ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

Е.И. МОЛЧАНОВА

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ЭКОНОМИКЕ

Электронный курс лекций

Иркутск 2012

УДК 004:338
ББК 65.39
М 76

Р е ц е н з е н т ы
Н.И. Абасова, к.т.н., доцент;
Н.П. Деканова, д.т.н., профессор

Молчанова Е.И.

М 76 Информационные технологии в экономике : электронный курс лекций. – Иркутск : ИрГУПС, 2012.

Электронный курс лекций предназначен для использования студентами в рамках федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки 080100 – Экономика.

Рассмотрены технологии доступа к данным в реляционных СУБД на примере Visual FoxPro, элементы объектно-ориентированного программирования в Visual FoxPro, вопросы использования компьютерных сетей.

Теоретический материал сопровождается большим количеством иллюстраций и примеров. Для проверки усвоения материала предлагаются контрольные вопросы и задания для компьютерного тестирования.

УДК 004:338
ББК 65.39

Оглавление

Литература

Учебники **(***библиотека)*

1. Информатика для юристов и экономистов. Учебник для вузов / Под ред. Симоновича С.В. - СПб.: ПИТЕР, 2001.
2. Экономическая информатика. Учебник / Под ред. Конюховского П.В., Колесова Д.Н. - СПб.: ПИТЕР, 2000.
3. Омельченко Л.Н. Самоучитель Visual FoxPro 7.0.

Методические указания **(***библиотека)*

По курсу «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ЭКОНОМИКЕ» часть 1, часть 2

Учебное пособие **(***библиотека)*

По СУБД Visual FoxPro

Базы данных и информационные системы

План лекции

1. **Компоненты информационных систем**
2. **Архитектура баз данных**
3. **Архитектура информационной системы**
4. **Модели данных**
5. **Проблемы проектирования реляционных БД**

Компоненты информационных систем

Информационная система - это механизм для хранения, поиска, извлечения и модификации информации.

Современные информационные системы базируются на компьютерных технологиях.

Анализ современного состояния рынка ИС показывает устойчивую тенденцию роста спроса на информационные системы организационного управления. Причем спрос продолжает расти именно на *интегрированные системы* управления. Автоматизация отдельной функции, например, бухгалтерского учета или сбыта готовой продукции, считается уже пройденным этапом для многих предприятий.

Ниже приведен перечень наиболее популярных в настоящее время программных продуктов для реализации ИС организационного управления различных классов.

Классификация рынка информационных систем

Локальные системы	Малые интегрированные системы	Средние интегрированные системы	Крупные интегрированные системы (IC)
•БЭСТ •Инотек •Инфософт •Супер-Менеджер •Турбо-Бухгалтер •Инфо-Бухгалтер	•Concorde XAL Exact •NS-2000 Platinum PRO/MIS •Scala SunSystems •БЭСТ-ПРО •1С-Предприятие •БОСС-Корпорация •Галактика •Парус •Ресурс •Эталон	•Microsoft-Business Solutions - Navision, Axapta •J D Edwards (Robertson & Blums) •MFG-Pro (QAD/BMS) •SyteLine (СОКАП/SYMIX)	•SAP/R3 (SAP AG) •Baan (Baan) •BPCS (ITS/SSA) •OEBS (Oracle E-Business Suite)

Существует *классификация ИС* в зависимости от **уровня управления**, на котором система используется:

1. Информационные системы оперативного уровня.
2. Информационные системы специалистов.
3. Информационные системы уровня менеджмента.
4. Стратегические информационные системы.

Информационная система *оперативного уровня* - *поддерживает исполнителей*, обрабатывая данные о сделках и событиях (счета, накладные, зарплата, кредиты, поток сырья и материалов).
Информационная система оперативного уровня является связующим звеном между фирмой и внешней средой.

Задачи, цели, источники информации и алгоритмы обработки на оперативном уровне заранее определены и в высокой степени структурированы.

Информационные системы специалистов - поддерживают работу с данными и знаниями, повышают продуктивность и производительность работы инженеров и проектировщиков.
Задача подобных информационных систем - интеграция новых сведений в организацию и помощь в обработке бумажных документов.

Информационные системы уровня менеджмента - используются работниками среднего управленческого звена для мониторинга, контроля, принятия решений и администрирования.

Основные функции этих информационных систем:

- сравнение текущих показателей с прошлыми;
- составление периодических отчетов за определенное время, а не выдача отчетов по текущим событиям, как на оперативном уровне;
- обеспечение доступа к архивной информации и т.д.

Стратегическая информационная система - компьютерная информационная система, обеспечивающая поддержку принятия решений по реализации стратегических перспективных целей развития организации.

Информационные системы стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, осуществлять долгосрочное планирование.

Основная задача - сравнение происходящих во внешнем окружении изменений с существующим потенциалом фирм, создать общую среду компьютерной телекоммуникационной поддержки решений в неожиданно *возникающих ситуациях*. Используя самые совершенные программы, эти системы способны в любой момент предоставить информацию из многих источников.

Информационная система включает: вычислительную систему, одну или несколько баз данных (БД), систему управления базами данных (СУБД) и набор прикладных программ, а также обслуживающий персонал.

База данных - это поименованный набор структурированной информации, предназначенной для совместного употребления несколькими пользователями.

Структурированность - отдельные элементы данных связаны между собой логическими связями.

Именно понятия "структурированность" и "для совместного использования" отличают базу данных от простых файлов данных.

Классический пример информационной системы - система бронирования билетов.

Основная ценность базы данных заключается в самих данных, в возможности быстрого поиска и модификации этих данных.

Система управления базами данных (СУБД) - программные средства, позволяющие пользователям работать с БД.

С помощью СУБД можно создавать и модифицировать БД и разрабатывать пользовательские приложения.

Пользовательские приложения - это прикладные программы, которые служат для автоматизации обработки данных БД, вычислений и формирования выходных документов.

Приложения могут создаваться как в среде СУБД, так и вне СУБД, с помощью систем программирования, например, DELPHI или C^{++}Builder.

Архитектура баз данных

Архитектура- описание сложной системы, состоящей из множества элементов, как единого целого.
В вычислительной технике архитектура определяет состав, назначение, логическую организацию и порядок взаимодействия всех аппаратных и программных средств, объединенных в единую вычислительную систему.
Иными словами, архитектура описывает то, как БД представляется пользователю.

Существует три основных варианта : *хост - архитектура; файл – серверная архитектура и архитектура клиент - сервер*.

Хост – архитектура появилась в 70-х годах. *Вся обработка данных осуществляется на хост -компьютере*. Пользователи связываются с хостом при помощи терминала или персонального компьютера (ПК).

Система резервирования авиабилетов.

Многие хост-системы за последние годы преобразованы в файл – серверные или клиент – серверные.

Файл – серверная архитектура появилась в 80-х годах.

Вся работа с БД осуществляется на пользовательских компьютерах, но данные хранятся на сервере локальной сети.

Все настольные БД возникли на основе этой архитектуры, например, *Visual Fox Pro*, *dBase*, *Access*, *Paradox*, *MS Jet*.

БД здесь просто совокупность файлов.

Архитектура клиент – сервер появилась в начале 90-х годов. *Процесс обработки БД разделен: одна его часть выполняется на ПК клиента, а другая – на сервере. На сервере, выполняются действия по поддержанию и защите БД. На компьютере пользователя выполняются функции поддержки «бизнес – правил», т.е. проверки данных на допустимость*. Такой подход позволяет достичь наибольшей производительности.

Приложениями архитектуры клиент - сервер являются программы *Oracle*, *Sybase*, *SQL- Server*, *Informix и MS SQL – Server*. Эти приложения называют БД SQL, т.к. язык работы с БД это SQL.

Преимущества - высокая производительность, безопасность и защита информации.

Недостаток – необходимость администрирования и высокая стоимость.

Архитектура информационной системы

С точки зрения **программно-аппаратной реализации** можно выделить ряд типовых архитектур ИС.

Традиционные архитектурные решения основаны на использовании выделенных файл-серверов или серверов баз данных. Существуют также варианты архитектур корпоративных информационных систем, *базирующихся на технологии* Internet (*Intranet-приложения*). Следующая разновидность архитектуры информационной системы *основывается на концепции "хранилища данных" (DataWarehouse) - интегрированной информационной среды, включающей разнородные информационные ресурсы*. И, наконец, для построения глобальных распределенных информационных приложений используется *архитектура интеграции информационно-вычислительных компонентов на основе объектно-ориентированного подхода*.

В настоящее время наиболее распространенной является архитектура клиент-сервер. Она предполагает наличие компьютерной сети и включает корпоративную БД и персональные БД. Корпоративная БД (многопользовательская) размещается на компьютере - сервере. Персональные БД размещаются на компьютерах сотрудников.

Компьютер-сервер осуществляет централизованное хранение и обслуживание корпоративной информации. Компьютеры-клиенты обеспечивают коллективный доступ к корпоративной БД. На компьютере-клиенте может также находиться *своя персональная БД*.

Корпоративные, многопользовательские БД создаются СУБД - серверами, например, Microsoft SQL Server Oracle Server, Informix.

Персональные БД создаются СУБД, такими, как Access, Visual FoxPro фирмы Microsoft или Paradox фирмы Borland.

В настоящее время границы между СУБД серверными и клиентскими стираются - рынки сбыта.

Модели данных

Модель данных - логическая структура данных БД, поддерживаемая СУБД.
К числу важнейших относятся следующие: иерархическая, сетевая, реляционная, объектно-ориентированная.
В иерархической модели данные представляются в виде древовидной (иерархической) структуры*. Она удобна для работы с иерархически упорядоченной информацией.*
Например, база данных для описания тематических сборников по некоторому предмету (рис.1).

Здесь *скорость поиска данных высока, если структура запроса совпадает со структурой данных*, например, просмотреть статьи, опубликованные в сборнике. ***Трудно осуществить поиск информации со сложными логическими связями***. Например, в скольких сборниках данный автор опубликовал статьи. Еще один *недостаток* - ***сложность внесения изменений (жесткость).***

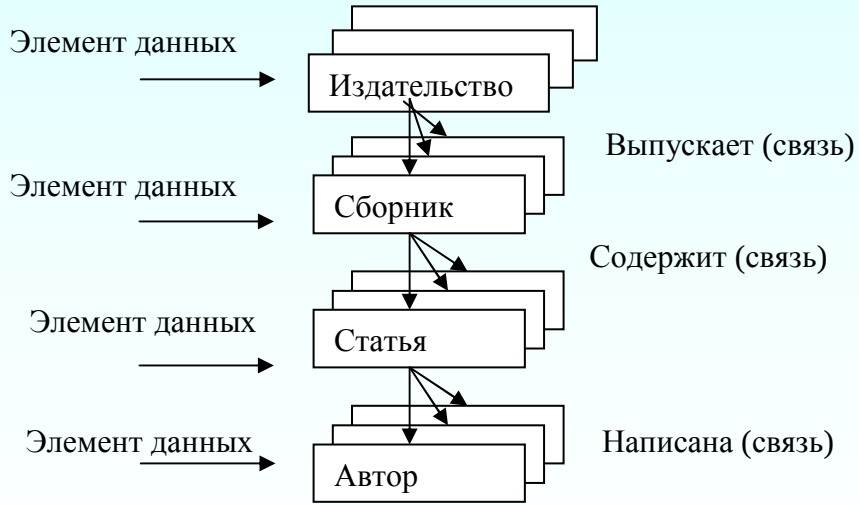


Рис.1. Иерархическая модель данных издательства

Сетевая модель* - является развитием иерархической.** ***Главное отличие, что к каждому элементу может идти связь не от одного элемента ("родителя"), а от нескольких. Например, генеалогическое дерево (рис.2), построенное только по мужской линии, является древовидной иерархической структурой, у каждого человека 1 родитель. Если включить обоих родителей, то это будет уже не дерево, а сеть (рис.3). Недостаток модели - ее жесткость.

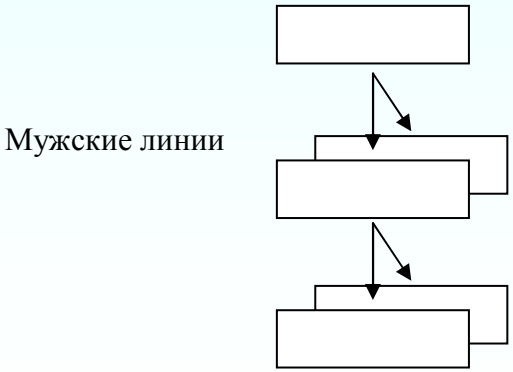
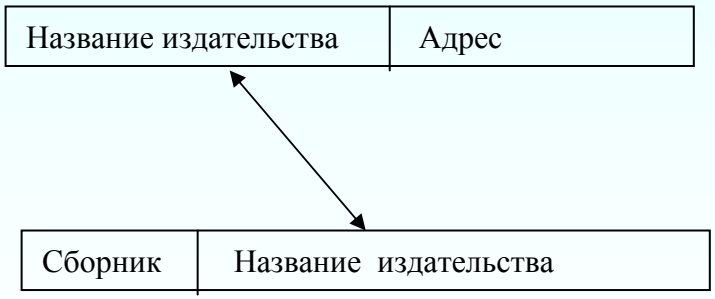


Рис.2. Иерархическая модель данных генеалогического дерева человека

Рис.3. Фрагмент сетевой модели данных

Реляционная модель. *Для реляционной СУБД выбрано представление данных на основе математического понятия отношение. Оно близко к понятию двумерной таблицы. По-английски, отношение это relation.*

Реляционная база данных - это набор таблиц. *Связь между таблицами строим динамически, в процессе поиска информации, через сравнение значений полей в разных таблицах.*



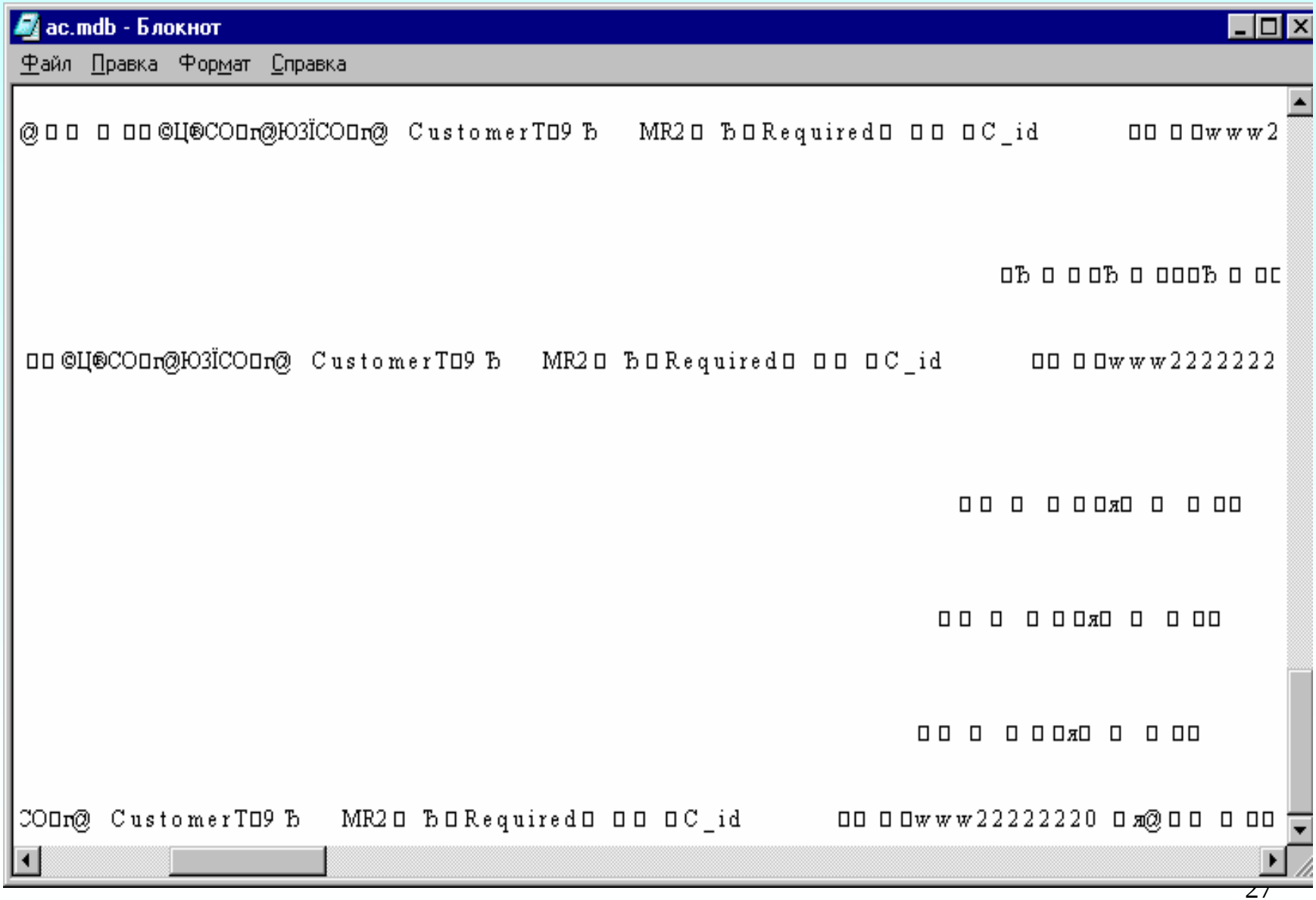
В нашем примере с издательством надо завести ряд таблиц. Например, первая с описанием издательств, вторая с описанием сборников.

По названию сборника можно найти адрес издательства

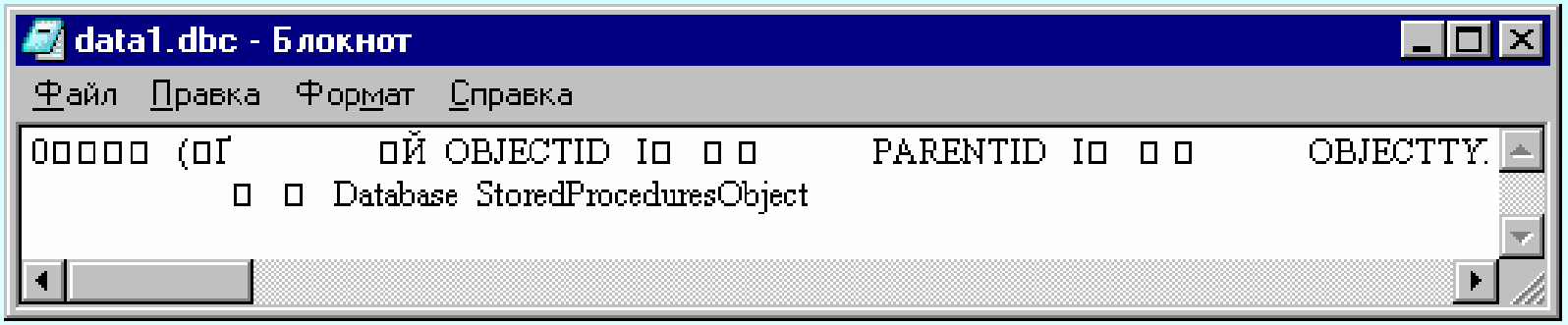
Основным достоинством реляционных СУБД является нефункциональность языков запроса. То есть мы формулируем не то, как нам надо найти данные, а что нам надо найти. ***Еще одним достоинством реляционных СУБД является высокая стандартизованность.*** Практически все производители СУБД поддерживают стандарты языка SQL.

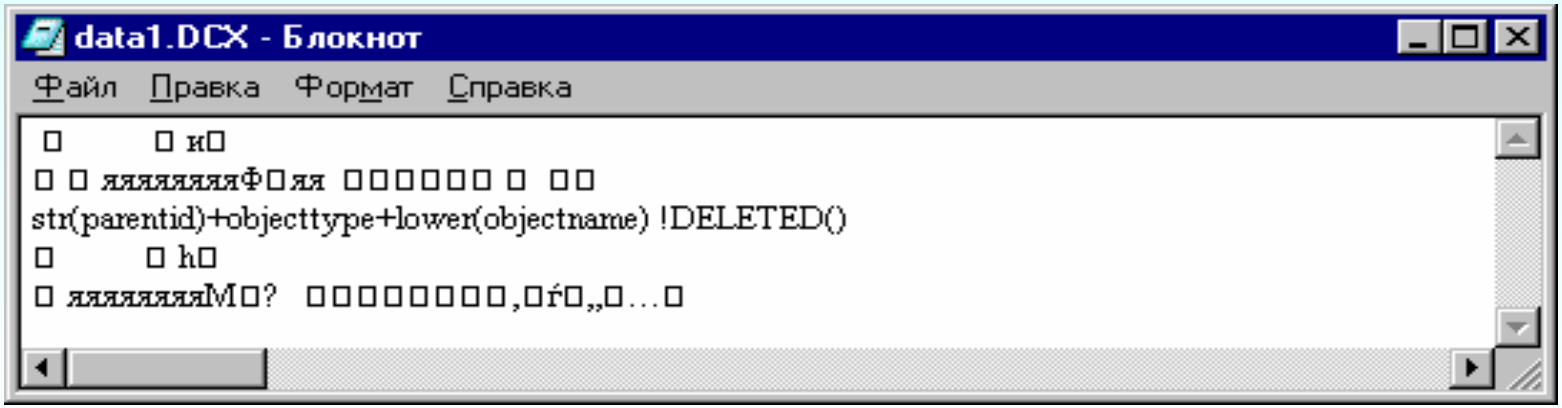
Реляционная БД – совокупность связанных таблиц.
БД созданные разными СУБД имеют разный формат, т.е. способ представления данных.
Файл БД Access *.mdb.
Файл БД VisualFoxPro - *.dbc, *.dbt, *dcx.
Файл БД хранит информацию о таблицах и связях между ними.

Фрагмент файл БД Access в окне текстового редактора



Фрагмент файлов БД VFP в окне текстового редактора





В каждой таблице содержится информация о объектах одного типа.
Таблица состоит из строк (записей) и столбцов (атрибутов, полей) и имеет уникальное имя в базе данных.

Строка хранит информацию об отдельном объекте, а столбец- представляет его отдельную характеристику (атрибут).

Создание таблицы – создание структуры таблицы.

Структура таблицы - информация о числе записей, количестве и наименовании столбцов, типе данных и т.д. В VFP она сохраняется в нулевой строке таблицы (строке заголовка).

Типы данных полей Visual FoxPro

Тип	**Наименование**
Текстовый	**Character**
Числовой	**Integer, Numeric, Float, Double**
Денежный	**Currency**
Дата	**Date**
Логический	**Logical**
Двоичное поле	**General**
Текстовое поле	**Memo**

Пример таблицы Клиенты (Customer)

Идентификатор клиента	Наименование фирмы клиента	Адрес клиента
C_id	*C_name*	*C_address*
10	Багира	Иркутск, Чкалова, 3
11	Гермес	Иркутск, Ленина, 15
12	Труд	Иркутск, Южная, 20
08	ЧП Беляев	Иркутск, Чехова,4
07	Луна и яичница	Иркутск, Челнокова, 15

back

Пример таблицы Заказы (Order)

Идентификатор заказа	Идентификатор клиента	Дата заказа	Дата выполнения
Or_id	*C_id*	*Data1*	*Data2*
101	10	01.04.00	02.10.00
102	11	01.02.00	03.05.00
103	12	05.11.00	02.12.00
104	08	06.12.00	20.12.00
105	07	05.02.00	09.04.00

Пример таблицы пункты заказа OrderLines

Идентификатор заказа	Пункт заказа	Идентификатор продукта
Or_id	***Or_lines***	*Prod_id*
101	1	005
101	2	006
101	3	011
102	1	002
102	2	005
103	1	009
104	1	001
104	2	005
105	1	006

Для реальной работы с объектами, описанными в таблице, нужно уметь их различать.

Ключ - это набор атрибутов (столбцов), которые позволяют идентифицировать запись внутри таблицы (отношения).

Ключ может быть простым и составным.

Простой ключ состоит из одного атрибута. (таблицы Клиенты (Customer) и Заказы (Order)).

Составной ключ состоит из двух или более атрибутов. (таблица Пункты заказа (OrderLines)).

user:

C_id в Customer

Ключ отношения, однозначно идентифицирующий запись, называется первичным (значения не повторяются -Primary).

Вторичным ключом называется атрибут (или группа атрибутов, значение которых может повторяться для нескольких записей -Regular). *Вторичные ключи используются в операциях поиска записей.*

user: *C_id в Order*

Ключи таблиц создают с помощью индексов.

Индексы

Назначение
- Быстрый поиск информации
- Служат ключами

Количество полей
- Простой индекс - одно поле
- Составной индекс – список из нескольких полей или ***индексное выражение***

индексное выражение формируется из имен столбцов, соединенных знаками арифметических операций.

Индекс хранит упорядоченные по возрастанию (Ascending) или убыванию (Descending) значения индексных столбцов или выражений и ссылку на связанную с ним запись.

Индексов может быть несколько, в зависимости от того, какую информацию необходимо часто извлекать из таблиц

Типы индексов

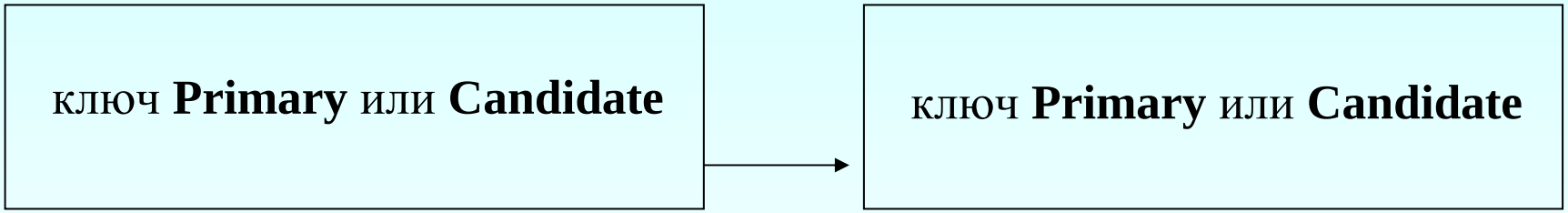
Тип индекса	Описание	Назначение
Primary (Первичный)	Значения индексного выражения ***не должны*** повторяться.	Для связывания таблиц и определения условий целостности данных.
Candidate (Потенциальный)	Обладает всеми качествами первичного ключа, но таблица не может содержать более одного первичного ключа.	Для связывания таблиц и поиска данных.
Regular (Обычный)	Значения индексного выражения ***могут*** повторяться. Каждое значение хранится отдельно.	Для связывания таблиц и поиска данных.
Unique (Уникальный)	Значения индексного выражения могут повторяться, но хранится из них только одно и ссылка на первую из записей с одинаковым значением.	Для поиска данных

Отношения (связи) между таблицами

БД обычно включает несколько таблиц, между которыми установлены постоянные отношения. Такие таблицы называются ***связанными****.*
Из двух связанных таблиц одна является главной, а другая — подчиненной. Главную таблицу называют ***родительской*** *(первичной), а подчиненную —* ***дочерней*** *(вторичной).*
Между таблицами существует четыре типа отношений:

один-к-одному *- каждая запись в родительской таблице соответствует только одной записи в дочерней.*

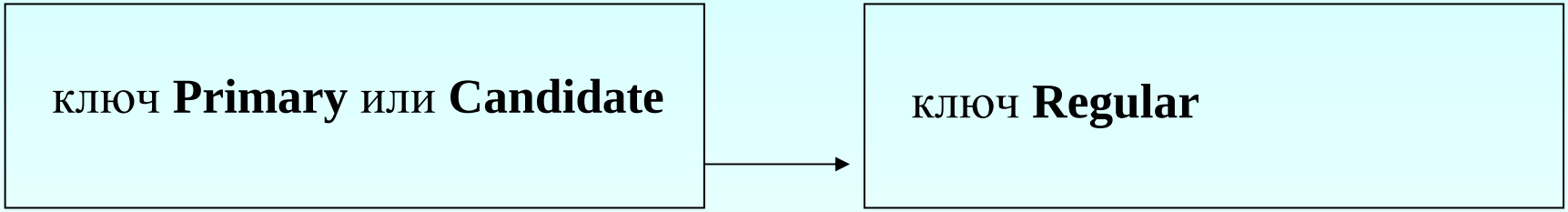
Родительская **Дочерняя**



один - ***к-одному***

один-ко-многим *- каждой записи в родительской таблице соответствует несколько записи в дочерней.*

Родительская **Дочерняя**



один - ***…-ко-многим***

многие-к-одному *– отношение один-ко-многим, рассматриваемое с другой точки зрения.*

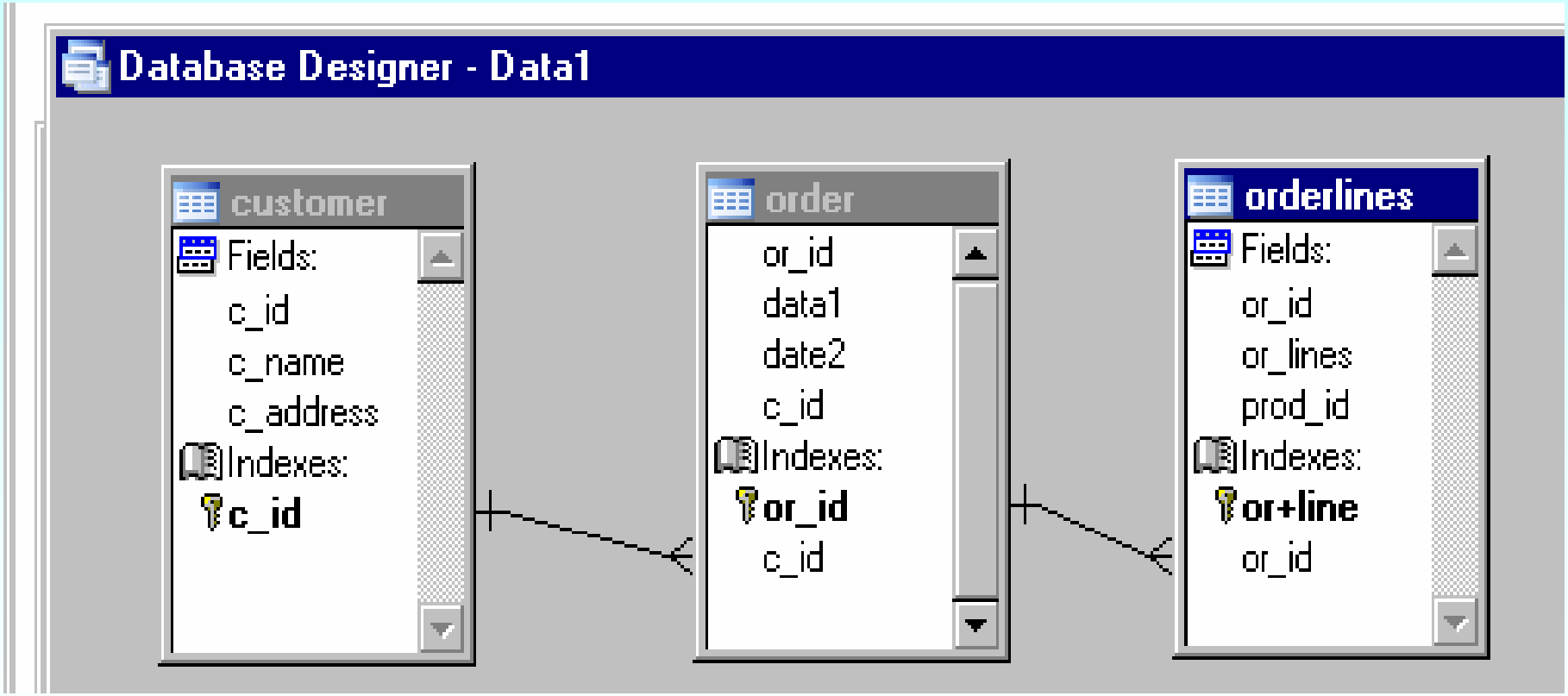
многие-ко-многим - *для записей каждой из таблиц имеется несколько соответствующих записей в другой таблице.*
Данный тип отношений может потребовать внесения *изменений в структуру базы данных*. С этой целью используется связующая таблица. Связующая таблица, создается для преобразования отношений типа многие-ко-многим между двумя таблицами в отношения типа один-ко-многим. Связующая таблица содержит первичные ключи обеих таблиц, связанных отношением многие-ко-многим.

При создании БД предпочтительнее использовать первые два типа отношений.

Создание связей

Связь таблиц устанавливается посредством совпадающих значений индексных выражений.
В родительской таблице из числа Candidate Key необходимо выбрать один Primary Key (первичный ключ), чтобы на него ссылались другие таблицы для получения информации из любой строки данной таблицы.
Когда этот ключ используется во второй (дочерней) таблице для получения ссылки на строки первой (родительской) таблицы, во второй таблице он называется Foreign Key (внешний ключ).

Поэтому, в родительской и в дочерней таблицах должны быть созданы индексы, использующие одинаковые индексные выражения.
Необходимо следить за совпадением в обеих таблицах типа данных и ширины ключевых полей!
В таблицах Клиенты (Customer) и Заказы (Order) совпадающим является поле C_id. Главная – Клиенты – индекс Primary, дочерняя Заказы – индекс Regular (см. таблицы). В таблицах Заказы (Order) и Пункты заказа (OrderLines) совпадающим является поле Or_id. Главная - Заказы– индекс Primary, дочерняя Пункты заказа – индекс Regular (см. таблицы).



