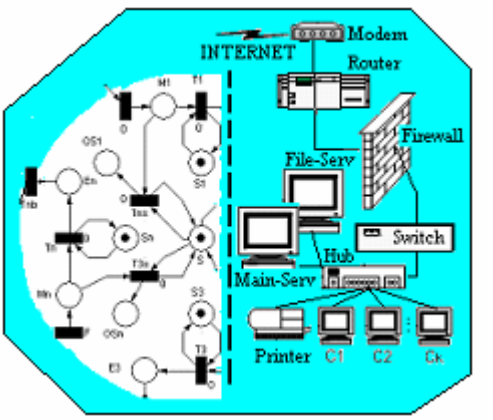
საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, ლია პეტრიაშვილი

განაწილებული ბიზნეს სისტემების მონაცემთა საცავის პროგრამული უზრუნველყოფა (DataWarehouse-Soft)



თბილისი 2007

გია სურგულაძე

სტუ-ს ინფორმატიკის ფაკულტეტის სრული პროფესორი, ტექნიკის მეცნიერებათა დოქტორი, ინფორმატიზაციის საერთაშორისო აკადემიის აკადემიკოსი. არის 190 სამეცნიერო-პედაგოგიური ნაშრომის ავტორი (10 მონოგრაფია, 10 სახელმძღვანელო, 20 დამხმარე სახელმძღვანელო /ლექციების კონსპექტი, 25 ელექტრონული წიგნი სტუ-ს ვებ-გვერდზე, 125 სტატია/თეზისი) მონაცემთა რელაციური ბაზების, ობიექტ-ორიენტირებული დაპროგრამების, მართვის საინფორმაციო სისტემების დაპროექტებისა და პეტრის ქსელების გამოყენებითი თეორიის საკითხებზე. გერმანულ-ქართული ერთობლივი სასწავლო-სამეცნიერო ცენტრის („GeoGer" Center) „ინფოტექნოლოგიები" დირექტორი (ბერლინის ჰუმბოლდტისა და ნიურნბერგ-ერლანგენის უნივერსიტეტებთან), საქართველოს ინფორმატიზაციის ტექნოლოგთა კავშირის თავმჯდომარე, ჟიული შარტავას სახელობის პრემიის ლაურეატი ტექნიკის დარგში.

ლილი პეტრიაშვილი

სტუ-ს ინფორმატიკის ფაკულტეტის „მართვის ავტომატიზებული სისტემების" კათედრის ასოც-პროფესორი, ტექნიკის მეცნიერებათა კანდიდატი, კათედრის გერმანულენოვანი ჯგუფების კურატორი. არის 20 სამეცნიერო ნაშრომის, 1-მონოგრაფიის, 4 დამხმარე სახელმძღვანელოს და ლექციების კონსპექტის (გერმანულ ენაზე) ავტორი ოპერაციული და საოფისე სისტემების, სტატისტიკური მოდელირების, მასობრივი მომსახურების სისტემებისა და პეტრის ქსელების თეორიის საკითხებზე. ასპირანტურაში სწავლების პერიოდში 1995-2000 წლებში ოჯახთან ერთად ცხოვრობდა გერმანიაში, სტაჟირებას გადიოდა ნიურნბერგ-ერლანგენის უნივერსიტეტში, „ოპერაციული სისტემებისა და ქსელების კათედრაზე". პარალელურად დაამთავრა ერლანგენის გერმანული ენის შემსწავლელი ინსტიტუტი, აქვს ატესტატი-სერტიფიკატი. არის ორი შვილის დედა.

სარჩევი

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 [1]. ბოლო პერიოდში იგი ითვლება, განსაკუთრებით პერსპექტიულ და დინამიკურ მიმართულებად საინფორმაცო 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.

შესავალში განხილულია მართვის საინფორმაციო სისტემების აგების თანამედროვე ტექნოლოგიებისა და ინსტრუმენტულ

საშუალებათა დამუშავებისა და სრყლყოფის საკითხების აქტუალობა და მნიშვნელობა. ჩამოყალიბებულია ნაშრომის ძირითადი მიზნები, ამოცანები, მათი გადაწყვეტის გზები და საბოლოო შედეგები.

ნაშრომში 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 .NET პლატფორმის საფუძველზე [2].

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).

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.

gsurg@gmx.net,
liapetri@Rambler.ru.

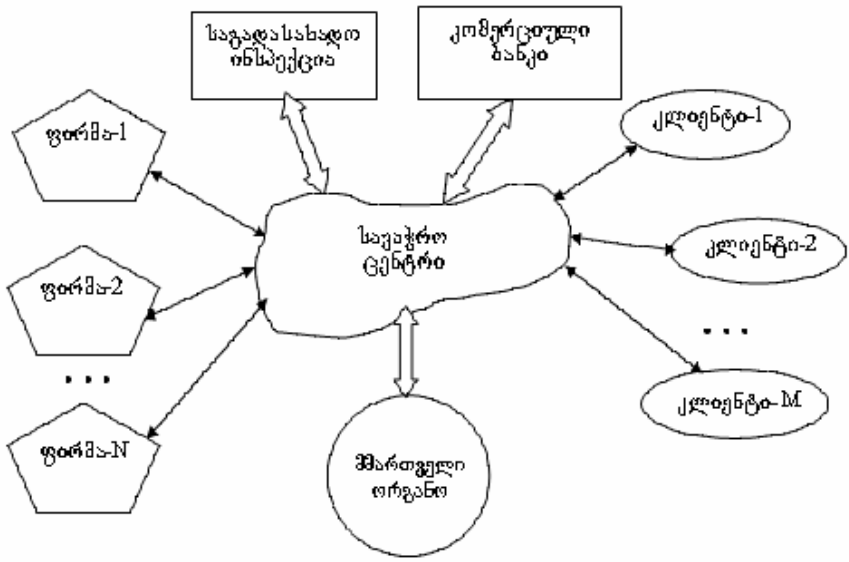
1.1. პროგრამული სისტემის არქიტექტურა და ინფორმაციული ნაკადები

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

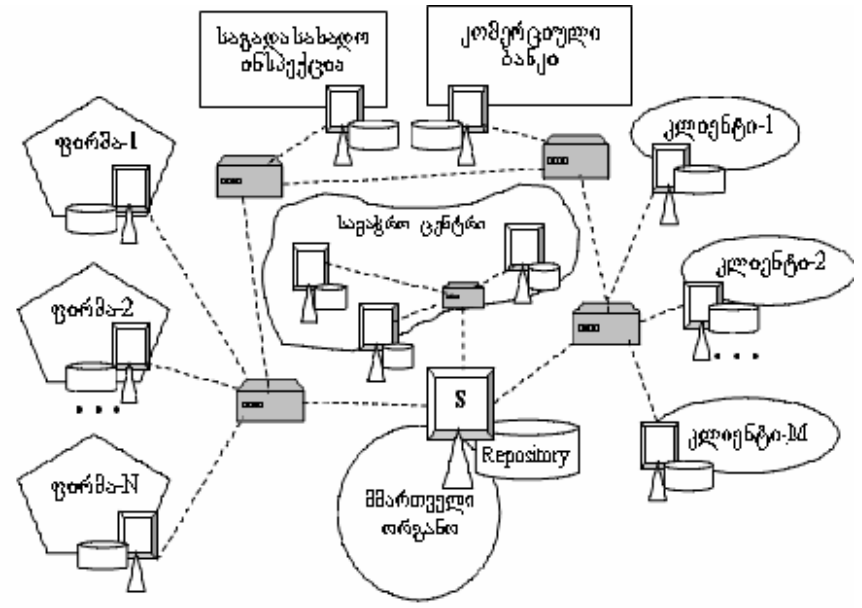
აპარატულში იგულისხმება კომპიუტერული და ქსელური ტექნიკა, რომლის საფუძველზედაც უნდა მოხდეს სისტემის გლობალურ/ლოკალურ ქსელში ფიზიკურად ჩართვის ორგანიზება.

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

1.1 ა და ბ ნახაზებზე მოცემულია კომერციული ბიზნესის ტრადიციული და ელექტრონული კომერციის ზოგადი სტრუქტურული სქემები.



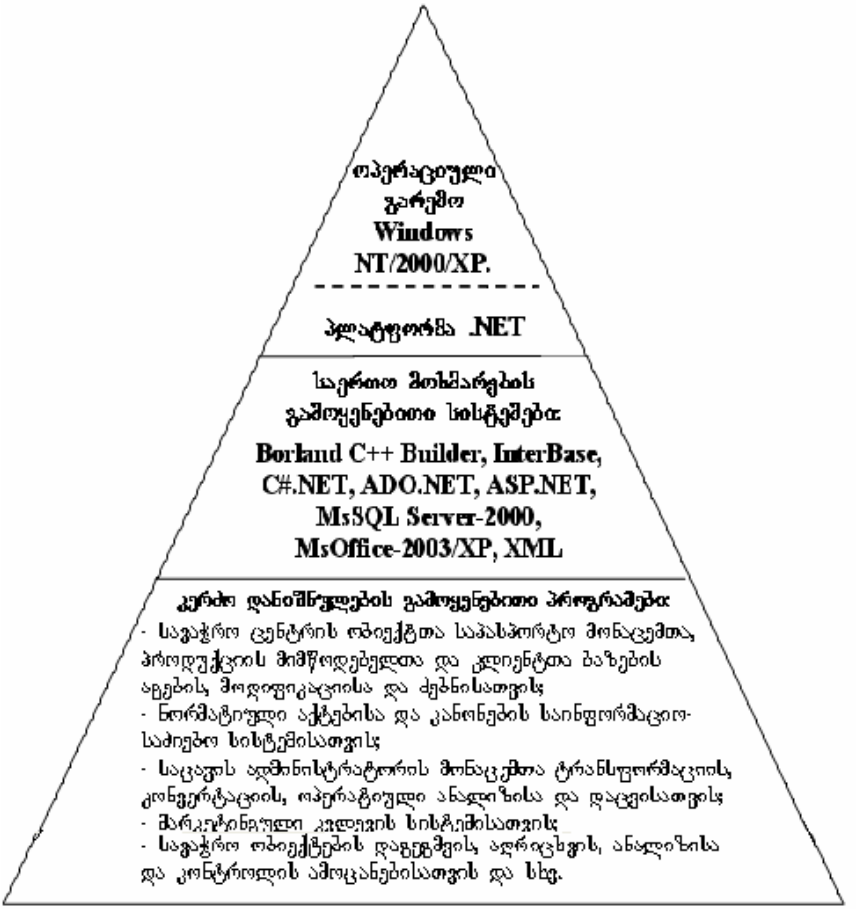
ნახ.1.1-ა. ტრადიციული კომერციის ზოგადი სტრუქტურა



ნახ.1.1-ბ. ელექტრონული კომერციის ზოგადი სტრუქტურა

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.

