И.И. Семенова

РАЗРАБОТКА БАЗ ДАННЫХ В MICROSOFT VISUAL FOXPRO

Часть 1

СОЗДАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

Омск – 2006

Федеральное агентство по образованию
Сибирская государственная автомобильно-дорожная академия
(СибАДИ)

И.И. Семенова

РАЗРАБОТКА БАЗ ДАННЫХ В MICROSOFT VISUAL FOXPRO

Часть 1

СОЗДАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

Учебно-методическое пособие

Омск
Издательство СибАДИ
2006

УДК 681.3.06
ББК 32.973.2
С 30

Рецензенты:
канд. техн. наук, доцент В.Г. Осипов
канд. техн. наук П.С. Ложников

Работа одобрена редакционно-издательским советом Сибирской государственной автомобильно-дорожной академии для специальностей 220200 «Автоматизированные системы обработки информации и управления», 351400 «Прикладная информатика в экономике», 075500 «Комплексное обеспечение информационной безопасности автоматизированных систем».

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.

Основной целью создания данного учебно-методического пособия стала необходимость закрепления навыков работы в одной из современных СУБД с целью создания приложений для различных предметных областей у студентов высших учебных заведений, изучающих дисциплину “Системы управления базами данных”.

Учебно-методическое пособие по курсу «Системы управления базами данных» предназначено для студентов, обучающихся на специальностях «Прикладная информатика в экономике», «Автоматизированные системы обработки информации и управления», «Комплексное обеспечение информационной безопасности автоматизированных систем».

Табл. 21. Ил. 22. Библиогр.: 15 назв.

ISBN 5–93204–242–7

Оглавление

Перечень обозначений и сокращений

MS	– Microsoft;
VFP	– Visual FoxPro;
АРМ	– автоматизированное рабочее место;
БД	– база данных;
вуз	– высшее учебное заведение;
СУБД	– система управления базами данных.

Введение

В основу пособия легли ключевые вопросы по проектированию баз данных, что позволит закрепить знания, полученные в рамках лекционного курса «Базы данных» или «Безопасность систем баз данных». Данное пособие будет полезно для студентов технических вузов, начинающих специалистов в области баз данных, а также может представлять интерес для студентов специальностей, связанных с проектированием и использованием информационных систем и программного обеспечения, для обучения работе с базами данных в качестве квалифицированного пользователя.

Пособие вводит в терминологию баз данных, знакомит с основами построения реляционных баз данных и реализации средствами современных СУБД. В качестве СУБД, которая является инструментом для реализации баз данных, избран Microsoft Visual FoxPro.

MS Visual FoxPro является системой управления реляционными базами данных, которые в настоящее время являются наиболее распространенными.

MS Visual FoxPro является объектно-ориентированным, визуально-программируемым языком, управляемым по событиям, и в полной мере соответствует новым требованиям, предъявляемым к современным средствам проектирования.

В первой главе представлены основные понятия баз данных и сведения о методике нормализации данных.

Вторая глава посвящена вопросам настройки параметров MS VFP и знакомству с проектом приложения.

В третьей главе изложены основы проектирования реляционной базы данных в Visual FoxPro, а также порядок создания таблиц и связей между ними.

В четвертой главе представлен пример разработки реляционной базы данных на тему «Учет заработной платы сотрудников предприятия».

В пятой главе описана работа с таблицами в режиме ввода и редактирования данных, а также способы модификации структур таблиц.

Вторая, третья и пятая главы содержат задания для самостоятельной работы.

В приложении представлен набор тем с краткими описаниями предметных областей, которые могут буть основой для выполнения практических заданий, представленных в данном пособии.

1. Реляционная модель данных

1.1. Общая информация

Первоначально реляционная модель данных была предложена не как проект какой-либо новой СУБД, а как теоретическая схема, основное положение которой таково: реляционная база данных представляет собой совокупность отношений.

Информационными единицами в реляционных моделях являются домены, атрибуты, кортежи и отношения.

Строка отношения описывает свойства определенного объекта предметной области и может рассматриваться как формальная запись знаний о свойствах этого объекта. Строки одного отношения описывают однородные, в определенном смысле, объекты предметной области. Логические отношения могут мыслиться как таблица, строками которой являются записи, описывающие конкретные объекты предметной области.

В отличие от традиционных таблиц порядок следования столбцов таблицы не важен. Каждая колонка имеет уникальное имя. Каждая строка отношения определена уникальным идентификатором. Количественные отношения определяются числом строк, степенные отношения – числом столбцов. Например, в базе данных «Поставщики товаров» (табл. 1), отношение степени – 4, количественное отношение – 3.

Таблица 1

База данных «Поставщики товаров», построенная по реляционной модели

Бар-код	**Товар**	**Поставщик**	**Адрес**
3562747	Гладильный стол	Фирбиматик	Италия, Болонья, …
3268879	Утюг	Реал Стар	Австрия, Вена …
6789123	Пароманекен	Бузетти	Германия, Берлин …

Количество строк подвержено более динамичным изменениям, чем количество столбцов – они удаляются, добавляются как администратором базы данных, так и пользователями. Поэтому определяющими для реляционной базы данных являются степенные отношения, так как любое изменение количества столбцов приводит к трансформации самой структуры базы данных и создает совершенно новые отношения.

В реляционной модели каждому объекту предметной области соответствует одно или несколько отношений. Если надо в явном виде зафиксировать связь между объектами, то она также выражается в виде отношения, в котором в качестве атрибутов присутствуют идентификаторы взаимосвязанных объектов. Таким образом, и объекты предметной области, и

связи между ними отражаются в реляционной модели посредством одинаковых информационных конструкций. Это упрощает модель.

Система является полностью реляционной, если она:

- поддерживает структурные аспекты реляционной модели;
- выполняет соответствующие ей правила включения, корректировки и исключения;
- обладает подъязыком данных таким же мощным, как алгебра отношений.

К недостаткам реляционной модели относятся длительность поиска и сложность сортировки.

1.2. Реляционная база данных

Реляционная база данных – это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Однако пользователи могут воспринимать такую базу данных как совокупность связанных таблиц.

Признаки реляционной базы данных

1. Каждая таблица состоит из однотипных строк и имеет уникальное имя.
2. Строки имеют фиксированное число полей (столбцов) и значений (множественные поля и повторяющиеся группы недопустимы). Иначе говоря, в каждой позиции таблицы на пересечении строки и столбца всегда имеется в точности одно значение или ничего.
3. Строки таблицы обязательно отличаются друг от друга хотя бы единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы.
4. Столбцам таблицы однозначно присваиваются имена, и в каждом из них размещаются однородные значения данных (даты, фамилии, целые числа или денежные суммы).
5. Полное информационное содержание базы данных представляется в виде явных значений данных, и такой метод представления является единственным.
6. При выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию. Этому способствует наличие имен таблиц и их столбцов, а также возможность выделения любой их строки или любого набора строк с указанными признаками (например, список сотрудников с должностью «бухгалтер»).

1.3. Нормализация отношений (таблиц БД)

Определения

Кортеж определяет свойства объекта как совокупность реквизитов (атрибутов), причем должен быть реквизит с уникальными свойствами, характеризующий только данный объект. Например, номер студенческого билета или зачетной книжки (паспорта). **Такой реквизит называют ключевым**.

Если отношение имеет более одного возможного ключа, **тогда выделяют один ключ, называемый первичным**. И наоборот, если отношение не имеет ни одного атрибута, который бы полностью определил объект (строку, кортеж) отношения, то **определяется составной ключ**, который схематически отображают двойной вертикальной чертой.

Функциональная зависимость

Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Например, в таблице **Оборудование** (табл. 2) поля **Наименование** и **Тип оборудования** функционально зависят от ключа **Код_оборудования**, а в таблице **Поставщики** (табл. 3) поле **Страна** функционально зависит от составного ключа (**Поставщик**, **Город**). Однако последняя зависимость не является функционально полной, так как **Страна** функционально зависит и от части ключа – поля **Город**.

Таблица 2

Таблица «Оборудование»

Код_оборудования	Наименование	Тип оборудования
01 001	Принтер	Периферийное оборудование
01 002	Сканер	Периферийное оборудование
02 001	Стол	Офисная мебель
02 002	Кресло	Офисная мебель
02 003	Полка	Офисная мебель

Таблица 3

Таблица «Поставщики»

Поставщик	Город	Страна
ОАО «Рассвет»	Омск	Россия
ОАО «Рассвет»	Киев	Украина
ООО «Петрол»	Омск	Россия

Полная функциональная зависимость

Поле B находится в полной функциональной зависимости от составного поля A, если оно функционально зависит от A и не зависит функционально от любого подмножества поля A.

Многозначная зависимость

Поле A многозначно определяет поле B той же таблицы, если для каждого значения поля A существует хорошо определенное множество соответствующих значений B.

Таблица 4

Производство товаров

Тип товара	**Наименование товара**	**Производитель**
Продовольствие	Картофель	ЗАО «Продовольствие»
Продовольствие	Картофель	ЗАО «Аргомаш»
Продовольствие	Свекла	ЗАО «Продовольствие»
Продовольствие	Свекла	ЗАО «Аргомаш»
…	…	…

Для примера рассмотрим таблицу «Производство товаров» (табл. 4). В ней есть многозначная зависимость «Тип товара – Наименование товара»: различные товары могут относиться к одному виду товара. Есть и другая многозначная зависимость «Наименование товара – Производитель»: разные производители работают в рамках производства одного типа и наименования товара. Улучшить данный пример путем разбиения исходной таблицы на справочники «Товары» и «Производители» и связующую таблицу «Товары–Производители».

Нормализация – процесс упорядочивания, структурирования представления данных.

Разберемся, зачем нужна нормализация при построении реляционных баз данных.

Для примера, обратим внимание на следующее отношение:

Производитель	**Адрес**	**Товары**
Юнион	Италия, Альбано…	Гладильный стол
Фирбиматик	Италия, Болонья…	Пароманекен
Юнион	Италия, Альбано…	Машина сухой чистки
…	…	…

Недостатки этой таблицы сразу бросаются в глаза.

Первый – это избыточность, т.к. наименование фирмы Юнион и ее адрес повторяются, в связи с этим занимают лишнее место в системе хране-

ния и приводят к значительному расходу ресурса памяти, снижая тем самым общую производительность системы.

Еще один недостаток – потенциальная противоречивость. При попытке изменить адрес фирмы, поставляющей машины сухой чистки, мы можем забыть изменить тот же адрес в записях о других товарах той же фирмы.

Также имеется опасность аномалии включения. В базу данных не может быть внесен адрес потенциального поставщика, который не поставляет пока никаких товаров. Это значит, что мы не получим от него ни одного счета на предоплату заказанных товаров.

Представляется проблемой и аномалия удаления. Если удалить все товары поставщика, мы навсегда утратим его адрес.

В реальной жизни одной таблицей обойтись невозможно и с точки зрения практичности, и с точки зрения правдивости хранимой информации.

Теперь вернемся к понятию нормализации. Следствием нормализации будет достаточно гибкий и надежный набор таблиц с минимальным дублированием информации.

Аппарат нормализации отношений был разработан Е.Ф.Коддом. В нем определялись три нормальные формы, каждая из которых ограничивает типы допустимых функциональных зависимостей отношения.

Современная теория реляционных отношений уже использует шесть нормальных форм. Рассмотрим наиболее распространенные из них.

Первая нормальная форма

Отношение находится в первой нормальной форме (1НФ), если значения атрибутов (то есть домены), из которых построено данное отношение, являются простыми, неделимыми, иначе говоря, атомарными значениями.

С точки зрения программиста это означает, что тип данных, приписанный домену, не может быть массивом, списком, множеством и т.д., а может быть только целым (integer), символьной строкой, логическим и т.д. Это обязательное требование ко всем реляционным СУБД. Одним из способов приведения отношения к первой нормальной форме является декомпозиция (разложение) его на несколько новых отношений, в совокупности эквивалентных исходному.

Вторая нормальная форма

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от составного ключа. Зависимость в этом случае означает, что существует функция, по которой, зная один атрибут, можно определить значение зависимого атрибута.

Приведение отношения ко второй нормальной форме крайне необходимо. Несоответствие базы данных требованиям этого вида нормализации говорит о плохом анализе предметной области.

Если все ключи в отношении состоят только из одного атрибута, то отношение автоматически имеет вторую нормальную форму, так как по любому ключу всегда можно однозначно определить любой неключевой атрибут. Отношение может не находиться во второй нормальной форме только в том случае, если вообще нет ключей или существует составной ключ.

Рассмотрим пример отношения, не находящегося во второй нормальной форме:

Адрес	**Товар**	**Поставщик**
Италия, Болонья, …	Гладильный стол	Фирбиматик
Австрия, Вена …	Утюг	Реал Стар
Италия, Болонья, …	Утюг	Фирбиматик
Италия, Болонья, …	Пароманекен	Фирбиматик
Германия, Берлин…	Пароманекен	Бузетти

Если проанализировать данное отношение, то увидим, что одинаковые товары выпускаются различными фирмами-производителями, следовательно, ключ в данном отношении может быть только составным – «товар + поставщик», но неключевой атрибут «адрес» зависит не от всего ключа целиком, а от его подмножества – атрибута «поставщик». Разрешение этой проблемы лежит в выборе другого ключа для данного отношения, который будет определять остальные атрибуты. Таким ключом может служить бар-код товара, и таблица будет выглядеть вот так:

Бар-код	**Адрес**	**Товар**	**Поставщик**
9862747	Италия, Болонья, …	Гладильный стол	Фирбиматик
3268879	Австрия, Вена …	Утюг	Реал Стар
9827644	Италия, Болонья, …	Утюг	Фирбиматик
9825467	Италия, Болонья, …	Пароманекен	Фирбиматик
6789123	Германия, Берлин, …	Пароманекен	Бузетти

Третья нормальная форма

Отношение находится в третьей нормальной форме, если оно имеет вторую нормальную форму и каждый неключевой атрибут нетранзитивно зависит от любого ключа в данном отношении. Иначе говоря, все неключевые атрибуты должны зависеть напрямую только от ключей, то есть исключается транзитивная зависимость атрибутов.

Второй вариант определения: отношение находится в третьей нормальной форме, если оно удовлетворяет определению второй нормальной формы и не одно из его неключевых полей не зависит функционально от любого другого неключевого поля.

Если мы определяем товары по бар-коду, то имеем следующее отношение, не находящееся в третьей нормальной форме:

Бар-код	**Товар**	**Поставщик**	**Адрес**
3562747	Гладильный стол	Фирбиматик	Италия, Болонья, …
3268879	Утюг	Реал Стар	Австрия, Вена …
6789123	Пароманекен	Бузетти	Германия, Берлин …

Рассматриваемое отношение не находится в третьей нормальной форме, так как адрес фирмы напрямую зависит от ее названия, а название (Поставщик) не является ни ключом, ни частью ключа в данном отношении. Корректировка таблицы заключается в ее разбиении на две таблицы: первая включает в себя три реквизита – бар-код, товар, наименование поставщика, а вторая – адреса фирм-поставщиков.

В большинстве случаев приведение отношений к третьей нормальной форме оказывается достаточным для дальнейшего успешного функционирования базы данных.

Остальные нормальные формы на практике используются гораздо реже, но с усложнением взаимосвязей в предметной области, интеграцией информационных потоков, внедрением сетевых технологий, все более насущным становится введение более сложных требований к нормализации отношений, и в жизнь вводятся новые нормальные формы.

Нормальная форма Бойса-Кодда

Отношение находится в нормальной форме Бойса-Кодда, если оно находится в третьей нормальной форме и существует некоторый неключевой атрибут A, который зависит от набора атрибутов B, и при этом B не включает A, то B должен обязательно включать некоторый ключ. Иначе говоря, если в таблице есть какие-то зависимые неключевые атрибуты, то они должны обязательно зависеть от ключа.

По другому варианту определения отношение находится в нормальной форме Бойса-Кодда, если и только если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от возможного ключа.

