Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра информационных систем

СИСТЕМЫ БАЗ ДАННЫХ

Методические указания к лабораторным работам
для студентов специальности
5В070300 – Информационные системы

Алматы 2014

СОСТАВИТЕЛЬ: Н.А. Водолазкина, Б.К. Каирбаева Системы баз данных. Методические указания к выполнению лабораторных работ для студентов специальности 5В070300 – Информационные системы – Алматы: АУЭС, 2014 – 40 стр.

Методические уаказания содержат необходимые материалы для выполнения лабораторных работ , контрольные вопросы и перечень рекомендуемой литературы по дисциплине «Системы баз данных ».

Методические указания предназначены для бакалавров специальности 5В070300 – Информационные системы.

Ил.-1, библиогр.- 4 назв.

Рецензент: к.т.н., доцент Куликов А.А.

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2013 г.

Содержание

Введение

После разработки проекта базы данных можно приступать к его реализации в конкретной *системе баз данных* (СБД). Современная волна информационных технологий управления данными основывается на использовании систем управления *реляционными* базами данных.

Спроектировать и создать базу данных еще недостаточно для успешной работы с данными: у нас должна быть возможность как-то обращаться к этим данным, дополнять и изменять их. В этом и состоит цель *клиентского приложения* - специальной программы, позволяющей работать с данными в удобной для пользователя форме. В общем случае с одной базой данных могут работать множество различных приложений. При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно и независимо друг от друга, и именно СБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями.

Среда программирования Delphi позволяет сравнительно легко и быстро создавать законченные приложения Windows. Поэтому она получила название RAD (Rapid Application Development – среда быстрой разработки приложение).

Среда программирования Delphi имеет встроенный компилятор, переводящий текст программы в машинный код, понятный процессору компьютера. В настоящее время этот компилятор является самым быстрым и позволяет обеспечивать высокую производительность среды программирования, необходимую для создания приложений Windows. Компилятор среды программирования Delphi создает автономные exe-файлы (исполняемые файлы). Поэтому приложения, созданные в Dephi, будут работать на компьютере, на котором не установлена среда Delphi.

Предлагаемые методические указания к выполнению лабораторных работ разработаны для освоения студентами основ разработки баз данных в среде Delphi. Описание каждой лабораторной работы содержит необходимую для его выполнения информацию из сопутствующего курса, пример решения аналогичной задачи.

По каждой лабораторной работе студент должен представить отчет и защитить работу.

1 Лабораторная работа № 1. Работа с Database Desktop, создание псевдонима БД

Цель работы:

1) Работа с утилитой Database Desktop.

2) Изучить процесс создания и редактирования таблиц с помощью Database Desktop.

Общие сведения

Database Desktop.

Database Desktop - это утилита, которая поставляется вместе с Delphi для интерактивной работы с таблицами различных форматов локальных баз данных - Paradox и dBase, а также SQL-серверных баз данных InterBase, Oracle, Informix, Sybase (с использованием SQL Links). Исполняемый файл утилиты называется DBD32.EXE. Для запуска Database Desktop просто дважды щелкните по ее иконке.

Запуск Database Desktop:

1) Запустить Delphi.
2) В меню Delphi выбрать раздел Tools.
3) В появившемся списке выбрать строку Database Desktop.
4) Щелкнуть по кнопке Пуск.
5) Из главного меню выбрать строку Программы.
6) В появившемся списке выбрать строку Delphi 7.
7) В следующем списке выбрать строку Database Desktop.
8) После запуска Database Desktop на экране появится окно (см. рисунок 1.1).

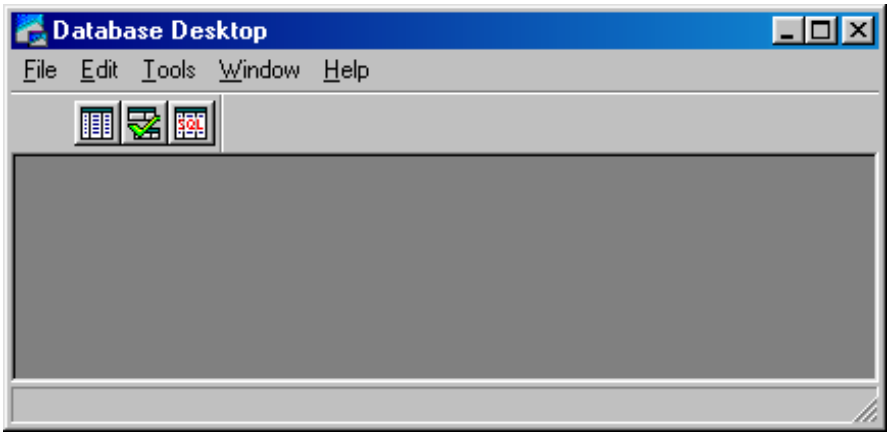

Рисунок 1.1

Создание таблиц в Database Desktop.

Запускаем программу Database Desktop;

В появившемся окне выбираем: File►New►Table (см. рисунок 1.2).

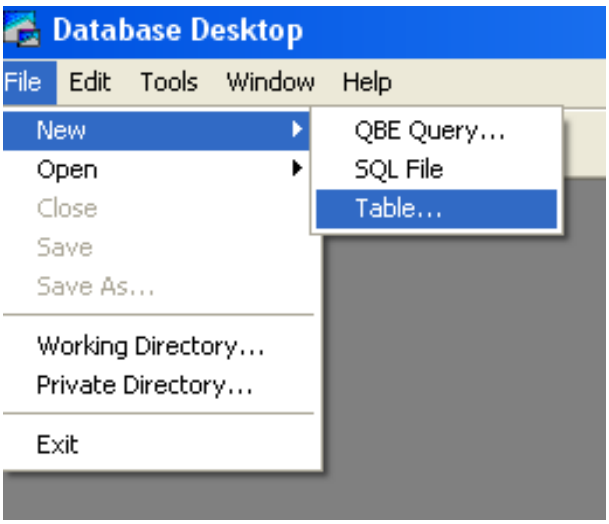

Рисунок 1.2

Формат таблицы выбираем Paradox. После этого появится окно создания таблицы, в котором можно определить поля таблицы и их тип (см. рисунок 1.3).

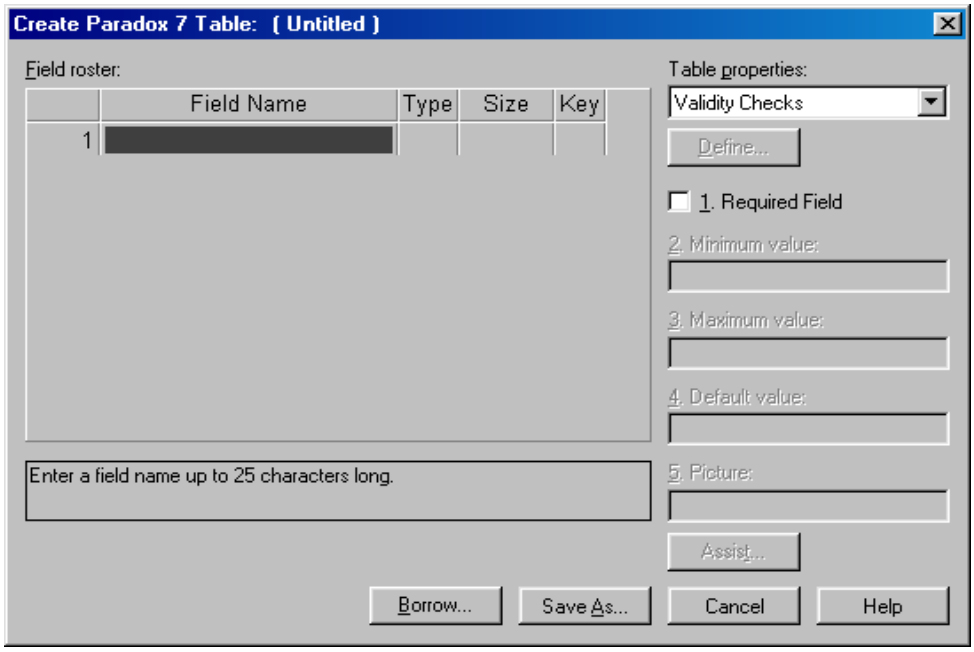

Рисунок 1.3

По умолчанию сразу после открытия окна в правой его части в списке Table properties выбран пункт Validity Chekcs, что позволяет контролировать содержимое полей. С помощью флажка Required Fields можно потребовать обязательного заполнения поля при вводе новой записи. Также можно контролировать минимальное и максимальное значение числового поля в строках Minimum Value и Maximum Value. В строке Default Value можно указать значение поля по умолчанию – при вводе новой записи значение в это поле поместит BDE. С помощью строки Picture можно задать шаблон для автоматического форматирования значения поля. Например, если задан

шаблон (###)###-#### и в поле введена строка 9054005647, она будет автоматически преобразована к виду (905)400-5647.

Кнопки:

-Borrow… – осуществляет копирование структуры таблицы из другой таблицы.

-Save as… – сохраняет изменения в структуре таблицы.

-Cancel – выход без сохранения.

-Help – вызов справки.

Чтобы определить структуру таблицы в этом окне необходимо заполнить следующие графы:

-Field Name - Имя поля.

-Type - Тип поля. Вызывает список допустимых типов, щелчком правой кнопки мыши или клавишей пробел.

-Size - Размер. Определяет размер поля. Не все типы полей имеют размер.

Большинство типов имеют стандартный размер, который не может быть изменен. -Размер в основном меняется у строковых типов (Alpha), бинарных (Binary) и др.

-Key - Ключ. Двойной щелчок мышью определяет ключевое поле. Ключевыми могут быть только первые поля, то есть второе поле сможет быть ключевым только вместе с первым.

Таблица 1.1Типы полей формата Paradox

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 - от 10-308 до 10308 с 15 значащими цифрами
$ (Money)	числовое поле, значение которого может быть положительным и отрицательным. По умолчанию, является форматированным 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 немного меньше, чем в других числовых форматах, однако точность - гораздо выше. Может иметь 0-32 цифр 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 символов, суммарная длина которых более 255 байт. Может иметь любую длину. При этом 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 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. Однако, Delphi в стандартной поставке не обладает возможностью работать с полями типа Formatted Memo
Graphic	поле, содержащее графическую информацию. Может иметь любую длину. Смысл размера - такой же, как и в Formatted Memo. Database Desktop “умеет” создавать поля типа Graphic, однако наполнять их можно только в приложении
OLE	поле, содержащее OLE-данные (Object Linking and Embedding) - образы, звук, видео, документы - которые для своей обработки вызывают создавшее их приложение. Может иметь любую длину. Смысл размера - такой же, как и в Formatted Memo. Database Desktop “умеет” создавать поля типа OLE, однако наполнять их можно только в приложении. Delphi “напрямую” не умеет работать с OLE-полями, но это легко обходится путем использования потоков
Logical	поле длиной 1 байт, которое может содержать только два значения - T (true, истина) или F (false, ложь). Допускаются строчные и прописные буквы
+ (Autoincrement)	поле длиной 4 байта, содержащее нередактируемое (read-only) значение типа long integer. Значение этого поля автоматически увеличивается (начиная с 1) с шагом 1 - это очень удобно для создания уникального идентификатора записи (физический номер записи не может служить ее идентификатором, поскольку в Парадоксе таковой отсутствует. В InterBase также отсутствуют физические номера записей, но отсутствует и поле Autoincrement. Его с успехом заменяет встроенная функция Gen_id, которую удобней всего применять в триггерах)
Binary	поле, содержащее любую двоичную информацию. 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. Это полнейший аналог поля BLOb в InterBase
Bytes	строка цифр длиной 1-255 байт, содержащая любые данные

Задания А, Б.

Выбрав вариант согласно последней цифре учебного шифра, создать псевдоним БД (см. приложения А), создать по 2 таблицы, отредактировать их (формат Paradox). Студентам все задания выполнять по двум вариантам (А, Б).

Методические указания.

