Практическая работа № 13, База данных «Чемпионат по футболу»

1. Постановка задачи

Создать базу данных *Чемпионат по футболу*, которая будет состоять из нескольких таблиц. Для заполнения таблиц создать формы. Предусмотреть возможность поиска информации по ключевым полям.

2. Проектирование базы данных

2.1. Основные понятия

База Данных – организованная совокупность данных, предназначенная для длительного хранения во внешней памяти ЭВМ, постоянного обновления и использования. (Ершов А.П. Школьный словарь по информатике).

Можно доказать, что любую структуру данных можно преобразовать в простую двумерную таблицу. Такое представление является наиболее удобным и для пользователя, и для машины.

Реляционная база данных - совокупность данных состоящих из *связанных* двумерных таблиц.

Примечание
Название произошло от английского слова «relation» - отношение.

Основные понятия реляционных баз данных

1. Любые совокупности данных представляются в виде **двумерных таблиц**, каждая из которых содержит информацию об объектах определенного типа.
2. Каждая таблица состоит из **фиксированного числа столбцов** и **переменного числа строк**.
3. **Запись** – строка таблицы.
   Каждая запись содержит информацию об отдельном экземпляре объекта.
4. **Поле** – столбец таблицы.
   Каждый столбец представляет собой конкретное данное – одну характеристику объекта (атрибут). Для каждого поля разработчик должен определить:
   - уникальное имя поля;
   - тип поля;
   - дополнительные характеристики (длину, формат) поля.
5. **Ключ** – одно или несколько полей для идентификации записей таблицы.
6. Описание полей, определяемое разработчиком, называется структурой таблицы.

7. Каждое поле может входить в несколько таблиц.

8. Изменение количества полей и (или) их типов является особой операцией.

Основная идея реляционного подхода – представить произвольную структуру данных в виде простой двумерной таблицы. Такой процесс называется **нормализацией** структуры.

2.2. Нормализация баз данных

При проектировании структуры базы данных могут возникнуть проблемы:

- избыточность информации;
- противоречивость информации;
- потеря целостности (взаимосвязь между данными).

Процесс проектирования базы данных с использованием метода нормальных форм является пошаговым и заключается в последовательном переводе по определенным правилам отношений из первой нормальной формы в нормальные формы более высокого порядка.

Приступим к разработке базы данных, в которой будет храниться информация о футбольном чемпионате страны (дата матча, играющие команды, забитые голы). Представим эту информацию в виде таблицы 1. В структуре таблицы указаны только названия полей, т.к. тип и размерность полей на данном этапе значения не имеют.

Таблица 1

№	Имя поля
1	Дата матча
2	Команда хозяев: название, город, тренер
3	Команда гостей: название, город, тренер
4	Игрок, забивший гол
5	Признак команды, к которой принадлежит игрок
6	Время (число минут от начала матча)

Существуют основные правила нормализации структуры базы данных. Приведем только правила, с которыми будем работать.

Правило 1: В таблице необходимо разделить составные поля на отдельные элементы данных. Каждое поле таблицы должно представлять уникальный тип информации. Т.е. необходимо избавиться от повторяющихся полей (групп).

Правило 2: Каждая таблица должна иметь уникальный идентификатор (первичный ключ), который может состоять из одного или нескольких полей.

Правило 3: В таблице не должно быть данных, не относящихся к объекту, определяемому первичным ключом.

1 шаг (Правило 1)

В таблице 1 второе и третье поле являются составными, и содержат информацию о названии команды, города, фамилии тренера. В соответствии с **Правилом 1** необходимо эти поля разделить. У нас получится новая таблица 2.

Таблица 2

№	Имя поля
1	Дата матча
2	Команда хозяев: название
3	Команда хозяев: город
4	Команда хозяев: тренер
5	Команда гостей: название
6	Команда гостей: город
7	Команда гостей: тренер
8	Игрок, забивший гол
9	Признак команды, к которой принадлежит игрок
10	Время (число минут от начала матча)

Еще одно требование, которое мы должны учесть – это необходимость избавления от повторяющихся полей (групп). На первый взгляд может показаться, что в таблице 2 повторяющимися группами полей являются поля с информацией о командах хозяев и гостей. Но эти поля имеют различное функциональное значение.

2 шаг (Правило 2)

Записи таблицы 2 не содержат уникального ключа, по которому однозначно можно определить проводимый матч. Поэтому введем в таблицу 2 дополнительное поле ключа – код матча. У нас получится новая таблица 3.

Таблица 3

№	Имя поля
1	Код матча (ключ)
2	Дата матча
3	Команда хозяев: название
4	Команда хозяев: город
5	Команда хозяев: тренер
6	Команда гостей: название
7	Команда гостей: город
8	Команда гостей: тренер
9	Игрок, забивший гол
10	Признак команды, к которой принадлежит игрок
11	Время (число минут от начала матча)

Для каждого гола в таблице 3 содержится повторяющаяся информация о дате матча, о командах. Поэтому разобьем эту таблицу на две таблицы, одна будет содержать данные о матчах, а другая – о голах, забитых в каждом конкретном матче. Структура этих таблиц приведена в таблицах 4 и 5.

Таблица 4

№	Имя поля
1	Код матча (ключ)
2	Дата матча
3	Команда хозяев: название
4	Команда хозяев: город
5	Команда хозяев: тренер
6	Команда гостей: название
7	Команда гостей: город
8	Команда гостей: тренер

Таблица 5

№	Имя поля
1	Код гола (ключ)
2	Код матча
9	Игрок, забивший гол
10	Признак команды, к которой принадлежит игрок
11	Время (число минут от начала матча)

Таблицы 4 и 5 связаны по полю *Код матча*, которое для таблицы 4 является уникальным. Чтобы обеспечить уникальность записей таблицы 5, в нее введен ключ *Код гола*.

3 шаг (Правило 3)

Для выполнения Правила 3 необходимо выделить в отдельную таблицу те поля, которые не зависят от ключа *Код матча*. В таблице 4 такими полями являются поля, которые определяют команду. Разобьем таблицу 4 на две таблицы: первая – информация о матчах, вторая – информация о командах (см. таблицы 6 и 7).

Таблица 6

№	Имя поля
1	Код матча (ключ)
2	Дата матча
3	Код команды хозяев
4	Код команды гостей

Таблица 7

№	Имя поля
1	Код команды (ключ)
2	Название
3	Город
4	Тренер

В результате наша база данных *Чемпионат по футболу* будет иметь структуру, показанную на рисунке 1.

