Методические рекомендации по дисциплине
«Технология разработки программного обеспечения»

ОБЩИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Согласовав вариант задания с преподавателем, студент приступает к выполнению лабораторных работ. Выполнив работу, студент должен предоставить преподавателю результаты в электронном виде (файл базы данных, исходные файлы, исполняемые файлы проекта) в зависимости от поставленной задачи.

Выполнив лабораторную работу, также необходимо оформить отчёт, который состоит из следующих разделов:

1. Постановка задачи.
2. Описание процесса выполнения работы.
3. Результаты выполнения работы.

При защите отчёта необходимо ответить на вопросы преподавателя.

ПЕРЕЧЕНЬ ЛАБОРАТОРНЫХ РАБОТ

1. Создание локальной базы данных.
2. Разработка 1-й версии программы.
3. Модификация базы данных. Создание запросов.
4. Разработка 2-й версии программы. Просмотр основных данных. Реализация быстрого поиска.
5. Разработка 2-й версии программы. Реализация функций добавления, редактирования и удаления основных данных.
6. Разработка 2-й версии программы. Формирование отчётов.
7. Документирование программного обеспечения. Составление Пояснительная записка к эскизному проекту на создание автоматизированной системы.
8. CASE-технология поддержки разработки и сопровождения

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

«Расписания движения поездов»

Информация хранится в базе данных следующей структуры:

- ***поезд*** (код поезда, № поезда, название, код станции отправления, код станции прибытия);
- ***станция*** (код станции, название станции);
- ***расписание поезда*** (код, код поезда, время в пути, время прибытия, время стоянки, время отправления, код станции);
- ***состав*** (код состава, код поезда, номер состава, дата отправления,

ФИО начальника поезда).

Вариант 2

«**Учёт работ строительной компании**»

Информация хранится в базе данных следующей структуры:

- ***подразделение*** (код подразделения, название подразделения);
- ***работник*** (код работника, ФИО работника, дата рождения, ИНН, № пенсионного страхового свидетельства, код подразделения, паспортные данные);
- ***справочник работ*** (код работы, название работы);
- ***заказчик*** (код заказчика, наименование, телефон, адрес, ИНН);
- ***заказ*** (код заказа, код заказчика, название объекта, содержание работ, дата начала работы, дата окончания работы);
- ***работа*** (код заказа, код работы, код работника, дата начала работы, дата окончания работы, описание работы).

Вариант 3

«**Учёт материалов на складе**»

Информация хранится в базе данных следующей структуры:

- ***материал*** (код материала, название материала, код категории);
- ***категория*** (код категории, название категории, единицы измерения);
- ***поступление материала*** (код поступления, код материала, количество, код накладной);
- ***накладная*** (код накладной, дата оформления);
- ***расход материала*** (код расхода, код материала, количество, код накладной).

Вариант 4

«**Учёт студентов, проживающих в общежитиях**»

Информация хранится в базе данных следующей структуры:

- ***группа*** (код группы, название группы, название факультета);
- ***общежитие*** (код общежития, название, адрес);
- ***комната*** (код комнаты, № комнаты, код общежития);
- ***студент*** (код студента, ФИО студента, дата рождения, пол, код группы, серия паспорта, номер паспорта, кем и когда выдан);
- ***информация о заселении*** (код комнаты, код студента, дата заселения, дата выселения).

Вариант 5

«**Учёт работы поликлиники**»

Информация хранится в базе данных следующей структуры:

- ***отделение*** (код отделения, название отделения);
- ***должность*** (код должности, название должности);
- ***врач*** (код врача, ФИО врача, дата рождения, код отделения, код должности, пол);
- ***пациент*** (код пациента, ФИО пациента, дата рождения, пол, № медицинского полиса, паспортные данные, пол);
- ***приём у врача*** (код приёма, код пациента, код врача, дата приёма, время приёма, отчёт о приёме).

Вариант 6

«**Учёт отпусков сотрудников**»

Информация хранится в базе данных следующей структуры:

- ***отдел*** (код отдела, название отдела);
- ***должность*** (код должности, название должности);
- ***сотрудник*** (табельный номер, ФИО сотрудника, дата рождения, ИНН, № пенсионного страхового свидетельства, паспортные данные);
- ***вид отпуска*** (код вида отпуска, вид отпуска);
- ***рабочее место*** (код работы, код сотрудника, код должности, код отдела, дата начала работы, дата завершения работы);
- ***отпуск*** (код отпуска, код вида отпуска, код работы, дата начала отпуска, дата окончания отпуска).

Вариант 7

«**Учёт арендуемых помещений**»

Информация хранится в базе данных следующей структуры:

- ***здание*** (код здания, название, адрес);
- ***помещение*** (код помещения, название помещения, площадь, код здания);
- ***арендатор*** (код арендатора, название фирмы, юридический адрес, ФИО руководителя, контактный телефон);
- ***аренда*** (код аренды, код помещения, код арендатора, № договора, дата оформления договора, дата начала аренды, дата окончания аренды).

Вариант 8

«**Учёт работы автомастерской**»

Информация хранится в базе данных следующей структуры:

- ***вид работ*** (код вида работ, вид работ);
- ***модели машин*** (код модели, название модели);

- ***автовладелец*** (код автовладельца, ФИО автовладельца, серия паспорта, номер паспорта, кем и когда выдан);
- ***автомобиль*** (код автомобиля, код модели, код автовладельца, номер автомобиля);
- ***мастер*** (код мастера, ФИО мастера, паспортные данные, дата рождения);
- ***работа*** (код автомобиля, код вида работ, дата начала работ, код мастера, дата окончания работ, описание проделанной работы).

Вариант 9

«**Учёт работы туристического агентства**»

Информация хранится в базе данных следующей структуры:

- ***туроператор*** (код туроператора, название туроператора, ФИО контактного лица, контактный телефон, факс, адрес в WWW, e-mail);
- ***место назначения*** (код места назначения, название);
- ***курорт*** (код курорта, код места назначения, название курорта);
- ***клиент*** (код клиента, ФИО клиента, паспорт, контактный телефон);
- ***тур*** (код тура, код курорта, начало тура, окончание тура, код клиента, № договора, дата оплаты, стоимость).

Вариант 10

«**Учёт библиотечного фонда**»

Информация хранится в базе данных следующей структуры:

- ***книга*** (код книги, наименование, год издания, цена, ISBN, код типа книги);
- ***тип книги*** (код типа книги, тип книги);
- ***поступления*** (№ экземпляра, код книги, код типа поступления, дата поступления);
- ***тип поступления*** (код типа поступления, тип поступления);
- ***списание*** (код списания, № экземпляра, дата списания, № акта, причина списания);
- ***абонемент*** (код записи в абонементе, № экземпляра, ФИО читателя, дата получения, дата возвращения).

Вариант 11

«**Учёт материальных ценностей**»

Информация хранится в базе данных следующей структуры:

- ***материально-ответственное лицо*** (код сотрудника, ФИО сотрудника, паспортные данные, подразделение);

- ***тип ценности*** (код типа ценности, тип ценности);
- ***акт приёма*** (код акта приёма, № акта, дата оформления);
- ***акт списания*** (код акта списания, № акта, дата оформления);
- ***материальная ценность*** (код ценности, инвентарный номер, название, стоимость, код акта приёма, код акта списания, код типа ценности, код сотрудника).

Вариант 12

«**Учёт выступлений студентов на научных конференциях**»

Информация хранится в базе данных следующей структуры:

- ***группа*** (код группы, название группы);
- ***студент*** (код студента, ФИО студента, № зачётки, дата рождения, пол студента, код группы);
- ***научный руководитель*** (код сотрудника, ФИО сотрудника, кафедра, должность);
- ***конференция*** (код конференции, название, место проведения, дата начала, дата окончания);
- ***доклад*** (код доклада, название доклада, дата выступления, код конференции, код студента, код сотрудника).

Вариант 13

«**Учёт работы научных конференций**»

Информация хранится в базе данных следующей структуры:

- ***конференция*** (код конференции, название конференции, дата начала, дата окончания);
- ***руководители секций*** (код сотрудника, ФИО сотрудника, кафедра, должность);
- ***секция*** (код секции, название секции, дата начала работы, дата завершения работы, код конференции, код сотрудника);
- ***участник*** (код участника, ФИО участника, место и должность работы, e-mail, контактный телефон, адрес);
- ***доклад*** (код доклада, название доклада, дата выступления, код секции, код участника).

Вариант 14

«**Учёт успеваемости студентов**»

Информация хранится в базе данных следующей структуры:

- ***специальность*** (код специальности, название специальности);
- ***учебная группа*** (код группы, название группы, код специальности);
- ***студент*** (код студента, ФИО студента, № зачётки, дата рождения, код группы);

- ***преподаватель*** (код преподавателя, табельный №, ФИО преподавателя, кафедра, должность);
- ***дисциплина*** (код дисциплины, название дисциплины, количество лекционных часов, количество часов практических занятий, количество часов лабораторных занятий, семестр);
- ***успеваемость*** (код записи, код дисциплины, код преподавателя, код студента, форма контроля, оценка, дата сдачи).

Вариант 15

«**Учёт посещаемости студентов**»

Информация хранится в базе данных следующей структуры:

- ***студент*** (код студента, ФИО студента, № зачётки, дата рождения, группа);
- ***вид занятия*** (код вида занятия, название вида занятия);
- ***преподаватель*** (код преподавателя, табельный №, ФИО преподавателя, кафедра, должность);
- ***дисциплина*** (код дисциплины, название дисциплины, семестр);
- ***занятия*** (код занятия, код дисциплины, код преподавателя, код вида занятия, дата, № пары, тема занятия);
- ***посещаемость*** (код занятия, код студента).

УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Рассмотрим пример разработки приложения для работы с базой данных «Информация о студентах».

База данных должна быть реализована в СУБД Microsoft Access версии 2003 (можно и выше). Для разработки приложения используется интегрированная среда разработки и технологии доступа ADO и OLE DB.

Лабораторная работа №1 Создание локальной базы данных

Для работы приложения в СУБД MS Access необходимо создать базу данных «Информация о студентах», которая должна храниться в файле «data.mdb» в том же каталоге, что и исходные файлы разрабатываемой программы.

База данных состоит из следующих таблиц:

- Grupa – Справочник студенческих групп;
- Inf – Контактная информация студента;
- Parents – Информация о родителях студентов;
- Student – Основные данные о студентах;
- Tipinf – Справочник типов контактной информации.

Структуры таблиц создаваемой базы описаны в следующих таблицах.

Таблица 1. Структура таблицы «grupa»

№	Поле	Тип данных	Дополнительно	Описание
1.	idgrupa	счётчик	ключ	Первичный ключ
2.	grupa	текстовый	250	Название группы

Таблица 2. Структура таблицы «inf»

№	Поле	Тип данных	Дополнительно	Описание
1.	idinf	счётчик	ключ	Первичный ключ
2.	idtipinf	числовой	Длинное целое	Внешний ключ
3.	idstudent	числовой	Длинное целое	Внешний ключ
4.	inf	текстовый	250	Контактная информация

Таблица 3. Структура таблицы «parents»

№	Поле	Тип данных	Дополнительно	Описание
1.	idparents	счётчик	ключ	Первичный ключ
2.	fioparents	текстовый	250	Ф.И.О. родителей
3.	step	текстовый	250	Степень родства
4.	infparents	поле MEMO		Дополнительная информация о родителях (место работы, контакты и т.д.)
5.	idstudent	числовой	Длинное целое	Внешний ключ

Таблица 4. Структура таблицы «student»

№	Поле	Тип данных	Дополнительно	Описание
1.	idstudent	счётчик	ключ	Первичный ключ
2.	nshifr	текстовый	6	№ зачётки
3.	fiostudent	текстовый	250	Ф.И.О. студента
4.	dbirth	дата/время	Краткий формат даты	Дата рождения
5.	pasport	текстовый	250	Паспортные данные
6.	resume	поле MEMO		Дополнительные сведения (характеристика)
7.	idgrupa	числовой	Длинное целое	Внешний ключ
8.	sex	текстовый	250	Пол (мужской/женский)

Таблица 5. Структура таблицы «tipinf»

№	Поле	Тип данных	Дополнительно	Описание
1.	idtipinf	счётчик	ключ	Первичный ключ
2.	tipinf	текстовый	250	Тип контактной информации.

Для поддержки целостности данных создаём схему данных. Устанавливаем связи между таблицами согласно рисунку 1.

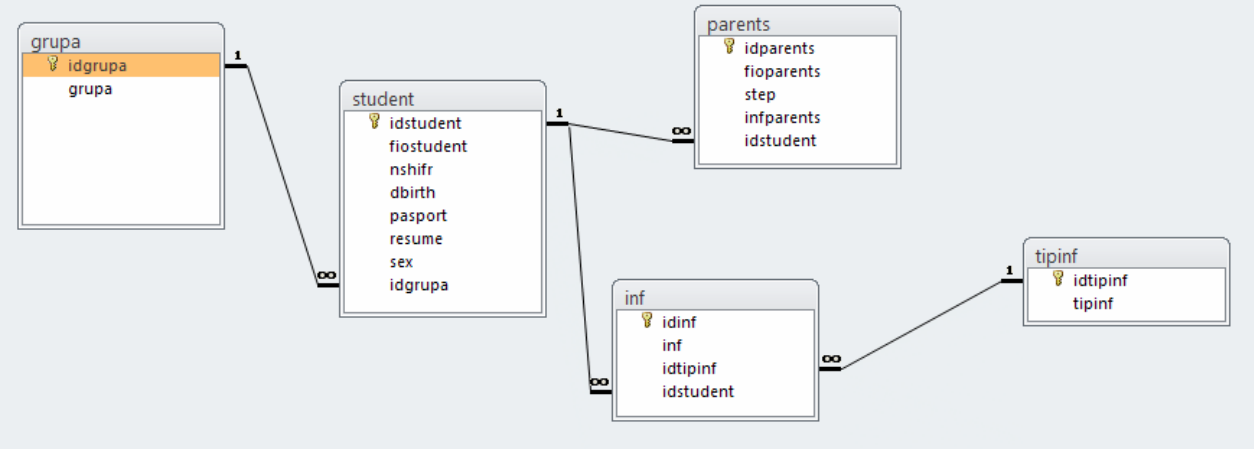

Рисунок 1. Схема данных базы данных «Студенты»

При установке связей между таблицами по соответствующим полям необходимо отметить свойства «Обеспечение целостности данных», «каскадное обновление связанных полей» и «каскадное удаление связанных записей» (рисунок 2), что позволит обеспечить целостность данных.

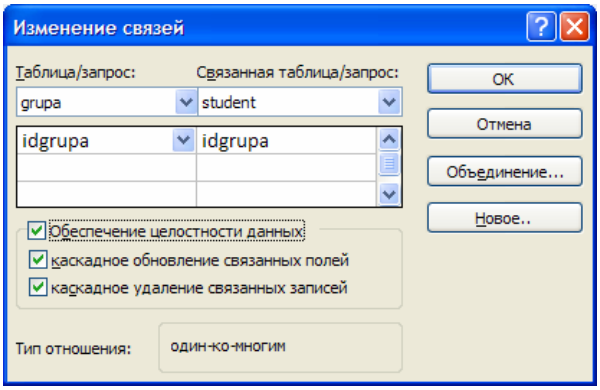

Рисунок 2. Свойства связи между таблицами «Student» и «Grupa»

Лабораторная работа №2 Разработка 1-й версии программы

Приступим к разработке первой версии программы. Это будет первый прототип программы.

Создаём каталог «stud_v1». Файл базы данных, созданный на предыдущем этапе, должен находиться в этом каталоге.

Открываем среду разработки приложений и создаём новый проект типа «VCL Forms Application».

В новом проекте автоматически создаётся форма. Эта форма будет для нашего проекта основной или главной. Для неё устанавливаем следующие свойства:

- name – fMain;
- position – poDesktopCenter;
- borderstyle – bsSingle;
- bordericons - [biSystemMenu,biMinimize];
- caption – Информация о студентах (v1).

Сохраняем изменения во всём проекте (File → Save All). При первом сохранении проекта или новых модулей необходимо указать для них имена, при последующих сохранениях этого не потребуется. Unit1, относящийся к главной форме, сохраняем под именем «ufMain». Проект сохраняем под именем «stud_v1».

Для размещения невизуальных компонентов и упрощения доступа к ним добавляем в проект Data Module (File → New → Other → Delphi Files → Data Module). Переименовываем его в «dm». Сохраняем изменения во всём проекте и у вновь появившегося модуля, который относится к «dm», изменяем имя на «udm». Сохранение всех изменений в проекте будет выполняться регулярно для сохранения изменений в уже созданных модулях, при добавлении новых модулей и перед компиляцией и запуском проекта.