Примером отношения, находящегося в третьей нормальной форме, но при этом не отвечающем условию Бойса-Кодда, может служить таблица, определяющая код международного телефонного номера по названию страны и названию города. При этом предполагается, что этот код может распространяться и на пригородные районы данного города, а также, что один и тот же город может иметь несколько кодов в зависимости от района города. В данном отношении имеет место составной ключ – наименование

страны плюс наименование города. В свою очередь, наименование страны зависит от телефонного кода страны – по коду можно определить страну.

Страна	**Город**	**Код**
Австралия	Мельбурн	61 396
Франция	Париж	10 572
Казахстан	Алматы	007 372
Италия	Болонья	39 051

Приведение отношения к нормальной форме Бойса-Кодда зачастую не является обязательным, им вполне можно пренебречь в данной таблице – она годна для использования и в таком виде.

Четвертая нормальная форма

В следующих нормальных формах учитываются не только функциональные, но и многозначные зависимости между полями таблицы. Для их описания познакомимся с понятием полной декомпозиции таблицы.

Полной декомпозицией таблицы называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым таблицы.

Если существует запись (x, y1,z1) и запись (x,y2,z2), то в силу симметричности определения должны существовать записи (x,y2,z1) и (x,y2,z1). Многозначные зависимости выявляются при логическом группировании некоторых атрибутов и их симметричном вхождении в отношении. Естественно было бы предположить, что необходимым требованием при этом является нахождение отношения в третьей нормальной форме. Рассмотрим пример не совсем удачной таблицы, содержащей сведения о поставщиках товаров в некоторой посреднической фирме.

Адрес	**Товар**	**Поставщик**
Италия, Болонья, …	Гладильный стол	Фирбиматик
Австрия, Вена …	Утюг	Реал Стар
Италия, Болонья, …	Утюг	Фирбиматик
Италия, Болонья, …	Пароманекен	Фирбиматик
Германия, Берлин…	Пароманекен	Бузетти

Из таблицы можно предположить, что поставщики Фирбиматик и Бузетти производят одинаковые по своим параметрам пароманекены, однако это совсем не так, следовательно, в такую таблицу должен быть занесен код каждого товара.

Если отношение не находится в четвертой нормальной форме, это зачастую означает, что в предметной области были пропущены какие-то

классы объектов или сущностей. В данном случае этой сущностью является код товара.

Пятая нормальная форма

Таблица находится в пятой нормальной форме тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный ключ. Таблица, не имеющая ни одной полной декомпозиции, также находится в пятой нормальной форме.

Методика нормализации

Нормализация, как видно из рассмотренных примеров, не является самоцелью. Она призвана упорядочить схему базы данных, сделать ее удобной для дальнейшей работы и заложить в нее заранее возможность реализации незапланированных запросов. В примерах мы исходили из того, что необходимо исправить уже допущенные ошибки построения схемы базы данных. Как же их избежать при первоначальном проектировании?

Первый шаг – тщательный анализ предметной области, правильное выделение классов объектов, самих объектов (сущностей), их классификация и определение для них тех атрибутов, которые отвечают потребностям решения именно данной задачи.

Следующий, второй, шаг – поиск удачных естественных ключей. Атрибуты, входящие в ключ, должны быть проанализированы на предмет своей уникальности, т.е. они должны однозначно идентифицировать тот или иной объект или сущность. Если таковых ключей не обнаружено, должен быть введен искусственный ключ.

Третий шаг заключается в детальной спецификации атрибутов для таблиц, в том числе определении доменов, т.е. типов данных, которые будут приписаны атрибутам.

Далее проанализируем созданную схему и определим, какие действия будут совершаться с формализованной таким образом информацией.

Подытоживая все сказанное, можно заключить, что общие правила методики нормализации выглядят следующим образом:

1. Схема должна соответствовать реальной предметной области.
2. Атрибуты должны зависеть только от всего ключа целиком и ни от чего другого.
3. Чем меньше зависимостей внутри таблиц, тем лучше.
4. Естественные ключи на проверку редко оказываются удобными.

Существует альтернативный вариант выполнения нормализации. На основе полученных сущностей и связей с определенными для них атрибутами (из диаграммы «сущность–связь», которая является конечным шагом концептуального проектирования) формируют кортежи. Полученные кортежи выстраиваются в один ряд, который считается первым уровнем нор-

мализации. Если в одном или нескольких кортежах наблюдается возможность появления дублирующих блоков записей, то выполняют декомпозицию, а именно разбиение кортежей на два или более с введением ключевых полей и полей связи между проекциями каждого кортежа. В результате получаем второй уровень нормализации. Далее описанные действия по декомпозиции повторяются столько раз, пока не будет сформирована структура данных, позволяющая избавиться от дублирования информации, как источника противоречий, но при этом позволяющая с помощью операций соединения всегда вернуться к первому уровню представления данных в кортежах, что подтвердит отсутствие потерь данных в полученной схеме.

2. **Запуск и настройка СУБД** MS VFP

Основные определения

Система – это совокупность взаимодействующих элементов, объединенных единством целей и образующих определенную целостность.

Элемент системы – часть системы, имеющая определенное функциональное назначение.

Управление – осуществление определенной последовательности формирования и реализации таких функций, как целеполагание, прогнозирование, планирование, учет, контроль и т.д.

Управление есть функция системы, обеспечивающая либо сохранение совокупности ее основных свойств, либо ее развитие в направлении определенной цели.

База данных – набор данных, организованных определенным образом.

Реляционная база данных обеспечивает хранение данных в одной или нескольких таблицах, связь между которыми осуществляется посредством значений одного или нескольких совпадающих полей.

СУБД – это набор инструментов для обеспечения взаимодействия пользователя с базами данных.

VFP 9.0 – новая версия СУБД Microsoft Visual FoxPro для Windows, 32-разрядное приложение, является объектно-ориентированным, визуально-программируемым языком, управляемым по событиям.

Для быстрой разработки несложных приложений можно воспользоваться набором мастеров построения форм, отчетов и др. VFP управляет реляционными базами данных. База данных в Microsoft Visual FoxPro – это совокупность связанных таблиц. В VFP встроены функции обмена данными с другими приложениями Windows, поддерживается доступ к наиболее популярным SQL-серверам баз данных, таким как: Microsoft SQL Server, Oracle, Informix и др., используя стандарт ODBC.

Запрос (на выборку) – средство отбора данных из одной или нескольких таблиц при помощи определенного пользователем условия. Запросы позволяют создавать виртуальные таблицы, которые состоят из вычисляемых полей или полей, взятых из других таблиц.

Форма – средство отображения данных на экране и управления ими.

Отчет – средство подготовки печатной формы документа на основании данных БД.

Класс – один из способов повышения производительности разработки приложений путем определения стандартных объектов создаваемого приложения и создания шаблонов для таких объектов. При разработке приложений для больших фирм классы часто используются для стандартизации пользовательского интерфейса. Можно определить класс форм, в котором задан определенный цвет фона и стандартный набор кнопок для управления данными. Для полной стандартизации полезно иметь один или несколько пользовательских классов для каждого базового класса. Например, можно определить отдельный класс для поля ввода, которое доступно только для чтения, и класс для редактируемого поля ввода. Если в дальнейшем понадобится изменить стандарт, будет достаточно изменить пользовательские классы (шаблоны), на основе которых создаются объекты приложения.

Проект приложения – основное средство объединения отдельных элементов приложения Visual FoxPro. Он выполняет следующие функции: запоминает расположение каждого включенного в него элемента, что создает удобства при объединении базы данных, программ, экранных форм, отчетов и упрощает управление приложением; сохраняет объектный код в Memo-полях, что уменьшает количество отдельных файлов .FXP; осуществляет поиск и собирает файлы, на которые есть ссылки в проекте; отслеживает текущие версии элементов; в случае необходимости перекомпилирует программы, обновляет экранные формы, меню и т.д.

Знакомство с главным окном Microsoft Visual FoxPro

Заголовок находится в верхней части главного окна.

Строка меню содержит меню текущего окна.

Окно Command позволяет выполнять команды Microsoft Visual FoxPro сразу после ввода, устанавливать настройки на текущий сеанс работы в VFP. Вызывается комбинацией клавиш Ctrl + F2.

Строка состояния (в нижней части окна) показывает текущие параметры во время работы VFP.

Панели инструментов есть для каждого вида работы в СУБД, вызываются через пункт меню View | ToolBars.

Для быстрого выполнения стандартных операций используются комбинации горячих клавиш (табл. 5).

Таблица 5

Клавиатура в Visual FoxPro 6.0 **(специальные и функциональные клавиши)**

Сочетание клавиш	**Пункт меню**	**Комментарий**
Ctrl+N	File | New	Создать новый файл
Ctrl+O	File | Open	Открыть существующий файл
Ctrl+S	File | Save	Сохранить текущий файл
Ctrl+P	File | Print	Печать
Ctrl+Z	Edit | Undo	Отменить действие
Ctrl+R	Edit | Redo	Повторить действие
Ctrl+X	Edit | Cut	Вырезать
Ctrl+C	Edit | Copy	Копировать
Ctrl+V	Edit | Paste	Вставить
Ctrl+A	Edit | Select All	Выделить все
Ctrl+F	Edit | Find	Найти в текущем файле
Ctrl+G	Edit | Find Again	Найти следующий
Ctrl+D	Program | Do	Запуск программы по выбору
Ctrl+E	–	Запуск текущего объекта
Ctrl+F2	Window | Command Window	Сделать активным окно Command
Alt	–	Активизация меню
Ctrl+F5	Восстановить	Возвращает окно к первоначальному размеру
Ctrl+F7	Переместить	Перемещает окно клавишами курсора
Ctrl+F8	Размер	Изменяет размер окна клавишами курсора
Ctrl+F9	Свернуть	Минимизирует окно до пиктограммы
Ctrl+F10	Развернуть	Максимизирует окно
Ctrl+F4	Закрыть	Закрывает текущее окно
Ctrl+F6	Следующее	Активизирует следующее окно
Alt+F4	Закрыть приложение	Закрывает Visual FoxPro

Настройка параметров Microsoft Visual FoxPro

Выберите пункт основного меню Tools | Options.

Разберем назначение некоторых параметров на вкладке Regional (рис.1).

Date Format – настройка формата даты (например, значение American – мм/дд/гг; значение German – дд/мм/гг).

В программе это свойство можно настроить с помощью оператора SET DATE TO <тип даты>(например, SET DATE TO German).

Date Separator – устанавливает знак разделителя в дате (например, если поставить точку, то получим дату в формате дд.мм.гг, если двоеточие, то – дд:мм:гг).

Century – по умолчанию отображает только два последних знака года, при включении данной настройки (поставить галочку) в дате год отображается четырехзначным.

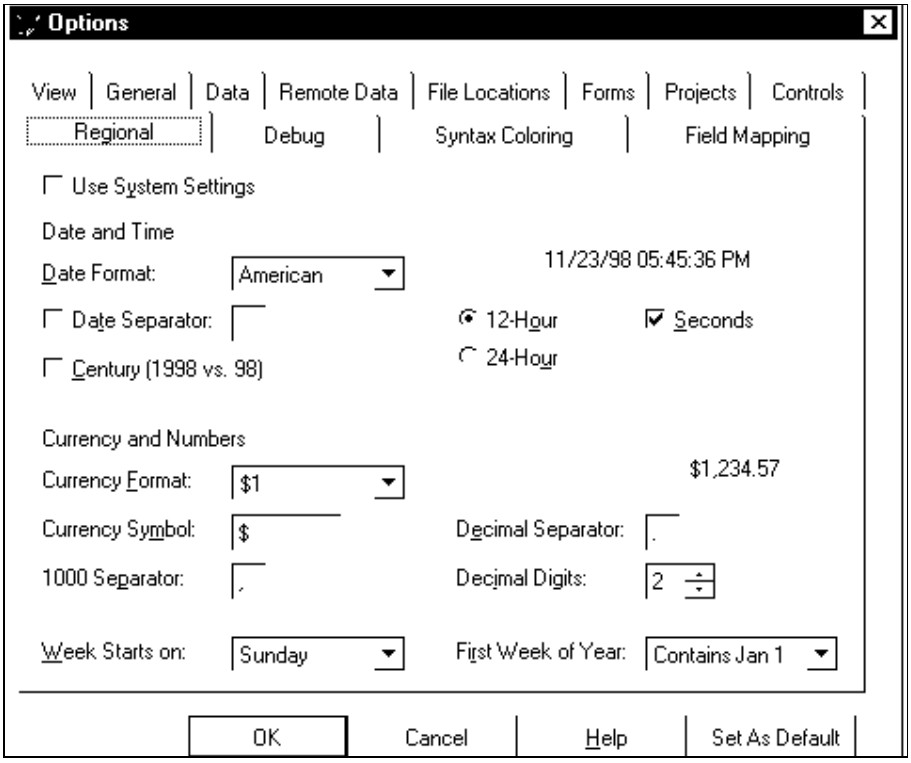


Рис. 1. Вкладка Regional – настройка даты, времени, дней недели и денежных единиц

В программе это свойство можно настроить с помощью оператора SET CENTURY ON | OFF (например, SET CENTURY ON выводит дату с четырехзначным годом).

Разберем назначение некоторых параметров на вкладке Data (рис. 2).

Open exlusive – открытие таблицы в монопольном режиме (к таблице имеет доступ только один пользователь).

В программе это свойство можно настроить с помощью оператора SET EXCLUSIVE ON | OFF (например, SET EXCLUSIVE OFF – открывать таблицы в многопользовательском режиме).

Show field names – при открытии таблицы в режиме редактирования данных (команда BROWSE) показывать названия полей.

Ignore deleted records – сделать записи, помеченные на удаление, невидимыми и исключить их из операций над данными.

В программе это свойство можно настроить с помощью оператора SET DELETED ON | OFF (например, SET DELETED OFF – сделать записи, помеченные на удаление, видимыми и разрешить им участвовать в операциях над таблицей).

Multiple record locks – при разработке сетевых приложений разрешает блокировать одновременно несколько записей в таблице.

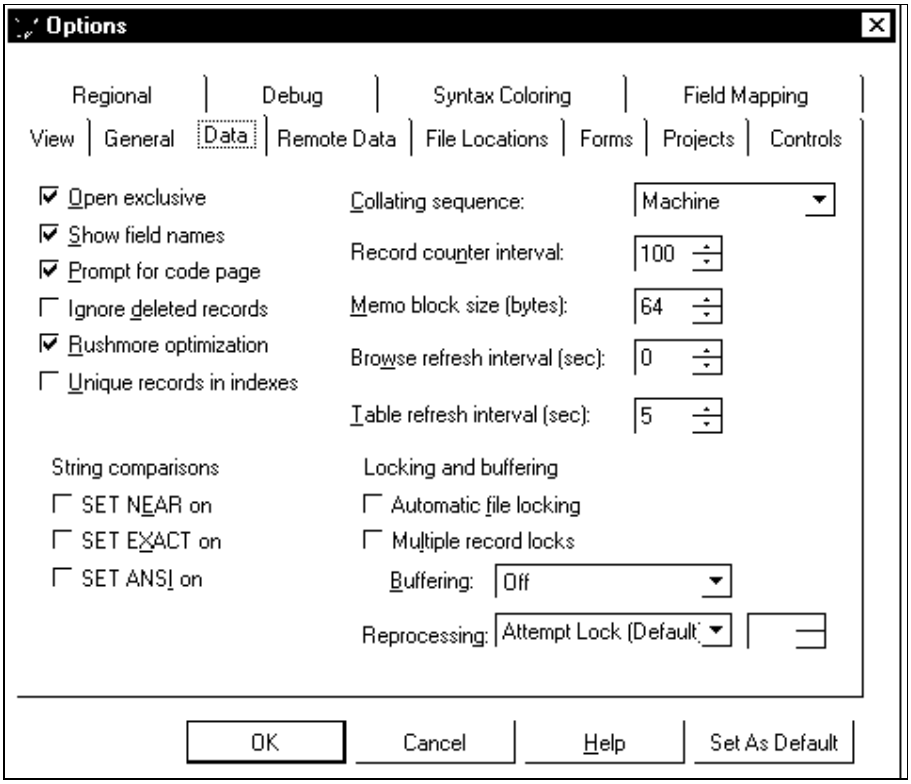


Рис. 2. Вкладка Data – настройка способа отображения данных в таблицах, правила сравнения данных

В программе это свойство можно настроить с помощью оператора SET MULTILOCKS ON | OFF (например, SET MULTILOCKS ON разрешает блокировать одновременно несколько записей в таблице).