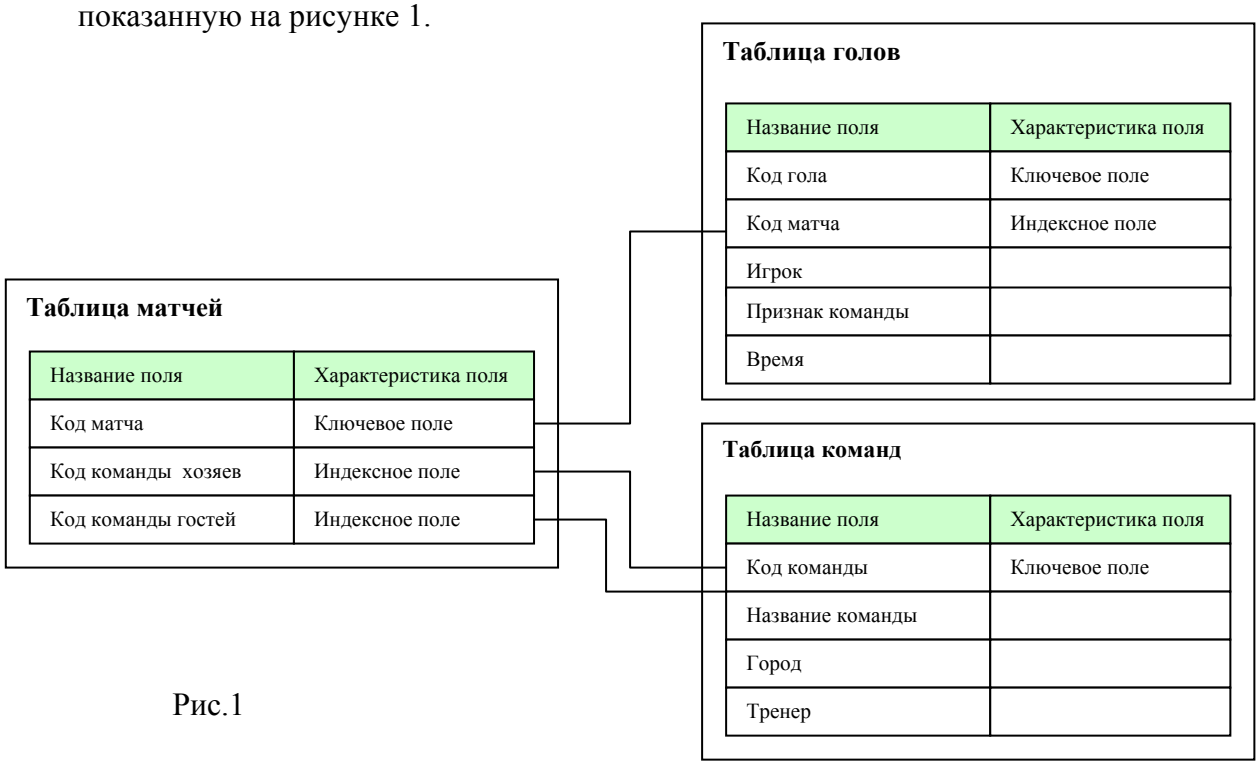

Рис.1

3. Пояснения к проекту

Проект будет состоять из пяти форм:

- главная форма программы, из которой открываются все остальные формы;
- форма *Список матчей*, содержащая список всех матчей чемпионата по футболу;
- форма *Список команд*, содержащая информацию о командах;
- форма *Список голов*, содержащая информацию о забитых голах;
- форма *Поиск*, в которой можно производить различные виды поиска.

4. Последовательность работы

4.1. Создание таблиц

4.1.1. Средства для работы с базами данных

Средства *Delphi*, предназначенные для работы с базами данных, можно разделить на два вида:

- *Инструментальные средства* – специальные программы, обеспечивающие обслуживание баз данных вне разрабатываемых приложений.
- *Компоненты*, предназначенные для создания приложений, осуществляющих операции с базами данных.

Примечание
Дальше будут рассмотрены инструментальные средства и компоненты, которые будут использоваться в данном проекте.

4.1.2. Инструментальные средства

- *Borland Database Engine* (BDE) – процессор баз данных, который представляет собой набор динамических библиотек и драйверов, предназначенных для организации доступа к базам данных из Delphi-приложений.
- *BDE Administrator* – утилита для настройки различных параметров BDE.
- Database Desktop – программа создания и редактирования таблиц, SQL-запросов.
- *SQL Explorer* – Проводник баз данных, позволяющий просматривать и редактировать базы данных.

4.1.3. Компоненты

Приведем компоненты, которые будут использованы в данном проекте.
Table – набор данных, основанный на таблице базы данных (страница *BDE*);
DataSource – источник данных (страница *Data Access*);
DBGrid – таблица (страница *Data Controls*);
DBNavigator – навигационный интерфейс (страница *Data Controls*);
DBEdit – однострочный редактор (страница *Data Controls*).

4.1.4. Псевдоним базы данных

Разрабатывая программу, трудно сразу предусмотреть на каком диске, в каком каталоге будут находиться файлы базы данных во время их использования. Для решения этой проблемы в *Delphi* используется псевдоним (*alias*), который указывает место нахождение файлов базы данных. Псевдоним – это короткое имя, поставленное в соответствие реальному, полному имени каталога базы данных. Псевдонимы сохраняются

в реестре, и потом все программы при запуске смогут по этим псевдонимам найти таблицу и прочитать необходимые настройки, которые надо использовать при доступе к данным.

Примечание
В принципе, можно обращаться к таблицам и без псевдонимов, но в этом случае путь придется жестко прописывать в программе. В этом случае лучше хранить таблицы и исполняемый файл в одной и той же папке.

4.1.5. Создание базы данных

Процесс создания базы данных может быть представлен как последовательность следующих шагов:

1. Создание папки.
2. Создание псевдонима.
3. Создание таблиц.

Создадим папку для нашего проекта и подпапку для базы данных с помощью средств Windows. Имя папки – *База Данных*, имя папки – *Данные*.

4.1.6. Создание псевдонима

Псевдоним (alias) может быть создан при помощи утилиты *BDE Administrator*:

C:\Program Files\Common Files\Borland Shared\BDE\bdeadmin.exe

На Рисунке 2 приведен вид диалогового окна *BDE Administrator* после запуска утилиты.

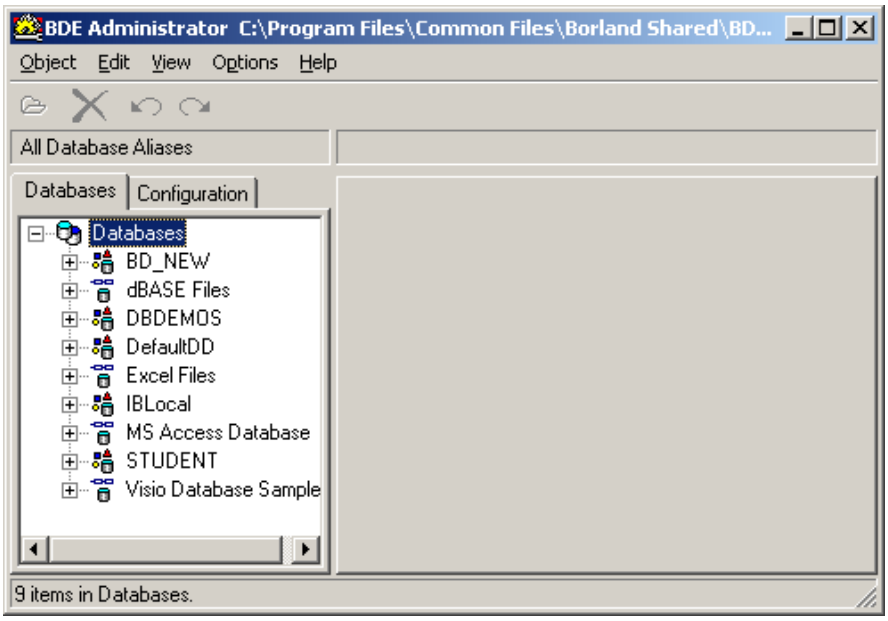

Рис.2

В левой части окна, на вкладке *Databases*, перечислены псевдонимы, зарегистрированные на данном компьютере. Для создания нового псевдонима необходимо выбрать команду меню *Object – New*. Откроется новое диалоговое окно *New Database Alias* (Рисунок 3) из списка *Database Driver Name* выберем драйвер (тип базы данных) *STANDARD*, который обеспечивает доступ к таблицам в формате *Paradox*.

