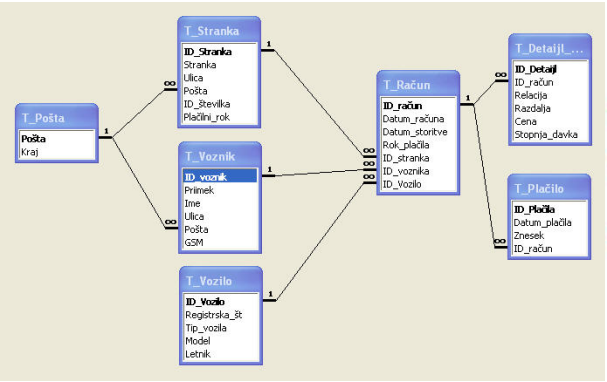
Базе података

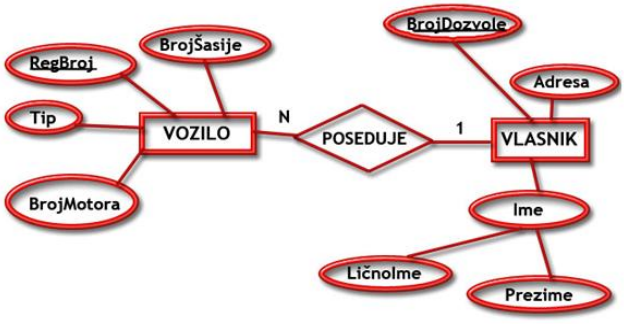
База података је скуп међусобно повезаних података, смештених на неком медију погодном за коришћење од стране великог броја корисника. Тема нашег интересовања је специјални облик базе података чије датотеке су смештене у спољашњој меморији рачунара. Подаци у овако организованој бази су истовремено доступни разним корисницима и апликацијским програмима. Унос, измена, брисање и читање података обавља се посредством заједничког софтвера. Корисници апликације приликом употребе базе не морају познавати детаље физичког приказа података, већ су упућени на логичку структуру базе. Такође, корисници не морају имати иста прева и овлашћења, већ могу постојати они којима је дозвољено само читање података и они који могу и уписивати податке или мењати и брисати постојеће. Права и овлашћења корисника базе могу се дефинисати постојањем шифара различитог нивоа или на неки други начин.

Систем за управљање базом података (***Data Base Management System - DBMS***) је сервер базе података. Он обликује физички приказ базе у складу с траженом логичком структуром. Такође, он обавља у име клијената све операције с подацима. Даље, он је у стању подржати разне базе, од којих свака може имати своју (другачију) логичку структуру, но у складу с истим моделом. Исто тако, брине се за сигурност података, аутоматизује административне послове с базом. Појавам рачунара и њихове масовне употребе проузроковала је појаву база података у електронском облику. У почетку су постојале значајне разлике у организацији података, што је као последицу имало немогућност коришћења база које су формиране у једном програму из апликација направљених у неком другом програму. Оваква некомфорност довела је до увођења стандарда за рад и коришћење база података. Највећи број данашњих ситема база података је у складу са ANSI (American National Standard Institute) стандард. Подаци у бази су логички организовани у складу с неким моделом података. Модел података је скуп правила која одређују како може изгледати логичка структура базе. Модел чини основу за планирање, пројектовање и имплементирање базе. Досадашњи ***DBMS***-и обично су подржавали неки од следећих модела:

Релациони модел. Заснован на математичком појму релације. И подаци и везе међу подацима приказују се "правоугаоним" табелама.

Мрежни модел. База је представљена усмереним графом. Чворови су типови записа, а лукови дефинишу везе међу типовима записа.

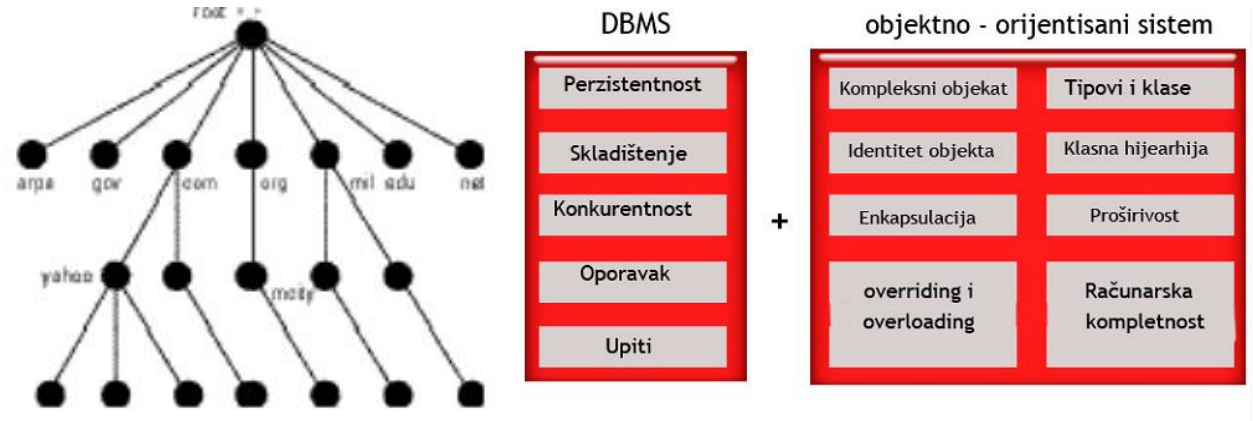




Хијерархијски модел. Специјални случај мрежног. База је представљена једним стаблом или скупом стабала. Чворови су типови записа, а хијерархијски однос "надређени-подређени" изражава везе међу типовима записа.

Објектни модел. Инспирисан је објектно-оријентисаним програмским језицима. База је скуп трајно сачуваних објеката који се састоје од својих интерних података и "метода" (операција) за руковање с тим подацима. Сваки објекат припада некој класи. Између класа се успостављају везе наслеђивања, агрегације, односно, међусобног коришћења операција.

Хијерархијски и мрежни модел били су у употреби у 60-тим и 70-тим годинама 20. века. Од 80-тих година, па све до данашњих дана преовладава релацијски модел. Очекивани прелаз на објектни модел за сада се није десио, тако да данашње базе података углавном још увек можемо поистоветити с релацијским базама.



Технологија база података настоји (и у великој мери успева) испунити следеће циљеве:

- **Физичка независност података**. Раздваја се логичка дефиниција базе од њене стварне физичке грађе. Значи, ако се физичка грађа промени (на пример, подаци се препишу у друге датотеке на другим дисковима), то неће захтевати промене у постојећим апликацијама.
- ***Логичка независност података***. Раздваја се глобална логичка дефиниција целе базе података од локалне логичке дефиниције за једну апликацију. Значи, ако се логичка дефиниција промени (на пример уведе се нови запис или веза), то неће захтевати промене у постојећим апликацијама. Локална логичка дефиниција обично се своди на издвајање само неких елемената из глобалне дефиниције, уз неке једноставне трансформације тих елемената.
- **Флексибилност приступа подацима**. У старијим мрежним и хијерархијским базама, стазе приступања подацима биле су унапред дефинисане, дакле корисник је могао претраживати податке једино оним редоследом који је био предвиђен у време пројектовања и имплементирања базе. Данас се захтева да корисник може слободно пребирати по подацима и по свом нахођењу успостављати везе међу подацима. Овом захтеву заиста задовољавају једино релацијске базе.
- **Истовремени приступ подацима**. База мора омогућити да већи број корисника истовремено користи исте податке. Притом ти корисници не смеју ометати један другога, сваки од њих треба имати утисак да сам ради с базом.
- **Чување интегритета**. Настоји се аутоматски сачувати коректност и конзистенција података, и то у ситуацији кад постоје грешке у апликацијама и конфликтне истровремене активности корисника.
- **Могућност опоравка након квара**. Мора постојати поуздана заштита базе у случају квара хардвера или грешака у раду системског софтвера.
- **Заштита од неовлашћеног коришћења**. Мора постојати могућност да се корисницима ограниче права коришћења базе, односно, да се сваком кориснику регулишу овлашћења шта он сме, а шта не сме радити с подацима.
- **Задовољавајућа брзина приступа**. Операције с подацима морају се одвијати довољно брзо, у складу с потребама одређене апликације. На брзину приступа може се утицати одабиром погодних физичких структура података и избором погодних алгоритама за претраживање.
- **Могућност подешавања и контроле**. Велика база захтева сталну бригу: праћење перформанси, мењање параметара у физичкој грађи, рутинско чување резервних копија података, регулисање овлашћења корисника. Такође, сврха базе се временом мења, па повремено треба подесити и логичку структуру. Овакви послови морају се обављати централизовано. Одговорна особа зове се **администратор** базе података. Администратор има на располагању разне алате и помагала.

Архитектура базе података састоји се од три "слоја" и односа међу слојевима, три нивоа апстракције:

Физички ниво односи се на физички приказ и распоред података на јединицама спољне меморије. То је аспект који виде само системски програмери (они који су развили DBMS). Сам физички ниво може се даље поделити на више поднивоа апстракције, од сасвим конкретних стаза и цилиндара на диску, до већ донекле апстрактних појмова датотеке и записа какве сусрећемо у класичним програмским језицима. Распоред чувања описује како се елементи логичке дефиниције базе пресликавају на физичке уређаје.

Глобални логички ниво односи се на логичку структуру целе базе. То је аспект који види пројектант базе, односно, њен администратор. Запис логичке дефиниције назива се **схема** (енгл. ***schema***). Схема је текст или дијаграм који дефинише логичку структуру базе, и у складу је са задатим моделом. Именују се и дефинишу сви типови података и везе међу тим типовима, у складу с правилима коришћеног модела. Такође, схема уводи и ограничења којима се чува интегритет података.

Локални логички ниво односи се на логичку представу о делу базе који користи појединачна апликација. То је аспект који види корисник или апликацијски програмер. Запис једне локалне логичке дефиниције зове се **поглед** (енгл. ***view***) или подсхема. То је текст или дијаграм којим се именују и дефинишу сви локални типови података и везе међу тим типовима, у складу с правилима коришћеног

модела. Такође, поглед задаје пресликавање којим се из глобалних података и веза изводе локални. За стварање базе података потребно је задати само схему и погледе. ***DBMS*** тада аутоматски генерише потребан распоред чувања и физичку базу. Администратор може само донекле утицати на физичку грађу базе, подешавањем њему доступних параметара. Програми и корисници не приступају директно физичкој бази, већ добијају или чувају податке посредством ***DBMS***-а. Комуникација програма, односно, корисника с ***DBMS***-ом обавља се на локалном логичком нивоу.

Језици за рад с базама података

Комуникација корисника, односно, апликацијског програма и ***DBMS***-а одвија се помоћу посебних језика. Ти језици традиционално се деле на следеће категорије.

Језик за опис података (***Data Description Language - DDL***). Служи пројектанту базе или администратору у сврху записивања схеме или погледа. Тим језиком дефинишемо податке и везе међу подацима, и то на логичком нивоу. Понекад постоји посебна варијанта језика за схему, а посебна за погледе. Наредбе ***DDL*** обично подсећају на наредбе за дефинисање сложених типова података у језицима попут ***COBOL, PL/I, C, Pascal***.

Језик за манипулирање подацима (***Data Manipulation Language - DML***). Служи програмеру за успостављање везе између апликацијског програма и базе. Наредбе ***DML*** омогућавају "маневрисање" по бази и једноставне операције као што су упис, промена, брисање или читање записа. У неким софтверским пакетима, ***DML*** је заправо библиотека потпрограма, односно, "наредба" у ***DML*** своди се на позив потпрограма. У другим пакетима заиста се ради о посебном језику, тј. програмер пише програм у којем су измешане наредбе двају језика, па такав програм треба преводити с два преводиоца (***DML-precompiler*** и обични компајлер).

Језик за постављање упита (***Query Language - QL***). Служи непосредном кориснику за интерактивно претраживање базе. То је језик који подсећа на говорни (енглески) језик Наредбе су непроцедуралне, односно, само спецификују резултат који желимо добити, али не и поступак за добијање резултата.

