საქართველოს ტექნიკური უნივერსიტეტი

ზ. წვერაიძე, თ. ჭკუასელი,
ე. როჭიკაშვილი

მონაცემთა ბაზის შექმნა Microsoft Access-ში

სახელმძღვანელო
დამტკიცებულია სტუ-ს სარედაქციო-საგამომცემლო
საბჭოს მიერ

თბილისი **2006**

უაკ 681,3.06

განხილულია მონაცემთა ბაზების დაპროექტების საკითხები, მათი შექმნის და გამოყენების საშუალებები, რომლებიც Microsoft Access პროგრამაშია რეალიზებული. მოყვანილია ბაზის ძირითადი ობიექტების – ცხრილების, მოთხოვნების, ფორმების და ანგარიშების როგორც აგების, ისე რედაქტირების და მართვის წესები. განხორციელებულია მათი ილუსტრირება მაგალითების და სავარჯიშოების სახით.

მეთოდური მითითებები განკუთვნილია სტუდენტებისა და სხვადასხვა დარგში მომუშავე სპეციალისტებისათვის, რომლებიც თავიანთ საქმიანობაში იყენებენ Microsoft Access პაკეტში შემავალ კომპიუტერულ ტექნოლოგიებს, კერძოდ, ინტერესს იჩენენ ინფორმაციის მონაცემთა ბაზებში მოთავსებისა და დამუშავების მიმართ.

რედაქტორი: დოც. ც. უთურგაიძე

რეცენზენტები: პროფ. ნ. ჩხაიძე,
პროფ. ი. მიქაძე

ISBN 99940-48-32-5

შესავალი

კაცობრიობა ათეულობით ათასწლეულის მანძილზე აგროვებს ცოდნას და მხოლოდ დღეს, XXI საუკუნის დასაწყისში, დაიწყო ამ უზარმაზარი მოცულობის ინფორმაციის ეფექტურად დამუშავება და დახარისხება. ამის საწყის წერტილად შეიძლება მივიჩნიოთ ის მომენტი, როდესაც კომპიუტერის საფუძველზე გაჩნდა პირველი მონაცემთა ბაზა. მისი ანალოგის შექმნა კომპიუტერის გარეშე წარმოუდგენელია. შესაძლებელია დაიწეროს წიგნი, დაიხაზოს ცხრილი, დაიხატოს სურათი, ანუ კომპიუტერის გარეშე შესრულდეს მრავალი სახის სამუშაო – გარდა მონაცემთა ბაზის შექმნისა. მაგალითად, არსებობდა და არსებობს კარტოთეკა, რომელიც ინფორმაციის შემცველი ბარათებისაგან შედგება და მისი შენახვისა და დახარისხების საქმეს ემსახურება. იგი მონაცემთა ბაზის პროტოტიპს წარმოადგენს, მაგრამ ბარათზე განთავსებული ინფორმაციის დამუშავების – მოპოვების, გამოყენების, შესწორების და ა.შ. პროცესების განხორციელება საკმაოდ რთულია. ამის გამო დღეს კარტოთეკა მონაცემთა ბაზის როლს ვეღარ ასრულებს.

მონაცემთა ბაზაში ორი სრულიად განსხვავებული ობიექტი იგულისხმება: თვით ბაზა, მონაცემების უზარმაზარი მასივები, რომელიც ნედლი მასალაა და პროგრამა – მონაცემთა ბაზის მართვის სისტემა (***მბმს***), რომელიც მთელი ამ ინფორმაციული სივრცის ეფექტურად გამოყენების საშუალებას იძლევა. სწორედ მბმს-ს შესწევს ნებისმიერ მოთხოვნაზე სავსებით კონკრეტული პასუხის გაცემის უნარი და დროის უმცირეს მონაკვეთში უზარმაზარი მასივებიდან *საჭირო* მონაცემებს *საჭირო* ფორმით მოგვაწოდებს.

მბმს-ს სამყარო დიდია და პრაქტიკულად უსაზღვრო, იგი გამოიყენება ყველგან – ინტერნეტით დაწყებული, საშინაო ოფისებით დამთავრებული. დღეს საკუთარი მბმს-ის გარეშე არ ფუნქციონირებს არც ერთი მსხვილი კორპორაცია, სახელმწიფო თუ სასწავლო დაწესებულება.

მსოფლიოში მონაცემთა ბაზების მართვის უამრავი სახის სისტემა არსებობს. მიუხედავად იმისა, რომ ისინი სხვადასხვა ობიექტებთან განსხვავებულად მუშაობენ და მომხმარებელსაც სხვადასხვა ფუნქციებს და საშუალებებს სთავაზობენ, მბმს-ის უმრავლესობა ძირითადი ცნებების ერთიან, ჩამოყალიბებულ კომპლექსს

ეყრდნობა. ეს უფლებას გვაძლევს განვიხილოთ ერთი სისტემა და მისი ცნებები, წესები და მეთოდები მბმს-ის მთელ კლასზე განვავრცოთ. ასეთ სასწავლო ობიექტად ავირჩიეთ მბმს **Microsoft Access,** რომელიც შედის **Microsoft Office** პაკეტში. შესაძლოა, სწორედ ამ პროგრამული სისტემის შესწავლამ დაუდოს საფუძველი თქვენს წარმატებულ კარიერას.

მონაცემთა ბაზის დაპროექტების ძირითადი პრინციპები

სახელმძღვანელო გათვლილია მენეჯერებზე, ეკონომისტებზე, იურისტებსა და სხვა მრავალი პროფილის სპეციალისტებზე. წიგნში წარმოდგენილი მასალების მიხედვით ამ დარგის სპეციალისტებს არ გაუჭირდებათ საკუთარი ბაზის შექმნა. თუმცა, შექმნილი ბაზა რამდენად გამოსადეგი და მოსახერხებელი იქნება, გამოჩნდება მაშინ, როდესაც გაიკვეთა, პრაქტიკაში დაინერგა, თუ დარჩა გამოუყენებელი. ეს ბევრად არის დამოკიდებული იმაზე, თუ რამდენად სწორად იქნება ამოცანა თავიდანვე გააზრებული. არაოპტიმალური გადაწყვეტილებები და შეცდომები, რომლებიც შესაძლოა დაპროექტების ეტაპზე იქნას დაშვებული, შემდგომ, ბაზასთან მუშაობის დროს, მრავალ სირთულეს და გაურკვევლობას შექმნის.

მონაცემთა ბაზის დაპროექტების საწყის ეტაპზე უნდა დადგინდეს შესასვლელი და გამოსასვლელი მონაცემები, რომლებსაც ბაზის დამპროექტებელი აყალიბებს თავისი სამუშაოს ხასიათის და მისი მართვის მიზნების გათვალისწინებით. მაგალითად, თუ იქმნება ბაზა უმაღლეს სასწავლებელში აბიტურიენტთა და სტუდენტთა აღრიცხვისათვის, საჭიროა შესწავლილ იქნას რექტორატსა და დეკანატებში არსებული დოკუმენტების შევსების და მათი მიმოქცევის წესები, აგრეთვე ისიც, თუ როგორ ხდება ლექტორებზე სასწავლო დატვირთვის, ჯგუფებზე აუდიტორიების, სტუდენტებზე საერთო საცხოვრებლის ფონდების განაწილება და ა.შ.

ძირითადი მონაცემების სიის ჩამოყალიბების შემდეგ იქმნება ბაზის სტრუქტურა: დგინდება ე.წ. *საბაზისო* ცხრილების ჩამონათვალი და მათში შესატანი ველების სიები. მონაცემების ველებში განაწილების კრიტერიუმად მიიღება სხვადასხვა ცხრილებში მათი გამეორების გამორიცხვა, რადგან ეს გამოიწვევს კომპიუტერის მეხსიერების არაეფექტურ გამოყენებას. ამ მიზნით ვიყენებთ რელაციურ **(relational)** მეთოდს, რომლის ძირითადი პრინციპი მდგომარეობს იმაში, რომ საჭირო ველების გარდა ცხრილებს ვამატებთ ურთიერთდამაკავშირებელ ველებს – *გასაღებ- ველებს.*

იმ შემთხვევაშიც, როდესაც ბაზა არანაირ მონაცემს არ შეიცავს (*ცარიელი ბაზა*), ის მაინც სრულფასოვანი მონაცემთა ბაზაა. მასში მონაცემები არ არის შეტანილი, მაგრამ არის ინფორმაცია – *ბაზის სტრუქტურა.* ეს უკანასკნელი ბაზაში მონაცემების შეტანის და შენახვის მეთოდებს სრულად განსაზღვრავს. მაგალითად, ავიღოთ მონაცემთა ბაზის უმარტივესი, არაკომპიუტერული მაგალითი – ყოველდღიური საქმიანი ჩანაწერების წიგნი, რომელშიც ყოველი კალენდარული დღისათვის გამოყოფილია ცალკე ფურცელი. იგი განსხვავდება უბის წიგნაკისგანაც და სამუშაო რვეულისგანაც, რადგან მას წინასწარ განსაზღვრული სტრუქტურა აქვს.

მონაცემთა ბაზის მართვის სისტემა შეიცავს სხვადასხვა ობიექტს, მათ შორის ძირითადი ცხრილებია. უმარტივესი მონაცემთა ბაზა შეიძლება შედგებოდეს თუნდაც ერთი ცხრილისაგან. ასეთ დროს უმარტივესი მონაცემთა ბაზის სტრუქტურა ცალსახად ცხრილის სტრუქტურით განისაზღვრება.

როგორც ვიცით, ორგანზომილებიან ცხრილს ქმნიან სვეტები და სტრიქონები. მონაცემთა ბაზის ცხრილის სტრუქტურაში მათ ანალოგებს *ველები* და *ჩანაწერები* წარმოადგენენ. საბაზისო ცხრილის ველების შემადგენლობის (ან მათი თვისებების) ცვლილებით იცვლება მონაცემთა ბაზის სტრუქტურა და, შესაბამისად, მიიღება ახალი მონაცემთა ბაზა.

სწორად დაპროექტებულ მონაცემთა ბაზაში ყოველი ჩანაწერი (მონაცემებით შევსებული სტრიქონი) ერთმნიშვნელოვნად უნდა იყოს იდენტიფიცირებული, ანუ მას მინიჭებული უნდა ჰქონდეს კოდი. ასეთი იდენტიფიკატორების შემცველ ველს (სვეტს) *პირველადი გასაღები* ეწოდება. იდენტიფიკატორის კონკრეტული მნიშვნელობა ველში შემავალ არც ერთ ჩანაწერში არ უნდა მეორდებოდეს. პირველადი გასაღები შეიძლება ერთი ან რამდენიმე ველისაგანაც შედგებოდეს. ზოგადად, ჩანაწერის იდენტიფიკატორად შეიძლება მონაცემთა რაიმე მახასიათებლის გამოყენება. მაგალითად, უმაღლესი სასწავლებლის ბაზაში სტუდენტის კოდად შეგვიძლია სტუდენტის ინდივიდუალური შიფრი ავიღოთ, სავაჭრო დაწესებულების ბაზაში – საქონლის არტიკული და სხვ. თუ ცხრილში ასეთი ველი არ მოიპოვება, მას ხელოვნურად ვქმნით – შევარჩევთ მთვლელის ან სხვა ტიპის ველს. მაგალითად, მთვლელის ტიპის ველში თითოეული ჩანაწერი დაინომრება ავტომატურად და ეს ნომერი იქნება მისი კოდი. ასეთი ტიპის ველში, მისი თვისებებიდან გამომდინარე, მნიშვნელობები არ გამეორდება. ამრიგად, წიგნის ან მაღაზიის დასახელების (40-50 სიმბოლოსგან შემდგარი) იდენტიფიკატორად გამოყენების ნაცვლად ვიყენებთ მის კოდს (5-6 სიმბოლო), რომელსაც ჩავწერთ შესაბამის პირველად გასაღებ-ველში.

თითქმის ყველა თანამედროვე მბმს მონაცემთა რელაციურ ბაზის მოდელს ეფუძნება. სახელწოდება "რელაციური" ნიშნავს იმას, რომ ბაზაში არსებული ყოველი ჩანაწერი შეიცავს ინფორმაციას, დაკავშირებულს ერთ კონკრეტულ ობიექტთან. არ არის გამორიცხული ორი ტიპის მონაცემთან, როგორც ერთ მთლიანთან მუშაობის შესაძლებლობა (მაგალითად, კლიენტი და მის მიერ განხორციელებული შეკვეთა), რადგან ისინი ურთიერთდაკავშირებული მონაცემებია. თუ კლიენტის გვარი, მისამართი და სხვა მონაცემი შეტანილი იქნება ერთდროულად კლიენტის შესახებ ცნობებშიც და ყოველ მის მიერ განხორციელებულ შეკვეთაში, მივიღებთ გამეორებულ ინფორმაციას, რაც, როგორც ითქვა, კომპიუტერის მეხსიერების გამოყენების თვალსაზრისით მიზანშეწონილი არ არის. ამიტომ რელაციურ ბაზაში შეკვეთის ჩანაწერი კლიენტის მხოლოდ კოდს (იდენტიფიკატორს) უნდა შეიცავდეს. იგი თავსდება *გარე გასაღებ-ველში.* მისი მეშვეობით ყოველი შეკვეთის ინფორმაცია დაკავშირდება კლიენტის შესახებ ცნობებთან

(გვარი, სახელი მისამართი და ა.შ.**)** შესაბამისი ცხრილის პირველად გასაღებთან გაერთიანებით. დუბლირების გამორიცხვის პროცესს *ნორმალიზაციის* პროცესი ეწოდება. გამეორებების სრული გამორიცხვაც შესაძლოა, ყოველთვის სასურველი არც იყოს და ამიტომ, ზოგიერთ შემთხვევაში, დაიშვება გამონაკლისი.

ბაზის უსაფრთხოება

მონაცემთა ბაზა – ეს იგივე ფაილია, მაგრამ მასთან მუშაობა გარკვეულწილად განსხვავდება სხვა ტიპის ფაილებთან მუშაობისაგან. როგორც ვიცით, **Windows**-ში ფაილური სტრუქტურის მომსახურებას თავის თავზე ოპერაციული სისტემა იღებს. მონაცემთა ბაზებში უსაფრთხოების თვალსაზრისით განსაკუთრებული მოთხოვნაა დაწესებული და, შესაბამისად, მონაცემების შენახვის მიმართაც სხვაგვარი მიდგომაა განხორციელებული.

ჩვეულებრივ, **Windows**-ში გამოყენებად პროგრამებთან მუშაობის დროს მონაცემების შესანახად გავცემთ შესაბამის ბრძანებას. თუ ფაილი ჯერ არ არსებობს, ვანიჭებთ სახელს და მთლიანად ვენდობით ოპერაციულ სისტემას. თუ ფაილს დამახსოვრების გარეშე დავხურავთ, მთელი შრომა ფაილის შექმნა-რედაქტირებაზე დაიკარგება. ამ მხრივ მონაცემთა ბაზები განსაკუთრებულ სტრუქტურებს მიეკუთვნებიან. ინფორმაცია, რომელსაც ისინი შეიცავენ, ხშირად საზოგადოებრივი ღირებულებისაა. როგორც წესი, ერთსა და იმავე ბაზასთან ათასობით ადამიანი მუშაობს. ამიტომ ბაზის შემცველობის *მთლიანობა* არაფრით არ შეიძლება რომელიმე მომხმარებლის რაიმე კონკრეტულ მოქმედებაზე ან ელექტროქსელის გაუმართაობაზე იყოს დამოკიდებული.

მონაცემთა ბაზების უსაფრთხოების პრობლემის გადასაჭრელად ინფორმაციის შენახვისადმი ორი მიდგომაა განხორციელებული. რიგ ოპერაციებში ყველაფერს ოპერაციული სისტემა განაგებს, მარგამ ზოგიერთი სახის შენახვის მოქმედება ოპერაციული სისტემის გვერდის ავლით ხორციელდება. მაგალითად, მონაცემთა ბაზის სტრუქტურის შეცვლის, ახალი ცხრილების ან სხვა ობიექტების შექმნის ოპერაციები ხდება მონაცემთა ბაზის ფაილის შენახვის დროს. ამის შესახებ მბმს აფრთხილებს მომხმარებელს. ეს *გლობალური ოპერაციებია*. ასეთ ოპერაციებს არასოდეს ახორციელებენ კომერციულ ექსპლუატაციაში მყოფ მონაცემთა ბაზაზე. ისინი სრულდება მხოლოდ ფაილის ასლებზე, რადგან ამ შემთხვევაში გამოთვლითი სისტემის მუშაობის ნებისმიერი შეფერხება არ იქნება სახიფათო.

ამავე დროს, თვით მონაცემების მნიშვნელობების შეცვლის ოპერაციები, რომლებიც არ ცვლიან ბაზის სტრუქტურას, მაქსიმალურად ავტომატიზებულია და შეტყობინების გარეშე სრულდება. თუ ცხრილთან მუშაობისას მონაცემების შედგენილობაში იცვლება რაიმე, ამ ცვლილების შენახვა მოხდება მყისიერად, ავტომატურად.

ჩვეულებრივ, დოკუმენტში რედაქტირების შედეგად მიღებული ცვლილების უარსაყოფად ფაილს უბრალოდ ვხურავთ დამახსოვრების გარეშე და ხელახლა გავხსნით წინამორბედის ასლს. მაგრამ ეს მეთოდი მბმს-ში არ მუშაობს. აქ მონაცემთა ცხრილში შეტანილი ყველა ცვლილება ავტომატურად შეინახება დისკოზე ჩვენგან დამოუკიდებლად, ამიტომ შენახვის გარეშე ფაილის დახურვის მცდელობა არაფერს გვაძლევს.

ჩამოთვლილი მიზეზების გამო სასწავლო ექსპერიმენტების ჩატარება ექსპლუატაციაში მყოფ მონაცემთა ბაზაზე დაუშვებელია. ამისათვის საჭიროა შეიქმნას სპეციალური სასწავლო ბაზა ან რეალური ბაზის **(**სტრუქტურების**)** ასლი.

მონაცემთა ბაზის მართვის სისტემა Microsoft Access

Microsoft Office პროგრამებიდან მონაცემთა ბაზების დამუშავების და მართვის სისტემა **Access** ინტერფეისის სირთულით გამოირჩევა, ამიტომ რიგით მომხმარებელს საწყის ეტაპზე მისი გამოყენება უძნელდება. ეს უფრო პროფესიული პროგრამების რიგს მიეკუთვნება, ვიდრე ფართო მოხმარებისას, როგორებიცაა **Word** ან **Excel.** წინამდებარე სახელმძღვანელოში მკითხველი გაეცნობა **Access**- ის მენიუს პუნქტებს, ცალკეულ ფანჯრებში წარმოდგენილ მართვის ბრძანებებს და შესაძლებლობებს, სისტემაში ჩართულ ფუნქციებს და სხვ. მოყვანილ სავარჯიშოებში წარმოდგენილია ბაზის შექმნის, რედაქტირების და გამოყენების მრავალი მაგალითი, საკმაოდ ბევრი წვრილმანის ჩათვლით. სისტემის ძირითადი შესაძლებლობების გაცნობა ხელს შეუწყობს მკითხველს საკუთარი მონაცემთა ბაზის შექმნაში.

MS Word-ის მომხმარებელი წრფივ, ერთგანზომილებიან სამყაროში მუშაობს, **MS Excel-**ისა – ოპერირებს ორი განზომილებით. მონაცემთა ბაზა კი საშუალებას გვაძლევს, დავინახოთ ინფორმაციული სამყარო სამ განზომილებაში. ფაქტიურად, **Excel-**ის ცხრილი მონაცემთა ბაზის სამუშაო სივრცის მხოლოდ პროექციაა სიბრტყეზე. ამავე დროს, **Excel-**ის ცოდნა ხელს შეუწყობს მომხმარებელს **Access-**ის ათვისებაში, რადგან ამ უკანასკნელის საფუძველია ისევ **Excel-**ის მსგავსი ცხრილები. თანაც, თუ ვიმუშავებთ ერთ ცხრილში, განსხვავება მათ შორის უმნიშვნელოა. მაგრამ თუ საქმე გვაქვს ცხრილების სისტემასთან, გვერდს ვერ ავუვლით **Access** პროგრამას. მაგალითად, **Excel-**ში თავისუფლად შეიძლება საკუთარი მუსიკალური კოლექციის ცხრილის შექმნა, მაგრამ თუ მოვინდომებთ ამ ცხრილისთვის ისეთი მონაცემების მიერთებას, როგორიცაა შემსრულებლების ბიოგრაფია, მათი ფოტოსურათები, დისკოგრაფები და სხვ., მაშინ უნდა გამოვიყენოთ **Access**.

თუ განვიხილავთ მცირე წარმოებას ან ოფისს, აქ საკმაოდ ხშირად გვიხდება ათობით და ასობით ერთმანეთთან დაკავშირებულ ცხრილებს შორის ურთიერთობის დამყარება, რათა ერთ უჯრედში შეტანილი ცვლილება აისახოს შესაბამისი გადაკვეთილი ინფორმაციის მქონე სხვა მრავალ ცხრილში. გარდა ამისა, მაგალითად, კარტოთეკის შექმნის დროს გვიხდება მონაცემების ბარათებზე გამეორება. მსგავსი რამ, ანუ მონაცემთა დუბლირება, სწორად დაპროექტებულ მონაცემთა ბაზაში გამორიცხულია. **Access**-ში ამას ყველაფერს წარმატებით შევასრულებთ მასში არსებული შესაძლებლობების გამოყენებით.

Access-ში მუშაობის დროს ვსარგებლობთ ყველა იმ საშუალებებით, რაც გააჩნია **Microsoft Windows** ოპერაციულ სისტემას: მუშაობა ფანჯრებთან, მენიუსთან, ინსტრუმენტთა პანელებთან, სიებთან და სხვ. აქაც გამოვიყენებთ "ამოჭრა/ კოპირება/ ჩასმის" ხერხებს, რამოდენიმე ობიექტთან ერთდროულად მუშაობის საშუალებებს, **Microsoft Office** პაკეტში შემავალ სხვა დოკუმენტებთან დაკავშირების მეთოდებს და ა.შ.

Microsoft Access-ის არქიტექტურა

Microsoft Access-ში შექმნილი მონაცემთა ბაზა შეინახება ფაილის სახით **.mdb** გაფართოებით. ფაილი რამდენიმე ობიექტისაგან შედგება, ესენია: ცხრილები, მოთხოვნები, ფორმები, ანგარიშები, გვერდები, აგრეთვე მაკროსები და მოდულები. წინამდებარე მეთოდურ მითითებებში და ქვემოთ მოყვანილ სავარჯიშოებში ჩვენ განვიხილავთ ცხრილების, მოთხოვნების, ფორმების და ანგარიშების შექმნის და მათი გამოყენების წესებს. მოკლედ დავახასიათოთ თითოეული მათგანი:

- **ცხრილი** ობიექტია, რომელიც იქმნება მონაცემების შესანახად. ყოველი ცხრილი შეიცავს საგნის ან სუბიექტის შესახებ გარკვეულ ინფორმაციას (მაგალითად, სავაჭრო დაწესებულებაში საქონლის ან კლიენტის მონაცემებს). ცხრილის *ველები* (სვეტები) ინახავენ საგნის (სუბიექტის) სხვადასხვა მახასიათებლებს (მაგალითად, კლიენტის გვარს, მისამართს), ხოლო *ჩანაწერები* (სტრიქონები) შეიცავენ კონკრეტული სუბიექტის შესახებ სრულ ცნობას. ყოველი ცხრილისთვის შესაძლებელია განვსაზღვროთ პირველადი გასაღები-ველი (ერთი ან რამდენიმე ველი), რომელშიც თავსდება უნიკალური მნიშვნელობა – იდენტიფიკატორი (კოდი) და გარე გასაღები-ველი, რომლითაც იგი დაუკავშირდება სათავო ცხრილს. ერთი ან რამდენიმე ველისთვის შეიძლება განისაზღვროს აგრეთვე *ინდექსი* – მონაცემებზე წვდომის დაჩქარების სპეციალური საშუალება.
- **მოთხოვნა** არის ობიექტი, რომელიც მომხმარებელს საშუალებას აძლევს მიიღოს საჭირო ცნობები ერთი ან რამდენიმე ცხრილიდან მის მიერვე ჩამოყალიბებული სტრუქტურით. მოთხოვნაში ისეთი ოპერაციები სრულდება, როგორიცაა ცალკეული მონაცემის ამორჩევა, მონაცემების რაიმე პრინციპით დახარისხება, გაფილტვრა და სხვა. მოთხოვნის მეშვეობით განხორციელდება აგრეთვე ცალკეულ მონაცემების ან მათი ჯგუფის განახლება, გაუქმება, დამატება. ყველა ამ ოპერაციის შესრულება შესაძლებელია უშუალოდ ცხრილში, მაგრამ დროის მოგების და უსაფრთხოების თვალსაზრისით უმჯობესია მათი მოთხოვნაში შესრულება. მოთხოვნის მეშვეობით შესაძლებელია შეიქმნას ახალი ცხრილიც, რომელშიც შეტანილი იქნება ერთი ან რამდენიმე ცხრილიდან ჩანაწერების ამონარჩევის და მათზე სხვადასხვა არითმეტიკული თუ ლოგიკური მოქმედებების შესრულების შედეგები. ამრიგად, მოთხოვნა მონაცემებს საბაზისო ცხრილებიდან იყენებს და ე.წ. *დამაჯამებელ* ცხრილს ქმნის. მიღებული დამაჯამებელი ცხრილის დროებითობის ხაზგასასმელად მას მყის სურათს უწოდებენ. როდესაც ვმუშაობთ მონაცემთა ბაზის ძირითად ცხრილებთან, ფიზიკურად საქმე გვაქვს მყარ დისკთან, ე.ი. ნელა მომუშავე მოწყობილობასთან (ეს დაკავშირებულია მონაცემების შენახვის თავისებურებასთან). ხოლო მოთხოვნის საფუძველზე მიღებული ცხრილი არის ელექტრონული ცხრილი, რომელსაც მყარ დისკზე ანალოგი არ გააჩნია – ეს არის ამორჩეული ველების და ჩანაწერების სახე. ბუნებრივია, ასეთ ცხრილთან მუშაობა გაცილებით სწრაფად ხორციელდება. ამრიგად, მონაცემების ძირითად ცხრილებში, მაგალითად, მოწესრიგებულად შეტანას (ანბანით ან სხვა პრინციპით) აზრი არ აქვს, რადგან ახალი ჩანაწერების შეტანისას ეს პრინციპი მაინც ირღვევა. ამ მიზნით ყოველთვის უნდა გამოვიყენოთ მოთხოვნა.
- **ფორმა** ობიექტია, რომლის დანიშნულებაა მონაცემების გამარტივებული წესით შეტანის უზრუნველყოფა (არა პირდაპირ ცხრილებში). ფორმა აგრეთვე მონაცემების ეკრანზე თვალსაჩინოდ წარმოდგენის, დათვალიერების და მონაცემთა ბაზის ზოგიერთი ოპერაციის მართვის საშუალებაა. შესაძლებელია ფორმის პრინტერზე დაბეჭდვა. ფორმაში მომხმარებელი მხოლოდ იმ ველებს ავსებს, რომლებიც მისთვის არის განკუთვნილი. აქ შეიძლება მართვის სპეციალური ელემენტებიც მოთავსდეს (მთვლელები, გასხნადი ჩამონათვლები, გადამრთველები, ალმები და სხვ.), რომლებიც შეტანის პროცესის ავტომატიზებას მნიშვნელოვნად უწყობს ხელს. მათი მეშვეობით ინფორმაციის შეტანა, ანუ ოპერატორის შრომა, მაქსიმალურად გამარტივებულია და დაშვებული შეცდომებიც მინიმუმამდე არის დაყვანილი.
- **ანგარიში** არის ობიექტი, განკუთვნილი შერჩეული შედეგების დაფორმატების, გამოთვლების ჩატარების და საბეჭდ მოწყობილობაზე გატანისათვის. ბეჭდვის წინ შესაძლებელია მისი წინასწარი გადახედვა ეკრანზე. ანგარიშებში გათვალისწინებულია სპეციალური საშუალებები მონაცემების დასაჯგუფებლად, გაფორმების სპეციალური ელემენტების გამოსატანად (ზედა და ქვედა კოლონტიტულები, გვერდის ნომერი, დამხმარე ინფორმაცია ანგარიშის შექმნის თარიღის შესახებ და სხვ.).
- **გვერდი** მონაცემთა ბაზის სპეციალური ობიექტია – მონაცემებზე წვდომის გვერდი. იგი **HTML** კოდში სრულდება და **Web**-გვერდზე თავსდება. თავისთავად ეს ობიექტი არ წარმოადგენს მონაცემთა ბაზას, მაგრამ შეიცავს კომპონენტებს მასთან კავშირის დასამყარებლად. ამ კომპონენტების მეშვეობით მომხმარებელს შეუძლია იმ ბაზის ჩანაწერების დათვალიერება, რომელიც სერვერზეა დამახსოვრებული.

ამრიგად, მონაცემებზე წვდომის გვერდები ასრულებენ ინტერფეისის როლს კლიენტს, სერვერსა და სერვერზე განთავსებულ მონაცემთა ბაზას შორის.

- **მაკროსი და მოდული –** ობიექტების ეს კატეგორია განკუთვნილია როგორც განმეორებადი ოპერაციების ავტომატიზების, ისე ახალი ფუნქციების შექმნის მიზნით. მაკროსი წარმოადგენს ერთი ან რამდენიმე მოქმედების სტრუქტურირებულ აღწერას და ავტომატურად სრულდება რომელიმე ხდომილების საპასუხოდ. მოდული შეიცავს **Visual Basic** ენაზე დაწერილ პროგრამას. იგი იქმნება რაიმე უზუსტობის აღმოჩენის და თავიდან აცილების მიზნით.

კავშირები ცხრილებს შორის

დაპროექტების ბოლო ეტაპზე ვამყარებთ კავშირებს ცხრილებს შორის, ანუ ვადგენთ *მონაცემთა სქემას.* კავშირის დასამყარებლად სათავო ცხრილთან მისაერთებელ ცხრილს **(**დაქვემდებარებულ ცხრილს**)** დავამატებთ გარე გასაღებ- ველს, რომლითაც იგი შეუერთდება სათავო ცხრილის პირველად გასაღებს. აქ უნდა იყოს მკაცრად დაცული შეერთებული ველების ტიპების თანხვედრა. მაგალითად, თუ პირველადი გასაღები ტექსტის ტიპისაა, გარე გასაღებიც ტექსტის ტიპის უნდა იყოს და ა.შ. გამონაკლისია მთვლელის ტიპი: თუ ამ ტიპისაა პირველადი გასაღები, მაშინ გარე გასაღები რიცხვითი ტიპის უნდა იყოს. მოსახერხებელი იქნება პირველადი და გარე გასაღებ- ველებისათვის ერთნაირი სახელების შერჩევაც.

ცხრილებს შორის კავშირების რამდენიმე სახეობა არსებობს. მათ შორის გამოვყოფთ კავშირებს *ერთი-მრავალთან* და *ერთი-ერთთან.* "ერთის" მხარეზე მყოფ ცხრილს პირველადი გასაღები-ველი აქვს, რომელშიც იდენტიფიკატორებს **(**კოდებს**)** მხოლოდ უნიკალური მნიშვნელობები უნდა ჰქონდეთ. "მრავლის" მხარეზე მყოფ ცხრილს გარე გასაღებ-ველი აქვს, ამ ველში შეიძლება იყოს გამეორებული მნიშვნელობები. მაგალითად, გაყიდვაში შეიძლება ერთი ავტორის რამდენიმე სახელწოდების წიგნი არსებობდეს, მაშინ სხვადასხვა ავტორის სიის შემცველი ცხრილი წიგნების ჩამონათვლის შემცველ ცხრილთან დასაკავშირებლად "ერთი-მრავალთან" სახის კავშირი ექნება. კავშირების სპეციალურ სქემაზე ერთის მხარეს გამოჩნდება 1, ხოლო მეორეზე, ანუ მრავლის მხარეზე, – უსასრულობის ∞ ნიშანი.

განვიხილოთ კავშირის ორივე მხრეს "მრავლის" არსებობა, ასეთ კავშირს *მრავალი-მრავალთან* ეწოდება. ეს ის შემთხვევაა, როდესაც სავაჭრო დაწესებულებაში მრავალი დასახელების საქონელია და მყიდველიც მრავალია: ერთ მყიდველს შეუძლია სხვადასხვა დასახელების საქონელი იყიდოს და, ამავე დროს, ერთი სახელწოდების საქონელი შეიძლება სხვადასხვა მყიდველმა შეიძინოს. **Access** სისტემა ასეთი ცხრილების პირდაპირ დაკავშირებას არ ახორციელებს, უნდა შეიქმნას დამატებითი *გადამკვეთი ცხრილი,* კერძოდ, შენაძენების ამსახველი ცხრილი **(**მაგალითად, ე.წ. ფაქტურა**)**, რომელშიც შეტანილი იქნება კლიენტების და საქონლის მხოლოდ კოდები, ანუ რომელი კოდის საქონელი რომელი კოდის მქონე კლიენტმა შეიძინა. ასეთ ცხრილში სხვა მონაცემების შეტანაც შეიძლება: როდის განხორციელდა ყიდვა, რა ფასად და სხვ. სათავო ცხრილების გადამკვეთ ცხრილებთან დაკავშირება პირველადი გასაღების გადამკვეთი ცხრილის გარე გასაღებთან შეერთებით მოხდება, ანუ განხორციელდება ორი დამოუკიდებელი კავშირი "ერთი მრავალთან".

ზოგიერთ შემთხვევაში ერთ ცხრილს ორ ან მეტ დამოუკიდებელ ცხრილად ყოფენ **(**მაგალითად, სახელებს და გვარებს ათავსებენ ერთ ცხრილში, ხოლო მისამართს, ტელეფონს და სხვა მონაცემებს – მეორეში**)**. ასეთ ცხრილებს ერთნაირი პირველადი გასაღებები უკეთდება და მათ შორის "ერთი-ერთთან" კავშირი მყარდება.

თუ ბაზის სტრუქტურა კარგადაა გააზრებული, ცხრილებს შორის რელაციური კავშირების დადგენა **Access-**ში მარტივია – მთელი საჭირო სამუშაო მიმდინარეობს მონაცემთა სქემის **Relationships** ფანჯარაზე. ფანჯარას ვიძახებთ ბაზის ძირითადი ფანჯრიდან **Tools → Relationships** ან იმავე სახელწოდების ღილაკით. გამოტანილი დამხმარე ფანჯრიდან, რომელშიც დასაკავშირებელი ცხრილების სიაა შეტანილი, ავარჩევთ **(**მოვნიშნავთ**)** შესაერთებელ ცხრილებს, დავაჭერთ ხელს **Add** ღილაკს და დავხურავთ, **Close**. სქემაზე განთავსდება მონიშნული ცხრილების მაკეტები **(**ველების ჩამონათვლით**)**. კავშირი განხორციელდება თაგვის მეშვეობით: სათავო ცხრილიდან პირველადი გასაღები-ველის სახელს გადმოვიტანთ მისაერთებელი ცხრილის გარე გასაღებზე**.**

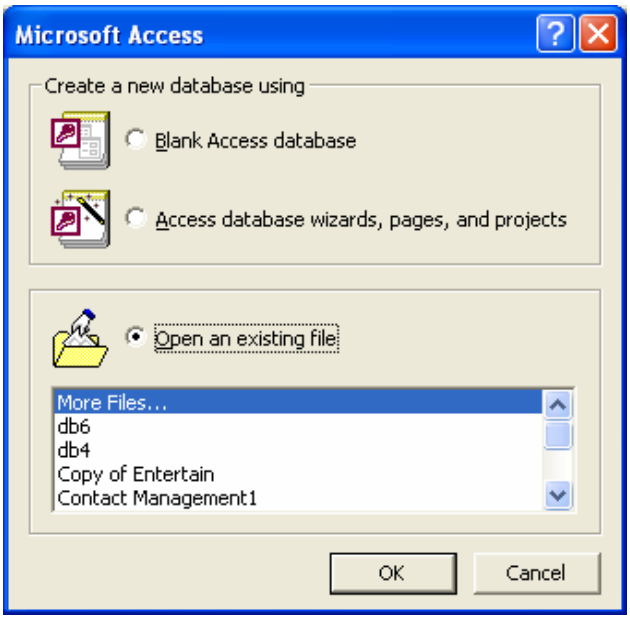
კავშირებს ორი ძირითადი დატვირთვა აქვთ: მონაცემების მთლიანობის დაცვა და ბაზის მომსახურების ავტომატიზება. ვთქვათ, კლიენტების ცხრილიდან ერთი კლიენტის ჩანაწერი გაუქმდა, მაგრამ სხვა ცხრილში, მაგალითად, კონტრაქტების ცხრილში, ამ კლიენტის კოდი ჯერ კიდევ არსებობს **(**არ გაუქმებულა**)**. ამრიგად, ირღვევა მონაცემების მთლიანობა – კონტრაქტების ცხრილში მითითებულია არარსებული კლიენტის კოდი. აქ შესაძლებელია საკითხის შემდეგნაირად განხილვა: ან მთლიანობა საერთოდ არ იქნას ბაზაში დაცული, ან აიკრძალოს სათავო ცხრილებიდან მონაცემების გაუქმება, ან დაშვებულ იქნას მიერთებული ცხრილების ადეკვატური დამუშავება. ამის შესრულება ხელით ძნელია. **Access-**ში გათვალისწინებულია ავტომატიზების საშუალებები: კავშირის ხაზზე დაწკაპუნებით და კონტექსტური მენიუს გამოძახებით ავარჩევთ კავშირის რედაქტირების **Edit Relationships** ფანჯარას. თუ ჩართულია მხოლოდ მონაცემების მთლიანობის უზრუნველყოფის ალამი **Enforce Referential Integrity**, მაშინ მონაცემის გაუქმება სათავო ცხრილში შეუძლებელი ხდება **(**სანამ არსებობს კავშირი**)**. მაგრამ თუ მასთან ერთად ჩართულია ალმები **Enforce Referential Integrity (**მიერთებულ ცხრილებში კასკადური განახლება**)** და **Cascade Delete Related Records (**დაკავშირებული ჩანაწერების კასკადური გაუქმება**),** მაშინ ეს ოპერაციები განხორციელდება ავტომატურად. ამრიგად, რელაციური კავშირების შექმნა, ერთი მხრივ, იცავს ბაზის

მთლიანობას – მონაცემების უნებლიედ გაუქმებისაგან, მეორე მხრივ, ერთ ცხრილში ცვლილების შეტანა იწვევს ავტომატურად მონაცემის ცვლილებას მასთან დაკავშირებულ ცხრილებში.