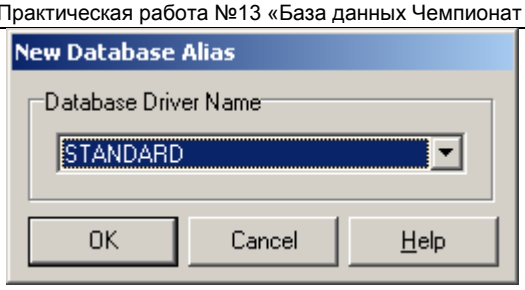

Рис.3

Для подтверждения выбора драйвера кликнем на клавише *OK*. В результате в список псевдонимов будет добавлен новый элемент (см. Рисунок 4).

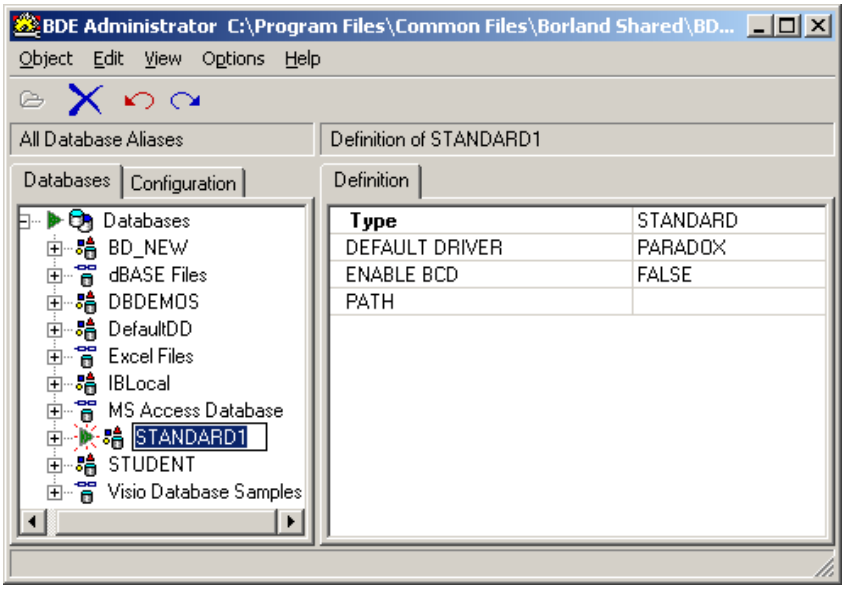

Рис.4

Теперь можно изменить автоматически созданное администратором имя псевдонима и задать путь к файлам базы данных.

Имя псевдонима можно изменить, щелкнув правой кнопкой мыши на имени псевдонима (на вкладке *Databases*), в открывшемся контекстном меню выбрать команду *Rename* и ввести новое имя – *SPORT*.

Путь к файлам базы данных вводится на вкладке Definition в поле Path с клавиатуры или с помощью стандартного диалогового окна Select Directory, которое открывается щелчком на кнопке с тремя точками, находящейся в конце поля Path (см. Рисунок 5).

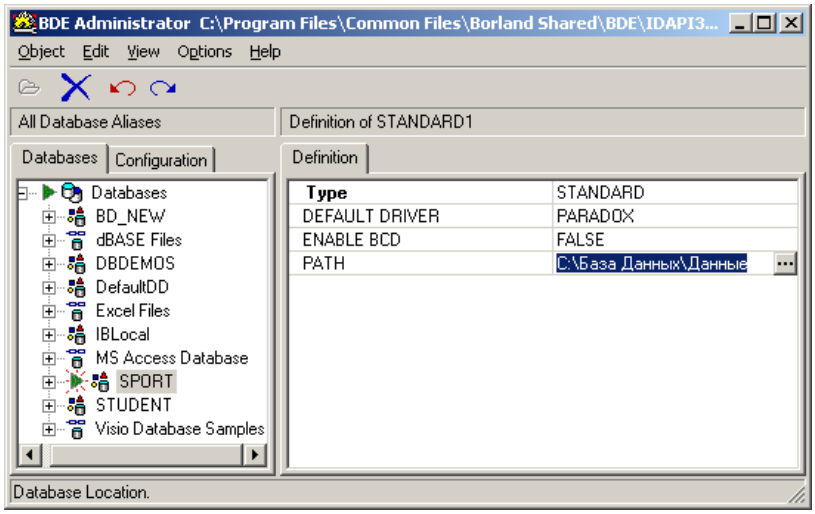

Рис.5

Для того чтобы созданный псевдоним был зарегистрирован в файле конфигурации (*idapi.cfg*), необходимо выполнить команду в меню *Object – Apple (Применить)*. В открывшемся диалоговом окне *Confirm* следует подтвердить необходимость сохранения изменений в файле конфигурации.

4.1.7. Создание таблиц

Приступим к созданию таблиц базы данных *Чемпионат по футболу*: таблица матчей – *Match*, таблица команд – *Team* и таблица голов – *Goal*. Структура этих таблиц приведена в таблицах 8, 9 и 10 соответственно.

Таблица матчей – *Match* Таблица 8

Field Name (имя поля)	Type (тип)	Size (размер)	Примечание
ID_M	N		Код матча (ключ)
DAT_M	D		Дата матча
HOST_M	N		Код команды хозяев
GUEST_M	N		Код команды гостей

Таблица команд – *Team* Таблица 9

Field Name (имя поля)	Type (тип)	Size (размер)	Примечание
ID_T	N		Код команды (ключ)
NAME_T	A	10	Название
STAT_T	A	14	Город
TRAINER_T	A	12	Тренер

Таблица голов – *Goal* Таблица 10

Field Name (имя поля)	Type (тип)	Size (размер)	Примечание
ID_G	N		Код гола (ключ)
MATCH_G	N		Код матча
DAT_G	A	12	Игрок, забивший гол
PR_G	A	1	Признак команды, к которой принадлежит игрок: 1 – хозяин, 2 – гость.
TIME_G	N		Время (число минут от начала матча)

Таблицы создаются с помощью входящей в состав *Delphi* утилиты *Database Desktop*. Эта утилита позволяет создавать, просматривать и модифицировать таблицы баз данных различных форматов. Вызвать утилиту *Database Desktop* можно:

C:\Program Files\Common Files\Borland Shared\Database Desktop\dbd32.exe

Для создания таблицы в окне *Database Desktop* выполните команду *File-New-Table...* Сначала в окне *Create Table* необходимо из раскрывающегося списка выбрать тип таблицы и нажать клавишу *Ok*. Пусть тип базы будет *Paradox7*. После этого открывается новое окно (см. рисунок 5), в котором необходимо создать структуру таблицы *Match*.