Оваква подела на три језика је данас прилично застарела. Наиме, код релацијских база постоји тенденција да се сва три језика обједине у један свеобухватни. Пример таквог интегрисаног језика за релацијске базе је ***SQL***. Он служи за дефинисање података, манипулисање и претраживање. Интегрисани језик се може користити интерактивно (преко ***on-line*** интерпретера) или се може појављивати уклопљен у апликацијске програме. Наравно, треба знати да поменуте врсте језика не представљају програмске језике. То су језици који су нам потребни да бисмо се повезали с базом, али они нам нису довољни за развој апликација које ће нешто радити с подацима из те базе. Традиционални начин развоја апликација које раде с базом је коришћење класичних програмских језика (***COBOL, PL/I, C, Pascal,*** ...) са уграђеним ***DML***-наредбама. У 80-тим годинама 20. века били су доста популарни и тзв. ***језици четврте генерације*** (***4-th Generation Languages - 4GL***). Реч је о језицима који су били намењени искључиво за рад с базама, те су зато у том контексту били продуктивнији од класичних програмских језика опште намене. Проблем са овим језицима је био у њиховој нестандардности, тј. сваки од њих је по правилу био део неког одређеног софтверског пакета за базе података и није се могао користити изван тог пакета (базе). У данашње време, апликације се најчешће развијају у стандардним објектно-оријентисаним програмским језицима (***Java, C++,*** ...). За интеракције с базом користе се унапред припремљене класе објеката. Оваква техника је довољно продуктивна због коришћења готових класа, а резултујући програм се лако дотерује, уклапа у веће системе или преноси с једне базе на другу.

Базе података се по правилу реализују коришћењем неког од проверених софтверских пакета. Готово сви данашњи софтверски пакети подржавају релацијски модел ***SQL***. Сваки од њих садржи свој ***DBMS***, уобичајене клијенте (на пример интерактивни интерпретер ***SQL***), библиотеке и алате за развој апликација. Сваки пакет испоручује се у верзијама за разне рачунарске платформе (оперативне системе). Конкуренција међу производачима софтвера за базе података је изузетно велика, тако да је последњих година често долазило до њиховог нестанка, спајања или преузимања. Листа релевантних софтверских пакета зато је сваке године све краћа. Једино освежење представља појава ***public-domain*** софтвера попут ***MySQL***.

Животни циклус базе података

Увођење базе података у неко предузеће или установу представља сложени задатак који захтева тимски рад стручњака разних профила. То је пројект који се може поделити у пет фаза: анализа потреба, моделирање података, имплементација, тестирање и одржавање.

Анализа потреба. Проучавају се токови информација у предузећу. Уочавају се подаци које треба чувати и везе међу њима. У великим предузећима, где постоје разне групе корисника, појавиће се разни "погледи" на податке. Те погледе треба ускладити тако да се елиминише редунданција и неконзистентност. На пример, треба у разним погледима препознати синониме и хомониме и ускладити терминологију. Анализом потреба такође треба обухватити анализу трансакција (операција) које ће се обављати над базом података, будући да то може имати утицаја на садржај и коначни облик базе. Важно је проценити фреквенцију и обим појединих трансакција и захтеве за перформансама. Резултат анализе је документ (писан неформално у природном језику) који се зове спецификација потреба.

Моделирање података. Различити погледи на податке, откривени у фази анализе, синтетишу се у једну целину - глобалну схему. Прецизно се утврдују типови података. Схема се даље дотерује ("нормализује") тако да задовољи неке захтеве квалитета, прилагођава се ограничењима које поставља задати модел података и додатно се модификује да би боље могла удовољити захтевима за перформансе. На крају се из схеме изводе погледи (подсхеме) за поједине апликације (групе корисника).

Имплементација. На основу схеме и подсхема и уз помоћ доступног ***DBMS***-а, физички се реализује база података на рачуналу. У ***DBMS***-у обично постоје параметри којима се може утицати на физичку организацију базе. Параметри се подешавају тако да се осигура ефикасан рад најважнијих трансакција. Развија се скуп програма који реализирају поједине трансакције те покривају потребе разних апликација. База се иницијално пуни подацима.

Тестирање. Корисници раде с пробном базом и проверавају да ли она задовољава све захтеве. Настоје се открити грешке које су се могле поткрасти у свакој од фаза развоја: дакле у анализи потреба, моделирању података, имплементацији. Грешке у ранијим фазама имају теже последице. На пример, грешка у анализи потреба узрокује да трансакције можда коректно раде, али не оно што корисницима треба већ нешто друго. Добро би било такве пропусте открити пре имплементације. Зато се у новије време, пре праве имплементације, развијају и приближни прототипови базе података и показују корисницима. Јефтину израду прототипова омогућавају језици четврте генерације и објектно-оријентисани језици.

Одржавање. Одвија се у време кад је база већ ушла у редовну употребу. Састоји се од: поправки грешака које нису откривене у фази тестирања; увођења промена због нових захтева корисника; подешавање параметара у ***DBMS*** ради побољшања перформанси. Одржавање захтева да се стално прати рад с базом, и то тако да то праћење не омета кориснике. Администратор базе података мора имати на располагању одговарајући алат (***utility*** програми).

Моделирање података

Након извршене детаљне анализе потреба корисника треба обликовати схему за базу података, усклађену с правилима релацијског модела. У стварним ситуацијама доста је тешко директно погодити релацијску схему. Зато се служимо једном помоћном фазом која се зове моделирање ентитета и веза (***Entity-Relationship Modelling*** - **ER** моделирање). Реч је о обликовању једне мање прецизне, концептуалне схеме, која представља апстракцију реалног света. Та тзв. ER-схема се даље, више-мање аутоматски, претвара у релацијску. Моделирање ентитета и веза захтева да се свет посматра преко три категорије:

- **ентитети** - објекти или догађаји који су нам од интереса;
- **везе** - односи међу ентитетима који су нам од интереса;
- **атрибути** - својства ентитета и веза која су нам од интереса.

Ентитет је нешто о чему желимо чувати податке, нешто што је у стању постојати или не постојати, и може се идентификовати. Ентитет може бити објекат или биће (на пример *кућа, ученик, ауто*), односно догађај или појава (на пример *фудбалска утакмица, празник, сервисирање аута*). Ентитет је описан атрибутима (на пример атрибути *куће* су: *адреса, број спратова, боја фасаде, ...*). Уколико неки атрибут и сам захтева своје атрибуте, тада га је боље сматрати новим ентитетом (на пример *модел аута*). Исто правило важи и ако атрибут може истовремено имати више вредности (на пример *квар* који је поправљен при сервисирању аута). Име ентитета, заједно са припадајућим атрибутима одредује тип ентитета. Може постојати много примерака (појава) ентитета задатог типа (на пример *ученик* је тип чији примерци су *Петровић Петар, Марковић Марко, ...*).

Кандидат за **кључ** је атрибут, или скуп атрибута, чије вредности једнозначно одређују примерак ентитета заданог типа. Дакле, не могу постојати два различита примерка ентитета истог типа с истим вредностима кандидата за кључ. (На пример за тип ентитета *ауто*, кандидат за кључ је атрибут *регистарски број*). Уколико један тип ентитета има више кандидата за кључ, тада бирамо један од њих и проглашавамо га **примарним кључем**. (На пример примарни кључ за тип ентитета *ученик* могао би бити атрибут *број уписнице*).

Везе се успостављају између два или више типова ентитета (на пример веза *игра за* између типова ентитета *играч* и *тим*). Заправо је реч о именованој бинарној или к-нарној релацији између примерака ентитета задатих типова. За сада ћемо се ограничити на везе између тачно два типа ентитета. Функционалност везе може бити:

Један-према-један (1 : 1). Један примерак првог типа ентитета може бити у вези с највише једним примерком другог типа ентитета и један примерак другог типа може бити у вези с највише једним примерком првог типа.

Један-према-много (1 : Н). Један примерак првог типа ентитета може бити у вези с 0, 1 или више примерака другог типа ентитета, али један примерак другог типа може бити у вези с највише једним примерког првог типа.

Много-према-много (М : Н). Један примерак првог типа ентитета може бити у вези с 0, 1 или више примерака другог типа ентитета и један примерак другог типа може бити у вези с 0, 1 или више примерака првог типа. Веза може имати и своје атрибуте које не можемо приписати ни једном од типова ентитета. Ако сваки примерак ентитета неког типа мора бити у датој вези, тада кажемо да тип ентитета има обавезно чланство у тој вези, у противном има необавезно чланство. Одлука да ли је чланство обавезно или необавезно често је ствар договора, односно, пројектантове одлуке. Обичај је да се ЕР-схема нацрта као дијаграм у којем правоугаоници представљају типове ентитета, а ромбови везе. Везе су повезане ивицама с одговарајућим типовима ентитета. Имена типова ентитета и веза и функционалност веза, унесени су у дијаграм. Посебно се прилаже листа атрибута за сваки ентитет односно везу. У тој листи можемо спецификовати обавезност чланства у везама. У стварним ситуацијама појављују се и сложеније везе од оних које смо до сада посматрали.. На пример:

- **Инволуирана веза** повезује један тип ентитета с тим истим типом. Дакле реч је о бинарној релацији између разних примерака ентитета истог типа. Функционалност такве везе може бити (1 : 1), (1 : Н), односно (М : Н).
- **Тернарне везе** успостављају се између три типа ентитета. Значи реч је о тернарној релацији између примерака трију типова ентитета. Постоје бројне могућности за функционалност тернарне везе, на пример (Н : М : П), (1 : Н : М), (1 : 1 : Н) или чак (1 : 1 : 1). Тернарну везу уводимо само онда кад се она не може раставити на две бинарне.

ЕР модел довољно је једноставан да га људи различитих струка могу разумети. Зато ЕР схема служи за комуникацију пројектанта базе података и корисника, и то у најранијој фази развоја базе. Постојећи ***DBMS*** не могу директно имплементирати ЕР схему, већ захтевају да се она детаљније разради и модификује у складу с правилима релацијског, мрежног, односно хијерархијског модела.

Релацијски модел

Релацијски модел био је теоретски заснован још крајем 60-тих година 20. века, у радовима Кода (***E. F. Codd***). Модел се дуго појављивао само у академским расправама и књигама. Прве реализације на рачунару биле су сувише споре и неефикасне. Захваљујући интензивном истраживању и напретку самих рачунара, ефикасност релацијских база постепено се побољшавала. Средином 80-тих година 20. века релацијски модел је постао превладавајући. И данас већина ***DBMS*** користи тај модел. Релацијски модел захтева да се база података састоји од скупа правоугаоних табела - тзв. **релација**. Свака релација има своје име по којем је разликујемо од осталих у истој бази. Један стубац релације обично садржи вредност једног **атрибута** (за ентитет или везу) - зато стубац поистовећујемо с атрибутом и обрнуто. Атрибут има своје име по којем га разликујемо од осталих у истој релацији. Вредности једног атрибута су подаци истог типа. Дакле, дефинисан је скуп дозвољених вредности за атрибут, који се зове **домен атрибута**. Вредност атрибута мора бити једнострука и једноставна (не може се раставити на делове). Под неким условима толеришемо ситуацију да вредност атрибута недостаје (није уписана). Један ред релације обично представља један примерак ентитета или означава везу између два или више примерака. Ред називамо **н-торка**. У једној релацији не смеју постојати две једнаке н-торке. Број атрибута је степен релације, а број н-торки је кардиналност релације. Наш појам релације одговара математичком појму н-нарне релације. У комерцијалним ***DBMS***, уместо "математичких" термина (релација, н-торка, атрибут), чешће се користе непосредни термини (*табела, ред, стубац*) или термини из традиционалних програмских језика (*датотека, запис, поље*).

Кључ К релације Р је подскуп атрибута од Р који има следећа "временски независна" својства:
1. Вредности атрибута из К једнозначно одређују н-торку у Р, односно, у Р не могу постојати две н-торке са истим вредностима атрибута из К.
2. Ако избацимо из К било који атрибут, тада се нарушава својство 1.

Будући да су све н-торке у Р међусобно различите, К увек постоји. Наиме, скуп свих атрибута задовољава својство 1. Избацивањем сувишних атрибута доћи ћемо до подскупа који задовољава и

својство 2. Дешава се да релација има више кандидата за кључ. Тада један он њих проглашавамо **примарним кључем**. Атрибути који чине примарни кључ зову се **примарни атрибути**. Вредност примарног атрибута не сме ни у једној н-торки остати неуписана. Грађу релације кратко описујемо тзв. схемом релације, која се састоји од имена релације и пописа имена атрибута у заградама. Примарни атрибути су подвучени. На пример, за релацију о аутомобилима схема изгледа овако: *ауто* (*регистарски број, произвођач, модел, година*) .