მრავალი ცხრილის საფუძველზე შექმნილი მოთხოვნა, ანუ ჩანაწერების ერთობლიობა ფორმირდება დაკავშირებული გასაღები ველების მნიშვნელობების თანხვედრით. მაგალითად, გაყიდვის მოთხოვნაში შეძენილი საქონლის სახელის კოდს ვაკავშირებთ საერთო ჩამონათვალიდან საქონლის კოდთან, კლიენტის კოდს – კოდთან მის შესახებ ინფორმაციიდან. შედეგად იქმნება საქონლის და კლიენტის შემცველი ერთი ჩანაწერი. მოთხოვნის ასეთ ტიპს *სიმეტრიული გაერთიანება* ეწოდება. ამ მოთხოვნაში, ბუნებრივია, არ იქნება მითითებული იმ საქონლის სახელი, რომელიც არ იყო გაყიდული. მაგრამ არის შემთხვევა, როდესაც საჭიროა გაყიდვის მოთხოვნაში ნაჩვენები იყოს იმ საქონლის სახელიც, რომელიც არ იყო გაყიდული. ასეთ დროს ვქმნით *გარე გაერთიანებას.* ამ წესით შექმნილ მოთხოვნაში დავინახავთ ყველა ჩანაწერს იმ საქონლის სახელის ჩათვლით, რომელიც არ იყო შეძენილი კლიენტის მიერ. შენაძენთა მოთხოვნაში კოდების ველში შეტანილი იქნება **Null** მნიშვნელობა. გარე გაერთიანების შესაქმნელად ვცვლით გაერთიანების პარამეტრებს, რისთვისაც გავხსნით მოთხოვნას კონსტრუქტორის რეჟიმში და ორჯერ დავაწკაპუნებთ ცხრილების მოდელების შემაერთებელ ხაზზე. გაიხსნება დიალოგი **Join Properties,** რომელშიც 3 ღილაკია გამოტანილი. თავდაპირველად ჩართულია პირველი, რაც შეესაბამება სიმეტრიულ გაერთიანებას. თუ ჩავრთავთ მეორე ღილაკს, პირველი ცხრილიდან ყველა ჩანაწერი იქნება გადატანილი მოთხოვნაში, შემაერთებელი ხაზი გადაიქცევა ისრად. თუ მოთხოვნის გამოყენება გვინდა გაუყიდავი საქონლის ამორჩევისათვის, მოვახდენთ მოთხოვნის გაფილტვრას: **Criteria** სტრიქონზე შევიტანთ პირობას **IsNull**.

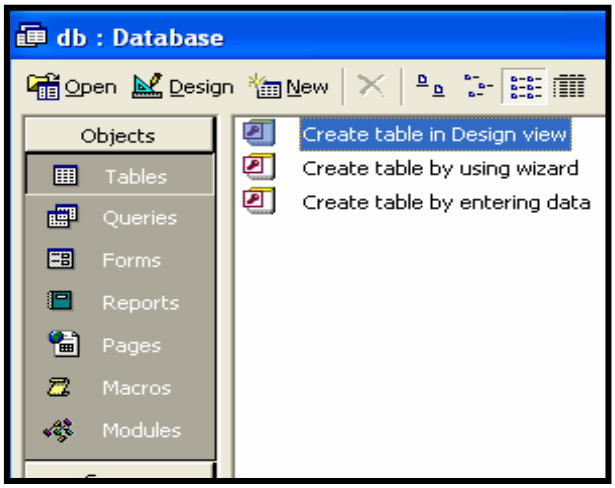
მონაცემთა ბაზის აგება Microsoft Access-ში

Microsoft Access-ის გაშვებისთანავე იხსნება დიალოგ- ფანჯარა (ნახ.1), რომელშიც სისტემა მუშაობის დაწყების 3 ვარიანტს გვთავაზობს: ახალი მონაცემთა ბაზის შექმნა **(Blank Access database)**, ბაზის შექმნა შაბლონების საფუძველზე ოსტატის მეშვეობით **(Access database wizards, pages, and projects)**, არსებული ბაზის გახსნა **(Open an existing file).**



ნახ. 1

თუ პირველ ვარიანტს ავარჩევთ, ფაილს ჩვეულებრივი წესით მივანიჭებთ სახელს და ხელს დავაჭერთ **Create** ღილაკს. გაიხსნება მონაცემთა ბაზის ძირითადი ფანჯარა (ნახ.2). ცარიელი ბაზის, ანუ ბაზის სტრუქტურის აგება უნდა დავიწყოთ სწორედ **Blank Access database** ჩართვით (ნახ1.). ბაზის ძირითად ფანჯარაზე (ნახ.2) ჩართულია ღილაკი **Tables** (ცხრილები). ფანჯრის ზედა პანელზე განლაგებულია 3 ღილაკი: ობიექტის გახსნის **(Open),** მონაცემების შეტანის, რედაქტირების ან განხილვის მიზნით, ობიექტის გახსნის კონსტრუქტორის რეჟიმში **(Design)** და ახალი ობიექტის შექმნის **(New)**. ამავე პანელზე მდებარე მომდევნო ღილაკები სტანდარტულია: ობიექტების გაუქმებისა და მათი ნიშნაკების სხვადასხვა ხედით გამოტანისათვის. ახალი ცხრილის შექმნის მიზნით ავარჩევთ **New** ღილაკს, გამოტანილ დიალოგ-ფანჯარაში (ნახ.3) შემოთავაზებული იქნება ახალი ცხრილის აგების



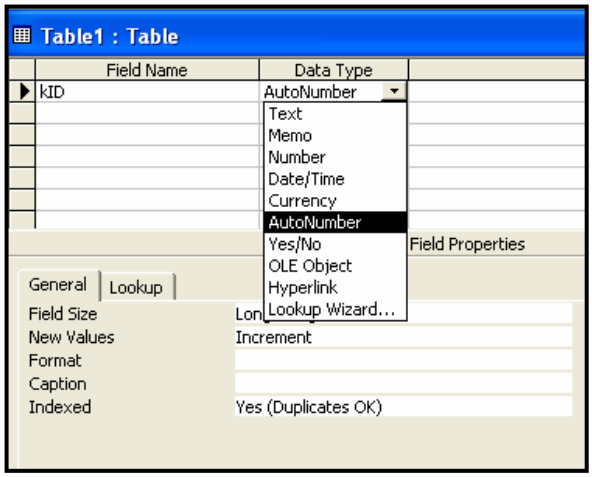
ნახ 2

რამოდენიმე ვარიანტი: ცხრილის შექმნა მონაცემების შეტანის რეჟიმში **(Datasheet View)**, ცხრილის სტრუქტურის შექმნა კონსტრუქტორის რეჟიმში **(Design View)**, ცხრილის შექმნა ოსტატის გამოყენებით **(Table Wizard)**, ცხრილის გადმოტანა სხვა პროგრამიდან, **(Import Table)**, კავშირის დამყარება **(Link Table)** სხვა პროგრამის ცხრილებთან.

ნახ. 3

ცხრილის შექმნა კონსტრუქტორის რეჟიმში

ცხრილის სტრუქტურას ვქმნით კონსტრუქტორის **(Design)** რეჟიმში (ნახ. 4). ფანჯრის ზედა ნაწილის ყოველ სტრიქონზე შევიტანთ ველების სახელებს და განვსაზღვრავთ იქ შესატანი



ნახ. 4

მონაცემების ტიპებს **(Data Type)**. ეს პროცესი ავტომატურად ხორციელდება მონაცემის შეტანის დროს, თუ ცხრილი იქმნება **Create table by entering data** ვარიანტით. კონსტრუქტორის რეჟიმში კი **(Design View)** ტიპებს და შესაბამის თვისებებს ვირჩევთ ჩვენ თვითონ.

ველების თვისებები ყალიბდება ფანჯრის ქვედა ნაწილში **(Fields Properties).** აქ გამოტანილია აქტიური ველის (სადაც კურსორია, ან რომელიც მონიშნულია) მონაცემის ტიპის შესაბამისი თვისებების ჩამონათვალი.

მონაცემთა ტიპები

Microsoft Access-ში შემდეგი ტიპის მონაცემებია განსაზღვრული:

- **Text** – ტექსტური მონაცემთა ტიპი, რომელიც გამოიყენება არაფორმატირებული შეზღუდული სიგრძის ტექსტის შესატანად (მეხსიერებაში იგი იკავებს 255 ბაიტს).
- **Memo** – მემო, მონაცემთა სპეციალური ტიპი, რომელიც გამოიყენება დიდი მოცულობის ტექსტის შესატანად (64 000 ბაიტამდე). ფიზიკურად ტექსტი ველში არ ინახება და თავსდება ბაზის სხვა ადგილზე, ველში კი ინახება მხოლოდ მასზე მიმართვა. ეს გარემოება მომხმარებლისთვის შეუმჩნეველი რჩება.
- **Number** – რიცხვითი მონაცემთა ტიპი, რომელიც გამოიყენება რიცხვების შესანახად (მისი სხვადასხვა ფორმატი არსებობს და მეხსიერებაში ისინი იკავებენ 1, 2, 4, 8 ან 12 ბაიტს).
- **Date/ Time** – თარიღის და დროის ტიპი (8 ბაიტი).
- **Currency** – ფულადი მონაცემთა ტიპი, რომელიც გამოიყენება თანხების შესატანად. თეორიულად ამ ტიპის მონაცემის შესანახად შეიძლებოდა რიცხვითი ტიპის მონაცემის გამოყენება, მაგრამ ფულადი თანხების მეხსიერებაში შენახვის გარკვეული წესების გათვალისწინებით (დამრგვალებასთან დაკავშირებით) მოსახერხებელია მონაცემთა სპეციალური ტიპის შერჩევა (8 ბაიტი).
- **Autonumber** – მთველელი, რიცხვთა სპეციალური ტიპი, რომელიც გამოიყენება უნიკალური (არაგანმეორებადი) ნატურალური რიცხვებისათვის. მათი შეტანა განხორციელდება ავტომატურად, ზრდადობის პრინციპის დაცვით **(Increment)** ან შემთხვევითი რიცხვების განსაზღვრით **(Random)**. იგი განახორციელებს ჩანაწერების დანომვრას. საყურადღებოა, რომ გაუქმებულ ჩანაწერთან ერთად უქმდება მისი ნომერი და მოცემულ ბაზაში იგი აღარ აღდგება (4 ბაიტი).
- **Yes/ No** – ლოგიკური ტიპი, გამოიყენება ლოგიკური ტიპის მონაცემების შესანახად, დაშვებულია მხოლოდ ორი მნიშვნელობა "კი" ან "არა" (1 ბიტი).
- **OLE Object** – მონაცემთა სპეციალური ტიპი **(OLE -Object Link Embedded).** განკუთვნილია ობიექტების სხვა პროგრამებიდან გადმოსატანად ან მათთან დასაკავშირებლად (მაგალითად, ნახატების, დიაგრამების და სხვ.). რეალურად ასეთი ობიექტი ცხრილში არ ინახება ისევე, როგორც **Memo** ტიპის მონაცემი. ცხრილში ინახება მხოლოდ მიმართვა მათზე, რაც ამარტივებს და აჩქარებს ცხრილთან მუშაობის პროცესს (1 გიგაბაიტამდე).
- **Hyperlink** ჰიპერკავშირები – სპეციალური ტიპი, რომელშიც ინახება ინტერნეტის **URL Web** ობიექტების მისამართები.
- **Lookup Wizard…** ოსტატი. ეს არ არის მონაცემის ტიპი. ამ ობიექტით შეიძლება ველში მონაცემების შეტანის ავტომატიზება ისე, რომ მონაცემი იქნას ამორჩეული გახსნილი ჩამონათვლიდან.

ცხრილის ველების თვისებები

ცხრილის ყოველ ველს შეიძლება მივანიჭოთ შერჩეული მონაცემის ტიპის შესაბამისი თვისება, რომელიც მოთავსებულია ფანჯრის ქვედა ნაწილის **General** ჩანართში **(**ნახ. 4**)**. უფრო დაწვრილებით ისინი განიხილება სავარჯიშოებში.

- **Field size –** ველის ზომა, რომელიც შეიძლება იყოს:
  - ტექსტისთვის 0-დან 255 სიმბოლომდე **(**თავიდან შემოთავაზებულია 50**)**;
  - რიცხვისთვის რამდენიმე ვარიანტი არსებობს: **Byte –** ბაიტი **(**რიცხვები 0 დან 255 დიაპაზონში, 1 ბაიტი**)**; **Integer** – მთელი რიცხვი **(**–32768 დან +32767 დიაპაზონში, 2 ბაიტი**)**; **Long Integer -** მთელი ტიპის რიცხვი **(**რიცხვები -2147483648-დან +2147483647 დიაპაზონში, 4 ბაიტი**)**; **Single –** ნამდვილი რიცხვი ათობითი მძიმის შემდეგ 7 ნიშნით **(**რიცხვები -3.410^{38} დან $+3.410^{38}$ დიაპაზონში, 4 ბაიტი**)**; **Double** – ნამდვილი რიცხვი ორმაგი, ათობითი მძიმის შემდეგ 15 ნიშნამდე სიზუსტით **(**რიცხვები -1.797^{308} დან $+1.797^{308}$ დიაპაზონში, 8 ბაიტი**)**; **Replication ID –** რეპლიკაციის კოდი გამოიყენება მხოლოდ დისპეტჩერის მიერ მართვად მონაცემთა ბაზებში.
- **Format –** ველის ფორმატი:
  - ტექსტისთვის შეგვიძლია შევარჩიოთ სპეციალური ფორმატი, მაგალითად, თუ ჩავწერთ **@@-@@@@,** მაშინ 6 სიმბოლოიანი ტექსტი გამოისახება 2 სიმბოლოს შემდეგ დეფისით. ამ ფორმატში ნიშანი \ გამოიყენება მისი მომდევნო სავალდებულო სიმბოლოს გამოსახვისათვის და სხვ.
  - რიცხვისთვის შეგვიძლია შევარჩიოთ ფულის, პროცენტის ან სხვა ფორმატი. გარდა ამისა, თუ ჩავწერთ 000000, ნებისმიერი რიცხვი ჩაიწერება, როგორც ექვსნიშნა, მაგრამ თუ რიცხვის თანრიგთა რაოდენობა 6-ს აღემატება, იგი გამოისახება სრულად.
  - თარიღისა და დროისთვის ასევე შესაძლებელია ავარჩიოთ შემოთავაზებულიდან ერთ-ერთი ფორმატი.
  - ლოგიკური სიდიდის ტიპისთვის ავარჩევთ: კი/ არა-ს **(Yes/No),** ჭეშმარიტს/ მცდარს **(True/False)** ან ჩართულს/ გამორთულს **(On/ Off) .**
- **Decimal Places –** ნამდვილ **(**წილადის შემცველ**)** რიცხვებში ათობით თანრიგთა სასურველი რაოდენობა.
- **Input Mask –** შეტანის ნიღაბი, მაგალითად, ნიღაბი !00/-00/-00 განაპირობებს 6-ნიშნა ტელეფონის ნომრის მარცხნიდან მარჯვნივ შევსებას და ყოველ ორ ციფრს შორის დეფისის ჩამატებას.
- **Caption –** მინაწერი. აქ შეტანილი ტექსტი მოთავსდება ველის სათავეში, როგორც ველის სახელი. ამ სახელში დაშვებულია წერტილის ან სხვა ნიშნის გამოყენება (რაც დაუშვებელია ველის სათაურში).
- **Default Value –** მონაცემი, რომელიც შემოთავაზებული იქნება საწყის მნიშვნელობად ყოველ ახალ ჩანაწერში.
- **Validation Rule –** ახალი ჩანაწერისათვის შესატანი მონაცემის მნიშვნელობაზე პირობის დასმა.
- **Validation Text –** შეტყობინება **Validation Rule**-ში დასმული პირობის შეუსრულებლობის შესახებ.
- **Required (**სავალდებულო ველი**)** – იმ შემთხვევაში, როდესაც უჯრედის გამოტოვება **(**არშევსება**)** დაუშვებელია, აქ შევიტანთ **Yes.**
- **Allow Zero Length –** ცარიელი სტრიქონის "" შეტანის დაშვება.
- **Indexed –** ველის ინდექსირება, რომელიც შექმნის ველისთვის დამატებით ცხრილს. იგი არ ჩანს, მაგრამ გამოიყენება ძებნის პროცესის დასაჩქარებლად.
- **Unicode Compression –** შემჭიდროვების საშუალება. ეს თვისება აქტიურია მხოლოდ ტექსტის ტიპის მონაცემებისთვის და წინასწარ ყოველთვის **Yes** არის შემოთავაზებული**.**

მონაცემების შეტანის პროცესის გასამარტივებლად და შეცდომების თავიდან ასაცილებლად ცხრილის ველში უჯრედს შეიძლება ჩამოშლადი სია **Lookup** დაერთოს. ამ სიის გამოტანა უჯრედის მარჯვენა მხარეს მდებარე ღილაკით განხორციელდება. ჩამოშლადი სიის დამატება **Lookup** ჩანართზე შემდეგი თვისებების შერჩევით ხდება:

- **Display Control –** მონაცემების შეტანის წესის მართვის ელემენტი. თავიდან შემოთავაზებულია **Text Box** თვისება**,** რაც შეესაბამება მონაცემის შეტანის ჩვეულებრივ წესს. **List Box** ან **Combo Box-**ით უჯრედს დავამატებთ მონაცემების ჩამოშლად სიას. სია შეიძლება ერთი ან რამდენიმე ველისაგან შედგებოდეს, რომლებსაც მხოლოდ ინფორმაციული დატვირთვა ექნებათ, რადგან მონაცემების ამორჩევა ხდება ერთი ველიდან. ჩამოშლილ სიაში შესატანი მონაცემები შეიძლება გადმოვიტანოთ სხვა ცხრილიდან ან მოთხოვნიდან, ან განვსაზღვროთ პირდაპირ **Row Source** სტრიქონზე.
- **Row Source Type –** სტრიქონთა წყაროს ტიპი. აქ მივუთითებთ: ცხრილს/მოთხოვნას - **Table/Query**, მნიშვნელობების სიას - **Value List**, ან ველების სიას - **Field List**.
- **RowSource –** სტრიქონთა წყარო. აქ მივუთითებთ **Table** ან **Query** სახელს. თუ მანამდე ჩართულია **Value List**, მაშინ სტრიქონზე დავწერთ სიაში შესატან მნიშვნელობებს, გამოყოფილს წერტილ-მძიმით.
- **Bound Column –** მიერთებული სვეტების რაოდენობა, მაგალითად, თუ მონაცემი ერთი სვეტიდან არის შესატანი, ჩავწერთ 1.
- **Count Column –** წყაროდან გამოტანილი სვეტების რაოდენობა, თუ ნაჩვენები უნდა იყოს ორი სვეტი, შევიტანთ 2.
- **Column Heads –** მიერთებული სვეტების სათაურების ჩვენება /არ ჩვენება.

- **Column Widths –** სვეტების სიგანე. თუ სვეტის ჩვენება არ არის სასურველი, შევიტანთ 0-ს.
- **List Rows –** სტრიქონთა მაქსიმალური შემოთავაზებული რაოდენობა სიაში. სტანდარტულად შემოთავაზებულია 8.
- **List Width –** სვეტების ჯამური სიგანე.

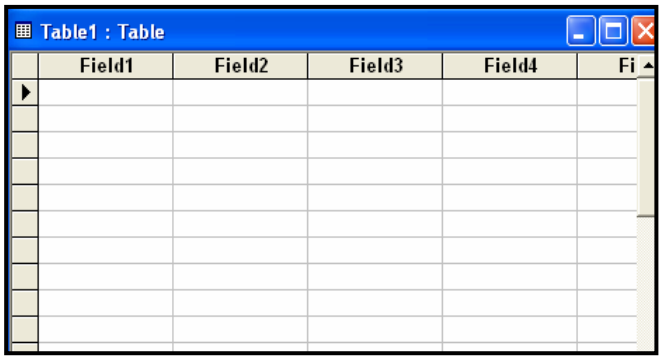
- **Limit to List –** შეზღუდვის არსებობა სიის ჩამონათვალში შემავალი სიდიდეების რაოდენობაზე.

ველების თვისებების განსაზღვრის შემდეგ უნდა განისაზღვროს პირველადი გასაღები-ველი, ამისათვის მოვნიშნოთ საჭირო ველი და შევასრულოთ ბრძანება **Edit → Primary key (**ან შესაბამისი ღილაკით**).** თუ სტრიქონი არ არის მონიშნული, მაშინ სისტემა თვითონ შექმნის პირველად გასაღებ-ველს – თუ ცხრილის რომელიმე ველს აქვს მონაცემების ტიპი **Autonumber,** მაშინ აირჩევს ამ ველს, თუ ასეთი ტიპის ველი ცხრილში არ გვაქვს, მაშინ თვითონ დაამატებს ველს, ავტომატური მთვლელით და მას მიანიჭებს გასაღების ფუნქციას.

ცხრილის სტრუქტურის შექმნის კონსტრუქტორის რეჟიმიდან უნდა მოხდეს ცხრილის ჩვენების რეჟიმში გადასვლა: მენიუს **View →Datasheet View** პუნქტით ან ფანჯრის იმავე სახელწოდების ღილაკით.

ცხრილის შექმნა მონაცემთა შეტანის წესით

თუ ცხრილის შექმნისათვის **Datasheet View** რეჟიმს **(**ნახ.5**)** ავირჩევთ, მაშინ მონაცემებს ელექტრონული ცხრილის **(Excel)** მსგავს ცხრილში შევიტანთ. ველებს სასურველ სახელებს ვანიჭებთ ასე: მოვნიშნავთ ველის სახელს და **Format Rename,** ან თავიდან შემოთავაზებული ველის სახელზე **(**მაგალითად, **Field1)** ორჯერ დაწკაპუნებით მივიღებთ კურსორს და შევიტანთ ახალ სახელს. ყველა მონაცემი ტექსტის ტიპისაა, ანუ შეტანილ რიცხვებზე არითმეტიკულ მოქმედებებს ვერ ჩავატარებთ. მონაცემების შეტანის შემდეგ ცხრილის შენახვასთან ერთად შემოთავაზებული იქნება გასაღები-ველის განსაზღვრის საშუალება, რაც ავტომატურად განხორციელდება ჩვენი თანხმობის შემდეგ. სხვა დანარჩენი ოპერაციები, როგორიცაა სვეტის/სტრიქონის ზომების შერჩევა, გასუფთავება, ამოღება, დამალვა და ა.შ. განხორციელდება **Windows** სისტემაში არსებული წესების ანალოგიურად.



ნახ. 5

მოთხოვნები

ცხრილების სტრუქტურის შექმნის შემდეგ, **Datasheet View** რეჟიმში გახსნილ ცხრილში შეგვაქვს მონაცემები. ამ რეჟიმშივე შეგვიძლია მონაცემების დათვალიერება და რედაქტირება **Windows-** ში არსებული საშუალებების გამოყენებით. ცხრილშივე შეგვიძლია მონაცემების დახარისხება **(**სორტირება**)** და ამორჩევა **(**გაფილტვრა**).** მაგრამ ეს პროცედურები უმჯობესია ჩავატაროთ *მოთხოვნაში* **(Query).** ხშირად საჭირო ხდება მონაცემებზე გამოთვლების ჩატარება, ან რამდენიმე ცხრილიდან გარკვეული მონაცემების ერთობლივად განხილვა და ა.შ. ასეთ შემთხვევაში **Microsoft Access** მოთხოვნაში ქმნის *ჩანაწერთა ნაკრებს* **(***დამაჯამებელ ცხრილს***),** რომელიც ცხრილებიდან გარკვეული წესით შერჩეული ველების საჭირო ჩანაწერების ამონაკრებს შეიცავს. ჩანაწერთა ნაკრებთან **(**დამაჯამებელ ცხრილთან**)** ისე ვმუშაობთ, როგორც ჩვეულებრივ ცხრილთან – განვიხილავთ მონაცემებს, ამოვარჩევთ ინფორმაციას, შევასრულებთ არითმეტიკულ თუ ლოგიკურ მოქმედებებს და ა.შ. უნდა გვახსოვდეს, რომ რეალური ცხრილებისაგან განსხვავებით მონაცემთა ბაზაში ჩანაწერთა ნაკრები **(**დამაჯამებელი ცხრილი**)** ფიზიკურად არ არსებობს. **Access** მას ცხრილებიდან მონაცემების გამოტანის შედეგად ქმნის და ისიც მხოლოდ მოთხოვნის შესრულების განმავლობაში.

მოთხოვნა შეიძლება შევქმნათ ერთი ან რამდენიმე ცხრილის საფუძველზე. როგორც ვთქვით, შექმნილი დამაჯამებელი ცხრილის ველები წარმოადგენენ სხვადასხვა ცხრილებიდან ამორჩეული ველების ნაკრებს, ხოლო ჩანაწერები – ამ ცხრილების გაფილტრულ, დახარისხებულ ან სხვა პირობებით გაერთიანებულ ჩანაწერებს. მოთხოვნის შესრულებაზე გაშვება ხდება ან მენიუს **Query → Run** პუნქტით, ძახილის ნიშნით (!) მოხატული ღილაკით, ან კონსტრუქტორის რეჟიმიდან ცხრილის რეჟიმში **View → Datasheet View** გადასვლით. მოთხოვნა შეიძლება შეიქმნას აგრეთვე სხვა მოთხოვნის ან მოთხოვნების საფუძველზე.

თუ მონაცემთა ბაზა კარგად არის გააზრებული, მაშინ ბაზასთან მომუშავე მომხმარებელმა უნდა დაივიწყოს, რომ არსებობს ცხრილები, კიდევ უკეთესი, თუ მას არაფერი ეცოდინება მათ შესახებ. ცხრილები

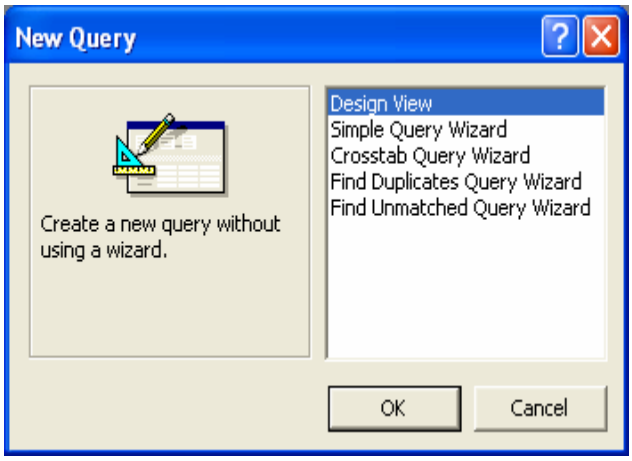
– ბაზის ფასეული ობიექტებია და მათთან ურთიერთობა მხოლოდ ბაზის დამმუშავებელს უნდა ჰქონდეს. როდესაც მომხმარებელს ბაზიდან მონაცემების რაიმე წესით მიღება სურს, მან მხოლოდ მოთხოვნები უნდა გამოიყენოს. კარგია, თუ ბაზის დამმუშავებელი პირი ყველა საჭირო მოთხოვნას წინასწარ მოამზადებს. ასეთ დროს მონაცემთა ბაზის ძირითად ფანჯარაში ჩავრთავთ მოთხოვნების **Queries** ღილაკს და შესაბამისი მოთხოვნის ნიშნაკზე ორჯერ დაწკაპუნებით გავხსნით დამაჯამებელ ცხრილს, რომელშიც ვიპოვით საჭირო ინფორმაციას. დამაჯამებელი ცხრილი მომხმარებლის ინტერესს შესაძლოა არ აკმაყოფილებდეს. ასეთ დროს მომხმარებელი ახალ მოთხოვნას თავად ქმნის ან მიმართავს ბაზის დამპროექტებელს.

ზოგადად, მოთხოვნები დაპროგრამების სპეციალურ **SQL (Struqtured Query Language)** ენაზე იწერება, მაგრამ რიგითი მომხმარებლისთვის მისი შესწავლა სავალდებულო არ არის, რადგან თითქმის ყველა ოპერაცია **Access**-ში თაგვის მეშვეობით და სპეციალური პროგრამებით (ე.წ. ოსტატებით) სრულდება. მოთხოვნას ვქმნით კონსტრუქტორის რეჟიმში **(Design),** რომელშიც იხსნება ე.წ. *მოთხოვნის ბლანკი.*

არსებობს მოთხოვნის აგების სხვადასხვა ვარიანტები. უმარტივესია მოთხოვნა ამორჩევაზე. არსებობს მოთხოვნა პარამეტრით (როდესაც ამორჩევის კრიტერიუმს მომხმარებელი სპეციალურ ფანჯარაში აწვდის), მოთხოვნა შედეგებით (ველებში შეტანილ მნიშვნელობებზე ჩატარებული მათემატიკური და ლოგიკური გამოთვლების შედეგებით), მოთხოვნა შეცვლაზე (როდესაც ხდება ველების მნიშვნელობების განახლება), ჯვარედინი მოთხოვნები (როდესაც დამაჯამებელი ცხრილი ანგარიშების ჩატარების შემდეგ იქმნება), სპეციფიკური მოთხოვნები, ჩაწერილი მოთხოვნების **SQL** ენაზე, მაგალითად, სერვერზე მიმართვის განსახორციელებლად.

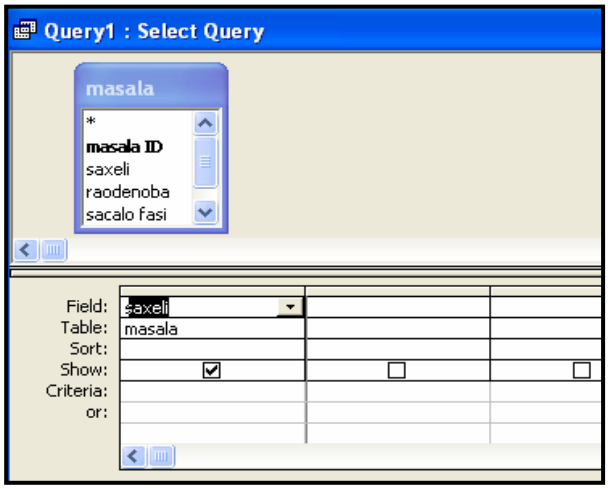
მოთხოვნის შექმნა კონსტრუქტორის რეჟიმში

კონსტრუქტორის რეჟიმში მოთხოვნის შექმნისათვის გავხსნით ბაზის ფანჯარას, ჩავრთავთ **Queries** ღილაკს და **New.** გამოტანილ დიალოგის ფანჯარაში (ნახ.6) შემოთავაზებული ჩამონათვლიდან ავარჩევთ კონსტრუქტორის რეჟიმს **Design View.**



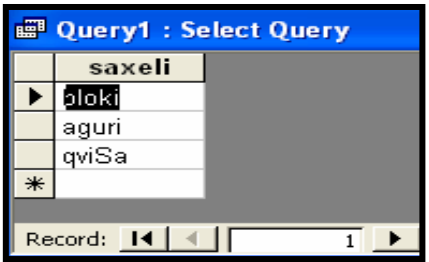
ნახ. 6

გამოტანილ **Show Table** ფანჯარაში ავარჩევთ იმ ცხრილს (ერთს ან რამდენიმეს), რომლის საფუძველზეც იქმნება მოთხოვნა. გაიხსნება მოთხოვნის ბლანკი (ნახ.7). ბლანკი ორი ნაწილისაგან შედგება: ზედა ნაწილში გამოტანილია შერჩეული ცხრილის ან ცხრილების მაკეტები ველების სიით. ცხრილებს შორის კავშირები აქვე იქნება ნაჩვენები (იგულისხმება, რომ ცხრილების დაკავშირება შესრულდა მანამდე). თუ კავშირი მანამდე არ შესრულებულა, ცხრილებს ვაკავშირებთ აქვე თაგვის მეშვეობით.



ნახ. 7

მე-7 ნახაზზე მოყვანილია ბლანკი იმ შემთხვევისათვის, როდესაც მოთხოვნა იქმნება ერთი ცხრილის საფუძველზე. ცხრილის ველების შედგენილობის განხილვისათვის მისი მაკეტიდან ფანჯრის ქვედა ნაწილის **Field** სტრიქონზე თაგვით გადმოვიტანთ სასურველი ველის სახელს (ნახ.7). ცხრილის რეჟიმში გადასვლის შემდეგ დავათვალიერებთ მოთხოვნას – განხილული მაგალითისთვის მასალების სახელების ჩამონათვალს (ნახ.8).



ნახ. 8

ველების თვისებები, როგორც ვიცით, მათი შექმნის დროს დგინდება. მაგრამ შეიძლება გარკვეული თვისების დაწესება მოთხოვნის ველებზეც, რომელსაც უფრო მაღალი პრიორიტეტი ექნება, ვიდრე ცხრილში განსაზღვრულს. ამისათვის შევასრულებთ ბრძანებას **View → Properties** ან დავაჭერთ ხელს ამავე სახელწოდების ღილაკს. გამოტანილ დიალოგში დავაწესებთ ფორმატს, შეტანის ნიღაბს, ველის სათაურს და სხვ.

მოთხოვნის ბლანკის ქვედა ნაწილში მდებარეობს **Show** სტრიქონი, რომელიც დამაჯამებელი ცხრილის ველის შემადგენლობის ეკრანზე ჩვენება/არჩვენებას განსაზღვრავს. იგულისხმება, რომ ყველა ველი, რომელიც მოთხოვნაშია ჩართული, ეკრანზე უნდა იქნას გამოტანილი, ამიტომ ყველა ალამი ჩართულია. მაგრამ ეს ყოველთვის სასურველი არ არის, ასეთ დროს ალამს გამოვრთავთ.

თუ სახელების ანბანით (ინგლისურენოვანი სიის), ან რიცხვითი მნიშვნელობების ზრდადობა-კლებადობით დალაგება გვსურს, დახარისხების **Sort** სტრიქონზე ავარჩევთ შესაბამის წესს. უნდა გვახსოვდეს, რომ ამორჩევის სხვა ვარიანტზე გადასვლის დროს მანამდე დასმული პირობები – კრიტერიუმები ან დახარისხების წესები უნდა მოიხსნას (მაგალითად, **not sorted).** თუმცა შესაძლებელია მრავალდონიანი დახარისხების შესრულებაც.

