САРАНСКИЙ КООПЕРАТИВНЫЙ ИНСТИТУТ (ФИЛИАЛ)

АВТОНОМАЯ НЕКОМЕРЧЕСКАЯ ОРГАНИЗАЦИЯ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ЦЕНТРОСОЮЗА РОССИЙСКОЙ ФЕДЕРАЦИИ

“РОССИЙСКИЙ УНИВЕРСИТЕТ КООПЕРАЦИИ’’

Кафедра информационно-вычислительных систем

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине:**

**Высокоуровневые методы информатики и**

**программирования**

**Учет общественной активности студентов академической группы**

Выполнил:

студент группы ПИ-21

специальности : «Прикладная

информатика в экономике»

Кленин Александр Александрович

Руководитель :

к.ф.м.н. доцент

Смолкин Георгий Александрович

Саранск 2011

Содержание

Введение......................................................................................................3

1. Среда программирования Delphi...........................................................6

1.1. Структура баз данных……………………………………………..6

1.2. Компоненты Delphi для работы с базами данных.........................8

2. Постановка задачи и основные функции приложения

«Учет общественной активности студентов академической группы»..11

3. Описание технологии решения задачи и её реализация.....................14

3.1. Организация доступа к данным.....................................................14

3.2. Организация работы с данными.....................................................17

3.3. Создание запроса «Ведомость»......................................................21

Заключение..,..............................................................................................23

Список используемых источников...........................................................24

Приложение................................................................................................25

**Введение**

В настоящее время задача обработки статистических данных является одной из самых популярных задач требуемых как в производстве , так и в сферах обслуживания, потребительской кооперации, образования, здравоохранения и других областях человеческой деятельности. Накопление и автоматическая обработка этих данных позволяет экономить достаточно много времени при построении даже не очень сложных отчетов и таблиц.

В последние десятилетия объем информации увеличивается за год в несколько раз, по этой причине возникает проблема её рационального хранения. В каждой организации существуют специальные программы, в которых хранятся сведения о личном составе, документации, о всей деятельности организации в целом и т.д.

Создание программных продуктов - достаточно трудоемкий процесс основанный на определенной технологии и инструментарии ее разработки.

Проектирование алгоритмов и программ - наиболее ответственный этап жизненного цикла программных продуктов, определяющий, насколько создаваемая программа соответствует спецификациям и требованиям со стороны конечных пользователей. Затраты на создание, сопровождение и эксплуатацию программных продуктов, научно-технический уровень разработки, время морального устаревания и многое другое - все это также зависит от проектных решений.

Проектирование алгоритмов и программ может основываться на различных подходах, среди которых наиболее распространены:

- структурное проектирование программных продуктов;

- информационное моделирование предметной области и связанных с ней приложений;

- объектно-ориентированное проектирование программных продуктов.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные составляющие. Типичными методами структурного проектирования являются:

- нисходящее проектирование, кодирование и тестирование программ;

- модульное программирование;

- структурное проектирование (программирование) и др. Объектно-ориентированный подход к проектированию программных

продуктов основан на:

- выделении классов объектов;

- установлении характерных свойств объектов и методов их обработки;

- создании иерархии классов, наследовании свойств объектов и методов их обработки

Каждый объект объединяет как данные, так и программу обработки этих данных и относится к определенному классу. С помощью класса один и тот же программный код можно использовать для относящихся к нему различных объектов.

Объектный подход при разработке алгоритмов и программ предполагает:

- объектно-ориентированный анализ предметной области;

- объектно-ориентированное проектирование.

Для проектирования программных продуктов разработаны объектно-ориентированные технологии, которые включают в себя специализированные языки программирования и инструментальные средства разработки пользовательского интерфейса.

Объектно-ориентированная технология разработки программных продуктов объединяет данные и процессы в логические сущности - объекты, которые имеют способность наследовать характеристики одного и более объектов, обеспечивая тем самым повторное использование программного кода. Это приводит к значительному уменьшению затрат на создание программных продуктов, повышает эффективность жизненного цикла программных продуктов. При выполнении программы объекту посылается сообщение, которое инициирует обработку данных объекта.

Задание курсового проекта создать приложение «Учет общественной активности студентов академической группы» в среде программирования Delphi, который содержит специальные объекты для работы с базами данных.

Это небольшая и несложная в обращении программа может существенно облегчить работу деканата.

Базы данных считаются основным козырем Delphi. Это действительно так. Хотя этот язык не создавался специально под эту сферу, но реализация работы с данными здесь просто поражает. Даже специализированные языки для работы с базами данных (такие, как MS Visual FoxPro) явно уступают по простоте и мощи программирования этого типа приложений.

Delphi скрывает все сложности и в то же время даёт величайшую мощь. Это достаточно мощная и насыщенная программа для создания с ее помощью приложений с использованием приемов программирования. А главное, что всё это реализовывается очень удобно и легко для понимания.

