condition pf monotonously decreasing, and from that functional we may derive the recurrent formulas for unknown coefficients of initial condition.

According to the National Annex of The Republic of Kazakhstan (NTP RK 01-01-5.1-2013) “General Loads. Temperature Effects”, the problem of determination of the deformation of construction resulting from the thermal effects, should be solved with the help of choosing the most rational model that properly describes characteristic distribution of the thermal loads and temperature field. Temperature effects on buildings caused by climatic and operational changes in temperature should be taken into account when determining the design parameters of a building if there is a possibility of exceeding the limiting states in terms of bearing capacity and usability due to temperature movements and / or stresses.

However, with the help of the theory of inverse problems, we may obtain unknown data by solving direct and conjugate problems and by performing elements of the functional analyses. An inverse problem in science is the process of calculating from a set of observations the causal factors that produced them: for example, calculating an image in X-ray computed tomography, source reconstruction in acoustics, or calculating the density of the Earth from measurements of its gravity field.

It is called an inverse problem because it starts with the results and then calculates the causes. This is the inverse of a forward problem, which starts with the causes and then calculates the results. Inverse problems are some of the most important mathematical problems in science and mathematics because they tell us about parameters that we cannot directly observe

Список литературы:

NTP RK 01-01-5.1-2013, “General Loads. Temperature Effects”;

Kaipio, J., & Somersalo, E. (2010). Statistical and computational inverse problems. New York, NY: Springer.

Tahmasebi, Pejman; Javadpour, Farzam; Sahimi, Muhammad (August 2016). "Stochastic shale permeability matching: Three-dimensional characterization and modeling". International Journal of Coal Geology. 165: 231–242. doi:10.1016/j.coal.2016.08.024.

Patric Figueiredo (December 2014). Development Of An Iterative Method For Solving Multidimensional Inverse Heat Conduction Problems. Lehrstuhl für Wärme- und Stoffübertragung RWTH Aachen

УДК 004.65

ОСОБЕННОСТИ СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ MS ACCESS ИЗ DELPHI

Скибин Д.А.

Костанайский Государственный Педагогический Университет

им. У. Султангазина, г. Костанай

Научный руководитель: Бегалин А.Ш.
Костанайский Государственный Педагогический Университет
им. У. Султангазина, г. Костанай

Аннотация. В данной работе подробно рассказывается план создания структуры управления базой данных MS ACCESS с помощью среды программирования DELPHI 7. Проектируемая база данных позволяет осуществлять хранение информации о выпускниках и дает возможность просматривать список, добавлять, изменять, удалять и осуществлять поиск информации о выпускниках и преподователях ГУ «Боровская школа-гимназия имени А. Чутаева» в базе данных.

Ключевые слова: база данных, система управления, MS Access, Delphi, связь, код, процедуры.

Annotation. This paper describes in detail the plan for creating a database management structure for MS Access using the Delphi programming language. The projected database allows storing information about graduates and provides an opportunity to view the list, add, change, delete and search for information about graduates of Public institution «Borovskaya school-gymnasium named after A. Chutaev» in the database.

Keywords: database, management system, MS Access, Delphi, communication, code, procedures.

Аннотация. Бұл жұмыста Delphi бағдарламалау тілінің көмегімен MS Access деректер базасын басқару құрылымын құру жоспары егжей-тегжейлі баяндалады. Жобаланатын деректер базасы бітірушілер туралы ақпаратты сақтауды жүзеге асыруға мүмкіндік береді және мәліметтер базасында "А. Шотаев атындағы Боровской мектеп-гимназиясы" ММ түлектері туралы ақпаратты қарауға, қосуға, өзгертуге, жоюға және іздестіруге мүмкіндік береді.

Түйін сөздер: деректер қоры, басқару жүйесі, MS Access, Delphi, связь, код, процедуралар.

В последнее время темпы роста количества баз данных в Республике Казахстан увеличились. Наряду с увеличением количества различных баз данных, число организаций, использующих базу данных, также растет. Им нужна многофункциональная база данных по доступной цене. Крупные организации могут заказать базу данных, которая специализируется именно на этом организации. Менее развитые компании используют готовые базы данных и настраивают их под себя. А как насчет малого бизнеса, школ, которые не могут позволить себе дорогие программы? Есть ли возможность создания небольшой специализированной базы данных, которая не требует специальных знаний для работы с ними. История знает немало фактов, когда недорогая, хорошо спланированная база данных превосходит более дорогую общую базу данных.

