საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, ლილი პეტრიაშვილი

მონაცემთა საცავის აგების ტექნოლოგია ინტერნეტული ბიზნესის სისტემებისთვის

დამტკიცებულია სტუ-ს
სარედაქციო-
საგამომცემლო
საბჭოს მიერ

თბილისი 2005

უაკ 681.3.06

გადმოცემულია ელექტრონული ბიზნესისა და კომერციის განაწილებული მართვის საინფორმაციო სისტემებისთვის მონაცემთა საცავების დაპროექტებისა და რეალიზაციის ტექნოლოგია თანამედროვე კომპიუტერული და საინფორმაციო პროგრამული პაკეტების ინტეგრირებული გამოყენებისა და ობიექტ-ორიენტირებული მიდგომის საფუძველზე.

მონოგრაფიაში შემოთავაზებულია ინტერნეტული ბიზნესის განაწილებული ობიექტების კლასიფიკაციის, მართვის პრობლემების კვლევისა და მათი გადაწყვეტის ახალი მექანიზმების შემუშავების ამოცანები ინტეგრირებული ინფორმაციული საცავების აგების საფუძველზე. გამოკვლეულია მათი ოპტიმალური ფუნქციონირების მახასიათებლები მასობრივი მომსახურების მეთოდებისა და სტოქასტიკური პეტრის ქსელების ბაზაზე. სარეალიზაციოდ გამოყენებულია რელაციური, ობიექტ-ორიენტირებულ მონაცემთა ბაზები (MS SQL Server, MS Access, InterBase), ვინდოუსის (MS_Office, BorlandC++Builder, C#), Web-აპლიკაციების (ASP.NET, ADO.NET, XML) და .NET ტექნოლოგიები.

რეცენზენტები: პროფ. ი. მიქაძე
პროფ. ზ. გასიტაშვილი

პროფ. გ. სურგულაძის რედაქციით

ISBN 99940-40-36-7

Georgian Technical University

Gia Surguladze, Lily Petriaschvili

TECHNOLOGY OF BUILDING DATA WAREHOUSES FOR E-COMMERCE SYSTEMS

Supported by DAAD
(Germany)

The problems of organization, design and realization of commercial distributed objects are considered with integrated use of modern object-oriented methods and instrumental means. On the basis of open and closed models of mass service the analysis of the developed network commercial control system indices is done. Using Markovian processes the model of service organs optimum number determination of user-server architecture and algorithmic schemes of its solution are worked out. The problems of internet commercial systems functioning realized with .NET technology (C#, ADO.NET, ASP.NET) and instruments of visual modeling and programming are described.

ISBN 99940-40-36-7

სარჩევი

ნაშრომი სრულდება გერმანიის აკადემიური გაცვლების სამსახურის (DAAD) პროექტის მხარდაჭერით

შესავალი

კომპიუტერული და ქსელური ინდუსტრიის განვითარებამ ბოლო ათწლეულებში მნიშვნელოვან შედეგებს მიაღწია, რამაც თითქმის მთლიანად შეცვალა მართვის საინფორმაციო სისტემების აგების ტექნოლოგია და ინსტრუმენტული საშუალებანი, გაჩნდა ახალი ცნებები და ტერმინები, რომლებიც თანამედროვე კონცეფციებსა და პროექტებს ედება საფუძვლად.

ორგანიზაციული სისტემების მართვის პრობლემების და ამოცანების გადასაწყვეტად საჭირო ხდება ახალი კომპიუტერული და ინფორმაციული ტექნოლოგიების გამოყენება, რომელთა შორის ერთ-ერთი აქტუალური და მნიშვნელოვანი მიმართულებაა მონაცემთა საცავების (data warehouse) აგება. ბოლო პერიოდში იგი ითვლება, განსაკუთრებით პერსპექტიულ და დინამიკურ მიმართულებად საინფორმაციო სისტემების დაპროექტებაში, იგი უკავშირდება მრავალდონიან განაწილებულ საინფორმაციო სისტემის შექმნას.

მონაცემთა საცავი განიხილება როგორც რომელიმე კონკრეტული ორგანიზაციის ან დიდი საწარმოსთვის განკუთვნილი სპეციალური სუპერ-ბაზა, სადაც მიმდინარე ოპერატიული სამუშაოს შესრულებისას თავს იყრის ქრონოლოგიურ ინფორმაციათა მთელი სპექტრი, რომელთა დანიშნულებაცაა მომხმარებლისთვის ინტერნეტ გვერდებზე მიზნობრივად განლაგებული ტექსტური, გრაფიკული და აუდიო-ვიზუალური საინფორმაციო ბლოკების მიწოდება.

თავისუფალ საბაზრო ეკონომიკის პირობებში ბიზნესის მართვის სისტემებისათვის გადაწყვეტლებათა მიღების ხელშემწყობი ეფექტური მექანიზმების დამუშავება და კვლევა თანამედროვე ქსელური საინფორმაციო ტექნოლოგიების ინტეგრირებული გამოყენებით, ერთ-ერთი მნიშვნელოვანი მიმართულებაა. ელექტრონული ბიზნესისა და კომერციის მომხმარებელს მიეცემა ტექნიკური საშუალება ინტერაქტიულ-დიალოგური კრიტერიუმებთ განსაზღვროს მისთვის საჭირო ინფორმაცია. ამ მიზნით ანალიზური პროცესების სისტემა ეყრდნობა მრავალგანზომილებიან მონაცემთა სტრუქტურებს.

სხვადასხვა საინფორმაციო წყაროებიდან მიღებული მონაცემები ტრანსფორმაციის, კონვერტაციის და ინტეგრირების შემდეგ თავსდება საცავის ობიექტ-ორიენტირებულ მონაცემთა ბაზებში (MS SQL Server, MS Access, InterBase ან სხვ.), ვინდოუსის(MS_Office, BorlandC++Builder, C#), Web-აპლიკაციების (ASP.NET, ADO.NET, XML) .NET ტექნოლოგიების საფუძველზე.

წინამდებარე მონოგრაფიაში შემოთავაზებულია ავტორთა მიერ საქართველოს ტექნიკურ უნივერსიტეტსა და გერმანიის უნივერსიტეტებში (ბერლინის ჰუმბოლდტისა და ნიურნბერგ-ერლანგენის ინფორმატიკის ინსტიტუტებში) მიღებული თეორიული და ექსპერიმენტული სამუშაოების სამეცნიერო შედეგები, კერძოდ:

- კვლევის შედეგები ელექტრონული ბიზნესისა და კომერციის მართვის საინფორმაციო სისტემების დაპროექტების თანამედროვე მეთოდებისა და ინსტრუმენტების შესახებ, მათი კლასიფიკაციისა და სტრუქტურული ანალიზის საფუძველზე;

- ბიზნესის განაწილებული მართვის სისტემებისთვის მონაცემთა საცავების ობიექტ-ორიენტირებული დაპროექტების და რეალიზაციის მეთოდები და ინსტრუმენტული საშუალებანი საინფორმაციო ობიექტებისა და მეტაინფორმაციის ლოგიკურად მთლიანი ორგანიზებისა და მიზნობრივი დამუშავების კლასთა-ფუნქციების საფუძველზე;

- განაწილებული მართვის საინფორმაციო სისტემების მახასიათებელთა კვლევის შედეგები მასობრივი მომსახურების მოდელების მრავალარხიანი ღია და ჩაკეტილი სისტემებით. ბიზნესისა და კომერციის განაწილებულ ობიექტებზე, კლიენტ-სერვერ არქიტექტურისათვის გამოკვლეულია მოგების ოპტიმიზაციის საკითხები;

- ბიზნესის განაწილებული საინფორმაციო სისტემების რესურსების ეფექტური მართვის დინამიკური მოდელი სტოქასტიკური პეტრის ქსელის გრაფო-ანალიზური ინსტრუმენტით;

- მონაცემებისა და მათი სტრუქტურების კლასტერიზაციის, კონვერტირებისა და აგრეგაციის ალგორითმული სქემები და პროგრამები;

- ბიზნეს-ობიექტების მომხმარებელთა მოთხოვნების წინმსწრები ანალიზისა და მათი დამუშავების სინქრონიზაციის რელაციური სქემები პეტრის ქსელის გრაფული მოდელების საფუძველზე.

ნაშრომის თეორიული და პრაქტიკული შედეგების გამოყენება შესაძლებელია:

– საწარმოებისა და ორგანიზაციების ბიზნესის მართვის სფეროში განაწილებული ინფორმაციული სისტემების ასაგებად და გადაწყვეტილების მიღების ხელშემწყობი მექანიზმების სრულყოფისათვის;

– განაწილებული სისტემების კომპიუტერული და ქსელური რესურსების ეფექტური მართვის ექსპერტული სისტემების ასაგებად;

– მთლიანი ქსელისა და მისი ცალკეული კომპონენტების დატვირთვის, მწარმოებლურობის, სისტემის მდგომარეობათა ალბათობების და სხვა მაჩვენებლების გასათვლელად, მოთხოვნათა შემოსვლისა და მათი მომსახურების ინტენსიურობების და ეკონომიკური მაჩვენებლების გათავალისწინებით.

შესავალში განხილულია მართვის საინფორმაციო სისტემების აგების თანამედროვე ტექნოლოგიებისა და ინსტრუმენტულ საშუალებათა დამუშავებისა და სრულყოფის საკითხების აქტუალობა და მნიშვნელობა. ჩამოყალიბებულია ნაშრომის ძირითადი მიზნები, ამოცანები, მათი გადაწყვეტის გზები და საბოლოო შედეგები.

პირველ თავში წარმოდგენილია დიდი საინფორმაციო სისტემების დაპროექტების თანამედროვე ტექნოლოგიების ანალიზი, რომელთა ძირითად სტრუქტურულ კომპონენტსაც მონაცემთა საცავი წარმოადგენს. მისი დამკვიდრება გამოწვეული იყო იმ აუცილებლობით, რომელიც განაპირობა მართვის დიდ და რთულ სისტემებში განაწილებული ინფორმაციული ბაზების ლოგიკურად ერთიან მონაცემთა ,,საწყობში“ თავმოყრამ. მონაცემთა საცავში ინახება კომპანიების მუშაობის ისტორიის მთელი საინფორმაციო სპექტრი. ისტორიული მონაცემების შენახვა არის ერთ-ერთი აუცილებელი პირობა და იგი მონაცემთა საცავის მთავარი ღირსებაა.

მომხმარებელს საშუალება ეძლევა ინტერნეტ გვერდებზე არსებული ინფორმაციის დახმარებით ეფექტურად და მიზნობრივად იმოქმედოს ბიზნესის სფეროში. საჭირო ხდება ენერგიის, დროის და, რა თქმა უნდა, ფინანსური რესურსების დაზოგვის ღონისძიებების გატარება, რისი უნიკალური საშუალებაცაა ელექტრონული კომერციის (E-Commerce) გამოყენება. როგორც ცნობილია, იგი გამოიყენება ინტერნეტ-ტექნოლოგიის დახმარებით და უზრუნველყოფს კომპიუტერის მონიტორიდან უშუალოდ მსოფლიო ბაზარზე

ყიდვა–გაყიდვის წარმოებას. დისერტაციაში აღწერილია ელექტრონული კომერციის ინფრასტრუქტურული სისტემა. წარმოდგენილია ეკონომიკური მოვლენის განმსაზღვრელი ცვალებადი ფაქტორები. გაანალიზებულია დამოკიდებულება პროდუქციის მიწოდებას, მოთხოვნასა და ფასს შორის თავისუფალ საბაზარო ეკონომიკის პირობებში.

მეორე თავში შემოთავაზებულია განაწილებული მართვის სისტემების მონაცემთა საცავების დაპროექტების ეტაპების, მისი ძირითადი კომპონენტების, ფუნქციებისა და კლასიფიცირებულ მონაცემთა მეტაინფორმაციის ეფექტური ორგანიზაციის, გარდაქმნისა და ძებნის მეთოდებისა და ალგორითმების აღწერა.

მონაცემთა საცავი ორიენტირებულია საგნობრივ სფეროზე და ორგანიზებულია მონაცემთა ოპერატიული ბაზიდან შემოსულ სტრუქტურულად გადამუშავებულ მონაცემთა ქვესიმრავლეების საფუძველზე. ინფორმაციის წყაროს წარმოადგენს სხვადასხვა ორგანიზაციათა დანართები (აპლიკაციები), რომლებიც გამოიყენებს განსხვავებულ პროგრამულ პლატფორმებს და უკავშირდება ოპერატიულ მონაცემთა ბაზას ინტერნეტის საშუალებით (on-line რეჟიმი).

მესამე თავში განხილულია ელექტრონული ბიზნესისა და კომერციის განაწილებულ ობიექტებზე კომპიუტერული ქსელის ფუნქციონირების პროცესების მოდელირებისა და ანალიზის საკითხები მასობრივი მომსახურების სისტემებისა და პეტრის ქსელების გამოყენებით.

მეოთხე თავში წარმოდგენილია განაწილებული ბიზნეს-ობიექტების სისტემების მონაცემთა საცავების საინფორმაციო ბლოკებისა და მათი დამუშავების მეთოდების პროგრამული რეალიზაციის საკითხები. განიხილება გლობალურ-ლოკალური ქსელის ტექნიკური უზრუნველყოფის საკითხები და მასში ინფორმაციის დაცვისა და აღდგენის საშუალებანი.

წარმოდგენილია web-გვერდებზე საინფორმაციო ბლოკების მიზნობრივად განლაგებისა და მოხმარებელთა შესაბამისი ინტერფეისების აგების ხერხები, მაიკროსოფთის უახლესი საინფორმაციო ტექნოლოგიის დოტ-NET პლატფორმის საფუძველზე, კერძოდ ASP.NET, ADO.NET, XML და C#.NET პაკეტებით.

ავტორები მადლობას უხდიან მკითხველებს და იტოვებენ იმედს მათგან საქმიანი შენიშვნების მისაღებად, მისამართზე: gsurg@gmx.net, liapetri@Rambler.ru.

თავი I. ბიზნესის მართვის განაწილებული საინფორმაციო სისტემების ანალიზი

1.1. ქსელური ინტეგრირებული საინფორმაციო მართვის სისტემების დაპროექტების ამოცანა

თანამედროვე კომპიუტერული ტექნიკისა და ტექნოლოგიების განვითარების პირობებში განაწილებული მართვის ავტომატიზებული სისტემების ერთ-ერთ მთავარ კომპონენტად მონაცემთა საცავებს (Data Warehouse, Repository) მიიჩნევენ [35]. აქ თავმოყრილია საავტომატიზაციო-საკვლევი მართვის ობიექტის სხვადასხვა ტიპის საინფორმაციო ბლოკები: ტექსტები და სუპერტექსტები, ცხრილები და გრაფიკული დიაგრამები, აუდიო და ვიზუალური მასალები და სხვ. ამგვარად, მონაცემთა საცავი აღნიშნული ტიპების ფაილთა მართვის სისტემის რთულ კონგლომერატს წარმოადგენს.

მონაცემთა შეკრება პირველადი წყაროებიდან, მათი გაფილტრა (მეთოდურად, სემანტიკურად და ტექნიკურად), ტრანსფორმაცია და კონვერტაცია (მონაცემთა წინასწარ განსაზღვრული სტრუქტურების მისაღებად), მეტაინფორმაციის იერარქიულად ორგანიზება (მონაცემთა კატალოგებისა და არქივების სამართავად), სტანდარტული და გამოყენებითი პროგრამული პაკეტების შექმნა (მონაცემთა ფუნქციურ-მიზნობრივ დასამუშავებლად), ინტერნეტ-გვერდების უახლესი საინფორმაციო ტექნოლოგიებით დაპროექტება (ფართო მომხმარებელთა მარტივი ინტერფეისების ასაგებად).

და ბოლოს, თვით განაწილებული საინფორმაციო სისტემის კომპიუტერული რესურსების ეფექტური მართვის საკითხების კვლევა (ქსელის არხები, კლიენტ-სერვერული პროცესორები, საინფორმაციო ბლოკები, ბაზები და ფაილები, მათი დაცვისა და განახლების მოდულები და ა.შ.) – უმნიშვნელოვანეს ამოცანათა კლასს შეადგენს თანამედროვე მართვის ინტეგრირებული სისტემების ასაგებად.

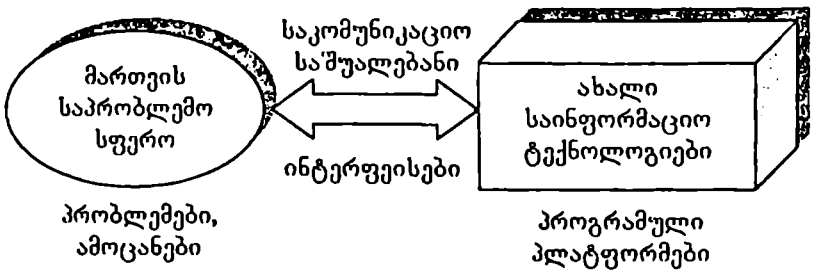
წინამდებარე წიგნის მიზანი სწორედ ამ საკითხების კვლევა და დამუშავებაა.

განაწილებული სისტემების დასაპროექტებლად და სარეალიზაციოდ ჩვენ ვიყენებთ დაპროგრამების ობიექტ-ორიენტირებულ და სტრუქტურულ მეთოდებს, უნიფიცირებული მოდელირების ენის UML–ტექნილოგიას [39,40,41].

1.1 ნახაზზე მოცემულია „ადამიანი-მანქანა“ ტიპის

ინტეგრირებული საინფორმაციო მართვის სისტემის (ისმს) ზოგადი სტრუქტურული სქემა. მოცემულ კონტექსტში კვლევის საგანს ამ სისტემის კომპონენტთა (საინფორმაციო ბლოკები, პროგრამული პაკეტები, ინტერფეისები და სხვ.) სიმრავლე და მათი ურთიერთკავშირები წარმოადგენს თანამედროვე საკომუნიკაციო და პროგრამული გარდამქმნელების ბაზაზე.

1.2 ნახაზზე მოცემულია ზემოგანხილული ინტეგრირებული



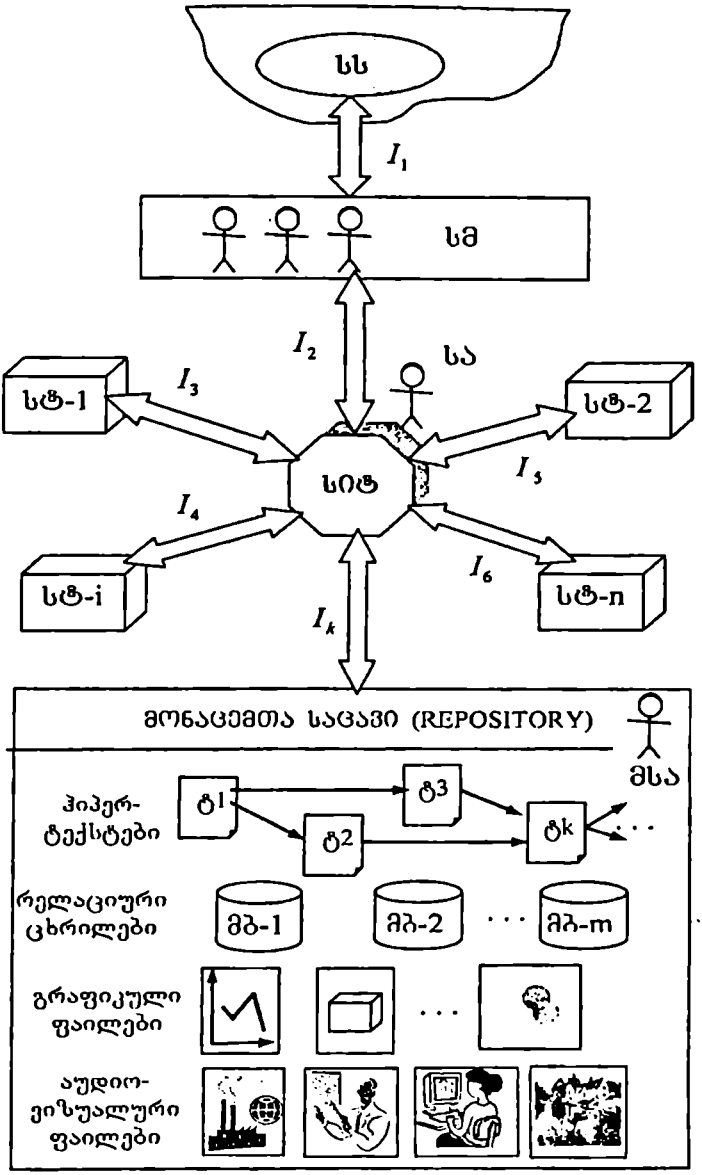
ნახ.1.1. ინტეგრირებული საინფორმაციო მართვის სისტემის ზოგადი სქემა

საინფორმაციო მართვის სისტემის გაშლილი ზოგადი სქემა. საპრობლემო სფერო (სს), ჩვენს შემთხვევაში ელექტრონული ბიზნესის ან კომერციის კომპლექსური ობიექტია (მაგ., დიდი სავაჭრო ცენტრი).

სისტემის მომხმარებლები (სმ) კლასიფიცირდება მათი ფუნქციური დანიშნულების მიხედვით (მაგ., სისტემის ადმინისტრატორი, მონაცემთა საცავის ადმინისტრატორი, საბოლოო მომხმარებლები - სავაჭრო ცენტრის ხელმძღვანელები და ფუნქციურ ქვედანაყოფთა სპეციალისტები და ა.შ.). I_1 – ინტერფეისია საპრობლემო სფეროს და სისტემის მომხმარებლებს შორის.

საბაზო ინფორმაციულ ტექნოლოგიას (სიტ) წარმოადგენს განაწილებული საინფორმაციო მართვის სისტემის ძირითადი პროგრამული პაკეტები. ესაა ოპერაციული სისტემა (მაგ., მაიკროსოფტის Windows 2000/XP და დოტ-NET პლატფორმები) და მაღალი დონის პროგრამული ენები, რომელთაც აქვს ობიექტ-ორიენტირებული მეთოდები და ინსტრუმენტული საშუალებანი (მაგ., C#.NET, Visual Basic.NET, JavaScript.NET, Visual C++.NET, Borland C++ Builder და სხვ.) [1,2,5].

საინფორმაციო ტექნოლოგიები (სტ-i, $i=\overline{1,n}$) შეიძლება

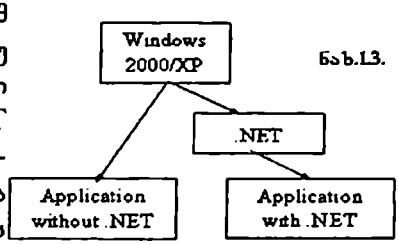

სს
I_1
სმ
I_2
სა
I_3
სტ-1
სტ-2
სიტ
I_4
I_5
I_6
სტ-i
სტ-n
I_k
მონაცემთა საცავი (REPOSITORY)
მსა
ჰიპერ-
ტექსტები
ტ1
ტ2
ტ3
ტk
რელაციური
ცხრილები
მბ-1
მბ-2
მბ-m
გრაფიკული
ფაილები
აუდიო-
ვიზუალური
ფაილები

ნახ.1.2.

განვიხილოთ ფართო სპექტრით. კონკრეტული ინტეგრირებული საინფორმაციო მართვის სისტემის დაპროექტებისას ობიექტ-ორიენტირებული ანალიზის ეტაპზე დაზუსტდება აუცილებელი პროგრამული პაკეტებისა და მათ შორის საკომუნიკაციო ინტერფეისების შედგენილობა. მაგ., დოტ-NET საბაზო პლატფორმის შემთხვევაში ინფორმაციული ტექნოლოგიები შეიძლება იყოს C#.NET, Visual Basic.NET, MsOffice, ASP.NET (Web-აპლიკაციისთვის), ADO.NET. მონაცემთა ბაზების მართვის სისტემების სახით გამოიყენება MsSQL Server-2000, Oracle, InterBase და ა.შ. ყოველ ინფორმაციულ ტექნოლოგიას აქვს საბაზო ტექნოლოგიასთან კომუნიკაციის საინტერფეისო ($I_i j = \overline{4, J}$)ენა [12,17,49].

მონაცემთა საცავებისა და ბაზებიდან ინფორმაციის ამოსაღებად ფართოდაა მიღებული სტანდარტული სტრუქტურირებადი მოთხოვნების ენის ინსტრუმენტის გამოყენება (SQL). სხვადასხვა ინფორმაციული ტექნოლოგიების (მაგ., დაპროგრამების სხვადასხვა ენები, სხვადასხვა ბაზები) გამოყენებისას მონაცემების ან პროგრამული კოდების თავსებადობისათვის შეიძლება CORBA (Common Object Request Broker Architecture) ან DCOM (Distributed Component Object Model) ტექნოლოგიების გამოყენება [19].

დოტ-NET პლატფორმის შემთხვევაში მაიკროსოფტმა ორიგინალურად გადაწყვიტა და გაამარტივა ეს პროცესი. მან შექმნა შუალედური გარდაქმნის ენა (IL- Intermediate Language). პროგრამები, რომელთა საწყისი კოდები დაწერილია, მაგალითად C#., VB. ან .NET-ის IL-ენაზე, კომპილირდება მანქანურ კოდში. .NET-ის ყველა ენა ფლობს CTS (Common Type System) მონაცემთა შესათანხმებელ საერთო ტიპებს, რათა ენების სტანდარტიზაცია იქნეს მიღწეული [19,22,23]. ამგვარად, ობიექტური კოდები IL-ენის საშუალებით ისე მიიღება, რომ მათში არაა დაფიქსირებული, თუ რომელ ენაზეა დაწერილი საწყისი კოდი. დოტ-NET პლატფორმა მდებარეობს Windows-ოპერაციულ სისტემასა და საავტომატიზაციო სისტემის აპლიკაციას შორის (იხ. ნახ.1.3).



ნახ.1.3.

1.2. ელექტრონული ბიზნესი

ელექტრონული ბიზნესი ისეთი საქმიანობაა, რომელიც იყენებს ინტერნეტ/ინტრანეტ საინფორმაციო ქსელებს კომპანიის შიგა და გარე კავშირებისათვის, ახორციელებს სამეურნეო, პარტნიორულ და საშუამავლო ურთიერთობებს მოგების მიღების მიზნით. იგი მარკეტინგული სისტემაა ინტერნეტული ტექნოლოგიებით.

ელექტრონული ბიზნესი ფირმებისათვის უზრუნველყოფს Web-საიტების შექმნას, კომპანიის ბიზნეს-პროცესების ინტეგრაციას, მაღალ კავშირს დამკვეთებთან და მომხმარებლებთან, ბაზრის მასშტაბების გაფართოებას.

ელექტრონული კომერცია ელექტრონული ბიზნესის ძთავარი შემადგენელი ნაწილია. იგი ტექნიკური და ორგანიზაციული ფორმების ერთობლიობაა, რომლის საშუალებითაც შესაძლებელია მატერიალური და ფინანსური აქტივების გადაცემა ბიზნესის ერთი სუბიექტიდან მეორეზე. ბიზნეს-ურთიერთობები იყოფა ოთხ კატეგორიად:

- Business-to-Business (B2B)- ელექტრონული კომერციაა საწარმოებს შორის, რომელიც მოიცავს ყველა დონის ინფორმაციულ კავშირებს კომპანიებს შორის. ასეთი სისტემები ძირითადად გამოიყენება წარმოების მომარაგებასა და მზა პროდუქციის გასაღებაში;
- Business-to-Consumer (B2C)- ელექტრონული კომერციის ვარიანტია, სადაც მყიდველები წარმოადგენენ კერძო პირებს. მაგალითად- ინტერნეტ-მაღაზია, ფასიანი საინფორმაციო სამსახური და ა.შ.;
- Consumer-to-Consumer (C2C)- ელექტრონული კომერციაა სხვადასხვა კერძო პირებს შორის, მაგალითად ელექტრონული აუქციონი;
- Business-to-Government (B2G)- ელექტრონული კომერციაა საწარმოსა და სახელმწიფო ორგანოებს შორის.

კომპანიის საბაზრო სტრატეგიის კუთხით ელექტრონული ბიზნესი და კომერცია გვთავაზობს ისეთი ფორმების არსებობას ინტერნეტში როგორიცაა ელექტრონული სავიზიტო ბარათი, ელექტრონული კატალოგი, ელექტრონული ვაჭრობა (e-trading), ელექტრონული მაღაზიები, (e-shop), ელექტრონული ბუხჰალტერია და ა.შ.

1.3. ელექტრონული კომერცია

ჩვენი ცხოვრების დღევანდელი დინამკური რიტმი და ტემპი მრავალ საწარმოოს და, უპირველეს ყოვლისა, თვით ადამიანს აიძულებს უფრო მოქნილი იყოს გარემოს ცვლილებათა მიმართ. ეკონომიკური საქმიანობის გლობალიზაციის და უნივერსალიზაციის პროცესი სულ უფრო და უფრო იწევა წინა პლანზე. საჭირო ხდება ენერგიის, დროის და რა თქმა უნდა, ფინანსური რესურსების დაზოგვის ღონისძიებების გატარება, რისი უნიკალური საშუალებაცაა ელექტრონული ბიზნესისა და კომერციის გამოყენება [6].

ელექტრონული კომერცია (E-Commerce), როგორც ცნობილია გამოიყენება ინტერნეტ-ტექნოლოგიის დახმარებით და მომხმარებელს მიეწოდება ინტერნეტ ქსელის საშუალებით [53].

„E" – electronic (ელექტრონული) ლათინურიდან, ნიშნავს სწრაფს.

„E" – economical, ინგლისურად ნიშნავს ეკონომიურს.

„E" – extended business, ნიშნავს, „ბიზნესი საზღვრების გარეშე".

ამ თვალსაზრისით ელექტრონული კომერცია არის სწრაფი და ბიზნესის ეკონომიური სახე, რომელიც არ ცნობს საზღვრებს. ინტერნეტის შესახებაც არსებობს სხვა-დასხვა მოსაზრებები. [26].

დღეს მსოფლიოში საკმაოდ პოპულარული თემაა ელექტრონული კომერცია – კომპიუტერის მონიტორიდან უშუალოდ მსოფლიო ბაზარზე ყიდვა–გაყიდვის წარმოება. მთელი რიგი კომპანიების საშუალებით წარმატებით იქნა ორგანიზებული თანამედროვე გასაღების ბაზარი. [21] სავაჭრო ორგანიზაციების ინტერნეტში განთავსების მიზეზი რამდენიმეა: ინტერნეტ მაღაზია ფუნქციონირებს მთელი დღე და ღამე (24სთ). მისაწვდომია მსოფლიოს ყველა ქვეყნის მოსახლეობისთვის და უზრუნველყოფს მომხმარებლის ინდივიდუალურ მომსახურებასაც. კორპორაციები IBM, Oracle, Microsoft დიდი ხანია დაინტერესდნენ და პერსპექტივით უყურებენ ამ მიმართულებას, აუმჯობესებენ თავიანთ პროდუქციას საწარმოებლად და ამ სფეროში დაკავებულ ბიზნესმენებს სთავაზობენ სხვადასხვა პროგრამულ უზრუნველყოფას (IBM NET, Commerce, Oracle ICS და სხვა)[54].

კომერციის ელექტრონული სისტემის რეალიზაციისათვის საჭიროა კომპლექსური პროგრამული სისტემა: რამდენიმე პროგრამული ენა და ტექნოლოგია (HTML, Java, JavaScript, JDBC, SQL, XML, მონაცემთა ბაზებისა და ვებ-ინსტრუმენტებით (ASP.NET).

ბრაუზერები (browser) – ვებ კვანძების დათვალიერების პროგრამები მომხმარებელს უზრუნველყოფს ინტერნეტში სამუშაოდ და სხვადასხვა ვებ-კვანძების და გვერდების დასათვალიერებლად. გრაფიკული ბრაუზერები Netscape და Internet-Explorer საშუალებას იძლევა მომხმარებელმა დაინახოს და „მოისმინოს“ ის ვებ-გვერდები, რომლებიც შეიცავს დინამიკურ ინფორმაციას, აუდიო და ვიდეო ფაილებს.

ვებ-სერვერების უმრავლესობას შეუძლია ერთდროულად დაამუშაოს რამდენიმე მოთხოვნა, ე.ი. ერთ ვებ კვანძს შეუძლია ერთდროულად რამდენიმე მომხმარებელი მიიღოს.

ვებ-სერვერს და მომხმარებელს ესაჭიროება ინფორმაციის გაცვლის ერთიანი პროტოკოლი (Hyper-Text Transfer Protocol HTTP). კლიენტ – სერვერის ურთიერთობის დროს ვებ გვერდის მოთხოვნა იგზავნება HTTP ფორმატში, ვებ სერვერი ინტერპრეტაციას უკეთებს ამ მოთხოვნას და უბრუნებს ბრაუზერს მონაცემებს.

ფირმა Visa International და კომპანია Master Card International პროტოკოლ SET-ში (Secure Electronic Transaction – დაცული ელექტრონული ტრანზაქცია) ხედავს ინტერნეტში ვაჭრობის ძლიერ ინსტრუმენტს, რომელიც აუმჯობესებს ქსელში ჩატარებული ოპერაციების უსაფრთხოებას და აზღვევს გარიგებაში მონაწილე ორივე მხარეს [55].

SET-მომავლის პროტოკოლია, დღეს კი ინტერნეტში ელექტრონული ტრანზაქციების ჩატარება ხდება SSL–ის (Secure Socket Layer) გამოყენებით. SSL არის შიფრაციის პროტოკოლი და პირველ რიგში გამოიყენება ქსელში გადაცემული ინფორმაციის დასაცავად. ელექტრონული კომერციის პრობლემები კი უფრო ფართო სპექტრს შეიცავს, ვიდრე შიფრაცია. აქ შეიძლება საქმე ეხებოდეს ისეთ პრობლემებს, როგორიცაა კლიენტის მიერ შეთანხმების პირობის ან მაღაზიის სინამდვილის შემოწმება. ამიტომ მსოფლიოს უდიდესი კომპანიები დიდ კაპიტალს აბანდებენ ამ მიმართულების გასანვითარებლად და რადიკალურ ფორმებს მიმართავენ მსოფლიო ბაზარზე საკუთარი ადგილის დასამკვიდრებლად, თუმცა მომავალი ციფრული ეკონომიკის ბირთვი (Digital economy) არის არა მხოლოდ სავაჭრო არამედ საქმიანი ტრანზაქციების ჩატარების საშუალება, რაც გულისხმობს ინდივიდუალურ და კორპორაციულ კონტრაქტებს, ინვესტიციების და კაპიტალის გადაადგილებას [8].

1.4. ელექტრონული კომერციის ინფრასტრუქტურული სისტემა

ელ-კომერციის მუშაობა დამოკიდებულია ქსელის გარეგნულ ინფრასტრუქტურაზე. ეს უკანასკნელი მოიცავს ინფორმაციის გადაცემის მატარებლებს [54]. ამიტომ მასში შედის ინტერნეტი, საკაბელო ტელევიზია, ტელეკომუნიკაციური და კერძო კორპორაციული ქსელები.

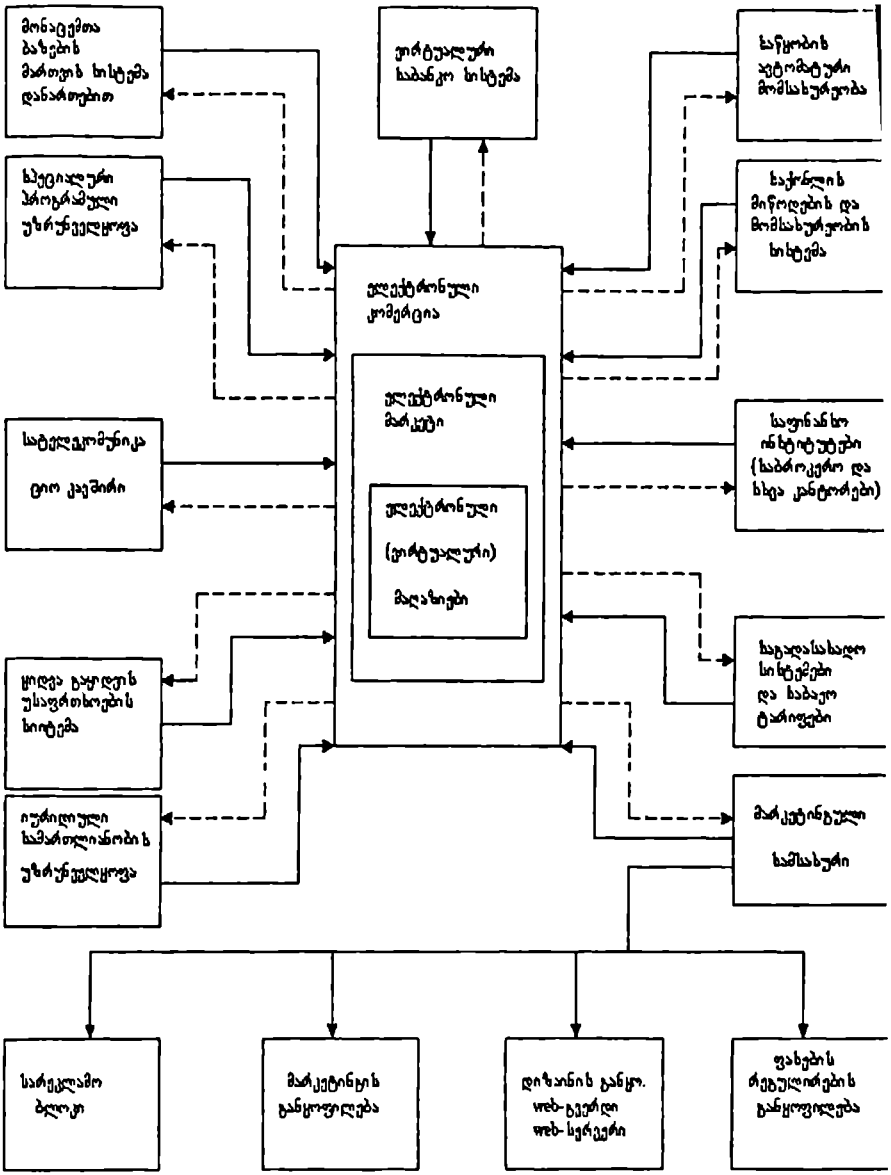
საწარმოო ინფრასტრუქტურა ფოკუსირდება პროდუქციაზე და იმაზე, რაც საჭიროა მის შესაქმნელად. გაყიდვის ინფრასტრუქტურის მიზანია შეათანხმოს საქონელი და მომსახურება მომხმარებელთან. მომსახურების ინფრასტრუქტურა კი მოიცავს გაყიდვის, გაყიდვის შემდგომი მხარდაჭერის და უსაფრთხოების პროცესებს [53].

ქსელის მუშაობა მთლიანად დამოკიდებულია პროტოკოლებზე – ქსელის მუშაობის წესებზე. პროტოკოლები განსაზღვრავს, თუ როგორ უკავშირდება ელ-კომერციის გამოყენებითი ნაწილი ქსელს, თუ როგორ იყოფა მონაცემები გამოყენებითი პაკეტებად საკაბელო გადაცემებისთვის. ამასთანვე როგორი ელექტრონული სიგნალები წარმოადგენს მონაცემებს ქსელურ კაბელში. მონაცემები დაჯგუფებულია ციფრული ქსელებისათვის გადასაცემად. პაკეტები ამ მონაცემების გარდა შეიცავს მათზე საკონტროლო ინფორმაციასაც. პროტოკოლის შემუშავებისას განისაზღვრება ის, თუ როგორ გაცვლის იგი მონაცემებს მეზობელ დონეებს შორის. მეცნიერულად დასაბუთებული და ეფექტურად ფუნქციონირებადი ელ-კომერციული სისტემა გვთავაზობს ძირითადი ელემენტების ჯგუფს, რომელიც უზრუნველყოფს კონკრეტული პირობების შესრულებას[55].

ელექრტონული კომერციის ძირითადი ელემენტები იყოფა ორ ჯგუფად: პირველში განთავსებული ელემენტები წარმოადგენს ელექრტონული კომერციის ინფრასტრუქტურული სისტემის ყველა კომპონენტს, ხოლო მეორე ჯგუფში განთავსებული ელემენტები კი მათი რეალიზაციის ორგანიზაციული ფორმის მრავალსახეობას.

პირველი ჯგუფის ძირითად ელემენტები მნიშვნელოვნად განსხვავდება იმ ელემენტებისგან, რომელთაც ადგილი ჰქონდა ტრადიციული მაღაზიების შემთხვევაში. ელ-კომერციის სისტემა იყენებს აბსოლიტურად განსხვავებულ ბიზნეს-საშუალებებს [57].

ელექრტონული კომერციის ინფრასტრუქტურული სისტემის


მონაცემთა ბაზების მართვის სისტემა დანართებით
ვირტუალური სამანქო სისტემა
საწყობის ავტომატური მომსახურეობა
სპეციალური პროგრამული უზრუნველყოფა
ელექტრონული კომერცია
საქონლის მიწოდების და მომსახურეობის სისტემა
ელექტრონული მაღაზია
საბანკო ინსტიტუტები (საბროკერო და სხვა კანტორები)
სატელეკომუნიკაციო ცენტრი
ელექტრონული (ვირტუალური) მაღაზიები
საგადასახადო სისტემები და სხვა ტარიფები
ცეცხლგადამდები უსაფრთხოების სისტემა
მართვადი სატრანსპორტო საშუალება
იურიდიული სამართლიანობის უზრუნველყოფა
საოფისე ავეჯი
საოფისე ტექნიკა
მართკუთხა ბლოკი
მართკუთხის განლაგება
დიზაინის განლ. web-გვერდი web-საიტები
ფანჯრის რეგულირების განლაგება

ნახ.1.4.

ძირითად ელემენტებს აქვს 1.4 ნახაზზე წარმოდგენილი სახე:

- სპეციალური პროგრამული უზრუნველყოფა;
- მონაცემთა ბაზების მართვის სისტემა;
- სატელეკომუნიკაციო კავშირები;
- უსაფრთხოების სისტემა საქონლის ყიდვა-გაყიდვის და მომსახურების სფეროში;
- იურიდიული სამართლიანობის უზრუნველყოფა;
- ვირტუალური საბაზო სისტემა;
- სპეციალური საგადამხდელო სისტემა;
- საწყობის ავტომატური მომსახურება;
- საქონლის მიწოდებისა და მომსახურეობის სისტემა;
- საფინანსო ინსტიტუტები (საბროკერო და სხვ.);
- საგადასახადო სისტემა და საბაჟო ტარიფები;
- მარკეტინგული მომსახურეობა, რომელიც მოიცავს: სარეკლამო თუ ფასების რეგულირების და დიზაინის განყოფილებებს: (web-გვერდები, web-სერვერი).

შემოვიტანოთ რამდენიმე კომპონენტის განმარტება.

ელექტრონული მაღაზია. იგი განთავსებულია ინტერნეტ-ქსელში, web-სერვერის სახით. ასეთი სახის მაღაზიის შექმნის მთავარი მიზანია უზრუნველყოს, დროის მინიმალურ შუალედში საქონლის გასაღება და დახმარება გაუწიოს ინტერნეტის სხვა მომხმარებლებს, რათა იაფად და სწრაფად შეარჩიონ მათთვის სასურველი პროდუქცია, ასევე ყველა მომხმარებელს საშუალება აქვს აწარმოოს შეკვეთები.

სპეციალური პროგრამული უზრუნველყოფა. პროგრამული უზრუნველყოფისთვის გამოიყენება დაპროგრამების შემდეგი ენები – Java, HTML, XML და ა.შ. მონაცემთა შეტანა-გამოტანის შაბლონები, დიზაინი და web-გვერდების მომზადების საშუალება, სპეციალური პროგრამული უზრუნველყოფა და ა.შ.

იურიდიული უზრუნველყოფა.

ელექტრონული კომერციის ორგანიზაცია, პირველ რიგში ბაზირებული უნდა იყოს ტრადიციულ იურიდიულ ნორმებით და წესებით, მეორე რიგში კი უნდა მოხდეს ახალი სპეციალიზირებული სამართლის ინსტიტუტების მიერ მიღებული ცვლილებების და პროცედურების გათვალისწინება. ამას გარდა აქტუალურია კანონმდებლობის უნიფიკაცია, აგრეთვე პროცედურების და წესების

გამარტივება, რომლებიც გამოიყენება სხვადასხვა ქვეყნებში. იგი ხელს უწყობს თანამშრომლობას ბიზნესის წარმოებასა და შესაბამის სახელმწიფო მართვის სტრუქტურებს შორის[23].

სპეციალური საგადასახადო სისტემა. ინტერნეტის საშუალებით გადახდის წარმოება ხორციელდება სხვადასხვა ბარათების დახმარებით.[54]

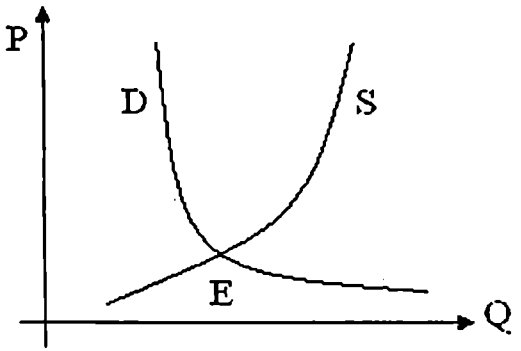
ინტერნეტ ქსელის შესაძლებლობების გამოყენება მომხმარებლის უშუალოდ აქტიური ჩარევით. ეს შესაძლოა იყოს, როგორც მინიმუმი web – გვერდი ან საკუთარი web- სერვერი, ხოლო როგორც მაქსიმუმი – უსაზღვრო შესაძლებლობები, რაც გამოიხატება ინტერაქტიული მაღაზიების არსებობაში და მსხვილი ფირმების ხელმძღვანელებთან უშუალო კავშირებში.

როდესაც ვსაუბრობთ ეკონომიკაზე და მასში მიმდინარე პროცესებზე, აუცილებელია გავითვალისწინოთ რამდენიმე ძირითადი პრინციპი:

ეკონომიკურ მოვლენას განსაზღვრავს მრავალი ცვალებადი ფაქტორი, ამიტომ ამ მოვლენის შესწავლის გასაადვილებლად აფიქსირებენ (უცვლელად თვლიან) ყველა ფაქტორს გარდა ერთისა და სწავლობენ თუ როგორ რეაგირებს მოვლენა ამ ერთი ფაქტორის ცვლილებაზე სხვა ფაქტორების უცვლელობის პირობებში.

მაგალითად, ყველასათვის ცნობილია, რომ მიწოდება, მოთხოვნა და ფასი ურთიერთდაკავშირებულია. იზრდება ფასი კლებულობს მოთხოვნა და პირიქით კლებულობს ფასი მოთხოვნა იზრდება, ეს ურთიერთკავშირი ეკონომიკაში აქსიომად ითვლება (არის გამონაკლისი შემთხვევები, როდესაც ეს პრინციპი ირღვევა).

მოთხოვნის და მიწოდების განმსაზღვრელი ცენტრალური ფაქტორი ფასია. გამყიდველებისთვის უკეთესია მღალი ფასი, ხოლო მყიდველისათვის – დაბალი, თუ ყველა სხვა პირობა ერთნაირია. თუ დავადებთ ერთმანეთს გრაფიკებს, მოთხოვნის და მიწოდების მრუდები გადაიკვეთება. 1.5 ნახაზზე ამას შემდეგნაირი სახე აქვს:



ნახ.15. მოთხოვნა, მიწოდება, წონასწორობა

D – მოთხოვნის მრუდი; S – მიწოდების მრუდი; P- ფასების ღერძი; Q - მიწოდებული საქონლის რაოდენობის ღერძი; E- წონასწორობის წერტილი.

მოცემული ნახაზიდან[54] ჩანს, რომ არსებობს ერთადერთი ფასი, რომლის დროსაც მომხმარებელთა და მიმწოდებელთა ინტერესები ერთმანეთს ემთხვევა. ამ ფასს წონასწორულ საბაზრო ფასს (Market Equilibrium Price) უწოდებენ. იგი ზუსტად იმ წერტილშია სადაც გადაიკვეთება მოთხოვნისა და მიწოდების მრუდი. ე.ი. წონასწორობის წერტილი – ეს ისეთი წერტილია, რომელშიც მოთხოვნისა და მიწოდების სიდიდეები ერთმანეთის ტოლია, ხოლო საბაზრო ფასი - ეს ისეთი ფასია, რომელიც განისაზღვრება მოთხოვნისა და მიწოდების საშუალებით.

1.4. მონაცემთა საცავის ანალიზის OLAP- ინსტრუმენტი

მონაცემთა მრავალგანზომილებიან კომპლექსური ანალიზის ტექნოლოგიას უწოდებენ **OLAP** (Online Analytical Processing) ტექნოლოგიას, რომელიც ასე განიმარტება: "მონაცემთა ოპერატიული ანალიზი", მონაცემთა საცავში იგი წარმოადგენს გასაღებურ კომპონენტს.

OLAP – ინსტრუმენტი პირველად მონაცემებს ასახავს ინფორმაციის სახით, რომლის დახმარებითაც შესაძლებელი ხდება

ცხრ.1.1

[illegible]	[illegible]	[illegible]	[illegible]-[illegible] [illegible]
ბერმანია	საყოფაცხო. ელექ.	2001	234
ვენესუელა	საყოფაცხო. ელექ.	2001	456
არგენტინა	საყოფაცხო. ელექ.	2002	321
ბერმანია	საყოფაცხო. ელექ.	2003	342
ვენესუელა	ტეზინის ნაკეთ.	2003	123
არგენტინა	საყოფაცხო. ელექ.	2001	657
ვენესუელა	ტეზინის ნაკეთ.	2002	143
ბერმანია	ტეზინის ნაკეთ.	2002	345
ვენესუელა	ტეზინის ნაკეთ.	2001	56
არგენტინა	საყოფაცხო. ლექ.	2003	234
ბერმანია	ტეზინის ნაკეთ.	2003	309
არგენტინა	ტეზინის ნაკეთ.	2002	345
ვენესუელა	საყოფაცხო. ელექ.	2003	117
არგენტინა	ტეზინის ნაკეთ.	2001	57
ბერმანია	ტეზინ.ნაკეთ.	2002	152
ვენესუელა	საყოფაცხო. ელექ	2002	672
არგენტინა	ტეზინის ნაკეთ.	2003	212

საწარმოო მოცულობის შესახებ ვიქონიოთ რეალური წარმოდგენა. ამავე დროს იგი უნიკალური ინსტრუმენტია, რომელიც საშუალებას გვაძლევს სხვადასხვა ჭრილით ჩავატაროთ ინფორმაციის მრავალგანზომილებიანი ანალიზი.

<u>OLAP = მრავალგანზომილებიანი წარმოდგენა = კუბი.</u>

OLAP წარმოადგენს სწრაფმოქმედ და ხელსაყრელ საშუალებას საქმიანი ინფორმაციის დათვალიერებისა და ანალიზისათვის [46,60].

იგი არ არის ცალკე აღებული პროგრამული პროდუქტი, პროგრამირების ენა ან რომელიმე კონკრეტული ტექნოლოგია.

OLAP–ს გააჩნია საკუთარი კონცეფცია, პრინციპები და მოთხოვნები, რომელიც ეყრდნობა პროგრამულ პროდუქტს და ანალიტიკოსებს უადვილებს მონაცემებთან ურთიერთობას.

ანალიტიკოსი არის განსაკუთრებული მომხმარებელი კორპორაციული ინფორმაციით. ანალიტიკოსის ამოცანაა დაამყაროს წესრიგი დიდ მონაცემთა მასივში, ამიტომ მისთვის მნიშვნელოვანი არ არის ცალკეული ფაქტები. მონაცემთა საცავში არსებული ცალკეული ფაქტებით შეიძლება დაინტერესდეს მაგალითად, ბუხჰალტერი ან რომელიმე განყოფილების ხელმძღვანელი, რომელთა კომპეტენციაშიც შედის საქმიანი შეთანხმებები.

ანალიტიკოსისათვის კი ერთი რომელიმე სახის მონაცემი თითქმის არაფერს წარმოადგენს, მისი ინტერესის სფერო შეიძლება გახდეს დროის რაღაც შუალედში (მაგ., ერთი თვის, ერთი წლის) რომელიმე მსხვილი ობიექტის, ყველა სახის სქმიანი შეთანხმება.

ამავე დროს ანალიტიკოსს შეუძლია არ გაამახვილოს ყურადღება ისეთ ინფორმაციაზე, როგორიცაა მაგ., მომხმარებლის მისამართი, ტელეფონის ნომერი, კონტრაქტის ინდექსი და ა.შ.

მისთვის საჭირო ინფორმაცია წარმოდგენილი უნდა იყოს რიცხვითი მნიშვნელობებით, ეს განაპირობებს მისი მოქმედების ძირითად არსს. ეს უკანასკნელი მიუთითებს იმაზე, რომ ანალიტიკოსი მუშაობს ასეთი სახის ცხრილთან:(ცხ.1.1) ქვეყანა, საქონელი და წელი წარმოადგენს ატრიბუტებს, ხოლო ყიდვა გაყიდვის მოცულობა გამოსახულია რიცხვითი მნიშვნელობით. ანალიტიკოსის ამოცანა მდგომარეობს ატრიბუტებსა და რიცხვით პარამეტრებს შორის დამოკიდებულების გარკვევაში. თუ ცხრილს დავაკვირდებითშევამჩნევთ, რომ ცხრილის წარმოდგენა შესაძლებელია სამ განზომილებაში: ერთი ღერძის გასწვრივ დალაგდება ქვეყნები, მეორეზე, საქონელი, ხოლო მესამეზე წელი. თავისი მნიშვნელობით იგი გამოსახავს სამგანზომილებიან მასივს შესაბამისი ყიდვა-გაყიდვის მოცულობით.

1.6 ნახაზზე განსხვავებული ფერი გამოხატავს სეგმენტს სადაც იმ წელს არ არსებობს მონაცემი მაგ. გერმანიას 2001 წელს რეზინის ნაკეთობებზე არა აქვს ინფომაცია. სწორედ ასეთი სახის სამ განზომილებიან მასივს უწოდებენ OLAP – კუბს.

აუცილებლობას არ წარმოადგენს კუბის ყველა ელემენტის შევსება, მაგ., თუ არ იქნება ინფორმაცია, როგორც აღნიშნეთ გერმანიაში 2001 წელს რეზინის ნაკეთობებზე, მაშინ შესაბამისი უჯრედი უბრალოდ არ იქნება განსაზღვრული OLAP – დანართში და ამავე დროს საჭიროებას არ წარმოადგენს დანართში მოხვდეს მრავალგანზომილებიანი სტრუქტურის ყველა ელემენტი, მთავარია მომხმარებელმა მიიღოს მხოლოდ საჭირო მონაცემები. კუბში მრავალგანზომილებიან მონაცემთა კომპაქტური შენახვა განაპირობებს, რომ არ მოხდეს მეხსიერების უსარგებლო დაკარგვა არასასურველ ინფორმაციაზე.

თუ ნამდვილ კუბს, შევადარებთ OLAP – კუბს, მაშინ მათ შორის შევნიშნავთ განსხვავებას – ნამდვილი კუბის ელემენტების რაოდენობა ყველა განზომილებაში ერთნაირია, ხოლო OLAP – კუბი არ არის ზუსტად განსაზღვრული. იგი შეიძლება იყოს როგორც ორგანზომილებიანი, ასევე სამ და მრავალგანზომილებიანი, დამოკიდებულია თუ როგორი სახის ამოცანასთან გვაქვს საქმე.

OLAP – კუბის თითოეული განზომილება შედგება ეგრეთ წოდებული შრეების ან ცალკეული ნაწილებისგან მაგ., განზომილება „ქვეყანა“ შედგება შრეებისგან: გერმანია, ვენესუელა, არგენტინა და ა.შ. თვითონ მრავალგანზომილებიანი კუბის გამოყენება ანალიზისათვის შეუძლებელია. მაგ., თუ მოცემული გვაქს ექვს ან ცხრამეტგანზომილებიანი კუბი მათი ადეკვატური გარდაქმნა სირთულეებთან არის დაკავშირებული, ამიტომ გამოყენების წინ მრავალგანზომილებიანი კუბიდან გამოყოფენ, ორგანზომილებიან ცხრილს ამ ოპერაციას უწოდებენ კუბის „გაკვეთას“. ანალიტიკოსები მრავალგანზომილებიან კუბიდან „ამოკვეთენ“ მათთვის საინტერესო შრეებს. შესაბამისად „გაუკვეთავი“ რჩება ორი განზომილება. როდესაც კუბი შეიცავს რამდენიმე სახის რიცხვით მნიშვნელობას, ერთი სახის მნიშვნელობები განთავსდება ცხრილის ერთ განზომილებაში.

თუ ზემოთ მოცემულ ცხრილს კარგად დავაკვირდებით, შევამჩნევთ რომ იქ მოცემული მონაცემები პირველადი კი არ არის, არამედ შედარებით მცირე ელემენტები დაჯგუფებულია და წარმოდგენილია სრულყოფილი სახით. მაგ., წელიწადი შეგვიძლია დავყოთ კვარტალებად, კვარტალები თვეებად, თვეები კვირეებად, კვირები დღეებად და ა.შ. ქვეყანა შედგება რეგიონებისგან რეგიონები – დასახლებული პუნქტებისგან.

OLAP – ში ასეთი სახის მრავალდონიან გაერთიანებას უწოდებენ იერარქიას. ცალკეული ელემენტი შეიცავს რამოდენიმე სახის იერარქიას: მაგ. დღე-კვირა-თვე ან დღე-დეკადა-კვარტალი. მონაცემთა დაჯგუფება იწყება ყველაზე დაბალი დონიდან და ბოლოს ჯგუფდება მაღალი დონის მონაცემამდე. იმისათვის, რომ დაჩქარდეს ერთი მდგომარეობიდან მეორეში გადასვლის პროცესი და მონაცემთა სხვადასხვა დონეზე დაჯგუფება, ყველა მონაცემი უნდა ინახებოდეს კუბში. ერთი შეხედვით მომხმარებელი ხედავს მხოლოდ ერთ კუბს, მაგრამ შეგვიძლია ვთქვათ, რომ ეს კუბი შეიცავს პრიმიტიული კუბების მთელ სიმრავლეს.

იერარქია ანუ მონაცემთა აგრეგატული სახით წარმოდგენა

ერთ-ერთი მნიშვნელოვანი გარემოება, რამაც OLAP – ტექნოლოგია წარმოაჩინა, როგორც ეფექტური და მაღალმწარმოებლური ინსტრუმენტი, ეს არის იერარქიული გაერთიანებების დამსახურება. ანალიტიკოსი, დამოუკიდებლად ან პროგრამისტის დახმარებით დასვავს შესაბამის SQL – მოთხოვნას და შედეგად ღებულობს მონაცემებს ელექტრონული ცხრილის სახით. ამ დროს წარმოიქმნება დაბრკოლებათა მთელი სიმრავლე, საიდანაც შეიძლება გამოვყოთ რამდენიმე მათგანი:

1. ანალიტიკოსი გარკვეული დროის განმავლობაში ელოდება პროგრამისტისგან ინფორმაციის მიღებას;

2. ინფორმაციის მიწოდება ხდება მხოლოდ ერთი ცხრილის საშუალებით, რის გამოც შეუძლებელია ჩატარდეს სრულყოფილი კვლევა. ამიტომ ერთსა და იმავე პროდუქტზე უნდა ჩატარდეს რამდენიმე გამოკვლევა;

3. როგორც ზემოთ აღვნიშნეთ, ანალიტიკოსს არ სჭირდება ყველა უმნიშვნელო ინფორმაცია, მას სჭირდება მხოლოდ მნიშვნელოვანი ინფორმაცია ერთდროულად. ეს იმას ნიშნავს, რომ კორპორაციულ რელაციური მონაცემთა ბაზების მართვის სისტემის სერვერმა, რომელსაც ანალიტიკოსი მიმართავს, უნდა მიაწოდოს ყველაზე აქტუალური ინფორმაცია და, ამავე დროს, დაბლოკოს სხვა დანარჩენი ტრანზაქციები.

არსებული პრობლემის გადასაჭრელად OLAP – ტექნოლოგია არის ერთ-ერთი საშუალება[58]. OLAP – კუბი წარმოადგენს მეტა-ანგარიშთა ნაკრებს. გაკვეთილი მეტა-ანგარიში (ანუ კუბი)

ანალიტიკოსს საშუალებას აძლევს სხვადასხვა განზომილებაში წარმოადგინოს მისთვის საინტერესო ორგანიზაციის სტრუქტურული მონაცემები ან ანგარიში.

სისტემის სრულყოფილი ანალიზისთვის OLAP – ინსტრუმენტის გამოყენებისას მნიშვნელოვანია 5 ძირითად პრინციპი, მას უწოდებენ FASMI – ტესტს. აქ კონკრეტულად განსაზღვრულია OLAP – პროდუქტის მოთხოვნები.

