საქართველოს ტექნიკური უნივერსიტეტი

გიორგი კივილაძე

მაღალი წვდომადობის მონაცემთა საცავის დაპროექტება გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის („112“)

წარმოდგენილია დოქტორის აკადემიური ხარისხის მოსაპოვებლად

სადოქტორო პროგრამა „ინფორმატიკა“, შიფრი 0401

საქართველოს ტექნიკური უნივერსიტეტი

თბილისი, 0175, საქართველო

ივლისი, 2018 წელი

თბილისი
2018 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკურ უნივერსიტეტში
ინფორმატიკისა და მართვის სისტემების ფაკულტეტი
მართვის ავტომატიზებული სისტემების
(პროგრამული ინჟინერიის) დეპარტამენტი

ხელმძღვანელი: პროფ. გია სურგულაძე

რეცენზენტები: ----------

დაცვა შედგება ---------- წლის ”------” ----------, --------- საათზე
საქართველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის სხდომაზე, კორპუსი ----------, აუდიტორია ----------
მისამართი: 0175, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ს ბიბლიოთეკაში,
ხოლო ავტორეფერატისა - ფაკულტეტის ვებგვერდზე

სადისერტაციო საბჭოს მდივანი პროფ. თინათინ კაიშაური

საქართველოს ტექნიკური უნივერსიტეტი

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

ჩვენ, ქვემორე ხელისმომწერნი ვადასტურებთ, რომ გავეცანით **გიორგი კივილაძის** მიერ შესრულებულ სადოქტორო ნაშრომს დასახელებით: **“მაღალი წვდომადობის მონაცემთა საცავის დაპროექტება გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის („112“)“** და ვაძლევთ რეკომენდაციას საქარველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის სადისერტაციო საბჭოში მის განხილვას დოქტორის აკადემიური ხარისხის მოსაპოვებლად.

თარიღი :

ხელმძღვანელი: პროფ. გია სურგულაძე

რეცენზენტი:

რეცენზენტი:

საქართველოს ტექნიკური უნივერსიტეტი

2018

ავტორი: **გიორგი კივილაძე**

დასახელება: **“მაღალი წვდომადობის მონაცემთა საცავის დაპროექტება გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის („112“)“**

ფაკულტეტი : ინფორმატიკისა და მართვის სისტემების

ხარისხი: დოქტორი

სხდომა ჩატარდა:

ინდივიდუალური პიროვნებების ან ინსტიტუტების მიერ ზემომოყვანილი დასახელების ნაშომის გაცნობის მიზნით მოთხოვნის შემთხვევაში მისი არაკომერციული მიზნებით კოპირებისა და გავრცელების უფლება მინიჭებული აქვს საქართველოს ტექნიკურ უნივერსიტეტს.

ავტორის ხელმოწერა

ავტორი ინარჩუნებს დანარჩენ საგამომცემლო უფლებებს და არც მთლიანი ნაშრომის და არც მისი ცალკეული კომპონენტების გადაბეჭდვა ან სხვა რაიმე მეთოდით რეპროდუქცია დაუშვებელია ავტორის წერილობითი ნებართვის გარეშე.

ავტორი ირწმუნება, რომ ნაშრომში გამოყენებული საავტორო უფლებებით დაცული მასალებზე მიღებულია შესაბამისი ნებართვა (გარდა ის მცირე ზომის ციტატებისა, რომლებიც მოითხოვენ მხოლოდ სპეციფიურ მიმართებას ლიტერატურის ციტირებაში, როგორც ეს მიღებულია სამეცნიერო ნაშრომების შესრულებისას) და ყველა მათგანზე იღებს პასუხისმგებლობას.

რეზიუმე

სადისერტაციო ნაშრომში განხილულია მაღალი წვდომადობის მონაცემთა საცავის დაპროექტების ამოცანა გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის („112“). განხილული იქნა მონაცემთა შენახვა-დამუშავების არსებული მეთოდები და ტექნოლოგიები. კრიტიკული ანალიზის შედეგად, შერჩეულ იქნა არარელაციური მიდგომები, კერძოდ კი დოკუმენტ-ორიენტირებული მონაცემთა ბაზა (MongoDB).

წარმოდგენილია რელაციური და არარელაციური ბაზების მახასიათებლების შედარება, განსაზღვრულია იმ ობიექტების კლასი, რომლებისთვისაც ეფექტიანია არარელაციური მონაცემთა ბაზების გამოყენება. განიხილება განაწილებული ბაზების მთლიანობისა და ACID პრინციპები. დოკუმენტ–ორიენტირებული, არარელაციური მონაცემთა ბაზის MongoDB საფუძველზე განიხილება ბაზის მწარმოებლურობის ძირითადი მახასიათებლები.

შემუშავებულ იქნა კრიტიკული სისტემის არქიტექტურა, რომლითაც შესაძლებელი ხდება მონაცემთა მთლიანობის და ამოცანისთვის საკმარისი მაღალი წვდომადობის მიღწევა. სისტემაში გამოყენებულია ჰორიზონტალური ზრდის მოდელი, რაც ითვლება თანამედროვე და ინოვაციურ მიდგომად. აღნიშნული მოდელის უპირატესობაა რომ საჭიროებისამებრ, მარტივად არის შესაძლებელი რესურსის დამატება. ასევე, აღნიშნული მოდელის გამოყენებისას, ტექნიკის დაზიანებისგან გამოწვეული ხარვეზები არ არის კრიტიკული და ადვილად გამოსწორებადია. მაქსიმალური წარმადობისა და სტაბილურობის მისაღწევად გამოყენებულია რეპლიკაციისა და შარდინგის ტექნოლოგიები. MongoDB მონაცემთა ბაზას გარკვეულ ამოცანებში უკვე წარმატებით ვიყენებთ შსს სსიპ „112-ში“.

მონაცემთა საცავის დაპროექტებასთან ერთად მნიშვნელოვანია მონაცემების ანალიზი და დამუშავებაც. „112“-თან თანამშრომლობის შედეგად მიღებული მონაცემების ანალიზის საფუძველზე განსაზღვრულ იქნა საკვლევი მიმართულებები რეალური პრობლემების გადასაწყვეტად:

• **სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანა**. ამოცანის მიზანი- ბრიგადების განაწილების გაუმჯობესების შედეგად, გამოძახების საპასუხოდ ისეთი ბრიგადის არჩევა, რომელსაც შეუძლია, ყველაზე სწრაფად მივიდეს შემთხვევის ადგილზე. სადისერტაციო ნაშრომის ფარგლებში შემუშავებულ იქნა ხელოვნური ნეირონული ქსელის ზოგადი მოდელი და დაიგეგმა ამოცანის გადაწყვეტის ეტაპები. პროგრამირების ენა Python-ის საშუალებით შესრულდა extract, transform, load (ETL) და მონაცემთა წინასწარი დამუშავების (data preprocessing) პროცესები, რის შედეგადაც მიღებულ იქნა მანქანური დასწავლისთვის საჭირო სტრუქტურის მქონე მონაცემთა კრებული. შემდგომ ეტაპზე Python პროგრამირებისა და Weka ხელსაწყოს გამოყენებით გაანალიზებულ იქნა მიღებული მონაცემთა

ნაკრები. გამოვლინდა, წაიშალა და შესწორდა ანომალიური შემთხვევები, რაც გაუთვალისწინებლობის დროს უარყოფითად აისახებოდა მანქანური დასწავლის შედეგებზე. ამჟამად მიმდინარეობს მანქანური დასწავლის მეთოდების განხილვა და შედეგების შედარება. აღსანიშნავია, რომ სწორედ ამ პროექტით წარვადგინეთ საქართველოს „112“ ევროპის საგანგებო ნომრის ასოციაციის EENA (European Emergency Number Association) კონფერენციაზე, 2018 წლის 25-27 აპრილს, ლუბლიანაში (სლოვენია).

• სასწრაფო დახმარების გამოძახებების პრიორიტეტების განსაზღვრის ამოცანა. დისპეტჩერიზაციის მოდელირება გრიპის სეზონური ეპიდემიის დროს. ამოცანის მიზანი - შემოსული საქმეების რიგითობის განსაზღვრა გადაუდებლობის მიხედვით, რაც გააუმჯობესებს დისპეტჩერიზაციის ხარისხს. გამოცდილ ექიმებთან კონსულტაციის შემდეგ შეირჩა გრიპის შემთხვევების ობიექტური მახასიათებლები, სტანდარტული გართულებები და მათი გადაუდებლობის შეფასებები. გაანალიზდა სამედიცინო ინფორმაცია და შემოღებულ იქნა პარამეტრებისა და მათი წონების სპეციფიური სისტემა. გრიპის შემთხვევებზე შემოსული გამოძახებების ტიპიური ნიმუშების მიხედვით შედგა საცდელი მოდელი. ამ მოდელზე გამოიცადა სხვადასხვა ალგორითმი გამოძახების მომსახურების გადაუდებლობის ინდექსის გამოსათვლელად. სხვადასხვა ალგორითმის ობიექტურობის გამოსავლენად გამოთვლის შედეგები შედარდა გამოცდილი ექიმის მიერ საცდელი მოდელის შემთხვევების გადაუდებლობის ინდექსის შეფასებებთან. ალგორითმის შერჩევისას განსაკუთრებული ყურადღება ექცეოდა ყველაზე უფრო პრიორიტეტული შემთხვევების გამოვლენას. უპირატესობა ენიჭებოდა იმ ალგორითმს, რომელიც გამოავლენდა ექიმის მიერ მითითებულ მაღალი გადაუდებლობის ინდექსის მქონე ყველა შემთხვევას, ან ასეთი შემთხვევების უმეტესობას. შემუშავებული ალგორითმების გამოყენებამ გამოავლინა გადაუდებლობის მაქსიმალური ინდექსის მქონე ყველა შემთხვევა საცდელი კრებულიდან. ამგვარად, ჩატარებულმა კვლევამ აჩვენა, რომ შემოთავაზებული სისტემა შეიძლება გამოვიყენოთ გრიპის სეზონის დროს დისპეტჩერიზაციის პროცესის გასაუმჯობესებლად.