Для начала необходимо обеспечить доступ к информации, которая находится в базе данных. Для этого мы используем технологию ADO. В «dm» устанавливаем следующие компоненты: 1 шт. – ADOConnection, 5 шт. – ADOTable и 5 шт. – DataSource. Используйте быстрый поиск в «Tool Palette», чтобы найти эти компоненты. ADOConnection необходимо для установки соединения с базой данных. ADOTable необходимы для подключения и работы с таблицами и запросами, созданными в базе данных. DataSource необходимы для создания источников данных, ассоциируемых с наборами данных (в нашем примере ADOTable).

ADOConnection переименуем в «ADOData». Этот компонент позволяет установить централизованное соединение с базой данных. Для этого выполняем некоторые настройки. Сначала сформируем строку соединения (свойство ConnectionString). Нажимаем  в поле этого свойства. Запускается окно, в котором выбираем источник соединения (рисунок 3).

Рисунок 3. Источник соединения

Можно использовать внешний файл, в котором будут храниться настройки соединения (файл с расширением «udl»). Этот способ будет эффективен при работе с сервером базы данных, так как параметры сервера могут меняться вне зависимости от разработчика. В нашем случае, при работе с локальной базой данных, эффективнее второй способ «Use Connection string». Для того чтобы сформировать строку подключения, мы нажимаем кнопку «Build». Формирование строки соединения происходит в несколько этапов. Сначала мы выбираем поставщика (рисунок 4), для связи с базой в формате СУБД MS Access 2003 используем Microsoft Jet 4.0 OLE DB Provider.

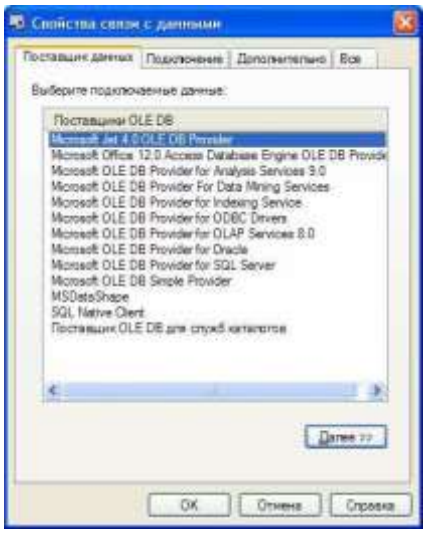

Рисунок 4. Выбор поставщика данных

Нажимаем «Далее» и указываем дополнительные параметры (рисунок 5). Для каждого поставщика настройки могут отличаться.

Если мы указываем просто имя файла, то будет использоваться файл из того же каталога. Так как в базе данных не предусмотрены пользователи и разграничение прав доступа, то мы указываем пользователя по умолчанию (в случае с Access это Admin) и отмечаем «Пустой пароль». На вкладке «Дополнительно» (рисунок 6) отмечаем права доступа «ReadWrite». На вкладке «Подключение» можно проверить, установлено подключение или нет, нажатием на соответствующую кнопку.

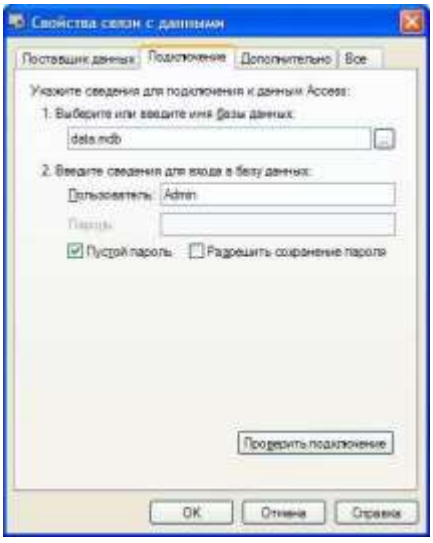

Рисунок 5. Свойства подключения

Рисунок 6. Дополнительные настройки

После этого нажимаем кнопку «ОК». В окне «Источник соединения» строка соединения заполнится, сохраняем ее нажатием кнопки «ОК». После того, как мы сформировали строку соединения, изменяем свойство «LoginPromt» на False. Активируем соединение с базой данных, изменив свойство «Connected» на True.

Рассмотрим настройку компонентов ADOTable и DataSource.

Выберем компонент ADOTable1. В свойстве Connection указываем тот компонент, через который будем подключаться к базе данных, в нашем случае – это «ADOData». В свойстве TableName из списка таблиц выбираем таблицу Grupa. Имя ADOTable1изменяем на tGrupa (имена для ADOTable всех в этом проекте формируется как приставка «t» плюс название таблицы или запроса, к которому подключаем этот компонент). Активируем этот компонент, изменяя свойство «active» на «True».

Выберем компонент DataSource1. В свойстве dataset указываем значение «tGrupa», имя компонента изменяем на «dstGrupa» (имена для всех DataSource формируется как приставка «ds» плюс название набора данных, к которому подключаемся).

В компоненте «tGrupa» двойным щелчком левой кнопки мыши или через контекстное меню открываем редактор полей (fields editor). В появившемся окне, вызвав контекстное меню, выбираем функцию «Add all fields». В список будут добавлены все поля из таблицы или запроса. Теперь у каждого поля в этом списке есть свои свойства, доступные через инспектор объектов. Необходимо для каждого поля изменить свойство «DisplayLabel» на русское название (смотри таблицы 1-5, поле «описание»).

По аналогии изменяем остальные ADOTable и DataSource. В результате у нас должны получиться следующие компоненты:

- tGrupa и связанный с ним dstGrupa;
- tInf и связанный с ним dstInf;
- tTipInf и связанный с ним dstTipInf;
- tParents и связанный с ним dstParents;
- tStudent и связанный с ним dstStudent.

Переходим к разработке графической части.

В базе данных можно выделить две таблицы: grupa и tipinf. Эти таблицы не зависят от других таблиц, и данные в них будут вноситься крайне редко. Условно назовём их «справочниками».

На форме разместим компонент PageControl и компонент StatusBar.

Переименуйте компонент PageControl1 в «pcMain». Свойство align устанавливаем в «alClient». С помощью контекстного меню компонента «pcMain» создаём три вкладки и изменяем их заголовки на «Информация о студентах», «Справочники» и «О программе».

Компонент StatusBar1переименуем в «sb». С помощью редактора панелей этого компонента создаём текстовую панель, которую будем использовать для вывода информации о количестве студентов.

Должна получиться следующая форма (рисунок 7).

На вкладке «Справочники» организуем работу с таблицами «Группы» и «Тип информации». Для остальных таблиц будем использовать вкладку «Информация о студентах».

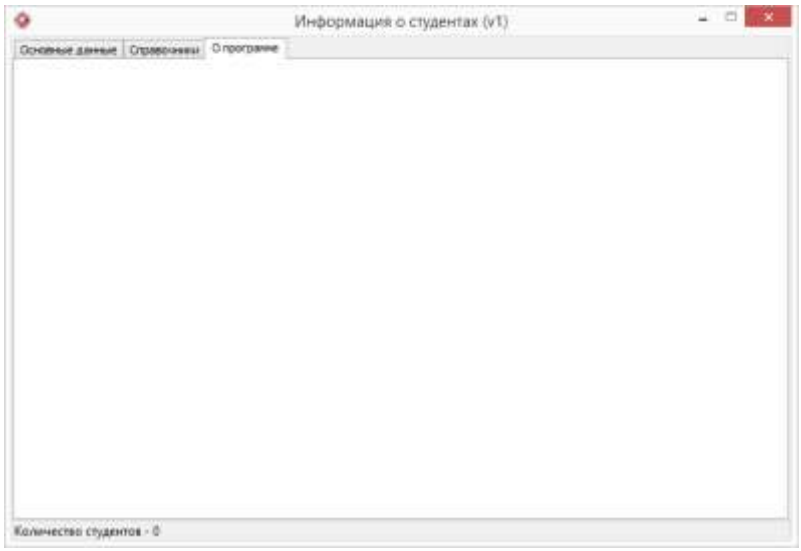

Рисунок 7. Главная форма приложения

На вкладке «Справочники» разместим два компонента «DbGrid» и два компонента «DbNavigator» (рисунок 8). Переименуйте компонента DbGrid'ы в dbgGrupa и dbgTipInf и соответствующие им компоненты DbNavigator'ы – в dbnGrupa и dbnTipInf.

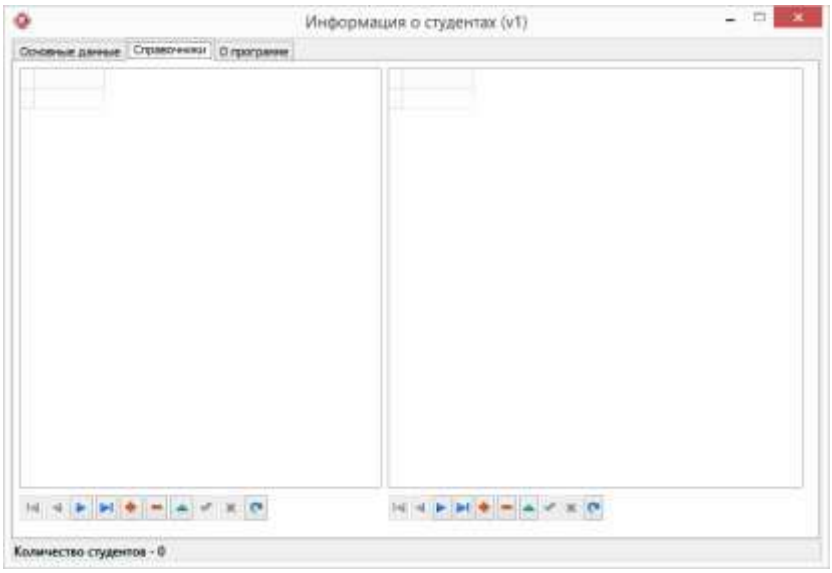

Рисунок 8. Вкладка «Справочники»

Компонент DbGrid используется для просмотра и редактирования информации из набора данных, а также для перемещения по записям набора данных.

Компонент DbNavigator используется для управления информацией из набора данных. По сути, это набор кнопок, которые могут выполнять следующие функции:

 – переход к первой записи (nbFirst);

 – переход к предыдущей записи (nbPrior);

 – переход к следующей записи (nbNext);

 – переход к последней записи (nbLast);

 – вставка пустой записи (nbInsert);

 – удаление выбранной записи (nbDelete);

 – переход в режим редактирования выбранной записи (nbEdit);

 – сохранение данных при добавлении и редактировании (nbPost);

 – отмена, возвращение к предыдущему состоянию (nbCancel);

 – обновить данные (nbRefresh).

Для работы со справочниками нам необходимо оставить только три кнопки: nbInsert, nbPost и nbCancel. Видимость кнопок можно настроить с помощью свойства VisibleButtons. Оставляя только эти кнопки, мы не позволяем удалять данные из справочников.

Настроим компоненты dbgGrupa и dbnGrupa для работы с таблицей «Группы».

В свойстве DataSource этих компонентов выбираем необходимый источник данных (dm.dstGrupa)[1]. Если набор данных активен, то в DbGrid должны отразиться поля набора данных, а DbNavigator активирует только те кнопки, которые можно использовать в данный момент.

Так как поле «код группы» заполняется автоматически, то нет необходимости отображать его в таблице. Также желательно настроить отображение в таблице так, чтобы не было горизонтальной полосы прокрутки. Всё это можно сделать с помощью редактора колонок таблицы, который вызывается с помощью контекстного меню или двойного клика левой кнопки мыши. Используя функцию «Add all fields», добавляем все поля, при этом будут созданы колонки, ассоциируемые с полями набора данных, если необходимо, удаляем лишние колонки и изменяем свойства оставшихся. Например, мы можем установить ширину отображаемой колонки с помощью свойства «Width» в значение 200.

По аналогии настраиваем вторую пару dbgTipInf и dbnTipInf. Используем источник данных dm.dstTipInf. В результате должно получиться следующее (рисунок 9).

[1] Для того чтобы было возможно обратиться к компонентам, находящимся в модуле данных dm, необходимо выполнить следующее: File → Use Unit …, выбрать необходимый модуль в списке и нажать кнопку OK.

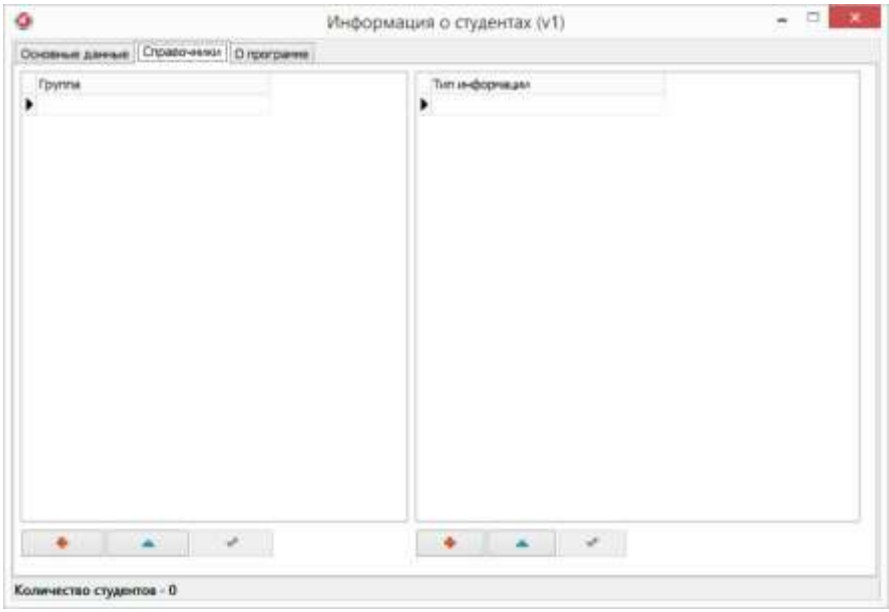

Рисунок 9. Модифицированная вкладка «Справочники»

Сохраняем изменения и запускаем проект. При первом запуске после настройки подключения к базе данных могут появиться несколько (в зависимости от количества наборов данных) сообщений о том, что не найдена база данных. Это связано с тем, что исполняемый файл помещается в каталог «Win32\Debug»[2] в каталоге проекта, и мы используем относительный путь к файлу базы данных. Для того чтобы все заработало, необходимо в этот каталог скопировать файл базы данных.

На вкладке «Справочники» мы можем добавлять и редактировать справочную информацию (рисунок 10). При работе с таблицей необходимо обращать внимание на маркеры строк, которые указывают на режимы работы связанного с ней набора данных: ▶ – режим просмотра; * – режим добавления; I – режим редактирования.

Перейдём к основным данным. К ним мы отнесли информацию о студентах, родителях и контактной информации студентов.

На вкладке «Основные данные» установите и настройте компоненты, как показано на рисунке (рисунок 11).

Компоненты DbGrid и DbNavigator переименовываем в dbgStudent и dbnStudent соответственно.

Компонент PageControl переименуем в pcStudent, в нём создаём 3 вкладки и изменяем заголовки, как показано на рисунке (рисунок 11).

[2] Название зависит от выбранной платформы и параметров конфигурации.

