ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Назарова Қанатбая студента 4-го курса факультета компьютерный инжиниринг по направлению информатика и информационной технологии

ТЕМА: Разработка программного обеспечения для анализа учебной статистической информации (на примере Ходжейлинского профколледжа связи)

Научный руководитель:_____Ю. Кутлымуратов

Зав. кафедрой:_____доц. Арзымбетов Т.З.

НУКУС - 2014 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время обработка и хранение информации не является чисто умозрительной задачей. Потеря информации или ее несвоевременное получение могут обернуться потерей денег. Именно этим обстоятельством можно объяснить столь бурный рост компьютерной техники и стремительное развитие электронных таблиц и систем управления базами данных (СУБД) в нашей стране и за рубежом. Для оперативного, гибкого и эффективного управления предприятиями, фирмами и организациями различных форм собственности, телекоммуникационными средствами гражданского и военного назначения, информационно – вычислительными, экологическими, радиолокационными системами широко внедряются системы автоматизированного управления, ядром которых являются базы данных (БД). При большом объеме информации и сложности, производимых с ней операций проблема эффективности средств организации хранения, доступа и обработки данных приобретет особое значение. Учитывая важность и значимость баз данных в современной жизни, весьма серьезные требования предъявляются к квалификации специалистов, создающих приложения на их основе.

Приложение баз данных, как следует уже из его названия, предназначено для взаимодействия с некоторым источником данных — базой данных (БД). Взаимодействие подразумевает получение данных, их представление в определенном формате для просмотра пользователем, редактирование в соответствии с реализованными в программе бизнес-алгоритмами и возврат обработанных данных обратно в базу данных. В качестве источника данных могут выступать как собственно базы данных, так и обычные файлы — текстовые, электронные таблицы и т. д. Базы данных обслуживаются специальными программами — системами управления базами данных (СУБД), которые делятся на локальные,

преимущественно однопользовательские, предназначенные для настольных приложений. Механизм внутреннего представления данных является ядром приложения баз данных. Он обеспечивает хранение полученных данных в приложении и предоставляет их по запросу других частей приложения. Пользовательский интерфейс обеспечивает просмотр и редактирование данных, а также управление данными и приложением в целом. Бизнес-логика приложения представляет собой набор реализованных в программе алгоритмов обработки данных. Между приложением и собственно базой данных находится специальное программное обеспечение (ПО), связывающее программу и источник данных и управляющее процессом обмена данными. Это ПО может быть реализовано самыми разнообразными способами, в зависимости от объема базы данных, решаемых системой задач, числа пользователей, способами соединения приложения и базы данных.

Сегодня деятельность в любой области разных направлении, требует от специалиста применения для анализа информации методов статистики.

Элементарные методы анализа статистических данных представляют собой базовые приемы обработки и анализа статистической информации, которые позволяют языком цифр характеризовать изменения, происходящие в области разных направлении.

Элементарные методы анализа статистических данных является вводным курсом к такой фундаментальной науке как “Статистика”. Элементарные методы дают первое представление о статистике и анализе информации; основываются на смысловых понятиях, рассматриваемых экономической теорией, имеет о методологии теории научного познания.

Методические указания по выполнению контрольной и расчетно-графической работ по дисциплине “Элементарные методы анализа статистических данных” созданы с целью помочь студентам о владеть основами статистической науки, научиться применять научные методы статистического исследования, приобрести практические навыки расчета статистических показателей, понимать экономический смысли с численных

показателей, анализировать их и изображать статистические данные графически.

В данной выпускной квалификационной работе рассматриваются проектирование и разработка программного обеспечения для анализа статистических данных профессиональных колледжей.

Работа состоит из введения, 4-х параграфов, заключения и литературы.

Во введении излагается актуальность и важность выбранной тематики исследовании.

В первом параграфе рассматривается элементарные методы анализа статистических данных для исследование предметной области.

В втором параграфе рассматривается возможности программного языка Delphi для анализа статистических данных.

Третий параграф посвящен создание интерфейса для доступа к базами данным в среде Delphi

В четвертом параграфе рассматриваются вопросы проектирования и разработка программного обеспечения для анализа статистических данных профессиональных колледжей.

В заключении кратком виде приведены полученные результаты и их значения.

В приложении приведена текст программы.

§1. Элементарные методы анализа статистических данных для исследование предметной области

Исследование предметного области выбрано статистического формулировка оформления расчетно-контрольную работу для учеников профессионального колледжа.

Выполненная контрольная работа должна соответствовать следующим требованиям:

-работа должна быть выполнена и представлена на рецензирование в срок, установленный преподавателем;

-задачи следует решать в том порядке, в каком они даны в задании;

-перед решением задачи должно быть полностью приведено ее условие;

-решение задач следует сопровождать необходимыми формулами, подробными расчетами и краткими пояснениями, изображать полученные показатели графически. Произведенные расчеты нужно проверять взаимосвязью между исчисленными показателями. Необходимо четко формулировать выводы, раскрывающие экономическое содержание и значение исчисленных показателей. Все расчеты относительных показателей нужно производить с принятой в статистике точностью до 0,001, апроценты до 0,1;

-работа должна быть написана разборчиво, без по мароки зачеркиваний и аккуратно оформлена. В работе допускаются лишь общепринятые сокращения. Страницы должны быть пронумерованы и иметь поля;

-в конце работы нужно привести список с пользованных источников.

Ученики, не получившие зачета по контрольной работе, к зачету не допускаются. Если выполнение контрольной работы вызывает затруднения, следует обратиться за устной или письменной консультацией на кафедру.

Перечень заданий для контрольной работы формируется по усмотрению преподавателя из предложенных задач.

Указания о порядке выполнения контрольной работы и ее содержание. Приступить к выполнению контрольной работы следует после проработки теоретического материала.

Контрольная работа состоит из шести вариантов. Каждый вариант включает 7 задач, выбор варианта определяется начальной буквой фамилии ученика (таблица 1.1):

Таблица 1.1

Начальные буквы фамилии ученика	**Номер выполняемого варианта**
А, Ж, Н, У, Щ	Первый
Б, З, О, Ф, Э	Второй
В, И, П, Х, Ю	Третий
Г, К, Р, Ц, Я	Четвёртый
Д, Л, С, Ч	Пятый
Е, М, Т, Ш	Шестой

Статистическое наблюдение – это массовое, планомерное, научно организованное наблюдение за явлениями социально – экономической жизни, которое заключается в регистрации отобранных признаков у каждой единицы совокупности.

Следует уяснить, что статистическое наблюдение является целенаправленным, научно организованным процессом. Полученная в ходе статистического наблюдения информация на последующих этапах статистического исследования позволяет обеспечить научно – обоснованные выводы о характере и закономерностях изучаемого явления.

Подготовка наблюдения включает в себя большой круг разного вида работ. Сначала необходимо решить вопросы программно –

методологические, такие как определение цели и объекта наблюдения; состава признаков, подлежащих регистрации; разработка документов для сбора данных; выбор отчетной единицы
и единицы, относительно которой будет проводиться наблюдение. Затем решают организационные вопросы, например, определение состава органов, проводящих наблюдение; подборка и подготовка кадров для проведения наблюдения; составление календарного плана работ; тиражирование документов для сбора данных.

Схема проведения статистического наблюдения может быть реализована в следующей последовательности:

1) формулировка цели статистического наблюдения;

2) определение объекта статистического наблюдения, единицы наблюдения, отчетной единицы;

3) разработка программы статистического наблюдения;

4) проектирование статистического формуляра, инструкции по заполнению статистического формуляра;

5) построение макетов статистических таблиц для подведения итогов статистического наблюдения;

6) определение критического момента, выбор места и времени наблюдения;

7) установление вида статистического наблюдения:

а) по степени охвата единиц совокупности: сплошное и не сплошное (выборочное, метод основного массива, монографическое обследование)

б) по учету факторов во времени: текущее (непрерывное) и прерывное наблюдение (периодическое, единовременное)

8) выбор способа статистического наблюдения: непосредственное, документальное, опрос (устный, саморегистрация, корреспондентский, анкетный, явочный);

9) указание формы статистического наблюдения: статистическая отчетность, специально – организованное, регистры;

10) обозначение вопросов организационного характера.

Пример – Для изучения успеваемости учеников второго курса специальности “Статистика” очной формы обучения по итогам зимней экзаменационной сессии необходимо провести статистическое обследование. Разработаем схему проведения статистического наблюдения:

1) цель статистического наблюдения – получение достоверной информации об успеваемости учеников;

2) объект статистического наблюдения – группа учеников второго курса специальности “Статистика”.

Единица наблюдения – ученик второго курса специальности “Статистика”.