აღნიშნული პროექტების შემდგომი განვითარება და რეალურ გარემოში დანერგვა დაეხმარება „112“-ის დისპეტჩერს გადაწყვეტილების მიღებაში და შეამცირებს რეაგირების დროს.

სადოქტორო კვლევა დაფინანსდა შოთა რუსთაველის ეროვნული სამეცნიერო ფონდის დოქტორანტურის კვლევითი გრანტის ფარგლებში № PhDF2016_219, რამაც საშუალება მომცა, გამომეყენებინა ფინეთისა და გერმანიის უნივერსიტეტებიდან მიღებული მოწვევები. აღნიშნული ნაშრომით დაინტერესება და ერთობლივი კვლევა შერჩეული თემის აქტუალობითა და ინფორმატიკის მკვლევართა წრეში მისი განსაკუთრებული პოპულარობით აიხსნება, რაც კიდევ უფრო მმატებს სტიმულს, კვლევა დოქტორანტურის დასრულების შემდგომაც აქტიურად განვაგრძო.

Abstract

Dissertation thesis represents the problem of planning high availability data center for Emergency Response Center ("112"). The data saving and processing methods and technologies are discussed. As a result of critical analysis non-relational approaches are chosen, particularly a document-oriented database (MongoDB).

Specifications of relational and non-relational databases are compared. The class of objects is defined, for which the application of non-relational databases is effective. Attention is focused on the issue of integrity of the distributed bases, the ACID principles are discussed. On the base of document-oriented non-relational database MongoDB main characteristics of base productivity are discussed.

The architecture of the critical system was worked out. That makes possible to achieve data integrity, horizontal scaling and sufficiently high availability for the problem. The MongoDB database is already successfully used by the Ministry of Internal Affairs of Georgia at LEPL "112".

Besides projecting of data center, it is important to perform the analyses and processing of the data. Based on the analysis of the data obtained from co-operation with "112", the study directions have been determined to solve real problems:

- **EMS brigade selecting.** The aim of the task is improvement of brigade distribution that results in selecting the ambulance car that is most proper for the fastest responding. The general model of Artificial Neural Network was worked out and the stages of problem solving were planned. Programming language Python was used to perform extract, transform, load (ETL) and data preprocessing processes. As a result the dataset structured for machine learning was received. On the next stage the received dataset was analyzed by means of Python programming and Weka toolbox. The anomaly cases that reflected negatively on machine learning results were detected and corrected or eliminated. Currently the research is going on: the machine learning methods are discussed and the results are compared. It is noteworthy that this project was presented to the European Emergency Number Association (EENA) conference on April 25-27, 2018 in Ljubljana (Slovenia).
- **EMS case priority defining. Modeling EMS dispatching in flu-seasons.** The aim of the task is defining priority of cases according to their emergency level that improves EMS service. Highly experienced medics were consulted for choosing the factors describing the state of a patient objectively. The emergency level of each case was estimated by an experienced doctor. Having in mind the possible clinical complications of influenza cases, a special system for processing medical information is worked out. The special parameters are chosen and their proper weights are defined. An original system is applied to process case-card information and supply dispatchers with its result – emergency level of each influenza case, as well as the priority line of cases of the current set. The model set was formed using typical EMS registered influenza cases gathered from the usual case-cards filled by operators of Georgian Emergency and Operative Response Center. The special system of converting medical information into the set of quantitative parameters is presented

in the thesis as well as the algorithms developed to compute the emergency index of influenza cases. The research presented in this work shows that the accuracy of revealing the most urgent cases by means of the applied algorithms is the same as that of an experienced medic. The applied system makes it possible to reveal all cases estimated as most urgent by the medic.

The automated system of EMS brigade dispatching should be considered as a real help for dispatchers in flu-seasons.

This research was supported by Shota Rustaveli National Science Foundation (SRNSF) [PhDF2016_219]. That gave me the possibility to hold 2 research visits in Finland (University of Tampere) and Germany (Friedrich-Alexander University Erlangen-Nürnberg) in 2017.

The fact that foreign colleagues are interested in this work can be explained by the acuteness of the selected topic and its popularity among the informatics researchers. The first results of the research show that the project should be continued. I am ready to work at it after finishing the dissertation.

შინაარსი

შესავალი.....17

1 თავი. ლიტერატურის მიმოხილვა. გამოყენებული ტექნოლოგიების კრიტიკული ანალიზი.....22

1.1. მონაცემების შენახვა-დამუშავების ტექნოლოგიების ევოლუცია.....22

1.1.1. მონაცემთა ბაზების დანიშნულება.....22

1.1.2. მონაცემთა ბაზების განვითარების ეტაპები და შედარებითი ანალიზი.....22

1.2. მონაცემთა რელაციური და არარელაციური ბაზების შედარება 27

1.2.1. მონაცემთა დამუშავების ტექნოლოგიების მიმოხილვა 28

1.2.2. რა ხარვეზები აქვს მონაცემთა რელაციური ბაზების ტექნოლოგიას?.....31

1.2.3. ვერტიკალური და ჰორიზონტალური სკალები 33

1.2.4. როდის ჯობია არარელაციური მონაცემთა ბაზების გამოყენება?.....33

1.3. მონაცემთა შენახვის ტიპები 35

1.3.1. გასაღები-მნიშვნელობა შენახვის ტიპის მონაცემთა ბაზა. 36

1.3.2. დოკუმენტ-ორიენტირებული მონაცემთა ბაზა 39

1.3.3. მონაცემთა შენახვის დოკუმენტზე ორიენტირებული და რელაციური მოდელების შედარება 41

1.3.4. სვეტებზე ორიენტირებული მონაცემთა ბაზა.....42

1.3.5. გრაფებზე ორიენტირებული მონაცემთა ბაზა 42

1.4. Hadoop – „დიდ მონაცემთა“ ტექნოლოგია 43

1.5. მანქანური დასწავლა.....45

1.6. ამოცანის დასმა.....50

1.7. პირველი თავის დასკვნა.....52

2 თავი. მონაცემთა საცავის დაპროექტება და სისტემის ინფრასტრუქტურა 53

2.1. CAP სამკუთხედის თეორემა.....53

2.2. მთლიანობის მოდელები CAP თეორემაში.....56

2.2.1. AP სისტემები - ძლიერი მთლიანობის მოდელი 57

2.2.2. AP სისტემები - სუსტი მთლიანობის მოდელი 57

2.3. ტრანზაქციის იზოლირების დონეები 58
2.4. მონაცემების დუბლირება – replication 60
2.5. განაწილებული გარემო – sharding 62
2.6. სერვერული უზრუნველყოფის გამართვა დუბლირებული მონაცემებისა და განაწილებული გარემოს გამოყენებით 64
2.7. მონაცემების ასახვა და მათი დამუშავება 70
2.7.1. მონაცემთა ბაზის აგება MongoDB გარემოში 70
2.7.2. RoboMongo პლატფორმა 71
2.7.3. სხვადასხვა ოპერაციების მაგალითები 75
2.8. მეორე თავის დასკვნა 79
3 თავი. ექსპერიმენტული ნაწილი: სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანა 80
3.1. საკვლევი სფეროს აქტუალობა, კვლევის სიახლე და პრობლემის ფორმულირება 80
3.2. დისპეტჩერიზაციის პროცესი „112“-ში 81
3.3. დისპეტჩერიზაციის კომპიუტერული სისტემა ბრიგადების შესარჩევად 83
3.4. მონაცემთა კრებული (Dataset) 85
3.4.1. მონაცემების შეგროვება (Data-gathering) 85
3.4.2. მონაცემების წინასწარი დამუშავება (Data preprocessing) .. 86
3.5. მონაცემთა ანალიზი და ვიზუალიზაცია 92
3.6. სასწრაფო დახმარების ბრიგადის მიერ დახარჯული დროის შეფასება 94
3.6.1. კლასიფიკაციის ამოცანა. პასუხი - ჰოსპიტალიზაციის ალბათობა 95
3.6.2. რეგრესიის ამოცანა. პასუხი - სავარაუდო დასახარჯი დრო 96
3.7. მესამე თავის დასკვნა 97
4 თავი. ექსპერიმენტული ნაწილი: სასწრაფო დახმარების გამოძახებების პრიორიტეტების განსაზღვრის ამოცანა. დისპეტჩერიზაციის მოდელირება გრიპის სეზონური ეპიდემიის დროს. 99
4.1. დისპეტჩერიზაციის ამოცანის სირთულე გრიპის სეზონური ეპიდემიის დროს 99
4.2. დისპეტჩერიზაციის მოდელირება 99
4.3. დისპეტჩერიზაციის მოდელირების შედეგების ანალიზი 109

4.4. დისპეტჩერიზაციის ალტერნატიული მოდელი 111
4.5. გამოძახებათა პრიორიტეტიზაციის ზოგადი პრინციპები 115
4.6. მეოთხე თავის დასკვნა 119
დასკვნა 120
გამოყენებული ლიტერატურა 122

ცხრილების ნუსხა

ცხრ. 1. ფაილური მონაცემთა ბაზები 23

ცხრ. 2. იერარქიული მოდელი..... 23

ცხრ. 3. ქსელური მოდელი..... 24

ცხრ. 4. მონაცემთა რელაციური და არარელაციური ბაზების ამოცანათა კლასების შედარება..... 34

