Мазмұны

Кіріспе

Курстық жұмыстың мақсаты: деректер базасын құрып, "Деректер қоймасын жобалау" тәсілін толық игергенімізді көрсету. Деректер қоймасын жобалау және ақпараттық жүйелерді құру кезінде Delphi деректер базасын басқару жүйесін қолданамын.

Delphi-де қосымшаларды құру процессі қарапайымдандырылған. Біріншіден ол программаны құрғанда уақыттың 80% кететін интерфейсі. Мұнда сіз жай ғана керекті компоненттеріңізді Windows-терезесіне(Delphi-де оны форма деп атайды) орналастырып, арнайы инструмент(Object Inspector) көмегімен олардың қасиеттерін баптайсыз және де бұл компоненттердің оқиғаларын байланыстыра аласыз (батырманы басу, тышқанмен тізімнен элементті таңдау ж.т.б).

Қосымша – бұл ақпаратты өңдеу процесін автоматтандаратын программа немесе программалар комплексі. Сонымен қатар құрастырушыға өңдеудің мықты құралдары беріледі, ыңғайлы анықтамалық жүйе(соның ішінде Microsoft API – ден де), жобаны құру барысында ұжымды жұмыс істеу құралдарын қолдану тағы да көптеген мүмкіншіліктері бар. ActiveX компоненттерін Microsoft IDL- ді қолданбай құруға болады, web-серверлердің мүмкіншіліктерін кеңейту ж.т.б.

Delphi-де деректер базаларымен жұмыс жасауы төменгі деңгейлі ядросына(деректер базасының Borland Database Engine (BDE) негізделген.

Әр бір ДББЖ функцияларына – деректерді анықтау, деректерді басқару және өңдеу жатады. ДББЖ келесі қарапайым төрт операцияны жасауға мүмкіндік беруі керек:

- Кестеге бір немесе бірнеше жазба енгізу;
- Кестеден бір немесе бірнеше жазбаны жою;
- өрістердің мәндерін жаңарту мүмкіндігі;
- берілген шарт бойынша бір немесе бірнеше жазбаларды табу.

Delphi осы операцияларды толығымен орындауға мүмкіндік береді.

1. Деректер базалары

1.1 Деректер базасын құру принциптері

Әрқашан үлкен көлемді деректерді басқару үшін деректер базалары құрылып, қолданылады. Деректер базалары қазіргі уақытта адамның қызмет көрсететін барлық салаларына кіретін ақпараттық процесстерін қамтамасыздандырудың негізін құрайды. Деректер базалары деректердің құрамын көрсететін және оларды басқаратын тиімді құрылғы болып табылады. Деректер базасының концепциясы ақпаратты сақтаудың интегралданған құрылғысын қолданады. Бұл кезде деректерді орталықтандырылған басқару және көптеген қолданушыларға қызмет етуді қарастырады.

Деректер базасы ЭЕМ ортасында - деректер базаларын басқару жүйесі(ДББЖ) деп аталатын бірыңғай бағдарламамен қамтамасыздандырылуы керек. Деректер банкі деп ДББЖ және орсалқы бағдарламаларды атайды.

Деректер құрылғының пәндік облысы - ол деректер базасы көмегімен құрылып немесе сипатталатын нақты уақыттың үзіндісі болып келеді. Пәндік салада деректер базасында сақталатын ақпараттық объектілер бөлінеді. Олар – идентификацияланатын объектілер, процесстер, жүйелер, ұғымдар және мәліметтер.

Деректер базасы(ДБ) объектілердің қатынасын және де қарастырылатын саласындағы аталған қатынастардың жиынтығы:

- деректерді базасын басқару жүйелері (ДББЖ) – деректер базасының құру, жүргізу, өңдеуін жеңілдету үшін көптеген қолданушыларға арналған тілдік және бағдарламалық құралы;
- деректер банкісі (БнД) –бағдарламалық, тілдік, ұйымдастырушылық және техникалық құралдар жүйесі ДБ технологиясына негізделіп деректерді жинақтап ұжым ортасында қолданылады;
- ақпараттық жүйе (АЖ) – автоматтандырылған жинақтау, деректерді өңдеу және басқару, және де деректерді өңдеудің техникалық құралдарын қоса алатын, бағдарламалық қамтамасыздандыруды жүзеге асыратын жүйе.

Толық ДББЖ өз құрамына циклдің барлық этаптарында (жобалау, құру, өндіру) әр түрлі деңгейлі қолданушылардың қажеттіліктерін қамтамасыздандыратын ДБ жүйесін орсады. Жобалау операциясы бағандарды таңдайды, таңдау операциясы - жолдарды, ал біріктіру операциясы біріктірілген кестелерден ақпаратты жинайды.

Деректер базасын жобалау – ақпараттық жүйесін құруына байланысты жобалаудың күрделі және жауапты есебі болып табылады. Оны шешу нәтижесінде ДБ-ның құрамы, болашақ қолданушылар үшін деректерді ұйымдастыру және деректерді басқару құралдарының тиімділігі анықталуы керек.

ДБ жобалау процесінің негізгі мақсаты болып келесі талаптарды қанағаттандыратын жоба алу:

1. ДБ сұлбасының нақтылығы, яғни база модельденетің пәндік саланың гомоморфті түрінде болуы керек. Мұнда пәндік саланың әрбір объектісіне

ЭЕМ жадысында сәйкес деректер, ал әр бір процеске – деректерді өңдеудің сәйкес адекватты процедуралары болады.

2. Шектеулердің қойылуы (тұрақты және жедел жадыларға және есептеу жүйесінің басқа да ресурстарына).
3. Қызмет нәтижелігі
4. Деректерді қорғау (аппаратты және программалық сбойлардан және рұқсатталмаған кірудің).
5. Қолданудың қарапайымдылығы және ыңғайлығы
6. Даму мүмкіндігі және пәндік саланың немесе қолданушылардың талаптарының ауысуына байланысты өзгеру мүмкіндіктері.
7. Жоба алыну үшін 1-4 талаптарды міндетті түрде қанағаттандыру тиіс.

1.2 Деректер базаларын құру тәсілдері

Қазіргі деректер базаларын құрудың негізгі үш тәсілі бар:

1. Иерархиялық
2. Желілік
3. Реляциялық

1.Деректердің құрама типі иерархиялық байланысты көрсетеді. Иерархиялық байланыс көмегімен жасалса, онда ол деректердің иерархиялық моделі деп аталады. Мәліметтердің иерархиялық модельдегі жазбаларының құрылымы тармақты құрылыммен жасалған. Иерархиялық модельдің артықшылықтарына: ЭЕМ жадысын тиімді пайдалануы және де негізгі операцияларды орындауының уақыт көрсеткіші жаман емес. Иерархиялық модель - иерархияны ретелген ақпараттан жұмыс жасауға ыңғайлы. Кемшілігі болып көлемі жағынан үлкендігі, логикалық байланыстары бойынша күрделігі, сонымен қатар жай қолданушыға түсіну қиындығы.

Иерархиялық модель негізінде бірнеше ДББЖ-рі құрылған. Олардың ішінде IMS, PC/Focus, Team-Up, Data Edge және де Ока, ИНЭК, МИРИС жүйелері

2.Жалпы жағдайда базада желіні сипаттайтын желілік байланыстар анықталуы мүмкін. Пәндік саланың жалпы түрдегі деректерін желілік құрылым ретінде көрсететін модельді – деректердің желілік моделі деп атайды. Желілік модельдің артықшылығына –иерархиялық модельге қарағанда еркін байланыс орнату мүмкіндіктері жатады. Кемшілігі болып – құрылымының күрделігі. Желілік модель негізінде құрылған жүйелер кең таралмаған. Олардың ішіндегі ең атақтыларына: IDMS,dk_VistaШ, Сеть, Сетор және Компас ДББЖ жатады.

3.Реляциялық модельдерде мәліметтердің элементтерімен байланыстарын бірдей көрсетуге болады.

1.3 Реляциялық деректер базаларының құрылу принциптері

Егер деректер базасын басқару реляциялық алгебраны қолданумен жүзеге асырылса, онда бұл деректер базасы- реляциялық деректер базасы деп аталады. Реляциялық модель мәліметтерді анықтауға, басқаруға және оларға әр түрлі операциялар жасауға мүмкіндік беретін арнайы тілді қолданылады(SQL). Реляция (Relation) – қатынас деген мағынаны білдіреді. Реляциялық мәліметтер кортында ақпарат кестелерде сақталады.

Доктор И.Ф. Кодд реляциялық модельдің авторы. Ол белгілі тізім жасаған және реляциялық модель осы тізімнің шарттарын қанағаттандыру керек. Бұл тізімді және «Коддтің 12 ережесі» деп атайды. Осы тізімнің реляциялық жүйелерге арналған кейбір ережелерін айтып кетуге болады:

- олар барлық ақпаратты кесте түрінде сақтау керек;
- деректердің логикалық құрылымын ұстану керек;
- құрылымды сұраныстарды орындау және деректер базасындағы ақпаратты өзгерту үшін жоғары деңгейлі тілді қолдану;
- негізгі реляциялық операцияларды(таңдау, жобалау, біріктіру) және де теоретикалы-жиынды операцияларды (біріктіру,қиылыстыру, қосу) сияқты операторларды қолдау керек;
- кестелерде белгісіз айнымалыларды айырып тану (nulls),нөлдік мәндер және деректердегі бос орындар;
- деректердің бүтіндігін, транзакцияның, қайта қалпына келтіруді қолдайтын механизмдерді қамтамасыздандыру.

Реляциялық деректер базасындағы барлық ақпарат кестедегі айнымалылармен беріледі. Реляциялық жүйеде кестелер жол мен бағандардан тұрады. Барлық мәліметтер кесте түрінде беріледі- деректер базасындағы ақпаратты қарайтын басқа жолы жоқ. Байланысқан кестелердің жиынтығы деректер базасын құрайды. Реляциялық базаларда кестелер бөлінген, бірақ құрылықтары тең. Деректердің әрбір элементі жол мен бағанның қиылысуымен анықталады. Деректердің керекті элементін табу үшін кестенің атауын, бағанын және алғашқы кілттің мәнін немесе идентификаторын білу керек.

Реляциялық деректер базалары – бұл бірінші ден кестелер жиынтығы, процедуралар және басқада объектілер. Кестеде атауы болады – идентификатор. Кесте бағандары объект қасиеттеріне сай келіп өрістер деп аталады. Әрбір өріс сақталатын деректерге байланысты атауы және типімен анықталады. Өріс атауы – идентификатор болып табылады және әр түрлі программаларда деректерді басқару үшін қолданылады. Ол кез келген идентификатор сияқты құрылады, яғни латын әріптерімен жазылады және бір сөзден тұрады. Өріс типі бұл - өрісте сақталатын деректердің типін анықтайды. Олар - жол, сан, булекті үлкен мәтіндер сияқты болуы мүмкін. Кестенің әр бір жолы – жазба деп аталады және бір объектке қатысты болады. Жазбада объектті сипаттайтын барлық өрістердің нәтижелерін көрсетуге болады.

Кесте құрғанда маңызды мәселе болып ақпараттың қарсыласпауы болып табылады. Бұндай жағдай болмау үшін кілттік өріс енгізіледі. Кілттік өріс - әр бір

жазбаның ерекшелігін көрсетеді. Бір немесе бірнеше өрістер кілттік бола алады. Кестедегі жазбалар реттіз немесе енгізілген реті бойынша бола алады, бірақ қолданушы өзіне ыңғайлы реті бойынша орналастыра алады. Мысалы: алфавит бойынша немесе туған жылы бойынша.

Деректерді реттеу үшін индекстер қолданылады, кестені қолданушыға ыңғайлы реті бойынша көрсетуін қамтамасыздандырады. Индекстер екі түрлі бола алады: бірінші ретті және екінші ретті болып бөлінеді.

Бірінші ретті индекс деп деректер базасы құрылғандағы өрістерді айтамыз. Екінші ретті индекстер басқа өрістер құрылғанда және деректер базаларын құрғанда немесе құрылған кезіндегі индекстерді айтамыз.Екінші ретті индекстерге ат, яғни идентификатор қойылады. Оларды идентификаторлар арқылы қолдану ыңғайлы болады. Егер индекс бірнеше өрістерден тұрса, деректер базаларын реттеу бірінші өріс бойынша, ал бірінші өрісте бірдей нәтижелері бар жазбалар үшін – екінші өріс бойынша. Мысалы: қызметкерлер деректер базасын отделдер бойынша, ал әрбір отдел ішіндегі деректерді – алфавит бойынша реттеуге болады.

Деректер базалары көптеген кестелерден тұрады. Мысалы: бір мекеменің деректер базасы әр бір орсалық бөлімдердің сипаттамалары бойынша бөлінген кестеден тұруы мүмкін. Бөлек, бөлек кестелер болған жақсы бірақ кестелер жиынтығынан көбірек ақпарат алуға болады. Мұндай жағдайда байланысқан кестелер жиынтығын қарастыру керек.

Байланысқан кестелерде біреуі басты кесте болып, ал екіншісі немесе басқалары – бағынышты болады және басты кестемен басқарылады. Басты және бағынышты кестелер кілт арқылы байланысады. Кілт ретінде екі кесте құрамына кіретін өрістер бола алады.

Деректер базаларын басқару жүйелері – деректер базаларын құрады және қойылған сұраныстарды өңдейді. Бір - бірімен бәсекелесетін, өз мүмкіндіктерімен ерекшеленетін көптеген ДББЖ бар. Олар: Paradox, dBase, Microsoft Access, FoxPro, Oracle, InterBase, Sybase және тағы басқалары.