შესაძლებელია ცხრილის გაფილტვრა, ანუ ველში არსებული მონაცემების სიიდან რაიმე ნიშნის მქონე მონაცემის (ან მონაცემების) ამორჩევა. ამისათვის კრიტერიუმის სტრიქონზე **(Criteria)** ჩავწერთ პირობას, რომელსაც უნდა აკმაყოფილებდეს გამოსატანი მონაცემები, მაგალითად, თუ ვეძებთ იმ ჩანაწერებს, რომლებიც "ა" სიმბოლოთი იწყება, **Criteria** სტრიქონზე მოცემული ველისათვის ჩავწერთ **a*.** თუ საწყისი ცხრილი შეიცავს ველს, რომელშიც რიცხვითი მონაცემია, მაგალითად, საქონლის ფასი ან სხვა, და გვინდა დამაჯამებელ ცხრილში გამოვიტანოთ მხოლოდ ის ჩანაწერები, სადაც ფასი ნაკლებია 200-ზე, მოთხოვნის **Criteria** სტრიქონში შევიტანოთ პირობა: **<200**. ბლანკის მომდევნო **or** სტრიქონზე გამოსახულების **>400** ჩაწერით, მივიღებთ უფრო რთული პირობის ამსახველ მოთხოვნას: **<200** ან **(or) >400**. ასეთი პირობის ჩაწერა შესაძლებელია ერთ სტრიქონზეც: **<200 or >400.** თუ უნდა განხორციელდეს თანკვეთის პირობა, მაგალითად, **>200** და **<400**, მას ჩავწერთ ასე: **>200 and <400.** ასეთივე თანკვეთა განხორციელდება ბლანკის მეზობელ სვეტებში პირობების დაწესებით.

პირობების ჩამოყალიბებისთვის **Access**-ს აქვს სპეციალური ფუნქციები, ზოგიერთი მათგანია:

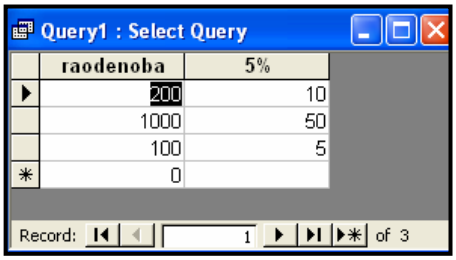
✓ **Between** განსაზღვრავს მნიშვნელობების დიაპაზონს, მაგალითად, **Between 10 and 20** იგივეა, რაც **>=10** და **<=20**;

✓ **In** განსაზღვრავს სიას, საიდანაც უნდა მოხდეს ამორჩევა, მაგალითად, **In("წითელი", "შავი")** იგივეა, რაც "წითელი" ან "შავი".

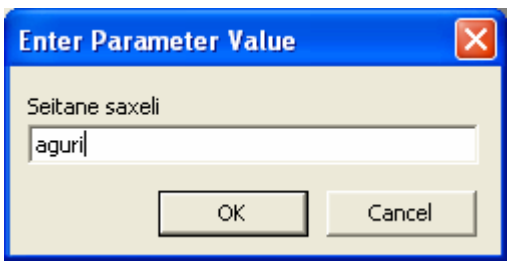
✓ **Like** გამოიყენება ტექსტურ ველებში ნიმუშების მოძებნის მიზნით. მაგალითად, **Like [0-9]** მიუთითებს ციფრების შემცველ დიაპაზონს, **Like [a-z]** – ყველა ლათინურ სიმბოლოს; ძახილის ნიშანი ! შეესაბამება სიმბოლოს გამორიცხვას, მაგალითად, **Like [!0-9]** შეესაბამება ციფრების გარდა ნებისმიერ სიმბოლოს.

✓ **Day(**თარიღი**)** - გამოიტანს დღის შესაბამის 1-31 რიცხვების დიაპაზონიდან, **Month(**თარიღი**)** – თვეს 1-12, **Year(**თარიღი**)** – წელს 100-9999, **Day()** გამოიტანს მიმდინარე თარიღს და ა.შ.

საბაზისო ცხრილების ველებში შეტანილ მნიშვნელობებზე გამოთვლების ჩასატარებლად შეგვიძლია ბლანკის **Field** სტრიქონზე ჩამოვიტანოთ შესაბამისი ველების სახელები, შევქმნათ მოთხოვნის ახალი ველი და იქ შევიტანოთ გამოსახულება. მაგალითად, **[raodenoba]*[erTeulis fasi]** (ველების სახელები ჩავსვათ კვადრატულ ფრჩხილებში). გამოთვლადი ველის შესაქმნელად შეგვიძლია გამოვიყენოთ **Access** პროგრამაში არსებული ნებისმიერი ფუნქცია. მაგალითად, ისევ **masala** ცხრილის მოთხოვნაში გვინდა შევქმნათ ველი, რომელშიც შეტანილი იქნება **raodenoba** ველში მოყვანილი რიცხვების 5%. ამისათვის მოთხოვნის ბლანკის ქვედა ნაწილის ახალ სვეტში მოვათავსებთ კურსორს, მარჯვენა მხარეს მდებარე ისრით გამოვიტანთ ცხრილში არსებული ველების სიას და ავარჩევთ **raodenoba** სახელს. შემდეგ სახელის ბოლოში ვაყენებთ კურსორს და დავამატებთ: *5/100 გამოსახულებას. შევინახოთ **(Save)** მოთხოვნა და ჩავრთოთ ალამი **Show** სტრიქონზე. ცხრილის რეჟიმში გადასვლით დავინახავთ დამატებულ სვეტს, რომელშიც მოყვანილია რაოდენობის შესაბამისი რიცხვების 5% (ნახ.9).



ნახ. 9



სვეტის სახელად **Access** ავტომატურად ირჩევს **Expr1,** მაგრამ თუ დავბრუნდებით კონსტრუქტორის რეჟიმში და გამოვიტანთ **Properties** ფანჯარას, **Caption** სტრიქონზე ჩავწერთ 5%-ს. ეს უკანასკნელი მოთავსდება ველის სათაურში (სახელის ორლში).

მონაცემების გაფილტვრისათვის პირობას უშუალოდ მოთხოვნის ბლანკში **Criteria** სტრიქონზე შევიტანთ. აქვე შეიძლება მოთხოვნაში პარამეტრის ჩართვაც, რომელიც საშუალებას

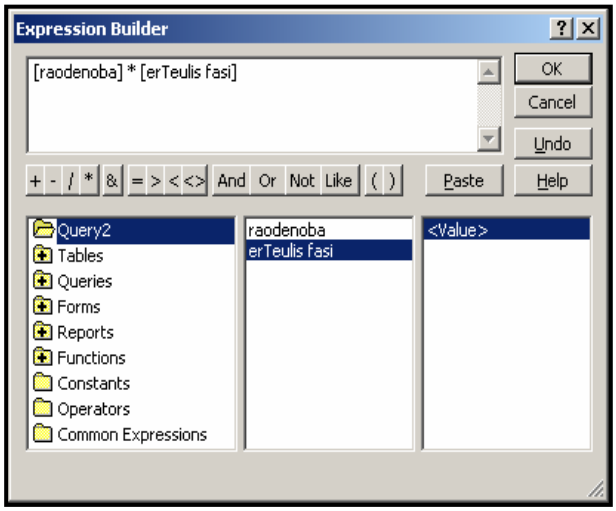
მოგვცემს მოთხოვნის შესრულების პროცესშივე მივუთითოთ ამორჩევის კონკრეტული პირობა. ამისათვის **Criteria** სტრიქონზე

ნახ.10

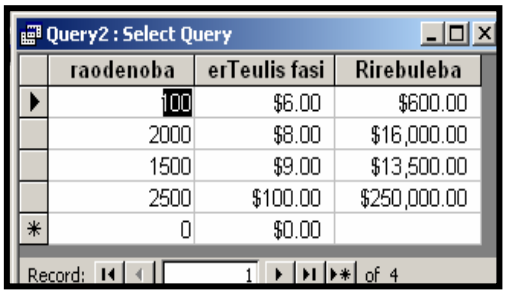
შევიტანოთ [] – კვადრატული ფრჩხილები. მოთხოვნის შესრულებაზე გაშვებისას (ცხრილის რეჟიმში გადასვლა) გამოჩნდება პარამეტრის მოთხოვნის ფანჯარა (ნახ.10), რომელშიც შევიტანთ გაფილტვრის პირობას. ტექსტი **Seitane saxeli** გამოჩნდება მაშინ, როდესაც იგი ჩაიწერება კვადრატულ ფრჩხილებში **Criteria** სტრიქონზე.

გამოსახულების შედგენა სპეციალური პროგრამის გამოყენებით

შედარებით უფრო მაღალი დონის გამოთვლების ჩასატარებლად მივმართავთ **Access**-ში არსებული გამოსახულების ამგებ პროგრამას **Expression Builder**. მაგალითისთვის გამოვიყენოთ ისევ **masala** ცხრილი და შევქმნათ მოთხოვნა, რომელშიც გამოსათვლელია მასალის ღირებულება, ანუ რაოდენობის ნამრავ-ლი ერთეულის ფასზე. მოთხოვნის კონსტრუქტორის რეჟიმში მოთხოვნის ბლანკში ჩამოვიტანოთ ველები **raodenoba** და **erTeulis fasi.** შევინახოთ მოთხოვნა შემოთავაზებული **Query1** სახელით. ამ ორი სვეტის მარჯვენა მხარეს მდებარე ცარიელი სვეტის პირველ სტრიქონზე დავაყენოთ კურსორი და მარჯვენა ღილაკით გამოტანილ კონტექსტურ მენიუში ავარჩიოთ **Built** პუნქტი. გაიხსნება **Expression Builder** ფანჯარა (ნახ.11), რომლის ზედა ნაწილი წარმოადგენს გამოსახულების შეტანის არეს. იგი თავიდან ცარიელია. ფანჯრის ქვედა ნაწილი დაყოფილია სამად, სადაც მოვძებნით გამოსახულებაში მონაწილე ველებს და **Access**-თან მიკავშირებულ ფუნქციებს **Functions**. მაგალითად, ღირებულების გამოსათვლელად უნდა გავამრავლოთ **masala** – ცხრილის **raodenoba** – ველის მნიშვნელობა ველში **erTeulis fasi** შეტანილ მნიშვნელობაზე. ფანჯრის შუა ნაწილში ორჯერ დავაწკაპუნებთ **raodenoba** სახელზე, შემდეგ დავაჭერთ ხელს გამრავლების ღილაკს და ისევ ორჯერ დავაწკაპუნებთ **ველის erTeulis fasi** სახელზე, **OK. Properties** ფანჯრის **Caption** სტრიქონზე ჩავწეროთ **Rirebuleba,** მოთხოვნა დავათვალიეროთ ცხრილის რეჟიმში (ნახ.12). მოთხოვნაში გამოთვლადი ველი აგრეთვე შეიძლება შეიცავდეს ტექსტური ან რიცხვითი ტიპის ველების მნიშვნელობების გაერთიანების (კონკა-



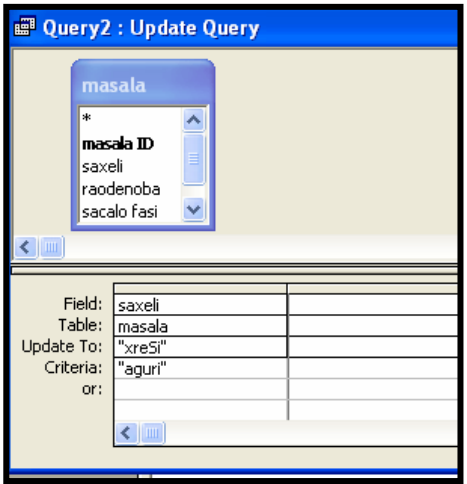
ნახ. 11



ნახ. 12

ტენაციის) შედეგს. ამ და სხვა ხშირად გამოსაყენებელ ფუნქციებს განვიხილავთ სავარჯიშოებში.

თუ გვაინტერესებს ცხრილის არა ცალკეული ჩანაწერები, არამედ მონაცემთა ჯგუფის საშედეგო მონაცემი, შევქმნათ საშედეგო მოთხოვნები. მაგალითად, გვაინტერესებს იმავე **masala** ცხრილში **raodenoba** სვეტში შეტანილი რიცხვთა ჯამი. ამისათვის დავაჭიროთ ხელი ღილაკს ∑. მოთხოვნის ბლანკში გაჩნდება სტრიქონი **Total,** რომლის ჩამოსაშლელი ისრით გამოვიტანთ ფუნქციების ჩამონათვალს: **Sum** გამოიტანს რიცხვთა ჯამს, მაგალითად, თუ სიაში ბევრგან არის შეტანილი მასალა **aguri** და ეს ველიც არის ბლანკზე ჩამოტანილი, გამოითვლის ყველა აგურის შესაბამის რაოდენობის ჯამს, **Avg** – იმავე წესით გამოითვლის რიცხვების საშუალო არითმეტიკულს, **Min** და **Max** იპოვის სიაში რიცხვებს შორის მინიმალურ და მაქსიმალურ მნიშვნელობებს, **Count** – დაითვლის სიაში არსებული მონაცემების რაოდენობას და ა.შ.



ნახ. 13

მოთხოვნას ვიყენებთ მონაცემთა დაჯგუფების მიზნითაც, როდესაც საჭიროა ჩანაწერების ერთობლიობის განახლება, ჩამატება, გაუქმება და ა.შ. ასეთ დროს იქმნება *მოთხოვნა ცვლილებაზე.* ჯერ კონსტრუქტორის რეჟიმში შევქმნით მოთხოვნას ჩანაწერების ჯგუფის ამორჩევაზე, რომელთა განახლებაც გვაქვს განზრახული.

მენიუს **Query** პუნქტში მოყვანილია შესაძლო ცვლილებების სახეობების ჩამონათვალი: განახლება **(Update Query),** დამატება **(Append Query),** გაუქმება **(Delete Query),** ახალი ცხრილის შექმნა **(Make-Table Query). Update Query** რეჟიმის ჩართვის შემდეგ, ბლანკის ქვედა ნაწილში გამოჩნდება სტრიქონი **Update To,** რომელშიც შევიტანთ ახალ მნიშვნელობას. მაგალითში **(**ნახ.13 **)** ყველა **saxeli** ველში, სადაც კი იქნება სიტყვა **aguri,** იგი უნდა შეიცვალოს **xreSi**-ით. შევასრულოთ მოთხოვნა **Run** ბრძანებით. ვინაიდან ამ მოთხოვნის შესრულების შემდეგ მანამდე არსებული ვარიანტის აღდგენა შეუძლებელია **(Windows**-ში არსებული **Undo** ბრძანება ამ შემთხვევაში არ მოქმედებს**),** სისტემა გამოიტანს შესაბამის შეტყობინებას. ცვლილება ცხრილში შეტანილი იქნება დასტურის შემდეგ. ანალოგიურად შესრულდება ჩანაწერთა ჯგუფების რამდენიმე ველში ერთდროული განახლების პროცესი.

მოთხოვნის გამოყენება შეიძლება აგრეთვე ჩანაწერთა ჯგუფის გასაუქმებლად, მაგალითად, **masala** ცხრილიდან გასაუქმებელია ჩანაწერები, რომლებშიც **raodenoba** ველში 1000-ზე ნაკლები რიცხვია შეტანილი, ანუ უნდა გაუქმდეს 1 და 3 ჩანაწერი **(**ნახ.14**).** შევქმნათ მოთხოვნა ამორჩევაზე, **Criteria** სტრიქონზე **raodenoba** ველისთვის შევიტანოთ პირობა **<1000** და მენიუს **Query** პუნქტიდან ჩავრთოთ **Delete Query.** ამის შემდეგ მოთხოვნა გავუშვათ შესრულებაზე **Run,** აქაც წაშლამდე არსებული ინფორმაციის აღდგენა შეუძლებელია **(**თუკი შემდეგ გაფრთხილების სისტემა ჩვენგან მიიღებს თანხმობას**).**

masala : Table

masala ID	saxeli	raodenoba	sacalo fasi	daxasiaTeba
1	bloki	200	$10.00	
2	xreSi	1000	$6.00	
4	qviSa	100	$100.00	
5	xreSi	2000	$6.00	
(AutoNumber)		0	$0.00	

Record: 1 of 4

ნახ. 14

მონაცემთა იმპორტი, ექსპორტი

ცხრილები შეგვიძლია შევქმნათ მონაცემების გადმოტანით სხვა მონაცემთა ბაზიდან ან ელექტრონული ცხრილიდან **(Excel**-დან**).** იმპორტის გარდა **Access**-ში უკუმოქმედებაც არის შესაძლებელი – ცხრილების **Access**-დან ექსპორტი სხვა პროგრამებში.

განვიხილოთ ელექტრონული ცხრილიდან მონაცემების **Access**-ში იმპორტის მაგალითი:

1. შევქმნათ **Excel**-ში მარტივი სტრუქტურის ცხრილი:

saxeli	qula
maka	100
ana	200
irakli	500

2. გავხსნათ მონაცემთა ბაზა, სადაც ვაპირებთ ამ ცხრილის იმპორტს, მაგალითად, **samS masalebi;**

3. ჩავრთოთ მენიუს პუნქტი **File → Get External Data → Import;**

4. გაიხსნება **Import** ფანჯარა, რომლის ქვედა ნაწილში **Files of type** დავაყენოთ **Microsoft Excel,** სიაში მითითებული იქნება ფაილი, სადაც შენახულია ჩვენს მიერ შექმნილი ცხრილი, ავარჩიოთ იგი და **Import;**

5. ამუშავდება **Wizard** პროგრამა, რომლის პირველ დიალოგ- ფანჯარაზე გამოტანილი იქნება ცხრილის მაკეტი, **Next;**

6. მომდევნო დიალოგ-ფანჯარაზე ჩართულია ალამი **First Row Contains Column Headings**, რომლითაც ცხრილის პირველი სტრიქონი **(saxeli qula)** განსაზღვრული იქნება, როგორც ცხრილის სათავე, **Next;**

7. მომდევნო დიალოგში ზუსტდება, თუ სად უნდა იყოს ცხრილი იმპორტირებული – ახალ ცხრილში თუ არსებულში, **Next;.**

8. მომდევნოზე შერჩეულ სვეტს შეგვიძლია ინდექსირების თვისება მივანიჭოთ, **Next;**

9. მომდევნოზე ჩართულია ღილაკი **Let Access add primary key,** რაც შეესაბამება იმას, რომ პირველადი გასაღები-ველი **ID** დაემატება სისტემის მიერ, **Next;**

10. ბოლო დიალოგში ცხრილს მინიჭებული აქვს სავარაუდო სახელი, შეგვიძლია იგი შევცვალოთ, **Finish.** მიღებულია ბაზაში იმპორტირებული ელექტრონული ცხრილი (ნახ. 15).

Sheet1 : Table

ID	saxeli	qula
1	maka	100
2	ana	200
3	irakli	500
(AutoNumber)		

ნახ. 15

ანალოგიურად მოხდება ცხრილების ექსპორტი სხვა პროგრამებში და კავშირის დამყარება **Link Tables** სხვა პროგრამების ცხრილებთან, რის შედეგად ცხრილში შეტანილი ცვლილება გამოიწვევს შესაბამისი უჯრედის შიგთავსის ცვლილებას.

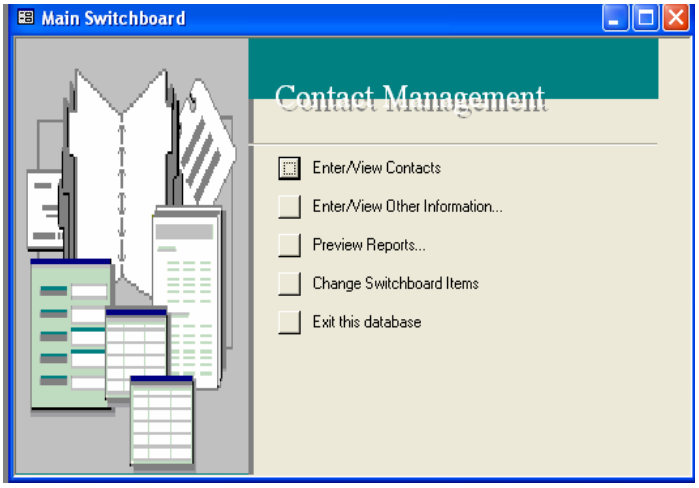
სავარჯიშო 1. მონაცემთა ბაზის შექმნა ოსტატის გამოყენებით

შევქმნათ მონაცემთა ბაზა ოსტატის **(Wizard)** დახმარებით. ავაგოთ იგი ერთ-ერთი შაბლონის საფუძველზე. ოსტატის მიერ შეიქმნება შაბლონის შესაბამისი ცხრილების სტრუქტურა, ფორმები და ანგარიშები. ასეთი ბაზა სრულად დააკმაყოფილებს ტიპურ ამოცანაში დასმულ საკითხებს. ოსტატის გამოსაყენებლად შევასრულოთ შემდეგი მოქმედებები:

1. **Start→ Programs→ Microsoft Access,** ჩატვირთვის შემდეგ ავარჩიოთ ოსტატის დახმარებით ბაზის შექმნის რომელიმე ვარიანტი არსებული შაბლონების საფუძველზე (**Access database wizards, pages, and projects**), დავაწკაპუნოთ ღილაკზე **OK** (მომდევნო ტექსტში მითითებული იქნება მხოლოდ ღილაკის სახელი).
2. გამოჩნდება დიალოგი-ფანჯარა **New,** რომლის **Databases** ჩანართზე იქნება მოყვანილი სხვადასხვა თემატიკის ბაზების შაბლონების ნიშნაკები.

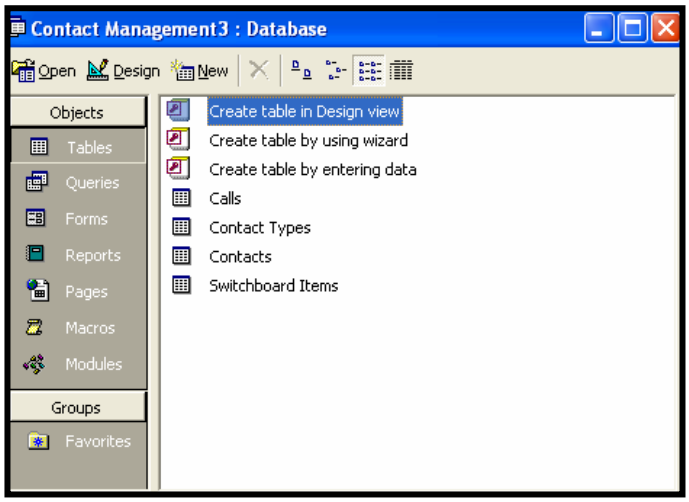
ნახ. 16

3. ვინაიდან შაბლონებთან მუშაობა ერთნაირად მიმდინარეობს, ბაზის შექმნის პროცესს განვიხილავთ ერთის მაგალითზე. შაბლონის ნიშნაკის მონიშვნის შემდეგ **Preview** ველში დავინახავთ სურათს, რომელიც ზოგად წარმოდგენას შეგვიქმნის შაბლონის დანიშნულებაზე. ამოვარჩიოთ, მაგალითად, შაბლონი **Contact Management**, **OK.** გაიხსნება დიალოგი-ფანჯარა **File New Database**. **Save in** ველში მივუთითოთ იმ საქაღალდის სახელი, სადაც გვსურთ ბაზის ფაილის შენახვა, მაგალითად, **My Documents,** ხოლო ველში **File Name** ჩავწეროთ მონაცემთა ბაზის ფაილის სახელი, **Create**.
4. ოსტატი დაიწყებს მუშაობას და გამოიტანს პირველ ფანჯარას, რომელშიც გვატყობინებს შესაქმნელი ცხრილების სახელებს, ამ შემთხვევაში, **Contact information, Call information, Next.** მომდევნო ფანჯრის მარცხენა მხარეს (ნახ.16) გამოტანილი იქნება ცხრილების ჩამონათვალი – 2 საბაზისო (**Contact information** და **Call information**) და 1 გადამკვეთი (**Contact Types**), ფანჯრის მარჯვენა მხარეს – მონიშნული ცხრილის ველების ჩამონათვალი. მათი გამორიცხვა დაუშვებელია, ხოლო შემოთავაზებულებიდან რომელიმეს ჩართვა – შესაძლებელი, **Next.**
5. მომდევნო ფანჯარაზე ოსტატი გვთავაზობს შესაქმნელი ფორმების სტილს, ავარჩიოთ ნებისმიერი, **Next.**
6. მომდევნოზე ოსტატი გვთავაზობს შესაქმნელი ანგარიშების სტილს, ავარჩიოთ ნებისმიერი, **Next.**
7. უკანასკნელ 2 ფანჯარაში ბაზას უნდა მივანიჭოთ სახელი და ოსტატს მივცეთ ბრძანება ბაზის დაუყოვნებლივ შექმნის შესახებ (**Start The Database**), **Next,** ბოლოს **Finish.**



ნახ. 17

8. ოსტატი შეუდგება მუშაობას და გარკვეულ დროში დაასრულებს ამ პროცესს: გამოიტანს ბაზის მთავარ, გადამრთველ დაფას (**Main Switchboard),** რომელზეც 5 ღილაკია განთავსებული (ნახ.17): პირველით გამოვიტანთ ე.წ. საკონტაქტო ფორმას ახალი ინფორმაციის შეტანისა და არსებულის დათვალიერების მიზნით, მეორით ინფორმაციის შეტანის ფორმას, მესამით დავათვალიერებთ ანგარიშებს, მეოთხე ღილაკით თვით ამ დაფის რედაქტირებაა შესაძლებელი, მეხუთე ღილაკი ბაზიდან გასვლის ღილაკია.
9. ბაზის ძირითადი ფანჯარა ჩაკეცილია, გავხსნათ იგი ჩვეულებრივი წესით (ნახ.18). ცხრილებს **(Tables)** შორის: 3 ცხრილია ინფორმაციის შემცველი და 1 გადამრთველი დაფის. ცხრილების გახსნისას **(Open)** დავინახავთ მხოლოდ მათ სტრუქტურას, მონაცემების გარეშე.



ნახ. 18

10. დავათვალიეროთ აგრეთვე ფორმები **Forms** და ანგარიშები **Reports.**

11. ბაზის სტრუქტურა შექმნილია, შევიტანოთ მონაცემები მიღებული ფორმების მეშვეობით, წავიკითხოთ ისინი ცხრილებში.

სავარჯიშო 2. ცხრილის შექმნა კონსტრუქტორის რეჟიმში

განვიხილოთ ამოცანა: სამშენებლო მასალებით მოვაჭრე ფირმის ხელმძღვანელი უკვეთავს სპეციალისტს მონაცემთა ბაზას. ბაზა ორ ცხრილს უნდა შეიცავდეს: ერთში მყიდველზე ორიენტირებული მონაცემები იქნება შესატანი (მასალის სახელწოდება, რაოდენობა, ერთეულის ფასი, დახასიათება), მეორეში – ფირმის საქმიანობის მაჩვენებლები (გაყიდვის თარიღი, დღე, მოკლე ინფორმაცია მყიდველის შესახებ).

აქ და შემდგომ ველების სახელები და მონაცემები პირობითად იქნება შერჩეული – მხოლოდ სასწავლო პროცესზე ორიენტირებით, კერძოდ, ველებში შესატანი მონაცემების ტიპების და მათი თვისებების შესწავლის მიზნით. სახელებისთვის გამოყენებულია ქართული სიტყვები, დაწერილი ლათინური ასოებით – **AcadNusx** შრიფტში, არსებული ქართული ასოების წარმოდგენის წესებზე დაყრდნობით. მიზანშეწონილი არ არის ქართულ შრიფტზე ცხრილშივე გადასვლა, ამ ოპერაციას ფორმის შექმნის დროს განვახორციელებთ.

1. ჩავტვირთოთ **Microsoft Access 2000 (Start→ Programs → Microsoft Access).**
2. გამოტანილ **Microsoft Access** ფანჯარაში ჩავრთოთ **Blank Access database, OK.**
3. **File New Database** ფანჯრის **Save in** ველში შევარჩიოთ **My Documents** ან სხვა სასურველი საქაღალდე, **File name** ველში შევიტანოთ ფაილის სახელი, მაგალითად, **samSmasalebi.** ფაილის ტიპში **Save as type** უნდა იყოს მითითებული **Microsoft Access Databases, Create.**
4. ბაზის ძირითად ფანჯარაში დავაჭიროთ ხელი **Tables** ღილაკს, შემდეგ **New.** გამოტანილ დიალოგში ავარჩიოთ **Design View, OK.** იმავეს მივაღწევთ **Create table in Design View** ნიშნაკზე ორჯერ დაწკაპუნებით.
5. ცხრილის სტრუქტურის განსაზღვრის ბლანკში შევიტანოთ ველების სახელები და შევარჩიოთ შესატანი მონაცემების ტიპები. აღწერილობის შევსება **Description** სავალდებულო არ არის.

Field Name	Data Type	Description
masala ID	Autonumber	
saxeli	Text	
raodenoba	Number	
erTeulis fasi	Currency	
daxasiaTeba	Memo	

6. მეორე ცხრილთან დასაკავშირებლად **masala ID** ველს მივანიჭოთ პირველადი გასაღები-ველის თვისება: ამისათვის ამ სტრიქონზე მოვათავსოთ კურსორი, ავარჩიოთ მენიუს **Edit → Primary Key** პუნქტი ან ხელი დავაჭიროთ იმავე სახელწოდების ღილაკს. სტრიქონის თავში გამოჩნდება გასაღების გამოსახულება. ამ ველში შესატანი მონაცემებისთვის შევარჩიოთ მთვლელის ტიპი **Autonumber**, რის შედეგად ჩანაწერების შეტანის დროს განხორციელდება მათი ავტომატური დანომვრა.
7. სხვა დანარჩენ ველებს, რომლებშიც შესატანი იქნება მასალების სახელები, რაოდენობა, ერთეულის ფასი და დახასიათება, მივანიჭოთ შესატანი მონაცემების შესაბამისი ტიპები: ტექსტის, რიცხვითი, ფულადი და მოზრდილი სიდიდის ტექსტის.
8. ცხრილი შევინახოთ – **File→Save**, შევარჩიოთ სახელი, მაგალითად, **masala, OK.**
9. თუ რა გავლენას ახდენს თითოეული ველის თვისება **Field Properties** ცხრილში შესატან მონაცემებზე, შევამოწმოთ მათი შერჩევის ან ცვლილების გზით (ყოველი ამ მოქმედების შემდეგ ცხრილი შევინახოთ**).** ბლანკის ქვედა ნაწილში **Field Properties** ყოველი ველისთვის ავტომატურად შერჩეული იქნება თვისებები, მაგალითად, ის, რომ პირველადი გასაღები-ველისთვის ინდექსირება საჭიროა, რომ მათი მნიშვნელობები არ უნდა მეორდებოდეს – **Indexed Yes (No Duplicates).**

10. ჩავატაროთ ექსპერიმენტი: **saxeli** ველის ზომა **Field Size** შემოთავაზებული 50 სიმბოლოს ნაცვლად ავიღოთ 5, კონსტრუქტორის რეჟიმიდან გადავიდეთ ცხრილის შევსების რეჟიმში **(View → Datasheet View** ან შესაბამისი ღილაკით).
11. კლავიატურის **Tab** კლავიშით ან თაგვით, კურსორი მოვათავსოთ **saxeli** ველში და შევიტანოთ 5-ზე მეტსიმბოლოიანი სახელი. დავინახავთ, რომ 5-ზე მეტი სიმბოლო შეტანილი არ იქნება, რის შესახებ გამოიცემა სასიგნალო ხმა. თანაც, დანარჩენი სიმბოლოები დაიკარგება აღდგენის საშუალების **(Undo)** გარეშე.
12. **Caption** სტრიქონზე შევიტანოთ ველის სათავის ახალი სახელი, რომელიც შეიცავს ველის სათავისთვის ზოგადად დაუშვებელ ნიშნებს: წერტილს, მძიმეს და სხვ. დავწეროთ, მაგალითად, **mas.saxeli.** ცხრილის რეჟიმში გადასვლისთანავე დავინახავთ ამ სახელს ველის სათავეში.
13. **Default Value** სტრიქონზე შეგვიძლია ჩავწეროთ ის მნიშვნელობა, რომელიც ჩვენი მოსაზრებით უნდა დაგვხვდეს ხოლმე ყოველ ახალ ჩანაწერში, მაგალითად, დავწეროთ **bloki.** მანამდე შეტანილ ჩანაწერებზე ეს გავლენას არ მოახდენს.
14. თვისებებში შეგვიძლია დავაწესოთ მოთხოვნა ველის შეუვსებლობის არდაშვების შესახებ **Required** – **Yes.** მაშინ, ჩანაწერის შეტანის დროს, ამ ველში მნიშვნელობის გამოტოვების შემთხვევაში გამოტანილი იქნება შენიშვნა იმის თაობაზე, რომ ველი არ უნდა შეიცავდეს ცარიელ სტრიქონს "", იგი უნდა შეივსოს რაიმე მნიშვნელობით. ეს არის დაცვა მონაცემის შემთხვევით გამოტოვებისაგან.
15. თუ ველის არშევსება დასაშვებია **(Required – No)** და გამოტოვების შემთხვევაში უნდა იყოს აღქმული, როგორც არა ცარიელი სტრიქონი "" (იმ გაგებით, მაგალითად, რომ კლიენტს ფაქსი არ აქვს), არამედ როგორც ცარიელი მნიშვნელობა **Null Value** (იმ გაგებით, რომ ფაქსი არის, მხოლოდ ნომერი უცნობია)**,** მაშინ თავიდანვე **Allow Zero Length** შევიტანოთ **Yes.** მონაცემის გამოტოვების შემთხვევაში იგი იქნება დაფიქსირებული, როგორც ცარიელი მნიშვნელობა **Null Value,** და ცხრილის შემდგომი ანალიზის დროს ეს გარემოება იქნება გამოყენებული.
16. ცხრილის რეჟიმში შევიტანოთ 6-7 ჩანაწერი. ერთი ან რამდენიმე ჩანაწერი მოვნიშნოთ და გავაუქმოთ **Edit → Delete** ან **Delete Record.** გაუქმებაზე დასტურის შემდეგ ჩანაწერის აღდგენა შეუძლებელია. ყურადღება მივაქციოთ იმას, რომ გაუქმებული ჩანაწერის მიმდინარე ნომერი – იდენტიფიკატორი (იგი **Autonumber** ტიპისაა) აღარ აღდგება.
17. კონსტრუქტორის რეჟიმში მეორე ველისთვის **raodenoba** თვისებების **Field Size** სტრიქონზე ავარჩიოთ რიცხვითი ტიპის სხვადასხვა ფორმატი: **Byte, Integer, Long Integer.** შევამოწმოთ ფორმატების გავლენა შეტანილ მონაცემებზე, რისთვისაც გადავიდეთ ცხრილის რეჟიმში და პირველი ფორმატის შერჩევის შემდეგ შევიტანოთ 4-ნიშნა, მეორის – 6-ნიშნა, ხოლო მესამის – 11-ნიშნა რიცხვი. ყოველი ამ შემთხვევისათვის გამოტანილი იქნება შეტყობინება ველის სიგრძის გადამეტების შესახებ.
18. დავაწესოთ იმავე **raodenoba** ველისთვის რიცხვის გამოსახვის თანრიგთა რაოდენობა: თუ **Format** სტრიქონზე შევიტანთ 000000, მაშინ ცხრილში შეტანილი რიცხვი, მაგალითად, 5, ჩაიწერება 000005 სახით.
19. თუ ამ ველისთვის გვსურს ზღვრული მნიშვნელობის დაწესება, რომლის გადამეტება დაუშვებელია, თვისებების **Validation Rule** სტრიქონზე შევიტანოთ პირობა, მაგალითად, **<200**. თუ ცხრილის რეჟიმში ამ უჯრედში შევიტანთ რიცხვს 200, გამოტანილი იქნება შეტყობინება პირობის შეუსრულებლობის შესახებ, რადგან 200 არ არის 200-ზე ნაკლები. შეტყობინების ტექსტი შეგვიძლია თავად ჩამოვაყალიბოთ **Validation Text** სტრიქონზე, მაგალითად, **"ar aris naklebi 200-ze!".**
20. ველისთვის **erTeulisfasi** თვისებების **Format** სტრიქონზე ავარჩიოთ **Currency**, ხოლო წილადში ათობით ციფრთა რაოდენობა **Decimal Places** ავიღოთ 2, მაშინ ცხრილის რეჟიმში რიცხვი 5 შეტანილი იქნება **$**5.00 სახით.
21. **daxasiaTeba** ველი ცხრილის რეჟიმში შევავსოთ ვრცელი ტექსტით, იგი მთლიანად არ გამოჩნდება ცხრილში. ტექსტის სრულად გამოსატანად შეგვიძლია ველის სიგანე გავადიდოთ ცნობილი ხერხებით: თაგვით ან **Format → Column Width → Best Fit** პუნქტით.
22. ახალი ველის ან ჩანაწერის ჩამატება/ამოღება განვახორციელოთ მიღებული წესით – როგორც ცხრილის, ისე კონსტრუქტორის რეჟიმში **(Edit →Delete/ Insert → Record/ Column).** მივაქციოთ ყურადღება, რომ ამ მოქმედებების უარყოფის **(Undo)** შესაძლებლობა არ გვაქვს.