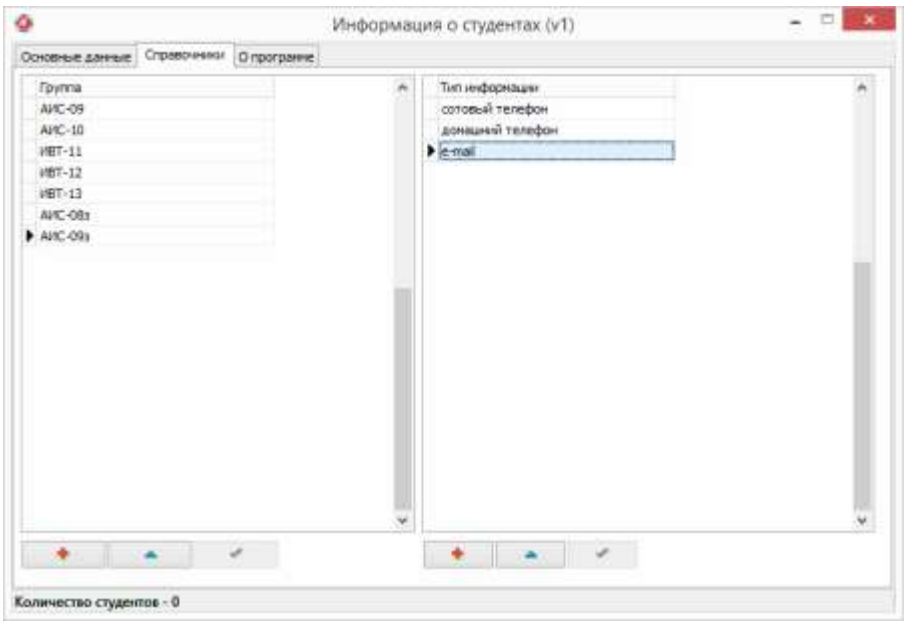

Рисунок 10. Работа со справочной информацией

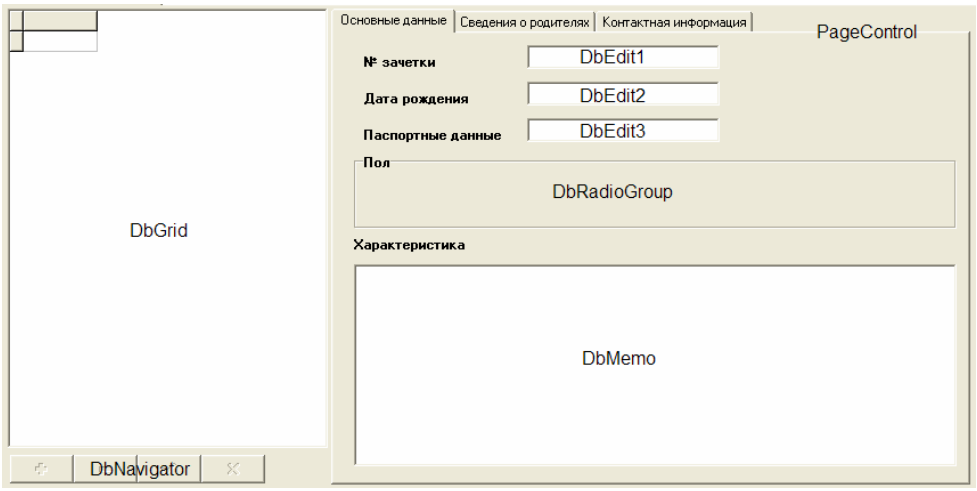

Рисунок 11. Расположение элементов на вкладке «Информация о студентах»

Следующие компоненты устанавливаем на вкладку «Основные данные» pcStudent и переименовываем:

- DbEdit1 – dbeNShifr;
- DbEdit2 – dbeDBirth;
- DbEdit3 – dbePasport;
- DbRadioGroup – dbrgSex;
- DbMemo – dbmResume.

Для заполнения поля idgrupa в таблице student в наборе данных tStudent создадим поле подстановки следующим образом.

Откроем редактор полей набора данных tStudent и в контекстном меню этого окна выберем функцию «New field». В появившемся окне устанавливаем свойства, как показано на рисунке (рисунок 12).

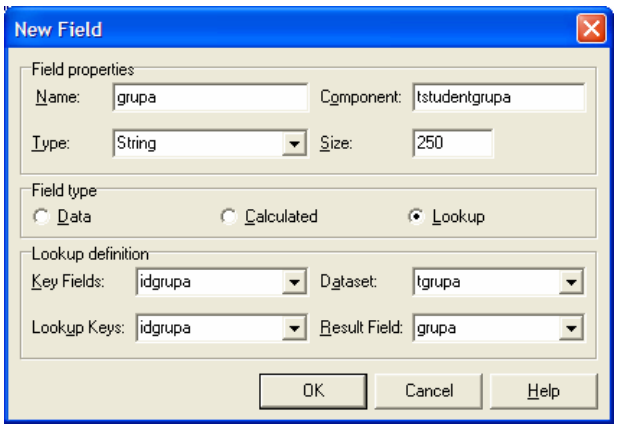

Рисунок 12. Параметры поля подстановки для заполнения idgrupa

Key Fields определяет поле, которое будет заполняться в результате подстановки. Dataset определяет набор данных, откуда берётся информация для подстановки. Lookup Keys определяет, что будет подставляться в Key Fields. Result Field определяет поле, значения которого видит пользователь в выпадающем списке.

Изменяем надпись (свойство Dysplay Label) у нового поля grupa в наборе tstudent на «Группа».

Настраиваем визуальные компоненты. В dbgStudent, dbnStudent, dbeNShifr, dbeDBirth, dbePasport, dbrgSex и dbmResume в свойстве datasource указываем dm.dstStudent. В dbnStudent оставляем видимыми 4 кнопки (nbInsert, nbDelete, nbPost, nbCancel). В dbgStudent отображаем колонки, связанные с полями grupa и fiostudent. Оставшиеся компоненты (dbeNShifr, dbeDBirth, dbePasport, dbrgSex ,dbmResume) связываем с полями в соответствии с названиями (свойство DataField). Особых настроек требует компонент dbrgSex. Помимо установки свойства DataField в значение «sex», в свойстве columns указываем значение 2 (чтобы выводились значения в строку), в свойстве items и values указываем значения «мужской» и «женский» в разных строках (эти значения в items и values могут и не совпадать). В items указываются значения, которые отображаются на форме, в values указываются значения, которые подставляются в таблицу. Запускаем программу и вносим тестовые данные. В результате получаем следующую программу (рисунок 13).

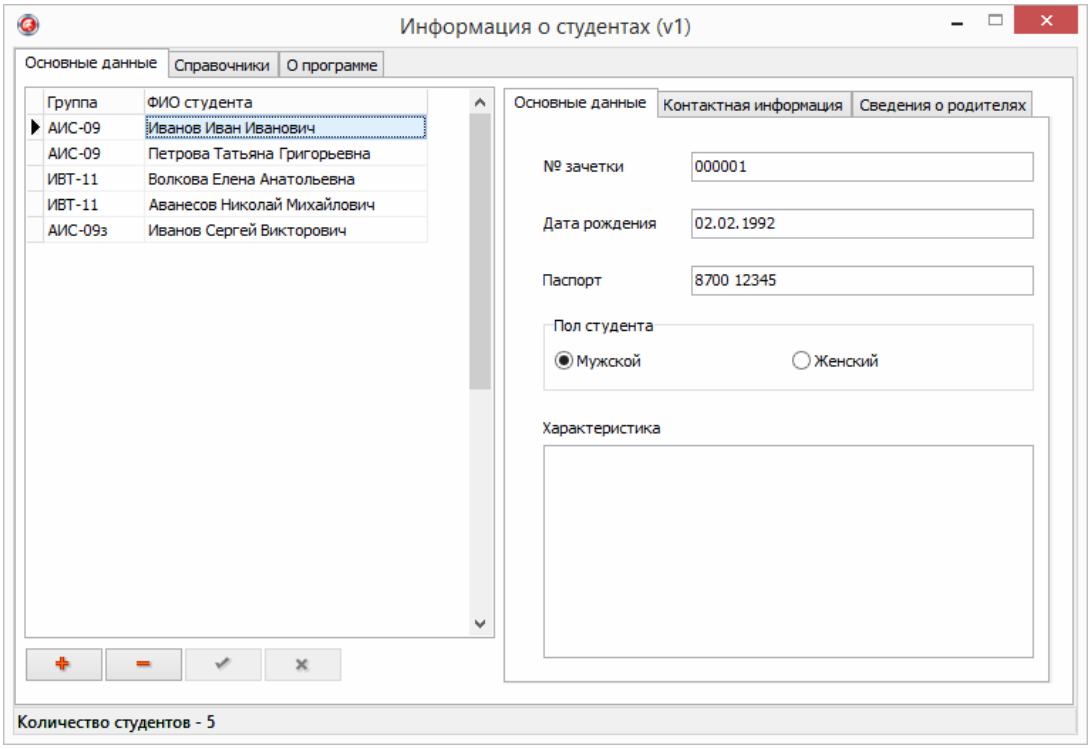

Рисунок 13. Главная форма, вкладка «Основные данные»

Для того чтобы данные в таблицах «родители» и «контактная информация», связанные с таблицей «студент», можно было удобно просматривать и редактировать, выполним некоторые настройки.

Связи между наборами данных устанавливаются через свойство MasterSource компонентов ADOTable. Свяжем их согласно следующей схеме (рисунок 14).

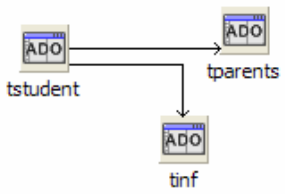

Рисунок 14. Связи между наборами данных

tParents и tInf будут зависеть от tStudent. Связь будет осуществляться по одноимённым атрибутам в наборах данных.

Для установки связи между наборами данных необходимо в свойстве MasterSource зависимого набора данных выбрать источник, к которому подключен главный набор данных, и в свойстве MasterFields указать поля, по которым будет осуществляться связь (рисунок 15).

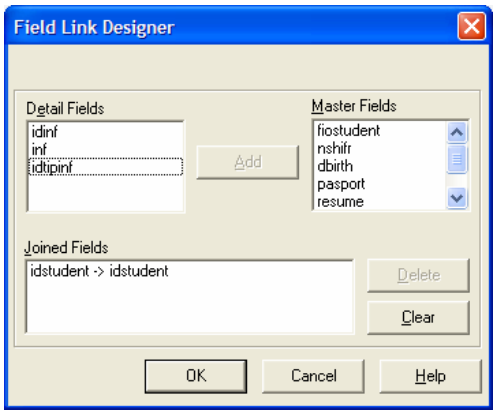

Рисунок 15. Редактор связи MasterFields

Необходимо выбрать нужные поля в 1-м и 2-м списках и нажать кнопку «Add». После этого нажать кнопку «OK».

Благодаря этим связям при выборе какой-либо записи в списке студентов мы увидим только те записи в таблицах «родители» и «контактная информация», которые соответствуют выбранному студенту. При добавлении записей в таблицы «родители» и «контактная информация» эти записи будут добавляться с кодом этого студента, то есть внешние ключи в этих зависимых таблицах будут заполняться автоматически.

Установите на вкладку «Сведения о родителях» компоненты DbNavigator и DbGtrlGrid.

Компонент DbNavigator переименуйте в dbnParents, подключите его к источнику данных dm.dstParents, оставьте видимыми 4 кнопки, как показано на рисунке (рисунок 16), свойство align установите в alBottom.

Компонент DbGtrlGrid переименуйте в dbcgparents, подключите его к источнику данных dm.dstParents, свойство align установите в alClient. Этот компонент для каждой записи создаёт определённый пользователем набор компонентов.

Нам понадобятся следующие компоненты:

- dbComboBox – для заполнения степени родства значением из выпадающего списка;
- dbEdit – для заполнения Ф.И.О. родителя;
- dbMemo – для заполнения информации о родителе.

Эти компоненты, на этапе разработки, помещаются в незаштрихованную панель. Разместите надписи и компоненты для просмотра информации из базы данных в эту панель и переименуйте их согласно рисунку (рисунок 16).

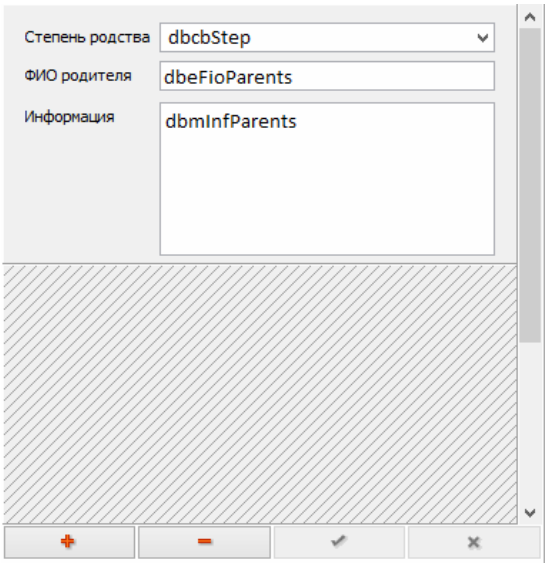

Рисунок 16. Главная форма. Вкладка «Информация о родителях»

Подключите эти компоненты к соответствующим полям в наборе данных «tParents» с помощью свойства DataField.

Для компонента «dbcbStep» в свойстве «Items» укажите значения, которые будут появляться в выпадающем списке. В нашем случае это будут: мать, отец, опекун.

После этого сохраняем изменения, запускаем проект и вносим данные о родителях уже введённых студентов (рисунок 17).

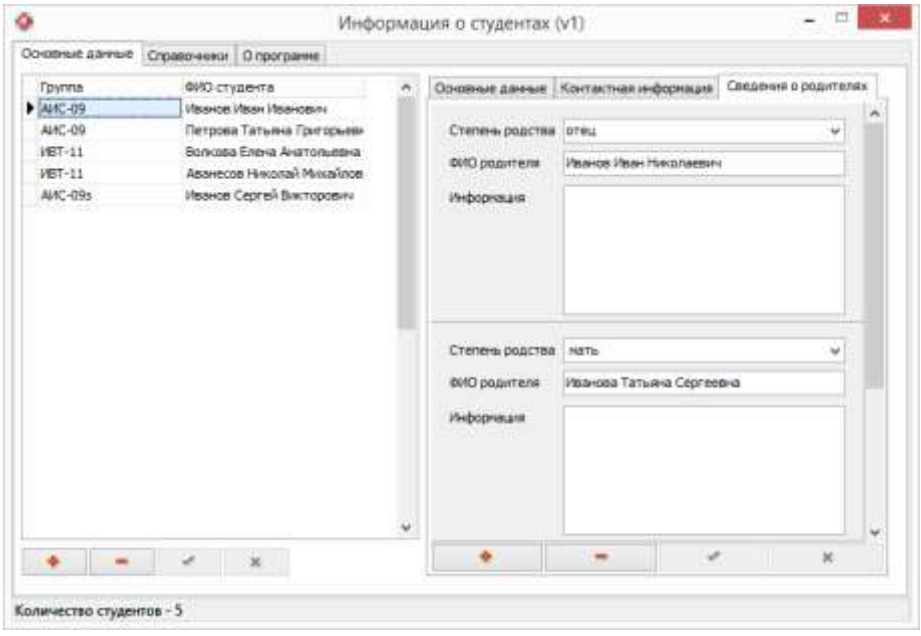

Рисунок 17. Заполнение сведений о родителях

Для заполнения поля idtipinf в tInf самостоятельно создайте поле подстановки, назовите его tipinf, а свойство DysplayLabel измените на «Тип информации». На вкладке «Контактная информация» разместите DbGrid и DbNavigator, назовите их dbgInf и dbnInf соответственно. Свяжите их с источником данных dm.dstInf. Разместите и настройте их, как показано на рисунке (рисунок 18).

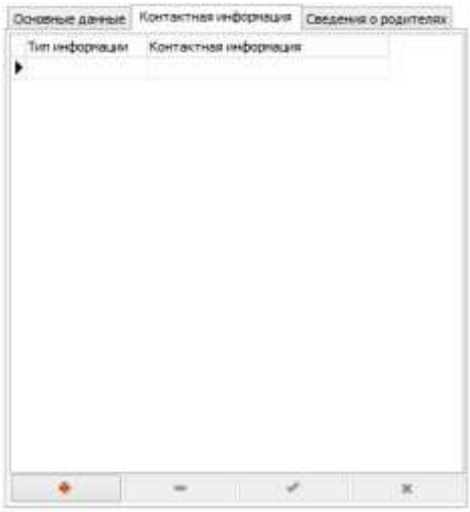

Рисунок 18. Главная форма. Вкладка «Контактная информация»

Сохраняем изменения, запускаем проект и вносим контактную информацию уже введённых студентов (рисунок 19).

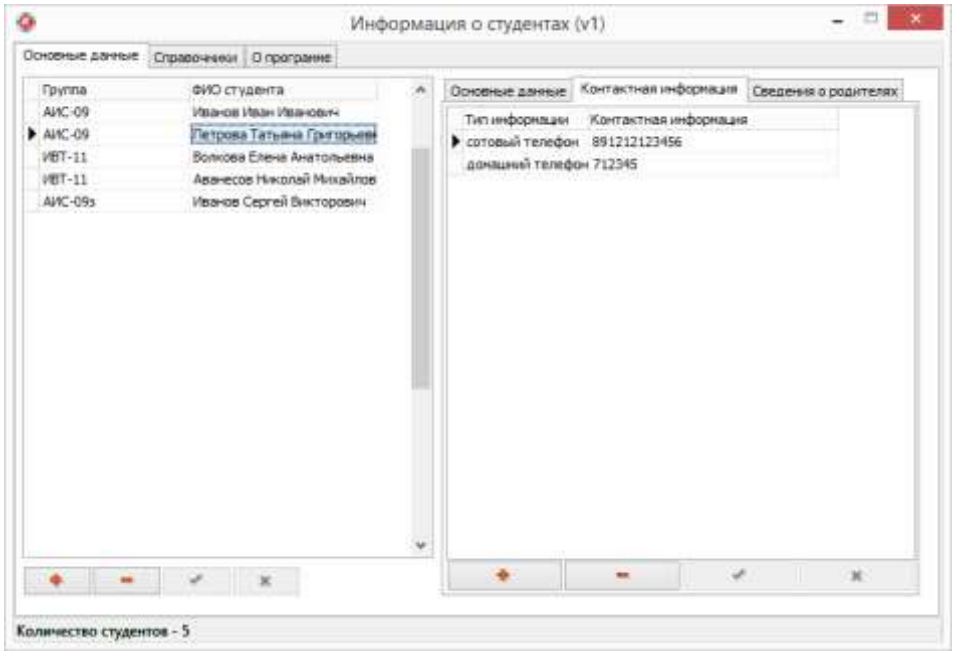

Рисунок 19. Заполнение контактной информации

На вкладку «О программе» поместите компонент Memo. Измените свойства Align и Alignment на alClient и taCenter соответственно. В свойстве Lines оставим описание программы. В результате должно получиться следующее (рисунок 20).