В Delphi можно создавать простые приложения даже со сложными базами без единой строчки кода.

В курсовом проекте использованы базы Access и современный формат xml. Лучше использовать эти базы в качестве локальных, потому что они поддерживаются большинством систем и отличаются высокой надёжностью.

**1. Среда программирования Delphi**

**1.1. Структура баз данных**

Цель любой информационной системы - обработка данных об объектах реального мира, База данных - это совокупность сведений о конкретных объектах реального мира в какой либо предметной области.

Базы данных делятся на локальные (установленные на компьютере клиента, там же где и работает программа) и удалённые (установленные на сервере, удалённом компьютере). Серверные базы данных располагаются на удалённом компьютере и работают под управлением серверного программного обеспечения. К их главным преимуществам можно отнести возможность работы с одной базой данных одновременно несколькими пользователями, и при этом осуществляется минимальная нагрузка на сеть.

Есть ещё и сетевые базы данных, которые создают слишком большую нагрузку на сеть и неудобны в работе, как для программиста, так и для конечного пользователя.

При клиент-серверной технологии программа клиент посылает простой текстовый запрос на сервер на получение каких-либо данных. Сервер обрабатывает его и возвращает только необходимую порцию данных. Когда нужно изменить какие-то данные, опять посылается запрос к серверу на их изменение и сервер изменяет данные в своей базе. Таким образом, по сети происходит перекачка в основном только текстовых запросов, которые в основном занимают меньше килобайта. Все данные обрабатывает сервер, а значит, машина клиента загружается намного меньше и не так сильно требовательна к ресурсам. Сервер отсылает клиенту только самые необходимые данные, а значит, отсутствует излишняя перекачка копии все базы.

Благодаря всему этому, сетевые базы данных уже устарели и практически не используются. Их практически полностью вытесняет технология клиент-сервер. А вот локальные базы данных будут жить всегда. Может измениться формат их хранения или добавиться какие-то новые функции, но сами базы данных будут существовать.

*Локальные базы данных*

Из локальных баз данных мы будем рассматривать реляционные, как самые распространённые. Что такое реляционная база данных? Это таблица, в которой в качестве столбцов выступают имена хранимых в ней данных, а каждая строка хранит сами данные. Таблица базы данных похожа на электронную таблицу Excel. Локальные таблицы баз данных могут храниться на локальном жёстком диске или централизовано сохраняться на сетевой диск файлового сервера. Эти файлы можно копировать с помощью стандартных средств как любой другой файл, потому что сами таблицы базы данных не привязаны к определённому месту расположения. Главное, чтобы программа могла найти твою таблицу.

В каждой таблице должно быть одно уникальное поле, которое однозначно будет идентифицировать строку. Это поле называется ключевым. Эти поля очень часто используются для связывания нескольких таблиц между собой. Но даже если связывания таблиц нет, ключевое поле всё равно обязательно. В качестве ключа желательно использовать численный тип и если позволяет база данных, то будет лучше, если он будет типа "autoincrement" (автоматически увеличивающееся/уменьшающееся число или счётчик).

Имена столбцов в таблице базе данных, также должны быть уникальными, но в этом случае не обязательно числовыми.

Каждый столбец (поле базы данных) обязательно должен иметь определённый тип. Количество типов и их разновидности зависит от типа базы данных, например формат dBASE (файлы с расширением DBF) поддерживает только 6 типов, a Paradox уже до 15. База данных может храниться в одном файле (Access) или в нескольких (Paradox, dBase). Точнее сказать, данные таблицы всегда хранятся в одном файле, а вот дополнительная информация может располагаться в отдельных файлах. В качестве дополнительной информации могут быть индексы, ограничения или список значений по умолчанию для конкретных полей. Если хотя бы один из файлов запортиться или будет удалён, то данные могут стать недоступными для редактирования.

Что такое *индексы!* Очень часто данные из таблиц подвергаются каким-то изменениям, поэтому прежде чем произвести редактирование над какой-либо строкой, необходимо её найти. Даже статические таблицы, использующиеся в качестве справочников, тоже подвергаются операциям поиска перед выводом запрашиваемых данных. Поиск достаточно трудоёмкая операция, особенно если таблица содержит очень много строк. Индексы направлены на ускорение этой процедуры, а так же могут использоваться в качестве отправной точки при сортировке. На данном этапе тебе достаточно знать, что не проиндексированное поле невозможно упорядочить.

Если тебе надо, чтобы какая-то таблица была упорядочена по полю *«Фамилия»,* то это поле надо сначала проиндексировать. Затем нужно только указать, что таблица должна работать сейчас с таким-то индексом, и она сортируется автоматически.