Отчетная единица – ученик второго курса специальности “Статистика”;

3) программа статистического наблюдения: успеваемость ученика; посещаемость занятий в течение семестра; время, затрачиваемое на подготовку к занятиям; использование учеником дополнительных источников при подготовке к занятиям; заинтересованность в учебе; жилищные условия; семейное положение; наличие работы у ученика;

4)

Статистический формуляр:

Ф.И.О. _____

1 Как Вы сдали зимнюю экзаменационную сессию?

а) отлично □ б) хорошо и отлично □ в) хорошо □ г) удовлетворительно □ д) неудовлетворительно □

2 Пропускали ли Вы занятия в течение семестра?

а) никогда □ б) иногда □ в) постоянно □ г) вообще не посещал занятия

3 Сколько времени ежедневно в течение семестра Вы затрачивали на подготовку к занятиям?

а) вообще не готовился к занятиям □ б) менее 1 часа □ в) от 1 до 3 часов □ г) 3 часа и более □

4 Для подготовки к занятиям пользовались ли Вы дополнительными источниками

(учебниками, монографиями, научными журналами и др.)

а) никогда □ б) иногда □ в) всегда □

5 Нравится ли Вам учиться?

а) да □ б) нет □

6 Ваше семейное положение:

а) замужем (женат) □ б) не замужем (не женат) □

7 Ваши жилищные условия:

а) проживаю с родителями □ б) снимаю квартиру или комнату □ в) проживаю в общежитии □

Инструкция к заполнению формуляра:

Формуляр заполняется черной гелиевой ручкой. В каждом вопросе необходимо отметить крестиком тот вариант, который вы считаете нужным. В каждом вопросе должен быть указан единственный вариант ответа.

5) макет статистической таблицы для подведения итогов статистического наблюдения:

№	Ф.И.О. ученика	Успевае мость	Посеща емость	Время нотовку к занятиям	Дополнительн ые источники при	Интерес к учебе	Семейное положение	Жилищные условия
1								
2								
3								
…								

6) место проведения статистического наблюдения.

Критический момент – 31.01.2004г.

Время наблюдения – с 09.02.2004 по 14.02.2004г;

7) вид статистического наблюдения:

а) по степени охвата единиц совокупности – сплошное;

б) по учету факторов во времени – прерывное (единовременное);

8) способ статистического наблюдения – опрос (экспедиционный);

9) форма статистического наблюдения–специально– организованное;

10) вопросы организационного характера:

- подготовка кадров, проведение инструктажей;

- подготовка документации обследования, ее размножение;

- проведение разъяснительных работ.

Задача 2 составлена на тему “Сводка и группировка статистических данных”.

В результате статистического наблюдения получают материалы, которые содержат данные о каждой единице совокупности. Дальнейшая задача статистики заключается в том, чтобы эти материалы систематизировать и сгруппировать.

Группировкой статистических данных называется расчленение сложного массового явления на однородные группы в качественном отношении по каким-либо существенным признакам.

Следует различать следующие виды статистических группировок:

1) типологическая группировка – это разделение исследуемой качественно разнородной совокупности на классы, социально-экономические типы, однородные группы единиц в соответствии с правилами научной группировки;

2) структурная группировка – это группировка, в которой происходит разбиение однородной совокупности на группы, характеризующие ее структуру по какому-либо варьирующему признаку;

3) аналитическая группировка – это группировка, выявляющая взаимосвязи между изучаемыми явлениями и их признаками. Взаимосвязь проявляется в том, что с возрастанием значения факторного признака систематически возрастает или убывает среднее значение результативного признака. Особенности аналитической группировки: в основу группировки

кладется факторный признак; каждая выделенная группа характеризуется средними значениями результативного признака.

Порядок проведения статистической группировки (по количественному признаку) может быть осуществлен в следующей последовательности:

1) первоначально необходимо решить вопрос о выборе группировочного признака, по которому проводится разбивка единиц совокупности на отдельные группы;

2) Определение числа групп производится по формуле:

$$n = 1 + 3{,}322 \cdot \lg N ,$$

где n – число групп;

N – число единиц совокупности;

3) после определения числа групп определяют интервалы группировки. Интервалы в зависимости от их величины бывают неравными и равными. Величину равного интервала можно определить по следующей формуле:

$$i = \frac{x_{\max} - x_{\min}}{n}$$

где x_{max} -максимальное значение признака в совокупности;

x_{min} - минимальное значение признака в совокупности.

По обозначению границ выделяют интервалы открытые (интервалы, у которых указана только одна граница: верхняя – у первого, нижняя – у последнего) и закрытые (интервалы, у которых обозначены обе границы);

4) проведение разбивки единиц совокупности по выделенным группам и подведение итогов.

§2. Программирования Delphi для анализа статистических данных профессиональных колледжей

Программа необходима для колледжа, так как с ее помощью можно следить за успеваемостью студентов. Программа будет написана на языке программирования Delphi, с использованием базы данных, созданных а MS ACSESS.

Язык программирования Delphi - это комбинация нескольких важнейших технологий:

- Высоко производительный компилятор в машинный код;

- Объектно-ориентированная модель компонент;

- Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;

-Масштабируемые средства для построения баз данных.[2]

Компилятор, встроенный в Delphi, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL. Кроме того, Delphi обеспечивает быструю разработку без необходимости писать вставки на C или ручного написания кода (хотя это возможно).

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в Delphi мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10-20 раз быстрее, чем

то же самое, сделанное при помощи интерпретатора. В Delphi компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.[2]

Основной упор в модели Delphi делается на максимальном рейс пользовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды Delphi. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Все в Delphi написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые можно создать.

В стандартную поставку Delphi входят основные объекты, которые образуют удачно подобранную иерархию из 270 базовых классов. Но если возникнет необходимость в решении какой-то специфической проблемы на Delphi, стоит просмотреть список свободно распространяемых или коммерческих компонент, разработанных третьими фирмами, количество этих фирм в настоящее время превышает число 250. Во многом это объясняется тем, что традиционно в среде Windows было достаточно сложно реализовывать пользовательский интерфейс. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в Delphi является самой простой задачей для программиста.

Среда Delphi включает в себя полный набор визуальных инструментов для скоростной разработки приложений (RAD - rapid application development), поддерживающей разработку пользовательского интерфейса и подключение к корпоративным базам данных. VCL - библиотека визуальных компонент, включает в себя стандартные объекты построения

пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE. Единственное, что можно поставить в вину Delphi, это то, что готовых компонент, поставляемых Borland, могло бы быть и больше. Однако, разработки других фирм, а также свободно распространяемые программистами freeware-компоненты уже восполнили этот недостаток.[2]

Соответствующий стандарт компонент назывался VBX. И этот стандарт так же поддерживается в Delphi. Однако, визуальные компоненты в Delphi обладают большей гибкостью.

В Delphi визуальные компоненты пишутся на объектном паскале, на том же паскале, на котором пишется алгоритмическая часть приложения. И визуальные компоненты Delphi получаются открытыми для надстройки и переписывания.

Объекты БД в Delphi основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав Delphi также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью.

Кроме того, Delphi включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме. Разработчик в среде Delphi, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата .dbf (как в dBase или Clipper) или .db (Paradox). Если же он будет использовать локальный InterBase for Windows 4.0 (это локальный SQL-сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

Одно и то же приложение можно использовать как для локального, так и для более серьезного клиент-серверного вариантов.[10]

Выпущены две версии Delphi - одна (Delphi Client-Server) адресована для разработчиков приложений в архитектуре “клиент-сервер”, а другая (Delphi for Windows) предназначена для остальных программистов. Приложения, разработанные при помощи Delphi, можно использовать без выплаты royalty-процентов и без оплаты runtime- лицензий.

Она адресована корпоративным разработчикам, желающим разрабатывать высокопроизводительные приложения для рабочих групп и корпоративного применения.

Клиент-серверная версия включает в себя следующие особенности:

- SQL Links: специально написанные драйвера для доступа к Oracle, Sybase, Informix, InterBase;

- Локальный сервер InterBase: SQL-сервер для Windows. СУБД для разработки в корпоративных приложений на компьютере, не подключенном к локальной сети;

- ReportSmith Client/server Edition: генератор отчетов для SQL-серверов;

- Team Development Support: предоставляет версионный контроль при помощи PVCS компании Intersolve (приобретается отдельно) или при помощи других программных продуктов версионного контроля;

- Visual Query Builder - это средство визуального построения SQL-запросов;

- лицензия на право распространения приложений в архитектуре клиент-сервер, изготовленных при помощи Delphi;

- исходные тексты всех визуальных компонент.