Әр түрлі ДББЖ деректерді әр түрлі дәрілайды және әр түрлі сақтайды. Мысалы: Paradox пен dbase әрбір кестеге бөлек файлды қолданады. Бұл жағдайда деректер базасы кесте файлдары сақталатын каталог ретінде беріледі. Microsoft Access пен InterBase-те бірнеше кесте бір файл ретінде сақталады.

Клиент/сервер типті жүйелер, мысалы Sybase немесе Microsoft SQL барлық деректерді бөлек компьютерде сақтайды және клиентпен арнайы SQL деп аталатын тіл көмегімен байланысады.

Псевдоним (alias) деректер базасына кіру үшін қажетті барлық ақпаратты сақтайды. Бұл ақпарат тек қана псевдоним құрылуында бір рет беріледі.

Деректер базасымен жұмыс жасау барысында барлық өзгерістерді кэштеуі қолданылады. Жасалатын барлық өзгерістер мысалы: жаңа жазба қосу, бар жазбаларды өшіру, яғни қолданушы енгізетін деректерді манипуляциялауы деректер базаларында жасалмай уақытша кэштелінген кестеде шығарылады. Тек қана тексерістен кейін қолданушыға өзгерістерді енгізуге немесе болған қалпына келтіруіне рұқсат етіледі.

Деректер базаларында өзгерістерді фиксациялау транзакция көмегімен жасалады. Транзакция деп мәліметтер қорын біртұтастығын жоғалтпай бір күйден екінші бір күйге ауыстыратын әрекетті айтамыз. Бұл деректер базаларын өзгертетін командалар жиынтығы. Оның нәтижесінде оған қатысты кестелердегі деректер өзгеріске ұшырайды. Егер транзакция кезінде мәліметтер қорына енгізілген бір өзгеріс нәтижесіз қалса, деректер қоры басталған қалпына қайтарылуы тиіс. Шындайында барлық өзгерістер жалыда сақталады және де қолданушыға транзакцияны аяқтау немесе барлық өзгерістерді реалды базаға енгізу, немесе өзгерістерден бас тартып болған күйіне оралу мүмкіндіктері беріледі. Транзакция түсінігі деректер қорының логикалық бүтіндігін қолдау үшін қажет. Транзакция бұл- ДББЖ-гі біртұтас қаралатын операциялардың жиынтығы немесе жүйесі. Транзакция басталуының алдында деректер базасының бүтіндік күйінен басталады және аяқталған соң бүтіндік күйін сақтайды. Транзакцияның бұл қасиеті оның деректер базасына қатысты қолданушының активетер бірлігі ретінде қолдануға мүмкіндік береді. Әрбір транзакцияның басы және соңы болады.Транзакцияны сақтауға немесе өшіруге болады. Егер транзакцияның орындалуы барысында операцияның бірі орындалмаса мәліметтер базасында транзакцияның бірде бір құрауышысы сақталмауы керек.

1.4 Реляциялық ДББЖ кестелерді байланыстыру түрлері

Деректер базасын жобалау кезінде ақпаратты әдетте бірнеше кестелерге бөледі. Реляциялық ДББЖ-парда кесте аралық байланыстарын көрсету үшін оларды байланыстыру операциясын өткізеді. Кестелер байланыстырылған болса, деректермен жұмыс жасау онайырақ болады.
Кесте байланыстарының негізгі түрлері:

1. бинарлы (екі кесте арасында)
2. тернарлы (үш кесте арасында)
3. n – арлы (жалпы жағдайда)

Екі кесте байланысқан болса, олларда негізгі және қосымша (бағыныштыы) кестелерді бөледі. Кестені логикалық байланыстыру кілт арқылы жасалады. ДБ-ның бүтіндігін сақтау үшін кестенің бір немесе бірнеше өрісіне кілт қойылады.

Байланыстырудың мақсаты болып – негізгі және бағыныштыы кестелер арасында байланыс орнату.

Екі кесте арасында байланыстың келесі төрт түрін бөледі:

- бірге – бір (1:1);
- бірге – көп (1:M);
- көпке – бір (M:1);
- көпке – көп (M:M немесе M:N).

Бірге – бір байланысында негізгі және қосымша кестенің барлық байланыс өрістері кілттік болып табылады. Екі кілттік өрісте де мәндер қайталанбайтындықтан өзара бірмәнді жазбалар болады. Мұндай байланыстағы кестелер тең құқығы болып табылады.

Бірге – көп байланысында бірінші кестенің бір жазбасына басқа кестенің бірнеше жазбасы сәйкес келеді.

Көпке – бір байланысқа негізгі кестенің бір немесе бірнеше өрістері қосымша кестенің бір өрісімен сәйкес келетін байланыс.

Көпке – көп байланысы бірінші кестенің бірнеше жазбалары қосымша кестенің бірнеше жазбасына сәйкес келетін болса, ұйымдастырылады. Практикада байланыстардың ішінен бірге – көп байланысы қолданылады.

1.4.1 Деректер базаларының түрлері

Әр түрлі есептемелер үшін деректер базаларының әр түрлі модельдерін қолдану ыңғайлы, яғни шағын мекеменің деректер базасы мен үлкен банктің деректер базасын әр түрлі құру керек.

Масштабтау деп нақты қосымшаға қай процесс ыңғайлы екенін анықтайтын процесті айтады. Мүмкін болатын деректер базаларының моделдерін қарастырайық. Өйткені бұл Delphi-де деректер базамызды құруға әсер етеді.

Деректер базаларының төрт моделін қарастырайық:

- Автономды
- Файл- серверлі
- Клиент/сервер
- Көпдеңгейлі бөлінген деректер базалары

Автономды деректер базалары - деректер базаларының ең қарапайым түрі болып табылады. Олар өздерінің деректерін локальды, файлды жүйеде басқару жүйесі мен деректер базасының машинасы орналасқан компьютерде сақтайды. Бұл жағдайда желі қолданылмайды. Сондықтан автономды деректер базаларын құрушы рұқсаттың параллельдік проблемасын есептемейді. Автономды деректер базасы көптеген қолданушылар қолданылатын қосымшаларды дамыту үшін пайдалы. Бұл жағдайда әрқайсысының бөлек деректер базасы болады. Мысалы: үлкен емес офис құжаттары, шағын көлемді кәсіпорынның кадрлік құрамын, кішкентай есеп бөлімдерінің есепшілік құжаттарын өңдеу қосымшалары.

Бұндай қосымшаның әрбір қолданушысы өз компьютерінде өзінің деректерін басқара алады. Қолданушыға басқа бір қолданушының деректеріне рұқсат керек емес, бөлек базалар мында өте ыңғайлы.

Файл – серверлі деректер базаларының, автономды деректер базаларынан айырмашылығы – желі арқылы көптеген клиенттерге қол жетімділігі болып табылады. Бұл өте ыңғайлы себебі өзгерістерді барлық қолданушылар көре алады. Деректер базасының өзі файл – серверде сақталады.Жұмыс кезінде әр бір клиентке локальді көшірмесі жасалады.

Файл – сервер деректер базасының кемшілігі болып желіні тиімсіз жүктеуі болып табылады. Клиенттің әр бір сұранысы бойынша локальды көшірмесіндегі ақпарат сервердегі деректер базасынан жаңартылып тұрады. Сұраныс бір жазбаға қатысты болса да барлық жазбалар автоматты түрде жаңартылады. Тағы бір кемшілігі болып деректердің бүтіндігі болып табылады, яғни қолданушылардың программалары өте жақсы ойластырылған болуы керек, себебі базаға қате енгізу оңайосылған байланысты бұл қате барлық қолданушыларға әсер етуі мүмкін.

Клиент/сервер деректер базасы. Үлкен көлемді деректер базалары үшін клиент/сервер платформасындағы деректер базасы жиі қолданылады.

Клиент/серверге белгілі бір операцияларды орындауға бұйрық бере алады. Қуатты сервер орындаған соң клиентке өз жұмысының нәтижелерін береді. Жұмыстың бұлай ұйымдастырылуы сервер көмегімен қосымшаларды тиімдірек орындауын көбейтеді, желіні еңгізеді, деректердің бүтіндігінің басқаруын жақсартады.

Клиент/сервер деректер базаларында қосымша проблемма болуы мүмкін – сервердің мүмкіндіктерін максималды, ал желіні минималды қолдануын жобалау. Бұл серверде сақталатын процедураларды жасау арқылы жүзеге асады. Олар бизнес- логиканы ұйымдастырады: деректерді өңдейді және қолданушыға, тек сұралған ақпаратты жібереді.

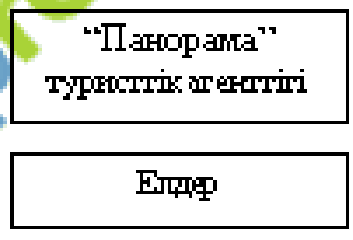
Көп деңгейлі бөлінген деректер базалары. Бұл бөлінген қосымшалар арқылы желідегі деректерді өңдеу түрі Ең кең тараған үш деңгейлі нұсқасы:

- Төменгі деңгейде қолданушының компьютерде клиенттердің қосымшалары орналасқан.
- Екінші деңгейде қосымшалардың сервері орналасқан. Мұнда қолданушымен бөлшектелген деректер базалары арасында мәліметтер алмастырылады. Қосымшалар сервері барлық клиенттерге рұқсат етілген желінің түйінінде орналасқан.
- Үшінші деңгейінде жоғылған деректер базалары орналасқан. Ол қосымшалар серверінен ақпаратты қабылдайды және оларды басқару функцияларын орындайды.

Бұл деректер базаларын құрудың ең күрделі түрі болып табылады. Delphi бұл жүйенің алғашқы екі деңгейлеріне ғана қосымшаларды құра алады. Және де төменгі деңгейінде қолданушы компьютерінде Borland Database Engine(BDE) қондыру міндетті емес. Көпдеңгейлі бөлшектеген деректер базаларының артықшылықтарының бірі болып осы ерекшелігін айтуға болады.

1.4.2 ER–диаграмманы деректер базасының сұлбасына түрлендіру

Деректер базасы деректер базасының сұлбасы негізіне құрылады. ER–диаграмманы деректер базасының сұлбасына түрлендіру үшін толық ER–диаграммасын келтірейік. ER–диаграмманы деректер базасының сұлбасына түрлендіру әрбір мәнді және атрибуттары бар әрбір байланысты, қатынастарды (ДБ кестелері) көрсету жолымен жасалады. Келесі сұлбада көрсетілген белгілерді қолданамыз:

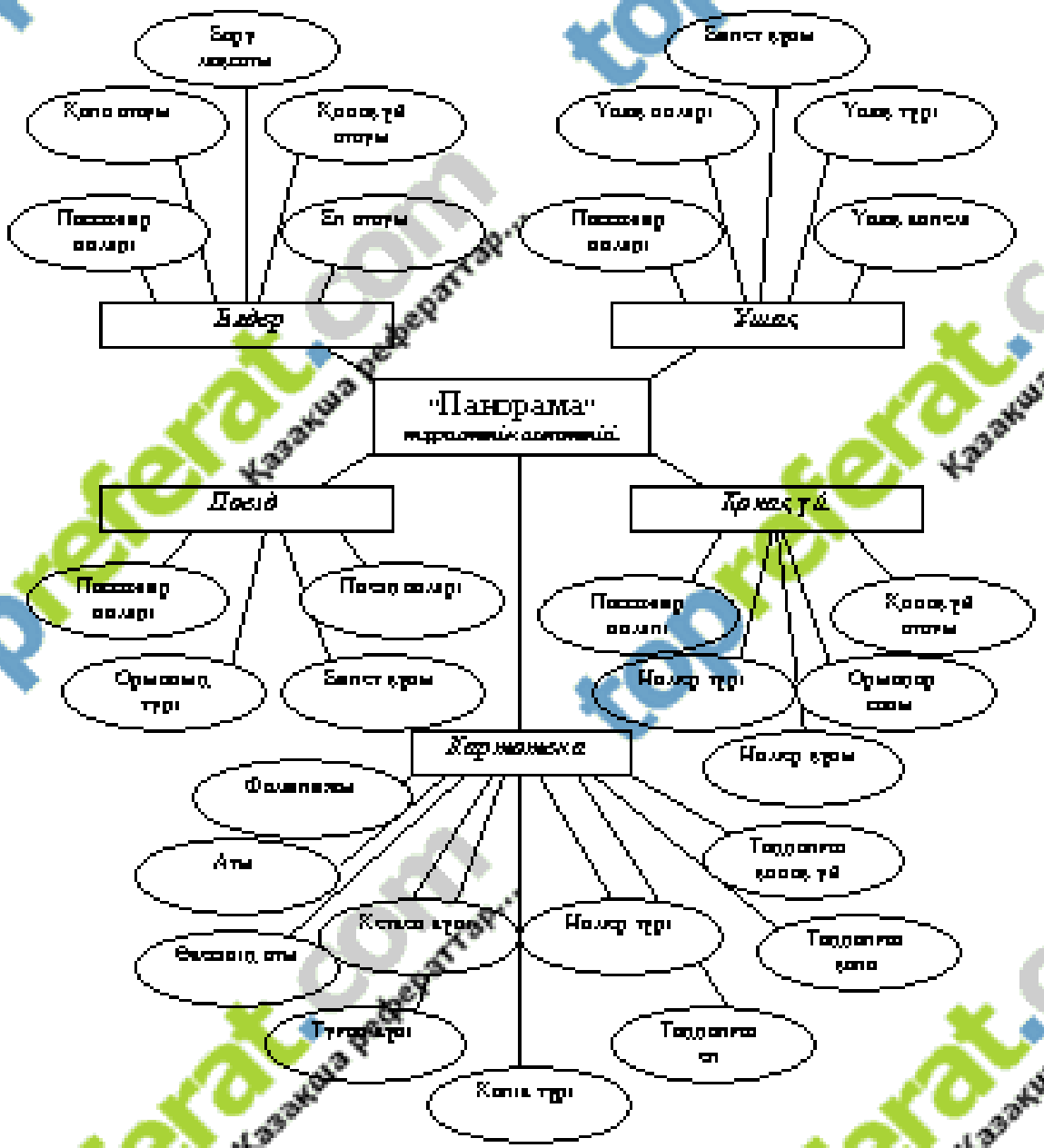


Бағынышты қатынас (бұл өріс бойынша байланыс жасалады)