Создадим таблицу с данными о студентах. Укажем такие данные, как: ФИО, дата рождения, на основании какого приказа студент принят в университет, номер зачетки, специальность, курс, стипендия. Во всех текстовых полях, необходимо указать размер (см. рисунок 1.4).

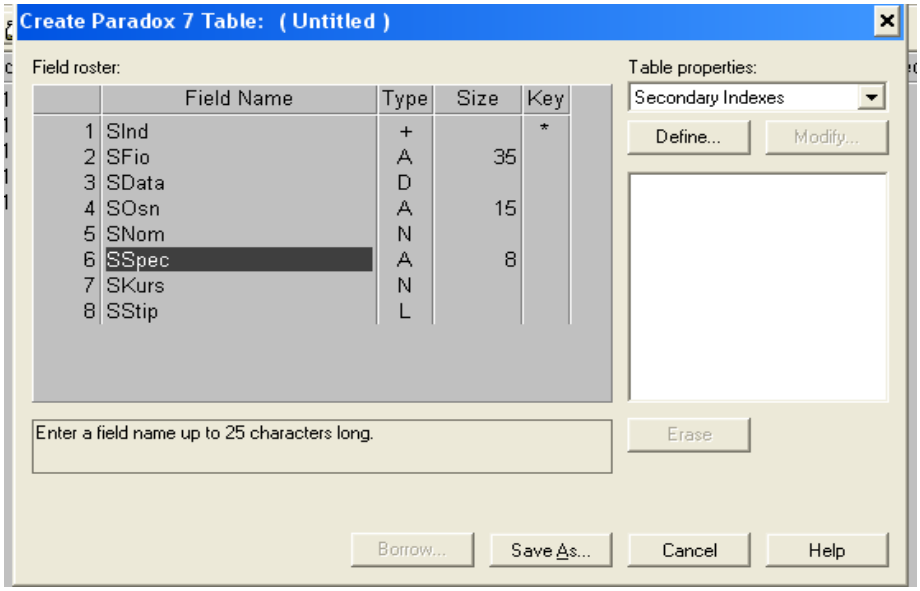

Рисунок 1.4

Созданную таблицу сохраняем под названием Student.db и закрываем окно создания таблиц. Иногда может понадобиться отредактировать уже созданную таблицу для того, чтобы добавить, изменить или удалить некоторые поля, изменить свойства таблицы.

Редактирование таблицы.

Определение вторичных индексов:

Открыть таблицу Student.db (File►Open►Table).

Из меню Table выбрать пункт Restructure. Откроется окно редактирования полей таблицы. В выпадающем списке Table properties выбрать Secondary Indexes и нажать кнопку Define. В окне Define Secondary Index определяются вторичные индексы (см. рисунок 1.5).

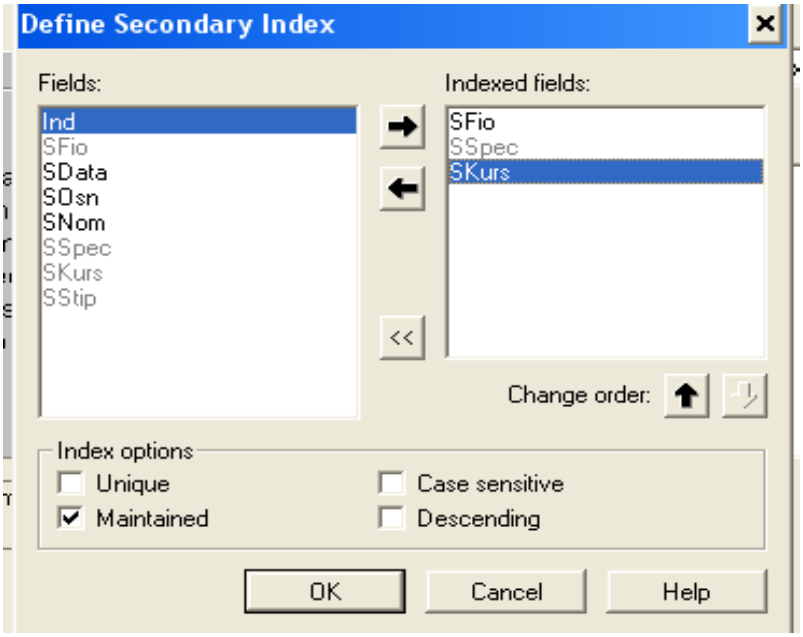

Рисунок 1.5

С помощью флажков группы Index options можно определить следующие особенности индекса:

Unique – индекс будет содержать уникальные значения;

Maintained – индексные поля сортируются по возрастанию значений;

Case sensitive – индекс чувствителен к регистру букв в текстовых полях;

Descending – индексные поля сортируются по убыванию значения.

Выбрать «SFio» из списка Fields и нажать кнопку с изображенной стрелкой вправо. В списке Indexed fields (индексированные поля) появится «SFio». То же самое проделать с полями «SSpec» и «SKurs». Закрыть окно Define Secondary Index

В появившемся окне ввести имя индекса IDFio и нажать "OK" (см. рисунок 1.6).

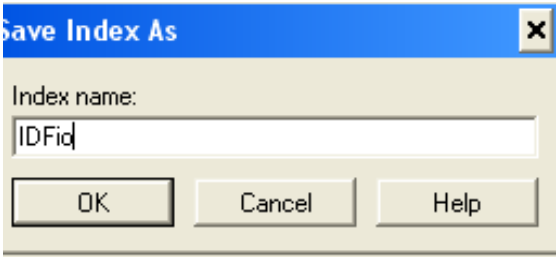

Рисунок 1.6

Теперь, так как на факультете всего две специальности, то можно переопределить тип поля «SSpec». Для работы будет гораздо удобнее, чтоб это поле было типа Logical. Навести курсор на тип поля «SSpec» и написать тип поля «L».

Определим языковой драйвер. Это следует делать для правильного отображения русскоязычного текста. Открыть окно редактирования полей таблицы Student.db.

В выпадающем списке Table properties выбрать Table Language и нажать кнопку Modify. В появившемся окне выбрать из списка Pdox ANSI Cyrillic.

Сохранить таблицу. Для логического типа значение по умолчанию зададим False.

Заполнение данными в таблице Student.db.

1) Открыть таблицу (File ► Open ► Table).
2) Выбрать Table ► Edit data или нажать клавишу F9.
3) Создать несколько записей.

Создание псевдонима.

Псевдоним указывает местонахождение файлов БД и представляет собой специальное имя для обозначения каталога. Использование псевдонимов существенно облегчает перенос файлов БД в другие каталоги и на другие компьютеры. При этом не требуется изменять приложение, которое осуществляет доступ к таблицам БД. Если в приложение местонахождения таблицы указано с помощью псевдонима, то после перемещения БД для обеспечения работоспособности приложения достаточно изменить путь, на который указывает псевдоним. Если же в приложение путь к БД указан в явном виде, то есть без псевдонима, то после перемещения БД нужно перемещать само приложение – вносить изменения в исходный код и заново его транслировать.

Регистрация псевдонима.

Воспользуемся приложением SQL Explorer, запускаемым командой Database►Explore. В левой части окна приводится список всех зарегистрированных в системе BDE баз данных, в правой – свойства текущей базы, выбранной в списке.

Создадим псевдоним для базы данных. Для этого выполним команду Object►New и в диалоговом окне выбора драйвера укажем значение Standart (см. рисунок 1.7).

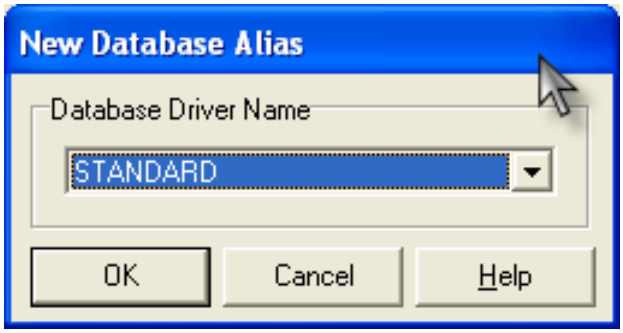

Рисунок 1.7

После щелчка на кнопке ОК в списке появится новый элемент, помеченный зеленым треугольником (см. рисунок 1.8).

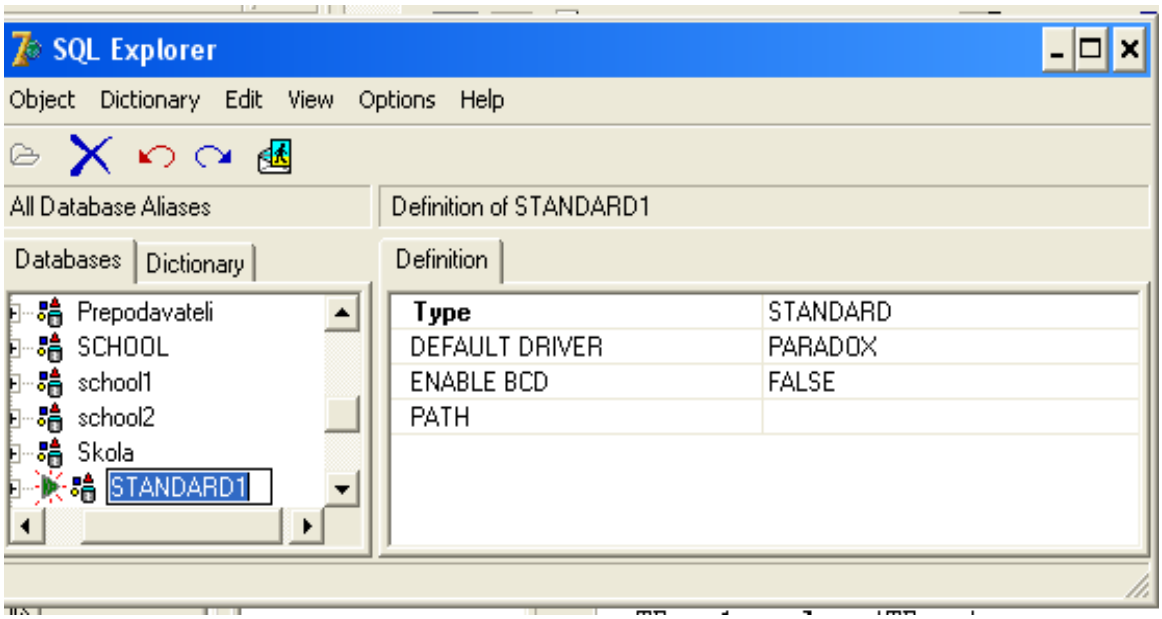

Рисунок 1.8

По умолчанию формируется имя базы данных Standard1, изменим его на Student. Убедимся, что в свойствах Default Driver (Драйвер по умолчанию) стоит значение Paradox. В свойстве Path укажем каталог, в котором хранится наша таблица.

Теперь зарегистрированную в системе BDE базу сохраним, выбрав для этого Apply в контекстном меню объекта Student. На вопрос о необходимости сохранения изменений нажать Yes. Теперь таблица доступны из среды BDE под именем Student.

Закройте Sql Explorer.

Контрольные вопросы.

1. Работа с утилитой Database Desktop.
2. Запуск Database Desktop.
3. Создание псевдонима.
4. Регистрация псевдонима.
5. Как заполнить данными таблицу?
6. Какие поля таблицы выбираем Paradox?
7. Редактирование таблицы.
8. Работа с индексами.

2 Лабораторная работа № 2. Работа с компонентами доступа и управления базы данных. Ввод и редактирование текста

Цель работы:

1) Изучить начальные этапы создания приложения для работы с базами данных в среде Delphi:

-работа с компонентами доступа к БД: TTable, TDataSource;

-работа с компонентами управления БД: TDBGrid, TDBNavigator.

2) Научиться вводить и редактировать текст в БД.

Методика создания приложения для работы с базой данных ничем не отличается от методики создания обычной программы: к форме добавляются необходимые компоненты, устанавливаются значения свойств компонент, разрабатываются необходимые процедуры обработки событий.