Настройки SET EXACT ON, SET ANSI ON, SET NEAR ON будут рассмотрены в других лабораторных работах.

Разберем назначение некоторых параметров на вкладке File Locations (рис. 3).

Default Directory – указывается полный путь, где будут сохраняться файлы «по умолчанию», а также путь, по которому будет осуществляться поиск запускаемых/открываемых файлов.

В программе это свойство можно настроить с помощью оператора cd (например, cd «D:\Temp\prog1\»).

Help File – указывается полный путь и название help-файла, который будет автоматически загружаться при нажатии на F1.

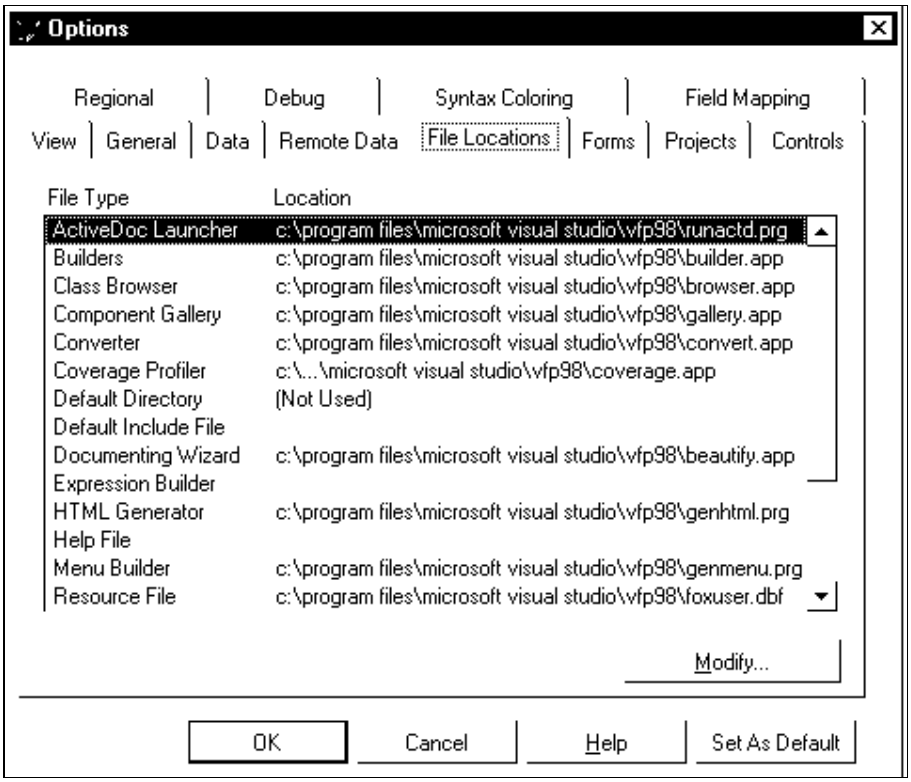


Рис. 3. Вкладка File Locations

Разберем назначение некоторых параметров на вкладке Forms (рис. 4).

Grid lines – выводит на форме в режиме редактирования сетку для облегчения выравнивания элементов формы.

Snap to grid – при включенной сетке на форме в режиме редактирования (Grid lines включен) каждый добавляемый элемент будет автоматически выравнивать свое местоположение и размер по узлам сетки.

Horizontal spacing – ширина квадрата сетки в пикселях.

Vertical spacing – высота квадрата сетки в пикселях.

Maximum desing area – максимальный размер формы (экранный размер), который можно создать в VFP, рекомендуется ставить 800x600.

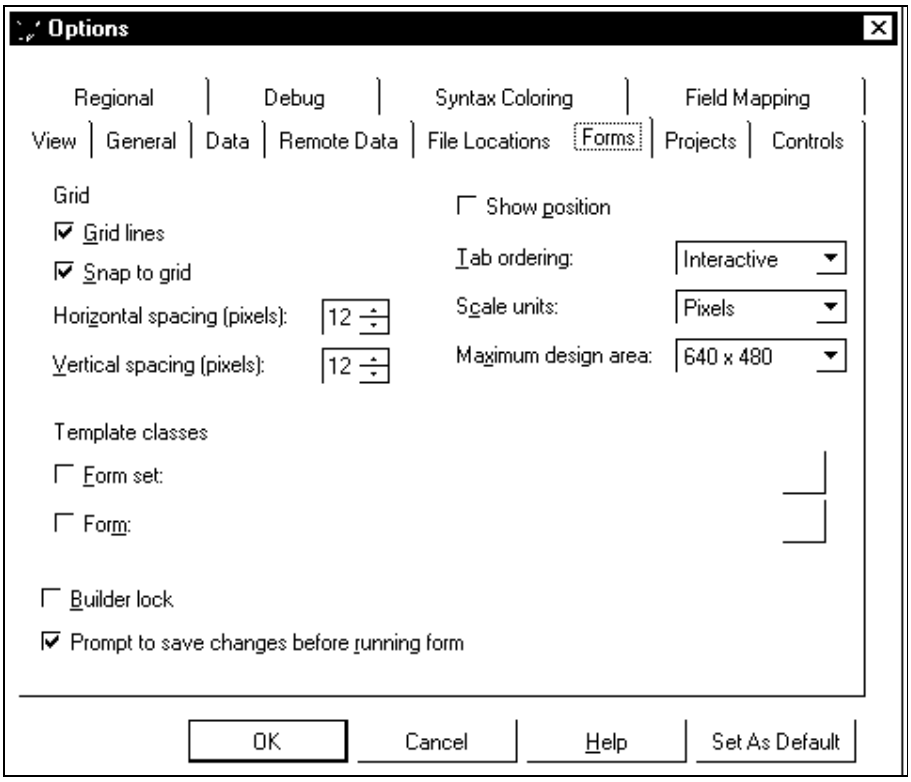


Рис. 4. Вкладка Forms

Создание нового проекта и свойства проекта

Проект VFP – набор отдельных файлов (табл. 6), объединенных под управлением единого менеджера проекта, который создается через меню File | New | New Project. Менеджер проекта позволяет создавать, удалять, переименовывать, редактировать и запускать компоненты проекта.

Таблица 6

Основные объекты Microsoft Visual FoxPro **и расширения имен файлов, в которых они хранятся**

Объект	**Расширения**
Проект (project)	PRJ, FPC, CAT, PJX, PJT
База данных (database)	DBC
Таблица Microsoft Visual FoxPro (table)	DBF
Простой индексный файл	IDX
Составной индексный файл	CDX
Memo-поле и поле типа General в отдельном файле	FPT
Форма (form)	SCX
Запрос (query)	QPR, QPX
Отчет (report)	FRX, FRM

Окончание табл. 6

Объект	**Расширения**
Почтовая наклейка (label)	LBX, LBL
Меню (menu)	MNX
Библиотека класса (class library)	VCX
Программа (program)	PRG, FXP, SPR, SPX, MPR, QPR
Рисунок	BMP
Звукозапись	WAV
Выполняемый файл приложения	APP
Файл с ошибками компиляции	ERR
Выполняемая программа	EXE
Файл с макрокомандами	FKY
Файл справки	HLP
Файл, содержащий переменные памяти	MEM

Рекомендуемая структура хранения файлов разрабатываемого приложения

Основная папка приложения хранит файл проекта, выполняемый файл приложения, выполняемую программу, файл с ошибками компиляции.

Далее в этой папке создаются подпапки для хранения отдельных элементов создаваемого приложения, что позволит избежать хаоса в массе файлов и не удалить нужные файлы.

Примерный список подпапок: IMAGE; CLASS; DATA; FORM; MENU; TMP; PROG; REPORT; QUERY; OTHER.

Особенность названий файлов и папок для VFP заключается в том, что они должны состоять из английских букв и цифр, в противном случае при запуске готового приложения будут генерироваться ошибки типа «File not found» или «Файл не найден».

Макросы клавиатуры

Макроопределения – это набор команд Microsoft Visual FoxPro, которые присвоены определенной клавише или комбинации. По умолчанию макроопределений восемь, назначенные клавишам с F2 по F9. Можно изменить макрокоманду и сохранить ее «по умолчанию» либо создать свое макроопределение и назначить своей комбинации клавиш. Можно назначить до 250 макроопределений.

{SPACEBAR} – имитация нажатия пробела.

{ENTER} – имитация нажатия клавиши ENTER.

Макрос удобно использовать для открытия вашего приложения, с которым вы постоянно работаете, можно включить выполнение различных установок (очистить экран, просмотреть переменные среды и т.д.).

Рассмотрим на первом примере два варианта настройки макроса клавиатуры.

Первый вариант настройки макроса для быстрого включения и отключения в таблицах просмотра удаленных записей (рис. 5):

1) выбрать пункт меню Tools | Macros…;
2) нажать кнопку «New»;
3) нажать клавиши в окне «Defined Key», которые станут вызывать макроопределение;
4) набрать в окне «Macro Contents»: SET DELETED OFF{ENTER}
5) нажать «OK»;
6) нажать «Set Default».

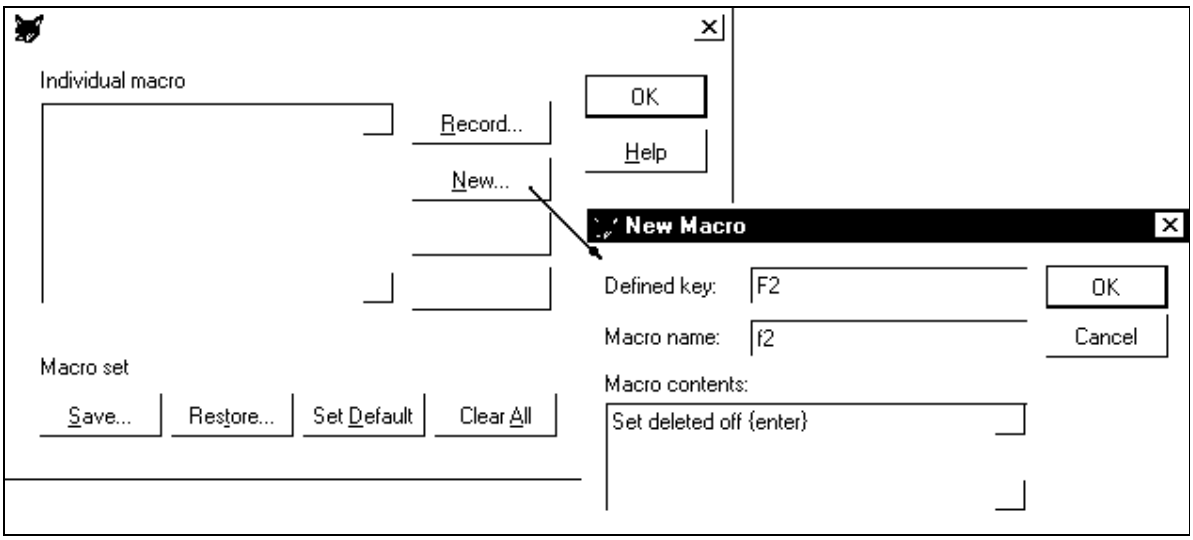


Рис. 5. Создание нового макроса клавиатуры

Другой вариант настройки макроса, если необходимо переопределить клавиши:

1) выбрать пункт меню Tools | Macros…;
2) выбрать из списка клавишу макроопределения и нажать кнопку «Edit»;
3) набрать в окне «Macro Contents»: SET DELETED ON{ENTER};
4) нажать «OK»;
5) нажать «Set Default».

Рассмотрим второй пример настройки макроса клавиатуры.

Чтобы быстро открывать ваш проект, который имеет адрес d:\vfp_prog_met\mainproject.pjx, необходимо выполнить следующие действия:

1) выбрать пункт меню Tools | Macros…;
2) нажать кнопку «New»;

3) нажать клавиши в окне «Defined Key», которые станут вызывать макроопределение;
4) набрать в окне «Macro Contents»:
   MODIFY PROJECT d:\irina\vfp_prog_met\mainproject.pjx{Enter};
5) нажать «OK»;
6) нажать кнопку «Set Default».

Для проверки работы созданных макросов установите курсор мыши на новой строке окна Command и нажмите одну из запрограммированных комбинаций клавиш.

Перечень заданий на лабораторную работу №1

1. Настроить свойства приложения MS VFP.
2. Создать папку, в которой будет храниться создаваемое в лабораторных работах приложение, определить структуру подкаталогов, создать проект.
3. Настроить макросы клавиатуры.

Ход выполнения лабораторной работы

1. Выбрать тему из списка, представленного в приложении, на основании которой будет построен весь цикл лабораторных работ. Темы у всех студентов группы должны быть разные.
2. Создать папку (по собственным фамилии и имени на английском), в которой будет храниться ваше приложение (помните, что путь должен состоять только из английских букв и цифр). Внутри созданной папки создать структуру подпапок, описанной выше.
3. Скопировать Help файл, данный преподавателем, в папку с установленным VFP.
4. Открыть VFP.
5. Выполнить настройки среды VFP в соответствии со следующими требованиями:
   - на вкладке Regional установить тип даты German, четырехзначный год, 24-часовой день;
   - на вкладке Forms установить рабочую зону формы 800x600;
   - на вкладке File Locations установить «путь по умолчанию» к вашей рабочей папке. Укажите путь к Help файлу в соответствующем свойстве;
   - сохранить изменения настроек «по умолчанию» (нажать на кнопку Set As Default).
6. Создать новый проект (название англоязычное в соответствии с темой).

7. Установить свойства проекта: ФИО автора, русское название проекта, сделайте и подключите свою иконку для проекта (которую сохраните в подпапке Image вашей основной папки, хранящей все приложение), установите кодирование исходного текста запускаемого приложения.
8. Настроить макрос клавиатуры на клавишу F2 так, чтобы сразу открывался ваш проект.

3. **Создание таблиц БД и связей между ними. Настройка целостности БД**

Основные определения

Атрибуты (на различных уровнях представления данных встречаются понятия: **поле, реквизит, столбец таблицы)** – это элементарные информационные единицы, характеристики сущности.

Домен — это множество значений. Примерами доменов могут служить множество целых чисел, множество названий городов, множество почтовых адресов, фамилий студентов и т.д.

Запись – аналог строки в таблице. Запись является стандартным блоком для хранения данных в таблице, выборке данных в запросе, форме, выводимой на экран и т.д.

Ключевое поле – поле, значение которого служит для однозначного определения записи в таблице. Ключевое поле помогает наиболее эффективно организовать поиск, хранение и объединение данных из разных таблиц.

Кортеж – совокупность атрибутов, образующих строку (запись). Кортеж определяет свойства объекта как совокупность реквизитов (атрибутов), причем должен быть реквизит с уникальными свойствами, характеризующий только данный объект. Например, номер студенческого билета или зачетной книжки (паспорта). **Такой реквизит называют ключевым.**

Если отношение имеет более одного возможного ключа, **тогда выделяют один ключ, называемый первичным**. И наоборот, если отношение не имеет ни одного атрибута, который бы полностью определил объект (строку, кортеж) отношения, то **определяется составной ключ**, который схематически отображают двойной вертикальной чертой. Например, в отношении нет атрибута, который бы однозначно определял кортеж, так как экзамены по одному учебному курсу (физика) могут сдаваться более чем в одном семестре.

Мощностью отношения называется число кортежей отношения (количество строк или записей).

Нормализация – процесс упорядочивания, структурирования представления данных.

Отношение (таблица) – это совокупность записей, называемых строками.