* 1. **Компоненты Delphi для работы с базами данных**

Для работы с базами в Delphi есть несколько наборов компонент. Каждый набор очень хорошо подходит для решения определённого круга задач. Почему такое разнообразие компонентов? Все они используют разные технологии доступа к данным и отличаются по возможностям. В отличие от Microsoft, которая встроила в свои продукты разработки только технологию доступа к данным ADO собственной разработки, фирма Borland дала разнообразие средств работающих через разные технологии и не ограничивает только своими разработками. Такое положение вещей даёт громадные преимущества перед другими программистами. Помимо этого есть группы, которые могут использоваться в любом случае, вот краткий обзор доступных нам средств. На закладке *Data Access* расположены основные компоненты доступа к данным. Эти компоненты общие для всех и могут использоваться совместно с другими группами компонентов.

На закладке *Data Controls* расположены компоненты для отображения и редактирования данных в таблицах. Эти компоненты так же используются в не зависимости от используемой технологии доступа к данным.

Закладка *BDE* содержит компоненты, позволяющие получить доступ к базам данных по технологии, разработанной фирмой Borland под названием Borland Database Engine. Эта технология сильно устарела и поставляется только для совместимости со старыми версиями. Не смотря на это, она хорошо работает со старыми типами баз данных, такими как Paradox и dBase

*DBExpress* - это новая технология доступа к данным фирмы Borland. Она отличается большей гибкостью и хорошо подходит для программирования клиент серверных приложений, использующих базы данных. Компоненты с одноимённой закладки я советую использовать с базами данных построенных по серверной технологии, например, Oracle, DB2 или MySQL.

ADO *{Active Data Objects)* - технология доступа к данным, разработанная корпорацией Microsoft. Очень хорошая библиотека, но я рекомендую её использовать только с базами данных Microsoft, а именно MS Access или MS SQL Server. Её так же можно использовать, если у тебя специфичный сервер баз данных, который может работать только через ODBC

Работа с базами данных Access идёт через специальную надстройку DAO, которая может устанавливаться на компьютер вместе с программой Office или идти как отдельная установка. Так что если программа не будет работать на компьютере клиента, то надо позаботиться об установке DAO на этот компьютер. В данном курсовом проекте мы рассмотрим технологию доступа к данным ADO *(Active Data Objects).*

**2. Постановка задачи и основные функции приложения «Учет общественной активности студентов академической группы»**

*Описание поставленной задачи*

Задание курсового проекта - разработать приложение, позволяющее вести учет общественной активности студентов академической группы. Разработать систему автоматизированного поиска и отчетности.

На первоначальном этапе создается в MS Access таблица, в которую заполняются данные студентов:

• код студента;

• фамилия;

• имя;

• год рождения;

• место жительства;

• группа;

• спорт

• КВН;

• староста.

Последние три поля определяют активность студентов в этих сферах деятельности. В зависимости от того, в каком виде деятельности участвует студент, ему начисляется надбавка к стипендии. Если студент участвует в одном виде деятельности, то начислить ему 100 руб. если в двух, но не староста, то 150 руб. если в двух, причем староста - 200 руб. в трех - 300 руб. Иначе начисления нет.

Эта небольшая программа существенно упрощает работу деканата и бухгалтерии любого учебного заведения. В деканате в любой момент могут потребоваться данные о каком-либо студенте, и они легко смогут найти эти данные в базе.

Важное значение имеет также удобный и понятный интерфейс приложения. В программе Delphi предусмотрено множество различных объектов и компонентов для создания такого интерфейса. Поэтому созданное приложение заключается в том, что оно будет понятно даже самому неопытному пользователю.

Неоспоримым «плюсом» использования автоматизированных программ является то, что применение компьютерных технологий не только позволяет избежать не нужных бумажных проволочек, но и повышает надежность работы, обеспечивает надлежащую наглядность, и, как уже говорилось выше, экономит не малое количество времени.

В любой момент времени в создаваемую по ходу процесса решения задачи базу данных можно вносить необходимые изменения. В обычных документах и ведомостях, написанных от руки и на бумажном носителе, делать такое гораздо проблематичнее, а порой и просто недопустимо.

Информация для формирования БД первоначально вносится в экранные формы. Простота их структуры предполагает удобство работы с ними.

*Структура таблицы «Студенты»:*

Есть определенные правила построения таблиц баз данных:

1. Каждое поле любой таблицы должно быть уникальным.

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

3. Для каждого значения первичного ключа должно быть одно и только одно значение любого из столбцов данных, и это значение должно относиться к объекту таблицы.

4. Должна иметься возможность изменять значения любого поля (не входящего в первичный ключ), и это не должно повлечь за собой изменение другого поля.

В своей базе данных создана одна таблица под названием «Студенты», со следующими полями: КодСтудента (тип поля: счетчик, оно является ключевым полем), Фамилия (тип поля: текстовый), Фамилия(тип поля: текстовый), Имя (тип поля: текстовый), ГодРождения (тип поля: дата/время), Местожительства (тип поля: текстовый), Группа (тип поля: текстовый), Спорт(тип поля: текстовый), КВН (тип поля: текстовый), Староста (тип поля: текстовый).