1.2 ნახაზზე მოცემულია ჩვენი სისტემის ფუნქციონირებისათვის აუცილებელი პროგრამული პლატფორმისა და პაკეტების არქიტექტურა.



ნახ. 1.2

სისტემის მუშაობის პირობები უნდა აკმაყოფილებდეს შემდეგ ძირითად მოთხოვნებს:

- სისტემის ფუნქციონირება უნდა მოხდეს პერსონალური კომპიუტერების ლოკალურ ქსელში ინტერნეტის გამოყენებით;
- სისტემამ უნდა უზრუნველყოს ინფორმაციის უსაფრთხოება და დაცვა;

- სისტემას უნდა ჰქონდეს მომხმარებლებთან ურთიერთობის მეგობრული ინტერფეისი;

- სისტემის მომსახურება უნდა იყოს ადვილი.

დიდ სავაჭრო 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_T: 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.

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

ინფორმაციული ნაკადების მოცულობების მახასიათებლები (ცხრ.1.1)

№	ინფორმაციული ნაკადი	სახე	მოცულობა (საშუალოდ)			
			თვეში მბ	კვარტ. მბ	წელში მბ	ხანგრძლივი, წლებით
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.	საქონელსპრუნვის კვლევები, 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 ცნობები	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

შენიშვნა: ცხრილში ინფორმაცია აღებულია ექსპერტული შეფასებების საფუძველზე.

საანგარიშო ფორმულებად ვიყენებთ შემდეგ გამოსახულებებს:

$$V_{jT} = k_j *, \text{ სადაც}$$

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:

$$V_{jA} = k_j *, \text{ სადაც}$$

A_i – აუდიო ინფორმაციის ფაილის ზომაა;

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

ვიზუალურისათვის შესაბამისად გვექნება:

$$V_{jV} = k_j *, \text{ სადაც}$$

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:

$$S = T_i * K_i *, \text{ სადაც}$$

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

K_i - i-ურ კომერციულ ობიექტზე ფილიალების რაოდენობაა;

- 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 200GB–იანი მეხსიერება (გაითვალისწინება

ძირითადი სტატისტიკური და ისტორიული ფაილების შენახვაც).

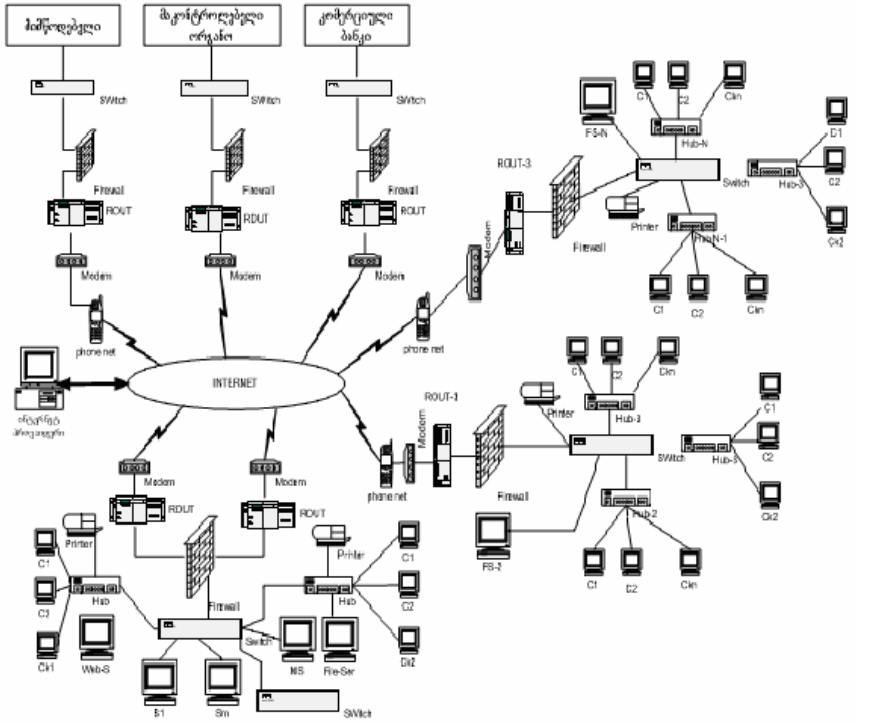
დიდი სავაჭრო ცენტრისათვის (მაღაზიების ქსელი) ეს რიცხვი უნდა გამრავლდეს ობიექტების რაოდენობაზე და დაემატოს მათი მენეჯმენტისათვის საჭირო ინფორმაციული ნაკადი. ამგვარად, მონაცემთა საცავისათვის საჭირო ფიზიკური მეხსიერება მიაღწევს რამდენიმე ტერაბაიტ-მოცულობას.

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

1.3 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 MS Windows Xp/2000, SQL Server-2000 Borland C++Builder, Interbase და სხვა ქსელური ბაზების სისტემებს, გააჩნია სპეციალური პაროლური სისტემები მომხმარებლების დასარეგის-ტრირებლად (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.



ნახ.1.3

რა თქმა უნდა, ასეთი კატეგორიის მომხმარებლები სხვადასხვა პრიორიტეტებით ისარგებლებენ.

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

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 შ შერვერ-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 ჰქონდეს სპეციალური არქივატორები (Backup-შეკუმშვა, Restore-გახსნა).

1.3. 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 [3].

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.

1.3.1. 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

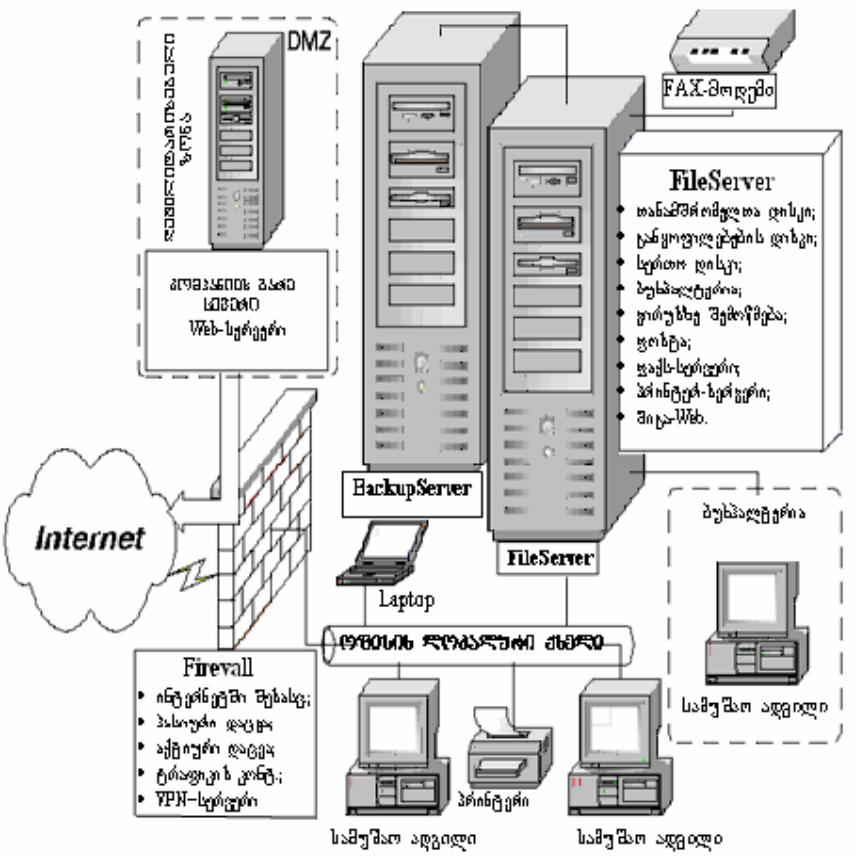
აქვს თუ არა უფლება დაუკავშირდეს ამ კონკრეტულ საიტს [4].

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.

1.3.2. DMZ –დემილიტარიზებული ზონა

შიგა და გარე ქსელის გარდა არსებობს ქსელი რომელსაც DMZ დემილიტარიზებული ზონა ეწოდება [3,4]. DMZ-ს ვიყენებთ იმ შემთხვევაში, როდესაც სავაჭრო ცენტრს ინტერნეტში აქვს თავის გარე ქსელი (ვებ-სერვერი) და იგი საჭიროებს დაცვას (ნახ.1.4). ამ შემთხვევაში ეს სერვერი თავსდება DMZ-დემილიტარიზებულ ზონაში, სადაც იგი გაცილებით დაცულია გარე „შემოტევებისგან“. ამასთან კომპანიის შიგა ქსელი მისთვის რჩება „უცხოდ“.



