МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Рубцовский индустриальный институт (филиал)
ФГБОУ ВПО «Алтайский государственный технический университет им. И.И. Ползунова»

Е.А. ДУДНИК

БАЗЫ ДАННЫХ В СУБД Visual FoxPro

Учебно-методическое пособие для студентов, обучающихся по направлению «Информатика и вычислительная техника» дневной формы обучения

Рекомендовано Рубцовским индустриальным институтом (филиал) ФГБОУ ВПО «Алтайский государственный технический университет им. И.И. Ползунова» в качестве учебного пособия для студентов, обучающихся по направлению подготовки «Информатика и вычислительная техника»

Рубцовск 2015

УДК 681.3

Дудник Е.А. Базы данных в СУБД Visual FoxPro: Учебно-методическое пособие для студентов, обучающихся по направлению «Информатика и вычислительная техника» дневной формы обучения /Рубцовский индустриальный институт. - Рубцовск, 2015. - 99 с.

Учебно-методическое пособие содержит краткий курс лекций, задания к практическим занятиям и лабораторным работам, а также рекомендации к их выполнению. Для самостоятельной работы студентов по основному курсу «Базы данных» представлен перечень тем курсовых работ. В пособии изложены теоретические и практические основы работы с системой управления базы данных в инструментальной объектно-ориентированной среде Visual FoxPro. Приведены основные команды и конструкции СУБД Visual FoxPro, изложены основные методы работы для создания проекта. Предназначено для студентов, изучающих дисциплину «Базы данных» в среде Visual FoxPro.

Рассмотрено и одобрено
на заседании НМС РИИ.
Протокол № 4 от 21.05.2015

Рецензент:
к.ф.-м.н. Г.А. Обухова

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4
1. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ 4
2.СРЕДА VISUAL FOXPRO 8
3. ОРГАНИЗАЦИЯ И ХРАНЕНИЕ ДАННЫХ В VISUAL FOXPRO 10
4. КОМАНДНЫЕ ФАЙЛЫ 43
5.СОЗДАНИЕ ФОРМ В VISUAL FOXPRO 50
6. ПОИСК В БАЗЕ ДАННЫХ 51
7.РАБОТА С НЕСКОЛЬКИМИ БАЗАМИ 53
8.УПРАВЛЕНИЕ ДАННЫМИ 55
9.ФОРМИРОВАНИЕ ОТЧЕТОВ 58
10.ОСНОВЫ ЯЗЫКА СТРУКТУРИРОВАННЫХ ЗАПРОСОВ 64
11.. ОРГАНИЗАЦИЯ МЕНЮ 67
ЛАБОРАТОРНЫЙ ПРАКТИКУМ 70
ТЕМЫ КУРСОВЫХ РАБОТ ПО ВАРИАНТАМ 89
ВАРИАНТЫ ЗАДАНИЙ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ 98
СПИСОК ЛИТЕРАТУРЫ 99

ВВЕДЕНИЕ

Методическое пособие предназначено для самостоятельной работы студентов по дисциплине «Базы данных».

Целью работы является познакомить с теорией реляционных баз данных и помочь быстро освоить мощную автоматизированную среду системы управления базы данных (СУБД) Visual FoxPro для разработки приложений.

Одной из основных функций баз данных является поиск, хранение, упорядочение и индексация информации. Как и в библиотечной картотеке, не нужно просматривать половину архива, чтобы найти нужную запись. Не все базы данных создаются на основе одних и тех же принципов, но традиционно в них применяется идея организации данных в виде записей. Каждая запись имеет фиксированный набор полей. Записи помещаются в таблицы, а совокупность таблиц формирует базу данных.

Для работы с базой данных необходима СУБД (система управления базами данных), т.е. программа, которая берет на себя все заботы, связанные с доступом к данным. Она содержит команды, позволяющие создавать таблицы, вставлять в них записи, искать и даже удалять записи.

1. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ играют особую роль в современном мире, все, с чем мы ежедневно встречаемся, зарегистрировано в той или иной базе данных, являющейся одним из распространенных средств хранения информации. Хорошо, грамотно организованная база данных решает несколько параллельных и достаточно независимых задач:

Работа с базой данных – это система интерфейсов: между программой и пользователем, между различными автономными базами, обменивающимися данными.

Работа с базой данных – это сложные конструкции данных, которые нужно сохранять, специально заботиться о компактности, защищённости от несанкционированного доступа, информационной достаточности для решаемой задачи.

СУБД – это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения, редактирования, визуализации информации. **Под визуализацией данных базы** понимается отбор отображений в соответствии с запросом, их упорядочение, оформление и последующие выдачи на устройства ввода-вывода или по каналам связи. Основной задачей СУБД является получение нужной информации из большого объема всей базы данных.

СУБД включает, как правило, следующие элементы:

Организация интерфейса пользователем;

Организация запросов пользователем;

Организация файловых систем;

Организация ввода-вывода.

Общение пользователя с базой данных заключается в запросах пользователя на выполнение нужных ему операций над данными и выдаче базой ответов на запросы. Операции над данными можно разбить на **систему классов операций**:

1. Ввод новых данных.
2. Редактирование имеющихся данных.
3. Просмотр данных.
4. Уничтожение данных.

Каждый класс может быть достаточно сложным и состоять из целой группы операций или даже из группы классов операций.

Например:

Просмотр данных:

– Печать данных.

– Вывод данных на экран монитора.

Вместе с рациональной организацией хранения данных такие средства образуют банк данных. **Банк данных** включает:

– базу данных, то есть специально организованный фонд с минимальной избыточностью данных;

– набор программных средств для ведения базы данных (СУБД);

– инструкции по заполнению базы данных и использованию программных средств.

– **Классы СУБД** различаются по способу организации информационного фонда:

– с иерархическим информационным фондом;

– с сетевой информационной моделью;

– реляционные СУБД.

При **реляционном** описании предметной области данные рассматриваются как совокупность множества элементов, между которыми существуют отношения. Описываемое множество разбивается на подмножества – домены, которым присваиваются имена (атрибуты). Конечное множество атрибутов определяет тип отношений и называется кортежем. Множество неравных между собой отношений составляет информационный фонд. Отношения могут быть заданы таблицами (схемами). Строки таблиц являются кортежами (записями), колонки определяют домены. Имя колонки – атрибут (имя поля записи).

Между отношениями можно задать **операции преобразований:**

сложение таблиц (объединение);

умножение (таблица из пересечения двух других);

вычитание (как вычитание множеств).

Существует и ряд других операций, например проектирование, соединение, выделение подсхем из таблицы-схемы.

Для реляционных моделей развит математический аппарат – реляционная алгебра и реляционное исчисление.

Применение законов и правил реляционной алгебры упорядочивает процесс создания программного обеспечения СУБД (Система управления базами данных).

Иерархические и сетевые модели предполагают наличие связей между данными, имеющими какой-либо общий признак. Такие связи могут быть отражены в виде дерева-графа, где возможны только односторонние связи от старших к младшим, обеспечивается прямой доступ к данным, но если только все возможные вопросы учтены.

Иерархические базы данных

Иерархические базы данных поддерживают древовидную организацию информации. Связи между записями выражаются в виде отношений предок/потомок, а у каждой записи есть ровно одна родительская запись. Это помогает поддерживать ссылочную целостность. Когда запись удаляется из дерева, все ее потомки также должны быть удалены.

Иерархические базы данных имеют централизованную структуру, т.е. безопасность данных легко контролировать. К сожалению, определенные знания о физическом порядке хранения записей все же необходимы, так как отношения предок/потомок реализуются в виде физических указателей из одной записи на другую.

Это означает, что поиск записи осуществляется методом прямого обхода дерева. Записи, расположенные в одной половине дерева, ищутся быстрее, чем в другой. Отсюда следует необходимость правильно упорядочивать записи, чтобы время их поиска было минимальным. Это трудно, так как не все отношения, существующие в реальном мире, можно выразить в иерархической базе данных. Отношения "один ко многим" являются естественными, но практически невозможно описать отношения "многие ко многим" или ситуации, когда запись имеет несколько предков. До тех пор пока в приложениях будут кодироваться сведения о физической структуре данных, любые изменения этой структуры будут грозить перекомпиляцией.

Сетевые базы данных

Сетевая модель расширяет иерархическую модель, позволяя группировать связи между записями в множества. С логической точки зрения связь — это не сама запись. Связи лишь выражают отношения между записями. Как и в иерархической модели, связи ведут от родительской записи к дочерней, но на этот раз поддерживается множественное наследование.

Следуя спецификации CODASYL, сетевая модель поддерживает DDL (Data Definition Language — язык определения данных) и DML (Data Manipulation Language — язык обработки данных). Это специальные языки, предназначенные для определения структуры базы данных и составления запросов. Несмотря на их наличие, программист по-прежнему должен знать структуру базы данных.

В сетевой модели допускаются отношения "многие ко многим", а записи не зависят друг от друга. При удалении записи удаляются и все ее связи, но не сами связанные записи.

В сетевой модели требуется, чтобы связи устанавливались между существующими записями во избежание дублирования и искажения целостности. Данные можно изолировать в соответствующих таблицах и связать с записями в других таблицах.

Программисту не нужно заботиться о том, как организуется физическое хранение данных на диске. Это ослабляет зависимость приложений и данных. Но в сетевой модели требуется, чтобы программист помнил структуру данных при формировании запросов.

Оптимальную структуру базы данных сложно сформировать, а готовую структуру трудно менять. Если вид таблицы претерпевает изменения, все отношения с другими таблицами должны быть установлены заново, чтобы не нарушилась целостность данных. Сложность подобной задачи приводит к тому, что программисты зачастую отменяют некоторые ограничения целостности ради упрощения приложений.

Реляционные базы данных

В сравнении с рассмотренными выше моделями реляционная модель требует от СУБД гораздо более высокого уровня сложности. В ней делается попытка избавить программиста от выполнения рутинных операций по управлению данными, столь характерных для иерархической и сетевой моделей.

В реляционной модели база данных представляет собой централизованное хранилище таблиц, обеспечивающее безопасный одновременный доступ к информации со стороны многих пользователей. В строках таблиц часть полей содержит данные, относящиеся непосредственно к записи, а часть — ссылки на записи других таблиц. Таким образом, связи между записями являются неотъемлемым свойством реляционной модели.

Каждая запись таблицы имеет одинаковую структуру. Например, в таблице, содержащей описания автомобилей, у всех записей будет один и тот же набор полей: производитель, модель, год выпуска, пробег и т.д. Такие таблицы легко изображать в графическом виде.

В реляционной модели достигается информационная и структурная независимость. Записи не связаны между собой настолько, чтобы изменение одной из них затронуло остальные, а изменение структуры базы данных не обязательно приводит к перекомпиляции работающих с ней приложений.

В реляционных СУБД применяется язык SQL, позволяющий формулировать произвольные, нерегламентированные запросы. Это язык четвертого поколения, поэтому любой пользователь может быстро научиться составлять запросы. К тому же существует множество приложений, позволяющих строить логические схемы запросов в графическом виде. Все это происходит за счет ужесточения требований к производительности компьютеров. К счастью, современные вычислительные мощности более чем адекватны.

Реляционные базы данных страдают от различий в реализации языка SQL, хотя это и не проблема реляционной модели. Каждая реляционная СУБД реализует какое-то подмножество стандарта SQL плюс набор уникальных команд, что усложняет задачу программистам, пытающимся перейти от одной СУБД к другой. Приходится делать нелегкий выбор между максимальной переносимостью и максимальной производительностью. В первом случае нужно придерживаться минимального общего набора команд, поддерживаемых в каждой СУБД. Во втором случае программист просто сосредоточивается на работе в данной конкретной СУБД, используя преимущества ее уникальных команд и функций.

2. СРЕДА VISUAL FOXPRO

Менеджер проекта

Project Manager (Менеджер проекта)– центр управления Visual FoxPro необходим для создания проектов, представления и управления файлами проекта. Содержит две основные вкладки:

I. **Data tab** (Вкладка данных);

1. базы данных;
2. таблицы;
3. запросы;
4. просмотры.

II. **Documents Tab** (Вкладка документов):

1. формы;
2. отчеты;
3. ярлыки.

Конструкторы Visual Foxpro

Table Designer (Конструктор таблиц) – помогает создавать и модифицировать свободные таблицы, таблицы баз данных, их поля и индексы.

Database Designer (Конструктор баз данных) – выводит на экран таблицы, просмотры связи, имеющиеся в базе.

Data Environment Designer (Конструктор среды данных) – помогает наглядно создавать и модифицировать среду данных, наборов форм и отчетов:

а) добавлять и удалять таблицы и просматривать среду данных;

б) вставлять таблицы и поля из среды данных, простой буксировкой их в формы и отчеты;

в) устанавливать, модифицировать связи среды данных.

Form Designer (Конструктор форм) – позволяет визуально создавать и модифицировать формы и наборы форм:

создавать, модифицировать, сохранять и настраивать формы;

добавлять средства управления к формам;

выполнять и тестировать функциональные возможности формы.

Label Designer (Конструктор ярлыков) – создает ярлыки, добавляет средства управления, определяет размеры ярлыка, создает макет ярлыка почтовой марки.

Menu Designer (Конструктор меню) – облегчает создание нового меню.

Query Designer (Конструктор запросов) и View Designer (Конструктор просмотров) – позволяет создавать и модифицировать запросы SQL и просмотры.

Report Designer (Конструктор отчетов) позволяет наглядно создавать и модифицировать отчеты, управляет выводом отчета.

Окна Visual Foxpro

1. **Окно Browse** – окно просмотра данных;
2. **Окно Code** – окно программного кода;
3. **Окно Command** – окно команды;
4. **Окно Debug** – окно отладки.

Мастера Visual Foxpro

Wizards (Мастера) – интерактивные инструментальные средства программирования, облегчающие выполнение общих задач, таких как создание форм, формирование отчетов и постановку запросов:

1. **Table Wizard** (Мастер таблиц) – создает таблицы на основе табличных структур Visual Foxpro)

2. **Form Wizard** (Мастер форм) – позволяет наглядно создать формы на основе многочисленных таблиц данных.

3. **One-to-many Form Wizard** (Мастер форм “ один ко многим”) – создает формы на основе двух связанных таблиц данных.

4. **Report Wizard** (Мастер отчетов) – создает отчеты, используя единственную таблицу.

5. **Group/Total Report Wizard** (Мастер группового/итогового отчета) – позволяет создавать итоговые отчеты.

6. **Query Wizard** (Мастер запросов) – создает запросы SQL, в которых можно определить базы данных, таблицы и поля, которые требуется включить в запрос.

7. **Cross-Tab Wizard** (Мастер перекрестных таблиц) – создает удобные перекрестные таблицы запросов на вывод результатов запросов SQL в виде электронной таблицы.

8. **Graph Wizard** (Мастер графиков) – используя Microsoft Graph, создает графики из данных, содержащихся в любой таблицы данных Visual Foxpro.

9. **Import Wizapd** (Мастер импорта) – упрощает импортирование данных из файлов других форматов в таблицы данных Visual FoxPro.

10. **Mail Merge** (Мастер слияния сообщений) – создает источник данных для документа Мшскщыщае Word или текстового файла, который может быть использован любым другим текстовым редакторо

3. ОРГАНИЗАЦИЯ И ХРАНЕНИЕ ДАННЫХ В VISUAL FOXPRO

I. Логическая архитектура базы данных в Visual FoxPro состоит из следующих источников данных:

Свободные таблицы

Базы данных;

Запросы.

Свободная таблица, как и таблицы базы данных, хранятся в файлах с расширением .dbf.

База данных хранится в файле с расширением .DBC и может содержать следующие объекты:

1. ***таблицы (Tables);***
2. ***локальные виды (Local Views);***
3. ***удаленные виды (Remote Views);***
4. ***связи (Connections);***
5. ***хранимые процедуры (Stored Prosedure).***

таблица базы данных представляет собой двумерный набор строк (записей) и столбцов (полей). Поле содержит информацию об одной категории данных и имеет тип и имя.

В таблице 3.1 приведены типы полей данных в Visual Foxpro:

Таблица 3.1

Основные типы данных

Тип данных	Описание
Character	Алфавитно-цифровой текст
Currency	Денежные единицы
Numeric	Числа
Float	Числа
Double	Числа удвоенной точности
Integer	Целые числа
Date	Месяц, день и год
DateTime	Date плюс часы, минуты и секунды
Logical	Истина или ложь
Memo	Символьный текст
General	OLE

В таблицах базы данных хранится дополнительная информация:

а) связи между таблицами;

б) длинные имена таблиц;

в) комментарии для полей и таблиц;

г) заголовки полей данных;

д) значение по умолчанию;

е) правила контроля при добавлении и изменении записей;

ж) триггеры и процедуры базы данных, соединение с внешним источником данных, локальные и удаленные виды.

Задание типов данных

Тип поля Character **-**Это наиболее распространенный тип данных для большинства таблиц. Символьные поля позволяют хранить от 1 до 254 символов, включающих буквы, числа, пробелы и знаки препинания. Обычное символьное поле не может содержать символы вида CHR(0). **CHR( ) –** Возвращает символ, который соответствует коду ANSI. Поля, требующие больше 254 символов, должны быть определены как поля типа Memo. Поля типа Character имеют фиксированный размер. Поля типа Character можно также использовать для хранения значений, состоящих полностью из чисел. Например, в виде символьных полей следует хранить такие данные, как почтовые индексы, номера телефонов и даже идентификационные номера заказчиков.

Тип поля Currency **-** Для хранения значений, выражающих денежные суммы, используется поле специального числового типа, именуемое Currency. Максимальное значение, которое может принимать поле этого типа, соответствует числу, немногим превышающему $922 триллиона. По умолчанию для значений полей типа Currency отводятся четыре десятичных знака и требуется восемь байтов памяти.

Типы полей Date и DateTime **–** Эти два поля схожи в том, что они оба используются для хранения дат в формате YYYYMMDD, причем для этого им требуется восемь байтов, независимо от того, какое значение задано в команде SET CENTURY – ON или OFF.(SET CENTURY – Определяет, должен ли Visual FoxPro показывать номер века в выражениях даты). Поле типа DateTime использует сжатый формат хранения времени в виде HHMMSS (ЧЧММСС), где HH имеет 24-часовое представление. Если преобразовать поле типа Date в поле типа DateTime, то по умолчанию время будет отображаться в формате 12:00:00AM.

Используя функцию DATETIME(), в записи, содержащей поле типа DateTime, можно зафиксировать текущие значения даты и времени (DATETIME()Возвращает текущую дату и время в формате значения заданных командами SET DATE, SET MARK, SET CENTURY, SET HOURS и SET SECONDS).

Для увеличения на один день даты, хранящейся в поле типа Date, необходимо к его значению добавить 1. Но чтобы увеличить на один день дату, хранящуюся в поле типа DateTime, необходимо добавить к его значению число 86400, так как в сутках содержится ровно 86400 секунд.

Поле типа Double **–** Поле типа Double (вещественное число с двойной точностью) представляет собой поле с плавающей точкой, позволяющее хра-

нить в сжатом формате число, представленное 18 цифрами. Поле занимает в таблице ровно 8 байт.

Поля типа Float и Numeric – Поля типа Float и Numeric (вещественные и числовые) предназначены для хранения значений, в представлении которых используется не более 20 цифр и не более 19 десятичных знаков. При этом для хранения каждой цифры в таблице требуется один байт памяти. FoxPro обрабатывает оба типа полей одинаково, что приводит к одной и той же степени точности.

В отличие от полей типа Double, поля типа Float и Numeric дают возможность определять число требуемых байтов, поскольку FoxPro хранит код ASCII каждой цифры в отдельном байте. Следовательно, если известно, например, что в качестве значений данного поля используются только целые числа, не превышающие 100000, то длину такого поля вполне достаточно определить цифрой 6 с нулевым числом десятичных знаков. Чтобы установить оптимальный размер поля типа Numeric, постарайтесь определить максимальное и минимальное возможные значения. Если окажется, что размерность поля недостаточна для сохраняемого в данном поле значения, FoxPro поместит в такое поле символы звездочки (*).

Поля типа General Чаще всего поле типа General (общий) используется для хранения графических изображений и представляет собой специализированное Memo-поле. FoxPro хранит поле типа General в том же FPT-файле, в котором хранятся и другие Memo-поля таблицы, но это не дает вам право использовать его таким же образом. Поле типа General первоначально предназначалось для хранения ссылок на связанные объекты OLE.

Поле типа Logical – В поле типа Logical (логическое поле) хранится двоичная информация в виде .T. или .F.. Это поле предназначено для запоминания данных, имеющих только два возможных значений. К какому типу, например, можно отнести данные, которые выражаются следующим образом: «мужчина/женщина», «отгружено/заказ не выполнен». Поле типа Logical часто служит источником информации для флажков опций.

Поле типа Memo – (поля примечаний) позволяют хранить большие символьные строки длиной более 254 символов. Для хранения каждой записи предоставляется переменный (нефиксированный) объем памяти, зависящий от размера блока. Блок имеет фиксированную в пределах всей таблицы длину. По умолчанию FoxPro использует блоки размером 64 байт. Это означает, что каждый фрагмент текста, состоящий из 64 символов, требует дополнительного блока.

Размер блока можно изменить с помощью команды SET BLOCKSIZE, которая устанавливает число байт в диапазоне от 33 до 511. Для определения блоков большего размера используйте целое число в диапазоне 1-32, которое означает, что длина блока определяется произведением заданного числа и числа 512. В Visual FoxPro можно задать нулевой размер блока, что заставит Visual FoxPro выделять память побайтово и тем самым не допустит напрасных потерь

памяти. Однако в этом случае пострадает быстродействие системы, чего не происходит при использовании блоков заданных размеров.

Размер блока должен быть задан командой SET BLOCKSIZE перед добавлением первой записи с Memo-полем. При добавлении первой записи FoxPro заносит текущий размер блока в Memo-файл. Изменяя размер блока уже существующего Memo-файла, необходимо перезаписать каждое Memo-поле. ***Однако, независимо от размера блока, не следует забывать, что в первом блоке резервируется восемь байтов для указателей.***

FoxPro хранит Memo-поля в файле с расширением .FPT, отдельно от DBF-файла. Независимо от количества Memo-полей в таблице, FoxPro хранит их в одном FPT-файле. А если в вашей таблице используются и поля General, FoxPro хранит их в том же самом файле, вместе с Memo-полями. В DBF-файле предусмотрены специальные указатели, которые отслеживают принадлежность Memo-информации каждой записи и полю.

Поскольку в указателях Memo-полей задан только один путь – из DBF- в FPT-файл, нужно следить за тем, чтобы эти DBF- и FPT-файлы не отделялись друг от друга. При копировании DBF-файла с одной машины на другую без FPT-файла копия этого файла не будет синхронизирована с текущим FPT-файлом, оставшимся на другой машине. Если это случилось и вы начнете добавлять записи, то вскоре обнаружится, что текст Memo-полей больше не соответствует «своим» записям. Эту проблему практически невозможно устранить без установки вручную указателей, определяющих путь из DBF- в FPT-файл.

Типичные случай использования Memo-полей:

Символьные поля, которые лишь иногда заполняются.

Очень большие по длине символьные поля или поля, длина которых не может быть заранее предсказана.

Текстовые файлы с резюме, деловыми письмами и архивами версии программ.

Локальный (или удаленный) вид (View) – это сохраняемый запрос, к результатом которого доступ осуществляется как к таблице.

Запросы (Query) хранятся в файлах с расширением .QPR и позволяют просматривать данные, удовлетворяющие описанным критериям. Результаты запроса могут быть отображены различными способами: при помощи окна таблицы или в отчете.

Главное окно Visual FoxPro

При загрузке системы FoxPro активизируются два основных объекта «**Главная Строка Меню**» и «**Окно Command**».

Главная строка меню:

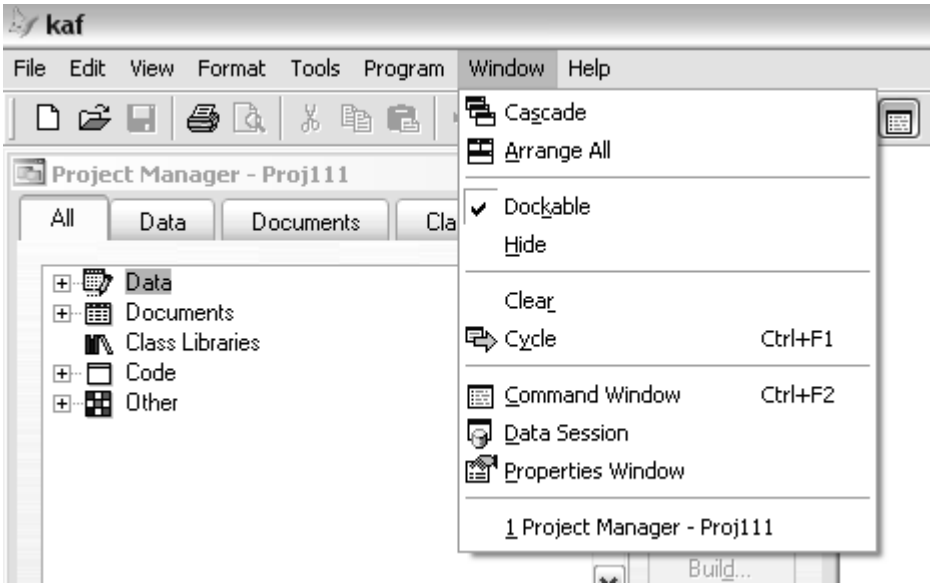
File Edit View Format Tools Program Windows Help

Таблица 3. 2

Команды меню View

Команда	Описание
Edit (Редактирование)	Включает режим Edit; для просмотра и редактирования записей. В этом режиме поля таблиц отображаются вертикально
Browse (Просмотр)	Включает режим Browse для просмотра и редактирования записей. В этом режиме поля таблиц отображаются горизонтально. Строки представляют записи
Append Mode (Добавление), (Расширенный режим)	Добавляет пустую запись в конец таблицы, при этом указатель записи устанавливается на ее первое поле
Design (Конструкторы)	Отображает окно Конструктора форм (Form Designer), или окно Конструктора отчетов (Report Designer), или окно Конструктора этикеток (Label Designer) — в зависимости от того, что тебе нужно
Tab Order (Порядок табуляции)	Определяет порядок обхода объектов в экранных формах при использовании клавиши <Tab>
Preview (Предварительный просмотр)	Отображает отчеты на экране такими, какими они будут после печати
Data Environment (Среда данных)	Определяет таблицы и отношения, данные из которых будут использованы в экранных формах и отчетах (этикетках)
Properties (Свойства)	Открывает диалоговое окно Properties (Свойства), в котором отражены все свойства экранных форм и элементов управления
Code (Код, программа)	Открывает окна программ для редактирования методов объектов
Form Controls Toolbar (Панель элементов экранной формы)	Открывает панель инструментов для разработки экранной формы во время работы в конструкторе форм
Report Control Toolbar (Панель инструментов отчета)	Открывает панель инструментов для разработки отчета во время работы в Конструкторе отчетов
Layout Toolbar (Панель инструментов для макетирования)	Используя эту панель инструментов, можно выравнивать элементы управления в формах и отчетах
Color Palette Toollbar (Панель предварительного просмотра отчета)	Панель палитры цветов позволяет выбирать цвет для элементов управления

Команда главного меню Help – справочная информация:
Команда главного меню Window – управление окнами:



- Hide – удалить окно (восстановить: Names of BD)
- Clear – очистить экран, окна остаются на экране
- Move ^F7 – перемещение окна
- Size ^F8 – изменение размера окна
- Zoom F10 – распахнуть окно во весь экран
- Zoom ^F9 – показать только заголовок окна
- Cycle ^F1 – циклический переход между окнами
- Color – установка цветов окон
- Command ^F2 – переход в командное окно
- Debug – вызов окна отладки программ
- Trace – вызов окна трассировки программ
- View – переход в окно меню View

Окно ***View*** содержит кнопки включения окон меню: установок <**Setup**>, открытия файла <**Open**>; команды: редактирования <**Browse**>, закрытия БД <**Close**>, задания дорожек к файлам <**Files**>; установки режимов работы системы <On/Off>, установки параметров даты, времени и др. <**Misc**>. В рабочей области (**Work Areas**) показаны имена открытых баз данных и имена свободных областей. Индексированные БД отмечены точкой перед именем.