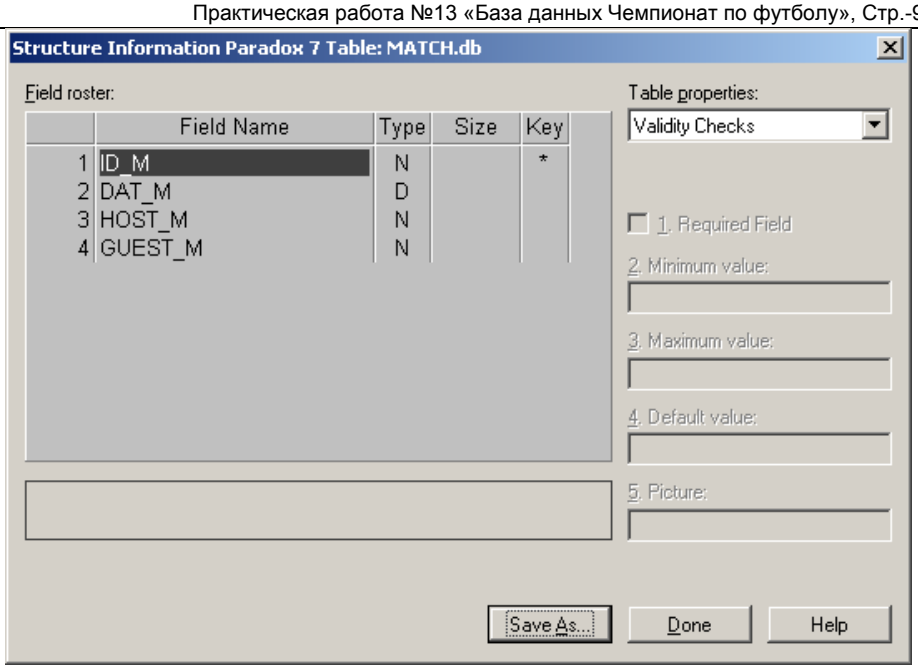

Рис. 5

Для каждого поля таблицы необходимо указать имя, тип, если нужно размер поля. Имя поля используется для доступа к данным. В качестве имени используется последовательность букв латинского алфавита и цифр длиной не более 25 символов. Для определения типа поля используйте клавишу пробел или правую клавишу мыши. Тип *Alpha* означает текстовый (строковый) тип поля. Для этого поля необходимо указать его длину. Для полей с типом *Number*, *Date* длину не указывают. Необходимо отметить признак ключевого поля *ID_M*, установив символ «*» в графе *Key*.

Примечание
Ключевые поля должны быть сгруппированы в верхней части таблицы.

После завершения заполнения таблицы сохраните ее, нажав кнопку *Save as*... В открывшемся окне *Save Table As*... в поле *Имя файла* введите имя таблицы *Match*, а в поле *Alias* выберите созданный ранее псевдоним *SPORT*. Для завершения работы нажмите клавишу *Save*.

При создании полей таблиц можно использовать задание ограничений на значения полей, которое заключаются в указании для этих полей следующих параметров:

1. Требование обязательного ввода значений (*Required Field*);
2. Минимальное значение (*Minimum value*);
3. Максимальное значение (*Maximum value*);
4. Значение по умолчанию (*Default value*);
5. Маска ввода (*Picture*).

На Рисунке 6 приведен пример заполнения поля *PR_G* (*Признак команды*), с указанием ограничений на значение поля.

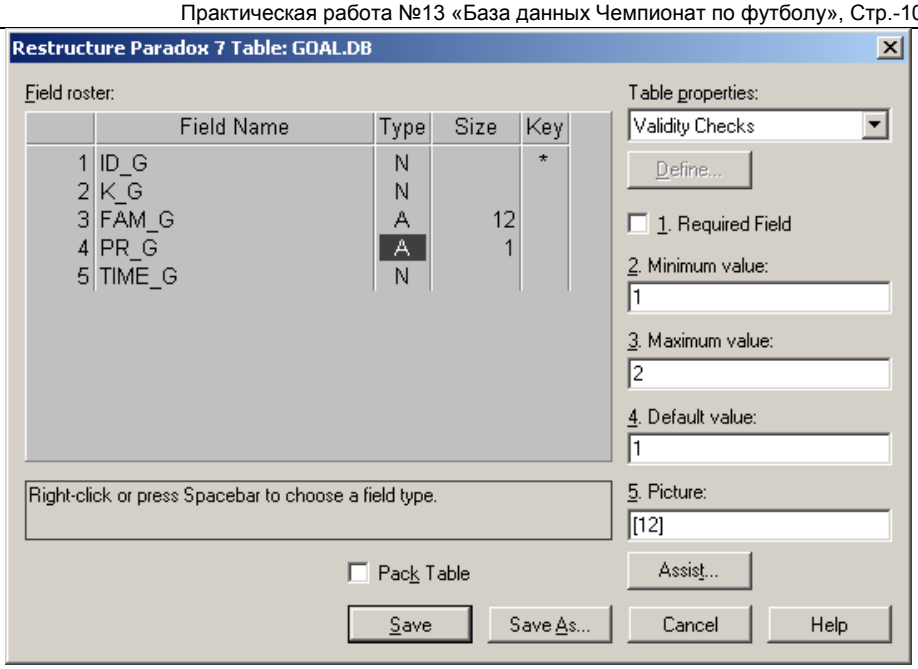

Рис.6

Аналогично создайте и сохраните таблицы команд – *Team* и голов – *Goal*.

Таблицы базы данных созданы, и теперь их можно приступить к разработке программы, использующей эти таблицы.

4.2. Создание форм

Создайте проект.

Таблица 10

Выделенная компонента	Окно инспектора объектов	Имя свойства	Действие
Form1	Properties	Caption	*База данных СПОРТ*
		Name	FormGlavn

Сохраните модуль и проект под именами UnitGlavn и ProjectGlavn в папке *База данных*.

Создайте четыре формы с помощью команды File-New-Other. В открывшемся окне *New Item* выберите на вкладке *New* объект *Form*. Дайте имена формам и сохраните модули под именами, указанными в таблице.

Таблица 11

Название формы	Имя формы	Имя модуля
Список матчей	FormMatch	UnitMatch
Список команд	FormTeam	UnitTeam
Список голов	FormGoal	UnitGoal
Поиск	FormPoisk	UnitPoisk

На главную форму поместите пять кнопок:

Список матчей, *Список команд*, *Список голов, Поиск, Выход*.

Для каждой кнопки напишите соответствующую процедуру для открытия окна (см.таблицу 12).

Таблица 12

Выделенная компонента	Окно инспектора объектов	Имя свойства	Действие
Button1	Properties	Caption	*Список матчей*
	Events	OnClick	FormMatch.Show;

Button2	Properties	Caption	*Список команд*
	Events	OnClick	FormTeam.Show;
Button3	Properties	Caption	*Список голов*
	Events	OnClick	FormGoal.Show;
Button4	Properties	Caption	*Поиск*
	Events	OnClick	FormPoisk.Show;
Button5	Properties	Caption	*Выход*
	Events	OnClick	Close;

В модуле главной формы после служебного слова *implementation* надо записать:

***Uses** UnitMatch, UnitTeam, UnitGoal, UnitPoisk;*

Сохраните изменения и запустите проект. Убедитесь, что все работает.