На этом разработка графической части проекта закончена. Необходимо вывести информацию о количестве студентов в строке статуса и сделать так, чтобы при загрузке программы были активны вкладки «Информация о студентах» и «Основные данные». Для этого на событие onActivate формы fMain создадим процедуру и в ней запишем следующий код.

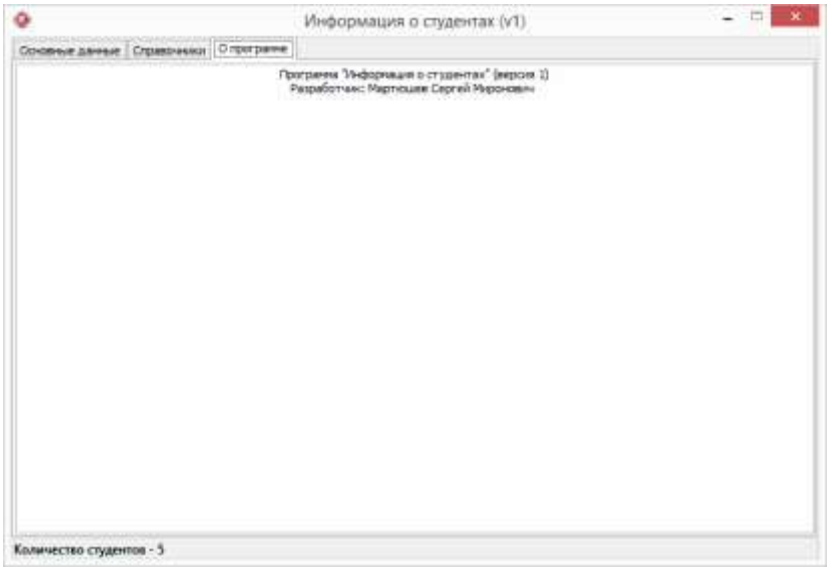

Рисунок 20. Вкладка «О программе»

```
 pcMain.ActivePageIndex:=0;
pcStudent.ActivePageIndex:=0;
fmain.sb.Panels[0].Text:='Количество студентов -
'+inttostr(dm.tstudent.RecordCount);
```

Для того чтобы при добавлении студентов и их удалении количество студентов изменялось, создадим процедуры на события AfterPost и AfterDelete набора данных tStudent и вставим в них:

```
fmain.sb.Panels[0].Text:='Количество студентов -
'+inttostr(dm.tstudent.RecordCount);
```

На этом разработка первой версии программы «Информация о студентах» закончена.

Для использования программы необходимо исполняемый файл stud_v1.exe и файл базы данных «data.mdb» скопировать в один отдельный каталог.

Протестируйте работу приложения. Проверьте, как изменяется значение счётчика в строке состояния при добавлении и удалении данных. Посмотрите, что происходит с данными в таблицах базы данных при работе с программой.

Лабораторная работа №3 Разработка 2-й версии программы. Модификация базы данных

В результате тестирования первой версии программы можно выявить ряд недостатков:

- информация не отсортирована;
- сложно контролировать целостность ввода данных;
- возможны ошибки при добавлении информации в зависимые таблицы, например при добавлении студента сохраняется возможность добавить данные в зависимые таблицы, даже если данные о студенте не сохранены;
- отсутствует возможность быстрого поиска и формирования отчётности.

Для устранения этих недостатков приступим к разработке 2-й версии программы. Разрабатывать вторую версию будем на основе первой версии.

Выполним следующие действия:

1) создаем каталог «stud_v2», в нём будем хранить все файлы проекта;
2) копируем все файлы и каталоги из каталога «stud_v1» в «stud_v2»;
3) копируем файл базы данных «data.mdb» из «Win32\Debug» в «stud_v2», заменяя ранее существующий файл;
4) открываем файл проекта stud_v1.dpr из каталога «stud_v2», сохраняем проект под именем «stud_v2» (File→Save Project As …);
5) запускаем проект, проверяем работоспособность и закрываем программу;
6) закрываем среду разработки и из каталога «stud_v2» и подкаталога «Win32\Debug» удаляем все файлы с именем stud_v1.

Для работы над второй версией программы нам потребуется немного модифицировать базу данных. Главное, не запутайтесь, какой файл базы данных вы модифицируете.

После вышеизложенных манипуляций в каталоге «stud_v2» находится тот файл базы данных, который использовался при тестировании 1-й версии. Его и будем модифицировать, для этого откроем его в СУБД Access.

Пол студента может принимать только два значения, поэтому нет необходимости использовать текстовое поле для хранения этой информации. Для этих целей будем использовать логический тип данных. Договоримся о том, что значение TRUE или 1 означает мужской пол, а FALSE или 0 – женский пол. Пока данных не так много, есть возможность сделать такие изменения вручную.

Откройте для редактирования таблицу student. В поле sex вместо значения «мужской» поставьте значение «1», а вместо значения «женский» – значение «0». Закройте таблицу и откройте её же в режиме конструктора. Измените тип данных поля sex с «текстовый» на «логический», сохраните изменения, согласитесь с предупреждением о том, что некоторые данные могут быть потеряны и закройте таблицу.

Для более наглядного представления информации и возможности организовать быстрый поиск создадим запросы, которые будем использовать для просмотра и редактирования информации в таблицах.

Так как мы ещё не очень хорошо знаем язык структурированных запросов (SQL), то будем создавать запросы с помощью встроенного конструктора. Нам необходимо создать следующие запросы:

- qStudent – основные персональные данные о студентах (рисунок 21);
- qInf – контактная информация студентов (рисунок 22);
- qParents – информация о родителях студентов (рисунок 23);
- qGrupa – информация об учебных группах (рисунок 24);
- qTipInf – информация о типах контактной информации (рисунок 25).

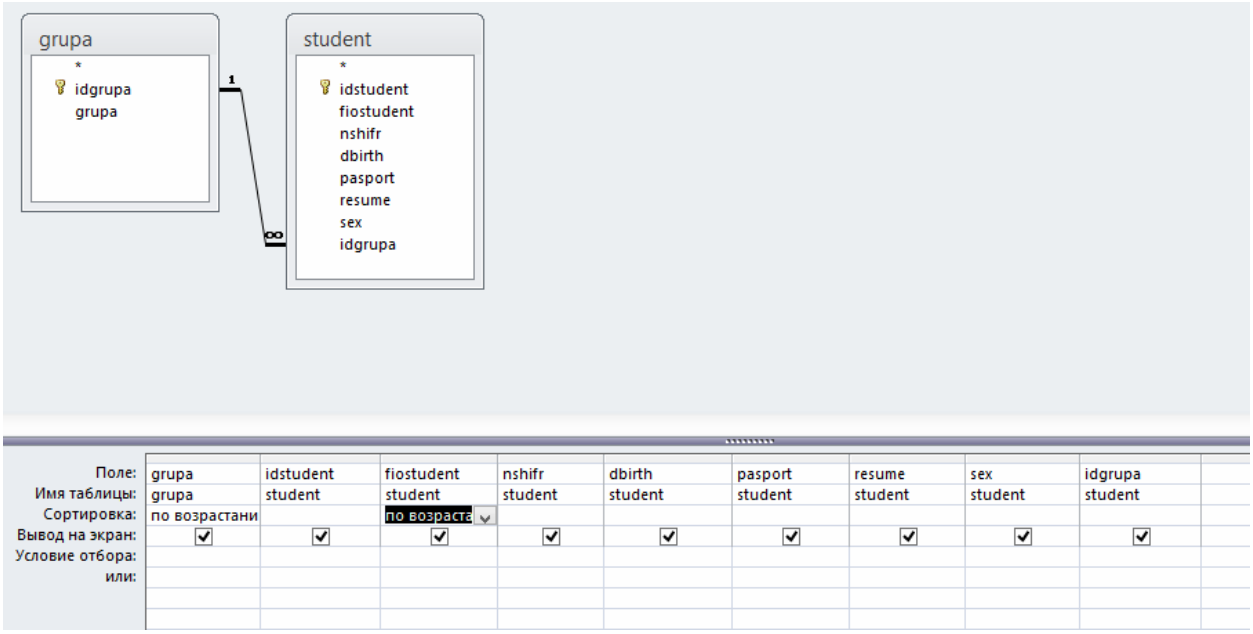

Рисунок 21. Запрос qStudent

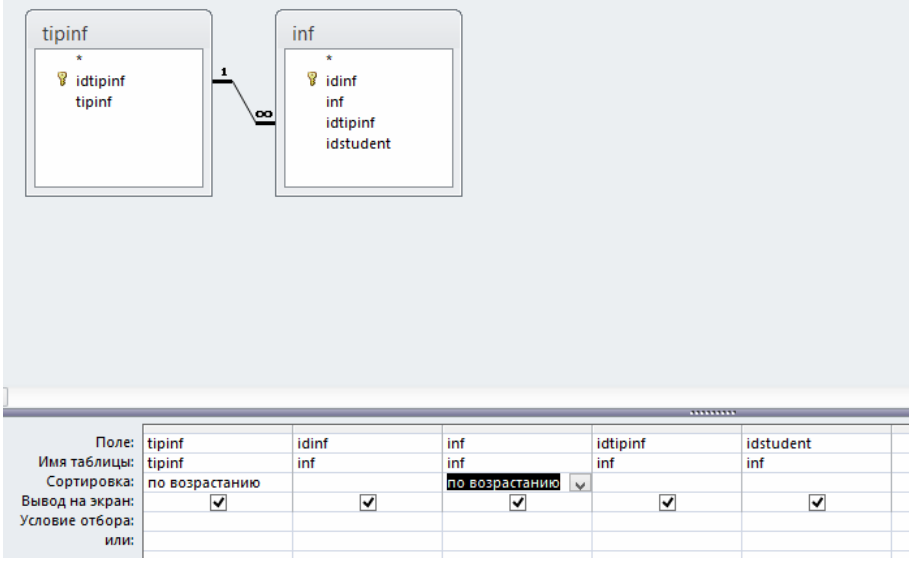

Рисунок 22. Запрос qInf

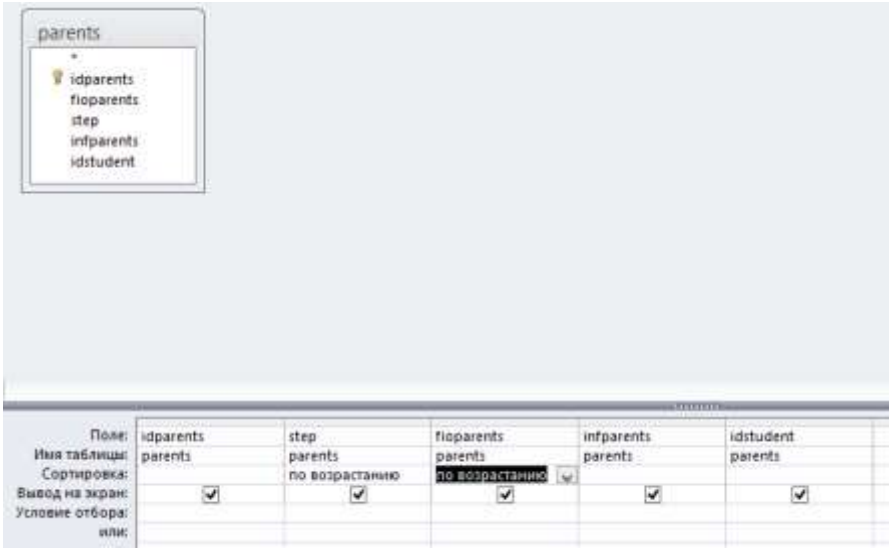

Рисунок 23. Запрос qParents

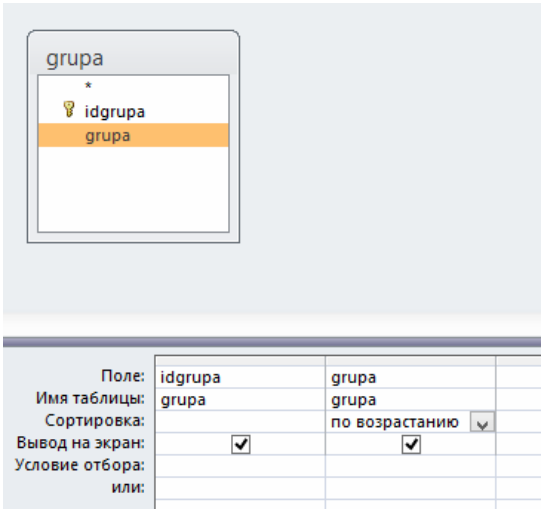

Рисунок 24. Запрос qGrupa

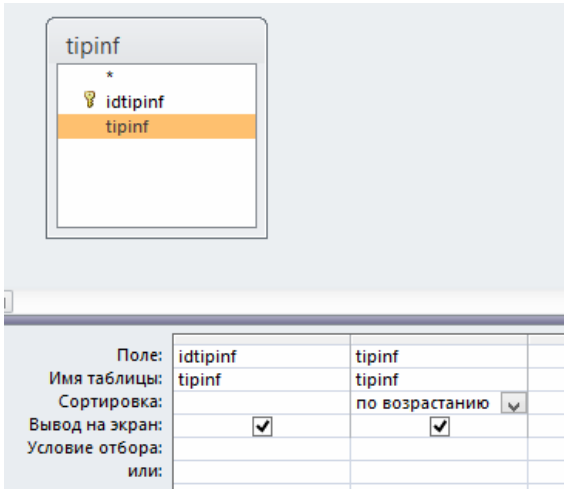

Рисунок 25. Запрос qTipInf

Имя запроса формируется по следующему принципу. Приставка q означает query (запрос), далее идёт имя таблицы, для которой предназначен этот запрос. При составлении запросов для нашей программы важно, чтобы все ключевые поля были из таблицы, для которой этот запрос создаётся. Порядок полей в запросе влияет на порядок сортировки.

После создания запросов можно посмотреть, что будет выводиться в результате, просто открыв их.

Модификация базы данных завершена. СУБД Access можно закрыть.

Изменённый файл базы данных из каталога «stud_v2» копируем в «Win32\Debug», заменяя файл.

Лабораторная работа №4 Просмотр основных данных. Реализация быстрого поиска. Работа со справочной информацией

Переходим к разработке приложения.

Открываем проект «stud_v2».

Так как в файле базы данных были сделаны изменения в структуре таблицы «Student», при открытии проекта будет выведено сообщение об ошибке в поле «Пол». Соглашаемся с замечаниями.

В проекте открываем модуль данных. Изменяем настройки наборов данных следующим образом:

1) свойство «Active» всех наборов данных устанавливаем в False;
2) в именах всех наборов данных вместо приставки t ставим q (например было tStudent стало qStudent);
3) в именах источников данных приставку dst меняем на dsq (например было dstInf стало dsqInf);

4) в редакторе полей наборов данных qStudent и qInf удаляем все поля и добавляем их снова (это необходимо, потому что уже нет необходимости в полях подстановки, а в таблице Student, к тому же, была изменена структура);
5) изменяем свойство «DisplayLabel» у полей наборов данных qStudent и qInf;
6) активируем все наборы данных;
7) добавим компонент ADOQuery. Он нам понадобится для удаления информации. Назовём его qDel и свяжем его с ADOData через свойство Connection.

В проекте делаем следующие изменения:

1) в заголовке главной формы изменяем номер версии;
2) изменяем номер версии на вкладке «О программе»;
3) удаляем программный код из процедур на события «AfterDelete» и «AfterPost» набора данных «qStudent»;
4) изменяем строку кода, которая выводит количество записей в строку состояния (StatusBar) на главной форме в процедуры на событие OnActivate главной формы на «*fmain.sb.Panels[0].Text:='Количество студентов - '+inttostr(dm.qStudent.RecordCount);*»;
5) в компоненте dbrgSex в свойстве values заменяем значения «Мужской» на «True» и «Женский» – на «False».