Приложение работы с базой данных должно содержать компоненты, обеспечивающие доступ к данным, возможность просмотра и редактирования содержимого полей. Компоненты доступа к данным находятся на вкладке Data Access и BDE палитры компонент, а компоненты отображения данных — на вкладке Data Controls.

Основные компоненты доступа и управления к базам данных.

Компоненты доступа к базам данных:

TTable  - обеспечивает взаимодействие с таблицей БД, т.е. компонента TTable указывает, откуда брать данные и какие поля будут составлять набор данных. Компонента TTable имеет следующие основные свойства:

-DatabaseName – база данных.

-TableName - имя таблицы.

-Active – активация таблицы (значение True активирует ее).

Свойство DatabaseName определяет базу данных, в которой находится таблица. Это свойство может содержать:

1) псевдоним (псевдоним);

2) путь для локальных БД;

3) путь и имя файла базы данных для Local InterBase;

4) локальный псевдоним, определенный через компоненту TDatabase.

Свойство TableName определяет имя таблицы базы данных.

TDataSource  - определяет связь между базой данных и компонентами управления данными, то есть компонента TDataSource является промежуточным звеном между компонентой Table1, соединенной с реальной БД и визуальными компонентами DBGrid1 и DBNavigator1, с помощью которых пользователь взаимодействует с таблицей.

В большинстве случаев, все, что нужно сделать с DataSource - это указать в свойстве DataSet соответствующий TTable. Затем, у визуальной компоненты вроде DBGrid или DBNavigator в свойстве DataSource указывается TDataSource, который используется в настоящее время.

Компоненты управления данными с палитры Data Contorls:

TDBGrid  - отображает содержимое таблицы БД в виде сетки, в котором столбцы соответствуют полям, а строки записям таблицы.

Компонента имеет следующие свойства:

Data Source – содержит ссылку на компоненту типа TDataSource, служащую источником данных;

Editor Mode – если содержит true, пользователь может редактировать ячейку после нажатия клавиши F2 или Enter. Игнорируется, если свойство Option включает значение goEditing или goAlwaysShowEditor;

Option – определяет вид и поведение компоненты;

dgEditing – разрешает изменение набора данных;

dgAlwaysShowEditor – автоматически переводит столбец в режим редактирования при его выделение;

dgTitles – показывает заголовки столбцов;

dgIndicator – показывает индикатор текущей строки в самом левом фиксированном столбце;

dgColumnResize – разрешает пользователю вручную изменять ширину столбцов;

dgColLines – показывает разделяющие вертикальные линии;

dgRowLines – показывает разделяющие горизонтальные линии;

dgTabs – разрешает переход от столбца к столбцу с помощью клавиши Tab;

dgRowSelect – разрешает выделение цветом всей выбранной строки;

dgAlwaysShowSelection – выделение текущей строки сохраняется, если компонента теряет фокус ввода;

dgConfirmDelete – удаление строки должно подтверждаться;

dgCancelOnExit – если пользователь вставил пустую строку и покинул ее, она не помещается в набор данных;

dgMultiSelect – разрешает множественный выбор строк.

TDBNavigator  - осуществляет перемещение и редактирование записей (вид и назначение кнопок указаны в пункте DBNavigator). С помощью свойства DataSource компонента связывается с нужным источником данных TDataSource – это все, что необходимо для его нормальной работы. Свойство ConfirmDelete управляет отображением диалогового окна с просьбой подтвердить удаление записи (значение True этого свойства выводит окно).

Задания А, Б.

Выбрав вариант согласно последней цифре учебного шифра, разместить по две таблицы на форму в программной среде Delphi (см. приложение А), разработать формы с применением компонент доступа и управления к базам данных. Студентам все задания выполнять по двум вариантам (А, Б).

Методические указания.

Создание приложения Студенты.

1) Запустить Delphi.

2) В свойстве Caption изменить имя формы на Студенты.

3) Установить на форму компоненту TTable.

Определить следующие свойства компоненты Table1.

1. Определить псевдоним - выбрать в свойстве DatabaseName инспектора объектов псевдоним «Student».

2. Задать имя таблицы - выбрать в свойстве TableName таблицу Student.

3. Активизировать таблицу - установить в свойстве Active значение true (это будет возможно после выполнения пункта.

Разместить на форму компоненту DataSource с закладки Data Access и в свойстве DataSet инспектора объектов выбрать компоненту Table1.

Расположить на форме компоненту DBGrid с закладки Data Controls и в свойстве DataSource выбрать DataSource1.

Если внесены несколько записей, то форма Студенты примет вид (см. рисунок 2.1).

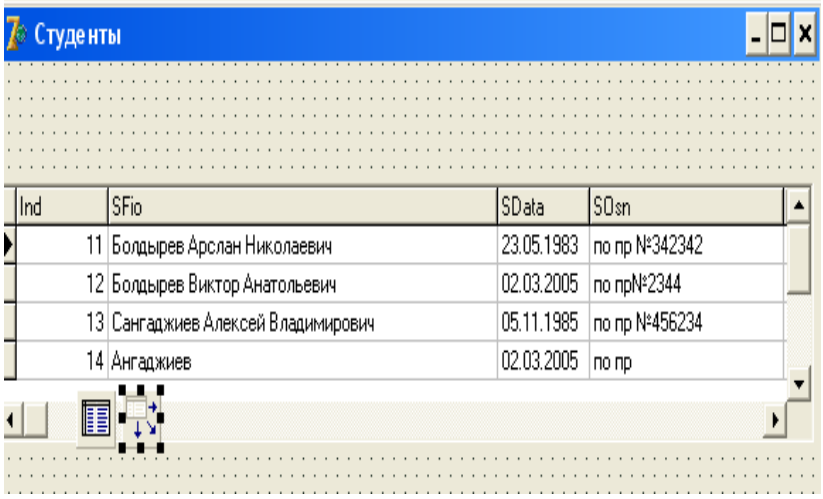

Рисунок 2.1

Редактор полей: для управления отображением данных таблицы используют специальный редактор полей - Editor Field.

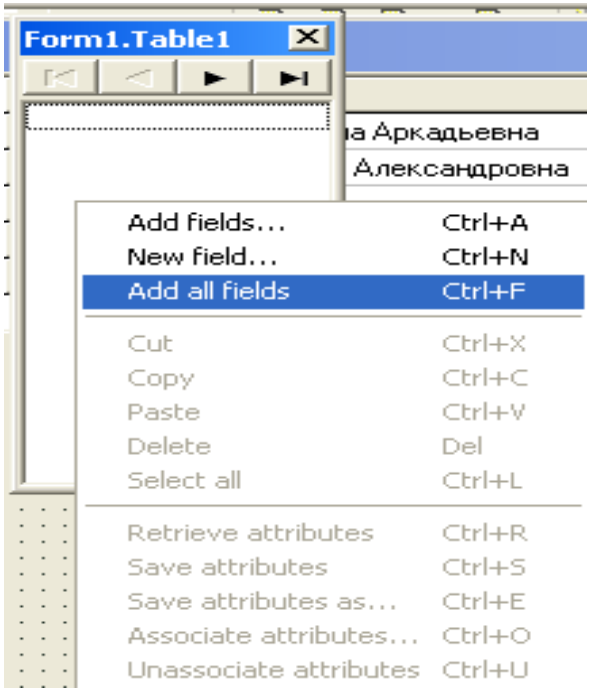

Рисунок 2.2

Для вызова Editor Field следует:

1. Дважды щелкнуть по Table1.

2. Для открывшегося окна вызвать контекстное меню и выбрать пункт Add All Field, если необходимо добавить все поля таблицы.

3. Add Field для выбора отдельного поля (см. рисунок 2.2).

Редактор полей имеет следующие свойства:

DisplayLabel – задает имя полю;

DisplayWidth – определяет количество символов, которое будет выводится в поле (см. рисунок 2.3).

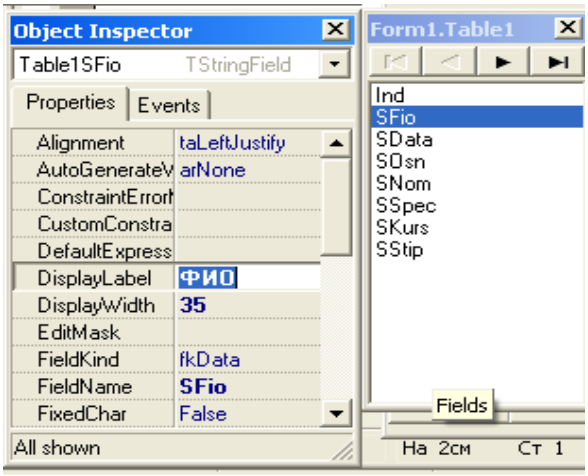

Рисунок 2.3

Определим свойства для полей таблицы Student.db.

1. Выбрать в окне редактора полей таблицы поле SFio и в свойстве DisplayLabel инспектора объектов изменить SFio на ФИО.
2. Выбрать свойство DisplayWidth и заменить размер на 35.
3. Так же поменять свойства других полей таблицы.
4. Для полей логического типа в свойстве DisplayValues можно написать варианты для значений True и False.
5. В поле SSpec в этом свойстве написать «Математика;Физика» (без пробела, разделяя «;»). Получиться как показано на рисунке.
6. Если возникнет необходимость можно скрыть любое поле, выбрав его и в свойстве Visible инспектора объектов, установив значение false.

После выполненных действий сетка DBGrid1 будет иметь вид (см. рисунок 2.4).

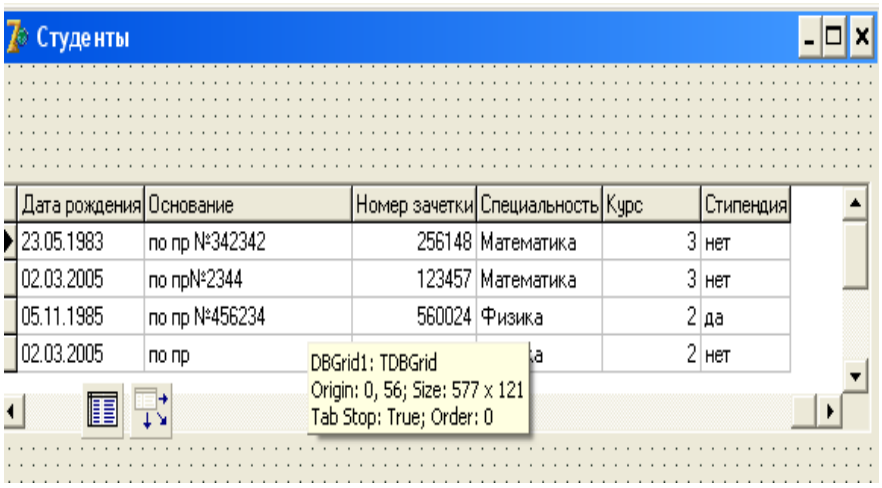

Рисунок 2.4

Ввод данных: Компоненты для организации доступа к таблицам БД позволяют выполнять всевозможные операции с наборами данных: добавлять или удалять записи, перемещаться по ним. При этом следует иметь в виду, что в любой момент времени доступна для выполнения конкретных действий только одна запись, называемая текущей. В этой лабораторной работе рассматриваются наиболее часто используемые методы компоненты Table.

Основные методы для организации доступа компоненты Table:

Append – добавить новую запись в конец таблицы.

Delete – удалить текущую строку.

Edit – перейти в режим редактирования. После этого можно изменять значения полей.

Insert – вставить новую строку в таблицу.

Post – принять все изменения.

Refresh – обновить информацию о данных.

UpdateRecord – обновить текущую запись.

Откроем созданное приложение.

Разместим на форме три компоненты SpeedButton из палитры Additional. Одна из кнопок будет добавлять запись, другая – изменять данные в записи, третья – удалять. Назовем их соответственно (см. рисунок 2.5).