Вернитесь к проекту.

4.3. Доступ к базе данных

Доступ к базе данных обеспечивают компоненты *Database*, *Table* и *DataSource*.

DataBase **Table** **DataSource** **Компонент отображения данных, например, DBGrid**

Рис.7

Компонент *Database* представляет базу данных как единое целое, т.е. совокупность таблиц, компонента *Table* – одну из таблиц базы данных. Компонента *DataSource* обеспечивает связь таблицы и компонента отображения или редактирования данных (см. рисунок 7).

4.4. Использование модуля данных

При конструировании формы невизуальные компоненты, используемые для доступа к данным, такие как *DataSource* или *Table*, размещаются на форме, но при выполнении приложения эти компоненты не видны. Поэтому их можно размещать в любом удобном месте формы, выступающей для них контейнером – модулем. Для размещения невизуальных компонентов, через которые осуществляется доступ к данным, предназначен специальный объект – модуль данных (см. рисунок 8).

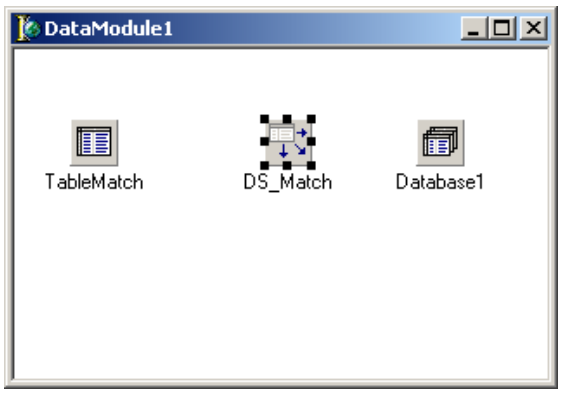

Рис.8

Создайте новый объект *DataModule*, выполнив команду *File-New-Data Module.* Сохраните его модуль под именем *UnitDModul* в папке *База данных*.

На лист окна *DataModule1* вставьте компонент Database (связь с сервером) со страницы *BDE*. В свойстве *AliasName* (имя псевдонима) выберите из списка: *SPORT*.

Добавьте в окно *DataModule1* компоненты *Table* (набор данных) со *BDE* и *DateSource* (источник данных) со страницы *Data Access* и расположите их рядом друг с другом (см. рисунок 8).

Активизируем таблицу *Match*. Для этого установим свойства компонент *Table1* и *DataSource1* в том порядке, в каком они перечислены в таблице 13.

Таблица 13

Выделенная компонента	Окно инспектора объектов	Имя свойства	Действие	Примечание
Table1	Properties	Name	*TableMatch*	Имя компонента для доступа к его свойствам.
		DatabaseName	*SPORT*	Имя базы данных, частью которой является таблица. Используется псевдоним базы данных.
		TableName	*Match.db*	Имя файла данных, для доступа к которому используется компонент.
		Active	*True*	Признак активизации файла данных (таблицы). *True* – открытие файла.
DataSource1		Name	*DS_Match*	Имя компонента для доступа к его свойствам.
		DateSet	*TableMatch*	Имя компонента – входные данные.

Выполните аналогичные действия для таблиц *Список команд – Team* и *Список голов – Goal*. В результате окно *DataModule1* будет выглядеть так, как приведено на рисунке 9.

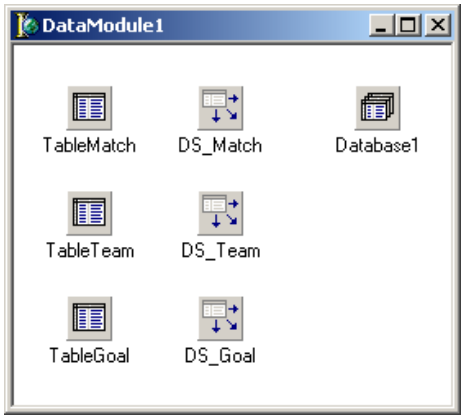

Рис.9

4.5. Навигация по таблицам базы данных

4.5.1. Форма *Список команд*

Активизируйте форму *Список команд*. Поместите на нее компонент *DBGrid* (таблица данных) со страницы *Data Controls* (управление данными). Для этого объекта следует прописать *DataSource* (источник данных). Откройте это свойство. Вы увидите, что

выбирать пока не из чего. В модуле формы *Список команд* после служебного слова *implementation* запишите:

***Uses** UnitDModul*;

Снова откройте свойство *DataSource* и выберите в нем единственную имеющуюся запись: *DataModule1*. *DS_Team*. Теперь компонент *DBGrid* и компонент *DataSource* связаны друг с другом. В компоненте *DBGrid* появились названия полей созданной таблицы *Team*.

Перейдите в окно *DataModule1* и щелкните два раза мышью по объекту *TableTeam*. Откроется небольшое окно *DataModule1*. *DS_Team*. Щелкните на поле этого окна правой кнопкой мыши и в контекстном меню выберите строку *Add all fields* (добавить все поля).

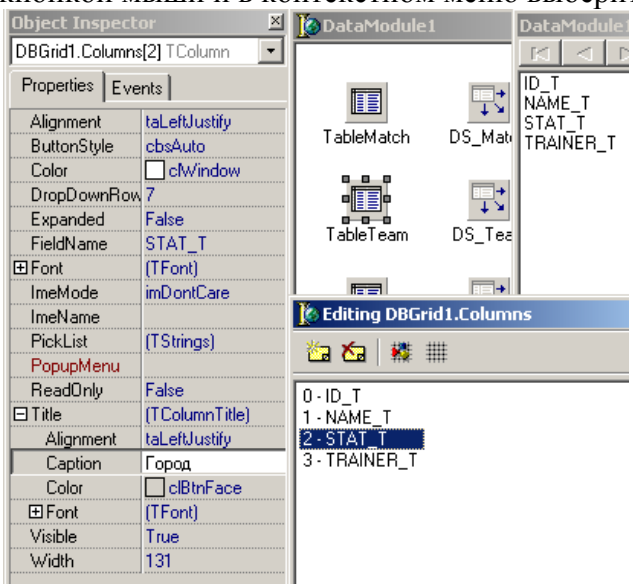

Рис.10

Перейдите к форме *Список матчей* и выполните двойной щелчок на объекте *DBGrid*. Открылось окно *Editing DBGrid1.Columns* (редактор столбцов). Щелкните на поле этого окна правой кнопкой мыши и в контекстном меню выберите строку *Add All Fields* (добавить все поля). В окне *Editing DBGrid1.Columns* появится список всех полей таблицы. Щелкните мышью на одном из появившихся названий полей. Откройте свойство *Title* (название) и для каждого поля в свойстве *Caption* запишите название: *Код команды, Название команды, Город, Тренер* (см. рисунок 10).

В результате этих действий русские названия полей отразятся в таблице *Список матчей*. Закройте окно *Editing DBGrid1.Columns*.

4.5.2. Перемещение по записям

Для перемещения указателя текущей записи в наборе данных используются следующие методы:

процедура *First* – установка на первую запись;

процедура *Next* – установка на следующую запись (для последней записи указатель не перемещается);

процедура *Last* – установка на последнюю запись;

процедура *Prior* – установка на предыдущую запись (для первой записи указатель не перемещается).