Delphi for Windows представляет из себя подмножество Delphi Client-Server и предназначен для разработчиков высокопроизводительных персональных приложений, работающих с локальными СУБД типа dBase и Paradox.Delphi Desktop Edition предлагает такую же среду для быстрой разработки и первоклассный компилятор как и клиент-серверная версия (Client/Server Edition). Эта среда позволяет разработчику быстро изготавливать персональные приложения, работающие с персональными

СУБД типа dBase и Paradox. Delphi позволяет также создавать разработчику DLL, которая может быть вызвана из Paradox, dBase, C++ или каких-нибудь других готовых программ:

- компилятор Object Pascal (этот язык является расширением языка Borland Pascal 7.0);
- генератор отчетов ReportSmith 2.5;
- среда визуального построителя приложений;
- библиотека визуальных компонент;
- Локальный сервер InterBase.

В продукт, выпущенный компанией Borland для Delphi в RAD Pack for Delphi входит набор полезных дополнений, которые помогут разработчику при освоении и использовании Delphi. Это учебник по объектному паскалю, интерактивный отладчик самой последней версии, Borland Visual Solutions Pack (набор VBX для реализации редакторов, электронных таблиц, коммуникационные VBX, VBX с деловой графикой и т.п.), Resource WorkShop для работы с ресурсами Borland Pascal 7.0, а также эксперт для преобразования ресурсов BP 7.0 в формы Delphi.

В первую очередь Delphi предназначен для профессионалов-разработчиков корпоративных информационных систем. Некоторые продукты, предназначенные для скоростной разработки приложений (RAD - rapid application development) прекрасно работают при изготовлении достаточно простых приложений, однако, разработчик сталкивается с непредвиденными сложностями, когда пытается сделать что-то действительно сложное.[10]

Delphi предназначен не только для программистов-профессионалов. Руководители предприятий, планирующие выделение средств на приобретение программных продуктов, должны быть уверены в том, что планируемые инвестиции окупятся. Программист на паскале способен практически сразу профессионально освоить Delphi. Специалисту, ранее использовавшему другие программные продукты, придется труднее, однако

самое первое работающее приложение он сможет написать в течение первого же часа работы на Delphi. Открытая технология Delphi является мощным гарантом того, что инвестиции, сделанные в Delphi, будут сохранены в течение многих лет.[3]

Локальный сервер InterBase - это инструмент предназначен только для автономной отладки приложений. В действительности он представляет из себя сокращенный вариант обработчика SQL-запросов InterBase, в который не включены некоторые возможности настоящего сервера InterBase. Отсутствие этих возможностей с лихвой компенсируется преимуществом автономной отладки программ.

Team Development Support - средство поддержки разработки проекта в группе. Позволяет существенно облегчить управление крупными проектами. Это сделано в виде возможности подключения такого продукта как Intersolve PVCS 5.1 непосредственно к среде Delphi.

Высокопроизводительный компилятор в машинный код - в отличие от большинства Паскаль - компиляторов, транслирующих в р-код, в Delphi программный текст компилируется непосредственно в машинный код, в результате чего Delphi- приложения исполняются в 10-20 раз быстрее (особенно приложения, использующие математические функции). Готовое приложение может быть изготовлено либо в виде исполняемого модуля, либо в виде динамической библиотеки, которую можно использовать в приложениях, написанных на других языках программирования.

Благодаря такой архитектуре приложения, изготовленные при помощи Delphi, работают надежно и устойчиво. Delphi поддерживает использование уже существующих объектов, включая DLL, написанные на C и C++, OLE сервера, VBX, объекты, созданные при помощи Delphi. Delphi имеет полностью объектную ориентацию, разработчики могут создавать свои повторно используемые объекты для того, чтобы уменьшить затраты на разработку.

Delphi предлагает разработчикам - как в составе команды, так и индивидуальным - открытую архитектуру, позволяющую добавлять компоненты, где бы они ни были изготовлены, и оперировать этими вновь введенными компонентами в визуальном построителе. Разработчики могут добавлять CASE-инструменты, кодовые генераторы, а также авторские help'ы, доступные через меню Delphi.

Two-way tools - однозначное соответствие между визуальным проектированием и классическим написанием текста программы. Это означает, что разработчик всегда может видеть код, соответствующий тому, что он построил при помощи визуальных инструментов и наоборот.

Визуальный построитель интерфейсов (Visual User-interface builder) дает возможность быстро создавать клиент-серверные приложения визуально, просто выбирая компоненты из соответствующей палитры.

Библиотека объектов включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE.

Delphi использует структурный объектно-ориентированный язык (Object Pascal), который сочетает с выразительную мощь и простоту программирования, характерную для языков 4GL, и эффективность языка 3GL. Программисты немедленно могут начать производить работающие приложения, и им не придется для этого изучать особенности программирования событий в Windows. Delphi полностью поддерживает передовые программные концепции включая инкапсуляцию, наследование, полиморфизм и управление событиями.[3]

Это очень важная особенность для разработчиков в среде Windows, поскольку в уже существующие Windows-приложения программист может интегрировать то, что разработает при помощи Delphi.[2]

Работа с базами данных в Delphi.

Общая характеристика технологии ADO

В Delphi имеется ряд компонентов, которые предназначены только для работы с MS Access, эти компоненты находятся в ADO. Чтобы обрабатывать некоторую структуру данных для неё должна быть написана программа, поставщик этих данных в соответствии с системными требованиями, такая программа называется OLE DB Provider. Такие поставщики сегодня реализованы для разных структур данных и разных СУБД. С помощью технологии OLE DB можно однотипным образом обрабатывать сложную и специфическую информацию. Однако работа с OLE DB достаточно сложна, поэтому фирма Microsoft разработала новую технологию ADO, представляющая собой набор простых компонентов. Если планируется создать новое приложение, ориентированную на работу с данными и независящая не от конкретной СУБД и не от способа хранения информации, то лучше использовать технологию ADO.[5]

Технология Microsoft ActiveX Data Objects (ADO) обеспечивает универсальный доступ к источникам данных из приложений БД. Такую возможность предоставляют функции набора интерфейсов, созданные на основе общей модели объектов COM и описанные в спецификации OLE DB.

Технология ADO и интерфейсы OLE DB обеспечивают для приложений единый способ доступа к источникам данных различных типов (Рис. 1). Например, приложение, использующее ADO, может применять одинаково сложные операции и к данным, хранящимся на корпоративном сервере SQL, и к электронным таблицам, и локальным СУБД. Запрос SQL, направленный любому источнику данных через ADO, будет выполнен.[5]

За серверы БД беспокоиться не стоит, обработка запросов SQL — это их основная обязанностью. OLE DB представляет собой набор специализированных объектов COM, стандартные функции обработки данных, и специализированные функции конкретных источников данных и интерфейсов, обеспечивающих передачу данных между объектами.

Согласно терминологии ADO, любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым при помощи провайдера данных взаимодействует приложение. Минимальный набор компонентов приложения может включать объект соединения, объект набора данных, объект процессора запросов. Технология ADO в целом включает в себя не только сами объекты OLE DB, но и механизмы, обеспечивающие взаимодействие объектов с данными и приложениями. На этом уровне важнейшую роль играют провайдеры ADO, координирующие работу приложений с хранилищами данных различных типов.

Провайдеры ADO обеспечивают соединение приложения, использующего данные через ADO, с источником данных (сервером SQL, локальной СУБД и файловой системой. Для каждого типа хранилища данных должен существовать провайдер ADO.

Провайдер «знает» о местоположении хранилища данных и его содержании, умеет обращаться к данным с запросами и интерпретировать возвращаемую служебную информацию и результаты запросов с целью их передачи приложению.[5]

Механизм доступа к данным через ADO и многочисленные объекты, и интерфейсы реализованы в VCL Delphi в виде набора компонентов, расположенных на странице ADO. Все необходимые интерфейсы, обеспечивающие работу компонентов, объявлены и описаны в файлах OleDB.pas и ADODB.pas.

Такая архитектура позволяет сделать набор объектов и интерфейсов открытым и расширяемым. Набор объектов и соответствующий провайдер может быть создан для любого хранилища данных без внесения изменений в исходную структуру ADO.

Так как технология ADO основана на стандартных интерфейсах COM, которые являются системным механизмом Windows, это сокращает общий объем работающего программного кода и позволяет распространять приложения БД без вспомогательных программ и библиотек.

Компоненты ADO.

На закладке ADO расположены компоненты:

а) Компоненты соединения:

- ADOConnection;

- ADOCommand;

б) Стандартные компоненты:

- ADODataSet - универсальный набор данных;

- ADOTable - таблица БД;

- ADOQuery - запрос SQL;

- ADOStoredProc - хранимая процедура.

На странице ADO Палитры компонентов Delphi, кроме компонентов соединения есть стандартные компоненты, обозначающие набор данных и адаптированные для работы с хранилищем данных ADO.

Компонент ADOConnection вобрал возможности перечислителя, источника данных и сессии с возможностями обслуживания транзакций. Текстовые команды ADO реализованы в компоненте ADOCommand. Наборы рядов можно получить при помощи компонентов ADOTable, ADOQuery, AOostoredProc.