Атрибут

Сұлба 1. Белгілер енгізу

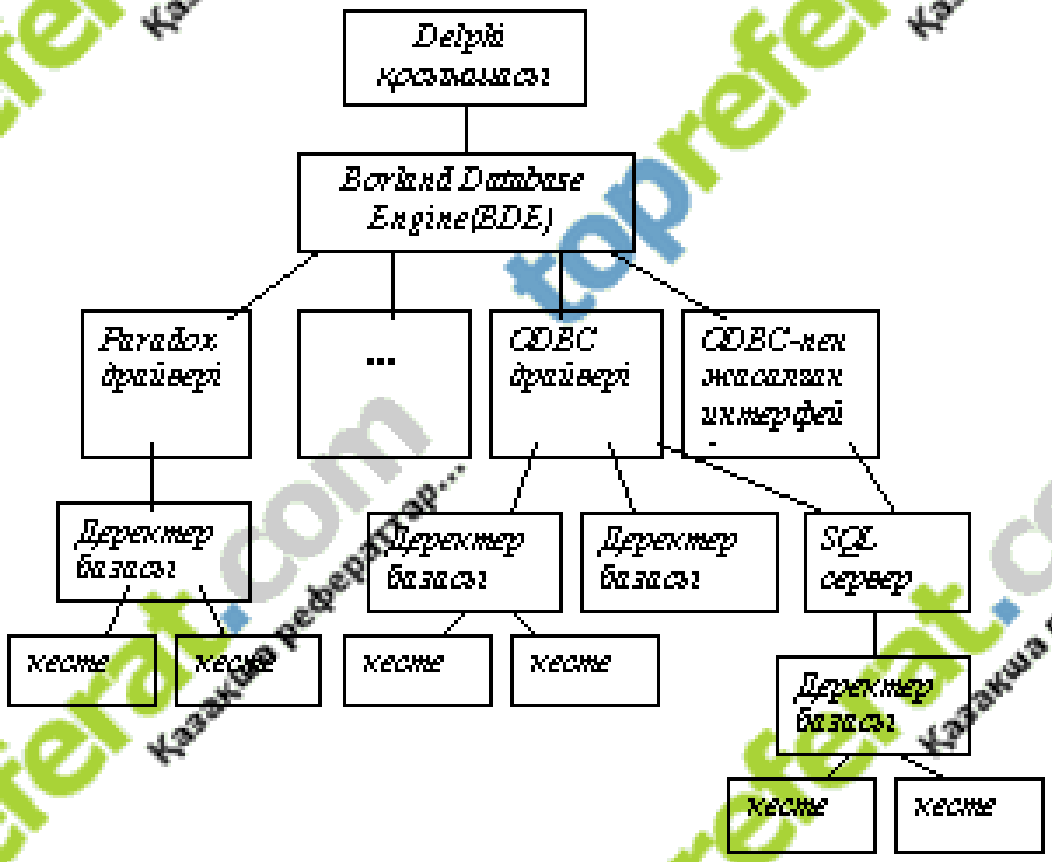
«Бірге – бір байланысы»
«Бірге – көп байланысы»



Сұлба 2. "Панорама" туристтік агенттігінің ER–диаграммасы

Деректер базаларымен Delphi-де байланыс орнату

Delphi-дің бірінші версияларында деректер базаларымен жұмысты жүзеге асыратын негізі болып Borland Database Engine(BDE) – Borland фирмасының деректер базаларының процессоры болды. Өз мағынасын ол қазірде жоғалтқан жоқ. Delphi қосымшасы деректер базасына BDE арқылы қосылады. Бұл жағдайда деректер базасымен байланысы келесі сұлба бойынша жасалады:



Сұлба 3. Delphi қосымшасының деректер базаларымен байланыстыру сұлбасы

Delphi қосымшасына деректер базасымен байланыс керек болған жағдайда әдетте BDE-ге жолығып, деректер базасының псевдонимін және ондағы қажетті кестенің атауын көрсетеді.

BDE-нің динамикалық қосылатын библиотекалары - DLL түрінде құрылған (IDAPI01.DLL, IDAPI32.DLL- файлдар). Олар әрбір библиотека сияқты API-мен қамтамасыздандырылған. API – Application Program Interfase – қолданбалы программалардың интерфейсі. IDAPI (Integrated Database Application Program Interfase) – бұл деректер базаларымен жұмыс жасауға керекті процедуралар мен функциялардың тізімі. BDE псевдоним бойынша деректер базасы үшін қажетті драйверді таба алады. Драйвер – бұл қосымша бағдарлама. Егер BDE-да ДББЖ сәйкес келетін өзінің драйвері болса, онда BDE ол арқылы деректер базаларын және керекті кестелерге байланысады, қолданушының сұранысын өңдейді және өңделген нәтижені қайтарады. BDE-нің: Microsoft Access, FoxPro, Paradox, dBase сияқты базаларға доступы бар. Егер керекті ДББЖ-ға қолдануға болатын BDE-нің өзінің драйвері болмаса, онда ODBC драйвері қолданылады. ODBC (Open Database Connectivity)- DLL, функциялары бойынша BDE-ге ұқсас бірақ Microsoft фирмасымен құрылған. Ол ODBC.DLL файлында ақталады. ODBC үшін әр бір ДББЖ-ға сәйкес келе алатын драйверлер жасалған. Borland фирмасы BDE –ға

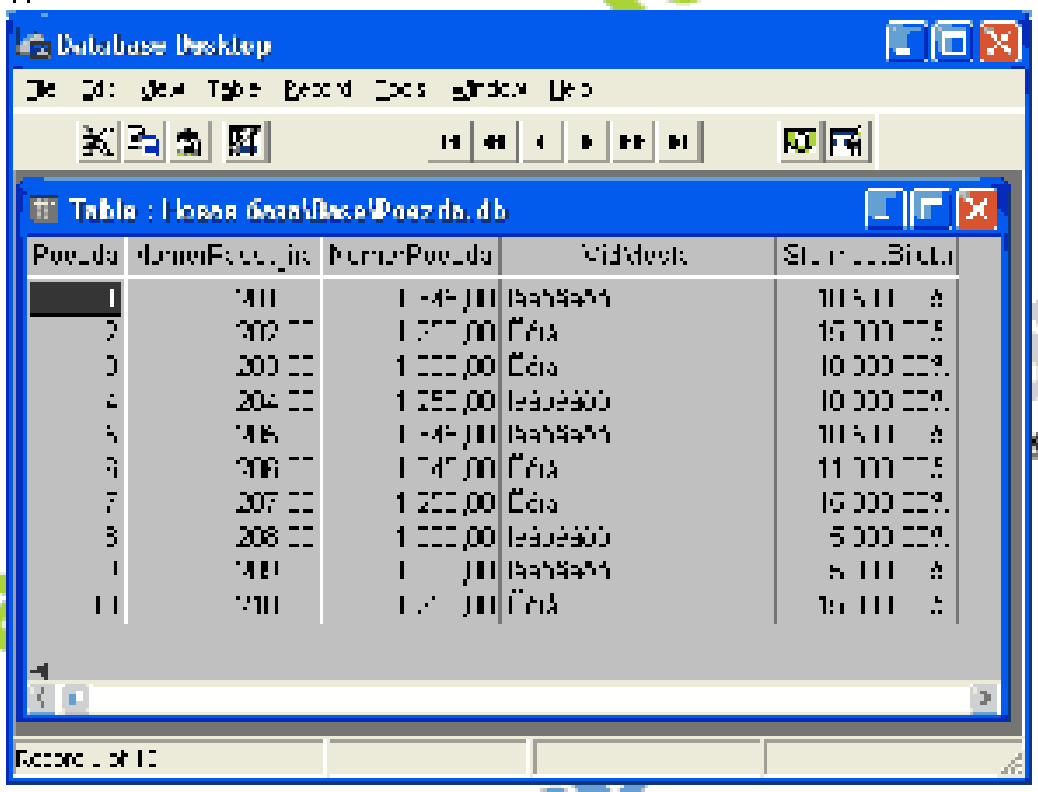
ODBC-ті қолдану үшін драйвер қосты. Бірақ ODBC арқылы жұмыс ДББЖ-ң BDE-ге ірсыпған өздерінің драйверлеріне қарағанда жайірек жасайды.
BDE-де SQL қолданылады.SQL – құрылымдалған сұраныстар тілі. Sybase, Microsoft SQL, Oracle, Interbase сияқты SQL – серверлерімен ақпаратты алмастыруға мүмкіндік береді. Бұл мүмкіндік клиент/сервер платформасында қолданылады.

Delphi 5-те BDE-ні қолданбай деректер базалармен жұмыс жасаудың басқа альтернативті мүмкіндігі болған. Бұл Microsoft-пен құрылған ActiveX Data Objects(ADO) технологиясы. ADO – бұл кез келген деректер типіне арналған қолданбалы интерфейс. Ол реляциялық және реляциялық емес, электронды почта, жүйелік, мәтіндік және графикалық файлдарды қосады. Мәліметтермен байланыс OLE DB технологиясы арқылы құрылады. ADO-ні қолданып деректермен жұмыс жасау тиімдірек болады. Және де интернетпен байланысты кейбір проблеммалар шешіледі. ADO-ні қолдану үшін сіздің компьютеріңізде ADO 2.1 системасы немесе оданда ерте версиясы, және де Microsoft SQL Server клиенттік системасы орнатылуы керек, ал ODBC-та OLE DB драйвері болуы керек. Delphi 5-те деректер базаларына доступтың тағы да бір альтернативті түрін қарастырған. Бұл Interbase от Interbase Express(IBX) технологиясының негізінде енгізілген. Delphi 5 компоненттерінің ішінде Interbase компоненті қосылды. Бұның ішіндегі компоненттері BDE-сіз Interbase-пен жұмыс жасауға қарастырылған. Бұл компонент енімділігі жағынан BDE-ге қарағанда тиімдірек болады. Delphi 6 және 7-де dbExpress технологиясы қосылды. Бұл SQL серверге доступті қамтамасыздандыратын драйверлер жиынтығы. Бұл курстық жұмыстың мақсаты BDE мүмкіндіктерін толығымен қарастыру.

2. ДБ қосымшасын жобалау этаптары

2.1 Database Desktop-ті қолданып деректер базасын құру. Database Desktop көмегімен жаңа кесте құру

Әдетте Database Desktop Delphi-дің басты мәзірінде Tools бөліміне қосылған болады. Егер бұл жасалынбаса оны Tools/Configure Tools командасы арқылы қосу керек. Database Desktop-ті шақырыңыз, үлгісін келесі кестеден көруге болады:



Сурет 2.1.1 Database Desktop бағдарламасының басты терезесі

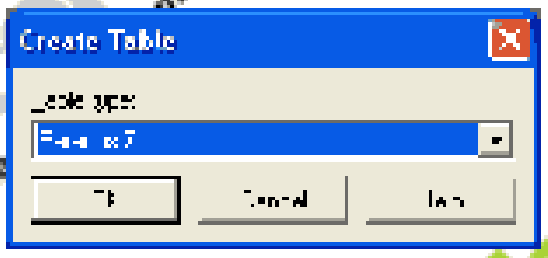
Database Desktop көмегімен ДББЖ Paradox7 кестесін құрдым. Paradox7- де деректер базалары – бұл кестелер орналасқан каталог. Мұндағы файлдардың кеңейтілуі - *.db

Енді Database Desktop File/new командасын орындадым. Мұнда QBE Query, SQL File, Table деген бөлімдерден тұратын мәзір ашылады.

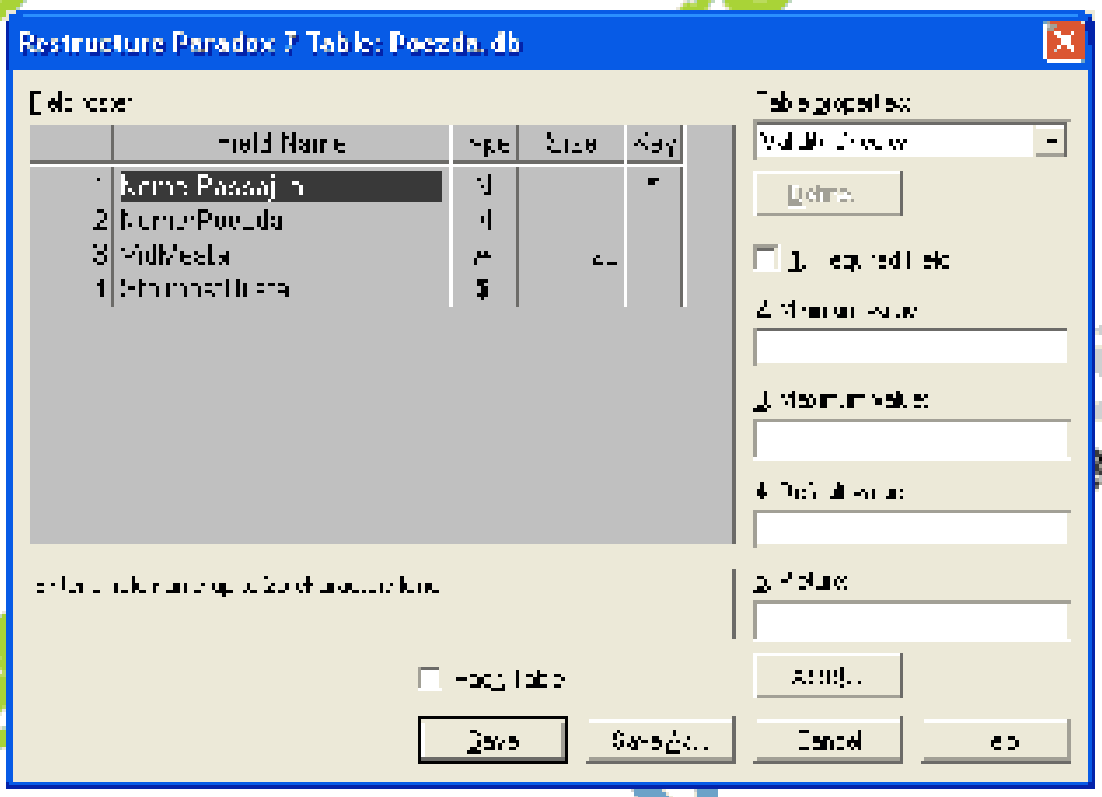
QBE Query – сұраныстар құрудың визуальды компоненті және сұраныстарды файлға жазады.

