Пример. Из базы данных PRIMER распечатать список сотрудников и указать отсутствие или наличие жилья. Заголовки столбцов выдать на русском языке.

select fam as 'фамилия' , iif (home,'имеется ','отсутствует');
as 'нал_жилья' from primer

Пример. Из базы данных PRIMER распечатать список сотрудников, наименования должностей, величину их окладов, количество детей на их иждивении, доплаты от предприятия на содержание детей и зарплату сотрудников. Для этого задания необходимо связать три базы данных PRIMER, KODDOL и DOPDET, отсортировать список по фамилиям в алфавитном порядке. Заголовки столбцов выдать на русском языке.

select fam as 'фамилия', dol as 'должность', oklad as 'оклад',;
koldet as 'количество_детей', dopl as 'доплата',;

oklad+dopl as 'зарплата';
from primer, koddol, dopdet order by fam;
where primer. koldet=dopdet. kol and primer. koddol=koddol. Kod

Результат работы команды представлены на рис. 3.24.

Рис. 3.24. Выборка из базы данных выполненная командой SQL

Пример. В базе данных PRIMER подсчитать отдельно количество женщин и мужчин и их суммарную зарплату.
select count(*), sum(zarp) from primer group by pol

Пример. Из базы данных PRIMER распечатать список сотрудников, которые обучаются на заочном отделении института.
select fam as 'фамилия' from primer
where biogr like '%Поступил%институт%'

Контрольные вопросы

1. Системный интерфейс FoxPro. Назначение пунктов меню.
2. Структура команд установки параметров операционной среды FoxPro.
3. Структура команд работы с данными.
4. Создание, дополнение и изменение структуры файла базы данных. Типы полей.
5. Заполнение базы данных исходными данными.
6. Редактирование данных с помощью команды BROWSE.
7. Внесение изменений в данные.
8. Фильтрация данных, поиск данных.
9. Индексирование и сортировка данных в базе данных.
10. Средства для получения числовых характеристик: суммы, количества, среднего значения.
11. Организация связи баз данных.
12. Операторы присваивания, ввода- вывода.
13. Оператор условия, выбора.
14. Организация циклов, операторы цикла.
15. Строковые функции.
16. Функции работы с датами.
17. Функции преобразования типов.
18. Назначение генераторов и их виды.
19. Приёмы работы в планшете генератора.

ГЛОССАРИЙ

Атрибут - элемент данных, содержащий информацию об объекте.

База данных (БД)- совокупность взаимосвязанных данных, используемых одним или несколькими пользователями и хранящихся с регулируемой избыточностью.

Банк данных - система, представляющая определенные услуги по хранению и поиску данных определенной группе пользователей по определенной тематике.

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

Данные - вид информационного ресурса, отличающийся высокой степенью форматированности в отличие от более свободных структур, характерных для речевой, текстовой и визуальной информации.

Запись логическая - поименованная совокупность данных, рассматриваемая пользователем как одно целое.

Запись физическая - совокупность данных записываемых (считываемых) одним блоком.

Запросы —объекты, которые служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию. С помощью запросов можно выполнять преобразования данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое наполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое.

Идентификатор - атрибут, значения которого однозначно определяют экземпляры объекта предметной области.

Ключ - элемент данных, используемый для идентификации или определения местоположения записи.

Модель данных - формально определенная структура, используемая для представления данных на логическом и физическом уровнях.

Концептуальный - определение, относящееся к обобщенному представлению данных, независимому от СУБД.

Логический - определение, относящееся к представлению или описанию данных, не зависящему от запоминающей среды или вычислительной системы.

Независимость данных - возможность изменения логической и физической структуры БД без изменения представлений пользователей.

Объект - термин, обозначающий факт, лицо, событие, предмет, о котором могут быть собраны данные.

Отчеты - по своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на принтер. В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов.

Распределенная база данных - единая БД, представленная в виде отдельных (возможно, избыточных и перекрывающихся) разделов на разных вычислительных средствах.

Связь - функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (родительской) таблице - первичный, второй — внешний ключ — во внешней (дочерней) таблице, как правило, первичным не является и образует связь "один ко многим" (1:N). В случае первичного внешнего ключа связь между таблицами имеет тип "один к одному" (1:1). Информация о связях сохраняется в базе данных.

