გია სურგულაძე, ნინო თოფურია,
ანა გავარდაშვილი

GTU

შავი ზღვის ეკოლოგიური მონიტორინგის და კვლევის საინფორმაციო სისტემა

სტუ-ს „IT კონსალტინგის ცენტრი“

საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, ნინო თოფურია,
ანა გავარდაშვილი

შავი ზღვის ეკოლოგიური მონიტორინგის და კვლევის საინფორმაციო სისტემა

დამტკიცებულია:
სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრის“ სარედაქციო კოლეგიის მიერ

თბილისი
2018

უაკ 004.5

განხილულია შავი ზღვის ეკოლოგიური მონიტორინგის საინფორმაციო სისტემის აგების საკითხები ახალი ინფორმაციული ტექნოლოგიების ბაზაზე. კერძოდ, საქართველოს შავი ზღვის აკვატორიაში ძირითადი ეკოლოგიური მაჩვენებლების განსაზღვრის, შეგროვებისა და სისტემის პორტალის მონაცემთა ბაზების სერვერზე მათი გადაგზავნის საკითხები მობილური და ჰიბრიდული ტექნოლოგიებით. აგრეთვე მოცემულია ანალიზური კვლევის და სტატისტიკური დამუშავების პროცედურების ჩატარების, ეკოლოგიური მონიტორინგის ფუნქციური ამოცანების შესრულებისა და გადაწყვეტილების მიღების შესაბამისი რეკომენდაციების გამომუშავების ბიზნეს-პროცესების ობიექტორიენტირებული და უნიფიცირებული მოდელირების ამოცანები ეკოსისტემის მხარდამჭერი პროგრამული უზრუნველყოფის შესაქმნელად. აგებულია ეკომონიტორინგის სისტემის პორტალის ექსპერიმენტული დემოვერსია ობიექტ-როლური მოდელირებით და მულტიმედიური მონაცემთა ბაზებით. *მონოგრაფია* გამიზნულია ინფორმატიკის, მართვის საინფორმაციო სისტემებისა და ეკოლოგიის სპეციალობის სტუდენტებზე, დოქტორანტებსა და ამ საკითხებით დაინტერესებულ მკითხველზე.

რეცენზენტები:

- პროფ. ეკატერინე თურქია (სტუ)
- პროფ. იბრაიმ დიდმანიძე (ბათუმის სახ. უნივერსიტეტი)

პროფ. გ. სურგულაძის რედაქციით

რედკოლეგია:

ა. ფრანგიშვილი (თავმჯდომარე), მ. ახობაძე, გ. გოგიჩაიშვილი, ზ. ბოსიკაშვილი, ე. თურქია, რ. კაკუბავა, ნ. ლომინაძე, ჰ. მელაძე, თ. ობგაძე, რ. სამხარაძე, გ. ჩაჩანიძე, ა. ცინცაძე, ზ. წვერაიძე, გ. სურგულაძე (რეადაქტორი),

ISBN 978-9941-8-0624-7

Georgian Technical University

Gia Surguladze, Nino Topuria,
Ana Gavardashvili

BLACK SEA ECOLOGICAL MONITORING AND RESEARCH INFORMATION SYSTEM

The present book issues discusses design and construction of Information System for monitoring ecology of Black Sea on the basis of new information technologies. In particular, issues related to determining the main ecological indicators in the Black Sea waters of Georgia, gathering and sending them to the server database of system portal with mobile and hybrid technologies. The following tasks are considered for creating ecosystem supporting software: analytical research and implementation of statistical processing procedures; performing relevant recommendations for fulfillment of functional tasks of ecological monitoring and decision-making; object-oriented and unified modeling of business processes. The experimental version of the software for ecological monitoring system of portal is realized with object-role modeling and multimedia databases. Monograph is intended for students of informatics, management information systems and ecology and readers interested in these issues.

ISBN 978-9941-8-0624-7

ავტორების შესახებ:

გია სურგულაძე - ტექნიკის მეცნიერებათა დოქტორი, ინფორმატიზაციის საერთაშორისო აკადემიის ნამდვილი წევრი (1994 წლიდან. გაეროსთან არსებული IIA), საქ. ტექნიკური უნივერსიტეტის პროფესორი. ინფორმატიკის ფაკულტეტის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერიის)“ აკადემიური დეპარტამენტის უფროსი. 76 წიგნის (მათ შორის 16 მონოგრაფიის), 60 ელ-სახელმძღვანელოს და 250-ზე მეტი სამეცნიერო ნაშრომის ავტორი მართვის საინფორმაციო სისტემების პროგრამული ინჟინერიის, მონაცემთა რელაციური და NoSQL ბაზების, დაპროგრამების ჰიბრიდული ტექნოლოგიების, იმიტაციური მოდელირების (პეტრის ფერადი ქსელებით) და სხვ. სფეროში. *წვლილი:* წინამდებარე ნაშრომში შეიმუშავა შავი ზღის ეკომონიტორინგის საინფორმაციო სისტემის ბიზნეს-პროცესების უნიფიცირებული მოდელები, მონაცემთა მულტიმედიური ბაზების დაპროექტებისა და გამოყენების კონცეფცია, სისტემის ინფრასტრუქტურა და მომხმარებელთა ინტერფეისული ფუნქციების პროგრამები ინფორმაციული ბაზის მენეჯმენტის მიზნით.

ნინო თოფურია - ტექნიკის მეცნიერებათა კანდიდატი, სტუ-ს ინფორმატიკისა და მართვის სისტემების ფაკ-ის ასოცირებული პროფესორი. სპეციალიზებულია ელექტრონული დოკუმენტბრუნვის (საქმისწარმოების) სისტემის აგებისა და ექსპლუატაციის საკითხებში, ინფორმაციის დისტანციური გაცვლის და მისი უსაფრთხოების უზრუნველყოფის ამოცანებში. არის 12 წიგნისა და 100-მდე სამეცნიერო ნაშრომების ავტორი. კითხულობს ლექციებს საპატრიარქოს ქართულ უნივერსიტეტსა და ქართულ-ამერიკულ უნივერსიტეტში (GAU). *წვლილი:* წინამდებარე ნაშრომში შეიმუშავა შავი ზღვის ეკომონიტორინგის სისტემის მონაცემთა ბაზის კონცეპტუალური, ობიექტ-როლური მოდელი და მისი შემდგომი პროგრამული რეალიზაციის პროცედურები სისტემის ვებ-პორტალის ასაგებად SharePoint-ის საფუძველზე.

ანა გავარდაშვილი - სტუ-ის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის დოქტორანტი (2015-2017), აკადემიური დოქტორი (ხელმძღ., პროფ. გ. სურგულაძე). 1- წელი სწავლობდა გრაცის ტექნიკურ უნივერსიტეტში (ავსტრია) „საინფორმაციო სისტემების“ განხრით. ჰქონდა რუსთაველის ფონდის გრანტი (#DO/159/4-130/14) ,,შავი ზღვის ეკოლოგიური პარამეტრების კვლევა მულტიმედიური ბაზების საფუძველზე“. *წვლილი:* აწარმოებდა შავი ზღვის საკონტროლო პუნქტებში საველე სამუშაოებს (როგორც ექსპერტი), ზღვის წყლის სინჯების ანალიზის შედეგების ინფორმაციას გადმოსცემდა მონაცემთა ბაზის სერვერს. კვლევის შედეგები მოხსენიებულ იქნა პირადად მის მიერ პარიზის და ამსტერდამის საერთაშორისო კონფერენციებზე (WASET, 2016-2017 წწ). არის 3 წიგნისა და 20-მდე სამეცნიერო ნაშრომის ავტორი შავი ზღვის ეკოლოგიის საინფორმაციო სისტემების პრობლემების სფეროში.

სარჩევი

შესავალი

საქართველოს გეოგრაფიული მდებარეობის მნიშვნელოვანი ფაქტორია დასავლეთი-დან შავი ზღვის სანაპიროს 310 კმ-იანი ზოლის არსებობა, რომელიც დიდ გავლენას ახდენს ქვეყნის ჰავის ფორმირებაზე, განსაზღვრავს მის პოლიტიკურ, სოციალურ და ეკონომიკურ პოტენციალს, განსაკუთრებით საგარეო კავშირურთიერთობების, საზღვაო მრეწველობისა და ტურიზმის განვითარების თვალსაზრისით. ამჟამად, შავი ზღვის აკვატორიის 200 კმ (აფხაზეთი) ანექსირებულია რუსეთის მიერ და, შესაბამისად, ჩვენი ქვეყნის იურისდიქცია ვრცელდება მისი აკვატორიის მხოლოდ 110 კმ სიგრძეზე.

კლიმატის გლობალურმა ცვლილებამ ბოლო წლებში გამოიწვია შავი ზღვის აკვატორიის მდინარეთა კალაპოტებში წყალდიდობების წარმოქმნის სიხშირის მატება. მდინარეთა ესტუარებში და მის მიმდებარე ტერიტორიებზე ხშირია ზღვის სანაპირო ზოლის აბრაზია, რაც მთავრდება ძალზე უარყოფითი ეკოლოგიური შედეგებით, კერძოდ, მიმდინარეობს ზღვის მიერ საქართველოს საზღვრებში სანაპირო ზოლის - ხმელეთის მიტაცება და უფრო შიგნით ხმელეთის სიღრმეში ზღვის შემოსვლა [1-5].

განსაკუთრებით უნდა აღინიშნოს შავი ზღვის სანაპიროზე განთავსებული ქვეყნის სტრატეგიული მიმართულების: ბათუმის, ფოთის, სოფელ ყულევის და მშენებარე ანაკლიის საზღვაო პორტების ეკოლოგიური უსაფრთხოების უზრუნველყოფა, რაც პირდაპირ უკავშირდება შავი ზღვის ეკოლოგიური პრობლემების მეცნიერულ შესწავლასა და მის სრულყოფას, პროგნოზირებას თანამედროვე კომპიუტერული პროგრამებისა და ტექნოლოგიების გამოყენებით [6-10].

შავი ზღვის ეკოლოგიური გარემო ძირითადად დამოკიდებულია იმაზე, თუ როგორია წყალში ბაქტერიების, ჟანგბადის რაოდენობა (%), მარილიანობა (TDS), ჰაერისა (t_1) და ზღვის ტემპერატურა t_2 (C^0), განსაკუთრებული ყურადღება უნდა მიექცეს ზღვის ვერტიკალურ ფენაში (სიღრმე 100 მ) ჟანგბადის რაოდენობის შესწავლას, სადაც ფოტოსინთეზი აქტიურია და ხელს უწყობს ზღვის ფლორისა და ფაუნის სიცოცხლის უნარიანობას. ყოველივე ზემოაღნიშნულის გათვალისწინებით, შავი ზღვის ეკოლოგიური საკითხების მეცნიერული კვლევა და მისი პროგნოზირება თანამედროვე ინფორმაციული ტექნოლოგიების გამოყენებით საქართველოსათვის მეტად აქტუალურია, იგი არის ქვეყნის სტრატეგიული მიმართულება, ხოლო შავი ზღვის სანაპირო ზოლისა და მისი მიმდებარე ტერიტორიების შენარჩუნება და დაცვა – ქვეყნის მთავრობას აღიარებული აქვს სახელმწიფოს პრიორიტეტულ მიმართულებად [1,11].

წინამდებარე ნაშრომში ასახულია შავი ზღვის სანაპიროს საკონტროლო პუნქტებში (საქართველოს ფარგლები) ძირითადი ეკოლოგიური მაჩვენებლების განსაზღვრის, შეგროვებისა და ონლაინ რეჟიმში მათი მონაცემთა ბაზების სერვერზე გადაგზავნის, ანალიზური კვლევის და სტატისტიკური დამუშავების პროცედურების ჩატარების, ეკოლოგიური მონიტორინგის ფუნქციური ამოცანების შესრულებისა და გადაწყვეტილების მიღების შესაბამისი რეკომენდაციების გამომუშავების საკითხები. კერძოდ:

პირველ თავში განხილულია შავი ზღვის ეკოლოგიური მონიტორინგის საინფორმაციო სისტემის პროგრამული უზრუნველყოფის აგების მიზნით საპრობლემო სფეროს (ეკო-მონიტორინგის სისტემა) ბიზნესპროცესების დიაგნოსტიკური ანალიზი და უნიფიცირებული მოდელირების ენის (UML) საფუძველზე ობიექტ-ორიენტირებული მოდელების (დიაგრამების) დაპროექტება, რის საფუძველზეც პროგრამისტ-დეველოპერები შეძლებენ კომპიუტერული სისტემის ფუნქციონალური ამოცანების დაპროგრამებას და ტესტირებას [10].

მეორე თავი ეხება შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის მონაცემთა ბაზების დაპროექტებისა და პროგრამული რეალიზაციის ამოცანების გადაწყვეტას. განხილულია ობიექტ-ორიენტირებული, რელაციური, NewSQL და NoSQL ბაზები, აგრეთვე მულტიმედიური მონაცემთა ბაზები და მათი გამოყენების კონცეფცია ეკოლოგიური მონიტორინგის სისტემებში [8].

მესამე თავში ასახულია ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის ვებ-პორტალის შექმნის კონცეფცია. განხილულია „მაიკროსოფთის“ კორპორაციის ახალი ტექნოლოგიების გამოყენება ამ სფეროში SharePoint სისტემის საფუძველზე [ნინო]. აქვე გადმოცემულია ჰიბრიდული და მობილური ტექნოლოგიების კომპლექსური გამოყენების საკითხები ეკო-მონიტორინგის სისტემაში ინფორმაციის გაცვლის მიზნით საკონტროლო პუნქტებსა და კვლევის ცენტრალურ სერვერს შორის [12].

მეოთხე თავში წარმოდგენილია ეკოლოგიური მონიტორინგის სისტემის სერვერზე განთავსებული ინფორმაციის დამუშავების ფუნქციონალური ამოცანების ალგორითმებისა და პროგრამების რეალიზაციის საკითხები, მათ საფუძველზე შესაბამისი ანალიზური და პროგნოზული შედეგების ფორმირება, რაც აუცილებელი ინფორმაციული მასალა იქნება მონიტორინგის სისტემის ხელმძღვანელობისთვის გადაწყვეტილების მიღების პროცესში [1].

წიგნის *დასკვნით* ნაწილში გადმოცემულია ავტორთა მიერ მიღებული თეორიული და პრაქტიკული შედეგების ანალიზი და მათი გამოყენების რეკომენდაციები. ყოველივე ზემოაღნიშნული საშუალებას მოგვცემს ახალი ინფორმაციული სისტემებისა და ტექნოლოგიების საფუძველზე კომპლექსურად შევაფასოთ შავი ზღვის თანამედროვე ეკოლოგიური პრობლემები და ეკოლოგიური მონიტორინგის სპეციალისტებმა ოპერატიულად და ეფექტურად დაგეგმონ ზღვის სანაპირო ზოლისა და მიმდებარე ტერიტორიების ეკოლოგიური უსაფრთხოების განსახორციელებელი ღონისძიებები.

ავტორები წინასწარ გამოხატავენ მადლიერებას მკითხველთა და სპეციალისტთა მიმართ მათი საქმიანი რჩევებისა და შენიშვნებისათვის.

I თავი

საპრობლემო სფეროს სისტემური ანალიზი უნიფიცირებული მოდელირების ენის საფუძველზე

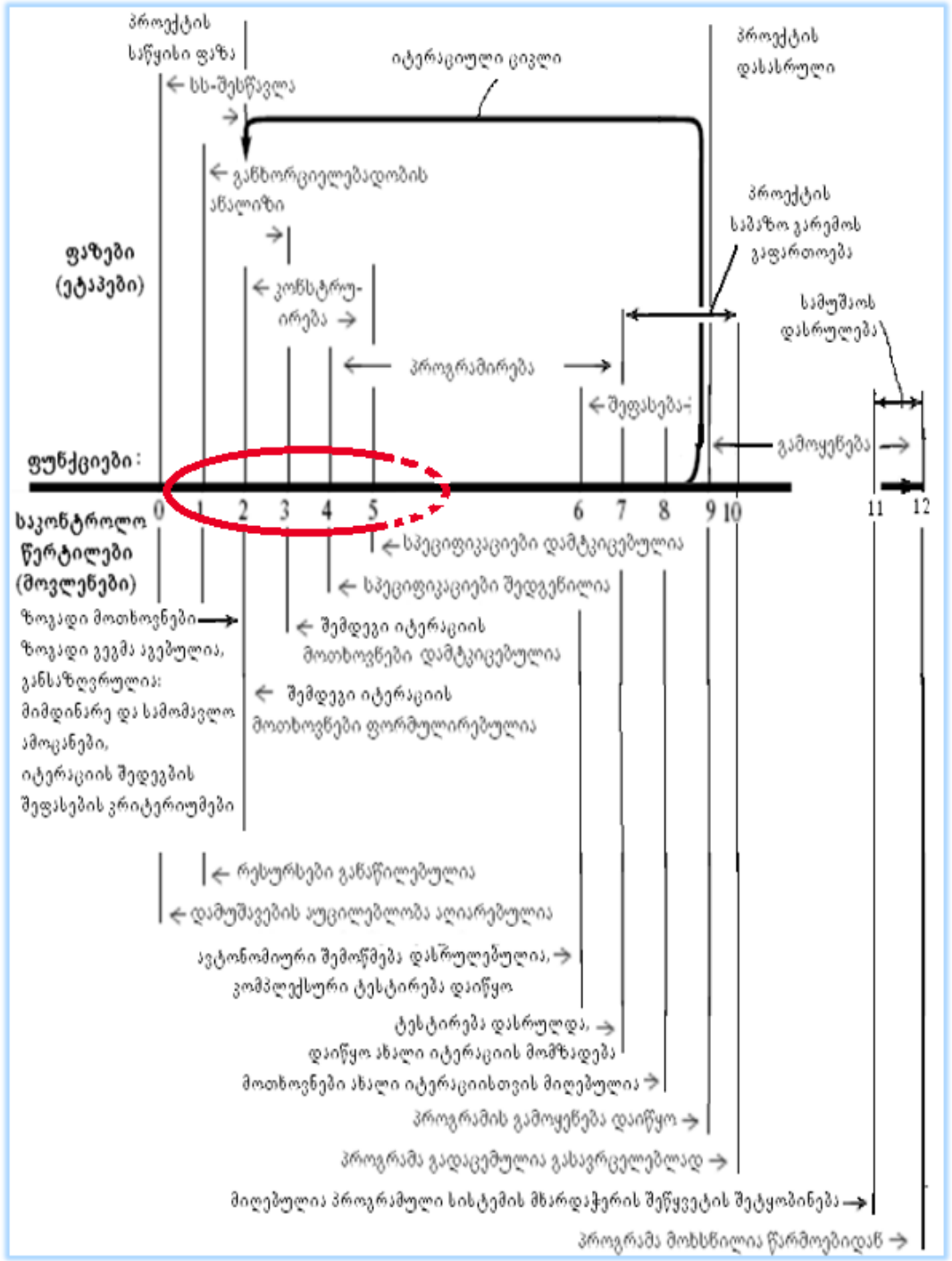
1.1. საინფორმაციო სისტემების პროგრამული უზრუნველყოფის ანალიზი და დაპროექტება UML ტექნოლოგიით

მართვის საინფორმაციო სისტემების სრულყოფილი, საიმედო და მოქნილი პროგრამული უზრუნველყოფის (Software) სწრაფად დაპროექტება, რეალიზაცია, დანერგვა და შემდგომი თანხლება სისტემის დამკვეთ ორგანიზაციაში მეტად მნიშვნელოვანი ამოცანაა. მისი ეფექტურად გადაწყვეტა ბევრადაა დამოკიდებული როგორც საპროექტო-დეველოპმენტის გუნდის შემადგენლობა-გამოცდილებაზე, ისე IT-ინფრასტრუქტურასა და CASE-ინსტრუმენტებზე [10].

განსაკუთრებით მნიშვნელოვანია დიდი სისტემების პროექტირების პროცესში ობიექტორიენტირებული დაპროგრამების მეთოდისა და უნიფიცირებული მოდელირების ენის (UML - ტექნოლოგია) გამოყენება. პროგრამის სისტემების სასიცოცხლო ციკლი მოითხოვს მისი აუცილებელი ეტაპების იტერაციულ განვითარებას (ნახ.1.1) [14,21]. აქ ნაჩვენებია პროგრამული უზრუნველყოფის დამუშავების სასიცოცხლო ციკლის ეტაპები განტერის „ფაზა-ფუნქციების“ მოდელის საფუძველზე [10].

პროგრამული სისტემის მენეჯმენტის საკონტროლო (0-12) წერტილებში, ეტაპების მიხედვით ხორციელდება იტერაციული სამუშაოები (დაბრუნება უკანა წერტილებში განმეორებითი პროცედურების ჩასატარებლად), სისტემის ფუნქციონალობის სისრულის დაზუსტების ან გაფართოების მიზნით. კონკრეტული პროექტის ამოცანებისა და მოთხოვნების შესაბამისად უნდა განისაზღვროს როგორც სამუშაო გუნდის შემადგენლობა (როლები), ასევე სისტემის ინფრასტრუქტურა, დაპროგრამების მეთოდები და ინსტრუმენტები. გუნდში მონაწილე როლები შეიძლება იყოს: დამკვეთი, პროექტის მენეჯერი, ბიზნეს-ანალიტიკოსი, სისტემის არქიტექტორი, დეველოპერი-პროგრამისტი, ტესტირების სპეციალისტი და ა.შ. პროგრამული სისტემის პროექტის მენეჯერი ახორციელებს ყველა საკონტროლო წერტილის მონიტორინგს (ნახ.1.1).

დიდი პროექტებისათვის, სადაც რესურსები და დროითი ფაქტორები შედარებით კრიტიკული არაა, ხდება ობიექტორიენტირებული მიდგომის ყველა ეტაპისა და ფაზის გამოყენება შესაბამისი საკონტროლო წერტილების აუცილებელი მონიტორინგით და რეპორტებით. ამ დროს სრული მოცულობით ხორციელდება უნიფიცირებული მოდელირების ენის (UML/2) და შესაბამისი ინსტრუმენტული საშუალების, მაგალითად, Enterprise Architect პაკეტის გამოყენება [22,23]. ეს ინსტრუმენტი რეალიზებულია აგრეთვე Ms Visual Studio.NET 2015-ის ვერსიაშიც, რომელიც საშუალებას იძლევა UML დიაგრამების (მაგალითად, Class-D, Sequence_D) გენერირებულ იქნას ავტომატურად C# კოდი [10,24].



ნახ.1.1. განტერის მოდელი იტერაციით ობიექტორიენტირებული პროგრამული სისტემის პროექტისათვის

მოდელირების უნიფიცირებული ენა – UML (Unified Modeling Language) შექმნილია, როგორც უნივერსალური მოდელირების ენა ობიექტორიენტირებული პროგრამირების სფეროში და არის სტანდარტული ვიზუალური მოდელირების ენა, რომელიც იძლევა საშუალებას, სისტემა აღიწეროს გრაფიკულად და ტექსტურად [13].

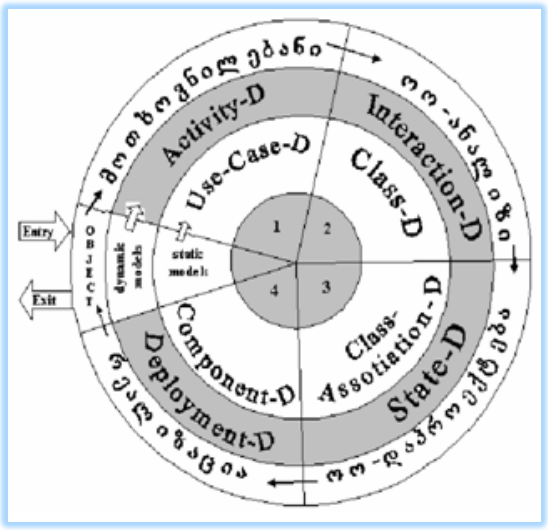
UML პრაქტიკულად წარმოადგენს ბუჩის მეთოდის (Booch Method), ობიექტის მოდელირების ტექნიკის (Object-modeling technique – OMT) და ობიექტორიენტირებული პროგრამული უზრუნველყოფის ინჟინერიის (Object-oriented software engineering - OOSE) სინთეზსს და გვევლინება როგორც ერთი, საერთო და ფართო გამოყენების მოდელირების ენა [14]. იგი მსოფლიოში ყველაზე ფართოდ გამოყენებადი უნიფიცირებული მოდელირების ენაა, შექმნილია საერთაშორისო ასოციაციის, ობიექტების მართვის ჯგუფის – OMG (Object Management Group) მიერ, რომელიც ქმნის ღია სტანდარტებს ობიექტორიენტირებული აპლიკაციებისათვის, ანუ UML არის გრაფიკული ენა, რომელიც გამოიყენება ობიექტ-ორიენტირებული მოდელირების აღწერისათვის პროგრამული უზრუნველყოფის სფეროში და, რაც მნიშვნელოვანია, იგი არის „ღია სტანდარტი“, რომელიც ხელმისაწვდომია ყველასათვის.

UML აერთიანებს მონაცემთა მოდელირების (entity relationship diagrams) ბიზნეს-მოდელირების (work flows), ობიექტების და კომპონენტების მოდელირების მეთოდებს. ის გამოიყენება პროგრამული უზრუნველყოფის და მისი ტექნიკური რეალიზაციის მთელი სასიცოცხლო ციკლის განმავლობაში. UML-ის მიზანია იყოს სტანდარტული მოდელირების ენა, რომლის საშუალებითაც შესაძლებელია პარალელური და განაწილებული სისტემის მოდელების შექმნა [25,26].

შემდეგ პარაგრაფში დეტალურად განვიხილავთ უნიფიცირებული მოდელირების ენის გამოყენების საფუძველზე შავი ზღვის ეკოლოგიური მონიტორინგის სისტემის მხარდამჭერი პროგრამული სისტემის დაპროექტების პროცესს შესაბამისი ფუნქციონალური ბიზნესპროცესების ანალიზისა და გრაფიკული დიაგრამების ფორმირებით [13].

1.2. შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის UML დიაგრამები

UML ტექნოლოგია განაწილებული მართვის საინფორმაციო სისტემების დაპროექ-ტების მეთოდოლოგიური საფუძველია. პროგრამული პაკეტების აგების პროცესის სტანდარტიზაცია სამი ძირითადი მიმართულების „გენეტიკური“ მემკვიდრეა: დაპროექტების ავტომატიზაცია, დაპროგრამების ავტომატიზაცია და მონაცემთა ბაზების აგების ავტომატიზაცა [27]. შავი ზღვის ეკოლოგიური მონიტორინგის საინფორმაციო სისტემის აგების მაგალითზე ჩვენ განვიხილავთ ამ ძირითადი მიმართულებების ამოცანებს და მათი გადაწყვეტის გზებს. 1.3 ნახაზზე ნაჩვენებია UML ტექნოლოგიის კლასიკური მოდელი 4 წყვილი დიაგრამით (ობიექტორიენტირებული ეტაპები). მათგან 4 სტატიკური მოდელია და 4 - დინამიკური [13].



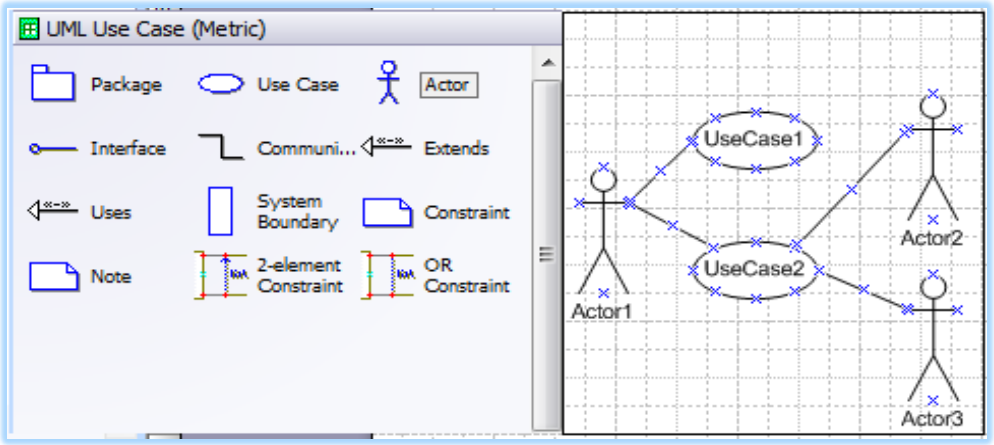
პირველი ეტაპი არის ასაგები კომპიუტერული სისტემის ფუნქციონალური მოთხოვნილებების განსაზღვრა. აქ მუშაობს ბიზნეს ანალიტიკოსი (საპრობლემო სფეროს სპეციალისტთან ერთად) და აგებს UseCase და Activity დიაგრამებს.

UseCase ასახავს სისტემაში მონაწილე როლებს და მათ ფუნქციებს, ხოლო Activity კი - ფუნქციების დეტალურ ბიზნეს-პროცესებს და ბიზნეს-წესებს.

ნახ.1.3. UML- ტექნოლოგიის 4-ეტაპიანი სქემა

1.2.1. Use Case დიაგრამა

პირველი მოდელი, რომელიც UML ტექნოლოგიით პროექტდება, არის გამოყენებით შემთხვევათა (პრეცედენტების) UseCase დიაგრამა. იგი როლების (Actors) და ფუნქციების (Actions) ურთიერთდაკავშირებული სქემაა (ნახ.1.4). აქ მაგალითად, UseCase1 ეკუთვნის მხოლოდ პირველ როლს, ხოლო UseCase2-ის შესასრულებლად სამივე როლი მონაწილეობს.



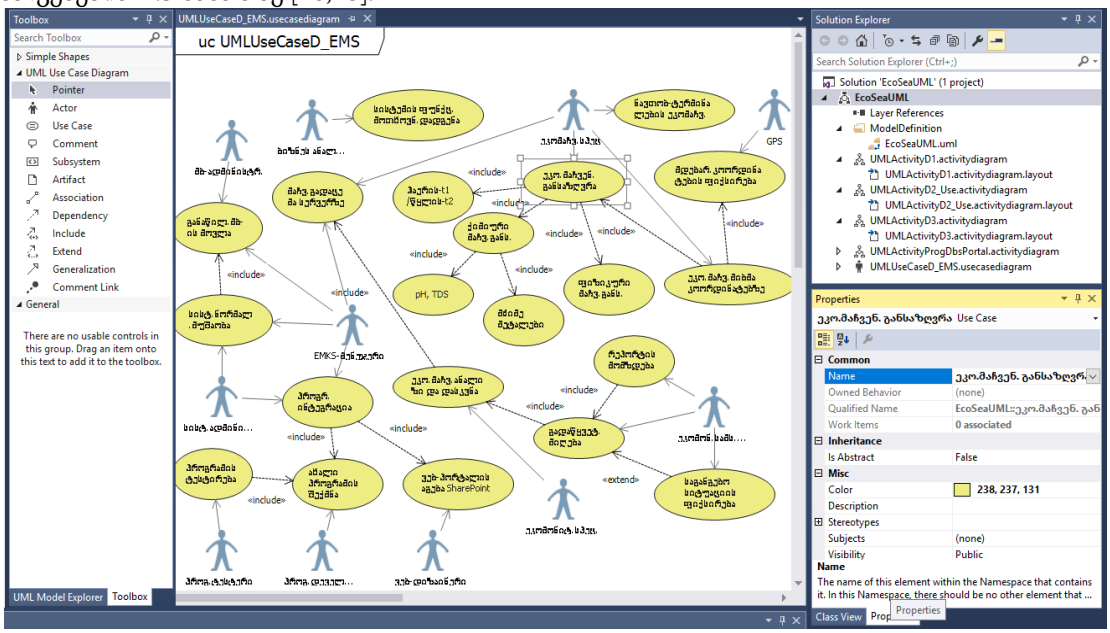
ნახ.1.4. UseCase დიაგრამის აგების ინტერფეისი (MsVisio)

შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის აგების და ექსპლუატაციის პროცესში მონაწილეობენ შემდეგი როლები (ფუნქციებით):

- **ბა** – ბიზნეს ანალიტიკოსი (ფუნქციური მოთხოვნილებების ანალიზი);

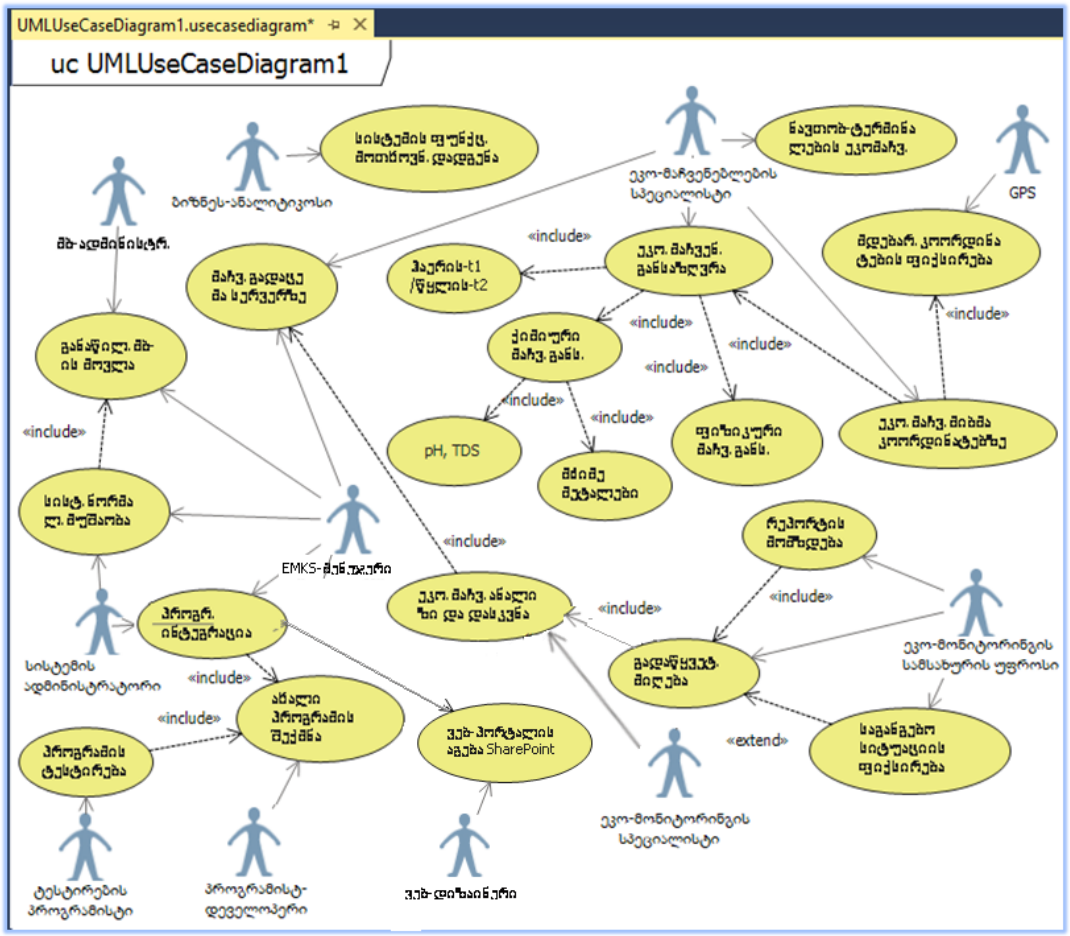
- **ეკს** – ეკოლოგიური მაჩვენებლების სპეციალისტი (ფიზიკური, ქიმიური ან სხვ. სახის ეკოლოგიური მაჩვენებლების განსაზღვრა);
- **ემს** – ეკოლოგიური მონიტორინგის სპეციალისტი (ეკოლოგიური მაჩვენებლების მნიშვნელობების ანალიზი და დასკვნების გაკეთება);
- **ემსუ** – ეკოლოგიური მონიტორინგის სამსახურის უფროსი (ეკოლოგიური მდგომარეობის შესაბამისად გადაწყვეტილების მიღება);
- **კს** – კომპიუტერული სისტემა (პროგრამული უზრუნველყოფა ეკოლოგიური მონიტორინგის განხორციელება);
- **GPS** – გლობალური პოზიციონირების სისტემა (ადგლიმდებარეობის კოორდინატების განსაზღვრა);
- **მბა** – მონაცემთა ბაზის ადმინისტრატორი (მონაცემების განახლება, კონტროლი და დაცვა);
- **სა** – სისტემის ადმინისტრატორი (ეკო-მონიტორინგის კს-ის ფუნქციონირება);
- **პდ** – პროგრამისტ-დეველოპერი (გამოყენებითი პროგრამების რეალიზაცია);
- **პტ** – პროგრამების ტესტირების სპეციალისტი (ახალი პროგრამების ტესტირება, ვერიფიკაცია და ვალიდაცია);
- და სხვ. (შესაძლებელია დაემატოს ახალი როლი კონკრეტული გარემოებიდან გამომდინარე).

Visual Studio.NET 2015 პაკეტის სამუშაო გარემო პრეცედენტების დიაგრამის ასაგებად ნაჩვენებია 1.5 ნახაზზე [10,28].



ნახ.1.5. Visual Studio.NET 2015-ში EcoSeaUML პროექტის აგება

თვით UseCase დიაგრამა შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემისათვის ნაჩვენებია 1.6 ნახაზზე.



ნახ.1.6. UseCase დიაგრამა ეკო-მონიტორინგის სისტემისათვის (EMS)

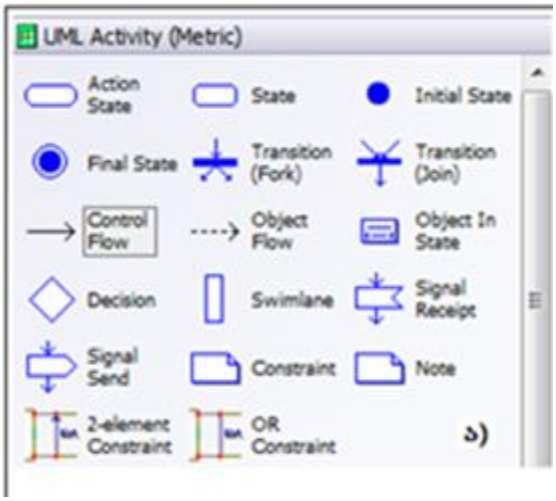
როგორც ფუნქციონალური მოთხოვნილებების ანალიზმა გვიჩვენა, ეკოლოგიური მონიტორინგის სისტემის ბიზნეს-პროცესები (საქმიანი ნაკადები) იწყება უშუალოდ ზღვის სანაპიროს საკონტროლო პუნქტებზე (მდინარეთა ესტუარები, ნავთობტერმინალები, პორტები და სხვ.) ჰაერისა და წყლის ტემპერატურების, წყლის მარილიანობისა და მჟავიანობის, აგრეთვე მძიმე მეტალებით დაბინძურების და სხვ. მონაცემების შეგროვებით (როგორც ნახაზიდან ჩანს) და ცენტრალურ სერვერზე გადაცემით [1,12], კომპიუტერული სისტემა სპეციალური (მონიტორინგის სპეციალისტების) პროგრამული უზრუნველყოფით ამუშავებს ამ მონაცემებს და ცხრილური და გრაფიკული შედეგების სახით გადასცემს მონიტორინგის სამსახურის უფროსს - გასაცნობად და საბოლოო გადაწყვეტილების მისაღებად.

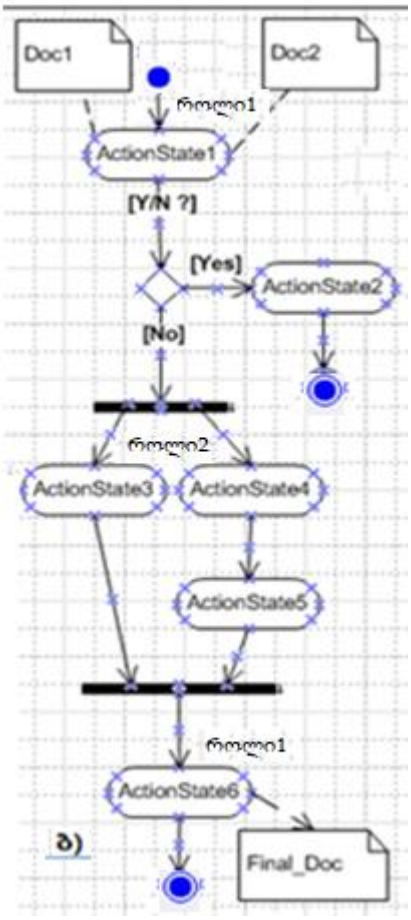
ბიზნეს-წესების თანახმად, თუ მონაცემთა დასაშვები მნიშვნელობები მეტია ზღვრულ ნორმატიულზე, მაშინ მონიტორინგის სამსახურის უფროსი აფიქსირებს საგანგებო სიტუაციას და ღებულობს გადაწყვეტილებას (არსებული რეგლამენტის მიხედვით) გარკვეული ღონისძიებების ჩატარების შესახებ.

აღნიშნული ბიზნეს-პროცესები და ბიზნეს-წესები UML-ტექნოლოგიაში აისახება აქტიურობის დიაგრამის (Activity-D) სახით, რომელსაც მომდევნო პარაგრაფში განვიხილავთ.

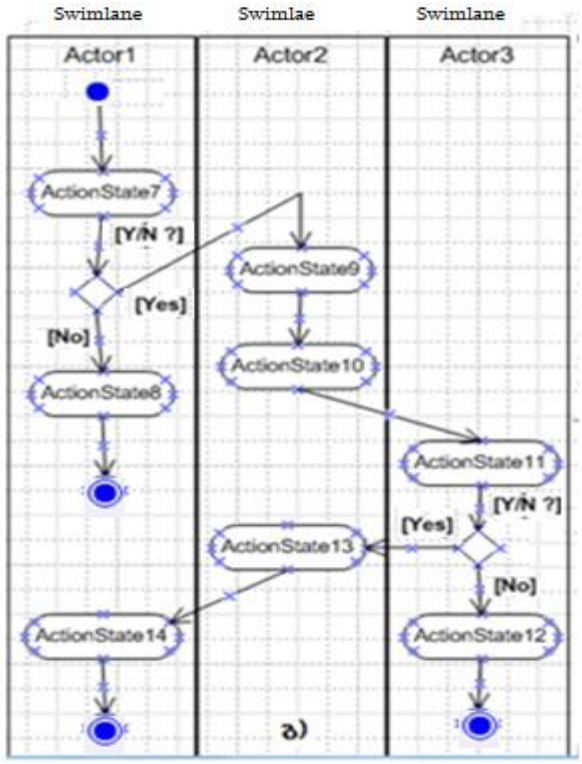
1.2.2. Activity დიაგრამა

ბიზნესპროცესებისა და ბიზნესწესების გაცნობის, ანალიზის და სტრუქტურული ფორმალიზაციის საფუძველზე აიგება აქტიურობის (ქმედებათა) დიაგრამა. იგი კონკრეტული როლის (როლების) კონკრეტული ფუნქციაა, რომელიც შედეგება იერარქიულად სივრცესა და დროში დალაგებული მიმდევრობით ან პარალელურად შესასრულებელი სუბქმედებებისგან. აქვს ერთი დასაწყისი და რამდენიმე შესაძლო დასასრული, ბიზნესწესებით განსაზღვრული განშტოების ან შეერთების პროცედურები, საწყისი, შუალედური ან საშედეგო დოკუმენტაცია და ა.შ. საილუსტრაციო მაგალითი მოცემულია 1.7-ა,ბ ნახაზზე [13].





ნახ.1.7. Activity დიაგრამის:
ა) ინსტრუმენტების პანელი; ბ) ქმედებათა სქემა (MsVisio)



Swimlane – ელემენტის გამოყენებით აგებენ ალტერნატიული ფორმის აქტიურობის დიაგრამას, რომელიც ცალკეული როლების შესაბამისი „საცურაო ბილიკების“ მსგავსია (ნახ.1.7-გ).

თითეულ ბილიკში (Swimlane) თავსდება ერთი კონკრეტული როლის (Actor) ყველა ქმედება (ActionState).

განშტოების ელემენტი (Decision-რომბი) სემანტიკურად წარმოადგენს ბიზნეს-წესს, რომლის მიხედვითაც განისაზღვრება მომდევნო შესასრულებელი ქმედება, შესასვლელზე თავსდება ლოგიკური პირობა, გამოსასვლელზე კი - პირობის შესრულების ან არშესრულების დროს ასარჩევი ქმედება.

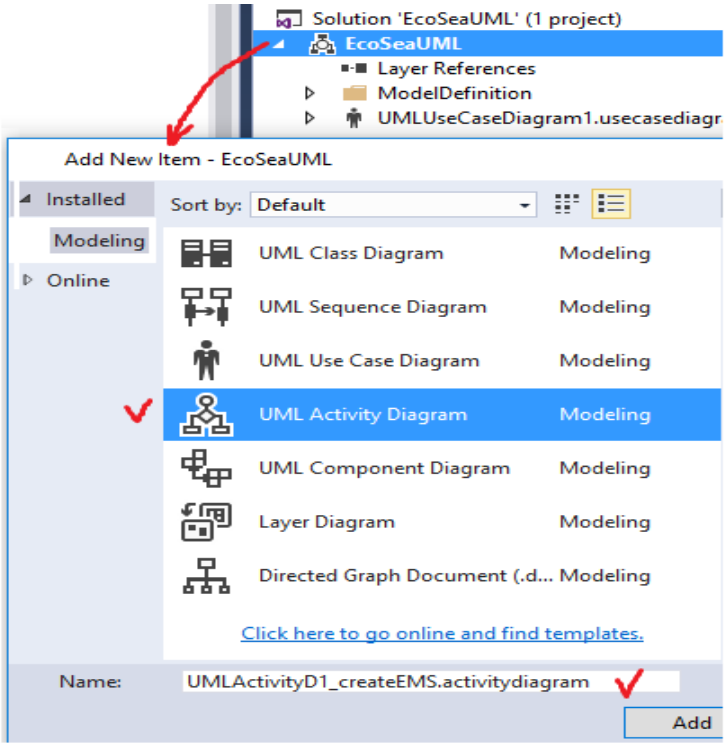
ნახ.1.7-გ. Activity დიაგრამის:
გ) ქმედებათა სქემა (მართვის სფეროების ან როლების ბილიკებით - MsVisio)

აქტიურობათა (ქმედებათა) დიაგრამა მიეკუთვნება პროცესების აღწერის დინამიკურ მოდელს, იგი ასახავს საკვლევი ობიექტის ქცევას. ასეთი დინამიკური მოდელების პროცესების გამოსაკვლევად, რიგ შემთხვევებში, გამოიყენება პეტრის ქსელები, როგორც იმიტაციური მოდელირების ინსტრუმენტი [29,30].

საბოლოოდ, ამ ორი სახის (UseCase, Activity) დიაგრამათა ერთობლიობის ანალიზის საფუძველზე კეთდება დასკვნები საავტომატიზაციო ობიექტის მართვის სისტემის ფუნქციური და არაფუნქციური მოთხოვნილებების განსაზღვრის შესახებ.

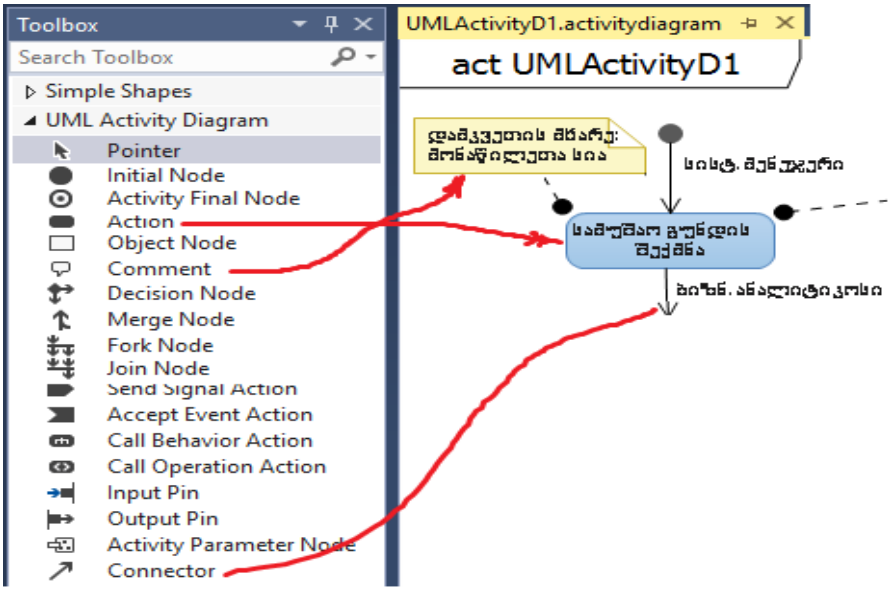
ამ ეტაპზე, ამავდროულად დგება მომავალი პროგრამული სისტემის შექმნის ტექნიკური დავალება შესაბამის ეკონომიკურ და ფინანსურ გაანგარიშებებთან ერთად, რომელიც, დამკვეთ ორგანიზაციის ხელმძღვანელობასთან კონსულტაციების შემდეგ, ორმხრივად მტკიცდება (ხელმოწერა, ბეჭედდასმა - დგება იურიდიული დოკუმენტი).

ახლა განვიხილოთ შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის შექმნისა და ექსპლუატაციის ბიზნეს-პროცესების აქტიურობათა დიაგრამების საილუსტრაციო მაგალითები. როგორც UseCase დიაგრამიდან (ნახ.1.6) ჩანს, როლები („კაცუნები“) და ფუნქციები („ოვალები“) დაკავშირებულია ერთმანეთთან გარკვეული მიკუთვნებისა და მიზეზ-შედეგობრივი მიმართებების საფუძველზე. ახლა პროექტს დავამატოთ აქტიურობის დიაგრამები Solution Explorer-დან Add New Item -ით(ნახ.1.8).



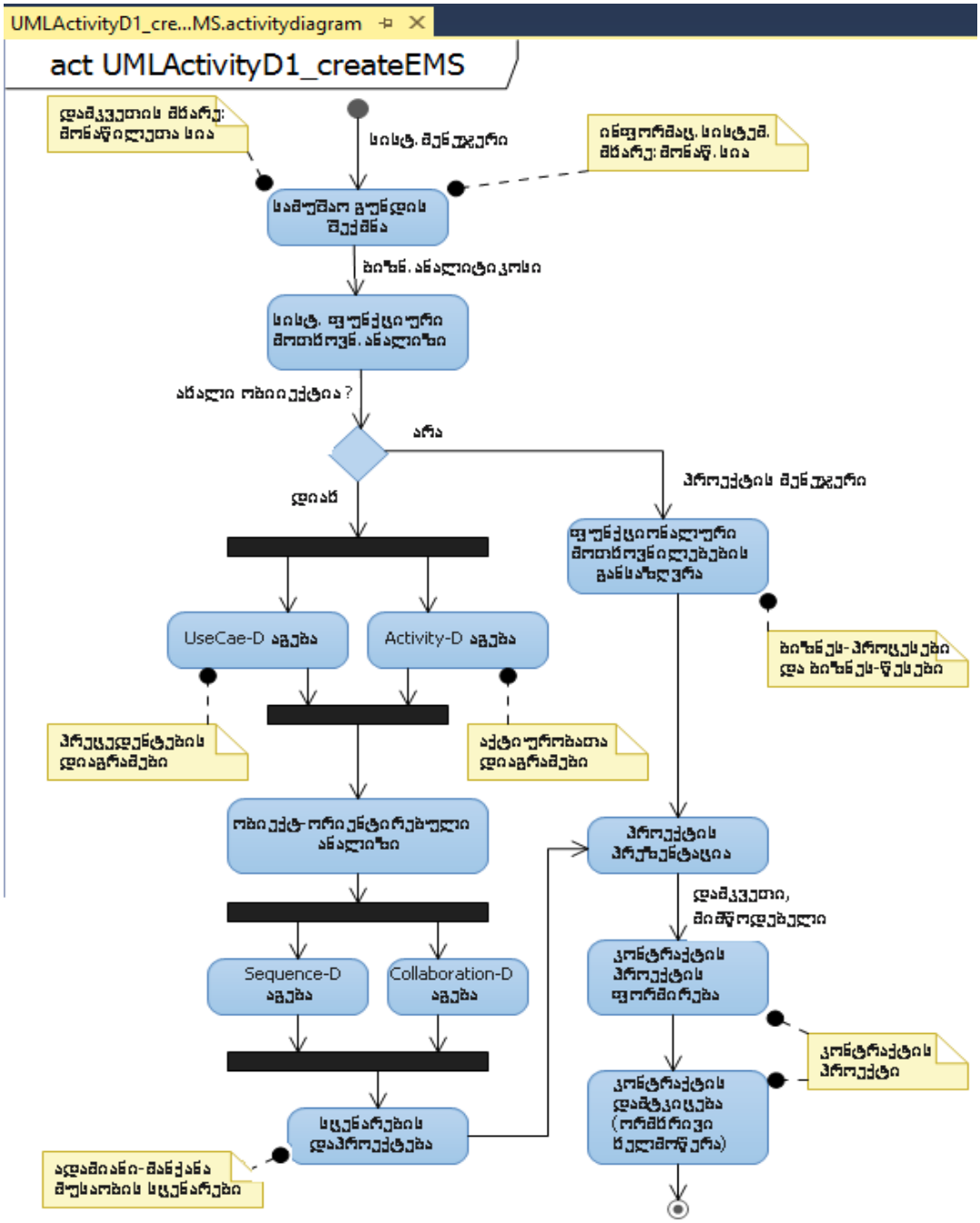
ნახ.1.8. EcoSeaUML პროექტს ემატება Activity დიაგრამა

1.9 ნახაზზე ნაჩვენებია აქტიურობის დიაგრამის აგების ინსტრუმენტების პანელი.

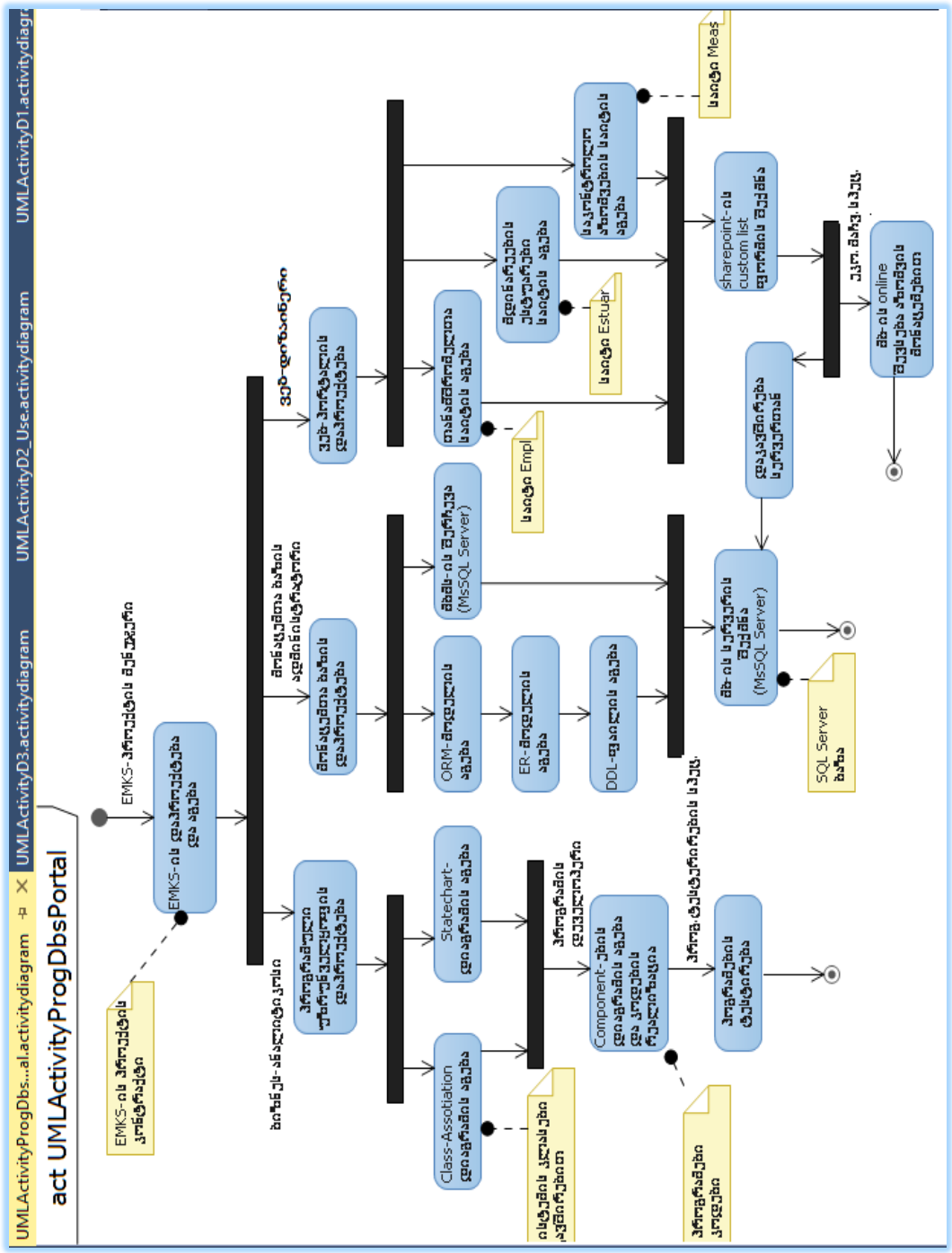


ნახ.1.9. UML Activity დიაგრამის ასაგები ინსტრუმენტის ელემენტები

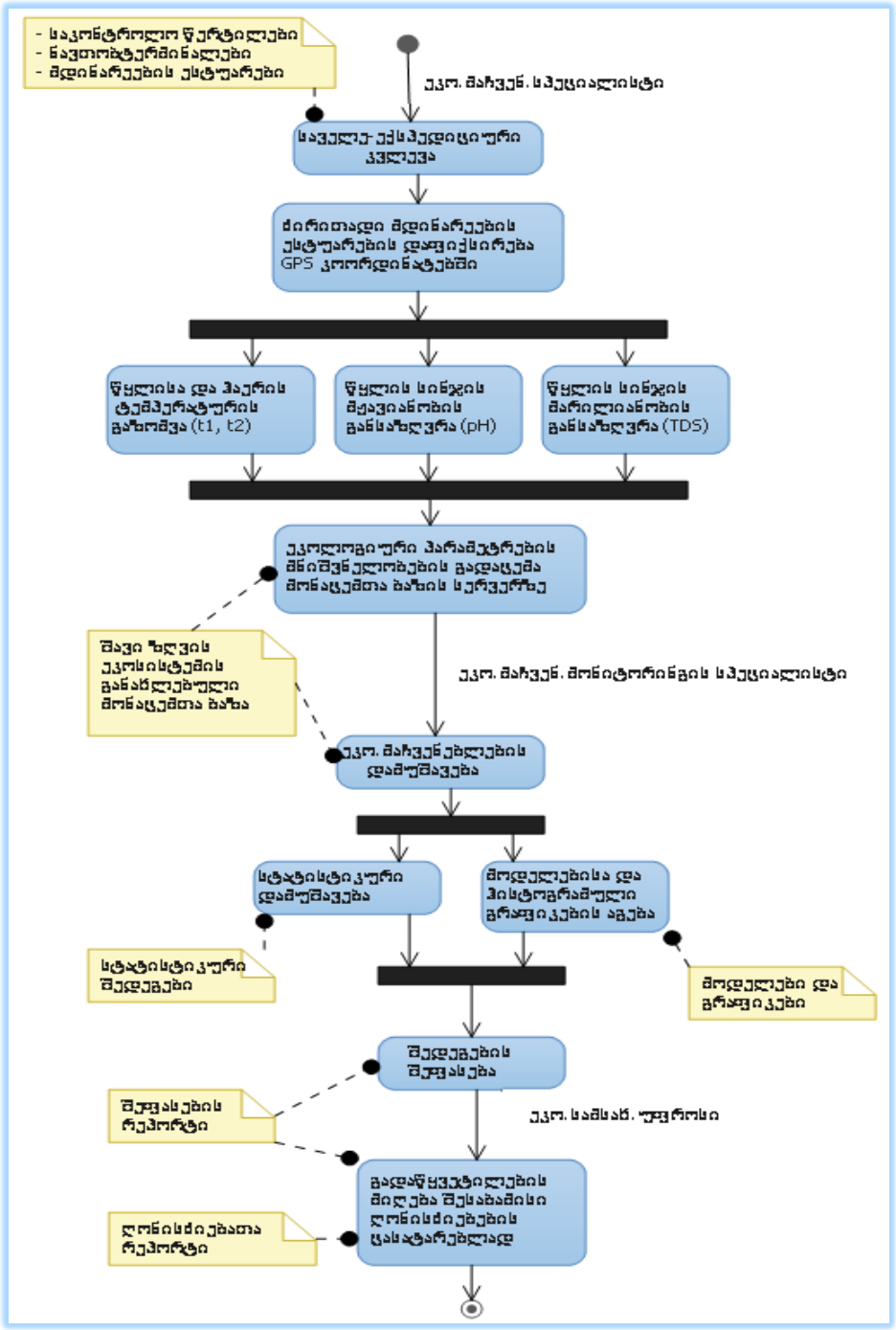
1.10-1.13 ნახაზებზე ნაჩვენებია ეკომონიტორინგის სისტემის (EMS) ანალიზის და დაპროექტების ბიზნეს-პროცესების ეტაპების აქტიურობათა დიაგრამები.



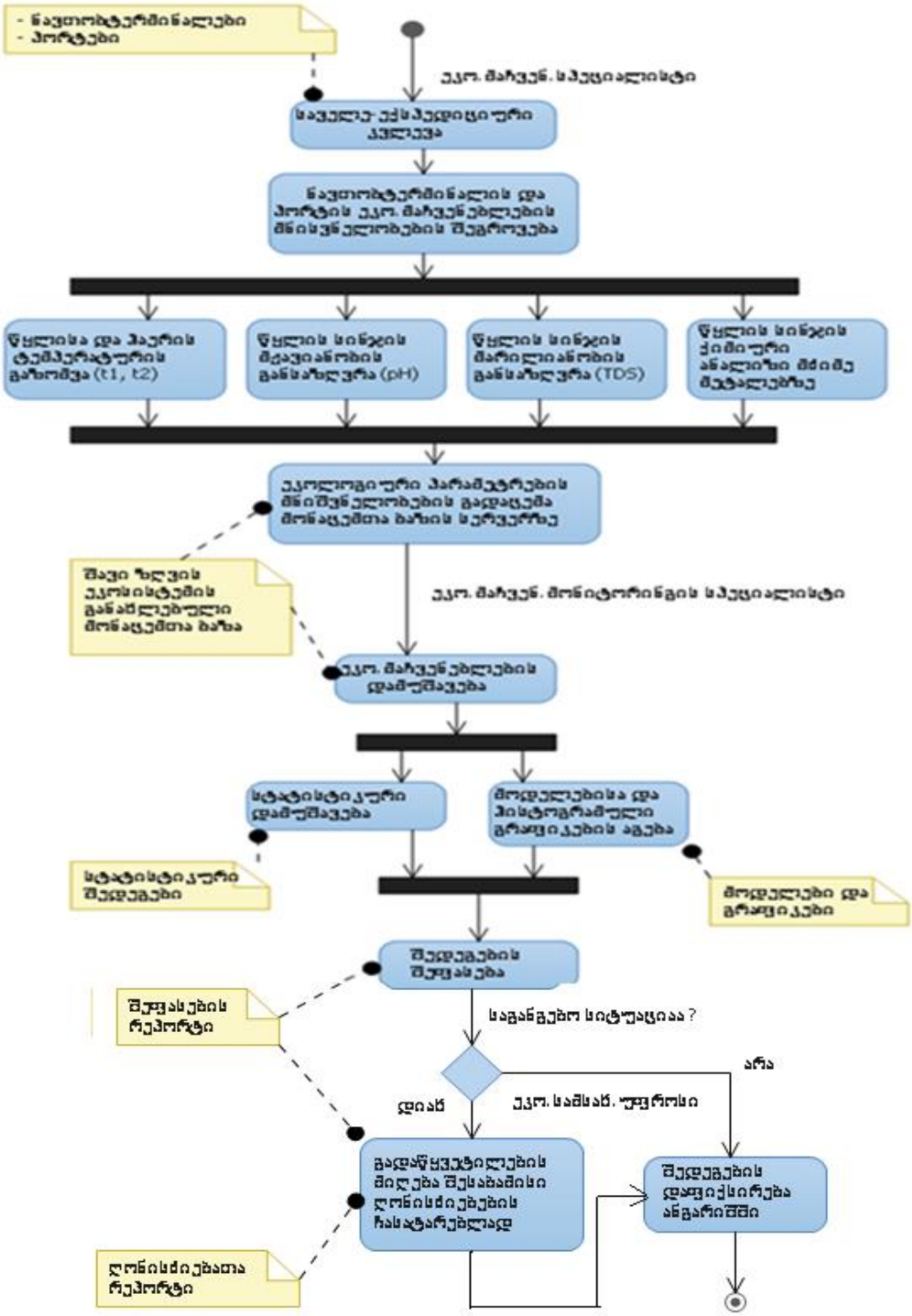
ნახ.1.10. EMS-ის ბიზნეს-პროცესების მოდელირებისა და პროექტის შესრულების კონტრაქტის გაფორმების აქტიურობის დიაგრამა (Activity-D1)



ნახ.1.11. EMCS დაპროექტების და პროგრამული რეალიზაციის აქტიურობის დიაგრამა (Activity-ProgDbPortal)



ნახ.1.12. EMCS-ის ფუნქციონირების ბიზნეს-პროცესი
შავი ზღვის მდინარეების ესტუარებისთვის (Activity-D2)

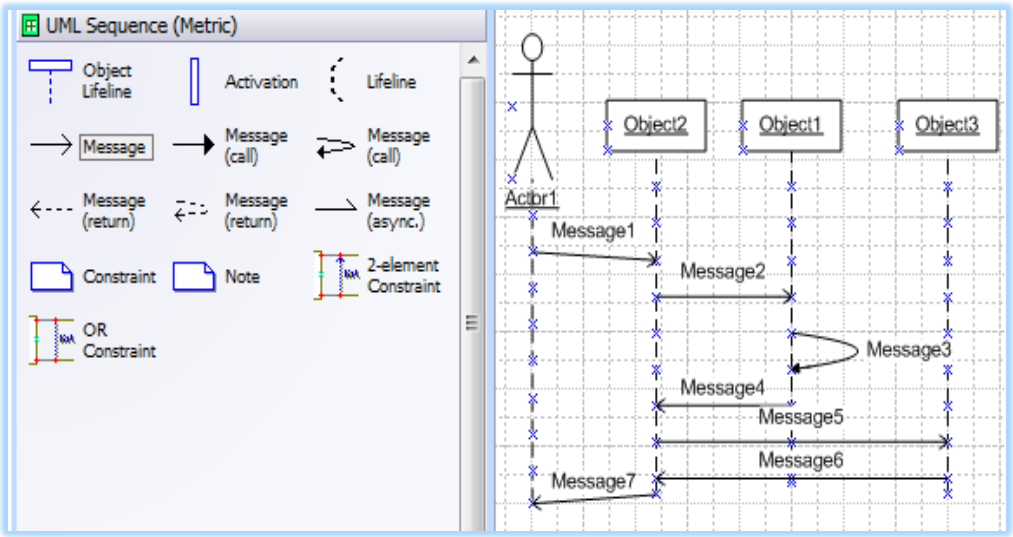


ნახ.1.1.13. EMCS-ის ფუნქციონირების ბიზნეს-პროცესი
შავი ზღვის ნავთობტერმინალებისა და პორტებისათვის (Activity-D3)

ამის შემდეგ იწყება ობიექტორიენტირებული ანალიზის ეტაპი, რომელზეც აიგება სისტემის მომხმარებელთა ინტერაქტიული სქემები, რომლებიც მათი კომპიუტერთან მუშაობის სცენარებს შეესაბამება. ესაა მიმდევრობითობის (Sequence-D) და თანამოქმედების (Collaboration-D) დიაგრამები [13,14].

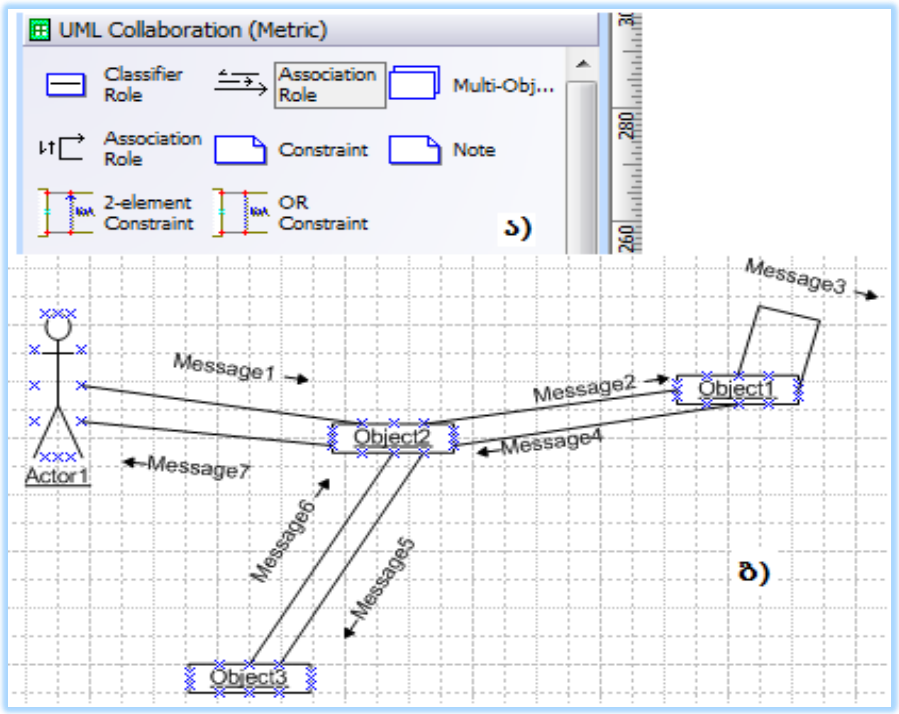
1.2.3. Sequence და Collaboration დიაგრამები

მიმდევრობითობის დიაგრამა აღწერს საპრობლემო სფეროს კონკრეტული როლის კონკრეტული ამოცანის შესრულების სცენარს. აქ ხდება როლის სისტემასთან ურთიერთ-ქმედების ბიზნესპროცესის ქმედებათა და მათი მაინიცირებელ, სინქრონულ ან ასინქრონულ შეტყობინებათა დროში მიმდევრობით განლაგება (ნახ.1.14) [13].



ნახ.1.14. Sequence დიაგრამისაგების ინტერფეისი (MsVisio)

1.15 ნახაზზე ნაჩვენებია თანამოქმედების დიაგრამის აგების ინსტრუმენტების პანელი (ა) და თვით დიაგრამა (ბ), რომელიც შესაბამისი მიმდევრობითობის დიაგრამის ტრანსფორმაციით იქნა მიღებული. აქ შეტყობინებათა და მონაცემთა გაცვლის მიმდევრობა არაა დროში დალაგებული, სამაგიეროდ ჩანს კლასის ობიექტებს შორის კავშირებისა და ინფორმაციული ნაკადების გაცვლის სემანტიკა. CASE ტექნოლოგიის მრავალ პროდუქტში (მაგალითად, ფირმა Rational Rose) თანამიმდევრობითობის დიაგრამის აგება ხდება ავტომატიზებულ რეჟიმში. ანუ ჯერ აიგება მიმდევრობითობის დიაგრამა და შემდეგ ინსტრუმენტის რომელიმე სწრაფი ღილაკით (მაგალითად, F5) სისტემა თვითონ ააგებს თანამოქმედების დიაგრამას.

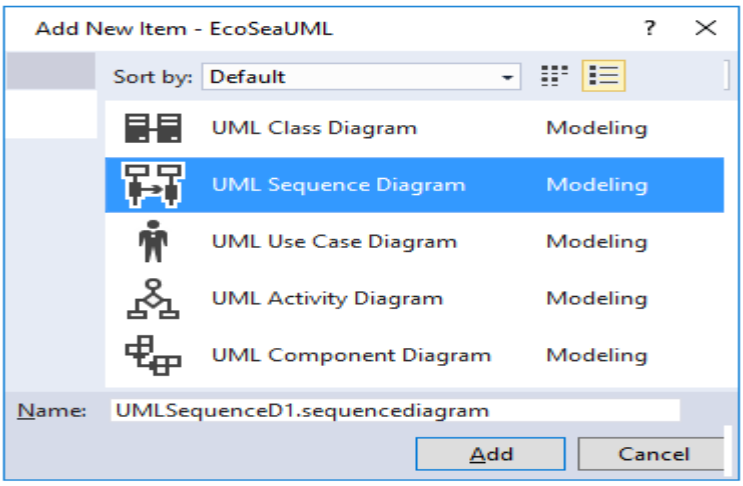


ნახ.1.15. Collaboration დიაგრამის აგების ინტერფეისი (ა)
და სქემის მაგალითი (ბ) (MsVisio)

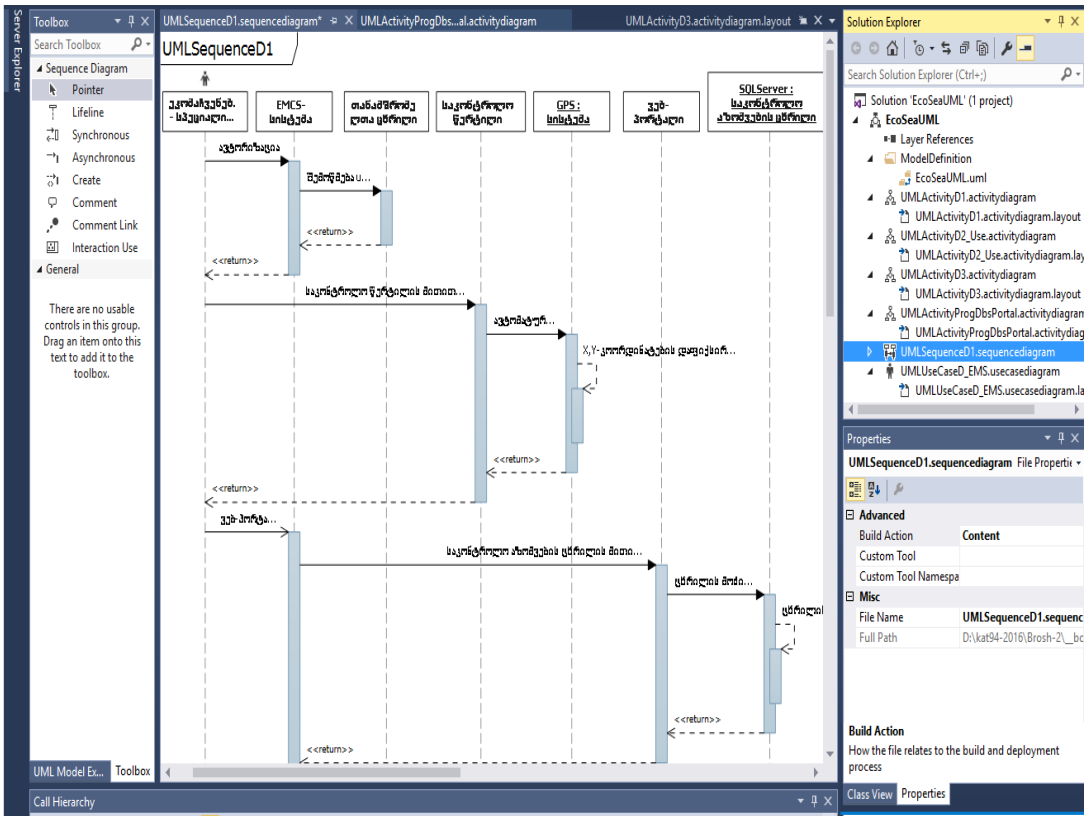
ახლა განვიხილოთ შავი ზღვის ეკოლოგიური მონიტორინგის კომპიუტერული სისტემის (EMCS) ექსპლუატაციის ამოცანებისთვის ინტერაქტიული დიაგრამები, კერძოდ, მიმდევრობითობის (Sequence) და თანამოქმედების (Collaboration) დიაგრამები.

ცხადია, რომ EMCS სისტემისა და მონაცემთა ბაზის ადმინისტრატორების და ვებ-დიზაინერის მიერ წინასწარ გამზადებულია პროგრამული აპლიკაცია. სისტემაში შეტანილია მომხამარებელთა user_name და password-ები. მაგალითად, ეკო_მაჩვენებლების სპეციალისტებისათვის (რომლებიც ზღვის სანაპირო ზოლის საკონტროლო წერტილებში, მდინარეების ესტუარებში, ზღვასთან მდებარე ნავთობტერმინალებზე და პორტებში იღებენ წყლის სინჯებს და ადგენენ მაჩვენებელთა მნიშვნელობებს).

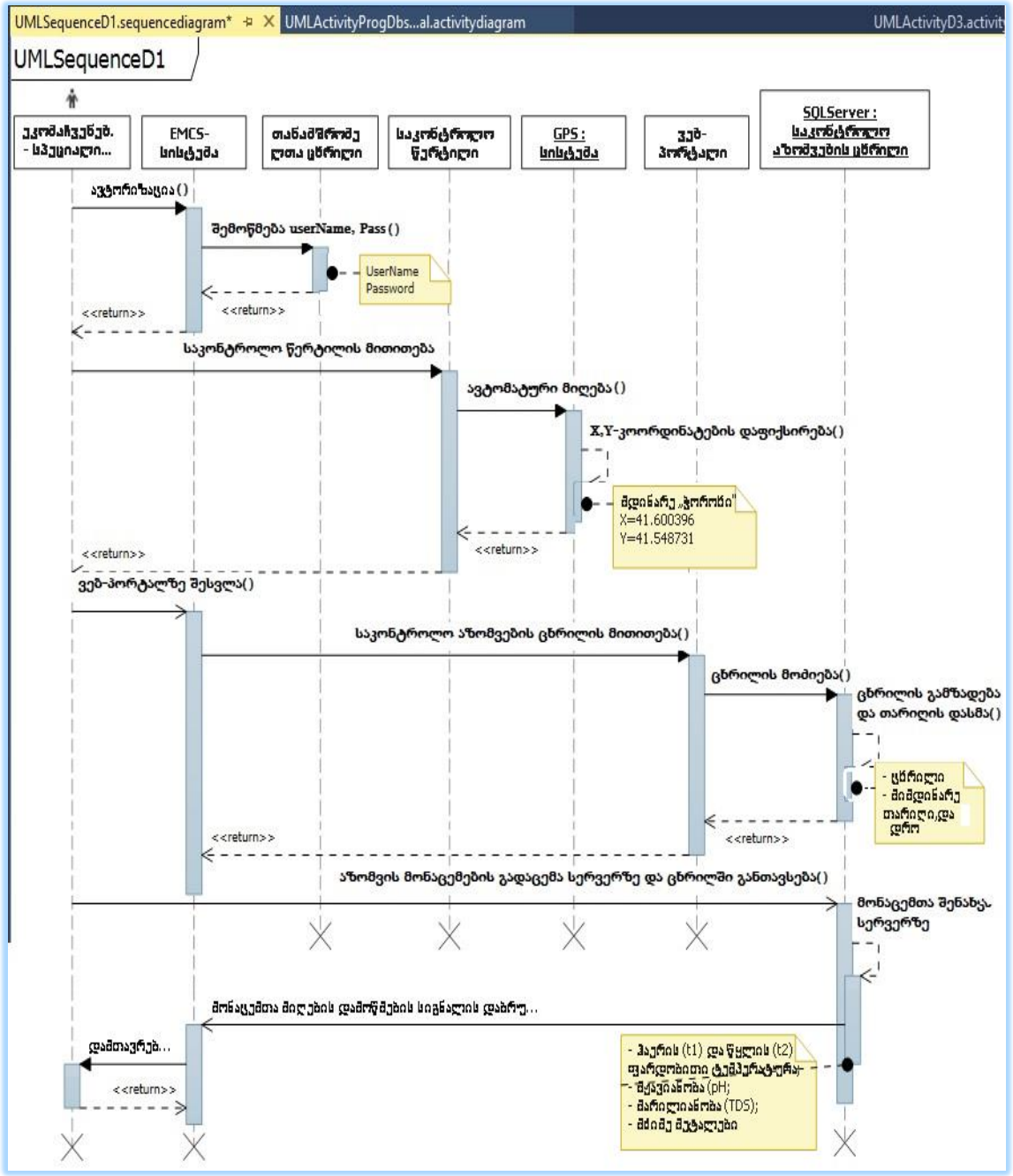
შექმნილად ითვლება ასევე მდინარეთა ესტუარების გეოგრაფიული კოორდინატები GPS სისტემის დახმარებით და საკონტროლო აზომვების შესანახი ცხრილები (MsSQL server-ზე) [12,15]. ამგვარად, „ეკომაჩვენებლების სპეციალისტის“ ფუნქციებიდან გამომდინარე, მათ მოუწევთ ზღვის საკონტროლო წერტილებიდან მონაცემების online-რეჟიმში გადაგზავნა ეკოსისტემის სერვერზე (პლანშეტით, მობილური ტელეფონით ან ნოუთბუკით). 1.16 ნახაზზე ნაჩვენებია ახალი კომპონენტის (SequenceD) დამატება Visual Studio.NET 2015 სამუშაო გარემოში (ნახ.1.17).