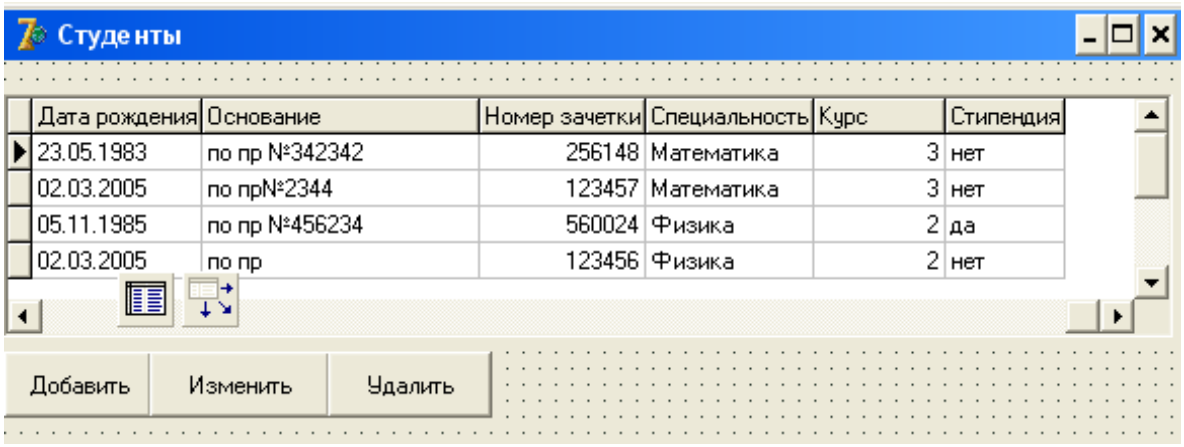

Рисунок 2.5

Создадим новую форму, которая будет вызываться нажатием кнопки «Добавить». На форме расположены 4 компоненты Edit, компонента DateTimePicker с закладки Win32, компонент CheckBox и компонента RadioGroup (см. рисунок 2.6).

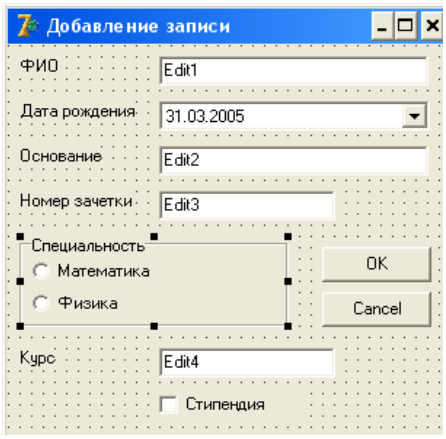

Рисунок 2.6

Текст процедуры для события OnClick кнопки «Добавить» на форме Студенты:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
    Form2.ShowModal; //открывает форму «Добавление записи»
  end;
```

Текст процедуры для события OnClick кнопки «ОК» на форме Добавление записи:

```
procedure TForm2.Button1Click(Sender: TObject);
```

```
begin
    Form1.Table1.Insert;
    Form1.Table1.FieldByName('SFio').Text:=Edit1.Text;
    Form1.Table1.FieldByName('SOsn').Text:=Edit2.Text;
    Form1.Table1.FieldByName('SNom').Text:=Edit3.Text;
    Form1.Table1.FieldByName('SKurs').Text:=Edit4.Text;
    Form1.Table1.FieldByName('SData').AsDateTime:=DateTimePicker1.Date;
    if CheckBox1.Checked then
       Form1.Table1.FieldByName('SStip').Text:='да'
         else
           Form1.Table1.FieldByName('SStip').Text:='нет';
               //при нажатии на флажок полю SStip (Стипендия) передается
           //значение True, в противном случае вводится передается
                //значение False
    case RadioGroup1.ItemIndex of
   0: Form1.Table1.FieldByName('SSpec').Text:='Математика';
   1: Form1.Table1.FieldByName('SSpec').Text:='Физика';
   end;
   if form1.Table1.Modified
      then form1.Table1.Post;
   close;
```

Комментарий: в строке Form1.Table1.Insert вызывается метод, который допускает вставку новой строки в таблицу, которая находится на форме «Студенты». Без вызова этого метода дальнейшая работа по вставке записи в таблицу невозможна.

Запись Form1.Table1.FieldByName('SFio').Text:=Edit1.Text означает, что текст, который находится в Edit1 по нажатии кнопки будет перенесен в таблицу на форме «Студенты» в новую запись в текстовое поле ФИО. Остальные записи в процедуре работают аналогичным образом. Запись if form1.Table1.Modified then form1.Table1.Post сохраняет изменения в таблице. Close – закрывает форму «Добавление записи».

По нажатии кнопки Cancel осуществляется выход. То же и на форме «Редактирование записи».

Текст процедуры для события OnClick при нажатии клавиши «Удалить» на форме Студенты:

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
      Table1.Delete              //удаляет текущую запись в таблице
   end;
```

Редактирование данных.

Компоненты, отражающие информацию, делятся на две категории – те, которые не связаны с таблицами БД, и компоненты, связанные с таблицами и

обменивающиеся с ними информацией. В первую категорию входят обычные компоненты Delphi. Компоненты второй категории расположены на странице Data Controls. Почти каждая из них имеет аналог среди обычных компонент; основные отличия заключаются в том, что они могут работать с данными, хранящимися в БД. К этой группе относится компонента DBEdit, которая используется для ввода текстовой однострочной информации.

Чтобы компонента DBEdit видела данные из поля таблицы, следует указать в свойствах:

DataSource – источник данных; DataField – поле для редактирования.

Эта компонента с заданными уже свойствами может появиться автоматически при перетаскивании имени поля из окна редактора полей.

Создадим новую форму: Редактирование записи (см. рисунок 2.7).

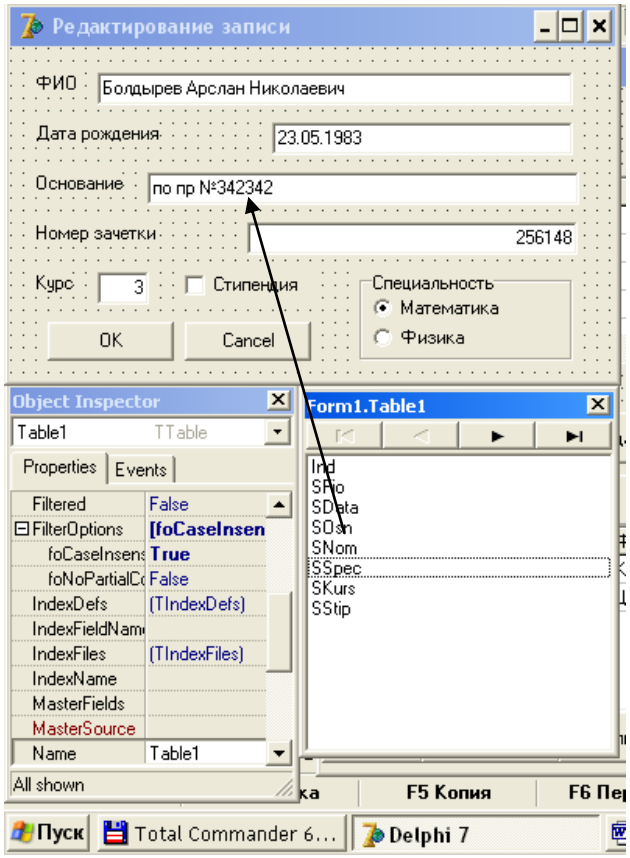

Рисунок 2.7

В случае перетаскивания поля логического типа, на форме автоматически устанавливается компонента DBCheckBox, что не всегда удобно.

Установим на форму компоненту DBRadioGroup с закладки Data Control. В свойстве DataSource укажем DataSource1.В свойстве DataField укажем SSpec.

Текст процедуры для события OnClick при нажатии клавиши «Изменить» на форму Студенты:

```
begin
Form3.ShowModal //вызов Form3
```

```
end;
```

Текст процедуры для события OnClick при нажатии клавиши «Сохранить» на форму Редактирование:

```
begin
if form1.Table1.Modified
        then form1.Table1.Post;        //все изменения в таблице сохраняются
close;
end;
```

Пользователь имеет возможность редактировать записи в таблице напрямую. Чтобы это предотвратить используется свойство компоненты DBGrid dgEditing. Нужно выделить DBGrid1 и в свойстве Options► dgEditing инспектора объектов поставить false.

Контрольные вопросы.

1. Назначение компонент доступа к БД.
2. Назначение компонент управления БД.
3. Объяснить текст процедуры для события OnClick при нажатии клавиши «Изменить».
4. Объяснить текст процедуры для события OnClick кнопки «Добавить».
5. Объяснить текст процедуры для события OnClick при нажатии клавиши «Сохранить».
6. Объяснить текст процедуры для события OnClick при нажатии клавиши «Удалить».
7. Основные методы для организации доступа компоненты Table.
8. Какие свойства имеет редактор полей ?

3 Лабораторная работа № 3. Сортировка записей в базе данных. Сортировка и поиск записей в базе данных

Цели работы:

1) Ознакомиться с сортировкой записей в базе данных.
2) Поиск полей с помощью методов Locate и Lookup.

Общие сведения.

Сортировка.

Порядок расположения записей в таблице БД может быть неопределенным. По умолчанию записи не отсортированы или сортируются, например, для таблиц Paradox по ключевым полям, а для таблиц dBase в порядке их поступления в файл таблицы.

С отсортированными записями набора данных работать более удобно. Сортировка заключается в упорядочивании записей по определенному полю в порядке возрастания или убывания содержащихся в нем записей.

Сортировка набора данных TTable выполняется автоматически по текущему индексу. При смене индекса происходит переупорядочивание записей. Таким образом, возможна по полям, для которых создан индекс. Для сортировки по нескольким полям нужно создать индекс, включающий эти поля.

Задать индекс, по которому выполняется сортировка записей, можно с помощью свойств:

1) IndexName – указывается имя индекса, установленное при его создании;

2) IndexFieldName – указываются имена полей, образующие соответствующий индекс.

Поиск.

Метод Locate ищет первую запись, удовлетворяющую критерию поиска, и если такая запись найдена, делает ее текущей. В этом случае в качестве результата возвращается значение True. Если запись не найдена, возвращается значение False и курсор не меняет своего положения.

function Locate (const KeyFields: String; const KeyValues: Variant;
Options: TLocateOptions): Boolean;

Список полей, по которым ведется поиск, задается в параметре KeyFields, поля разделяются точкой с запятой. Параметр KeyValues типа Variant указывает значение полей для поиска. Если поиск ведется по одному полю, то параметр содержит одно значение, соответствующие типу поля, заданного для поиска.

Параметр Options позволяет задать значение, которое обычно используется при поиске строк. Этот параметр принадлежит к множественному типу TLocateOptions и принимает комбинации следующих значений:

LoCaseInsensitive –регистр букв не учитывается;

LoPartialKey – допускается частичное совпадение.

Задания А, Б.

Выбрав вариант согласно последней цифре учебного шифра, разработать формы сортировкой записей и с использованием методов Locate и Lookup в программной среде Delphi (см. приложение А). Студентам все задания выполнять по двум вариантам (А, Б).

Методические указания.

Сортировка.

1. Открыть приложение «Студенты».

2. Добавить на форму компоненты ComboBox и Button (см. рисунок 3.1).

