МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНАЯ МЕТАЛЛУРГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ

Кафедра прикладной математики и вычислительной техники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ ПО ДИСЦИПЛИНЕ

«Системы управления базами данных»

для студентов направления 6.020105
«Документоведение и информационная деятельность»

Днепропетровск 2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНАЯ МЕТАЛЛУРГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ

Кафедра прикладной математики и вычислительной техники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ ПО ДИСЦИПЛИНЕ

«Системы управления базами данных»

для студентов направления 6.020105
«Документоведение и информационная деятельность»

Днепропетровск 2012

Лабораторная работа №1

Тема: Создание таблиц базы данных с помощью СУБД Access

Access - это программа, с помощью которой можно быстро и довольно просто создавать базы данных, наполнять их и работать с ними. входит в состав пакета **Microsoft Office**, и запустить его можно через **Пуск – Программы- Создать документ Microsoft Office – Новая база данных**. Файлы, созданные с помощью **Access** имеют расширение *.mdb.

СОЗДАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХИ УСТАНОВЛЕНИЕ СВЯЗЕЙ МЕЖДУ ТАБЛИЦАМИ

Создайте на диске D (Doki) в Вашей индивидуальной папке папку СУБД_ЛАБ, а в ней папку Лабораторная_1.

1. Создайте базу данных ДЕКАНАТ, выполнив следующие действия:загрузите Access, в появившемся окне выберите пункт НОВАЯ БАЗА ДАННЫХ, затем нажмите кнопку ОК; в окне ФАЙЛ НОВОЙ БАЗЫ ДАННЫХ задайте имя – Lab_1 (пункт ИМЯ ФАЙЛА) и выберите папку Лабораторная_1 (пункт ПАПКА), где ваша база будет находиться (по умолчанию Access предлагает вам имя базы db1., а тип файла – БАЗЫ ДАННЫХ ACCESS), нажмите кнопку СОЗДАТЬ.

2. Создайте структуру таблицы ПРЕПОДАВАТЕЛИ. Для этого: в окне базы данных выберите вкладку ТАБЛИЦЫ, а затем нажмите кнопку СОЗДАТЬ; в окне НОВАЯ ТАБЛИЦА выберите пункт КОНСТРУКТОР и нажмите кнопку ОК; в результате проделанных операций открывается окно таблицы в режиме конструктора, в котором следует определить поля таблицы; определите поля таблицы в соответствии с табл 1.

Таблица 1. ПРЕПОДАВАТЕЛИ

Имя поля	**Тип данных**	**Размер поля**
Код преподавателя	Числовой	Целое
Фамилия	Текстовый	15
Имя	Текстовый	13
Отчество	Текстовый	15
Дата рождения	Дата/время(маска)	
Должность	поле со списком (Мастер подстановок)	
Стаж	Числовой	Целое (условие на значение > 0, вывод соответствующего сообщения об ошибке)
Кафедра	Текстовый	25

Телефон	Текстовый	11
Зарплата	Денежный	

При определении поля ДАТА РОЖДЕНИЯ используем маску для удобного ввода даты (т. е. в датах точки будут вводится автоматически). Для этого в Свойства полей на вкладке Общие установите курсор на поле маска, справа появится кнопка с тремя точками – нажмите на нее. В появившемся окне создания масок выбирайте КРАТКИЙ ФОРМАТ ДАТЫ. В поле ДОЛЖНОСТЬ используем мастер подстановок для того, чтобы не вводить, а выбирать из списка нужную должность с использованием ввода должности, которой нет в списке. В режиме СОЗДАНИЯ ПОДСТАНОВОК выбираем ФИКСИРОВАННЫЙ НАБОР ЗНАЧЕНИЙ, далее создаем 1-й столбец с должностями: профессор;

доцент;

старший преподаватель;

ассистент.

Закончив создание списка в режиме конструктора на вкладке ПОДСТАНОВКА, посмотрите появившиеся изменения после работы мастера. Проверьте строку ОГРАНИЧИТЬСЯ СПИСКОМ, в котором должно стоять слово НЕТ. В поле СТАЖ в общих свойствах поля установите УСЛОВИЕ НА ЗНАЧЕНИЕ > 0, СООБЩЕНИЕ ОБ ОШИБКЕ введите – стаж должен быть больше 0. В поле ТЕЛЕФОН наберите маску для ввода 999-99-99, которая позволит не набирать тире в номере телефона при вводе в поле (подробнее о маске см. help). В качестве ключевого задайте поле КОД ПРЕПОДАВАТЕЛЯ

Закройте таблицу ПРЕПОДАВАТЕЛИ в режиме конструктора.

Откройте таблицу ПРЕПОДАВАТЕЛИ и заполните первую строку. 5. При вводе стажа преподавателя введите отрицательный стаж – 17. Остальная часть этой таблицы также будет заполняться из приложения, созданного в среде Delphi.

3. Создайте в приложении EXCEL таблицу СТУДЕНТ с перечисленными ниже полями (табл. 2) и сохраните файл в своей папке. Далее созданная таблица будет импортирована из таблицы EXCEL Меню ФАЙЛ/ВНЕШНИЕ ДАННЫЕ/ИМПОРТ.

Отредактируйте созданную таблицу в режиме конструктора.

Таблица 2 СТУДЕНТ

Имя поля	**Тип данных**	**Размер поля**
Код студента	Числовой	Целое
Номер группы	Числовой	Целое
Имя	Текстовый	15

Фамилия	Текстовый	15
Отчество	Текстовый	12
Адрес	Текстовый	40
Телефон	Текстовый	(маска 9)
Медалист	Текстовый4	Поле со списком (Да/Нет) по умолчанию Нет

В качестве ключевого задайте поле КОД СТУДЕНТА. Для этого щелкните по полю КОД СТУДЕНТА и выполните команду ПРАВКА КЛЮЧЕВОЕ ПОЛЕ.

Для удобства ввода телефона задайте маску (см. поле ТЕЛЕФОН таблицы ПРЕПОДАВАТЕЛИ). В поле МЕДАЛИСТ создайте ПОЛЕ СО СПИСКОМ без ввода новых значений, а также задайте ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ “Нет” (кавычки обязательны).

4. Создайте структуру таблицы ДИСЦИПЛИНЫ аналогично пункту 2 .

Таблица 3 ДИСЦИПЛИНЫ

Имя поля	**Тип данных**	**Размер поля**
Код дисциплины	Числовой	Целое
Название дисциплины	Текстовый	30
Номер семестра	Числовой	Целое
Код преподавателя	Числовой	Целое
Экзамен	Текстовый	(мастер подстановок, поле со списком Экз./Зач.)

В качестве ключевого задайте поле КОД ДИСЦИПЛИНЫ. Поле КОД ПРЕПОДАВАТЕЛЯ будет заполнятся при помощи мастера подстановок из таблицы ПРЕПОДАВАТЕЛИ. Из доступных полей таблицы ПРЕПОДАВАТЕЛИ выберите: КОД ПРЕПОДАВАТЕЛЯ, ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, скройте ключевое поле. После работы мастера при заполнении поля КОД ПРЕПОДАВАТЕЛЯ таблицы будут отображаться ФИО преподавателя для выбора, но в таблице ДИСЦИПЛИНЫ поле КОД ПРЕПОДАВАТЕЛЯ будет оставаться числовым целым. Поле ЭКЗАМЕН заполняйте при помощи поля со списком двух значений Экз. или Зач. Закройте таблицу ДИСЦИПЛИНЫ. Откройте таблицу ДИСЦИПЛИНЫ и введите первую строку. При вводе кода преподавателя выберите любуюфамилию преподавателя. Оставшаяся часть таблицы будет заполняться из приложения, созданного в среде Delphi.

5. Таблица ОЦЕНКИ должна быть создана Вами в новой базе данных СУБД Access (имя файла OZENKI.mdb, имя таблицы OZEN)и будет импортирована из этой базы в базу Lab_1 Меню ФАЙЛ/ВНЕШНИЕ ДАННЫЕ/ИМПОРТ.

6. Отредактируйте структуру таблицы ОЦЕНКИ аналогично в соответствии с табл. 4.

Таблица 4 ОЦЕНКИ

Имя поля	Тип данных	Размер поля
Код студента	Числовой	Целое
Код дисциплины	Числовой	Целое
Номер семестра	Числовой	Целое
Оценки	Числовой	Целое

Ключ будет составной: КОД СТУДЕНТА, КОД ДИСЦИПЛИНЫ, НОМЕР СЕМЕСТРА (в режиме конструктора выделите три поля и задайте ключ).

Разработайте схему данных, т.е. создайте связи между таблицами. Для этого: Выполните команду СЕРВИС СХЕМА ДАННЫХ. На экране появится окно СХЕМА ДАННЫХ. Выполните команду СВЯЗИ ДОБАВИТЬ ТАБЛИЦУ. В появившемся окне будет выделено название одной таблицы. Нажмите кнопку ДОБАВИТЬ. Переведите выделение на имя следующей таблицы и нажмите кнопку ДОБАВИТЬ. Аналогично добавьте оставшиеся две таблицы. Закройте окно нажав кнопку ЗАКРЫТЬ. Создайте связь между таблицами ДИСЦИПЛИНЫ и ОЦЕНКИ. Для этого подведите курсор мыши к полю КОД ДИСЦИПЛИНЫ в таблице ДИСЦИПЛИНЫ, нажмите левую клавишу мыши и, не отпуская ее, перетащите курсор на поле КОД ДИСЦИПЛИНЫ в таблице ОЦЕНКИ, а затем отпустите левую клавишу мыши. На экране откроется окно СВЯЗИ. Щелкните по ячейке ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ ДАННЫХ – в ней должна появиться галочка. Щелкните по ячейкам КАСКАДНОЕ ОБНОВЛЕНИЕ СВЯЗАННЫХ ПОЛЕЙ и КАСКАДНОЕ УДАЛЕНИЕ СВЯЗАННЫХ ЗАПИСЕЙ. Информация. Задание каскадного обновления связанных полей и каскадного удаления связанных записей позволит редактировать записи только в таблице ДИСЦИПЛИНЫ, а в таблице ОЦЕНКИ эти действия будут со связанными записями выполняться автоматически. Например, если вы удалите из таблицы ДИСЦИПЛИНЫ один предмет, то в таблице ОЦЕНКИ удалятся все строки, связанные с этим предметом. Нажмите кнопку СОЗДАТЬ. Связь будет создана. Аналогично создайте связи между полем КОД ПРЕПОДАВАТЕЛЯ в таблице ПРЕПОДАВАТЕЛИ и полем КОД ПРЕПОДАВАТЕЛЯ в таблице ДИСЦИПЛИНЫ, а также между полем КОД СТУДЕНТА в таблице СТУДЕНТЫ и полем КОД СТУДЕНТА в таблице ОЦЕНКИ. Закройте окно схемы данных, ответив ДА на вопрос о сохранении макета. Результаты работы представьте преподавателю.

Лабораторная работа №2

Тема: Утилита Database Desktop. Создание структуры таблиц на платформе Paradox.

Ути́лита (англ. *utility* или *tool*) — компьютерная программа, расширяющая стандартные возможности оборудования и операционных систем, выполняющая узкий круг специфических задач.

Утилиты предоставляют доступ к возможностям (параметрам, настройкам, установкам), недоступным без их применения, либо делают процесс изменения некоторых параметров проще (автоматизируют его).

Утилиты могут входить в состав операционных систем, идти в комплекте со специализированным оборудованием или распространяться отдельно.

При помощи утилиты можно создавать и редактировать базы данных в формате dBASE и Paradox, а также выполнять SQL запросы. Данная утилита позволяет редактировать все поля данных, за исключением BLOB полей.

BLOB (англ. Binary Large OBject — двоичный большой объект) — массив двоичных данных. В СУБД **BLOB** — специальный тип данных, предназначенный, в первую очередь, для хранения изображений, аудио и видео, а также компилированного программного кода. В самом поле содержится лишь ссылка на отдельный файл базы данных, в котором хранится двоичный массив. В Database Desktop данный тип полей указан как Binary и Graphic.

Утилита запускается из программной группы среды Delphi. При первом запуске программы следует указать рабочие каталоги. Определение рабочих каталогов выполняется посредством команд **File/Working Directory** и **File/Private Directory**.

СОЗДАНИЕ ТАБЛИЦ

1. Создание новой таблицы выполняется при помощи команды **File/New/Table**. После выполнения этой команды появится диалоговое окно **Create Table**, в поле списка которого выбирается тип (формат) таблицы (рис.1).

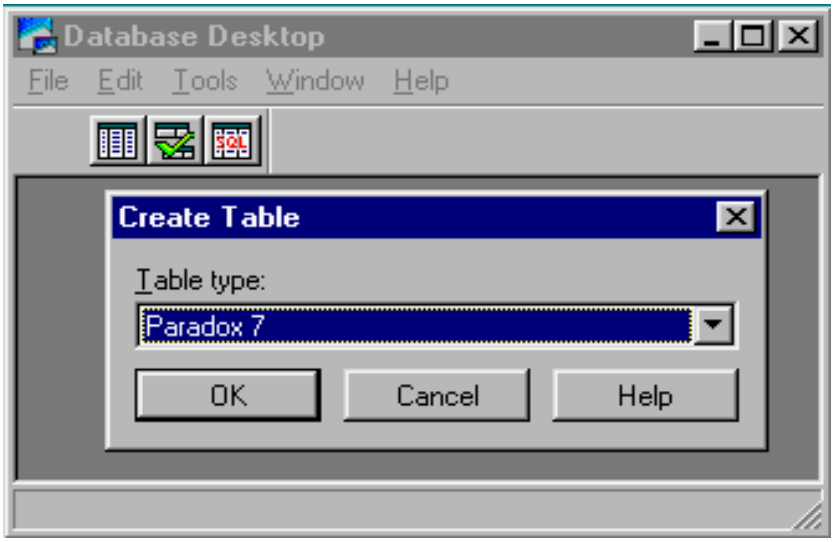


Рис.1. Создание новой таблицы

2. После выбора формата создаваемой таблицы открывается новое диалоговое окно (рис.2). Поля новой таблицы определяются в области **Field roster**, а именно в столбцах вводятся: **Field Name →** имя поля , **Type →** тип поля, **Size →**размер поля и **Key →**первичный ключ **(**вводится значок ***)**. Область **Table Properties** используется для выбора значений индексов и драйвера языка таблицы.