ცხრ. 5. გასაღები-მნიშვნელობა (Key-Value) ტიპის მონაცემთა ბაზა..... 36

ცხრ. 6. გასაღები-მნიშვნელობა შენახვის ტიპის მაგალითი..... 38

ცხრ. 7. დოკუმენტ-ორიენტირებული მონაცემთა ბაზა 40

ცხრ. 8. სვეტებზე ორიენტირებული მონაცემთა ბაზა..... 42

ცხრ. 10. რისკ ფაქტორი r_a სხვადასხვა ასაკობრივი ჯგუფისათვის..... 101

ცხრ. 11. რისკ ფაქტორი r_c განსხვავებული ტემპერატურის შემთხვევაში... 101

ცხრ. 12. რისკ ფაქტორის სიდიდე r_s კლინიკური სიმპტომებისათვის..... 102

ცხრ. 13. რისკ ფაქტორი r_d გრიპის თანმხლები დაავადებებისა და განსაკუთრებული მდგომარეობის დროს..... 104

ცხრ. 14. გრიპის ტიპიური გამოძახებებით შედგენილი საცდელი მოდელი 108

ცხრ. 15. ალგორითმების გამოყენებით მიღებული პრიორიტეტულობის რიგი 110

ცხრ. 16. გადაუდებლობის ინდექსის გამოთვლილი r_1 და r_2 სიდიდეები გადაუდებლობის h/m/l დონეები ექიმის შეფასების მიხედვით 113

ნახაზების ნუსხა

ნახ. 1. რელაციური მოდელის სქემა 25

ნახ. 2. ტერმინებისა და ცნებების შედარება მონაცემთა ბაზებში 28

ნახ. 3. ფუნქციონალისა და შესაძლებლობების შედარება მონაცემთა ბაზებში 29

ნახ. 4. ვერტიკალური და ჰორიზონტალური სკალირება 33

ნახ. 5. მონაცემთა შენახვის დოკუმენტზე ორიენტირებული და რელაციური მოდელების სტრუქტურული შედარება 41

ნახ. 6. Hadoop ეკოსისტემა 44

ნახ. 7. ხელოვნური ინტელექტის სფეროები 46

ნახ. 8. კონტროლირებადი, არაკონტროლირებადი და გაძლიერებული დასწავლა 46

ნახ. 9. მანქანური დასწავლის გამოყენების სფეროები 48

ნახ. 10. CAP თეორემა - მონაცემთა ბაზების მხარდაჭერის მიხედვით 53

ნახ. 11. CAP თეორემა 54

ნახ. 12. რეპლიკაციის პროცესი 61

ნახ. 13. ინფორმაციის გადანაწილება Sharding ტექნოლოგიით 63

ნახ. 14. Sharding: დოკუმენტებისა და ბლოკების ორგანიზება 63

ნახ. 15. სერვერული უზრუნველყოფა დუბლირებული მონაცემებისა და განაწილებული გარემოს (replication + sharding) გამოყენებით 66

ნახ. 16. კავშირის გახსნა MongoDB სისტემასთან 71

ნახ. 17. Robomongo - ტერმინალთან დაკავშირება 72

ნახ. 18. აქტიური მონაცემთა ბაზის არჩევა 72

ნახ. 19. მონაცემების დამატება movies კოლექციაში 73

ნახ. 20. RoboMongo intellisense 73

ნახ. 21. მონაცემების ვიზუალიზაცია ხისებრი სტრუქტურით 74

ნახ. 22. მონაცემების ვიზუალიზაცია ცხრილის სტრუქტურით 74

ნახ. 23. მონაცემების ვიზუალიზაცია JSON სტრუქტურით 75

ნახ. 24. სასურველი ველების პროექცია 77

ნახ. 25. 112-ში საქმის მართვის ძირითადი გეგმა 81

ნახ. 26. ოპერატორის მიერ შევსებული ბარათის მიმთითებელი ნაწილი 82

ნახ. 27. პაციენტის ასაკი 85

ნახ. 28. სასწრაფოს ბრიგადის მიერ პაციენტთან გატარებული დროისა, წინა ჰოსპიტალიზაციის შემთხვევების რაოდენობის და ჰოსპიტალიზაციის მოხდენის ურთიერთ დამოკიდებულება 92
ნახ. 29. გამოძახებათა რაოდენობის კავშირი სხვადასხვა პარამეტრთან 93
ნახ. 30. კავშირი ინციდენტის ტიპსა და სასწრაფოს ბრიგადის გათავისუფლებას შორის..... 94
ნახ. 31. პაციენტის ჰოსპიტალიზაციის ალბათობის გამოთვლა 95
ნახ. 32. სასწრაფო დახმარების ბრიგადის მიერ პაციენტის მისამართზე სავარაუდოდ დასახარჯი დროის შეფასება 96
ნახ. 33. გრიპის შემთხვევების მახასიათებლები..... 100
ნახ. 34. გრიპის შემთხვევათა პრიორიტეტულობის დადგენის სქემა 106
ნახ. 35. დისპეტჩერიზაციის ალტერნატიული მოდელი..... 112

დისერტაციაში გამოყენებული აბრევიატურები

ACID	Atomicity, Consistency, Isolation, Durability
ETL	Extract, Transform, Load
EENA	European Emergency Number Association
SQL	Structured Query Language
NoSQL	Not Only SQL
AI	Artificial Intelligence
ML	Machine Learning
DBMS	Database Management System
RDBMS	Relational Database Management System
IMS	Information Management System
CODASYL	Conference on Data Systems Languages
JSON	JavaScript Object Notation
OLTP	Online Transaction Processing
OLAP	On-line Analytical Processing
ERM	Entity-Relationship Model
API	Application Programming Interface
CMS	Content Management System
URI	Unified Resource Identifier
RDF	Resource Description Framework
OWL	Web Ontology Language
XML	Extensible Markup Language
IOT	Internet Of Things
GPS	global positioning system
HDFS	Hadoop Distributed File System
YARN	Yet Another Resource Negotiator
CAP	Consistency, Availability, Partition tolerance

მადლიერება

სადოქტორო კვლევა დაფინანსდა შოთა რუსთაველის ეროვნული სამეცნიერო ფონდის დოქტორანტურის კვლევითი გრანტის ფარგლებში № PhDF2016_219. დიდად მადლობელი ვარ ამ ფონდისა.

განსაკუთრებულად მადლიერი ვარ ჩემი ხელმძღვანელის, ბატონი გია სურგულაძის, რომელიც დისერტაციაზე მუშაობის პერიოდში დაუზარებლად მიწევდა კონსულტაციებს და მუდმივად გვერდით მედგა. ასევე, მადლიერი ვარ საქართველოს ტექნიკური უნივერსიტეტისა, კერძოდ კი მართვის ავტომატიზებული სისტემების დეპარტამენტის კოლეგებისა.

საკვლევი მიმართულებების განსაზღვრისა და კვლევის განხორციელების საშუალება მომეცა, რადგან თითქმის 5 წელია ვმუშაობ გადაუდებელი დახმარების ოპერატიული მართვის ცენტრში („112“) მონაცემთა ბაზებისა და ტელეფონიის ადმინისტრატორის პოზიციაზე. დიდად მადლობელი ვარ „112“-ის ხელმძღვანელობისა და თანამშრომლებისა მხარდაჭერისა და დახმარებისათვის.

2017 წელს განვახორციელე 2 კვლევითი ვიზიტი ფინეთში (ტამპერეს უნივერსიტეტი) და გერმანიაში (ერლანგენ-ნიურნბერგის უნივერსიტეტში). დიდად მადლობელი ვარ ჩემი კონსულტანტებისა ამ უნივერსიტეტებში: პროფესორები იურკი ნუმენმაა და კლაუს მეიერ-ვეგენერი. მადლობელი ვარ ფინეთის „112“-ის პროგრამული არქიტექტორის აპო კოსკის, საინტერესო სამეცნიერო დისკუსიებისა და სასარგებლო რჩევებისათვის.

რამდენადაც დისერტაციის ექსპერიმენტული ნაწილი ეხებოდა სამედიცინო პრობლემებს, კვლევისათვის დამჭირდა კონსულტაციების მიღება გამოცდილი ექიმებისაგან. დიდად მადლობელი ვარ მათი: პროფესორების - ფატი გაბუნიას, თამარ სარალიძის და გივი სარალიძის.

შესავალი

კაცობრიობის უმთავრესი მონაპოვარი ინფორმაციაა. მოაზროვნეები ცდილობდნენ, რომ მნიშვნელოვანი მიღწევები საიმედოდ შეენახათ, იხვეწებოდა ინფორმაციის კომპაქტურად ჩაწერისა და დაცვის მეთოდები, ინფორმაციის რაოდენობის ზრდასთან ერთად იქმნებოდა უფრო და უფრო მოქნილი სისტემები.

ისეთ მუდმივად მზარდ მონაცემთა საცავშიც კი, როგორიც აქვს გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-ს, ფუნქციონირების 6 წელიწადში ჯერ არ დაგროვილა განსაკუთრებით დიდი რაოდენობის მონაცემები, მაგრამ მაინც, მონაცემთა საცავის დაპროექტებისას დაშვებული ხარვეზები მონაცემების ზრდასთან ერთად თანდათან უფრო მეტად იჩენს თავს. რა პრობლემები აღმოჩნდება კიდევ 6 წლის შემდეგ? ამის წინასწარ განსაზღვრას სჭირდება განსაკუთრებით დიდ და კრიტიკულ სისტემებთან მუშაობის გამოცდილება.

კვლევის მეთოდოლოგიად არჩეულ იქნა მსგავსი ამოცანებისთვის უკვე შემუშავებული გადაწყვეტილებების შესწავლა და მათი ანალიზის შედეგად დასმული ამოცანისთვის სპეციფიური გადაწყვეტილების შემუშავება.