Каждый из них реализует способ доступа к конкретному типу представления данных в хранилище. Применительно к компонентам Delphi, совокупность возвращаемых из хранилища данных строк будем называть набором записей. Набор свойств и методов компонентов ADO обеспечивает реализацию всех необходимых приложению БД функций. Способы использования компонентов ADO немногим отличаются от стандартных компонентов VCL доступа к данным. Однако при необходимости разработчик может использовать все возможности интерфейсов ADO, обращаясь к ним через соответствующие объекты ADO. Ссылки на объекты имеются в компонентах.[10]

Настройка соединения база данных с ADO.

Перед созданием соединения необходимо определить его параметры. Для этого, как уже говорилось, предназначено свойство ConnectionString.

Набор параметров изменяется в зависимости от типа используемого провайдера и может настраиваться как вручную, так и с помощью редактора. Для того чтобы вызывать редактор соединений, необходимо дважды щелкнуть на компоненте TADOConnection.

В результате будет активировано диалоговое окно, показанное на рис. 2.

В этом окне можно настроить соединение, используя поле Use Connection String, или загрузить параметры соединения из файла в разделе Use Data Link File. Параметры соединения хранятся в файлах UDL, представляющих собой обычные текстовые файлы, содержащие параметры соединения.

Для того чтобы настроить соединение сданным провайдером, необходимо нажать на кнопку Build. Появится окно, в котором будет опубликован список доступных провайдеров.

На вкладке Provider можно выбрать подходящий провайдер данных OLE DB для конкретного источника данных. В списке провайдеров также присутствуют провайдеры, предназначенные для доступа к конкретным службам операционной системы. На вкладке Connection необходимо указать путь к базе данных или сервер. Вкладка Advanced предназначена для указания режима доступа, аналогично свойству Mode. Вкладка АН предназначена для более «тонкой» настройки специфичных свойств провайдера. Для дальнейшей работы нужно выбрать провайдер Microsoft Jet 4.0 OLE DB Provider. Затем нужно перейти на вкладку Connection.

В появившемся окне необходимо указать путь к базе данных. В поле Select or enter a database name нужно указать путь к демонстрационной базе dbdemos.mdb. После указания пути к базе данных и задания остальных необходимых параметров нужно проверить созданное соединение при помощи кнопки Test Connection. Если параметры соединения указаны верно,

появится сообщение Test connection succeeded. После закрытия этого окна в строке соединения будет отображена информация, с помощью которой провайдер сможет получить доступ к данным.

Компонент TADOQuery TADOQuery позволяет выполнять SQL-запросы при работе с данными через ADO. Соединение с хранилищем данных осуществляется стандартным методом. Текст запроса содержится в свойстве SQL.

Параметры запроса содержатся в свойстве Parameters. В случае, если компонент возвращает набор данных, его следует открывать методом Open или присвоить свойству Active значение True. Если запрос не должен возвращать набор данных (операторы INSERT, UPDATE, DELETE и CREATE TABLE), то запрос следует выполнять вызовом метода ExecSQL. Метод возвращает число обработанных запросом записей.

Свойство RowsAffected содержит число записей, которые затронул последний выполнявшийся запрос.

Компонент TADOTable используется для доступа к хранилищам данных ADO и представления информации из них в табличном виде. Компонент предоставляет прямой доступ к каждой записи и ее полям, наследуя свойства и методы класса TCustomADODataSet. Компонент связывается с базой данных через свойства Connection или ConnectionString.

Имя таблицы указывается в свойстве TableName. Свойство TableDirect указывает, каким образом набор данных связывается с хранилищем данных. Так как не все провайдеры поддерживают прямое соединение с набором данных, то в некоторых случаях для связи с хранилищем данных приходится использовать SQL-операторы. При установке свойству значения True компонент использует фоновые SQL-запросы для доступа к данным.

Используя свойство Readonly, можно установить ограничение «только для чтения» на данную таблицу, запретив, таким образом, возможность изменять данные. В свойстве MasterSource указывается компонент

TDataSource, используемый для создания отношения ссылочной целостности Master-Detail.

Метод GetlndexNames возвращает список индексов, доступных компоненту в качестве списка.

Компонент TDataSource - этот компонент связывается с набором данных. Эта связь осуществляется через свойство DataSet, которое содержит информацию о текущем состоянии набора данных. У этого компонента существует набор свойств и методов, которые облегчают работу с ним.

Свойство AutoEdit автоматически переводит набор данных в состояние редактирования, если имеет значение True, когда связанный элемент ввода получает фокус.

Метод Edit переводит связанный набор данных в состояние редактирования. Метод-обработчик OnDataChange вызывается при редактировании данных в связанном визуальном компоненте.

Метод-обработчик события OnUpdateData вызывается перед тем, как измененные данные будут сохранены в наборе данных. Обработчик вызывается перед выполнением метода Post.

Метод-обработчик события OnStateChange вызывается, когда изменяется состояние связанного набора данных.

Набор данных - массив записей, полученный приложением по собственному запросу, называется набором данных. Набор данных как объект ведет свое начало от класса TDataSet и наследует его свойства.[10]

§3. Создание интерфейса для доступа к данным в среде Delphi

Особенности компонентов доступа к данным.

Компоненты доступа к данным являются невизуальными компонентами. Таблицы БД располагаются на диске и являются физическими объектами. Для операций с данными, содержащимися в таблицах, используются наборы данных. В терминах системы Delphi набор данных представляет собой совокупность записей, взятых из одной или нескольких таблиц БД. Записи, входящие в набор данных, отбираются по определенным правилам, при этом в частных случаях набор данных может включать в себя все записи из связанной с ним таблицы или не содержать ни одной записи. Набор данных является логической таблицей, с которой можно работать при выполнении приложения. Взаимодействие таблицы и набора данных напоминает взаимодействие физического файла и файловой переменной.[5]

В Delphi для работы с наборами данных служат компоненты DBTable и ADOTable, DBQuery и ADOQuery, DataAccess, DataControl, DecisionQuery и StoredProc.

Компонент StoredProc используется для вызова хранимых процедур при организации взаимодействия с удаленными БД, а компонент UpDateSQL обеспечивает работу с кэшированными изменениями в записях. Компонент DecisionQuery применяется при построении систем принятия решений. Наиболее универсальными и, соответственно, часто используемыми являются компоненты Table и Query, задающие наборы данных.[5]

Компонент доступа к данным DataSet.

Базовые возможности доступа к БД обеспечивает класс DataSet, представляющий наборы данных в виде совокупности строк и столбцов (записей и полей). Этот класс предоставляет основные средства навигации и редактирования наборов данных.

Компонент DataSet предназначен для представления набора данных из хранилища данных ADO. Он прост в использовании, имея только несколько собственных свойств и методов.

Это единственный компонент ADO, инкапсулирующий набор данных, для которого опубликованы свойства, позволяющие управлять командой ADO. В результате компонент представляет собой гибкий инструмент, который позволяет (в зависимости от типа команды и ее текста) получать данные из таблиц, запросов SQL, хранимых процедур, файлов и т. д.[5]

Компоненты доступа к данным Table и Query.

Компоненты Table и Query являются производными от класса ADODataSet потомка класса DataSet. Они демонстрируют схожие с базовыми классами характеристики и поведение, но каждый них имеет и свои особенности.

Компонент ADOTable обеспечивает использование в приложениях Delphi таблиц БД, подключенных через провайдеры OLE DB. По своим функциональным возможностям и применению он подобен стандартному табличному компоненту. В основе компонента лежит использование команды ADO, но ее свойства настроены заранее и изменению не подлежат.

Другие свойства и методы компонента обеспечивают применение индексов. Так как не все провайдеры ADO обеспечивают прямое использование таблиц БД, то для доступа к ним может понадобиться запрос SQL.

Компонент Table представляет собой набор данных, который в некоторый момент времени может быть связан только с одной таблицей БД. Этот набор данных формируется на базе навигационного способа доступа к данным, поэтому компонент Table рекомендуется использовать для локальных БД, таких как dBase и Paradox. При работе с удаленными БД следует использовать компонент Query. Связь между таблицей и компонентом Table устанавливается через его свойства TableName, которое задает имя таблицы.

При задании значения свойства TableName указывается имя файла и расширение имя файла. На этапе разработки приложения имена всех таблиц доступны в раскрывающемся списке Инспектора объектов. В этот список попадают таблицы, файлы которых расположены в каталоге, указанном в свойстве TableName.

Компонент ADOQuery обеспечивает применение запросов SQL при работе с данными через ADO. По своей функциональности он подобен стандартному компоненту запроса.