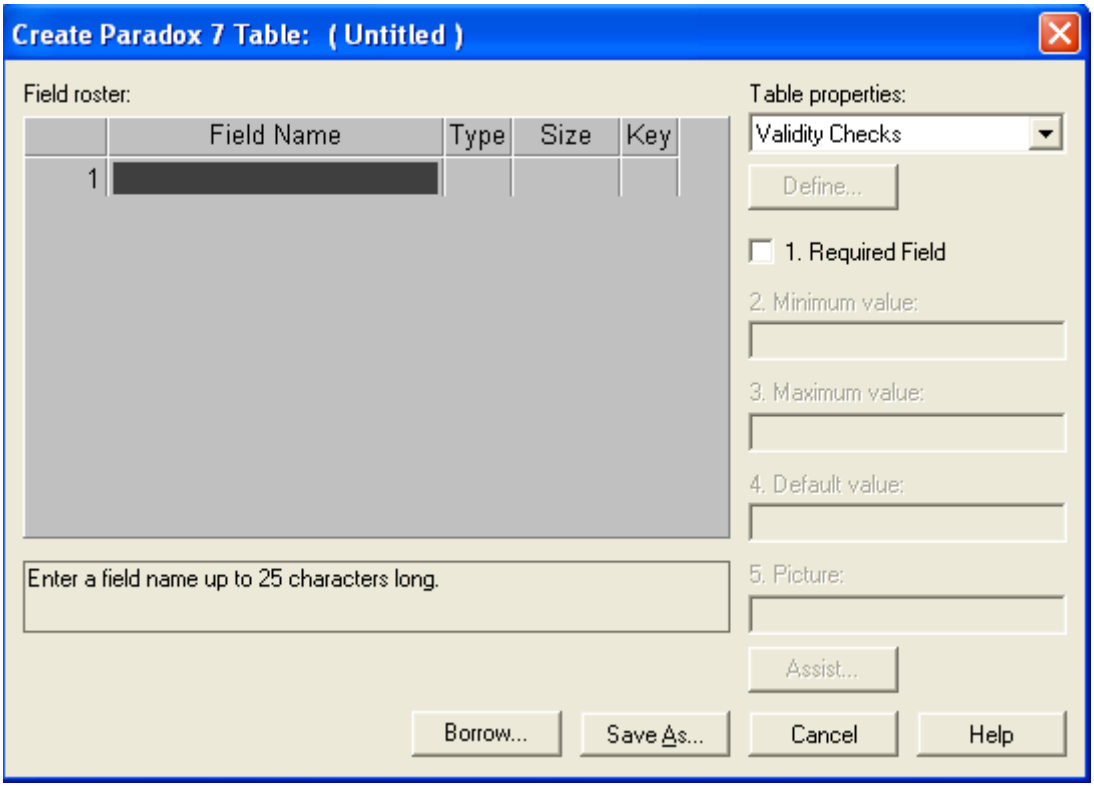


Рис.2.

3. На рис.3 представлена структура таблицы Biolife.

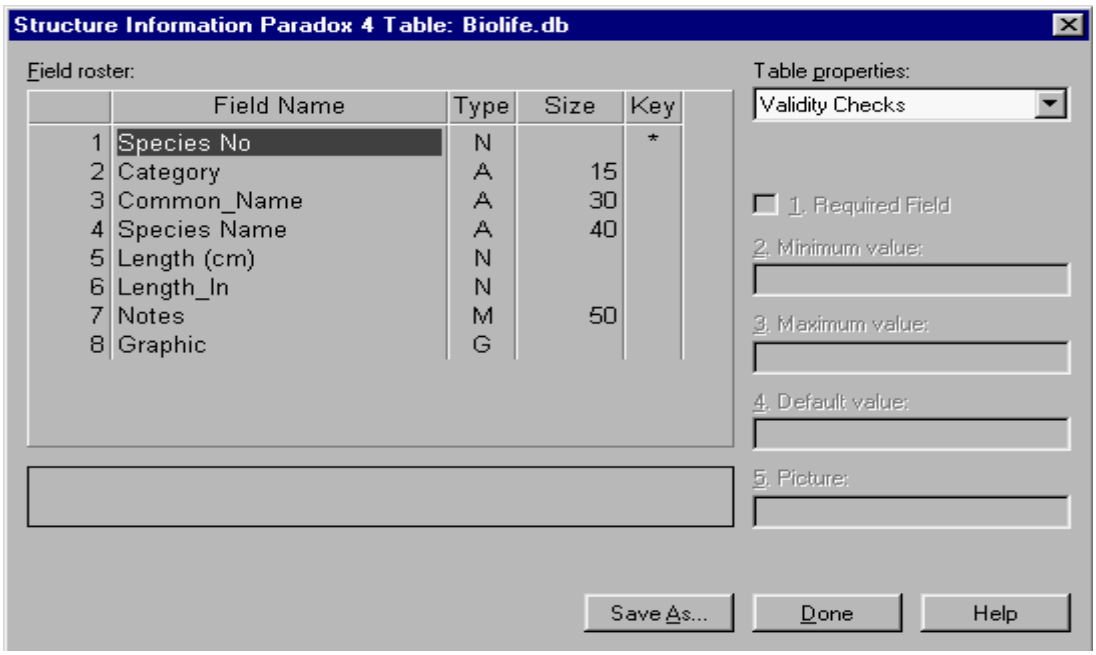


Рис.3

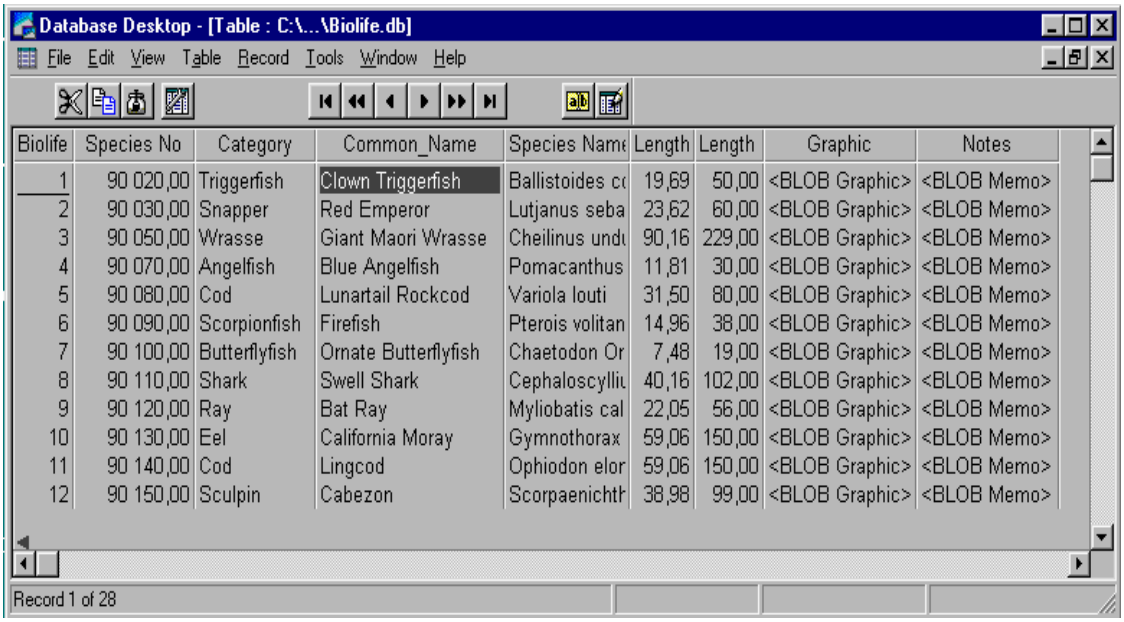
На рис.3 представлена структура таблицы **Biolife**, созданная в формате **Paradox 7**. Данная таблица содержит цифровые поля (**N**), строковые поля (**A**) и двоичные поля (**M**) и (**G**) предназначенные для сохранения текста и изображения. Первое поле таблицы определено как ключевое (первичный ключ – Primary → *).

Внимание!!! Для таблиц формата **Paradox 7** первое поле всегда определяется ключевым и должно быть отмечено знаком *!

Для работы с новой таблицей, ее следует сохранить, используя команду **Save As**. Вывод на экран таблицы выполняется командой **File/Open/Table**.

На рис.4 представлена предварительно открытая заполненная таблица **Biolife**. Редактирование данных полей таблицы, за исключением **BLOB** полей, выполняется командами меню **Edit**.

Заполненные поля данных таблицы Biolife в утилите Database Desktop представлены на рис.4, а фрагмент этих же данных через СУБД представлен на рис.5.



Biolife	Species No	Category	Common_Name	Species Name	Length	Length	Graphic	Notes
1	90 020,00	Triggerfish	Clown Triggerfish	Ballistoides c	19,69	50,00	<BLOB Graphic>	<BLOB Memo>
2	90 030,00	Snapper	Red Emperor	Lutjanus seba	23,62	60,00	<BLOB Graphic>	<BLOB Memo>
3	90 050,00	Wrasse	Giant Maori Wrasse	Cheilinus und	90,16	229,00	<BLOB Graphic>	<BLOB Memo>
4	90 070,00	Angelfish	Blue Angelfish	Pomacanthus	11,81	30,00	<BLOB Graphic>	<BLOB Memo>
5	90 080,00	Cod	Lunartail Rockcod	Variola louti	31,50	80,00	<BLOB Graphic>	<BLOB Memo>
6	90 090,00	Scorpionfish	Firefish	Pterois volitan	14,96	38,00	<BLOB Graphic>	<BLOB Memo>
7	90 100,00	Butterflyfish	Ornate Butterflyfish	Chaetodon Or	7,48	19,00	<BLOB Graphic>	<BLOB Memo>
8	90 110,00	Shark	Swell Shark	Cephaloscyllu	40,16	102,00	<BLOB Graphic>	<BLOB Memo>
9	90 120,00	Ray	Bat Ray	Myliobatis cal	22,05	56,00	<BLOB Graphic>	<BLOB Memo>
10	90 130,00	Eel	California Moray	Gymnothorax	59,06	150,00	<BLOB Graphic>	<BLOB Memo>
11	90 140,00	Cod	Lingcod	Ophiodon elor	59,06	150,00	<BLOB Graphic>	<BLOB Memo>
12	90 150,00	Sculpin	Cabezon	Scorpaenichth	38,98	99,00	<BLOB Graphic>	<BLOB Memo>

Рис.4

Номер подвида	Категория	Название	Подвид	Размер (см)	Максимальный размер	Изображение	Заметки
90 030,00	Snapper	Red Emperor	Lutjanus seba	23,62	60,00		Есть около 105 разновидностей рыбы Грубияна, широко распространяемая морская рыба семьи Lutjanidae(или Lutianidae);

Рис.5

Создание таблиц в форматах **Paradox** и **dBASE** выполняются по установленным правилам.

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;

Предупреждение!!! *Некоторые СУБД очень «не любят» пробелы в названии полей, поэтому вместо пробелов набирайте нижний дефис: вместо* Species No *набирайте* Species_No!!!

- 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.

Имя поля в таблице формата **dBase** представляет собой строку, написание которой подчиняется правилам, отличным от **Paradox**:

- Имя должно быть не длиннее 10 символов;
- пробелы в имени недопустимы.

Имена полей в формате **dBase** подчиняются более строгим правилам, чем имена полей в формате **Paradox**. При совместном использовании платформ **dBase** и **Paradox** рекомендуется присваивать имена полей в формате **Paradox** по правилам формата **dBase**.

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. Смысл размера поля такой же, как и в **Formatted Memo**. **Database Desktop** позволяет создавать поля типа **Graphic**, однако заполнять их можно только в приложении.
- **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. Смысл размера поля такой же, как и в **Formatted Memo**. **Database Desktop** позволяет создавать поля типа **OLE**, однако наполнять их можно только в приложении.
- **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** (тип файла: 3D-модели, изображения).
- **Bytes** – данное поле предназначено для хранения двоичной информации, представляет собой строку цифр длиной 1-255 байт.

Для ввода типа поля достаточно набрать только подчеркнутые символы.

В формате **dBase** поля таблиц могут иметь следующий тип:

- **Character (alpha)** – поле представляет собой строку длиной 1-254 байт, содержащую любые печатаемые символы.
- **Float (numeric)** – числовое поле размером 1-20 байт в формате с плавающей точкой, значение которого может быть положительным и отрицательным. Поле может содержать большие величины, однако следует иметь в виду ошибки округления, возникающие при работе с полем данного типа. Число цифр после десятичной точки (параметр **Dec**) должно быть по крайней мере на 2 меньше, чем размер всего поля, поскольку в общий размер включаются сама десятичная точка и знак.
- **Number (BCD)** – числовое поле размером 1-20 байт, содержащее данные в формате **BCD (Binary Coded Decimal**). Скорость вычислений значений данного поля немного меньше, чем скорость вычислений в других числовых форматах, при этом, точность вычислений значительно выше. Число цифр после десятичной точки (параметр **Dec**) также должно быть, по крайней мере, на 2 меньше чем размер всего поля, поскольку в общий размер включаются сама десятичная точка и знак.
- **Date** – поле даты длиной 8 байт. По умолчанию, используется формат короткой даты (**ShortDateFormat**).
- **Logical** – поле длиной 1 байт, которое может содержать только значения «**true**» или «**false**». Допускаются применение строчных и прописных букв. Также допускается применение букв «**Y**» и «**N**» (сокращение от **Yes** и **No**).
- **Memo** – поле для хранения символов, суммарная длина которых более 255 байт. Поле может иметь любую длину. Данное поле хранится в отдельном файле.

Database Desktop не обладает возможностью модифицировать данные в поле типа **Memo**.

- **OLE** – поле, содержащее **OLE** объекты (**Object Linking and Embedding**) – образы, звук, видео, документы – которые для своей обработки вызывают создавшее их приложение. Поле может иметь любую длину. Это поле также сохраняется в отдельном файле. **Database Desktop** обладает возможностью только создавать поля типа **OLE**, однако наполнять их можно только в приложении.
- **Binary** – поле, содержащее любую двоичную информацию. Может иметь любую длину. Данное поле сохраняется в отдельном файле с расширением *.**DBT (**тип файла: файлы баз данных).