- **Претварање типова ентитета**. Сваки тип ентитета приказује се једном релацијом. Атрибути типа постају атрибути релације. Један примерак ентитета приказан је једном н-торком. Примарни кључ ентитета постаје примарни кључ релације. Додуше, учешће ентитета у везама може захтевати да се у релацију додају још неки атрибути који нису постојали у одговарајућем типу ентитета.
- **Претварање бинарних веза**. Ако тип ентитета E2 има обавезно чланство у (H : 1) вези с типом E1, тада релација за E2 мора укључити примарне атрибуте од E1. Кључ једне релације који је преписан у другу релацију зове се **страни кључ** (у тој другој релацији). Ако тип ентитета E2 има необавезно чланство у (H : 1) вези с типом E1, тада везу можемо приказати на претходни начин, или увођењем нове релације чији атрибути су примарни атрибути од E1 и E2. Посебна релација за приказ везе је препоручива ако релација има своје атрибуте.
- **Претварање инволуираних веза**. Обавља се слично као за бинарне везе.
- **Претварање подтипова**. Подтип се приказује посебном релацијом која садржи примарне атрибуте надређеног типа и атрибуте специфичне за тај подтип.
- **Претварање тернарних веза**. Тернарна веза приказује се посебном релацијом, која садржи примарне атрибуте свих трију типова ентитета заједно с евентуалним атрибутима везе.

Разлике између релацијског, мрежног и хијерархијског модела

Мрежни и хијерархијски модел су прилично слични. Уствари, хијерархијски модел можемо сматрати специјалном врстом мрежног. С друге стране, релацијски модел се по свом приступу битно разликује од остала два. Основна разлика је у начину приказивања веза међу ентитетима и начину коришћења тих веза. У мрежном или хијерархијском моделу могуће је директно приказати везу. Додуше, постоје ограничења на функционалност везе, те на конфигурацију свих веза у схеми. Веза се "материјализује" помоћу поинтера, тј. у једном запису пише адреса другог (везаног) записа. Мрежни и хијерархијски ***DML*** омогућава само једноставне операције с једним записом (упис, промена, брисање, читање), те "маневрисање" кроз схему путем веза (дохват првог записа, који је у задатој вези са заданим записом, дохват идућег записа, ...). Овакав приступ има своје предности и мане. Предност је да је рад с базом у техничком погледу брз и ефикасан. Мана је да корисник може употребити само оне везе које су предвиђене схемом, па су у складу са тим и материјализоване. У релацијском моделу везе су само имплицитно назначене тиме што се исти или сличан атрибут појављује у више релација. Веза није материјализована, већ се динамички успоставља за време рада с подацима, упоређивањем вредности атрибута у н-торкама разних релација. Релацијски ***DML***, осим једноставних операција с једном релацијом, мора омогућити слободно комбиновање података из разних релација. И овај приступ има своје предности и мане. Мана је да се веза сваки пут мора поново успостављати; потребно је претраживање података, а то троши време. Предност је да корисник може успоставити и оне везе које нису биле предвиђене у фази моделирања података. Чак, као критеријум повезивања, осим једнакости за вредност атрибута, могу послужити и разни други (сложенији) критеријуми. То још више повећава флексибилност коришћења базе. Из овога се види да је у релацијском моделу тежиште бачено на динамички аспект (мање чувања, а више манипулисања). Зато употребљивост релацијског ***DBMS*** битно зависи о изражајним могућностима његовог језика за рад с подацима. Пожељно је, такође, да тај језик буде у што већој мери непроцедуралан и разумљив непосредним корисницима. Релацијске језике треба сматрати саставним делом релацијског модела.

Типови база података

С обзиром на сложеност постоје три различита типа база података:

- ***Локалне базе података***. Ово је најједноставнији тип. Налази се на једном рачунару. Измене података се уписују директно у базу. Представници су: ***Paradox, dBase, Access***.
- ***Клијент/сервер базе података***. База се налази на једном рачунару - серверу. Корисници су независни један од другог и свако може приступити бази, а конфликтне ситуације разрешава сервер јер има уграђене механизме за решавање проблема истовременог захтева за приступ бази. Корисници не приступају бази директно него преко програма на локалним рачунарима - клијент програмима који брину о томе да корисници поштују одређена правила и не дозвољавају рад са базом који може да наруши интегритет базе података
- ***Дистрибуиране базе података***. Посебна врста клијент/сервер базеподатака.

Појам датотеке

Организовани скуп података са сличним својствима, над којима се могу вршити различите операције, назива се ***датотека*** (***file***). Датотека може садржати податке различитог типа: бројеве, знакове, низове, датуме, времена, слике, звучне и визуелне записе и друго. Датотеке се чувају на спољашњим носиоцима меморије.

База података се састоји из једне или више датотека које имају структуру која не мора бити иста. Један податак у датотеци се може састојати из више различитих података које не морају бити истог типа. Будући да се овде ради о подацима који су сложенији од до сада коришћених увешћемо нови појам: структурни тип података састављен из коначног, унапред познатог броја компоненти које могу бити различитог типа назива се ***слоговни тип***. Податак слоговног типа назива се ***слог*** (***record***), а његове компоненте називамо ***пољима*** (***field***). Једноставни пример слога је адреса на коверти за писма. Да би писмо стигло ономе коме је упућено потребно је написати име и презиме примаоца, његову улицу и број, поштански број места, место и државу (ако је прималац ван земље пошиљаоца), а понекад и неки додатни податак (пријава на конкурс, за учешће у наградној игри или слично).

Поља у слогу су одређена својим карактеристикама:

- ***назив поља*** (***Field Name***)- низ карактера различите дужине (најчешће 8 или 10) при чему први карактер мора бити слово и у низу не сме бити интерпункцијских и других знакова (осим цртице за подвлачење) ни размака.
- ***тип*** (***Type***) - подаци у оквиру једног поља морају бити истог типа, карактер, нумерички, логички, датум, али и аудио и/или визуелни и други.
- ***величина*** (***Size***) - код карактер или нумеричких података мора се дефинисати максимална дужина податка целобројном вредношћу.
- ***број децимала*** (***Dec***) - нумерички подсци могу имати децимале, мора се унапред дефинисати њихов број целобројном вредношћу која је обавезно мања од величине поља

Када се ради са слоговима морамо схватити да су то сложени подаци и да се према њима морамо понашати битно другачије. Већина функција и операције се не може применити на слог као целину већ се примењује на поједина поља слога. Зато када се обраћамо пољима слога морамо обавезно навести коме то поље припада и коју његову карактеристику мењамо при чему обавезно морамо водити рачуна о типу поља.

Две основне операције карактеристичне за датотеке су:

- упис података у датотеку
- читање података из датотеке

Подаци у датотеци могу бити организовани на различите начине. Најједноставније су ***секвенцијалне датотеке***, датотеке у које се подаци уписују један за другим (секвенцијално). Код оваквих датотека подацима се мора приступати редом од првог до последњег што их чини спорим и непрактичним. Код брисања неког податка постоји два приступа:

- читају се сви подаци и формира се низ, затим се из низа елиминише податак који желимо да обришемо; сада се поново читају сви подаци до податка који се брише, а од њега па надаље уписују се подаци који који треба да остану у датотеци; на крају се датотека скрати за један податак
- формира се нова датотека са истом структуром, затим се у њу преписују сви подаци из прве датотеке до податка који се брише и од податка који следи податак који се брише; прва датотека се брише, а другој се мења име у име прве датотеке.

Датотеке са директним приступом су ***расуте датотеке***, код њих је приближно исто време приступа било ком податку из датотеке. Овакве датотеке се могу чувати само на спољашњим меморијама са директним приступом. Код расутих датотека могуће је и једноставно брисање појединих података.

У даљем раду нећемо користити секвенцијалне датотеке.

Датотеке су једнозначно одређене својом структуром која се дефинише пре првог коришћења и не може се у току извршавања програма мењати што не представља неки велики проблем с обзиром да рад са датотекама подразумева систематичност и прецизну анализу проблема који се

решава, па самим тиме и креирање тачно одређене датотеке, тј. анализом се утврђује тачна структура датотеке, односно, који подаци ће се у њој складиштити и какве су они структуре.

Ако се база података састоји из више датотека онда се веза између њих остварује посредством ***примарног кључа***. Примарни кључ једнозначно одређују сваки слог у датотеци и може га чинити једно или више поља. Примарни кључ дефинише и основно уређење слогова у датотеци. Примарни кључ је једно поље ако је слог њиме једнозначно одређен (на пример ЈМБГ или број пасоша или слично). Ако не постоји такво поље у датотеци, онда примарни кључ мора садржати више поља (име, презиме, име оца, име мајке, датум рођења и место рођења).

Датотека се, осим помоћу примарног кључа, може уређивати и на друге начине, односно, и по другим пољима. Таква поља зову се ***секундарни кључеви***.

Ако једном податку у једној датотеци може одговарати више података у другој онда се поље по којем се остварује веза у тој другој датотеци зове ***страни кључ***.

Приказаћемо креирање датотеке помоћу услужног програма ***DatabaseDesktop*** који је саставни део програмског пакета ***Delphi Enterprise***. Наравно, постоји и прецизно дефинисан поступак креирања датотеке из програма, али се тиме нећемо бавити с обзиром на постављене циљеве овог курса.

Програмирање база у делфију

Фазе у програмирању базе података

- Први корак

Проблем који се решава нашом апликацијом мора бити детаљно анализиран. Морамо знати све опције које треба обухватити програмом. Могуће је и касније додавање захтева и њихова обрада процедурама и функцијама, али је то тежи начин и захтева много више времена за прилагођавање и базе и апликација које их обрађују. Најбоље би било унапред знати све што се од нашег програма очекује и затим систематски одрадити програмерски део посла. Анализа проблема се врши у сарадњи са корисником и другим особама које важе за признате стручњаке из те области. Почиње глобалним рашчлањењем основног проблема на неколико мањих, а затим се сваки од ових даље рашчлањује на још ситније све док се не дође до најједноставнијих послова који се могу решити једном функцијом или процедуром. У овом кораку се креира група података које треба чувати у датотекама и са којима ће радити наша апликација. Подаци се групишу у једну или више датотека по неком критеријуму и одмах се дефинишу везе међу датотекама, кључна поља која ће бити заједничка за парове датотека помоћу којих је могуће извршити једнозначно придруживања групе података једне датотеке групи података друге датотеке. На крају ове фазе требало би да имамо глобалну слику апликације, организоване целине које одговарају главном менију и подменијима будуће апликације.

- Други корак

Сада треба дизајнирати спољашни изглед апликације и формирати систем менија. Апликација мора да буде прилагођена естетским захтевима корисника, а систем менија треба да буде онолико разгранат колико је то претходном анализом утврђено. Такође, треба разрадити или дизајнирати и помоћне екране који ће одговарати појединим захтевима израженим кроз опције главног менија. Паралелно са овим пословима дизајнирања, потребно је креирати иницијалне датотеке које могу бити празне или да садрже тест примере података који ће се користити у фази програмирања. Датотеке могу бити сталне - ако се у њима чувају подаци који чине базу података и привремене - које се користе код креирања различитих приказа података или извештаја. Овај део посла обавићемо помоћу услужног програма ***DatabaseDesktop***. У овом кораку би требало и осмислити екранске и штампане извештаје из захтева корисника. Штампани извештаји могу да се креирају директно из програма, али је много ефикасније направити шаблоне помоћу неког услужног програма и позвати их касније на погодном месту у апликацији. Ми ћемо извештаје креирати помоћу услужног програма ***Rave Reports*** који је у стандардном пакету верзије 7 делфија, али могуће је користити и неке друге програме који могу да се повежу са делфи апликацијом.

- Трећи корак

Након спољашног дизајнирања апликације и креирања главних и помоћних датотека које ће се користити у њој прелазимо на програмирање. Треба програмски решити захтеве дате у првом кораку. Ово је, вероватно, најзанимљивији део посла јер сада можемо да покажемо сву своју генијалност, искористимо своје програмерско искуство и расположиве алатке програмског језика водећи рачуна о типу рачунара на коме ће се апликација извршавати и способности оператера који ће са њом радити. Ако је први корак коректно одрађен, овде не би требало да буде неких проблема. Све процедуре које треба написати су стандардне и само их треба прилагодити конкретном случају, мада се неки од захтева могу решити и на неколико начина, што нам оставља могућност да будемо маштовити и креативни.

- Четврти корак

Ово је најтежи и најдужи део посла - тестирање. Готово 80% укупног времена потребног за програмирање утроши се на тестирање апликације, а ипак се дешава да нам неке грешке промакну.