Delphi предоставляет возможность перемещаться по набору данных с помощью управляющих элементов, в качестве которых можно использовать компоненты *DBGrid* и *DBNavigator*. Управление этими элементами приводит к автоматическому вызову ранее перечисленных методов.

Перейдем на форму *Список команд*. Добавим на форму компонент *DBNavigator* (навигатор базы данных) со страницы *Data Controls* (управление данными). Навигатор содержит кнопки, обеспечивающие выполнение различных операций с набором данных путем автоматического вызова соответствующего метода. Состав кнопок определяется

свойством VisibleButtons. На рисунке 11 представлен общий вид компоненты *DBNavigator*.

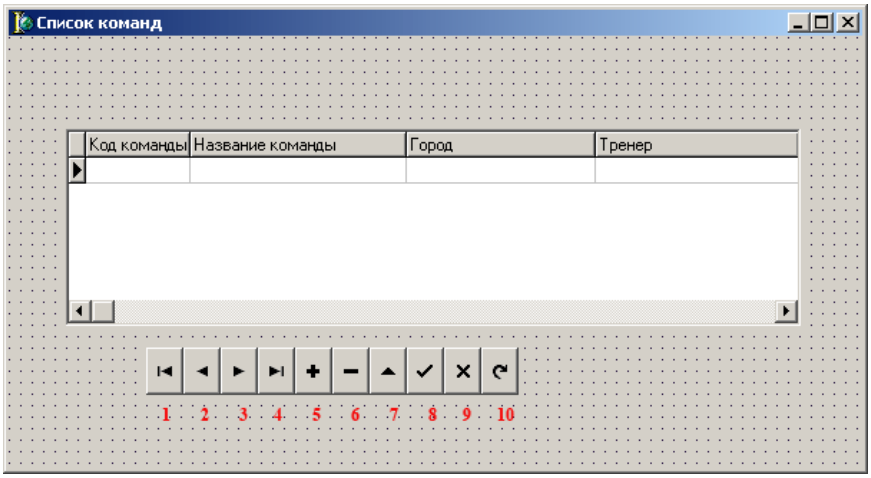

Рис.11

Кнопки навигатора выполняют следующие действия:

Таблица 14

Номер кнопки на рисунке	Обозначение кнопки	Действие
1	First	Перемещение к первой записи
2	Prior	Перемещение к предыдущей записи
3	Next	Перемещение к следующей записи
4	Last	Перемещение к последней записи
5	Insert	Вставка новой записи перед текущей
6	Delete	Удаление текущей записи
7	Edit	Редактирование текущей записи
8	Post	Сохранение отредактированной информации в базе данных.
9	Cancel	Отмена результата редактирования или добавления новой записи
10	Refresh	Очистка буфера, связанного с набором данных

Внесите изменения в свойства компонента *DBNavigator*.

Таблица 15

Выделенная компонента	Окно инспектора объектов	Имя свойства	Действие
DBNavigator	Properties	DataSource (источник данных)	*DataModule1.DS_Team* (установление связи объектов)
		ShowHint (показать подсказку)	True
		Hints (подсказка)	Щелкнуть на кнопке с тремя точками, расположенными справа. В появившемся окне встроенного редактора *String List Editor* заменить английские на русские названия кнопок: *Первая запись* *Предыдущая запись* *Следующая запись* *Последняя запись*

			Вставка записи *Удаление записи* *Редактирование записи* *Сохранение изменений* *Отменить изменения* *Обновить изменения* Завершить работу, щелкнув на кнопке OK.

Сохраните изменения и запустите проект. Убедитесь, что все работает.

4.5.3. Форма *Список матчей*

Активизируйте форму *Список матчей*. Поместите на нее компонент *DBGrid* и выполните аналогичные действия п.4.5.1 для таблицы *Match*.

По таблице можно перемещаться программно, без использования компоненты *DBNavigator*. Для этого внесем изменения в этот компонент, становим свойства, которые позволят использовать для навигации по таблице *Match* только четыре кнопки First, Prior, Next, Last (см. таблицу 12). Установите эти свойства в соответствии с рисунком 12.

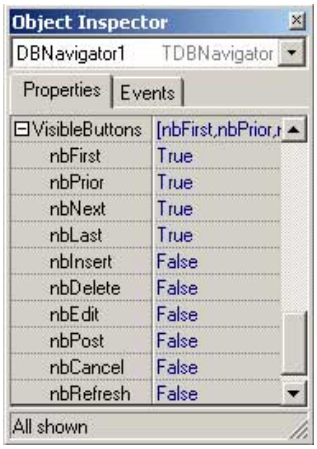

Рис.12

Добавьте на форму компоненты Button (*Изменить, Добавить, Удалить, Подтвердить, Отменить)*, Label для вывода состояния записи (Просмотр, Удаление, Редактирование, Вставка) и *CheckBox* для включения или выключения режима редактирования, как показано на рисунке 13. А так же разместите на форме компоненты меток и рядом с ними соответствующие компоненты для редактирования полей.

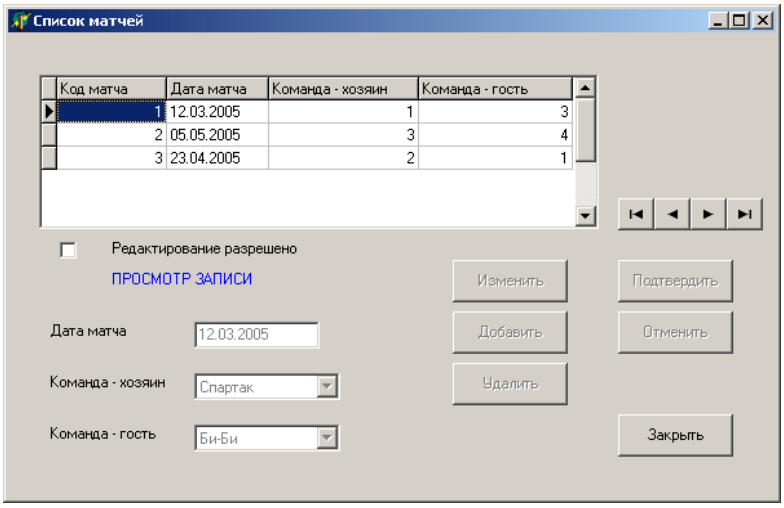

Рисунок 13

При заполнении полей *Команда – хозяин*, *Команда – гость* таблицы *Список матчей*, целесообразно значения этих полей выбирать из списка. Если для поля задана таблица выбора, то в него можно ввести только значение, содержащееся в таблице выбора. Это гарантирует, что в поле не будет введено недопустимое значение. Для этого воспользуемся компонентом *DBLookupComboBox*, с помощью которого можно выбирать нужную информацию из таблицы *Team*.

Выполните действия, приведенные в таблице 16.

