**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ**
**НАЦИОНАЛЬНАЯ МЕТАЛЛУРГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ**

**Г.Г. ШВАЧИЧ, А.В ОВСЯННИКОВ, В.В. КУЗЬМЕНКО,**
**А.В. СОБОЛЕНКО, В.В. БАЙРАК**

# СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

## Часть II. Основы разработки систем управления базами данных в интегрированной среде Delphi

Утверждено на заседании Учёного совета академии
в качестве учебного пособия

**Днепропетровск НМетАУ 2008**

УДК 004 (075.8)

Системы управления базами данных. Часть II. Основы разработки систем управления базами данных в интегрированной среде Delphi: Учеб. пособие / Г.Г. Швачич, А.В. Овсянников, В.В. Кузьменко, А.В. Соболенко, В.В. Байрак. – Днепропетровск: НМетАУ, 2008. – 48 с.

Изложены основы создания систем управления базами данных в интегрированной среде разработки Delphi

Предназначено для студентов специальности 6.020100 – документоведение и информационная деятельность, а также для студентов всех специальностей и иностранных студентов.

Илл. 47. Библиогр.: 5 наим.

Издается в авторской редакции.

Ответственный за выпуск Г.Г. Швачич, канд. техн. наук, проф.

Рецензенты: Б.И. Мороз, д-р техн. наук, проф. (АТСУ)
Д.Г. Зеленцов, д-р. техн. наук, доц. (УГХТУ)

# Тема 1. Категории и их описание. Информационная модель. Информационно-логическая структура. Понятия сущность и свойства, их описание

Предположим, что необходимо разработать расписание занятий на один семестр. На первый взгляд, задача очень простая, но как мы увидим далее это не совсем так. Итак, мы должны определить, какая информация должна быть введена в базу данных и сохранятся в ней. Также необходимо определить отношения либо дней недели к преподавателям и преподавателей к группам – стандартное расписание занятий, которое мы видим на доске расписаний, либо отношения занятий к преподавателям и дням недели, либо другие отношения, т.е. нам необходимо описать схему связей. Также необходимо определить сущности (таблицы) и описать (определить) их структуру.

А если рассмотреть задачу таким образом, что расписание занятий (план на семестр) – это только малая часть информационной системы «Кафедра», которая должна обеспечить предоставление полной информации о ее деятельности, включая:

- список преподавателей кафедры их должности, служебные и домашние телефоны, домашний адрес, ученую степень и т.д.;
- учебный план (программу работы) преподавателя;
- учебную нагрузку преподавателя;
- отчет преподавателя о его научной деятельности;
- отчет преподавателя о выполнении учебного плана;
- расписание занятий проводимых в помещениях кафедры;
- другую информацию, характеризующую деятельность кафедры.

Как видно из предлагаемой задачи информационная система «Кафедра» представляет собой емкую базу данных, имеющую довольно сложную структуру.

Не зависимо от того, рассматриваем мы первую или вторую системы нам необходимо определить главную категорию (информацию), относительно которой будут выстраиваться отношения с другими категориями. Если внимательно рассмотреть каждый пункт задачи, то становится очевидным, что каждая из задач, в нашем случае, является описанием деятельности преподавателя.

Итак, мы определили, что главная категория (сущность, объект, класс), относительно который мы будем строить отношения – это преподаватель. Все другие категории, такие как: «программа работы», «учебная нагрузка», «научная деятельность» относятся и описывают деятельность преподавателя.

Напомним, что понятие категория (класс) в ООП – это абстрактный тип данных, который включает собственные свойства и методы. Таким образом, категория «Преподаватель» – это список сотрудников кафедры, содержащий необходимую информацию (свойства, поля данных) о каждом сотруднике.

В общем виде информационно-логическая структура (модель) базы данных (информационной системы), описывается графом, который демонстрирует отношения категорий (рис. 1.1.). Категории могут быть простыми и сложными.


Рис. 1.1. Общий вид информационно логической структуры СУБД

Простая категория представляет собой множество данных, однозначно описывающих определённую категорию. Она является, по сути, двумерным массивом данных (таблицей), содержащей имена полей – столбцы таблицы (атрибуты) и записи – строки таблицы (кортежи). На пересечении строки и столбца таблицы находится значение атрибута (значение свойства конкретной записи).

Сложной категорией является такая категория, которая не может однозначно быть определена (описана) в двумерном представлении, т.е. представляет собой n – мерный массив.

Так, как концепция построения реляционных баз данных определяет, что категории должны быть простыми, то сложная категория должна быть представлена в виде нескольких взаимоподчиненных простых категорий. В качестве примеров сложной категории можно рассмотреть объект (класс) TForm (Delphi) имеющий свойство Pixels, определенное через свойство Canvas (Form1.Canvas.Pixels[X,Y]) или отношение шрифта (Font) для диапазона ячеек (Range) объекта (Лист1): Лист1.Range("A1:C1"). Font в среде VBA.

Если на начальной стадии проектирования базы данных затруднительно определить, является ли категория сложной, то в информационно логической схеме допускается ее представление, как условно простой категории, а детализация такой категории осуществляется на стадии описания сущностей (объектов, таблиц) и их отношений между собой.

Вернемся к первой задаче и представим модель будущей системы (базы данных «Расписание занятий») как отношение:

Преподаватель ---- **à** его расписание.

Проанализируем категорию «Преподаватель». Это простая категория, которая описывается двумя свойствами (полями) относительно требуемого объема информации, представляемой в расписании занятий:

- фамилия и инициалы;
- занимаемая должность.

Проанализируем категорию «Расписание». Данная категория содержит информацию о группах, в которых преподаватель проводит занятия.

| Дисциплина – предмет занятий, дни недели, в которые проводятся занятия и порядок, в котором эти занятия проводятся. |
|---|

Помимо указанного, существует такое понятие как «числитель/знаменатель».

Таким образом, категория «Расписание» имеет сложную информационную структуру, содержащую внутренние отношения:

День недели --- **à** расписание.

Исходя, из проведенного анализа следует, что информационно – логическая схема (модель) базы данных имеет вид:

Преподаватель --- **à** День недели - **à** Расписание.

После определения структуры информационной модели системы мы можем приступить к описанию каждой категории – сущности.

Рассмотрим, каким свойством (значением поля) можно реализовать отношение категории «Преподаватель» к другим категориям. На первый взгляд фамилия и инициалы преподавателя позволяют нам построить такое отношение, т.е. является первичным ключом (первичным индексом) будущей таблицы. Однако это не совсем так потому, что фамилии и инициалы людей довольно часто совпадают.

Например: Иванов И. А. (Иван Александрович) и Иванов И. А. (Игорь Алексеевич) являются разными людьми. Стоит вопрос, а может ли быть ключевым полем должность или в комплексе фамилия – должность (составной ключ)? Конечно же нет, так как Иванов И. А. – доцент и другой Иванов И. А. то же доцент. К тому же два разных человека могут иметь одинаковые фамилии имена и отчества. Поэтому даже если мы выделим для имени и отчества отдельные поля, то создать уникальный составной ключ невозможно, т.е. в любой момент времени функционирования системы она может дать сбой, обусловленный неоднозначностью (противоречивостью, недостоверностью) предоставленной информации. Стоит вопрос, как – же поступить в таком случае. Чтобы решить данную задачу достаточно определить, какой уникальной информацией (аутентификацией) может обладать человек. Ответ простой – это его идентификационный код, присвоенный налоговой службой и действующий на территории государства или табельный номер, действующий в рамках предприятия. Следовательно, определим, что в качестве первичного ключа мы будем использовать табельный номер, так как система должна функционировать только в рамках предприятия и в будущем, возможно, сопряжена с другими системами предприятия.

Исходя из выше изложенного материала, категорию «Преподаватель» можно описать таблицей «Список преподавателей» (рис. 1.2.).

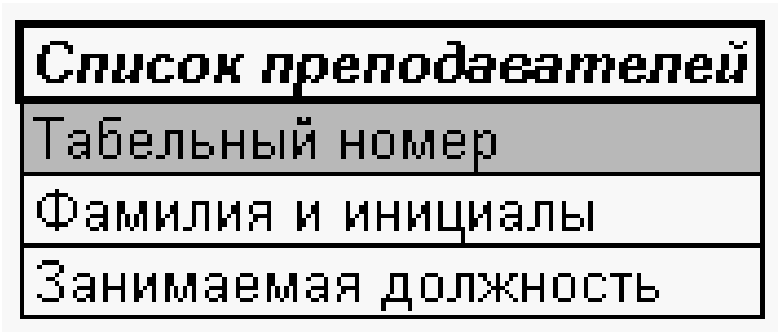


Рис. 1.2 Таблица «Список преподавателей»

Временно упростим задачу и рассмотрим ее как расписание занятий для одного дня недели, т.е. исключим из схемы категорию «День недели». В таком случае данная категория описывается полями:

- дисциплина, по которой проводятся занятия;
- группа, в которой проводятся занятия;
- порядковый номер занятия;
- № аудитории, в которой проводятся занятия;
- признак (числитель/знаменатель).

Помимо указанных полей, для обеспечения отношения с таблицей «Список преподавателей», рассматриваемая таблица должна обязательно включать поле «Табельный номер» (рис. 1.3.).


Рис. 1.3. Таблица «Расписание занятий»

Таким образом, упрощенная схема базы данных, обеспечивающая расписание занятий для одного дня будет иметь вид, представленный на рис. 1.4.


Рис. 1.4. Упрощенная схема базы данных

В приведенной схеме внешний ключ «Табельный номер» таблицы «Расписание занятий» ссылается на первичный ключ «Табельный номер» таблицы «Список преподавателей». Такое отношение называется один ко многим и позволяет представить (выбрать) информацию в виде, приведенном на рис. 1.5.

*Список преподавателей*

| Таб. № | Фамилия и иниц. | Должность |
|---|---|---|
| 45674 | Иванов А.П. | доцент |
| 53897 | Петров И.Н. | доцент |
| 78345 | Сидоров Е.Н. | ассистент |

*Расписание занятий (один день)*

| Таб. № | № занятия | Признак | Код группы | Дисциплина | Аудитория |
|---|---|---|---|---|---|
| 45674 | 1 | чс | АП-06 | Информатика | 449 |
| 45674 | 2 | зн | МО-04 | Ивф. Технол. | А-602 |
| 45674 | 4 | чс/зн | МКМ-06 | Информатика | 451 |
| 53897 | 2 | чс/зн | ДИ-05 | СУБД | 470 |
| 53897 | 4 | чс/зн | ИСД-05 | Электр. докум. | А-601 |
| 78345 | 1 | чс/зн | ТПП-06 | Информатика | 472 |
| 78345 | 3 | зн | ИСД-05 | СУБД | 470 |

Рис. 1.5. Представление отношения

Вернемся теперь к рассмотрению полной задачи и представим, каким образом реализовать расписание занятий для всех дней недели. Сразу возникает предложение создать пять таблиц «Расписание занятий» и связать их с главной таблицей «Список преподавателей» по полю «Табельный номер», т.е. каждая из таблиц «Расписание занятий» будет отражать собственную сущность – день занятия.

Построенная по такой схеме база данных будет вполне работоспособна, однако такая схема противоречит концепции реляционных баз данных – минимизации информации. Так, как каждая таблица представляет собой отдельный файл или объект в файле такая реализация приведет, в нашем случае, к затратам ресурсов

приблизительно в пять раз превышающих необходимые ресурсы. Следовательно, необходимо найти другой способ реализации.

Внимательно проанализируем, какой информацией описывается категория «День недели».

День недели может быть представлен последовательностью чисел 1, 2, 3, 4, 5 или перечисляемым типом Пн, Вт, Ср, Чт, Пт. В любом случае мы имеем дело с постоянными (константами). Такая информация называется условно постоянной, следовательно, мы можем ее использовать как справочную. Существует два варианта идентифицировать запись путем создания простой справочной таблицы (так поступают СУБД «Microsoft Access»), либо включить в схему обычное поле со списком, таким как масштаб документа в приложениях Word, Excel. Второй способ является предпочтительным, так как позволяет избежать создания дополнительной таблицы (файла). Для реализации второго способа необходимо и достаточно в состав полей таблицы «Расписание занятий» включить поле «День недели» и реализовать отношение к полю, содержащему список констант (рис. 1.6.).


Рис. 1.6. Информационная схема (структура) базы данных «Расписание занятий»

Рассмотрим еще один вопрос. Какую цель преследует отношение «один к одному», ведь по своей сути это не что иное, как продолжение таблицы? Ответов на данный вопрос несколько.

Во-первых, различные платформы ядер баз данных имеют ограничения по количеству полей таблицы. Во-вторых, часто возникает необходимость отделить blob поля, содержащие объемную информацию, такую как вложенные документы, изображения, музыку, фильмы и т.д., с целью повышения быстро-

действия системы поиска и доступа к данным. В-третьих, отделение общедоступной информации от конфиденциальной. Например, если бы мы рассматривали базу данных «Кафедра», то информацию, касающуюся личных данных преподавателя (домашний телефон, домашний адрес и т.д.) потребовалось описывать как отдельную сущность. В-четвертых, если возникла необходимость, дополнить какой-либо информацией уже существующую базу данных. Возможны и другие ответы на поставленный вопрос.

# Тема 2. Выбор платформы базы данных. Разработка структуры таблиц. Структурная схема базы данных

Прежде чем приступить к разработке структурной схемы базы данных необходимо выбрать платформу с поддержкой которой будет функционировать СУБД. От того, какая платформа определена, зависит производительность СУБД, ее архитектура, функциональные возможности и стоимость, как разработки, так и эксплуатации.

Для правильного выбора платформы необходимо, прежде всего, проанализировать информацию (типы данных и их значения), возможные объемы хранимой информации, функциональные особенности СУБД, администрирование системы, требования к надежности, финансовые затраты, связанные с приобретением лицензии и поддержкой в эксплуатации.

Правильный выбор среды разработки приложения также определяет время разработки системы, финансовые затраты и возможность дальнейшего развития системы.

## 2.1. Требования к информации

По существу в таблицах нашей базы данных в основном необходимо хранить текстовую информацию с размерами строки, не превышающими 255 символов, т.е. мы будем использовать строковый тип данных. Единственным числовым полем является «Номер занятия» в таблице «Расписание занятий». Объем информации, который предполагается хранить в рабочем цикле обновления

данных, достаточно мал. Таким образом, никаких ограничений для любой платформы по данному пункту нет.

### 2.2. Требования к функциональности

С точки зрения функционального назначения, расписание занятий представляет собой справочную систему, которая может быть реализована на одном компьютере как персональная. Но с точки зрения удобства пользования желателен многопользовательский режим функционирования, в котором внесение изменений в расписание различными людьми не предусматривается. Планирование занятий выполняет, как правило, один человек. Поэтому определим, что база данных должна представлять собой локальную систему, в которой существует много пользователей и один администратор. Внесение данных в систему и их изменение выполняет только администратор, а другим пользователям доступен только просмотр информации. Такая система может быть реализована на основе стандартной сети без использования технологии MIDAS (удаленный доступ) и без наличия в системе выделенного сервера.

### 2.3. Требования к администрированию

Система «Расписание занятий» специальных требований к администрированию не предъявляет. Администрирование может быть реализовано отсутствием возможностей модификации данных в приложении пользователя (только просмотр).

### 2.4. Требования к надежности

Специальных требований к надежности функционирования системы, сохранности и достоверности информации не предъявляется, так как отсутствуют риски связанные с финансовыми и другими потерями. Надежность функционирования системы определяется надежностью операционной системы и сетевого обеспечения.

На основании проведенного анализа можно сделать вывод, что для реализации СУБД «Расписание занятий» может быть выбрана любая из общедоступных недорогих платформ, таких как dBASE и Paradox. Платформа Paradox является предпочтительной, так как в платформе dBase отсутствуют первичные ключи, реализация которых возлагается на разработчика СУБД.

## 2.5. Выбор среды разработки

Также как и качественный проект базы данных, правильный выбор среды разработки предопределяет успех самой разработки.

Вследствие того, что на стадии разработки нашего проекта определена платформа Paradox корпорации Borland inc., и последняя поддерживается ядром баз данных IDAPI, процессора Borland Database Engine, а также, учитывая тот факт, что разрабатываемое приложение должно функционировать под управлением операционной системы Windows**,** представляется выбор одной из двух общедоспупных сред разработки приложений Builder C++ и Delphi вышеуказанной корпорации.

Следует отметить, что реализовать рассматриваемый проект возможно и в таких средах, как Visual C++, Visual FoxPro, Visual Basic корпорации Microsoft и др. Однако это потребует подключения нестандартных ActiveX объектов (OCX), третьих фирм и динамических библиотек (DLL). К тому же возможности указанных сред значительно уступают возможностям Builder C++ и Delphi.

Обе среды разработки приложений Builder C++ и Delphi представляют мощные системы разработки корпоративных проектов и систем управления данными. Среды Builder C++ и Delphi отличаются друг от друга технологией взаимодействия с операционной средой и языками программирования.

Обе среды разработки включают в свой состав большое количество объектов или компонентов, а также имеют возможность подключения мощных объектов или компонентов поддержки баз данных третьих компаний, что трудно сделать выбор между указанными средами. С точки зрения выбора языка программирования языки C++ и Object Pascal оба настолько мощны, что спор специалистов продолжается и по сей день.