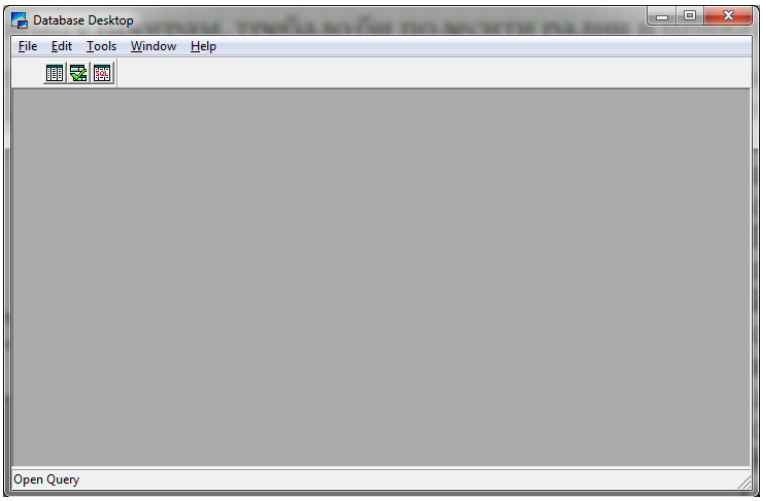
- Пети корак

Формирање документације. Ово је корак који не мора бити самосталан већ се провлачи кроз сваки од претходно поменутих - тако је лакше, али се обично тако не ради. Зато је формирање документације мукотрпан и дуготрајан посао који програмери избегавају кад год и док могу.

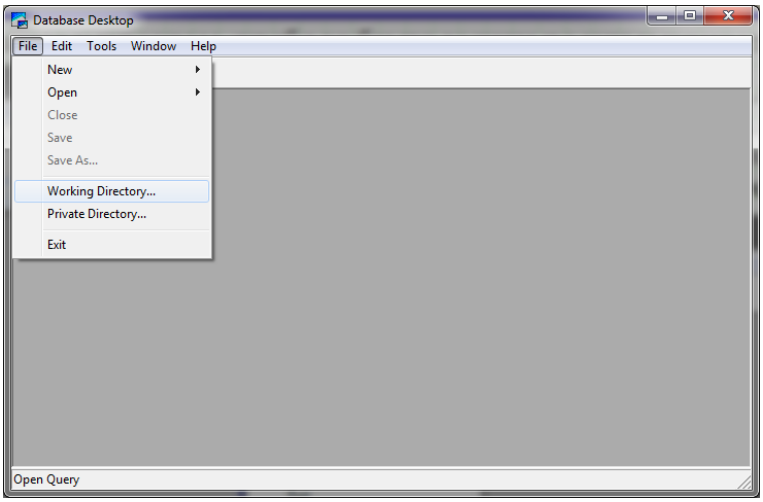
Документација је, међутим, једнако важна као и сам програм у шта се уверио свако ко је бар једном покушао да исправи нечији иоле озбиљнији програм.

Услужни програм *Database Desktop*

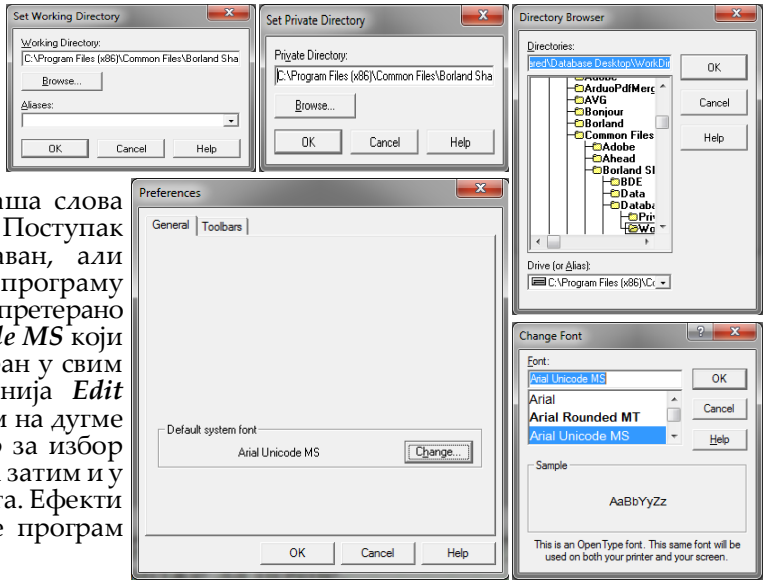
Database Desktop је услужни програм у коме можемо креирати и мењати структуру слога различитих типова датотека, дефинисати различите индексне датотеке, уносити, исправљати и брисати податке и по жељи их сортирати. Такође, у овом програму можемо креурати упите и SQL табеле. Програм је прављен за старије верзије оперативног система и има пуно недостатака, али прилично поједностављује фазу креирања датотека. Насловна страна програма је приказана на слици (ако је из овог програма претходно креирана нека датотека она ће бити отворена у прозору). Ако први пут улазимо у програм, требало би подесити радни и приватни директоријум.



Радни директоријум треба да буде име фолдера у коме ћемо чувати датотеке програма који пишемо, а приватни директоријум треба да је име фолдера коме можемо приступити (ако немамо администраторски налог на рачунару). Поступак дефинисања ових параметара је врло једноставан. Из подменија ***File*** изабере се опција ***Working Directory...*** или ***Private Directory...*** и кликом на дугме ***Browse...*** пронађе се одговарајући фолдер и означи. Кликом на дугме ***OK*** у другом, а затим и у првом прозору потврђује се избор фолдера и поставља параметар.

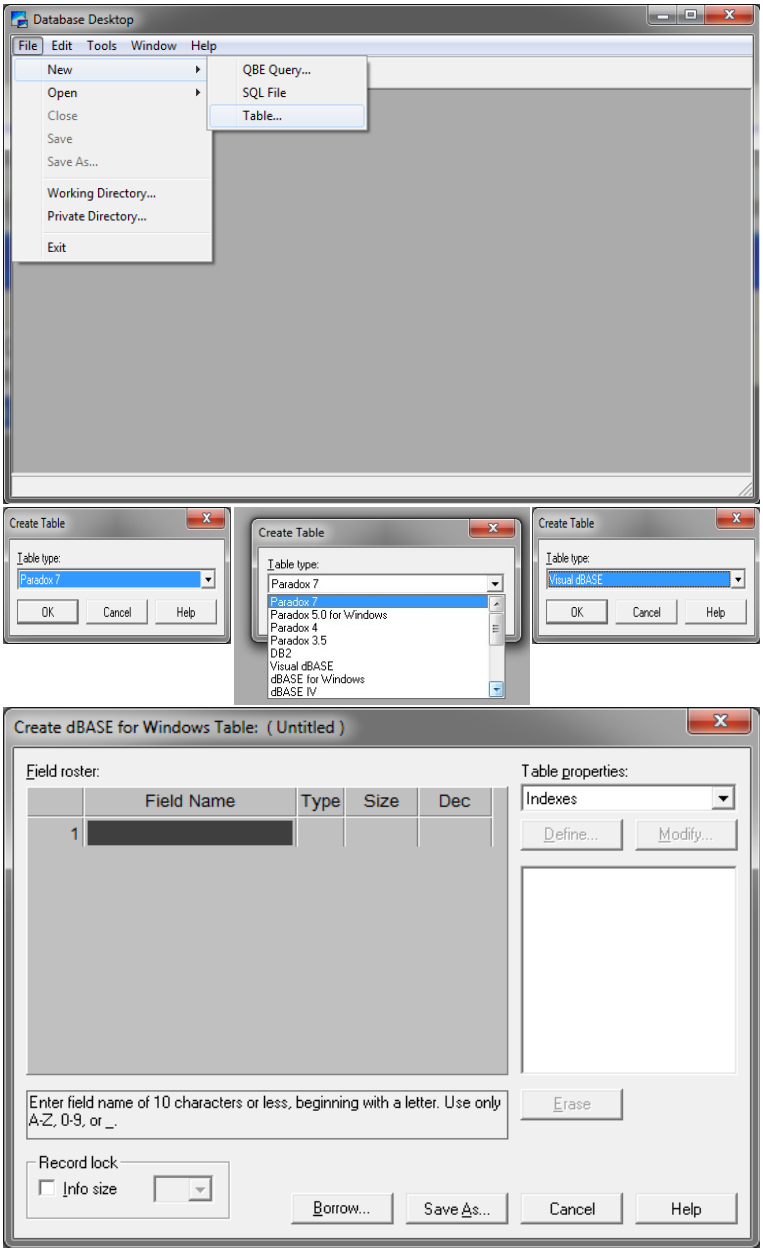


Ако ћемо програм користити за унос или за преглед унетих података морамо водити рачуна о кодној страни подешеној у оперативном систему. Ако користимо нашу ћирилицу или латиницу, да би се текстуални подаци коректно приказали морамо подесити и основни (*default*) фонт (пошто је програм прављен за много старији оперативни систем не могу се сва наша слова приказивати у свим фонтовима). Поступак посављања фонта је врло једноставан, али морамо знати који фонт ће у програму приказивати наша слова. Да не бисмо претерано лутали користићемо фонт ***Arial Unicode MS*** који би требало да је стандардно инсталиран у свим оперативним системима. Из подменија ***Edit*** изабере се опција ***Preferences...*** Кликом на дугме ***Change...*** отвара се дијалошки прозор за избор фонта. Кликом на дугме ***OK*** у другом, а затим и у првом прозору потврђује се избор фонта. Ефекти овог подешавања виде се тек када се програм поново покрене.



Сада можемо да креирамо жељену датотеку.

Креирање датотека у овом програму започињемо отварањем подменија ***File***. Изабраћемо најпре опцију ***New***, па затим ***Table...*** опцију из новоотвореног подменија. Отвориће се следећи дијалошки прозор (*Create Table*) из њега ћемо, левим кликом на стрелицу на крају оквира изабрати одговарајући тип датотеке који креирамо. Ми ћемо користити ***Visual dBase*** тип, с обзиром да ћемо креирати једноставнију локалну базу података. Сада треба кликнути на тастер ***OK*** и креирање датотеке може да почне.



Отвориће се дијалошки прозор у коме ћемо, утврђеним редоследом, уносити имена поља и њихове карактеристике:

- назив (***Field Name***),
- тип (***Type***),
- дужину (***Size***) и
- број децимала (***Dec***) ако се ради о нумеричком податку.

Име поља може имати највише 10 карактера и мора почети словом, сме садржати само слова енглеске абецеде, цифре декадног бројног система и црту за подвлачење. Размаци у имену нису дозвољени. Не могу постојати два поља са истим именом.

Ако ћемо креирати датотеку са структуром која је слична структури неке већ постојеће, онда је корисно употребити тастер ***Borrow*** којим се копира структура те постојеће у структуру датотеке коју креирамо, а затим изменити карактеристике свих поља која се разликују, додати нова или избрисати поља која нам нису потребна. Поље се брише комбинацијом тастера ***Ctrl + Delete***. Треба пажљиво брисати поља јер је поступак неповратан, тј. обрисана дефиниција поља не може се вратити већ се мора поново унети.

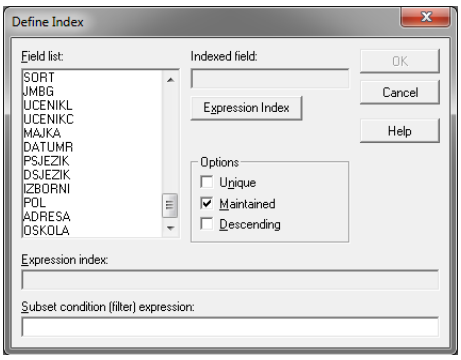
Тип поља може бити: ***Character*** *(словни)*, ***Float*** *(нумерички)*, ***Number*** *(нумерички)*, ***Date*** *(датумски)*, ***Logical*** *(логички)*, ***Memo*** *(текст)*, ***OLE (O**bject **L**inking and **E**mbedding - визуелни, аудио и други типови датотека креирани у другим програмима)* и ***Binary*** *(различити типови датотека у бинарном облику, звук, слика, мултимедија и слично)*. Словни тип мора имати дефинисану дужину, нумерички и дужину и број децимала, а датумски и логички типови имају предефинисану дужину која се не може мењати и зато се не дефинише, а остали типови су посебне датотеке тако да се њихова величина унапред не зна и зато се, такође, не дефинише.

Када подаци буду уписани у датотеку (у каснијем раду или приликом креирања почетних података за тестирање програма) згодно је да се они могу прегледати у неком посебном поретку - азбучни редослед, ако се ради о текстуалним подацима, растући или опадајући поредак, ако се ради о нумеричким подацима. То се може урадити на два начина: сортирањем и индексирање.

- **Сортирање** је поступак којим се преуређују подаци у датотеци тако да се добије изабрани поредак. Ово се може испрограмирати, али мора се схватити да је то поступак који траје неко време (што је већи број података у датотеци то више времена треба за њено сортирање) јер се подаци физички премештају са позиција где су сачувани код уписа на нове. Врло често је случај да нам у једном тренутку треба један редослед, а мало касније другачији, то значи да би се датотека морала поново сортирати кад год имамо потребу за другачијим уређењем података - то је прилично губљење времена, зато се ова метода ретко користи.