FASMI – აბრევიატურა განისაზღვრება ტესტის შემადგენელი თითოეული პუნქტისგან:

- Fast (სწრაფი) – OLAP დანართმა უნდა უზრუნველყოს ანალიზური მონაცემების მიღება მინიმალურ დროში. ანალიზი უნდა ჩატარდეს რაც შეიძლება სწრაფად და უნდა მოხდეს ყველა ინფორმაციული ასპექტის გათვალისწინება. ამ პერიოდის ხანგრძლივობა საშუალოდ უნდა იყოს ხუთი წამი და ურფო ნაკლები;
- Analysis (ანალიზი) – OLAP დანართმა მომხმარებელს უნდა მისცეს საშუალება განახორციელოს რიცხვითი და სტატისტიკური ანალიზი;
- Shared (ერთდროულად გამოყენება) – OLAP დანართმა უნდა უზრუნველყოს ერთსა და იმავე დროს რამდენიმე მომხმარებლის ერთდროული მუშაობა. ამავე დროს უნდა შემოწმდეს, რამდენად დაცულია კონფიდენციალური ინფორმაცია;
- Multidimensional (მრავალგანზომილება) – ეს არის OLAP –ის დამახასიათებელი არსებითი თვისება;
- Information (ინფორმაცია) – OLAP დანართმა მომხმარებელს უნდა მისცეს საშუალება, მიიღოს საჭირო ინფორმაცია, დამოუკიდებლად მისი მოცულობისა და შენახვის ადგილისა.

მონაცემთა მრავალგანზომილებიანი შენახვის ტექნიკური ასპექტები.

მრავალგანზომილებიან მონაცემთა საცავში თავმოყრილი აგრეგატული მონაცემები სხვადასხვა დონეზე ერთმანეთისგან განსხვავდება. მაგ., ყიდვა-გაყიდვის მოცულობა ერთ დღეში, თვეში, წელიწადში, საქონლის კატეგორია და ა.შ.

აგრეგატულ მონცემთა შენახვის მიზანია მოთხოვნათა დაკმაყოფილებაზე დახარჯული დროის შემცირება, რამდენადაც უმეტეს შემთხვევაში ანალიზის და პროგნოზის სფეროს წარმოადგენს არა

დეტალური ინფორმაცია, არამედ გაერთიანებული, საბოოლო სახემდე მიყვანილი დაჯგუფებული მონაცემები. ამიტომ მრავალგანზომილებიან მონაცემთა ბაზის შექმნის დროს, ყოველთვის გასათვალისწინებელია, რომ შეტანილი იქნეს რამდენიმე აგრაგატული მონაცემი. გასათვალისწინებელია ის ფაქტიც, რომ ყველა მონაცემის აგრაგატული სახით შენახვა გაუმართლებელია, რადგან მონაცემთა მოცულობაში ახალი განზომილების დამატების შემთხვევაში კუბი სადაც ამ მონაცემის ჩამატება მოხდება, დაიწყებს ზრდას ექსპონენციალურად ამ პროცესს უწოდებენ მონაცემთა მოცულობის „ფეთქებად ზომას“, ამიტომ აგრეგატულ მონაცემთა მოცულობის ზომის ხარისხი დამოკიდებულია კუბების რაოდენობაზე და იერარქიის სხვადასხვა დონეზე მიმდინარე გაზომვებზე. 'ფეთქებად ზომის“ პრობლემის გადასაჭრელად გამოიყენება სხვადასხვა სახის მონაცემთა შენახვის სქემები : MOLAP, ROLAP, JOLAP და HOLAP.

MOLAP – (მრავალგანზომილებიანი **OLAP**) პროდუქტს საფუძვლად უდევს მონაცემთა არარელაციური სტრუქტურა, რომელიც უზრუნველყოფს მონაცემთა მრავალგანზომილებიან შენახვას.

ROLAP – (რელაციური **OLAP**) ასეთ ინსტრუმენტში მრავალგანზომილებიანი სტრუქტურა რეალიზდება რელაციური ცხრილებით, ხოლო მონაცემები ანალიზის პროცესში შესაბამისად შეირჩევა მონაცემთა რელაციური ბაზებიდან ანალიზური ინსტრუმენტით.

JOLAP – ინსტრუმენტის გამოყენებისას თითოეული მოთხოვნის მოდელირება შესაძლებელია, როგორც ერთიან ობიექტთა ჯგუფი, რის განხორციელებაშიც ეხმარება გრაფიკულ ინტერფეისში ოპერაციათა თანამიმდევრული სტრუქტურა და მოთხოვნათა მოდიფიკაციის განსაზღვრა. **JOLAP** (Java Specification Requet-69) გმოიყენება სხვადასხვა სპეციფიკაციაში მაგ., როგორიცაა: ჯგუფი OMG მეტამოდელისთვის Common Warehouse Metamodel (CWM), ობიექტური მოდელი Meta Object Facility (MOF), XML Metadata Interchange (XMI), და აგრეთვე Java Metadata Interface (JMI,JSR-40). ამ ინტერფეისის არქიტექტურა ერთდროულად J2EE და J2SE ;

მრავალგანზომილებიანი **OLAP** უზრუნველყოფს მაღალ მწარმოებლურობას, მაგრამ სტრუქტურები არ შეიძლება გამოვიყენოთ დიდი მოცულობის მონაცემთა დამუშავებისთვის, რადგანაც დიდი გაბარიტები მოითხოვს დიდ აპარატულ რესურსებს. ამასთან ერთად

ჰიპერკუბის დატვირთვა შეიძლება იყოს ძალიან მაღალი, რაც შემდგომში თავს იჩენს ინფორმაციის მიწოდების სისწრაფეზე.

შეიძლება ვთქვათ პირიქითაც, რელაციური **OLAP** უზრუნველყოფს შენახული მონაცემების დიდი მასივების დამუშავებას, ასე რომ შეიძლება უზრუნველვყოთ უფრო ეკონომიური შენახვა, მაგრამ ამასთან ერთად, მნიშვნელოვან როლს ითამაშებს მრავალგანზომილებიან სამუშაოს შესრულების სიჩქარეზე. მსგავსს მსჯელობას მივყავართ ახალი კლასის გამოყოფაზე, რომელიც არის **HOLAP** ანალიზური ინსტრუმენტი.

ეს არის ჰიბრიდული მონაცემთა ოპერატიულ-ანალიზური დამუშავების ინსტრუმენტი. ასეთი კლასის ინსტრუმენტი საშუალებას იძლევა შეთანხმდეს ორივე მიდგომა – რელაციური და მრავალგანზომილებიანი. ხელმისაწვდომი გახდება როგორც მრავალგანზომოლებიანი მონაცემთა ბაზები ასევე რელაციური მონაცემები.

1.4. მონაცემთა საცავის შექმნის და მოდელირების ინსტრუმენტი

მონაცემთა საცავების შექმნისა და მოდელირების ინსტრუმენტი შეიძლება გამოყენებულ იქნას სხვადასხვა წყაროიებიდან მიღებულ მონაცემთა მოგროვებისთვისა ამ ინფორმაციის გარდაქნისა და აგრეთვე მის გლობალურ ან ლოკალურ დონეზე საცავში ჩასატვირთად.

საცავის სამრეწველო რეჟიმში მართვისათვის გამოიყენება IBM DB2 Warehouse Manager, ინსტრუმენტი, რომელიც წარმოადგენს საცავის ძირითადდ მექანიზმს[12].

მონაცემთა საცავის დაპროექტების ეტაპზე Data Warehouse Center – წარმოადგენს სათადარიგო ინსტრუმენტულ საშუალებას მონაცემთა საცავის შექმნისათვის. მის შემადგენლობაში შედის Warehouse Schema Modeler და Process Modeler. ისინი გამოიყენება დაპროექტებისთვის, სხვადასხვა სქემათა გენერაციისათვის და მონაცემთა ჩატვირთვისათვის, აგრეთვე შესაძლებელია ჩატარებულ მოქმედებათა გრაფიკული აღწერა, რომელიც სრულდება საცავის აგებისას. Process Modeler ახორციელებს მონაცემთა უპირობო

დამუშავებას, არსებულ მოვლენებზე შეტყობინებას და მართვის კალენდარულ დაგეგმვას.

IBM DB2 UDB Data Warehouse Editions – ახალი, მრავალფუნქციური პაკეტია მონაცემთა საცავების ორგანიზებისათვის, რომლის ბაზასაც IBM DB2 Universal Database Version 8.1 Enterprise Server Edition (ESE)–წარმოადგენს, ამავე დროს იგი არის დამატებითი საშუალება საცავის ადმინისტრირებისათვის, მონაცემთა ინტეგრაციისათვის, მართვისა და ანალიზისთვის.

DB2 Data Warehouse Enterprise Edition –პაკეტი ორიენტირებულია ნებისმიერი ზომის მონაცემთა საცავის შექმნაზე ნებისმიერ დარგში, რაც საშუალებას გვაძლევს ავაგოთ მძლავრი საცავი ფართო ანალიზური დანართებისთვის. პაკეტის მოცულობა შეიძლება იყოს 10 ტერაბაიტზე მეტი.

DB2 Data Warehouse Enterprise Edition-შეიძლება გამოყენებული იქნას შემდეგ პლატფორმებზე: IBM AIX, Microsoft Windows 2000/ 2003, Linux Sun Solaris.

IBM DB2 Cube Views -წარმოადგენს გადაწყვეტილების მიღების თანამედროვე საშუალებას, რომელიც OLAP – ოპერაციებს ახორციელებს მონაცებთა ბაზების მართვის IBM DB2 სისტემებში. მისი დახმარებით მომხმარებელს შეუძლია შექმნას მონაცემთა სწრაფად გარდამქმნელი და იოლად მართვადი OLAP – გადაწყვეტილება, რომელიც ხელს უწყობს ამაღლდეს ანალიზური დანართების მთელი სპექტრის წარმადობა.

IBM DB2 Cube Views–გამჭვირვალედ წარმოადგენს მრავალგანზომილებიან მონაცემთა ბაზების სტრუქტურას შემდეგი ფაქტორების დახმარებით:

ქმნის მეტამონაცემებს სხვადასხვა განზომილების, იერარქიის, ატრიბუტების და ანალიზური ფუნქციისათვის;

ხელს უწყობს და ამაღლებს OLAP – ინსტრუმენტის ეფექტურობას;

- DB2–კატალოგში შეტანილი OLAP მეტამონაცემები ამაღლებს ამ ინსტრუმენტების მაღალეფექტურობას;
- DB2–მაღალეფექტური ტექნოლოგიის გამოყენება იმდენად მარტივია, რომ იგი შეიძლება შევადაროთ ჯამურ ცხრილებსა და ანალიზურ ფუნქციებს.

IBM DB2 Query Patroller – მოთხოვნათა მართვის მაღალმწარმოებლური სისტემა, რომელიც ახორციელებს დინამიკურ კონტროლს მოთხოვნათა ნაკადზე და DB2–მონაცემთა ბაზაზე შემდეგი საშუალებებით:

- მოთხოვნათა შორის რაციონალურად უნდა განაწილდეს სისტემის რესურსები, მიუხედავად იმისა როგორი სირთულისაა მოთხოვნა;
- მოთხოვნათა ავტომატური გადაყვანა ლოდინის რეჟიმში;
- კონტრილიდან გამოსულ მოთხოვნათა სისტემიდან გამოყვანა.

Query Patroller - უზრუნველყოფს მონაცემთა საცავზე დატვირთვის რეგულირებას, მცირე და დიდ მოთხოვნათა სწრაფად დასაკმაყოფილებლად სისტემის რესურსების ეფექტურ გამოყენებას. აგრეთვე იგი ხელს უწყობს მონაცემთა მოგროვებას და ინფორმაციის ანალიზს მოთხოვნაზე პასუხის გასაცემად.

საცავების მმართველი IBM DB2 Warehouse Manager – სისტემა

IBM DB2 Warehouse Manager – უზრუნველყოფს მონაცემთა საცავების ავტომატიზებულ ორგანიზებას [59]. მისი მიზანია ბიზნესთან დაკავშირებული ინფორმაციის ღრმად ჩაწვდომა და მართვა, რათა სწორად მოახდინოს ბიზნესის წარმოება. ამ ინფრასტრუქტურაში შეიძლება გაერთიანდეს სხვა ინსტრუმენტები და Business Intelligence (ინტელექტუალური ბიზნესი) დანართები, რომლებიც გვაძლევს იმის გარანტიას, რომ მოვიღებთ იმ დროისთვის არსებული ყველაზე აქტუალური ინფორმაცია, რაც დაგვეხმარება ბიზნესის სხვადასხვა სფეროში გააზრებული გადაწყვეტილების მიღებაში. IBM DB2 – არის მონაცემთა ბაზების მართვის საუკეთესო სისტემა, რომელიც ხელს უწყობს საინფორმაციო საცავის შექმნას.

DB2- საცავის მენეჯერი: გვთავაზობს გაფართოებულ ფუნქციონალურ გაერთიანებას, სადაც საბაზისო DB2-ს შესაძლებლობას ემატება (ETL – Extraction, Transformation and Loading) მონაცემთა გარდაქმნის და ჩატვირთვის მექანიზმები;

- მეტამონაცემთა და საინფორმაციო კატალოგის მართვა. საინფორმაციო კატალოგი იძლევა მეტამონაცემთა დასმული მოთხოვნის გაერთიანების საშუალებას;

- განაწილებულ სისტემებში მონაცემთა ჩატვირთვას და გარდაქმნას- ETL;

მასში გაერთიანებულია QMF for Windows – დანართი,რომელიც ხელს უწყობს DB2 პაკეტისთვის, Windows ან Web ინტერფეისის დახმარებით მოთხოვნების დაყენებას.

აღსანიშნავია აგრეთვე DB2 Warehouse Manager V8-ის შემდეგი მხარე:

- გამოყენებისას იგი საკმაოდ მარტივია, გამოირჩევა მაღალი მწარმოებლურობით;
- ერთდროულად ახორციელებს რამდენიმე ოპერაციას;
- ახდენს საწყის მონაცემთა თავმოყრას და წარმოადგენს გაერთიანებული საცავის სახით;
- ახედნს მონაცემთა სწრაფად შევსებას;
- საინფორმაციო კატალოგისთვის ახალი ინტერფეისის სრულყოფა;
- უზრუნველყოფს IBM AIX და Linux – ოპერაციული სისტემისთვის მონაცემთა საცავის სერვერის დაკავშირებას.

II თავი. მონაცემთა საცავის დაპროექტება

2.1. მონაცემთა საცავის სტრუქტურა და ძირითადი ელემენტები

მონაცემთა საცავის (Data Warehouse - DWH) იდეა, რომელიც წარმოადგენს მართვის საინფორმაციო სიტემების დაპროექტების და რეალიზაციის ერთ-ერთ უახლეს ტექნოლოგიას, აქტუალური გახდა 90-იანი წლებიდან. ამ იდეის ფუძემდებლად მოიხსენიება ამერიკელი მეცნიერი ვ. ჰ. ინმონი [36].

მონაცემთა საცავი აღიწერებოდა, როგორც „მონაცემთა სუპერმარკეტი", „სუპერ მონაცემთა ბაზა". ამ მიმართულებით პირველი პროექტი „ევროპის ბიზნეს ინფორმაციული სისტემა" 1988 წელს IBM ფირმის მიერ განხორციელდა. გლობალურ მონაცემთა შენახვა და მათი ანალიზი სავაჭრო კომპანიების მუშაობისას არა მარტო ამაღლებს მუშაობის ეფექტურობას, არამედ უძლებს მკაცრ კონკურენციას, რომელიც საერთაშორისო ბაზარზე ყოველწლიურად ძლიერდება.

მონაცემთა საცავის ფართო გამოყენება, მსოფლიო ბიზნეს გაერთიანებაში ადასტურებს იმ ფაქტს, რომ ამ ტექნოლოგიაზე დაყრდნობით მოგება ყოველწლიურად იზრდება ათობით მილიარდი დოლარით. ბანკებში ჩატარებულმა ანალიზმა აჩვენა, რომ საბაზო ტრანზაქციების და სასურველ მონაცემებზე არსებული ინფორმაციის მოცულობა ძალიან დიდია. პირველადი ანალიზისათვის უნდა მომზადდეს გაფართოებული მონაცემები და მიეცეს ინდექსაცია. ამისათვის საჭიროა ფართო კომპიუტერული რესურსები, რომელიც საშუალებას იძლევა მცირე დროის განმავლობაში შესრულდეს ძებნა რამდენიმე ცხრილიში, რომლიც ათობით მილიონ ჩანაწერს შეიცავს და განხორციელდეს მონაცემთა შერჩევა.

მონაცემთა საცავი არის კომპლექსური სისტემა, რომელიც შედგება შემდეგი ძირითადი ფუნქციური ბლოკებისაგან:

- მონაცემთა განაწილებული, რელაციური ბაზების მართვის სისტემები;
- ინფორმაციის წყაროებიდან ოპერატიულ მონაცემთა ჩატვირთვის და გარდაქმნის საშუალება;
- საცავის დაპროექტების მეთოდური და ინსტრუმენტული საშუალებანი;

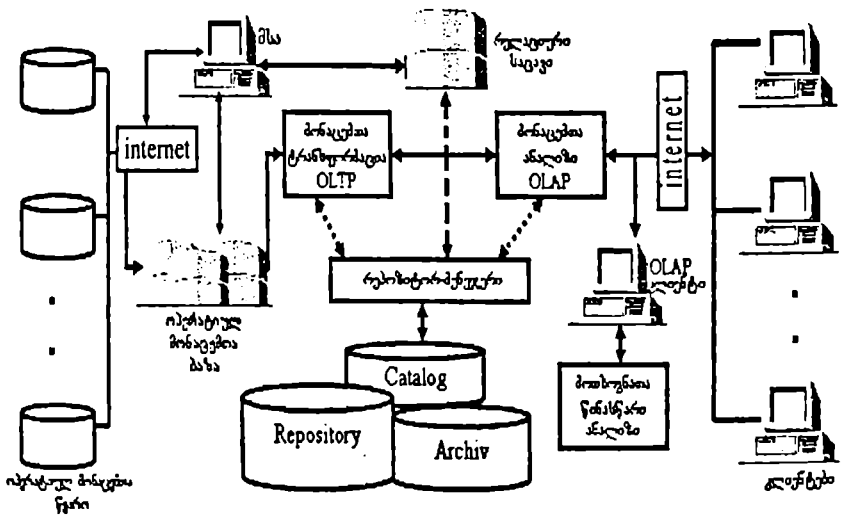
- საცავის აგებისა და მოდიფიკავციის საშუალებანი;

- საცავის მეტამონაცემთა იერარქიული ორგანიზების ჰიპერლინკური საშუალებანი;

- საცავის ფუნქციურ მომხმარებელთა მოთხოვნების წინასწარი ანალიზისა და ტრანზაქციების ეფექტურად შესრულების დაგეგმვის საშუალებანი;

- საცავის საინფორმაციო ბლოკებისა და არქივის ოპერატიული ანალიზის ინსტრუმენტული საშუალებანი.

2.1 ნახაზზე მოცემული გვაქვს განაწილებული ავტომატიზებული მართვის სისტემის მონაცემთა საცავის ზოგადი სქემა.



ნახ.2.1.

მონაცემთა საცავის მუშაობის პრინციპი ასეთია: პირველ ეტაპზე DWH–ს ტექნოლოგიის გამოყენების საშუალებით, რელაციურ ბაზებში ერთად თავმოყრილი მონაცემები ლაგდება გარკვეული სტრუქტურული თანამიმდევრობით, ხდება მათი „დაწმენდა". მეორე ეტაპზე წარმოებს მათი ტექნოლოგიური დამუშავება მონაცემთა ოპერატიული ანალიზის OLAP – ტექნოლოგიის გამოყენებით. ხოლო მესამე ეტაპზე ეს მონაცემები მომხმარებლებს მიეწოდებათ ინტერნეტის საშუალებით.

ინფორმაციული ბლოკები, რომლებიც მონაცემთა საცავებშია განაწილებული, მიზანმიმართულად თავსდება ინტერნეტ – გვერდებზე და ხელმისაწვდომია ფართო მომხმარებლისთვის.

მონაცემთა საცავის კონცეფციის ავტორი, ვ. ინმონი აღნიშნავდა, რომ მონაცემთა საცავი არის: „საგნობრივ ორიენტირებულ მონაცემთა ქრონოლოგიური ნაკადი, რომლის მიზანსაც ორგანიზებული მართვა წარმოადგენს“ [47,48].

მონაცემთა რელაციურ მოდელებზე აგებულ ბაზებისა და საცავებისათვის SQL (Structured Query Language) მონაცემთა სტრუქტურირებადი ენა არის ერთ–ერთი ეფექტური, საუკეთესო საშუალებაა მონაცემთა მანიპულირებისათვის. იგი აღიარებულია საერთაშორისო სტანდარტად, ამიტომაც ინფორმაციული საცავებისათვის მიზანშეწონილია რელაციური პლატფორმა.

რელაციური ცხრილებით წარმოდგენილ მონაცემებსა და საინფორმაციო ბლოკებს შორის არსებობს ლოგიკურ ურთიერთმიმართებათა ოთხი სახე, რომლებიც გამოიყენება კონცეპტუალური მოდელირებისათვის:

1. ერთი-ერთთან (1:1) – განხილულ ატრიბუტს აქვს იერარქიული ატრიბუტის ჩვეულებრივი რეკვიზიტის სახე. მაგ., ორგანიზაციის სრული დასახელება, რომელიც კავშირშია იურიდიულ პირთა ცნობარში, მისი რეგისტრაციის უნიკალურ ნომერთან;

2. მრავალი-ერთთან (M :1) – განხილული ატრიბუტი შეიძლება წარმოადგენდეს რომელიმე ცნობარის ელემენტს, რომელიც შესაბამისად არის მნიშვნელოვანი ატრიბუტი. მაგ., ორგანიზაციულ – ტერიტორიული კავშირის აღწერა, რომელშიც წარმოდგენილია ორგანიზაცია, თავისი უნიკალური კოდით, ხოლო ტერიტორია შეიძლება წარმოვადგინოთ, როგორც თეორიულად შეუზღუდავი ორგანიზაციათა სიმრავლე. ე.ი. ატრიბუტი „ტერიტორია “ აღწერს ორგანიზაციებსაც;

3. ერთი-მრავალთან (1:M) – ატრიბუტი წარმოადგენს ეგზემპლიარების განუსაზღვრელ რაოდენობას, ხოლო განხილული ატრიბუტის ეგზემპლიარი დაკავშირებულია მხოლოდ ერთ იერარქიულად მაღლა მდგომ ეგზემპლიართან. მაგ., საბაზო ანგარიშებსა და ერთ მესაკუთრეს (იურიდიულ პირს) შორის კავშირი;

4. მრავალი-მრავალთან (M:N) – განსახილველი ატრიბუტები ერთმანეთს უკავშირდება მრავალი საშუალებით, რაც რომელიმე ერთ ეგზემპლიარს საშუალებას აძლევს დაუკავშირდეს ნებისმიერ დანარჩენს. მაგ., იურიდიული პირების და დამფუძნებელთა სია. ყოველი იურიდიული პირი შეიძლება იყოს დამფუძნებელი და ყოველი დამფუძნებელი - იურიდიული პირი.

მონაცემთა საცავი ორიენტირებულია განსაზღვრულ საგნობრივ სფეროზე და ორგანიზებულია მონაცემთა ოპერატიული ბაზიდან შემოსულ სტრუქტურულად გადამუშავებულ მონაცემთა ქვესიმრავლეების საფუძველზე. ინფორმაციის წყაროს წარმოადგენს სხვადასხვა ორგანიზაციათა დანართები (აპლიკაციები), რომლებიც გამოიყენებს განსხვავებულ პროგრამულ პლატფორმებს და უკავშირდება ოპერატიულ მონაცემთა ბაზას ინტერნეტის საშუალებით (on-line რეჟიმი). შესაძლებელია აგრეთვე სხვა სახის კავშირების (off-line რეჟიმი) გამოყენებაც.

მონაცემთა საცავში ინახება მონაცემთა სტრიქონების არა მთელი სიმრავლე, არამედ ამა თუ იმ ხარისხით გაერთიანებული (აგრეგატული) ინფორმაცია, რაც ხელს უწყობს მეხსიერების ეფექტურად გამოყენებას.

დღეს აზრთა სხვადასხვაობას იწვევს ის საკითხი, თუ რა განსხვავებაა მონაცემთა საცავს, მონაცემთა ბაზასა და ოპერაციულ სისტემას შორის. ამ განსხვავების ასახსნელად მნიშვნელოვანია თვით მონაცემთა საცავის ფუნქციური ხასიათის გაგება.

ნებისმიერი ოპერაციული სისტემა ან მონაცემთა ბაზა ორიენტირებულია კომპიუტერული რესურსების მართვის ან მონაცემთა დამუშავების ოპერაციებზე, მაშინ როდესაც მონაცემთა საცავში ყველაზე მნიშვნელოვანია თვით საინფორმაციო ობიექტი და მისი მიზნობრივად დამუშავებული შედეგების მომხმარებელზე მიწოდების მხარდაჭერა. საცავის ოპერატიულ მონაცემთა ბაზაში ინახება მხოლოდ აქტუალური (ახლად შემოსული) მონაცემები, მაშინ როდესაც საცავში და არქივში თავმოყრილია, კომპანიების მუშაობის ისტორიის მთელი საინფორმაციო სპექტრი. ისტორიული მონაცემების შენახვა არის ერთ-ერთი აუცილებელი პირობა და იგი წარმოადგენს მონაცემთა საცავის მთავარ ღირსებას, რადგან ამ ინფორმაციის დახმარებითაა შესაძლებელი სრულყოფილ ანალიზის წარმოება.

საცავში მონაცემები ინახება ცალკეული ფაქტების ცხრილების სახით, რომელთა განხილვაც შესაძლებელია სხვადასხვა კუთხით.

საბაზო სისტემების მონაცემთა საცავში ინფორმაცია თავმოყრილია არა მხოლოდ ანალიზისათვის. შეიძლება ინფორმაცია მივიღოთ საბაზო საქმიანობის ისეთ ფუნქციურ საკითხებზე, როგორიცაა საბუხჰალტრო საქმე, ბანკის მართვა, სახაზინო საქმე, კადრების მართვა და ა.შ.

საცავში არსებული ინფორმაციის საფუძველზე გადასაწყვეტი საკითხები მონაცემთა ანალიზის ინსტრუმენტის დახმარებით შეიძლება

განხილულ იქნეს სხვადასხვა დონეზე, დაწყებული გაერთიანებული ანგარიშების შემადგენლობიდან, დამთავრებული ბიუჯეტით და ბანკის საქმიანობის ანალიზით.

კლიენტთა ბაზის და საბაზო პროდუქტების შეფასების საშუალება, მათი სტრუქტურული და დინამიკური ცვლილების ანალიზი, აგრეთვე აქტივის და პასივის ხარისხის შეფასება და ბანკებში მიმდინარე ბიზნეს-ოპერაციები, საშუალებას გვაძლევს დროულად შევცვალოთ ტაქტიკური და სტრატეგიული გადაწყვეტილებები.

მაგალითად, მონაცემთა საცავის ანალიზის დახმარებით შესაძლებელია გადავჭრათ ისეთი რთული ამოცანა, როგორიცაა სხვადასხვა ობიექტების ეკონომიკური უსაფრთხოების (იგულისხმება გაკოტრება) დაცვა.

2.2. მონაცემთა საცავის დაპროექტების მეთოდურ-ორგანიზაციული საფუძვლები

მონაცემთა საცავის ცნება აქტუალური გახდა 2000 წლიდან და იგი დღემდე, საინფორმაციო სიტემების დაპროექტებაში, პერსპექტიულ მიმართულებად ითვლება. საწყის ეტაპზე საინფორმაციო სისტემების დაპროექტება ვითარდებოდა „ინფოლოგიკური“ გზით, ხოლო ახალი მიმართულების განვითარების ტენდენციამ ფართო გამოვლინება ჰპოვა „მონაცემთა საცავის“ სახით. ინფორმატიკაში როგორც ბევრი ტერმინი, ასევე ეს ცნებაც მოკლებულია ზუსტ განმარტებას, რაც პრაქტიკაში წარმოქმნის შეუთავსებლობას. არსებობს განმარტების მრავალგვარი ახსნა რომელთაგან ერთ-ერთი ასეთია:

მონაცემთა საცავი არის რომელიმე ორგანიზაციისთვის განკუთვნილი სპეციალური ბაზა, სადაც მიმდინარე ოპერატიული სამუშაოს შესრულებისას თავს იყრის ქრონოლოგიურ ინფორმაციათა მთელი სპექტრი, რომელთა დანიშნულებაცაა მომხმარებელს მიაწოდოს ინტერნეტ გვერდებზე მიზნობრივად განლაგებული საინფორმაციო ბლოკები.

ასეთი სახის მონაცემთა ბაზის გამოყენება იმ შემთხვევაშია შესაძლებელი, თუ უზრუნველყოფილ იქნება შემდეგ ფუნქციათა შესრულება:

F1( ) - მონაცემთა შერჩევა.

„მონაცემთა მოხვედრა საცავში“- არის მთელი რიგი პროცესებისა, რომელიც ემყარება გარკვეულ კანონზომიერებას. პირველ რიგში ერთ-ერთი მთავარი პირობაა, რომ მონაცემები საცავში უნდა მოხვდეს საჭირო სახით, ხოლო მეორე რიგში უნდა ხდებოდეს ამ მონაცემთა რეგულირება საჭიროებისამებრ. „საჭირო სახეში“ იგულისხმება, როგორც მინიმუმ, ინფორმაცია უნდა მოვიყვანოთ საჭირო ფორმატში. ერთი და იმავე სახის მონაცემი შეიძლება მივიღოთ სხვადასხვა წყაროებიდან განსხვავებული ფორმატით, რაც წარმოქმნის შეუთავსებლობას, ამიტომ უნდა მოხდეს დასახელებათა უნიფიცირება.

მონაცემთა დამუშავების კიდევ ერთ სახეს წარმოადგენს მონაცემთა პირველადი გადამუშავება. საცავში წარმოდგენილ მონაცემებს თუ სიღრმისეულად დავუკვირდებით, შევნიშნავთ რომ სხვადასხვა კონტექსტში განხილვისას ისინი იძენს სხვადასხვა მნიშვნელობებს, რამაც შეიძლება მიგვიყანოს არასასურველ შედეგებამდე. პირველადი დამუშავებისას ხდება შეცდომების გამოაშკარავება სტატისტიკური დამუშავების სხვადასხვა საშუალებებით, შესაძლებელია უკვე გამოყენებულ მონაცემთა ჩამოცილება და აგრეთვე გამოტოვებულ მონაცემთა აღდგენა. ოპერატიული დამუშავების სისტემიდან წამოსული მონაცემები არ არის ერთჯერადი გამოყენების, ამიტომ მისი ნორმალური ფუნქციონირებისათვის უნდა მოქმედებდეს საცავში მონაცემთა გადაცემის პროცედურის შემსრულებელი პროგრამები და მათი პირველადი დამუშავება კავშირში უნდა იყოს გარე მოვლენებთან.

F2( ) - მონაცემთა ურთიერთშეთანხმება.

როგორც ყველა მონაცემთა ბაზა, მონაცემთა საცავიც შეიძლება განაწილებული იყოს კომპიუტერული ქსელის კვანძებში, სადაც მოთავსებული ინფორმაცია მიიღება განსხვავებული წყაროებიდან. იმისათვის, რომ ურთიერთშეთანხმებულად მოქმედებდეს ინფორმაციის მომწოდებელი სხვადასხვა წყაროები, მონაცემთა საცავის ფუნქციონირების ქვესისტემა აუცილებელია სარგებლობდეს მონაცემთა სტრუქტურის აღწერით. ასეთი აღწერისათვის იყენებენ ლექსიკონ-ცნობარს, სადაც თავმოყრილია ცნობები – მონაცემთა ფორმატზე, სტრუქტურაზე, ინფორმაციის მიმღებ არხებზე, წყაროებზე და ა.შ.

F3( ) -მონაცემთა მოპოვება.

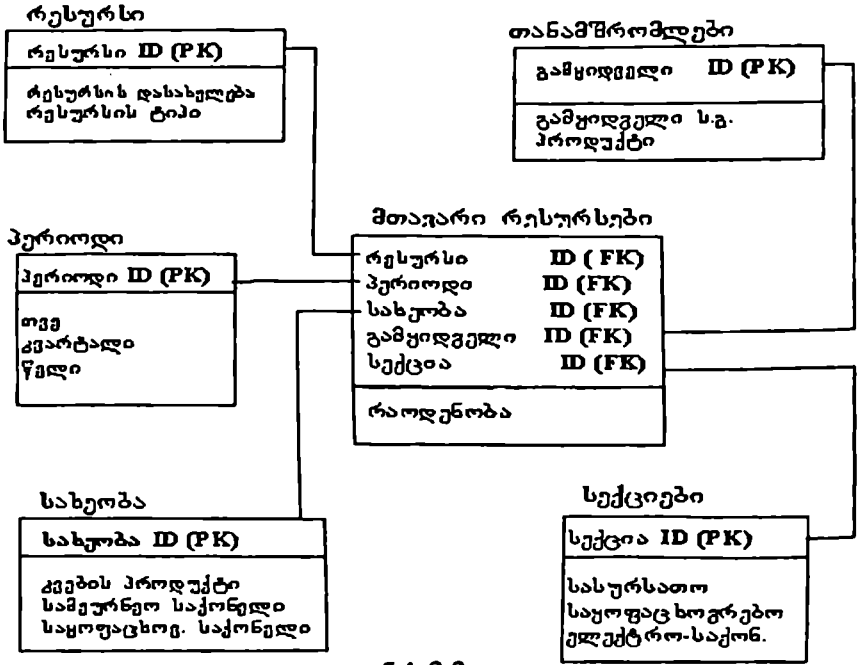
მონაცემთა საცავსა და ოპერატიულ მონაცემთა ბაზას შორის არსებული განსხვავების შესაბამისად, არსებობს მათთან მიდგომის განსხვავებული საშუალებებიც. მონაცემთა საცავთან დაკავშირებისთვის გამოიყენება არარეგლამენტირებულ მოთხოვნათა სისტემა (ad hoc query). ეს სისტემა მოთხოვნათა ფორმულირებისას გამოირჩევა მოქნილობით იგი საშუალებას გვაძლევს მონაცემებზე ანალიზის ჩატარებისას ვისარგებლოთ როგორც ზოგადი, ასევე მათი დეტალური ანალიზით.

ანალიზის მეორე სახე, რომელიც გამოიყენება ასეთ სისტემებში, გათვლის შესაძლო შემთხვევებს (what-if-analysis), როგორიცაა: რეალურ მონაცემებში, რამდენიმე მაჩვენებლის შეცვლა და მათი ანალიზი, რის შედეგადაც იცვლება საერთო მდგომარეობა. ზოგადად, არარეგლამენტირებულ მოთხოვნათა სისტემის მომსახურებისთვის შეიძლება გამოვიყენოთ 'მონაცემთა დამუშავების ოპერატიულ ანალიზური სისტემა" (OLAP) [46,54,60].

მონაცემთა საცავი ეყრდნობა რელაციურ და მრავალგანზომილებიან ბაზებს. პირველადი სახის მონაცემები ინახება რელაციურ მონაცემთა ბაზაში (უფრო ზუსტად რელაციურ-ორიენტირებულში) და განთავსებულია სპეციფიურ სქემაში, რომელსაც „რადიალური" სქემა ან (star schema), ეწოდება.

ვ. ინმონი მონაცემთა საცავის კონკრეტულ სტრუქტურას თითქმის არ განიხილავს, ხოლო რელაციური ბაზების ფუძემდებელმა, ე.ფ. კოდმა წარმოადგინა ოპტიმიზირებული მონაცემთა ბაზების სქემა [33,34,35]. მას გააჩნია ერთი მთავარი ცხრილი, რომელთანაც დაკავშირებულია რამდენიმე დამოკიდებული ცხრილი, მთავარი ცხრილის საშუალებით ხდება დამოკიდებულ ცხრილთა მონაცემებით მომარაგება. მაგ., კომერციული ობიექტის სტრუქტურის რადიალური სქემა (ნახ.2.2). მთავარი ცხრილი ანუ ფაქტების ცხრილი ასახავს კვლევის ობიექტს, იგი შესაძლოა შედგებოდეს მილიონობით სტრიქონისგან.

მონაცემთა ორგანიზების განსხვავებული მიდგომაა მონაცემთა საცავში, კერძოდ მრავალგანზომილებიანი სისტემის გამოყენება. აქ მონაცემები ინახება არა ცხრილში არამედ მრავალგანზომილებიან კუბში. ჰიპერკუბის მოდელი მომხმარებლისთვის არის თვალსაჩინო და OLAP ანალიზისთვის ხელსაყრელი.



ნახ.2.2

მონაცემთა მრავალგანზომილებიან წარმოდგენას ახლავს გარკვეული სირთულეებიც, ინფორმაციის დამუშავების მეთოდოლოგიის შერჩევის თვალსაზრისით [45].

F4( ) - კლასიფიკაცია.

ეს არის მონაცემთა კვლევის ერთ-ერთი ცნობილი მეთოდი, სადაც განიხილება მონაცემთა ტიპები (კლასები). კლასი არის ერთი ტიპის ობიექტების სიმრავლე, რომელთაც აქვთ მსგავსი სტრუქტურა და ქცევა [50]. მათი საშუალებით შეგვიძლია განვსაზღვროთ მოვლენები, სიტუაციები და პროცესები.

ობიექტები (ობიექტი განიხილება, როგორც გარკვეული არსი რომელიც ხასიათდება მდგომარეობით და ქცევით) უნდა აღიწეროს შემდეგი სიდიდეებით, როგორიცაა: სიმპტომები, მაჩვენებლები, პარამეტრები. ინფორმაცია ყოველი კლასის შესახებ მოცემულია ობიექტების დახმარებით (დაკვირვება, პრეცენდენტი) აქედან გამომდინარე შეგვიძლია განვსაზღვროთ, ობიექტთა გარკვეულ ჯგუფი თუ რომელ კლასს მიეკუთვნება.

უნდა ვიპოვოთ კრიტერიუმები,რათა განვსაზღვროთ თუ რომელი კლასიფიკაციის კატეგორიას მივაკუთვნოთ ობიექტი. კრიტერიუმის შერჩევა ეყრდნობა უკვე კლასიფიცირებული ობიექტის ხასიათის შესწავლას.

ამოცნობის ალგორითმი გამოიყენება მედიცინაშიც, როდესაც გარკვეული სიმპტომებით და ამბულატორიული გამოკვლევის მონაცემებით ხდება დაავადების დიაგნოსტიკა, აგრეთვე ტექნიკაში, როდესაც მაკონტროლებელი მოწყობილობის მაჩვენებლებით და ექსპერტული მონაცემების შეჯერებით შეგვიძლია დავადგინოთ ოპტიმალური კრიტერიუმები [45].

ბიზნესის ბევრ სფეროში შეინიშნება მუდმივი დამკვეთის დაკარგვის პრობლემა. კლასიფიკაციის ინსტრუმენტის გამოყენებით შესაძლოა გამოვყოთ დამკვეთის მოთხოვნათა ხასიათის შესაბამისი მონაცემთა ჯგუფი [41]. სავარაუდოა, რომ წინასწარ შექმნილ მოდელს გამოუჩნდება სასურველი დამკვეთი, ასეთი სახის მოდელი შესაძლოა იყოს დამკვეთზე ზემოქმედების ისეთი საშუალება, როგორიცაა რეკლამა, იგი საშუალებას იძლევა სხვადასხვა კატეგორიის დამკვეთის მოსაზიდად.

F5( ) –კლასტერიზაცია.

კლასტერიზაცია მოგვაგონებს კლასიფიკაციის ფორმას, იმ განსხვავებით, რომ კლასიფიკაციის კრიტერიუმები არ არის მოცემული. კლასტერიზაცია მონაცემთა კვლევისას საშუალებას გვაძლევს აღმოვაჩინოთ მონაცემები, რომლებიც დაჯგუფებულია რაიმე ნიშან თვისებების მიხედვით ისე, რომ ერთ ჯგუფში გაერთიანებული მონაცემები „მსგავსია“, ხოლო სხვადასხვა ჯგუფში „არაა მსგავსი“.

კლასტერიზაციის ალგორითმი, როგორც პირველადი ანალიზის ინსტრუმენტი, შეუცვლელია მრავალგანზომილებიან მონაცემთა ჯგუფის დამუშავებისას, ამასთან იგი ძალიან ეკონომიურია [61].

F6( ) – ოპერატიული ანალიზი OLAP-ინსტრუმენტით.

მონაცემთა საცავების დაპროექტებისა და მისი ფუნქციონირებისათვის, მეთოდური თვალსაზრისით ჩვენ ვიყენებთ ე.ფ. კოდის მიერ ჩამოყალიბებულ პრინციპებს[58]. ესაა ის 12 წესი, რომელსაც უნდა აკმაყოფილებდეს ნებისმიერი განაწილებული სისტემა მონაცემთა საცავით, რათა ჩატარდეს საინფორმაციო ბლოკების სრულფასოვანი ოპერატიული ანალიზური სამუშაოები.

1. მონაცემთა მრავალგანზომილებიანი კონცეპტუალური წარმოდგენა. მომხმარებელ-ანალიტიკოსი საპრობლემო სფეროს, თავისი ბუნების მიხედვით ხედავს როგორც მრავალგანზომილებიანს. შესაბამისად OLAP- მოდელიც უნდა იყოს მრავალგანზომილებიანი. ასეთი ტიპის კონცეპტუალური სქემა (მომხმარებელთა წარმოდგენები) აიოლებს მოდელირებას, ანალიზს და გამოთვლებს;

2. გამჭვირვალეობა. OLAP წარმოდგენილი უნდა იყოს ღია არქიტექტურის კონტექსტში, სადაც მომხმარებელს საშუალება ექნება დროის ნებისმიერ მონაკვეთში, ანალიზური ინსტრუმენტის საშუალებით დაუკავშირდეს სერვერს და მიიღოს მისთვის სასურველი ინფორმაცია;

3. მიღწევადობა. OLAP – ის მომხმარებელ ანალიზატორს უნდა ჰქონდეს ანალიზის ჩატარების საშუალება, რომელიც ემყარება საერთო კონცეპტუალურ სქემას, რომელშიც განთავსებულია რელაციური მონაცემთა ბაზა სწარმოთა შესახებ არსებული ყველა ახალი და ძველი მონაცემებით. ეს ნიშნავს, რომ OLAP – მა უნდა წარმოადგინოს თვისი საკუთარი ლოგიკური სქემა, რათა შეასრულოს შესაბამისი გარდაქმნა და მომხმარებელს წარუდგინოს მონაცემები. გარდა ამისა აუცილებელია წინასწარ იმაზე ზრუნვა, თუ სად, როგორ და როგორი სახის ფიზიკური ორგანიზაციის მონაცემები იქნას გამოყენებული. OLAP სისტემამ უნდა შეასრულოს ისეთი მონაცემების დამუშავება, რომელთა მოთხოვნაც რეალურად არსებობს;

4. ანგარიშთა დამუშავებისას მუდმივი წარმადობა. თუ ანალიტიკოსის მიერ ჩატარებული გაზომვათა რაოდენობა ან მონაცემთა ბაზების რიცხვი მნიშვნელოვნად იზრდება, მომხმარებელ ანალიზატორისთვის ეს პროცესი უნდა დარჩეს შეუმჩნეველი და არ უნდა აისახებოდეს საწარმოო პროცესების წარმადობის შემცირებაზე;

5. კლიენტ სერვერის – არქიტექტურა. მონაცემთა დიდი ნაწილი, რომელიც მოითხოვს ოპერატიულ ანალიზურ გადამუშავებას, უნდა მუშაობდეს კლიენტ-სერვერულ რეჟიმში. ამ თვალსაზრისით აუცილებელია, რომ ანალიზური ინსტრუმენტის სერვერული კომპონენტები იყოს „ინტელექტუალური“, რადგან განსხვავებულ კლიენტებს შეეძლოთ დაუკავშირდნენ სერვერს და გამოიყენონ პროგრამული პაკეტები. „ინტელექტუალურ“ სერვერს უნდა შეეძლოს მონაცემთა ბაზის შეუთავსებადი ლოგიკური და ფიზიკური სქემის ასახვა და გაერთიანება. ეს უზრუნველყოფს გამჭვირვალეობას და

საშუალებას იძლევა აიგოს საერთო კონცეპტუალური, ლოგიკური და ფიზიკური სქემები;

6. მრავალგანზომილება. გაზომვის ყოველი მცდელობის დროს გამოყენებული უნდა იყოს მიუკერძოებელი სტრუქტურა და ოპერაციული შესაძლებლობა. დამატებითი ოპერაციული შესაძლებლობა უნდა მიეცეს ერთ-ერთ რომელიმე ცდას და თუ ეს გაზომვა სიმეტრიული იქნება სხვა გაზომვის შედეგების, მაშინ ცალკე აღებული ფუნქცია შეიძლება წარმოვადგინოთ ნებისმიერი გაზომვის სახით;

7. დინამიკური მართვა გამონთავისუფლებული რეჟიმით. **OLAP** – ინსტრუმენტის ფიზიკური სქემა უნდა ადაპტირდებოდეს სპეციფიკურ ანალიტიკურ მოდელთან, რათა ოპტიმალურად მართოს გამონთავისუფლებული მატრიცა. ერთი დაცლილი მატრიცისთვის არსებობს ერთადერთი ოპტიმალური ფიზიკური სქემა. **OLAP** – ინსტრუმენტის ბაზური ფიზიკური მონაცემები პრაქტიკული ოპერაციებისთვის, რომელთაც აქვთ დიდი ანალიზური მოდელი უნდა კონფიგურირდებოდეს ნებისმიერ ქვესიმრავლესთან. თუ **OLAP** – ინსტრუმენტს არ შეუძლია გააკონტროლოს და დაარეგულიროს საანალიზებელი მონაცემების მნიშვნელობები, ის ჩაითვლება უსარგებლოდ და არასაიმედოდ;

8. მრავალმომხმარებლობა. ხშირ შემთხვევაში მომხმარებელ-ანალიტიკოსი დასმულ მოთხოვნებს აყენებს ერთ ანალიზურ მოდელთან ან ქმნის განსხვავებულ მოდელს ერთი სახის მონაცემებიდან. **OLAP** – ინსტრუმენტი კი საშუალებას გვაძლევს მივიღოთ უსაფრთხო, სრულყოფილი და ზუსტი ანალიზური შედეგები;

9. შეუზღუდავი გადამკვეთი ოპერაციები. მონაცემთა შემოწმების სხვადასხვა დონე და გაერთიანების გზა, მათი იერარქიული ბუნების გათვალისწინებით მჭიდრო კავშირშია **OLAP** - მოდელთან ან დანართთან. თვითონ ინსტრუმენტი უნდა მოიაზრებოდეს შესაბამის გამოთვლებთან და არ უნდა მოსთხოვოს მომხმარებელს თავიდან განსაზღვროს გამოთვლები და ოპერაციები. გამოთვლები მოითხოვს რომელიმე გამოყენებულ ენაში განსხვავებული ფორმულების განსაზღვრას. ასეთი ენა შეიძლება გამოვიყენოთ ნებისმიერი სიდიდის მონაცემთა მანიპულირებისთვის და არ შეზღუდოს მონაცემები არსებული კუბის უჯრედებს შორის და კონკრეტული უჯრედების საერთო ატრიბუტებზე;

10. **მონაცემთა ინტუიციური მანიპულაცია.** მონაცემთა დეტალიზაციის, გაერთიანების და სხვა მანიპულაციები უნდა იყენებდეს ცალკეულ უჯრედებზე ანალიზური მოდელის შედეგებს და არ უნდა იყენებდეს მომხმარებლის ინტერფეისებს. მომხმარებელ ანალიტიკოსს უნდა ჰქონდეს ყველა აუცილებელი პირობა იმისა, რომ მიიღოს სრულყოფილი ინფორმაცია;

11. **ანგარიშების მიღების მოქნილი საშუალება.** შეტყობინებათა დამუშავება და პასუხის გაცემა უნდა იყოს მოქნილი და ელასტიური. მომხმარებელს უნდა შეეძლოს მონაცემთა კომბინირება და გაანალიზება. მოქნილობა მნიშვნელოვანია, რათა ყურადღება გამახვილდეს მონაცემთა განმასხვავებელ ნიშნებზე. თუ რაიმე სირთულე წარმოიქმნება, უნდა შევჩერდეთ ინდივიდუალურ ინფორმაციულ მოთხოვნაზე და შეირჩეს მხოლოდ მომლოდინე მოთხოვნა;

12. **შეუზღუდავი ზომები და აგრეგაციათა რაოდენობა.** გამოკვლევებმა აჩვენეს, რომ აუცილებელი გაზომვა ერთდროულად შეიძლება ჩატარდეს 19-ჯერ. აქედან გამომდინარე შეიძლება ვთქვათ, რომ ანალიზური ინსტრუმენტი საშუალებას გვაძლევს ერთდროულად ვაწარმოოთ 15-დან 20-მდე გაზომვა, ამასთან თითოეული გაზომვის მცდელობა არ არის შემოსაზღვრული დადგენილი რიცხვით.

ეს პირობები შეიძლება ჩავთვალოთ, ოპერატიული ანალიზური დამუშავების თეორიულ ბაზისად. როგორც უკვე ავღნიშნეთ OLAP-ში ძევს მონაცემთა დამუშავების მრავალგანზომილებიანი სტრუქტურის იდეა. როდესაც ვსაუბრობთ OLAP–ზე, უნდა ვიგულისხმოთ, რომ ეს არის მონაცემთა ლოგიკური სტრუქტურის მრავალგანზომილებიანი ანალიზური ინსტრუმენტი.

2.3. მონაცემთა ტიპიზაციის, კლასიფიკაციისა და კატალოგიზაციის ამოცანის გადაწყვეტა

საავტომატიზაციო ობიექტის განაწილებული სისტემის მონაცემთა საცავში განსათავსებელი საინფორმაციო ბლოკები შეივსება ოპერატიულ მონაცემთა ბაზებიდან, მათი წინასწარი დამუშავების საფუძველზე (მეთოდური საფუძვლები წინა პარაგრაფში იყო განხილული). როგორც ვიცით, ოპერატიულ ბაზაში საწყისი მონაცემები თავს იყრის გარე ორგანიზაციებიდან ინტერნეტის საშუალებით, ან სხვა სახის კომუნიკაციებიდან.

საინფორმაციო ბლოკები, ან უფრო ზოგადად, მონაცემთა მანქანური ფაილები განსხვავებული ტიპებისაა: ტექსტები და სუპერტექსტები (H), ცხრილები (T), გრაფიკები (G), აუდიო (A) და ვიზუალური (V) მასალა და ა.შ.

მათემატიკური მოდელი შეიძლება ჩავწეროთ ტეტრადით:

$$B = < R, M, S, F >, \quad \text{სადაც} \qquad (2.1)$$

R – საცავის კლასიფიცირებული ობიექტებია.

$R = \{ R_i \}, \quad i = \overline{1,5}$. აქ i კლასიფიკაციის ტიპია და იგი შეესაბამება ზემოაღნიშნულ ცვლადებს: {H, T, G, A, V}.

M – მონაცემთა ბლოკების მეტაინფორმაციაა, განთავსებული სამდონიან იერარქიულ კატალოგსა (C_R, C_A, C_V) და ინდექსურ ცხრილებში (T_I).

$$M = \{C_R, C_A, C_V, T_I\}. \quad \text{სადაც} \qquad (2.2)$$

C_R - საინფორმაციო ბლოკების აღწერის რელაციათა სიმრავლეა: $C_R = \{ C_{R_j} \}$, სადაც $j = \overline{1,m}$ რელაციათა ინდექსებია (რელაციათა სიმრავლის სიმძლავრე).

C_A - საინფორმაციო ბლოკების აღწერის ატრიბუტების სიმრავლეა: $C_A = \{ C_{A_k} \}$, სადაც $k = \overline{1,n}$ ატრიბუტების ინდექსებია (ატრიბუტთა სიმრავლის სიმძლავრე).

C_V - ატრიბუტების მნიშვნელობათა სიმრავლეა:

$C_V = \{ C_{V_z}^{A_k} \}$, სადაც k= A_k-ატრიბუტის ინდექსები, ხოლო z ამ ატრიბუტის შესაბამისი მნიშვნელობათა ინდექსებია (ატრიბუტის ქვესიმრავლის სიმძლავრე).

T_I - ინდექსების ცხრილის კორტეჟების (ან სტრიქონების, ჩანაწერების) სიმრავლეა: $T_I = \{ T_{A_k V_z}^{R_j} \}$, სადაც $j = \overline{1,m}$ რელაციათა ინდექსები, $k = \overline{1,n}$ A_k-ატრიბუტის ინდექსები, ხოლო V_z ამ ატრიბუტის შესაბამისი მნიშვნელობათა ინდექსებია. ინდექსების ცხრილის სვეტების რაოდენობაა n, ხოლო სტრიქონების რაოდენობა - m.

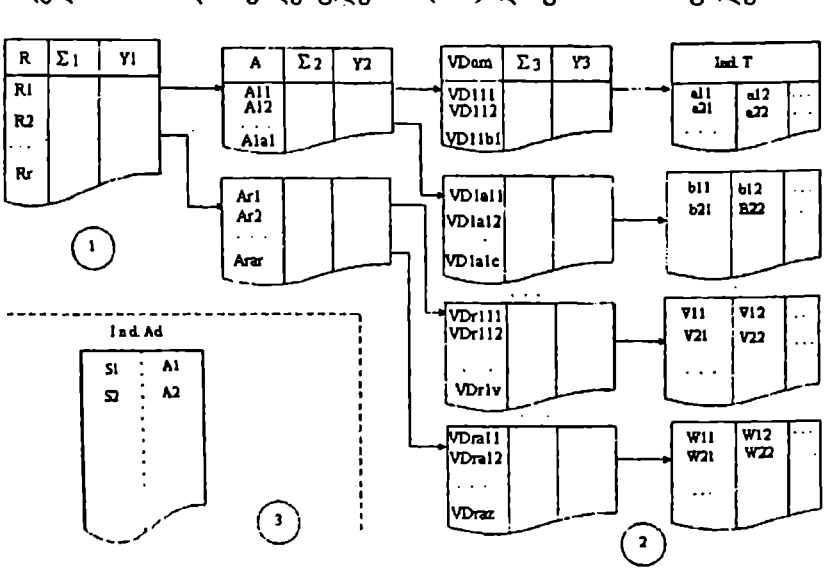
S – საცავის მეხსიერების ცენტრალური ცხრილია, რომელთა კორტეჟებშიც მოთავსებულია საინფორმაციო ბლოკების ფიზიკური

მისამართები და ინდექსური ცხრილების ელემენტთა მნიშვნელობები. ამგვარად, მყარდება ლოგიკური კავშირი მოთხოვნაში დასახელებულ j–ურ რელაციის k-ურ ატრიბუტის z-ურ მნიშვნელობასა და მეხსიერების ფიზიკურ a–ურ მისამართს შორის.

F – მონაცემთა საცავის ელემენტების (R), ანუ კლასიფიცირებული ობიექტების დამუშავების ოპერაციებია (კლასთა მეთოდებია). ჩვენ წინა პარაგრაფში განვიხილეთ ზოგიერთი ძირითადი ოპერაცია, მაგალითად: F1() - მონაცემთა შერჩევა, F2() - მონაცემთა ურთიერთშეთანხმება, F3() -მონაცემთა მოპოვება, F4() – კლასიფიკაცია, F5() –კლასტერიზაცია, F6() – ოპერატიული ანალიზი OLAP-ინსტრუმენტით და სხვ. აქ ემატება მონაცემთა ტრანსფორმაციის, კონვერტირების და რესტუქტურიზაციის ფუნქციებიც.

მომხმარებელთა მოთხოვნების დამუშავების ამოცანა განიხილება 2.4 პარაგრაფში.

2.3 ნახაზზე მოცემულია მონაცემთა საცავის მეტაინფორმაციის ორგანიზებისა და მართვის სამდონიანი კატალოგისა და ინდექსური ცხრილების გრაფიკული წარმოდგენა, 2.4-ზე კი ილუსტრირებულია საინფორმაციო ობიექტების დეტალური აღწერის კონცეპტუალური მოდელი არსთა-დამოკიდებულების (ER) დიაგრამის საშუალებით.



ნახ.2.3

პროგრამული რეალიზაციის საკითხი დაკავშირებულია მონაცემთა განაწილებული ბაზების მართვის სისტემის, მაგალითად, Ms SQL Server-2000-ის გამოყენებასთან.

პირველ ეტაპზე საჭიროა დაპროექტდეს მონაცემთა საცავის საინფორმაციო ობიექტების მეტაინფორმაციის სტრუქტურა, რომელიც საერთო იქნება ყველა ტიპის (H, T, G, A, V) კლასისათვის. ქვემოთ მოცემულია ამ სტრუქტურის ცხრილი შესაბამისი ატრიბუტებითა და განმარტებებით:

ინფორმაციულ-ობიექტთა მეტაინფორმაციის სტრუქტურა. ცხრ.2.1

№	ატრიბუტი	დასახელება	ტიპი	შენიშვნა
1.	ObjID	ობიექტის იდენტიფიკატორი	int	უნიკალური
2.	Shifri	ობიექტის შიფრი	varchar	
3.	Name	ობიექტის დასახელება	varchar	
4.	Category	ობიექტის კატეგორია	smallint	მეორადი გასაღები
5.	StateCode	მდგომარეობის კოდი	smallint	მეორადი გასაღები
6.	Summary	ობიექტის ანოტაცია	varchar	MEMO-ველი
7.	DescrID	ბირთად დესკრიპტორთა იდ.	smallint	მეორადი გასაღები
8.	FileSize	ფაილის ზომა ბაიტებში	int	
9.	CreatDate	ობიექტის შექმნის თარიღი	Date	
10.	ModifDate	ბოლო მოდიფიკაციის თარიღი	Date	
11.	PriceA	თვითღირებულება	Float	შენახვის ხარჯი
12.	PriceB	ფასი	Float	გასაყიდი ფასი
13.	PhizAdrID	ფიზიკური მისამართის იდენტ.	char	მეორადი გასაღები

2.4. მოთხოვნების წინმსწრები ანალიზისა და ტრანზაქციების სინქრონიზაციის სერიალიზაციის ამოცანის გადაწყვეტა

დაპროგრამების ობიექტ-ორიენტირებული მეთოდისა და უნიფიცირებული მოდელირების ენის (UML) საფუძველზე კლასი, თავისი მონაცემ-წევრებითა (ატრიბუტები) და ფუნქცია-წევრებით (მეთოდები), აგრეთვე კლასთაშორისი კავშირები (ClassAssotoation) ითვლება საინფორმაციო სისტემების პროგრამული უზრუნველოფის ძირითად კომპონენტებად [41,42].

ასევე მნიშვნელოვანია საკითხი აღნიშნული მეთოდების გააქტიურების მომენტების განსაზღვრისათვის. ეს მომენტები კი დამოკიდებულია დროზე და სისტემაში შემოსულ შეტყობინებებზე (მოთხოვნებზე).

შეტყობინებათა წინმსწრები ანალიზით შესაძლებელია დადგინდეს თუ რომელი საინფორმაციო ობიექტი აინტერესებს ამა თუ იმ მომხმარებელს (ან ჯგუფს).

წინმსწრები ანალიზის ალგორითმის მიზანია დროის ერთეულში შემოსული მოთხოვნების პაკეტის (რამდენიმე შეტყობინება) დესკრიპტორული ანალიზის ჩატარება და მათი დაკმაყოფილების მიზნით აუცილებელი და საკმარისი ინფორმაციული რესურსების მოძიებისა და პასუხების მომზადების მიმდევრობით-პარალელური ოპერაციების დაგეგმვის განხორციელება.

„პაკეტის“ წინმსწრები განხილვის აზრი მდგომარეობს იმაში, რომ გამოვლინდეს სხვადასხვა შეტყობინებაში ერთი და იმავე საინფორმაციო ობიექტებზე მოთხოვნა, შემდგომ ეტაპზე საცავში განმეორებითი ძებნის ოპერაციების გამოსარიცხად. ეს ამაღლებს სისტემის მწარმოებლურობას და ამცირებს მოთხოვნების დაკმაყოფილების საერთო დროს.