Компонент Query представляет собой набор данных, записи которого формируются в результате выполнения SQL-запроса и основаны на реляционном способе доступа к данным. При работе с удаленными БД рекомендуется использовать набор данных Query. При работе с удаленными базами данных следует обращаться к средствам языка SQL. Это относится и к таким операциям, как перемещение по набору данных или вставка в него записей. Если же для компонента Query используются методы типа Next и Insert, то вместо реляционного способа доступа к данным будет применён навигационный. В результате набора данных Query будет мало чем отличаться от набора данных Table. В отличие от компонента Table. Набор данных Query может включать в себя записи более чем одной таблицы БД. Текст запроса, на основании которого в набор данных отбираются записи, содержится в свойстве SQL типа Strings. Запрос включает в себя команды на языке SQL и выполняется при открытии набора данных. Запрос SQL иногда называется SQL-программой. SQL-запрос можно формировать и изменять динамически, внося изменения в его текст непосредственно при выполнении приложения.[5]

Компоненты для работы с данными.

Понятие и особенности визуальных компонентов в среде Delphi.

Визуальные компоненты для работы с данными расположены на странице DataControls Палитры компонентов и предназначены для построения интерфейсной части приложения. Они используются для

навигации по набору данных, а также для отображения и редактирования записей. Часто эти компоненты называются элементами, чувствительные к данным.[5]

Одни визуальные компоненты для работы с данными предназначены для выполнения операций с полями отдельной записи, они отображают и позволяют редактировать значение поля текущей записи. К таким компонентам относятся, например, однострочный редактор Edit и графический обзор Image.

Другие компоненты служат для отображения и редактирования сразу нескольких записей. Примерами таких компонентов являются сетки DBGrid и DBCtrlGrid, выводящие записи набора данных в табличном виде. Визуальные компоненты для работы с данными похожи на соответствующие компоненты страниц Standard и Additional и отличаются, в основном, тем, что ориентированны на работу с БД и имеют дополнительные свойства DataSource и Datafield. Первое из них указывает на источник данных, а второе - на поле набора данных, с которым связан визуальный компонент. Например, Edit отображает строковое значение, позволяя пользователю изменять его.

Для всех визуальных компонентов, предназначенных для отображения и редактирования значений полей, при изменении пользователем их содержимого набор данных автоматически переводится в режим редактирования. Произведённые с помощью этих компонентов изменения автоматически сохраняются в связанных с ними полях при наступлении определённых событий.

При программном изменении содержимого эти визуальных компонентов набор данных автоматически в режим редактирования не переводится. Для этой цели в коде должен предварительно вызываться метод Edit набора. Чтобы сохранить изменения в поле (полях) текущей записи, мы должны также предусмотреть соответствующие действия, например, вызов метода Post или переход к другой записи набора данных.[5]

Общая характеристика визуальных компонентов.

В библиотеке визуальных компонентов для всех компонентов, в том числе и предназначенных для работы с данными, базовым является классу Control. Он обеспечивает основные функциональные атрибуты такие, как положение и размеры элемента, его заголовок, цвет и другие параметры. Класс Control включает в себя общие для визуальных компонентов свойства, события и методы. В целом визуальные компоненты можно разделить на две группы: оконные и неоконные.

Оконный элемент управления представляет собой специализированное окно, предназначенное для решения конкретной задачи. К таким элементам относятся, например, поля редактирования, командные кнопки, полосы прокрутки.

Такие компоненты, как Edit, DBEdit, Memo или DBMemo при получении фокуса ввода отображают в своей области курсор редактирования. Компоненты, не связанные с редактированием информации, получение фокуса ввода обычно отображают с помощью с помощью пунктирного черного прямоугольника.

К неоконным элементам управления базовым является класс GraphiControl, производимый непосредственно от класса Control. Неоконные элементы управления на могут получать фокус ввода. Их достоинством является менее расходования ресурсов.

Свойства и события визуальных компонентов для работы с данными.

Свойства позволяют управлять внешним видом и поведением компонентов при проектировании и при выполнении приложения. Обычно установка значений большинства свойств компонентов выполняется на этапе проектирования с помощью инспектора объектов.

Свойство Name указывает на имя компонента, которое используется для управления компонентов во время выполнения приложения. Каждый новый компонент, помещаемый на форму, получает имя по умолчанию, автоматически образуемое путем добавления к названию компонента его

номера в порядке помещения на форму. На этапе разработки приложения мы можем изменять имя компонента на более осмысленное и соответствующее назначению компонента.

Свойство Aling определяет способ выравнивания компонента на самой форме, на которой оно находится. Выравнивание используется в случае, когда требуется, чтобы какой-либо интерфейсный элемент занимал определённое положение.

Свойство Caption содержит строку для надписи заголовка компонента. Отдельные символы в заголовке могут быть подчеркнуты, они означают комбинации клавиш быстрого доступа.

Свойство Color определяет цвет фона. Часто удобно задавать цвета с помощью констант. Отображаемые цвет зависит от параметров видеокарты и монитора, в первую очередь от установленного цветного разрешения.

Визуальные компоненты способны генерировать и обрабатывать достаточно большое число событий различных видов. К наиболее общим группам событий можно отнести следующие действия:

- Выбор управляющего элемента;
- Перемещение указателя мыши;
- Нажатие клавиш клавиатуры;
- Получение и потеря управляющим элементом фокуса ввода;
- Перемещение объектов методом drag-and-drop.

Существуют и более сложные события, требующие передачи дополнительных параметров, например событие, связанное с перемещением указателя мыши, передаёт координаты указателя.

Способы доступа к данным. Сущность навигационного способа доступа к данным.

Навигационный способ доступа заключается в обработки каждой записи набора данных. Достоинством этого способа является простота кодирования операции, а основной недостаток состоит в том, что приложение получает все записи набора независимо от того, сколько их на

самом деле требуется обработать. Это приводит к большой нагрузке на сеть, особенно при интенсивном обмене данными. Поэтому применение навигационного способа доступа обычно ограничивается локальными БД.

При навигационном способе доступа операции выполняются с отдельными записями. Каждый набор данных имеет указатель текущей записи, то есть записи, с полями которой могут быть выполнены такие операции, как редактирование или удаление. Компоненты Table и Query позволяют управлять положением этого указателя.[5]

Навигационный способ доступа даёт возможность осуществлять следующие операции:

- сортировка записей;
- навигация по набору данных;
- редактирование записей;
- вставка и удаление записей;
- фильтрация записей;
- поиск записей.

Редактирование, вставка и удаление записей.

Редактирование записей заключается в изменении значений их полей. Редактирована, может быть только текущая запись, поэтому перед действиями, связанными с редактированием, обычно выполняются операции по поиску и перемещению на требуемую запись. После того. Как указатель текущей записи установлен на нужную запись, и набор данных находится в режиме просмотра, для редактирования записи следует:

- перевести набор данных в редактирование;
- изменить значение полей записи;
- подтвердить изменения или отказаться от них.

Метод Insert переводит набор данных в режим вставки и добавляет к нему новую пустую запись. Для добавления записи нужно:

- перевести набор данных в режим записи;
- задать значение полей новой записи;

- подтвердить изменения или отказаться от них.

Удаление текущей записи выполняет метод Delete, который работает только с модифицируемым набором данных. В случае успешного удаления записи текущей становится следующая запись, если же удалялась последняя запись, то курсор перемещается на предыдущую запись, которая после удаления становится последней. В отличие от некоторых СУБД, в Delphi удаляемая запись действительно удаляется из набора данных.

Сортировка записей и навигация по набору данных.

Порядок расположения записей может быть неопределённым. По умолчанию записи не отсортированы или сортируются. С отсортированными наборами записей работать более удобно. Сортировка заключается в упорядочивании записей по определённому полю в порядке возрастания или убывания содержащихся в нём значений. Сортировку можно выполнять и по нескольким полям.[5]

При сортировки по двум полям записи сначала упорядочиваются по значениям первого поля, а затем группы записей с одинаковым значением первого поля сортируются по второму полю. Сортировка наборов данных Table и Query выполняется различными способами. Сортировка наборов данных Table выполняется автоматически по текущему индексу. При смене индекса происходит автоматическое переупорядочивание записей. Таким образом, сортировка возможна по полям, для которых создан индекс.

Для сортировки по нескольким полям нужно создавать индекс, включающий эти поля. Направление сортировки определяется параметром ixDescending текущего индекса, по умолчанию он выключен, и упорядочивание выполняется в порядке возрастания значений. Если для индекса признак ixDescending включен, то сортировка выполняется в порядке убывания значений. Поля, по которым сортируются записи, устанавливаются через свойств IndexName.

Перемещение по набору данных заключается в управлении указателем текущей записи. Этот указатель определяет запись, с которой будут выполняться такие операции, как редактирование или удаление.