სადოქტორო ნაშრომის თემაა “მაღალი წვდომადობის მონაცემთა საცავის დაპროექტება გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის „112“. სადოქტორო ნაშრომის ფარგლებში ჩატარებულ იქნა სიღრმისეული კვლევა, მოძიებულ და გარჩეულ იქნა საკვლევ პრობლემატიკასთან დაკავშირებული სამეცნიერო ლიტერატურა და არსებული გადაწყვეტილებები. დაგროვილი გამოცდილება გამოყენებულ იქნა გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის „112“ მონაცემთა საცავის დაპროექტებასა და რეალური პრობლემების გადაწყვეტილების შეთავაზებაში.

საგანგებო სიტუაციების მართვის ცენტრ „112“–ში მონაცემთა ბაზის ადმინისტრატორის პოზიციაზე 5 წლიანი მუშაობის შედეგად

მნიშვნელოვანი გამოცდილება დამიგროვდა რელაციურ მონაცემთა ბაზების კუთხით. ამასთან, ბოლო პერიოდში დავინტერესდი Hadoop ტექნოლოგიითა და მონაცემთა ბაზების ახალი მიმდინარეობით, არარელაციური, NoSQL ტექნოლოგიით, რასაც დავუთმე სამაგისტრო ნაშრომი და ასევე, წარმატებით გავიარე MongoDB-ს ოფიციალური საწყისი და სიღრმისეული კურსები:

- M102 - MongoDB for DBAs – მარტი 2015
- M202 - MongoDB Advanced Deployment and Operations – მაისი 2015
- M310 - MongoDB Security – მარტი 2017
- M312 - MongoDB Security - მარტი 2017

აღსანიშნავია, რომ კვლევის მიმდინარეობაში მეტად მნიშვნელოვანი ეტაპი იყო კვლევითი ვიზიტები ფინეთსა და გერმანიაში. ასევე, კვლევისთვის დამჭირდა სხვადასხვა ტექნიკური ლიტერატურისა და კურსის მოძიება-შესყიდვა და გარჩევა.

დღევანდელი პრობლემატიკა და შესაბამისად, მათი გადაწყვეტის ტექნოლოგიები იმაზე ბევრად სწრაფად მიიწევს წინ, ვიდრე ჩვენ ვვითარდებით. ეს ბუნებრივიცაა, საქართველოზე ბევრად დიდ და ტექნოლოგიურად განვითარებულ სახელმწიფოებს რამდენიმე ათეული წლით ადრე დაუდგათ დღის წესრიგში განსაკუთრებით დიდ და კრიტიკულ მონაცემებთან მუშაობის პრობლემა. ჩვენ შემთხვევაში კი ეს პრობლემა მხოლოდ ბოლო ათწლეულში გახდა აქტუალური.

კვლევაში ძირითადი აქცენტი გაკეთებულია უახლეს ტექნოლოგიებზე, მათ კრიტიკულ ანალიზსა და გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-თან დაკავშირებული პრობლემების გადაწყვეტაზე.

დისერტაცია შედგება ოთხი თავისა და დასკვნისგან. ქვემოთ გადმოცემულია მოკლედ თითოეულ თავში განხილული საკითხები, ლიტერატურული მომიხილვის, დასმული ამოცანების გადაწყვეტის თეორიული და პრაქტიკული მხარეების აღწერით და ბოლოს, სისტემის ექსპერიმენტული რეალიზაციით.

პირველ თავში განხილულია მონაცემთა ბაზების განვითარების ეტაპები და მოცემულია მონაცემთა რელაციური ბაზების გამოყენების ასპექტები. შედარებულია რელაციური და არარელაციური ბაზების მახასიათებლები, განსაზღვრულია იმ ამოცანათა კლასი, რომლებისთვისაც ეფექტიანია არარელაციური მონაცემთა ბაზების გამოყენება. ფართოდ გავრცელებული რელაციური მონაცემთა ბაზების გვერდით მოცემულია დღეისათვის ერთ-ერთი აქტუალური ტექნოლოგია - NoSQL, რომელიც ხშირად ახალი ტიპის არარელაციურ მონაცემთა ბაზების ოჯახის სახით განიხილება. მართვის საინფორმაციო სისტემებისთვის შედარებულია მონაცემთა რელაციური და არარელაციური ბაზების მახასიათებლები. კონსისტენტურობის მოდელების ანალიზთან ერთად განხილულია Apache Hadoop ეკოსისტემაც, რაც მნიშვნელოვნად ზრდის მონაცემთა შენახვისა და ანალიზის წარმადობასა და შესაძლებლობებს. მიმოხილულია მანქანური დასწავლისა და მისი ქვე-მიმართულებების ტექნოლოგიები. ნაშრომში აღწერილი ყველა პროგრამული უზრუნველყოფა პროგრამულად განხორციელებულია Python ენაზე.

წარმოდგენილ ტექნოლოგიებს აქტიურად ვიყენებთ საქართველოს გადაუდებელი დახმარების ოპერატიული მართვის ცენტრში „112“.

მეორე თავში განხილულია მონაცემთა განაწილებული ბაზების მთლიანობის საკითხი, ACID პრინციპები და კონსისტენტური მოდელების ყველაზე გავრცელებული ვარიანტები მონაცემთა ბაზებისათვის. განსაზღვრულია ამოცანები და სფეროები კონსისტენტურობის სხვადასხვა დონეთა ეფექტური გამოყენებისათვის.

მონაცემთა მთლიანობის, ჰორიზონტალური ზრდისა და ამოცანისთვის საკმარისი მაღალი წვდომადობის მისაღწევად შემუშავებულ იქნა კრიტიკული სისტემის არქიტექტურა.

აღწერილია სერვერული უზრუნველყოფის გამართვის ეტაპები. მოყვანილია მონაცემთა ბაზის შექმნისა და მასზე მუშაობის მაგალითები MongoDB სისტემისთვის.

მესამე თავში ნაჩვენებია, რომ დიდი რაოდენობით დაგროვებულმა ინფორმაციამ ხელოვნური ინტელექტის (Artificial Intelligence) მიმართულებით განვითარების საშუალება მოგვცა. კონკრეტულად, განხილულია სასწრაფო სამედიცინო დახმარების ბრიგადის ამორჩევის ამოცანით ბრიგადების მართვის გაუმჯობესება; განხილულია სასწრაფო სამედიცინო დახმარების ბრიგადის გათავისუფლებისა და ახალ მისამართზე მისვლის სავარაუდო დროის არცოდნით გამოწვეული სირთულეები.

შემუშავებულ იქნა ხელოვნური ნეირონული ქსელის ზოგადი მოდელი და დაიგეგმა ამოცანის გადაწყვეტის ეტაპები. დაპროგრამების ენა Python-ის საშუალებით შესრულდა extract, transform, load (ETL) და მონაცემთა წინასწარი დამუშავების (data preprocessing) ამოცანები, რის შედეგადაც მიღებულ იქნა მანქანური დასწავლისთვის საჭირო სტრუქტურის მქონე მონაცემთა კრებული, რაც შემდგომ ეტაპზე გაანალიზებულ იქნა Python პროგრამირებისა და Weka ხელსაწყოს გამოყენებით.

ნაშრომის მესამე თავში შეჯამებულია მანქანური დასწავლის მიმართულებით დაწყებული კვლევის შედეგები და ჩამოყალიბებულია კვლევის გაგრძელების სამომავლო მიზნები.

მეოთხე თავში განხილულია სასწრაფო დახმარების გამოძახებების გადაუდებლობის ხარისხის განსაზღვრის ამოცანა. ყურადღება გამახვილებულია გრიპის სეზონურ ეპიდემიებზე, როდესაც გამოძახებები არსებულ ბრიგადებთან შედარებით ბევრად მეტია და ხშირად ერთი და იმავე პრიორიტეტის („დაბალი“, „საშუალო“ ან „მაღალი“) მქონე 50-მდე „ჩამოკიდებული“ საქმე ელოდება თავისუფალ ბრიგადას.

შექმნილია კომპიუტერული სისტემის მოდელი, რომელიც ობიექტური გამოთვლების საფუძველზე საშუალებას გვაძლევს, გამოვლინდეს ყველაზე მაღალი კატეგორიის გადაუდებლობის მქონე შემთხვევები (გამოძახებები).

შესასრულებელი სამუშაოს მასშტაბურობიდან გამომდინარე, „112“-თან თანამშრომლობა დოქტორანტურის დასრულების შემდეგაც გაგრძელდება. ასევე, გაგრძელდება თანამშრომლობა საქართველოს ტექნიკურ

უნივერსიტეტთანაც, რადგან მსგავსი აქტივობები მნიშვნელოვნად დაეხმარება Big Data ტექნოლოგიებისა და მანქანური დასწავლის მიმართულებების განვითარებას უნივერსიტეტსა და მთლიანად, ქვეყანაში.

დისერტაციის ბოლოს მოცემულია დასკვნები და გამოყენებული ლიტერატურის სია. სადისერტაციო თემაზე მუშაობის დროს ავტორის მიერ გამოქვეყნებულია 7 სამეცნიერო ნაშრომი.

1 თავი. ლიტერატურის მიმოხილვა. გამოყენებული ტექნოლოგიების კრიტიკული ანალიზი.

1.1. მონაცემების შენახვა-დამუშავების ტექნოლოგიების ევოლუცია

1.1.1. მონაცემთა ბაზების დანიშნულება

Encyclopaedia Britannica-ს განმარტების მიხედვით **მონაცემთა ბაზა** არის მონაცემთა ან ინფორმაციის (დამუშავებული მონაცემები) შემნახველი სისტემა, რომელიც ორიენტირებულია სწრაფ ძებნასა და დაბრუნებაზე. მონაცემთა ბაზა შესაძლებელს ხდის მონაცემთა შენახვას, მომხმარებლისთვის სათანადო ფორმით მიწოდებას, ცვლილებასა და წაშლას შესაბამისი დაცული ოპერაციების საშუალებით. მონაცემთა ბაზის სამართავი პროგრამა (**DBMS - Database Management System**) უზრუნველყოფს ინფორმაციის მიღება-დამუშავებას შესაბამისი მოთხოვნების საფუძველზე [1].