Тем не менее, все чаще базовой средой разработки корпоративных проектов известные программные компании выбирают Delphi. Тот факт, что Delphi “пишется” на Delphi, т.е. Delphi имеет открытую для разработчика архитектуру и исходные тексты самой среды Delphi**,** и то, что последние версии Delphi поддерживают практически все известные информационные технологии и стандарты предопределяет выбор профессиональных разработчиков в пользу Delphi.

Также следует отметить тот факт, что такие известные корпорации как Microsoft в отличие от корпорации Borland скрывают от пользователя не только

исходные тексты, а целые классы, их методы и заголовки классов и методов, что делает затруднительным понимание возможностей их разработок.

Другой важной особенностью среды Delphi является элемент управления – компонент. В отличии от ActiveX объекта C++ компонент Delphi не распространяется совместно с приложением, не требует регистрации своих классов в Os Windows, а компилируется непосредственно в исполняемый EXE файл и только в той его части, программный код которого задействован в разрабатываемом приложении. Какая технология обеспечивает компактность разработанного приложения и высокую надежность его функционирования.

Среда Delphi обладает уникальным компилятором, позволяющим создавать исполняемый файл практически не отличимый от программы написанной на языке низкого уровня Assembler и транслированном непосредственно в машинный код. В то же время компиляторы с языка C++ создают исполняемый файл в p–коде, интерпретируемом виртуальной p- машиной. Естественно, приложение разработанное в среде Delphi функционирует на «родном» для процессора ЭВМ машинном языке значительно быстрее, чем приложение скомпилированное в p-код.

Исходя из изложенного, определим Delphi средой реализации нашего проекта.

## 2.6. Структура формата Paradox

Прежде чем приступить к разработке структурной схемы базы данных необходимо ознакомится с правилами создания таблиц в формате Paradox. Имя поля в таблице формата Paradox представляет собой строку, написание которой подчиняется следующим правилам:

- имя поля может содержать не более 25 символов;
- имя поля не должно начинаться с пробела, но может содержать пробелы;
- имя поля не должно содержать квадратные, круглые или фигурные скобки, тире, а также знаки больше и меньше;
- имя поля не должно быть только символом #, хотя этот символ может присутствовать в имени среди других символов;
- не рекомендуется в имени поля использовать точку (.), так как она зарезервирована в Delphi для других целей.

**Поля таблиц формата Paradox могут иметь следующий тип:**

- Alpha – строка длиной 1-255 байт, содержащая любые печатаемые символы.
- Number – числовое поле длиной 8 байт, значение которого может быть положительным и отрицательным. Диапазон чисел представляется от $10^{-308}$ до $10^{308}$ с 15 значащими цифрами.
- $ (Money) – числовое поле, значение которого может быть положительным и отрицательным. По умолчанию, данное поле форматировано для отображения десятичной точки и денежного знака.
- Short – числовое поле длиной 2 байта, которое может содержать только целые числа в диапазоне от -32768 до 32767.
- Long Integer – числовое поле длиной 4 байта, которое может содержать целые числа в диапазоне от -2147483648 до 2147483648.
- # (BCD) – числовое поле, содержащее данные в формате BCD (Binary Coded Decimal). Скорость вычислений значений в данном формате немного меньше, чем в других числовых форматах, однако, точность вычислений значительно выше. Поле может содержать 0-32 знака после десятичной точки.
- Date – поле даты длиной 4 байта, которое может содержать дату от 1 января 9999 г. до нашей эры - до 31 декабря 9999 г. нашей эры. Корректно обрабатывает високосные года и имеет встроенный механизм проверки правильности даты.
- Time – поле времени длиной 4 байта, содержит время в миллисекундах от полуночи и ограничено 24 часами.
- @ (Timestamp) – обобщенное поле даты длиной 8 байт - содержит и дату и время.
- Memo – поле для хранения текста. Может иметь любую длину. Размер, указываемый при создании таблицы, означает количество символов, сохраняемых в таблице (1-240) – остальные символы сохраняются в отдельном файле с расширением .MB.
- Formatted Memo – поле, аналогичное полю Memo, с добавлением возможности задавать шрифт текста. Также может иметь любую длину. При этом размер, указываемый при создании таблицы, означает количество символов, сохраняемых в таблице (0-240) – остальные символы сохраняются в отдельном файле с расширением .MB.

- Graphic – поле, содержащее графическую информацию. Может иметь любую длину.
- OLE – поле, содержащее OLE (Object Linking and Embedding) объекты: звук, видео, а также документы, которые для своей обработки вызывают создавшее их приложение. Данное поле может иметь любую длину.
- Logical – поле длиной 1 байт, которое может содержать только два значения – T (true) или F (false). Допускаются строчные и прописные буквы.
- **(+)** Autoincrement – автоинкрементное поле длиной 4 байта, содержащее не редактируемое (read-only) значение типа: long integer. Значение этого поля для каждой новой записи автоматически увеличивается на единицу. Начальное значение поля соответствует 1. Применение этого поля удобно для создания уникального идентификатора записи.
- Binary – это поле, содержащее любую двоичную информацию. Может иметь произвольную длину. При этом размер, указываемый при создании таблицы, означает количество символов, сохраняемых в таблице (0-240) – остальные символы сохраняются в отдельном файле с расширением .**MB**.
- Bytes – данное поле предназначено для хранения двоичной информации, представляет собой строку цифр длиной 1-255 байт.

## 2.7. Структурная схема базы данных «Расписание занятий» в формате Paradox

Структурную схему базы данных «Расписание занятий», реализующую все необходимые отношения, представим в виде, приведенном на рис. 2.1. Такое представление полностью соответствует структуре дизайнера таблиц утилиты DataBase Desktop, что значительно облегчает процедуру создания таблиц.

В формате Paradox приняты следующие правила построения структур таблиц:

- ключевое поле (первичный ключ) должно всегда быть первым полем таблицы;
- если необходимо реализовать отношение один ко многим, то в подчиненной таблице первым должно быть автоинкрементное поле (счетчик), являющиеся ключевым полем. Поле, которое обеспечивает связь с главной таблицей (внешний ключ) должно быть индексированным (вторичный индекс);

- поля, по которым предполагается осуществлять поиск или выполнять сортировку данных должны быть индексными полями;
- желательно все индексированные поля располагать в начале таблицы.

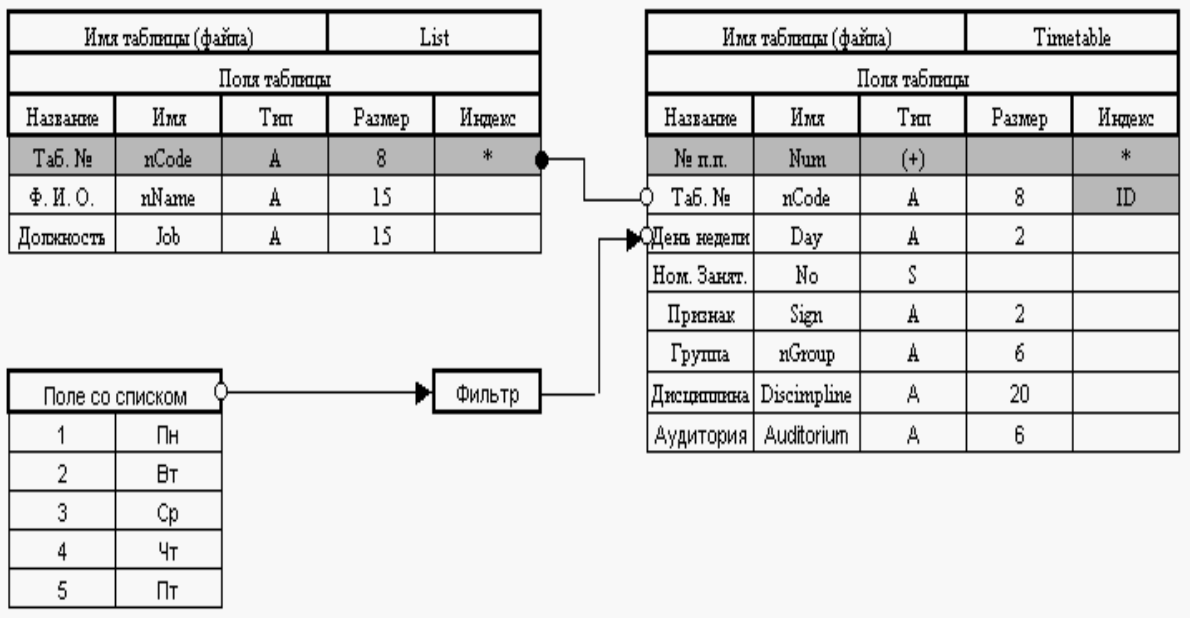


Рис. 2.1. Структурная схема базы данных в формате Paradox

Структурная схема базы данных, разработана на основе проведенного анализа. Выбранная платформа и среда разработки являются основанием для конструирования приложения, реализующего СУБД, т.е. Техническим заданием на разработку системы «Расписание занятий».

## ТЕМА 3. ПРИМЕР РАЗРАБОТКИ СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЮ ДАННЫХ

**Прежде чем приступить к разработке приложения необходимо создать рабочую папку на диске, например ME_06_LAB_1, в которой будут размещаться файлы проекта и в дальнейшем скомпилированный файл приложения (exe). Для размещения таблиц будущей базы данных рекомендуется создать внутри рабочей папки вложенную папку с именем DATA.**

### 3.1. Создание алиаса (псевдонима)

### Создание таблиц, индексирование полей таблиц

**Алиас** – это имя (идентификатор) посредством которого указывается путь к таблицам базы данных. Поэтому прежде, чем приступить к созданию таблиц рекомендуется для папки DATA присвоить алиас. Алиас должен быть уникальным для каждой базы данных, например: ME_06_IVANOV. Алиас удобно создать, используя утилиту *DataBase Desktop*, вызов которой возможен из программой группы *Delphi* главного меню.

Для вызова окна *Alias Manager* (рис. 3.1.) необходимо выполнить команду *Alias Manager* меню *Tools*.

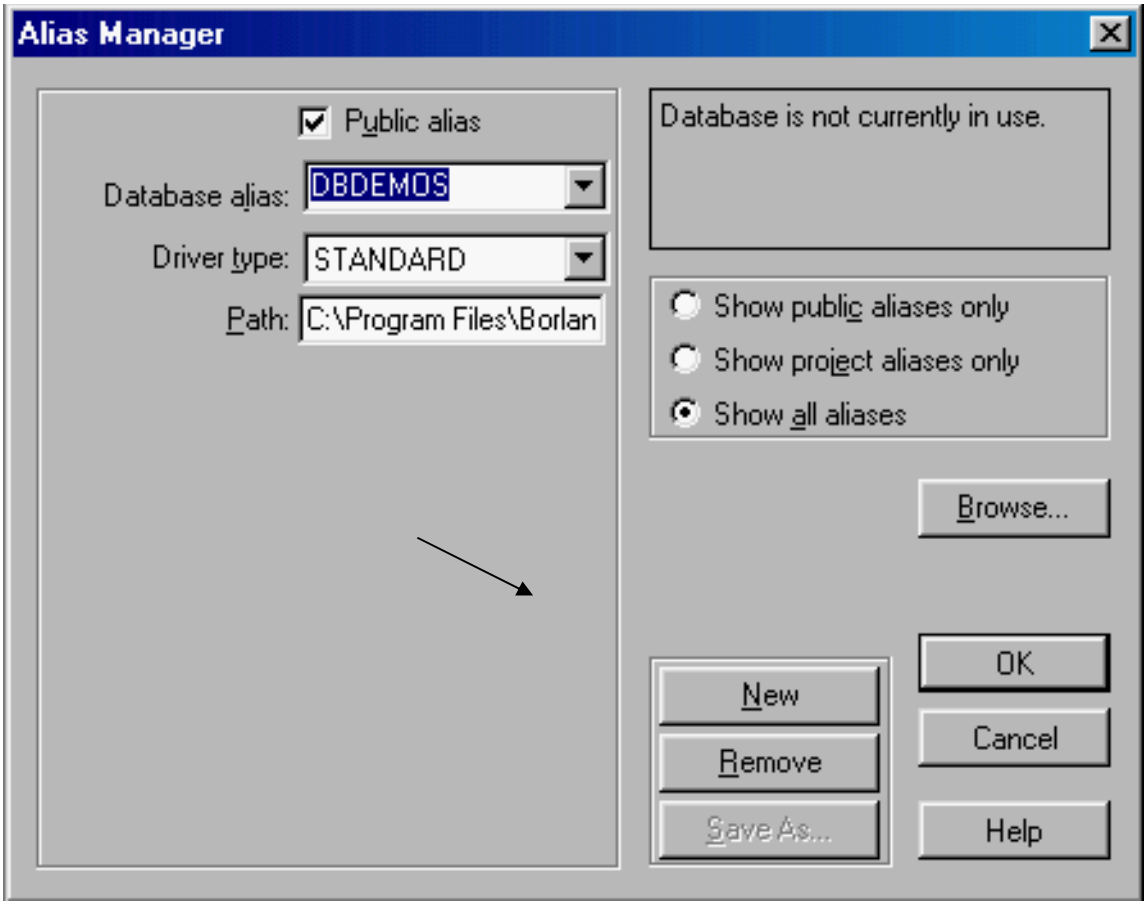


Рис. 3.1. Alias Manager

Далее выполнить щелчок по кнопке *New* ввести в окно *Database Alias* значение нового псевдонима, в нашем случае ME_05_IVANOV (рис. 3.2.).

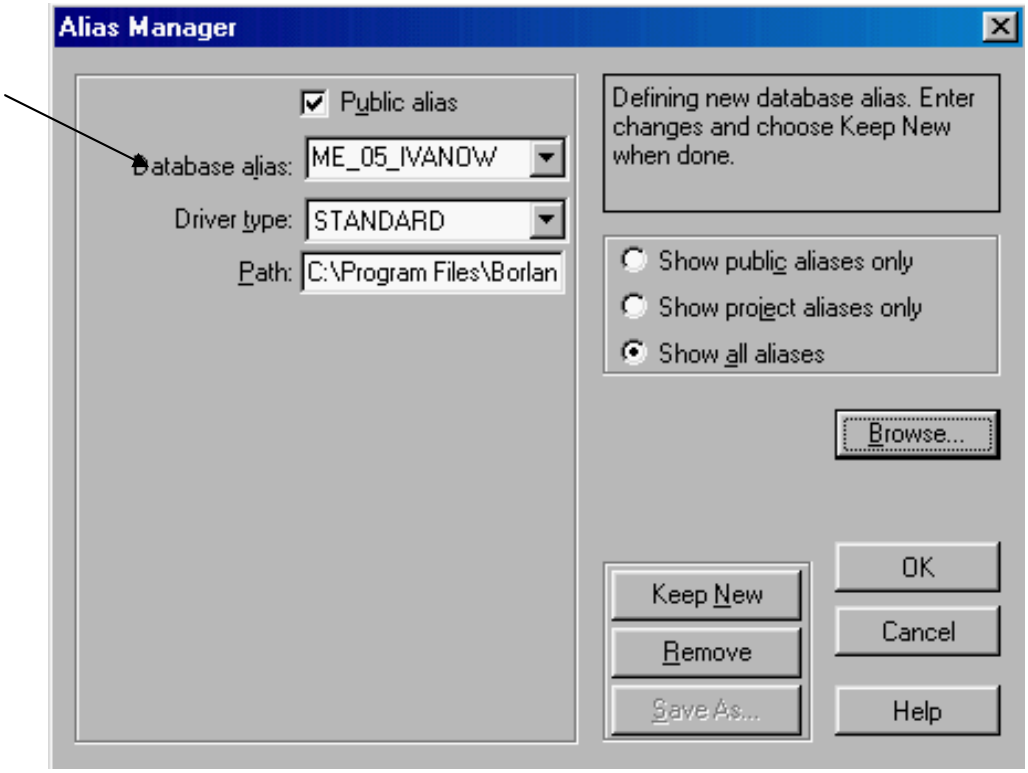


Рис. 3.2. Ввод значения алиаса

Затем необходимо выполнить щелчок по кнопке *Browse* и в дереве списка папок выбрать папку DATA и нажать на кнопку OK (рис. 3.3.).

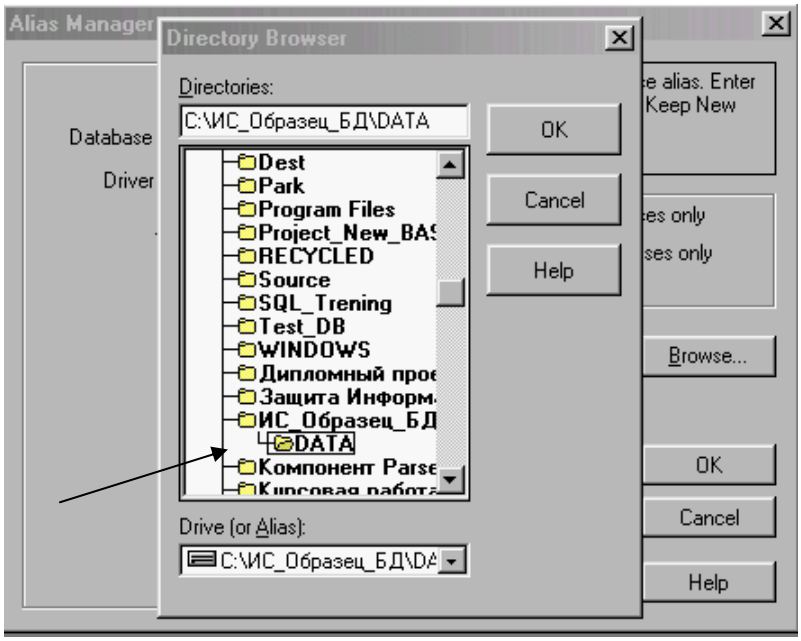


Рис. 3.3. Указание папки базы данных

Запись нового псевдонима в файл конфигурации *IDAPI* будет выполнена после подтверждения записи (Кнопка «Да») – рис. 3.4.