Сохраняем изменения и запускаем проект (рисунок 26).

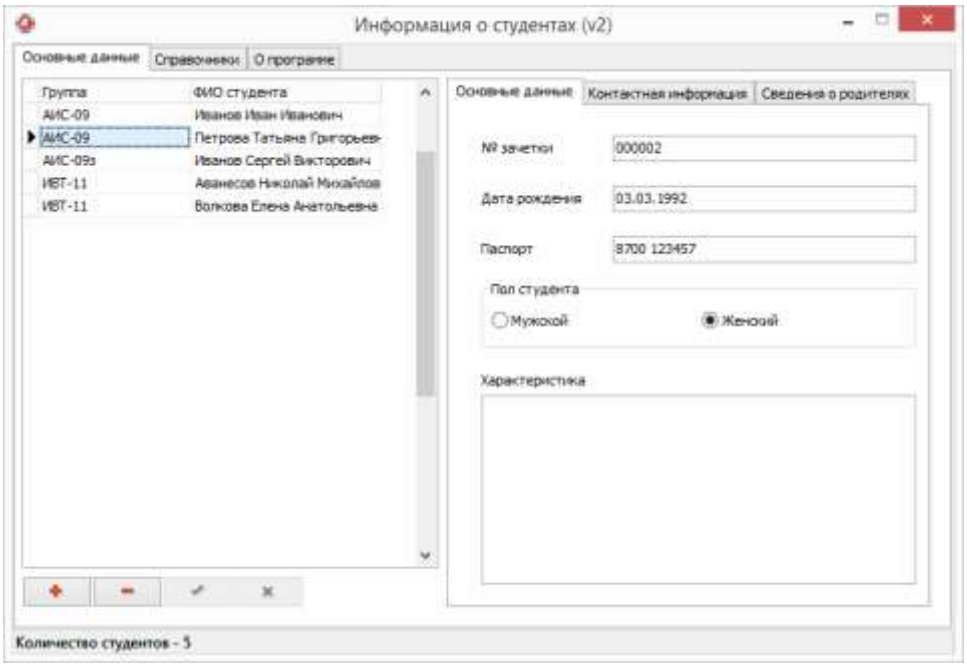

Рисунок 26. Результат запуска программы

Если вы попытаетесь сделать некоторые изменения, может произойти ошибка, поэтому в этом варианте второй версии мы проверяем только вывод информации.

В графическом интерфейсе программы будет сделан ряд изменений.

Для начала разместим информацию о программе на отдельной форме.

Добавляем в проект новую форму. Переименовываем её в «fAbout». Для неё устанавливаем следующие свойства:

- position – poDesktopCenter;
- borderstyle – bsSingle;
- bordericons – [biSystemMenu];
- caption – О программе.

Сохраняем изменения во всём проекте и модуль для новой формы сохраняем по именем «ufAbout».

У компонента с вкладки «О программе» главной формы свойство «ReadOnly» выставите в True. Этот компонент переместите, используя буфер обмена на форму fAbout.

Чтобы форма сразу не создавалась при запуске программы, а только тогда, когда это необходимо, поместим форму в группу возможных. Для этого в опциях проекта (Project→Options) в разделе «Forms» переместите fAbout из Auto-create forms в Available forms.

Со всеми остальными формами мы будем поступать аналогичным образом. Как вызывать такие формы, рассмотрим далее.

На главной форме будет отображаться только содержимое вкладки «Основные данные». Работа с остальными данными и реализация функций добавления, редактирования и удаления будет осуществляться через отдельные формы. Для доступа к ним будем использовать главное и контекстное меню. Впоследствии можно организовать доступ к функциям через панель инструментов. Чтобы имелась возможность достаточно просто привязывать функции к графическим элементам и их событиям, будем использовать ActionList (Список действий).

Добавим в модуль данных компонент ActionList и переименуем его в «al».

В дальнейшем весь программный код (функционал) будет организован через действия.

Доступ к действиям, их свойствам и событиям осуществляется через ActionList. Действия можно группировать по категориям. Программный код действия записывается в процедуре на событие OnExecute. Действие может быть привязано к основному событию визуального компонента через свойство Action или может быть вызвано внутри процедуры или функции.

То есть мы должны так организовать основной программный код, чтобы он весь был представлен через действия и при необходимости привязывался к визуальному компоненту, если это возможно, или вызвался в процедурах или функциях.

Для начала необходимо реализовать события в al для тех процедур, которые уже реализованы в программе.

Открываем ActionList Editor компонента al с помощью контекстного меню. В списке actions (действия) создаём два новых действия Action1 и Action2. Action1 переименуем в «fMainActivate», свойство «Category» изменим на «fMain», свойство «Caption» изменим на «Активация формы». Action2 переименуем в «fMainStatus», в свойстве «Category» выберем «fMain», свойство «Caption» изменим на «Обновление строки статуса».

Создаём процедуру для обработки события OnExecute действия «fMainStatus». В процедуру вставляем следующий программный код.

```
fmain.sb.Panels[0].Text:='Количество студентов -
'+inttostr(dm.qStudent.RecordCount);
```

Создаём процедуру для обработки события OnExecute действия «fMainActivate». В процедуру вставляем следующий программный код.

```
fmain.pcStudent.ActivePageIndex:=0;
fMainStatus.Execute;
```

Строка «fMainStatus.Execute;» и отвечает за выполнение действия внутри другой процедуры или функции.

Создаём процедуру на событие OnActivate формы fMain. Прописываем в ней следующий код.

```
dm.fMainActivate.Execute;
```

Теперь, если захотим что-либо добавить к событию на активацию, мы это сделаем через соответствующее действие.

Сохраняем изменения и запускаем проект. Если все работает как и раньше, значит, вы всё сделали правильно.

Добавляем в модуль данных компонент для создания главного меню и переименуем его в mmfMain.

Сформируйте главное меню следующей структуры (рисунок 27).

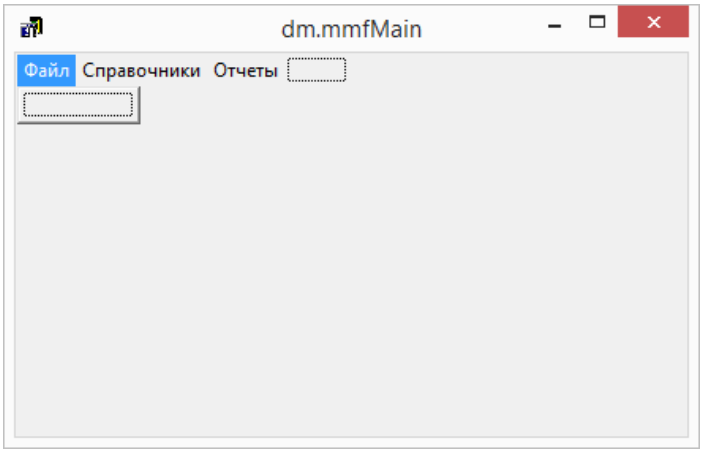

Рисунок 27. Структура главного меню программы

Подключите главное меню к главной форме. Для этого в свойстве формы «Menu» выберите нужный элемент. Если необходимо, измените размеры формы.

Создадим действие с именем «fMainClose» и заголовком «Выход», а также действие с именем «fAboutOpen» и заголовком «О программе». Первое поместите в категорию «fMain», для второго укажите категорию «fAbout».

Для действия «fMainClose» пропишите следующий код.

```
fMain.Close;
```

Для действия «fAboutOpen» укажите программный код.

```
Application.CreateForm(TfAbout,fAbout);
fAbout.ShowModal;
```

Прежде чем использовать форму, которая находится в категории возможных, её необходимо создать с помощью Application.CreateForm. Для того чтобы не возникало ошибок с Application, необходимо в раздел Uses добавить библиотеку «Vcl.Forms». Чтобы не возникало ошибок с fAbout, необходимо эту форму сделать доступной для модуля данных.

Настраиваем главное меню. Открываем его в модуле данных. Для первого подпункта раздела «Файл» в свойстве «Action» выбираем «fAboutOpen»; для второго в свойстве «Caption» указываем «-», чтобы сделать разделитель; для третьего в свойстве «Action» выбираем «fMainClose». В результате должно получиться следующее (рисунок 28).

Сохраните изменения и запустите программу.

Все элементы с вкладки «Справочники» необходимо удалить. Работу со справочниками организуем через отдельную форму. Так как наши справочники имеют относительно простую структуру, можно организовать работу с ними через одну форму, при необходимости указывая нужные параметры.

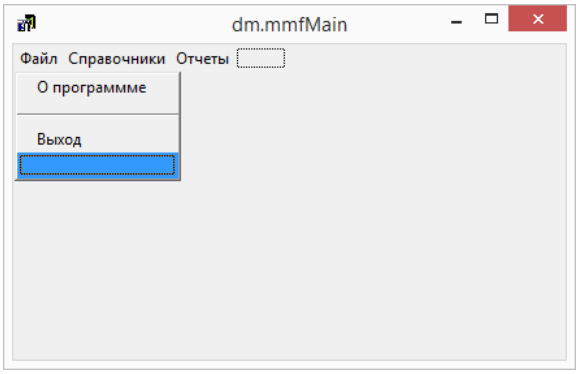

Рисунок 28. Добавленные пункты «О программе» и «Выход»

Справочниками мы считаем таблицы grupa и tipinf. Редактирование этих таблиц осуществляется редко, их структура проста и похожа. В обеих таблицах мы работаем с одним полем, ключевое заполняется автоматически. Для работы со справочниками мы будем использовать одну форму и использовать такие же элементы управления, как и в первой версии программы.

Добавляем в проект новую форму. Переименуем её в «fSpr», в каталог проекта сохраняем её под именем «ufSpr», остальные настройки формы совпадают с настройками формы «О программе». Не забудьте перенести её в категорию возможных в свойствах проекта!

На форму устанавливаем dbgrid, dbnavigator и statusbar. Компоненты переименовываем в «dbg», «dbn» и «sb» соответственно. У компонента «dbn» оставляем видимыми три кнопки (также, как и при работе со справочниками в первой версии программы), у компонента «sb», который будем использовать для вывода информации о количестве записей в редакторе панелей, добавьте текстовую панель. Свойство Align компонента dbn необходимо выставить в alBottom (выравнивание по нижнему краю), а у компонента dbg – в alClient. В результате должно получиться, как на рисунке (рисунок 29).

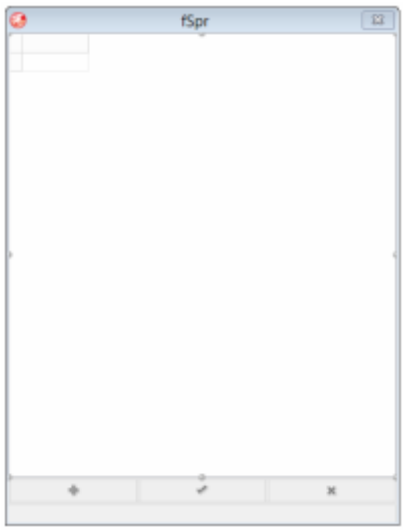

Рисунок 29. Дизайн формы fSpr

Для возможности обращения к новой форме из модуля данных сделаем форму fSpr доступной, используя функцию «Use Unit».

При работе со справочниками пользователь может закрыть форму fSpr, не завершив операцию добавления или редактирования, при этом набор данных не будет находиться в режиме просмотра, что может привести к ошибкам при работе с информацией на других формах. Для разрешения этой ситуации создадим действие, которое будет, при закрытии формы fSpr, проверять, находится ли набор данных, с которым работаем в данный момент времени, в состоянии редактирования или добавления, и если да, то автоматически будет выполнена отмена, и набор данных при закрытии fspr перейдёт в режим просмотра.

Создаём новое действие в «al», переименуем его в fSprOnClose, в категории укажем fSpr. Создаём процедуру на событие onExecute и пропишем в ней следующий программный код.

```
if (fspr.dbg.DataSource.DataSet.State=dsEdit) or
(fspr.dbg.DataSource.DataSet.State=dsInsert) then
fspr.dbg.DataSource.DataSet.Cancel;
```

Значение свойства state dsEdit определяет, что набор данных находится в режиме редактирования, а dsInsert – в режиме добавления.

На событие onClose формы fSpr вызовем действие fSprOnClose, прописав этот вызов в процедуре. Для этого может потребоваться сделать модуль данных доступным для формы fSpr.

Так как вывод количества записей в справочнике в строке статуса формы необходимо будет делать при открытии справочника, при добавлении и редактировании записей в справочниках создадим отдельное действие fSprStatus в категории fSpr и в процедуре для этого действия напишем следующий код.

```
fspr.sb.Panels[0].Text:='Количество записей - ' +
inttostr(fspr.dbg.DataSource.DataSet.RecordCount);
```

Обратите внимание на то, каким образом мы обращаемся к наборам данных.

Организуем работу со справочником «Группы». Для этого создадим новое действие в категории fSpr, назовём его «fSprGrupaOpen», а в свойстве caption укажите «Учебные группы». Создаём процедуру на событие onExecute, содержащую следующий код.

```
Application.CreateForm(TfSpr,fSpr);
fSpr.Caption:='Учебные группы';
```

```
fSpr.dbg.DataSource:=dm.dsqGrupa;
fSpr.dbn.DataSource:=dm.dsqGrupa;

fSpr.dbg.Columns.Clear;
fSpr.dbg.Columns.Add;
fSpr.dbg.Columns[0].Field:=dm.qGrupaGrupa;
fSpr.dbg.Columns[0].Width:=350;

fSpr.Width:=350+60;
fSprStatus.Execute;
fSpr.ShowModal;
```

Для того чтобы при добавлении данных в справочник изменялась информация о количестве записей и при изменении данных в справочнике изменялась информация в запросе, связанном со справочником, и названия групп в запросе qStudent, создадим действие qGrupaRefresh и укажем категорию qGrupa. Создаём процедуру на событие onExecute, содержащую следующий код.

```
qGrupa.Close;
qGrupa.Open;
qStudent.Close;
qStudent.Open;
fSprStatus.Execute;
```

Так как удаление в справочниках запрещено, то обновление будет происходить только на событие AfterPost. Организуем вызов данного действия в процедуре на это событие в наборе данных qGrupa.

В подпункте меню «Справочники» в свойстве Action выберите действие fSprGrupaOpen.

Сохраните изменения и запустите проект. Попробуйте добавить новые группы и изменить названия уже использованных групп и посмотрите, как отреагирует на это программа. Если все получилось, организуйте работу со вторым справочником TipInf.

Должно получиться следующее.

```
procedure Tdm.fSprTipInfOpenExecute(Sender: TObject);
begin
```

```
  Application.CreateForm(TfSpr,fSpr);
  fSpr.Caption:='Тип информации';

  fSpr.dbg.DataSource:=dm.dsqTipInf;
  fSpr.dbn.DataSource:=dm.dsqTipInf;

  fSpr.dbg.Columns.Clear;
  fSpr.dbg.Columns.Add;
  fSpr.dbg.Columns[0].Field:=dm.qTipInftipinf;
  fSpr.dbg.Columns[0].Width:=350;

  fSpr.Width:=350+60;
  fSprStatus.Execute;
  fSpr.ShowModal;
end;

procedure Tdm.qTipInfRefreshExecute(Sender: TObject);
begin
  qTipInf.Close;
  qTipInf.Open;

  qInf.Close;
  qInf.Open;

  fSprStatus.Execute;
end;

procedure Tdm.qTipInfAfterPost(DataSet: TDataSet);
begin
qTipInfRefresh.Execute;
end;
```

В результате проделанной работы в нашей программе появилась возможность работы со справочниками, со справочником «Учебные группы» (рисунок 30) и «Тип контактной информации» (рисунок 31).

Рисунок 30. Справочник «Учебные группы»

Рисунок 31. Справочник «Тип контактной информации»

Приступаем к проработке графического интерфейса главной формы.

Размещать информацию на главной форме мы будем по аналогии со вкладкой «Основные данные». При разработке графического интерфейса главной формы следует учесть следующее:

- пользователь не должен иметь возможность редактировать информацию на главной форме;
- пользователь может менять размер формы;
- должна быть предусмотрена возможность быстрого поиска по ФИО студента и названию группы.

Удаляем компонент «pcMain». В качестве примера можно использовать первую версию программы.

Для того чтобы пользователь не мог редактировать данные, мы будем использовать компоненты dbText там, где это возможно, или компоненты, которые использовали ранее, но свойство ReadOnly будем устанавливать в True. Для того чтобы наш интерфейс подстраивался под размеры формы, там, где это возможно, будем использовать свойство align или будем использовать компонент GridPanel.

Устанавливаем на форму компонент GridPanel. Свойство Caption очищаем, в свойстве Align выбираем alClient. Сам компонент переименовываем в grpMain.