Таблица 16

Выделенная компонента	Окно инспектора объектов	Имя свойства	Действие
Label1	Properties	Caption	*Дата матча*
DBEdit1 со страницы Data Control	Properties	DataSource	*DataModule1.DS_Match*
		DataField	*Dat_M*
Label2	Properties	Caption	*Команда - хозяин*
DBLookupComboBox1	Properties	DataSource	*DataModule1.Ds_Match*
		DataField	*HOST_M*
		ListSource	*DataModule1.DS_Team*
		KeyField	*ID_T*
		ListField	*NAME_T*
Label3	Properties	Caption	*Команда - гость*
DBLookupComboBox2	Properties	DataSource	*DataModule1.DS_Match*
		DataField	*GUEST_M*
		ListSource	*DataModule1.DS_Team*
		KeyField	*ID_T*
		ListField	*NAME_T*

Для того чтобы программа, которую мы будем писать, легко читалась, введем обозначения созданных кнопок и меток. Для этого необходимо изменить свойство *Name* у соответствующих компонентов (см. таблицу 17).

Таблица 17

Компонента	Условное обозначение	Свойство *Name*
TButton	*Изменить*	BtnEdit
TButton	*Добавить*	BtnInsert
TButton	*Удалить*	BtnDelete
TButton	*Подтвердить*	BtnChangeOK
TButton	*Отменить*	BtnChangeCancel
TButton	*Закрыть*	BtnClose
TLabel	*Состояние записи*	LblChangeKind
TDBEdit	*Дата матча*	DbeDat
TDBLookupComboBox	*Команда - хозяин*	DBLHost
TDBLookupComboBox	*Команда - гость*	DBLGuest
TDBGrid	*Таблица*	DBGridMatch
TCheckBox	*Режим редактирования*	cbCanEdit

В окне редактора форм перейдите в форму *UnitMatch*.

1. В разделе ***Use*** должны быть включены следующие стандартные модули:

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

DB, DBTables, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, StdCtrls, Mask;

2. А в разделе описания переменных перед ***implementation*** должна быть запись:

```
var
  FormSpisok: TFormSpisok;
```

3. После записи {$R *.dfm} вставим две вспомогательные процедуры:

```
procedure TFormMatch.StateChange(Sender: TObject);
begin
        btnEdit.Enabled:=false;
        btnInsert.Enabled:=false;
        btnDelete.Enabled:=false;
        btnChangeOK.Enabled:=true;
        btnChangeCancel.Enabled:=true;
end;

procedure TFormMatch.StateBrowse(Sender: TObject);
begin
        cbCanEditClick(Sender);
        btnChangeCancel.Enabled:=false;
        btnChangeOK.Enabled:=False;
end;
```

4. Перед разделом ***private*** { Private declarations }в раздел описания ***Type*** вставим две строки:

```
        procedure StateChange(Sender: TObject);
        procedure StateBrowse(Sender: TObject);
```

5. Для каждой кнопки напишите соответствующую процедуру.

BtnEdit – *Изменить* – OnClick

```
    DataModule1.DS_Match.Dataset.Edit;
    lblChangeKind.Font.Color:=clTeal;
    lblChangeKind.Caption:='РЕДАКТИРОВАНИЕ ЗАПИСИ';
    StateChange(Sender);
    if DbeDat.CanFocus then DbeDat.SetFocus;
```

BtnInsert – *Добавить* – OnClick

```
Var Nomer : Integer;
. . .
    // Подтверждение в режим вставки
    If MessageDlg('Добавить запись?',
    mtConfirmation, [mbYes, mbNo], 0) <> mrYes then Exit;
    DataModule1.DS_Match.Dataset.Last;
    Nomer:=DataModule1.DS_Match.Dataset.FieldByName('ID_M').AsInteger;
    DataModule1.DS_Match.Dataset.Append;
    // Номер матча формируется автоматически, путем увеличения номера в последней записи
    DataModule1.DS_Match.Dataset.FieldByName('ID_M').AsInteger:=Nomer+1;
    lblChangeKind.Font.Color:=clGreen;
    lblChangeKind.Caption:='ВСТАВКА ЗАПИСИ';
    StateChange(Sender);
    if DbeDat.CanFocus then DbeDat.SetFocus;
```

BtnDelete – *Удалить* – OnClick

```
// Запрос на подтверждение перехода в режим просмотра удаляемой записи
      If MessageDlg('Удалить запись?',
      mtConfirmation, [mbYes, mbNo], 0) <> mrYes then Exit;
      lblChangeKind.Font.Color:=clRed;
      lblChangeKind.Caption:='УДАЛЕНИЕ ЗАПИСИ';
      StateChange(Sender);
      if btnChangeCancel.CanFocus then btnChangeCancel.SetFocus;
```

BtnChangeOK – *Подтвердить* – OnClick

```
// Утверждение изменений в текущей записи (редактируемой или новой)
// или удаления текущей записи (просматриваемой)
If DataModule1.TableMatch.State in [dsEdit, dsInsert]
      then
      begin
            // Проверка заполнения полей
            If dbeDat.Text='' then
            begin
                  Beep;
                  MessageDlg('Не задана дата матча', mtError, [mbOK], 0);
                  if DbeDat.CanFocus then DbeDat.SetFocus;
                  Abort;
            end;
            If DBLHost.Text='' then
            begin
                  Beep;
            MessageDlg('Не задана команда - хозяин', mtError, [mbOK], 0);
                  if DBLHost.CanFocus then DBLHost.SetFocus;
                  Abort;
            end;
            If DBLGuest.Text='' then
            begin
                  Beep;
            MessageDlg('Не задана команда - гость', mtError, [mbOK], 0);
                  if DBLGuest.CanFocus then DBLGuest.SetFocus;
                  Abort;
            end;
            DataModule1.TableMatch.Post
      end
      else  if lblChangeKind.Caption='УДАЛЕНИЕ ЗАПИСИ'
      then DataModule1.TableMatch.Delete;
StateBrowse(Sender);
```

BtnChangeCancel – *Отменить* – OnClick

```
// Если набор данных находился в режиме просмотра (при удалении записи),
// то никаких действий метод Cancel не выполняет
      DataModule1.TableMatch.Cancel
      StateBrowse(Sender);
```

BtnClose – *Закрыть* – OnClick

```
     Close;
```

cbCanEdit – OnClick

```
var bm1: TBookmark;
. . .
// Запоминание положения текущей записи
      bm1:=DataModule1.Ds_Match.Dataset.GetBookmark;
// Отключение отображения изменений данных в визуальных компонентах
      DataModule1.Ds_Match.Dataset.DisableControls;
      If not cbCanEdit.Checked then begin
      DataModule1.TableMatch.Active:=false;
```

```
      DataModule1.TableMatch.ReadOnly:=true;
      DataModule1.TableMatch.Active:=true;
// Блокировка элементов, связанных с переходом
// в режиме изменения записей
      btnEdit.Enabled:=false;
      btnInsert.Enabled:=false;
      btnDelete.Enabled:=false;
      btnChangeCancel.Enabled:=false;
      btnChangeOK.Enabled:=false;
      lblChangeKind.Font.Color:=clBlue;
      lblChangeKind.Caption:='ПРОСМОТР ЗАПИСИ';
      DBEDat.Enabled:=false;
      DBLHost.Enabled:=false;
      DBLGuest.Enabled:=false;
      end
      else begin
      DataModule1.TableMatch.Active:=false;
      DataModule1.TableMatch.ReadOnly:=false;
      DataModule1.TableMatch.Active:=true;
// Разблокирование элементов, связанных с переходом
// в режиме изменения записей
      btnEdit.Enabled:=true;
      btnInsert.Enabled:=true;
      btnChangeCancel.Enabled:=true;
      btnChangeOK.Enabled:=true;
      DBEDat.Enabled:=true;
      DBLHost.Enabled:=true;
      DBLGuest.Enabled:=true;
// Если набор данных пуст, то удаление записей запрещено
      If DataModule1.Ds_Match.Dataset.RecordCount>0
            then btnDelete.Enabled:=true
            else btnDelete.Enabled:=false;
      end;
// Возврат в текущую запись
      If DataModule1.Ds_Match.Dataset.BookmarkValid(bm1)
            then DataModule1.Ds_Match.Dataset.GotoBookmark(bm1);
      If DataModule1.Ds_Match.Dataset.BookmarkValid(bm1)
            then DataModule1.Ds_Match.Dataset.FreeBookmark(bm1);
// Включение отображения изменений данных в визуальных компонентах
      DataModule1.Ds_Match.Dataset.EnableControls;
```