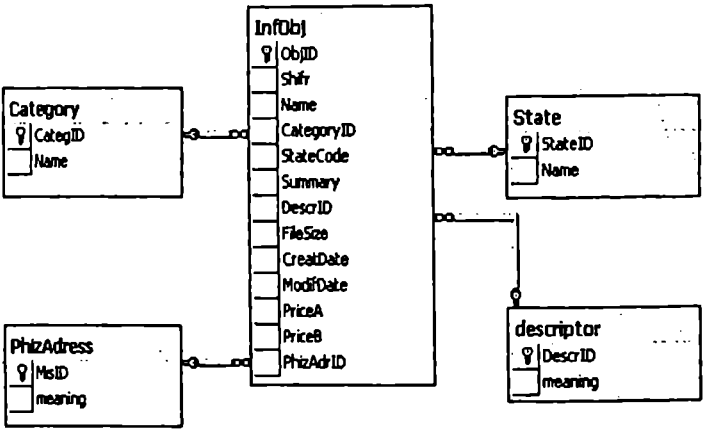
2.1 მათემატიკურ მოდელში შევიტანოთ მომხმარებლის მოთხოვნის, ანუ შეტყობინების ელემენტი (Q - Query). მოდელი მიიღებს შემდეგ სახეს:

$$B = < R, M, S, F, Q > \qquad (2.3)$$

ჩვენთვის აუცილებელია ლოგიკური მექანიზმის შემუშავება, რომელიც ყოველ კონკრეტულ შეტყობინებას ცალსახად განუსაზღვრავს მისთვის საჭირო კლასიფიცირებულ ობიექტთა სიმრავლეს მონაცემთა საცავიდან.

აღნიშნული ამოცანის გადასაწყვეტად წინასწარ საჭიროა დამუშავდეს მონაცემთა საცავის საინფორმაციო ობიექტებისა და მათი მეტაინფორმაციის (კატალოგების) სისტემის კლასთაშორისი-კავშირების დიაგრამა.

მისი აგება შესაძლებელია წინა პარაგრაფში დაპროექტებული კონცეპტუალური მოდელის საფუძველზე. 2.5 ნახაზზე ნაჩვენებია ობიექტ-ორიენტირებული დიაგრამის კომპიუტერული რეალიზაციის ფრაგმენტი.



ნახ.2.5

ამოცანის გადაწყვეტის რეალიზაციის მიზნით ჩვენ ვიყენებთ მონაცემთა რელაციური ბაზების მართვის სისტემის MsSQLServer_2000 პროგრამულ პაკეტს. ინფორმაციულ მონაცემთა და მათი მეტაინფორმაციის ძირითადი ცხრილები აგებულია რელაციური პრინციპით. მონაცემთა ძირითად დესკრიპტორთა სიმრავლისათვის შექმნილია ინდექსური ფაილები (სწრაფი მოძებნის ცხრილები). მომხმარებელთა შეტყობინებების ფორმირება ხორციელდება სტანდარტული SQL-ენის საფუძველზე.

მოთხოვნების დამუშავების პროცედურები (მაგ., სელექცია) ბაზებში ძირითადად შეიცავს ისეთ კრიტერიუმებს, რომელთა შესრულება ხორციელდება რელაციური ალგებრის ოპერაციებით ცხრილების კორტეჟებსა და ატრიბუტებზე.

2.3 მოდელში Q შეტყობინებათა წინმსწრები ანალიზის ჩასატარებლად ვიყენებთ:

- ე. კოდის რელაციური მოდელებისა და ალგებრის კონცეფციას:

$$R_a = < R, f > \text{, სადაც} \qquad (2.4)$$

R – რელციური დამოკიდებულებებია მონაცემთა საცავის საინფორმაციო ობიექტების ცხრილების ატრიბუტთა სიმრავლეზე განსაზღვრული, ხოლო f – სიმრავლეთა თეორიისა და კოდის ალგებრის ოპერაციები [50].

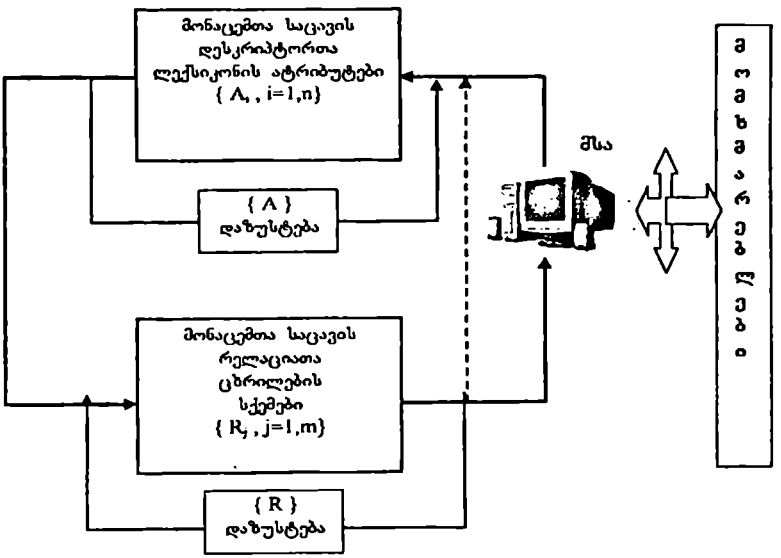
- პეტრის ქსელების გრაფულ წარმოდგენას :

$$P = < S, T, I, O > , \quad \text{სადაც} \qquad (2.5)$$

S – ქსელის პოზიციებია (ობიექტის ან შესაბამისი ცხრილის მდგომარეობები), T - ქსელის გადასასვლელები (რელაციური ოპერაციები გარკვული დაყოვნებით), I და O შესაბამისად არის შემავალი და გამომავალი ფუნქციები. ქსელის პოზიციაში შეიძლება იყოს მარკერი, რომელიც ამოდელირებს მაგალითად, ქსელში მოთხოვნის შემოსვლას, არსებობას ან შედეგის მიღებას.

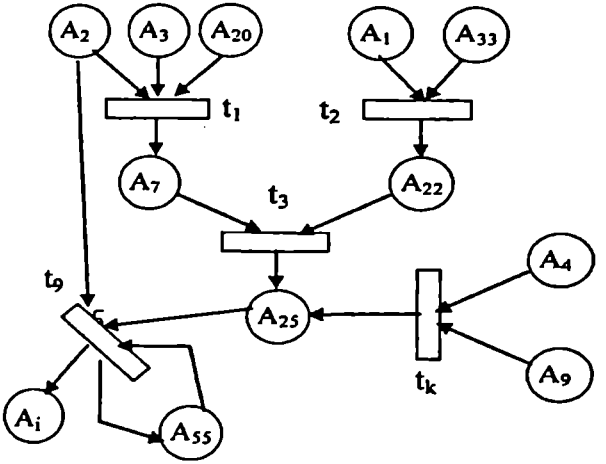
ვინაიდან მომხმარებლის მოთხოვნა პრედიკატულ ფორმაში ყალიბდება (SQL სტანდარტი), მასში ძირითადად ლოგიკური და რელაციური პროცედურები სჭარბობს.

რა თქმა უნდა მოთხოვნაში შეიძლება იყოს მათემატიკური, სტატისტიკური, აგრეგაციის, დაჯგუფებისა და სხვა სახის ფუნქციები, რომლებიც პროგრამულად მეთოდების სახითაა რეალიზებული და შენახულია პროგრამულ ბიბლიოთეკაში. 2.6 ნახაზზე ნაჩვენებია მომხმარებელთა მოთხოვნების წინმსწრები ანალიზის ორეტაპიანი მართვის პროცესის სქემა.

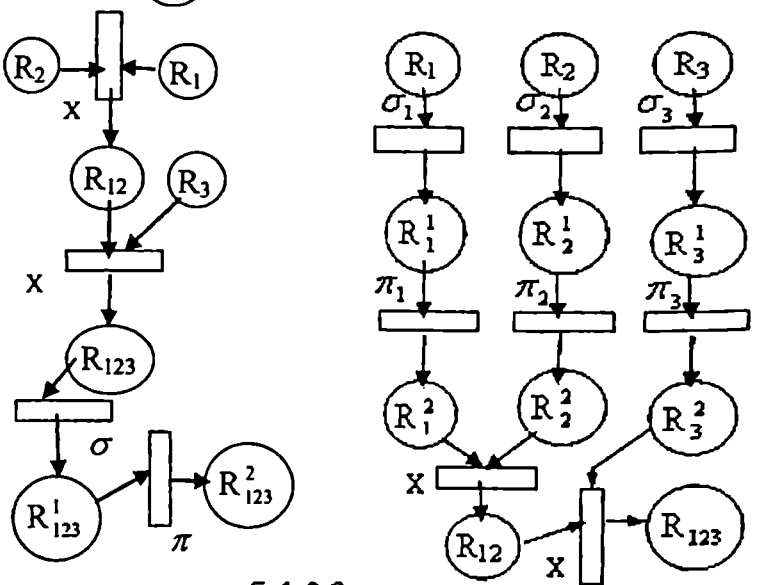


ნახ 2.6

მისი შესაბამისი პეტრის ქსელის გრაფის ფრაგმენტები ატრიბუტებისა და რელაციათა სქემებისათვის წარმოდგენილია 2.7 და 2.8 ნახაზებზე.



ნახ. 2.7



ნახ2.8

აღნიშნული საკითხის გადასაწყვეტად მულტიმომხმარებლურ რეჟიმისათვის ნაშრომში შემოთავაზებულია ტრანზაქციათა სინქრონიზაციის სერიალიზაციის ალგორითმი, რომელიც რეპოზიტორ-მენეჯერ პროგრამის კლასის ერთ-ერთ მეთოდს წარმოადგენს.

მომხმარებელთა ტრანზაქციები (საინფორმაციო ობიექტების წაკითხვა/ჩაწერის პროცედურები) იყენებს ქსელის საერთო გამოყენების რესურსებს, ამიტომაც საჭიროა მათი ბლოკირება-დებლოკირების კონფლიქტური სიტუაციების მართვა, რათა თავიდან ავიცილოთ „ჩიხური", უსასრულო მოლოდინის მდგომარეობები[13].

ტრანზაქციათა სერიალიზაცია კი უზრუნველყოფს მათ პარალელურ შესრულებას. რომელიმე საინფორმაციო ობიექტში ცვლილებების შეტანისას ერთი ტრანზაქციით იგი ბლოკირებულია სხვა ტრანზაქციებისათვის ამ პროცესის დამთავრებამდე.

ამასთანავე, მონაცემთა საცავის მთლიანობის მოთხოვნის დაცვის მიზნით რეპოზიტორ-მენეჯერი მთლიან საცავში (კატალოგებსა და განაწილებულ ბაზებში) განახორციელებს შესაბამის ცვლილებებს.

2.5. მონაცემთა საცავის პროგრამული კომპონენტები

მონაცემთა საცავი მოიცავს ტრანსფორმაციისა და კონვერტაციის პროგრამებს, საბაზო მეტამონაცემთა სისტემას, არქივირებული შენახვის სისტემას და ინტეგრირებულ მონაცემთა საცავს [9].

<u>მონაცემთა ბაზების სიტემა:</u>

მონაცემთა საცავს ემსახურება ისე, როგორც სხვა მონაცემთა ბაზების სისტემები, რომელთა საშუალებითაც ხდება, მონაცემთა დამოუკიდებელ პროგრამათა ინტეგრაცია, შენახვა და მართვა. მონაცემთა ბანკების მართვის სისტემა მომხმარებელთათვის უზრუნველყოფს ადამიანსა და კომპიუტერს შორის ინფორმაციის გაცვლას. იგი შეიცავს მონაცემთა დიდ რაოდენობას, რომელთა შორისაც დამყარებულია კანონზომიერი კავშირები. მონაცემთა ბანკების მართვის სისტემა შეადგენს მონაცემთა საცავის ცენტრალურ სისტემას.

<u>ტრანსფორმაციის პროგრამა:</u>

ახორციელებს ინტერფეისის ფუნქციას მონაცემთა საცავსა და მონაცემთა წყაროებს შორის. მონაცემები (ინფორმაცია) განსხვავებულ მონაცემთა ბაზებიდან (იერარქიულ, რელაციურ ან ობიექტ-

ორიენტირებულ) ან თანამიმდევრულ განსხვავებული ფორმატის ფაილებიდან (ASCII, ANSI, EBCDIC და ა.შ.) ექსტრადირდება. ტრანსფორმაციის წესის თანახმად ისინი ერთიანდება (Bridging), როგორც ინტეგრირებული, სუბიექტ-ორიენტირებული, მუდმივი და დროში ცვალებადი სტრუქტურები. ტრანსფორმაციის პროგრამამ უნდა უზრუნველყოს ტრანსპორტირებისათვის ფუნქციათა წარმოდგენა და აგრეთვე მონაცემთა მომზადება საცავში გადასაგზავნად. მონაცემთა საცავში შიგა მონაცემების (ინფორმაციის) უდიდესი ნაწილი შესაძლოა მიღებული იქნას განაწილებული ოპერატიული სისტემიდან. მონაცემთა საცავში ისტორიული და მიმდინარე მონაცემების შევსება, ხდება ბაზებიდან, სადაც პერიოდულად მიმდინარეობს მონაცემთა აქტუალიზაცია. თუ ოპერატიულ მონაცემთა ჩანაწერების რეგისტრაციაში მონაცემის შეტანის თარიღი უფრო ახალია, ვიდრე ბოლო ტრანსფორმაციის შეტანის დრო, მაშინ მოხდება ამ უკანასკნელის ლიკვიდაცია. ე.ი. აუცილებლად გათვალისწინებული უნდა იქნეს ცალკეულ ტრანსფორმაციის პროცესებს შორის ვადები.

მონაცემთა გარე მიმწოდებლებია შეტყობინებათა სამსახურები, ბირჟები, პოლიტიკური საინფორმაციო სამსახურები, სამეცნიერო კვლევითი ინსტიტუტები და ბაზარი. ინფორმაციის მოწოდებისათვის გამოიყენება ისეთი საშუალებები, როგორიცაა მაგალითად: Internet, CD-ROM, Flash-Memory, FD და ა. შ. ბევრი ამ მონაცემთაგან ტრანსფორმაციის პროგრამის საშუალებით, სანამ გადავა მონაცემთა საცავში უნდა წარმოდგეს გარკვეული Internet-სტანდარტული-ფორმატის სახით.

<u>მეტამონაცემთა საბაზო სისტემა:</u>

მეტამონაცემები არის მონაცემები - მონაცემების შესახებ. მეტამონაცემთა საბაზო სისტემა მონაცემთა საცავის შემადგენელი ნაწილია. მონაცემთა საცავში იგი ერთ-ერთ მთავარ როლს ასრულებს, მეტამონაცემების დანიშნულებაა მონაცემთა საცავში არსებული მონაცემების აღწერა და მათზე დამატებითი ცნობების მოგროვება. მეტამონაცემთა ინსტრუმენტის გამოყენება მომხმარებელს საშუალებას აძლევს აწარმოოს მონაცემთა მასივებში მანევრირება და ეხმარება მას მონაცემთა საცავში ორიენტირებაისათვის.

მეტამონაცემთა საბაზო სისტემა მომხმარებელს ეხმარება მოთხოვნების შესაბამისად მონაცემთა შერჩევაში. ეს ხორციელდება მონაცემთა საცავში მეტამონაცემთა გამჭვირვალე ასახვის შედეგად.

<u>არქივირებული შენახვის სისტემა;</u>

უზრუნველყოფს მონაცემთა დაცვას და მათ არქივირებულ შენახვას მონაცემთა საცავში. მონაცემთა არქივირებული შენახვა, როგორც ცალკე სისტემა, მონაცემთა საცავში ამცირებს მეხსიერების უჯრედებს და ზრდის მუშაობის ეფექტურობის ხარისხს. არქივირების ეფექტური სისტემა მნიშვნელოვანია, რადგან მოკლე ვადაში შესაძლებელია მონაცემთა გადმოტვირთვა მომხმარებელთა მოთხოვნების შესაბამისად.

ხშირად სისტემაში თავს იყრის უსარგებლო ინფორმაციათა ნაკადი და იკავებს დიდ ადგილს, რაც აფერხებს სისტემის მუშაობის ეფექტურობას, არქივირებული სისტემის დახმარებით ხდება ასეთი ინფორმაციის განადგურება. შესაძლოა ასევე დეფექტური ტრანზაქციის შედეგად მოხდეს მონაცემთა 'დაზიანება". ამ შემთხვევაში ამოქმედდება მონაცემთა დაცვის სისტემა, რაც უზრუნველყოფს დეფექტების აღმოფხვრას და არასასურველი ინფორმაციის განადგურებას.

არქივირებული შენახვისას ყურადღება უნდა გავამახვილოთ ინფორმაციის შენახვის კანონებზე, რომელიც ითვალისწინებს არქივში ინფორმაციის შენახვის ვადებს, რადგან შენახული ინფორმაცია გარკვეული პერიოდის შემდეგ კარგავს აქტუალობას.

<u>მონაცემთა საცავის ანალიზური დამუშავება OLAP – ტექნოლოგიის გამოყენებით.</u>

მონაცემთა მრავალგანზომილებიან კომპლექსური ანალიზის ტექნოლოგიას უწოდებენ OLAP (Online Analytical Processing) ტექნოლოგიას, რომელიც ნიშნავს „მონაცემთა ოპერატიული ანალიზს". მონაცემთა საცავში იგი წარმოადგენს მნიშვნელოვან კომპონენტს.

OLAP – ინსტრუმენტი პირველად მონაცემებს წარმოადგენს ინფორმაციის სახით, რომლის დახმარებითაც შესაძლებელი ხდება საწარმოს მოცულობის შესახებ ვიქონიოთ რეალური წარმოდგენა. ამავე დროს იგი უნიკალური ინსტრუმენტია, რომლიც საშუალებას გვაძლევს ანალიზური ჭრილით ჩავატაროთ ინფორმაციის მრავალგანზომილებოანი

გვაძლევს სხვადასხვა ანალიზური ჭრილით ჩავატაროთ ინფორმაციის მრავალგანზომილებიანი ანალიზი.

2.6. ოპერატიულ მონაცემთა ბაზის ინფორმაციის კონვერტირების ინსტრუმენტი

მონაცემთა საცავებში ინფორმაციული ბლოკები შეივსება ოპერატიულ მონაცემთა ბაზებიდან, სადაც გარე ორგანიზაციების საწყისი მონაცემები თავს იყრის ინტერნეტის ან სხვა სახის კომუნიკაციებიდან მოსული ინფორმაციით.

როგორც ანალიზმა გვიჩვენა, ელექტრონული ბიზნესისა და კომერციის ობიექტებზე (მაგ., დიდი სავაჭრო ცენტრები) თითქმის 60-70% გამოიყენება MsExcel-პაკეტზე აგებული ცხრილები (შეკვეთები, ფაქტურები და სხვა ანგარიშები). მონაცემთა საცავი რელაციური ტიპის ბაზებთან სამუშაოდაა ორიენტირებული, როგორებიცაა, მაგ., MsAccess, SQL Server-2000, Oracle, InterBase და ა.შ.

ამ მონაცემთა ბაზებს გააჩნიათ როგორც ერთმანეთთან, ასევე Ms Excel-თან მონაცემთა ცხრილების გაცვლის შესაძლებლობანი (Import/ Export, MsQuery - ფუნქციების ან ინსტრუმენტების სახით და ODBC დრაივერით).

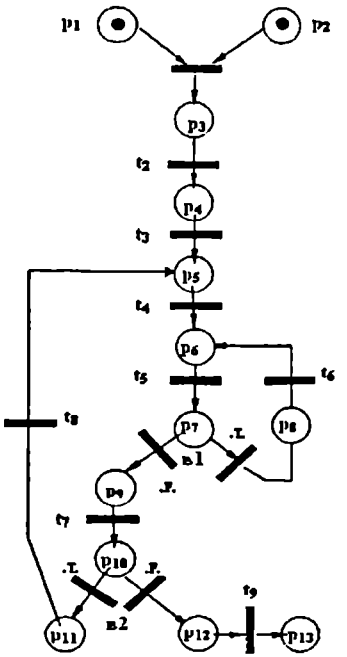
წინამდებარე პარაგრაფში ჩვენს მიერ შემუშავებული და წარმოდგენილია ექსცელის ცხრილების კონვერტაციის ფუნქციის ალგორითმული სქემა მონაცემთა განაწილებული ბაზების მართვის სისტემის, InterBase-ს სტრუქტურაში. ამ მონაცემთა ბაზას ფართოდ იყენებს ბორლანდის ფირმა (Borland C++ Builder და Borland Delphi საბაზო ტექნოლოგიების პროგრამული აპლიკაციებით) და ODBC დრაივერი აქ ვერ გამოიყენება. ამიტომაც ის აქტუალურად მიგვაჩნია.

ამოცანა მდგომარეობს მონაცემთა საცავის განაწილებული რელაციური ბაზის ავტომატიზებულად შესავსებად მონაცემთა პირველადი წყაროების ელექტრონული ცხრილებიდან.

2.9 ნახაზზე ნაჩვენებია აღნიშნული ამოცანის გადაწყვეტის ალგორითმის შესაბამისი პეტრის ქსელის გრაფი, რომელშიც ასახულია შესასრულებელ პროცედურათა მიმდევრობა.

2.10 და 2.11 ნახაზებზე ილუსტრირებულია რეალიზებული C++პროგრამის ფრაგმენტი და შესაბამისი საინტერფეისო ფანჯარა მონაცემთა ბაზის ადმინისტრატორისათვის.

სისტემა რეალიზებულია Borland C++Builder გარემოში.



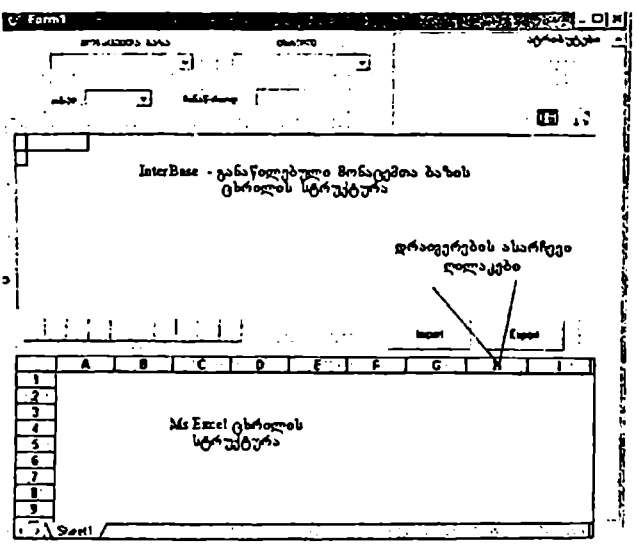
პოზიციები:
p1 – შექმნილია FlBook - Excel ცხრილი;
p2 – შექმნილია InterBase ცხრილი;
p3 – გამოყოფილია Excel ცხრილის დასამუშავებელი არე;
p4 – გახსნილია InterBase ბაზის ალიასი;
p5 – გახსნილია InterBase ბაზის ალიასის ცხრილი;
p6 – InterBase –ში ჩამატებულია ცხრილის ცარიელი სტრიქონი;
p7 – InterBase-ს ცხრილი დამატებული ველით სტრიქონში;
p8 – პოზიცია მარკერით შემდეგი ველისათვის;
p9 – შევსებულია ცხრილის სტრიქონი;
p10 – InterBase-ის ცხრილში ჩაწერილი სტრიქონი;
p11 – პოზიცია მარკერით შემდეგი სტრიქონისთვის;
p12 – InterBase-ს შევსებული ცხრილი;
p13 – მონაცემთა ბაზის დახურული ფაილი.

გადასვლები:
t1 – Excel ცხრილის უბნის გამოყოფა;
t2 – InterBase ბაზაში ალიასის გახსნა;
t3 – InterBase ბაზაში ალიასით ცხრილის გახსნა;
t4 – InerBase ბაზის ცხრილში ცარიელი სტრიქონის ჩამატება;
t5 – Excel-ის გამოყოფილი უბნის კოპირება ბაზის ცხრილში;
t6 – Excel ცხრილის ახალი ველის არჩევა;
t7 – ოპერატორული მეხსიერებიდან ცხრილის სტრიქონის გადაგზავნა დისკზე;
t8 – Excel ცხრილში ახალი სტრიქონის არჩევა;
t9 – მონაცემთა ბაზაში ცხრილების დახურვა;
B1 – არის .FIRST ველი ? B2 - არის .FIRST სტრიქონი ?

ნახ29. პეტრის ქსელის გრაფი დილაკისათვის
ButtonClick: - Import

```
//—— Unit1.cpp —— პროგრამის ტექსის ფრაგმენტი ———
int i,j, Ob_Num, Dan_Num, Rec_Num;
void __fastcall TForm1::FormCreate(TObject *Sender)
  { Session->GetAliasNames(cbAlias->Items); }
void __fastcall TForm1::cbAliasChange(TObject *Sender)
  { Session->GetTableNames(cbAlias->Text,"",true,false,cbTable->Items);
 cbTable->ItemIndex = 0;
 if (cbTable->Text == "")
 { ShowMessage("Table not selected"); return; }
  Table1->Active = false;
  Table1->DatabaseName = cbAlias->Text;
```

```
  Table1->TableName = cbTable->Text;
   Table1->Active = true;
  if(Table1->Active) Table1->GetFieldNames(cbField->Items);
}
//---------------- Button of Import ----------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ AnsiString Info;
  if(Table1->Active)
  {  int t=Table1->FieldCount;
     for(i=3; i<Rec_Num+3; i++)     // str_num
     { Table1->Append();
       for(j = 1; j < t; j++)       // col_num
         {  Label3->Caption=t;
            Table1->FieldValues["OBJNUM"] = Ob_Num;
            Info = Table1->Fields->Fields[j]->FieldName;
            Label1->Caption=j;
           Excel->SetSelection(i,j,i,j);     // choice of Range
           if(Excel->Text != 0)
             { Label2->Caption=Excel->Text;
               Table1->FieldValues[Info] = Excel->Text; }
       } // for j
       Table1->Post
     } // for i
   } // for if( )
  }
```



ნახ.2.11

2.7. OLAP-კუბში მონაცემთა აგრეგაცია გრაფების თეორიის გამოყენებით

ნაშრომში მოცემულია მონაცემთა ფორმალიზების აგრეგაციის მექანიზმი, სადაც ფართოდ გამოიყენება გრაფების თეორია.

მონაცემთა მრავალგანზომილებიანი მოდელის ძირითადი პრინციპებია:

მაჩვენებელიანი – სიდიდე (იგი ჩვეულებრივი რიცხვია), რომელიც გამოიყენება საგნობრივი ანალიზისათვის. მაგ რომელიმე პროდუქტის ყიდვა-გაყიდვის მოცულობა, ან ამ პროცესიდან მიღებული შემოსავალი. ერთი OLAP- კუბი შეიძლება შეიცავდეს ერთ ან რამდენიმე მაჩვენებელს.

განზომილება (dimension) – ერთი ან რამდენიმე სახის ობიექტთა სიმრავლეა, რომელიც ორგანიზებულია იერარქიული სტრუქტურით და უზრუნველყოფს რიცხვითი მაჩვენებლის საინფორმაციო კონტექსტს. გაზომვა ვიზუალურად უნდა აისახებოდეს მრავალგანზომილებიანი კუბის გვერდებზე.

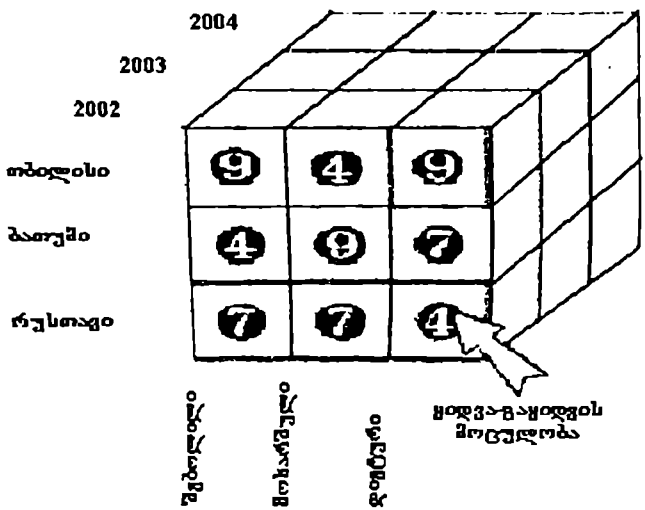
ობიექტები – მათი ერთობლიობა წარმოქმნის განზომილებას, ანუ ეგრეთწოდებულ განზომილებათა წევრებს (members). განზონილებათა წევრები ვიზუალურად აისახება, როგორც ცალკეული წერტილები ან უბნები,რომელიც მოთავსებულია ჰიპერკუბის ღერძებზე. მაგ., დროითი განზომილება: დღე, თვე, კვარტალი, წელი: 26 მაისი 2002 წელი, მე-2 კვარტალი 2002 წელი და 2002 წელი. განზომილებაში მოცემული ობიექტები შეიძლება იყოს სხვადასხვა სახის მაგ: „მწარმოებელი“, „პროდუქტის საფირმო ნიშანი“, „წელი“, „კვარტალი“. ეს ობიექტები ორგანიზებულნი უნდა იყოს იერარქიული სტრუქტურით. ერთი სახის ობიექტებს შეესაბამება, იერარქიის მხოლოდ ერთი დონე.

უჯრედი (cell) – კუბის ატომური სტრუქტურაა, რომელიც შეესაბამება რომელიმე მაჩვენებლის კონკრეტულ მნიშვნელობას. უჯრედები ვიზუალურად მოთავსებულია კუბის შიგნით და მათზე ასახულია მაჩვენებლების შესაბამისი მნიშვნელობები.

განზომილება თამაშობს ინდექსების როლს, იგი გამოიყენება ჰიპერკუბის უჯრედში მაჩვენებლის მნიშვნელობის იდენტიფიკაციისათვის. სხვადასხვა განზომილებით მიიღებული კომბინაციები თამაშობს კოორდინატების როლს, რომელიც განსაზღვრავს კონკრეტული მაჩვენებლის მნიშვნელობას.

ყველა განზომილების შედეგად მიღებული კომბინაციებით შესაძლებელია რამდენიმე უჯრედის მნიშვნელობის განსაზღვრა. ამიტომ უჯრედის ერთმნიშვნელოვანი იდენტიფიკაციისათვის უცილებელია მივუთითოთ განზომილების ყველა კომბინაცია და მაჩვენებელი. ამავე დროს დაცული უნდა იყოს განზომილების იერარქიული დონეები, რათა აგრეგაციისა და მაჩვენებელთა მნიშვნელობის დეტალიზაციისათვის ამას დიდი მნიშვნელობა ენიჭება.

არსებობს იერარქიის შემდეგი სახე: **ბალანსირებული (balanced)** – იერარქია, რომელშიც დონეთა რიცხვი განისაზღვრება მისი სტრუქტურით და მუდმივია ამავე დროს, იერარქიული ხის თითოეული



ნახ.2.12 საშუალოზომილებიანი კუბი

ტოტი შეიცავს ობიექტებს ყოველი დონიდან. მაგ. ძეხვის ნებისმიერ მწარმოებელს შეუძლია აწარმოოს რამდენიმე სახის ძეხვი სხვადასხვა საფირმო ნიშნით (ნახ.2.12).

ამ შემთხვევაში შეგვიძლია ვისაუბროთ სამდონიან იერარქიაზე, რომლის პირველ დონეზე იქნება „მწარმოებელი“, მეორეზე 'სახეობა“ ხოლო მესამეზე 'საფირმო ნიშანი“.

ბალანსირებული ირერარქიის ფორმირებსათვის აუცილებელია არსებობდეს კავშირი „ერთი-მრავალთან“. მისი ყოველი დონე შეიძლება წარმოვადგინოთ, როგორც ცალკეული განზომილება, მაგრამ ეს განზომილება დამოკიდებული იქნება სხვა რომელიმე განზომილებაზე, ამიტომ საჭირო გახდება კუბის გაკვეთა.

დაუბალანსებელი (unbalanced) – იერარქია რომელშიც დონეთა რიცხვი შესაძლებელია შევცვალოთ, ხოლო იერარქიული ხის თითოეული ტოტი შეიძლება შეიცავდეს ობიექტებს არა ყოველი დონიდან, არამედ მხოლოდ პირველიდან. აუცივლებელია აღვნიშნოთ, რომ დაუბალანსებელ იერარქიაში არსებული ყველა ობიექტი არის ერთი სახის. მაგ., იერარქია „ხელმძღვანელი–დამფასოებელი“ ყველა ობიექტი ერთიანდება სახეში: „თანამშრომელი“.

არათანაბარი იერარქია, რომელშიც დონეთა რიცხვი განისაზღვრება მისი სტრუქტურით და მუდმივია. ბალანსირებულისგან განსხვავდება მხოლოდ იმით, რომ იერარქიული ხის რომელიმე ტოტი შეიძლება არ შეიცავდეს რომელიმე დონის ობიექტებს. ასეთი სახის იერარქია შეიცავს ისეთ წევრებს როგორიცაა ე.წ. ლოგიკური „მშობელი“ და იგი უშუალოდ არ მდებარეობს რომელიმე დონეზე. ტიპიურ მაგალითს წარმოადგენს გრაფიკული იერარქია, რომელსაც შემდეგი დონეები აქვს: „ქვეყანა“, „შტატი“ და „ქალაქი“. თუ მონაცემთა ნაკრებში მოცემული გვექნება ქვეყნები, რომელთაც არ გააჩნიათ „შტატი“, მაშინ იგი მოთავსდება დონეებს შორის „ქვეყანა“ და „ქალაქი“.

ჩვენ განვიხილავთ მხოლოდ ბალანსირებულ იერარქიას, რადგან დაუბალანსებელი და არათანაბარი პრაქტიკაში თითქმის არ გამოიყენება.

აგრეგატებს უწოდებენ განსაზღვრული პირობის შესაბამისად მახასიათებლის საწყისი მნიშვნელობათა აგრეგირებას. აქ იგულისხმება მცირე რაოდენობის აგრეგატების მნიშვნელობათა ფორმირების ნებისმიერი პროცედურა, რომელიც ეყრდნობა დიდი რაოდენობის საწყის მნიშვნელობებს.

ნაწილობრივი და სრული აგრეგაცია

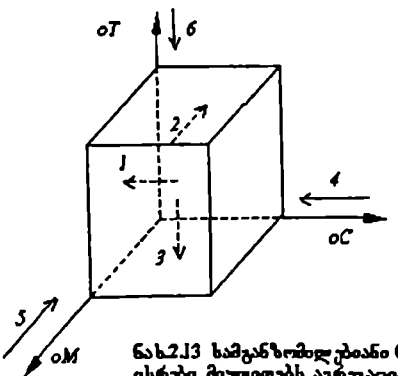
კუბის აგრეგაციის ხარისხი გამოითვლება ფორმულ: $\alpha = \frac{a}{a^*}$ (1)

სადაც a–არის აგრეგირებულ მახასიათებლის მნიშვნელობის რეალური რაოდენობა, a* - კუბის საწყისი მონაცემების აგრეგატული მნიშვნელობის მაქსიმალური რაოდენობა [68].

ფორმულის გამოყვანისას a და a* - სთვის განვიხილოთ მარტივი შემთხვევები, ორ და სამ განზომილებიანი სივრცისთვის, რასაც მოგვიანებით გაერთიანებული სახით ვნახავთ. იგივე შეეხება განზომილებში დონეთა იერარქიას: თავიდან განიხილება მარტივი შემთხვევა (ერთ დონიანი), ხოლო შემდგომ განვიხილავთ რამოდენიმე დონეს, რომელიც ფორმულაში წარმოდგენილი იქნება გაერთიანებული სახით. ასეთი მიდგომა აიოლებს მახასიათებლის აგრეგატული მნიშვნელობის მიღების პროცესს და თვალსაჩინოს ხდის დიდი განზომილების სივრცეების წარმოდგენის მექანიზმს.

<u>მარტივი განზომილების შემთხვევა.</u>

თავიდან განვიხილოთ სამგანზომილებიანი სივრცის მარტივი შემთხვევა. მაგალითისათვის განვიხილოთ საკვები პროდუქტის ყიდვა-გაყიდვის მოცულობა. მრავალგანზომილებიანი კუბის სტრუქტურა მოიცავს შემდეგ ობიექტებს: (ნახ.2.13)



ნახ2.13 სამგანზომილებიანი OLAP- კუბი მარტივი განზომილებით ისრები მიუთითებს აგრეგაციის მიმართულებას

- ერთი მახასიათებელი: გაყიდული პროდუქტის რაოდენობა;
- სამი განზომილება:

1) ყიდვა-გაყიდვის მენეჯერი (ღერძი oM);
2) პროდუქტის სახეობა (ღერძი oC);
3) განზომილება, დროის ერთეული - „თვე“ (ღერძი oT).

შესაბამის განზომილებაზე აღვნიშნოთ C, M,T წევრთა სიმრავლე – „პროდუქტის სახეობა“, „მენეჯერი“ და „თვეები“. ყოველ განზომილებაში წევრთა რაოდენობა $n_c = |C|$, $n_m = |M|$, $n_t = |T|$. ჰილების წევრებს შესაბამისად აღვნიშნავთ m_c, m_m, m_t.

აგრეგირებული მნიშვნელობის მისაღებად კვეთში მენეჯერი და თვე ჩვენ უნდა დავაჯგუფოთ მახასიათებლის საწყისი მნიშვნელობები, ყოველი კომბინაციის ყველა მოდელისათვის (m_t, m_m) მენეჯერი და თვე. 2.13 ნახაზზე ისარი 1 აღნიშნავს აგრეგირების მიმართულებას, ამ თვალსაზრისით აგრეგირების რაოდენობა მნიშვნელობით ტოლია $n_m n_t$. ასეთი ფორმით გამოსახული აგრეგირებისას მახასიათებლის მნიშვნელობები მოთავსდება სიბრტყეზე (oM, oT). განზომილების „მენეჯერი“ ყველა წევრის მახასიათებელთა შეკრებით მივიღებთ ყველა აგრეგატთა რიცხვს (m_c, m_t) კომბინაციისათვის. იგი ტოლია $n_t n_t$ ყველა (m_c, m_m) კომბინაციისათვის აგრეგატთა რიცხვი დროითი განზომილების აგრეგაციისას ტოლია: $n_m n_c$. ახლა კი აუცილებელია განზომილებათა კვეთში განვსაზღვროთ აგრეგატთა რიცხვი. ცხადია, რომ ასეთ აგრეგატთა რაოდენობა ტოლია შესაბამის განზომილებათა წევრთა n_m, n_c და n_t. რაოდენობის. გასათვალისწინებელია, რომ, როდესაც გვინდა სრულყოფილად წარმოვადგინოთ ყიდვა–გაყიდვის მოცულობა, ყველა მოდელით, მენეჯერით და მთელი დროითი პერიოდით, აგრეგატთა მთლიანი რაოდენობა ტოლი იქნება თითოეულ აგრეგატთა ჯამური მნიშვნელობის:

$$a^* = n_m n_c + n_c n_t + n_m n_t + n_c + n_m + n_t + 1$$

აგრეგატთა რაოდენობა, რომელიც ზოგიერთი განზომილების აგრეგირების შედეგად მიიღება, ტოლია:

$$a_{l1..li..lm} = \prod_{i \in I \setminus I^0} n_i \qquad (2)$$

განზომილებათა რიცხვითი წარმოებულის, ყველა შესაძლო აგრეგატთა რაოდენობა კი შეიძლება წარმოვადგინოთ შემდეგი სახით:

$$a^* = \sum_{k=1}^{m} \sum_{j=1}^{C_m^k} \prod_{i=1}^{m-k} n_i \qquad a^* = \sum_{k=1}^{m} a^{m-k}$$

თანამიმდევრობით $l_1,\ldots,l_i,\ldots,l_m$ ტოლია ჯამის $a_{l1\ldots li\ldots lm}$, ასეთი ვარიანტი ტოლი იქნება 2^m-1.

მეორეს მხრივ საწყის მონაცემთა აგრეგაციისას ყველა შესაძლო k განზომილებით, სადაც k^{1,...,m}, ჩვენ მივიღებთ ერთობლივ აგრეგაციათა სიმრავლეს ერთი და იმავე დეტალიზაციის დონით $l = m - k$. ეს ერთობლიობა აღვიშნოთ A^l, რომელსაც გააჩნია a^lაგრეგატთა რაოდენობა. აგრეგაციის სიმრავლეთა $A_{l1\ldots li\ldots lm}$ â A^{m-k} რაოდენობის გამოსათვლელად აუცილებელია ინდექსში გამოვთვალოთ ნულთან m პოზიციით მიახლოების ვარიანტთა რაოდენობა. ეს შემთხვევა ასე გამოისახებაC^k_mე.ი. სადაც $|I_0| = k$. ეს ფორმულა გათვლილია აგრეგატთა მთელი რიცხვისათვის, ფორმულა (2)-ის გათვალიწინებით იგი შეიძლება წარმოვადგინოთ შემდეგი სახით: სადაც,

$$\sum_{j=1}^{C_m^k}\prod_{i=1}^{m-k} n_i \quad (3)$$

ჯამი, წარმოადგენს ყველა შესაძლო განზომილებათა რაოდენობას.

საბოლოოდ საწყის მონაცემთა გამორიცხვით მივიღებთ:

$$a^* = \prod_{i=1}^{m}(n_i + 1) - \prod_{i=1}^{m} n_i \quad (4)$$

იერარქიული განზომილების შემთხვევა

განვიხილოთ შემთხვევა როდესაც $i-$ური განზომილება მდებარეობს იერარქიის l_i^* დონეზე.

წევრთა რაოდენობა i-ურ განზომილების j - n_{ij}, ასე რომ გვექნება წევრები n_{ij1}, n_{ij2}, სადაც j1 <j2. წევრთა საერთო $i-$ური

განზომილების რაოდენობა ტოლია: $n_i = \sum_{j=1}^{i^*} n_{ij}$.

განზომილებების მოწესრიგებულად დალაგების შემდეგ, მივიღებთ მიმდევრობას: $D_1,...,D_i,...,D_m$, სადაც i არის განზომილების რიგითი ნომერი. ასეთი ნომრით მიღებული სიმრავლე აღვნიშნოთ I . განვსაზღვროთ რიცხვთ მიმდევრობა $l_1,...,l_i,...,l_m$, სადაც $l_i = 0...l_i^*$, მიღებული მიმდევრობა შეგვიძლია ჩავთვალოთ, როგორც აგრეგაციის მდგომარეობა,მაშინ $A_{l1...\,li\,...lm}$ არის აგრეგატთა სიმრავლე,იგი მიიღება ყოველი მ განზომილების li– იური დონის საწყის მონაცემთა $a_{l1\,...\,li\,...lm} = |A_{l1...\,li\,...lm}|$. აგრეგირებით. მარტივი განზომილების შემთხვევაში, ქვესიმრავლეთა რიგითი ნომერი აღვნიშნოთ $I^o = \{ i_k \mid i_k I_i \,\&\, l_{ik} = 0 \}$, $k=1...p$, $p<=m$.

იერარქიის ნულოვანი დონე $l_i = 0$ შეიძლება წარმოვიდგინოთ, როგორც აბსტრაქტული დონის ძირი, რომელიც ყოველთვის შეიცავს ერთეულ რიცხვებს ($n_{i0}=1$), როგორც წესი მას საგნობრივ სფეროში, არ გააჩნია რეალური ობიექტი. ასეთ შემთხვევაში i განზომილების ყველა წევრის აგრეგაცია შეიძლება წარმოვადგინოთ, როგორც აგრეგაცია, რომელიც მოხდა i განზომილების საწყის დონეზე (ძირზე).

როგორც მარტივი განზომილების შემთხვევაში, აქაც მას ვუწოდებთ დეტალიზაციის დონეს, რომლის აგრეგატთა სიმრავლე ტოლია: $A_{l1...\,li\,...lm}$,

ხოლო რიცხვითი დონე: $l = l_1+...+l_i+...+l_m$

გახვიხილოთ ორგანზომილებიანი შემთხვევა შემდეგი სტრუქტურით:

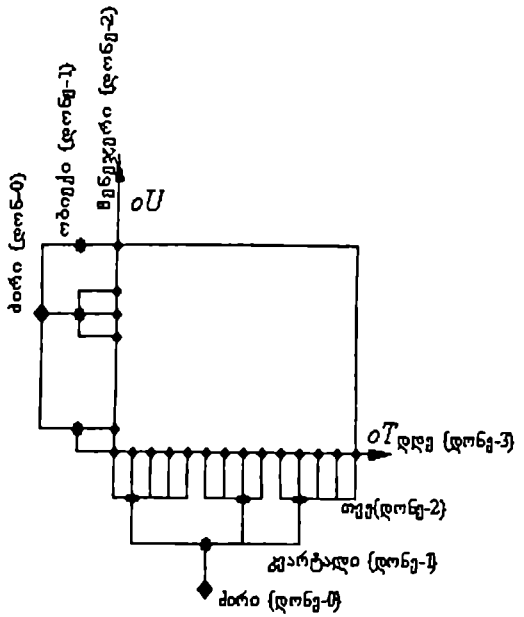
მახასიათებელი: გაყიდული პროდუქტის რაოდენობა,

განზომილება: დროითი განზომილება (ღერძი oT): კვარტალი – თვე – დღე, ფუნქციონალური ერთეული(ღერძი oU) : სავაჭრო ობიექტი – მენეჯერი.

როდესაც დროითი განზომილების (T) წევრები გადააჭარბებს ფუნქციონალური ერთეულის (U) წევრთა რაოდენობას, მაშინ

- $T = D_1$, $U = D_2$;
- $l_1^* = 3$, $l_2^* = 2$;
- T დაU განზომილებათა საერთო რიცხვი, შესაბამისად ტოლია n_1 და n_2;

- კვარტალის, თვეების და დღეთა რაოდენობა დროით ღერძზე შესაბამისად ტოლია n_{11}, n_{12} და n_{13};
- სავაჭრო ობიექტების და მენეჯერის საერთო რაოდენობა შესაბამისად ტოლია n_{11} და n_{22};
- საწყის მონაცემთა რაოდენობა ტოლია n_{13} n_{22}.



ნახ2.14 იერარქიული განზომილების ორგანზომილებიანი

აგრეგატთა რაოდენობა მარტივად შეიძლება გამოვსახოთ:

კვეთში მენეჯერი და თვეები $a_{22} = n_{12}\, n_{22}$,

- კვეთში დღეები და სავაჭრო ობიექტი $a_{31} = n_{13}\, n_{21}$,
- კვეთში მხოლოდ დღეები $a_{30} = n_{13}$,
- კვეთში თვეები და სავაჭრო ობიექტი $a_{21} = n_{12}\, n_{21}$,
- კვეთში მხოლოდ თვეები $a_{20} = n_{12}$,
- კვეთში კვარტალი და სავაჭრო ობიექტი $a_{11} = n_{11}\, n_{21}$,

- კვეთ'ში მხოლოდ მენეჯერი $a_{02}= n_{22}$,
- კვეთ'ში მხოლოდ სავაჭრო ობიექტი $a_{01}= n_{21}$
- კვეთში მხოლოდ კვარტალი $a_{10}= n_{11}$,
- სრული აგრეგატი $a_{00}= 1$.

ამ თვალსაზრისით

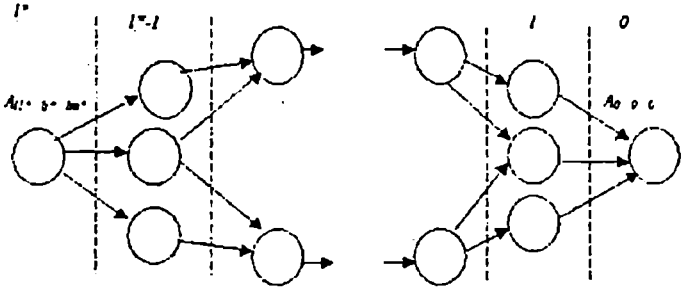
$a^* = n_{12}\, n_{22} + n_{13}\, n_{21} + n_{13} + n_{12}\, n_{21} + n_{11}\, n_{22} + n_{12} + n_{11}\, n_{21} + n_{22}, + n_{21} + n_{11} + 1$.განსაზღვრით $A_{l1\,..\,li\,..lm-}$ იდან ,ფორმულა (2)–ის გათვალისწინებით ზოგადი შემთხვევისათვის მივიღებთ:

აგრეგატთა ფორმირების პროცედურები

აგრაგატთა სიმრავლე შეიძლება წარმოვადგინოთ ქსელური გრაფის მწვერვალების დახმარებით. ასეთი გრაფის საწყისი მწვერვალი წარმოადგენს მონაცემთა საწყის სიმრავლეს, ხოლო საბოლოო – სრულ აგრეგატთა მნიშვნელობას (ნახ.2.15)

$$a^* = \sum_{l^*} \prod_{i=1}^{m} n_{i,li}$$

ფორმულაში ყველა შესაკრები წარმოადგენს განზომილებათა უნიკალურ წევრს



ნახ.2.15 აგრეგატთა წარმოდგენა ქსელური გრაფით

i –ურ განზომილებაში აგრეგირების ოპერაციის ჩატარებისას აგრეგატთა ნებისმიერ სიმრავლიდან $A_{l1\ \ li\,..lm}$ ($l_i \in \{0,\dots l_{i^*-1}\}$) შეიძლება გადავიდეთ სხვა $A_{l1\,..li+1\,..lm}$ სიმრავლეში. ამისათვის აუცილებელია ვაწარმოოთ შესაბამისი გამოთვლითი დანახარჯები, რომელიც გადაიტანება გრაფის გვერდებზე.

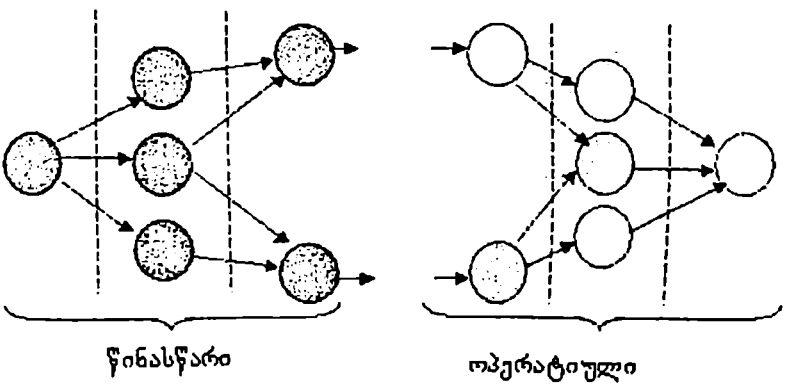
აგრეგატთა ოპერატიული ფორმირების პროცედურები.

აგრეგატთა ოპერატიული ფორმირების პროცედურების ქვეშ იგულისხმება პროცედურები რომელიც მიიღება აგრეგატთა

ერთობლიობით, მომხმარებელთა მოთხოვნების დასაკმაყოფილებლად. აგრეგატთა ერთობლიობით მიღებული შედეგის გამოთვლისას გამოყენებული უნდა იქნას წინასწარ ფორმირებული მახასიათებელთა აგრეგირების მნიშვნელობა, რის შედეგადაც აგრეგატთა ოპერატიულ ფორმირებაზე გაწეული დანახარჯები იქნება 0-ის ტოლი.

ვთქვათ წინასწარ ფორმირებულ აგრეგატთა შედეგში მივიღეთ ყველა შესაძლო სიმრავლე მინიმალური l დეტალიზაციის დონით. დავუშვათ შედეგის მისაღებად აუცილებელია გამოვიყენოთ აგრეგატთა $A_{l'1, \ldots, l'm}$ სიმრავლე რომლის დეტალიზაციის დონე l', l'< l. აუცილებელია ერთ რომელიმე ძირითად l –ური დონის $A_{l1, \ldots, lm}$ სიმრავლიდან მივიღოთ შედეგობრივი სიმრავლე, რისთვისაც საჭიროა:

აგრეგატთა საწყის სიმრავლემ უნდა უზრუნველყოს საბოლოო სახით მიღებულ სიმრავლეზე გაწეული ხარჯების მინიმალური რაოდენობა, მოკლედ რომ ვთქვათ აუცილებელია განვსაზღვროთ ქსელურ გრაფში უმოკლესი გზა.(ნახ.2.16)



ნახ.2.16 აგრეგატთა წინასწარი და ოპერატიული ფორმირება ქსელურ მოდელში

ოპერატიული ფორმირების პროცედურა უზრუნველყოფს მომხმარებელთა მოთხოვნების დაკმაყოფილებას მინიმალურ დროში მინიმალური დანახარჯებით.

III თავი
განაწილებული სისტემის რესურსების მართვის პროცესის მოდელირებისა და ანალიზის ალგორითმები

განაწილებული სისტემების ქვეშ ჩვენ ვიხილავთ ელექტრონული ბიზნესისა და კომერციის ობიექტებს, მაგალითად დიდ სავაჭრო ცენტრებს (სუპრტმარკეტების კომპლექსს), რომელთაც გააჩნია განაწილებული მონაცემთა საცავი და პროგრამული პაკეტები, კლიენტ-სერვერ არქიტექტურის კონფიგურაცია ჰომოგენური (ან ჰეტეროგენული) კომპიუტერული რესურსებით.

სისტემის მიზანია მომხმარებელთა მოთხოვნების (უშუალოდ მარკეტში ან ინტერნეტიდან) მაქსიმალური დაკმაყოფილება, რაც უზრუნველყოფს ფირმის მოგების სტაბილურობას. მასობრივი მომსახურების თეორიის მიხედვით, პირობითად „მომსახურე ორგანოს“ ქვეშ შეიძლება ვიგულისხმოთ სუპერმარკეტის თანამშრომელი (მომხმარებელთან უშუალო კონტაქტი) ან კომპიუტერული ქსელის სერვერი (ელექტრონული კონტაქტი). ორივე შემთხვევაში პროცესი მსგავსი მოდელით აიგება (მოთხოვნების ნაკადი, მომსახურების დრო, რიგების სიგრძე და ა.შ.), ოღონდაც თვით ამ მაჩვენებელთა მნიშვნელობები იქნება განსხვავებული.

ასეთი მულტიპროცესორული ქსელური კონფიგურაციის სისტემების დაპროექტებისას საჭიროა მრავალი მახასიათებლის გათვალისწინება, რომელთა ოპტიმალური მნიშვნელობების შერჩევა ძალზე მნიშვნელოვანია და ამვე დროს რთულიც. ამ სიდიდეთა ოპტიმიზაცია არა მარტო გაზრდის კომპიუტერული ქსელის წარმადობას, არამედ შეამცირებს მის შესაქმნელად საჭირო ხარჯებსაც. მნიშვნელოვანია გავითვალისწინოთ ისეთი მომენტები, როგორიცაა სიმძლავრეების, საერთო რესურსების და ა.შ. ოპტიმალური განაწილება.

კომპიუტერულ ქსელებში მიმდინარე მოვლენების (დინამიკური პროცესების) მოდელირებისათვის მოსახერხებელია პეტრის ქსელების გამოყენება, რაოდენობრივი მახასიათებლების

ანალიზისათვის კი - მასობრივი მომსახურების სისტემების თეორია [65,66,67].

ჩვენი მიზანია შევიმუშავოთ ალგორითმული სქემები და შესაბამისი პროგრამული პაკეტები, რომელთა დანიშნულებაცაა კომპიუტერული ქსელის სიმძლავრეების ანალიზი და მათი ოპტიმალური განაწილება.

3.1. განაწილებული სისტემის რესურსების მართვის პროცესის მასობრივი მომსახურების მოდელი სტაციონარული რეჟიმისათვის

განვიხილოთ კომპიუტერული ქსელი, სადაც არის რამდენიმე მომხმარებელი და რამდენიმე სერვერი (მომსახურე). დავუშვათ, რომ სერვერთაგან ერთ-ერთი ასრულებს გამანაწილებლის ფუნქციას, ე.ი. იღებს მომხმარებლისაგან მოთხოვნას და უგზავნის მას მომსახურებისათვის იმ სერვერს, რომელიც თავისუფალია. თუ ყველა სერვერი დაკავებულია, მოთხოვნა დგება რიგში და ელოდება ერთ-ერთი მათგანის განთავისუფლებას.

სერვერი, მიიღებს რა მოთხოვნას გამანაწილებელი სერვერიდან, ემსახურება მას და უბრუნებს ისევ გამანაწილებელ სერვერს, რომელიც, თავის მხრივ პასუხს უბრუნებს მომხმარებელს.

უნდა ვიგულისხმოთ, რომ მოთხოვნები მომხმარებლებისგან მოდის უწყვეტად, გარკვეული სიხშირით. თითოეული სერვერი ერთეული მოთხოვნის მომსახურებას ანდომებს გარკვეულ დროს. იმ შემთხვევაში როდესაც, მოთხოვნათა ფორმირების სიხშირე დიდია, გამანაწილებელ სერვერთან წარმოიქმნება რიგი. თუკი მოთხოვნათა ფორმირების სიხშირე ძალზე დიდია ქსელი შეიძლება გადაიტვირთოს და ვეღარ შეძლოს ფუნქციონირება.

ჩვენი მიზანია ქსელის არსებული პარამეტრების მეშვეობით დავადგინოთ მისი მუშაობის კრიტიკული წერტილი, შევარჩიოთ ისეთი მახასიათებლები, რომლებიც უზრუნველყოფს მის ნორმალურ ფუნქციონირებას და შევქმნათ პროგრამული პროდუქტი, რომელიც ყოველივე ამას განახორციელებს. მასობრივი

მომსახურების თეორიის თვალსაზრისით ზემოთ აღწერილი სისტემა არის M/M/m ტიპის [51].

განვიხილოთ მახასიათებლები და მათ შორის კავშირები, რომლებიც გააჩნია ქსელს. აქვე უნდა აღვნიშნოთ რომ ქსელის ფუნქციონირებას განვიხილავთ სტაციონარულ რეჟიმში. ამ შემთხვევაში, როგორც ცნობილია, გარკვეულ იდეალიზაციასთან გვაქვს საქმე. რეალურად დროის ყოველ t მომენტში სისტემაში არსებობს მოთხოვნათა რაღაც k რაოდენობა. ალბათობა იმისა, რომ დროის მოცემულ t მომენტში სისტემაში იმყოფება კ მოთხოვნა, აღვნიშნოთ $P_k(t)$-თი. ჩვენ უნდა ვიგულისხმოთ, რომ t-ს ზრდასთან ერთად ალბათობა $P_k(t)$ თანდათან მუდმივი ხდება. ამ შემთხვავაში $P_k(t)$-ს ნაცვლად შეიძლება გამოვიყენოთ P_k, რომელიც უკვე აღარ არის დროის ფუნქცია. ეს დაშვება არ გულისხმობს იმას, რომ სისტემა არ გადადის ერთი მდგომარეობიდან მეორეში, რა თქმა უნდა დროის მიხედვით იცვლება ქსელში არსებული მოთხოვნების რაოდენობა, მაგრამ ალბათობა იმისა, რომ სისტემაში საკმარისად დიდი დროის გასვლის შემდეგ იმყოფება k მოთხოვნა [52].

პროგრამული პაკეტში საანგარიშო ფუნქციების დასაპროგრამებლად გამოვიყენოთ აღნიშნული კლასიკური მოდელები. ამგვარად, სერვერების რაოდენობით, შემოსულ მოთხვნათა ინტენსივობით და დროით, რომელსაც ანდომებს სერვერი თითოეული მოთხოვნის მომსახურებას, შეგვეძლება დავადგინოთ ქსელის სხვადასხვა მახასიათებელი.

აღვნიშნოთ მოთხოვნათა მოსვლის ინტენსივობა λ-თი, ხოლო თითოეული მოთხოვნის მომსახურების დროს Ts-ით. ამ შემთხვევაში ერგოდიულობის პირობა არის: λ*Ts<1 .

ქსელს გააჩნია შემდეგი მახასიათებლები:

1. მოძრაობის ინტენსივობა: u=λ*Ts.
2. სერვერის დატვირთვა: ρ =u/m.

იმისათვის, რომ სისტემა იყოს სტაბილური, სერვერს უნდა შეეძლოს თავი გაართვას მოთხოვნათა მოსვლის საშუალო ინტენსივობას, ეს კი ნიშნავს, რომ მოძრაობის ინტენსივობა უნდა იყოს სერვერთა რაოდენობაზე ნაკლები, ან რაც იგივეა, სერვერის დატვირთვა უნდა იყოს ერთზე ნაკლები, ე.ი. $u<m$ ან $\rho<1$. M/M/m სახის სისტემების კვლევისას მნიშვნელოვანი ადგილი უკავია ერლანგის ფუნქციას [52,53]. ეს ფუნქცია განსაზღვრავს იმის ალბათობას, რომ ყველა სერვერი დაკავებულია, და იმავდროულად იმის ალბათობასაც, რომ მოსულ მოთხოვნას მოცდა მოუწევს. ერლანგის ფუნქციისთვის გამოვიყენებთ გამოსახულებას:

$$Ec(m,u) = (u^m / m!) / (u^m / m! + (1-\rho)\sum_{k=0}^{m-1}(u^m / m!))$$

მომხმარებლისთვის დიდი მნიშვნელობა აქვს მოთხოვნის რიგში დგომის (მოცდის) საშუალო დროს, იგი გამოითვლება ფორმულით: $Tw = \frac{Ec(m,u)Ts}{m(1-\rho)}$

აუცილებელია განვსაზღვროთ მოთხოვნის სისტემაში ყოფნის საშუალო დრო: Tq=Tw+Ts.