ნახ.1.16. Sequence დიაგრამის დამატების ინიციალიზაცია



ნახ.1.17. მიმდევრობითობის დიაგრამის აგება Swquence D1
(Visual Studio_2015)



ნახ.1.18. UML SequenceD როლისათვის „ეკომაჩვენებლის სპეციალისტი", ბიზნეს-პროცესი: „ეკომავენებლების გადაცემა საკონტროლო წერტილიდან ცენტრალურ სერვერზე"

ნახაზზე მართკუთხედები ასახავს კლასის ობიექტებს, რომელთანაც როლს „ეკომაჩვენებლების სპეციალისტს“ აქვს ურთიერთქმედება შეტყობინებების (Messages) გაცვლის დონეზე. შეტყობინების საფუძველზეც უნდა ამოქმედდეს შესაბამისი კლასის მეთოდები (გარკვეული ფუნქციების შესასრულებლად).

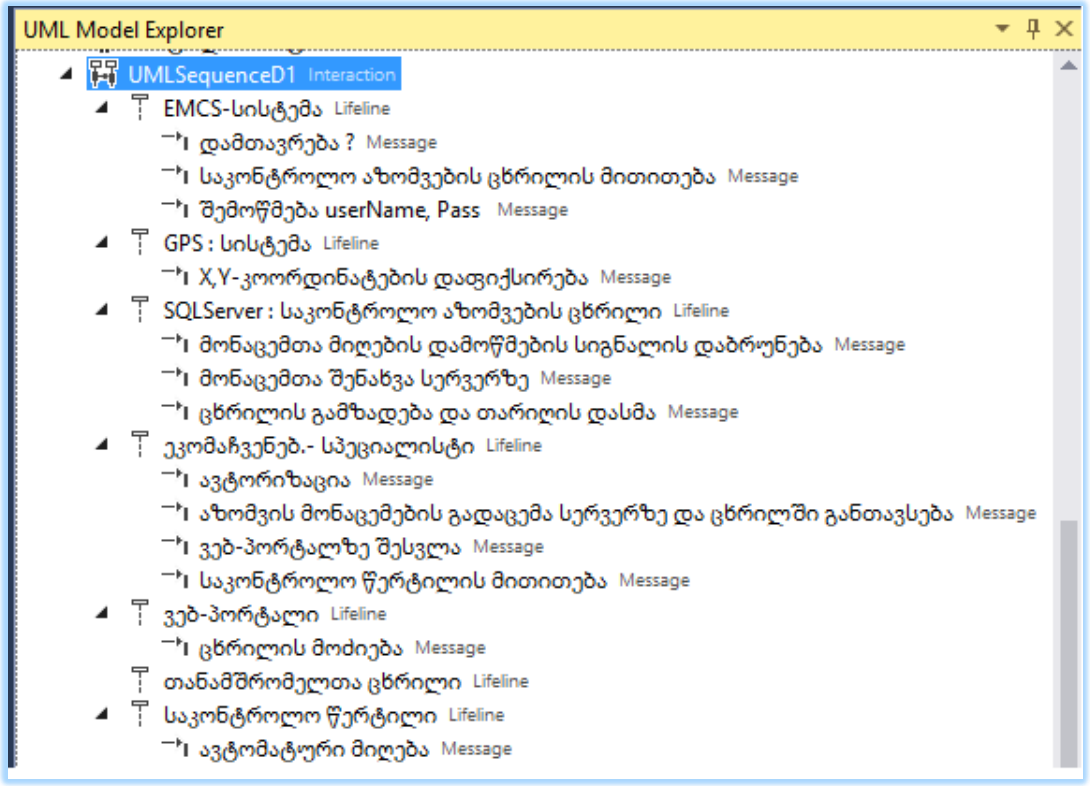
როლის ქმედებები ჩალაგებულია დროში მიმდევრობით ზემოდან ქვემოთ.

ეკომაჩვენებლების სპეციალისტი, როგორც შევთანხმდით, იღებს საკონტროლო წერტილში ჰაერის და წყლის ტემპერატურებს, წყლის სინჯით განსაზღვრავს მის მჟავიანობას და მარილიანობას და ა.შ. მონაცემებს აფიქსირებს თავის პირად ჟურნალში. ექსპერიმენტის ბოლოს მან მიღებული შედეგები უნდა გადააგზავნოს ეკოსისტემის ცენტრალურ სერვერზე. ამისათვის იგი მობილურით (ან ნოუთბუკით) აწვდის ეკოსისტემას თავის UserName და Password მნიშვნელობეს. EMCS ამოწმებს მომხმარებელთა ელექტრონულ ჟურნალს და თუ აღნიშნული UserName და Password მნიშვნელობები ნაცნობია სისტემისთვის, იგი <return> ფუნქციით აბრუნებს ნებართვის პასუხს, ანუ ჩვენ შემთხვევაში მომხმარებელი შედის ეკოსისტემაში მისი როლის შესაბამისი უფლებებით.

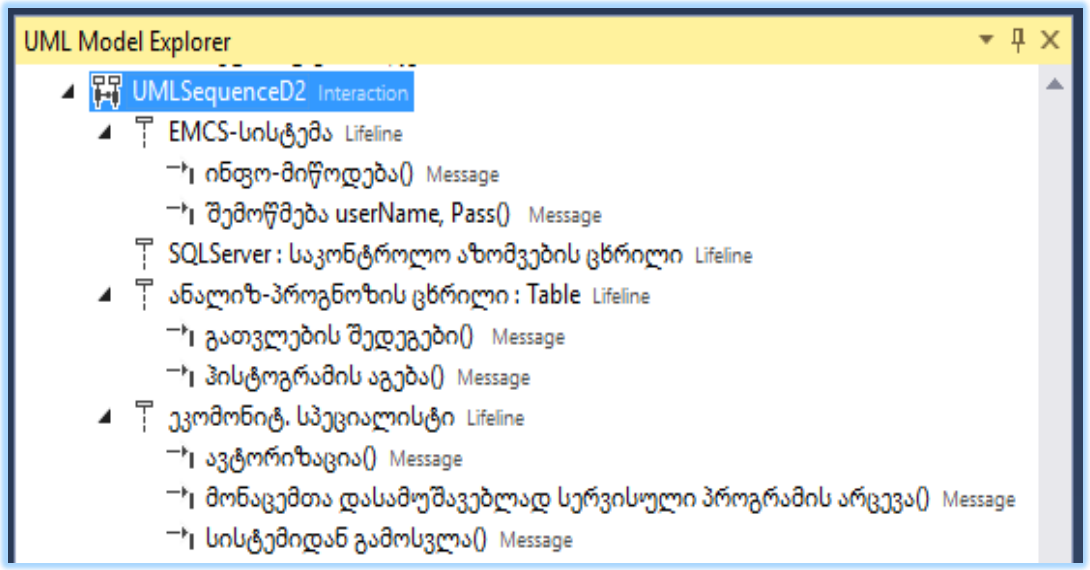
შემდეგ ეკომაჩვენებლების სპეციალისტი უერთდება ეკოსისტემის ვებ-პორტალს და მიუთითებს მას საკონტროლო წერტილის ნომერს (ან უბნის დასახელებას, სადაც იმოფება), რის საფუძველზეც GPS (გეოპოზიციონირების სისტემა) აფიქსირებს ამ მომხმარებლის საკონტროლო წერტილის ადგილმდებარეობას გეოგრაფიულ X,Y კოორდინატებში. ამასთანავე ავტომატურად აფიქსირებს მომდინარე თარიღს და დროს.

სპეციალისტი მიუთითებს სისტემას საკონტროლო აზომვების ცხრილის სახეს, რომლის მონაცემებიც უნდა გადასცეს მას. სისტემაში იხსნება შესაბამისი ცხრილი და სპეციალისტს შეაქვს მონაცემები: ჰაერის და წყლის ტემპერატურათა ფარდობითი მნიშვნელობა, წყლის მჟავიანობა და მარილიანობა. შესაძლებელია ასევე ქიმიური ლაბორატორიის შედეგების გადაცემაც, როგორიცაა, მაგალითად, წყლის დაბინძურების დონე მძიმე მეტალებით. 1.19 ნახაზზე ნაჩვენებია პროექტის შიგთავსის ფრგამენტი Sequence-D1 დიაგრამისთვის.

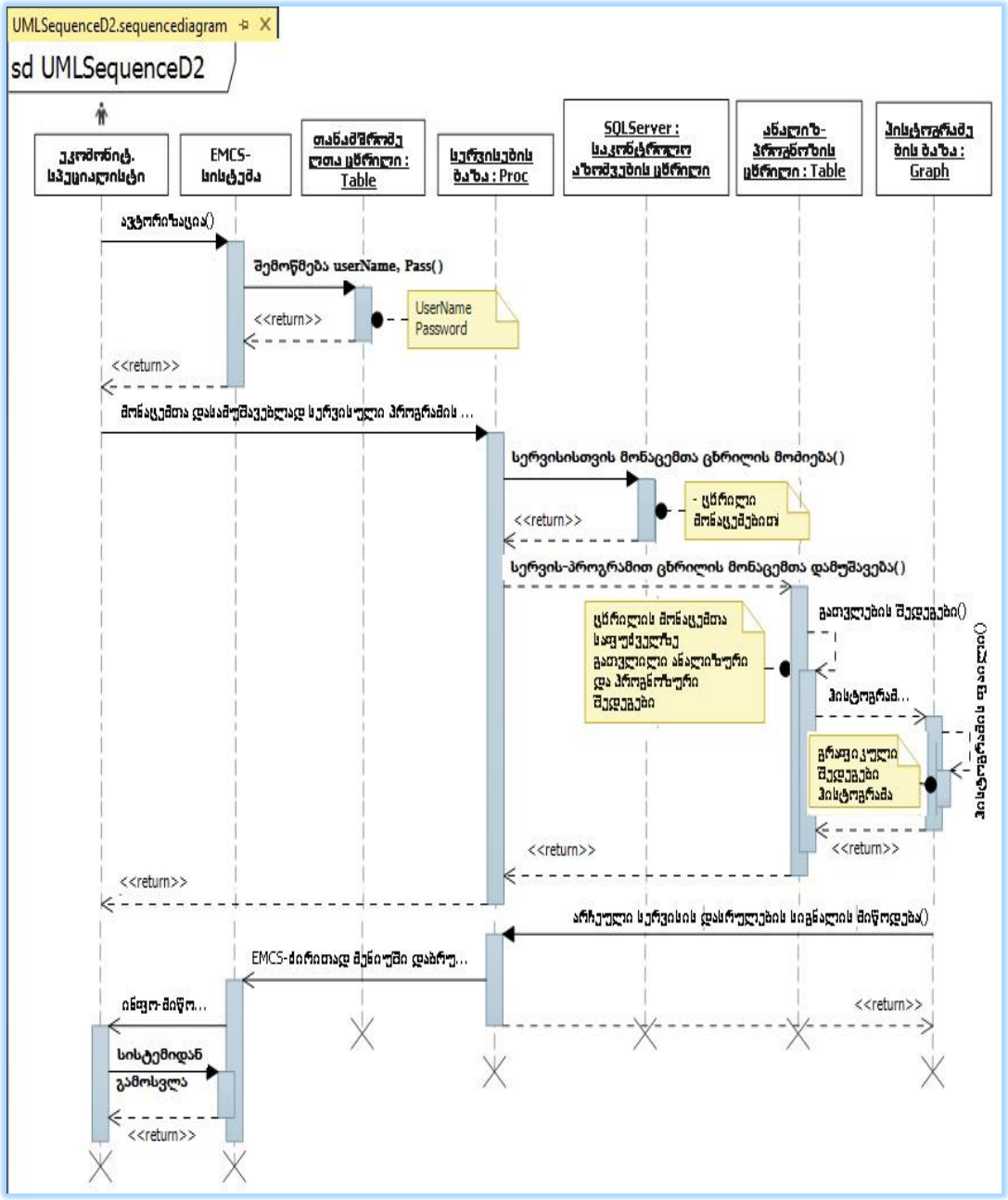
ელექტრონული ეკომონიტორინგის სისტემის ფუნქციური მომხმარებელი, რომელიც სიტემის სერვერ-ბაზიდან იღებს მონაცემებს და ატარებს კომპლექსურ დამუშავებას, არის „ეკო-მონიტორინგის სპეციალისტი“. მისი როლის შესაბამისი ფუნქცია ჩანს UseCase დიაგრამიდან (ნახ.1.6). იგი იყენებს ეკოლოგიური მაჩვენებლების სპეციალისტების მიერ შავი ზღვის საკონტროლო წერტილებში ჩატარებულ საველე-კვლევითი სამუშაოებისა და ქიმიური ლაბორატორიული გამოკვლევის შედეგებს, რომლებიც გადმოცემულია EMC-სისტემის MsSQL Server მონაცემთა ბაზაში და განთავსებულია შესაბამის ცხრილებში. ეკო-მონიტორინგის სპეციალისტის ინტერფეისის შესაქმნელად საჭიროა აიგოს მიმდევრობითობის დიაგრამა (Sequence-D2), რომლის შიგთავსი 1.20 ნახაზზეა მოცემული, ხოლო თვით დიაგრამა 1.21-ზე.



ნახ.1.19. UML Model Explorer-ში SequenceD1-ის შიგთავსი



ნახ.1.20. UML Model Explorer-ში SequenceD2-ის შიგთავსი



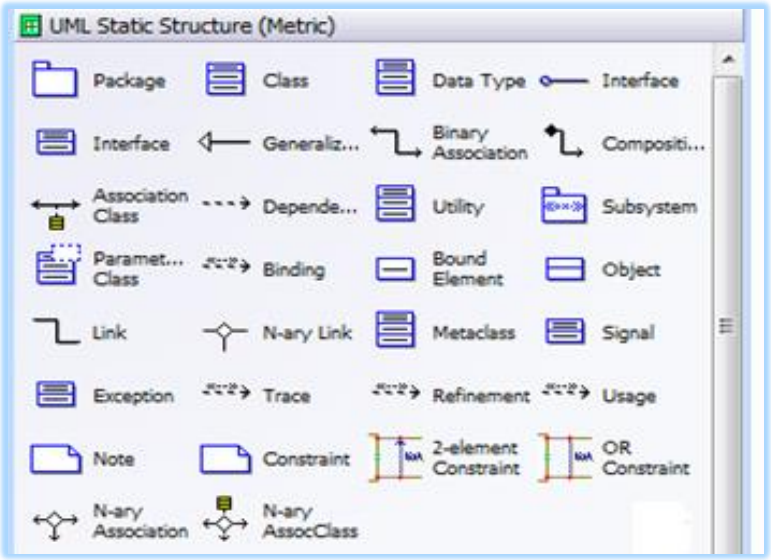
ნახ.21. UML SequenceD2 როლისათვის „ეკომონიტორინგის სპეციალისტი“, ბიზნეს-პროცესი: „ეკომაჩვენებლების ცხრილის ანალიზი და პროგნოზი“

ასეთივე მიდგომით შესაძლებელია ეკოსისტემის სხვა მომხმარებლების (ნახ.1.6) სამუშაო სცენარების აგება Sequence დიაგრამებით.

1.2.4. კლასების (Class) დიაგრამა

კლასი (Class) ერთგვაროვან ობიექტთა ერთობლიობის სტრუქტურაა. მაგალითად, ყველა თანამშრომელი, ეკოლოგიური მაჩვენებლების ყველა სპეციალისტი, მდინარის ესტუარები, ნავთობტერმინალები, ეკოლოგიური პარამეტრები და ა.შ.,

ტერმინი „კლასიფიკაცია“ სწორედ განსახილველი სფეროს ობიექტების სისტემატურ მოწესრიგებას ემსახურება (გენერალიზაცია (იერარქიაში განზოგადება ზევით) და სპეციფიკაცია (იერარქიაში დეტალიზაცია ქვევით). მოდელირების MsVisio პაკეტში კლასებისა და კლასთაშორის კავშირების მოდელირებისათვის გამოიყენება Static Structure ინსტრუმენტების პანელი (ნახ.1.22).

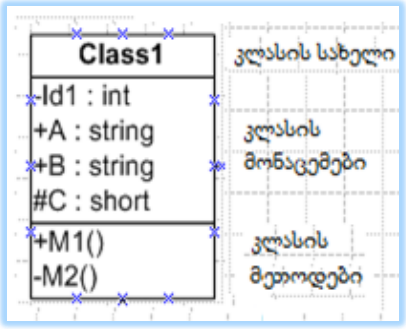


ნახ.1.22. ClassD-ის ინსტრუმენტების პანელი (MaVisio)

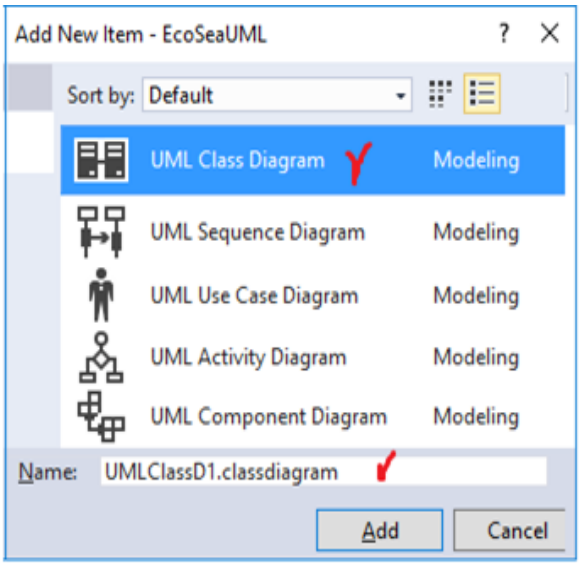
ობიექტ-ორიენტირებული მოდელირებისა და პროგრამირების გაგებით, კლასი არის „კლასის დასახელების“, „კლასის მონაცემებისა“ და „კლასის მეთოდების“ ინკაფსულაცია (ნახ.1.23).

კლასის ატრიბუტებს შეესაბამება მონაცემთა გარკვეული ტიპი (int, float, string ან სხვ.) და „ხილვადობის“ ანუ მონაცემთა წვდომის მოდიფიკატორები („-“ private, „+“ public, „#“ protection). ასევეა კლასის მეთოდებისთვისაც.

კლასის მეთოდები (ან ფუნქციები) ის პროგრამული მოდულებია, რომლებიც ამუშავებს ამ კლასის მონაცემებს. მათი ინიციალიზაცია ხდება გარედან შემოსულ შეტყობინებათა საფუძველზე.



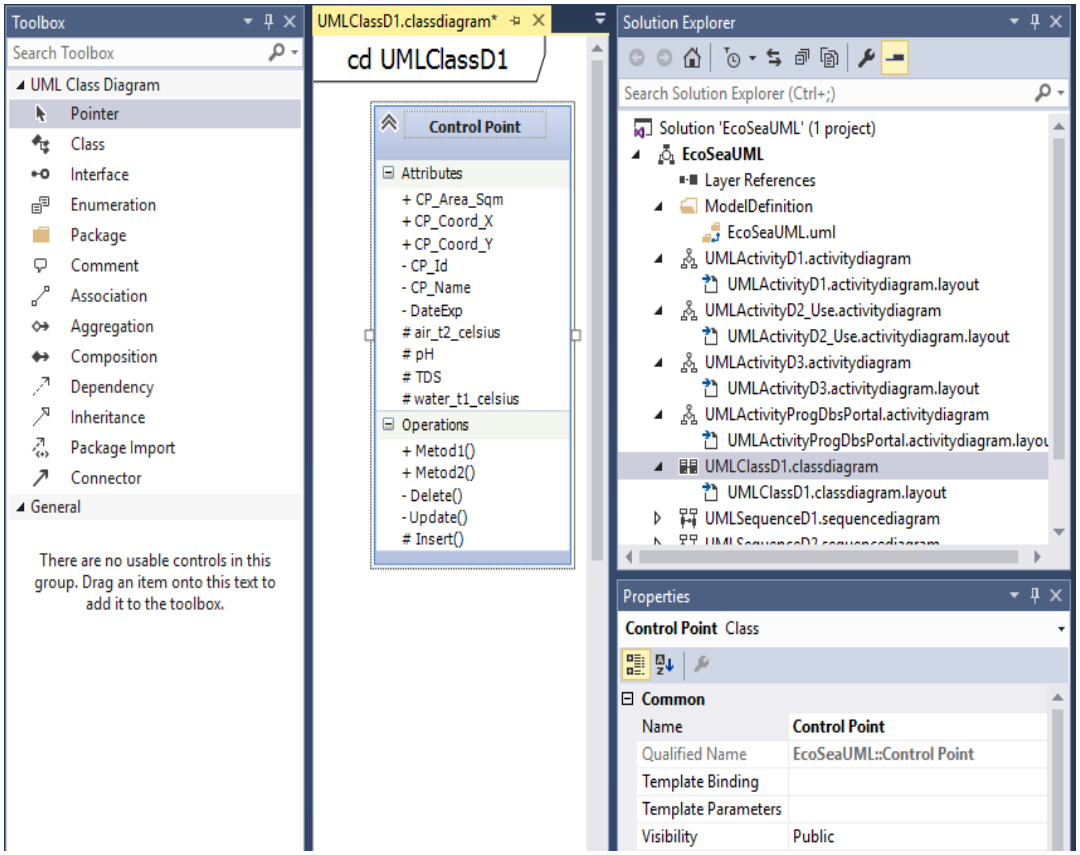
ნახ.1.23. ინკაფსულაცია



ახლა ავაგოთ კლასი „საკონტროლო წერტილი“ (Control poont) - Visual Studio.NET 2015 პლატფორმის გარემოში. Solution Explorer-ში EcoSeaUML პროექტზე მარჯვენა ღილაკით გავააქტიუროთ ახალი ელემენტის დამატების ფანჯარა (ნახ.1.24).

1.25 ნახაზზე ნაჩვენებია Control Point კლასის ინკაფსულირებული (სახელი, ატრიბუტები და ოპერატორები) ვერსია მათი ხილვადობის (წვდომადობის) თვისებებით (public, private, protection) (ნახ.1.26).

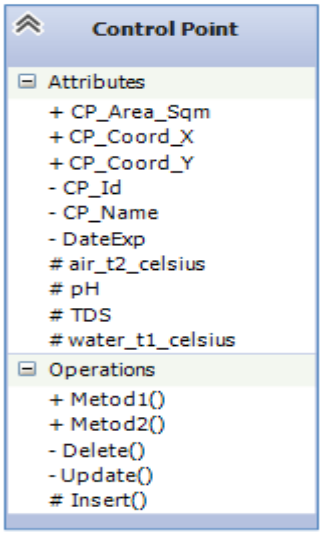
ნახ.1.24. UML Class Diagram -ის არჩევა



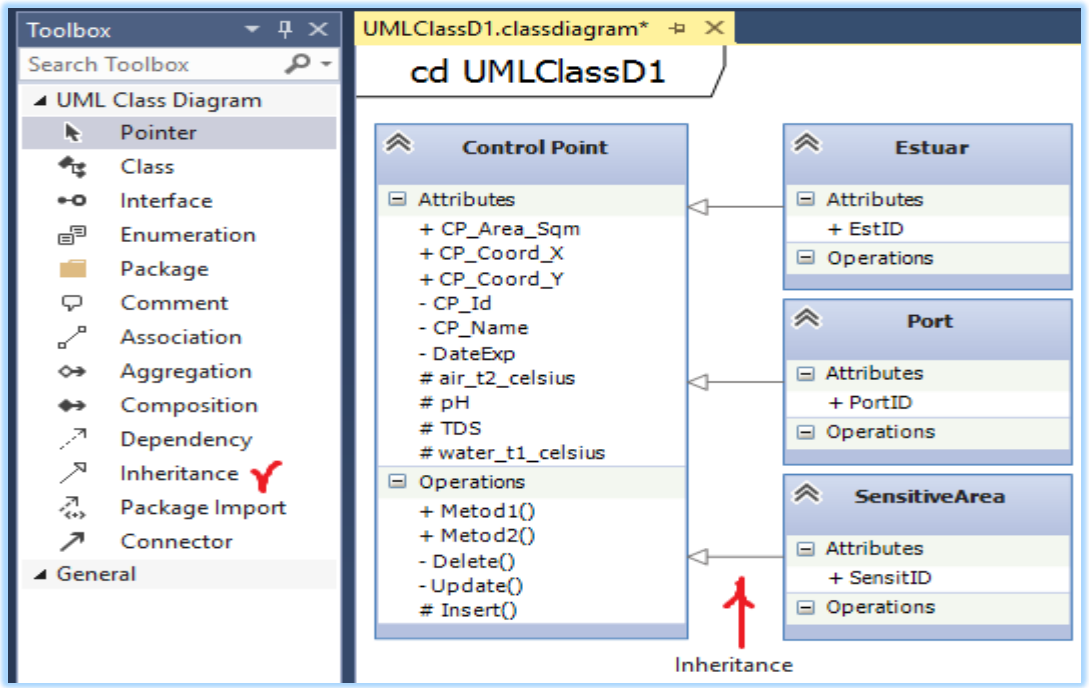
ნახ.1.25. კლასების დიაგრამის აგება Visual Studio.NET 2015 პლატფორმაზე

კლასი Control Point აბსტრაქტული კლასია და იგი კონკრეტულ ეგზემპლარებს არ შეიცავს. მას შეესაბამება კონკრეტული კლასები (ან ქვეკლასები subclasses), როგორიცაა, მაგალითად, მდინარის ესტუარი (RiverEstuar), პორტი (Port), სენსიტიური უბანი (SensitiveArea) და სხვ. (ნახ.1.27). მათ აქვს იდენტიფიკატორები და დანარჩენ ატრიბუტებს და მეთოდებს ღებულობენ, როგორც „შვილობილი“ ქვეკლასები, აბსტრაქტული „მშობელი“ კლასიდან (ControlPoint).

აქ მნიშვნელოვანია მათ შორის კავშირი „Inheritance“ (მემკვიდრეობითი კავშირი). დანარჩენ შესაძლო კავშირებს კლასებს შორის განვიხილავთ მოგვიანებით.



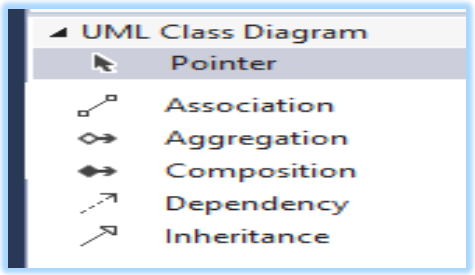
ნახ.1.26. კლასი „საკონტროლო პუნქტი“



ნახ.1.27. „მშობელი-შვილი“ კლასები მემკვიდრეობითობის კავშირთ (VS.NET2015)

კლასთაშორისი კავშირების განსაზღვრა მიეკუთვნება ობიექტ-ორიენტირებული დაპროექტების ეტაპს (ნახ.1.3) და მას მომდევნო პარაგრაფში დეტალურად განვიხილავთ.

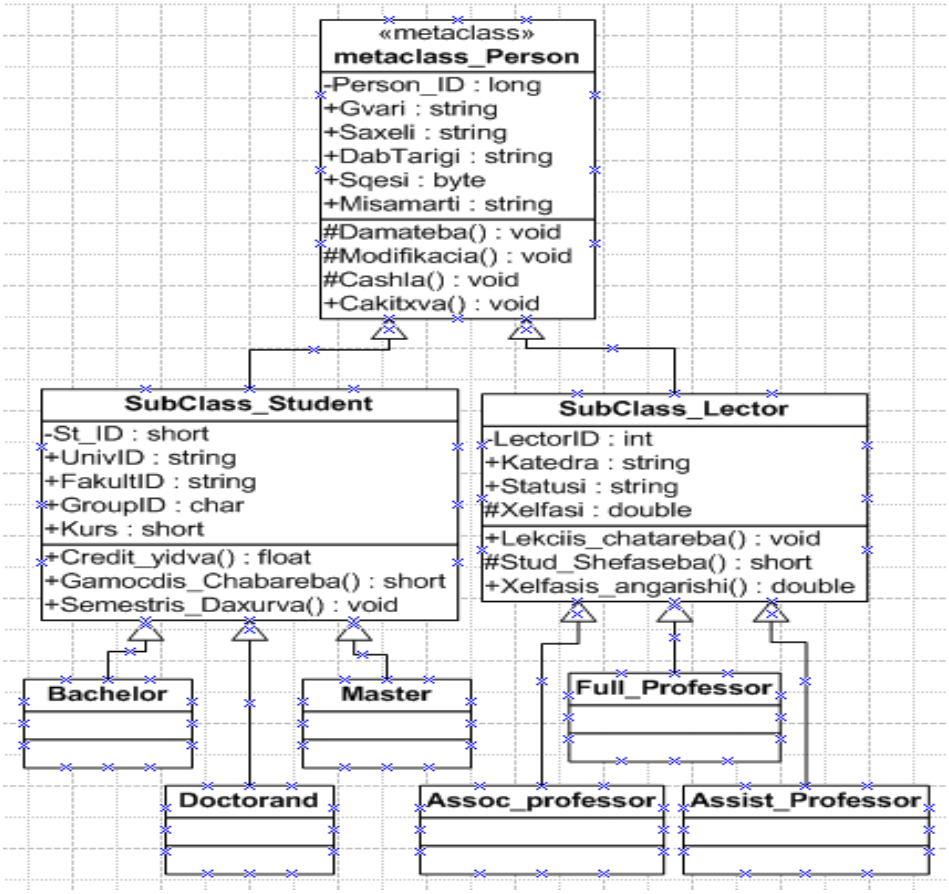
1.2.5. Class-Assotiation დიაგრამა



ზოგადად, კლასთაშორისი კავშირები შეიძლება იყოს: მემკვიდრეობითი, აგრეგატული, რელაციური და ასოციაციური (ნახ.1.28):

ნახ.1.28. კლასთაშორისი კავშირები

- ***ასოციაციური*** (Assotiation) ნიშნავს სემენტიკურ კავშირს კლასებს შორის. ის შეიძლება გამოისახოს ერთ- ან ორმიმართულებიანი (იგივეა, რაც უისრო) ხაზით. ისარი გვიჩვენებს შეტყობინების გადაცემის მიმართულებას. ასოციაციური კავშირის რეალიზება ხდება ერთ კლასში დამატებით მეორე კლასის ატრიბურტის ჩასმით. ეს ჰგავს პირველადი (PrimaryKey) და მეორეული (ForeignKey) გასაღებური ატრიბუტების შეერთებას;
- ***აგრეგირებული*** (Aggregation) არის ასოციაციურის კერძო შემთხვევა და ნიშნავს კავშირს „მთელი–ნაწილი“. მაგალითად, კლასი „ავტომობილი“ შედგება ქვეკლასებისგან: „ძარა“, „ძრავი“, „საბურავები“ და სხვ.“. ეს დეკომპოზიციაა. ქვეკლასი „ძრავი“ შეიძლება განვიხილოთ როგორც კლასი მისი შემადგენელი სხვა ნაწილებისგან და ა.შ. თუ კლასი „ავტომობილი“ წავშალეთ, მისი ქვეკლასები მაინც რჩება, არ იკარგება. აგრეგაციის ზედა კლასი არ გადასემს მის ქვედა (შემადგენელ) კლასებს „მემკვიდრულ“ თვისებებს და მეთოდებს;
- ***კომპოზიციური*** (Composition) არის აგრეგირებულის კერძო შემთხვევა, აქაც „მთელი-ნაწილის“ მიმართება გამოიყენება, ოღონდაც ეს ნაწილები მთელშია ჩადგმული და თუ მთავარი კლასი წაიშალა, ასევე წაიშლება მისი ეს ქვეკლასებიც. მაგალითად, „ეკომონიტორინგის პროგრამული სისტემა“ (.exe ფაილი) შედგება „მომხმარებელთა ინტერფეისების პროგრამების“, „ფუნქციური ამოცანების პროგრამების“ და „მონაცემთა დაცვის პროგრამებისაგან“ (რომლებიც კოდირებულია აპლიკაციის შიგნით). თუ წაიშალა (ან დაზიანდა) „ეკომონიტორინგის პროგრამული სისტემა“, მაშინ წაიშლება ასევე მისი ნაწილებიც;
- ***რელაციური*** (Dependency) ნიშნავს ერთი კლასის სემანტიკურ დამოკიდებულებას მეორეზე. პირველში ცვლილება მეორეშიც მოითხოვს შესაბამის ცვლილებას. იგი ერთ-მიმართულებიანი წყვეტილი ისრით გამოიხატება. მასში დამატებითი დამაკავშირებელი ატრიბუტები არ გამოიყენება;
- ***მემკვიდრეობითი*** (Inheritance ან Generalization) ასახავს „გენეტიკურ“, განზოგადო-ებულ კავშირებს კლასებს შორის. ასეთ დროს ერთი კლასი („შვილი“) მთლიანად იღებს მეორე კლასის („მშობელი“) ყველა ატრიბუტს, მეთოდს და კავშირს (ნახ.1.29).



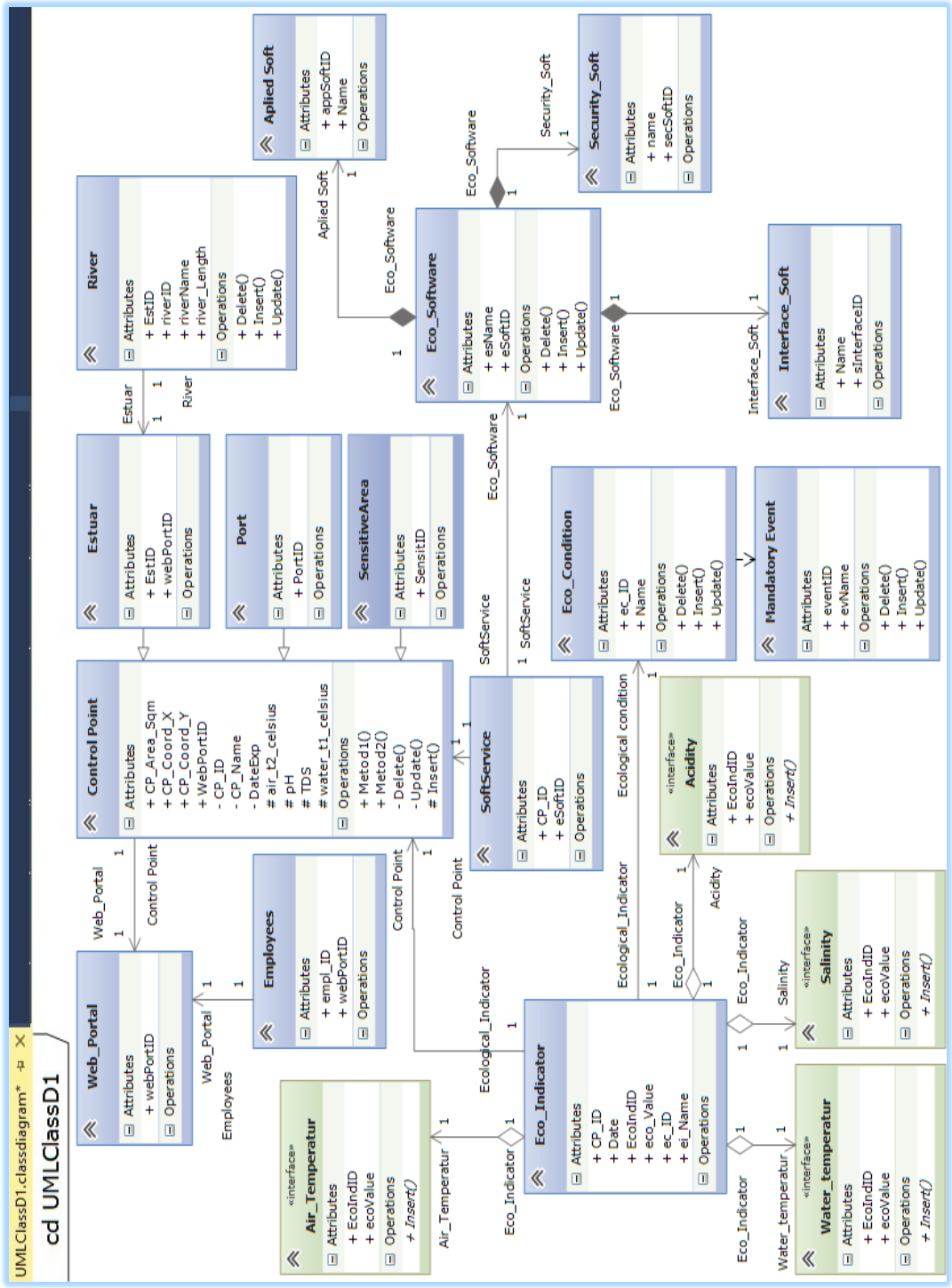
ნახ.1.4. Class-Assotiation დიაგრამა „მემკვიდრეობითი“ კავშირებით (MsVisio)

მშობელი კლასი ლიტარატურაში ზოგჯერ „მეტაკლასად“ (MetaClass) მოიხსენიება, რომელიც შედგება ქვეკლასებისაგან (SubClasses). შეიძლება იერარქიაში ქვეკლასი იყოს მის ქვევით მდგარი კლასისათვის მეტაკლასი. მაგალითად, SubClass_Student არის ქვეკლასი MetaClass_Person კლასისათვის და, ამავდროულად იგი არის მეტაკლასი სამი ქვეკლასისთვის: Bachelor, Master და Doctorand. იგივე შეიძლება ითქვას კლასებისათვის:

Metaclass_Person <- SubClass_Lector <- {Full_Professor, Assoc_Professor, Assist_Professor},

სადაც როლები ასეა განაწილებული: „მშობელი“-Person, „შვილი“-Lector და „შვილიშვილები“ Full_Professor, Assoc_Professor, Assist_Professor.

ახლა საილუსტრაციოდ განვიხილოთ ჩვენი, ეკომონიტორინგის სისტემის კლასთა-ასოციაციის დიაგრამა და მისი აგების შედეგები Visual Studio.NET 2015 პლატფორმაზე (ნახ.1.30).

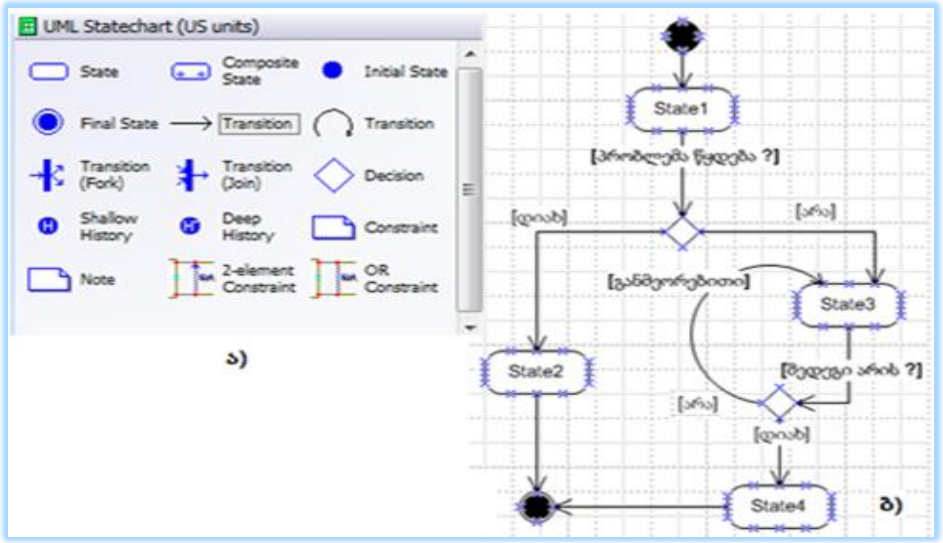


1.30. ეკომონიტორინგის სისტემის კლასთა-ასოციაციის დიაგრამა (VS.NET 2015): ასოციაციური, აგრგატული, კომპოზიციური, რელაციური და მემკვიდრეობითი კავშირებით

1.2.6. მდგომარეობათა (Statechart) დიაგრამა

ყოფაქცევის დიაგრამებიდან ადრე განვიხილეთ ქმედებათა (Activity) დიაგრამა და ინტერაქტიული (Sequence, Collaboration) დიაგრამები. არსებობს კიდევ ერთი ასეთი სახის დიაგრამა, *კლასების მდგომარეობათა* დიაგრამა - Statechart-D. იგი აღწერს ქმედებებს, ობიექტთა მდგომარეობებს, მდგომარეობათა გადასვლებს და მოვლენებს.

1.31-ა,ბ ნახაზებზე ნაჩვენებია MsVisio პაკეტის ინსტრუმენტული პანელისა და მდგომარეობათა დიაგრამის ფრაგმენტი ზოგადი მაგალითისთვის [13].



ნახ.1.31. Statechart-ის პანელი (ა) და ზოგადი დიაგრამა (ბ) (MsVisio)

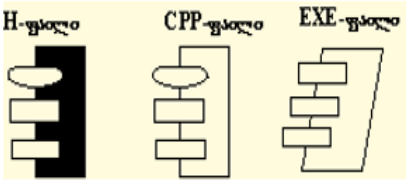
მისი გამოყენება ყველა კლასისათვის არაა საჭირო. აუცილებელია მხოლოდ მაშინ, როდესაც კლასი შეიძლება იმყოფებოდეს რამდენიმე მდგომარეობაში და თითოეულ მათგანში მისი ქცევა უნდა იყოს სხვადასხვანაირი.

მაგალითად, კლასების დიაგრამაზე (ნახ.1.30) სამი კლასი: ეკოლოგიური მაჩვენებელი (Ecological Indicator), ეკოლოგიური მდგომარეობა (Ecological Condition) და სავალდებულო ღონისძიება (Mandatory Event) ერთმანეთთან „მიზეზ-შედეგობრივი“ მიმართებითაა დამოკიდებული. თუ ეკოლოგიური მაჩვენებლები ნორმაშია, მაშინ ეკოსამსახურის უფროსი ადგენს რეპორტს ამ სიტუაციის შესაბამისად, თუ არსებობს რომელიმე მაჩვენებლის მნიშვნელობის ნორმებიდან გადახრა, მაშინ იგი ამზადებს სპეციალურ რეპორტს აუცილებელი ღონისძიებების გასატარებლად. ყველა შემთხვევაში ბიზნეს-პროცესები და ბიზნეს-წესები განსხვავებულია და მათთვის უნდა დაიწეროს შესაბამისი აქტიურობათა და მიმდევრობითობის დიაგრამები, რომელთა გათვალისწინებითაც პროგრამისტ-დეველოპერები დაწერენ შესაბამის პროგრამულ სერვისებს.

VS.NET 2015-ში მდგომარეობათა დიაგრამის სერვისი არაა წარმოდგენილი.

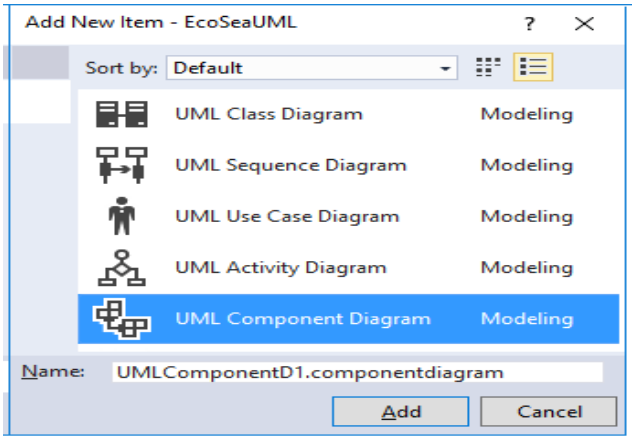
1.2.7. კომპონენტების (Components) დიაგრამა

პროგრამული რეალიზაციის ეტაპზე აიგება კომპონენტების დიაგრამა (Component-D), რომელშიც იგულისხმება პროგრამული კოდების (მაგალითად: .cs, .cpp, .h, .dll, .exe და ა.შ.) დამუშავება (ნახ.1.32).



ნახ.1.32. კომპონენტების დიაგრამის ზოგადი მაგალითი (MsVsio)

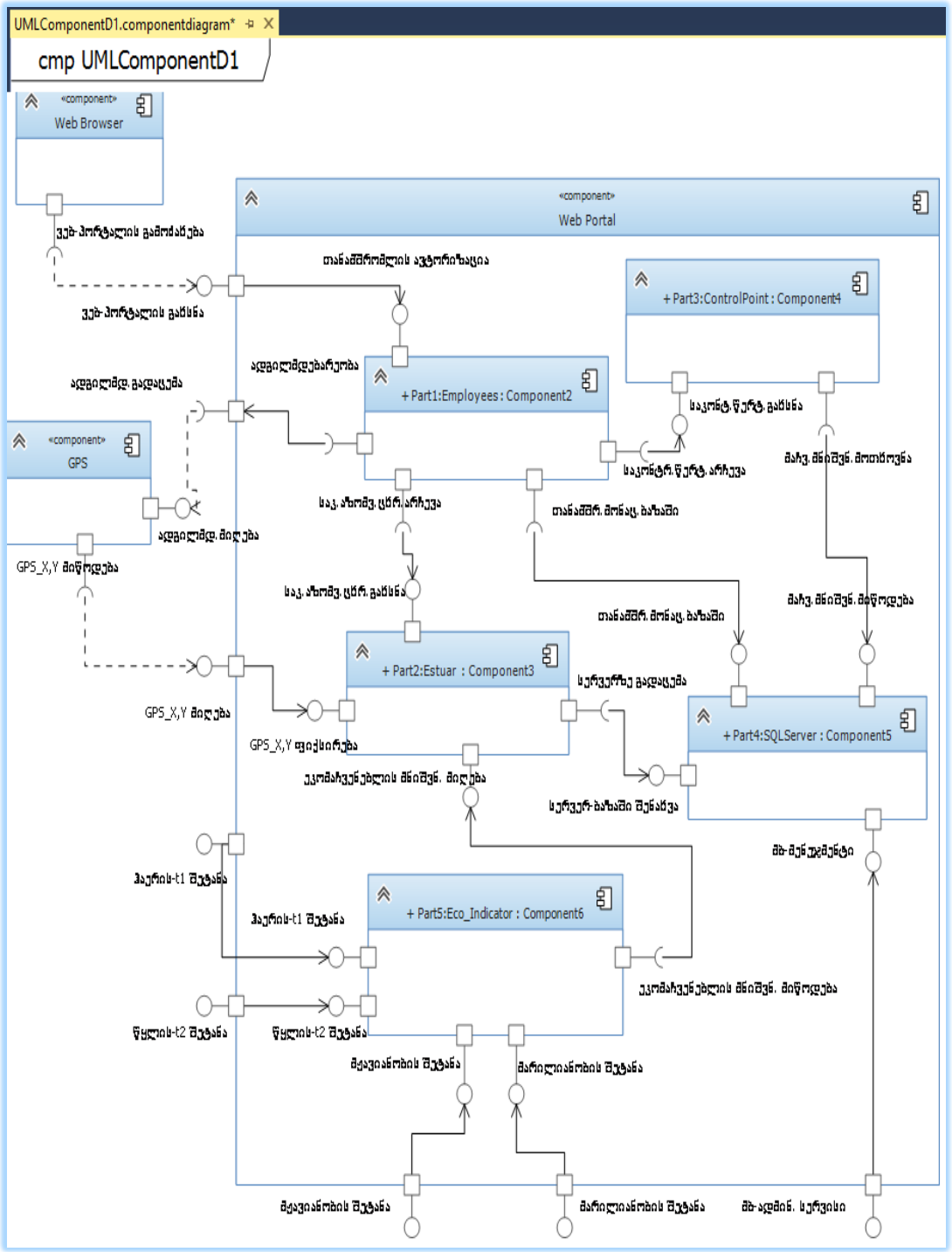
MsVisual Studio.NET 2015 პლატფორმაზე Solution Explorer ფანჯარაში EcoSeaUML პროექტს ვუმატებთ ახალ ელემენტს კომპონენტების დიაგრამის ასაგებად (ნახ.1.33).



ნახ.1.33. კომპონენტების დიაგრამის დამატება (VS.NET 2015)

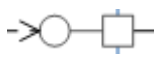
კომპონენტი არის მოდულური ერთეული. მისი შიგთავსი დაფარულია, მაგრამ მას აქვს ერთი ან მეტი კარგად განსაზღვრული *მიმწოდებელი ინტერფეისი* (Provided interfaces), რომელთა მეშვეობითაც მისი ფუნქციები ხელმისაწვდომია სხვა კომპონენტებისთვის (ცხრ.1.1). კომპონენტს ასევე შეუძლია ჰქონდეს *მოთხოვნების ინტერფეისები* (Required interfaces). ეს ინტერფეისი განსაზღვრავს თუ რა ფუნქციებს ან მომსახურებას მოითხოვს იგი სხვა კომპონენტებისგან. რამდენიმე კომპონენტის მიმწოდებელი და მოთხოვნების ინტერფეისების დაკავშირებით შესაძლებელია უფრო დიდი კომპონენტის შექმნა. სრული პროგრამული უზრუნველყოფის სისტემა შეიძლება გაგებული იყოს როგორც კომპონენტი.

1.34 ნახაზზე ნაჩვენებია ეკომონიტორინგის სისტემის კომპონენტების დიაგრამის ფრაგმენტი Visual Studio.NET 2015 პლატფორმაზე.



ნახ.1.34. EMS-ის Component დიაგრამა (VS.NET 2015)

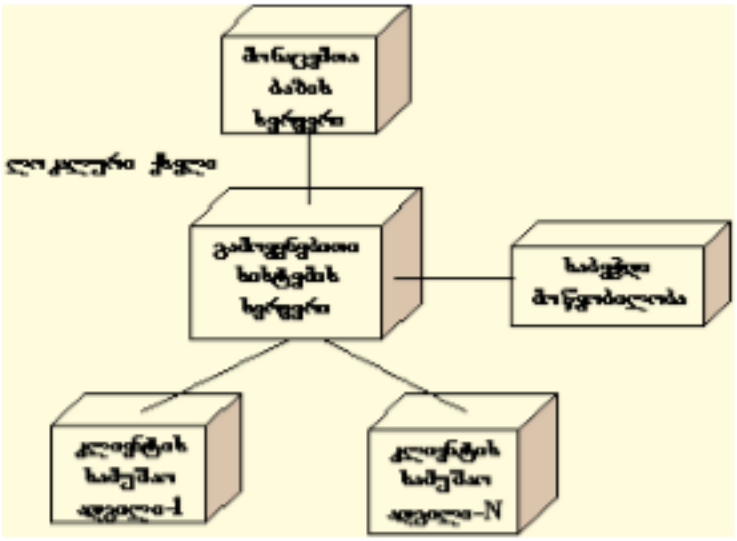
პორტის სახეები კომპონენტების დიაგრამაზე. ცხრ.1.1

პორტი	დანიშნულება
	*მიმწოდებელი ინტერფეისი* (Provided Interface) უზრუნველყოფს კომპონენტში განხორციელებულ ოპერაციებს და შეიძლება გამოყენებულ იქნას სხვა კომპონენტების მიერ. მაგალითად, ინტერფეისი ვებ მომსახურება .NET ინტერფეისი, ან ფუნქციების კოლექცია პროგრამირების ნებისმიერ ენაზე.
	*მოთხოვნების ინტერფეისი* (Required Interface) არის კომპონენტის მოთხოვნა (საჭირო ოპერაციების ჯგუფი), რომელიც უნდა უზრუნველყოფდეს სხვა კომპონენტებს ან გარე სისტემებს. მაგალითად, ვებ ბრაუზერი მოითხოვს ვებ სერვერებს ან აპლიკაციის დანამატი (ძირითად აპლიკაციაში დასამატებელი პროგრამული უტილიტა, ზედნაშენი) მოითხოვს სერვის-პროგრამებს აპლიკაციისგან.

კომპონენტს შეიძლება ჰქონდეს პორტების შეუზღუდავი რაოდენობა.

1.2.8. განთავსების (Deployment) დიაგრამა

განაწილებული სისტემის რეალიზაციის ბოლო ეტაპზე აიგება განთავსების დიაგრამა (Deployment-D), რომელიც აღწერს კომპონენტების განაწილებას „კლიენტ-სერვერის" ქსელში (ნახ.1.35) [13].



ნახ.1.35. განთავსების დიაგრამა (MsVisio)

VS.NET 2015-ში განთავსების დიაგრამის აგების სერვისი არაა წარმოდგენილი.

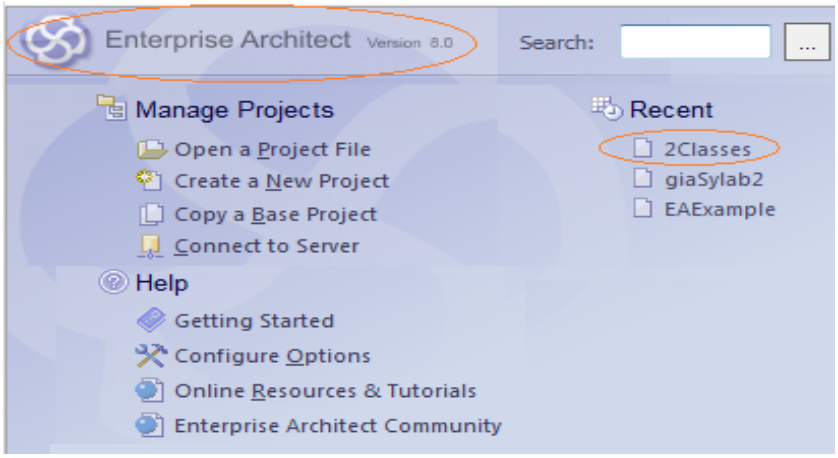
1.3. კლასების დიაგრამიდან პროგრამული კოდის გენერაცია

CASE (Computer-Aided Software Engineering) ტექნოლოგიები, რომლებიც სისტემების დაპროგრამების ავტომატიზაციაზეა ორიენტირებული, მაგალითად, Rational Rose, Visual Paradigm, Enterprise Architect და სხვ. [10,23], აგრეთვე Visual Sstudio .NET Framework, ახორციელებს რევერსული დაპროგრამების კონცეფციას. ანუ კლასების დიაგრამიდან შესაძლებელია პროგრამული კოდის გენერაცია და პირიქითაც, კოდიდან ავტომატურად აიგება გრაფიკული დიაგრამა [10,13].

MsVisio არ მიეკუთვნება ასეთი სიმძლავრის ინსტრუმენტს. მისი საშუალებით დიალოგურ რეჟიმში იხაზება დიაგრამები (UML-ის სტანდარტულ აღნიშვნათა საერთაშორისო ნორმებით), მაგრამ კოდის გენერაცია არაა შესაძლებელი.

Ms Visual Studio .NET Framework- ისთვის შექმნილია ინსტრუმენტები და მათი ინტეგრაციით .NET გარემოში, შესაძლებელია დიაგრამებიდან კოდის გენერაცია. ასეთი პაკეტები ყოველთვის ფასიანია და ძვირადღირებული.

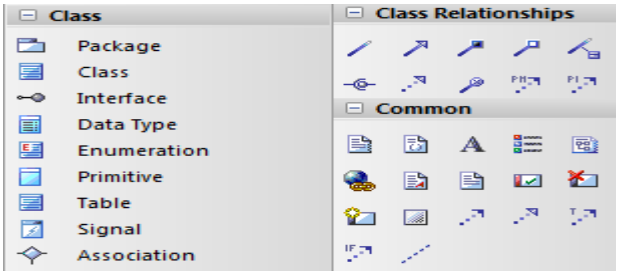
ახლა (შედარების მიზნით) განვიხილოთ SparX ფირმის საკმაოდ მძლავრი ინსტრუმენტი (თავსებადი VS.NET-პაკეტთან) - Enterprise Architect პროდუქტი, კერძოდ, კლასების დიაგრამიდან კოდის გენერაციის ამოცანა. 1.36 ნახაზზე ნაჩვენებია პაკეტის ამუშავების საწყისი გვერდი.



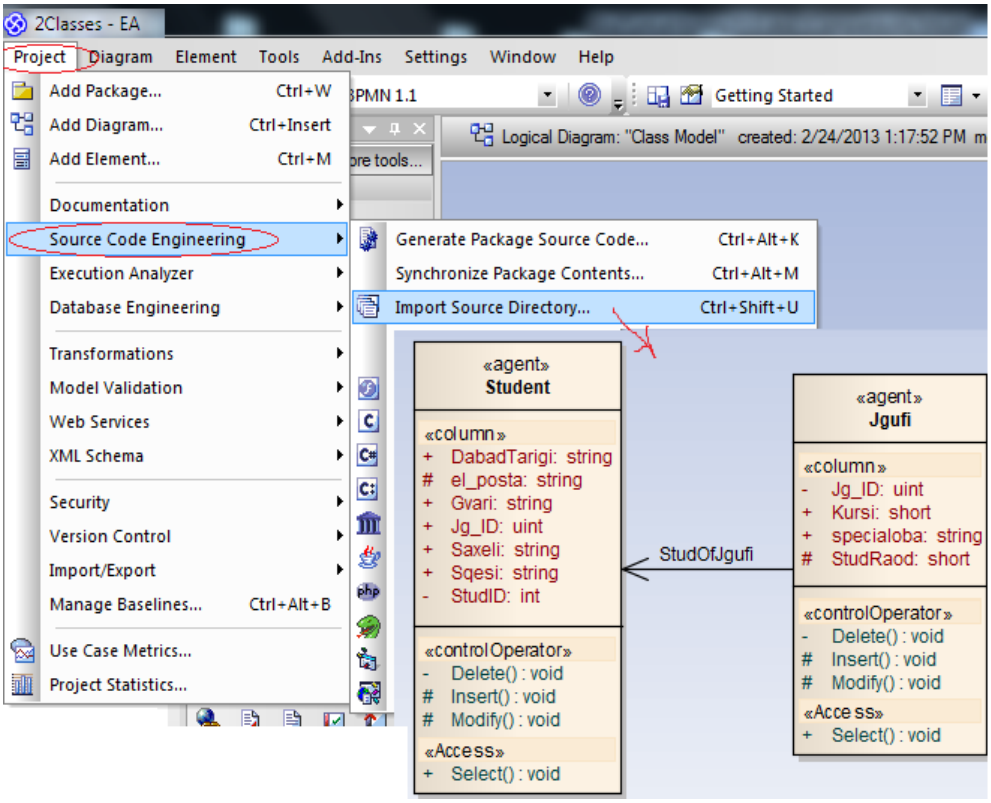
ნახ.1.36. Enterprose Architect საწყისი გვერდი

მაგალითისთვის ვიხილავთ ამ გარემოში ორი კლასის (2Clases მოდელი) აგების და მათი პროგრამულ კოდში გადაყვანის ამოცანას. 1.37 ნახაზზე მოცემულია Enterprise Architect პაკეტის კლასთა დიაგრამის აგების ინსტრუმენტების პანელი. ვირჩევთ (ნახ.1.38):

Report->Source Code Engineering->Import Source Directory



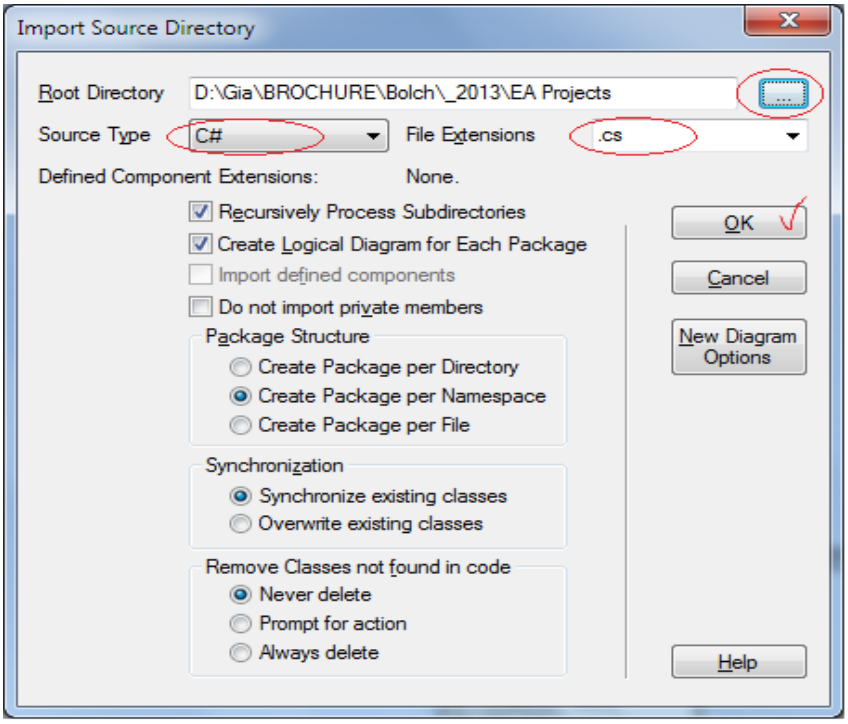
ნახ.1.37. ინსტრუმენტების პანელი



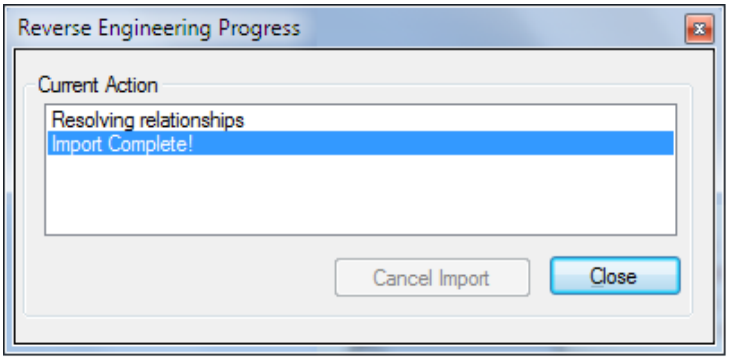
ნახ.1.38. Student და Jgufi კლასების მომზადება „Code Engineering“ პროცესისათვის Enterprise Architect გარემოში

არჩევის შემდეგ გამოვა 1.39 ნახაზზე ნაჩვენები ფანჯარა, რომელშიც უნდა განისაზღვროს ზოგიერთი მნიშვნელოვანი პარამეტრი, მაგალითად, ენა (C#), მომავალი კოდის შესანახი ადგილი (დირექტორია) და ა.შ.

ბოლოს „Ok“ და მივიღებთ 1.40 ნახაზზე მოცემულ შედეგს.

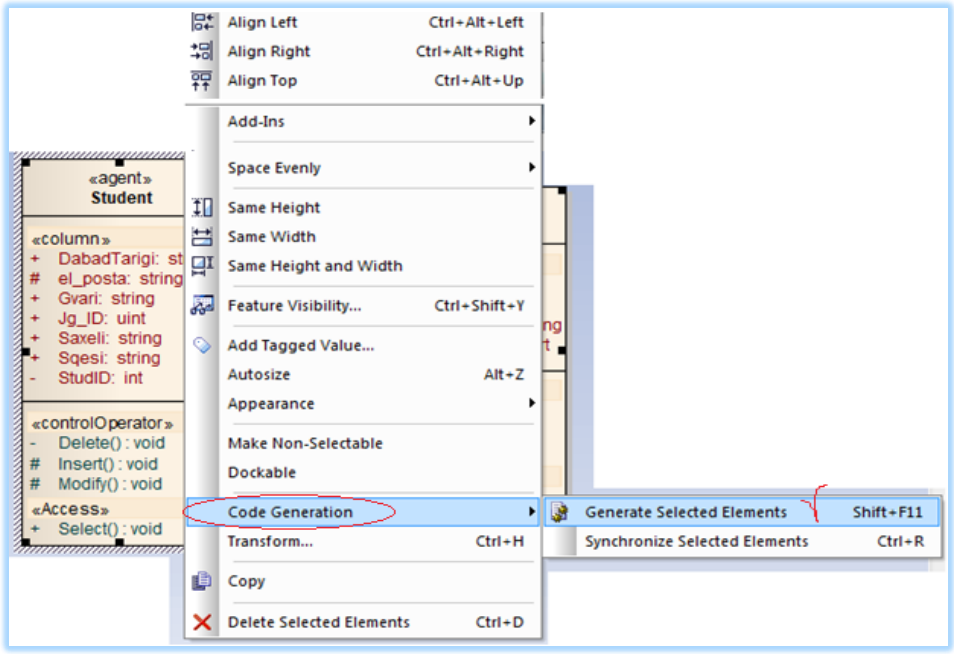


ნახ.1.39. განისაზღვროს C#-კოდის დირექტორია

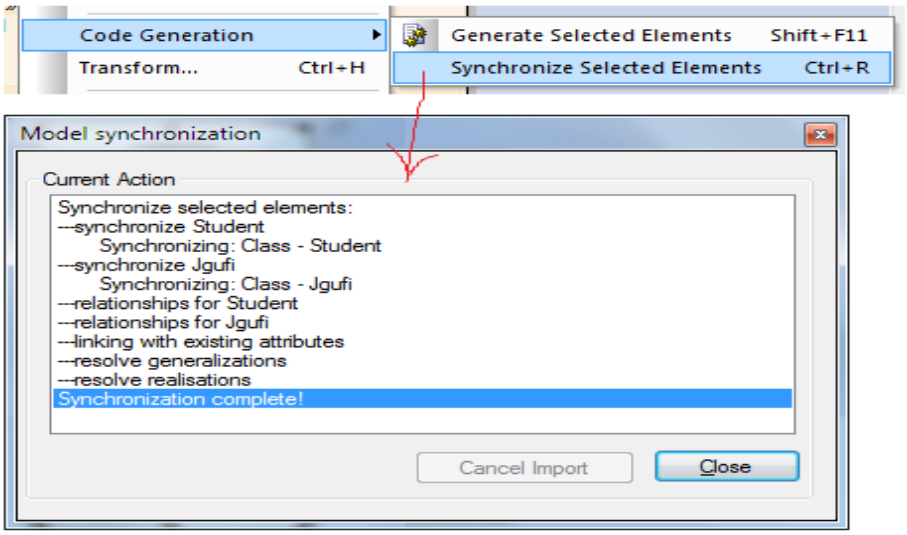


ნახ.1.40. იმპორტი დასრულებულია

ახლა უნდა ჩატარდეს უშუალოდ კოდის გენერაცია წინასწარ მომზადებული (Student-Jgufi) კლასების დიაგრამიდან. ვააქტიურებთ კლასებს და მაუსის მარჯვენა ღილაკით გამოტანილ კონტექსტური მენიუდან ვირჩევთ „Code Generation“-ს (ნახ.1.41-42).

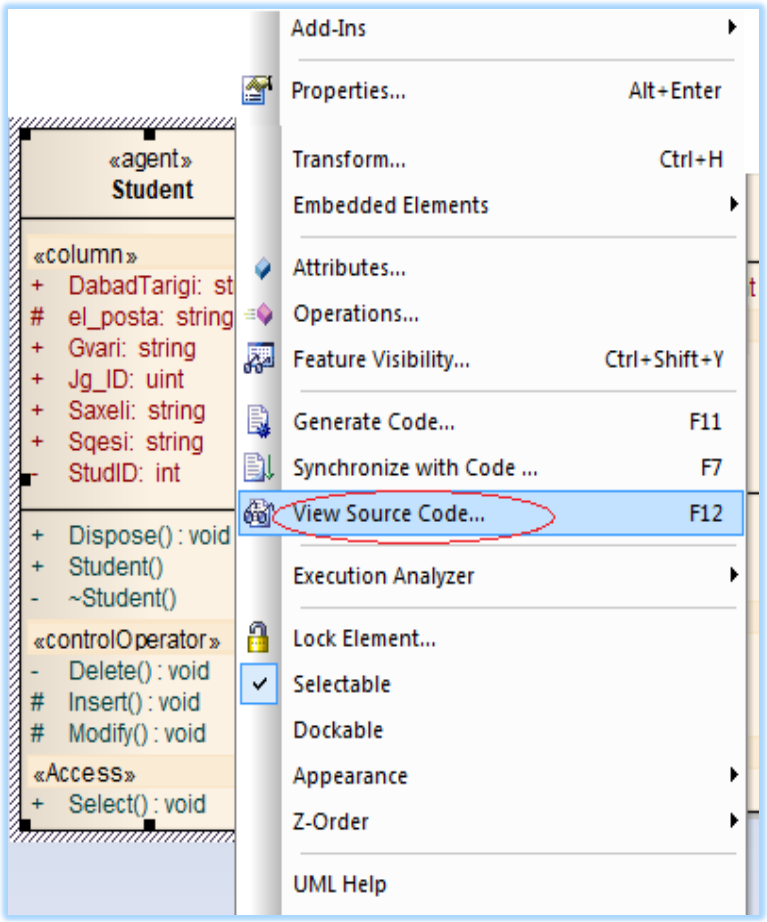


ნახ.1.41. კოდის გენერაციის დაწყება



ნახ.1.42. არჩეული ელემენტის სინქრონიზაციის შედეგი

ბოლო ფაზაზე საჭიროა ეკრანზე გამოვიტანოთ კლასების ბაზაზე გენერირებული კოდის ლისტინგები (ნახ.1.43).

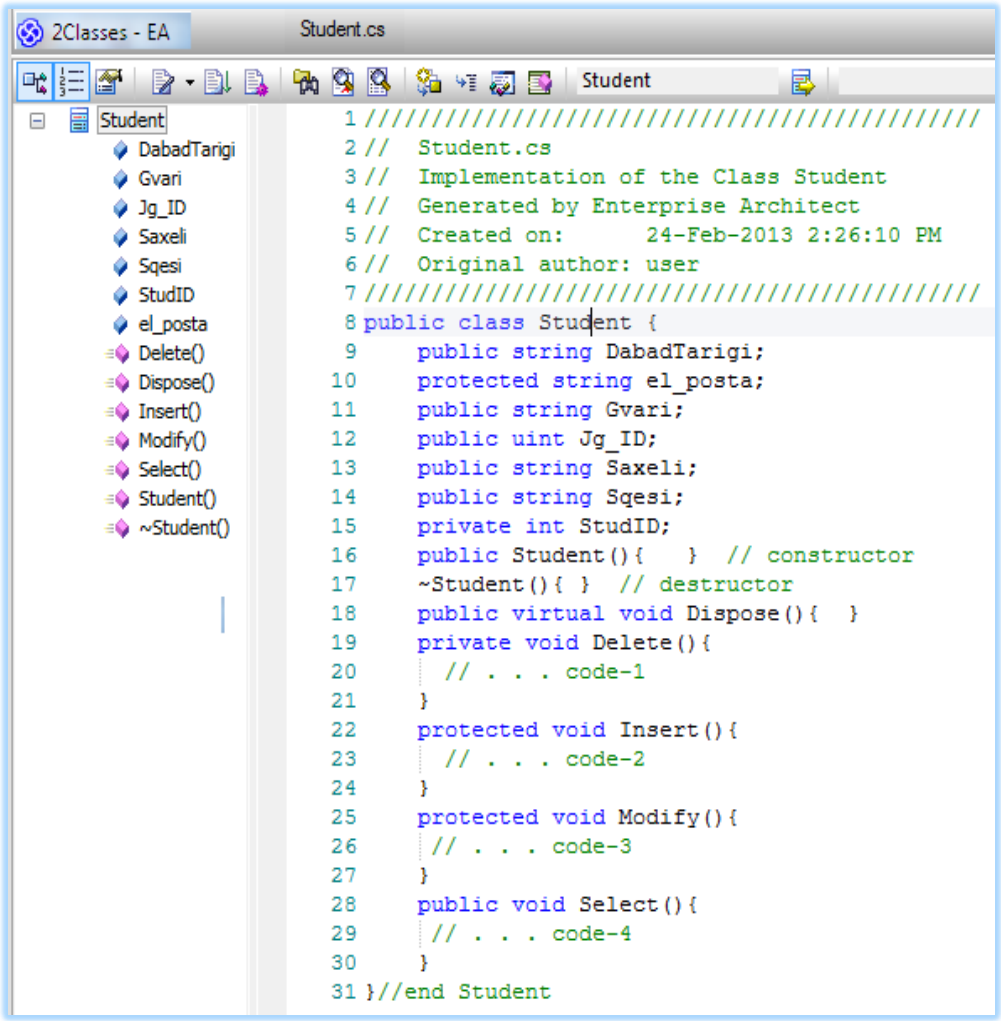


ნახ.1.43. კოდის გამოტანის პუნქტი მენიუში

1.44 ნახაზზე ნაჩვენებია Enterprise Architect გარემოში Student კლასის დიაგრამიდან ავტომატურად გენერირებული C#-კოდის საწყისი ტექსტი.

ფანჯრის მარცხენა ნაწილში მოთავსებულია Student კლასის კაფსულა, თავისი მონაცემებით და მეთოდებით, მათ შორის კონსტრუქტორით და დესტრუქტორით (მეთოდები, რომელთაც აქვს კლასის იდენტური სახელი).

ფანჯრის მარჯვენა ნაწილში სისტემას გამოაქვს პროგრამის ტექსტი, რომელიც შედგება კომენტარული ნაწილის (სტრიქონები 1-7) და კლასის აღწერის ნაწილისაგან (8-31).



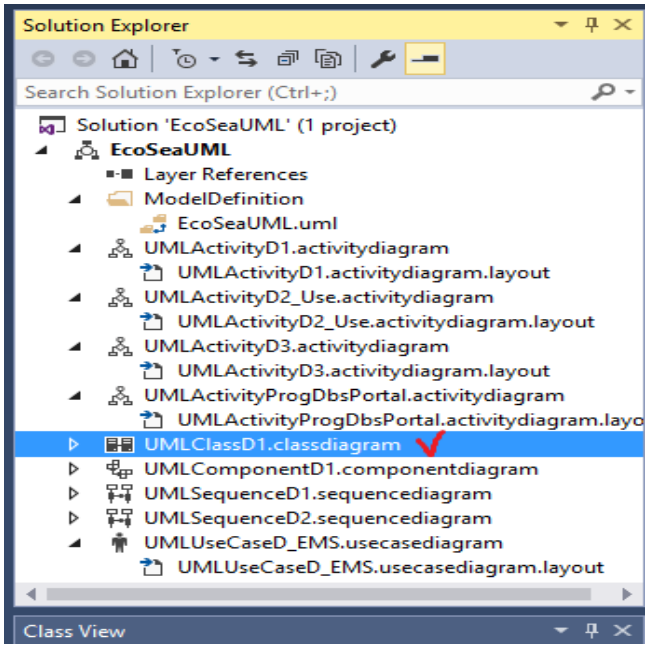
ნახ.1.44. C#-კოდის ლისტინგი Student კლასისათვის

მომდევნო ლისტინგში მოცემულია Jgufi კლასის საწყისი ტექსტი. აქაც, C# კოდის ტექსტი შედგება კომენტარული ნაწილისაგან, რომელშიც ასახულია პროგრამის ზოგადი მახასიათებლები, სახელი, ინსტრუმენტი, შექმნის თარიღი, ავტორი. პროგრამის ტექსტი კლასიკური ფორმატით აღიწერება კლასის მონაცემები ხილვადობის private, public და protection ატრიბუტებით. შემდეგ მოსდევს კონსტრუქტორის public Jgufi(){ } და დესტრუქტორის ~Jgufi(){ } სტრიქონები. Dispose() მეთოდი გამოიყენება პროგრამის შესრულების დამთავრების შემდეგ ოპერაციული სისტემის მიერ გამოყოფილი რესურსების გასათავისუფლებლად.

```
////////////////// ლისტინგი ///////////////////////
//  Jgufi.cs
//  Implementation of the Class Jgufi
//  Generated by Enterprise Architect
//  Created on:    24-Feb-2016 2:26:15 PM
//  Original author: user
public class Jgufi {
        private uint Jg_ID;
        public short Kursi;
        public string specialoba;
        protected short StudRaod;
        public Student m_Student;
        public Jgufi() {  }     // კონსტრუქტორი
        ~Jgufi() {  }    // დესტრუქტორი
        public virtual void Dispose() {  }
        private void Delete(){
        // . . . code-1
        }
        protected void Insert(){
        // . . . code-2
        }
        protected void Modify(){
       // . . . code-3
        }
        public void Select(){
       // . . . code-4
        }
}//end Jgufi
```

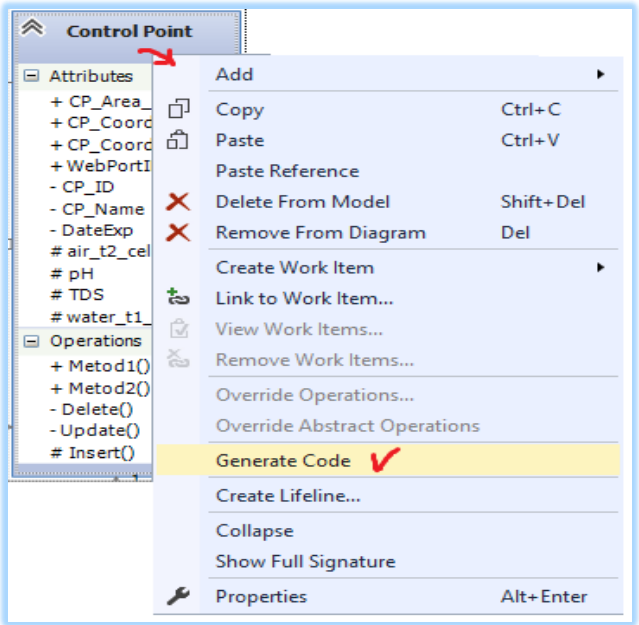
ამჯერად გადავიდეთ ეკოლოგიური მონიტორინგის სისტემის კლასთა ასოციაციის დიაგრამის საფუძველზე (ნახ.1.30) C# კოდის ავტომატური გენერაციის მაგალითის განხილვაზე. აღნიშნული კლასთა ასოციაციის დიაგრამა, ზემოგანხილულთან შედარებით, საკმაოდ რთულია, შედგება ორზე მეტი კლასისა და კლასთა ტიპისაგან (ტიპი: კლასი, ინტერფეისი).

Solution Explorer ფანჯარაში EcoSeaUML პროექტისთვის საჭიროა ავირჩიოთ UMLClassDa.classdiagram პუნქტი (ნახ.1.45). მუშა ფანჯარაში გამოჩნდება შესაბამისი კლასები არსებული კავშირებით (იხ. ნახ. 1.30).



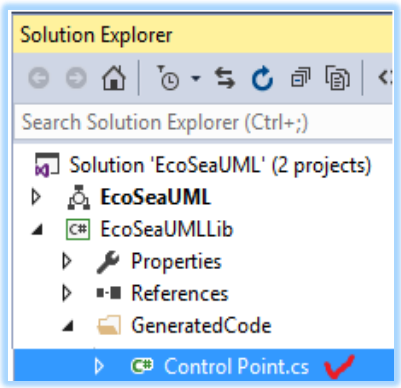
ნახ.1.45

კლასების დიაგრამაზე ვურჩევთ ერთ კლასს (ან რამდენიმეს, ან ყველას) და მაუსის მარჩვენა ღილაკით მასზე ვხსნით კონტექსტურ მენიუს (ნახ.1.46).



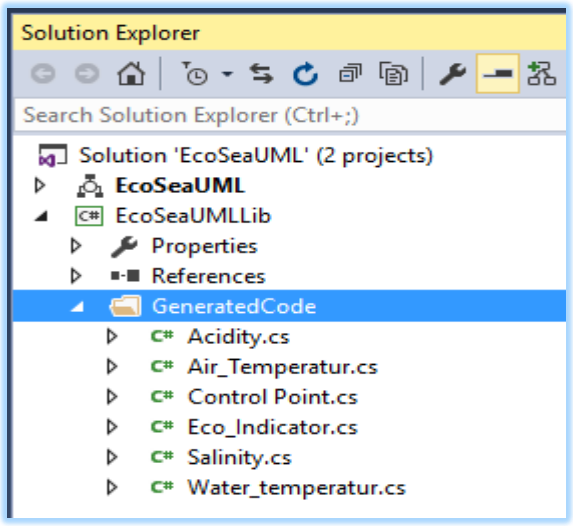
ნახ.1.46. Generate Code - ფუნქციის არჩევა

კოდის გენერაციის შედეგად Solution Explorer ფანჯარაში გამოჩნდება C# Control Point.cs სტრიქონი (ნახ.1.27).



ნახ.1.47. C# კოდი

დავამატოთ კიდევ რამდენიმე კლასი, მაგალითად ინტერფეისის სტერეოტიპით. როგორებიცაა ჰაერის და წყლის ტემპერატურების, მჟავიანობისა და მარილიანობის მაჩვენებელთა მნიშვნელობების შეტანის ინტერფეისები. მათთვისაც ჩავატაროთ კოდის გენერაციის პროცედურა. მივიღებთ შემდეგ სურათს (ნახ.1.48).