*Перечень функций, реализованных в приложении «Учет общественной активности студентов академической группы»*

Приложение «Учет общественной активности студентов академической группы» реализует следующие функции:

1 Просмотр всех студентов;

2 Редактирование данных студента;

3 Добавление нового и удаление старого студента;

4 Поиск данных по фамилии и по группе студента;

5 Сортировка по фамилии студента и по группе (по возрастанию);

6 Получение списка студентов, занимающихся определенным видом деятельности (спорт, КВН или староста);

7 Получение списка активных студентов;

8 Получение списка неактивных студентов;

Приложение предназначено для облегчения выявления активных и неактивных студентов и в последующем доплаты к базовой части стипендии по степени активности студентов, в зависимости от видов деятельности.

**3. Описание технологии решения задачи и её реализация**

**3.1. Организация доступа к данным**

Компоненты работы с данными являются не визуальными и поэтому их удобно размещать в специальном модуле типа Data Module. Для этого после создания (открытия) нового проекта данный модуль создается командой File/New/Data Module, при этом Delphi откроет специальное окно (DataModule2), в которое и помещаются компоненты доступа к данным. При работе с данными MS Access необходимо использовать компоненты вкладки ADO.

Вначале размещается компонент ADOConnection и настраивается соеди­нение, которое должно быть указано в свойстве connectionstring. Для этого надо дважды щелкнуть кнопкой мыши по строке connectionstring (или дважды щелкнуть по самому компоненту).

Здесь нужно остановиться на варианте, предлагаемом по умолчанию (использовать строку подключения - Use Connection String). Щелчком по кнопке Build открывается окно Свойства связи с данными (Рис. 3.1.2.).

В нем на вкладке Поставщик данных (Provider) перечислены все доступные ADO драйверы доступа к данным. В данном случае нужно использовать драйвер Microsoft Jet 4.0 OLE DB Provider (устанавливается с Windows по умолчанию). После этого нажать кнопку Далее (Next), или перейти на вкладку Подключение (Connection).

В строке Выберете или введите имя базы данных (Select or enter a database name), вводится имя базы данных и путь. Так как существующая база данных Студенты.mdb располагается в той же директории, что и исполняемый файл приложения путь указывать не нужно. Значения других полей можно оставить по умолчанию.

Далее нажатием кнопки Проверить подключение (Test Connection), тестируется соединение. Если все указано правильно, то должно появиться сообщение Тестирование соединения прошло удачно (Test connection succeeded). Все окна настроек закрываются нажатием кнопок ОК. Далее в свойствах компонента ADOConnection нужно отключить свойство LoginPrompt, выставив его в false. Это нужно для того, чтобы при каждом обращении к базе не происходил вызов окна ввода пароля. Далее выставим свойство Connected в true, чтобы произошло соединение с базой. На этом соединение можно считать оконченным.

Теперь надо получить доступ к таблице «Студенты». Для этого нужно поместить на форму компонент ADOTable (вкладка ADO). У компонента ADOTable в свойстве Name нужно задать имя LichDan, в свойстве connection указать созданный ранее компонент соединения с базой данных путем выбора из выпадающего списка единственного пункта ADOConnection1. В свойстве TableName выбрать имя таблицы (Студенты). Далее можно активизировать данные (подключаться) путем выставления свойства Active в true.

Для отображения данных из таблицы устанавливается компонент DataSource (в данном случае DataSourseLD) вкладки Data Access. Этому компоненту указывается, какую именно таблицу он должен отображать, путем выбора из выпадающего списка свойства DataSet компонента LichDan, который связан с таблицей Студенты. Приготовления в окне DataModule2 модуля данных Unit2 на этом завершаются и можно приступать реальному отображению данных.

На первом этапе необходимо определить связь компонентов главной формы (модуль Unitl) с модулем данных Unit2. Это можно сделать вручную или командой File/Use Unit (в этот момент должно быть выделено окно кода главной формы, так как модуль подключается именно к ней). В разделе implementation модуля Unitl появится запись uses Unit2;. Затем устанавливается на форме компонент-сетка - DBGrid с вкладки Data Controls. Он может отображать данные в виде таблицы, добавлять, удалять и редактировать строки таблицы. Компонент сетка DBGrid1 путем указания в свойстве DataSource созданного ранее компонента DataSourseLD связывается с компонентом отображения таблицы.

Далее можно запускать данное приложение и затем создать несколько строк, отредактировать уже существующие или удалить что-нибудь. Для вставки строки используется клавиша <Ins>, а для удаления — сочетание клавиш <Ctrl>+<Del>.

Для простоты редактирования данных можно добавить компонент DBNavigator и задать свойство DataSource-DataModule2.DataSourseLD (из списка) именно так и сделано на форме «Студенты».