1.1.2. მონაცემთა ბაზების განვითარების ეტაპები და შედარებითი ანალიზი

ციფრულ მონაცემთა ბაზების განვითარება 1960–იანი წლებიდან იწყება და რამდენიმე სხვადასხვა ეტაპი გაიარა [2,3]:

- ჯერ კიდევ **1725** წლიდან გამოჩნდა პერფობარათების პირველი ფორმა;
- **1890** წლიდან 1960-იან წლებამდე გამოიყენებოდა IBM-ის დამაარსებლის ჰერმან ჰოლერიცის (Herman Hollerith) მიერ შემუშავებული მექანიკური გამომთვლელი მანქანა, რომლის საშუალებითაც პერფობარათებიდან შესაძლებელი იყო მილიონობით მონაცემის სტატისტიკის დამუშავება.
- **1968** წ. – შეიქმნა მონაცემთა ფაილური ბაზების ტექნოლოგია (File-Based)

მონაცემთა ფაილურ ბაზებში გამოიყენებოდა ერთგვაროვანი ფაილები (Flat Files), მკაცრად განსაზღვრული სტრუქტურის გარეშე.

ქვემოთ მოცემულია მონაცემთა ფაილური ბაზების დადებითი და უაროფითი მახასიათებლები (ცხრ.1).

ცხრ. 1. ფაილური მონაცემთა ბაზები

დადებითი	უარყოფითი
წვდომის სხვადასხვა მეთოდები.	პროგრამული თვალსაზრისით რთულად დასამუშავებელია.
	პირდაპირი გზით შეუძლებელია სხვადასხვა პროგრამიდან ერთსა და იმავე ფაილზე მოთხოვნის განცალკევება.
	მონაცემების დუბლირება – ერთი და იგივე მონაცემები მეორდება სხვადასხვა ფაილებში.
	დაბალი დონის უსაფრთხოება.

• **1968-1980** წლები – მონაცემთა იერარქიული ბაზების ტექნოლოგიები [2];

IBM-მა შეიმუშავა მონაცემების იერარქიული მოდელი IMS (Information Management System). ამ მოდელში ფაილები ერთმანეთს უკავშირდებოდა მშობელი/შვილი სახით. მე-2 ცხრილში მოცემულია აღნიშნული მოდელის დახასიათება.

ცხრ. 2. იერერქიული მოდელი

დადებითი	უარყოფითი
ეფექტური ძიება.	პროგრამული თვალსაზრისით რთულად დასამუშავებელია.
ნაკლებად დუბლირებული მონაცემები.	რთულდება M:N კავშირის ჩამოყალიბება და მონაცემების განახლება.
დაცულია მონაცემთა ბაზის უსაფრთხოება და მთლიანობა.	არ აქვს სტრუქტურული დამოუკიდებლობა.

- 1971 წლიდან **CODASYL** ჯგუფის მიერ (Conference on Data Systems Languages) სტანდარტად ჩამოყალიბდა მონაცემთა ბაზის ქსელური მოდელი **IDS** (Integrated Data Store) [2].

ქსელურ მოდელში ფაილები ერთმანეთთან მფლობელი-წევრი ტიპის მიმართებით არის დაკავშირებული, სადაც თითოეულ წევრს შეიძლება ჰყავდეს ერთზე მეტი მფლობელი (ცხრ.3).

ცხრ. 3. ქსელური მოდელი

დადებითი	უარყოფითი
შესაძლებელია უფრო მეტი კავშირის ტიპის ჩამოყალიბება	სისტემა კომპლექსურია და რთულდება დიზაინის შემუშავება
გამარტივებული წვდომა მონაცემებზე	რადგან მონაცემებზე წვდომის მეთოდი ცვალებადია, არ აქვს სტრუქტურული დამოუკიდებლობა.
მონაცემების მთლიანობა და დამოუკიდებლობა	

- **1970 წლიდან – დღემდე:** მონაცემთა რელაციური ბაზების ერა [1,3,4,5];

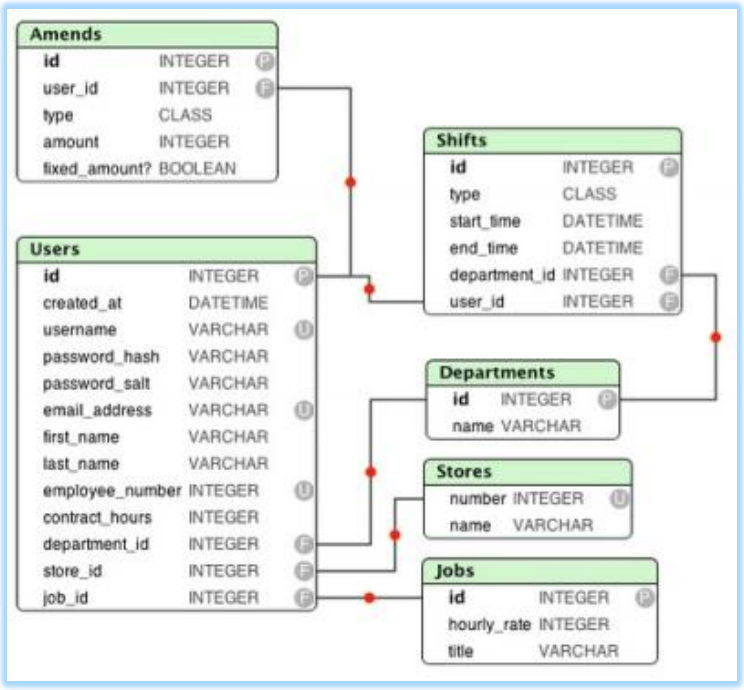
წარმოშობით ინგლისელმა და ამერიკის შეერთებულ შტატებში მცხოვრებმა, IBM-ის თანამშრომელმა ედგარ კოდმა **1970** წელს შეიმუშავა მონაცემთა ბაზების რელაციური მოდელი **RDBMS** (Relational Database Management System). რელაციური მოდელი მკაცრად ეფუძნება მათემატიკურ თეორიებს: სიმრავლეთა თეორიასა და პრედიკატების აღრიცხვის ლოგიკას. იმ დროისათვის ამ მოდელს არარეალურ მიდგომად მიიჩნევდნენ, რადგანაც მონაცემების შენახვა მოდელში უნდა მომხდარიყო ცხრილების სახით და ყოველ ცხრილში უნდა განთავსებულიყო ინფორმაცია მხოლოდ ერთი ლოგიკური ობიექტის შესახებ. ეს ობიექტები კი ერთმანეთს გარე მაჩვენებლების ნაცვლად დაუკავშირდებოდა ამ ცხრილებშივე განთავსებული ინფორმაციის საშუალებით [2,5].

მონაცემთა რელაციური ბაზების სისტემები გვაძლევს ობიექტებზე ორიენტირებული პროგრამირების მეთოდებთან მუშაობის შესაძლებლობას.

რელაციური მოდელი შეგვიძლია წარმოვადგინოთ ორი ძირითადი ტერმინოლოგიით:

○**Instance** - სვეტებისა და სტრიქონების შემცველი ცხრილი;

○**Schema** - რელაციის სახელის, თითოეული სვეტის სახელისა და ტიპის განმსაზღვრელი სტრუქტურა (ნახ.1).



ნახ. 1. რელაციური მოდელის სქემა

• **2000**–ანი წლებიდან დღემდე – **NoSQL :**

ტექნოლოგიური განვითარებისა თვისობრივად განსხვავებული ამოცანების გამოჩენამ გამოიწვია ისეთი მოთხოვნები, რომლებსაც რელაციური მონაცემთა ბაზები ვეღარ აკმაყოფილებს.

NoSQL მიმდინარეობა პოპულარული გახდა Google Big Table (2004 წ.) და Amazon DynamoDB (2012წ.) პროდუქტების გამოსვლის შემდეგ.

NoSQL-ის საჭიროებამდე ორივე კომპანია მონაცემთა რელაციური ბაზების პრობლემებმა მიიყვანა, დღეს კი უკვე 225-ზე მეტი სხვადასხვა სისტემაა ცნობილი [6].

მონაცემთა არარელაციური ბაზებია MongoDB, ArangoDB, Neoj4, DynamoDB და ასე შემდეგ, სხვადასხვა სისტემები ერთმანეთისგან

სტრუქტურულად განსხვავდება. მაგალითად, გრაფებზე აგებული ბაზები (ArangoDB, Neo4j), Key Value სტრუქტურის მქონე (DynamoDB, Riak), მონაცემთა დოკუმენტზე ორიენტირებული ბაზები, მაგალითად MongoDB და CouchDB, სადაც დოკუმენტები JSON ობიექტებად ინახება.

მონაცემთა არარელაციურ ბაზებში ინახება ***უსქემო*** მონაცემები, ადვილია მათი დაჯგუფება (clustering), გამოირჩევა დიდი სისწრაფით და ამარტივებს ამოცანის გადაწყვეტას. ძირითადად, მონაცემთა არარელაციური ბაზები არ იყენებენ SQL–ს, თუმცა ზოგიერთი მათგანი query ენის გამოყენების საშუალებას გვაძლევს – მაგალითად Cassandra SQL [7].

„**უსქემო**“-ში მოიაზრება, რომ MongoDB-ს აქვს დინამიკური სქემა - მასში შეგვიძლია შევქმნათ კოლექციები (collection) სტრუქტურის წინასწარ აღწერისგარეშე.