SQL File – сұранысты SQL-де құру және файлға жазу.

Table – жаңа кесте құру. Table-ді таңдадым. Ашылған диалогтік терезеден керекті ДББЖ-ні таңдауға болады. Paradox7-ні таңдағанда, кестенің құрылымы құрылатын терезе ашылады.



Сурет 2.1.2 Деректер базасын басқару жүйесін таңдау терезесі



Сурет 2.1.3 Paradox кестесінің құрылымын құру терезесі

Кесте келесі өрістерден тұрады:
NomerPassajira – «Number» типті сандық жол. Пассажир нөмірін енгізу үшін арналған.
NomerPoezda – «Number» типті сандық жол. Поезд нөмірін енгізу үшін арналған.
VidMesta - «Alpha» типті жолдық өріс (ұзындығы 20 символ). Орынның түрін енгізуге арналған. Мысалы, плацкарт немесе купе.
StoimostBileta – «Money» типті ақшалық жол. Билеттің құнын енгізу үшін арналған.

2.2 Жазбаларды орнату

Құрылатын кестенің әр бір жазбасына ат, яғни идентификатор қойылады. Ол 25-ке дейін символдардан тұра алады және пробел қойылынбайды. Содан соң жазбаның типі(type) таңдалады. Бұл үшін type бөлімінде етіп тышқанның оң жақ

Батырмасын бастым. Берілген жиптердің ішінен керектісін таңдадым. Paradox-та қолданылатын типтердің түсіндіктемелер келесі кестеден көруге болады.

Paradox кестелері өрістерінің негізгі типтері:

Өріс типі	Database Desktop белгіленуі	Сипаттамасы
Autoincrement	+	
Alpha	A	Символдық жол. Ұзындығы 255 символмен шектелген
BSD	#	Сандарды екілік – оңдық бейнелеу формасы.
Binary	B	Байттар тізбегі түріндегі екілік сандар. Ұзындығы шектелмеген.Алғашқы 240 байт кесте файлында сақталады, ал қалғандардың кеңейтілуі *.mb файлда сақталады.
Bytes	Y	255-ке дейінгі байттар тізбегі.
Date	D	Дата мәні. Мәні 01.01.9999 ден 31.12.9999 дейін.
Formated Memo	F	Форматталған символдардың шектелмеген тізбегі.
Graphic	G	*.bmp,*.eps, *.gif, *.pcx, *.tif форматтарының бірін қолданатын графикалық бейнелер. Кесте өрісіне сурет жүктелгеннен кейін *.bmp форматына түрленеді. Мәні *.mb кеңейтілуі бар файлда сақталады.
Logical	L	Булевтік(логикалық) айнымалы. True(ақиқат) немесе false(жалған) мәндерінің бірін қабылдайды.
Long Integer	I	Бүтінсандар типі. -2147483648 ден 2147483647 аралығындағы диапазон.
Memo	M	Шектеусіз символдар тізбегі Алғашқы 240 байт кесте файлында сақталады, ал қалғандардың кеңейтілуі *.mb файлда сақталады.
Money	$	Ақшалық шамаларды сақтайды. Number типінен айырмашылығы ақшалық таңбасында. Таңба символы операциялық жүйелер

		баптауларынан тәуелді.
Number	N	Қалқымалы үтірлі сан -10^{307} дейін 10^{308} диапозоны аралығында мәндерді қабылдайды. Берілу дәлдігі ондық нүктеден кейін 15 таңба.
OLE	O	OLE(object linking Enbeding) технологиясын қолдайтын деректерді сақтайды. Мәні * mb кеңейтінді бар файлды сақтайды.
Short	S	Бүтін сан – 32768-ден 32767 аралығындағы мәндерді қабылдайды.
Time	T	Уақыттық шамадан тұрады.
Time Stamp	@	Даталық және уақыттық шамадан табады.

Кесте 1

Кейбір типтер үшін өлшемін(size) көрсету керек. Мысалы: Alpha жолдық тип үшін өлшем - бұл символдар саны. Memo, Graphic, Binary т.б. типтер үшін өлшемін көрсету керек емес.

Кілттік өрістер «*» символымен көрсетілуі керек. Бұл символды жою немесе жою үшін соңғы бағанның сәйкес жолында екі рет шерту керек немесе белгілеп бос орын пернесін басу керек.

Егер бірнеше кілттік өрістер болатын болса онда Paradox кестелерінде олар бірінші болуы керек.

2.3 Кестенің қасиеттерін көрсету

Терезенің оң жағында кесте қасиеттері көрсетіледі(Table properties). Жоғарыында бірнеше бөлімдерден тұратын тізім берілген. Validity Checks – нетиже дұрыстығын тексеру Бұл бөлімді таңдаған кезде жазбаның келесі қасиеттерін көрсетуге болады:

Required Field – бұл индикаторда әр бір жазба да міндетті түрде болатын жазбалар бекітіледі.

Minimum – ең кіші мағынаның ұзындығын білдіреді. Бұл қасиетті сандық шамалар үшін қолдануға болады.

Maximum - ең үлкен мағынаның ұзындығын білдіреді. Бұл қасиетті сандық шамалар үшін қолдану керек.

Default – үнсіздік режиміндегі мән. Бұл қасиетті сандық, логикалық және де кейбір символды типтер үшін орнату керек.

Picture – деректерді енгізу шаблоны. Мысалы: телефон нөмірінің шаблонын құруға болады(## - ## - ##).

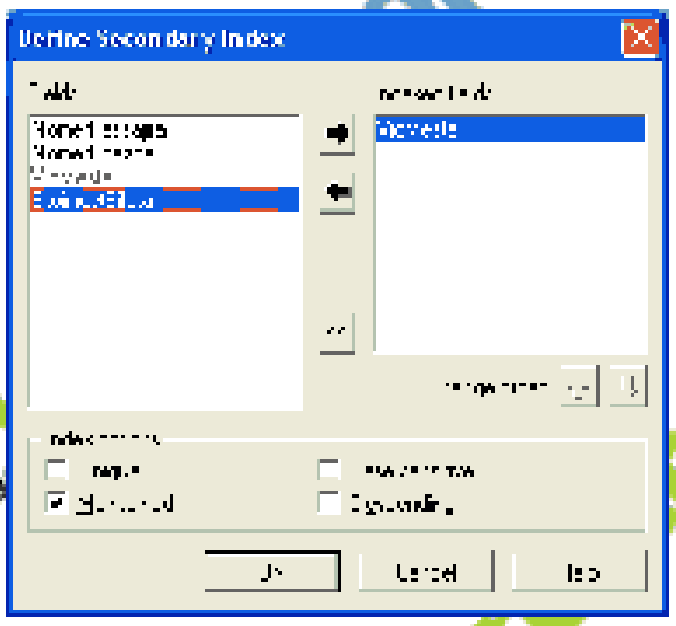
Assist – бұл батырма Picture шаблонын құруға көмектесетін диалогтік терезені шақыр2.4 Secondary Indexes – екінші ретті индекстер

Кесте қасиеттерінің тізімінің келесісі Secondary Indexes– екінші ретті индекстер. Бұл бөлімде жұмыс үшін қажетті – екінші ретті индекстер алғашқы индекс қатты өрістермен жасалады.

Екінші ретті индекс құру үшін Define- анықтау, батырмасын басамыз 1.8.4.1 суреттегі диалогтік терезе ашылады.

Терезенің сол жақ Fields өрістер тізімі көрсетіледі, ал оң жақ Idexed fields белгісінде индекстелетің өрістерді реттей аласыз. Сол жақтағы өрісті терезенің оң жақ бөлігіне ауыстыру үшін, қажетті өріс немесе бірнеше өрістерді белгілеп, оң жаққа көрсетілген тұрған батырманы басу керек.

Change order батырмасы көмегімен индекстегі өрістердің ретін өзгертуге болады.



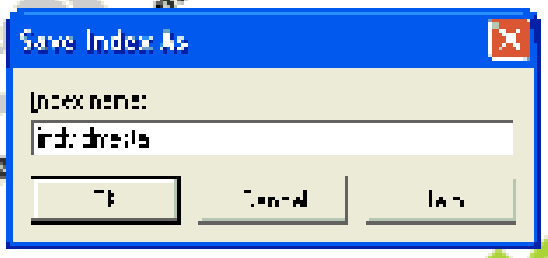
Екінші ретті индексті орнату терезесі Сурет 2.4.1

Index Options (индекстің операциялары) келесі қасиеттерді орнатуға болады:

Unigue	Кестені индекстеуге мүмкіндік береді
Desending	Бұл қасиетін орнатқан кезде кестедегі мәндер кему реті бойынша сұрыпталады (үндіксіз режімінде реттеу өсу реті бойынша жасалады).
Case Sensitive	Бұл қасиеті көмегімен символдар орнатылған регистр іске қосылады.
Maintained	Бұл қасиет орнатылып тұрса, онда индекс кестедегі әр бір өзгерісайын жаңартылып тұрады.

Кесте2

Индекс құрылғаннан кейін Ок батырмасын басамыз, ашылған терезеде индекстің атын көрсетеміз.



Индексті атау терезесі Сурет 2.4.2

Referential Integrity – сілтемелер деңгейіндегі бүтіндік.

Бөлек-бөлек кестелер арасындағы байланысты қамтамасыздандыру үшін қолданылады. Егер сілтемелер деңгейіндегі бүтіндік екі кесте арасында орнатылатын болса, онда біреуі басты, ал екіншісі қосымша кесте деп бөлінеді. Қосымша кестеде өз мәндерін басты кестенің кілттік өрістерінен ала-алатын өріс (немесе кілттік өрістер) көрсетіледі, және де басты кестеде өзгерістер болса қосымша кестенің мәндері автоматты түрде өзгертілетін операцияларды орнатуға болады. Яғни сілтемелер деңгейіндегі бүтіндік кестелерінің барлық типтерінде қарастырылмайды, бірақ Paradox'та бұл мүмкіндіктер бар. Referential Integrity құрудың алдында байланыстайдырылатын екі кестені құрып қою керек(Басты және бағыныштты). Мұндай байланыс орнатылмастан бұрын File/Working Derectory командасын орындамастан бұрын кестелерді бір каталогқа орналастыру керек. Содан соң қосымша (бағыныштты) кестені ашып (File/open командасы), оның реструктуризация (Table/Restructure командасы) режиміне кіріп және Table properties терезесінде Referential Integrity-ді таңдаймыз. Define батырмасын шерткенде диалогтік терезеде ашылады. Сол жақ Fields панелінде басты кестенің бір өрісін (немесе бірнеше өрістерін) Childfields тізіміне ауыстыра алсыз. Содан соң Table оң жақ панелінде басты панелді көрсетіп және Parent's Key кестесінің кілттік өрісіне ауыстыру керек.

Егер басты кестедегі қосымша кестемен байланысты бір өрістерін жойып жіберсеніз оны Update радиобатырмалары анықтайды. Егер бұндағы Prohibit опциясын орнатсаныз, онда Database Desktop мұндай қимылға жол бермейді, ал егер Cascade-ті орнатсаныз басты кестенің кілттік өрістеріндегі өрістер автоматты түрде қосымша кестеге өзгерістер әкеледі. Диалогтік терезенің төменгі жағындағы Strict Referential Integrity индикаторын орнатсаныз Paradox-қ алғашқы версияларында (DOS-қа арналған Paradox) сілтемелер деңгейіндегі бүтіндік бар кестелерін ашып және бұлдіріп тастауға мүмкіндік бермейді.

Қажетті операцияларының барлығын орындаған соң Ok батырмасын басып және ашылған диалогтік терезеде сілтеменің атын енгіземіз.

Password Security – рұқсат парольдері

Кестенің қасиеттерінің тізіміндегі келесісі - ол Passoword Security. Paradox әр бір кесте үшін рұқсат сөзін бере алады. Define батырмасын шерткен кезде терезе ашылады. Мұнда сіз негізгі пароль енгізе аласыз(Master Passoword терезесі), дұрыстығын дәлелдеп (Verify Master Passoword терезесі), содан соң Auxiliary Passoword (қосымша парольдер) батырмасын шерту арқылы жаңа

диалогтік терезе ашамыз. Қосымша парольдер енгізіп олардың рұқсат ережелерін анықтаймыз.
Current Passoword терезесінде ағымды парольді енгізуге болады. Ол терезеге кірген парольмен сәйкес келуі керек.