Окно конструктора БД

Целостность данных

Целостность – состояние данных, при котором они сохраняют свое информационное содержание в условиях различных воздействий. Например, при изменении значений ключевых полей.

Значение ключевого поля в родительской таблице можно:

- ***Update (Изменить);***
- ***Delete (Удалить);***
- ***Insert (Вставить).***

Автоматически можно выполнить:

- **Cascade *(каскадные изменения)*** - изменение значения ключа (или соответственно каскадное удаление) всех записей из дочерней таблицы, связанных с изменяемой (или удаляемой) записью .
- **Restrict *(запрет изменений)*** не позволяет изменять ключевое поле (или удалять записи) в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на изменяемую (удаляемую) запись.
- **Ignore *(игнорировать)*** позволяет изменять (удалять) записи в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается.

Проблемы проектирования реляционных БД

Проектирование БД осуществляется на физическом и логическом уровнях.

Решение проблемы на физическом уровне во многом автоматизировано и скрыто от пользователя.

Логическое проектирование включает: определение числа и структуры таблиц, формирование запросов к БД, определение типов отчетных документов, создание форм и т.д.

На этапе проектирования БД осуществляется определение состава и структуры данных предметной области.

Структура данных предметной области может отображаться информационно-логической моделью.

Информационно-логическая модель **отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет структуру данных, подлежащих хранению в базе данных.**

Информационный объект **(ИО) – информационное описание некоторой сущности предметной области – реального объекта, процесса, явления или события. Информационный объект образуется совокупностью логически связанных *атрибутов*, представляющих качественные и количественные характеристики сущности.**

Методы разработки баз данных

Наиболее часто используют два метода проектирования БД.

Первый - ER-метод (Essence-Relationship), что означает сущность-связь. Он основан на построении ***ER-диаграмм***.

Сущности (таблицы) отображаются прямоугольниками, список атрибутов (столбцов) помещают внутри них.

Первичный ключ помещают в верхнюю строку.

Связи представляют в виде направленных стрелок.

Второй– метод нормальных форм. Основан на выявлении функциональных зависимостей между атрибутами (столбцами) отношения (таблицы).

Для реализации ER- метода разработки модели данных необходимо:

<u>На первом этапе</u> - выделить информационные объекты, соответствующие требованиям нормализации данных.

<u>На втором этапе</u> – выделить одно - многозначные отношения между информационными объектами.

<u>На третьем этапе</u> каждый информационный объект отобразить реляционной таблицей и установить связи между таблицами. Результат- логическая структура БД.

Рассмотрим первый этап.

Для выделения информационных объектов из описания предметной области выделяют существительные и определяют их атрибуты (характеристики).

Информационный объект имеет множество реализаций (экземпляров). Студент, Преподаватель, Кафедра, Факультет…

Экземпляр объекта образуется совокупностью конкретных значений атрибутов и однозначно определяется значением ключа.

Т.Е. **атрибуты подразделяются на ключевые и описательные. Описательные атрибуты функционально зависят от ключа**.

Зависеть – одновременно изменяться.

Функциональная полная зависимость описательного атрибута от ключа означает, что одному значению ключа соответствует только одно значение описательного (зависимого атрибута).

Совокупность атрибутов выделенного информационного объекта должна отвечать требованиям нормализации:

1. ***ИО содержит уникальный идентификатор (ключ).***
2. ***Все описательные атрибуты взаимонезависимы.***
3. ***Каждый описательный атрибут полностью зависит от ключа.***
4. ***При составном ключе описательный атрибут зависит от всей совокупности атрибутов, образующих ключ.***
5. ***Описательный атрибут не зависит от ключа транзитивно, т.е. через другой промежуточный реквизит.***

Для достижения - расщепление совокупности реквизитов.

Нормализация

Теоретически можно хранить информацию в одной большой таблице. *Недостатки такой большой универсальной таблицы сводятся к следующему:*

- *жесткость (обязательная модификация таблицы при изменении постановки задачи);*
- *ненадежность (потенциальная противоречивость);*
- *повышенный расход ресурсов;*
- *громоздкость (избыточность).*

Процесс упорядочивания, структурирования представления данных называется нормализацией. Цель нормализации – приведение в порядок функциональных зависимостей, т.е. все столбцы должны зависеть от первичного ключа этой таблицы, но не должны зависеть друг от друга или от первичного ключа другой таблицы.

Процесс нормализации основан на идее функциональной зависимости значения столбцов от одного или более других столбцов.

Например, от номера зачетной книжки в таблице СТУДЕНТЫ зависит фамилия студента.

Теория нормализации оперирует с пятью нормальными формами таблиц. Каждая последующая нормальная форма должна удовлетворять требованиям предыдущей и некоторым дополнительным условиям. *При практическом проектировании БД четвертая и пятая формы, как правило, не используются*, в связи с чем, мы ограничимся рассмотрением первых трех нормальных форм.

Таблица в **первой** нормальной форме должна удовлетворять следующим требованиям:

- *все атрибуты (столбцы) таблицы должны быть простыми* (не составными);
- *в таблице должен быть столбец, однозначно идентифицирующий строку*. Таких столбцов может быть несколько - **Candidate Key** (возможный ключ). Из них выбирают один **Primary Key** (первичный ключ), чтобы на него ссылались другие таблицы для получения информации из любой строки данной таблицы. Когда **Primary Key** используется во второй таблице для получения ссылки на строки первой таблицы, во второй таблице он называется **Foreign Key** (внешний ключ) (см.14);
- *таблица не должна иметь повторяющихся строк (записей);*
- *таблица не должна иметь повторяющихся групп столбцов;*
- *строки и столбцы должны быть не упорядочены.*

Первое условие достигается с помощью разбиения составных значений столбцов на простые.

Для удовлетворения второго условия каждая таблица должна иметь уникальный ключ.

*Таблица во **второй** нормальной форме должна* удовлетворять следующим требованиям:

- *иметь первую нормальную форму;*
- *каждый не ключевой атрибут (столбец) зависит от ключа целиком, а не от его отдельных составляющих.*

Зависимость здесь означает, что, зная один атрибут, можно определить соответствующее значение зависимого атрибута.

Если все ключи в таблице состоят только из одного атрибута, то отношение автоматически имеет вторую нормальную форму, В нашем примере с приказом о назначении стипендии. *Для приведения таблиц ко второй нормальной форме выделяют атрибуты, зависящие только от одного из компонентов составного ключа, и помещают их в отдельную таблицу.*

Составной ключ может включать комбинацию нескольких отдельных атрибутов (столбцов).

*Таблица в **третьей** нормальной форме должна удовлетворять следующим требованиям:*

- *иметь вторую нормальную форму;*
- *неключевые атрибуты зависят только от ключа, а не от других неключевых столбцов.*

Для приведения таблиц к третьей нормальной форме выделяют атрибуты, зависящие от других неключевых атрибутов, и помещают их в отдельную таблицу.

Замечание

Следует избегать первичных ключей, имеющих собственный смысл. Лучшим типом первичного ключа является простой ключ, например, возрастающее простое число.

Рассмотрим классический *пример нормализации* таблиц, содержащих информацию *о клиентах фирмы и сделанных ими заказах*.

Предположим, для выполнения заказа необходимо оперировать следующими данными, представленными в табл.1.

Таблица 1

Список возможных атрибутов таблиц с данными о заказах фирмы

Идентификатор заказа	Пункт заказа	Идентификатор клиента	Наименование фирмы клиента	Адрес клиента	Дата заказа	Дата выполнения	Идентификат продукта
Or_id	*Or_lines*	*C_id*	*C_name*	*C_address*	*Data1*	*Data2*	*Prod_id*
101	1,2,3	10	Багира	Иркутск, Чкалова, 3	01.04.00	02.10.00	005, 006, 011
102	1,2	11	Гермес	Иркутск, Ленина, 15	01.02.00	03.05.00	002, 005
103	1	12	Труд	Иркутск, Южная, 20	05.11.00	02.12.00	009
104	1,2	08	ЧП Беляев	Иркутск, Чехова,4	06.12.00	20.12.00	001, 005
105	1	07	Луна и яичница	Иркутск, Челнокова, 15	05.02.00	09.04.00	006

Проверим, удовлетворяет ли такая таблица требованиям первой нормальной формы.

Таблица 1а

Вариант разбиения значения составного атрибута Or_lines на отдельные компоненты

Идентификатор заказа	Пункт 1	Пункт 2	Пункт 3
Or_id	*Or_lines1*	*Or_lines2*	*Or_lines3*
101	1	2	3
102	1	2	-
103	1	-	-
104	1	2	-
105	1	-	-

Однако это приведет к повторению данных в столбцах и непроизводительному увеличению числа столбцов, особенно, если пунктов заказа может быть много. Лучше собрать значения Or_lines в один столбец и повторять строки для одного заказа, как показано в табл.2.

Таблица 2.2

Пример таблицы в первой нормальной форме

Идентификатор заказа	Пункт заказа	Идентификатор клиента	Наименование фирмы клиента	Адрес клиента	Дата заказа
Or_id	***Or_lines***	*C_id*	*C_name*	*C_address*	*Data1*
101	1	10	Багира	Иркутск, Чкалова, 3	01.04.00
101	2	10	Багира	Иркутск, Чкалова, 3	01.04.00
101	3	10	Багира	Иркутск, Чкалова, 3	01.04.00
102	1	11	Гермес	Иркутск, Ленина, 15	01.02.00
102	2	11	Гермес	Иркутск, Ленина, 15	01.02.00
103	1	12	Труд	Иркутск, Южная, 20	05.11.00
104	1	08	ЧП Беляев	Иркутск, Чехова,4	06.12.00
104	2	08	ЧП Беляев	Иркутск, Чехова,4	06.12.00
105	1	07	Луна и яичница	Иркутск, Челнокова, 15	05.02.00

Столбцы Data1, Data2, C_id, C_name, C_address зависят только от одной составляющей первичного ключа **Or_id** и не зависят от столбца **Or_lines**, что *противоречит требованиям второй* нормальной формы.

Чтобы выполнить требования *второй нормальной формы*, разделим исходную таблицу на две. Таблицу OrderLines (пункты заказа) табл.3 .

Таблица 3

Пример таблицы OrderLines во второй нормальной форме

Идентификатор заказа ***Or_id***	Пункт заказа ***Or_lines***	Идентификатор продукта *Prod_id*
101	1	005
101	2	006
101	3	011
102	1	002
102	2	005
103	1	009
104	1	001
104	2	005
105	1	006

Вторая таблица - Order (заказы).

Таблица 3а

Пример таблицы Order во второй нормальной форме

Идентификатор заказа	Идентификатор клиента	Наименование фирмы клиента	Адрес клиента	Дата заказа	Дата выполнения
Or_id	*C_id*	*C_name*	*C_address*	*Data1*	*Data2*
101	10	Багира	Иркутск, Чкалова, 3	01.04.00	02.10.00
102	11	Гермес	Иркутск, Ленина, 15	01.02.00	03.05.00
103	12	Труд	Иркутск, Южная, 20	05.11.00	02.12.00
104	08	ЧП Беляев	Иркутск, Чехова,4	06.12.00	20.12.00
105	07	Луна и яичница	Иркутск, Челнокова, 15	05.02.00	09.04.00

Таблица *OrderLines* отвечает требованиям третьей нормальной формы.
В таблице *Order* столбцы C_name и C_address зависят не только от первичного ключа **Or_id**, но и от неключевого атрибута C_id, что противоречит требованиям третьей нормальной формы.
Выделим таблицу Customer (клиенты) в отдельную таблицу.

Таблица 4

Пример таблицы Customer в третьей нормальной форме

Идентификатор клиента	Наименование фирмы клиента	Адрес клиента
C_id	*C_name*	*C_address*
10	Багира	Иркутск, Чкалова, 3
11	Гермес	Иркутск, Ленина, 15
12	Труд	Иркутск, Южная, 20
08	ЧП Беляев	Иркутск, Чехова,4
07	Луна и яичница	Иркутск, Челнокова, 15

Таблица 4а

Пример таблицы Order в третьей нормальной форме

Идентификатор заказа	Идентификатор клиента	Дата заказа	Дата выполнения
Or_id	*C_id*	*Data1*	*Data2*
101	10	01.04.00	02.10.00
102	11	01.02.00	03.05.00
103	12	05.11.00	02.12.00
104	08	06.12.00	20.12.00
105	07	05.02.00	09.04.00

Таблица OrderLines без дополнительных преобразований отвечает требованиям третьей нормальной формы.

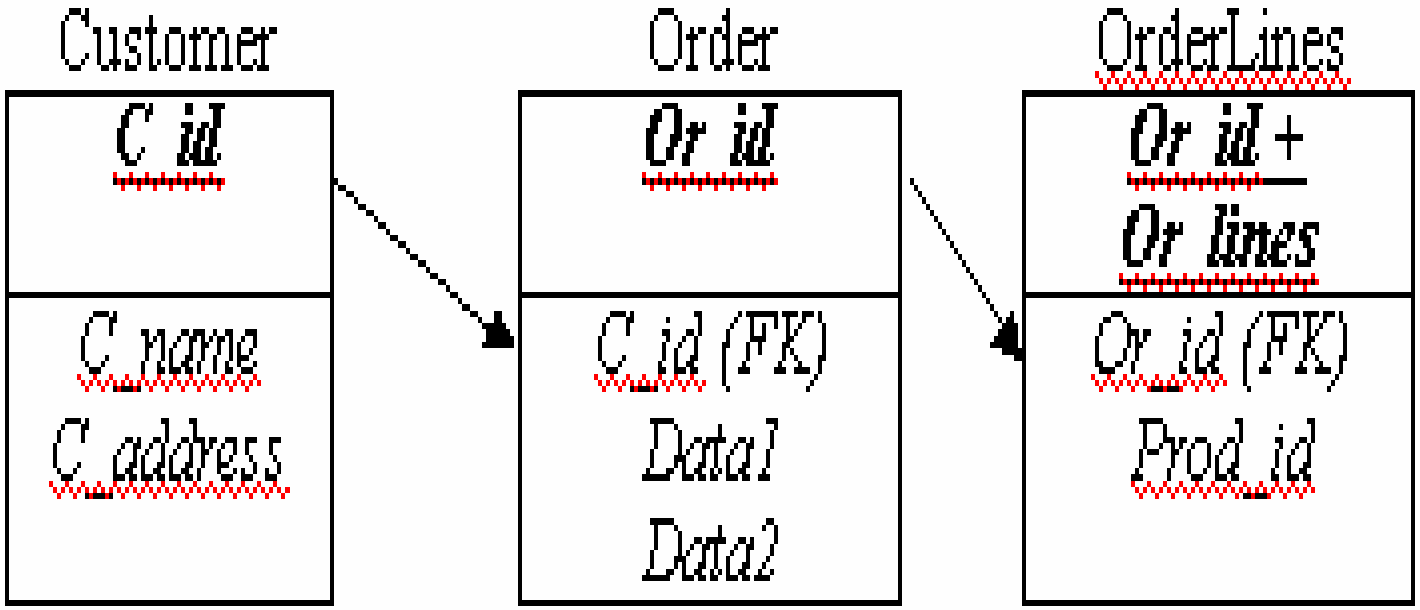
В результате получим набор из трех таблиц, находящихся в третьей нормальной форме:

Customer(**C_id**[1], C_name, C_address)

Order(**Or_id**, C_id, Data1, Data2)

OrderLines(**Or_id, Or_lines**, Prod_id).

[1] Подчеркнуты атрибуты первичного ключа таблицы.



Графическое изображение объектов в ER-методе

Этапы проектирования

1. Выделение сущностей и связей между ними.
2. Построение ***ER-диаграмм*** с учетом всех сущностей и их связей.
3. Формирование предварительного набора отношений (1:1, 1:M, M:1, M:M) между сущностями.
4. Добавление неключевых атрибутов в отношения.
5. Приведение предварительных отношений к третьей нормальной форме.
6. Пересмотр ***ER-диаграмм*** (при необходимости возврат к этапу 1).

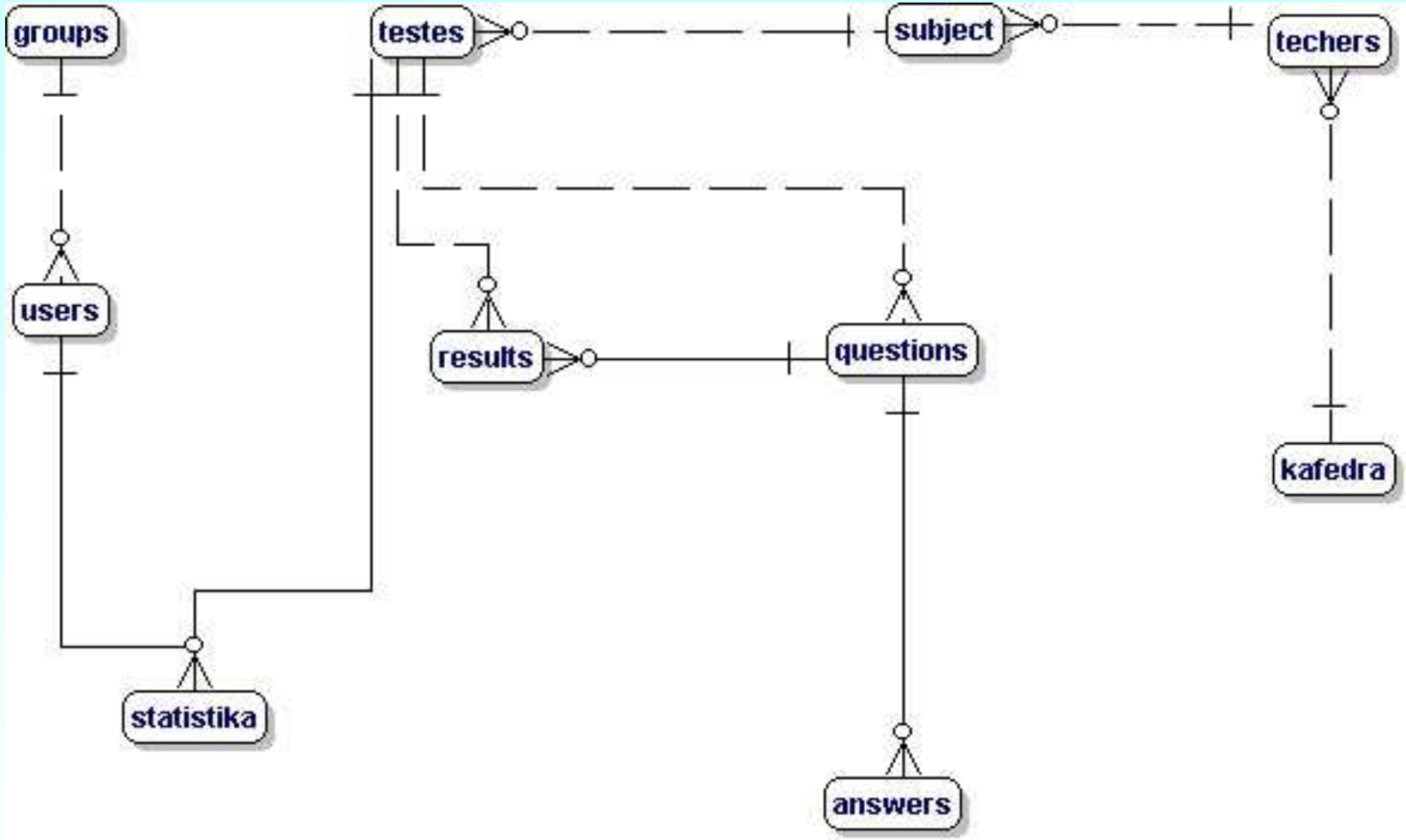
Созданы специальные программные пакеты, основанные на использовании этого метода, *например, ErWin/ERX фирмы Logic Works. Отметим, что нормализация здесь должна выполняться в ручную.*

Недавно появился новый ORM-метод (Object Role Modeling), что означает объектно-ролевое моделирование. Например, программа Infomodeler компании Asymetryx. Версия ORM для ПК включает драйверы FoxPro. Метод гарантирует получение полностью нормальных таблиц и хорошо подходит для сложного моделирования, когда заранее неизвестно, как должны выглядеть таблицы.

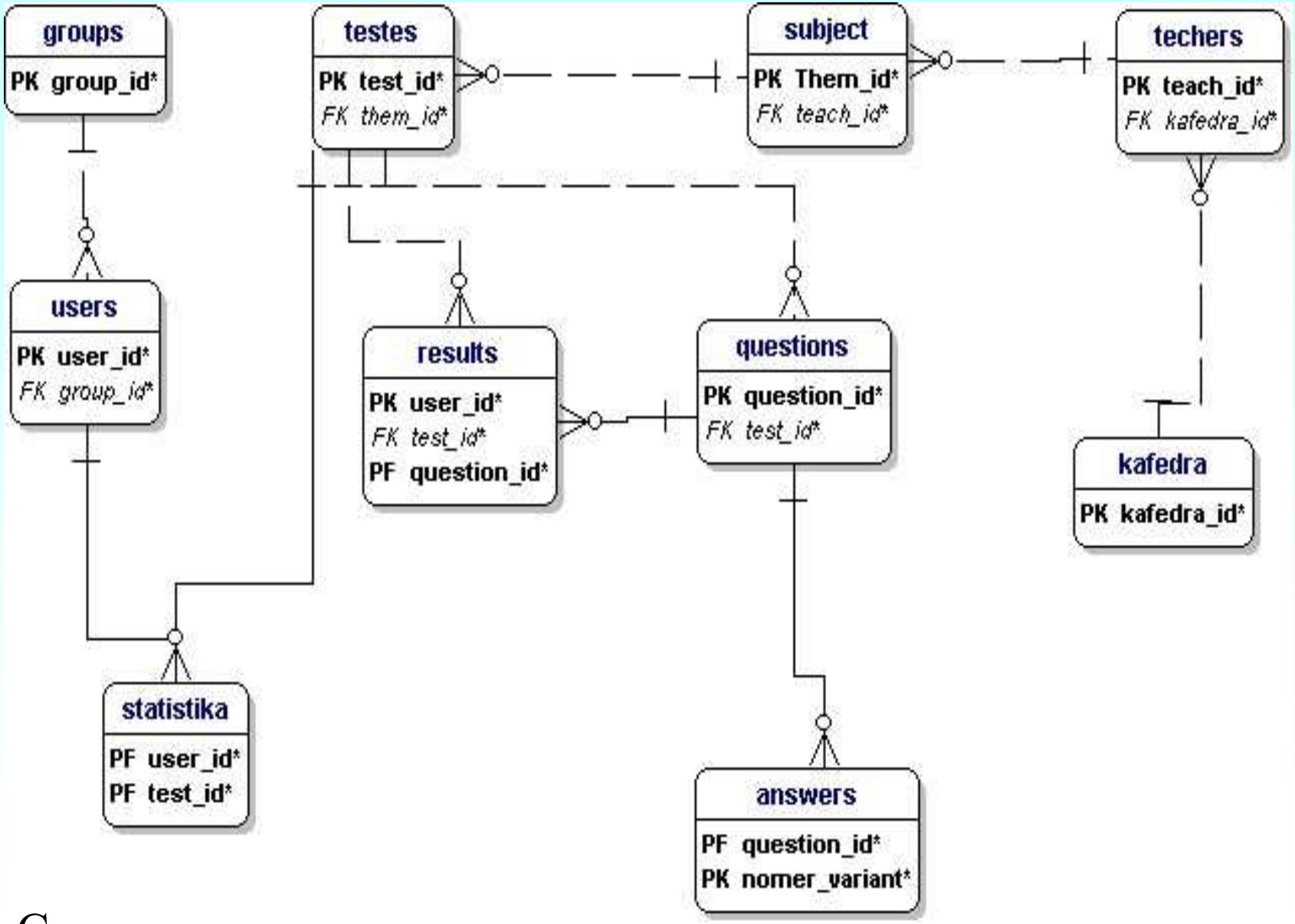
Подобные программные пакеты носят общее название **CASE**-средств.

CASE *(Computer Aided Software Engineering)*
– разработка программного обеспечения с помощью компьютера –
автоматизация разработки информационных систем.

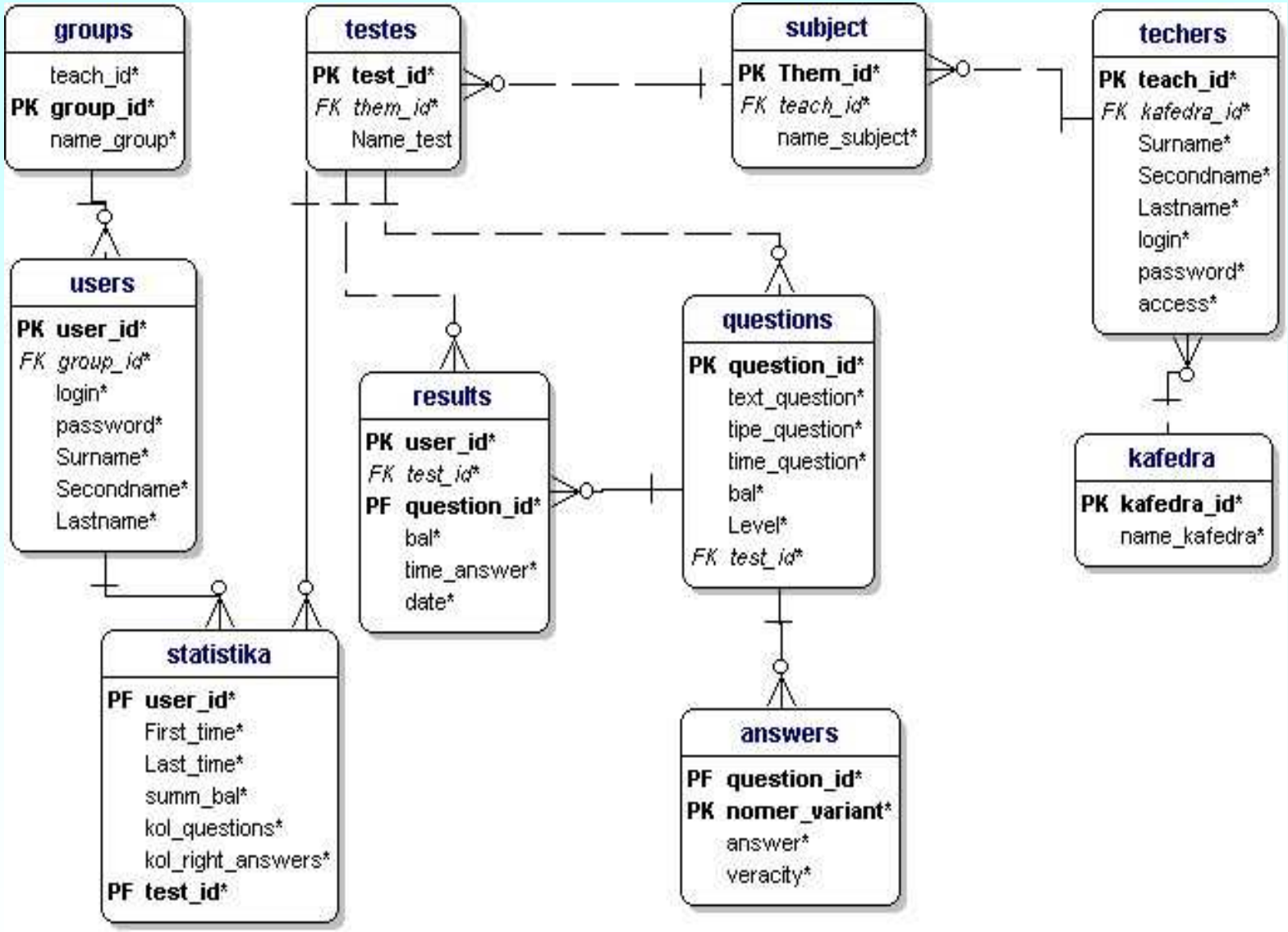
Пример БД для тестирования студентов
DeZign for Databases (version 3.2.1)



Сущности и связи



Сущности и ключи



Сущности, ключи и неключевые атрибуты

На представленных Er-диаграммах имеются два типа связей (отношений):

— Identifying relationship -выделяющие отношения, отношения, в которых первичный ключ родительской таблицы становится частью первичного ключа дочерней .

--- Non-identifying - не-выделяющие отношения, в которых первичный ключ родительской таблицы становится частью не-ключевой области дочерней.

На концах линий диаграммы используются четыре символа, определяющие тип отношения сущностей:

- | -один и только один (обязательное отношение);
- 0| - нуль или один;
- >| - один или более (обязательное отношение);
- >0 – нуль, один или более.

Концепции доступа к данным

СУБД часто поддерживают два способа доступа к данным: навигационный и реляционный.

Эти способы не противопоставляют, а чередуют, выбирая тот, который больше подходит для решения конкретной задачи.

Навигация – перемещение указателя по записям.

Навигационная концепция

***Навигационный доступ основывается на физическом или логическом порядке данных, т.е. последовательном перемещении указателя на запись*.**

Чтобы найти какую-либо запись в середине файла таблицы, указатель должен последовательно переместиться от начала (первой записи) к концу (последней записи) или наоборот.

При навигационном доступе к данным используют термины «файл», «запись», «поле».

С таким доступом к данным мы имели дело при изучении процедурного языка Visual Fox Pro

Реляционная концепция доступа к данным

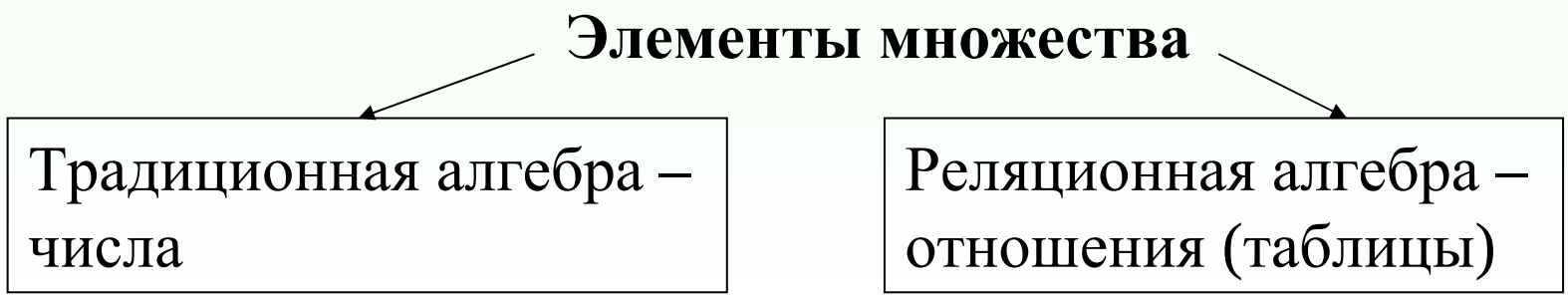
При реляционном доступе к данным используются термины «таблица» (файл), «строка» (запись), «столбец» (поле). Здесь операции выполняются сразу со всем набором данных, а не с его частью. Пользователю ничего не нужно знать о физическом расположении строк в таблице. При этом исключается необходимость сложной навигации по строкам. *Таблицы должны быть нормализованы*.

Доктор Кодд первоначально описал реляционную модель как область применения реляционной алгебры.

Алгебра - система определения множеств и операций над ними.

Множество - совокупность уникальных элементов, объединенных по какому-то признаку.

Оператор - символическая запись правила преобразования, выполняемого над одним или несколькими элементами множества.

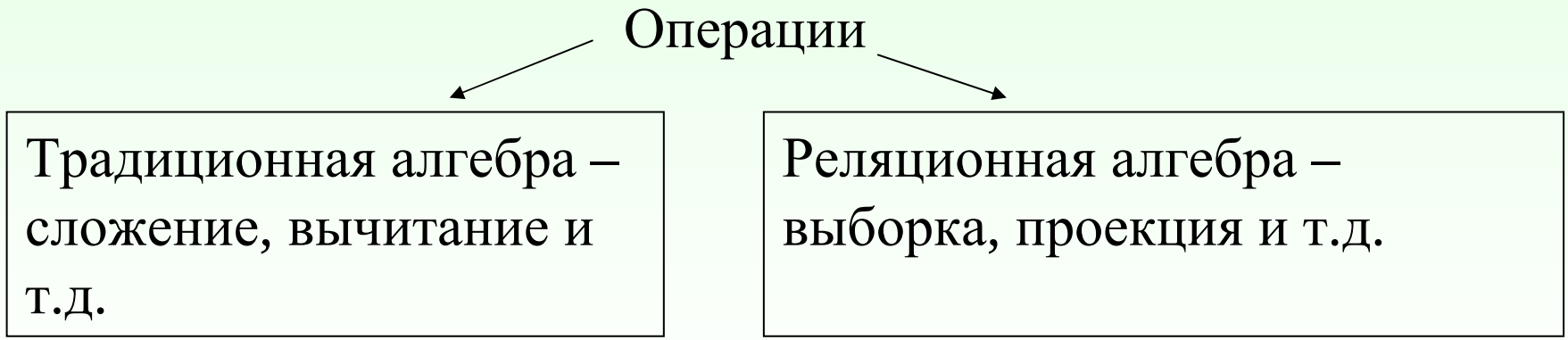


Таблицы называют отношениями, поскольку каждая строка таблицы неявно связана со всеми остальными строками, разделяя с ними общую форму записи.

Запись называют кортежем, т.е. набором взаимосвязанных атрибутов (столбцов).

Связи между записями реализуются в виде ключей.

Информация о связях хранится отдельно от данных.



Реляционные операции

Для реляционной модели представляют интерес только те операции, результатом которых является множество.