საგარჯიშო 3. ცხრილის ველების თვისებების რედაქტირება

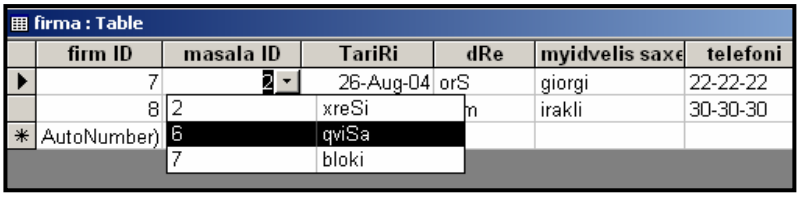
მე-2 საგარჯიშოში მოყვანილი ამოცანის გასაგრძელებლად გავიმეოროთ 1-ლი პუნქტი. შემდეგ:

1. ავარჩიოთ არსებული ბაზის გახსნის **Open an existing file** პუნქტი, მოვნიშნოთ ბაზის სახელი **samSmasalebi**, **OK.**
2. გავიმეოროთ მე-2 სავარჯიშოში მოყვანილი მე-4 პუნქტი. შევიტანოთ ველების სახელები და განვსაზღვროთ მათი ტიპები:

Field Name	Data Type	Description
firm ID	**Autonumber**	
masala ID	**Number**	
TariRi	**Date/Time**	

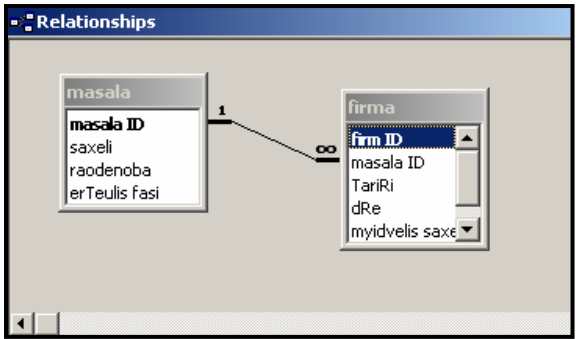
dRe	**Text**
myidvelis saxeli	**Text**
telefoni	**Text**

3. ცხრილის შენახვისას (წინა სავარჯიშოს პუნქტი 8) მივანიჭოთ მას სახელი **firma.** განვსაზღვროთ ცხრილში საკუთარი პირველადი გასაღები-ველის **firm ID** ტიპი, როგორც **Autonumber,** ხოლო ამ **firma** ცხრილის **masala** ცხრილთან დასაკავშირებლად დავამატოთ შესაბამისი **masala ID** ველი, როგორც გარე გასაღები. **masala** ცხრილში იგი იყო **Autonumber** ტიპის**,** ამ ცხრილში იგი უნდა განისაზღვროს, როგორც **Number** ტიპი. ზოგადად, სხვა დანარჩენი ტიპებისთვის მისაკავშირებელი ველები ერთნაირი ტიპებით განისაზღვრებიან.
4. დანარჩენ ველებს, რომლებშიც შესატანი იქნება გაყიდვის თარიღი, მყიდველის სახელი და ტელეფონი, მივუსადაგოთ მონაცემების **Date/Time** და **Text** ტიპები.
5. თუ რა გავლენას ახდენს თითოეული ველის თვისება **Field Properties** ცხრილში შესატან მონაცემებზე, შევამოწმოთ მათი შერჩევის ან ცვლილების გზით (ყოველი ამ მოქმედების შემდეგ ცხრილი შევინახოთ). თარიღის ფორმატად **Format** სტრიქონიდან ავარჩიოთ ერთ-ერთი, მაგალითად, **Short Date**. შესაძლებელია აგრეთვე ყოველი ახალი ჩანაწერისთვის წინასწარ განისაზღვროს კალენდარული თარიღი, ამისათვის **Default Value** სტრიქონზე შევიტანოთ **Date().**
6. ტელეფონის ნომერი განსაზღვრულია, როგორც ტექსტური ტიპი. ეს აიხსნება იმით, რომ ტელეფონის ნომერს არ გააჩნია რიცხვითი მონაცემის შინაარსი – ნომრებზე შეკრების, გამრავლების და სხვა მოქმედებების ჩატარება არ ხდება. ტელეფონის ნომრის შეტანისთვის გამოვიყენოთ ნიღაბი, რაც გააადვილებს შეტანის პროცესს და გამორიცხავს ნომრის არასწორად აკრეფას. ბლანკის ქვედა ნაწილში **Field Properties Input Mask** სტრიქონზე შევიტანოთ, მაგალითად, გამოსახულება !00\-00\-00. იგი უზრუნველყოფს ტელეფონის ნომრის მარცხნიდან მარჯვნივ ორ-ორი ციფრით შევსებას და მათ შორის დეფისის ავტომატურად ჩამატებას, მაგალითად, 22-22-22.
7. განვახორციელოთ **Access**-ში არსებული ჩანაცვლების **Lookup** მეთოდი, რომელსაც ვიყენებთ შეტანის პროცესის გამარტივების და შეცდომების გამორიცხვის მიზნით. მაგალითად, **dRe** ველში შესატანია შემდეგი მოცემებიდან ერთი: ორშ, სამ, ოთხ, ხუთ, პარ, შაბ, კვირა. კონსტრუქტორის რეჟიმში მოვნიშნოთ ეს ველი (დავაყენოთ კურსორი) და **Lookup** ჩანართის **Display Control** სტრიქონზე ავარჩიოთ **List Box, Row Source Type** სტრიქონზე – **Value List**, **Row Source** სტრიქონზე შევიტანოთ ჩამონათვალი: **orS; sam; oTx; xuT; par; Sab; kvira**. სხვა დანარჩენ სტრიქონებზე უცვლელი დავტოვოთ ავტომატურად გამოტანილი მნიშვნელობები. ცხრილის რეჟიმში გადასვლის შემდეგ ველის მარჯვენა მხარეს გაჩნდება სიის ჩამოსაშლელი ისარი.
8. ჩანაცვლების შესაძლებლობა გამოვიყენოთ მაშინაც, როდესაც მასალის სახელის შესაბამისი კოდის შეტანაა საჭირო.



ნახ. 19

9. კონსტრუქტორის რეჟიმში მოვნიშნოთ **firma** ცხრილის **masala ID** ველი (დავაყენოთ კურსორი), **Lookup** ჩანართის **Display Control** სტრიქონზე ავარჩიოთ **Combo Box, Row Source Type** სტრიქონზე – **Table/Query**, **Row Source** სტრიქონზე დავაწკაპუნოთ **masala** ცხრილზე. მისაკავშირებელი სვეტების რაოდენობა **Bound Column** ავიღოთ 1, ჩამოსაშლელი სვეტების რაოდენობა **Column Count** – 2, ჩამოშლილი ცხრილის სათაურები **Heads – No (**არ იყოს ნაჩვენები), თითოეული სვეტის სიგანე **Column Widths -1",** მთლიანად სიის (ორივე სვეტის ერთად) სიგანე **List Widths - 2",** დანარჩენი ავტომატურად გამოტანილი თვისებები დავტოვოთ უცვლელი.
10. განვახორციელოთ არსებულ ცხრილში მონაცემის ტიპის შეცვლა. როგორც ვიცით, ეს შესაძლებელია გარკვეული წესების დაცვით. თუ ველი განსაზღვრულია, როგორც რიცხვითი, მაშინ მისი ტექსტურად გადაკეთება არ იწვევს პრობლემას. ანალოგიურად იცვლება სხვა ტიპები. კონსტრუქტორის რეჟიმში მოვნიშნოთ ნებისმიერი ველი და არსებული ტიპი შევცვალოთ ტექსტურად. გამონაკლისს წარმოადგენს მთვლელის ტიპი, განსაკუთრებით მაშინ, როდესაც ცხრილებს შორის კავშირი დამყარებულია ამ ველის საშუალებით.
11. ტექსტური ტიპის რიცხვად, თარიღად ან სხვა ტიპად გადაკეთება არ წარმოადგენს პრობლემას, თუ მონაცემები ციფრების სახითაა შეტანილი. ჩვეულებრივი ტექსტის რიცხვად გადაკეთება კი შეუძლებელია. დროებით შევცვალოთ **TariRi** ველის თარიღის ტიპი – რიცხვით ტიპად, ყველა შეტანილი თარიღის მაჩვენებელი შეიცვლება ამ თარიღის შესაბამისი რიცხვითი მნიშვნელობით. შევასრულოთ უკუპროცესი.



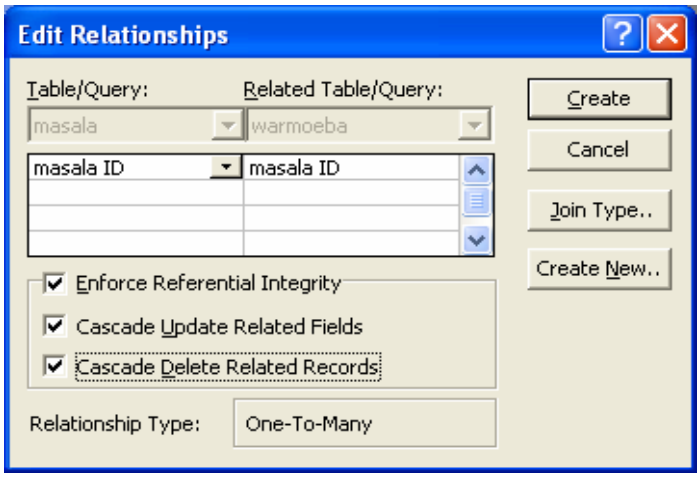
სავარჯიშო 4. ცხრილებს შორის კავშირების დამყარება

განვიხილოთ მე-2 – მე-3 სავარჯიშოებში ნაჩვენები მონაცემთა ბაზა **samSmasalebi** ცხრილებით **masala** და **firma,** დავამყაროთ მათ შორის კავშირი. გავიმეოროთ მე-3 სავარჯიშოს 1-ლი პუნქტი. შემდეგ:

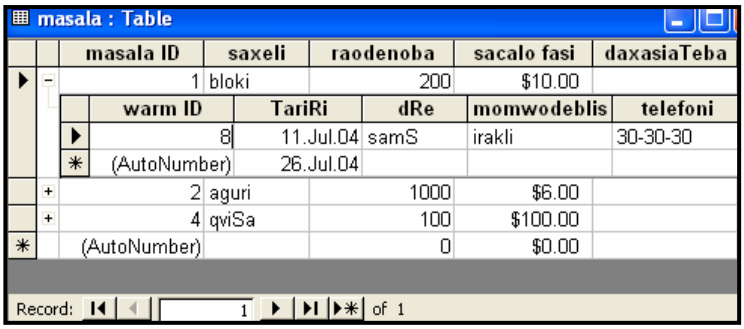
ნახ.20

1. გახსნილი ბაზის მთავარ ფანჯარაში ორი ცხრილია: **masala** და **firma,** რომელთა შორის უნდა დამყარდეს კავშირი. ჩავრთოთ მენიუს პუნქტი **Tools → Relationships** ან დავაჭიროთ ხელი იმავე სახელწოდების ღილაკს, გამოტანილი იქნება ცხრილების ჩვენების **Show Table** ფანჯარა. ცხრილების სახელების მონიშვნის შემდეგ **Add**-ით გადავიტანოთ სახელები სქემის **Relationships** ფანჯარაზე და დავხუროთ **(Close).** თუ სქემის ფანჯარაზე ცხრილების დამატება კიდევ იქნება საჭირო, ჩავრთავთ მენიუს პუნქტს **View→ Show Table** ან დავაჭერთ ხელს იმავე სახელწოდების ღილაკს.

ნახ.21



2. სქემის ფანჯარაზე **Relationships** ნაჩვენებია ცხრილების მაკეტები ველების სიებით (ნახ.20). **masala** ცხრილიდან თაგვით პირველადი გასაღების **masala ID** ველი გადმოვიტანოთ **firma** ცხრილის იმავე სახელწოდების **masala ID** გარე გასაღებზე. გამოტანილი იქნება დიალოგი **Edit Relationships** (ნახ.21). აქ **Relationship Type**-ში მითითებულია კავშირი "ერთი მრავალთან" **One-To-Many,** რაც სწორია. ჩავრთოთ ალამი **Enforce Referential Integrity,** რითაც დაცული იქნება მონაცემები, მაგალითად, არარსებული **masala** კოდის **firma** ცხრილში შეტანისაგან.



ნახ. 22

3. შევამოწმოთ ეს უკანასკნელი ვითარება: **masala** ცხრილში შევიტანოთ ჩანაწერები კოდებით 1, 2 და 4. **firma** ცხრილში **masala ID** ველში შევიტანოთ ჩანაწერი არარსებული კოდით 3. შედეგად გამოტანილი იქნება შესაბამისი შეტყობინება.
4. **Enforce Referential Integrity** ჩართვის შემდეგ გააქტიურდებიან ალმები **Cascade Enforce Referential Integrity** და **Cascade Delete Related Records**, რომლებითაც ყოველი ცვლილება "ერთის" მხარეზე გამოიწვევს ცვლილებას "მრავალის" მხარეზე. შევამოწმოთ ეს ვითარება.
5. ზოგიერთ შემთხვევაში საჭირო ხდება კავშირის გაუქმება, ამისათვის გავხსნათ **Relationships** ფანჯარა, მოვნიშნოთ კავშირის ხაზი და კლავიატურის **Delete** ღილაკით გავაუქმოთ იგი.
6. დავამყაროთ ცხრილებს შორის კავშირი თავიდან, გავხსნათ **masala** ცხრილი და მივაქციოთ ყურადღება მარცხენა მხარეს დამატებულ + ნიშნებს. ამ ნიშანზე დაწკაპუნებით გაიხსნება **firma** ცხრილის ის ნაწილი, რომელიც შეესაბამება **masala ID** ველის კოდის მნიშვნელობას (ნახ.22).

სავარჯიშო 5. გადამკვეთი ცხრილის შექმნა "მრავალი მრავალთან" კავშირის განსახორციელებლად; კავშირი "ერთი ერთთან"

განვიხილოთ მე-2 სავარჯიშოში შესრულებული **samSmasalebi** მონაცემთა ბაზა.

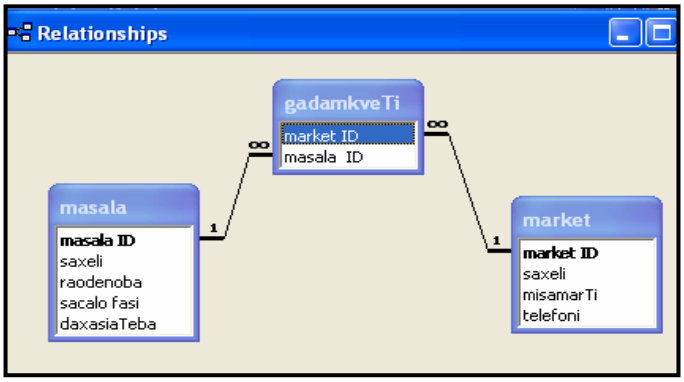
1. შევქმნათ ახალი ცხრილი იმ სავაჭრო დაწესებულებების **marketi** მონაცემებით, რომლებშიც უნდა განხორციელდეს მასალების რეალიზაცია. აქ შემდეგი მონაცემები იქნება შესატანი: მაღაზიის სახელი, მისამართი და ტელეფონი. მისი ველები კონსტრუქტორის რეჟიმში ჩამოვაყალიბოთ ასე:

Field Name	Data Type	Description
market ID	Autonumber	
saxeli	Text	
misamarTi	Text	
telefoni	Text	

2. **market ID** ველს მივანიჭოთ პირველადი გასაღები-ველის თვისება, რათა მოხდეს მისი დაკავშირება **masala** ცხრილთან. მაგრამ ეს კავშირი იქნება “მრავალი მრავალთან”, რადგან ერთ სავაჭრო დაწესებულებაში შეიძლება გაიყიდოს სხვადასხვა დასახელების მასალა, ამავე დროს, ერთი დასახელების მასალა შეიძლება გაიყიდოს სხვადასხვა მაღაზიაში. შევქმნათ კონსტრუქტორის რეჟიმში ახალი – გადამკვეთი **gadamkveTi** ცხრილი, რომელშიც ორი გარე გასაღები-ველია **market ID** და **masala ID.**

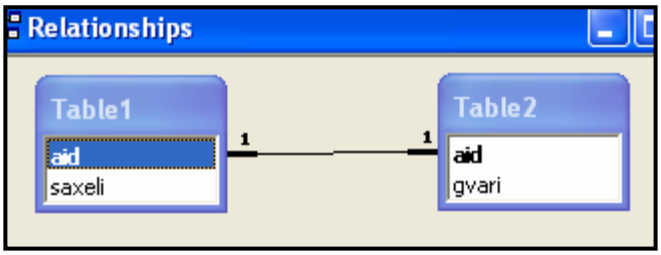
Field Name	Data Type	Description
market ID	Number	
masala ID	Number	

3. შევიტანოთ გადამკვეთ ცხრილში მონაცემები: მასალის კოდები და მაღაზიების კოდები, რომლებიც ასახავენ კონკრეტული მასალის რეალიზაციას კონკრეტულ სავაჭრო დაწესებულებაში.



ნახ. **23**

4. დავაკავშიროთ ეს სამი ცხრილი ერთმანეთთან: გავხსნათ **Relationships** ფანჯარა და **Show Table**-დან გადმოვიტანოთ ცხრილები **masala, gadamkveTi** და **market.** თაგვით შევაერთოთ **masala** და **market** პირველადი გასაღებები **gadamkveTi** ცხრილში შესაბამის გარე გასაღებებთან (ნახ.23).
5. შევამოწმოთ კავშირი, მაგალითად, **masala** ცხრილი გავხსნათ ცხრილის რეჟიმში და დავაწკაპუნოთ + ნიშანზე.
6. შევქმნათ ისეთი ცხრილები, რომლებსაც კავშირი ექნებათ “ერთი ერთთან”. ასეთ კავშირს ვიყენებთ მაშინ, როდესაც ერთი მრავალველიანი ცხრილი უნდა დაიყოს ნაწილებად. ავიღოთ უმარტივესი ცხრილი, რომელშიც, მაგალითად, სახელები და გვარებია. წარმოვადგინოთ ასეთი ცხრილი ორი ცხრილის სახით: ერთში შევიტანოთ მხოლოდ სახელები, მეორეში – მხოლოდ გვარები. ორივეში დავტოვოთ ერთნაირი პირველადი გასაღებები.



ნახ. 24

7. დავაკავშიროთ ეს ცხრილები ერთმანეთთან. კავშირის ტიპი ასე აღინიშნება: **One-To-One** (ნახ.24).

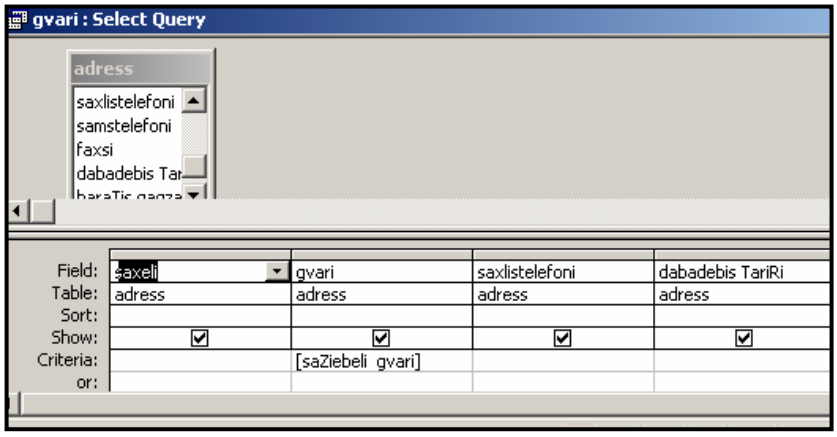
საგარჯიშო 6. მონაცემთა ბაზა “სამისამართო წიგნი”. მოთხოვნის შექმნა მონაცემების ამორჩევაზე

შევქმნათ უბის წიგნაკის ანალოგი ბაზა. ამისათვის სრულიად საკმარისია ერთი ცხრილი, რომელშიც მოცემული იქნება ჩვენთვის საჭირო ინფორმაცია ნებისმიერი პიროვნების შესახებ. ველები, რომლებიც უნდა შეადგენდნენ ასეთ ცხრილს, ჩამოთვლილია ქვემოთ.

1. ჩავტვირთოთ **Microsoft Access (Start→Programs→Microsoft Access).** დიალოგის საწყის ფანჯარაში ჩავრთოთ **Blank Database, OK.** დიალოგის ფანჯარაში **File New Database**, ველში **Save In** მივუთითოთ იმ საქაღალდის სახელი, სადაც შევინახავთ მონაცემთა ბაზის ფაილს, ხოლო ველში **File Name** ჩავწეროთ ფაილის სახელი, ფაილის სახელად ავირჩიოთ **samisamarTo wigni**, **Create.**
2. ბაზის ძირითად ფანჯარაში ჩავრთოთ **Tables** ღილაკი და მივცეთ კონსტრუქტორის რეჟიმში ცხრილის შექმნის ბრძანება **Create tables in Design view.** ცხრილის ველებია:

Field Name	Data Type	Description
kodiID	**Autonumber**	
saxeli	**Text**	
gvari	**Text**	
misamarTi	**Text**	
qalaqi	**Text**	
safostoindeqsi	**Text**	
qveyana	**Text**	
el fosta	**Text**	
saxlistelefoni	**Text**	
samstelefoni	**Text**	
faqsi	**Text**	
dabadebis TariRi	**Text**	
baratis gagzavna	**Boolean**	
SeniSvna	**Memo**	

3. **kodiID** ველს მივანიჭოთ გასაღების თვისება. ველებისთვის **saxlis telefoni** და **samstelefoni** განვსაზღვროთ შეტანის ნიღაბი: ველში **Input Mask** შევიტანოთ გამოსახულება **!00\-00\-00.**
4. შევარჩიოთ მონაცემთა ტიპები და ველების თვისებები ზემოთ აღწერილი მეთოდებით.
5. ცხრილის სტრუქტურის აგების შემდეგ შევინახოთ იგი: **Table name** ველში მივუთითოთ ცხრილის სახელი **misamarTi.**
6. მოვნიშნოთ ცხრილის სახელი და გავხსნათ იგი ჩვეულებრივი წესით **(Open)**, შევავსოთ ველები და შევქმნათ რამდენიმე ჩანაწერი.
7. საჭიროა შეიქმნას მოთხოვნა, რომელიც საბაზისო ცხრილიდან ამოარჩევს მხოლოდ იმ ჩანაწერებს, რომელთა მნიშვნელობა **gvari** ველში ემთხვევა მოთხოვნაში მითითებულს.
8. ჩავტვირთოთ მონაცემთა ბაზა **samisamarTo wigni.** ბაზის ძირითად ფანჯარაში დავაჭიროთ ხელი **Queries** ღილაკს. ფანჯრის მარჯვენა არეში შევარჩიოთ კონსტრუქტორის რეჟიმში მოთხოვნის შექმნის ბრძანება **Create query in Design view.**



ნახ. 25

9. დიალოგის ფანჯარაში **Show Tables** ავარჩიოთ ჩანართი **Tables**, რომელშიც მოვნიშნოთ ცხრილი **misamarTi**. ღილაკით **Add** გადავიტანოთ ცხრილის მაკეტი მოთხოვნის შექმნის ბლანკზე (ნახ.25), **Close.**
10. მოთხოვნის ფანჯრის ზედა ნაწილში მოთავსებული ცხრილის მაკეტში ავარჩიოთ ველები, რომლებიც უნდა ფიგურირებდეს მოთხოვნაში. ისინი რიგრიგობით თაგვით ან ორჯერ დაწკაპუნებით გადავიტანოთ მოთხოვნის ბლანკის ველის **Field** სტრიქონზე. მოთხოვნაში აუცილებლად უნდა გვქონდეს ველი **gvari**, დანარჩენი ველების მოთხოვნაში ჩართვა მოხდება ჩვენი შეხედულებისამებრ.
11. თუ გვაინტერესებს ამა თუ იმ გვარის პიროვნებას როდის აქვს დაბადების დღე და გვჭირდება მისი ტელეფონი, მაშინ მოთხოვნაში გვართან ერთად ჩავრთავთ ველებს **dabadebis TariRi** და **saxlis telefoni.**
12. იმისათვის, რომ ჩანაწერების ამორჩევა მოხდეს გვარების მიხედვით, ველისათვის **gvari, Criteria** სტრიქონზე შევიტანოთ გამოსახულება **[saZiebeli gvari].** ამორჩევის პირობის უჯრაში კვადრატული ფრჩხილების არსებობა უზრუნველყოფს მოთხოვნის შესრულებას წინასწარ მითითებული პარამეტრით, ხოლო ტექსტი, რომელიც მოთავსებულია კვადრატულ ფრჩხილებში, წარმოადგენს პარამეტრის მოთხოვნის ფანჯრის სათაურს.

ნახ. 26

13. დავხუროთ მოთხოვნის კონსტრუქტორის ფანჯარა. სისტემა მოითხოვს ცვლილებების შენახვის დადასტურებას, **Yes**. შემდეგ

ეტაპზე მოთხოვნას მივანიჭოთ სახელი **gvari**. ამ მოქმედებების შესრულების შემდეგ ბაზის ძირითად ფანჯარაში მოთხოვნების ჩამონათვალს დაემატება შექმნილი მოთხოვნა **gvari**.

14. მოვნიშნოთ მოთხოვნა და გავხსნათ ჩვეულებრივი წესით **Open**. ეკრანზე გამონათდება ფანჯარა პარამეტრის მოთხოვნით (ნახ.26). შევიტანოთ ის გვარი, რომლის შესახებ მონაცემების ამოკრებაც გვინდა ძირითადი ცხრილიდან. მოთხოვნის მუშაობის შედეგი იქნება დამაჯამებელი ცხრილი, რომელიც შეიცავს მოთხოვნაში ჩვენს მიერ ჩართულ ველებს და მხოლოდ იმ ჩანაწერებს, რომელიც შეიცავს პარამეტრად მითითებულ გვარს.

სავარჯიშო 7. მონაცემთა ბაზა "სამისამართო წიგნი", მოთხოვნიდან ჩანაწერების ამოკრება გარკვეული პირობებით

შევქმნათ მოთხოვნა, რომელიც საბაზისო ცხრილიდან **misamarTi** ამოარჩევს მხოლოდ იმ ჩანაწერებს, რომელთა ველში **dabadebis TariRi** თვის და დღის რიცხვითი მნიშვნელობები ემთხვევა კომპიუტერის კალენდარულ თარიღს (ე.ი. მიმდინარე დღეს აქვს დაბადების დღე). აგრეთვე შეიქმნას მოთხოვნები, რომელშიც იქნება გამოთვლილი პიროვნების წელთა რაოდენობა და მოთხოვნა, რომელიც საბაზისო ცხრილიდან ამოარჩევს მხოლოდ იმ ჩანაწერებს (შესაბამისი პიროვნებების გვარებს), რომლებსაც ჩვენს მიერ მითითებულ თვეში აქვთ დაბადების დღე და საჭიროა მისალოცი ბარათის გაგზავნა. ჩავტვირთოთ მონაცემთა ბაზა **samisamarTo wigni.** მოთხოვნის შესაქმნელად შევასრულოთ წინა სამუშაოს მე-8 და მე-9 პუნქტები.

1. პირველი მოთხოვნის შესაქმნელად გადმოვიტანოთ ცხრილის შემდეგი ველები: **gvari, saxeli** და **dabadebis TariRi.** შეიძლება სხვა ველების შეტანაც.

2. **dabadebis TariRi** ველის შესაბამის **Criteria** სტრიქონზე შევიტანოთ გამოსახულება, რომელიც აღნიშნულ ველში მითითებულ თარიღს შეადარებს კომპიუტერის მიმდინარე კალენდარულ თარიღს. ამ გამოსახულებას ასეთი სახე ექნება: **Day(Date())=Day([misamarTi]![dabadebisTariRi]) and month(Date())=month([misamarTi]![dabadebisTariRi]).** ფუნქცია **Date()** ითვლის მიმდინარე თარიღის რიცხვით მნიშვნელობას. პირველი ტოლობა უზრუნველყოფს ჩანაწერის დღის რიგითი ნომრის დამთხვევას მიმდინარე თარიღის დღის ნომერთან, ხოლო მეორე ტოლობა – ჩანაწერის თვის ნომრის ტოლობას მიმდინარე თვის ნომერთან.

3. დავხუროთ ფანჯარა და მივანიჭოთ მოთხოვნას სახელი **dabadebis dRe.** გამოსახულების ჩასაწერად შეგვიძლია **Built** პროგრამის გამოყენება.

4. შექმნილი მოთხოვნა გავხსნათ ჩვეულებრივი წესით. მივიღებთ დამაჯამებელ ცხრილს ჩანაწერებით, რომლებიც შეეხება მიმდინარე დღეს დაბადებულ პიროვნებებს.

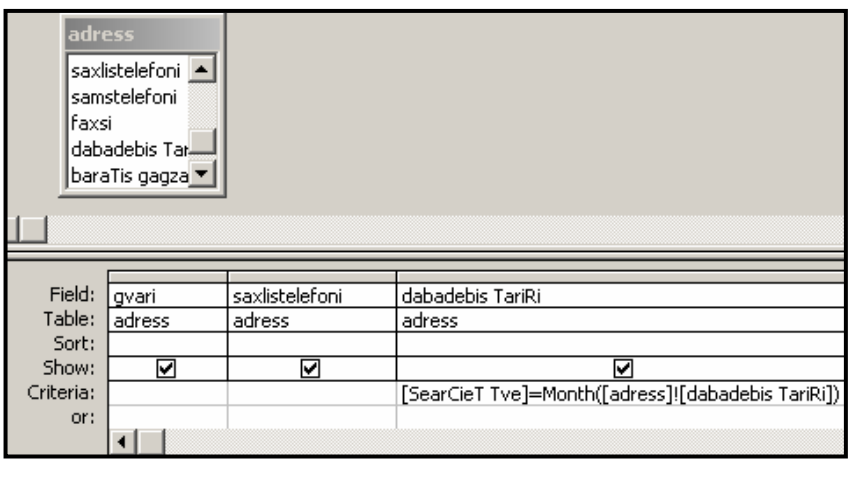
5. შევქმნათ მეორე მოთხოვნა, რომელშიც გამოთვლილი იქნება პიროვნების წელთა რაოდენობა, ამისათვის მოთხოვნაში შევიტანოთ ველები: **saxeli** და **dabadebis TariRi**. დავამატოთ ველი, რომელშიც ჩავწეროთ გამოსახულება **Built** პროგრამის გამოყენებით: **Year(Date())-Year([misamarTi]![dabadebis TariRi]).** დავათვალიეროთ დამაჯამებელი ცხრილი.

6. შევქმნათ მესამე მოთხოვნა, რომელიც ამოკრებს ჩანაწერებს იმ პიროვნებების შესახებ, რომელთა დაბადების დღეც მითითებულ თვეშია და საჭიროა მისალოცი ბარათის გაგზავნა. ამისათვის დაგვჭირდება ველები: **gvari dabadebis TariRi და baraTis gagzavna**

7. ველში **dabadebis TariRi** ამორჩევის პირობა უნდა იყოს თარიღის თვის რიცხვითი მნიშვნელობის და წინასწარ მითითებული პარამეტრის (რომელიმე თვის ნომერი) ტოლობა. ამისათვის ამ ველის შესაბამის **Criteria** სტრიქონზე შევიტანოთ გამოსახულება **[SearCieT Tve]= Month([misamarTi]![dabadebis TariRi])** (ნახ.27). კვადრატული ფრჩხილები გამოსახულების თავში უზრუნველყოფს მოთხოვნის პარამეტრის ფანჯრის გამოტანას, ხოლო ფრჩხილებში მოთავსებული ტექსტი **[SearCieT Tve]** – ამ ფანჯრის დასათაურებას. ტოლობის ნიშნის მარჯვენა ნაწილში ხდება თვის ნომრის პარამეტრთან ტოლობის შემოწმება. გარდა წინა პირობისა, ჩანაწერები უნდა აკმაყოფილებდეს შემდეგ პირობასაც: ველს **baratis gagzavna** უნდა ჰქონდეს ჭეშმარიტი მნიშვნელობა. ამისათვის ამ ველის შესაბამის **Criteria** სტრიქონზე შევიტანოთ **Yes.**

8. დავხუროთ კონსტრუქტორის ფანჯარა და მოთხოვნას მივანიჭოთ სახელი **milocva**.

ნახ. 27



9. შექმნილი მოთხოვნა გავხსნათ ჩვეულებრივი წესით, პარამეტრის მოთხოვნის ფანჯრის ტექსტურ ველში შევიტანოთ რაიმე რიცხვი 1-დან 12-მდე (თვის ნომერი). დავათვალიეროთ დამაჯამებელი ცხრილი.

სავარჯიშო 8. მონაცემთა ბაზა "ღვინის სარდაფი"

როგორც ვიცით, მონაცემთა ბაზის შესაქმნელად სრულად უნდა იყოს გააზრებული, რა სახის ინფორმაციის შენახვა და შემდგომში გაცემა მოითხოვება ბაზისაგან. ამ შემთხვევაში ბაზაში სასურველია შეტანილი იყოს ინფორმაცია სარდაფში არსებული პროდუქციის შესახებ (ცნობები პროდუქციის ყოველ ერთეულზე), და მონაცემები ამ პროდუქციის შესყიდვაზე (რაოდენობა, თარიღი და ა.შ.). აქედან გამომდინარე, გვჭირდება ორი ცხრილი: ღვინის ჩამონათვლის და მისი შესყიდვის. ღვინის ჩამონათვალში ყოველი ერთეულისათვის განისაზღვრება ღვინის ტიპი (დასახელება, ფერი, ტკბილია თუ მშრალი, მოსავლის წელი და ა.შ.). მაგრამ აქ გვერდს ვერ ავუვლით მონაცემების გამეორებას, რაც არ არის სასურველი. ამიტომ დავამატებთ მესამე ცხრილს, რომელშიც ჩამოთვლილი იქნება მხოლოდ ღვინის ტიპები. ბაზაში აგრეთვე შევქმნით მოთხოვნებს.