ალბათობა იმისა, რომ მოთხოვნის სისტემაში ყოფნის დრო ნაკლებია t-ზე დამოკიდებულია u=m-1 თუ არა. თუ ეს პირობა სრულდება, მაშინ ადგილი აქვს შემდეგ ტოლობას:

$$P(\text{სისტემაში ყოფნის დრო}<t) = 1-(1+\frac{t}{Ts}Ec(m,u))e^{\frac{-t}{Ts}}$$

წინააღმდეგ შემთხვევაში: P(სისტემაში ყოფნის დრო<t)=

$$1+\frac{B+Ec(m,u)}{B}e^{\frac{-t}{Ts}}+\frac{Ec(m,u)}{B}e^{-(m-u)\frac{t}{Ts}}$$

სადაც B=m-1-u.

დროის ყოველ მომენტში ქსელში იარსებებს მოთხოვნათა გარკვეული რაოდენობა. რაც ნაკლები მოთხოვნაა ქსელში, მით უკეთ ფუნქციონირებს იგი. ალბათობა იმისა, რომ ქსელში არის k

მოთხოვნა არის P_k სადაც

$$P_k = \frac{u}{k!} P_0 , \qquad \text{როცა} \quad k<=m$$

და $$P_k = \frac{u^k}{m! m^{k-m}} P_0 \quad \text{როცა} \quad k>=m$$

P_0 არის ალბათობა იმისა, რომ ქსელში საერთოდ არაა მოთხოვნა.

ეს რაც შეეხებოდა ალბათობებს, თვით სისტემაში არსებულ მოთხოვნათა რაოდენობა კი არის Lq, სადაც

$$Lq = u + \frac{pEc(m,u)}{1-\rho}$$

თუკი ქსელში არის m ან m-ზე ნაკლები მოთხოვნა, მაშინ იმ მოთხოვნების რაოდენობა, რომლებიც რიგში დგას 0-ის ტოლია. ხოლო თუ ვიცით, რომ x მოთხოვნა რიგში დგას, მაშინ მთლიანად სისტემაში იქნება x+m მოთხოვნა. ასე, რომ გვაქვს შემდეგი მახასიათებლები-ალბათობა იმისა, რომ არცერთი მოთხოვნა არ იცდის:

$$\text{P(არცერთი მოთხოვნა არ იცდის)} = \sum_{k=0}^{m} P_k k$$

ალბათობა იმისა, რომ x მოთხოვნა დგას რიგში:

$\text{P(x მოთხოვნა იცდის)} = P_{x+m}$ სადაც x>m

მომლოდინე მოთხოვნათა საშუალო რიცხვი:

$$Lw = \frac{pEc(m,u)}{1-\rho}$$

ობიექტ-ორიენტირებული დაპროგრამების საფუძველზე, კომპიუტერული ქსელისათვის მიზანშეწონილია შევქმნათ კლასი, რომლის დახურული პარამეტრები იქნება მოთხოვნათა მოსვლის სიხშირე, მომსახურების დრო, სერვერების რაოდენობა და ა.შ. ფუნქციათა წევრების სახით კი რეალიზებული იქნება ყველა ზემოთ ჩამოთვლილი მახასითებლების გამოთვლა, შესაბამისი ფორმულების გამოყენებით [53].

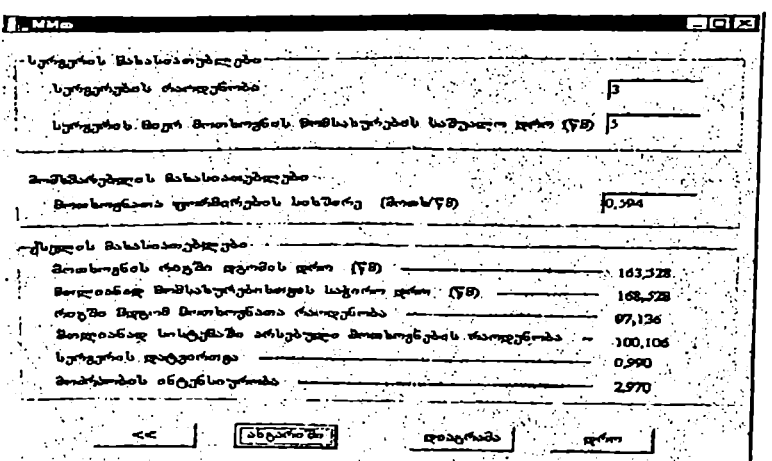
ცალკე კლასებად იქნება რეალიზებული მომხმარებლის ინტერფეისი და პარამეტრთა შორის დამოკიდებულებათა გრაფიკების აგებისა და ვიზუალიზაციის ფუნქციები.

ზემოთ განხილული სიდიდეები სრულად ახასიათებს კომპიუტერული ქსელის მუშაობის სტაციონალურ რეჟიმს. ჩვენს მიერ შექმნილი პროგრამული საშუალება სწორედ ამ სიდიდეებს და ფორმულებს იყენებს ქსელის პარამეტრების ანალიზისათვის და მათი ოპტიმალური მნიშვნელობის შერჩევისათვის.

იგი, იღებს რა ინფორმაციას ქსელში მოთხოვნების მოსვლის სიხშირეზე, სერვერთა რაოდენობასა და თითოეული მოთხოვნის მომსახურების დროზე, ანგარიშობს ისეთ პარამეტრებს როგორიცაა მოთხოვნის რიგში დგომის დრო, ბუფერში მოთავსებული მომლოდინე მოთხოვნათა რაოდენობა, სერვერის დატვირთვა და მოძრაობის ინტენსივობა, სხვადასხვა ალბათობები და ა.შ.

გარდა ამისა, გამოითვლის მოცემულ პირობებში ოპტიმალური მუშაობისათვის საჭირო პარამეტრებს და აგებს მათ შორის დამოკიდებულებათა გრაფიკებს.

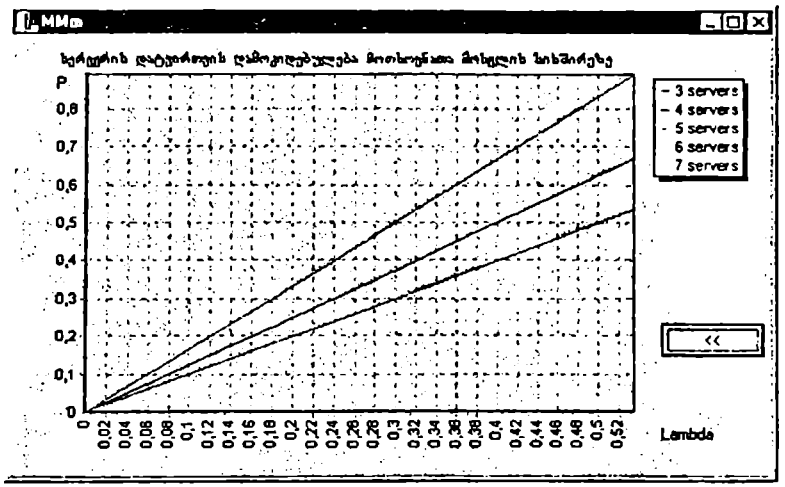
3.1 ნახაზზე მოცემულია Borland_C++Builder ინსტრუმენტით აგებული მომხმარებლის ინტერფეისი, რომელშიც მუშაობა წარიმართება ვიზუალური და ტრადიციული დაპროგრამების კომპონენტების რევერსული ტექნოლოგიით [54,55].



ნახ.3.1

როგორც ნახაზიდან ჩანს, მომხმარებელს შეუძლია შეიტანო, (და ცვალოს) სამი პარამეტრის მნიშვნელობა: სერვერების რაოდენობა, მომსახურების საშუალო დრო და მოთხოვნათა რაოდენობის ინტენსიურობა. ღილაკით „ანგარიში“ სისტემა გაიანგარიშებს ქსელის ძირითად მახასიათებლებს, კერძოდ: სერვერის დატვირთვა, მოძრაობის ინტენსიურობა, მოთხოვნის რიგში დგომის დრო, მთლიანად მომსახურებისთვის საჭირო დრო, რიგში მდგომ მოთხოვნათა რაოდენობა, სისტემაში მყოფ მოთხოვნათა საერთო, რაოდენობა.

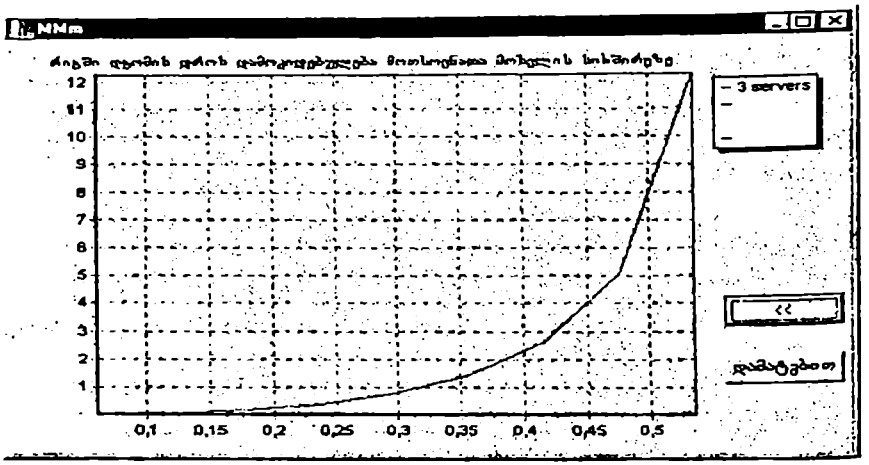
ღილაკით „დიაგრამა“ გამოიტანება გაანგარიშების შედეგად მიღებული გრაფიკები. კერძოდ, 3.2 ნახაზზე მოცემულია პროგრამულად მიღებული დიაგრამა სერვერის დატვირთვის დამოკიდებულებისა მოთხოვნათა მოსვლის სიხშირეზე სერვერების სხვადასხვა რაოდენობისათვის (მაგ., 3-:-7).



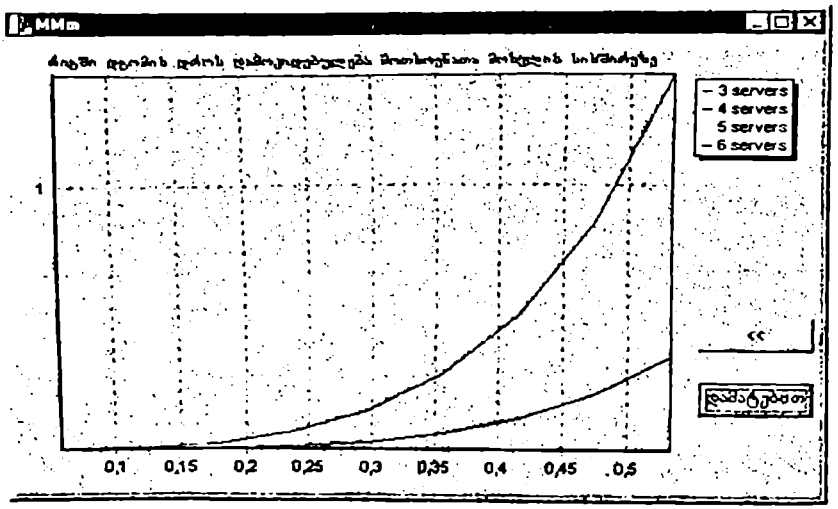
ნახ.3.2.

3.3 და 3.4 ნახაზებზე კი მოცემულია დიაგრამები მოთხოვნათა რიგში დგომის დროის დამოკიდებულებისა მოთხოვნათა მოსვლის სიხშირეზე სერვერების სხვადასხვა რაოდენობის (მაგ., 3-:-6) შემთხვევაში. ბოლო დიაგრამებიდან კარგად ჩანს, თუ როგორ

იკლებს მოთხოვნათა რიგში დგომიდ დრო მომსახურე არხების მომატებით.



ნახ.3.3



ნახ.3.4

3.2 განაწილებული სისტემის რესურსების მართვის პროცესის კვლევა პეტრის ქსელის გრაფით დინამიკურ რეჟიმში

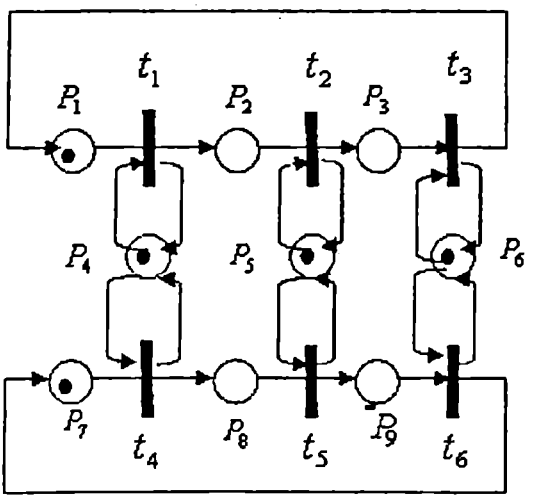
ახლა გავაანალიზოთ კომპიუტერული ქსელის მოქმედება დინამიკურ რეჟიმში. ამ შემთხვევაში სერვერების მიერ კლიენტთა მოთხოვნების დაკმაყოფილების პროცესი შეიძლება მოდელირებულ იქნას ტრანზიტული დროითი პეტრის ქსელის (Timed Transition Petri Net) საშუალებით [70].

პეტრის ქსელის დროითი გაფართოება ჩვენს შემთხვევაში იქნება ალბათური (სტოქასტიკური). ამგვარად, ასეთი ქსელის ანალიზი შესაძლებელია მარკოვის მეთოდების გამოყენებით, რომელშიც დრო ექსპონენციალურადაა განაწილებული [66].

სტოქასტური პეტრის ქსელის მისაღებად საჭიროა „პოზიცია-გადასასვლელების ქსელს“ დაემატოს გადასასვლელთა გაშვების (დაყოვნების, მოლოდინის) დროთა მომენტები:

მაგალითად, μ_1, μ_2 . . . . . μ ı).

განვიხილოთ კერძო მაგალითი კომპიუტერების ქსელისათვის, ორი სერვერითა და სამი კლიენტით. 3.5 ნახაზზე მოცემულია შესაბამისი სტოქასტური პეტრის ქსელის საწყისი მდგომარეობა.



ნახ.3.5.

მარკერის არსებობა S1(p1,p2,p3) და S2(p7,p8,p9) სერვერებში მიუთითებს მათ მზადყოფნაზე კლიენტების მომსახურებისათვის.

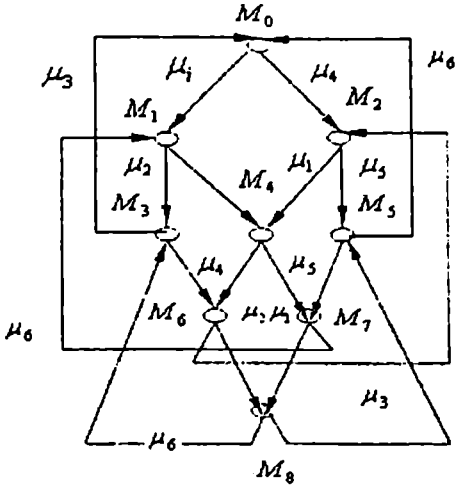
დავუშვათ, რომ C(p4,p5,p6) კლიენტის პოზიციებში მარკერები მუდმივადაა, ე.ი. მოთხოვნები არსებობს და ისინი ელოდება სერვერის მომსახურებას. როგორც აღვნიშნეთ, Tj გადასასვლელის გახსნის დროა (ანუ მომსახურების დაყოვნების დრო). Tj-ური გადასასვლელის გახსნის საშუალო დრო იქნება $1/\mu$, სადაც μ გადასასვლელის გახსნის ინტენსივობაა. სისტემის მდგომარეობა, ანუ მარკერების სიმრავლე შეიძლება ასე ჩაიწეროს:

p პოზიციები და t გადასასვლელები.

M1—100111001
M2—010111001
M3—100111100
M4—010111001
M5—100111010
M6—001111100
M7—010111010
M8—001111010

სადაც Mi, 0<=i<=K მდგომარეობებია (მარკირებები); Tj, 1<=j<=L გადასასვლელები; λ_j, 1<=j<=L – დაყოვნების დრო გადასასვლელის გასაღებად. ჩვენს შემთხვევაში m=3 და n=2, ამიტომ კომბინაცია იქნება $m^n=9$.

გადასასვლელების გახსნის ორგანიზება, როცა სისტემა ყველა მდგომარეობას გადის ნაჩვენებია 3.6 ნახაზზე, რომელსაც პეტრის ქსელის მიღწევადობის გრაფს უწოდებენ.



ნახ.3.6.

ასეთი სტოქასტური პეტრის ქსელის რაოდენობრივი ანალიზი შეიძლება განხორციელდეს შესაბამისი მარკოვის პროცესების ანალიზით. განვიხილოთ მარკოვის ჯაჭვის მაგალითი, რისთვისაც ამ ნახაზზე გრაფის რკალებზე მივამაგროთ გადასასვლელების გაშვების μ კოეფიციენტები.

ჩვენთვის საინტერესოა დავადგინოთ სისტემის თითოეულ მდგომარეობაში გადასვლის ალბათობები, ამისათვის საჭიროა შევადგინოთ კოლმოგოროვის განტოლებათა სისტემა:

$$P_5 * \mu_6 + P_3 * \mu_3 - P_0 * (\mu_1 + \mu_4) = 0$$
$$P_7 * \mu_6 + P_0 * \mu_1 - P_1 * (\mu_2 + \mu_4) = 0$$
$$P_0 * \mu_4 + P_6 * \mu_3 - P_2 * (\mu_1 + \mu_5) = 0$$
$$P_6 * \mu_3 + P_0 * \mu_4 - P_3 * (\mu_1 + \mu_5) = 0$$
$$P_1 * \mu_4 + P_2 * \mu_1 - P_4 * (\mu_2 + \mu_5) = 0$$
$$P_2 * \mu_5 + P_8 * \mu_3 - P_5 * (\mu_1 + \mu_6) = 0$$
$$P_3 * \mu_4 + P_4 * \mu_2 - P_6 * (\mu_3 + \mu_5) = 0$$
$$P_4 * \mu_5 + P_5 * \mu_1 - P_7 * (\mu_2 + \mu_6) = 0$$
$$P_6 * \mu_5 + P_7 * \mu_2 - P_8 * (\mu_3 + \mu_6) = 0$$
$$P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 = 1$$

მაგალითად თუ დავუშვებთ, რომ:

$$\mu_1 = 3, \mu_2 = 5, \mu_3 = 2, \mu_4 = 3, \mu_5 = 1, \mu_6 = 7;$$

მაშინ ალბათობათა მნიშვნელობები, შესაბამისად იქნება:

$$P_0 = 0,11; P_1 = 0,05; P_2 = 0,07; P_3 = 0,26; P_4 = 0,06;$$

$$P_5 = 1; P_6 = 0,37; P_7 = 0,01; P_8 = 0,05.$$

უნდა აღინიშნოს, რომ კომპიუტერული ქსელისათვის ერთი მდგმარეობიდან მეორეში გადასვლის ინტენსივობა μ არის სერვერის მიერ შესაბამისი კლიენტის მოთხოვნის მომსახურებისათვის საჭირო დროის შებრუნებული სიდიდე, ე.ი. $1/Ts$.

ამ განტოლებათა სისტემის გაუსის მეთოდით ამოხსნით მივიღებთ სისტემის ერთი მდგომარეობიდან მეორეში გადასვლის ალბათობებს $P_0, P_1, P_2, \ldots P_8$.

3.3 ბიზნეს პროცესების მოდელირება მასობრივი მომსახურების ჩაკეტილი სისტემებით

თავისუფალ საბაზრო ეკონომიკის პირობებში ბიზნეს-პროცესების მართვის ერთ-ერთი ძირითადი მეთოდია მათემატიკური მოდელირება. ფაქტობრივად სისტემური ანალიზისა და ოპერაციათა გამოკვლევის მათემატიკური მეთოდები ხდება ეკონომიკურ გამოკვლევათა განუყოფელი ნაწილი.

ამასთან, ეკონომიკურ-მათემატიკური მოდელირების ერთ-ერთი მნიშვნელოვანი ნაწილია მასობრივი მომსახურების თეორია სახელდობრ, მასობრივი მომსახურების ჩაკეტილი სისტემები. სწორედ მათი მეშვეობით აღიწერება მრავალი ეკონომიკური სისტემა რომელთა ფუნქციობაში გადამწყვეტ როლს ასრულებს შემთხვევითი (სტოქასტური) ფაქტორები.

მეორეს მხრივ, ასეთი სისტემების მართვის ერთ-ერთი არსებითი ატრიბუტია მათი ფუნქციობის ეფექტიანობის ეკონომიკური მაჩვენებლის (კრიტერიუმის) შერჩევა, რაც თავის მხრივ ოპტიმალური სამოქმედო გადაწყვეტილებათა მიღების საფუძველია. ეს იმას ნიშნავს, რომ ალტერნატიული სამოქმედო ვარიანტებიდან

აირჩევა ის, რომელსაც შეესაბამება ეფექტიანობის ეკონომიკური მაჩვენებლის მეტი ან ნაკლები მნიშვნელობა.

ბუნებრივია, რომ ეფექტიანობის კრიტერიუმში გათვალისწინებული უნდა იყოს სისტემაში არსებული დანახარჯები, შემოსავლები, მოგება, საჯარიმო სანქციებით გამოწვეული ზარალი და სხვა ეკონომიკური ასპექტები, რომლებიც ობიექტურად ასახავენ სისტემის საბაზრო მდგომარეობას.

აღნიშნულ გარემოებათა გონივრული შეხამება წარმატებული ბიზნესის ძირითადი საწინდარია.

„ჩვეულებრივ ადამიანს ჰგონა, რომ კარგად იცის თავისი ბიზნესის სპეციალური მხარე, თუ რამდენი მოგება უნდა ჰქონდეს ყოველდღიურად ან ყოველ კვირა, მაგრამ ძალზე იშვიათია, რომ იგი მის საერთო ნაწილში ერკვეოდეს. გაკოტრების მეტი წილი მართვის, გასაღების და ფინანსების გადანაწილების უკმარისობითაა გამოწვეული, რომელსაც მოჰყვება არასწორი ტაქტიკური გადაწყვეტილებანი, რაც იწვევს გაკოტრებას“[14,15].

ჩვენი ქვეყნის მძიმე სოციალურ ეკონომიური მდგომარეობისა და არსებული საგადასახადო კანონმდებლობის გათვალისწინებით საკმაოდ რთულია თავი დავაღწიოთ გაკოტრებას და ვაწარმოოთ წარმატებული კონკურენტუნარიანი ბიზნესი. ჩვენი მიზანია ბიზნესის მართვის სისტემისთვის შევქმნათ გადაწყვეტილებათა მიღების ხელშემწყობი, ისეთი ეფექტიანი მოდელი, რომლის მთავარი ორიენტირი იქნება წარმატებისათვის აუცილებელი სამუშაოების შესრულების ოპტიმალური პირობების შექმნა.

ბიზნესის ერთ-ერთ საყოველთაოდ გავრცელებულ სახეობას წარმოადგენს ე.წ. ‘სავაჭრო ობიექტთა ბიზნესი“, სადაც საქმე გვაქვს კლიენტების მომსახურებასთან. ერთი შეხედვით ამ მარტივ საქმიანობასთან დაკავშირებულია საკმაოდ დიდი სირთულეები.

კონკრეტულად განვიხილოთ მაღაზიათა მუშაობის პრინციპი. მაღაზია ყოველთვის დროულად უნდა იყოს უზრუნველყოფილი კლიენტის მოთხოვნილების შესაბამისი მაღალი ხარისხის პროდუქციით, რადგან არარეალიზებადი საქონელი იგივეა, რაც მაღაზიაში არსებული ცარიელი დახლები. (არსებობს კლიენტთა მოზიდვის სხვა მრავალი ფაქტორი, რომელთა განხილვასაც აქ არ შევუდგებით). კლიენტთა მოთხოვნილების დაკმაყოფილების

პირობის უკან დგას ერთ-ერთი მნიშვნელოვანი ფაქტორი, რაც მდგომარეობს მაღაზიის მომსახურებაში.

თუ ჩვენ სათანადო ანალიზის გარეშე გავზრდით მომსახურე პერსონალს, რათა დროულად და მაღალ დონეზე მოხდეს შესაბამისი მომსახურება, შესაძლებელია მიღებული ხარჯები იმდენად დიდი აღმოჩნდეს, რომ ბევრად გადააჭარბოს არსებულ მოგებას, რაც საბოლოოდ მიგვიყვანს გაკოტრებამდე. ამიტომ უნდა შევარჩიოთ საქმიანობის ისეთი მოდელი, რომელიც ზუსტად განსაზღვრავს, ჩვენს მიერ შერჩეულ ობიექტის მომგებიან მუშაობას.

ეკონომიკის სფეროში თეორიულ და განსაკუთრებით პრაქტიკულ დონეზე ხშირად ისმის ისეთი ამოცანები სადაც, საქმე ეხება დიდი რაოდენობით მომხმარებელთა მომსახურების პროცესს. სწორედ ასეთი ტიპის ამაცანათა გადასაჭრელად ყველაზე ეფექტური საშუალებაა მასობრივი მომსახურების თეორიის (მმთ) გამოყენება.

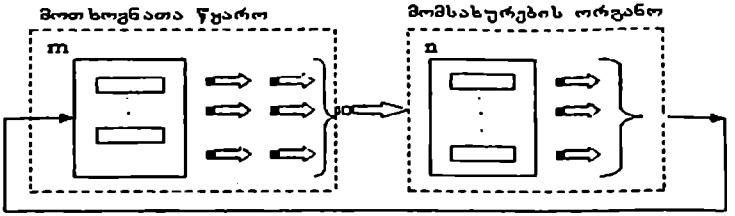
მმთ-ს გამოყენებით საშუალება გვეძლევა გამოვიკვლიოთ და შევისწავლოთ სისტემის მართვის პროცესები, დავადგინოთ სათანადო კანონზომიერება ამ სისტემის ცალკეულ ელემენტებს შორის და მიღებული შედეგები გამოვიყენოთ აღნიშნულ პროცესთა სრულყოფაში.

ჩვენი შემდგომი მიზნებისათვის განვიხილავთ შემდეგი სახის სისტემას.

მასობრივი მომსახურების სისტემა შედგება ორი სახეობის ელემენტისაგან 1. მომხმარებელი ობიექტები, რომლებიც საჭიროებენ სათანადო მომსახურებას და 2. მომსახურე ობიექტები, რომლებიც განკუთვნილნი არიან მომხმარებელს გაუწიონ მომსახურება. ჩვენ განვიხილავთ ასეთ ამოცანას: მოცემული გვაქვს სავაჭრო ფირმა m რაოდენობის მაღაზიებით. ამ მაღაზიათა ქსელს, საქონლის მომარაგებაზე ემსახურება n ბრიგადა. როგორც კი რომელიმე მაღაზიაში გაჩნდება მოთხოვნა რომელიმე სახის საქონელზე, მაღაზიის ხელმძღვანელი უკავშირდება საწყობს და უკვეთავს საქონელს. ყოველ ბრიგადას ერთსა და იმავე დროს შეუძლია მოემსახუროს მხოლოდ ერთ მაღაზიას, ხოლო თუკი შეკვეთის მოსვლის დროს ყველა ბრიგადა დაკავებულია, მაშინ საქონლის მიტანა გადაიდება მანამდის, სანამ არ განთავისუფლდება რომელიმე

ბრიგადა. თუ მოთხოვნათა რაოდენობა გადააჭარბებს ბრიგადების რაოდენობას მაშინ წარმოიქმნება რიგი მომსახურებაზე.

მომხმარებელ ობიექტს წარმოადგენს მაღაზიათა m რაოდენობა, (ნახ.3.7) რაც შეადგენს მოთხოვნათა m მოცულობის სასრულ წყაროს. მომსახურეობის ობიექტს კი შეადგენს n რაოდენობის მომარაგების ბრიგადა.



ნახ.3.7

შემოვიტანოთ აღნიშვნები:

1. ერთი მაღაზიიდან შემოსულ მოთხოვნათა საშუალო რაოდენობა დროის ერთეულში - λ

2. ერთი მოთხოვნის მომსახურების საშუალო დრო – $1/\mu$ მომსახურებული მოთხოვნების საშ. რაოდენობა დროის ერთეულში $-\alpha$.

3. ერთი მოთხოვნის მომსახურებიდან შემოსული საშუალო შემოსავალი დროის ერთეულში – c_1.

4. მომსახურების ერთი ორგანოს შენახვა დროის ერთეულში ჯდება – c_2 ლარი

5. კლიენტის ლოდინი დროის ერთეულში იწვევს ჯარიმას– c_3 ლარი.

სისტემაში დაყენებული მოთხოვნა გადაეცემა მომსახურე ორგანოებს და იწყება მოთხოვნის დაკმაყოფილება. იმ შემთხვევაში თუ რომელიმე მომსახურე ორგანო თავისუფალია, წინააღმდეგ შემთხვევაში მოთხოვნა დგება რიგში და ელოდება, სანამ არ განთავისუფლდება რომელიმე მომსახურე ორგანო და მისი დაკმაყოფილება მოხდება მხოლოდ იმ შემთხვევაში, როცა დაკმაყოფილდება მის წინ მდგომი ყველა მოთხოვნა. მაშასადამე მოქმედებს მომსახურების არაპრიორიტეტული დისციპლინა FCFS

(First Come, First Served) – „მოვიდა პირველი, მომსახურდა პირველი“).

მომსახურების სისტემაში სრულდება ორი სახის ოპერაცია: 1) მოთხოვნის „წარმოშობა“; 2) მოთხოვნის მომსახურება. პირველი არის ფიქტიური ოპერაცია, მეორე რეალური. დავუშვათ, რომ ორივე სახის ოპერაციის ხანგრძლივობები ექსპონენტურად განაწილებული შემთხვევითი სიდიდეებია. ასეთ პირობებში სისტემის ფუნქციობა აღიწერება მარკოვის პროცესით მდგომარეობათა დისკრეტული სივრცითა და უწყვეტი დროით.

სისტემის ყოფაქცევა დროში აღიწერება ფუნქციებით

P(i,t)= P{t მომენტში მომსახურებაზე და რიგში მყოფი მოთხოვნათა რაოდენობა არის i }. $i=\overline{0,m}$.

სტანდარტული ალბათური მსჯელობის საფუძველზე მიღებულია დიფერენციალურ განტოლებატა სისტემა P(i,t) ფუნქციების მიმართ (კოლმოგოროვის განტოლებები).

სისტემის სტაციონალური მდგომარეობა აღიწერება ალგებრულ განტოლებათა შემდეგი სისტემით:

$$\begin{cases} -m\lambda P(0)+\mu P(1)=0 \\ -((m-i)\lambda+i\mu)P(i)+(i+1)\mu P(i+1)+\lambda(m-i+1)P(i-1)=0 & 1\le i<n. \\ -((m-i)\lambda+n\mu)P(i)+n\mu P(i+1)+\lambda(m-i+1)P(i-1)=0, & n\le i<m \\ -n\mu P(i)+(m-i+1)\lambda P(i-1)=0 \end{cases}$$

სადაც, $\lim_{t\to\infty} P(i,t)= P(i)$

ამასთანავე $\sum_{i=1}^{m} P(i)=1$

– სისტემის ეკონომიკური ეფექტიანობის მაჩვენებელი აღვნიშნოთ F. იგი არის დროის ერთეულში არსებული სუფთა მოგება: F=F(m,n, λ , μ);

– დროის ერთეულში სისტემის შემოსავალი შეადგენს - $c_1\alpha$.

– სისტემის დანახარჯები შეადგენს – $c_2 n$;

– კლიენტის ლოდინით გამოწვეული ჯარიმა იქნება – δ, სადა δ - არის დროის ნებისმიერ მომენტში მომსახურებაში მყოფი მოთხოვნების საშ. რაოდენობა.

ალბათობის თეორიიდან ცნობილია, რომ

$$\delta = \sum_{i=0} iP(i)$$

ცხადია, რომ δ არის m,n,λ,μ პარამეტრების ფუნქცია

$$\delta = \delta(m,n,\lambda,\mu)$$

ასევე α - არის იგივე პარამეტრების ფუნქცია $\alpha = \alpha(m,n,\lambda,)$. საბოლოოდ მივიღებთ სისტემის მოგების ფუნქციას:

$$F=F(m,n,\lambda,\mu) = c_1\alpha\ (m,n,\lambda,\mu) - c_2 n - c_3\delta(m,n,\lambda,\mu).$$

ამ ფუნქციის გამოყენებით შეიძლება დაისვას და ამოიხსნას მათემატიკური დაპროგრამების შემდეგი ამოცანები:

1) m,λ,μ - პარამეტრები ფიქსირებულია, ვიპოვოთ n-ის ის მნიშვნელობა, რომელიც F – ფუნქციას მიანიჭებს მაქსიმალურ მნიშვნელობას.

2) m,λ,μ - პარამეტრები ფიქსირებულია, ვიპოვოთ m - ის ის მნიშვნელობა, რომელიც F – ფუნქციას მიანიჭებს მაქსიმალურ მნიშვნელობას.

№	m, n	P(0)	P(1)	P(2)	P(3)	P(4)	P(5)
1	5, 1	0.3*10-6	0.008	0.0007	0.014	0.19	0.79
2	5, 2	0.25*10-6	0.00002	0.0006	0.012	0.15	0.08
3	5, 3	0.003*10-5	0.0004	0.00004	0.00012	0,2	0,8
4	5, 4	0.260.10-5	0.00015	0.0046	0.06	0.3	0.55
5	5, 5	0.24*10-5	0.0002	0.0052	0.07	0.42	0.50

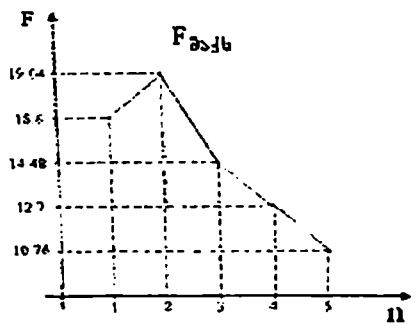
ფორმულის გათვალისწინებით შეგვიძლია დავწეროთ, როცა:

1) n=1, m=5, δ=4,7;
2) n=2, m=5, δ= 1,1;
3) n=3, m=5, δ= 4,8;
4) n=4, m=5, δ= 4,5;
5) n=5, m=5, δ= 4,4;

საბოლოოდ: F მნიშვნელობისათვის შეგვიძლია დავწეროთ:

№	c_1	c_2	c_3	n	α	δ	F
1	3.4	2.01	0.68	1	5	4.7	18.6
2	3.4	2.01	0.68	2	5	1.1	19.04
3	3.4	2.01	0.68	3	5	4.8	14.48
4	3.4	2.01	0.68	4	5	4.5	12.7
5	3.4	2.01	0.68	5	5	4.4	10.76

ნახ.3.8



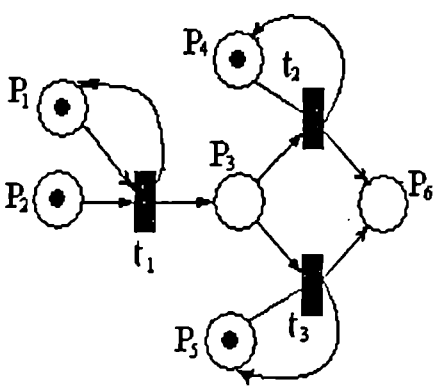
განხილული ხუთი შემთხვევიდან ოპტიმალური ვარიანტია მე-2 შემთხვევა, როცა m=5, n=2; ვიზუალურად შეიძლება ეს პროცესი გამოვსახოთ გრაფიკის საშუალებით, თუ აბსცისათა ღერძზე მომსახურების ობიექტთა მნიშვნელობებს გადავზომავთ, ხოლო ორდინატთა ღერძზე შესაბამის მოგების კოეფიციენტებს, მივიღებთ: $F_{მოგ.} = \max$ (ნახ.3.8)

ფაქტობრივად ეს არის მთელრიცხვა დაპროგრამების ამოცანები, რომელთა ამოხსნა m-ის და n-ის რეალური მნიშვნელობისათვის $n \le m \le 30$ შესაძლებელია კომპიუტერული გადარჩევის მეთოდით.

3.4. კომპიუტერული ქსელის რესურსების სინქრონიზების პროცესის მოდელირება პეტრის ქსელებით მრავალმომხმარებლურ რეჟიმში

პეტრის ქსელების თეორიის გამოყენება რეალური სამყაროს განსხვავებული საგნობრივი სფეროების მოდელირებისა და ანალიზისათვის პოპულარულად არის წარმოდგენილი მრავალ ნაშრომში [10-14]. ძირითადად განიხილება პროცესების, გამთვლითი სისტემების ტექნიკური და პროგრამული უზრუნველყოფის სხვადასხვა სახის ამოცანების მოდელირების საკითხები. მათ შორის პარალელური პროცესების ეფექტური მართვისათვისაც. ნახ.3.9-ზე მოცემულია პეტრის ქსელის გრაფით კომპიუტერული სისტემის ძირითადი რესურსების წარმოდგენის ტიპური მაგალითი [1].

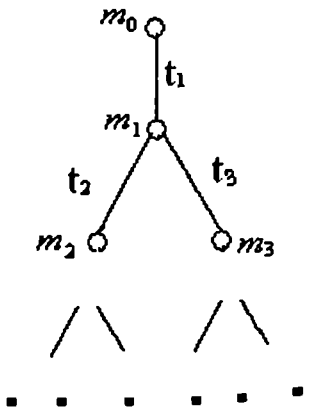
აქ P_1 აღწერს მოვლენას, რომ ცენტრალური პროცესორი თავისუფალია; P_2 – მოთხოვნის შემოსვლა და მოლოდინის გადასვლა; P_3 – მოთხოვნა შეტანა-გამოტანის მოლოდინის მდგომარეობაშია; P_4 – პირველი შ/გ-1 მოწყობილობა თავისუფალია; P_5 – შ/გ-2 თავისუფალია; P_6 – მოთხოვნა შესრულებულია.



ნახ. 3.9

გადასასვლელები ასახავს შემდეგ პროცესებს: t_1 – ცენტრალური პროცესორი ასრულებს მოთხოვნას; t_2 – მუშაობს შ/გ-1 და t_3 – შ/გ-2 მოწყობილობანი. გამთვლითი სისტემის მუშაობის პროცესის მოდელირება მდგომარეობათა სივრცეში გამოისახება შემდეგ მარკირებათა ვექტორერებით:

	P_1	P_2	P_3	P_4	P_5	P_6
m_0	1	1	0	1	1	0
m_1	1	0	1	1	1	0
m_2	1	0	0	1	1	1



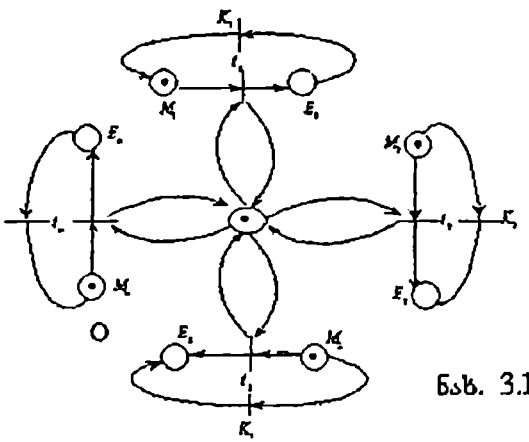
ნახ. 3.10

3.10 ნახაზზე მოცემულია მისი შესაბამისი მიღწევადობის ხის ფრაგმენტი. ამ მაგალითიდან კარგად ჩანს, რომ t_1-ის გაშვების შემდეგ (იხ. სტრიქონი m_1) მარკერი გადაადგილდება P_2-დან P_3-ში და ამ სიტუაციაში წარმოიშობა კონფლიქტური სიტუაცია: შეიძლება t_2-ის ან t_3-ის გაშვება, მაგრამ ერთის გაშვება აბლოკირებს მეორეს. m_1 მდგომარეობაში შესაძლებელია ახალი მოთხოვნის მოსვლა (P_1-ში ჩნდება მარკერი) და ამ სიტუაციაში შესაძლებელია პარალელურად t_1 და t_2 ან t_1 და t_3-ის შესრულება. ახლა განვიხილოთ ჩვენი ერთ-ერთი ამოცანისათვის პეტრის ქსელის ინსტრუმენტის გამოყენება შესაძლებლობა, კერძოდ ლოკალურ ქსელში მონაცემთა განაწილებული ბაზების სინქრონიზებული გამოყენება მრავალმომხმარებლურ რეჟიმში.

ქსელის ტოპოლოგია შევირჩიოთ საერთო სალტით (ან ვარსკლავური) ერთი სერვერით. დავუშვათ, რომ მონაცემთა ბაზები განაწილებულია კვაძებში შერეული სტრატეგიის შესაბამისად (როგორც ყველაზე რთული მოდელი), ხოლო მათი კატალოგი მოთავსებულია ცენტრალიზებულ სერვერ-მონაცემთა ბაზაში. კვანძებში დასაშვებია მათი ლოკალური ბაზების მონაცემთა კატალოგების ქვესიმრავლეების არსებობა. მომხმარებლები სარგებლობენ თანაბარი პრიორიტეტით. მათი მოთხოვნების (ტრანზაქციების) მოსვლის ალბათობა განიხილება განაწილების პუასონის კანონით [22,23,32].

მოთხოვნები მუშავდება ლოკალურ კვანძებში თუ მათ სჭირდებათ სხვა ბაზების მონაცემები, წინააღმდეგ შემთხვევაში ისინი მიმართავენ სერვერ-ეგმს და თუ შესაბამისი გადაცემის არხი და ბაზის ფაილები თავისუფალია, მიიღებენ გარკვეული პრედიკატით დამუშავებულ ინფორმაციას. თუ მონაცემები ბლოკირებულია სხვა ტრანზაქციებით, მაშინ ეს მოთხოვნა დგება შესრულების რიგში ან ხელმეორედ მიეწოდება გარკვეული დროის ინტერვალის შემდეგ.

პეტრის ქსელების გამოყენებით მსგავსი პროცესების მოდელირება შესაძლებელია სინქრონიზაციის უნარის მქონე პროცედურებით [12]. ასეთ ამოცანათა კლასს მიეკუთვნება პროცესების ურთიერთგამორიცხვის, ჩიხების რეგულირების ('ბრძენი ჩინელების შესახებ"), p და v -ოპერაციები სემაფორზე და სხვა. ნახაზზე 3.11 წარმოდგენილი გვაქვს მრავალმომხმარებლურ რეჟიმში, ზოგადად n-კვანძისა და ერთი სერვერ-მანქანის შეთანხმებული ფუნქციონირების პროცესის ფრაგმენტი პეტრის ქსელის გრაფით, კერძოდ მოთხოვნების (ტრანზაქციების) დასაკმაყოფილებლად.



ნახ. 3.11

სქემის პოზიციებია: M – მოთხოვნა ელოდება შესრულებას (პოზიციაში არის მინიმუმ ერთი მარკერი), ან არ ელოდება (არაა მარკერი); E _ დამუშავებული მოთხოვნა (მარკერი ≥ 1), ან დაუმუშავებელი $(=0)$; S – სინქრონიზაციის პოზიციაა და მისი საშუალებით აქ მოდელირდება საერთო რესურსი (მაგალითად, მონაცემები, გადაცემის არცი და ა.შ.) მისი მნიშვნელობაა – რესურსი

თავისუფალია გამოსაყენებლად (მარკერი ≥1), ან არ არის თავისუფალი (=0). სქემის გადასასვლელებია; t – მოთხოვნა მუშავდება; k – მოთხოვნა დამუშავდა. საწყის მდგომარეობაში შეიძლება დავუშვათ, რომ მოთხოვნა ყველა კვანძში შემოსულია და ელოდება შესრულებას , S სერვერ-მანქანის რესურსი თავისუფალია (S=1). მოცემულ სიტუაციაში შესაძლებელია ნებისმიერი ერთი გადასასვლელის გახსნა. აქ ადგილი აქვს კონფლიქტურ სიტუაციას, რადგან ერთის გახსნა აბლოკირებს დანარჩენებს. ვთქვათ, შესრულდა t_1, მაშინ მარკერი M_1-დან გადაადგილდება E_1-ში და მარკერი S-ში იგივე რჩება (გაიცემა 1 და ემატება 1). მეორე ბიჯზე შესაძლოა ორი პარალელური პროცედურის შესრულება:

1) K_1-ის გაშვება და ახალი მოთხოვნის მომზადება პირველ კვანძში;

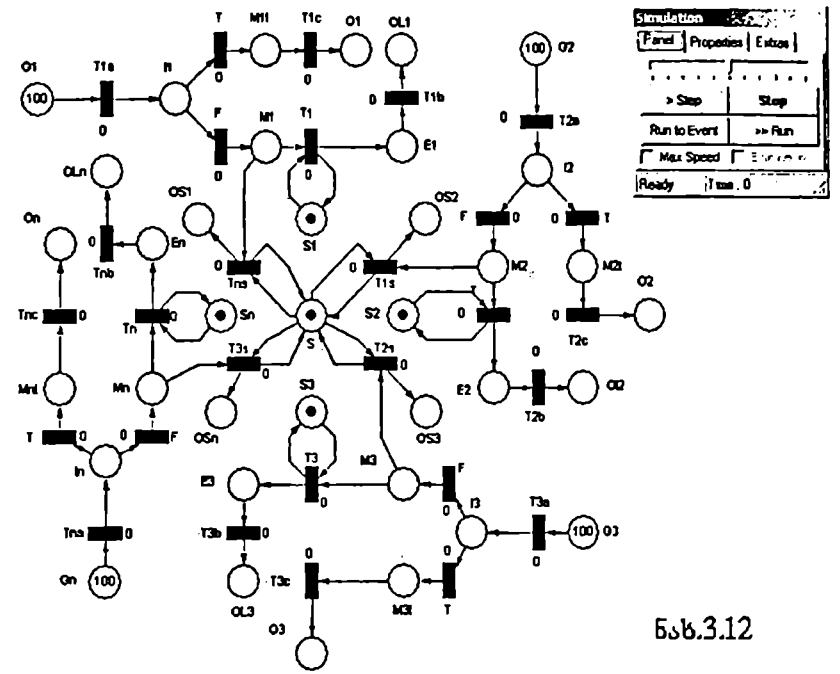
2) ერთი რომელიმე გადასასვლელის გახსნა.

ამგვარად, მიმდევრობით გადამუშავდება მოთხოვნები ყველა კვანძში, თუმცა შესაძლოა პროცესი ისე წარიმართოს, რომ 1 და 2-ის მონაცვლეობით მივიღოთ უსასრულო მოლოდინის ციკლი [12]. ახლა შედარებით გავაღრმავოთ (დავაზუსტოთ) მოდელირების პროცესი:

სინქრონიზაციის პოზიციის (S) დეტალიზებული აღწერა, მოთხოვნების ფორმირების გენერატორის შემოტანით (G), მოთხოვნის ანალიზისა (I) და მისი დამუშავების შედეგების (O) გაცემით. 3.11-ე ნახაზზე მოცემულია ამ პროცესის პეტრის ქსელის ფრაგმენტი. აქ არის მოთხოვნის ანალიზის ჩატარება რესურსების შესახებ (ლოკალურ კატალოგში). თუ მოთხოვნისათვის ყველა საჭირო რესურსი აქვეა, მაშინ (–I–.T/.F.) გადაწყვეტილების ბლოკი განსაზღვრავს ერთი მდგომარეობიდან მეორეში გადასვლას (ლოკალურ ბაზასთან მიმართვა), რომლის შემდეგაც გაიშვება გადასასვლელი (შედეგის გაცემა) O_i-ში.

თუ გადაწყვეტილების ბლოკი იძლევა F-ს, მაშინ მოთხოვნა ფორმირდება საერთო რესურსების გამოსაყენებლად და იგი მიმართავს სერვერ-მბ-ს (უფრო ზუსტად მოთხოვნა გადადის დამუშავების მოლოდინის მდგომარეობაში), M_i -ში თავსდება მარკერი და იგი ელოდება სინქრონიზაციას -1 პოზიციიდან. ზოგადად შეიძლება

მივიღოთ, რომ სინქრონიზაციის s ვექტორის ელემენტის გადასასვლელების მიმდევრობითი (სინქრონული გაშვებით) ღებულობენ მარკერს, როდესაც კვანძში მოთხოვნაა და პოზიციაში მოვიდა მარკერი, მაშინ იხსნება გადასასვლელი. მოთხოვნა ღებულობს არხით საჭირო მონაცემებს და მარკერი გადადის E_i პოზიციაში. S_i-დან მარკერი გაიცემა t_j-ის გაშვების დროს, ამიტომ გადასასვლელი ბლოკირებულია. t_j-ის გაშვებით S_i-ში ბრუნდება მარკერი, რის შემდეგაც შესაძლოა მარკერის გადაგზავნა შემდეგი კვანძისათვის. შესაძლებელია ნახ. 3.12-ის განხილვა, როგორც ალტერნატიული ქსელის ფრაგმენტი.



ნახ.3.12

ამგვარად კვლევის ობიექტის შემდგომი დაზუსტება, რაც ასახული იქნება პეტრის ქსელის შესაბამის დეტალიზებულ ფრაგმენტში. ამიტომ დამპროექტებელი თვითონ წყვეტს, თუ ობიექტის ქცევის რომელი დეტალიზაციის მოდელირებას აპირებს. პეტრის ქსელის

ანალიზისათვის შესაძლებელია აგრეთვე გადასასვლელებისათვის დაყოვნების დროითი პარამეტრების განხილვა. ამ მხრივ საყურადღებოა პეტრის ქსელის დეტერმინირებული და სტოქასტური მოდელირების ფუნქციონირების ანალიზი [12].

3.5. პეტრის ქსელების მიზეზ-შედეგობრივი პროცესების პრედიკატულ ფორმაში ასახვა.

გერმანიის სანქტ-ავგუსტინის უნივერსიტეტის პროფესორის, ვოლფგანგ რეისიგის მიერ (კარლ პეტრის მოწაფე) ღრმად იქნა შესწავლილი და წარმოდგენილი პეტრის ქსელების გამოყენების შედეგები ქცევის მოდელირებისა და ანალიზისათვის, მათი კავშირი ნაწილობრივ მოწესრიგებულ სისტემებთან, ასახვის პრედიკატულ და რელაციურ ფორმებთან და ა.შ. [2-3].

ჩვენი მიზანია ზემოთ წარმოდგენილი განაწილებული რესურსების სინქრონიზაციის მოდელის შესაბამისი პეტრის ქსელის ანალიზი და პრედიკატული ფორმით წარმოდგენა. 3.13 (ა,ბ) ნახაზის შესაბამისად განვიხილოთ მიზეზშედეგობრივი დამოკიდებულების მოდელის ფრაგმენტი, აქ m და e პრედიკატები განიხილება როგორც პოზიციები, ხოლო M_i, E_i–მოთხოვნები – როგორც მარკერები. ქსელის პოზიციების განზოგადებით მივიღებთ პრედიკატებს(ნახ.3.14):

$m(M_1,M_2, \ldots M_n)$ – „მომლოდინე მოთხოვნები“ ($m(P_i,i=2,n)$;

$e(E_1,E_2, \ldots E_n)$ – „დამუშავებული მოთხოვნები“ ($e(P_1)$;

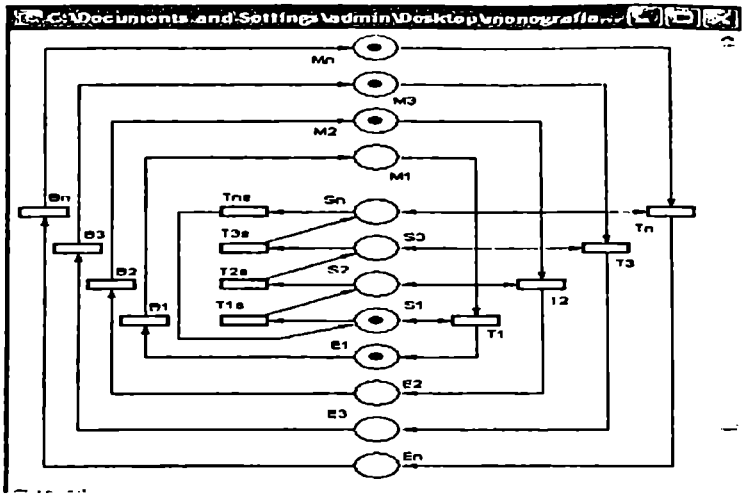
$s(S_1,S_2, \ldots S_n)$ – „თავისუფალი რესურსები“ ($s(S_1)$ და ა.შ. მომდევნო ბიჯზე განვაზოგადებთ გადასასვლელებს და მივიღებთ 3.15 ნახაზზე წარმოდგენილ სურათს. აქ x ცვლადია, რომელიც იღებს კონკრეტულ მნიშვნელობებს, ხოლო z(x) – ით აღიწერება სემანტიკური კანონი, მაგალითად, რესურსების განაწილების შესახებ.

Z:P->S, სადაც და P_i->P_i->S_i+1,

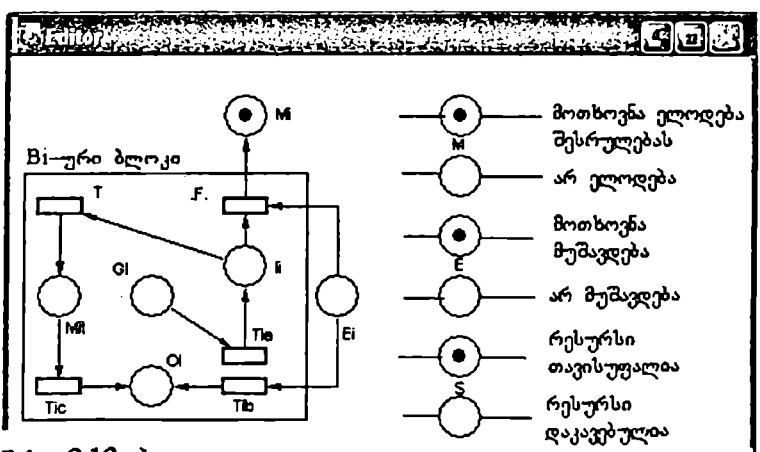
if $\overline{i=1,n-1}$ და P_n->S_1

პრედიკატებს შორის არსებული ფაქტორების ასახვის სქემა მოცემულია აქვე. ფაქტებს აქვთ შემდეგი ლოგიკური გამოსახვა და სემანტიკა:

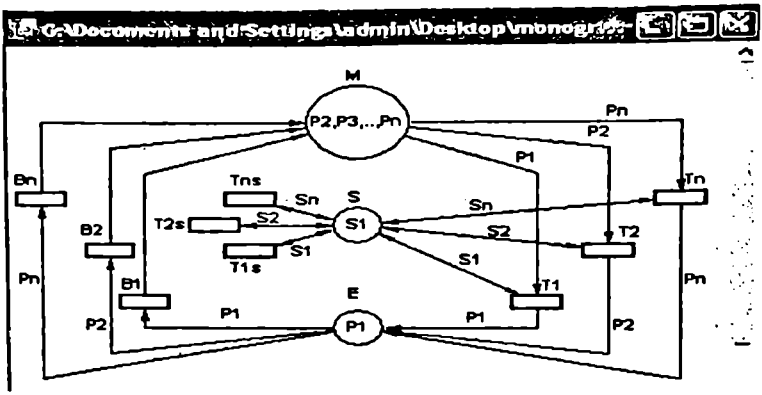
$T_1 : \neg(m(x) \wedge e(x))$: მოთხოვნა (ტრანზაქცია) რომელიც მუშავდება, არ ელოდება შესრულებას და მოთხოვნა, რომელიც ელოდება შესრულებას, არ მუშავდება;



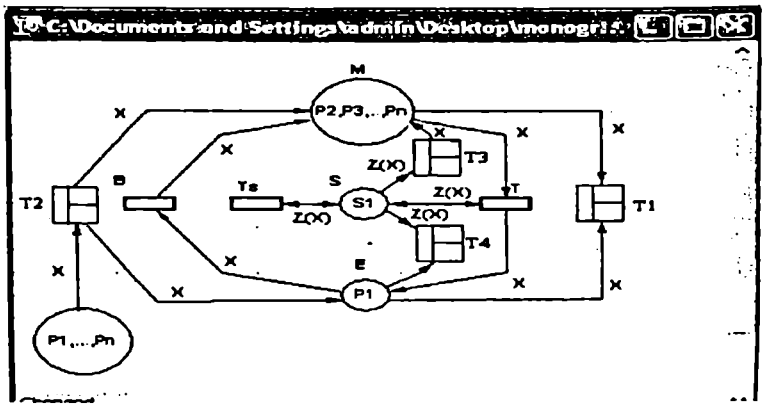
ნახ. 3.13 - ა



ნახ. 3.13 - ბ



ნახ. 3.14



ნახ. 3.15

T_2: $P(x) \rightarrow m(x) \vee e(x)$: ყოველი მოთხოვნა მუშავდება ან ელოდება შესრულებას, სხვა მოქმედება გამორიცხულია.

T_3 : s (z (x)) $\rightarrow m(x)$: თუ რესურსები თავისუფალია, მაშინ მოთხოვნა არ მუშავდება;

T_4 : $\neg(e(x) \wedge s(z(x)))$: თუ მოთხოვნამუშავდება, მაშინ რესურსი დაკავებულია, და თუ რესურსი არაა დაკავებული, მაშინ მოთხოვნა არ მუშავდება.

3.6. განაწილებულ სისტემებში მონაცემთა საცავის ინფორმაციული ბლოკების დამუშავების პროცესების ასახვა პეტრის ქსელების გამოყენებით

თანამედროვე მართვის საინფორმაციო სისტემების დაპროექტება და რეალიზაცია ხორციელდება უახლესი ინფორმაციული ტექნოლოგიებით, რომელთა საფუძველსაც ობიექტ – ორიენტირებული მიდგომა წარმოადგენს.

კომპიუტერული და ქსელური ინდუსტრიის განვითარებამ ბოლო ათწლეულებში მნიშვნელოვან შედეგებს მიაღწია, რამაც თითქმის მთლიანად შეცვალა მართვის საინფორმაციო სისტემის აგების ტექნოლოგია და ინსტრუმენტული საშუალებანი, გაჩნდა ახალი ცნებები და ტერმინები, რომლებიც ახალ კონცეფციებსა და პროექტებს ედება საფუძვლად.

ორგანიზაციული სისტემების მართვის პრობლემების და ამოცანების გადაწყვეტის თანამედროვე კომპიუტერული და ინფორმაციული ტექნოლოგიების გამოყენების ერთ - ერთი აქტუალური და მნიშვნელოვანი მიმართულებაა „მონაცემთა საცავი“ (**data warehaus**) [9,35,39], რომელიც უკავშირდება მრავალდონიან განაწილებულ საინფორმაციო სისტემის შექმნას.

მონაცემთა საცავის დამკვიდრება გამოწვეული იყო იმ აუცილებლობით, რომელიც განაპირობა მართვის დიდ და რთულ სისტემებში განაწილებული ინფორმაციული ბაზების დიდ მონაცემთა "საწყობში“ თავმოყრაში. ეს უკანასკნელი შესაძლებელს ხდის მომხმარებელს მიეცეს მონაცემთა ინტერაქტიული კრიტერიუმებით განსაზღვრის ტექნიკური საშუალება, საჭირო მონაცემთა სტრუქტურის ინტენსიურობის დასადგენად. ამის გამო ანალიზური პროცესების სისტემა ეყრდნობა მრავალგანზომილებიან მონაცემთა სტრუქტურას, რომელიც მონაცემთა ადეკვატურ პრეზენტაციას უწყობს ხელს.

ბოლო პერიოდის განმავლობაში პროგრამული ინდუსტრიის სწრაფმა განვითარებამ მნიშვნელოვან წარმატებას მიაღწია. მათ შორის უნდა აღინიშნოს მონაცემთა ბაზების მართვის სისტემების პროგრამული პაკეტების უდიდესი წილი მსოფლიო ბაზარზე, რომელთა შორის ღირსეული ადგილი უკავია ORACLE, SQL Server და სხვ[3,10].

როგორც აღვნიშნეთ, მრავალდონიანი განაწილებული საინფორმაციო სისტემების მონაცემთა საწყობებში თავს იყრის ამ ობიექტისათვის მნიშვნელოვანი ინფორმაციული ბლოკები, რომელთა კლასიფიკაცია შეიძლება კონტექსტური ასპექტებით განხორციელდეს.