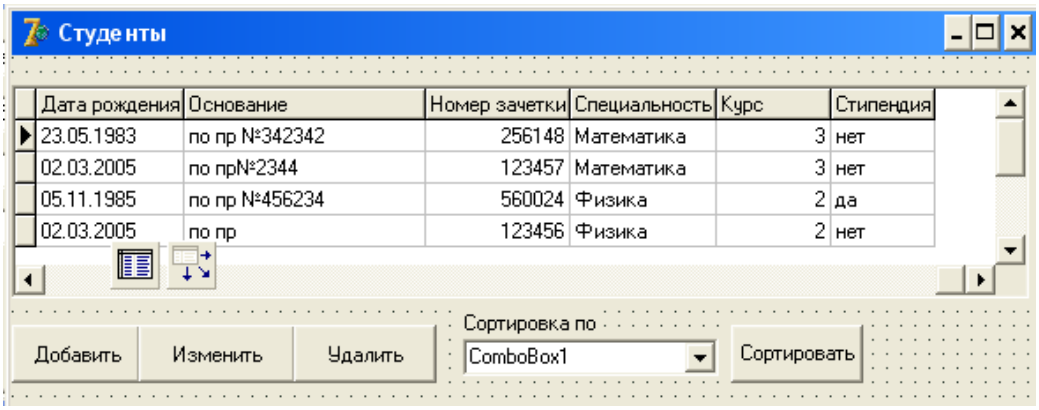

Рисунок 3.1

В свойстве Items компоненты ComboBox запишем параметры сортировки: Фамилия, Специальность, Курс, Дата рождения, Номер зачетки.

Условия сортировки задаются вторичными индексами. То есть сортировка по фамилии происходит по вторичному ключу IDFio так как в него первым входит поле SFio. Для того, чтобы сортировка проходила по выбранным параметрам необходимо вхождение соответствующих полей в разные вторичные ключи.

Текст процедуры для события OnClick при нажатии кнопки «Сортировка» на форме Студенты:

```
begin
  Case ComboBox1.ItemIndex of
    0: Table1.IndexFieldNames:='SFio'; //при выборе строки «Фамилия»
                    //сортировка идет по вторичному индексу IDFio
    1: Table1.IndexFieldNames:='SSpec';
    2: Table1.IndexFieldNames:='SKurs';
    3: Table1.IndexFieldNames:='SData';
    4: Table1.IndexFieldNames:='SNom';
  end;
end;
```

Замечание:

Во вторичный индекс IDFio входят поля: SFio, SKurs, SSpec. То есть при совпадении фамилии сортировка идет уже по курсу и т.д. (см. рисунок 3.2).

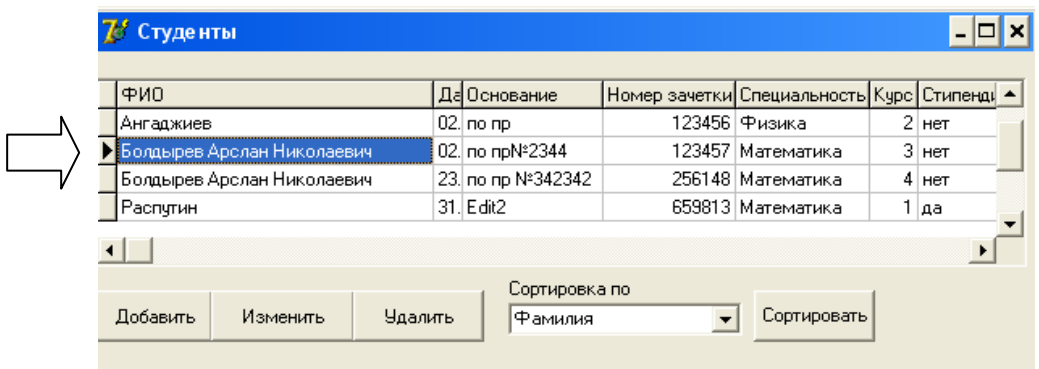

Рисунок 3.2

Поиск информации.

На форму добавить компоненту Edit.

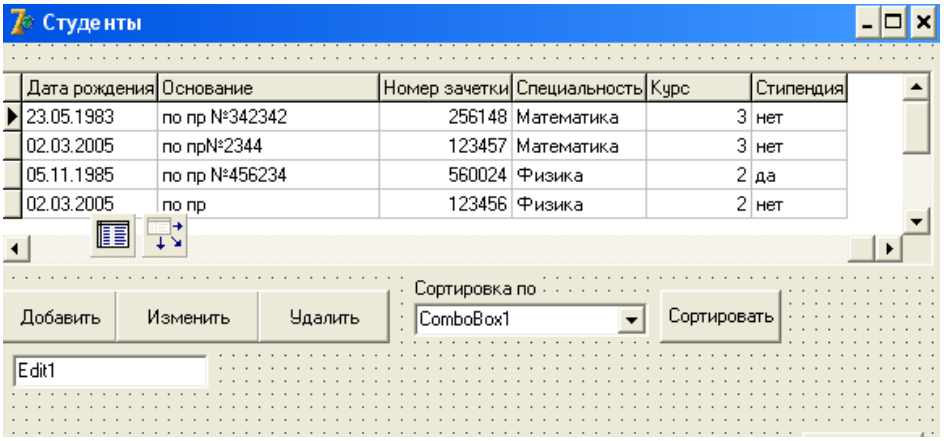

Рисунок 3.3

Текст процедуры для события OnChange компоненты Edit на форме Студенты:

```
begin
   table1.Locate('SFio',Edit1.Text,[loPartialKey]);
end;
```

Поиск записи по фамилии организован. Регистр букв не учитывается (см. рисунок 3.3).

Метод Lookup находит запись, удовлетворяющую условию поиска, но не делает ее текущей, а возвращает значения некоторых ее полей. Независимо от результата поиска записи указатель текущей записи в НД не изменяется. В отличие от метода Locate, метод Lookup осуществляет поиск только на точное соответствие критерию поиска значения поля поиска записи.

function Lookup (const KeyFields: String; const KeyValues: Variant;
const ResultFields: String): Variant;

В параметре ResultFields перечисляются поля, значения которых требуется получить в случае успешного поиска. Тип результата – Variant или вариантный массив.

Добавить на главную форму новую кнопку «Поиск».

Открыть новую форму и ввести компоненты как показано на рисунке. Эта форма вызывается нажатием кнопки поиска на главной форме.

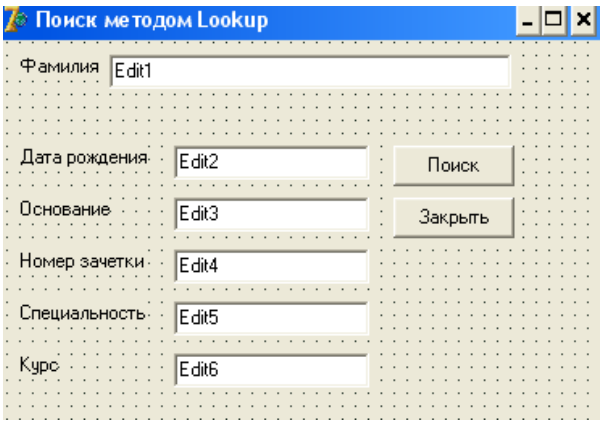

Рисунок 3.4

Поиск будет происходить по фамилии введенной в компоненте Edit1 после нажатия *кнопки на форме «Поиск методом Lookup».*

```
procedure TForm5.Button1Click(Sender: TObject);
var  LookupResult: Variant;
begin
LookupResult:=Form1.Table1.Lookup  ('SFio',  Edit1.Text,  'SData;  SOsn;
SNom; SSpec; SKurs');   //ищем поля  'Дата рождения'
//’Основание’, ’Номер зачетки’, ’Специальность’, ’Курс’
            if VarIsArray (LookupResult) then
                  begin
                  Edit2.Text:=LookupResult[0]; //записывает значения
                 Edit3.Text:=LookupResult[1]; // в искомых полях в
                  Edit4.Text:=LookupResult[2]; //соответствующие
                 Edit5.Text:=LookupResult[3]; //компоненты
                 if Edit5.Text='False' then
                        Edit5.Text:='Физика'   //поиск полей логического типа
                              else Edit5.Text:='Математика';
                 Edit6.Text:=LookupResult[4];
                 end;
end;
```

Контрольные вопросы.

1. Текст программы кнопки «Поиск методом Lookup».
2. Текст процедуры для события OnChange компоненты Edit.
3. Как выполняется поиск информации?
4. Что такое сортировка информации?
5. Методы поиска.
6. Что такое индексы?
7. Метод Locate.
8. Метод Lookup.

4 Лабораторная работа № 4. Фильтрация записей

Цели работы:

1) Ознакомиться с фильтрацией записей.

2) Разработать форму с поиском информации.

Общие сведения.

Фильтрация – выбор из набора данных только тех записей, которые удовлетворяют конкретным условиям.

Например, можно указать отображение только записей, в которых поле «Фамилия» содержит значение «Иванов». Применение фильтра к набору данных определяется свойством Filtered логического типа. Значение True определяет применение в качестве фильтра выражения, указанного в свойстве Filter:

Поле [Оператор сравнения] ‘Значение’

Например, если отобразить все записи, в которых поле «Фамилия» равно значению «Сидоров», то нужно указать:

Table1.Filter:=’Фамилия=’’Сидоров’’;

Задания А, Б.

Выбрав вариант согласно последней цифре учебного шифра, разработать формы с поиском информации и фильтрацией записей в среде Delphi (см. приложение А). Студентам все задания выполнять по двум вариантам (А, Б).

Методические указания.

Фильтрация записей.

Открыть приложение «Студенты».

Добавить на форму компоненту TEdit.

Текст процедуры для события OnChange:

```
begin
  Table1.Filtered:=true;          //включение фильтрации
  Table1.Filter:='SNom = ‘+Edit2.Text;
//задает критерий фильтрации
end;
```

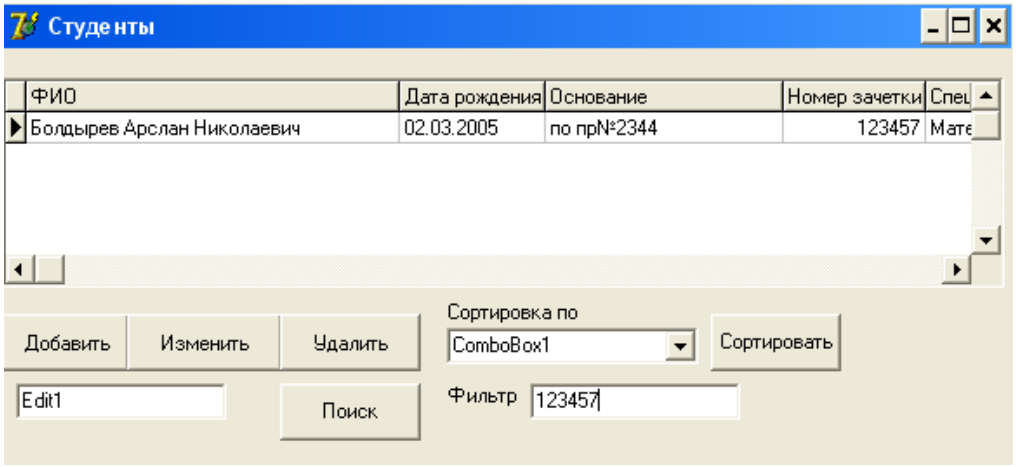

Рисунок 4.1

Этот способ фильтрации пригоден только для числовых полей (см. рисунок 4.1).

При применении фильтра можно указать свойства:

1) foCaseInsensitive – нечувствительность к регистру букв;

2) foNoPartialCompare – поиск на точное соответствие.

Для фильтрации текстовых полей, например по полю «Фамилия» необходимо изменить текст процедуры.

```
procedure TForm1.Edit2Change(Sender: TObject);
begin
Table1.Filtered:=true;
Table1.Filter:='SFio='+#39+Edit2.Text+'*'+#39;
end;
```

В этом случае фильтрация проходит по текстовому полю. Знак «#39» означает знак апострофа, так как ввод фамилии при использовании фильтра происходит в апострофах. А символ «*» означает любые символы, то есть при вводе только одной буквы на экране появятся все фамилии начинающиеся на букву (см. рисунок 4.2).