Предметная область – это часть реального мира, данные о которой мы хотим отразить в базе данных. Например, в качестве предметной области можно выбрать бухгалтерию какого-либо предприятия, отдел кадров, банк, магазин и т.д. Предметная область бесконечна и содержит как существенно важные понятия и данные, так и малозначащие или вообще не значащие данные. Так, если в качестве предметной области выбрать учет товаров на складе, то понятия «накладная» и «счет-фактура» являются существенно важными понятиями, а то, что сотрудница, принимающая накладные, имеет двоих детей – это для учета товаров неважно. Однако с точки зрения отдела кадров данные о наличии детей являются существенно важными. Таким образом, важность данных зависит от выбора предметной области.

Степень отношения – число входящих в него атрибутов (количество столбцов).

Сущность – объекты предметной области. Примеры: служащий, студент, накладная, путевой лист и т.п.

Таблица – основной объект реляционной БД, хранилище информации. Состоит из полей (столбцов) и записей (строк).

Связь – это ассоциация, установленная между несколькими сущностями (или таблицами). Формируется путем создания однотипных индексных полей в паре таблиц. Существуют связи 1:1, 1: М.

Связь «один к одному» (обозначается 1:1). Это означает, что в такой связи сущности с одной ролью всегда соответствует не более одной сущности с другой ролью.

Связь «один ко многим» (1:**М).** В данном случае сущности с одной ролью может соответствовать любое число сущностей с другой ролью.

Создание базы данных в проекте приложения

Порядок создания базы данных в VFP представлен на рис. 6 - 10.

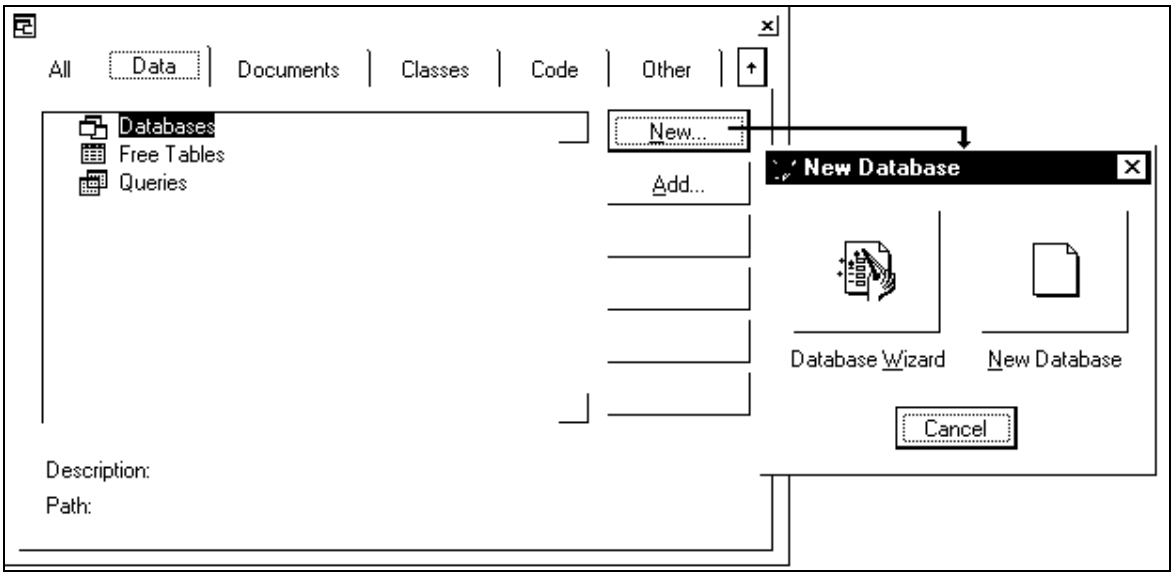


Рис. 6. Первый шаг создания новой базы данных в проекте приложения

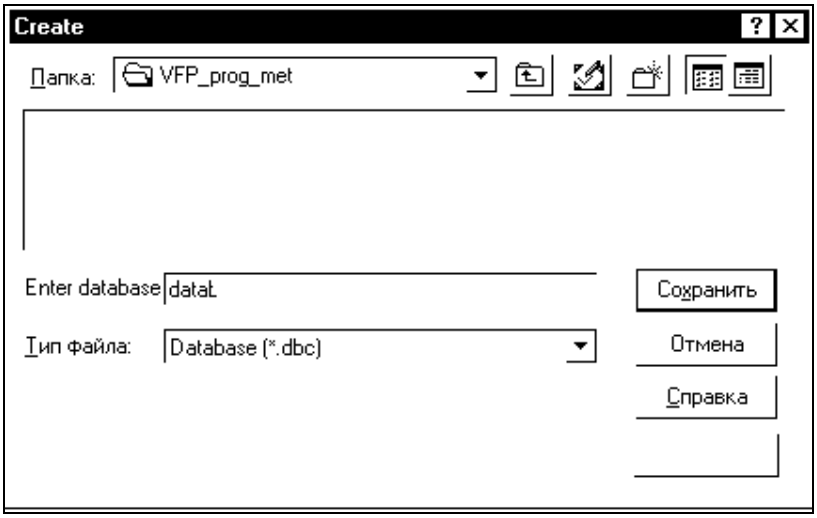


Рис. 7. Сохранение новой базы данных на диске

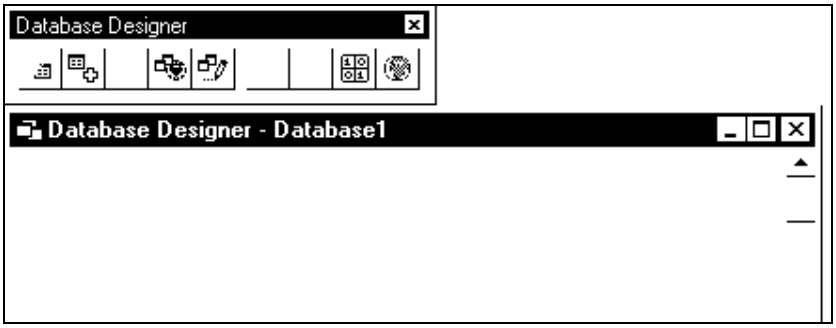


Рис. 8. Внешний вид пустой БД

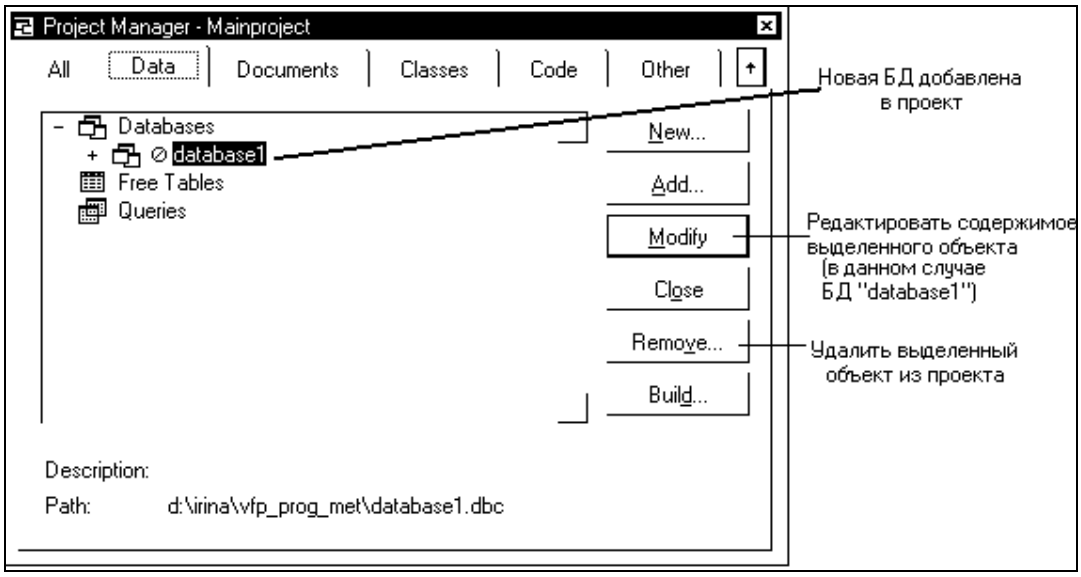


Рис. 9. Вид проекта приложения после создания новой БД

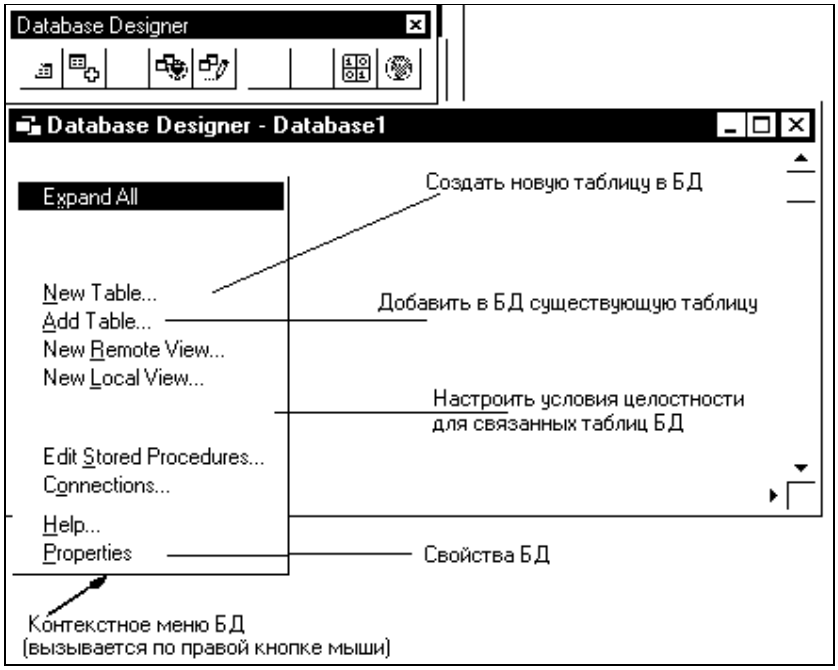


Рис. 10. Основные операции, которые можно выполнять в БД

Создание таблицы с использованием конструктора таблиц

Свободная таблица (Free Table)– таблица, не принадлежащая ни одной БД (создается в проекте приложения, на закладке Data, выбирается пункт Free Tables, далее кнопка New, задается название таблицы, сохраняется и перед Вами конструктор таблицы).

Связанная таблица – таблица, принадлежащая какой-либо одной БД. Создается в проекте приложения на закладке Data, выбирается пункт Databases, далее выбирается конкретная БД, нажимается кнопка Modify для редактирования структуры БД, по правой кнопке в БД вызывается контекстное меню, показанное на рис. 10, далее New Table, задается название таблицы, сохраняется и перед Вами конструктор таблицы (рис. 11, 12).

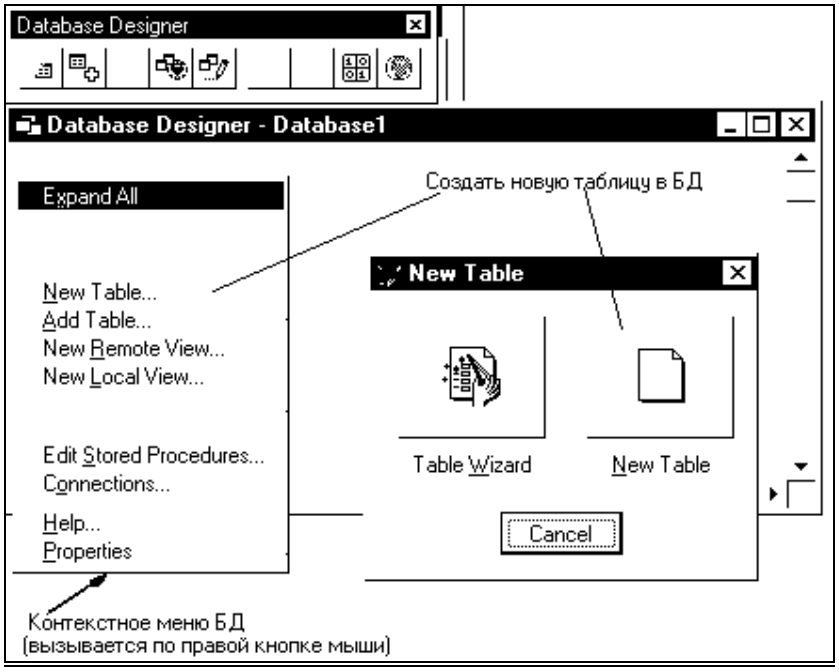


Рис. 11. Создание связанной таблицы в БД

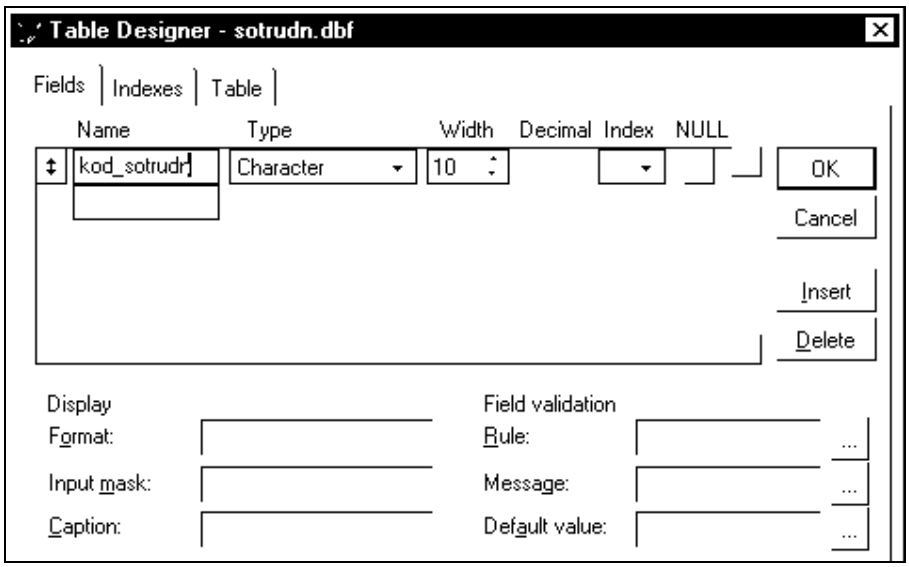


Рис. 12. Конструктор таблицы

Name – задается название (заголовок) поля таблицы. Рекомендуется задавать имя длиной не более 11 английских символов, так как при переводе таблицы в разряд свободной в ней не будут поддерживаться длинные имена полей.

Type – тип поля, определяет, какие данные могут вводиться в поле (табл. 7).

Таблица 7

Типы данных (Type) в таблицах VFP

Тип поля	Назначение	Размер
Character	Строка	1 байт на символ до 254 символов
Numeric Float Double Integer	Число	8 байтов в памяти; от 1 до 20 байтов в таблице 8 байтов 4 байта
Currency	Денежный тип	8 байтов
Date	Дата	8 байтов
DateTime	Дата и время	8 байтов
Logical	Логический	1 байт
Memo	Динамическое текстовое поле переменной длины	4 байта в таблице
General	Ссылка на OLE	4 байта в таблице

Width – общая длина данных, которые можно будет ввести (например, для полей типа Character длина до 254 символов).

Decimal – количество знаков после запятой (длина дробной части) для числовых типов данных (например, типы numeric, float, double).

Index – создание по указанному полю одноименного индекса, позволяющего вводить повторяющиеся значения в поле, который будет сортировать записи по данному полю в индексном файле либо по возрастанию, либо по убыванию.

Input mask – маска ввода, задает форму ввода и отображения данных в поле (например, XXXXXX – можно ввести в поле 6 любых символов; 999–999 – можно ввести шесть цифр, а знак «тире» будет отображаться автоматически). Если длина маски меньше длины поля (параметра Width), то она не даст ввести дополнительные символы сверх длины маски.

Caption – пояснение к полю, которое будет выводиться вместо названия поля, но при программировании нужно использовать название поля, указанное в Name.