მაგალითად, თუ განვიხილავთ დიდ სავაჭრო ცანტრებს, ინფორმაციული ობიექტის ისტორიას, პროდუქციის კონიუნქტურას, ფასების, საქონელბრუნვის, თანამშრომელთა, სავაჭრო გეგმისა და ფაქტობრივი ფინანსური შემოსავლების, პარტნიორებისა და კონკურენტების, კლიენტებსა და საერთო ბაზრის მოთხოვნილებების შესახებ და ა.შ. ინფორმაციული ბლოკები, რომლებიც მონაცემთა საცავებშია განაწილებული, მიზანმიმართულად თავსდება ინტერნეტ – გვერდებზე და ხელმიასწვდომია ფართო მოხმარებისთვის.

ობიექტ–ორიენტირებული ანალიზისა და დაპროექტების მიდგომა გულისხმობს არა მხოლოდ „საინფორმაციო საცავების“ შექმნას, არამედ მისი ინფორმაციული ბლოკების მიზნობრივად დამუშავების პროგრამების პაკეტების შექმნას და გამოყენებას. კლასისათვის, როგორც მართვის საინფორმაციო სისტემის ძირითადი ელემენტებისათვის დამახასიათებელია ინკაფსულაციის, მემკვიდრეობითობის და პოლიმორფიზმის თვისებები. ინკაფსულაცია კი გულისხმობს გარკვეული მეთოდების შემუშავებას ინფორმაციული ბლოკების დასამუშავებლად.

მაგალითად, დიდ სავაჭრო ცენტრებისთვის ასეთი მეთოდები შეიძლება იყოს მომხმარებელთა მოთხოვნის განსაზღვრა (მარკეტული გამოკვლევის ინფორმაციული ბლოკი), სავაჭრო გეგმებისა და საქონელბრუნვის მართვა (ისტორიული, ტრადიციული, რეგიონალური ასპექტების გათვალისწინებით), მომხმარებელთა(კლიენტთა) მომსახურების ფორმებისა და პროცესის გაუმჯობესება (რიგების შემცირება, ყურადღებიანი მომსახურება და ა.შ.) პროგნოზის, ანალიზის, ოპტიმიზაციის მეთოდების გამოყენება და ა.შ.

მაგალითისთვის წინამდებარე ნაშრომში ჩვენ შევეხებით მასობრივი მომსახურების ისეთ პროგრამულ პაკეტს, როგორიცაა GPSS World –სისტემა, რომელიც დამუშავებულია Minuteman Software (აშშ) კომპანიის მიერ[34]. ეს კომპლექსური მოდელირების ინსტრუმენტი გამოიყენება დისკრეტული და უწყვეტი კომპიუტერული

მოდელირებისათვის. იგი მაღალ დონეზე ასახავს ინტერაქტიულობას და ინფორმაციის ვიზუალურ წარმოდგენას.

GPSS World –ს Windows ოპერაციული სისტემისთვის აქვს გაფართოებული შესაძლებლობა, ამასთანავე იგი წარმოადგენს ინტერნეტიდან ინტეგრირებული ფუნქციის მუშობის სამომხმარებლო საშუალებას.

GPSS World – ის ძლიერი მხარეა მისი გამჭვირვალობა მომხმარებლისთვის. ეს თვისება ღირებულია სამი მიზეზის გამო: პირველ რიგში საშიშია გამოვიყენოთ უცნობი „შავი ყუთის" ტიპის მოდელირება, რომლის შიდა მექანიზმის ფუნქციონირება დაფარულია მომხმარებლისთვის. ამ შემთხვევაში არ შეიძლება იყო დარწმუნებული, მიესადაგება თუ არა რომელიმე კონკრეტულ შემთხვევას. აგრეთვე არ გვაქვს არანაირი გარანტია, რომ ის იმუშავებს ჩვენი მოთხოვნების შესაბამისად.

მეორე შემთხვევაში, წარმატებული იმიტაციური მოდელი ღირებული და ვარგისია დროის ხანგრძლივ მონაკვეთში. შეიძლება საჭირო გახდეს ახალი მომხმარებლებისთვის მოდელირების შიდა პროცესების გაცნობა, ეს კი თითქმის შეუძლებელია, თუ მოდელი არ არის მაღალ დონეზე გამჭვირვალე.

მესამე კი ერთ-ერთი ყველაზე ეფექტური და ნაკლებად ცნობილი მოდელია – სისტემის ფუნქციონირების არსში შეღწევა, როდესაც გამოცდილ პროფესიონალს მოდელირებისას შეუძლია დაინახოს მოდელირების შიგა დინამიკა[34].

განვიხილოთ მაგალითი: „სავაჭრო ცენტრში კლიენტებთან მოლარის მომსახურება". კლიენტები შემოდიან მარკეტში, შეარჩევენ მათთვის სასურველ პროდუქციას და ელოდებიან მომსახურებას. სიტუაცია შეიძლება იყოს ასეთი: მოლარე დაკავებულია და კლიენტი დგება რიგში, ან თავისუფალია და ემსახურება მას. დავუშვათ ყოველი კლიენტის გამოჩენის მომენტი და მომსახურების დრო მოლარისთვის ცნობილია. ჩვენი მიზანია შევადგინოთ ისეთი სისტემის ფუნქციონირების იმიტაცია, რომლის დროსაც ცნობილი იქნება მოლარე დროის რამდენი პროცენტის განმავლობაში იქნება თავისუფალი და დროის რა შუალედი დასჭირდება კლიენტს მარკეტში ყოფნისათვის, რათა არ მოხდეს დიდი რიგების წარმოქმნა და კლიენტის დაკარგვა.

დროის განმავლობაში სისტემის მდგომარეობის ცვლილების დინამიკური ასახვა პირველ რიგში უნდა განისაზღვროს სამოდელირებელი სისტემის მდგომარეობის განსაზღვრით, ჩვენს შემთხვევაში მოლარის მდგომარეობით (დაკავებულია იგი თუ თავისუფალი) და კლიენტთა რაოდენობით.

სისტემის მდგომარეობა შეიცვლება შემდეგ შემთხვევებში 1) მარკეტში კლიენტთა შემოსვლით 2) მოლარის მომსახურების დამთავრებით. იმიტაციის ილუსტრირებისთვის სისტემის მდგომარეობას ჩვენ განვსაზღვრავთ ხდომილებათა დროითი მოწესრიგებით, იგი შეესაბამება კლიენტთა შემოსვლას და გასვლას [14].

იმიტაციის შედეგი გამომავალი მონაცემების შესაბამისად შეიძლება წარმოვადგინოთ ცხრ.3.1-ის სახით:

ცხრილი 3.1

კლიენტის ნომერი	შემოსვლის მომენტი	მომსახურეობის დრო
1	4,6	4,9
2	12,5	3,6
3	17,2	4,5
4	17,9	2,4

ამ ცხრილის გათვალისწინებით შეიძლება შევადგინოთ მეორე 3.2 ცხრილი რომლისთვისაც საჭიროა ჩავთვალოთ, რომ დროის საწყის მომენტში სისტემაში არ არის კლიენტი, მოლარე თავისუფალია და პირველი კლიენტი შემოდის როცა ის 4,6-ის ტოლია. მე-2 ცხრილში 1 და 2 სვეტები არსებულია პირველიდან. მომსახურეობის საწყისი დრო არის მესამე სვეტში, რომელიც დამოკიდებულია იმაზე მომდევნო კლიენტმა დატოვა თუ არა მარკეტი. მეოთხე სვეტში წასვლის დრო გამოითვლება, როგორც მესამე სვეტის შესაბამისი ელემენტების ჯამი და მოცემული კლიენტის მომსახურების დრო,

რომელიც გამოითვლება 1-ელი ცხრილის დახმარებით. ყოველი კლიენტის ყოფნის დრო რიგში და მარკეტში გამოითვლება მე-2. ცხრილის დახმარებით. ამ გარდაქმნათა საშუალო მნიშვნელობა შესაბამისად ტოლია 2,27 და 5,97.

ცხრილი 3.2 შეიცავს შემავსებელ ინფორმაციას კლიენტთან მიმართებაში, მაგრამ არ შეიცავს ცნობებს მოლარეზე და რიგის სიგრძეზე. ამ ინფორმაციის მიღებისთვის საჭიროა გამოვიკვლიოთ ამ სიტუაციასთან დაკავშირებული მოვლენები. ხდომილების დადგომისას „კლიენტის შემოსვლა" შემდგომი სიტუაცია მარკეტში განისაზღვრება მოლარის მდგომარეობით. თუ მოლარე თავისუფალია, ის გადადის მდგომარეობაში „დაკავებულია" და ემსახურება კლიენტს. ამით იგეგმება ხდომილება „მოცემული კლიენტის წასვლა", რომელიც დროის მომენტში ტოლია მიმდინარე დროს დამატებული მომსახურეობის დრო.

ცხრილი 3.2

კლიენტის ნომერი (1)	შემოსვლის მომენტი (2)	მომსახურების დაწყების მომენტი (3)	წასვლის მომენტი (4)	რიგში დგომის დრო (5)=(3)-(2)	ყოფნის დაყოვნილი დრო (6)=(4)-(2)
1	4,6	4,6	9,5	0	4,9
2	12,5	12,5	16,1	0	3,6
3	13,6	17,2	26,2	3,6	12,6
4	14,8	17,9	19,3	3,1	4,5

თუ მოლარე დაკავებულია, კლიენტთა მომსახურება არ შეიძლება დაიწყოს და შესაბამისად კლიენტი დგება რიგში (რიგის სიგრძე იზრდება ერთით). მოვლენათა დამუშავების ლოგიკა „კლიენტი ტოვებს მარკეტს" დამოკიდებულია რიგის სიგრძეზე. თუ რიგში დგას ერთი კლიენტი მაინც, მაშინ მოლარე ითვლება დაკავებულად, თუ კლიენტი ვერ შეძლებს დალოდებას, ტოვებს რიგს (რიგის სიგრძე შემცირდება ერთით). თუ რიგი თავისუფალია მოლარე ჩაითვლება თავისუფლად.

ცხრილი 3.3

შემოსვლის მომენტი	კლიენტის ნომერი	ხდომილების სახეები	რიგის სიგრძე	კლიენტთა რაოდენობა	მოლარის მდგომარეობა	მოლარის დაუსაქმებელი დრო
0,0	-	დასაწყისი	0	0	თავისუფ.	-
4,6	1	შემოსვლა	0	1	დაკავებ.	4,6
5,2	1	გასვლა	0	0	თავისუფ.	-
5,9	2	შემოსვლა	1	2	დაკავებ.	
7,3	3	შემოსვლა	2	3	დაკავებ.	
8,2	4	გასვლა	0	0	თავისუფ.	
12,5	5	შემოსვლა	1	1	დაკავებ.	3,6
13,6	3	შემოსვლა	1	4	დაკავებ.	8,3
13,9	6	გასვლა	1	1	დაკავებ.	
14,8	7	გასვლა	2	3	დაკავებ.	2,1
18,7	4	შემოსვლა	0	0	თავისუფ.	
19,4	5	გასვლა	1	1	დაკავებ.	
20,5	8	გასვლა	0	0	თავისუფ.	
26,8	9	შემოსვლა	2	2	დაკავებ.	
28,7	10	შემოსვლა	0	0	თავისუფ.	
30	10	გასვლა	0	0	თავისუფ.	

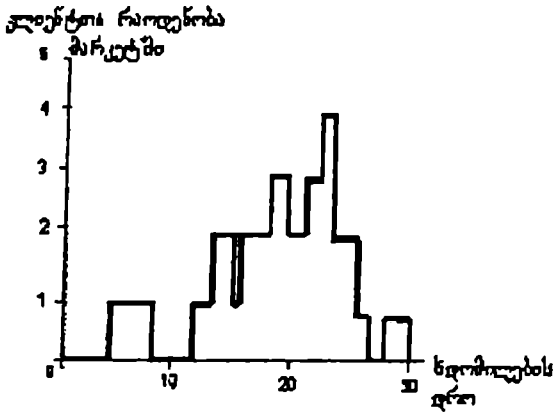
3.3 ცხრილში აღწერილია მოლარის მდგომარეობის ხდომილების-ორიენტირებული აღწერა და მარკეტში კლიენტთა რაოდენობა ამ სიდიდეთა ცვლილება დროში შეიძლება გამოვსახოთ გრაფიკულად

იმიტაციის შედეგი გვიჩვენებს რომ პირველი 30 წუთის განმავლობაში მარკეტში საშუალოდ ერთდროულად იმყოფებოდა 0,9 კლიენტი, ხოლო მოლარე თავისუფალი იყო დროის 18,6% პროცენტით. ქრონოლოგიურად,

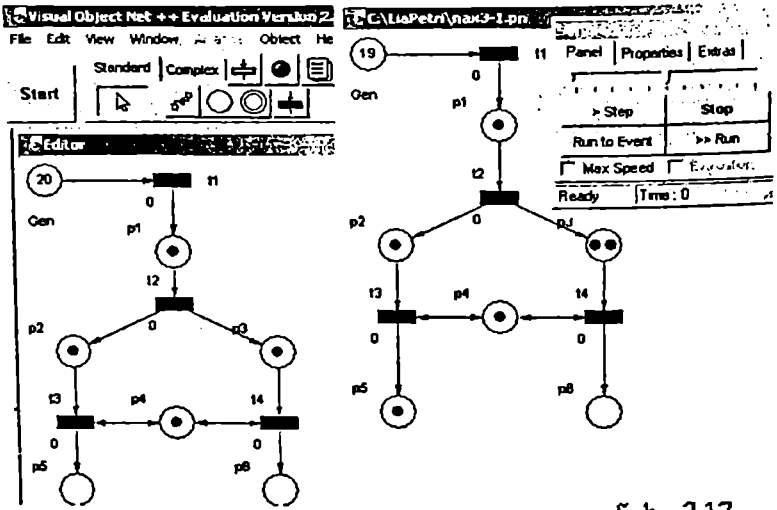
რომ დავალაგოთ შემოსვლის და გასვლის ხდომილებები, საჭიროა შემოვიტანოთ ხდომილებათა ჩანაწერები, რომელიც ექვემდებარება შემდეგ დამუშავებას უნდა მოხდეს მომდევნო ხდომილების დადგომის მომენტის დაფიქსირება და აგრეთვე გასვლის.

სისტემის მდგომარეობის ცვლილება შეიძლება განვიხილოთ ორ ასპექტში:

1) როგორც პროცესი რომელშიც კლიენტს ემსახურებიან (კლიენტის თვალსაზრისი) და 2) როგორც ხდომილებათა თანამიმდევრობა, რომელიც იწვევს მოლარის მდგომარეობის ცვლილებას (მოლარის თვალსაზრისი).



ნახ.3.16



ნახ. 3.17

ნახ. 3.17 წაემოდგენილი პეტრის ქსელი საშუალებას გვაძლევს შევარჩიოთ მომსახურების ისეთი ოპტიმალური მდგომარება, როდესაც კლიენტთა მომსახურება შესაძლებელია მინიმალური დროის განმავლობაში.

თავი IV

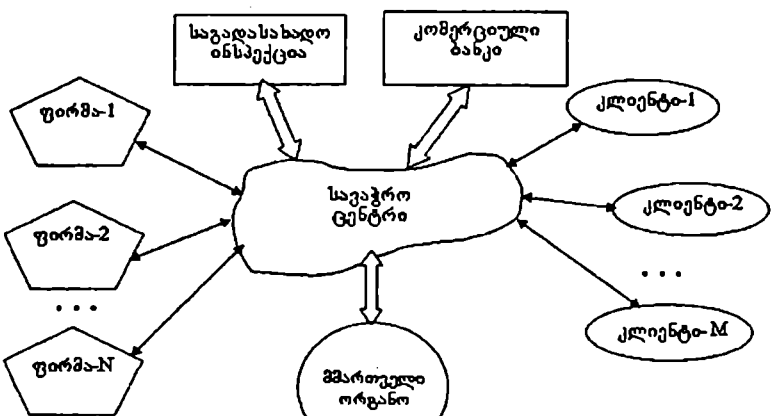
განაწილებული სისტემების მონაცემთა საცავის პროგრამული და ტექნიკური უზრუნველყოფა

4.1. პროგრამული სისტემის არქიტექტურა და ინფორმაციული ნაკადები

ელექტრონული ბიზნესისა და კომერციის განაწილებული სისტემების დაპროექტების ტექნიკური რეალიზაციის მხარე მოითხოვს მისი ცალკეული კვანძების ფუნქციური ანალიზის საფუძველზე აპარატული და პროგრამული უზრუნველყოფების ამოცანების გადაწყვეტას.

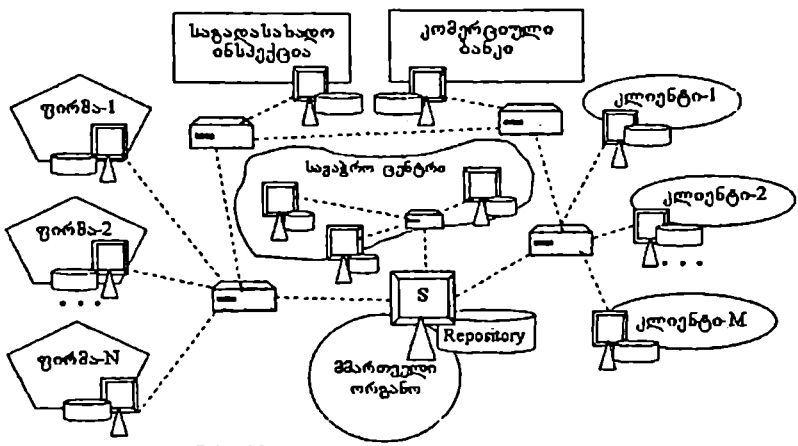
აპარატულში იგულისხმება კომპიუტერული და ქსელური ტექნიკა, რომლის საფუძველზედაც უნდა მოხდეს სისტემის გლობალურ/ლოკალურ ქსელში ფიზიკურად ჩართვის ორგანიზება.

პროგრამული წარმოადგენს ქსელში ჩართული ოპერაციული სისტემის, პლატფორმის, საერთო-სერვისული გარემოს და კერძო-ფუნქციური პაკეტების ერთობლიობას.



ნახ.4.1-ა. ტრადიციული კომერციის ზოგადი სტრუქტურა

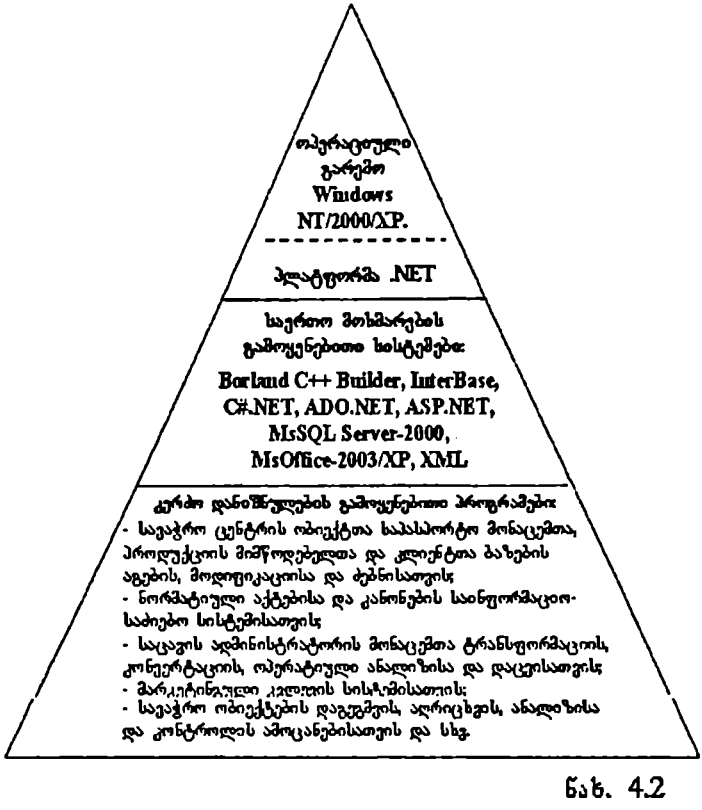
4.1 ნახაზზე მოცემულია კომერციული ბიზნესის ტრადიციული და ელექტრონული კომერციის ზოგადი სტრუქტურული სქემები. კომპიუტერული ტექნიკისა და საინფორმაციო ტექნოლოგიების საფუძველზე რეალიზებულია თვისობრივად ახალი ურთიერთობები სავაჭრო ცენტრებს (მაღაზიათა ქსელი), პროდუქციის მიმწოდებელ ფირმებს, მაკონტროლებელ ორგანოებს, კომერციულ ბანკებსა და კლიენტებს შორის. განაწილებული კომერციული სისტემის მონაცემთა საცავის არსებობა ხელს უწყობს აღნიშნულ ობიექტებს შორის ინფორმაციის დროულად გაცვლისა და გამჭვირვალე პროცესების მართვის უზრუნველყოფას. ახალი სისტემის პირობებში საჭირო ხდება საცავების შესაბამისი აპარატული და პროგრამული დაცვის მექანიზმების გამოყენება.



ნახ.4.1-ბ. ელექტრონული კომერციის ზოგადი სტრუქტურა

4.2 ნახაზზე მოცემული გვაქვს ჩვენი სისტემის ფუნქციონირებისათვის აუცილებელი პროგრამული პლატფორმისა და პაკეტების არქიტექტურა. სისტემის მუშაობის პირობები უნდა აკმაყოფილებდეს შემდეგ ძირითად მოთხოვნებს:

- სისტემის ფუნქციონირება უნდა მოხდეს პერსონალური კომპიუტერების ლოკალურ ქსელში ინტერნეტის გამოყენებით;



ნახ. 4.2

- სისტემამ უნდა უზრუნველყოს ინფორმაციის უსაფრთხოება და დაცვა;

- სისტემას უნდა ჰქონდეს მომხმარებლებთან ურთიერთობის მეგობრული ინტერფეისი;

- სისტემის მომსახურება უნდა იყოს ადვილი.

დიდ საჯარო ცენტრებში (მაღაზიათა ქსელი) ელექტრონული კომერციის განსახორციელებლად საჭიროა შიგა და გარე ინფორმაციული ნაკადების ანალიზის ჩატარება. არსებობს შემდეგი სახის ინფორმაციული ნაკადები:

- კორესპოდენცია და წერილები;
- ნორმატიული აქტები და კანონები;
- კონტრაქტები (ხანგრძლივი) და შეკვეთები (ერთჯერადი);
- ფაქტურები;
- სავაჭრო ობიექტების საქონელბრუნვის გეგმები, ფაქტობრივი შესრულებები, ანალიზის მასალები;
- ინტერნეტიდან მიღებული ინფორმაცია (ფურცლები);
- აუდიო და ვიდეო ინფორმაცია (ელექტრონული გამოფენები, სალონები, პროდუქციის კატალოგები);
- საბანკო ანგარიშები, ბუღალტრული აღრიცხვა;
- კადრების აღრიცხვისა და შრომითი დასაქმების დოკუმენტაცია;
- ინფორმაცია პარტნიორებისა და კონკურენტების შესახებ;
- ინფორმაცია პროდუქციის ადგილობრივი და საერთაშორისო ბაზრების შესახებ (კონიუნქტურა, ფასები);
- სტატისტიკური ანალიზის მასალები. და სხვ.

ინფორმაციული ნაკადების მოცულობათა საანგარიშოდ ინფორმაციის ერთეულად მივიღოთ: $I_{_}$: ერთი ნაბეჭდი A4 ფორმატის ფურცლის ტექსტური ინფორმაციის სიდიდე; I_A: ერთი აუდიო ინფორმაციის სიდიდე; I_V: ერთი ვიდეო ინფორმაციის სიდიდე; პირობითად მივიღოთ, რომ I_T=4 Kb, I_A=20 Kb, I_V=30 Kb. ინფორმაციული ნაკადების ზომები დამოკიდებული იქნება კომერციული ობიექტების მასშტაბებზე (ზომები, კონიუნქტურა, წლიური ფონდბრუნვა და საქონელბრუნვა, ფილიალების რაოდენობა, სეზონი, რეგიონი და ა.შ.). ინფორმაციული ნაკადების მოცულობების საანგარიშოდ შეიძლება ჩავატაროთ მიახლოებითი, გასაშუალებული გათვლები (თვის, კვარტლის, წლის და ხანგრძლივი პერიოდისთვის), რომელთა საფუძველზე შესაძლებელი იქნება საერთო ინფორმაციული

ფონდის მოცულობის შეფასება და მონაცემთა განაწილებული საცავის ფიზიკური მოწყობილობების საჭირო მეხსიერების დადგენა.

4.1 ცხრილში განიხილება ერთი პირობითი კომერციული ობიექტის ინფორმაციული ნაკადების მოცულობები (საშუალოდ). სავაჭრო ცენტრებისათვის ის გამრავლდება, შესაბამისად ფუნქციური ფილიალების რაოდენობაზე და გამოაკლდება საერთო გამოყენების ინფორმაციის რაოდენობა.

საანგარიშო ფორმულებად ვიყენებთ შემდეგ გამოსახულებებს:

$$V_{jT} = k_j * \sum_{i=1}^{n} R_i * I_T \text{ , სადაც}$$

V_{jT} არის ტექსტური სახის ინფორმაციის თვიური, კვარტალური და წლიური დოკუმენტების ჯამური მოცულობა მეგაბაიტებში;

k_j – თვიური, კვარტალური და წლიური კოეფიციენტი (1,3,12);

R_i – ტექსტური დოკუმენტის A4-ფურცლების რაოდენობა;

აუდიო ინფორმაციული ნაკადებისათვის შესაბამისად გვექნება:

$$V_{jA} = k_j * \sum_{i=1}^{m} A_i \text{ , სადაც}$$

A_i – აუდიო ინფორმაციის ფაილის ზომაა;

საჭიროა გავითვალისწინოთ ხმის გადაცემის მახასიათებელი, რომელიც საშუალოდ წარმოშობს 64 Kbit/sec წარმადობის ინფორმაციულ ნაკადებს.

ვიზუალურისათვის შესაბამისად გვექნება:

$$V_{jV} = k_j * \sum_{i=1}^{m} V_i \text{ , სადაც}$$

V_i – ვიდეო ინფორმაციის ფაილის ზომაა;

საჭიროა გავითვალისწინოთ, რომ ვიდეო გამოსახულების გადაცემა არქივირების გარეშე წარმოშობს 9.216 Mbit/sec, ხოლო არქივირებით 1.5 Mbit/sec წარმადობის ინფორმაციულ ნაკადებს.

მთლიანად ინფორმაციული ნაკადების ჯამური მოცულობა იქნება:

$$S = T_i * K_i * \sum_{j=1}^{n} V_j^i \text{, სადაც}$$

T_i - i-ური კომერციული ობიექტის არსებობის მთლიანი პერიოდია (წლები) ;

K_i - i-ურ კომერციულ ობიექტზე ფილიალების რაოდენობაა;

V_j^i - i-ური კომერციული ობიექტის j-ური სახის ინოფორმაციული ნაკადის მოცულობა.

ექსპერტული ინფორმაციის საფუძველზე, როგორც ჩვენი პირობითი გათვლებიდან გამომდინარეობს, ერთ (მცირე ბიზნესის) კომერციულ ობიექტზე დაახლოებით 10 წლიანი არსებობის პერიოდში მონაცემთა საცავისათვის საშუალოდ დაგვჭირდება 200 GB–იანი მეხსიერება (გაითვალისწინება ძირითადი სტატისტიკური და ისტორიული ფაილების შენახვაც).

ინფორმაციული ნაკადების მოცულობის მახასიათებლები (ცხრ.4.1)

№	ინფორმაციული ნაკადი	სახე	მოცულობა (საშუალოდ)			
			თვეში Mb	კვარტ. Mb	წელიწად Mb	მაქსიმალური, გიგაბაიტი-Gb
1.	კორესპონდენცია და წერილები	T	8	24	288	5
2.	ნორმატიული აქტები და კანონები	T			200	3
3.	კონტრაქტები და შეკვეთები	T	16	48	576	10
4.	ფაქტურები	T	20	60	720	12
5.	საკონსალტინგო გეგმები, ფაქტობრივი შესრულებები, ანალიზის მასალები	T			400	6
6.	ინტერნეტიდან მიღებული ინფორმაცია (ფურცლები)	T	20	60	720	12
7.	აუდიო და ვიდეო ინფორმაცია (ელექტრონული გამოფენები, სალონები, პროდუქციის კატალოგები)	A, V	50	150	1800	30
8.	საბანკო ანგარიშები, ბუღალტრული აღრიცხვა	T	20	60	720	12
9.	კადრების აღრიცხვისა და შრომითი დასაქმების დოკუმენტაცია	T			10	2
10	ინფორმაცია პარტნიორებისა და კონკურენტების შესახებ	T	5	15	180	3
11.	ინფორმაცია პროდუქციის ადგილობრივი და საერთაშორისო ბაზრების შესახებ (კონიუნქტურა, ფასები)	T	15	45	540	10

შნიშვნა: ცხრილში ინფორმაცია აღებულია ექსპერტული შეფასებების საფუძველზე.

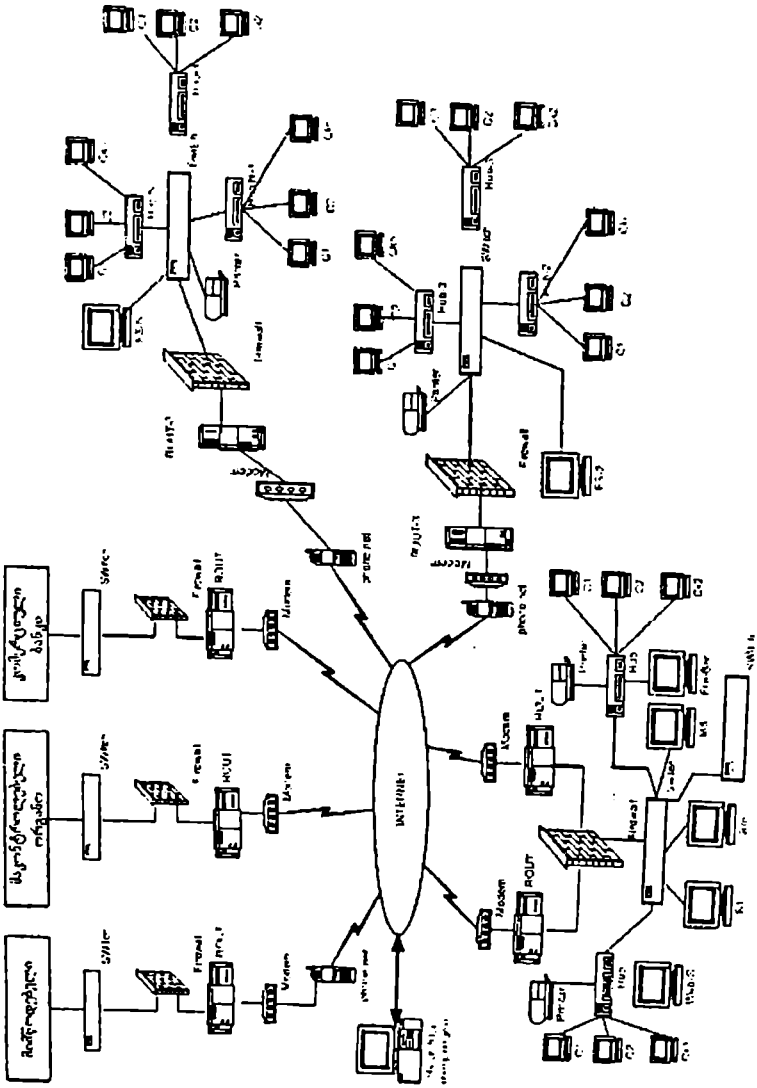
დიდი სავაჭრო ცენტრისათვის (მაღაზიების ქსელი) ეს რიცხვი უნდა გამრავლდეს ობიექტების რაოდენობაზე და დამატოს მათი მენეჯმენტისათვის საჭირო ინფორმაციული ნაკადი. ამგვარად, მონაცემთა საცავისათვის საჭირო ფიზიკური მეხსიერება მიაღწევს რამდენიმე ტერაბაიტ-მოცულობას.

4.2. ქსელური არქიტექტურა და მონაცემთა საცავის დაცვისა და უსაფრთხო გამოყენების მექანიზმები

4.3 ნახაზზე მოცემულია დიდი კომერციული ობიექტის (მაღაზიათა ქსელი) ზოგადი ინტერნეტ/ინტრანეტ ქსელური სქემა. ჩვენ აქ შევეხებით სისტემასთან უსაფრთხო მუშაობისა და ინფორმაციის დაცვის ორგანიზაციულ და ტექნიკურ საკითხებს. კომერციული ობიექტების ქსელური კომპიუტერული სისტემა ორიენტირებულია მრავალ-მომხმარებლურ რეჟიმში სამუშაოდ, რაც აუცილებლად მოითხოვს სისტემის ფუნქციონირების უსაფრთხოების გარანტიებს მაიკროსოფტის Windows Xp/2000, SQL Server-2000 Borland C++Builder, Interbase და სხვა ქსელური ბაზების სისტემებს, გააჩნია სპეციალური პაროლური სისტემები მომხმარებლების დასარეგისტრირებლად (Login ID).

ასევე მნიშვნელოვანია სისტემის მომხმარებელთა ლოგიკური როლების განაწილების საკითხი. აქ წყდება მომხმარებელთა მიკუთვნება რომელიმე წინასწარ განსაზღვრულ როლზე, ხოლო როლს გააჩნია მონაცემთა ბაზაში ინფორმაციის წვდომის ფარგლები (შეზღუდვებით). მაგალითად, მომხმარებელი შეიძლება იყოს სისტემური ადმინისტრატორი, მონაცემთა ბაზის ადმინისტრატორი ან საბოლოო მომხმარებელი. რა თქმა უნდა, ასეთი კატეგორიის მომხმარებლები სხვადასხვა პრიორიტეტებით ისარგებლებენ.

მომხმარებლებს, მათი როლების შესაბამისად, განესაზღვრებათ ბაზიდან მონაცემთა მხოლოდ წაკითხვის და/ან წაკითხვა-ჩაწერის პრიორიტეტებიც.



ნახ.4.3

შეიძლება ითქვას, რომ მონაცემთა დაცვის მექანიზმები კარგადაა დამუშავებული განაწილებული რელაციური მონაცემთა ბაზების მართვის თანამედროვე სისტემებში. იმისდა მიხედვით, თუ რომელი პროგრამული პაკეტი იქნება არჩეული (MS SQL Server, Access, InterBase ან სხვ.) სისტემის სარეალიზაციოდ, მოხდება შესაბამისი უსაფრთხოების სისტემის გამოყენება.

კომერციული ობიექტის კომპიუტერული სისტემა ქსელური მოხმარების სისტემაა, ამიტომაც მრავალმომხმარებლური რეჟიმიდან გამომდინარე საჭიროა სისტემის ექსპლუატაციის დროს გარკვეული რეგლამენტის შემუშავება მონაცემთა დაცვის თვალსაზრისით.

აქ პირველ რიგში იგულისხმება სისტემის საიმედოობის გაზრდა (როგორც პროგრამული პაკეტების, ასევე მონაცემთა ბაზების ფაილებისათვის). ერთის მხრივ, უსაფრთხო მუშაობის პრინციპი (პაროლური სისტემა, როლები და ა.შ.) გარკვეულად ამცირებს არაავტორიზებულ მიმართვებს მონაცემთა ბაზებთან, მაგრამ მეორეს მხრივ, აუცილებელია არსებობდეს არაკორექტული მონაცემების დამახსოვრების თავიდან აცილების შესაძლებლობა.

ინფორმაციის არაკორექტულობა შეიძლება გამოწვეული იყოს შემთხვევით, მონაცემების ან პროგრამის შეცდომით ჩაწერისას ბაზაში ან წინასწარი განზრახვით. ამგვარად, ინფორმაციის დაცვა ორი ამოცანის გადაწყვეტას ითხოვს: მონაცემთა მთლიანობის უზრუნველყოფას და საიდუმლოების გარანტიას (მონაცემთა მისაღებად შეზღუდვების დაყენებას).

მონაცემთა ბაზის მთლიანობის უზრუნველსაყოფად გამოიყენება სტრუქტურული შეზღუდვები ან უშუალოდ მონაცემთა მნიშვნელობების შეზღუდვები.

შეზღუდვები სტრუქტურულ დონეზე ეფუძნება მონაცემთა ბაზებში ფუნქციონალური დამოკიდებულებების (რელაციების, ატრიბუტების და ა.შ.) აღწერას. შემოიტანება სპეციალური გასაღებური ატრიბუტების, ინდექსების ცნებები (მარტივი ან შედგენილი). მათი საშუალებით ხორციელდება რელაციურ ფაილებში ინფორმაციის მოწესრიგება, ძებნა და ამორჩევა.

მთლიანობის შეზღუდვები მონაცემთა მნიშვნელობების ცვლილებებზე ხორციელდება სპეციალური მათემატიკური

დამოკიდებულებებით. თუ მონაცემთა მნიშვნელობა გამოვა განსაზღვრული დიაპაზონიდან, ან არ შეესაბამება არსებულ მათემატიკურ დამოკიდებულებას, მაშინ ხდება სპეციალური დამცველი ფინქციების ამუშავება, რათა არ დაირღვეს ბაზის მთლიანობა. ასეთი ფუნქციის როლს თანამედროვე მონაცემთა ბაზების მართვის სისტემებში ტრიგერები (Triggers) ასრულებს. ესაა ჩართვა-გამორთვის ფუნქციები, რომლებიც უზრუნველყოფს მონაცემთა მთლიანობას ლოგიკურად დაკავშირებულ ცხრილებში (რელაციებში). სისტემაში სტანდარტული ან კერძო ფუნქციის ამუშავება განისაზღვრება მომხმარებლის მიერ, ტრიგერები კი არაა დამოკიდებული პროგრამაზე. იგი ჩაირთვება ყოველთვის, როდესაც ადგილი აქვს მონაცემთა ბაზაში ინფორმაციის განახლებას: ახლის ჩამატებას, ძველის წაშლას ან შეცვლას. ამგვარად, ტრიგერების ერთ-ერთი მთავარი ფუნქციაა სისტემაში მონაცემთა ცვლილების სტატისტიკის წარმოება.

tempdb.mdf და tempdblog.ldf - ბაზაში ინახება დროებითი ცხრილები. იგი SQL Server-ის გლობალური რესურსია. მომხმარებლის მიერთებისას SQL Server-თან ყოველთვის იხსნება ეს ბაზა, მუშაობის დამთავრებისას კი იგი ავტომატურად წაიშლება.

მონაცემთა საიმედოობის უზრუნველსაყოფად კოლექტიური მოხმარების კომპიუტერულ სისტემებში, სადაც მაღალია მონაცემთა დაკარგვის ან დამახინჯების ალბათობა (რისკი), გამოიყენებენ, როგორც ზემოთ აღვნიშნეთ, მონაცემთა ბაზების პერიოდულ დუბლირებას. მაგალითად, რეგლამენტით დადგინდება, რომ ყოველი დღის ბოლოს (ან კვირის ბოლოს, ეს განისაზღვრება ორგანიზაციის ხელმძღვანელობის მიერ) მოხდეს არსებული მონაცემთა ბაზების არქივირება და შენახვა. თუ მომდევნო დღეს (კვირას) მოხდა ინფორმაციის დაკარგვა ან დაზიანება, მაშინ არსებული არქივირებული ფაილიდან მოხდება წინა დღის (კვირის) ინფორმაციის აღდგენა. ეს კი ნიშნავს, რომ დაიკარგება მხოლოდ ბოლო დღის (კვირის) მონაცემები, რაც უკეთესია, ვიდრე საერთოდ დაკარგვა.

არქივირების პროგრამები, როგორიცაა მაგალითად, Winzip, Winrar და სხვა პაკეტები, ასრულებს უნივერსალურ ფუნქციას. ასევე შეიძლება მონაცემთა ბაზების სისტემებსაც ჰქონდეთ სპეციალური არქივატორები.(Backup-შეკუმშვა, Pestwe-გახსნა).

4.3. ანტივირუსული აპარატულ-პროგრამული საშუალებანი

ინტერნეტში საიმედო და უსაფრთხო მუშაობისათვის აუცილებელია სპეციალური აპარატული და პროგრამული მექანიზმების გამოყენება ვირუსების წინააღმდეგ. როგორც ცნობილია, ამ სფეროს განსაკუთრებული ყურადღება ექცევა [69].

ჩნდება სურვილი, რომ ლოკალურ საოფისე ქსელსა და ვებ ბრაუზერს შორის მოხდეს გადატიხვრა, ისე რომ ლოკალურ ქსელში ვერ შეძლონ არავტორიზებული შეღწევა.

იმისათვის, რომ საიმედოდ გადავტიხროთ ჩვენი საოფისე ქსელი და დავიცვათ გარე „მავნე“ ქსელისგან გამოიყენება პროგრამულ–აპარატული კომპლექსი, რომელიც ემსახურება დამცავი ეკრანის შექმნას შიგა ლოკალურ საოფისე ქსელსა და ინტერნეტის საშიშ ქსელს შორის. აუცილებელია აგრეთვე პროგრამული ანტივირუსების გამოყენებაც და მათი ხშირად განახლება. ბრანდმაუერები (Firewall) ქმნის ვირუსებისთვის „ცეცხლოვან კედელს“ და ისინი ვერ აღწევს მომხმარებელთა კომპიუტერებამდე.

4.3.1. Firewall –ქსელური ეკრანი

ქსელური ეკრანის მუშაობის ყველაზე მარტივი სქემა გამოიყურება შემდეგნაირად: ყველა ქსელი იყოფა „გარე“ (რომლისგანაც ვიცავთ თავს) და „შიგა“ (რომელსაც ვიცავთ) ქსელისგან. თუ გარე ქსელისგან წამოვა მოთხოვნა, რომელიც მოითხოვს შიგა კომპიუტერულ ქსელში ჩართვას უმეტეს შემთხვევაში firewall (ქსელის დამცავი მოწყობილობა) ამ მოთხოვნას უარყოფს, მხოლოდ იმ შემთხვევაში თუ შიგა ქსელიდან წამოვა მოთხვნა, რომელიც ითხოვს გარე ქსელთან დაკავშირებას, (მომხმარებელი ცდილობს ბრაუზერში რომელიმე საიტზე გვერდის

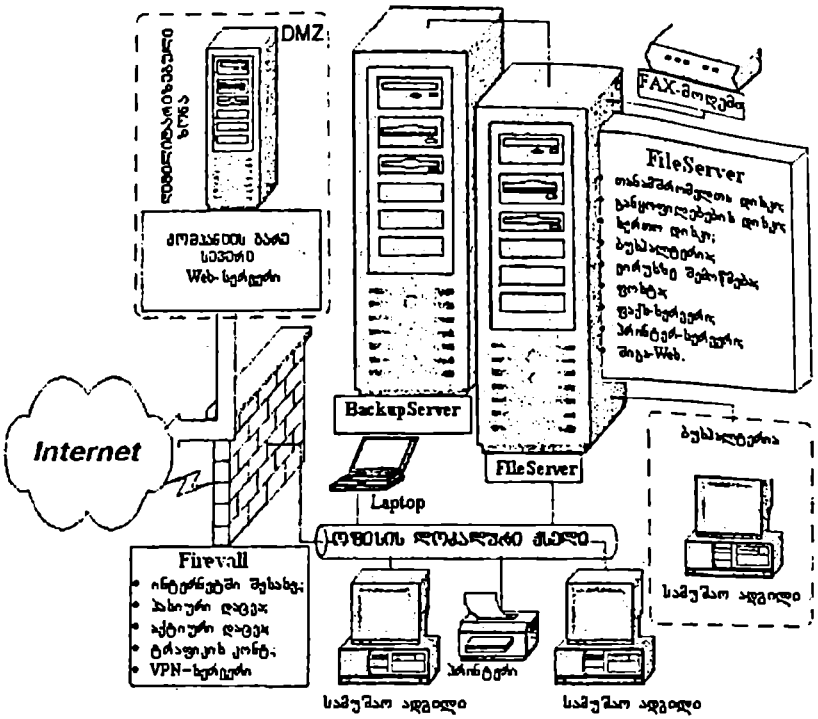
გახსნას) მაშინ firewall ამოწმებს, მომხმარებელს აქვს თუ არა უფლება დაუკავშირდეს ამ კონკრეტულ საიტს [71].

ზოგადად Firewall -ს აქვს შემდეგი ფუნქციები:

1. ახდენს ინტერნეტის ანალიზს;
2. უკრძალავს გარედან შემოსულ მოთხოვნებს (თანხმობის გარეშე) შიგა ქსელთან დაკავშირებას.
3. არეგულირებს ურთიერთობას შიგა ლოკალურ საოფისე ქსელსა და გარე ქსელს შორის;
4. ამოწმებს ვირუსზე ისეთ მომსახურებას, როგორიცაა საფოსტო ფუნქციები;
5. წარმოადგენს სტატისტიკას, რომელიც ასახავს გარე ქსელიდან უარყოფილ და მიღებულ მოთხოვნებს;
6. ასახავს შიგა მომხმარებელთა სტატისტიკას (ვინ, სად, როდის და რამდენჯერ შევიდა ინტერნეტში);
7. ახორციელებს სისტემურ ადმინისტრატორთან ავტომატურ შეტყობინებას, რომელიც ასახავს მნიშვნელოვან მოვლენებს;
8. ახორციელებს ავტომატურ დაცვა-დამისამართების ბლოკირებას.
9. უზრუნველყოფს საკუთარი პროგრამული პაკეტის მთლიანობის დაცვას;
10. გამოავლენს უჩვეულო და საშიშ პროცესებს შიგა მოვლენების ავტომატური ანალიზის საფუძველზე.
11. უზრუნველყოფს ინტერნეტ ქსელთან დაშიფრულ დაკავშირებას.

4.3.2. DMZ –დემილიტარიზებული ზონა

შიგა და გარე ქსელის გარდა არსებოს ქსელი რომელსაც "DMZ" დემილიტარიზებული ზონა ეწოდება [71]. DMZ-ს ვიყენებთ იმ შემთხვევაში, როდესაც სავაჭრო ცენტრს ინტერნეტში აქვს თავის გარე ქსელი (ვებ-სერვერი) და იგი საჭიროებს დაცვას. ამ შემთხვევაში ეს სერვერი თავსდება DMZ-დემილიტარიზებულ ზონაში, სადაც იგი გაცილებით დაცულია გარე „შემოტევებისგან". ამასთან კომპანიის შიგა ქსელი მისთვის რჩება „უცხოდ".



ნახ.4.4. სავაჭრო ცენტრის გარე ქსელი დემილიტარიზებულ ზონაში.

4.3.3. ოპტიმიზაციის სიტემა და ინტერნეტ-ტრაფიკზე კონტროლი (Proxy-სერვერი)

ინტერნეტში მუშაობის დროს ხშირად არის შემთხვევევა, როდესაც რამდენიმე თანამშრომელი მუშაობს ერთსა და იმავე დოკუმენტთან ან შედიან ერთსა და იმავე საიტზე. ასეთი შემთხვევებისთვის იყენებენ ოპტიმიზაციის ინტერნეტ-ტრაფიკ სისტემას. ეს სისტემა ყველა ტრაფიკს უშვებს თავის- თავის გავლით და ინახავს მათ ასლებს დისკზე. იმ შემთხვევაში თუ მომხმარებელს ექნება სურვილი განმეორებით გამოიძახოს გვერდი, მაშინ მას აღარ დასჭირდება ინტერნეტის გამოძახება, პირდაპირ დაუკავშირდება ოპტიმიზაციის ინტერნეტ-ტრაფიკ სისტემურ დისკს. ეს პროცესი გაცილებით ეფექტურია (მიმდინარეობს სწრაფად) და იაფი. ამ სისტემას უწოდებენ პროქსი-სერვერს (Proxy) [71]. Proxy-სერვერი არეგულირებს კომპანიის თანამშრომლებს შორის ინტერნეტ-რესურსების გადანაწილებას. მას გააჩნია მოქნილი მოწყობილობა, რომელიც განსაზღვრავს მომხმარებელთა ურთიერთობას კონკრეტულ საიტთან, აკონტროლებს ინტერნეტიდან გადმოტვირთულ ინფორმაციის მოცულობას და უზრუნველყოფს ამა თუ იმ ინტერნეტ მომხმარებლის მუშაობას დროის ნებისმიერ მომენტში.

Proxy-სერვერი მომხმარებელს საშუალებას აძლევს არ გამოვიდეს თავის ინტერნეტ მისამართიდან და მოთხოვნის წინ შეცვალოს მისამართი. მოთხოვნა დააყენოს არა როგორც ქსელის რეალური მომხმარებლის სახელით, არამედ თავის პირადი მისამართით. ეს ეხმარება არა მხოლოდ შიგა ქსელის სტრუქტურის დამალვას უსაფრთხოების დაცვის მიზნით, არამედ თავის ინტერნეტ-პროვაიდერს არ მოუწევს თითოეულ თანამშრომლის მისამართზე დამატებითი იჯარის გაცემა, რადგან ფიქსირებული იქნება Proxy-სერვერისთვის ერთადერთი მისამართი.

Proxy-სერვერი ახორციელებს ინტერნეტიდან მონაცემთა კეშირებას (მონაცემთა შენახვა განმეორებითი მოთხოვნისთვის), და განმეორებითი მოთხოვნის დროს იგი პირდაპირ თავის დისკიდან მიაწვდის მომხმარებელს ინფორმაციას. ეს მნიშვნელოვნად აჩქარებს მრავალი გვერდის ჩატვირთვას და რაც მთვარია პროვაიდერს უმცირდება თანხა ინტერნეტიდან ჩატვირთულ გვერდების მოცულობაზე.

4.3.4. ვირტუალური კერძო ქსელი - VPN (Virtual Private Network)

ხშირად ვხვდებით შემთხვევას, როდესაც ფირმას აქვს სხვადასხვა ტერიტორიაზე განთავსებული ობიექტები (მაგალითად, მაღაზიათა ქსელი). ასეთი ფირმების არსებობის შემთხვევაში დგება საკითხი თუ როგორ უნდა დაუკავშირდეს ფილიალები ერთმანეთს ერთ სერთო საინფორმაციო ქსელში. ამასთნ ერთად ეს კავშირი უნდა იყოს მაქსიმალურად მოხერხებული, უსაფრთხო და იაფი.

იმ მიზნით, რომ შეექმნათ ამ პირობების გათვალისწინებით ლოკალური ქსელი ინტერნეტ ქსელის ინფრასტრუქტურის გამოყენებით, უნდა სრულდებოდეს შემდეგი პირობები:

1. გადაწყვეტილება უნდა იყოს უნივერსალური და შეესაბამისობაში მოდიოდეს დანართებთან. ეი ის არ უნდა იყოს დამოკიდებული მხოლოდ ერთ კონკტეტულ პროგრამაზე, არამედ ყველა ფილიათან უდა იყოს შესაბამისობაში და ემსაუხრებოდეს ერთ მიზანს;
2. მომხმარებელს არ უნდა შეხვდეს რაიმე სირთულე ამ ტექნოლოგიის გამოყენებისას;
3. დიდად არ უნდა განსხვავდებოდეს უკვე არსებულ ქსელისგან.
4. ფილიალებს შორის ურთიერთკავშირისთვის უნდა გამოიყენებოდეს უკვე არსებული ინტერნეტ ქსელი ან საკუთარი კავშირის ხაზები;
5. გადაწყვეტილების მიღება არ უნდა უკავშირდებოდეს კონკრეტულ მწარმოებელს, იგი ორიენტირებული უნდა იყოს ღია პროტოკოლზე და სპეციფიკაციაზე;
6. ნებისმიერ შემთხვევაში ძირითადი პროტოკოლი უნდა უკავშირდებოდეს უკვე არსებულ პროტოკოლს, რომელიც ფართოდ გამოიყენება ინტერნეტში- TCP/IP;
7. ქსელებს შორის ინფორმაციული კავშირი უნდა იყოს საიმედოდ დაშიფრული;
8. დაშიფვრის ალგორითმი უნდა იყოს საიმედო, მრავალმხრივად შემოწმებული, ამასთან შესაძლებელი უნდა იყოს მომხმარებელმა თავის შეხედულებისამებრ შეარჩიოს დაშიფვრის ალგორითმი და მისი პარამეტრები;

ამ ტექნოლოგიის გამოყენება საკმაოდ იაფია, რადგან მისი ძირითადი პრინციპია კავშირებისთვის გამოიყენოს ლოკალური

განაწილებული ქსელი ინტერნეტთან ერთად. VPN–ტექნოლოგიის გამოყენების ძირითად არსს წარმოადგენს ყველა არხის შიფრირება, რომელიც ქსელში ინფორმაციის გადაცემის დროს გამოიყენება [71].

ორივე ფილიალში დგას VPN–თან მიერთებული ბრანდმაუერი, რომელიც უკავშირდება ერთმანეთს ინტერნეტით. როდესაც რომელიმე მომხმარებელი ქსელიდან LAN–1 უკავშირდება მეორე LAN–2 ქსელს, მისი მოთხოვნა გაივლის VPN+Firewall–1 ბარიერს. ეს ნიშნავს, რომ LAN–1 დან ბარიერის გავლით გასული მოთხოვნის მონაცემები და მისამართი გაივლის შიფრაციას და ინტერნეტის საშუალებით გადაეცემა ბრანდმაუერს, რომელსაც თავის VPN+Firewall–2 დამცველი ბარიერი აქვს იგივე შემოწმების შემდეგ მიაღწევს LAN–2-თან. ანალოგიური პროცესი ხორციელდება უკუკავშირის დროსაც. მოცემული ტექნოლოგიის დანერგვის შემდეგ უსაფრთხო ხდება ლოკალური ქსელის დახმარებით ფილიალებს შორის ურთიერთ კავშირი და საკმაოდ ამაღლებს კომპანიის მუშაობის წარმადობას.

4.3.5. მობილური კლიენტი

ხშირად არის შემთხვევა, როდესაც რომელიმე ფირმის რამდენიმე თანამშრომელი მივლინებაში და მათთვის აუცილებლობას წარმოადგენს შეუიდნენ თავიანთ კომპანიის შიგა ქსელში, დაათვალიერონ ფირმის განახლებული, „პრაიზ ლისტი“, შიგა ბრძანებები, ანდაც მათ უშუალოდ შეძლონ თანამშრომელთა შემოწმება და ბრძანებების გაცემა. ასეთი პროცესების რეალიზაციის საფუძველს წარმოადგენს პროტოკოლები: ინტერნეტ-პროტოკოლი IPSec და Microsoft ფირმის პროტოკოლი PPTP [71].

4.3.6. განაწილებული ოფისი

ერთ ორგანიზაციას შეიძლება ჰქონდეს რამდენიმე ფილიალი რომელიმე ქალაქში ან უფრო ფართო მასშტაბით, მთელ ქვეყანაში. მაგალითად, ეს შეიძლება იყოს მაღაზიათა ქსელი, რომელთაც ერთი საერთო საწყობი აქვს.

მაღაზიათა ეფექტური მუშაობისათვის აუცილებელია ინფორმაციის ოპერატიული გაცვლა. მაგ., თუ საწყობში არ იქნება

რომელიმე სახის პროდუქცია, ეს უნდა ეცნობოს ყველა ფილიას, რადგან გარკვეული დროის განმავლობაში არ მოხდეს შეკვეთების მიღება. ასევე თუ ოფისის ადმინისტრაცია იღებს რაიმე ბრძანებას, იგი დაუყოვნებლივ ცნობილი უნდა გახდეს ყველა ფილიალისთვის.

ასეთ შემთხვევათა გამო ყველა დიდ ფირმის ხელმძღვანელის წინაშე დგება მოთხოვნა, რათა მოხდეს ყველა ფილიალის ინფორმაციული ქსელის ინტეგრაცია ერთ საერთო ქსელთან.

რამდენიმე ფილიალის ლოკალური ქსელის ერთ ქსელში გაერთიანებისათვის საჭიროა სრულდებოდეს პირობები [71]:

1. გადაწყვეტილება უნდა იყოს უნივერსალური და ეწყობოდეს ყველა დანართსა და ქსელს, მაგრამ არ უნდა იყოს დაკავშირებული რომელიმე კონკრეტულ პროგრამასა და მთელ სისტემაზე;
2. მომხმარებლისთვის უნდა იყოს მარტივი;
3. ფილიალთა ურთიერთკავშირისათვის უნდა გამოვიყენეოთ უკვე არსებული ინტერნეტ კავშირი ან საკუთარი კავშირის ხაზები;
4. გადაწყვეტილება არ უნდა იყოს დაკავშირებული კონკრეტულ მეწარმესთან, სასურველია ეყრდნობოდეს ღია პროტოკოლებსა და მათ სპეციფიკაციას;
5. ნებისმიერ შემთხვევაში ძირითადი დამაკავშირებელი პროტოკოლი უნდა იყოს უკვე არსებული, რომელსაც ინტერნეტ ქსელში ფართო გამოყენება აქვს–TCP/IP;
6. ქსელებს შორის ინფორმაციის გაცვლა არ უნდა ხდებოდს შიფრირებულად;
7. დაშიფვრის ალგორითმი უნდა იყოს საიმედო და სპეციალისტების მიერ მრავალმხრივად შემოწმებული. ამას გარდა მომხმარებელს უნდა შეეძლოს თავის შეხედულებისამებრ შეცვალოს დაშიფვრის ალგორითმი და მისი პარამეტრები.

რამდენიმე წლის წინ IETF (Internet Engineering Task Force)-ორგანიზაცია, რომელიც ამუშავებდა და ამტკიცებდა ინტერნეტ ქსელის ყველა სტანდარტსა და პროტოკოლს, შემოგვთავაზა ახალი პროტოკოლი, რომელიც აკმაყოფილებს ყველა ზემოთ წამოყენებულ მოთხოვნას. პროტოკოლს ჰქვია IPSec. მისი დანიშნულებაა VPN ვირტუალური პერსონალური ქსელის აგება.

მისი გამოყენება ყველაზე ეფექტურია განაწილებულ ოფისებში, იგი ითვლება როგორც ყველაზე საიმედო და მოხერხებული უნივერსალური საშუალება, რომელიც ფილიალებს აკავშირებს ორგანიზაციასთან.

საშუალოდ ყოველი ფილიალის ლოკალური ქსელის კონფიგურაცია აწყობილია ბაზაზე, სადაც მუშაობს 10-150 თანამშრომელი. ყოველ ფილიალში ფირმები აყენებენ ქსელურ HUB-მოწყობილობას, რომელიც ასრულებს შემდეგ ფუნქციებს: იგი უზრუნველყოფს ფილიალების დაკავშირებას ინტერნეტთან, ასრულებს ქსელური ეკრანის(Firewall) ფუნქციას და აგრეთვე მოიცავს VPN ტექნოლოგიას. VPN-ხაბი შედგება რამდენიმე ვირტუალური არხისგან და უზრუნველყოფს გამჭვირვალე კავშირს (მხოლოდ თანამშრომლებისთვის) ყველა ფილიალის ლოკალურ ქსელთან.

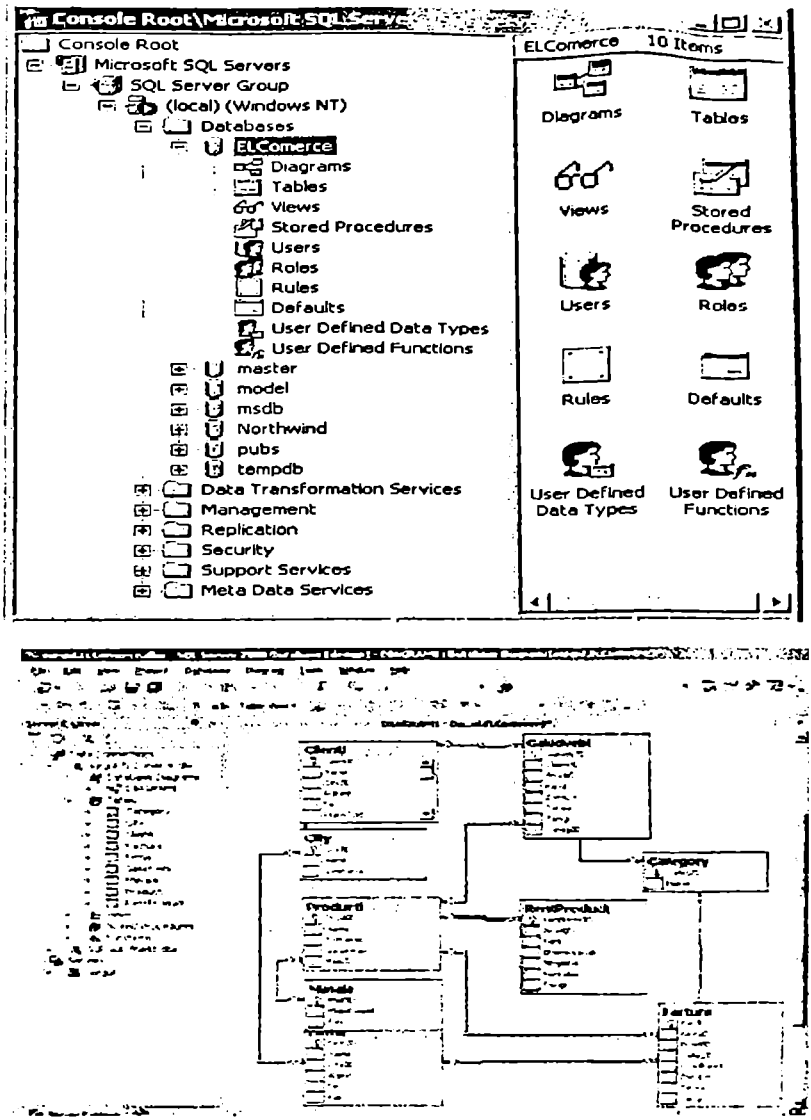
ეს ვირტუალური არხები ყველა ფილიალს აკავშირებს ფირმასთან. ამასთან ერთად ყველა ფილიალში ინახება ქსელის პირვანდელი სტრუქტურა, რადგან ყოველი თანამშრომლისთვის ლოკალური ქსელი ფართოვდება და მოიცავს ფირმის ყველა ფილიალს. ამიტომ ყველა თანამშრომელს შეუძლია იოლად გამოიყენოს ნებისმიერი დოკუმენტი, რომელიც ფირმის ნებისმიერ კომპიუტერშია მოთავსებული. ამასთან არცერთ თანამშრომელს არ შეუძლია წაშალოს კომპანიის რაიმე დოკუმენტი. ყველა მონაცემი გადაიცემა შიფრირებულად და მისი გაშიფრვა შეუძლებელია სპეციალური შიფრატორის გარეშე, რომელზეც მხოლოდ ფირმის ხელმძღვანელს მიუწვდება ხელი.