Работа с базами данных производится через **пункты Главного Меню File**.

Главное Меню File.

- New - создать новый файл
- Open - открыть записанный ранее на диске файл
- Close - закрыть активное окно
- Save - сохранить файл на диске (^W)
- Save as - записать файл на диске под новым именем
- Revert - убрать изменения, внесенные после открытия файла

– Printer Setup - установить параметры для печати
– Print - печать файла
– Quit - выход из FoxPro.

Основная работа с содержимым баз данных осуществляется через **Главное Меню Database** – работа с файлами БД, модификация структуры, создание БД:

– SetUp - работа с индексными файлами, установка фильтров и полей
– Browse - создание меню и окна редактирования БД
– Append From - добавление записей в конец БД из другой БД
– Copy To - копирование записей из активной БД в другую БД
– Sort - сортировка записей с созданием новой БД
– Total - вычисление суммы значений в полях БД
– Average - вычисление средних значений в полях БД
– Count - подсчет числа записей в полях БД
– Sum - суммирование числовых данных в полях БД
– Calculate - вычисления в БД
– Report - работа с созданным ранее файлом отчета
– Label - работа с созданным ранее файлом визиток
– Pack - удаление помеченных маркером (*) записей в БД
– Reindex - обновление индексного файла после внесения в БД изменений (при закрытом индексном файле).

При редактировании текущей записи в БД используются команды верхнего **Главного Меню Record.**

– Record - работа с записями БД
– Append - режим автодобавления пустых записей в БД
– Change - показать изменения в записях БД после выполнения команд установки фильтров, полей доступа и т.п.
– Goto - переход к указанной записи
– Locate - поиск записи
– Continue - продолжить поиск записи (^K)
– Seek - поиск в индексированной БД
– Replace - произвести вычисления и замену записей в поле БД
– Delete - пометить маркером записи к удалению
– Recall - убрать маркер для удаления записей.
– Для работы с программой предназначено меню **Program:**
– Do...(^D) - выполнить программу,
– Cancel - прервать выполнение программы,
– Resume (^M) - продолжить выполнение программы,
– Compile... - компиляция программы,
– Generate... - генерировать программный файл,
– FoxDoc - работа с документацией,
– FoxGraph... - работа с графикой.

При активизации пунктов меню появляются всплывающие меню, описанные ниже.

Окно Command – для ввода команд в интерактивном режиме. ***При работе с меню выполняемые команды дублируются в окне команд (Command), что позволяет контролировать и повторять введенные ранее команды меню.***

Создание однотабличной базы данных и индексирование

Создать базу данных можно одним из следующих способов:

– В окне диспетчера проекта «Project Manager» выбрать вкладку Databases и нажать командную кнопку <New>

– Использовать команду CREATE DATABASE

– Выполнить команду File / New и выбрать опцию Database (**Data Designer**) или **Data Wizard**

При любом из перечисленных способов создания базы данных появится окно «Create», в котором необходимо указать имя создаваемой базы данных.

При создании и изменении таблиц используются **Table Designer** (Конструктор таблиц) или **Table Wizard** (Мастер таблиц) – более автоматизированный процесс.

Добавить таблицу можно только в открытую базу данных путем создания новой таблицы или подключения уже существующей свободной таблицы:

- использовать диспетчер проекта, Data ((New) или (Add)).
- для открытой базы данных выполнить команду CREATE TABLE;
- выполнить для открытой базы данных команду ADD TABLE.

Table Designer Visual Foxpro.

Table Designer Конструктора таблиц можно вызвать из строки главного меню: **File→New→New File**. В диалоговом окне ввести имя таблицы, сохранить. В окне **Table D**esigner имеются две основные вкладки: **Fields и Indexes**.

Для создания таблицы во вкладке Fields необходимо ввести данные:

– имена полей (name);

– типы полей (type);

– размерность(Width, decimal);

– определить свойства полей (Field Properties);

– правило проверки правильности значения поля (Validation Rule);

– текст сообщения об ошибке (message);

– используемое значение по умолчанию;

– заголовок поля в окне Browse (Caption);

– текст комментария для поля данных (Field Comment);

– отметить значение поля как неопределенное (NULL), пока не введены данные, если поле должно быть всегда заполнено.

Создать индексный файл.

Для создания простого индексного файла по имени поля с заданной сортировкой (index (Regular)).

Для создания более сложных индексных файлов можно воспользоваться вкладкой **Indexes**, нужно задать:

Order – порядок сортировки (возравтания/убываение);

Name – имя создаваемого индексного файла;

Type – тип индекса (Primary – первичный индекс (позволяет вводить только уникальное значение поля), Regular - простой для любых значений, Unique –обрабатывающий только несовпадающие значения поля);

Expression – создание ключа с помощью построителя выражения с использованием математических, логических, строковых операций и операций с полями типа дата.

Filter – определить условие выбора с помощью построителя выражения;

II Table Wizard (Мастер таблиц)

Данный способ более автоматизирован и состоит из 4 шагов:

1 шаг – выбор полей для таблица

2 шаг – корректировка параметров выбранных полей

3 шаг – определение ключевых полей

4 шаг – сохранение таблицы

Table Properties содержит имя базы данных, в этом окне определяются правила проверки данных и триггеры. Триггерами называют выражения или процедуры, выполняемые при внесении изменений в базу данных. Insert trigger , update trigger, delete trigger . Триггеры всегда выполняются после всех других проверок и в основном предназначены для обеспечения поддержки целостности данных.

Структура команд СУБД FoxPro

Команды ориентированы на обработку файлов баз данных и имеют два типа структур команды.

Структура команды первого типа:

НАЗВАНИЯ [<границы>][<список выражений>] [FOR<условие>] [WHILE<условия>],

где

НАЗВАНИЕ – имя команды:

<границы> – границы действия команды, которые могут иметь одно из следующих значений:

ALL – все записи;

REST – все записи, начиная с текущей, до конца базы;

NEXT <N> – следующие N записей начиная с текущей;

RECORD <N> – запись номер N;

FOR <условие> – выполнение команды только для записей, удовлетворяющих <условию>;

WHILE<условия> – выполнение команды только до тех пор, пока не перестанет выполняться <условие>;

[…] – в квадратных скобках указывается необязательная, но возможная часть конструкции команды;

<…> – в угловые скобки программист должен поместить нужное выражение.

Команды второго типа можно назвать командами установок. Структура команды второго типа:

SET <параметр команды> TO <значение команды>

SET <параметр команды> OFF/ON

Такие команды не влекут какие-то немедленные действия, а определяют условия работы других команд.

Например:

CREATE Name - создать файл с именем Name.dbf.

SET CLOCK ON - отобразить текущее время.

Функции возвращают значения, состоят из собственного имени и аргумента, заключенного в скобки. Например:

DATE( ) - текущая дата, YEAR (DATE( )) - текущий год.

RECCOUNT( ) - число записей БД.

Для **преобразования одного типа поля** или переменной в другое существует набор функций:

- DTOC() - дата преобразуется в символьный тип,
- CTOD() - перевод символьного типа в дату,
- INT() - преобразование чисел в целое, знаки после точки отбрасываются,
- ROUND() - округление вещественного числа, STR(<число>,<длина>,[дробное]) - преобразование числа в символьную переменную, <длина> - задает длину строки, последний параметр определяет количество знаков после запятой,
- VAL(символ) - преобразование символьного типа в числовой.

Переменные содержат значение, имя состоит не более чем из 10 символов, имеют такие же типы, что и поля баз и переменные оперативной памяти.

Например, командой STORE присвоим значение "Рубцовск" переменной CITY:

STORE 'Рубцовск' TO CITY или CITY = ” Рубцовск ” .

.STORE "12" TO A && занесение в переменную A числа 12.

.STORE "13" TO B

.? VAL(A)+VAL(B) && на экране распечатается результат

25.00

Кроме переменных полей базы существуют переменные оперативной памяти, их может быть до 256. Тип переменной определяется системой при записи в нее данных.

Например:

m1=259.22 && числовая переменная,

m2=’ПЕТРОВ А.’=’ПЕТ’ && логическая переменная, в результате: .T.

m3='01/01/95' && символьная переменная, задающая дату,

m4=CTOD(01/01/95) && символьную переменную перевели в формат даты.

Переменные памяти могут быть локальными и глобальными. ***Глобальные переменные*** объявляются:

PUBLIC <список переменных>

Для задания ***локальных переменных*** служат команды:

PRIVATE <список переменных>

PRIVATE ALL && все переменные локальные

Локальные переменные используются во внутренних командных файлах, тогда разные переменные в разных файлах могут иметь одинаковые имена. Локальные переменные "невидимы" во внешнем командном файле.

Константы содержат значения, в зависимости от типа заключаемые в определенные символы:

- Символьный (тип C) – в апострофы, например: 'O.K.' или "все в порядке";
- Дата (тип D) – в фигурные скобки, например: {30.01.88}, {01/30/88};
- Логический (тип L) – в точки, например: .Y., .T. (верно), .N., .F. (ложно).

В системе FoxPro применяются следующие **типы файлов:**

1. **.dbf** файл базы данных;
2. **.fpt** файл текстов примечаний;
3. **.mem** файл для сохранения временных переменных;
4. **.idx, .cdx** индексный файл, мультииндексный;
5. **.prg** файл-программа на языке FOX;
6. **.txt**текстовые файлы передачи параметров в другие системы.

Работа с массивами переменных

Переменные типа массивов определяются:

DIMENSION имя массивы (<размерность>)

Могут быть одномерные и двумерные массивы. Индекс массива начинается с 1. Данные элементов массива могут быть любого типа. Если массив сформирован как двумерный, к нему можно обращаться и как к одномерному.

ПРИМЕР:

X(1,1); X(1,2); X(1,3); X(2,1); X(2,2); X(2,3) – перечислены элементы двумерного массива, который задан:

```
DIMENSION X(2,3)
```
К элементу X(2,2) можно обратиться как к 5-му элементу, то есть X(5).

ПРИМЕР. *В разные элементы заносятся различные типы данных.*

```
DIMENSION X(2,3)
STORE "GOODBAY" TO X(1,2)
STORE 99 TO X(5)
STORE .T. TO X(2,3) && занесение логической переменной.
```

Данные из массива можно переместить в поля одной записи базы с помощью команды:

GATHER FROM <массив> **[FIELDS** <список полей базы>**]**

Данные перемещаются из элементов массива в указанные поля текущей записи. Для обратного перемещения команда

SCATTER [FIELDS <список полей>**] TO** <массив>

Если параметр FIELDS отсутствует, то перемещаются все поля.

Организация процедур

В FoxPro предусмотрена возможность использования процедур, которые могут быть как внешними, так и внутренними. Частным случаем процедуры являются процедуры функции.

Структура **внутренней процедуры**:

PROCEDURE <имя процедуры>

<тело процедуры>

RETURN [TO MASTER/<выражение>],

где параметры последней команды RETURN, дающей выход в файл верхнего уровня, имеют значения:

TO MASTER – дает возврат на командный файл самого верхнего уровня,

<выражение> – содержит возвращение значения, если процедура представляет функцию.

Внутри процедуры может быть команда CANCEL, которая создает прерывания и выход в инструментальную среду FoxPro .

Процедуры могут быть собраны в процедурный файл – **внешние процедуры**. Процедура вызывается в оперативную память открытием процедурного файла:

SET PROCEDURE TO <имя файла>

Обращения к процедуре выполняется командой:

DO <имя процедуры> [WITH <список параметров>] [IN <файл>],

где WITH – список передаваемых и получаемых параметров;

IN – имя файла, внутри которого находится процедура.

При вызове командного файла или процедуры с передачей параметров в этом случае программа, куда передаются параметры, должна начинаться с команды

PARAMETERS <список параметров>

Просмотр содержимого базы

Для того, чтобы работать с данными базы, необходимо открыть файл. Под работу с файлом выделяется при этом оперативная память – рабочая область, где будет находиться файл базы. Если файлов несколько, для них **назначаются рабочие области**:

SELECT <название или номер рабочей области>

Номера могут быть от 1 до 10 или от A до J, первая область назначается по умолчанию. По этой же команде активизируются файлы, то есть становятся текущими.

Команда открытия

USE [[имя файла]/?] [ALIAS псевдоним],

где

— /? – распечатка имен файлов .DBF, открытых с помощью каталога открытия;

— ALIAS – задание псевдонима, дополнительного имени базы, это имя присваивается рабочей области;

Вывод содержимого базы

LIST [FIELDS <список имен полей>] [<границы>] [FOR <условие>] [WHILE<условие>] [TO PRINTER | TO FILE <имя файла>]

Где – <границы> – задание диапазона, по умолчанию ALL; FIELDS – задание списка имен выводимых полей; FOR – выборка записей по заданному условию;TO PRINTER – вывод данных на печать;TO FILE <имя файла> – вывод данных в файл.

В LIST не распечатываются поля MEMO, необходимо указать имя поля - memo.

Например, для распечатки:

LIST FIELDS имя-поля-Memo.

Вывод текущей записи

DISPLAY [<границы>] [FIELDS <список>] [FOR <выражение>] [WHILE <выражение>] [OFF] [TO PRINT]

где <границы>– задание диапазона, по умолчанию - текущая запись;

— OFF – снятие номера записи, по умолчанию поля нумеруются;

— TO PRINT - вывод записи на принтер.

В DISP не распечатываются поля MEMO. Для их распечатки DISP ALL FIELDS имя-поля-Memo.

Например, активизируются файлы AM01, AM02, AM03.

```
USE AM01 ALIAS A1
SELECT 2  && Назначение рабочей области файлу
USE AM02 ALIAS A2
SELECT 3
USE AM03 ALIAS A3
LIST 1
```

```
SELECT A2 && активен файл A2
GOTO 3&& доступ в A2 к записи 3
DISPLAY FIO, A2.NAM, A3.ZAP && обращение к полям разных файлов.
```

Создание структуры таблицы

Во-первых, решите, какую информацию следует помещать в таблицу.

- Фамилия, имя, отчество сотрудника;
- Дата рождения;
- Профессия;
- Заработная плата;
- Комментарий;

Каждая из перечисленных величин будет храниться в соответствующем поле таблицы.

Как создать структуру таблицы

В меню "Файл" выберите команду "Создать". Появится диалог "Создать файл". Выберите опцию "Таблица/DBF", а затем нажмите кнопку "Создать". Появится диалог "Структура таблицы"

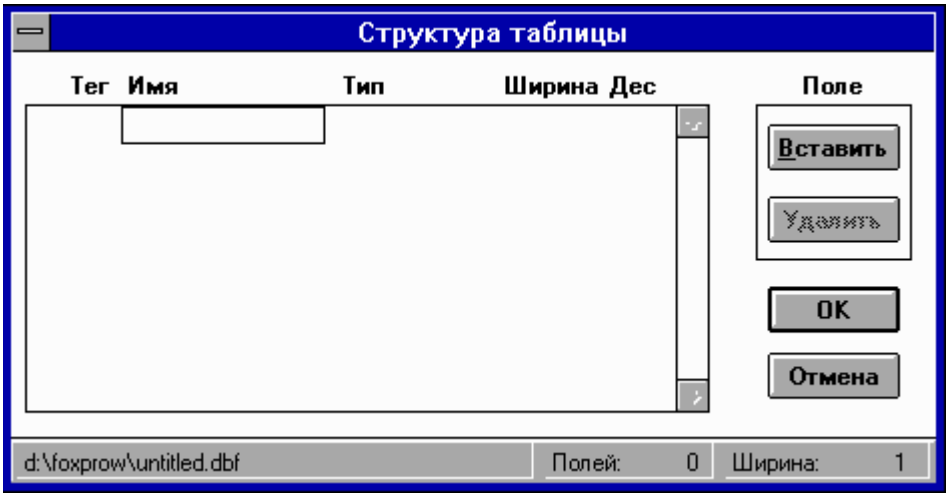


Рис. 3.1. Создание структуры таблицы

Курсор находится в текстовом поле "Имя". Теперь можно добавить имя, тип и ширину каждого поля.

Как описать поля таблицы

Наберите "фио" в текстовом поле "Имя". Нажмите клавишу Tab или щелкните мышью в списке "Тип". Установите тип "Character". Нажмите клавишу Tab или щелкните мышью на поле "Ширина", после чего наберите 25 или с помощью стрелок увеличьте ширину до 25. Далее опишите остальные поля.

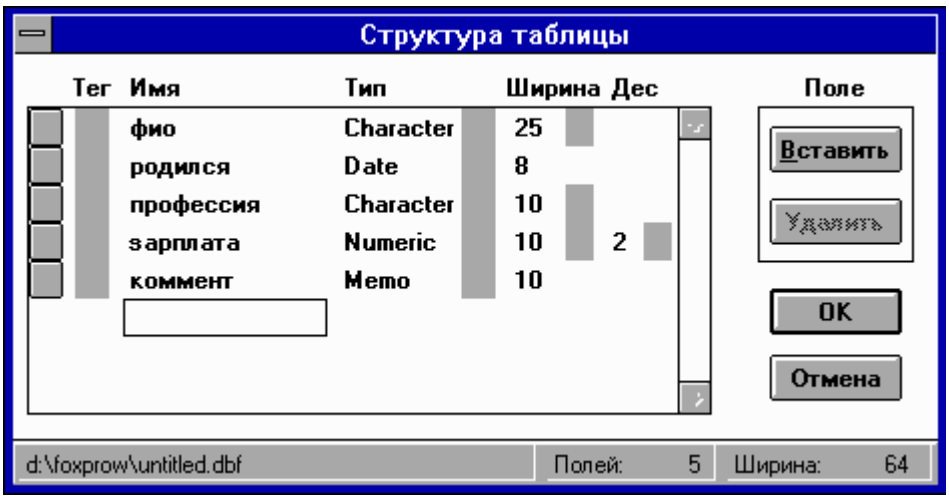


Рис. 3.2. Описание полей таблицы

Поле "Дес" указывает число десятичных разрядов в числовом поле. Так как в поле "зарплата" хранится значение денежной суммы, то следует задать число 2.

После ввода всех значений нажмите кнопку "Ok". Появится диалог "Сохранить как". Введите имя таблицы: TEST, если необходимо, укажите, в какой каталог ее записать, и нажмите кнопку "Сохранить". Затем FoxPro спросит: "Вводить записи данных сейчас ?" Нажмите кнопку "Да".

Ввод данных в таблицу

Появится окно просмотра в режиме изменения по записям. Добавляем в таблицу следующие данные:

- Иванов Сергей Николаевич
- 01/01/6
- бухгалтер
- 350000

Для ввода информации в поле memo нажмите клавиши CTRL+PGDN или дважды щелкните на этом поле. Появится memo-окно, в которое введите комментарии.

Закройте memo-окно. Заметьте, что буква "M" в слове Memo прописная, это означает, что данные в поле уже введены. Добавьте еще нескольких сотрудников и закройте окно просмотра.

Теперь, предположим, Вы приобрели сканер и желаете добавить в таблицу фотографии своих сотрудников.

Как изменить структуру таблицы

В меню "База" выберите команду "Настройка". В левом верхнем углу диалога "Настройка базы" нажмите кнопку "Изменить" Появится диалог "Структура таблицы". Подведите курсор к полю "комментарий" и нажмите кнопку "Вставить". Над полем "комментарий" появится "новое_поле". Замените "новое_поле" на "фото". В списке "Тип" выберите значение "General", а затем

нажмите кнопку "Ok". На запрос FoxPro о том, необходимо ли сохранить изменения, нажмите кнопку "Да". В диалоге "Настройка базы" нажмите кнопку "Ok".

Ввод в таблицу графических данных

Прежде всего Вам необходимо внести в буфер обмена фотографию сотрудника.

В меню "Запись" выберите команду "Править". В окне просмотра устанавливается режим изменения по записям. Дважды щелкните по полю "фото". Появится окно редактирования поля типа General. В меню "Правка" выберите команду "Вставить". Картинка, содержащаяся в буфере обмена, появится в окне.

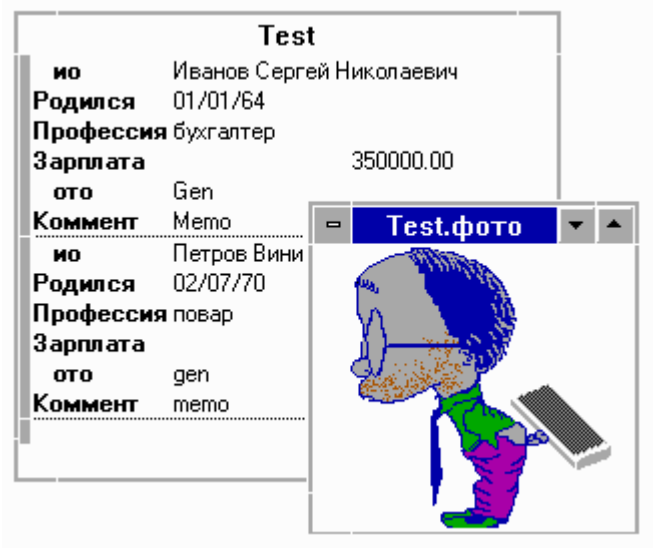


Рис. 3.3. Запись картинки в таблицу

Как добавить новые записи

В меню "Запись" выберите команду "Добавить". Появится окно просмотра в режиме добавления, куда Вы можете добавить новых сотрудников.

Как удалить запись

В режиме просмотра таблицы следует пометить поля для удаления. Для этого слева от поля нажать маркер удаления.

Как упаковать таблицу

В меню "База" выберите команду "Упаковать". Появится сообщение с запросом о том, хотите ли Вы упаковать таблицу. Нажмите кнопку "Да". Теперь в меню "База" выберите команду "Просмотр" и убедитесь в том, что все выделенные записи были удалены.

Использование конструктора таблиц

Для организации файлов будем использовать диспетчер проектов Project Manager. Прежде всего необходимо создать проект. В FoxPro это можно сделать по-разному: с помощью команд Visual FoxPro, опций меню или мастера проектов Project Wizard.

Для начала из основного системного меню выберите команду File→New (Файл → Создать). Откроется диалоговое окно New (Создать).

Установите переключатель типа файла в положение Project (Проект) и щелкните на кнопке New File (Новый файл). Поскольку при создании любого файла в FoxPro требуется его имя, то сначала откроется диалоговое окно Create (Создать), позволяющее задать имя файла проекта. FoxPro использует это же диалоговое окно для всех запросов имени файла.

В диалоговом окне Create показан текущий каталог или каталог, действующий по умолчанию. Если в этом каталоге уже есть проекты, они будут выделены в окне списка пониженной яркостью, чтобы их нельзя было выбрать. Они служат для напоминания о том, что не следует использовать одно и то же имя проекта. При этом, чтобы сохранить проект, у вас есть возможность перейти в другой каталог или на другой диск. После выбора каталога необходимо ввести новое имя файла проекта вручную.

Если при создании таблиц вы предпочитаете более прямой метод, введите имя файла в оке Command с использованием команды: **CREATE PROJECT –** Открывает диспетчер проектов, в котором можно создать проект

Использование окна команд Visual FoxPro – более быстрый путь, но при этом необходимо знать синтаксис команды, а также конкретное имя и каталог, где будет сохранен проект.

Создав проект, FoxPro автоматически открывает его. Комплекс информации о проекте оформлен в виде набора вкладок. Все вкладки представляют файлы разных типов, которые обозначены на их корешках в верхней части диалогового окна. Чтобы создать таблицу, щелкните на вкладке Data (Данные). Создайте сводную таблицу, выбрав в списке элемент Free Tables и щелкнув на кнопке New (Создать). Затем Visual FoxPro для создания таблиц предложит сделать выбор: воспользоваться услугами мастера Table Wizard или конструктора таблиц Table Designer.

Затем FoxPro снова запрашивает имя файла, но на сей раз для таблицы.

Далее FoxPro открывает диалоговое окно Table Designer. Конструктор таблиц содержит три вкладки. в первой вкладке (Fields) определяется структура таблицы, во второй (Indexes) – индексы, а в третьей (Table) отображается статус таблицы.