Так же о своевременности и актуальности рассматриваемой проблемы говорит тот факт, что большую часть своего времени организации тратят на оформление различной документации и отчетов, в частности в ГУ «Боровская школа-гимназия имени А. Чутаева» отдела образования акимата Мендыкаринского района, не имеется

автоматизированной базы для хранения данных о выпускниках. Основанием для разработки программного продукта «База данных выпускников ГУ «Боровская школа-гимназия имени А. Чутаева» послужило задание на дипломную работу. В качестве языка программирования был выбран язык Delphi с базой данных MS Access.

Базу данных мы спроектировали, таблицы сделали. Осталась еще половина работы - проект Delphi, работающий с этой базой данных. Загружаем Delphi, делаем новый проект. В основной форме мы размещаем три простые панели. Свойству Align верхней панели присвоил значение alTop (весь верх). Затем свойству Align нижней панели присвоил значение alBottom. Затем поместил компонент Splitter с вкладки Additional панели инструментов, и его свойству Align также присвоил alBottom. С его помощью пользователь может изменить размер нижней панели с помощью мыши. И, наконец, свойству Align средней панели присвоил значение alClient.

В последнем разделе верхней панели есть еще две кнопки BitBtn. Первая - редактировать текущую запись, вторая - добавлять новую.

Вторая и третья панели содержат только один компонент DBGrid на вкладке DataControls палитры компонентов, свойства которого настраиваются на уровне AlClient.

Свойству Name формы присвоено значение fMain, свойство Caption формы имеет текст «Выпускные классы», модуль сохранен под именем Main.pas, а проект в целом называется vk (Выпускные классы).

Далее в проект добавлен модуль данных (File -> New -> Data Module). Он предназначен для размещения в нем компонентов подключения к данным, компонентов - наборов данных (TTable/ADOTable, TQuery/ADOQuery, TStoredProc/ADOStoredProc) и компонентов DataSource, которые обеспечивают связь наборов данных и компонентов отображения/редактирования данных.

Свойству Name модуля данных мы присвоим имя fDM, а модуль сохраним как DM.pas. Добавляем в модуль компонент ADOConnection с вкладки ADO палитры компонентов. Этот компонент обеспечит связь других компонентов с базой данных при помощи механизма ADO. Связь обеспечивается свойством компонента ConnectionString.

Двойной щелчок свойства ConnectionString компонента ADOConnectión открывает окно для добавления компонента в ADO. При нажатии кнопки «Создать» открывается новое окно с настройками соединения.

С помощью клавиши <Shift> выделяем все четыре ADOTable, и в их свойстве Connection выберите нашу связь ADOConnection1. Таким образом, все четыре ADOTable мы подключили к базе данных.

Выделите первый компонент ADOTable. Переименуйте его свойство Name в TLichData, а в свойстве TableName выберите главную таблицу базы - LichData. Рядом с компонентом ставим компонент DataSource из вкладки Data Access палитры компонентов. Компонент DataSource предназначен для организации связи с наборами данных, и служит посредником между такими компонентами НД, как ADOTable, ADOQuery и между компонентами отображения данных. Свойство Name компонента

DataSource переименуйте в DSLichData (DS - DataSource). В свойстве DataSet выберите таблицу TLichData.

То же самое нужно проделать еще три раза, подключая аналогичным образом компоненты DataSource к другим таблицам. Затем свойство Active таблиц переведите в True, открыв их.

Переходим на главную форму и выбираем команду File -> Use Unit, подключаем к ней модуль DM. Теперь мы сможем видеть таблицы из главной формы. На вкладке DataControls сосредоточены визуальные компоненты отображения данных, такие как DBGrid, DBEdit, DBMemo, за исключением компонента - DBNavigator.

Выделите верхнюю сетку DBGrid, в ее свойстве DataSource выберите fDM.DSLichData. В таком же свойстве нижней сетки выберите fDM.DSAdres. Сетки среагировали, и вы можете видеть названия полей.