Модификация структуры таблицы. Для изменения структуры таблицы (добавления, удаления, изменения свойств полей и индексов) в про-

екте приложения на закладке Data выполняется выбор нужной таблицы и нажимается клавиша Modify.

Индексы (создание первичного ключа, составного ключа). Для того чтобы создать индекс, выберите таблицу, откройте ее в режиме конструктора таблиц «Table Designer» и выберите вкладку «Indexes» (рис. 13).

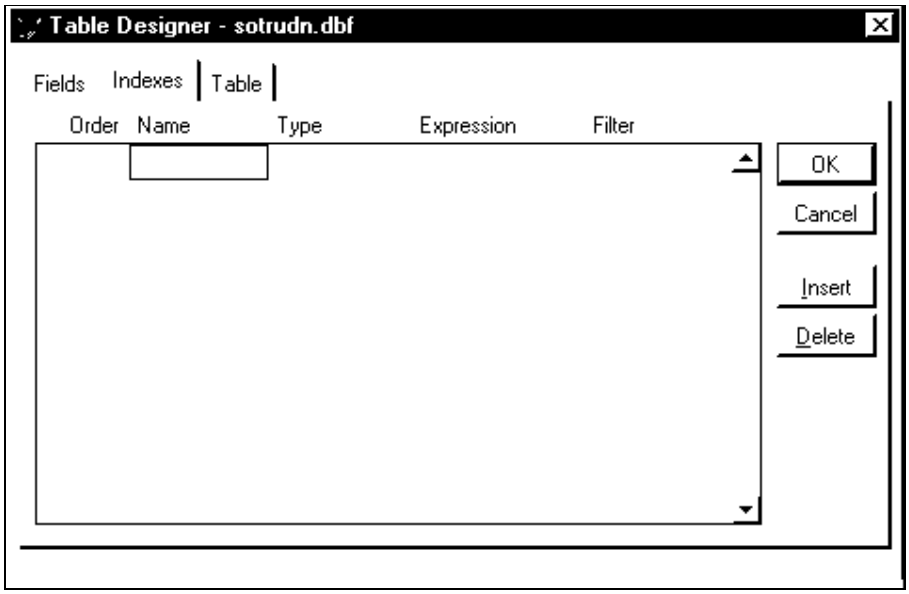


Рис. 13. Вкладка «Indexes» конструктора таблицы для создания индекса

Каждый индекс в Visual FoxPro имеет имя, на которое можно в дальнейшем ссылаться при упорядочении отображения данных в соответствии с данным индексом. Имя индекса содержится в поле Name. С левой стороны имени индекса располагается переключатель, определяющий порядок упорядочивания значений индексного выражения. Это имя может совпадать с именем поля, по которому создается таблица, но может иметь и собственное название.

Список Type используется для установки типа создаваемого индекса. Описание значений данного списка представлено в табл. 8.

Вы можете ввести имя индексного выражения (имя поля или выражения из группы полей, по которому/ым создается индекс) непосредственно в поле ввода Expression или же выбрать кнопку, расположенную правее поля ввода, для формирования выражения используется окно диалога «Expression Builder» (Конструктор выражений), представленное на рис. 14.

Таблица 8

Типы создаваемых индексов

Тип индекса	**Описание**
Regular	Хранятся значения индексного выражения для всех записей таблицы. Если несколько записей имеют одинаковое значение индексного выражения, то каждое значение хранится отдельно и содержит ссылку на связанную с ней запись
Unique	Хранятся только неповторяющиеся значения индексного выражения. Если две или более записей содержат одинаковое значение индексного выражения, то будет храниться только одно значение и ссылка на первую из записей с одинаковым значением индексного выражения. Таблица может иметь несколько уникальных индексов
Candidate	Создается уникальный индекс, который не содержит полей с пустыми значениями. Этот индекс обладает всеми качествами первичного ключа и не является им только по той причине, что таблица не может содержать более одного первичного ключа
Primary	Создается уникальный индекс, который используется для связывания таблиц и определения условий целостности данных. Поля, входящие в первичный ключ, не должны допускать ввод пустых значений. В отличие от уникального индекса, таблица может иметь только один первичный ключ

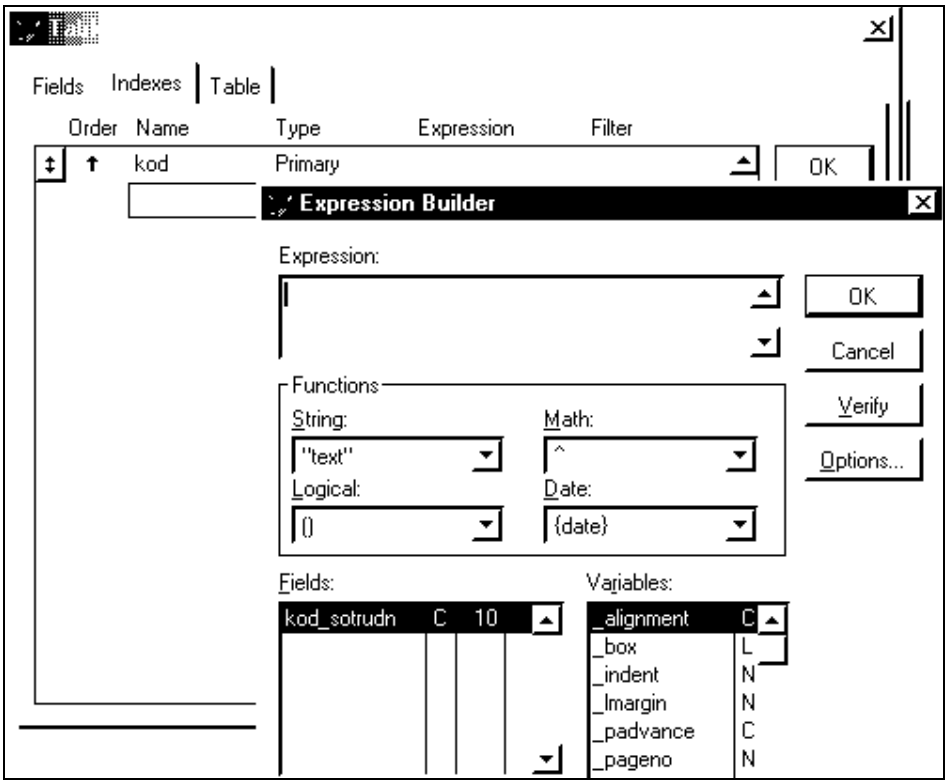


Рис. 14. Окно диалога «Expression Builder»

При использовании в выражении полей разных типов вы должны проверять допустимость введенного выражения. Например, вы можете создать индекс для таблицы Staff (из примера), в котором данные будут упорядочены по дате поступления на работу и ФИО сотрудника:

DTOC(Birthday)+Surname+Name+Lastname

В окне диалога создания индекса в поле Filter вы можете определить для индекса фильтр, используемый для ограничения формируемых индексных значений. Результат выражения, используемого для фильтра, должен иметь логический тип.

Индексы служат для организации связей между таблицами, а также для ускорения поиска. Но если используется индекс, в состав которого входят текстовые поля большой ширины, размер индексного файла может оказаться сравнимым с размером самой таблицы.

В качестве индексных полей не могут быть использованы поля типа Memo и General.

Пример создания первичного ключа

Рассмотрим последовательность действий для создания первичного ключа для таблицы Staff (смотрите следующий раздел с примером создания БД), в качестве которого используется поле t_number (Код сотрудника или табельный номер).

1. Откройте окно конструктора таблиц для таблицы Staff. Для этого в окне проекта установите курсор на модифицируемую таблицу и нажмите кнопку Modify конструктора проекта.
2. В окне диалога «Table Designer» выберите вкладку «Indexes».
3. В открывшемся окне диалога в поле Name введите имя индекса cd_number.
4. Из списка возможных типов индекса в поле Type выберите Primary.
5. Перейдите в поле Expression и введите выражение для индекса t_number.
6. Проверьте синтаксис выражения и закройте окно конструктора выражений.
7. Установите переключатель **По возрастанию**. Сохраните результат.

Определение отношений между таблицами

В Visual FoxPro вы можете устанавливать постоянные отношения между таблицами, которые будут поддерживаться при создании форм, отчетов и запросов. При определении отношений одна из таблиц является родительской, другая — дочерней. Для родительской таблицы должен быть

определен первичный ключ или ключ-кандидат, а для дочерней — индекс для связи с родительской таблицей. Например, при задании отношений между таблицами Staff **и** Paies, для таблицы Staff определен первичный ключ, таблица Paies содержит индекс, не являющийся уникальным, выражение индекса которого состоит из кода сотрудника, по которому осуществляется связь между таблицами.

Для определения отношений между таблицами откройте окно конструктора базы данных и выполните следующие действия:

1. Выберите родительскую таблицу. Установите курсор мыши на первичный ключ таблицы Staff (**на название индекса**).
2. Нажмите кнопку мыши и, не отпуская ее, переместите курсор мыши на индекс дочерней таблицы Paies, по которому устанавливается связь (к названию индекса).
3. Отпустите кнопку мыши. В окне конструктора базы данных между таблицами появилась линия, отображающая созданное отношение между таблицами.

Обратите внимание, что при установке курсора мыши на первичный ключ таблицы и нажатии кнопки мыши, вместо курсора образуется копия ключа, которую вы и переносите на индекс дочерней таблицы.

Установите курсор мыши на линию, соединяющую таблицы, и нажмите дважды курсор мыши. На экране откроется окно диалога «Edit Relationship», в котором слева приведены наименование и раскрывающийся список индексов родительской таблицы, а справа аналогичная информация о дочерней таблице. В этом же окне диалога приведен тип отношений между таблицами. Для сохранения отношения нажмите кнопку **ОК**, а для отказа — кнопку Cancel.

Для удаления установленного отношения между таблицами установите курсор на линию, соединяющую связываемые таблицы, и нажмите правую кнопку мыши. Линия при этом увеличится в толщине и на экране появится контекстное меню. Выберите из этого меню команду Remove Relationship, и описание отношения будет удалено из базы данных.

Определение условий целостности данных

Установленные связи между таблицами могут быть использованы для задания условий целостности данных. Целостность данных является одним из самых важных требований, предъявляемых к базам данных.

При первой попытке настройки условий целостности необходимо предварительно в меню выбрать пункт Database, а затем подпункт Clean Up Database, представленный на рис. 15.

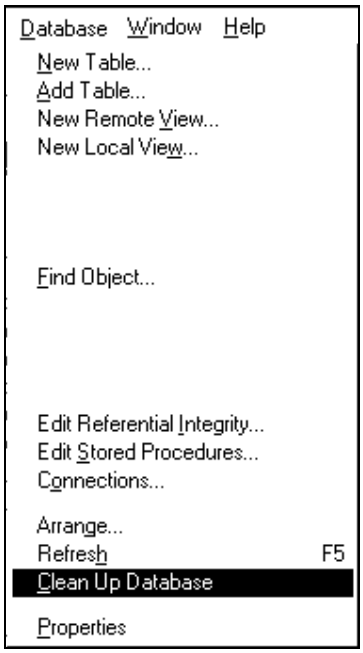


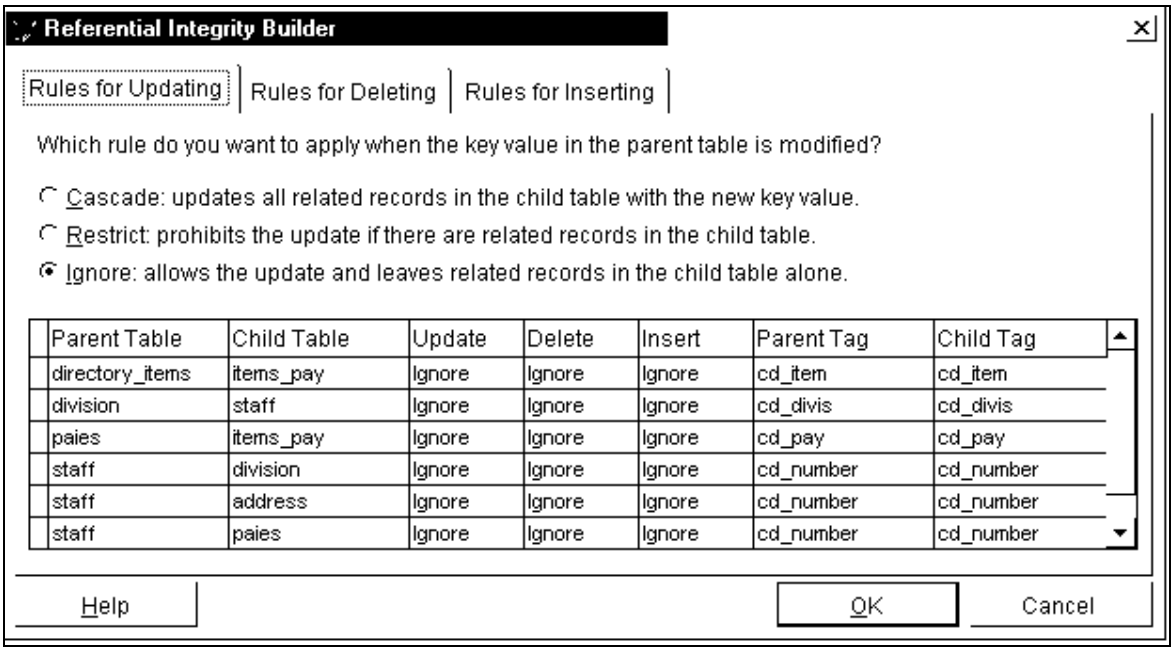
Рис. 15. Содержание
подменю Database

Для определения условия целостности данных в окне конструктора баз данных выберите команду контекстного меню Edit Referential Integrity или соответствующую ей кнопку панели инструментов «Database Designer» (рис. 16).

Рис. 16. Содержание
всплывающего меню
в Database Designer

В результате откроется окно конструктора условий целостности данных «Referential Integrity Builder» (рис. 17), которое содержит перечень всех установленных отношений между таблицами. Таблица описания условий содержит наименования родительской и дочерней таблиц, наимено-

вания индексов, используемых для связи, а также типы действий, выполняемых при модификации данных, добавлении и удалении записей.



Parent Table	Child Table	Update	Delete	Insert	Parent Tag	Child Tag
directory_items	items_pay	Ignore	Ignore	Ignore	cd_item	cd_item
division	staff	Ignore	Ignore	Ignore	cd_divis	cd_divis
paies	items_pay	Ignore	Ignore	Ignore	cd_pay	cd_pay
staff	division	Ignore	Ignore	Ignore	cd_number	cd_number
staff	address	Ignore	Ignore	Ignore	cd_number	cd_number
staff	paies	Ignore	Ignore	Ignore	cd_number	cd_number

Рис. 17. Окно «Referential Integrity Builder» для определения условия целостности данных

В верхней части окна диалога отображается краткая справка о возможном выборе действий и три переключателя, предназначенных для установки типа выполняемого действия.

При изменении значения первичного ключа или ключа-кандидата в родительской таблице возможны варианты действий, представленные в табл. 9.

При удалении записи в родительской таблице возможны варианты действий, представленные в табл. 10.

При добавлении новой записи в дочернюю таблицу или редактировании в ней существующей записи возможны следующие варианты действий, представленные в табл. 11.

После завершения определения условий целостности данных нажмите кнопку **ОК**. В результате вся введенная вами информация будет сохранена в словаре базы данных. Теперь независимо от приложений при изменении, добавлении и удалении записей будут выполняться указанные действия, обеспечивающие целостность данных.