ნახ.1.48. 6 გენერირებული C# კოდი

Control Points.cs კოდის ტექსტი ასე გამოიყურება (ლისტინგი_1.1).

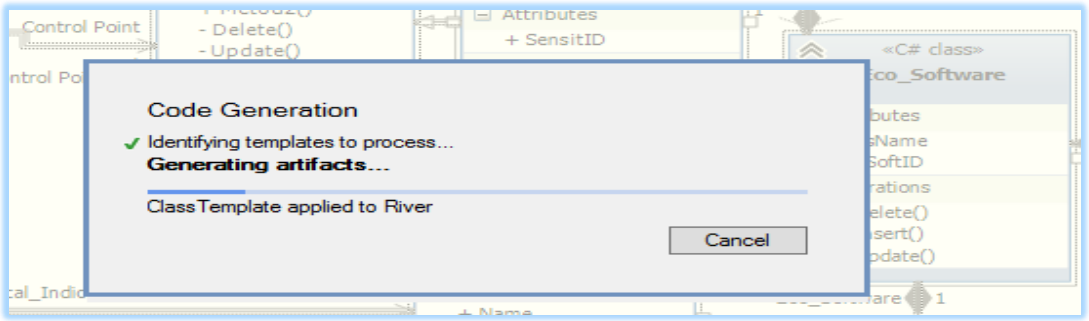
```
//--- ლისტინგი_1.1 ------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool
//     Changes to this file will be lost if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Control Point
{
       private object CP_ID { get; set;
        }
       private object CP_Name {get; set;
        }
       public virtual object CP_Coord_X { get; set;
        }
       public virtual object CP_Coord_Y { get; set; }
      public virtual object CP_Area_Sqm {get; set;
        }
      protected virtual object water_t1_celsius {get; set; }
```

```
        protected virtual object air_t2_celsius { get; set;
         }
        protected virtual object pH { get; set;
         }
        protected virtual object TDS{ get; set;
         }
        private object DateExp{ get; set;
         }
        public virtual object WebPortID{ get; set;
         }
        public virtual Web_Portal Web_Portal{ get; set;
         }
        public virtual void Metod1() {
                throw new System.NotImplementedException();
         }
        public virtual void Metod2() {
                throw new System.NotImplementedException();
         }
        private void Update() {
                throw new System.NotImplementedException();
         }
        private void Delete() {
                throw new System.NotImplementedException();
         }
        protected virtual void Insert(){
                throw new System.NotImplementedException();
         }
       }
```

1.2_ლისტინგში მოცემულია წყლის ტემპერატურის მაჩვენებლის მნიშვნელობის შეტანის ინტერფეისის ავტომატურად გენერირებული კოდის ტექსტი.

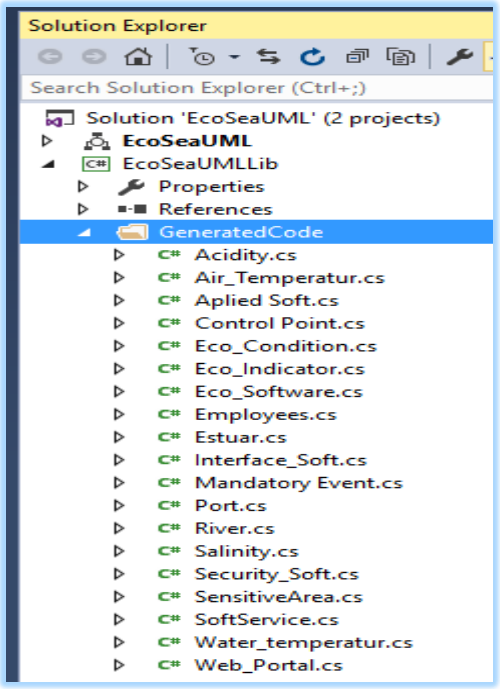
```
//--- ლისტინგი_1.2 ------------------------------------------------
// <auto-generated>
//     This code was generated by a tool
//     Changes to this file will be lost if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public interface Water_temperatur
{
    object ecoValue { get;set; }
    object EcoIndID { get;set; }
    void Insert();
}
```

ჩვენი კლასების დიაგრამის (ნახ.1.30) მთლიანი სქემის შესაბამისი კოდის გენერაციისათვის საჭიროა მოვნიშნოთ ყველა კლასი და ინტერფეისი (მაგ., Ctrl-A) და კონტექსტური მენიუდან ავირჩიოთ generate Code. გამოჩნდება შუალედური პროცესის მიმდინარეობის საინფორმაციო ფანჯარა (ნახ.1.49).



ნახ.1.49. ყველა კლასის C# კოდი გენერირდება მიმდევრობით

პროცესის დასრულების შემდეგ Solution Explorer შედგება 19 კლასის შესაბამისი C# კოდის ტექსტისაგან (ნახ.1.50).



ნახ.1.50. მთლიანი კოდის გენერაციის დასარული

ეკომონიტორინგის სისტემის ერთ-ერთი მნიშვნელოვანი კომპონენტებია შავ ზღვაში ჩამდინარე მდინარეები (საქართველოს საზღვრებში და მათი ესტუარები (ზღვასთან

შეერთების განსაზღვრული ფართობი). ქვემოთ ნაჩვენებია მათი შესაბამისი კლასების C# კოდები (ლისტინგი 1.3 და 1.4).

```
//-- ლისტინგი_1.3. --- მდინარეები ----------------------------------
// <auto-generated>
//------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public class River
{
     public virtual object riverID {get; set;
       }
     public virtual object river_Length { get; set;
       }
     public virtual object riverName { get; set;
       }
     public virtual object EstID { get; set;
       }
     public virtual Estuar Estuar { get; set;
       }
     public virtual void Insert() {
               throw new System.NotImplementedException();
       }
     public virtual void Update() {
               throw new System.NotImplementedException();
       }
     public virtual void Delete() {
               throw new System.NotImplementedException();
       }
}

//--- ლისტინგი_1.4 ---ესტუარები ----------------------------------
// <auto-generated>
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public class Control_Point
{
     public virtual object EstID { get; set;
       }
     public virtual object webPortID {get; set;
       }
  }
```

ამით ვამთავრებთ უნიფიცირებული მოდელირების ენის და მისი ინსტრუმენტების მოკლე აღწერას ეკომონიტორინგის სისტემის ოო-მოდელირების მიზნით.

II თავი

მულტიმედიური მონაცემთა ბაზები შავი ზღვის ეკომონიტორინგის სისტემისათვის

2.1. მონაცემთა მოდელები მულტიმედიური სისტემებისათვის

მონაცემთა მულტიმედიური ბაზების მართვის სისტემების (MMDBMS - MultiMedia DataBase Management Systems) ძირითადი ცნებების განვითარება უნდა დაიწყოს მონაცემთა მოდელების დისკუსიით [31]. მონაცემთა მოდელი არის ერთგვარი სახე (ფორმალური) ენისა, რომელშიც მომხმარებელს შეუძლია სინამდვილის ფრაგმენტების აღწერა. მონაცემთა მოდელს ასევე მიეკუთვნება ოპერაციები (მეთოდები) მონაცემთა: წარმოების, წაკითხვის, მიერთების (ლინკის), მოწესრიგების (დალაგების), ცვლილების, ძებნის და წაშლის შესახებ. მონაცემთა მოდელი განსაზღვრავს მომხმარებლის თვალსაზრისს მონაცემებზე. აქ მისი რეალიზაცია დაფარულია და მონაცემთა ბაზების მართვის სისტემის მიერ შესაძლებელია მისი განხორციელება სხვადასხვა გზით [10].

დღეისათვის, ტექნიკისა და ტექნოლოგიების განვითარების მოცემულ დონეზე, პრაქტიკულად ფართოდ გამოიყენება ე. კოდის რელაციური მოდელი, იერარქიულ და ქსელურ კლასიკურ მოდელებთან შედარებით [32-35].

ამ პერიოდში შემოთავაზებულ იქნა აგრეთვე სხვა, ახალი სახის მოდელებიც, რომლებსაც ექსპერიმენტულადაც იკვლევდნენ მწარმოებლურობის ამაღლების თვალსაზრისით, რელაციურთან შედარებით. ასეთი მოდელების მაგალითებია სემანტიკური, ფუნქციონალური და ობიექტ ორიენტირებული [36,37]. მაგრამ მათი რეალიზაცია დაკავშირებული იყო დიდ დანახარჯებთან, რაც რელაციური მოდელის უპირატესობას ამტკიცებდა, მისი სიმარტივის გამო. ამ თვალსაზრისით უკეთესი კონცეფცია გახდა ისევ რელაციური მოდელის გაფართოებით მიღებული შედეგები, როგორიც იყო „ობიექტრელაციური“ მონაცემთა ბაზის მართვის სისტემები (ორ-მბმს) [31]. ისინი, თავიანთ შესაძლებლობათა გამო, მისაღები გახდა ასევე მულტიმედიური სისტემებისთვისაც. ამგვარად, მნიშვნელოვანია გამოკვლეულ იქნას მედიამონაცემთა შესაბამისი მონაცემთა ტიპების ისეთი განსაზღვრება, რომელიც ხელს შეუწყობს შემდგომში მათ ჩაშენებას განსხვავებულ მონაცემთა მოდელებში ან, სხვა სიტყვებით რომ ვთქვათ, მოითხოვება ამ მონაცემთა ტიპების განსაზღვრება მონაცემთა მოდელებისაგან დამოუკიდებლად. ჩვენ ქვემოთ განვიხილავთ ამ საკითხს ობიექტრელაციური და ობიექტორიენტირებული ბაზებისთვის.

ტრადიციული მონაცემთა მოდელების შემოთავაზება ფორმატირებული მონაცემებისა და მათი კავშირების სამართავად ასევე გამოყენებადია მულტიმედიური მბმს-თვისაც. როგორც ცნობილია, მედიაობიექტები (ტექსტი, გრაფიკა, სურათი, აუდიო და ვიდეო ფაილები) გვხვდება ყოველთვის მათ აღწერასთან ერთად [38]. ამასთანავე ისინი

დაკავშირებულია ერთმანეთთან და სხვა ნებისმიერ ფორმატირებულ მონაცემებთან. ეს შეიძლება შემდეგი ფორმებით გამოვლინდეს:

- ატრიბუტები (არსების ან ობიექტების). ობიექტი (პერსონა, მანქანა და ა.შ.) შეიძლება დამატებით აღიწეროს მისი სურათის, ხელმოწერის, ხმის საშუალებით. მედიაობიექტი აღადგენს გამოსახული ობიექტის თვისებას;
- კომპონენტები კომპლექსური ობიექტებისათვის. დოკუმენტი შედგება ცვლადი რაოდენობის მედიაობიექტისაგან: ტექსტის ბლოკების, სურათების, გრაფიკის და შესაძლო აკუსტიკური კომენტარებისაგან. ამჯერად მედიაობიექტები არის არა მხოლოდ ატრიბუტები, არამედ თვით ობიექტებიც. ამათ და დოკუმენტის ობიექტების შორის არსებობს რელაციური კავშირები, აგრეგატული დამოკიდებულების მსგავსად. შეიძლება განვიხილოთ ასევე სინქრონიზაციის დამოკიდებულება ვიდეოფილმის სურათებსა და ხმის ბილიკს შორის;
- ერთიდაიმავე ინფორმაციის ალტერნატიული წარმოდგენა. ხშირად ერთიდაიგივე ინფორმაცია აისახება სხვადასხვა მედიის საშუალებით, მაგალითად, ცხრილებისა და გრაფიკების სახით. არაა ყოველთვის შესაძლებელი ერთი სახის ინფორმაციის მიღება (გარდაქმნით) სხვა სახიდან გონივრული ხარჯებით. ამიტომ უფრო მისაღებია ორივე, როგორც მედიაობიექტები, იქნას ცხადად შენახული.

ამ მედიაობიექტებს შორის ეკვივალენტობის დამოკიდებულების ისეთი სახე უნდა გამოიყენებოდეს, რომელიც გამომავალი მოწყობილობის წვდომის საშუალებითა და მომხმარებლის მიდრეკილებით მბმს-თან, შეძლებს ერთი ან მეორე სახის არჩევას.

მონაცემთა მოდელი უნდა ცნობდეს დამოკიდებულებათა ამ განსხვავებულ ტიპებს მედიაობიექტებსა და ფორმატირებულ მონაცემებს შორის, ან თვით მედიაობიექტებს შორის, რაც შემდეგ პროგრამის შესრულებისას ოპერაციებმა უნდა გაითვალისწინოს.

2.1.1. მონაცემთა ახალი ტიპები

ძირითადი ამოცანა მდომარეობს მონაცემთა ტიპების (ან კლასების) განსაზღვრაში მედიამონაცემებისთვის [31]. შემოთავაზებული ოპერაციების სიმრავლე, განსაკუთრებული ყურადღებით უნდა შეირჩეს ტექსტისათვის, რასტრული სურათისა და გრაფიკისათვის და ა.შ.

მონაცემთა ტიპი Text. ახალი საბაზო მონაცემთა ტიპის განსაზღვრა განვიხილოთ Text ტიპის მაგალითზე.

ცხადია, რომ მონაცემთა ტიპი Text უნდა იყოს მეტი, ვიდრე char[]. ამიტომაც Text ტიპის ობიექტს საკმაო რაოდენობის ოპერაციები მიეწოდება, რომლებიც იღებენ განსხვავებულ ინფორმაციას ამ ობიექტიდან:

```
interface Text {
    public int length ();
```

```
public int alphabet ();  // 0 == ISO Latin-1, . . .
public int alphabetSize ();
public int language (); // 0 == English, 1 == German, …
public char charAt (int n);
public byte[] getASCII ();
public byte getEBCDIC ();
public String getUnicode ();
. . .
};
```

Text -ს აქვს ყოველთვის ერთი სიგრძე (გამოყენებული სიმბოლოების რაოდენობა). ტექსტი ხშირ შემთხვევაში განიხილება როგორც სტრიქონსტრუქტურირებული, რომელშიც კიდევ უკვე განსაზღვრულია ქვეტიპი. სტრიქონებად დაყოფა შეიძლება ინფორმაციის შემცველი იყოს როგორც ლექსებში. დაყოფა სიტყვებად დამოკიდებულია ენაზე და სიმბოლოების ერთობლიობაზე, მაგრამ, პრინციპულად, ყოველ ტექსტში შესაძლებელია. ამ სტრუქტურებიდან გამომდინარე, შეიძლება შემდეგი ოპერაციების განხილვა, რომლებიც სისტემისა და ინსტრუმენტისთვისაც ცნობილია:

```
// გამოაქვს შედეგად სტრიქონების რაოდენობა ტექსტში
public int lineCount ();

// გამოაქვს შედეგად სიტყვების რაოდენობა ტექსტში
public int wordCount ();

// გამოაქვს ტექსტიდან სტრიქონები მითითებული ნომრით
public char [ ] line (int lineNo);

// გამოაქვს ტექსტიდან სიტყვა მითითებული ნომრით
public char [ ] word (int wordNo);
```

ცხადია, რომ Text ტიპის ობიექტისთვის მრავალი ოპერაციის შეთავაზებაა შესაძლებელი, რომლებიც ტექსტის მთლიანი თუ ცალკეული ნაწილების დასამუშავებლად იქნება გამოყენებული. მაგალითად,

```
public boolean print (Printer p);
public boolean display (window w);
```

ეს ოპერატორები შედეგად იძლევა მხოლოდ დასტურს, რომ მონაცემთა გამოტანის პროცესი დასრულდა წარმატებით.

განვიხილოთ ცვლილების ოპერაციების მაგალითები:

```
public void replaceLine (int lineNo, char [] newLine);
public void insertLine (int lineNo, char [] newLine);
public void concatenate (Text t);
```

Text ტიპის ობიექტის საწარმოებლად საჭიროა ოპერაციის მომზადება, რომელიც სხვა ტიპის ობიექტებიდან შეძლებს Text ტიპის ობიექტში ასახვას. მაგალითად, java ენაზე შეიძლება დაიწეროს კლასის შემდეგი კოდი კონსტრუქტორით:

```
class TextClass implements Text {
   int length,
   int charLength,
   int code,          // 0 == ASCII, 1 == EBCDIC, . . .
   int formatter,     // 0 == none, 1 == PostScript, . . .
   byte endOfLine,
   byte [] characters
};
. . .
}
```

ასეთი ოპერაციით მოითხოვება შესაძლოდ ფართო გამოყენება, იგი საშუალებას იძლევა ტექსტის მრავალი სახე სხვადასხვა სისტემური გარემოდან გადასცეს (მონაცემთა ბაზის) ობიექტს, Text ტიპისას.

აქ განხილულ ოპერაციებს Text ტიპისათვის ჰქონდა მხოლოდ საილუსტრაციო ხასიათი. ისინი გვიჩვენებს, რომ საჭიროა მსგავსი ოპერაციების მომზადება მონაცემთა ასეთი ტიპის გამოყენებისას. მხოლოდ ასეთი ტექსტობიექტებით იქნება შესაძლებელი მულტიმედიურ მონაცემთა ბაზებში მუშაობა. საჭიროა სხვა ოპერაციების შემუშავებაც, რომლებიც განსაზღვრულ სამუშაო გარემოსთვის იქნება მორგებული. ეს შესაძლებელია უშუალოდ ტიპის გაფართოებით მოხდეს ან მისი სპეციალიზაციით.

მონაცემთა ტიპი Image აღიწერება ხშირად გამოყენებადი ოპერაციების საშუალებით. მისი შინაგანი სტრუქტურის შესახებ მომხმარებელთა უმრავლესობას ზოგადი წარმოდგენა აქვს, რომ იგი შედგება ფერთა ცხრილისა და პიქსელების მატრიცისაგან. არსებული Image-ობიექტების წასაკითხად საჭიროა შესაბამისი ოპერაციების მომზადება, რომლებიც შეძლებს ცალკეულ კომპონენტზე მიმართვას:

```
interface Image {
  public int height ();
  public int width ();
  . . .
 }
```

დანარჩენ ოპერაციებს შეუძლია სურათების სტატისტიკური შეფასება. მაგალითად,

public int pixelcount (byte [] pixelvalue);

ითვლის სიხშირეს, რომლითაც განსაზღვრული პიქსელის მნიშვნელობა ხდება.

სურათის წაკითხვისას მონაცემთა ბაზიდან შესაძლებელია მისი შემადგენელი ელემენტარული ნაწილების (პიქსელების) გამოძახება, ასევე შესაძლებელია გამოსახულების დართვა გამოთვლითი გარემოს სპეციფიურ მონაცემთა სტრუქტურაზე. მაგალითად, SUN-კომპიუტერებზე რასტრული გამოსახულება სათანადო დროის მანძილზე იმართება მონაცემთა სტრუქტურაზე, სახელით Pixrect, რომელიც შეიცავს მრავალ ზემოაღნიშნულ ნაწილს. ეს შეიძლება ასევე პირდაპირ იქნას წარმოებული:

public Pixrect getPixrect ();

ინტერაქტიულ გარემოში სურათის ნახვა ყველაზე გავრცელებული გამოყენების სახეა, ამიტომ, ყოველი შემთხვევისათვის იგი უნდა იყოს მხარდაჭერილი საკუთარი ოპერაციით:

public boolean display (Device);

სურათის ცვლილება შესაძლებელია და მისი განხორციელებისას ყურადღება უნდა გამახვილდეს მთლიანობის შენარჩუნებაზე: მხოლოდ პირველადი მონაცემების ცვლილება შესაბამისი რეგისტრაციის მონაცემების ადაპტაციის გარეშე, ან პირიქით პროცესის დროს, უნდა იქნას თავიდან აცილებული. შემდეგი ოპერაციები ასახავს განახლების მაგალითს, ასეთი მრავალი შეიძლება იყოს.

public Image window (int x0, int y0, int x1, int y1);

ჩამოიჭრება ფანჯრის გარეთ მდებარე სურათის ნაწილი,

```
public Image replaceColormap {
   Code encoding;
public int colormapLength;
public int colormapDepth,
public int [] [] colormap
};
```

სურათის ფერების ცხრილის შეცვლა შემდეგი სახით;

```
public Image replacePixelvalue {
   int x, int y,
   byte [] pixelvalue
};
```

იცვლება ცალკეული პიქსელის მნიშვნელობა.

როგორც ტექსტური ტიპისათვის იქნა ახსნილი, აქაც ეს ოპერაციები შეიძლება იყოს რეალიზებული პროცედურების სახით (ანუ შედეგის ტიპით void), თუ ეს სათანადო გადაწყვეტად იქნება მიჩნეული.

Image ტიპის მონაცემთა ობიექტის საწარმოებლად შემოთავაზებული ოპერაცია

class ImageClass implements Image {

```
    public ImageClas {
        int height,
        int width,
        int depth,
        float aspectRatio,
        Code encoding,
        int colormapLength,
        int colormapDepth,
        int [] [] colormap,
        byte [] pixelmatrix
    };
. . .}
```

მონაცემთა ტიპები Graphic, Sound, Video ანალოგიურად განისაზღვრება. აქ მათი განხილვა არაა წარმოდგენილი, ვინაიდან ახალ ასპექტებს არ შეიცავს.

მულტიმედიური მონაცემთა ტიპების გაფართოება მათი რეზიუმირებისათვის (შინაარსის მოკლე მიმოხილვისათვის) ხდება იმავე პრინციპით, როგორც ეს იყო ოპერაციების განსაზღვრის დროს, განსაკუთრებით შედარების ოპერაციებისათვის.

რადგან რეზიუმეების დროს საქმე ეხება დამოკიდებულ კომპონენტებს, რომლებიც მულტიმედიური ობიექტების გარეშე არ არსებობს, ისინი კაპსულირებული იქნება ობიექტთან. ეს მოითხოვს დამატებით ოპერაციებს მულტიმედიური ობიექტებისათვის. ქვემოთ ნაჩვენებია მაგალითი Image ტიპისათვის, რომლის მსგავსი იქნება ასევე სხვა მულტიმედია ტიპებისთვისაც. გამოსახულება გვიჩვენებს პროცედურებს.

```
interface Image {
    . . .
    public void newDescr (String descr);
    }
```

შეაქვს სურათის რეზიუმე descr-ში და ცვლის აქ მონაცემებს.

public void **extendDescr** (String descr);

აფართოვებს ძველ რეზიუმეს (თუ არსებობს) descr -ში მოცემულს.

public int **descrLength** ( );

იძლევა რეზიუმის სიგრძეს.

public String **getDescr** ( );

გამოაქვს რეზიუმე სტრიქონის ფორმატში.

public boolean **contains** (String query);

იკვლევს, რეზიუმე ხომ არ შეიცავს query-ს, ანუ ძებნის გამოსახულებას, რომელიც query-ს შეესაბამება. ესაა წარმომადგენელი შედარების ოპერაციებიდან, რომლებიც სურათებისთვის და სხვა მულტიმედიატიპებისათვის შეიძლება იყოს განსაზღვრული.

2.1.2. კლასთა იერარქიის აგება განზოგადების საფუძველზე

მულტიმედიური მონაცემთა ტიპების რეზიუმეების ოპერაციების განზოგადება შესაძლებელია გენერალიზაციის იერარქიის აგებით, რომელშიც „მშობელი" მონაცემთა ტიპი (სუპერკლასი) MediaObject არის მოთავსებული.

მისთვის განსაზღვრულია ოპერაციები, რომლებიც ყველა მედიაობიექტისთვისაა გამოყენებადი. შეიძლებოდა ასევე, მაგალითად, ერთი (ვირტუალური ან აბსტრაქტული) მეთოდის output აგება, რომლის კონკრეტული რეალიზაცია იქნებოდა თითოეული არსებული მულტიმედიური მონაცემთა ტიპი.

განზოგადების იერარქიის შემდგომი გამოყენება შესაძლებელია, რათა სუბკლასების დახმარებით მოხერხდეს ერთგვარი სიბინძურის თავიდან აცილება, რომელსაც აქამდე Image -სთვის განსაზღვრული ოპერაციები მიუთითებდნენ.

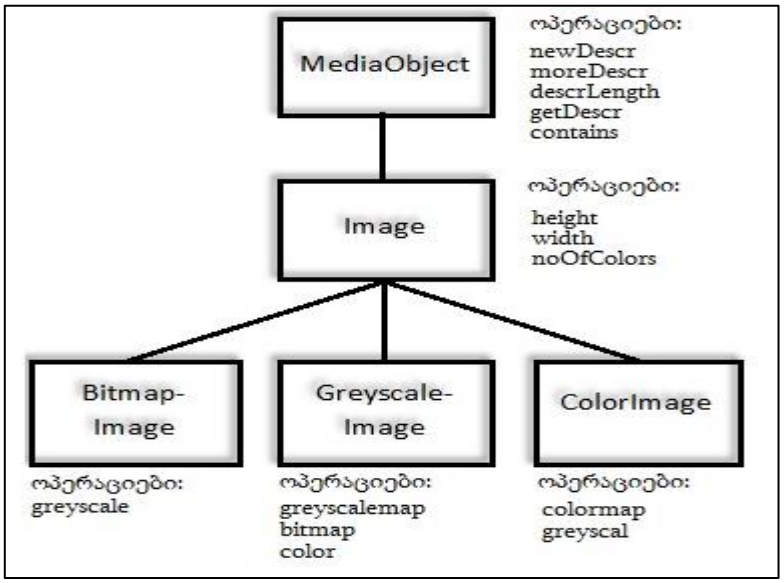
მაგალითად, არსებობს სურათები Pixel-ის სიღმით 1, რომლებიც არ იყენებენ, ფერთა ცხრილებს. Pixel მნიშვნელობით 0 , ნიშნავს თეთრს, ხოლო 1 კი - შავს. პრინტერებისა და მონიორიების დიდი კლასისათვის (Bitmap-დისპლეები) ასეთი სურათები მეტად მნიშვნელოვანია. მათთვის ისეთი ოპერაციების განსაზღვრა, როგორიცაა colormapLength და getColormap არის უაზრო.

ამავე დროს არსებობს სხვა ოპერაციები, რომელთა შემოტანა ასეთი ტიპის Bitmap-სურათებისათვის არის სასარგებლო [31].

ამგვარად, შემოთავაზებულია სამი ტიპის სურათებისათვის ცალკე სუბკლასების წარმოდგენა (ნახ.2.1).

Image-ს სუბკლასები ასახავს განსხვავებულ ინფორმაციულ შინაარსს რასტრული გამოსახულებებისათვის. თუ წარმოვიდგენთ ერთ ფოტოსურათს როგორც Bitmap-სურათს, მერე როგორც შავ-თეთრს და, ბოლოს როგორც ფერადს, ინტუიციურად მივხვდებით, რომ ფერადი სურათი მეტ ინფორმაციას შეიცავს, ხოლო Bitmap-სურათი-ყველაზე ნაკლებს.

ფერადი სურათის გარდაქმნას შავ-თეთრში ანგარიშობენ ფერის სიკაშკაშის მნიშვნელობით ან წაკითხულ იქნება სპეციალური ცხრილიდან, და ის დაკავშირებულია ყოველთვის ინფორმაციის დაკარგვასთან. იგივეა შავ-თეთრი სურათის გადაყვანისას Bitmap-სურათში. ასეთი ინფორმაციული დანაკარგები მხედველობაში უნდა იქნას მიღებული, როცა მათი წარმოდგენა იქნება საჭირო სრულიად განსაკუთრებულ გამომტან მოწყობილობებზე. ამისათვის არის გათალისწინებული ოპერაციები greyscale ColorImage-სთვის და bitmap GreyscaleImage--თვის.



ნახ.2.1. კლასთა იერარქია რასტრული სურათებისათვის

ყოველი რასტრული სურათი შეიძლება იყოს Bitmap-სურათი, შავ-თეთრი ან ფერადი. იგი მიეკუთვნება ყოველთვის მხოლოდ ერთ სუბკლასს. ამიტომაც თითოეული ფლობს თავის განსაკუთრებულ კონსტრუქტორებს. აქ მიღებული რასტრული სურათების სპეციალიზაციები ამიტომ არის გადაუკვეთადი და სრული. მონაცემთა ბაზების სისტემას უნდა შეეძლოს ასეთი მთლიანობის პირობის დაცვის გარანტირება.

2.1.3. SQL/MM: მულტიმედიური მონაცემები და სტანდარტები

მულტიმედიურ მონაცემთა ტიპების დაპროექტების პროცესის უნიფიკაციის საკითხებმა ასახვა პოვა ჯერ SQL:1999 გაფართოებაში, შემდეგ კი ერთიან საერთაშორისო ISO/IEC 13249 SQL/MM სტანდარტში [10,36,39]. სტანდარტი ხუთ პაკეტად დაიყო: ფრეიმვორკი (ან ჩარჩო, სტრუქტურა), სრული ტექსტი, ვექტორული გრაფიკა, რასტრული სურათები და მონაცემთა ინტელექტუალური ანალიზი (Data Mining).

ამასთანავე, SQL:1999 სინტაქსის ტერმინების გამოყენების თვალსაზრისით, კლასების ცნებისათვის გამოყენებულ იქნება „მომხამარებელის მეიერ განსაზღვრული ტიპები“ (UDT – User-Defined Types) ხოლო ოპერაციებისათვის - „მომხმარებლის მიერ განსაზღვრული ფუნქციები“ (UDF – User-defined Functions).

- **SQL/MM სრული ტექსტი**

ეს პაკეტი განსაზღვრავს UDT FullText-ს ტექსტური მონაცემებისათვის და სხვა UDT FT_Pattern-ს საძებნი შაბლონებისათვის (ან მოდელებისთვის). FullText მიუთითებს ოთხ საძებნ მეთოდზე, რომელთაგან ორი განსხვავდება მხოლოდ მათი პარამეტრების ტიპებით.

კერძოდ, ესაა ან სიმბოლოების ჯაჭვი (სტრიქონი) ან FT_Pattern-ის ტიპის შაბლონი. მეთოდების სახელები გადატვირთვადია (overloaded).

ორი Contains-მეთოდი ახორციელებს ლოგიკურ ძებნას. შედეგის სახით მიიღება მხოლოდ „დიახ“ ან „არა“. ხოლო ორი Rank-მეთოდი კი, მეორე მხრივ, განსაზღვრავს რიცხვს მცოცავი წერტილით, რომელიც გამოიყენება პოზიციონირებისათვის (ranking).

ამას გარდა FullText-ს აქვს ორი კონსტრუქტორი. ერთი აწარმოებს ტექსტის ობიექტს სტრიქონიდან, მეორე კი დამატებით არეგისტრირებს ტესქტის ენას. დასასრულ, არსებობს კიდევ ერთი ფუნქცია FullText_to_Character სტრიქონების მისაღებად, ტექსტის გამოტანის მიზნით.

SQL:1999 სინტაქსით UDT-განსაზღვრება სტანდარტულად შემდეგი სახით ჩაიწერება:

```
create type FullText as (
    Contents character varying (FT_MaxTextLength),
    Language character varying (FT_MaxLanguageLength),
. . .
)
method Contains (pattern FT_Pattern)
        returns integer
method Contains (pattern character varying (FT_MaxTextLength))
        returns integer
method Rank (pattern FT_Pattern)
        returns double precision
method Rank (pattern character varying (FT_MaxTextLength))
        returns double precision
method FullText (String character varying (FT_MaxTextLength))
        returns FullText
method FullText (
        String . . . ,
        Language character  varying (FT_MaxTextLength)
)
        returns FullText;
create cast (FullText as
      character varying (FT_MaxTextLength)
      with FullText_to_Character);
create type FT_Pattern as
      character varying (FT_MaxPatternLength);
```

FT_Pattern მნიშვნელობები უნდა იყოს გამოსახულებები, აღწერილი ბეკუს-ნაურის ფორმის ენის საშუალებით. შეფასება აღიწერება წესების საშუალებით ენის სიმბოლოების მიხედვით. განვიხილოთ მაგალითები ამ საკითხებზე.

ტექსტური ობიექტი აქ მოიცემა aText სახით და შინაარსით: „ეს სექცია წარმოადგენს SQL/MM სტანდარტს. ამ სტანდარტით განსაზღვრულია მულტიმედიური ობიექტების ტიპები და ქვეპროგრამები“.

უმარტივესი ძებნის შაბლონი არის ერთი სიტყვა:

aText.Contains(‘ “სექცია“ ‘) = 1

ეს გამოსახულება ასახავს შემთხვევას, როცა შედეგი „ჭეშმარიტია“ („true“ ანუ 1).

შესაძლებელია ძებნის განხორციელება სიტყვების სიმრავლითაც, სადაც გამოყენებულ იქნება სიტყვებსშორისი შემავსებლები (placeholder, wildcards), მაგალითად „ქვედა ტირე“.

aText.Contains(‘ “სექცია_“ ‘) = 0.

შედეგი მიღებულა ამჯერად „მცდარი“ (false ანუ 0) მნიშვნელობით.

ძებნის შაბლონებში შესაძლებელია აგრეთვე თეზაურუსის გამოყენებაც, მასზე მიმართვით. იგი გაფართოებული შაბლონით აფორმირებს ახალ სიტყვას, მაგალითად, მსგავსი სიტყვები, განზოგადებული სიტყვები, სპეციალური სიტყვები, სინონიმები ან წარმოებული:

```
aText.Contains ( ‘
      thesaurus “Informatik”
      expand synonym term of “Norm”
  ‘ ) = 1
```

აქ იგულისხმება, რომ „ინფორმატიკის“ თეზაურუსი სიტყვისათვის „Norm“ პოულობს სინონიმს „Standard“ (გერმანული ენა). სიტყვათა მიმდევრობა და დაცილება შესაძლებელია განისაზღვროს კონტექტური შაბლონის ფორმით:

```
aText.Contains ( ‘
      (“მოხსენება“)  near “სტანდარტი”
      within 0 sentences in order
  ‘ ) = 1
```

დასასრულ, შესაძლებელია ასევე ე.წ. კონცეფციური შაბლონის (concept pattern) განხილვა ტექსტის მნიშვნელობის მისამართად:

```
aText.Contains ( ‘
      is about “Internationaler Standart
      yur Volltextsuche”
  ‘ ) = 1
```

ამ დროს ღიადაა დატოვებული, თუ is about როგორ იქნება რეალიზებული.

ყველა ზემოგანხილული მაგალითი იყო ცალკეული ძებნის-ფრაზები. შესაძლებელია მათი კომბინაციური გამოყენებაც, კონიუნქციებისა და სხვა ლოგიკური ოპერაციების გაერთიანების საფუძველზე.

შეიძლება განვიხილოთ უმარტივესი შაბლონი, რომელიც მოთხოვნის მაგალითის კონტექსტში იქნება ნაჩვენები:

```
select * from myDocs
where Doc.Rank(‘ “ Standard” ‘) > 0.8
```

ამგვარად, სისტემა უნდა აკონტროლებდეს (შეფასების თავალსაზრისით), რომ მოხდა ინფორმაციის სწორი ამოცნობა, ანუ სიტყვის, წინადადების, აბზაცის მიხედვით (კონტექსტური შაბლონებით). გაფართოებული შაბლონებით კი მიიღწევა სწორი (ანალოგიური, მსგავსი) სიტყვების პოვნა.

- **SQL/MM სივრცითი ტიპი (Spatial)**

მონაცემთა სივრცითი ტიპი Spatial გამოიყენება “graphic” ტიპის ობიექტებისათვის [39-41]. განიხილება UDT-ები 2D-მონაცემებისა (წერტილი, წრფეები, სიბრტყეები) და მათი კოლექციებისათვის. მათ შეიცავს პროგრამები სივრცითი მონაცემების მანიპულაციის, ძებნისა და შედარებისათვის, აგრეთვე კონვერტირებისათვის UDT და სიმბოლოებს ან ორობით წარმოდგენებს შორის.

ცალკეული ტიპები ორგანიზებულია განზოგადებული იერარქიით. ფესვის ტიპს ეწოდება ST_Geometry. ყოველ გეომეტრიულ ობიექტს მიეკუთვნება ერთი SRID (Spatial reference system identifier), რომელიც ახასიათებს სივრცით ათვლის სისტემას. იგი ეფუძნება ცნობილ ათვლის სისტემებს: გეოგრაფიული საკოორდინატო სისტემა (განედი და გრძედი), საპროექციო საკოორდინატო სისტემა (X-ით, Y-ით და Z-ით). ისინი აღიწერება, მაგალითად ასეთი სახის BNF-ით:

```
<geographic system> ::= <projected cs> | <geocentric cs>
<geographic cs> ::= GEOGCS <left delimiter>
   <double quote> <name> <double quote> <comma>
   <datum> <comma>
   <prime meridian> <comma>
   <angular unit>
<right delimiter>
```

ST_Geometry ტიპის კოლექციის ელემენტებისათვის და ST_Geometry ველის შიგნით უნდა იქნას შერჩეული ერთი და იგივე ათვლის სისტემა.

დეტალურად რომ დავახასიათოთ, არსებობს შემდეგი ტიპები:

- „ნულგანზომილებიანი“ - განიხილება წერტილი: ST_Point;

- ერთგანზომილებიანი ობიექტები არის ზოგადად ST_Curve, რომელშიც შესაძლებელია ქვეტიპების შექმნა, რომლებიც განსაზღვრავს ინტერპოლაციას ცალკეულ წერტილებს შორის: ST_LineString – წრფივი ინტერპოლაციაა; ST_CircularString – წრიული ფორმისა და ST_CompoundString – შერეული ფორმისაა;
- ორგანზომილებიანი ობიექტები - ST_Surface, სადაც ST_CurvePolygon განისაზღვრება გარე და შიგა ST_CompoundString საზღვრებით, მაშინ, როცა ST_CurvePolygon იყენებს მხოლოდ ST_LineString საზღვრებს.

დამატებით არსებობს აგეთვე კოლექციის ობიექტები, რომლებიც აკეთებს იმას, რაც სხვა სისტემებისგან ცნობილია „დაჯგუფების“ (grouping) სახით. ისინი მოითხოვს, როგორც უკვე აღინიშნა, ერთნაირ ათვლის სისტემას ყველა ელემენტისათვის. არსებობს ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface და ST_MultiPolygon, რომელთა სახელები ცხადად მიუთითებს, თუ ელემენტის რომელი სახისთვის არის თითოეული დანიშნული.

ST_Geometry-სა და ქვეტიპებზე განსაზღვრულია შემდეგი მეთოდები: საშუალო (წერტილთა სიმრავლის საშუალო), სხვაობა და გაერთიანება, მანძილის გაანგარიშება, ტესტირება (contains, overlaps, touches, croses,…), და ათვლის სისტემის განსაზღვრა.

ამასათანავე ხდება აგრეთვე რამდენიმე სხვა მეთოდის დამატება ქვეტიპებისათვის, როგორიცაა, მაგალითად, length ST_Curve-თვის, area და perimeter ST_Surface-თვის.

➢ SQL/MM სტილი სურათი/ფოტო (Styl Image)

განვიხილოთ სურათის (ან ფოტოს) სტილის სტანდარტის გამოყენების საკითხი. თანამედროვე სტანდარტების შემოთავაზებით, მაგალითად, UDT-ს SI_StillImage გამოიყენება სურათის (ფოტოს) მონაცემებისათვის, SI_Feature - სურათების თვისებების-თვის და SI_FeatureList ასეთი თვისებების სიისათვის. SI_StillImage-ს შემთხვევაში, საინტერესოა, რომ ატრიბუტების სია, ანუ შინაგანი წარმოდგენა ღიაა და შესაძლებელია მასზე მიმართვა:

```
create type SI_StillImage  as (
    SI_content binary large object (SI_MaxContLength),
    SI_contentLength integer,
    SI_format character varying (8),
    SI_height integer,
    SI_width integer,
    . . .
)
```

ატრიბუტი SI_content შეიცავს სათაურს, ფერის ცხრილებს და სხვა მსგავს მონაცემებს, ანუ სარეგისტრაციო მონაცემებს და აღწერის მონაცემებსაც. „Container“-ის სახით განიხილება მთლიანი სურათი.

ატრიბუტი SI_format ის ფორმატია, რომლითაც სურათი ინახება კონტეინერში, მონაცემთა მასისვის სუფიქსის მსგავსად. განასხვავებენ „მხარდამჭერ“ და „მომხმარებლის_მიერ_ განსაზღვრულ“ ფორმატებს. მხარდამჭერი ფორმატები იკითხება თვით მბმს-ის მიერ, ისე, რომ ხდება BLOB-ის შინაარსის ინტერპრეტაცია. მაგალითად, მას შეუძლია სურათის თვისებების ამოღება. მომხმარებლის მიერ განსაზღვრული ფორმატები კი, თავის მხრივ, იმართება მბმს-ით მხოლოდ როგორც სახელები. ინტერპრეტაციის საკითხი რჩება აპლიკაციის გადასაწყვეტი.

ცხდად ჩანს, რომ მონაცემთა დამოუკიდებლობა აქ არ თამაშობს განსაკუთრებულ როლს. ანუ ის მოთხოვნები, რომლებიც ზემოთ იქნა განსაზღვრული სხვა ტიპებისათვის, ნაწილობრივ ხორციელდება. როგორც ჩანს, იგი უფრო ორიენტირებულია ფაილური სისტემების (მასივების) მოდელებზე, ვიდრე მონაცემთა ბაზებზე ტიპებით.

ეს საკითხი შემდგომ აისახება ასევე ოპერატორებზეც. SI_StillImage-ს აქვს ორი კონსტრუქტორი, რომელთაგან ერთი შესასვლელზე ელოდება უბრალოდ BLOB-ს, ხოლო მეორე, პირიქით, BLOB-ის გარდა სხვა ფორმატის მონაცემებს.

ამიტომ არსებობს ცვლილებათა ორი მეთოდი: პირველი ჩაანაცვლებს შიგთავსს (Content-ს) ახალი BLOB-ით, და შეიძლება იმედი ვიქონიოთ, რომ სხვა ატრიბუტებიც მლიანობას მოერგება. მეორე მეთოდი ცვლის ფორმატს და ამან უნდა მოახდინოს Content-ის (შიგთავსის) შინაარსის კონვერტირება.

არსებობს აგრეთვე წაკითხვის ორი მეთოდი მინიატურული სურათის (Thumbnails) საწარმოებლად:

```
method SI_StillImage (
    content binary large object (SI_MaxContLength)
) returns SI_StillImage

method SI_StillImage (
    content binary large object (SI_MaxContLength)
format character varying (. . .)
) returns SI_StillImage

method SI_setContent (
    content binary large object (SI_MaxContLength)
) returns SI_StillImage

method SI_changeFormat (
    targetFormat character varying (. . .)
) returns SI_StillImage
```

ფორმატის ცვლილების მეთოდები გამოიყენება მხოლოდ არსებული (მხარდაჭერილი) ფორმატებისთვის.

აღწერილი მონაცემებისათვის შესაძლებელია სურათის მახასიათებლების მიღება და შენახვა ცალკე სვეტებში სურათის გვერდით. SI_Feature საბაზო ტიპს აქვს შემდეგი ქვეტიპები:

- SI_AverageColor - ასახავს მთლიან სურათს ერთ ფერში, უბრალოდ საშუალო ფერით;
- SI_ColorHistogram - მართავს ფერთა ჯგუფების სიხშირეებს [36];
- SI_PositionalColor - მიიღება სურათის დაშლით მართკუთხედებად, რომლითაც განისაზღვრება შემდეგ საშუალო ფერი;
- SI_Texture - შეიცავს განმეორებადი ელემენტების სიდიდეებს, სიკაშკაშის ვარიაციებს და დომინირებად მიმართულებებს.

ყველა მახასიათებელი ხელმისავდომია SI_Score მეთოდით, რომელიც ითვლის სურათის დისტანციას მახასიათებლებში და იძლევა შედეგს 0 -:-1 ინტერვალში.

SI_Feature-ს ყველა ქვეტიპს აქვს მახასიათებლების ამოღების ფუნქცია (იმავე სახელით, როგორიც აქვს ქვეტიპს), რომელიც სურათს გამოიყენებს არგუმენტის სახით და იძლევა ქვეტიპის ობიექტს შედეგის სახით.

SI_AverageColor და SI_ColorHistogram ქვეტიპთა ობიექტებს შეუძლია ამასთანავე უშუალოდ იქნას აგებული კონსტანტების ჩაწერის საშუალებით. საილუსტრაციო მაგალითი მოცემულია საშუალო ფერის სინტაქსის აღწერით.

```
create type SI_Feature
method SI_Score (image SI_StillImage)
    returns double precision

create type SI_AverageColor under SI_Feature as
        (SI_ AverageColorSpec SU_Color)
method SI_ AverageColor (
        ReadValue integer,
        GreenValue integer,
        BlueValue integer
)  returns SI_AverageColor

create function SI_AveageColor (Image SI_StillImage)
    returns SI_AverageColor
```

სურათი შეიძლება აღიწეროს სხვადასხვა მახასიათებლით (თვისებებით), და მისი საშუალებით შეუძლებელია შედარების თვითშეფასების განხორციელება, ამიტომ შემოთავაზებულია შესაძლებლობა, ისინი გაერთიანდეს ერთიან თვისება-მნიშვნელობის წყვილების სიაში. იგი ქმნის ტიპს SI_FeatureList. მას აქვს აგრეთვე SI_Score მეთოდი, რომელიც შეწონილ საშუალო მნიშვნელობას იძლევა ცალკეული Score შეფასებისათვის:

```
        self.SI_Features[1].SI_Score(img) * self.SI_Weights[1]
      + self.SI_Features[2].SI_Score(img) * self.SI_Weights[2] + . . .
      / (self.SI_Weights[1] + self.SI_Weights[2] + . . .)
```

კონსტრუქტორის მიერ ხდება ზუსტად ერთი თვისებისა და ერთი წონის სიის ინიციალიზაცია, რომელიც SI_Append მეთოდით ამატებს მომდევნო თვისებას თავის წონასთან ერთად.

```
create type SI_FeatureList as (
      SI_Features SI.Feature array[SI_MaxFeatureNumber],
      SI_Weights double precision array[SI_MaxFeatureNumber]
  )
method SI_FeatureList (firstFeature SI_Feature, weight double precision)
    returns SI_FeatureList
method SI_Append (feature SI_Feature, weight double precision)
    returns SI_FeatureList
```

განვიხილოთ ერთი მაგალითი, რომელშიც შეჯამებულია აღწერილი შესაძლებლობები SQL-ის კონტექსტში. წარმოდგენილია Logos რელაცია, რომელიც სურათის ერთი Logo ატრიბუტით უნდა იქნას გამოკვლეული. შესადარებელი ობიექტის სახით გამოიყენება bspLogo, რომლიდანაც მიიღება ტექსტურა და ფერთა ჰისტოგრამა. ამ მახასიათებლებიდან აიგება თვისებათა სია, რომელშიც ჯერ შეიტანება ტექსტურა წინით 0.8 (კონსტრუქტორი) და შემდეგ ფერთა ჰისტოგრამა წონით 0.2 (SI_Append მეთოდი). შედგენილი თვისებების სიიდან გამოიძახება SI_Score, და, კერძოდ, ველით Logo პარამეტრის სახით. მსგავსებისთვის შეწონილი საშუალო მნიშვნელობა უნდა იყოს 0.7-ზე მეტი.

```
        select * from Logos
        wher  SI_FeatureList (
                SI_Texture (bspLogo), 0.8
          ).SI_Append (
              SI_ColorHistogram (bspLogo, 0.2)
           ).SI_Score (Logo) > 0.7
```

დასასრულ, შეიძლება მოკლედ შევაჯამოთ SQL/MM სტანდარტის წინადადებები.

Spatial და Still Image-ს გამოყენების დროს გვხვდება ყველგან ST ან SI პრეფიქსები, რომლებიც Full Text შემთხვევაში არ არსებობდა. Full Text -ს აქვს ფუნქცია Rank, რომელიც Still Image-სთვის გვხვდება Score-ს სახით. ვინაიდან სპეციალიზება ხდება ტიპებზე, შეიძლებოდა ასევე MM_Object-თვის განზოგადება ან მსგავსი ხერხის შემოთავაზება, რომლის დროსაც ერთი (ვირტუალური) MM_Score მეთოდი თავის ადგილს იპოვნის.

2.2. ობიექტრელაციური მულტიმედიური მონაცემთა ბაზის სისტემები

წინა პარაგრაფში შემოტანილ იქნა მონაცემთა ახალი ტიპები მულტიმედიური ობიექტებისათვის და იყო მცდელობა, შეძლებისდაგვარად, განხორციელებულიყო ჯერ არარსებული კავშირების ასახვა მონაცემთა მოდელზე. ამჟამად ჯერ კიდევ არ არსებობს მულტიმედიურ მონაცემთა (მაგალითად, მულტიმედიური დოკუმენტები აგრეგაციული სტრუქტურებით) ობიექტების მოდელირების საშუალებები [31,36].

შემოთავაზებულია მონაცემთა მოდელების იმ შესაძლებლობათა გამოყენება, რომლებშიც დამოკიდებულებები და აგრეგაციები (სხვადასხვა ფორმით) ხელმისაწვდომია. ამას მივყავართ კითხვამდე, თუ როგორ იქნება ახალი მონაცემთა ტიპები ჩაშენებული ამ მონაცემთა მოდელების კონტექსტში და როგორი სახე ექნება შემდგომ მათ გამოყენებას.

ეს საკითხი, პირველ რიგში, გამოკვლეულ იქნება რელაციური ბაზების სისტემის მაგალითზე. დღეს ბევრს საუბრობენ ობიექტრელაციურ სისტემებზე, ამასთანავე მათი გაფართოებები არის ძალზე სასარგებლო მულტიმედიური ობიექტებისათვის. ასეთ სისტემების საფუძველი ყოველთვის რელაციური მოდელებია. მათ უდავოდ აქვს უდიდესი პრაქტიკული მნიშვნელობა.

ჩვენ ქვემოთ დავახასიათებთ ჯერ მათ იმ განსაკუთრებულ თვისებებს, რომლებიც მულტიმედიისათვისაა მთავარი, შემდეგ ნაჩვენები იქნება, როგორ პროექტდება სქემები, რომლებიც მულტიმედიური ობიექტების ჩადგმულ მონაცემთა ტიპებს გამოიყენებს. და ბოლოს, ნაჩვენები იქნება, თუ როგორ შეიძლება SQL-მოთხოვნების ენით, რომელიც ასეთი სისტემების მნიშვნელოვანი ენაა, განხორციელდეს მონაცემებთან მიმართვა ობიექტრელა-ციურ მონაცემთა ბაზებში.

- **მოკლე დახასიათება**

ობიექტრელაციური მონაცემთა ბაზების სისტემა (ორმბს) წარმოადგენს მცდელობას, რელაციური მბს ისე განვითარდეს და ახალი კონცეფციებით გამდიდრდეს, რომ მან შეძლოს, მინიმუმ, ფუნქციონალობათა ერთი ნაწილის შემოთავაზება, რომელიც აქამდე ცნობილი იყო ობიექტორიენტირებული მბს-თვის. ესენია, კერძოდ, ობიექტთა იდენტურობა, მომხმარებელთა მიერ განსაზღვრებადი მონაცემთა ტიპები და ტიპთა იერარქიები. თუმცა ძირითადი სტრუქტურა ყველა მონაცემთა ორგანიზაციისა და მონაცემთა შენახვისა ისევ ცხრილები რჩება. ასე რომ, ყველა ეს ახალი გაფართოება უნდა დაექვემდებაროს, როგორც ადრე, ამ (წინა) კონცეფციას.

მულტიმედიური მონაცემებისათვის თანამედროვე სტანდარტებით განიხილება დიდი ობიექტები (large objects), ორი ფორმატით: სიმბოლოების ობიექტი (character large object - CLOB) და ბინარული ობიექტები (binary large object - BLOB).

არსებობს გარკვეული შეზღუდვები ამ ტიპის ატრიბუტებისათვის: ისინი არ შეიძლება იყოს განსაზღვრული როგორც პირველადი გასაღებური ატრიბუტები, UNIQUE ან

მეორეული გასაღებური ატრიბუტები. აგრეთვე არ შეიძლება მათი გამოყენება GROUP BY ან ORDER BY კონსტრუქციებში.

განსაკუთრებით მნიშვნელოვანია სტრუქტურირებული, მომხარებალთა მიერ განსაზღვრული ტიპები (UDT's). ეს ტიპები შეიცავს ატრიბუტებს და მეთოდებს. ასევე ძველ ფუნქციებს და პროცედურებს, რომლებიც 1996 წლიდად SQL92-ის სტანდარტის გაფართოებით იქნა განსაზღვრული, შეეძლოთ ასეთი ტიპის პარამეტრებით სარგებლობა [42].

პროცედურები განსაზღვრული იყო ისე, რომ ისინი შედეგის მნიშვნელობას უკან არ აბრუნებდა, მაგრამ შემავალი და გამომავალი პარამეტრები შეიძლება ჰქონოდა.

ფუნქციები, პირიქით, აბრუნებს უკან მიღებულ შედეგებს და აქვს მხოლოდ შემავალი პარამეტრები.

ტიპები შეიძლება იყოს ორგანიზებული იერარქიებად. ამავე დროს, ტიპები მიეკუთვნება ეგზემპლარულს (instantiable), თუ მათ შეუძლია უშუალოდ შედეგის მნიშვნელობების მოცემა. საწინააღმდეგო შემთხვევაში ტიპები აბსტრაქტული ან ვირტუალურია, რაც ნიშნავს, რომ მათ შეუძლია მხოლოდ ქვეტიპების მნიშვნელობებზე მითითება. ქვეტიპების განსაზღვრა ამით თავიდან აცილებულია, რადგანაც ტიპი თვითონ ასრულებს ამ ფუნქციას.

იერარქიულობის ეს პრინციპი არის ურთიერთშენაცვლების: სადაც ტიპის მნიშვნელობა შეიძლება მოთავსდეს, იქ შეიძლება ასევე ქვეტიპების მნიშვნელობათა არსებობა. იმისათვის, რომ ეს შესრულდეს, ტიპის ატრიბუტები და მეთოდები ასევე წვდომადი უნდა იყოს მისი ყველა ქვეტიპისათვის, ანუ მემკვიდრეობითობის თვისება უნდა იყოს რეალიზებული.

განვიხილოთ მაგალითი ტიპისა და ქვეტიპებისათვის.

```
create type Student
    under Person
     as (
         MatrNr  integer,
         Studienfach varchar(30),
    )
    instance method Durchschnittsnote()
           return real
           language SQL
           deterministic
           contains SQL
    . . . ;
```

მეთოდები მხოლოდ ცხადდება ტიპების სღწერაში, ანუ განისაზღვრება მათი სიგნატურები. რეალიზაცია შეიძლება მოგვიანებით განხორციელდეს (ბრძანება create

method). ჩვეულებისამებრ, განასხვავებენ ეგზემპლარულ მეთოდებს (instance method) კლასთა მეთოდებს (static method).

დამატებითი ინფორმაცია მეთოდისათვის შემდეგია:

- returns - განსაზღვრავს უკან დასაბრუნებელ მნიშვნელობის ტიპს. იგი აღწერილია create-method ბრძანებაში;
- language SQL – მიუთითებს, რომ მეთოდის ტანის რეალიზაცია მთლიანად ხორციელდება SQL -ში;
- deterministic – მიუთითებს, რომ უშუალოდ ერთმანეთის მომდევნო გამოძახებები ერთი და იმავე პარამეტრების მნიშვნელობებით, ყოველთვის ერთსა და იმავე შედეგს იძლევა;
- contains SQL – მიუთითებს, რომ მეთოდის რეალიზაცია შეიცავს SQL-ბრძანებას, რომელიც არ ეხება ეგზემპლარის ან კლასის მონაცემებს, არამედ მხოლოდ სხვა მონაცემებს ეხება. ალტერნატიული ინფორმაციაა no SQL, როცა არაა მოცემული SQL-ბრძანება; reads SQL data, როცა ეგზემოლარის ატრიბუტები მხოლოდ უნდა წაკითხულ იქნას; modifies SQL data, როცა ატრიბუტები ასევე უნდა შეიცვალოს;
- returns null on null input - მიუთითებს, რომ შედეგი არ ბრუნდება უკან, როცა შემავალი პარამეტრია Nullwert (zero values);

ერთი ტიპის ყოველი ატრიბუტისათვის ორი მეთოდი ავტომატურად გენერირდება:

დამკვირვებელი (Observer) პასუხისმგებელია წაკითხვის წვდომაზე.

```
instance method MatrNr ()
    returns integer
    language SQL
    deterministic
    contains SQL
```

ვინაიდან მეთოდის გამოძახებისას პარამეტრების გარეშე ყოველთვის ხდება ფრჩხილების გამოტოვება, ამ მეთოდის ვიზუალური გამოძახება არ განსხვავდება ატრიბუტებთან ნორმალური წვდომისაგან: Student1.MatrNr.

მუტატორი ახდენს ატრიბუტის მნიშვნელობის ცვლილებას (ჩანაცვლებას):

```
instance method MatrNr (new value)
    returns Student
    self as result
    language SQL
    deterministic
    contains SQL
    returns null on null input
```

შემდეგი მაგალითი უფრო გამოკვეთილად შეესაბამება მულტიმედიურ შემთხვევას:

```
create type ImageType
    under <Supertyp>
```

```
        as  (< Attributliste> )
        static method countImages ()
        returns integer
        language SQL
        deterministic
        contains SQL
    instance method height ()
        returns integer
        language SQL
        deterministic
       rads SQL data
```

კიდევ ერთხელ, საბოლოოდ უნდა გაესვას ხაზი იმას, რომ კორტეჟები და ცხრილები შეუცვლელად თამაშობს მნიშვნელოვან როლს: იგი უშუალოდ დაკავშირებულია კორტეჟების შეტანასთან ცხრილებში, სხვა ობიექტები ან მნიშვნელობები არ იქმნება. ასევე ყოველთვის შესაძლებელია მოთხოვნები ცხრილებისადმი, რომლებიც გამოსასვლელზე იძლევა ისევ ცხრილებს.

- **ობიექტრელაციური ბაზის სტრუქტურები მულტიმედია ობიექტისათვის**

ობიექტრელაციურ ბაზის სქემაში დასაშვებია მონაცემთა ახალი ტიპების გამოყენება მულტიმედიური ობიექტებისათვის, როგორც მნიშვნელობათა არეები (domains). ატრიბუტები შეიძლება იყოს ტიპებით text, image და სხვ. მაგალითად, ეკომონიტორინგის ყოველი თანამშრომლისათვის კორტეჟში უშუალოდ ჩაისმება პირადი ნომერი, გვარი, დაბადების თარიღი, და ა.შ., ასევე ფოტო (სურათი), თითების ანაბეჭდი, ხმა და მოხდება მისი შენახვა:

```
ecoEmployee ( Pers_N integer,
        PersName varchar (50),
        BirthsDate date,
        Adress varchar (100),
            . . .
       PersFoto image,
       PersFingerprints image,
       PersVoice sound)
```

დავარქვათ ამ სტრუქტურას, პირობითად, რელაციური სქემა (ტიპი_1), რათა შემდგომში შემოტანილი სხვა ტიპისაგან განვასხვავოთ.

ნორმალურ ფორმათა თეორიის საფუძველზე, სქემათა ოპტიმალური სტრუქტურის მისაღებად, ხშირად საჭიროა ერთი რელაციის დეკომპოზიცია ორ ან მეტ რელაციად, რომლებშიც მოხდება შესაბამისი კორტეჟების გადანაწილება, პირველადი და მეორეული გასაღებური ატრიბუტების განსაზღვრა და ა.შ.

ჩვენი მაგალითისათვის შესაძლებელია თანამსრომლის პირადი ტექსტური მონაცემების ერთ რელაციაში შენახვა, ხოლო მულტიმედიური მონაცემებისა მეორეში. ქვემოთ ნაჩვენებია ეს შემთხვევა:

```
ecoEmployee ( Pers_N integer,
        PersLastName varchar (50),
        PersFirstName varchar (30),
        BirthsDate date,
        Adress varchar (100),
        Status varchar(30),
            . . .
        )
 ecoEmployee_MM_Daten( Pers_N integer,
        PersFoto image,
        PersFingerprints image,
        PersVoice sound)
```

მიღებული სტრუქტურა არის რელაციური სქემა (ტიპი_2). ამ შემთხვევვაში თანამშრომლის ვინაობა (გვარი, სახელი,...) უკავშირდება მულტიმედიური მონაცემების რელაციას პირველადი გასაღებური ატრიბუტით Pers_N. ამ ორი რელაციის საფუძველზე მოთხოვნების დასამუშავებლად საჭირო იქნება „Join" გაერთიანების ოპერაციის გამოყენება.

არსებობს შემთხვევები, როდესაც რელაციებს შორის არსებობს m:n დამოკიდებულება. ამ დროს საჭიროა რელაციებსშორის კავშირების რეალიზაცია დამატებითი ინდექსების საფუძველზე, მათ შორის პირველადი და მეორეული გასაღებებითაც. განვიხილოთ მაგალითი, რომელიც ასახავს თანამშრომლის (ეკომაჩვენებლების სპეციალისტის სტატუსით) (ecoEmployee) განაწილებას საკონტროლო წერტილებზე (მაგალითად, კონკრეტულ ესტუარებზე) (Estuaries). ერთი სპეციალისტი შეიძლება იღებდეს წყლის სინჯებს რამდენიმე მდინარის ესტუარში, ასევე ერთ ესტუარზე შეიძლება მუშაობდეს რამდენიმე სპეციალისტი (ერთდროულად ან სხვადასხვა დროს), ანუ საქმე გვაქვს m:n დამოკდებულებასთან:

```
ecoEmployee (Pers_N integer,
        PersName varchar (50),
        . . .
        PersFoto image
        )
Estuar ( estuarID integer,
        Est_Coord_X integer,
        Est_Coord_Y integer,
        riverID integer,
```

```
    area_Sqm integer
  )
Results_List (estuarID integer,
    Pers_N integer,
    dateExp date,
    air_T1_Celsius decimal(5,2),
    water_T2_Celsius decimal(5,2),
    pH integer,
    TDS integer
  )
```

მივიღეთ რელაცირი სქემა (ტიპი_3). ამგვარად, შეიძლება შევაჯამოთ შედეგები:

- სამივე ტიპის რელაციური სქემა მულტიმედიურ ობიექტებსა და არსებს შორის შეიძლება აისახოს 1: 1, 1 : n, m : n დამოკიდებულებათა სახით, სპეციალური სემენტიკის გარეშე;
- მულტიმედიური ობიექტები შეიძლება გამოვლინდეს როგორც ატრიბუტები ან როგორც არსები;
- მიმართვა ზოგიერთ რელაციურ სქემაზე შეიძლება იყოს მოუხერხებელი და მოითხოვდეს რამდენიმე გაერთიანების ოპერაციის (Join) გამოყენებას.

2.3. მოთხოვნების დამუშავება ობიექტრელაციურ მონაცემთა ბაზებში

რელაციურ მონაცემთა ბაზების შესახებ, რომლებშიც ტრადიციული, ტექსტური ტიპის ატრიბუტების გარდა არის აგრეთვე მულტიმედიური ტიპებიც, როგორიცაა image, graphics და ა.შ., აქამდე ზედაპირულად იყო დახასიათებული. აქვე ნახსენები იყო მომხმარებლისათვის მოუხერხებელი გამოყენების საშუალება გაერთიანების ოპერაციის სახით.

იმისათვის, რომ აქ შედარებით ზუსტი სურათის გადმოცემა მოხერხდეს, საჭიროა წარმოდგენილ იქნას ახალ ტიპებზე განსაზღვრული ოპერაციების სავარაუდო ჩანერგვა (embedding) რელაციურ მოთხოვნის ენაში. ამას გვთავაზობს SQL ენა, რომელიც სტანდარტის სახით დამკვიდრდა მოთხოვნათა ენებისათვის.

განვიხილოთ მარტივი, ორატრიბუტიანი რელაციის მაგალითი:

```
Aerial_image ( Nr integer,
               Picture image )
```

კორტეჟების შესატანად ბაზაში საჭიროა SQL-ენის insert-ოპერატორის გამოყენება. ამ დროს ცალკეული ატრიბუტების მნიშვნელობები გადაეცემა როგორც კონსტანტები ან პროგრამის ცვლადები. მაგალითად,

```
insert into  Aerial_image
values (:nr, image(:pr, :cm ));
```

სადაც pr-ში ინახება სურათი, ხოლო cm-ში ფერთა ცხრილები.

გამოსახულებაში ორი წერტილის შემდეგ დგას პროგრამული ცვლადები, რითაც ისინი სინტაქსურად განსხვავდება რელაციების და ატრიბუტების სახელებისაგან.

value წინადადების პირველი ჩანაწერი ეკუთვნის ატრიბუტის ნომერს და უნდა იყოს integer ტიპის. მეორე ჩანაწერი ეკუთვნის სურათს და ტიპი image. კომპილატორი ცნობს image ტიპის ოპერაციების პარამეტრებისა და შედეგის ტიპებს და შეუძლია შესაბამისი შემოწმების ჩატარება. ეს პრინციპი ძალაშია შემდგომი განხილული ოპერატორებისთვისაც.

თუ კორტეჟი სურათის ატრიბუტებით ერთხელ უკვე შენახულია მონაცემთა ბაზაში, მაშინ შესაძლებელია მათი SQL-ის update ბრძანებით ცვლილება.

```
update  Aerial_image
set  Picture = Picture.replaceColormap(:yuv, 4096, 24, : cm )
where Nr = 1286;
```

შინაარსობრივი მონაცემების მისაღებად შესაძლებელია შემდეგი სახის ბრძანების ჩაწერა:

```
update Aerial_image
set Picture = Picture.newDescr (
      „მოედანს, რომლის ცენტრში ძეგლი და გარშემო\n
        ყვავილებია, უერთდება ხუთი ქუჩა“
 where Nr  = 1234;
```

განსაზღვრული კორტეჟების მოსაძებნად (განსაზღვრული სურათებისთვის) შესაძლებელია ჩვეულებრივი SQL გამოსახულებების გამოყენება. ატრიბუტთა მნიშვნელობების შედარება კონტანტებთან, რომელიც ამ დროს მთავარ როლს თამაშობს, დასაშვებია, უპირველეს ყოვლისა, მხოლოდ სტანდარტული ტიპებისათვის. მულტი-მედიური ტიპებისათვის კი არსებობს სპეციალური შედარების ოპერაციები.

პროგრამაზე გადაცემის დროს image ტიპის ატრიბუტებს არ შეუძლია უშუალოდ პროგრამის ცვლადებზე მინიჭება, რადგან ცვლადთა ტიპები სხვადასხვა დანართში შეიძლება სხვადასხვა იყოს. პირიქით, კომპონენტების ამორჩევა და მასთან დაკავშირებული ტიპების შერჩევა (ადაპტაცია) ხდება ცხადად შესაბამისი image - ოპერაციებით:

```
select Picture.getPixrect(), Picture.getColormap()
into :pr,  :cm
from Aerial_image
where Nr = :k;
```

ამ მაგალითში k-ცვლადში მოცემულია სურათის ნომერი, რომელშიც სურათი (Picture) იქნება გამოძახებული მონაცემთა ბაზიდან Pixrect ფორმატში. შსაძლებელია ასევე სურათის

ატრიბუტების თვისებათა შერჩევა, თუ იგი წინასწარ image-ოპერაციების გამოყენებით მონაცემთა ტიპებისთვის შეიქმნა, რომლებზეც განსაზღრულია შედარების ოპერაციები:

```
select Picture.height(),  Picture.width()
into :hoehe,  :breite
from  Aerial_image
where Picture.pixelcount( :dunkelbraun) < 1000
and Picture.noOfColors() > 4095;
```

შეიძლება ალტერნატიული ვარიანტის განხილვაც image - მონაცეთა ტიპის შედარების ოპერაციისთვის:

```
select Picture.description(), . . .
from Aerial_image
where Picture.contains("ცენტრში ძეგლი");
```

ეს, უპირველეს ყოვლისა, უჩვენებს მომხმარებლის ინტერფეისის სიმარტივეს. ზემოთ აღწერილი select-ბრძანება იძლევა სურათს ნომრით Nr1234, ვინაიდან ამ აღწერაში არის სიტყვები „ცენტრში ძეგლი“ ნახსენები, და იგი როგორც საძებნი გამოსახულება „ცენტრში ძეგლი“, ისე მოიაზრება.

შემდეგ, შენახვისა და მოთხოვნის შემთხვევებში შეიძლება გამოიცეს შეცდომის შეტყობინება, როგორიცაა, მაგალითად, „ამ სიტყვას არ ვიცნობ“ ან „ეს ტექსტი არ მესმის“ და სხვ. ესაა ის ღირებულება, რომლითაც მარტივი ტექსტების შედარებისაგან განსხვავებით მომხმარებლის ინტერფეისი საძებნი პროცედურის უკეთესი ხარისხით გამოირჩევა.

instert – ბრძანებაში დაუშვებელია კომბინაცია values-წინადადებისა (კონსტანტების ან პროგრამის ცვლადების მონაცემები) და ქვეარჩევის (subselect) (ატრიბუტების ახალი მნიშვნელობების შეკრება მონაცემთა ბაზისთან მიმართვის შესახებ). ასეთი პროცედურა უფრო მაშინაა საჭირო, როდესაც სურათის ნაწილი, ამოღებული მონაცემთა ბაზიდან, უნდა იქნას შენახული სხვა კორტეჟში.

ამ ახალი კორტეჟის ფორმატირებული ატრიბუტები მიიღება არა მონაცემთა ბაზიდან, არამედ პროგრამებიდან. მიზანშეწონილია კონსტანტების მიწოდება select-წინადადებაში, რაც არც ძალიან გასაგებად გამოიყურება:

```
insert into Aerial_image
        select  :neueNr, Bild window (50,50,100,100)
        from Aerial_image
        where Nr = :alteNr;
```

update - ბრძანების სემანტიკა, სპეციალური set-წინადადება, არის მნიშვნელობის ჩანაცვლება (აღდგენა), და არა მნიშვნელობის მოდიფიკაცია. მნიშვნელობის მინიჭების კონსტრუქციის მარჯვენა ნაწილში შეიძლება ნებისმიერი სირთულის გამოსახულება იყოს, რომელსაც უნდა ჰქონდეს სწორად შერჩეული შედეგის ტიპი. მაგალითად:

```
update  Aerial_image
apply Picture.replaceColormap ( :cm);
where Nr = 1234;
```

SQL - ბრძანებები პროგრამული ენების სინტაქსის მსგავსად პროცედურების გამოძახებას ჰგავს, და არა ფუნქციურ გამოსახულებას, რომელთაც მნიშვნელობა აქვს. ამიტომაც ბრძანებები, როგორც მაგალითად ქვემოთაა ნაჩვენები, ძალზე მგრძნობიარეა მონაცემთა ახალი ტიპების მიმართ, რათა თავიდან იქნას აცილებული შრომატევადი დუბლირების პროცესები:

```
var := select . . . from . . . where . . . ;
writeScreen(select Picture.pixelmatrix ( ) );
```

მიზანშეწონილია, რომ მეთოდები აღიწეროს დეტალურად. ეს ნიშნავს იმას, რომ მოხდეს განსხვავება წასაკითხსა და ჩასაწერს შორის, და მულტიმედიური კონტექსტის თვალსაზრისით, უპირველეს ყოვლისა, განასხვავონ დროზე დამოკიდებული და დამოუკიდებელი პროცესები.

რელაციური მონაცემთა ბაზები, მოთხოვნების დამუშავების თვალსაზრისით, წლების განმავლობაში ასრულებს განსაკუთრებულ როლს, სხვა ახალი მონაცემთა ბაზებისაგან განსხვავებით. მათი გაფართოება ახალი ტიპის, მულტიმედიალური მონაცემებით ძალზე ეფექტურია და პრაქტიკულად ღირებული. მომხმარებელს, მათი გამოყენების მიზნით, სჭირდება ამ მიმართულებით დამატებითი ცოდნის მიღება.

2.4. ობიექტორიენტირებული მულტიმედიური მონაცემთა ბაზის სისტემები

პირველი ნაშრომები მულტიმედიური მონაცემთა ბაზების შესახებ გამოჩნდა 80-ან წლებში, როდესაც მონაცემთა რელაციური მოდელების თემატიკა, როგორც ახალი მეცნიერული მიმართულება, ძალზე აქტუალური და დომინირებადი იყო [43-46].

გასაოცარი არაა, რომ ამ დროს მულტიმედიური მონაცემთა ბაზების სისტემების პირველი პროექტები მოიაზრებოდა როგორც აუცილებლად ობიექტ-ორიენტირებული სისტემები [41,47]. მაგრამ ამ პერიოდში ვერ მოხერხდა ასეთი სისტემების პრაქტიკული რეალიზაცია.

ამ პრობლემებზე მუშაობა და მნიშვნელოვანი მიღწევები მიღებულ იქნა 90-ანი წლებიდან, როდესაც ODMG (Object Data Management Group) ჯგუფმა წარმოადგინა რეალაციური ბაზებისათვის SQL-ის მსგავსი ეფექტური კონცეფცია [48].

- **მონაცემთა მულტიმედიური ტიპების ჩაშენება**

ობიექტორიენტირებულ სისტემებში მულტიმედიური მონაცემთა ტიპების ასახვა ხდება უშუალოდ როგორც კლასები. ეგზემპლარები უნდა იყოს ინკაფსულირებული, ისე, რომ წვდომის განხორციელება შეიძლებოდეს მხოლოდ კლასის მეთოდებზე.

როგორც ობიექტრელაციურ მოდელში, ეს კლასები შეიძლება იყოს ორგანიზებული განზოგადებული (გენერალიზებული) იერარქიით. ისინი გამორიცხავს საერთო მეთოდების განმეორებად განსაზღვრებებს და ნებას იძლევა მონაცემთა სპეციალური ტიპების შესატანად, რომლებისთვისაც დამატებითი მეთოდებია განსაზღვრული.

მონაცემთა ტიპები Text, Image, ასევე Graphics, Audio და Video განიხილება როგორც MediaObject-ის ქვეკლასები, და მემკვიდრეობით იღებს მის მეთოდებს. ისინი აფართოებენ ამ მეთოდებს საკუთარი ოპერაციებით, რომლებიც მორგებულია სპეციალურ მულტიმედიურ მონაცემთა ტიპებს, მაგალითად, image-ს დროს height, ან Text-ის დროს length.

ORION სისტემა ან MIM (Multimedia Information Manager)-ით აღნიშნული კლასების კოლექცია წარმოადგენს კიდევ ერთ შრეს MediaObject-სა და Text ან Image ტიპებს შორის, რომლების შეიცავს კლასებს „წრფივი მედიაობიექტები“ და „სივრცითი მედიაობიექტები“. პირველი მოიცავს ტექსტებს და აუდიოს, ხოლო მეორე სურათებს, გრაფიკას და ვიდეოს [36]. მართლაც, ამ მეთოდებს, როგორიცაა length და height, შეუძლია ფაქტობრივად განსაზღვრულ იქნას ამ კლასებზე და შემდეგ მემკვიდრეობით იქნას გამოყენებული. სხვა მხრივ უფრო მნიშვნელოვნად ჩანს კლასიფიკაცია როგორც „დროზე დამოკიდებული“ და „დროზე დამოუკიდებელი“. ამიტომ გადაწყვეტა ასეთი შუალედური შრის შესახებ ჯერჯერობით არ განიხილება.

კლასების იერარქიის გენერალიზაცია და აგება საშუალებას იძლევა მოვახდინოთ არა მხოლოდ მულტიმედიური ობიექტების სასარგებლო ჩანერგვა, არამედ აგრეთვე ობიექტებისაც (Entities), რომლებსაც ისინი ასახავს ან აღწერს.

წინა პარაგრაფში ობიექტრელაციური მოდელების დისკუსიამ გვიჩვენა, რომ სათანადო შემთხვევა სხვადასხვა is.presented_in დამოკიდებულება ამოდელირებს და ეს ძებნის დროს ყველა ცხადად უნდა იქნას გათვალისწინებული.

გენერალიზაცია, პირიქით ნებას იძლევა, რომ ეკომაჩვენებლის სპეციალისტი, ეკომონიტორინგის სპეციალისტი და ქიმიური_ლაბორატორიის_თანამშრომელი ზემნიშვნელობით (Superclass) გავაერთიანოთ, როგორიცაა „თანამშრომელი“ (ან „პიროვნება“) და მათი ეგზემპლარები სურათებით დავაკავშიროთ. რომელი ზემნიშვნელობა მიესადაგება დასმულ მიზანს, დამოკიდებულია იმაზე, თუ რომელი ობიექტებია სურათებზე, ტექსტებში, გრაფიკებზე და ა.შ. უფრო რელევანტური (შესაბამისი) გამოსაყენებლად. ყველაზე ზოგად შემთხვევაში, შეძლებისდაგვარად, თუ მოცემულია პრესიდან სურათი (ფოტო), საჭიროა ისევ „Object“ კლასზე მიმართვა.

ობიექტორიენტირებულ მოდელს შეუძლია მე-3 ტიპის სქემის (იხ. წინა პარაგრაფი) უკეთესად წარმოდგენა, ვიდრე რელაციურ მოდელს. როგორი მდგომარეობაა სქემის სხვა ტიპებისათვის კლასის ეგზემპლარები გამოისახება კორტეჟების სახით ატრიბუტების მნიშვნელობებზე, რომელთა მნიშვნელობათა არეები დადგენილია კლასების განსაზღვრებაში. კლასიკური რელაციური მოდელისაგან განსხვავებით მნიშვნელობათა ეს

არეები არ უნდა იყოს არჩეული ერთი წინასწარგანსაზღვრული სიმრავლიდან, არამედ შესაძლებელია ასევე იყოს ნებისმიერი თვითგანსაზღვრებადი კლასი.

მაგალითად, employee კლასს შეუძლია თავისი ეგზემპლარებისათვის განსაზღვროს ატრიბუტი (ეგზემპლარის ცვლადი) Portrait, რომელთა მნიშვნელობები შეიძლება იყოს GreyscaleImage კლასის ეგზემპლარები. ეს შეესაბამება სქემის 1-ელ ტიპს რელაციური ბაზისათვის. ობიექტორიენტირებული სისტემები არ მოითხოვს ნორმალიზაციას თავისი კლასებისა და ეგზემპლარებისათვის, ასე რომ ნებადართულია ატრიბუტები რამდენიმე მნიშვნელობით. აუცილებლობის შემთხვევაში საჭიროა ტიპების კონსტრუქტორის, როგორიცაა list ან set გამოყენება. შესაბამისად იგი მე-2 ტიპის სქემასთან შედარებით ოდნავ ჭარბია. Portrait ატრიბუტს შეუძლია აგრეთვე მარტივად ჰქონდეს რამდენიმე სურათი.

მრავალი ობიექტორიენტირებული სისტემა არ განსხვავდება სუფთად ობიექტის ატრიბუტებითა და კომპონენტებით. ხშირად აგრეგაცია გამოისახება მარტივად ატრიბუტთა დახმარებით. ODMG-მოდელი მკაფიოდ გამოყოფს ატრიბუტებს დამოკიდებულებებისაგან და თუ ატრიბუტებს შეუძლია მხოლოდ მნიშვნელობების მიღება, ობიექტები უკავშირდება ერთმანეთს დამოკიდებულებებით.

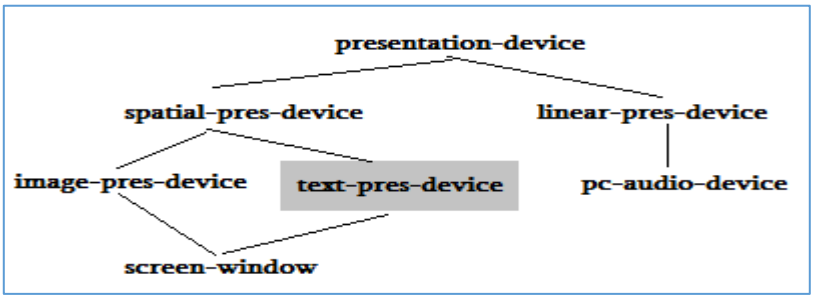
ობიექტორიენტირებული სისტემების სუსტი ადგილია სიმრავლეებზე ორიენტაციის არარსებობა და მასთან დაკავშირებული ძებნა. ეს გასაგები იყო, ამიტომაც ცდილობდნენ ობიექტორიენტირებული ბაზების მართვის სისტემებისათვის მოთხოვნების ენის განსაზღვრას [49].

OMDG-მოდელში არსებობს ამისთვის OQL-ენა (Object Query Language), რომელიც ორიენტირებულია SQL-ზე და ზოგიერთ შემთხვევაში სრულად თავსებადიცაა. მას მოაქვს თან ყველა სასურველი თვისება, მაგრამ არ შეესაბამება ობიექტებზე პირველად წვდომას, რისთვისაც იყო შექმნილი ოო-ბაზების სისტემები: იგულისხმება ირიბი მიმართვა. ვინაიდან OQL ენა ძალზე რთულია, მისი გამოყენება ნაკლებად გვხვდება პროგრამულ პროდუქტებში.