**3.2. Организация работы с данными**

Для редактирования данных и управления приложением удобно создать главное меню. В моем приложении главное меню содержит следующие подпункты:

1. Файл;

2. Редактирование;

3. Сортировка;

4. Поиск;

5. Активность студентов;

6. Ведомость;

7. О программе.

Рассмотрим работу некоторых из них.

*Добавление, редактирование и удаление данных*

В пункте Редактирование создаем три подпункта: Добавить запись, Редактировать запись, Удалить запись. Для этого создадим новую форму, которую подключим к модулю Data Module2.

Напротив надписей (Label) создадим компоненты DBEdit с вкладки Data Controls. Эти компоненты представляют собой простые строки ввода для поля в базе данных. Чтобы компонент видел данные из нужного поля, следует указать для него в свойстве Data Source необходимую таблицу, а в свойстве Data Field указать поле, которое надо редактировать.

Для обработчика кнопки «Сохранить» код следующий:

if DataModule2.ADOTablel.Modified then DataModule2.ADOTablel.Post; Для кнопки «Отмена» нужно использовать метод Cancel, отменяющий изменения текущей строки, если они не были сохранены еще с помощью метода Post, т.е. написать следующий код:

DataModule1.ADOTable1.Cancel.

После разработки формы редактирования, ее можно использовать в обработчике пункта меню Добавить запись, написав код:

DataModule2.ADOTable1.Insert; // вставка строки в таблицу EditForm.ShowModal; // вызов формы редактирования

Обработчик меню Удалить запись может реализовать следующий код: if Application.MessageBox (РСhаr('Вы действительно хотите удалить запись?' +DataModule2.ADOTablei1.DesignerData), 'Внимание!!!', MB OKCANCEL)=id\_ OK

then DataModule2.ADOTablel .Delete;

*Сортировка данных* В любой базе данных существует понятие индексного поля. Индексы увеличивают скорость поиска данных и позволяют сортировать все записи. Обязательным является только главный индекс (ключевое поле) и существует всегда. Можно создавать любое количество дополнительных индексированных полей, однако индексирование отнимает дополнительное место на диске, и если таких полей будет много, то можно снизить быстродействие программы.

Для обеспечения сортировки данных по каждому полю достаточно написать обработчики событий. В своем приложении сортировка

производится по полям: «По фамилии», «По группе».

Далее приведен пример кода для выполнения сортировки по фамилии:

procedure TForml.N9Click(Sender: TObject);

begin

DataModule2.ADOTable1.IndexFieldNames:='Фамилия'; DataModule2.ADOTable2.IndexFieldNames:='Фамилия';

end;

В данном случае сортировка будет производиться по возрастанию. Для отмены сортировки необходимо указать пустую строку.

Следует помнить о том, что, если вы хотите делать сортировку по какому-либо полю, то при объявлении названия поля в MS Access оно должно состоять из одного слова.

*Фильтрация данных*

Одним из эффективных и востребованных инструментов обработки данных является организация их фильтрации. Для организации фильтрации в компоненте TADOTable есть свойства Filter и Filtered. Свойство Filtered определяет является ли таблица фильтруемой и только если оно имеет значение true, то воспринимается свойство Filter как строка содержащая текст условия определяемого в виде:

Поле [Оператор сравнения] 'Значение'.

В приложении «Учет общественной активности студентов академической группы» реализована функция фильтрации данных для пункта главного меню Активность студентов.

Данный пункт содержит следующие подпункты :

1. По видам деятельности:

1.1. Спортсмены;

Код:

procedure TForml.N19Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTablel.Filter:='Cпopт="+"';

DataModule2.ADOTable2.Filter:='Cпopт="+'";

end;

1.2. КВНщики;

1.3. Старосты.

2. Актив Код:

procedure TForml.N17Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTablel.Filter:='Cпopт="+" and КВН ="+" and Староста ="+'";

DataModule2.ADOTable2.Filter:='Cпopт="+" and КВН ="+" and Староста ="+'";

end;

3. Неактивные студенты

4. Отменить отбор

Код:

procedure

TForml.N25Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=false; DataModule2.ADOTable2.Filtered:=false;

end;

Нужно обратить внимание на то, что значения нужно указывать в одинарных кавычках. Но так как одинарные кавычки используются для ограничения строк, то чтобы внутри строки поставить одинарную кавычку, ее нужно поставить дважды. Именно таким способом ставиться одинарная кавычка перед значением. После значения нужно поставить одинарную кавычку и закрыть строку, поэтому ставиться три одинарных кавычки (две для того, чтобы поставить кавычку для значения и одна для конца строки).