Перед перемещение указателя текущей записи набор данных автоматически переводится в режим просмотра. Если текущая запись находилась в режимах редактирования или вставки, то перед перемещением указателя сделанные в записи изменения вступают в силу, для чего набор данных автоматически вызывает метод CheckBrowseMode. Для перемещения указателя текущей записи в наборе данных используются методы:

- Процедура First – установка на первую строку;
- Процедура Next – установка на следующую строку;
- Процедура Last – установка на последнюю строку;
- Процедура Prior – установка на предыдущую строку.

При перемещении по записям набора данных связанные с ним визуальные компоненты отображают изменения данных, причем смена отображения может происходить достаточно быстро.

Побочным эффектом выполнения ряда операций с наборами данных является изменение положения указателя текущей записи. Часто этот эффект ненадёжен, так как после выполнения такой операции указатель находится не в том месте, где был до начала операции. При этом приходится снова отыскивать нужную запись и ставить на нее указатель, что неудобно даже при небольшом количестве записей.

Фильтрация записей.

Фильтрация – это задание ограничений для записей, отбираемых в набор данных. Набор данных представляет собой записи, набранные из одной или нескольких таблиц. Состав записей в наборе данных в данный момент времени зависит от установленных ограничений, в том числе и от фильтров. Система Delphi дает возможность осуществлять фильтрацию записей:

- по выражению;

- по диапазону.

По умолчанию фильтрация записей не ведётся, и набор данных Table содержит все записи связанной с ним таблицы БД, а в набор данных Query включаются все записи, удовлетворяющие SQL-запросу, содержащемуся в свойстве SQL. Фильтрация похожа на SQL–запросы, но является менее мощным средством. По сравнению с SQL–запросами фильтрация менее эффективна, так как ограничивается количество записей, видимых в наборе.

При использовании фильтрации по выражению набор данных ограничивается записями, удовлетворяющими выражению фильтра, задающему условия отбора записей.

Достоинством фильтрации по выражению является то, что она применима к любым полям, в том числе и неиндексированным. В связи с тем, что в процессе отбора просматриваются все записи таблицы, фильтрация по выражению эффективна при небольшом количестве записей.

Для задания выражения фильтра представляет собой конструкцию, в состав которой могут входить следующие элементы:

- имена полей таблицы;
- литералы;
- операции сравнения;
- арифметические операции;
- логические операции;
- круглые и квадратные скобки.

Если имя поля содержит пробелы, то его заключают в квадратные скобки. Литерал представляет собой значение, чаще всего - число, строка или символ. Имена переменных в выражении фильтра использовать нельзя. Если в выражении фильтра требуется включить значение переменной или свойства какого-либо компонента, то это значение должно быть преобразовано в строковый тип. Операции сравнения представляют собой обычные отношения <,>,=,<=,>=,<>. Арифметическими являются операции +, -,* и /. В качестве логических операций можно использовать AND, OR и NOT.

Круглые скобки применяются для изменения порядка выполнения арифметических и логических операций.

Набор данных Table допускается два способа задания условии фильтрации: су помощью выражения фильтра Filter и в обработчике события OnFilterRecord.

В случае набора данных Query для записей можно использовать: SQL-запрос, обработчик событияOnFilterRecord, выражения фильтра.

При фильтрации по диапазону в набор данных включаются записи, значения полей которых попадают в заданный диапазон, т.е. условием фильтрации является выражение вида «значение>нижняя граница AND значение < верхней границы». Такая фильтрация применяется к наборам данных Table.

Достоинство фильтрации по диапазону является высокая скорость обработки записей. В отличие от фильтрации по выражению, когда последовательно просматриваются все записи таблицы, фильтрация по диапазону ведётся индексно-последовательным методом, поэтому этот способ фильтрации применим только для индексированных полей. Индекс поля, диапазон которого задан как текущий с помощью свойства IndexName или IndexFieldName. Если текущий индекс не установлен, то по умолчанию используется главный индекс.

Для включения и выключения фильтрации по диапазону применяются методы ApplyRange и CancelRange. Первый из них активизирует фильтрацию, а другой - деактивизирует. Предварительно для индексного поля или полей, по которому выполняется фильтрация, следует задать диапазон допустимых значений. Когда одна из границ диапазона не задана, то диапазон открыт, то есть нижняя граница становится равной минимально возможному, а верхняя граница – максимально возможному значению этого поля.

Поиск записей.

Поиск записи, удовлетворяющей определённым условиям, означает переход на эту записи. Поиск во многом похож на фильтрацию, так как в процессе поиска также выполняется проверка полей записей на некоторые условия. Отличие заключается в том, что в результате поиска количество записей в наборе данных не изменяется.

При организации поиска записей важное значение имеет наличие индекса для полей, по которым ведётся поиск. Индексирование значительно повышает скорость обработки данных, кроме того, ряд методов может работать только с индексированными полями.

Для поиска записей по полям служат методы Locate и Lookup, причем поля могут быть не индексированными. Функция Locate ищет запись с заданными значениями полей. Если удовлетворяющие условиям поиска записи существует, то указатель текущей записи устанавливается на первую из них. Если запись найдена, функция возвращает значение True, в противном случае значение False. Обычно при разработки приложения пользователю предоставляется возможность влиять на процесс поиска су помощью управляющих элементов, расположенных на форме. При этом действия пользователя по управлению поиском в наборе данных мало, чем отличаются от аналогичных действий при выполнении фильтрации. Функция Lookup так же осуществляет поиску записи, удовлетворяющей определённым условиям, но, в отличие от метода Locate, не перемещает указатель текущей записи, а считывает информацию из полей записи. Еще одно отличие между двумя методами заключается в том, что метод Lookup осуществляет поиск на точное соответствие значений для поиска и значений в полях записей с учетом регистра букв.

Для набора данных Table имеются методы, позволяющие вести поиск записей только по индексным полям. Перед вызовом любого из этих методов следует установить в качестве текущего индекс, построенный по используемым для поиска полям. Методы поиска можно разделить на две группы, в первой группе методы, предназначенные для поиска на точное

соответствие, а другую образуют методы, допускающие только частичное совпадение заданных для поиска значений и значений полей записей. Если заданным условиям поиска соответствует несколько записей, то указатель текущей записи устанавливается на первую из них. Порядок сортировки и, соответственно, порядок расположения записей в наборе даны, определяется текущим индексом, по которому и ведется поиск.[5]

§4. Проектирование и разработка программного обеспечения для анализа статистических данных профессиональных колледжей

Этапы проектирования и создания базы данных определяются следующей последовательностью:

- построение информационно-логической модели данных предметной области;
- определение логической структуры реляционной базы данных;
- конструирование таблиц базы данных;
- создание схемы данных;
- ввод данных в таблицы (создание записей);
- разработка необходимых форм, запросов, макросов, модулей, отчетов;
- разработка пользовательского интерфейса.

В процессе разработки модели данных необходимо выделить информационные объекты, соответствующие требованиям нормализации данных, и определить связи между ними. Эта модель позволяет создать реляционную базу данных без дублирования, в которой обеспечивается однократный ввод данных при первоначальной загрузке и корректировках, а также целостность данных при внесении изменений.[2]

При разработке модели данных могут использоваться два подхода. В первом подходе сначала определяются основные задачи, для решения которых строится база, выявляются потребности задач в данных и соответственно определяются состав и структура информационных объектов. При втором подходе сразу устанавливаются типовые объекты предметной области. Наиболее рационально сочетание обоих подходов. Это связано с тем, что на начальном этапе, как правило, нет исчерпывающих сведений обо всех задачах. Использование такой технологии тем более оправдано, что гибкие средства создания реляционных баз данных позволяют на любом этапе разработки внести изменения в базу данных и модифицировать ее структуру без ущерба для введенных ранее данных.[2]

Процесс выделения информационных объектов предметной области, отвечающих требованиям нормализации, может производиться на основе интуитивного или формального подхода. Теоретические основы формального подхода были разработаны и полно изложены в монографиях по организации баз данных известного американского ученого Дж. Мартина.

При интуитивном подходе легко могут быть выявлены информационные объекты, соответствующие реальным объектам. Однако получаемая при этом информационно-логическая модель, как правило, требует дальнейших преобразований, в частности преобразования многозначных связей между объектами. При таком подходе возможны существенные ошибки, если отсутствует достаточный опыт. Последующая проверка выполнения требований нормализации обычно показывает необходимость уточнения информационных объектов.

Рассмотрим формальные правила, которые могут быть использованы для выделения информационных объектов:

- на основе описания предметной области выявить документы и их атрибуты, подлежащие хранению в базе данных;

-определить функциональные зависимости между атрибутами;

- выбрать все зависимые атрибуты и указать для каждого все его ключевые атрибуты, т. е. те, от которых он зависит;

- сгруппировать атрибуты, одинаково зависимые от ключевых атрибутов. Полученные группы зависимых атрибутов вместе с их ключевыми атрибутами образуют информационные объекты[2].