- **მულტიმედიური ინფორმაციული მენეჯერი (MIM)**

ობიექტორიენტირებული მულტიმედიური მონაცემთა ბაზის სისტემების კონკრეტული მაგალითი განვიხილოთ ORION სისტემის მაგალითზე, რომელიც ასევე ცნობილია მულტიმედიალური ინფორმაციული მენეჯერის (MIM – Multimedia Information Manager) სახელით [36,49-51]. ესაა ობიექტ-ორიენტირებული კლასების ბიბლიოთეკა. ასეთი ობიექტორიენტირებული მონაცემთა ბაზების სისტემა მომხმარებლისთვის აადვილებს მულტიმედიური ობიექტები დაიყვანონ სუბკლასებამდე და შემდეგ ისარგებლონ MIM ბიბლიოთეკის ოპერაციებით მემკვიდრეობითობის მექანიზმის გამოყენებით [49].

MIM-ის განსაკუთრებული თავისებურება მდგომარეობს იმაში, რომ ყველა მოწყობილობა წარმოიდგინება როგორც ობიექტები ORION-ში, კერძოდ, როგორც შეტანა/გამოტანის, ასევე მეხსიერების მოწყობილობები. გამოტანის მოწყობილობისთვის მომზადებულია კლასების იერარქია, რომელიც 2.2 ნახაზზეა მოცემული.



ნახ.2.2. MIM-ში კლასთა იერარქია გამოსატანი მოწყობილობებისთვის

გრაფის წიბოები ასახავს მემკვიდრეობითობის კავშირებს, სადაც სუბკლასები მოთავსებულია კლასების ქვეშ. მონიშნული კლასები text-pres-device არის მაგალითი გაფართოებისა, რომელიც თვით მომხმარებელმა აირჩია. ასეთი კლასი არ ეკუთვნის MIM-ბიბლიოთეკას.

კლასთა ეგზემპლარები ამ იერარქიაში არაა განაწილებული ცალსახად გამომტანი მოწყობილობებისათვის, არამედ ისინი სპეციფიცირებულია:

- სადაა მოწყობილობაზე ასახული (მაგალითად, ეკრანის რომელ ნაწილში);
- მედიალური ობიექტის რომელი ნაწილი (ამონაჭერი) არის ასახული.

შედეგად შესაძლებელია რამდენიმე ეგზემპლარის მიცემა, რომლებიც ერთი და იმავე ფიზიკურ მოწყობილობას ასახავს.

შეიძლება ასევე მათი ინტერპრეტაცია შეზღუდვებით როგორც „გამოცემის ფორმატი“ მედიური ობიექტებისათვის. მაგალითად, კლასი spatial-pres-device ასე განმარტავს მისი ეგზემპლარის შემდეგ ატრიბუტებს:

upper-left-x,
upper-left-y,
width,
height.

ამ ატრიბუტების მნიშვნელობები შეესაბამება მედიური ობიექტის ამონაჭერს (მაგალითად, რასტრულ სურათს), რომელიც უნდა გამოიცეს სივრცით გამომტან მოწყობილობაზე.

ქვეკლასში screen-window სპეციფიცირდება ატრიბუტები:

win-upper-left,
win-upper-right,
win-width,
win-height

დამატებით ეკრანის ნაწილზე, რომელიც უნდა იქნას გამოყენებული ასახვისთვის.

screen-window -სთვის განსაზღვრულია ასევე მეთოდები, კერძოდ:

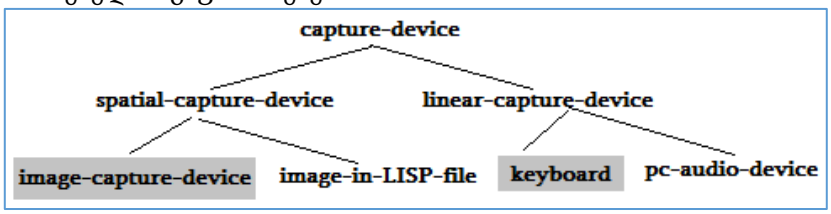
present,

capture,

present-pres,

რომლებიც გამოიყენება განსაზღვრული მულტიმედიური ობიექტების გამოსატანად და, აუცილებლობის შემთხვევაში, გამოიძახება წაკითხვის აღსადგენადაც.

ანალოგიურად არის MIM-ში განსაზღვრული კლასთა იერარქია შესატანი მოწყობილობებისთვის (ნახ.2.3). აქ მონიშნული კლასები, image-capture-device და keyboard აღწერეს მომხმარებელთა გაფართოებებს.



ნახ.2.3. MIM-ში კლასთა იერარქია შესატანი მოწყობილობებისათვის

აქაც, როგორც წინა შემთხვევაში, კლასის ეგზემპლარები მეტია, ვიდრე მხოლოდ სპეციფირებული მოწყობილობები. ისინი ასევე მიუთითებს თუ მულტიმედიური ობიექტის რომელი ნაწილი განიხილება და როგორაა მოწყობილობა დაკომპლექტებული. შედეგად შესაძლებელია ერთი ფიზიკური მოწყობილობისათვის კვლავ capture-device ტიპის რამდენიმე ეგზემპლარის მიცემა.

spatial-capture-device სუბკლასის ეგზემპლარები მიუთითებს შემდეგ ატრიბუტებს:

upper-left-x,

upper-left-y,

width,

height.

იგი სპეციფიცირებას უკეთებს სივრცითი მულტიმედიური ობიექტის ამონაჭერს, რომელიც მიიღება შემტანი მოწყობილობით ან ჩანაცვლებით. მომხმარებლის მიერ განსაზღვრულ image-capture-device კლასს შეუძლია

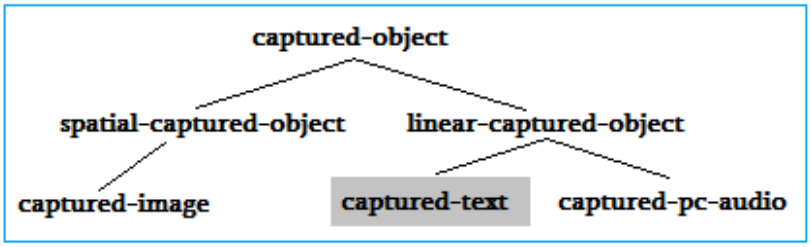
cam-width,

cam-height,

bits-per-pixel

ატრიბუტების და capture მეთოდის წარმოდგენა.

დამახსოვრებული მულტიმედიური ობიექტები ორგანიზებულია ასევე კლასთა იერარქიის სახით. ამ დროს კვლავ განასხვავებენ სივრცით და წრფივ მულტიმედიურ ობიექტებს. რასტრული სურათები და გრაფიკები მიეკუთვნება სივრცითს, ხოლო ტექსტი და აუდიო კი წრფივს. ქვეკლასები ასახულია 2.4 ნახაზზე.



ნახ.2.4. MIM-ში დამახსოვრებული მედიური ობიექტების კლასთა იერარქია

captured-object კლასებში ყველა ქვეკლასისთვის განსაზღვრულია შემდეგი ატრიბუტები:

storage-object – მიუთითებს storage-device კლასის ერთ ეგზემპლარზე, რომელიც ქვემოთ შემოიტანება.

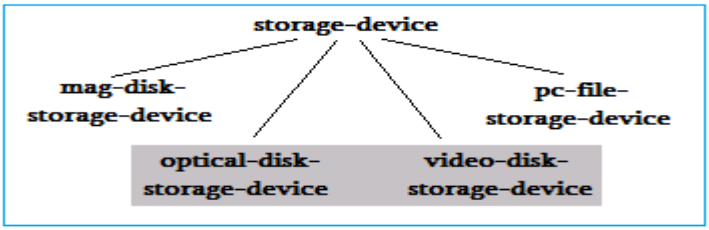
logical-measure - განსაზღვრავს საწყისი მონაცემების ელემენტარულ ერთეულებს მომხმარებლის თვალსაზრისით, რომლებიც არ უნდა ემთხვეოდეს სისტემტექნიკურ ერთეულებს (ბიტი, ბაიტი ან მეხსიერების სიტყვა (memory word)). ასეთი ლოგიკური ზომის ერთეულებია, მაგალითად, წამები აუდიოს დროს, ან კადრები - ვიდეოს დროს.

დამოკიდებულება ფიზიკურ და ლოგიკურ ზომის ერთეულებს შორის აისახება phys-logic-ratio ატრიბუტში. ეს იქნება მაშინ ბაიტი/წამში აუდიოსათვის ან ბაიტი/კადრი ვიდეოსათვის.

spatial-captured-object სუბკლასი იძლევა ატრიბუტებს width, height და row-major. უკანასკნელი გვიჩვენებს, დამახსოვრება მოხდა სტრიქონულად თუ სვეტურად.

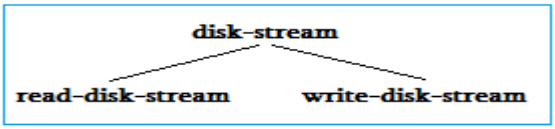
და ბოლოს, არსებობს ატრიბუტი bits-per-pixel, რომელიც ცხადყოფს, რომ საქმე ეხება ტიპურ სარეგისტრაციო მონაცემებს.

დამახსოვრებისთვის არსებობს მოწყობილობები, რომლებიც შესატან და გამოსატან მოწყობილობათა მსგავსად ნაწილობრივ გამოყენებაშია და storage-device კლასის ეგზემპლარების საშუალებით წარმოიდგინება (ნახ.2.5).



ნახ.2.5. MIM-ში შენახვის მოწყობილობების კლასთა იერარქია

და ბოლოს, კლასებში არსებობს კიდევ ორგანიზებული მონაცემთა ობიექტები, რომლებიც ასახავს ცალკეულ წაკითხვის ან ჩაწერის პროცესებს (ნახ.2.6).



ნახ.2.6. MIM-ში წაკითხვის და ჩაწერის პროცესების კლასთა იერარქია

მათი წარმოება ხდება დინამიკურად და შეტანის ან გამოტანის დამთავრების შემდეგ ისევ იშლება. ამავდროულად, disk-stream შეიცავს ორივე შემთხვევისათვის საჭირო storage-object ატრიბუტს, რომელიც მიმართავს storage-device კლასის წასაკითხ ან ხელახლაჩასაწერ ეგზემპლარს. read-disk-stream -ში არსებობს დამატებთ read-block-list ატრიბუტი, რომელც აღწერს პოზიციონირების მარკირებასა და აიდენტიფიცირებს მულტიმედიური ობიექტის მომდევნო წასაკითხ ბლოკს. მსგავს როლს ასრულებს write-block-list ატრიბუტი write-disk-stream -ში.

ამჯერად ილუსტრირებული გვაქვს აღნიშნული კლასების ურთიერთმოქმედება და მეთოდების ჩადგმული გამოძახება რასტრული სურათის გამოცემის მაგალითზე.

დავუშვათ, რომ არსებობს car კლასი (ავტომობილი), რომელიც თავისი ეგზემპლარებისათვის განსაზღვრავს ატრიბუტებს image (სურათი) ტიპით captured-image და output_ device (გამოსატანი მოწყობილობა) ტიპით image-pres-device.

ავტომანქანის აღწერას ეკუთვნის აგრეთვე გამოსატანი მოწყობილობა, რომელზეც მანქანის სურათი კარგად უნდა აისახოს. ამის გარდა ეს კლასი შეიცავს ასევე მეთოდს show_image (სურათის ჩვენება), რომელიც ახორციელებს დამახსოვრებული სურათის გამოცემას წინასწარგანსაზღვრულ გამოსატან მოწყობილობაზე. ამის მისაღწევად, საჭიროა show-image მეთოდმა გააგზავნოს present შეტყობინება ობიექტისაკენ, რომელიც წარმოადგენს გამოსატან მოწყობილობას და მას ერთ პარამეტრში დაუსახელოს გამოსატანი სურათი. თუ ასეთი გადაცემის ან გამოძახების ბრძანებები LISP-ენის სახის მეთოდოლოგიის სინტაქსით აიგება, მიმღები ობიექტი და პარამეტრი სპეციფიცირდება, მაშინ show-image-ს შესაბამისი ბრძანება შეიძლება ასე ჩაიწეროს:

(present output-device Image)

output-device ატრიბუტით იდენტიფიცირებული image-pres-device კლასის ეგზემპლარი ასრულებს ამის შემდეგ მის present მეთოდს. მისი ატრიბუტები upper-lef-x, upper-lef-y, width და height უთითებს სურათის რომელი ნაწილი (ამონაჭერი) იხილება. ეს მართკუთხა ამონაჭერი გაითვლება წრფივ კოორდინატებში. ეს შესაძლებელია მხოლოდ captured-image ეგზემპლარზე წვდომის საშუალებით, რომელიც იდენტიფიცირებულია image პარამეტრით, რადგან აქ row-major ატრიბუტი მხოლოდ გვეუბნება, რომ რეგისტრაციის მონაცემები ხელმისაწვდომია. ამჯერად შენახული სურათი წასაკითხად იხსნება:

(open-for-read Image [start-offset])

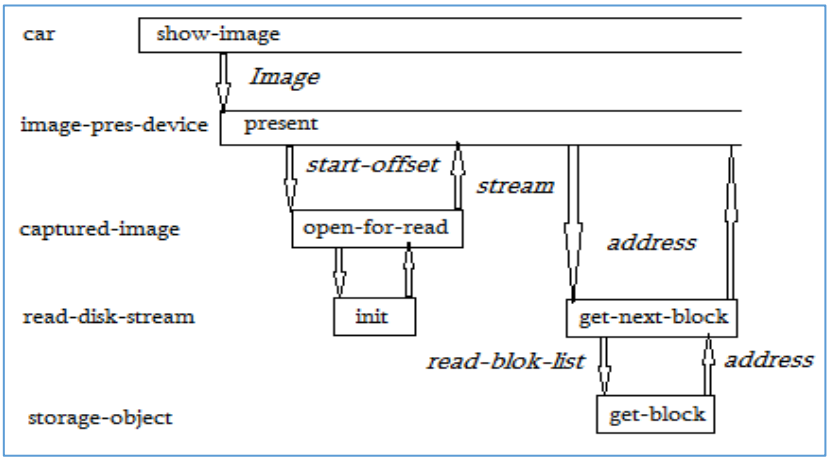
Image-ს საშუალებით დასახელებული capture-image ეგზემპლარი ასრულებს open-for-read მეთოდს. ამ დროს იგი აწარმოებს ახალ read-disk-stream ეგზემპლარს, რომლის სახელს (ობიექტის იდენტიფიკატორს) იგი image-pres-device -ს უკან უგზავნის. თუ შესაძლებელია, სპეციფიკაცია start-offset ემსახურება read-block-list -ის ინიციალიზაციას. გამოსატანი მოწყობილობა აგზავნის ამჯერად (present-ის შესრულების გაგრძელება) ამ ნაკადისთვის (straem) შეტყობინებას წაკითხვის შესახებ:

(get-next-block read-disk-stream)

შემდეგ ნაკადი (straem) თვითონ აგზავნის ისევ:

(get-block storage-object read-block-list)

ამ გამოძახებით მიიღება ბუფერის მისამართი, სადაც ინახება სასურველი ბლოკი. შემდეგ ნაკადი ზრდის მაჩვენებელს და აწვდის image-pres-device–ს უკან ბუფერის მისამართს, როგორც get-next-device მეთოდის შედეგს. 2.7 ნახაზი ასახავს ზემოაღწერილ პროცესს გრაფიკულად. მარცხენა მხარეს შემოტანილია კლასის სახელები, რომელთაგან თითოეული ზუსტად ერთ ეგზემპლარში მონაწილეობს. მართკუთხედები აღნიშნავს მეთოდის შესრულებას, რომლებიც დახრილი (italic) სიმბოლოებით აღწერილი პარამეტრებით გამოიძახება და ასევე კურსივით აღწერილ მნიშვნელობებს აბრუნებს.



ნახ.2.7. MIM-მეთოდების ჩადგმული გამოძახება სურათის გამოცემის დროს

შემდეგი მსვლელობისას გამომტანი მოწყობილობა გადასცემს წაკითხული ბლოკის შინაარსს ტექნიკურ უზრუნველყოფას და აცნობებს ნაკადს, რომ ბუფერის სახელები კვლავ ხელმისაწვდომია:

(free-block read-disk-stream)

სურათის ყველა ბლოკის წაკითხვისა და გათავისუფლების შემდეგ (close-read read-disk-stream)-ით დაიხურება წაკითხვის პროცესი. ნაკადი შეიძლება კვლავ წაიშალოს.

captured-object კლასი უზრუნველყოფს თავისი ეგზემპლარებისათვის კიდევ სხვა მეთოდებს. make-captured-object-version-ის საშუალებით იწარმოება შენახული მულტიმედია ობიექტის ახალი ვერსია და ირიბად ასევე დაკავებული მეხსიერების მოწყობილობის ახალი ვერსია (storage-device). ძველი და ახალი ვერსიები თავიდან იკავებს დისკზე ერთსა და იმავე ბლოკებს.

დასასრულ, არსებებოს კიდევ delete-captured-object და delete-part-of-captured-object მეთოდები. უკანასკნელი ელოდება პარამეტრებს start-offset და delete-count, რომლებიც მიუთითებს, რომელი ბაიტ - პოზიციიდან და რამდენი ბაიტი უნდა იქნას წაშლილი. ეს ოპერაცია მოითხოვს სიფრთხილეს, ვინაიდან იგი იმავდროულად არ სრულდება სარეგისტრაციო მონაცემების ცვლილებით. ასეთი ოპერაცია არაა გათვლილი სისტემის საბოლოო მომხმარებელზე.

ამგვარად, ORION სისტემა MIM-თან ერთად ასახავს ყველაზე სრულყოფილ წინადადებას დღემდე არსებულ მულტიმედიურ მონაცემთა ბაზების მართვის სისტემებს შორის. იგი მომხმარებელს სთავაზობს დიდ მოქნილობას, რათა არსებული კლასების იერარქია საკუთარი სუბკლასების სპეციალიზებით გაფართოვდეს, რაც დამატებითი დამუშავებისა და წვდომის ოპერაციების გამოყენების შესაძლებლობას იძლევა [36].

ამგვარად, ობიექტრელაციური და ობიექტორიენტირებული მონაცემთა ბაზების მართვის სისტემების შედარება გვიჩვენებს, რომ ზოგადად არსებობს ერთიანობა მათში ახალი ტიპების განსაზღვრისა და მართვისათვის, ან ობიექტორიენტირებული სისტემის კონტექსტში (მაგალითად, ORION), ან რეალიურ სისტემაში (მაფალითად, SQL / MM).

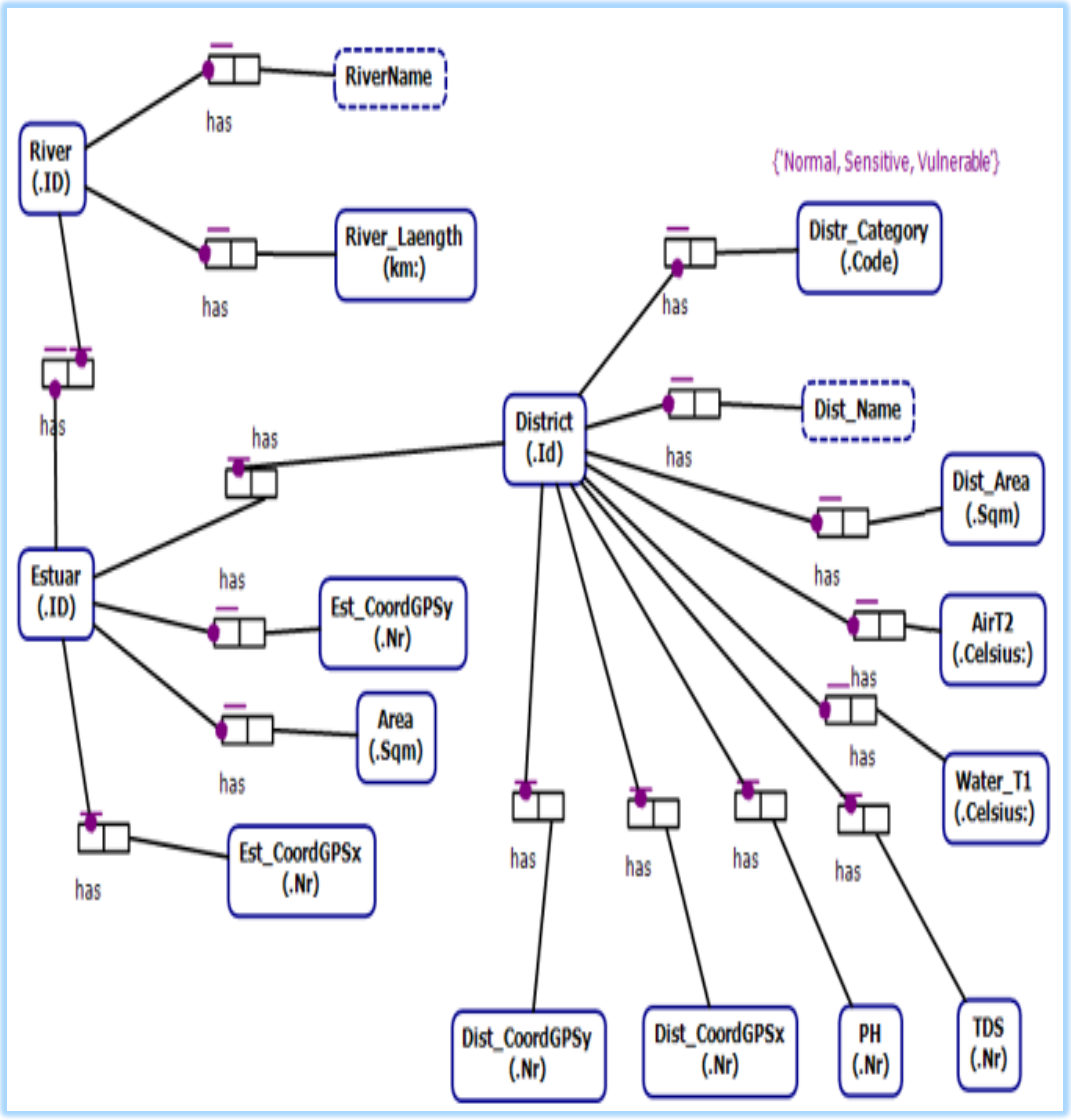
2.5. მულტიმედიური რელაციური ბაზის ოპტიმალური სტრუქტურის დაპროექტება

შავი ზღვის ეკომინიტორინგის კომპიუტერული სისტემის მონაცემთა რელაციური ბაზის კონცეპტუალური სქემის დასაპროექტებლად ჩვენ გამოვიყენეთ ობიექტ-როლური მოდელირების მეთოდი და ინსტრუმენტული საშუალება (NORMA). ავტომატიზებული პროცესი მომხმარებლის მიერ სემანტიკუი ფაქტების აღწერით იწყებოდა, რომლის საფუძველზე ფორმირდებოდა კონცეპტუალური სქემა (1-ელი დონე) ORM დიაგრამის სახით (ნახ.2.8), ობიექტებით, პრედიკატებით (კავშირები ობიექტებს შორის), ატრიბუტებით და მათი მნიშვნელობებით [15].

ვინაიდან საპრობლემო სფეროს აღწერა სისტემური ანალიტიკოსის, ან საბოლოო მომხმარებლის მიერ ხდება (ან ორივეს თანამშრომლობით), სემანტიკური ფაქტების სიმრავლე, შემდეგ ORM დიაგრამა და ბოლოს, ERM სქემა შეიძლება იყოს განსხვავებული. ანუ მიიღება ეკვივალენტური კონცეპტუალური სქემები. რომელია მათ შორის ოპტიმალური, ან უკეთესი სისტემის მონაცემთა ბაზის საბოლოო რეალიზაციისათვის?

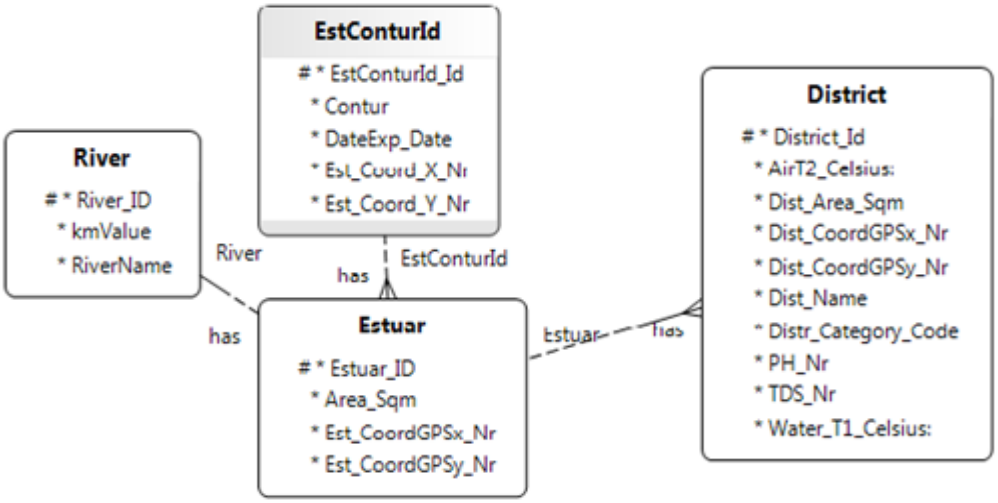
შესაძლებელია თუ არა დაპროექტების წინა სტადიებზე განისაზღვროს უკეთესი ვარიანტი და გამოირიცხოს არაპერსპექტიული ვარიანტები? როგორ შევაფასოთ მოსალოდნელი შედეგი წინასწარ?

ამგვარად, წინამდებარე პარაგრაფში გვინდა წარმოვადგინოთ ამ საკითხის გადაჭრის გზა, მიდგომა და ფორმალიზებული ალგორითმული სქემები.



ნახ.2.8. კონცეპტუალური სქემა ORM დიაგრამით

2.9 ნაჩვენებია მიღებული ORM დიაგრამის ეკვივალენტური კონცეპტუალური (მე-2 დონე) ERM სქემა. იგი აგებულია ბარკერის დიაგრამის სახით.



ნახ.2.9. ER-მოდელის ოთხი ცხრილით (ბარკერის დიაგრამა)

ER-მოდელის ცხრილების ოპტიმალური შემადგენლობის განსაზღვრის ამოცანა კავშირშია რელაციური ბაზის სტრუქტურის ნორმალიზაციის საკითხთან [52,53].

განვიხილოთ ახლა ამ ამოცანის გადაწყვეტა, შევიმუშავოთ მისი ალგორითმი და პროგრამული მოდული.

დავუშვათ, მოცემულია საპრობლემო სფეროს აღწერის ატრიბუტთა $U=\bigcup_i U_i$ სიმრავლე. მონაცემთა ბაზის $F^0(\overline{A})$ საწყისი სქემა შეიძლება გამოვსახოთ უნივერსალური დამოკიდებულების სახით: $\overline{S}^0=\{\overline{R}=<U,P>\}$, სადაც $\overline{R}$ დამოკიდებულებათა უნივერსუმია, ხოლო P სემანტიკურ შეზღუდვათა კლასები ან ფუნქციონალურ დამოკიდებულებათა (ფდ) ერთობლიობა [59, 66-68].

ამოცანის მიზანია $\overline{S}^0$ სქემის ეკვივალენტური $\overline{S}$ სქემის კონსტრუირება $\overline{S}=\{R_i=<U_i,P_i>\}$, სადაც R_i არის $\overline{R}$ -ის პროექცია, ხოლო P_i ეთანადება ფდ-ის განსაზღვრულ კლასს, მაგალითად, ფდ, სრული-ფდ (სფდ), ტრანზიტული-ფდ (ტფდ), ფსევდოტრანზიტული-ფდ (ფტფდ), მრავალსახა (მსდ), ზოგადი არაფუნქციონალური (ზად).

ნორმალურ ფორმათა (ნფ) თეორიის გამოყენებით ხორციელდება უნივერსუმის მიმდევრობითი დეკომპოზიცია შემდეგი სქემით:

ანფ -> 1ნფ -> 2ნფ -> 3ნფ -> 4ნფ -> 5ნფ -> . ? . -> ბნფ ,

სადაც ანფ - არანორმალიზებული ფორმაა, 1ნფ - პირველი ნფ, . . . , ბნფ - ბინარული ნფ.

ძირითადი სქემის განშტოებაში შეიძლება განვიხილოთ ბოის-კოდის ნფ (3ნფ-დან), პირველი რიგის იერარქიული დეკომპოზიცია (4ნფ-დან), ურთიერთდამოკიდებულებანი

(5ნფ-დან) და ა.შ. ნიშანი „ ? " მიუთითებს იმაზე, რომ დამოკიდებულებათა თვისებების კვლევა გრძელდება, მათი შემდგომი ოპტიმიზაციის მიზნით.

დღეისათვის მრავლად არსებობს სხვა სახის ნფ-ებიც, მაგრამ ნორმალიზაციის კლასიკურ თეორიაში ისინი ნაკლებადაა ასახული [34,54,55].

დამოკიდებულებათა დეკომპოზიციის დროს ბნფ-ები მიიღება სქემის დაპროექტების ცალკეულ ეტაპზე. არადეკომპონირებადი დამოკიდებულებისთვის კი საჭიროა ხელოვნურად ფიქტიური ატრიბუტის (ნატურალურ რიცხვთა სასრული სიმრავლე) შემოტანა. ინფორმაციის სემანტიკური მთლიანობის უზრუნველყოფა დეკომპონირებულ დამოკიდებულებათა შორის ხორციელდება ინდექსური კავშირების დუბლირების საშუალებით, ე.ი. შემოტანილია გარკვეული სიჭარბე. რაც მაღალია ნფ-ის რიგი, მით ნაკლებია ინფორმაციული და მეტია ინდექსური სიჭარბე:

1nf->2nf->. . . ->bnf, 1nf -> 2nf ->. . . ->bnf,

V1-inf>=V2-inf>=...>=Vb_inf, V1-ind<=V2-ind<=...<=Vb-ind

აქედან გამომდინარე, დაისვა ოპტიმალური სიჭარბის განსაზღვრის კომპრომისული ამოცანა განახლების დროის მინიმიზაციის მოთხოვნით. ამოცანის გადაწყვეტის შედეგად შესაძლებელი გახდა მონაცემთა ბაზის სქემის დამოკიდებულებათა ოპტიმალური ნფ-ების დადგენა, რის შემდეგაც ავტომატიზებულად დაპროექტდება ნორმალურ ფორმათა სტრუქტურები.

დავუშვათ, რომ მოცემულია სემანტიკურად თავსებადი ფდ-ების სიმრავლე:

$$\begin{cases} R_1(k_1, k_2, \ldots k_{n1}, A_1, A_2, \ldots, A_{a1}) \\ R_2(k_1, k_2, \ldots k_{n2}, B_1, B_2, \ldots, B_{a2}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_l(k_1, k_2, \ldots k_{nl}, Z_1, Z_2, \ldots, Z_{al}) \end{cases} \quad (2.1)$$

სადაც k ატრიბუტების გასაღებური, ხოლო A-:-Z არაგასაღებური ნაწილებია.

სემანტიკური მთლიანობა შედეგია იმ ფაქტისა, რომ სიმრავლე მიღებულია ერთი უნივერსუმის დეკომპოზიციით. თუ ჩავთვლით, რომ

$$k_1, k_2, \ldots k_{n1} \supseteq k_1, k_2, \ldots k_{n2} \supseteq k_1, k_2, \ldots k_{nl},$$

(1) სისტემის კომპოზიციით, მაშინ შესაძლებელია ერთი შედარებით დაბალი ნფ-ის მიღება:

$$R(k_1 \ldots k_{n1}, A_1 \ldots A_{a1}, B_1 \ldots B_{a2}, \ldots Z_1 \ldots Z_{al}) \quad (2.2)$$

დავუშვათ აგრეთვე, რომ წინასწარ ცნობილია R_i-ის ცვლილების რაოდენობა μ_i დროის განსაზღვრულ ინტერვალში და მართებულია შემდეგი მოწესრიგება.

$$\mu_1 \geq \mu_2 \geq \ldots \geq \mu_l$$

(1) და (2) გამოსახულებებისთვის განახლებათა მოცულობები შესაბამისად გამოითვლება შემდეგნაირად:

$$Q_{dec} = \sum_{i=1}^{l} \mu_i * (n_i + a_i) \text{ da } Q_{com} = \mu_1 * (n_1 + \sum_{j=1}^{l} (a_j - r))$$

სადაც r ატრიბუტების ის რაოდენობაა, რომლითაც სრულდება შეერთების („Join“) ოპერაცია. შემდგომში შეიძლება იგნორირება.

თუ დავუშვებთ, რომ (1) და (2) ნფ-ებს შორის არსებობს შუალედური ნფ, მაშინ მისთვის განახლებათა მოცულობა შეადგენს:

$$Q = \sum_{j=1}^{s} \mu_j * (n_j + \sum_{k=1}^{l} a_k)$$

სადაც S ფდ-ების რაოდენობაა შუალედურ ნფ-ში. მართებულია შემდეგი უტოლობა:

$$\mu_1 * (n_1 + \sum_{k=1}^{l} a_k) \geq \ldots \geq \sum_{j=1}^{s} \mu_j * (n_j + \sum_{k=1}^{l} a_k) \geq \ldots \geq \sum_{i=1}^{s} \mu_i * (n_i + a_i) \quad (2.3)$$

სადაც უტოლობის მარცხენა მხარე ეთანადება (i-1)ნფ-ს, ნაპირა მარჯვენა მხარე - (i+1)ნფ-ს, ხოლო ცენტრალური - i ნფ-ს, სადაც $i >= 4$.

გავაანალიზოთ დეტალურად ორი მოსაზღვრე ნფ, მაგალითად, i და i+1. (3)_დან შეიძლება მივიღოთ:

$$\sum_{j=1}^{s} \mu_j n_j + \sum_{j=1}^{s} \sum_{k=1}^{l} \mu_j a_k \geq \sum_{i=1}^{l} \mu_i n_i + \sum_{i=1}^{l} \mu_i a_i \quad (2.4)$$

აქედან მართებულია შემდეგი გამოსახულებანი:

$$\sum_{i=1}^{l} \mu_i n_i - \sum_{j=1}^{s} \mu_j n_j = \sum_{i=s+1}^{l} \mu_i n_i \quad (2.5)$$

$$\sum_{j=1}^{s} \sum_{k=1}^{l} \mu_j a_k - \sum_{i=1}^{l} \mu_i a_i = \sum_{j=1}^{s} \sum_{k=1, j \neq k}^{l} \mu_j a_k - \sum_{i=s+1}^{l} \mu_i a_i \quad (2.6)$$

თუ (5) და (6) ტოლობათა მარჯვენა ნაწილებს ჩავსვამთ (4)-ში, მივიღებთ:

$$\sum_{j=1}^{s} \sum_{k=1, j \neq k}^{l} \mu_j a_k \geq \sum_{i=s+1}^{l} \mu_i n_i + \sum_{i=s+1}^{l} \mu_i a_i \quad (2.7)$$

უტოლობის ორივე მხარე გავყოთ $\sum_{i=s+1}^{l} \mu_i a_i$ - ზე, გვექნება:

$$\frac{\sum_{j=1,k=1}^{s}\sum_{j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} \geq \frac{\sum_{i=s+1}^{l}\mu_i n_i}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{i=s+1}^{l}\mu_i a_i}{\sum_{i=s+1}^{l}\mu_i a_i} \tag{2.8}$$

ვინაიდან $[1:l]=[1:s]\cup[s+1:l]$ ამიტომ :

$$\frac{\sum_{j=1}^{s}\sum_{k=1,j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} = \frac{\sum_{j=1}^{s}\sum_{k=1,j\neq k}^{s}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{j=1}^{s}\mu_j \sum_{k=s+1}^{l} a_k}{\sum_{i=s+1}^{l}\mu_i \sum_{i=s+1}^{l} a_i}$$

ამგვარად, (8)-დან მივიღებთ ვედეკინდ-სურგულაძის მოდელს [52]:

$$\frac{\sum_{j=1}^{s}\sum_{k=1,j\neq k}^{l}\mu_j a_k}{\sum_{i=s+1}^{l}\mu_i a_i} + \frac{\sum_{j=1}^{s}\mu_j}{\sum_{i=s+1}^{l}\mu_i} \geq \sum_{i=s+1}^{l}\frac{n_i}{a_i} + 1 \tag{2.9}$$

სადაც $l \geq 2, s \geq 1$ და $l > s$.

პრაქტიკაში ხშირად გამოიყენება შემთხვევა, როცა *l*=2 და *s*=1, მაშინ (9) იღებს შემდეგ სახეს:

$$\frac{\mu_1}{\mu_2} \geq \frac{n_2}{a_2} + 1 \tag{2.10}$$

რაც, როგორც ცნობილია, არის ვონგ-ვედეკინდის მოდელი [56].

იგი არის (2.9) გამოსახულების კერძო შემთხვევა. (2.10)-ის გამოყენების დიაპაზონია მე-3ნფ-მდე, ხოლო (2.9)-ისა მთელი დიაპაზონი ნფ-ებისა, ამგვარად იგი უნივერსალურია.

ახლა გამოვიკვლიოთ შემთხვევა, როდესაც კორტეჟის არაგასაღებური ატრიბუტების მნიშვნელობათა ცვლილების სიხშირე მაღალია, ხოლო გასაღებურისა _ დაბალი. დავუშვათ, *l*=2, *s*=1 და მოცემულია რელაციათა სქემები:

$$\begin{cases} R_1(k_1,k_2,\ldots k_{n1},A_1,A_2,\ldots,A_{a1}) \\ R_2(k_1,k_2,\ldots k_{n2},B_1,B_2,\ldots,B_{a2}) \\ \\ R_{12}(k_1,k_2,\ldots k_{n1},A_1,A_2,\ldots,A_{a1},B_1,B_2,\ldots,B_{a2}) \end{cases}$$

რომლებშიც მართებულია შემდეგი პირობები:

$$k_1,\ldots k_n \supseteq k_1,\ldots k_{n2} \text{ da } \mu_1 > \mu_2 \qquad (2.11)$$

(2.9)-დან გამომდინარე, მოცემული R_1, R_2 და R_{12} სქემებისათვის, გასაღებურ ატრიბუტთა მნიშვნელობების ცვლილების მაღალი სიხშირის დროს მიზანშეწონილია R_1 da R_2 დამოკიდებულებათა კომპოზიცია R_{12}- ში, თუ სრულდება პირობა:

$$\mu_1(n_1+a_1)+\mu_2(n_2+a_2) > \mu_1(n_1+a_1+a_2).$$

აქედან გამომდინარეობს, რომ:

$$\frac{n_2}{a_2} > \frac{\mu_1}{\mu_2} - 1.$$

თუ განიხილება არაგასაღებური ატრიბუტების მნიშვნელობათა ცვლილება გასაღებური ატრიბუტების ცვლილების გარეშე, მაშინ მართებულია შემდეგი გამოსახულება:

$$\mu_1 a_1 + \mu_2 a_2 > \mu_1(a_1+a_2).$$

აქედან გამომდინარეობს, რომ:

$$\mu_2 > \mu_1.$$

რაც ეწინააღმდეგება (2.11)-ს.

ამგვარად, დამოკიდებულებათა სქემები, რომლებისთვისაც დომინირებადია არაგასაღებურ ატრიბუტთა ნაწილის ცვლილება, მიზანშეწონილია გამოისახოს მაღალი რიგის ნფ-ებით.

- სქემის გარდასახვა კონცეპტუალურ დონეზე შეიძლება გამოყენებულ იყოს იმისათვის, რომ უფრო ნათელი გახდეს კონცეპტუალური მოდელი ან გაუმჯობესდეს მონაცემთა ბაზის აპლიკაციის ხარისხი;

- მონაცემთა ბაზის დამოკიდებულებანი უნდა წარმოდგენილი იქნეს სხვადასხვა რიგის ნორმალური ფორმებით (3ნფ-:-ბნფ), განახლების სიხშირესა და კავშირების ტიპებზე დამოკიდებულებით მოცემულ კონტექსტში;

- მოცემული μ--თვის შეიძლება (3.9) გამოსახულებით დადგინდეს მოსახერხებელი (ოპტიმალური) ნფ-ები;

- თუ დამოკიდებულებათა გასაღებური ატრიბუტების ცვლილებების სიხშირე მაღალია, მაშინ მათთვის სასურველია დაბალი რიგის ნფ-ების გამოყენება, ხოლო თუ არაგასაღებურ ატრიბუტთა ცვლილების სიხშირე დომინირებადია, მაშინ - შედარებით მაღალი რიგის ნფ-ებისა.

2.6. გრაფულ-ორიენტირებული NoSQL მონაცემთა ბაზები /ლიტ-MongoDB-დან/

გრაფული მონაცემთა ბაზის მართვის სისტემებისთვის დამახასიათებელია მონაცემთა გრაფული მოდელი, ანუ ინფორმაციის შენახვა ხდება არა „ცხრილებით“ (tables) და ატრიბუტებით (როგორც რელაციური ბაზებში), არამედ გრაფული სტრუქტურებით, ანუ კვანძებით (nodes) და მათ შორის კავშირებით (გრაფის წიბოები - edges) [17]. ასეთი კავშირები შეიძლება რამდენიმე დონეს მოიცავდეს (ღრმა კავშირები) და ისინი ძალზე აქტუალურია დიდი სოციალური პროექტების (ქსელების), ბიოინფორმატიკის, რთული მარშრუტების, სემანტიკური ქსელის (Web, HTTP გვერდებით) და სხვა სფეროს ამოცანების გადასაწყვეტად.

გრაფული მონაცემთა ბაზა არის ქსელური მოდელის (ან RDF-მოდელის) რეალიზაციის ნაირსახეობა [57,58]. მისი კონცეფცია ჯერ კიდევ 80-იან წლებში გამოჩნდა, ხოლო პირველი გრაფული რეალიზაცია 2007 წელს, Neo4j სისტემის სახით. დღეისთვის უკვე არსებობს რამდენიმე ათეული ასეთი ბაზებისა, მაგალითად: ArangoDB, OrientDB, MarkLogic, Oracle Spatial and Graph და სხვ.

RDF (Resource Description Framework) - რესურსის აღწერის გარემო შეიქმნა WWW კონსორციუმის მიერ როგორც მონაცემთა აღწერის მოდელი - მეტამონაცემებით.

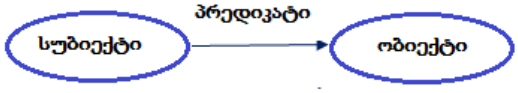
რესურსი RDF-ში შეიძლება იყოს ნებისმიერი არსი, როგორც ინფორმაციული (მაგალითად, ვებ-გვერდი, გამოსახულება), ასევე არაინფორმაციული (მაგალითად, ადამიანი, მანქანა ან აბსტრაქტული ცნება). რესურსის შესახებ გამონათქვამის მტკიცებას აქვს სამეულის (Triple Stores - სამადგილიანი შენახვა) სახე:

„სუბიექტი — პრედიკატი — ობიექტი“

მაგალითად, მტკიცება „დოლიძე არის პროფესორი“, RDF-ის ტერმინოლოგიით ჩაიწერება ასე:

სუბიექტი — „დოლიძე“, პრედიკატი — „აქვს თანამდებობა“, ობიექტი — „პროფესორი“.

გრაფიკულად იგი 2.10 ნახაზზეა მოცემული



ნახ.2.10. RDF-ის სამეული

ამგვარად, RDF-ის მტკიცებულებები (ფაქტები) ქმნის ორიენტირებულ გრაფს, რომელშიც კვანძებია სუბიექტები და ობიექტები, ხოლო წიბოები - მათ შორის მიმართებები. RDF არის მონაცემთა აბსტრაქტული მოდელი, ანუ იგი აღწერს მოცემულ სტრუქტურას, დამუშავების ხერხებს და მონაცემთა ინტერპრეტაციებს.

ქვემოთ მოცემულია NoSQL ტიპის ოჯახის ზოგიერთი პოპულარული მბმს, რომლებიც გრაფულ ან ჰიბრიდულ ბაზებს მიეკუთვნება [17].

AllegroGraph – დამუშავებულია W3C სტანდარტით Common Lisp ენაზე Triple Store (პრედიკატული სამეული) სახით მონაცემთა RDF მოდელისთვის, Windows, Linux და Mac OSX -თვის, კლიენტის ინტერფეისებით: Java, Python, Ruby, Perl, C#, Clojure და Common Lisp.

ArangoDB – დაწერილია C++ და JavaScript ენებზე მულტიმოდელური ბაზის სახით. გამოიყენება როგორც key/value, document, და graph data ბაზები და აქვთ ერთი საერთო მოთხოვნების ენა [59]. 2011 წლამდე გამოდიოდა AvocadoDB სახელით.

DataStax Enterprise Graph (DSE) – აგებულია Java ენაზე Web-საიტებისა და მობილური ტექნიკისთვის. სერვერის მხარეს Backend-ის სახით იყენებს Apache Cassandra-ს. შეუძლია დაამუშაოს წამში პეტაბაიტი ინფორმაცია და ერთდროულად მოემსახუროს ათას მომხმარებელს. ბაზა განაწილებულია კვანძების კლასტერებში და აქვს მასშტაბირებადი არქიტექტურა [58,60]. მასში ჩადგმულია OLAP ანალიზის და გრაფში ძებნის მხარდაჭერა. აქვს უსაფრთხოების დამატებითი პარამეტრები კონფიდენციალური მონაცემებისთვის.

MarkLogic – მულტიმოდელური ბაზაა სემანტიკური გრაფით და RDF სამეულით. ინახავს დოკუმენტებს JSON (JavaScript Object Notation) და XML ფორმატებში [24,29]. აქვს ჩადგმული საძიებო სისტემა, ACID მახასიათებლების მქონე ტრანზაქციები – მაღალი წვდომადობა და ავარიული აღდგენის უნარი, გარანტირებული უსაფრთხოება, მოქნილობა და მასშტაბურობა.

Neo4j – კომპანია Neo Technology-ის პროდუქტი, დამუშავებულია java ენაზე და ერთ-ერთი ყველაზე პოპულარული გრაფული ბაზაა [58,61]. აპლიკაციების დაპროგრამების ინტერფეისი მონაცემთა ბაზისთვის რეალიზებულია მრავალი ენისთვის, მათ შორის: Java, Python, Ruby, PHP და სხვ.

OrientDB – გრაფული- და დოკუმენტ-ორიენტირებული ბაზის სისტემაა, დამუშავებულია Java ენაზე Orient Technologies LTD ფირმის მიერ Windows, Linux, Mac და სხვა ოპერაციული სისტემებისთვის [58,62]. მოთხოვნების ენისათვის აქვს SQL-ის მხარდაჭერა (ამიტომაც მას NewSQL ბაზასაც მიაკუთვნებენ). იგი არ იყენებს JOIN ოპერაციას. მის მაგივრად აქვს სუპერ-სწრაფი მუდმივი მიმთითებლები ჩანაწერებს შორის, რომლებიც გრაფული ბაზებისთვისაა დამახასიათებელი. ეს უზრუნველყოფს ჩანაწერების ცალკეული ან მთლიანი ხეების და გრაფების გადასინჯვას რამდენიმე მილიწამის ფარგლებში.

Stardog – არის კროსპლატფორმული, სემანტიკური გრაფული ბაზის სიტემა, რეალიზებული Java ენაზე, RDF-ის და OWL (Web Ontology Language)-ის მხარდაჭერით [63]. OWL ენით აღიწერება კლასები და მიმართებები მათ შორის, რომლებიც დამახასიათებელია ვებ-დოკუმენტებისა და აპლიკაციებისთვის [64].

III თავი
მონაცემთა ტიპები და ბაზების ობიექტები

3.1. მონაცემთა ბაზების მართვის სისტემა Ms SQL Serever

გამოყენებითი სფეროს აპლიკაციის მონაცემთა ბაზა (Database) შედგება ურთიერთდაკავშირებული ცხრილებისაგან (Tables). ეს კავშირები ძირითადად რეალიზებულია პირველადი (Primary key - PK) და მეორეული (Foreign key - FK) გასაღებებით. შესაძლებელია ინდექსების გამოყენებაც (ინდექსური ფაილების შესაქმნელად) [10,39,65].

ჩვენ ვგულისხმობთ, რომ მკითხველს აქვს საწყისი წარმოდგენა რელაციურ ბაზებთან სამუშაოდ და, კერძოდ, MsSQLServer-თან. ამიტომ აქ ჩვენ მოკლედ განვიხილავთ ზოგიერთ საკითხს (გამეორების თვალსაზრისით), რაც დაგვჭირდება პროგრამული აპლიკაციების გასამართად, მომხმარებელთა ინტერფეისების სამუშაოდ მონაცემთა ბაზებთან.

3.1.1. მონაცემთა ძირითადი ტიპები

მონაცემთა ბაზის ცხრილი (Table), როგორც ვიცით სვეტების (ატრიბუტების) და სტრიქონებისგან (კორტეჟებისგან) შედგება. იგი სტრუქტურაა, რომლის ელემენტები შეიძლება მონაცემთა სხვადასხვა ტიპით განისაზღვროს (ცხრ.3.1).

R1=„Student” **ცხრ.3.1**

ID მთელრიცხვა	A1 სტრიქონული	A2 ნამდვილ- რიცხვა	…	An თარიღი
101	აბესაძე ავთანდილ	19		21/05/76
102	ბურძგლა დიმიტრი	27		03/01/59
115	. . .	. . .	. . .	. . .

MsSQL Server-ში მონაცემთა ტიპები მრავალფეროვანია [10]. ერთი სვეტის (ატრიბუტის) ყველა მნიშვნელობა ერთი ტიპისაა. გამონაკლისია მხოლოდ SQL_VARIANT ტიპი, რომელშიც შესაძლებელია ერთდროულად რამდენიმე ტიპის მონაცემის შენახვა. მაგალითად, რიცხვითი, სტრიქონული ან თარიღის ტიპის მონაცემები. მონაცემთა ტიპები იყოფა შემდეგ კატეგორიებად (ცხრ.3.2-:-3.6):

- რიცხვითი ტიპები;
- სიმბოლური ტიპები;
- დროითი ტიპები (თარიღი და დრო);
- სხვა ტიპები.

რიცხვითი ტიპები ცხრ.3.2

მონაცემთა ტიპი	ბაიტი	აღწერა
INT	4	მთელრიცხვა: -2^{31}-:-$2^{31}-1$
SMALLINT	2	მთელრიცხვა: -2^{15}-:-$2^{15}-1$
TINYINT	1	მთელრიცხვა: 0 - 255
BIGINT	8	მთელრიცხვა: -2^{63}-:-$2^{63}-1$
DEC(p,[s]) ან NUMERIC	5-:-17	p -სიზუსტის წილადი -2^{38} +1 238 -1 s –თანრიგები წერტილის მარჯვნივ
REAL		მცოცავწერტილიანი დადებითი 2,23E –308 -:- 1,79E +308, უარყოფ. –1,18E –38 -:- –1,18E +38
FLOAT[(p)]	4	მცოცავწერტილიანი if p < 25
	8	if p >= 25
MONEY	8	ფულის ტიპი: -2^{63} -:- 2^{63} -1
SMALLMONEY	4	ფულის ტიპი: -2^{31} -:- 2^{31} -1

სიმბოლური ტიპები ცხრ. 3.3

მონაცემთა ტიპი	ბაიტი	აღწერა
CHAR[(n)]	min 1	ფიქსირებული სიგრძის სტრიქონი n = 1-:- 8000 (სიმბოლო)
VARCHAR[(n)]	min 1	ცვლადი სიგრძის სტრიქონი 0 < n < 8000 (სიმბოლო)
NCHAR[(n)]	min 2	ფიქს.სიგრძის Unicode სტრიქონი n = 1-:- 4000 (სიმბოლო)
NVARCHAR[(n)]	min 2	ცვლადი სიგრძის Unicode სტრიქონი 0 < n < 4000 (სიმბოლო)

დროითი ტიპები (თარიღი და დრო) ცხრ.3.4

მონაცემთა ტიპი	ბაიტი	აღწერა
DATETIME	4	თარიღი და დრო დიაპაზონით: 01/01/1753 -:- 31/12/9999
SMALLDATETIME	2	თარიღი და დრო დიაპაზონით: 01/01/1900 -:- 06/06/2079

DATE	3	თარიღი დიაპაზონით: 01/01/0001-:-31/12/9999 მაგ.: 'mmm dd yyyy' ('Jan 07 2016'). SET DATEFORMAT-ით იცვლება
TIME	3-:-5	დრო 100 ნანოსეკ. სიზუსტით. მაგ.: 'hh:mm' ('21:45')
DATETIME2	6-:-8	თარიღი და დრო დიდი სიზუსტით
DATETIMEOFFSET	6-:-8	თარიღი და დრო დროის სარტყელით

ორობითი და ბიტური ტიპები **ცხრ.3.5**

მონაცემთა ტიპი	ბაიტი	აღწერა
BINARY[(n)]	=n	ფიქსირ. სიგრძის ბიტების სტრიქონი 0 < n < 8000
VARBINARY[(n)]	n-მდე	ცვლადი სიგრძის ბიტების სტრიქო- ნი 0 < n < 8000
BIT	1 ბიტი	ლოგიკური მნიშვნელობა: FALSE, TRUE და NULL

დიდი ობიექტების ტიპები **ცხრ.3.6**

მონაცემთა ტიპი	აღწერა
VARCHAR(max)	LOB : ობიექტი 2GB - მდე
NVARCHAR(max)	LOB : ობიექტი 2GB - მდე
VARBINARY(max)	BLOB: ატრიბუტით FILESTREAM შეინახება მონაცემები NTFS ფაილურ სისტემაში

UNIQUEIDENTIFIER – გლობალური უნიკალური იდენტიფიკატორების (Global Unique Identifier, GUID) შენახვის ტიპი;

TIMESTAMP – დროითი შტამპი, რომელიც აფიქსირებს დროს სტრიქონის ყოველი ცვლილებისას;

HIERARCHYID – ინახავს სრულ იერარქიას (მაგ.: თანამშრომელთა იერარქია, კატალოგში ფოლდერების იერარქია და ა.შ.);

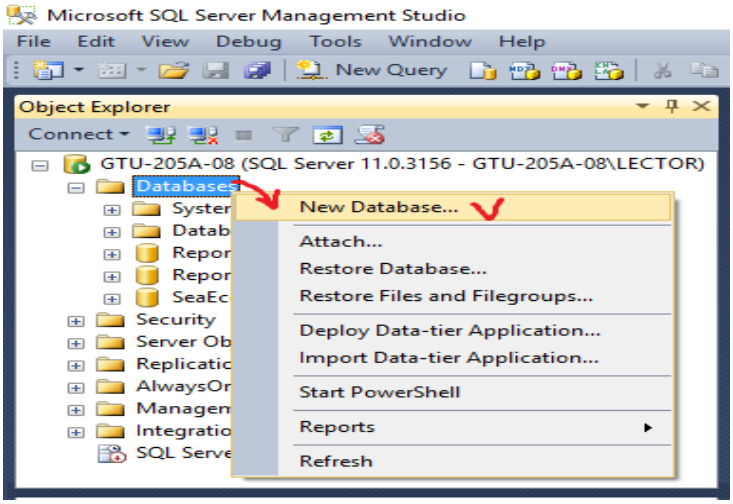
SPARSE – „მეჩხერი" სვეტების (შეიცავს ბევრ NUL მნიშვნელობას) შენახვის მოცულობის ოპტიმიზაცია.

3.1.2. მონაცემთა ბაზის ობიექტების შექმნა

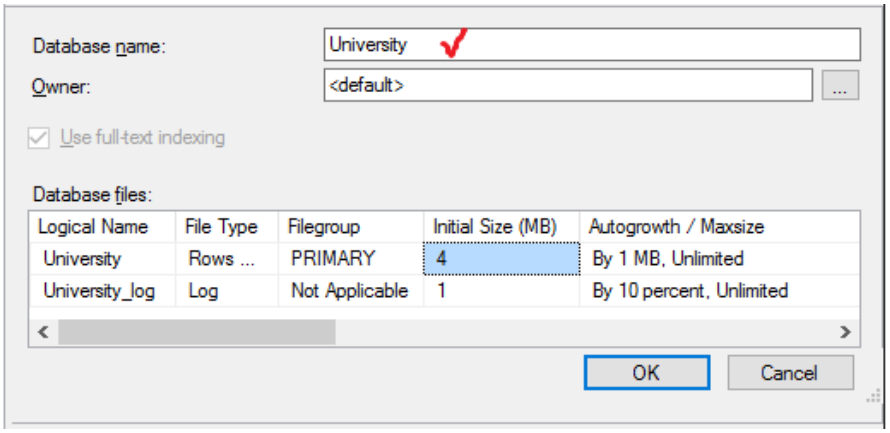
მონაცემთა ბაზის ობიექტები არის ფიზიკური (დისკებზე), როგორიცაა ფაილები და ფაილთა ჯგუფები, ან ლოგიკური - მომხმარებელთა წარმოდგენები მონაცემთა ბაზის შესახებ. ლოგიკური ობიექტების მაგალითებია ცხრილები (Tables), სვეტები (Columns) და ვირტუალური ცხრილები (Views - წარმოდგენები).

მონაცემთა ბაზის ობიექტი, რომელიც პირველ რიგში უნდა შეიქმნას, არის თვით მონაცემთა ბაზა. Database Engine კომპონენტი მართავს სისტემურ და მომხმარებელთა მონაცემთა ბაზებს. სისტემური ბაზები იქმნება მონაცემთა ბაზის ინსტალირების დროს, ხოლო მომხმარებელთა ბაზები კი - თვით ავტორიზებული მომხმარებლის მიერ.

მონაცემთა ბაზის შექმნა ორი მეთოდითაა შესაძლებელი. პირველი, SQL Server Management Studio-ს დახმარებით, რომლითაც ხდება დიალოგურ რეჟიმში პროცესების წარმართვა (ნახ.3.1, 3.2.).



ნახ.3.1. ახალი ბაზის შექმნის დაწყება



ნახ.3.2. იქმნება University ბაზა

მეორე, Transact-SQL ენის CREATE DATABASE ინსტრუქციის დახმარებით:

```
CREATE DATABASE db_name
    [ON [PRIMARY] {file_spec1},.]
    [LOG ON {file_spec2},.]
        [COLLATE collation_name]
  [FOR {ATTACH ATTACH_REBUILD_LOG}]
```

ჩვენ აქ Transact-SQL ენას დეტალურად არ განვიხილავთ.

საჭიროა აღვნიშნოთ, რომ ერთ სისტემას შეუძლია 32 767 მონაცემთა ბაზის მართვა. ყველა ბაზა ინახება ფაილებში, რომლებიც შეიძლება ცხადად იყოს მითითებული ადმინისტრატორის მიერ ან არაცხადად იყოს წარმოდგენილი სისტემის მიერ. თუ CREATE DATABASE ინსტრუქცია შეიცავს ON პარამეტრს, მაშინ ბაზის ყველა ფაილი ცხადად არის გამოცხადებული.

დიდი ბაზებისთვის სასურველია ფაილთა ჯგუფების გამოყენება. ფაილი ინახავს ერთი ბაზის მონაცემებს. ფაილთა ჯგუფები საშუალებას იძლევა გადანაწილდეს მონაცემები სხადასხვა დისკებზე, შესრულდეს სარეზერვო დუბლირება და მონაცემთა ბაზის ნაწილის აღდგენის პროცედურა.

PRIMARY პარამეტრი მიუთითებს პირველ (მნიშვნელოვან) ფაილზე, რომელიც შეიცავს სისტემურ ცხრილებს და სხვა საჭირო შიგა ინფორმაციას ბაზის შესახებ.

COLLATE ოფციაში მიეთითება მოწესრიგების მიმდევრობა.

FOR ATTACH ოფცია მიუთითებს, რომ მონაცემთა ბაზა იქმნება უკვე არსებული ფაილების მიერთებით.

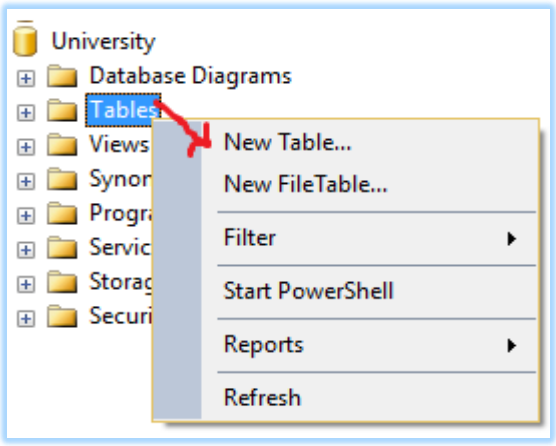
არსებული მონაცემთა ბაზის **მომენტალური სურათის** შექმნა CREATE DATABASE ინსტრუქციით (როცა ბაზაში დასრულებულია გარკვეული ტრანზაქციები და საჭიროა დუბლის შექმნა):

```
CREATE DATABASE database_snapshot_name
    ON (NAME = logical_file_name,
        FILENAME = 'C:\temp\file_name') [,...n ]
       AS SNAPSHOT OF source_database_name
```

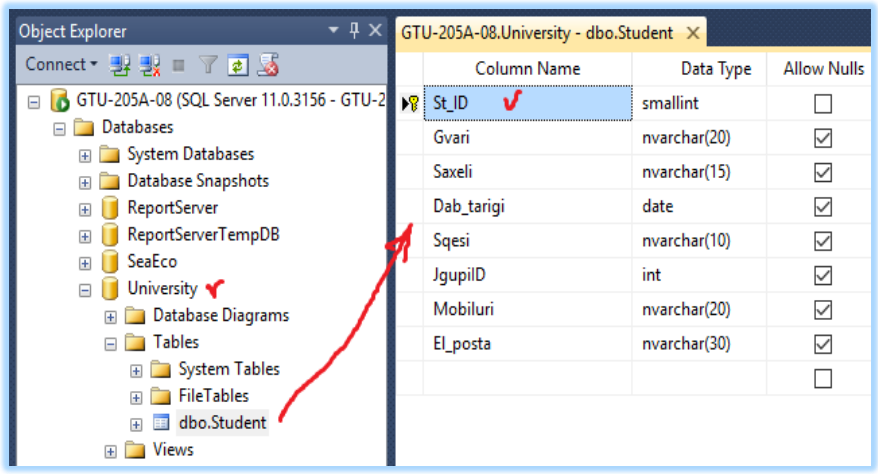
ამისათვის აქ გამოიყენება სტრიქონი AS SNAPSHOT OF. თავიდან უნდა შეიქმნას ბაზის მომენტალური სურათის შესანახი კატალოგი, მაგალითად, C:\temp\... . (NTFS - ფაილურ სისტემაში).

უნდა აღინიშნოს, რომ მომენტალური სურათი შეიცავს მონაცემთა ბაზის მხოლოდ შეცვლილ ნაწილს და ყოველი გადაღებული კადრი არ მოითხოვს დიდ დისკურ მეხსიერებას.

მონაცემთა ბაზის ცხრილის (Table) შექმნა ხორციელდება ინტერაქტიულ რეჟიმში SQL Server Management Studio-ს დახმარებით (ნახ.3.3). მომხმარებელს შეაქვს ცხრილის ატრიბუტები მათი ტიპების და სხვა მახასიათებლების მითითებით (ნახ.3.4).



ნახ.3.3. ცხრილის შექმნის დაწყება



ნახ.3.4. Student ცხრილის შექმნა

3.5 ნახაზზე მოცემულია Student ცხრილის შევსების მაგალითი რამდენიმე სტრიქონით.

ცხრილების შექმნის მეორე გზა Transact-SQL ენის CREATE TABLE ინსტრუქციაა, რომლ;ის საბაზო ფორმა ასეთია:

CREATE TABLE table_name

(col_name1 type1 [NOT NULL | NULL)

[{, col_name2 type2 [NOT NULL | NULL]} ...])

აქ col_name ველის სახელია, ხოლო type - შესაბამისი ველის ტიპი.

GTU-205A-08.University - dbo.Student X | GTU-205A-08.University - dbo.Student

St_ID	Gvari	Saxeli	Dab_tariqi	Sqesi	JgupilD	Mobiluri	El_posta
1	აბაშიძე	აკაკი	1995-05-17	მამრობითი	108550	577102030	abashidze.ak@gmail.com
2	ბურდული	ბუდუ	1997-01-31	მამრობითი	108550	599707070	burdubu@yahoo.com
3	ბახტაძე	მერაბ	1995-01-01	მამრობითი	108551	591111222	bakhtadze.m@gmail.com
4	გაგუა	ნინო	1998-08-20	მდედრობითი	108551	577223355	gaguani@yahoo.com
5	კაკუბავა	ნინო	1998-05-09	მდედრობითი	108555	595777555	kakunino@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ნახ.3.5. Student ცხრილის ფრაგმენტი

ცხრილების მაქსიმალური რაოდენობა ერთ ბაზაში შეზღუდულია მონაცემთა ბაზის ობიექტების რაოდენობით, რომელთა რაოდენობა არ უნდა აღემატებოდეს 2 მილიარდს. აქ შედის ცხრილები (Tables), წარმოდგენები (Views), შენახვადი პროცედურები (Store procedures), ტრიგერები (Triggers) და შეზღუდვები (Constraints).

3.1.3. დეკლარაციული მთლიანობის შეზღუდვები და დომენები

მონაცემთა ბაზების მართვის სისტემის ერთ-ერთი ყველაზე მნიშვნელოვანი მოთხოვნაა მონაცემთა მთლიანობის უზრუნველყოფა. შეზღუდვებს, რომლებიც გამოიყენება მონაცემთა შესამოწმებლად მათი ცვლილების ან დამატებისას, უწოდებენ მთლიანობის უზრუნველყოფის შეზღუდვებს (Integrity constraints) [39].

მონაცემთა მთლიანობის უზრუნველყოფას გამოყენებით პროგრამაში ახორციელებს მომხმარებელი ან თვით მონაცემთა ბაზების მართვის სისტემა. ამ უკანასკნელ შემთხვევაში გვაქვს შემდეგი უპირატესობები:

- მაღლდება მონაცემთა საიმედოობა;
- მცირდება დაპროგრამების დრო;
- მარტივდება ტექნიკური მომსახურება.

მთლიანობის უზრუნველსაყოფად მონაცემთა ბაზების მართვის სისტემაას აქვს ორი ტიპის შეზღუდვა: დეკლარაციული შეზღუდვები და პროცედურული შეზღუდვები (ტრიგერებით რეალიზებადი).

დეკლარაციული შეზღუდვები განისაზღვრება DDL ენის CREATE TABLE და ALTER TABLE ინსტრუქციებით, სვეტების ან ცხრილების დონეზე [72]. ყოველ დეკლარაციულ შეზღუდვას ენიჭება სახელი. იგი ენიჭება ცხადად CONSTRAINT ოფციის გამოყენებით CREATE TABLE ან ALTER TABLE ინსტრუქციაში.

დეკლარაციული შეზღუდვები შეიძლება დაჯგუფდეს შემდეგ კატეგორიებში [62]:

- DEFAULT ;
- UNIQUE;

- PRIMARY KEY;
- CHECK;
- FOREIGN KEY.

დომენი (domain) მონაცემთა ბაზის ცხრილში სვეტის (ველის) დასაშვებ მნიშვნელობათა ერთობლიობაა. მათთვის, ჩვეულებისამებრ, გამოიყენება მონაცემთა ტიპები: INT, CHAR, DATE და სხვ. დომენის მთლიანობის უზრუნველყოფის მიზნით ასეთი მეთოდი არაა საკმარისი [10,39]. მაგალითად, უნივერსიტეტის (University) მონაცემთა ბაზის ლექტორთა (Lector) ცხრილში ველისათვის ქალაქის კოდი (zip) შეიძლება როგორც SMALLINT, ისე CHAR(5) ტიპების გამოყენება, მაგრამ არც ერთი არ იქნება ზუსტი. პირველში რიცხვთა დიაპაზონი მოიცავს დადებითთან ერთად უარყოფით მნიშვნელობებსაც. მეორეში კი 5-სიმბოლოიანი მნიშვნელობა შეიძლება შედგებოდეა ალფაბეტის სიმბოლური, რიცხვითი და სხვა მნიშვნელობისგან. ჩვენ კი გვჭირდება მხოლოდ დადებით რიცხვთა დიაპაზონი ინტერვალში 00001 -:- 00099.

მეტნაკლები სიზუსტით დომენების მთლიანობის უზრუნვეყოფა შესაძლებელია CHECK შეზღუდვით ინსტრუქციაში CREATE TABLE ან ALTER TABLE.

Transact-SQL ენას ამ მიზნით დომენებისათვის აქვს მონაცემთა ტიპების ფსევდონიმების შექმნის შესაძლებლობა CREATE TYPE ინსტრუქციით.

მონაცემთა ტიპების ფსევდონიმი (alias data type) - მომხმარებლის მიერ განსაზღვრული სპეციალური ტიპია, რომელიც გამოიყენება არსებული საბაზო ტიპების საფუძველზე. მისი შექმნის ზოგადი სინტაქსი ასეთია:

```
CREATE TYPE [type_schema_name.] type_name
{[FROM base_type[(precision[ , scale ])] [NULL | NOT NULL]]
   | [EXTERNAL NAME assembly_name [.class_name]]}
```

ჩვენ მიერ ზემოთ განხილული კონკრეტული მაგალითისათვის გვექნება შემდეგი ტექსტი:

```
USE University;
CREATE TYPE zip
FROM SMALLINT NOT NULL;
```

ახლადშექმნილი ფსევდონიმური ტიპის გამოყენაბა შეგვიძლია ასე:

```
USE University;
CREATE TABLE Lector
    (LectorID INT NOT NULL,
     Lector_name CHAR(20) NOT NULL,
     city CHAR(20),
     zip_code ZIP,
     CHECK (zip_code BETWEEN 101 AND 199));
```

ამ მაგალითში Lector ცხრილის zip_code ველისათვის ტიპი განისაზღვრება zip ფსევდონიმური ტიპით. ამ ველის დასაშვებ მნიშვნელობათა შეზღუდული დიაპაზონი იქნება მთელი რიცხვები 101-199, რაც CHECK -ითაა მოცემული.

CREATE TYPE ინსტრუქციით კი ასეთი ტიპის შექმნა შემდეგნაირად ხორციელდება:

```
USE University;
CREATE TYPE Lector_table AS TABLE
    (gvari VARCHAR(30), xelpasi DECIMAL(8,2));
```

მომხმარებლის მიერ შექმნილი მონაცემთა ცხრილურ ტიპს Lector_table აქვს ორი ველი gvari და xelpasi.

ძირითადი სინტაქსური განსხვავება ცხრილურ და ფსევდონიმურ მონაცემთა ტიპებს შორის არის AS TABLE წინადადების არსებობა. მომხმარებლის მიერ განსაზღვრული ცხრილური ტიპები გამოიყენება მაშინ, როდესაც საჭიროა ცხრილურ მნიშვნელობათა პარამეტრების უკან დაბრუნება.

3.2. მონაცემთა ბაზების უსაფრთხოების სისტემა

მონაცემთა რელაციური ბაზების მართვის სისტემის ერთ-ერთი მნიშვნელოვანი კომპონენტია Database Engine, რომლის მომხმარებელთა ინტერფეისის დახმარებითაც საგრძნობლად მარტივდება სისტემასთან მუშაობა [66,67]. აგრეთვე აქვს სხვადასხვა ინსტრუმენტი მონაცემთა ბაზის ობიექტების შესაქმნელად, დანართების ასაწყობად და სისტემური ადმინისტრირების ამოცანების სამართავად.

ჩვენ განვიხილავთ Database Engine კომპონენტის სისტემას მონაცემთა უსაფრთხოების უზრუნველყოფის მიზნით. ასეთია ავტორიზაციის (მონაცემთა ბაზასთან სანქცირებული მიმართვის) საკითხი, აუტენთიფიკაცის (მონაცემებთა მომხმარებელთა წვდომის პრივილეგიების) საკითხი, აგრეთვე მონაცემთა მოდიფიკაციის თანხლების შესაძლებლობები Database Engine კომპონენტით.

ამგვარად, მონაცემთა ბაზების უსაფრთხოების დაცვის ძირითადი კონცეფციებია:

- აუტენთიფიკაცია (მოხმარებლის სახელი, პაროლი);
- შიფრაცია (ინფორმაციის კოდირება, კრიპტოგრაფია);
- ავტორიზაცია (ბაზის რესურსების ფლობის ნებართვა);
- ცვლილებების მონიტორინგი (ბაზაში ყველა მიმართვის და ცვლილების ფიქსირება და დოკუმენტირება).

SQL Server სისტემაში მონაცემთა უსაფრთხოების მოდელი შედგება სამი განსხვავებული კატეგორიისაგან:

❖ პრინციპალები (principals) - სუბიექტებია, რომელთაც აქვთ წვდომის ნებართვა ბაზის განსაზღვრულ არსებთან (entities). არსებობს ასევე Windows-ჯგუფები და SQL Server-

როლები. მომხმარებელს მიენიჭება შესაბამისი ჯგუფის ან როლის სააღრიცხვო ჩანაწერი, რაც მის უფლებებს განსაზღვრავს;

- დაცული ობიექტები (securables) - რესურსებია, რომლებზეც წვდომა რეგულირდება ბაზის ავტორიზაციის სისტემით;
- ნებართვები (permissions) - ყოველ დაცულ ობიექტს აქვს მასთან დაკავშირებული ნებართვა, რომელიც წარედგინება პრინციპალს.

3.2.1. აუტენთიფიკაცია

Database Engine კომპონენტის უსაფრთხოების სისტემა შედგება ორი განსხვავებული ქვესისტემისგან:

- Windows უსაფრთხოების სისტემები;
- SQL Server უსაფრთხოების სისტემები.

Windows უსაფრთხოების სისტემა განსაზღვრავს უსაფრთხოებას ოპერაციული სისტემის დონეზე. ესაა მეთოდი, რომლის დახმარებითაც მომხმარებელი შედის Windows სისტემაში.

SQL Server უსაფრთხოების სისტემა განსაზღვრავს დამატებით უსაფრთხოებას, რომელიც საჭიროა მონაცემთა ბაზის დონეზე. ესაა ხერხი, რომლითაც მომხმარებელი უკვე Windows სისტემაშია და ცდილობს მონაცემთა ბაზის სერვერთან დაკავშირებას.

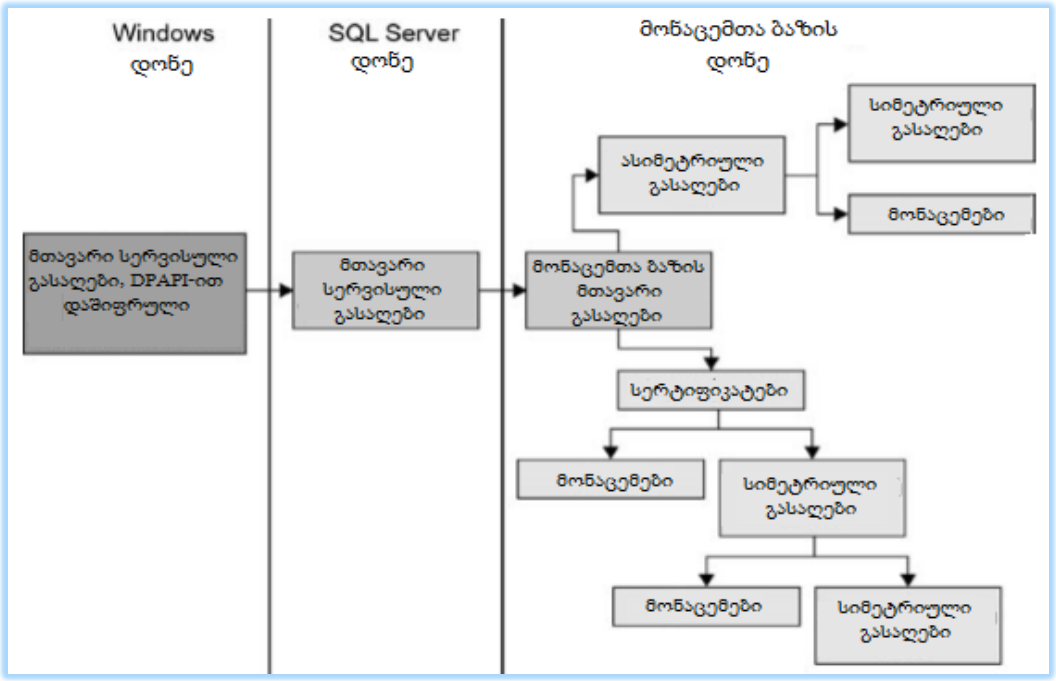
3.2.2. მონაცემთა შიფრაცია

შიფრაცია არის მონაცემების კოდირება (მაგალითად, კრიპტოგრაფიის მეთოდებით), რომლის დროსაც გაუგებარი ხდება მათი შინაარსი. Database Engine კომპონენტი უზრუნველყოფს მონაცემთა დაცვას შიფრაციის იერარქიული დონეების მიხედვით და გასაღებების მართვის ინფრასტრუქტურით. შიფრაციის ყოველი დონე იცავს მის მომდევნო დონეს, იყენებს სერტიფიკატების, სიმეტრიული და ასიმეტრიული გასაღებების კომბინაციას (ნახ.3.6).

მთავარი სერვისული გასაღები მართავს ყველა დანარჩენ გასაღებს და სერტიფიკატებს. იგი იქმნება ავტომატურად Database Engine კომპონენტის დაყენებისას. ეს გასაღები დაშიფრულია API- Windows მონაცემთა დაცვის ინტერფეისის დახმარებით (DPAPI – Data Protection API) [67].

მონაცემთა ყოველ ბაზას აქვს თავისი ერთი მთავარი გასაღები, რომელიც იქმნება CREATE MASTER KEY ინსტრუქციით. ეს გასაღები დაცულია სისტემის მთავარი სერვისული გასაღებით, რომელსაც შეუძლია ავტომატურად მისი გაშიფრვა.

მონაცემთა ბაზის მთავარი გასაღებით შესაძლებელია მომხმარებელთა გასაღებების შექმნა: სიმეტრიული, ასიმეტრიული და სერტიფიკატი. სიმეტრიულის დროს ორივე მხარეს (გადამცემი, მიმღები) აქვს ერთი გასაღები, ხოლო ასიმეტრიულის დროს - სხვადასხვა. პირველი უფრო მარტივია, მეორე კი - საიმედო.



ნახ.3.6. შიფრაციის იერარქიული კომპონენტი [11]

SQL Server-ში გამოიყენება მონაცემთა შიფრაციის ორი ხერხი: შიფრაცია სვეტების დონეზე და გამჭვირვალე შიფრაცია (სიმეტრიული გასაღების საფუძველზე). უფრო დეტალურად შეიძლება ამ საკითხების გაცნობა [39,66].

3.3. სივრცითი მონაცემთა ტიპები SQL Server მონაცემთა ბაზაში

21-ე საუკუნის დასაწყისიდან ბიზნესის, გეოსისტემებისა და სხვა სფეროებში მზარდი მოთხოვნილება გაჩნდა და მნიშვნელოვანი განვითარება დაიწყო სივრცითი მონაცემების მოდელირების, მონაცემთა ბაზაში შენახვისა და მათი ასხვის ტექნოლოგიების ტექნოლოგიებმა. ამ პროცესებზე განსაკუთრებით დიდი გავლენა იქონია „მაიკროსოფთის“ კორპორაციის Virtual Earth სისტემის და GPS ინსტრუმენტების (Global Positioning Systems – ნავიგაციის გლობალური სისტემა და ადგილმდებარეობის განსაზღვრა) გამოჩენამ [1,68].

მოდელები. სივრცითი მონაცემების ასახვის მიზნით დღეისათვის გამოიყენება ორი სახის მოდელი:

- გეოდეზიური სივრცითი მოდელები და
- ბრტყელი სივრცითი მოდელები.

პლანეტები, როგორც რთული ობიექტები შეიძლება წარმოდგენილ იყოს სფეროიდების სახით. ამის ერთ-ერთი კარგი მაგალითია გლობუსი – დედამიწის მოდელი. მისი ზედაპირის კონკრეტული წერტილი აღიწერება განედით (პარალელები, ეკვატორის ჩრდილოეთით და სამხრეთით) და გრძედით (მერიდიანები, მაგალითად, 0-ოვანი მერიდიანის დასავლეთით ან აღმოსავლეთით). ასეთ მოდელებს გეოდეზიურს უწოდებენ.

ბრტყელ სივრცით მოდელებში, მაგალითად დედამიწის ასახვის მიზნით, გამოიყენება ორგანზომილებიანი რუკები.