Table Rights (кестеге рұқсат ережелері) радиобатырмалар төменде кестедегі рұқсат деңгейін анықтауға болады:

All	Кестенің құрылымын өзгерту, өшіру, парольдерді өзгерту және жою сияқты операцияларға рұқсат беріледі
Insert&Delete	Жазбалармен кез-келген операцияларға рұқсат беріседе, бірақ кестенің құрылымын өзгертуге немесе кестені өшіруге рұқсат берілмейді
Data Entry	Мәліметтерді өзгертуге және жазбаларды қосуға мүмкіндік беріледі, бірақ жазбаларды өшіруге және кестенің құрылымын өзгертуге немесе жоюға тыым салынған
Updata	Кестені көруге және кілттік емес өрістерін өзгертуге рұқсат беріледі
Read Only	Кестені тек көруге мүмкіндік беріледі

Кесте 3

Field Rights терезесінде(өріске рұқсат ережелері) әр бір өріске қосымша рұқсат ережелерін бере аламыз, бірақ ол кестеге берілген ережелерінен аспауы керек:

all	өріске берілген барлық рұқсаттырды береді
Read Only	бұл өрістің мәліметтерін оқуға ғана мүмкіндік береді
none	бұл өрісті не көруге, не өзгертуге мүмкіндік бермейді

Кесте 4

Осы қосымша парольдің барлық рұқсаттарын қойғаннан кейін Add батырмасын басқан кезде пароль Password парольдер тізіміне енгізіледі. Содан соң New батырмасы арқылы жаңа қосымша парольді енгізуін бастай беруге болады. Change батырмасымен енгізіліп тұрған қосымша парольді өзгертіп немесе Change батырмасынан кейін Delete басып жойып жіберуге болады.

Table language – кестенің тілін таңдау

Table properties тізімінің бұл бөлімі кестенің тілін енгізуге немесе Modifi батырмасы арқылы тілін анықтап қоюға болады. Әдетте BDE Administrator бағдарламасы көмегімен ДББЖ-нің драйверінде үнсіздік режимінде қойылады. Таңдалған тілдің дұрыстығы кестедегі орыс мәтіндер оқылмадыма екенің анықтайды.

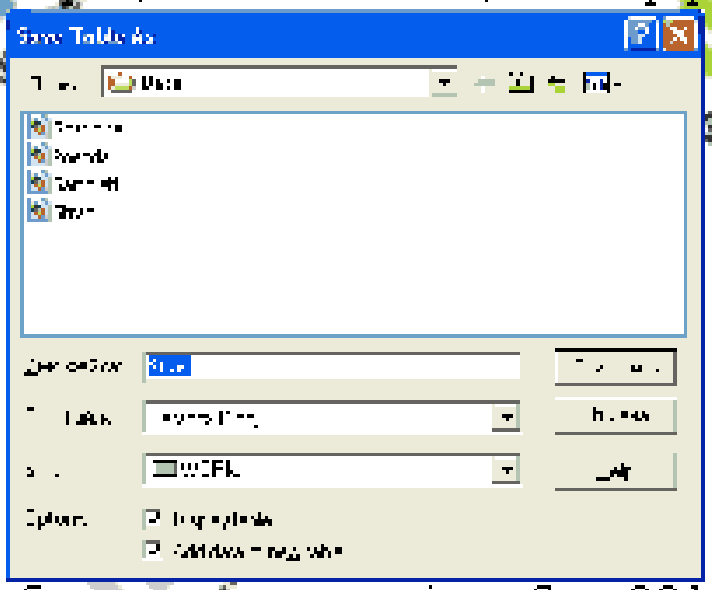
Dependent tables – тәуелді кестелер

Table properties тізімінде бұл соңғы бөлім. Мұнда - бұл бүтіндікпен Referential Integrity сілтемелер деңгейінде байланысқан тәуелді кестелердің тізімін көруге болады.

Кесте құрылымы сақтау кезені

Кестенің құрылымы жөніндегі барлық мәліметтер енгізілген соң Save as батырмасын басамыз. Ашылатын диалогтік терезе қарапайым сақтау терезесіне ұқсайды бірақ қарапайым терезеден бұл терезе Alias тізімімен ерекшеленеді. Бұл тізімнен кестені сақтайтын деректер базасын таңдауға болады. Егер сізге кестені

арнайы бір деректер базасына сақтау міндетті емес болса, онда жай ғана сақтау дегенді таңдаймыз. Және де мұнда жаңа бумa(каталог) құра аласыз. Paradox үшін деректер базасы – кестелер сақталған каталог екенін есіңізге түсіріңіз.



Кестені деректер базасында сақтау терезесі Сурет 2.9.1

Терезесін ашқан кезде төменгі жағында екі бөлікті көре аламыз. Біріншісі бұл – Display Table опциясы, ол кесте сақталғаннан соң оның автоматты түрде ашылуын анықтайды. Екіншісі – Add Data to New Table опциясына рұқсат тек қана кесте құрылған емес, құрылымы өзгертілген жағдайда беріледі.

2.4 ДБ пәнедрономін құру

Деректерді бір базада сақтау керек. Бұл үшін «Database Desktop» -та меню "Tools - AliasManager" командасын орындадым. Шыққан терезеде келесі әрекеттерді орындадым:

- Жаңа база құру үшін «New» батырмасын бастым;
- «Driver type» тізімінде «STANDARD» нұсқасын таңдадым;
- «Browse» батырмасын бастым және
  «Strani.db»
  «Samoleti.db»,
  «Poezda.db»,
  «Gostinica.db»,
  «Kartoteka.db»
  файлдары орнатылған директорияны таңдадым;
- «Database alias» өрісінде жаңа деректер базасының атын «Keep New» батырмасын басып енгіздім;
- «Ok» батырмасын бастым.

Database Desktop көмегімен кестені толтыру немесе құрылымын өзерту

Кесте құрылған соң оны File/Open командасы көмегімен аштым. Және де егер сіз Displey Table опциясын қолданған болсаңыз, онда кесте автоматты түрде ашылады. Table View Data командасы арқылы кесте құрылымын көре аласыз немесе Table/Edit Data командасымен кестені өзгертуге болады.

Table InfoStructure кесте құрылымы туралы ақпаратты көруге мүмкіндік береді, ал Table Restructure кестенің қандайда бір сипаттамаларын немесе құрылымын өзгертуге мүмкіндік береді. Бұл команданы орындаған жағдайда кесте құрылымын жасаған кездегі терезеге ұқсас терезе пайда болады.

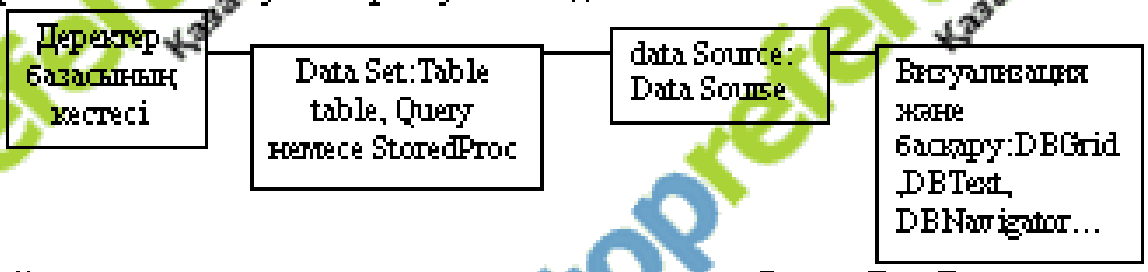
BDE деректер базаларымен байланысу үшін қолданылатын компоненттер

Delphi 7 және 6 VSL библиотекасындағы ДБ-мен жұмыс жасайтын компоненттері алғашқы версияларына қарағанда мүлдем басқа. Delphi 6-ға дейінгі версияларында BDE арқылы деректерге доступ беретін компоненттер DataAccess бетінде орналасқан. Delphi 7 және 6-да бұл бетте тек қана DataSource компоненті қалып қалғандары BDE бетіне ауыстырылған. Мәліметтерді редактілеу компоненттерінің барлығы DataControl бетінде орналасқан.

Деректер базасын қолданатын қосымшаның әр біреуінде келесі үш типтің кем дегенде біреуі болады:

- ДБ байланысатын деректерді теру (DataSet) – компоненттері, BDE үшін мұндай компоненттерге Table, Query, StoredProc сияқты компоненттер жатады;
- Деректер көзі (DataSource)' компоненті. Бірінші типті компоненттер мен визуализация компоненттері. Бұндай компонентке DataSource жатады;
- Визуализация және деректерді басқару компоненттері: DBGrid, DBText, DBEdit және т.б.

Мына схема көмегімен бұл компоненттердің бір бірімен және деректер базаларымен байланысуын көрсетуге болады:



Айтылған компоненттермен қосымшалардан басқа DataBase компоненті орналасуы мүмкін. Бұл компонент әдетте клиент/сервер платформасында жұмыс жасайтын қосымшаларды қолданылады. Ол жойылған сервермен байланысты қамтиды, транзакцияны жүзіге асырады, парольдермен жұмыс жасайды. Delphi автоматты түрде құратын компоненттерге Session компоненті жатады. Бұл деректер базаларымен жұмыс жасайтын қосымшалардың басты компоненті. Оны көпесепті қосымшаларға енгізбеңі немесе егер оларда параллельді бірнеше ақпарат ағымы өңделетін болған жағдайларда қолдану керек.

2.5 ДБ қосымшасын құру

Delphi –ді ашып, менюдің «File-New-Application» командасын орындадым. Бұл кезде қосымшаның жобасы «Form1» формасы құрылады.

«BDE» закладкасының палитра компоненттерінен «Table»-ді таңдадым.

Форманың кез келген жеріне орналастырдым. Нәтижесінде кесте символының суреті шықты.

«Data Access» закладкасына шертіп «DataSource» - ті таңдап, формаға орналастырдым.

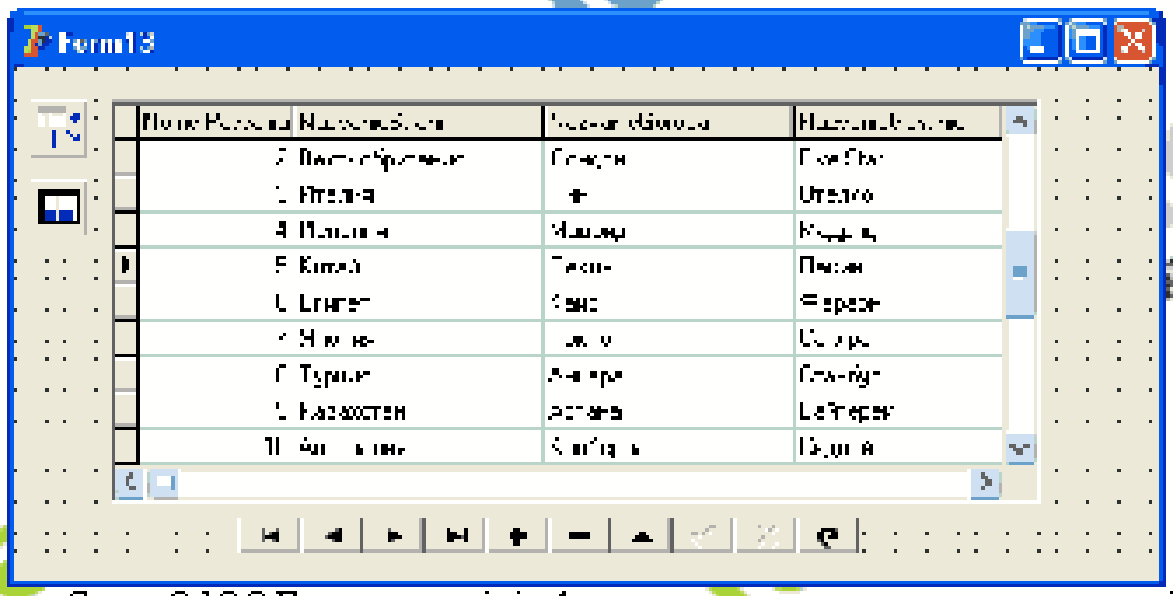
«Data Controls»- ті шертіп «DBGrid»-ті таңдап, формаға шерткен кезде енгізу облысы шықты.

Енді таңдалған элементтер арасында сіптемелер орнату керек. Формада «DBGrid1» компонентін белгіледім. «Object Inspector» (әдетте сол жақтың төменгі бұрышында орналасады) терезесінде «Properties»-ке өттім. «DataSource» жолын белгілеп берілген тізімнен «DataSource1» нұсқасын таңдадым.
Формамыздағы «DataSource1» компонентін белгілеп «Object Inspector» терезесінде «Properties»-ке өттім «DataSet» жолын белгілеп, тізімнен берілген «Table1» нұсқасын таңдадым.
Формамыздағы «Table1» » компонетімізді белгілеп «Object Inspector» терезесінде «Properties»-ке өттім. «DatabaseName» жолын белгілеп, оның тізімінен «Tur» нұсқасын таңдадым. Енді «TableName» жолын белгілеп, «Strani.db» кестесін таңдадым. Содан соң «Active» жолына өтіп «True» нұсқасына ауыстырдым. «DBGrid1» компонент терезесінде «Strani» кестесі шықты. 2.13.1 суретінде көрсетілген.

Сурет 2.13.1Table компонентін ДБ байланыстырып, ел кестесінің формасын құру

Data Controls компоненттер палитрасынан «DBNavigator» батырмасына бастым. Форманың кез келген жерде шерткен кезде келесі суретте көрсетілгендей «DBNavigator» батырмалары шығуы керек. «Object Inspector» терезесінде «Properties»-те «DataSource» жолын белгілеп, «DataSource1» нұсқасын таңдадым. Навигатор «DataSource1» ДБ-мен жұмыс жасаған кезде қолданушымызға керек болады. «Object Inspector» терезесінде «Properties» қасиеттерінің ішінен Visible Buttons қасиеті көмегімен керек емес батырмаларды алып тастауға болады. Мысалы, егер сіз қолданушыға жаңа жазба енгізуге рұқсат бергіңіз келмесе false-та nbInsert батырмасын орнатасыз. Егер редактілеуге түгел жол бергіңіз келмесе, онда тек қана nbFirst, nbPrior, nbNext, nbLast батырмаларын қалдырып, қалғандарын алып тастайсыз.
DBNavigator компоненті деректерді басқару үшін бірнеше батырмалары бар.