ნახ.1.4

1.3.3. 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) [3]. 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.

1.3.4. 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. ამასთან ერთად ეს კავშირი უნდა იყოს მაქსიმალურად მოხერხებული, უსაფრთხო და იაფი.

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

1. გადაწყვეტილება უნდა იყოს უნივერსალური და შეესაბამისობაში მოდიოდეს დანართებთან. ე.ი 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 [4].

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.

1.3.5. მობილური კლიენტი

ხშირად არის შემთხვევა, როდესაც რომელიმე ფირმის რამდენიმე თანამშრომელი მიემგზავრება მივლინებაში და მათთვის აუცილებლობას წარმოადგენს შევიდნენ 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 [2].

1.3.6. 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 [3]:

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.

1.4. 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 [5].

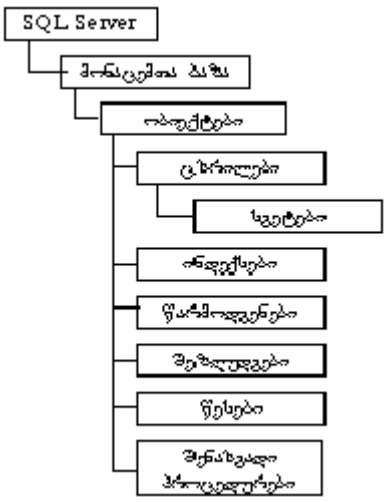
SQL Server მონაცემებთან მიმართვისათვის იყენებს ოთხ ძირითად ინტერფეისს: OLE DB, ODBC, DB Library და Transact-SQL. მომხმარებლისათვის, რომელიც მუშაობს Windows-სისტემასთან, ეს ინტერფეისები რეალიზებულია დინამიკურად მიერთებადი ბიბლიოთეკის, DLL-ფაილების სახით. Web-კლიენტებისთვის ქსელური ბიბლიოთეკის გამოძახება 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 საადრიცხვო 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 ქვესიმრავლედ: DDL - მონაცემთა ბაზების ცხრილებისა და წარმოდგენების შესაქმნელად, DML-მოთხოვნების შესაქმნელად და მონაცემთა დასამუშავებლად, DCL (Data Control Language) - მონაცემთა ბაზასთან მიმართვის პროცედურების სამართავად. 1.5 ნახაზზე მოცემულია SQL Server-ის არქიტექტურა [5].



ნახ.1.5.

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).

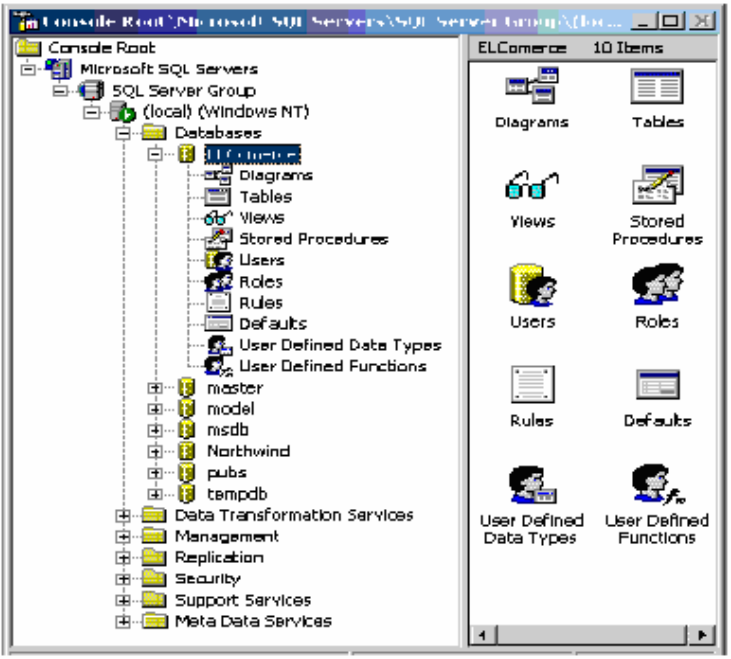
msdb.mdf და msdblog.ldf - ეს ბაზა შეიცავს ინფორმაციას დავალებების (jobs), მოვლენებისა (alerts) და ოპერატორების (operators) შესრულებათა მიმდევრობების დასაგეგმად.

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.

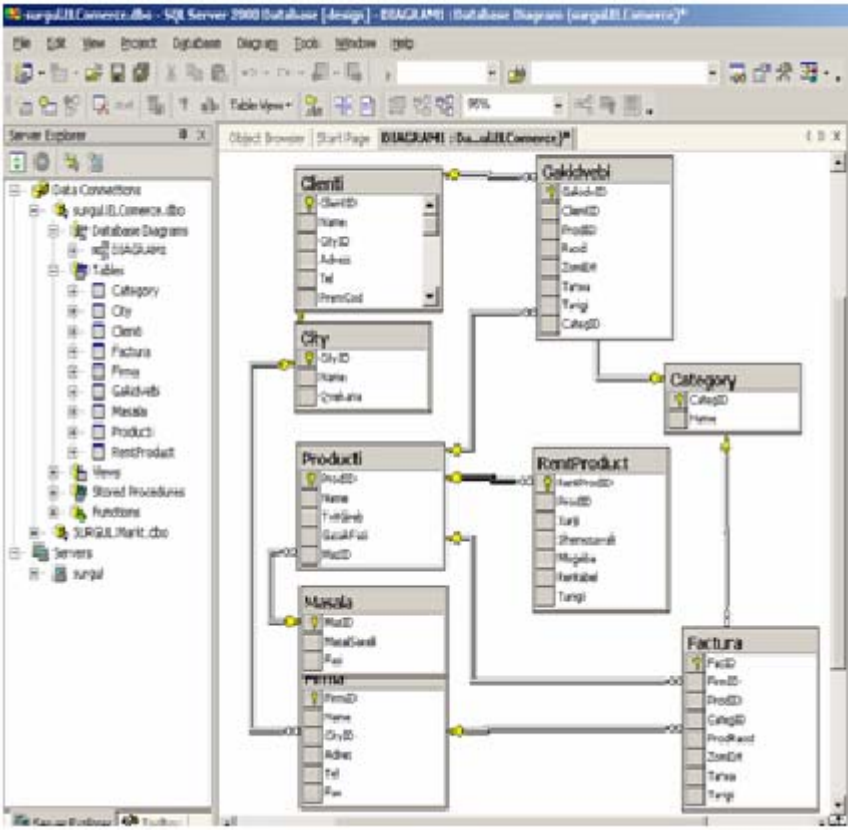
1.5. MsSQL Server - პაკეტის გამოყენების ვიზუალური ინსტრუმენტი

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

1.6 ნახაზზე მოცემულია ელექტრონული კომერციის სისტემის (ELCommerce) რეალიზებული მონაცემთა ბაზის ერთ-ერთი ფრაგმენტი MsSQL Server-2000-ის მაგალითზე.

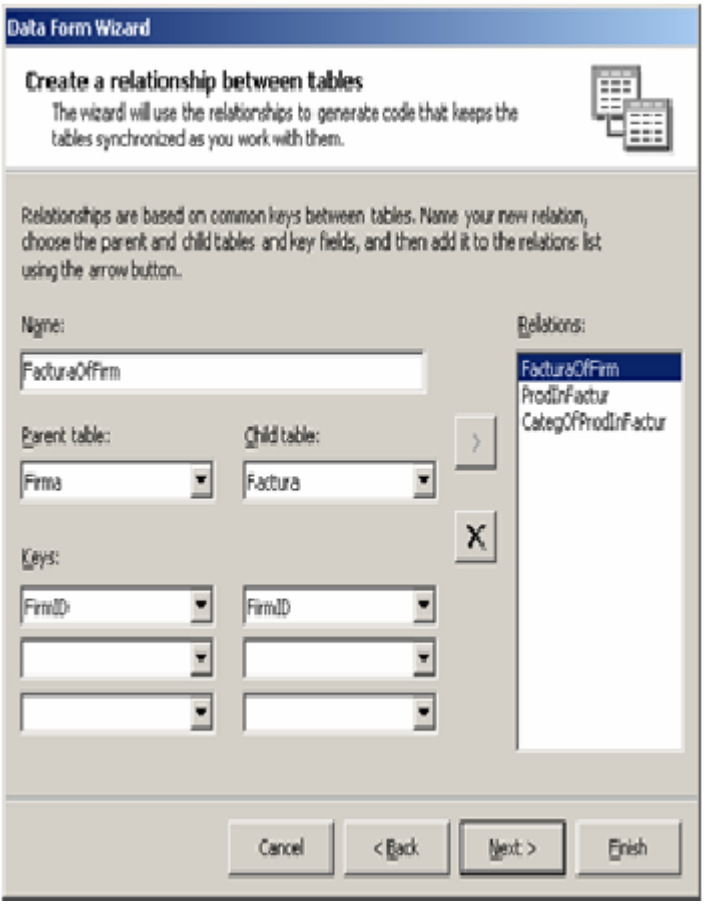

Console Root
Microsoft SQL Servers
SQL Server Group
(local) (Windows NT)
Databases
Diagrams
Tables
Views
Stored Procedures
Users
Roles
Rules
Defaults
User Defined Data Types
User Defined Functions
master
model
msdb
Northwind
pubs
tempdb
Data Transformation Services
Management
Replication
Security
Support Services
Meta Data Services
ELComerce
10 Items
Diagrams
Tables
Views
Stored Procedures
Users
Roles
Rules
Defaults
User Defined Data Types
User Defined Functions