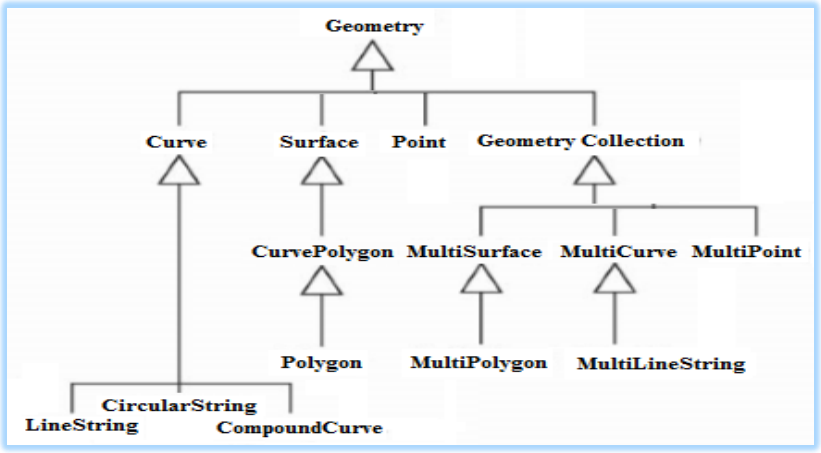
ამგვარად, ჩვენ განვიხილავთ სივრცითი მონაცემების ორ ტიპს:

- GEOMETRY – ბრტყელი სივრცითი მოდელი;
- GEOGRAPHY – გეოდეზიური მოდელი.

მონაცემთა ასეთი ტიპები რეალიზებულია და გამოიყენება მონაცემთა ბაზების მართვის სისტემის, SQL Server 14 და უდრო ახალ ვერსიებში [39,69].

3.3.1. მონაცემთა ტიპი GEOMETRY

ტერმინი - გეომეტრიული ობიექტი შემოიტანა OGC (Open Geospatial Consortium) კონსორციუმმა, რათა შესაძლებელი ყოფილიყო ისეთი სივრცითი თვისებების ასახვა, როგორიცაა წერტილები და წრფეები. ამიტომაც გეომეტრიული ობიექტი მონაცემებს წარმოადგენს ორგანზომილებიან სივრცეში წერტილების, წრფეებისა და მრავალ-კუთხედების სახით. გეომეტრიული ობიექტის მონაცემთა ტიპს აქვს ქვეტიპები, როგორც ეს 3.7 ნახაზზეა ნაჩვენები [10].



ნახ.3.7. ტიპების იერარქია GEOMETRY ფესვური ტიპით

როგორც ნახაზიდან ჩანს, ქვეკლასები დაყოფილია ორ კატეგორიად: საბაზო გეომეტრიულ ქვეკლასებად და ერთგვაროვან კოლექციათა ქვეკლასებად.

საბაზო გეომეტრიული ქვეკლასები შედგება Point, LineString და Polygon ქვეკლასებისაგან, ხოლო ერთგვაროვანი კოლექციები - MultiPoint, MultiLineString და MultiPolygon ქვეკლასებისგან, მათ საკუთარი თვისებებიც აქვს.

ახლა განვიხილოთ სივრცითი მონაცემების აღნიშნული ტიპების მოკლე დახასიათება:

- **Point (წერტილი)** – არის ორგანზომილებიანი გეომეტრიული ობიექტი კოორდინატთა X და Y მნიშვნელობებით, ამ ტიპის ეგზემპლარს შეიძლება ჰქონდეს ორი დამატებითი კოორდინატა: დონე (Z) და ზომა (M). Point ტიპის ეგზემპლარები გამოიყენება რთული სივრცითი ტიპების ასაგებად;
- **MultiPoint (მრავალწერტილოვანი)** – არის კოლექცია ნულოვანი ან მეტი რაოდენობის წერტილებისა. არაა აუცილებელი მრავალწერტილოვან ტიპში წერტილები იყოს განსხვავებული;
- **LineString (ტეხილი ხაზი)** – არის გეომეტრიული ობიექტი, რომელსაც აქვს სიგრძე და შეინახება წერტილების მიმდევრობით, რომელიც განსაზღვრავს ხაზის გზას. ტეხილ ხაზს აქვს საკონტროლო წერტილები (გადახრის ადგილი). ტეხილი ხაზი მარტივია (simple), თუ იგი არ კვეთს თავის თავს. თუ ტეხილი ხაზის საწყისი და საბოლოო წერტილები ემთხვევა, იგი ჩაკეტილია (closed). ჩაკეტილ, მარტივ ტეხილს უწოდებენ რგოლს (ring).
- **MultiLineString (მრავალტეხილი ხაზები)** – არის კოლექცია ნულოვანი ან მეტი რაოდენობის ტეხილი ხაზის.
- **Polygon (მრავალკუთხედი)** – არის ორგანზომილებიანი გეომეტრიული ფიგურა ზედაპირით. მრავალკუთხედი ინახება მიმდევრობითი წერტილების სახით, რომლებითაც განისაზღვრება მისი გარეგანი შემომსაზღვრელი პერიმეტრი (ring) და შიგა მრავალკუთხედები (ნული ან რამდენიმე). გარე და შიგა ფიგურები იძლევა მრავალკუთხედის საზღვრებს, ხოლო სივრცე მათ შორის კი - განსაზღვრავს შიგა ნაწილს.
- **MultiPolygon ()** – მრავალკუთხედების კოლექციაა (0 ან მეტი).
- **GeometryCollection** – გეომეტრიული ეგზემპლარების კოლექციაა 0 ან მეტი გეომეტრიული ობიექტით. ამ ტიპის ეგზემპლარი შეიძლება შეიცავდეს GEOMETRY ტიპის ნებისმიერ ქვეტიპს.

3.7 ნახაზიდან ჩანს, რომ არსებობს სამი ქვეტიპი, რომლისთვისაც შეიძლება ეგზემპლარების შექმნა: CircularString, CompoundCurve და CurvePolygon. ეს ქვეტიპები ახალია SQL Server 2012 -ში და გამოიყენება სივრცით მონაცემებთან სამუშაოდ.

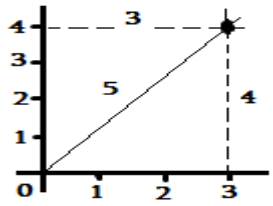
3.3.2. მონაცემთა ტიპი GEOGRAPHY

როგორც აღინიშნა, გეომეტრიული ტიპი მონაცემებს ინახავს X და Y კოორდინატებით, ხოლო გეოგრაფიული მონაცემები კი – GPS სისტემის განედისა და გრძედის გამოყენებით. გრძედი არის ჰორიზონტალური კუთხე (-180^0–:– $+180^0$) დიაპაზონში, ხოლო განედი – ვერტიკალური კუთხე (-90^0–:– $+90^0$) დიაპაზონში.

გეომეტრიულისგან განსხვავებით გეოგრაფიული ტიპისთვის უნდა მიეთითოს კონკრეტული კოორდინატთა სივრცითი სისტემა შესაბამისი მთელრიცხვა იდენტიფიკატირით SRID (Spatial Reference ID). მონაცემთა GEOMETRY ტიპის ეგზემპლარები გამოყენება აგრეთვე GEOGRAPHY ტიპისათვის.

განსხვავება GEOMETRY და GEOGRAPHY ტიპებს შორის, როგორც ადრე აღვნიშნეთ ისაა, რომ გეომეტრიული მონაცემები გამოიყენება ბრტყელ სივრცით მოდელებში, ხოლო გეოგრაფიულ კი გეოდეზიურ მოდელებში.

ძირითადი განსხვავება სივრცით მონაცემთა ამ მოდელებს შორის ისაა, რომ GEOMETRY ტიპის მონაცემებისათვის მანძილები და ფართობები მიეთითება განზომილების იმავე ერთეულებში, როგორც ეს გამოიყენებოდა ეგზემპლარებია კოორდინატებში. მაგალითად, მანძილი ორ (0,0) და (3,4) წერტილს შორის იქნება 5 ერთეული (ნახ.3.8). მართკუთხა სამკუთხედის ჰიპოტენუზა – პითაგორას თეორემის საფუძველზე.



ნახ.3.8. ორ წერტილს შორის სიგრძე გეომეტრიულ კოორდინატებში

GEOGRAPHY ტიპის მონაცემებისათვის გამოიყენება ელიფსოიდური კოორდინატები, რომლებიც იზომება განედისა და გრძედის კუთხეებში. ამასთანავე, ამ ტიპის მონაცემებზე გაითვალისწინება გარკვეული შეზღუდვები, მაგალითად, ასეთი ტიპის ყოველი ეგზემპლარი უნდა მოთავსდეს ერთი ნახევარსფეროს შიგნით.

მონაცემთა გარე ფორმატები. SQL Server მონაცემთა ასახვის მიზნით იყენებს სამი ტიპის გარე ფორმატს მათი რეალიზაციის ფორმისაგან დამოუკიდებლად:

- WKT (Well-Known Text – ცნობილი ტესტური ფორმატი) –გამოიყენება სივრცით სტრუქტურათა სივრცითი კოორდინატების სისტემების ასახვისათვისა და გარდაქმნებისათვის კოორდინატთა სისტემებს შორის;
- WKB (Well-Known Binary – ცნობილი ორობითი ფორმატი) – არის WKT-ს ორობითი ეკვივალენტი;
- GML (Geography Markup Language – გეოგრაფიული ფორმატირების ენა) – XML ენის გრამატიკაა, განსაზღვრული OGC კონსორციუმის მიერ, გეოგრაფიული თვისებების გამოსახვისათვის.

ესაა მონაცემთა გაცვლის ღია ფორმატი გეოგრაფიული ტრანზაქციების შესასრულებ-ლად ინტერნეტში.

საილუსტრაციოდ განვიხილოთ WKT ფორმატის მაგალითები:

– POINT(3,4) – მნიშვნელობები მიუთითებს X და Y კოორდინატებს;

– LINESTRING(0 0, 3 4) – პირველი ორო რიცხვი საწყისი წერტილის X და Y კოორდინატების, ხოლო მესამე და მეოთხე – ბოლო წერტილისა;

– POLYGON(300 0, 150 0, 150 150, 300 150, 300 0) – რიცხვების ყოველი წყვილი არის წერტილი მრავალკუთხედზე. ჩვენ შემთხვევაში პირველი და ბოლო წერტილები ემთხვევა.

3.4. სივრცით მონაცემებთან მუშაობა

SQL Server -ში სივრცითი მონაცემებისათვის გამოიყენება, როგორც აღვნიშნეთ, ორი ტიპი: GEOMETRY და GEOGRAPHY. ესაა მომხმარებლის მიერ განსაზღვრული მონაცემთა ტიპები, რომლებიც რეალიზდება SQL Server დანართის სახით CLR გარემოს გამოყენებით. ამ ტიპებს აქვს ქვეტიპები, რომლებიც, რომლებიც შეიძლება იყოს ეგზემპლარირებადი ან არაეგზემპლარირებადი.

ეგზემპლარირებადი ქვეტიპების შემთხვევაში შესაძლებელია ეგზემპლარების შექმნა და მათთან მუშაობა. ეს ეგზემპლარები შეიძლება შენახული იყოს როგორც ცხრილის სვეტები, ან როგორც ცვლადების მნიშვნელობები ან პარამეტრები.

არაეგზემპლარირებადი ქვეტიპებისათვის ეგზემპლარების შექმნა არ შეიძლება. კლასთა იერარქიაში ფესვური (აბსტრაქტული) კლასი არაეგზემპლარირებადია.

3.4.1. GEOMETRY ტიპის მონაცემებთან მუშაობა

მაგალითი_1: ცხრილი „უალკოჰოლო_სასმელები“ შედგება სამი სვეტისაგან. პირველი - id გენერირდება სისტემის მიერ, მეორე - name გამოიყენება სამელის დასახელების შესანახად, მესამე - shape შეიცავს ინფორმაციას ბაზრის სფეროს ფორმის შესახებ, რომელშიც მყიდველები უფრო მეტ უპირატესობას ანიჭებენ რომელიმე კონკრეტულ სამელს. პირველი სამი INSERT ინსტრუქცია ქმნის სამ სფეროს, რომლებშიც კონკრეტული სასმელია პრიორიტეტული. ყოველი ეს სამი სფერო არის მრავალკუთხედი. მეოთხე INSERT ინსტრუქცია სვამს წერტილს, ვინაიდან ეს სპეციფიკური სასმელი იყიდება მხოლოდ ერთ ადგილას.

3.1 ლისტინგზე მოცემულია ცხრილის აგების Transact_SQL პროგრამის ტექსტი.

ლისტინგიდან ჩანს მონაცემთა გეომეტრიულ ტიპებთან მუშაობის მეთოდი `GEOMETRY::STGeomFromText()`. ეს სტატიკური მეთოდი გამოიყენება გეომეტრიული ფიგურების (მრავალკუთხედი, წერტილები) კოორდინატების ჩასასმელად.

გეომეტრიულ ტიპებთან მუშაობა /ლისტინგი 3.1/

```
USE sample;
CREATE TABLE beverage_markets
 ( id INTEGER IDENTITY(1,1),
   name VARCHAR(25), shape GEOMETRY);
   INSERT INTO beverage_markets
   VALUES ('Coke1', GEOMETRY::STGeomFromText
    ('POLYGON ((0 0, 150 0, 150 150, 0 150, 0 0))', 0));
   INSERT INTO beverage_markets
   VALUES ('Pepsi', GEOMETRY::STGeomFromText
('POLYGON ((300 0, 150 0, 150 150, 300 150, 300 0))', 0));
   INSERT INTO beverage_markets
   VALUES ('7UP', GEOMETRY::STGeomFromText
('POLYGON ((300 0, 150 0, 150 150, 300 150, 300 0))', 0));
   INSERT INTO beverage_markets
   VALUES ('Almdudler', GEOMETRY::STGeomFromText
('POINT (50 0)', 0));
```

ტიპს შეიძლება ჰქონდეს მეთოდების ორი განსხვავებული ჯგუფი: სტატიკური მეთოდები და კლასის ეგზემპლარების მეთოდები. პირველი გამოიყენება მთლიანი კლასისათვის, ხოლო მეორე - კლასის სათანადო ეგზემპლარებისათვის. ამ ჯგუფთა მეთოდების გამოძახება სხვადა-სხვანაირად ხდება.

სტსტიკურისთვის გამოიყენება სიმბოლო „ :: “, ხოლო ეგზემპლარის მეთოდისათვის კი – „ . “, მაგალითად:

GEOMETRY :: STGeomFromText;

@g.STContains;

გარდა აღნიშნულისა, გამოიყენება ასევე სამი სტატიკური მეთოდებიც:

- STPointFromText() – დააბრუნებს POINT მონაცემთა ტიპის ეგზემპლარის WKT წარმოდგენას;
- STLineFromText() – დააბრუნებს LINESTRING მონაცემთა ტიპის ეგზემპლარის WKT წარმოდგენას, სიმაღლის და ზომის მნიშვნელობათა დამატებით;
- STPolyFromText() – დააბრუნებს MULTIPOLYGON მონაცემთა ტიპის ეგზემპლარის WKT წარმოდგენას, სიმაღლისა და ზომის მნიშვნელობათა დამატებით;

სივრცით მონაცემებზე მოთხოვნები სრულდება ისევე, როგორც რელაციურ მონაცემებზე. ჩვენი მაგალითისათვის, დავუშვათ, რომ ვირჩევთ ინფორმაციას beverage_markets ცხრილის shape სვეტიდან.

3.2 ლისტინგში აგებულია მოთხოვნა, რომელიც განსაზღვრავს „არსებობს თუა არა Almdudler სასმელის გამყიდველი მაღაზია იმ რაიონში, სადაც პრიორიტეტულია სასმელი Coce“.

მოთხოვნის მაგალითი_1 /ლისტინგი 3.2/

```
DECLARE @g geometry;
DECLARE @h geometry;
SELECT @h = shape FROM beverage_markets
      WHERE name ='Almdudler';
SELECT @g = shape FROM beverage_markets
      WHERE name = 'Coke';
SELECT @g.STContains(@h);
```

ეს მოთხოვნა აბრუნებს პასუხს 0-ს, რომ ამ რაიონში ასეთი მაღაზია არაა.

STContains() მეთოდი დააბრუნებს 1-ს, თუ GEOMETRY ტიპის მონაცემის ერთი ეგზემპლარი შეიცავს ამავე ტიპის მეორე ეგზემპლარს, რომელიც მეთოდის პარამეტრშია მითითებული. 3.3 ლისტინგში ასახულია მოთხოვნა, რომელიც განსაზღვრავს „shape სვეტის სიგრძეს და წარმოდგენას იმ მაღაზიისათვის, რომელიც ყიდის სასმელს Almdudler“.

მოთხოვნის მაგალითი_2 /ლისტინგი 3.3/

```
SELECT id, shape.ToString() AS wkt, shape.STLength() AS length
    FROM beverage_markets
    WHERE name = 'Almdudler';
```

შედეგი გამოიტანება შემდეგი სახით:

Id	wkt	Length
4	POINT (500)	0

ამ მაგალითში STLength() მეთოდი აბრუნებს GEOMETRY მონაცემთა ტიპის ელემენტის საერთო სიგრძეს. 0 ნიშნავს, რომ მნიშვნელობა წერტილია. ToString() მეთოდი აბრუნებს სტრიქონს wkt ფორმატში, რომელიც შეიცავს მოცემული ეგზემპლარის ლოგიკურ წარმოდგენას.

3.4 ლისტინგში განიხილება STIntersects() მეთოდის გამოყენება. კერძოდ, მოთხოვნაა: „განისაზღვროს იკვეთება თუ არა რაიონი, რომელშიც იყიდება Coce, რაიონთან, რომელშიც იყიდება Pepsi”.

მოთხოვნის მაგალითი-3 /ლისტინგი 3.4/

```
USE sample;
DECLARE @g geometry;
DECLARE @h geometry;
SELECT @h = shape FROM beverage_markets WHERE name = 'Coke'; SELECT
@g = shape FROM beverage_markets WHERE name = 'Pepsi'; SELECT
@g.STIntersects(@h);
```

STIntersects() მეთოდის გამოყენების შედეგად მიიღება 1 (TRUE), რაც ნიშნავს, რომ ეს ორი გეომეტრიული ფიგურა, რომლებიც ასახავს გაყიდვების სფეროებს, იკვეთება.

3.5 ლისტინგში განიხილება STIntersection() მეთოდის გამოყენება.

მოთხოვნის მაგალითი-4 /ლისტინგი 3.5/

```
USE   sample;
DECLARE @poly1 GEOMETRY = 'POLYGON ((1 1, 1 4, 4 4, 4 1, 1 1))';
DECLARE @poly2 GEOMETRY = 'POLYGON ((2 2, 2 6, 6 6, 6 2, 2 2))';
DECLARE Sresult GEOMETRY;
SELECT Sresult = @poly1.STIntersection(@poly2);
SELECT Sresult.STAsText();
```

შედეგად მიიღება შემდეგი სტრიქონი (დამრგვალებული მნიშვნელობებით):

POLYGON ((22, 42, 44, 24, 22))

მეთოდი STIntersection() აბრუნებს ობიექტს, რომელიც ასახავს წერტილებს, სადაც GEOMETRY მონაცემთა ტიპის ეგზემპლარი იკვეთება ამავე ტიპის სხვა ეგზემპლართან. ამიტომაც. ამ მოთხოვნის საფუძველზე მიიღება მართკუთხედი, რომელზეც იკვეთება მრავალკუთხედი (გამოცხადებული @poly1 ცვლადით) მეორე მრავალკუთხედთან (გამოცხადებული @poly2 ცვლადით).

STAsText() მეთოდი აბრუნებს შედეგს wkt ფორმატში.

3.4.2. GEOGRAPHY ტიპის მონაცემებთან მუშაობა

GEOGRAPHY ტიპის მონაცემების დასამუშავებლადაც ასევე გამოიყენება იგივე სტატიკური და ეგზემპლარის მეთოდები, რაც იყო GEOMETRY მონაცემთა ტიპებისთვის. 3.6 ლისტინგში ნაჩვენებია ერთი მაგალითი:

მოთხოვნის მაგალითი-5 /ლისტინგი 3.6/

```
USE AdventureWorks;
SELECT SpatialLocation, City
    FROM Person.Address
```

ამ მოთხოვნის შესრულების შედეგები ნაჩვენებია ქვემოთ, SpatialLocation, City ცხრილში.

AdventureWorks მონაცემთა ბაზის Address ცხრილი შეიცავს SpatialLocation სვეტს, რომელიც არის GEOGRAPHY მონაცემთა ტიპის. ჩვენი მაგალითში მოითხოვება იმ თანამშრომელთა გეოგრაფიული მდებარეობის დადგენა, რომლებიც ცხოვრობენ ქალაქ დალასში (ლისტინგი_3.7).

მოთხოვნის მაგალითი-6 /ლისტინგი 3.7/

SpatialLocation	City
0xE6100000010C4DD260393369404026C0A31BF73458C0	Dallas
0xE6100000010C10A810D1886240403A0F0653663158C0	Dallas
0xE6100000010C4346160AA26440406340F0E64F3 B58C0	Dallas
0xE6100000010C107E16DAAD6540403DA892EAD52C58C0	Dallas
0xE6100000010C8044A1422D5F4040F66D784F983758C0	Dallas
0xE6100000010C8E345943826A4040839B00B8E03358C0	Dallas
0xE6100000010CAA5BBD5FAB69404087866D198D3C58C0	Dallas

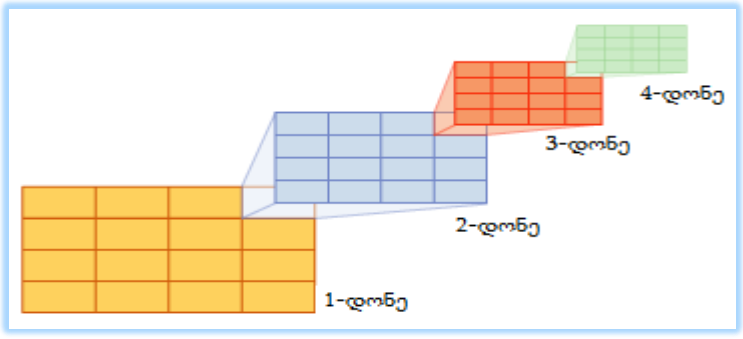
ამ ცხრილის SpatialLocation სვეტის მნიშვნელობებში ასახულია მისამართების გრძედი და განედი 16-ით კოდში თითოეული თანამშრომლისთვის ქალაქ დალასიდან.

შემდგომში ჩვენ განვიხილავთ ამ მაგალითს SQL Server Management Studio -ს გამოყენებით.

3.5. სივრცით ინდექსებთან მუშაობა

ინდექსირება, როგორც ცნობილია, გამოიყენება მონაცემებთან სწრაფი მიმართვის განსახორციელებლად. სივრცით მონაცემებთან მიმართვის დასაჩქარებლად კი საჭიროა ე.წ. სივრცითი ინდექსები, რომლებიც განისაზღვრება ცხრილის სვეტისთვის GEOMETRY ან GEOGRAPHY მონაცემთა ტიპით.

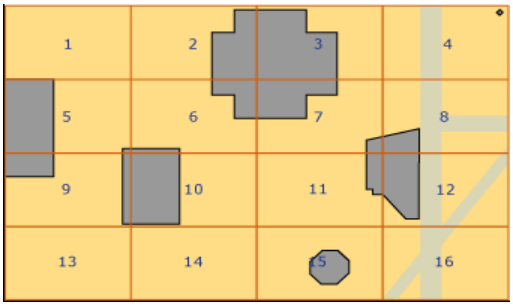
SQL Server-ში ასეთი ინდექსების ასაგებად გამოიყენება B-ხეები. ინდექსები აქ ასახავს ორ განზომილებას B-ხეების წრფივ რიგში. სივრცით ინდექსში მონაცემების მოთავსებამდე სისტემა ყოფს სივრცეს იერარქიულად ერთგვაროვანი წესით. ინდექსის შექმნის პროცესი სივრცეს ყოფს ოთხდონიანი ბადისებრი იერარქიის სახით (ნახ.3.9) [10,39]. ცხრილს უნდა ჰქონდეს კლასტერიზებული პირველადი გასაღები.



ნახ.3.9. სივრცითი ინდექსის ოთხდონიანი ბადისებრი იერარქია

ყველა დონის ბადეს უნდა ჰქონდეს უჯრედების ერთნაირი რაოდენობა (მაგალითად, 4x4 ან 8x8) და ყველა უჯრა ერთი ზომისაა. ნახაზზე ნაჩვენებია ბადის ზედა მარჯვენა უჯრის დეკომპოზიცია ბადის იერარქიის ზედა დონეებზე. დეკომპოზიცია სრულდება ყველა უჯრედისათვის. ამგვარად, მთლიანი სივრცის დეკომპოზიცია 4x4 ბადეზე ქმნის 65 536 უჯრედს მეოთხე დონეზე.

3.10 ნახაზზე ნაჩვენებია უჯრედების ნუმერაციის მაგალითი და ბადეზე განლაგებული მრავალკუთხედები (ობიექტები მაგალითად, სახლები, ქუჩები და ა.შ.).



ნახ.3.10. სივრცით ბადეზე ობიექტების განთავსება

ბადის სიმკვრივე განისაზღვრება უჯრედების რაოდენობით, რაც მეტია, მით უფრო მკვრივია იგი. მაგალითად, ბადე 8x8 (64 უჯრედით) უფრო მკვრივია, ვიდრე ბადე 4x4 (16 უჯრედით).

სივრცითი ინდექსების შესაქმნელად გამოიყენება ინსტრუქცია CREATE SPATIAL INDEX, რომლის GRIDS წინადადებით შეიძლება განსხვავებული სიმკვრივეების მითითება სხვადასხვა დონეზე (LOW - 4x4, MEDIUM – 8x8, HIGH. – 16x16).

გამოიყენება დამატებით ოფციები და წინადადებები:

- GEOMETRY_GRID – ეს წინადადება განსაზღვრავს გეომეტრიული ობიექტის ბადის ტესელაციის (tessellations) გამოყენებულ სქემას. ამ პროცესში ობიექტი თავსდება ბადისებრ იერარქიაში, უკავშირდება ბადის უჯრედებს, რომლებთანაც მას აქვს შეხება. სვეტს უნდა ჰქონდეს გეომეტრიული ტიპი;
- BOUNDING_BOX – ეს ოფცია განსაზღვრავს კომპლექტს ოთხი რიცხვითი მნიშვნელობით (მართკუთხედის ოთხი კოორდინატისთვის): Xmin და Ymin (ქვედა მარცხენა კუთხე) და Xmax და Ymax (ზედა მარჯვენა კუთხე). ეს პარამეტრი გამოიყენება მხოლოდ GEOMETRY_GRID წინადადებისთვის;
- GEOGRAPHY_GRID – ეს წინადადება განსაზღვრავს გეოგრაფიული ობიექტის ბადის ტესელაციის (tessellations) სქემას. სვეტს უნდა ჰქონდეს გეოგრაფიული ტიპი;

3.8 ლისტინგში ნაჩვენებია სივრცითი ინდექსის შექმნის მაგალითი beverage_markets ცხრილის shape სვეტისათვის.

მოთხოვნის მაგალითი-7 /ლისტინგი 3.8/

```
USE sample;
GO
ALTER TABLE beverage_markets
      ADD CONSTRAINT prim_key PRIMARY KEY(id);
      GO
      CREATE SPATIAL INDEX i_spatial_shape
           ON beverage_markets(shape)
           USING GEOMETRY_GRID
            WITH (BOUNDING_BOX = (xmin=0, ymin=0, xmax=500, ymax=200),
            GRIDS = (LOW, LOW, MEDIUM, HIGH),
            PAD_INDEX = ON);
```

სივრცითი ინდექსის შექმნა შეიძლება მახოლოდ მაშინ, თუ სივრცითი სვეტის მქონე ცხრილისათვის ცხადად არის განსაზღვრული პირველადი გასაღები. ამ მიზეზით ლისტინგის პირველ ALTER TABLE ინსტრუქციაში ეს შეზღუდვაა განსაზღვრული. შემდეგ ინსტრუქციაში CREATE SPATIAL INDEX იქმნება სივრცითი ინდექსი, გამოიყენება GEOMETRY_GRID.

პარამეტრი BOUNDING BOX იძლევა საზღვრებს, რომელშიც მოთავსდება shape სვეტის ეგზემპლარი. GRIDS პარამეტრში მიეთითება ბადის სიმკვრივე ტესელაციის სქემის ყოველ დონეზე. პარამეტრი PAD_INDEX კავშირშია FILEFACTOR პარამეტრთან, რომელიც განსაზღვრავს პროცენტებში სივრცის თავისუფალ მოცულობას ინდექსის გვერდების საერთო მოცულობიდან. PAD_INDEX პარამეტრი კი მიუთითებს, რომ FILEFACTOR პარამეტრის მნიშვნელობა გამოიყენება როგორც ინდექსის გვერდებთან, ისე ინდექსში მონაცემთა გვერდებთან.

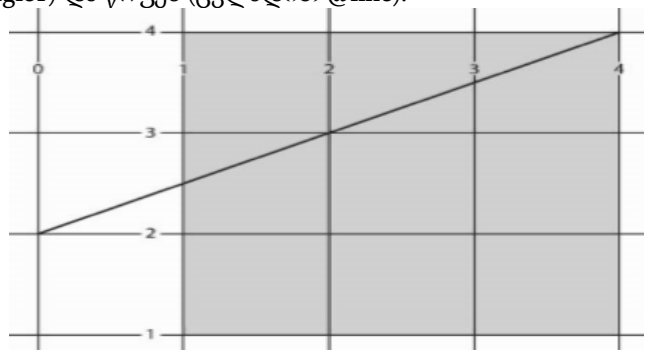
3.6. ინფორმაციის ასახვა სივრცითი მონაცემების შესახებ

SQL Server Management Studio 2012 ვერსიიდან (და უფრო ზევით) გაფართოვდა სივრცითი მონაცემების ასახვის შესაძლებლობები გრაფიკულ ფორმატში. განვიხილოთ ეს საკითხები GEOMETRY და GEOGRAPHY მონაცემების ტიპებზე, შესაბამისად 3.9 და 3.10 ლისტინგების საფუძველზე.

მოთხოვნის მაგალითი-8 /ლისტინგი 3.9/

```
USE sample;
DECLARE @rectangle1 GEOMETRY = 'POLYGON((1 1, 1 4, 4 4, 4 1, 1 1))';
DECLARE @line GEOMETRY = 'LINESTRING (0 2, 4 4)';
SELECT @rectangle1
UNION ALL
SELECT @line
```

ამ მოთხოვნის შესრულების შედეგების მისაღებად პანელზე ვირჩევთ Spatial Results ჩანართს (Results-ის მარჯვნივ). შედეგად ვიღებთ 3.11 ნახაზზე მიღებულ მართკუთხედს (ცვლადით @rectangle1) და წრფეს (ცვლადით @line).



ნახ.3.11. შედეგის გრაფიკული წარმოდგენა
(3.9 ლისტინგის მიხედვით)

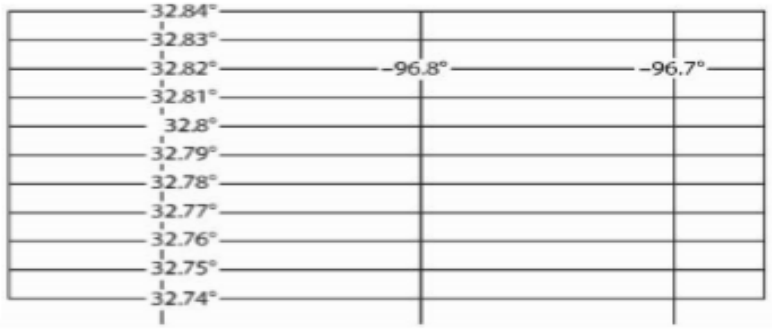
GEOMETRY ტიპის რამდენიმე ობიექტის ერთდროულად ასახვის მიზნით (ერთი ცხრილის რამდენიმე სტრიქონი), 3.9 ლისტინგში გამოყენებულია ორი SELECT ინსტრუქცია, რომლებიც ერთიანდება UNION ALL წინადადებით.

3.10 ლისტინგში ნაჩვენებია მოთხოვნა GEOGRAPHY მონაცემთა ტიპის გრაფიკული ასახვის სადემონსტრაციოდ.

მოთხოვნის მაგალითი-9 /ლისტინგი 3.10/

```
USE AdventureWorks;
SELECT SpatialLocation, City
   FROM Person.Address
   WHERE City = 'Dallas';
```

აქაც ვიყენებთ შედეგების პანელზე Spatial Results ჩანართს და მივიღებთ 3.12 ნახაზს.



ნახ.3.12. შედეგის გრფიკული წარმოდგენა
(3.10 ლისტინგის მიხედვით)

3.7. სიახლეები სივრცით მონაცემებთან სამუშაოდ

SQL Server 2012 და უფრო ახალ ვერსიებში გაუმჯობესდა სივრცით მონაცემებთან მუშაობის შესაძლებლობები კერძოდ, შემოტანილია:

- წრის რკალის ახალი ქვეტიპები;
- ახალი სივრცითი ინდექსები;
- ახალი სისტემური შენახვადი პროცედურები, დაკავშირებული სივრცით მონაცემებთან.

განვიხილოთ თითოეული მათგანი:

➢ **წრის რკალის ახალი ქვეტიპები.** ეფუძნება ANSI SQL/MM სტანდარტს. წრის რკალი ზოგადად არის მრუდის ჩაკეტილი სეგმენტი ორგანზომილებიან სიბრტყეზე. ამგვარად, რკალი წრის სეგმენტია, რომელიც მოიცემა დამოუკიდებლად ან წრფის სეგმენტთან ერთად. ამასთანავე იგი შეიძლება გამოყენებულ იქნას როგორც საფუძველი ახალი ტიპის მრავალკუთხედისათვის, რომელიც შეიცავს ერთ ან მეტ რკალურ კომპონენტს.

წრის რკალები მხარდაჭერილია GEOMETRY და GEOGRAPHY მონაცემთა ტიპებით და განისაზღვრება WKT, WKB და GML ფორმატებში.

არსებობს წრის რკალების სამი ახალი ტიპი:

- **CircularString;**

3.7 ნახაზზე GEOMETRY ტიპის იერარქიიდან ჩანს, რომ CircularString ტიპი არის Curve-ს ქვეტიპი. ამიტომაც CircularString ტიპის ობიექტები მრუდის საბაზო ქვეტიპია. CircularString ტიპის ობიექტის განსაზღვრისათვის საჭიროა მინიმუმ სამი წერტილის მოცემა. პირველი მიუთითებს რკალის დასაწყისს, მეორე - მის ბოლოს, ხოლო მესამე უნდა იყოს რკალის რომელიმე ადგილზე. CircularString ტიპის ეგზემპლარები შეიძლება გაერთიანდეს, სადაც წინა მრუდის ბოლო წერტილი ხდება მომდევნო მრუდის საწყისი წერტილი. 3.11 ლისტინგში ნაჩვენებია CircularString ტიპის ობიექტის განსაზღვრა @g ცვლადის გამოყენებით.

მოთხოვნის მაგალითი-10 **/ლისტინგი 3.11/**

```
DECLARE @g GEOMETRY;
SET @g =  GEOMETRY::STGeomFromText
      (‘CIRCULARSTRING(0 -12.5, 0 0, 0 12.5)’, 0);
```

- **CompoundCurve;**

განსაზღვრავს ახალ შედგენილ მრუდებს, რომლებიც შედგება ან მხოლოდ CircularString ტიპის ეგზემპლარებისაგან ან CircularString და LineString ეგზემპლარებისაგან. ყოველი წინა კომპონენტის ბოლო წერტილი უკავშირდება მომდევნო საწყის წერტილს. 3.12 ლისტინგში ნაჩვენებია შედგენილი მრუდის შექმნა CircularString ტიპის რამდენიმე სხვადასხვა ობიექტით.

მოთხოვნის მაგალითი-11 /ლისტინგი 3.12/

```
DECLARE @g GEOGRAPHY;
SET @g = GEOGRAPHY::STGeomFromText('
COMPOUNDCURVE(CIRCULARSTRING(0 -23.43778, 0 0, 0 23.43778),
CIRCULARSTRING(0 23.43778, -45 23.43778, -90 23.43778),
CIRCULARSTRING(-90 23.43778, -90 0, -90 -23.43778),
CIRCULARSTRING(-90 -23.43778, -45 -23.43778, 0 -23.43778) )' ,4326);
```

აქ იქმნება GEOGRAPHY მონაცემთა ტიპის ეგზემპლარი, რომელიც მიენიჭება @g ცვლადს. ეს ცვლადი შედგება CompoundCurve ტიპის ეგზემპლარისაგან, რომელიც თავის მხრივ შედგება CircularString ტიპის ეგზემპლარებისაგან. საყურადღებოა STGeomFromText() მეთოდის ბოლო არგუმენტის მნიშვნელობა 4326. ესაა SRID (Spatial reference ID) იდენტიფიკატორი GEOGRAPHY მონაცემთა ტიპისათვის და შეესაბამება WGS_82 კოორდინატთა სივრცით სისტემას [11].

- **CurvePolygon.**

CurvePolygon ტიპის ობიექტი შედგება LineString და CircularString ტიპის ობიექტებისაგან, აგრეთვე CompoundCurves ტიპის ობიექტებისგან. 24.6 ნახაზიდან ჩანს, რომ CurvePolygon არის უშუალო ქვეტიპი Surface ტიპისა და სუპერტიპი Polygon ტიპისა. ასეთი წრის შიგნით CurvePolygon ობიექტის მომდევნო კომპონენტის პირველი წერტილი ემთხვევა მომდევნო კომპონენტის ბოლო წერტილს.

- **ახალი სივრცითი ინდექსები**

SQL Server-ში GEOMETRY и GEOGRAPHY ტიპის მონაცემებისათვის დამატებულია ახალი სივრცითი ინდექსი auto_grid_index. მისი ფუნქციონალობა ორივესთვის მსგავსია, ამიტომ მხოლოდ ერთს განვიხილავთ.

ამ ინდექსში გამოიყენება ტესელაციის რვა დონე სხვადასხვა ზომის ობიექტის უკეთესი აპროქსიმაციის მიზნით.

3.13 ლისტინგში ნაჩვენებია ავტომატური ინდექსის შექმნა გეომეტრიული ობიექტისათვის.

მოთხოვნის მაგალითი-12 /ლისტინგი 3.13/

```
CREATE SPATIAL INDEX auto_grid_index
    ON beverage_markets(shape)
    USING GEOMETRY_AUTO_GRID
        WITH (BOUNDING_BOX = (xmin=0, ymin=0, xmax=500, ymax=200),
        CELLS_PER_OBJECT = 32, DATA_COMPRESSION = page);
```

3.8. ახალი სისტემური შენახვადი პროცედურები სივრცითი მონაცემებისთვის

SQL Server 2012 ვერსიიდან სივრცითი ტიპის მონაცემებისათვის დამატებულია ახალი სისტემური შენახვადი პროცედურები:

- sp_help_spatial_geometry_index;
- sp_help_spatial_geography_index;

რომელთა გამოყენების სინტაქსიც ორივესი მსგავსია.

sp_help_spatial_geometry_index სისტემური შენახვადი პროცედურა აბრუნებს უკან გეომეტრიული ობიექტის სივრცითი ინდექსის მითითებული თვისებების ერთობლიობის დასახელებებს და მნიშვნელობებს. შედეგები აისახება ცხრილურ ფორმატში. აგრეთვე შესაძლებელია ძირითად თვისებათა ერთობლიობის ან ინდექსის ყველა თვისების მოთხოვნა. 3.14 ლისტინგში მოცემულია ამ სისტემური შენახვადი პროცედურის კოდი.

მოთხოვნის მაგალითი-13 **/ლისტინგი 3.14/**

```
DECLARE @query geometry
 ='POLYGON((-90.0 -180.0, -90.0 180.0, 90.0 180.0, 90.0 -180.0, -90.0 -180.0))' ;
EXEC sp_help_spatial_geometry_index 'beverage_markets',
                                    'auto_grid_index', 0, @query;
```

ამ მაგალითში sp_help_spatial_geometry_index სისტემური პროცედურა ასახავს auto_grid_index სივრცითი ინდექსის თვიებებს, რომელიც შეიქმნა 3.12 ლისტინგით.

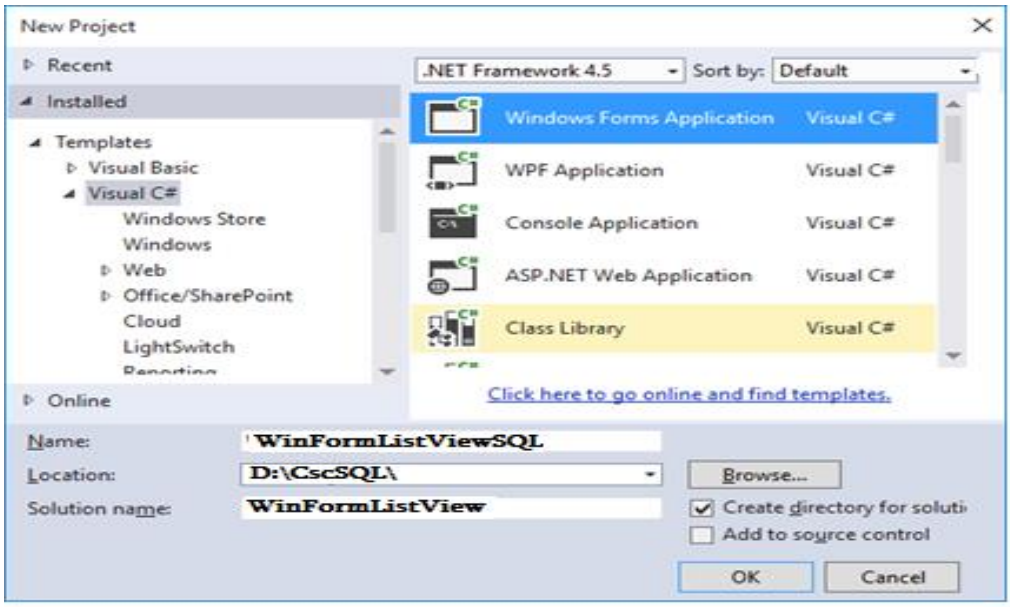
IV თავი
შავი ზღვის ეკომონიტორინგის სისტემის პროგრამული აპლიკაციის აგება

4.1. შავი ზღვის საქართველოს მდინარეების მონაცემთა ბაზის აგებისა და განახლების ინტერფეისის დამუშავების ამოცანა

განიხილება ვიზუალური დაპროგრამების C# ენის ListView კლასის საფუძველზე მონაცემთა მენეჯმენტის საკითხი, SQL Server ბაზის განახლების მეთოდების Insert, Update და Delete ასაგებად [70,71].

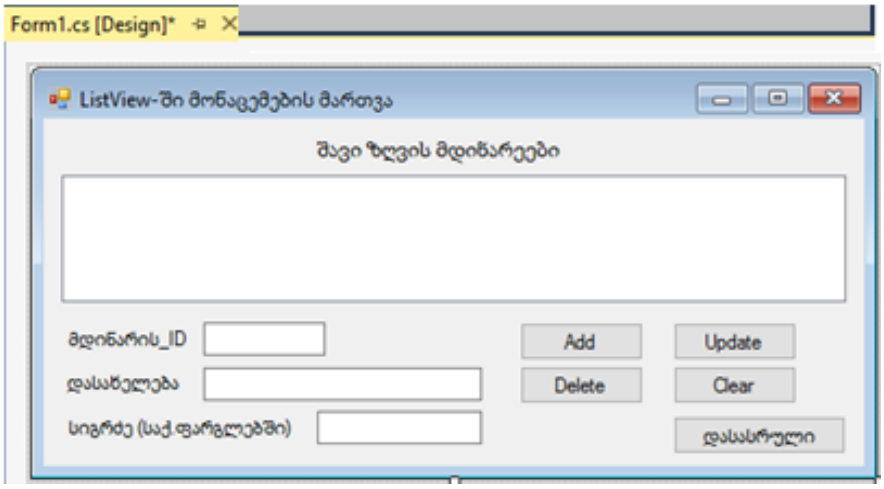
პროგრამული პროექტის აგების მაგალითისათვის განვიხილოთ ამოცანა საქართველოს აკვატორიაში შავი ზღვის მდინარეების მონაცემთა ბაზის შექმნისა და მისი ცხრილების შევსების და განახლების ინტერფეისული პროგრამის დამუშავება.

Visual Studio.NET 2013/15 გარემოში შევქმნათ ახალი პროექტი სახელით: WinFormListViewSQL (ნახ.4.1).



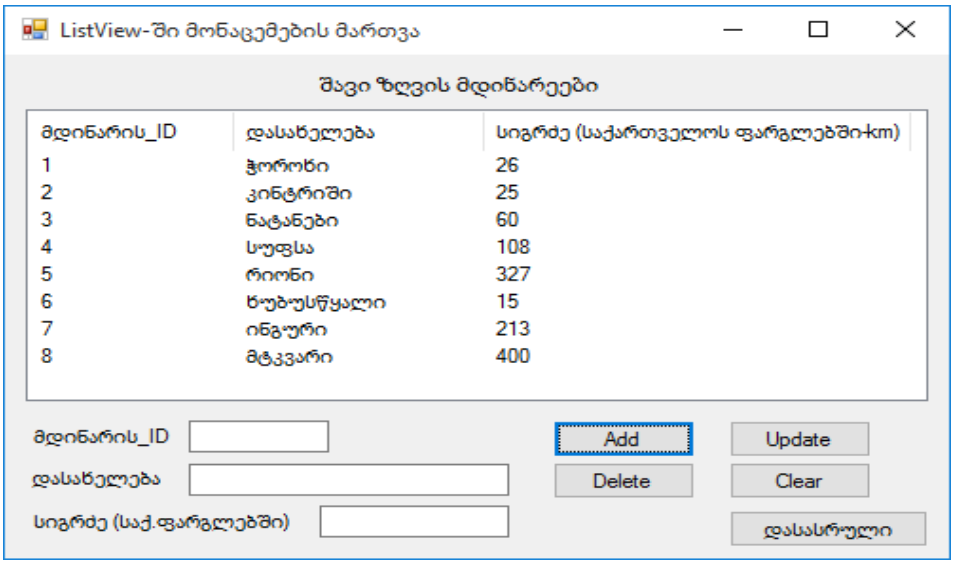
ნახ.4.1

ჩვენ მიერ დაპროექტებული ინტერფეისის ფორმა მოცემულია 4.2 ნახაზზე.



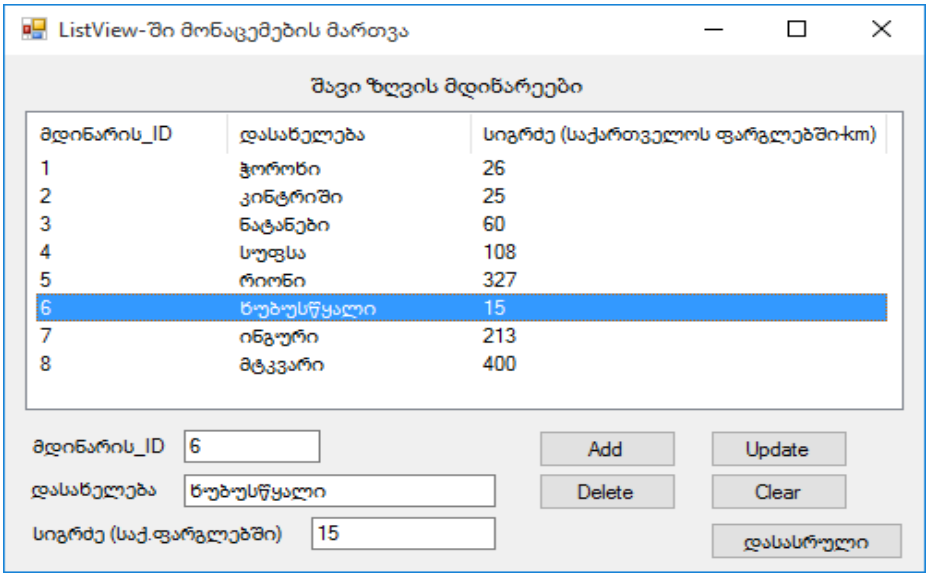
ნახ.4.2

მონაცემთა ბაზის ჩანაწერებიგამოიტანება ListView სვეტებში (ნახ.4.3).



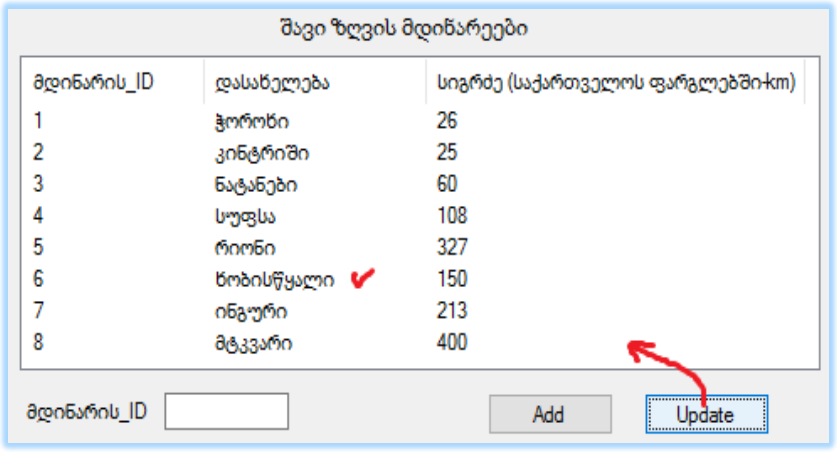
ნახ.4.3

მე-6 სტრიქონში შეცდომაა და საჭიროა ცვლილების განხორციელება Update მეთოდით. ვირჩევთ სტრიქონს (ნახ.4.4).



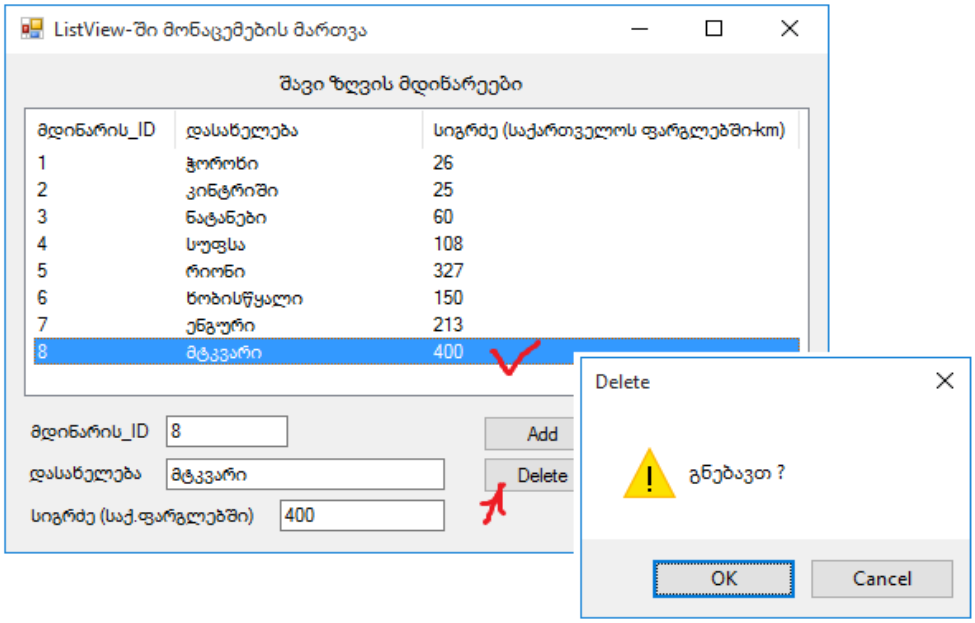
ნახ.4.4

„ხუბუსწყალი“ შევცვალოთ „ხობისწყალი“-თ (ნახ.4.5).



ნახ.4.5

დარჩა ერთი შესასწორებელი სტრიქონი - მდინარის სახელი „ინგური“ უნდა შეიცვალოს სახელით „ენგური“, ხოლო „მტკვარი“ არაა შავი ზღვის მდინარე, ამიტომ ის უნდა წაიშალოს სიიდან Delete მეთოდით (ნახ.4.6).



ნახ.4.6

4.1_ლისტინგში მოცემულია ListView-ში მონაცემების დამატების, ცვლილებისა და წაშლის მეთოდების კოდები.

```
// -- ლისტინგი_4.1 ---- ListView-ის  Add, Update Delete მეთოდები --
using System;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WinFormListViewSeqD
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
      // ListView-ის თვისებები -----
```

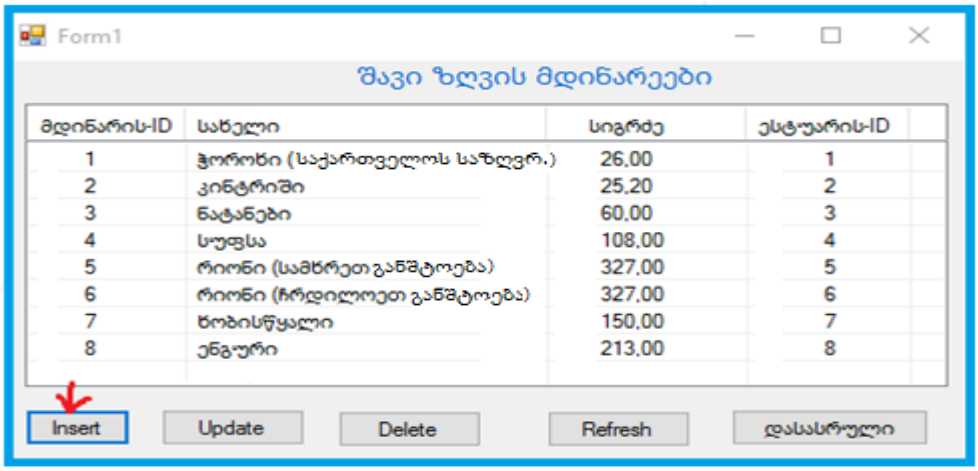
```
      listView1.View = View.Details;
      listView1.FullRowSelect = true;
      // ListView-ის ველების სიგანე-----
      listView1.Columns.Add("მდინარის_ID", 50);
      listView1.Columns.Add("დასახელება", 150);
      listView1.Columns.Add("სიგრძე (საქართველოს
                ფარგლებში-კმ)",50);
    }
 // Add
private void add(String id, String name, String length)
 {
   //row---
   String[] row = {id, name, length};
   ListViewItem item = new ListViewItem(row);
   listView1.Items.Add(item);
 }
// update-----
private void update()
 {
  listView1.SelectedItems[0].SubItems[0].Text =
       idTxt.Text;
  listView1.SelectedItems[0].SubItems[1].Text =
       nameTxt.Text;
  listView1.SelectedItems[0].SubItems[2].Text =
       lengthTxt.Text;
  // clear txt
   idTxt.Text = "";
   nameTxt.Text = "";
   lengthTxt.Text = "";
 }
  // delete -----
  private void delete()
   {
    if (MessageBox.Show("გნებავთ წაშლა ?", "DELETE",
                  MessageBoxButtons.OKCancel,
          MessageBoxIcon.Warning) == DialogResult.OK)
      {
```

```
listView1.Items.RemoveAt(listView1.SelectedIndices[0]);
       // clear txt
        idTxt.Text = "";
        nameTxt.Text = "";
        lengthTxt.Text = "";
      }
   }
private void button1_Click(object sender, EventArgs e)
  {
    add(idTxt.Text, nameTxt.Text, lengthTxt.Text);
    // clear txt
      idTxt.Text = "";
      nameTxt.Text = "";
      lengthTxt.Text = "";
  }
private void button2_Click(object sender, EventArgs e)
  {
      update();
  }
private void button3_Click(object sender, EventArgs e)
  {
      delete();
  }
private void button4_Click(object sender, EventArgs e)
  {
    listView1.Items.Clear();
    // clear txt
    idTxt.Text = "";
    nameTxt.Text = "";
    lengthTxt.Text = "";
   }
 private void listView1_MouseClick(object sender,
            MouseEventArgs e)
   {
idTxt.Text=listView1.SelectedItems[0].SubItems[0].Text;
nameTxt.Text=listView1.SelectedItems[0].SubItems[1].Text;
```

```
lengthTxt.Text=listView1.SelectedItems[0].SubItems[2]
        .Text;
   }
private void button5_Click(object sender, EventArgs e)
   {
       Close();
   }
  }
}
```

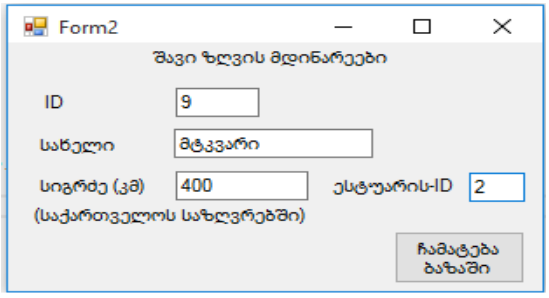
ამოცანა_2: ახლა განვიხილოთ ListView-ში მონაცემების გამოტანა SQL Server-დან და მასში ცვლილებების განხორციელება.

4.7 ნახაზზე მოცემულია ListView საწყისი ფანჯარა ბაზიდან ამოღებული სტრიქონებით:



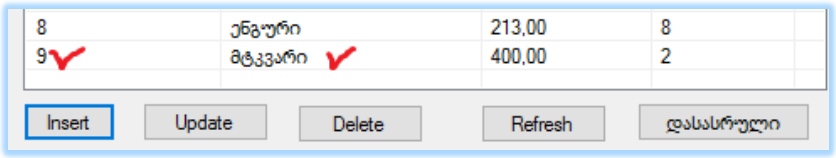
ნახ.4.7

Insert ღილაკის არჩევით გამოიტანება 4.8 ფანჯარა და ვამატებთ ახალ სტრიქონს :



ნახ.4.8

მდინარე მტკვარი ჩაემატა SQL Server ბაზაში და განახლდა ListView სია თავიდან ახალი ვერსიით (ნახ.4.9).



ნახ.4.9

- Insert მეთოდის შესაბამისი კოდი მოცემულია 4.2 ლისტინგში

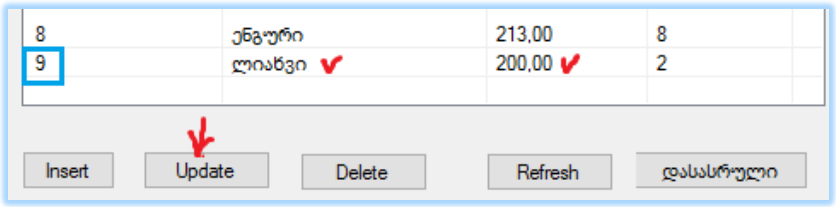
```
public partial class Form1 : Form
{  // SQL Server ბაზასთან მიერთება----
  SqlConnection con = new SqlConnection(@"Data Source=gtu-205A-08;Initial
Catalog=SeaEco;Integrated Security=True");
...
//... ლისტინგი_4.2 ---- Insert() ------
private void button1_Click(object sender, EventArgs e)
      {
          Form2 frm = new Form2(nID+1, null);
          //დამატება მოვლენის დასაჭერად Form2-დან---
          frm.UpdateListView += new
               EventHandler(frm_UpdateListView);
          frm.Show();
      }
  void frm_UpdateListView(object sender, EventArgs e)
        {
         //როგორც კი მონაცემები დაემატება Form2-ში,
         // მოვლენა განაახლებს ListView1-ს----
           ListViewLoad();
        }
```

- Delete მეთოდის შესაბამისი კოდი მოცემულია 4.3 ლისტინგში

```
//.. ლისტინგი_4.3 ---- Delete
private void button4_Click(object sender, EventArgs e)
      {
          SqlDataAdapter dass = new SqlDataAdapter();
          con.Open();
          DataTable dt = new DataTable();
          string deleete_r = listView1.SelectedItems[0].SubItems[0].Text;
```

```
        try
          {
             dass.DeleteCommand = new
          SqlCommand(@"DELETE FROM River WHERE riverID =
                    @riverID", con);

        dass.DeleteCommand.Parameters.Add("@riverID",
            SqlDbType.Int).Value = int.Parse(deleete_r);
        dass.DeleteCommand.ExecuteNonQuery();
        con.Close();
            }
          catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            ListViewLoad();
        }
```

4.10 ნახაზზე მოცემულია Update მეთოდის შედეგი, ანუ დავდექით მე-9 ჩანაწერზე და ავამოქმედეთ Update ღილაკი. ამ მოვლენამ აამუშავა Update() მეთოდი. შევიტანეთ მდინარე „ლიახვი 200 კმ“. კოდი ნაჩვენებია 4.4 ლისტინგში.



ნახ.4.10

```
// ლისტინგი_4.4 - -- Update() ---
private void button3_Click(object sender, EventArgs e)          {
            Form2 frm = new Form2(nID, listView1);
            frm.UpdateListView += new EventHandler(frm_UpdateListView);
            frm.Show();
        }
```

შეცვლილი ბაზის ხელახალი ჩატვირთვა ListView ფანჯარაში ხორციელდება შემდეგი კოდით (ლისტინგი 4.5).

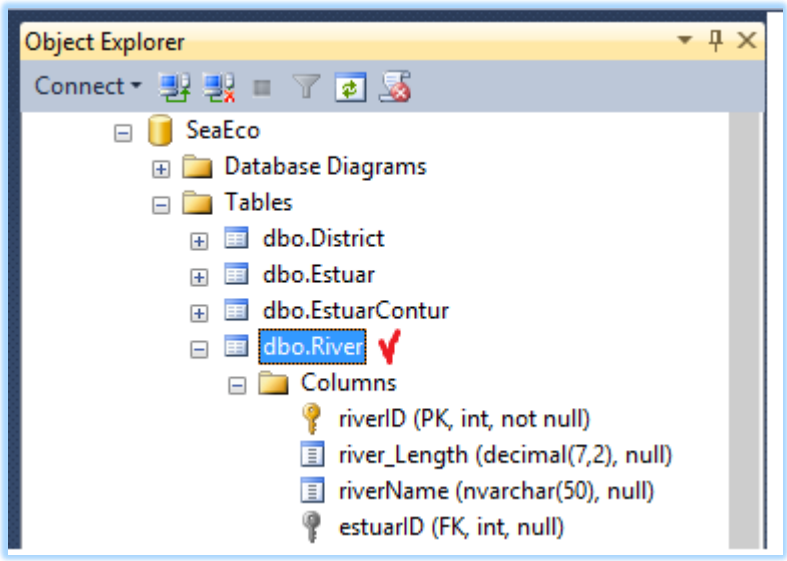
```
// -- ლისტინგი_4.5 ---- ---
void ListViewLoad()
 {
```

```
  SqlDataAdapter da = new SqlDataAdapter("SELECT riverID,
    riverName, river_Length, estuarID FROM River", con);
  DataTable dt = new DataTable();
  // მონაცემების გადატანა ბაზიდან ცხრილში ---
   da.Fill(dt);
  // ListView-ს შევსება ცხრილიდან---
   listView1.Items.Clear();
   listView1.View = View.Details;
   listView1.GridLines = true;

  foreach (DataRow dr in dt.Rows)
   {
     ListViewItem lvi = new
           ListViewItem(dr["riverID"].ToString());
    lvi.SubItems.Add(dr["riverName"].ToString());
    lvi.SubItems.Add(dr["river_Length"].ToString());
    lvi.SubItems.Add(dr["estuarID"].ToString());
    listView1.Items.Add(lvi);
    // გლობალური ცვლადი (int), გამოცხადებულია დასაწისში---
     nID =int.Parse( dr["riverID"].ToString());
   }
}
```

4.2. მონაცემთა ბაზის განახლება ADO.NET დრაივერისა და DataGridView კლასის გამოყენებით

ამოცანა_3. მოცემულია Ms SQL Server მონაცემთა ბაზა (მაგალითად, „შავი ზღვის ეკოლოგიური სისტემა“), რომლის ერთ-ერთი ცხრილი (Table) არის River.dbo (მდინარეები). საჭიროა ავაგოთ C# პროექტი (მომხმარებლის ინტერფეისი), რომელიც მონაცემთა ბაზიდან ამოიღებს მდინარეების მონაცემებს DataGridView ცხრილში, შეძლებს Insert, Update და Delete ოპერაციების განხორციელებას. გამოყენებულ უნდა იქნას ADO.NET დრაივერის საშუალებები.

4.11 ნახაზზე ნაჩვენებია საწყისი მონაცემთა ბაზა, რომელიც Ms SQL Server 2012 ვერსიაშია რეალიზებული. (ა) შეესაბამება ბაზის ცხრილების იერარქიას და აქვე ჩანს River ცხრილის სტრუქტურაც, მონაცემთა შესაბამისი ტიპებით. (ბ)-ზე მოცემულია ჩანაწერები, რომელთა შეტანა მოხდა წინასწარ (თუმცა ჩვენი პროგრამისათვის არაა აუცილებელი მისი არსებობა. შესაძლებელია იგი შეივსოს ინტერფეისიდან).



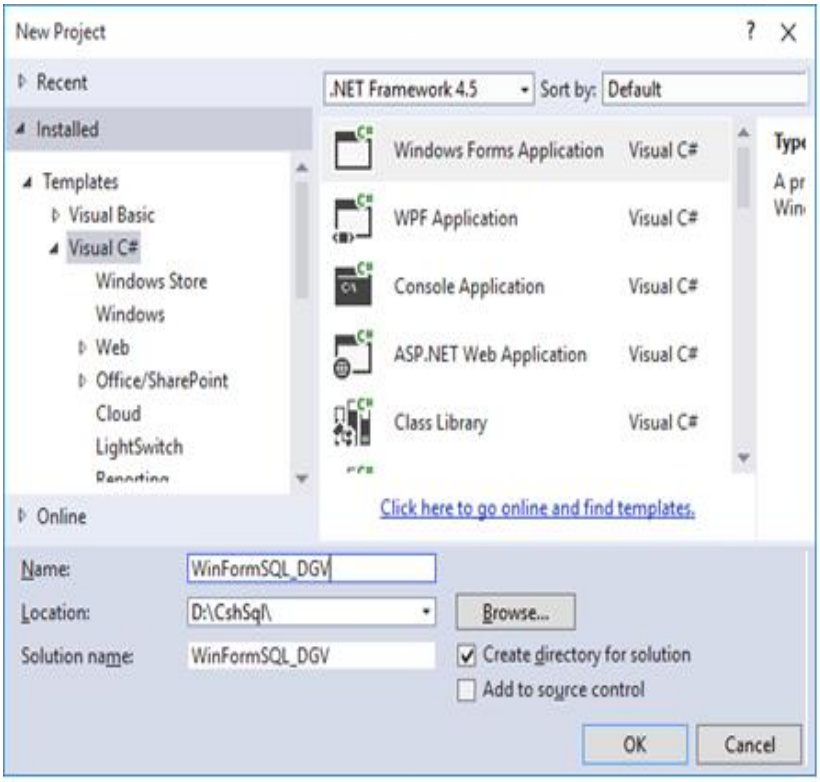
ნახ.4.11-ა. მონაცემთა ბაზა Ms SQL Server-ში

GTU-205A-08.SeaEco - dbo.River | GTU-205A-08.SeaEco - dbo.River

	riverID	river_Length	riverName	estuarID
▶	1	26,00	ჭოროხი (საქართველოს საზღვრებში)	1
	2	25,20	კინტრიში	2
	3	60,00	ნატანები	3
	4	108,00	სუფსა	4
	5	327,00	რიონი (სამხრეთ განშტოება)	5
	6	327,00	რიონი (ჩრდილოეთ განშტოება)	6
	7	150,00	ხობისწყალი	7
	8	213,00	ენგური	8
*	NULL	NULL	NULL	NULL

ნახ.4.11-ბ. River (მდინარეების) საწყისი ცხრილი
Ms SQL Server-ში

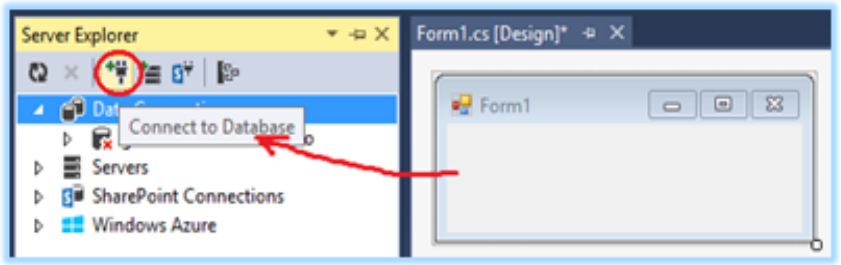
დავიწყოთ ახალი პროექტის აგება Ms Visual Studio.NET 2013/15 სამუშაო გარემოში. 4.12 ნახაზზე ნაჩვენებია პროექტის შექმნის პროცედურა.



ნახ.4.12. WinFormSQL_DGV პროექტის შექმნა

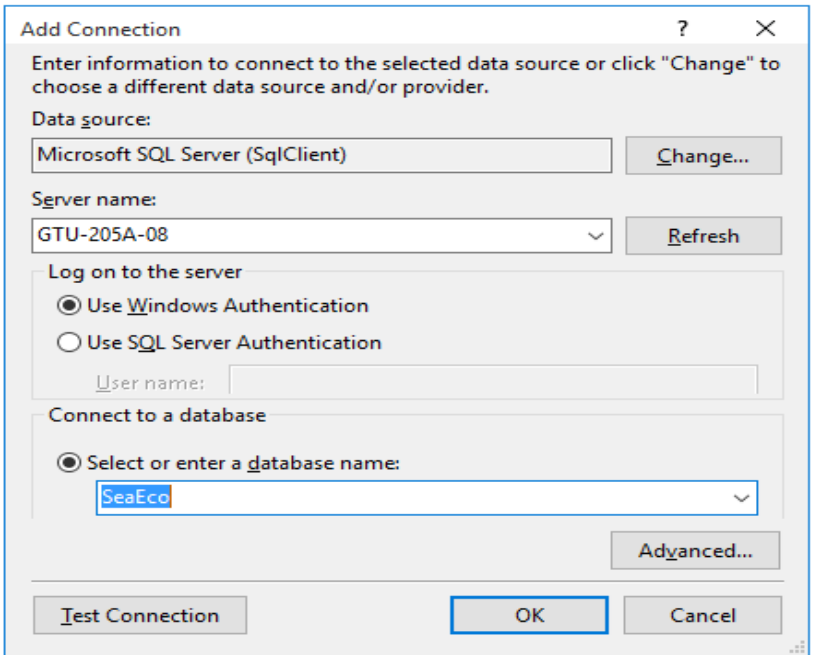
ვირჩევთ პროექტის სახელს (Name), მისი შენახვის ადგილს (Browse-ს დახმარებით) და Solutionname-ს. შემდეგ OK.

საჭიროა პროექტისთვის განვახორციელოთ მონაცემა ბაზის მიერთება (Connect to Database), რაც 4.13 ნახაზზეა ნაჩვენები.



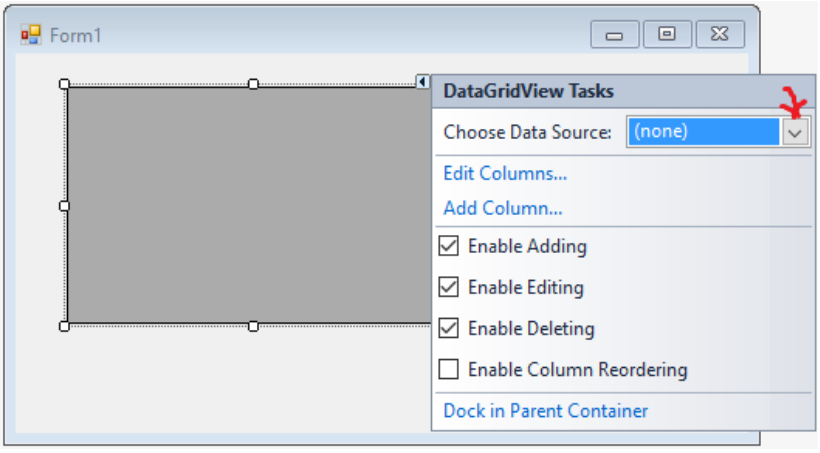
ნახ.4.13. Connect Database ამოქმედება

გამოიტანება ახალი ფანჯარა (ნახ.4.14), რომელშიც უნდა შეირჩეს შესაბამისი წყარო (Data Source), სერვერი (Server name) და მონაცემთა ბაზა (Database name).



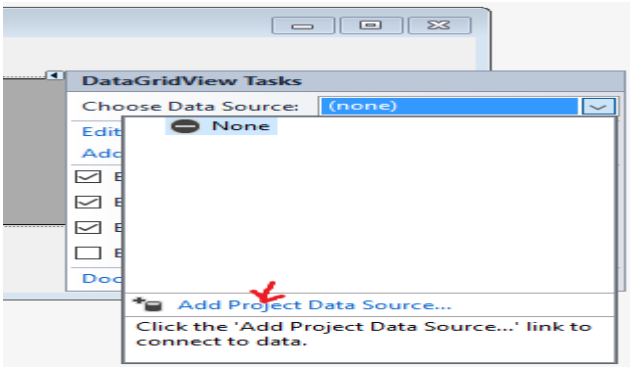
ნახ.4.14. მონაცემთა წყაროს, სერვერისა და ბაზის არჩევა

ინსტრუმენტების პანელიდან ფორმაზე გადავიტანოთ DataGridView ელემენტი და ზედა მარჯვენა კუთხე პატარა ისრით ავამოქმედოთ. მივიღებთ 4.15 ნახაზს.



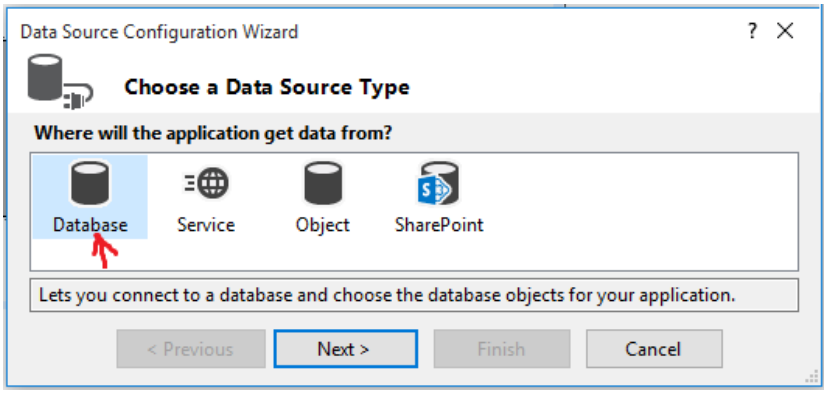
ნახ.4.15. პარამეტრების განსაზღვრა

ნახაზზე ჩანს, რომ ჩამატების, რედაქტირებისა და წაშლის ოპერაციები ნებადართულია (ჩეკბოქსები მონიშნულია). ავირჩიოთ Choose Data Source კომბობოქსის ღილაკი, მივიღებთ 4.16 ნახაზზს.



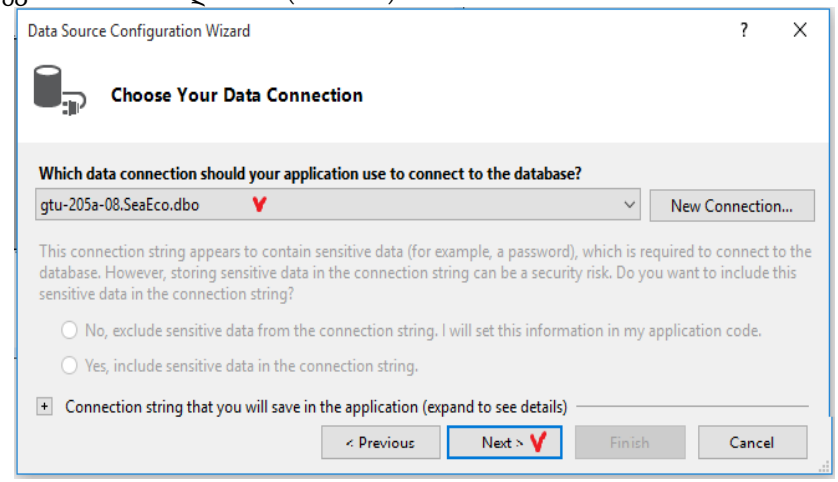
ნახ.4.16. მონაცემთა წყაროს პროექტის დამატება

ავამოქმედოთ Add Project Data Source და გადავიდეთ 4.17 ნახაზზე.



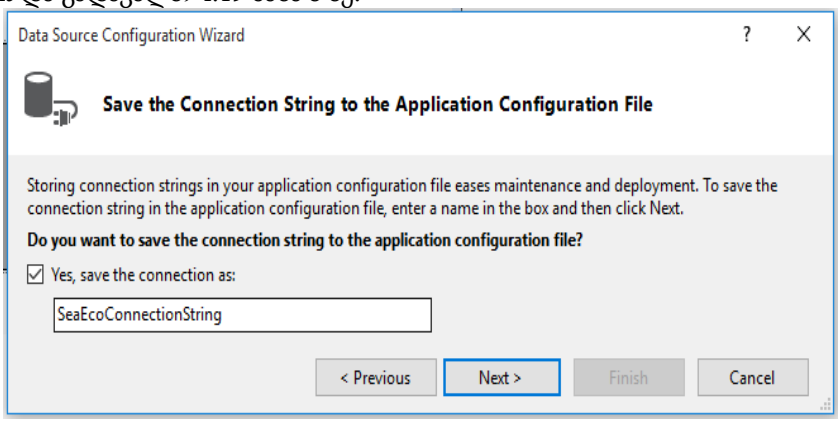
ნახ.4.17. მონაცემთა წყაროს ტიპის არჩევა

ვირჩევთ Database-ს და Next (ნახ.4.18).



ნახ.4.18. მონაცემთა Connection (მიერთების) შერჩევა

Connection პარამეტრი ყველა კომპიუტერს ექნება თავისი. მისი განსაზღვრა შესაძლებელია Server Explorer-იდან (ჩვენ შემთხვევაში იგი არის: GTU-205a-08.SeaEco.dbo). ბოლოს Next და გადავალთ 4.19 ნახაზზე.



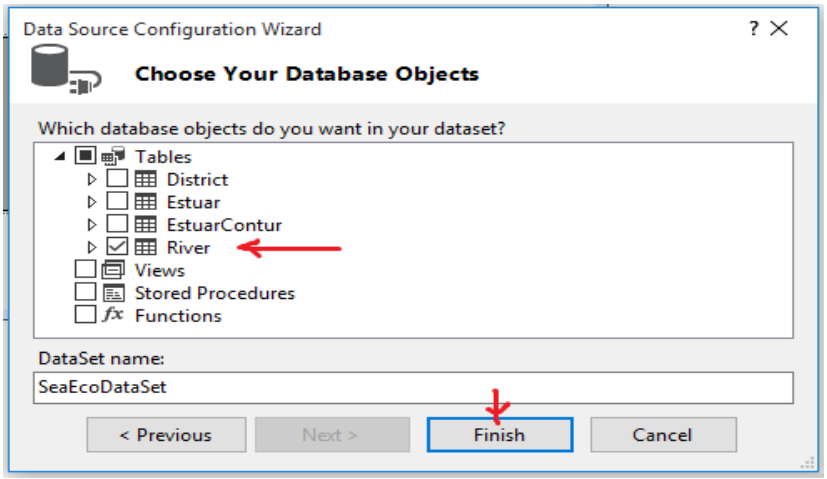
ნახ.4.19. Connection String-ის შენახვა აპლიკაციის კონფიგურაციის ფაილში

შემდეგ გამოიძახება მონაცემთა ბაზის ობიექტების არჩევის ფანჯარა (ნახ.4.20).

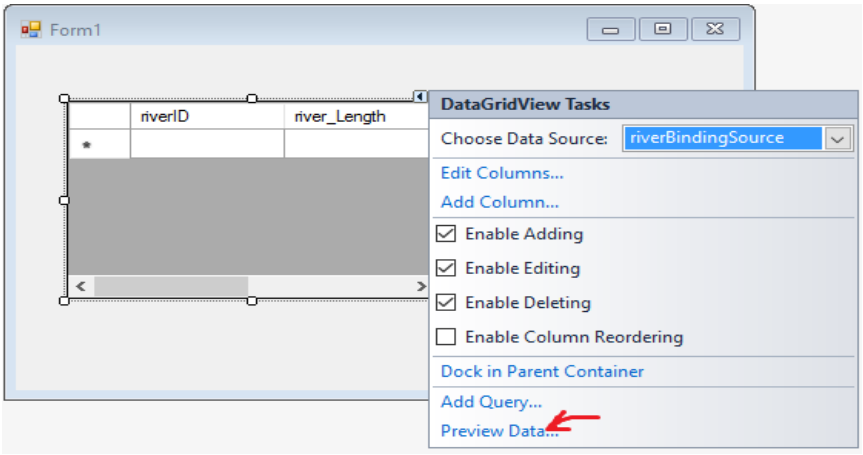
4.21 ნახაზზე ნაჩვენებია მონაცემთა წყაროს (Data Source) განსაზღვრის შედეგის მნიშვნელობა, ჩვენ შემთხვევაში იგი არის:

riverBindingSource.