Таблица 9

Настройка ссылочной целостности при изменении значения ключевого поля в родительской таблице

Наименование	Описание
Cascade	При изменении значений полей первичного ключа или ключа-кандидата в родительской таблице автоматически осуществляется каскадное изменение всех соответствующих значений в дочерней таблице. Например, если это правило применить к отношению между таблицами Staff и Paies, при изменении кода сотрудника в таблице Staff автоматически будут изменены коды и в таблице Paies
Restrict	Не позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на изменяемую запись. Данное правило также можно применить к отношению между таблицами Staff и Paies, если в создаваемом приложении коды сотрудника определяются на этапе ввода нового сотрудника и в дальнейшем не редактируются
Ignore	Позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице, независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 10

Настройка ссылочной целостности при удалении записи в родительской таблице

Наименование	Описание
Cascade	При удалении записи в родительской таблице автоматически осуществляется каскадное удаление всех записей из дочерней таблицы, связанных с удаляемой записью. Данное правило целесообразно применять к отношению между таблицами Staff и Paies
Restrict	Не позволяет удалять записи в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на удаляемую запись. При попытке удаления записи возникает ошибка, которую вы можете обработать программно. Если данное правило применить к отношению между таблицами Staff и Paies, вы можете быть уверены в том, что удаление записей в таблице Staff не приведет к нарушению целостности данных
Ignore	Позволяет удалять записи в родительской таблице независимо от существования связанных записей в дочерней таблице. Очевидно, что целостность данных при этом не поддерживается

Таблица 11

Настройка ссылочной целостности при добавлении записи в дочерней таблице

Наименование	Описание
Restrict	Не позволяет вводить запись, если значение индексного выражения дочерней таблицы не соответствует ни одной из записей родительской таблицы. Данное правило вы можете применять к отношению между таблицами Staff и Paies
Ignore	При вводе данных в дочернюю таблицу не анализируется содержимое индексного выражения. Целостность данных при этом не поддерживается

Перечень заданий на лабораторную работу №2

1. Проанализировать предметную область темы, данной преподавателем (темы в приложении).
2. Определить сущности, информация о которых будет отражена в БД.
3. Составить общий список атрибутов каждой сущности, которые необходимо учесть в таблицах БД (примерно 30-40 шт.).
4. Атрибуты распределить по отдельным таблицам в соответствии с правилами нормализации, установить связи между таблицами с помощью индексированных и ключевых полей.
5. Определить для таблиц и атрибутов англоязычные названия, определить типы атрибутов, размеры и свойства с учетом особенностей СУБД VFP.
6. В созданном ранее проекте приложения (лабораторная работа №1) на VFP реализовать таблицы и связи между ними.
7. Настроить условия целостности БД.

4. Пример разработки базы данных

Создадим проект приложения для автоматизации учета заработной платы на предприятии.

1. Определим тему БД: **Учет заработной платы сотрудников предприятия**.

2. Предметная область.

На предприятии необходимо вести учет сотрудников, а также хранить всю информацию по выданным заработным платам с их расшифровкой по

доходным и расходным статьям (например: оклад, районный коэффициент, подоходный налог и т.д.).

Входными документами для учета могут служить:

– листок по учету кадров;
– справочник статей для расчета заработной платы;
– законы и инструкции по расчету заработной платы.

Выходные документы – ведомость на получение заработной платы.

3. При анализе предметной области определяем ключевые объекты или сущности:

– Сотрудник;
– Зарплата сотрудника;
– Список статей для расчета зарплаты.

4. Выделим атрибуты для каждой сущности (будем рассматривать учебный вариант БД, поэтому используем приближенный вариант предметной области с небольшим набором атрибутов сущностей).

Для сущности **Сотрудник** можно выделить следующие атрибуты:

– Фамилия;
– Имя;
– Отчество;
– Дата рождения;
– Телефон домашний;
– Адрес;
– Должность;
– Дата поступления;
– Тип сотрудника;
– Подразделение.

В реальной базе данных атрибутов для описания сущности **Сотрудник** будет гораздо больше, но в примере ограничимся лишь перечисленными.

Для сущности **Зарплата сотрудника**:

– Сотрудник;
– Дата получения зарплаты;
– Размер зарплаты;
– Название статьи зарплаты (доход/ расход);
– Сумма по статье (доход/ расход).

Для сущности **Список статей для расчета зарплаты**:

– Название статьи зарплаты;
– Процентная ставка;
– Комментарий.

5. Проверим выделенные атрибуты на атомарность (неделимость), так как каждый атрибут должен быть простым и определенного типа.

Атрибуты **Адрес** и **Подразделение** сущности **Сотрудник** являются составными (например, **Адрес** состоит из реквизитов Индекс, Город, Улица, Дом, Квартира, а **Подразделение** состоит из реквизитов ФИО начальника, Название подразделения, Местоположение или кабинет, Телефон), поэтому выделим их в самостоятельные сущности, а на их месте в сущностях будут находиться связующие элементы (Связь с адресом, Связь с подразделением).

Атрибут **Сотрудник** сущности **Зарплата сотрудника** также является составным и уже выделен как отдельная сущность, следовательно, на месте данного атрибута должен быть связующий элемент с сущностью **Сотрудники** (связь с сотрудником).

6. Отобразим структуры сущностей в виде таблиц или функций и преобразуем по правилам нормализации.

Представим сущность **Сотрудники** в виде функции:

Сотрудники (Фамилия, Имя, Отчество, Дата рождения, Телефон домашний, Связь с адресом, Должность, Дата поступления, Тип сотрудника, Связь с подразделением).

Проанализируем таблицу **Сотрудники** на наличие естественного простого или составного ключа, который бы однозначно определял каждого сотрудника в таблице. Такого ключа нет, поэтому введем искусственное ключевое поле **Табельный номер**.

Сотрудники (Фамилия, Имя, Отчество, Дата рождения, Телефон домашний, Связь с адресом, Должность, Дата поступления, Тип сотрудника, Связь с подразделением, Табельный номер).

Чтобы определить, по какому адресу проживает сотрудник, необходимо определить, каким будет связующее поле между таблицами **Адрес** и **Сотрудник**.

Сущность **Адрес** (Индекс, Город, Улица, Корпус, Дом, Квартира).

В таблице **Адрес** нужно ввести поле, которое обеспечивало бы связь с таблицей **Сотрудники**. Можно ввести поле **Код адреса**, которое было бы уникальным, и аналогичное добавить в таблицу **Сотрудники**, но тогда для сотрудников с двумя и более адресами пришлось бы в таблице **Сотрудники** заводить несколько записей, которые бы дублировали всю информацию, кроме кода адреса. Поэтому введем в таблицу **Адрес** для связи поле **Табельный номер**, которое может хранить повторяющиеся значения, если

у одного сотрудника несколько адресов, а из таблицы **Сотрудники** удалим поле **Связь с адресом**.

Сущность **Адрес** (Индекс, Город, Улица, Корпус, Дом, Квартира, Табельный номер).

В таблице **Адрес** нет ключевого поля, которое бы определяло всю запись в целом, поэтому введем искусственное поле **Код записи**.

Сущность **Адрес** (Индекс, Город, Улица, Корпус, Дом, Квартира, Табельный номер, Код записи).

Сущность **Зарплата сотрудника** (Связь с сотрудником, Дата получения зарплаты, Размер зарплаты на руки, Название статьи зарплаты, Сумма по статье).

В таблице **Сотрудники** появилось поле **Табельный номер**, которое однозначно определяет каждого сотрудника, поэтому на место поля **Связь с сотрудником** ставим поле **Табельный номер**. Его отличие от поля в таблице **Сотрудники** в том, что оно будет допускать ввод повторяющихся значений, чтобы можно было в таблице **Зарплата сотрудника** хранить все записи по зарплатам, относящимся к одному сотруднику.

Взглянем на пример заполнения таблицы **Зарплата сотрудника** (табл.12).

Таблица 12

Пример заполнения таблицы «Зарплата сотрудника»

Табельный номер	**Дата получения зарплаты**	**Размер зарплаты на руки**	**Название статьи зарплаты**	**Сумма по статье**
1	12.10.2003	5000	Оклад	4000
1	12.10.2003	5000	Премия	500
1	12.10.2003	5000	Районный коэффициент	1000
1	12.10.2003	5000	Налоги	-500
2	12.10.2003	4000	Оклад	3000
2	12.10.2003	4000	Премия	400
2	12.10.2003	4000	Районный коэффициент	1000
2	12.10.2003	4000	Налоги	-400

В таблице наблюдается дублирование блоков данных, что противоречит правилам нормализации. Оно возникает из-за того, что в одной таблице хранятся и итоговые сведения по зарплате, и расшифровка зарплаты по статьям. Следовательно, нужно разбить таблицу на две: **Зарплата сотрудника** (табл. 13) и **Расшифровка по статьям** (табл. 14).

Таблица 13

Пример заполнения таблицы «Зарплата сотрудника»

Табельный номер	**Дата получения зарплаты**	**Размер зарплаты на руки**
1	12.10.2003	5000
2	12.10.2003	4000

Таблица 14

Пример заполнения таблицы «Расшифровка по статьям»

Название статьи зарплаты	**Сумма по статье**
Оклад	4000
Премия	500
Районный коэффициент	1000
Налоги	-500
Оклад	3000
Премия	400
Районный коэффициент	1000
Налоги	-400

Между таблицами необходимо установить связь, чтобы можно было однозначно понять, к какой зарплате относится список с расшифровками по статьям. Для этого введем в таблицу **Зарплата сотрудника** поле **Код зарплаты**, который будет уникальным. В таблицу **Расшифровка по статьям** также введем **Код зарплаты**, но это поле будет допускать ввод повторяющихся значений.

Сущность **Зарплата сотрудника** (Табельный номер, Дата получения зарплаты, Размер зарплаты на руки, Код зарплаты).

Сущность **Расшифровка по статьям** (Название статьи зарплаты, Сумма по статье, Код зарплаты).

Кроме того, поле **Название статьи зарплаты** содержит ограниченный набор названий статей, которые нет смысла каждый раз набирать и хранить (излишнее увеличение объема таблицы), когда есть отдельная справочная сущность **Список статей для расчета зарплаты**. Поэтому в таблице **Расшифровка по статьям** заменим поле **Название статьи зарплаты** на **Код статьи**. А в таблицу **Список статей для расчета зарплаты** введем одноименное поле **Код статьи**, которое будет уникальным и однозначно определять каждую запись.

Сущность **Расшифровка по статьям** (Код статьи, Сумма по статье, Код зарплаты).

Сущность **Список статей для расчета зарплаты** (Код статьи, Название статьи зарплаты, Процентная ставка, Комментарий).

Сущность **Подразделение** (ФИО начальника, Название подразделения, Кабинет, Телефон).

В таблице **Подразделение** присутствует атрибут **ФИО начальника**, который, по сути, является сотрудником предприятия. Поэтому можно заменить это поле на поле **Табельный номер**, чтобы получить всю информацию о начальнике через связь с таблицей **Сотрудники**.

Сущность **Подразделение** (Табельный номер, Название подразделения, Кабинет, Телефон).

В таблице **Сотрудники** необходимо определять, кто к какому подразделению относится. Для этого в таблице **Подразделение** введем поле **Код подразделения**, которое будет уникальным. А в таблице **Сотрудники** заменим поле **Связь с подразделением** на **Код подразделения**, которое сможет хранить повторяющиеся значения, так как несколько сотрудников могут работать в одном подразделении.

Сущность **Подразделение** (Табельный номер начальника, Название подразделения, Кабинет, Телефон, Код подразделения).

Сущность **Сотрудники** (Фамилия, Имя, Отчество, Дата рождения, Телефон домашний, Должность, Дата поступления, Тип сотрудника, Код подразделения, Табельный номер).

В результате получили БД, состоящую из 6 таблиц:

- Сотрудники;
- Подразделение;
- Зарплата сотрудника;
- Расшифровка по статьям;
- Список статей для расчета зарплаты;
- Адрес.

Между таблицами **Сотрудники** и **Подразделение** образовалась пара связей типа 1:M, напоминающая петлю. Чтобы данные связи не вызвали аномалий при работе с таблицами, данную петлю разрываем через ввод новой таблицы **Руководители подразделений**.

В результате таблица **Сотрудники** по **Табельному номеру** будет связана с таблицей **Руководители подразделений**, из таблицы **Подразделение** уберем поле **Табельный номер** начальника и осуществим связь с таблицей **Руководители подразделений** по **Коду подразделения**.

Преобразованная сущность **Подразделение** (Название подразделения, Кабинет, Телефон, Код подразделения).

Сущность **Руководители подразделений** (Табельный номер начальника, Код подразделения).

В результате получили БД, состоящую из 7 таблиц:

- Сотрудники;
- Подразделение;
- Зарплата сотрудника;
- Расшифровка по статьям;
- Список статей для расчета зарплаты;
- Адрес;
- Руководители подразделений.

В таблицах **Руководители подразделений** и **Расшифровка по статьям** отсутствуют уникальные ключи, однозначно определяющие каждую запись в таблице, но в учебном примере мы этого делать не будем, так как это не повлияет на работу учебной БД.

В таблицах **Руководители подразделений** и **Сотрудники** не реализована возможность сохранения изменений должностей и других перемещений внутри организации за весь период работы.

7. Определим названия таблиц, полей, свойства полей, типы индексных полей и их названия в соответствии с требованиями СУБД VFP (табл. 15-21).

Таблица 15

Структура таблицы «Подразделение» («Division»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Код подразделения	Id_division	Integer	—	Cd_divis, Primary
Название подразделения	Name_divis	Character	15	—
Кабинет	Num_cab	Integer	—	—
Телефон	Phone	Numeric	10	—

Таблица 16

Структура таблицы «Сотрудники» («Staff»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Фамилия	Surname	Character	15	—
Имя	Name	Character	15	—
Отчество	Lastname	Character	15	—
Дата рождения	Birthday	Date		—
Телефон домашний	Phone	Numeric	10 | 0	—
Должность	Post	Character	15	—
Дата поступления	Date_input	Date		—
Тип сотрудника (ИТР, служащий, рабочий)	Type_post	Character	7	—
Код подразделения	Id_division	Integer	—	Cd_divis, Regular
Табельный номер	T_number	Integer	—	Cd_number, Primary

Таблица 17

Структура таблицы «Адрес» («Address»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Код записи	Code_rec	Integer	—	Cd_record, Primary
Табельный номер	T_number	Integer	—	Cd_number, Regular
Индекс	Index_	Numeric	6 | 0	—
Город	City	Character	15	—
Улица	Street	Character	15	—
Корпус	Case	Numeric	1 | 0	—
Дом	Home	Numeric	3 | 0	—
Квартира	Apartment	Numeric	3 | 0	—

Таблица 18

Структура таблицы «Руководители подразделений» («Chiefs»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Код подразделения	Id_division	Integer	—	Cd_divis, Candidate
Табельный номер начальника	Chief_num	Integer	—	Cd_chief, Candidate

Таблица 19

Структура таблицы «Зарплата сотрудника» («Paies»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Табельный номер	T_number	Integer	—	Cd_number, Regular
Дата получения зарплаты	Pay_day	Date	—	—
Размер зарплаты на руки	Sum_pay	Numeric	9 | 2	—
Код зарплаты	Code_pay	Integer	—	Cd_pay, Primary

Таблица 20

Структура таблицы «Расшифровка по статьям» («Items_pay»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Код статьи	Code_item	Integer	—	Cd_item, Regular
Сумма по статье	Item_sum	Numeric	9 | 2	—
Код зарплаты	Code_pay	Integer	—	Cd_pay, Regular

Таблица 21

Структура таблицы «Список статей для расчета зарплаты» («Directory_items»)

Название атрибута	**Название поля в** VFP	**Тип поля**	**Длина (и количество знаков после запятой для чисел)**	**Название индексного поля, тип индекса**
Код статьи	Code_item	Integer	—	Cd_item, Primary
Название статьи зарплаты	Name_Item	Character	20	—
Процентная ставка	percent	Numeric	5 | 1	—
Комментарий	comment	Memo	—	—

8. Рекомендуемая структура хранения файлов разрабатываемого приложения.

Сама основная папка приложения хранит файл проекта, выполняемый файл приложения, выполняемую программу, файл с ошибками компиляции.