ნახ.1.6-ა

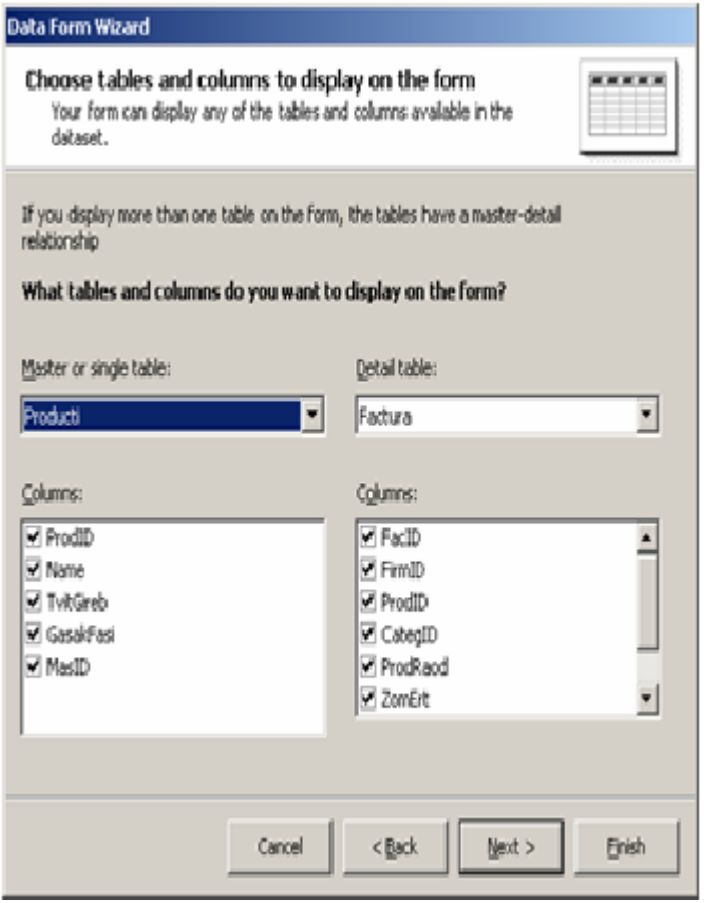

Clienti
Gakidvebi
City
Category
Producti
RentProduct
Masala
Firma
Faktura

ნახ.1.6-ბ

1.7 ნახაზზე ნაჩვენებია სისტემის კატალოგი, ცხრილთა კავშირების სტრუქტურა (ა) და ფორმაზე გამოსატანი ცხრილების და ველების (ბ) ამორჩევის ფრაგმენტი.



ნახ.1.7-ა


Data Form Wizard
Choose tables and columns to display on the form
Your form can display any of the tables and columns available in the dataset.
If you display more than one table on the form, the tables have a master-detail relationship
What tables and columns do you want to display on the form?
Master or single table:
Producti
Detail table:
Factura
Columns:
ProdID
Name
TvltGareb
GasakFasi
MasID
Columns:
FacID
FirmID
ProdID
CategID
ProdRaod
ZomErt
Cancel
< Back
Next >
Finish

ნახ.1.7-ბ

1.6. ინტერფეისის აგება ADO.NET პაკეტით და C# ენით

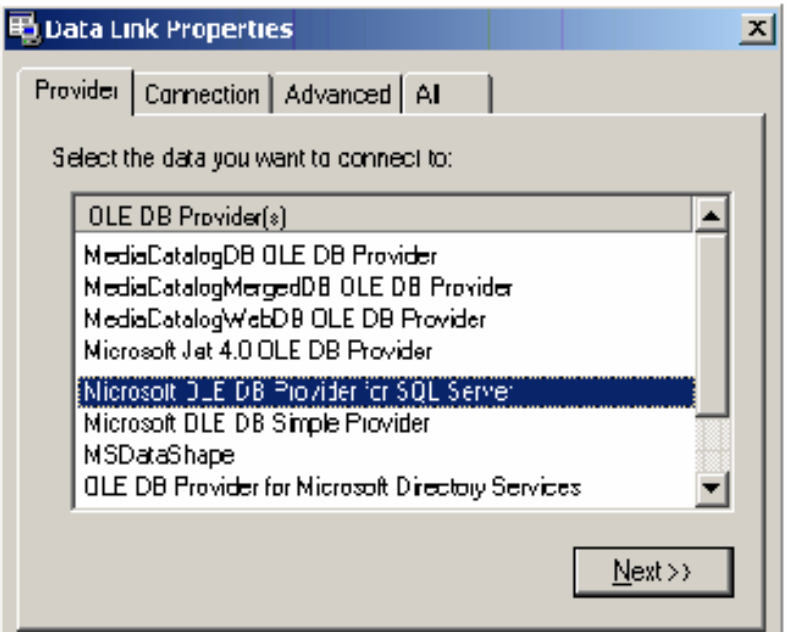
განვიხილოთ მომხმარებელთა ინტერფეისის აგებისა და მონაცემთა ბაზიდან ინფორმაციის გამოტანის საკითხები.

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

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

View | Server Explorer.

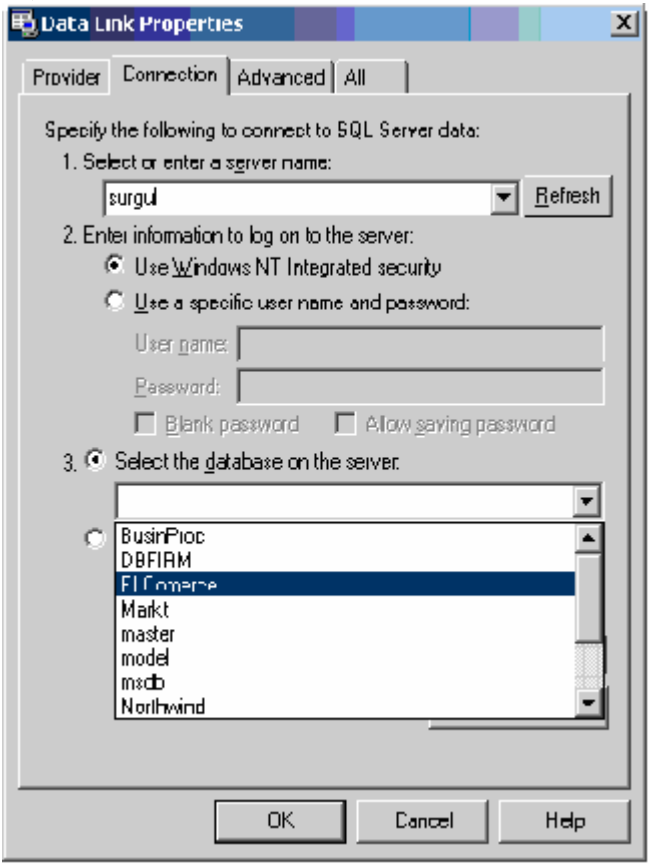
ეკრანის მარცხენა ზედა ნაწილში Data Connections-ზე მაუსის მარჯვენა ღილაკით ავირჩიოთ Add Connection. გამოჩნდება დამხმარე ფანჯარა Data Link Properties, რომელშიც უნდა შევირჩიოთ ჩვენთვის საჭირო მონაცემთა ბაზის დრაივერი (ნახ.1.8).



ნახ.1.8

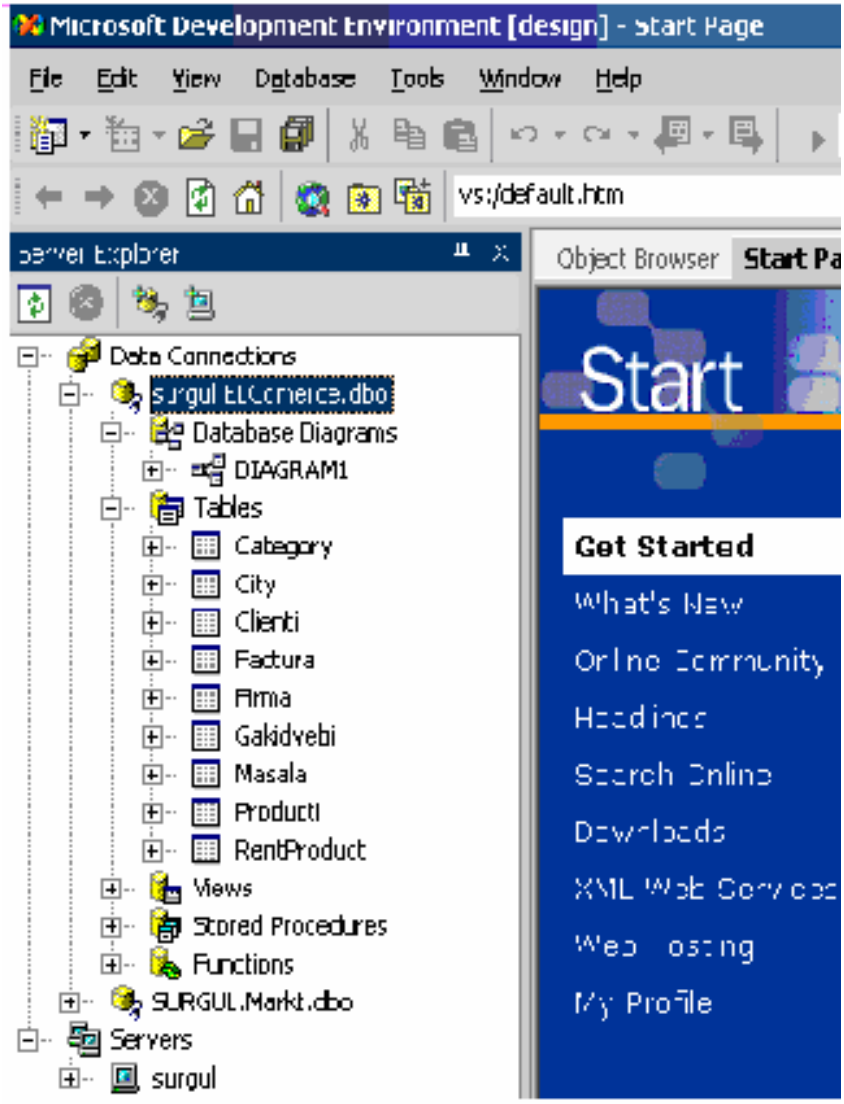
აქ ნაჩვენებია MsSQLServer-ის დრაივერის მაგალითი. თუ საჭიროა MsAccess, მაშინ ავირჩევთ Microsoft Jet 4.0 OLE DB-ს.