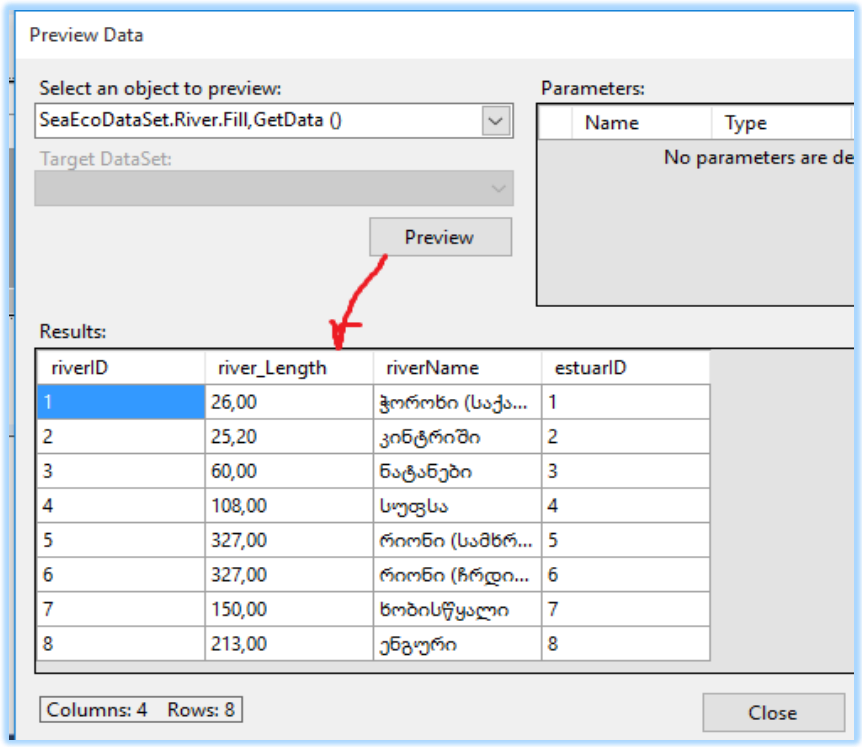
აქვე შეიძლება გამოვიყენოთ Preview Data ღილაკი და დავათვალიეროთ წინასწარ მიერთებული ბაზის ცხრილის ჩანაწერები (ნახ.4.22).



ნახ.4.20. ობიექტების მონიშვნა (მაგალითად, River)

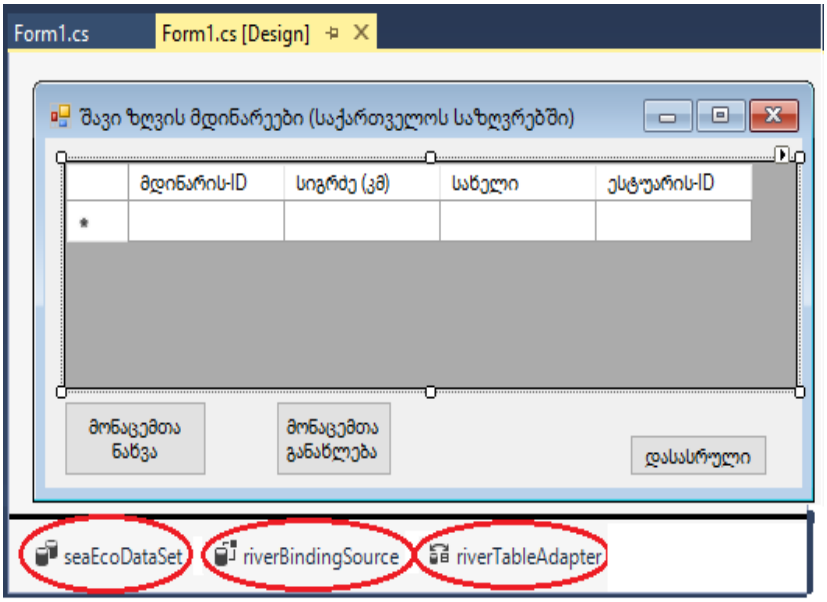


ნახ.4.21. Data Source შედეგი: "riverBindingSource"



ნახ.4.22. Preview Data ცხრილი

ბოლოს, Form1-ს Properties-ში შევუცვალოთ სახელი „შავი ზღვის მდინარეები (საქართველოს საზღვრებში)", ინსტრუმენტების პანელიდან გადმოვიტანოთ სამი ღილაკი (Button1,2.3), დავარქვათ სახელები. მიიღება 4.23 ნახაზი.



ნახ.4.23. პროექტის ძირითადი ინტერფეისი

ახლა გადავიდეთ ღილაკების ფუნქციების დაპროგრამებაზე, ანუ უნდა განხორციელდეს SQL Server -შესაბამისი ბაზის ცხრილის ჩანაწერების დათვალიერება, ჩამატება, შეცვლა და წაშლა.

თავდაპირველად ჩავამატოთ სახელსივრცის სტრიქონი:

```
using System.Data.SqlClient;
```

შემდეეგ გამოვაცხადოთ გლობალური ცვლადები:

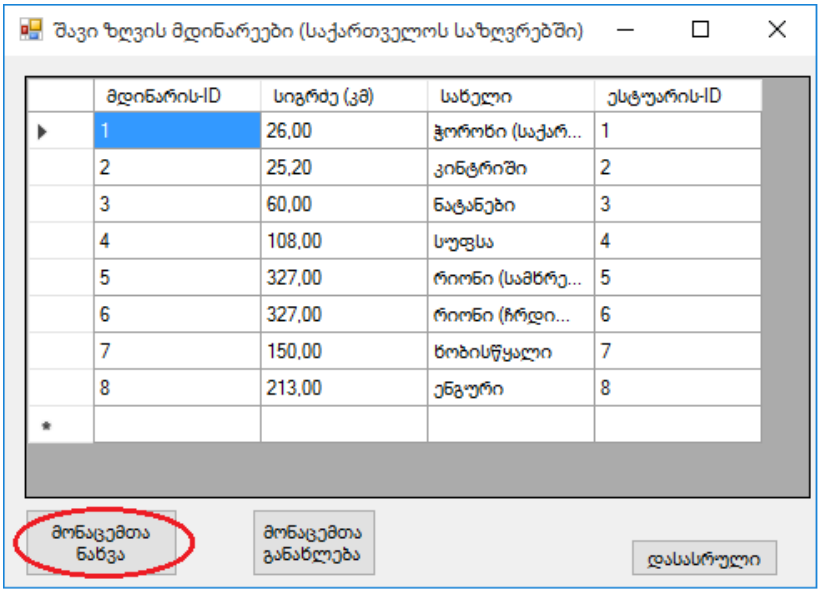
```
SqlDataAdapter sda;
SqlCommandBuilder scb;
DataTable dt;
```

„მონაცემთა ნახვის“ ღილაკისთვის (button1) გვექნება შემდეგი კოდი:

```
private void button1_Click(object sender, EventArgs e)
 {
   SqlConnection con = new SqlConnection("Data Source=
                GTU-205A-08;Initial Catalog=SeaEco;
                Integrated Security=True");
   sda = new SqlDataAdapter(@"SELECT riverID,
                    river_Length, riverName,
                    estuarID from River", con);
```

```
    dt = new DataTable();
    sda.Fill(dt);
    dataGridView1.DataSource = dt;
}
```

პროგრამის ამუშავებით, თუ მასში არაა შეცდომები, მიიღება 4.24 ნახაზი.



ნახ.4.24. „მონაცემთა ნახვის“ (DataShow) ღილაკის ამოქმედებით მიღებული შედეგი

„მონაცემთა განახლების“ ღილაკის კოდი (Insert, Update, Delete) ნაჩვენებია ქვემოთ:

```
private void button2_Click(object sender, EventArgs e)
  {
    scb = new SqlCommandBuilder(sda);
    sda.Update(dt);
  }
```

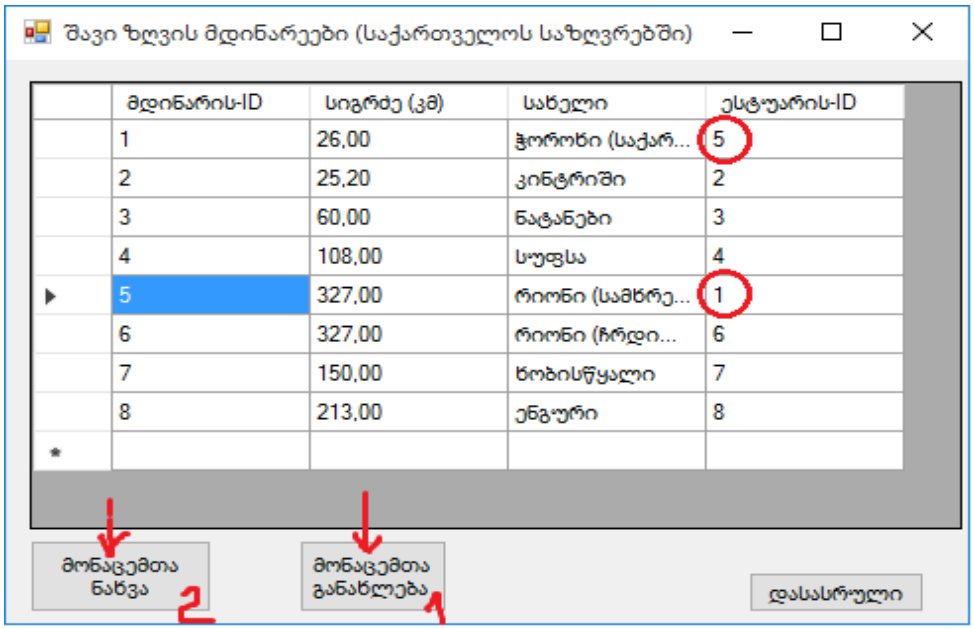
მთლიანი პროგრამის კოდი მოცემულია 4.6 ლისტინგში.

```
// --- ლისტინგი_4.6 --- Insert, Update, Delete for DataGridView_SQL----
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

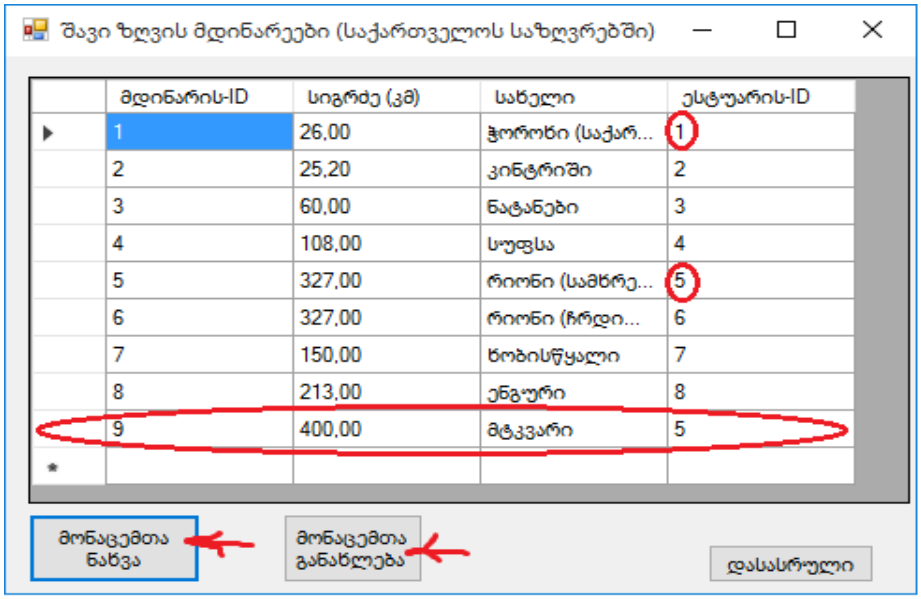
```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace WinFormSQL_DGV
{
  public partial class Form1 : Form
   {
      SqlDataAdapter sda;
      SqlCommandBuilder scb;
      DataTable dt;
      public Form1()
       {
          InitializeComponent();
       }
    private void Form1_Load(object sender, EventArgs e)
     {

     }
       // მონაცემთა ნახვა
    private void button1_Click(object sender, EventArgs e)
    {
      SqlConnection con = new SqlConnection("Data
          Source=GTU-205A-08;Initial Catalog=SeaEco;
               Integrated Security=True");
      sda = new SqlDataAdapter(@"SELECT riverID,
                  river_Length, riverName, estuarID
                  from River", con);
      dt = new DataTable();
      sda.Fill(dt);
      dataGridView1.DataSource = dt;
    }

   // განახლება
   private void button2_Click(object sender, EventArgs e)
      {
         scb = new SqlCommandBuilder(sda);
```

```
            sda.Update(dt);
        }

    private void button3_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

4.25 ნახაზზე მოცემულია არსებულ ცხრილში ორი მნიშვნელობის (ესტუარისID) შეცვლა (ახალი რიცხვები ნაჩვენებია წრეში), შემდეგ 1-ელი და მე-2 ღილაკების ამოქმედებით ბაზაში ჩაიწერება ეს შეცვლილი მნიშვნელობები (შეიძლება დავხუროთ პროგრამა და თავიდან ავამუშავოთ).

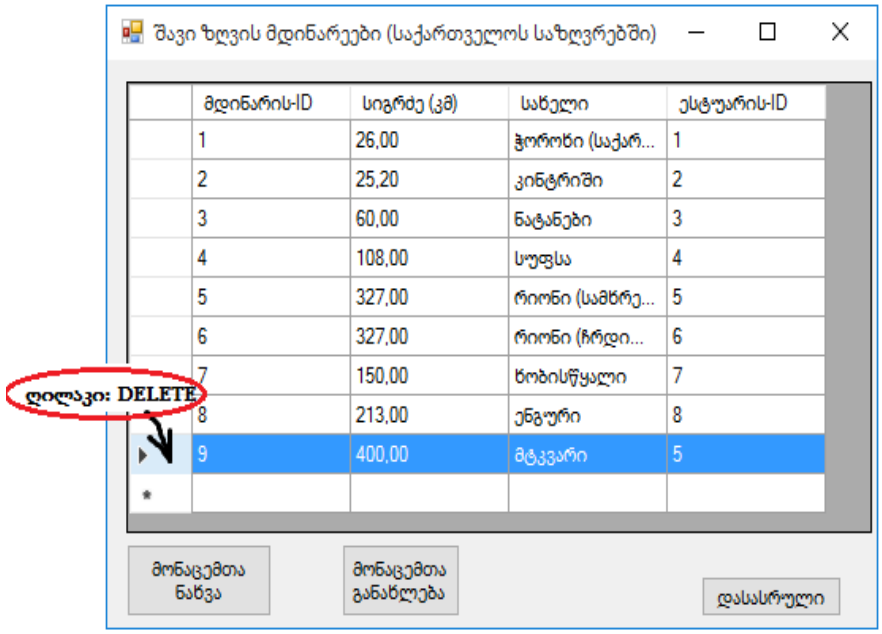


ნახ.4.25. Update ცვილების განხორციელება



ნახ.4.26. Insert ოპერაციის განხორციელება

ბოლოს ნაჩვენებია სტრიქონის წაშლის (Delete) ოპერაცია. იგი ხორციელდება, მაგალითად, „**მდინარის-ID**“ შესაბამისი სტრიქონის მონიშვნით და შემდეგ კომპიუტერის კლავიატურის Delete- ღილაკის ამოქმედებით (ნახ.4.27).



ნახ.4.27. Delete ოპერაციის განხორციელება

4.3. ობიექტ როლური მოდელი (ORM) და მონაცემთა ბაზის დაპროექტების ავტომატიზაცია

როგორც ცნობილია, მონაცემთა ბაზის აპლიკაციის ხარისხი კრიტიკულადაა დამოკიდებული მის დაპროექტებაზე. საპრობლემო სფეროს ინფორმაციული სტრუქტურის დაპროექტება თავდაპირველად სასურველია მოხდეს კონცეპტუალურ დონეზე. იგი გვეხმარება დავაფიქსიროთ მომხმარებლისაგან მიღებული სემანტიკა და მოვახდინოთ მოდელის რეალიზაცია სხვადასხვა პლატფორმაზე.

ბიზნეს-პროცესი არის „ორგანიზმი“, რომელიც განუწყვეტლივ ვითარდება. მოდელირებისა და მოთხოვნების ენას უნდა შეეძლოს დააფიქსიროს ამ სფეროში მიმდინარე რთული პროცესები და ამავე დროს მარტივი უნდა იყოს მასში ცვლილებების შეტანა. სწორედ ასეთი ლინგვისტური სტრუქტურაა - ობიექტ-როლური მოდელირება (ORM) [16,72-74].

ობიექტ-როლური მოდელირება ესაა დაპროექტებისა და მონაცემთა მოდელირების მეთოდი კონცეპტუალურ დონეზე, სადაც აპლიკაცია აღწერილია მომხმარებლისათვის გასაგებ ენაზე, ნაცვლად იმისა, რომ წარმოდგენილ იქნეს მონაცემთა სტრუქტურების ტერმინებში.

იგი საპრობლემო არეს აღწერს, როგორც ობიექტებს, რომლებიც გარკვეულ როლებს ასრულებს. ბუნებრივი ენის და ინტუიციური დიაგრამების (რომელთა ჩაწერაც ხდება მაგალითებით) გამოყენება და ასევე საპრობლემო სფეროს აღწერა ელემენტარული ფაქტების საფუძველზე საგრძნობლად ამარტივებს დაპროექტების პროცესს. ეს ფაქტები შეიძლება დაყოფილ იქნას უფრო მცირე ფაქტებად, ინფორმაციის დაკარგვის გარეშე.

ობიექტ-როლური მოდელირების ადრეული ვერსია 1970-იან წლებში გამოჩნდა ევროპაში. მას შემდეგ იგი იყო გაფართოებული და დახვეწილი მკვლევარების მიერ ავსტრალიაში, ევროპაში, აშშ-სა და სხვა ქვეყნებში. ტერი ჰალპინს თავის სტატიებში დაწვრილებით აქვს აღწერილი ობიექტ-როლური მოდელირების პროცესები [16].

კონცეპტუალური მოდელირებისათვის ORM-მოდელს გააჩნია გარკვეული უპირატესობები არსთა დამოკიდებულებათა მოდელთან (ER-მოდელთან) შედარებით, რადგან ხსნის ყოველგვარ ბარიერს დამპროექტებელსა და კლიენტს შორის.

ზემოაღწერილი მიზეზებიდან გამომდინარე კონცეპტუალური მოდელირებისათვის ჩვენ ვირჩევთ ORM-ს. საინფორმაციო სისტემების ცხოვრების ციკლი მოიცავს რამდენიმე სტადიას: ტექნიკურ-ეკონომიკური დასაბუთება, მოთხოვნათა ანალიზი, მონაცემებისა და ოპერაციების კონცეპტუალური დაპროექტება; ლოგიკური დაპროექტება; გარე დაპროექტება; მაკეტირება; შიგა დაპროექტება და შესრულება; ტესტირება და შესწორების შეტანა; მომსახურება (თანხლება) (პარაგრაფი 1.1).

ORM-ის კონცეპტუალური მოდელირების სქემის პროცედურა ანუ CSDP (Conceptual Sschema Ddesign Procedure) ყურადღებას ამახვილებს მონაცემების ანალიზსა და დაპროექტებაზე. კონცეპტუალური სქემა აღწერს აპლიკაციის ინფორმაციულ სტრუქტურას:

ფაქტების ტიპები, რომლებიც წარმოადგენს ინტერესის სფეროს; მასზე არსებული შეზღუდვები და შესაძლოა წარმოქმნის (პროდუქციის) წესები, რათა მივიღოთ ესა თუ ის ფაქტი სხვა ფაქტებიდან [72,73].

თვითონ CSDP შედგება შვიდი ბიჯისაგან:

1. ელემენტარული ფაქტების ფორმირება და მათი ადეკვატურობის შემოწმება;
2. ფაქტების ტიპებისათვის დიაგრამის აგება და სისრულის შემოწმება;
3. იმ ობიექტთა ტიპების შემოწმება, რომლებიც უნდა გაერთიანდეს და მათი მათემატიკური წარმომავლობის დაფიქსირება;
4. დაემატოს უნიკალურობის შეზღუდვა და შემოწმდეს ფაქტების ტიპების ოპერანდების რაოდენობა;
5. დაემატოს როლების იძულებითი შეზღუდვები და შემოწმდეს მათი ლოგიკური წარმომავლობა;
6. დაემატოს ელემენტები, სიმრავლეთა შედარება და ქვეტიპის შეზღუდვები;
7. დაემატოს სხვა შეზღუდვები და მოხდეს საბოლოო შემოწმება.

ORM დიაგრამაში გამოიყენება შემდეგი ტიპის შეზღუდვები სქემის აგების დროს:

- ***იძულების შეზღუდვები:***

• – ობიექტი ასრულებს ზუსტად გარკვეულ როლს.

⊙ – როლების დიზუნქცია არის იძულებითი. ყოველი ობიექტი უნდა ასრულებდეს მხოლოდ ერთ როლს ობიექტების ტიპების ნაკრებიდან.

- ***უნიკალურობის შეზღუდვები:***

⟷ – ერთ ან მეტ როლში მონაწილეობა ხდება არა უმეტეს ერთხელ.

⊗ – როლების გარე უნიკალურობის შეზღუდვა (წყვილის გამორიცხვის შეზღუდვა).

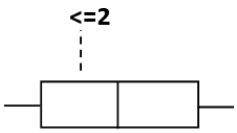
- ***სიმრავლეების შედარების შეზღუდვები:***

⊆ – პირველი ობიექტის სიმრავლე ყოველთვის უნდა იყოს მეორის ქვესიმრავლე.

= – პირველი ობიექტის სიმრავლე ყოველთვის უნდა იყოს მეორის ტოლი.

⊄ – პირველი ობიექტის სიმრავლე არ შედის მეორეში.

- ***სიხშირის შეზღუდვა:***



– ობიექტმა რამდენჯერ შეიძლება შეასრულოს ეს როლი.

- ***ბუდის ტიპის ობიექტი:***

– ობიექტი მხოლოდ ერთ როლს ასრულებს და ეს როლი არ არის სავალდებულო.

- *ქვეტიპი:*

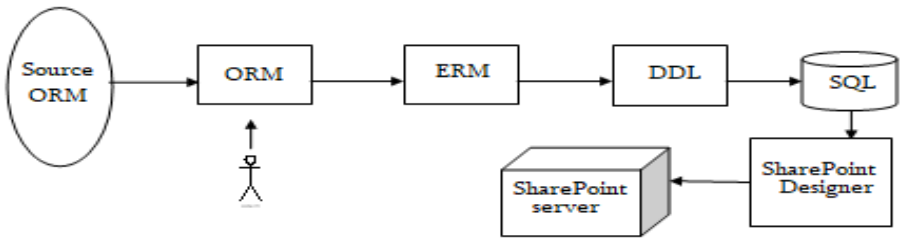
– ერთი ობიექტი არის მეორის ქვეტიპი

- *წრიული შეზღუდვები:*

ანტირეფლექსურობა	O*ir*	iff for all *x*, ˜*xRx*
სიმეტრიულობა	O*sym*	iff for all *x, y, xRy* → *yRx*
ასიმეტრიულობა	O*as*	iff for all *x, y, xRy* → ˜*yRx*
ანტისიმეტრიულობა	O*ans*	iff for all *x, y, x* ≠ *y & xRy* → ˜*yRx*
ანტიტრანზიტულობა	O*it*	iff for all *x, y, z, xRy & yRz* → ˜*xRz*
აციკლურობა	Oac	iff for all *x, y, z, xRy & yRz* → ˜*zRx*

მას შემდეგ, რაც მოხდება ORM-დიაგრამის აგება ყველა წესის დაცვით, შესაძლებელი იქნება ავტომატურად არსთა-დამოკიდებულების ER-მოდელის აგება.

Microsoft Visual Studio.NET-ის NORMA (Natural ORM Architect)- პროგრამული პაკეტი საშუალებას იძლევა ფაქტებზე დაყრდნობით დავაპროექტოთ ობიექტ-როლური მოდელი [77]. შემდეგ, ORM-დიაგრამიდან ავტომატურად ავაგოთ EM-მოდელი, რომლის საფუძველზეც შეიქმნება რელაციურ მონაცემთა ბაზების ლოგიკური სტრუქტურის აღწერა, ანუ .DDL ფაილები. SQL-Server-ის ან სხვა მონაცემთა ბაზების მართვის სისტემაში .DDL ფაილები ავტომატურად ააგებს ლოგიკურ და ფიზიკურ სტრუქტურებს, რომელსაც SharePoint Designer-ის საშუალებით მივუერთებთ SharePoint Server-ს (ნახ.4.28).



ნახ.4.28. მონაცემთა ბაზის ავტომატიზებული დაპროექტების პროცესი

4.4. შავის ზღვის ეკომონიტორინგის სისტემის კონცეპტუალური ORM/ERM მოდელების აგება ეკოლოგიური პარამეტრებისათვის

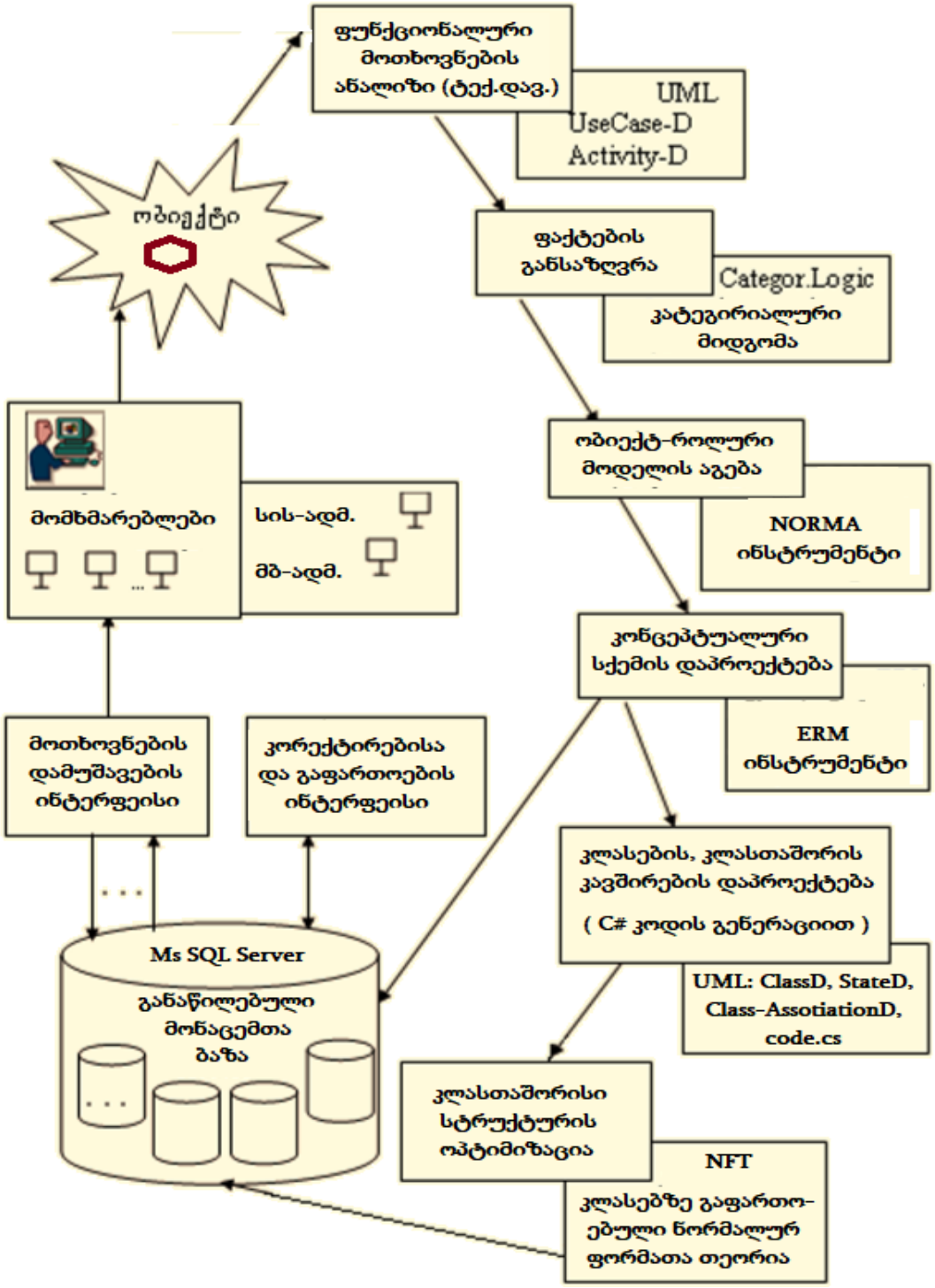
ჩვენი კვლევის საპრობლემო სფეროა შავი ზღვის ეკოლოგიური მონიტორინგის სისტემა, კერძოდ მისთვის მონაცემთა ბაზის დაპროექტების პროცესის ავტომატიზაცია. საწყის ეტაპზე საჭიროა განისაზღვროს ის ობიექტები, რომლებიც აღწერს სინტაქსურად და სემანტიკურად ზღვის ეკოსისტემის ძირითად პარამეტრებს. ჩვენ მიერ ჩატარებული სისტემური ანალიზის საფუძველზე გამოიკვეთა შემდეგი ობიექტები:

- ზღვა (SeaID, Name, Length_EastWest, Length_NorthSouth,Area, Water_volume, Average_depth, Max_depth);
- **მდინარე** (RiverID, მდინარის_დასახელება, წყალშემკრები_აუზის_ფართობი, კმ^2, აბსოლუტური_ნიშნული, მ, მდინარის_სიგრძე, კმ, საშუალო_ქანობი, i, აუზის_საშუალო_ სიმაღლე_მონაკვეთზე, მ, ჩამონადენის_საშუალო_მოდული, ლ/წმ.კმ^2, საშუალო წლიური ხარჯი მ^3/წმ);
- **ესტუარი** (EstuarID, RiverID, CoordGPSx, CoordGPSy, Area,);
- **მოწყვლადი_უბანი** (Vulnerable_districtsID, CoordGPSx, CoordGPSy, Area, T1/T2, pH, TDS);
- GPS_კოორდინატები (CoordGPSx, CoordGPSy);
- **სენსიტიური_უბანი** (SensitiveAreasID, CoordGPSx, CoordGPSy)
- **უბანი** (DistrictID, Name, CoordGPSx, CoordGPSy, Area, T1/T2, pH, TDS);
- წყლის_სინჯის_ფაქტორები (WaterTestID, WaterT1, AirT2, Water_acidityPH, WaterSalinityTDS);
- ეკოლოგიური_პარამეტრი (...);
- რაოდენობრივი_მახასიათებელი (...);
- თვისობრივი_მახასიათებელი (...);
- შავიზღვის_ეკოლოგიური_პრობლემები (...);
- ეკო_უსაფრთხოების_ღონისძიება (ActionID, Name, DateBegin, DateEnd, …);
- საზღვაო_პორტი (PortID, DistrictsName, CoordGPSx, CoordGPSy);

და სხვ.

მონაცემთა ბაზის დაპროექტება უნდა განვახორციელოთ ობიექტროლური მოდელირების ინსტრუმენტისა და მისი პრინციპების საფუძველზე [76,156]. ინსტრუმენტის სახით ვიყენებთ Natural ORM Architect პაკეტს, რომელიც თავსებადია Visual Studio.NET Framework ინტეგრირებულ სისტემასთან [9,72,76].

კონცეპტუალური მოდელი (ORM) ან სქემა არის საპრობლემო სფეროს ძირითად ტერმინთა ერთობლიობა და მათ შორის კავშირები, რომლებიც ასახავს საკვლევი სფეროს ბიზნეს-პროცესებს და ბიზნეს-წესებს. იგი თეორიულად ეფუძნება კატეგორიალური მიდგომის (ენის გრამატიკული წესები) მათემატიკური ლოგიკის (ალგებრის) ერთობლივ გამოყენებას [72]. მთლიანი ტექნოლოგიური პროცესი მოცემული გვაქვს 4.29 ნახაზზე.



4.29. საპრობლემო სფეროს მონაცემთა ბაზის დაპროექტება ORM ტექნოლოგიით

ასეთი მიდგომა ჩადებულია NORMA-ინსტრუმენტში, რომელიც დამკვეთ-მომხმარებლის ცოდნას დასაპროექტებელი ობიექტების შესახებ გადაიტანს ე.წ. ობიეტების, მათი თვისებების და პრედიკატების (ბინარული,... , n-არული) სახით.

ობიექტების აღწერა მომხმარებლის მიერ ხდება NORMA პაკეტის სამუშაო ინტერფეისით და შეიტანება ჯერ ერთი ობიექტი, შემდეგ მეორე და ა.შ.

ბოლოს თვით NORMA-სისტემა გვაძლევს ინტეგრირებულ კონცეპტუალურ მოდელს, რომელიც 4.30 ნახაზზეა ნაჩვენები.

ჩვენ საილუსტრაციოდ აღვწერეთ სამი ობიექტი: „მდინარე“ (River), „ესტუარი“ (Estuar) და „უბანი“ (District). მათ შორის კავშირები აგებულია „has“ („is”, „works” და სხვ.) პრედიკატებით.

პრედიკატები არის ფაქტები, მაგალითად:

f1: River has RiverName

f2: River has RiverLength

f3: River has Estuar

f4: Estuar has River

. . .

f15: District has Category

f16: District_Category is Normal or Sensitive or Vulnerable

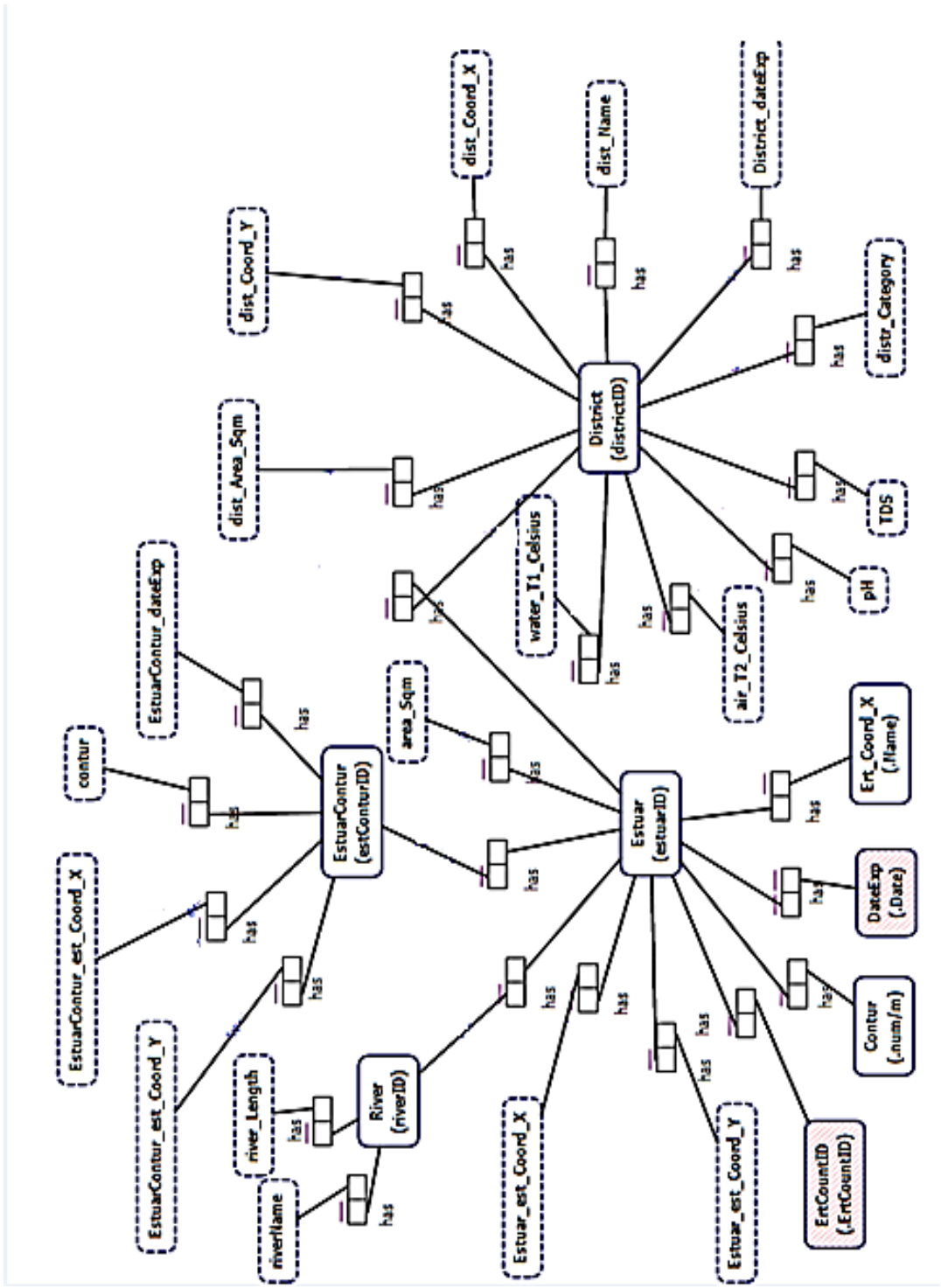
და ა.შ.

ობიექტს „უბანი“ აქვს პარამეტრი „უბნის_კატეგორია“, რომელიც არის მნიშვნელობა სიმრავლიდან {ნორმალური, სენსიტიური, მოწყვლადი}. თუ რომელი იქნება კონკრეტული უბანი, დამოკიდებულია მისი ეკოლოგიური პარამეტრების მნიშვნელობებზე.

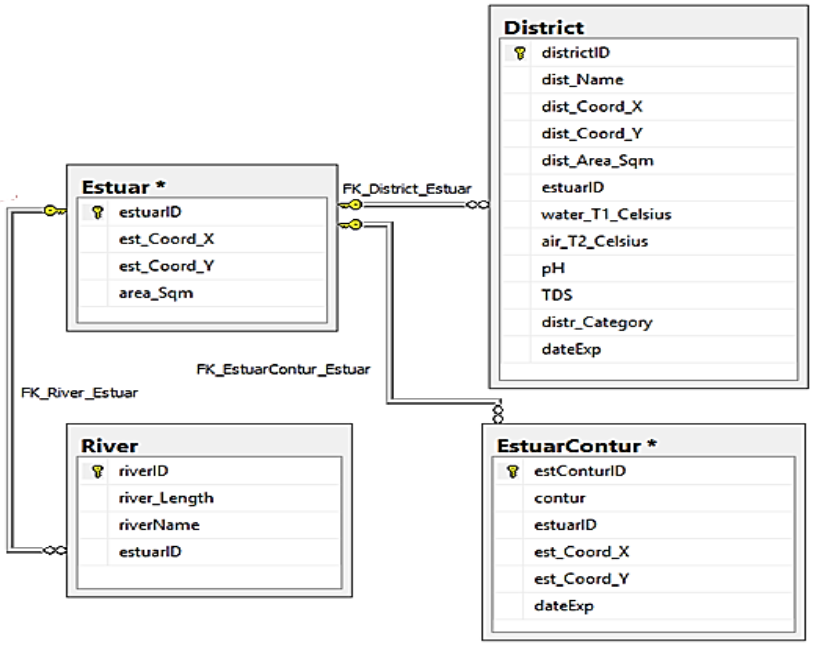
შეიძლება ითქვას, რომ ORM-მოდელირების ინსტრუმენტის გამოყენებით პირველი დონის კონცეპტუალური სქემის აგება შეუძლია არაპროგრამისტ (მონაცემთა ბაზების აგების არმცოდნე) მომხმარებელსაც. მან იცის საპრობლემო სფეროს არსი, ამოცანები, ფუნქციები და ამიტომ, იგი, შედარებით მცირე კონსულტაციის შემდეგ, NORMA გარემოში ადვილად საქმიანობს - გადააქვს თავისი ცოდნა კომპიუტერში. შედეგად მიიღება მონაცემთა ბაზის ORM მოდელი (ნახ.4.30) [12].

რა თქმა უნდა, შესაძლებელია შედეგში იყოს უზუსტობები, რომლებიც ქსელის მოდიფიკაციის რეჟიმში ადვილად სწორდება თვით მომხმარებლის მიერ მანამ, სანამ არ მიიღება საბოლოო მომხმარებლისათვის მისაღები კონცეპტუალური მოდელი.

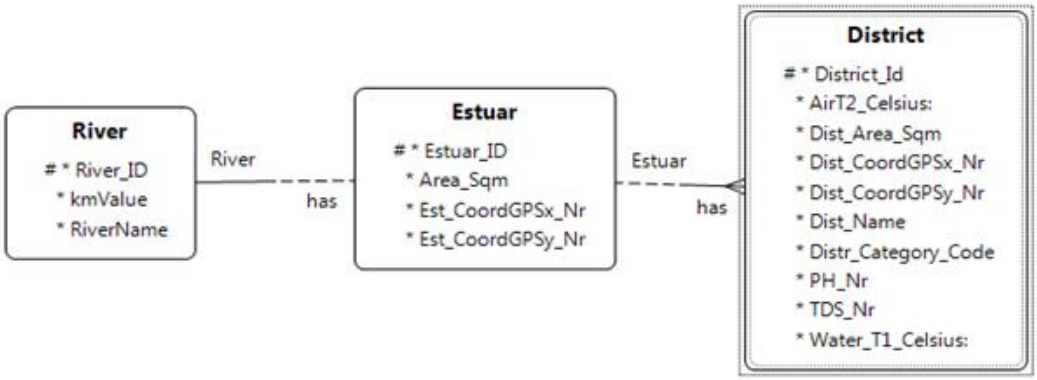
შემდეგი ეტაპი ეხება ORM მოდელის (კონცეპტუალური სქემის) საფუძველზე არსთა დამოკიდებულების მოდელის (ERM-სქემის) დამუშავებას. 4.31 ნახაზზე ნაჩვენებია მისი კლასიკური ვარიანტი, რომელიც ავტომატურად მივიღეთ VS.NET გარემოში NORMA პაკეტიდან. 4.32 ნახაზზე მოცემულია ალტერნატიული ვარიანტი - ბარკერის მოდელი (მას აქტიურად იყენებს ფირმა Oracle) [71].



ნახ.4.30. ეკომონიტორინგის სისტემის ORM სქემის ფრაგმენტი



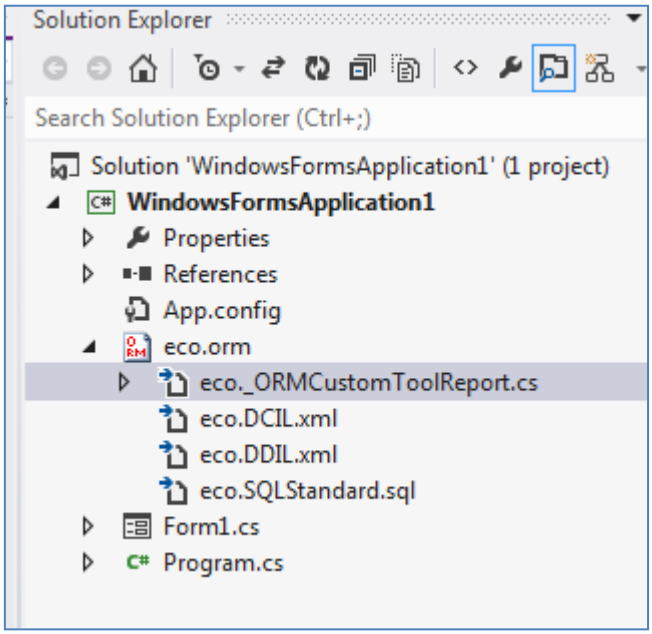
ნახ.4.31. ERM სქემის ფრაგმენტი (კლასიკური მოდელი)



ნახ.4.32. ბარკერის კონცეპტუალური მოდელი

ER-მოდელიდან SQL Server-ის ცხრილების მისაღებად, საჭიროა შემდეგი მოქმედებების შესრულება

Visual Studio.Net გარემოში შევქმნათ ახალი Windows Form Application პროექტი და მივუერთოთ ადრე მიღებული ORM-დიაგრამა (ნახ.4.33).



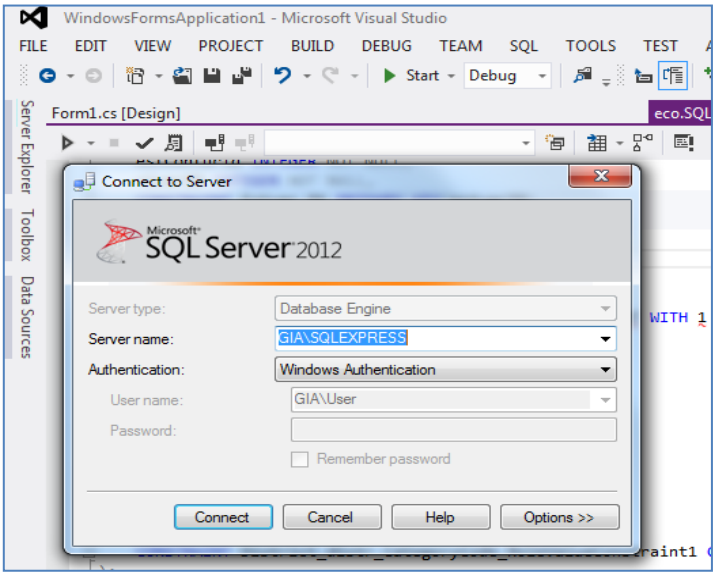
ნახ.4.33. ახალი Windows Form Application-ის შექმნა

Visual Studio.NET პლატფორმაზე NORMA ინსტრუმენტით ავტომატიზებულ რეჟიმში განვახორციელოთ მონაცემთა ბაზის .DDL კოდის გენერირება. მისი შესაბამისი ლისტინგი მოცემულია ქვემოთ:

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE, READ WRITE;

CREATE SCHEMA ORMModel1 DEFAULT CHARACTER SET UTF8;

SET SCHEMA 'ORMMODEL1';

CREATE TABLE ORMModel1.River
(
	riverID INTEGER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1) NOT
NULL,
	river_Laength DECIMAL NOT NULL,
	riverName CHARACTER VARYING NOT NULL,
	CONSTRAINT River_PK PRIMARY KEY(riverID)
);

CREATE TABLE ORMModel1.Estuar
(
	estuarID INTEGER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1)
NOT NULL,
	areaSqm INTEGER NOT NULL,
	est_CoordGPSxNr INTEGER NOT NULL,
```

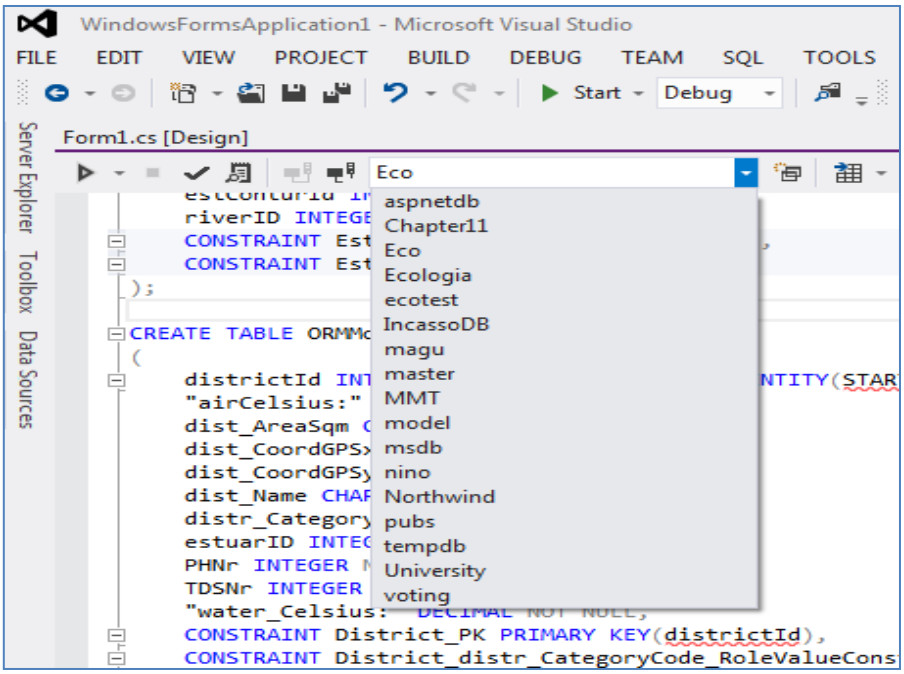
```
        est_CoordGPSyNr INTEGER NOT NULL,
        estConturId INTEGER NOT NULL,
        riverID INTEGER NOT NULL,
        CONSTRAINT Estuar_PK PRIMARY KEY(estuarID),
        CONSTRAINT Estuar_UC UNIQUE(riverID)
);

CREATE TABLE ORMModel1.District
(
  districtId INTEGER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1) NOT
NULL,
        "airCelsius:" INTEGER NOT NULL,
        dist_AreaSqm CHARACTER VARYING NOT NULL,
        dist_CoordGPSxNr INTEGER NOT NULL,
        dist_CoordGPSyNr INTEGER NOT NULL,
        dist_Name CHARACTER VARYING NOT NULL,
        distr_CategoryCode CHARACTER NOT NULL,
        estuarID INTEGER NOT NULL,
        PHNr INTEGER NOT NULL,
        TDSNr INTEGER NOT NULL,
        "water_Celsius:" DECIMAL NOT NULL,
        CONSTRAINT District_PK PRIMARY KEY(districtId),
        CONSTRAINT District_distr_CategoryCode_RoleValueConstraint1 CHECK
(distr_CategoryCode IN ('Normal, Sensitive, Vulnerable'))
);

CREATE TABLE ORMModel1.EstConturId
(
  estConturId INTEGER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1) NOT
NULL,
        contur CHARACTER VARYING(30) NOT NULL,
        dateExpDate CHARACTER NOT NULL,
        est_Coord_XNr INTEGER NOT NULL,
        est_Coord_YNr INTEGER NOT NULL,
        CONSTRAINT EstConturId_PK PRIMARY KEY(estConturId)
);
ALTER TABLE ORMModel1.Estuar ADD CONSTRAINT Estuar_FK1 FOREIGN KEY (estConturId)
REFERENCES ORMModel1.EstConturId (estConturId) ON DELETE RESTRICT ON UPDATE
RESTRICT;

ALTER TABLE ORMModel1.Estuar ADD CONSTRAINT Estuar_FK2 FOREIGN KEY (riverID)
REFERENCES ORMModel1.River (riverID) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE ORMModel1.District ADD CONSTRAINT District_FK FOREIGN KEY (estuarID)
REFERENCES ORMModel1.Estuar (estuarID) ON DELETE RESTRICT ON UPDATE RESTRICT;