Словарь данных - набор обобщенных описаний данных БД, обеспечивает логически централизованное хранение метаданных.

Система управления базой данных (СУБД) - совокупность программных средств, обеспечивающих управление БД на всех уровнях.

Схема - описание логической структуры данных, специфицированное на языке описания данных и обрабатываемое СУБД.

Сущность - примитивный объект данных, отображающий элемент предметной области (человек, место, вещь и т.д.).

Система баз данных - совокупность СУБД, прикладного программного обеспечения, базы данных, операционной системы и технических средств, обеспечивающих информационное обслуживание пользователей.

Таблицы — это основные объекты любой базы данных. Во–первых, в таблицах хранятся все данные, имеющиеся в базе, а во–вторых, таблицы хранят и структуру базы (поля, их типы и свойства). Таблица предназначена для хранения данных в виде записей (строк) и полей (столбцов). Обычно каждая таблица используется для хранения сведений по одному конкретному вопросу.

Транзакция - процесс изменения файла или базы данных, вызванный передачей одного входного сообщения.

Указатель - идентификатор, который ведет к заданной записи из какой-то другой записи в базе данных.

Файл - совокупность аналогично построенных хранимых записей фиксированной или переменной длины одного типа.

Формы — средства для ввода и просмотра данных.

Хранимая запись - совокупность связанных элементов данных, соответствующая одной или нескольким логическим записям и содержащая все необходимые служебные данные.

Целостность данных - устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей.

Элемент данных - наименьшая единица данных, имеющая смысл при описании информации; наименьшая единица поименованных данных.

Язык базы данных - общий термин, относящийся к классу языков, которые используются для определения и обращения к базам данных.

Язык манипулирования данными - командный язык, обеспечивающий доступ к содержимому БД и его обработку.

Язык описания данных - предназначен для описания данных на концептуальном, логическом и физическом уровнях на основе соответствующих схем.

Язык запросов (SQL) - высокоуровневый язык манипулирования данными, обеспечивающий взаимодействие пользователей с БД.

Null - значение поля таблицы, показывающее, что информация в данном поле отсутствует. Разрешение на возможность существования значения Null может задаваться для отдельных полей таблицы.

СПИСОК ЛИТЕРАТУРЫ

1. *Голицына О.Л.* Базы данных. М.: Форум, Сер: Профессиональное образование. 2006.
2. *Гончаров А.* FoxPro в примерах. Версия 2.5,2.6,3.0 — СПб.:Питер, 1995. — 160 с.: ил.
3. *Золотова С. И.* Практикум по Access - М: Изд. - Финансы и статистика, 2008
4. *Марков А.С.* Базы данных. Введение в теорию и методологию. - М.: Финансы и статистика, 2006.
5. *Михеева Е. В.* Информационные технологии в профессиональной деятельности - М: Изд. -Академия, 2008
6. *Попов А. А.* FoxPro 2.5/2.6. Создание приложений для FOXPRO 2.5/2.6 в DOS и WINDOWS. —М.: ДЕСС и ZmeiKa. 2001. — 672 с.
7. *Хомоненко А.* Базы данных: Учебник для высших учебных заведений. — М.: Бином. Лаборатория знаний, 2006.
8. *Хомоненко А. Д., Цыганков В. М., Мальцев М. Г.* Базы данных: Учебник для высших учебных заведений / Под ред. проф. А. Д. Хомоненко, — 4-е изд., доп. и перераб. — СПб.: КОРОНА принт, 2004. — 736 с.

ОГЛАВЛЕНИЕ

УЧЕБНОЕ ИЗДАНИЕ

Баканов Максим Владимирович
Романова Вера Васильевна,
Крюкова Татьяна Петровна

БАЗЫ ДАННЫХ.
СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Учебное пособие

Для студентов вузов

Оригинал-макет изготовлен в редакционно-издательском отделе
Кемеровского технологического института
пищевой промышленности
650056, г. Кемерово, б-р Строителей, 47

ПЛД №44-09 от 10.10.99
Отпечатано в лаборатории множительной техники
Кемеровского технологического института
пищевой промышленности
650010, г. Кемерово, ул. Красноармейская, 52