*Организация поиска данных*

В данном приложении организован поиск по фамилии, по группе, и по обоим параметрам. Он организован в виде формы

Далее приведен пример кода для выполнения поиска по фамилии: procedure TForm4.Button1Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:= 'Фaмилия='"+Edit1.Text+"";

DataModule2.ADOTable2.Filter:= 'Фaмилия='"+Edit1.Text+"";

end;

Для отмены поиска используется код:

procedure TForml.N14Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=false;

DataModule2.ADOTable2.Filtered:=false;

end;

**3.3. Создание запроса «Ведомость»**

Для реализации данного запроса создадим еще один ADOTable и DataSourse, которые подсоединяются к той же таблице Студенты, но выполняют чуть другие функции.

Данный запрос содержит поля КодСтудента, Фамилия, Год рождения, Группа и Стипендия. Последнее поле является вычисляемым. Для его создания нужно щелкнуть правой кнопкой на элементе ADOTable2 и выбрать пункт New Field. В появившемся окне свойств нового поля заполнить следующие поля: Name - назвать поле Summa: Type - тип Float: Field Type (тип поля) - выбрать calculated, чтобы создать вычисляемое поле.

Теперь создадим обработчик события OnCalcFields. Это событие вызывается каждый раз, когда надо пересчитать вычисляемые поля.

Для этого выделим компонент ADOTable2 и выберем в Инспекторе объектов вкладку «События». Затем сделаем двойной щелчок в пустой строке, справа от пункта OnCalcFields. Появится окно кода. В этом обработчике для нахождения стипендии каждого студента пишем код (см. Приложение).

Для отображения данных в элементе DBGrid1 по компоненту ADOTable2 в код включаем следующие строки:

procedure TForm1.N23Click(Sender: TObject);

begin

DBGrid1.DataSource:=DataModule2.DataSource2;

end;

При выборе пункта меню Ведомость - Ведомость по стипендии выходит отчет, который содержит информацию о стипендии каждого студента. Данный показатель зависит от степени активности каждого студента. При этом базовая стипендия = 500 руб.

Для данного запроса также реализованы функции сортировки, поиска, отбора студентов по видам деятельности, но добавить и редактировать запись здесь не возможно.

**Заключение**

Использование компьютерных технологий, при использовании автоматизированных программ позволяет избежать ненужных бумажных проволочек и повышает надежность работы, обеспечивает надлежащую наглядность, экономит немалое количество времени.

В любой момент времени в создаваемую по ходу процесса решения задачи базу данных можно вносить необходимые изменения. В обычных документах и ведомостях, написанных от руки и на бумажном носителе, делать такое гораздо проблематичнее, а порой и просто недопустимо.

Ещё десять лет назад, программирование баз данных было очень сложным занятием. За какие-либо достижения в этой области многие программисты получили в своё время докторские степени. Сейчас уже такое трудно себе представить, потому что благодаря Delphi, процесс написания программ упростился, а количество разновидностей баз данных уже исчисляется десятками.

В данном курсовом проекте разработана архитектура базы «Студенты» и показана реализация связи данной базы данных с программой Delphi через компонент ADOConnection.

Разработанное приложение может существенно облегчить работу деканата любого учебного заведения.

Работа базы на контрольных данных (вымышленных) показала ее работоспособность и эффективность.

**Список используемых источников**

1. Архангельский А.Я. Программирование в Delphi7 - М.: ООО «Бином-Пресс», 2005 г., 1150 стр.

2. База данных: способы модернизации // Журнал «Справочник по управлению персоналом» № 9, сентябрь 2003.

3. Барановская Т.П., Лойко В.И. Информационные системы и технологии в экономике: Учебник. - 2-е изд.,доп. и перераб./. - М.: Финансы и статистика, 2003.

4. Компьютерные программы для службы кадров // Журнал «Справочник кадровика» №2, февраль 2002.

5. Теоретическое учебное пособие РГТК «Тантал» по дисциплине «Технология разработки программного обеспечения».

6. Титоренко Г.А. Информационные технологии управления: Учеб. Пособие для вузов /- 2-е изд., доп. - М.:ЮНИТИ-ДАНА, 2003.

7. Фленов М.Е. Библия Delphi. - СПб.: БХВ-Петербург, 2004. - 880 с: ил.

Электронные учебники:

8. Иллюстрированный самоучитель по Delphi 7 для начинающих

9. Иллюстрированный самоучитель по Delphi 7 для профессионалов

**Приложение**

***Form1***

unit Unit 1;

interface

uses

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

Type

TForm1 = class(TForm)

DBGridUTDBGrid;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

N8: TMenuItem;

N9: TMenuItem;

N10: TMenuItem;

N11: TMenuItem;

N12: TMenuItem;

N13: TMenuItem;

N14: TMenuItem;

N15: TMenuItem;

N16: TMenuItem;

N17: TMenuItem;

N18: TMenuItem;

N19: TMenuItem;

N20: TMenuItem;

N21: TMenuItem;

N22: TMenuItem;

N23: TMenuItem;

N24: TMenuItem;

Button1:TButton;

N25: TMenuItem;