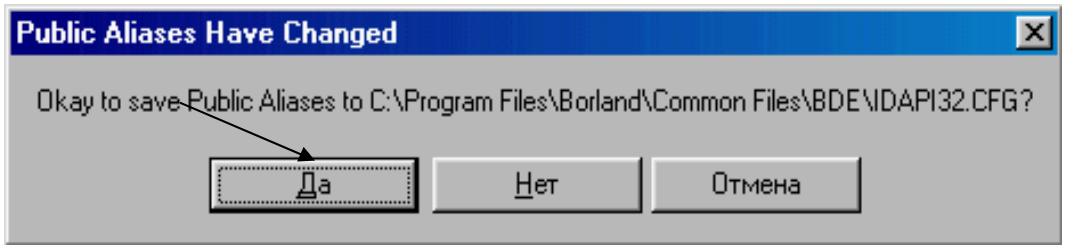


Рис. 3.4. Подтверждение нового алиаса

### 3.2. Создание таблиц

Таблицы будущей базы данных создаются также при помощи утилиты ***DataBase Desktop***, согласно их структурных схем. Порядок создания таблиц – произвольный.

Для создания новой таблицы необходимо выполнить команду ***New/Table*** меню ***File.*** В результате выполнения указанного действия будет выведено окно выбора возможных платформ (типов) СУБД. По умолчанию предлагается платформа ***Paradox 7,*** которой мы воспользуемся (рис. 3.5.).

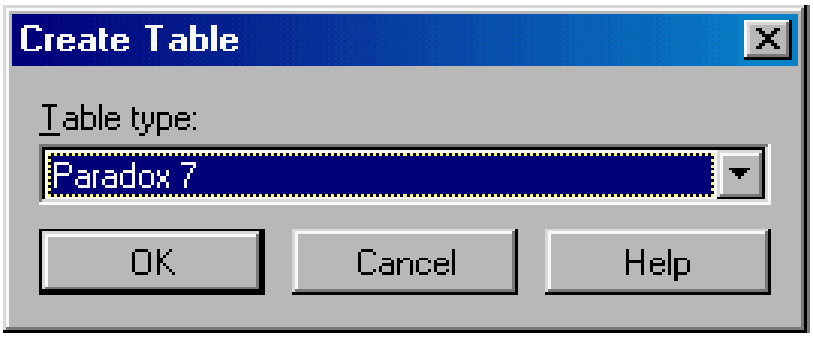


Рис. 3.5. Окно выбора платформ

В результате подтверждения выбора платформы будет выведена форма конструктора таблиц (рис. 3.6.).

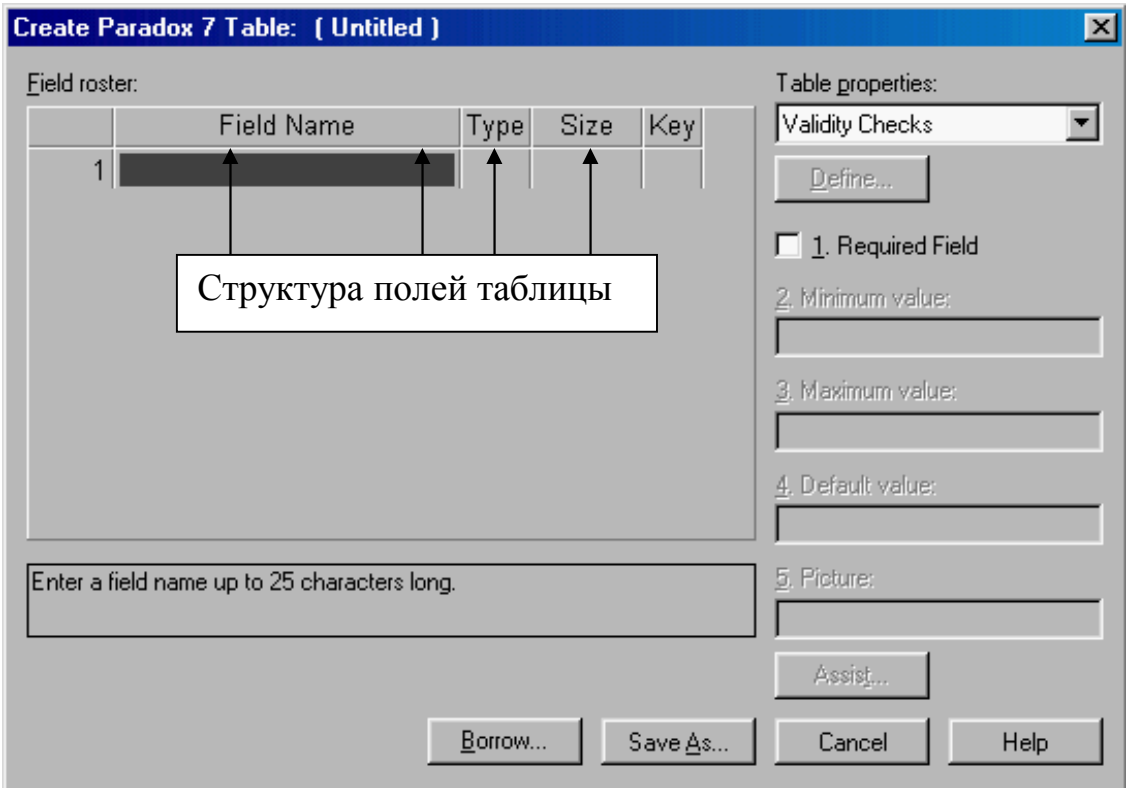


Рис. 3.6. Форма конструктора таблиц

Описание структуры таблицы выполняется в следующей последовательности:

- в поле ***Field Name*** вводятся имена полей таблицы;
- в поле ***Type*** указывается тип данных поля (*тип поля можно выбрать в контекстном субменю из списка типов полей или набрать на клавиатуре символ первой буквы типа поля*);
- в поле ***Size*** указывается размер поля (*количество хранимых символов или знаков*) только для полей, для которых данный параметр предопределен;
- в поле ***Key*** устанавливается первичный ключ. Данная операция выполняется двойным щелчком указателя курсора мыши по полю. (*Первичным ключом может быть только первое поле таблицы*).

Перемещение по записям удобно выполнять клавишей [*Tab*] клавиатуры, а завершать ввод клавишей [*Enter*].

После описания структуры таблицы (рис. 3.7.) ее необходимо сохранить, для чего достаточно щелкнуть по кнопке *Save As* и в диалоговом окне (рис. 3.8.) ввести в поле *Имя файла* имя таблицы, а в поле *Alias* выбрать свой псевдоним.

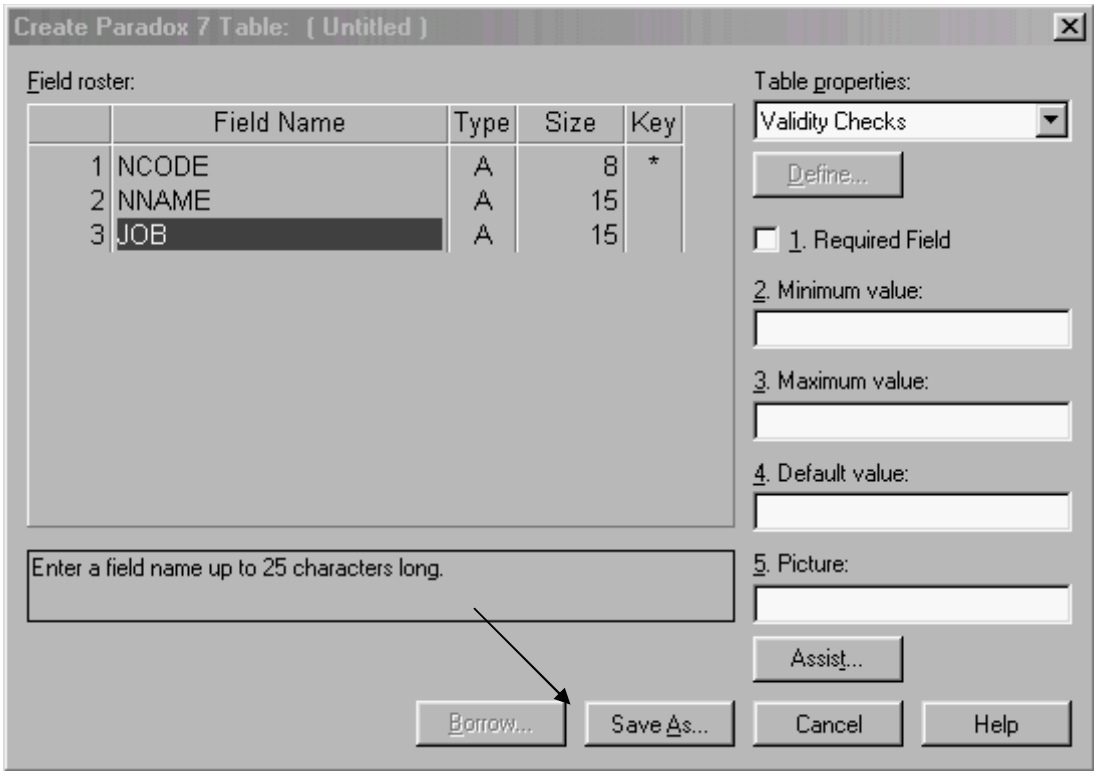


Рис. 3.7. Структура таблицы List

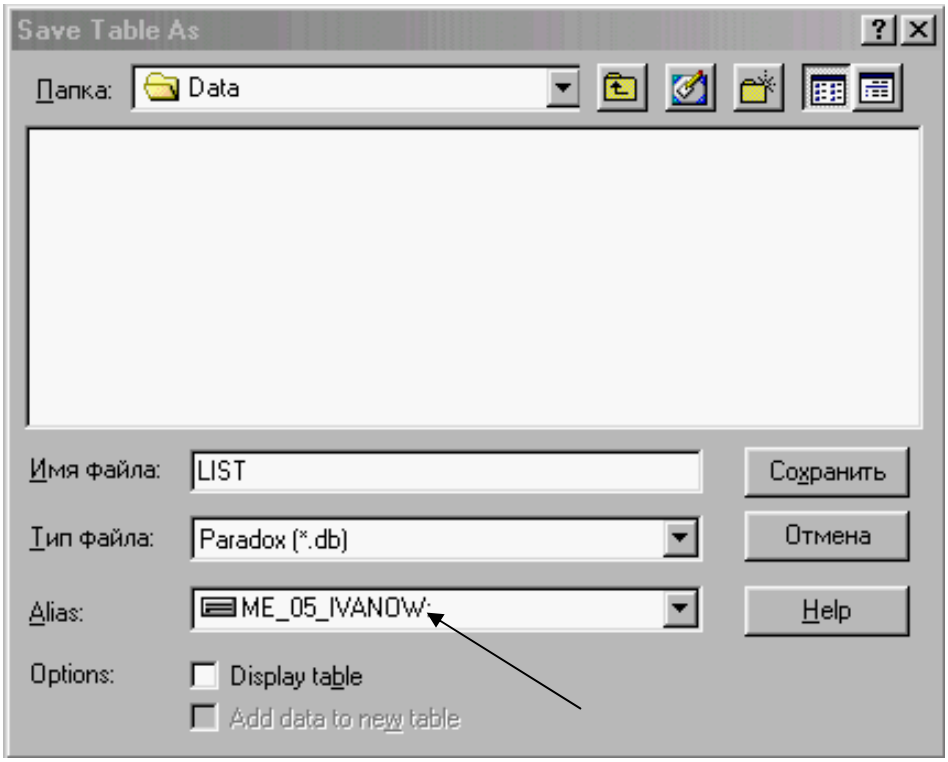


Рис. 3.8. Диалоговое окно сохранения файла

Другие таблицы создаются в ранее описанной последовательности. ***Индексирование полей***

Индексирование полей может быть выполнено как в процессе описания их структуры, так и после сохранения их как файлов. Для присвоения полям соответствующих индексов достаточно выбрать из списка свойств *Table properties*, значение *Secondary Index* и щелкнуть по кнопке *Define* (рис. 3.9.).

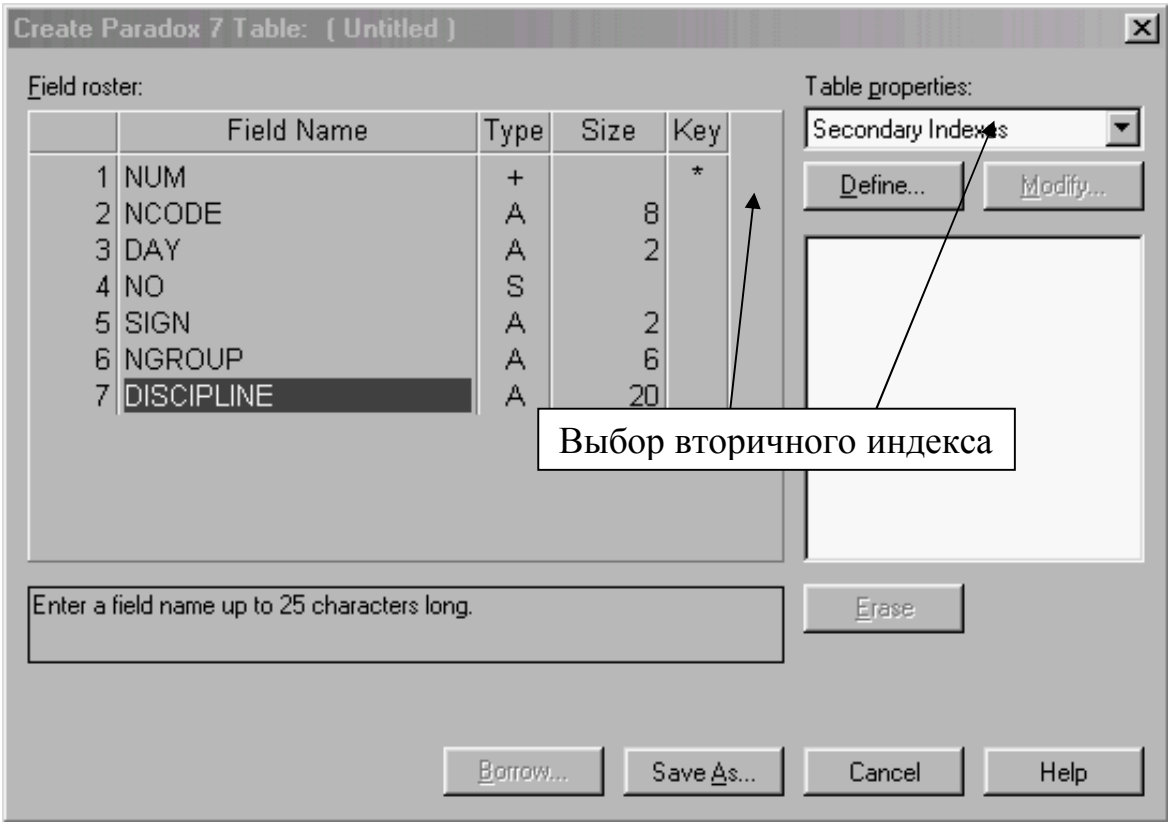


Рис. 3.9. Диалоговое окно сохранения файла

Далее, воспользовавшись кнопкой со стрелкой [->], перенести необходимое поле из списка полей *Fields* в список полей *Index Fields* (рис. 3.10.). В случае если создается комплексный индекс для группы *полей* (*реализация отношения многие ко многим*), то необходимо в окно *Index Name* переместить все поля, участвующие в групповой выборке.

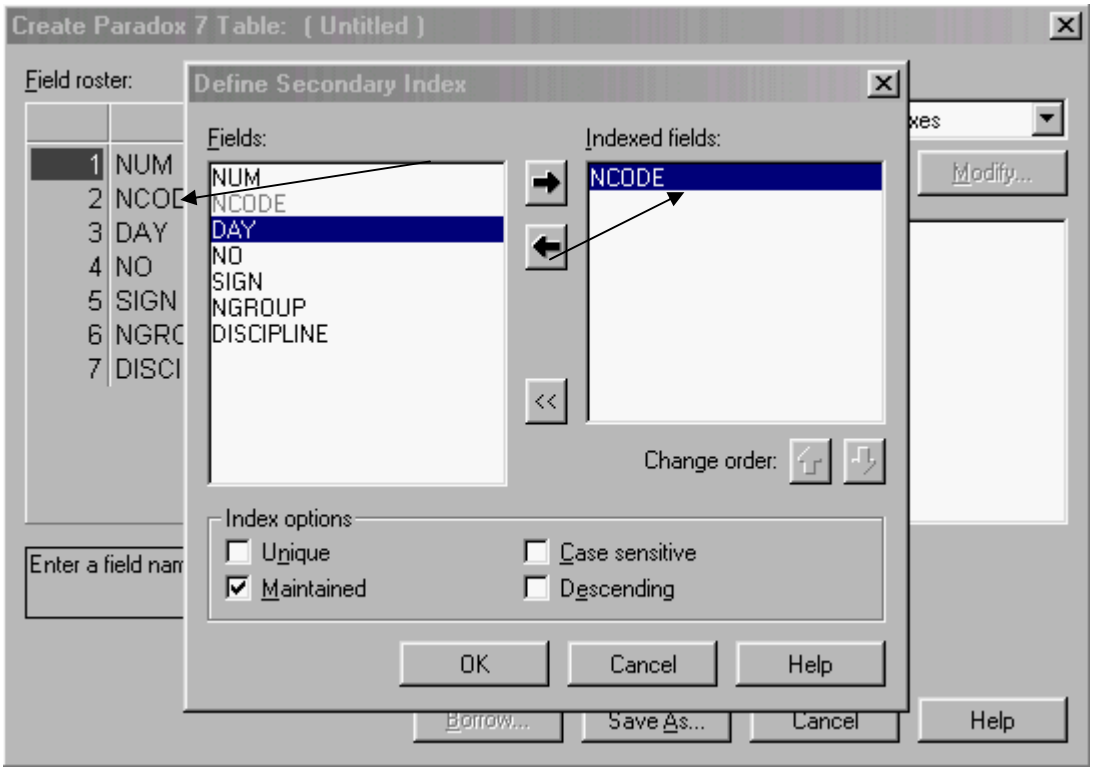


Рис. 3.10. Создание индекса

После подтверждения (*выполнения команды OK*) будет выведено окно присвоения имени индексу (рис. 3.11.). Индекс может иметь любое имя.