4.4. SQL Server - პაკეტის გამოყენების ვიზუალური ინსტრუმენტი

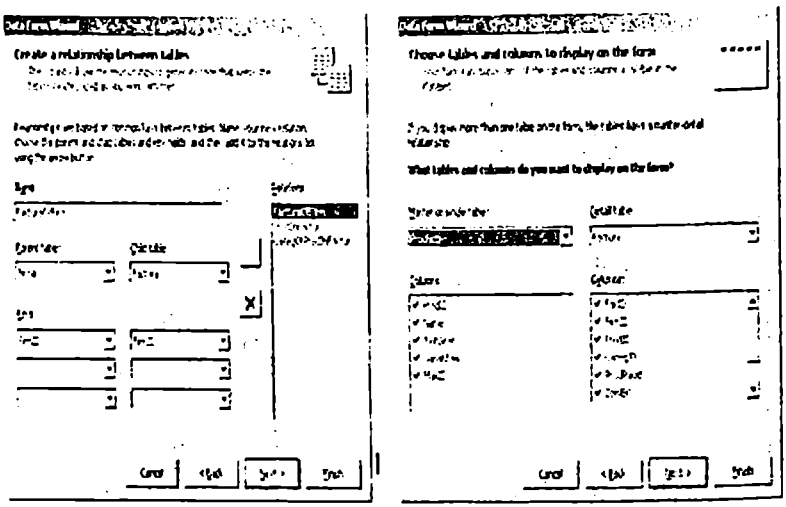
SQL (Struqtured Query Language- სტრუქტურირებული მოთხოვნების ენა) გამოიყენება მონაცემთა ბაზების ცხრილებიდან საჭირო ინფორმაციის ამოსაღებად. ამისათვის იწერება მოთხოვნათა (Query) ფირმა მაიკროსოფტის მიერ დამუშავებულ SQL-ენის სტანდარტის შესაბამისად.

4.5 ნახაზზე მოცემულია ელექტრონული კომერციის სისტემის

(ELCommerce) რეალიზებული მონაცემთა ბაზის ერთ-ერთი ფრაგმენტი MsSQL Server-2000-ის მაგალითზე.



ნახ.4.5



ნახ. 4.6 :LCommerce სისტემის კატალოგი, ცხრილთა კავშირების სტრუქტურა და ფორმაზე გამოსატანი ცხრილების და ველების ამორჩევა

4.5 Ms SQL Server სისტემის არქიტექტურა

MicroSoft ფირმამ Windows-2000 Server პლატფორმაზე შექმნა მონაცემთა განაწილებული რელაციური ბაზების მართვის სისტემა SQL Server. იგი მუშაობს აგრეთვე Unix, Linux, Macintosh და სხვა ოპერაციულ პლატფორმებთანაც, რაც მის უნივერსალურობასა და მოქნილობაზე მეტყველებს[77].

SQL Server მონაცემებთან მიმართვისათვის იყენებს ოთხ ძირითად ინტერფეისს: OLE DB, ODBC, DB Library და Transact-SQL. მომხმარებლისათვის, რომელიც მუშაობს Windows-სისტემასთან, ეს

ინტერფეისები რეალიზებულია დინამიკურად მიერთებადი ბიბლიოთეკის, DLL-ფაილების სახით. Web-კლიენტებისთვის ქსელური ბიბლიოთეკის გამოძახება ხდება IPC (Interprocess Communication) კომპონენტებით.

MsSQL Server შედგება ოთხი ძირითადი კომპონენტისგან:

- Open Data Services SQL Server - უზრუნველყოფს ინტერფეისს ქსელურ ბიბლიოთეკებსა და თვით MSSQL Server-ის ბირთვს შორის;

- MSSQLServer - მართავს მონაცემთა ბაზის ყველა ფაილს, ამუშავებს მომხმარებელთა მოთხოვნებს, ანაწილებს სისტემურ რესურსებს, ამოწმებს მომხმარებელთა საიღრიცხვო ჩანაწერებს;

- SQLServer Agent - ახორციელებს დავალებათა დაგეგმვას და SQLServer მოვლენათა დამუშავების ავტომატიზაციას;

- MSDTC (Micrsoft Distributed Transaction Coordinator) - როგორც განაწილებული ტრანზაქციების კოორდინატორი იგი მართავს მოთხოვნების შესრულებას მონაცემთა ბაზების რამდენიმე სერვერთან. MSDTC სერვისი შეიძლება ამუშავდეს როგორც SQL Server ბირთვიდან, ასევე კლიენტთა გამოყენებითი სისტემიდან.

SQL Server სისტემა მოთხოვნების დამუშავების SQL-ენის ნაცვლად იყენებს Transact-SQL დიალექტს. ეს არის მონაცემთა ბაზის ცხრილების, სვეტების, ჩანაწერების, ტრიგერებისა და შენახვადი პროცედურების შექმნის, მოდიფიკაციისაა და წაშლის ენა. ამ ენის ინსტრუქციები დაყოფილია სამ ქვესიმრალედ: DDL - მონაცემთა ბაზების ცხრილებისა და წარმოდგენების შესაქმნელად, DML- მოთხოვნების შესაქმნელად და მონაცემთა დასამუშავებლად, DCL (Data Control Language) - მონაცემთა ბაზასთან მიმართვის პროცედურების სამართავად. 4.7 ნახაზზე მოცემულია SQL Server-ის არქიტექტურა[21,23].

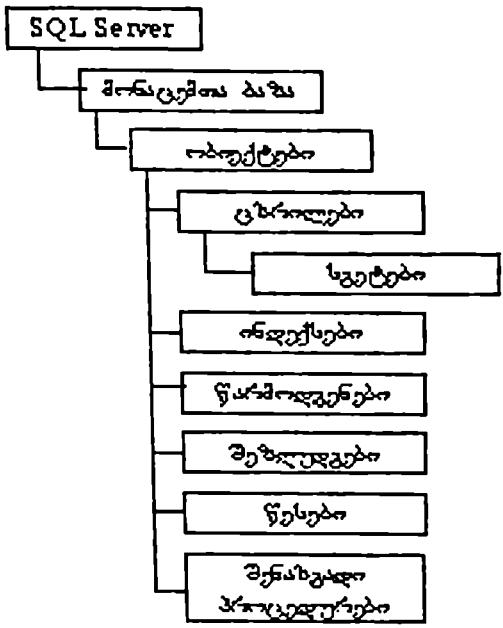
SQL Server სისტემის ინსტალირების დროს იქმნება ოთხი წყვილი სისტემური საბაზო ფაილი:

master.mdf - მონაცემთა ფაილი და mastlog.ldf - ტრანზაქციების ჟურნალის ფაილი. ამ ბაზებში ინახება SQL Server-ის კონფიგურაციისა და ფუნქციონირების შესახებ სრული ინფორმაცია. აქვეა მონაცემები სერვერის პარამეტრების, რეგისტრირებულ მომხმარებელთა და სისტემაში არსებულ სხვა ბაზების შესახებ. model.mdf და modellog.ldf - ეტალონური (შაბლონური) მონაცემთა ბაზა, რომელიც გამოიყენება მომხმარებელთა ახალი ბაზების შესაქმნელად. იგი ავტომატურად

გადასცემს ახალ ბაზას თავის პარამეტრებს (ცვლილებები დასაშვებია).

msdb.mdf და msdblog.ldf - ეს ბაზა შეიცავს ინფორმაციას დავალებების (jobs), მოვლენებისა (alerts) და ოპერატორების (operators) შესრულებათა მიმდევრობების დასაგეგმად.

tempdb.mdf და tempdblog.ldf - ბაზაში ინახება დროებითი ცხრილები. იგი SQL Server-ის გლობალური რესურსია. მომხმარებლის მიერთებისას SQL Server-თან ყოველთვის იხსნება ეს ბაზა, მუშაობის დამთავრებისას კი იგი ავტომატურად წაიშლება.



ნახ.4.7

4.5.1. SQL Server-ის მონაცემთა ბაზის ობიექტები

ახლა განვიხილოთ SQL Server-ის ძირითადი ობიექტები (ნახ.4.5). მათ საილუსტრაციოდ გამოვიყენებთ SQL Server Enterprise Manager უტილიტას (სერვისული პროგრამა).

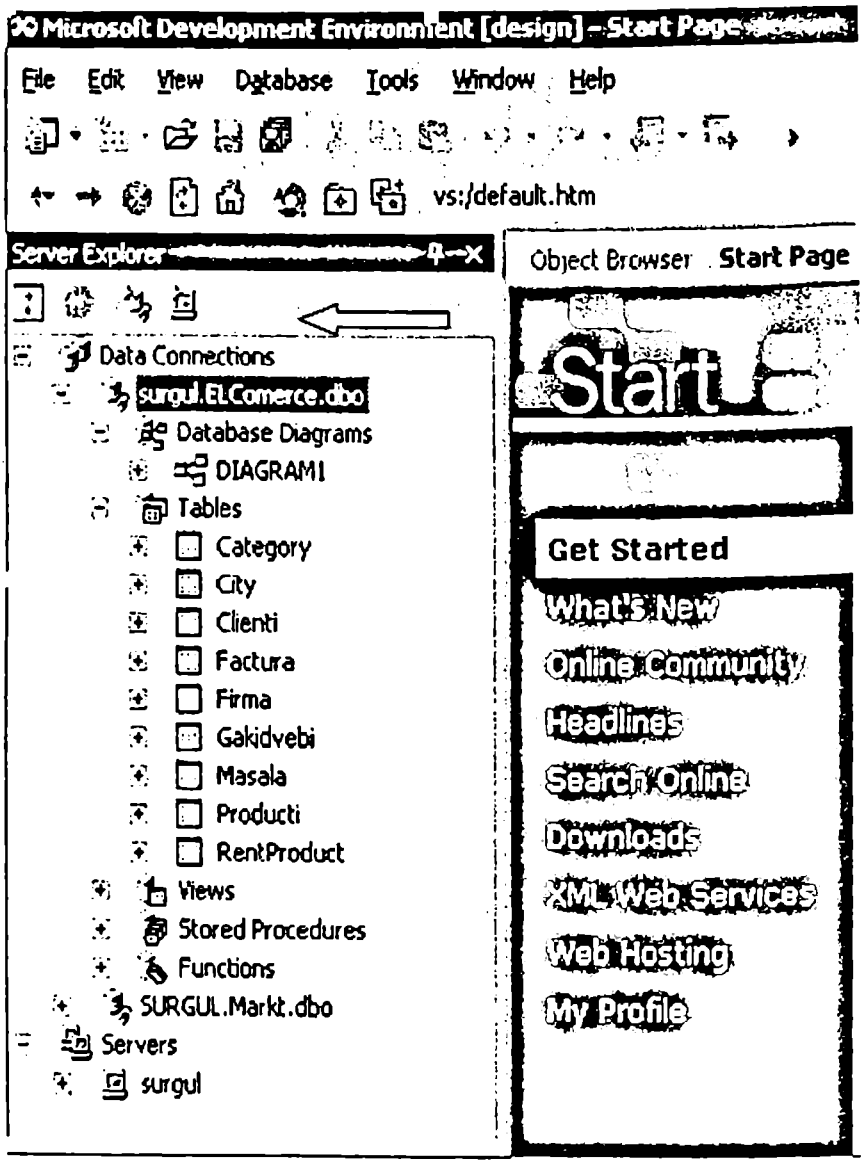
Tables - ცხრილები მონაცემთა ბაზის ჩანაწერების (სტრიქონების) შესანახი კომპონენტია. იგი შედგება სახელმინიჭებული ატრიბუტებისგან (სვეტებისაგან), რომელთაც აქვს განსაზღვრული ტიპი და სიგრძე. SQL Server-ში არსებობს სისტემური და მომხმარებლის ცხრილები. სისტემურის სახელები იწყება sys-ით. 4.8 ნახაზზე ნაჩვენებია ახალი ცხრილის შექმნის ინტერფეისი, ხოლო 4.7-ზე უკვე კონკრეტული მონაცემებით შევსებული "მწარმოებლის" ცხრილის ფრაგმენტი.

Columns - სვეტები, ანუ ცხრილის ველები (Column name), რომელთა მნიშვნელობები შეირჩევა მათი ტიპებისა (Data type) და სიგრძეების (Length) მიხედვით (ნახ.4.7).

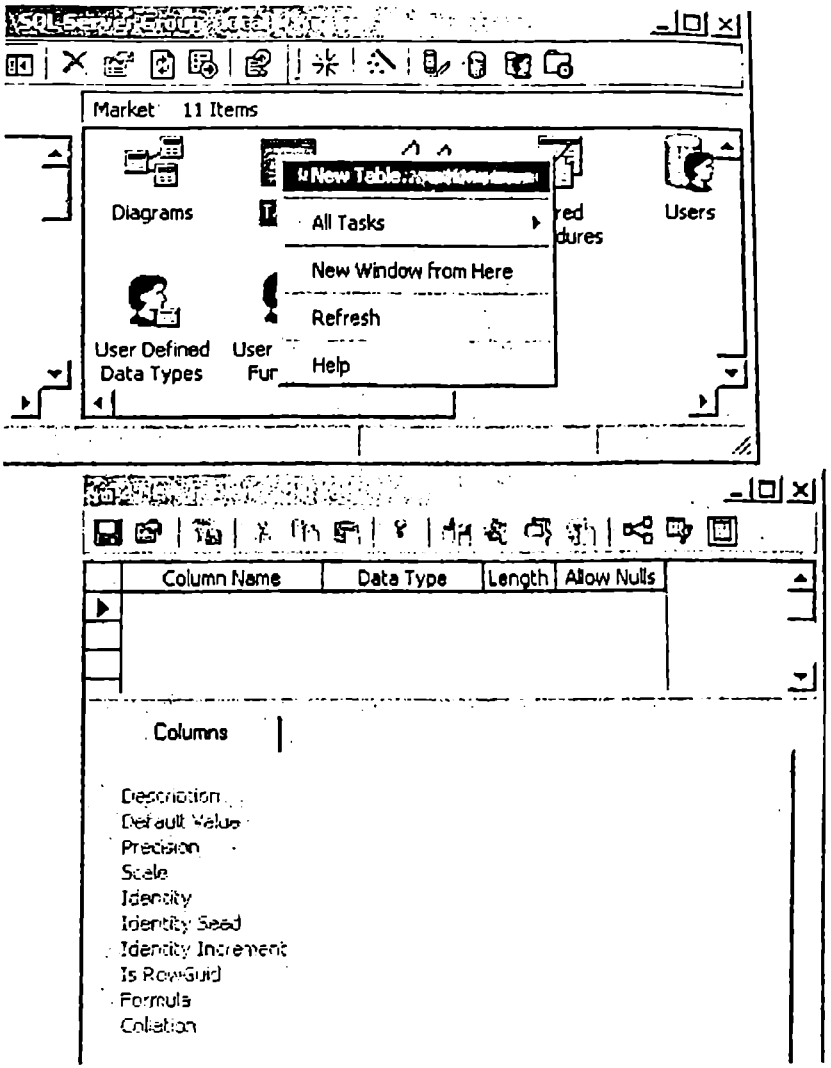
SQL Server-ში მონაცემთა 30-მდე ტიპია. მაგალითად, int - მთელრიცხვა (4 ბაიტით), smallint - მთელრიცხვა (2 ბაიტით), real - ნამდვილრიცხვა (4 ბაიტით), float - ნამდვილრიცხვა (8 ბაიტით), money - ფულადი ტიპი (8 ბაიტით), char[n] - სიმბოლური ფიქსირებული სიგრძით (n ბაიტით), varchar[n] - ცვლადი სიგრძის სიმბოლოები (n ბაიტით) და ა.შ. მონაცემთათვის კონკრეტული ტიპის შერჩევა შესაძლებელია Data type სვეტში "თაგუს" მარჯვენა ღილაკით მენიუს გამოძახებით.

Indexes - ინდექსები გამოიყენება დიდ ცხრილებში მონაცემთა მოწესრიგებული შენახვისათვის და შემდგომ ძებნის ოპერაციების დასაჩქარებლად. ერთ ცხრილს შეიძლება რამდენიმე სხვადასხვა ინდექსი ჰქონდეს. ფიზიკურად ესაა ცალკე ფაილი საკუთარი სახელით, რომელსაც კავშირი აქვს ძირითად ცხრილთან, სადაც მონაცემები ფიზიკურად ინახება. 4.8 და 4.9 ნახაზებზე ნაჩვენებია ეს შემთხვევები.

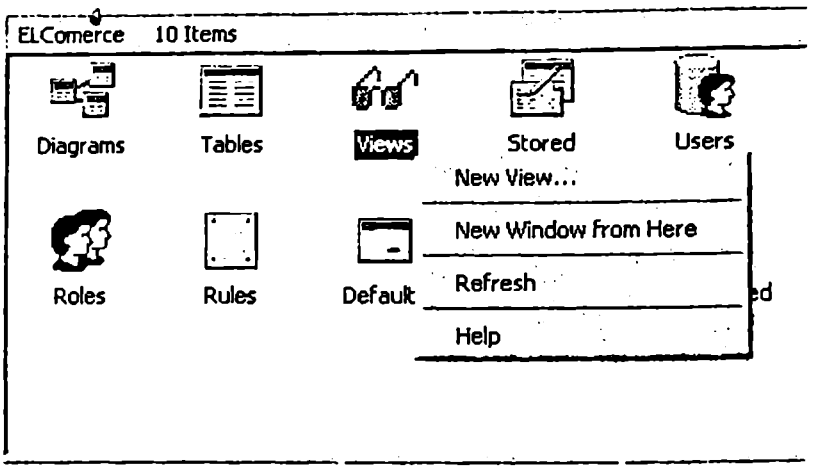
ინდექსი, რომელიც უნიკალურია ცხრილისათვის (ანუ მასში არ ხდება გასაღებური ველის მნიშვნელობის გამეორება), შეიძლება მასში პირველად გასაღებად იქნეს არჩეული. ასეთია ჩვენს შემთხვევაში


Microsoft Development Environment [design] - Start Page
File Edit View Database Tools Window Help
vs:/default.htm
Server Explorer
Object Browser
Start Page
Data Connections
surgul.ELComerce.dbo
Database Diagrams
DIAGRAM1
Tables
Category
City
Clienti
Factura
Firma
Gakidvebi
Masala
Producti
RentProduct
Views
Stored Procedures
Functions
SURGUL.Markt.dbo
Servers
surgul
Start
Get Started
What's New
Online Community
Headlines
Search Online
Downloads
XML Web Services
Web Hosting
My Profile

ნახ. 4.8



6.5.6.4.9. New Table


ELComerce 10 Items
Diagrams
Tables
Views
Stored
Users
Roles
Rules
Default
New View...
New Window from Here
Refresh
Help

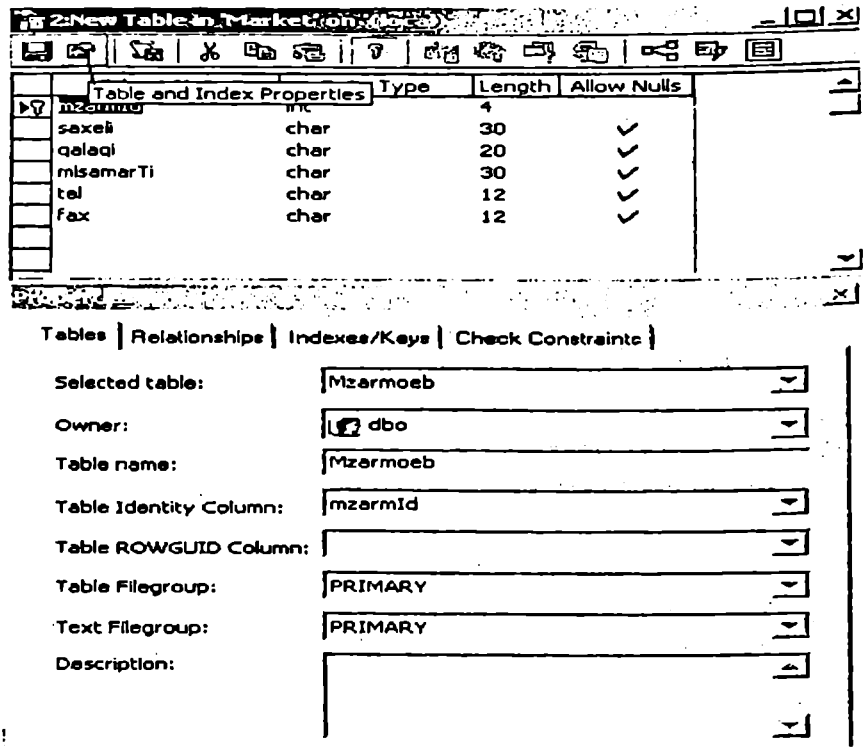
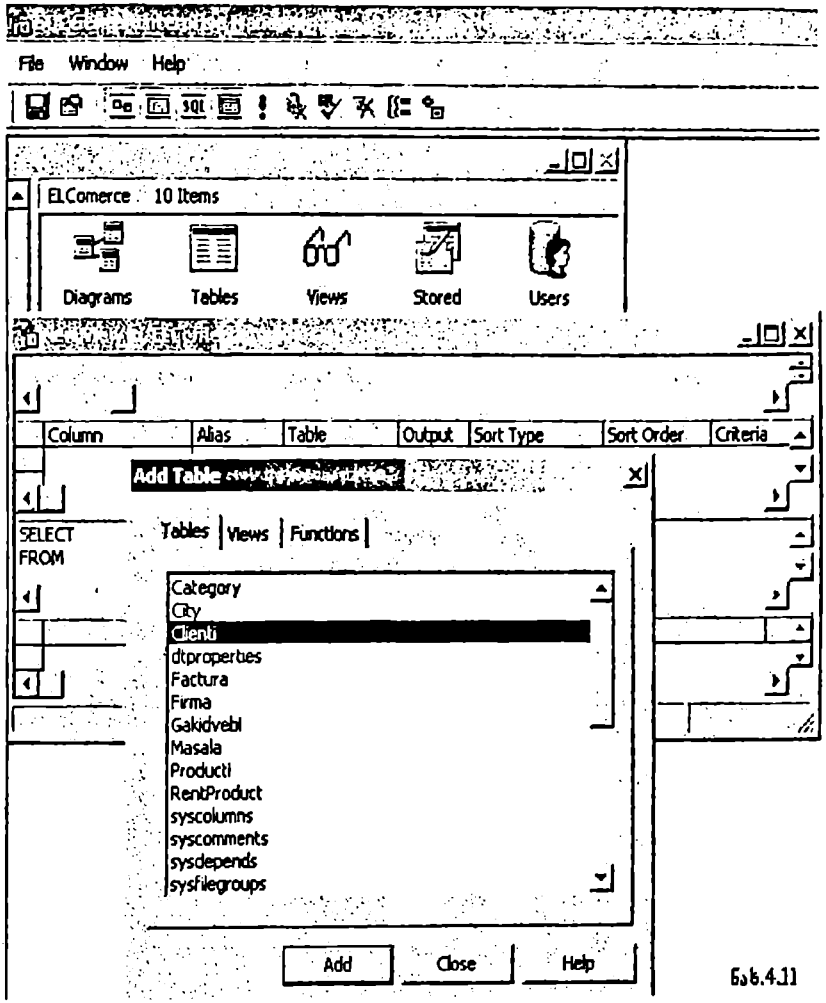


Table and Index Properties
Type
Length
Allow Nulls
mzarmId
int
4
saxeli
char
30
qalaqi
char
20
misamarTi
char
30
tel
char
12
fax
char
12
Tables
Relationships
Indexes/Keys
Check Constraints
Selected table:
Mzarmoeb
Owner:
dbo
Table name:
Mzarmoeb
Table Identity Column:
mzarmId
Table ROWGUID Column:
Table Filegroup:
PRIMARY
Text Filegroup:
PRIMARY
Description:

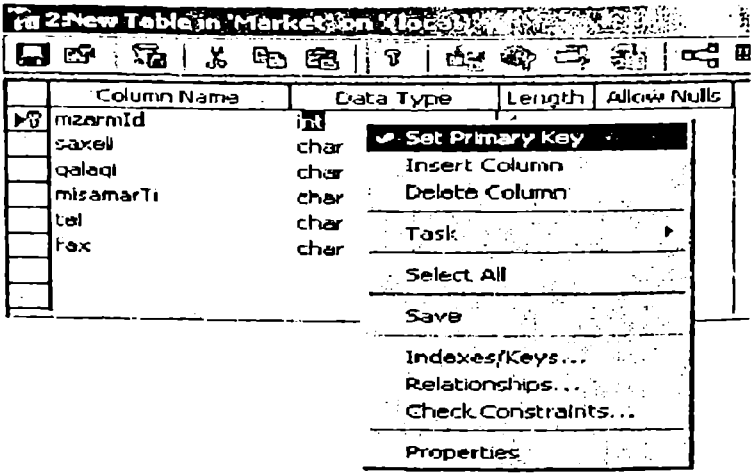
ნახ.4.10



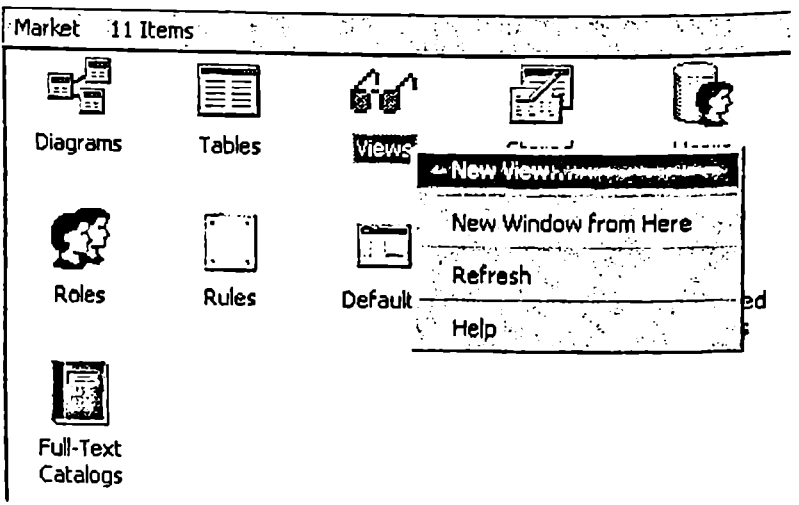
ნახ.4.11

SQL Server-ში გამოიყენება ორი ტიპის ინდექსი: კლასტერული და არაკლასტერული. პირველის შემთხვევაში ცხრილი ფიზიკურად მოწესრიგდება ამ ინდექსით (ასეთია მაგ., გასაღებური ველი). ერთ ცხრილში დასაშვებია მხოლოდ ერთი კლასტერული ინდექსი.

არაკლასტერულია დანარჩენი ინდექსები (მაქსიმუმ 249), რომლებიც



ნახ.4.12. **Table Properties Primary Key**



ნახ.4.13. **View -წარმოდგენის შექმნა**

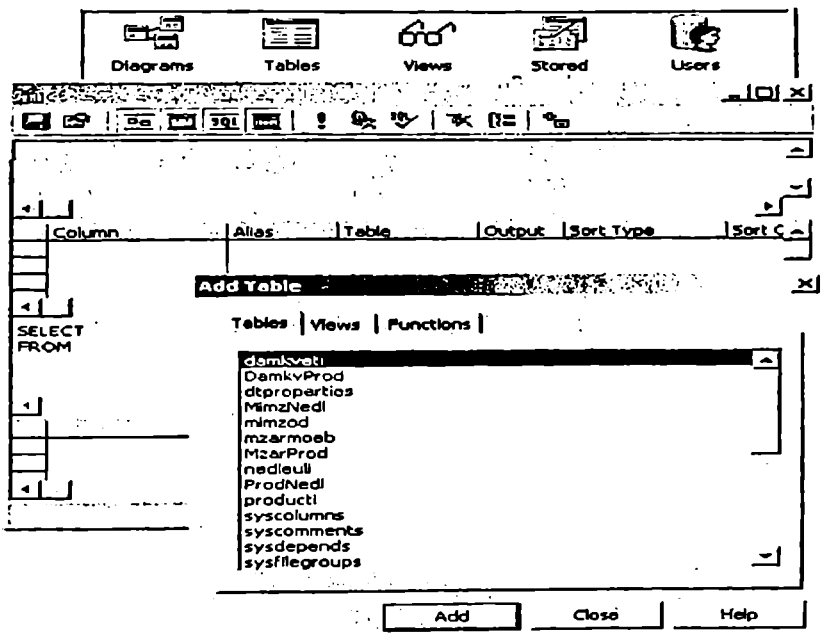
ამ ინდექსირებული ველის მოწესრიგებითაა დალაგებული, მაგრამ ძირითად ცხრილში ჩანაწერების მიმდევრობა არ იცვლება.

ინდექსური ველი შეიძლება შედგენილი იყოს ცხრილის რამდენიმე ველის სახელისგან. მაშინ ჩანაწერების მოწესრიგება ხდება ზრდადობით ან კლებადობით ამ ველების შესაბამისად. ჯერ ლაგდება პირველი ველით, შემდეგ მის შიგნით მეორით და ა.შ.

Views - წარმოდგენები იქმნება შესაბამისი ცხრილ[ებ]იდან მომხმარებელთა მოთხოვნების საფუძველზე (მაგ., Select-ინსტრუქციით). მასში არაა ჩაწერილი რეალური მონაცემები, იგი ვირტუალური ცხრილია,სადაც შეიძლება გარკვეული მანიპულაციების ჩატარება მონაცემთა მიზნობრივი დამუშავებისთვის.

4.10-4.14 ნახაზებზე ნაჩვენებია ახალი წარმოდგენის შექმნის ვიზუალური პროცედურები ცხრილისათვის - "დამკვეთი".

შედეგში გამოსატანი ველები უნდა მოინიშნოს ცხრილის მართკუთხედებში. ავტომატურად შეივსება ქვედა ცხრილი (Column, Alias, . . .), პარალელურად კი ფორმირდება SQL-მოთხოვნაც.



ნახ.4.14.

აქ შესაძლებელია მოთხოვნის კორექტირება ხელითაც, მაგ., ჩავამატოთ სტრიქონი WHERE qalaqi='Tbilisi'.

Constraints - შეზღუდვები მთლიანობაზე უზრუნველყოფს მონაცემთა მთლიანობას ცხრილების ან სვეტების დონეებზე. SQL Server-ში გამოიყენება ხუთი ტიპის შეზღუდვა:

- PRIMARY KEY : შეზღუდვა პირველად გასაღებზე. ცხრილის საზღვრებში პირველადი გასაღების მნიშვნელობა უნიკალურია (არ არსებობს მისი ორი ერთნაირი, ან Null-ის ტოლი მნიშვნელობა). ეს

ამუშავება (Run) ხდება მენიუდან " ! " - პიქტოგრამით. საშედეგო წარმოდგენა მოცემულია 4.15 ნახაზის ქვედა ცხრილში. ჩვენს შემთხვევაში ერთი სტრიქონია "დამკვეთი თბილისიდან".

შეზღუდვა უზრუნველყოფს მონაცემთა ლოგიკურ მთლიანობას.

- FOREIGN KEY : შეზღუდვა მეორეულ გასაღებზე. ცხრილის საზღვრებში პირველადი გასაღების მნიშვნელობა უნიკალურია (არ არსებობს მისი ორი ერთნაირი, ან Null-ის ტოლი მნიშვნელობა). ეს შეზღუდვა უზრუნველყოფს მონაცემთა ლოგიკურ მთლიანობას.

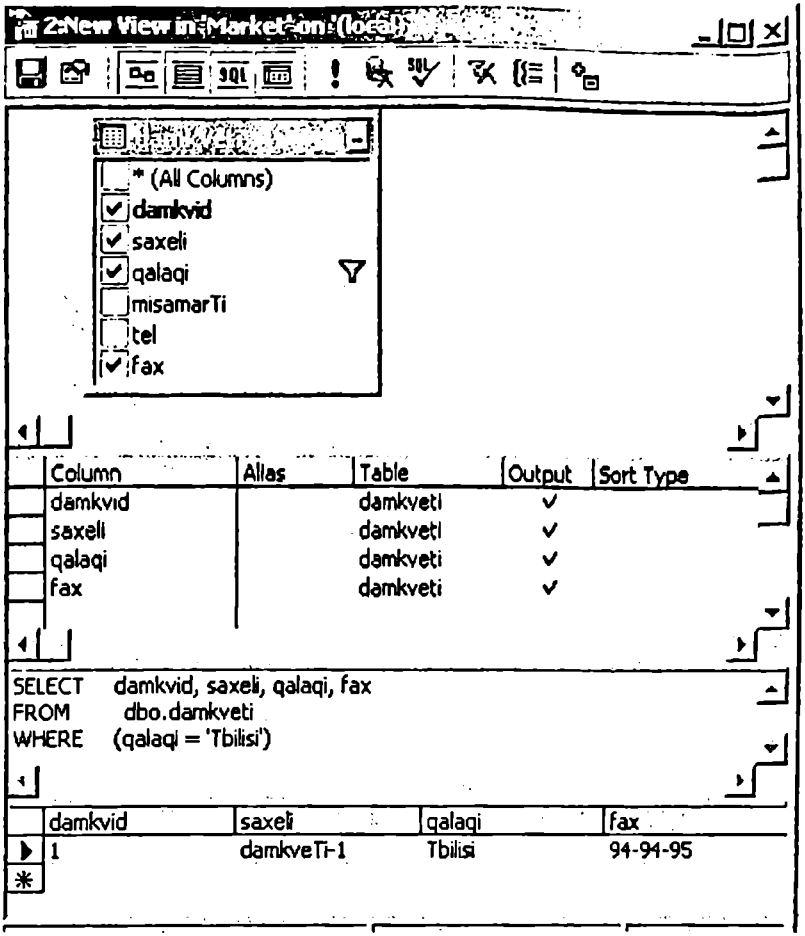
- FOREIGN KEY : შეზღუდვა მეორეულ გასაღებზე უზრუნველყოფს ცხრილთაშორისი კავშირების მთლიანობას. ეს კავშირები აიგება პირველადი და მეორეული გასაღებების ბაზაზე.

- UNIQUE : შეზღუდვა უნიკალურობაზე უზრუნველყოფს სვეტისათვის მნიშვნელობათა განუმეორებლობას. უნიკალურობა პირველადი გასაღების ფუნქციაა. შეიძლება ასევე უნიკალურობით ინდექსის შექმნა სხვა სვეტისათვისაც. მისთვის დასაშვებია Null მნიშვნელობაც.

- CHECK : შეზღუდვა მნიშვნელობაზე უზრუნველყოფს შესატანი მონაცემების კონტროლს. საკონტროლო მნიშვნელობათა დიაპაზონები წინასწარ განისაზღვრება სვეტებისათვის.

- NOT NULL : შეზღუდვა განუსაზღვრელ მნიშვნელობაზე უზრუნველყოფს სვეტისათვის არანულოვანი (განუსაზღვრელი) მნიშვნელობის თავიდან აცილებას.

Rules - წესები გამოიყენება ცხრილის სვეტებზე მნიშვნელობათა შეზღუდვების (CHECK) მსგავსად. ერთ (ან რამდენიმე) სვეტზე მხოლოდ ერთი წესია მიმაგრებული. ყველა სვეტს შეიძლება საკუთარი წესი ჰქონდეს. CHECK-ით კი შეიძლება რამდენიმე შეზღუდვის გამოყენება ერთ სვეტზე. ამიტომაც რეკომენდებულია სისტემებში მისი ხმარება.



ნახ.4.15

Defaults - მნიშვნელობები გამოუცხადებლად (ავტომატურად) მიენიჭება სვეტებს ცხრილის შექმნის დროს.

Trigger - ტრიგერი შენახვადი პროცედურაა, რომელიც სრულდება ავტომატურად SQL Server-ის ცხრილის განახლების დროს UPDATE, INSERT ან DELETE ინსტრუქციებით. ტრიგერების ინსტრუქციების ჩაწერა ხორციელდება Transact-SQL ენის ოპერატორთა ერთობლიობით.

ისინი გამოიყენება როგორც FOREIGN KEY-შეზღუდვები ცხრილთაშორისი კავშირების მთლიანობის უზრუნველსაყოფად, ოღონდ შედარებით რთული კავშირების აღსაწერად. ტრიგერების ამუშავება დამოკიდებულია მონაცემთა მნიშვნელობებზე. მაგალითად, თუ ცხრილში მოხდება რაიმე მონაცემთა ცვლილება, ტრიგერს შეუძლია მისი დაფიქსირება ტრანზაქციების ჟურნალში.[32]

Stored procedure - შენახვადი პროცედურა არის Transact-SQL ენის ინსტრუქციების ერთობლიობა, რომელიც შექმნის დროს კომპილირდება სპეციალურ ფორმატში (შესრულების გეგმა). ესაა მონაცემთა ბაზის ადმინისტრირების ძალზედ მოქნილი და ეფექტური საშუალება. მასზე შეიძლება ჩაიწეროს მონაცემთა დამუშავების რთული და მრავალფეროვანი ლოგიკური პროცესები.

შენახვადი პროცედურის შექმნის შემდეგ მისი შესაბამისი შესრულების გეგმა სისტემის მიერ განიცდის ოპტიმიზაციას გამოყენების ეფექტურობის ამაღლების თვალსაზრისით.

Extended stored procedures - გაფართოებადი შენახვადი პროცედურები იქმნება dll ფაილების სახით დაპროგრამების ენების საფუძველზე, მაგალითად Visual Basic, C++, C#, Java და სხვ.

მათი შექმნის შემდეგ ყოველი ფუნქცია უნდა დარეგისტრირდეს SQL Server-ში sp_addextendedproc შენახვადი პროცედურით.

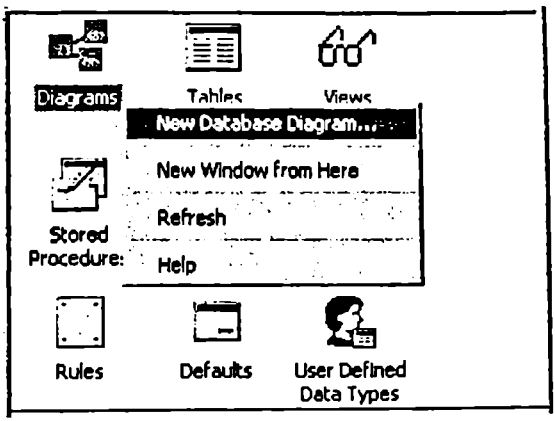
4.5.2. SQL Server-ის მონაცემთა ბაზის ER-დიაგრამები

მონაცემთა რელაციური ბაზების დაპროექტებისას განსაკუთრებული მნიშვნელობა აქვს ცხრილებისა და ცხრილთაშორის კავშირების აგების პროცესის ავტომატიზაციას. ამ მხრივ მონაცემთა ბაზების მართვის სისტემების თანამედროვე პროგრამული პაკეტები, როგორებიცაა: Oracle, SQL Server, Access, SyBase, InterBase, Visual FoxPro და სხვ., ფლობს სპეციალურ ინსტრუმენტულ საშუალებებს.

ER-დიაგრამების (Entity Relationship diagrams) ასაგებად SQL Server-ში გამოიყენება როგორც Transact-SQL ენის პროგრამული ინსტრუქციები (მაგ.,CREATE TABLE),ასევე Enterprise Manager

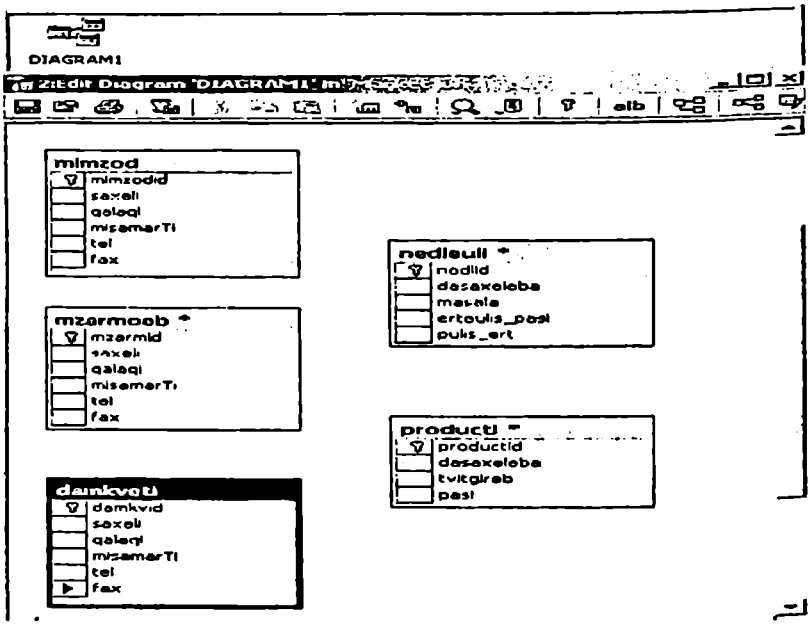
ინსტრუმენტის Diagrams ვიზუალური დაპროექტების კომპონენტი (ნახ. 4.5). განვიხილოთ ეს ინსტრუმენტი ჩვენი სადემონსტრაციო მაგალითისათვის. ცხრილებს შორისი კავშირების ვიზუალურად ასაგებად, როგორც ზემოთ აღვნიშნეთ, საჭიროა პირველადი (PRIMARY KEY) და მეორეული (FOREIGN KEY) გასაღებების არსებობა. ასეთ შემთხვევაში "თაგუს" მეშვეობით დავაკავშირებთ მათ ერთმანეთთან.

4.16 ნახაზზე ნაჩვენებია მონაცემთა ბაზის დიაგრამის გამოძახების ფრაგმენტი.

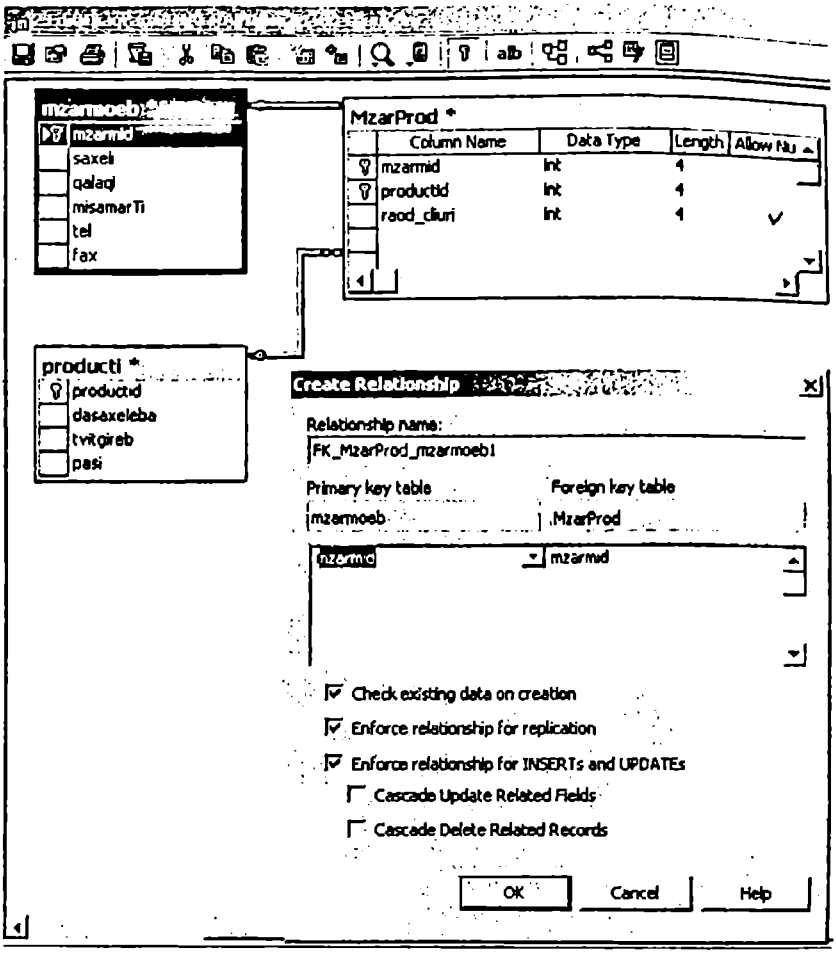


ნახ.4.16. Database Diagrams გამოძახება

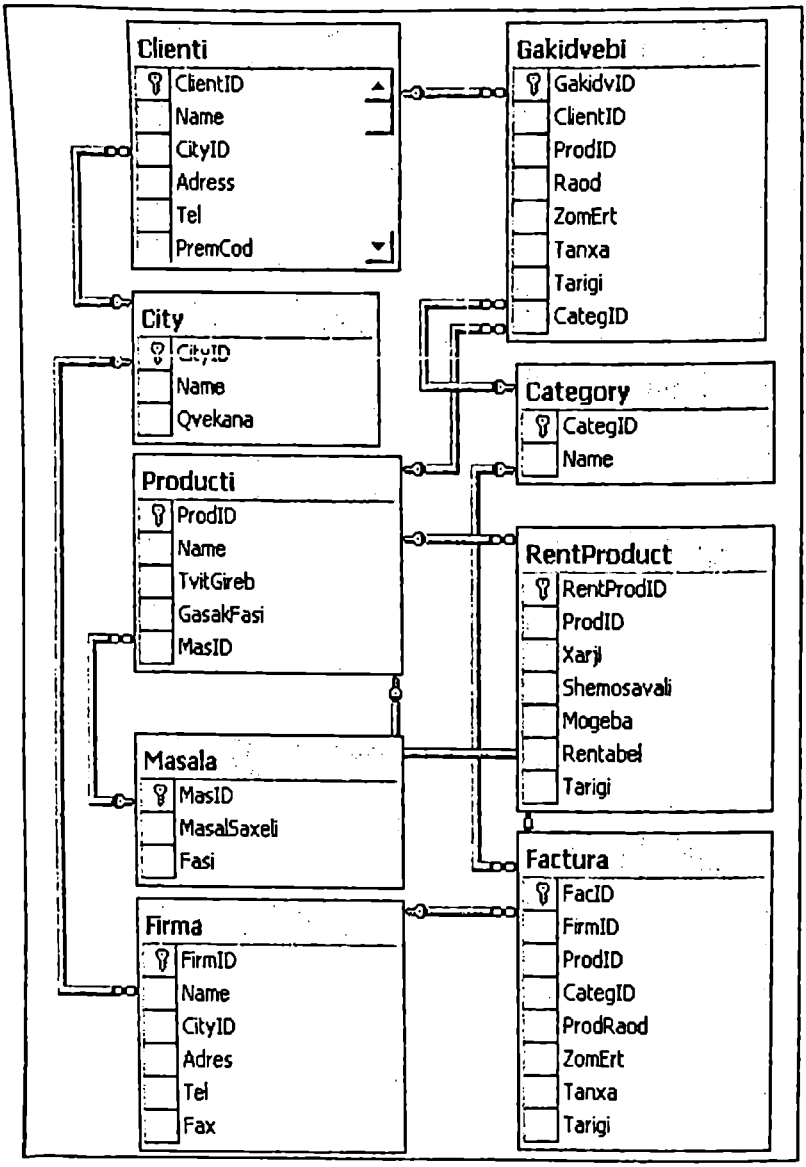
შესაძლებელია ცხრილების გამოტანა მხოლოდ მათი ველების დასახელებებით (ნახ.4.17-18) ან მხოლოდ გასაღებური ველების ჩვენებით (ნახ.4.19).



ნახ.4.17. Database Diagram



ნახ.4.18. **Create Relationship**


Clienti
ClientID
Name
CityID
Adress
Tel
PremCod
Gakidvebi
GakidvID
ClientID
ProdID
Raod
ZomErt
Tanxa
Tarigi
CategID
City
CityID
Name
Qvekana
Category
CategID
Name
Producti
ProdID
Name
TvitGireb
GasakFasi
MasID
RentProduct
RentProdID
ProdID
Xarji
Shemosavali
Mogeba
Rentabel
Tarigi
Masala
MasID
MasalSaxeli
Fasi
Factura
FacID
FirmID
ProdID
CategID
ProdRaod
ZomErt
Tanxa
Tarigi
Firma
FirmID
Name
CityID
Adres
Tel
Fax

ნახ.4.19

4.5.3. SQL Server-ის მონაცემთა ცხრილებთან მუშაობა

ჩვენ მიერ შექმნილი ცხრილები ინახება მონაცემთა Market ბაზაში, რომელთა სია მოცემულია 4.20 ნახაზზე. ცხრილებში მონაცემთა შესატანად ან შესაცვლელად საჭიროა მათი ამორჩევა (ნახ.4.21) და კონკრეტული მანიპულაციების ჩატარება (ნახ.4.22). აქ ნაჩვენებია ბაზის ძირითადი ცხრილების "დამკვეთი", "მიმწოდებელი", "ნედლეული", "მწარმოებელი" და "პროდუქცია" ფრაგმენტები-ცხრილების არჩევის ალტერნატიული შესაძლებლობა ასახულია 4.25 ნახაზზე. ხოლო 4.24-4.26 ნახაზებზე ილუსტრირებულია ცხრილთაშორის დამოკიდებულებათა ცხრილების აგების ფრაგმენტები.

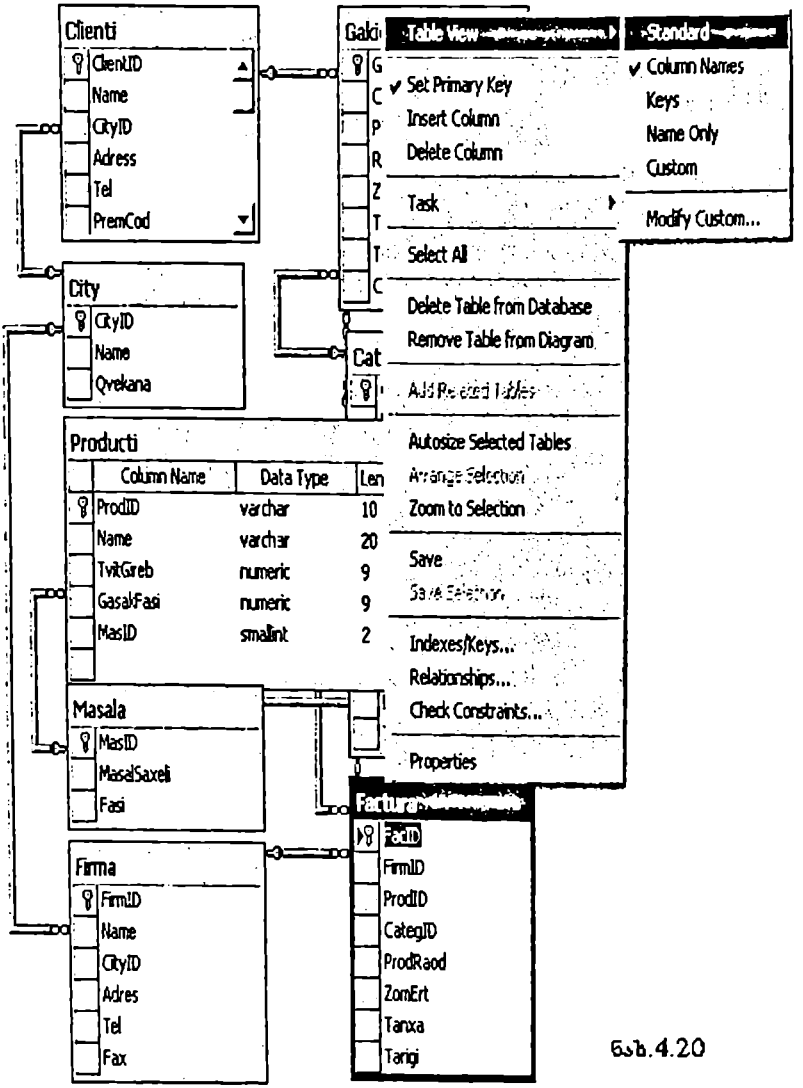
4.5.4. SQL Server-ში ინდექსური ფაილების აგება

მონაცემთა ცხრილებისათვის შესაძლებელია ინდექსური ფაილების შექმნა, რომლებიც, როგორც აღვნიშნეთ, ძებნის პროცედურების დაჩქარებას უწყობს ხელს. 4.27 ნახაზზე ნაჩვენებია ცხრილისათვის "პროდუქტი" დამხმარე მენიუდან Indexes/Keys პუნქტის ამორჩევა, რის შემდეგაც გამოიტანება 4.28 კადრი (Properties). კომბობოქსის გრაფაში Selected Index ჩანს PK_producti, რაც პირველადი გასაღების არსებობას მიუთითებს.

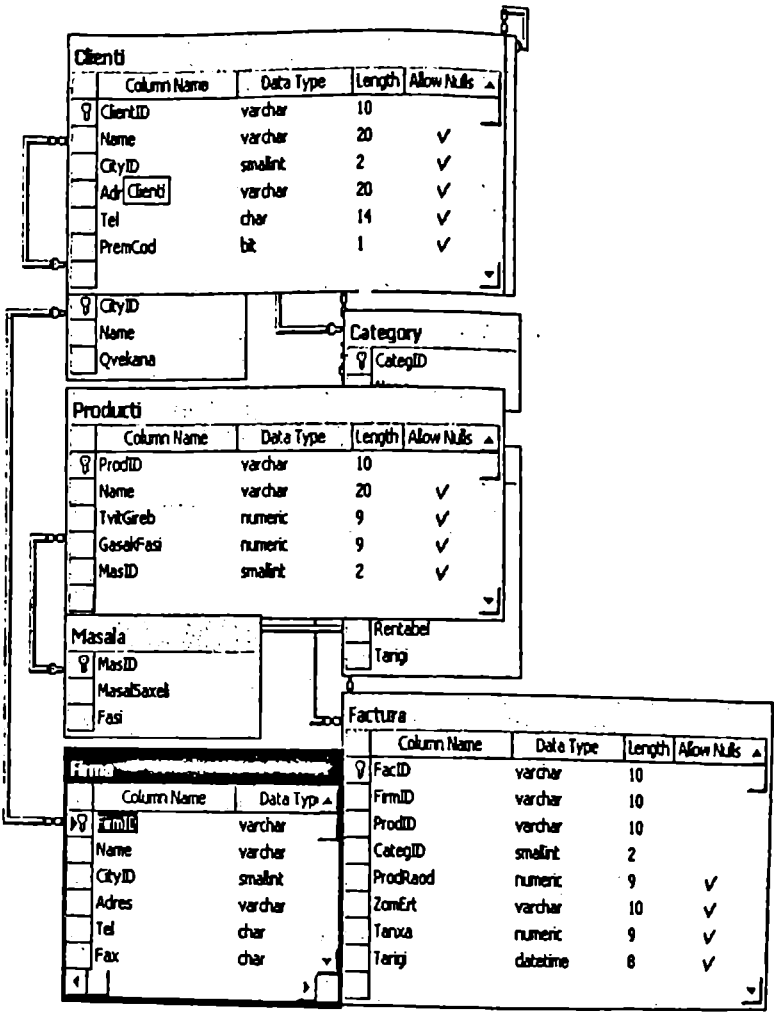
ახალი ინდექსის შესაქმნელად ვირჩევთ New-ღილაკს და Index_name გრაფაში უნდა ჩავწეროთ ინდექს-ფაილის სახელი, მაგ., IX_producti_1. Column name კომბობოქსში ავირჩევთ ველის მნიშვნელობას, რომლისთვისაც იქმნება ინდექსი. ბოლოს Order გრაფაში მივუთითებთ ამ ცხრილის სვეტში მონაცემთა მოწესრიგების მიმდევრობას Ascending-ზრდადობით ან Descending-კლებადობით.

მოცემულ ნახაზზე ნაჩვენებია ერთი ცხრილისთვის სამი სხვადასხვა ინდექს-ფაილის შექმნა პროდუქციის "დასახელების", "თვითღირებულების" და "ფასის" ველების მიხედვით.

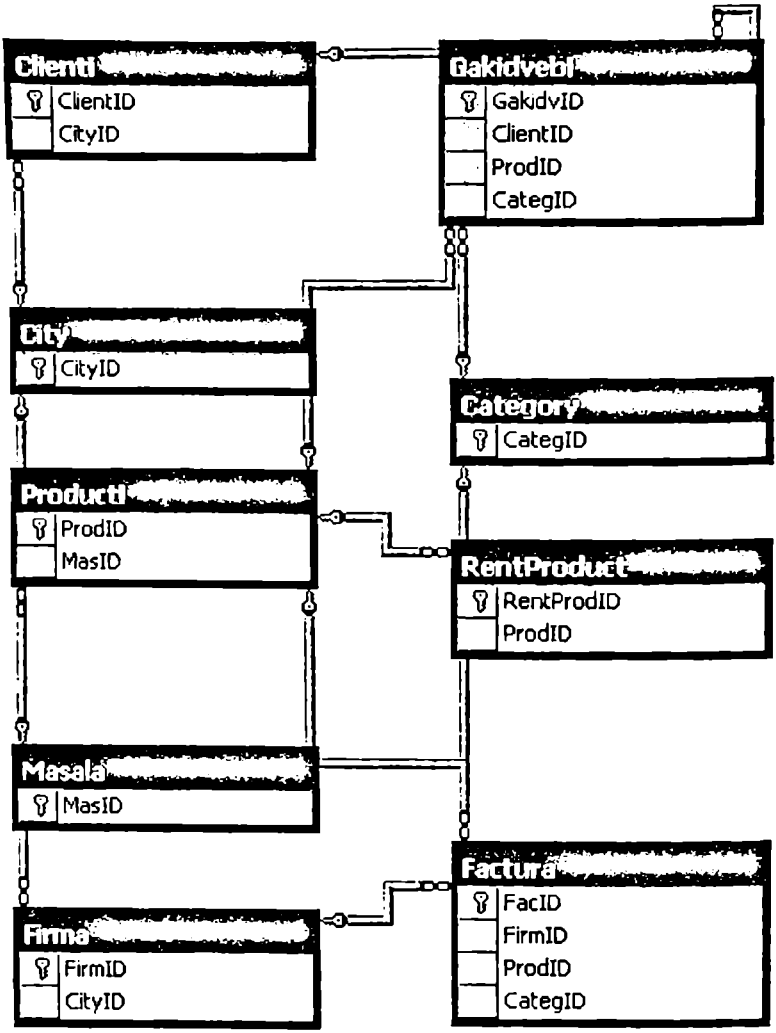
2.28 ნახაზზე ნაჩვენებია ცხრილისათვის - "მწარმოებელი" ინდექსური ფაილის IX_mzarmoeb შექმნის შემთხვევა saxeli-ველის ზრდადი მოწესრიგებით.


Clienti
ClientID
Name
CityID
Adress
Tel
PremCod
City
CityID
Name
Qvekana
Producti
Column Name
Data Type
Len
ProdID
varchar
10
Name
varchar
20
TvitGreb
numeric
9
GasalFasi
numeric
9
MasID
smallint
2
Masala
MasID
MasalSaxeli
Fasi
Firma
FirmID
Name
CityID
Adres
Tel
Fax
FirmID
ProdID
CategID
ProdRaod
ZomErt
Tanxa
Tarigi
Table View
Set Primary Key
Insert Column
Delete Column
Task
Select All
Delete Table from Database
Remove Table from Diagram
Autosize Selected Tables
Zoom to Selection
Save
Indexes/Keys...
Relationships...
Check Constraints...
Properties
Standard
Column Names
Keys
Name Only
Custom
Modify Custom...