1. ბაზის შექმნა დავიწყოთ ფაილის განსაზღვრით, ამისათვის შევასრულოთ მე-6 სავარჯიშოს 1-ლი და მე-2 პუნქტები, **File name** ველში შევიტანოთ ფაილის სახელი **Rvinis sardafi**.
2. კონსტრუქტორის რეჟიმში რიგრიგობით შევქმნათ სამი ცხრილის სტრუქტურა. ღვინის ტიპის **(Rvinis tipi)** ცხრილის ველებია:

Field Name	Data Type	Description
Rvinis tipisID	**Autonumber**	
Rvinis tipi	**Text**	
feri	**Text**	

3. დანარჩენ ცხრილებთან კავშირების დასამყარებლად პირველი ველი განვსაზღვროთ, როგორც პირველადი გასაღები. ამისათვის მოვნიშნოთ იგი და შევასრულოთ ბრძანება **Edit→Primary Key.**
4. რადგან ღვინის ფერის მნიშვნელობების მხოლოდ სამი ვარიანტი არსებობს, **feri** ველის შევსებისათვის გამოვიყენოთ ჩანაცვლების მეთოდი. მოვნიშნოთ ველი და **Field Properties → Lookup** ჩანართში შევარჩიოთ შემდეგი თვისებები: **Display Control – List Box, Row Source Type – Value List, Row Source –** შევიტანოთ ფერების ჩამონათვალი, რომლებიც ერთმანეთისაგან წერტილმძიმით იქნებიან გამოყოფილი: **TeTri; wiTeli; Savi**.
5. ჩანაცვლების მეთოდი შეიძლება გამოვიყენოთ აგრეთვე ველისთვის **Rvinis tipi,** ან სახეობები მშრალი, ტკბილი და სხვა შევიტანოთ პირდაპირ ცხრილში.
6. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ ცხრილს სახელი **Rvinis tipi.**
7. გავხსნათ ცხრილი ჩვეულებრივი წესით **Open**, შევავსოთ მონაცემებით – შევქმნათ რამდენიმე ჩანაწერი.
8. მეორე ცხრილის – ღვინის ჩამონათვლის **(Rvinis CamonaTvali)** ველებია:

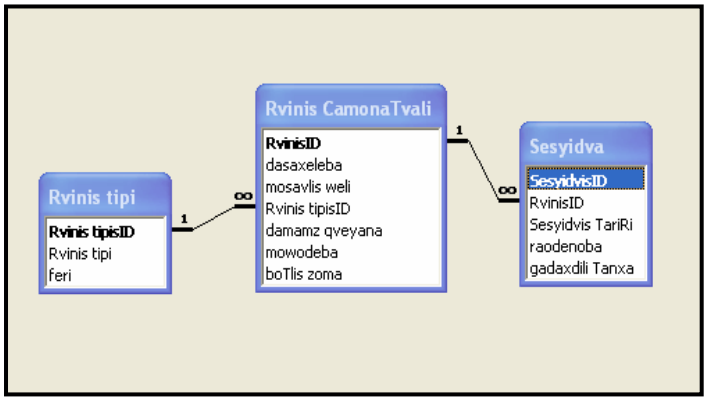
Field Name	Data Type	Description
RvinisID	**Autonumber**	
dasaxeleba	**Text**	
mosavlis weli	**Number**	
Rvinis tipisID	**Number**	
damamz.qveyana	**Text**	
mowodeba	**Memo**	
boTlis zoma	**Number**	

9. ცხრილის პირველი ველი **RvinisID** განვსაზღვროთ, როგორც პირველადი გასაღები-ველი.
10. ცხრილის შევსებისას ველისთვის **RvinistipisID** გამოვიყენოთ ჩანაცვლების მეთოდი – მასში ჩასაწერი მონაცემები (კოდები) ავარჩიოთ უკვე არსებული **Rvinis tipi** ცხრილიდან. ამისათვის მოვნიშნოთ ველი და **Field Properties → Lookup** ჩანართში შევარჩიოთ შემდეგი თვისებები: **Display Control – Combo Box, Row Source Type – Table/Query, Row Source –** ამოვარჩიოთ ცხრილი **Rvinis tipi, Bound Column – 1, Column Count – 3, Column Heads –Yes, Column Widths – 1, List Widths – 3,** დანარჩენი თვისებები დავტოვოთ უცვლელად.
11. ველისათვის **boTlis zoma** თვისებების **Field Properties** ჩანართში **General** შევარჩიოთ შემდეგი თვისებები: **Field Size—Single, Decimal Places -- 2**. ეს უზრუნველყოფს რიცხვის წარმოდგენას წილადის სახით ისე, რომ ათობითი წერტილის შემდეგ გამოტანილი იქნას ორი ციფრი.
12. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ ცხრილს სახელი **Rvinis CamonaTvali.**
13. გავხსნათ ცხრილი ჩვეულებრივი წესით **(Open)**, შევავსოთ მონაცემებით – შევქმნათ რამოდენიმე ჩანაწერი.
14. მესამე ცხრილის – შესყიდვის **(Sesyidva)** ველებია:

Field Name	Data Type	Description
SesyidvisID	**Autonumber**	
RvinisID	**Number**	
Sesyidvis TariRi	**Date/Time**	
raodenoba	**Number**	
gadaxdili Tanxa	**Currency**	

15. ცხრილის პირველი ველი **SesyidvisID** განვსაზღვროთ, როგორც პირველადი გასაღები-ველი .

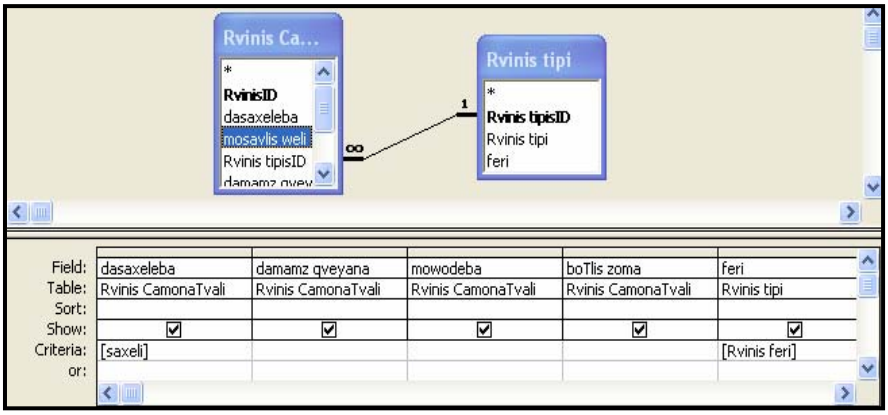
16. **RvinisID** ველის შევსებისას შესაძლებელია ჩანაცვლების მეთოდის გამოყენება – მონაცემები (კოდები) ავარჩიოთ უკვე არსებული ცხრილიდან **Rvinis CamonaTvali**. მოვნიშნოთ ველი და **Field Properties Lookup** ჩანართში შევარჩიოთ შემდეგი თვისებები: **Display Control – Combo Box, Row Source Type – Table/Query, Row Source** – ამოვარჩიოთ ცხრილი **Rvinis CamonaTvali, Bound Column – 1, Column Count – 7, Heads – Yes, Column Widths –1", List Width – 7",** დანარჩენი თვისებები დავტოვოთ უცვლელი.
17. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ ცხრილს სახელი **Sesyidva**.
18. გავხსნათ ცხრილი ჩვეულებრივი წესით, შევავსოთ მონაცემებით – შევქმნათ რამოდენიმე ჩანაწერი.
19. დავამყაროთ ცხრილებს შორის კავშირი (ნახ.28).



ნახ. 28

20. შევქმნათ მოთხოვნა, რომელიც ამოარჩევს პროდუქციას დასახელების და ფერის მიხედვით: საჭიროა შესრულდეს **Rvinis CamonaTvali** ცხრილიდან იმ პროდუქციის ამორჩევა, რომელიც მითითებული ფერის და ტიპის იქნება, ანუ მოთხოვნაში ველებისათვის **Rvinistipi** და **feri** შევასრულოთ მოთხოვნა პარამეტრით.

ნახ.29



21. მოთხოვნის ასაგებად დაგვჭირდება ორი ცხრილი **Rvinis tipi** და **Rvinis CamonaTvali**. გავხსნათ მოთხოვნა კონსტუქტორის რეჟიმში **Create query in Design view** და დიალოგის ფანჯარაში **Show Tables**, ავარჩიოთ ჩანართი **Tables**, ზემოთ აღნიშნული ცხრილები გადავიტანოთ მოთხოვნის შექმნის ფანჯარაში. ფანჯრის ქვედა ნაწილში ჩამოვიტანოთ ველები **Rvinis tipi, feri** და **dasaxeleba,** შესაძლებელია სხვა ველების ჩამოტანაც (ნახ.29).
22. ველში **Criteria** არსებული ჩანაწერი უზრუნველყოფს ძებნას პარამეტრებით. კვადრატულ ფრჩხილებში მითითებული ტექსტები **[saxeli]** და **[Rvinis feri]** წარმოადგენს პარამეტრების ფანჯრების სათაურებს. ეს ტექსტები არ უნდა იყოს ველების სათაურების ზუსტი ასლები (ერთი ცარიელი სიმბოლოთი მაინც უნდა განსხვავდებოდეს). დავხუროთ კონსტრუქტორის ფანჯარა და მოთხოვნას მივანიჭოთ სახელი **amorCeva**.
23. გავხსნათ მოთხოვნა ჩვეულებრივი წესით, შევავსოთ ჯერ ღვინის სახელის პარამეტრის მოთხოვნის ტექსტური ველი, ხოლო შემდეგ ღვინის ფერის პარამეტრის. დავათვალიეროთ დამაჯამებელი ცხრილი.

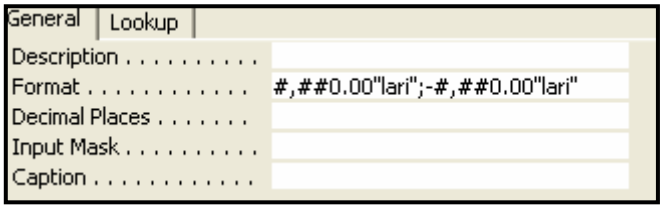
საგარჯიშო 9. მონაცემთა ბაზა "ღვინის სარდაფი". მოთხოვნების შექმნა

შევქმნათ ისეთი მოთხოვნა, რომელშიც იქნება ჩასატარებელი გამოთვლები, მაგალითად, შევქმნათ მოთხოვნები "პროდუქციის ერთეულის ფასი" და "საშუალო ფასი". ამოცანის შინაარსი ასეთია: ერთი და იმავე **ID** კოდის მქონე პროდუქციის პარტია შესაძლებელია სხვადასხვა დროს სხვადასხვა ფასად იქნას შესყიდული. შესაბამისად, ერთი და იმავე დასახელების პროდუქციისათვის, რომელიც სხვადასხვა დროს არის შესყიდული და ე.ი. ერთმანეთისაგან მხოლოდ **SesyidvisID** კოდით განსხვავდებიან, ერთეულის ფასი იქნება განსხვავებული.

1. ჩავტვირთოთ ბაზა "ღვინის სარდაფი", შევქმნათ მოთხოვნა კონსტუქტორის რეჟიმში **Create query in Design view.** მოთხოვნის შექმნის ფანჯარაში შევიტანოთ ცხრილები **Sesyidva** და **Rvinis CamonaTvali. Field** ველში

გადმოვიტანოთ ღვინის ჩამონათვლის ცხრილიდან **RvinisID** და **dasaxeleba** ველები, შესყიდვის ცხრილიდან – **SesyidvisID, raodenoba, dagaxdili Tanxa.**

2. ერთეულის ფასი გამოიანგარიშება პარტიაში გადახდილი თანხის (ველის **gadaxdili Tanxa**) რაოდენობასთან (ველთან **raodenoba**) შეფარდებით. ამისათვის შემოგვაქვს ველი **erTeulis fasi**, რომელშიც ჩავწერთ გამოსახულებას: გადახდილი თანხა, გაყოფილი რაოდენობაზე, ანუ **Field** ველში შევიტანთ გამოსახულებას: **erTeulis fasi:([Sesyidva]![gadaxdili Tanxa]/ [Sesyidva]![raodenoba]).** აქ შეიძლება **Built** პროგრამის გამოყენებაც.
3. როგორც ვიცით, ცხრილის ველების დაფორმატება ხდება ცხრილების აგების პროცესში. ველი **erTeulis fasi** არც ერთ ჩვენს მიერ შექმნილ ცხრილში არ ფიგურირებს და, ე.ი. არ არის დაფორმატებული. ამ ველის დასაფორმატებლად შევასრულოთ შემდეგი მოქმედებები: დავაყენოთ კურსორი **Field** ველის **erTeulis fasi** უჯრაში და მენიუს **View → Properties (**ან კონტექსტური მენიუდან მარჯვენა ღილაკით), **General** ჩანართის **Format** ველში ავარჩიოთ ველის დაფორმატების ტიპი (ნახ.30).



ნახ.30

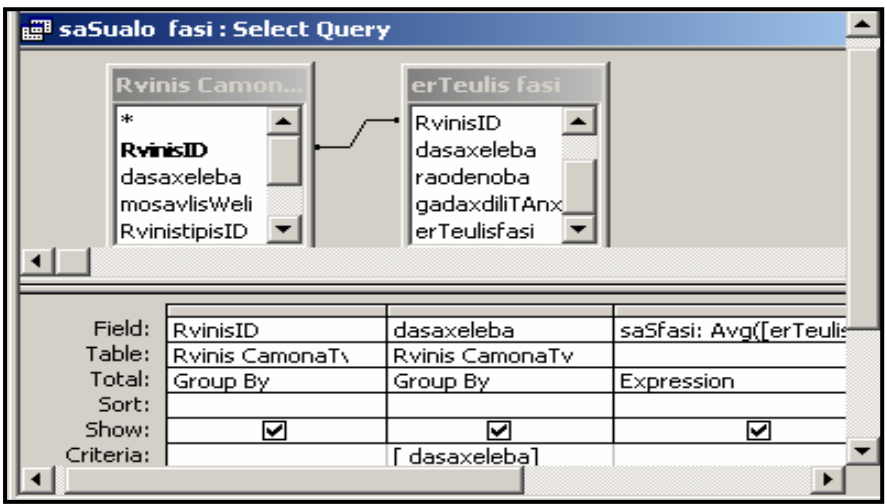
4. დავხუროთ დიალოგის ფანჯარა და მოთხოვნას მივანიჭოთ სახელი **erTeulis fasi.** გავხსნათ მოთხოვნა ჩვეულებრივი წესით, დავათვალიეროთ დამაჯამებელი ცხრილი (ნახ.31).

SesyidvisID	RvinisID	dasaxeleba	raodeno	gadaxdili Tan	erTeulis fasi
1	1	manavi	600	4,052.00lari	6.75lari
2	2	nafareuli	500	3,150.00lari	6.30lari
3	5	mukuzani	400	2,006.00lari	5.02lari
4	5	mukuzani	500	3,010.00lari	6.02lari
5	3	saferavi	600	5,060.00lari	8.43lari
6	3	saferavi	500	4,600.00lari	9.20lari
8	5	mukuzani	300	1,800.00lari	6.00lari
9	4	xvanWkara	800	6,080.00lari	7.60lari
10	6	nafareuli	500	4,005.00lari	8.01lari
(AutoNumber)	(utoNumber)				

ნახ. 31

5. შევქმნათ მოთხოვნა “საშუალო ფასი”. როგორც წინა მოთხოვნის შექმნისას დავინახეთ, ერთი და იმავე კოდის მქონე პროდუქციას შეიძლება სხვადასხვა ფასი ჰქონდეს. მიუხედავად ამისა, ამ პროდუქციისათვის შესაძლებელია საშუალო ფასის გამოანგარიშება, რომელიც იქნება ერთი და იმავე **ID**-ის მქონე პროდუქციის სხვადასხვა ფასების საშუალო არითმეტიკული.
6. გავხსნათ მოთხოვნის კონსტუქტორის რეჟიმი **Create query in Design view.** მოთხოვნის შექმნის ფანჯარაში შევიტანოთ ცხრილი **Rvinis CamonaTvali.** შემდეგ მენიუს **View → Show Tables** პუნქტით გამოვიტანოთ ფანჯარა **Show Tables**, ავარჩიოთ ჩანართი **Queries** და დავამატოთ მოთხოვნა **erTeulis fasi.**
7. ჩამონათვლის ცხრილიდან **Field** ველში გადმოვიტანოთ **RvinisID** და **dasaxeleba** ველები.
8. შემოვიტანოთ ახალი ველი **saS fasi,** რომელიც იქნება ერთნაირი კოდის მქონე დასახელებების ფასების საშუალო არითმეტიკული. ამისათვის **field** ველში შევიტანოთ გამოსახულება: **saSfasi:Avg([erTeulis fasi]! erTeulis fasi) (**ნახ.32**).**
9. წინა მოთხოვნაში აღწერილი გზით დავაფორმატოთ ველი **saSfasi,** როგორც ფულადი ტიპის მონაცემი **(Currency).**

10. იმისათვის, რომ საშუალო ფასის ამორჩევა მოხდეს კონკრეტული დასახელების პროდუქციისათვის, ველი **dasaxeleba** გამოვიყენოთ ამორჩევის კრიტერიუმის მისათითებლად. ამ ველის **Criteria** უჯრაში შევიტანოთ პარამეტრის მოთხოვნის გამოსახულება **[dasaxeleba].** აქ პარამეტრის სათაურს წინ დაურთეთ ცარიელი პოზიცია, რათა ველის სახელისაგან იყოს განსხვავებული.



ნახ. 32

11. დავხუროთ დიალოგის ფანჯარა და მოთხოვნას მივანიჭოთ სახელი **saSfasi.**
12. გავხსნათ მოთხოვნა ჩვეულებრივი წესით, შევავსოთ პარამეტრი **dasaxeleba** და დავათვალიეროთ დამაჯამებელი ცხრილი.

სავარჯიშო 10. მონაცემთა ბაზა “სამკერვალო”

მონაცემთა ბაზა “სამკერვალო” უნდა შეიცავდეს შემდეგი სახის ინფორმაციას: თანამშრომლების ვინაობა (სახელი, გვარი და ა.შ.), დაწესებულებაში არსებული სამუშაოთა სახეობები, ანაზღაურების ოდენობა და სხვ. ბაზის მეშვეობით აგრეთვე უნდა არსებობდეს ისეთი ინფორმაციის მიღების საშუალება, როგორიცაა, თითოეული თანამშრომლის მიერ შესრულებული სამუშაოს ოდენობა, სახეობა, დროის შუალედი და ა.შ. ბაზის სასწავლო დანიშნულებიდან გამომდინარე, ამოცანაში ზოგიერთი საკითხი გამარტივებულია, კერძოდ, დაშვებულია პირობა, რომ ერთ კონკრეტულ შეკვეთას მხოლოდ ერთი თანამშრომელი ასრულებს. ბაზის შესაქმნელად ჩამოვაყალიბოთ სამი ცხრილი: შესაძლო შეკვეთების ჩამონათვალი – შემოთავაზებული მომსახურება, თანამშრომლების შესახებ ინფორმაცია და მონაცემები შესრულებული სამუშაოს შესახებ.

ბაზის ასაგებად შევქმნათ ფაილი სახელით **samkervalo**. ცხრილები შევქმნათ კონსტრუქტორის რეჟიმში:

1. დავიწყოთ ახალი ბაზის შექმნა, ფაილს მივანიჭოთ სახელი **samkervalo**. შევქმნათ პირველი ცხრილი, სტრუქტურის ბლანკში შევიტანოთ ცხრილის შემოთავაზებული მომსახურება **(SemoTavazebuli momsaxureba)** ველების სახელები: დასახელება, ანაზღაურება და შესრულებისათვის საჭირო დღეები. შევარჩიოთ შესატანი მონაცემების ტიპები:

Field Name	Data Type	Description
momsID	Autonumber	
dasaxeleba	Text	
anazRaureba	Currency	
Sesr saWiro dReebi	Number	

2. პირველი ველი განვსაზღვროთ, როგორც პირველადი გასაღები, ამისათვის მოვნიშნოთ იგი და ჩავრთოთ **Edit→Primary Key.**
3. შევინახოთ ცხრილი და კონსტრუქტორის რეჟიმიდან გადავიდეთ ცხრილის რეჟიმში – შევიტანოთ რამდენიმე ჩანაწერი.
4. შევქმნათ მეორე ცხრილი, სტრუქტურის ბლანკში შევიტანოთ ცხრილის – თანამშრომლების **(TanamSromlebi)** ველების სახელები და შევარჩიოთ შესატანი მონაცემების ტიპები:

Field Name	Data Type	Description
TanamSromlisID	Autonumber	
gvari	Text	
saxeli	Text	

5. გავიმეოროთ მე-2 პუნქტი, შევინახოთ ცხრილი და კონსტრუქტორის რეჟიმიდან გადავიდეთ ცხრილის რეჟიმში – შევიტანოთ რამდენიმე ჩანაწერი.
6. შევქმნათ მესამე ცხრილი, სტრუქტურის ბლანკში შევიტანოთ ცხრილის – შესრულებული სამუშაოს **(Sesrulebuli samuSao)** ველების სახელები და შევარჩიოთ შესატანი მონაცემების ტიპები:

Field Name	Data Type	Description
SekveTisID	Autonumber	
momsID	Number	
TanamSromlisID	Number	
SekveTis TariRi	Date/Time	

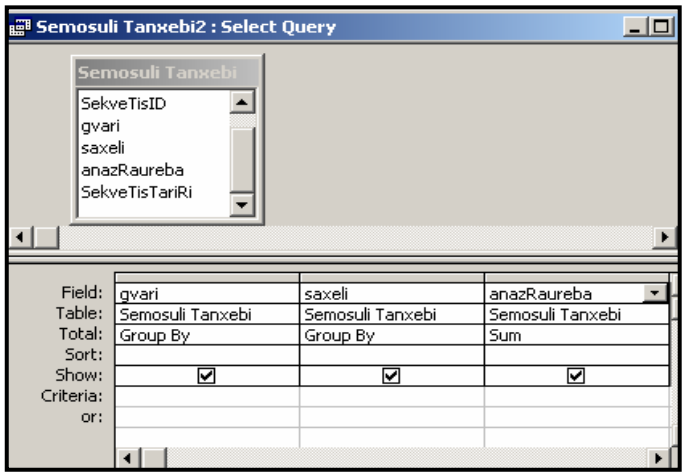
7. გავიმეოროთ მე-5 პუნქტი. ცხრილში ველის **momsID** შევსების გამარტივების მიზნით გამოვიყენოთ ჩანაცვლების მეთოდი – მონაცემები ავარჩიოთ არსებული **SemoTavazebuli momsaxureba** ცხრილიდან**.** ამისათვის კონსტრუქტორის რეჟიმში მოვნიშნოთ **momsID** ველი და **Field Properties → Lookup** ჩანართში შევარჩიოთ შემდეგი თვისებები: **Display Control – Combo Box; Row Source Type – Table/Query; Row Source –** ცხრილი **SemoTavazebuli momsaxureba; Bound Column – 1; Column Count – 3; Column Widths – 1”, List Width – 3”.** დანარჩენი თვისებები დავტოვოთ უცვლელად.
8. იმავე წესით გამოვიყენოთ ჩანაცვლების მეთოდი ველისათვის **TanamSromlisID,** წყაროდ ავირჩიოთ ცხრილი **TanamSromlebi.**
9. შევინახოთ ცხრილი და კონსტრუქტორის რეჟიმიდან გადავიდეთ ცხრილის რეჟიმში – შევიტანოთ რამდენიმე ჩანაწერი.
10. დავამყაროთ ცხრილებს შორის კავშირი**.** ამისათვის: **Tools→Relationships, Show Table** ფანჯრის **Tables** ჩანართში მოვნიშნოთ სამივე ცხრილი, **Add**. დავხუროთ დიალოგის ფანჯარა, **Close**. **Relationships** ფანჯარაში ცხრილი **SemoTavazebuli momsaxureba** პირველადი გასაღები ველით დავაკავშიროთ ცხრილ **Sesrulebuli samuSao-**ს გარე გასაღებ-ველთან **momsID**. გამონათებულ **Edit Relationships** ფანჯარაში ჩავრთოთ შესაბამისი ალმები, **Create.** ანალოგიურად დავაკავშიროთ ცხრილი **TanamSromlebi** გასაღები ველით ცხრილთან **Sesrulebuli samuSao** გარე გასაღებ-ველთან **TanamSromlisID.** დავხუროთ ფანჯარა, **Yes.**

11. შევქმნათ მოთხოვნა, რომელიც განსაზღვრავს შეკვეთის შესრულების ვადას. როგორც უკვე ვნახეთ, ცხრილების შექმნისას ყოველი შეკვეთისათვის განსაზღვრულია შესასრულებლად საჭირო დღეთა რაოდენობა, ე.ი. თუ ვიცით შეკვეთის მიღების დღე, შეგვიძლია განვსაზღვროთ შეკვეთის დამთავრების თარიღიც. მოთხოვნის შესაქმნელად გამოვიყენოთ სამივე ცხრილი: გავხსნათ მოთხოვნა კონსტრუქტორის რეჟიმში (**Create query in Design view)** და დიალოგის ფანჯრიდან **Show Tables** ჩანართ **Tables**-ში მოვნიშნოთ სამივე ცხრილი, **Add**. ბლანკზე თაგვით **Fields** სტრიქონზე ჩამოვიტანოთ ველები **SekveTisID (**ცხრილი **Sesrulebuli samuSao)** და **dasaxeleba (**ცხრილი **SemoTavazebuli momsaxureba).**
12. მოთხოვნის **Fields** სტრიქონზე დამატებით სვეტში შევქმნათ ველი **Semsrulebeli** ორი ველის **gvari** და **saxeli** (ცხრილიდან **TanamSromlebi)** გაერთიანების გზით. ამისათვის ახალ სვეტში შევიტანოთ გამოსახულება: **Semsrulebeli: [TanamSromlebi]![saxeli] & " " & [TanamSromlebi]![gvari].** გამოსახულების ჩასაწერად შეგვიძლია აგრეთვე **Built** პროგრამის გამოყენებაც.
13. ამავე მოთხოვნაში შევქმნათ კიდევ ერთი ველი – შეკვეთის ჩაბარების ვადა **(SekveTis Cabarebis vada).** ამ ველის მნიშვნელობა გამოითვლება შეკვეთის მიღების თარიღზე შეკვეთის შესასრულებლად საჭირო დღეების დამატებით. ველში **SekveTis Cabarebis vada** შევიტანოთ გამოსახულება: **SekveTis Cabarebis vada: DateAdd(“d”, [SemoTavazebulimomsaxureba]![SesrsaWiro- dReebi], [SesrulebulisamuSao]![SekveTisTariRi]).** გამოსახულების ჩასაწერად შეგვიძლია **Built** პროგრამის გამოყენება. გამოსახულებაში ჩართულია ფუნქცია **DateAdd,** მისი ზოგადი ჩანაწერია: **DateAdd («interval», «number», «date»).** არგუმენტის **«interval»**-ში მივუთითებთ, თუ რა ინტერვალი ემატება შეკვეთის თარიღს: დღე, თვე თუ წელი. ჩვენს შემთხვევაში უნდა დაემატოს დღეები, ამიტომ ჩავწეროთ **"d"** (თვეების დამატების შემთხვევაში ჩავწერთ **“mm”,** წლების - **“yyyy”), «number»** ნაწილში შევიტანთ დასამატებელ რიცხვს, ამ შემთხვევაში მივუთითებთ ველს **SesrsawirodReebi** ცხრილიდან **SemoTavazebuli samuSao, «date»** ნაწილში შევიტანთ იმ თარიღის მნიშვნელობას, რომელზეც ხდება ამა თუ იმ ინტერვალის დამატება – ველს **SekveTisTariRi** ცხრილიდან **Sesrulebuli samuSao.**
14. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ მოთხოვნას სახელი **SekveTis Cabareba.** გავხსნათ მოთხოვნა ჩვეულებრივი წესით, დავათვალიეროთ დამაჯამებელი ცხრილი.

სავარჯიშო 11. მონაცემთა ბაზა “სამკერვალო”. სხვადასხვა ხასიათის მოთხოვნების შექმნა

მონაცემთა ბაზაში “სამკერვალო” შევქმნათ შემდეგი სახის მოთხოვნები: პირველში მოხდეს შეკვეთების შესრულებიდან შემოსული მთლიანი თანხის დათვლა კონკრეტულ თვეში, მეორეში - შერჩეული თანამშრომლის მიერ კონკრეტულ თვეში ყველა შეკვეთის შესრულებით შემოტანილი მთლიანი თანხა, მესამეში – თითოეული თანამშრომლის მიერ გამომუშავებული ხელფასი.

1. პირველი მოთხოვნისათვის დაგვჭირდება შემდეგი ველები: ცხრილიდან **SesrulebulisamuSao – SekveTisID** და **SekveTisTariRi,** ცხრილიდან **TanamSromlebi – gvari, saxeli,** ცხრილიდან **SemoTavazebulimomsaxureba – anazRaureba.**შევქმნათ მოთხოვნა კონსტუქტორის რეჟიმში **(Create query in Design view)** და დიალოგის ფანჯრიდან **Show Tables** გადავიტანოთ სამივე ცხრილი მოთხოვნის ფანჯარაში. ზემოთ აღნიშნული ველები თაგვით სათითაოდ გადმოვიტანოთ მოთხოვნის ბლანკზე.
2. ახალი სვეტის **Criteria** სტრიქონზე შევიტანოთ გამოსახულება **[Tve]=Month([SesrulebulisamuSao]![SekveTisTariRi]),** რათა მოთხოვნა შესრულდეს პარამეტრით. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ მოთხოვნას სახელი **Semosuli Tanxebi.**
3. გავხსნათ მოთხოვნა ჩვეულებრივი წესით, მივუთითოთ პარამეტრი, ანუ თვის რიგითი ნომერი და დავათვალიეროთ დამაჯამებელი ცხრილი.

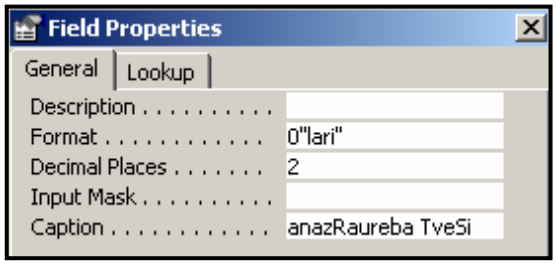


ნახ. 33

4. მეორე მოთხოვნა უნდა შეიქმნას უკვე არსებული მოთხოვნის **Semosuli Tanxebi** საფუძველზე. შევქმნათ მოთხოვნა კონსტრუქტორის რეჟიმში **(Create query in Design view)** და დიალოგის ფანჯარაში **Show Tables** ავარჩიოთ ჩანართი **Query** ჩამონათვალიდან – მოთხოვნა **Semosuli Tanxebi.** მოთხოვნის ბლანკზე აღნიშნული მოთხოვნის მაკეტიდან **Fields** სტრიქონზე გადავიტანოთ ველები **gvari, saxeli** და **anazRaureba** (ნახ.33).

5. ჩავრთოთ ჯგუფური ოპერაციები იარაღების პანელზე არსებული Σ ღილაკით. ველებისათვის **gvari** და **saxeli** დავტოვოთ (**Group By**), ხოლო ველისათვის **anazRaureba** – ავარჩიოთ დაჯამების ოპერაცია (**Sum**). ვინაიდან ველის "ანაზღაურება" გამოთვლა ხდება მოთხოვნაში, საჭიროა მისი ფორმატის განსაზღვრა. ამისათვის შევასრულოთ შემდეგი მოქმედებები: ველში **anazRaureba** დავაჭიროთ ხელი მენიუს პუნქტს **Properties**, **General**

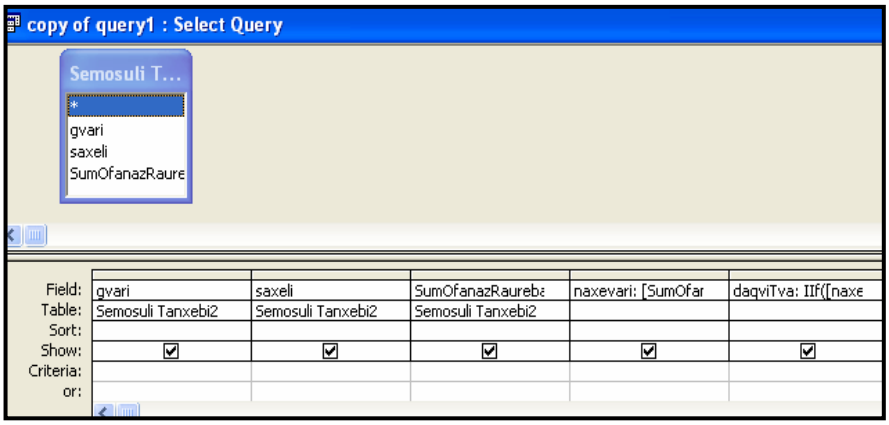
ნახ. 34



ჩანართის **Format** ველში ავარჩიოთ 0"lari" ფორმატი, **Decimal Places**-ში 2, **Caption** ველში შევიტანოთ ველის სახელი "ანაზღაურება თვეში" (ნახ.34).

6. დავხუროთ კონსტრუქტორის ფანჯარა და მივანიჭოთ მოთხოვნას სახელი **Semosuli Tanxebi2** .
7. შევასრულოთ მოთხოვნა ჩვეულებრივი წესით, პარამეტრის ფანჯარაში (სისტემა აუცილებლად მოითხოვს პარამეტრს, რადგან ეს მოთხოვნა აგებულია ისეთ მოთხოვნაზე, რომლის შესრულება საჭიროებს პარამეტრის მითითებას), შევიტანოთ კონკრეტული თვის რიცხვითი მნიშვნელობა და დავათვალიეროთ მიღებული ცხრილი.
8. მესამე მოთხოვნაში გამოთვლილი უნდა იყოს თითოეული თანამშრომლის მიერ შემოტანილი თანხის ოდენობის საფუძველზე გამოანგარიშებული ხელფასის მნიშვნელობა (აქ ხელფასის გამოსათვლელი ფორმულები მაქსიმალურად გამარტივებულია). ჩავთვალოთ, რომ შემოტანილი თანხის ნახევარი გამოიწერება ხელფასის სახით. გამოწერილი თანხიდან განხორციელდება სხვადასხვა სახის დაქვითვა. დავუშვათ, რომ დაქვითვები ჯამში შეადგენენ: 15%-ს 150 ლარზე დაბალ გამოწერილ თანხაზე, ხოლო 20%-ს – 150 ლარზე მაღალ თანხაზე. მოთხოვნა შევქმნათ მანამდე შექმნილი მოთხოვნის **Semosuli Tanxebi2** საფუძველზე.
9. გავხსნათ მოთხოვნა კონსტრუქტორის რეჟიმში (**Create query in Design view**) და დიალოგის ფანჯარაში **Show Tables** ჩანართიდან **Query** მოვნიშნოთ მოთხოვნა **Semosuli Tanxebi2**. მოთხოვნის მაკეტიდან ბლანკზე გადმოვიტანოთ ველები **gvari, saxeli** და **SumOfanazRaureba** (ნახ.35).

ნახ. 35



10. შევქმნათ ახალ სვეტში ველი გამოწერილი თანხის ნახევრის დასათვლელად. ამისათვის სვეტის სათაურში შევიტანოთ გამოსახულება: **naxevari:[SumOfanazRaureba]* 0.5.**
11. მომდევნო სვეტში გამოვითვალოთ **daqviTva.** ამ სიდიდის გამოსათვლელად შევიტანოთ გამოსახულება: **daqviTva:IIf([naxevari]<150, [naxevari]*0.15, [naxevari]*0.2).**
12. გამოვითვალოთ ხელზე ასაღები თანხა, ამისათვის მომდევნო ახალი სვეტის სათაურში შევიტანოთ გამოსახულება **xelzeasaRebi:[naxevari]-[daqviTva].** სამივე ველი დავაფორმატოთ წინა მოთხოვნაში აღწერილი წესით.
13. დავხუროთ კონსტრუქტორის ფანჯარა და მოთხოვნას მივანიჭოთ სახელი **xelfasi.** მოთხოვნის შესრულებაზე გაშვებისას გამოჩნდება პარამეტრის ფანჯარა, რომელშიც შევიტანთ კონკრეტული თვის რიცხვით მნიშვნელობას და დავათვალიერებთ დამაჯამებელ ცხრილს.

ფორმები

მონაცემთა ბაზაში გამოყენების სიმარტივის და სიხშირის თვალსაზრისით ფორმა ყველაზე მნიშვნელოვანი ობიექტია. იგი ბაზის გარე სამყაროსთან ურთიერთობის საუკეთესო საშუალებაა, რადგან მომხმარებელი პროგრამის ყოველი გაშვებისას მუშაობს ფორმასთან – შეაქვს მონაცემები, მართავს პროცესებს და ა.შ. ამ დროს ცხრილები და მოთხოვნები მისთვის რჩება უხილავი. გამომდინარე აქედან, ფორმის შექმნის დროს დიდი ყურადღება ექცევა მის გარეგნულ სახეს.