При определении логической структуры реляционной базы данных па основе модели каждый информационный объем адекватно отображается реляционной таблицей, а связи между таблицами соответствуют связям между информационными объектами.

В процессе создания сначала конструируются таблицы базы данных, соответствующие информационным объектам построенной модели данных. Далее может создаваться схема данных, в которой фиксируются

существующие логические связи между таблицами. Эти связи соответствуют связям информационных объектов. В схеме данных могут быть заданы параметры поддержания целостности базы данных, если модель данных была разработана в соответствии с требованиями нормализации. Целостность данных означает, что в БД установлены и корректно поддерживаются взаимосвязи между записями разных таблиц при загрузке, добавлении и удалении записей в связанных таблицах, а также при изменении значений ключевых полей.[2]

После формирования схемы данных осуществляется ввод непротиворечивых данных из документов предметной области.

На основе созданной базы данных формируются необходимые запросы, формы, макросы, модули, отчеты, производящие требуемую обработку данных базы и их представление.

С помощью встроенных средств и инструментов базы данных создается пользовательский интерфейс, позволяющий управлять процессами ввода, хранения, обработки, обновления и представления информации базы данных.[2]

Данная программа создана для учета успеваемости учеников профессиональных колледжей.

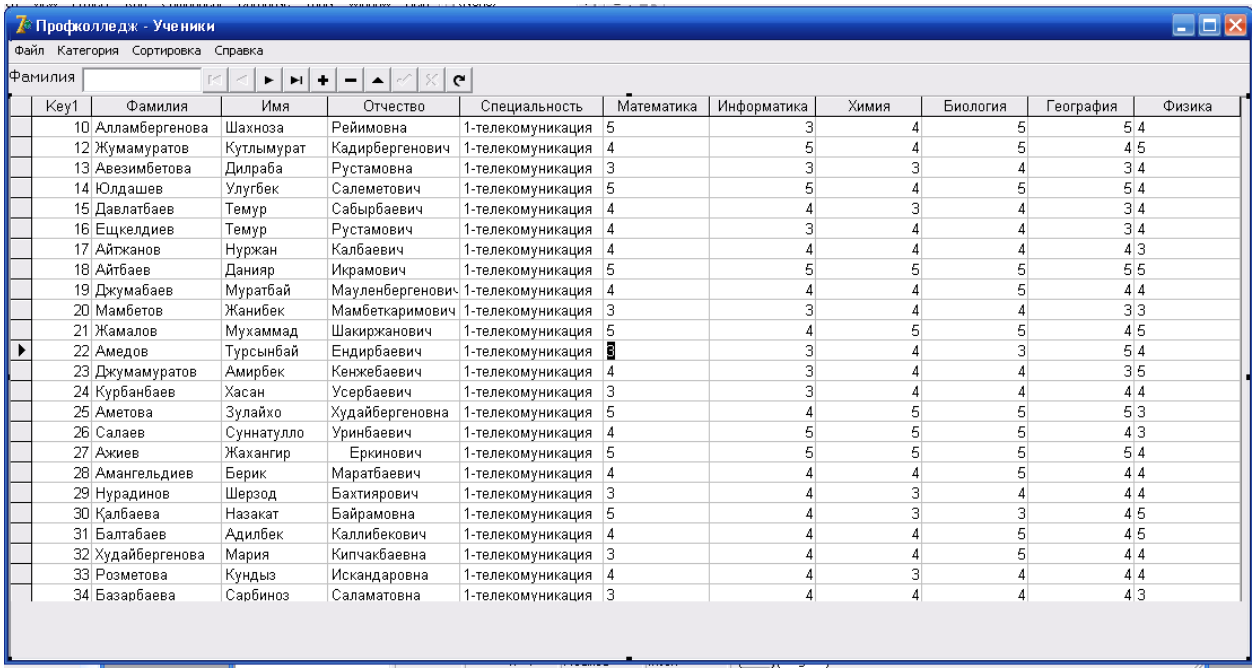
Для работы с программой необходимо нужные группы или списки студентов копировать из имеющегося списка, в электронную таблицу «spisok.xls». Списки копируются однажды, после чего могут удаляться напрямую из базы данных «poseshaemost.mdb». После копирования списков или групп необходимо принять данные с помощью кнопки «Сервис - принять данные», в появившемся окне нажать кнопку «Принять данные». Для ввода специальности для выбранной группы нажать на кнопку «справочники-специальности», в появившемся окне ввести группу и код специальности, списки копируются однажды, после чего могут удаляться напрямую из базы данных «poseshaemost.mdb». Для добавления дисциплины для этой группы необходимо нажать на кнопку «справочники- дисциплины», в появившемся

окне ввести дисциплины. Дисциплины копируются однажды, после чего могут удаляться напрямую из базы данных «poseshaemost.mdb».

Для ввода оценок нужно нажать на кнопку «Данные – Разноска», где выбирается месяц и год обучения, а также дисциплина, специальность и группа. После ввода данных в эти строки следует «Добавить данные по дисциплине», на экране выведется группа, фамилия имя и отчество студентов, а также пустая графа оценок по выбранной дисциплине, которую необходимо заполнить. Корректировка данных производиться при помощи команды «Данные - корректировка», где корректируются месяц и год обучения, а также дисциплина, специальность и группа. В графе отчеты можно подвести итоги по успеванию и качеству знаний по БПЭК. Подвести итоги можно как по группе, так и по специальности и по колледжу в целом за определенный месяц и год обучения. После нажатия кнопки «Вычислить». Выведется успеваемость и качество. Также в графе подведение итогов можно посмотреть списки студентов неуспевающих (у которых есть хотя бы одна оценка «2»), а также успевающих на отлично (средний бал успеваемости которых «5»). После нажатия кнопки «Вывести», выведется список в электронной таблице Excel, неуспевающих студентов, либо успевающих на отлично. В графе «Отчет» за введенный месяц после нажатия кнопки «Пересчитать итоги» и «Отчет» выведется группа, месяц и год, а также количество студентов и процентное соотношение успеваемости общей и качество в электронной таблице Excel.

```
program SchoolProject;
uses
  Forms,
  StudentUnit in 'StudentUnit.pas' {StudentForm},
  DataModuleUnit in 'DataModuleUnit.pas' {DataModule1: TDataModule},
  PrepodUnit in 'PrepodUnit.pas' {PrepodForm},
  UserUnit in 'UserUnit.pas' {UserForm},
  Unit1 in 'Unit1.pas' {PersonalForm},
```

```
  Unit2 in 'Unit2.pas' {UsersForm};
{$R *.res}
begin
  Application.Initialize;
  UserForm:=TUserForm.Create(Application);
  UserForm.Visible:=false;
  UserForm.ShowModal;
  UserForm.Hide;
  UserForm.Free;
  Application.CreateForm(TStudentForm, StudentForm);
  Application.CreateForm(TDataModule1, DataModule1);
  Application.CreateForm(TPrepodForm, PrepodForm);
  Application.CreateForm(TPersonalForm, PersonalForm);
  Application.CreateForm(TUsersForm, UsersForm);
  Application.Run;
end.
```



Key1	Фамилия	Имя	Отчество	Специальность	Математика	Информатика	Химия	Биология	География	Физика
10	Алламбергенова	Шахноза	Реймовна	1-телекомуникация	5	3	4	5	5	4
12	Жумамуратов	Кутлымурат	Кадирбергенович	1-телекомуникация	4	5	4	5	4	5
13	Авезимбетова	Дилраба	Рустамовна	1-телекомуникация	3	3	3	4	3	4
14	Юлдашев	Улугбек	Салеметович	1-телекомуникация	5	5	4	5	5	4
15	Давлатбаев	Темур	Сабырбаевич	1-телекомуникация	4	4	3	4	3	4
16	Ещкелдиев	Темур	Рустамович	1-телекомуникация	4	3	4	4	3	4
17	Айтжанов	Нуржан	Калбаевич	1-телекомуникация	4	4	4	4	4	3
18	Айтбаев	Данияр	Икрамович	1-телекомуникация	5	5	5	5	5	5
19	Джумабаев	Муратбай	Мауленбергенович	1-телекомуникация	4	4	4	5	4	4
20	Мамбетов	Жанибек	Мамбеткаримович	1-телекомуникация	3	3	4	4	3	3
21	Жамалов	Мухаммад	Шакиржанович	1-телекомуникация	5	4	5	5	4	5
22	Амедов	Турсынбай	Ендирбаевич	1-телекомуникация	5	3	4	3	5	4
23	Джумамуратов	Амирбек	Кенжебаевич	1-телекомуникация	4	3	4	4	3	5
24	Курбанбаев	Хасан	Усербаевич	1-телекомуникация	3	3	4	4	4	4
25	Аметова	Зулайхо	Худайбергеновна	1-телекомуникация	5	4	5	5	5	3
26	Салаев	Суннатулло	Уринбаевич	1-телекомуникация	4	5	5	5	4	3
27	Ажиев	Жахангир	Еркинович	1-телекомуникация	5	5	5	5	5	4
28	Амангельдиев	Берик	Маратбаевич	1-телекомуникация	4	4	4	5	4	4
29	Нурадинов	Шерзод	Бахтиярович	1-телекомуникация	3	4	3	4	4	4
30	Қалбаева	Назакат	Байрамовна	1-телекомуникация	5	4	3	3	4	5
31	Балтабаев	Адилбек	Каллибекович	1-телекомуникация	4	4	4	5	4	5
32	Худайбергенова	Мария	Кипчакбаевна	1-телекомуникация	3	4	4	5	4	4
33	Розметова	Кундыз	Искандаровна	1-телекомуникация	4	4	3	4	4	4
34	Базарбаева	Сарбиноз	Саламатовна	1-телекомуникация	3	4	4	4	4	3