FormMatch – OnCreate

```
// Первоначально изменение записей запрещено
      cbCanEdit.Checked:=false;
// Запрет автоматического перехода в режим редактирования
     DataModule1.DS_Match.AutoEdit:=false;
     DBGridMatch.Columns[0].ReadOnly:=True;
```

FormMatch – OnShow

```
// Исходное состояние управляющих элементов
      StateBrowse(Sender);
```

4.5.4. Форма *Список голов*

Самостоятельно разработайте процесс заполнения и навигацию в форме *Список голов*.

4.5.5. Задание для самостоятельной работы

1. На форме *Список матчей* добавить для каждой команды (команда-хозяин, команда-гость) информацию о городе команды и фамилии тренера.

2. На форме *Список матчей* в таблице матчей убрать (видимость) поле код матча.

3. На форме *Список команд* добавить кнопку *Закрыть* форму.

На оценку 5

Самостоятельно разработать форму поиска матчей:

1. по дате
2. по команде.

Для найденного матча показать список всех забитых голов и общий счет.

Описать процесс создания формы по примеру данного проекта.

5. Приложение. Пример реализации поиска

Добавьте на форму компоненты DBGrid, GroupBox *(Найти)*, Button (*Поиск, Выход)*, CheckBox *(По фамилии, По факультету)*, Edit для ввода ключевых значений для поиска по полям *DAT* и *FAK*, как показано на рисунке 14.

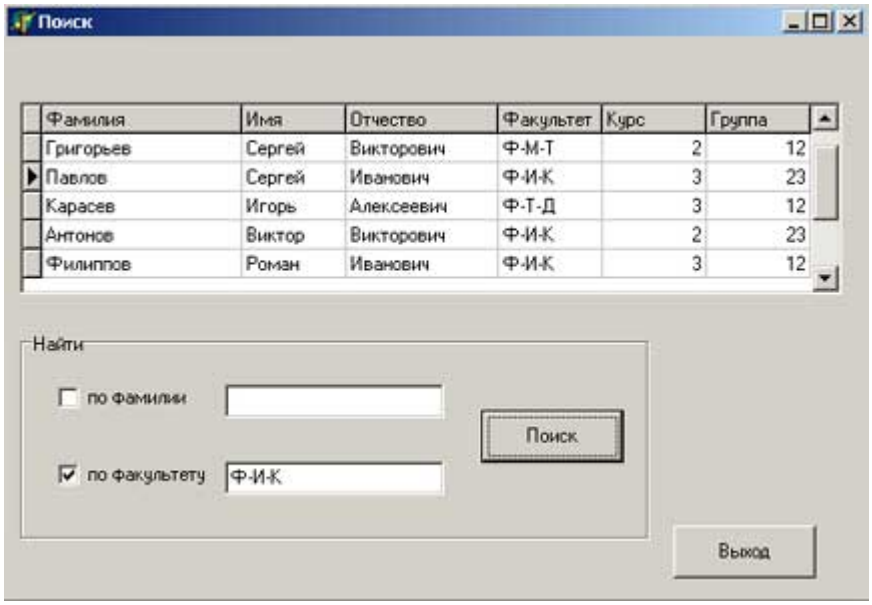

Рисунок 14

Для того, чтобы программа, которую мы будем писать, легко читалась, введем обозначения созданных кнопок и меток. Для этого необходимо изменить свойство *Name* у соответствующих компонентов.

Таблица 16

Компонента	Условное обозначение	Свойство *Name*
TButton	*Поиск*	BtnFind
TButton	*Выход*	BtnClose
TGroupBox	*Найти*	GroupBox
TCheckBox	*По фамилии*	cbFindDAT
TCheckBox	*По факультету*	cbFindFAK
TEdit	Для ввода фамилии	EditDAT
TEdit	Для ввода факультета	EditFAK
TDBGrid	*Таблица*	DBGridSpis

В окне редактора форм перейдите в форму *UnitPoisk*.

1. В разделе ***Use*** должны быть включены следующие стандартные модули:

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
DB,Dialogs, StdCtrls, ExtCtrls, Grids, DBGrids;

2. Для поиска записей по полям служат методы Locate и Lookup, причем поля могут быть неиндексированными.
Функция Locate (const KeyFields: String;
const KeyValues: Variant;
Options: TLocateOptions): Boolean
ищет запись с заданными значениями полей. Если удовлетворяющие условиям поиска записи существуют, то указатель текущей записи устанавливается на первую из них и функция возвращает значение True. Список полей, по которым ведется поиск, задается в параметре KeyFields (поля разделяются точкой с запятой). Параметр KeyValues указывает значения полей для поиска. Параметр Options задает значения LoCaseInsensitive (регистр букв не учитывать) и LoPartialKey (допускается частичное совпадение значений).

3. Для кнопки Поиск напишите соответствующую процедуру.

BtnFind – *Поиск* – OnClick

```
procedure TFormPOISK.btnFindClick(Sender: TObject);
Var KeyFields: String;
   KeyValues: Variant;
   Options  : TLocateOptions;
begin
  if not (cbFindDAT.Checked or cbFindFAK.Checked)
     then
       begin
         MessageDlg('Не заданы условия поиска!', mtInformation,[mbOK],0);
         exit;
       end;

  //Поиск одновременно по двум полям DAT и FAK
  if cbFindDAT.Checked and cbFindFAK.Checked
     then
       begin
          KeyFields:='DAT;FAK';
          KeyValues:=VarArrayOf([editDAT.Text, editFAK.Text]);
       end
  //Поиск по одному из полей
    Else
       begin
         //По полю DAT
         if cbFindDAT.Checked
            then
              begin
                 KeyFields:='DAT';
                 KeyValues:=editDAT.Text;
              end;
         //По полю FAK
        if cbFindFAK.Checked
           then
             begin
                KeyFields:='FAK';
                KeyValues:=editFAK.Text;
             end;
```

```
        end;
  //Поиск выполняется независимо от регистра букв
  //с возможностью частичного совпадения
  Options:=[LoCaseInsensitive,LoPartialKey];
  //Запись не найдена
  If not DataModule1.Ds_Spisok.Dataset.Locate(KeyFields, KeyValues, Options)
     then
        begin
          Beep;
          MessageDlg('Запись не найдена...', mtInformation, [mbOK],0);
          exit;
        end;
end;
```