Чтобы определить структуру таблицы, введите информацию о полях, включающую тип поля, его ширину и количество десятичных знаков.

При нажатии на клавишу <Tab> фокус получает поле Type (Тип), которое по умолчанию установлено равным значению Character, в этом поле используется раскрывающийся список.

Для полей типа Numeric используется столбец Decimal (Десятичные знаки), предназначенный для задания количества десятичных знаков после запятой. FoxPro активизирует этот столбец только для полей типа Numeric.

Следующий столбец используется для определения направления индекса (по возрастанию или убыванию).

Последний столбец в структуре позволяет определить, допустимы ли для текущего поля значения NULL.

С помощью кнопок, расположенных слева от имен полей, можно изменить порядок следования полей при использовании команд: **BROWSE –** Открывает окно просмотра и выводит записи из текущей или указанной таблицы. **LIST –** Непрерывным потоком отображает информацию о таблице или о среде окружения.

Завершив ввод структуры таблицы, щелкните на кнопке ОК. Для новой таблицы FoxPro обязательно поинтересуется, собираетесь ли вы вводить в нее записи.

Имя только что созданной таблицы теперь появится в разделе Free Tables в окне диспетчера проектов. Чтобы увидеть его, следует кликнуть на знаке «плюс», расположенном слева от раздела Free Tables. А слева от имени таблицы вы должны увидеть два символа: знак «плюс» и перечеркнутый круг. Первый указывает на дополнительные уровни, которые в настоящее время скрыты или свернуты в текущей строке. При щелчке на знаке «плюс», стоящем рядом с именем таблицы, диспетчер проектов развернет список полей таблицы. При этом знак «плюс» заменится знаком «минус», это означает, что все уровни в настоящий момент раскрыты. FoxPro использует второй символ (перечеркнутый круг) на этапе компиляции, чтобы указать, что этот файл не должен компилироваться.

Окно редактирования базы данных

Для работы с содержимым базы данных необходимо ее открыть, выбрать из главного меню **Database** → меню **Browse,** появляется новый пункт верхнего меню, позволяющий конфигурировать окно редактирования БД и содержащий ряд команд для работы с записями и с полями.

Browse – редактирование содержимого базы данных

– Browse/Change,Append – горизонт./вертикальное расположение полей
– Grid On/Off – установка линий разделения полей
– Link/Unlink Partitions – синхронизация курсора в окнах
– Change Partition ^H – переход курсора между окнами
– Size Field – изменить отображаемый размер поля
– Move Field – переместить поле
– Resize Partitions – разделение окна на две части
– Goto – переход к указанной записи
– Seek – индексный поиск записи
– Toggle Delete ^T – пометка записи к удалению
– Append Record ^N – добавить одну пустую запись

Команды Size и Move не изменяют структуру БД и нужны только для представления этой структуры в форме, удобной для редактирования.

Команда поиска Goto позволяет перейти к первой (Top), последней (Bottom), указанной (Record N) записи, перескочить (Skip N) через N записей при редактировании БД.

ФОРМАТ КОМАНДЫ

BROWSE [FIELDS <список>] [LOCK <выражение>] [FREEZE <поле>] [WIDTH <выражение>] [NOAPPEND],

где параметры

— LOCK - количество полей слева, которые не смещаются по экрану;

— FREESE - запрещение изменений в полях, кроме указанного;

— WIDTH - ограничение количества символов в поле;

— NOAPPEND - закрытие на ввод новых записей.

В интерактивном режиме данные можно **редактировать** в базе вызовом команд **EDIT** или **CHANGE,** в которых осуществляется корректировка базы по полям в полноэкранном режиме с помощью:

CHANGE [<границы>] [FIELDS <список полей>] [FOR <условие>]

Высвечивается одна строка с вертикальным расположением полей.

Команда **INSERT** организует вставку записи после текущей. Ее формат INSERT [BEFORE] [BLANK]

— BLANK –- вставка пустой записи,

— BEFORE – вставка записей впереди текущей.

Использование значений NULL

В более ранних версиях FoxPro невозможно было определить, преднамеренно ли оставил пользователь поле пустым или просто забыл о нем. В FoxPro незаполненное поле интерпретируется либо как пустая символьная строка, либо как числовое значение 0, либо как логическое значение False (Ложь). Любое из этих значений может быть недопустимым для поля.

Для использования значений полей NULL в поле любого типа необходимо выполнить две операции. Во-первых, в окне Command или в программе необходимо выполнить команду SET NULL ON. Во-вторых, необходимо модифицировать структуру и щелкнуть по кнопке NULL для каждого поля, которое допускает нулевые значения. Если этого не сделать, то при добавлении записей с помощью команд APPEND FROM или INSERT SQL FoxPro выдаст ошибку и не присвоит значения этим полям. FoxPro заносит лексему .NULL. в поля, которым разрешено оставаться пустыми и которые не содержат никаких значений.

Если после выполнения команды SET NULL ON не был установлен флажок Null, FoxPro не станет использовать пустые значения или пробелы в полях первичных или потенциальных (candidate) ключей.

По умолчанию в FoxPro не разрешается задание значений NULL. после выполнения команды SET NULL ON нельзя пройти поле, не введя в него никакой информации.

ПРАВИЛА ИСПОЛЬЗОВАНИЯ НУЛЕВЫХ ЗНАЧЕНИЙ:

– По умолчанию команда APPEND BLANK не заносит нулевую лексему во все поля новой записи.

– Если поле типа Character модифицируется таким образом, что в нем разрешается хранить значения NULL, пустые поля так и остаются пустыми.

– Если поле типа Numeric модифицируется таким образом, что в нем разрешается хранить значения NULL, поля со значением 0 останутся в неприкосновенности.

– Если поле типа Character модифицируется таким образом, что в нем запрещается хранить значения NULL, это поле заполняется пустой строкой.

– Если поле типа Numeric модифицируется таким образом, что в нем запрещается хранить значения NULL, такое поле заполняется числом 0.

Создание базы данных

В Visual FoxPro под базой данных понимают совокупность таблиц.

Сначала для хранения таблиц создайте контейнер базы данных. Следующая команда создает новую базу данных и одновременно присваивает ей имя:

CREATE DATABASE SALES

Это можно также сделать, выбрав из системного меню команду File→New→Database (Файл → Создать→ База данных), однако этот метод требует открытия нескольких диалоговых окон.

Для того, чтобы узнать, открыта ли в данный момент хоть одна база данных, достаточно посмотреть раскрывающийся список Databases (Базы данных), расположенный на панели инструментов. Обычно он пуст. Если же открыта одна или несколько баз данных, то их имена отображаются в упомянутом списке, что дает возможность переходить от одной базы к другой.

DBC( ) Возвращает имя текущей базы данных и путь к ней.

SET DATABASE Задает текущую базу данных.

ADATABASES( ) Помещает в массив переменных памяти имена всех открытых баз данных и описания путей для этих имен.

MODIFY DATABASE Открывает конструктор баз данных, давая возможность в диалоговом режиме модифицировать текущую базу данных.

Модификация структуры таблицы

На определенных этапах развития любого проекта может потребоваться изменение структуры таблицы. Серьезность таких изменений можно оценить тем, насколько они изменяют саму таблицу и индексные файлы.

Добавление поля, например, можно назвать незначительным изменением, так как оно не затрагивает уже существующие поля, хотя и требует перезаписи всего DBF-файла. Переименование рядового поля также требует минимальных затрат, фактически это приводит лишь к перезаписи заголовка DBF-файла. Однако если вы переименовываете поле, входящее в индекс, то это индекс или его часть должны быть также модифицированы. Удаление полей, не входящих в индекс или в его часть, требует перезаписи всего DBF-файла, но не более того. С другой стороны, изменение размеров полей и количества десятичных знаков

вынуждает FoxPro к перезаписи всего DBF-файла с возможной при этом потерей данных. При изменении типа поля FoxPro пытается автоматически преобразовать данные к новому типу, что может привести к появлению «мусора» в данных, если FoxPro не знает, что с ними делать, или если преобразование не имеет смысла.

Добавление полей

Добавление новых полей к существующей таблице – одно из самых безопасных действий в модификации структуры таблицы. Проблемы могут возникнуть только в том случае, если попытаться дважды использовать одно и то же имя поля. Но даже и в такой ситуации FoxPro реагирует автоматически.

FoxPro не позволит вам выйти из столбца имени, если вы использовали дубликат существующего имени поля. Естественно, FoxPro отреагирует на это «безобразие» сообщением “Invalid or duplicate name” (Неверное или повторяющееся имя поля) и предоставит возможность отредактировать его.

Удаление полей

На определенном этапе одно или несколько полей таблицы могут стать ненужными. Чтобы не тратить попусту память, лучше их из таблицы удалить. Для удаления поля достаточно отобразить таблицу в диалоговом окне Table Designer, выделить ненужное поле и щелкнуть на кнопке Delete.

Переименование полей

При переименовании полей может потребоваться перезапись заголовка DBF-файла с новым именем поля. Для переименования поля откройте диалоговое окно Table Designer, выделите нужное имя поля, отредактируйте его, а затем сохраните структуру.

Если на поле, имя которого изменяется, не ссылается открытый индекс, FoxPro выполняет переименование при сохранении структуры. Если переименованное поле содержится в открытом индексе, FoxPro отобразит предупреждение “Variable «имя переменной» is not found”. При щелчке на кнопке ОК в этом окне предупреждения Visual FoxPro откроет вкладку Index диалогового окна Table Designer, чтобы можно было исправить индексное выражение. Можно переопределить индексное выражение, скорректировав в нем имя поля. Visual FoxPro не переименовывает автоматически имя поля, участвующее в индексном выражении.

Переопределение полей

Переопределение типов полей, длины или количества знаков после запятой может быть простым или сложным в зависимости от характера вносимых изменений. Например, можно открыть диалоговое окно Table Designer, выделить нужное поле и увеличить его размеры без всяких проблем: в результате FoxPro лишь перезапишет после этого DBF-файл. При этом поля типа Character получают дополнительные пробелы, а поля типа Numeric будут иметь больше зна-

чащих цифр. Никаких проблем не будет даже при изменении размера индексированных полей. Visual FoxPro послушно пересоздаст индекс при закрытии конструктора таблиц. С другой стороны, уменьшение размера поля или числа знаков после запятой может привести к потере данных. FoxPro принимает изменения и при выходе из диалогового окна Table Designer спрашивает, нужно ли обновить структуру таблицы. Если вы согласны, FoxPro изменяет размеры полей в соответствии с новыми требованиями.

Самые большие неприятности могут вызвать некоторые изменения, связанные с типами полей. При замене поля Numeric полем Character число просто преобразуется в строку, как если бы при перезаписи таблицы была использована функция: **STR( )** – Возвращает символьный эквивалент заданного числового выражения.

Точно так же замена типа Character типом Numeric приводит к использованию функции: **VAL( ) –** Возвращает числовое значение символьного выражения, состоящего из цифр.

При этом использование строк, начинающихся цифрами, приводит к созданию чисел, сохраняемых в поле типа Numeric. Если же преобразуемые строки начинаются с буквенных символов (отличных от пробела), то при их переводе в числовые значения поля Numeric записываются нули.

Точно так же можно преобразовать поля типа Date в поля типа Character. FoxPro переводит данные, используя функцию: **DTOC( ) –** Возвращает дату в символьном виде, исходя из выражения типа Date или DateTime.

Можно также преобразовывать даты, представленные в виде форматированных строк символов, в настоящие поля типа Date.

Большинство других преобразований приводит к потере данных.

Добавление существующих таблиц в базу данных

Для добавления существующей таблицы в текущую базу данных щелкните на пиктограмме Add Table (Добавить таблицу) панели инструментов Database Designer или выберите из системного меню команду Database → Add Table (База данных → Добавить таблицу). Когда выбранная таблица появляется в окне проекта, она может накладываться или частично перекрывать существующие таблицы. С окнами определения полей можно обращаться так же, как с любыми другими: их можно перетаскивать или менять размеры.

Чтобы изменить структуру таблицы, щелкните правой кнопкой мыши в любом месте окна и выберите из контекстного меню команду Modify (Изменить). Можно также щелкнуть на таблице левой кнопкой мыши (чтобы выбрать ее), а затем щелкнуть на пиктограмме Modify Table (Изменить таблицу) панели инструментов Database Designer. При этом откроется диалоговое окно Table Designer, в котором можно изменить любую из характеристик таблицы.

Длинные имена полей

Одно из первых изменений, которое желательно выполнить, - переименование полей с длинными именами. Имена полей ограничиваются 128 символа-

ми. Для смены имени поля достаточно выделить его и ввести новое. Пробелы в именах полей недопустимы (FoxPro не будет знать, где завершается имя поля). Для создания более ясного и наглядного имени поля можно вместо пробела использовать символ подчеркивания. Однако это не рекомендуется в новом соглашении о присвоении имен полей таблицы. Лучше всего для первого символа каждого значимого слова в имени поля использовать прописную букву, а для остальных – строчные. К сожалению, в окне Table Designer не поддерживается смешанный регистр букв. Более того, Visual FoxPro в различных своих окнах отображает имена полей так, как «хочется» данному окну, а не вам. Например, названия столбцов в окне Browse отображаются с использованием только первых прописных букв.

Таблицы, входящие в базу данных, могут иметь 128-символьные имена. Для переименования таблицы используйте поле Name (Имя) во вкладке Table (Таблица) диалогового окна Table Designer или команду:

CREATE TABLE Создает таблицу с описаниями полей.

Например:

CREATE TABLE orddet1 NAME order_details

Кроме того, согласно общепринятому соглашению о присвоении имен полям таблиц, имени поля должен предшествовать префикс типа поля.

При переходе от коротких имен полей к длинным необходимо выполнить ряд действий. Во-первых, если уже имеются запросы, отчеты или программы, использующие 10-символьные имена полей, отредактируйте их, прежде чем перейти к использованию более длинных имен полей. Во-вторых, если вы создадите запросы, отчеты или программы, использующие длинные имена полей, то при удалении таблицы из базы данных они, возможно, не будут работать, так как при этом имена полей усекаются до 10 символов. Если такое усечение не коснется первых 10 символов, то никаких проблем не возникает. Однако следует помнить, что FoxPro может автоматически заменить один или два последних символа последовательными номерами, чтобы обеспечить уникальность всех имен полей в таблице.

Во многих окнах списков полей, используемых в Visual FoxPro (особенно в окнах мастеров), отводится ограниченное количество символов для отображения имен полей. Если в имени поля используются префиксы, идентифицирующие таблицу, или если имена связанных полей начинаются с некоторого общего первого слова, то не исключено, что вы можете не отличить одно поле от другого.

Если таблица задана с длинными именами полей, то нужно использовать только эти имена. Нельзя использовать ни более короткие имена, сохраненные в DBF-файле, ни усеченные 10-символьные имена, о которых уже шла речь в этом разделе.

Формат поля и маска ввода

В базе данных можно хранить желаемый формат представления значения поля и маску ввода. При этом вводимые вами варианты формата и маски ввода совпадают с возможными вариантами в формах и отчетах. Фактически, смысл сохранения их в базе данных станет ясен при создании в экранной форме соответствующего элемента управления для данного поля. Следовательно, когда вы создаете элемент управления для поля путем перетаскивания его из таблицы на поле формы, формат поля, заданный в диалоговом окне Table Designer, автоматически устанавливает нужные свойства для данного элемента управления полем.

Контроль данных на уровне поля

В диалоговом окне Table Designer можно также добавить правила контроля данных на уровне поля. Чтобы добавить такое правило, введите его непосредственно в тестовое поле Rule (Правило) раздела Field Validation (Контроль поля) или щелкните на кнопке Expression Builder (Построитель выражений), расположенной сразу справа от поля и помеченной троеточием.

При определении правил контроля данных можно использовать любое логическое выражение. Можно даже вызывать сложные функции логического анализа, которые трудно разместить в одной инструкции. К таким выражениям предъявляется лишь одно требование: они должны возвращать значение .T. или .F.. Механизм анализа соблюдения правил запускается при любой попытке покинуть поле или изменить его значение с помощью инструкций:

INSERT Вставляет в текущую таблицу новую запись.

REPLACE Обновляет записи в таблице.

Значения полей по умолчанию

Значения полей по умолчанию можно задавать для любого поля таблицы. Например, для присвоения в качестве значения по умолчанию текущей системной даты необходимо в текстовое поле Default Value (Значение по умолчанию) поместить функцию

DATE( ) Возвращает текущую системную дату, которая контролируется операционной системой.

У выражения, определяемого в качестве значения по умолчанию, тип должен совпадать с типом поля, которому присваивается это значение. В противном случае FoxPro сгенерирует ошибку. FoxPro помещает значения по умолчанию в поля таблицы всякий раз, когда добавляется новая запись, либо с помощью команды APPEND и аналогичной ей, либо в интерактивном режиме в окнах Browse или Edit.

Значения заголовков

При просмотре или редактировании таблицы в качестве заголовков столбцов FoxPro использует значение, заданное в поле Caption (Заголовок). По умолчанию с этой целью FoxPro использует имя поля. Однако это не всегда оказы-

вается самым лучшим вариантом хотя бы потому, что при соблюдении какого-либо соглашения о присвоении имен часто используются префиксы таблицы или другие префиксы, которые совсем не стоит отображать как заголовки полей в окнах Browse и Edit. Можно определить заголовок на основе содержимого другого поля или другой переменной, но в большинстве случаев для этого вводится простая символьная строка в текстовое поле Caption. Это значение также используется для создания в формах надписи для элемента управления поля с помощью метода «перетащить и опустить» (drag-and-drop) или при использовании мастера создания форм Form Wizard.

Комментарий к полю

Комментарий к полю – это примечание, которое должно напомнить пользователю о назначении и смысле информации в поле. FoxPro хранит этот элемент как поле типа Memo, следовательно, оно может быть любой длины. Комментарий может содержать список возможных значений, таблицы, используемые для проверки достоверности вводимых значений, и другую информацию. Пользователь сам решает, обращаться или нет к полю комментария. Чем больше информации записано в определении таблицы, тем легче поддерживать ее в будущем.

Текст комментария отображается при выделении поля после надписи Description (Описание) в нижней части окна Project Manager. Это значение также используется для свойства Comment элемента управления соответствующего поля при его включении в экранную форму методом «перетащить и опустить».

Использование первичного и потенциального ключей

Щелкните на вкладке Indexes диалогового окна Table Designer и раскройте список Type. в этом списке представлены все четыре типа индексов. Только потенциальный индекс может быть назначен первичным. Это означает, что индексное выражение должно иметь уникальное значение для каждой записи в таблице. Первичный индекс используется для формирования связей с другими таблицами.

Проверка первичного и потенциального индексов выполняется при обновлении записи. Поэтому, если окажется, что при вводе новой записи нарушается требование уникальности ключа, соответствующее сообщение будет сформировано только в момент, когда пользователь попытается выйти из текущей записи.

Открытие и просмотр таблицы

Таблица - это файл, содержащий структурированную информацию. Такие файлы иногда называют файлами .DBF, так как они имеют расширение .DBF.

Система Visual FoxPro содержит очень мощный инструмент для просмотра и редактирования информации в таблице. Для ознакомления с этими возможностями мы воспользуемся уже созданной таблицей: CUSTOMER.DBF, которая содержит информацию о заказчиках.

Как открыть и просмотреть таблицу

Установите курсор в окно команд, щелкнув в нем. Наберите set default to d:\foxpro\tutorial и нажмите клавишу ENTER.

В меню "Файл" выберите команду "Открыть". Появится диалог "Открыть". Сделайте активным каталог TUTORIAL. В списке "Тип" выберите "Таблица/DBF". И, выбрав файл CUSTOMER.DBF, нажмите кнопку "Открыть".

Теперь эта таблица является активной, и для ее просмотра в меню "База" выберите команду "Просмотр". Появится окно просмотра. Информация в окне просмотра представлена в строках и столбцах. Столбцы называются полями, строки – записями. Окно просмотра обычно недостаточно велико, чтобы дать возможность увидеть всю таблицу сразу. Для того чтобы увидеть различные части таблицы, нужно прокрутить окно просмотра по горизонтали и по вертикали. В окне просмотра можно управлять отображением данных на экране.

Например,

Можно **изменить ширину отдельных полей.** Установите курсор мыши между заголовком поля COMPANY и заголовком поля CONTACT. Эта линия называется разделителем заголовков. При попадании на разделитель заголовков указатель мыши изменяет вид. И теперь, нажав кнопку мыши, установите требуемый размер поля.

Можно **переместить поле**. Укажите с помощью мыши на заголовок поля CNO. И, нажав кнопку мыши, перемещайте заголовок, пока он не окажется между COMPANY и CONTACT.

Можно увидеть **все поля одновременно, перейдя в режим просмотра по записям.** В этом режиме поля каждой записи располагаются одно под другим. В меню "Просмотр" выберите команду "По записям". Режим просмотра изменится.

Еще одним способом просмотра данных является **разбиение окна просмотра на два раздела** при помощи маркера разбиения – черного прямоугольника в левом нижнем углу окна просмотра. Установите мышь на маркере разбиения. Указатель мыши изменится. Переместите маркер разбиения окна за поле CONTACT.

Разбиение окна просмотра позволяет:

- прокручивать поля в одном разделе, не перемещая поля в другом;
- работать с одним разделом в режиме просмотра по записям, а с другим в режиме табличного просмотра.

Можно выбрать раздел и изменить его режим. Выберите правый раздел, установив на нем указатель мыши, и, щелкнув в меню "Просмотр", выберите команду "По записям".

Так как разделы связаны, то при прокрутке выделяется одна и та же запись для обоих разделов.

Можно **устранить связь между разделами** и прокрутить записи в одном разделе, оставив записи другого раздела без изменения. В меню "Просмотр" выберите команду "Связать разделы". Маркер, расположенный рядом с командой "Связать разделы", исчезнет. Прокрутите несколько записей.

Закрыть окно просмотра. Закрытие окна производится стандартно для системы WINDOWS, дважды щелкните на кнопке управляющего меню окна просмотра или нажмите клавиши CTRL+F4.

Работа с полями типа Memo

Теперь откроем еще одну таблицу, содержащую информацию о продавцах, участвовавших в некоторых коммерческих сделках, это файл: SALESMAN.DBF

В этой таблице определено поле memo NOTES, находящееся между полями ONO и NAME. Это поле специального типа; оно может содержать неограниченное количество любой информации. Так как объем информации может быть очень большим, ее трудно отобразить в маленькой области. Для просмотра и редактирования этой информации служит окно редактирования memo-поля. Прописная буква "M" указывает на то, что memo-поле содержит информацию, а строчная буква "m" - на то, что поле пусто. Для просмотреть данных memo-полей откройте окно редактирования memo-поля. Для этого дважды щелкните на слове Memo.

В окне редактирования memo-поля вы можете его не только просмотреть, но и отредактировать. Для закрытия окна просмотра memo-поля щелкните дважды на кнопке управляющего меню окна просмотра.

Индексирование

Упорядочение и сортировка записей в таблице

Чтобы отсортировать записи в таблице, можно использовать команду **SORT,** которая на основе существующей таблицы создает новую, упорядочивая записи по одному или нескольким полям. Приведенная ниже команда создает новую таблицу с именем CUSTLIST, отсортированную по фамилии клиента:

```
SORT TO CUSTLIST ON Last
```

При использовании более сложного варианта команды SORT создается новая таблица с именем CURCUST с записями, отсортированными по убыванию идентификационного номера клиента и относящимся только к тем клиентам, которые совершили покупки в этом году. Команда для создания этой таблицы имеет следующий вид:

```
SORT TO CURCUST ON cCustId /D FOR Goods_Ytd>0
```

Этот метод имеет два недостатка. Во-первых, каждая новая сортировка дублирует всю первоначальную таблицу или отфильтрованную ее часть. Если нужно отсортировать большую таблицу с разными вариантами сортировки, можно быстро исчерпать дисковое пространство. Во-вторых, существование более одной копии таблицы неизбежно приведет к противоречивости данных. Если не обновлять все таблицы одновременно, то скоро получим несколько

таблиц, в каждую из которых внесены лишь некоторые изменения, но далеко не все.

Сортировка записей все же имеет право на существование. Если имеется редко изменяемая таблица с привилегированным порядком сортировки, то, вероятно, имеет смысл хранить ее отсортированную версию. Однако индексы обеспечивают более эффективный способ, позволяющий пользователям просматривать и выбирать данные из таблицы в определенном порядке. Так как с таблицей можно связать больше одного индекса, можно определить разные индексы для различных представлений или отчетов.

Поиск в отсортированной таблице выполняется лишь немногим быстрее, чем в неотсортированной, но индексированной по тем же полям.

Сравнение автономного со структурным и неструктурным индексами

Когда в системе баз данных было впервые введено понятие индекса, то разрешалось определять только одиночные индексы. Например, для индексирования таблицы CUST по полям, содержащим номер клиента и его фамилию, необходимо было создавать два индекса, как показано в примере:

```
USE CUST
INDEX ON cCustId TO CUSTID
INDEX ON cLast TO CUSNAME
```

Эти команды создают два индексных файла: CUSTID.IDX и CUSTNAME.IDX, которые называют автономными индексами, так как каждый из них содержит одиночный индексный элемент и не зависит от других. Может быть определено любое количество автономных индексов, ограниченное лишь значением, задаваемым инструкцией FILES в файле CONFIG.SYS. При открытии таблицы можно открыть все связанные с ней индексные файлы:

```
USE CUST INDEX CUSTID, CUSTNAME
```

Можно открыть таблицу только с одним индексом:

```
USE CUST INDEX CUSTID
```

В обоих случаях первый индекс после ключевого слова INDEX управляет порядком, в котором FoxPro получает доступ к записям таблицы. В первом примере при добавлении, удалении или изменении записей FoxPro обновляет оба индекса. Во втором случае в текущем сеансе FoxPro работает только с открытым индексом. Если вносятся изменения в файл CUST.DBF, то индекс CUSTNAME.IDX может потерять согласованность с таблицей.