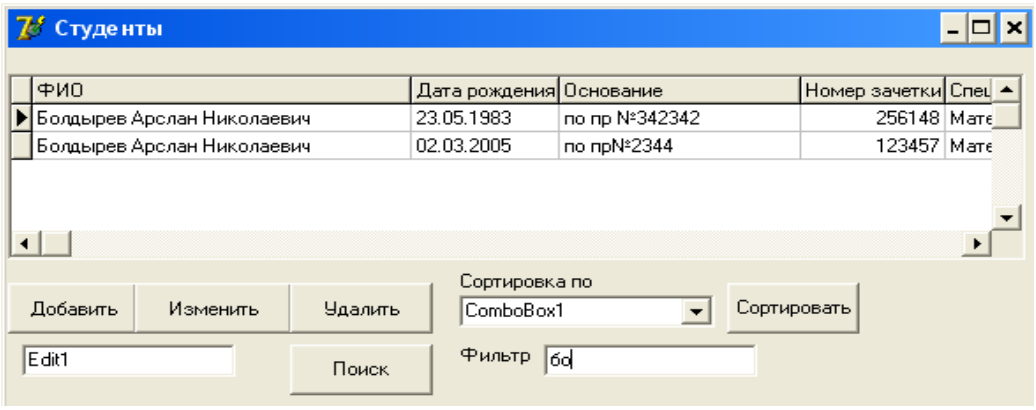

Рисунок 4.2

Контрольные вопросы.

1. Что означает знак «#39»?

2. Какой текст процедуры необходим для фильтрации текстовых полей?
3. Текст процедуры для события OnChange.
4. Назначение свойства foCaseInsensitive.
5. Назначение свойства foNoPartialCompare.
6. Каким свойством определяется применение фильтра к набору данных?
7. Какие команды необходимы для фильтрации текстовых полей?
8. Как задается критерий фильтрации?

5 Лабораторная работа № 5. Установка связи между таблицами

Цели работы:

1) Изучить возможность связи между таблицами.
2) Установка связи между таблицами.

Общие сведения.

Организация данных и связь между таблицами.

При рассмотрении ключей и индексов мы много раз упоминали про организацию связи между различными таблицами в рамках БД. И хотя теоретически БД может состоять из одной единственной таблицы, содержащей лишь имена и даты рождения сотрудников, на практике СУБД для подобных целей не применяются: такую информацию можно хранить в виде электронной таблицы, а то и вообще в обычном текстовом файле. Поэтому рассмотрим более типичную ситуацию, когда БД состоит из нескольких взаимосвязанных таблиц.

Организация отношений между таблицами называется связыванием, или соединением таблиц и является одним из ключевых моментов разработки БД. Связи задаются как при создании таблиц (назначение ключей и индексов), так и в процессе разработки и даже работы приложения, для чего используются различные средства, предоставляемые СУБД.

Для связывания таблиц используют поля, называемые полями связи. Такие поля должны быть индексированными. При связывании таблиц одна из них назначается главной (Master), а другая - подчиненной (Slave, Detail). При этом индекс, отвечающий за соединение в подчиненной таблице, называют внешним ключом. Что касается главной таблицы, то ее поле, участвующее в связывании, должно быть главным индексом (разумеется, если в применяемой СУБД есть такое понятие). Типы полей, участвующих в связывании, должны совпадать, например, оба они должны быть целочисленными.

Допустим, у нас имеется БД, состоящая из 2 таблиц, в одной из которых хранится список клиентов (назовем ее CUSTOMER), а в другой - накладные, выписанные этим клиентам (BILL). В таком случае ключом первой таблицы должен быть идентификатор клиента (скажем, поле автоинкрементного типа CUST_ID), а во второй должно иметься, по возможности, индексированное поле, содержащее информацию о том, какому клиенту выдана данная

накладная (поле BILL_CUST). При этом ключевым полем второй таблицы будет номер накладной (поле BILL_ID). В получившейся связи связанными полями будут CUST_ID и BILL_CUST, причем последнее будет являться внешним ключом для таблицы BILL (см. рисунок 5.1).

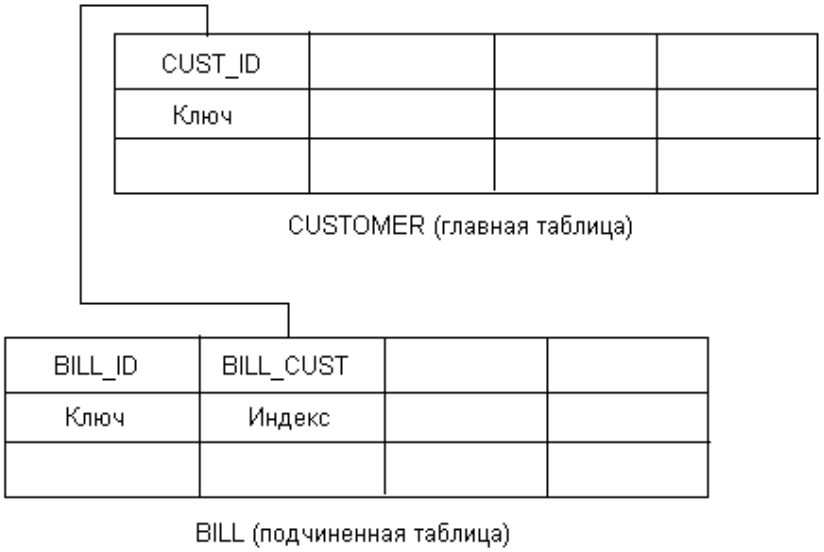

Рисунок 5.1Связи между таблицами

Связать между собой можно 2 таблицы или несколько таблиц сразу. При этом одна из таблиц обычно выступает главной, а другие - подчиненными. Такую связь обычно называют «главный-подчиненный». Вообще же, в зависимости от заданных условий, эта связь может получиться одного из 4 видов:

1) «Один к одному», когда каждой записи в главной таблице соответствует 1 запись в подчиненной;

2) «Один ко многим», когда каждой записи в главной таблице соответствует 0 или больше записей в подчиненной;

3) «Много к одному», когда нескольким записям в главной таблице соответствует 1 в подчиненной;

4) «Много ко многим», когда произвольному числу записей в главной таблице соответствует такое же неопределенное число записей в подчиненной.

Чаще всего встречается 2-й вариант, включая рассмотренный пример, где каждой записи о клиенте может соответствовать произвольное число записей в таблице накладных, включая полное их отсутствие, если клиент еще ничего не купил. После того, как связь между таблицами установлена, выбор определенной записи в главной таблице автоматически приведет к выборке в подчиненной (см. рисунок 5.2).

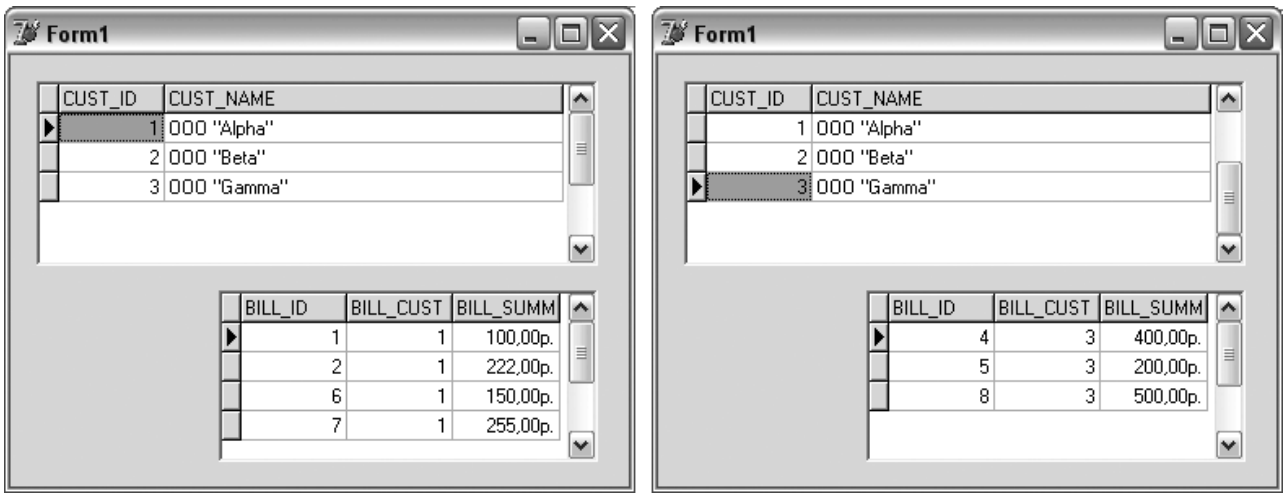

Рисунок 5.2Связь «один ко многим»

Хотя технология связывания таблиц средствами IDE будет рассматриваться позже, забегая вперед, отметим, что в данном случае для таблицы BILL была выбрана Master-таблица CUSTOMER и была установлена связь через объединение полей типа BILL_CUST ' CUST_ID**.**

Задание А, Б.

Выбрав вариант согласно последней цифре учебного шифра, установить связи между таблицами в среде Delphi (см. приложение А), Студентам все задания выполнять по двум вариантам (А, Б).

Методические указания.

Для демонстрации связи между таблицами необходимо создать еще одну таблицу. Создайте таблицу успеваемости студентов. В нее войдут поля: учебный год, сессия (зима или лето), предмет, ФИО преподавателя, дата аттестации по предмету, дата сдачи, оценка.

Эта таблица будет дочерней для таблицы Студенты. В таблице Успеваемость надо ввести дополнительно числовое поле и определить его вторичным ключом.

Разместите таблицу успеваемости на форме Студенты.

В свойстве компоненты Table2 Master Source написать Data Source1. Это означает, что вторая таблица станет дочерней для первой. Двойным щелчком по свойству Master Fields вызовем окно Field Link Designer (Дизайнер поля связи) (см. рисунок 5.3).

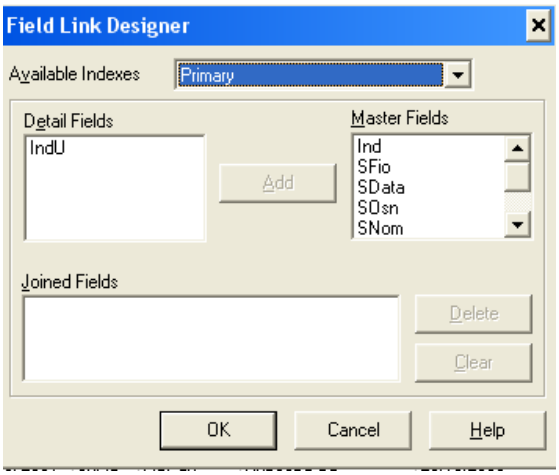

Рисунок 5.3

Выбрать в списке Available Index (Доступные индексы) из окна Field Link Designer индекс IDGod (это вторичный индекс второй таблицы).

В левом списке Detail Field выделить IdU, а в правом списке Master Field (Основа) выделить Ind. Нажать на кнопку Add (Добавить) и закрыть окно (см. рисунок 5.4).

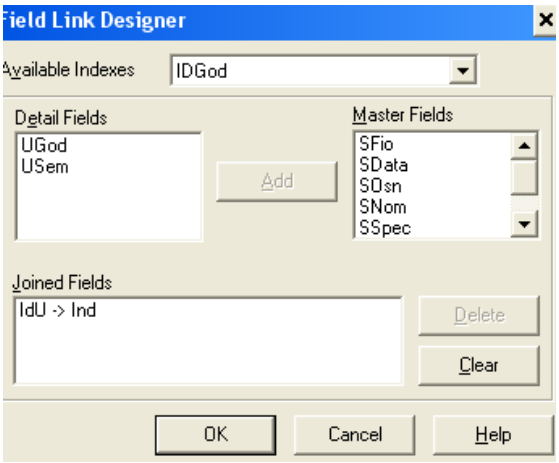

Рисунок 5.4

Таким образом, между таблицами установилась связь. Она называется связь один ко многим.

К дочерней таблице добавить кнопки для ввода и редактирования данных также как и для таблицы Студенты (см. рисунок 5.5).