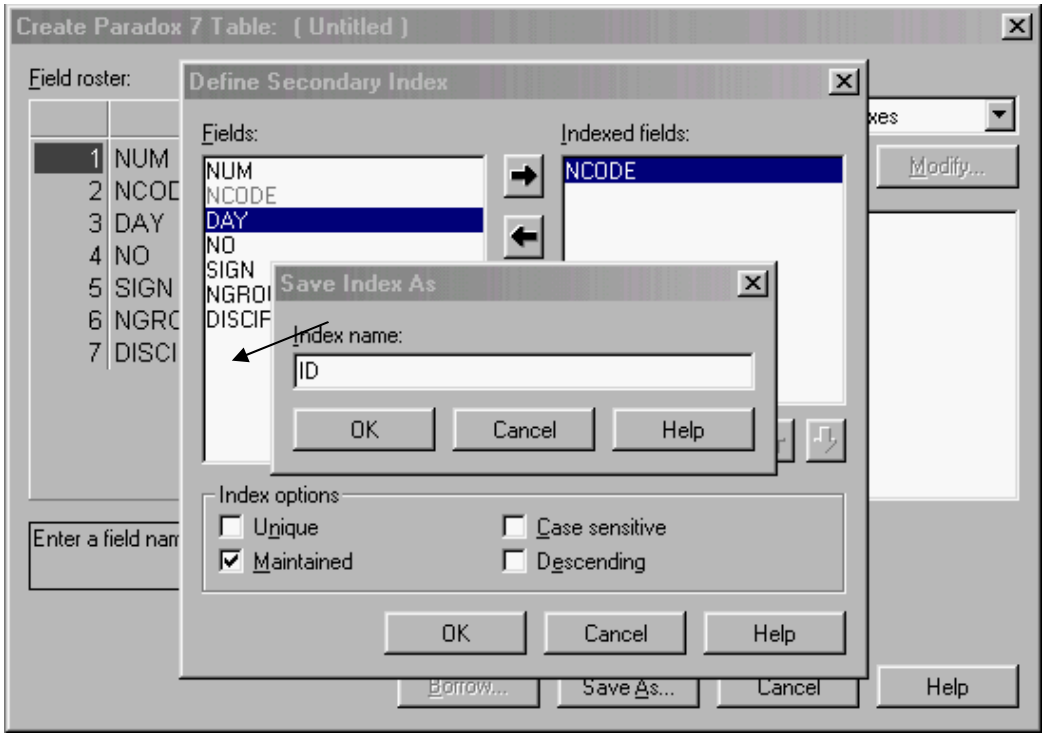


Рис. 3.11. Присвоение имени индексу

Индексированную таблицу необходимо сохранить, выполнив стандартные процедуры сохранения файлов.

## ТЕМА 4. ПРИМЕР РАЗРАБОТКИ ДИЗАЙНА ПРИЛОЖЕНИЯ. ВЫБОР КОМПОНЕНТОВ И УСТАНОВКА ИХ В ФОРМУ. СОЗДАНИЕ СВЯЗАННЫХ КУРСОРОВ

Дизайн приложения выполняется непосредственно в среде *Delphi*. В среде *Delphi* приняты следующие правила работы:

- Непосредственно после запуска *Delphi* необходимо сохранить файл проекта, который для простого приложения будет содержать собственно файл проекта и файл главного модуля (служебные файлы не рассматриваем).
- Имя файла проекта и имя главного модуля не должны совпадать и должны иметь уникальные имена (по умолчанию *Project1.dpr* и *Unit1.pas* соответственно).
- Форме приложения также должно быть присвоено уникальное имя.

Вследствие того, что данная работа выполняется студентами впервые, предопределенные в *Delphi* имена файлов и форм, изменять не будем.

Первое сохранение файлов выполним, применив команду *Save Project As* меню файл (последовательно выводятся два окна сохранения файлов). Файлы необходимо сохранить в рабочей папке проекта (в нашем случае, D:\ME_05_Иванов). В последующем, при выполнении дизайна рекомендуется регулярно выполнять сохранения файлов, используя команду *Save All*.

Дизайн приложения выполним в следующем порядке:

- Шаг 1. Установим в форму два компонента доступа к данным и связи с данными *Table* и *DataSource*, расположенные на станице *Data Access* палитры компонентов (рис. 4.1.).

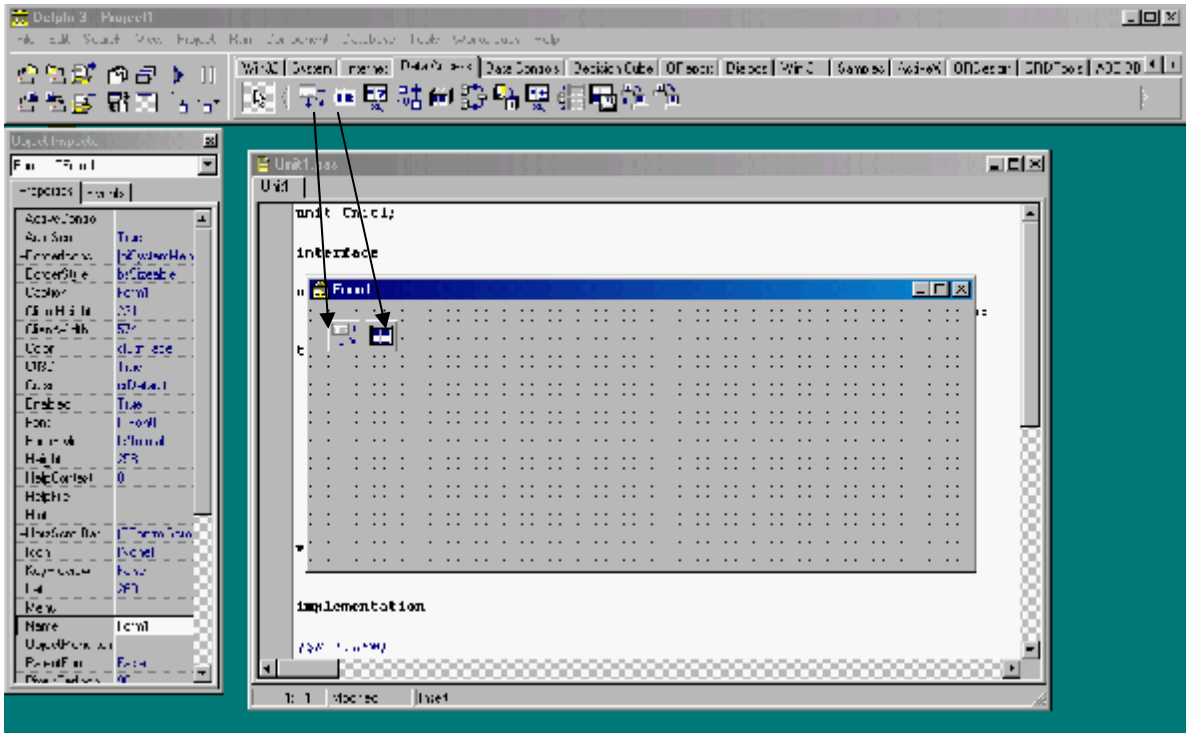

Рис. 4.1. Дизайн приложения (шаг 1)

- Выберем в форме компонент *DataSource* и в инспекторе объектов установим его связь с компонентом, посредством свойства *DataSet* (рис. 4.2.).

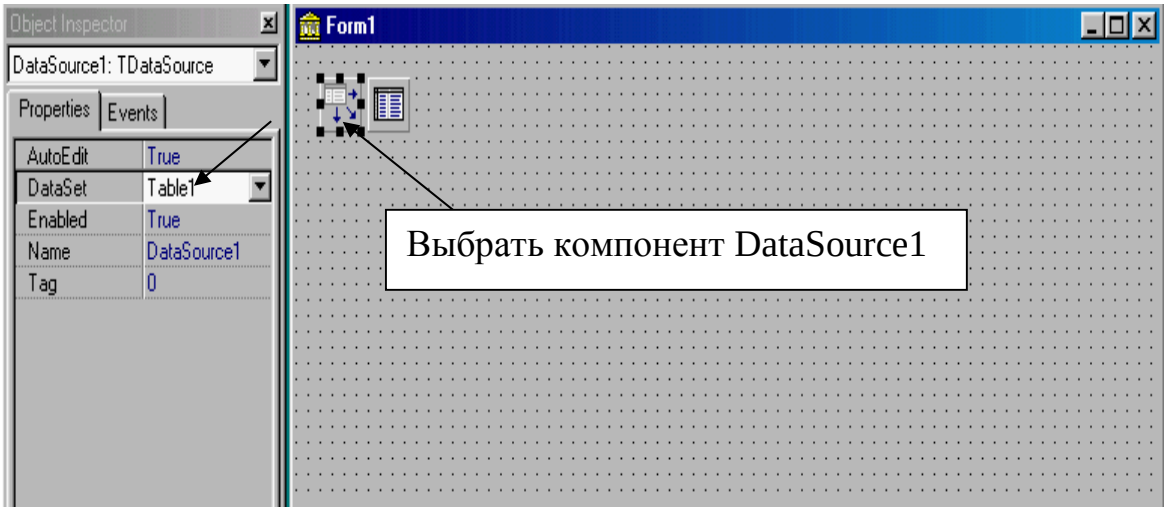


Рис. 4.2. Дизайн приложения (шаг 2)

- Шаг 3. Выберем в форме компонент *Table1* и в инспекторе объектов определим свойств *Database Name*, значение которого должно соответствовать алиасу базы данных. Значение данного свойства выбирается из списка доступных алиасов. Далее, аналогично, в свойстве *Table Name* укажем имя первой таблицы, выбрав

его из списка доступных таблиц (рис. 4.3.). Для того, чтобы в дальнейшем структура таблицы и ее содержание были визуально доступны, установим свойство *Active* в состояние *True*.

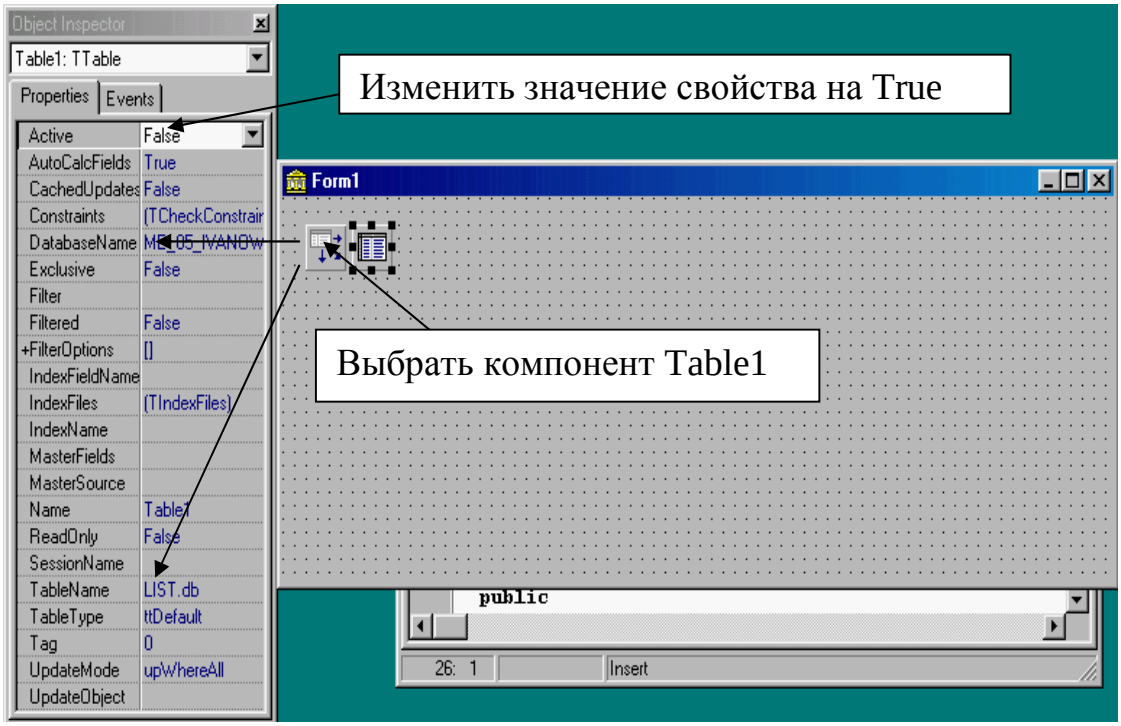


Рис. 4.3. Дизайн приложения (шаг 3)

- Шаг 4. Установим в форму элемент управления *DbGrid*, расположенных на странице *Data Controls* палитры компонентов Delphi (рис. 4.4.). Для отображения стуктуры и данных таблицы *List* (компонент *Table1*) выберем для свойства *DataSource* элемента управления *DbGrid* значение *DataSource1*. После выполнения данного действия в компоненте появится образ структуры таблицы.

**Примечание: В случае, если образ структуры не появился, то не было предварительно установлено свойство Active := True компонента Table1 в предыдущем шаге дизайна приложения.**

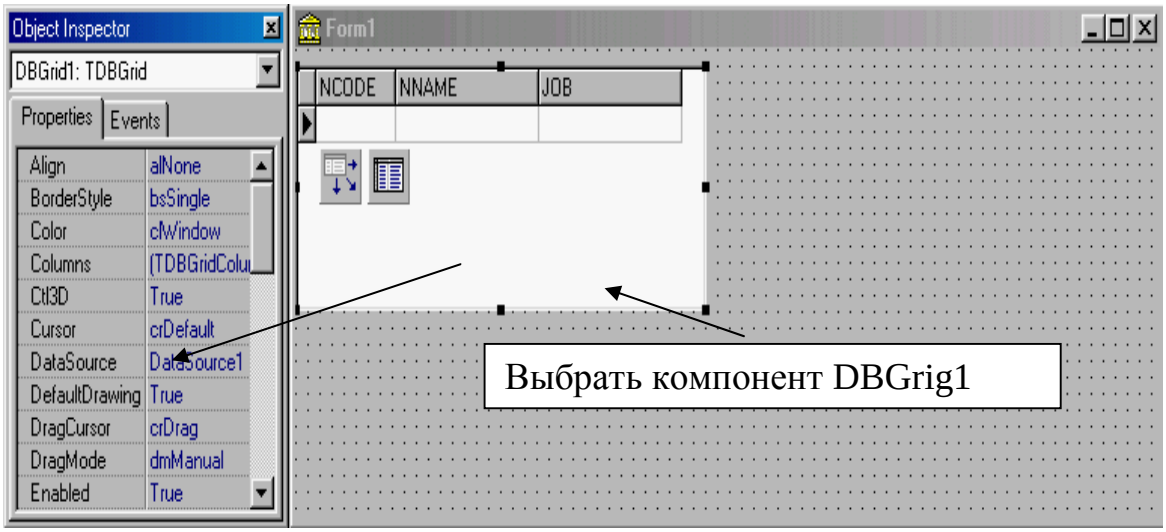


Рис. 4.4. Дизайн приложения (шаг 4)

- Шаг 5. Для обеспечения навигации и управления записью данных установим в форму элемент управления – компонент *DbNavigator* и свяжем его с источником данных посредством свойства *Datasource* (рис. 4.5.).