Для ввода типа поля достаточно набрать только подчеркнутые символы.

Для таблиц в формате **Paradox** обязательно надо определить поле (или поля), составляющее первичный ключ, причем это поле должно быть расположено в начале таблицы.

ВЫПОЛНИТЬ

1. Создать в D:\«Ваша индивидуальная папка»\ СУБД_ЛАБ папку Лабораторная_2.

1. Законспектировать типы полей и правила заполнения имен полей для таблиц формата **Paradox.**

2. Вызвать утилиту Database Desktop: Пуск → Программы (Все программы) →Borland Delphi 7 → Database Desktop и создать структуру таблицы **Biolife** (рис.3.). При заполнении поля **Type** рекомендуется вызывать контекстное меню на данном поле и в открывшемся окне (рис.6.) выбирать нужное значение.

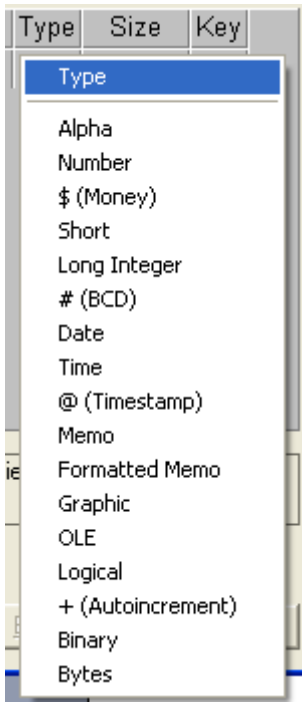


Рис.6. Типы полей

3. Каждая таблица, созданная через утилиту Database Desktop, хранится в виде файла. После заполнения всех полей сохраните таблицу нажатием на кнопку SaveAs (*таблицу сохранять только в Вашу личную папку Лабораторная_1*). При сохранении назовите Вашу таблицу **Biolife**, одновременно это будет именем файла, в котором сохранена таблица.

4. Выполнив команду **File/Open/Table**, выведите на экран созданную таблицу. Нажатие на кнопку **Restructure** (рис.7) позволяет переходить из режима, отображающего структуру таблицы, в режим ее заполнения. Чтобы заполнить

таблицу информацией через утилиту Database Desktop надо перейти в режим редактирования, нажав кнопку **Edit Data** .

Рис.7

5. Заполните данными (рис.4) две строки в Вашей таблице.

6. Самостоятельно создайте структуру таблицы и заполните ее данными через утилиту Database Desktop, в соответствии с рисунком 8. Имена полей и название таблицы должны набираться *латинскими* символами (режим языка EN).

№ п/п	№ студенческого	Ф.И.О.	Дата рождения	Время рождения	Адрес	Телефон	Стипендия, грн	Автобиография
1	10856	Дубовская А.П.	12.12.1995	12:00:00	Днепропетровск, пр. Гагарина 6, ком.25	0935484365	700,00р.	Я, Дубовская Ангелина Павловна, родилась...
2	10857	Коваленков Р.Р.	03.01.1996	23:05:30	Днепропетровск, пр. Правды 16, кв.225	0672355689	850,00р.	Я, Коваленков Роман Романович, родился...

Рис.8.

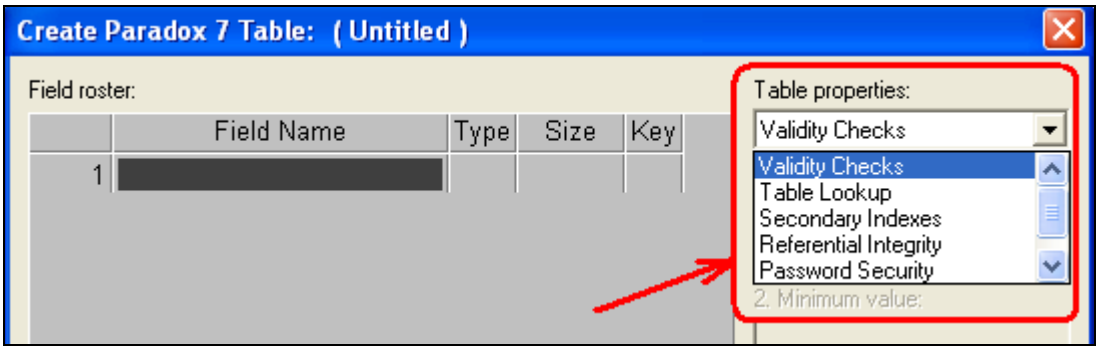
7. При защите лабораторной работы знать что такое:

- утилита;
- назначение утилиты Database Desktop;
- Paradox и dBASE;
- типы данных;
- правила набора имен полей;
- что обозначается символом * ;
- что вводится в столбцы Field Name, Type, Size и Key при создании структуры таблицы;
- что такое первичный ключ;
- в виде чего хранится созданная через утилиту Database Desktop таблица;
- как заполнить данными таблицу через утилиту Database Desktop;
- что такое *Primary* в терминологии *баз данных*;
- обязательно ли первое поле в таблице на платформе Paradox должно быть отмечено как первичный ключ?

Лабораторная работа №3

Тема: Утилита Database Desktop. Работа со свойствами создаваемой таблицы (TableProperties).Алиас.

После создания структуры таблицы, с ней можно связать некоторые свойства, перечень которых зависит от формата таблицы. Так, для таблиц формата **Paradox** можно задать:



- **Validity Checks** – это свойство проверяет минимальное и максимальное значение данных, а также значение по умолчанию. Кроме того, позволяет задать маску ввода.
- **Table Lookup** – данное свойство позволяет вводить значение в таблицу, используя уже существующее значение в другой таблице.
- **Secondary Indexes** – вторичные индексы. Создание вторичных индексов позволяет осуществлять доступ к данным в порядке, отличном от порядка, заданного первичным ключом.
- **Referential Integrity** – ссылочная целостность. Данное свойство позволяет задать связи между таблицами и поддерживать эти связи на уровне ядра базы данных. Как правило, **Referential Integrit** задается после создания всех таблиц в базе данных.
- **Password Security** – данное свойство позволяет закрыть таблицу паролем.
- **Table Language** – данное свойство предназначено для выбора языкового драйвера таблицы.

В таблицах формата **dBase** не существует первичных ключей. Однако, это обстоятельство можно преодолеть путем определения уникальных (**Unique**) и поддерживаемых (**Maintained**) индексов (**Indexes**). Кроме того, для таблиц формата **dBase** можно определить и язык таблицы (**Table Language**) т.е. установить языковый драйвер, управляющий сортировкой и

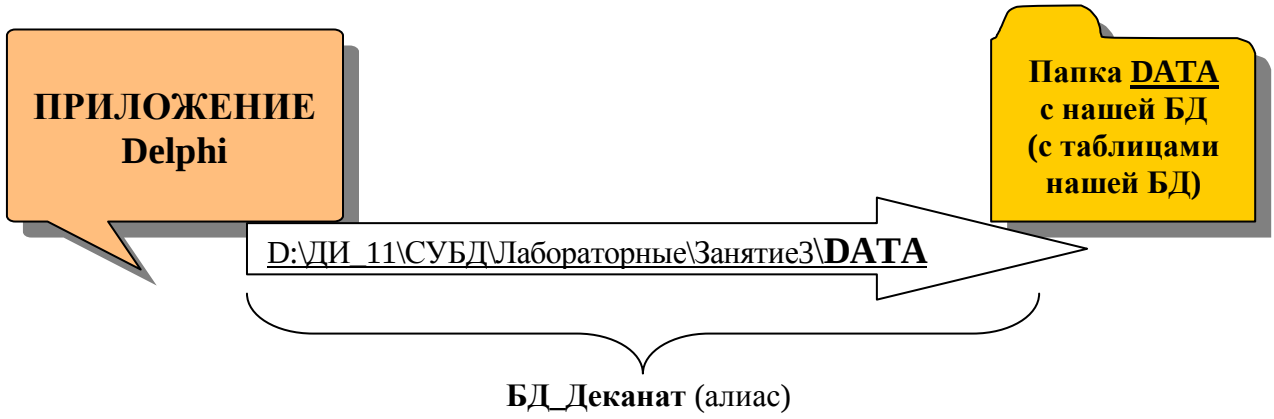
отображением символьных данных.

Определения дополнительных свойств таблиц всех форматов доступны через кнопку «**Define**». Дополнительные свойства можно устанавливать не только при создании таблиц, но и для существующих таблиц. С этой целью в **Database Desktop** включены команды **Table|Restructure Table** (для открытой в данный момент таблицы) и **Utilities|Restructure** (для выбора таблицы). В том случае, если Вы попытаетесь изменить структуру или добавить новые свойства в таблицу, которая в данный момент уже используется другим приложением, **Database Desktop** выдаст сообщение об отказе, так как данная операция требует монопольного доступа к таблице. Тем не менее, все произведенные в структуре изменения сразу же начинают функционировать, если Вы определите ссылочную целостность для пары таблиц.

Database Desktop обладает возможностью создавать таблицу любого формата путем копирования структуры уже существующей таблицы. Для этого достаточно воспользоваться кнопкой «**Borrow**», которая расположена в левом нижнем углу окна формы. Появляющееся диалоговое окно позволит Вам выбрать существующую таблицу и включить/выключить дополнительные опции, совпадающие с уже перечисленными свойствами таблиц. Это наиболее легкий способ создания таблиц.

АЛИАС

Алиас (в буквальном переводе означает "прозвище, кличка"**)** – это имя (идентификатор) посредством которого указывается путь к таблицам базы данных. Применение **алиаса** позволяет значительно ускорить доступ к БД.



На представленной картинке *имя* **БД_Деканат** – это алиас для пути D:\ДИ_11\СУБД\Лабораторные\Занятие3**DATA** Т.е., чтобы добраться до

нашей папки **DATA**, надо прописывать указанный путь или ввести «волшебное слово» **БД_Деканат**.

ВНИМАНИЕ!!! Алиас должен регистрироваться для папки, в которой хранятся таблицы Вашей БД!!!

ВЫПОЛНИТЬ

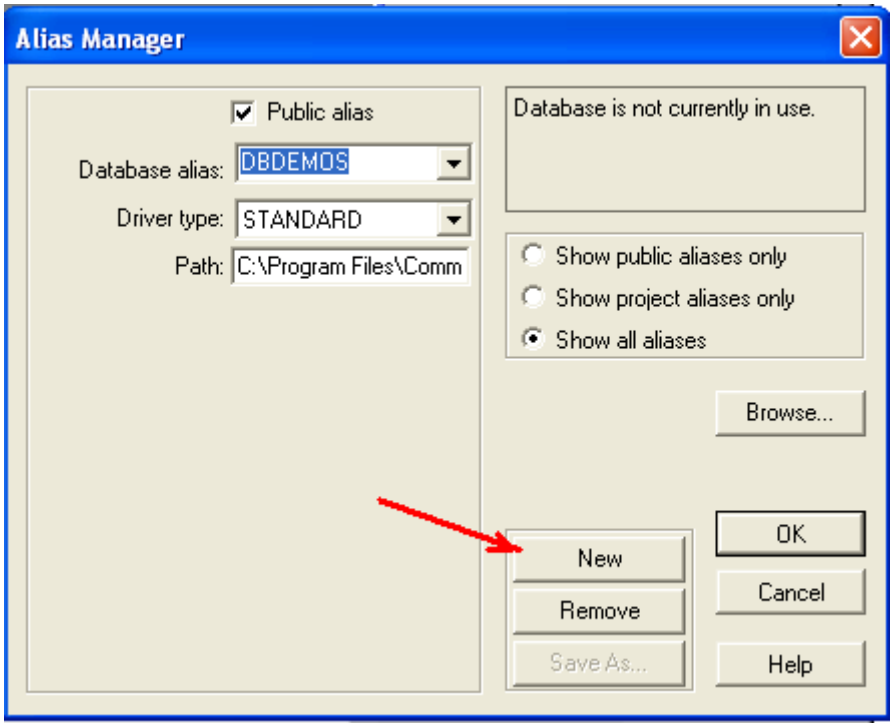
1. Создайте на диске D (Doki) в Вашей индивидуальной папке папку СУБД_ЛАБ, в ней папку Лабораторная_3, а в ней папку **DATA**.

D:\ДИ_11\СУБД\Лабораторная_3**DATA**

В папке DATA в дальнейшем будем сохранять таблицы, которые построим на этом занятии (или скопируем в неё те таблицы, которые мы создали во время выполнения Лабораторной№2). Поэтому прежде, чем приступить к созданию (или копированию) таблиц, рекомендуется для папки DATA, в которой будут храниться таблицы, зарегистрировать алиас. Алиас должен быть уникальным (неповторимым) для каждой папки, в которой храниться БД (таблицы БД), например: IVANOV_DI_11 или BAZA1 или 111. Алиас удобно создать, используя утилиту DataBase Desktop, вызов которой возможен из программой группы Delphi главного меню.

2. Вызвать утилиту DataBase Desktop → меню Tools → *Alias Manager*.

3. Далее выполнить щелчок по кнопке *New* ввести в окно *Database Alias* значение нового псевдонима (алиаса), в нашем случае Вашу фамилию IVANOV (рис. 2.).



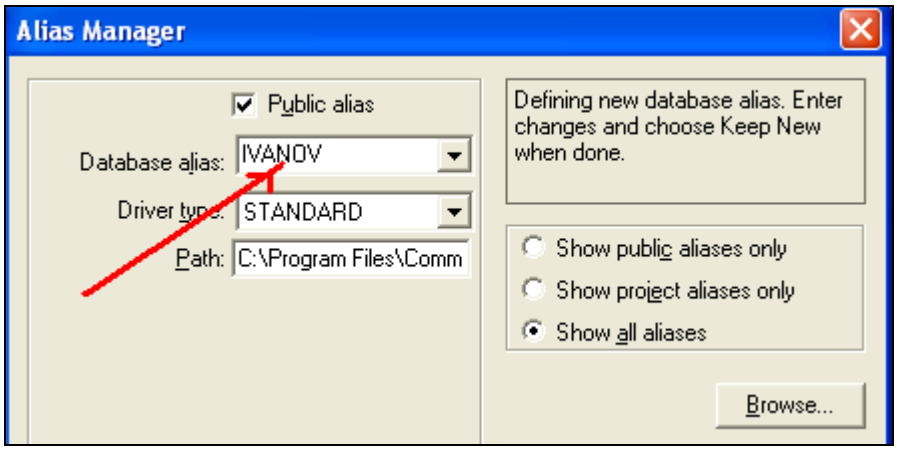


Рис. 2. Ввод значения алиаса

Затем необходимо выполнить щелчок по кнопке *Browse* и в дереве списка папок выбрать папку DATA и нажать на кнопку OK (рис. 3).