Теперь нужно между таблицами установить связь. Это требуется не только для того, чтобы в нижней сетке выходили данные только на учащегося, выделенного в верхней сетке, но и для того, чтобы мы смогли в дальнейшем вводить связанные данные в окне редактора. Снова выделите модуль данных. Щелкните дважды по первой таблице, чтобы открыть редактор полей. Правой кнопкой щелкните по этому редактору и выберите команду Add all fields (добавить все поля). В окне редактора полей появились все поля таблицы.

Поле «Ключ» у нас автоинкрементное, предназначено для связи с другими таблицами. Пользователю его видеть не обязательно. Выделите его, и в свойстве Visible установите False. Теперь для пользователя оно будет невидимым. Здесь у нас есть два логических поля - «Класс» и «Пол». Чтобы True и False выходили на экране так, как нам нужно, свойству DisplayValues первого из этих полей присвойте значение «9;11», а второго - «Мужской;Женский». Первым здесь идет значение, которое будет обозначать True, вторым - False. Эти значения разделяются точкой с запятой, пробелы не нужны.

Таким же образом добавьте все поля в остальные три таблицы. У них невидимым следует сделать поле «Выпускник» - этому полю автоматически будет присвоено такое же число, как у поля Ключ соответствующей записи. Логических полей у них нет. Однако для поля «Телефон» таблицы Telephones следует изменить свойство EditMask. Щелкните по нему дважды, открыв редактор маски, и в поле Input Mask введите маску «#(###)-###-##-##». Сохраните ее, нажав кнопку ОК. Для полей типа Дата в этом свойстве (в таблице LichData два таких поля) введите маску «##.##.####».

Далее кнопкой <F12> перейдите в редактор кода. В нижней части окна вы можете увидеть вкладку Diagram, перейдите на нее.

Для начала в окно диаграмм нужно добавить наши таблицы. Найдите их в окне дерева объектов Object TreeView. Если у вас это окно закрыто, откройте его клавишами <Shift+Alt+F11> либо командой меню View -> Object TreeView. Ухватитесь в этом окне мышью за название главной таблицы LichData {TLichData} и перетащите ее в окно диаграмм. Таблица вместе с полями отобразится в окне.

Здесь в поле Detail Fields нужно выбрать поле, по которому будет осуществляться связь, в нашем случае это поле «Выпускник». В поле Master Fields выбираем ключевое поле «Ключ». Затем нажимаем кнопку Add и кнопку OK. Связь установлена.

Пойдем дальше. Теперь нам нужно сделать окно редактора данных. Создайте новую форму (File -> New -> Form). Ее свойство Name переименуйте в fEditor, а при сохранении формы дайте модулю имя Editor. Командой File -> Use Unit подключите к форме модуль данных DM.

Здесь сделали следующее: установили четыре панели GroupBox на вкладке «Standard», GrOupBox на каждой таблице. Введите «Личные данные» в свойстве Caption компонента GroupBox, который появится в имени панели. Затем вам нужно установить восемь компонентов DBEdit с вкладки DataControls палитры компонентов на этой панели, два DBCheckBgex для логической обработки данных и один компонент DBComboBox для списка. Вы сами установили и настроили пояснительные компоненты этикетки. Давайте немного изменим компонент DBComboBox. Дважды щелкните его свойство Items, чтобы открыть редактор. В нем введите две строки: муж жен. Сохраните текст, нажав кнопку OK. Теперь пользователь сможет указать пол выпускника, выбрав нужную строку из списка.

Для таблицы Doljnost все еще проще: на панели GroupBox всего два компонента DBEdit и два поясняющих Label.

Для таблицы Adres используйте три DBEdit.