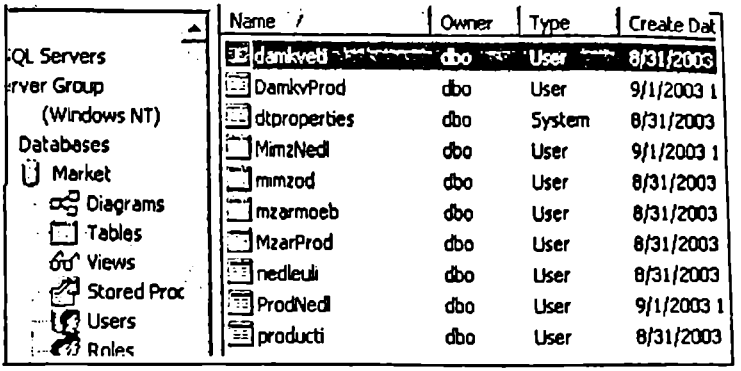
ნახ.4.20


Clienti
Column Name
Data Type
Length
Allow Nulls
ClientID
varchar
10
Name
varchar
20
CityID
smallint
2
Adr
varchar
20
Tel
char
14
PremCod
bit
1
CityID
Name
Qvekana
Category
CategID
Producti
ProdID
varchar
10
Name
varchar
20
TvitGireb
numeric
9
GasakFasi
numeric
9
MasID
smallint
2
Masala
MasID
MasalSaxeli
Fasi
Rentabel
Tanxi
Factura
FacID
varchar
10
FirmID
varchar
10
ProdID
varchar
10
CategID
smallint
2
ProdRaod
numeric
9
ZomErt
varchar
10
Tanxa
numeric
9
Tarigi
datetime
8
Column Name
Data Typ
FirmID
varchar
Name
varchar
CityID
smallint
Adres
varchar
Tel
char
Fax
char

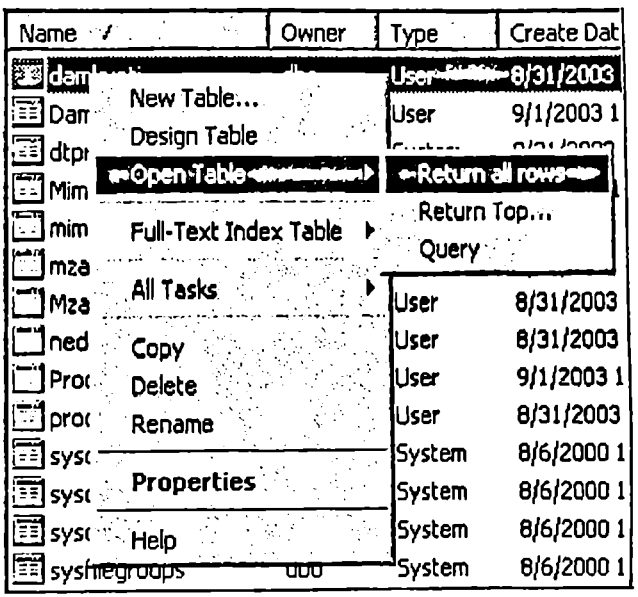
ნახ.4.21


Clienti
ClientID
CityID
GakidvebI
GakidvID
ClientID
ProdID
CategID
City
CityID
Category
CategID
Producti
ProdID
MasID
RentProduct
RentProdID
ProdID
Masala
MasID
Factura
FacID
FirmID
ProdID
CategID
Firma
FirmID
CityID

ნახ.4.22

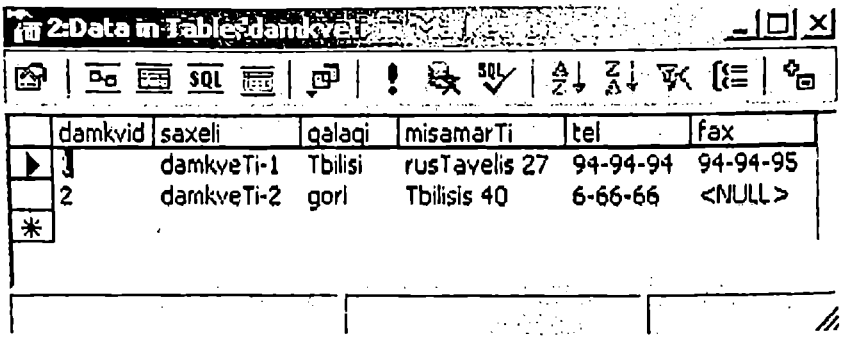


ნახ.4.23



ნახ.4.24. ამორჩეული ცხრილის სტრიქონებთან მუშაობა

დამკვეთი



მიმწოდებელი

mimzodid	saxeli	qalaqi	misamarTi	tel	fax
1	mimzodebeli-1	Tbilisi	didubis 15	34-34-34	34-34-35
2	mimzodebeli-2	Tbilisi	v-fSavelas 250	30-39-39	39-30-40
3	mimzodebeli-3	Kutaisi	bagratis 20	15-15-15	15-15-16
4	mimzodebeli-4	rusatavi	metalurebis 17	19-19-19	19-19-20
5	mimzodebeli-5	zugdidi	dadianis 5	6-23-23	<NULL>

ნედლეული

nedlid	dasaxeleba	masala	erteulis pasi	pulis ert
1	nedleuli-1	masala-1	5	lari
2	nedleuli-2	masala-1	12.5	lari
3	nedleuli-2	masala-2	6.75	lari
4	nedleuli-4	masala-1	8.65	dolari
5	nedleuli-5	masala-3	25	lari
6	nedleuli-5	masala-4	40	lari
7	nedleuli-5	masala-5	15.7	lari

მწარმოებელი

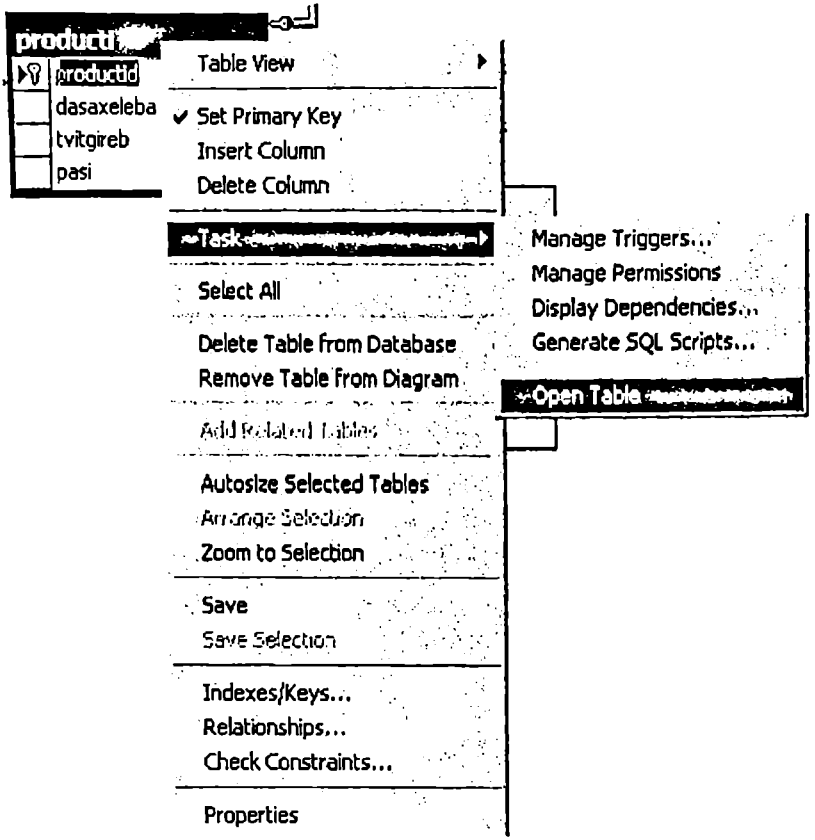
mzarmid	saxeli	qalaqi	misamarTi	tel	fax
1	mzarmoebeli-1	Tbilisi	didi diRomi	34-20-20	34-20-21
2	mzarmoebeli-2	Gori	stalinis 79	2-22-22	<NULL>
3	mzarmoebeli-3	Tbilisi	muxiani	60-60-60	60-60-61
4	mzarmoebeli-4	baTumi	abaSiZis 100	6-66-66	6-66-99

ნახ.4.25. ცხრილის სტრიქონების შევსება/კორექტირება

პროდუქტი

	productid	dasaxeleba	tvitgireb	pasi
	1	produkti-1	105	155
	2	produkti-2	40	75
	3	produkti-2	67	95
	4	produkti-1	80	120
	5	produkti-3	29.57	40
	6	produkti-4	39	52
	7	produkti-4	57	80
	8	produkti-4	100	145
	9	produkti-5	69	104
	10	produkti-6	15	27
▶				

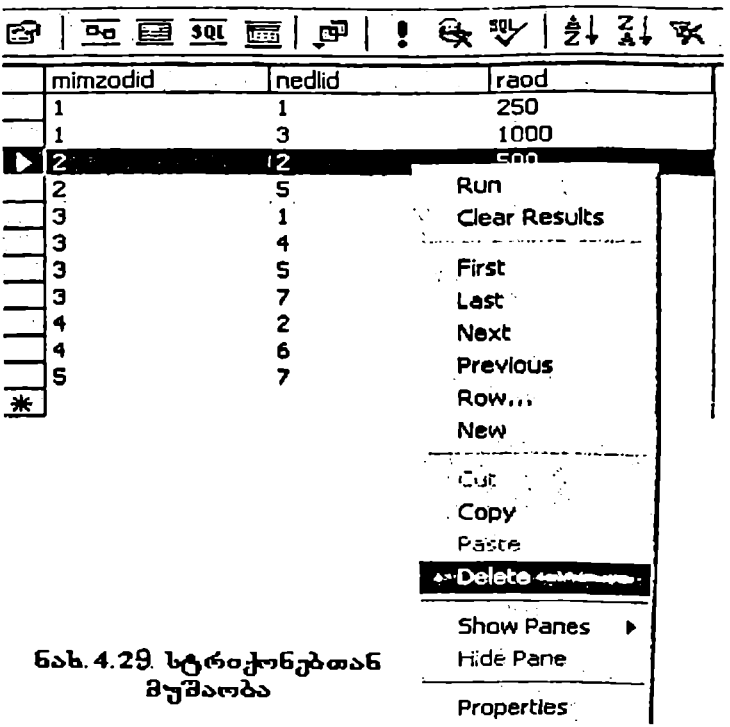
ნახ.4.26. გაგრძელება



ნახ.4.27. ცხრილებთან მუშაობის ალტერნატიული ვარიანტი

	mimzodid	nedlid	raod
	1	1	250
	1	3	1000
	2	2	500
	2	5	1500
	3	1	500
	3	4	2000
	3	5	1000
	3	7	389
	4	2	800
	4	6	4000
▶	5	7	9786
*			

ნახ. 4.28



ნახ. 4.29. სტრიქონებთან მუშაობა

პროდუქცია-ნედლეული

	productid	nedlid	nedlraod	zomis_ert	pasi
	1	1	5	cali	<NULL>
	1	3	10	kg	<NULL>
	1	6	1	cali	<NULL>
	2	2	10	m	<NULL>
	3	3	10	m	<NULL>
	4	2	20	m	<NULL>
	4	5	1	t	<NULL>
	5	2	50	m	<NULL>
	5	5	2	t	<NULL>
	6	6	15	cali	<NULL>
	7	3	20	kg	<NULL>
	7	1	10	cali	<NULL>
	7	5	12	m	<NULL>
	8	7	45	l	<NULL>
	9	1	10	cali	<NULL>
	9	5	1	t	<NULL>
	10	3	50	m	<NULL>
	10	5	1	t	<NULL>
▶	10	7	20	l	<NULL>
*					

მწარმოებელი-პროდუქცია

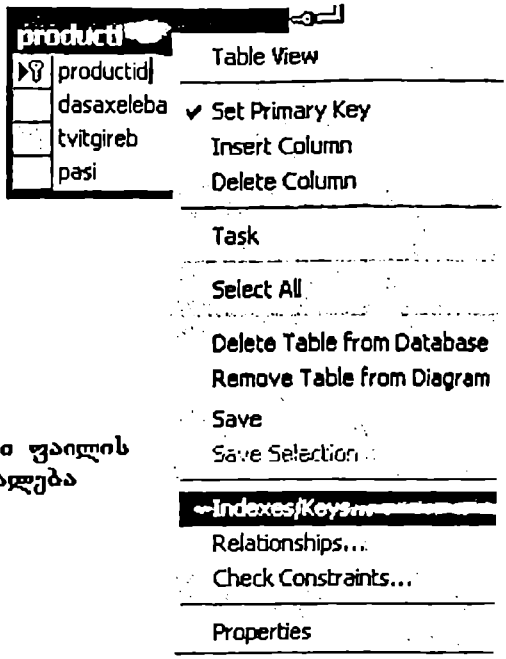
	mzarmid	productid	raod_cliuri
▶	1	1	500
	1	4	1000
	1	5	2000
	1	7	850
	2	1	1000
	3	3	1700
	3	10	2500
	4	2	10000
	4	6	350
	4	8	5000
	2	9	789
	2	4	400
*			

ნახ.4.30. დამოკიდებულებათა დანარჩენი ცხრილები

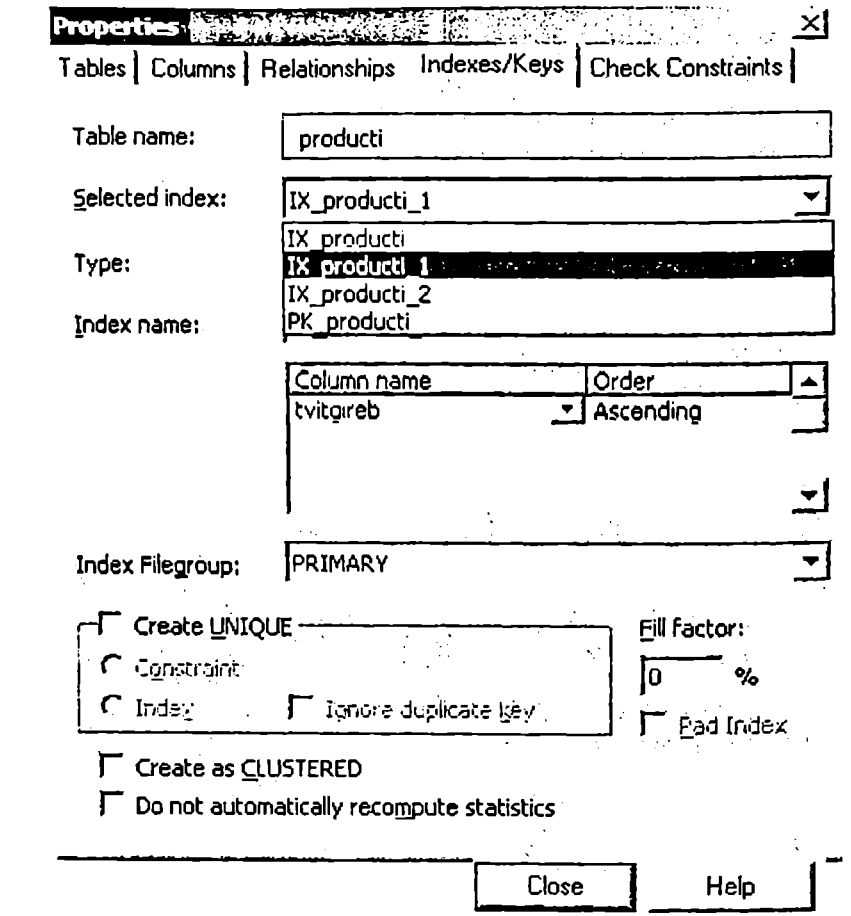
დამკვეთი-პროდუქცია

	damkvid	productid	raod	tarigi	tanxa
	1	1	100	12/25/2003	<NULL>
	1	3	500	3/1/2004	<NULL>
	1	7	1000	10/1/2004	<NULL>
	2	1	400	2/2/2004	<NULL>
	2	2	987	5/31/2004	<NULL>
	2	5	20	10/31/2003	<NULL>
	2	6	250	11/30/2003	<NULL>
	1	6	200	12/30/2003	<NULL>
	1	4	600	4/20/2004	<NULL>
	2	8	300	12/15/2003	<NULL>
▶	[illegible]	10	70	1/26/2004	<NULL>
*					

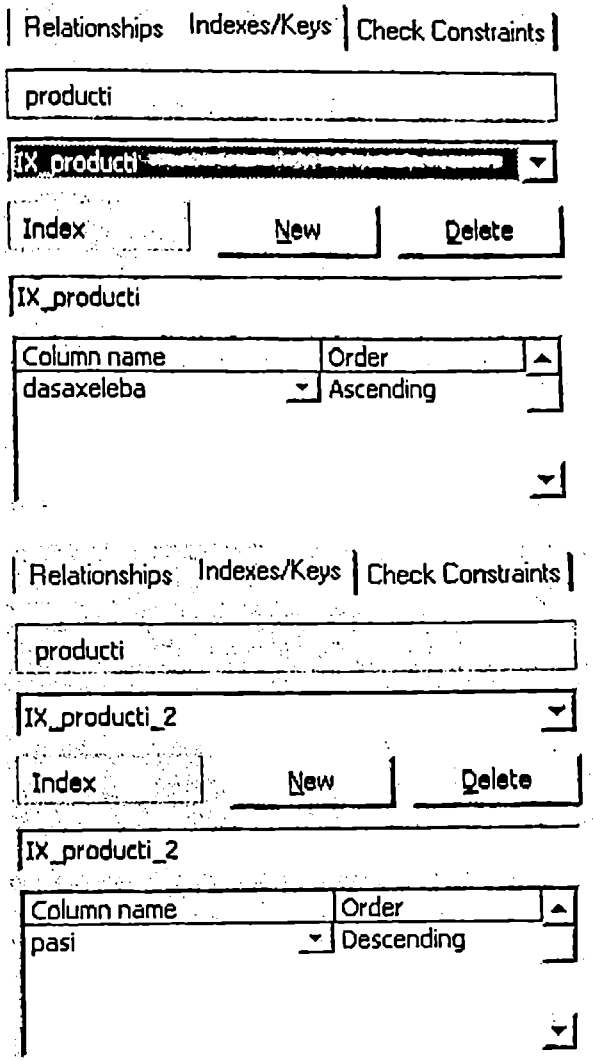
ნახ.4.31. გაგრძელება



ნახ. 4.32. ინდექსური ფაილის შექმნის საშუალება

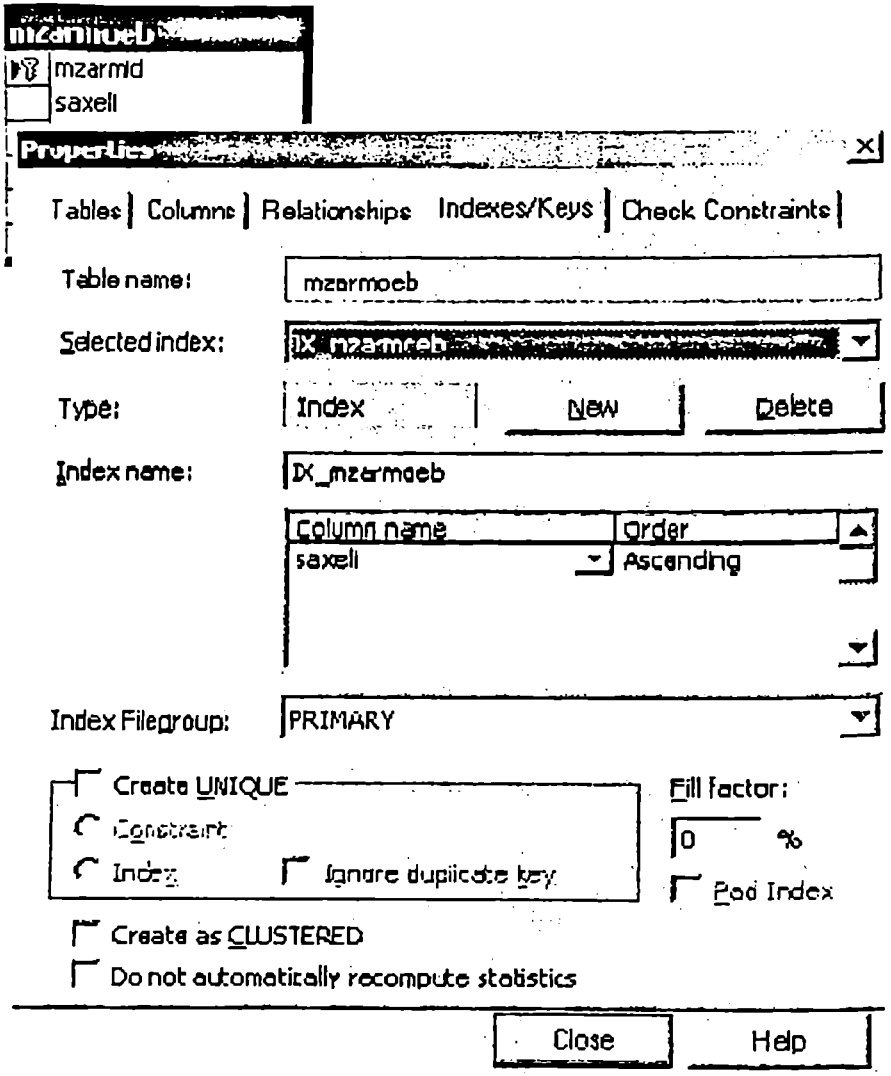


ნახ.4.33. სამი ინდექსური ფაილის შექმნა ცხრილისათვის **producti** მოწესრიგების ველებით **dasaxeleba, tvitgireb, pasi** (კლებადობით)



ნახ.4.34. გაგრძელება

ნახ.4.35. ინდექსური IX-mzarmoeb ველის შექმნა ცხრილისათვის mzarmoeb მოწესრიგების ველით saxeli



4.6. ASP.NET პროგრამული პაკეტით Web-გვერდის აგების მაგალითი

Web მოდული უზრუნველყოფს კლიენტების და პროგრამის კავშირს. იგი წარმოადგენს ASP.NET ვებ-გვერდების და მათზე მიბმული კოდის ფაილების ერთობლიობას. ამ მოდულში ხდება მომხმარებლის მიერ სხვადასხვა პროდუქციის ნახვა, ძებნა, არჩევა, შეკვეთა, მომხმარებლის რეგისტრაცია სისტემაში და პირადი მონაცემები შეტანა.

ამ მოდულს აქვს სხვა მოდილებთანაც კავშირი. მასში ხდება მომხმრებლის მოქმედებების დამუშავება და სხვა მოდულების გამოძახება, რომლებიც უზრუნველყოფს მონაცემთა ბაზასთან ურთიერთქმედებას (მონაცემთა ამოკრეფა, ჩაწერა, შეცვლა და სხვა). განვიხილოთ 5 ძირითადი მოდული:

Web - უზრუნველყოფს კლიენტების და პროგრამის კავშირს.

- **Business Facade** - უზრუნველყოფს აპლიკაციის ლოგიკასა და მომხმარებლის ინტერფეისს შორის კავშირს. ჩვენს შემთხვევაში, ახორციელებს ჭებ-ის კლიენტების პროფილების, კატეგორიების დათვალიერების და პროდუქციის შეძენის ოპერაციებს ინტერფეისით. იგი ჩართულია პროექტში როგორც BusinessFacade პროექტი, ემსახურება მას როგორც საიზოლაციო დონე და უზრუნველყოფს მომხმრაბლელთა გამოყოფას ინტერფეისისაგან სხვადასხვა ბიზნეს ლოგიკის განხორციელებისგან. მონაცემთა ბაზასთან ურთიერთობა ხდება ამ პროგრამის საშუალებით.

- **Business Rules** - უზრუნველყოფს მოქმედების (უსინესს ღულეს) წესების ინკაპსულაციას. იგი შეიცავს მონაცემების დმუშავების სხვადასხვა წესებს, როგორიცაა მომხმარებლების პროფილების შემოწმებას და პროდუქციის შეკვეთის პროცედურას.

- **Data Access** - ახორციელებს მონაცემთა ბაზასთან მუშაობას. იგი უზრუნველყოფს მონაცემთა დამუშავების ფუნქციებით Business Rules მოდულს.

- **SystemFramework** - მოდული შეიცავს პროგრამის კონფიგურაციის მონაცემებს, შეცდომების დამუშავებას და პროცესების მონიტორინგს.

თითოეულ მათგანს გააჩნია კავშირები სხვადასხვა მოდულებთან. მათი უმრავლესობა წარმოადგენს კომპილირებად ბიბლიოთეკებს, რომლებშიც მოთავსებულია ფუნქციები და პროცედურები. ისინი უზრუნველყოფს სხვადასხვა დანიშნულების ოპერაციების ჩატარებას, რომლებიც სრულდება ინტერნეტ მომხმარებლის მიერ ინიციალიზირებულ მოვლენებისათვის. მდგრადი პროგრამების შექმნისა.

ASP.NET არის კომპილირებადი, .NET-პლატფორმაზე დაფუძნებული გარემო. ნებისმიერი ASP.NET–ზე შექმნილი პროგრამისთვის შესაძლებელია ამ ტექნოლოგიის სხვადასხვა საშუალებების გამოყენება, როგორიცაა მაგალითად: უნივერსალური ენის გარემო (CLR),როცა შესაძლებელია ASP.NET–ის მოდულები დაიწეროს .NET-ის სხვდასხვა ენების საშუალებით Visual Basic.Net, C#, JScript.Net; მონაცემთა ტიპების უსაფრთხოება,მემკვიდრეობითობა და სხვ.

ASP.NET საშუალებას იძლევა გამოვიყენოთ HTML ელემენტები როგორც ობიექტები. მათი მეთოდების, თვისებების და მოვლენების გამოყენება ბევრად აიოლებს დაპროგრამების მოდელს.

ASP.NET-ის გვერდების კოდი არის კომპილირებული,ვიდრე მათი გამოძახება მოხდება. კომპილირებული კოდი უმჯობესია ინტერპრეტირებად კოდზე, რადგანაც მასში ნაკლებია შეცდომების გაპარვის შესაძლებლობა და უფრო სწრაფადაც სრულდება. აგრეთვე არის გვერდების კეშირების საშუალება, რაც ზრდის სისტემის ეფექტურობას და მწარმოებლურობას.

XML Web service საშუალებას იძლევა გამოვიყენოთ სერვერის ფუნქციური მეთოდები. შესაძლებელს ხდის მონაცემთა გაცვლას კლიენტ-სერვერულ და სერვერი-სერვერი სისტემებში HTTP და XML შეტყობინებების სტანდარტების გამოყენებით მონაცემთა გაცვლისათვის ქსელური დაცვის აპარატულ-პროგრამულ საშუალებებს შორის. XML Web service არ არის რომელიმე ტექნოლოგიაზე დამოკიდებული. საბოლოოდ პროგრამები, დაწერილი რომელიმე ენაზე,

გამოიყენებს რა რაიმე კომპონენტს და ნებისმიერ ოპერაციულ სისტემაში შეუძლია გამოიყენოს XML Web service.

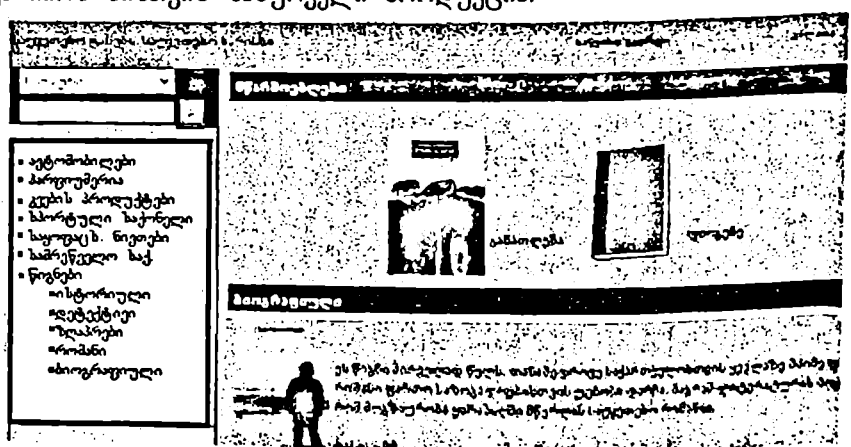
ASP.NET არის ობიექტ-ორიენტირებული და სტრუქტურული ენა. მასში ძალზედ გამარტივებულია მონაცემთა ბაზებთან მუშაობა. იძლევა საშუალებას იოლად დაიწეროს პროგრამის მუშაობის ლოგიკა აპლიკაციისათვის. ეს ლოგიკა შეიძლება შეიცავდეს სხვა აპლიკაციის დონის მოვლენების დამუშავებას.

.NET Framework და ASP.NET საშუალებას იძლევა ავტორიზაციისა და აუთენტიფიკაციისათვის. პროგრამისტს საშუალება აქვს შეცვალოს, დაამატოს ან საერთოდ ამოაგდოს სქემები.

ASP.NET კონფიგურაციის მონაცემები ინახება XML ფაილში, რომლის წაკითხვა და შეცვლა ადვილია. ASP.NET-ის თითოეულ პროგრამას შესაძლებელია ჰქონდეს საკუთარი კონფიგურაციის ფაილი პროგრამის მოთხოვნილებების დასაკმაყოფილებლად.

IIS 6.0 იყენებს პროცესების ახალ მოდელს, რომელიც საშუალებას იძლევა რომ თითოეული მუშა პროცესი წარიმართოს იზალაციურ რეჟიმში, რაც უზრუნველყოფს პროცესის მდგრადობას და მწარმოებლურობის ამაღლებას.

4.36-ა და ბ ნახაზებზე მოცემულია ASP.NET ტექნოლოგიის გამოყენებით ინტერნეტში განთავსებული სავაჭრო ცენტრის საინფორმაციო გვერდი, რომელიც მომხმარებელს საშუალებას აძლევს შეარჩიოს მისთვის სასურველი პროდუქცია.



ნახ. 4.36-ა

ნახ.4.36-ბ. ინტერნეტში განთავსებული სავაჭრო ცენტრის საინფორმაციო გვერდი.

4.7. C# და XML პროგრამათა ფრაგმენტები

C# დაპროგრამების ენა .NET პლატფორმის ერთ-ერთი ყველაზე მძლავრი ინსტრუმენტია, რომელიც java ენის მსგავსად, კლასების საფუძველზე აგებს აპლიკაციებს. წინა პარაგრაფში ჩვენს მიერ რეალიზებული Web-გვერდი ASP.NET სისტემაში, რომელიც SQL Server-თან კავშირშია, აღიწერება C# კოდით:

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Security;
using System.ComponentModel;
using System.Data;
using Magazia.SystemFramework;
using Magazia.Common;
using Magazia.Common.Data;
namespace Magazia.Web
{
  public class Checkout : PageBase
  {
    private const String KEY_STAGE = "stage";
     protected System.Web.UI.WebControls.Panel ShippingPanel;
    protected System.Web.UI.WebControls.TextBox ShipToNameTextBox;
    protected System.Web.UI.WebControls.TextBox AddressTextBox;
    protected System.Web.UI.WebControls.TextBox CountryTextBox;
    protected System.Web.UI.WebControls.TextBox PhoneNumberTextBox;
    protected System.Web.UI.WebControls.TextBox FaxTextBox;
     protected System.Web.UI.WebControls.Panel PaymentPanel;
     protected System.Web.UI.WebControls.Label AmountLabel;
            protected System.Web.UI.WebControls.DropDownList
CartTypeDropDownList;
    protected System.Web.UI.WebControls.TextBox NameOnCardTextBox;
    protected System.Web.UI.WebControls.TextBox CardNumberTextBox;
    protected System.Web.UI.WebControls.TextBox BillingInfoTextBox;
```

```
        protected System.Web.UI.WebControls.CustomValidator
BillingInfoCustomValidator;
        protected System.Web.UI.WebControls.DropDownList
ExpMonthDropDownList;
        protected System.Web.UI.WebControls.DropDownList
ExpYearDropDownListBox;
    protected System.Web.UI.WebControls.Panel SummaryPanel;
    protected System.Web.UI.WebControls.DataGrid ShoppingCartDataGrid;
    protected System.Web.UI.WebControls.Label SubTotalLabel;
    protected System.Web.UI.WebControls.Label TaxLabel;
    protected System.Web.UI.WebControls.Label ShippingHandlingLabel;
    protected System.Web.UI.WebControls.Label TotalLabel;
    protected System.Web.UI.WebControls.ImageButton NextImageButton;
        protected System.Web.UI.WebControls.ImageButton
PreviousImageButton;
    protected Magazia.Web.CheckoutModule ModuleCheckout;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator1;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator2;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator3;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator4;
      protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator1;
      protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator2;
        protected System.Web.UI.WebControls.ValidationSummary
ValidationSummary1;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator5;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator6;
      protected System.Web.UI.WebControls.RegularExpressionValidator
RegularExpressionValidator3;
        protected System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator7;
```

```
        protected System.Web.UI.WebControls.ValidationSummary
ValidationSummary2;
    private const String KEY_TRANSAMOUNT = "transamount";
     private int stage;
       public Checkout()
          {
              Page.Init += new System.EventHandler(Page_Init);
     }
     protected void Page_Load(Object sender, EventArgs e)
     {
      stage = IsPostBack ? Int32.Parse(ViewState[KEY_STAGE].ToString())
:0;
            Cart checkoutShoppingCart = base.ShoppingCart(false);
                 ApplicationAssert.Check(!checkoutShoppingCart.IsEmpty,
"Empty Shopping Cart at Checkout", ApplicationAssert.LineNumber - 4);
       if (checkoutShoppingCart == null || checkoutShoppingCart.IsEmpty)
       {
          Response.Redirect(PageBase.UrlBase + @"/shoppingcart.aspx",
false);
         return;
       }

       if (Customer == null)
       {
         checkoutShoppingCart.Customer = null;
         FormsAuthentication.SignOut();
         Response.Redirect(@"checkout.aspx", false);
         return;
       }
                                  ShoppingCartDataGrid.DataSource =
(ICollection)checkoutShoppingCart.OrderItems.DefaultView;
      checkoutShoppingCart.CalculateOrderSummary();
           DataRow row = checkoutShoppingCart.OrderSummary.Rows[0];
             SubTotalLabel.Text = System.String.Format("{0:N}",
row[OrderData.SUB_TOTAL_FIELD]);
                TaxLabel.Text = System.String.Format("{0:N}",
row[OrderData.TAX_FIELD]);
     ShippingHandlingLabel.Text = System.String.Format("{0:N}",
row[OrderData.SHIPPING_HANDLING_FIELD]);
```

```
        TotalLabel.Text = System.String.Format("{0:N}",
row[OrderData.TOTAL_FIELD]);
            ShoppingCartDataGrid.DataBind();
          BillingInfoTextBox.TextMode = TextBoxMode.MultiLi
AmountLabel=
System.String.Format("{0:N}",row[OrderData.TOTAL_FIELD]);
                if (!IsPostBack)
        {
          row = checkoutShoppingCart.ShippingAddress.Rows[0];
ShipToNameTextBox.Text=
row[OrderData.SHIP_TO_NAME_FIELD].ToString();
          AddressTextBox.Text =
row[OrderData.ADDRESS_FIELD].ToString();
          CountryTextBox.Text =
row[OrderData.COUNTRY_FIELD].ToString();
          PhoneNumberTextBox.Text =
row[OrderData.PHONE_NUMBER_FIELD].ToString();
        FaxTextBox.Text = row[OrderData.FAX_FIELD].ToString();
                SetPanelDisplay();

          int startYear = DateTime.Now.Year;
                        for (int i=0; i < 10; i++)
                        {

ExpYearDropDownListBox.Items.Add(startYear++.ToString());
                        }

      }
    }
        protected void Page_Init(object sender, EventArgs e)
    {
      InitializeComponent();
    }
        private void InitializeComponent()
                {
                        this.PreviousImageButton.Click += new
System.Web.UI.ImageClickEventHandler(this.PreviousImageButton_Click);
                        this.NextImageButton.Click += new
System.Web.UI.ImageClickEventHandler(this.NextImageButton_Click);
```

```
                    this.Load += new
System.EventHandler(this.Page_Load);

                }
                public void NextImageButton_Click (object sender,
System.Web.UI.ImageClickEventArgs e)
    {
      bool stepSuccess = false;
                    ShipToNameTextBox.Text =
ShipToNameTextBox.Text.Trim();
      AddressTextBox.Text = AddressTextBox.Text.Trim();
      CountryTextBox.Text = CountryTextBox.Text.Trim();
      PhoneNumberTextBox.Text = PhoneNumberTextBox.Text.Trim();
      FaxTextBox.Text = FaxTextBox.Text.Trim();
      NameOnCardTextBox.Text = NameOnCardTextBox.Text.Trim();
      CardNumberTextBox.Text = CardNumberTextBox.Text.Trim();
      BillingInfoTextBox.Text = BillingInfoTextBox.Text.Trim();
       foreach (IValidator val in Page.Validators)
       {
         val.Validate();
       }

        switch (stage)
       {
          case 0:
               stepSuccess = ValidateShipping();
               stepSuccess = ValidatePayment();
              break;
           case 1:
                stepSuccess = ValidatePayment();
               break;
            case 2:
                stepSuccess = SubmitOrder();
                break;
           }
        if (stepSuccess)
       {
        ++stage;
       }
```

```
      SetPanelDisplay();
    }
  public void PreviousImageButton_Click (object sender,
System.Web.UI.ImageClickEventArgs e)
    {
      —stage;
       SetPanelDisplay();
    }
    private void SetPanelDisplay()
    {
      switch (stage)
      {
        case 0:
          ShowPanel(ShippingPanel, true);
          ShowPanel(PaymentPanel, false);
          ShowPanel(SummaryPanel, false);
          NextImageButton.Enabled = true;
          NextImageButton.ImageUrl = "../images/next.gif";
           PreviousImageButton.Enabled = false;
      PreviousImageButton.ImageUrl = "../images/  previousdisabled.gif";
           break;
        case 1:
                ShowPanel(ShippingPanel, false);
                 ShowPanel(PaymentPanel, true);
                ShowPanel(SummaryPanel, false);
                NextImageButton.Enabled = true;
                 NextImageButton.ImageUrl = "../images/next.gif";
                 PreviousImageButton.Enabled = true;
           PreviousImageButton.ImageUrl = "../images/     previous.gif";
           break;
        case 2:
                ShowPanel(ShippingPanel, false);
          ShowPanel(PaymentPanel, false);
          ShowPanel(SummaryPanel, true);
          NextImageButton.Enabled = true;
          NextImageButton.ImageUrl = "../images/confirm.gif";
           PreviousImageButton.Enabled = true;
           PreviousImageButton.ImageUrl = "../images/previous.gif";
           break;
```

```
    }
        ViewState[KEY_STAGE] = stage.ToString();
          ModuleCheckout.Stage = stage;
}
    private bool ValidateShipping()
{
  bool errorsFound = !Page.IsValid;
  if (errorsFound)
  {
    return false;
  }
  else
  {
  ShoppingCart(false).SetShippingAddress(ShipToNameTextBox.Text,
                                AddressTextBox.Text,
                                CountryTextBox.Text,
                                PhoneNumberTextBox.Text,
                                FaxTextBox.Text);
    return true;
  }
}
   private bool ValidatePayment()
{
  bool errorsFound = false;

if (BillingInfoTextBox.Text.Length >= 255)
  {
   BillingInfoCustomValidator.IsValid = false;
    errorsFound = true;
 }
 else
 {
  BillingInfoCustomValidator.IsValid = true;
 }

 bool errorsFound = !Page.IsValid
 if (errorsFound)
   return false;
}
```

```
        else
        {
         String expDate = ExpMonthDropDownList.SelectedItem.Value + "/"
+ ExpYearDropDownListBox.SelectedItem.Value;
          ShoppingCart(false).SetPayment(CartTypeDropDownList.Items
         [CartTypeDropDownList.SelectedIndex].Text,
                              NameOnCardTextBox.Text,
                              CardNumberTextBox.Text,
                               expDate,
                              BillingInfoTextBox.Text,
                               "AuthorizationCode");
          return true;
        }
      }
         private bool SubmitOrder()
      {    ValidatePayment();
       ApplicationLog.WriteTrace("Magazia.Web.Checkout.SubmitOrder:");
            ShoppingCart(false).AddOrder();
            Response.Redirect("order.aspx", false);
            return false;
      }       private void ShowPanel(Panel panel, bool visible)
      {
        IValidator validator;
        foreach (Control ctrl in panel.Controls)
        {
          if ( ctrl is IValidator)
          {
            validator = (IValidator)ctrl;
            ctrl.Visible = visible;
            if (!visible)
            {  validator.Validate();   }
          }
        }
        panel.Visible = visible;
      }
    }
}
```

სისტემა იყენებს აგრეთვე XML და HTML ენებს. eXtensible Markup Language (XML მონაცემთა ფორმატირების გაფართოებული ენა)

თანამედროვე ინფორმაციული ტექნოლოგიების გამოყენებისას მეტად აქტუალურია. ის არის (მეტაენა), რომლის საშუალებითაც შესაძლებელია სხვადასხვა ტიპის მონაცემთა გადაცემა განსხვავებულ აპლიკაციებს შორის პლატფორმის დამოუკიდებლად.

XML დოკუმენტი არის ჩვეულებრივი ASCII ფაილი, რომელიც თავისუფლად გადაიცემა ინტერნეტში. იგი ტექსტური ტიპის პროტოკოლური ფაილია, რომელსაც ადვილად კითხულობს ადამიანიც და მანქანაც.

XML დოკუმენტის ფორმატირების საშუალებას იძლევა XSL (eXtensible Stilesheet Language) სტილური ცხრილი [ლ]. იგი იძლევა ბრაუზერის ეკრანზე ელემენტების გამოსახვის პროცესის მართვის და დოკუმენტში საჭირო ფრაგმენტების ძიების საშუალებას. სტილური ცხრილების დოკუმენტი მოიცავს აგების წესების ერთობლიობას, რომელთაგან თითოეული წესი დაყოფილია ცალკეულ ბლოკად, ორგანიზაციული ტეგებით.

XML კოდის ტექსტი ჩვენი სისტემისათვის მოცემულია ქვემოთ:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
 <configSections>
  <section name="ApplicationConfiguration"
type="Magazia.SystemFramework.ApplicationConfiguration,
Magazia.SystemFramework" />
  <section name="DuwamishConfiguration"
type="Magazia.Common.DuwamishConfiguration, Magazia.Common" />
  <section name="SourceViewer"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934c089" />
 </configSections>
 <system.web>
  <customErrors defaultRedirect="errorpage.aspx" mode="On" />
  <compilation debug="true" />
  <sessionState cookieless="false" timeout="20" mode="InProc"
stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data
source=127.0.0.1;user id=sa;password=" />
  <globalization responseEncoding="utf-8" requestEncoding="utf-8" />
  <authentication mode="Forms">
   <forms name=".ADUAUTH" loginUrl="secure\logon.aspx"
protection="All">
   </forms>
                </authentication>
                <authorization>
```

```
        <allow users="*" />
                </authorization>
 </system.web>
 <ApplicationConfiguration>
  <add key="SystemFramework.Tracing.Enabled" value="False" />
  <add key="SystemFramework.Tracing.TraceFile" value="C:\Program
Files\Microsoft Visual Studio .NET 2003\Enterprise Samples\Duwamish 7.0
CS\\DuwamishTrace.txt" />
  <add key="SystemFramework.Tracing.TraceLevel" value="4" />
  <add key="SystemFramework.Tracing.SwitchName"
value="DuwamishTraceSwitch" />
  <add key="SystemFramework.Tracing.SwitchDescription" value="Error
and information tracing for Duwamish" />
  <add key="SystemFramework.EventLog.Enabled" value="True" />
  <add key="SystemFramework.EventLog.Machine" value="." />
  <add key="SystemFramework.EventLog.SourceName" value="Magazia" />
  <add key="SystemFramework.EventLog.LogLevel" value="1" />
 </ApplicationConfiguration>
 <DuwamishConfiguration>
   <add key="Duwamish.DataAccess.ConnectionString"
value="server=(local);User
ID=sa;Password=sa;database=ShopDB;Connection Reset=FALSE" />
  <add key="Duwamish.Web.EnablePageCache" value="False" />
  <add key="Duwamish.Web.PageCacheExpiresInSeconds" value="3600" />
  <add key="Duwamish.Web.EnableSsl" value="False" />
 </DuwamishConfiguration>
 <SourceViewer>
  <add key="." value=" " />
  <add key="modules" value=" " />
  <add key="..\common\data" value=" " />
  <add key="..\systemframework" value=" " />
   <add key="..\business\facade" value=" " />
   <add key="..\business\rules" value=" " />
   <add key="..\dataaccess" value=" " />
  <add key="secure" value=" " />
  <add key="docs\common" value=" " />
   <add key="docs\dataaccess" value=" " />
   <add key="docs\facade" value=" " />
```

```
  <add key="docs\rules" value=" " />
  <add key="docs\web" value=" " />
 </SourceViewer>
</configuration>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
        <HEAD>
                <TITLE>Internet Shop - Checkout</TITLE>
                <META content="http://schemas.microsoft.com/
intellisense/ie5" name="vs_targetSchema">
                <META http-equiv="content-type" content="text/html;
charset=utf-8">
                <META content="Microsoft Visual Studio 7.0"
name="GENERATOR">
                <META content="C#" name="CODE_LANGUAGE">
                <LINK href="../css/duwamish.css" type="text/css"
rel="Stylesheet">
        </HEAD>
        <body>
                <form id="frmCheckout" method="post" runat="server">
                        <!—PAGE HEADER MODULE—
><MODULE:HEADER id="ModuleBanner" runat="server"
PathPrefix="..."></MODULE:HEADER>
                        <table height="100%" cellSpacing="0"
cellPadding="0" width="100%" border="0">
                                <tr>
                                        <td class="navtable"
vAlign="top" width="1%">
                                                <table
cellSpacing="10" cellPadding="0" width="100%" border="0">
 <!—BEGIN DYNAMIC LEFT MODULE LIST—>
 <tr>
```

```
<td style="PADDING-TOP: 5px" align="left"><MODULE:CHECKOUT
id="ModuleCheckout" runat="server"></MODULE:CHECKOUT></td
</tr>
        <tr>
         <td style="PADDING-TOP: 5px" align="left"></td>
        </tr>
<!— END DYNAMIC LEFT MODULE LIST —></table>
        </td>
        <td vAlign="top" width="99%">
<table cellSpacing="10" cellPadding="0" width="100%" border="0">
        <tr>
        <td style="PADDING-TOP: 5px" align="left">
        <!— BEGIN DYNAMIC RIGHT MODULE LIST —><asp:panel
id="ShippingPanel" runat="server"><!—BEGIN SHIPPING ADDRESS
MODULE—>
        <TABLE cellSpacing="0" cellPadding="5" width="100%">
        <TR class="rheader">
        <TD class="rheadercol" align="left"
height="25">ÛØáÐÛÐà×Ø</TD>
        <TD class="rheadercol" align="right" height="25">ÜÐÑØïØ 1(3)
        </TD>
        </TR> <!— SPACER ROW —
        . . .
```

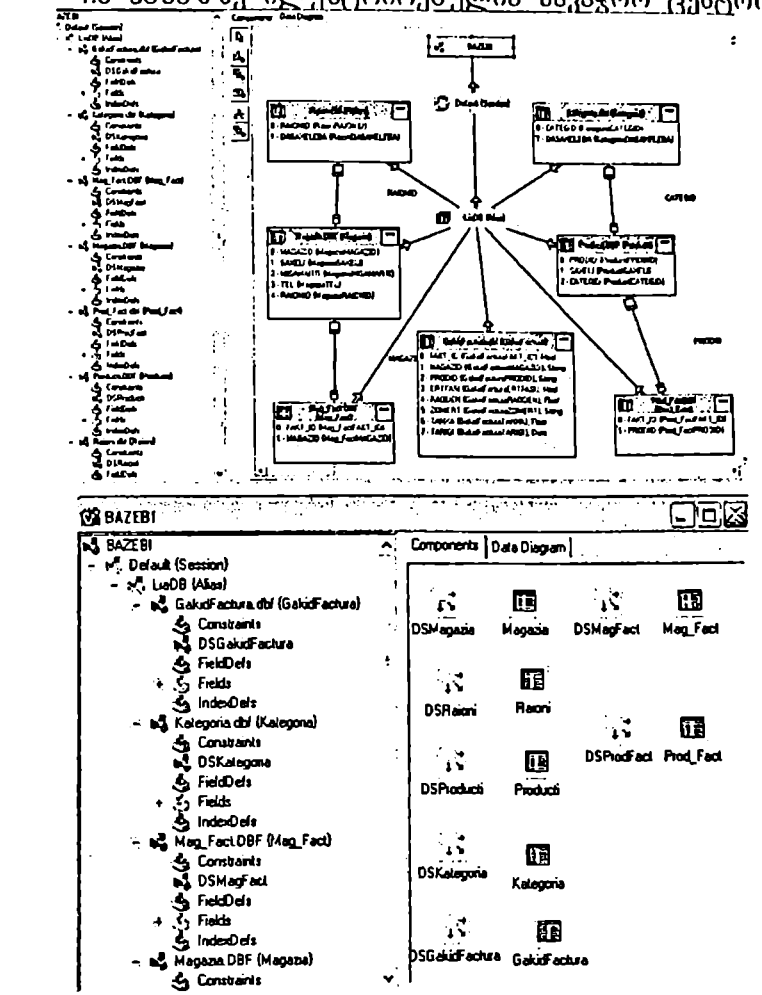
4.8. მონაცემთა მრავალგანზომილებიანი ანალიზის პაკეტის აგება Decision Cube კომპონენტებით

მონაცემთა ბაზებიდან მიღებული ინფორმაციის მრავალფაქტორული ანალიზისათვის C++Builder-ის სპეციალური Decision_Cube კომპონენტებით ავაგეთ პროგრამული პაკეტი.

კუბის ყოველი განზომილება წარმოვადგინეთ მონაცემთა ბაზის ცხრილების ველების სახით (მაღაზიები, პროდუქტები, კლიენტები, კატეგორიები და სხვ.), სადაც დიდი სავაჭრო ქსელის ოპერატიული მართვისა და ეფექტური მუშაობისთვის (რომლის ფილიალები სხვადასხვა რაიონებშია განთავსებული) რეალიზებული გვაქვს სავაჭრო ობიექტებზე საქონელბრუნვის გეგმების შესრულების ოპერატიული ანალიზის ინსტრუმენტი, რომელიც საშუალებას იძლევა ინფორმაცია წარმოვადგინოთ სხვადასხვა ჭრილში.

4.37 ნახაზზე მოცემულია ჩვენი სისტემის მონაცემთა ბაზის სტრუქტურა და ცხრილების კომპონენტები. ინფორმაციის შესატანად გამოიყენება მომხმარებელთა ინტერფეისი, რომელიც 4.38 ნახაზზეა ნაჩვენები.

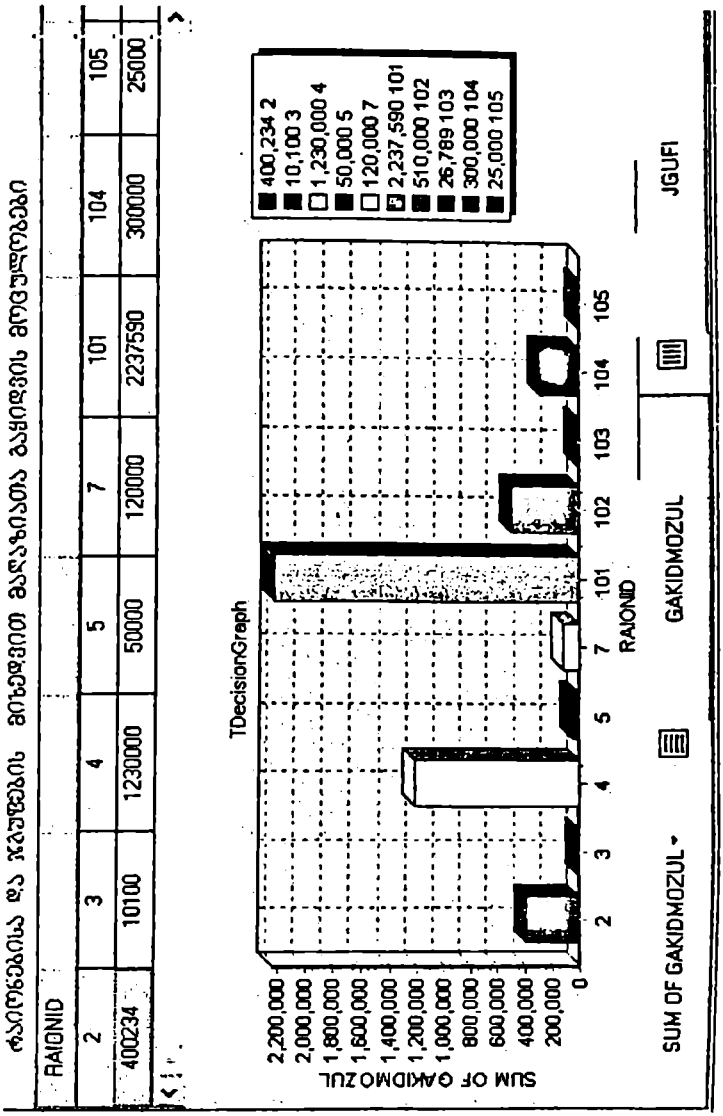
4.8 ნახაზზე ილუსტრირებულია საეჭრო ცენტრის



ნახ.4.37 მონაცემთა ბაზების სტრუქტურა

ნახ.4.4.38. საინტერფეისო ფანჯარა

4.39 ნახაზზე ილუსტრირებულია სავაჭრო ცენტრის მენეჯერის ინტერფეისი ინფორმაციის მრავალფაქტორული ანალიზისათვის

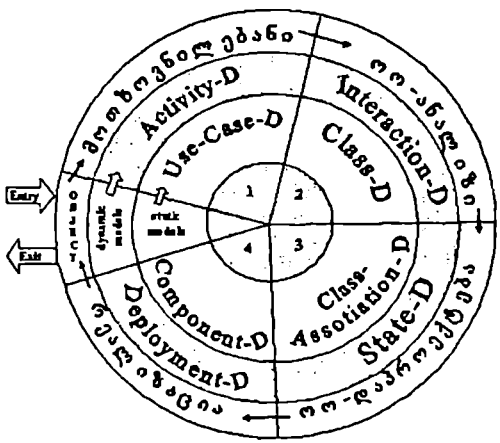


ნახ. 4.39

4.9. ვიზუალური, ობიექტ-ორიენტირებული მოდელირება Ms Visio პაკეტით

განაწილებული სისტემების დასაპროექტებლად და სარეალიზაციოდ ჩვენ ვიყენებთ დაპროგრამების ობიექტ-ორიენტირებულ და სტრუქტურულ მეთოდებს, უნიფიცირებული მოდელირების ენის UML–ტექნილოგიას [17].

პროგრამული პაკეტების შექმნა UML-მეთოდოლოგიით ოთხ ეტაპად ხორციელდება (საკვლევი ობიექტის ავტომატიზაციის მოთხოვნილებების დადგენა, მისი ობიექტ-ორიენტირებული (ოო) ანალიზი, ოო-დაპროექტება (დეტალური დონე) და რეალიზაცია (პროგრამული კოდი). 4.46 ნახაზზე მოცემულია ეს ეტაპები, სადაც ოო-მოდელირება სტატიკური და დინამიკური დიაგრამებით (D) ხორციელდება.

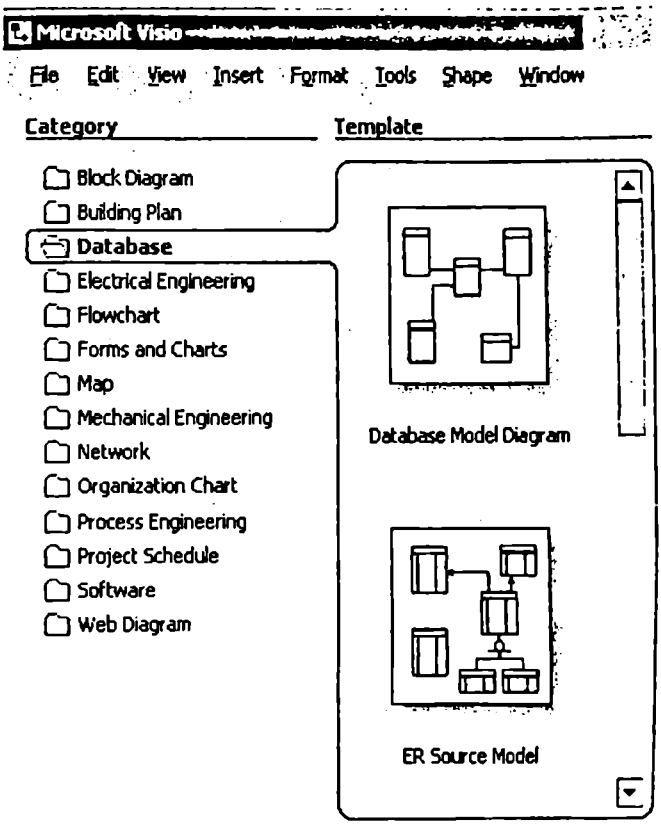


ნახ.1.1. UML–ეტაპები

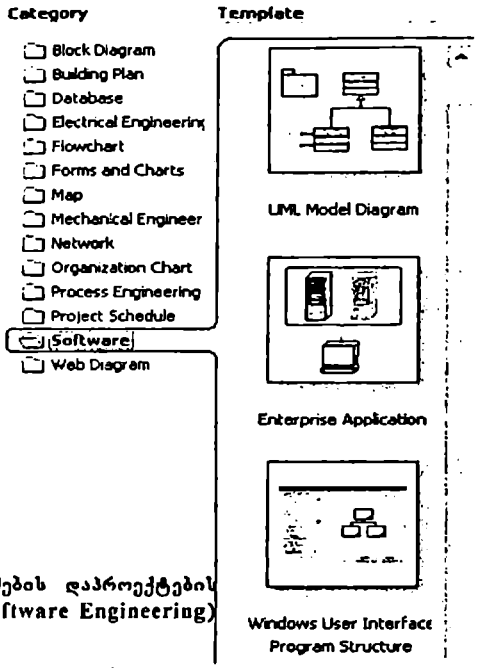
UML–ტექნილოგიის ერთ-ერთი ახალი და ეფექტური ინსტრუმენტული საშუალებაა მაიკროსოფტის ფირმის ბოლო პროგრამული პროდუქტი Ms-Visio, რომელიც კომპიუტერული სისტემებისა და ქსელური ტექნოლოგიების დაპროექტებისა და მოდელირებისათვის გამოიყენება. იგი ინსტალირდება Microsoft Visual Studio.NET პაკეტით, თავსდება ვინდოუსის მთავარ მენიუში:

Start | Programs | Microsoft Visio.