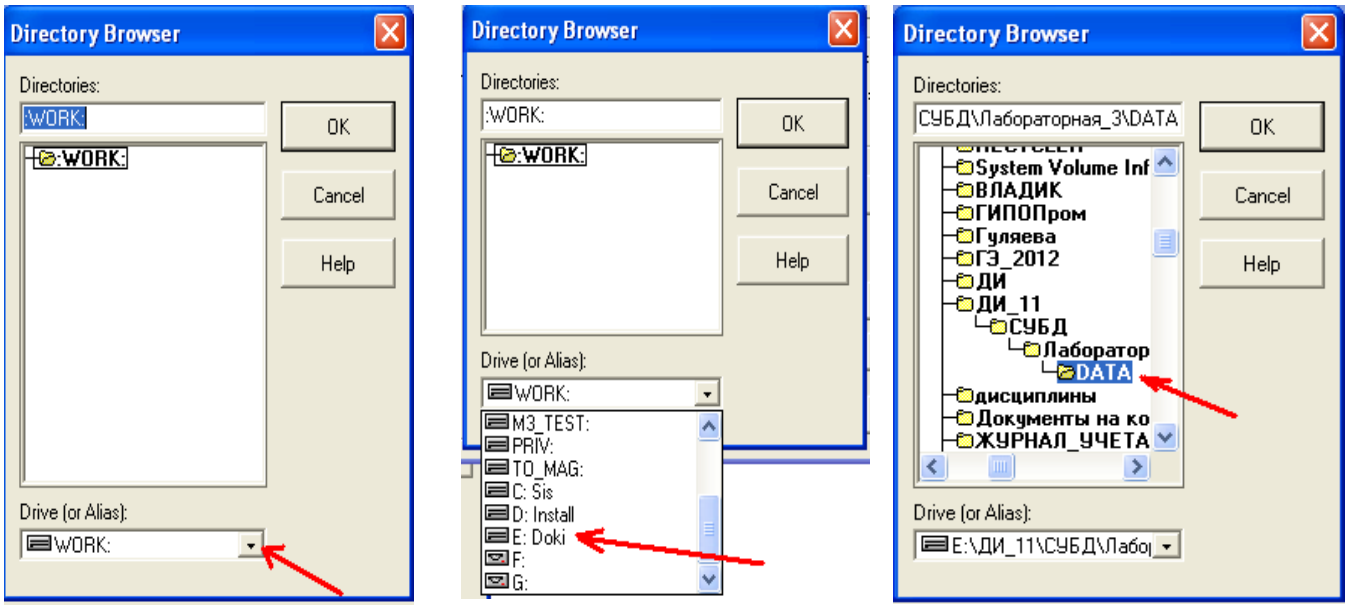


Рис.3.

4. Нажмите кнопку OK в окне **DirectoryBrowser**.

5. Нажмите кнопку OK в окне **Alias Manager.**

6. Запись нового псевдонима в файл конфигурации *IDAPI* будет выполнена после подтверждения записи (Кнопка «Да») – рис. 4.

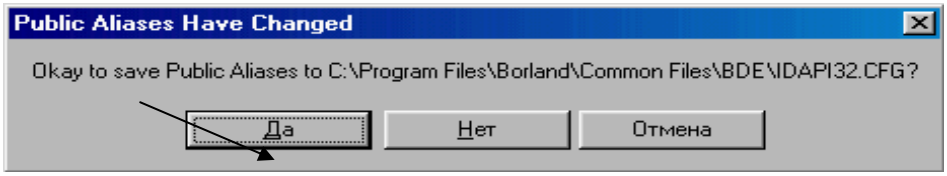


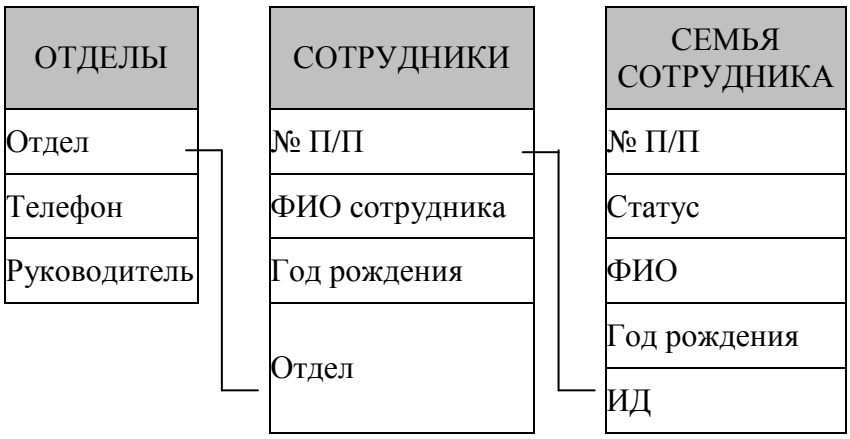
Рис. 4. Подтверждение нового алиаса

7. Создайте через утилиту Database Desktop структуры трех таблиц: ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ и сохраните их в ранее созданную папку DATA.

8. В таблицах ОТДЕЛЫ и СОТРУДНИКИ поля *Отдел* должны иметь *одинаковый тип* и *одинаковый размер*.

9. В таблице СОТРУДНИКИ поле №П/П должно иметь тип Autoincrement.

10. В таблице СЕМЬЯ СОТРУДНИКОВ поле ИД должно иметь тип LongInteger (длинное целое).



11. Первые поля во всех трех таблицах должны быть отмечены как ключевые (в столбце KEYставиться знак *).

12. При защите лабораторной работы знать:

- что такое алиас и для чего он нужен;
- как зарегистрировать алиас;
- для каких папок есть смысл регистрировать алиас;
- какие свойства (и где) можно задать для таблиц формата **Paradox.**

ЛАБОРАТОРНАЯ РАБОТА №4

Тема: ИНДЕКСИРОВАНИЕ ПОЛЕЙ (ВТОРИЧНЫЙ ИНДЕКС)

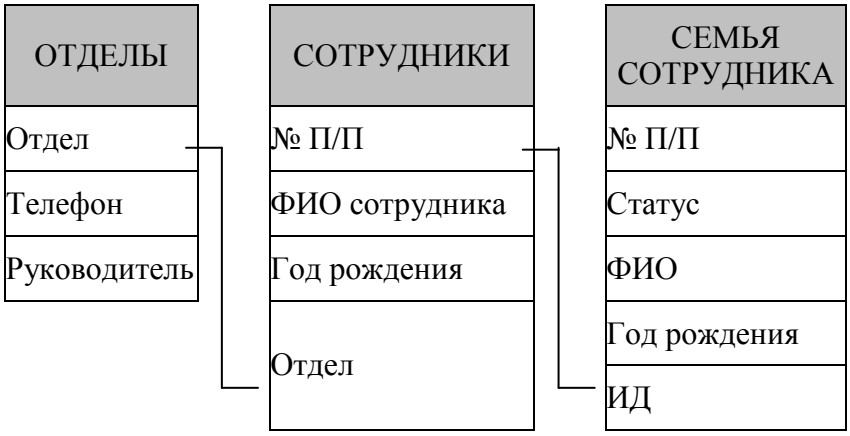
Ключ представляет собой **поле** (или комбинацию полей), данные в которых однозначно определяют (идентифицируют) каждую запись в таблице. Поля, по которым построен ключ, называют *ключевыми* и значения этих полей **не могут повторяться!!!** В таблице может быть только один ключ. *Ключ* также называют *первичным ключом* или первичным индексом (главным Primary).

Кроме *первичного индекса* существует *вторичный индекс* (в дальнейшем просто *индекс*). **Индекс**, как и ключ, строится по полям таблицы, однако он может допускать повторение значений составляющих его полей – в этом и состоит его основное отличие от ключа. Он применяется для ускорения доступа при нахождении данных в отношении (в таблице), а также для выборки и сортировки данных. Таким образом, *индексом* называется *дополнительный объект*, который создается по одному или нескольким столбцам (полям) таблицы для облегчения быстрого доступа к данным.

Из двух логически связанных таблиц (отношений)одну называют главной таблицей (или **master**-отношением), а другую подчиненной таблицей(**detail**-отношением).

Индексирование полей может быть выполнено как в процессе описания их структуры (*т.е. в процессе создания структуры таблицы*), так и после сохранения таблицы как файла.

На предыдущей лабораторной работе Вами была создана база данных из трех таблиц ОТДЕЛЫ, и СЕМЬЯ СОТРУДНИКА. Назовем эту базу данных **«Предприятие».** Эта БД реляционная, т.е таблицы между собой связаны логическими связями. Однако, обратите внимание на тот факт, что в самой папке DATA таблицы хранятся как отдельные файлы и ни как между собой не связаны. При создании в Delphi приложения по управлению БД **«Предприятие»** нам придется самим, с помощью определенных *свойств* определенных *компонентов* устанавливать *механизм связи* между нашими таблицами. Работа этого механизма возможна только в том случае, если поля, по которым настраивается связь *проиндексированы* (т.е., то ли это поле - *первичный ключ* Primary →*, то ли *проиндексированное* поле – вторичный *индекс*).



Структурная схема БД «Предприятие»

ВЫПОЛНИТЬ

Проиндексировать поля: *СОТРУДНИКИ* и *ИД* . Поля: **Отдел** в таблице *ОТДЕЛЫ* и **№ПП** в *двух остальных таблицах* должны быть указаны как ключевые (*первичный ключ* Primary →*).

Для присвоения полям вторичного индекса (**Secondary Index)** необходимо:

- открыть через утилиту Database Desktop таблицу, поле в которой надо проиндексировать и перевести ее в режим, отображающий структуру таблицы.

- выбрать из списка свойств **Table properties**, значение **Secondary Index,** а затем щелкнуть по кнопке **Define** (рис. 1.).

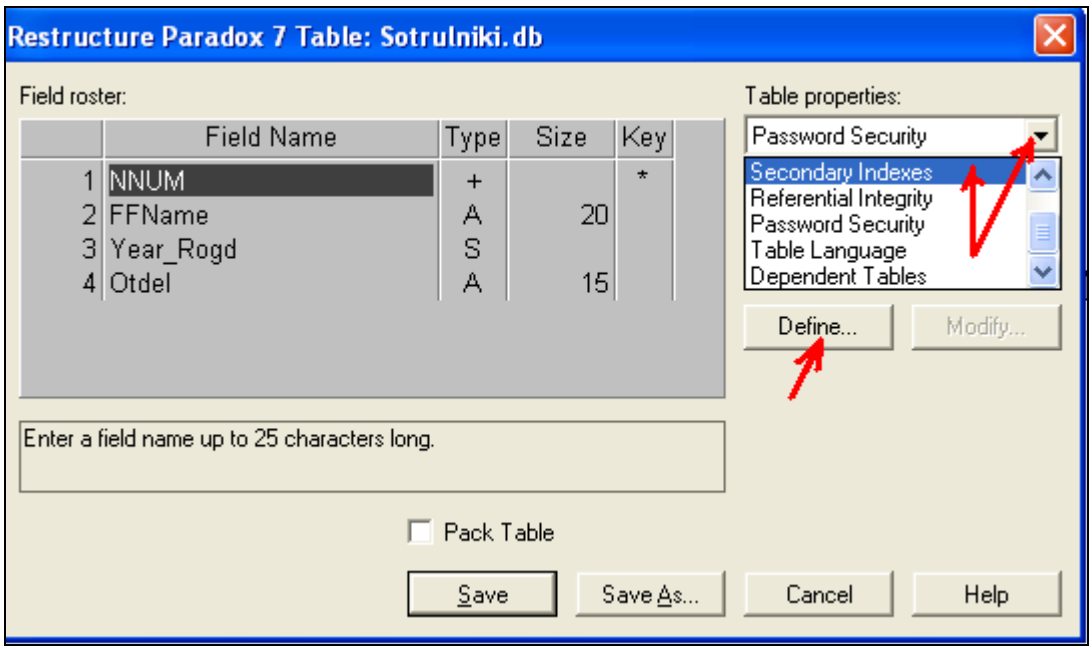


Рис.1.

Далее, воспользовавшись кнопкой со стрелкой ➔, перенести необходимое поле из списка полей *Fields* в список полей *Index Fields* (рис.2). 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.

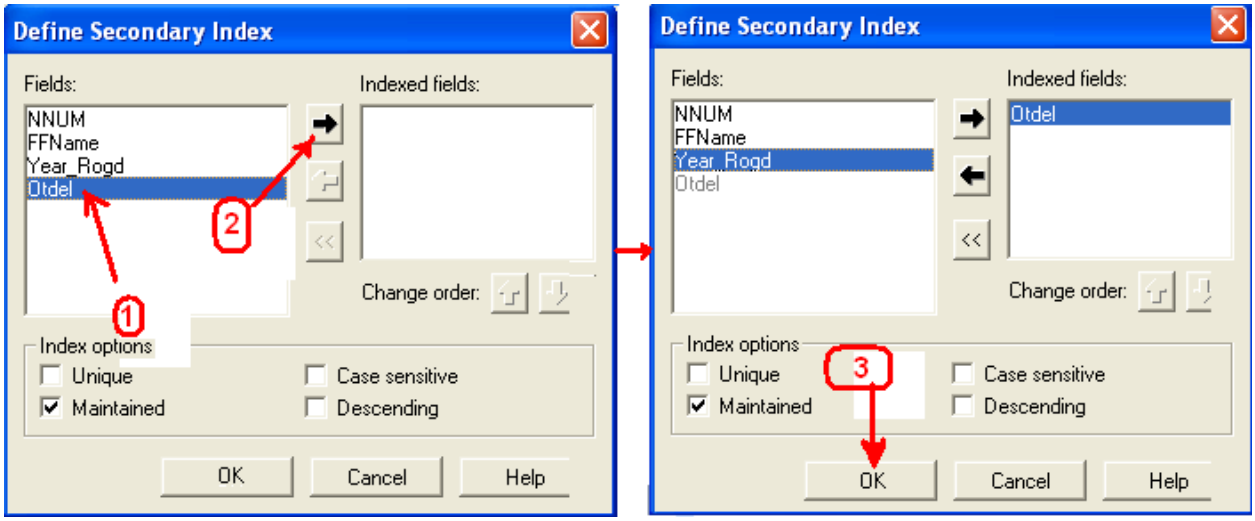


Рис.2

После подтверждения (*выполнения команды OK*) будет выведено окно присвоения *имени* индексу (рис.3). Имя индекса выбирается произвольно (мы ввели имя **Otd**).