Далее в этой папке создаются подпапки для хранения отдельных элементов создаваемого приложения, что позволит избежать хаоса в массе файлов и не удалить нужные файлы. Примерный список подпапок:

- IMAGE;
- CLASS;
- DATA;
- FORM;
- MENU;
- TMP;
- PROGRAM;
- REPORT;
- QUERY;
- OTHER.

Создайте в основной папке проект приложения (Project) с названием Wages.

9. Создание таблиц в проекте приложения VFP и связей между ними (рис. 18-21).

Базу данных сохраним в подпапке DATA с именем DB_Wages. В этой же папке сохраним и все таблицы БД.

Пример создания таблицы Staff представлен на рис. 18, 19.

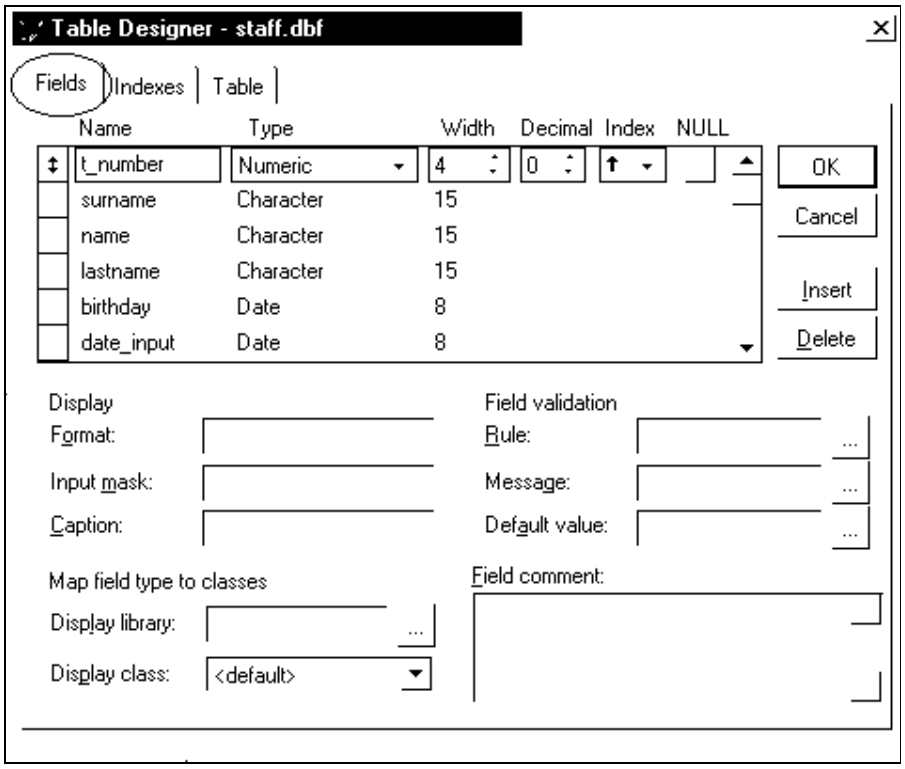


Рис. 18. Таблица Staff в режиме редактирования структуры полей

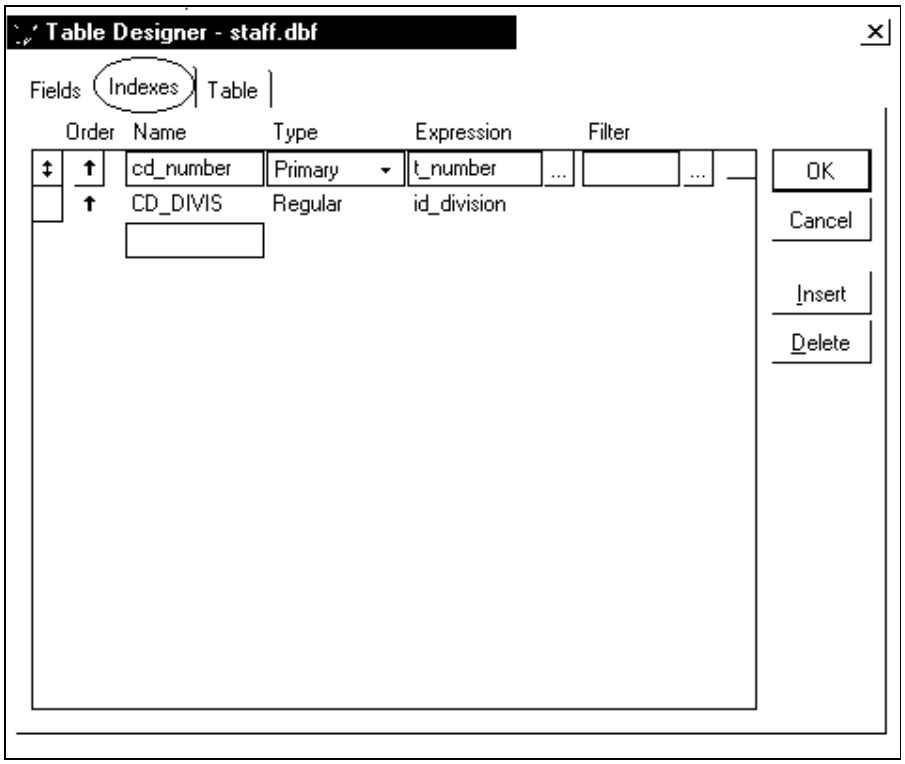


Рис. 19. Таблица Staff в режиме редактирования структуры индексов

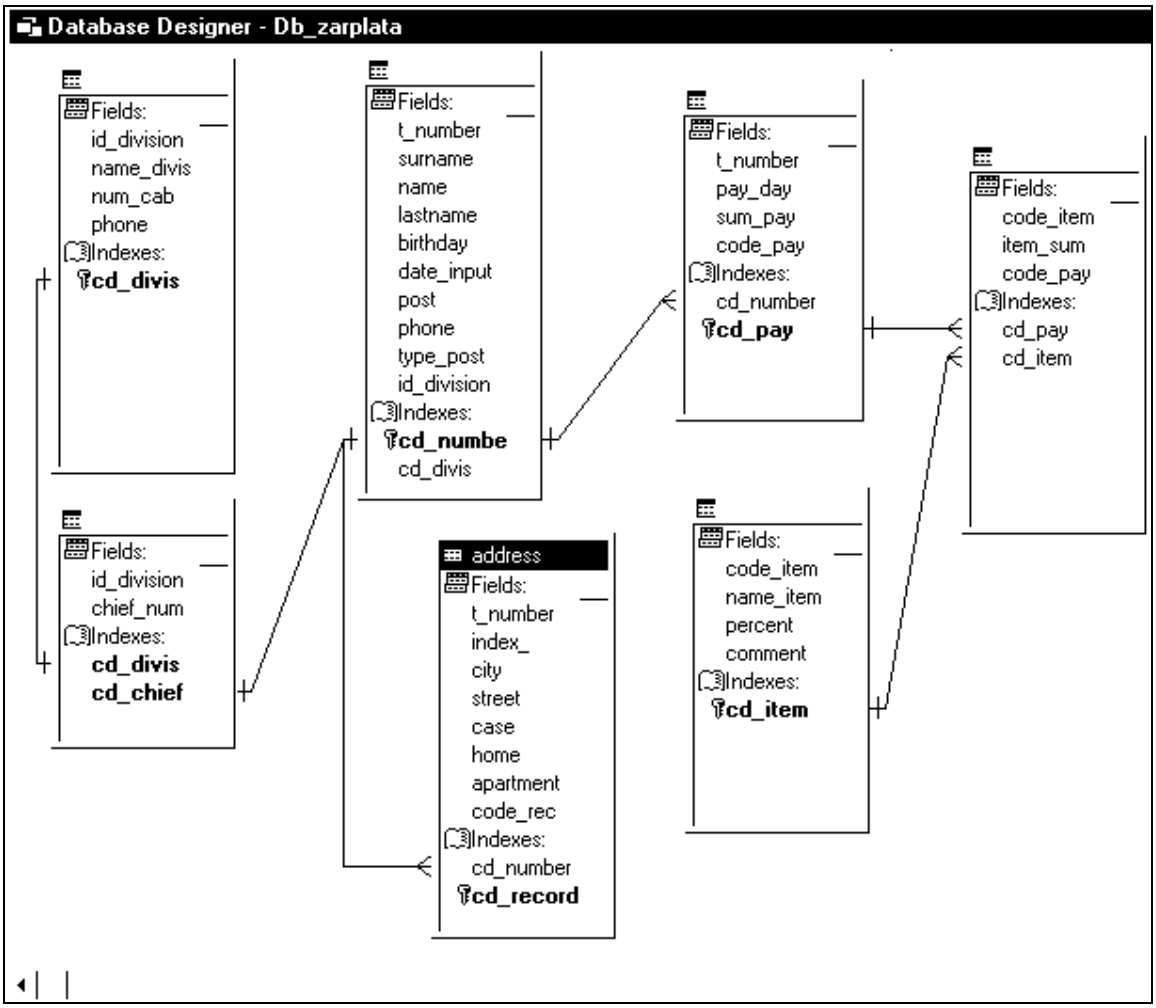


Рис. 20. Результат создания БД в VFP

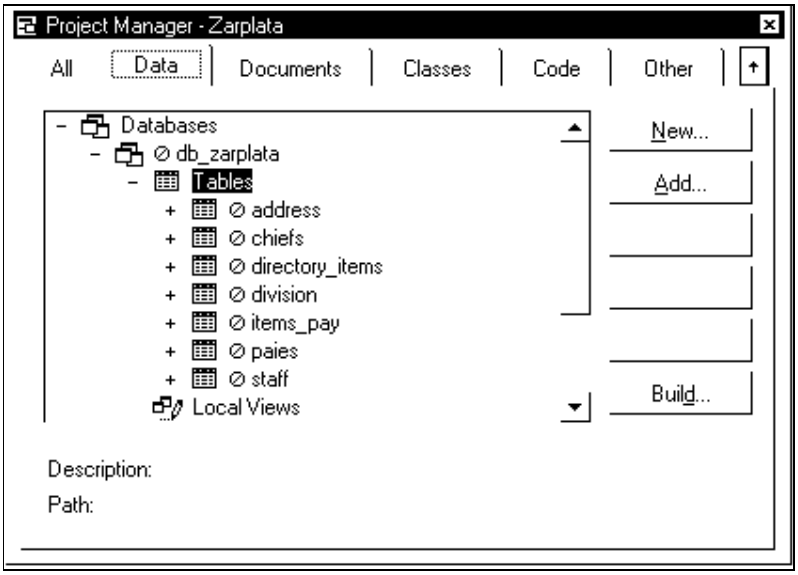


Рис. 21. Состав проекта приложения
после создания БД и связанных таблиц

5. Работа с таблицами в режиме ввода и редактирования данных. Модификация структур таблиц

Модификация структуры таблицы

Если описали всю структуру таблицы, не сделав при этом ни одной ошибки, необходимо сохранить структуру созданной таблицы.

Однако вполне вероятно, что при задании структуры могли допустить ошибки. Visual FoxPro предоставляет вам средства для исправления ошибок. К их числу относятся:

1. Изменение наименования поля и/или его типа.
2. Вставка пропущенного поля.
3. Удаление ошибочно введенного поля.
4. Изменение порядка следования полей в таблице.
5. Переопределение полей.

Для модификации структуры таблицы, входящей в базу данных, установите в окне проекта курсор на модифицируемую таблицу и нажмите правую кнопку мыши. Из появившегося на экране контекстного меню выберите команду Modify. В качестве альтернативного способа вы можете открыть окно конструктора базы данных и воспользоваться кнопкой Modify Table панели инструментов «Database Designer». В результате на экране появится окно диалога «Table Designer», содержащее структуру выбранной таблицы.

Для открытия окна конструктора базы данных и модификации таблиц созданной ранее базы данных вы можете воспользоваться командой MODIFY DATABASE, которая имеет следующий синтаксис:

MODIFY DATABASE [имяБазыДанных |?] [NOWAIT] [NOEDIT]

Эта команда может быть использована в программе или в командном окне.

Изменение наименования поля и/или его типа

Предположим, что при задании имени поля или его типа была допущена ошибка. При этом ошибка была обнаружена только после окончания ввода поля. Установите курсор на наименование поля или тип, которые требуется изменить. Для удаления в имени поля неправильно введенных символов используйте клавишу *Backspace* или *Del*. После этого введите правильное имя поля.

Изменение типа поля осуществляется аналогично его присвоению.

Вставка и удаление поля

Если необходимо ввести поле в середине таблицы, воспользуйтесь кнопкой Insert. Для вставки поля с помощью клавиш-стрелок установите

курсор на строку, перед которой вы хотите вставить пропущенное поле, и нажмите кнопку Insert. Курсор при этом может находиться в любом столбце. Visual FoxPro вставит пустую строку и присвоит полю имя NewFld. После этого вы можете ввести требуемое имя поля и его тип, как было описано ранее.

Если после создания структуры таблицы окажется, что ошибочно введено лишнее поле, вы можете удалить его. Для этого с помощью клавиш-стрелок установите курсор на поле, которое требуется удалить, и нажмите кнопку Delete.

Изменение порядка следования полей

Если вам необходимо изменить порядок следования полей в таблице, воспользуйтесь кнопкой, расположенной слева от наименования поля. Для этого установите курсор на поле, местоположение которого требуется изменить. На кнопке появится пиктограмма. Установите курсор мыши на пиктограмму, нажмите кнопку мыши и, удерживая ее в нажатом состоянии, перенесите пиктограмму на нужное место. В заключение отпустите кнопку мыши. Поле окажется перемещенным на новое место.

Переопределение полей

Изменение типов, длин или числа десятичных знаков полей может оказаться как простым, так и сложным делом в зависимости от самого изменения. Например, можно легко открыть конструктор таблиц, выделить нужное поле и увеличить его длину. Файл DBF при этом будет переписан, размер его увеличится, к символьным полям добавятся пробелы, а в числовых полях можно будет использовать больше знаков. Можно также увеличить длину индексных полей – Visual FoxPro автоматически перепишет индексные файлы при выходе из конструктора таблиц.

А вот уменьшение размеров полей может вызвать потерю данных, FoxPro, конечно, запросит подтверждение на изменение структуры при выходе из конструктора таблиц, но при вашем положительном ответе будет производить требуемые изменения.

Изменять типы полей также нужно с осторожностью. Так, преобразование числового поля в символьное возможно всегда, если для получившейся строки отводится достаточное число символов. FoxPro просто переписывает таблицу, используя для нужного поля функцию STR(). Аналогично, преобразование символьных полей в числовые производится с помощью функции VAL() и возможно только в том случае, если строка начинается с цифр или пробелов, иначе получившееся числовое поле будет содержать нулевое значение.

Заполнение таблицы данными

Добавление новой строки в таблицу в режиме BROWSE – Ctrl+Y.

Пометка записи на удаление (рис. 22) выполняется нажатием клавиши мыши на ячейку слева перед началом самой записи. Удаленные записи помечены черным прямоугольником.

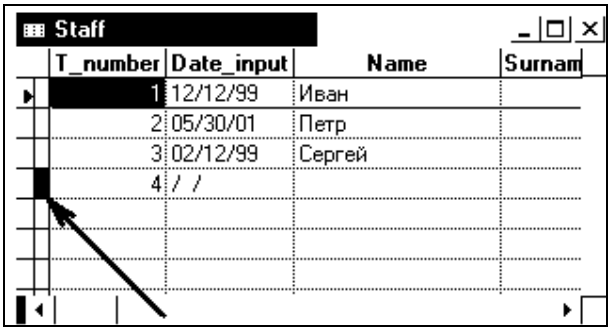


Рис. 22. Пример заполнения таблицы данными

Перечень заданий на лабораторную работу №3

1. Познакомиться с правилами заполнения и редактирования данных в таблице в режиме BROWSE, в HELP-файле найти всю информацию по режиму BROWSE.
2. Набрать по десять записей в каждой таблице БД, учитывая значения связующих полей (индексов) между таблицами.
3. Удалить некоторые записи, включить/ отключить режим просмотра удаленных записей (используя настройки VFP в меню Tools | Options, смотри теорию к лабораторной работе в пункте 2 или через окно Command с применением настройки SET DELETED OFF | ON).
4. Модификация структуры таблицы. Назначить в одной из таблиц (самой главной в Вашей БД) значения свойству Caption каждого поля.
5. Отсортировать данные в таблице по полю, которое является индексным.
6. Создать форму для главной таблицы с помощью мастера таблиц (Wizard). Познакомиться с работой формы.
7. Создать форму для пары связанных таблиц с помощью мастера таблиц (Wizard). Обратить внимание на правильность указания главной (родительской или Parent) таблицы и зависимой (дочерней или Child). Познакомиться с работой формы.