დრაივერის არჩევის შემდეგ Connection გვერდზე (ნახ.1.9) შევირჩევთ სერვერის სახელს (მაგ., surgul), გადავრთავთ UseWindows NT security და ავირჩევთ მონაცემთა ბაზას (მაგ., ElComerce).



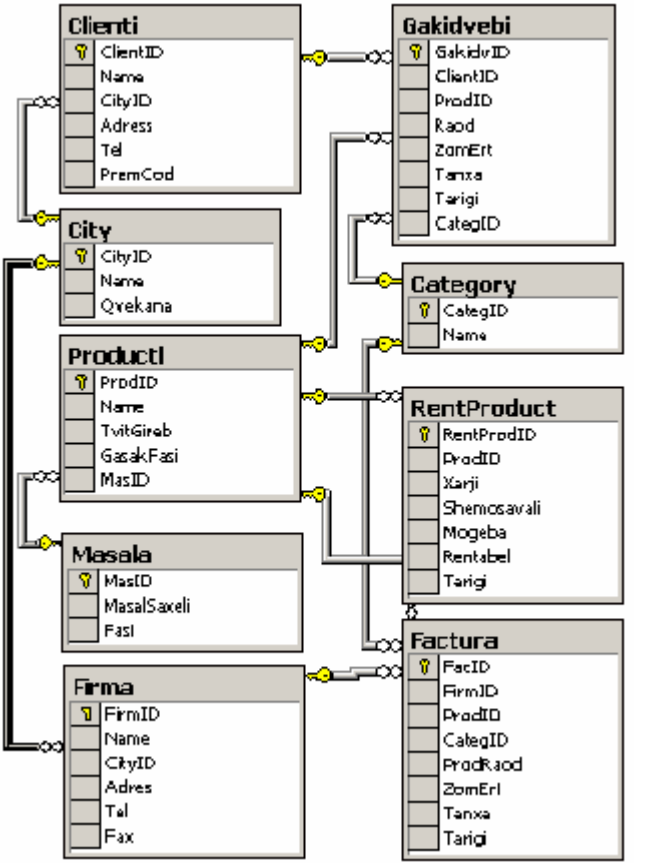
ნახ.1.9

მივიღებთ 1.10 ნახაზზე მოცემულ კადრს, რომელზედაც ჩანს MsSQLServer-ში აგებულ E_Comerce მონაცემთა ბაზის ცხრილები (Tables) და დიაგრამა (DIAGRAM1). ამგვარად, დაკავშირება ბაზასთან განხორციელდა.


Microsoft Development Environment [design] - Start Page
File Edit View Database Tools Window Help
vs:/default.htm
Server Explorer
Object Browser
Start Page
Data Connections
Database Diagrams
DIAGRAM1
Tables
Category
City
Clienti
Factura
Firma
Gakidvebi
Masala
Produkti
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

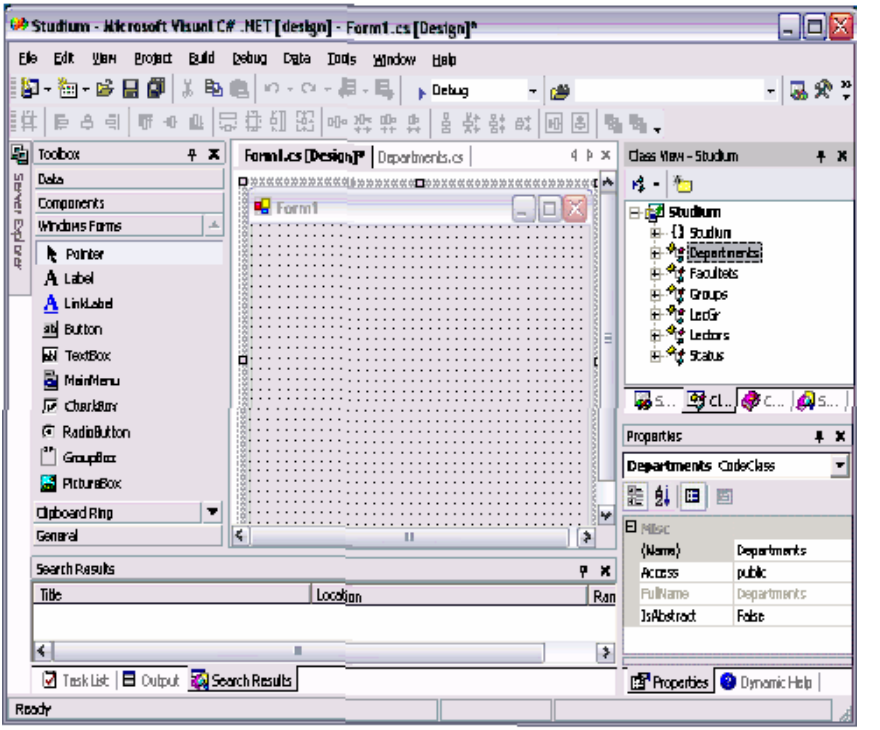
ნახ.1.10

თუ გავხსნით DIAGRAM1-ს, მაშინ გამოჩნდება ADO.NET-ში ასახული ElComerce-ბაზის ცხრილთაშორის კავშირების სქემა (ნახ.1.11).



ნახ.1.11

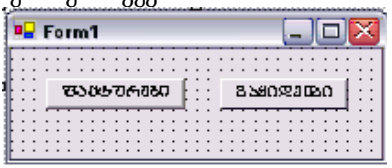
შემდეგ ეტაპზე .NET-ძირითად მენიუში (ნახ.1.12) ვირჩევთ: File | Add New Item | Data Form Wizard, რომლითაც შესაძლებელია ფორმის აგება და მონაცემთა გამოტანა ოსტატი-პროგრამით (Wizard).



ნახ.1.1.12

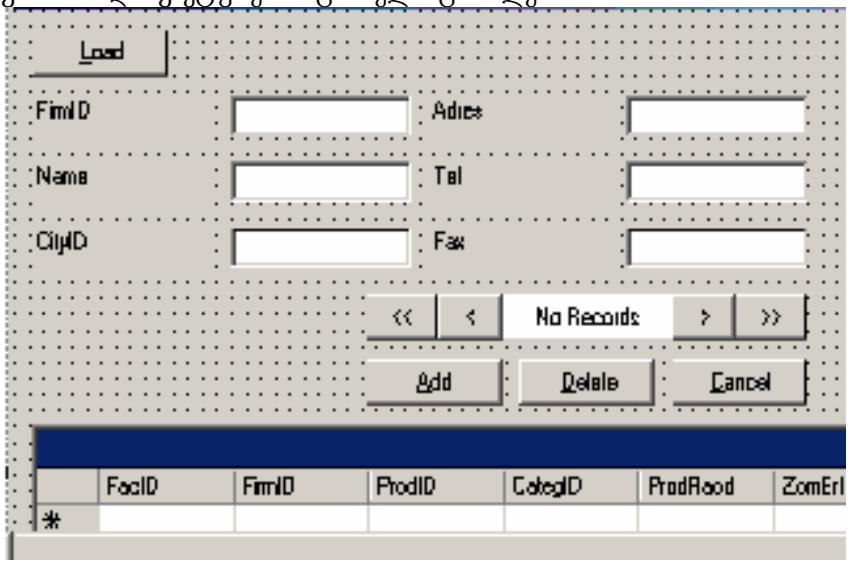
ეს პროგრამა თვითონ წარმართავს დიალოგს, რაც აადვილებს მუშაობას. აქ მთავარია ჩვენ სწორად შევირჩიოთ ცხრილები და ცხრილთაშორისი კავშირები.

ფორმაზე Form1 ინსტრუმენტების პანელიდან გადმოვიტანოთ ორი ღილაკი (button1, button2). მოვნიშნოთ პირველი ღილაკი და Properties-ში შევცვალოთ მისი თვისებები. მაგ., ქართული შრიფტი (Font ->LitMtavrPS-ით) და წარწერა „ფაქტურები“ (Text). 1.13 ნახაზზე ნაჩვენებია ეს შემთხვევა.