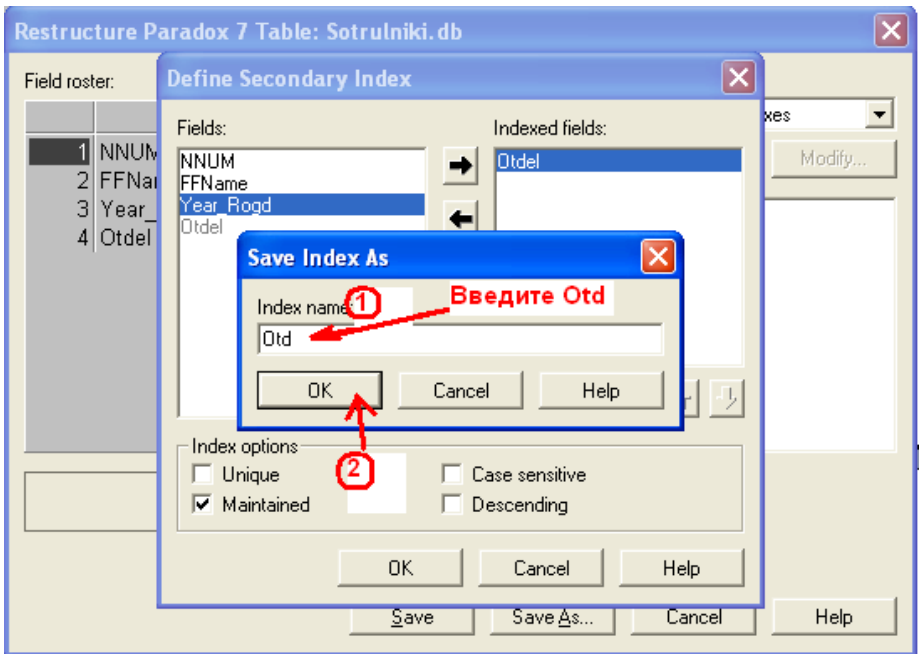


Рис.3

Индексированную таблицу необходимо сохранить, выполнив команду Save (или SaveAs, если это первичное сохранение).

При защите лабораторной работы знать:

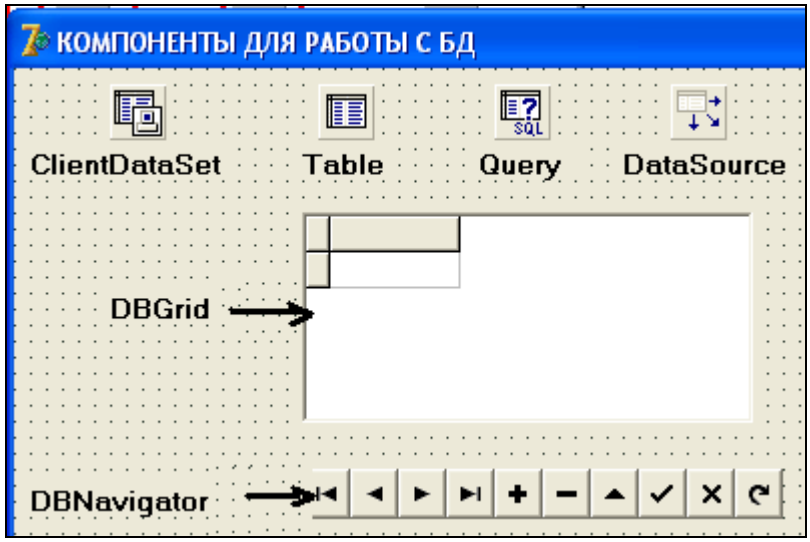
- что такое ключ (в терминологии БД) и для чего он нужен;
- что такое индекс (в терминологии БД) и для чего он нужен;
- чем ключ отличается от индекса;
- как проиндексировать поле в уже созданной таблице;
- как называется поле, у которого индекс имеет имя Primary;
- в каждой ли таблице, построенной на платформе **Paradox,** имеется поле, проиндексированное индексом Primary;
- где должно быть расположено поле с индексом Primary при создании структуры таблицы;
- по каким правилам выбирается имя для индекса.

Лабораторная работа №5

Тема Компоненты для работы с бд. Настройка связи между отношениями (связывание таблиц).

Среда **Delphi** предоставляет в распоряжение пользователя компоненты, позволяющие получить **доступ** к базам данных и осуществить их **редактирование**. Компоненты, расположенные на страницах **Data Access и BDE** палитры компонентов (в дальнейшем П/К) предназначены для *доступа* к базам данных. Компоненты, расположенные на странице **Data Controls**, представляют собой элементы *управления* данными. Эти компоненты подобны компонентам, расположенным на страницах **Standard** и **Additional**, однако отличаются от них тем, что имеют свойства, обеспечивающие связь с полями таблицы базы данных.

Компоненты для работы с базами данных



В данной лабораторной работе мы будем рассматривать:

компоненты доступа к данным

- **ClientDataSet** - клиентский набор данных (П/К **Data Access**);
- **Table** - таблица базы данных (П/К BDE);
- **Query** - предназначен для доступа к базе данных посредством SQL запроса (П/К BDE);
- **DataSource** - источник данных (П/К **Data Access**);

компоненты управления данными

- **DBGrid** - позволяет представить таблицу базы данных в виде похожем на электронную таблицу (П/К **Data Controls**);
- **DBNavigator** - представляет собой кнопочный переключатель, посредством которого можно перемещать курсор по записям таблицы и выполнять редактирование записей (П/К **Data Controls**).

Рассмотрим наиболее важные свойства представленных компонентов.

Table Наиболее важными свойствами компонента являются:

- **Active** – данное свойство служит для получения доступа к таблице базы данных. Этому свойству присваивается значение **true** после установки нижеследующих свойств.
- **DataBaseName** – в данном свойстве указывается имя базы данных, доступ к таблице которой должен получить компонент **Table**. Вместо имени базы

данных можно указывать ее псевдоним (алиас) или полный путь к каталогу, содержащему таблицы.

- **TableName** – в данном свойстве указывается конкретная таблица базы данных. При необходимости иметь в форме доступ к нескольким таблицам следует для каждой таблице определить свой компонент **Table**.

Компонент **Query**, как и компонент **Table** имеет свойство **DataBaseName**, но не имеет свойства **TableName**. Необходимая таблица создается автоматически при выполнении той или иной команды SQL. Свойству SQL присваивается текст одноименной команды, как при дизайне приложения, так и в процессе выполнения приложения. Для создания SQL команды в процессе дизайна приложения достаточно щелкнуть по кнопке расположенной рядом со свойством SQL в инспекторе объектов и в окне редактора команды ввести соответствующий текст SQL команды.

DataSource:

- свойство Enabled похоже на свойство Active таблицы или Connected у базы данных, т.е. делает активным или неактивным соединение;
- свойство AutoEdit, будучи включенным, обеспечивает возможность правки записей без написания какого-либо дополнительного кода;
- свойство State информирует о том, в каком состоянии в текущий момент находится источник данных;
- свойство DataSet определяет источник данных - таблицу, запрос и т.д.

DBGrid:

- DataSource, свойство, указывающее на источник данных, отображающихся в данной сетке (**Grid**-е).

DBNavigator:

- DataSource, свойство, указывающее на объект навигации (на таблицу, по записям которой предстоит выполнять перемещение указателя);
- + VisibleButtons – установка/удаление кнопок на навигаторе (пр умолчанию все кнопки на навигаторе видимы).

ВЫПОЛНИТЬ

Для выполнения данной работы Вы должны знать (вспомнить):

а) в какой папке сохранены таблицы (ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ), созданные на Лабораторной работе №3 – это должна быть папка **DATA**;

б) какой алиас Вы регистрировали для этой папки (Лаб№4). Если не помните алиас, то зарегистрируйте его вновь. В дальнейшем таблицы ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ будем называть БД «Предприятие».

1. Создать СУБД «Предприятие». В качестве компонента доступа к данным использовать компонент **Table**.

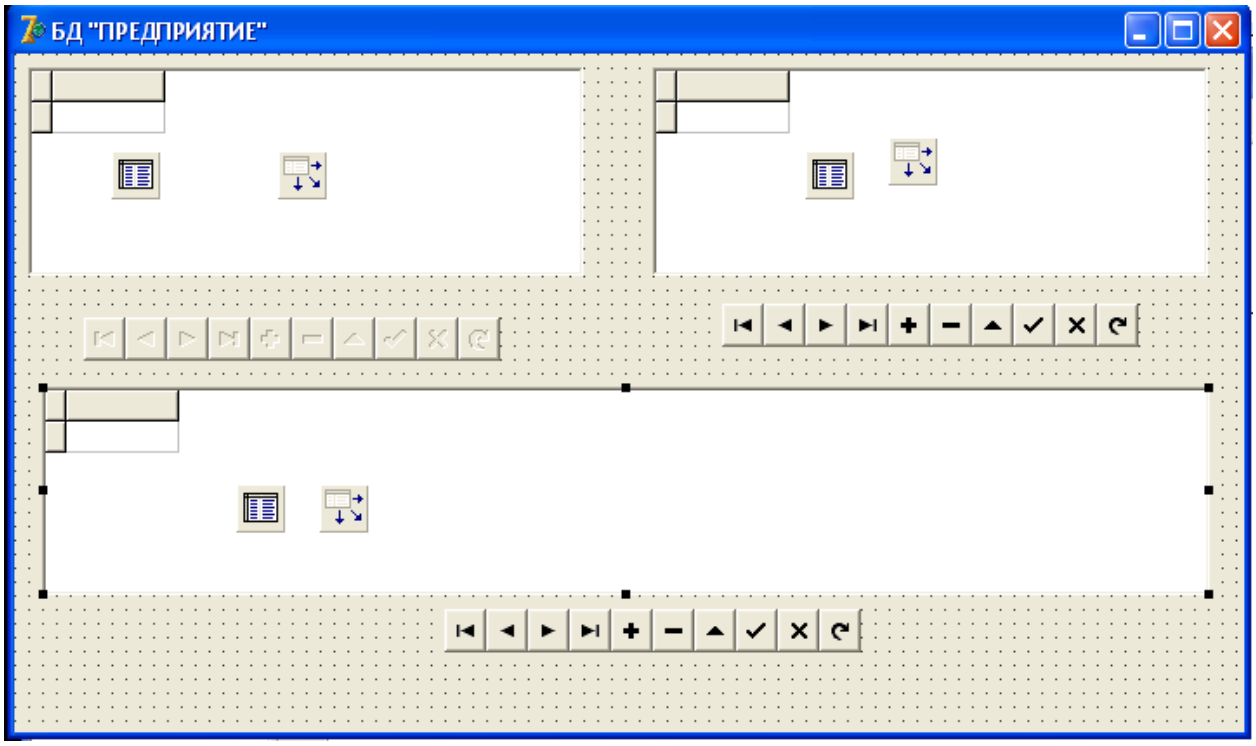
2. Далее пошагово описан путь создания приложения.

3. Создайте в своей личной папке папку Лабораторная_5.

4. Запустите среду Delphi и сохраните еще пустой проект в созданную папку (File → SaveProjectAs).

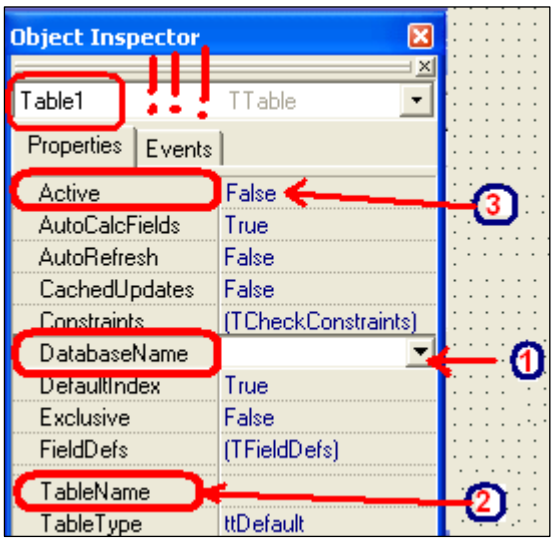
5. Нанесите на форму набор следующих компонентов:

- Table1, DataSource1, DBGrid1, DBNavigator1;
- Table2, DataSource2, DBGrid2, DBNavigator3.
- Table3, DataSource3, DBGrid3, DBNavigator3.



6. Активизируйте компонент **Table1** и в инспекторе объектов (далее И/О) для перечисленных ниже *свойств* установите значения:

- **DataBaseName** – тот алиас, который вы зарегистрировали на папку **DATA** (с таблицами ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ).



- **TableName** – имя таблицы ОТДЕЛЫ;
- **Active** → true.

7. Активизируйте компонент **Table2** и в инспекторе объектов для перечисленных ниже *свойств* установите значения:

- **DataBaseName** – тот алиас, который вы зарегистрировали на папку **DATA** (с таблицами ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ).
- **TableName** – имя таблицы СОТРУДНИКИ;
- **Active** → true.

8. Активизируйте компонент **Table3** и в инспекторе объектов для перечисленных ниже *свойств* установите значения:

- **DataBaseName** – тот алиас, который вы зарегистрировали на папку **DATA** (с таблицами ОТДЕЛЫ, СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКОВ).
- **TableName** – имя таблицы СЕМЬЯ СОТРУДНИКОВ;
- **Active** → true.