- **Индексирање** је битно другачији поступак. Ту се код уписивања података у основну датотеку аутоматски формира индексна датотека која се придружује основној и у којој су подаци сортирани у задатом поретку. Основној датотеци може се придружити произвољан број индексних датотека (индекса). Главна предност индексне датотеке је у томе што је физички мања јер не садржи сва поља из основне датотеке већ само поља по којој се индексира и редне бројеве података из основне датотеке. Индексне датотеке се креирају у фази креирања основне датотеке зато нешто успоравају унос и измену података, али вишеструко убрзавају рад код приказивања података у различитим редоследима зато се овај поступак, готово, обавезно користи.

Објаснићемо поступак креирања индексне датотеке у услужном програму *Database Desktop*.



Кликом на тастер ***Define*** отвориће се дијалошки прозор у коме ћемо дефинисати индексне датотеке.

Постоје прости и сложени индекси.

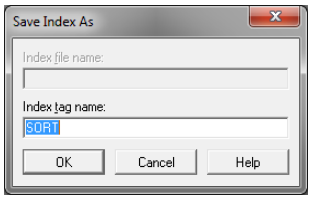
Прости индекси (по једном пољу) се креирају избором двокликом индексног поља у левом оквиру које се аутоматски исписује у оквиру означеном са ***Indexed field***.

Сложени индекси (по више поља) се креирају кликом на тастер ***Expression Index*** и затим уносом одговарајућег израза (називи поља раздвојени ознаком ; или +) у истоимени оквир.

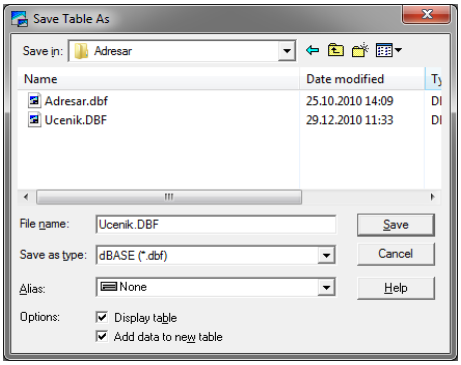
Сада је потребно дефинисати карактеристике индексне датотеке у оквиру ***Options***:

- ***Unique*** - индекс је јединствен, ако има више података који су исти приказује се само један;
- ***Maintained*** - индексирање са чувањем поретка;
- ***Descending*** - подаци се индексирају у опадајућем поретку.

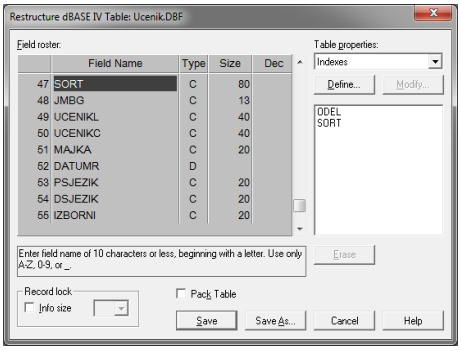
Код сложеног индекса могуће је дефинисати и додатне услове за индексирање у оквиру ***Subset condition (filter) expression:***.



Који год индекс да смо креирали потребно је на крају кликнути на тастер ***OK*** када се отвара са дијалошки прозор у коме имамо могућност да дамо име индексној датотеци и запамтимо је. Код простих индекса име нам је већ понуђено (увек се поклапа са именом индексног поља) и може се потврдити или променити, а код сложених индекса име није понуђено већ се мора уписати. Не могу постојати две индексне датотеке са истим именом.

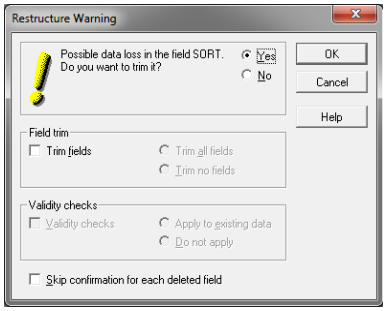


Када смо завршили са креирњем основне датотеке и свих потребних индексних датотека сачуваћемо их кликом на тастер ***Save As***. Отвориће се следећи дијалошки прозор. Датотеке снимамо у изабраном фолдеру, најбоље у фолдеру где ћемо касније сачувати и апликацију, али није обавезно. На овом дијалошком прозору имамо чекбокс ***Display table*** који омогућава да се сачувана датотека одмах отвори ради уписа иницијалних или тест података, ако је чекиран. Чекбокс ***Add data to new table*** користимо ако смо већ постојећу датотеку мењали, па желимо да је сачувамо са новим именом и да у њој остану постојећи подаци. Кликом на тастер ***Save*** завршава се поступак креирања датотеке. Овај поступак треба поновити потребан број пута ради креирања свих основних датотека које ће се користити у апликацији.



Структуру већ креиране датотеке можемо мењати. Поступак измене је следећи: из ***File*** менија изабраћемо опцију ***Open***, па ***Table...*** изаберемо датотеку чију структуру мењамо и кликнемо на тастер ***Restructure*** (четврти тастер у линији са алаткама која се појавила након отварања датотеке). Отвара се дијалошки прозор сличан оном за креирање датотеке. Можемо мењати име или неку другу карактеристику поља, брисати или додавати поља. Једино, код измене типа и величине поља треба водити рачуна о подацима који су уписани у датотеку (ако је поље било словно дужине 80, промена типа поља у нумерички довешће до губитка података из овог поља, а смањење дужине довешће до исецања унетих података на нову дужину). Овде, такође, можемо мењати, брисати и додавати индексне датотеке. Поступак измене је сличан описаном креирању индексних датотека (изабере се индексна датотека кликом мишем на њено име, а затим се кликне на тастер ***Modify***). Брисање индексне датотеке омогућава тастер ***Erase*** (изабере се индексна датотека кликом мишем на њено име, а затим се кликне на тастер).

Ако смо брисали податке из датотеке, то није коначно брисање података, већ само означавање слогова за брисање. Постоји метода за враћање обрисаних поља (***ReCall***) у делфију. А трајно (неповратно) брисање слогова из датотеке омогућава чекбокс ***Pack Table*** на овом дијалошком прозору. ***Запамтимо***, брисање слогова из датотеке се не може поништити. Могу се поништити само измене у структури датотеке или индексних датотека и то само пре чувања измењене структуре.



На крају је потребно кликнути на тастер ***Save*** да би смо сачували измене у датотеци, ***Save As...*** да бисмо измене сачували у датотеци са другим именом или на тастер ***Cancel*** да бисмо одустали од измена у датотеци. Приликом чувања нове структуре већ постојеће датотеке, ако је било измена карактеристика поља, програм ће тражити потврду измене сваког поља пре него се датотека сачува са измењеном структуром. Ако желимо да избегнемо ову потврду код свих поља чија је дефиниција мењана или су брисана потребно је чекирати опције ***Trim fields*** и ***Skip confirmation for each deleted field***.

Поменућемо сада и остале тастере у линији са алаткама.

Прва три тастера су ***Cut, Copy, Paste*** и користе се за копирање података у пољима датотеке.

Тастери са стрелицама (навигатор) се користе за кретање по датотеци (*први у датотеци, први на екрану, претходни, наредни, последњи на екрану, последњи у датотеци*).

Последња два тастера су: ***Field View*** и ***Edit Data***. Први користимо да укључимо курсор унутар селектованог поља, а други да дозволимо измену податка у пољима датотеке.

Технологија BDE

Да би омогућио приступ локалним и клијент/сервер базама података делфи користи технологију ***Borland DataBase Engine*** (***BDE***). BDE је скуп услужних програма који омогућују приступ разним базама података.

Апликација ↔ ***BDE*** ↔ *База података*

Код инсталације нашег програма на неки кориснички рачунар обавезно морамо да инсталирамо и услужни пакет BDE. Делфи компајлер не мора бити инсталиран да би програм радио на корисничком рачунару тако да нема потребе за инсталацијом делфија на било којем корисничком рачунару.

Програмски пакет делфи се испоручује у различитим верзијама: стандардној, професионалној и клијент/сервер верзији. У зависности од инсталиране верзије, биће подржан већи или мањи број различитих база података.

Формати база података различитих пеобог тога се BDE испоручује са скупом драјвера који омогућавају програмима да комуницирају с различитим типовима података. Ови драјвери преводе наредбе базама података у команде специфичне за одређени тип података. Овакав систем рада дозвољава корисничким програмима да приступају базама података без обавезе да знају на који начин те базе функционишу. Све верзије делфија имају драјвер ***Standard***.

За конкретан приступ бази података BDE користи алиас (***alias***). BDE алиас представља скуп параметара који описују начин повезивања с базом података. Алиас указује BDE-у који драјвер да употреби и где се на диску налазе датотеке с базом података. Приликом инсталирања програмски пакет Делфи је инсталирао неке алиасе, као и неке базе података. Ми ћемо у раду користити инсталирани алиас ***dBASE Files***, али је једноставно направити и неки други, ако то желимо, коришћењем програма ***BDE Administrator*** који је инсталиран приликом инсталације делфија.

Највећи број компоненти за рад са базама података налазе се на листићима ***BDE, Data Access*** и ***Data Control***. Оне могу бити визуелне и невизуелне, тј. неке од њих могу да се поставе на форму и да утичу на њен изглед, а друге се постављају на форму, али се у фази извршавања програма не виде, само утичу на рад. Нешто касније ћемо објаснити неке од њих.

Невизуелне и визуелне компоненте не могу директно комуницирати. За комуникацију невизуелних и визуелних компоненти користи се још једна невизуелна компонента ***DataSource*** чију употребу ћемо, такође, објаснити касније.

Визуелне компоненте ↔ ***DataSource*** ↔ *Невизуелне компоненте* ↔ *База података*

У раду са базама података користићемо појам ***DataSet*** који представља скуп података до којих се може доћи на основу података садржаних у бази. То значи да тај скуп података не мора бити у једној датотеци.

Главни мени и лице апликације

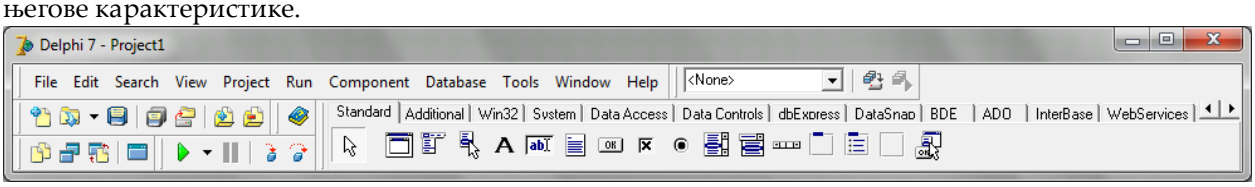
Дизајнирање апликације започећемо креирањем насловне стране и главног менија. Насловна страна је оно што ће корисник најчешће видети када користи нашу апликацију и зато је она врло важна, чак најважнија, неважна ствар апликације.

- Насловна страна

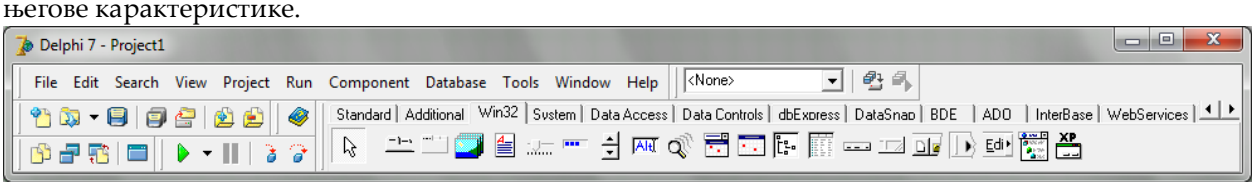
Уласком у делфи апликацију креирана је форма стандардних димензија и карактеристика. Подесићемо неке њене карактеристике:
- Кликнућемо на плусић испред карактеристике ***BorderIcons*** и поставити на ***false*** вредности за ***biSystemMenu, biMinimize, biMaximize*** и ***biHelp***. Овим смо искључили системски мени и тастере за минимизирање и максимизирање прозора апликације. Сада корисник неће моћи да мења димензије прозора апликације нити да затвори прозор на тастер **X** (јер се тастери неће видети када се програм покрене);
- Поставићемо на ***bsSingle*** карактеристику ***BorderStyle*** тако да корисник неће моћи да мења величину прозора хватајући и померајући рубове прозора мишем;
- Карактеристика ***Caption*** као вредност имаће назив апликације;
- Карактеристике ***ClientHeight*** и ***ClientWidth*** дефинишу величину радне површине прозора, поставићемо их на 930 и 1270 (ово значи да резолуција екрана корисника мора бити бар 1280 x 1024 јер су спољне димензије прозора апликације 1278 x 992);
- Карактеристике ***Top*** и ***Left*** поставићемо на 0 или ћемо карактеристику ***Position*** поставити на ***poDesktopCenter*** да би прозор апликације био центриран на екрану када га корисник отвори;
- Карактеристику ***Font*** поставићемо на ***Arial, Regular, 11*** (или неки други фонт који ћемо најчешће користити у апликацији јер ће се ова карактеристика преносити на сваки следећи објекат који поставимо на форму);
- Карактеристику ***Icon*** користимо ако желимо да наша апликација има десктоп икону која се разликује од стандардне (делфи седмица са муњом у овом случају). Уколико нам то није нарочито важно, прескочићемо ову карактеристику;
- Карактеристику ***ShowHint*** поставићемо на ***true*** да бисмо омогућили појављивање различитих порука - хинтова када се мишем стане на форму или неки објекат на њој.