Заключение

В рамках данного учебно-методического пособия рассмотрены вопросы построения нормализованных таблиц реляционной базы данных, а также представлен набор практических занятий, целью которых является выработка навыков проектирования и физической реализациии БД в современной СУБД Microsoft Visual FoxPro.

В следующем учебно-методическом пособии будут рассмотрены вопросы программирования в СУБД Microsoft Visual FoxPro и разработки интерфейса для работы с БД.

ПРИЛОЖЕНИЕ

Темы для комплекса лабораторных работ

Тема 1

ПРИЕМНАЯ КОМИССИЯ

На каждого абитуриента заводится карточка. Она включает: ФИО, дату рождения, учебное заведение (название, номер, город), дату окончания, номер диплома, наличие красного диплома или медали, домашний адрес. Из поступивших формируются группы для сдачи приемных экзаменов. В вузе определен список специальностей и экзамены по каждой из них.

Основные функции:

- ввод/редактирование анкеты;
- ввод/редактирование специальностей и сдаваемых предметов по специальности;
- ввод полученных оценок по предметам.

Отчеты / запросы:

- выдача ведомости оценок по указанной группе /экзамену;
- отчет «Средний балл» по специальности и каждому абитуриенту;
- отчет «Не сдавшие экзамен» пофамильно и количественно по каждой специальности;
- анализ абитуриентов по месту жительства (село, город, другой город, другая республика);
- анализ абитуриентов по полученным оценкам;
- сводный анализ по абитуриентам (перекрестный запрос и диаграмма «Специальности – количество абитуриентов, получивших оценки 5, 4, 3, 2»).

Тема 2

ТОВАРООБОРОТ В МАГАЗИНЕ

В магазин поступают товары с базы. По мере надобности формируется заказ на товары. При поступлении товар наценивается (наценка формируется по каждой накладной).

Основные функции:

- ввод/редактирование актов списания товара;
- ввод/редактирование актов переоценки товара;
- формирование заявки на поставку товара (тех, количество которых меньше нормозапаса);
- оформление получения товара, добавление торговой наценки;
- оформление ежедневной продажи продукции.

Отчеты / запросы:

- справки о наличии товара, его качестве, изготовители;
- подбор товара клиентам по указанной сумме;
- ежедневный отчет об ассортименте проданного товара;
- сводный анализ товаров, пользующихся спросом;
- перекрестный запрос и диаграмма «Группа товаров – квартал»;
- отчет по прибыли (с указанием товаров, сумм, сумм НДС, сумм наценки, сумм возврата, сумм возврата тары).

Тема 3

КНИЖНЫЙ КИОСК

Киоск может работать с несколькими издательствами и фирмами-поставщиками. Документооборот включает приходные и расходные накладные, оплата с поставщиками производится безналичным путем, через оформление платежного поручения. Продажа осуществляется за наличный расчет. Киоск может оформить подписку, продать и купить книги. Оптовую продажу оформляют торговые агенты. На каждую книгу заводится карточка, включающая сведения об авторе, названии, поставщике, закупочной цене, цене подписной (если в подписке) или скидке, цене продажной оптом и в розницу, сроках поставки, адресе издательства/ фирмы.

Основные функции:

- оформление подписки на книгу, серию и оплаты подписки;
- оформление документов/ расчетов с поставщиками;
- оформление оптовой продажи (по агентам).

Отчеты / запросы:

- списки наиболее пользующихся спросом серии и отдельные издания (по районам города);
- выборка книг, интересующих клиента: по автору, издательству, серии, цене, тематике;
- сводный анализ эффективности работы торговых агентов (перекрестный запрос и диаграмма «Агент – сумма продаж по кварталу (месяцу)»);
- отчет о книготорговле за период (приход – расход);
- выборка подписок клиента и состояние подписки (сколько выкупил);
- выборка поставщиков, нарушивших сроки поставки;
- выборка «Кому должен киоск».

Тема 4

БИБЛИОТЕКА

Ведение библиотеки. Библиотека формируется различными путями: покупка книг в магазине, оформление заказа по почте с предоплатой и наложенным платежом.

Основные функции:

- ведение картотеки книг;
- оформление выдачи книг;
- отслеживание почтовых операций;
- акты списания книг.

Отчеты / запросы:

- выборки по различным критериям: по автору, издательству, серии, цене, тематике;
- отчет по книгам, находящимся у читателей;
- списки наиболее пользующихся спросом серии и отдельные издания (по районам города);
- сводный анализ эффективности работы библиотекарей (перекрестный запрос и диаграмма «ФИО – разновидность литературы (газеты/журналы/справочники/художественная/ научно-популярная)»);
- списки книг, ожидаемых по почте по видам оплат;
- списки «должников» по различным периодам в зависимости от спроса книги (1день–7дней–15дней–больше месяца –больше полгода –неисправимые).

Тема 5

СПРАВОЧНАЯ СИСТЕМА «СПОРТШКОЛА»

Основные функции:

- ввод/редактирование сведений о спортшколах (спортивные направления, контингент тренеров);
- ввод/редактирование сведений об учениках-призерах соревнований;
- ввод/редактирование о соревнованиях и школах-участницах;
- ввод/редактирование результатов (выполнение нормативов и пр.).

Отчеты / запросы:

- сводный анализ участия школ в соревнованиях (перекрестный запрос и диаграмма);
- эффективность работы тренеров (перекрестный запрос и диаграмма «ФИО – число победителей-учеников»).

Тема 6

МЕНЮ СТОЛОВОЙ

Имеются утвержденные рецептуры блюд (состав продуктов, порция, количество), справочник продуктов, имеющихся на момент составления меню. Цены на продукты изменяются (по поставкам).

Основные функции:

- ввод/редактирование ассортимента продуктов и их количества;
- составление меню;
- расчет цен блюд.

Отчеты / запросы:

- ведомость изменения цен блюд за месяц;
- сравнительная ведомость по месяцам (средняя цена обеда) (перекрестный запрос и диаграмма).

Тема 7

ТУРИСТИЧЕСКАЯ ФИРМА

Фирма может работать с несколькими агенствами по туризму.

Фирма предлагает различные виды путевок:

- заграничные;
- по городам России;
- по Омской области.

Данные виды различаются классом /условиями:

- групповые;
- индивидуальные;
- вид проезда.

Документооборот включает документы по оформлению путевок, регистрацию деятельности распространителей. Оплата путевок производится наличным путем.

Основные функции:

- регистрация данных клиентов;
- заключение договоров с зарубежными представителями турфирм и отелей;
- оформление документов/расчетов с клиентами.

Отчеты/ запросы:

- списки наиболее пользующихся спросом путевок (по районам города и видам);
- выборка путевки, интересующих клиента: по виду, срокам, категории, руководителям, цене, стране;
- выборка поставщиков по тем же критериям;

– сводный анализ эффективности работы агентов (перекрестный запрос и диаграмма «Вид /категория/ – количество путевок по кварталу (месяцу)»);
– отчет о торговле за период (приход – расход).

Тема 8
ВИДЕОТЕКА
Осуществляет следующие операции:
– приемка кассет по накладным от различных поставщиков;
– прокат кассет;
– продажа кассет.
Основные функции:
– ведение картотеки клиентов, видеокассет;
– оформление выдачи кассет;
– оформление документов/расчетов с поставщиками;
– оформление продажи (сводная за день).
Отчеты / запросы:
– выборки по различным критериям: по тематике, актерам, киностудиям, цене, продолжительности;
– отчет по кассетам, находящимся у клиентов;
– списки наиболее пользующихся спросом кассет: по районам города, тематике;
– сводный анализ эффективности работы служащих (перекрестный запрос и диаграмма «ФИО – сумма продаж / сумма проката»);
– ежедневная оборотная ведомость (покупка/ продажа);
– списки «должников» по различным периодам.

Тема 9
УЧЕБА
На каждого студента заводится карточка. Она включает ФИО, дату рождения, домашний адрес, специальность, факультет, группа. В карточку заносят результаты экзаменов и зачетов по каждой сессии.
Основные функции:
– ввод/редактирование анкеты;
– ввод/редактирование сдаваемых предметов в сессию (УЧЕБНЫЙ ПЛАН);
– ввод экзаменационных и зачетных ведомостей.
Отчеты / запросы:
– выдача ведомости оценок по указанной группе /экзамену/сессии;
– отчет «Средний балл» по специальности и каждому студенту, группе;

– отчет «Не сдавшие экзамен» пофамильно и количественно по каждой специальности;
– выборка студентов по различным критериям (по анкете);
– сводный анализ по абитуриентам (перекрестный запрос и диаграмма «Специальности – количество абитуриентов, получивших оценки 5, 4, 3, не сдавших или продливших сессию).

Тема 10

РАЙОНЫ ОМСКОЙ ОБЛАСТИ

Требуется систематизировать информацию о районах Омской области для получения данных по сельскохозяйственному производству.

Основные функции:

– ввод/редактирование карточки района (численность, размер территории, посевные площади);
– ввод/редактирование информации о выпускаемых продуктах, показатели их производства, урожайность, размеры госзакупок.

Отчеты / запросы:

– списки производимых продуктов в сопоставлении между районами по себестоимости (диаграммы);
– списки произведенных продуктов за указанный период (перекрестные запросы по показателям ПРОДУКТ – ГОД);
– какие районы перевыполнили план производства;
– различные выборки по районам (по специализирующим отраслям с/х, по орошаемым площадям).

Тема 11

ОТДЕЛ КАДРОВ

Учет работающих на предприятии. На работающих заводится карточка. Оформляются различные приказы:

– о замещении;
– о переводе на другую должность, отдел;
– об отпуске;
– о премиях и выговорах;
– об увольнении.

Основные функции:

– формирование картотеки работающих;
– оформление приказов различных видов.

Отчеты / запросы:

– выдача статистики по работающим в различных разрезах: по подразделениям предприятия, семейному положению, возрасту;
– выдача статистики по уволенным в различных разрезах;

– выдача списков людей, имеющих юбилеи в указанном году;
– список людей, имеющих день рождения в указанный день;
– списки людей предпенсионного возраста;
– сводный анализ текучки кадров: перекрестный запрос и диаграмма по кварталам, кодам, подразделениям.

Тема 12

МУЗЫКАЛЬНЫЙ КИОСК

Киоск может работать с несколькими фирмами-поставщиками. Документооборот включает приходные накладные. Оплата с поставщиками производится безналичным путем, через оформление платежного поручения. Продажа осуществляется за наличный расчет. Киоск может оформить оптовую и розничную продажу. Киоск располагает для продажи аудио-, видеокасетами, журналами, компакт-дисками.

Основные функции:

– оформление документов/расчетов с поставщиками;
– оформление оптовой продажи;
– оформление ежедневной продажи продукции.

Отчеты / запросы:

– списки наиболее пользующихся спросом (по районам города и видам товаров);
– выборка товара, интересующая клиента: по исполнителю, альбомам, категории классика/современность, композитору, цене, стране;
– выборка поставщиков по тем же критериям;
– сводный анализ эффективности работы киоска (перекрестный запрос и диаграмма «Вид/ категория/ – сумма продаж по кварталу (месяцу)»);
– отчет о торговле за период (приход – расход).

Тема 13

ХИМЧИСТКА

В химчистке ведется учет предоставляемых услуг со стоимостью выполнения. При поступлении вещи в химчистку в базу заносится информация: перечень загрязнений; перечень работ для данной вещи; стоимость каждой работы; ФИО клиента; номер и дата заказа; дата возврата вещи.

Основные функции:

– оформление документов на прием вещей в химчистку;
– ведение справочника услуг и цен на выполняемые работы;
– регистрация выдачи вещей клиентам после химчистки.

Отчеты/ запросы:

- отчёт - оформление заказа;
- список заказов, выполненных за последний месяц;
- вывод информации по конкретному заказу.

Тема 14

СЕКРЕТАРИАТ

Основными задачами секретаря являются учет поступивших документов в журнале входящих документов и регистрация исходящих документов в журнале исходящих документов, а также составление расписания посещений руководителя и учет телефонных звонков.

Основные функции:

- ведение журнала входящих и исходящих документов;
- ведение учета сотрудников предприятия;
- ведение учета исполнения документов;
- составление расписания посещений руководителя;
- регистрация звонков руководителю.

Отчеты /запросы:

- отчет в виде журнала входящих документов за определенный период (месяц, квартал);
- отчет в виде журнала исходящих документов за определенный период (месяц, квартал);
- запрос по документам, работы по которым не выполнены в указанный срок.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Проектирование БД

1. *Дейт К. Д.* Введение в системы баз данных.– М.: Вильямс, 2001.– 1071 с.
2. *Коннолли Т. и др*. Базы данных: проектирование.– М.: Диалектика, 2001.–1120 с.
3. *Крёнке Д.* Теория и практика построения баз данных: Пер. с англ.– СПб.: Питер, 2003.–800 с.
4. *Кузнецов В.В*. Проектирование баз данных: Учебное пособие.– М.: Маркетинг, 2001.–Ч. 1.–125 с.
5. *Остринская Л.И., Семенова И.И*. Базы и банки данных: теория и практика проектирования: Учебное пособие.– Омск: Изд-во СибАДИ, 2004.–241 с.
6. *Райордан Р*. Основы реляционных баз данных.– М.: Русская редакция, 2001.–384 с.
7. *Ролланд Ф*. Основные концепции баз данных: Пер. с англ.– М.: Вильямс, 2002.– 256 с.
8. *Фёдоров А., Елманова Н*. Базы данных для всех.– М.: Компьютер-Пресс, 2001.–256 с.
9. *Харрингтон Д. Л.* Проектирование реляционных баз данных: Просто и доступно.– М.: Лори, 2000.– 230 с.
10. *Чекалов А.П.* Базы данных: от проектирования до разработки приложений.– СПб.: БХВ–Петербург, 2003.–384 с.

Разработка БД в VFP

11. *Баженова И. Ю*. Visual FoxPro 6.0.– М.: Диалог–МИФИ, 2001.– 416 с.
12. *Каратыгин С.А. и др*. Visual FoxPro 6.– М.: БИНОМ, 1999. – 773 с.
13. *Каратыгин С.А., Тихонов А.Ф., Тихонова Л.Н*. Visual FoxPro 7.: Руководство пользователя с примерами.– М.: БИНОМ, 2003.– 656 с.
14. *Мусина В., Пушенко В. А*. Visual FoxPro 7.0. Учебный курс.– М.: Век +, BookStar, 2001.– 400 с.
15. *Омельченко Л.Н*. Самоучитель Visual FoxPro 8.– СПб.: БХВ–Петербург, 2003.– 688 с.

Учебное издание

Ирина Ивановна Семенова

РАЗРАБОТКА БАЗ ДАННЫХ В MICROSOFT VISUAL FOXPRO
Часть 1
Создание структуры базы данных

Учебно-методическое пособие

Редактор Т.И. Калинина

***

Подписано к печати
Формат 60x90 1/16. Бумага писчая
Оперативный способ печати
Гарнитура Таймс
Усл.п. л. 4,0 , уч.-изд. л. 3,8
Тираж 100 экз. Заказ № _____
Цена договорная

***

Издательство Сибирской государственной
автомобильно-дорожной академии
644099, Омск, ул. П. Некрасова, 10

Отпечатано в ПЦ издательства СибАДИ
644099, г. Омск, ул. Некрасова, 10