Можно открывать каждый индекс отдельно, используя команду **SET INDEX:**

```
USE CUST
```

```
SET INDEX TO CUSTID
SET INDEX TO CUSTNAME ADDITIVE
```

Теперь FoxPro открывает оба индекса, и индекс CUSTID управляет последовательностью доступа к записям.

Проблемы с автономными индексами очевидны. Так как имена индексных файлов обычно не несут информации о том, к каким DBF-файлам они относятся, можно легко забыть, какие индексы принадлежат каждой таблице. Кроме того, если не отрыть все индексы при редактировании таблицы, FoxPro не сможет обновить отсутствующие индексы.

С появлением FoxPro можно работать со структурными и неструктурными индексами, которые также называют составными индексами. Они представляют собой специальные индексные файлы, которые могут содержать несколько индексных определений. Теперь все индексные определения можно хранить в одном физическом файле.

Определение составного индекса показано в следующем примере:

```
USE CUST
INDEX ON cCustId TAG CUSTID OF CUSTSORT
INDEX ON cLast TAG CUSNAME OF CUSTSORT
USE CUST INDEX CUSTSORT
```

По последней команде USE открывается таблица CUST вместе с неструктурным индексом CUSTSORT. FoxPro определяет составной индекс как неструктурный, когда его базовое имя отличается от имени DBF-файла. Можно сделать индекс структурным, присвоив ему такое же имя, как и у DBF-файла.

```
USE CUST
INDEX ON cCustId TAG CUSTID OF CUST
INDEX ON cLast TAG CUSNAME OF CUST
USE CUST
```

Если опустить член OF в инструкции INDEX ON, то определение индекса создается или добавляется в структурный индекс автоматически.

Когда структурный индекс существует, FoxPro автоматически открывает его при открытии таблицы. При таком подходе просто невозможно забыть открыть какой-либо индекс, если его выражение включено в виде тэга в структурный индекс. Структурные индексы всегда синхронизированы с DBF-файлом.

Определение регулярного и уникального индексов

Чтобы создать индексы для файла, следует в диалоговом окне Table Designer выбрать вкладку Indexes.

Каждое определение индекса начинается с имени тэга, за которым следует тип индекса (по умолчанию предлагается Regular - регулярный), индексное выражение и фильтр. При использовании Table Designer FoxPro автоматически подразумевает создание структурного индекса. Если нужно создать автономный или неструктурный индекс, следует перейти в окно Command и ввести с клавиатуры команды, описанные ранее.

Стрелки, расположенные слева от имени индекса, означают возрастающий или убывающий порядок индексации. Для изменения направления выберите нужную строку и щелкните на кнопке со стрелкой. Регулярный индекс означает, что FoxPro сохраняет в индексе значения, сгенерированные индексным выражением для каждой записи таблицы. Если несколько записей имеют одно и то же значений, FoxPro заносит это значение в индекс несколько раз, но с различными значениями указателей для каждой записи.

Щелкнув на кнопке со стрелкой, относящейся к столбцу Type (Тип), в раскрывшемся списке можно увидеть другой тип индекса, именуемый Unique (Уникальный). В индекс этого типа включаются только уникальные значения индексного выражения. А если несколько записей генерируют одно и то же значений индексного выражения, в индексе хранится значение только первой встретившейся записи.

Потенциальный и первичный ключи

Третий тип индекса, именуемый потенциальным (Candidate), создает уникальный индекс, в который включаются все записи в таблице. Потенциальные индексы запрещают повторяющиеся значения для любых двух записей в таблице. Возникает вопрос, что произойдет, если заменить существующий регулярный индекс потенциальным. После выполнения замены FoxPro предложит сохранить модификацию структуры. При этом в диалоговом окне появится флажок для подтверждения проверки данных таблицы на соответствие требованиям такой индексации. Независимо от того, установлен флажок или нет, FoxPro при наличии повторяющихся значений в индексе предупреждает об ошибке, связанной с нарушением уникальности, и возвращает определение индекса снова к регулярному. Таким образом, перед внесением этих изменений данные должны быть модифицированы соответствующим образом.

Записи, отмеченные для удаления, не игнорируются при проверке на наличие повторяющихся значений первичным (Primary) или потенциальным (Candidate) индексами. Следовательно, при попытке добавления новой записи, в которой значение поля, включенного в определение первичного или потенциального индекса, дублирует значение из записи, отмеченной для удаления, возникает ошибка нарушения уникальности. Эту запись нельзя будет добавить до тех пор, пока таблица не будет упакована.

Любая сводная таблица может иметь потенциальный индекс, и только таблицы, включенные в контейнер базы данных, могут иметь первичные индексы. Иногда таблица может иметь несколько полей, уникально идентифицирующих каждую запись. Любой из этих индексов считается потенциальным и может

быть квалифицирован как потенциальный первичный ключ. Однако таблица может иметь только один первичный индекс. На базе первичных ключей часто формируются отношения между несколькими таблицами, и они служат для подстановки значений в ссылочной таблице.

Индексирование по сложным выражениям

FoxPro не ограничивает индексные выражения только одним полем. В качестве индексного выражения может выступать любая комбинация полей. При создании потенциального индекса следует остерегаться чрезмерно сложных выражений.

Для построения сложного выражения щелкните по кнопке, расположенной справа от текстового поля Expression (Выражение) во вкладке Indexes окна Table Designer. FoxPro запустит построитель выражений Expression Builder.

Диалоговое окно Expression Builder «оснащено» множеством операций, которые помогут в построении сложных индексов. В разделе Function (Функции) предоставляется доступ к встроенным функциям FoxPro, которые разделены на четыре группы: Date (Функции работы с датами), Logical (Логические)? Math (Математические) и String (Строковые). Чтобы отобразить раскрывающийся список функций, кликните на кнопке со стрелкой, расположенной справа от поля функции нужного типа. FoxPro отображает выбранную функцию в верхнем левом поле списка функций и в текстовой области формируемого выражения. При выборе функции FoxPro автоматически помещает курсор внутри круглых скобок, чтобы можно было сразу ввести параметры функции.

В нижней части диалогового окна Expression Builder приводится список полей, определенных в текущей таблице. Здесь можно выбрать поле, которое будет добавлено в позицию вставки текстового поля Expression. Чтобы выбрать поле из другой таблицы, кликните на стрелке справа от текстового поля From Table (из таблицы). В этом списке показаны только открытые в данный момент таблицы. Если нужно сослаться на таблицу, которая в настоящее время не открыта, выйдите из диалогового окна Expression Builder, откройте таблицу в новой рабочей области, а затем снова откройте диалоговое окно Expression Builder. Хотя и не запрещается создавать индексы, используя поля из разных таблиц, но этого делать не рекомендуется. Однако в иных ситуациях при работе с окном Expression Builder вам еще понадобится возможность доступа к другим таблицам.

В списке Variables (Переменные) перечисляются текущие переменные памяти и системные переменные. Этот список не имеет непосредственного отношения к определению индексов, но не следует забывать о том, что FoxPro использует диалоговое окно Expression Builder и во многих других случаях.

Прежде чем щелкнуть по кнопке ОК и тем самым запомнить индексное выражение, воспользуйтесь утилитой FoxPro, которая проверит синтаксис построенного вами выражения. Для этого щелкните на кнопке Verify (Проверить): FoxPro выполнит проверку синтаксиса и выдаст сообщение об ошибках, если не сможет интерпретировать выражение. В процессе проверки синтаксиса FoxPro

обнаруживает такие типичные ошибки, как несоответствие открывающих и закрывающих круглых скобок или отсутствие необходимой запятой. Если выражение верно, то в строке состояния появится сообщение, подтверждающее правильность индексного выражения.

Использование функций, определенных пользователем

Наряду с встроенными функциями FoxPro можно использовать и функции, определенные программистом. Определенная пользователем функция (user-defined function) представляет собой группу инструкций, сохраненных в виде отдельного файла либо в виде отдельной процедуры или функции в той же программе.

Использование хранимых процедур

Главный недостаток использования в индексном выражении определяемой пользователем функции состоит в том, что программный файл этой функции всегда в зоне досягаемости FoxPro, чтобы можно было сформировать индекс для новых или модифицируемых записей. Так как функция не хранится вместе с индексом, файл может оказаться «не в том месте», его можно удалить или просто забыть при переносе таблицы на другую систему. Если таблица связана с базой данных, определяемые пользователем функции могут сохраняться в базе данных, что исключает процесс поиска процедуры или функции.

Понятие об индексных ограничениях

Ограничения, связанные с индексным выражением, накладываются как на длину строки самого индексного выражения, так и на его значение. В индексных файлах резервируется ограниченный объем памяти для индексного выражения и его значения.

Для автономного индекса длина индексного выражения не может превышать 220 символов, а итоговое значение индекса – 100 символов. Если определить автономный индекс как компактный (специальный вариант автономного индекса, требующий меньшего объема дискового пространства), то индексное выражение будет совместно использовать общую память с выражением FOR. В этом случае общая длина обеих строк индексных выражений не может превышать 512 символов, а значение каждого индексного выражения – 240 символов. Подобные требования применяются и к составным индексам. Составные индексы – не что иное, как компактные индексы с несколькими тэгами, таким образом, на них накладываются те же самые ограничения.

Короткие индексные выражения более эффективны, чем длинные.

Выбор активного индекса в процессе выполнения

В приложениях, использующих автономные индексы, довольно просто задать активный индекс. Если инструкция USE открывает несколько индексов, то по умолчанию активным становится первый индекс в списке. **SET ORDER –** Задает управляющий файл или тег индекса для данной таблицы.

Первая из приведенных ниже инструкций открывает файл CUSTOMER с двумя автономными индексами, а команда SET ORDER заменяет управляющий индекс CUSTID индексом CUSTNAME:

```
USE CUSTOMER INDEX CUSTID, CUSTNAME
SET ORDER TO 2
```

Чтобы сделать команду более понятной, можно вместо соответствующего номера позиции в списке задать имя нужного индекса.

Если команда USE не открывает индексные файлы, выполните команду:

SET INDEX – Открывает один или несколько файлов индексов для текущей таблицы

```
USE CUSOMER
SET INDEX TO CUSTID, CUSTNAME ORDER 2
```

Эти варианты остаются работоспособными и в Visual FoxPro. Однако с введением составных индексов, в которых каждый индекс имеет имя тэга, вам придется добавлять аргумент TAG:

```
USE CUSTOMER
SET ORDER TO TAG CUSTNAME
```

В этом примере подразумевается, что таблица CUSTOMER имеет структурный индекс, составленный из тэгов с именами CUSTID и CUSTNAME. Хотя FoxPro и открывает структурный индекс автоматически, но при этом не устанавливает автоматически индекс управления. Поэтому необходимо использовать команду SET ORDER TO TAG с указанием выбранного тэга.

Ситуация усложняется, если одновременно используются и структурные, и автономные индексы. В следующей команде снова открывается таблица CUSOMER.DBF со структурным индексом CUSTOMER.CDX, но FoxPro также открывает и автономный индекс с именем CUSTZIP:

```
USE CUSTOMER INDEX CUSTZIP
```

В приведенной выше команде не задан структурный индекс – FoxPro открывает его автоматически. CUSTZIP – неструктурный индекс и поэтому имеет приоритет над другими индексами. Следовательно, в данном случае именно автономный индекс CUSTZIP будет управлять последовательностью просмотра записей.

Добавление записей с помощью окон Browse или Edit

Для добавления записей можно открыть окно Append (добавить записи) и внести необходимые изменения. С другой стороны, достаточно открыть таблицу и ввести в окно Command команду

APPEND Добавляет одну или несколько записей в конец таблицы

следующим образом:

```
USE CUSTOMER
APPEND
```

При выполнении этих команд открывается окно редактирования, в таблицу добавляется пустая запись и в первое поле помещается курсор. После введения данных во все поля первой записи FoxPro автоматически добавляет следующую пустую запись и помещает курсор в ее первое поле. Этот процесс продолжается до тех пор, пока не будет закрыто окно редактирования.

Окно Edit (Редактирование) можно также открыть непосредственно с помощью команды **EDIT,** а окно Browse (Просмотр) - командой **BROWSE.** FoxPro открывает эти окна для редактирования или просмотра существующих записей. Чтобы начать ввод новой записи, нельзя просто переместить курсор за пределы последней записи. Для этого необходимо нажать клавиши <Ctrl+Y> или выбрать команду меню Table→ Append New Record (Таблица → Добавить новую запись).

4. КОМАНДНЫЕ ФАЙЛЫ

Общие сведения

Команды при работе в интегрированной среде FoxPro отображаются в командном окне, и любую из них можно вновь выполнить, подведя к ней курсор и нажав Enter. Набор команд можно оформить в виде отдельной программы (командных файлов). Для создания (или открытия) программы в **Главном Меню File** выбирают **New** (или **Open**) и выбирают тип файла – **Program.**

FoxPro выступает как интерпретатор строк, программа представлена командным файлом. Каждая команда программы на языке FoxPro должна начинаться с новой строки. Одну команду можно записать в нескольких строках, разделяя их точкой с запятой (;). Текст программы можно редактировать с помощью команды из **Главного Меню EDIT.** Вызов текстового редактора в окне Command командой:

MODIFY COMMAND <имя файла.prg>

ПРИМЕР:

```
*          Программа вывода таблицы премий.
* очистка экрана
CLEAR        && очистка экрана
* открыть  БД premia
USE premia  && открыть  БД premia
* установить удобный формат даты
```

```
SET DATE GERMAN     && установить удобный формат даты
* открыть окно BROWSE с общим заголовком и
* описанием полей
BROWSE TITLE ‘Премиальная ведомость’ ;
 FIELDS   Fam: H=" Фамилия" : 12,;
 data_r: H=" Дата рождения" ,;
money: H=" Премия"
CLOSE DATA
```

Эта программа предназначена для вывода в удобной форме БД из трех полей: fam – фамилия, data_r – дата рождения, money – премия.

Знак "*" в начале строки программы преобразует всю строку в комментарий. Команда открытия окна **BROWSE** включает ряд опций:

TITLE – заголовок в верхней строке окна,

FIELDS – назначения для полей

(:**H**="имя" – смена отображения названия, через двоеточие можно задать размер поля).

Всего команда BROWSE может иметь около 40 параметров. Некоторые из них:

:**R** – разрешен только просмотр поля,

:**V="**логическое выражение" - проверка вводимых данных на соответствие указанному выражению. Может быть усилено добавлением

:**F** – контроль имевшейся в поле информации.

:W="логическое выражение" – контроль входа в поле,

:B="нижняя граница", "верхняя граница" - допустимый диапазон чисел или дат, может быть усилен добавлением :F,

:E="Сообщение при неверном вводе данных в поле" - заменить стандартное системное сообщение (Invalid input) на указанное,

:P="шаблон" – задание формата отображения данных,

FOR "условие" – фильтр для отображения записей,

COLOR SCHEME "номер" – установить номер цветовой схемы,

PARTITION "номер" – раздел окна на две части по колонке с указанным номером,

TIMEOUT "сек." – время ожидания ввода данных

Допустимо отображать вычислимые поля, например:

nalog = IIF(zar<10000, 0.05*zar, 0.12*zar) :H= "Налог" :P= "####.#" – поле прогрессивного налога с зарплаты zar по формату.

Ключевые слова программы отделяются пробелами. Пробелы нельзя игнорировать при записи команды в несколько строк.

Набрав программу, необходимо запомнить ее в виде файла командой ^W.

Запуск на выполнение через меню (Do...) потребует поиска имени программы в списке файлов типа Program, удобнее **запускать программу** из окна команд (Do "имя программы"), **для редактирования** – Modify command "имя программы".

При наличии ошибок в программе текст команды, содержащий ошибку, выделяется, и появляется запрос на дальнейшие действия:

– **Cancel** – прервать выполнение программы,

– **Ignore** – продолжить со следующей команды,

– **Suspend** – приостановить выполнение с возможностью выполнить промежуточные команды. Возобновление работы при этом производится командой **Resume**.

Основные команды языка

Типы команд ввода-вывода

Одно из мощных средств программирования FoxPro – комплекс команд @...SAY...GET, предназначенный для ввода/вывода в заданных координатах экрана. Используется в основном тогда, когда стандартные формы редактирования Browse/Change неприемлемы.

Общая структура команды:

@ Y,X **SAY** "выражение"
PICTURE "шаблон" FUNCTION "коды"
COLOR SCHEME "номер"/COLOR "список цветовых пар"
GET "переменная или поле"
PICTURE "шаблон" FUNCTION "коды"
DEFAULT "начальное значение переменной"
ENABLE/DISABLE MESSAGE "сообщение"
(OPEN) WINDOW "имя окна для Memo-поля"
RANGE "диапазон допустимых значений вводимой величины"
SIZE "число строк, число колонок (размер области ввода)"
VALID "контроль вводимой переменной"
ERROR "сообщение об ошибке контроля VALID"
WHEN "условие входа в редактируемое поле"
COLOR SCHEME "номер"/COLOR "список цв.пар"

Наиболее употребимые коды шаблонов PICTURE:

– **A** - ввод только букв,
– **L** - ввод только логических значений,
– **N** - ввод только букв и цифр,
– **X** - ввод любого символа,
– **Y** - ввод только логических Y или N,
– - ввод только цифр (и знаков в числовых данных),
– **#** - ввод только цифр, пробелов и знаков,
– **!** - преобразовывать строчные буквы в прописные (только англ.),
– **.** - определяется позиция десятичной точки.

Наиболее употребимые FUNCTION-коды :

– **A** - ввод только буквы,
– **B** - выравнивать числа по левой границе,
– **I** - центрирование текста внутри поля,

– **J** - выравнивание текста по правой границе поля,

– **K** - режим "радикального" изменения содержимого поля,

– **M "список"** - список возможных значений как набор элементов, разделенных запятыми,

– **R** - вывод в шаблоне символов, не являющихся частью данных (и не запоминающихся),

– **S "число"** - ограничение ширины вывода указанным числом символов,

– **T** - исключить конечные пробелы,

– **!** - строчные буквы (только англ.) преобразовывать в прописные.

Действие FUNCTION-кодов, в отличие от PICTURE, распространяется не на весь объект, а на отдельные его символы. Слово "FUNCTION" отдельно можно не выводить, а вставить коды (поставив вначале "@", а перед ними пробел) внутрь шаблона "PICTURE".

ПРИМЕР:

```
@ 2,2 SAY 'Зарплата: ' GET zar PICTURE '####.##'
@ 7,8 SAY 'Паспортные данные: ' GET p ;
PICTURE '@R! Серия AAA-XX номер 999999'
```

READ && Задержка и считывание данных, предъявляемых на редактирование командами SAY и GET.

Команды ? / ??

В текущей позиции экрана выводят значения заданных выражений.

Формат:

? / ?? "выражение_1" PICTURE "шаблон" FUNCTION "коды" AT "номер столбца" STYLE "шрифт при выводе на печать" "выражение_2" ...

Если установлено **SET PRINT ON**, то вывод направляется как на экран, так и на принтер. **SET CONSOLE OFF** отключает вывод на экран.

Команда "**?**" вызывает перевод строки перед выводом. При этом вывод начнется с крайней левой позиции, если не определено иного в шаблоне FUNCTION. Если не определено никакого выражения, то просто выводится пустая строка.

Команда "**??**" выводит результат в текущей строке на текущей позиции экрана.

Коды функций и шаблонов те же, что и для @...SAY...GET. Дополнительный код:

V"число" – реализует вывод на экран в форме вертикальной полосы текста, ограниченной заданным числом горизонтальных позиций по ширине.

Опция "**AT**" может использоваться для определения номера колонки, с которой производить вывод, что удобно для создания таблиц.

ПРИМЕР:

```
? 15 * (10+10)
?? 1/7 PICTURE('##.#####') AT 4
? 'Вертикаль' FUNCTION 'V1' 'Квадрат' FUNCTION 'V3' AT 9
```

Команда вывода блока текста:

TEXT

сообщение

ENDTEXT

осуществляет вывод "сообщения" один к одному. Удобно для вывода больших блоков текста типа Help.

Команда вывода строк текста \ или \\.

Осуществляет вывод строк текста, результатов выражений и функций, содержимого переменных памяти. Текст можно выводить в файл для создания писем или программ.

Команда "\\" осуществляет (в отличие от "\") переход на новую строку. Выводимый текст может содержать выражения в разделителях << >>. Вывод значения выражения происходит при установке **TEXTMERGE ON**.

ПРИМЕР:

```
SET TEXTMERGE TO letter.txt NOSHOW
USE base_1
SET TEXTMERGE ON
\ <<DAY(DATE( ))>>, <<CMONTH(DATE( ))>>
\\                          <<YEAR(DATE( ))>>
\
\  Дорогой <<name>>  <<fam>> !
\
\        Текст письма
\
MODIFY FILE  letter.txt  NOWAIT
```

Часто применяется вывод информации в виде псевдографических диаграмм с использованием функции

REPLICATE ("набор символов","числовое выражение"),

организующей повтор заданного набора символов столько раз, сколько указано в числовом выражении.

ПРИМЕР:

```
USE base_z
CALCULATE MAX(zar) TO mz
 SCAN
   ? REPLICATE("_", ROUND(60*zar/mz,0))
   ?? fam AT 61
 ENDSCAN
```

Условный оператор:команда IF

Команды управления являются важнейшим средством построения программ.

IF <выражение>
<команды>
[ELSE <команды>]
ENDIF

Допускаются вложенные IF, для каждого из них должно присутствовать ENDIF.

ПРИМЕР.

```
IF WOSR <18
```

@ 10,12 SAY 'ЮНЫЙ ДРУГ'
@ 12,8

Команда выбора

Команда выбора имеет формат:

DO CASE
CASE <условие>
< команды>
CASE < условие >
< команды>
.....
ENDCASE

Проверяется истинность одного из выражений, выполняются операторы, относящиеся к блоку, где выражение истинно. Команда CASE может заменить выражение IF.

Организация цикла

Для организации циклов применяется структура с предусловием:

DO WHILE <выражение>
< команды >
[LOOP]
< команды >
<EXIT>
ENDDO

Цикл выполняется до тех пор, пока выражение истинно. По EXIT выход из цикла и передача управления оператору, следующему за ENDDO. Команда LOOP дает переход на начало цикла, пропуска следующих за ним команд до ENDDO.

Добавление записей из программы

Добавлять записи в таблицу из программы можно разными способами в зависимости от источника записей.

*Программа добавляет записи в таблицу и непосредственно обновляет значения полей

SELECT <таблица>
APPEND BLANK

*Вызываем предварительно созданную форму, в которую пользователь может вводить *значения полей

DO FillForm

Однако не всегда удобно непосредственно редактировать поля таблицы. В приведенном ниже примере программы из таблицы создается набор переменных памяти, которые после ввода их значений, сохраняются в таблице по запросу.

SELECT <таблица>

```
SCATTER MEMVAR MEMO
SAVE
Savelt = FillForm()
*Если пользователь кликнет на кнопке SAVE, то добавляется пустая запись и полям
*присваиваются значения соответствующих переменных памяти
IF Savelt
  APPEND BLANK
  GATHER MEMVAR MEMO
ENDIF
```

Второй фрагмент кода лучше первого, потому что в нем не добавляется новой записи в таблицу до тех пор, пока пользователь не решит сохранить данные.

Добавление записей из других таблиц

Нужно добавить записи в текущую таблицу из другой таблицы. Можно читать по одной записи, сохранять значения полей в переменных памяти, а затем добавлять эти значения в запись второй таблицы:

```
SELECT EMPLOYEE
SCAN
  SCATTER MEMVAR
  SELECT EMPL9
  APPEND BLANK
  GATHER MEMVAR
  SELECT EMPLOYEE
ENDSCAN
```

Вместо того, чтобы «разбрасывать» значения по переменным памяти, можно использовать массив для сохранения значений полей из одной записи.

```
SELECT EMPLOYEE
SCAN
  SCATTER TO EMPLOYEE
  SELECT EMPL9
  APPEND BLANK
  GATHER FROM EMPLOYEE
  SELECT EMPLOYEE
ENDSCAN
```

Если две таблицы имеют одинаковую структуру, то можно поступить гораздо проще. Для этого следует воспользоваться командой

APPEND FROM Добавляет записи в текущую таблицу из другого файла.

При выполнении следующих команд все записи из таблицы CURPROD добавляются в таблицу PRODHITS:

```
SELECT PRODHITS
APPEND FROM CURPROD
```

Если из таблицы CURPROD требуется добавить только те записи, у которых значение поля lInProduction равно .F., используйте следующие строки программы:

```
SELECT PRODHITS
APPEND FROM CURPROD FOR NOT lInProduction
```

С помощью члена FIELDS можно даже указать и перечень добавляемых полей. Член FIELDS определяет поля, в которые добавляются записи, а не поля, из которых эти значения копируются. Более того, добавляемые поля должны иметь одинаковые имена и параметры в обеих таблицах.

5. СОЗДАНИЕ ФОРМ В VISUAL FOXPRO

Новая форма имеет следующие функциональные возможности:
1) установка свойств формы,
2) добавление средств управления,
3) установка свойств средств управления,
написание программы обработки события.

Создание новой формы.

1.Установка опций формы
Tools – Option – вкладка Forms:
Grid Lines (Линия сетки),
Snap to Grid (Перемещение по ячейкам).
Horizontal Spacing (Интервал по горизонтали)
Vertical Spacing (Интервал по вертикали)
Show Position – выключить
Tab Ordering (Упорядочение таблицы) – by list (по списку)
Scale Units (Единица масштаба) – Pixels Maximum Design Area (Максимальная площадь проектировки)

2. Использование Form Wizard для начала создания новой формы.

Step 1. Выбор полей
Step 2. Style

3. Список свойств и методов средства управления

List Box Builder (Построитель окон списка) выберите его, затем правой кнопкой выберите Builder из контекстного меню.

Step 1. – выбор полей

Step 2.- стиль

Step 3.- размещение

Step 4. – имя поля.

6. ПОИСК В БАЗЕ ДАННЫХ

Фильтрация базы данных

После открытия или создания новой БД можно работать не со всей БД, а с ограниченным числом полей и записей. Для этого после ввода команд меню Database, SetUp устанавливаются поля для доступа: [X]Set Fields и фильтр для записей: [X]Filter.