Есть восемь простейших операций: выборка, проекция, пересечение, сложение, вычитание, умножение, деление и переименование. На их основе строятся более сложные операции – объединения.

Результатом операции над таблицей будет временная таблица. Она не сохраняется в базе данных, но к ней можно обращаться в течении некоторого времени.

1. **Выборка – выполняется над одной таблицей. Результирующая таблица будет содержать все исходные столбцы, но не все строки – это фильтрация строк.**

Таблица Manager (сотрудники)

Код сотрудника	Фамилия	Дата рождения	Телефон	Должность
001	Иванов	01.01.1960	42-42-42	менеджер
002	Петров	01.01.1979	34-34-34	ст.менеджер
003	Иванов	01.01.1970	31-31-31	менеджер
004	Сидоров	02.02.1976	33-33-33	менеджер

Пусть нам требуется получить список менеджеров, не старше 30 лет (до 1975 г.р.).
результат операции «выборка»:

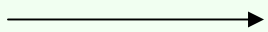
Код сотрудника	Фамилия	Дата рождения	Телефон	Должность
002	Петров	01.01.1979	34-34-34	ст.менеджер
003	Сидоров	02.02.1976	33-33-33	менеджер

2. **Проекция – возвращает все записи исходной таблицы, но не все столбцы. Это фильтрация столбцов.**

Пусть нам требуется получить фамилии менеджеров.

Код сотрудника	Фамилия	Дата рождения	Телефон	Должность
001	Иванов	01.01.1960	42-42-42	менеджер
002	Петров	01.01.1979	34-34-34	ст.менеджер
003	Иванов	01.01.1970	31-31-31	менеджер
004	Сидоров	02.02.1976	33-33-33	менеджер

Проекция по столбцу Фамилия



Фамилия
Иванов
Петров
Сидоров

В результирующей таблице повторяющиеся значения недопустимы.

3. **Пересечение - выполняется над двумя таблицами идентичной структуры. В результирующую таблицу помещают только те записи, которые встречаются в обеих исходных таблицах.**

Пусть наша фирма имеет два филиала. В БД обеих филиалов созданы таблицы **Manager.** Узнаем, **фамилии** менеджеров, сотрудничающих с обоими филиалами.

Фамилии менеджеров из 1 филиала
Иванов
Петров
Черных
Сидоров

∩

Фамилии менеджеров из 2 филиала
Иванов
Демин
Васин
Сидоров

=

Пересечение
Иванов
Сидоров

4. **Сложение - выполняется над двумя таблицами идентичной структуры. В результирующую таблицу помещают все записи исходных таблиц.**

Например, можно получить полный список менеджеров, работающих в филиалах:

Фамилии менеджеров из 1 филиала
Иванов
Петров
Черных
Сидоров

$\cup$

Фамилии менеджеров из 2 филиала
Иванов
Демин
Васин
Сидоров

=

Сложение
Иванов
Петров
Черных
Демин
Васин
Сидоров

5. **Вычитание – возвращает строки первой таблицы, отсутствующие во второй таблице.**
Определим фамилии менеджеров, сотрудничающих только с первым филиалом фирмы.

Фамилии менеджеров из 1 филиала
Иванов
Петров
Черных
Сидоров

-

Фамилии менеджеров из 2 филиала
Иванов
Демин
Васин
Сидоров

=

Вычитание
Петров
Черных

6. **Умножение – объединяет каждую строку первой таблицы с каждой строкой второй таблицы. Эту операцию еще называют декартовым произведением**.
Количество строк результирующей таблицы равно произведению числа строк исходных таблиц. Столбцами результирующей таблицы будут все столбцы первой таблицы, за которыми следуют все столбцы второй таблицы.
Менеджеры Иванов и Сидоров сотрудничают с двумя филиалами:

Код	**Фамилия**
001	Иванов
004	Сидоров

×

Код филиала	**Филиалы**
ф1	Ангарск
ф2	Братск

=

Код	**Фамилия**	**Код филиала**	**Филиалы**
001	Иванов	ф1	Ангарск
001	Иванов	ф2	Братск
004	Сидоров	ф1	Ангарск
004	Сидоров	ф2	Братск

7. Деление - выполняется над двумя таблицами, первая из которых состоит из двух столбцов, а вторая из одного. Значения второй таблицы сравниваются со значениями первого столбца первой таблицы, и если обнаруживаются совпадения, то соответствующие значения второго столбца первой таблицы включаются в результирующую таблицу. Результирующая таблица состоит из одного столбца.

Фамилия	Филиалы
Иванов	Ангарск
Иванов	Братск
Сидоров	Ангарск
Сидоров	Братск
Черных	Ангарск

/

Фамилия
Иванов
Черных

=

Филиалы
Ангарск
Ангарск

Определили филиалы, в которых работают менеджеры (дубликаты исключены)

8. **Переименование – назначает столбцу другое имя. Такие имена называются псевдонимами.**

Одним из классов более сложных операций являются объединения.
Объединение – частный случай произведения, при котором к таблице произведения применяются дополнительные критерии фильтрации (условия отбора).

Объединение может быть:
- **Внутренним.**
- **Внешним (левое, правое, полное).**

Результат объединения таблиц

Тип	В результирующую таблицу…
Inner join (Внутреннее объединение)	Выбираются только те записи, которые содержат совпадающие значения в поле связи
Left join (Объединение слева)	Выбираются все записи из левой таблицы, а из правой- , только те, в которых значение поля связи совпадают со значениями поля связи левой таблицы
Right join (Объединение справа)	Выбираются все записи из правой таблицы, а из левой- , только те, в которых значение поля связи совпадают со значениями поля связи правой таблицы
Full join (Полное объединение)	Выбираются все записи из правой и левой таблиц
Cartesian product (Декартово произведение)	В отсутствии критерия объединения, выбираются все записи из обеих таблиц, при этом каждая запись одной таблицы объединяется с каждой записью другой таблицы. Число результирующих записей равно числу записей в первой таблице, умноженному на число записей во второй таблице.

Стандартным языком реляционного доступа является SQL.
В командах языка SQL формулируется критерий обработки и способы преобразования информации. Это декларативный, непроцедурный язык.
SQL сам открывает нужные таблицы, открывает и создает нужные индексные файлы и определяет наиболее эффективные последовательности операций для получения результата.
Язык SQL использует небольшое количество команд. Каждая из них функционально эквивалентна нескольким обычным командам.

Практически все производители СУБД поддерживают стандарты языка SQL (Structured Query Language) - языка структурированных запросов. ***В составе SQL могут быть выделены следующие группы инструкций***:

- *язык описания данных DDL - создание таблиц (изменение, удаление);*
- *язык манипулирования данных DML - запросы (добавление, изменение, удаление);*
- *язык управления транзакциями.*

Транзакция - логически завершенная единица работы, создавшая одну или более элементарных операций обработки данных. Все действия должны выполняться полностью, либо полностью не выполняться.

Упрощенный синтаксис некоторых команд языка SQL

Команды языка определения данных (**DDL**)
Создание таблицы

CREATE TABLE имя таблицы [**FREE**]
(Имя поля1 тип данных [(ширина [, число десятичных знаков])]
[**NULL** | **NOT NULL**] {Разрешает|запрещает иметь в поле значения NULL}
[**DEFAULT выражение1**] {Задает значение, принимаемое в поле по умолчанию}
[**PRIMARY KEY** | **UNIQUE**] {Создает для поля первичный индекс}
[**REFERENCES** имя таблицы2 [**TAG** имя индекса]]
{Задает родительскую таблицу, с которой устанавливается постоянное отношение. Если опустить аргумент TAG **имя индекса**, отношение устанавливается на основе ключа первичного индекса родительской таблицы}

Продолжение табл.

[, Имя поля2 ...]
[, **PRIMARY KEY** Выражение2 **TAG** имя индекса]
(Задает первичный индекс, где Выражение2 есть произвольное поле или комбинация полей таблицы. Имя тега индекса может включать до 10 символов. Нельзя включать более одного предложения **PRIMARY KEY**.)
[, **FOREING KEY** Выражение3 **TAG** имя индекса3[**NODUP**]
(Создает внешний (отличный от первичного) индекс и устанавливает отношение с родительской таблицей. Выражение3 задает выражение ключа внешнего индекса. NODUP -индекс-кандидат)
REFERENCES Имя таблицы3 [**TAG** имя индекса4]])
(Задает родительскую таблицу, с которой устанавливается постоянное отношени)

Значения, используемые в качестве **FieldType** (тип данных), и указывается, необходимо ли при этом задавать **nFieldWidth** (ширину) и **nPrecision** (число десятичных знаков)

FieldType	nFieldWidth	nPrecision	Описание типа данных
C	n	-	Символьное поле ширины n
D	-	-	Дата
T	-	-	Тип DateTime
N	n	d	Числовое поле ширины n, содержащее d десятичных знаков
F	n	d	Плавающее числовое поле ширины n, содержащее d десятичных знаков
I	-	-	Целочисленное
B	-	d	Двойной
Y	-	-	Тип Currency
L	-	-	Логический
M	-	-	Memo
G	-	-	General

2. Создать таблицу Characteristics с полями :

C_id	FirstName	SecondName	LastName
03	Рожкова	Ольга	Ивановна

Пример:

```
CREATE TABLE Characteristics (C_id C(3), ;
FirstName C(15), SecondName C(10), LastName C(10),;
FOREIGN KEY C_id TAG C_id REFERENCES Customer TAG C_id)
```

Имя поля

Тип данных

Ширина

Внешний ключ

Связь с родительской таблицей

Символ «;» обозначает продолжение команды на следующей строчке.

В окне команд Visual FoxPro команды языка SQL, состоящие из нескольких строк, выполняются как единое целое. В процессе их набора для перехода на следующую строку используются клавиши управления курсором. Чтобы выполнить команду, ее выделяют с помощью курсора мыши, затем нажимают клавишу Enter.

Модификация структуры таблицы

```
ALTER TABLE имя таблицы
ADD | ALTER имя поля1
        тип поля [(ширина [, число десятичных знаков])]
        [NULL | NOT NULL]
        [DEFAULT выражение1]
        [PRIMARY KEY | UNIQUE]
        [REFERENCES имя таблицы2 [TAG имя индекса]]
```

Пример:

В таблицах Customer и Characteristics изменить тип индекса C_id на **Primary:**

ALTER TABLE customer ADD PRIMARY KEY C_id ;
TAG C_id
ALTER TABLE characteristics ADD PRIMARY KEY ;
C_id TAG C_id

Команды языка управления данными (DML)
Заполнение таблицы данными
INSERT INTO имя таблицы (список полей, разделенных запятой) **VALUES** (список значений, разделенных запятой)

Пример:

Добавить 2 различные записи в таблицу Characteristics (таблица должна быть открыта – Data Session).

Список полей

Соответствующие значения

INSERT INTO characteristics(C_id, firstname,secondname, lastname);
VALUES ("01","Матюшин","Виктор","Петрович")

INSERT INTO characteristics(C_id, firstname,secondname, lastname);
VALUES ("02",”Разгильдяев", “Данил","Петрович")

Изменение данных в полях таблицы
UPDATE [имя б.д.!]имя таблицы; (имя б.д., отличное от текущей) **SET** имя поля1 = выражение1; [,имя поля2 = выражение2 ...]; **WHERE** условие фильтра1 [**AND** | **OR** условие фильтра 2 ...]]

Пример:

Изменить в таблице Customer название компании Гермес на Гермес-Союз.

UPDATE Customer;
SET C_Name= «Гермес-Союз»;
WHERE ALLTRIM(C_Name)= «Гермес»

Удаление строк из таблицы
DELETE FROM [имя б.д.!]имя таблицы; {имя б.д., отличное от текущей} [WHERE условие фильтра1 [AND | OR условие фильтра 2..]]

Пример:

Удалить из таблицы Characteristics запись о клиенте Матюшине.

DELETE FROM Characteristics;

WHERE ALLTRIM(FirstName)= «Матюшин»

Пометка на удаление !!!

Выборка данных из одной таблицы
SELECT имя поля FROM имя таблицы

Пример:

Просмотреть значения всех идентификаторов клиентов (аналог использования цикла Scan).

SELECT C_id **FROM** Customer

Окончание табл. 3.2

Выборка данных из нескольких таблиц
SELECT [имя таблицы1.] имя поля1 [**AS** новое имя1] [, [имя таблицы2.] имя поля2 [**AS** новое имя2] ...] **FROM** [имя базы данных!]имя таблицы1 Тип объединения(**INNER JOIN**) [, [имя базы данных!] имя таблицы2] ...] **ON** имя таблицы1. имя поля= имя таблицы2.имя поля [[**INTO** назначение] | [**TO FILE** имя файла [**ADDITIVE**]	**TO PRINTER** [**PROMPT**]	**TO SCREEN**]] [**WHERE** условие объединения1 [**AND** условие объединения2...] [**AND**	**OR** условие отбора1 [**AND**	**OR** условие отбора2 ...]]] [**GROUP BY** имя поля3[, имя поля4 ...]] [**HAVING** условие отбора3] [**ORDER BY** имя поля5 [**ASC**	**DESC**] [, имя поля6 [**ASC**	**DESC**] ...]]

Пример:

Отобрать данные из таблиц Customer и Order и отсортировать их по полю Customer.C_name .

SELECT Customer.C_id, Customer.C_name, Order.Or_id;
FROM имя БД!Customer **INNER JOIN** имя БД!Order ;
ON Customer.C_id= Order.C_id;
ORDER BY Customer.C_name

Доступ к удаленным данным

1. ***Интерфейс ODBC***
2. ***Термин - источник данных***
3. ***Типы источников данных***
4. ***Создание источника данных***
5. ***Удаленный доступ к данным в среде Visual FoxPro***
6. ***Использование функций SQL Path Through***

Интерфейс ODBC

Удаленные (внешние) данные - данные, хранящиеся в формате других приложений, например, серверов БД (Oracle, MS SQL Server).

В среде Visual FoxPro данные Access будут считаться удаленными, и наоборот.

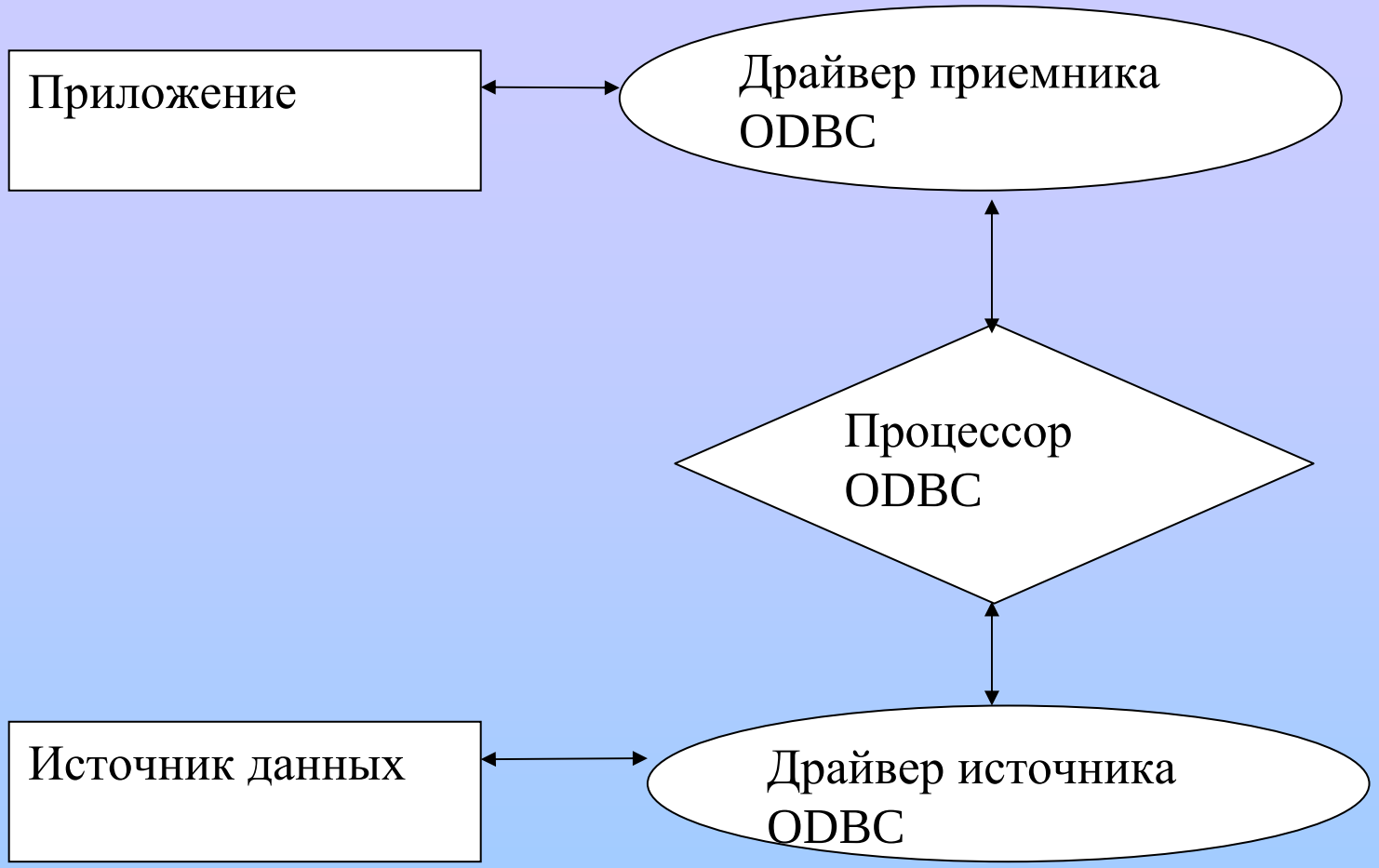
Локальные данные - данные хранящиеся в формате обращающегося к ним приложения.

Для доступа из одной СУБД к данным различных внешних источников предназначен интерфейс стандарта ODBC (Open DataBase Connectivity – Открытый доступ к данным). Через этот интерфейс приложения Microsoft могут получать доступ к различным БД, использующим SQL.

ODBC – это технология, встроенная в среду WINDOWS.

В ее основе лежит использование ODBC-процессора WINDOWS.

В ее функционировании можно выделить 3 уровня: ODBC-процессор WINDOWS, драйвер ODBC для приемника данных и драйвер ODBC для источника данных.



Блок-схема ODBC-процесса

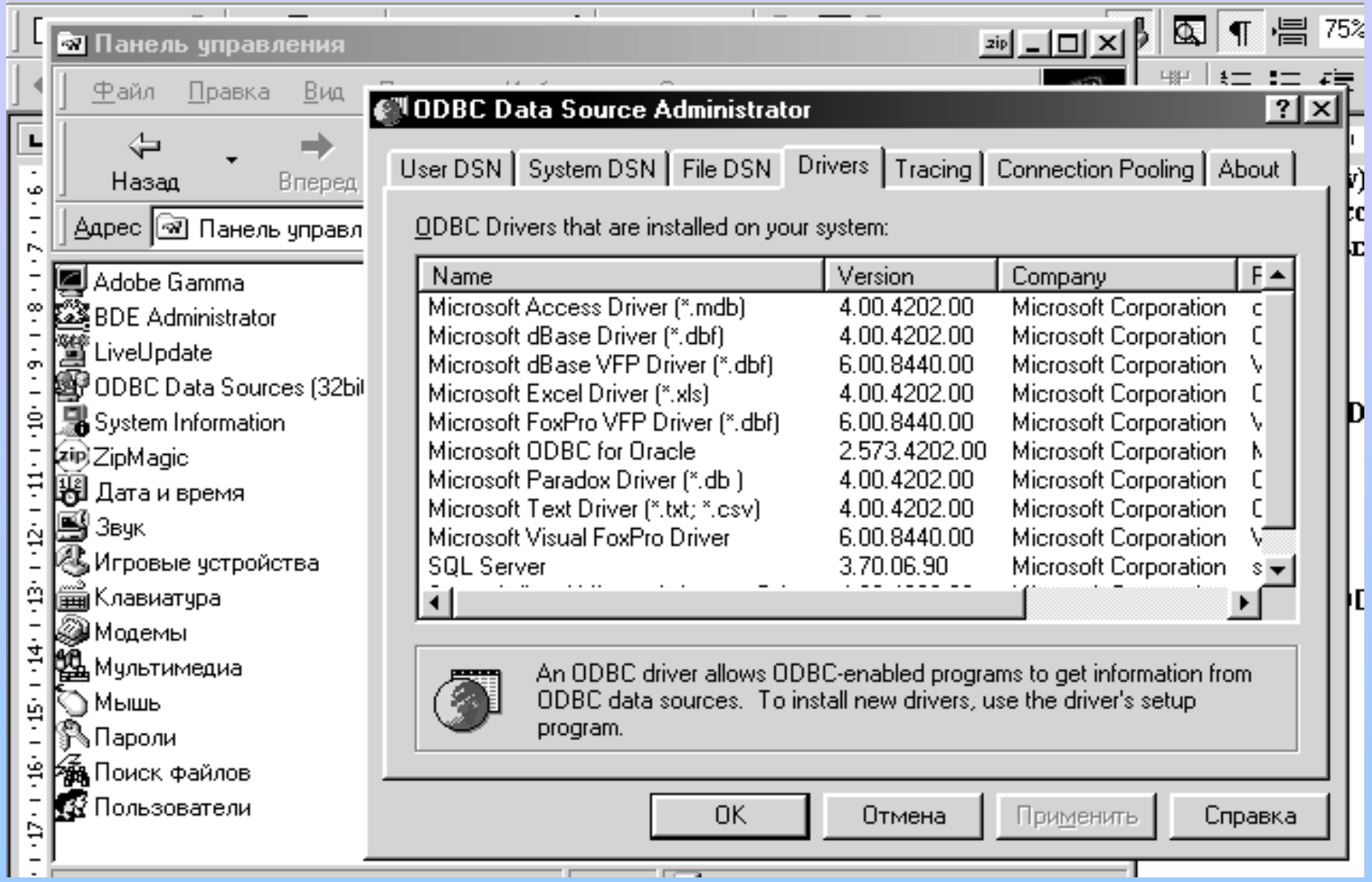
Термин - источник данных

Источник данных ODBC - это термин, используемый для ссылки на внешнюю БД. Он включает базу данных и информацию, необходимую для доступа к этой базе данных.
К источникам данных обращаются по именам.

Чтобы обратиться к любым ODBC-источникам данных, необходимо установить драйвер источника данных.

Для установки ODBC-драйвера используется 32-разрядный администратор ODBC (ODBC Data Sources) - Панель управления.

Установленные драйверы находятся в подкаталоге System в директории WINDOWS. Файлы драйверов ODBC имеют расширение .DLL, а их названия начинаются с ODBC. Просмотреть установленные в ОС драйверы можно на вкладке Drivers в окне Администратора ODBC (ODBC Data Sources Administrator) Панели управления (рис.).



Окно Администратора ODBC

Типы источников данных

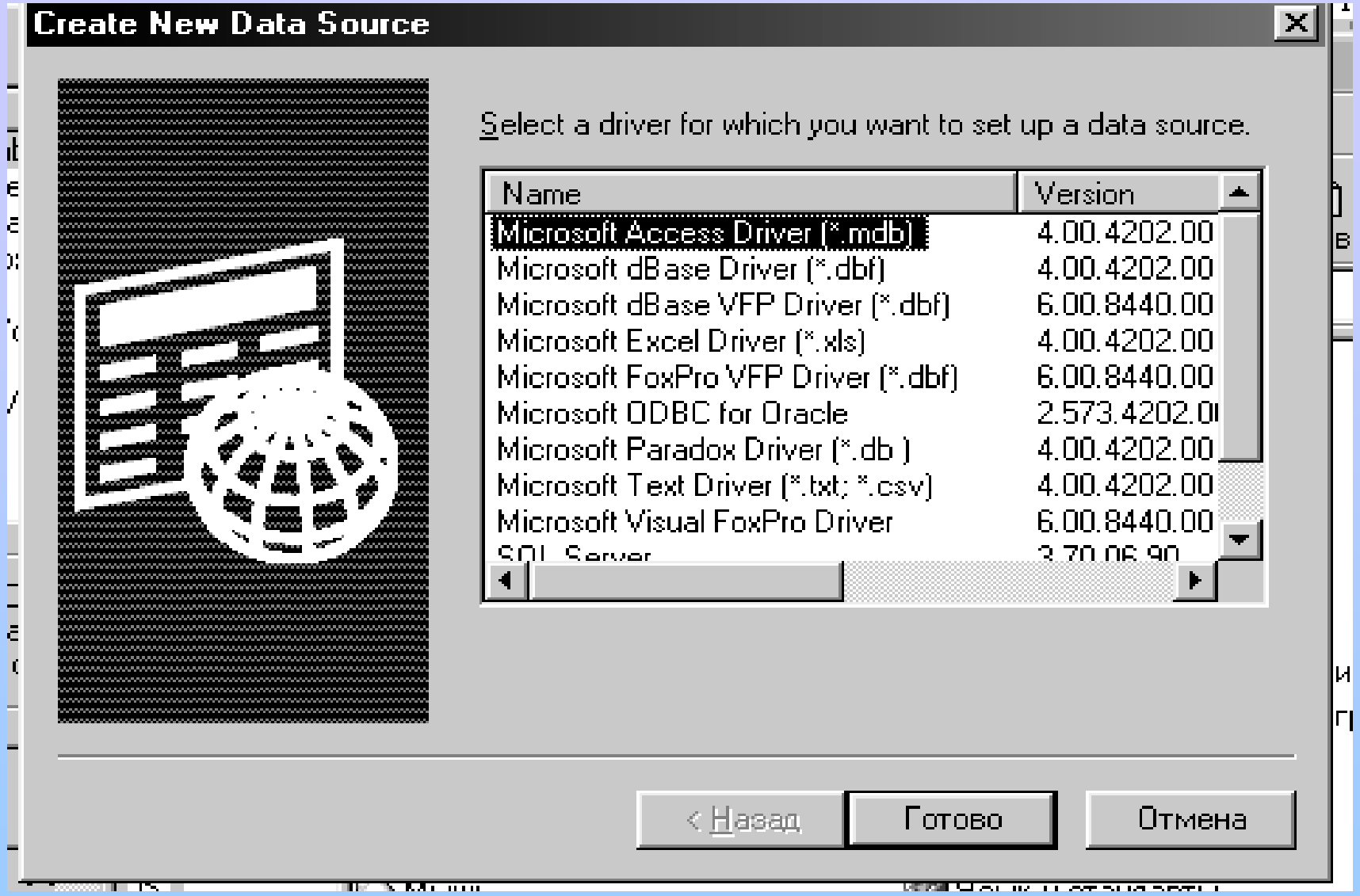
Три типа источников данных (см. соответствующие вкладки окна **ODBC Data Sources**):

- *пользовательский источник* данных **User DSN**. Применяться только одним пользователем, работающим на данном компьютере;
- *системный источник* данных **System DSN**. Предназначен для всех пользователей и системных служб на данном компьютере;
- *файловый источник* данных **File DSN**. Обеспечивают доступ к БД многих пользователей разных компьютеров сети. Описание источника сохраняется в файле с расширением .dsn. Этот файл должен быть доступен для всех компьютеров сети.

Создание источника данных

Рассмотрим пример создания пользовательского источника данных.

1. ***Проверить, установлен ли драйвер требуемого источника данных***, например, Microsoft Access Driver или Microsoft Visual FoxPro Driver. (Панель управления - 32-разрядный администратор ODBC (**ODBC Data Sources**) - вкладке **Drivers**).
2. ***В окне ODBC Data Sources перейти на вкладку User DSN (пользовательский DSN).*** Первоначально на ней нет источников данных для Microsoft Access и Microsoft Visual FoxPro.
3. ***Нажать кнопку Add (Добавить).*** В окне **Create New Data Source** (Создание нового источника данных) ***выбрать драйвер,*** для которого создается источник, например, Microsoft Access Driver или **Microsoft Visual FoxPro Driver** (рис.) и нажать кнопку **Готово**.

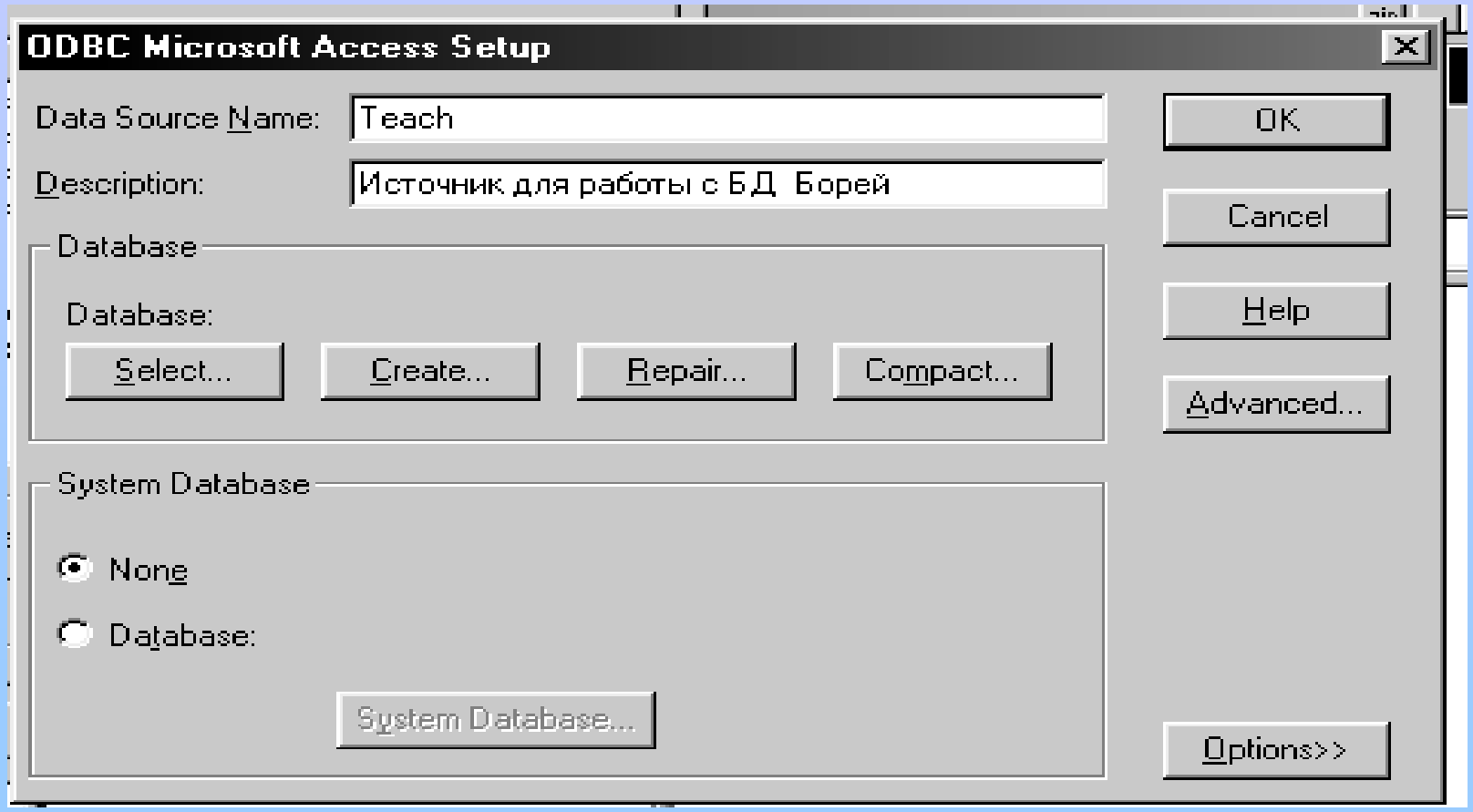


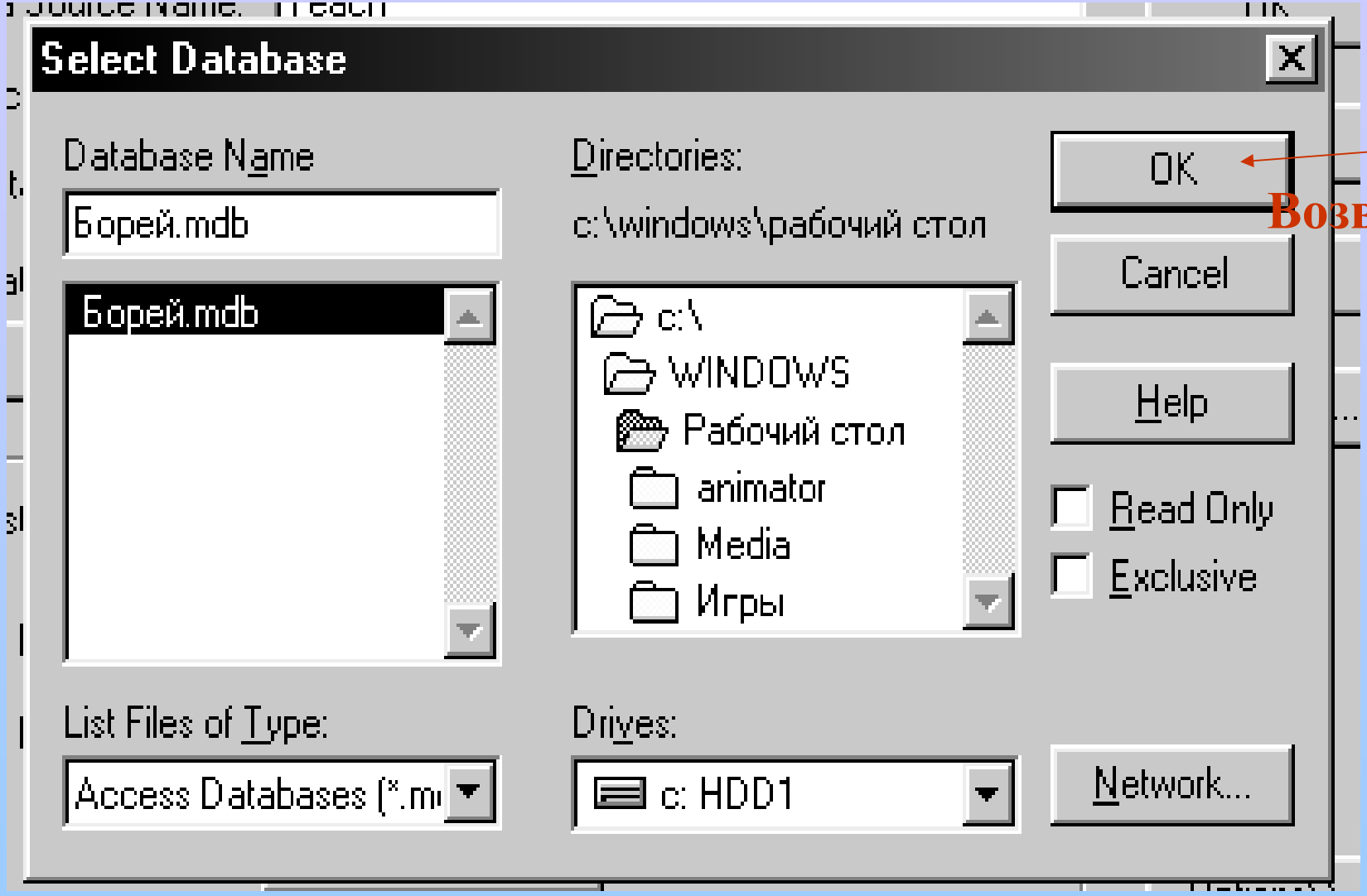
Выбор драйвера источника данных

К работе подключается мастер создания нового источника данных.