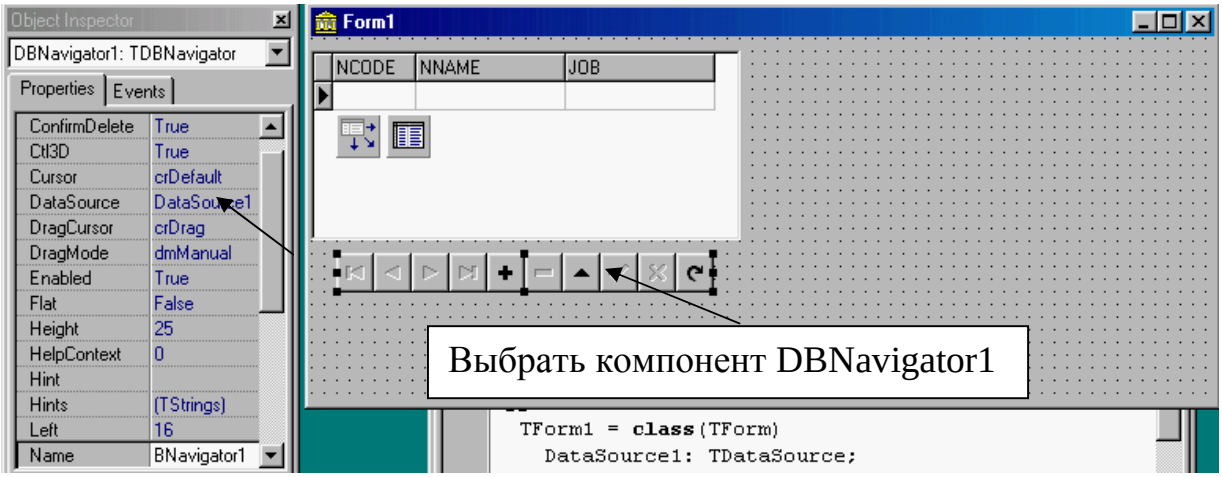


Рис. 4.5. Дизайн приложения (шаг 5)

- Шаг 6. Сохраним все файлы проекта, выполнив команду *SaveALL* меню *File Delphi* и выполним команду Run меню Run для компиляции приложения.
- Шаг 7. Установим в форму следующую пару компонентов доступа к данным (компоненты *Table* и *DataSource*), которым по умолчанию *Delphi* присвоит имена *Table2* и *DataSource2*. Свяжем указанные компоненты между собой и с таб-

лицей *TimeTable* базы данных, выполнив действия, аналогичные описанным в пунктах шаг 1 – шаг 3 настоящего описания (рис. 4.6.).

- Шаг 8. Для отображения структуры второй таблицы и управления записью данных установим в форму соответствующие компоненты *DbGrid и DbNavigator*, выполнив действия, аналогичные описанным в пунктах шаг 4 и шаг 5 (рис.4.7.).

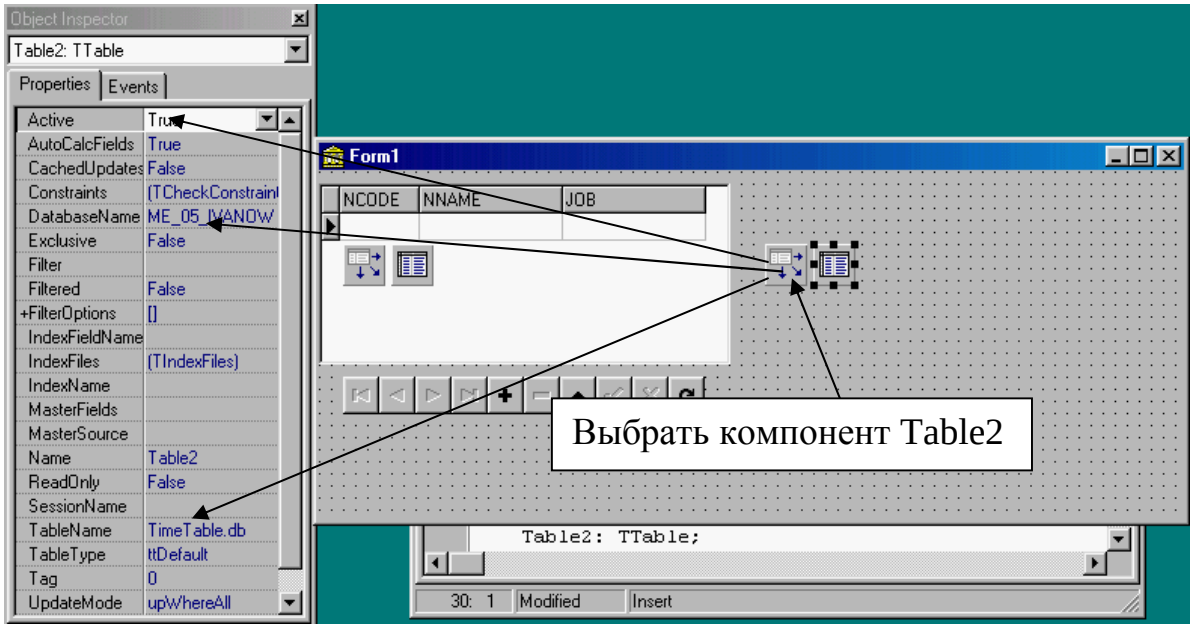


Рис. 4.6. Дизайн приложения (шаг 7)

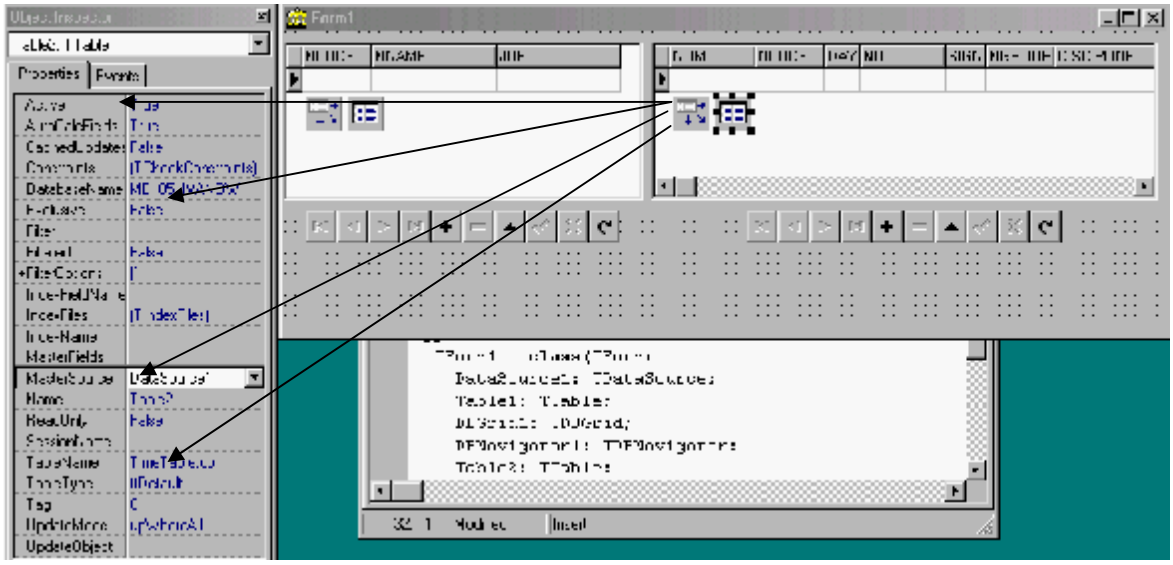

Рис.4.7. Дизайн приложения (шаг 8)

- Шаг 9. Сохраним все файлы проекта, выполнив команду *SaveALL* меню *File* Delphi и выполним команду Run меню Run для компиляции приложения.
- Шаг 10. Для создания связанных курсоров (реализация отношения один ко многим между таблицами 1 и 2) необходимо прежде всего установить значение свойства *MasterSource* компонента *Table2* в значение *DataSource1,* т.е. обеспечить связь между таблицами. Поля таблиц (только индексированные) связываются между собой посредством дизайнера связей, который вызывается при определении свойства *MasterFields* компонента *Table2*. В нашем случае для установления связи между полями NCODE таблиц *List* и *TimeTable* (структурная схема БД) необходимо в поле *Available Indexes* выбрать индекс *ID*. Затем отметить курсором мыши имя поля NCODE в списке *Detail Fields* и аналогичное имя в списке *Maser Fields* (рис. 4.8.). Далее необходимо выполнить команду добавить (кнопка ADD), которая переместит связанные поля в окно *Joned Fields* (рис. 4.9.). Установление связи завершается командой OK.

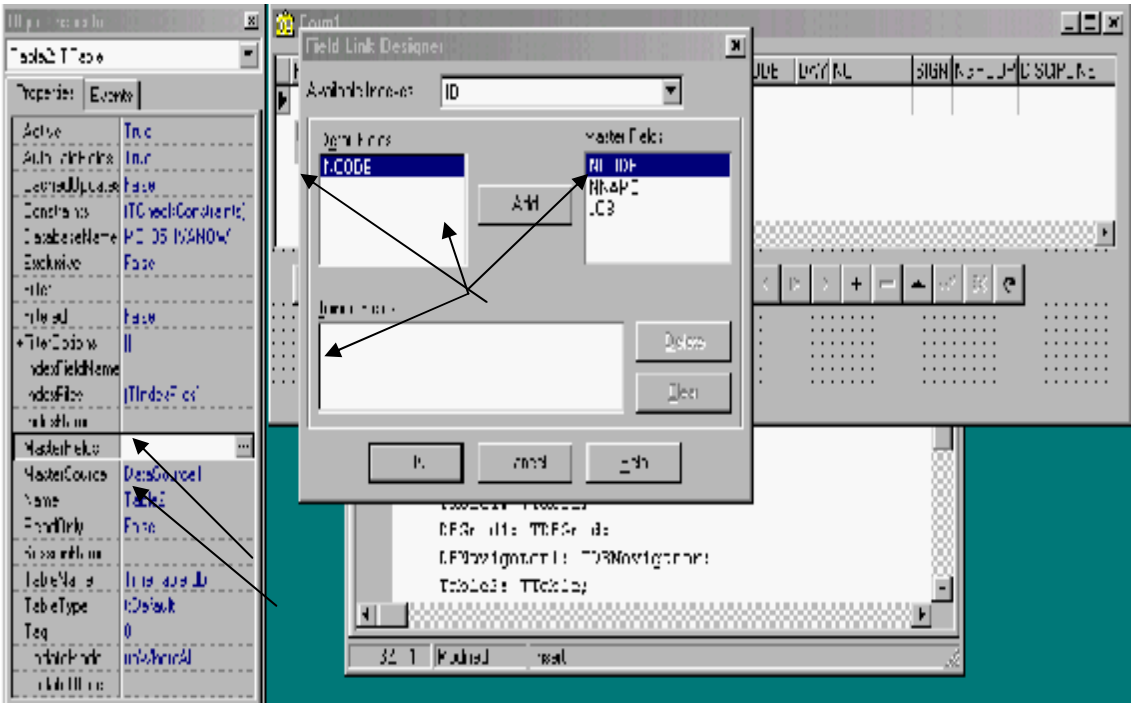


Рис. 4.8. Создание связанных курсоров

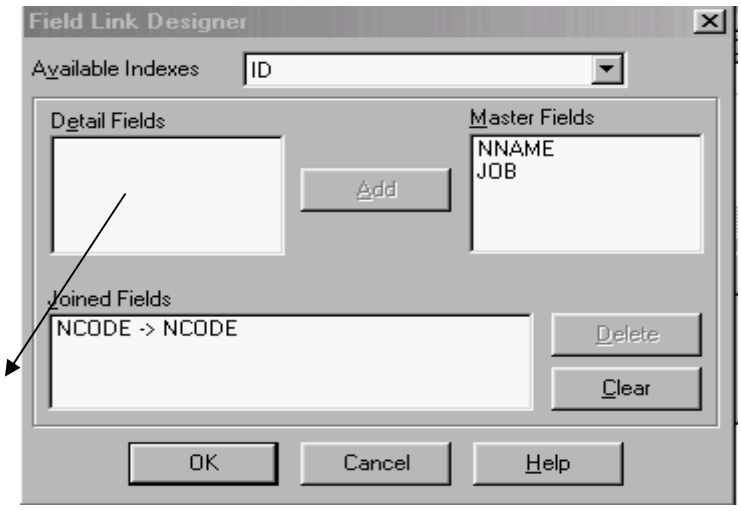


Рис. 4.9. Создание связанных курсоров

- Шаг 11. Сохраним все файлы проекта, выполнив команду *SaveALL* меню *File* Delphi и выполним команду Run меню Run для компиляции приложения.

На этой стадии первый этап дизайна приложения можно считать завершенным.

Теперь необходимо проверить функционирование созданного приложения. Функционирование приложения можно проверить, как в среде Delphi, так и автономно, без среды Delphi. Воспользуемся вторым способом. Закроем приложение *Delphi* и запустим разработанное нами приложение управления СУБД непосредственно из рабочей папки (файл Project1.exe). Запущенное приложение должно иметь вид, аналогичный виду, приведенному на рис. 4.10.

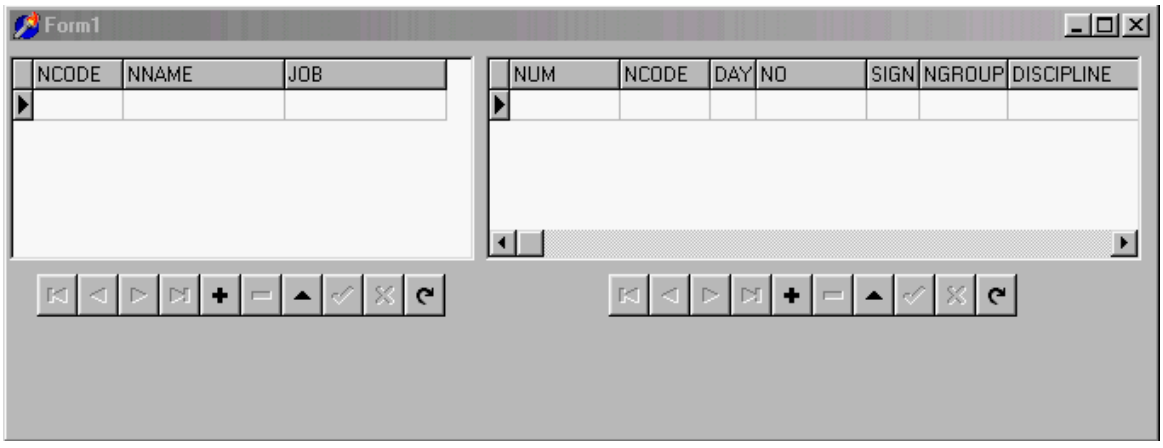


Рис. 4.10. Разработанное приложение

Первым этапом проверки работоспособности приложения является заполнение таблиц. Введите в таблицу List (список преподавателей) произвольные записи, например, такие как приведенные на рис. 4.11. Ввод записей выполняйте, используя кнопки навигатора «вставить – [+]» и «сохранить [V]».


Рис. 4.11. Ввод записей в таблицу List

Заполнение второй таблицы «Расписание» (рис. 4.12.) выполняйте относительно записей, выбранных в первой таблице. В поля NUM и NCODE данные вносить нет необходимости, т.к. поле NCODE формируется автоматически при нажатии на кнопку [+], а поле NUM при сохранении записи на диске (кнопка POST - [V]). В дальнейшем, отображение значений в автоматически заполняемых полях мы скроем.

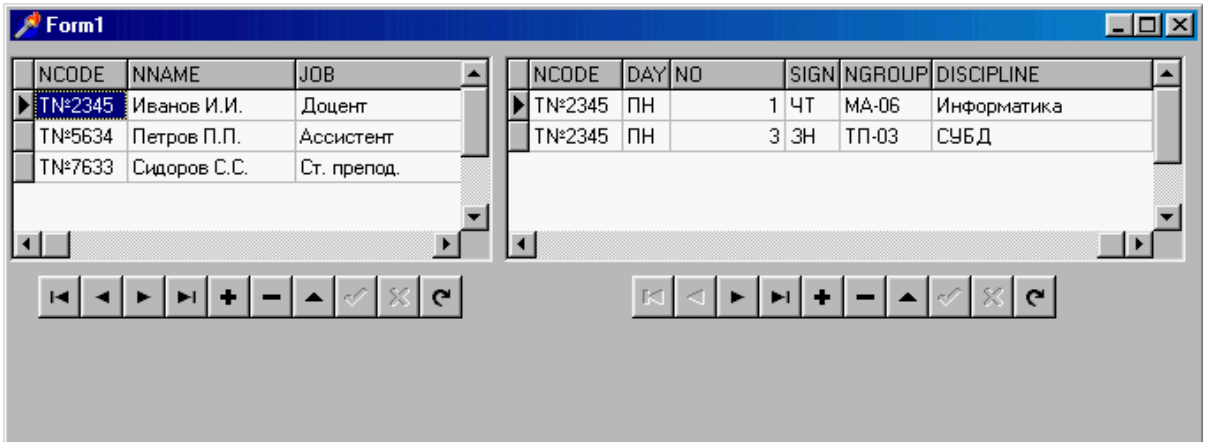


Рис. 4.12. Ввод записей в таблицу TimeTable

# ТЕМА 5. РЕАЛИЗАЦИЯ ФИЛЬТРА. ПОИСК ЗАПИСЕЙ В ТАБЛИЦЕ

Для реализации отношения «*День недели – расписание занятий*» в структурной схеме *СУБД* предложено использовать фильтр. Реализовать фильтр возможно следующим образом:

- Шаг 1. Необходимо установить в форму компонент *ComboBox*, который находится на странице *Standard* палитры компонентов *Delphi* (рис. 5.1.).