N26: TMenuItem;

procedure N2Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N5Click(Sender: TObject);

procedure N6Click(Sender: TObject);

procedure N7Click(Sender: TObject);

procedure N9Click(Sender: TObject);

procedure N10Click(Sender: TObject);

procedure N11Click(Sender: TObject);

procedure N13Click(Sender: TObject);

procedure N14Click(Sender: TObject);

procedure N19Click(Sender: TObject);

procedure N20Click(Sender: TObject);

procedure N21Click(Sender: TObject);

procedure N17Click(Sender: TObject);

procedure N18Click(Sender: TObject);

procedure N23Click(Sender: TObject);

procedure N24Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure N25Click(Sender: TObject);

procedure N26Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Forml: TForm1;

implementation

uses Unit2, Unit3, Unit4, Unit5;

{$R \*.dfm}

procedure TForm1.N2Click(Sender: TObject);

begin

if DataModule2.ADOTablel .Modified then DataModule2.ADOTablel .Post;

end;

procedure TForm1.N3Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.N5Click(Sender: TObject);

begin

DataModule2.ADOTable1.Insert;

Form3.ShowModal;

end;

procedure TForml.N6Click(Sender: TObject);

begin

Form3.ShowModal;

end;

procedure TForm1.N7Click(Sender: TObject);

begin

if application.MessageBox(PChar('Вы действительно хотите удалить запись'+ DataModule2.ADOTable1.DesignerData), 'Внимание!!!',MB\_OKCANCEL)=id\_OK

then DataModule2.ADOTablel .Delete;

end;

procedure TForml.N9Click(Sender: TObject);

begin

DataModule2.ADOTable1.IndexFieldNames:='Фамилия';

DataModule2.ADOTable2.IndexFieldNames:='Фамилия';

end;

procedure TForml.N10Click(Sender: TObject);

begin

DataModule2.ADOTable1.IndexFieldNames:= 'Гpyппa';

DataModule2.ADOTable2.IndexFieldNames:= 'Гpyппa';

end;

procedure TForml.N11Click(Sender: TObject);

begin

DataModule2.ADOTable1.IndexFieldNames:=";

DataModule2.ADOTable2.IndexFieldNames:=";

end;

procedure TForml.N13Click(Sender: TObject);

begin

Form4.Show;

end;

procedure TForml.N14Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=false;

DataModule2.ADOTable2.Filtered:=false;

end;

procedure TForml.N19Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Cпopт="+'";

DataModule2.ADOTable2.Filter:='Cпopт="+'";

end;

procedure TForm1.N20Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='KBH="+'";

DataModule2.ADOTable2.Filter:='KBH="+'";

end;

procedure TForml.N21Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Cтароста="+'";

DataModule2.ADOTable2.Filter:='Староста="+'";

end;

procedure TForml.N17Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Cпopт="+" and KBH="+" and Староста="+'";

DataModule2.ADOTable2.Filter:='Cпopт="+" and KBH="+" and Староста="+'";

end;

procedure TForml.N18Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Cпopт="-" and KBH="-" and Староста="-'";

DataModule2.ADOTable2.Filter:='Cпopт="-" and KBH="-" and Староста="-'";

end;

procedure TForml.N23Click(Sender: TObject);

begin

DBGrid1.DataSource:=datamodule2.DataSource2;

end;

procedure TForm1.N24Click(Sender: TObject);

begin

Form5.Show;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

Close;

end;

procedure TForml.N25Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=false;

DataModule2.ADOTable2.Filtered:=false;

end;

procedure TForm1.N26Click(Sender: TObject);

begin

DBGrid1.DataSource:=datamodule2.DataSource1;

end;

end.

***DataModule2***

unit Unit2;

interface

uses

SysUtils, Messages, Classes, DB, ADODB;

type

TDataModule2 = class(TDataModule)

ADOConnection1: TADOConnection;

ADOTablel:TADOTable;

DataSourcel: TDataSource;

ADOTablelDSDesigner: TAutoIncField;

ADOTable1DSDesigner2: TWideStringField;

ADOTable1DSDesigner3: TWideStringField;

ADOTable1DSDesigner4: TDateTimeField;

ADOTable1DSDesigner5: TWideStringField;

ADOTable1DSDesigner6: TWideStringField;

ADOTable1DSDesigner7: TWideStringField;

ADOTable1DSDesigner8: TWideStringField;

ADOTable1DSDesigner9: TWideStringField;

ADOTable2: TADOTable;

DataSource2: TDataSource;

ADOTable2DSDesigner: TWideStringField;

ADOTable2DSDesigner2: TWideStringField;

ADOTable2DSDesigner3: TDateTimeField;

ADOTable2DSDesigner4: TWideStringField;

ADOTable2DSDesigner5: TWideStringField;

ADOTable2DSDesigner6: TWideStringField;

ADOTable2DSDesigner7: TWideStringField;

ADOTable2Summa: TFloatField;