Рис.-4.1. Общий вид программы для анализа статистических данных профессиональных колледжей

Описание результатов исследований. В результате разработки программного продукта, была создана программа по учету успеваемости студентов. В данной программе есть возможность вывода итогов в электронную таблицу Excel. Позволяет формировать списки студентов, учащихся на отлично и на неудовлетворительно.

В результате формирования списка выводится отчет со списком студентов, учащихся на отлично или на удовлетворительно.

Также можно сформировать табель успеваемости за месяц.

Также ведется подсчет успеваемости и качества по группе, специальности и колледжа в целом за любой введенный месяц и год обучения.

ЗАКЛЮЧЕНИЕ

Задачей выпускного квалификационного работа являлась разработка программного продукта по учету успеваемости студентов профессиональных колледжей. Для реализации этой задачи были выполнены следующие этапы:

Спроектирована логическая модель базы данных;

Спроектирована физическая модель базы данных;

Организован ввод оценки за каждый месяц по каждой специальности, группе, студенту, предмету;

Предусмотрена возможность корректировки данных;

Организована возможность подведения итогов успеваемости по каждой группе, специальности и в целом по колледжу (процентное содержание успеваемости и качественной успеваемости)

Предусмотрена возможность сравнения итоговых данных с итогами предыдущего месяца

Обеспечено формирование списков неуспевающих и успевающих на оценку «отлично»

В результате была создана программа «Учет успеваемости», которая может быть использована в подобных учебных заведениях.

В ходе выполнения дипломной работы был создан программный продукт, дающий возможность вести учет успеваемости. Программа проста и понятна для любого пользователя, для ее использования не нужно специального обучения. При этом она выполняет в основном все функции, необходимые для нормальной работы.

Для создания базы данных использовался Microsoft Office Access. Для написания программы использовался язык программирования Delphi, имеющий широкие возможности для работы с базами данных. Соединение с базой данных произведено при помощи технологии ADO. Обработка данных производилась при помощи операторов языка SQL, что значительно ускоряет работу программы.

СПИСОК ЛИТЕРАТУРЫ

1 Гофман В. Э., Хомоненко А. Д. Delphi. Быстрый старт. — СПб.: БХВ-Петербург, 2003. — 288 с.

2 Гофман В. Э., Хомоненко А. Д. Работа с базами данных в Delphi. — СПб.: БХВ-Петербург, 2001. — 656 с.

3 Боровский А. Н. Программирование в Delphi 2005. — СПб.: БХВ-Петербург, 2005. - 448 с.

4 Дарахвелидзе П. Г., Марков Е. П. Delphi 2005 для Win32. - СПб.: БХВ-Петербург, 2005. - 1136 с.

5 Сорокин А. В. Delphi. Разработка баз данных. — СПб.: Питер, 2005. — 477 с.

6 Фленов М. Е. Библия Delphi. — СПб.: БХВ-Петербург, 2004. — 880 с.

7 Фленов М. Е. Программирование в Delphi глазами хакера. — СПб.: БХВ-Петербург, 2003. - 368 с.

8 Фленов М. Е. Delphi в шутку и всерьез: что умеют хакеры (+CD). — СПб.: Питер. 2006. — 271 с.

9 Архангельский Л.Я. Delphi 2006. Справочное пособие: Язык Delphi, классы, функции Win32 и .NET. — М.: ООО «Бином-Пресс», 2006 г. — 1 152 с.

10 Фаронов В. В. Delphi 2005. Разработка приложений для баз данных и Интернета. — СПб.: Питер, 2006. — 603 с.

11 Карпова Т. С. Базы данных: модели, разработка, реализация. — СПб.: Питер, 2001. — 304 с.

12 Иллюстрированный самоучитель по Delphi для начинающих. Электронное пособие.

13 Иллюстрированный самоучитель по Delphi для профессионалов. Электронное пособие.

12. http://www.ziyonet.uz/

Приложение

```
unit Unit1;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, Menus, DB, ADODB;
type
 TForm12 = class(TForm)
 MainMenu1: TMainMenu;
 N1: TMenuItem;
 N2: TMenuItem;
 N3: TMenuItem;
 N4: TMenuItem;
 N8: TMenuItem;
 N9: TMenuItem;
 ADOConnection1: TADOConnection;
 N10: TMenuItem;
 N11: TMenuItem;
 N12: TMenuItem;
 procedure N9Click(Sender: TObject);
 procedure N11Click(Sender: TObject);
 procedure N12Click(Sender: TObject);
 procedure FormCreate(Sender: TObject);
 procedure N2Click(Sender: TObject);
 procedure N3Click(Sender: TObject);
 procedure N4Click(Sender: TObject);
 private
```

```
 { Private declarations }
 public
 { Public declarations }
 end;
var
 Form12: TForm12;
implementation
uses Unit3, Unit5, Unit4, Unit6, Unit7, Unit8;
{$R *.dfm}
procedure TForm12.N9Click(Sender: TObject);
begin
 Form13.Show;
end;
procedure TForm12.N11Click(Sender: TObject);
begin
Form15.Show;
end;
procedure TForm12.N12Click(Sender: TObject);
begin
Form14.Show;
end;
procedure TForm12.FormCreate(Sender: TObject);
begin
//Form16.Show;
end;
procedure TForm12.N2Click(Sender: TObject);
begin
Form16.Show;
end;
procedure TForm12.N3Click(Sender: TObject);
```

```
begin
Form17.show;
end;
procedure TForm12.N4Click(Sender: TObject);
begin
Form18.show;
end;
end.
unit Unit2;
interfa
ceuses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,Unit1,
Forms,
 Dialogs, ExcelXP, OleServer, Grids, DBGrids, StdCtrls, DB, ADODB;
type
 TForm13 = class(TForm)
 Label1: TLabel;
 DBGrid1: TDBGrid;
 ExcelWorkbook1: TExcelWorkbook;
 ExcelApplication1: TExcelApplication;
 ADOQuery1: TADOQuery;
 DataSource1: TDataSource;
 Button1: TButton;
 ADOQuery2: TADOQuery;
 ADOQuery1gruppa: TWideStringField;
 ADOQuery1fio: TWideStringField;
 ADOQuery1Datapr: TWideStringField;
 ADOQuery1Prim: TWideStringField;
 procedure Button1Click(Sender: TObject);
 private
```

```
 { Private declarations }
 public
 { Public declarations }
 end;
var
 Form13: TForm13;
implementation
{$R *.dfm}
procedure TForm13.Button1Click(Sender: TObject);
var n: OleVariant;
i:integer;
//S:String;
begin
ADOQuery2.ExecSQL;
AdoQuery1.Open;
n:='d:\55\spisok.xls';
ExcelApplication1.Workbooks.Add(n,0);
Excelworkbook1.ConnectTo(ExcelApplication1.ActiveWorkbook);
i:=2;
ADOQUERY1.Insert;
ExcelApplication1.Visible[0]:=False;
While i<2000 do
begin;
ADOQuery1.FieldByName('gruppa').AsString:=ExcelApplication1.Cells.Ite
m[i,1].Value;
ADOQuery1.FieldByName('Fio').AsString:=ExcelApplication1.Cells.Item[i,
2].Value;
ADOQuery1.FieldByName('Datapr').AsString:=ExcelApplication1.Cells.Ite
m[i,3].Value;
```

```
ADOQuery1.FieldByName('Prim').AsString:=ExcelApplication1.Cells.Item[
i,4].Value;
 If ADOQuery1.FieldByName('gruppa').AsString='' then
 begin
 ADOQUERY1.Delete;
 i:=2001;
 end;
ADOQUERY1.Insert;
i:=i+1;
 end;
ADOQuery1.Open;
Showmessage('Прием данных выполнен');
end;
end.
```