Сада смо завршили почетно подешавање форме. Наравно, не морамо за сваку апликацију подесити карактеристике на овај начин. Димензије и положај форме зависи од потреба корисника и врсте апликације, а шта ћемо дозволити кориснику од наше процене шта корисник може да учину у току извршавања програма. Сада ћемо даодати још неколико компоненти које су важне за изглед и рад апликације. Ово је само један од начина креирања насловне стране. Увек је могуће и другачије дизајнирати је, па све ове активности посматрамо само као један предлог приступа пројектовању апликације.

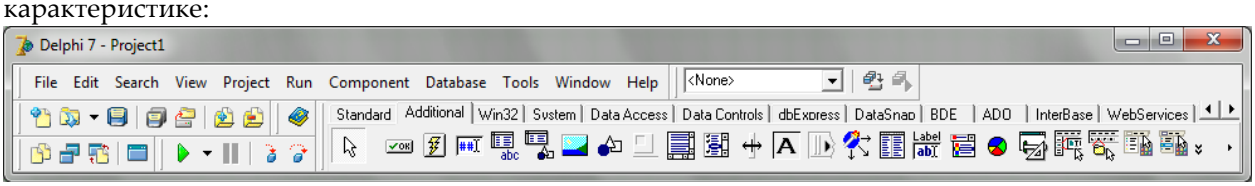
Са листића ***Standard*** поставићемо други објекат - ***MainMenu***, а касније ћемо подешавати његове карактеристике.



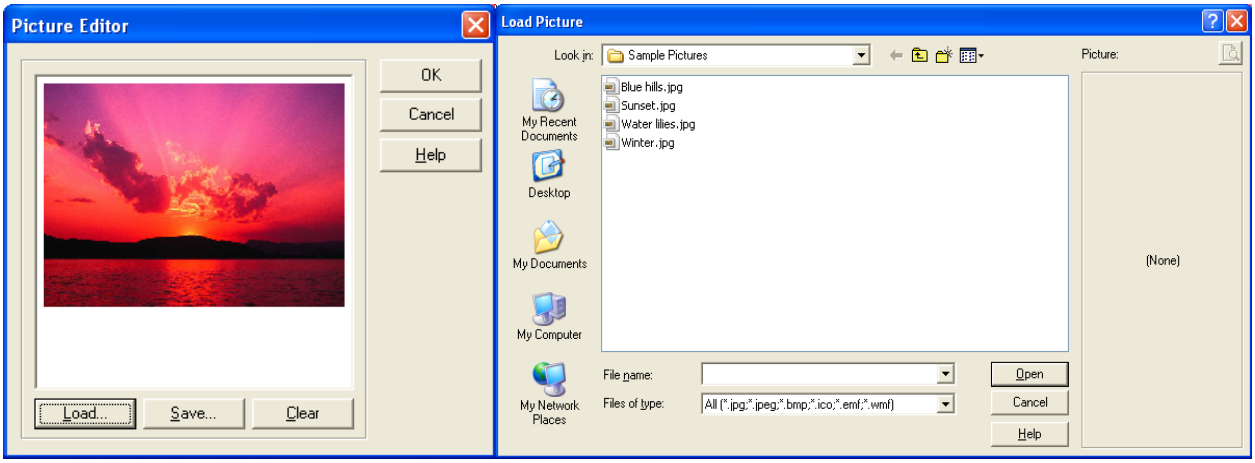
Са листића ***Win32*** поставићемо петнаести објекат - ***StatusBar***, а касније ћемо подешавати његове карактеристике.



Са листића ***Additional*** поставићемо шести објекат - ***Image*** на форму и одмах подесити његове карактеристике:

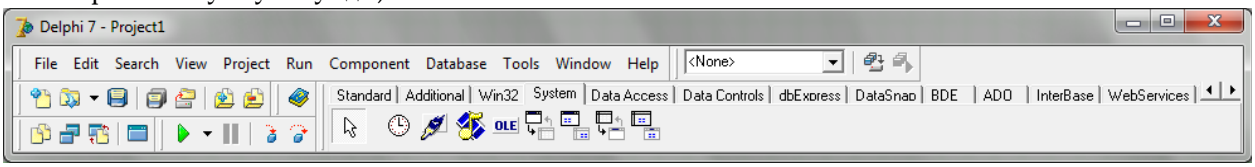


- ***Align*** на ***alClient*** да би слика заузимала целокупни расположиви простор радне површине,
- ***Picture*** на путању којом се стиже до слике која ће бити насловна страна програма (двоклик на поље вредности ове карактеристике отвара едитор, кликом на тастер ***Load*** отвара се ***Open dialog*** у коме треба пронаћи одговарајућу слику и селектовати је, затим клик на ***Open*** и на крају на ***OK***,



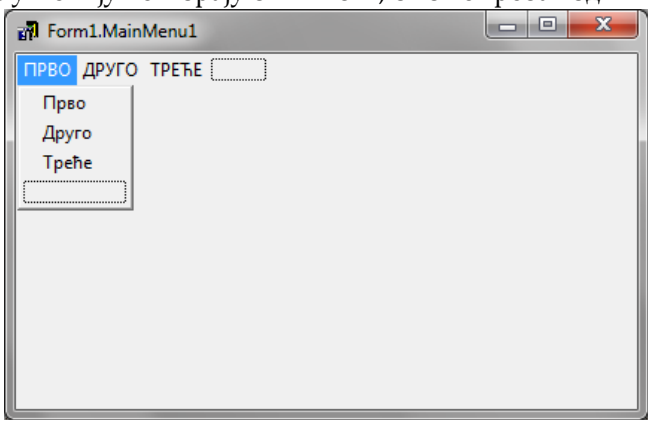
- ***Stretch*** на ***true*** да би слика заузимала читав оквир (наравно, слика се мало деформише, али то се пуно не примећује ако формат слике не одступа пуно од димензија и пропорција оквира за слику, у нашем случају 1270 x 915).

Са листића ***System*** поставићемо први објекат (сличица у облику сата) - ***Timer*** и поставити његову карактеристику ***Interval*** на ***100*** (процедура која ће бити повезана са овом компонентом ће се активирати 10 пута у секунди).



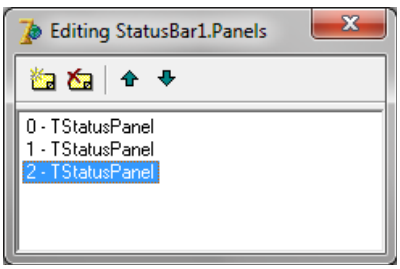
Сада ћемо подешавати карактеристике и осталих постављених компоненти.

Двокликом на објекат ***MainMenu1*** отвориће се едитор у коме ћемо креирати систем менија. Натпис који се појављује на форми и име ставке у менију не морају бити исти, о томе треба водити рачуна приликом писања програма. Натписи могу бити ћирилични са две или више размакнутих речи, а назив мора бити латиничан, без наших слова, размака и знакова интерпункције. Ако користимо ћирилична слова за називе ставки имена ће аутоматски добијати ознаке N1, N2, ..., ако користимо латинична слова размаци и специјални знаци биће у именима ставки игнорисани.



Први ниво менија, натписи који ће се појавити у траци менија биће исписани свим великим словима, а називе подменија ћемо писати почетним великим и осталим малим словима. Ово није обавезно правило, само се аутору ових редова тако чинило занимљивим. Прелазак из једне у другу ставку прве линије менија остварује се стрелицом на лево или десно н тастатури, а прелазак на ставке подменија притиском тастера *Ентер*. Прелазак из било које у било коју ставку менија најједноставније је остварити кликом миша. Ако у оквиру неке од ставки подменија треба креирати још један ниво менија то ћемо урадити десним кликом на ту ставку и избором опције ***Create Submenu***. Креирање новог нивоа менија препоручује се ако је број ставки претходног нивоа превелик (рецимо десетак ставки у неком менију је баш превише и требало би ту опцију менија организовати са подменијима, но, наравно, не изгледа лепо ако смо креирали подмени за групу послова који нису сродни, односно, бесмислен је подмени ради подменија).

За објекат ***StatusBar1*** подесићемо карактеристику ***Height*** на ***20***, а карактеристику ***Font*** поставићемо на ***Arial, BoldItalic, 10*** (или неки други фонт који нам се више свиђа). Затим ћемо, двокликом на вредност карактеристике ***Panels***, отворити едитор у коме ћемо кликом на први тастер (са звездицом) додати три панела (ако смо грешком додали више панела него што желимо кликом на други тастер са иксићем обрисаћемо селектовани панел). Сада ћемо селектовати први од панела и дефинисати његове карактеристике и то: ***Alignment*** са ***taCenter*** и ***Width*** са ***400***. У овом панелу ће се

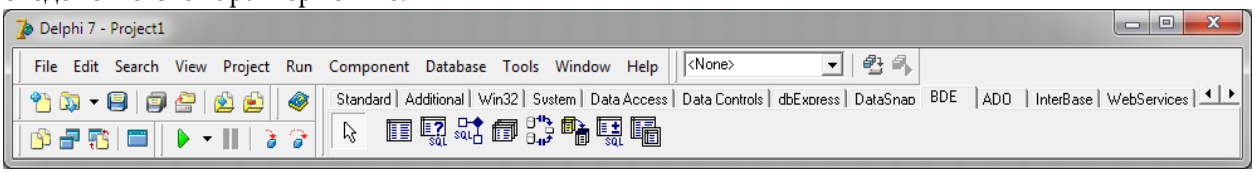


исписивати назив процедуре која се тренутно извршава или ће бити празан када ниједна процедура не буде активна што ће бити програмски дефинисано када се буду писале одговарајуће процедуре. Селектоваћемо други панел и дефинисати његове карактеристике: ***Alignment*** са ***taCenter***, ***Text*** - назив програма, име аутора, година завршетка прве и последње верзије програма и слични подаци које сматрамо важним и ***Width*** са ***570***. Селектоваћемо трећи панел и дефинисати његове карактеристике: ***Alignment*** са ***taCenter*** и ***Width*** са ***300***. У овом панелу ће се исписивати датум и време када будемо опремили одговарајућом процедуром објекат ***Timer1***.

Објекти за рад са датотеком и пољима датотеке

Пре него што почнемо да пишемо процедуре наше апликације, описаћемо објекте које ћемо користити у програму, а који су специфични за рад са датотекама и подацима у пољима датотеке.

Са листића ***BDE*** (***B****orland* ***D****atabase* ***E****ngine*) користићемо први објекат ***Table***. Објекат се користи за повезивање датотеке на диску са нашим програмом у делфију. За овај објекат подешаваћемо следеће његове карактеристике:



TableName - име датотеке на диску са путањом до ње. Датотека би требало да је у истом фолдеру као и делфи програм. Програм мора бити претходно снимљен, а затим учитан из делфија и тада, кликом на вредност ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку. Ако датотеке из било којег разлога нема на листи њено име и путања до ње се може уписти помоћу тастатуре.

Active - отвара датотеку за употребу ако је постављено на ***true***, у противном се датотека и подаци у њој не могу користити.

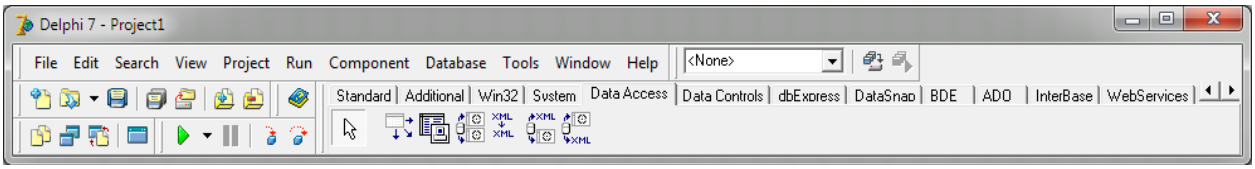
IndexName - дефинише име индексне датотеке која ће бити у употреби (кликом на вредност ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, а ако датотеке нема на листи онда нисмо дефинисали ни једну индексну датотеку).