COMMIT WORK;
```

შემდეგ ეტაპზე საჭიროა SQL Server-თან დაკავშირება (ვერსია 2012 ან უფრო მაღალი) (ნახ.4.34).



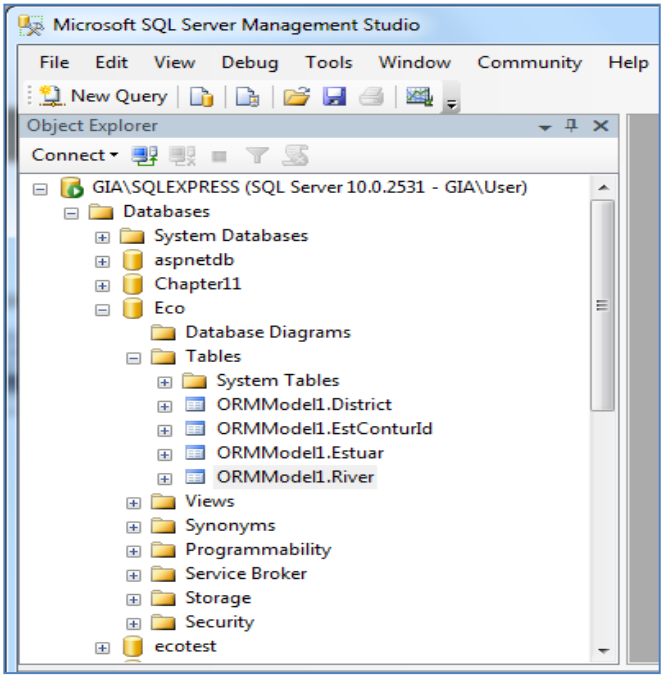
ნახ.4.34. SQL Server-თან დაკავშირების პროცესი

ჩამონათვალიდან ავირჩიოთ ჩვენი ბაზა, სახელად Eco, და ავირჩიოთ Execute ღილაკი.



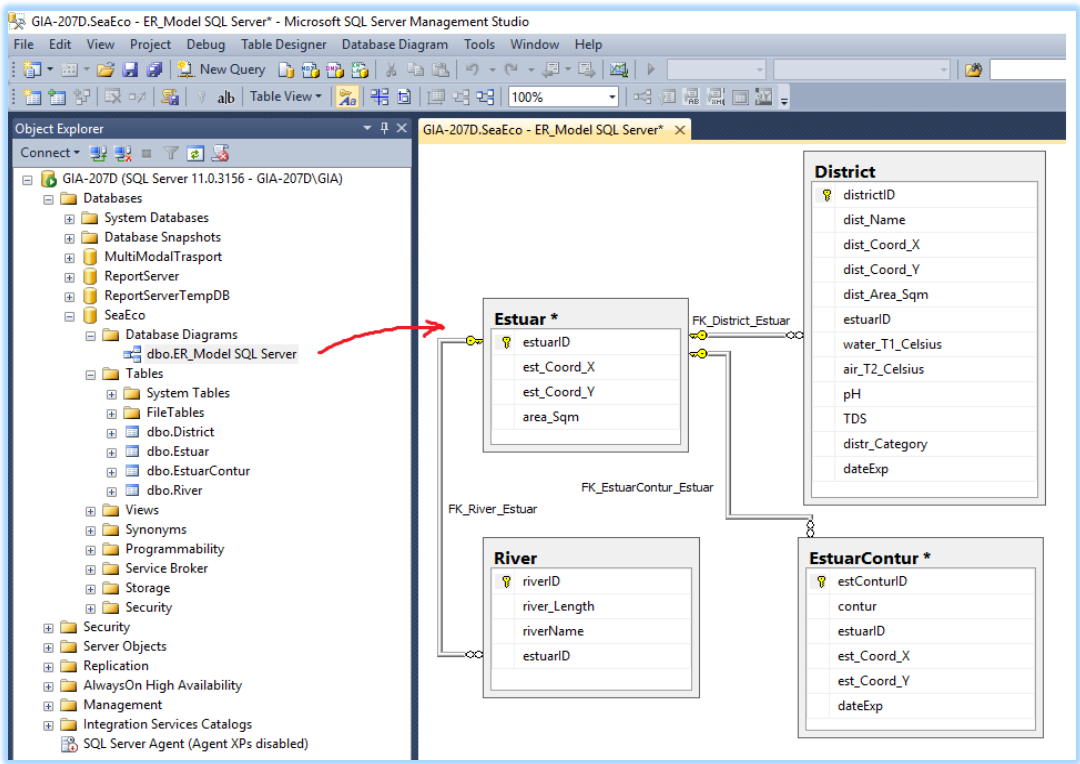
ნახ.4.35. Eco-ბაზასთან დაკავშირება

გავააქტიუროთ SQL Server, სადაც გამოჩნდება ავტომატიზებულ რეჟიმში მიღებული ცხრილები (ნახ.4.36).



ნახ.4.36. ORM-მოდელიდან მიღებული ცხრილები

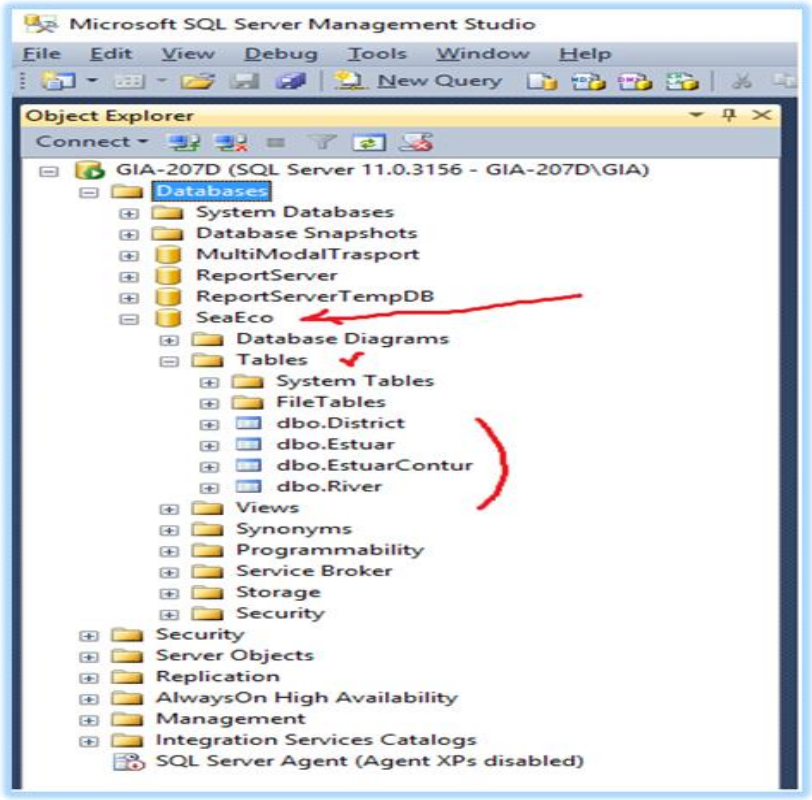
შავი ზღვის ეკოსისტემის სადემონსტრაციო მონაცემთა გაფართოებადი ბაზის საწყისი სტრუქტურა, ავსახოთ Visual Studio.NET გარემოში (ნახ.4.37).



ნახ.4.37. ER მოდელი მონაცემთა ბაზის სტრუქტურა, აგებული Visual Studio.NET გარემოში

4.5. შავი ზღვის ეკოლოგიური პარამეტრების მონაცემთა ბაზის აგება Ms SQL Server პაკეტის სამუშაო გარემოში

Ms SQL Server პაკეტის სამუშაო გარემოში მონაცემთა ბაზის აგების ორი ალტერნატიული გზაა. პირველი არის უშუალოდ Ms SQL Server Management Studio - გარემოში მუშაობა (ნახ.4.38). ამ შემთხვევაში საჭიროა აღნიშნული პროგრამული პაკეტის ცოდნა.



ნახ.4.38. MsSQL Server სამუშაო გარემო SeaEco ბაზით

მოცემულ ნახაზზე ნაჩვენებია ჩვენს მიერ MsSQL Server პაკეტით აგებული მონაცემთა რელაციური ბაზა „SeaEcfo".

ბაზის დემოვერსია შედგება ოთხი ცხრილისგან (Tables). ეს ცხრილებია:

- District.dbo (ნახ.4.39-ა,ბ),
- Estuar.dbo (ნახ.4.40-ა,ბ),
- EstuarContur.dbo (ნახ.4.41-ა,ბ) და
- River.dbo (ნახ.4.42-ა,ბ).

GIA-207D.SeaEco - dbo.District | GIA-207D.SeaEco - dbo.District

Column Name	Data Type	Allow Nulls
districtID	int	☐
dist_Name	nvarchar(50)	☑
dist_Coord_X	int	☑
dist_Coord_Y	int	☑
dist_Area_Sqm	int	☑
estuarID	int	☑
water_T1_Celsius	decimal(5, 2)	☑
air_T2_Celsius	decimal(5, 2)	☑
pH	int	☑
TDS	int	☑
distr_Category	nvarchar(50)	☑
dateExp	date	☑

ნახ.4.39-ა. უბნის (district) ცხრილის სტრუქტურა

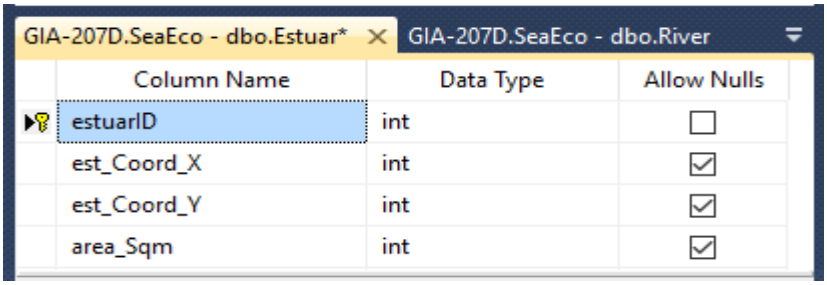
ნახაზზე ასახულია მდინარეების ესტუარების შესაბამისი უბნები GPS კოორდინატებისა და სხვა ატრიბუტებით.

GIA-207D.SeaEco - dbo.District | GIA-207D.SeaEco - ER_Model SQL Server*

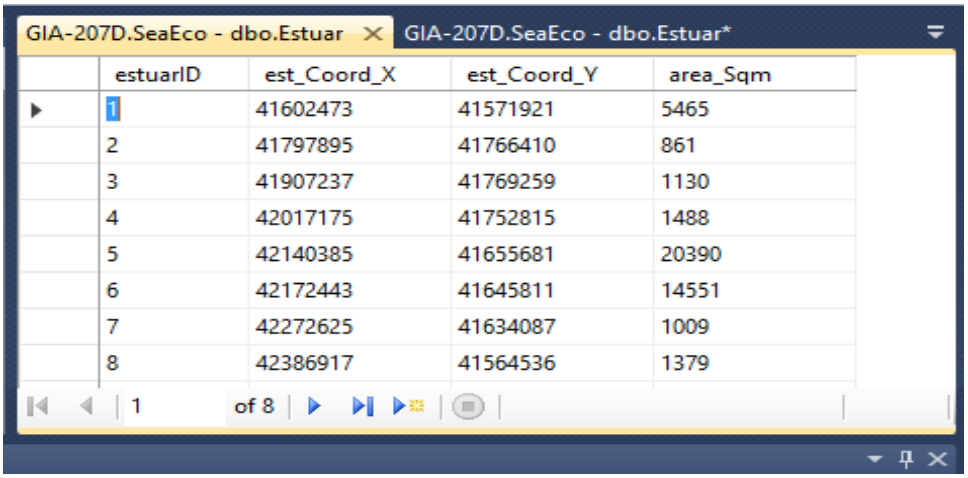
districtID	dist_Name	dist_Coord_X	dist_Coord_Y
1	სარფი	41526956	41548731
2	კვარიათი_1	41545542	41561587
3	კვარიათი_2	41554651	41563841
4	გონიო	41574588	41565589
5	ჭოროხი-მარცხენა	41596952	41569943
6	ჭოროხი-მარჯვენა	41607866	41577288
7	ადლია	41614371	41583944
8	ბათუმი (დელფინარიუმთან)	41649103	41621114
9	ბათუმი (დასაწყისი)	41650823	41666129
10	ბათუმი (ბენზე)	41662161	41678955
11	მახინჯაური (რკინიგზის სადგურთან)	41677322	41694925
12	. . .	41723714	41727073
32	ყულევი	42259918	41637102
33	ანაკლია (რეპერთან)	42382543	41577101
34	ანაკლია (სასტუმროსთან)	42382744	41563028
35	ანაკლია (მდ. ენგურის მარცხენა ნაპირი)	42389302	41560674

1 of 35

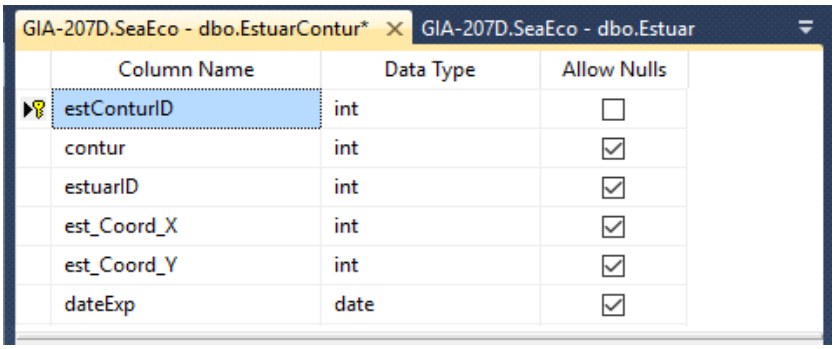
ნახ.4.39-ბ. უბნის (district) ცხრილი ჩანაწერებით (სულ 35 საკონტროლო წერტილია)



ნახ.4.40-ა. ესტუარების ცხრილის სტრუქტურა (Estuar)
SQL Server- მონაცემთა ბაზაში



ნახ.4.40-ბ. ესტუარების ცხრილი



ნახ.4.41-ა. ესტუარების კონტურების (EstuarContur) ცხრილის სტრუქტურა
SQL Server- მონაცემთა ბაზაში

GIA-207D.SeaEco - dbo.EstuarContur | GIA-207D.SeaEco - dbo.EstuarContur*

estConturID	contur	estuarID	est_Coord_X	est_Coord_Y	dateExp
1	1	1	41602473	41571921	2015-06-10
2	2	1	41600256	41570205	2015-06-10
3	3	1	41604317	41561104	2015-06-10
4	4	1	41618219	41539946	2015-06-10
5	5	1	41628109	41550019	2015-06-10
6	6	1	41620934	41573706	2015-06-10
7	7	1	41610627	41569531	2015-06-10
8	8	1	41604571	41573116	2015-06-10
9	1	2	41797895	41766410	2015-06-11
10	2	2	41799470	41763083	2015-06-11
11	3	2	41805091	41756342	2015-06-11
12	4	2	41813037	41760451	2015-06-11
13	5	2	41806051	41765812	2015-06-11
14	6	2	41802590	41768295	2015-06-11
15	1	3	41907237	41769259	2015-06-11
16	2	3	41910538	41766637	2015-06-11
17	3	3	41917843	41761913	2015-06-11
18	4	3	41917843	41754145	2015-06-11
19	5	3	41930396	41754482	2015-06-11
20	6	3	41927038	41763958	2015-06-11
21	7	3	41913224	41767610	2015-06-11

22 of 22

ნახ.4.41-ბ. ესტუარების კონტურების ცხრილი

GIA-207D.SeaEco - dbo.River | GIA-207D.SeaEco - dbo.District

Column Name	Data Type	Allow Nulls
riverID	int	☐
river_Length	decimal(7, 2)	☑
riverName	nvarchar(50)	☑
estuarID	int	☑

ნახ.4.42-ა. მდინარეების (River) ცხრილის სტრუქტურა SQL Server- მონაცემთა ბაზაში

GIA-207D.SeaEco - dbo.River | GIA-207D.SeaEco - dbo.River | GIA-207D.SeaEco - dbo.District

riverID	river_Length	riverName	estuarID
1	26,00	ჭოროხი (საქართველოს საზღვრებში)	1
2	25,20	კინტრიში	2
3	60,00	ნატანები	3
4	108,00	სუფსა	4
5	327,00	რიონი (სამხრეთ განშტოება)	5
6	327,00	რიონი (ჩრდილოეთ განშტოება)	6
7	150,00	ხობისწყალი	7
8	213,00	ენგური	8

1 of 8

ნახ.4.42-ბ. მდინარეების ცხრილი

V თავი
შავი ზღვის ეკომონიტორინგის სისტემის ვებ-პორტალის აგება და ვებ-სერვისების პროგრამული რეალიზაცია

განხილულია საქართველოს შავი ზღვის აკვატორიაში მდინარათა ესტუარების, ნავთობტერმინალებისა და პორტების, აგრეთვე სხვა საკონტროლო წერტილების ეკოლოგიური მდგომარეობის მონიტორინგის კომპიუტერული სისტემის ვებ-დეველოპმენტის ამოცანის გადაწყვეტა მაიკროსოფთის Visual Studio.NET პლატფორმაზე SharePoint ტექნოლოგიით. აგებულია სისტემის ვებ-პორტალი, ვებ-საიტები და ვებ სერვისები, მათი პროგრამული რეალიზებისა და მომხმარებლის ინსტრუქციებით.

5.1. Ms SharePoint ტექნოლოგიის გამოყენება ვებ-პორტალის ასაგებად

Microsoft ფირმის ტექნოლოგია SharePoint არის კორპორაციული ქსელების ინფორმაციულ მოთხოვნებზე მორგებული პროგრამული უზრუნველყოფა, რომელიც მომხმარებლებლებს თანამშრომლობის და ჯგუფური სერვისების გამოყენების მოქნილ შესაძლებლობებს სთავაზობს [18]. კერძოდ, პროგრამის მეშვეობით შესაძლებელი ხდება ვებ-ბაზირებული სერვისების სწრაფი შექმნა ჯგუფური სამუშაოობისთვის, რაც კორპორაციულ ქსელებში მიმდინარე პროცესების მზარდ ავტომატიზაციას უწყობს ხელს.

MsSharePoint-ის დადებითი მხარეა, ისიც რომ ადვილია მისი გამოყენება. ამისათვის კლიენტის მხარეს საკმარისია ნებისმიერი ინტერნეტ-ბრაუზერი (Internet Explorer, Mozilla Firefox, Google Chrome, Opera), ხოლო სერვერზე სერვისთა სირთულის მიხედვით შეიძლება გამოყენებულ იქნას სხვადასხვა პროგრამები.

MsSharePoint საშუალებით მომხმარებლებს ეძლევათ საშუალება შექმნან სხვადასხვა შინაარსისა და დანიშნულების ვებ-საიტები. საერთო საიტები, რომელიც ცნობილია, როგორც გუნდური საიტები ან ჯგუფურად სამუშაო საიტები, საშუალებას აძლევს ორგანიზაციის თანამშრომლებს უფრო კომფორტულად ისამუშაოონ ერთმანეთთან. მათ შეუძლიათ საიტი ერთმანეთისათვის გამოიყენონ საჭირო დოკუმენტაციის გასაზიარებლად, საქმეების დასაგემად/გადასანაწილებლად, ღონისძიებების გასაზიარებლად ვებ-კალენდრის საშუალებით და ა.შ. SharePoint-ის ამ დანიშნულებით გამოყენებას ეწოდება გუნდური თანამშრომლობის სისტემა.

ბევრი კომპანია SharePoint Server-ს იყენებს, როგორც დოკუმენტაციის შენახვის და მათი გაცვლის ცენტრალურ საშუალებას. ამას ეწოდება დოკუმენტაციის ელექტრონული მართვის სისტემა.

დაბოლოს, ზოგიერთი კომპანია იყენებს SharePoint-ს როგორც თავისი ინტერნეტ-საიტების პლატფორმას. ნებისმიერ მსურველს ამ საიტების საშუალებით შეუძლია გაეცნოს კომპანიის საქმიანობას, მის სერვისებს, ღონისძიებებს. SharePoint საიტებს აქვთ ბევრი ისეთი ჩაშენებული ფუნქცია, როგორიცაა მაგალითად, ინფორმაციის ჩამონათვალი, დაცვის

მენეჯმენტი, დოკუმენტაციის ბიბლიოთეკა (დოკუმენტების საცავი), ძიების სისტემა, რისი წაყლობითაც ისინი არიან უფრო მეტად მოსახერხებელი და გამოყენებადი. SharePoint-ის ამ სახის გამოყენება ცნობილია, როგორც ვებ-კონტენტის მართვის სისტემა. SharePoint-ის გამოყენების მრავალგვაროვნება მიუთითებს მის მოქნილობაზე.

საიტის საშუალებით შესაძლებელია:

- განყოფილების დოკუმენტებისა და მონაცემთა ბაზების შენახვა;
- პროექტზე ერთდროული მუშაობის დაგეგმვა;
- შეხვედრის მასალები და შეტყობინებების მომზადება;
- სპეციალური ვებ-მონაცემთა ბაზების და ცოდნის ბაზების მომზადება;
- ბიზნეს-პროცესების ავტომატიზაცია;
- ბლოგების შექმნა.

თითოეული შექმნილი საიტისათვის შესაძლებელია ადმინისტრატორის განსაზღვრა - რაც საშუალებას იძლევა ადვილად გაანაწილოთ პასუხისმგებლობა ინფორმაციის სხვადასხვა ბლოკების მართვაზე. საიტის შექმნა შეუძლია ადმინისტრატორის უფლების მქონე მომხმარებელს.

SharePoint-ის ძირითადი კომპონენტები ასე შეიძლება წარმოვიდგინოთ: შინარსი (Content) და ძიება (Search), გარე (მაგალითად, საოფისე) აპლიკაციების ინტეგრირება ერთიან გარემოში (Insights) და წინასწარ დასამუშავებული ფუნქციური ბლოკების გამოყენება ბიზნეს-გადაწყვეტილებათა შესაქმნელად (Composities) (ნახ.5.1).

ნახ.5.1. SharePoint-ის ძირითადი კომპონენტები

თითოეული ამ სერვისისთვის დეტალურად განისაზღვრება წვდომათა მართვის სიები, რის საფუძველზეც კორპორაციული ქსელი ერთიან ინფორმაციულ პორტალს წარმოადგენს, სადაც კომფორტულად მუშაობის საშუალებები გააჩნიათ ორგანიზაციული იერარქიის სხვადასხვა დონეზე მდგომ მომხმარებლებს.

შეზღუდვების დაწესებისა და უფლებათა გაცემის პროცესს სისტემაში მართავას სუპერუფლებების მქონე ერთი ან რამდენიმე ადმინისტრატორი. ყველა დანარჩენი მომხმარებელი სისტემაში მეტნაკლებად შეზღუდულია.

ღრუბლოვანი სერვისის ასაგებად მომხმარებელს შეუძლია ისარგებლის სხვადასხვა ობიექტებით; ესენია: ბიბლიოთეკები, სიები, ვებ-გვერდი, ვებ-საიტი. მომხმარებელი კი ირჩევს ობიექტს იმის მიხედვით, თუ რომელი ფუნქციონალი სჭირდება: იქნება ეს მონაცემთა ბაზა, შიგთავსის მართვა, თუ სხვა.

ბიბლიოთეკები (Libraries) - ყველა ტიპის დოკუმენტის საცავი (ბიბლიოთეკები დოკუმენტების, გრაფიკული ინფორმაციის, ანგარიშების, სლაიდებისა და სხვა ტიპის ინფორმაციის შესანახად და გამოსატანად);

სიები (Lists) - კონტეინერული ტიპის ელემენტი ინფორმაციის სხვადასხვა ტიპის წყაროებიდან ინფორმაციის მოპოვებისა და კონსოლიდირებული ასახვისთვის; სიების ნიმუშებად შეიძლება დავასახელოთ კალენდარი, კონტაქტების ბაზა, დავალებები, სადისკუსიო პლატფორმა და სხვა.

ვებ-გვერდი (Page) – ახალი გვერდი არსებული ვებ-საიტის ფარგლებში;

ვებ-საიტი (Site) - ახალი ვებ-საიტი რომელიმე დავალების შესასრულებლად. ვებ-საიტების კატეგორიებია საძიებო საიტი, გუნდური სამუშაოობის საიტი, ბლოგი, საკონტაქტო ინფორმაციის მართვის საიტი და სხვა მრავალი.

შესაძლებელია ელექტრონული დოკუმენტბრუნვის ორგანიზება. დოკუმენტის რედაქტირებისა და შენახვის შემდეგ სხვა მომხმარებლებს მაშინვე შეუძლიათ ნახონ მოდიფიცირებული დოკუმენტი და ის ცვლილებები, რომლებიც იქნა განხორციელებული. იმ დროს, როდესაც ერთი მომხმარებელი ახორციელებს რაიმე ფაილის რედაქტირებას, შეიძლება სხვაც მუშაოობდეს ამავე დოკუმენტზე. ამ პრობლემების თავიდან აცილების მიზნით საჭიროა გამოვიყენოთ SharePoint-ის check-in და check-out შესაძლებლობები.

თუ დოკუმენტში გვინდა ცვლილებების შეტანა, ავირჩიოთ ბრძანება Check-out-ს საჭირო დოკუმენტისთვის, რადგან ჩვენთან ერთად სხვა მომხმარებელმაც არ განახორციელოს რაიმე ცვლილება ამ დოკუმენტზე და არ გამოვიწვიოთ კონფლიქტი. შემდეგ ვირჩევთ იმ პროგრამას, რომელშიც უნდა გავხსნათ და შევცალოთ დოკუმენტი; ვცვლით დოკუმენტის შიგთავსს, ვირჩევთ ბრძანებას Check-in და ვიმახსოვრებთ. არსებობს საშუალება ერთდროულად რამდენიმე დოკუმენტზე ჩავრთოთ check-in და check-out ბრძანებები.

როდესაც ერთი მომხმარებელი მუშაობს დოკუმენტზე სხვა ვერ ხედავს ცვლილებებს მანამ, სანამ ეს უკანასკნელი არ ჩართავს ბრძანებას check-in-ს. თუ ისინი აირჩევენ

დოკუმენტის დათავლიერებას ნახავენ დოკუმენტის იმ ვერსიას, რომელიც იყო მანამდე, სანამ დაიწყებოდა რედაქტირება და ჩართული იქნებოდა check-out ბრძანება. ასევე მათ არ შეუძლიათ განახორციელონ ცვლილებები აღნიშნულ დოკუმენტზე. სხვები ვერ ნახავენ რედაქტირებულ დოკუმენტს იმ შემთხვევაშიც კი, თუ მას ცვლილებების მერე შევინახავთ. იმისათვის, რომ შეცვლილი დოკუმენტის ნახვა შეძლოს ყველამ, საჭიროა check-in ბრძანების ჩართვა.

მომხმარებლებს, რომლებსაც აქვთ უფლებების სრული პაკეტი, შეუძლიათ ყველა იმ დოკუმენტზე ჩართონ check-in, რომლებზეც სხვა მომხმარებლებს აქვთ დადებული check-out. ეს ძალიან მნიშვნელოვანია, რომ „სუპერ უფლებების“ მქონე მომხმარებელს შეეძლოს სხვა მომხმარებლების მიერ დადებული ბრძანების შეცვლა, რადგან თუ რომელიმე მომხმარებელმა დატოვა კომპანია, საჭიროა ვინმე სხვას ჰქონდეს წვდომა მის მიერ განხორციელებულ სამუშაოზე.

დოკუმენტების ან სიის ელემენტების ეკრანზე გამოტანის რომელიმე ხერხის არჩევა შესაძლებელია წარმოდგენების (view) საშუალებით.

წარმოდგენების შექმნის ფუნქცია აქვს დოკუმენტების ყველა ბიბლიოთეკას. ისინი განსაზღვრავენ, თუ რომელი ბიბლიოთეკა რა ფორმატში იყოს წარმოდგენილი მომხმარებლებისათვის. მომხმარებელს, რომელიც სარგებლობს მართვის უფლებებით, შეუძლია შექმნას დოკუმენტაციის წარმოდგენის ახალი რეჟიმი.

ასევე შესაძლებელია განვსაზღვროთ მიმართვის პარამეტრები:

- დოკუმენტების დათვალიერების ეს რეჟიმი ყველასთვის წვდომადი იყოს, თუ პერსონალური;
- რომელი თანამიმდევრობით იყოს დალაგებული დოკუმენტაცია;
- როგორი იყოს სვეტების სორტირება;
- რა რეჟიმში იყოს წარმოდგენილი დოკუმეტების განლაგება;
- გამოჩნდეს თუ არა ფოლდერებში მოთვსებული დოკუმენტები;
- რამდენი დოკუმენტი გამოჩნდეს და ა.შ.

SharePoint-ში არსებობს დოკუმენტაციის შენახვის, მართვის და მათთან წვდომის სხვადასხვა გზები, რაც ზრდის SharePoint-ის პლატფორმის შესაძლებლობებს ბიზნეს-პროცესებისათვის.

სამუშაო პროცესები (Workflow) საშუალებას აძლევს მომხმარებლებს ერთობლივად იმუშაონ დოკუმენტებზე, მართონ პროექტის დავალებები, დანერგონ ბიზნეს-პროცესები დოკუმენტებისათვის და Microsoft Office SharePoint Server-ის კვანძის ელემენტებისათვის. სამუშაო პროცესი - ეს არის განსაზღვრული გზა, რომელსაც ბიზნეს-პროცესი გაივლის შესრულებამდე, ანუ ყველა ის შესაბამისი მოქმედება და დავალება რომლებიც საჭიროა ბიზნეს-პროცესის გამართვისათვის.

სამუშაო პროცესები საშუალებას აძლევს ორგანიზაციებს შეათანხმონ თავიანთი ბიზნეს-პროცესები, ამავე დროს ეფექტურად მართონ ისინი.

სამუშაო პროცესს განიხილავენ, როგორც დავალებების გარკვეულ თანამიმდევრობას, რომელსაც მოაქვს შედეგი; კერძოდ, დოკუმენტების ან ელემენტების ავტომატიზებული მოძრაობა, რომლებიც დაკავშირებულია ბიზნეს-პროცესებთან. სამუშაო პროცესების გამოყენება შესაძლებელია საწარმოს ზოგადი ბიზნეს-პროცესების მუდმივად სამართავად ისე, რომ ჩართული იყოს ბიზნეს-ლოგიკა. არსებითად, ბიზნეს-ლოგიკა წარმოადგენს ინსტრუქციების ნაკრებს, რომელიც მართავს და აკონტროლებს დოკუმენტების /ელემენტების მოძრაობას.

სამუშაო პროცესები საშუალებას იძლევა დავზოგოთ დანახარჯები და დრო, რომელიც საჭიროა ზოგადი ბიზნეს-პროცესების კოორდინაციისათვის, ისეთის როგორიცაა პროექტის დამტკიცება ან დოკუმენტის შემოწმება. SharePoint-ში სამუშაო პროცესების შექმნა შესაძლებელია საიტისთვის, სიებისთვის ან ბიბლიოთეკებისთვის. არსებობს სხვადასხვა სახის ჩაშენებული (Built in) სამუშაო პროცესები, რომლებიც გამოიყენება ზოგად ბიზნეს-სცენარებში [18];

Approval – დამტკიცებითი სამუშაო პროცესი გამოიყენება იმ შემთხვევაში, თუ საჭიროა დოკუმენტის დამოწმება. დამმოწმებლებს აქვთ საშუალება დაეთანხმონ (Approve), უარყონ (Reject), განმეორებით დაეთანხმონ (Reassign) ან მოითხოვონ დოკუმენტის შეცვლა სამუშაო (Reqvest changes) პროცესის განმავლობაში;

Three-State – სამეტაპიანი სამუშაო პროცესი აფიქსირებს დოკუმენტის ან სიის ელემენტის სამ მდგომარეობას: აქტიური (Active), მზადაა გადასახედად (Ready for Review), დასრულდა (Complete);

Collect feedback – უკუკავშირების სამუშაო პროცესი გამოიყენება იმ შემთხვევაში თუ საჭიროა დოკუმენტის ან სიის ელემენტის გადამოწმება.

ეს სამუშაო პროცესი სასარგებლოა მაშინაც, თუ გვჭირდება დავაფიქსიროთ, რომ ბიზნესის დაინტერესებულმა მხარეებმა გადახედეს დოკუმენტაციას და დააფიქსირეს შენიშვნები.

უკუკავშრის პროცესის პარამეტრების ცხრილი დამტკიცებითი სამუშაო პროცესის პარამეტრების ცხრილის ანალოგიურია. უკუკავშირების კოლექციის სამუშაო პროცესი მოითხოვს პროცესის საიტის მახასიათებლების აქტივიზაციას.

Collect signature – ხელმომწერების სამუშაო/ტექნიკური პროცესი. მომხმარებლები ღებულობენ ხელმოწერის მოთხოვნას და ციფრული ხელმოწერით ადასტურებენ დოკუმენტს;

Disposition approval — ლიკვიდაციის დამოწმება. ეს სამუშაო პროცესი აკონტროლებს დოკუმენტის მოქმედების ვადას, მისი შენახვის პერიოდს და სამუშაო პროცესის მონაწილეს საშუალებას აძლევს თვითონ გადაწყვიტოს ვადაგასული დოკუმენტების ბედი - შეინახოს თუ წაშალოს.

სამუშაო პროცესი შეიძლება ჩავრთოთ დოკუმენტების ბიბლიოთეკაში, საიდანაც განსაზღვრული დოკუმენტი მიეწოდება ადამიანთა ჯგუფს დასამტკიცებლად. როდესაც დოკუმენტის ავტორი გაუშვებს შესრულებაზე მოცემულ სამუშაო პროცესს, სამუშაო პროცესი ქმნის დოკუმენტის დამტკიცების დავალებას, ნიშნავს ამ სამუშაო პროცესის შემსრულებლებს, შემდეგ მათ ელექტრონული ფოსტით უგზავნის შეტყობინებას

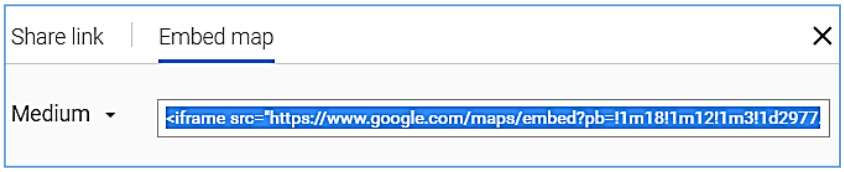
შეტყობინებაში მითითებულია გარკვეული ინსტრუქციები ამ დავალების შესასრულებლად და ლინკი იმ დოკუმენტზე, რომელიც უნდა დამტკიცდეს. სამუშაო პროცესის მიმდინარეობისას, მის მფლობელს ან მონაწილეებს შეუძლიათ ნახონ, როგორ მიმდინარეობს სამუშაო პროცესი, უკვე ვინ შეასრულა იგი და ა.შ.

მას შემდეგ, რაც ყველა მონაწილე შეასრულებს მათ დავალებებს, სამუშაო პროცესი დასრულდება, ავტომატურად წყვეტს მუშაობას, ხოლო მისი შემქმნელი ასევე ავტომატურად ღებულობს შეტყობინებას ამის შესახებ.

SharePoint-ს ასევე აქვს Excel Sevices-ების მხარდჭერა. ეს სერვისი უზრუნველყოფს Excel-ის დიაგრამების გამოტანას კორპორაციული საიტის ვებ-გვერდზე. Excel Services საშუალებას იძლევა დავათვალიეროთ Excel-ის მონაცემები უშუალოდ ბრაუზერში. იგი წარმოადგენს SharePoint ტექნოლოგიას, რომელიც აფართოებს Excel-ს სერვერული ტექნოლოგიებით. მომხმარებელს შეუძლია მიმართოს და აწარმოოს ნებისმიერი სახის გამოთვლები Excel-ის სამუშაო წიგნში უშუალოდ ბრაუზერში, მაშინ როდესაც სერვერი უზრუნველყოფს მონაცემთა უსაფრთხოებასა და მართვას.

- ბუნებრივია, ამ სერვისს თავდაპირველად სჭირდება შესაბამისი სერვისების კონფიგურირება.
- შესაძლებელია Google Map-ის ჩასმა SharePoint Server-ის გვერდზე მარტივად, შემდეგი ბრძანებების შესრულებით:

- გავააქტიუროთ Google Maps აპლიკაცია;
- Text Box-ში მივუთითოთ ის მისამართი, რომლის ჩვენებაც გვინდა;
  - ავირჩიოთ რუკის მარცხენა ზედა ნაწილში მდებარე Menu ღილაკი;
  - ავირჩიოთ Share or embedded map ბრძანება, გაიხსნება Google Maps ფანჯარა;
  - ავირჩიოთ სასურველი ზომა რუკისთვის, რომელიც უნდა ჩანდეს Web გვერდზე;
  - დავიმახსოვროთ (Copy) რუკის HTML კოდი.



ნახ. 5.2. რუკის HTML კოდი.

5.2. Web-პორტალის დაპროექტება ეკომონიტორინგის სისტემისათვის

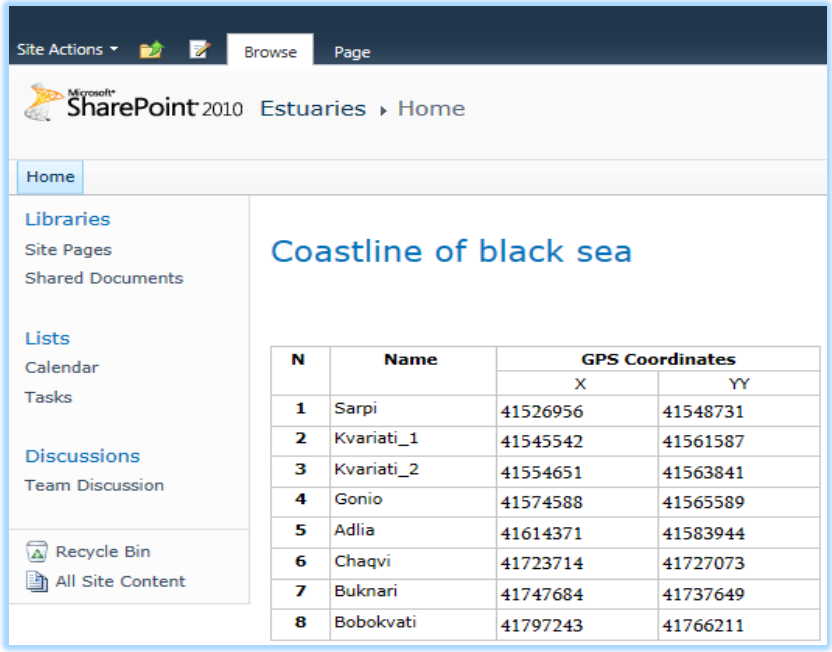
შავი ზღვის ეკოლოგიური მონიტორინგის მიზნით საველე კვლევების ჩატარება, კერძოდ სხვადასხვა მაჩვენებლების დაფიქსირება GPS-კოორდინატების მიხედვით მეტად მოხერხებულია, როდესაც მონაცემების შეტანა შესაძლებელია ტერიტორიულად დაშორებული კომპიუტერიდან ან უფრო უკეთესი - მობილური ტელეფონიდან.

მონაცემების ოპერატიული შეტანა, უშუალოდ კვლევის წერტილიდან გამორიცხავს ე.წ. „ჟურნალის“ წარმოების აუცილებლობას. მონაცემები შეტანისთანავე აისახება კორპორატიული პორტალის ვებ-გვერდზე და ტერიტორიულად დაშორებულ SQL Server-ის ბაზაში.

პირველ ეტაპზე აუცილებელია *ვებ-პორტალის* დაპროექტება, სადაც შექმნილია ვებ-გვერდები თანამშრომლებისათვის, ესტუარების საკონტროლო კოორდინატების ასახვისა და აზომვებისათვის [12].

5.3 ნახაზზე ნაჩვენებია დასაპროექტებელი პორტალის საწყისი გვერდი.

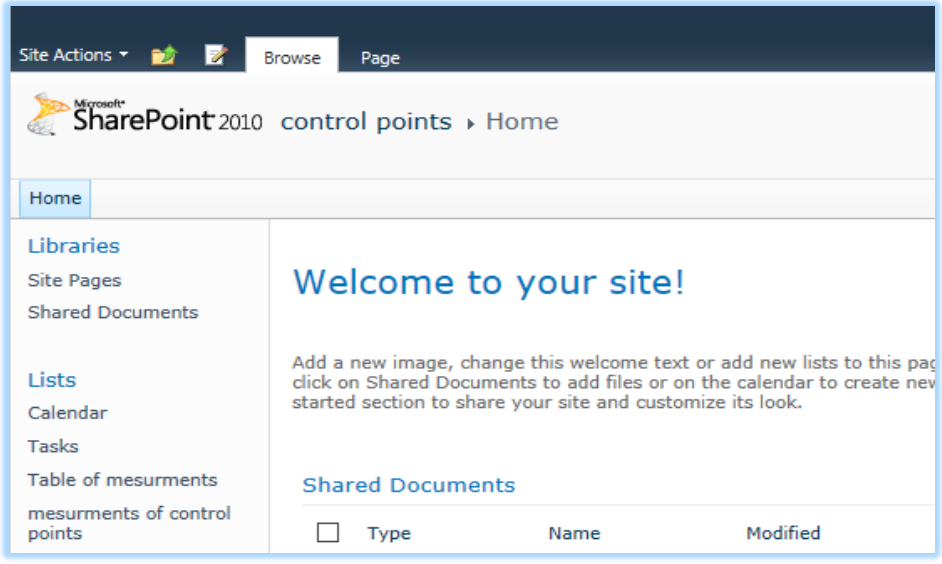
ნახ.5.3 ვებ-პორტალის საწყისი გვერდი



N	Name	GPS Coordinates	
		X	YY
1	Sarpi	41526956	41548731
2	Kvariati_1	41545542	41561587
3	Kvariati_2	41554651	41563841
4	Gonio	41574588	41565589
5	Adlia	41614371	41583944
6	Chaqvi	41723714	41727073
7	Buknari	41747684	41737649
8	Bobokvati	41797243	41766211

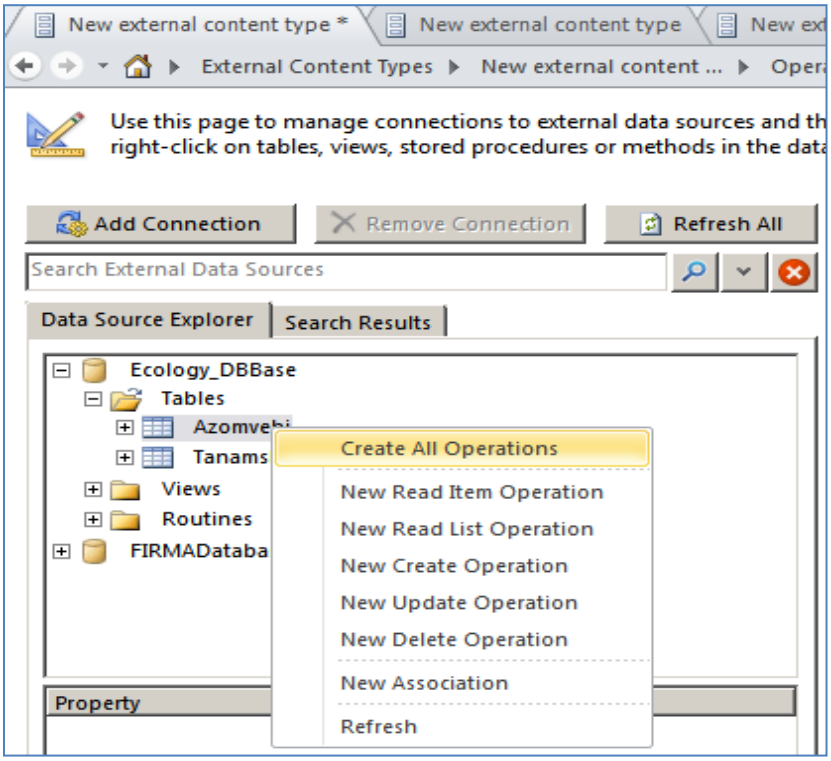
ნახ.5.4 „ვებ-გვერდი Estuaries”

ესტუარებისთვის განკუთვნილ ვებ-გვერდზე ასახულია ესტუარების სახელწოდებები და მათი GPS კოორდინატები (ნახ.5.4), 5.5 ნახაზზე კი - საკონტროლო წერტილების ვებ-გვერდი.



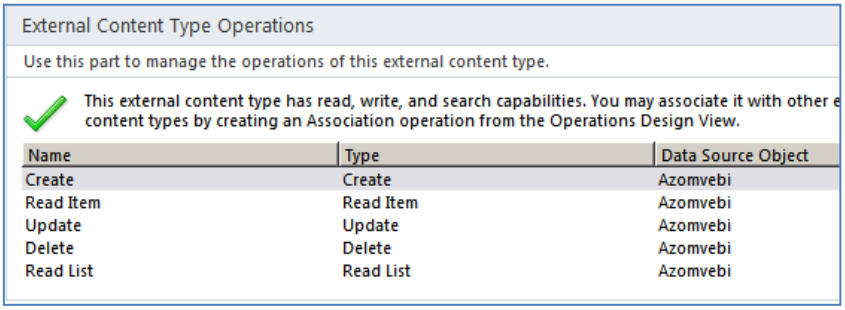
ნახ.5.5. ვებ-გვერდი „Control Points”

შემდეგ ეტაპზე გვჭირდება SQL Server-ის მონაცემთა ბაზის დაკავშირება ვებ-პორტალთან. ამ ამოცანის გადასაჭრელად გამოყენებულია Sharepoint Designer-ი. 5.6 ნახაზზე ნაჩვენებია მონაცმთა ბაზასთან დაკავშირება ვებ-პორტალის External List-თან.



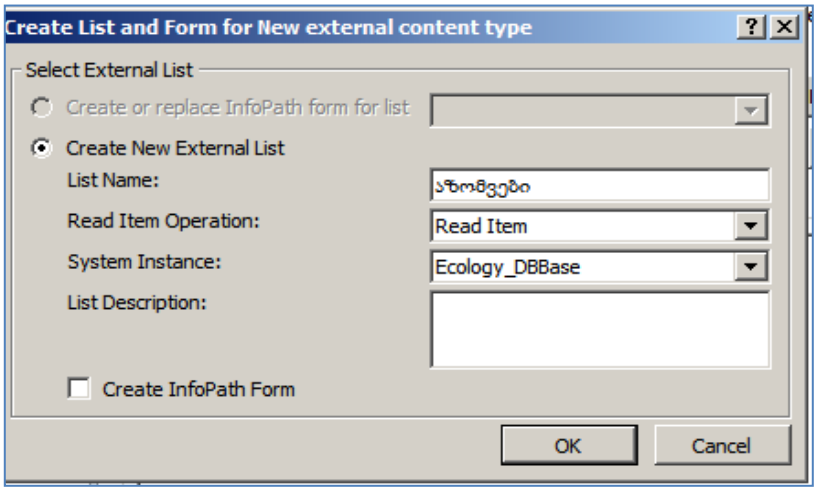
ნახ.5.6. SQL Server-ის მონაცემთა ბაზის დაკავშირება ვებ-პორტალთან

როგორც სურათიადან ჩანს, მოცემული გარე სიის საშუალებით შესაძლებელია მონაცემების შექმნა, წაკითხვა, განახლება, წაშლა (ნახ.5.7).



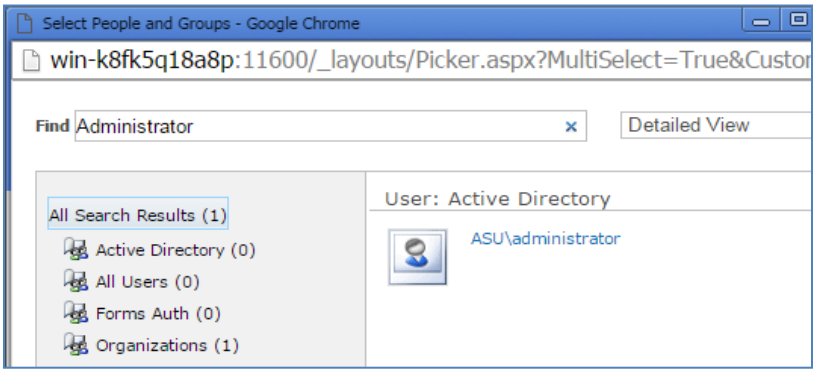
ნახ.5.7. გარე სიიდან მისაწვდომი ოპერაციები

5.8 ნახაზზე მოცემულია გარე სიის შექმნის პროცესის ფანჯარა, რომელშიც ვებ-პორტალი უკვე დაკავშირებულია მონაცემთა ბაზასთან.



ნახ.5.8. გარე სიის შექმნის პროცესი

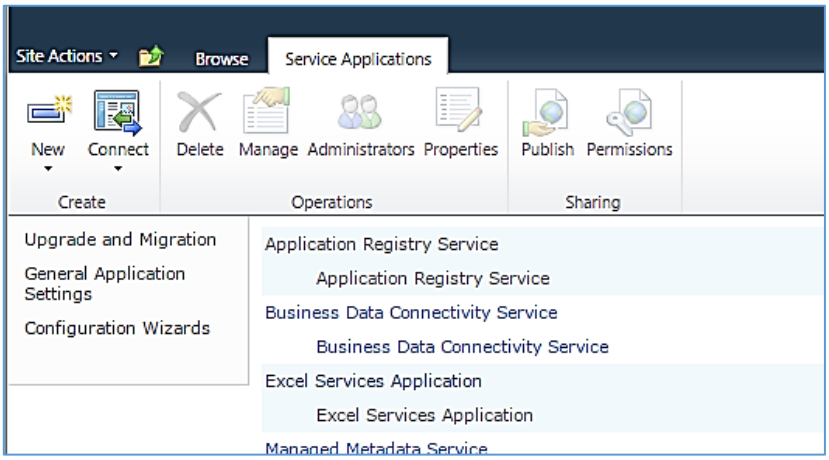
მომდევნო ეტაპზე აუცილებელია Business Connectivity Services (BCS) კონფიგურაციის პარამეტრების მითითება. კერძოდ, სისტემის ადმინისტრატორმა უნდა მიანიჭოს უფლებები თითოეულ მომხმარებელს მონაცემთა ბაზასთან წვდომის მიზნით (ნახ.5.9).



ნახ.5.9. მომხმარებლისთვის უფლებების მინიჭება

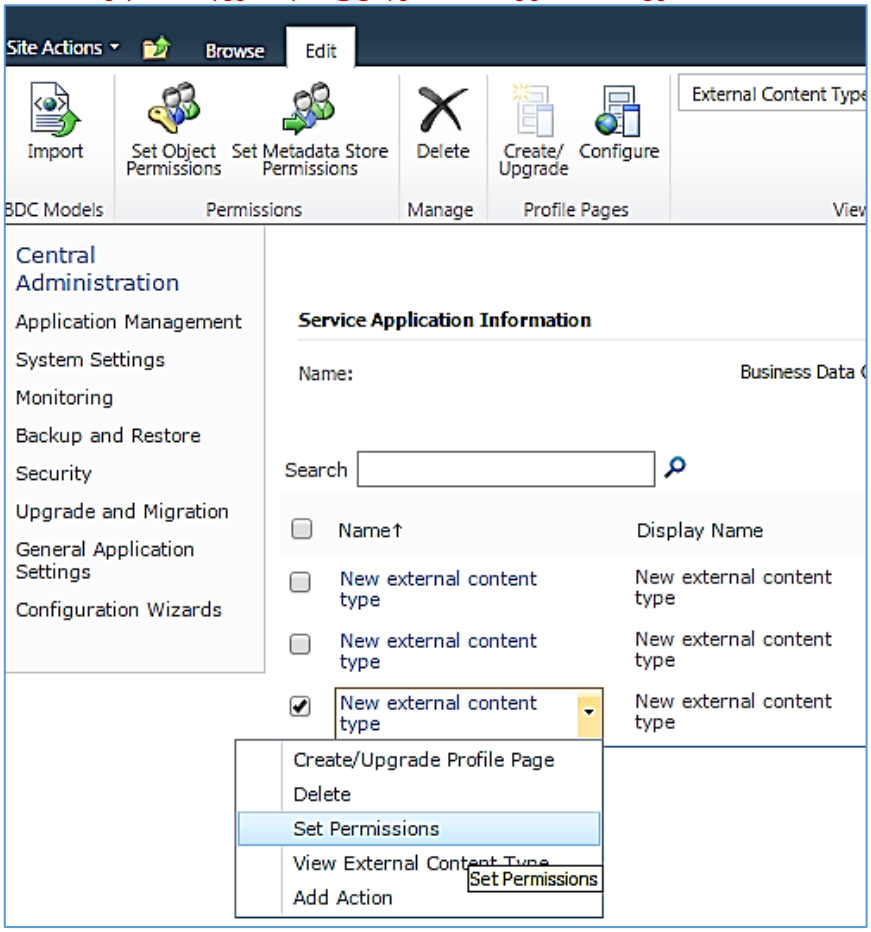
Business Connectivity Services არის SharePoint-ისა და Office-ს ცენტრალიზებული ინფრასტრუქტურა, რომელიც მონაცემებთან მუშაობის ინტეგრირებულ გადაწყვეტილებებს უზრუნველყოფს. ეს სერვისი საშუალებას გვაძლევს გამოვიყენოთ SharePoint და Office კლიენტები ისეთ მონაცემებთან სამუშაოდ, რომლებიც განთავსებულია SharePoint-ის გარეთ. ჩვენს შემთხვევაში ეს მონაცემები განთავსებულია SQL Server-ზე და მათი გამოტანა SharePoint-ის საიტზე მოხდება Business Connectivity Services-ის საშუალებით.

5.10. ნახაზზე მოცემულია Business Connectivity Services-ის-ის გააქტიურება.



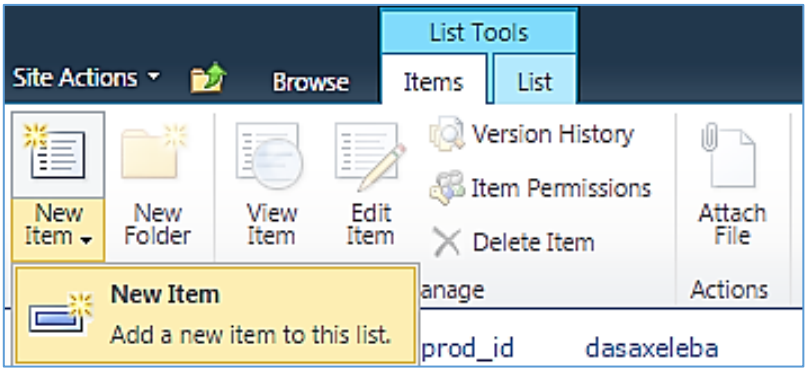
ნახ.5.101. Secure Store Service-ის გააქტიურება

5.11 ნახაზზე წარმოდგენილია უფლების მინიჭების პროცესი.



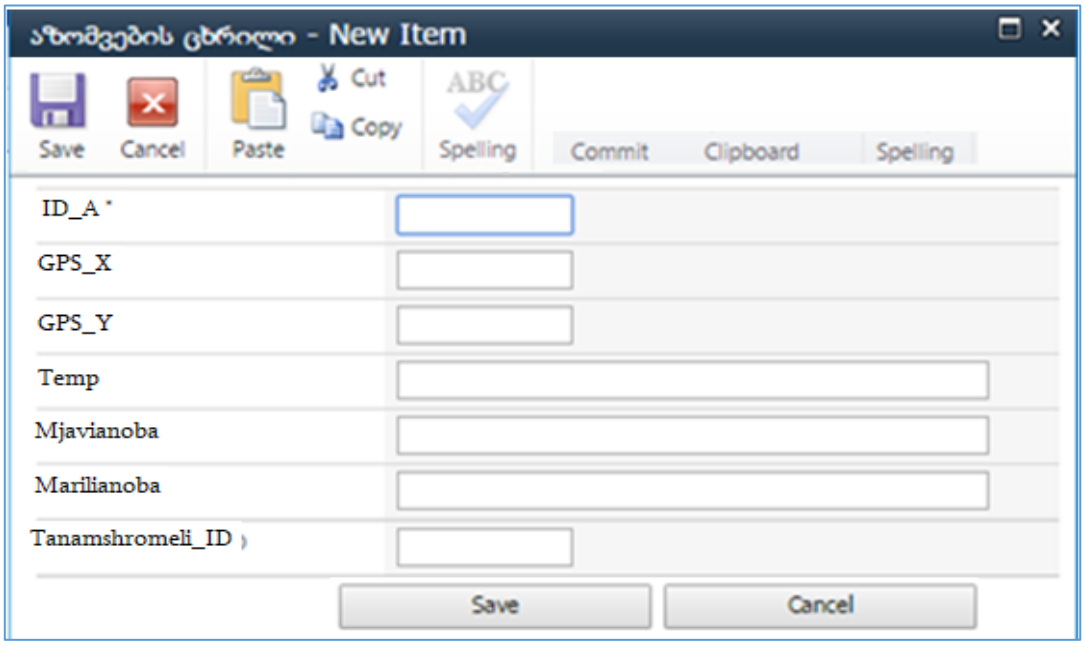
ნახ.5.11. გარე სიისთვის უფლებების მინიჭება

ახალი ჩანაწერის ჩასამატებლად საჭიროა ავირჩიოთ Add Item ბრძანება (ნახ. 5.12).



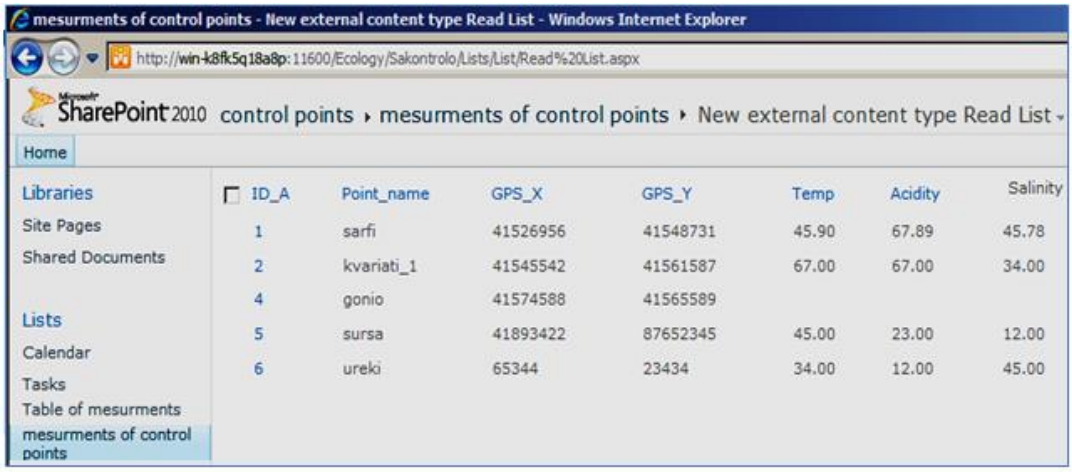
ნახ.5.122. ახალი ჩანაწერის ჩამატება

მონაცემების შეტანის პროცესის დიალოგური ფანჯარა ნაჩვენებია 5.13 ნახაზზე.



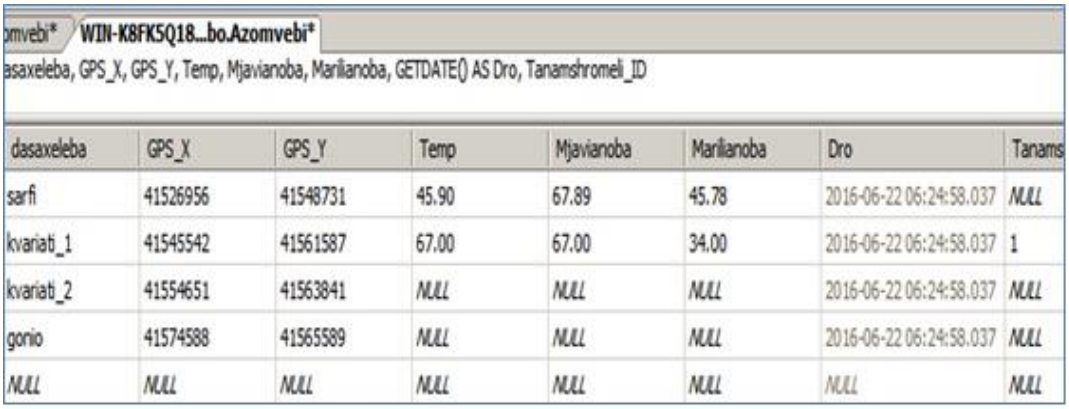
ნახ.5.13. დიალოგური ფანჯარა

5.14 ნახაზზე ასახულია საკონტროლო წერტილების აზომვების ცხრილი ორგანიზაციის ვებ-პორტალზე.



ნახ.5.14. დიალოგური ფანჯარის სქემა

5.15 ნახაზზე ჩანს მაჩვენებლები საკონტროლო წერტილებიდან, სადაც მონაცემების შეტანის ***დრო*** ფიქსირდება ავტომატურად.



ნახ.5.15. მონაცემთა დაფიქსირება SQL Setver-ის ცხრილში

ამრიგად, ყოველივე ზემოთ აღნიშნული საშუალებას მოგვცემს კომპლექსურად შევაფასოთ შავი ზღვის თანამედროვე ეკოლოგიური პრობლემები და დაიგეგმოს მისი სანაპირო ზოლისა და მიმდებარე ტერიტორიების ეკოლოგიური უსაფრთხოების ღონისძიებები SQL Server-ის მონაცემთა ბაზის გამოყენებით.

ბუნებრივია, იმისათვის რომ ვიმუშაოთ მობილური მოწყობილობიდან აუცილებელია კონფიგურაციის პარამეტრების მითითება უშუალოდ მობილური მოწყობილობიდან. Sharepoint Server თავსებადია მობილური მოწყობილობების შემდეგ ოპერაციულ სისტემებთან: Windows Phone, Windows 7, iOS, Android [19].

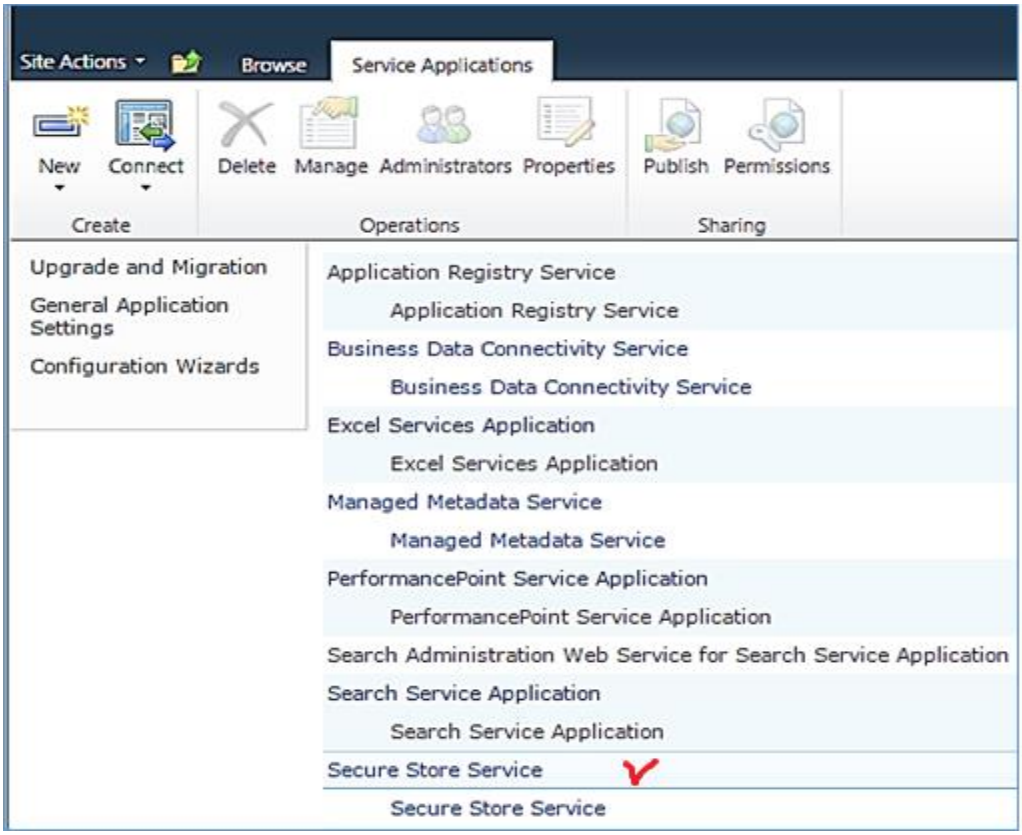
5.3. კონფიდენციალური მონაცემების შენახვის სერვისი (Secure Store Service)

მონაცემთა უსაფრთხოდ შენახვის სამსახური - Secure Store Service რეალურად წარმოადგენს ნამდვილობის შემოწმების პროცედურას, რომლის განხორციელებაც ხდება სერვერზე.

მონაცემთა უსაფრთხოდ შენახვის სამსახური მოიცავს მონაცემთა ბაზას, სადაც ინახება მომხმარებელთა სააღრიცხვო ჩანაწერები (ასევე მათი პაროლები) იმ აპლიკაციების იდენტიფიკატორებისათვის, რომლებსაც იყენებს ეს აპლიკაციები საერთო რესურსებთან ავტორიზებული მიმართვის დროს. მაგალითად, SharePoint Server-ის უსაფრთხოდ შენახვის მონაცემთა ბაზა შეიძლება გამოყენებულ იქნეს სააღრიცხვო ჩანაწერების შესანახად ან ამოსაღებად გარე მონაცემებთან მიმართვის დროს.

Secure Store Service გააქტიურება ხდება შემდეგი თანამიმდევრობით:

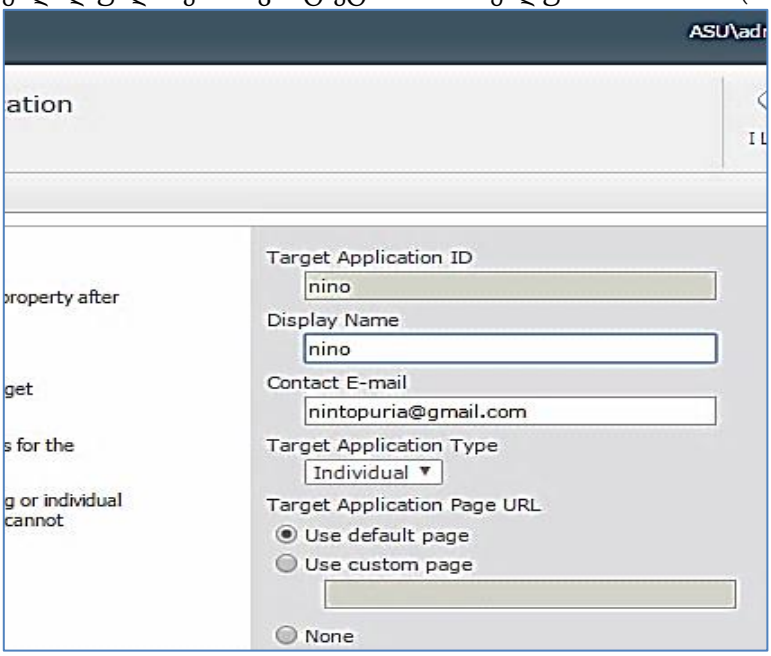
ვააქტიურებთ Sharepoint central administration სამსახურს და ვირჩევთ Secure Store Service (ნახ.5.16).



ნახ.5.16. Secure Store Services არჩევა

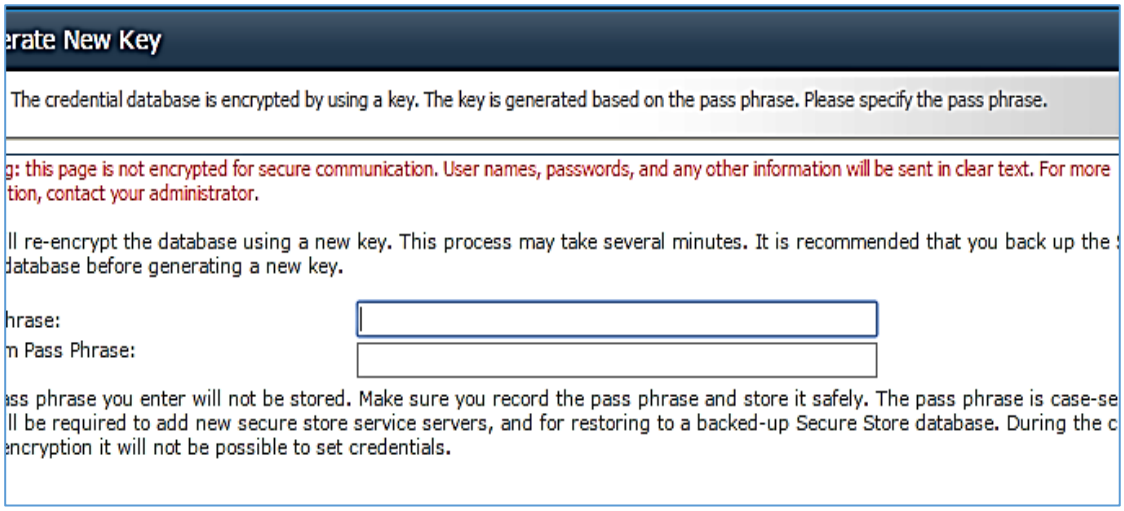
Secure Store Services ინახავს კონფიდენციალურ მონაცემებს, ამიტომ საჭიროა მისი დაშიფვრა. თავდაპირველად აუცილებელია იმ გასაღების გენერირება, რომლითაც მოხდება შინაარსის დაშიფვრა. ასევე აუცილებელია დაშიფვრის გასაღების არქივაცია.

შევქმნათ ახალი Secure Store Target Application. შევავსოთ ფორმა. მითითებული ი-მეილი აუცილებლად უნდა იყოს საკონტაქტო პირის რეალური მისამართი (ნახ.5.17).



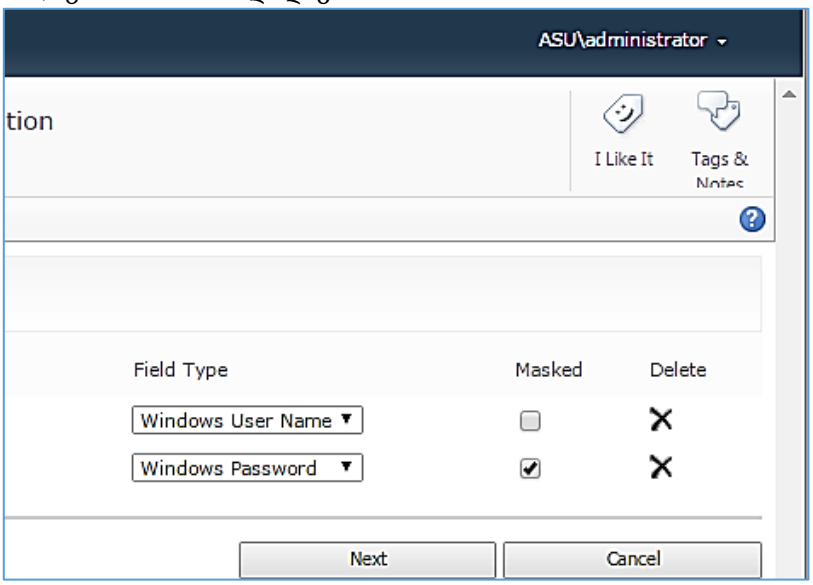
ნახ.5.17. Secure Store სერვისის არჩევა

5.18 ნახაზზე ნაჩვენებია ფრაზა-პაროლის მითითების პროცესი.



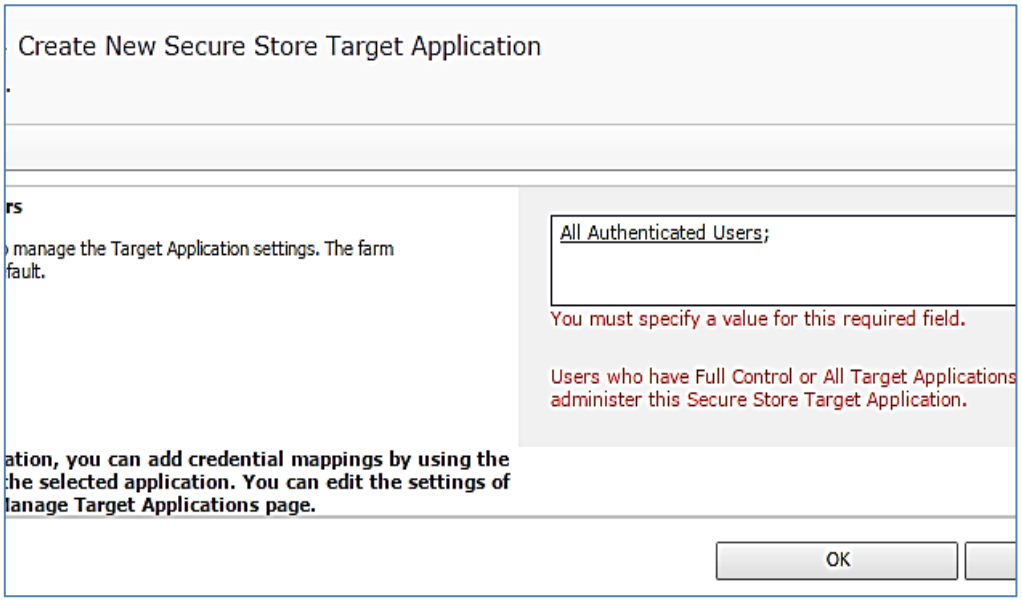
ნახ.5.18-ა. ფრაზა-პაროლის მითითება

5.18-ბ ნახაზზე მოცემულია Secure Store Services ფანჯარა, სადაც ცვლილებების შეტანა არ არის საჭირო; ავირჩიოთ Next ღილაკი.



ნახ.5.18-ბ. ფრაზა-პაროლის მითითება

5.19 ნახაზზე მოცემულია საბოლოო აპლიკაციის ადმინისტრატორის და აპლიკაციის პარამეტრების მითითების პროცესი.

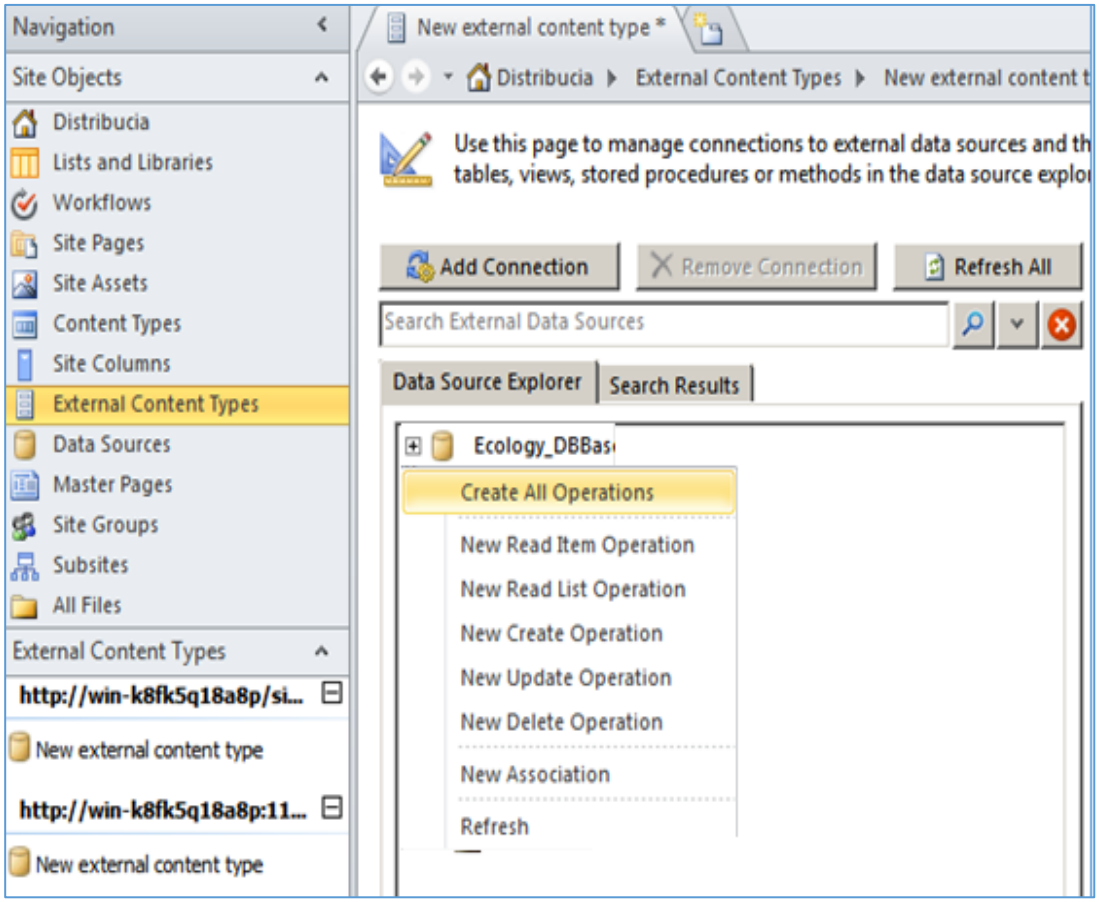


ნახ.5.19. არჩეული მომხმარებელი

Secure Store აპლიკაციის საბოლოო კოდი წარმოადგენს უნიკალურ იდენტიფიკატორს. საბოლოო აპლიკაციის შემდეგ, მისი შეცვლა შეუძლებელია.

ამგვარად, შეიქმნა Secure Store Application ID, რომელიც აუცილებელია გარე მონაცემებთან კავშირის დროს. შემდგომში, გარე მონაცემებთან მიმართვისას საჭირო იქნება პაროლის შეტანა, რათა მოხდეს მომხმარებლის ავტორიზაცია.

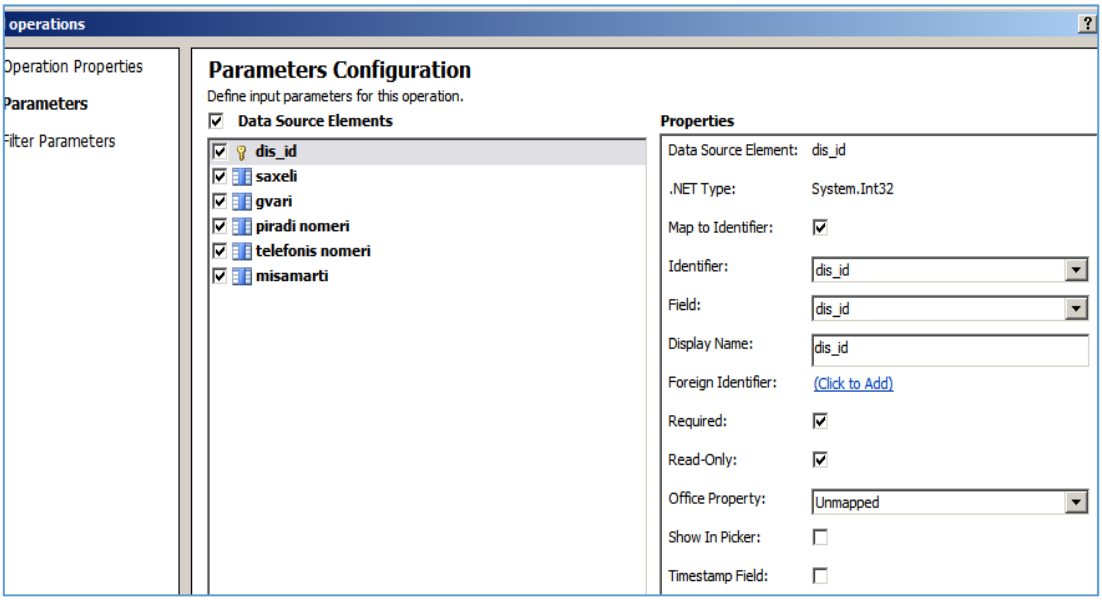
კავშირი დამყარებულია, Data Source Explorer-ის ფანჯარაში გამოჩნდება ჩვენს მიერ არჩეული მონაცემთა ბაზის სახელი. ავირჩიოთ ცხრილი, რომელთანაც ვაპირებთ მუშაობას და მისი კონტექსტური მენიუდან ავიჩიოთ Create All Operations ბრძანება (ნახ. 5.20)



ნახ.5.20.. ოპერაციების არჩევა

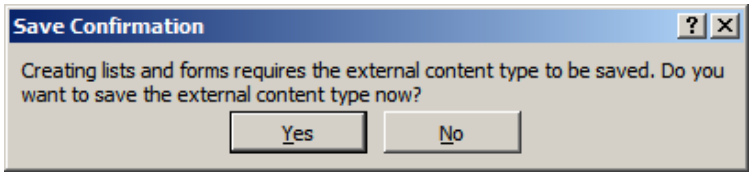
ვირჩევთ Next ღილაკს (ნახ. 5.21). მოვნიშნოთ ის ველები, რომლებიც გვინდა რომ ჩანდეს ვებ-გვერდზე და ავირჩიოთ Finish ღილაკი (ნახ. 5.22).

ნახ.5.21. ოპერაციების ფანჯარა



ნახ.5.22. არჩეული ველების ფანჯარა

დავიმახსოვროთ მონაცემები (ნახ.5.23).



ნახ.5.23. ცვლილებების დამახსოვრების ფანჯარა

5.4. შავი ზღვის ეკომონიტორინგის კომპიუტერული სისტემის ექსპერიმენტული კვლევა

შავი ზღვის აკვატორიაში საქართველოს საზღვრებში საველე-ექსპედიციური კვლევები განხორციელდა ავტორების მიერ (დოქტორანტი ა. გავარდაშვილი, სამეცნიერო ხელმძღვანელი გ. სურგულაძე) შოთა რუსთაველის სამეცნიერო ფონდის საგრანტო პროექტის (#DO/159/4-130/14) „შავი ზღვის ეკოლოგიური პარამეტრების კვლევა მულტიმედიური ბაზების საფუძველზე“ ფინანსური მხარდაჭერით [1]. საველე კვლევები განხორციელდა 2015-2017 წლებში. წინამდებარე პარაგრაფში გადმოცემულია როგორც აღნიშნული საველე კვლევების ანალიზური მასალები, ასევე მათი დამუშავების შემდეგ მიღებული შედეგები.

5.4.1. შავი ზღვის აკვატორიაში განხორციელებული საველე-ექსპედიციური კვლევები და მათი შეფასება

საველე-ექსპედიციური კვლევების მიზანი იყო ძირითადი მდინარეების ესტუარე-ბისა და მის მიმდებარე მოწყვლადი უბნების დაფიქსირება GPS კოორდინატებში; ზღვის სანაპირო ზოლის სენსიტიურ უბნებზე თურქეთის საზღვრიდან სარფიდან წყლის სინჯების აღება მათი ძირითადი ფაქტორების ადგილზე გაზომვით: წყლისა (t1) და ჰაერის (t2) ტემპერატურის დაფიქსირება; წყლის მჟავიანობისა (pH) და მარილიანობის (TDS) რაოდენობრივი მაჩვენებლების განსაზღვრა (ნახ.5.24).

ნახ.5.24. GPS-კოორდინატების დაგენა და წყლის სინჯების პარამეტრების დადგენა შავ ზღვაზე (ა. გავარდაშვილი, ოქტომბერი, 2015)

შავი ზღვის 110 კმ სიგრძის სანაპირო ზოლში სარფიდან სოფელ განმუხურამდე შერჩეულ იყო 35 უბანი, სადაც დაფიქსირდა ზემოაღნიშნული სიდიდეები (ნახ.5.25), ხოლო საველე კვლევის ჯამური მონაცემები მოყვანილია 5.1 ცხრილში.

შავი ზღვის ეკოლოგიური პრობლემების შეფასების მიზნით განხილულია საველე კვლევების სტატისტიკური რიგისა და კომპიუტერული თანამედროვე პროგრამების ერთობლივი გამოყენების მცდელობა.

ნახ..5.25. საქართველოს საზღვრებში შავი ზღვის სანაპირო ზოლში საველე კვლევების ჩატარების სქემა (2015-2017)

საქართველოს საზღვაო პორტებში, ნავთობსადენის ტერმინალებსა და მათ მიმდებარე ტერიტორიებზე შავი ზღვის ეკოლოგიური პარამეტრების დადგენის მიზნით 2016 წლის აპრილ-მაისის და ივნისის თვეში განხორციელდა საველე-სარეკოგნოსცირებო სამეცნიერო კვლევები. ისინი ითვალისწინებდა ჩვენ მიერ წინასწარ შერჩეულ ადგილებში შავი ზღვის წყლის ანალიზის აღებას და მის ლაბორატორიულ გამოკვლევას, როგორც სტაციონალურ ასევე არასტაციონალურ პირობებში [20].

საველე კვლევის შედეგები (ოქტომბერი, 2015) ცხრ.5.1

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატურა (t1/t2)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
		X	Y			
1	სარფი	41526956	41548731	0,93	8,49	13,45
2	კვარიათი_1	41545542	41561587	0,93	8,38	12,54
3	კვარიათი_2	41554651	41563841	0,92	8,42	13,20
4	გონიო	41574588	41565589	0,92	8,37	13,23
5	ჭოროხი-მარცხენა	41596952	41569943	0,92	8,38	12,80
6	ჭოროხი-მარჯვენა	41607866	41577288	0,96	8,42	12,88
7	ადლია	41614371	41583944	0,91	8,23	6,60
8	ბათუმი(დელფინარი-უმთან)	41649103	41621114	0,86	8,34	11,80
9	ბათუმი (დასაწყისი)	41650823	41666129	0,86	8,36	8,66
10	ბათუმი (ბენზე)	41662161	41678955	0,86	8,11	7.07
11	მახინჯაური (რკინიგზის სადგურთან)	41677322	41694925	0,88	8,28	13,20
12	ჩაქვი	41723714	41727073	0,75	8,27	12,86
13	ბუკნარი	41747684	41737649	0,86	8,38	6,23
14	ციხისძირი	41772644	41755505	0,85	8,32	6,15
15	ბობოყვათი	41797243	41766211	0,88	8,35	11,3
16	ქობულეთი (მდ. კინტრიშთან)	41811607	41771416	1,00	8,29	5,23
17	ქობულეთი – აღმაშენებლის ქ. 270	41833693	41775383	0,86	8,43	12,98
18	ქობულეთი – აღმაშენებლის ქ. 522	41856175	41777474	0,86	8,56	14,65
19	ქობულეთის ბოლო	41880721	41772763	0,86	8,45	13,87
20	მდ. ჩოლოქი	41895964	41770675	0,90	8,21	15,27
21	მდ. ნატანები	41913572	41767241	0,85	8,45	4,34
22	სოფ. შეკვეთილი	41938746	41764857	0,91	8,43	12,90
23	Dდაბა ურეკი	41997287	41758008	0,91	8,38	11,76
24	მდ. სუფსა	42016078	41753594	0,90	8,36	12,67
25	სოფ. გრიგოლეთი (დასაწყისი)	42038751	41735281	0,90	8,43	13,50
26	სოფ. გრიგოლეთი (ბოლო)	42056578	41723947	0,83	8,25	13,60
27	მდ. მალთაყვა	42092887	41,695954	0,87	8,34	7,22
28	ქ. ფოთი (მდ. რიონის სამხრეთი ჩადინება)	42134187	41659283	0,87	8,39	15,32
29	ქ. ფოთი (მაშველები)	42140873	41657926	0,83	8,41	6,37
30	ქ. ფოთი(მოლი)	42147686	41,655392	0,78	8,48	13,60
31	Nაბადა (მდ. რიონის ჩრდილოეთი ჩადინება)	42177666	41,641295	0,74	8,72	7,80
32	ყულევი	42259918	41,637102	0,78	8,37	15,12
33	ანაკლია (რეპერთან)	42382543	41,577101	0,78	8,32	14,45
34	ანაკლია (სასტუმროსთან)	4238744	41563028	0,78	8,29	7,67
35	ანაკლია (მდ. ენგურის მარცხენა ნაპირი)	42389302	41560674	0,78	8,32	12,67

საკვლევი ტერიტორიის სენსიტიური უბნები, ანუ წერტილების მდებარეობები განსაზღვრულ იქნა GPS კოორდინატებით, რომლებიც დატანილ იქნა ციფრულ რუკაზე (ნახ.5.26), ხოლო სენსიტურ უბნებზე აღებული შავი ზღვის წყლის სინჯების ლაბორატორიული კვლევის შედეგები კი მოყვანილია 5.2-:-5.4 ცხრილებში.

ნახ.5.26. შავი ზღვის აკავტორიაში სენსიტიური წერტილების მდებარეობა

საქართველოს საზღვაო პორტებსა და ნავთობსადენის ტერმინალებში 27.04.16 მონაცემები ცხრ.5.2

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატურა (t1/t2)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
		X	YY			
1	მდინარე ჭოროხი	41600395	41548731	0,91	8,33	12,8
2	ბათუმის პორტი	41662161	41678955	0,86	8,08	6,97
3	მდინარე სუფსა	42016078	41753594	0.90	8,30	11,2
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინალი	42038751	41735281	0,83	8,38	12,3
5	მდინარე რიონი	42132201	41660636	0,87	8,37	13,1
6	ფოთის პორტი	42147724	41655297	0,83	8,22	6,25
7	ყულევის ნავთობის ტერმინალი	42276524	41631693	0,78	8,38	12,3
8	მდინარე ხობისწყალი	42259918	41637102	0,87	8,33	13,5
9	ანაკლიის მშენებარე პორტი	42382543	41577101	0,78	8,29	10,4
10	მდინარე ენგური	42389302	41560674	0,78	8,32	9,29

საქართველოს საზღვაო პორტებსა და ნავთობსადენის

ტერმინალებში 27.05.16 მონაცემები ცხრ.5.3

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატურა (t1/t2)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
		X	YY			
1	მდინარე ჭოროხი	41600395	41548731	0,79	8,53	12,3
2	ბათუმის პორტი	41662161	41678955	0,76	8,42	11,80
3	მდინარე სუფსა	42016078	41753594	0,78	8,60	6,22
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინალი	42038751	41735281	0.94	8,53	10,8
5	მდინარე რიონი	42132201	41660636	0,58	8,56	12,1
6	ფოთის პორტი	42147724	41655297	0,58	8,01	6,34
7	ყულევის ნავთობის ტერმინალი	42276524	41631693	0,89	8,53	5,84
8	მდინარე ხობისწყალი	42259918	41637102	0,78	6,56	7,09
9	ანაკლიის მშენებარე პორტი	42382543	41577101	0,92	8,03	10,82
10	მდინარე ენგური	42389302	41560674	0,89	8,53	5,84

საქართველოს საზღვაო პორტებსა და ნავთობსადენის

ტერმინალებში 6.06.16 მონაცემები ცხრ.5.4

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატურა (t1/t2)	წყლის მჟავიანობა pH	მარილიანობა (TDS)
		X	**YY**			
1	მდინარე ჭოროხი	41600395	41548731	0,92	8,22	6,60
2	ბათუმის პორტი	41662161	41678955	0,86	8,42	11,80
3	მდინარე სუფსა	42016078	41753594	0,85	8,13	6,15
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინალი	42038751	41735281	0.90	8,36	11,67
5	მდინარე რიონი	42132201	41660636	0,78	7,84	6,37
6	ფოთის პორტი	42147724	41655297	0,78	8,65	13,34
7	ყულევის ნავთობის ტერმინალი	42276524	41631693	0,78	8,37	15,12
8	მდინარე ხობისწყალი	42259918	41637102	0,82	7,71	6,76
9	ანაკლიის მშენებარე პორტი	42382543	41577101	0,76	8,52	13,45
10	მდინარე ენგური	42389302	41560674	0,77	8,32	12,67

5.27 ნახაზზე ნაჩვენებია მდინარე ენგურის კალაპოტიდან წყლის სინჯების აღებისა და გაზომვის პროცესი.

ნახ.5.27. მდ. ენგური, ანაკლიის მიმდებარე ტერიტორია.
სინჯების აღება (ა. გავარდფაშვილი, 05.2016)

ამრიგად, შავი ზღვის საქართველოს საზღვრებში ჩატარებული საველე-სამეცნიერო კვლევებისა და წყლის სინჯებზე ლაბორატორიული კვლევის შედეგად დადგინდა, რომ 2016 წლის გაზაფხულ-ზაფხულის პერიოდში:

- წყლის და ჰაერის ფარდობითი ტემპერატურის მნიშვნელობა იცვლებოდა (t_1/t_2) = 0,76 - 0,92 საზღვრებში;
- შავი ზღვის წყლის მარილიანობის მნიშვნელობა (TDS) = 5,84 – 15,12 საზღვრებში;
- შავი ზღვის წყლის მჟავიანობის მნიშვნელობა (pH) = 7,71 – 8,65 საზღვრებში.

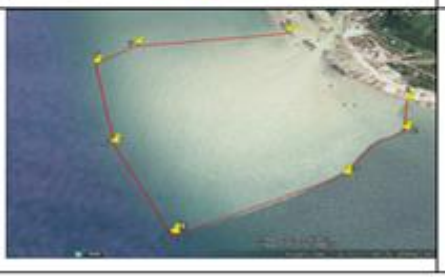
შავი ზღვის ძირითადი მდინარეების: ჭოროხის, კინტრიშის, ნატანების, სუფსის, ხობის წყალის, რიონის და ენგურის - ესტუარებში საველე-სამეცნიერო კვლევა თანამედროვე ეკოლოგიური მდგომარეობის შესასფასებლად განხორციელდა 2015 წლის ივნის-ივლისის თვეში.

საველე-სამეცნიერო კვლევის მიზანი იყო შავი ზღვის აუზის საქართველოს ზემოთ დასახელებული ძირითადი მდინარეების ესტუარების ფართოებების დაზუსტება GPS-ს კოორდინატებში და მათი დატანა ციფრულ რუკებზე, ზღვის წყლისა (t_1) და ჰაერის (t_2) ტემპერატურების დაფიქსირება, ასევე ზღვის მოწყვლად უბნებსა და ესტუარებში ზღვის წყლის ანალიზის აღება და მათი ქიმიური ლაბორატორიული გამოკვლევა.

5.28 ნახაზზე მოცემულია ზემოაღნიშნულ მდინარეთა ესტუარების კონტურები GPS კოორდინატებში შესაბამისი ფართობებით, ხოლო 5.5 ცხრილში - კი მდინარეთა ძირითადი ჰიდროლოგიური მაჩვენებლები.

მდინარეები და წერტილის კოორდინატები	ციფრულ რუკაზე ესტუარის საერთო ხედი	ესტუარის ფართობი (კმ²)
I. ჭოროხი (საქართველოს საზღვრებში) 1. X- 41602473, Y- 41571921 2. X - 41600256, Y- 41570205 3. X- 41604317, Y- 41561104 4. X - 41618219, Y- 41539946 5. X- 41628109, Y- 41550019 6. X- 41620934, Y- 41573706 7. X - 41610627, Y- 41569531 8. X - 41604571, Y- 41573116		5.465
II. კინტრიში 1. X- 41797895; Y- 41766410 2. X- 41799470; Y- 41763083 3. X- 41805091; Y- 41756342 4. X- 41813037; Y- 41760451 5. X- 41806051; Y- 41765812 6. X- 41802590; Y- 41768295		0.861
III. ნატანები 1. X- 41907237, Y- 41769259 2. X- 41.909746, Y- 41766637 3. X- 41910538, Y- 41761913 4. X- 41917843, Y- 41754145 5. X- 41930396, Y- 41754482 6. X- 41927038, Y- 41763958 7. X- 41913224, Y- 41767610		1.130
IV. სუფსა 1. X- 42017175, Y- 41752815 2. X- 42011160, Y- 41748014 3. X- 42009462;Y- 41744850 4. X- 42018012, Y- 41738489 5. X- 42025248, Y- 41730800 6. X- 42019232, Y- 41731400 7. X- 42023220, Y- 41741095 8. X- 42022795, Y- 41748376		1.488
V. რიონი (სამხრეთ განშტოება) 1.X- 42140385; Y- 41655681 2.X- 42122645; Y- 41650961 3.X- 42124069; Y- 41635599 4.X- 42.146301; Y-41592190 5.X- 42181975; Y- 41595853 6.X- 42147957; Y- 41642238 7.X- 42149272; Y- 41646819 8.X- 42147875; Y- 41654381		20.390
VI. რიონი (ჩრდილოეთ განშტოება) 1. X- 42172443; Y- 41645811 2. X- 42173884; Y- 41629814 3. X- 42194850; Y- 41593968 4. X- 42234461; Y- 41609054 5. X- 42261137; Y- 41626465 6. X- 42244261; Y- 41632567 7. X- 42216245; Y- 41631737		14,551

ნახ.5.28. შავი ზღვის აჭარის საქართველოს ძირითადი მდინარების ესტუარის მონაცემები

VII. ხობისწყალი 1. X - 42272625; Y- 41634087 2. X - 42269928; Y- 41630695 3. X - 42269375; Y- 41627515 4. X - 42270267; Y- 41624979 5. X - 42277002; Y- 41623763 6. X - 42281359; Y- 41627407 7. X - 42276524; Y- 41631693		1.009
VIII. მდ. ენგური 1. X- 42386917; Y- 41564536 2. X- 42383361; Y- 41561612 3. X- 42379605; Y- 41552244 4. X- 42387503; Y- 41546123 5. X- 42390662; Y- 41546307 6. X- 42392457; Y- 41548551 7. X- 42392600; Y- 41557359		1.379

ნახ.5.28. დასასრული

შავი ზღვის აუზის საქართველოს ძირითადი მდინარეების

ჰიდროლოგიური მაჩვენებლები ცხრ.5.5

#	მდინარის დასახელება	წყალშემკრები აუზის ფართობი, კმ2	საშუალო სიმაღლე, მ	მდინარის სიგრძე, კმ	საშუალო ქანობი, ი	აუზის საშუალო სიმაღლე მონაკვეთზე, მ	ჩამონადენის საშუალო მოდული, ლ/წმ.კმ²	საშუალო წლიური ხარჯი მ³/წმ
1	ჭოროხი (საქ. საზღ.)	368,1	55,0	26,0	0.00211	1420	57,5	283,0
2	კინტრიში	284,0	2450,0	25.2	0.09722	835	65,0	18,50
3	ნატანები	489,7	2475,0	60,0	0.04125	870	50,0	24,50
4	სუფსა	1105,5	2709,7	108,0	0.02509	970	41,0	45,20
5	რიონი (სამხრეთ განშტოება)	13418,2	2347,0	327,0	0.00717	1660	38,6	415,0
6	რიონი (ჩრდილ. განშტოება)	13418,2	2347,0	327,0	0.00717	1660	38,6	415,0
7	ხობისწყალი	1340	2325,6	150,0	0.01550	590	46,0	51,20
8	ენგური	4058,5	2614	213,0	0.01227	1840	48,5	173,00

საველე-სამეცნიერო კვლევების პერიოდში ჩვენი ყურადღება ასევე გამახვილდა საქართველოს საზღვაო პორტებში (ბათუმი, ფოთი, ანაკლია) და ნავთობსადენის ტერმინალების (ბაქო-თბილისი-სუფსა და ყულევი) აკვატორიაში თუ როგორ იცვლებოდა წყლის მჟავიანობისა (pH) და მარილიანობის (TDS) რაოდენობრივი მაჩვენებლები. საველე კვლევის შედეგები მოყვანილია 5.6 და 5.7 ცხრილებში.

მდინარეთა ესტუარებში საველე კვლევების შედეგები ცხრ.5.6

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპ. (t1/t2)	წყლის მჟავიანობა pH	მარილიანობა (TDS)
		X	YY			
1	ჭოროხი-მარცხენა ნაპირი	41596952	41569943	0,69	7,14	9,70
2	ჭოროხი-მარჯვენა ნაპირი	41600395	41571039	0,69	7,14	9,78
3	კიწტრიში	41811607	41771416	0,72	7,15	5,23
4	ნატანები	41913572	41767241	0,70	7,45	4,34
5	სუფსა	42016078	41753594	0,92	7,13	9,67
6	რიონი (სამხრეთ განშტოება)	42134187	41659283	0,78	7,84	6,37
7	რიონი (ჩრდილ. განშტოება)	42177666	41641295	0,77	7,82	7,80
8	ხობისწყალი	42259918	41637102	0,82	7,71	6,76
9	ენგური	42389302	41560674	0,73	7,73	7,67

საქართველოს საზღვაო პორტებში და ნავთობსადენის ტერმინალებში 2015 წ. 6.06-ის მონაცემები ცხრ.5.7

#	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპ. (t1/t2)	წყლის მჟავიანობა pH	მარილიანობა (TDS)
		X	YY			
1	მდინარე ჭოროხი	41600395	41548731	0,92	8,22	6,60
2	ბათუმის პორტი	41662161	41678955	0,86	8,42	11,80
3	მდინარე სუფსა	42016078	41753594	0,85	8,13	6,15
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინ.	42038751	41735281	0.90	8,36	11,67
5	მდინარე ხობისწყალი	42259918	41637102	0,82	7,71	6,76
6	ყულევის ნავთობ-ბის ტერმინალი	42276524	41631693	0,78	8,37	15,12
7	მდინარე რიონი	42134187	41659283	0,78	7,84	6,37
8	ფოთის პორტი	41655297	42147724	0,78	8,65	13,34
9	მდინარე ენგური	42389302	41560674	0,77	8,32	12,67
10	ანაკლიის მშენებარე პორტი	42382543	41577101	0,76	8,52	13,45

შავი ზღვის აკვატორიაში ჩვენს მიერ ჩატარებული საველე-სამეცნიერო კვლევები იძლევა საშუალებას სტატისტიკური რიგის, საიმედოობისა და რისკის თეორიების გამოყენებით [75,76], მონაცემთა ბაზის სერვერში მოგროვილი ინფორმაციის საფუძველზე, მოვახდინოთ ეკოლოგიურ მონაცემთა დამუშავება და დავადგინოთ შავი ზღვის ეკოლოგიური პარამეტრების ფუნქციათა ცვლილების ხასიათი. მომდევნო პარაგრაფში განვიხილავთ ამ საკითხებს.

5.4.2. შავი ზღვის ეკოლოგიური პარამეტრების კვლევა საიმედოობისა და რისკის თეორიის გამოყენებით

5.4.2.1. შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის პროგნოზი

შავი ზღვის წყლისა და ჰაერის ტემპერატურის პროგნოზის მიზნით სადოქტორო გრანტის პროგრამის ფარგლებში განხორციელდა საველე-სამეცნიერო კვლებები საქართველოს საზღვრებში, 2015 წლის 4 სეზონის პერიოდში. საველე კვლევის შედეგები მოცემულია ცხრილში # 7, 8 ,9 და10.

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის საშუალო მნიშვნელობა ტოლია:

$$(t_1 / t_2) = \frac{\sum_{i=1}^{n}(t_1/t_2)_i}{N} = \frac{\sum_{i=1}^{140} 11{,}195}{140} = 0{,}79 \qquad (5.1)$$

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის ჰისტოგრამის გრაფიკის ასაგებად საჭიროა ვიცოდეთ ფარდობითი ტემპერატურის მნიშვნელობების გადანაწილება ინტერვალებში და ამ სიდიდეების სიხშირე(m_i), რომელთა სიდიდეებიც მოყვანილია 5.8 ცხრილში.

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის მნიშვნელობები შესაბამინი ინტერვალებით **ცხრ.5.8**

ინტერვალები (t_1/t_2)	0 - 0,3	0,3 – 0,6	0,6 -0,9	0,9 – 1,2
სიხშირე - m_i	2	23	70	45
$f(t_1/t_2)$	0,014	0,164	0,500	0,321

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის ფუნქციის სიდიდე ინტერვალებში გამოითვლება ასე:

$$f(t_1/t_2) = m_i / N, \qquad (5.2)$$

სადაც, N არის ფარდობითი ტემპერატურის სტსტისტიკური რიგის რაოდენობა, რომელიც ჩვენ კონკრეტული მაგალითისათვის ტოლია N = 140.

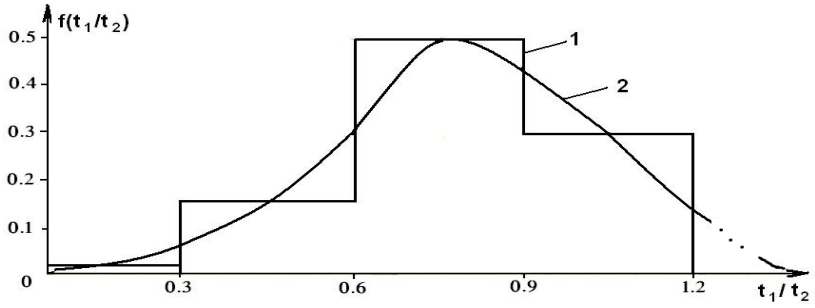
შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t1/ t2)* მათემატიკური ლოდინი გამოითვლება შემდეგი დამოკიდებულებით:

$$m = \sum_{i=1}^{n} f(t_1/t_2) = 0{,}15 * 0{,}014 + 0{,}45 * 0{,}164 + 0{,}75 * 0{,}500 + 1{,}05 * 0{,}321 = 0{,}78 \quad (5.3)$$

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t1/ t2)* საშუალო კვადრატული გადახრა იანგარიშება შემდეგი ფორმულით:

$$\sigma = \sqrt{\frac{\sum_{i=i}^{n}[(t_1/t_2)_i - (t_1/t_2)]^2}{N}} = \sqrt{\frac{0{,}1936}{140}} = 0{,}089 \quad (5.4)$$

5.8 ცხრილის მონაცემების მიხედვით ვაგებთ შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t1/ t2)* ჰისტოგრამასა და შესაბამის თეორიულ მრუდს [75,76], მისი გრაფიკებიც მოცემულია 5.29 ნახაზზე.



ნახ.5.29. შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t1/ t2)* ჰისტოგრამის გრაფიკი და შესაბამისი თეორიული განაწილების მრუდი

შავი ზღვის წყლისა (*t1*) და ჰაერის (*t2*) ტემპერატური ფარდობითი საველე მონაცემები (04.2015 წ., გაზაფხული) ცხრ.5.9

N	*(t1/ t2)i*	$[(t_1/t_2)_i - (t_1/t_2)]$	$[(t_1/t_2)_i - (t_1/t_2)]^2$
1	0.93	0.07	0.0049
2	0.93	0.07	0.0049
3	0.92	0.06	0.0036
4	0.92	0.06	0.036
5	0.92	0.06	0.0036
6	0.96	0.1	0.01
7	0.91	0.05	0.0025
8	0.86	0	0.01
9	0.86	0	0.01
10	0.86	0	0.01
11	0.88	0.02	0.004

12	0.75	-0.11	0.0121
13	0.86	0	0
14	0.85	-0.01	0.0001
15	0.88	0.02	0.004
16	1,00	0.14	0.0196
17	0.86	0	0
18	0.86	0	0
19	0.86	0	0
20	0.9	0.04	0.0016
21	0.85	-0.01	0.0001
22	0.91	0.05	0.0025
23	0.91	0.05	0.0025
24	0.90	0.04	0.0016
25	0.9	0.04	0.0016
26	0.83	-0.03	0.0009
27	0.87	0.01	0.0001
28	0.87	0.01	0.0001
29	0.83	-0.03	0.0009
30	0.78	-0.08	0.0064
31	0.74	-0.12	0.0144
32	0.78	-0.08	0.0064
33	0.78	-0.08	0.0064
34	0.78	-0.08	0.0064
35	0.78	-0.08	0.0064
$\sum$	29.28	0	0.1936

შავი ზღვის წყლისა (t_1) და ჰაერის (t_2) ტემპერატური ფარდობითი საველე მონაცემები (06.2015 წ, ზაფხული) ცხრ.5.10

N	$(t_1/t_2)_i$	$[(t_1/t_2)_i - (t_1/t_2)]$	$[(t_1/t_2)_i - (t_1/t_2)]^2$
1	1.07	0.11	0.0121
2	1.05	0.09	0.0081
3	1.06	0.1	0.01
4	1.01	0.05	0.0025
5	0.97	0.01	0.0001
6	0.98	0.02	0.0004

7	1.01	0.05	0.0025
8	1	0.04	0.0016
9	1.02	0.06	0.0036
10	1	0.04	0.0016
11	0.96	0	0
12	0.95	-0.01	0.0001
13	0.95	-0.01	0.0001
14	0.95	-0.01	0.0001
15	0.91	-0.05	0.0025
16	0.95	-0.01	0.0001
17	0.93	0.03	0.0009
18	0.93	0.03	0.0009
19	0.96	0	0
20	1	0.04	0.0016
21	1	0.04	0.0016
22	1	0.04	0.0016
23	1.03	0.07	0.0049
24	1.03	0.07	0.0049
25	0.85	-0.11	0.0121
26	0.86	-0.1	0.01
27	0.85	-0.11	0.00121
28	0.91	-0.05	0.0025
29	0.91	-0.05	0.0025
30	0.87	-0.09	0.0081
31	0.82	-0.14	0.0196
32	0.81	-0.15	0.0225
33	0.79	-0.17	0.0289
34	0.78	-0.18	0.0324
35	0.77	-0.19	0.0361
∑	32.94	0	0.237

შავი ზღვის წყლისა (t_1) და ჰაერის (t_2) ტემპერატურის
ფარდობითი საველე მონაცემები (10.2015, შემოდგომა) ცხრ.5.11

N	$(t_1/t_2)_i$	$[(t_1/t_2)_i - (t_1/\bar{t}_2)]$	$[(t_1/t_2)_i - (t_1/\bar{t}_2)]^2$
1	0.67	-0.14	0.0196
2	0.68	-0.13	0.0169
3	0.68	-0.13	0.0169
4	0.7	-0.11	0.0121
5	0.72	-0.09	0.0081
6	0.81	0	0.001
7	0.83	0.02	0.0004
8	0.91	0.1	0.01
9	0.92	0.11	0.0121
10	0.92	0.11	0.0121
11	0.93	0.12	0.0144
12	0.73	-0.08	0.0064
13	0.74	-0.07	0.0049
14	0.75	-0.06	0.0036
15	0.76	-0.05	0.0025
16	0.68	-0.13	0.0169
17	0.96	0.15	0.0225
18	0.97	0.16	0.0256
19	0.97	0.16	0.0256
20	0.97	0.16	0.0256
21	0.77	-0.04	0.0016
22	0.91	0.1	0.01
23	0.90	0.09	0.0081
24	0.79	-0.02	0.0004
25	0.8	-0.01	0.0001
26	0.81	0	0
27	0.82	0.01	0.0001
28	0.83	0.02	0.0004
29	0.84	0.03	0.0009
30	0.84	0.03	0.0009
31	0.87	0.06	0.0036
32	0.8	-0.01	0.0001
33	0.79	-0.02	0.0004

34	0.77	-0.04	0.0016
35	0.78	-0.03	0.0009
$\sum$	28.62	0	0.286

შავი ზღვის წყლისა (t_1) და ჰაერის(t_2) ტემპერატურის ფარდობითი საველე მონაცემები (12.2015 წ., ზამთარი) ცხრ.5.12

N	$(t_1/t_2)_i$	$[(t_1/t_2)_i - (t_1/t_2)]$	$[(t_1/t_2)_i - (t_1/t_2)]^2$
1	0.58	0.03	0.0009
2	0.58	0.03	0.0009
3	0.63	0.08	0.0064
4	0.63	0.08	0.0064
5	0.63	0.09	0.0081
6	0.65	0.1	0.001
7	0.62	0.07	0.0049
8	0.58	0.03	0.0009
9	0.58	0.03	0.0009
10	0.58	0.03	0.0009
11	0.59	0.04	0.0016
12	0.33	-0.22	0.0484
13	0.58	0.03	0.0009
14	0.57	0.02	0.0004
15	0.59	0.04	0.0016
16	0.63	0.08	0.0064
17	0.3	-0.25	0.0625
18	0.68	0.13	0.0169
19	0.58	0.03	0.0009
20	0.58	0.03	0.0009
21	0.58	0.03	0.0009
22	0.61	0.06	0.0036
23	0.57	0.02	0.0004
24	0.61	0.06	0.0036
25	0.61	0.06	0.0036
26	0.56	0.01	0.0001
27	0.59	0.04	0.0016
28	0.55	0	0

29	0.55	0	0
30	0.56	0.01	0.0001
31	0.37	-0.18	0.0324
32	0.3	-0.25	0.0625
33	0.37	-0.18	0.0324
34	0.37	-0.18	0.0324
35	0.37	-0.18	0.0324
$\sum$	19.06	**0**	**0.378**

5.4.2.2. შავი ზღვის წყლის მარილიანობის (TDS) პროგნოზი

შავი ზღვის ეკოლოგიური პარამეტრებიდან განსაკუთრებული მნიშვნელობა ენიჭება ზღვის წყლის მარილიანობის დაგენას, რომელიც ძირითადად განსაზღვრავს ფლორისა და ფაუნის თანამდროვე მდგომარეობას.

ჩვენს მიერ განხორციელებული საველე კვლევების შედეგად მიღებული სტატისტი-კური რიგი 140 წერტილის ოდენობით იძლევა იმის საშუალებას, რომ საიმედოობისა და რისკის თეორიის გამოყენებით განისაზღვროს ზღვაში მარილიანობის ცვლილების დასადგენი ფუნქციის განაწილების კანონი.

საველე კვლევის შედეგები ასახულია 5.13-:-5.16 ცხრილებში.

შავი ზღვის წყლის მარილიანობის (TDS) საველე მონაცემები (04.2015 წ., გაზაფხული) ცხრ.5.13

#	$(TDS)_i$	$[(TDS)_i - (\overline{TDS})]$	$[(TDS)_i - (\overline{TDS})]^2$
1	13.45	1.79	3.2041
2	12.54	0.88	0.7744
3	13.2	1.54	2.3716
4	13.23	1.57	2.4649
5	12.8	1.14	1.2996
6	12.88	1.22	1.4884
7	6.6	-5.06	25.6036
8	11.8	0.14	0.0196
9	8.66	-3	9
10	7.07	-4.59	21.0681
11	13.2	1.54	2.3716
12	12.86	1.2	1.44

13	6.23	-5.43	29.4849
14	6.15	-5.51	30.3601
15	11.3	-0.36	0.1296
16	5.23	-6.43	41.3449
17	12.98	1.32	1.7424
18	14.65	2.99	8.9401
19	13.87	2.21	4.8841
20	15.27	3.61	13.0321
21	4.34	-7.32	53.5824
22	12.9	1.24	1.5376
23	11.76	0.1	0.01
24	12.67	1.01	1.0201
25	13.5	1.84	3.3856
26	13.6	1.94	3.7636
27	7.22	-4.44	19.7136
28	15.32	3.66	13.3956
29	6.37	-5.29	27.9841
30	13.6	1.94	3.7636
31	7.8	-3.86	14.8996
32	15.12	3.46	11.9716
33	14.45	2.79	7.7841
34	7.67	-3.99	15.9201
35	12.67	1.01	1.0201
$\sum$	408.08	**0**	**380.776**

შავი ზღვის წყლის მარილიანობის (TDS) საველე მონაცემები (06.2015 წ., ზაფხული) ცხრ.5.14