რელაციურ ბაზაში სქემის გარეშე არაფერი არსებობს, მონაცემთა არარელაციურ ბაზებში კი სქემა სასურველია, მაგრამ არა აუცილებელი.

რელაციურ ბაზაში ინფორმაციას ვერ შევიტანთ წინასწარ ჩამოყალიბებული სტრუქტურის გარეშე. არარელაციურ ბაზებში მსგავსი პრობლემა არ გვაქვს, – ინფორმაციის შეტანა მასში ნებისმიერი სახით შეგვიძლია. ამისთვის ცხრილის წინასწარ შექმნაც კი არაა საჭირო. მიუხედავად ამგვარი თავისუფლებისა, გარკვეული სტრუქტურის დაცვა მაინც სასურველია, რომ შემდგომ ინფორმაციის გაფილტვრა არ გართულდეს.

• **2010**–იანი წლებიდან დღემდე ვითარდება **NewSQL** მონაცემთა ბაზების ტექნოლოგიები:

NewSQL მონაცემთა ბაზები ცდილობენ რელაციური/SQL მოდელის გამოყენებით, მიაღწიონ NoSQL ბაზის მაღალ სისწრაფეს. ამასთან, შენარჩუნებული უნდა იყოს ACID კრიტერიუმები, რომ საიმედოდ განხორციელდეს OLTP ტრანზაქციები. NewSQL ტიპის მოწინავე მონაცემთა ბაზებია VoltDB, MemSQL და NuoDB. ეს სისტემები NoSQL-ის უმთავრეს უპირატესობებთან ერთად ტრადიციულ SQL ენასაც ინარჩუნებენ [6].

1.2. მონაცემთა რელაციური და არარელაციური ბაზების შედარება

რელაციურ მოდელში მონაცემები აისახება ერთი სტრუქტურით (ERM – Entity-Relationship Model), აპლიკაციაში კი გამოიყენება სრულიად განსხვავებული სახით. ამგვარად, მონაცემთა სტრუქტურა, რომელიც წარმოდგენილია ბაზაში, სრულიად განსხვავდება ოპერატიულ მეხსიერებაში მისი შესაბამისი სტრუქტურისაგან, შესაბამისად, პროგრამული უზრუნველყოფა მონაცემების დამუშავების დროს განსაკუთრებით დიდ დროსა და რესურსს უთმობს შემდგომ ოპერაციებს:

- მონაცემების წაკითხვა სხვადასხვა ცხრილებიდან;
- სხვადასხვა ცხრილებიდან ამოკითხული მონაცემების გადაბმა (Join) აპლიკაციისათვის სასურველ ობიექტში;
- მიღებული ობიექტის დამუშავება ისეთი ოპერაციებით, როგორიცაა, მაგალითად, პროექცია, შეზღუდვა, დაჯგუფება და სხვა;
- შედეგად მიღებული ობიექტის დაშლა და ცვლილებების განთავსება შესაბამის ცხრილებში (ამ დროს გასათვალისწინებელია ბაზის მთლიანობის შენარჩუნებლად დახარჯული დრო და რესურსიც).

მონაცემთა რელაციური ბაზებისგან განსხვავებით დოკუმენტზე ორიენტირებული NoSQL ბაზები მონაცემებს ინახავენ JSON ფორმატში [8]. თითოეული JSON დოკუმენტი არის ობიექტი, რომელიც მიეწოდება აპლიკაციას. ამგვარი ობიექტი შეიძლება ერთ დოკუმენტში აერთიანებდეს რამდენიმე რელაციური ცხრილის გადაბმით მიღებულ ინფორმაციას. შესაბამისად, ერთი მოთხოვნისთვის აღარ ხდება საჭირო სხვადასხვა ობიექტების დამუშავება, რაც ამარტივებს და ასწრაფებს ჩაწერა/წაკითხვის ოპერაციებს.

გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-ისთვის სტატისტიკური და ანალიტიკური ამოცანების გადასაწყვეტად შერჩეულ იქნა მონაცემთა ბაზის open-source სისტემა, კერძოდ MongoDB, რომელიც მონაცემებს JSON ტიპის დოკუმენტებში ინახავს. დოკუმენტების სტრუქტურა შეიძლება მოთხოვნის შესაბამისად იცვლებოდეს [9,10].

შესაბამისი ინფორმაცია ინახება გაერთიანებულ ობიექტებში/დოკუმენტებში და შესაძლებელია ჩანაწერების შექმნა სტრუქტურის (ველებისა და მნიშვნელობათა ტიპების) წინასწარი განსაზღვრის გარეშე, შემდგომ კი მარტივად ხდება შესაძლებელი ჩანაწერების სტრუქტურის შეცვლა ახალი ველის დამატებით ან არსებულის წაშლით.

ეს მოდელი იერარქიული კავშირების მარტივად წარმოდგენის საშუალებას გვაძლევს. კოლექციაში შემავალ დოკუმენტებს არ სჭირდება იდენტური ველები და მონაცემთა დენორმალიზაცია ჩვეულებრივი მოვლენაა. MongoDB მონაცემთა ბაზა გვაძლევს მაღალ წვდომადობასა და მასშტაბირების რეალიზაციის შესაძლებლობას, შესაძლებელს ხდის რეპლიკაციას და ავტომატურ სეგმენტაციას (auto-sharding).

1.2.1. მონაცემთა დამუშავების ტექნოლოგიების მიმოხილვა

ქვემოთ მოყვანილია MySQL და MongoDB მონაცემთა ბაზებში არსებული რამდენიმე ძირითადი ტერმინის შესაბამისობა (ნახ.2) [9].

MySQL	MongoDB
Table	Collection
Row	Document
Column	Field
Joins	Embedded documents, linking

ნახ. 2. ტერმინებისა და ცნებების შედარება მონაცემთა ბაზებში

კოლექცია (Collection) MongoDB-სთვის არის დოკუმენტების ჯგუფი. რელაციური მონაცემთა ბაზებისთვის კოლექციის ეკვივალენტი ცხრილია.

დოკუმენტი (Document) – კოლექციის ძირითადი ნაწილია, - რელაციურ ბაზებში იგივე კორტეჟი (Row), იმ განსხვავებით, რომ მონაცემთა არარელაციურ ბაზებში დოკუმენტები არის დინამიური. რელაციური ცხრილის სვეტის (Column) ნაცვლად MongoDB-ში გვაქვს ველი (Field). Join დაკავშირების ოპერაციის ნაცვლად გვაქვს ჩადგმული დოკუმენტები. MongoDB და SQL ბაზები გვთავაზობენ მრავალ საერთო თუ განსხვავებულ ფუნქციას (ნახ.3) [9].

დასახელება	MySQL	MongoDB
Rich Data Model	No	Yes
Dynamic Schema	No	Yes
Typed Data	Yes	Yes
Data Locality	No	Yes
Field Updates	Yes	Yes
Easy for Programmers	No	Yes
Complex Transactions	Yes	No
Auditing	Yes	Yes
Auto-Sharding	No	Yes

ნახ. 3. ფუნქციონალისა და შესაძლებლობების შედარება მონაცემთა ბაზებში

MongoDB მონაცემთა ბაზის მაღალფუნქციონალური მოთხოვნების ენას შეუძლია ტექსტებთან და სივრცით მონაცემებთან (Geospatial Queries) მუშაობა. აღნიშნული მოთხოვნების გამოყენება შესაძლებელია სხვადასხვა ტიპის მონაცემებთან.

მიმდინარე ქვეთავში შედარებულია მონაცემთა დამუშავების ტექნოლოგიები MySQL და MongoDB ბაზების მაგალითზე:

➢ „შევიტანოთ ინვენტარიზაციის ცხრილში/კოლექციაში ახალი ინვენტარის შესახებ ინფორმაცია (კორტეჟი/დოკუმენტი), რომელშიც გვაქვს შემდეგი ველები: ინვენტარის იდენტიფიკატორი (inv_ID), ინვენტარის დასახელება (Name), ღირებულება (price) და ინვენტარის კატეგორიის იდენტიფიკატორი (cat_ID)“.

• MySQL:

```
INSERT INTO inventory (inv_ID, name, amount, price, cat_ID)
VALUES (753, 'მაგიდა', 15, 230, 12)
```

• MongoDB:

```
db.inventory.insert ({
    inv_ID: 753,
    name: 'მაგიდა',
    amount: 15,
    price: 230,
```

cat_ID: 12

})

- გამოვიტანოთ ინვენტარის სრული სია:

• MySQL:

```
SELECT  * FROM  inventory
```

• MongoDB:

```
db.inventory.find()
```

- გამოვიტანოთ არსებული ინვენტარის მინიმალური და მაქსიმალური ფასები.

• MySQL:

```
SELECT MIN(price) FROM inventory;  // მინიმალური
SELECT MAX(price) FROM inventory;  // მაქსიმალური
```

• MongoDB:

```
db.inventory.find().sort({price:+1}).limit(1) // მინიმალური
db.inventory.find().sort({price:-1}).limit(1)  //  მაქსიმალური
```

- გამოვიტანოთ ინვენტარის საერთო რაოდენობა კონკრეტული კატეგორიისთვის (cat_ID = 12)“.

• MySQL:

```
SELECT  count ( inv_ID)
FROM    inventory i
WHERE  i.cat_ID = 12;
```

• MongoDB:

```
db.inventory.count ( {cat_ID = 12 } )
```

- შევამციროთ მე-12 კატეგორიის („მაგიდა“) ინვენტარის რაოდენობა ერთით“:

• MySQL:

```
UPDATE inventory SET amount = amount - 1
WHERE cat_ID = 12
```

• MongoDB:

```
db.inventory.update(
   { cat_ID: 12 },
   { $set: { amount: amount -1 } }
   { multi: true }
}
```

➢ წავშალოთ სავარძლის მონაცემები (cat_ID = 13)“.