Для задания полей для доступа активизировать Set Fields. Появляется новое окно с форточками: поля БД (Database Fields) и поля доступа (Selected Fields). Для перемещения <Move> поля нужно выбрать имя нужного поля в форточке Database Fields и нажать Enter. Для исключения поля из доступа – то же в форточке Selected Fields.

В окне установок SetUp можно включить (On) или выключить (Off) назначенную установку полей. По умолчанию все поля БД доступны для редактирования и просмотра.

Для установки фильтра включить кнопку Filter. В новом окне имеется панель для записи выражения (Expression) с подсказкой допустимого типа: <expL>, кнопками справок функций и операторов: Math, String, Logical, Date. Из форточек имен полей (Field Names) и переменных (Variables) можно перемещать имена в панель для записи выражений. Для этого необходимо установить курсор на имени поля и нажать клавишу Enter.

При использовании команд обращения к базе можно создать избирательное действие с помощью FOR или организовать постоянный фильтр выбора записей:

SET FILTER TO <выражение> <ВК>

SET FILTER TO <ВК> && снять фильтр

ПРИМЕР

SET FILTER TO SUMMA >50 <ВК>

LIST <ВК>

&& Распечатаются записи с полями SUMMA > 50

Организация поиска информации в базе данных

Поиск самая распространенная операция в системе обработки данных. Важнейшим элементом любой системы управления базами данных является

наличие средств поиска данных. Ускоренный поиск обычно реализуется введением индексных файлов, организацией прямого доступа к данным.

Последовательный поиск

В программе возможна организация удобного для пользователя поиска данных в базе. Команда LOCATE осуществляет последовательный поиск одной самой первой записи в базе данных, удовлетворяющей заданному FOR-условию среди записей, находящихся в заданных границах, и до тех пор, пока соблюдается WHILE-условие.

LOCATE [<границы>] [**FOR** <выражение>] [**WHILE** <выражение>]

Границы поиска могут быть заданы следующими параметрами:

ALL – поиск по всем записям базы;

NEXT N - поиск по следующим указанным записям номеров до N;

RECORD N - поиск указанного номера записи;

REST - поиск, начиная с текущей, и до последней записи;

FOR и WHILE – устанавливает условие поиска, например, поиск с указанным именем поля;

Команда продолжения поиска записей, начатого ранее командой LOCATE:

CONTINUE

В результате работы команды указатель записи переходит на следующую запись, удовлетворяющую условиям поиска в команде LOCATE.

Для поиска всех записей по заданным условиям команду CONTINUE нужно поместить в цикл. Если командой LOCATE или CONTINUE не было найдено нужных записей, указатель записи устанавливается на конец файла.

При успешном поиске, когда указатель устанавливается на найденную запись, можно определить номер этой записи с помощью функции RECNO(), Функция FOUND(), оценивающая результат поиска, принимает значения «Истина» (.true.).

ПРИМЕР. Поиск по указанному значению поля KOMU.

LOCATE FOR KOMU ="ALPHA"

В результате работы команды указатель записи переходит на первую запись, в которой поле KOMU имеет значения ALPHA. Если значения не найдено, то указатель переходит на конец файла.

Ускоренный поиск

Индексный файл не только упорядочивает базу данных для просмотра, но и ускоряет поиск в ней по ключу, заданному в индексе. Применяется специальный алгоритм ускоренного поиска, в котором база просматривается не сплошь, а в соответствии с информацией, содержащейся в индексе. Для поиска с использованием индексного файла применяется команды SEEK <ключевое выражение> или FIND <ключевое выражение>. По команде FIND можно осуществить поиск только символьных полей. Чтобы искать и символьные, и числовые поля, применяется команда SEEK. В результате работы команды устанавливается указатель на номер записи в активном файле с активным индексным

файлом, ключ индекса которого совпадает с поисковым. Так как база данных упорядочена, то следующая запись будет соответствовать искомой записи, имеющей нужное значение ключа, тогда для перехода на следующую запись можно воспользоваться командой:

SKIP [число] "число" – на такое количество записей нужно передвинуть системный указатель. Для поиска по заданным условиям всех записей в базе команду SKIP нужно поместить в цикл.

ПРИМЕР

USE RASH INDEX KUDA && открыли основной и индексный файл

FIND TE && перемещается указатель на запись, где ключ имеет начало "TE"

DISPLAY && распечатка записи первой записи, имеющей нужное значение ключа.

ПРИМЕР

SEEK 'A89'

DISP

Символьные поля выделяются при этом апострофами.

Для поиска записей в индексированной БД через главное меню применяются команды главной строки меню Record, Seek. В панели указывается запись для поиска в соответствии с индексным ключом, например: для ключа POL+FAM запишем 'м С' – поиск мужчины с фамилией на букву "С" (число позиций до буквы "С" должно быть равно длине поля POL).

7. РАБОТА С НЕСКОЛЬКИМИ БАЗАМИ

Одновременная работа с несколькими БД позволяет получать сведения, хранящиеся в разных базах и принадлежащие одним и тем же объектам.

Например, паспортные данные и сведения о месте работы могут храниться в разных базах данных. При извлечении всех данных необходимо воспользоваться одновременно несколькими БД. Одна из баз называется старшей, остальные – младшими. Записи в базах данных должны иметь общее поле (например, FAM), и младшие базы индексируются по этому полю.

Откроем БД. Для этого необходимо в окне меню View поставить курсор на рабочие области Work Areas: A, B, C... и, нажимая Enter или SpaceBar, последовательно открыть все нужные БД по одной в каждой области.

Далее проиндексируем младшие базы по полю FAM.

Зададим старшую базу: выбрать имя старшей базы, нажать SpaceBar и включить кнопку <Relations>.

Затем отметим младшую базу: выбрать имя младшей базы, нажать SpaceBar и задать поле связи FAM.

Для старшей базы снова включим кнопку <Relations>, отметим следующую базу, зададим поле связи и т.д. В форточке Relations будет отмечаться связь параллельного типа. Последовательный тип применяется при связи по нескольким полям (в виде каскада).

Для вывода данных применяются команды (в окне Command):

Display all, List, Browse Fields со списком полей, записи в которых выводятся на экран. Указываются составные имена полей ("имя рабочей области_имя поля" либо "псевдоним_имя поля"), например: List A.FAM, B.FAM, B.POL, C.FAM, C.DATA_R.

ПРИМЕР

```
SELECT A
USE BRIG1
SELECT B
USE KADR INDEX KADRTAB IN B
SELECT A
SET RELATION TO BRIGTAB INTO B
LIST A.FAM, B.POL,  A.DATA_R.
```

Связь вида одна_запись_с_одной

Связь вида одна_запись_с_одной – команда связывает указатель записи в активной рабочей области с указателем записи из других рабочих областей, имена которых указаны после слова INTO, по заданному общему ключевому полю.

Формат команды:

SET RELATION TO <КЛЮЧ> INTO <ОБЛАСТЬ>[,<КЛЮЧ>INTO <ОБЛАСТЬ>] [ADDITIVE]

Связь вида одна_запись_со_многими

Связь вида одна_запись_со_многими – при этом с каждой записью из старшей базы могут быть сцеплены несколько записей из младшей базы. Связь может быть установлена сразу с несколькими базами, находящимися в разных областях.

Формат команды:

SET SKIP TO [<ОБЛАСТЬ1>],[<ОБЛАСТЬ2>]

Прежде чем использовать команду SET SKIP TO, необходимо выполнить начальное сцепление вида связь_одна_запись_с_одной командой SET RELATION. УДАЛЕНИЕ связи вида одна_запись_со_многими осуществляется командой SET SKIP TO без параметров.

Рассмотрим пример организации связи одной базы с несколькими базами, причем эта связь будет реализована по схеме одна_запись_со_многими. Рабочие могут работать сразу в нескольких бригадах. Требуется предъявить все выработки для каждой фамилии из базы KADR.DBF.

ПРИМЕР.

```
SELE A
USE KADR IN A
SELE B
USE BRIG3 IN B INDEX BRIG3
SELE C
USE BRIG5 IN B INDEX BRIG5
SELE A
```

```
SET RELATION TO TAB INTO B, TAB INTO C
SET SKIP TO B,C
BROWSE FIELDS A.FAM:H=’ФАМИЛИЯ’,; A.TAB:H=’ТАБЕЛЬ’,
B.TAB:H=’БРИГ.3’, B.VIR:’ВЫР.’,;
C.TAB:H=’БРИГ.5’, C.VIR:’ВЫР.’
SET RELATION TO
```

Результат работы программы:

ФАМИЛИЯ	ТАБЕЛЬ	БРИГ.3	ВЫРА-БОТКА	БРИГ.5	ВЫРА-БОТКА
ЕФИМОВ	446	446	280.60	446	50.00
		446	130.00	446	80.00
				446	100.00
ЛАРИОНОВ	321	321	25.70	321	650.00
		321	60.00		

8. УПРАВЛЕНИЕ ДАННЫМИ

Что делать, если окажется, что нужно просмотреть информацию заданного типа? Например, может потребоваться узнать все о покупателях, израсходовавших свыше $5000, или узнать лишь имя и номер каждого покупателя. Visual FoxPro позволяет быстро извлекать и просматривать информацию, отвечающую заданным условиям. Процесс извлечения указанной информации называется "запросом".

Как открыть окно запроса

В меню "Файл" выберите команду "Открыть" и в каталоге FOXPROW\TUTORIAL\ выберите файл CUSTOMER.DBF и нажмите кнопку "Открыть". Затем в меню "Файл" выберите команду "Создать". В диалоге "Создать файл" установите опцию "Запрос", а затем нажмите кнопку "Создать". Появится окно запроса.

В окне запроса указывается информация, которую требуется извлечь из таблиц. Извлеченная информация помещается в окно просмотра. Можно работать с этим окном так, как это делалось в предыдущей главе.

В списке "Поля результата" окна запроса указывается, какие поля требуется отобразить в окне просмотра. По умолчанию во вновь открываемой таблице отображаются все поля.

В области "Критерий отбора" указывается, какие записи требуется извлечь из таблицы. Если область пуста - запрос извлекает все записи из таблицы.

Допустим, нет необходимости просматривать поля CNO, ADDRESS, ZIP и ONO. Тогда нам нужно указать только те поля, которые нужны.

Как задать поля для результата запроса

В окне запроса установите флажок "Поля". Появится диалог "Выбор полей запроса".

Нажмите кнопку "Удалить все", чтобы очистить список "Выбранные поля". И поместите в список "Выбранные поля" COMPANY, CNO, CONTACT, PHONE, CITY, STATE и YTDPURCH, для этого выделите нужное поле в списке "Поля таблицы" и нажмите кнопку "Добавить". Каждое из перечисленных выше полей появится на экране с префиксом CUSTOMER, это означает принадлежность поля к таблице CUSTOMER. По окончании нажмите кнопку "Ok". Выбранные поля будут отображены в списке "Поля результата" окна запроса. При выполнении запроса поля в окне просмотра появляются в порядке, указанном в окне запроса.

Как выполнить запрос

В окне запроса нажмите кнопку "Выполнить". И у вас появится окно просмотра с выбранными полями.

Если в процессе работы с окном просмотра окажется, что больше нет необходимости в поле CONTACT, то его просто из запроса удалить.

Как удалить поле результата из запроса

Закройте окно просмотра, в котором отображен результат запроса. Установите флажок "Поля". Удалите поле CONTACT из списка "Выбранные поля", выделив его в списке и нажав кнопку "Удалить". Нажмите кнопку "Ok". Обратите внимание, что поля CONTACT больше нет в списке "Поля результата". И выполните запрос.

Теперь, когда выбраны поля результата, можно задать порядок, в котором будут представлены данные. Например, можно отобразить данные по городам в алфавитном порядке.

Как упорядочить поля в запросе

Закройте окно просмотра с отображенным результатом запроса. В окне запроса установите флажок "Порядок". В диалоге "Порядок полей запроса" поместите CUSTOMER.SITY в список "Критерий порядка", для этого выделите его и нажмите кнопку "Добавить". И установите флажок "По возрастанию".

Для возвращения в окно запроса нажмите кнопку "OK". Обратите внимание, что в списке "Поля результата" рядом с CITY появились символы 1 и стрелка вверх. Цифра 1 означает, что CITY имеет первый приоритет при упорядочении данных, а стрелка вверх означает, что данные отображаются в порядке возрастания.

Нажмите кнопку "Выполнить". Данные можно упорядочить любым способом. Например, можно упорядочить данные по штатам и по городам внутри штата.

Как упорядочить по нескольким полям

Закройте окно просмотра. Установите флажок "Порядок". Поместите CUSTOMER.STATE в список "Критерий порядка". Установите порядок полей, для этого мышью переместите кнопку слева от STATE так, чтобы поле STATE было выше поля CITY. Нажмите кнопку "Ok". Теперь перед STATE стоит цифра 1, а перед CITY - 2, что означает приоритет по упорядочению данных. И нажмите кнопку "Выполнить".

До сих пор в таблице CUSTOMER просматривались все записи. Но предположим, что нужно просмотреть только те компании, которые расположены в штате Огайо. Используя область "Критерий отбора" окна запроса, можно выполнить такой запрос почти сразу же.

Задание условий отбора

Чтобы извлечь только заданные записи, следует задать условия отбора. Каждое условие отбора состоит из поля таблицы, варианта сравнения и значения, которое должно сравниваться с содержимым поля.

Как задать условия отбора

Закройте окно просмотра. В окне запроса щелкните на прямоугольнике ниже "Имя поля", чтобы вывести список "Имя поля". Выделите CUSTOMER.STATE

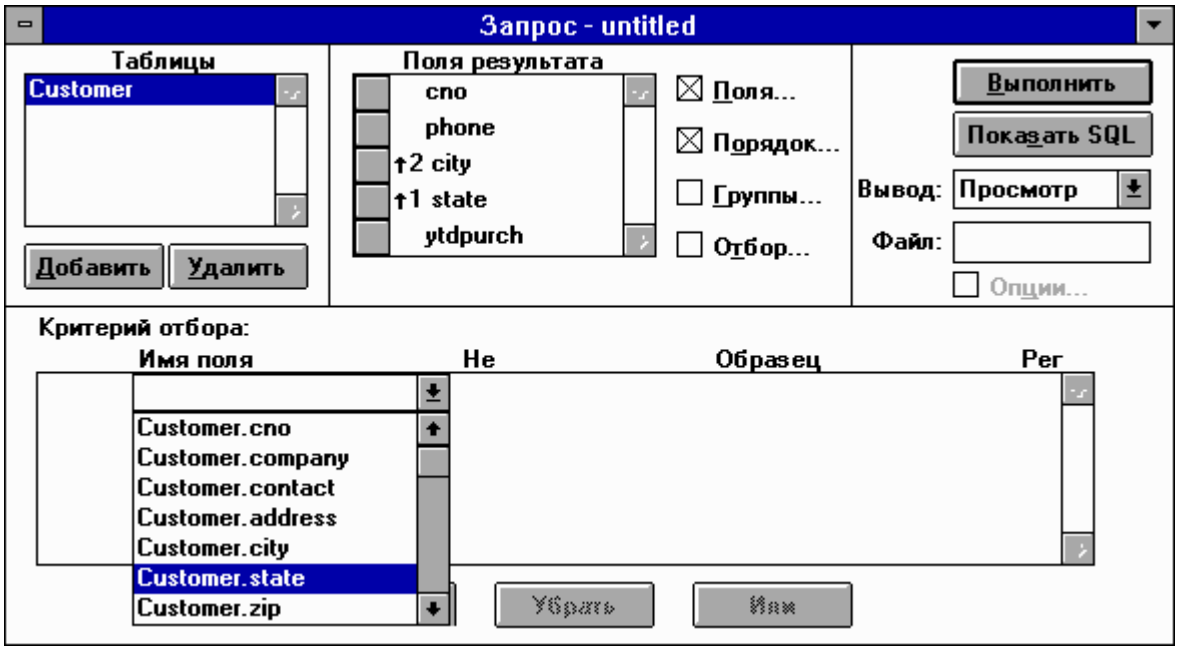


Рис. 8.1. Как задать условия отбора

В списке вариантов сравнения по умолчанию устанавливается "похоже на". Наберите OH (Ohio) в текстовом поле ниже надписи "Образец". Так создано условие отбора для просмотра записей только о компаниях в штате Огайо. Таким же образом можно задать несколько условий отбора. Введенное условие отбора требует, чтобы поле STATE было похоже на OH. STATE - это поле, "по-

хоже на" - это вариант сравнения, а OH - значение, с которым сравнивается содержимое поля. Вариант сравнения "похоже на" указывает, что поле STATE должно иметь значение OH для того, чтобы запись удовлетворяла требованиям запроса.

Другие варианты условия отбора

Попробуем вывести компании, расположенные в штате Огайо или Мичиган. Закройте окно просмотра. Щелкните на поле ввода вариантов сравнения и выберите значение "в списке". В поле "Образец" наберите MI (Michigan) после OH. Не забудьте поставить запятую.

Нажмите кнопку "Выполнить". Появится окно просмотра, в котором будут выведены все компании, расположенные в штатах Огайо и Мичиган, потому что "в списке" означает, что для включения в результат STATE должно содержать один из элементов, указанных в текстовом поле "Образец".

Другие условия отбора: равно, больше, меньше, в пределах; работают аналогично. Для более подробного ознакомления обратитесь к справочнику: "Руководство пользователя. FoxPro. Система управления реляционными базами данных для Windows".

Как открыть запрос

В меню "Файл" выберите команду "Открыть". В появившемся окне "Открыть" выберите каталог TUTORIAL. В списке "Тип" выберите "Запрос" Выберите нужный файл (GSCHAP5.QPR) и нажмите кнопку "Открыть".

В качестве первого отчета составим список телефонных номеров всех клиентов из Калифорнии. Необходимо сформировать отчет с телефонным списком. Включите следующие поля: COMPANY, PHONE, CONTACT, CITY и STATE. Критерий отбора: STATE похоже на CA. И главное, в списке "Вывод" выберите значение "Отчет/Этикетка". Нажмите кнопку "Выполнить".

Отчет начинает прокручиваться на экране. После просмотра отчета нажмите клавишу ESC. Вышеприведенный отчет содержит корректную информацию, но его наглядность можно повысить.

9. ФОРМИРОВАНИЕ ОТЧЕТОВ

Ранее говорилось об отображении информации в окне просмотра с помощью запроса. Сейчас описывается, как использовать окно запроса для быстрого составления отчетов на основе имеющихся данных. Другим способом представления информации является отчет. Для создания отчета необходим запрос. Предполагаем, что запрос создан.

Как сформировать бланк отчета

Установите флажок "Опции". В диалоге "Параметры ввода запроса" выберите опцию "Отчет", установите флажок "Стандартный отчет", и в диалоге "Стандартный отчет запроса" в поле "Сохранить как" наберите имя gschap6.frx.

Нажмите кнопку "Ok" и кнопку "Выполнить".

Полное изображение целой страницы отчета можно увидеть в окне "Просмотр страницы", как показано на рис. 9.1.

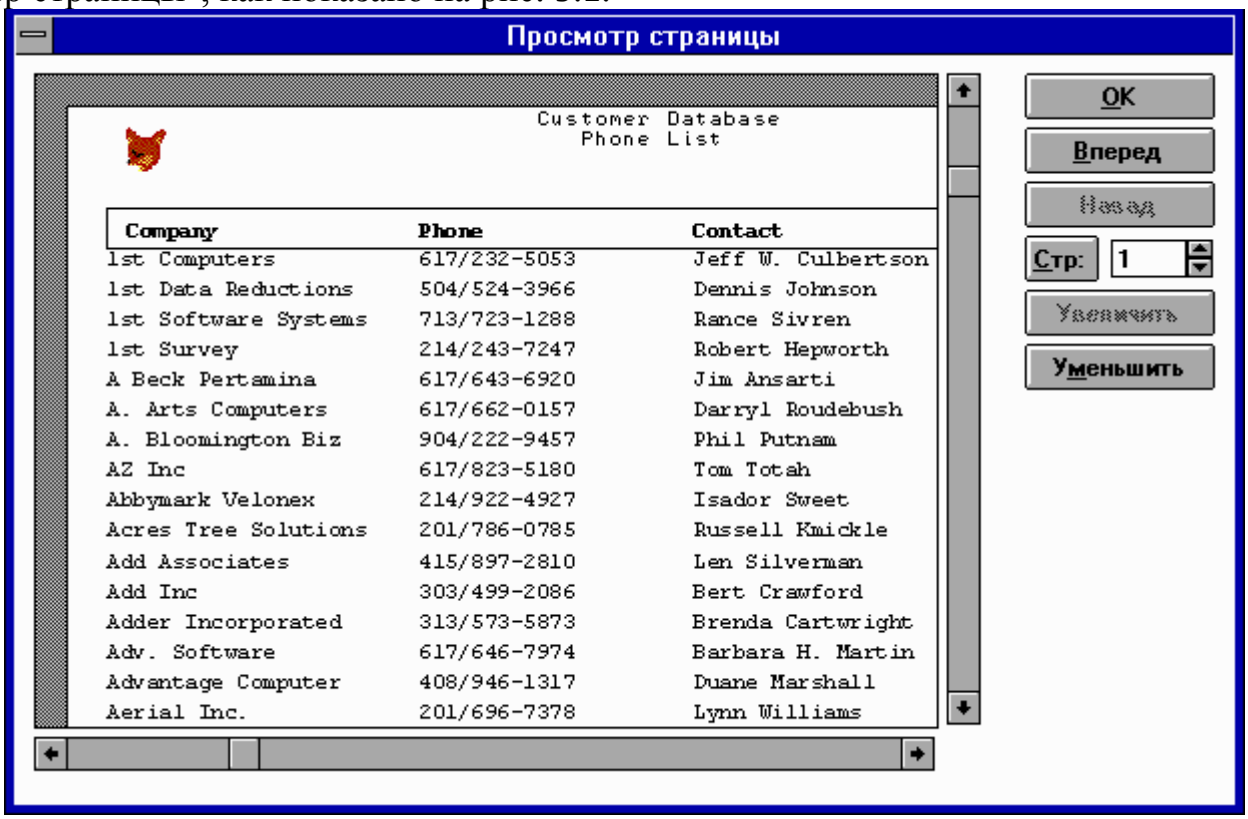


Рис. 9.1. Изображение страницы отчета

Изображение страницы отчета дает представление о том, как отчет будет выглядеть на бумаге.

В окне "Просмотр страницы" Вы можете увеличить изображение (кнопка Увеличить), можно уменьшить (кнопка Уменьшить), просмотреть следующую или предыдущую страницу (кнопки Вперед/Назад). Для выхода нажмите кнопку "OK".

Теперь рассмотрим созданный бланк отчета и внесем в него некоторые усовершенствования в окне разметки отчета.

Как открыть бланк отчета

В меню "Файл" выберите команду "Открыть". В списке "Тип" выберите значение "Отчет" и откройте файл: GSCHAP6.FRX. В окне разметки отчета появится бланк отчета GSCHAP6, изображенный на рис.9.2.

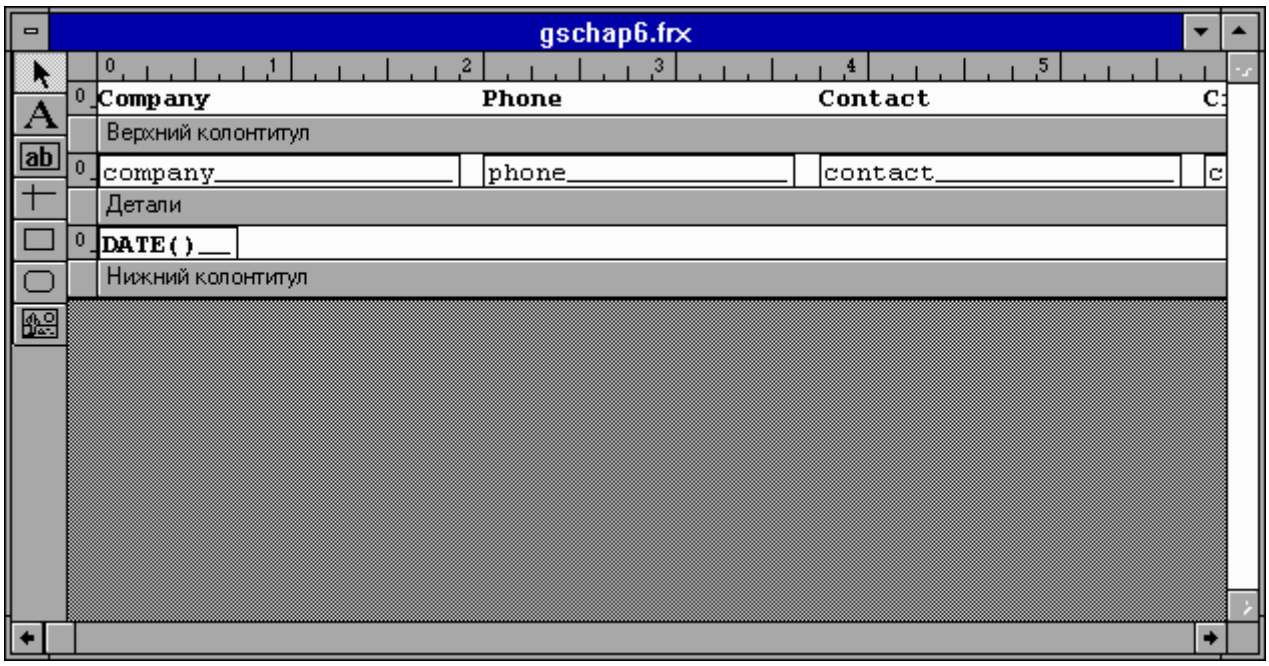


Рис. 9.2. Бланк отчета

Бланк отчета разделен на три полосы:

1 Полоса "Верхний колонтитул" содержит информацию, отображаемую вверху каждой страницы отчета.

2 Полоса "Детали" печатается один раз для каждой записи, удовлетворяющей критерию отбора.

3 Полоса "Нижний колонтитул" содержит информацию, отображаемую вниз каждой страницы отчета.