N	$(TDS)_i$	$[(TDS)_i - (\overline{TDS})]$	$[(TDS)_i - (\overline{TDS})]^2$
1	25.7	0.35	0.1225
2	27.1	1.75	3.0625
3	25.8	0.45	0.2025
4	28.2	2.85	8.1225
5	18.8	-6.55	42.9025
6	18.6	-6.75	45.5625
7	26.8	1.45	2.1025
8	27.8	2.45	6.0025

9	25.8	0.45	0.2025
10	25.7	0.35	0.1225
11	26.4	1.05	1.1025
12	23.5	-1.85	3.4225
13	24.7	-0.65	0.4225
14	26.3	0.95	0.9025
15	26.9	1.55	2.4025
16	7.22	-18.13	328.6969
17	24.2	-1.15	1.3225
18	26	0.65	0.4225
19	26.7	1.35	1.8225
20	26.8	1.45	2.1025
21	26.7	1.35	1.8225
22	26.3	0.95	0.9025
23	26.3	0.95	0.9025
24	14.7	-10.65	113.4225
25	26.7	1.35	1.8225
26	26.5	1.15	1.3225
27	26.3	0.95	0.9025
28	24.3	-1.05	1.1025
29	24.4	-0.95	0.9025
30	25.3	-0.05	0.0025
31	24.8	-0.55	0.3025
32	27.3	1.95	3.8025
33	28.2	2.85	8.1225
34	28.1	2.75	7.5625
35	15.3	-10.05	101.0025
$\sum$	408.08	**0**	**380.776**

შავი ზღვის წყლის მარილიანობის (TDS) საველე მონაცემები (10.2015 წ., შემოდგომა) ცხრ.5.15

#	$(TDS)_i$	$[(TDS)_i - (\overline{TDS})]$	$[(TDS)_i - (\overline{TDS})]^2$
1	13.41	2.02	4.0804
2	12.45	1.06	1.1236
3	13.11	1.72	2.9584
4	13.18	1.79	3.2041

5	12.65	1.26	1.5876
6	12.77	1.41	1.9881
7	6.58	-4.78	22.8484
8	11.76	0.37	0.1369
9	8.46	-2.93	8.5849
10	7.01	-4.38	19.1844
11	13.11	1.72	2.9584
12	12.76	1.37	1.8769
13	6.1	-5.29	27.9841
14	6.09	-5.3	28.09
15	11	-0.39	0.1521
16	5.09	-6.3	39.69
17	12.87	1.48	2.1904
18	14.58	3.19	10.1761
19	13.81	2.42	5.8564
20	15.21	3.82	14.5924
21	4.31	-7.08	50.1264
22	12.7	1.31	1.7161
23	11.29	-0.1	0.01
24	12.45	1.06	1.1236
25	13.12	1.73	2.9929
26	13.28	1.89	3.5721
27	6.87	-4.52	20.4304
28	14.89	3.5	0.1225
29	5.87	-5.52	30.4704
30	13.34	1.95	3.8025
31	6.89	-4.5	0.2025
32	14.36	2.97	8.8209
33	14.14	2.75	7.5625
34	7.23	-4.16	17.3056
35	11.78	0.39	0.1521
$\sum$	398.88	**0**	**347.674**

შავი ზღვის წყლის მარილიანობის (TDS) საველე
მონაცემები (12.2015 წ., ზამთარი) ცხრ.5.16

N	$(TDS)_i$	$[(TDS)_i - (\overline{TDS})]$	$[(TDS)_i - (\overline{TDS})]^2$
1	13.25	2.26	5.1076
2	12.31	1.32	1.7424
3	13.1	2.11	4.4521
4	13.07	2.08	4.3264
5	12.41	1.42	2.0164
6	12.51	1.52	2.3104
7	6.43	-4.56	20.7936
8	6.43	-4.56	20.7936
9	11.7	0.71	0.5041
10	8.36	-2.63	6.9169
11	7	-3.99	15.9201
12	12.96	1.97	3.8809
13	12.7	1.71	2.9241
14	5.95	-5.04	25.4016
15	6	-4.99	24.9001
16	12.93	1.94	3.7636
17	4.96	-6.03	36.3609
18	11.7	0.71	0.5041
19	12.51	1.52	2.3104
20	15	4.01	16.0801
21	4.25	-6.74	45.4276
22	12.4	1.41	1.9881
23	11.12	0.13	0.0169
24	12.33	1.34	1.7956
25	13	2.01	4.0401
26	13.17	2.18	4.7524
27	6.5	-4.49	20.1601
28	13.44	2.45	6.0025
29	5.61	-5.38	28.9444
30	13.11	2.12	4.4944
31	6.58	-4.41	19.4481
32	13.96	2.97	8.8209
33	14	3.01	9.0601

34	7.11	-3.88	15.0544
35	11.66	0.67	0.4489
∑	384.64	**0**	**371.46**

შავი ზღვის მარილიანობის (*TDS*) საშუალო მნიშვნელობა ტოლია:

$$(\overline{TDS}) = \frac{\sum_{i=1}^{n}(TDS)_i}{N} = \frac{\sum_{i-1}^{140} 2079,12}{140} = 14,85 \qquad (5.5)$$

შავი ზღვის წყლის მარილიანობის (*TDS*) ჰისტოგრამის გრაფიკის ასაგებად საჭიროა ვიცოდეთ მარილიანობის (*TDS*) მნიშვნელობების გადანაწილება ინტერვალებში და ამ სიდიდეების სიხშირე (m_i), რომელთა სიდიდეებიც მოყვანილია 5.17 ცხრილში.

შავი ზღვის წყლის მარიალიანობის (*TDS*) მნიშვნელობები
შესაბამისი ინტერვალებით **ცხრ.5.17**

ინტერვალები (*TDS*)	0 - 6	6 – 12	12 - 18	18 – 24	24-30
სიხშირე m_i	14	36	59	2	29
f (TDS)	0,100	0,257	0,421	0,014	0,207

ამ ცხრილში მოყვანილი შავი ზღვის მარილიანობის (*TDS*) ფუნქციის სიდიდე ინტერვალებში გამოითვლება შემდეგი დამოკიდებულებით:

$$f(TDS) = m_i / N, \qquad (5.6)$$

სადაც, N არის მარილიანობის (*TDS*) სტსტისტიკური რიგის რაოდენობა, რომელიც ჩვენ კონკრეტული მაგალითისათვის ტოლია N = 140.

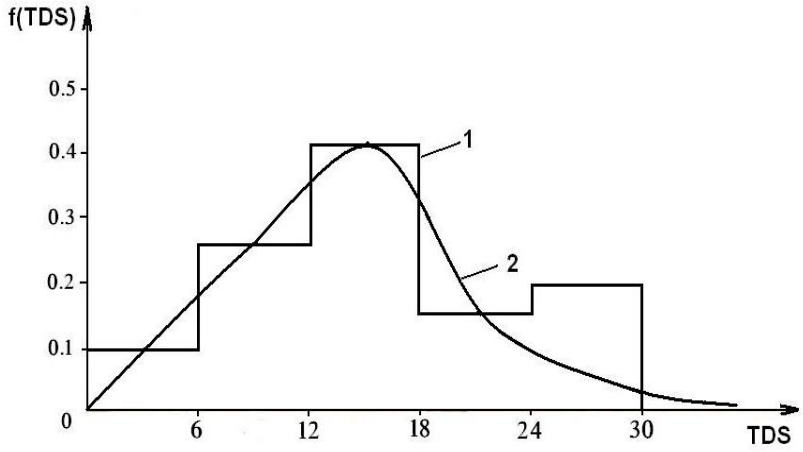
შავი ზღვის წყლის მარილიანობის (*TDS*) მათემატიკური ლოდინი გამოითვლება შემდეგი დამოკიდებულებით:

$$m = \sum_{i=1}^{n} f(TDS) = 0.10 * 3.0 + 0.257 * 9.0 + 0.421 * 15.0 + \\ + 0.014 * 21.0 + 0.207 * 27.0 = 14.81 \qquad (5.7)$$

შავი ზღვის მარილიანობის (*TDS*) საშუალო კვადრატული გადახრა იანგარიშება შემდეგი ფორმულით:

$$\sigma = \sqrt{\frac{\sum_{i=i}^{n}[(TDS)_i - (\overline{TDS})]^2}{N}} = \sqrt{\frac{1804,34}{140}} = 3,59 \qquad (5.8)$$

5.17 ცხრილის მონაცემების მიხედვით ვაგებთ შავი ზღვის მარილიანობის(*TDS*) ჰისტოგრამასა და შესაბამის თეორიულ მრუდს, რომლის გრაფიკებიც მოცემულია 5.30 ნახაზზე.



ნახ.5.30. შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(TDS)* ჰისტოგრამის გრაფიკი და შესაბამისი თეორიული განაწილების მრუდი

5.4.2.3. შავი ზღვის წყლის მჟავიანობის (pH) პროგნოზი

საქართველოს საზღვრებში შავი ზღვის აკვატორიაში ჩვენს მიერ ჩატარებული საველე კვლევები, რომელთა სტატისტიკურმა ასევე შეადგინა 140 წერტილი ასევე იძლევა საშუალებას დადგენილ იქნეს ზღვის წყლის მჟავიანობა (Ph), სტატისტიკური მონაცემები მოყვანილია 5.18-5.21 ცხრილებში.

შავი ზღვის წყლის მჟავიანობის (Ph) საველე მონაცემები (04.2015 წ., გაზაფხული) ცხრ.5.18

N	$(Ph)_i$	$[(Ph)_i - (\bar{P}h)]$	$[(Ph)_i - (\bar{P}h)]^2$
1	8.49	-0.12	0.0144
2	8.38	-0.23	0.0529
3	8.42	-0.19	0.0361
4	8.37	-0.24	0.0576
5	8.38	-0.23	0.0529
6	8.42	-0.19	0.0361
7	8.23	-0.38	0.1444
8	8.34	-0.27	0.0729
9	8.36	-0.25	0.0625
10	8.11	-0.5	0.25

11	8.28	-0.33	0.1089
12	8.27	-0.34	0.1156
13	8.38	-0.23	0.0529
14	8.32	-0.29	0.0841
15	8.35	-0.25	0.0625
16	8.29	-0.32	0.1024
17	8.43	-0.18	0.0324
18	8.56	-0.05	0.0025
19	8.45	-0.16	0.0256
20	8.21	-0.4	0.16
21	8.45	-0.16	0.0256
22	8.43	-0.18	0.0324
23	8.38	-0.23	0.0529
24	8.36	-0.25	0.0625
25	8.43	-0.18	0.0324
26	8.25	-0.36	0.1296
27	8.34	-0.27	0.0729
28	8.39	-0.22	0.0484
29	8.41	-0.2	0.04
30	8.48	-0.13	0.0169
31	8.72	0.11	0.0121
32	8.37	-0.24	0.0576
33	8.32	-0.29	0.0841
34	8.29	-0.32	0.1024
35	8.32	-0.29	0.0841
$\sum$	301.35	**0**	**2.3806**

შავი ზღვის წყლის მჟავიანობის (Ph) საველე მონაცემები
(06.2015 წ., ზაფხული) **ცხრ.5.19**

N	$(Ph)_i$	$[(Ph)_i - (\bar{P}h)]$	$[(Ph)_i - (\bar{P}h)]^2$
1	8.35	-0.2	0.004
2	8.49	-0.06	0.0036
3	8.44	-0.11	0.0121
4	8.41	-0.14	0.0196
5	8.34	-0.21	0.0441

6	8.31	-0.24	0.0576
7	8.38	-0.17	0.0289
8	8.41	-0.14	0.0196
9	8.38	-0.17	0.0289
10	8.47	-0.08	0.0064
11	8.35	-0.2	0.004
12	8.33	-0.22	0.0484
13	8.49	-0.06	0.0036
14	8.33	-0.22	0.0484
15	8.45	-0.1	0.01
16	8.2	-0.35	0.1225
17	8.44	-0.11	0.0121
18	8.28	-0.27	0.0729
19	8.38	-0.17	0.0289
20	8.34	-0.21	0.0441
21	8.33	-0.22	0.0484
22	8.31	-0.34	0.1156
23	8.2	-0.35	0.1225
24	7.96	-0.59	0.3481
25	8.35	-0.2	0.04
26	8.33	-0.22	0.0484
27	8.37	-0.18	0.0324
28	8.25	-0.3	0.09
29	8.26	-0.29	0.0841
30	8.37	-0.18	0.324
31	7.71	-0.84	0.7056
32	8.36	-0.19	0.0361
33	8.35	-0.2	0.04
34	8.35	-0.2	0.04
35	7.89	-0.66	0.4356
$\sum$	299.32	**0**	**3.1305**

შავი ზღვის წყლის მჟავიანობის (Ph) საველე მონაცემები

(10.2015 წ., შემოდგომა) ცხრ.5.20

N	*(Ph)i*	$[(Ph)_i - (\overline{P}h)]$	$[(Ph)_i - (\overline{P}h)]^2$
1	8.41	-0.15	0.0225
2	8.32	-0.24	0.0576
3	8.4	-0.16	0.0256
4	8.31	-0.25	0.0625
5	8.37	-0.19	0.0289
6	8.41	-0.15	0.0225
7	8.18	-0.38	0.1444
8	8.29	-0.37	0.1369
9	8.32	-0.24	0.0576
10	8.07	-0.49	0.2401
11	8.22	-0.34	0.1156
12	8.19	-0.37	0.1369
13	8.33	-0.23	0.0576
14	8.28	-0.38	0.1444
15	8.32	-0.24	0.0576
16	8.21	-0.35	0.1225
17	8.39	-0.17	0.0289
18	8.51	-0.05	0.0025
19	8.41	-0.15	0.0225
20	8.17	-0.39	0.1521
21	8.41	-0.15	0.0225
22	8.37	-0.19	0.0361
23	8.34	-0.22	0.0484
24	8.31	-0.25	0.0625
25	8.41	-0.15	0.0225
26	8.17	-0.39	0.1521
27	8.29	-0.37	0.1369
28	8.33	-0.23	0.0529
29	8.38	-0.18	0.0324
30	8.44	-0.12	0.0144
31	8.66	0.1	0.01
32	8.32	-0.24	0.0576
33	8.29	-0.37	0.1369

34	8.25	-0.31	0.0961
35	8.3	-0.26	0.0676
∑	299.7	**0**	**2.588**

შავი ზღვის წყლის მჟავიანობის (Ph) საველე მონაცემები
(12.2015 წ., ზამთარი) ცხრ.5.21

N	$(Ph)_i$	$[(Ph)_i - (\bar{P}h)]$	$[(Ph)_i - (\bar{P}h)]^2$
1	8.35	0.11	0.0121
2	8.21	-0.03	0.0009
3	8.3	0.06	0.0036
4	8.25	0.01	0.0001
5	8.31	0.07	0.0049
6	8.35	0.11	0.0121
7	8.07	0.17	0.0289
8	8.11	-0.13	0.0169
9	8.18	-0.06	0.0036
10	8	-0.24	0.0576
11	8.14	-0.1	0.01
12	8.09	-0.15	0.0225
13	8.23	-0.01	0.0001
14	8.19	-0.05	0.0025
15	8.17	-0.07	0.0049
16	8.15	-0.09	0.0081
17	8.21	-0.03	0.0009
18	8.37	0.13	0.0169
19	8.4	0.16	0.0256
20	8.08	-0.16	0.0256
21	8.27	0.03	0.0009
22	8.14	-0.1	0.01
23	8.21	-0.03	0.0009
24	8.17	-0.07	0.0049
25	8.33	0.09	0.0081
26	8.04	-0.2	0.0004
27	8.11	-0.13	0.0169
28	8.27	0.03	0.0009

29	8.29	0.05	0.0025
30	8.31	0.07	0.0049
31	8.5	0.26	0.0676
32	8.25	0.01	0.0001
33	8.21	-0.03	0.0009
34	8.17	-0.07	0.0047
35	8.21	-0.03	0.0009
∑	288.38	**0**	**0.3824**

შავი ზღვის წყლის მჟავიანობის (Ph) საშუალო მნიშვნელობა ტოლია:

$$(\bar{P}h) = \frac{\sum_{i=1}^{n}(Ph)_i}{N} = \frac{\sum_{i-1}^{140} 1188{,}75}{140} = 8{,}49 \qquad (5.9)$$

შავი ზღვის წყლის მჟავიანობის (Ph), ჰისტოგრამის გრაფიკის ასაგებად საჭიროა ვიცოდეთ მჟავიანობის (Ph), მნიშვნელობების გადანაწილება ინტერვალებში და ამ სიდიდეების სიხშირე(*mi),* რომელთა სიდიდეებიც მოყვანილია 3.15 ცხრილში.

შავი ზღვის წყლის მჟავიანობის (Ph) მნიშვნელობები შესაბამინი ინტერვალებით **ცხრ.5.22**

ინტერვალები (Ph)	7,7 – 8,0	8,0 – 8,3	8,3 – 8,6	8,6 – 8,9
სიხშირე *mi*	4	50	84	2
f(Ph)	0,028	0,357	0,600	0,014

ცხრილში მოყვანილი შავი ზღვის წყლის მჟავიანობის (Ph) ფუნქციის სიდიდე ინტერვალებში გამოითვლება დამოკიდებულებით:

$$f(\mathrm{Ph}) = m_i / N, \qquad (5.10)$$

სადაც, N არის მჟავიანობის (Ph), სტსტისტიკური რიგის რაოდენობა, რომელიც ჩვენ კონკრეტული მაგალითისათვის ტოლია N = 140.

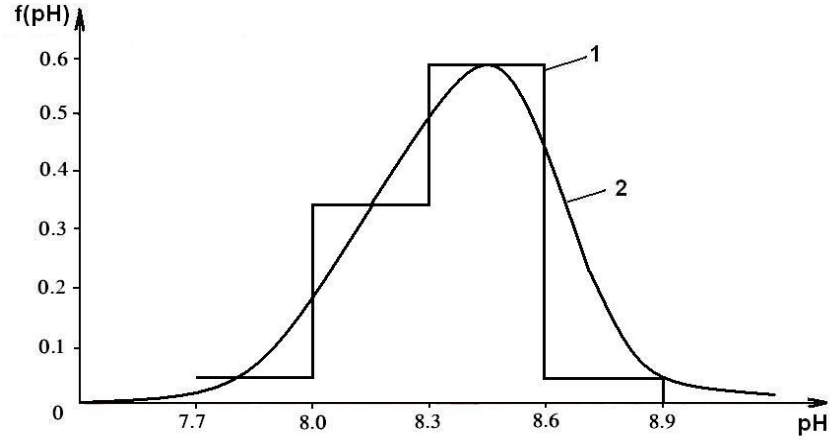
შავი ზღვის წყლის მჟავიანობის (Ph), მათემატიკური ლოდინი გამოითვლება შემდეგი დამოკიდებულებით:

$$m = \sum_{i=1}^{n} f(Ph) = 0{,}028 * 7{,}85 + 0{,}357 * 8{,}15 + 0{,}600 * 8{,}45 + \\ + 0{,}014 * 8{,}75 = 8{,}32 \qquad (5.11)$$

შავი ზღვის წყლის მჟავიანობის (Ph), საშუალო კვადრატული გადახრა იანგარიშება ფორმულით:

$$\sigma = \sqrt{\frac{\sum_{i=i}^{n}[(Ph)_i - (\bar{P}h)]^2}{N}} = \sqrt{\frac{8{,}4815}{140}} = 0{,}25 \qquad (5.12)$$

5.22 ცხრილის მონაცემების მიხედვით ვაგებთ შავი ზღვის წყლის მჟავიანობის (Ph) ჰისტოგრამასა და შესაბამის თეორიული განაწილების მრუდს (ნახ.5.31).



ნახ.5.31. შავი ზღვის წყლის მჟავიანობის (pH) ჰისტოგრამის გრაფიკი და შესაბამისი თეორიული განაწილების მრუდი

ამრიგად სტატისტიკურ მასალაზე დაყრდნობით, რომელთა რიგი უტოლდება 420 წერტილს, საიმედოობისა და რისკის თეორიის გამოყენებით დადგენილია შავი ზღვის წყლისა *(t_1)* და ჰაერის (t_2) ფარდობითი ტემპერატურის (t_1/t_2), წყლის მარილიანობის (*TDS*) და მჟავიანობის (*pH*) ფუნქციის სიმკვრივის განაწილების კანონი.

5.4.2.4. საქართველოს საზღვაო პორტებში და ნავთობსადენის ტერმინალებში შავი ზღვის ეკოლოგიური პარამეტრების კვლევა

საქართველოს საზღვაო პორტებსა და ნავთობსადენის ტერმინალებში შავი ზღვის ეკოლოგიური პარამეტრების დამუშავების მიზნით, რომელშიც გათვალისწინებულია ზღვის წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), წყლის მჟავიანობისა (pH) და მარილიანობის (TDS) რაოდენობრივი მაჩვენებლების პროგნოზი, გამოყენებულ იქნა ის სტატისტიუკური რიგი, რომლიც შექმნილია ჩვენს მიერ ჩატარებული საველე-ექსპერიმენტული და ლაბორატორიული კვლევების მიხედვით, მათი რაოდენობრივი მაჩვენებლები მოყვანილია 5.23 ცხრილში.

შავი ზღვის საველე მონაცემები ცხრ.5.23

N	ფარდობითი ტემპერატურა (t_1/t_2)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
1	0,91	8,33	12,8
2	0,86	8,08	6,97
3	0.90	8,30	11,2
4	0,83	8,38	12,3
5	0,87	8,33	13,5
6	0,78	8,38	12,3
7	0,87	8,37	13,1
8	0,83	8,22	6,25
9	0,78	8,32	9,29
10	0,78	8,29	10,4
11	0,79	8,53	12,3
12	0,76	8,42	11,80
13	0,78	8,6	6,22
14	0.94	8,53	10,8
15	0,78	6,56	7,09
16	0,89	8,53	5,84
17	0,58	8,56	12,1
18	0,58	8,01	6,34
19	0,89	8,53	5,84
20	0,92	8,03	10,82
21	0,92	8,22	6,60
22	0,86	8,42	11,80
23	0,85	8,13	6,15
24	0.90	8,36	11,67
25	0,82	7,71	6,76
26	0,78	8,37	15,12
27	0,78	7,84	6,37
28	0,78	8,65	13,34
29	0,77	8,32	12,67
30	0,76	8,52	13,45
31	0,33	8,22	23,3
32	0,40	8,41	25,6
33	0,50	8,18	15,6
34	0,50	8,31	26,9

35	0,53	8,34	16,3
36	0,55	7,97	25,6
37	0,60	8,35	26,2
38	0,50	7,85	25,1
39	0,55	8,21	25,6
40	0,55	8,13	26,8

აზომილი მონაცემების საფუძველზე სტატისტიკური მეთოდით ჩატარდა გაანგარიშებები, რომელთა რაოდენობრივი მნიშვნელობები მოყვანილია 5.24-:-5.26 ცხრილებში.

შავი ზღის წყლისა და ჰაერისა ფარდობითი ტემპერატურის $(t_1/t_2)_i$ სტატისტიკური გაანგარიშება ცხრ.5.24

N	ფარდობითი ტემპერატურის $(t_1/t_2)_i$ მნიშვნელობები	$[(t_1/t_2)-(t_1\bar{/}t_2)]$	$[(t_1/t_2)-(t_1\bar{/}t_2)]^2$
1	0,91	0,24	0,0576
2	0,86	0,19	0,0361
3	0.90	0,23	0,0529
4	0,83	0,16	0,0256
5	0,87	0,17	0,0289
6	0,78	0,11	0,0121
7	0,87	0,17	0,0289
8	0,83	0,16	0,0256
9	0,78	0,11	0,0121
10	0,78	0,11	0,0121
11	0,79	0,12	0,0144
12	0,76	0,09	0,0081
13	0,78	0,11	0,0121
14	0.94	0,27	0,0729
15	0,78	0,11	0,0121
16	0,89	0,22	0,0484
17	0,58	-0,09	0,0081
18	0,58	-0,09	0,0081
19	0,89	0,22	0,0484
20	0,92	0,25	0,0625
21	0,92	0,25	0,0625
22	0,86	0,19	0,0361
23	0,85	0,18	0,0324
24	0.90	0,23	0,0529
25	0,82	0,15	0,0225
26	0,78	0,11	0,0121

27	0,78	0,11	0,0121
28	0,78	0,11	0,0121
29	0,77	0,1	0,01
30	0,76	0,09	0,0081
31	0,33	-0,34	0,1156
32	0,4	-0,27	0,0729
33	0,5	-0,17	0,0289
34	0,5	-0,17	0,0289
35	0,53	-0,14	0,0196
36	0,55	-0,12	0,0144
37	0,6	-0,07	0,0049
38	0,5	-0,17	0,0289
39	0,55	-0,12	0,0144
40	0,55	-0,12	0,0144
Σ	**26,81**	**2,57**	**1,1907**

შავი ზღის წყლის მჟავიანობისა (*pH*)

სტატისტიკური გაანგარიშება ცხრ.5.25

N	ფარდობითი (*pH*) მნიშვნელობები	$[(pH)-(p\bar{H})]$	$[(t_1/t_2)-(t_1/t_2)]^2$
1	0,91	0,24	0,0576
2	0,86	0,19	0,0361
3	0.90	0,23	0,0529
4	0,83	0,16	0,0256
5	0,87	0,17	0,0289
6	0,78	0,11	0,0121
7	0,87	0,17	0,0289
8	8,22	-0,03	0,0009
9	8,32	0,07	0,0049
10	8,29	0,04	0,0016
11	8,53	0,28	0,0784
12	8,42	0,17	0,0289
13	8,6	0,35	0,1225
14	8,53	0,28	0,0784
15	6,56	-1,69	2,8561
16	8,53	0,28	0,0784
17	8,56	0,31	0,0961
18	8,01	-0,24	0,0576
19	8,53	0,28	0,0784

20	8,03	-0,22	0,0484
21	8,22	-0,03	0,0009
22	8,42	0,17	0,0289
23	8,13	-0,12	0,0144
24	8,36	0,11	0,0121
25	7,71	-0,54	0,2916
26	8,37	0,12	0,0144
27	7,84	-0,41	0,1681
28	8,65	0,4	0,16
29	8,32	0,07	0,0049
30	8,52	0,27	0,0729
31	8,22	-0,03	0,0009
32	8,41	0,16	0,0256
33	8,18	-0,07	0,0049
34	8,31	0,06	0,0036
35	8,34	0,09	0,0081
36	7,97	-0,28	0,0784
37	8,35	0,1	0,01
38	7,85	-0,4	0,16
39	8,21	-0,04	0,0016
40	8,13	-0,12	0,0144
Σ	**329,8**	**4,2**	**4,6987**

შავი ზღის წყლის მარილიანობის (*TDS*)
სტატისტიკური გაანგარიშება ცხრ.5.26

N	წყლის მარილია-ნობის (*TDS*) i მნიშვნელობები	$[(TDS)_i - (\bar{T}DS)]$	$[(TDS)_i - (\bar{T}DS)]^2$
1	12,8	-0,65	0,4225
2	6,97	-6,48	41,9904
3	11,2	-2,25	5,0625
4	12,3	-1,15	1,3225
5	13,5	0,05	0,0025
6	12,3	-1,15	1,3225
7	13,1	-0,35	0,1225
8	6,25	-7,2	51,84

9	9,29	-4,16	17,3056
10	10,4	-3,05	9,3025
11	12,3	-1,15	1,3225
12	11,8	-1,65	2,7225
13	6,22	-7,23	52,2729
14	10,8	-2,65	7,0225
15	7,09	-6,36	40,4496
16	5,84	-7,61	57,9121
17	12,1	-1,35	1,8225
18	6,34	-7,11	50,5521
19	5,84	-7,61	57,9121
20	10,82	-2,63	6,9169
21	6,6	-6,85	46,9225
22	11,8	-1,65	2,7225
23	6,15	-7,3	53,29
24	11,67	-1,78	3,1684
25	6,76	-6,69	44,7561
26	15,12	1,67	2,7889
27	6,37	-7,08	50,1264
28	13,34	-0,11	0,0121
29	12,67	-0,78	0,6084
30	13,45	0	0
31	23,3	9,85	97,0225
32	25,6	12,15	147,6225
33	15,6	2,15	4,6225
34	26,9	13,45	180,9025
35	16,3	2,85	8,1225
36	25,6	12,15	147,6225
37	26,2	12,75	162,5625
38	25,1	11,65	135,7225
39	25,6	12,15	147,6225
40	26,8	13,35	178,2225
Σ	**538,2**	**104,22**	**1822,04**

შავი ზღვის წყლის ტემპერატურის ფარდობითი (t_1/ t_2), მნიშვნელობის საშუალო სიდიდე ტოლია:

$$(t_1 / t_2) = \frac{\sum_{i=1}^{n}(t_1 / t_2)_i}{N} = \frac{\sum_{i-1}^{40}26{,}81}{40} = 0{,}67 \qquad (5.13)$$

წყლის მჟავიანობის (pH) კი,

$$(pH) = \frac{\sum_{i=1}^{n}(pH)_i}{N} = \frac{\sum_{i-1}^{40}329{,}81}{40} = 8{,}25 \qquad (5.14)$$

მარილიანობის შემთხვევაში მისი საშუალო მნიშვნელობა ტოილია:

$$(TDS) = \frac{\sum_{i=1}^{n}(TDS)_i}{N} = \frac{\sum_{i-1}^{40}538{,}19}{40} = 13{,}45 \qquad (5.15)$$

5.23 ცხრილში მოყვანილი სტატისტიკური მასალის გამოყენებით ვაგებთ ზღვის წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), წყლის მჟავიანობისა (pH) და მარილიანობის (TDS) ჰისტოგრამებს, რომლის მონაცემებიც ნაჩვენებია 5.27-:-5.29 ცხრილებში.

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის (t_1/t_2) მნიშვნელობები შესაბამისი ინტერვალებით ცხრ.5.27

ინტერვალები (t_1/t_2)	0 - 0,33	0,33 – 0,66	0,66 -0,99
სიხშირე m_i	1	11	28
$f(t_1/t_2)$	0,025	0,275	0,700

შავი ზღვის წყლის მჟავიანობის (*pH*) მნიშვნელობები შესაბამისი ინტერვალებით ცხრ.5.28

ინტერვალები (pH)	6-7	7-8	8-9
სიხშირე m_i	1	4	35
f(pH)	0,025	0,100	0,875

შავი ზღვის წყლის მარილიანობის (*TDS*) მნიშვნელობები შესაბამისი ინტერვალებით ცხრ.5.29

ინტერვალები (*TDS*)	0-8	8-16	16-24	24-32
სიხშირე m_i	11	20	2	7
f(*TDS*)	0,275	0,500	0,050	0,0175

საველე-ექსპერიმანტული კვლევებისა და სტატისტიკური მეთოდების გამოყენებით აგებულია საკვლევი სიდიდეების ჰისტოგრამა და თეორიული განაწილების მრუდი, რომელთა გრაფიკებიც მოცემულია 5.32 -:- 5.34 ნახაზებზე.

საქართველოს საზღვაო პორტებში და ნავთობსადენის ტერმინალებში შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t_1/t_2)* მათემატიკური ლოდინი გამოითვლება შემდეგი დამოკიდებულებით:

$$m = \sum_{i=1}^{n} f(t_1/t_2) = 0.165 * 0.025 + 0.495 * 0.275 + \\ + \ 0.825 * 0.70 = 0.717 \qquad (5.16)$$

შავი ზღვის წყლისა და ჰაერის ფარდობითი ტემპერატურის *(t_1/t_2)* საშუალო კვადრატული გადახრა იანგარიშება ფორმულით:

$$\sigma_{(t_1/t_2)} = \sqrt{\frac{\sum_{i=i}^{n}[(t_1/t_2)_i - (t_1/t_2)]^2}{N}} = \sqrt{\frac{1{,}1907}{40}} = 0{,}173 \qquad (5.17)$$

შავი ზღვის წყლის მჟავიანობისათვის (pH) გვექნება:
მათემატიკური ლოდინი:

$$m = \sum_{i=1}^{n} f(Ph) = 0.025 * 6.5 + 0{,}10 * 7.5 + \\ + 0.875 * 8.5 = 8{,}344 \qquad (5.18)$$

შავი ზღვის წყლის მჟავიანობისათვის (pH) საშუალო კვადრატული გადახრა ტოლია:

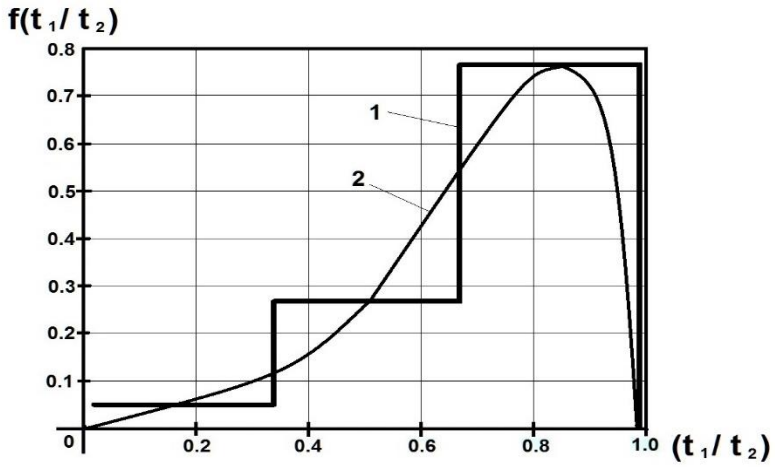
$$\sigma_{(pH)} = \sqrt{\frac{\sum_{i=i}^{n}[(t_1/t_2)_i - (t_1/t_2)]^2}{N}} = \sqrt{\frac{4{,}6987}{40}} = 0{,}343 \qquad (5.19)$$

შავი ზღვის წყლის მარილიანობისათვის (*TDS*) გვექნება:
მათემატიკური ლოდინი:

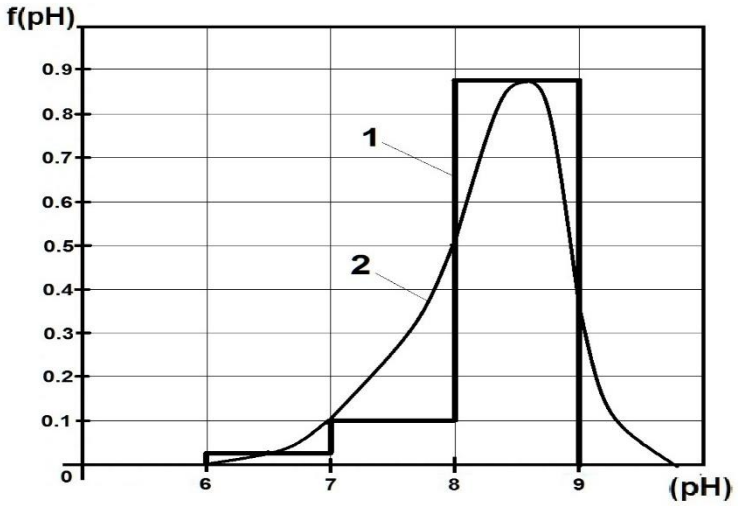
$$m = \sum_{i=1}^{n} f(TDS) = 0.275 * 4.0 + 0.500 * 12.0 + 0.050 * 20.0 + \\ + 0.0175 * 28.0 = 8.59 \qquad (5.20)$$

შავი ზღვის წყლის მარილიანობის (*TDS*) საშუალო კვადრატული გადახრა:

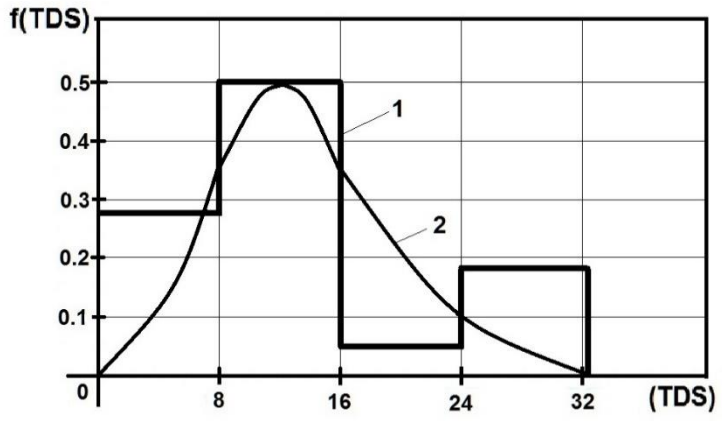
$$\sigma_{(TDS)} = \sqrt{\frac{\sum_{i=i}^{n}[(t_1/t_2)_i - (t_1/t_2)]^2}{N}} = \sqrt{\frac{1822{,}04}{40}} = 6{,}749 \qquad (5.21)$$



ნახ.5.32. ზღვის წყლის ტემპერატურისა (*t1*) და ჰაერის ტემპერატურის (*t2*) ფარდობითი სიდიდეების (*t1/ t2*) ჰისტოგრამა (1) და თეორიული განაწილების გრაფიკი (2)



ნახ.5.33. ზღვის წყლის მჟავიანობის (PH) ჰისტოგრამა (1)
და თეორიული განაწილების გრაფიკი (2)



ნახ.5.34. ზღვის წყლის მარილიანობის (TDS) ჰისტოგრამა (1)
და თეორიული განაწილების გრაფიკი (2)

ამრიგად, საიმედოობისა და რისკის თეორიის გამოყენებით განსაზღვრულია შავი ზღვის აკვატორიაში წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), წყლის მარილიანობის (TDS) და მჟავიანობის (pH) პროგნოზი, რისთვისაც დადგენილია ზემოთ აღნიშნული სიდიდეების ფუნქციის ჰისტოგრამა და თეორიული განაწილების მრუდები.

5.4.2.5. შავი ზღვის აკვატორიაში ზამთრის სეზონის საველე კვლევების განხორციელება

რისთაველის ფონდის საგრანტო პროექტის სამეცნიერო გეგმის მიხედვით შავი ზღვის აკვატორიაში განხორციელდა სველე-სამეცნიერო კვლევები 2016 წლის 25 დეკემბრიდან, 2017 წლის 28 თებერვლის ჩათვლით [1]. ჩვენს მიერ GPS ის კოორდინატებით დაფიქსირებულ რეპერებზე ტრადიციულად გაიზომა წყლის (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), წყლის მჟავიანობისა (PH) და მარილიანობის (TDS) რაოდენობრივი მაჩვენებლები, რომელთა რიცხობრივი მაჩვენებლები მოყვანილია 5.30 –:- 5.32 ცხრილებში. თუ ჩავატარებთ გაზომილი სიდიდეების შეფასებას, დავინახავთ, რომ:

- წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), მნიშვნელობა იცვლება (t_1/t_2) = 0,22 – 0,60 ;
- წყლის მჟავიანობისა (PH) = 7,85 – 8,42 და
- მარილიანობისა (TDS) = 7,29 – 28,1.

ამრიგად თუ შევადარებთ მიღებულ მონაცემებს ჩვენს მიერ წინა წლებში გაზომილ რაოდენობრივ მნიშვნელობებთან, დავინახავთ, რომ რაოდენობრივი მაჩვენებლები ნორმის ფარგლებშია, განსხვავებით ზღვისა და ჰაერის ტემპერატურის ფარდობითი მნიშვნელობისა, რომლის მაჩვენებლებიც იცვლება (t_1/t_2) = 0,22 – 0,60 ფარგლებში.

საველე-ლაბორატორიული სამუშაოები (25.12.2016 -:- 8.01.2017) ცხრ.5.30

N	ადგილის დასახელება	GPS - კოორდინატები		წყლის ტემპ. (C°)	ჰაერის ტემპ. C°	წყლის მჟავიანობა (pH)	მარილი-ანობა (TDS)
		X	YY				
1	სარფი	41526956	41548731	9	2	8,21	25,4
2	კვარიათი_1	41545542	41561587	9	2	8,32	27,3
3	კვარიათი_2	41554651	41563841	9	3	8,35	27,6
4	გონიო	41574588	41565589	9	3	8,41	28,1
5	ჭოროხი-მარცხენა	41596952	41569943	9	3	8,22	23,3
6	ჭოროხი-მარჯვენა	41607866	41577288	9	3	8,21	23,6
7	ადლია	41614371	41583944	10	4	8,38	26,4
8	ბათუმი(დელფინარიუმთან)	41649103	41621114	10	4	8,41	25,6
9	ბათუმი (დასაწყისი)	41650823	41666129	10	4	8,31	25,8
10	ბათუმი (ბენზე)	41662161	41678955	10	5	8,41	25,5
11	მახინჯაური (რკინიგ.სადგ.)	41677322	41694925	10	5	8,28	26,2
12	ჩაქვი	41723714	41727073	10	5	8,30	23,2
13	ბუკნარი	41747684	41737649	10	4	8,42	23,3
14	ციხისძირი	41772644	41755505	9	4	8,31	26,1
15	ბობოყვათი	41797243	41766211	9	4	8,41	26,0
16	კობულეთი (მდ. კიწტრიშთან)	41811607	41771416	9	5	8,17	7,29
17	ქობულეთი აღმაშენებლის 270	41833693	41775383	9	5	8,33	26,4
18	ქობულეთი აღმაშენებლის 552	41856175	41777474	9	5	8,17	26,9
19	ქობულეთის ბოლო	41880721	41772763	9	5	8,23	26,8
20	მდ. ჩოლოკი	41895964	41770675	10	5	8,31	24,5
21	მდ. ნატანები	41913572	41767241	10	5	8,27	25,7
22	სოფ. შეკვეთილი	41938746	41764857	10	5	8,32	21,9
23	Dდაბა ურეკი	41997287	41758008	10	5	8,30	25,9
24	მდ. სუფსა	42016078	41753594	10	5	8,18	15,6
25	სოფ. გრიგოლეთი (დასაწყ.)	42038751	41735281	10	5	8,31	26,9
26	სოფ. გრიგოლეთი (ბოლო)	42056578	41723947	10	6	8,30	26,7
27	მდ. მალთაყვა	42092887	41,695954	10	6	8,32	26,2
28	ქ. ფოთი (მდ. რიონის სამხრეთი ჩადინება)	42134187	41659283	10	6	8,31	25,5
29	ქ. ფოთი (მაშველები)	42140873	41657926	10	6	8,33	26,7
30	ქ. ფოთი (მოლი)	42147686	41,655392	10	5	8,35	26,2
31	ნაბადა (მდ. რიონის ჩრდილოეთი ჩადინება)	42177666	41,641295	9	5	7,85	25,1
32	ყულევი	42259918	41,637102	9	5	7,97	25,6
33	ანაკლია (რეპერთან)	42382543	41,577101	9	5	8,11	26,1
34	ანაკლია (სასტუმროსთან)	4238744	41563028	9	5	8,13	26,6
35	ანაკლია (მდ. ენგურის მარცხენა ნაპირი)	42389302	41560674	9	5	8,21	25,6

საველე კვლევის შედეგები საქართველოს საზღვაო პორტებში და ნავთობსადენის ტერმინალებში (25.12.16 -:- 8.01.17) ცხრ.5.31

N	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატ. (t ₁ / t₂)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
		X	YY			
1	მდინარე ჭოროხი	41600395	41548731	0,33	8,22	23,3
2	ბათუმის პორტი	41662161	41678955	0,40	8,41	25,6
3	მდინარე სუფსა	42016078	41753594	0,50	8,18	15,6
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინალი	42038751	41735281	0,50	8,31	26,9
5	მდინარე ხობისწყალი	42259918	41637102	0,53	8,34	16,3
6	ყულევის ნავთობის ტერმინალი	42276524	41631693	0,55	7,97	25,6
7	მდინარე რიონი	42134187	41659283	0,60	8,35	26,2
8	ფოთის პორტი	41655297	42147724	0,50	7,85	25,1
9	მდინარე ენგური	42389302	41560674	0,55	8,21	25,6
10	ანაკლიის მშენებარე პორტი	42382543	41577101	0,55	8,13	26,8

საქართველოს საზღვაო პორტებში და ნავთობსადენის ტერმინალებში 2017 წლის 25-26 თებერვალი ცხრ.5.32

N	ადგილის დასახელება	GPS - კოორდინატები		ფარდობითი ტემპერატურა (t ₁ / t₂)	წყლის მჟავიანობა (pH)	მარილიანობა (TDS)
		X	YY			
1	მდინარე ჭოროხი	41600395	41548731	0,65	8,20	23,1
2	ბათუმის პორტი	41662161	41678955	0,61	8,39	25,3
3	მდინარე სუფსა	42016078	41753594	0,62	8,11	15,2
4	ბაქო-თბილისი-სუფსის ნავთობ-სადენის ტერმინალი	42038751	41735281	0,60	8,13	26,5
5	მდინარე ხობისწყალი	42259918	41637102	0,63	8,28	15,8
6	ყულევის ნავთობის ტერმინალი	42276524	41631693	0,54	8,07	25,9
7	მდინარე რიონი	42134187	41659283	0,52	8,32	26,8
8	ფოთის პორტი	41655297	42147724	0,53	7,80	24,9
9	მდინარე ენგური	42389302	41560674	0,60	8,19	25,2
10	ანაკლიის მშენებარე პორტი	42382543	41577101	0,54	8,09	26,3

რაც შეეხება შავი ზღვის აკვატორიაში საზღვაო პორტებსა და ნავთობსადენის ტერმინალებში დაფიქსირებულ მონაცემებს, იგი შეადგენს:

- წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/t_2), მნიშვნელობა იცვლება (t_1/t_2) = 0,33 – 0,60;

- წყლის მჟავიანობისა (PH) = 7,97 – 8,41 და

- მარილიანობისა (TDS) = 15,6 – 26,9.

საველე კვლევების განხორციელების პერიოდში ჩვენს მიერ შავ ზღვაზე დაფიქსირდა 3-4 ბალიანი შტორმი სადაც ზღვის სანაპიროზე ტალღის სიმაღლე იცვლებოდა 0,6 -1,5 მეტრი. 5.35 ნახაზზე ნაჩვენებია ბათუმში ზღვის სანაპირო ზოლის საერთო ხედი ზღვის ღელვის პროცესში, ხოლო 5.36-ზე კი - ზღვის მიქცევის პროცესი.

ნახ.5.25. ბათუმში ზღვის 3-4 ბალიანი ღელვის პროცესი

ნახ.5.26. ზღვის მიქცევის პროცესი

5.4.2.6. შავი ზღვის წყლის დაბინძურების ლაბორატორიული ქიმიური გამოკვლევა

შავი ზღვის აკვატორიაში, კერძოდ, ბათუმის პორტის მიმდებარე ტერიტორიაზე, რომლის გეოგრაფიული კოორდინატებია **X** = 41.649103, **Y**= 41.621114, აღებულ იქინა წყლის სინჯი და შპს „მულტიტესტში“, სადაც ზღვის წყალს ჩაუტარდა ლაბორატორიული ანალიზი, დადგინდა შავი ზღვის წყლის მძიმე მეტალებით - თუთია (Zn^{2+}), რკინა (Fe), კადმიუმი (Cd), სპილენძი(Cu) და ტყვია (Pb) დაბინძურების მაჩვენებლები;

ლაბორატორიული კვლევის მონაცემები მოცემულია 5.33 ცხრილში.

შავი ზღვის წყლის მძიმე მეტალებით დაბინძურების რაოდენობრივი მაჩვენებლები ცხრ.5.33

#	ნივთიერების დასახელება	შედეგი	განზომი-ლება	მეთოდი
1	თუთია (Zn^{2+})	< 0,01	მგ/დმ^3	სსტ ისო 11885:2007
2	რკინა (Fe)	0,11	მგ/ლ	სსტ ისო 11885:2007
3	კადიუმი (Cd)	0,003	მგ/ლ	სსტ ისო 11885:2007
4	სპილენძი (Cu)	0,008	მგ/ლ	სსტ ისო 11885:2007
5	ტყვია (Pb)	0,0085	მგ/ლ	სსტ ისო 11885:2007

შავი ზღვის წყლის დაბინძურების დასადგენად გამოყენებულია ევროპის ურთიერთდახმარების საბჭოს რეკომენდაციები, კერძოდ, „წყლის ხარისხის საერთო კრიტერიუმები”, რომლის მიხედვით გამოყოფილია წყლის ხარისხის 6 კლასი. მისი რაოდენობრივი მაჩვენებლები მოცემულია 5.34 ცხრილში.

სამრეწველო არაორგანული დამაბინძურებელი ნივთიერებების მაჩვენებლები ცხრ.3.34

მაჩვენებლები	წყლის ხარისხის კლასები					
	I	II	III	IV	V	VI
კადმიუმი, მკგ/ლ	<3	5	10	20	30	>30
ტყვია,მკგ/ლ	<10	20	50	100	200	>200
სპილენძი, მკგ/ლ	<20	50	100	200	500	>500
თუთია, მგ/ლ	<0.2	1.0	2.0	5.0	10.0	>10.0

ჩვენს მიერ ჩატარებული ანალიზის შედეგები შედარებულ იქნა ზემოაღნიშნულ კრიტერიუმებს, რომლის მიხედვითაც: მეტალების (თუთია, რკინა, კადმიუმი, სპილენძი და ტყვია) შემცველობა შეესაბამება წყლის ხარისხის I კლასს.

შედეგები ასევე შედარებულ იქნა ზედაპირული წყლების დაბინძურების სანიტარულ ნორმებთან (СанПиН №4630-88 Министерство здравоохранения СССР Москва-1988 г.),

რომლის მიხედვითაც ყველა მეტალის შემცველობა ნაკლებია აღნიშნული დოკუმენტით განსაზღვრულ ზღვრულ დასაშვებ კონცენტრაციაზე.

გამოვიყენეთ ასევე საქართველოს გარემოს დაცვის სამინისტროს გარემოს ეროვნული სააგენტოს ზღვის და მტკნარი წყლების ჰიდროქიმიური მაჩვენებლების ზღვრული დასაშვები კონცენტრაციები (ზდკ), რომლის მიხედვითაც ყველა პარამეტრიც ნაკლებია ზღვრულ დასაშვებ კონცენტრაციაზე.

5.35 ცხრილში მოცემულია გარემოს ეროვნული სააგენტოს ზღვის და მტკნარი წყლების ჰიდროქიმიური მაჩვენებლების ზღვრული დასაშვები კონცენტრაციები (ზდკ).

მტკნარი წყლების ჰიდროქიმიური მაჩვენებლების
ზღვრული დასაშვები კონცენტრაციები ცხრ.5.35

მეტალის დასახელება	ზღვრული დასაშვები კონცენტრაციები (ზდკ), მგ/ლ
რკინა (Fe)	0,05
სპილენძი (Cu)	0,05
კადმიუმი (Cd)	0,01
თუთია (Zn)	0,05
ტყვია (Pb)	0,01

ამრიგად, საქართველოს საზღვრებში შავი ზღვის წყლის მძიმე მატელებით დაბინძურების ხარისხი, შეფასებული ევროკავშირისა და საქართველოს ნორმატივების მიხედვთ, ნაკლებია აღნიშნული დოკუმენტით განსაზღვრულ ზღვრულ დასაშვებ კონცენტრაციაზე.

დასკვნა:

შავი ზღვის ეკოლოგიური მონიტორინგის მხარდამჭერი კომპიუტერლი სისტემის პროგამული უზრუნველყოფის შესამუშავებლად წინამდებარე ნაშრომის ფარგლებში ჩატარდა რიგი საპროექტო, ანალიზური და კვლევითი სამუშაოები, რომელთა საფუძველზეც შესაძლებელია შემდეგი დასკვნების ჩამოყალიბება:

- შავი ზღვის ეკოლოგიური მონიტორინგის ელექტრონული სისტემა მიეკუთვნება ორგანიზაციული მართვის რთული და დიდი სისტემების კლასს, რომლის შექმნა, დანერგვა და ექსპლუატაცია დაკავშირებულია ტექნიკურ, ტექნოლოგიურ, ეკონომიკურ და საკადრო პრობლემებთან, გარკვეული ინვესტიციებისა და კომპლექსური ღონისძიებების გატარების აუცილებლობასთან, მათ შორის ახალი გამოყენებითი პროგრამული სისტემების შექმნასთან სპეციალური მობილური და ჰიბრიდული ინფორმაციული ტექნოლოგიებისა და სერვის-ორიენტირებული არქიტექტურის ბაზაზე;
- შავი ზღვის ეკოლოგიური მონიტორინგის ტრადიციული სისტემის ბიზნეს-პროცესების ობიექტ-ორიენტირებული ანალიზის საფუძველზე, რომელიც განხორციელდა უნიფიცირებული მოდელირების ენის (UML – Unified modling language) ტექნოლოგიით, განისაზღვრა დასაპროექტებელი კომპიუტერული სისტემის ფუნქციონალური მოთხოვნილებანი და მისი პროგრამული რეალიზაციის ხელშემწყობი ინფრასტრუქტურა;
- განსაკუთრებული ყურადღება ექცევა შავი ზღვის ეკოლოგიური პარამეტრების განსაზღვრის, მათი მოპოვების, შენახვის და ცენტრალურ სერვერზე გადაცემის საკითხებს. ამასთანავე სპეციალური მულტიმედიური, ობიექტ-რელაციური ბაზების დაპროექტების პროცესების ავტომატიზაციას და სისტემის სერვერზე მათ განთავსებას. მონაცემთა ბაზის ლოგიკური სტრუქტურის ასაგებად გამოყენებულ იქნა ეკომონიტორინგის საპრობლემო სფეროს კონცეპტუალური სქემის ავტომატიზებული დაპროექტება ობიექტ-როლური მოდელირების (ORM/ERM) საფუძველზე;
- შავი ზღვის ეკომონიტორინგის კომპიუტერული სისტემის ფუნქციონალური მომხმარებლებისთვის, მათი ამოცანების გადაწყვეტის სცენარების ანალიზის საფუძველზე, შემუშავებულ იქნა შესაბამისი როლური ინტერფეისები და მონაცემთა ბაზასთან წვდომის და ინფორმაციის განახლებისა და დამუშავების სერვისული პროგრამები;
- ეკომონიტორინგის სისტემის ინფრასტრუქტურა რეალიზებულია სერვის-ორიენტირებულ არქიტექტურით, პროგრამირების მობილური და ჰიბრიდული ტექნოლოგიების გამოყენების მიზნით. სისტემის ვებ-პორტალი აგებულია „მაიკროსოფთის“ SharePoint ტექნოლოგიით, რომელიც დაკავშირებულია Ms SQL Server-თან. ღრუბლოვანი სერვისების საშუალებით მოქნილად ხორციელდება ზღვის სანაპიროს საკონტროლო წერტილებიდან

ეკოლოგიური მაჩვენებლების ოპერატიული მნიშვნელობების გადაცემა სერვერზე მომხმარებელთა აუტენთიფიკაციის, GPS კოორდინატების და თარიღი/დროითი პარამეტრების დაცვით;

- კორპორაციული აპლიკაციების ინტეგრირებული პაკეტი Ms SharePoint Server თავსებადია მობილური მოწყობილობების ოპერაციულ სისტემებთან: Windows Phone, Windows 7-10, iOS, Android. შესაბამისად, მეტად მოსახერხებელია ის ფაქტი, რომ ეკომონიტორინგის კომპიუტერული სისტემის თანამშრომელი მონაცემების შეტანას შეძლებს ტერიტორიულად დაშორებული კომპიუტერიდან ან მობილური ტელეფონიდან. მონაცემები შეტანისთანავე აისახება კორპორაციული პორტალის ვებ-გვერდზე და ტერიტორიულად დაშორებულ SQL Server-ის ბაზაში;
- შავი ზღვის აკვატორიაში GPS და GIS პროგრამების გამოყენებით განხორციელებული საველე-სამეცნიერო კვლევებით დაფიქსირდა 35 სენსიტიური უბანი, რომლის მნიშვნელობებიც დატანილ იქნა ციფრულ რუკაზე და სადაც 3 წლის განმავლობაში მიმდინარეობდა მეცნიერული კვლევები. აღნიშნულ სენსიტიურ უბნებზე, ძირითადი მდინარეების ესტუარებზე, ნავთობტერმინალებსა და საზღვაო პორტებში აღებული ზღვის წყლის ანალიზები: წყლისა (t_1) და ჰაერის (t_2) ტემპერატურები, ლაბორატორიულ პირობებში დადგენილი ზღვის წყლის მჟავიანობის (pH) და მარილიანობის რაოდენობრივი მაჩვენებლები (TDS) იგზავნებოდა სისტემის სერვერ ბაზაში;
- დაზუსტებულია საქართველოს ძირითადი მდინარეების: ესტუარების ფართობები რომელთა სიდიდეც იცვლება 0,861 – 20,390 კმ^2 საზღვრებში, ხოლო ლაბორატორიული კვლევებით დადგენილია რომ შავი ზღვის წყლის მჟავიანობა (pH) მდინარეთა ესტუარებში იცვლებოდა pH = 7,71 – 8,22, საზღვრებში, ხოლო პორტების აკვატორიაში კი - pH = 8,42 - 8,65, რაც შეეხება მარილიანობის მაჩვენებელს(TDS) ის იცვლება მდინარეთა ესტუარებში TDS = 6,15 – 12,67, პორტების შემთხვევაში კი TDS = 11,80 – 13,67 საზღვრებში, რაც აუცილებლივ მხედველობაში უნდა იქნას მიღებული შავი ზღვის სანაპირო ზოლში ეკოლოგიური პარამეტრების კომპლექსური კვლევისას;
- საველე-სამეცნიერო კვლევების შედეგად მიღებული სტატისტიკური რიგისა და საიმედოობისა და რისკის თეორიის გამოყენებით დადგენილია შავი ზღვის აკვატორიაში განთავსებული საზღვაო პორტებში და ნავთობსადენის ტერმინალებში შავი ზღვის ეკოლოგიური პარამეტრების - ზღვის წყლისა (t_1) და ჰაერის ტემპერატურის (t_2) ფარდობითი სიდიდეების (t_1/ t_2), წყლის მჟავიანობისა (pH) და მარილიანობის (TDS) ჰისტოგრამა და თეორიული განაწილების მრუდი, რომელიც ზემოთ აღნიშნული ეკოლოგიური პარამეტრების პროგნოზირების საშუალებას იძლევა;

- ლაბორატორიულ პირობებში დადგინდა შავი ზღვის წყლის მძიმე მეტალებით - თუთია(Zn^{2+}), რკინა(Fe), კადიუმი(Cd), სპილენძი(Cu), ტყვია(Pb) დაბინძურების რაოდენობრივი მაჩვენებლები, რომელთა მაჩვენებლებიც ევროკავშირისა და საქართველოს ნორმატივების მიხედვთ ნაკლებია აღნიშნული დოკუმენტით განსაზღვრულ ზღვრულ დასაშვებ კონცენტრაციაზე;
- დადგენილ იქნა, რომ ეკომონიტორინგის კომპიუტერული სისტემის საფუძველზე შესაძლებლობა იქნება, რომ ოპერატიულად და კომპლექსურად შეფასდეს შავი ზღვის თანამედროვე ეკოლოგიური პრობლემები და დაიგეგმოს მისი სანაპირო ზოლისა და მიმდებარე ტერიტორიების ეკოლოგიური უსაფრთხოების ღონისძიებები.

ლიტერატურა:

1. გავარდაშვილი ა. შავი ზღვის ეკოლოგიური პარამეტრების კვლევა მულტი-მედიური ბაზების საფუძველზე. მონოგრ., ISBN 978-9941-26-067-4. სტუ-ს ც. მირცხულავას სახ. წყალთა მეურნეობის ინსტ., გამომც. „უნივერსალი“, თბ., 2017

2. დიაკონიძე რ., შენგელია ე., გავარდაშვილი გ., ჩახიაი გ., წულიკიძე ლ., ვარაზაშვილი ზ., სუპატაშვილი თ., დიაკონიძე ბ. საქართველოს შავიზღვისპირა კურორტების მოკლე დახასიათება, ზღვის აკვატორიაში და მასში ჩამდინარე მდინარეების წყლის ხარისხის შეფასება. გამომც. ,,უნივერსალი“ თბილისი, 2016

3. გრიგოლია გ., კერესელიძე დ., ბილაშვილი კ., ტრაპაიძე ვ. კლიმატის ცვლილებასთან დაკავშირებით მდინარეთა დელტაში წყალდიდობებისა და წყალმოვარ-დნების რისკების შეფასება მდინარე რიონის მაგალითზე. 1-საერთ.კონფ. შრ.კრ. „კოლხეთის დაბლობის წყლის ეკოსისტემები - დაცვა და რაციონალური გამოყენება“. თბ., 2013, გვ.11-15

4. ფრანგიშვილი ა., ციხელაშვილი ზ., გველესიანი თ. შავი ზღვის აუზის ქვეყნების მდინარეების წყლის დაბინძურების ხარისხის შეფასება-პროგნოზირების ერთიანი მონიტორინგის სუსტენის შექმნისა და ეკოლოგიური-პრევენციული ღონისძიებების შემუშავების შესახებ. სამეცნ.ტექნ. ჟურნ.,,მშენებლობა“ #1(16) , თბ., 2010, გვ.22-25

5. Кордзадзе А.А., Деметрашвили Д.И. Краткосрочный прогноз гидрофизических полей в восточной части Чёрного моря. Изв. РАН, Физика атмосферы и океана. -М., 2013, т.49, № 6, с.733-745.

6. Gavardashvili A. Results of the field-and-scientific study in the water area of the estuaries of the major rivers of the Black Sea and sea ports on the territory of Georgia. 17th Intern.Conf. on Environmental Sciences and Engineering. Paris, France, 2015, pp. 2305-2309

7. Surguladze G., Petriaschvili L,, Surguladze Gio. Decision Support System for Optimization of Seaport Resources with considering multimodal transportation. III internat. Scientific Conference "Computing/Informatics, Education Sciences, Teacher Education". Batumi, Georgia, 2015. pp.139-143

8. Chogovadze G., Surguladze G., Topuria N., Gavardashvili A., Namchevadze Ts. Computer-Aided Design of the Information Ecosystem for the Monitoring of the Black Sea Water Resources. Georg. National Academy of Sciences. Bulletin "Moambe", vol. 12, N.2, 2018, Tb., pp.19-26

9. სურგულაძე გ., თოფურია ნ., გავარადაშვილი ა. ვებ-სერვისის რეალიზაცია შავი ზღვის მდინარეთა ესტუარების მონიტორინგის სისტემისათვის. სტუ შრ.კრებ..: „მას", N 2(22). თბ., 2016, გვ. 190-193

10. ჩოგოვაძე გ., ფრანგიშვილი ა., სურგულაძე გ. მართვის საინფორმაციო სისტემების დაპროგრამების ჰიბრიდული ტექნოლოგიები და მონაცემთა მენეჯმენტი. მონოგრ., ISBN 978-9941-20-790-7. სტუ, „ტექნიკური უნივერსიტეტი", თბ., 2017. -1001 გვ.

11. Gavardashvili A. The Program Software to Create United Database of Black Sea Ecological Characteristics. *Collected Papers of Water Management Iinstitute of Georgian Technical University, vol. 68*, Tbilisi, Georgia, 2013. pp. 27-32

12. Surguladze G., Topuria N., Gavardashvili A., Namchevadze Ts. Automation of Web-portal and Database Construction Processes for the Black Sea Ecosystem Monitoring. International Journal of Environmental and Ecological Engineering. World Academy of Scientific (WASET), v.12, N1. ISSN 1307-6892 Amsterdam, 2018, pp.169-174

13. გოგიჩაიშვილი გ., ბოლხი გ., სურგულაძე გ., პეტრიაშვილი ლ. (2013). მართვის ავტომატიზებული სისტემების ობიექტ-ორიენტირებული დაპროექტების და მოდელირების ინსტრუმენტები (MsVisio, WinPepsy, PetNet, CPN). სახელმძღვ., ISBN 99940-56-77-8. სტუ.. თბ., „ტექნიკური უნივერსიტეტი".

14. Booch G., Jacobson I., rambaugh J. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 1996

15. Surguladze G., Turkia E., Topuria N., Gavardashvili A. Construction of the Multimedia Databases and Users Interfaces for Ecological System of Black Sea with Orm/Erm. "Information and Computer Technology, Modeling and Control". Chapt.45. Nova Science Publishers. © Copyright, 3rd Quarter. 2017, **USA**, pp. 1-8. https://www.novapublishers.com/catalog/product_info.php?products_id=62232

16. Halpin T. (2005). ORM 2 Graphical Notation, Neumont University. http://www.orm.net/pdf/ORM2_TechReport1.pdf.

17. სურგულაძე გ., კივილაძე გ. შესავალი NoSQL მონაცემთა ბაზებში (MongoDB). ISBN 978-9941-0-9642-6. სტუ. თბ., „ITკონსალტინგის ცენტრი". 2017

18. Topuria N. Business Processes Automation SharePoint Server-based, GTU, "Technical University", Tbilisi. (in Georg.). 2017

19. Advanced Android Application Development (4th Edition) (Developer's Library). www.amazon.com/gp/product/0133892387/ref=oh_aui_detailpage_o06_s00?ie=UTF8&psc=1

20. კერესელიძე დ., ბილაშვილი კ., ტრაპაიძე ვ., ბრეგვაძე გ. ზოგადი ოკეანოლოგია და ოკეანეების ჰიდროლოგია. თსუ, თბილისი, 2011

21. სურგულაძე გ., გულიტაშვილი მ., კაკულია ი., ჩერქეზიშვილი გ., ჯავახიშვილი ი. პროგრამული სისტემების სასიცოცხლო ციკლის პროცესის მოდელირება უნივერსალური და ექსტრემალური პროგრამირების პრინციპების კომპრომისული გადაწყვეტით. სტუ-ს შრ.კრ. მას-#1(8), 2008. გვ.63-70

22. სურგულაძე გ. დაპროგრამების მეთოდები: საკურსო პროექტების აგება (UML, MsVisio, C++). ISBN 978-9941-14-125-5. სტუ, თბ., 2007

23. თურქია ე. ბიზნეს–პროექტების მართვის ტექნოლოგიური პროცესების ავტომატიზაცია. სტუ. თბ., 2010

24. სურგულაძე გ., პეტრიაშვილი ლ., ბიტარაშვილი მ. კორპორაციული მენეჯმენტის დაპროგრამების ტექნოლოგია (საკურსო პროექტებისთვის). ISBN 978-9941-0-9843-7. სტუ, თბ., 2017

25. The Unified Modeling Language. http://www.uml-diagrams.org/ უკანასკნ. გადამოწმ. 10.05.14

26. UML: Basics Principles and Background. http://sourcemaking.com/uml

27. სურგულაძე გ., ბოტჰე კ. (გერმანია), კაშიბაძე მ. მემკვიდრეობითობა მართვის ინფორმაციული სისტემების დაპროგრამებაში: მონაცემთა ბაზებიდან UML-ტექნოლოგიამდე. საერთ.კონფ. შრ.კრებ. N4(437), თბ., გვ.55-62

28. სურგულაძე გ., თურქია ე. პროგრამული სისტემების მენეჯმენტის საფუძვლები. ISBN 978-9941-20-651-1. სტუ, „ტექნიკური უნივერსიტეტი“, თბ., 2016

29. სურგულაძე გ., გულუა დ. განაწილებული სისტემების ობიექტ-ორიენტირებული მოდელირება უნიფიცირებული პეტრის ქსელებით. ISBN 99940-48-07-4. სტუ, „ტექნიკური უნივერსიტეტი“. თბ., 2005

30. სურგულაძე გ., ოხანაშვილი მ., სურგულაძე გ. მარკეტინგის ბიზნეს-პროცესების უნიფიცირებული და იმიტაციური მოდელირება. ISBN 978-9941-14-377-9. სტუ, თბ., 2009

31. მეიერ-ვეგენერი კ., სურგულაძე გ., ბასილაძე გ. საინფორმაციო სისტემების აგება მულტიმედიური მონაცემთა ბაზებით. ISBN 978-9941-20-468-5. სტუ, „ტექნიკური უნივერსიტეტი“, თბ., 2014.

32. Codd E.F. Further normalization of the database relational model. In Data Base Systems, Courant Computer Science Symposia 6. Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 65-98.

33. ჩოგოვაძე გ., სურგულაძე გ., ქაჩიბაია ვ. მონაცემთა ბაზების მართვის სისტემები. სტუ. „ტექნიკური უნივერსიტეტი“, თბ., 1988

34. Чоговадзе Г., Сургуладзе Г., Качибая В. Теория реляционных зависимостей и проектирование логической схемы баз данных. Монография. „Мецниереба“, Тбилиси, 1988

35. ჩოგოვაძე გ., სურგულაძე გ., შონია ო. მონაცემთა და ცოდნის ბაზების აგების საფუძლები. სტუ. „ტექნიკური უნივერსიტეტი“, თბ., 1996

36. Meyer-Wegener K. Multimediale Datenbanken- B. G. Teubner Stuttgart· Leipzig· Wiesbaden 2003.

37. Mattos N., Meyer-Wegener K., Mitschang B. A Grand Tour of Concepts for Object-oriented from a Database Point of View. Data & Knowledge Engineering 9 (1992/93). pp. 321-352.

38. Гогичаишвили Г., Сургуладзе Г. Разработка прикладного программного обеспечения интегрированных информационных систем управления на основе UMLю Georgian Electronic Scientific Journal, 2002, N1, 42-48.

39. Петкович Д. (2013). Microsoft SQL Server 2012. Руководство для начинающих: Пер. с англ. - СПб.: БХВ-Петербург.

40. Create Spatial Index (Transact-SQL). https://msdn.microsoft.com/en-us/libra-ry/bb93 4196.aspx

41. Spatial Indexes Overview. https://msdn.microsoft.com/en-us/en-ue/librarybb895 265.aspx#decompose

42. Eisenberg A. New Standard for Stored Procedures in SQL. ACM SIGMOD Record 25. 12, 1996

43. Fagin R.. A Normal Form for Relational Databases That Is Based on Domains and Keys. IBM Research Laboratory. ACM Transactions on Database Systems, Vol. 6, No. 3, September. 1981, pp. 387-415

44. Christodoulakis S. Multimedia Data Base Management: Application and Problems –A Position Paper. In S. Navathe (Hrsg.), Proc. ACM SIGMOD-85. Conf. on Management of Data (Austin, TX, May 1985). Bd14. No4, pp. 304-305

45. Сургуладзе Г. Построение структур реляционных баз данных в АСУ. Автореф. канд. диссерт. Эл-Технический Унив., Сн-т Петербург. 1980. 18 ст.

46. Чоговадзе Г., Сургуладзе Г., Качибая В. Теория реляционных зависимостей и проектирование логической схемы баз данных. Тб. Госуд. Унив., Тбилиси. 230 ст.

47. Woelk D., Kim W. Multimedia Information Management in an Object-Oriented Database System. In: Stocker P.M., Kent W. (Hrsg.), Proc.13th Int.Conf. on VLDB (Brighton, England, Sept. 1987). Los Altos, CA: Morgan Kaufmann Publ., 1987. pp.319-329

48. Cattell R.G., Barry D.K. The Object Database Standard: ODMG 3.0. San Francisco. Morgan Kaufmann Publ. 2000

49. Bancilhon F. Query Language for Object-Oriented Database Systems: Analysis and a Proposal. In: Härder (Hrsg.), Datenbanksysteme für Büro,Technik und Wissenschaft, Proc.GI/SI-Fachtagung (Zürich, März 1989), Informatik-Fachber, Nr.204, Berlin, Springer Verl, 1989, pp. 1-18.

50. Kim W., Garza J.F., Ballou N., Woelk D. Architecture of the ORION Next-Generation Database System. 1990. http://dl.acm.org /citation.cfm?id=627402

51. Rakow T.C., Neuhold E.J., Löhr M. Multimedia Database Systems. The Notions and the Issues. 1995. 31p. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.4279&rep=rep 1& type=pdf

52. Сургуладзе Г., Качибая В., Кортуа Т. О выборе приемлемых нормальных форм логической схемы реляционных БД. Сб.научн. тр. ГПИ., "Техническая кибернетика", N 10(267),Тб., 1983. ст. 47-51.

53. ჩოგოვაძე გ., გოგიჩაიშვილი გ., სურგულაძე გ., შეროზია თ., შონია ო. მართვის ავტომატიზებული სისტემების დაპროექტება და აგება: თეორიული და პრაქტიკული ინფორმატიკა. სტუ. „ტექნიკური უნივერსიტეტი", თბ., 2001, -740 გვ.

54. Сургуладзе Г., Реттер В., Шония О. Программа дальнейщей декомпозиции структуры функциональных зависимостей. Гос.фонд "Алгор.и Программ", N5(56), Рег.N П006378. 1983

55. Чоговадзе Г., Сургуладзе Г., Чачанидзе Г., Качибая В. Метод определения минимальной структуры функциональных зависимостей. "Al.I.Cusa", Iasi, TomXXX, s.I.a, Matematica, N5ю Analele stiintifice ale Universitatii. Roumania,1984

56. Wang C.P., Wedekind H. Segment synthesis in logical database design. IBM journ. Of Res. And Develop., vol.19, #1, 1975, pp.71-77

57. https://en.wikipedia.org/wiki/Graph_database

58. NoSQL For Dummies®. Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com Copyright © 2015, New Jersey

59. www.arangodb.com

60. https://docs.datastax.com/en/latest-dse/datastax_enterprise/graph/dseGraphAbout.html

61. MarkLogic. https://en.wikipedia.org/wiki/MarkLogic

62. OrientDB. https://en.wikipedia.org/wiki/OrientDB

63. StardogDB. https://en.wikipedia.org/wiki/Stardog

64. https://en.wikipedia.org/wiki/Web_Ontology_Language

65. სამხარაძე რ., გაჩეჩილაძე ლ. SQL სერვერი. სტუ. თბ., „ტექნ. უნივერსიტეტი". 2016.

66. SQL Server Security. https://msdn.microsoft.com/en-us/library/bb669074(v=vs.110).aspx

67. https://en.wikipedia.org/wiki/Data_Protection_API

68. Global Positioning System - GPS Satellite Systems. https://www.jmu.edu/cisr/research /sic/gps/satellite.htm

69. Dustin T. Duncan, Seann D. Regan, Donna Shelley, Kristen Day, Ryan R. Ruff, Maliyhah Al-Bayan, Brian Elbel. Application of global positioning system methods for the study of obesity and hypertension risk among low-income housing residents in New York City: a spatial feasibility study. Geospat Health. 2014 Nov; 9(1): 57–70. doi: 10.4081/gh.2014.6. 2014

70. სურგულაძე გ., პეტრიაშვილი ლ. ვიზუალური დაპროგრამება (C#.NET, Workflow Foundation.NET). ISBN 978-9941-20-880-5. სტუ, „IT კონსალტინგის სამეცნიერო ცენტრი", თბ., 2017. -232 გვ.

71. სურგულაძე გ., პეტრიაშვილი ლ. მონაცემთა მენეჯმენტის თანამედროვე ტექნოლოგიები (Oracle, MySQL, MongoDB,Hadoop). ISBN 978-9941-27-176-2. სტუ, „IT კონსალტინგის სამეცნიერო ცენტრი", თბ., 2017. -202 გვ.

72. ვედეკინდი ჰ., სურგულაძე გ., თოფურია ნ. განაწილებული ოფის-სისტემების მონაცემთა ბაზების დაპროექტება და რეალიზაცია UML-ტექნოლოგიით). ISBN 99940-57-17-0. სტუ, „ტექნიკური უნივერსიტეტი" თბ., 2006.

73. სურგულაძე გ., თოფურია ნ. მონაცემთა ბაზების მართვის სისტემები: ობიექტ-როლური მოდელირება (ORM/ERM, SQL Server). ISBN 99940-995-3-7. სტუ, „ტექნიკური უნივერსიტეტი" თბ., 2007

74. სურგულაძე გ., თოფურია ნ., ბაკურია კ., ლომიძე მ. (2014). საინფორმაციო სისტემის დაპროექტება ობიექტ-როლური მოდელირებისა და სერვის-ორიენტირებული არქიტექ-ტურის ბაზაზე. სტუ-ს შრ.კრ., „მას".N1(17). თბ., გვ. 32-45.

75. Солонин С.И. Метод гистограмм. Quality Management: Инструментарий улучшения качества. 7QC-tools. Екатеринбург: Изд-во УМЦ УПИ, 2014. https://study.urfu.ru/Aid/Publication/12495/1/Solonin_2.pdf

76. Toneva D. Assessment of Environmental Risks in the Black Sea. Sustainable Development, 4(25): 90-94· August 2015. https://www.researchgate.net/publication/284259415_ Assessment_of_ Environmental_Risks_in_the_Black_Sea

77. Surguladze G., Gavardashvili A., Topuria N. Determination of the Ecological Parameters of the Black Sea and Designing its Multimedia Base based on the Object-Role Modeling. XXVII internat. Scientific Conference „Problems of Decision Making under Uncertainties". 978-996-7166-39-7, Kiev-Batumi, 2016. pp. .65-68.

გადაეცა წარმოებას 15.12.2018. ხელმოწერილია დასაბეჭდად 20.12.2018. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 13. ტირაჟი 100 ეგზ.

სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრი"
(თბილისი, მ.კოსტავას 77)

ISBN 978-9941-8-0624-7