ნახ.1.1.13

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

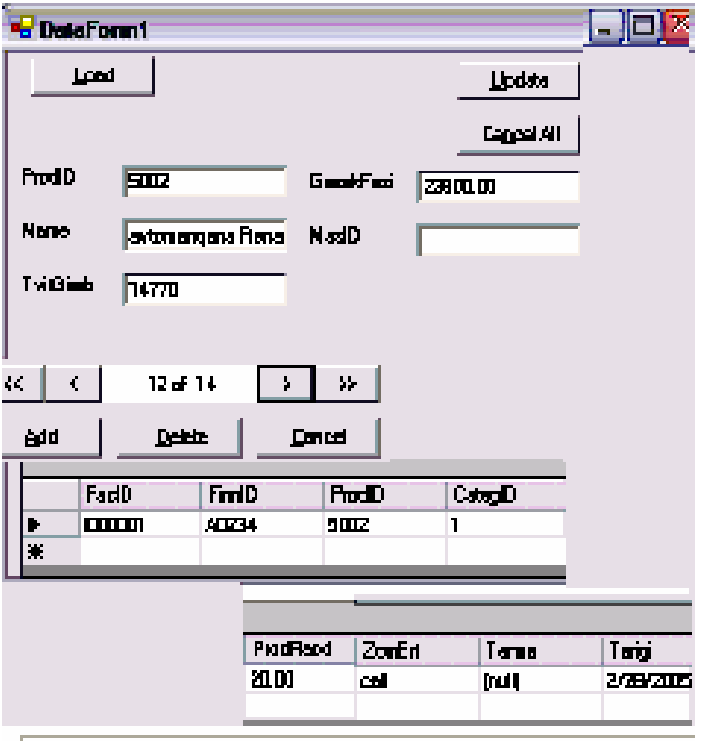


ნახ.1.14

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:

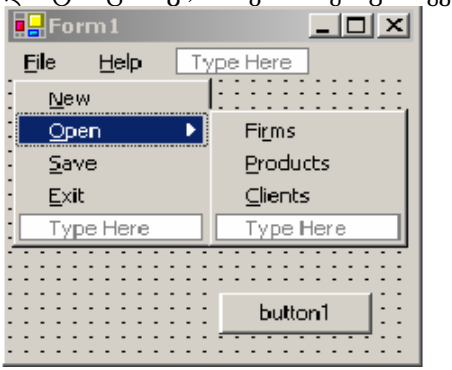
```
private void button1_Click(object sender, System.EventArgs e)
    { // შესატანი ტექსტის ორი სტრიქონი:
      DataForm1 ob=new DataForm1( );
       ob.Show( );
     }
```

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 ფორმაზე (ნახ.1.15).



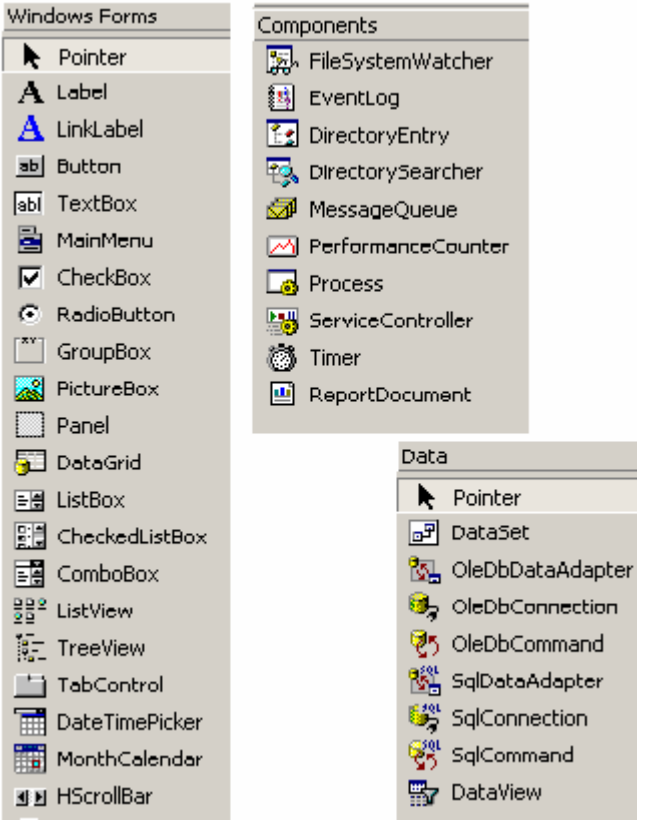
ნახ.1.15

ღილაკის მსგავსად ფორმაზე შესაძლებელია სხვა ვიზუალური კომპონენტების გადმოტანაც. მაგ., მთავარი მენიუს შექმნა (ნახ.1.16).



ნახ.1.16

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



ნახ.1.17

1.7. ASP.NET პროგრამული პაკეტით Web-გვერდის აგების საილუსტრაციო 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. ნემისმიერი 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.

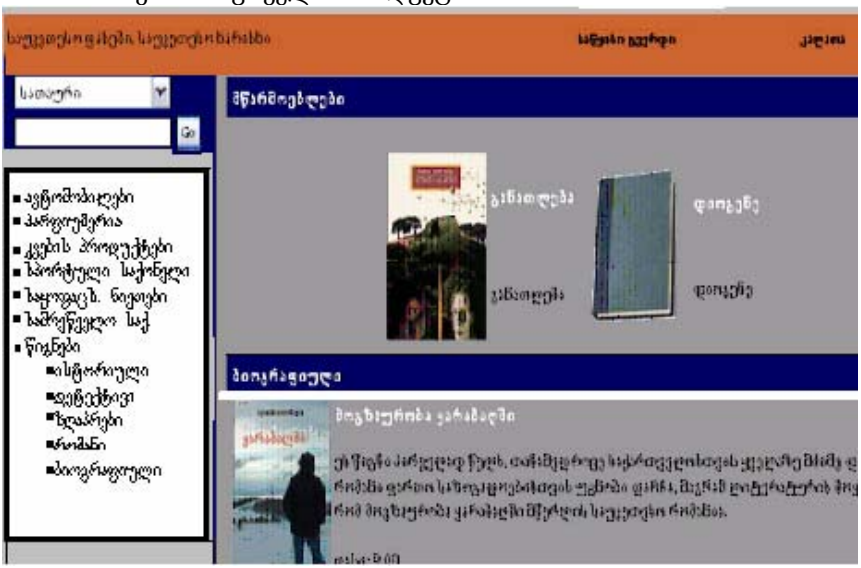
იძლევა საშუალებას იოლად დაიწეროს პროგრამის მუშაობის ლოგიკა აპლიკაციისათვის. ეს ლოგიკა შეიძლება შეიცავდეს სხვა აპლიკაციის დონის მოვლენების დამუშავებას.

.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.

1.18. ნახაზზე მოცემულია ASP.NET ტექნოლოგიის გამოყენებით ინტერნეტში განთავსებული სავაჭრო ცენტრის საინფორმაციო გვერდი, რომელიც მომხმარებელს საშუალებას აძლევს შეარჩიოს მისთვის სასურველი პროდუქცია.



ნახ.1.18-ა

ნახ.1.18-ბ

1.8. .NET პლატფორმის არსი და C# დაპროგრამების ენის 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.

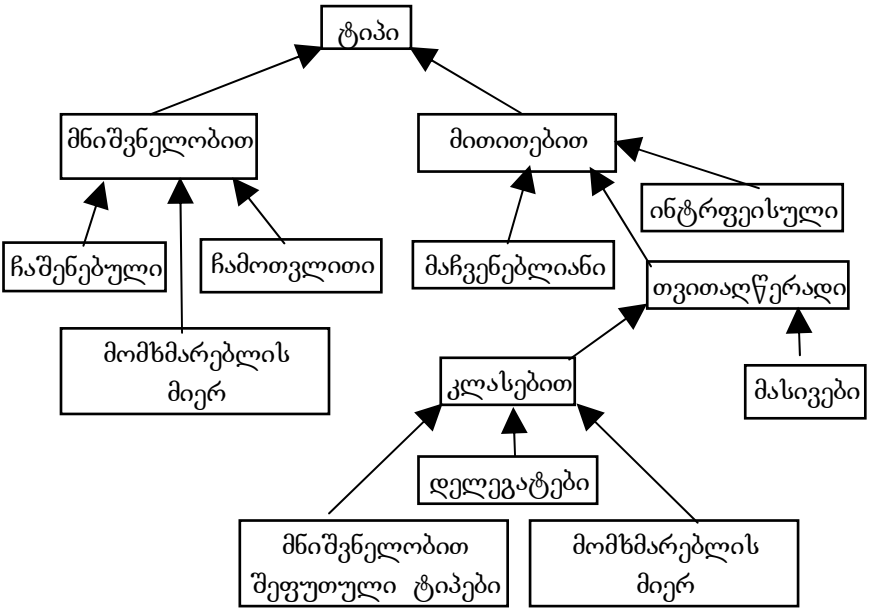
.NET პლატფორმა შეიქმნა სპეციალურად განაწილებული გამოყენებითი სისტემების ასაგებად დიდი მოცულობის ინფორმაციის დასამუშავებლად კლიენტ-სერვერ არქიტექტურის ბაზაზე.

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 Windows-დანართები.

C++ ენა კომპილირდებოდა ასემბლერულ კოდში. C# კი 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).

1.19 ნახაზზე მოცემულია აღნიშნული ტიპების იერარქიათა კლასიკური საილუსტრაციო მაგალითი მემკვიდრეობითობის კავშირის გამოყენებით.