• MySQL:

```
DELETE FROM inventory
WHERE cat_ID = 13
```

• MongoDB:

```
db.inventory.remove (
   ( cat_ID:  13  }
)
```

და ა.შ.

MongoDB მონაცემთა ბაზა მასშტაბირებისა და მონაცემთა სტრუქტურის ეფექტურობის ხარჯზე აპლიკაციების შექმნის და დიდი მოცულობის ინფორმაციის დამუშავების დროსა და სირთულეს ერთნაირად ამცირებს.

მონაცემთა სქემის ცვლილება, რელაციურ მონაცემთა ბაზებში განახლების შედარებით დიდ დროს მოითხოვს, როდესაც MongoDB-ს მოქნილი მოდელის - დინამიური სქემის გამო ასეთი პროცედურები სწრაფად ხორციელდება.

1.2.2. რა ხარვეზები აქვს მონაცემთა რელაციური ბაზების ტექნოლოგიას?

მონაცემთა რელაციურ ბაზებს არარელაციურთან შედარებით შემდეგი შეზღუდვები აქვთ:

• რელაციური მონაცემთა ბაზები ნორმალიზაციის რელაციურ მიდგომას იყენებენ, რაც გარკვეულწილად ზღუდავს პროგრამულ აპლიკაციას. ზოგიერთი სტრუქტურის წარმოსადგენად გვიწევს მონაცემებითა და/ან პროგრამით არასასურველი მანიპულირება.

- მონაცემების განახლებისას ვერსიების კონტროლის მექანიზმების გამოყენება ხშირად სხვადასხვა პრობლემებს იწვევს. მონაცემთა ბაზაში update ოპერაცია საერთოდ არ არის სასურველი, რადგან აღნიშნული ოპერაცია ამახინჯებს ინფორმაციას. იმის გათვალისწინებით, რომ ინფორმაციის შენახვა სულ უფრო და უფრო იაფი ჯდება, უმჯობესია თავიდან დავამატოთ ჩანაწერი, ვიდრე არსებული შევცვალოთ (რა თქმა უნდა, ეს მიდგომა ყველა ამოცანას არ ეხება).
- რელაციურ მონაცემთა ბაზებში ნორმალიზაცია უარყოფითად აისახება სისტემის წარმადობაზე, რადგან ნორმალიზაცია მოითხოვს მეტ ცხრილს, სხვადასხვა ცხრილების გადაბმებს, მეტ გასაღებებსა და ინდექსს, რაც მთლიანობაში მრავალჯერ ზრდის ოპერაციათა რაოდენობას.
- ასევე, მნიშვნელოვანი განსხვავებაა საქმის **დელეგირება**: ტრადიციული მიდგომით მოთხოვნათა უდიდესი ნაწილი მონაცემთა ბაზის სისტემაში სრულდება. შესაბამისად, კლიენტების რაოდენობის ზრდასთან ერთად მნიშვნელოვნად იზრდება ბაზის დატვირთვაც. რატომ უნდა შეუსრულოს ცენტრალიზებულმა სერვერმა/სერვერებმა კლიენტ აპლიკაციებს ისეთი დავალებები, როგორიცაა, მაგალითად, მიღებული შედეგის სორტირება, რისი შესრულებაც მომხმარებლებსაც მარტივად შეუძლიათ? თანაც, უკვე თითქმის აღარავის აქვს იმდენად სუსტი კომპიუტერი, ინფორმაციის მცირე ულუფის დამუშავება რომ ვერ გაუჭირდეს. NoSQL მიდგომა ითვალისწინებს საქმის დელეგირებას მომხმარებლის მხარეს. სორტირების მომხმარებბლის მხარეს გატანასთან ერთად სერვერი კლიენტს მთელ შედეგს ერთბაშად არ უბრუნებს რითაც თავს იცავს არასაჭირო რესურსის დახარჯვისგან, მაგალითად, მოთხოვნილი 1000 ჩანაწერის ნაცვლად სერვერი გზავნის პირველ ასეულს, მიღების დადასტურების შემდეგ შემდეგ ასეულს და ა.შ.

1.2.3. ვერტიკალური და ჰორიზონტალური სკალები

ინფორმაციული ტექნოლოგიების გამოყენების სფეროში საგრძნობლად გაიზარდა დიდი მოცულობის მონაცემებისა და პარალელური მოთხოვნების დამუშავების რაოდენობა. ეს ხშირად საჭიროს ხდის არსებული სისტემების არქიტექტურისა და კონფიგურაციის სრულიად შეცვლას.

ტრადიციულ რელაციურ ბაზებში, თუ იზრდება მოთხოვნების დატვირთვა, ვავითარებთ სერვერს (vertical scale). გარკვეული ზღვრის შემდეგ სერვერის გაძლიერება საკმაოდ ძვირი ჯდება და ნაკლებად ეფექტურია.

სპეციალისტები მივიდნენ დასკვნამდე, რომ რამდენიმე მძლავრ სერვერს სჯობს დიდი რაოდენობის შედარებით იაფფასიანი სერვერები (horizontal scale), რომლებიც იმუშავებენ როგორც ერთი კლასტერი (ნახ.4).

ნახ. 4. ვერტიკალური და ჰორიზონტალური სკალირება

1.2.4. როდის ჯობია არარელაციური მონაცემთა ბაზების გამოყენება?

სასურველია, გვესმოდეს ორივე, როგორც რელაციური, ასევე არარელაციური ბაზების პრინციპები და კონკრეტული ამოცანისთვის შევარჩიოთ სასურველი სისტემა. მე-4 ცხრილში მოყვანილია რელაციურ და არარელაციურ მონაცემთა ბაზების შესაბამის ამოცანათა მახასიათებლების აღწერა.

ცხრ. 4. მონაცემთა რელაციური და არარელაციური ბაზების ამოცანათა კლასების შედარება

NoSQL	• მონაცემთა საცავზე ველოდებით განსაკუთრებით დიდ დატვირთვას. • ტრანზაქციები ძირითადად OLAP ტიპის იქნება (სისტემის კონფიგურაციის მიხედვით შესაძლებელია OLTP მოთხოვნებზე ოპტიმიზირებაც). • გჭირდება საცავი, რომლის განვითარებაც ჰორიზონტალურად არის საჭირო (horizontal scale). • სასურველია სიმარტივე, რთული SQL მოთხოვნების გარეშე.
RDBMS	• მონაცემთა საცავს უნდა შეეძლოს ასევე დიდი დატვირთვის გამკლავება, მაგრამ ტრანზაქციები ძირითადად OLTP სახისაა. • ვირჩევთ, თუ გვჭირდება უფრო დახვეწილი და რთული სტრუქტურის გამოყენება. • ვირჩევთ, თუ გვჭირდება SQL ენის გამოყენება.

ქვემოთ მოყვანილია რამდენიმე ამოცანა, რომლის გადაწყვეტაც მონაცემთა არარელაციური ბაზების მეშვეობით უფრო მოსახერხებელია, ვიდრე რელაციური ბაზებით:

• Logging/Archiving – ლოგ–სერვერები – Log-mining ტიპის სისტემებისთვის მონაცემთა არარელაციური ბაზა ძალიან მოსახერხებელია, რადგან ადვილია სხვადასხვა სერვერებიდან log ჩანაწერების აგროვება, დამუშავება და გაანალიზება.

• Social Computing Insight – გამოთვლები განაწილებულ გარემოში.

• External Data Feed Integration – მრავალ კომპანიას უწევს ბიზნეს–პარტნიორებისგან შემოსული ინფორმაციის ინტეგრაცია საკუთარ მონაცემთა ბაზაში. როგორც წესი, ამგვარ მონაცემებზე მანიპულაცია არაერთ რთულ პროცესთან არის დაკავშირებული, თანაც, შემომავალი მონაცემების სტრუქტურა მიმღები კომპანიისგან დამოუკიდებლად იცვლება ხოლმე. ცხადია, ამგვარი ამოცანის გადაწყვეტა უსქემო სტრუქტურით, მონაცემთა დოკუმენტზე ორიენტირებული ბაზის მეშვეობით უმჯობესია.

• Real-time stats/analytics – ზოგჯერ გვჭირდება მონაცემთა ბაზის გამოყენება იმისთვის, რომ თვალი ვადევნოთ ვებგვერდის დატვირთვას

(უნიკალური ვიზიტები, მომხმარებლების მოქმედების სტატისტიკის წარმოება...).

ლოგ–სერვერების ან სოციალური ქსელის ტიპის ამოცანებში ახალი ჩანაწერების დამატების სისწრაფე უფრო მნიშვნელოვანია, ვიდრე ახალი ინფორმაციის ყველა მომხმარებელთან ერთდროულად ასახვა. ამ შემთხვევაში სისტემას აკონფიგურირებენ fire-and-forget პრინციპით, რაც გულისხმობს, რომ ახალ ჩანაწერებს უბრალოდ „ვისვრით“ და არ ველოდებით ჩანაწერის დადასტურებას.