С помощью набора значков в левой части окна разметки можно внести необходимые изменения в отчет. Например, на рис.9.3 показан бланк отчета с внесенными изменениями.

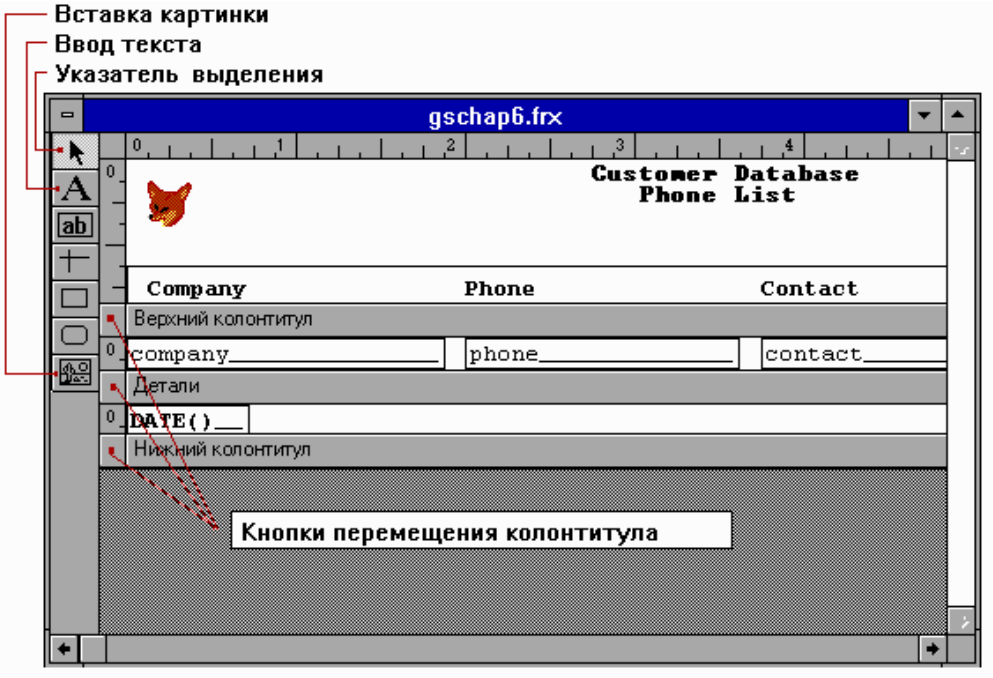


Рис. 9.3. Внесение изменений в отчет

Построение отчетов

Для построения отчета необходимо открыть БД и ввести команды меню: File, New, (*)Report. На экране появляется окно с бланком отчета и меню:

Report - команды построения отчета

- Page Layout – окно меню форматирования бланка отчета
- Page Preview ^I – предварительный просмотр отчета
- Data Grouping – группирование данных отчета
- Title/Summary – создание заголовка/заключения отчета
- Variables – создание новых переменных
- Box ^B – режим построения линий и рамок в отчете
- Field ^F – режим построения выражений в отчете
- Text ^T – режим набора текста в отчете
- Add Line ^N – добавить строку в бланк отчета
- Remove Line ^O – удалить строку из бланка отчета
- Bring to Front ^G – прямой порядок обхода отмеченной группы
- Send to Back ^J – обратный порядок обхода группы
- Center – центрирование группы
- Group –формирование группы отмеченных объектов
- Ungroup – отмена группировки отмеченных объектов
- Quick Report окно меню быстрого построения отчета

Редактирование бланка отчета удобно начинать с команды **Quick Report**. В окне меню зададим (*)Column Layout - для размещения содержимого полей колонками, либо (*)Form Layout - строками. Для вывода заголовков с именами полей включим [X]Titles. Выбор полей, используемых в отчете, производится кнопкой [X]Fields. Далее начнем работать с командами меню Report.

Команда ***Page Layout*** открывает окно меню с кнопкой <Options>, содержащей установки:

- прогон страницы (Page eject) до/после печати отчета,
- печать без заголовков страницы (Plain page),
- печать только заголовков и заключений (Summary report),
- не печатать пустые строчки (Suppress blank lines),
- добавлять псевдоним к имени поля (Add alias),
- число строк на странице (Page length),
- отступы сверху и снизу (Top/Bottom margin),
- левый отступ (Printer indent),
- число колонок в отчете (Right margin) .

Заданные параметры можно сохранить <Save> и обновить <Restore>.

Команда **Data Grouping** позволяет объединять данные в группы, например, по полю City_r - жителей различных городов. Желательно упорядочить записи в БД, проиндексировав с нужным ключом.

В окне меню группировки можно задать:

- Add - добавление новой группы,
- Change - внесение изменений,

– Delete - удаление группы,

– Group - включить построитель выражений для формирования групп.

Набрав имя поля, например City_r, получим группы данных по городам. Каждая группа может:

- начинаться с новой страницы (New Page);
- иметь при печати свой заголовок/заключение (Swap Page Header/Footer вместо страничных PgHeard/PgFoot) либо дополнительный (Reprint Header);
- иметь свою нумерацию страниц (Reset Page Number) и пустые строки, добавляемые в заключение (# of rows following header).

Команда **Title/Summary** открывает окно с кнопками создания строк заголовка/заключения для всего отчета(Title/Summary Band) и печати заголовка/заключения на отдельных страницах (New Page).

Команда **Variables** позволяет добавлять (Add) переменные для вычислений в отчете. В окне меню можно задать:

- имя переменной (Variable Name),
- выражение для расчета (Value to Store),
- начальное значение (Initial Value), границы расчета (Reset),
- функцию (Calculate), где можно выбрать следующее:
- расчет числа записей (Count),
- сумма значений (Sum), среднее арифметическое (Average),
- наименьшее/ наибольшее значение (Lowest/Highest),
- среднеквадратичное отклонение (Std.Deviation),
- дисперсия (Variance).

НАПРИМЕР, для расчета числа жителей города введем переменную: N_city, начальное значение: 0, выражение и границы расчета: CITY_R, функция: Count.

Команда **Box** позволяет вывести в отчете линию или рамку. Установив курсор на рамку и нажав Enter, в окне меню зададим тип линии: одинарная/двойная/панельная/символьная. Установив курсор на рамку и нажав пробел, можно перемещать рамку курсором, ^SpaceBar – изменять размеры рамки. Для фиксации - Enter.

Команда **Field** позволяет ввести в бланк отчета названия полей и переменных, а также выражения. Здесь же задается:

– Format – формат данных,

– Style – стиль шрифта,

– Calculate – функция для расчета,

– Comment – комментарии в окне меню,

– Suppress – подавление повторяющихся данных (вывод только первой записи из одинаковых).

НАПРИМЕР, поместив курсор в строку бланка отчета PgHead, введем: "стр". Далее по ^F в построителе выражений выберем из форточки переменную _PAGENO. В отчете будут проставляться номера страниц.

В строке бланка отчета Detail в окне меню нужно определиться с кнопками:

Stretch Vertically - задает вывод длинных записей поля (например, Memo) в несколько строк установленной ширины Width.

Float as Band Stretches – позволяет разместить содержимое полей, расположенных в бланке отчета, ниже полей, отмеченных [X]Stretch Vertically.

Поставив курсор в бланке отчета на панель с именем поля или переменной, можно нажатием клавиши Enter вызвать окно меню, SpaceBar – переместить панель, ^SpaceBar - изменить размер.

Команда **Group** позволяет обьявлять группы отмеченных объектов (не путать с Data Grouping). Для создания группы необходимо отметить нужные надписи в бланке отчета (имена полей, переменные, заголовки и т.п.), нажав SpaceBar вне надписи, курсором заключить надписи в рамку и - SpaceBar/Enter. Либо нажать Shift-SpaceBar и, не отпуская клавиши Shift, перемещаться курсором и помечать надписи клавишей SpaceBar, затем ввести команду меню Group. Ungroup - отменить объявление объектов группой.

Создание визиток

Для создания визиток, этикеток или карточек необходимо открыть БД и ввести команды меню: File, New, (*)Label. На экране появляется окно с бланком визитки и меню:

Label – команды создания визиток

- Page Preview ^I – предварительный просмотр визиток на экране
- Expression ^E – построитель выражений в визитке
- Environment ^N – сохранить параметры среды создания визиток
- Style ^Y – выбор типа шрифта и размещения текста
- Add Alias – добавлять к именам полей псевдоним
- Layout ^L – выбор формата визитки из списка
- Save Layout – сохранить новый формат визитки
- Delete Layout – удалить формат визитки из списка

В окне с бланком визитки показаны размеры (в поле Remarks) и расположение визиток на листе. Предлагаемые установки можно изменить и сохранить (команда Save Layout меню Label). Для задания готовых размеров визитки запустить команду Layout.

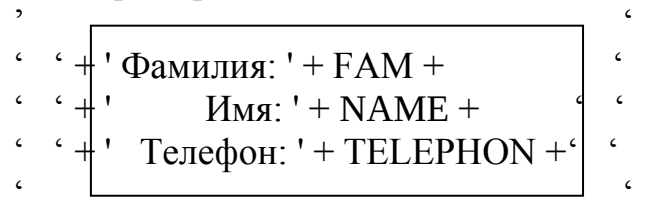
Можно изменять следующие размеры визитки:

- левый отступ (Margin),
- высота визитки (Height),
- ширина визитки (Wight),
- расстояние между визитками по ширине листа (Space Between),
- расстояние между визитками по высоте листа (Lines Between),
- число визиток по ширине листа (Number Across).
- Команда Style позволяет задать тип шрифта:
  - утолщенный (Bold),
  - наклонный (Italic),
  - подчеркнутый (Underline),

- верхний индекс (Superscript),
- нижний индекс (Subscript),
- выравнивание границ текста: по левому краю, по правому краю, по обоим краям.

При заполнении в бланке визитки указываются имена полей, данные из которых используются, функции, переменные и константы, соединенные в каждой строчке знаком "+".

Пример *РЕДАКТИРОВАНИЯ БЛАНКА ВИЗИТКИ*:



ПРИМЕЧАНИЕ: ровный правый край получается при одинаковой длине полей FAM, NAME, TELEPHON или подстраивается изменением количества пробелов в кавычках.

10. ОСНОВЫ ЯЗЫКА СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

Как выполнить запрос

Сверните окно отчета. Нажмите кнопку "Выполнить". Обратите внимание, что отчет содержит внесенные изменения. И этот отчет Вы можете распечатать на принтере.

SQL – язык запросов, основан на наборах записей. Он не имеет никаких команд для отображения или управления информацией. SQL предназначен для хранения и поиска данных.

Для создания запроса можно использовать окно Command или исходный текст программного модуля. В результате запроса получается файл, содержащий данные, удовлетворяющие условию запроса. В SQL выбирается и обрабатывается множество записей, к примеру, результатом команды SEEK является конкретная запись.

Команда SELECT языка SQL

1. **Запрос данных**

Select * <имя таблицы>

В результате выполнения команды создается файл, содержащий все поля данной таблицы.

2. **Запрос данных с использованием арифметических выражений**

Select <имя поля1…[, имя поля2]…[, арифметическое выражение] from <имя таблицы>

В результате выполнения команды создается файл, содержащий указанные поля из данной таблицы и результаты арифметических выражений.

3. **Отбор запросов с помощью условия WHERE**

Select * <имя таблицы> where <условие>

В условиях Where используются следующие предикативы поиска:

а) операторы сравнения (<,>,>=,<=,=,<>);

б) диапозона BETWEEN – предназначен для проверки на вхождения в диапазон требуемых значений:

Where <имя поля> between <значение1> and <значение2> ;

в) вхождения IN – определяет, есть ли заданное значение в списке

where <поле> in (значение1,…, значениеN);

г) сопоставление с образцом Like – так же, как в DOS, можно использовать символы маски (символы ‘ _’,’%’ аналогичны ‘?’,’*’ в DOS):

where <имя поля> like “C%” - выбор записей, у которых заданное имя поля начинается с буквы “C”;

д) неопределенное значение IS NULL - проверят неопределенное значение, которое не задавалось. Неопределенные значения означают, что имеются проблемы с записью или что данные должны быть отброшены.

Where <имя поля> not IS NULL - данное поле определено;

е) составные предикативы (not, and,or).

4. **Упорядочение результатов запроса ORDER BY**

Можно упорядочить результаты по заданному ключу:

Select * from <имя таблицы> order by <ключ>

5. **Операция объединения**

а) внутреннее объединение – двух и более таблиц по заданному условию:

select <псевдоним1>.<имя поля1 > [,… < псевдоним2>.<имя поляN>… <арифметические выражения>] **where** <условия объединения

т.е.<псевдоним1>.<имя поля1> = <псевдоним2>.<имя поля1 >>;

б) внешнее объединение – когда для связи между двумя и более таблицами используется команда WHERE:

where <псевдоним1>.<имя поля1> = <псевдоним2>.<имя поля1 >

6. Выполнение операций добавление, изменение и удаление

а) Добавление записи Inset:

Insert into <имя таблицы> (<имя поле1>[, имя поле2, …]) values (<значения1>[, значения2,…])

Добавляет запись с заданным значением для каждого поля.

б) Изменение записи update:

update < имя таблицы > set <имя поле1> = <значение>

в) Удаление записи delete:

delete from < имя таблицы > where <условия>

В результате действия этой команды записи, удовлетворяющие данному условию, будут помечены на удаление.

Просмотры

Просмотр – это способ визуализации данных.

1. Создание определения просмотра:

Create SQL view <имя просмотра> as select * from <имя таблицы>

2. Использование просмотра:

Use < имя просмотра >

Browse

Использование команды SQL и сохранение команды в виде просмотров избавляет от повторной операции набора команды.

Создание представления данных (Local views)

Главное отличие от запросов: запросы хранятся в отдельных файлах с расширением .qpr, а представление в самой базе данных, что позволяет обновлять данные, можно использовать в отчетах, формах и запросах.

Создать новое представление

- в системном меню File| New||View
- во вкладке Data | Local Views
- в окне Command: Create View

Можно использовать два режима: мастера (Wizard) или конструктора (Designer).

Диалоговое окно View Designer в режиме конструктора представлено на рис. 10.1.

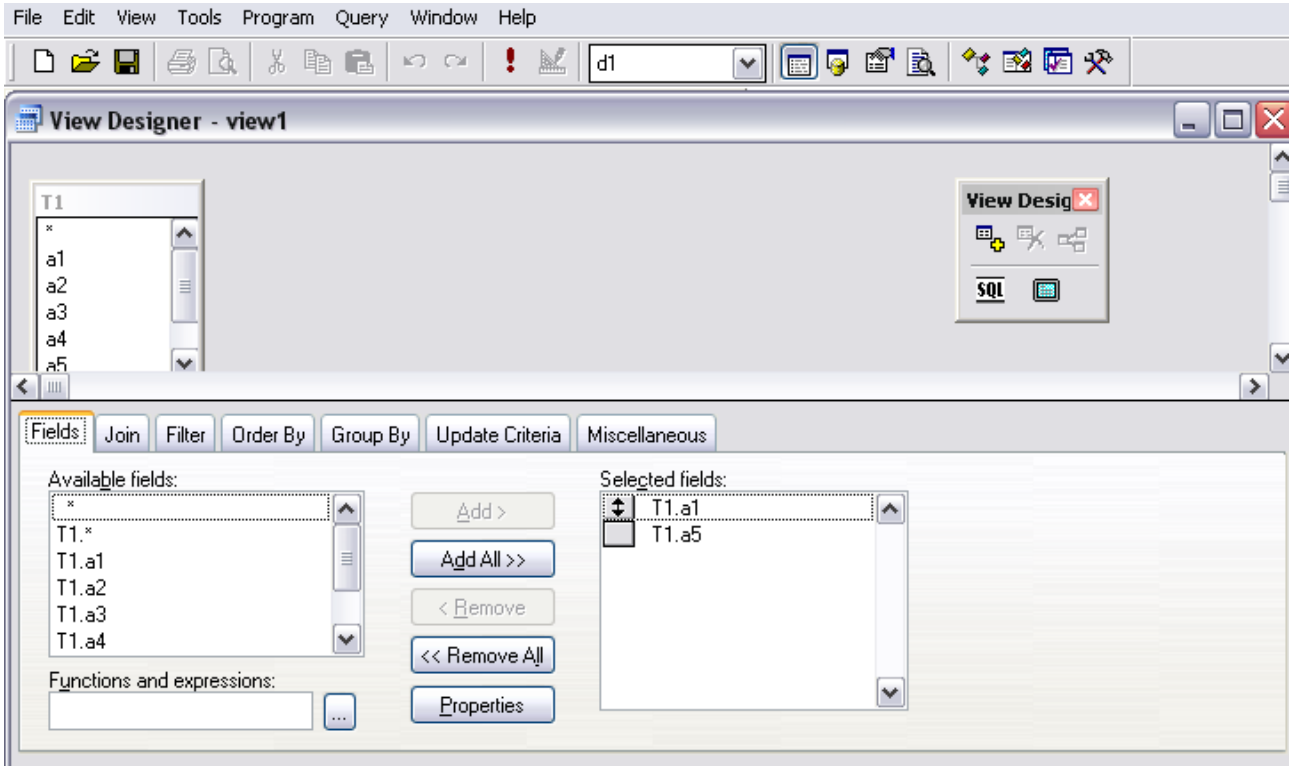


Рис. 10.1. Диалоговое окно View Designer

Вызов контекстного меню правой кнопкой мыши. Add Table позволяет добавить таблицы и представления.

Вкладки:

Fields – выбор полей и выражений

Join – задает условие объединение полей

Filter – задает условие выборки данных

Order by – задает порядок отображения записей

Group by – группирует данные по заданным полям

Update Criteria – содержит опции, связанные со способностью редактирования данных в таблице (Send SQL update)

Miscellaneous – дополнительные критерии

11.ОРГАНИЗАЦИЯ МЕНЮ

Организация FOX – меню

Мощный инструмент FoxPro – создание разнообразных меню. Простейшие из таких программ:

FOX-LIGHTBAR-меню.

Используется команда

@ X,Y PROMPT "строка меню" MESSAGE "сообщение"

ПРИМЕР. Демонстрация меню с подсветкой элементов

```
CLEAR
SET MESSAGE T0 18 CENTER
*&& Вывод сообщений в строку 18 по центру
@ 4, 31 SAY "Заголовок меню"
@ 7, 31 PROMPT "функция 1"
@ 9, 31 PROMPT "функция 2" MESSAGE "сообщение о функции 2"
@ 11,31 PROMPT "функция 3" MESSAGE "сообщение о функции 3"
@ 13,31 PROMPT "Завершение работы" MESSAGE "Выход"
MENU TO R
*&& Переменной R присваиваем номер выбранного пункта меню.
```

В этой программе командой @ X,Y SAY "Текст заголовка" выводится заголовок в месте экрана с координатами X – номер столбца, Y – номер строки экрана.

Команда @ X,Y PROMPT "Текст строки меню" MESSAGE "Сообщение" выводит пункт меню в строке Y и столбце X, а также сообщение в месте, заданном командой SET MESSAGE TO.

По созданному меню можно перемещаться курсорными клавишами. Выбор пункта осуществляется при нажатии Enter или SpaceBar.

В зависимости от выбранного пункта можно запрограммировать выполнение той или иной функции.

ПРИМЕР *создания вертикального меню (FOX-POPUP-меню):*

```
 DIMENSION a(4)          && задание массива пунктов меню
a(1)="Первая функция"
a(2)="Вторая функция"
a(3)="Третья функция"          && a(1)...a(4) - пункты меню
a(4)="Четвертая функция"
n=3            && номер активного пункта при открытии меню
* Описание меню:
@ 8,30 MENU a, 4 TITLE "Заголовок меню"
```

* 8 – номер строки, 30 – номер столбца, *a* – название массива пунктов меню,

* 4 – число пунктов меню,

* TITLE - заголовок над пунктами

READ MENU TO n && активизация меню, n – начальная позиция выбора, при выборе пункта меню этот выбор номера пункта запоминается в переменной n.

Для организации повторного обращения к пунктам меню достаточно поставить в цикл (do while .t.) только команды активизации меню (MENU TO…; READ MENU TO…).

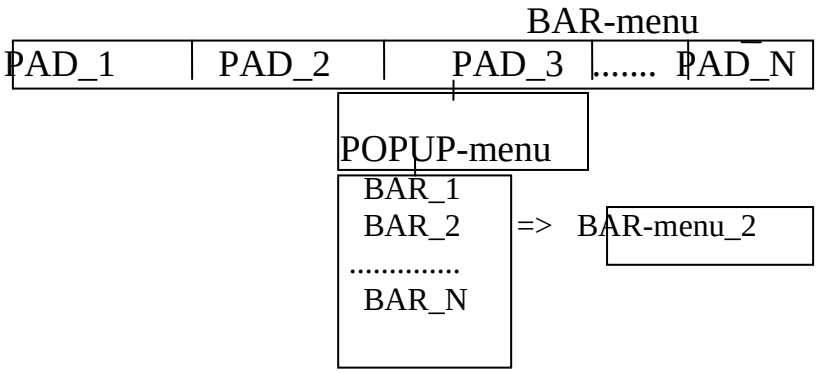
Организация dBase – меню

Общая структура dBase-меню: именованная линейка меню (BAR-меню), состоящая из заголовков (PAD-пунктов). Каждому заголовку может быть назначено POPUP-меню – именованное вертикальное меню, состоящее из BAR-строк (не путать с BAR-меню!).

BAR-меню и POPUP-меню можно использовать и отдельно. BAR-строке можно назначить выбор других BAR-меню или POPUP-меню, что позволяет создавать многоуровневые меню.

Основные операции при работе с dBase-меню:

- описание (DEFINE),
- определение реакций выбора (ON SELECTION, ON),
- активация (ACTIVATE),
- деактивация (DEACTIVATE) или удаление (CLEAR, RELEASE).



Универсальные опции при описании меню:

IN WINDOW "имя окна" / IN SCREEN – вывести в окно или на экран,

KEY "метка клавиши" – клавиша активизации,

MARK <выр.С1>] – поместить перед каждой опцией символьную метку (например, MARK CHR(254)),

MESSAGE "сообщение",

COLOR "список цв.пар"/ COLOR SCHEME "номер"

ПРИМЕР *описания и активации горизонтального меню:*

```
DEFINE MENU _main BAR AT LINE 1 KEY ALT+Z
DEFINE PAD _p1  OF _main  PROMPT 'пункт_1'
```

```
DEFINE PAD _p2  OF _main  PROMPT 'пункт_2'
DEFINE PAD _p3  OF _main  PROMPT 'пункт_3'
DEFINE PAD  quit OF _main  PROMPT '\<Выход'
ON SELECTION MENU _main  DO  proc_1  WITH  MENU(), PAD()
ACTIVATE MENU _main  NOWAIT
PROCEDURE proc_1
PARAMETER mmenu, mpad ..... && текст процедуры
IF mpad = 'QUIT'
      DEACTIVATE MENU _main
      RELEASE MENU _main
ENDIF
RETURN
```

ПРИМЕР *описания и активации вертикального меню*:

```
ON PAD _p1 OF _main ACTIVATE POPUP pop_1
DEFINE POPUP pop_1 FROM 4,1
DEFINE BAR 1 OF pop_1 PROMPT '\<1-я строка'
DEFINE BAR 2 OF pop_1 PROMPT '\<2-я строка'
DEFINE BAR 3 OF pop_1 PROMPT '\<3-я строка'
ON SELECTION POPUP pop_1 DO proc_1  WITH POPUP(), PROMPT()
ON SELECTION BAR 1 OF pop_1 DO proc_11
ON BAR 3 OF pop_1 ACTIVATE POPUP pop_21 *&& вызов меню 2-го уровня
```

Для организации повторного обращения к пунктам меню достаточно поставить в цикл (do while .t.) команду активизации меню ACTIVATE MENU _main.

Организация клавишных меню

К простейшим меню относят клавишные, представляющие собой просто набор клавишных назначений без какого-либо явного их отображения на экране. Такое отображение обычно организуют дополнительно в виде текстовой строки подсказки. Формат клавишных назначений:

ON KEY LABEL "метка клавиши" "команда"

Метка клавиши – это символ или цифра самой клавиши (или комбинации клавиш) или имя, присвоенное клавише. Например: LeftArrow, Backspace, F1, Ctrl-F1, Alt-0, Ctrl-RightArrow.

ПРИМЕР:

```
ON KEY LABEL   Alt-F1  DO _help
ON KEY LABEL   Alt-F2  DO _menu1
ON KEY LABEL   Alt-F3      Browse
ON KEY LABEL   Alt-F4  ?  'Нет операции'
```

Другие возможности работы с базами данных

Создание экрана.

Система Visual FoxPro используется для просмотра данных различными

способами. Можно просмотреть таблицу данных в окне просмотра, можно просмотреть поля каждой записи, а также с помощью "создания экрана" Вы можете создать свой метод просмотра/изменения данных, помещая данные в нужные места экрана или страницы. Возможности FoxPro в этом плане просто уникальны.

Создание меню.

Вы можете за считанные минуты создать свою систему меню, со всеми необходимыми командами для работы с вашими базами данных, а не пользоваться стандартной системой меню FoxPro.

Создание проекта.

Все элементы: экраны, отчеты, запросы, меню... можно объединить в единое целое, называемое проектом.

Проект служит для управления элементами в процессе создания приложения. Создав проект и задав файлы, которые должны в него войти, можно использовать этот проект для сборки приложения (.APP) или исполняемого файла (если у Вас есть пакет FoxPro Distribution Kit).

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Целью лабораторного практикума является усвоение и закрепление теоретического материала, приобретение практических навыков работы с СУБД при выполнении основных элементарных операций с базами данных, знакомство с инструментальными системами FoxPro, Visual FoxPro, Access. Приведены восемь лабораторных работ, охватывающие основные разделы курса. Каждое описание содержит перечень необходимых теоретических вопросов, индивидуальные задания по вариантам и порядок выполнения работы.