ნახ.1.19

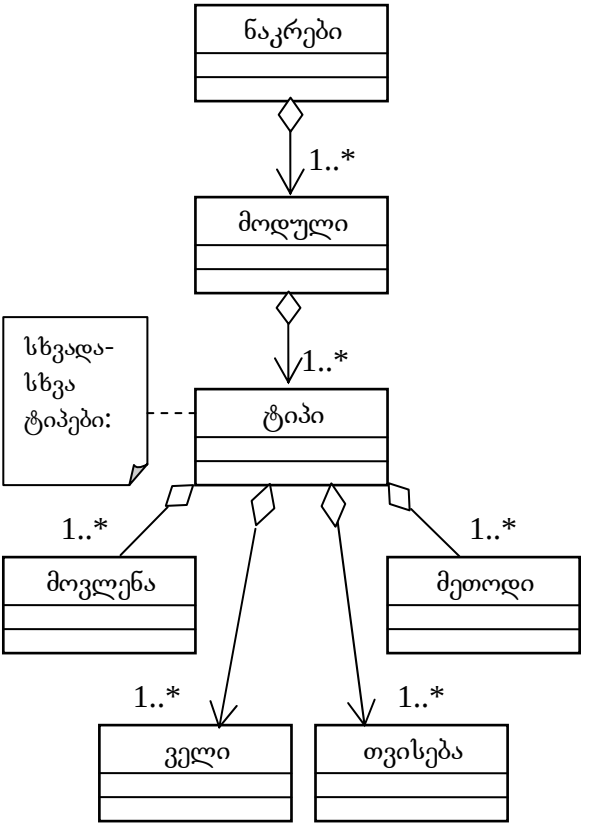
გამოყენებითი კოდების აგება გაადვილებულია .NET პლატფორმის საბაზო კლასების მდიდარი ბიბლიოთეკის არსებობით (იგი გამოიყენებოდა ადრე მაიკროსოფტის მიერ Windows API - Application Programming Interface ამოცანებისათვის).

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

.NET პლატფორმის კომპონენტებიდან ერთ-ერთი მთავარი ბლოკია Assembly (ანაწყობი, ნაკრები), რომელიც ლოგიკურად აერთიანებს კოდს, რესურსებს და მეტამონაცემებს. 1.20 ნახაზზე ნაჩვენებია მისი ზოგადი იერარქიული სტრუქტურა აგრეგაციის კავშირის გამოყენებით.

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 კოლიზიისა და ნაკრებთა ვერსიების კონტროლის თვალსაზრისით).



ნახ.1.20

კლასებს შორის სახელთა კოლიზიის აღმოფხვრის 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.

მაგალითად, ჩვენი C#-ის პროგრამის ტექსტში:

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

.NET პლატფორმის მნიშვნელოვანი ელემენტია „დანართთა არეები“. მათი დანიშულებაა 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 წერა, 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.

a) **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;
}
```

b) **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.

წინა პარაგრაფში ჩვენ განვიხილეთ ინტერნეტ-მაღაზიის რეალიზაციის მაგალითი ASP-სისტემაში, რომელიც კავშირში იყო MsSQL Server-თან. ქვემოთ მოცემულია ამ საილუსტრაციო მაგალითის C# პროგრამის ტექსტი:

```
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.MultiLine;
        AmountLabel.Text =
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;
        }


        errorsFound = !Page.IsValid;

        if (errorsFound)
        {
            return false;
        }
        else
        {
            String expDate = ExpMonthDropDownList.SelectedItem.Value + "/" +
ExpYearDropDownListBox.SelectedItem.Value;

ShoppingCart(false).SetPayment(CartTypeDropDownList.Items[CartTypeDropDownLi
st.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;
    }
  }
}
```

1.9. XML და HTML პროგრამათა ფრაგმენტები

eXtensible Markup Language (XML მონაცემთა ფორმატირების გაფართოებული ენა) თანამედროვე ინფორმაციული ტექნოლოგიების გამოყენებისას მეტად აქტუალურია [6,7].

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 [8]. 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.

განვიხილოთ XML კოდის ფრაგმენტის ტექსტი ინტერნეტ-მაღაზიის რეალიზაციის მაგალითზე ASP-სისტემაში:

```
<?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=b77a5c561934e089" />
  </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>
```

მომდევნო ტექსტი კი შეესაბამება HTML-კოდის აღწერას:

```
<!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">მისამართი</TD>
            <TD class="rheadercol" align="right" height="25">ნაბიჯი 1(3)
             </TD>
          </TR> <!-- SPACER ROW -->
          <TR class="rbody">
            <TD class="rbodycol" align="center" height="25"> 
             </TD>
          </TR>
             <TR class="rbody">
              <TD class="rbodycol" style="WIDTH: 389px" align="center"
height="25"><TABLE width="100%">
               <TR>
          <TD width="30%"></TD>
          </TR>
          <TR>
          <TD vAlign="top" width="30%">სახელი:</TD>
          <TD vAlign="top">
          <asp:TextBox id="ShipToNameTextBox" runat="server"
MaxLength="40"></asp:TextBox>(შეავსეთ)
          <asp:RequiredFieldValidator id="RequiredFieldValidator1"
runat="server" display="dynamic" errormessage="შეავსეთ სახელის ველი"
```