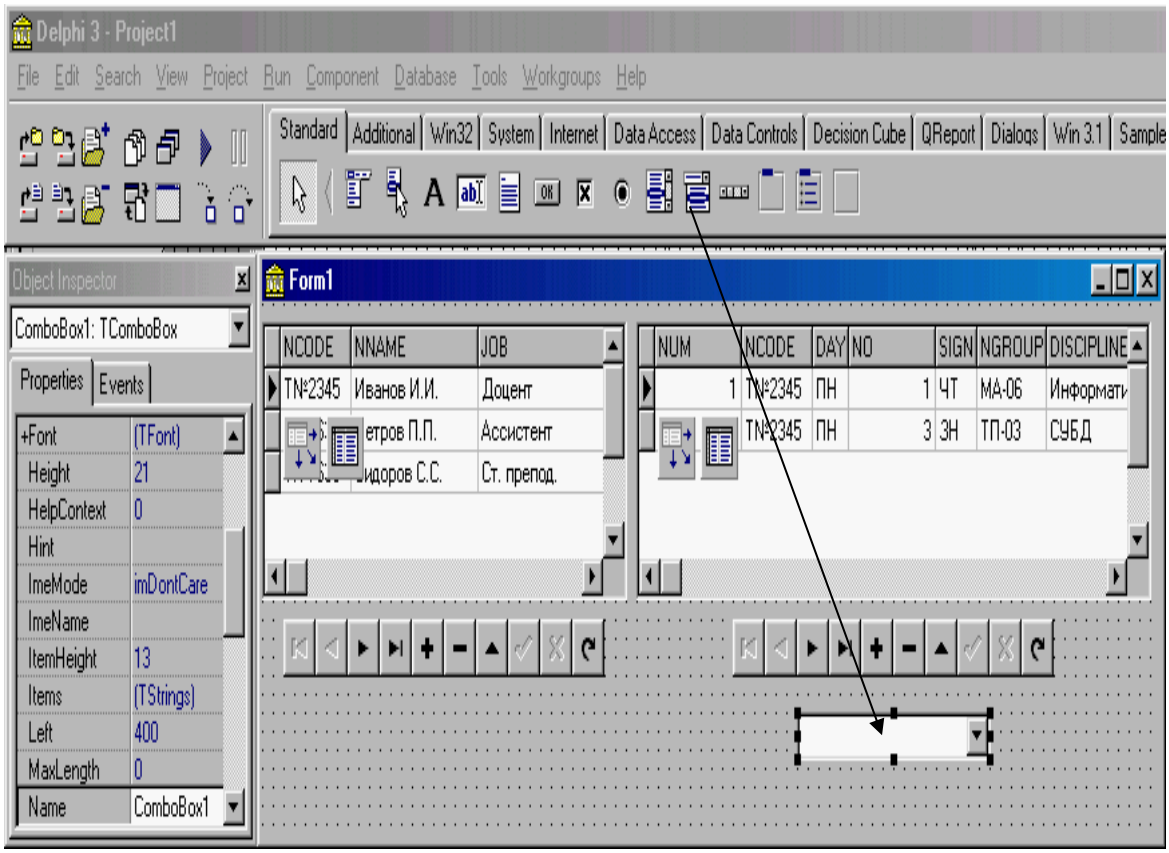


Рис. 5.1. Установка компонента ComboBox в форму

- Шаг 2. Выбрать компонент *ComboBox* в форме и вызвать редактор списка свойства *Item*, выполнив щелчок по кнопке поля *TString*. Ввести в окне редактора список дней недели (‘ПН’… ‘ПТ’) (Рис. 5.2.) и нажать на кнопку *OK*.

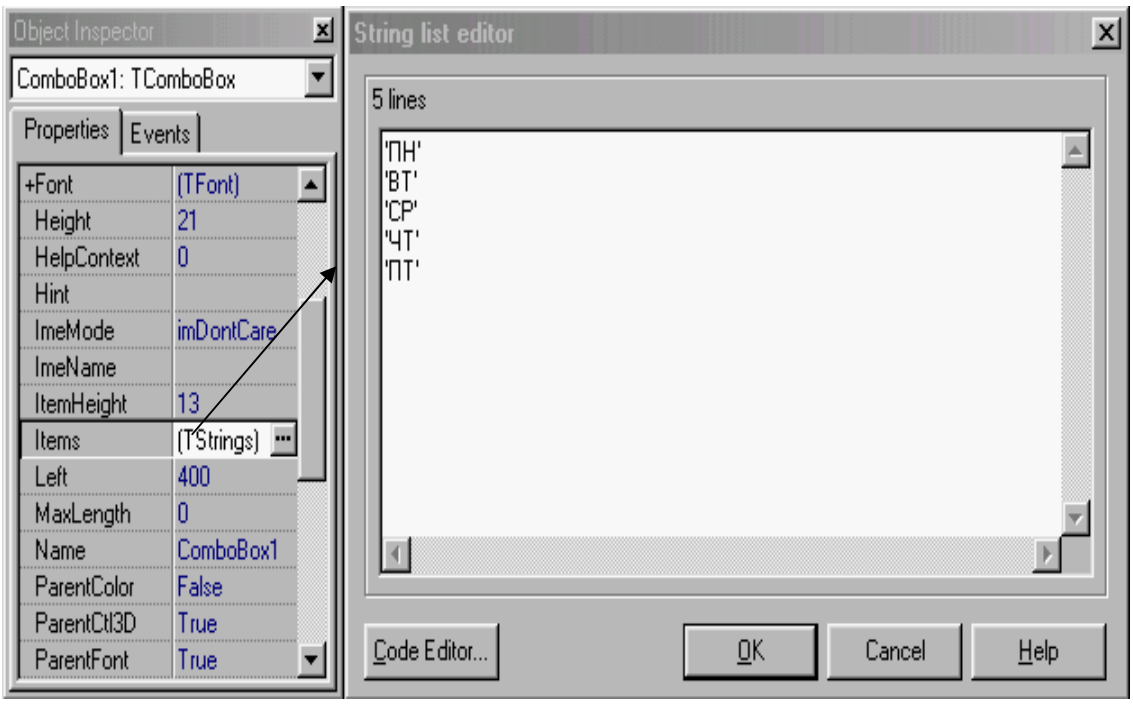


Рис. 5.2. Ввод значений в список

- Шаг 3. Присвоить свойству Text значение ‘ПН’ (ввести значение ‘ПН’ в поле Text) (рис. 5.3.).

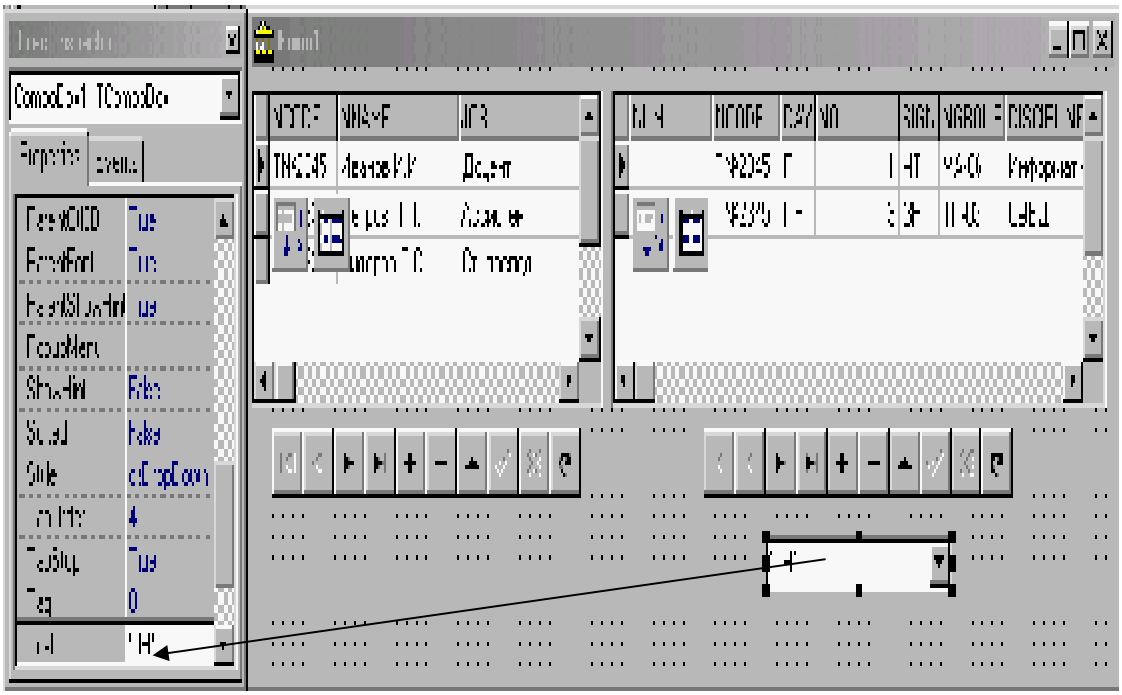

Рис. 5.3. Ввод значения в поле Text

- Шаг 4. С целью обеспечения доступа из программы к свойству *Filter* таблицы необходимо присоединить к имени таблицы поля таблицы. Данная операция выполняется в следующей последовательности:

1. Выбрать в форме компонент *Tabe2* и выполнить двойной щелчок указателем мыши по компоненту.
2. В открывшемся окне необходимо вызвать контекстное меню (правая кнопка мыши) и выполнить команду *ADD Fields.*
3. В окне добавления полей (*Add Fields*) нажать на кнопку *OK* (рис. 5.4.). После выполнения данной команды в окне *Form1 Table2* появится список всех полей таблицы (рис. 5.5.).
4. Закрыть окно *Form1 Table2* и выполнить сохранение файлов проекта, выполнив команду *Save All* меню *Delphi*.

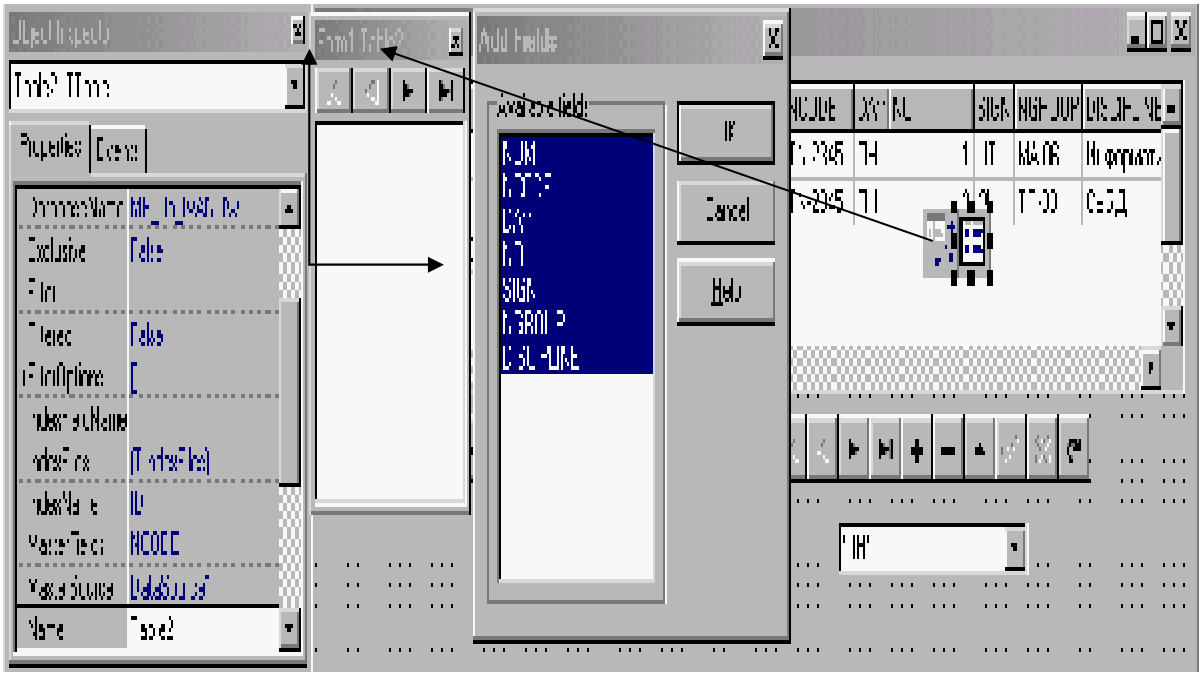

Рис. 5.4. Добавление полей таблицы

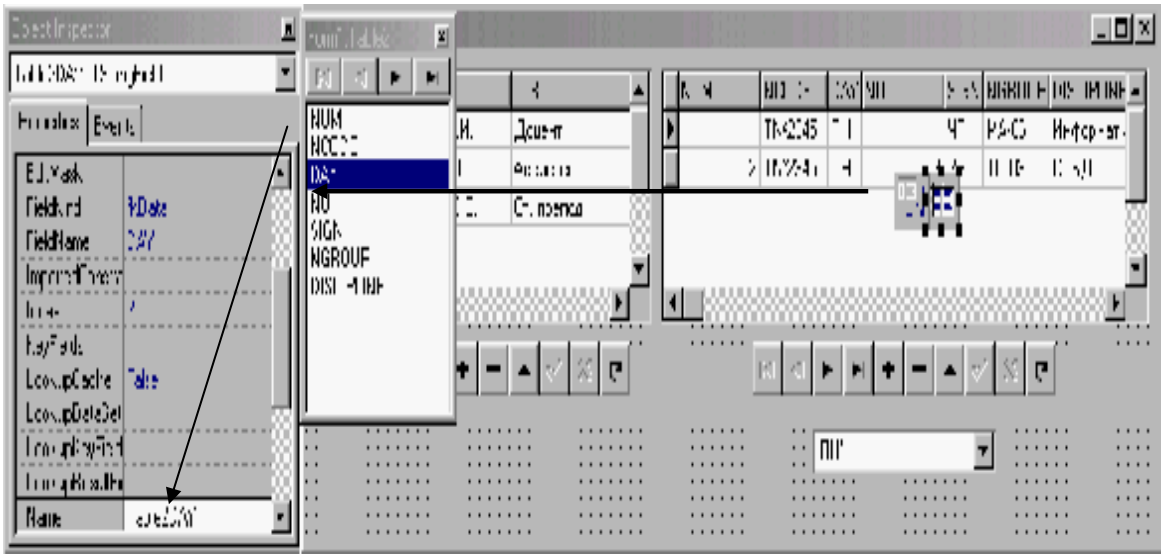

Рис. 5.5. *Присоединенные поля таблицы*

Собственно, реализация фильтра осуществляется исполнением строк программы в процедуре обработки события *OnChange* (обработчике события обмена данными) элемента управления (компонента) *ComboBox1*. Заголовок процедуры обработки события в *Delphi* создается автоматически при выполнении двойного щелчка по выбранному полю на странице *Events* инспектора объектов. В нашем случае необходимо выбрать первый (главный) обработчик события *OnChange* компонента *ComboBox1*. При выполнении указанного действия в редакторе программного кода *Delphi* будет создан заголовок процедуры обработки события обмена данными:

```
procedure TForm1.ComboBox1Change(Sender: TObject);
begin


end;
```

Внутри операторных скобок процедуры, между словами *begin … end*, необходимо записать следующие строки программы (рис. 5.6.).

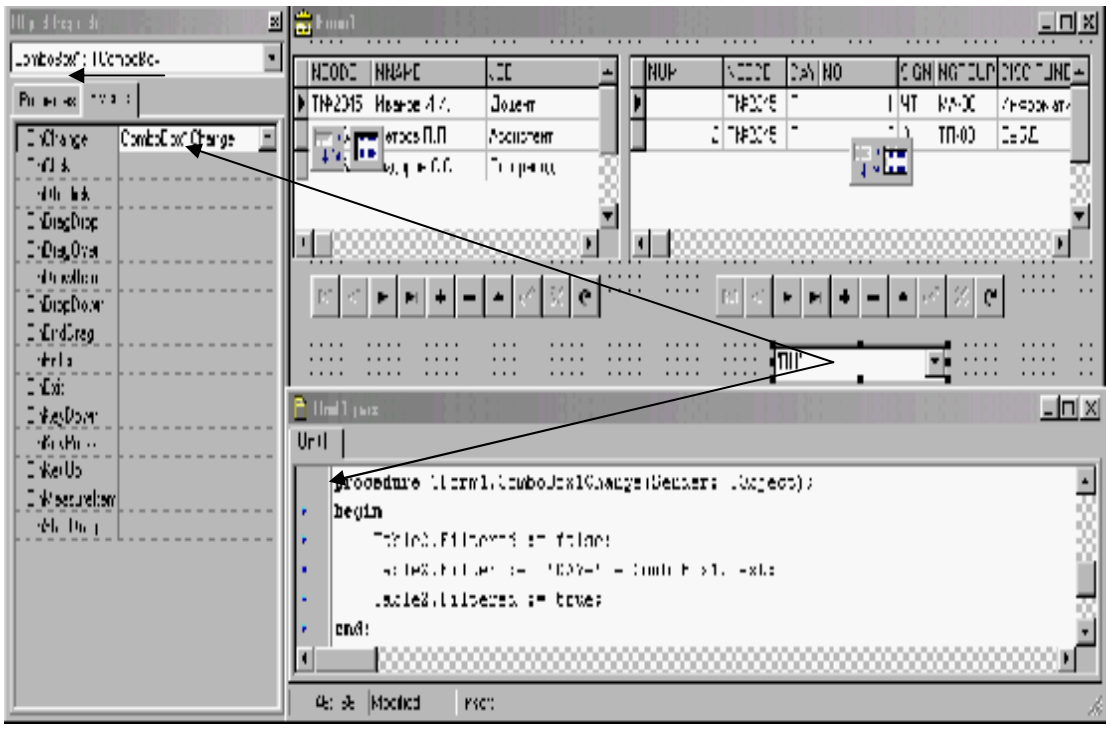

Рис. 5.6. Программа, реализующая фильтр выборки дня занятий

Ввод строк программы можно выполнить следующим образом:

- выбрать в форме компонент *Table2*;
- выбрать в инспекторе объектов свойство *Name* и выполнить команду копировать [Ctrl + C];
- установить в окне редактора кода курсор в нужное место ввода программы и выполнить команду вставить [Ctrl + V];
- после занесения имени компонента в окно программного кода поставить точку за именем (данная операция выведет окно выбора свойств и методов компонента);
- в списке окна достаточно указать необходимое свойство и нажать на клавишу *Enter* или *Space* клавиатуры;
- далее ввести символы присвоения значения (:=) и написать значения свойства;
- после ввода всей строки программы необходимо поставить признак конца программной строки (;);
- последовательно записать все строки программы.