4. В окне мастера следует ввести:

- **Data Source Name** - имя источника данных;
- **Description** (описание) комментарий;
- Кнопка **Select** - выбрать БД по умолчанию.





Окно выбора базы данных

➢ *Кнопка* **Advanced** *- выбор режима регистрации при подключении к источнику данных. По умолчанию - регистрация по доверительному соединению*, при котором пользователь, зарегистрировавшийся в сети, не проверяется дополнительно. Можно задать идентификатор пользователя (**Login Name**) и пароль (**Password**), которые будут проверяться самим приложением Microsoft Access.

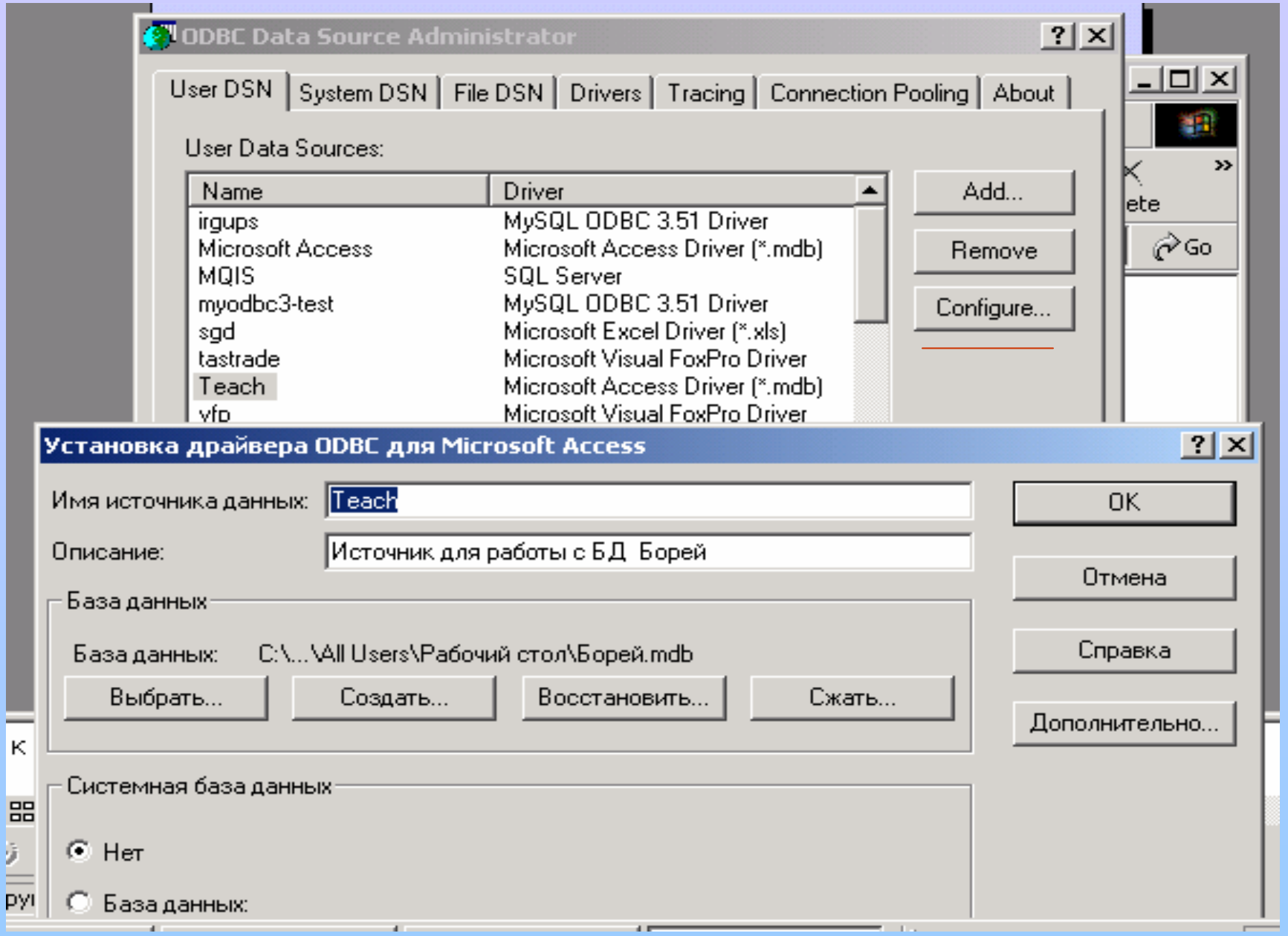
После завершения ввода в окне мастера создания нового источника данных следует нажать кнопку **ОК (сл.10)**.

Просмотреть параметры созданного источника данных:

-Выбрать его имя в списке на вкладке **User DSN** окна администратора ODBC (**ODBC Data Sources**) ;

-Воспользоваться кнопкой **Configure** (настройка);

-Выполнить необходимые действия.


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
User Data Sources:
Name
Driver
irgups
MySQL ODBC 3.51 Driver
Microsoft Access
Microsoft Access Driver (*.mdb)
MQIS
SQL Server
myodbc3-test
MySQL ODBC 3.51 Driver
sgd
Microsoft Excel Driver (*.xls)
tastrade
Microsoft Visual FoxPro Driver
Teach
Microsoft Access Driver (*.mdb)
vfp
Microsoft Visual FoxPro Driver
Add...
Remove
Configure...
Установка драйвера ODBC для Microsoft Access
Имя источника данных:
Teach
Описание:
Источник для работы с БД Борей
OK
Отмена
Справка
Дополнительно...
База данных
База данных:
C:\...\All Users\Рабочий стол\Борей.mdb
Выбрать...
Создать...
Восстановить...
Сжать...
Системная база данных
Нет
База данных:
Go

Удаленный доступ к данным в среде Visual FoxPro

Доступ к удаленным данным реализуется с помощью удаленных представлений ***Remote View*** *или сквозных команд SQL* (**Pass through**).

Создание удаленных представлений Microsoft Visual FoxPro

Представление - это вид запроса, используемый в среде Visual FoxPro.
Представления могут оперировать удаленными, Remote View, и локальными, Local View, данными.
Представления являются прекрасным способом доступа и редактирования данных, сохраняемых в нескольких связанных файлах.

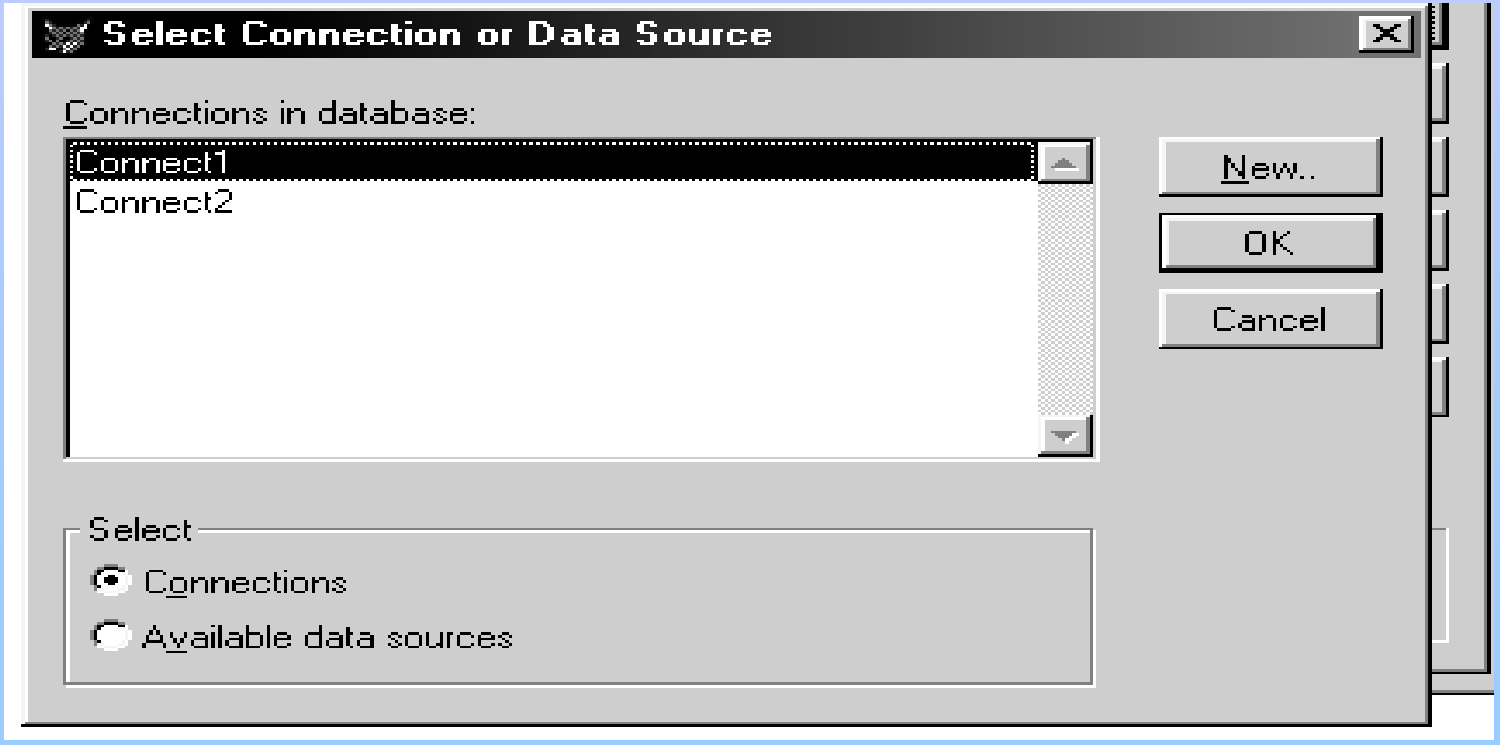
Принципиальные отличия между представлениями и запросами заключаются в следующем:

- *представления можно использовать для обновления данных в исходных таблицах;*
- *представления хранятся в самой БД, а не в отдельных .Qpr- файлах, поэтому доступ к ним мы получаем, только при открытой БД;*
- *возможность создания параметризованных представлений.*

В окне диспетчера проекта представления **Remote View** и **Local View** находятся на вкладке **Data** (данные).

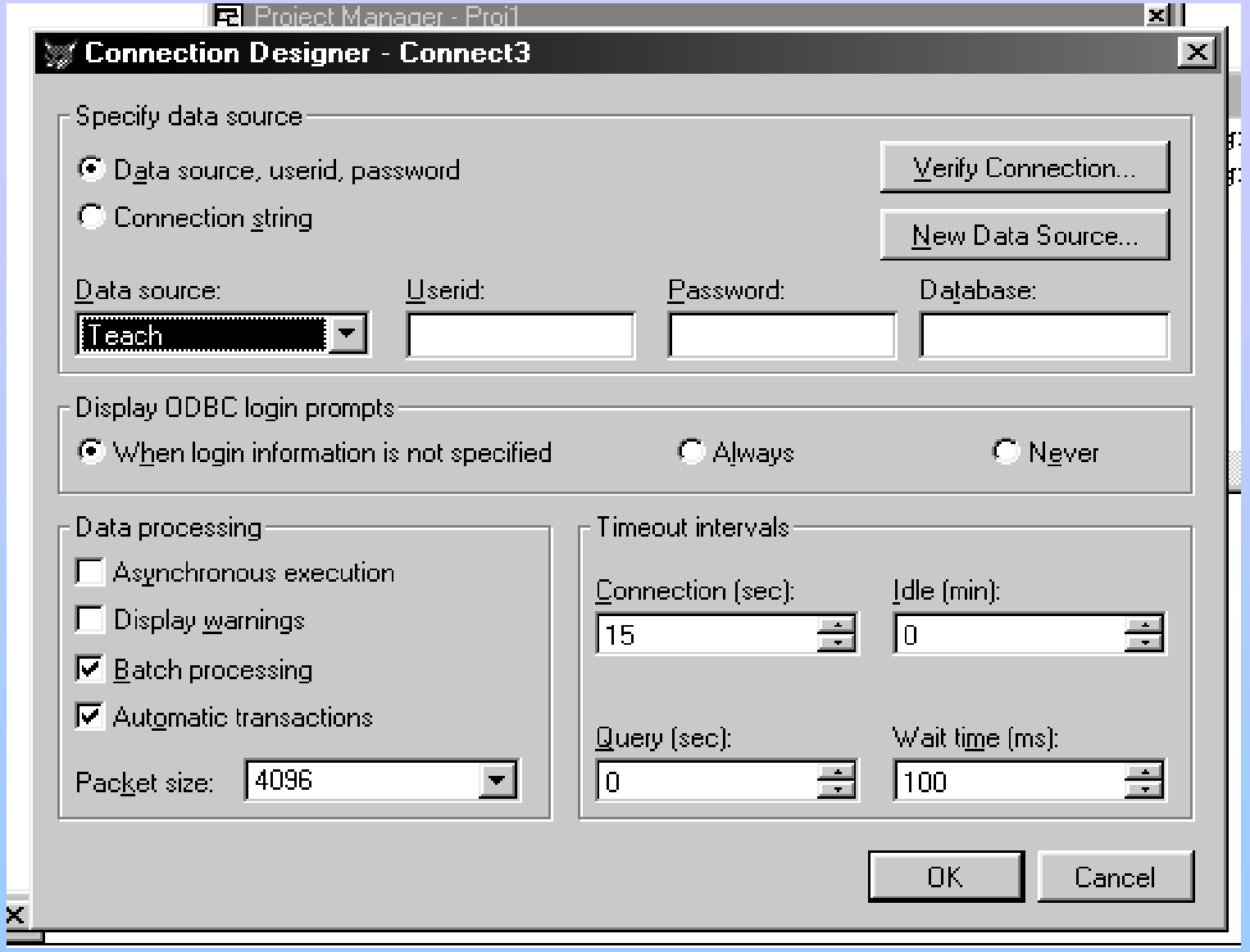
Представление Remote View можно создать на основе соединения Connections или источника данных Data Sources.

При нажатии кнопки **New** (новое) на вкладке **Data** (данные) открывается диалоговое окно **Select Connection or Data Source,** в котором с помощью группы переключателей **Select** можно выбрать один из вариантов: **Connections** или **Data Sources.**



Если выбран переключатель* Data Sources**, *то в* ***окне* Available Data Sources** *можно* ***выбрать источник данных, *например*, **Teach**, *обеспечивающий доступ к заданной удаленной БД – Борей.mdb.*

Для того чтобы для каждого удаленного представления не создавать отдельный источник данных, в Visual FoxPro предусмотрен механизм создания совместно используемых соединений.
Соединения также находятся на вкладке **Data** *окна диспетчера проекта и входят в состав БД.*
Соединение можно создать как обычный объект Visual FoxPro.
В окне **Select Connection or Data Source (см.18)** *при установке переключателя в положение* **Connections** *можно выбрать существующее соединение или создать новое.*
При нажатии кнопки **New** *открывается окно* **Connection Designer.**



Окно конструктора соединений

Группа переключателей ***Specify data source*** *(определение источника данных) задает метод определения соединения.*
Установка переключателя **Connection string** требует ввода строки соединения, которую необходимо отправить серверу БД (см.20).
Если установить переключатель в положение **Data source, userid, password**, как показано на рис, нужно заполнить соответствующие поля ввода:

- **Data Source -** имя источника данных ODBC (выбрать из списка);
- **Userid –** идентификатор пользователя;
- **Password –** пароль;
- **DataBase –** имя базы данных.

Поля ввода **Userid** и **Password** заполняются, если предусмотрена парольная защита источника данных.

Группа переключателей **Display ODBC login prompts** задает, когда следует отображать приглашения ODBC на регистрацию пользователя.

Здесь возможны три варианта:

- когда пользователь не определен;
- всегда;
- никогда.

Опции раздела **Data Processing** (обработка данных) позволяют:

- **Asynchronous Execution –** выбирать данные в фоновом режиме;
- **Display Warnings –** отображать предупреждения;
- **Patch Processing –** пакетную (одновременную) обработку данных;
- **Automatic Transactions –** выполнять транзакции при модификации таблиц;

Packet Size – определять размер пакета пересылаемых данных.

Раздел **Timeout Intervals** (интервалы времени ожидания) задает следующие параметры:

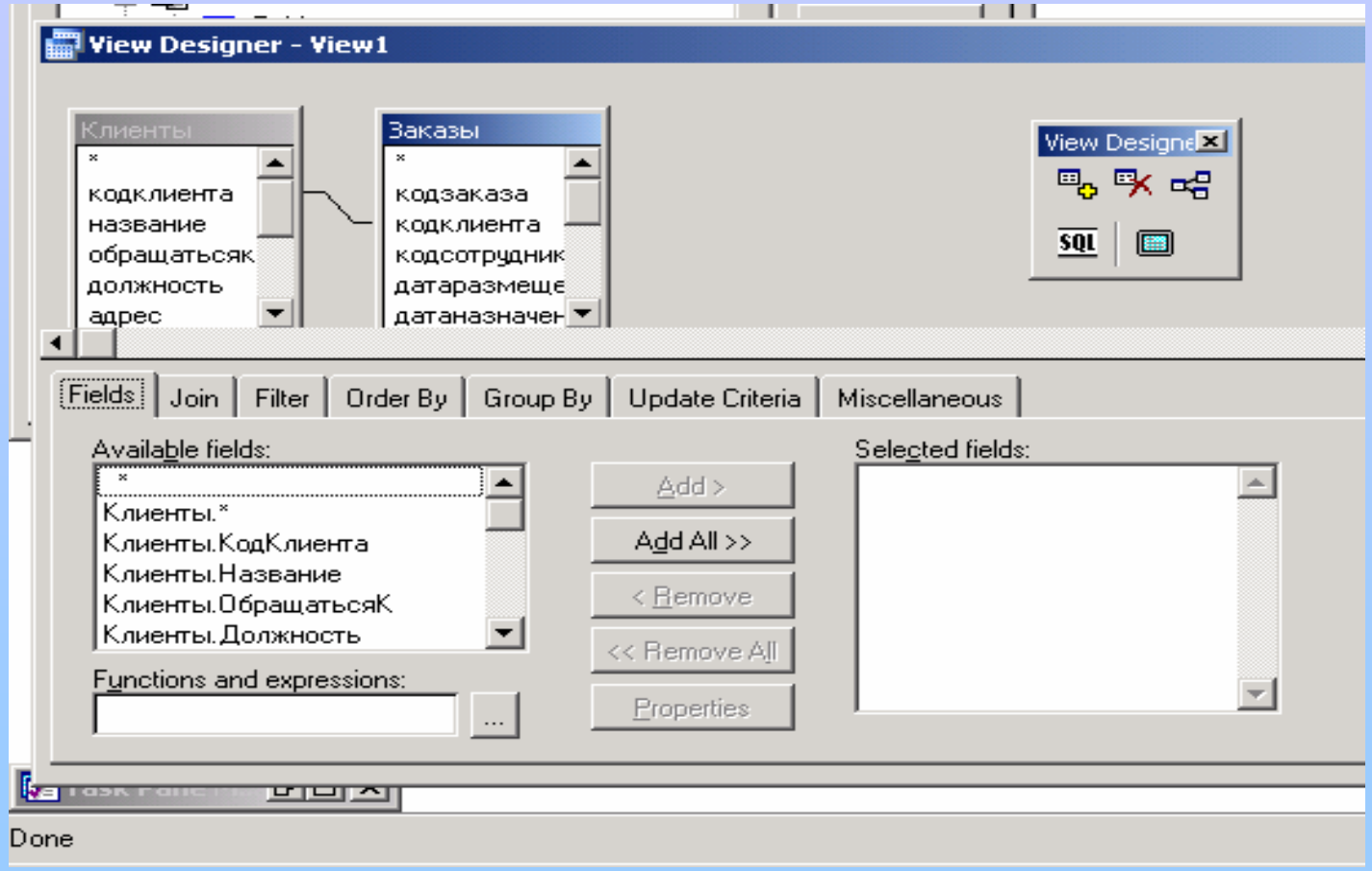
- **Connection** (sec) - время в сек, по истечении которого Visual FoxPro сообщит об ошибке времени ожидания соединения;
- **Idle** (min) – время простоя, по истечении которого Visual FoxPro разорвет соединение;
- **Query** (sec) – время, по истечении которого формируется общая ошибка времени ожидания;
- **Wait Time** (ms) – время, выделенное для ожидания выполнения инструкций SQL.

Кнопка **New Data Source** позволяет создать новый источник данных для устанавливаемого соединения.
*Кнопка **Verify Connection** предоставляет возможность проверить устанавливаемое соединение, если все параметры заданы верно, Visual FoxPro сгенерирует соответствующее сообщение:*

После настройки параметров соединения, ему присваивают имя и сохраняют.

Описав способ доступа к удаленным данным перейдем к их использованию.

Окно конструктора **Remote View** подобно окну конструктора запросов.



Необходимые таблицы *добавляются* в окно конструктора представления.

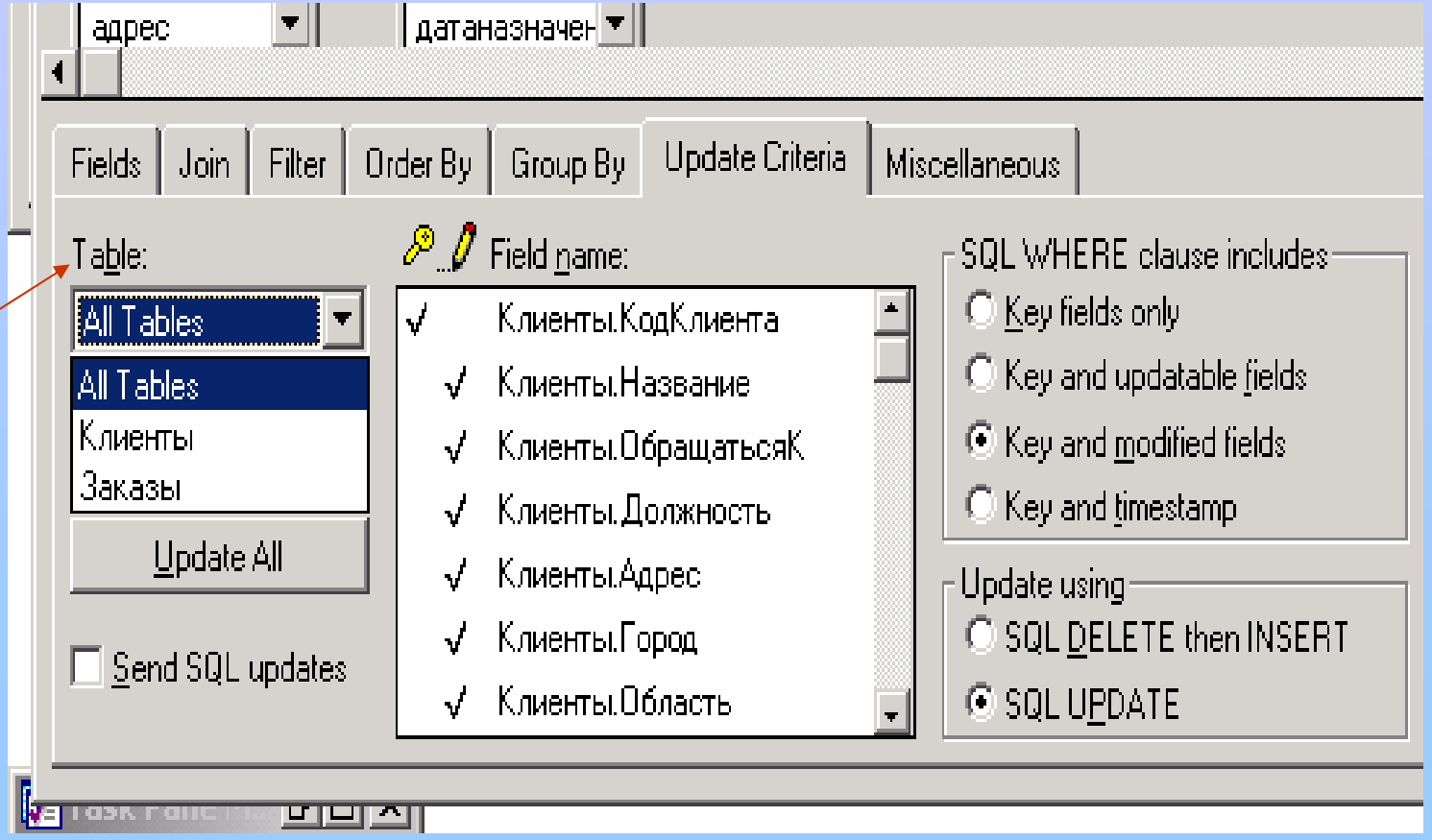
Между таблицами устанавливаются временные связи по значениям совпадающих полей с помощью окна диалога Join Condition.

Представления формируются не из таблиц Visual FoxPro, поэтому установление постоянных отношений не возможно. В VFP6.0 Сформированное в окне ***Join Condition*** *условие отбора записей таблиц будет помещено на вкладку* ***Filter*** *(фильтр), в VFP9.0 на вкладку* ***Join***.

Вкладка Fields – выбор полей результирующей таблицы.
Замечание ***Поле связи таблиц в списке выбранных полей должно встречаться один раз***.
При выборе полей из списка ***Available fields*** *второй таблицы на вкладке* ***Fields*** *следует отказаться от выбора поля связи.*
Например, для таблиц Клиенты и Заказы поле Код клиента следует выбирать только из таблицы Клиенты, где оно является первичным ключом.

Окно конструктора представлений имеет вкладку Update Criteria - критерий обновления данных.

По умолчанию Visual FoxPro запрещает обновление всех полей в представлении. Чтобы разрешить обновление каких-то полей из одной таблицы, необходимо выбрать эту таблицу с помощью комбинированного списка Table.



В число выбранных полей обязательно должны входить поля первичного или потенциального ключа таблиц.

Эти поля отмечаются щелчком мыши в первом столбце, имеющем символ ключа.

Столбец в списке ***FieldName****, помеченный символом карандаша, позволяет выбрать поля, в которые можно вносить изменения.*

Кнопка ***Update All*** *(обновить все) позволяет выбрать все поля, кроме ключевых.*

Для обновления ключевых полей предварительно необходимо определить в исходной БД правила ссылочной целостности данных.

SQL не выполняет обновление в исходных таблицах, если не будет установлен флажок опции Send SQL Updates (Отправлять обновления).

Группа опций **SQL Where clause includes** предназначена для задания условий фильтрации обновляемых записей (сл.29) Предложение **Where** инструкции SQL задает, что в поиске записей, которые следует обновить, используются:

- **Key fields only** –только ключевые поля (наиболее мягкий вариант);
- **Key and update fields** – ключевые и все обновляемые поля, даже, если они не были изменены;
- **Key and modified fields** – ключи и модифицированные поля;
- **Key and timestamp** – ключи и метка времени (обновления не принимаются, если были изменены ключи или метка времени).

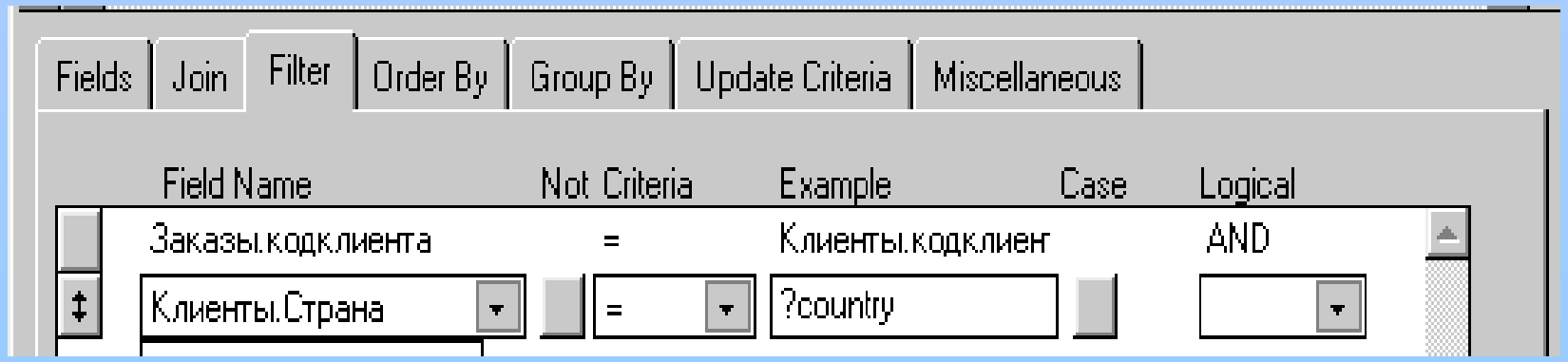
Опции **Update Using** выбирают метод обновления исходных данных. SQL может сначала удалить исходные записи, затем вставить новые (**SQL Delete then insert**), либо просто обновит существующие (**SQL Update**) (сл.29).

Отметим, что некоторые серверы БД выполняют первые операции быстрее.

Как и при создании запроса, при заполнении вкладок окна конструктора представлений формируется инструкция SQL **SELECT**. *Выполнить эту инструкцию* можно из пункта Главного меню **Query**, команда **Run Query.** После сохранения представления его можно выполнить с помощью кнопки **Browse**.

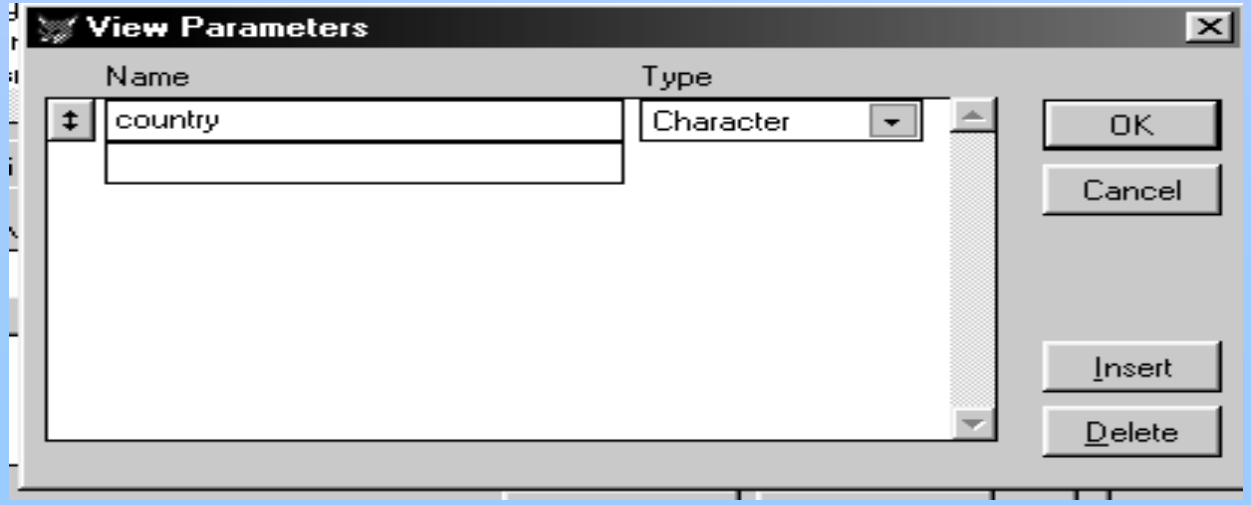
Представления, в отличие от запросов, не только позволяют обновлять данные таблиц, но и сами являются обновляемыми, т.е. каждый раз выполняются заново с использованием последней версии данных.