GridPanel – это сетка разметки, состоящая из любого количества колонок и строк. По умолчанию она состоит из 2-х строк и 2-х колонок. Для каждой строки или столбца можно задать один из трёх способов определения размера:

абсолютное значение (ssAbsolut), задаётся в пикселях; в процентах от основного размера (ssPercent); автоматически по содержимому (ssAuto). Эти параметры задаются в свойствах ColumnCollection и RowCollection для каждой строки или колонки, там же задаётся количество колонок и строк. Параметры строк или столбцов лучше менять с конца списка. При установке размера через процент значение может не сразу установиться в нужное, может потребоваться сделать это несколько раз.

В ячейке можно разместить 1 графический компонент, в том числе и какой-либо контейнер. При установке компонентов в GridPanel они размещаются, начиная с левой верхней ячейки слева направо и сверху вниз.

Для компонента можно задать, в какой колонке и строке он будет находиться (свойство Row и Column), нумерация начинается с нуля. Компонент может занимать несколько строк и/или столбцов (свойство RowSpan и ColumnSpan).

Для grpMain установим следующие параметры.

Открываем коллекцию строк (RowCollection). Увеличиваем количество строк до 7. Для них устанавливаем следующие параметры.

Таблица 6. Параметры RowCollection для grpMain

Member	Size Type	Value
Row0	Absolute	10
Row1	Percent	50% (примерно)
Row2	Absolute	10
Row3	Absolute	22
Row4	Absolute	10
Row5	Percent	50% (примерно)
Row6	Absolute	10

Открываем коллекцию колонок (ColumnCollection). Добавляем ещё три столбца. Для них устанавливаем следующие параметры.

Таблица 7. Параметры ColumnCollection для grpMain

Member	Size Type	Value
Column0	Absolute	10
Column 1	Percent	30% (примерно)
Column 2	Absolute	10
Column 3	Percent	70% (примерно)
Column 4	Absolute	10

Устанавливаем компоненты в grpMain.

Устанавливаем компонент ComboBox. Он должен появиться в левом верхнем углу. Для этого компонента устанавливаем следующие свойства:

- Column – 1;
- Row – 1;
- ColumnSpan – 3;
- Align – alTop;
- Name – cbSearch;
- Text очищаем.

После этого для Row1 в коллекции строк можно установить параметр Auto.

Выделяем grpMain, устанавливаем в него DBGrid. Он также должен появиться в левом верхнем углу. Для него устанавливаем следующие свойства:

- Column – 1;
- Row – 3;
- RowSpan – 3;
- Align – alClient;
- Name – dbgStudent.

Устанавливаем в grpMain PageControl. Для него устанавливаем следующие свойства:

- Column – 3;
- Row – 5;
- Align – alClient;
- Name – pcStudent.

Устанавливаем в grpMain dbText. Для него устанавливаем следующие свойства:

- Column – 3;
- Row – 3;
- Align – alTop;
- Name – dbtGrupaFio.

Теперь можно изменить некоторые параметры формы, а именно, BorderStyle устанавливаем в bsSizeable, BorderIcons – в [biSystemMenu,biMinimize,biMaximize] и WindowState – в wsMaximized.

Сохраните изменения и запустите программу. Проверьте, как изменяются размеры при изменении размеров окна.

Настройте компонент dbgStudent так, как он был настроен до удаления pcMain. В дополнение свойство ReadOnly устанавливаем в True, а свойство dgEditing (группа свойств Options) устанавливаем в False.

В pcStudent создаём 3 вкладки и изменяем заголовки, как показано на рисунке (рисунок 32).

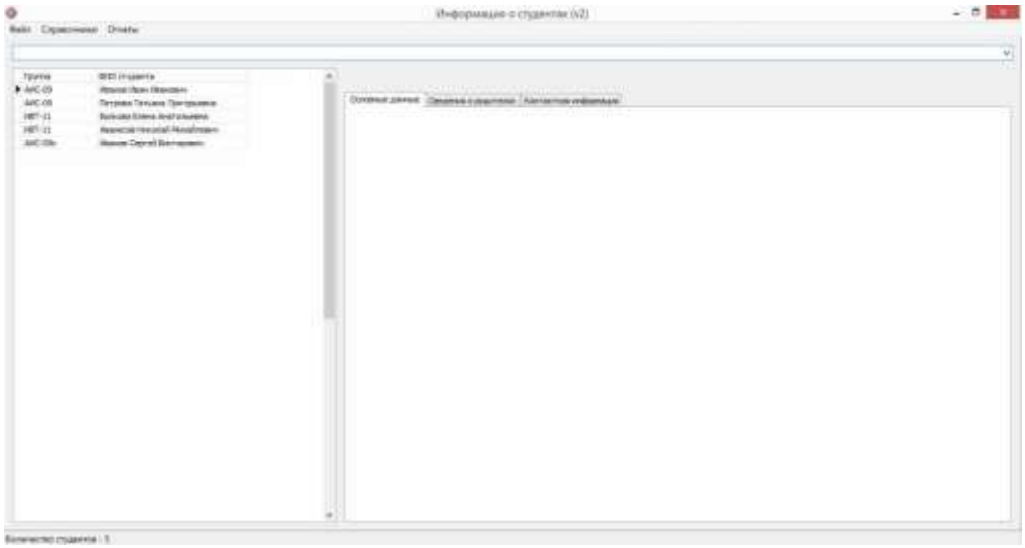

Рисунок 32. Главная форма

Иногда, для подведения каких-либо итогов или повышения наглядности, необходимо выполнять дополнительные операции с информацией из таблиц или запросов базы данных. Например, подсчитать итоговую сумму, зная количество и стоимость, или сложить вместе несколько текстовых значений для того, чтобы вся информация была в одном поле. Для решения этой задачи используются вычисляемые поля.

В нашем проекте будем использовать вычисляемое поле для вывода информации в компонент dbtGrupaFio, в который будет выводиться информация о группе студента и его ФИО.

В наборе данных qStudent создаём новое поле и указываем параметры, как показано на рисунке (рисунок 33).

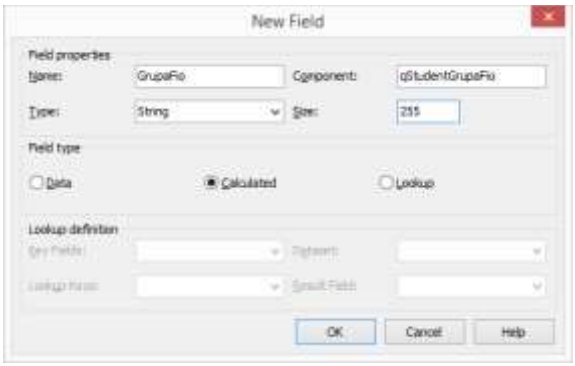

Рисунок 33. Настройки поля GrupaFio

Добавляем новое действие. Называем его qStudentOnCalcFields, в свойстве категория указываем qStudent. В процедуре для этого действия указываем следующий код.

```
qStudentGrupaFio.Value:=qStudentgrupa.Value+' '+qStudentfiostudent.Value;
```

Чтобы эти вычисления выполнялись постоянно, на события OnCalcFields создаём процедуру и вызываем это действие.

```
qStudentOnCalcFields.Execute;
```

Привязываем dbtGrupaFio к новому полю в наборе данных qStudent, сохраняем изменения и запускаем проект.

Организуем вывод контактной информации. Так как для вывода будем использовать только DBGrid, обойдёмся без GridPanel. На вкладку «Контактная информация» установим DBGrid. Переименуем его в dbgInf. Настройте его так, как он был настроен до удаления pcMain. В дополнение свойство ReadOnly устанавливаем в True, а свойство dgEditing (группа свойств Options) устанавливаем в False.

Организуем вывод информации на вкладку «Основные данные». Здесь снова будем использовать GridPanel. Устанавливаем этот компонент на вкладку. Свойство Caption очищаем, в свойстве Align выбираем alClient. Сам компонент переименовываем в grpStudent.

Открываем коллекцию строк (RowCollection). Увеличиваем количество строк до 11. Для них устанавливаем следующие параметры.

Таблица 8. Параметры RowCollection для grpMain

Member	Size Type	Value
Row0	Absolute	10
Row1	Percent	20% (примерно)
Row2	Absolute	10
Row3	Percent	20% (примерно)
Row4	Absolute	10
Row5	Percent	20% (примерно)
Row6	Absolute	10
Row7	Percent	20% (примерно)
Row8	Absolute	10
Row9	Percent	20% (примерно)
Row610	Absolute	10

Открываем коллекцию колонок (ColumnCollection). Добавляем ещё три столбца. Для них устанавливаем следующие параметры.

Таблица 9. Параметры ColumnCollection для grpMain

Member	Size Type	Value
Column0	Absolute	10
Column 1	Percent	30% (примерно)
Column 2	Absolute	10
Column 3	Percent	70% (примерно)
Column 4	Absolute	10

Устанавливаем на GridPanel компоненты. Для надписей будем использовать обычные Label для вывода информации из таблицы DBText, DBRadioGroup и DBMemo. Внешне будет похоже на то, как отображалась информация в первой версии, но вместо DBEdit будем использовать DBText. Разместите компоненты так, как показано на рисунке (рисунок 34).

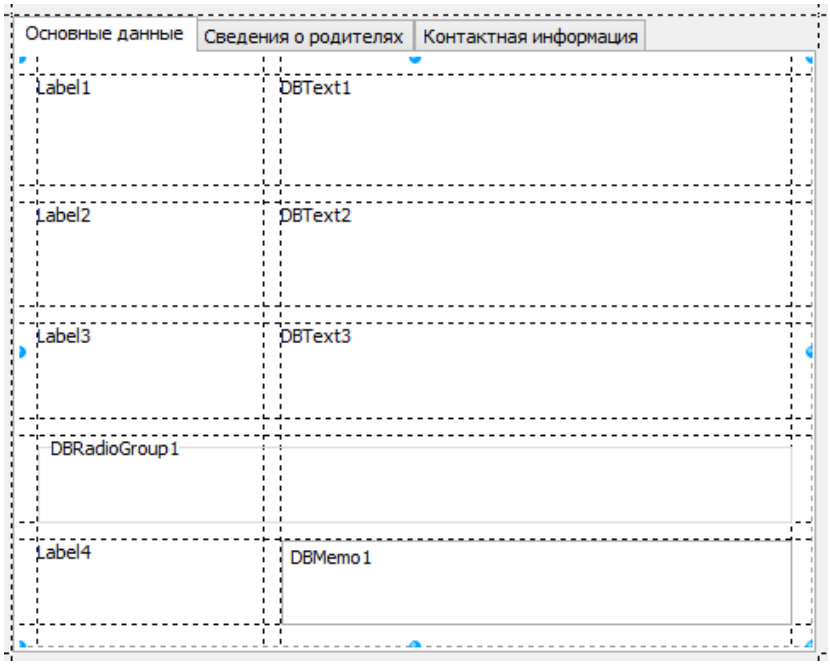

Рисунок 34. Схема расположения компонентов

У компонентов Label1, Label2, Label3, Label4, DBText1, DBText2, DBText3, DBRadioGroup1 свойство Align установите в alTop. У DBMemo1 устанавливаем в alClient. Свойство Caption у Label1 меняем на «№ зачётки», Label2 на «Дата рождения», Label3 на «Паспорт», DBRadioGroup1 на «Пол», Label4 на «Характеристика».

Компоненты DBText1, DBText2, DBText3, DBRadioGroup1 и DBMemo1 связываем с источником данных dsqStudent. Каждый из этих компонентов связываем со своим полем данных, исходя из надписей. У компонентов DBRa-

dioGroup1 и DBMemo1 свойство ReadOnly устанавливаем в True. DBRadioGroup1 настраиваем так же, как и dbrgSex до удаления pcMain.

Изменяем параметры строк в grpStudent.

Таблица 10. Параметры RowCollection для grpMain.

Member	Size Type	Value
Row0	Absolute	10
Row1	Auto	Auto
Row2	Absolute	10
Row3	Auto	Auto
Row4	Absolute	10
Row5	Auto t	Auto
Row6	Absolute	10
Row7	Auto	Auto
Row8	Absolute	10
Row9	Percent	100%
Row610	Absolute	10

Сохраните изменения и запустите проект.

Если необходимо, уменьшите ширину колонки grpStudent.

Вывод информации о родителях организуйте самостоятельно по аналогии, за исключением того, что GridPanel, который необходимо назвать grpParents, разместите в панели DBCtrlGrid. Если сложно представить, какое количество колонок и строк потребуются, попробуйте использовать Excel (рисунок 35).

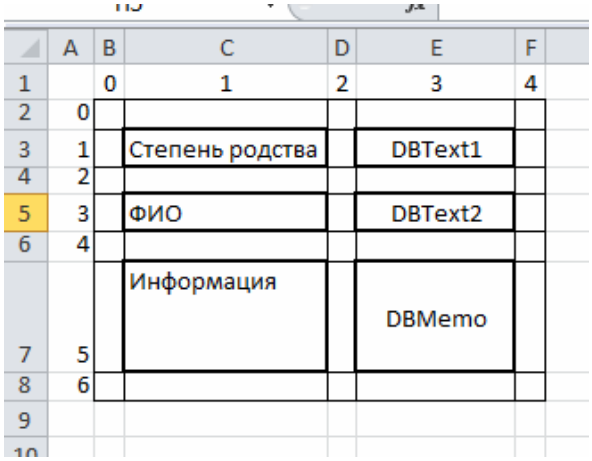

Рисунок 35. Макет размещения элементов в GridPanel

Так должно выглядеть на этапе разработки (рисунок 36).

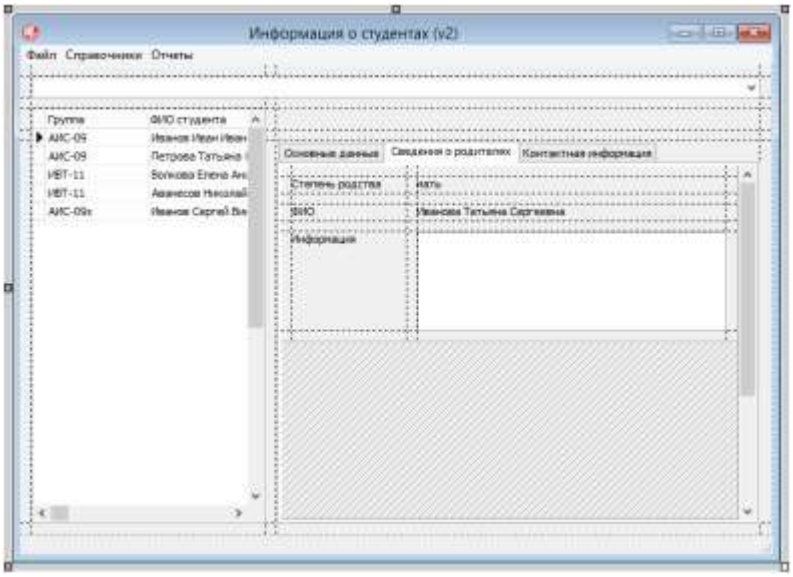

Рисунок 36. Вкладка «Сведения о родителях» на этапе разработки

А так это будет выглядеть после запуска проекта (рисунок 37).

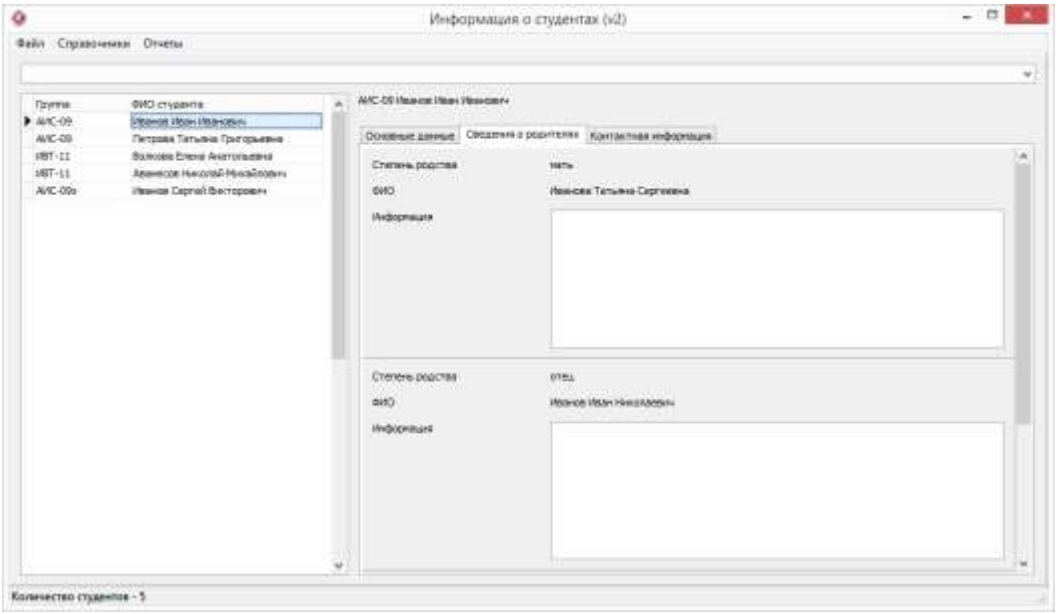

Рисунок 37. Результат после запуска

В процессе работы с данными иногда требуется быстро найти какую-либо информацию. В нашем случае может понадобиться вывести список группы или информацию о каком-либо студенте.