```
          controlToValidate="ShipToNameTextBox">*</asp:RequiredFieldValida
tor></TD>
          </TR>
         <TR>
           <TD vAlign="top" width="30%">მისამართი:</TD>
            <TD vAlign="top">
          <asp:TextBox id="AddressTextBox" runat="server"
MaxLength="255"></asp:TextBox>(შეავსეთ)
          <asp:RequiredFieldValidator id="RequiredFieldValidator2"
runat="server" display="dynamic" errormessage="შეავსეთ მისამართის უჯრა"
          controlToValidate="AddressTextBox">*</asp:RequiredFieldValidator>
           </TD>
        </TR>
          <TR>
          <TD vAlign="top" width="30%">რეგიონი:</TD>
          <TD>
          <asp:TextBox id="CountryTextBox" runat="server"
MaxLength="40"></asp:TextBox>(შეავსეთ)
          <asp:RequiredFieldValidator id="RequiredFieldValidator3"
runat="server" display="dynamic" errormessage="შეავსეთ რეგიონის უჯრა"
          controlToValidate="CountryTextBox">*</asp:RequiredFieldValidator>
         </TD>
       </TR>
          <TR>
          <TD vAlign="top" width="30%">ტელეფონი:</TD>
          <TD>
          <asp:TextBox id="PhoneNumberTextBox" runat="server"
MaxLength="14"></asp:TextBox>(შეავსეთ)
          <asp:RequiredFieldValidator id="RequiredFieldValidator4"
runat="server" display="dynamic" errormessage="შეავსეთ ტელეფონის უჯრა"
controlToValidate="PhoneNumberTextBox">*</asp:RequiredFieldValidator>
<asp:RegularExpressionValidator id="RegularExpressionValidator1"
runat="server" errormessage="ტელეფონის ნომერი უნდა იყოს შემდეგი
ფორმატის nn-nn-nn"
          ValidationExpression="\d{2}-\d{2}-\d{2}"
       ControlToValidate="PhoneNumberTextBox">
      *</asp:RegularExpressionValidator> </TD>
     </TR>
          <TR>
          <TD vAlign="top" width="30%">ფაქსი:</TD>
          <TD>
          <asp:TextBox id="FaxTextBox" runat="server"
MaxLength="14"></asp:TextBox>
          და ა.შ. . . .
```

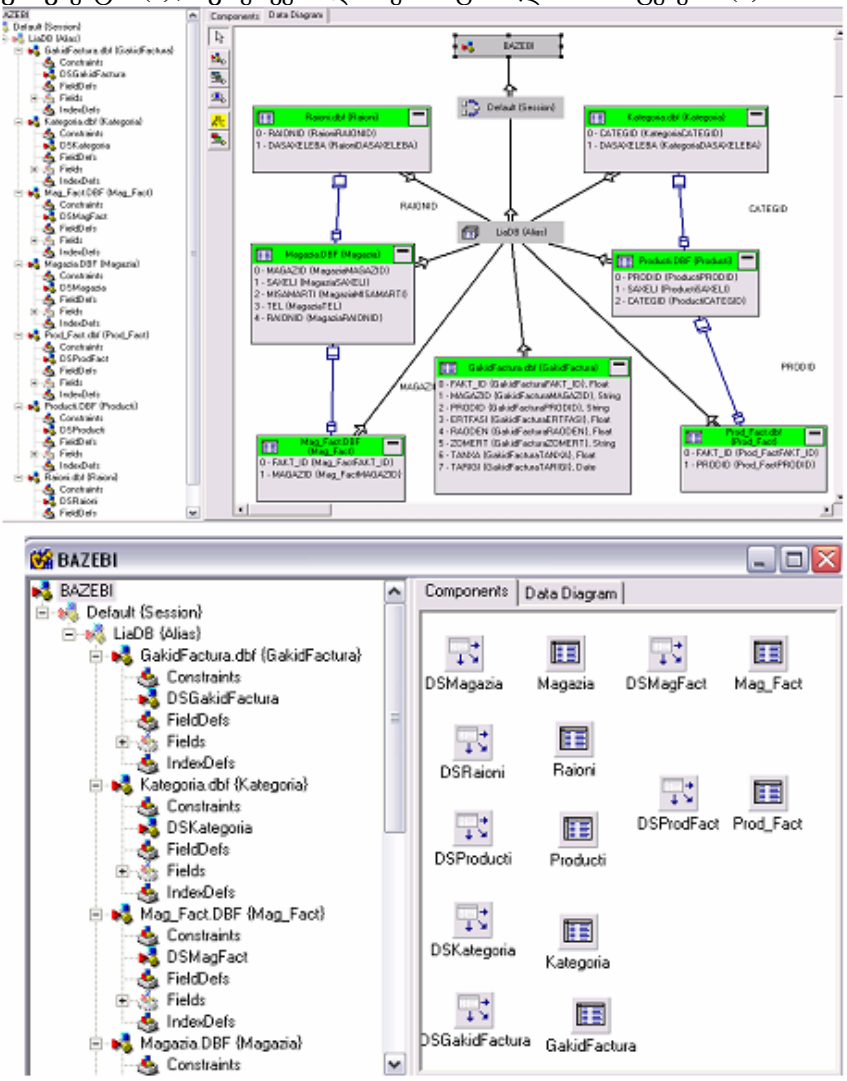
```
            <asp:Label id="TotalLabel" runat="server"
enableViewState="false"></asp:Label></TD>
            </TR>
            </TABLE>
            </TD>
            </TR>
</TABLE> <!--END ORDERSUMMARY MODULE--></asp:panel></td>
      </tr>
        <tr>
         <td style="PADDING-TOP: 5px" align="center"><asp:imagebutton
id="PreviousImageButton" runat="server" enableViewState="false"
imageurl="../images/previous.gif"></asp:imagebutton><asp:imagebutton
id="NextImageButton" runat="server" enableViewState="false"
imageurl="../images/next.gif"></asp:imagebutton></td>
      </tr>
<!-- END DYNAMIC RIGHT MODULE LIST --></table>
            </td>
          </tr>
        </table>
      </form>
    </body>
</HTML>
```

1.10. მონაცემთა მრავალგანზომილებიანი ანალიზის პაკეტის აგება Decision Cube კომპონენტებით

მონაცემთა ბაზებიდან მიღებული ინფორმაციის მრავალფაქტორული ანალიზისათვის C++Builder-ის სპეციალური Decision Cube კომპონენტებით ავაგეთ პროგრამული პაკეტი [13,14].

კუბის ყოველი განზომილება წარმოვადგინეთ მონაცემთა ბაზის ცხრილთა ველების სახით (მაღაზიები, პროდუქტები, კლიენტები, კატეგორიები და სხვ.). აქ 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.

2.21 ნახაზზე მოცემულია მონაცემთა ბაზის სტრუქტურა და ცხრილთა კავშირების ვიზუალური კომპონენტით აგების სქემის ფრაგმენტი (ა), აგრეთვე მაღაზიების ცხრილის მონაცემები (ბ).

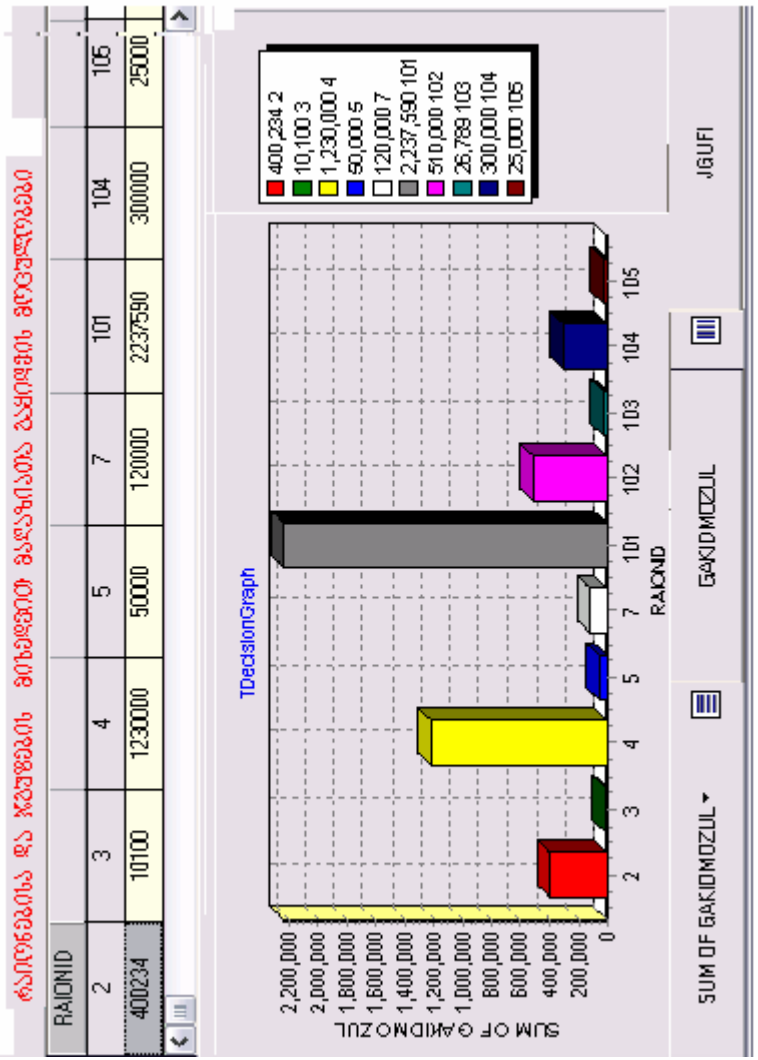


ნახ.2.21-ა

C++Builder 5 - Project1
File Edit Search View Project Run Component Database Tools Help
Standard Additional Win32 System Data Access Data Controls ADO
Object Inspector
Label1: TLabel
Properties Events
Align alNone
Alignment taLeftJustify
Anchors [akLeft,akTop]
AutoSize true
BiDiMode bdLeftToRight
Caption mnRazebis sia
Color clBtnFace
Constraints (TSizeConstrair
Cursor crDefault
DragCursor crDrag
DragKind dkDrag
DragMode dmManual
Enabled true
FocusControl
Font (TFont)
Height 19
Hint
Form3

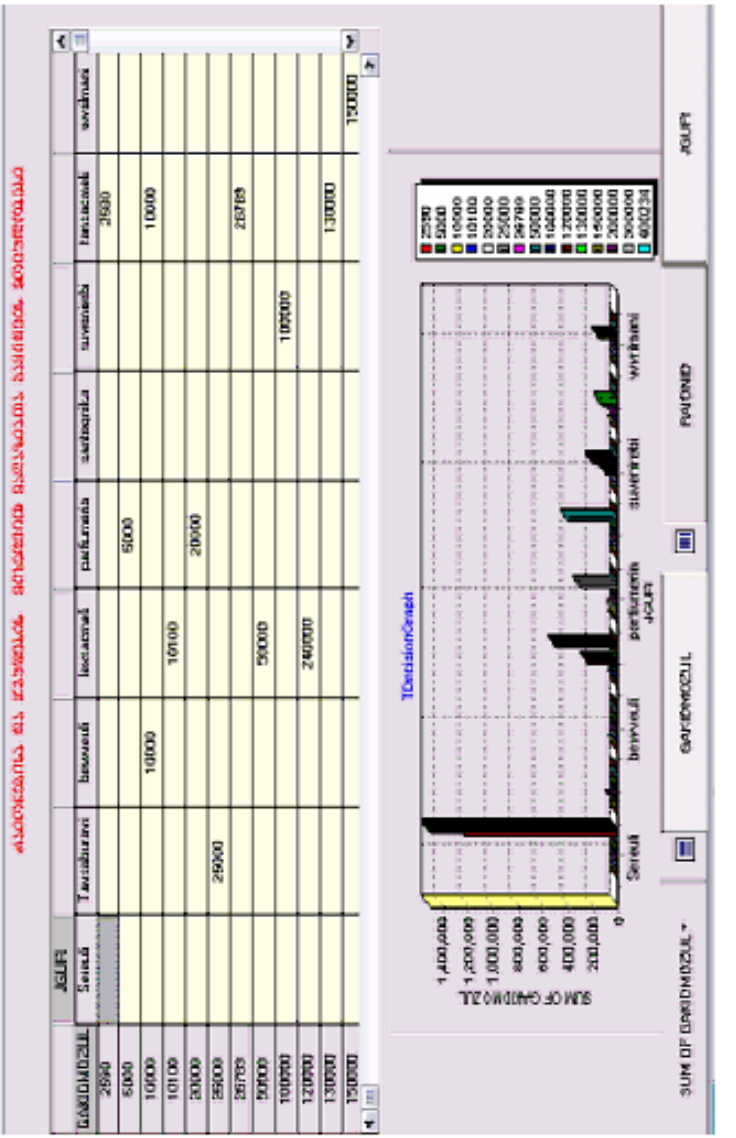
ნახ.2.21-ბ

2.22 ნახაზზე წარმოდგენილია სავაჭრო ცენტრის მენეჯერის ინტერფეისი სამგანზომილებიანი სტრუქტურით.



ნახ.2.22

2.23 ნახაზზე მოცემულია სავაჭრო ცენტრის მონაცემთა მრავალფაქტორული ანალიზის ინტერფეისის მაგალითი.



ნახ.2.23

გამოყენებული ლიტერატურა:

1. Albrecht, J., Hummer,W., Lehnen W., Using Semantics for Query Derivbility in Data Warehouse Applications. (FQAS, 2000 Warschau).
2. სურგულაძე გ., დოლიძე თ., ყვავაძე ლ. კომპონენტურ-ვიზუალური დაპროგრამება სტუ, თბილისი. 2006.
3. http://www.lanoffice Firewall.htm
4. A:\LANOffice Technical Firewall.htm
5. სურგულაძე გ., შონია ო., ყვავაძე ლ. მონაცემთ განაწილებული ბაზების მართვის სიტემები. სტუ, თბილისი. 2004.
6. David Hay, XML: What is It, Anyway? **www.essentialstrategies.com**
7. Язык XML — практическое введение (www.citforum.ru).
8. ბოტჰე კ., სურგულაძე გ., დოლიძე თ., შონია ო., სურგულაძე გ. თანამედროვე პროგრამული პლატფორმები და ენები // თბილისი, 2003.