Существует возможность во время выполнения представления задавать разные критерии отбора данных, например, сегодня мы хотим просмотреть заказы клиентов из Германии, завтра – из России и т.д.
Для обеспечения такой возможности на вкладке ***Filter*** *(фильтр) в поле* ***Example*** *задаем не наименование* страны, а специальный ***параметр представления, имя которого начинается со знака вопроса, например, ?country***



Чтобы создать не один параметр, а несколько, можно воспользоваться командой **View Parameters** пункта Главного меню **Query**.

В окне диалога **View Parameters** следует определить имя параметра и тип вводимых в него данных. После этого созданные параметры, как переменные, можно использовать в поле **Example** на вкладке **Filter** (фильтр) при формировании критерия отбора данных.



При выполнении представления последует запрос на ввод значения каждого из созданных параметров.

При следующем запуске представления можно ввести другое значение параметра country и, например, просмотреть заказы клиентов из России.

Результатом работы представления, как и запроса, является таблица, которая остается открытой до поступления специальной команды.

В связи с чем перед повторным запуском представления, чтобы увидеть изменения, внесенные в данные, необходимо закрыть его предыдущую версию.

Для этого можно использовать, например, окно сеанса данных **Data Session Window**, а в нем кнопку **Close** (закрыть).

Использование функций SQL Path Through

В Visual FoxPro понятие SQL Path Through означает набор функций, позволяющих посылать команды непосредственно через ODBC. Этот способ доступа к удаленным данным имеет как свои преимущества, так и свои недостатки.

В зависимости от решаемой задачи следует отдавать предпочтение либо функциям **SQL Path Through,** либо созданию удаленных представлений. Отметим, что оба способа сопоставимы по быстродействию.

Преимущества использования функций **SQL Path Through:**

- функции **SQL Path Through** могут передать источнику данных любую корректную команду, тогда как удаленное представление -только одну команду **SQL SELECT**;
- функции **SQL Path Through** могут возвращать несколько результирующих наборов данных, а удаленное представление – один;
- в отличие от удаленных представлений, функции **SQL Path Through** обеспечивают доступ к средствам обработки транзакций источника данных.

К *недостаткам* функции **SQL Path Through** можно отнести:

- курсор результата изначально является немодифицируемым;
- команды SQL необходимо писать в виде программного кода;
- создаваемое соединение необходимо поддерживать программным способом.

Рассмотрим пример программы, выбирающей из таблицы СОТРУДНИКИ базы данных Борей.mdb поля Фамилия, Имя и Должность:

```
*команды и функции SQL Pass Through для доступа через
*соединение Connect1 к таблице Сотрудники базы данных
Борей.mdb
local lnhandle
lnhandle=SQLConnect("Connect1")
? lnhandle
if lnhandle>-1
        SQLExec(lnhandle, "SELECT Фамилия, ;
Имя, Должность From Сотрудники", "Results")
        select Results
        Browse
        SQLDisconnect(lnhandle)
endif
```

В приведенной программе использованы только три функции **SQL Pass Through: SQLConnect( ), SQLExec( )** и **SQLDisconect( )**, всего их разработано 12.

Описание этих функций можно найти в справочном файле.

Методы разработки многопользовательских приложений

Типы блокировок в Visual FoxPro

В многопользовательской среде конфликты могут возникать в трех ситуациях:

- ***при редактировании данных, когда два или более пользователя пытаются вносить изменения в одну таблицу;***
- ***при создании запросов или отчетов, когда один из пользователей запускает длительный запрос, изменения, внесенные другими пользователями, могут сделать результаты запроса недействительными;***
- ***при сопровождении БД (создание резервных копий, переиндексирование, изменение структуры и т.д.) кто-то может попытаться получить эксклюзивный (исключительный) доступ к таблицам.***

При использовании современных сред визуального программирования, конкурирующие запросы могут поступать от нескольких процессов или экземпляров форм на одной и той же машине.

В Visual FoxPro поддерживается два типа блокировок.

Типы блокировок

Тип блокировки	Пользователь, установивший блок		Другие пользователи	
	Чтение	Запись	Чтение	Запись
Эксклюзивный	Да	Да	Нет	Нет
Блокировка изменений	Да	Да	Да	Нет

Эксклюзивную блокировку **Visual FoxPro обеспечивает только** ***на уровне таблицы и БД***, нельзя эксклюзивно блокировать строку или набор строк в таблице.

Если один пользователь установил эксклюзивную блокировку БД или таблицы, то другой пользователь при попытке доступа к этим объектам получит сообщение **File access is denied.**

Блокировка изменений в Visual FoxPro доступна на уровне строк, таблиц и БД.

Все блокировки, которые устанавливает Visual FoxPro, являются временными и существуют, пока блокировка не будет снята, или пока выполняется приложение. Они хранятся в операционной системе локальной сети (LAN), но не хранятся в БД или таблице.

Рассмотрим блокировки в порядке сужения сферы их действия.

Способы блокировки на уровне БД

Эксклюзивная блокировка

При открытии БД в Visual FoxPro всегда имеет определенный статус блокировки: либо эксклюзивная блокировка (exclusive), либо – общий доступ (shared).

Статус блокировки можно задать следующими способами:

- командой:

SET EXCLUSIVE ON|OFF

- в команде:

OPEN DATABASE <имя БД> <статус>

например,

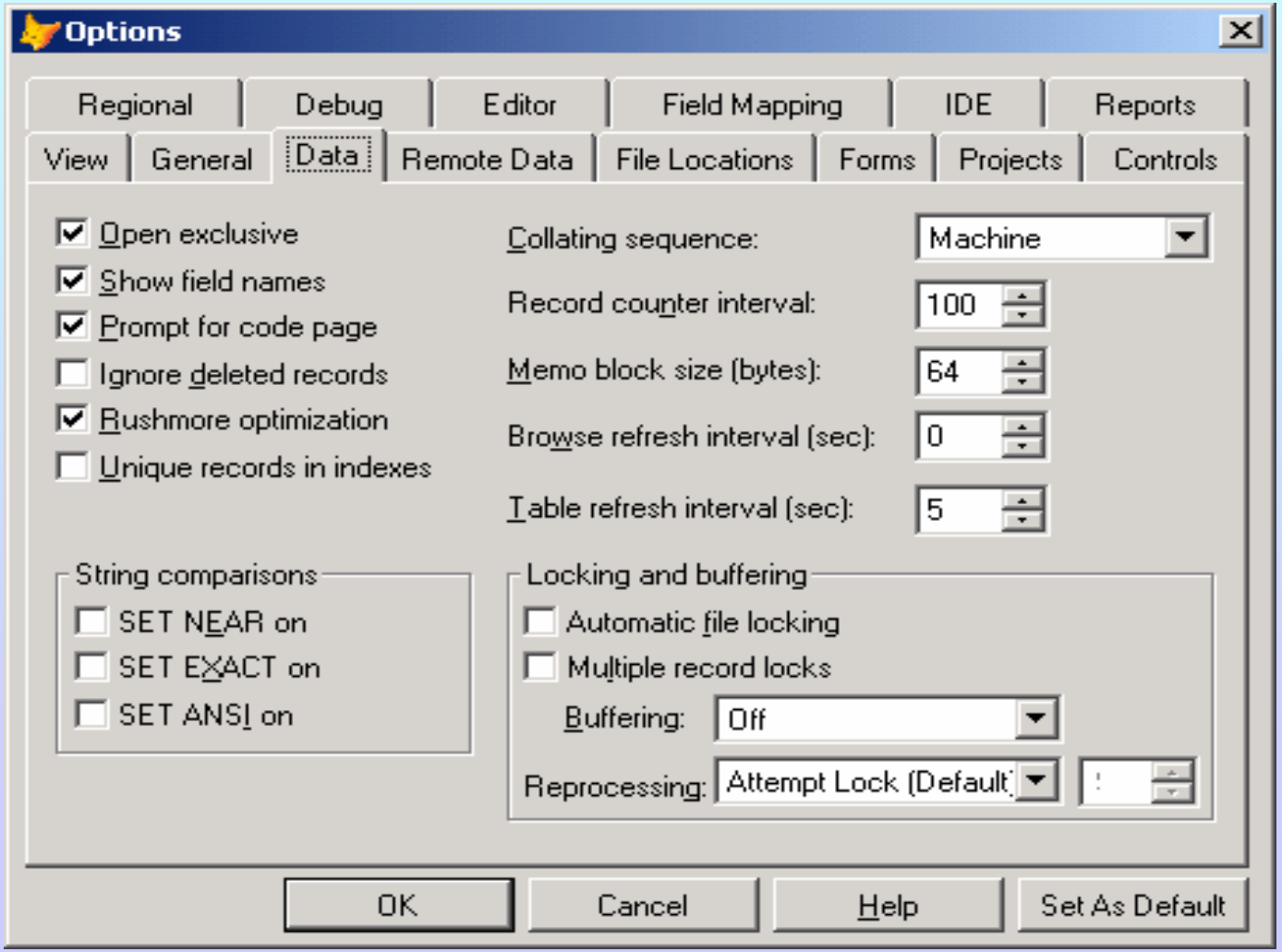
OPEN DATABASE b0100231 EXCLUSIVE| SHARED

Если параметр <статус> = **EXCLUSIVE,** БД открывается в монопольном режиме. Другие пользователи при попытке обращения будут получать ошибку.

В случае, когда <статус> = **SHARED,** БД открывается в режиме совместного пользования. Другие пользователи также получат к ней доступ.

Установка* SET EXCLUSIVE *по умолчанию определяется

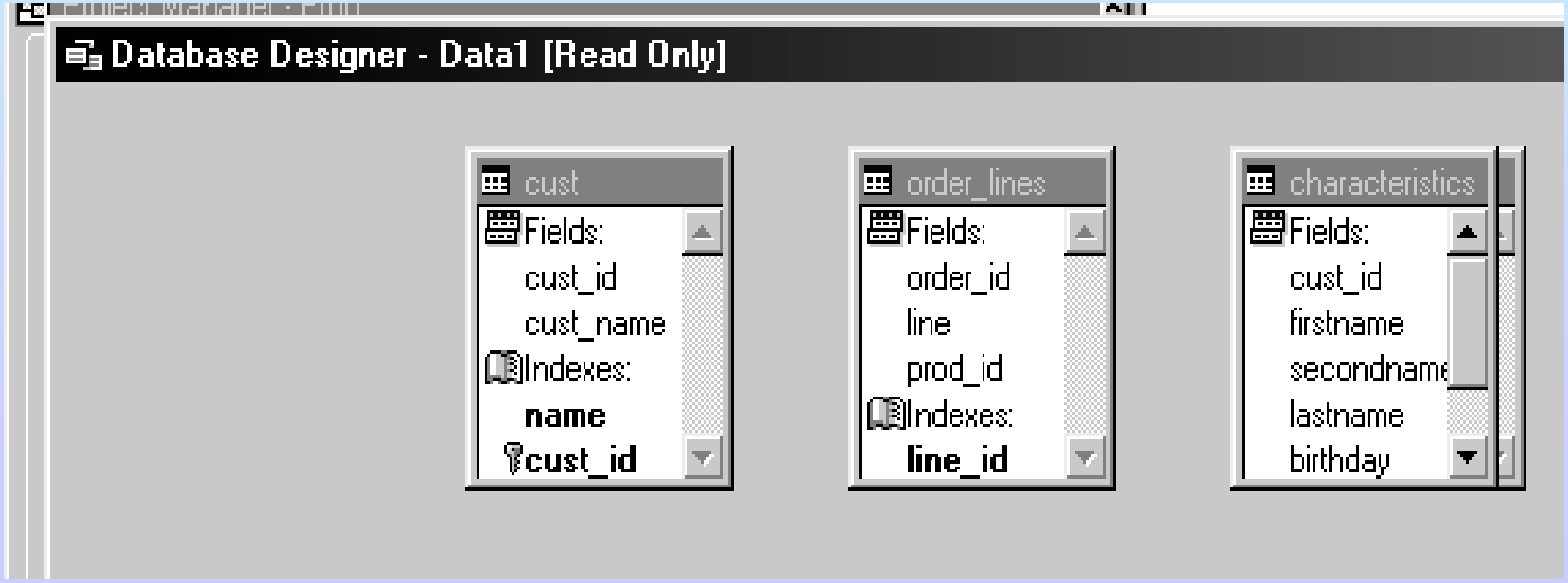
Tools - Options - вкладка **Data -** *снять*|*установить* флаг **Open Exclusive.**



Блокировка изменений

В команде OPEN DATABASE параметр <статус> может иметь еще два значения:

- **Noupdate** – ***блокировка изменений БД. Она функционирует в режиме Read Only (только для чтения), при этом в текущем сеансе данных запрещены операции добавления таблиц, установления связей и т.д.***



➢**Validate –*проверка целостности таблиц при открытии БД*.**

Visual FoxPro проверяет, доступны ли на диске таблицы и индексы, на которые имеются ссылки в базе данных. Кроме того, Visual FoxPro проверяет существуют ли в этих таблицах и индексах указываемые поля и теги.

Использование этих ключевых слов не связано с действием установки Set exclusive On|Off, приоритет которой выше.

Режим открытия базы данных не влияет на режим открытия таблиц.

***Блокировка БД действует до ее закрытия*.**

Способы блокировки на уровне таблиц

На уровне таблиц также возможна эксклюзивная блокировка и блокировка изменений.
Блокировка изменений удобна при запуске запроса, чтобы данные в таблице не были изменены во время его выполнения.
Таблица не может быть заблокирована двумя пользователями одновременно. Если запрос выполняется при заблокированной таблице, другой запрос не сможет заблокировать ее, пока первый не снимет блокировку.

Эксклюзивная блокировка

***Эксклюзивно*е EXCLUSIVE *или общее* SHARED**

***использование таблицы задается при ее открытии*:**

USE <имя таблицы> **EXCLUSIVE | SHARED**

Например,

USE customer **EXCLUSIVE**

***Проверить, в каком режиме вами открыта таблица или БД, позволяет функция* ISEXCLUSIVE().**

***Она возвращает логическое значение "истина"* (.T.) *при монопольном использовании, в противном случае - "ложь"* (.F.).**

? ISEXCLUSIVE(<имя таблицы| имя БД>, [N]**)**

***Параметр N равный 1 означает проверку статуса блокировки таблицы, 2 - БД. Если таблица находится в текущей рабочей области, этот параметр можно опустить*.**

Например:

? ISEXCLUSIVE(customer**)**

Блокировка изменений

Режим блокировки изменений включается с помощью функции FLOCK().

Если блокировка прошла успешно, то эта функция возвращает значение .T.

Если кто-либо другой уже заблокировал таблицу, функция FLOCK() возвратит значение .F.

В каждый момент времени установить блокировку таблицы может только один пользователь.

Установить блокировку изменений можно и в таблице, открытой в общем режиме.

Блокировка таблицы действует до ее закрытия или выполнения команды UNLOCK.

Определить, заблокировали ли вы изменения в таблице, можно по логическому значению, возвращаемому функцией ISFLOCKED(<имя таблицы>).
Определить, заблокирована ли таблица другим пользователем, можно только по результату функции = FLOCK().

Способы блокировки строк

Блокировка строк - это наиболее распространенная блокировка**.** На этом уровне разрешают конфликты редактирования. ***Блокировку записи может получить один пользователь, остальные будут ждать, пока он не закончит редактирование и не освободит запись.***

Открытая блокировка строк

***Чтобы заблокировать больше одной строки, установка MULTILOCKS должна быть включена в положение* ON:**

SET MULTILOCKS ON| OFF

Блокировка текущей строки в текущей рабочей области осуществляется с помощью функции RLOCK().

***Проверить результат ее выполнения можно, просмотрев возвращаемое значение (.T. или .F.), например, в команде*:**

? RLOCK()

Блокировка произвольной строки в произвольной рабочей области выполняется командой:

? RLOCK(“номер строки”, “имя таблицы”)

Например, заблокируем 12 и 15 строки в таблице customer:

? RLOCK(“12, 15”, “customer”)

Проверить, заблокирована ли конкретная строка, можно установив на нее указатель и выполнив функцию:

? ISRLOCKED()

Снять блокировку строк позволяет команда UNLOCK, действие которой распространяется на всю таблицу, или закрытие таблицы, пользователем, установившим блокировку.

Скрытая блокировка строк

На время обновления данных Visual FoxPro устанавливает блокировки автоматически, это так называемая скрытая блокировка строк. В связи с чем, в каждый момент времени только кто-то один может делать записи в общие данные. ***Скрытая блокировка строк устанавливается на время работы команд*** **UPDATE, INSERT INTO, GATHER MEMVAR, REPLACE,** ***функции*** **TABLEUPDATE()** ***и при редактировании таблицы с помощью формы или в режиме*** **BROWSE**.

Блокировка строки заголовка

Каждая таблица Visual FoxPro, кроме строк данных, содержит еще строку заголовка, в которой сохраняется структура таблицы. Строка заголовка имеет номер, равный нулю, и включает определение параметров строк: имена столбцов, их длину, тип данных столбцов и счетчик числа строк в таблице.

При добавлении пустой записи строка заголовка скрыто блокируется, чтобы увеличить значение счетчика записей. После выполнения операции добавления, блокировка снимается.

Можно открыто установить блокировку строки заголовка:
? RLOCK(“0”, “customer”)
Снять эту блокировку можно командой:
UNLOCK RECORD 0
В случае, если попытка установить блокировку изменения оказалась неудачной, Visual FoxPro повторит попытку и в строке состояния появится сообщение:
Waiting for lock…(Ожидается блокировка).

Чтобы установить новую блокировку, необходимо снять предыдущую. Все блокировки снимаются при закрытии таблиц и БД.

Буферизация строк и таблиц

Visual FoxPro обеспечивает еще две стратегии разрешения конфликтов при редактировании:

- ***буферизация строк и таблиц;***
- ***обработка транзакций.***

Принципиальное отличие этих двух стратегий заключается в том, что буферизация строк и таблиц ориентирована на отдельные таблицы, обработка транзакций - на множество таблиц в БД.

Буферизация строк и таблиц позволяет отменить те изменения, которые внесены пользователем при редактировании отдельной таблицы.

Метод транзакций гарантирует, что все изменения двух или более таблиц либо полностью принимаются, либо полностью отвергаются. В Visual FoxPro транзакции применимы только к таблицам, входящим в БД.

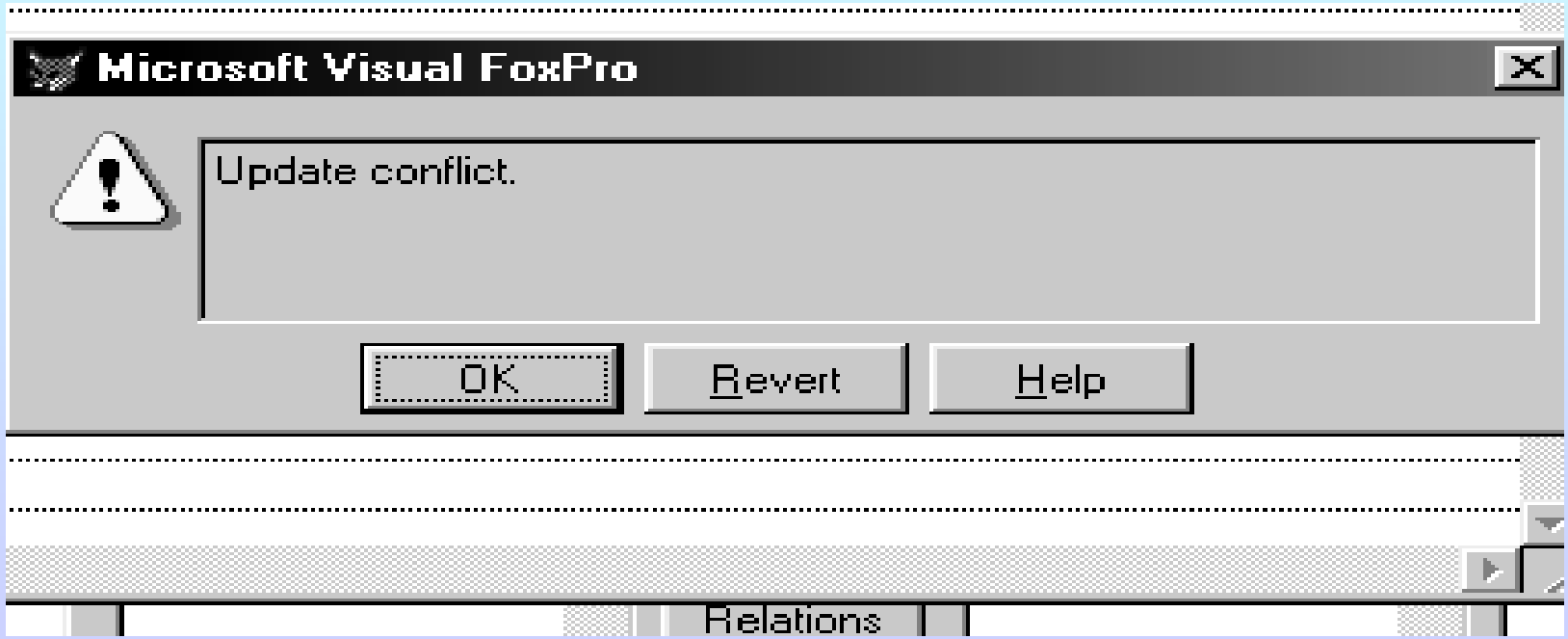
Обе стратегии основаны на создании временных копий данных, по которым можно восстановить исходные данные, если обновление закончится неудачей.

При использовании буферизации можно применить оптимистический или пессимистический тип блокировок на уровне строк и таблиц.
Оба типа блокировок могут быть применены одновременно к разным строкам одной таблицы.

Оптимистический тип блокировок

Оптимистическая блокировка откладывает все блокировки данных до момента обновления. Поэтому кажется, что один и тот же набор строк может редактировать несколько пользователей одновременно. Моментом обновления для таблицы является ее закрытие или выполнение функции Tableupdate(), для строки - перемещение указателя на следующую строку или закрытие таблицы (т.е. запись на диск внесенных изменений).

Если другой пользователь успел обновить данные строки, пока вы ее редактировали, Visual FoxPro выдаст сообщение об ошибке



При оптимистической блокировке таблиц пользователь может редактировать большое количество записей до момента обновления таблицы. Если хотя бы одна запись была изменена кем-то после начала буферизации, то все изменения в буфере рассматриваются как недействительные и Visual FoxPro сообщит об ошибке. Можно пренебречь этим сообщением, можно отменить изменения.

Оптимистическая блокировка подходит, когда требуется минимизировать расходы времени на блокировку, если нельзя позволить пользователю установить блокировку на неопределенно долгий срок.

Пессимистический тип блокировок

При пессимистической блокировке строк Visual FoxPro устанавливает блокировку изменений на уровне строки на все время процесса редактирования строки.

Этому типу блокировки следует отдать предпочтение, если пользователь редактирует отдельную таблицу построчно, и данные в таблице настолько важны, что нельзя позволить, чтобы один пользователь писал поверх изменений внесенных другим, а затраты времени на блокировку не имеют значения.

При пессимистической блокировке таблицы Visual FoxPro автоматически блокирует все строки таблицы.

Остальные пользователи не видят изменений до момента обновления.

Выбор режима буферизации

По умолчанию при прямом редактировании данных в таблице Visual FoxPro применяется не буферизованная блокировка строк.

Режим буферизации таблиц можно включить на вкладке Data диалогового окна Options, открываемого в пункте Главного меню Tools.

При визуальном создании формы можно задать буферизованное редактирование данных с помощью окна свойств Properties.

Эти установки можно перекрыть с помощью свойств среды окружения Data Environment.

Использование многопользовательских форм

С целью проверки способов буферизации на уровне строк и таблиц, конфликты между двумя и более пользователями можно имитировать, запуская несколько экземпляров одной и той же формы. Для того чтобы запустить несколько экземпляров формы, каждый экземпляр формы должен иметь собственный (Private) сеанс данных. Собственный (Private) сеанс данных можно задать с помощью свойства DataSession в окне свойств формы на вкладке Data. Значение этого свойства при этом должно быть равно двум.

Обработка транзакций

Транзакция - «логическая единица работы». Она позволяет либо принять все изменения, либо не принимать ни одно из них.

В Visual FoxPro транзакция задается с помощью команды Begin Transaction. Когда она выполняется, все обновления данных записываются не в БД, а в специальный буфер транзакции. Внести эти изменения из буфера в БД позволяет команда End Transaction.

Для возврата БД в исходное состояние применяется команда RollBack (откат транзакции).

Обработку транзакций в Visual FoxPro можно применять только к таблицам, входящим в одну БД. Изменения свободных таблиц записываются напрямую, и их нельзя отменить с помощью отката транзакции. В этом отличие метода транзакции от буферизации. Буферизацию можно применить к свободным таблицам, но только к одной таблице за один раз.

Элементы объектно-ориентированного программирования (ООП) в Visual FoxPro

1. *Терминология ООП*
2. *Создание классов в Visual FoxPro*

2.1 Базовые классы

2.2 Создание класса с помощью конструктора классов Class Designer

2.3 Создание простого класса программными средствами

3. *Создание объектов как экземпляров класса*

Элементы объектно -ориентированного программирования (ООП) в Visual FoxPro

Терминология ООП

Основная идея ООП – построить новое приложение из заранее созданных компонентов.

<u>Объект</u>

Объектом можно назвать любой предмет. *Работая с формами в курсе «Информатика», мы уже знакомились с понятием объект. Форма – объект контейнер.*

Объект - совокупность данных (свойств) и функций (методов выполнения некоторых действий). Свойства можно считать физическими атрибутами объекта, или другими словами - это данные, которые инкапсулированы в объекте.

Например, объект командная кнопка в форме имеет следующие атрибуты (свойства):

- *расположение на форме;*
- *ширина;*
- *высота;*
- *цвет;*
- *надпись на кнопке;*
- *тип шрифта, размер и стиль текста надписи.*

Объектами необязательно должны быть элементы интерфейса, это может быть, например, список сотрудников какой-то фирмы. Свойства объекта СОТРУДНИК в этом случае могут включать:

- *номер служащего:*
- *оклад;*
- *текущий заработок.*

Каждый атрибут описывает объект, а их сочетание делает его уникальным.

Свойства в Visual FoxPro - это переменные памяти, которые присоединены к объекту и имеют область видимости объекта (т.е. время жизни объекта и его свойств одинаково).
Чтобы настроить значение свойства, надо присвоить это значение соответствующей переменной.
В конструкторе форм ***Form Designer*** *свойства каждого визуального объекта доступны для редактирования в окне свойств* ***Properties***.
Для этого необходимо выделить объект, например, **TextBox** (текстовое поле) и выполнить команду **Properties** либо из пункта Главного меню **View** (вид), либо из контекстного меню объекта, либо с помощью соответствующей кнопки на панели инструментов.

Чтобы программно обратиться к свойству, нужно указать:
<имя объекта>.<имя свойства>

Например, вывести на объект **label**1 надпись "Я умница!!!", можно, присвоив необходимое значение его свойству **CAPTION**:

label1**.caption**= "Я умница!!!"

Узнать значения свойства можно с помощью вывода его на экран, как и для простой переменной:

? label1**.caption**

Кроме свойств, для объектов существуют встроенные методы. Методы, это действия, которые выполняет объект при наступлении связанных с ними событий. Событие – нечто происходящее во внешнем мире, на что объект должен реагировать. Каждому событию должен отвечать программный код метода обработки этого события, который вызывается автоматически.

Например, событие щелчок левой кнопкой мыши **ClickEvent** по объекту – командной кнопке обрабатывается одноименным методом **Click**, т.е. будет выполнена процедура, программный код которой находится в методе **Click**.

***В конструкторе форм* Form Designer *программный код метода доступен для редактирования в окне свойств* Properties *на вкладке* Methods.**

Программно вызвать метод на выполнение можно командой:

<имя объекта>.<имя метода>([<список параметров>])

Например, для кнопки **CmdOK** *метод* **Click**( ) *можно вызвать командой*:

CmdOK.Click( )

Даже если <список параметров> отсутствует, скобки ( ) все равно необходимо ставить, чтобы отличать методы объекта от его свойств.

Класс

Класс – заготовка, которая будет использоваться для создания объектов при разработке новых приложений. Он содержит описание объекта и действий над этим объектом.

Однотипные объекты создаются как экземпляры одного класса, это позволяет не создавать каждый объект индивидуально.

Мы уже знакомы с такими классами, как форма, отчет, элементы визуального интерфейса.

В Visual FoxPro классы сохраняются в файлах с расширением .VCX.

Инкапсуляция

В старых языках процедурного типа данные и программный код существовали отдельно. Инкапсуляция - объединение в понятии объект данных (свойств) и программного кода (методов). Код может оперировать над этими данными. Разработчик должен обеспечить открытый интерфейс классов. Интерфейс определяет, какие программные коды и данные (методы и свойства) должны быть доступны извне объекта. Этим достигается изменение объекта без изменения программного кода самого объекта.

Например, мы изменяем цвет фона формы или ее заголовок с помощью интерфейса, реализованного в виде окна Properties *и панели инструментов, даже не видя программного кода объекта формы.*

Полиморфизм

Используя ООП, можно определять различные объекты с похожими общими интерфейсами. В Visual FoxPro полиморфизм определяют как явление, когда два объекта имеют идентичный интерфейс и потенциально разные реализации.

Например, можно создать объект СЧЕТ и объект ЗАКАЗ и для них открытый интерфейс ПЕЧАТАТЬ ДОКУМЕНТ. Этот интерфейс можно представить в виде окна, в полях ввода которого задать выбор типа принтера, количества копий, размер бумажного листа и т.д. Этот интерфейс будет направлять документ программе печати. Реализация программы печати будет отличаться для обоих объектов, т.к. они имеют разные свойства, но для пользователя эти детали скрыты.

Наследование

Наследование – способность передачи функциональных возможностей от базового класса к подклассу.

Технология наследования позволяет определять новый объект, основываясь на определении исходного объекта.

Например, есть объект командная кнопка, которая выполняет какую-то процедуру (действие) при щелчке на ней левой кнопкой мыши. Мы создавали с Вами кнопки Next, Prev, Bottom, Top для перемещения по записям. Если изменить класс объекта командная кнопка, так, чтобы действие выполнялось при щелчке на ней правой кнопки мыши, то все приложения, использующие этот объект, перестанут работать.

Чтобы этого не происходило, необходимо, во-первых, создать новый класс, который основан на определении исходного; во–вторых, добавить нужный программный код, не внося изменений в исходный класс.

Класс, наследующий свойства и методы от другого класса, называют подклассом. Такое наследование называют иерархией класса. Как только происходит наследование от класса, возникает новый элемент этой иерархии.

Иерархия класса похожа на генеалогическое дерево человека.

Иерархии классов следует отличать от *иерархий вложенности*. *Первые – связаны с наследованием, вторые имеют дело с формами и другими классами контейнерами, которые могут содержать такие элементы, как кнопки, окна редактирования, поля ввода со списком и др.*

В Visual FoxPro эти элементы обязательно должны находиться в какой-нибудь форме. Форма – это объект - контейнер, который содержит другие элементы, при этом форма является родителем, а элемент – потомком.

С иерархий вложенности мы имели дело, когда в разделе 1.1 говорили об обращении к свойствам и методам объекта, например, следующая команда присваивает свойству ***ControlSource*** *объекта* ***oTextBox1*** *на форме* ***oForm1*** *значение переменной nPrice:*

oForm1.oTextBox1.ControlSource=nPrice

[1] Префикс «о» принято добавлять для обозначения имени объекта.

Создание классов в Visual FoxPro

Базовые классы

Классы в Visual FoxPro можно создать двумя способами: с помощью средств визуального интерфейса или посредством программного кода.

Новые классы создают на основе готовых базовых компонентов, которые называют классами предков. В Visual FoxPro базовыми классами (суперклассами) являются:

- ***ActiveDoc (Объект активного документа, который можно поместить в броузер);***
- ***CheckBox (Флаговое окно);***
- ***Column (Столбец объекта класса Grid);***
- ***ComboBox (Поле со списком);***
- ***CommandButton (Командная кнопка);***

- ***CommandGroup (Группа командных кнопок);***
- ***Conteiner*** *(Контейнер);*
- ***Control*** *(Базовый класс элементов управления);*
- ***Cursor*** *(Определение курсора в среде данных);*
- ***Custom*** *(Пользовательский класс);*
- ***DataEnvironment*** *(Совокупность курсоров и таблиц, которую можно открыть как единое целое);*
- ***EditBox*** *(Элемент редактирования);*
- ***Form*** *(Форма)*;*
- ***FormSet*** *(Набор форм)*;*
- ***Grid*** *(Сетка, таблица)*;*
- ***Header*** *(Заголовок столбца объекта класса Grid);*
- ***HyperLink*** *(Гиперссылка);*
- ***Image*** *(Гарфический образ);*

- ***Label*** *(Метка);*
- ***Line*** *(Линия);*
- ***ListBox*** *(Список);*
- ***OptionButton*** *(Кнопки с возможностью выбора одного параметра);*
- ***OptionGroup*** *(Группа объектов* ***OptionButton);***
- ***OleBoundControl*** *(Контроллер объектов* ***OLE****);*
- ***OleControl*** *(Элемент управления* ***OLE****);*
- ***Page*** *(Единственная вкладка в составе объекта* ***PageFrame)****;*
- ***PageFrame*** *(Страничный блок);*
- ***ProjectHook*** *(Объект открытого проекта, в котором обеспечен программный доступ к событиям проекта);*
- ***Relation*** *(Отношение между двумя курсорами в среде данных);*

- ***Separator*** *(Разделитель);*
- ***Shape*** *(Графическая фигура);*
- ***Spinner*** *(Счетчик);*
- ***TextBox*** *(Поле ввода);*
- ***Timer*** *(Таймер);*
- ***ToolBar*** *(Панель инструментов).*

Базовые классы поставляются вместе с Visual FoxPro и находятся в файле _base.VCX.