Filter - дефинише услов који подаци из датотеке морају испуњавати да би се могли приказивати (филтрира датотеку). Вредност карактеристике је логички израз.

Filtered - дозвољава, ако је постављено на ***true*** или забрањује филтрирање датотеке. Наиме, када се услов у претходној карактеристици постави датотека се не филтрира, већ се само даје могућност за то, а тек када се овом карактеристиком потврди датотека је филтрирана.

Name - дефинише име датотеке којим ћемо јој се обраћати у програму, предефинисана вредност је ***Table1***, али се по жељи може променити (најчешће се као име користи стварно име датотеке без наставка).

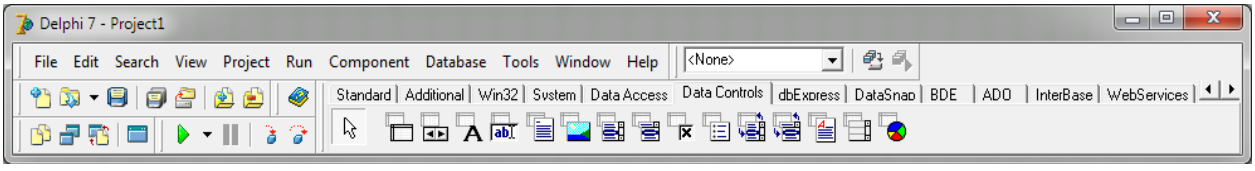
Са листића ***Data Access*** користићемо први објекат ***DataSource***. Објекат користимо да омогућимо програму приступ подацима из датотеке. За овај објекат подешаваћемо само две карактеристике:



DataSet - повезује овај објекат са одговарајућим објектом ***Table*** (кликом на вредност ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку)

Name - дефинише име којим ћемо се обраћати овој компоненти у програму, предефинисана вредност је ***DataSource1***, али се по жељи може променити (најчешће се као име користи стварно име датотеке без наставка продужено за број 0 или 1).

Са листића ***Data Controls*** користићемо готово све објекте, па ћемо зато их укратко описати.



- **dbGrid**

Први објекат, користимо га када желимо да део или све податке из датотеке прикажемо у облику табеле. Грид дозвољава измену и додавање нових података, али се у те сврхе врло ретко користи. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

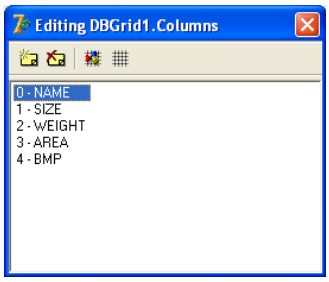
DataSource - дефинише из које датотеке су подаци који ће се приказивати у гриду (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску)

Align - дефинише положај грида у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***

BorderStyle - дефинише да ли ће грид имати оквир или не. Могуће вредности су: ***bsNone, bsSingle***

Color - дефинише боју редова и колона у гриду

Columns - најсложенија карактеристика објекта којом се подешава комплетан изглед грида. Двокликом на поље вредности карактеристике отвара се едитор колона (види слику) у коме можемо додавати и брисати колоне и мењати њихов редослед, али и подешавати карактеристике сваке колоне. Први тастер линије са алаткама нам даје могућност додавања колоне у грид, помоћу другог се брише селектована колона, трећи додаје сва поља из датотеке, а четврти грид формира како је предефинисано у програмском језику. Селектовањем једне или групе колона у едитору добијамо могућност њиховог подешавања у ***Object Inspector***-у помоћу карактеристика:



Alignment - дефинише начин исписа података у колони, могуће вредности су: ***taLeftJustify, taCenter, taRightJustify***
Color - дефинише боју позадине селектованих колона (може се разликовати од боје дефинисане у истоименој карактеристици грида, за све карактеристике важи да ако се дефинишу на гриду и у едитору колона за селектовану колону важи се ова друга)
FieldName - дефинише назив поља које се приказује у тој колони, а користи се само када смо додали колону помоћу тастера на едитору или смо копирали неки постојећи грид, па га прилагођавамо новој ситуацији
Font - дефинише врсту фонта, начин исписа, величину и боју података у селектованим колонама
ReadOnly - дозвољава ако је постављена на *false* или забрањује измену података у колони
Title карактеристике се дефинишу кликом на плусић испред наслова и има их четири: ***Alignment*** - дефинише начин исписа заглавља колоне, могуће вредности су: ***taLeftJustify, taCenter, taRightJustify***; ***Caption*** - дефинише текст у заглављу колоне, ако се не постави као заглавље се користи назив поља из датотеке; ***Color*** - дефинише боју позадине заглавља селектованих колона; ***Font*** - дефинише врсту фонта, начин исписа, величину и боју слова у заглављу селектованих колона
Visible - омогућава да се колона види ако је постављена на *true*, у противном сакрива колону
Width - дефинише ширину селектованих колона

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из грида

FixedColor - дефинише боју непокретних делова грида (заглавље и почеци редова)

Font - дефинише врсту фонта, начин исписа, величину и боју података у свим колонама

Height - одређује висину грида

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад грида

Left - дефинише растојање грида од левог руба радне површине прозора апликације

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у колонама грида

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке

TitleFont - дефинише врсту фонта, начин исписа, величину и боју слова заглавља свих колона

Top - дефинише растојање грида од горњег руба радне површине прозора апликације

Visible - дозвољава (*true*) или забрањује (*false*) приказивање грида на радној површини

Width - одређује ширину грида

- **dbNavigator**

Други објекат на листићу ***Data Controls***, користимо га за навигацију (кретање) по слоговима у датотеци. Састоји се из десет тастера: ***nbFirst, nbPrior, nbNext, nbLast*** (први податак, претходни, наредни, последњи податак у датотеци), ***nbInsert, nbDelete*** (додај податак, избриши податак), ***nbEdit, nbPost, nbCancel, nbRefresh*** (дозволи измену података, запамти измену, поништи измену, сачувај све промене). Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску)

Align - дефинише положај навигатора у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***

ConfirmDelete - захтева (*true*) или не (*false*) потврду брисања слога из датотеке

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће користити тастере навигатора

Height - одређује висину објекта

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта

Hints - дефинише поруку која ће се исписивати када се миш нађе изнад неког тастера навигатора, искључује, ако се дефинише, поруку из претходне карактеристике.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини

VisibleButtons - укључује (*true*) или искључује (*false*) поједине тастере навигатора (кликом на плусић појављују се тастери у листи, па се појединачно, сваки укључује или искључује)

Width - одређује ширину грида

• dbText

Трећи објекат на листићу ***Data Controls***, користи се за приказивање података у оквиру једног поља слога датотеке. Објекат не даје могућност исправке или додавања података. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску)

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље)

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***

Alignment - дефинише начин исписа података у објекту, могуће вредности су: ***taLeftJustify, taCenter, taRightJustify***

AutoSize - дозвољава (*true*) или забрањује (*false*) аутоматску промену величине објекта ако се промени фонт карактеристика

Color - дефинише боју позадине објекта

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта

Font - дефинише врсту фонта, начин исписа, величину и боју исписа податка у објекту

Height - одређује висину објекта

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта

Left - дефинише растојање објекта од левог руба радне површине прозора апликације

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације

Transparent - искључује (*true*) или укључује (*false*) провидност објекта

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини

Width - одређује ширину објекта

WordWrap - дозвољава (*true*) или забрањује (*false*) пренос текста у нови ред ако не може стати у један ред предвиђене ширине објекта

• dbEdit

Четврти објекат на листићу ***Data Controls***, користи се за приказивање података у оквиру једног поља слога датотеке. Објекат даје могућност исправке или додавања података. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

AutoSize - дозвољава (*true*) или забрањује (*false*) аутоматску промену величине објекта ако се промени фонт карактеристика.

BorderStyle - дефинише да ли ће се оквир видети или не, могуће вредности су: ***bsNone, bsSingle***.

CharCase - одређује да ли ће се у објекат моћи уписивати само мал, само велика или и мала и велика слова, могуће вредности су: ***ecLowerCase, ecNormal, ecUpperCase***.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

MaxLength - дефинише максималан број карактера који се могу уписати у објекат (0 - неограничено или 1 - 255 карактера).

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

- **dbMemo**

Пети објекат на листићу ***Data Controls***, користи се за приказивање, измену и унос вишередног текстуалног података у оквиру једног поља слога датотеке. Објекат даје могућност исправке или додавања података. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

Alignment - дефинише начин исписа података у објекту, могуће вредности су: ***taLeftJustify, taCenter, taRightJustify***.

BorderStyle - дефинише да ли ће мемо имати оквир или не, могуће вредности су: ***bsNone, bsSingle***.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

MaxLength - дефинише максималан број карактера који се могу уписати у објекат (0 - неограничено или унети број карактера).

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ScrollBars - омогућава приказ скрол барова ако је текст обимнији, па не може стати у предвиђени оквир, могуће вредности су: ***ssBoth, ssHorizontal, ssNone, ssVertical***.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

WordWrap - дозвољава (*true*) или забрањује (*false*) пренос текста у нови ред ако не може стати у један ред предвиђене ширине објекта.

- **dbImage**

Шести објекат на листићу ***Data Controls***, користи се за приказивање графичких података (слика) у оквиру једног поља слога датотеке. Објекат не даје могућност исправке или додавања података. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

BorderStyle - дефинише да ли ће мемо имати оквир или не, могуће вредности су: ***bsNone, bsSingle***.

Center - одређује да ли ће слика бити центрирана у оквиру или не, ако је оквир шири од слике.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Stretch - одређује да ли ће слика бити деформисана да би стала у оквир, ако оквир и слика нису одговарајућих димензија и пропорција.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

- **dbListBox**

Седми објекат на листићу ***Data Controls***, користи се за унос податка избором са листе понуђених. Може се изабрати само један податак са листе. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

AutoComplete - даје могућност допуњавања уноса података на основу уписаног почетног слова или групе слова, када упишемо прво слово програм проналази први податак у листи који почиње тим словом и аутоматски га уписује. Избор податка је прецизнији ако се упишу прва два, три, ... слова.

BorderStyle - дефинише да ли ће мемо имати оквир или не, могуће вредности су: ***bsNone, bsSingle***.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

ItemHeight - дефинише висину поља у који се уписују подаци, подешава се само ако желимо да висину аутоматски дефинисану на основу величине фонта променимо.

Items - кликом на поље вредности ове карактеристике отвара се едитор у који треба, у сваком новом реду уписати по један од података од којих се бира један за упис у датотеку.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

• dbComboBox

Осми објекат на листићу ***Data Controls***, користи се за унос података избором са падајуће листе понуђених. Може се изабрати само један податак са листе. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље)

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

AutoComplete - даје могућност допуњавања уноса података на основу уписаног почетног слова или групе слова, када упишемо прво слово програм проналази први податак у листи који почиње тим словом и аутоматски га уписује. Избор података је прецизнији ако се упишу прва два, три, ... слова.

AutoDropDown - ако је постављено на *true* аутоматски се отвара падајућа листа када објекат буде активиран, у противном мора се кликнути на стрелицу на објекту.

CharCase - одређује да ли ће се у објекат моћи уписивати само мал, само велика или и мала и велика слова, могуће вредности су: ***ecLowerCase, ecNormal, ecUpperCase***.

Color - дефинише боју позадине објекта.

DropDownCount - одређује број редова падајуће листе који ће се видети. Редови који се не могу видети добијају се померањем клизача у скрол бару објекта или коришћењем стрелица горе и доле на тастатури.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

ItemHeight - дефинише висину поља у који се уписују подаци, подешава се само ако желимо да висину аутоматски дефинисану на основу величине фонта променимо.

Items - кликом на поље вредности ове карактеристике отвара се едитор у који треба, у сваком новом реду уписати по један од података од којих се бира један за упис у датотеку.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Sorted - ако се постави на *true* аутоматски уређује падајућу листу у растућем поретку (абецедно) или ако се постави на *false* приказивање листу онако како је формирана.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

• dbCheckBox

Девети објекат на листићу ***Data Controls***, користи се за унос податка потврђивањем кликом на квадратић испред одговарајућег текста. У датотеку може да се унесе логичка вредност *true* или *false* или предефинисана вредност која одговара овим логичким вредностима. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље)

Alignment - дефинише положај квадратића за потврду и описа значења објекта, могуће вредности су: ***taLeftJustify, taRightJustify***.

Caption - дефинише текст који се исписује поред квадратића за потврду.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

ValueChecked - дефинише шта ће се уписати у датотеку ако је квадратић чекиран.

ValueUnchecked - дефинише шта ће се уписати у датотеку ако квадратић није чекиран.