9. Активизируйте компонент DataSource1 и в инспекторе объектов для *свойства* DataSet установите значение **Table1**;

10. Активизируйте компонент DataSource2 и в инспекторе объектов для *свойства* DataSet установите значение **Table2**;

11. Активизируйте компонент DataSource3 и в инспекторе объектов для *свойства* DataSet установите значение **Table3**;

12. Активизируйте компонент **DBGrid1** и в инспекторе объектов для *свойства* DataSource установите значение DataSource1;

13. Активизируйте компонент **DBGrid2** и в инспекторе объектов для *свойства* DataSource установите значение DataSource2;

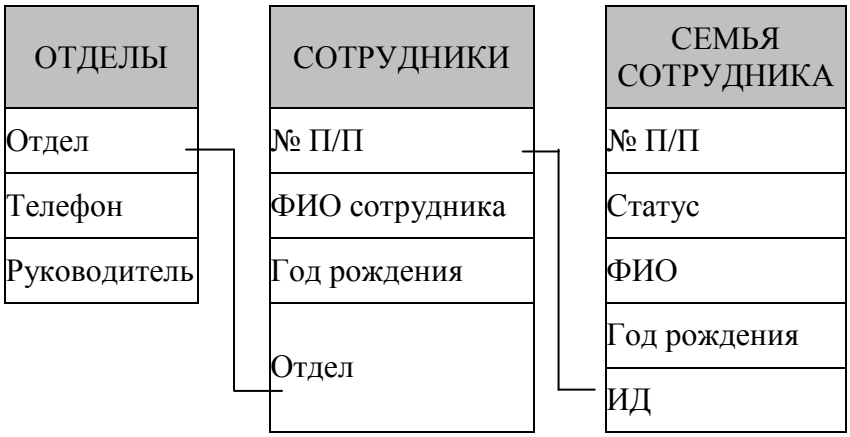
14. Активизируйте компонент **DBGrid3** и в инспекторе объектов для *свойства* DataSource установите значение DataSource3;

15. Активизируйте компонент **DBNavigator1** и в инспекторе объектов для *свойства* DataSource установите значение DataSource1;
16. Активизируйте компонент **DBNavigator2** и в инспекторе объектов для *свойства* DataSource установите значение DataSource2;
17. Активизируйте компонент **DBNavigator3** и в инспекторе объектов для *свойства* DataSource установите значение DataSource3;
18. Скомпилируйте приложение (Run→Run или F9), остановите его, и если нет ошибок, сохраните командой SaveAll.

Ранее оговаривалось, что мы имеем дело с реляционной БД, однако сами по себе таблицы никак не связаны. Сейчас приступаем к настройке механизма связывания таблиц (отношений) между собой. Внимание!!! Поля по которым настраивается связь **ОБЯЗАТЕЛЬНО** должны быть проиндексированы (толи первичный ключ, то ли вторичный индекс).

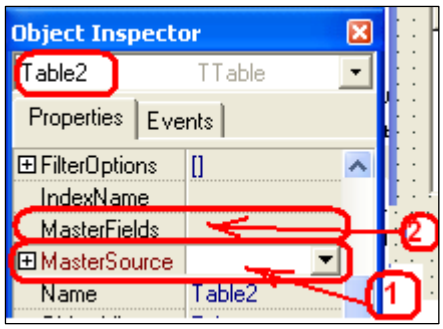
ПРАВИЛО. Связь всегда настраивается **ОТ** *подчиненной* таблицы (**detail-отношение**) *к главной* таблице (**master-отношение**).

Обратите внимание на логическую схему БД «Предприятие». Таблица «СОТРУДНИКИ» подчинена таблице «ОТДЕЛЫ», т.е. связь будет настраиваться от таблицы «СОТРУДНИКИ» →«ОТДЕЛЫ» по полям Отдел → Отдел соответственно.
Таблица «СЕМЬЯ СОТРУДНИКА» подчинена таблице«СОТРУДНИКИ», т.е. связь будет настраиваться от таблицы «СЕМЬЯ СОТРУДНИКА» «СОТРУДНИКИ» → «СОТРУДНИКИ» по полям ИД → №ПП соответственно.

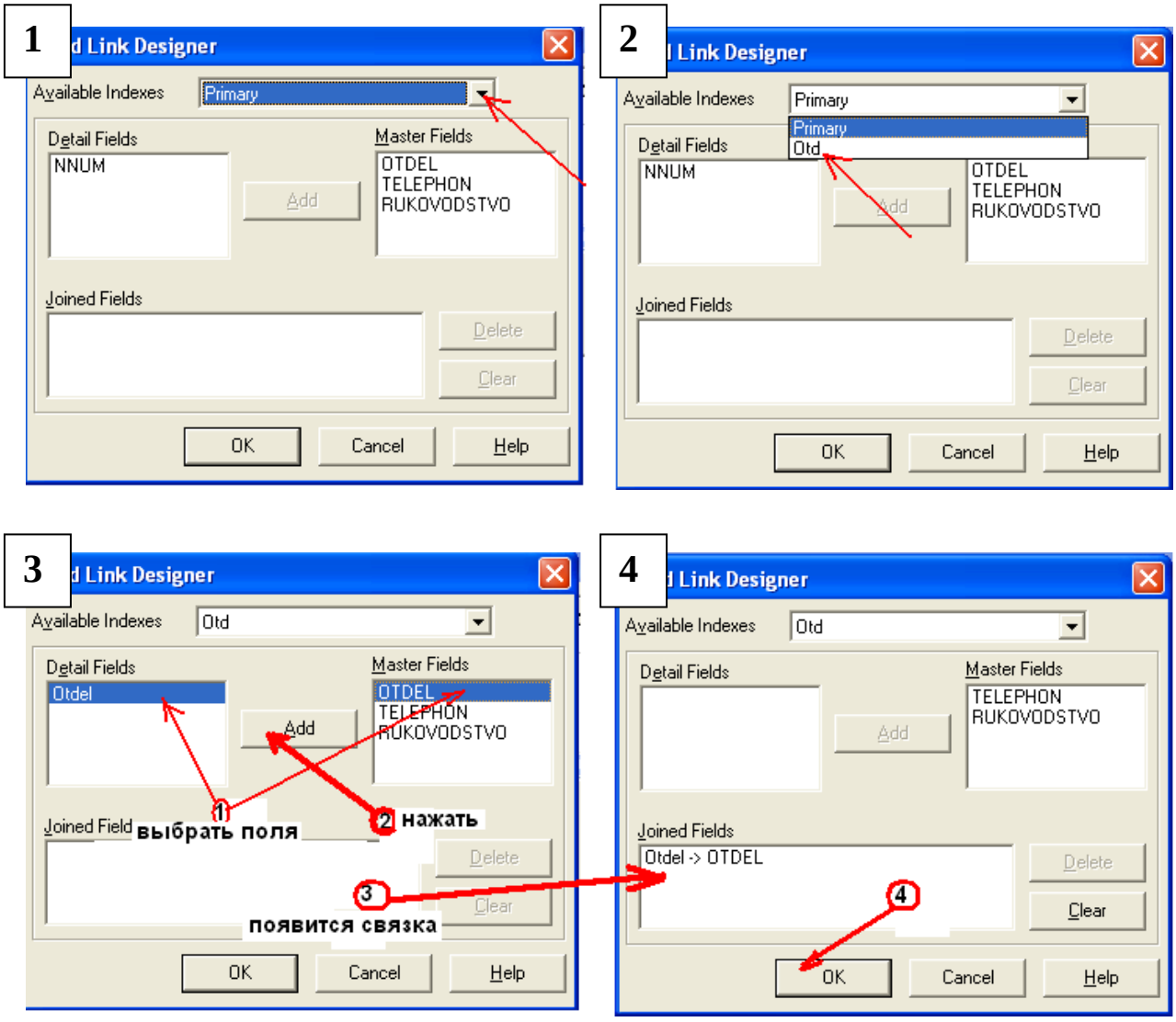


19. Выберите компонент **Table2 (** таблица «СОТРУДНИКИ») и в И/О вы берите:

- свойство MasterSource → DataSource1;
- свойство MasterFields → […] ;



Свойство MasterFields дает возможность связать между собой таблицы по нужным полям (это действие ещё называют *связывание курсоров*). Ниже представлен рисунок, показывающий последовательность настраивания связи между двумя таблицами: **master (**главная**)** и **detail (**подчиненная**).**



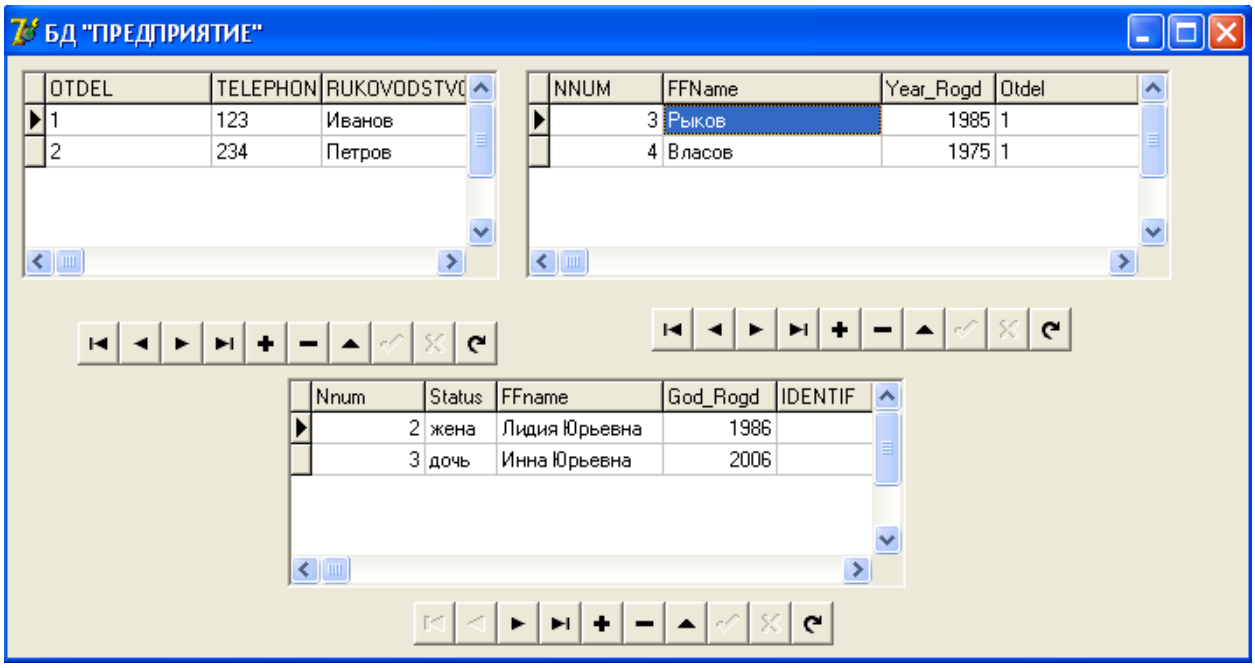
Описание рисунка. 1-2. В окне индексов Available Indexes выбирается индекс **ПОДЧИНЕННОГО** поля. 3. Этот индекс автоматически вносится в поле DetailFields. Затем мышкой выбирается поле связи главной (Master) таблицы и нажимается кнопка Add . 4. После добавления связки в поле JoinedField окно Мастера связи закрывается командой подтверждения OK.

20. Следующий шаг – это соединение таблиц «СЕМЬЯ СОТРУДНИКА» - Detail и «СОТРУДНИКИ»- Master, поля связи для этих таблиц ИД и №ПП соответственно. Не забудьте, что эти поля должны иметь индексы!!! Поле №ПП в соответствии с правилами создания структуры таблиц в формате Paradox долино быть ключевым – Primery, а вот для поля ИД должен быть установлен вторичный индекс.

21. Далее, по аналогии с пунктом 20 настройте связь между таблицами «СЕМЬЯ СОТРУДНИКА» и «СОТРУДНИКИ».

22. Скомпилируйте приложение (Run→Run или F9), остановите его, и если нет ошибок, сохраните командой SaveAll.

23. Вновь запустите приложение и попробуйте заполнить строки в таблицах.

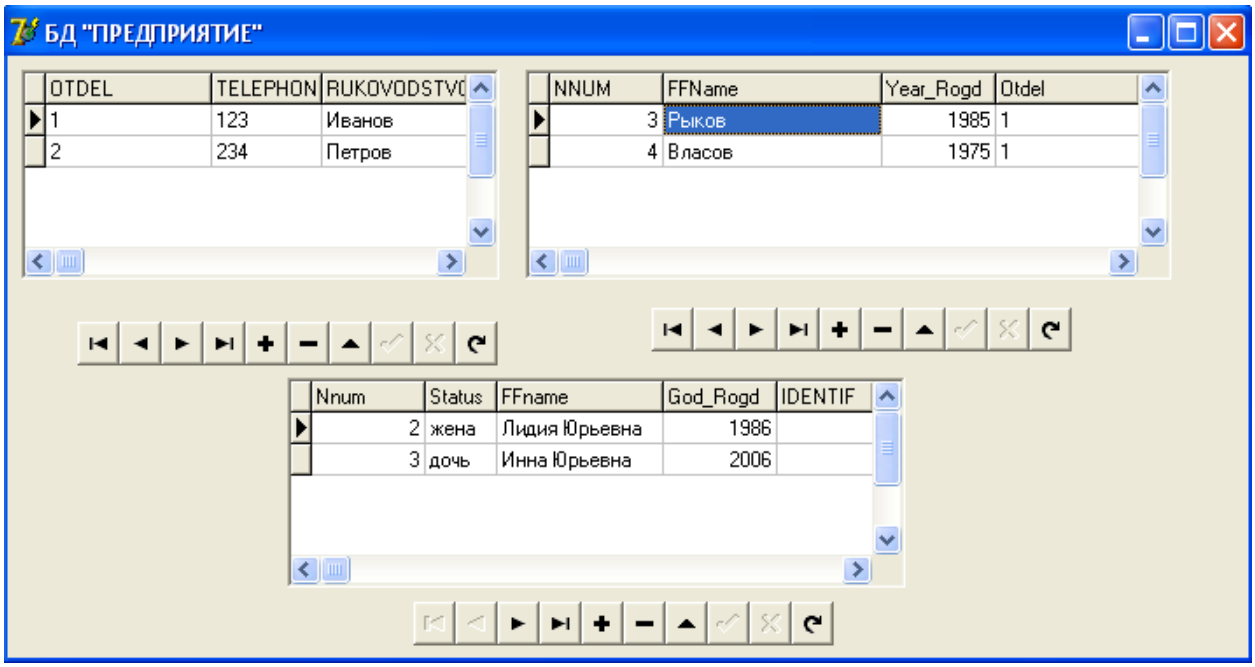


Лабораторная работа №6

Тема Организация поиска данных посредством компонента *Edit*. Доступ к данным в БД. Комонент *DBText*. Русификация заголовков столбцов в *DBGrid-е*.