Класс Custom является самым простым и часто используется как суперкласс для пользовательских классов.

Всем базовым классам присущи следующие базовые свойства, события и методы.

Свойство	Описание
Class	Имя класса
BaseClass	Имя суперкласса
ClassLibrary	Путь к файлу библиотеки, в которой содержится определение класса
ParentClass	Имя класса объекта-родителя, в который включен объект данного класса

Базовые события и методы, присущие всем базовым классам

Событие	Описание
Init	Вызывается при создании экземпляра класса. Метод-обработчик возвращает значение **.F.**, если объект не создан
Destroy	Вызывается при уничтожении экземпляра класса
Error	Вызывается при возникновении ошибки

Методы объектов, используемые в лабораторной работе

- **Load** – загрузка объекта в ОЗУ;
- **LostFocus** – потеря объектом фокуса ввода;
- **Refresh –** обновление;
- **Click** – щелчок LM;
- **Init –** создание объекта.

Создание класса с помощью конструктора классов Class Designer

Окно Class Designer во многом похоже на окно конструктора форм Form Designer.

Здесь с помощью панелей инструментов также можно добавлять в создаваемый класс элементы, задавать их свойства и т.д.

Отличия Form Designer от Class Designer заключаются в том, что первый обеспечивает возможности, специфические для разработки форм, такие, как создание среды окружения и интерактивное переключение из режима разработки в режим выполнения (Form| Run).

Конструктор Class Designer создает заготовки для объектов, и эти заготовки нельзя запустить на выполнение непосредственно.

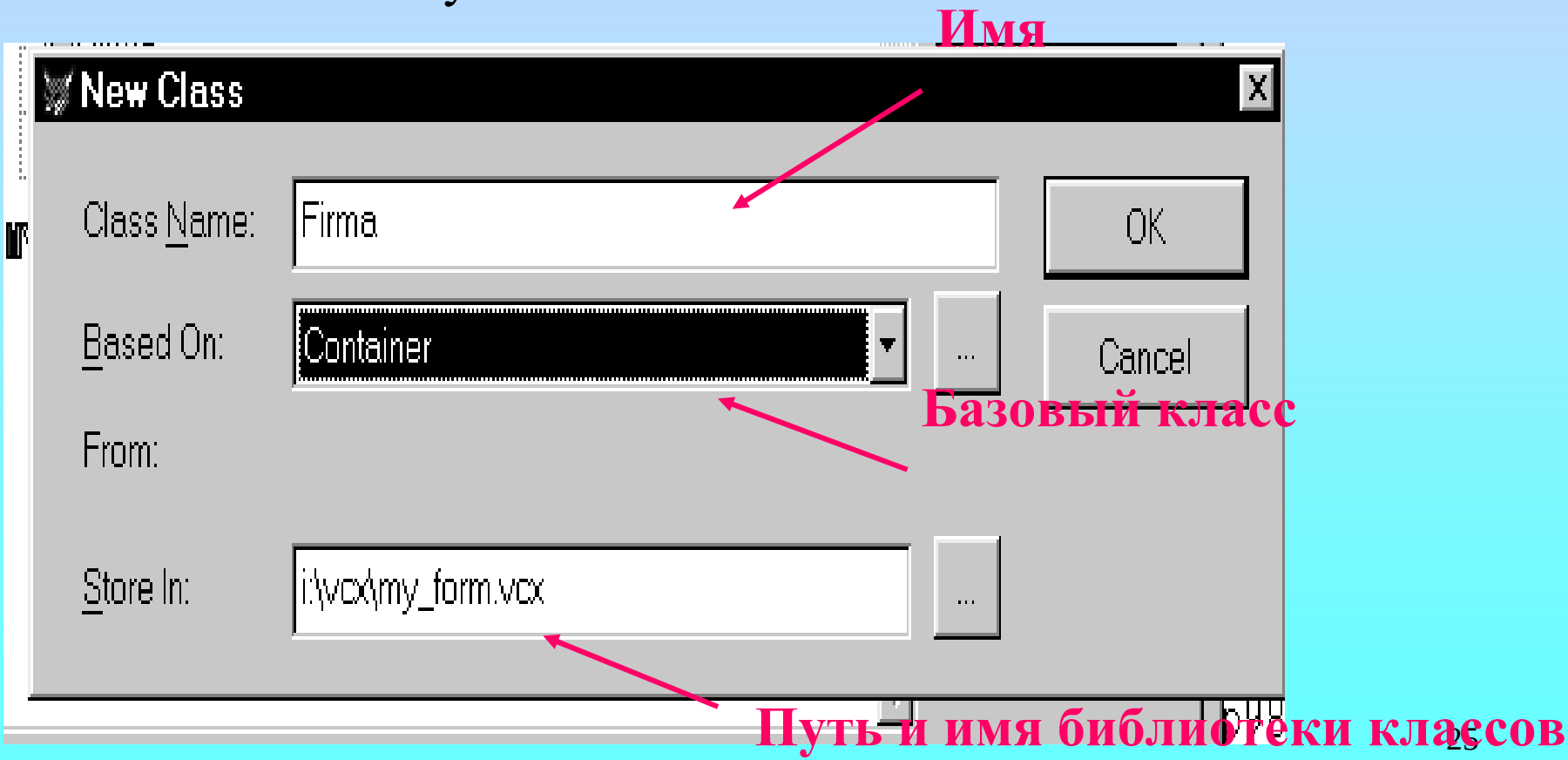
Чтобы проверить результат создания класса, необходимо создать объект данного класса.

Например, нельзя запустить на выполнение класс форма, необходимо создать объект класса формы.

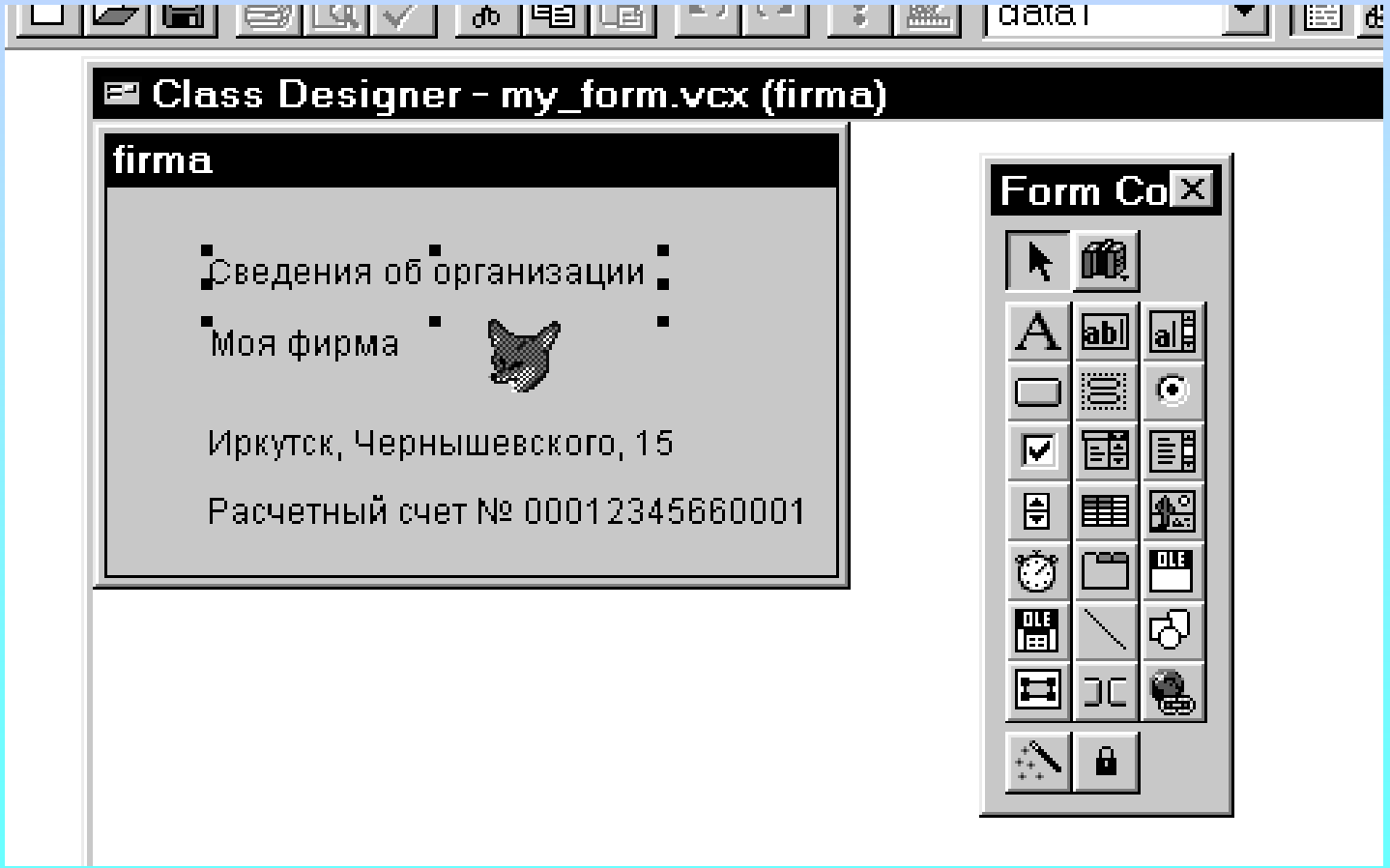
Пользовательские библиотеки классов предпочтительно размещать в отдельных папках. Можно создать специальную папку с названием VCX, куда помещать все созданные библиотеки классов.

Рассмотрим пример создания класса **Firma** штампа фирмы, который можно будет помещать в любые формы проекта. Для этого необходимо выполнить следующие действия:

1. В окне диспетчера проекта перейти на вкладку **Classes.**
2. Нажать кнопку **New.**



В окне конструктора классов с помощью объектов **Label** (**Caption**) и **Image (Picture),** выбранных на панели инструментов **Form Control**, расположить на заготовке класса следующую информацию о фирме



Фон объектов класса Firma предпочтительнее задать прозрачным (**BackStyle** - 0 **Transparent**), свойства рамки, ограничивающей объект, начинаются со слова **Border**, их можно задать по усмотрению разработчика. Чтобы рамка была невидима, на вкладке **Layout** в окне свойств **Properties** класса **Firma** следует присвоить свойству **BorderWidth** (ширина рамки) значение 0.

3. Закрыть окно конструктора с сохранением класса **Firma.**

Создание простого класса программными средствами

Новый пользовательский класс можно создать с помощью команды DEFINE CLASS:

DEFINE CLASS <Имя нового класса> AS <Имя родительского класса>

<Список добавляемых или изменяемых свойств>

<Список добавляемых или изменяемых методов>

ENDDEFINE

В качестве родительского класса может быть использован базовый или любой другой пользовательский класс. Все свойства и методы родительского класса автоматически наследуются новым (дочерним) классом. В связи с чем в команде DEFINE CLASS необходимо указать только добавляемые или изменяемые свойства и методы.

При написании кода класса, когда еще не существует сам объект, при обращении к его свойствам, вместо имени объекта (экземпляра класса), используется ключевое слово THIS. Это означает, что выполняется обращение к свойству самого объекта.

В лабораторной работе №6а в курсе «Информатика» мы рассматривали пример создания объекта командной кнопки, завершающей событие чтения данных в объекте **Grid**. Объект командной кнопки создавали на основе нового пользовательского класса, названного CmndBtn1. Пользовательский класс CmndBtn1 создавали на основе базового класса **COMMANDBUTTON** с помощью следующего программного кода**:**

DEFINE CLASS CmndBtn1 **AS COMMANDBUTTON**

Caption = 'Нажми, чтобы закрыть' && надпись на кнопке

Left = 50 && столбец формы

Top = 220 && строка формы

Height = 30 && высота кнопки

Width =200 && ширина кнопки

PROCEDURE Click && метод обработчик события **Click**

CLEAR EVENTS && прекращение чтения, закрытие формы

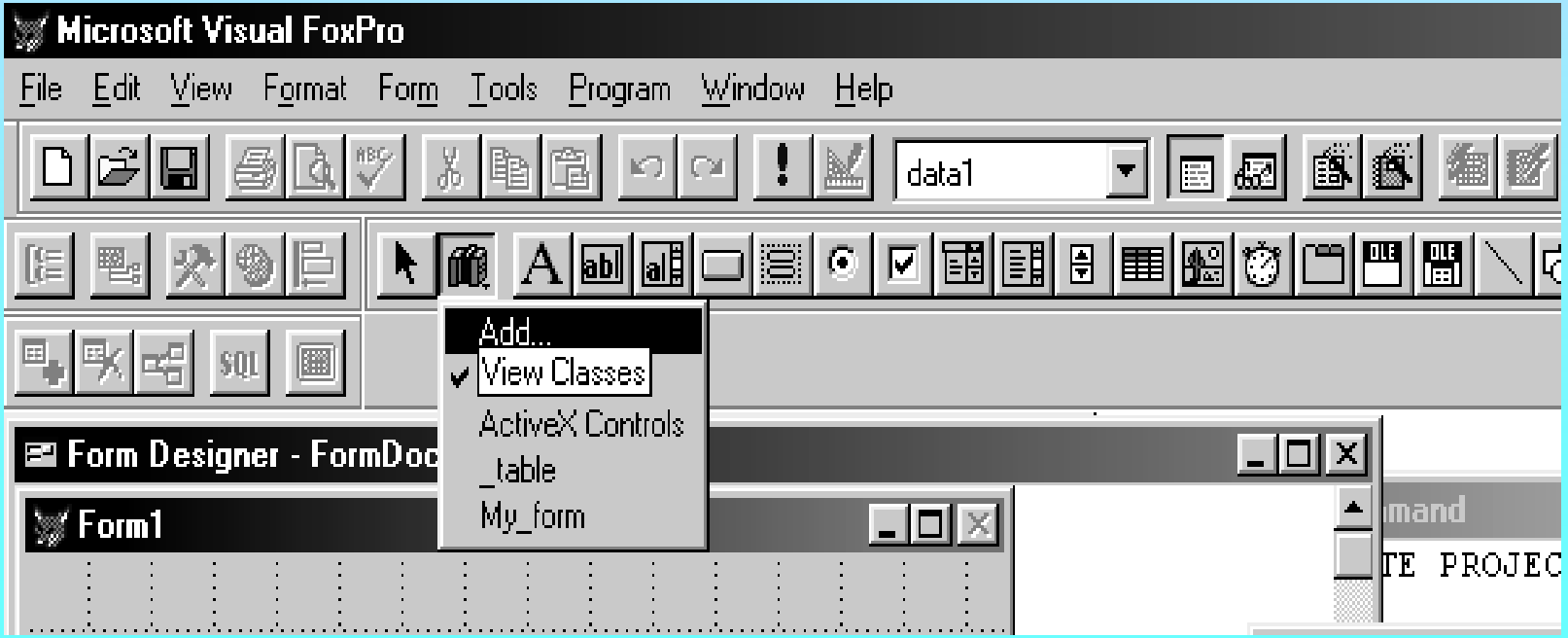
ENDDEFINE

Создание объектов как экземпляров класса

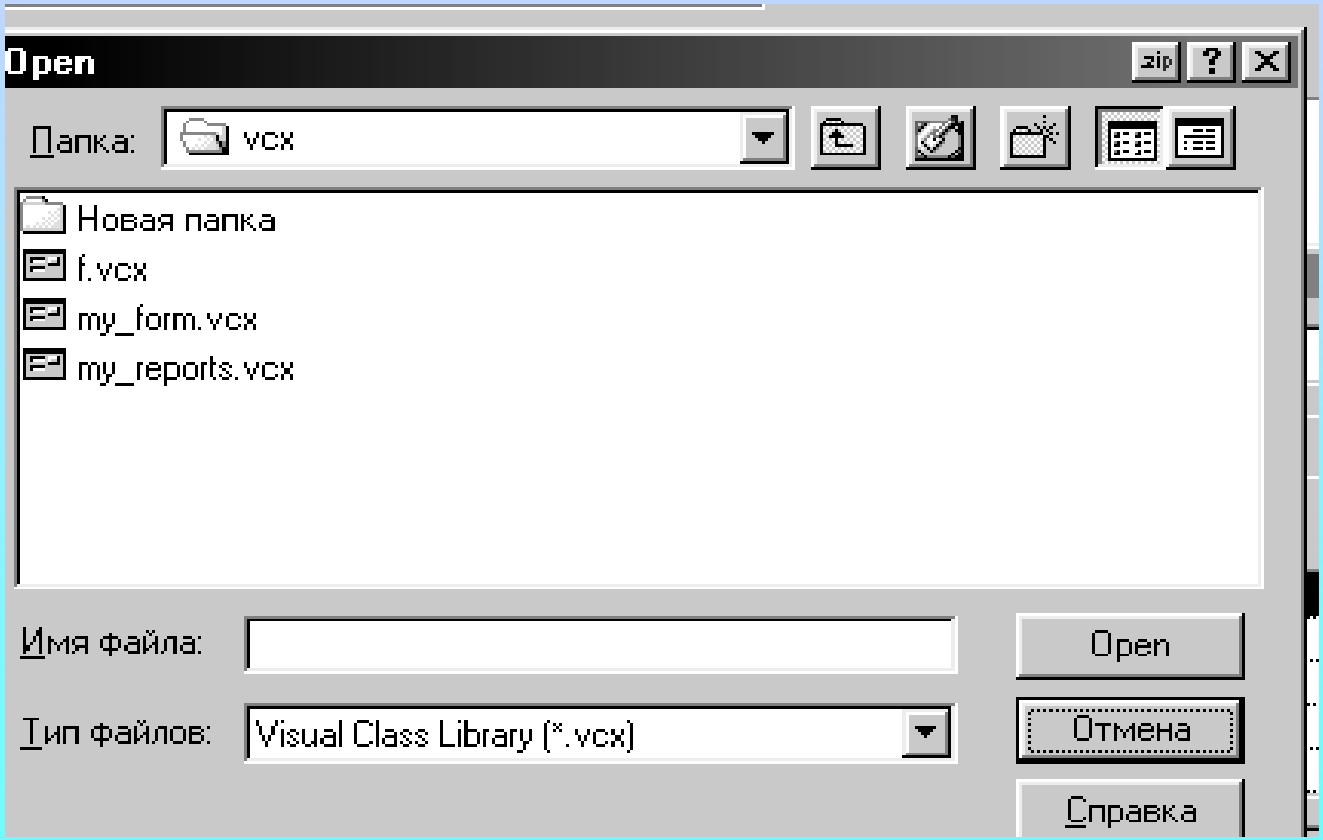
Чтобы проверить результат создания класса, необходимо создать объект данного класса.

В случае, когда пользовательский класс помещен в библиотеку классов, как, например, класс Firma, *его можно использовать при разработке любых форм проекта с помощью конструктора.* ***Чтобы добавить объект класса Firma на вновь создаваемую форму, необходимо открыть окно конструктора формы Form Designer .***

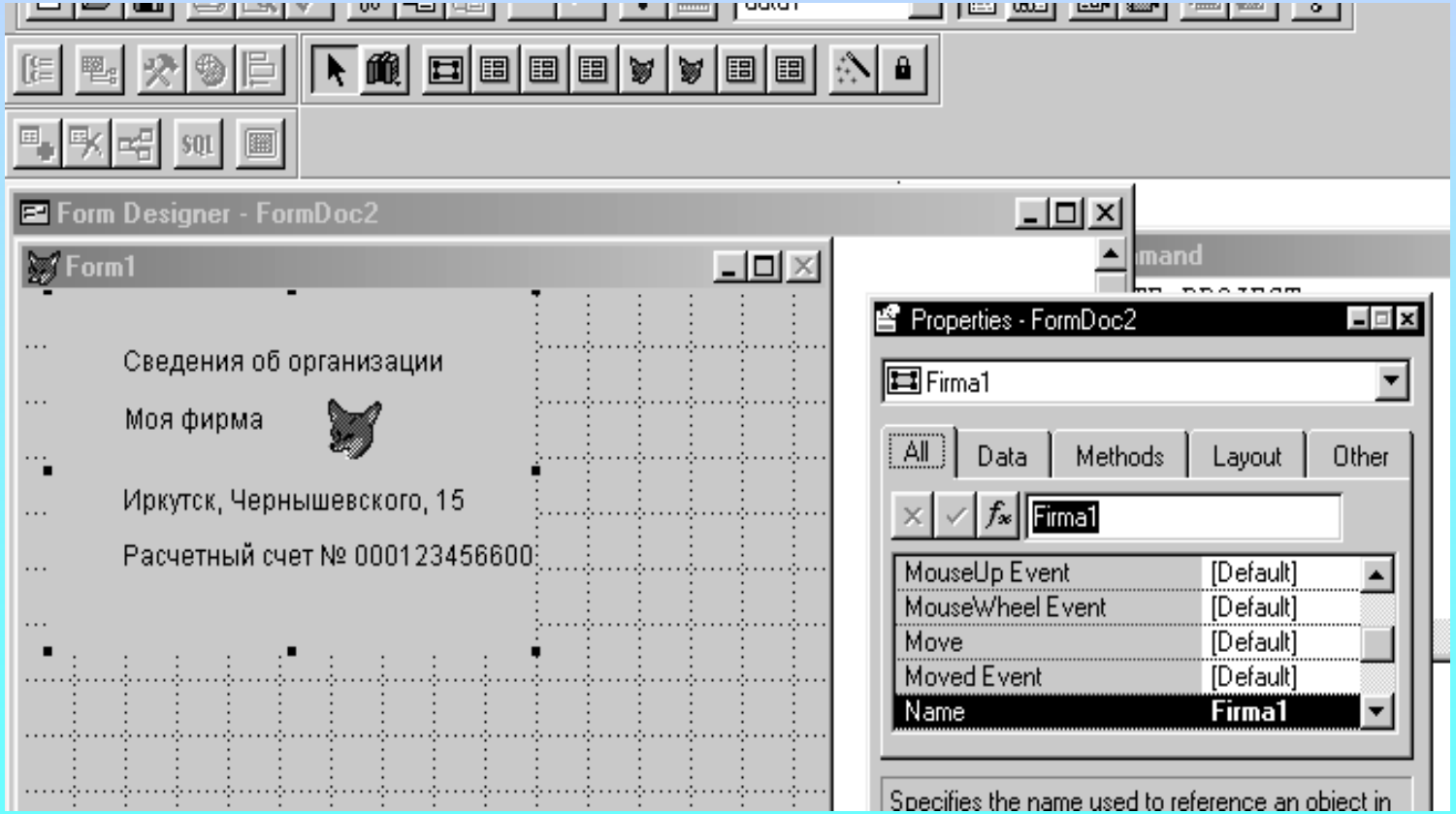
На панели инструментов **FormControl** следует нажать кнопку **View Classes**, предназначенную для просмотра списка библиотек классов, доступных через панель инструментов. Чтобы на панели инструментов были представлены кнопки, соответствующие членам библиотеки класса, необходимо выбрать ее имя в открывшемся списке, например, **My_form.**



Если имя библиотеки отсутствует в списке, то команда **Add** в меню **View Classes** с помощью окна **Open**-диалога позволяет добавить на панель инструментов любую библиотеку классов



Теперь на панели инструментов **FormControl** появится кнопка, соответствующая классу **Firma**. Если перетащить эту кнопку на форму и указать размер размещаемого объекта класса **Firma**, то на новой форме появится штамп фирмы



При запуске формы на выполнение автоматически будет создан объект композитного класса **form,** включающий объекты других классов (в частности Firma), добавленных на форму с помощью конструктора.

Для восстановления стандартной панели инструментов **FormControl,** необходимо снова нажать кнопку **View Classes** и выбрать из списка панель **Standard**.

Программное создание объектов

Экземпляр класса (объект) можно создать с помощью функции **CreateObject()**:

<имя переменной типа **object>=CreateObject(“**<имя класса>**”,** [<список параметров, которые можно передать методу **INIT**()>])

В переменной типа **object** сохраняется ссылка на созданный объект.

Например, следующая команда создает объект loForm класса **FORM**:

loForm=**CREATEOBJECT(“FORM”)**

Объект может быть создан функцией **CreateObject()**, если в этот момент *определение класса будет доступно программе*.

- Рассмотрим пример программы, помещающей в объект gr класса **GRID** на форме form1 таблицу Customer. Для завершения работы программы используется командная кнопка btn1 пользовательского класса CmndBtn1:

```
form1=CREATEOBJECT("form")   && Создать объект form1
                                *класса form
form1.show()                 && Визуализировать объект form1
form1.Enabled=.t.     && Сделать объект form1 доступным для
                                *пользователя
Use Customer      && Открыть таблицу
form1.addobject("gr","grid")   && добавить на форму объект ;
                                *класса GRID с именем gr
```

form1.gr**.visible=.T.** && визуализировать объект gr

Form1.AddObject('Btn1','CmndBtn1') && добавить на форму объект ;

*класса CmndBtn1 с именем Btn1

Form1. Btn1**.Visible =.T.**

read events && Активизировать событие чтения

DEFINE CLASS CmndBtn1 **AS COMMANDBUTTON** && создание

*класса командной кнопки для завершения события чтения

Caption = 'Нажми, чтобы закрыть'

Left = 50 && столбец формы

Top = 220 && строка формы

Height = 30 && высота кнопки

Width =200 && ширина кнопки

PROCEDURE Click

CLEAR EVENTS && прекращение чтения, закрытие формы

ENDDEFINE

Если код класса находится в библиотеке классов, то эту библиотеку необходимо открыть в дополнение к системной:

SET CLASSLIB TO <полное имя библиотеки классов> **ADDITIVE**

Например,

SET CLASSLIB TO “I:\vcx\My_form.vcx” **ADDITIVE**

Объект класса, включенного в библиотеку классов, можно создать с помощью функции **NEWOBJECT().** При использовании этой функции библиотеку классов заранее открывать не требуется, так как ее полное имя передается как параметр функции:

<имя переменной типа **object>=NEWOBJECT(“**<имя класса>**”, ”**< полное имя библиотеки классов >**”)**

По умолчанию объекты класса **FORM** создаются невидимыми. Для визуализации объекта класса **FORM** следует обратиться к его методу **SHOW():**

<имя переменной типа **object>. SHOW( )**

У других визуальных объектов, таких как **GRID** или **COMMANDBUTTON** с этой целью используется свойство **Visible,** которому следует присвоить значение **.T.:**

<имя объекта класса Form>.<имя объекта>**.Visible =.T.**

Например,

Form1.Btn1**.Visible =.T.**

Использование фундаментальных классов в Visual FoxPro

Назначение фундаментальных классов

Преимуществом ООП является повторное использование классов, что ускоряет разработку приложений. Фирма Microsoft в среде Visual FoxPro поставляет набор библиотек фундаментальных классов, содержащий более 100 классов.

Библиотеки фундаментальных классов хранятся в VCX-файлах и сосредоточены в каталоге \VFP98\FFC.

Большинство имен фундаментальных классов начинаются символом подчеркивания.

Этот символ добавляется в имена классов, производных от базовых классов, которые находятся в библиотеке _base.VCX.

Это говорит о том, что если внести усовершенствования в базовые классы, то они будут унаследованы и соответствующими фундаментальными классами.

Существует 14 библиотек фундаментальных классов, сгруппированных по категориям.

Категория	Файл библиотеки классов
Application	_app.vcx
Automation	autgraph.vcx
Buttons	_miscbtns.vcx
Data Navigation	_datanav.vcx
Data Query	_dataquery.vcx
Date / Time	_datetime.vcx
Dialogs	_dialogs.vcx
Internet	_hiperlink.vcx
Menu	_table2.vcx
Multimedia	_multimedia.vcx
Output	_reports.vcx
Text Formatting	_format.vcx
User Control	_ui.vcx
Utility	registry.vcx

В большинстве случаев классы из набора фундаментальных классов Visual FoxPro можно непосредственно вставить в экранную форму без корректировки программного кода.

Замечание

Классы, производные от Form, FormSet и ToolBar, которые сами являются формами, можно добавить только в проект и запускать программно.

Доступ к фундаментальным классам с помощью панели инструментов Form Control

При разработке пользовательских форм кнопки, соответствующие компонентам библиотеки классов, могут быть доступны через панель инструментов **Form Control.**

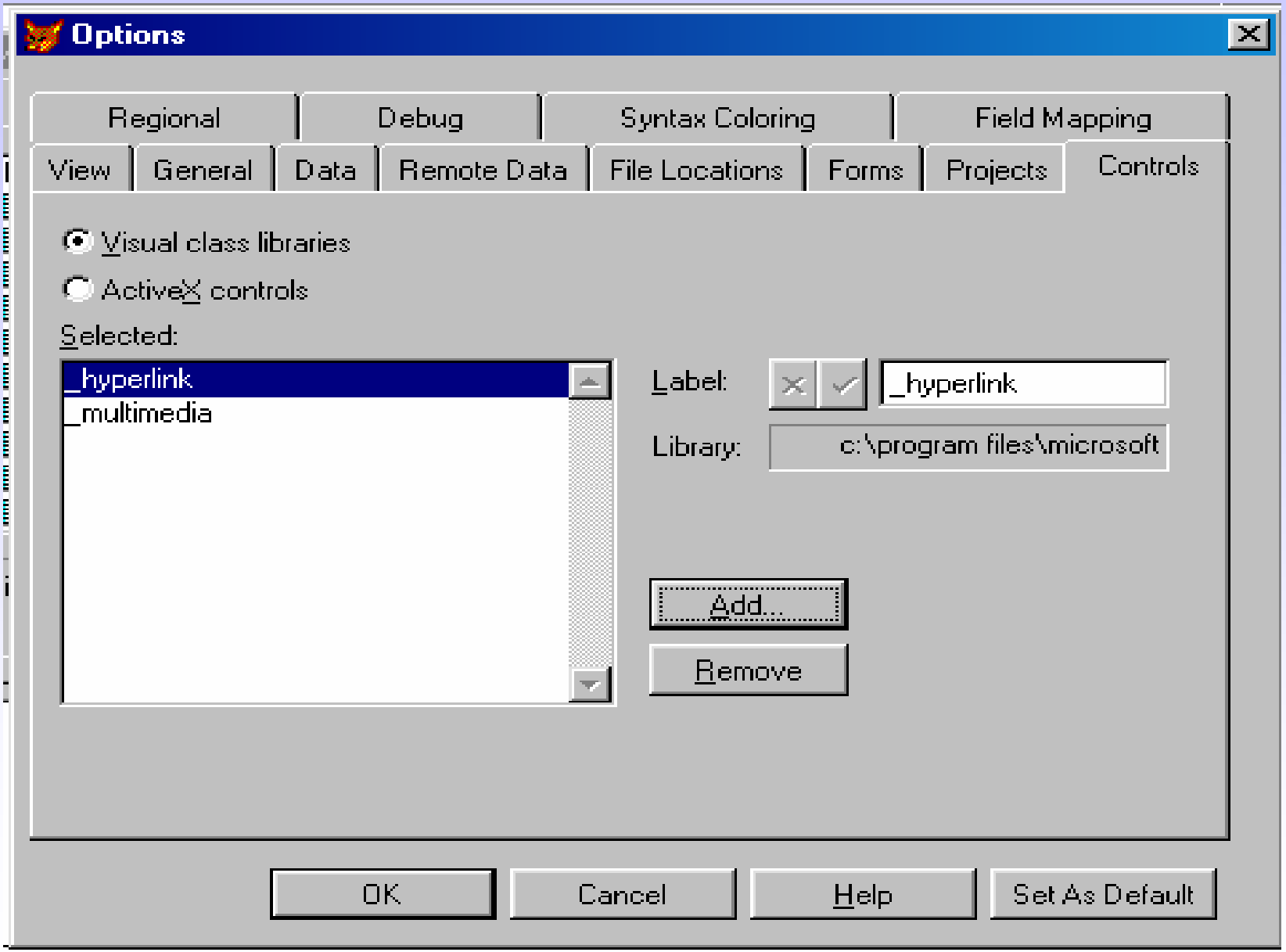
С этой целью, имя библиотеки необходимо добавить в список, открывающийся с помощью кнопки **View Classes.**

Если библиотека необходима только в текущем сеансе работы, достаточно после щелчка на пиктограмме **View Classes** ***выбрать опцию*** **Add** ***(добавить) и, выбрать файл библиотеки из каталога*** **\VFP9\FFC.**

В случае, когда компоненты библиотеки фундаментальных классов должны быть постоянно представлены на панели инструментов **Form Control**, следует выполнить следующие действия:

- В пункте Главного меню **Tools** выбрать команду **Options**.