Для решения данной задачи реализуем быстрый поиск по группе и ФИО студента.

На форме уже находится компонент cbSearch, в который пользователь может ввести искомые данные.

Свойство Showhint этого компонента на True, а в свойстве Hint укажите «Быстрый поиск по ФИО студента и учебной группе». Теперь при наведении курсора мыши на эти элементы будет появляться подсказка.

Так как список учебных групп – это справочник, то есть список возможных значений ограничен, то для ввода названия группы пользователь будет использовать выпадающий список. Поиск по ФИО студента будет осуществляться по части имени, фамилии или отчества или всего сразу. Реализуем быстрый поиск в реальном времени, то есть при вводе или выборе значения.

Прежде чем приступить к реализации поиска, необходимо организовать заполнение списка групп в выпадающем списке, так как этот элемент не связан напрямую с набором данных.

В категории fMain создаём действие cbSearchFill. На событие OnExecute этого действия создаём процедуру содержащий следующий программный код.

```
fmain.cbSearch.Items.Clear;

fmain.cbSearch.Items.Add('');

qGrupa.First;
while not qGrupa.Eof do
 begin
  fmain.cbSearch.Items.Add(qGrupagrupa.Value);
  qGrupa.Next;
 end;
fmain.cbSearch.ItemIndex:=0;
```

Не забудьте вставить вызов этого действия в qGrupaRefresh и fMainActivate.

Переходим к реализации поиска информации. Для поиска мы будем использовать возможность фильтрации, которая есть у наборов данных. При использовании в наборе данных связи Master Source фильтры использовать нельзя. Создаём новое действие в категории qStudent под именем qStudentFilter. Код процедуры представлен ниже.

```
if fMain.cbSearch.Text<>'' then
begin
  qStudent.Filter:='(grupa='+quotedstr(fMain.cbSearch.Text)+') or (fiostudent like '
+ quotedstr('%'+fMain.cbSearch.Text+'%')+')';
  qStudent.Filtered:=True;
  fMain.sb.Panels[0].Text:='Найдено записей -
'+inttostr(dm.qStudent.RecordCount);
end
```

```
else
begin
  qStudent.Filtered:=False;
  fMainStatus.Execute;
end;
```

Оператор like в строке фильтра позволяет искать по части текста, этот оператор работает только с текстовыми полями. Символ «%» означает любое сочетание символов.

Необходимо прописать вызов этого действия в процедуре на событие OnChange компонента cbSearch.

Лабораторная работа №5 Реализация функций добавления, редактирования и удаления основных данных

Вызов функций добавления, редактирования и удаления основной информации о студентах будем осуществлять с помощью контекстного меню, которое будет подключено к соответствующим графическим элементам (свойство PopupMenu). Сами функции будут реализованы с помощью действий (action). Добавление и редактирование будет осуществляться через отдельные формы. Причём добавление и редактирование будет осуществляться через одну форму, так как добавление, по сути, это то же самое, что и редактирование, только пустой новой строки. Сохранение значений и при добавлении, и редактировании происходит при выполнении метода Post.

Приступим к реализации этих функций для таблицы Student.

В проект добавим новую форму. Её необходимо переименовать в fStudent, свойство caption не изменяем, оно будет меняться при вызове формы, а остальные настройки устанавливаются по аналогии с формой fSpr. Сохраните проект полностью и модуль новой формы сохраните под именем ufStudent в каталоге проекта.

На форме установите компоненты, настройте их, как показано на рисунке (рисунок 38) и в описании ниже. Надписи сделаны с помощью компонентов Label.

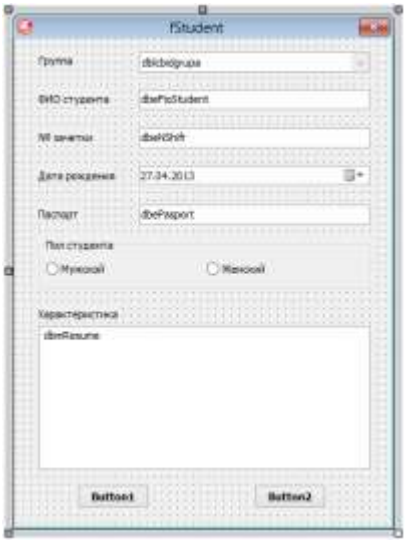

Рисунок 38. Дизайн формы fStudent

Описание компонентов:

- dblcbIdGrupa: TDBLookupComboBox (DataSource = dm.dsqstudent, DataField = 'idgrupa', ListSource = dm.dsqgrupa, KeyField = 'idgrupa', ListField = 'grupa');
- dbeFioStudent: TDBEdit (DataSource = dm.dsqstudent, DataField = 'fiostudent');
- dbeNShifr: TDBEdit (DataSource = dm.dsqstudent, DataField = 'nshifr');
- dtpdbirth: TDateTimePicker (очистите свойство Time);
- dbrgSex: TDBRadioGroup (Caption = «Пол», Columns = 2, DataSource = dm.dsqstudent , DataField = 'sex', Items.Strings = («Мужской», «Женский»), Values.Strings = («True», «False»));
- dbePasport: TDBEdit (DataSource = dm.dsqstudent , DataField = 'pasport');
- dbmResume: TDBMemo (DataSource = dm.dsqstudent, DataField = 'resume');
- button1: TButton (Font.Style = [fsBold]);
- button2: TButton (Font.Style = [fsBold]).

Создаём новые действия в категории qStudent:

- fStudentInsert – вызывает форму fStudent в режиме добавления (необходимо изменить caption на «Добавить»);
- fStudentEdit – вызывает форму fStudent в режиме редактирования (необходимо изменить caption на «Изменить»);
- qStudentDelete – удаляет информацию выбранного студента, причём будет удаляться вся зависящая от этой записи информация в таблицах inf и parents (необходимо изменить caption на «Удалить», обязательно проверьте содержимое базы данных, когда будете тестировать эту функцию);

- fStudentSave – сохранение введённых данных при редактировании или добавлении (необходимо изменить caption на «Сохранить»);
- fStudentCancel – отмена добавления или редактирования (необходимо изменить caption на «Отмена»);
- fStudentOnClose – действия на закрытие формы fStudent;
- pmStudentOnPopup – действие для проверки наличия записей и блокирования или разблокирования функций;
- qStudentRefresh – обновление набора данных.

Исходный программный код процедур для этих действий представлен ниже.

```
procedure Tdm.fStudentInsertExecute(Sender: TObject);
begin
  Application.CreateForm(TfStudent,fStudent);
  fStudent.Caption:='Добавить студента';

  qStudent.Insert;
  fStudent.dtpDBirth.Date:=date();

  fStudent.ShowModal;
end;

procedure Tdm.fStudentEditExecute(Sender: TObject);
begin
  Application.CreateForm(TfStudent,fStudent);
  fStudent.Caption:='Изменить данные студента';

  qStudent.Edit;
  fStudent.dtpdbirth.Date:=qStudentDBirth.Value;
  fStudent.ShowModal;
end;

procedure Tdm.qStudentDeleteExecute(Sender: TObject);
begin
  if Application.MessageBox(pchar('Удалить данные?'), 'Студенты',
MB_YESNO)=IDYES then
  begin
```

```
   qDel.SQL.clear;
   qDel.sql.add('Delete from Student where idstu-
dent='+inttostr(qStudentidstudent.value));
   qDel.execSQL;

   qStudentRefresh.Execute;

   Application.MessageBox(pchar('Данные удалены.'), 'Студенты', MB_OK);
  end;
end;

procedure Tdm.fStudentSaveExecute(Sender: TObject);
begin
  if not qstudentidgrupa.IsNull then
    if not qstudentfiostudent.IsNull then
      if qstudentfiostudent.Value<>'' then
      if not qstudentnshifr.IsNull then
         if qstudentnshifr.Value<>'' then
           if not qstudentsex.IsNull then
           begin
             qstudentdbirth.Value:=fstudent.dtpdbirth.Date;
             qstudent.Post;
             fstudent.Close;
           end
           else Application.MessageBox(pchar('Не выбран пол студента.'), 'Студен-
ты', MB_OK)
         else Application.MessageBox(pchar('В № зачетки пусто.'), 'Студенты',
MB_OK)
       else Application.MessageBox(pchar('Не указан № зачетки.'), 'Студенты',
MB_OK)
     else Application.MessageBox(pchar('В ФИО студента пусто.'), 'Студенты',
MB_OK)
    else Application.MessageBox(pchar('Не указана ФИО студента.'), 'Студенты',
MB_OK)
  else Application.MessageBox(pchar('Не выбрана учебная группа.'), 'Студенты',
MB_OK);
end;
```

```
procedure Tdm.fStudentCancelExecute(Sender: TObject);
begin
fStudent.Close;
end;

procedure Tdm.fStudentOnCloseExecute(Sender: TObject);
begin
  if (qStudent.State=dsInsert)or(qStudent.State=dsEdit)
    then qStudent.Cancel;
end;

procedure Tdm.pmStudentOnPopupExecute(Sender: TObject);
begin
  if dm.qStudent.IsEmpty then
  begin
  fStudentInsert.Enabled:=true;
  fStudentEdit.Enabled:=false;
  qStudentDelete.Enabled:=false;
  end
  else begin
  fStudentInsert.Enabled:=true;
  fStudentEdit.Enabled:=true;
  qStudentDelete.Enabled:=true;
  end;
end;

procedure Tdm.qStudentRefreshExecute(Sender: TObject);
begin
  qStudent.Close;
  qStudent.Open;

  fMainStatus.Execute;
end;
```

Поясним некоторые моменты реализации.

Метод IsEmpty проверяет, является ли этот набор данных пустым или нет, если пустой, то возвращается значение true.

Свойство Enable делает действие доступным или недоступным. Если есть какой-либо элемент управления (кнопка или пункт меню), связанный с этим действием через свойство Action, то элемент также будет доступным или недоступным.

Метод insert переводит набор данных в режим добавления (значение dsInsert свойства state), то есть добавляет новую пустую запись и устанавливает на неё курсор, соответственно, в визуальных элементах управления будут пустые значения.

Так как компонент DateTimePicker не может быть связан с набором данных, но с его помощью проще работать с данными типа дата и время, то есть необходимость программно устанавливать в нём значение, при добавлении записи устанавливается текущая дата (функция Date()), а при редактировании записи – значение из поля dbirth набора данных qStudent. Главное, не забыть очистить свойство Time у этого компонента.

При сохранении значений необходимо проверить данные или на заполняемость, или на правильность заполнения в действии fStudentSave, мы проверяем основные и важные поля на заполняемость с помощью метода isNull поля набора данных.

Перед сохранением значений необходимо дату из dtpdbirth присвоить полю dbirth набора данных qStudent.

При удалении данных необходимо организовать диалог с пользователем, для использования Application.MessageBox необходимо, чтобы в разделе Uses были прописаны модули Vcl.Forms и Winapi.Windows. Пропишите недостающие.

Удаляем данные с помощью запросов на язык SQL через компонент qDel.

Переходим к дальнейшей настройке.

Необходимо организовать вызов действия fStudentOnClose на событие OnClose формы fStudent.

Действия fStudentSave и fStudentCancel привязать через свойство Action к кнопкам button1 и button2 соответственно форме fStudent.

В модуль данных добавляем компонент PopupMenu. Переименуйте его в pmStudent. Подключите к пунктам этого меню с помощью свойства Action действия fStudentInsert, fStudentEdit и qStudentDelete. В результате должно получиться, как на рисунке (рисунок 39).

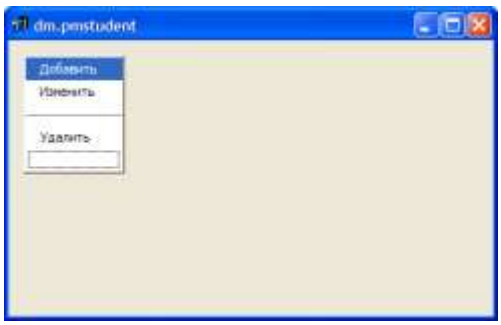

Рисунок 39. Дизайн контекстного меню pmStudent

Вызовите действие pmStudentOnPopup в процедуре на событие OnPopup этого контекстного меню.

Подключите это контекстное меню к визуальным компонентам, которые связаны с dm.qStudent на форме fMain, используя свойство PopupMenu.

Вызовите действие qStudentRefresh в процедуре на событие AfterPost набора данных qStudent.

Все новые действия должны быть либо вызваны в процедурах или функциях, либо подключены к компонентам через Action.

Сохраните изменения и запустите проект. Попробуйте добавить записи о студентах, изменить существующие и удалить. Посмотрите, что произойдёт с содержимым базы данных при удалении какой-либо записи о студенте.

В общем виде алгоритм работы выглядит следующим образом:

- добавляем форму, с помощью которой будем добавлять и редактировать данные, размещаем на ней необходимые элементы управления и настраиваем их;
- создаём необходимые действия в списке действий;
- создаём процедуры по обработке этих действий;
- добавляем контекстное меню в модуль данных, формируем его, связываем пункты меню с действиями и подключаем его к соответствующим визуальным элементам на главной форме;
- настраиваем вызов действий из других процедур или связываем действия с графическими элементами.

Организуйте выполнение таких же функций для оставшихся таблиц: inf и parents, согласно вышеописанному алгоритму. Результат описан дальше.

Добавляем в проект новую форму, переименовываем её в fInf. Основные настройки аналогичны форме fStudent.

Форма fInf представлена на рисунке (рисунок 40).

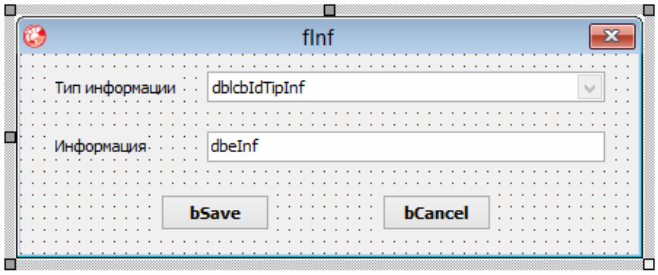

Рисунок 40. Дизайн формы fInf

На этой форме установлены следующие компоненты с изменёнными свойствами:

- dblcbIdTipInf: TDBLookupComboBox (DataSource = dm.dsqinf, DataField = ' idtipinf', ListSource = dm.dsqtipinf, KeyField = ' idtipinf', ListField = ' tipinf');
- dbeInf: TDBEdit (DataSource = dm.dsqinf, DataField = ' inf');
- bSave: TButton (Font.Style = [fsBold]);
- bCancel: TButton (Font.Style = [fsBold]).

Создаём новые действия и указываем для них категорию qInf:

- fInfInsert – вызывает форму fInf в режиме добавления (необходимо изменить caption на «Добавить»);
- fInfEdit – вызывает форму fInf в режиме редактирования (необходимо изменить caption на «Изменить»);
- qInfDelete – удаляет выбранную контактную информацию (необходимо изменить caption на «Удалить»);
- fInfSave – сохранение введённых данных при редактировании или добавлении (необходимо изменить caption на «Сохранить»);
- fInfCancel – отмена добавления или редактирования (необходимо изменить caption на «Отмена»);
- fInfOnClose – действия на закрытие формы fInf;
- pmInfOnPopup – действие для проверки наличия записей и блокирования или разблокирования функций;
- qInfRefresh – действия, выполняемые после добавления, изменения или удаления записей.