ОРГАНИЗАЦИЯ ПОИСКА ДАННЫХ

В реальных *СУБД* часто применяется поиск необходимых записей по вводу первых символов отыскиваемого слова. Как известно поиск осуществляется по **индексированным** полям (*вторичный индекс*).



Организуем поиск руководителя конкретного отдела по фамилии, т.е. поиск будет вестись в первой таблице (компонент Table1) по полю RUKOVODSTVO. Для этого необходимо:

- проверить наличие *вторичного* индекса у поля RUKOVODSTVO и если это поле не проиндексировано, выполнить его индексацию (через утилиту *DataBase Desktop*);

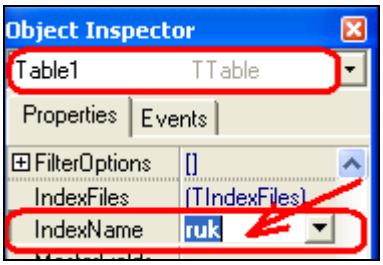
- установить на форму компонент *Label1*(он нужен для подписи действий, совершаемых в компоненте *Edit1*);

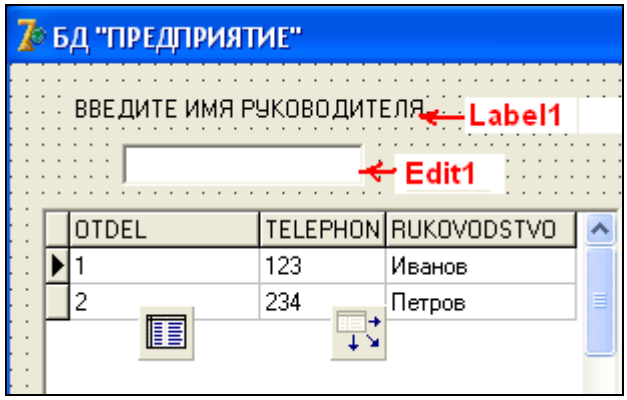
- установить на форму компонент *Edit1*;

- программная реализация процедуры поиска выполняется в обработчике события *OnChange* компонента *Edit1*:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
    table1.SetKey;
    table1.FieldByName('RUKOVODSTVO').AsString:=Edit1.Text;
    table1.GotoNearest;
end;
```

- для компонента *Table1* в свойстве *IndexName* установите имя вторичного индекса поля RUKOVODSTVO.





- для компонента ***Edit1*** в свойстве ***Text*** удалите значение и оставьте пустоту.

Скомпилируйте приложение (Run→Run или F9) и попробуйте ввести в компонент *Edit1* начальные буквы разыскиваемого сотрудника. Остановите приложение и если нет ошибок, сохраните командой SaveAll.

ВНИМАНИЕ!!! *Данный механизм поиска применим* ТОЛЬКО *к такой таблице БД, у которой для компонента* Table *свойство* IndexName НЕ ЗАДЕЙСТВОВАНО *для других операций*!!!

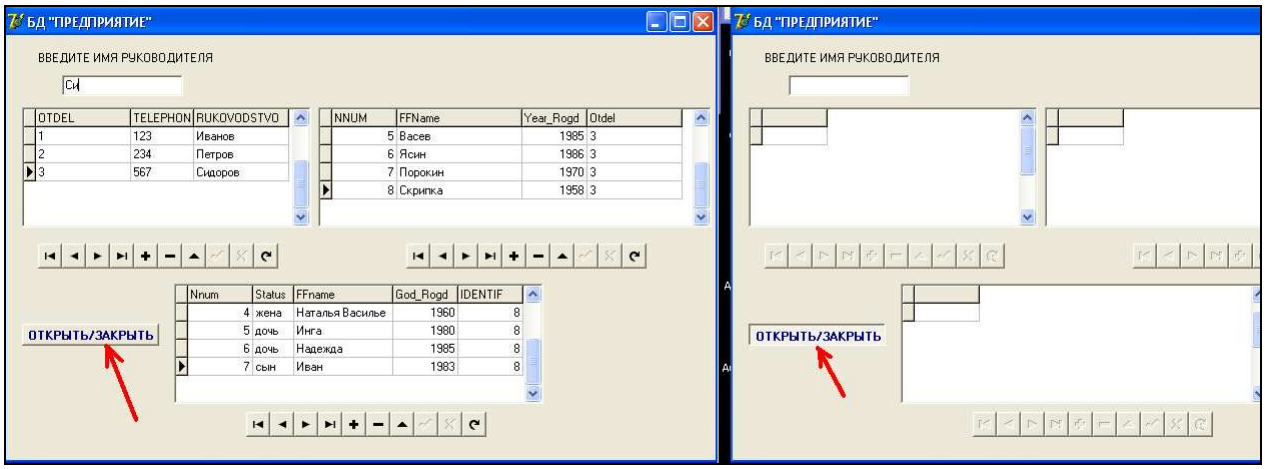
ДОСТУП К ДАННЫМ В БД

Установите на форму компонент SpeedButton1 (П/К Additional) и для нижеперечисленных свойств установите значения:

- AllowAllUp → true;
- GroupIndex → 1;
- Caption → ОТКРЫТЬ/ЗАКРЫТЬ.

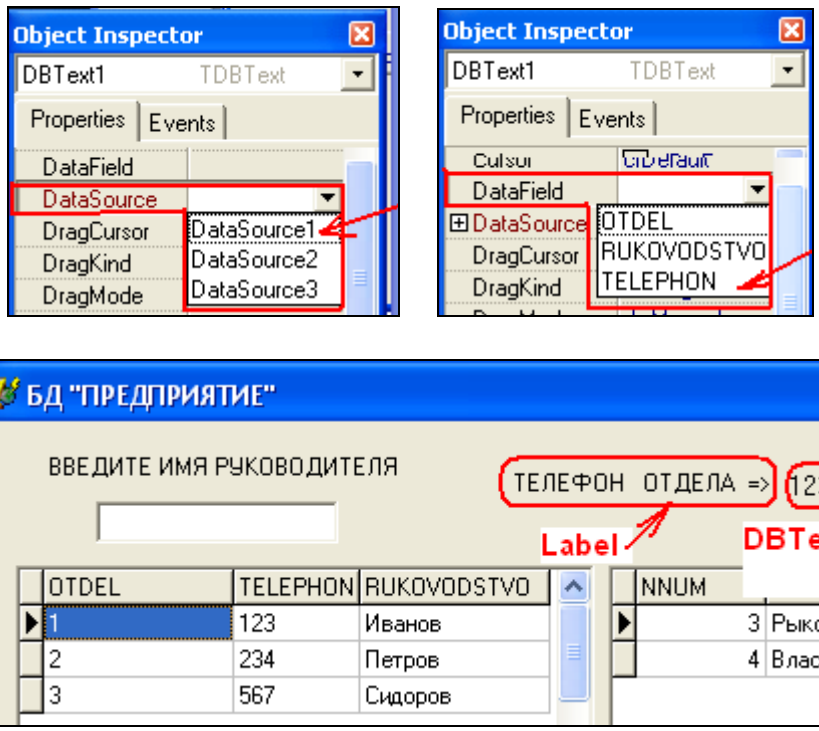
В обработчике событий *OnClik* кнопки *SpeedButton1* запишите следующий программный код:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
    if SpeedButton1.Down = true  then
      begin
        table1.Close;
        table2.Close;
        table3.Close;
      end
    else
      begin
        table1.Open;
        table2.Open;
        table3.Open;
       end;
end;
```



КОМОНЕНТ *DBText*

Иногда, для более качественного дизайна конструируемого приложения, информация об объекте может отображаться не в ***DBGrid-e***, а в специальном компоненте ***DBText***. В нашем приложении такой необходимости НЕТ!!! Однако рассмотрим применение компонента ***DBText,*** выведя в него телефон ОТДЕЛА. Для этого установим на форму компонент ***DBText*** (П/К DataControls) и посредством свойств DataSource и DataField выполним поставленную задачу:



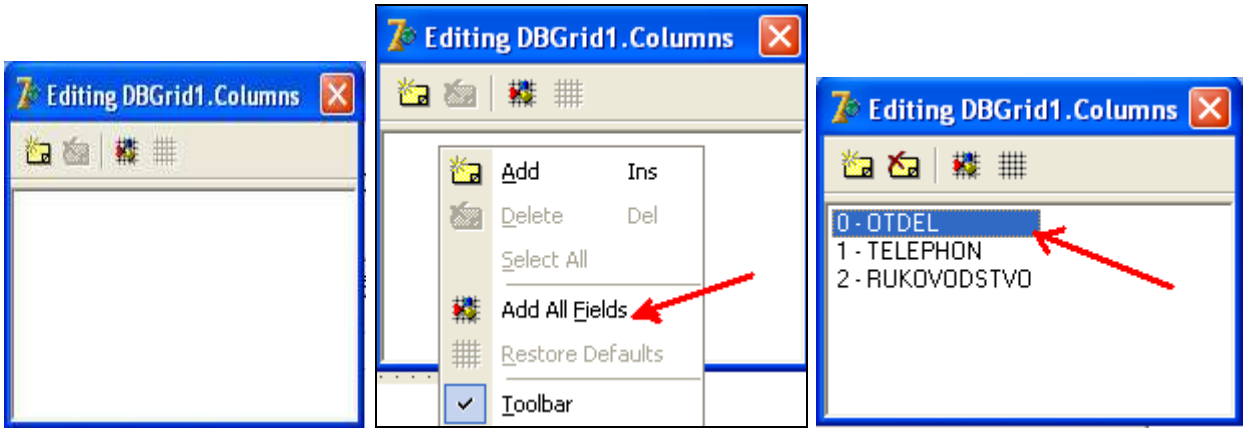
Русификация заголовков столбцов в *DBGrid-e*

Для возможности русификации заголовков столбцов в *DBGrid-e* необходимо поля таблицы присоединить (подключить) к самому компоненту *DBGrid*. Это выполняется следующим образом:

- выполнить двойной щелчок по компоненту DBGrid1;

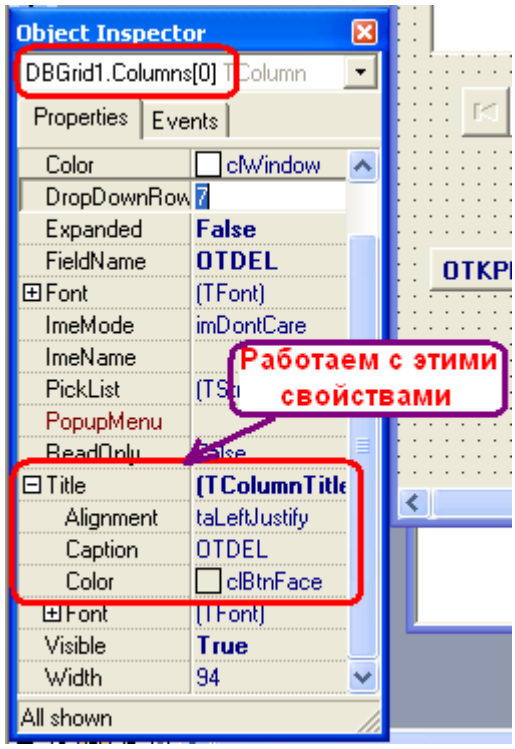
- в открывшемся окне *EditingDBGrid1.Columns* **(1)** вызвать контекстное меню и выбрать пункт *AddAllFields* **(2)**;

1) **2)** **3)**



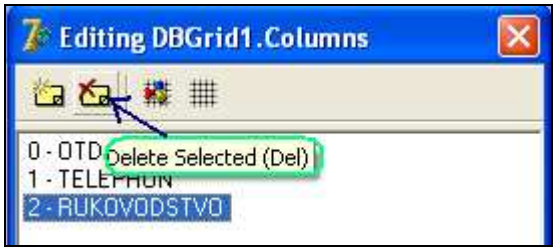
- после подключения всех полей к редактору *EditingDBGrid1* необходимо мышкой выделить первое поле **(3),** при этом в инспекторе объектов отобразятся **свойства** именно этого поля;

- далее работаем со свойством ⊞ Title и вложенными в него свойствами: Alignment (выравнивание положения заголовка), Caption (надпись самого заголовка), Color (заливка заголовка цветом);



УДАЛЕНИЕ ОТОБРАЖЕНИЯ НЕНУЖНЫХ СТОЛБЦОВ ИЗ DBGrid-a

Первоначально, при разработке приложения, в компоненте **DBGrid** отображаются все поля, составляющие структуру таблицы, однако никакой ценности эти поля для пользователя не имеют. К таким полям относятся: порядковый номер записи в таблице; иногда вспомогательные поля, по которым настраивается связь между таблицами и т.п. Для удаления ОТОБРАЖЕНИЯ этих полей из сетки (из компонента **DBGrid),** вызывают окно редактора компонента **DBGrid** и из него, путем нажатия клавиши Delete на клавиатуре или соответствующей кнопки в самом редакторе удаляют **ВЫДЕЛЕННОЕ** поле:



В данном примере будет удалено отображение поля **RUKOVODSTVO**.

ВЫПОЛНИТЬ

1. Организовать поиск в таблице ОТДЕЛЫ по номеру телефона.
2. Поставить на форму компонент ***GroupBox*** (П/К Standard), поместить в него два компонента ***DBText*** и самостоятельно вывести в них информацию из

таблиц СОТРУДНИКИ и СЕМЬЯ СОТРУДНИКА. Для компонента ***GroupBox*** самостоятельно изучить свойства:

- Align ;
- Caption;
- Color;
- Cursor;
- Font.