После ввода строк программы необходимо сохранить все файлы проекта (команда *Save All*) и выполнить компиляцию приложения (команда *Run*).

Программа, приведенная на рис. 5.6. выполняется следующим образом:

- при выборе в поле со списком ***ComboBox1*** значения дня недели, первым выполняется метод, деактивирующий фильтр – ***Table2.Filtered := false;***
- затем свойству фильтр присваивается значение для поля день (*DAY*), состоящее из имени поля и значения выбранного из списка компонента ***ComboBox1 – Table2.Filter := 'DAY=' + ComboBox1.Text;***
- последняя строка программы устанавливает фильтр в активное состояние (метод *Filtered*) – ***Table2.Filtered := true.***

После выполнения всех вышеописанных действий запустите приложение и введите все записи в таблицу «Расписание занятий» относительно каждого преподавателя и дня недели.

В реальных *СУБД* часто применяется поиск необходимых записей по вводу первых символов отыскиваемого слова. Как известно поиск осуществляется по индексированным полям (*вторичный индекс*). Для реализации возможности быстрого нахождения фамилии преподавателя необходимо проиндексировать поле *NNAME* таблицы *List*. Данная операция может быть выполнена при помощи утилиты *DataBase Desktop*. Для этого достаточно открыть таблицу *List*, нажать на кнопку *Restructure* и установить вторичный индекс, например *SR* для поля NNAME. После выполнения данной операции повторно индексированную таблицу необходимо сохранить.

В разрабатываемой *СУБД* поиск фамилии будем выполнять посредством ввода символов в поле ввода (компонент *TEdit*). Дополним разрабатываемый проект компонентом *Edit1* (рис. 5.7.).

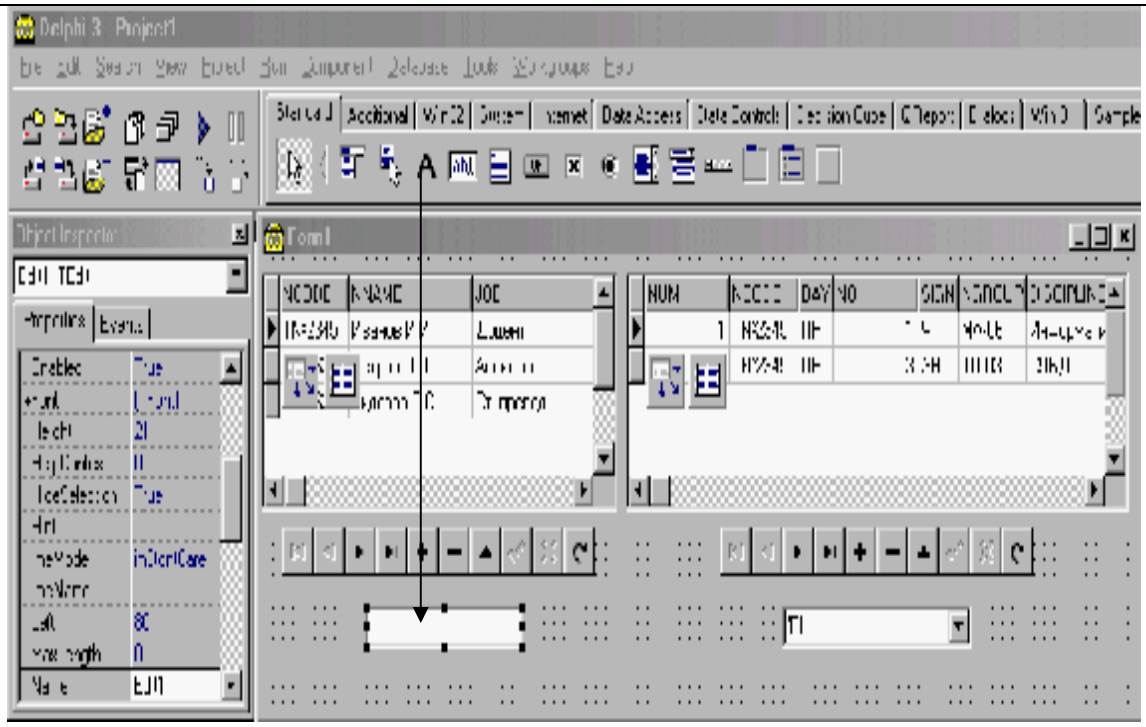

Рис.5.7. Установка в форму компонента Edit

Далее необходимо в форме выбрать компонент *Table1* и для свойства *IndexName* установить значение *SR* (рис. 5.8.).

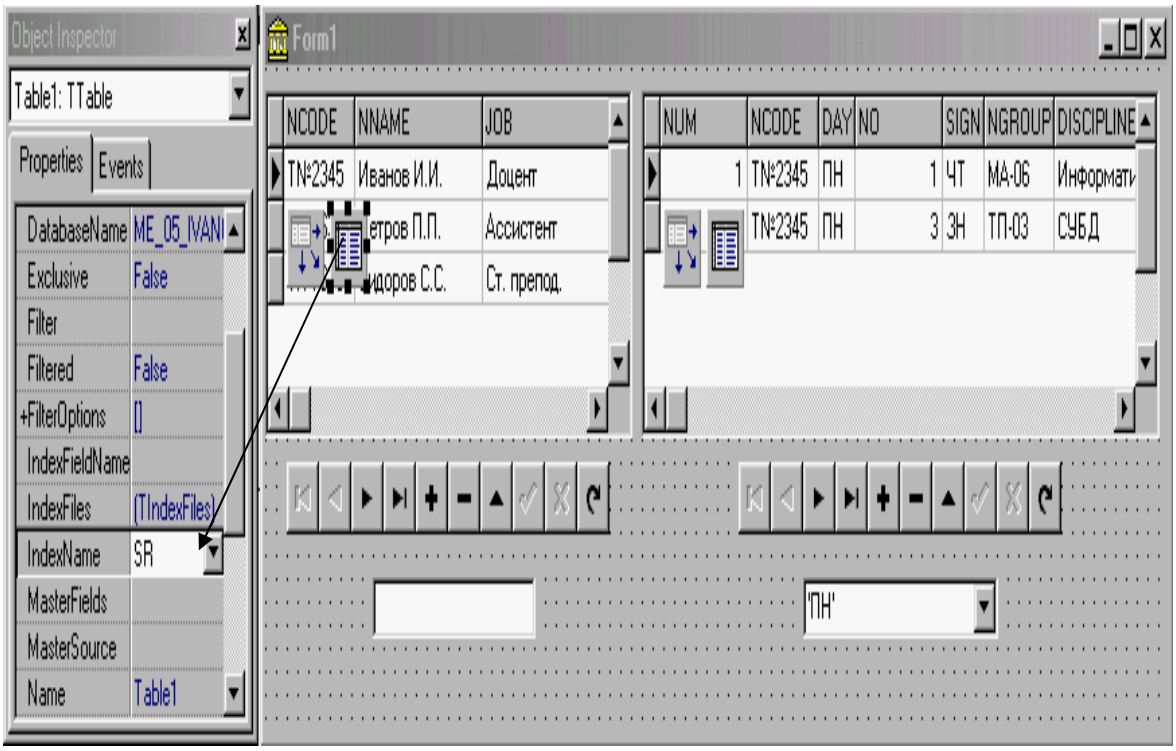


Рис. 5.8. Установка значения индекса

Программная реализация процедуры поиска выполняется в обработчике события *OnChange* компонента *Edit1* (рис. 5.9.).

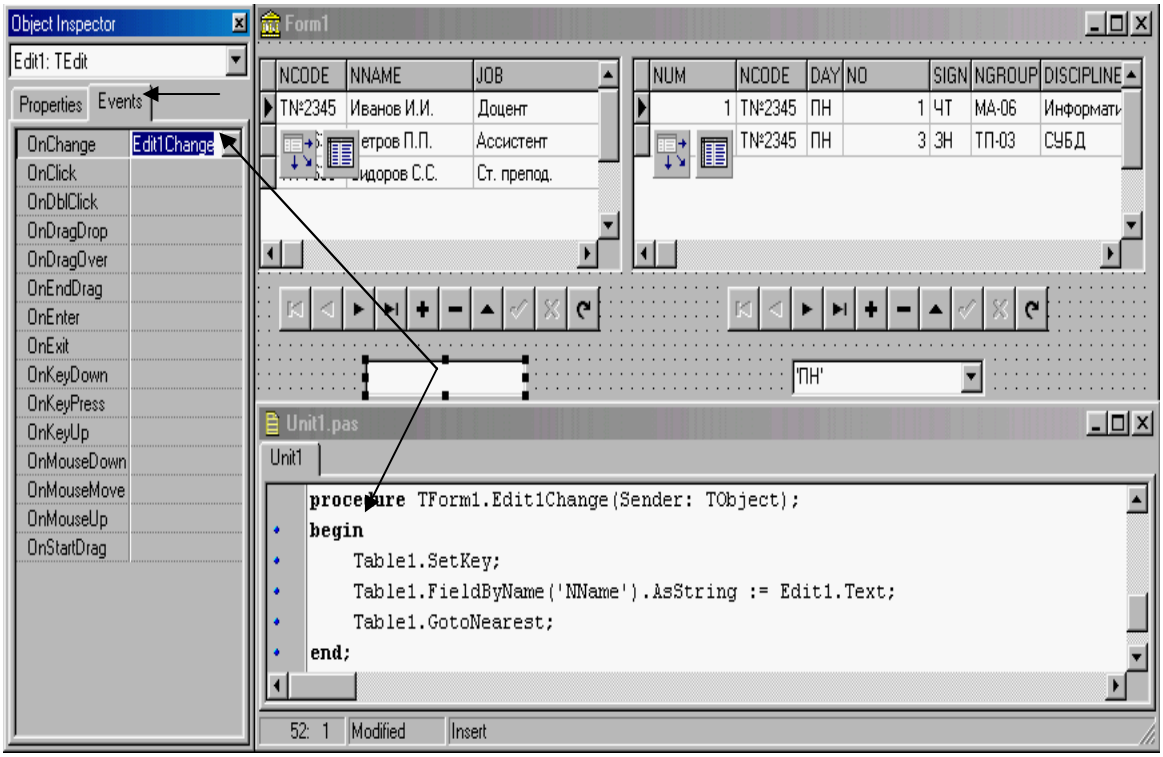


Рис. 5.9. Программная реализация процедуры поиска

После записи строк программы необходимо сохранить все файлы проекта и скомпилировать приложение.

## Тема 6. Администрирование СУБД

Разработанное приложение еще не является завершенным, так как требует доработки его дизайна. К тому же функциональное назначение приложения – это составление расписания занятий, т.е. его административный вариант. С точки зрения пользователя (преподавателя), разработанное приложение является избыточным, т.к. содержит ненужную (служебную) информацию и функции ввода и модификации информации.

Рассмотрим, каким образом, можно модифицировать приложение в пользовательский вариант, так чтобы интерфейс приложения позволял пользователю только просматривать собственное расписание занятий.

- Шаг 1. Переместите все компоненты в нижнюю часть формы таким образом, чтобы была возможность установить в форму панель. Компонент *Panel* находится на станице *Standard* палитры компонентов (рис. 6.1.).

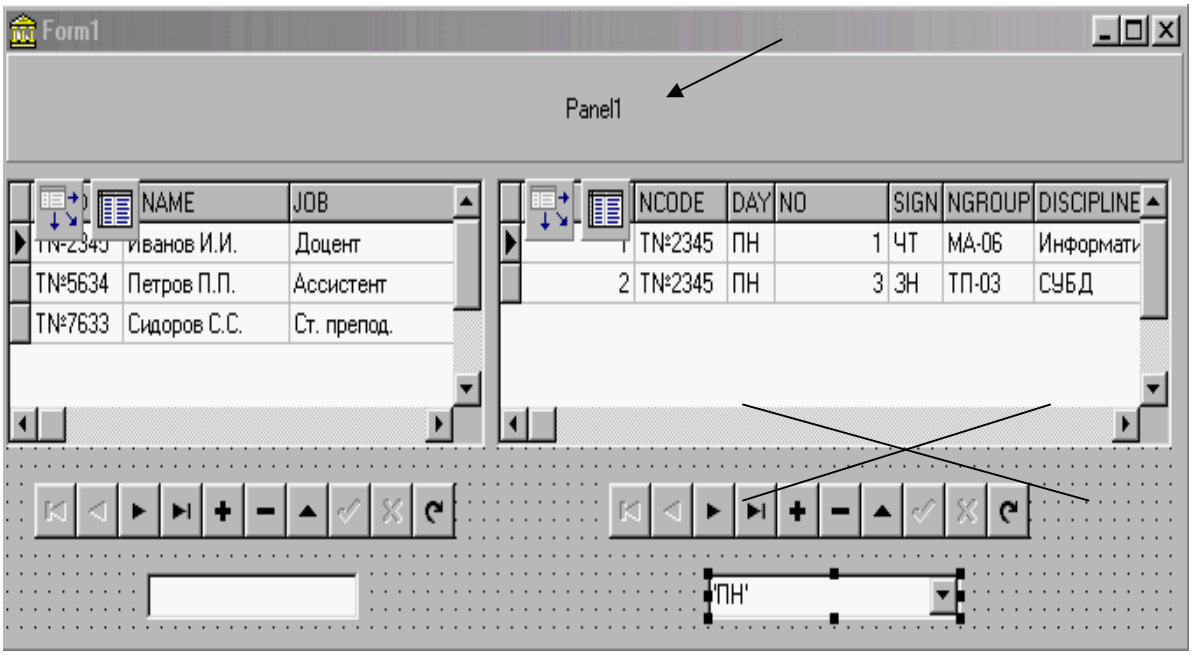


Рис. 6.1. Установка в форму компонента Panel

- Шаг 2. Для компонента *Panel1* установите свойство *Align* в состояние *alTop* и свойство *BorderStyle* в состояние *bsSingle*. Также удалите слово *Panel1* в свойстве *Caption*.
- Шаг 3. Удалите из формы компонент *DBNavigator2*. (*Для удаления компонента достаточно выбрать его в форме и нажать на клавишу* Delete *клавиатуры*).
- Шаг 4. Переместите в панель компоненты: *DBNavigator1*, *Edit1* и *ComboBox1* (рис. 6.2.). (*Для перемещения компонента из формы в панель необходимо выбрать перемещаемый компонент в форме, выполнить команду меню* Edit/Cut, *затем, выбрать компонент* Panel1 *и выполнить команду* Edit/Paste).
- Шаг 5. Удалите заголовок *Edit1* в свойстве *Text* компонента *Edit1*.

- Шаг 6. Так, как пользователь не должен иметь возможности модифицировать информацию необходимо в навигаторе сделать доступными только кнопки перемещения по записям. Для этого достаточно для навигатора (компонент *DBNavigator1*) установить значения списка свойств *VisibleButtons* в соответствии с рис. 6.3.