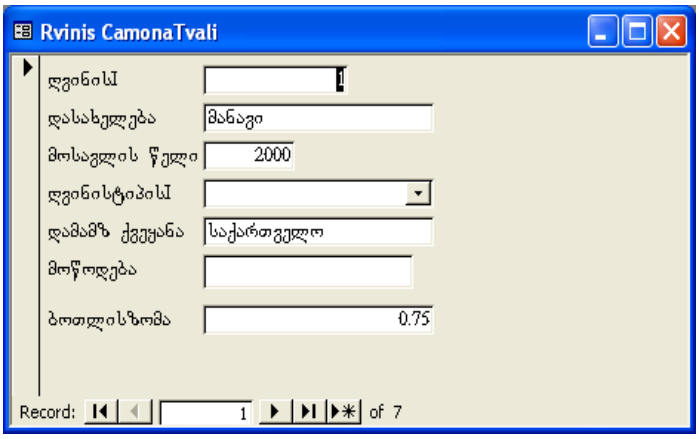
ფორმა მომხმარებელს, ერთის მხრივ, მონაცემების უშუალოდ ცხრილებში შეტანის საშუალებას აძლევს (თანაც ცხრილებთან შეუხებლად), მეორეს მხრივ – მოთხოვნების დამაჯამებელ ცხრილებს ლამაზი და თვალსაჩინო ხედით წარმოაჩენს. ამრიგად, ფორმას ვიყენებთ: მონაცემების შეტანის, რედაქტირების (მონაცემების დამატება, ამოღება, დამალვა, გამოთვლა და ა.შ.), ბაზის მუშაობის მიმდინარეობის მართვის (მონაცემების ძებნა, გაფილტვრა, მაკროსებისა და **Visual Basic** პროცედურების გამოყენება და სხვ.) და, აგრეთვე, ინფორმაციის საბეჭდ მოწყობილობაზე გატანის მიზნით.

ფორმების უმეტესობა ცხრილების და მოთხოვნების საფუძველზე იქმნება. შესაძლებელია კომბინირებული ვარიანტიც. **Access**-ს აქვს ფორმის აგების ავტომატიზების საშუალებები – *ავტოფორმები*. ავტოფორმებით იგი გარკვეულ სტანდარტულ სტრუქტურებს გვთავაზობს. ფორმა შეიძლება აგრეთვე *ოსტატის* დახმარებითაც შეიქმნას. ფორმის ოსტატი სპეციალური პროგრამული საშუალებაა, რომელიც მის სტრუქტურას ბაზის დამმუშავებელთან დიალოგში ქმნის. მაგრამ ჩვენი ფანტაზიის და გემოვნების გამოვლინებისთვის ფორმა უმჯობესია ავაგოთ კონსტრუქტორის რეჟიმში (**Design View**). აქ შესაძლებელია ყველა წვრილმანის გათვალისწინება ფორმის გარეგნული სახის საუკეთესოდ წარმოჩენისა და მართვის ეფექტურად განხორციელებისათვის. ფორმას განვიხილავთ და მასთან ვმუშაობთ ე.წ. ფორმის რეჟიმში (**Form View**).

თუ ფორმა მხოლოდ ერთ ობიექტს ეფუძნება, მას *მარტივი* ეწოდება, ხოლო თუ იგი რამდენიმე დაკავშირებულ ობიექტზეა აგებული – *რთული*. ასეთ დროს იგი წარმოადგენს ფორმების კომპოზიციას. რთული ფორმის აგების წესის მიხედვით განვასხვავებთ *დაქვემდებარებულ* და *დაკავშირებულ* ვარიანტებს. ხშირად ვიყენებთ აგრეთვე *მოდალურ* და *ტივტივა* ფორმებს. პირველი მუშაობის გასაგრძელებლად მომხმარებლისაგან შესაბამის შეტყობინებას ითხოვს, მეორე ყოველთვის აქტიურია, განლაგებულია სხვა ფანჯრებთან შედარებით წინა პლანზე.

ავტოფორმები

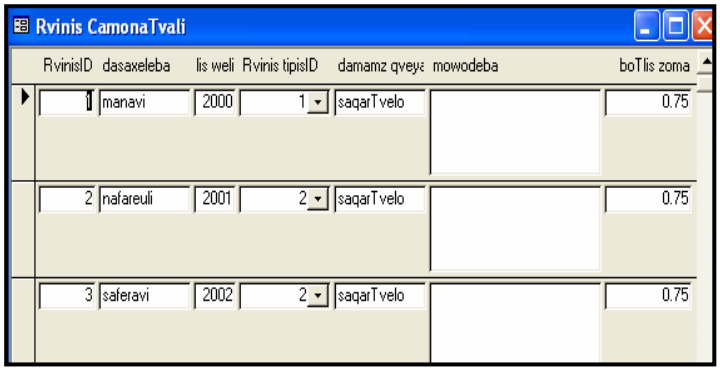
ავტოფორმის გამოყენებით ფორმის აგების სამი ვარიანტი არსებობს: *სვეტებად* (**Columnar,** ნახ.36), *სტრიქონებად* (**Tabular,** ნახ.37) და *ცხრილად* (**Datasheet**). **Columnar** ტიპის ფორმა ასახავს ერთი ჩანაწერის ყველა ველს, იგი მოსახერხებელია მონაცემების



ნახ. 36

შეტანისა და რედაქტირებისათვის. **Tabular** ტიპის ფორმა ასახავს ჩანაწერთა ჯგუფს, მისი გამოყენება მოსახერხებელია მონაცემების საბეჭდ მოწყობილობაზე გატანის დროს. **Datasheet** ტიპის ფორმა გარეგნულად არ განსხვავდება საბაზო ცხრილისაგან. ფორმის შესაქმნელად ავტოფორმების გამოყენებით ბაზის ძირითად ფანჯარაში ჩავრთავთ ღილაკს **Forms**, შემდეგ **New.** გამოტანილ დიალოგში ავარჩევთ ავტოფორმის სახეობას. ფანჯრის ქვედა ნაწილში, ველში **Choose the table or query where the object's data comes from** ჩამოსაშლელი სიიდან ავარჩევთ იმ ცხრილს ან მოთხოვნას, რომლის საფუძველზე იგება ფორმა, **OK.**

ფორმა ავტომატურად შეიქმნება და მზადაა გამოსაყენებლად – მონაცემების შეტანის და სხვა მოქმედებების ჩასატარებლად. ყურადღება მივაქციოთ იმას, რომ ავტოფორმა მხოლოდ ერთ ობიექტს (ცხრილს ან მოთხოვნას) ეფუძნება.

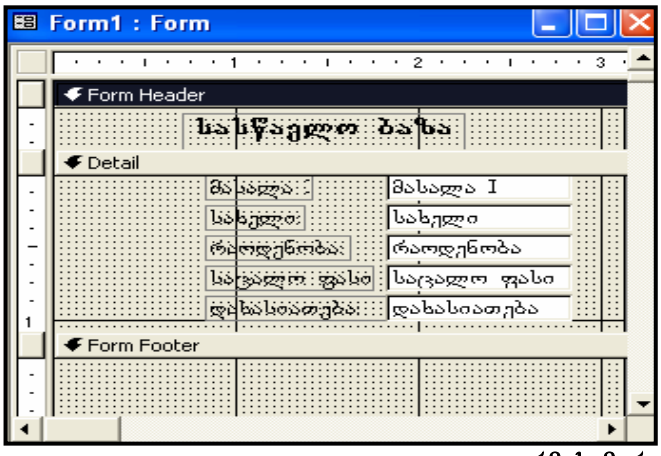


-.

ნახ. 37

ფორმის სტრუქტურა

კონსტრუქტორის რეჟიმის ჩართვით გავეცნობით ფორმის სტრუქტურას: ბაზის ძირითადი ფანჯრიდან ჩავრთავთ **Forms→ New→ Design View** (ნახ.38). ფორმა დაფარულია ბადით, რომელიც აადვილებს მასზე ობიექტების ოპტიმალურად განთავსებას. ბადეს ჩავრთავთ ან გამოვრთავთ მენიუდან **View → Grid.** როგორც ნახატიდან ჩანს, ფორმა 3 ძირითადი განყოფილების (სექციისაგან) შედგება: სათაურის (**Form Header**), მონაცემების (**Detail**) და შენიშვნების (**Form Footer**). ინფორმაცია, რომელიც საბაზო ცხრილიდან ან მოთხოვნიდან შემოდის, თავსდება ფორმის შუა ნაწილში **Detail** უბანზე. სათაურის და შენიშვნების არეებს, რომლებსაც *კოლონტიტულები* ეწოდება, მხოლოდ გაფორმების დანიშნულება აქვთ. მათი შინაარსი ცხრილებთან ან მოთხოვნებთან არ არის დაკავშირებული. აქ შეტანილი ინფორმაცია ყველა ჩანაწერისთვის ერთია. ზოლები, რომლითაც ეს განყოფილებები ერთმანეთისაგან გამოიყოფა, გადაადგილდებიან ვერტიკალურად თაგვის მეშვეობით, რითაც მათ სასურველ ზომას შევურჩევთ. თუ ფორმა თავსდება ქაღალდის ერთ ფურცელზე, მას პრინტერზე გავიტანთ **Windows**-ში არსებული წესით (ფორმის რეჟიმში **File → Print**). თუ ფორმების რაოდენობა ან ზომა ქაღალდის ფურცლის ზომებს აღემატება, მაშინ ბეჭდვის გაგრძელება უნდა მოხდეს მომდევნო გვერდზე გადასვლით. ამ დროს დავამატებთ გვერდის სათავისა და შენიშვნების არეებს **Page Header, Page Footer.** ფორმის და გვერდის კოლონტიტულების ჩართვა/გამორთვა ხდება მენიუს პუნქტიდან **View → Form Header/Form Footer** და **View → Page Header/Page Footer.** გვერდის სათავეში შევიტანთ გვერდის ნომერს, რისთვისაც ჩავრთავთ მენიუს პუნქტს **Insert → Page Number.**



ნახ. 38

ფორმის მართვის ელემენტები

კონსტრუქტორის რეჟიმში მუშაობის დროს ვიყენებთ მართვის ელემენტებს. მათი დახმარებით ვქმნით სხვადასხვა დანიშნულების ველებს, რომლებშიც განვახორციელებთ მონაცემთა შეტანას ან გამოტანას, ვათავსებთ წარწერებს, რომლებიც ზრდიან ფორმის თვალსაჩინოებას და ინფორმატიულობას, ვამატებთ გადამრთველებს, ალმებს, სიებს და ელემენტებს. მართვის ელემენტები მოთავსებულია *ელემენტთა პანელზე* **Toolbox** (ნახ.39). მას გამოვიტანთ მენიუს პუნქტით **View → Toolbox** ან იმავე სახელწოდების ღილაკით. მართვის ელემენტების პანელი ფორმის შექმნის თავისებური "მართვის ცენტრია", რომლიდანაც ვასრულებთ ამა თუ იმ ბრძანებას ან გარკვეული დანიშნულების პროგრამას. როდესაც

ნახ.39

ჩავრთავთ პანელის რომელიმე ღილაკს, თაგვის მაჩვენებელი იღებს ამ ელემენტის ნიშნაკის ფორმას. თაგვის მაჩვენებელს დავაწკაპუნებთ **Details** არეში ან დავაყენებთ იქ, სადაც მართვის ელემენტის მარცხენა ზედა კუთხე უნდა მოთავსდეს. შემდეგ თაგვის ღილაკის დაჭერილ მდგომარეობაში მაჩვენებელს სასურველი არის მარჯვენა ქვედა კუთხეში გადავიტანთ და თაგვის ან ავიღებთ ხელს. სხვა ღილაკის ჩართვის შემდეგ არჩეული ღილაკი გამოირთვება. ღილაკის იაოთული მდგომარეობის დასაფიქსირებლად მასზე ორჯერ დავაწკაპუნებთ.

ელემენტთა პანელის ღილაკები შემდეგ ფუნქციებს ახორციელებენ:

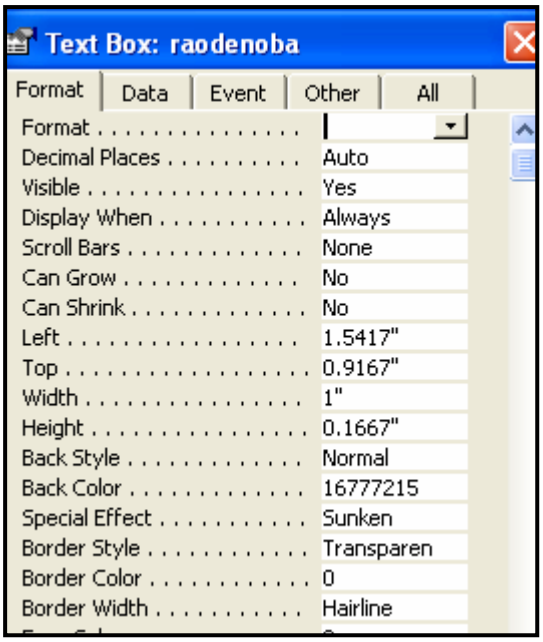
- **Select Objects** – ობიექტის მონიშვნის საშუალება. იგი თავიდანვეა ჩართული.
- **Control Wizards** – ოსტატების ჩამრთველი ღილაკი. ოსტატს გამოვიყენებთ ელემენტების დაჯგუფების, სიების ან დამატებითი ღილაკების ფორმაზე მოთავსების და სხვა ოპერაციების განხორციელების მიზნით.
- **Label** – წარწერა. მართვის ელემენტებს, როგორც წესი, მინაწერი (**Label**) აქვთ დართული. ამ ღილაკს გამოვიყენებთ აგრეთვე ისეთი ტექსტების შესატანად (სათაურები, ინსტრუქციები და სხვა), რომლებიც არ არიან ამა თუ იმ ელემენტთან მიერთებული.

- **Text Box** გამოიყენება ისეთი ველის შესაქმნელად, რომელშიც ტექსტის, რიცხვის, თარიღის შეტანა/გამოტანის საშუალება გვექნება. შესაძლებელია მისი დაკავშირება ცხრილის ან მოთხოვნის რომელიმე ველთან. ეს ელემენტი შეგვიძლია გამოვიყენოთ აგრეთვე გამოთვლების შედეგების გამოსატანად.
- **Option Group** გამოიყენება მართვის ელემენტების დასაჯგუფებლად. ჯგუფში შეიძლება რამდენიმე გადამრთველის, ალმის და სხვა. ელემენტის გაერთიანება. მათი რედაქტირება (ზომების ან ადგილმდებარეობის შეცვლა) ხდება ერთიანად. თუ ჯგუფი მიერთებულია საბაზისო ცხრილის ან მოთხოვნის რომელიმე ველთან, მაშინ ჯგუფში შემავალი ელემენტის შერჩევით მიერთებულ ველს ახალი მნიშვნელობა ენიჭება.
- **Toggle Button** გამოიყენება მართვის ელემენტის ე.წ. "ჩამრთველის" შესაქმნელად, რომელიც იღებს მნიშვნელობებს: *ჩართული / გამორთული, ჭეშმარიტი / მცდარი* ან *კი / არა*. ჩართვის შემთხვევაში მისი მნიშვნელობა 1-ის ტოლია: *ჩართული, ჭეშმარიტი* ან *კი*. თუ ერთხელაც დავაჭერთ ღილაკს ხელს, იგი მიიღებს მნიშვნელობას 0: *გამორთული, მცდარი* ან *არა*. ჩამრთველი ღილაკი მიერთებულია საბაზისო ცხრილის ან მოთხოვნის რომელიმე ლოგიკური ტიპის ველთან.
- **Option Button** გამოიყენება მართვის ელემენტის "გადამრთველის" შესაქმნელად, იღებს ისეთივე მნიშვნელობებს, როგორც "ჩამრთველი". გადამრთველების გაერთიანება შეიძლება ელემენტთა ჯგუფში. შესაძლებელია ამ ელემენტის მეშვეობით ველისათვის უნიკალური რიცხვითი მნიშვნელობის მინიჭება.
- **Check Box** გამოიყენება მართვის ელემენტის "ალმის" შესაქმნელად, იგი ისეთივე მნიშვნელობებს იღებს, როგორც "ჩამრთველი". შესაძლებელია ალმების ჩართვა ელემენტთა ჯგუფში და ველისათვის უნიკალური მნიშვნელობის მინიჭება.
- **Combo Box** გამოიყენება მართვის ელემენტის "სიის შემცველი ველის" შესაქმნელად, რომელიც შეიცავს სიას. სიის შიგთავსის განსაზღვრა შეიძლება **Row Source** სტრიქონზე მნიშვნელობების შეტანით. შესაძლებელია აგრეთვე ცხრილის ან მოთხოვნის ველის მიერთება. მნიშვნელობა, რომელსაც ფორმაზე ავარჩევთ, მყისიერად დაფიქსირდება საბაზო ობიექტში.
- **List Box –** სია. იგი ანალოგიურია **Combo Box** ელემენტის, მხოლოდ სია გამოტანილი იქნება სრულად, ველის ჩამოსაშლელი ღილაკის გარეშე. თუ მის ზომას (სიმაღლეს) შევამცირებთ, ფორმის რეჟიმში გადასვლისას მას ავტომატურად დაემატება გადაფურცვლის ზოლი (ღილაკებით).
- **Command Button** გამოიყენება როგორც ჩამრთველი ღილაკი, რომელიც შეასრულებს მაკროსს ან **Visual Basic**-ის პროცედურას. ამ პროცესებს ვახორციელებთ ოსტატების დახმარებით.
- **Image –** გამოსახულება. საშუალებას გვაძლევს, ფორმას დავურთოთ სტატიკური ნახატი რედაქტირების შესაძლებლობის გარეშე. ნახატი ფორმის ფონადაც შეგვიძლია გამოვიყენოთ – ჯერ ნახატის, ხოლო შემდეგ ველების გამოტანით.
- **Unbound Object Frame** საშუალებას გვაძლევს ფორმას ობიექტი (ნახატი, დიაგრამა და ა.შ.) მივუერთოთ სხვა პროგრამული დანართიდან, რომელიც ახორციელებს **ActiveX** ტექნოლოგიას. თვით ობიექტი ფორმაზე არ ინახება, აქ გვაქვს მხოლოდ მასზე მიმართვა.
- **Bound Object Frame – "**ობიექტთან მიერთებული ჩარჩო". გამოიყენება **ActiveX** ობიექტის მისაერთებად. მონაცემის ეს ტიპი ან ცხრილშივე უნდა იყოს განსაზღვრული, როგორც **OLE Object,** ან ფორმის თვისებებში (**Properties)**. თვით ნახატი ან დიაგრამა ფორმაზე აისახება **Form View** რეჟიმში. ამ ელემენტის გამოყენება მოსახერხებელია თითოეული ჩანაწერისთვის სხვადასხვა ნახატის გამოტანის საჭიროების შემთხვევაში.
- **Page Break** გამოიყენება მრავალგვერდიან ფორმაში.
- **Tab Control** გვაძლევს ფორმაზე ჩანართების გამოყენების საშუალებას.
- **Subform/Subreport –** "დაქვემდებარებული ფორმა/ანგარიში". საშუალებას გვაძლევს ფორმას დავურთოთ სხვა ფორმა. მაგალითად, დაქვემდებარებული ფორმის დანერგვით შეგვიძლია იმ მონაცემების გამოტანა, რომლებიც ძირითად ფორმასთან არიან კავშირში.
- **Line** – ხაზი. გამოიყენება ფორმაზე სწორი ხაზის გასატარებლად – თვალსაჩინოების გაზრდის მიზნით.
- **Rectangle –** მართკუთხედი. ასევე აუმჯობესებს გარეგნულ სახეს.
- **More Controls –** მართვის სხვა ელემენტები. ამ ღილაკზე დაწკაპუნებით გამოჩნდება **ActiveX** სისტემაში ჩართული პროგრამები, შესაძლოა ზოგიერთმა მათგანმა **Microsoft Access -**ში არ იმუშაოს. მაგალითად, პროგრამების საკმაოდ დიდი ჩამონათვალიდან შეგვიძლია **Calendar Control 9.0** გამოვიყენოთ, რომელიც ფორმის ველში კომპიუტერის კალენდრიდან თარიღის შეტანის საშუალებას გვაძლევს.

ფორმის განყოფილებების და ელემენტების თვისებები

ფორმის განყოფილებებს (სათაურის, მონაცემების ან შენიშვნების არეებს) და მართვის ელემენტებს (ღილაკებს, გადამრთველებს და სხვ.) მათთვის სპეციფიკური თვისებები ენიჭებათ თვისებების ფანჯრის **(Properties)** საშუალებით. თვისებების დასათვალიერებლად ან ახალი მნიშვნელობის მისანიჭებლად განყოფილება უნდა მოინიშნოს, შემდეგ **View → Properties**. თვისებების ჩამონათვალი თვით ელემენტის ტიპზეა დამოკიდებული, მაგალითად, **masala** ცხრილის (ბაზა **samS masalebi) raodenoba** ველისთვის (მონიშნული უნდა იყოს თვით ველი და არა მინაწერი). თვისებების ჩამონათვალი მოყვანილია თვისებების **Format** ჩანართზე (ნახ.40). **Format** და **Desimal Places** ისეთივე თვისებებია, როგორსაც ვიყენებდით ცხრილის კონსტრუქტორის რეჟიმში მუშაობის დროს **(Field Properties); Visible** უზრუნველყოფს ველის მნიშვნელობის ჩვენება/ არჩვენებას

ფორმის რეჟიმში გადასვლის შემდეგ; **Display When** განაპირობებს მის არსებობას მხოლოდ ეკრანზე ან მხოლოდ ქაღალდზე (პრინტერზე გატანის დროს);



ნახ. 40

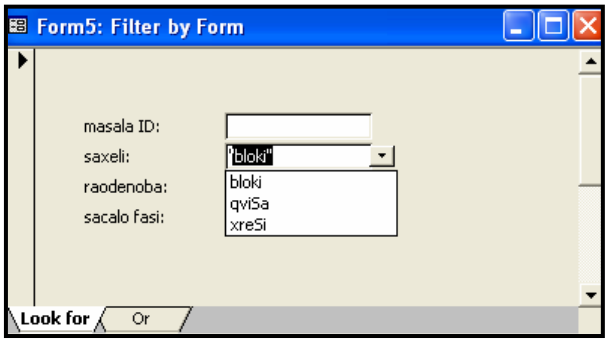
Scroll Bars უზრუნველყოფს ველში გადაფურცვლის ზოლების არსებობას; **Can Grow, Can Shrink** – განაპირობებს შემცირების ან გაჭიმვის საშუალებას; **Left, Top** – მიუთითებს ელემენტის მიერ დაკავებული არის მარცხენა ზედა კუთხის კოორდინატებს; **Width, Height** – ველის ჩარჩოს სიგანეს/სიმაღლეს; **Back Style, Back Color, Special Effect** – აგვარჩევინებს ველის გაფორმების ვარიანტებს; **Border Style, Border Color, Border Width-** ჩარჩოს გაფორმების ვარიანტებს, **Fore Color, Font Name, Font Size, Font Weight, Font Italic, Font Underline** – შრიფტის მახასიათებლებს, **Text Align** და სხვ. – ტექსტის სწორების საშუალებებს. თვისებების დანარჩენ ჩანართებზე შევარჩევთ მონაცემების და სხვა დანიშნულების ელემენტების თვისებებს, გამოტანის წესებს (მაგალითად, ტექსტისათვის – ანბანით, რიცხვებისათვის – ზრდადობით ან კლებადობით და ა.შ.). თვისებები შეგვიძლია აგრეთვე ფორმის ცალკეული განყოფილებისთვისაც განვსაზღვროთ, თუ მოვნიშნავთ შესაბამის ზოლს.

ფორმაში მონაცემების მოძიება, დახარისხება, გაფილტვრა

ფორმებში განვახორციელებთ მონაცემების მოძიების, დახარისხების და გაფილტვრის პროცესებს. ამისათვის ისეთივე საშუალებებს ვიყენებთ, როგორსაც ცხრილებში მუშაობის დროს. მაგალითად, ძებნისათვის მოვნიშნავთ ველს (დავაყენებთ იქ კურსორს) და ჩავრთავთ მენიუს პუნქტს **Edit → Find.** გამოტანილი დიალოგის ველში შევიტანთ საძიებელ მონაცემს და **Find Next**. შედეგი მიიღება მოძიებული ჩანაწერების სახით. ანალოგიურად განვახორციელებთ შენაცვლების **(Replace)** პროცესს.

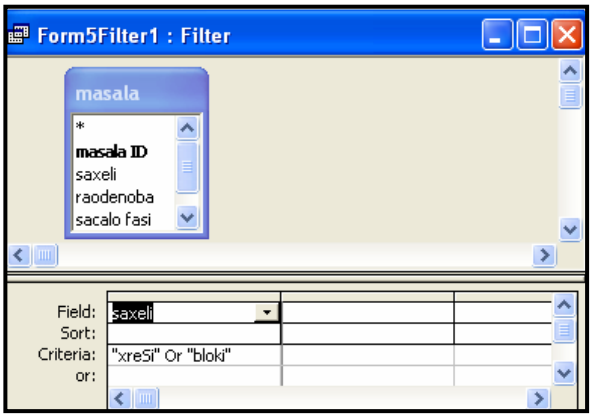
სწრაფი დახარისხება ცხრილში არსებული წესის მსგავსად ხდება: მოვნიშნავთ ველს და მენიუს პუნქტიდან **Records** ან ინსტრუმენტთა პანელის ღილაკებით **Sort Ascending** ან **Sort Descending** განვახორციელებთ ზრდადობით (თუ ტექსტია – ანბანით) ან კლებადობით დალაგებას. დაქვემდებარებულ ფორმებში სწრაფი ძებნა და დახარისხება არ განხორციელდება.

ზოგადად, ფილტრის გამოყენების არსი მდგომარეობს იმაში, რომ მხოლოდ იმ მონაცემების შემცველი ჩანაწერი უნდა იყოს გამოტანილი, რომელიც დასმულ პირობებს აკმაყოფილებს. მაგალითად, თუ ფილტრი უნდა განხორციელდეს გამოყოფილი ველის (რომელშიც კურსორია მოთავსებული) მნიშვნელობის მიხედვით, გამოვიყენებთ მენიუს პუნქტს **Records** ან ინსტრუმენტთა პანელის ღილაკს **Filter By Selection.** გამოიტანილი იქნება ფორმა მხოლოდ გაფილტრული (ამორჩეული) ჩანაწერებით, ანუ იმ ჩანაწერებით, რომლებიც შეიცავენ მონიშნულ ველში მითითებულ მნიშვნელობას. ქვედა ზოლში ნაჩვენები იქნება მათი რაოდენობა, მაგალითად, **of 2 (Filtered),** ანუ ფილტრის შედეგად ნაპოვნია 2 ჩანაწერი. ფილტრის გამორთვა, ანუ ყველა ჩანაწერის აღდგენა ხდება ღილაკით **Remove Filter.** გაფილტვრა შესაძლებელია რამდენიმე ველის მიხედვითაც (ველებში მნიშვნელობების მითითებით). **Access**-ში გაფილტვრა შეიძლება სპეციალური ფორმითაც განხორციელდეს. ამისათვის ჩავრთოთ ღილაკი **Filter By Form,** რომლითაც გაიხსნება ამავე სახელწოდების ფანჯარა აქ მოვნიშნავთ ველს, რომლის მიხედვითაც ჩაირთვება ფილტრი. ველის მარჯვენა მხრიდან მდებარე ისრით გამოვიტანთ იმ მონაცემების სიას, რომლებიც ერთხელ მაინც ფიგურირებენ. ველში შეტანილ სიაში (ნახ.41). ავარჩევთ მონაცემს და ინსტრუმენტთა პანელის ღილაკით **Apply Filter** განვახორციელებთ გაფილტვრას. გაფილტვრის შედეგი გამოტანილი იქნება ფორმის სახით, რომლშიც ნაჩვენები იქნება მხოლოდ გაფილტვრის შედეგად არჩეული ჩანაწერები. ფილტრს გამოვრთავთ ღილაკით **Remove Filter.** გაფილტვრა შეგვიძლია ჩავატაროთ ერთდროულად რამდენიმე ლოგისათვის, თუ **Filter By Form** ფანჯარაზე ჩავრთავთ **Or** ჩანართს.



ნახ. 41

შესაძლებელია გაფილტვრის შედეგის დათვალიერება, ამისათვის ჩავრთავთ ინსტრუმენტთა პანელის **Records → Filter → Advanced Sort/ Filter** ღილაკს, რომელიც გამოიტანს მოთხოვნის ფანჯარას იქ დაწესებული დახარისხების თუ გაფილტვრის პირობებით (ნახ.42).

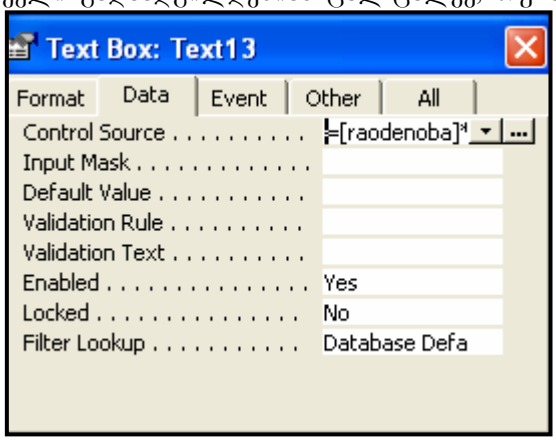


ნახ. 42

მარტივი ფორმის აგება კონსტრუქტორის რეჟიმში

კონსტრუქტორის რეჟიმში ფორმის ასაგებად ბაზის ძირითად ფანჯარაში ჩავრთავთ **Forms** და შემდეგ **New.** გამოტანილ დიალოგში ავარჩევთ **Design View** რეჟიმს, **Choose the table or query where the object's data comes from** ველში მოვნიშნავთ ცხრილს ან მოთხოვნას, რომლის საფუძველზეც იქმნება ფორმა. ფორმის სტრუქტურის ფანჯარაში გამოტანილი იქნება ელემენტთა პანელი და შერჩეული ცხრილის ან მოთხოვნის მაკეტი ველების ჩამონათვალით. წინააღმდეგ შემთხვევაში ჩავრთავთ მენიუს პუნქტს **View → Toolbox** და **View → Field List.** სიაში ველის ან მათი ერთობლიობის მონიშვნის შემდეგ თაგვით გადმოვიტანთ მათ **Details** არეში. ამრიგად, აქ მოთავსდება გადმოტანილი ველები თავისი მინაწერებით.

ველების რედაქტირებისათვის შეგვიძლია მოვნიშნოთ თითოეული ელემენტი ცალკე ან რამდენიმე ერთად (**Shift** კლავიშით ან სახაზავზე თაგვის მაჩვენებლის გადატარებით) და ერთობლივად განვიხილოთ მათი თვისებები, რისთვისაც ჩავრთოთ მენიუს პუნქტი **View → Properties** ან ინსტრუმენტთა პანელზე განლაგებული შესაბამისი ღილაკი. ელემენტის ზომების შეცვლისთვის მოვნიშნოთ იგი და ვიმუშაოთ არსებული წესით მარკერებით. ადგილმდებარეობის შეცვლისათვის თაგვის მაჩვენებელი დავაყენოთ მონიშნული არის მარცხენა ზედა კუთხეში: თუ იგი საჩვენებელი თითის ფორმას მიიღებს, მინაწერი და ველი გადაადგილდებიან ცალ-ცალკე, თუ ხელის გაშლილი მტევნის ფორმას – ერთად.



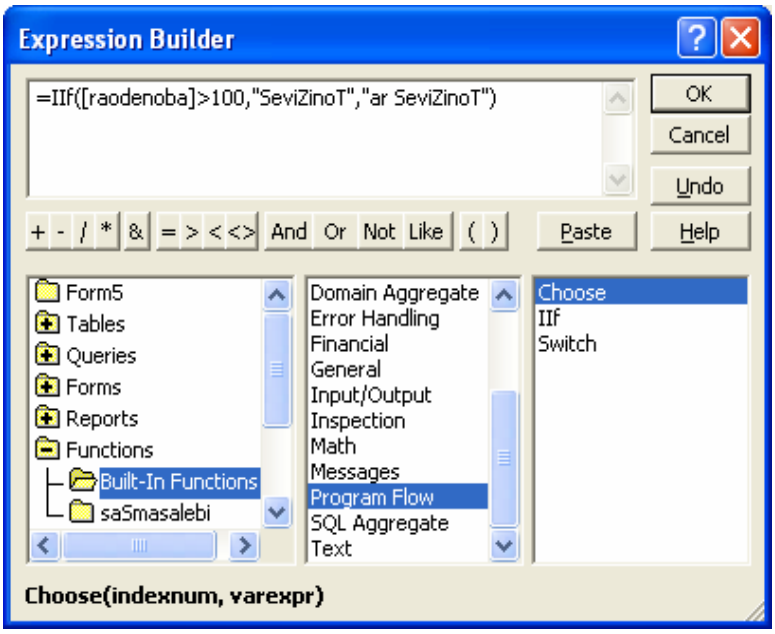
ნახ. 43

ადგილმდებარეობის და ზომების შერჩევის გარდა ინსტრუმენტთა პანელიდან ან მენიუს **Format** პუნქტით შეგვიძლია შევცვალოთ შრიფტი, ტექსტის სწორება (მარცხენა კიდეზე მიბჯენით, ცენტრის მიმართ სწორებით და ა.შ.), სიმბოლოების ან ფონის ფერი და სხვ. ფორმას შევინახავთ მიღებული წესით და მივანიჭებთ ახალ სახელს ან დავტოვებთ შემოთავაზებულს.

არსებული ველების გარდა ფორმაზე შეიძლება ახალი ველის დამატება, მაგალითად, გამოთვლების ან სხვა ფუნქციებით მიღებული შედეგების გამოსატანად. მაგალითად, კონსტრუქტორის რეჟიმში ავაგოთ ფორმა **samS masalebi** ბაზის **masala** ცხრილის საფუძველზე. **Text Box** ღილაკით ფორმის **Details** არეში შემოვხაზოთ მართკუთხედი. **Properties** ფანჯრის **Data** ჩანართის (ნახ.43) **Control Source** სტრიქონზე დავაჭიროთ ხელი იქ არსებულ სამ წერტილს. გამოტანილი იქნება გამოსახულების ამგები **Expression Builder** ფანჯარა. ფანჯრის შუა ნაწილში მოყვანილია ფორმის ველების და მინაწერების სახელები. ორჯერ დავაწკაპუნოთ **raodenoba** სახელზე, შემდეგ დავაჭიროთ ხელი გამრავლების ნიშანს და ისევ ორჯერ დავაწკაპუნოთ **erTeulis fasi** ველის სახელზე, **OK.** ველის მინაწერში შევიტანოთ სიტყვა "ღირებულება". გადავიდეთ ფორმის განხილვის რეჟიმში **Form View** და დავათვალიეროთ შედეგი. გამოსახულების შეტანა შეგვიძლია **Builder** პროგრამის გარეშეც, თუ თვისებების **Data** ჩანართის **Control Source** ველში შევიტანთ გამოსახულებას: **=[raodenoba]*[erTeulis fasi].**

ისევ **samS masalebi** ბაზაში გამოვიყენოთ პირობის შემოწმების ფუნქცია **IIf ,** ამისათვის **Expression Builder** ფანჯრის მარცხენა არეში გავხსნათ **Function → Built- In Function** საქაღალდე. შუაში გამოტანილ სიაში მოვნიშნოთ **Program Flow** და მარცხენა მხარეს – ორჯერ დავაწკაპუნოთ **IIf** ფუნქციის სახელზე. ფანჯრის ზედა ნაწილში გამოჩნდება ფუნქციის ზოგადი სახე **IIf("expr","truepart","falsepart").** მოვნიშნოთ

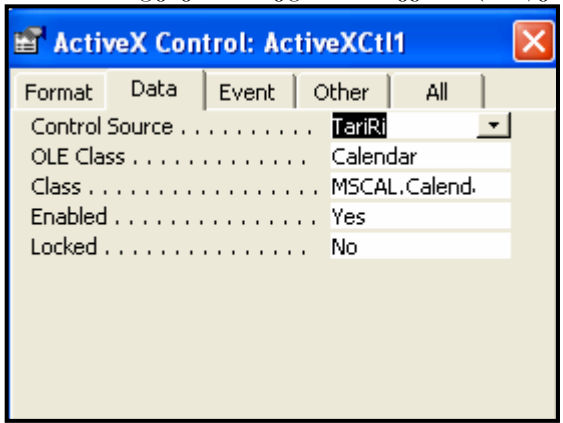
“expr”, შემდეგ გავხსნათ ამჟამად აქტიური **Form5** და შუა არიდან ავარჩიოთ **raodenoba** (ორჯერ დაწკაპუნებით). ფუნქციაში იგი მოთავსდება კვადრატულ ფრჩხილებში, დავასრულოთ პირობის შემცველი გამოსახულება და მივუწეროთ >100. მოვნიშნოთ **”truepart”** ნაწილი და შევიტანოთ ის ინფორმაცია, რომელიც შეესაბამება პირობის შესრულებას, **”falsepart”**-ში შევიტანოთ ინფორმაცია, რომელიც შეესაბამება პირობის არშესრულებას (ნახ.44). ამის შემდეგ დავათვალიეროთ სხვადასხვა ჩანაწერი ფორმის **View → Datasheet View** რეჟიმში: იმ ჩანაწერის ახალ ველში, სადაც რაოდენობა აღემატებოდა 200-ს, ჩაიწერება “შევიძინოთ”, სადაც ნაკლებია ან ტოლია 200-ის – “არ შევიძინოთ”.