  В окне диалога **Options** перейти на вкладку **Controls.**
- Установить переключатель в положение **Visual class libraries**.
- Нажать кнопку **Add** и с помощью **Open**-диалога добавить файл библиотеки, например, **_multimedia** в список **Selected**.
- Щелкнуть по кнопке **Set As Default** (назначить по умолчанию).
- Нажать кнопку ОК для закрытия окна **Options.**



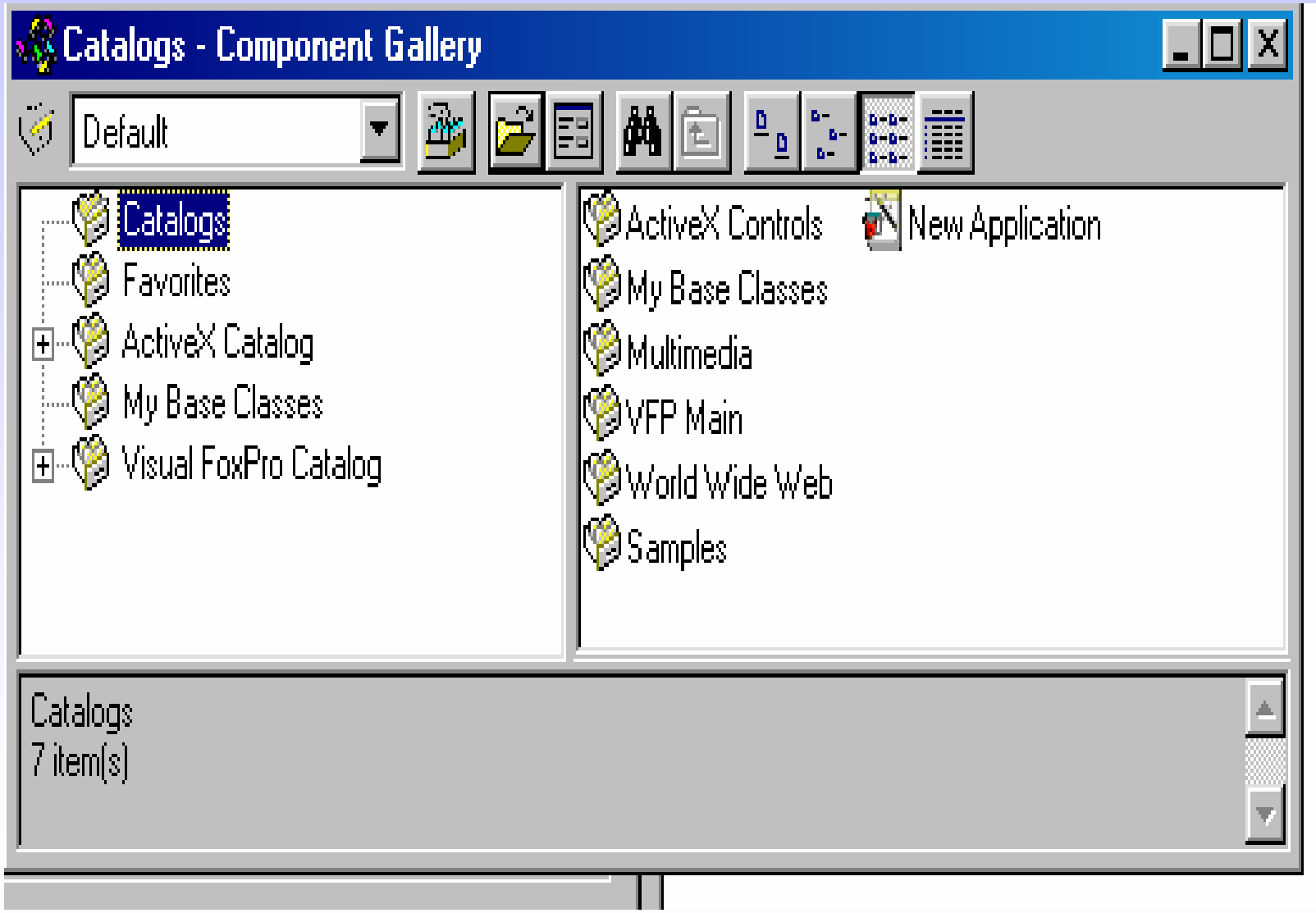
Вкладка Controls окна **диалога** Options

Component Gallery (Галерея компонентов)

Галерея компонентов - контейнер, в котором сосредоточены все повторно используемые ресурсы Visual FoxPro, например, классы, мастера, шаблоны.

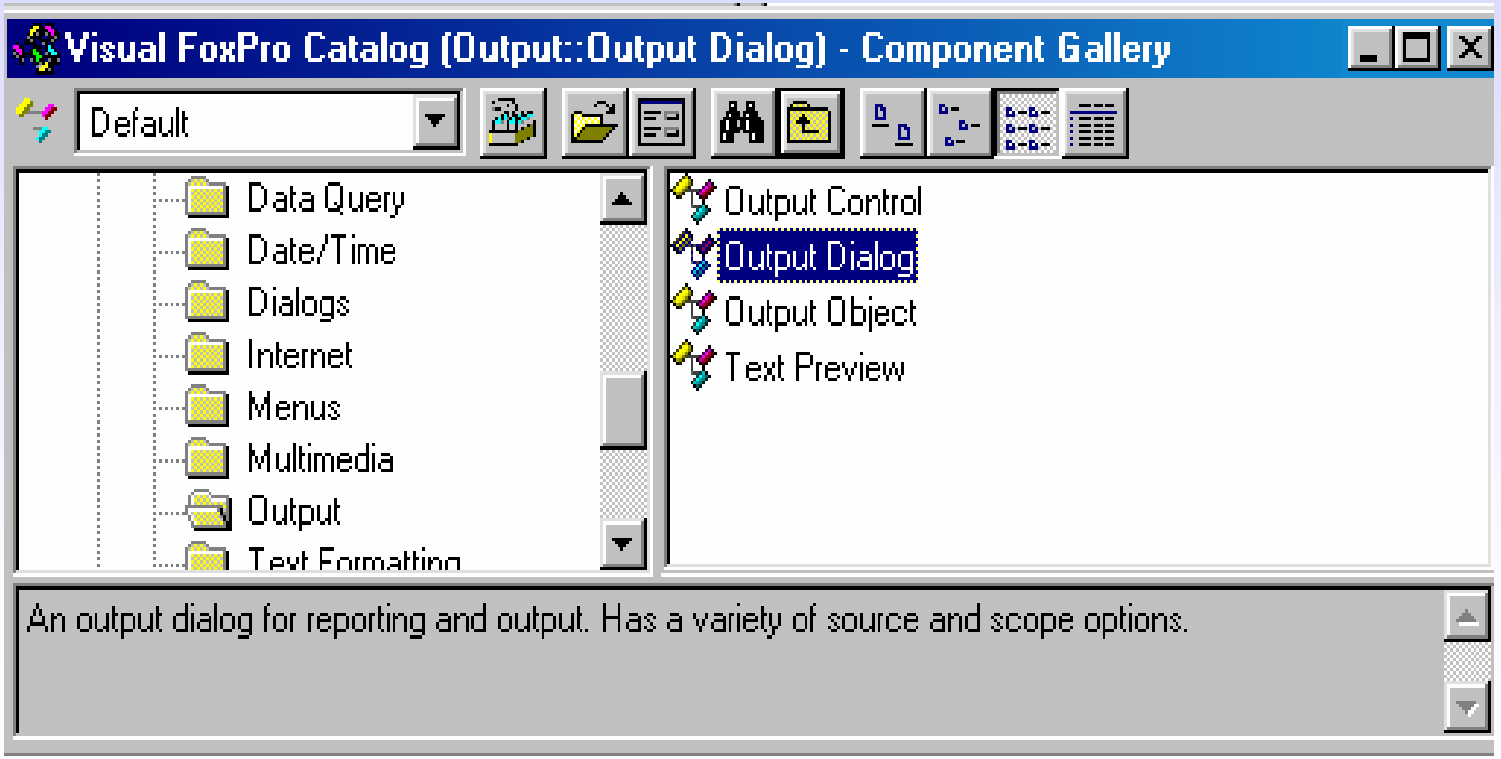
Использование Component Gallery обеспечивает удобный доступ к компонентам, а также позволяет создавать классы, производные от фундаментальных.

Получить доступ к компонентам Component Gallery можно с помощью команды Tools - Component Gallery. Структура окна Component Gallery напоминает проводник WINDOWS.



. Окно **Component Gallery**

После щелчка на папке категории в левой части окна, в правой - появится ее содержимое. После щелчка на пиктограмме класса на правой панели на панели в нижней части окна Component Gallery появится описание этого класса.



Просмотр описания выделенного класса

Значительная часть Component Gallery поставляется в составе Visual FoxPro. Сразу после установки Visual FoxPro на компьютер пользователя формируется система каталогов по умолчанию.

Самый простой способ вставить объект фундаментального класса в приложение - это перетащить пиктограмму класса с правой панели окна **Component Gallery** в проектируемую форму или окно **Project Manager**. ***Добавить объект фундаментального класса в форму или проект, можно с помощью контекстного меню выбранного класса.***

Класс производный от класса формы нельзя посадить на форму. Этот класс можно только добавить в проект (**Add to Project**) или создать объект новой формы на базе этого класса (**Create Form**).

Модель объектных компонентов Component Object Model (COM)

Терминология

COM – стандарт Microsoft, регламентирующий обмен информацией между приложениями. Стандартизация распространяется на передачу и прием сообщений. Благодаря этому, объекты, построенные различными языковыми средствами, могут взаимодействовать друг с другом.

Некоторые COM-объекты по своей природе являются визуальными, это так называемые элементы управления Active X.

Модель COM основана на объектно-ориентированном подходе. В рамках этой технологии можно напрямую обращаться к классам одного приложения из другого. Приложение COM-клиент запускает на выполнение код приложения COM-сервера. Здесь происходит перенос концепции специализации на уровень программных продуктов.
COM-объекты существуют в виде отдельных файлов (.exe, .dll) на диске и могут быть вызваны по сети с другого компьютера. Обычно доступная по сети программа выполняется на компьютере клиента (рабочей станции).

Существует более сложная модель, называемая распределенной моделью DCOM (Distributed COM). В этой схеме COM-объект существует на первом компьютере, использует ресурсы памяти и процессора другого компьютера, а результат можно получить на третьем. Преимуществом этой модели является увеличение числа обслуживаемых клиентов.

1.1. Механизм использования COM-объектов

В отличие от обычного ООП, при использовании COM-объектов пользователю не требуется создавать эти объекты самостоятельно. Вся работа сводится к освоению методов и свойств применяемого COM-объекта.

Чтобы получить ссылку на COM-объект, как и при создании обычного объекта (экземпляра класса), следует использовать функцию CreateObject(“имя класса”). Единственная сложность заключается в том, что требуется знать имя класса этого объекта.

Имя класса состоит из двух слов, первое – имя приложения, второе – имя класса внутри приложения. Этим имя класса COM-объекта отличается от имени класса Visual FoxPro, которое состоит из одного слова.

Для приложения Excel, например, это будет Excel.Application, для Word – Word.Application. Автоматизация OLE-объектов Windows (технология встраивания и связывания объектов) - это часть COM-модели.

Если из среды Visual FoxPro запускается другое приложение, то Visual FoxPro играет роль клиента, а другое приложение – роль сервера.

При установке на компьютере приложения COM-сервера в реестре Windows регистрируется как само приложение, так и все его классы.

При вызове функции CreateObject(“имя класса”) операционная система из реестра находит путь к вызываемому приложению. После запуска приложения-сервера можно обращаться к его методам и свойствам, так же как к методам и свойствам объекта Visual FoxPro.

Назначение полей данных таблиц типа General

Для хранения OLE-объектов в таблицах употребляются поля типа General. Поле типа General содержит десятибайтовую ссылку на действительное содержимое поля, созданное другим приложением: электронную таблицу, документ текстового процессора, изображение. Тип и объем реальных данных зависят от OLE-сервера, в котором создавался объект, и от того, какая технология применяется: связывание или внедрение OLE-объекта. Если применяется связывание, то в таблице содержатся только ссылки на данные и на приложение, в котором они были созданы. Если применяется внедрение OLE-объекта, то в таблице помимо ссылки на приложение, в котором данные созданы, содержится еще и копия данных.

Включение объекта OLE в таблицу

Для размещения объектов OLE в таблице, в нее необходимо включить поле типа General. После чего можно внести документы в таблицу, связав или внедрив их в поле General.

Поле типа General добавляют в структуру таблицы с помощью конструктора. После включения в структуру таблицы поля типа General, можно добавить в каждую запись объекты OLE.

Например, можно связать или внедрить документ Word в поле типа General или связать электронную таблицу Excel с формой.

Связывания и внедрение различаются местом хранения данных.

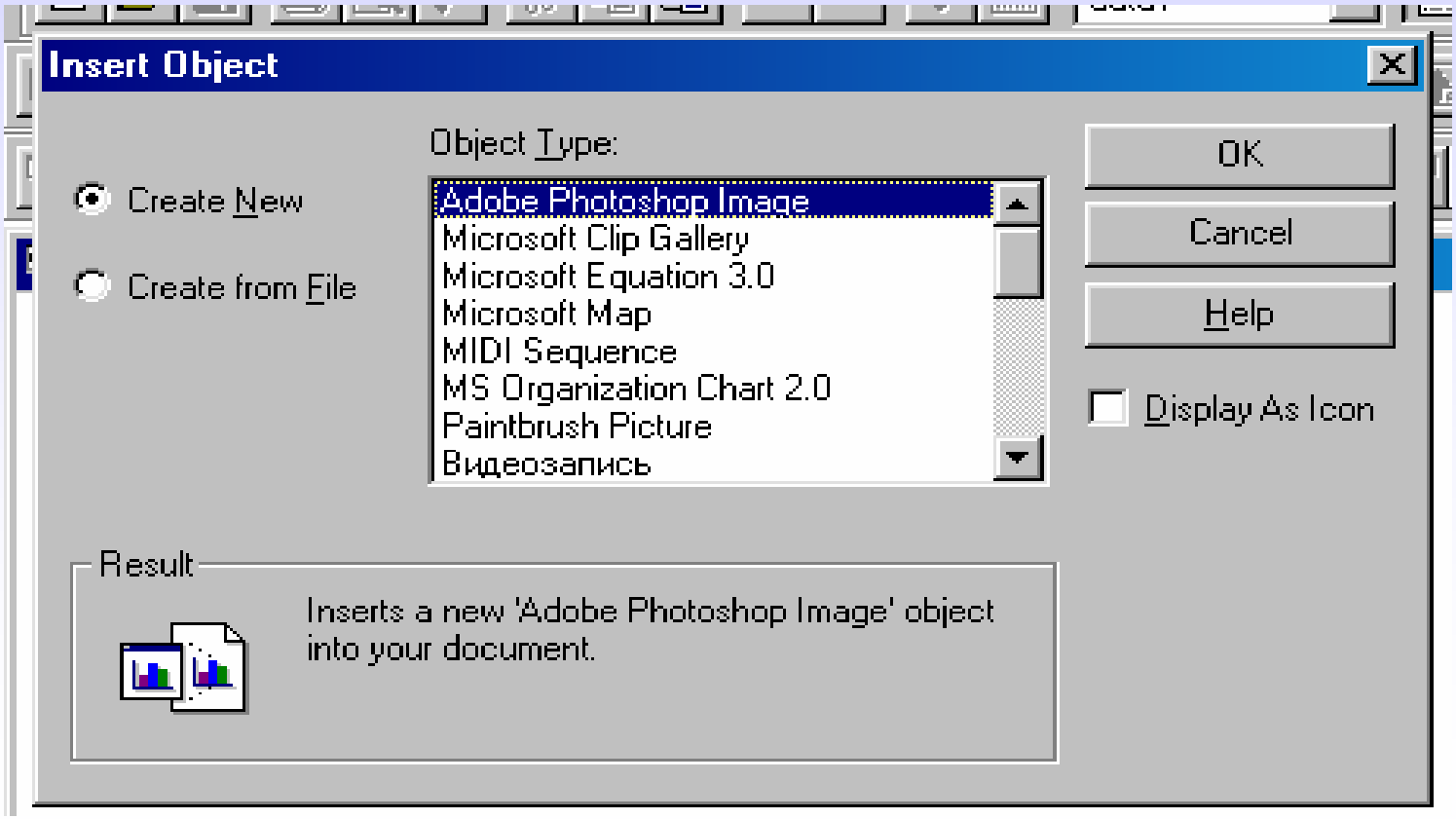
Связываемые данные хранятся в исходном файле, а не в таблице или форме* Visual FoxPro. *Сама таблица или форма хранит только расположение исходного файла и отображает представление связанных данных. Эти данные изменяются при модификации исходного файла и сохраняют соединение до тех пор, пока вы сами его не разорвете.

***Внедренные данные хранятся в вашей таблице или форме. Они не содержат соединение с исходным файлом. Изменения, вносимые в исходный файл, не отображаются в приложении* Visual FoxPro.**

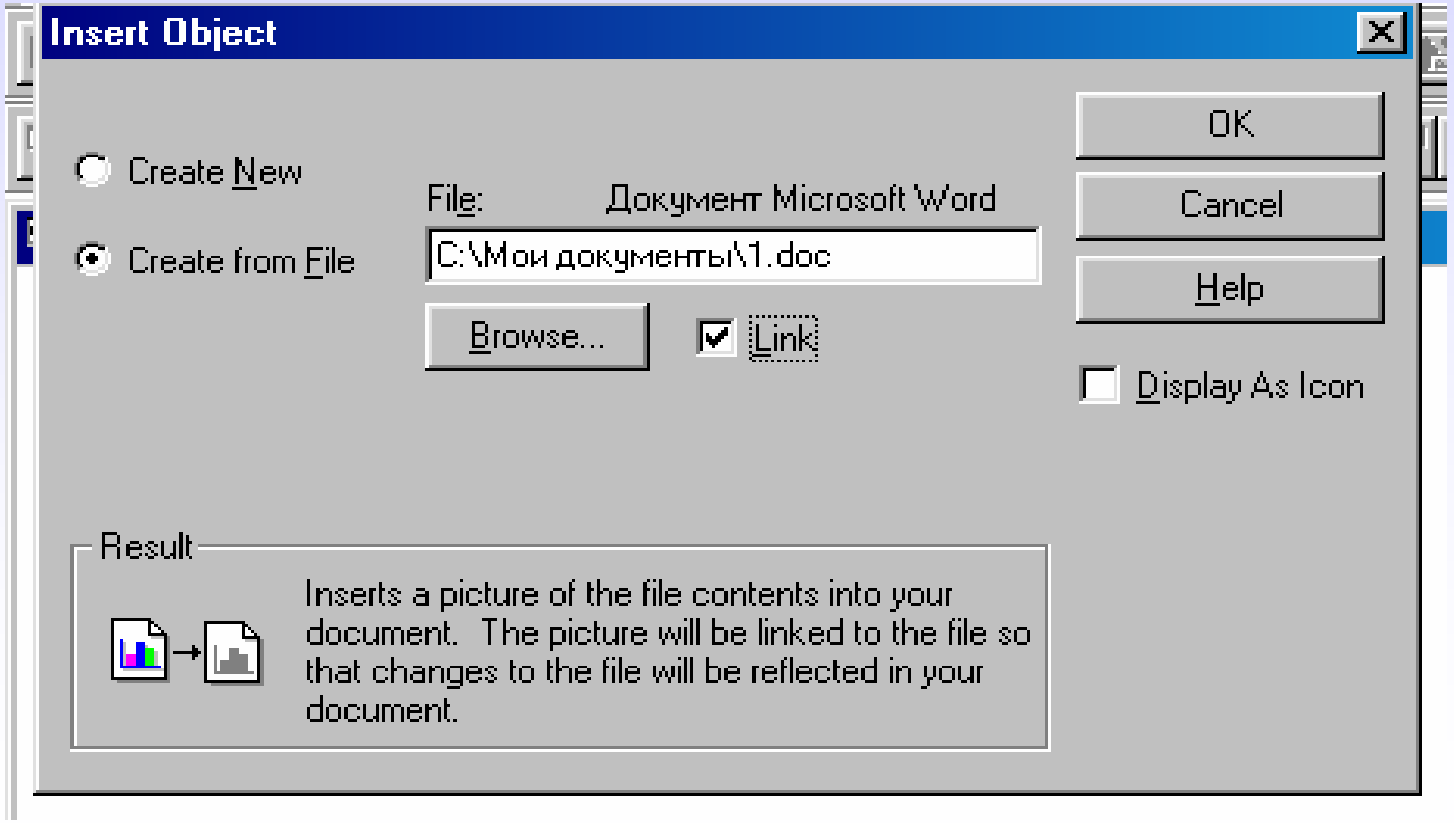
Для того чтобы включить данные из других приложений Windows (OLE-объекты) в таблицу Visual FoxPro, необходимо открыть и просмотреть соответствующую таблицу в режиме **Browse**, а затем выполнить следующие действия:

- Щелкнуть дважды поле типа **General**, в которое будут вставлены данные.
- В меню **Edit** выбрать команду **Insert Object**.
- На экране появится диалоговое окно **Insert Object** со списком данных, которые можно вставить в выбранное поле.

- Чтобы самостоятельно создать данные, надо выделить команду **Create New** и затем выбрать соответствующий тип данных из списка **Object Type**. После нажатия кнопки **OK** будет открыто окно выбранного приложения, в котором можно создать документ.



- Чтобы использовать существующие данные, следует установить параметр **Create from File**. Затем ввести имя файла, содержащего данные, или выбрать команду **Browse** и выделить нужный файл. Если предполагается установить связь с файлом, надо включить флажок **Link**.



- После закрытия окна приложения **Ole** -сервера **Ole-**объект будет вставлен в поле **General** одной записи таблицы.
- Чтобы включить данные из других приложений Windows (**OLE**-объекты) в другие записи таблицы Visual FoxPro, необходимо повторить перечисленные действия.

Включение объектов OLE в формы

В конструкторе форм можно включать в формы отдельные объекты OLE (элемент управления OLE Control). Кроме того, предусмотрена возможность отображения объектов OLE из полей типа General с помощью элемента управления OLE Bound Control.

Связать или внедрить отдельный Ole-объект в форму можно следующим способом:

- ***Создать или открыть форму в конструкторе.***
- ***На панели инструментов Form Controls нажать кнопку OLE Control и перетащить ее, чтобы поместить в форму.***
- ***На экране появится диалоговое окно Insert Object, выводящее типы файлов для связи и внедрения.***
- ***Создать новый файл, выделив Create New, или использовать уже существующий файл, выделив Create from File. Ввести имя файла, содержащего данные, или выбрать Browse и выделить нужный файл. Установить флажок Link, если необходимо установить связь с файлом.***

В случае, когда необходимо *отобразить объект* **OLE** из поля типа **General** *таблицы* Visual FoxPro, последовательность действий может быть следующей:

- В конструкторе форм включить в форму объект **OLE**, привязанный к полю таблицы (**OLE Bound Control**).
- Задать в свойстве объекта **ControlSource** имя поля типа **General**. Например, если имя таблицы Employee, а имя поля типа **General -** photo, то в качестве значения свойства **ControlSource** необходимо задать Employee.photo.
- Включить в форму кнопки или команды меню для просмотра поля типа **Genera**l, определенного в свойстве **ControlSource**.

Например, можно включить в форму класс **BUTTONS.VCX** из каталога Visual FoxPro SAMPLES\CONTROLS.

После запуска формы, нажатие кнопок позволит просмотреть содержимое поля **General**.

Если при заполнении данными таблицы поле **General** было оставлено пустым, то после запуска формы можно внести в них данные. С этой целью выбирают в процессе выполнения формы элемент управления **OLE Bound Control.** На экране появится диалоговое окно **Insert Object,** и новый объект будет сохранен в поле, к которому привязан данный элемент управления.

Компьютерные сети

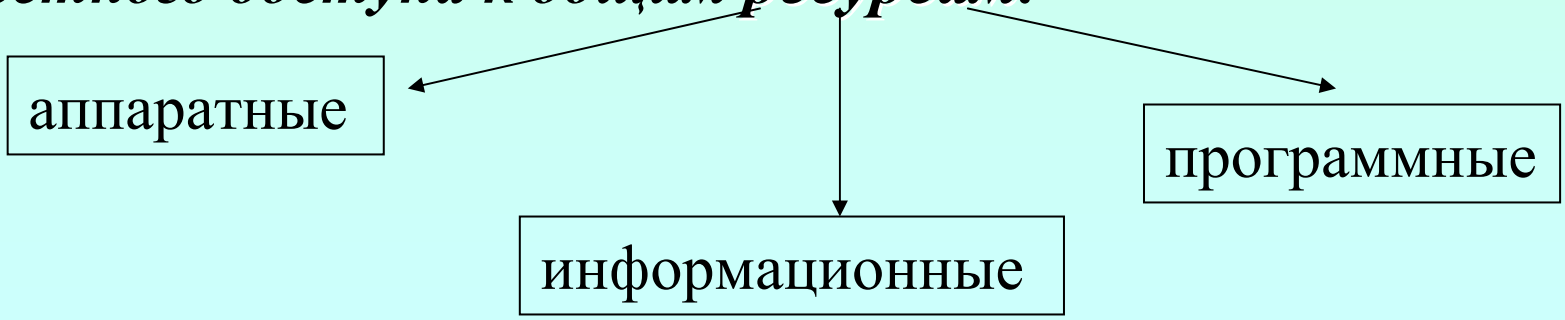
1. ***Назначение***
2. ***Основные понятия***
3. ***Классификация КС***
4. ***Политика сети***
5. ***Понятие Intranet***
6. ***Системное программное обеспечение КС***

Назначение

При физическом соединении двух или более компьютеров образуется компьютерная сеть (КС).

Для создания компьютерных сетей необходимо специальное аппаратное обеспечение (сетевое оборудование) и программное обеспечение (сетевые программные средства).

Назначение любых компьютерных сетей – обеспечение совместного доступа к общим ресурсам.



***Аппаратные* – принтеры, диски.**

Программные – используются, когда вычислительное задание отправляется на удаленную большую ЭВМ, а по окончании, результат возвращается обратно.

***Информационные* – данные, хранящиеся на удаленных компьютерах.**

В сети происходит совместное использование всех типов ресурсов.

Основные понятия

Протокол – стандарт для обеспечения в компьютерных сетях аппаратной и программной совместимости.

Аппаратные протоколы – характер аппаратного взаимодействия компонентов сети.

Программные – характер взаимодействия программ и данных.

Физически функции поддержки протоколов исполняют аппаратные устройства – интерфейсы
и программные средства – программы поддержки протоколов.

Эти программы часто тоже называют протоколами.

назад

Классификация КС

КС классифицируют по территориальному признаку или по масштабу сети.

По территориальному признаку выделяют: локальные сети* Local Area Network *(LAN) и глобальные* Wide Area Network *(WAN).

LAN – объединяют компьютеры, как правило одной организации, которые располагаются на расстоянии нескольких километров. Здесь используют новые высококачественные линии связи, поэтому время обращения к сетевым ресурсам соизмеримо со временем обращения к локальным ресурсам рабочих станций (PC).

WAN – объединяют компьютеры, располагающиеся на значительном расстоянии друг от друга (в любой точке земного шара).

При организации ***WAN***-сетей используются уже существующие линии связи, например, телефонные линии. Качество таких линий связи и скорость обмена данными существенно ниже, чем в ***LAN***-сетях.

По масштабу сети выделяют:

1. ***LAN рабочих групп.*** Объединяют небольшое число компьютеров сотрудников, работающих над одним проектом.
2. ***LAN отделов.***
3. ***Корпоративные сети.*** Объединяют компьютеры в рамках одного предприятия или корпорации. Такие сети могут охватывать любую часть земного шара и использовать любые современные средства связи.

Политика сети

У участников рабочих групп могут быть разные права для доступа к общим ресурсам сети.

Политика сети - совокупность приемов разделения и ограничения прав участников КС.

Администрирование сети - управление сетевыми политиками.

Системный администратор - лицо, управляющее организацией работы КС.

При подключении LAN к WAN важную роль приобретает понятие сетевой безопасности.

Должен быть ограничен доступ в ***LAN*** из вне для посторонних лиц, а также выход за пределы ***LAN*** сотрудников, не имеющих прав.

Для обеспечения сетевой безопасности между LAN и WAN устанавливают брандмауэры.

Это может быть специальный компьютер или программа.

Понятие Intranet

Intranet технология – обозначает применение служб глобальных сетей для целей LAN.

Процесс переноса технологий WAN в мир LAN получил в последнее время широкое распространение.

Системное программное обеспечение КС

Сетевая операционная система (ОС), установленная на отдельном компьютере состоит из следующих частей:

1. ***Средства управления локальными ресурсами компьютера.***
2. ***Средства предоставления собственных ресурсов в общее пользование - серверная часть*** (например, функции блокировки файлов и записей, обработку запросов удаленного доступа к собственной БД и т.д.).
3. ***Средства доступа к удаленным ресурсам – клиентская часть*** (направление запросов и прием ответов от серверов).

В зависимости от задач, решаемых с помощью сетевого компьютера, на него устанавливается определенный набор модулей сетевой ОС.

Сетевые компьютеры делятся на серверы и клиенты.

Клиент не предоставляет сети свои локальные ресурсы.

Он посылает запрос в сеть для получения доступа к сетевым ресурсам и обрабатывает ответ. ***Это ПК с локальной ОС и небольшим набором сетевых функций***.

Сервер обеспечивает совместный доступ к своим ресурсам.

Он обрабатывает запросы от клиентов и отправляет их обратно.

На компьютеры-серверы устанавливаются специальные серверные варианты сетевых ОС.

***Признанными лидерами сетевых ОС с выделенным сервером являются* Windows NT и Novell Net Ware**.

Первая включает как серверную, так и клиентскую часть, вторая - только серверную.

Novell Net Ware ***обеспечивает возможность доступа к файловому серверу со стороны рабочих станций, работающих под управлением различных* ОС: DOS, OS/2, Unix, Macintosh, Windows.**

На рабочей станции под управлением* Windows** ***должен быть установлен соответствующий сетевой клиент ОС Net Ware – Novell Client.

Конфигурация клиента может изменяться при помощи свойств сетевого окружения и свойств клиента.

Для получения на PC доступа пользователя к ***LAN*** необходимо зарегистрироваться - указать имя пользователя, пароль и другую информацию.

После успешной регистрации в Сетевом окружении появляются дополнительные устройства (диски, принтеры). В меню Пуск появляются ярлыки программ, расположенных на серверах.

При установленном клиенте в правом нижнем углу есть значок **N**, через который можно получить доступ к свойствам сетевого подключения пользователя.

Возможность изменения этих свойств зависит от прав, которые даны пользователю.

Полный набор сервиса по изменению параметров сетевого подключения пользователя, и не только, предоставляет специальная утилита Net Ware Administrator (Сервер инф. технологий www.citforum.ru).

Интернет

1. ***Основные понятия***
2. ***Система доменных имен DNS***
3. ***Службы Интернета***
4. ***Язык разметки гипертекста HTML***
5. ***URL – адрес***
6. ***Вопросы компьютерной безопасности***
7. ***Основные виды нарушения режима сетевой безопасности***
8. ***Несколько способов защиты от нарушения сетевой безопасности***

Основные понятия

В дословном переводе internet это межсеть – объединение сетей. Сейчас под этим термином подразумевают Всемирную компьютерную сеть.

Стек протоколов TCP/IP – это два сетевых протокола, лежащих на разных уровнях.

Протокол TCP – протокол транспортного уровня. Он управляет процессом передачи данных и устанавливает логическое соединение.

Протокол IP – адресный протокол, определяет куда происходит передача данных.

Согласно протоколу TCP отправляемые данные нарезаются на небольшие пакеты. Каждый пакет маркируется, чтобы в нем были данные для правильной сборки на компьютере получателя.

Согласно протоколу IP каждый участник Всемирной сети должен иметь свой уникальный IP - адрес. Он состоит из четырех байтов и записывается в виде четырех десятичных чисел, разделенных точками,

например, 194.85.120.66

IP-адрес состоит из двух логических частей: номера сети и номера узла в сети.

Номер в сети Интернет выдает специальное подразделение – InterNIC (Internet Network Information Center).

Номер узла определяет администратор сети.

Структура IP-адреса организована так, что каждый компьютер, через который проходит какой-либо TCP-пакет, может определить какому из ближайших соседей надо переслать пакет, чтобы он оказался «ближе» к получателю.

Выбор маршрута осуществляют аппаратные средства – маршрутизаторы. Учитывается не только расстояние, но и качество линий связи.

Система доменных имен DNS

Человеку удобнее иметь дело не с числовыми ***IP***-адресами, а с символьными именами. ***Для замены числовых IP-адресов символьными, используется система доменных имен. Младшая часть доменного имени соответствует конечному узлу сети. Составные части отделяют друг от друга точкой. Совокупность имен, у которых несколько старших частей доменного имени совпадает, называется доменом.***

mail.econ.pu.ru

www.econ.pu.ru

Самым главным является корневой домен. Далее следуют домены первого, второго и третьего уровней. Корневой домен управляется InterNIC.

Корневые домены назначаются для каждой страны (это 2 или 3 буквенные аббревиатуры) – ru, us.

Несколько имен корневых доменов закреплено для различных типов организаций:

.com – коммерческие (ibm.com);

.edu – образовательные (spb.edu);

.gov – правительственные (loc.gov);

.org – некоммерческие (w3.org);

.net - поддерживающие сети (rinp.net).

Для каждого имени домена создается свой DNS-сервер, который хранит БД соответствий IP-адресов и доменных имен, расположенных в данном домене, а также ссылки на DNS-сервер нижнего уровня.

Службы Интернета

Служба – пара программ, взаимодействующих между собой по определенным правилам, называемым протоколами.

Одна из программ – сервер, вторая – клиент.

Разные службы имеют разные протоколы.

Чтобы воспользоваться одной из служб Интернета, необходимо установить на компьютере клиентскую программу и подключить ее к серверной программе.

Терминальный режим (Telnet) – удаленное управление компьютером (телескопами, видеокамерами).

Электронная почта (e-mail).