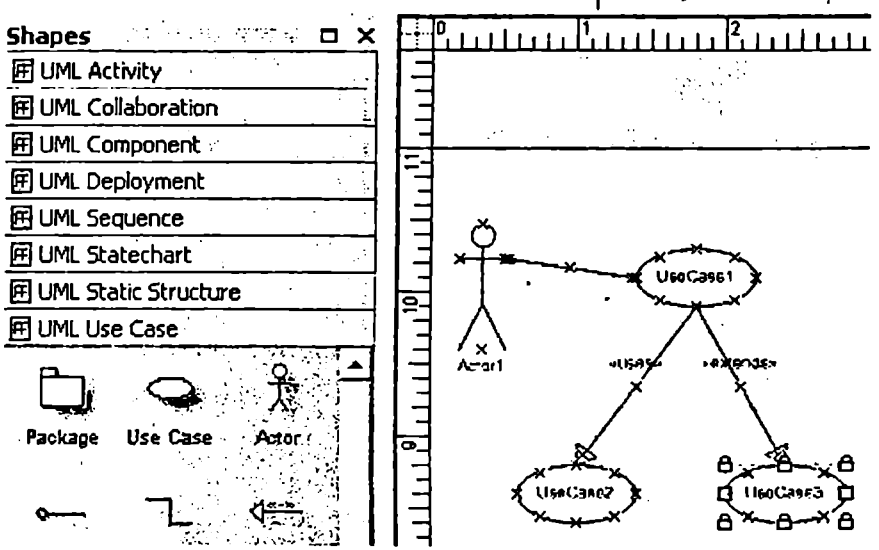
4.47 ნახაზზე ნაჩვენებია ამ პაკეტის სამუშაო გარემო. მის მარცხენა მხარეს Category-ში მოთავსებულია მიზნობრივი საქაღალდეები, ხოლო მარჯვნივ Template-ში არჩეული საქაღალდის შესაბამისი ტიპური შაბლონები (კლასები). მაგ., ჩვენს შემთხვევაში Database-ს შეესაბამება Database Model Diagram, ER Source Model და სხვ. UML–ტექნილოგიის გამოსაყენებლად Category-ში ვირჩევთ Software სტრიქონს და UML Model Diagram (ნახ.4.48). გადავალთ ფორმაზე ვიზუალური კომპონენტების პალიტრით (ნახ.4.49). აქ Shapes ქვეშ ჩანს დიაგრამების ტიპები, მაგ., **UML Use Case, Activity** და ა.შ. შესაბამისი ინსტრუმენტების პანელიდან ფორმაზე გადავიტანთ საჭირო კომპონენტს (მაგ.,Actor, UseCase) შევაერთებთ მათ შესაბამისი



ნახ.4.47. სამუშაო გარემო და Database-ს არჩევა



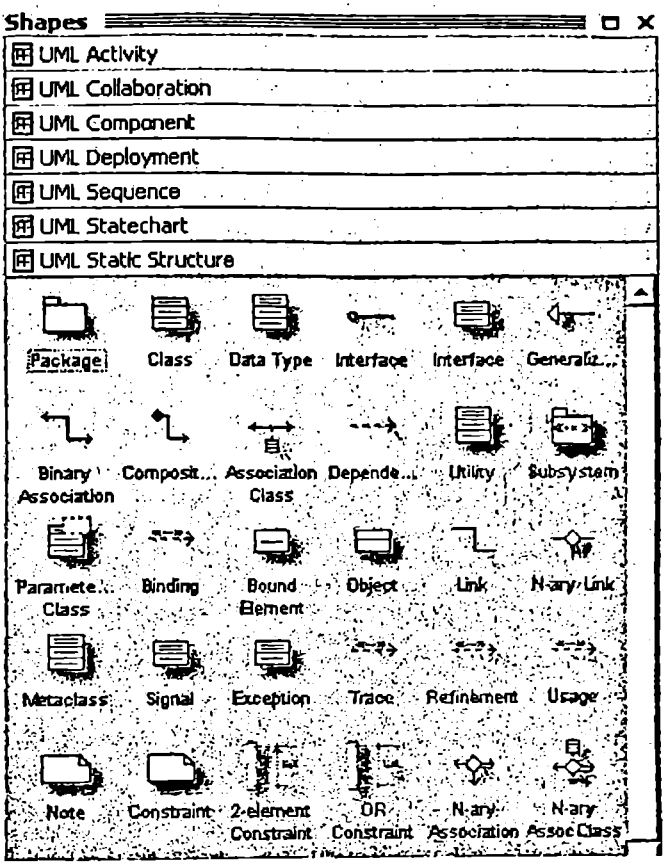
ნახ.4.48. პროგრამების დაპროექტების ინსტრუმენტი (Software Engineering)



ნახ.4.49. UseCase-დიაგრამის აგება ფორმაზე

კავშირის ხაზებით (Communicate, Uses, Extends) და ა.შ.

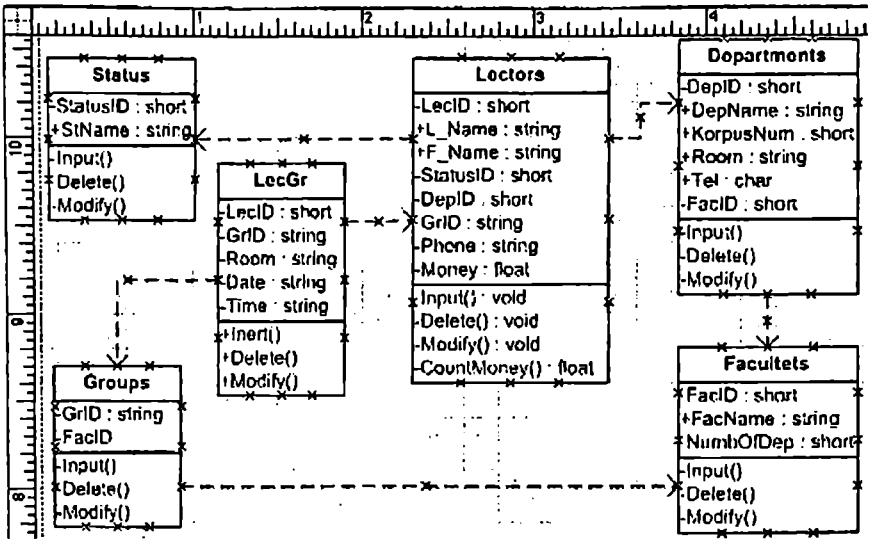
კლასების დიაგრამის ასაგებად გვჭირდება **Static Structure**, რომელიც 4.50 ნახაზზეა ნაჩვენები. ფორმაზე გადმოვიტანთ



ნახ.4.50. კლასების დიაგრამის ინსტრუმენტი

კომპონენტს Class და ჩავწეროთ მასში სახელს.

შემდეგ შევიტანთ ატრიბიტთა (Attributes) დასახელებებს,ტიპებითა და ხილვადობის პარამეტრებით (public, private, protection), კლასის მეთოდებს (Operation) მის ფუნქციათა სახელებისა და ხილვადობის პარამეტრებით. ამგვარად აიგება **Class-Assotiation** დიაგრამის სხვა კლასებიც და დამთავრებისთანავე ეკრანზე გამოჩნდება 4.51 ნახაზზე



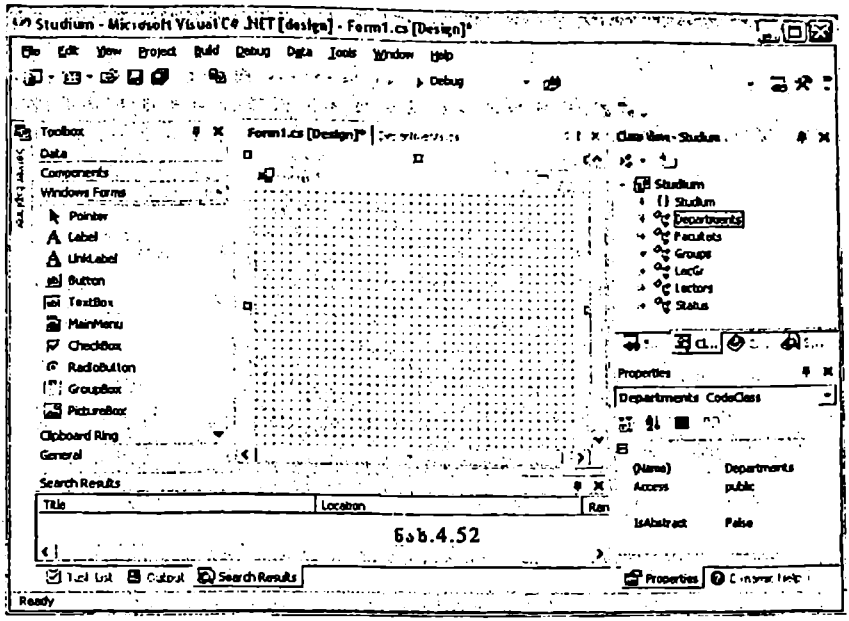
ნახ.4.51.

მოცემული სურათი.

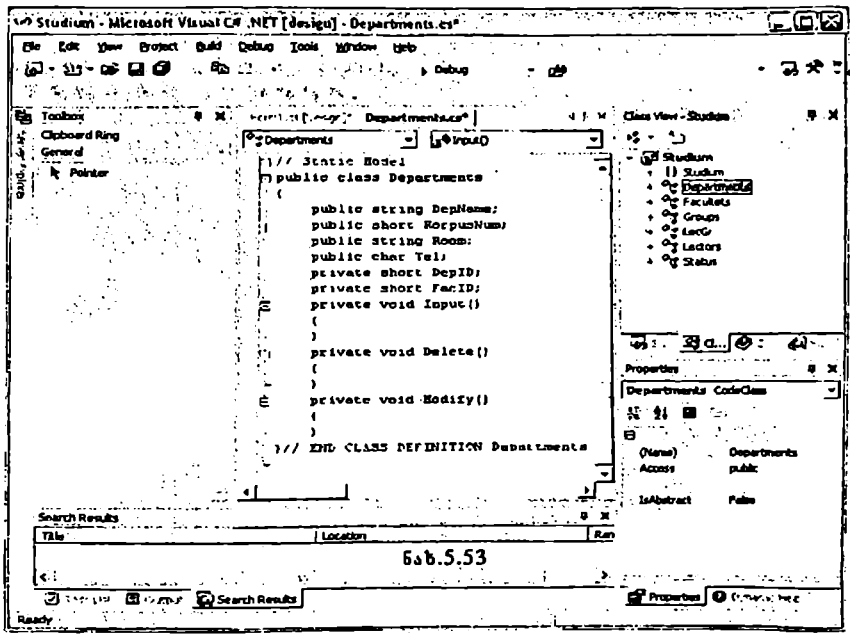
პროგრამული კოდის ავტომატური გენერაცია შესაძლებელია კლასთა კავშირების დიაგრამის აგების შემდეგ. მენიუდან ავირჩევთ UML | Code | Generate, რის შემდეგაც სისტემა შემოგვთავაზებს დაპროგრამების ენის არჩევას (C# , C++, Visual Basic). აქვე უნდა ავირჩიოთ პროექტის სახელი (Name), Browse-ს გამოყენებით პროგრამული მოდულების ჩასაწერი კატალოგი (Location), მაგ. : D:\Gia\Visio-1\ და მარჯვენა ნაწილში ამოვირჩიოთ კლასები,რომელთა დაპროგრამებასაც ვაპირებთ. ნახაზზე ყველა კლასია მონიშნული. დავამოწმოთ შედეგი ღილაკით Ok.

გენერატორი შექმნის ჩვენი კლასთა კავშირების შესაბამისი დიაგრამის პროგრამულ კოდს C#-ენაზე (ასევე შეეძლო მას C++, Visual Basic ან სხვ. ენის გამოყენება).

.NET პლატფორმის სამუშაო გარემო მოცემულია 4.52 ნახაზზე. Department-კლასის არჩევით გამოვა მისი პროგრამული კოდი (ნახ.5.53).



Бъл.4.52



Бъл.5.53

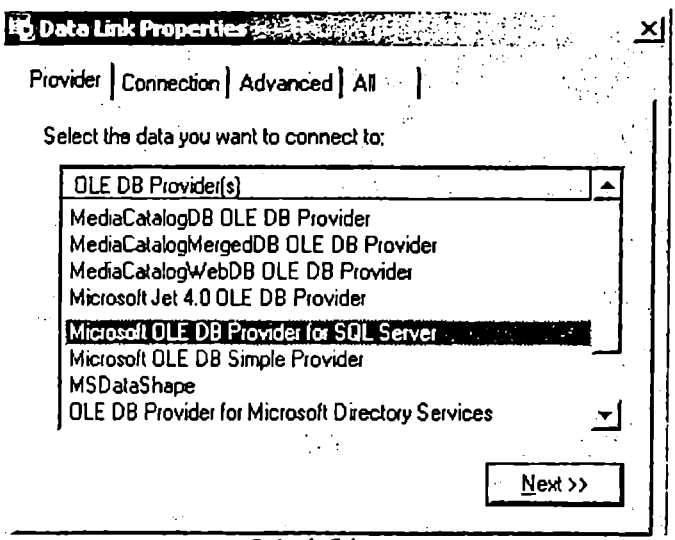
4.52 ნახაზზე ნაჩვენებია .NET-პლატფორმის სამუშაო გარემო, სადაც შესაძლებელია მომხმარებელთა ინტერფეისის აგება და მონაცემების გამოტანა ბაზიდან. განვიხილოთ ეს საკითხი.

დავუშვათ, რომ არსებობს რომელიმე პროგრამულ პაკეტში აწყობილი მონაცემთა ბაზა, მაგ., MsAccess ან MsSQL_Server.

მოხმარებლის ინტერფეისის დასაპროგრამებლად და მონაცემთა ამოსაღებად მითითებული ბაზიდან, საჭიროა პროგრამული პაკეტის ADO.NET გამოყენება. ამისათვის .NET-ში (ნახ4.52) მენიუდან ავირჩიოთ:

View | Server Explorer.

ეკრანის მარცხენა ზედა ნაწილში Data Connections-ზე მაოსის მარჯვენა ღილაკით ავირჩიოთ Add Connection. გამოჩნდება დამხმარე ფანჯარა Data Link Properties, რომელშიც უნდა შევირჩიოთ ჩვენთვის

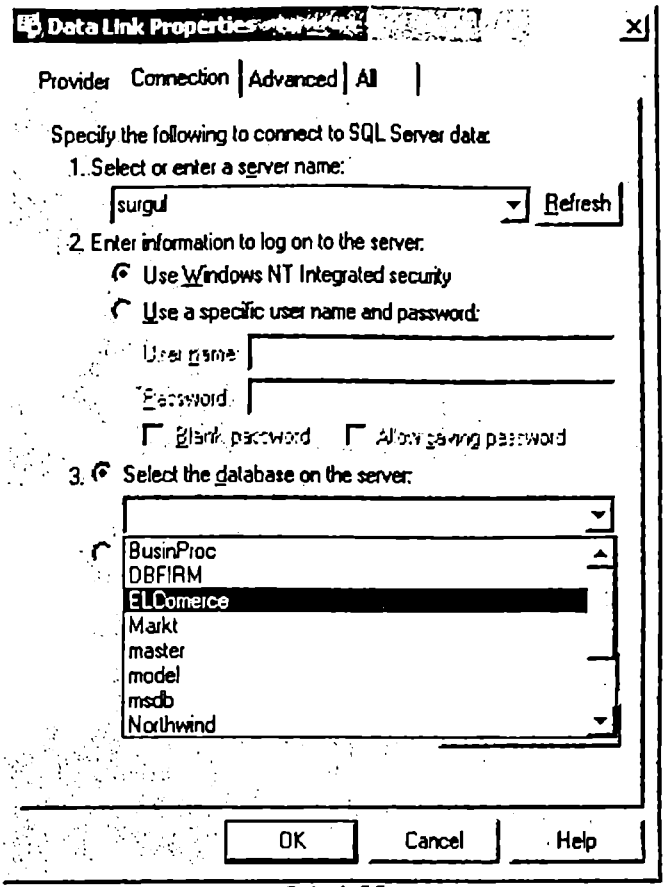


ნახ.4.54

საჭირო მონაცემთა ბაზის დრაივერი (ნახ.4.54).

აქ ნაჩვენებია MsSQLServer-ის დრაივერის მაგალითი. თუ საჭიროა MsAccess, მაშინ ავირჩევთ Microsoft Jet 4.0 OLE DB-ს.

დრაივერის არჩევის შემდეგ Connection გვერდზე (ნახ.4.55)

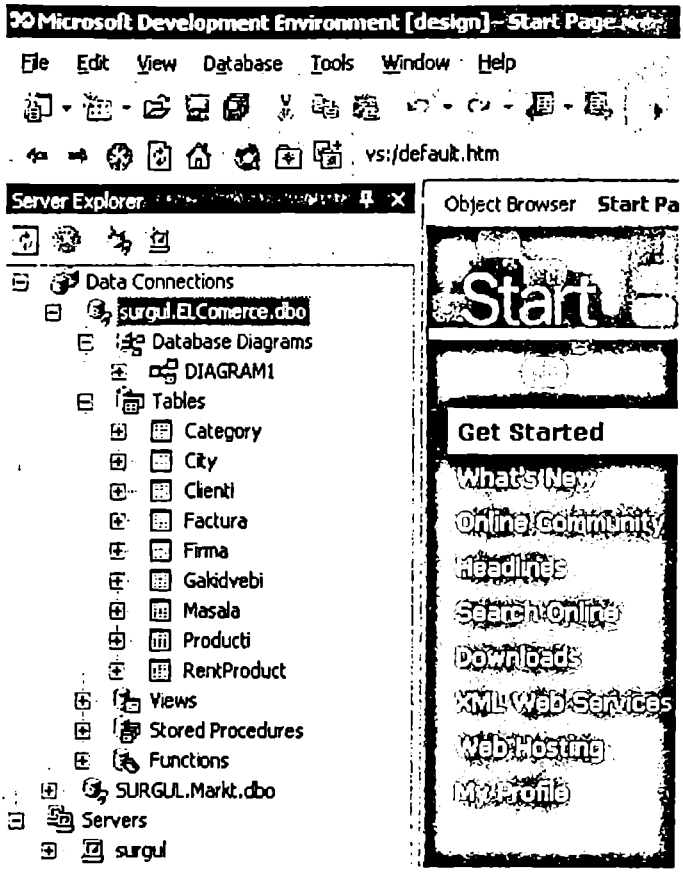


ნახ.4.55

შევირჩევთ სერვერის სახელს (მაგ.,surgul),გადავრთავთ Use Windows NT security და ავირჩევთ მონაცემთა ბაზას (მაგ., ElComerce).

მივიღებთ 4.56 ნახაზზე მოცემულ კადრს, რომელზედაც ჩანს MsSQLServer-ში აგებულ E_Comerce მონაცემთა ბაზის ცხრილები (Tables) და დიაგრამა (DIAGRAM1). ამგვარად, დაკავშირება ბაზასთან განხორციელდა.

თუ გავხსნით DIAGRAM1-ს, მაშინ გამოჩნდება ADO.NET-ში ასახული ElComerce-ბაზის ცხრილთაშორის კავშირების სქემა (ნახ.4.57).

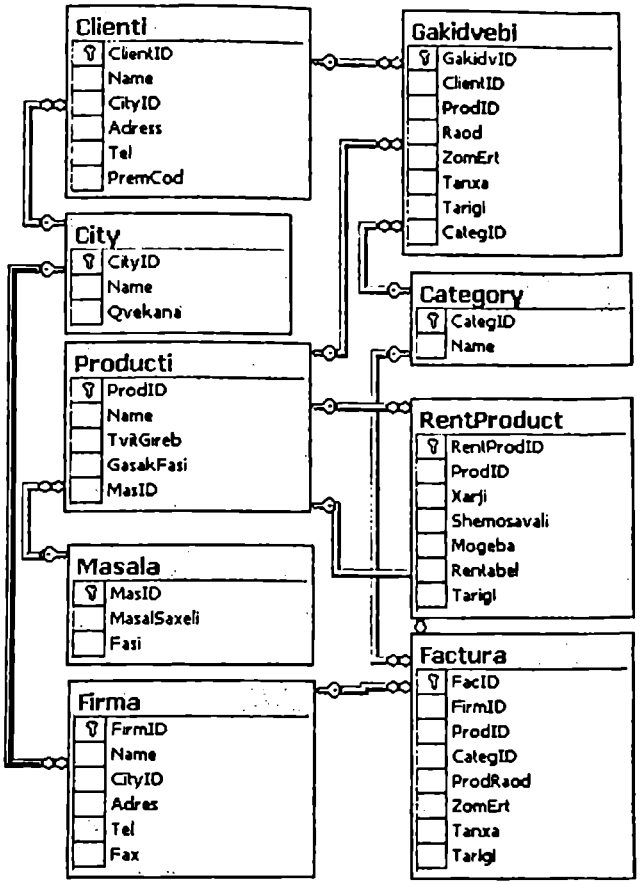


ნახ.4.56

შემდეგ ეტაპზე 4.52 ნახაზის ფანჯარაში მენიუდან ვირჩევთ: File | Add New Item | Data Form Wizard.

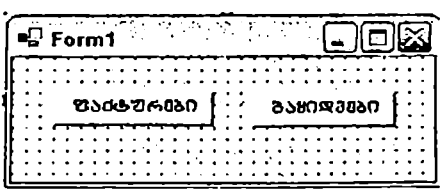
ამით შესაძლებელია ფორმის აგება და მონაცემთა გამოტანა ოსტატი-პროგრამით (Wizard). ეს პროგრამა თვითონ წარმართავს დიალოგს, რაც აადვილებს მუშაობას. აქ მთავარია ჩვენ სწორად შევირჩიოთ ცხრილები და ცხრილთაშორისი კავშირები.

ფორმაზე Form1 ინსტრუმენტების პანელიდან გადმოვიტანოთ ორი ღილაკი (button1, button2). მოვნიშნოთ პირველი ღილაკი და Properties-ში შევცვალოთ მისი თვისებები.



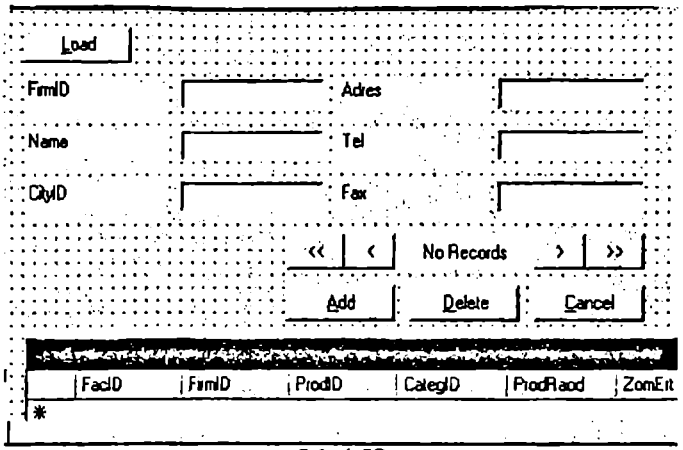
ნახ.4.57

მაგ., ქართული შრიფტი (Font ->LitMtavrPS-ით) და წარწერა „ფაქტურები“ (Text). 4.58 ნახაზზე ნაჩვენებია ეს შემთხვევა.



ნახ.4.58

4.59 ნახაზზე ნაჩვენებია მეორე ფორმა DataForm1, რომელიც ფირმისა და ფაქტურების ცარიელი ცხრილებია.

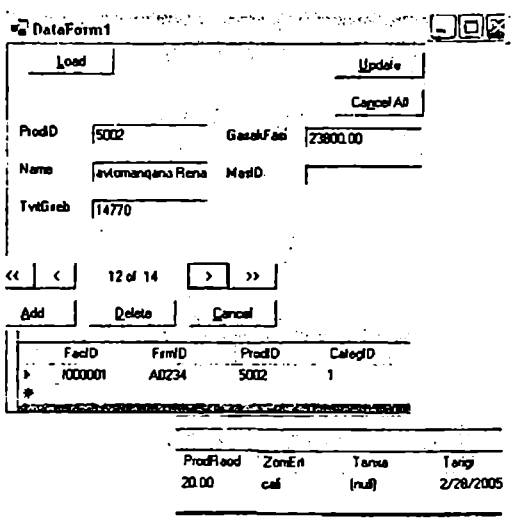


ნახ.4.59

პირველი ფორმიდან რომ გამოვიძახოთ მეორე ფორმა, საჭიროა ღილაკზე მაოსით 2-ჯერ დავაწკაპუნოთ და C# კოდის ტექსტში, სადაც შეჩერებულია კურსორი (button1_Click) ჩავწეროთ ხელით მე-3 და მე-4 სტრიქონები:

```
private void button1_Click(object sender, System.EventArgs e)
    { // შესატანი ტექსტის ორი სტრიქონი:
    DataForm1 ob=new DataForm1( );
     ob.Show( );
    }
```

ამის შემდეგ Standard-პანელის Start-ღილაკით შევასრულებთ კოდის კომპილირებას და ეკრანზე გამოვა 1-ელი ფორმა, რომლის „ფაქტურის“ ღილაკის არჩევით მივიღებთ მე-2 ფორმას (შეუვსებელი). აქ საჭიროა Load-ის არჩევა და მონაცემები განლაგდება

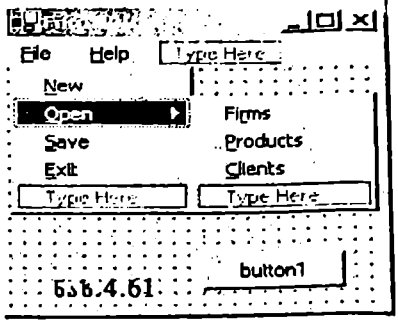


ნახ.4.60

ფორმაზე (

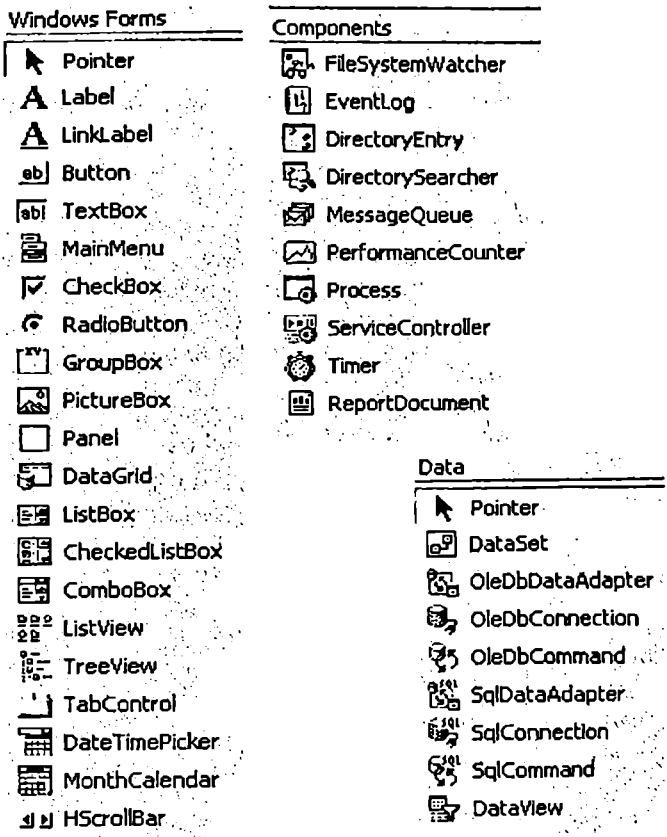
ღილაკის მსგავსად ფორმაზე

შესაძლებელია სხვა ვიზუალური კომპონენტების გადმოტანაც. მაგ., მთავარი მენიუს შექმნა (ნახ.4.61).



ნახ.4.61

4.62 ნახაზზე მოცემულია .NET-ის სამუშაო გარემოში ფორმების, მონაცემთა და კომპონენტთა ისტრუმენტების პანელები. მათი გამოყენება ეფექტურ შედეგს იძლევა.



ნახ.4.62

- ActiveX – კომპონენტური ობიექტების მოდელის (COM) ელემენტთა (კომპონენტების) შესაქმნელი ინსტრუმენტი, მაგალითად, button, Grid და სხვ., მისი შემადგენელი ნაწილია OLE (Object Linking and Embeding – ობიექტების დაკავშირება და ჩანერგვა)
- ADO (ActiveX Data Objects) – მონაცემებთან მიმართვის ტექნოლოგია, დამუშავებული ფირმა მაიკროსოფტის მიერ კომპიუტერული დანართების (აპლიკაციების) ასაგებად COM ტექნოლოგიით. (არ გააჩნია XML-მონაცემთა დამუშავების საშუალებანი)
- ADO.NET – მონაცემებთან. მიმართვის ახალი ტექნოლოგია მაიკროსოფტის Visual_Studio-7.0 პაკეტში აპლიკაციების ასაგებად CLR ტექნოლოგიით (შეუძლია XML-მონაცემთა დამუშავება)
- ad hoc query - არარეგლამენტირებული მოთხოვნა (არ ინახება მონაცემთა ბაზაში)
- aggregation - აგრეგაცია
- Applied Process - გამოყენებითი პროცესი
- API (Application Programming Interface) – გამოყენებითი დანართების დაპროგრამების ინტერფეისი (Windows Application)
- ASP (Active Server Pages) - ვებ-დანართების დაპროგრამების ინსტრუმენტი (Web-Application)
- ASP.NET - ვებ-დაპროგრამების ახალი პლატფორმა, რომელიც უზრუნველყოფს ვებ-დანართების აგებას მონაცემთა ბაზებითა და ADO.NET ტექნოლოგიით
- ATL (ActiveX Template Library) – შაბლონების ბიბლიოთეკაა, რომელიც გამოიყენება ActiveX–ის მართვის ელემენტების ასაგებად
- B2B (Business-to-Business) – ბიზნესის სახეობა (ელექტრონული კომერცია) საწარმოებს შორის. ასეთი სისტემები ძირითადად გამოიყენება წარმოების ნედლეულით მომარაგებისა და მზა პროდუქციის გასაღებაში
- B2C (Business-to-Consumer) - ბიზნესის სახეობა (ელექტრონული კომერცია) საწარმოებსა და კლიენტებს შორის. მყიდველები წარმოადგენენ კერძო პირებს, როგორიცაა ინტერნეტ-მაღაზია, ფასიანი საინფორმაციო სამსახური და ა.შ.

- Internet provider - პირი ან ორგანიზაცია, რომელიც ემსახურება ინტერნეტის ქსელში ჩართვას
- Internet Proxy - სერვერი ახორციელებს ინტერნეტიდან მონაცემთა შენახვას განმეორებითი მოთხოვნისთვის
- Internet - მსოფლიო საინფორმაციო კომპიუტერული ქსელი
- Intranet - კომპიუტერების კორპორაციული ქსელი ან ლოკალური გამოთვლითი ქსელების ჯგუფი, რომელთაც კავშირის საერთო პროტოკოლი აერთიანებთ
- MOLAP - მონაცემთა მრავალგანზომილებიანი ანალიზი
- ODBC (Open DataBase Connectivity) – მონაცემთა ბაზებთან მიმართვის ღია ინტერფეისი
- OLAP (Online Analytical Processing) - მონაცემთა ოპერატიული ანალიზი
- OLE DB – სხვადასხვა ტიპის მონაცემებთან მიმართვის ტექნოლოგია COM-ში
- Remote Data Objects (RDO) – დისტანციური მონაცემთა ობიექტები
- Restore - გახსნა (Backup-ით მიღებული ფაილებისათვის)
- Proxy - სერვერი არეგულირებს კომპანიის თანამშრომლებს შორის ინტერნეტ-რესურსების გადანაწილებას
- SQL (Structured Query Language) - მონაცემთა სტრუქტურირებადი ენა
- SQL Server – მონაცემთა ბაზის მართვის სისტემა (მაიკროსოფტის)
- SSL (Secure Sockets Layer) - ინტერნეტში ელექტრონული ტრანზაქციების ჩატარება (მონაცემთა უსაფრთხო გადაცემა ქსელში)
- Timed Transition Petri Net - ტრანზიტული დროითი პეტრის ქსელი
- UML - უნიფიცირებული მოდელირების ენა
- UNIX – ქსელური ოპერაციული სისტემა
- VPN (Virtual Private Network) - ვირტუალური კერძო ქსელი
- Web – ქსელი, რომელიც უზრუნველყოფს კლიენტების კავშირს ინტერნეტში
- XML (eXtensible Markup Language) - გაფართოებადი დაფორმატების ენა, აქვს უნივერსალური ბაზური ფორმატი მონაცემთა გაცვლისათვის
- XSL (eXtensible Stylesheet Language) - სტილური ცხრილების გაფართოებადი ენა, რომლითაც აღიწერება XML დოკუმენტი

- connected object – მიერთებადი ობიექტი
- constraint – შეზღუდვა
- Data definition Language (DDL) - მონაცემთა აღწერის ენა
- Data Manipulation Language (DML) - მონაცემთა მანიპულირების ენა
- data warehouse - მონაცემთა საცავი
- descriptor- პაროლი ან კოდური სიტყვა, რომლის მეშვეობითაც ხდება იდენტიფიკაცია
- DMZ – დემილიტარიზებული ზონა
- e-businss - ელექტრონული ბიზნესი
- E-Commerce - ელექტრონული კომერცია
- extendend business - ბიზნესი საზღვრების გარეშე
- FCFS (First Come, First Served) - პირველი მოვიდა პირველი მომსახურდა
- Firewall - ქსელური ეკრანი ('ცეცხლოვანი კედელი"), რომელიც წარმოადგენს დამცავ საშუალებას ლოკალურ და გლობალურ ქსელს შორის
- GPSS (General-Purpose Simulation System) – მოდელირების უნივერსალური სისტემა, GPSS-ენა
- GUID (Globally Unique Identifier) – გლობალურად უნიკალური იდენტიფიკატორი. გამოიყენება COM-ობიექტების ინტერფეისთა უნიკალური სახელებისათვის
- Hypertext – ჰიპერტექსტი. დიდი ტექსტების იერარქიული განლაგების ხერხი, გამოიყენება ინტერნეტულ სისტემებში
- HOLAP - ჰიბრიდულ მონაცემთა ანალიზი
- HTML (HyperText Markup Language) - ჰიპერტექსტების დაფორმატების ენა, რომელიც გამოიყენება Web-გვერდების ასაგებად
- HTTP (HyperText Transfer Protocol) - ინტერნეტში ჰიპერტექსტების გადაცემის პროტოკოლი
- HUB - ცენტრალური რგოლი ან კვანძი, სადაც თავს იყრის ყველა წრედი (ვარსკვლავური ტოპოლოგიის ქსელში)
- IETF (Internet Engineering Task Force)- ორგანიზაცია, რომელიც ამუშავებს და ამტკიცებს ინტერნეტ ქსელის ყველა სტანდარტსა და პროტოკოლს
- Internet provider - პირი ან ორგანიზაცია, რომელიც ემსახურება ინტერნეტის ქსელში ჩართვას

- Internet Proxy - სერვერი ახორციელებს ინტერნეტიდან მონაცემთა შენახვას განმეორებითი მოთხოვნისთვის
- Internet - მსოფლიო საინფორმაციო კომპიუტერული ქსელი
- Intranet - კომპიუტერების კორპორაციული ქსელი ან ლოკალური გამოთვლითი ქსელების ჯგუფი, რომელთაც კავშირის საერთო პროტოკოლი აერთიანებთ
- MOLAP - მონაცემთა მრავალგანზომილებიანი ანალიზი
- ODBC (Open DataBase Connectivity) – მონაცემთა ბაზებთან მიმართვის ღია ინტერფეისი
- OLAP (Online Analytical Processing) - მონაცემთა ოპერატიული ანალიზი
- OLE DB – სხვადასხვა ტიპის მონაცემებთან მიმართვის ტექნოლოგია COM-ში
- Remote Data Objects (RDO) – დისტანციური მონაცემთა ობიექტები
- Restore - გახსნა (Backup-ით მიღებული ფაილებისათვის)
- Proxy - სერვერი არეგულირებს კომპანიის თანამშრომლებს შორის ინტერნეტ-რესურსების გადანაწილებას
- SQL (Structured Query Language) - მონაცემთა სტრუქტურირებადი ენა
- SQL Server – მონაცემთა ბაზის მართვის სისტემა (მაიკროსოფტის)
- SSL (Secure Sockets Layer) - ინტერნეტში ელექტრონული ტრანზაქციების ჩატარება (მონაცემთა უსაფრთხო გადაცემა ქსელში)
- Timed Transition Petri Net - ტრანზიტული დროითი პეტრის ქსელი
- UML - უნიფიცირებული მოდელირების ენა
- UNIX – ქსელური ოპერაციული სისტემა
- VPN (Virtual Private Network) - ვირტუალური კერძო ქსელი
- Web – ქსელი, რომელიც უზრუნველყოფს კლიენტების კავშირს ინტერნეტში
- XML (eXtensible Markup Language) - გაფართოებადი დაფორმატების ენა, აქვს უნივერსალური ბაზური ფორმატი მონაცემთა გაცვლისათვის
- XSL (eXtensible Stylesheet Language) - სტილური ცხრილების გაფართოებადი ენა, რომლითაც აღიწერება XML დოკუმენტი

ლიტერატურა

1. ბოლხი გ., სურგულაძე გ., პეტრიაშვილი ლ., ჩიხრაძე ბ. მულტიპროცესორული სისტემების რესურსების მართვის პროგრამული უზრუნველყოფის დამუშავება Borland_C++Builder ინსტრუმენტით// სტუ-ს შრომები, № 4(437), თბილისი, 2001.

2. ბოტჰე კ., სურგულაძე გ., დოლიძე თ., შონია ო., სურგულაძე გ. თანამედროვე პროგრამული პლატფორმები და ენები // თბილისი, 2003.

3. გოგიჩაიშვილი გ., სურგულაძე გ., შონია ო. დაპროგრამების მეთოდები // თბილისი, 1997.

4. გოგიჩაიშვილი გ., სურგულაძე გ., დოლიძე თ., შონია ო., პოჩოვიანი ს., თურქია ე., პეტრიაშვილი ლ., გულუა დ. ეკონომიკური რეფორმების მართვის სრულყოფა ინფორმაციული ტექნოლოიით // სტუ-ს შრომები, №4(446), თბილისი, 2002.

5. გოგიჩაიშვილი გ., სურგულაძე გ., შონია ო. დაპროგრამების მეთოდები C&C++ ენებზე. თბილისი: ტექნიკური უნივერსიტეტი. 1997.

6. ვაჩნაძე რ., თურქია გ., კიკვაძე ტ., ლომსაძე პ. ბიზნესი ESM-თბილისი, 2003 წ.

7. კაკუბავა ი., კუბეცია ი., პეტრიაშვილი ლ., უსენაშვილი ნ. ბიზნეს პროცესების მოდელირება მასობრივი მომსახურების ჩაკეტილი სისტემებით სტუ-ს შრომები, № 12(451),2005.

8. პეტრიაშვილი ლ. მონაცემთა საცავის ინფორმაციული ბლოკების დამუშავების პროცესების ასახვა მარშალის მრუდის გამოყენებით// საქ.მეცნიერებათა აკადემია, შრ. კრ. №7,თბილისი,2003.

9. პეტრიაშვილი ლ. განაწილებულ სისტემებში მონაცემთა საცავის ინფორმაციული ბლოკების დამუშავების პროცესების

ასახვა პეტრის ქსელების გამოყენებით სტუ-ს შრომები № 1(451), თბილისი, 2004.
10. რეისიგი ვ., სურგულაძე გ., გულუა დ., ვიზუალური ობიექტ-ორიენტირებული დაპროგრამების მეთოდები // თბილისი, 2002.
11. რეისიგი ვ., სურგულაძე გ., დოლიძე თ., პეტრიაშვილი ლ. გულუა დ. საავადმყოფოთა რესტრუქტურიზაციის პროცესის მართვა პეტრის ქსელების თეორიისა და მონაცემთ ბაზის სერვერის იდეოლოგიის საფუძველზე// სტუ-ს 80-წლისთავისადმი მიძღვნილი პროფესორ-მასწავლებელთა ღია საიუბილეო სამეცნიერო-ტექნიკური კონფერენციის თეზისები, თბილისი, 2002.
12. სურგულაძე გ., შონია ო., პეტრიაშვილი ლ. კომპიუტერული ქსელის რესურსების სინქრონიზების პროცესის მოდელირება პეტრის ქსელებით მრავალმომხმარებლურ რეჟიმში // შრ.კრ.„ინტელექტი“ №1, თბილისი, 1997.
13. სურგულაძე გ., კაშიბაძე მ. ოპერაციული სისტემები: პროცესების მართვის პეტრის ქსელების თეორიის გამოყენებით. თბ. სტუ. 1993.
14. სურგულაძე გ., პეტრიაშვილი ლ. განაწილებულ სისტემებში მონაცემთა საცავის ინფორმაციული ბლოკების დამუშავების პროცესების ასახვა პეტრის ქსელების გამოყენებით//
სტუ-ს შრომები, № 1(451), თბილისი, 2004
15. სურგულაძე გ., პეტრიაშვილი ლ. კუბში მონაცემთა აგრეგაცია გრაფების თეორიის გამოყენებით// ჟურნალი ‘ინტელექტი“ №3(20),თბილისი, 2004.
16. სურგულაძე გ., მონაცემთა ბაზების სამაგიდო სისტემები (დაპროექტება-რეალიზაცია). სტუ, 2004.

17. სურგულაძე გ., შონია ო., ყვავაძე ლ. მონაცემთ განაწილებული ბაზების მართვის სიტემები. სტუ., 4004.
18. სურგულაძე გ., დაპროგრამების ვიზუალური მეთოდები და ინსტრუმენტები (UML, MsVisio, C++Builder). stu,2005.
19. სურგულაძე გ., ყვავაძე ლ., მონაცემთა სტრუქტურების და ფაილების დამუშავება C ენაზე. სტუ,2004.
20. ჩოგოვაძე გ., ინფორნაცია // თბილისი, 2003.
21. ჩოგოვაძე გ., გოგიჩაიშვილი გ., სურგულაძე გ., შეროზია თ., შონია ო. სტუ. თბილისი 2001.
22. ჩოგოვაძე გ., სურგულაძე გ. რელაციური ალგებრის ოპერაციების შესრულების ეფექტური პროცედურის აგების ერთი ინსტრუმენტის შესახებ მონაცემთა ბაზებში. საქ.მეცნ.აკად. „მოამბე“ 148-№3, 1993.
23. ჩოგოვაძე გ., სურგულაძე გ., შონია ო. მონაცემთა და ცოდნის ბაზების აგების საფუძვლები. თბ., განათლება, 2001.
24. ძაძამია რ., ხარატიშვილი ზ. მმართველობითი ფინანსური აღრიცხვა. ბაფ. 2003. თბილისი.
25. წერეთელი გიორგი. ეკონომიკის მათემატიკური მოდელირების მეთოდოლოგიური საფუძვლები „მეცნიერება“ თბილისი 2000წ.
26. ჯენსონი ე.ა., მოვსესიანი ა.გ., ოგნივცევი ს.ბ. რა არის ეკონომიკა. (თარგ. რუსულიდან თბილისი 2003) მსოფლიო ეკონომიკა ISBN 99928-0-637-0 მოსკოვი 2003წ.
27. ВВЕДЕНИЕ В ТЕОРИЮ МАССОВОГО ОБСЛУЖИВАНИЯ. Москва "Наука" Главная редакция Физико—Математической Литературы 1978
28. ИВЧЕНКО Г.И., КАШТАНОВ В.А., КОВАЛЕНКО И.Н. ТЕОРИЯ МАССОВОГО ОБСЛУЖИВАНИЯ. Москва "Высшая школа" 1982

29. Язык XML — практическое введение (www.citforum.ru). Шелли Пауэрс. Чего мы ждем от XML?" Мир ПК", № 3, Москваб 1998.
30. Какубава Р.В. Баиашвили З.А. Проблема снижения экономических потер в многоканальних системах. Труди ГТУ #4 (450, 2003).
31. Какубава Р.В., Хуродзе Р.А. О надежностном планировании технических систем: Математически Подход. Вестник РАЕН, — 4 М. 2003.
32. Кроув Т., Эйвисон Д. Бдзы Данных в административных информационных системах. Москва 'Финанси и статистика˜ 1998.
33. Томашевский В., Жданова Е. Имитационное Моделирование в среде GPSS. УДК 681.32 ISBN 5—98158—004—6 Москва 2003.
34.Теория сетей петри и моделирование систем Джю Питерсон. Москва, Мир. 1984.
35. Albrecht, J., Hummer, W., Lehnen W., Using Semantics for Query Derivbility in Data Warehouse Applications. (FQAS, 2000 Warschau)
36. Albrecht, J., Grundlagen der Anfrageferarbeitung in relationalen Datenbanksystemen
37. Albrecht J. Anfrageoptimierung in Data -Warehouse – Szstemen auf Grundlage des multidimensionalen Datenmodells. Friedrich-Alexander-Universitdt Erlangen – Nürnberg, 2003.
38. Albrecht J. Anfrage optimierung in Data-Warehause-Systemen auf grundlage des multidimensionalen

Datenmodele.// Fridrich-Alexander-Universitet, Erlangen-Nurnberg. 2002.
39. Bauer A., Gunzel G. Data Warehouse, Architektur, Entwicklung, Anwendung . Dpunkt Verlag, Heidelberg,2000.
40. Bolch G., Greiner S, deMeer H., Trivedi K.S., "Queueing Networks and Markov Chains", JOHN WILEY&SONS, INC.
41. Both K., Reverse Engineering: the Challenge of Large-Scale Real-World Educational Project, Conference on Software Engineering Education and Traning, Charlotte, USA, Febr. 2001.
42. Both K., Scklews U. Praxisn „ahe druch Reverse Engineering-Projekte: Erfahrungen und Ver-allgemeinerungen, SEUH 2001, Z" urich, Febr. 2001.
43. Brown S., Wilde N., Carlin J. A Software Maintenance Process Architecture, 9th Conference Software Engineering Education and Training, Daytona Beach, 1996.
44. Bruegge B. From Toy Systems to Real Software Development: Improvements in Software Engineering Eduction "SEUH" 94.
45. Bauer A. Managment of multidimensional Aggregates for efficient online Analytical Processing.// Montreal, Canada 1999,S. 156-164.
46. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 1996.
47. Chuck K. W. H. Inmon. Rdb/Vms: Developing the Data Warehouse. Author: Chuck Kelley, W. H. Inmon. Rdb/Vms: www.freisslersoft.com.

48. Data Warehousing and **OLAP**. A comprehensive list of Data Warehouse and **OLAP** papers (with link to full paper), links, and conferences. www.freisslersoft.com.
49. *Data Warehouse* von Barry Devlin.
50. *Data Warehouse Design Solutions, w. CD-ROM* von Christopher Adamson, Michael Venerable
51. Data warehousing von Schlesinger,L.
52. .Database Engineering von Cabibbo,L.
53. Database Management - General von Bello,R., Dias,K.,Downing,A.
54. Data Warehousing and **OLAP**. A comprehensive list of Data Warehouse and **OLAP** papers (with link to full paper), links, and conferences.
55.E-commerce-tp://www.microsoft.com/rus/sql/casestudies/ecommerce_site.asp] http://www.aqtime.ru/econs/ebiz.html (Universitat Erlangen-Nurnberg 2002)
56. http://www.OLAP report.com/fasmi.htm
57. http://www.lanoffice.ru/middle_net.html
58. http://www.lanoffice.ru/mail.html
59. Inmon, W.H.: Building the Data Warehouse ... York 1996. Inmon, W.H.: Building the Operational Data Store ... www.ruckriegel.org
60. Information Center for **OLAP**, Data Warehouse and Business Intelligence.
61. Kleinrock L. Queueing Systems. volume 1, volume 2, JOHN WILEY & SONS, INC
62. **OLAP** Definitions
63. **OLAP** AND **OLAP** SERVER DEFINITIONS. **OLAP**: ON-LINE ANALYTICAL PROCESSING. Defined terms ... This is achieved through use of an **OLAP** Server.

64. Ozsu M.T., Wong K.I., Koon T.M. Systems modeling and analysis using Petri nets. System analysys, modeling, simulation. v. 5, №1, 1988.
65. Pierce A. Teaching Software Engineering Principles using Maintenance-Based Projects, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997.
66. Reisig W. Petrinetye. Eine Einfuhrung. Springer-Verlag, Berlin-Heidelberg. 1986.
67. Reisig W. Das Verhalten verteilten Algorithmen mit Petri Netyen bei unterschiedlicher Vereinfachung des Datenabhängigen Steuerflusses. Berlin IIR, 1990.
68. Slimnick J. An Undergraduate Course in Software Maintenance and Enhancement. 10th Conference on Software Engineering Education and Traning, Virginia Beach, 1997.
69. Sikkel K., Spil T., van de Weg: Replacing a Hospital Information System:an Example Real-World Case Study, 12th Conference on Software Engineering Education and Training, New Orleans 1999.
70. A:\LANOffice Technical Firewall.htm
71. *The Data Warehouse Toolkit* von Ralph Kimball, Margy Ross.
72. http://www.lanoffice Firewall.htm
73. Vetter M. Objektmodellierung. Stuttgart, Teubner, 1995
74. William H. Inmon, John A. Zachman, Jonathan G. Geiger, W. H. Inmon. Data Warehous und OLAP. (TODS 2001 Weimar)
75. www.olapinfo.de/
76. www.ondelette.com/OLAP/dwbib.html
77. Wotey G., Conte P. SQL Server 2000, Developers Guide. Osborn, 2001.
78. Developing Windows-based applications with Microsoft Visual Basic.NET and Visual C#.NET. MIcrosoft Corporation, 2002.
79. C# - искусство программирования. Энциклопедия программиста. Пер. с анг. С-Петербург, ДиаСофт ЮП, 2002

მოკლე კომენტარი C# დაპროგრამების ენისა და .NET პლატფორმის გამოყენების შესახებ

მაიკროსოფტის უახლესი პროგრამული ტექნოლოგია .NET პლატფორმის სახით სულ უფრო ფართოდ იკიდებს ფეხს მსოფლიოს მოწინავე ქვეყნების საუნივერსიტეტო-სამეცნიერო და საწარმოო ფირმების ბიზნესის სფეროებში. იგი გამოიყენება Windows, Unix და Linux ოპერაციული სისტემებისათვის.

.NET პლატფორმა შეიქმნა სპეციალურად განაწილებული გამოყენებითი სისტემების ასაგებად დიდი მოცულობის ინფორმაციის დასამუშავებლად კლიენტ-სერვერ არქიტექტურის ბაზაზე.

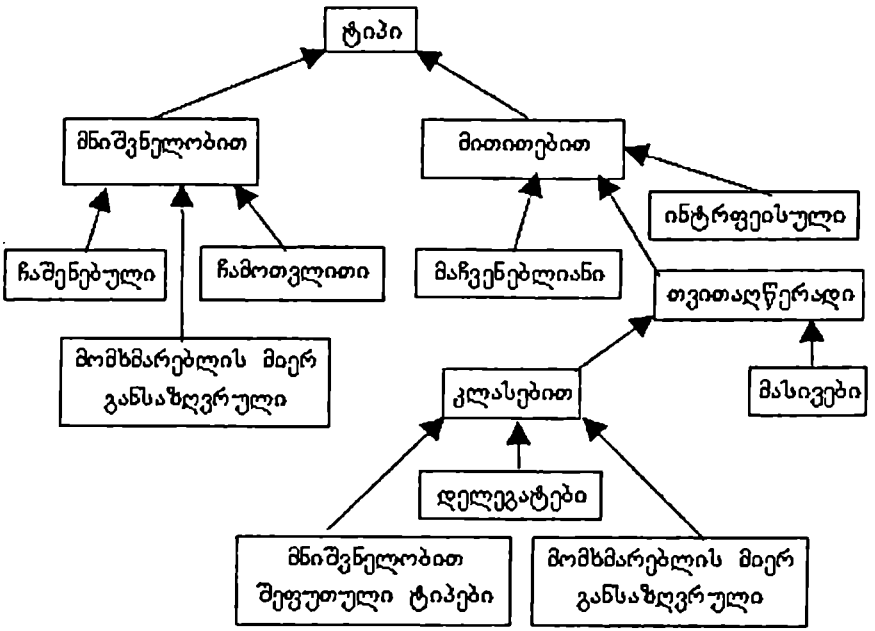
C# („სი შარფ") ენა ობიექტ-ორიენტირებული ენების ერთ-ერთი ახალი და მძლავრი წარმომადგენელია, რომელიც შეიქმნა სპეციალურად .NET პლატფორმისათვის და თავსებადია Windows-ის თანამედროვე ვერსიებთან და ინტერნეტთან. მან შეითვისა თავისი წინამორბედების, C++ და Java ენების საუკეთესო თვისებები და დახვეწა არსებული ნაკლოვანებანი. ამ ენაზე იწერება საიმედო, მოქნილი და მაღალმწარმოებლური ქსელური და ინტერნეტული დანართები, დინამიკური web-გვერდები, მონაცემთა ბაზებთან მიმართვის კომპონენტები,აგრეთვე ტრადიციული ფანჯრული Windoes-დანართები.

წიგნის მე-4 თავში ჩვენ განვიხილეთ ინტერნეტ-მაღაზიის რეალიზაციის მაგალითი, რომლისთვისაც წარმოვადგინეთ C# პროგრამის კოდიც. წანამდებარე დანართში ჩვენ განვიხილავთ სწორედ .NET პლატფორმაზე C# ენის თავისებურებებს (C++ ენასთან შედარებით).

მაგალითად, თუ C++ ენა კომპილირდებოდა ასემბლერულ კოდში, C# კომპილირედება IL (Intermediate Language) შუალედურ ენაში. IL-ის დანიშნულებაა პლატფორმული და ენობრივი დამოუკიდებლობის განხორციელება ობიექტ-ორიენტირებულ გარემოში. Java ენაც უზრუნველყოფს პლატფორმულ (Windows, Unix, Linux) დამოუკიდებლობას, მაგრამ მისი ბაიტ-კოდის შესრულების ეტაპზე იგი ინტერპრეტირდება (IL კი კომპილირდება).

.NET პლატფორმისათვის ენობრივი თავსებადობა ხორციელდება IL-ენაში არსებული ტიპების დიდი რაოდენობით, რომლებიც ორგანიზებულია ტიპთა-იერარქიის ობიექტ-ორიენტირებული პრინციპებით. ამას უზრუნველყოფს ტიპების ზოგადი სპეციფიკაცია (CTS - Common Typs Specification).

ქვემოთ, დ1- ნახაზზე მოცემულია აღნიშნული ტიპების იერარქიათა კლასიკური საილუსტრაციო მაგალითი მემკვიდრეობითობის კავშირის გამოყენებით [78].

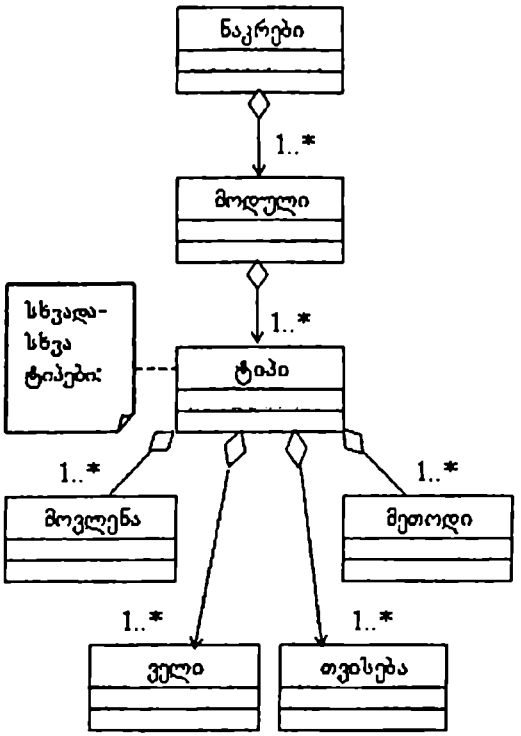


ნახ.-დ1. ტიპების ზოგადი სისტემა

გამოყენებითი კოდების აგება გაადვილებულია .NET პლატფორმის საბაზო კლასების მდიდარი ბიბლიოთეკის არსებობით (იგი გამოიყენებოდა ადრე მაიკროსოფტის მიერ Windows API - Application Programming Interface ამოცანებისათვის).

შეიძლება ითქვას, რომ .NET პლატფორმის საბაზო კლასების დიდი ნაწილი დაწერილია სწორედ C# ენის გამოყენებით.

.NET პლატფორმის კომპონენტებიდან ერთ-ერთი მთავარი ბლოკი Assembly (ანაწყობი, ნაკრები), რომელიც ლოგიკურად აერთიანებს კოდს, რესურსებს და მეტამონაცემებს [79]. დ2-ნახაზზე ნაჩვენებია მისი ზოგადი იერარქიული სტრუქტურა აგრეგაციის კავშირის გამოყენებით.



ნახ.-დ2. ნაკრების ზოგადი სტრუქტურა

პროგრამული ნაკრები შეიძლება იყოს ორი ტიპის: კერძო და საერთო გამოყენების. პირველ შემთხვევაში ნაკრები ინსტალირდება კერძო მომხმარებლს კატალოგში და მასთან სხვა მიმართვები გამორიცხულია. საერთო გამოყენების ნაკრები შეიცავს პროგრამულ ბიბლიოთეკებს, რომელთაც იყენებს სხვადასხვა დანართები. აქ საჭიროა სპეციალური დაცვის მექანიზმების გამოყენება (სახელების კოლიზიის და ნაკრებთა ვერსიების კონტროლის თვალსაზრისით).

კლასებს შორის სახელთა კოლიზიის ალმოფხვრის მიზნით .NET პლატფორმა იყენებს „სახელთა სივრცეს“ (namespace). ესაა მონაცემთა ტიპების უბრალო დაჯგუფება. ყველა მონაცემთა ტიპის სახელს მოცემულ სახელთა სივრცეში ავტომატურად ემატება პრეფიქსი, რომელიც შედგენილია სახელთა სივრცის დასახელებისგან. ასევე შესაძლებელია ჩადგმული სახელთა სივრცეების შექმნა.

წიგნის მე-4 თავში მოცემულია C#-ის პროგრამის ტექსტი. აქ შეიძლება შემდეგი კომენტარის გაკეთება:

მაგ-1:

სახელთა სივრცე	დანიშნულება
using System;	ძირითადი კლასები საერთო გამოყენების ტიპებისათვის
System.Collections;	ინტერფეისები და კლასები, ობიექტთა კოლექციებისათვის
System.Web;	ძირითადი კლასები Web–ში ბრაუზერ-სერვერის კავშირისათვის
System.Web.UI;	მარშუტის ელემენტებისა და Web-გვერდების მხარდაჭერა
System.Web.UI.WebControls;	Web-გვერდების სპეციფიური მარშუტის ელემენტების მხარდაჭერა
System.Web.Security;	ASP.NET–ის დაცვის კლასები
System.ComponentModel;	კომპონენტების ქცევის მარშუტის საშუალებანი შესრულებისა და დაშუშაკების დროს
System.Data;	ADO.NET–ის არქიტექტურის კლასები

მაგ.-2:

```
namespace Magazia.Web  // აქ მითითებულია სახელი Magazia.Web
{
  public class Checkout : PageBase
  {
     // და ა.შ.
```

.NET პლატფორმის მნიშვნელოვანი ელემენტია „დანართთა არეები“. მათი დანიშნულებაა ერთდროულად და ერთმანეთთან მომუშავე დანართების იზოლაცია, რათა არ მოხდეს მონაცემთა არასასურველი დამუშავება. პროგრამული დანართების იზოლაციისათვის Windows გამოიყენებს „პროცესის“ ცნებას, რომელიც მისამართების სივრცეს ეხება. ყოველ პროცესს გამოეყოფა 4 გიგაბაიტი ვირტუალური მეხსიერება. ისინი დისკზე სხვადასხვა ფიზიკური მისამართებითაა და არ გადაიკვეთება. პროცესებს აქვს მინიჭებული განსაზღვრული პრივილეგიები და ოპერაციული სისტემა აკონტროლებს მათ, თუ რომელ ოპერაციას რომელი პროცესის გამოყენება შეუძლია.

დანართთა არეების გამოყენების იდეა მდგომარეობს იმაში, რომ პროცესებს შორის მოხერხდეს მონაცემთა გაცვლა. ამიტომაც პროცესი

იყოფა რამდენიმე დანართის არედ. თითოეულ დანართის არეში თავსდება ერთი დანართის კოდი.

.NET პლატფორმის მნიშვნელოვანი საშუალებაა JIT (Just-In-Time) კომპილატორი. იგი ახორციელებს პროგრამული კოდის ცალკეული ნაწილის დროულად კომპილირებას (საჭიროების შემთხვევაში).

Visual. Studio.NET არის პროგრამული სისტემების დამუშავების ინტეგრირებული გარემო, რომელშიც შესაძლებელია კოდების დაწერა, კომპილირება და გამართვა VB.NET, C++.NET, C#.NET, ASP.NET, ADO.NET და სხვა ტექნოლოგიებით.

ახლა მოვიტანოთ C++ და C# მარტივი კოდები, რათა დავინახოთ მსგავსება-განსხვავება ამ ენების სინტაკსებს შორის:

ა) C++ -ის კოდის ფრაგმენტი:

```
#include <iostream.h> // C++ ————————
#include <Windows.h>
int main(int argc, char *argv)
{
 cout << "Hello, my friend ! ";
MessageBox(NULL, "By-By !", ", MB_OK);
return 0;
}
```

ბ) C#-ის კოდის ფრაგმენტი:

```
using System; // C# ————————
using System.Windows.Forms;
namespace Console1;
{
  class Class1
  {
    static int Main(string[] args)
      {
        Console.WriteLine("Hello, my friend !");
        MessageBox.Show("By-By !");
          return 0;
      }
  }
}
```

.NET სისტემის და C# ენის აღნიშნული და სხვა საკითხები მოითხოვს ცალკე დისციპლინის შესწავლას, რასაც ჩვენ მომავალში დავუბრუნდებით.

იბეჭდება ავტორთა მიერ
წარმოდგენილი სახით

გადაეცა წარმოებას 28.05.2005. ხელმოწერილია დასაბეჭდად
08.06.2005. ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი
თაბახი 12,5. საალრიცხვო-საგამომცემლო თაბახი 11.
ტირაჟი 200 ეგზ. იბეჭდება ავტორთა ხარჯით.

გამომცემლობა „ტექნიკური უნივერსიტეტი“, თბილისი,
კოსტავას 77