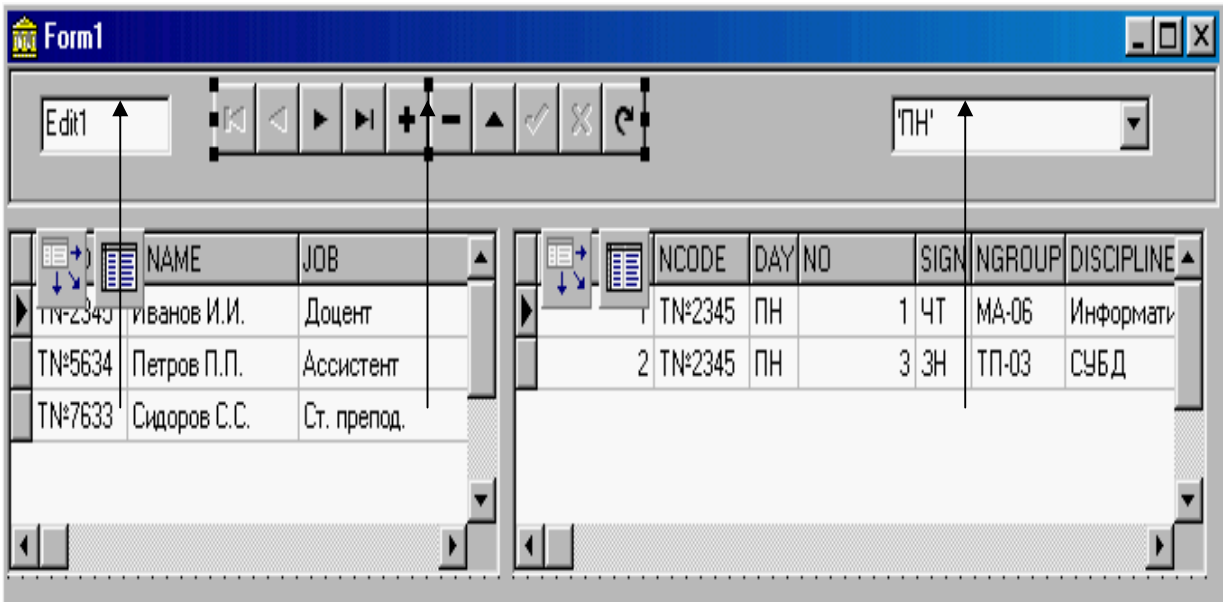


Рис. 6.2. Перемещение компонентов в панель

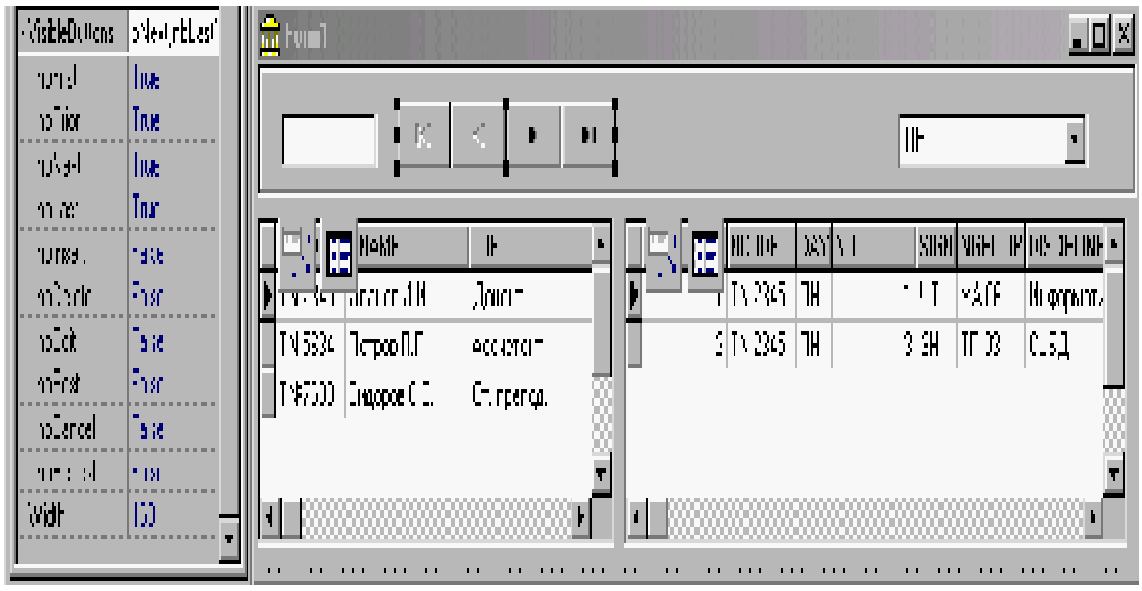

Рис. 6.3. Установка доступных кнопок навигатора

- Шаг 7. В связи с тем, что в сетках (компоненты *DbGrid*) отображается служебная информация, которая не представляет интереса для пользователя ее необходимо скрыть. Для сокрытия информации в сетках необходимо выполнить двойной щелчок мышью по выбранному компоненту *DbGrid*, что приведет к выводу окна редактирования столбцов (рис. 6.4.). Для редактирования внешнего вида полей их необходимо включить в редактор, выполнив команду *Add All Fields*. Для того чтобы поле не отображалось в сетке, достаточно выбрать его в окне редактора и нажать на кнопку *Delete*. В нашем случае, для списка преподавателей необходимо сокрыть поле *NCODE* (Табельный номер).

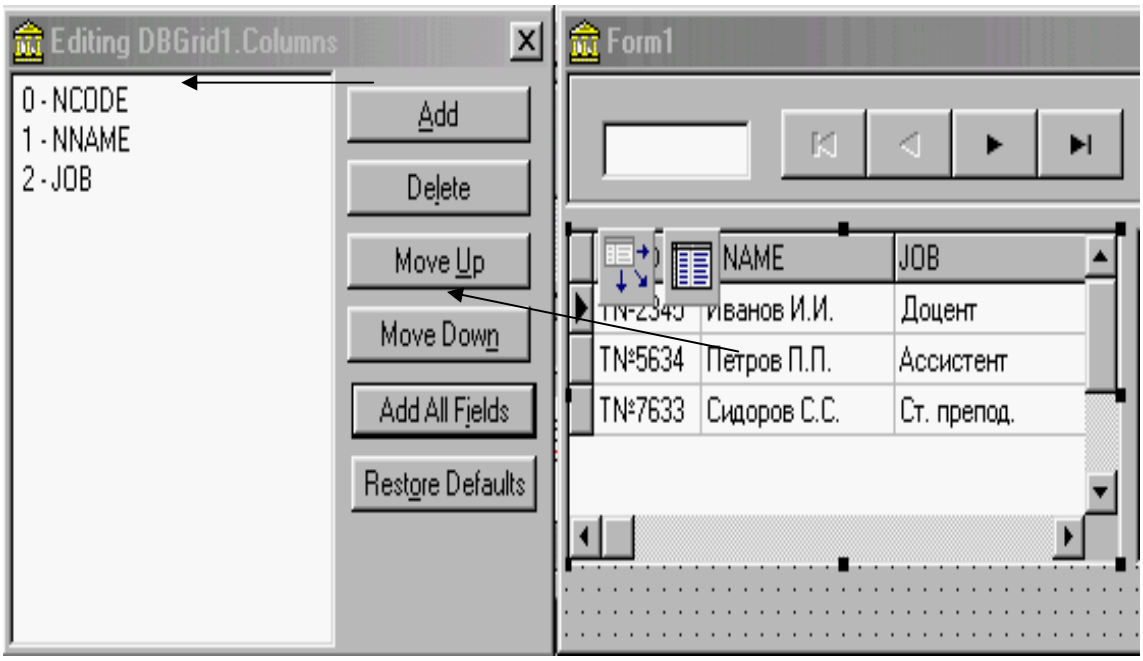


Рис. 6.4. Редактирование отображаемых полей

- Шаг 8. С целью отображения названия полей на русском (украинском) языке заголовки полей необходимо русифицировать. Для этого достаточно выбрать необходимое поле в списке редактора полей и заменить значение *Caption* в списке свойства *Title* (рис. 6.5.). Цвет заголовка и положение надписей определяются свойствами *Color* и *Aligment*.

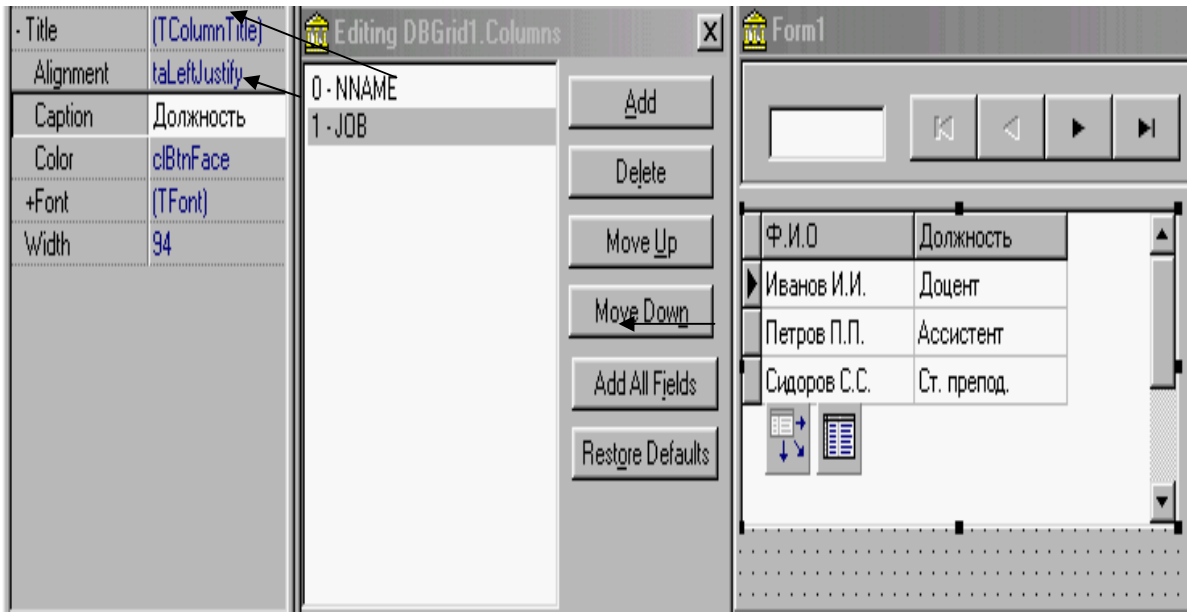


Рис. 6.5. Русификация названий полей

- Шаг 9. Для сетки, отображающей расписание занятий, самостоятельно скройте поля *NUM, DAY и NCODE* и выполните русификацию видимых заголовков полей.

После выполнения описанных действий необходимо сохранить проект и выполнить компиляцию приложения.

Заключительным этапом разработки пользовательского приложения «Расписание занятий» является его эргономический дизайн рис. 6.6.

Внешний вид пользовательского приложения «Расписание занятий»

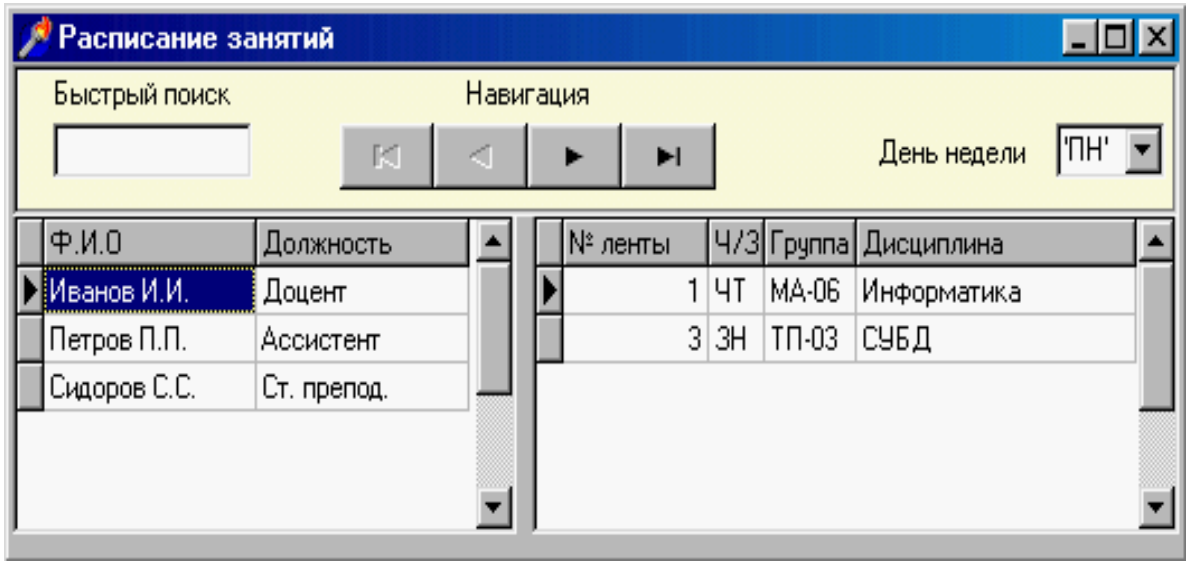


Рис. 6.6.

# ТЕМА 7. ОБЪЕКТ QUERY.
# СОЗДАНИЕ СВЯЗАННЫХ ТАБЛИЦ С ИСПОЛЬЗОВАНИЕМ SQL ЗАПРОСА

Рассмотрим пример создания связанных таблиц. Для демонстрации примера необходимо установить в форму два набора компонентов. Первый набор включает Table1, DataSource1 и DBGrid1. Данный набор должен быть связан с таблицей CUSTOMER базы данных DBDEMOS. Второй набор содержит объекты Query1, DataSource2 и DBGrid2 связанные между собой (рис. 7.1.) Свойство SQL объекта Query1 должно содержать текст SQL запроса вида: **select * from Orders where CustNo = :CustNo**

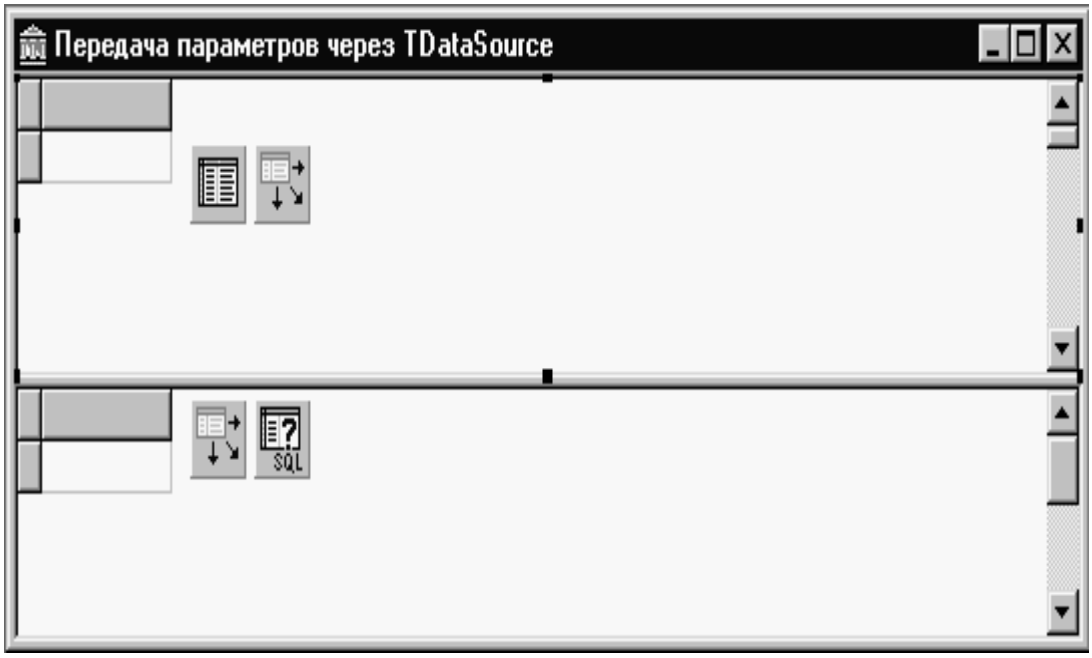


Рис. 7.1. Размещение компонентов в форме

В свойстве DatabaseName объекта Query1 необходимо указать псевдоним базы данных DBDEMOS, а в свойстве DataSource Query1 необходимо указать источник DataSource1.

После установки свойств Active = True объектов Table1 и Query1 приложение будет иметь вид приведенный на рис. 7.2. В выполняемом приложении таблицы CUSTOMER и ORDERS будут связаны между собой.

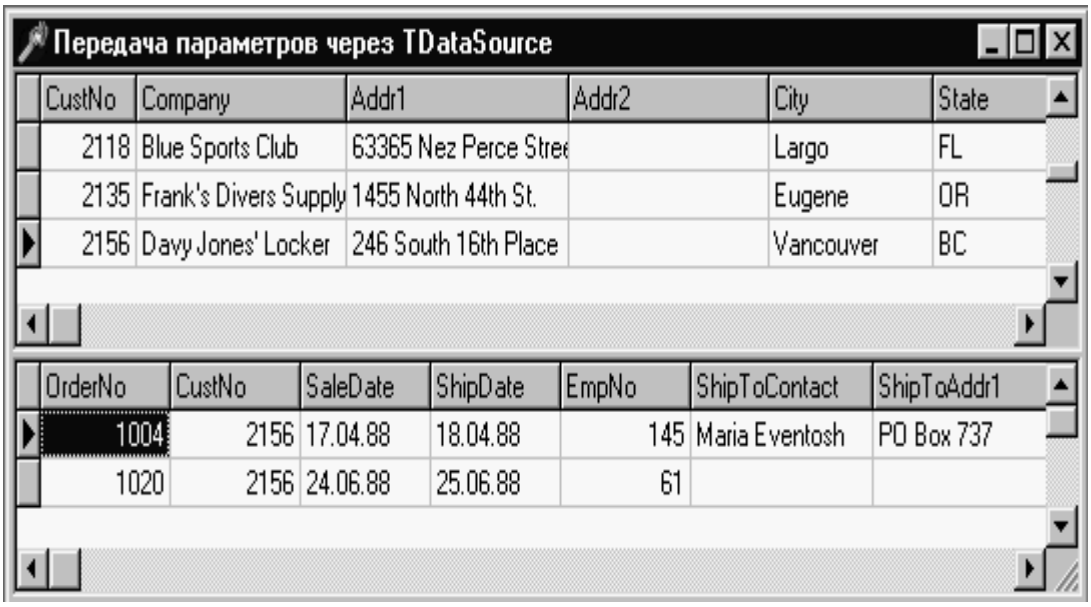


Рис. 7.2. Приложение, в период выполнения

## Литература


1. Боас Р., Фервай М., Гюнтер Х. Delphi 4. Полное Руководство. – К.: BHV, 1998. – 448 с.
2. Developer’s Guide for Delphi 3, Borland Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249
3. Developer’s Guide for Delphi 5, Borland Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249
4. Object Pascal Language Guide, Borland Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249
5. Анталогия Delphi, http://www.Torry.ru

Содержание

Учебное издание

Швачич Геннадий Григорьевич
Овсянников Александр Васильевич
Кузьменко Вячеслав Витальевич
Соболенко Александр Викторович
Байрак Виталий Васильевич

# Системы управления базами данных
# Часть II. Основы разработки систем управления базами данных в интегрированной среде Delphi

Учебное пособие

Тем. план 2008, поз. 183.

Підписано до друку 28.02.08. Формат 60x84 1/16. Папір друк. Друк плоский. Облік.-вид. арк. 2,82. Умов. друк. арк. 2,79. Тираж 100 пр. Замовлення №

Національна металургійна академія України
49600, м. Дніпропетровськ-5, пр. Гагаріна, 4

Редакційно-видавничий відділ НМетАУ