ნახ. 44

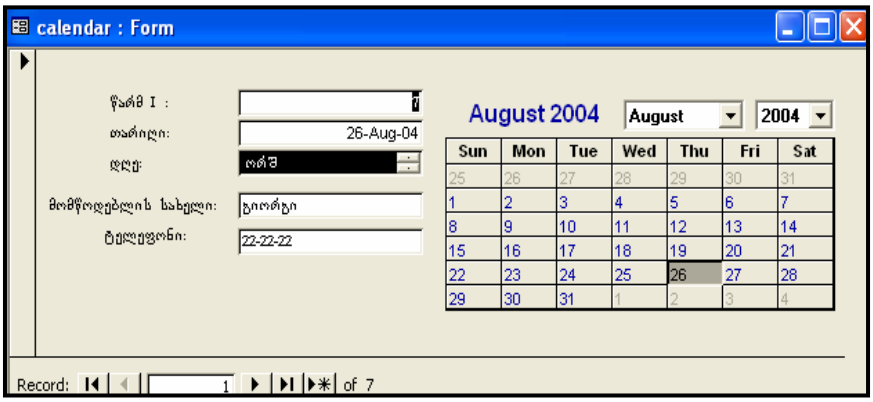
ცხრილის კონსტრუქტორის რეჟიმში მუშაობისას გარკვეული ველისთვის ვიყენებდით **Lookup** შესაძლებლობას, რათა საშუალება გვქონოდა მონაცემი შეგვეტანა წინასწარ შექმნილი სიიდან ან სხვა ცხრილში არსებული ველიდან. მონაცემის ასეთი წესით შეტანის ორგანიზების საშუალება არსებობს ფორმის კონსტრუქტორის რეჟიმშიც. მაგალითად, თუ ფორმიდან საბაზო ცხრილში ჩასაწერია საგნის კოდი, შეგვიძლია შეტანის პროცესი განვახორციელოთ მისი სახელის ჩვენებით. ამისათვის ელემენტთა პანელზე ჩავრთოთ **Combo Box** ღილაკი (ფორმის ამ ველს ცხრილის კონსტრუქტორის რეჟიმში **Lookup** ჩანართზე **Text Box** თვისება უნდა ჰქონდეს მინიჭებული). ამუშავდება ოსტატი. საბოლოოდ, ფორმიდან ცხრილში შევიტანთ კოდს სახელის მიხედვით. ეს აადვილებს კოდების შეტანის პროცესს. მაგალითად, სასწავლო ბაზა **samS masalebi** ცხრილში **gadamkveTi** შესატანია მასალების და სავაჭრო წარმოების კოდები. შევქმნათ **gadamkveTi** ცხრილის ფორმა, რომელშიც ორი ველია, ორივე გარე გასაღებების დანიშნულებისა. ბაზის ძირითად ფანჯარაში ჩავრთოთ **Forms**, შემდეგ **New,** ამოსარჩევი სიიდან მოვნიშნოთ **gadamkveTi, OK.** შემდეგ:

1. ელემენტთა პანელზე დავაჭიროთ ხელი **Combo Box** ღილაკს და **Details** არეში შემოვხაზოთ მცირე ზომის მართკუთხედი.
2. ოსტატის დიალოგის პირველ ფანჯარაზე ჩავრთოთ ღილაკი **I want the combo box to look up the values in a table or query, Next.**
3. ოსტატის მომდევნო დიალოგის **View**-ში ჩავრთოთ **Tables** და ცხრილების სიიდან ავარჩიოთ **masala,** რადგან სწორედ ამ ცხრილშია მოყვანილი საგნის შესაბამისი კოდი**, Next.**
4. მომდევნო ფანჯრის მარცხენა მხარეს მყოფი ველების სიიდან გადავიტანოთ **masala ID** და **saxeli** ველები მარჯვენა მხარეს, **Next.**
5. მომდევნო ფანჯარაზე ჩავრთოთ ალამი **Hide key column,** რათა ფორმაზე ჩანდეს მხოლოდ სახელი (კოდის გარეშე), **Next.**
6. მომდევნო ფანჯარაზე ჩავრთოთ ღილაკი **Store that value in this field** და შემოთავაზებული სიიდან ავარჩიოთ **masala** ცხრილის **masala ID** ველი**,** სადაც უნდა მოთავსდეს ფორმაზე არჩეული სახელის შესაბამისი კოდი, **Next.**
7. მომდევნოზე შეირჩევა თვით ფორმის სახელი, **Finish.**
8. მონაცემების შეტანის შექმნილი წესი შევამოწმოთ ფორმის რეჟიმში. ფორმასთან ერთად გავხსნათ **gadamkveTi** ცხრილი და დავაკვირდეთ, თუ როგორ ხდება ფორმაზე ამა თუ იმ სახელის შერჩევით **gadamkveTi** ცხრილში შესაბამისი კოდის შეტანა. გამოვიყენოთ თარიღის ტიპის ველში მონაცემის შეტანის საშუალება უშუალოდ კალენდრიდან. ფორმის აგების კონსტრუქტორის რეჟიმში ელემენტთა პენელზე დავაჭიროთ ხელი **More…** ღილაკს.



ნახ. 45

გამოტანილი სიიდან ავარჩიოთ **Calendar Control 9.0** პროგრამა. ფორმის **Details** არეში შემოვხაზოთ მართკუთხედი, რომელშიც მოთავსდება კალენდარი. მის თვისებებში **Data** ჩანართზე (ნახ.45) **Control Source** სტრიქონზე ავარჩიოთ **TariRi** ველი, რომელშიც შესატანი იქნება თარიღი კალენდრიდან. ფორმის რეჟიმში გადასვლისას თარიღი-ველის გააქტიურების შემდეგ დავაწკაპუნებთ კალენდრის საჭირო თარიღზე, სხვა ველზე გადასვლის შემდეგ დავინახავთ შეტანილ თარიღს (ნახ.46).



ნახ. 46

მარტივი ფორმა შეიძლება შეიქმნას აგრეთვე მოთხოვნის საფუძველზე, სადაც გამოტანილი იქნება მოთხოვნაში გაერთიანებული ცხრილების ველები.

თუ ველს ორი განსხვავებული მნიშვნელობა უნდა მიენიჭოს, გამოვიყენებთ გადამრთველს ან ალამს. ცხრილის შესაბამისი ველი თავიდანვე ლოგიკურ ტიპად უნდა იყოს განსაზღვრული. ორზე მეტი ვარიანტიდან ერთის შერჩევის აუცილებლობის დროს ასეთ ელემენტებს გავაერთიანებთ ჯგუფში. ჯგუფში მხოლოდ ერთ ელემენტს შეიძლება "ჭეშმარიტი" მნიშვნელობა მიენიჭოს. ჯგუფის შესაქმნელად ელემენტთა პანელზე ხელს დავაჭერთ **Option Group** ღილაკს და ფორმის არეში შემოვხაზავთ მართკუთხედს. დანარჩენი პროცედურა შესრულდება ოსტატთან დიალოგში.

რთული ფორმის შექმნა

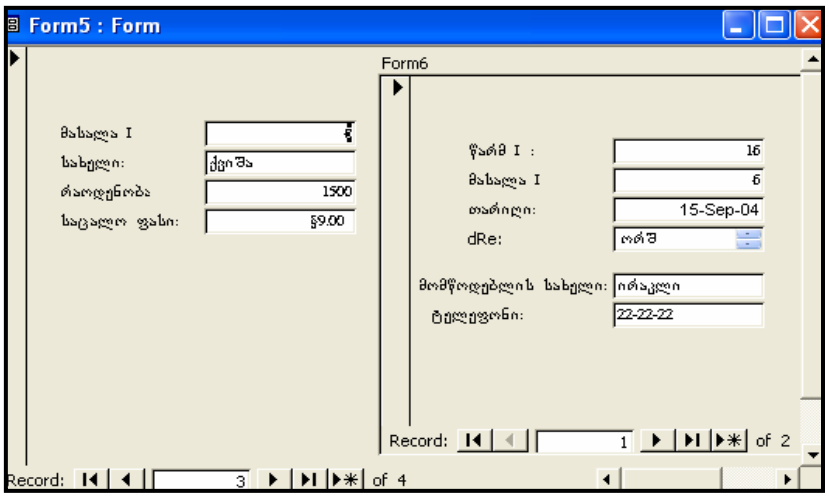
განვიხილოთ რამდენიმე ცხრილის მონაცემების ერთ ფორმაში გაერთიანების საშუალებები, ანუ შევქმნათ რთული ფორმა. მაგალითად, ძირითადი და მისი დაქვემდებარებული ფორმების გაერთიანებით განვახორციელოთ ორი ცხრილის ან მოთხოვნების მონაცემებთან ერთდროულად მუშაობა. ფორმების დაკავშირება შეიძლება ერთ ფორმაზე მეორის მოთავსებით ან სპეციალურად შექმნილი გადამრთველი ღილაკებით.

მაგალითად, რთული ფორმის შესაქმნელად გამოვიყენოთ ორი ცხრილი: "ერთის" მხარეზე მყოფი **masala** ცხრილი – ბაზიდან **samSmasalebi** და "მრავალის" მხარეზე მყოფი იმავე ბაზის ცხრილი **warmoeba**. შევქმნათ ორი ფორმა **masala** ცხრილის საფუძველზე **(Form5)** და **warmoeba** ცხრილის საფუძველზე **(Form6).** გავხსნათ **Form5** კონსტრუქტორის რეჟიმში, ელემენტთა პანელზე დავაჭიროთ ხელი **Subform/Subreport** ღილაკს. **Detail** არეში შემოვხაზოთ მართკუთხედი, რის შემდეგ ამუშავდება ოსტატი. მის პირველ ფანჯარაზე მოვნიშნავთ **Form6, Next,** ბოლოს **Finish**. მიღებულ ფორმაზე

ნახ.47

მასალის ყოველი დასახელების შესაბამისი მონაცემები გამოტანილი იქნება **Form6** ფორმაზე (ნახ.47).

ქვეფორმის გამოსაძახებლად ღილაკის გამოყენებით აგრეთვე მივმართავთ ოსტატს, რომელიც ჩაირთვება ელემენტთა პანელზე მოთავსებული **Command Button**-ის ჩართვის შემდეგ. მაგალითი განხილულია სავარჯიშოში.



ანგარიშები

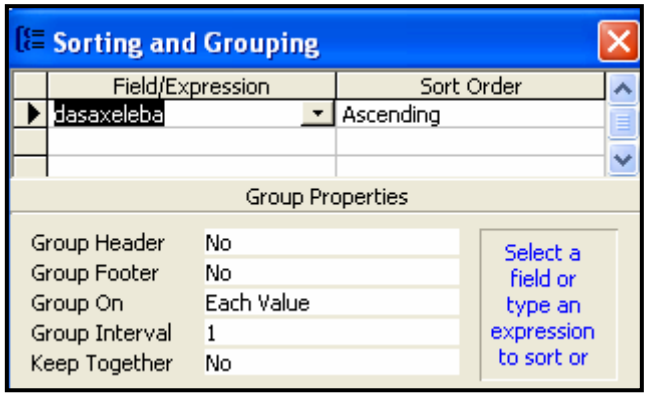
მონაცემთა ბაზის ცალკეული ობიექტი (ცხრილი, მოთხოვნა, ფორმა) შეგვიძლია დავბეჭდოთ ჩვეულებრივი წესით **File – Print** ბრძანებით. მაგრამ მონაცემების უფრო თვალსაჩინო და მიმზიდველი ფორმით წარმოდგენისთვის **Access**-ში გათვალისწინებულია სპეციალური ობიექტი, სახელწოდებით *ანგარიში* **(Report)**. ანგარიშებში შესაძლებელია დიდი რაოდენობის

მონაცემების დალაგება იერარქიულად, დაჯგუფება რაიმე ნიშნით, საშუალედო და საშედეგო გამოთვლების ჩატარება ჯგუფში მოთავსებულ მონაცემებზე და ა.შ. ანგარიშებით შესაძლებელია შეკვეთების და სხვა საპრეზენტაციო დანიშნულების მასალების ლამაზად გაფორმებაც.

ბაზის სხვა ობიექტის მსგავსად, ანგარიშის შექმნა შესაძლებელია სხვადასხვა გზით, მაგალითად, ე.წ. *ავტოანგარიშის* გამოყენებით. ამისათვის ბაზის ძირითად ფანჯარაში ვაჭერთ ხელს **Reports** ღილაკს და **New.** გამოტანილ დიალოგში შევარჩევთ სვეტებად **(AutoReport: Columnar)** ან სტრიქონებად გასაფორმებელ ავტოანგარიშს **(AutoReport: Tabular)**. ჩამოსაშლელ სიაში შევარჩევთ ცხრილს ან მოთხოვნას, რომლის საფუძველზეც იქმნება ანგარიში. ავტოანგარიშის შექმნა, ავტოფორმის მსგავსად, მხოლოდ ერთი ცხრილის ან მოთხოვნის საფუძველზე შეიძლება. თუ რამდენიმე ცხრილის მონაცემების ჩართვაა საჭირო, ჯერ იქმნება მოთხოვნა რამდენიმე ცხრილის მონაცემების გაერთიანებით, შემდეგ, ამ მოთხოვნის საფუძველზე – ანგარიში.

ანგარიშის შესაქმნელად შეგვიძლია გამოვიყენოთ აგრეთვე ანგარიშების ოსტატი. იგი რამდენიმე ცხრილის და მოთხოვნის გამოყენების საშუალებას მოგვცემს.

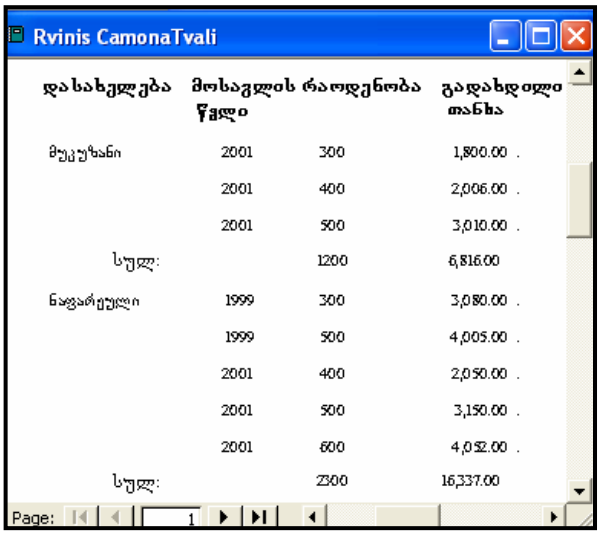
განვიხილოთ მაგალითი: შევქმნათ ანგარიში მონაცემთა ბაზა **Rvinis sardafi** ისეთი მოთხოვნის საფუძველზე, რომელშიც **Rvinis CamonaTvali** ცხრილიდან მხოლოდ ორი ველია გამოტანილი **dasaxeleba** და **mosavlis weli.** გამოვიყენოთ **Report:Tabular** სქემა. მიღებული ანგარიშის კონსტრუქტორის რეჟიმში გადასვლით ფორმა
ში მუშაობის მსგავსად შევარჩიოთ ქართული შრიფტი და ანგარიშის სათაური.



ნახ. 48

აქვე შევარჩიოთ დალაგების წესი – მენიუს პუნქტით **View → Sorting and Grouping** ან იმავე დასახელების ღილაკით გამოვიტანოთ შესაბამისი ფანჯარა (ნახ.48). აქ პირველ სტრიქონზე ისრით ჩამოსაშლელი სიიდან ავარჩიოთ **dasaxeleba** ველი და მარჯვენა მხარეს ჩავურთოთ მას ზრდადობის **Ascending** წესი, რითაც მოხდება ანგარიშში მოყვანილი სიის ანბანით მოწესრიგება. თუ სახელების დაჯგუფება გვინდა მოსავლის წლების მიხედვით, მაშინ **Sorting and Grouping** ფანჯარაზე ავირჩევთ **mosavlis weli** ველს და ანგარიშში ღვინის სახელები დაჯგუფებული იქნება წლების მიხედვით.

შუალედური ან საბოლოო შედეგების (ჯამის, საშუალო არითმეტიკულის და სხვა) გამოტანის ნიმუშის განსახილველად შევარჩიოთ ისევ განხილული ბაზის მოთხოვნა, რომელშიც თავმოყრილი იქნება **dasaxeleba**, **mosavlis weli**, **raodenoba** და **gadaxdili Tanxa.** ასეთი მოთხოვნის საფუძველზე ავაგოთ ანგარიში ისევ **Tabular** სქემით და დავაჯგუფოთ **mosavlis weli** და **raodenoba** ველებით, ამასთან, **Group Properties** განყოფილებაში **Group Footer** სტრიქონზე ავარჩიოთ **Yes.** კონსტრუქტორის რეჟიმში **Group Footer** არეში **Toolbox**-დან **Text Box** ღილაკით შემოვხაზოთ არე, რომელშიც ჩავწეროთ გამოსახულება: **=Sum([raodenoba]).** საბოლოო შედეგის, ანუ საერთო ჯამის გამოსატანად იგივე გამოსახულება ჩავწეროთ **Report Footer** განყოფილებაში. გარდა ამისა, **dasaxeleba** ველის (მოვნიშნოთ იგი) თვისებების **Properties Hide Duplikates** სტრიქონზე ჩავრთოთ **Yes,** რის შემდეგაც აღარ იქნება გამოტანილი გამეორებული სახელები (ნახ.49). ანალოგიურად შესრულდება საშუალო არითმეტიკულის **Avg,** მაქსიმუმის **Max** და სხვა ფუნქციების გამოთვლა. ფორმების მსგავსად ანგარიშებშიც შესაძლებელია



ნახ. 49

დაქვემდებარებული
ანგარიშების შექმნა, ნახატებით და ფოტოებით გაწყობა.

მონაცემთა ბაზის შეკუმშვა

ერთი ობიექტის გაუქმების და მეორეს შექმნის შედეგად მონაცემთა ბაზის ფაილი შეიძლება დისკზე არაოპტიმალურად განაწილდეს, ანუ გახდეს ფრაგმენტირებული. ამის შედეგად ის მეხსიერებაში უფრო დიდ ადგილს დაიკავებს, ვიდრე ესაჭიროება. ამ გაფანტული გამოუყენებელი უბნების გასაუქმებლად დროდადრო საჭიროა ბაზის შეკუმშვა. შეკუმშვის წინ ყველა ბაზა დახურული უნდა იყოს. მენიუს პუნქტით **Tools → Database Utilities → Compact and Repair Database** Access გახსნის ფანჯარას, რომელშიც ბაზების სიაა გამოტანილი. სისტემა ითხოვს შეკუმშული ბაზის ასლის სახელს. სიიდან შესაკუმში ბაზის არჩევის და სახელის მინიჭების შემდეგ დავაჭერთ ხელს **Compact** ღილაკს. **Access** შეკუმშავს მონიშნულ ბაზას, მოათავსებს დროებით გამოყოფილ ფაილში და თუ შეკუმშვის პროცესი წარმატებით დასრულდა, გააუქმებს დედანს და მიანიჭებს მას მანამდე არსებულ სახელს.

სავარჯიშო 12. ფორმის შექმნა ოსტატის გამოყენებით

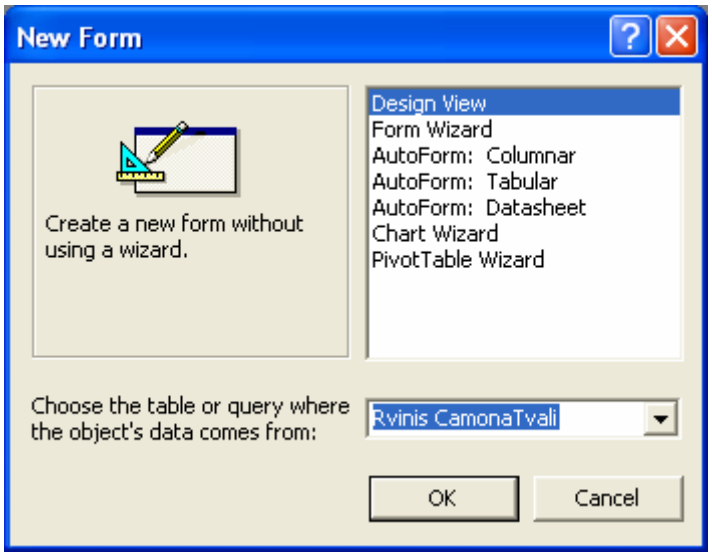
ოსტატის გამოყენებით შევქმნათ მარტივი ფორმა ბაზის ცხრილის საფუძველზე:

1. გავხსნათ რომელიმე ბაზა, მაგალითად, **samS masalebi.** მის ძირითად ფანჯარაში ჩავრთოთ **Forms** ღილაკი და ორჯერ დავაწკაპუნოთ **Create form by using wizard** ღილაკზე.
2. გამოტანილ დიალოგ-ფანჯარაზე **Tables/ Queries** ველში ავარჩიოთ ცხრილი ან მოთხოვნა, მაგალითად, **masala,** რომელიც საფუძვლად დაედება შესაქმნელ ფორმას. სიიდან **Available Fields** მოვნიშნოთ ფორმაზე გადასატანი ველები და ისრით (>) გადავიტანოთ ისინი მარჯვენა მხარეს. ერთობლივად ველების გადასატანად გამოვიყენოთ ორმაგი ისარი (>>). მოქმედების უარსაყოფად მონიშნული ველისთვის გამოვიყენოთ მარცხნივ მიმართული ისარი (<), ყველა ველისთვის – ორმაგი ისარი (<<), **Next.**
3. მომდევნო 2 ფანჯარაზე ოსტატი გვთავაზობს ფორმის სახეობის და სტილის არჩევას, **Next.**
4. უკანასკნელ ფანჯარაზე შემოთავაზებულია ფორმის სათაური, შეგვიძლია შევცვალოთ იგი. ფანჯარაზე ჩართულია ღილაკი, რომელიც ფორმას გამოიტანს განხილვის რაჟიმში, **Finish.**
5. დავათვალიეროთ ფორმა განხილვის **Form View** რეჟიმში, **Record** სტრიქონიდან განვახორციელოთ ჩანაწერიდან ჩანაწერზე გადასვლა. შევიტანოთ ახალი ჩანაწერები, შევინახოთ ფორმა, დავხუროთ.
6. ბაზის ძირითადი ფანჯრიდან ჩავრთოთ **Tables**, გავხსნათ და დავათვალიეროთ ის ცხრილი, რომელიც საფუძვლად დაედო ფორმას და რომელშიც შეტანილ იქნა (ფორმიდან) ახალი ჩანაწერი ან გადაკეთდა არსებული.
7. გავხსნათ ახლად შექმნილი ფორმა, გადავიდეთ კონსტრუქტორის რეჟიმში, მოვნიშნოთ რომელიმე ველი და კლავიატურის **Delete** კლავიშით გავაუქმოთ იგი. გადავიდეთ ფორმის რეჟიმში და დავათვალიეროთ შედეგი – ფორმა გაუქმებული ველით.
8. გავიმეოროთ სავარჯიშოს ყველა პუნქტი – ჩავატაროთ ველების რედაქტირება, ფორმის სხვადასხვა სახეობის და სტილის შერჩევა.

სავარჯიშო 13. მარტივი ფორმის აგება კონსტრუქტორის რეჟიმში

შევქმნათ მარტივი ფორმა ცხრილის საფუძველზე. ფორმის კონსტრუქტორის რეჟიმში შექმნისათვის შევასრულოთ შემდეგი პუნქტები:

1. ჩავტვირთოთ, მაგალითად, მონაცემთა ბაზა **Rvinis sardafi.** ბაზის ძირითად ფანჯარაში ჩავრთოთ **Forms**, **New.**

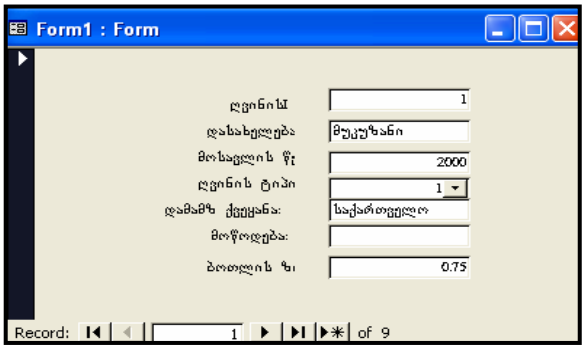


ნახ. 50

2. გამოტანილ დიალოგში (ნახ.50) ავარჩიოთ **Design View** რეჟიმი, ხოლო ველში **Coose the table or query …** მოვნიშნოთ იმ ცხრილის ან მოთხოვნის სახელი, რომლის საფუძველზე იქმნება ფორმა, მაგალითად, **Rvinis CamonaTvali, OK.**
3. არსებული წესით გავზარდოთ კონსტრუქტორის რეჟიმში გამოტანილი ფანჯრის ზომები. ცხრილის **Rvinis CamonaTvali** მაკეტის სათავეზე ორჯერ დაწკაპუნებით მოვნიშნოთ ყველა ველი (ან რომელიმე

სასურველი) და თაგვით გადმოვიტანოთ ისინი ფორმის დაფაზე **Details** არეში ისე, რომ მარცხნიდან დარჩეს 3სმ-მდე მანძილი (ველების მინაწერებისათვის). თუ ცხრილის მაკეტი გამოტანილი არ არის, ჩავრთოთ მენიუს პუნქტები **View → Field List.**

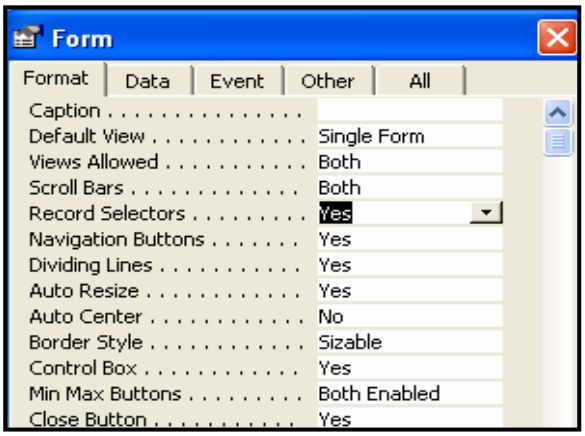
4. **Details** არეში გადმოტანილი ველები და შესაბამისი მინაწერები იქნება მონიშნული. მონიშვნა შეგვიძლია გავაუქმოთ და მიღებული წესით (ველზე დაწკაპუნებით) მოვნიშნოთ ცალკეული ელემენტი.
5. ყველა ელემენტის ერთობლივად მოსანიშნად ვისარგებლოთ მენიუს პუნქტით **Edit → Select All**. იმავეს მივაღწევთ თაგვის მაჩვენებლის სახაზავზე გადატარებით (მაჩვენებელი მიიღებს ქვევით მომართული შავი ისრის ფორმას) ან ცალკეულ ელემენტზე დაწკაპუნებით **Shift** კლავიშის დაჭერილ მდგომარეობაში. შევარჩიოთ შრიფტი არსებული წესით (მაგალითად, **AcadNusx)**, შრიფტის ზომა, სტილი, ფერი და სხვა.
6. გადავიდეთ ფორმის რეჟიმში, დავათვალიეროთ იგი. ფორმის რედაქტირების მიზნით – ელემენტების ზომების, ადგილმდებარეობის, შრიფტის და სხვა პარამეტრების შერჩევის ან გადაკეთებისათვის დავბრუნდეთ კონსტრუქტორის რეჟიმში.
7. მოვნიშნოთ რომელიმე ველი და შევცვალოთ მისი ზომა, ადგილმდებარეობა თაგვის ან **Properties** ფანჯრის გამოყენებით.
8. შევასრულოთ მინაწერების სწორება მარჯვენა კიდის მიმართ – მოვნიშნოთ ისინი და ჩავრთოთ მენიუს **Format → Align → Right** პუნქტი. ანალოგიურად შევასრულოთ ველების სწორება მარცხენა ან ზედა კიდის მიმართ.
9. გავაერთიანოთ რამდენიმე ველი ერთობლივი რედაქტირების მიზნით – მოვნიშნოთ ველები ერთობლივად, მენიუს **Format → Group** პუნქტით დავაჯგუფოთ ისინი, განვახორციელოთ მათი რედაქტირება (ფონის ფერის, ჩარჩოს სტილის და სხვა პარამეტრების შერჩევა).
10. გარეგნული იერსახის გაუმჯობესებისათვის გამოვიყენოთ ხაზები, მართკუთხედები. მენიუს პუნქტით **Format → Autoformat**



ნახ. 51

ავარჩიოთ ველებისათვის შემოთავაზებული განსხვავებული სტილი.

11. დავათვალიეროთ ფორმა **Form View** რეჟიმში (ნახ.51). ფორმაზე ველების კომპაქტურად განთავსების მიზნით დავაწკაპუნოთ მარცხენა კიდეზე მდებარე მონიშვნის ზოლზე (ნახატზე იგი გაშავებულია) და გამოვიყენოთ მენიუს პუნქტი **Window → Size to Fit Form.**
12. მენიუს პუნქტით **View → Properties** ან შესაბამისი ღილაკით გამოვიტანოთ ფორმის თვისებების ფანჯარა (ნახ.52). პუნქტით **Caption** ფორმას მივანიჭოთ სასურველი სახელი, **Views Allowed** სტრიქონზე ჩავრთოთ **Both** – ცხრილის და ფორმის (ორივე ვარიანტის) ჩვენების დაშვებისათვის.
13. გამოვრთოთ (მიუწვდომელი გავხადოთ **None Enabled**) ფანჯრის სტანდარტული **Min, Max** ღილაკები: **Min, Max Buttons –None.** იმავე წესით გავაუქმოთ **Close** – დახურვის ღილაკი. ამის შემდეგ დავიმახსოვროთ ფორმა და გადავიდეთ კონსტრუქტორის რეჟიმში. ფორმის რეჟიმში დაბრუნების შემდეგ ეს სამი ღილაკი მოცილებული ექნება. ამ ღილაკების



ნახ. 52

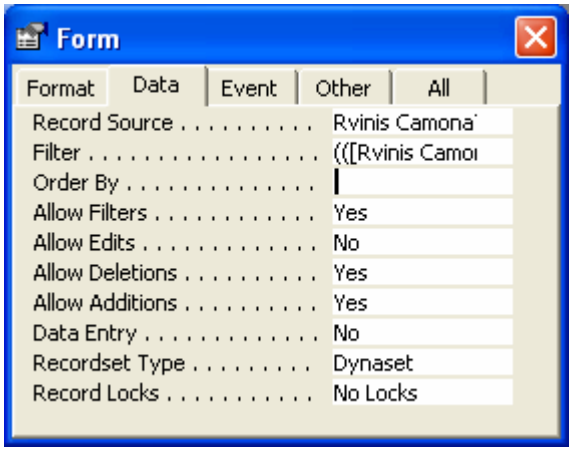
აღდგენა ხდება უკუსვლით. ფორმის დახურვა შეიძლება **Alt+F4** კლავიშებით (კლავიატურიდან).

14. თვისებას **Navigation Buttons** მივანიჭოთ **No,** რასაც მოჰყვება **Record** სტრიქონზე არსებული ჩანაწერების გადამრთველი ღილაკების გაუქმება. მათი აღდგენა შესაძლებელია **Undo** ბრძანებით.

სავარჯიშო 14. მარტივი ფორმის რედაქტირება, მონაცემების მოწესრიგება, ფილტრის გამოყენება

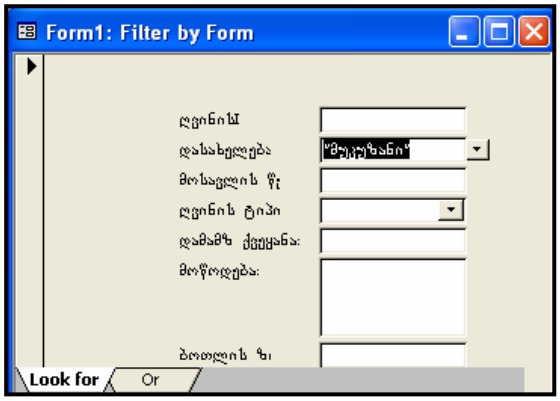
შევქმნათ ფორმა ბაზის ცხრილის საფუძველზე, მაგალითად, ჩავტვირთოთ ბაზა **Rvinis sardafi.** მონაცემების რედაქტირების, მოწესრიგების და გაფილტვრის მიზნით შევასრულოთ შემდეგი პუნქტები:

1. გავიმეოროთ წინა სავარჯიშოს პირველი 3 პუნქტი.
2. რედაქტირების მიზნით გამოვიყენოთ ფორმის თვისებები. გამოვიტანოთ წინა სავარჯიშოში შექმნილი ფორმა **Form View** რეჟიმში, დავაწკაპუნოთ მარცხენა კიდეზე მონიშვნის არეზე და გამოვიტანოთ თვისებების ფანჯარა **Properties.** ჩავრთოთ ჩანართი **Data** (ნახ.53). თუ თვისება **Allow Edits** (რედაქტირების დაშვება), უარყოფილია (**No**), მაშინ შესწორების შეტანა ფორმაში შეუძლებელი იქნება.



ნახ. 53

3. ანალოგიურად შევამოწმოთ თვისებები **Allow Deletions** და **Allow Additions** მონაცემების გაუქმების და ჩანაწერების დამატების დაშვება/ არ დაშვება.
4. ფორმის რეჟიმში განვახორციელოთ ჩანაწერების დახარისხება კოდების ან სახელების მიხედვით (კურსორი დავაყენოთ შესაბამის ველში). ამისათვის ჩავრთოთ მენიუს **Record → Sort Ascending** (ან **Descending**) პუნქტი. იმავეს შევასრულებთ მენიუდან ან ინსტრუმენტთა პანელზე არსებული ღილაკებით.
5. ფორმის რეჟიმში განვახორციელოთ ჩანაწერების გაფილტვრის ორი ვარიანტი: მონაცემის მონიშვნით **Filter By Selection** და სპეციალური ფორმის გამოყენებით **Filter By Form.** ბრძანებები განვახორციელოთ ან მენიუს **Records** პუნქტიდან ან ინსტრუმენტთა პანელზე არსებული ღილაკებით.
6. დავაყენოთ კურსორი, "დასახელება" ველში, რომელშიც წერია, მაგალითად, "მუკუზანი". ღილაკზე **Filter By Selection** ხელის დაჭერით მთელი ცხრილიდან ამოკრებილი იქნება მხოლოდ მითითებული სახელის შემცველი ჩანაწერები. გამოვრთოთ ფილტრი **Remove Filter** ღილაკით, რათა ფორმა აღდგეს ყველა არსებული ჩანაწერით.
7. დავაყენოთ კურსორი ისევ "დასახელება" ველში. **Filter By Form** ღილაკზე ხელის დაჭერით გამოტანილი იქნება სპეციალური ფორმა (ნახ.54), რომელშიც აქტიური ველის მარჯვენა მხარეს გამოტანილია სიის ჩამოსაშლელი ისარი. სიიდან ამოვარჩიოთ ის



ნახ. 54

სახელი, რომლითაც გვსურს გაფილტვრა, მაგალითად, "მანავი". **Remove Filter** ღილაკით აღვადგინოთ არსებული ფორმა.

8. განვახორციელოთ ორჯერადი ამორჩევა "ან" პირობის გამოყენებით. გავიმეოროთ წინა პუნქტი, არჩევის შემდეგ გადავიდეთ მეორე – **Or** ჩანართზე და ავარჩიოთ, მაგალითად, "მუკუზანი". გაფილტრულ ფორმაში იქნება მხოლოდ შერჩეული სახელების შემცველი ჩანაწერები. აღვადგინოთ ფორმა **Remove Filter** ღილაკით.