В предлагаемых заданиях структура данных соответствует оперативной информации по заданной теме, или информации, изменяющейся после определенного периода расчетов. В оперативную информацию не входят поля, имеющие большую длину, такие данные хранятся в базе для справочной информации. Связь оперативной и справочной информации обычно реализуется через коды или ключевые поля. В заданиях, где необходимо создать несколько таблиц баз данных, создаются справочники расшифровки кодов, содержащиеся в оперативных данных.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Создание проекта, базы данных и таблиц

Цель работы: *знакомство с интерфейсом СУБД, умение создавать проект, базу данных, таблицы, вводить, удалять и редактировать данные в таблице*

Задание

Создать новый проект, базу данных, таблицу в соответствии со структурой в выбранном варианте задания. Заполнить таблицу, вывести данные на экран, выполнить редактирование структуры и содержимого таблицы базы данных. Выполнить задание в режиме конструктора.

Пример

1. Создание БД. В БД допускаются следующие основные типы полей: **символьный** (C) – в поле может содержаться строка; **числовой** (N) – поле хранит целые или вещественные числа; **дата** (D) – хранится величина типа «дата». Длина такого поля всегда =8; **логические** (L) – хранится величина типа «истина-ложь». Истина вводится как Y,y,T или t, ложь как N,n,F или f. Длина такого поля всегда =1;

Команды:

create имя Создание БД, т.е. файла типа *.dbf. При создании необходимо ввести структуру БД – т.е. информацию об именах, типах и длинах полей базы. При создании числового поля для него также можно указать число позиций для вывода дробной части числа (по умолчанию 0)

use имя Позволяет открыть БД с указанным именем для работы с ней;

modify structure Позволяет изменить структуру открытой БД, если в ней обнаружены неточности;

list structure Позволяет просмотреть структуру открытой БД

edit Позволяет вызвать окно редактирования открытой БД

list Позволяет просмотреть открытую БД

Задание: Создайте БД с именем, соответствующим номеру Вашей группы, например «352.dbf», следующей структуры:

Имя	Тип	Ширина	Точность
FIO	C	20	
DR	D	8	
POL	L	1	
GRUP	C	3	
MAT	N	1	0
FIZ	N	1	0
TERMEX	N	1	0
CTRMEX	N	1	0

Комментарии: FIO – фамилия и инициалы, например, в виде Иванов А.А.; DR – дата рождения в формате ДД.ММ.ГГ. Чтобы дата правильно вводилась в этом формате, нужно предварительно отдать команду set date german; POL – пол студента, GRUP – номер группы. Поля MAT, FIZ, TERMEX, CTRMEX содержат, соответственно, оценки за последний семестр по математике, физике, теоретической механике и строительной механике.

Введите информацию о студентах своей группы.

Просмотрите структуру базы и исправьте тип поля POL на символьный.

Просмотрите измененную базу данных.

Порядок выполнения работы по варианту

1. Создать базу данных по варианту (file->New->Database->New).
2. Создать таблицу в базе данных (New table).
3. При создании структуры таблицы задать наименование поля, маску ввода, формат вывода, правило ввода (Rule), значение по умолчанию (Default Value).
4. Создать индексный файл (вкладка Index)
5. Заполнить таблицу (5 записей).
6. Посмотреть содержимое таблиц (команда Browse).
7. Добавить записи в таблицу базы данных с помощью команды Append blank (3 записи).
8. Вывести содержимое таблиц на экран с распечатанным полем Мемо (команда List).
9. Изменить структуру таблицы базы данных (Добавьте поле-ADDI(числовое)) (Modify structure). Проверьте реорганизацию таблицы базы.
10. В одну из записей поля примечаний скопировать (из окна Command) текст выполненных команд (или составить протокол работы), снабдив его соответствующими комментариями.

Контрольные вопросы:

1. Что такое база данных?
2. Что такое СУБД? Классы СУБД по способам организации информационного фонда.
3. Два режима работы в Visual FoxPro.
4. Структура таблиц базы данных в FoxPro. Запись. Типы полей.
5. Основные объекты баз данных в Visual FoxPro.
6. Команды модификации структуры таблицы базы данных.
7. Команды просмотра.
8. Команды добавления новых записей.
9. Команды заполнения и редактирования таблиц.
10. Особенности работы с полем Мемо.
11. Какие операции выполняют следующие команды:

CREATE,

MODIFY STRUCTURE,

SET STATUS ON,
SET TALK OFF,
CLOSE DATABASE,
APPEND [BLANK],
BROWSE

Подготовить отчет о выполнении лабораторной работы (№, тема, цель работы, порядок выполнения работы (указать имена базы данных, таблицы, структуру таблице, содержание таблиц), ответить на контрольные вопросы).

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: **Добавление, удаление, редактирование и упорядочения данных в таблице**

Цель работы: *Научиться основным операциям ведения базы, добавления записей из другой таблицы, редактирования, удаления и выбора нужной информации*

Пример

При выполнении задания записывайте в отчет условие каждого пункта и команды, необходимые для его выполнения.

1. Откройте БД stud.dbf в первой рабочей области. Во второй рабочей области создайте БД stud2.dbf со следующей структурой:

Имя поля	**Тип**	**Длина и точность**	**Назначение**
Fio	C	20	Фамилия И.О.
Grup	C	3	Номер группы
Ball	N	4.1	Средний балл

Командой **append from stud fields fio,grup** скопируйте информацию из базы stud (поля fio и grup). Просмотрите получившуюся базу stud2.

2. Просмотрите записи с номерами а) 5; б) 1; в) последнюю запись;

3. С помощью команды **Browse** просмотрите следующие записи:

а) список студентов мужского пола в вашей группе;

б) список студентов в вашей группе, имеющих средний балл не ниже 4;

в) список студентов, имеющих средний балл от 3.5 до 4.5 включительно;

г) список студентов, имеющих «5» по математике или физике или по этим двум предметам.

д) выведите только фамилии и номера групп студентов, имеющих все пятерки.

е) выведите фамилии, номера групп и даты рождения студентов, имеющих только одну тройку.

Помните, что при вводе выражений можно использовать логические операции .AND., .OR., а при использовании операций сравнения нужно соблюдать тип выражения – строки заключать в ’одинарные кавычки’, логические значения записывать как .T. или .F., даты – в зависимости от установленного в си-

стеме формата. При создании длинных выражений используйте буфер обмена для копирования похожих блоков команд.

4. В базе stud2 пометьте для удаления следующие записи: а) записи о студентах своей группы; б) две последние записи. Добавьте по одной записи в начало и конец базы. Просмотрите таблицу с помеченными на удаление записями. Снимите все пометки удаления.

Команды СУБД, использованные в лабораторной работе:

create имя_файла_БД	Создание базы данных
append from имя_файла_БД	Добавление записей из другой базы
copy to имя_файла_БД	Копирование записей в другую базу
go номер записи	Переход к записи по номеру
go top	Переход к первой записи
go bottom	Переход к последней записи
list record номер записи	Вывод записи с нужным номером
list next количество_записей	Вывод определенного числа записей
list for условие	Просмотр записей по условию
delete	Пометка записей для удаления
insert	Вставка записи
recall	Снятие пометок удаления
pack	Удалить запись физически

Задание

Создать таблицу 2 из нескольких полей таблицы 1 по варианту задания. Заполнить таблицу 2 из таблицы 1, вывести данные на экран по условию, создать индексный файл. Выполнить сортировку данных.

Порядок выполнения работы по варианту

1. Создать таблицу 2 в базе данных по варианту.
2. Скопировать данные из таблицы 1 в таблицу 2.
3. Создать отсортированный файл. Посмотреть содержимое таблиц (команда Browse).
4. Создать индексный файл (вкладка Index).
5. Добавить записи в таблицу 2 базы данных с помощью команды (3 записи).
6. Отредактировать содержимое поле Мемо (команда Browse).
7. По условию вывести записи из таблицы 1 за январь месяц с организацией расчетного поля (команда Browse).

Контрольные вопросы:

1. Перечислите команды навигации по таблице.
2. Перечислите команды открытия таблиц в рабочей области.
3. Какова общая структура команд в Visual FoxPro.

4. Какие параметры в команде определяют границы действия в таблице?

5. Перечислите команды, которые работают с текущей записью.

6. Перечислите полноэкранные команды.

7. Как пометить записи на удаление?

8. Команды восстановления записи.

9. Можно ли восстановить удаленную запись после выполнения команды pack?

10. Какие операции выполняют следующие команды:

ИМЯ_ФАЙЛА_БД>APPEND FROM <ИМЯ_ФАЙЛА_БД>

COPY TO <ИМЯ_ФАЙЛА _БД>

SORT ON СПИСОК ПОЛЕЙ TO <ИМЯ_ФАЙЛА _БД>

INDEX ON <Ключевые_поля> TO <ИМЯ_ФАЙЛА_ИДЕКСНОГО>

BROWSE

Подготовить отчет о выполнении лабораторной работы (№, тема, цель работы, порядок выполнения работы (указать имя базы данных, отсортированной таблицы и структуру таблицы 2, содержание таблиц), ответить на контрольные вопросы).

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Создание проекта, работа с командными файлами. Индексирование, сортировка данных

Цель работы: Научиться создавать проекты, приобрести навыки работы с командным файлом, освоить операции индексирования, сортировки

Задание 1

Создать проект, в котором реализован алгоритм решения квадратных уравнений. Создать таблицу, содержащею коэффициенты квадратного уравнения (a,b,c,d), дискриминант, два решения (x1 и x2) и результат («нет решения», «одно решения», «два решения»).

Порядок выполнения работы

1. Создать проект (File->New->Project->New file

2. Создать базу данных – файл с расширением *.dbc. (Вкладка проекта: Data)

3. Создать таблицу со структурой (a, b,c,d, x1, x2 N-9(5).9(2) – числовые значение, rezl – C(10) , строковая переменная. (Вкладка проекта: Data)

4. Создать командный файл prog1. prg. (Вкладка проекта: Code) . Кнопка New.Набрать следующий код программы для решения квадратного уравнения:

```
CLOSE DATABASES
  clear
  select tab_kv
  @ 1,1 say "Решение квадратного уравнения"
```

```
WAIT  "Добавить новую запись Y/N" TO key
IF key="Y"
 APPEND BLANK
ENDIF
BROWSE
scan
 replace d WITH b*b-4*a*c
 IF d<-0.0000001
   replace rezl WITH " нет корней",x1 WITH VAL(' '), x2 WITH VAL(' ')
 ELSE
    IF d>0.00000001
      replace rezl WITH два корня", x1 WITH (-b-SQRT(d))/(2*a), x2 WITH (-
b+SQRT(d))/(2*a)
     ELSE
      x11=(-b+SQRT(d))/(2*a)
      replace rezl WITH " один корень", x1 WITH x11,x2 WITH 0
     ENDIF
  ENDIF
? 'a=',a,' b=',b,' c=',c,' d=',d,' ',rezl,' x1=',x1,' x2=',x2
Endscan
```

Сохранить Cnt+W

2. Запустить программу на выполнение, кнопка в окне проекта RUN. На запрос о добавлении записи ответить Y. Ввести значения коэффициентов. Выйти из окна команды Browse. Запустить программу снова и на запрос ответить N.

Задание 2

Добавить в проект БД STUD, выполнить в командном файле, сортировку и индексирования данных

Порядок выполнения работы

1. В проект добавить созданную базу данных в предыдущих лабораторных работах.
2. Создать командный файл prog2. prg. .
3. Используя сортировку БД, выполнить над БД STUD следующие действия:

а) записать в БД stud0 алфавитный список всех студентов, независимо от номера группы (только поле FIO);

б) сделать в БД stud3 записи о студентах 3-го курса, отсортированные по полям, "группа", "дата рождения" и "фамилия";

в) записать в БД stud4 фамилии и поля баллов всех студентов, отсортированные по возрастанию балла по физике и по фамилии, если баллы по физике одинаковы;

г) отсортировать базу stud по убыванию года рождения, по возрастанию номера группы для студентов с одинаковым годом и по алфавиту в рамках одной группы; результат сортировки поместить в БД stud5;

д) отсортировать БД по номеру группы (по возрастанию) и фамилии (по убыванию); результат сортировки поместить в БД stud6. Для сортировки по убыванию использовать мультииндексный файл.

4. Проиндексировать БД STUD по следующим критериям:

а) по номеру группы и дате рождения;

б) по номеру группы, среднему баллу и фамилии;

в) используя индексирование, вывести информацию о студентах Вашей группы в порядке убывания среднего балла.

Основные команды СУБД, использованные в лабораторной работе:

browse for условие	редактирование записей, отвечающих условию
sort to имя_файла on список_полей	сортировка записей в файле БД
index on условие to имя_файла	индексирование файла БД
use имя_базы index имя_файла	открытие индекса при открытии БД
set index to имя_файла	открытие индекса после открытия БД

Контрольные вопросы:

1. Объяснить каждую команду из задания 1, п.4.
2. Команда Replace.
3. Команда Append blank.
4. Команды ввода - вывода.
5. Использование переменных.
6. Типы индексных файлов.
7. Виды циклических команд.
8. Команда сортировки.
9. Команды создания индексных файлов.
10. Команды разветвления (if, case).

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Выбор информации. Создание локальных запросов

Цель работы: *Научиться создавать локальные запросы для выбора данных из таблиц базы данных.*

Задание 1

Создать проект, в котором реализован выбор заданной информации из таблиц. Создать запрос (Local View) для просмотра определенного содержимого таблицы базы данных.

Порядок выполнения работы:

1. Открыть проект с базой данных STUD .

2. С помощью Local View выбрать нужную информацию:

– содержимое всей таблицы;

– запрос с расчетным полем;

– данные удовлетворяющие сложному логическому условию (вкладка Filter);

– данные за текущий месяц (вкладка Filter);

– запрос с группировкой.

3. Посмотреть SQL-код .

4. Скопировать содержимое sql-кода в командный файл с расширением *.prg. Запустить на выполнение (!).

5. Используя фильтрацию записей, вывести следующую информацию:

а) записи о студентах, у которых в поле POL отсутствуют буквы T или F. Используя browse, исправить записи. Закрыть фильтр;

б) записи о студентах, у которых запись в поле FIO не начинается с большой буквы. Внести исправления.

6.С помощью Local View выбрать нужную информацию:

а) студентов-мужчин, родившихся в 1979 или 1980 годах с группировкой по номеру группы;

б) студентов, родившихся в субботу во второй половине 1979 или 1980 года, создать расчетное поле == средний бал по математике;

в) посчитать кол-во студентов, у которых день рождения совпадает с номером месяца;

г) студентов, у которых в фамилии или инициалах есть буквы "В" или "в";

д) студентов, которые родились в нечетное число;

е) студентов, у которых сумма цифр в дне рождения совпадает с суммой цифр в номере месяца;

ж) студентов, у которых в инициалах есть буквы "Б" или "б";

з) студентов, номер группы которых кратен 2 и 4, а сумма цифр номера группы больше 1.

Основные функции СУБД, использованные в лабораторной работе:

Str()
Val()
Year()
Month()
Day()
Date()
substr()
set filter to
ltrim()
rtirm()
ctod()

dtoc()
count for < условие>
locate for < условие>
continue
seek
Average
IIF(, ,)
Контрольные вопросы:

Контрольные вопросы:

1. Как получить нужную информацию из базы данных?
2. Основные вкладки экранных запросов (Local View).
3. Синтаксис команд count, Average.
5. Команды поиска locate for < условие> , Continue, Seek.

FOUND Возвращает .T., если последняя команда успешно выполнена
IIF
INKEY([[] [,]]) Возвращает целое значение, соответствующее последнему нажатию клавиши или одному нажатию мыши
INT() Возвращает целую часть числового выражения
ISALPHA() Возвращает .T., если начинается с буквы
ISLOWER() Возвращает .T., если первый символ в - буква в нижнем регистре
ISUPPER() Возвращает .T., если первый символ в - буква в верхнем регистре
LEFT(,) Возвращает указанное количество символов
LEN()Возвращает длину символьного выражения

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Организация структурированных запросов (SQL) в базе данных

Цель работы: Н*аучиться получать нужную информацию с помощью SQL запросов*

Задание

В командном файле организовать структурированный запрос SQL в базе данных по своему варианту.

Порядок выполнения работы

1. Открыть проект.
2. В командном файле организовать структурированный запрос SQL из таблиц базы данных.

а) выборка всех полей из описанных таблиц;

б) вывод минимального, максимального и среднего значения числового выражения (поле или арифметическое выражение), сгруппировать по первому ключевому полю;

в) выбор всех полей таблицы, у которых значение поля «дата» соответствует текущему месяцу, вывод сделать в порядке возрастания поле «дата»;

г) выбор записей с суммированием числового поля и с подсчетом всех записей, попавших в выборку по каждому значению ключа в файл (DBF или текстовый файл), с последующим просмотром;

д) ввести значения ключевых выражений, выбрать все записи из трех таблиц, которые удовлетворяют заданным значениям ключевого выражения;

е) выбрать минимальное и максимальное значения числового выражения таблицы, которые сгруппированы по первому ключевому полю и удовлетворяют заданному условию.

Контрольные вопросы:

1. Команда SELECT … FROM … WHERE…GROUP BY … HAVING… ORDER BY
2. GROUP BY … HAVING
3. INTO TABLE
4. TO FILE
5. Подзапросы SELECT … FROM … WHERE (SELECT. …GROUP BY …)

Использование подзапросов

Задание: Выполнить выбор записей по диапазону значений для заданной базы данных. Осуществить выборку из нескольких таблиц. Продемонстрировать приобретенные навыки построения многотабличных запросов. Одна таблица (минимально) должна содержать поля: факультет, идентификатор факультета в обязательном порядке, другая (минимально) - курс, идентификатор факультета в обязательном порядке.

Вариант 3: Выполнить запросы:

1) вывод фамилий студентов, обучающихся в 7 и 2 корпусах и в возрасте больше, чем средний возраст студентов 2 и 3 курса факультета ФАМ:

```
SELECT predmet5.фамилия, predmet5.корпус,;
predmet5.курс, faq.name_faq,;
(YEAR(date())-YEAR(predmet5.дата_рожд)) AS Возраст;
FROM predmet5, faq ;
WHERE predmet5.корпус IN (2,7) ;
AND faq.id_faq=predmet5.факультет ;
AND (YEAR(date())-YEAR(predmet5.дата_рожд))>;
(SELECT avg(YEAR(date())-YEAR(predmet5.дата_рожд));
```

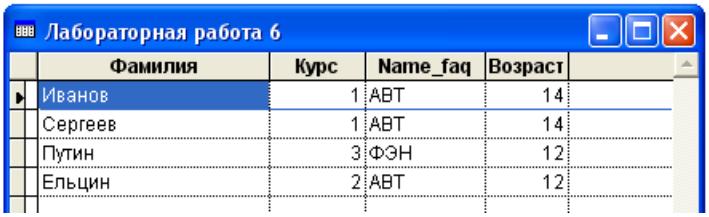
FROM predmet5, faq WHERE faq.name_faq='ФАМ';
AND predmet5.курс IN (2,3) AND faq.id_faq=predmet5.факультет)

Фамилия	Корпус	Курс	Name_faq	Возраст
Иванов	7	1	АВТ	14
Ельцин	2	2	АВТ	12

Рисунок 1 – Результат выполнения запроса

2) вывод фамилий всех студентов, кроме обучающихся на 1 и 2 курсе факультетов ФЭН, РТФ и в возрасте больше, чем средний возраст студентов 2 и 3 курса факультета ФАМ:

SELECT predmet5.фамилия, predmet5.курс, faq.name_faq,;
(YEAR(date())-YEAR(predmet5.дата_рожд)) AS Возраст;
FROM faq, predmet5 ;
WHERE (predmet5.курс NOT IN (1,2) ;
OR faq.name_faq NOT IN ('ФЭН','РТФ')) ;
AND predmet5.факультет = faq.id_faq ;
AND (YEAR(date())-YEAR(predmet5.дата_рожд))>;
(SELECT avg(YEAR(date())-YEAR(predmet5.дата_рожд));
FROM predmet5, faq WHERE faq.name_faq='ФАМ';
AND predmet5.курс IN (2,3) AND faq.id_faq=predmet5.факультет)



Фамилия	Курс	Name_faq	Возраст
Иванов	1	АВТ	14
Сергеев	1	АВТ	14
Путин	3	ФЭН	12
Ельцин	2	АВТ	12

Рисунок 2 – Результат выполнения запроса

ЛАБОРАТОРНАЯ РАБОТА № 6

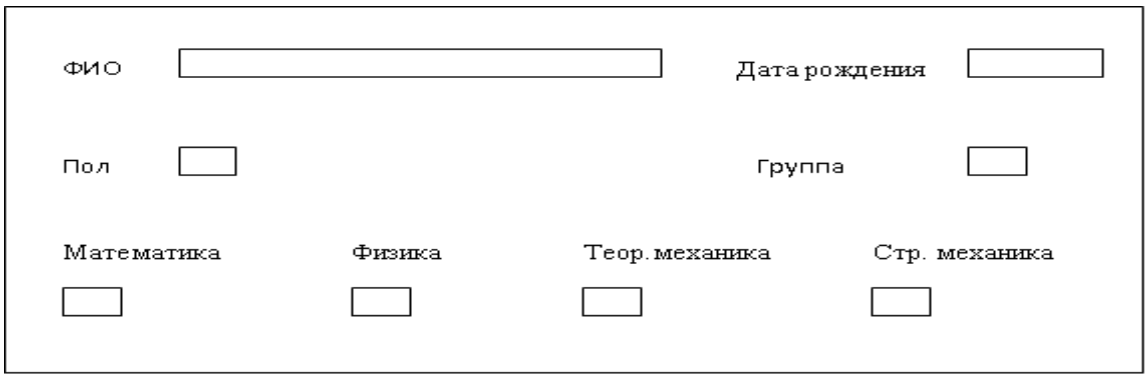
Тема: Создание экранных форм

Цель работы: *получить навыки работы создания экранных форм в режиме конструктора и мастера*

Задание

Создать однотабличную форму с использованием Мастера форм (Form Wizard).

В режиме конструктора создать форму для работы с записями таблицы базы данных с элементами управления, реализовать все операции введения базы данных.



Добавить кнопки навигации по записям таблицы базы данных: «На начала», «Следующая», «Предыдущая», «На конец», «Выход».

Порядок выполнения работы:

1. Открыть базу данных в проекте.
2. В таблице базы данных создать индекс по полю ФИО.
3. Создать форму в режиме Мастера форм (Form Wizard).
4. Создать форму в режиме конструктора форм (Form Designer).
5. В контекстном меню вызвать окно Data Environment, добавить таблицу и проверить свойства таблицы: ControlSource(имя таблицы), Order (имя индексного файла).
6. Перетащить поля таблицы на форму. Добавить кнопки перехода на начало файла, на конец файла, добавление записи, удаление записи, выход.
7. Установить нужные опции для создания формы (размер и расположение формы).
8. Изменить расположение элементов формы.
9. Добавить новое поле или элементы управления (LISTBOX) в форму.

Контрольные вопросы:

1. Режимы создания экранной формы в системе.
2. Шаги создания экранной формы с использованием мастера Form Wizard.
3. Создание экранной формы с использованием конструктора Form Designer.
4. Функциональные возможности экранных форм.
5. Элементы управления формы в среде Visual FoxPro.

ЛАБОРАТОРНАЯ РАБОТА № 7
Тема: Создание отчетов

Цель работы: *знакомство с организацией различных видов отчетов*

Задание

Создать отчет по структуре данных, соответствующий заданию по варианту, с выводом итогов. Организовать группировку данных и вводом итогов по группе. Создать итоговый отчет.

Порядок выполнения работы

1. Открыть проект, можно использовать базу Stud. .
2. Создать отчет в Visual FoxPro с использованием Мастера отчета (Report Wizard).
3. Составить отчет в режиме конструктора

- содержимого таблиц базы данных в виде табуляграммы с заголовком документа и шапкой на каждой страницы.
- С итогом по отчету
- С группировкой с заголовком и итогом по группе
- Создать итоговый отчет
- Создать отчет в два столбца
- Создать свободный отчет

4. Результат итогового отчета записать в файл с расширением **.txt**.
5. Использовать Конструктора отчета (Report Designer) для настройки и корректировки отчета.

Контрольные вопросы:

1. Структура документа.
2. Назначение полос на планшете отчета.
3. Кнопки панели Repot Control в системе Visual FoxPro.
4. Предварительный просмотр отчета.
5. Запуск на выполнение отчета.
6. Запись отчета в текстовый файл.

ЛАБОРАТОРНАЯ РАБОТА № 8
Тема: Организация меню в приложении

Цель работы: *получить практический опыт создания меню работы для приложения в командном файле*

Задание

Создать командный файл, в котором организовано меню введения базы данных. Создание меню-приложения

Порядок выполнения работы

1. Открыть проект, можно использовать базу Stud.

Примечание: Можно установить свой путь к пользовательским файлам (Tools->Options->File Locations->Default directory)

2. Создать меню с помощью конструктора MenuDesigner.

Горизонтальное MENU-BAR может содержать следующие пункты:

Справочники, Учет, Отчеты, Сервис, Помощь, Выход

a. ***Справочники:*** (submenu) – Формы для одиночных таблиц (например, список товаров, список клиентов)

Примечание: запуск формы на выполнение с помощью команды:

DO FORM <имя формы>

*b.**Учет:*** (submenu) – Многотабличная форма, Запросы (procedure или команды)

c. ***Отчеты:*** (submenu) – Все виды отчетов, в том числе многотабличные

Примечание: запуск отчета на выполнение с помощью команды:

REPORT FORM <имя отчета> PREVIEW

d. ***Сервис:*** (submenu или procedure).– Просмотр каталога, копирование, сохранение, экспорт данных

Примечание*: создать подменю в командном файле

e.. ***Помощь:*** (команда или процедура) вызов текста подсказки

j. ***Выход:***(команда, процедура)

3. В командном файле создать fox-меню из 3 пунктов, выбор которых выведет содержимое базы данных в окне Browse с фильтрацией записей.

4. В командном файле создать dbase-menu клавишное меню из 4 пунктов с использованием команд SQL Клавишные назначения отобразить на экране дисплея.

5. Создать клавишное меню (пометить запись к удалению и восстановить удаленные записи).