Олар: nbFirst – бірінші жазбаға өту;
nbPrior- алдыңғы жазбаға өту;
nbNext – келесі жазбаға өту;
nbLast – соңғы жазбаға өту;
nbInsert – ағымды жазбаның алдына жаңа жазба орнату;
nbDelete - ағымды жазбаны жою;
nbEdit – ағымды жазбаны түрлендіру;
nbPast – түрлендірілген ақпаратты деректер базасына жіберу;
nbCancel – жаңа жазба қосу немесе редактілеу нәтижелеріне жол бермеу;
nbRefresh – буферді тазалау.



Сурет 2.13.2 Елдер кестесінің формасына навигатор қосылған суреті

Содан соң осы әрекеттерді қайталап, қалған кестелерім үшін де осыған ұқсас формаларын құрдым.

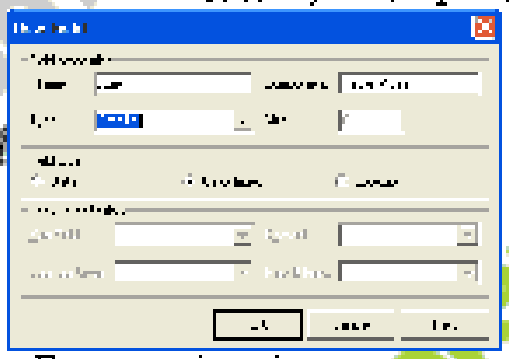
Өрістер редакторынан өрістерді формаға ауыстыру

Өрістер редакторының тағы да бір қасиетін қарастырайық. Бұл мүмкіндікті жасап көру үшін жұмыс істеп отырған қосымшамды сақтап қойып, жаңа қосымша аштым. Формаға Table компонентін орнатып, оны Kartoteka.db кестесімен байланыстырдым. Table1-ді екі рет шертіп, содан соң өрістер редакторында оң жақ батырмасын бастым. Мұнда барлық өрістерді (Add all fields) белгімін таңдаймыз.Терезеде кесте өрістерінің барлық атаулары шығады. Display Label қасиетінен олардың атауларын өзгертуге болады. Енді тышқанның оң жақ батырмасын басып Select all – барлық өрістерді белгілеу бөлімін таңдаймыз. Барлық өрістер белгіленген болады. Оларды тышқан көмегімен формаға апарамыз. Формада автоматты түрде әрбір өрістің деректерін көрсететін компонентер, автоматты түрде құрылғының көресіз және Display Label-де ауыстырған белгілері шыққанын көресіз. Енді оларды керекті ретті бойынша орналастырып, Table1 компонентінде Active қасиетін true-деп ауыстыру керек. Навигатор қосқаннан кейін қосымша жұмысқа дайын деп санауға болады.

Формада Data Source 1 – деректер көзін көресіз. Ол Table1 мен Data Set қасиеті бойынша байланысқан. Сонымен қатар формада тоғыз DBEdit компоненттері орналасқан. Оларда: Нөмірі, Фамилиясы, Аты, Әкесінің аты, Таңдалған ел, Қала, Қонақ үй ерістерінің мағұлматтарын көрсетеді. Мұнда байланысты Data Source қасиеті арқылы жасадым. Бұл қасиетін қарасаңыз ол автоматты түрде Data Source1 деп ауысқанын көресіз. Әрбір компоненттің Data Field қасиеттерінің мәндері өріс атауына сай келеді. Терезелерінде ағымды жазбаның мәндері көрсетіліп тұрады.Ал егер сіз жұмыс істеген кезде терезедегі мәнді өзгертсеңіз, онда бұл өзгерістер деректер базасына жазылады.

Есептелетін өрістер

Енді кестеде болмаған, жаңа өрісті қосу операцияларын қарастырайық. Бұның мәні басқа өрістер мәндерінің негізінде есептеледі. Мұндай өрістер есептелетін өрістер (Calculated fields) деп аталады. Біздің формамызға біз қызметкердің туған жылы бойынша жасын көрсететін өрісті қоямыз. Өрістер редакторын шақыру үшін Table1-ді екі рет шертеміз. Өрістер редакторында оң жақ батырмасын шертіп New бөлімін таңдадым, жаңа өріс қосу терезесі шығады.



Сурет 2.13.2.1 Есептелетін өріске ақпарат енгізу терезесі

Field properties бөлімінде өрістің атауын көрсетеміз, мысалы Age деп қойыңық. Типін(Type) - Smallint деп таңдаймыз, кейбір типтер үшін - мөлшерін(Size) көрсетеміз. Барлық мәндерді енгізгеннен кейін Field type радиобатырмалары Calculated – қа ауыстыма екенін тексеріңіз(бұл автоматты түрде жасалады). Содан соң Ok батырмасын шертіп өрістер редакторында Age өрісін көресіз.

Объектілер инспекторында Display Label-ді «жасы» деп өзгертесіз. Біз Age есептелетін өрісін енгіздік бірақ программаға оны қалай есептелетінің көрсеткен жоқсыз. Есептеу процедурасын көрсету үшін өрістер редакторына шығып Table1 компонентінің белгіленіз. Объектілерінспекторында оқиға бетіне өтіп OnCalcFields оқиғасына шертіңіз. Бұл оқиға есептелетін өрісті жаңарту керек кезінде орындалып тұрады.

Туған жылы бойынша жасын есептеу процедурасын былай жазуға болады.

```
Table1Age.Value:=2008-JasiValue;
```

Бірақ бұл оператор тек осы жылға, келесі жылы оны өзгерту керек болады. Бір екі жол қосып бұл операторды универсальды етуге болады:

```
ProcedureTForm1.Table1CalcFields(Data Set:TData Set);
Var Year, Month, Day:Word;
Begin
```

```
DecodeDate(Data, Year, Month, Day);
Table1.Age.Value:=Year-Jasi.Value;
End;
```

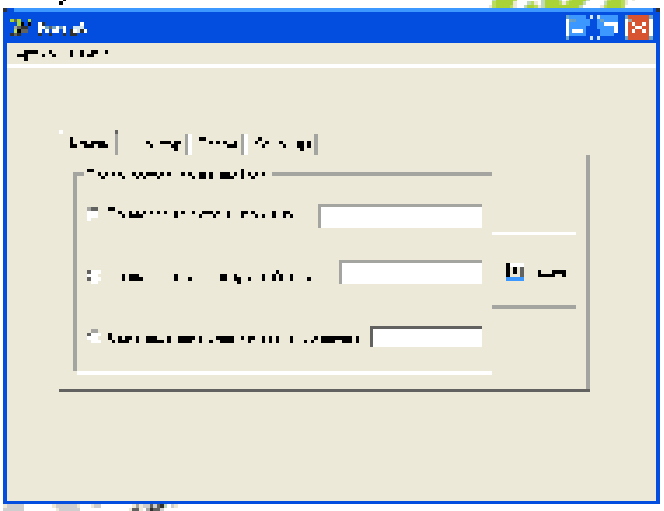
Бұл кодта Year, Month, Day айнымалылары енгізілген, олар ағымды жыл, ай және күнді сақтау үшін енгізілген.

Кестелердегі жазбаларды іздеуге арналған форманы құру

Енді жазбаларды іздеу үшін қолданған компоненттерді қарастырамын. Бұл үшін жаңа форма құрдым. Формаға WIN32 бөлігінен PageControl компонентін орналастырдым. Бұл компонент барлық кестелермен бірден жұмыс жасау үшін қолданылады. PageControl компонентіне төрт парақша қосып, кестелерге сәйкес атаулар қойдым.

Әрбір парақшасына бір RadioGroup компонентінен орналастырдым. RadioGroup компонентінің Items қасиетіне іздеу жасалатың өрістерін көрсеттім. Сонымен қатар Edit компонентінің керекті санын енгіздім. Бұлар ізделінетің жолдарды енгізу үшін қажет болады. Содан соң формаға «Іздеу» батырмасын енгізіп келесі код жаздым:

```
If RadioGroup1.ItemIndex=0 then
Begin
  Form2.Table1.IndexName:='nastrana';
  Form2.Table1.Active;
  Form2.Table1.SetKey;
Form2.Table1.FieldByName('NazvanieStrani').AsString:=Edit1.Text;
  Form2.Table1.GotoNearest;
```



Сурет 2.13.3.1ДБ кестелерінен деректерді іздеу формасы.

Жасалған әрекеттерді PageControl компонентінің барлық парақшаларына жасап шықтым. Және де өзімде қолданушыға жұмыс жасау барысында оңайырақ болу үшін MainMenu компонентін қолданып қосымша мәзір жасадым. Мысалы, Артқа сөзін шертіген кезде, оның ішінен қосымша: Негізгі форма, Елдер, Ұшақтар, Поезд, Қонақ үй деген сөздері пайда болады. Олардың әрбіреуіне өтіп, керекті формада қолданушы жұмысын жалғастыра бере алады.

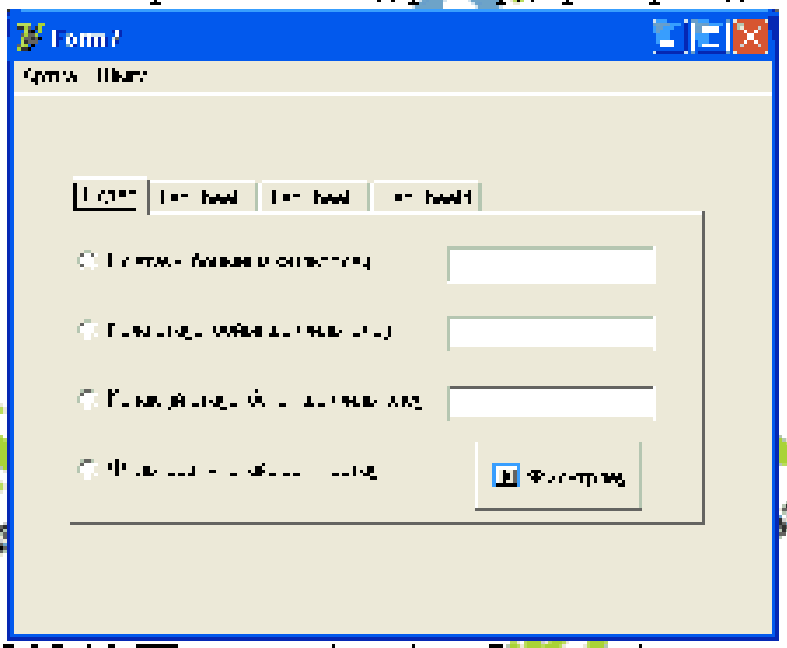
Кестелердегі жазбаларды фильтрлеу формасын құру

Жазбаларды фильтрлеу үшін келесі компоненттерді қарастырдым. Бұл үшін тағы бір жаңа форма ашып жоғарыда айтылып кеткен PageControl компонентің

орналастырдым. PageControl1 компонентіне төрт жаңа парақша қостым. Әр бір парақшасына RadioButton компонентінің керекті санын енгіздім. Атауларын Caption қасиетінен өзгертіп алдым. Енді RadioButton компоненттерінің санына сәйкес Edit компоненттерін енгіздім. ДБ кестелеріндегі деректерді қандай қасиеттері бойынша фильтрлеу керек екенің енгізу үшін Edit-ті қолландым. Іздеу формасындағыдай «Фильтрлеу» батырмасын енгізіп келесі кодті жаздым:

```
Form2.Table1.Filtered:=true;
if RadioButton6.Checked then
   Form2.Table1.Filter:='NazvanieStrani="'+Edit1.Text +'"';
```

Мына код «NazvanieStrani» өрісі бойынша деректерді фильтрлейді.



Сурет 2.13.4.1 ДБ кестелеріндегі жазбаларды фильтрлеу

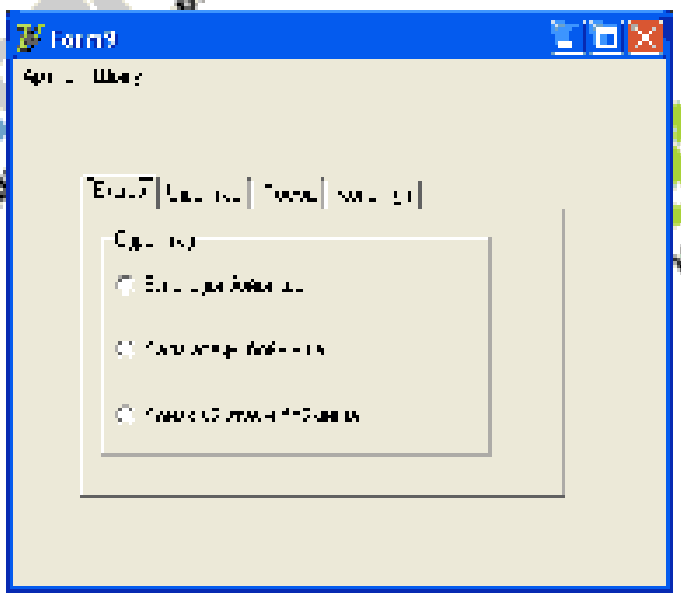
PageControl1 компонетінің қалған парақшаларына жоғарыда жасалынып кеткен әрекетерді қайталап шығамыз. Фильтрлеу формасы үшін да жоғарыда айтылған MainMenu1 компонентін қолландым.