А вот для таблицы Telephones понадобится один DBEdit, один DBComboBox, сетка DBGrid и кнопка BitBtn. Сетка нужна для контроля введенных телефонов, ведь здесь связь один-ко-многим, и телефонов может быть несколько. В редакторе Items компонента DBComboBox введите две строки: «Домашний», «Мобильный»

Теперь добавим компоненты управления. Удерживая нажатой клавишу <Shift>, берем все компоненты управления на первой панели (кроме Label). В свойстве DataSource берем fDM.DSLichData, который добавляет компоненты в требуемый набор данных. Удалите общий выбор и сначала выбираем DBEdit. В свойстве DataField берем поле «Фамилия». Затем добавьте компоненты других таблиц в каждую таблицу и в соответствующее поле. Сетка DBGrid, конечно, добавляется в fDM.DSTelephônes и не имеет поля.

Для удобства пользователя в нижней правой части установили навигационный компонент DBNavigator со вкладки Data Contrøls. Этот компонент используется для перемещения записей, включения режима редактирования записей, сохранения или отмены изменений, добавления новой записи или удаления существующей записи. В его свойстве DataSource выбрали fDM.DSLichData, чтобы добавить компонент в основную таблицу. Из этого компонента мы должны иметь возможность перейти к началу или концу таблицы, к следующей или предыдущей записи. Поэтому свойство VisibleButtóns (вид кнопки компонента) задаем для всех кнопок значение False, кроме nbFirst, nbPrior, nbNext и nbLast. Когда вы нажимаете эти кнопки, появляются соответствующие методы компонента ADOTable.

От ключевого поля зависят остальные таблицы. Это поле не имеет смысла, пока мы не сохраним запись. Поэтому данные в других таблицах не могут относиться к некоторым записям в основной таблице. Поэтому берем первый GroupBox и дважды щелкните событие onExit на вкладке событий инспектора объектов - Events. Это событие происходит, когда пользователь переключается на другую панель GroupBox или на кнопки в нижней части окна. Напишите код в процедуре:

```
{ LichData}
procedure TfEditor.GroupBox1Exit(Sender: TObject); begin
if fDM.TLichData.Modified then fDM.TLichData.Post;
end;
```

Измененное свойство компонента ADOTable - Modified - имеет логический тип - «True», если оно изменено, и «False» если нет. Метод Post этого компонента сохраняет измененную запись таблицы. В этом случае автоматически назначаемое значение вводится в ключевое поле. Таким образом, введенный код означает, что если запись была изменена, она должна быть сохранена. Для остальных панелей GroupBox берем событие onExit и сохраняем изменения.

Создаем событие, нажав кнопку «Добавить» в GroupBox с данными телефона. С помощью этой кнопки мы добавляем новые записи в таблицу, поскольку у одного выпускника может быть несколько телефонов. Код в процедуре выглядит следующим образом: если fDM.TTelephones.Modified / fDM.TTelephones.Post; fDM.TTelephones.Append; DBEdit14.SetFocus;

Сначала мы сохраняем измененные значения, если они есть. Затем мы добавляем новую запись в таблицу, используя метод Append.

Код в процедуре нажатия кнопки «Сохранить и выйти» прост:

```
if fDM.TLichData.Modified then
fDM.TLichData.Post; if fDM.TDoljnost.Modified then
fDM.TDoljnost.Post; if fDM.TAdres.Modified then
fDM.TAdres.Post; if fDM.TTelephones.Modified then
fDM.TTelephones.Post; Close;
```

Здесь мы просто сохраняем изменения во всех таблицах и закрываем окно. Итак, у нас есть кнопка «Добавить выпускника», код для которой - fDM.TLichData.Append; fDM.TDoljnost.Append; fDM.TAdres.Append; fDM.TTelephones.Append; DBEdit1.SetFocus;

Переходим к главной форме. Для добавления нового выпускника к аналогично названной кнопке применяем следующий код: fDM.TLichData.Append; fDM.TDoljnost.Append; fDM.TAdres.Append; fDM.TTelephones.Append; fEditor.ShowModal.