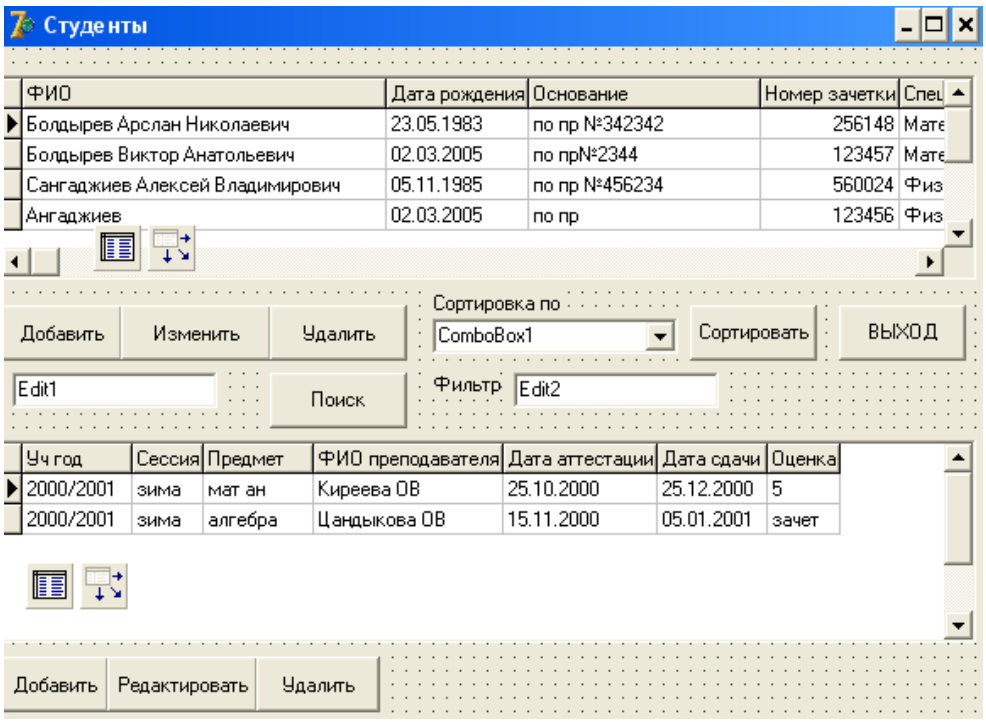

Рисунок 5.5

Контрольные вопросы.

1. Компонента Master Source.
2. Рассказать о видах связей между таблицами.
3. Рассказать о связи "Один к одному".
4. Назначение связи"Один ко многим".
5. Связь"Много к одному.
6. Связь "Много ко многим".
7. Параметры списка Available Index.
8. Назначение дизайнер поля связи.

6 Лабораторная работа № 6. Создание отчетов

Цели работы:

1) Ознакомиться с возможностью создания отчета.

Общие сведения.

Создание отчетов.

Отчет — это печатный документ, содержащий записи БД. В Delphi для создания отчетов служит генератор отчетов QuickReport, содержащий обширный набор компонентов.

Компоненты, предназначенные для создания отчетов, находятся на закладке QReport палитры компонентов.

Главным элементом отчета является компонент-отчет QuickRep, представляющий собой основу, на которой размещаются другие

компоненты. Компонента QuickRep обычно размещается на отдельной форме, предназначенной для создания отчета.

Свойства компоненты QuickRep:

1) Bands – здесь указываются компоненты размещаемые в QuickRep.

2) DataSet – здесь указывается набор данных из которой отчет будет брать данные.

3) Frame – здесь указывается параметры рамки.

4) Options – здесь доступны три параметра. Если FirstPageHeader равно true, то заголовок печатается только на первой странице отчета. Если LastPageFooter равен true, то нижний колонтитул печатается только на последней странице отчета. Если установить свойство Compression в true, то отчет будет сохраняться в сжатом виде.

5) ReportTitle – здесь находится заголовок печатаемого документа.

6) SnapToGrid – нужно ли выравнивать компоненты по установленной сетке.

7) Zoom – масштаб отображения данных.

Настройку параметров отчета можно выполнить с помощью окна Report Settings, вызываемый двойным щелчком мыши по компоненте QuckRep. Предпочтительно пользоваться именно этим окном, так как здесь всегда можно просмотреть будущий результат (см. рисунок 6.1).

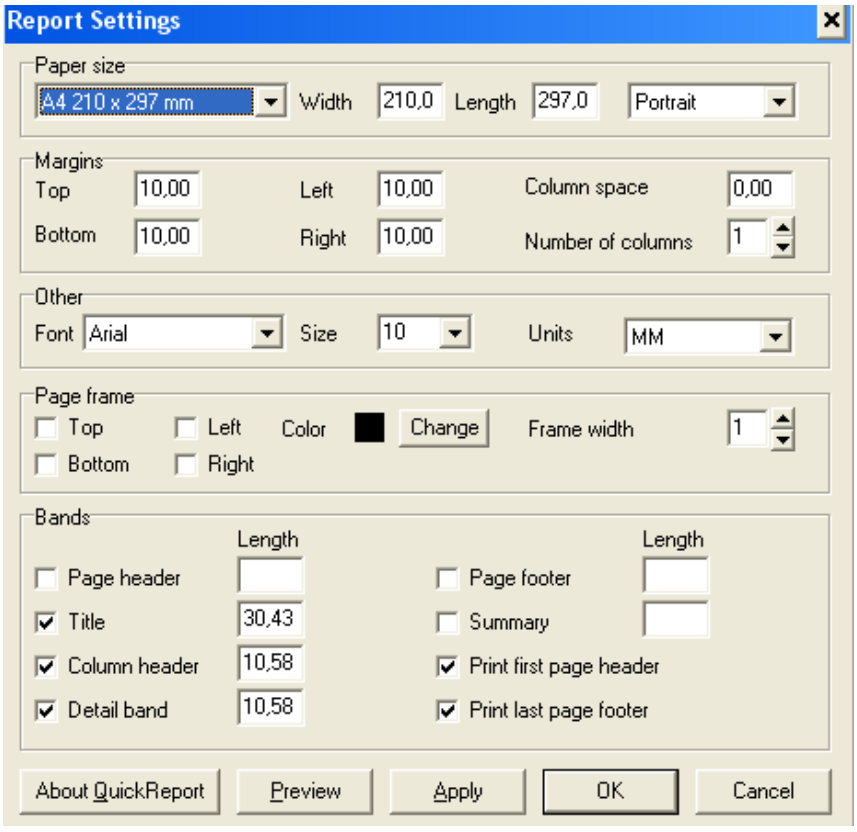

Рисунок 6.1

Методические указания.

1. Открыть приложение «Студенты».

2. Добавить на главную форму кнопку «Создание отчета».

3. Создать новую форму «Отчет», которая будет вызываться нажатием на кнопку «Создание отчета».

4. На форму установить компоненту QuickRep с закладки QReport. Выделить этот компонент и в объектном инспекторе включить параметры HasTitle и HasDetail свойства Bands (см. рисунок 6.2).

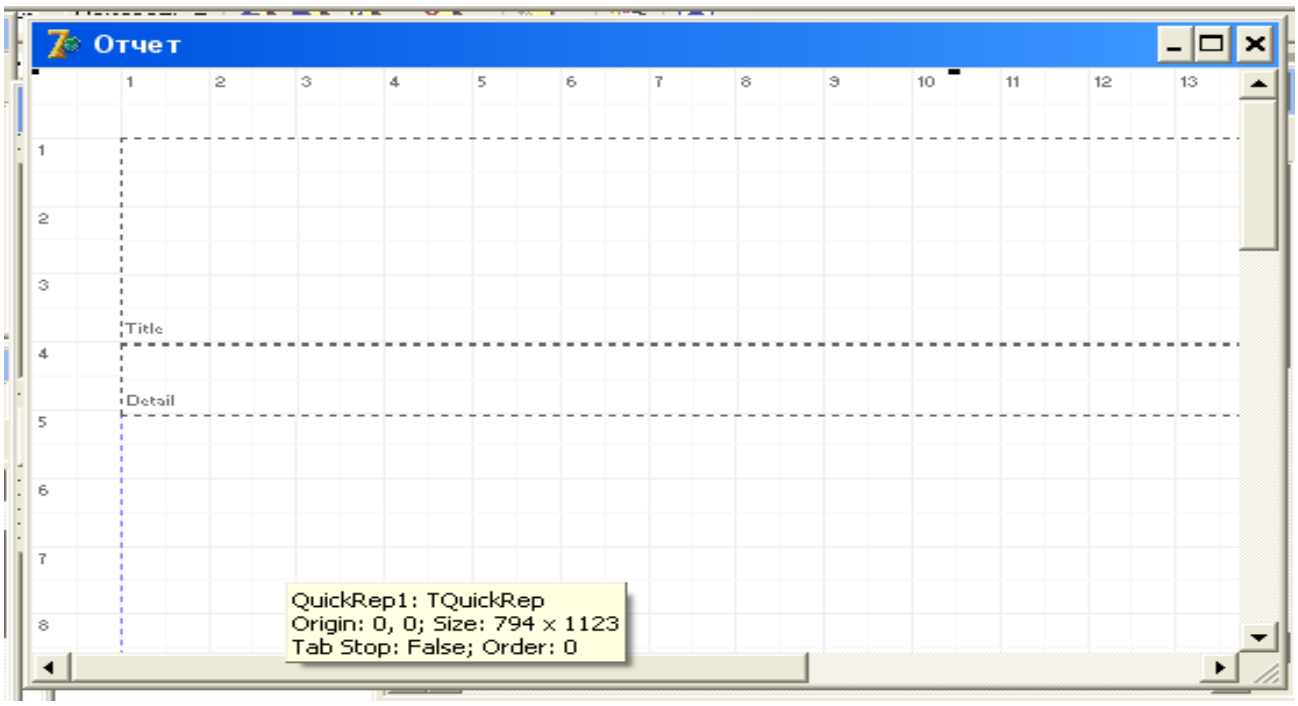

Рисунок 6.2

5. Расположим компоненты в секциях QuickRep1, которые будут отображать нужную информацию отчета.

На закладке QReport палитры компонент доступны следующие компоненты, которые можно расположить в этих разделах:

-QRLabel – надпись. Эта компонента похожа на стандартную компоненту TLabel и просто отображает нужные данные.

-QRDBText – данные. Эта компонента тоже похожа на TLabel, только она предназначена для отображения значения какого либо поля из базы данных.

-QRSysData – системная информация. Это опять копия TLabel только с возможностью отображать системную информацию – дату, время, номер страницы, номер строки в таблицы, общее количество страниц и т.д.

-QRImage – картинка. Компонента схожа с TImage.

6. Увеличить область заголовка Title. В верхний угол поместите одну компоненту QRSysData. Выделить ее и в свойстве Data выбрать значение qrsDateTime. Теперь эта компонента будет отображать в правом, верхнем углу дату распечатки документа.

7. В центре области Tittle установить компоненту QRLabel, увеличь шрифт в свойстве Font и написать в свойстве Caption текст «Студенты».

8. Расположить в области Tittle компоненты QRLabel и дать им заголовки: ФИО, Дата рождения, Номер зачетки, Специальность, Курс.

9. Перейти к области Detail. Под заголовками поставить пять компонент QRDBText. Установить в свойстве DataSet компонент QRDBText набор данных - Form1.Table1, а в свойстве DataField для QRDBText1 указать SFio.

10. У всех остальных компонент QRDBText указать соответствующие имена полей (см. рисунок 6.3).