ADOTable2DSDesigner8: TAutoIncField;

procedure ADOTable2CalcFields(DataSet: TDataSet);

procedure ADOTable2AfterOpen(DataSet: TDataSet);

private

{ Private declarations }

public

{ Public declarations }

end;

var

DataModule2: TDataModule2;

implementation

uses Unit1, Unit3, Unit4;

{$R \*.dfm}

procedure TDataModule2.ADOTable2CalcFields(DataSet: TDataSet);

var

stip: Real;

const

bazstip: Integer=500;

profsous: Real= 0.01;

begin

//--Проверка на активиста

if (ADOTable2DSDesigner4.Value='+') OR

(ADOTable2DSDesigner5.Value='+')OR

(ADOTable2DSDesigner6.Value='+')then

begin

//--Студент точно активист,

//--проверка на старосту.

if (ADOTable2DSDesigner6.Value='+') AND

((ADOTable2DSDesigner4.Value='+') OR (ADOTable2DSDesigner5.Value='+')) then

begin

//--Студент староста,

//--проверка занимается ли он еще и всем остальным

if (ADOTable2DSDesigner4.Value='+') AND

(ADOTable2DSDesigner5.Value='+')then

//--Студент староста и занимается еще 2 видами деятельности

//стипендия базовая +300

begin

stip:=(bazstip+300)\*(1-profsous);

ADOTable2Summa.Value:=stip;

end

else

//--Студент староста, и занимается еще 1 видом деятельности

//--стипендия базовая +200

begin

stip:=(bazstip+200)\*(l-profsous);

ADOTable2Summa.Value:=stip;

end;

end

else

//--Студент активист, но не староста

//--проверка на 2 вида деятельности

begin

if (ADOTable2DSDesigner4.Value='+') AND

(ADOTable2DSDesigner5.Value='+') then

//--Студент занимается 2 видами деятельности

//--стипендия базовая +150

begin

stip:= (bazstip+150)\*(1-profsous);

ADOTable2Summa.Value:=stip;

end

else

//--Студент занимается только 1 видом деятельности

//--стипендия базовая +100

begin

stip:=(bazstip+100)\*(1-profsous);

ADOTable2Summa.Value:=stip;

end;

end;

end

else

begin

//--Студент не активист, базовая

stip:= bazstip;

ADOTable2Summa.Value:=stip;

end;

end;

procedure TDataModule2.ADOTable2AfterOpen(DataSet: TDataSet);

begin

ADOTable2.First;

end;

end.

***Form3***

unit Unit3;

interface

uses

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

Dialogs, ExtCtrls, DBCtrls, StdCtrls, Mask;

type

TForm3 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

DBEditl: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBEdit6: TDBEdit;

DBEdit7: TDBEdit;

DBEdit8: TDBEdit;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

DBNavigator1: TDBNavigator;

Buttonl: TButton;

Button2: TButton;

procedure Button2Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

implementation

uses Unit1, Unit2;

{$R \*.dfm}

procedure TForm3.Button2Click(Sender: TObject);

begin

if DataModule2.ADOTablel .Modified then DataModule2.ADOTablel .Post;

end;

procedure TForm3.Button1Click(Sender: TObject);

begin

DataModule2.ADOTable1.Cancel;

end;

end.

***Form4***

unit Unit4;

interface

uses

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

Dialogs, StdCtrls;

type

TForm4 = class(TForm)

Label 1: TLabel;

Label2: TLabel;

Edit1:TEdit;

Edit2:TEdit;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

implementation

uses Unit1, Unit2, Unit3;

{$R\*.dfm}

procedure TForm4.ButtonlClick(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Фамилия='"+Edit1.Text+"";

DataModme2.ADOTable2.Filter:='Фамилия='"+Edit1.Text+"";

end;

procedure TForm4.Button2Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:= 'Группа='''+Edit2.Text+'''';

DataModule2.ADOTable2.Filter:= 'Группа='''+Edit2.Text+'''';

end;

procedure TForm4.Button3Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=true;

DataModule2.ADOTable2.Filtered:=true;

DataModule2.ADOTable1.Filter:='Фамилия='"+Edit1.Text+'" or Группа='"+Edit2.Text+"";

DataModule2.ADOTable2.Filter:='Фамилия='"+Edit1.Text+'" or Группа='"+Edit2.Text+"";

end;

procedure TForm4.Button5Click(Sender: TObject);

begin

DataModule2.ADOTable1.Filtered:=false;

DataModule2.ADOTable2.Filtered:=false;

end;

procedure TForm4.Button4Click(Sender: TObject);

begin

Form4.close;

end;

end.

***Form5***

unit Unit5;

interface

uses

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

Dialogs, ExtCtrls, StdCtrls, jpeg;

type

TForm5 = class(TForm)

Image 1: TImage;

Label 1: TLabel;

private

{ Private declarations }

public

{ Public declarations }

end;

var Form5: TForm5;

implementation

{$R \*.dfm}

end.