Деректерді сұрыптау формасын құру

Сұрыптауға қолданылған компонентерді қарастырайық. Жаңа форма басқа формаларда қолданылып көрген PageControl1 компонентін орналастырамыз. PageControl1 компонентін таңдаған себебім барлық кестелердегі деректерге бірден өту мүмкіндігі, сондықтан бір формадан басқа формаға өтудің қажеті жоқ. PageControl1 компонентіне төрт жаңа парақша қосамыз. Әр бір парақшасына RadioGroup компонентін енгізіп, Items қасиетінде сұрыптау жүргізілетін өрістерімізді енгіздік. Әр бір RadioGroup компонентіне келесі кодті енгіздік:

```
Form2.Table1.IndexName:='';
  case RadioGroup1.ItemIndex of
  0:Form2.Table1.IndexName:='indstrani';
```

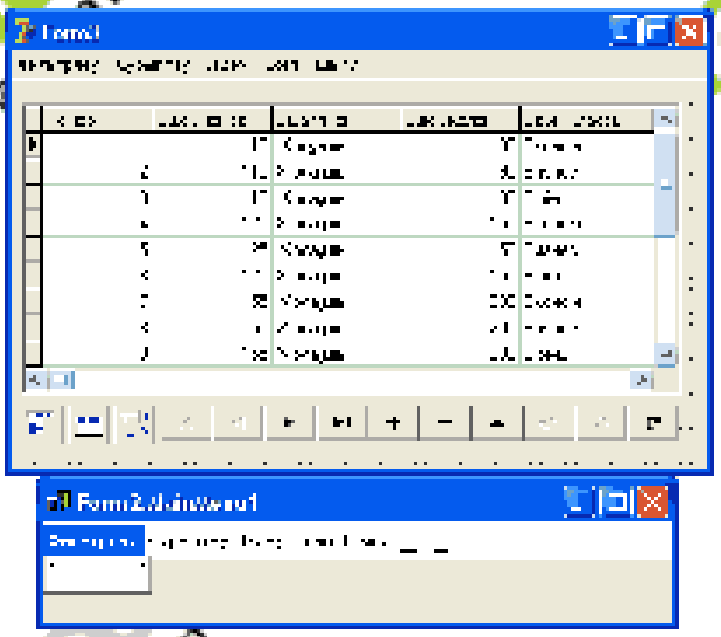
Бұл код тек қана бір өріс бойынша сұрыптайды, бірақ керек болса қажетті өрістеріңізді қоса аласыз.



Сурет 2.13.5.1Деректер базасын сұрыптау формасы

PageControl –дің қалған парақшаларына да осы әрекеттерді қайталаймыз.

Іздеу, фильтрлеу, сұрыптау, есеп, формадан шығу сияқты командаларды орындау оңайырақ болу үшін MainMenu компонентің қолданамыз. Items қасиетінде мәзіріміздің жарып аламыз.



Сурет 2.13.6.2 MainMenu компонентімен мәзір құру

Және де мәзірдің әр бір пунктіне сәйкес кодын жазып шықтық. Мысалы, «Іздеу» үшін:

```
Form2.Hide;
Form6.Show;
```

Бұл код іздеу формасына өтуге мүмкіндік береді. Осылай мәзіріміздің әрбір түймесіне мына кодқа ұқсас кодтарын жазып шықтық.

Есептер құру

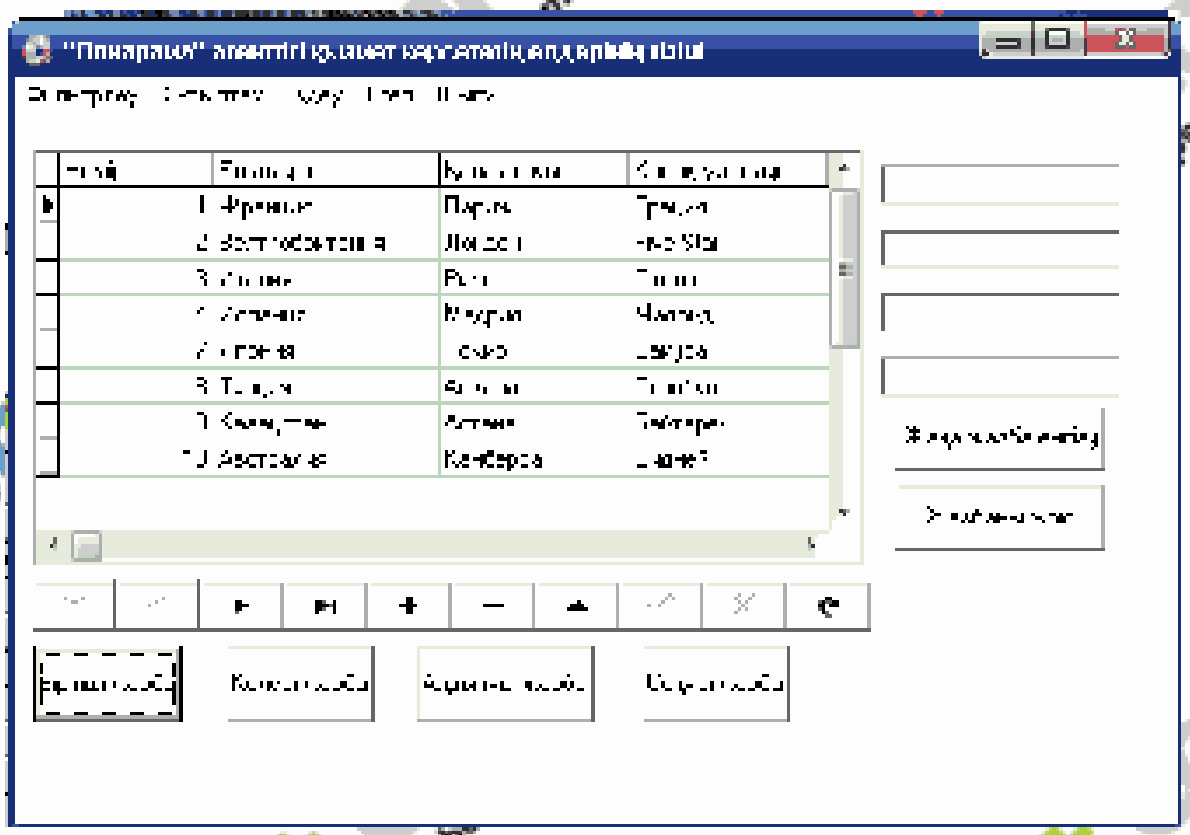
Есептер құру үшін мен Rave редакторын қолдандым. Бұл үшін әр бір формаға RvProject(RpRave) пен RvDataSetConnection компоненттерін орналастырдым. Содан соң Tools/Rave Designer командасын орындам Rave редакторына өттім. Мұнда File/New Report командасын орындап жаңа есеп құрдым. Есепке керекті жазбаларыңызды енгізіп болғаннан кейін File/Save as командасын орындап ДБ сақталып тұрған бумамды көрсетіп сақтадым. Енді Қайта формамызға ораламыз. Мұнда RvDataSetConnection1 компонентінің DataSet- тен Table1 – ді таңдадым, ал RvProject(RpRave)1 компонентінің ProjectFile қасиетіне шертіп есебім орналасқан жерін көрсеттім. Формамда алдын ала MainMenu компоненті көмегімен құрылып қойған мәзірімдегі «Есеп» деген сөзін шертіп келесі кодті жаздым: RvProject1.Execute; Осылай мен әр бір кестем бойынша, яғни бес кесте бойынша есептерді құрып, енгізіп қойдым.

2.6 Қолданушыға нұсқау

Курстық жұмыстың бұл бөлімінде Delphi 7 программалау ортасында құрылған «Панорама» туристтік агенттігінің деректер базасымен жұмыс жасау түсініктемесі жасалды. Бірінші қолданушыға құрылған деректер базамның тақырыбы мен мақсұдынын айтып кетейін. «Панорама» туристтік агенттігі демалуға арналған әлемнің әр түрлі жерлерін, сонымен қатар барған жерлерінде ең жақсы қонақ үйін ұсынады. Адамдар таңдалған қалаларына поезд немесе ұшақпен бара алады. Және де клиенттердің жеке мәліметтері бөлімінде біздің көмегімізді қолданған адамдардың жеке мәліметтері тіркеліп тұрады. Деректер базасы 11 формадан құрылған. Мұнда әр бір кестеге бөлек форма және сұрыптау, фильтрлеу, есеп, іздеу сияқты мүмкіндітері да бөлек формаларда құрылған. Қолданушы программаны ашқан кезде бірінші ден «Панорама» туристтік агенттігінің негізгі формасы ашылады. Негізгі формада ДБ кестелеріне жылдам және оңай өту үшін: «Елдер», «Ұшақтар», «Поезд», «Қонақ үй», «Клиенттер», «Программа туралы» және жұмысты аяқтағанда шығып кету үшін «Жұмысты аяқтау» батырмалары енгізілген.

Сурет 2.6.1 Деректерге өтудің басты терезесі

Бұл терезедегі батамалар арқышы басқа, керекті форманыға тез және оңай өте аласыз. Яғни әр бір батырманы шерткен кезде қосымша терезе ашылпады. Мысалы, «Елдер» батырмасын шерткен кезде, келесі терезе ашылпады:



Сурет 2.6.2 Демалу жерін анықтау терезесі

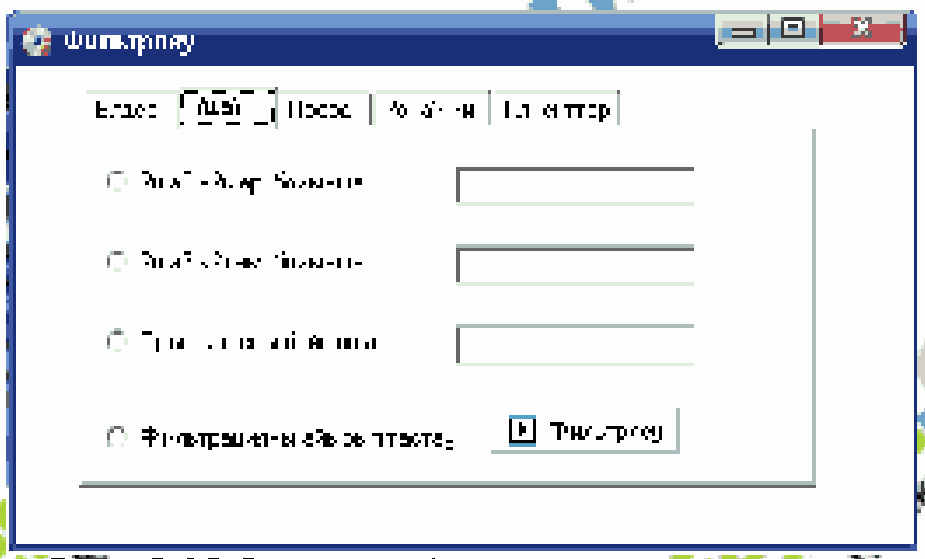
Бұл форма «Демалу жерін анықтау» деп аталады, яғни мұнда «Панорама» туристтік агенттігінің менеджері келген клиентке қызмет керсете алатын елдерінің, қалаларының, қонақ үйлерінің тізімін бере алады. Тізімнің астында навигатор орналасқан. Оның көмегімен қолданушы келесі операцияларды орындай алады:

- бірінші жазбаға өту;
- алдыңғы жазбаға өту;
- келесі жазбаға өту;
- соңғы жазбаға өту;
- жаңа жазба енгізу;
- жазбаны жою;
- ағымды жазбаны түрлендіру;
- түрлендірілген ақпаратты деректер базасына жіберу;
- редактілеу нәтижелеріне жол бермеу;
- буферді тазалау.

Қолданушы жұмыс істегенде навигаторды немесе «Бірінші жазба», «Келесі жазба», «Алдыңғы жазба», «Соңғы жазба» батырмалары көмегімен аналогті операцияларды орындай алады. Сонымен қатар терезенің оң жақ бөлігінде терт

ұяшық берілген, оларға жаңа жазба енгізіп «Жаңа жазба енгізу» батырмасын басқан кезде тізімде жазбаның енгізілгенін көресіз, ал «Жазбаны жою» батырмасы көмегімен керек емес жолды жойып жіберуге болады. Навигатормен қатар сәйкес функцияларды орындайтын батырмаларды енгізген себебім, қолданушының ыңғайлығы, яғни кейбір қолданушыларға навигатор ыңғайлы, ал кейбіреуі батырмалармен жұмыс істеуге үйренген.

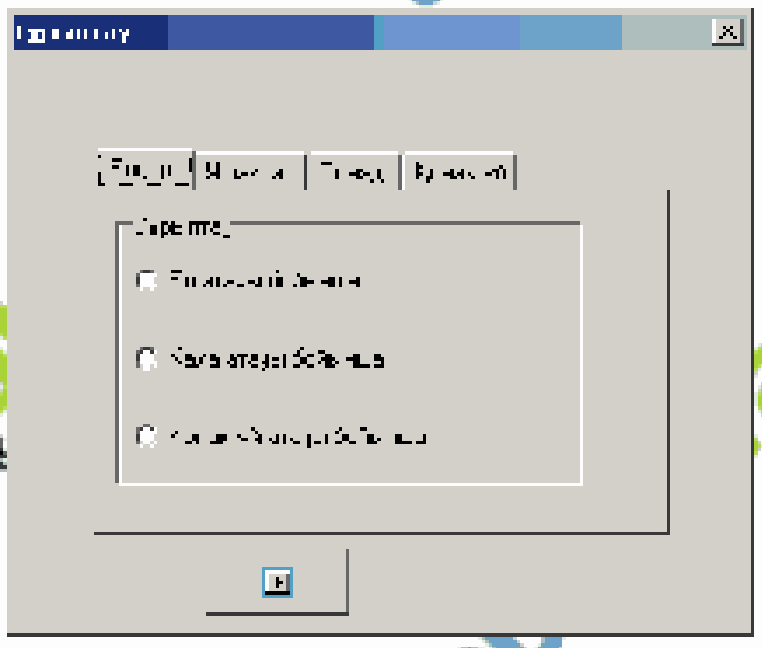
Және де терезенің жоғарғы бөлігінде қосымша мәзір берілген. Олар фильтрлеу, сұрыптау, іздеу немесе есеп дайындау керек болғанда қолданылады. Яғни «Фильтрлеу»-ді басқан кезде келесі терезе пайда болады:



Сурет 2.6.3 Фильтрлеуді орындауға арналған терезе

Мұнда берілген шарттары бойынша, ұяшықтарға мәліметтерді енгізіп «Фильтрлеу» батырмасын басасыз. Немесе фильтрацияны мүлдем айырып тастай аласыз.