3. Удалить отображение поля Nnum и IDENTIF из таблицы «СЕМЬЯ СОТРУДНИКА».
4. Русифицировать заголовки всех полей в таблице «СЕМЬЯ СОТРУДНИКА».

При защите лабораторной работы знать ответы на такие вопросы:
- какой компонент применяется для организации поиска определенной информации;
- понимать программный код, реализующий поиск информации;
- обязательно ли должно быть проиндексировано поле, по которому будет выполнятся поиск информации, ответ обоснуйте;
- уметь самостоятельно написать (и объяснить) программный код для открытия и закрытия доступа данных через компонент Table;
- назначение компонента **DBText** и его основные свойства;
- назначение компонента **GroupBox** и его основные свойства;
- как вызвать редактор **EditingDBGrid,** для чего он применяется.

ЛАБОРАТОРНАЯ РАБОТА №7

Тема ФИЛЬТРАЦИЯ ДАННЫХ

Фильтрация данных выполняется при помощи элемента управления ComboBox. Реализация фильтра осуществляется исполнением строк программы в процедуре обработки события OnChange (обработчик события обмена данными) этого компонента.

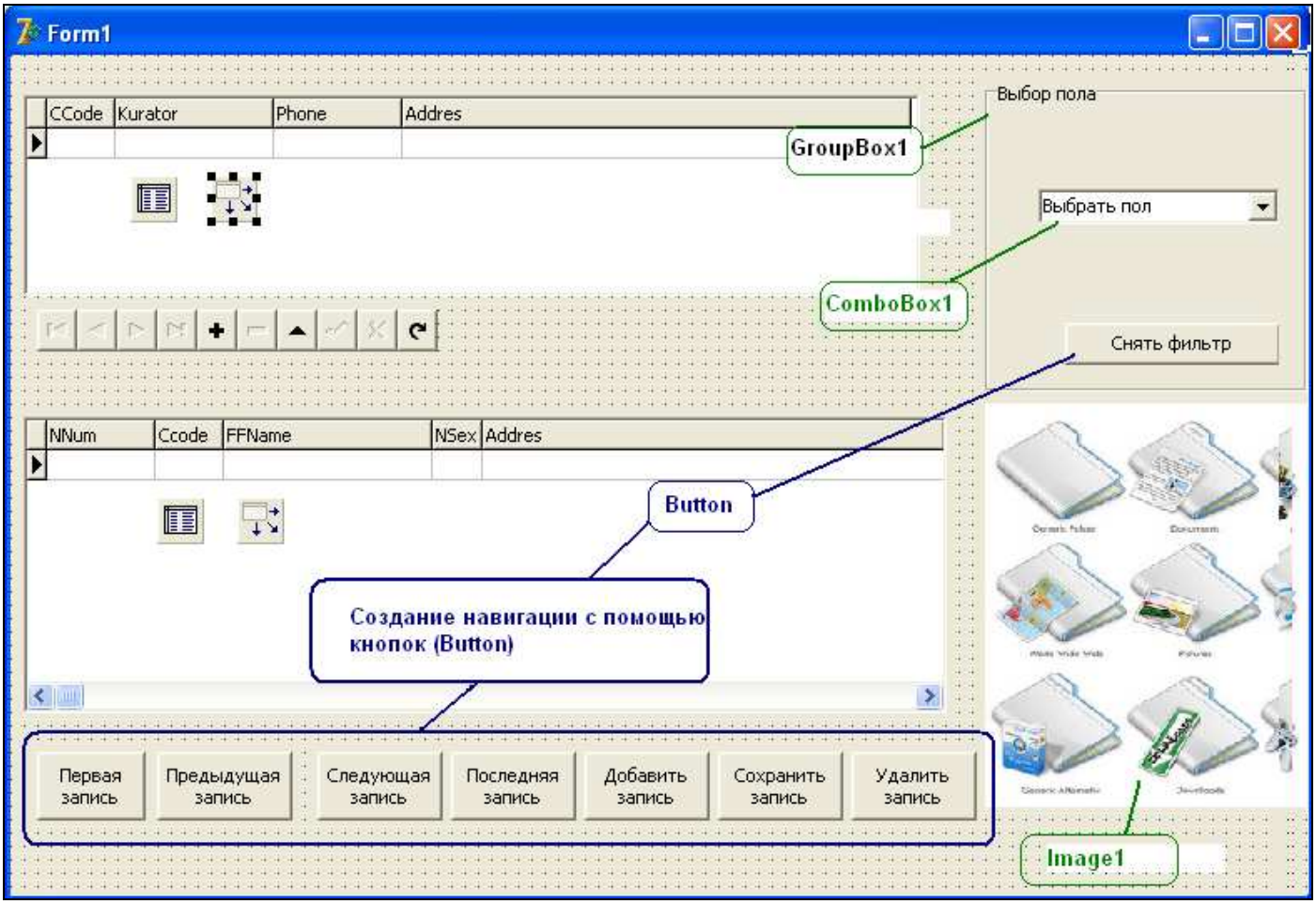
Выполнить
1. Создать в Вашей папке папку Лабораторная7 и зарегистрировать на неё алиас Student.

2. Создать (и сохранить в папке Лабораторная7) через утилиту DatabaseDesktop структуру следующих двух таблиц:

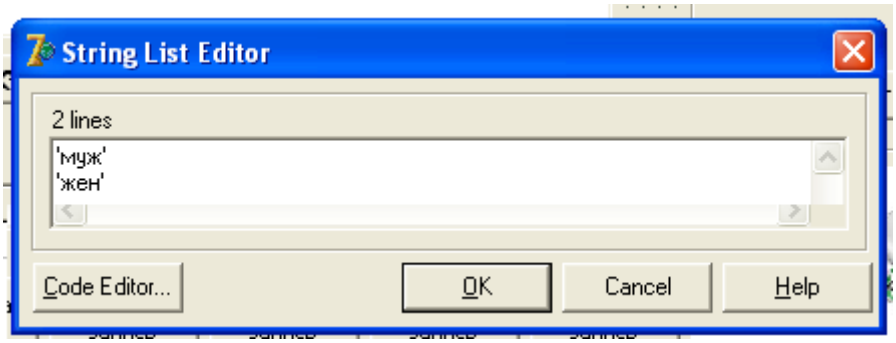
ГРУППА		
Шифр группы	A	6
ФИО куратора	A	15
Тел.куратора	A	10
Дом.адрес куратора	A	50

СТУДЕНТЫ		
№ ПП	+	
Шифр группы	A	6
ФИО студента	A	15
Пол	A	3
Дом.адрес	A	50

3. Нанести на форму компоненты в соответствии с рисунком:



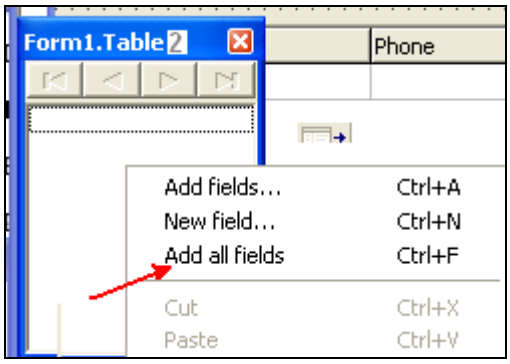
4. Соединить таблицы между собой в соответствии с логической схемой.
5. Для компонента ComboBox1 для свойства Items → [...] внести следующее значение (строки):



6. С целью обеспечения доступа из программы к свойству *Filter* таблицы Студенты (Table2) необходимо присоединить к имени таблицы поля таблицы. Данная операция выполняется в следующей последовательности:

а) Выбрать в форме компонент *Table2* и выполнить двойной щелчок указателем мыши по компоненту.

б) В открывшемся окне необходимо вызвать контекстное меню (правая кнопка мыши) и выполнить команду *Add all Fields*.



После выполнения данной команды в окне *Form1 Table2* появится список всех полей таблицы.

Для компонента **ComboBox1** в обработчик события **OnChange** внести следующую программу:

```
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
Table2.Filtered := false;
Table2.Filter := 'NSex=' + ComboBox1.Text;
Table2.Filtered := true;
end;
```

Чтобы снять действие фильтра для кнопки Снять фильтр внесите следующую программу:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     Table2.Filtered := false;
end;
```

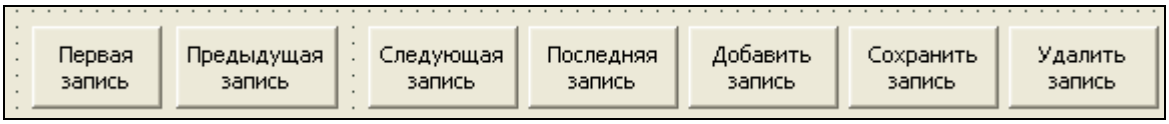
7. В компонент **Image1** внесите любую картинку.

8. **СОЗДАНИЕ КНОПОЧНОГО НАВИГАТОРА**.

Действие методов:

- Table.First – возвращает указатель на первую запись в таблице;
- Table.Prior - возвращает указатель на в таблице;
- Table.Next - возвращает указатель на следующую запись в таблице;
- Table.Last - возвращает указатель на последнюю запись в таблице;
- Table.Insert – добавляет запись в таблицу;
- Table.Delete – удаляет строку (запись) из таблицы;
- Table.Post- сохраняет внесенную в таблицу запись.

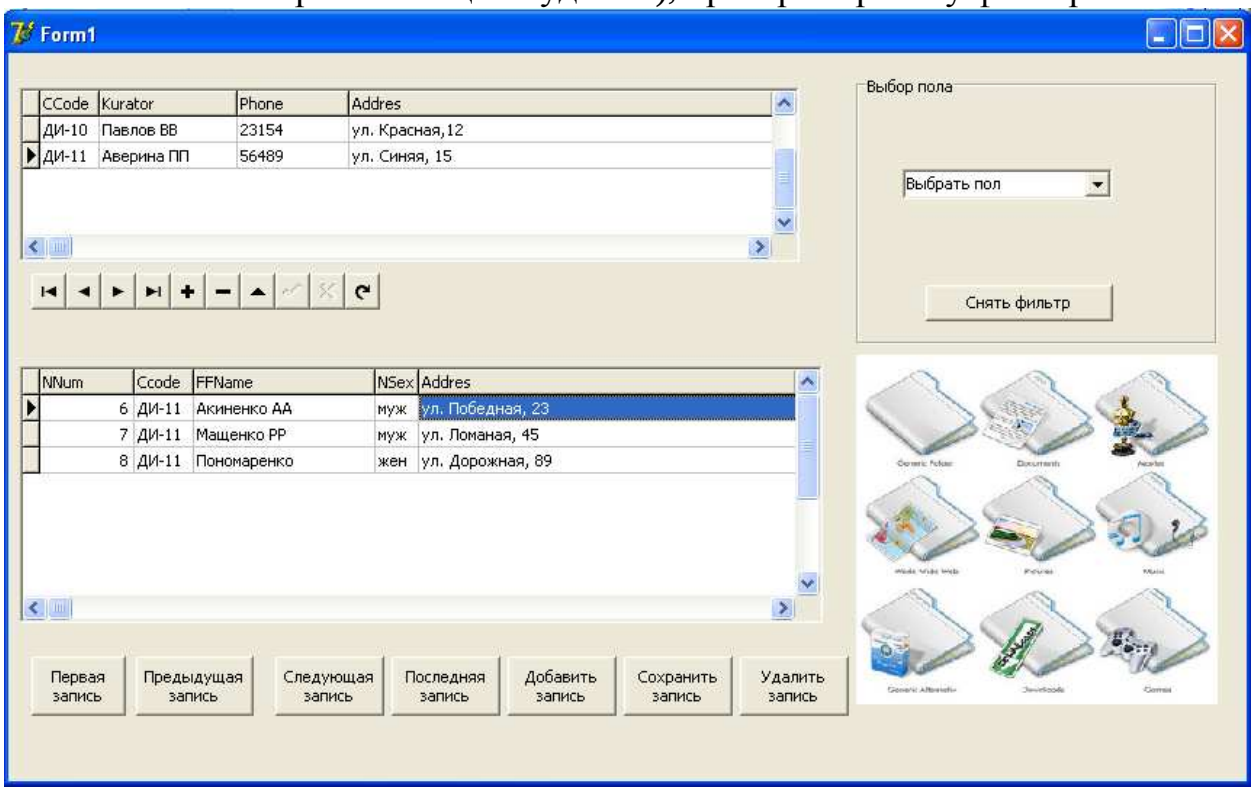
В соответствии с описанными выше методами запрограммируйте кнопочный навигатор (кнопки)



используя такие программы:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Table2.First;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
Table2.Last;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Table2.Prior;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
Table2.Next;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
Table2.Insert;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
Table2.Post;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
Table2.Delete;
end;
```

Запустите приложение, заполните таблицы (три записи в таблице Группа и восемь записей во второй таблице Студенты), проверьте работу фильтра.



9. Далее выполняется завершение дизайна: русификация заголовков полей в таблицах, центрирование надписей, выделение цветом некоторых компонентов (например GroupBox1).

ЛАБОРАТОРНАЯ РАБОТА №8

Тема ЗАКРЕПЛЕНИЕ НАВЫКОВ, ПРИОБРЕТЕННЫХ НА ЛАБОРАТОРНЫХ РАБОТАХ №1-7

Необходимо построить СУБД для детского садика. Очерчена предметная область, в которой выявлены такие объекты и их атрибуты: группа (имеется в виду: ясельная, младшая, средняя и подготовительная), в каждой группе два воспитателя и одна нянечка, в группе имеется телефон, по каждому ребенку детсада известны: ФИО ребенка, год рождения, домашний телефон, телефон одного из родителей и домашний адрес.

Выполнить:

- построить логическую схему БД (в тетради);
- создать структуру таблиц на платформе Paradox7;
- создать в среде Delphi приложение, позволяющее: заносить данные в БД «Детский сад» и редактировать их.

Приложение должно иметь следующее наполнение:

- для наглядности установить два компонента GroupBox, в каждом из которых разместить по компоненту DBGride;
- каждый GroupBox должен иметь свой цвет;
- организовать поиск по фамилиям воспитателей, работающих в данном садике;
- организовать программный доступ к данным (имеется в виду установка кнопки, которая открывает и закрывает отображение таблиц в DBGride, лабораторная работа №6);
- русифицировать заголовки полей в DBGride.