Далее приступим к программированию переключателей. Согласно нашему плану, когда вы откроете программу, данные главной таблицы появятся в верхней сетке DBGrid, а данные в нижней адресной таблице. Переключатель с пометкой «Адрес» также будет подсвечен. Если пользователь хочет увидеть класс или номер телефона текущего выпускника, он нажимает соответствующую кнопку, предназначенную для этого, и эта

информация должна отображаться в DBGrid. Берем первый переключатель с пометкой «Адрес» и создаем событие, которое происходит, когда пользователь нажимает на него. Вводим следующий код в процедуру для этого события: if RadioButton1.Checked then / DBGrid2.DataSource := fDM.DSAdres;

Здесь мы проверили, включен ли этот переключатель. Если да, мы изменим соединение подсети DBGrid и добавим его в таблицу адресов. Это потому, что таблица добавляется в таблицу через соответствующий компонент DataSource, и у нас их четыре. Подключившись к любому источнику данных, мы можем программно изменить таблицу, отображаемую в сетке

Переключатели с пометкой «Телефоны» для события «onClick» содержат следующий код: if RadioButton2.Checked then / DBGrid2.DataSource := fDM.DSTelephones;

Таким образом, в нижней сетке мы показываем ту или иную подчиненную таблицу, и каждый раз данные этого выпускника отображаются в этих таблицах. Нам остается только сохранить проект и приступить к его практическому использованию. Следовательно мы раскрыли на примере особенности системы управления базой данных MS Access из Delphi.

Процедуры, использованные в программе

Процедура сохранение данных:

```
procedure TEdit.Button2Click(Sender: TObject);
begin
messagedlg('Данные сохраненны',mtInformation,[mbOk],0);
main.ADOTable1.Post;
main.ADOTable1.Refresh;
button2.Enabled:=false;
end;
```

Процедуры удаления данных:

```
procedure TEdit.Button3Click(Sender: TObject);
begin
if application.MessageBox('Удалить
запись?','Сообщение',mb_yesno+mb_iconquestion)=idYES then
main.ADOTable1.Delete;
main.ADOTable1.Refresh;
end;
```

Процедура сохранения новых данных:

```
procedure TEdit.Button2Click(Sender: TObject);
begin
messagedlg('Данные сохраненны',mtInformation,[mbOk],0);
```

```
main.ADOTable1.Post;
main.ADOTable1.Refresh;
button2.Enabled:=false;
end;
```

Список литературы

1. Онлайн учебник по Delphi 7 / [Электронный ресурс] https://delphi.support.uz/
2. Марков, Е.П.; Никифоров, В.В. Delphi 2005 для .NET; [Электронный ресурс] / https://codernet.ru/books/delphi/delphi_2005_dlya_net_markov/

ӘОЖ 004.4

РОБОТОТЕХНИКА САБАҒЫНДА HUMANOID РОБОТЫН ЖАСАУ

Төлеубаев А.

Ө. Сұлтанғазин ат. Қостанай мемлекеттік педагогикалық университеті,
Қостанай қ.

Ғылыми жетекші: Айтбенова А.А.
Ө. Сұлтанғазин ат. Қостанай мемлекеттік педагогикалық университеті

Аннотация. Бұл мақалада робототехника, робот ұғымдары, оны қолдану салалары, Robotis Bioloid Premium жинағы қарастырылған. Сабақ уақытында жинаған Humanoid роботының үлгісі көрсетілген. Түйінді сөздер: Робототехника, робот, Humanoid.

Аннотация. В этой статье рассматриваются понятия робототехники, роботов, области их применения, коллекция Robotis Bioloid Premium. Приведен пример робота-гуманоида, собранного во время урока. Ключевые слова: Робототехника, робот, Humanoid.

Annotation. This article discusses the concepts of robotics, robots, their applications, the Robotis Bioloid Premium collection. An example of a humanoid robot assembled during a lesson is given. Keywords: Robotics, Robot, Humanoid.

Заманауи технологиялар әлемінде бізді робототехника көбейтіп жатыр. Робототехника - қазіргі әлемнің маңызды бөлігі. Күнделікті өмірде - мектепте, үйде біз көптеген техникалық құрылғыларды қолданамыз: ұялы телефондар, кір жуғыш машиналар, компьютерлік техника және басқалары - бұлардың бәрі роботтар. Жыл сайын ғылым дамиды, зерттеулер тоқтамайды. Бұл сала әлемде тез дамып келеді.

Қазіргі уақытта робототехника барлық салаларда және кәсіптерде қолданылады: өнеркәсіпте, медицинада, соғыста, ғарышта, роботтар бізге үйде көмектеседі, барлық