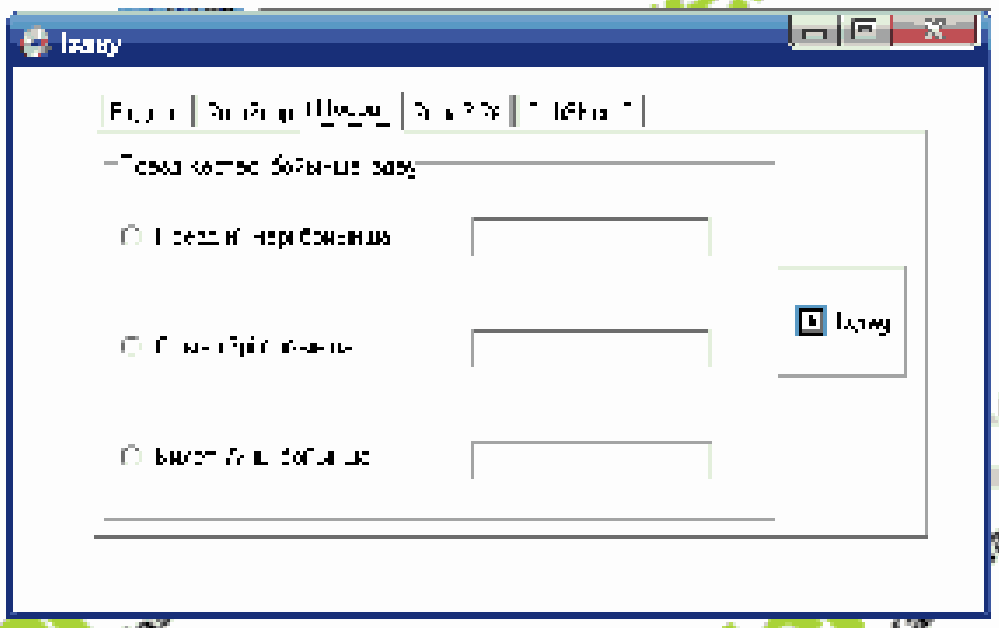
«Сұрыптау»-ды басқан кезде келесі терезе шығады:



Сурет 2.6.4 Сұрыптауды орындауға арналған терезе

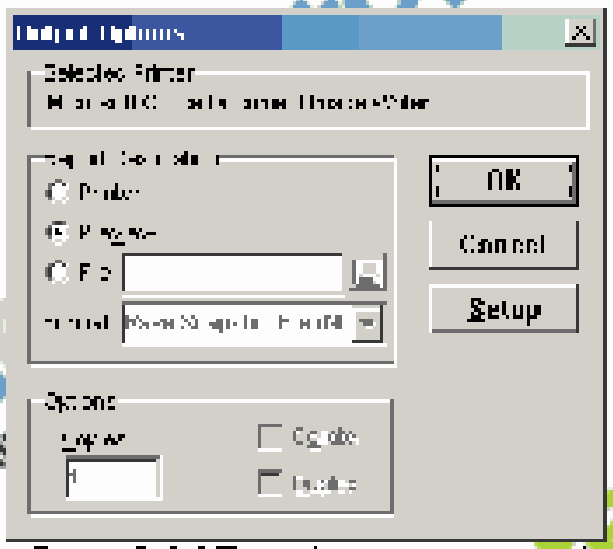
Мұнда сұрыптау шартын таңдайсыз, яғни сол шарт бойынша тізіміңіз сұрыпталып тұрады.

«Іздеу»-ді басып ашылған терезеде берілген ұяшыққа мәнін енгізіп, іздеу батырмасын басамыз. Мысалы, ел атауын енгізсеңіз, тізімде дәл ел белгіленіп тұрады.



Сурет 2.6.5 Іздеуді орындауға арналған терезе

«Есеп»-ті шерткен кезде Output Options терезесі шығады, мұнда OK батырмасын шерткен кезде есебіңізді көресіз.



Сурет 2.6.6 Есепті шығару терезесі

Демалу орнын таңдау кәсіпорнының есебі

MediaTravel

Номері	Ел атауы	Қала атауы	Қонақүй атауы
1	Франция	Париж	Трианон
2	Англия	Лондон	Ройял
3	Италия	Рим	Отелло
4	Испания	Мадрид	Мадрид
5	Қытай	Пекин	Пекин
6	Египет	Каир	Нефертити
7	Япония	Токио	Сакура
8	Түркия	Анкара	Стамбул
9	Қытайстан	Астана	Тай-Орда
10	Австралия	Канберра	Окенэй

Сурет 2.6.7 Құрылған есеп

Басты терезенің басқа батырмаларын басып, қарастырылған мысалдағыдай операцияларды орындауға болады.

Қорытынды

«Панорама» туристтік агенттігінің деректер базасын құру - мен үшін өте маңызды және қызықты жұмыс болды. Себебі бұл менің бірінші үлкен жұмысым болып саналады. Сондықтан оны құрған кезде көптеген қиындықтар болды, бірақ қазір осы курстық жұмыс көмегімен, мен өзімнің білім деңгейімді көтергенімді анық түсіндім.

«Панорама» туристтік агенттігінің деректер базасын шағын туризммен айналысатын мекемелерде қолдануға болады деп айтаймын. Себебі ол қолданушыға оңай және тез жұмыс жасауға мүмкіндік береді. Мұнда қысқаша, туризммен айналысатын мекемеге клиент келген кезде қолданушы бірден қандай елдер бойынша қызметтер ұйымдастыратын немесе қандай келісімге жетуге болатындығын көрсете алады.

Қазір адам қызмет ететін әрбір саласында компьютерді қолдану мүмкіндіктері өте жылдам дамып келе жатыр. Сондықтан болашақта осындай және одандағы күрделі деректер базаларын құру қажет болады екені анық.

Қолданылған әдебиеттер:

1. А.Д. Хомоненко, В.М. Цыганков «Базы данных», Санкт-Петербург, 2004г
2. А. Я. Архангельский «Программирование в Delphi 7», Москва Издательство БИНОМ, 2003
3. А. Я. Архангельский «Приемы программирование в Delphi», Москва Издательство БИНОМ, 2003
4. Гофман В., Хомоненко А. «Delphi 6» СП БХВ – Петербург, 2003г.
5. Гофман В., Хомоненко А. «Delphi. Быстрый старт», СП БХВ – Петербург, 2003г.
6. Фаронов В.В. «Delphi. Программирование на языке высокого уровня», Питер, 2003г.
7. Грофф Дж., Вайнберг П.Энциклопедия SQL. 3-е изд+CD. СПб: «Питер»,2003.
8. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных.
9. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. Базы данных: Учебник для высших учебных заведений /Под ред. проф. А. Д. Хомоненко. Спб, КОРОНА принт, 2000
10. Бейт К Дж. Введение в системы баз данных . Издание шестое М., 1998г.
11. Джек Л. Харрингтон. Проектирование реляционных баз данных М., 2000г.
12. Мартин Грабен Введение в SQL М.,2000г.
13. Мартин Дж. Организация баз данных в вычислительных системах. М., «Мир», 1985г.
14. Под редакцией А. Д. Хоменко. Базы данных : учебников для высших и средних учебных заведений. С-Нб. 2003.
15. Понамарева К.В. Кузьмин Л.Г. Информационное обеспечение АЛС М., 1991 г.
16. Роберт С. Microsoft Access М . ,2000: Учебный курс С – Нб. 2002г.
17. Ю. Бекаревич Н. Пушкина . Microsoft Access 2000. 1999 г.
18. Дейт К. «ВВедение в системы баз данных», Москва, 1980 г.

Негізгі форма мәтіндері

```
unit Unit11;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, DBCtrls, StdCtrls, Mask, DB, DBTables, Menus, RpCon,
  RpConDS, RpDefine, RpRave, Grids, DBGrids;

type
  TForm11 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    Table1NomerPassajira: TFloatField;
    Table1Fam: TStringField;
    Table1Name: TStringField;
    Table1SecName: TStringField;
    Table1Birthday: TDateField;
    Table1VidTransporta: TStringField;
    Table1Strana: TStringField;
    Table1Gorod: TStringField;
    Table1Gostinica: TStringField;
    Table1VidNomer: TStringField;
    Table1DataViezda: TDateField;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    Label5: TLabel;
    DBEdit5: TDBEdit;
    Label6: TLabel;
    DBEdit6: TDBEdit;
    Label7: TLabel;
    DBEdit7: TDBEdit;
    Label8: TLabel;
    DBEdit8: TDBEdit;
    Label9: TLabel;
    DBEdit9: TDBEdit;
    Label10: TLabel;
    DBEdit10: TDBEdit;
```

```
    Label11: TLabel;
    DBEdit11: TDBEdit;
    DBNavigator1: TDBNavigator;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    RvProject1: TRvProject;
    RvDataSetConnection5: TRvDataSetConnection;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    procedure N2Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N4Click(Sender: TObject);
    procedure N5Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form11: TForm11;
implementation
uses Unit2, Unit1, Unit6, Unit7;
{$R *.dfm}
procedure TForm11.N2Click(Sender: TObject);
begin
Form11.Close;
end;
procedure TForm11.N1Click(Sender: TObject);
begin
Form11.Hide;
Form1.Show;
end;
procedure TForm11.N3Click(Sender: TObject);
begin
RvProject1.Execute;
end;
procedure TForm11.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Show;
end;
```

```
procedure TForm11.N4Click(Sender: TObject);
begin
Form7.Show;
end;
procedure TForm11.N5Click(Sender: TObject);
begin
Form6.Show;
end;
end.
```

Екінші форма листингі

```
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, DBTables, Grids, DBGrids, StdCtrls, ExtCtrls, DBCtrls, Menus,
  RpCon, RpConDS, RpDefine, RpRave;
type
  TForm2 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    DBNavigator1: TDBNavigator;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Table1NazvanieStrani: TStringField;
    Table1NazvanieGoroda: TStringField;
    Table1NazvanieGostinici: TStringField;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    Table1NomerPassajira: TFloatField;
    DBGrid1: TDBGrid;
    RvProject1: TRvProject;
    RvDataSetConnection1: TRvDataSetConnection;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
```

```
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
uses Unit1, Unit6, Unit4, Unit9;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Table1.First;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
Table1.Next;
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
Table1.Prior;
end;

procedure TForm2.Button4Click(Sender: TObject);
begin
Table1.Last;
end;
procedure TForm2.Button5Click(Sender: TObject);
begin
Table1.Insert;
Table1.FieldByName('NomerPassajira').AsString:= AnsiUpperCase(Edit1.Text);
Table1.FieldByName('NazvanieStrani').AsString:= AnsiUpperCase(Edit2.Text);
```

```
Table1.FieldByName('NazvanieGoroda').AsString:=AnsiUpperCase( Edit3.Text);
Table1.FieldByName('nazvanieGostinici').AsString:=AnsiUpperCase(Edit4.Text);
end;
procedure TForm2.Button6Click(Sender: TObject);
begin
if MessageDlg('Жазбаны жойғыңыз келеме?',mtConfirmation,
[mbYes, mbNo] ,0)=mrYes then
Table1.Delete;
end;
procedure TForm2.N5Click(Sender: TObject);
begin
Form2.Hide;
Form1.Show;
end;
procedure TForm2.N3Click(Sender: TObject);
begin
Form6.Show;
Form6.PageControll .TabIndex:=0;
end;

procedure TForm2.N1Click(Sender: TObject);
begin
Form7.Show;
Form7.PageControll .TabIndex:=0;
end;
procedure TForm2.N2Click(Sender: TObject);
begin
Form9.Show;
Form9.PageControll .TabIndex:=0;
end;
procedure TForm2.N4Click(Sender: TObject);
begin
RvProject1.Execute;
end;
procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Show;
end;
end.
```

Үшінші форма листингі

```
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
  Dialogs, Menus, DB, DBTables, Grids, DBGrids, ExtCtrls, DBCtrls, RpCon,
  RpConDS, RpDefine, RpRave;
type
  TForm3 = class(TForm)
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    Table1: TTable;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    DBNavigator1: TDBNavigator;
    Table1NomerPassajira: TFloatField;
    Table1NomerSamoleta: TFloatField;
    Table1VidSamoleta: TStringField;
    Table1VmestimostSamoleta: TIntegerField;
    Table1Klassi: TStringField;
    Table1StoimostBileta: TCurrencyField;
    RvProject1: TRvProject;
    RvDataSetConnection2: TRvDataSetConnection;
    procedure N5Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
implementation
uses Unit1, Unit6, Unit9, Unit7;
{$R *.dfm}
procedure TForm3.N5Click(Sender: TObject);
begin
Form3.Hide;
Form1.Show;
end;
procedure TForm3.N3Click(Sender: TObject);
begin
```

```
Form6.Show;
Form6.PageControll.TabIndex:=1;
end;
procedure TForm3.N2Click(Sender: TObject);
begin
Form9.Show;
Form9.PageControll.TabIndex:=1;
end;
procedure TForm3.N1Click(Sender: TObject);
begin
Form7.Show;
Form7.PageControll.TabIndex:=1;
end;
procedure TForm3.N4Click(Sender: TObject);
begin
RvProject1.Execute;
end;
procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Show;
end;
end.
```