Служба передачи файлов (FTP). Необходимость в ней возникает при приеме файлов программ, пересылке крупных документов, или архивных файлов. Большинство броузеров WWW (Explorer) реализуют простейшие операции протокола FTP.

Служба IRC (Internet Relay Chat) – прямое общение нескольких человек в режиме реального времени.

Служба ICQ (акроним I seek you) – Интернет пейджер. Поиск сетевого IP-адреса человека, подключенного в данный момент к Интернету. Постоянные IP- адреса имеют только компьютеры, постоянно включенные в Интернет. Большинство пользователей не имеют постоянного IP-адреса. Им выдается динамический IP-адрес, действующий в течении данного сеанса. Этот адрес выдает сервер, через который происходит подключение.

Для пользования этой службой надо зарегистрироваться на ее центральном сервере (http://www.icq.com) и получить персональный идентификационный номер UIN, который можно сообщить партнерам.

Зная номер UIN-партнера, можно через центральный сервер службы отправить ему предложение установить соединение.

Служба World Wide Web (WWW). Эту службу нередко отождествляют с Интернетом, хотя это одна из служб.

WWW – единое информационное пространство, состоящее из взаимосвязанных электронных документов, хранящихся на Web-серверах. Отдельные документы называют Web-страницами. Группы тематически объединенных Web-страниц называют Web-узлами.

Один Web-сервер может содержать много Web-узлов.

***Web-сервер* – программа, обеспечивающая доступ к ресурсам компьютера-сервера**. Задача Web-сервера – переадресация запроса клиентского броузера нужному Web-приложению.

Web-приложение – программа, которая в ответ на запрос формирует текстовую страницу в формате HTML и передает ее серверу. Сервер посылает эту страницу клиенту. Web-приложения часто называют модулями расширения сервера.

***Web-броузер* - это программа- клиент, которая преобразует описание полученной от сервера страницы в ее видимое изображение**.

В мире доминирует броузер MS Internet Explorer (IE), входящий в поставку Windows. Пользователи других ОС предпочитают Netscape Communicator. Часто используется броузер Opera норвежского производства.

Язык разметки гипертекста HTML

От обычных текстовых документов ***Web-страницы*** отличаются отсутствием привязки к бумажному носителю (т.е.параметров страницы). Они предназначены для просмотра на экране компьютера. Оформление ***Web-***документа выполняется во время его воспроизведения на компьютере клиента.

Броузеры отображают документы на экране в соответствии с командами, внедренными в его текст.

Такие команды называют тегами.

Правила записи тегов содержатся в спецификации языка HTML (Hyper Text Markup Language) – язык разметки гипертекстов.

Документ, размеченный такими тегами, имеет расширение .htm.

Теги представляют собой английские слова, обрамленные угловыми скобками. Большинство тегов используются парами: открывающий и закрывающий. Закрывающий тег начинается с символа «/». Документ разделен на две неравные части – заголовок (теги <HEAD> и </HEAD>) и тело (<BODY> и </BODY>).

Если опустить заголовок, вместо него в строке заголовка броузера окажется имя файла HTML-страницы.

Теги **
** и **<P>** завершают абзац, поэтому не имеют парных тегов. Тег **<P>** завершает абзац и вставляет пропуск перед следующим абзацем.

Например,

<HEAD>

<TITLE> Простой пример документа < /TITLE >

</HEAD>

<BODY>

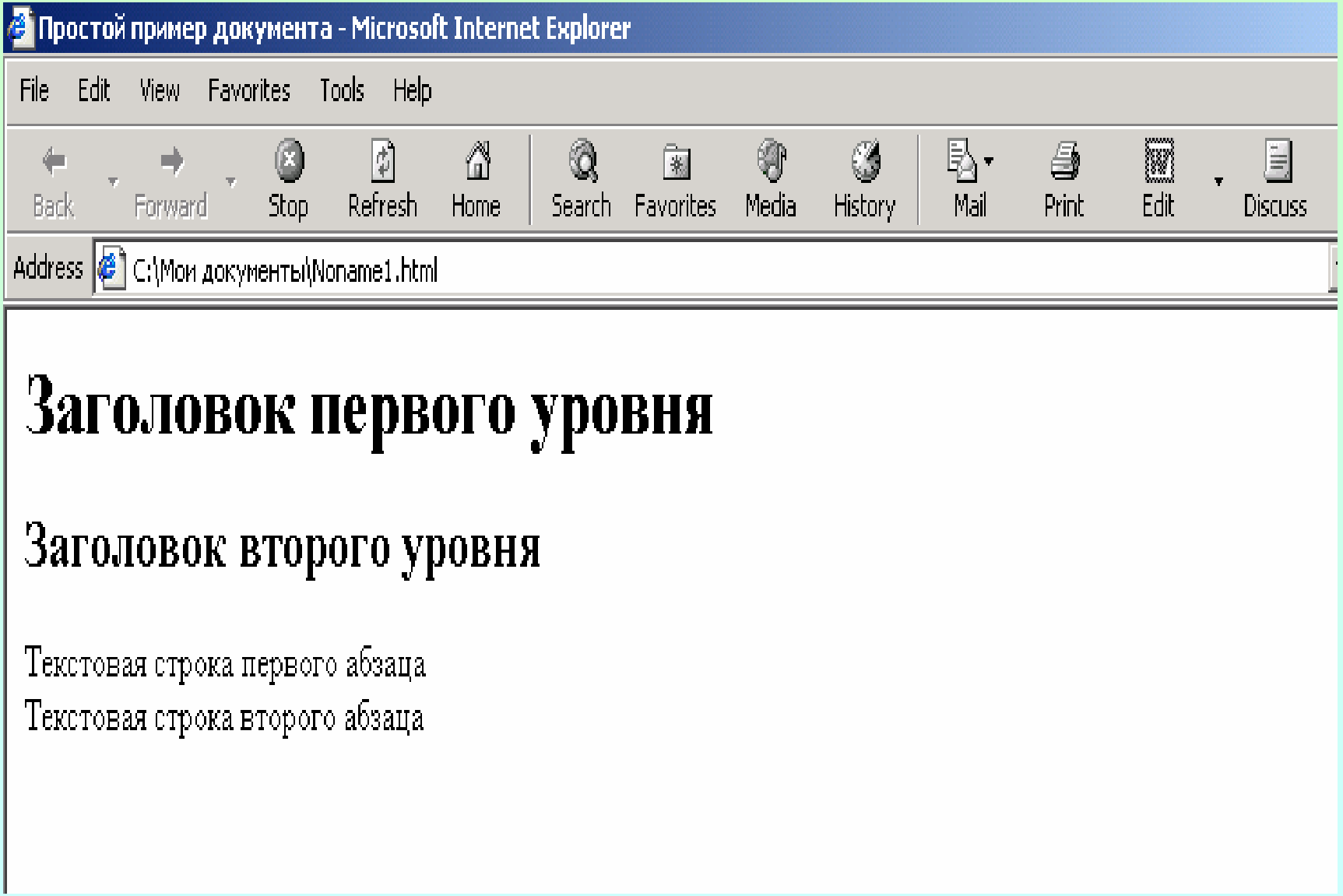
<H1> Заголовок первого уровня </H1>

<H2> Заголовок второго уровня </H2>

Текстовая строка первого абзаца ***
***

Текстовая строка второго абзаца

</BODY>


Простой пример документа - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address C:\Мои документы\Noname1.html
Заголовок первого уровня
Заголовок второго уровня
Текстовая строка первого абзаца
Текстовая строка второго абзаца

Сложные теги, кроме ключевого слова имеют дополнительные атрибуты и параметры.

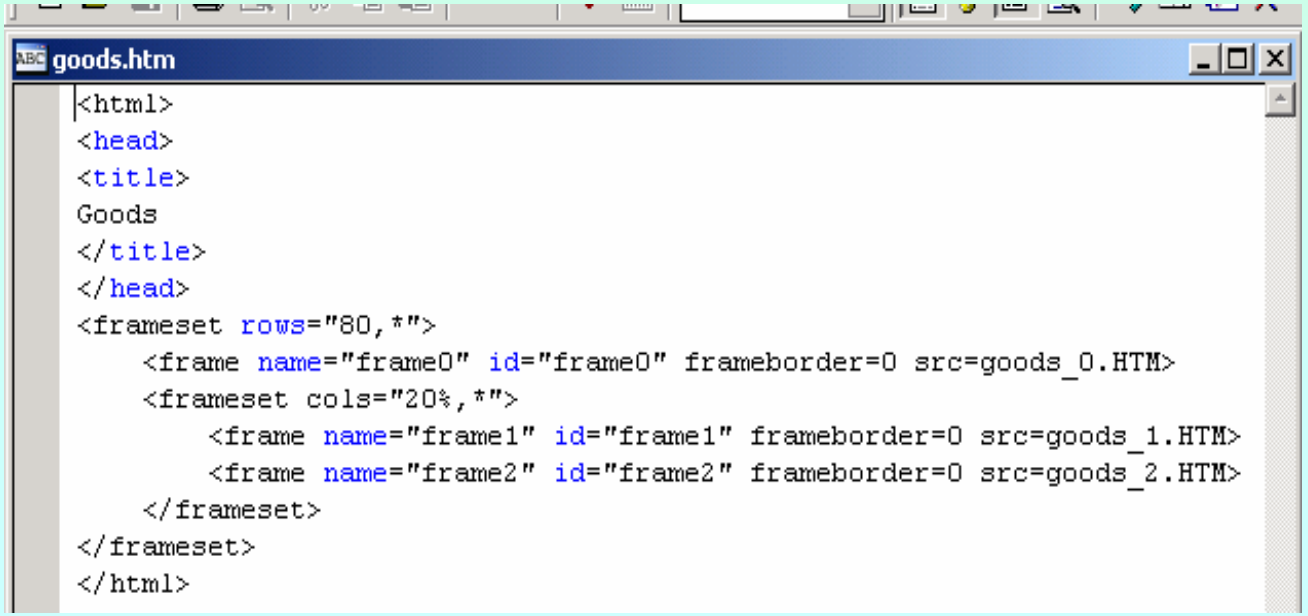
Например, шрифт

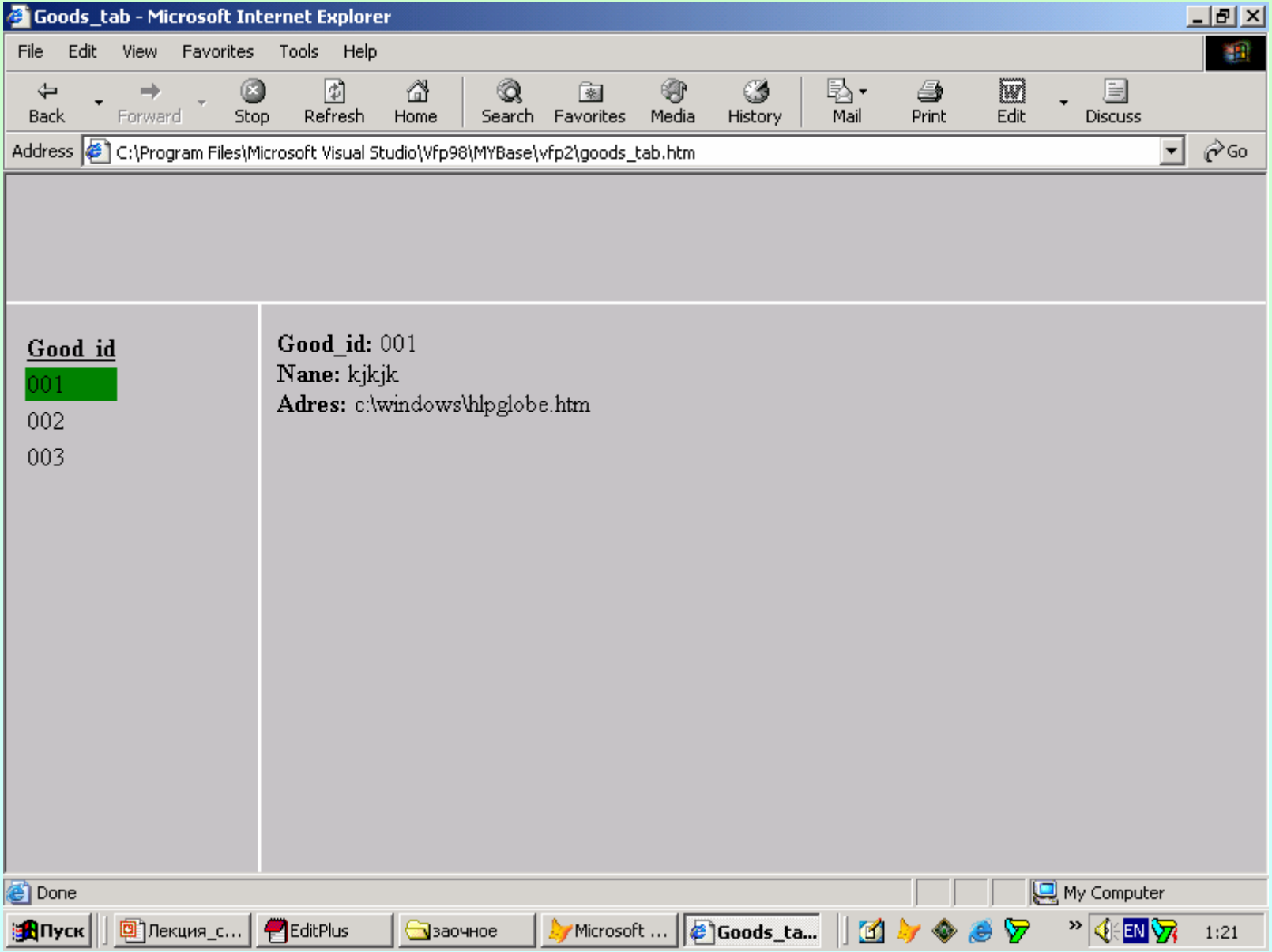
<FONT SIZE=5; COLOR=Green>

Текст

***</FONT>
***

<u>Фреймы.</u> Фрейм – рамка. Они позволяют разделить страницу на отдельные фрагменты. На странице с фреймами вместо тегов <Body> используются теги <FRAMESET> и <FRAME>. Первый тег описывает структуру страницы: количество горизонтальных (row) и вертикальных (col) полос и их оформление. Тег <FRAME> обязательно содержит параметр SRC с помощью, которого создается страница (источник).

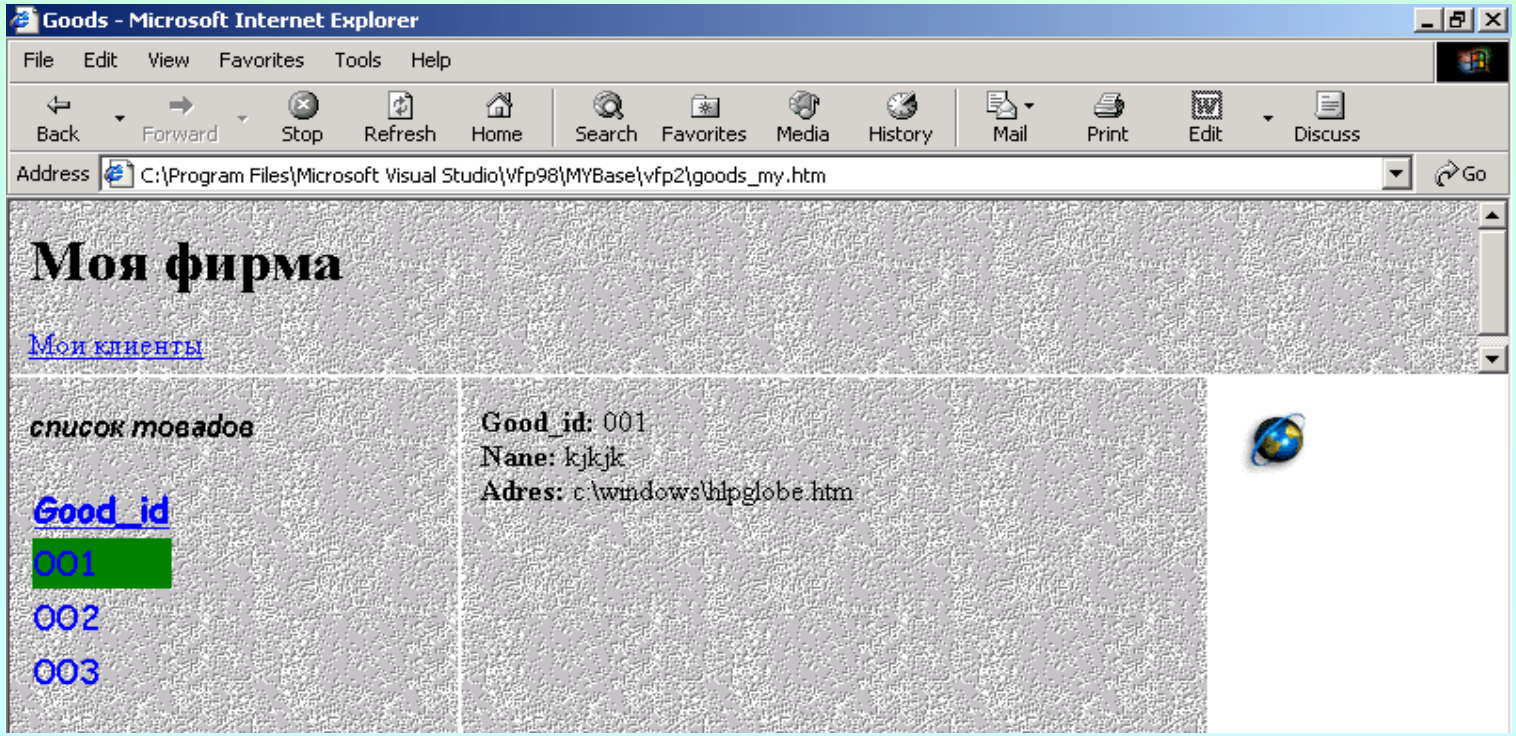



Goods_tab - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address C:\Program Files\Microsoft Visual Studio\Vfp98\MYBase\vfp2\goods_tab.htm
Go
Good_id
001
002
003
Good_id: 001
Nane: kjkjk
Adres: c:\windows\hlpglobe.htm
Done
My Computer
Пуск
Лекция_с...
EditPlus
заочное
Microsoft ...
Goods_ta...
EN
1:21

Существуют специальные теги для внедрения графических и мультимедийных объектов (звук, музыка, видеоклипы).

Встретив такой тег, броузер делает запрос на доставку файла и воспроизводит его.

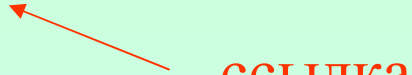
<IMG SRC="имя.gif" BORDER=1 ALIGN=LEFT>



Наиболее важной чертой Web-страниц являются гипертекстовые ссылки. С любым фрагментом текста с помощью тегов можно связать другой Web-документ, т.е. установить гиперссылку.

<A HREF="имя файла" > текст

</A>



<u>URL - адрес</u>

Возможность использования гиперссылок обеспечивается тем, что каждый документ в гипертекстовом пространстве обладает уникальным адресом.

Адрес любого файла во всемирном масштабе определяется унифицированным указателем ресурса – URL.

Адрес URL состоит из трех частей:

1. ***Наименование службы, обеспечивающей доступ к файлу (обычно имя протокола данной службы).***

Для WWW – протокол HTTP (Hyper Text Transfer Protocol – протокол передачи гипертекста). После имени протокола ставится “:” и два знака “/”

http://

2. ***Доменное имя сервера, на котором хранится ресурс***

http://www.abcde.com

3. ***Полный путь доступа к файлу на сервере***

(в качестве разделителя используется символ “/”)

http://www.abcde.com**/files/new/index.htm**

При записи URL адреса важно точно соблюдать регистр символов.

В отличие от правил работы в Windows, в Интернете строчные и прописные символы считаются разными.

Вопросы компьютерной безопасности

Это понятие включает надежность *работы компьютера, сохранность ценных данных, защиту информации от внесения несанкционированных изменений, сохранение тайны переписки в электронной связи.*

Надежность работы компьютерных систем основана *на мерах самозащиты.*

Основным средством защиты информации является резервное копирование наиболее ценных данных.

Защита информации в Интернете

Насколько ресурсы Всемирной сети открыты каждому клиенту, настолько же и ресурсы компьютерной сети клиента могут быть открыты для людей, обладающих необходимыми средствами.

Пользователи Интернета не должны нарушать законодательства тех стран, на территории которых расположены серверы Интернета.

Как и в локальной сети, при работе во Всемирной сети абсолютно все действия протоколируются и данные сохраняются.

Клиент и сервер в сети работают совместно по установленным правилам, закрепленным в протоколе.
Те операции, которые выходят за рамки утвержденных протоколов, считаются небезопасными. Если эти операции, к тому же выполняются несанкционированно, то они считаются запрещенными. Они образуют состав административного или уголовного правонарушения.
Новые протоколы создаются редко, а старые действуют иногда десятки лет. За это время возникает разрыв между техническими возможностями сети и тем, что разрешено устаревшими протоколами.

Серверы предлагают множество различных расширений и дополнений к броузеру.

Любое расширение свойств клиентской программы сопровождается определенным отходом от стандартного протокола и может сопровождаться какой-то угрозой.

Поэтому вопросы сетевой безопасности находятся на грани между желаемым и позволенным. Чем сложнее система, тем труднее в ней контроль за должным уровнем безопасности.

Стандартные сетевые средства Windows 98 не обеспечивают никакого уровня безопасности. При корпоративном подключении к Интернету служебных компьютеров необходимо принимать специальные меры (труд квалифицированных системных администраторов), либо обеспечить на подключаемых отсутствие служебной информации, либо использовать другие ОС (Unix, Linux).

Основные виды нарушения режима сетевой безопасности

1. ***Угроза удаленного администрирования.***

Это несанкционированное управление удаленным компьютером. Например, копирование и модифицирование данных, установка вредоносных программ.

2. ***Угроза активного содержимого. Активное содержимое – активные объекты, встроенные в Web-страницы. Эти объекты включают программный код, который на компьютере жертвы может работать как средства удаленного администрирования, либо производить разрушение данных.***

3. ***Угроза перехвата или подмены данных на путях транспортировки.*** Например, данные об платежных картах могут быть перехвачены при электронных платежах.

4. Угроза вмешательства в личную жизнь. В целях коммерческих интересов рекламных организаций собираются персональные сведения о клиентах (интересы, вкусы и т.д.).

5. Угроза поставки неприемлемого содержимого (сточки зрения морально –этических и религиозных норм.

Несколько способов защиты от нарушения сетевой безопасности

Удаленное администрирование.

- ***Ограничить доступ посторонних лиц к сетевым компьютерам (парольная защита).***
- ***Не запускать программы, пришедшие по электронной почте , независимо от письма.***
- ***Не отправлять программы по E-mail.***
- ***Не устанавливать сетевые служебные программы со сборников на CD.***

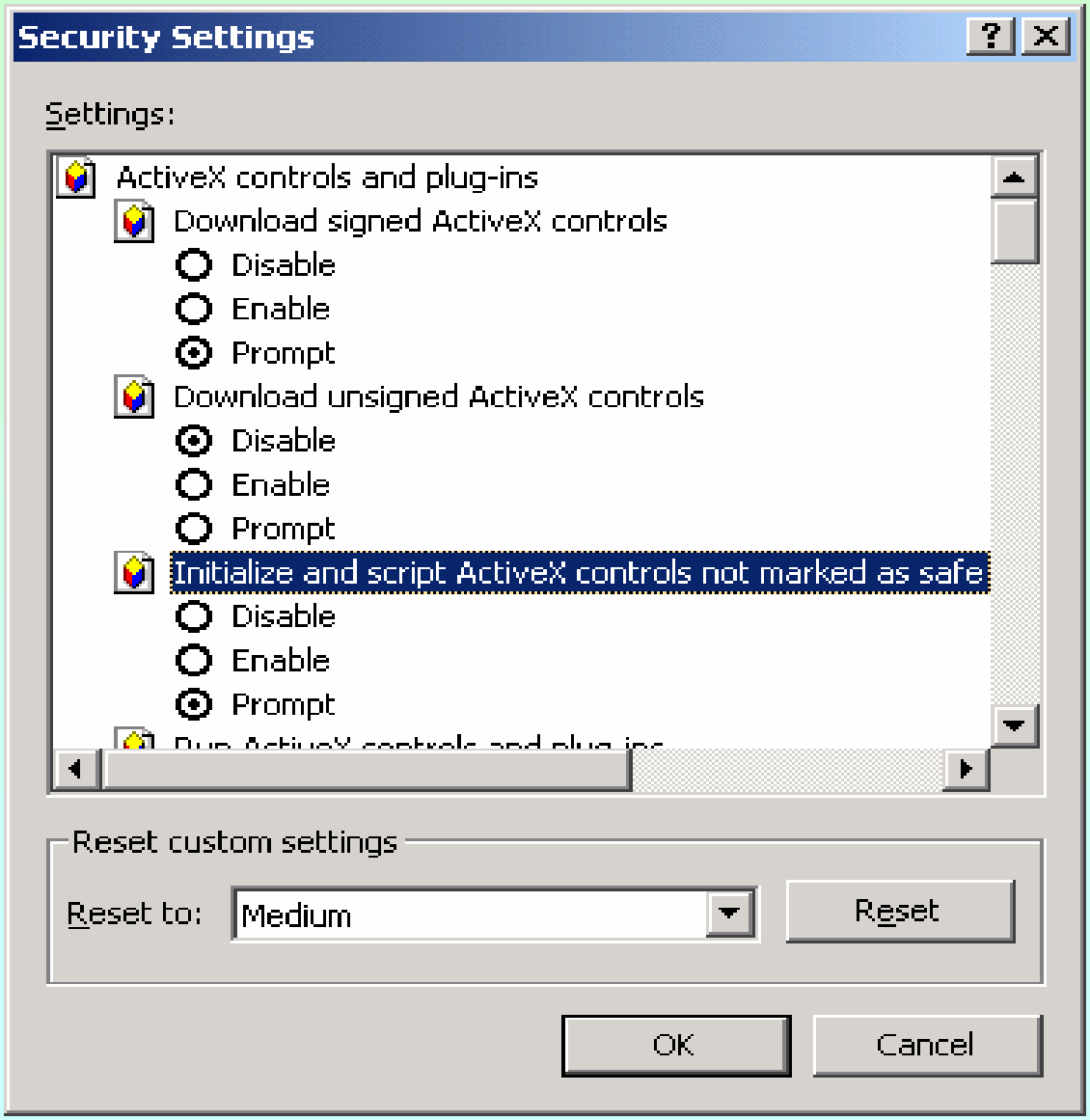
Активное содержимое.

•***Отключить прием Java- апплетов, элементов ActiveX и активных сценариев.*** В каждом конкретном случае можно запрашивать необходимость отключения. Решение принимают в зависимости от надежности узла.

Защита данных на путях транспортировки
Используется идентификация партнеров с помощью криптографических методов .

Источником персональной информации о клиенте иногда являются маркеры cookie. Эти маркеры сервер устанавливает для идентификации компьютера клиента во время текущего сеанса. Они должны быть временными, хранящимися в оперативной памяти. Почти всегда они переносятся на жесткий диск. При последующих сеансах они считываются в ОЗУ и предъявляются установившим серверам (могут считываться и другими серверами).

Explorer – Сервис – Свойства обозревателя – Безопасность



Контрольные вопросы

Проектирование реляционной базы данных

1. Назовите компоненты информационных систем.
2. Какие Вы знаете варианты архитектуры БД?
3. Назовите модели данных, используемые при разработке БД.
4. Сформулируйте цель нормализации таблиц БД.
5. Назовите три нормальные формы R-таблиц.

Навигационный доступ к данным таблиц

1. На чем основан навигационный доступ к данным?
2. Какая команда добавляет пустую запись в таблицу?
3. Каким образом при выполнении пункта 6 лабораторной работы перемещается указатель на запись?
4. Назовите команды перемещения указателя на запись.
5. Приведите синтаксис команд, использованных для поиска данных.
6. Как проверить результат выполнения поиска данных в поле таблицы?
7. В какой записи осуществляет замену команда REPLACE?
8. Как называется таблица, полученная в пункте 13 лабораторной работы?
9. Какая команда позволяет создать индекс?
10. Назовите назначение команды SET ORDER TO <индекс>.
11. Как установить связь между указателями записей двух таблиц в отношении «многие-к-одному» ?
12. Как установить связь между указателями записей двух таблиц в отношении «один-ко-многим» ?
13. Как просмотреть данные из полей разных тблиц?

Реляционный доступ к данным таблиц

1. Назовите формат и назначение команд языка определения данных (DDL), использованных в работе.
2. Назовите формат и назначение команд языка определения данных (DML), использованных в работе.

Доступ к удаленным данным в среде Visual FoxPro

1. Как проверить, установлен ли драйвер источника данных на вашем компьютере?
2. Как создать новый источник удаленных данных?
3. Опишите последовательность создания удаленного представления с использованием данных приложения файл-сервера.
4. Что означают опции, заданные на вкладке Update Criteria?
5. Можно ли установить постоянные отношения между таблицами в окне конструктора удаленных представлений?
6. Как создать параметризованный критерий запроса?
7. Каково назначение функции SQLConnect()?
8. Какие три параметра принимает функция SQLExec()?
9. Какой символ в инструкции SELECT обозначает выбор всех полей из таблицы?

Типы блокировок в Visual FoxPro

1. Назовите четыре уровня блокировки данных в VFP.
2. Какие типы блокировок применимы для баз данных и таблиц?
3. Какие ключевые слова используются для задания типа блокировки баз данных и какие операции соответственно будут заблокированы?
4. Какие ключевые слова используются для задания типа блокировки таблиц и какие операции соответственно будут заблокированы?
5. Каким образом можно заблокировать несколько записей в таблицах?

6. Как заблокировать заголовок таблицы и какие операции будут при этом недоступны?
7. Какие функции позволяют определить, заблокировал ли пользователь таблицу или запись соответственно, данные какого типа они возвращают и как можно просмотреть результат?
8. Какие операции с данными вызывают скрытую блокировку записей?
9. Назовите способы снятия блокировок.

Создание функционального класса в среде Visual Fox Pro

1. Что означают термины «класс» и «объект»?
2. Что такое «инкапсуляция» и «наследование»?
3. Опишите последовательность создания класса с помощью конструктора.
4. Какие методы выполняются при инициализации объекта и при потере объектом фокуса ввода?
5. Какие команды создают экземпляр объекта и делают его видимым?
6. Какая команда уничтожает объект?
7. Какая команда служит для определения программного кода класса?

Использование OLE-объектов в среде Visual Fox Pro

1. Какой тип данных используется для помещения в поле таблицы объекта, созданного приложением-сервером?
2. Как поместить объект, созданный OLE-сервером в поле таблицы?
3. Какие объекты используются для изображения на форме полей типа General из таблицы?
4. Как «посадить» на форму кнопки для перемещения по записям?
5. Какое свойство управляет режимом просмотра OLE-объекта на форме?
6. Чем режим «встраивания» Ole-объекта отличается от режима «связывания» и как они устанавливаются?
7. Что такое ***Component Gallery?***
8. Какова структура окна ***Component Gallery?***
9. Что такое **ActiveX**-компонент?
10. Производные каких классов можно добавить только в проект, а не на форму?
11. Можно ли добавить на форму объект **Splash?**
12. Как добавить компонент из ***Component Gallery*** на фому?
13. Объясните механизм использования COM-объектов.

Создание WEB-страницы в средеVisual Fox Pro

1. Как в среде VFP6.0 создать объект URL Open Dialog.
2. Назовите шаги мастера Web Publishing.
3. Как добавить фрейм на Web-страницу.
4. Как создать Web-страницу с данными таблицы с помощью фундаментального класса _dbf2html.
5. Какие тэги используется для создания изображения и гиперссылки в HTML-документе?

Библиографический список

1. Омельченко Л.Н. Самоучитель Visual FoxPro 7.0. – СПб. : БХВ-Санкт. Петербург, 2002. – 672 с.
2. Менахем Базиян и др. Использование Visual FoxPro 6.0: Полное справочное руководство. Специальное издание : пер. с англ. – М. : Вильямс, 2000. – 928 с.
3. Пэддок Р., Петерсен Д., Тэлмейдж Р. Visual FoxPro 6. Разработка корпоративных приложений. – М. : ДМК, 1999. – 592 с.
4. Мусина Т.В., Пушенко В.А. Visual FoxPro 7.0: Учебный курс. – К.: ВЕК+, Киев : BookStar, 2001. – 400 с.
5. Каратыгин С.А., Тихонов А.Ф., Тихонова Л.Н. Visual FoxPro 7. – М. : Бином-Пресс, 2003 г. – 656 с.
6. Visual FoxPro 9 / Лебедев А.Н. – М. : НТ Пресс, 2005. – 328 с. : ил. – (Самоучитель).
7. Семенова И.И. Разработка баз данных в Microsoft Visual FoxPro : Часть 1. Создание структуры базы данных : учебно-методическое пособие. – Омск : Изд-во СибАДИ, 2006. – 63 с.
8. Гурвиц Г.А. Разработка реального приложения с использованием Microsoft Visual FoxPro 9 : учеб. пособие. – Хабаровск : Изд-во ДВГУПС, 2007. – 198 с.
9. Клепинин В., Агафонова Т. Visual FoxPro 9: Наиболее полное руководство в подлиннике. – СПб. : BHV, 2008. – 1210 с.

Учебное издание

Молчанова Елена Ивановна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ЭКОНОМИКЕ

Электронный курс лекций

Издано в авторской редакции
Компьютерный набор *Е.И. Молчанова*

План 2012 г.