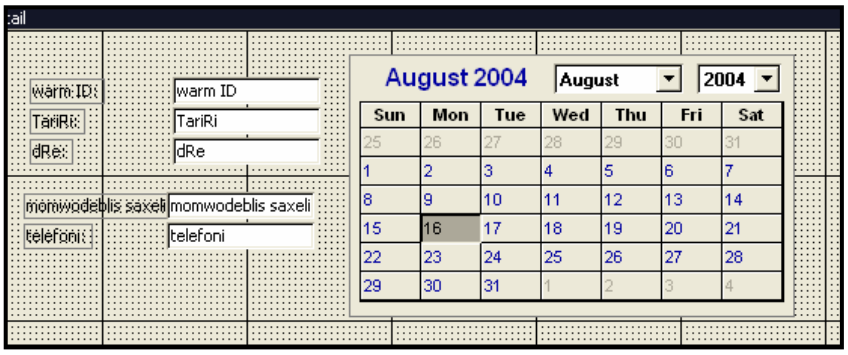
სავარჯიშო 15. ნახატის დამატება და კალენდრის გამოყენება ფორმაზე

შევქმნათ ფორმა ბაზის ცხრილის საფუძველზე, მაგალითად, ჩავტვირთოთ ბაზა **samS masalebi.** ფორმაზე ნახატის და კალენდრის დამატების მიზნით შევასრულოთ შემდეგი პუნქტები:

1. ბაზის ძირითად ფანჯარაში ჩავრთოთ **Forms**, **New.**
2. გამოტანილ დიალოგში ავარჩიოთ **Design View** რეჟიმი, ველში **Choose the table or query …** იმ ცხრილის ან მოთხოვნის სახელი, რომლის საფუძველზე იქმნება ფორმა, მაგალითად, **warmoeba, OK.**
3. ცხრილის მაკეტიდან **Details** არეში გადმოვიტანოთ ველების სია.
4. ფორმას დავამატოთ სტატიკური ნახატი, რისთვისაც **Toolbox**-ზე დავაჭიროთ ხელი **Image** ღილაკს. ჩაირთვება **Insert Picture** ფანჯარა, რომლიდანაც ავარჩევთ წინასწარ შექმნილი ნახატის შემცველ ფაილს, **OK.**
5. დავათვალიეროთ ნახატი ფორმის რეჟიმში სხვადასხვა ჩანაწერზე გადასვლით – იგი ერთი და იგივეა ყველა ჩანაწერისთვის.
6. გავაუქმოთ ველები (მოვნიშნოთ ყველა და დავაჭიროთ ხელი **Delete** კლავიშს), გავზარდოთ ნახატის ზომები ისე, რომ მან ფორმის მთელი არე დაიკავოს. მაკეტიდან გადმოვიტანოთ ველები. შედეგი დავათვალიეროთ ფორმის რეჟიმში – ფორმას ნახატი დადებული ექნება ფონად.

7. კონსტრუქტორის რეჟიმში გავაუქმოთ ნახატი და ფორმას შევუქმნათ სადა ფერადი ფონი: მოვნიშნოთ **Details** ზოლი და მენიუს პუნქტით **Fill → Back Color** ავარჩიოთ ფერი. ფერის შერჩევა შეიძლება **Properties** ფანჯრიდანაც – **Back Color** სტრიქონზე ხელი დავაჭიროთ 3 წერტილს. გამოტანილ ფანჯარაში **Basic Colors**-ში შემოთავაზებული ფერის ნაცვლად შეგვიძლია თავად ავაწყოთ ფერის ტონალობა – გამოვიყენოთ **Custom Colors: Define Custom Colors** ღილაკზე ხელის დაჭერით გავხსნათ ფერების შერჩევის ფანჯარა, ავაწყოთ ფერის ახალი ვარიანტი და **Add to Custom Colors** ღილაკით დავაფიქსიროთ არჩეული ფერი მონიშნულ **Custom Colors** უჯრედში, **OK.** დამატებით დავაწკაპუნოთ ფორმის არეში.
8. კონსტრუქტორის რეჟიმში ფორმას დავამატოთ ნახატი ე.წ. თავისუფალი ჩარჩოს ხერხის გამოყენებით - **Unbound Object Frame.** ჩავრთოთ **Toolbox** პანელზე ამავე სახელწოდების ღილაკი და შემოვხაზოთ მართკუთხედი. გაიხსნება **Insert Object** ფანჯარა. აქ ჩავრთოთ ან **Create New** ღილაკი (თუ ნახატს ვქმნით ფორმაზე), ან **Create from File** (თუ გამოგვაქვს ნახატის შემცველი ფაილი). ჩავრთოთ პირველი ღილაკი, **Object Type** სიაში მოვნიშნოთ, მაგალითად, **Bitmape Image** – ხატვის პროგრამა. შევქმნათ ნახატი – იგი მოთავსდება ფორმაზე და ყველა ჩანაწერისთვის ერთი იქნება.
9. ფორმას დავამატოთ კალენდარი, რათა ველში **TariRi** შესაბამისი თარიღის მონაცემი შევიტანოთ გამარტივებული წესით. კონსტრუქტორის რეჟიმში **Toolbox**-ზე დავაჭიროთ ხელი **More Controls** ღილაკს და გამოტანილ სიაში მოვნიშნოთ **Calendar Control 9.0,** ფორმის თავისუფალ არეში შემოვხაზოთ მართკუთხედი (ნახ.55).

ნახ. 55



TariRi ველში იგი აისახება მომდევნო ველზე გადასვლის შემდეგ.

10. მოვნიშნოთ კალენდარი და დავაჭიროთ ხელი **Properties** ღილაკს, **Data** ჩანართის **Control Source** სტრიქონზე ავარჩიოთ **TariRi** ველი, რომელშიც უნდა იქნას შეტანილი კალენდარზე შერჩეული თარიღი.
11. ფორმის რეჟიმში გადასვლით კურსორი დავაყენოთ **TariRi** ველში, კალენდარზე ავარჩიოთ და ჩავრთოთ შესატანი თარიღი.

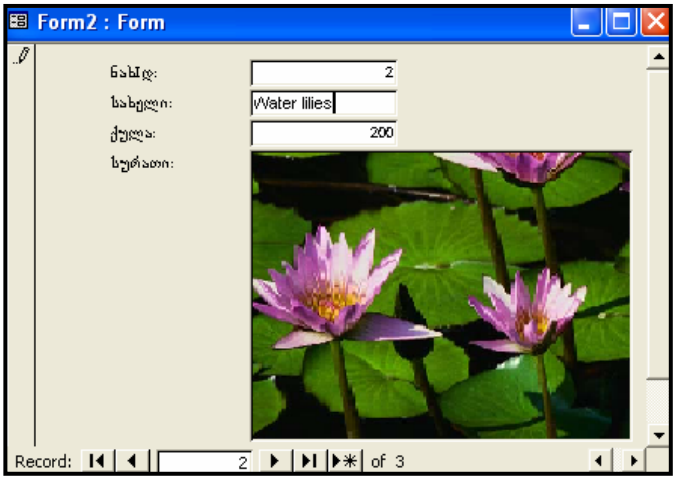
სავარჯიშო 16. ნახატის დაკავშირება ჩანაწერთან, დიაგრამის აგება

ავაგოთ ისეთი ფორმა, რომლშიც თითოეულ ჩანაწერს შეესაბამება განსხვავებული ნახატი. ავაგოთ აგრეთვე ფორმა, რომელზეც მოვათავსებთ ცხრილში არსებული რიცხვითი მონაცემების ამსახველ დიაგრამას. პირველი ამოცანის გადასაწყვეტად წინასწარ მოვამზადოთ მცირე ფორმატის სურათები: გავხსნათ **Start → Programs → Accessories → Paint,** მენიუს **Edit → Paste From** პუნქტით გამოვიტანოთ რომელიმე სურათი, მაგალითად, ნაკრებიდან **Sample Pictures,** მოვნიშნოთ ნაწილი (ფრაგმენტი) და **Edit → Copy To** პუნქტით შევინახოთ ფაილი **My Document** საქაღალდეში სახელით **nax1.** იმავე წესით შევინახოთ კიდევ 2-3 ფრაგმენტი. შევქმნათ კონსტრუქტორის რეჟიმში ცხრილი **naxati** შემდეგი ველებით:

Field Name	Data Type	Description
naxID	Autonumber	
saxeli	Text	
qula	Number	
suraTi	OLE Object	

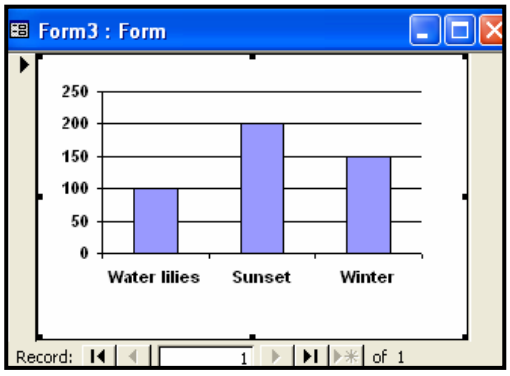
1. მონაცემების ცხრილში შეტანის დროს **suraTi** ველში ყოველ ჩანაწერს ჩავურთოთ შექმნილი ფაილის სახელი (მენიუს **Insert Object** პუნქტით). ამ ცხრილის საფუძველზე აგებულ ფორმაში დავათვალიეროთ ჩანაწერები მიერთებული სურათებით.
2. იმავე ეფექტს მივიღებთ სურათების უშუალოდ ფორმაში შეტანით (ცხრილში წინასწარ სურათების შემცველი ფაილების მიერთების გარეშე): ავაგოთ ფორმა – ცხრილი **naxati-s** საფუძველზე. ფორმის რეჟიმში, ველში **saxeli** შევიტანოთ სურათის სახელი, ველში **qula** – რიცხვი, ველში **suraTi** მენიუს **Insert Object** პუნქტით ან კონტექსტური მენიუდან გამოვიძახოთ დიალოგ-ფანჯარა, რომელშიც ჩავრთოთ ღილაკი **Create from File,** შემდეგ **Browse**, მოვნიშნოთ სასურველი ფაილის სახელი, მაგალითად, **nax1, OK.**

3. იმავე თანმიმდევრობით შევიტანოთ ახალი ჩანაწერები და მივუერთოთ ნახატები.



ნახ. 56

4. დავათვალიეროთ ფორმის რეჟიმში ჩანაწერები შესაბამისი ნახატებით (ნახ.56).
5. დიაგრამის ასაგებად გამოვიყენოთ **naxati** ცხრილში **qula** ველში შეტანილი რიცხვები. ავაგოთ ახალი ფორმა: **Forms, New, Chart Wisard.** გამოტანილ დიალოგში ავარჩიოთ ველები **saxeli** და **qula, Next.**



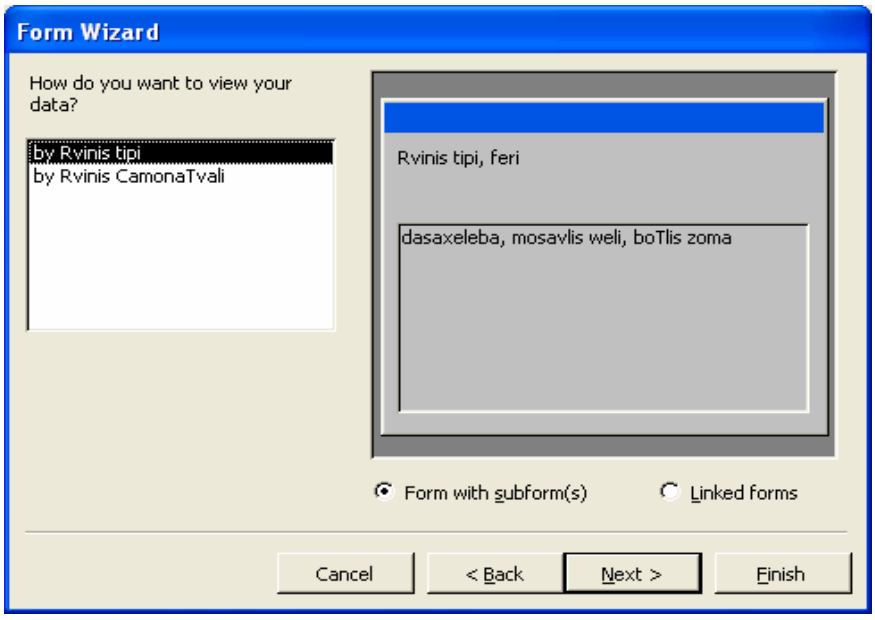
ნახ.57

6. მომდევნო დიალოგებში ავარჩიოთ დიაგრამის ტიპი, მინაწერების სტილი, ჩავრთოთ ღილაკი: **No, don't display a legend** – დიაგრამის გამოტანის რეჟიმი განმატრებითი წარწერების გარეშე, მივანიჭოთ დიაგრამას სახელი, **Finish.**
7. კონსტრუქტორის რეჟიმში დიაგრამას ჩავუტაროთ რედაქტირება არსებული წესების გამოყენებით.

საავარჯიშო 17. რთული ფორმის აგება ოსტატის დახმარებით

რთული ფორმის შესაქმნელად გამოვიყენოთ მოთხოვნა, რომელიც შეიცავს ურთიერთდაკავშირებული ცხრილების მონაცემებს. მაგალითად, ბაზაში "ღვინის სარდაფი" შევქმნათ მოთხოვნა **Query1**, რომელშიც ჩავრთოთ ველები ცხრილებიდან **Rvinis CamonaTvali** და **Rvinis tipi.** შევქმნათ ფორმა ამ მოთხოვნის საფუძველზე. ასეთ ფორმაში შეგვიძლია შევიტანოთ ახალი ან შევცვალოთ არსებული ჩანაწერი.

1. ბაზის ძირითად ფანჯარაში ჩავრთოთ **Forms, New.** ფორმა ავაგოთ ოსტატის გამოყენებით **Form Wizard. Coose the table or query…** ველში მივუთითოთ **Query1, OK.**
2. გამოტანილ დიალოგში მოვნიშნოთ მოთხოვნის სასურველი

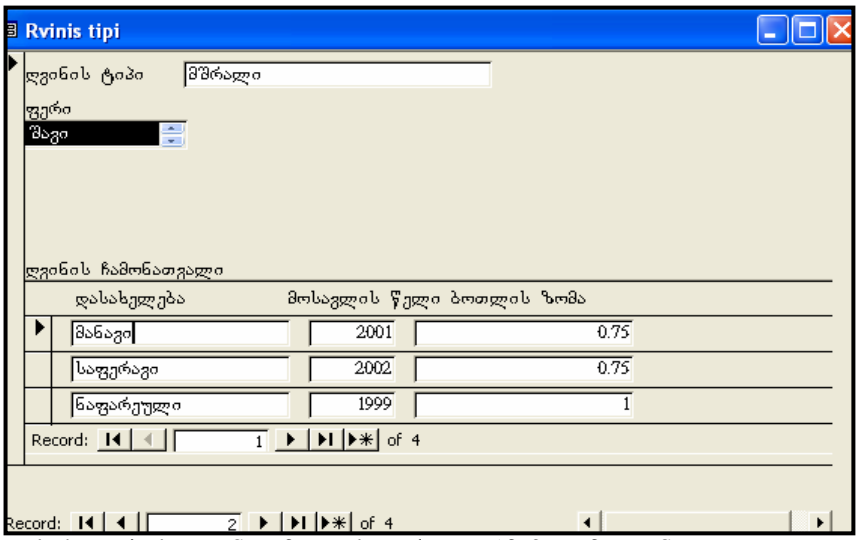


ნახ.58

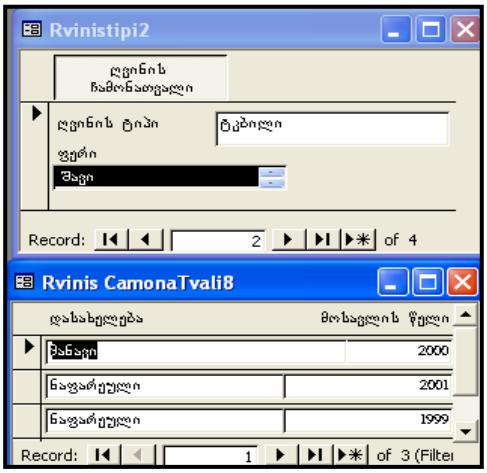
ველები (მაგალითად, დასახელება, მოსავლის წელი, ბოთლის ზომა, ღვინის ტიპი, ფერი), ან ორმაგი ისრით გადავიტანოთ ყველა ველი მარჯვენა მხარეს, **Next.**

3. მომდევნო დიალოგ-ფანჯარაში უნდა მივუთითოთ, თუ რა სტრუქტურით გვსურს ამ მონაცემების ურთიერთგანლაგება (ნახ.58). რადგან მოთხოვნაში **Query1** ველები ორი ცხრილიდანაა ჩართული, რომელთაგანაც ღვინის ტიპი **Relationships** კავშირში "ერთი"-ს მხარეზეა, ხოლო ღვინის ჩამონათვალი "მრავალი"-ს, ამიტომ პირველ კითხვაზე **How do you want to view your data** მოვნიშნოთ **by Rvinis tipi,** ფანჯრის ქვედა ნაწილში დავტოვოთ ჩართულ მდგომარეობაში ღილაკი **Form with subform(s)** (ფორმის აგება ქვეფორმით), **Next.**
4. მომდევნო დიალოგებში შევარჩიოთ სტილი, მაგალითად, **Tabular** და **Standard, Next.**

ნახ. 59



5. ბოლო საფეხურზე ოსტატი გვთავაზობს ფორმის და ქვეფორმის სახელებს, **Finish.**
6. რთულ ფორმაში მუშაობის თანმიმდევრობა ასეთია: მაგალითად, ნახატზე (ნახ.59) ღვინის ტიპისთვის აღებულია მე-2 ჩანაწერი (სულ ამ ცხრილში 4 ჩანაწერია). ქვეფორმა "ღვინის ჩამონათვალ"-ში ამ ტიპს (მშრალი, შავი) შეესაბამება



3 დასახელების ღვინო, მათი სია ქვეფორმაზეა მოყვანილი.

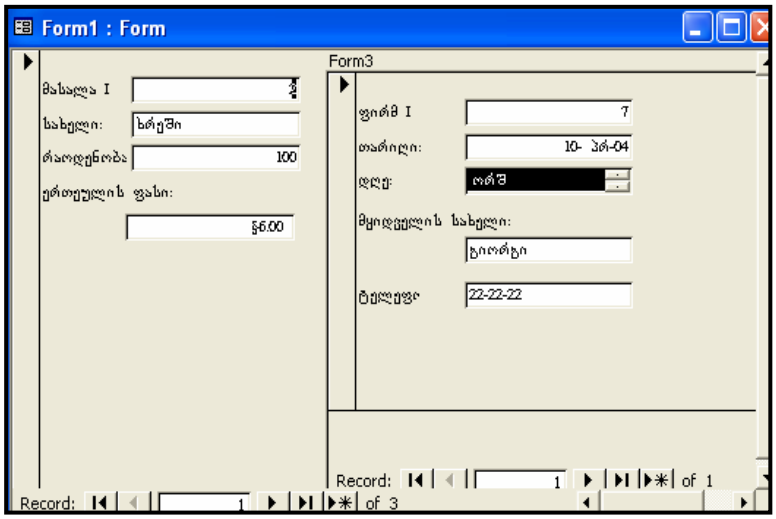
ნახ. 60

7. ისევ ოსტატის გამოყენებით ავაგოთ ახალი ფორმა მითითებული მოთხოვნის **Query1** საფუძველზე. ჯერ გავიმეოროთ სავარჯიშოს პირველი 2 პუნქტი.
8. მე-3 პუნქტში მითითებულ დიალოგ-ფანჯარაში მოვნიშნოთ **Rvinis CamonaTvali** ცხრილი, ანუ ის, რომელიც "მრავალი"-ს მხარეზეა. ჩაირთვება ღილაკი **Single Form** და მომდევნო დიალოგების გავლის შემდეგ მივიღებთ მარტივ ფორმას, რომელზეც გამოტანილი იქნება ორივე ცხრილის ველები.
9. გავიმეოროთ პირველი 2 პუნქტი, მე-3 პუნქტში მითითებულ დიალოგ-ფანჯარაში მოვნიშნოთ **Rvinis tipi** და ჩავრთოთ ღილაკი **Linked Forms.** მომდევნო ფანჯრების ჩვეულებრივი წესით გავლის შემდეგ მივიღებთ ორ ურთიერთდაკავშირებულ ფორმას. ღვინის ტიპის ფანჯარაზე ოსტატი განათავსებს ღილაკს "ღვინის ჩამონათვალი", რომლის ჩართვისას გამოვიტანთ მის შესაბამის ფორმას (ნახ.60).

სავარჯიშო 18. რთული ფორმის აგება კონსტრუქტორის რეჟიმში

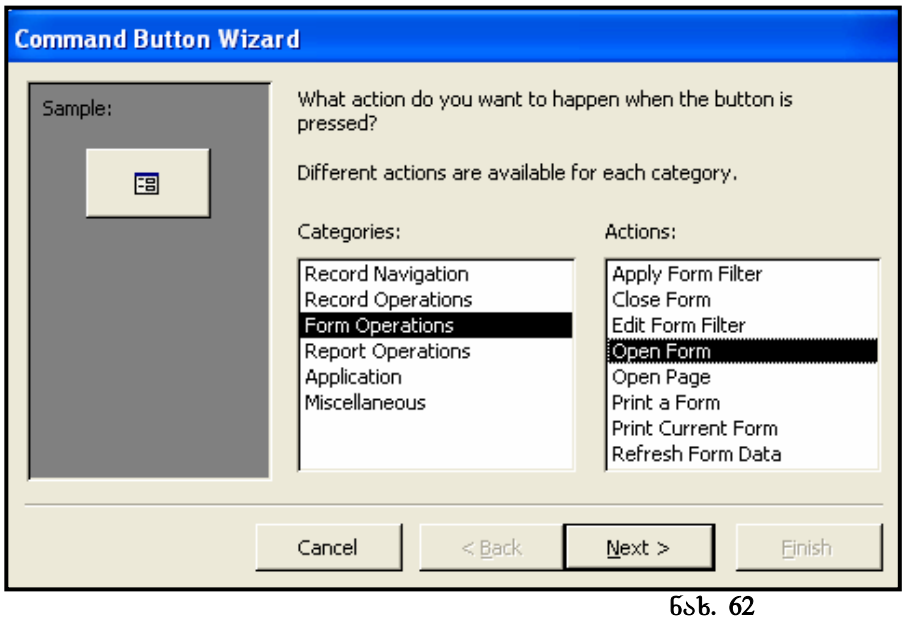
ავაგოთ რთული ფორმა 2 ცხრილის საფუძველზე, რომელთა შორის "ერთი-მრავალთან" კავშირია დამყარებული, მაგალითად, ცხრილების **masala** და **firma** მონაცემთა ბაზა **samS masalebi** -დან. ერთის მხარეზეა **masala,** მრავლის – **firma.** შევქმნათ ჯერ მარტივი ფორმა **firma** ცხრილის საფუძველზე, შემდეგ კი **firma** ცხრილის ფორმა მოვათავსოთ **masala** ცხრილის ფორმაზე. შევასრულოთ შემდეგი პუნქტები:

1. ჩავტვირთოთ მონაცემთა ბაზა **samS masalebi.** კონსტრუქტორის რეჟიმში შევქმნათ მარტივი ფორმა **firma** ცხრილის საფუძველზე.



ნახ.61

2. შეექმნათ ძირითადი ფორმა **masala** ცხრილის საფუძველზე. ფორმის არეში ველების გადმოტანის შემდეგ **Toolbox**-ზე ჩავრთოთ ქვეფორმის (ან დაქვემდებარებული ანგარიშის) დასამატებელი ღილაკი - **Subforms/ Subreports.** ფორმის თავისუფალ არეში შემოვხაზოთ მართკუთხედი.
3. ჩაირთვება ოსტატის პროგრამა, რომელიც პირველ დიალოგ- ფანჯარაზე გვთავაზობს იმ ფორმის არჩევას (**firma** ცხრილის შესაბამისს), რომელიც უნდა დაუკავშირდეს ძირითად ფორმას. ჩავრთოთ მეორე ღილაკი და მოვნიშნოთ **firma** ცხრილის შესაბამისი ფორმა.
4. მომდევნო დიალოგში დავტოვოთ ჩართული ღილაკი **Choose from a list, Next,** ბოლოს **Finish.**
5. დავათვალიეროთ შექმნილი რთული ფორმა (ნახ.61), რომელსაც შესაბამისობაში მოჰყავს ურთიერთდაკავშირებული ჩანაწერები.
6. შეექმნათ **masala** ცხრილის საფუძველზე ახალი ფორმა. მასთან დაკავშირებული **firma** ცხრილის ფორმის გახსნა განვახორციელოთ ჩვენს მიერ სპეციალურად დამატებული ღილაკით. ამისათვის ცხრილის ველების გადმოტანის შემდეგ **Toolbox**-ზე ჩავრთოთ ღილაკი **Command Button.**
7. ჩაირთვება ოსტატის პროგრამა, დიალოგ-ფანჯრის მარცხენა მხარეს შევარჩევთ **Form Operations** პუნქტს, მარჯვენა მხარეს – **Open Form** (ნახ.62), **Next.**

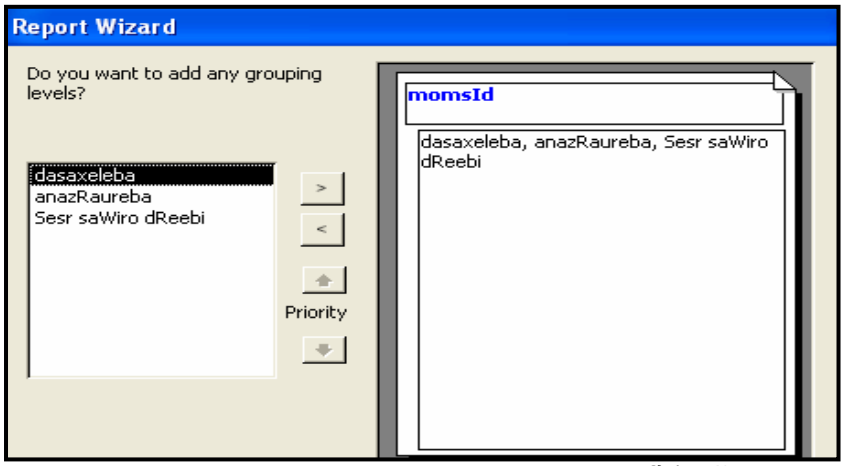


ნახ. 62

8. მომდევნო ფანჯარაზე მოვნიშნოთ ღილაკით გამოსატანი ფორმის სახელი, **Next.** დავტოვოთ ჩართული ღილაკი **Open the form and show all the records, Next.**
9. მომდევნო დიალოგ-ფანჯარაზე ჩავრთოთ ღილაკი **Text,** თუ ღილაკს უნდა გაუკეთდეს წარწერა, ან **Picture,** თუ ღილაკის გაფორმება გვსურს ნახატით, **Finish.**
10. ფორმის რეჟიმში შევამოწმოთ ძირითადი ფორმიდან დაკავშირებული ფორმის ჩართვის პროცესი დამატებული ღილაკის მეშვეობით.

სავარჯიშო 19. ანგარიშის შექმნა ოსტატის გამოყენებით

ოსტატის გამოყენებით მონაცემთა ბაზის ანგარიშის შექმნისათვის შეასრულოთ შემდეგი:



ნახ. 63

1. ჩავტვირთოთ ბაზა, მაგალითად, **samkervalo.** ბაზის ძირითად ფანჯარაში ჩავრთოთ **Reports** ღილაკი და ორჯერ დავაწკაპუნოთ **Create report by using wizard** ღილაკზე, ან **Reports New Report Wizard.**
2. ამუშავდება ოსტატის პროგრამა, რომლის პირველ დიალოგ- ფანჯარაზე ველის **Tables/ Queries** სიიდან ავარჩიოთ იმ ცხრილის სახელი, რომლის საფუძველზეც იქმნება ანგარიში, მაგალითად, **TanamSromlebi**. ოსტატის მეორე დიალოგ- ფანჯარაში ცხრილის ყველა ველი გადავიტანოთ მარცხნიდან მარჯვნივ, **Next.**
3. მომდევნო დიალოგ-ფანჯარაზე განვსაზღვროთ ველების პრიორიტული განლაგება (ან დაჯგუფების წესი) მათი ანგარიშზე თვალსაჩინოდ წარმოჩენის მიზნით (ნახ.63), **Next.**
4. მომდევნო დიალოგ-ფანჯრებზე შეგვიძლია განვსაზღვროთ სიდიდეების ზრდადობით (თუ ტექსტია – ანბანით) ან კლებადობით გამოტანის წესი, ავარჩიოთ აგრეთვე ანგარიშის ფურცელზე მათი წყობა, სტილი, **Next.**
5. შესაძლოა, ანგარიში არ მოთავსდეს ერთ გვერდზე, ან ველის მინაწერის ადგილმდებარეობა არ იყოს თვით მონაცემის მიმართ შეფარდებული. ასეთ შემთხვევაში ჩავრთავთ **Design** რეჟიმს და ჩვენთვის ცნობილი წესების გამოყენებით ჩავატარებთ მის რედაქტირებას.
6. მიღებული ანგარიში შეგვიძლია გავიტანოთ პრინტერზე.

სავარჯიშო 20. ანგარიშის შექმნა კონსტრუქტორის რეჟიმში

შევქმნათ ანგარიში კონსტრუქტორის რეჟიმში, გამოვიტანოთ რიცხვითი მნიშვნელობების ჯამი, საშუალო არითმეტიკული და სხვა ინფორმაცია. შეასრულოთ შემდეგი:

1. ჩავტვირთოთ, მაგალითად, მონაცემთა ბაზა **samkervalo**, ჩავრთოთ ღილაკი **Reports, New.** ავარჩიოთ **Design View** რეჟიმი, ანგარიში შეასრულოთ ცხრილის **SemoTavazebuli samuSaoebi** საფუძველზე, **OK.**
2. ფორმის მსგავს ანგარიშის არეში სამი განყოფილებაა: **Page Header, Details** და **Page Footer.** ცხრილის მაკეტიდან მოვნიშნოთ და გადმოვიტანოთ ანგარიშის **Details** განყოფილებაში ველები: **dasaxeleba, gadasaxdeli fasi** და **SesrulebisaTvis saWiro dReebi.**

ნახ. 64

3. ველების მინაწერები მოვათავსოთ ანგარიშის სათავეში – მოვნიშნოთ ისინი, “ამოვჭრათ” და გადავიტანოთ **Page Header** არეში მოვნიშნოთ მინაწერები ერთობლივად (**Shift** კლავიშის გამოყენებით), შევურჩიოთ მათ ქართული შრიფტი, სასურველი სტილი, ზომა და სწორება ზედა კიდის მიმართ: **Format→Align→Top.**
4. მნიშვნელობების გამოსატანი ველები მოვათავსოთ სათავეების ქვემოთ.
5. ანგარიშის ერთიანი სათაურის ან შენიშვნების მოთავსების მიზნით დავამატოთ შესაბამისი არეები: **View→Report Header/ Report Footer. Report Header** არეში **Toolbox**-დან **Label** ღილაკის ჩართვით შემოვხაზოთ სათაურის არე და შევიტანოთ სათაურის ტექსტი.
6. ჯამის მნიშვნელობა მოვათავსოთ **Report Footer** არეში, ამისათვის შევიტანოთ ტექსტი “სულ:”, ხოლო რიცხვების ჯამის გამოსათვლელად **Toolbox**-დან **Text Box** ღილაკით შემოვხაზოთ არე, იგი მოინიშნება. ჩავურთოთ მას **Properties** დიალოგ- ფანჯარა: **Data** ჩანართის **Control Source** სტრიქონზე დავაჭიროთ ხელი 3 წერტილს, ჩაირთვება **Expression Builder** ფანჯარა, რომელშიც ჩავრთავთ **Sum** ფუნქციას ველისთვის **gadasaxdeli fasi** ცხრილიდან **SemoTavazebuli samuSaoebi.** იმავე წესით გამოვითვალოთ ჯამური მნიშვნელობა ველისთვის **Sesr saWiro dReebi.**
7. დავათვალიეროთ ანგარიში **View** რეჟიმში, რედაქტირებისათვის დავბრუნდეთ კონსტრუქტორის რეჟიმში.

ლიტერატურა

1. Джон Вейскас , Эффективная работа с **Microsoft Access 2000**, Питер, Санкт-Петербург , 2001.
2. Информатика для экономистов и юристов , под. ред. К. Симоновича , Питер, Санкт-Петербург , 2002.

შინაარსი

შესავალი 2
მონაცემთა ბაზის დაპროექტების ძირითადი პრინციპები 3
ბაზის უსაფრთხოება 4
მონაცემთა ბაზის მართვის სისტემა Microsoft Access 4
Microsoft Access-ის არქიტექტურა 5
კავშირები ცხრილებს შორის 6
მონაცემთა ბაზის აგება Microsoft Access -ში 7
ცხრილის შექმნა კონსტრუქტორის რეჟიმში 8
მონაცემთა ტიპები 8
ცხრილის ველების თვისებები 9
ცხრილის შექმნა მონაცემთა შეტანის წესით 10
მოთხოვნები 10
მოთხოვნის შექმნა კონსტრუქტორის რეჟიმში 11
გამოსახულების შედგენა სპეციალური პროგრამის გამოყენებით .. 13
მონაცემთა იმპორტი, ექსპორტი 14
სავარჯიშო 1. მონაცემთა ბაზის შექმნა ოსტატის გამოყენებით .. 14
სავარჯიშო 2. ცხრილის შექმნა კონსტრუქტორის რეჟიმში 16
სავარჯიშო 3. ცხრილის ველების თვისებების რედაქტირება 17
სავარჯიშო 4. ცხრილებს შორის კავშირების დამყარება 18
სავარჯიშო 5. გადამკვეთი ცხრილის შექმნა “მრავალი მრავალთან” კავშირის განსახორციელებლად; კავშირი “ერთიერთთან” 19
სავარჯიშო 6. მონაცემთა ბაზა “სამისამართო წიგნი”. მოთხოვნის შექმნა მონაცემების ამორჩევაზე 20
სავარჯიშო 7. მონაცემთა ბაზა “სამისამართო წიგნი”. მოთხოვნიდან ჩანაწერების ამოკრება გარკვეული პირობებით 22
სავარჯიშო 8. მონაცემთა ბაზა “ღვინის სარდაფი” 22
სავარჯიშო 9. მონაცემთა ბაზა “ღვინის სარდაფი”. მოთხოვნების შექმნა 24
სავარჯიშო 10. მონაცემთა ბაზა “სამკერვალო” 26
სავარჯიშო 11. მონაცემთა ბაზა “სამკერვალო”. სხვადასხვა ხასიათის მოთხოვნების შექმნა 27
ფორმები 28
ავტოფორმები 29
ფორმის სტრუქტურა 30
ფორმის მართვის ელემენტები 30
ფორმის განყოფილებების და ელემენტების თვისებები 31
ფორმაში მონაცემების მოძიება, დახარისხება, გაფილტვრა 32
მარტივი ფორმის აგება კონსტრუქტორის რეჟიმში 33
რთული ფორმის შექმნა 35
ანგარიშები 35
სავარჯიშო 12. ფორმის შექმნა ოსტატის გამოყენებით 37
სავარჯიშო 13. მარტივი ფორმის აგება კონსტრუქტორის რეჟიმში 37
სავარჯიშო 14. მარტივი ფორმის რედაქტირება, მონაცემების მოწესრიგება, ფილტრის გამოყენება 38
სავარჯიშო 15. ნახატის დამატება და კალენდრის გამოყენება ფორმაზე 39
სავარჯიშო 16. ნახატის დაკავშირება ჩანაწერთან. დიაგრამის აგება 40
სავარჯიშო 17. რთული ფორმის აგება ოსტატის დახმარებით 41
სავარჯიშო 18. რთული ფორმის აგება კონსტრუქტორის რეჟიმში. 42
სავარჯიშო 19. ანგარიშის შექმნა ოსტატის გამოყენებით 43
სავარჯიშო 20. ანგარიშის შექმნა კონსტრუქტორის რეჟიმში 44
ლიტერატურა 45