Контрольные вопросы:

1. Конструктор меню в системеVisual FoxPro.
2. Создание Quick Menu в системеVisual FoxPro.
3. Организация fox-menu, dbase-menu, клавишного меню в командном файле.
4. Команды клавишного меню.

ЛАБОРАТОРНАЯ РАБОТА № 9
Тема: Создание многотабличных форм

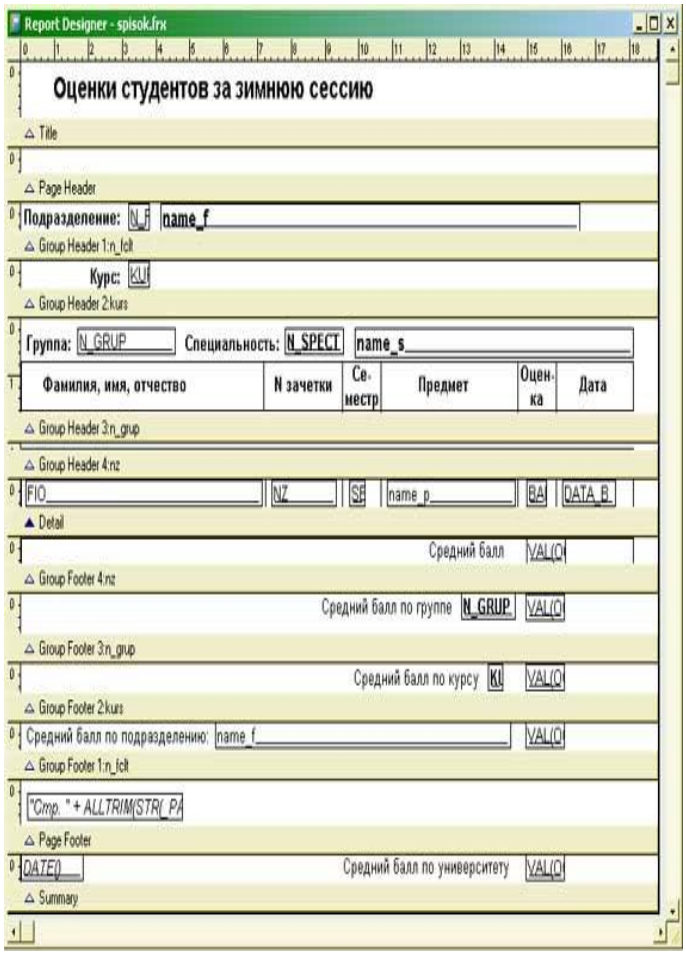
Цель работы: *получить навыки работы создания экранной формы с двумя таблицами*

Задание

Создать форму с несколькими таблицами базы данных с элементами управления, с выводом итогов.

Порядок выполнения работы

1. Открыть проект, можно использовать базу Stud.
2. Примечание: Можно установить свой путь к пользовательским файлам (Tools В режиме мастера создать многотабличную форму.
3. В режиме Конструктора форм добавить экранные средства управления.
4. Добавить элементы управления, например ListBox.
5. Добавить элементы управления, например ListBox



Контрольные вопросы:

1. Как связать таблицы?
2. Как создать многотабличную форму в системеVisual FoxPro?
3. Элементы управления в формах.

ЛАБОРАТОРНАЯ РАБОТА № 10
Тема: Организация отчетов в приложении

Цель работы: *навыки создания различных видов отчетов, содержащих данные из двух таблиц*

Задание

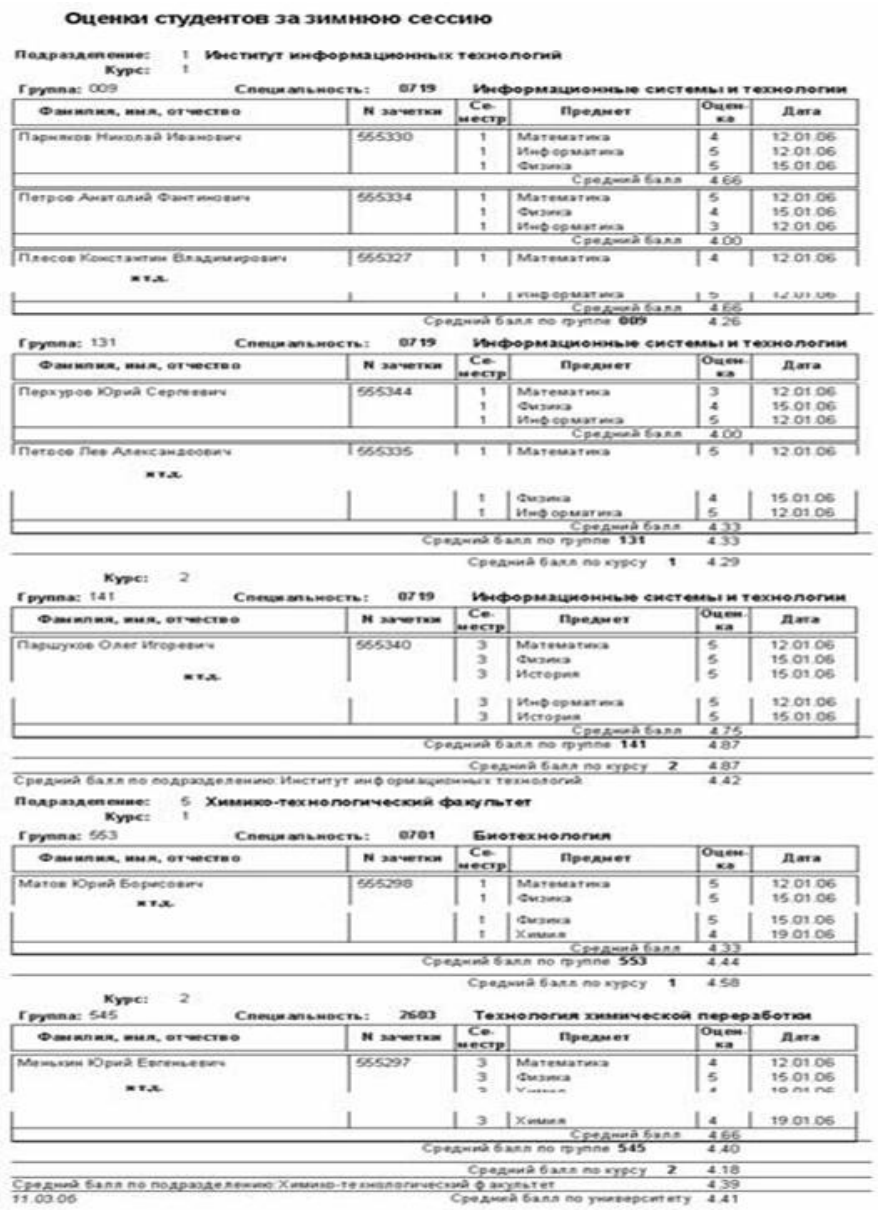
Создать различные виды многотабличных отчетов с группировками, с итогами.

Порядок выполнения работы

1.Открыть проект, можно использовать базу Stud.

2. Составить отчет, содержащий двух таблиц базы данных в виде табуляграммы, использовать группировку данных и расчет итоговых сумм по числовым реквизитам.

3. Составить итоговый отчет.



Оценки студентов за зимнюю сессию

Подразделение: 1 Институт информационных технологий
Курс: 1

Группа: 009 Специальность: 0719 Информационные системы и технологии

Фамилия, имя, отчество	N зачетки	Семестр	Предмет	Оценка	Дата
Парняков Николай Иванович	555330	1	Математика	4	12.01.06
		1	Информатика	5	12.01.06
		1	Физика	5	15.01.06
			Средний балл	4.66	
Петров Анатолий Фантинович	555334	1	Математика	5	12.01.06
		1	Физика	4	15.01.06
		1	Информатика	3	12.01.06
			Средний балл	4.00	
Плесов Константин Владимирович	555327	1	Математика	4	12.01.06
и т.д.					
			Средний балл	4.66	
			Средний балл по группе 009	4.26	

Группа: 131 Специальность: 0719 Информационные системы и технологии

Фамилия, имя, отчество	N зачетки	Семестр	Предмет	Оценка	Дата
Перхуров Юрий Сергеевич	555344	1	Математика	3	12.01.06
		1	Физика	4	15.01.06
		1	Информатика	5	12.01.06
			Средний балл	4.00	
Петров Лев Александрович	555335	1	Математика	5	12.01.06
и т.д.					
		1	Физика	4	15.01.06
		1	Информатика	5	12.01.06
			Средний балл	4.33	
			Средний балл по группе 131	4.33	
			Средний балл по курсу 1	4.29	

Курс: 2

Группа: 141 Специальность: 0719 Информационные системы и технологии

Фамилия, имя, отчество	N зачетки	Семестр	Предмет	Оценка	Дата
Паршуков Олег Игоревич	555340	3	Математика	5	12.01.06
		3	Физика	5	15.01.06
и т.д.		3	История	5	15.01.06
		3	Информатика	5	12.01.06
		3	История	5	15.01.06
			Средний балл	4.75	
			Средний балл по группе 141	4.87	
			Средний балл по курсу 2	4.87	
Средний балл по подразделению Институт информационных технологий				4.42	

Подразделение: 5 Химико-технологический факультет
Курс: 1

Группа: 553 Специальность: 0701 Биотехнология

Фамилия, имя, отчество	N зачетки	Семестр	Предмет	Оценка	Дата
Матов Юрий Борисович	555298	1	Математика	5	12.01.06
и т.д.		1	Физика	5	15.01.06
		1	Физика	5	15.01.06
		1	Химия	4	19.01.06
			Средний балл	4.33	
			Средний балл по группе 553	4.44	
			Средний балл по курсу 1	4.58	

Курс: 2

Группа: 545 Специальность: 2603 Технология химической переработки

Фамилия, имя, отчество	N зачетки	Семестр	Предмет	Оценка	Дата
Меньшин Юрий Евгеньевич	555297	3	Математика	4	12.01.06
и т.д.		3	Физика	5	15.01.06
		3	Химия	4	19.01.06
			Средний балл	4.66	
			Средний балл по группе 545	4.40	
			Средний балл по курсу 2	4.18	
Средний балл по подразделению Химико-технологический факультет				4.39	
11.03.06			Средний балл по университету	4.41	

Контрольные вопросы:

1. Как создать Конструктор отчетов в системеVisual FoxPro?
2. Как правильно группировать данные?
3. Как создать многотабличную форму?
4. Какие виды отчетов вы знаете?

ЛАБОРАТОРНАЯ РАБОТА № 11
Тема: Проектирование баз данных

Цель работы: *получить практический опыт проектирования баз данных*

Задание

Создать инфологическую и даталогическую модели базы данных по заданному варианту. Создать формы для ввода данных. Сформировать запросы для получения информации из базы данных.

Порядок выполнения работы

1. Описать инфологическую модель.
2. Описать даталогическую модель.
3. Составить схему базы данных из нескольких таблиц.
4. Создать проект.
5. Создать таблицы базы данных по схеме п.3.
6. Организовать постоянную связь между таблицами третьей нормальной формы.
7. Создать формы для ввода данных в таблицы, для создания многотабличной формы использовать

   a. Create Form Set .
   b. Wizart Onev to Many Form
   c. В Formy one-to one добавить элемент «кнопка Поиска»
   d. Для поля типа Мемо использовать элемент Editbox
   e. Добавить элементы ListBox или Combobox

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF nkeycode=13
DO case
CASE EMPTY(this.Value)=.f.
APPEND BLANK
replace nomfak WITH ALLTRIM(this.Value)
this.Value=""
thisform.Refresh()
ENDCASE
ENDIF
```

Задание для следующей лабораторной работы

8. Составить запросы пяти различных видов
9. Составить отчеты: с заголовком и итогом, с группировкой, итоговый и свободный отчеты.
10. Создать меню
11. Создать проект.

ЛАБОРАТОРНАЯ РАБОТА № 12

Тема: Формирование многотабличных отчетов

Цель работы: *навыки создания различных видов отчетов, содержащих данные из двух таблиц*

Задание

Создать различные виды многотабличных отчетов с группировками, с итогами.

Порядок выполнения работы:

1. Составить отчет, содержащий двух таблиц базы данных в виде табуляграммы, использовать группировку данных и расчет итоговых сумм по числовым реквизитам.
2. Составить итоговый отчет.

Контрольные вопросы:

1. Как правильно группировать данные?
2. Как создать многотабличную форму?
3. Какие виды отчетов вы знаете?

ЛАБОРАТОРНАЯ РАБОТА № 13
Тема: Создание меню-приложения

Цель работы: *научиться создавать меню работы для создания приложения в режиме конструктора*

Задание

Организовать меню-приложения в режиме конструктора (Menu Designer). Экспортировать данные в текстовый файл, затем импортировать в систему Visual FoxPro.

Порядок выполнения работы

1. Запустить проект.
2. Создать меню с помощью конструктора MenuDesigner.

Контрольные вопросы:

1. Конструктор меню в системеVisual FoxPro.
2. Создание Quick Menu в системеVisual FoxPro.

ЛАБОРАТОРНАЯ РАБОТА № 14
Тема: Создание проекта и выполняемого файла

Цель работы: *создание проекта «под ключ»*

Задание

Разработать приложение: объединить все элементы приложения для их компиляции и создания выполняемого файла с расширением .exe или app.

Порядок выполнения работы

1. Запустить проект.
2. Включить таблицы, формы, отчеты, меню, используемые в этом проекте.

3. Set Main опции PROECT главной строки меню установить в качестве главной программы проекта – меню (пользовательское).

ТЕМЫ КУРСОВЫХ РАБОТ ПО ВАРИАНТАМ

1 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Телефонные переговоры**" для хранения необходимой информации.

Создайте таблицы и связи между ними. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество абонента.
2. Телефонный номер абонента.
3. Домашний адрес абонента. (Мемо)
4. Телефонный код и название города, куда звонил абонент.
5. Тариф за 1 минуту разговора с указанным городом.
6. Дата разговора, время разговора.
7. Продолжительность разговора.
8. Телефон, по которому звонил абонент

---

2 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Библиотека**" для хранения необходимой информации.

База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество читателя.
2. Домашний адрес читателя. (Мемо)
3. Телефон читателя.
4. Дата рождения читателя.
5. Номер читательского билета.
6. Автор книги, которую взял читатель.
7. Название книги, жанр, год издания и издательство.
8. Цена книги.
9. Дата выдачи книги.
10. Дата возврата книги плановая и фактическая.

---

3 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Резервирование билетов**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество пассажира.
2. Домашний адрес пассажира. (Мемо)
3. Телефон пассажира
4. Номер поезда и название маршрута.
5. Номер вагона.
6. Дата отправления.
7. Время отправления/прибытия.
8. Пункт назначения.
9. Стоимость проезда.

---

4 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Фотоуслуги**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество клиента.
2. Домашний адрес клиента. (Мемо).
3. Телефон клиента.
4. Наименование фотоуслуги.
5. Описание фотоуслуги (характеристика)
6. Количество единиц заказа.
7. Цена за единицу.
8. Дата приемки заказа.
9. Дата выдачи заказа.

---

5 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Коммунальные услуги**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество квартиросъёмщика.
2. Домашний адрес квартиросъёмщика. (Мемо)
3. Номер лицевого счета.
4. Количество жильцов.
5. Площадь квартиры, кв.м.
6. Вид услуги (название платежа).
7. Стоимость услуги на квадратный метр площади.
8. Стоимость услуги на 1 жильца.
9. Дата оплаты.
10. Скидка, % (льготы при оплате).

6 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Прокат товаров**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество клиента.
2. Серия и номер паспорта клиента.
3. Домашний адрес клиента. (Мемо)
4. Телефон клиента.
5. Наименование товара.
6. Описание товара.
7. Стоимость товара.
8. Дата выдачи.
9. Дата возврата плановая и фактическая.
10. Стоимость проката за сутки.

---

7 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Кино**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Название фильма.
2. Жанр фильма.
3. Страна-производитель фильма.
4. Название кинотеатра.
5. Адрес кинотеатра. (Мемо)
6. Время показа.
7. Цена билета.

---

8 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Классный журнал**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество ученика.
2. Домашний адрес ученика. (Мемо)
3. Домашний телефон ученика.
4. Наименование предмета.
5. ФИО учителя, ведущего предмет.
6. Дата получения оценки.
7. Вид работы (контрольная, самостоятельная, ответ у доски, тест).

8. Оценка.

---

9 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Шахматный турнир**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия и имя шахматиста.
2. Клуб, от которого выступает шахматист.
3. Тренер шахматиста.
4. Дата проведения игры.
5. Фамилия и имя соперника.
6. Клуб, от которого выступает соперник.
7. Тренер соперника.
8. Итог игры.
9. Доп. информация о соревновании. (Мемо)

---

10 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Инвесторы**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия и имя отчества инвестора.
2. Банк. реквизиты инвестора.
3. Сумма.
4. Дата вложений.
5. Телефон.
6. Цель инвестиции. (Мемо)
7. Банковские реквизиты предприятия.

---

11 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Зарплата**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия и имя отчества.
2. Банк. реквизиты работника.
3. Оклад.
4. Дата выплаты зарплаты.
5. Телефон.
6. Должность.

7. Расчетный месяц.
8. Домашний адрес. (Мемо)

---

12 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Книги**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество автора.
2. Название книги.
3. Кол-во страниц.
4. Дата издания.
5. Издательство.
6. Количество экземпляров.
7. Цена книги.
8. Адрес издательства (Мемо).

---

13 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Врачи**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество врача.
2. Номер кабинета.
3. Специальность.
4. Дни приема пациентов.
5. Время приема.
6. Домашний адрес врача (Мемо).

---

14 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Поликлиника**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество пациента.
2. Адрес.
3. Телефон.
4. Дата рождения.
5. Шифр заболевания.
6. Дата открытия больничного.
7. Количество дней на больничном.
8. Доп. информация о заболевании. (Мемо).

15 вариант. Проектирование базы данных

16. Разработайте структуру базы данных "**Школа**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество учителя.
2. Предмет.
3. Номер кабинета.
4. Количество уроков в учебном году.
5. Вид контроля (опрос, контрольная работа).
6. Количество интерактивных занятий.
7. Дата рождения.
8. Доп. информация об учителе. (Мемо)

---

16 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Интернет**" для хранения необходимой информации.

Создайте таблицы и связи между ними. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество абонента.
2. Телефонный номер абонента.
3. Домашний адрес абонента.
4. Телефонный код.
5. Тариф за скорость соединения.
6. Дата оплаты.
7. Скорость соединения.

---

17 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Каталог в Библиотеке**" для хранения необходимой информации.

База данных должна хранить следующую информацию:

1. Фамилия, Имя, Отчество читателя.
2. Жанр.
3. Автор книги.
4. Название книги, год издания и издательство.
5. Тип каталога (алфавитный, предметный).
6. Тип издания (справочник, сборник, многотомник и т.д.).
7. Количество.

8. Дата издания книги.
9. О содержании книги (Мемо).

---

18 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Продажа билетов**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество пассажира.
2. Номер поезда и название маршрута.
3. Номер вагона.
4. Дата отправления.
5. Время отправления/прибытия.
6. Пункт назначения.
7. Стоимость проезда.
8. Доп. информация о поезде. (Мемо)

---

19 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Видеопрокат**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество клиента.
2. Домашний адрес клиента. (Мемо)
3. Телефон клиента.
4. Наименование видеофильма.
5. Количество единиц заказа.
6. Цена за единицу.
7. Дата приемки заказа.
8. Дата выдачи заказа.

---

20 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Расчет потребления электроэнергии**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество квартиросъёмщика.
2. Домашний адрес квартиросъёмщика. (Мемо)
3. Номер лицевого счета.
4. Показания предыдущие.
5. Показания текущие.

6. Тариф.
7. Дата оплаты.
8. Скидка, % (льготы при оплате).

---

21 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Лыжная база**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество клиента.
2. Серия и номер паспорта клиента. (Мемо)
3. Домашний адрес клиента.
4. Телефон клиента.
5. Описание товара.
6. Стоимость за час.
7. Дата выдачи.
8. Время выдачи.
9. Время возврата.
10. Стоимость.

---

22 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Киностудия**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Название фильма.
2. Жанр фильма.
3. Страна-производитель фильма.
4. Название киностудии.
5. Режиссёр.
6. Звукооператор.
7. Стоимость.
8. Дата выпуска фильма.
9. О содержании книги (Мемо).

---

23 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Банковские расчеты**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество клиента.

2. Банк. реквизиты.
3. Сумма дебета.
4. Дата вложений.
5. Сумма кредита.
6. Цель платежа. (Мемо)
7. Дата операции.

---

24 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Расчет премии**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество.
2. Оклад.
3. Дата выплаты.
4. Процент премии.
5. Должность.
6. Расчетный месяц.
7. Основание выплаты (Мемо).

---

25 вариант. Проектирование базы данных

Разработайте структуру базы данных "**Книжный магазин**" для хранения необходимой информации. База данных должна хранить следующую информацию:

1. Фамилия, имя, отчество автора.
2. Название книги.
3. Дата издания
4. Издательство.
5. Количество экземпляров в упаковке.
6. Цена книги оптовая.
7. Цена книги розничная.
8. Описание книги. (Мемо).

ВАРИАНТЫ ЗАДАНИЙ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Задание №1

1. Создать базу данных (записать команду и имя файла с расширением).
2. Создать таблицу БД (записать команду и имя файла с расширением).
3. Составить структуру таблицы: «Лицевой счет», «ФИО клиента», «Дата переговоров», «Город», «Телефон», «Кол-во минут», «Тариф», «Стоимость» (записать структуру таблицы).
4. Записать команды: изменить структуру таблицы, добавления записи, редактирования записи, пометить на удаление запись, удалить записи, восстановить записи, вывод содержимого таблицы на экран.
5. Записать набор команд, с помощью которых можно скопировать данные из одной таблицы в другую.
6. Отсортировать таблицу по фамилии клиента, по дате переговоров, за 1 квартал.

а) записать в БД tab0 алфавитный список всех клиентов (только поле FIO);

б) записать в БД tab1 записи о клиентах, отсортированные по полям: "город", "дата " и "фамилия";

в) записать в БД tab2 отсортированные записи по городам, кол-во минут, за 1 квартал;

г) записать в БД tab3 отсортированные записи по фамилии, стоимости, за месяц;

д) записать в БД tab3 отсортированные записи по тарифу (по возрастанию) и стоимости (по убыванию);

7. Создать индексный файл

а) по лицевому счету;

б) по лицевому счету и дате переговоров;

в) по фамилии, городу;

г) команды создания и открытия индексного файла.

8. Используя фильтрацию записей, вывести следующую информацию:

а) записи о клиентах, у которых в фамилии отсутствуют буквы “Т “и “т”.

б) записи о клиентах, звонивших в понедельник в первом квартале 2012-2014 годах;

в) записи о клиентах, у которых переговоры в текущем месяце стоят более 100 и меньше 1000.

9. Записать командный файл, в котором с помощью команды цикла рассчитывается сумма оплаты за текущий месяц за переговоры.

10. Записать командный файл, в котором по координатам вершин считаются длины сторон треугольника, проверяется: существует ли треугольник (a+b>c). Подсчитывается площадь треугольника. Структура таблицы: (x1, y1, x2, y2, x3,y3, a,b,c, результат «Да», «нет», площадь треугольника)

Задание № 2

1. Описать инфологическую модель **(10 баллов)**.
2. Описать даталогическую модель **(10 баллов)**.
3. Создать запросы **(50 баллов)**:

а) выборка всех полей из описанных таблиц и сохранить в файле (DBF или текстовом файле);

б) сгруппировать по первому ключевому полю таблицы и вывести минимальное, максимальное и среднее значения числового поля (или арифметического выражения);

в) выбор всех полей из двух таблиц, у которых значение поля «дата» соответствует 2 кварталу, вывод сделать в порядке возрастания первого поля в таблицы;

г) ввести значения переменной с клавиатуры, выбрать записи из трех таблиц, которые удовлетворяют заданным значениям с клавиатуры;

е) сгруппировать данные по первому ключевому полю таблицы и выбрать записи с минимальным числовым выражением, большим нуля.

4. Описать структуру отчетов **(20 баллов):**

а) Отчет по всем полям с заголовком и итогом по числовым полям;

б) Отчет с группировкой по ключевому полю (primary), итогами в одной из таблиц.

5. Описать структуру меню **(10 баллов).**

СПИСОК ЛИТЕРАТУРЫ

1. Шапорев Д.С. Visual FoxPro. Уроки программирования.- СПб.: БХВ-Петербург, 2007.-480 с. (20 экз.).
2. Лебедев В.Н. Visual FoxPro 9 / Самоучитель/ 2005. -327с. (50 экз).
3. Фуфаев, Э.В. Базы данных: [текст]: Учеб. пособие/ Э.В. Фуфаев, Д.Э. Фуфаев. - М.: Академия, 2009. - 320 с. (10экз.).
4. Малыхина, М.П. Базы данных: основы, проектирование, использование: [текст]: Учеб. пособие/ М.П. Малыхина. - СПб.: БХВ-Петербург, 2007. - 528 с. – (10экз.).
5. Базы данных: [текст]: Учебник/ Ред. А.Д. Хомоненко. – СПб.: Корона-Век, 2010. - 736 с. (1экз).
6. Кузин, А.В. Базы данных/ А.В. Кузин, С.В. Левонисова. - М.: Академия, 2008. - 320 с. (4экз.).
7. Краткий справочник по Visual FoxPro /http://prog.agava.ru
8. Библиотека программиста. http://www.booksgid.com/16175-.html/
9. Сайт Интернет университета «Информационные технологии»/ http://www.intuit.ru/catalog/database/

Евгения Александровна Дудник

БАЗЫ ДАННЫХ В СУБД Visual FoxPro

Учебно-методическое пособие для студентов, обучающихся
по направлению «Информатика и вычислительная техника»
дневной формы обучения

Редактор Е.Ф. Изотова

Подписано к печати 04.06.15. Формат 60×84 1/16.
Усл.п.л. 6,19. Тираж 35 экз. Заказ 151443. Рег. № 74.

Отпечатано в ИТО Рубцовского индустриального института
658207, Рубцовск, ул. Тракторная, 2/6.