Width - одређује ширину објекта.

WordWrap - дозвољава (*true*) или забрањује (*false*) пренос текста у нови ред ако не може стати у један ред предвиђене ширине објекта.

• dbRadioGroup

Десети објекат на листићу ***Data Controls***, користи се за унос података потврђивањем на листи понуђених. Подаци су означени кружићима. Може се изабрати само један податак са листе кликом на одговарајући кружић у који се исцртава тачка као ознака да је нешто селектовано. У датотеку се може унети редни број изабраног податка или предефинисана вредност која одговара том редном броју. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље)

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

Caption - дефинише текст који се исписује на оквиру изнад понуђене листе.

Color - дефинише боју позадине објекта.

Columns - дефинише број колона у којима ће се исписивати листа.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Items - кликом на поље вредности ове карактеристике отвара се едитор у који треба, у сваком новом реду уписати по један од података од којих се бира један за упис у датотеку.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Values - дефинише шта ће се уписати у датотеку када се неки кружић чекира.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

• dbLookupListBox

Једанаести објекат на листићу ***Data Controls***, користи се за унос података избором са листе која се формира из неке друге датотеке. Може се изабрати само један податак са листе. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од важнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

ListSource - повезује објекат са датотеком из које се формира листа (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

ListField - одређује која поља из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље, а затим се додају имена осталих поља раздвојена тачка-зарезом).

KeyField - одређује које поље из слога лист датотеке ће се преносити у главну датотеку (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

BorderStyle - дефинише да ли ће мемо имати оквир или не, могуће вредности су: ***bsNone, bsSingle***.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

RowCount - одређује број редова који ће се приказивати у листи.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

• dbLookupComboBox

Дванаести објекат на листићу ***Data Controls***, користи се за унос података избором са падајуће листе која се формира из неке друге датотеке. Може се изабрати само један податак са листе. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље)

ListSource - повезује објекат са датотеком из које се формира листа (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

ListField - одређује која поља из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље, а затим се додају имена осталих поља раздвојена тачка-зарезом).

KeyField - одређује које поље из слога лист датотеке ће се преносити у главну датотеку (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Color - дефинише боју позадине објекта.

DropDownAlign - одређује хоризонтално уређење текста у падајућој листи, могуће вредности су: ***daCenter, daLeft, daRightJustify***.

DropDownCount - одређује број редова падајуће листе који ће се видети. Редови који се не могу видети добијају се померањем клизача у скрол бару објекта или коришћењем стрелица горе и доле на тастатури.

DropDownWidth - одређује ширину падајуће листе, ако се не унесе подразумева се да је ширина листе једнака са ширином објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

• dbRichEdit

Тринаести објекат на листићу ***Data Controls***, користи се за користи се за приказивање, измену и унос вишередног текстуалног података у оквиру једног поља слога датотеке. Разлика, у односу на одговарајући мемо објекат је у додатним могућностима форматизације изгледа текста. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

DataField - одређује које поље из слога датотеке ће се приказивати у објекту (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељено поље).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

Alignment - дефинише начин исписа података у објекту, могуће вредности су: ***taLeftJustify, taCenter, taRightJustify***.

BorderStyle - дефинише да ли ће мемо имати оквир или не, могуће вредности су: ***bsNone, bsSingle***.

Color - дефинише боју позадине објекта.

Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.

Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.

Height - одређује висину објекта.

Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.

Left - дефинише растојање објекта од левог руба радне површине прозора апликације.

MaxLength - дефинише максималан број карактера који се могу уписати у објекат (0 - неограничено или унети број карактера).

ReadOnly - забрањује (*true*) или дозвољава (*false*) исправку података у објекту.

ScrollBars - омогућава приказ скрол барова ако је текст обимнији, па не може стати у предвиђени оквир, могуће вредности су: ***ssBoth, ssHorizontal, ssNone, ssVertical***.

ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.

Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.

Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.

Width - одређује ширину објекта.

WordWrap - дозвољава (*true*) или забрањује (*false*) пренос текста у нови ред ако не може стати у један ред предвиђене ширине објекта.

• dbCtrlGrid

Четрнаести објекат на листићу ***Data Controls***, користи се за приказ, измену и унос податка у облику табеле специјалног облика. Објекат се састоји из блокова који су распоређени у произвољном броју редова и колона. На блок треба поставити неки од претходно објашњених објеката и одговарајуће их повезати. У току извршења програма у сваком блоку ће се приказивати различити слогови из датотеке. Најчешће коришћене карактеристике овог објекта ћемо објаснити почевши од најважнијих, а затим абецедно.

DataSource - повезује објекат са датотеком (кликом на поље вредности ове карактеристике отвара се падајућа листа из које треба изабрати жељену датотеку, тј. објекат ***DataSource*** који представља везу са стварном датотеком на диску).

Align - дефинише положај објекта у односу на рубове радне површине прозора апликације. Могуће вредности су: ***alBottom, alClient, alCustom, alLeft, alNone, alRight, alTop***.

AllowDelete - дозвољава (*true*) или забрањује (*false*) брисање података.

AllowInsert - дозвољава (*true*) или забрањује (*false*) додавање података.

ColCount - дефинише број колона у објекту.

Color - дефинише боју позадине објекта.
Enabled - дефинише да ли јесте (*true*) или није (*false*) могуће селектовати податке из објекта.
Font - дефинише врсту фонта, начин исписа, величину и боју исписа података у објекту.
Height - одређује висину објекта.
Hint - дефинише поруку која ће се исписивати када се миш нађе изнад објекта.
Left - дефинише растојање објекта од левог руба радне површине прозора апликације.
Orientation - дефинише начин приказивања слогова,могуће вредности су: ***goHorizontal, goVertical***.
PanelHeight - одређује висину блокова.
PanelWidth - одређујеширину блокова.
RowCount - дефинише број редова у објекту.
SelectedColor - дефинише боју позадине блока са слогом који је тренутно активан у датотеци.
ShowFocus - дозвољава (*true*) или забрањује (*false*) приказивање оквира око блока са слогом који је тренутно активан у датотеци.
ShowHint - дозвољава (*true*) или забрањује (*false*) приказивање хинт поруке.
Top - дефинише растојање објекта од горњег руба радне површине прозора апликације.
Visible - дозвољава (*true*) или забрањује (*false*) приказивање објекта на радној површини.
Width - одређује ширину објекта.

- **dbChart**

Четрнаести објекат на листићу ***Data Controls***, користи се за грахички приказ нумеричких података из датотеке. Начин коришћења и подешавање карактеристика је нешко компликованије него код претходних објеката и зато ћемо то оставити за неку другу прилику (код excel-a је све то урађено уз помоћ чаробњака, овде се све мора дефинисати и испрограмирати).

Сваки од поменутих објеката има још велики број карактеристика које се могу, а не морају подешавати, у зависности од тога колико се желимо играти или дизајнирати изглед појединих маски и екрана у програму.

Једноставне пратеће процедуре

Написаћемо неколико процедура које нису директно везане за датотеку и рад са подацима у њој, али спадају у процедуре које треба да садржи сваки добро организован програм. Наравно, ово је само предлог како се то може урадити, будућим програмерима је остављено на вољу да то ураде другачије, можда боље, можда ефектније.

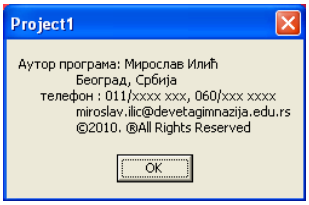
Двокликом на ставку менија КРАЈ РАДА дефинисаћемо проведуру која ће омогућити излазак из програма на следећи начин:

```
procedure TForm1.KRAJ1Click(Sender: TObject);
begin Application.Terminate;
end;
```

Дволкиком на објекат ***Timer1*** дефинисаћемо процедуру која ће приказивати датум и време у последњем панелу статусне линије на следећи начин:

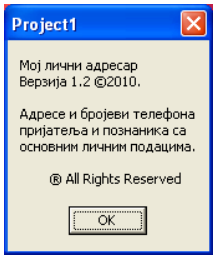
```
procedure TForm1.Timer1Timer(Sender: TObject);
var t,d:string;
begin t:=TimeToStr(time);
      d:=FormatDateTime('dddd, DD.MM.YYYY.',date);
      StatusBar1.Panels[2].Text:=d+' '+t;
end;
```

Двокликом на ставку менија ***О аутору*** која се налази у подмениј ПОМОЋ дефинисаћемо процедуру којом се исписују подаци о аутору програма (како би корисници нашег програма могли да нас контактирају по разним питањима) на следећи начин:




```
procedure TForm1.Oautoru1Click(Sender: TObject);
begin ShowMessage('Аутор програма: Мирослав Илић'+
          chr(13)+'                Београд, Србија'+
          chr(13)+'      телефон : 011/xxxx xxx, 060/xxx xxxx'+
          chr(13)+'                miroslav.ilic@devetagimnazija.edu.rs'+
          chr(13)+'                ©2010. ®All Rights Reserved');
end;
```

Двокликом на ставку менија ***О програму*** која се налази у подменију ПОМОЋ дефинисаћемо процедуру којом се исписују подаци о програма (назив програма, намена програма, верзија програма, година производње и слично) на следећи начин:




```
procedure TForm1.Oprogramu1Click(Sender: TObject);
begin ShowMessage('Мој лични адресар'+
          chr(13)+'Верзија 1.2 ©2010.'+chr(13)+
          chr(13)+'Адресе и бројеви телефона'+
          chr(13)+'пријатеља и познаника са '+
          chr(13)+'основним личним подацима.'+chr(13)+
          chr(13)+'        ® All Rights Reserved');
end;
```

Двокликом на ставку менија ***Упутство*** која се налази у подменију ПОМОЋ дефинисаћемо процедуру којом се исписује корисничко упутство за коришћење програма. Подразумева се да смо пре тога на форму ставили мемо поље и да смо у њега уписали детаљно упутство за коришћење програма (како ми то нисмо урадили наредне две процедуре написаћемо онда када то будемо урадили). Упутство се може написати у посебној текст датотеци која се придружује програму, у мемо пољу, у ***RichEdit*** пољу или некако другачије, а може се применити и метод стандардног виндоус хелпа, али ми ћемо то урадити на овај начин:

```
procedure TForm1.Uputstvo1Click(Sender: TObject);
begin Memo1.Visible:=true; // отвара мемо поље у коме ће се исписивати упутство
      MenuItem1.Enabled:=false; // искључујемо све главне опције менија
      MenuItem2.Enabled:=false; // број опција зависи од дизајна апликације
      MenuItem3.Enabled:=false;
      ...;
end;
```

Излазак из корисничког упутства дефинисаћемо кликом на мемо поље. У ***Object Inspector***-у на листићу ***Events*** пронаћи ћемо догађај ***OnClick*** и двокликом дефинисати следећу процедуру:

```
procedure TForm1.Memo1Click(Sender: TObject);
begin Memo1.Visible:=false; // затвара мемо поље у коме ће се исписивати упутство
      MenuItem1.Enabled:=true; // укључујемо све главне опције менија
      MenuItem2.Enabled:=true; // број опција зависи од дизајна апликације
      MenuItem3.Enabled:=true;
      ...;
end;
```

Излазак из свих процедура решићемо процедуром коју ћемо дефинисати када направимо маску за унос података тако што ћемо додати објекат ***BitBtn*** опремити га одговарајућом сличицом и двокликом на њега дефинисати следећу процедуру:

```
procedure TForm1.Povratak(Sender: TObject);
begin MenuItem1.Enabled:=true; // укључујемо све главне опције менија
      MenuItem2.Enabled:=true; // број опција зависи од дизајна апликације
      MenuItem3.Enabled:=true;
      ...;
      Panel1.Visible:=false; // затварамо све панеле које смо користили
      Panel2.Visible:=false;
      Panel3.Visible:=false;
      ...;
     Table1.Filtered:=false; // искључујемо све постављене филтере
     Table1.First;           // постављамо показивач на први податак у табели
end;
```

Улазак у сваку од процедура омогућићемо процедурама које ћемо дефинисати дволкиком на ставку менија из које је позивамо. Све ће оне личити на овакву једну процедуру:

```
procedure TForm1.NazivProcedure(Sender: TObject);
begin MenuItem1.Enabled:=true; // укључујемо све главне опције менија
      MenuItem2.Enabled:=true; // број опција зависи од дизајна апликације
      MenuItem3.Enabled:=true;
      ...;
      Panel1.Visible:=true;
end;
```

Ово је био уопштени приказ приступа креирању апликације. Сада ћемо се бавити конкретним проблемом и приказати комплетну израду апликације за рад са базом података која се састоји из једне датотеке која се налази на корисничком рачунару (локална база).

Па, да почнемо...