Исходный программный код представлен ниже.

```
procedure Tdm.fInfInsertExecute(Sender: TObject);
begin
  Application.CreateForm(TfInf,fInf);
  finf.Caption:='Добавить контактную информацию';
```

```
  qInf.Insert;
finf.ShowModal;
end;

procedure Tdm.fInfEditExecute(Sender: TObject);
begin
  Application.CreateForm(TfInf,fInf);
  finf.Caption:='Изменить контактную информацию';

  qInf.Edit;
finf.ShowModal;
end;

procedure Tdm.qInfDeleteExecute(Sender: TObject);
begin
  if Application.MessageBox(pchar('Удалить данные?'), 'Студенты',
MB_YESNO)=IDYES then
  begin
    qDel.SQL.clear;
    qDel.sql.add('Delete from Inf where idinf='+inttostr(qInfidinf.value));
    qDel.execSQL;

    qInfRefresh.Execute;

    Application.MessageBox(pchar('Данные удалены.'), 'Студенты', MB_OK);
  end;
end;

procedure Tdm.fInfSaveExecute(Sender: TObject);
begin
  if not qinfidtipinf.IsNull then
    if not qinfinf.IsNull then
      if qinfinf.Value<>'' then
      begin
        qinf.Post;
        finf.Close;
```

```
    end
    else Application.MessageBox(pchar('В контактной информации пусто.'),
'Студенты', MB_OK)
   else Application.MessageBox(pchar('Не указана контактная информация.'),
'Студенты', MB_OK)
  else Application.MessageBox(pchar('Не выбран тип информации.'), 'Студенты',
MB_OK);
end;

procedure Tdm.fInfCancelExecute(Sender: TObject);
begin
finf.Close;
end;

procedure Tdm.fInfOnCloseExecute(Sender: TObject);
begin
  if (qinf.State=dsInsert)or(qinf.State=dsEdit)
    then qinf.Cancel;
end

procedure Tdm.pmInfOnPopupExecute(Sender: TObject);
begin
  pmStudentOnPopup.Execute;
  if qinf.IsEmpty then
  begin
  fInfEdit.Enabled:=false;
  qInfDelete.Enabled:=false;
  end
  else
  begin
    fInfEdit.Enabled:=true;
  qInfDelete.Enabled:=true;
  end;
end;

procedure Tdm.qInfRefreshExecute(Sender: TObject);
begin
```

```
  qInf.Close;
qInf.Open;
end;
```

Переходим к дальнейшей настройке.

Необходимо организовать вызов действия fInfOnClose на событие OnClose формы fInf.

Действия fInfSave и fInfCancel привязать через свойство Action к кнопкам bsave и bcancel соотвественно, формы fInf.

В модуль данных добавляем компонент PopupMenu. Переименуйте его в pmInf. Подключите к пунктам этого меню с помощью свойства Action действия fInfInsert, fInfEdit и qInfDelete. В результате должно получиться, как на рисунке (рисунок 41).

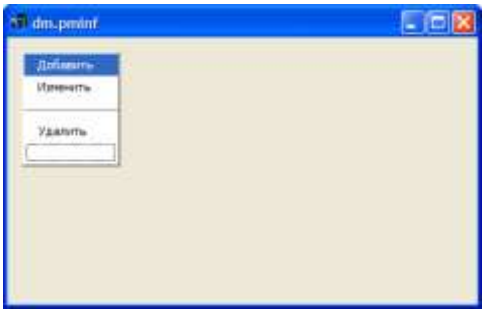

Рисунок 41. Дизайн контекстного меню pminf

Вызовите действие pmInfOnPopup в процедуре на событие OnPopup этого контекстного меню.

Подключите это контекстное меню к визуальным компонентам, которые связаны с dm.qInf, на форме fMain используя свойство PopupMenu.

Вызовите действие qInfRefresh в процедуре на событие AfterPost набора данных qInf.

По аналогии самостоятельно реализуйте такие же функции для таблицы parents.

При добавлении и редактировании данных для указания степени родства можно использовать поле с выпадающим списком (dbComboBox). Это позволит упростить и ускорить ввод данных о степени родства. Недостаток такого способа заполнения заключается в том, что для изменения значений списка необходимо перекомпилировать программу.

В результате должна получиться следующая форма для добавления и редактирования данных о родителях (рисунок 42).

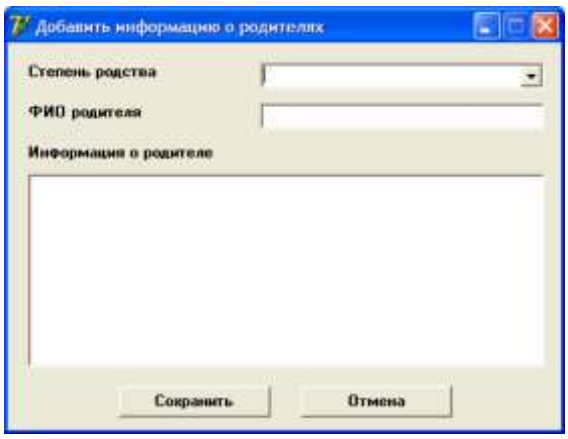

Рисунок 42. Форма fparents при добавлении информации о родителях

Исходный программный код должен получиться следующий.

```
procedure Tdm.fParentsInsertExecute(Sender: TObject);
begin
  Application.CreateForm(TfParents,fParents);
  fParents.Caption:='Добавить информацию о родителе';

  qParents.Insert;
fParents.ShowModal;
end;

procedure Tdm.fParentsEditExecute(Sender: TObject);
begin
  Application.CreateForm(TfParents,fParents);
  fParents.Caption:='Изменить информацию о родителе';

  qParents.Edit;
fParents.ShowModal;
end;

procedure Tdm.qParentsDeleteExecute(Sender: TObject);
begin
  if Application.MessageBox(pchar('Удалить данные?'), 'Студенты',
MB_YESNO)=IDYES then
  begin
    qDel.SQL.clear;
    qDel.sql.add('Delete from Parents where idpar-
ents='+inttostr(qParentsidparents.value));
```

```
   qDel.execSQL;

   qParentsRefresh.Execute;

   Application.MessageBox(pchar('Данные удалены.'), 'Студенты', MB_OK);
  end;
end;

procedure Tdm.fParentsSaveExecute(Sender: TObject);
begin
  if not qParentsstep.IsNull then
    if qParentsstep.Value<>'' then
      if not qParentsfioparents.IsNull then
        if qParentsfioparents.Value<>'' then
        begin
          qParents.Post;
        fparents.Close;
        end
        else Application.MessageBox(pchar('В ФИО родителя пусто.'), 'Студенты',
MB_OK)
      else Application.MessageBox(pchar('Не указано ФИО родителя.'),
'Студенты', MB_OK)
    else Application.MessageBox(pchar('В степени родства пусто.'), 'Студенты',
MB_OK)
  else Application.MessageBox(pchar('Не указана степень родства.'),
'Студенты', MB_OK);
end;
procedure Tdm.fParentsCancelExecute(Sender: TObject);
begin
   fParents.Close;
end;

procedure Tdm.fParentsOnCloseExecute(Sender: TObject);
begin
  if (qParents.State=dsInsert)or(qParents.State=dsEdit)
    then qParents.Cancel;
end;
```

```
procedure Tdm.pmParentsOnPopupExecute(Sender: TObject);
begin
 pmStudentOnPopup.Execute;
 if qParents.IsEmpty then
 begin
 fParentsEdit.Enabled:=false;
  qParentsDelete.Enabled:=false;
 end
 else
 begin
  fParentsEdit.Enabled:=true;
 qParentsDelete.Enabled:=true;
 end;
end;

procedure Tdm.qParentsRefreshExecute(Sender: TObject);
begin
qParents.Close;
qParents.Open;
end;
```

В процессе работы над функциями с таблицами Parents и Inf изменился программный код действия pmStudentOnPopup. Замените имеющийся программный код на следующий.

```
 if dm.qStudent.IsEmpty then
 begin
 fStudentInsert.Enabled:=true;
 fStudentEdit.Enabled:=false;
 qStudentDelete.Enabled:=false;
 fInfInsert.Enabled:=false;
 fParentsInsert.Enabled:=false;
 end
 else begin
  fStudentInsert.Enabled:=true;
  fStudentEdit.Enabled:=true;
  qStudentDelete.Enabled:=true;
```

```
  fInfInsert.Enabled:=True;
 fParentsInsert.Enabled:=True;
 end;
```

Лабораторная работа №6 Формирование отчётов

Для того чтобы пользователь мог не только просмотреть информацию, но и распечатать её, необходимо предусмотреть возможность формирования отчётов.

Отчёт можно сформировать в каком-либо универсальном формате, например в формате HTML. Рассмотрим этот способ подробнее.

Например, нам необходимо составить отчёт, в котором бы содержались списки всех групп, отсортированные по алфавиту. Для реализации этой функции создадим действие ReportStudHTML и поместим его в категорию Reports. В качестве заголовка этого действия задайте «Список студентов по группам (HTML)». На событие onExecute этого действия создадим процедуру и в ней напишем следующий код. После написания кода необходимо это действие связать с подпунктом в разделе меню «Отчёты».

```
procedure Tdm.ReportStudHTMLExecute(Sender: TObject);
var
  tfile:TextFile;
i:integer;
filename:string;
begin
  filename:='report.html';

  AssignFile(tfile,filename);
  rewrite(tfile);
  writeln(tfile,'<HTML>');
  writeln(tfile,'<HEAD><TITLE>');
  writeln(tfile,'Список студентов по группам');
  writeln(tfile,'</TITLE></HEAD>');
  writeln(tfile,'<BODY>');
  writeln(tfile,'<H1>Список студентов по группам</H1>');

  qgrupa.First;
  while not qgrupa.Eof do
  begin
   qstudent.Filter:='idgrupa='+inttostr(qgrupaidgrupa.Value);
```

```
  qstudent.Filtered:=true;
  if not qstudent.IsEmpty then
  begin
    writeln(tfile,'<H2>'+qgrupagrupa.value+'</H2>');
    writeln(tfile,'<TABLE BORDER=1>');
    writeln(tfile,'<TR>');
    writeln(tfile,'<TH>№');
    writeln(tfile,'<TH>ФИО студента');
    writeln(tfile,'<TH>№ зачетки');
    writeln(tfile,'<TH>Дата рождения');
    writeln(tfile,'<TH>Паспорт');
    writeln(tfile,'<TH>Пол');

    i:=1;
    qstudent.First;
    while not qstudent.Eof do
    begin
    writeln(tfile,'<TR>');
      writeln(tfile,'<TD>'+inttostr(i));
      writeln(tfile,'<TD>'+qstudentfiostudent.Value);
      writeln(tfile,'<TD>'+qstudentnshifr.Value);
      writeln(tfile,'<TD>'+datetostr(qstudentdbirth.Value));
      writeln(tfile,'<TD>'+qstudentpasport.Value);
      if qstudentsex.Value
        then writeln(tfile,'<TD>Мужской')
      else writeln(tfile,'<TD>Женский');

      i:=i+1;
      qstudent.Next;
    end;

    writeln(tfile,'</TABLE>');
    writeln(tfile,'<P>Количество студентов -
'+inttostr(qstudent.RecordCount)+'</P>');

  end;
 dm.qgrupa.Next;
 end;
```

```
  qstudent.Filtered:=false;

  writeln(tfile,'<P>Всего студентов - '+inttostr(qstudent.RecordCount)+'</P>');

  writeln(tfile,'</BODY></HTML>');
  CloseFile(tfile);
ShellExecute(Application.Handle,'open',pchar(filename),nil,nil,SW_SHOW);
end;
```

Мы создаём текстовый файл и заносим в него информацию из таблиц, оформляя с помощью элементов разметки HTML. Файл-отчёт сохраняется в том же каталоге, что и исполняемый файл. После сохранения и закрытия файла мы открываем его с помощью команды *ShellExecute*. Этот файл автоматически откроется с помощью браузера. Для работы этой функции необходимо подключить библиотеку Winapi.ShellAPI в раздел Uses. После этого содержимое можно сохранить или распечатать.

Выполняя эти функции, получим следующий результат (рисунок 43).

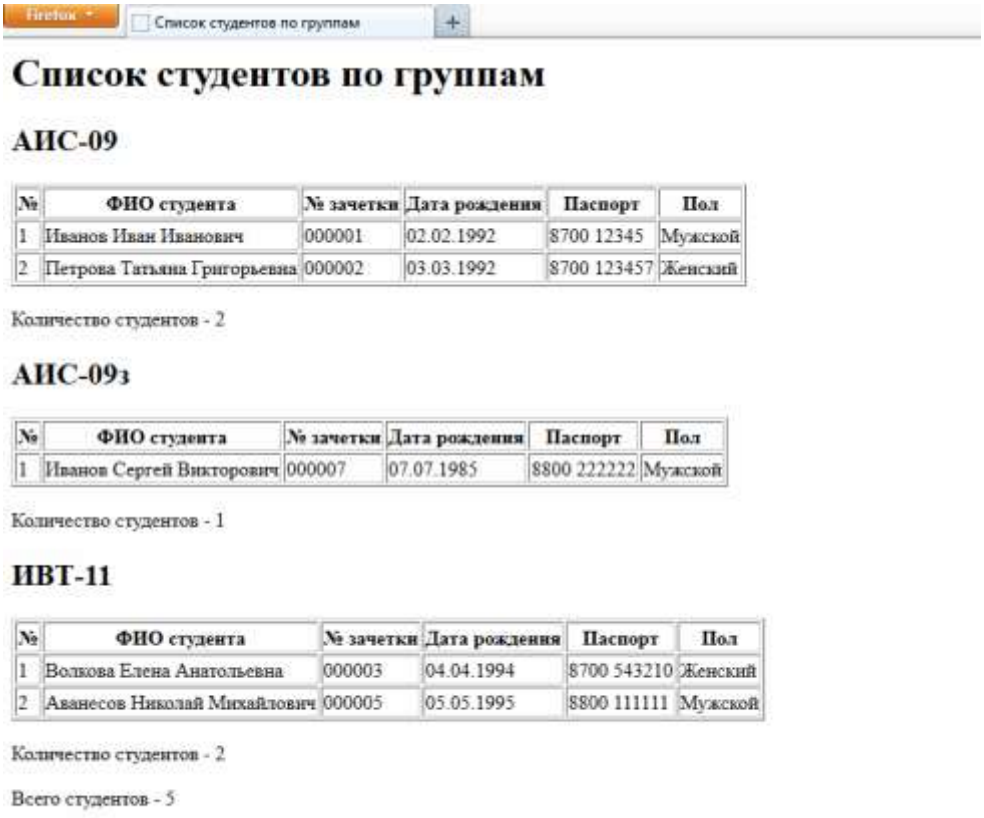

Рисунок 43. Отчёт в формате HTML

Этот способ недостаточно эффективен. Если потребуется изменить вид отчёта, то необходимо перекомпилировать проект, что не всегда возможно.

Лабораторная работа №7 Документирование программного обеспечения.

Пояснительная записка пишется на естественном языке в терминах понятных и пользователю и разработчику программного обеспечения и может содержать следующие разделы:

1. **Заголовок к программе.**
2. **Условие задачи.**

Формулируется условие задачи, краткое описание разрабатываемой программы, ее назначение и необходимые уточнения.

3. Начало/окончание работы.

Указывается месяц и год начала/окончания разработки программы.

4. Основание для разработки программы.

Основанием для разработки программы может быть заказ пользователя, задание администрации учебного заведения, контракт учебного заведения с другой организацией и пр.

5. Краткая характеристика объекта разработки.

Описывается объект разработки: как решается поставленная задача в настоящее время без разрабатываемой программы и какая часть ручной работы будет заменена программой.

6. Пользователь.

Указываются пользователи программы.

7. Цель и назначение разработки.

8. Основные требования.

Описываются требования пользователя к разрабатываемой программе.
Здесь же с точки зрения пользователя следует подробно перечислить функции программы.

9. Входная информация.

Перечисляются все входные данные программы с точки зрения их содержания и назначения - отчеты, файлы, записи, поля данных, таблицы... Их возможные носители и средства отображения информации и т.д.

10. Выходная информация.

Описываются выходные данные так же, как в пункте 9.

11. Требования к аппаратному и программному обеспечению.

Описывается конфигурация аппаратуры и программного обеспечения, в которых разрабатываемая программа может работать, другие программные продукты, от которых она зависит.

12. Внешние ограничения.

13. Эффективность.

Цели производительности, такие, как временные и объемные характеристики, пропускная способность, использование ресурсов и пр.

14. Безопасность данных от несанкционированного доступа.

15. Эргономические характеристики.

Эргономическими характеристиками изделия являются такие свойства, которые обеспечивают надежность, комфорт и продуктивность работы пользователей и операторов. Эргономика (греч.) - труд + закон - отрасль знания, изучающая трудовые процессы с целью создания наи¬лучших условий труда.

16. Мобильность.

Описываются требования и цели обеспечения переноса программного продукта из одних рабочих условий в другие.

17. Окупаемость капиталовложений.

Определяется прибыль, которую даст создание программного продукта в понятиях, соответствующих целевому назначению организации.

18. Другие соглашения сторон.

19. Терминология.

Четко определяется вся терминология, которая может оказаться специфической для данной разработки.

Лабораторная работа №8 CASE-технология поддержки разработки и сопровождения.

Разработка DFD-модель по выполняемому варианту