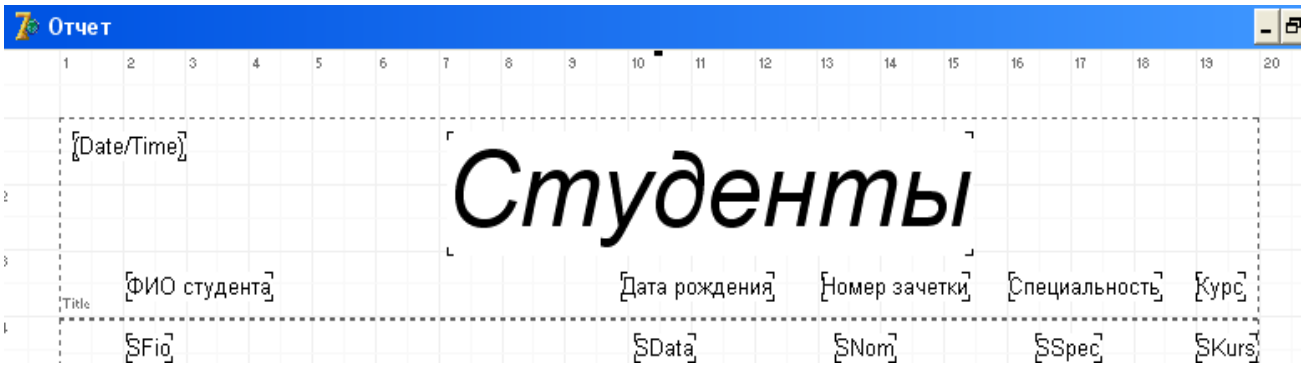

Рисунок 6.3

11. Перейти в главный модуль и *по нажатию кнопки «Печать»* написать следующий код.

```
procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
Form4.QuickRep1.Preview; //вызывается метод Preview
//компонента QuickRep. Этот метод показывает окно
//предварительного просмотра созданного  документа.
end;
```

12. Запустить программу, выделить какую-нибудь строку и нажать кнопку печати.

Откроется окно предварительного просмотра (см. рисунок 6.4).

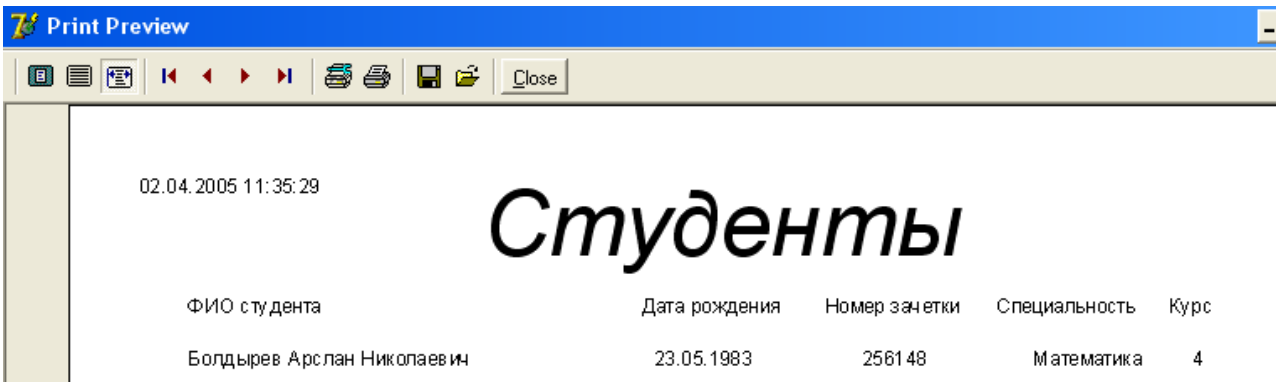

Рисунок 6.4

13. Выделить компоненту QuickRep1 и в свойстве DataSet указать таблицу Form1.Table1. Если сделать это, то компонента QuickRep1

автоматически будет перебирать все записи из этой таблицы и использовать их в компонентах, которые стоят в блоке DetailBand1.

14. После этого в отчете появятся все записи таблицы (см. рисунок 6.5).

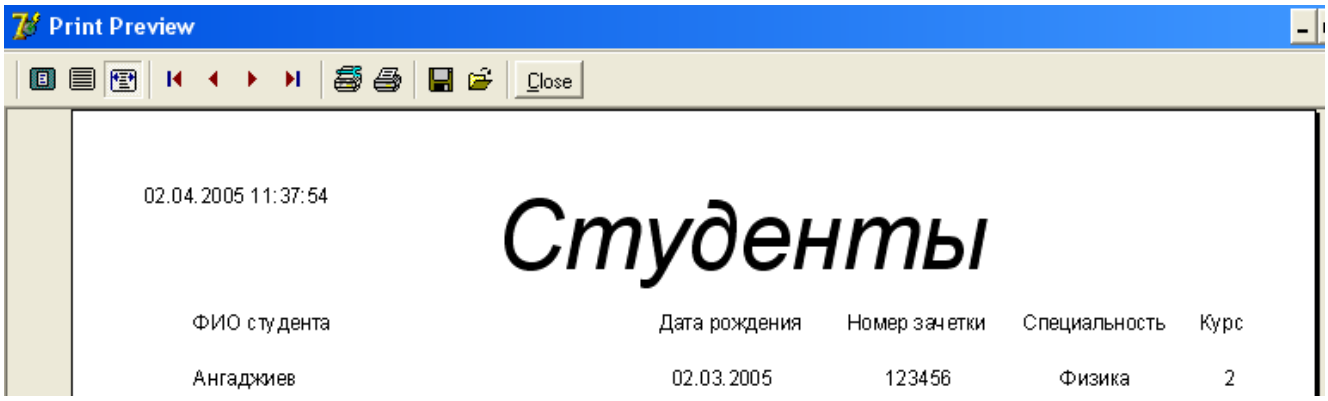

Рисунок 6.5

15. Установить на форму отчета компоненту – QRSubDetail с закладки QReport. Эта компонента предназначена для перебора данных относящихся к подчиненным таблицам.

16. Установить следующие свойства: DataSet – Form1.Table2, чтобы связать блок с таблицей Uspevaemost.db, которая является подчиненной к основной Studenti.db.

17. В свойстве Master нужно указать главную компоненту с основными данными. Выбрать в этом свойстве QuickRep1.

18. Расположить на компоненте QRSubDetail компоненты QRDBText в свойстве указав, к каким полям подчиненной таблицы они обращаются.

Получится следующий вид отчета (см. рисунок 6.6).

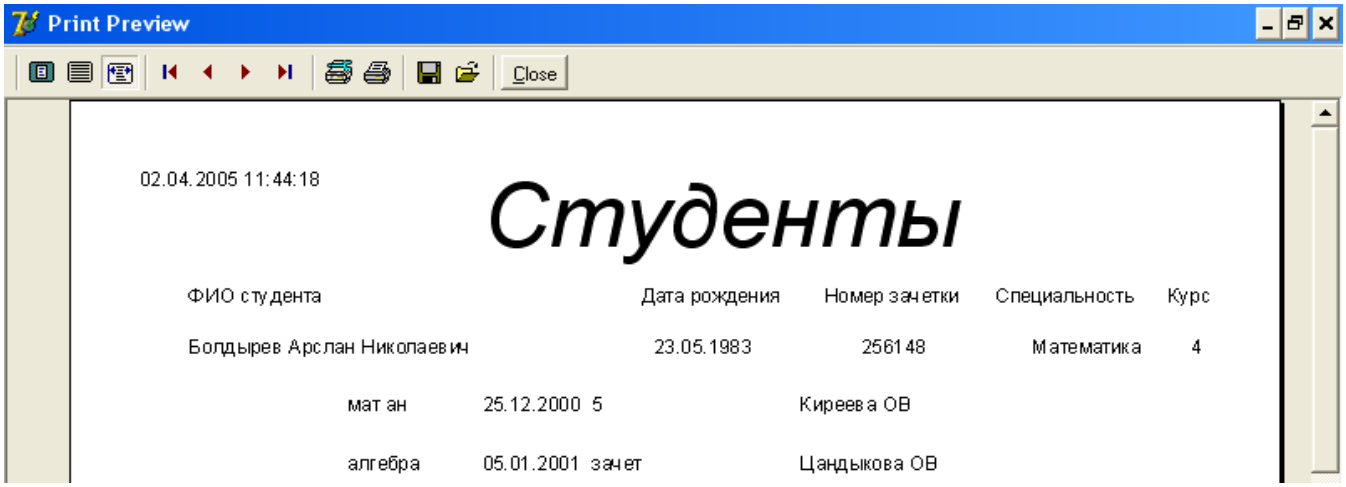

Рисунок 6.6

Контрольные вопросы.

1. Что такое отчет?
2. Как создается отчет?
3. Код кнопки «Печать».
4. Компоненты закладки QReport.
5. Рассказать о компоненте QuickRep
6. Функции компоненты QRSubDetail.
7. Назначение компоненты QRSysData.
8. Назначение компоненты QRLabel.

Приложение А

Перечень тем баз данных (БД).
Вариант А.
1. Разработка БД «Реализация товаров».
2. Разработка БД и приложения «Склад».
3. Разработка БД и приложения «Учет готовой продукции».
4. Разработка БД и приложения «Учет коммунальных услуг».
5. Разработка БД и приложения «Сбыт продукции».
6. Разработка БД и приложения «Учет готовых перевозок».
7. Разработка БД «Налоговые платежи».
8. Разработка БД «Составление акта об экспертизе качества товаров».
9. Разработка БД «Учет движения товаров в магазине».
10. Разработка БД «Учет и контроль инструментов на складе».
11. Разработка БД «Начисление заработной платы».
12. Разработка БД «Продажа билетов на автобус (поезд, самолет)».
13. Разработка БД «Обслуживание читателя в библиотеке».
14. Разработка БД и приложения «Отдел кадров».
15. Разработка БД «Торговый дом».
16. Разработка БД службы быта.
17. Разработка БД таможенной службы.
18. Разработка БД промышленного предприятия.
19. Разработка БД финансового органа.
20. Разработка БД транспортной службы.

Вариант Б.
1. Разработка БД «Акимат» (любой государственной службы).
2. Разработка БД страховой компании.
3. Разработка БД сертификации электрооборудования.
4. Разработка БД «Материально-техническое снабжение».
5. Разработка БД сертификации гостиничного комплекса.
6. Разработка БД «Аудиторская фирма».
7. Разработка БД «Строительная компания».
8. Разработка БД «Розничное торговое предприятие».
9. Разработка БД «Предприятие общественного питания».
10. Распределенные базы данных.
11. Использование БД для статистической обработки результатов.
12. Разработка БД производства продукции промышленного класса.
13. Разработка экспертной системы с применением СУБД сферы услуг.
14. Разработка БД для КСК.
15. Разработка БД «Музей».
16. Разработка БД «Расписание занятий».
17. Разработка БД «Учет добычи и сбыта угля».
18. Разработка БД для медицинского учреждения.

19. Разработка БД для гостиничного комплекса.
20. Разработка БД для определения выбора программного обеспечения.

Список литературы

1. Дейт К.Дж. Введение в системы баз данных. — М.: Издательский дом «Вильямс», 2008.

2. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. — М.: Издательский дом «Вильямс», 2003.

3. Кузнецов С.Д. Основы баз данных. — 1-е изд. — М.: «Интернет-университет информационных технологий - ИНТУИТ.ру», 2005.

4.Харрингтон Дж. Разработка баз данных. – М.: ДМК Пресс, 2005.

5. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. - Базы данных. Учебник для вузов. – М.: Корона-Принт, 2004.

6. Хансен Г., Хансен Д. Базы данных: разработка и управление. – М.: ЗАО «Издательство БИНОМ», 1999.

7. Кандзюба С.П., Громов В.Н. Delphi 6/7. Базы данных и приложения. – СПб: ООО «ДиаСофтЮП», 2002.

8. Григорьев Ю. А., Ревунов Г. И. Банки данных. – М.: Изд. МГТУ им. Н. Э. Баумана, 2002.

9. Харрингтон Д. Проектирование реляционных баз данных просто и доступно. – М.: Изд. «Лори», 2004.

Сводный план 2013 г. поз. 199

Наталья Александровна Водолазкина
Балнур Каирбаевна Каирбаева

СИСТЕМЫ БАЗ ДАННЫХ

Методические указания к лабораторным работам
для студентов специальности
5В070300 – Информационные системы

Редактор Н.М. Голева
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать ___.___._____. Формат 60х84 1/16
Тираж 30 экз. Бумага типографская №1
Объем 2,6 уч.- изд. л. Заказ ____Цена 1300 тн.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, Байтурсынулы, 126