1.3. მონაცემთა შენახვის ტიპები

NoSQL ბაზებში მონაცემების შენახვის 4 ძირითადი ტიპი არსებობს:

1. **Key-Value Store** – აქვს დიდი ჰეშ ცხრილი გასაღებებისა და მათი მნიშვნელობებისათვის. მონაცემებზე წვდომისთვის გამოიყენება პირველადი გასაღები (Primary Key). გავრცელებული გასაღები-მნიშვნელობა ტიპის მონაცემთა ბაზებია DynamoDB, Riak, Redis…;

2. **Document-based Store** - მონაცემებს ინახავს იარლიყიანი ელემენტე-ბისგან (tagged elements) შემდგარი დოკუმენტების სახით. ამ ტიპის მონაცემთა ბაზებს JOIN ოპერატორის მხარდაჭერა არ აქვთ. ცხრილების გადაბმის/გაერთიანების ლოგიკა აპლიკაციის მხარეს უნდა დაიწეროს. სამაგიეროდ, ობიექტზე ორიენტირებული აპლიკაციისთვის ძალიან მარტივია მთელი საჭირო ინფორმაციის ერთი დოკუმენტიდან ამოღება. გავრცელებული დოკუმენტ-ორიენტირებული მონაცემთა ბაზებია MongoDB, Elastic, ArangoDB, Azure DocumentDB…;

3. **Column-based Store** - მეხსიერების თითოეული ბლოკი ინახავს მხოლოდ ერთი სვეტის ინფორმაციას, რაც ბევრად ამარტივებს აგრეგატული ფუნქციების (MIN, SUM, AVG, COUNT…) შესრულებას. ამ ტიპის მონაცემთა ბაზები კვლავ ცხრილურ სტრუქტურას იყენებენ, მაგრამ JOIN ოპერატორის მხარდაჭერა არ აქვთ. ცხრილების გადაბმის/გაერთიანების ლოგიკა აპლიკაციის მხარეს არის დასაწერი. გავრცელებული მონაცემთა სვეტებზე ორიენტირებული ბაზებია HBase, Cassandra, Apache Flink, Cloudata;

4. **Graph-based**-ქსელური ტიპის მონაცემთა ბაზა, რომელიც იყენებს წიბოებსა და კვანძებს მონაცემების შესანახად და წარმოსადგენად. გავრცელებული მონაცემთა გრაფული ბაზებია Neo4J, ArangoDB, GraphBase….

ქვემოთ განხილულიამონაცემთა შენახვის თითოეული ტიპის მახასიათებლები.

1.3.1. გასაღები-მნიშვნელობა შენახვის ტიპის მონაცემთა ბაზა

Key-Value ყველაზე მარტივი სტრუქტურაა: გვაქვს უნიკალური პარამეტრი key და შესაბამისი მნიშვნელობა - value (ცხრ.5).

ცხრ. 5. გასაღები-მნიშვნელობა (Key-Value) ტიპის მონაცემთა ბაზა

პოპულარული მონაცემთა ბაზები	Redis, Voldemort, DynamoDB, Oracle BDB, Amazon SimpleDB, Riak [6]
ტიპური გამოყენება	Content caching (განაწილებულ სერვერებზე დიდი ოდენობის ინფორმაციის დამუშავებაზე ორიენტირებული სისტემები), ლოგ–სერვერები...
მოდელი	გასაღები-მნიშვნელობა წყვილების კოლექცია
დადებითი	სწრაფი ძებნა
უარყოფითი	შენახულ მონაცემებს არ აქვს სქემა

შესაბამისი ლოგიკა რელაციურ ბაზაში ორი სვეტით აეწყობოდა, – Primary key და მასთან დაკავშირებული სვეტი მონაცემებისთვის. განსხვავება ის იქნებოდა, რომ რელაციურ ბაზაში წინასწარ უნდა მიგვეთითებინა მონაცემების სვეტის ტიპი, NoSQL ბაზებში კი ეს მნიშვნელობა შეიძლება იყოს რიცხვი, ტექსტი, სურათი და ასევე, არ არის აუცილებელი, რომ ყველა სტრიქონში ერთი და იმავე ტიპის ინფორმაცია ინახებოდეს.

მონაცემების ამგვარი შენახვა საშუალებას იძლევა, რომ ერთი ცხრილის სხვადასხვა ჩანაწერები კლასტერის სხვადასხვა კვანძებზე მოვათავსოთ.

როგორც წესი, ამგვარი გადანაწილება გასაღების მნიშვნელობის მიხედვით ხდება.

Key-Value ტიპის ბაზებში რთულია ატომარობისა და კონსისტენტურობის დაცვა – სანამ ერთი მომხმარებელი ჩანაწერის განახლების ოპერაციას ასრულებს, სხვა მომხმარებელმა შეიძლება წასაკითხად მიაკითხოს იმავე ჩანაწერს. ამგვარ შემთხვევებში გვაქვს ორი ვარიანტი – მივაწოდოთ მომხმარებელს ჩანაწერის ბოლო განახლებული ვერსია, ან მივაწოდოთ ყველა არსებული ვერსია და თავად მივცეთ იმის საშუალება, რომ გაარკვიოს, თუ რომელი ვერსიის გამოყენება ურჩევნია.

ამგვარი ლოგიკით, აპლიკაციის მხარეს შესაძლებელი ხდება კონსისტენტურობის დაცვაც, მაგრამ თუ დასმული ამოცანისთვის კრიტიკულია ბაზის ტრანზაქციულობა, მაშინ key-value storage საიმედო გადაწყვეტილება ვერ იქნება.

როგორც წესი, key-value მეთოდი იყენებს ჰეშ ცხრილებს, რომლებშიც ინფორმაცია ლოგიკურ გაერთიანებებად (bucket) არის დაყოფილი. რეალურად, გასაღები არის ჩვენს გასაღებს + Bucket მნიშვნელობების გაერთიანებული ჰეში – hash (Bucket+ Key)

CAP თეორემას თუ მივუბრუნდებით, ცხადი ხდება, რომ key-value მეთოდი იდეალურია Availability და Partition თვისებების მხარდასაჭერად. მაგრამ საგრძნობლად მოიკოჭლებს Consistency-ის კუთხით.

მონაცემების გასაღები-მნიშვნელობა შენახვის ტიპის საკმაოდ კარგი გამოყენებაა სესიის ინფორმაციის შენახვა, როგორც key - სესიის იდენტიფიკატორი და value - სესიასთან დაკავშირებული ინფორმაციის კრებული.

ასევე, რეალური ამოცანაა მომხმარებლის პირადი ინფორმაციის შენახვა, სადაც გასაღები (key) არის მომხმარებლის უნიკალური იდენტიფიკატორი, მნიშვნელობა (value) კი მთელი ის ინფორმაცია, რაც ახასიათებს მომხმარებელს.

მე-6 ცხრილში Key-value პრინციპის გამოყენებით ასახულია საქართველოს ბანკის ფილიალები დასახლებების მიხედვით.

ცხრ. 6. გასაღები-მნიშვნელობა შენახვის ტიპის მაგალითი

Key	Value
“ლანჩხუთი”	{“ჟორდანიას ქუჩა #101″}
“ბათუმი”	{“ნინოშვილის ქუჩა #11, სასტუმრო ინტურისტ პალასი″,“აღმაშენებლის ქუჩა #21“}
“ამბროლაური“	{“კოსტავას ქუჩა #2“}
“თბილისი”	{“ვეკუას ქუჩა #1თბილისი","პუშკინის ქუჩა #3თბილისი","კოსტავას ქუჩა #24თბილისი","თაბუკაშვილის ქუჩა #38თბილისი"}

გასათვალისწინებელია, რომ გასაღებად ტექსტური მნიშვნელობების არჩევის შემთხვევაში მნაცემების დაგროვებასთან ერთად უფრო და უფრო გაგვიჭირდება უნიკალურობის დაცვა.

მონაცემთა გასაღები-მნიშვნელობა ტიპის ბაზა ინფორმაციის დასამუშავებლად მომხმარებელს შემდეგ ფუნქციებს სთავაზობს:

- Get(key), აბრუნებს პარამეტრად მიღებული key გასაღების შესაბამის value მნიშვნელობას;
- Put(key, value), ერთმანეთთან აკავშირებს მნიშვნელობასა და გასაღებს;
- Multi-get(key1, key2, .., keyN), აბრუნებს პარამეტრად მიღებული გასაღებების შესაბამისი value მნიშვნელობების მიმდევრობას;
- Delete(key), მონაცემთა ბაზიდან წაშლის შესაბამის ჩანაწერს.

გასაღები-მნიშვნელობა და რელაციურ მოდელებს თუ შევადარებთ, შემდეგ შესაბამისობებს მივიღებთ:

- Table -> bucket
- Row -> key-value
- Rowid -> key

1.3.2. დოკუმენტ-ორიენტირებული მონაცემთა ბაზა

დოკუმენტ-ორიენტირებული მონაცემთა ბაზა (Document-Oriented Database) არის NoSQL მიდგომით რეალიზებული მონაცემთა ბაზების მართვის სისტემა, რომელიც გამოიყენება დოკუმენტების (მონაცემთა იერარქიული სტრუქტურების) შესანახად და დასამუშავებლად [7,8,11,12].

დოკუმენტ-ორიენტირებულ მონაცემთა ბაზის მართვის სისტემას საფუძვლად უდევს დოკუმენტების საცავი (document Store), რომელსაც აქვს ხის სტრუქტურა - იწყება ფესვური კვანძით და შეიძლება შეიცავდეს რამდენიმე შიგა კვანძს და ფოთლების კვანძს. ფოთლების კვანძი შეიცავს მონაცემებს, რომლებიც დოკუმენტის დამატების დროს შედის ინდექსებში, რაც უზრუნველყოფს საჭირო მონაცემებისკენ მიმავალი გზის პოვნას რთული სტრუქტურების შემთხვევაშიც კი [13].

მოთხოვნის საფუძველზე API (Application Programming Interface) ეძებს დოკუმენტებსა და მათ ნაწილებს. დოკუმენტები შეიძლება იყოს ორგანიზებული (დაჯგუფებული) კოლექციებში. ეფექტური ინდექსირების მიზნით უმჯობესია კოლექციებში მსგავსი სტრუქტურების მქონე დოკუმენტების გაერთიანება.

დოკუმენტ-ორიენტირებული მონაცემთა ბაზები გამოიყენება შინაარსის მართვის სისტემებში (CMS - Content Management System), საგამომცემლო საქმეში, დოკუმენტების საძიებო სისტემებში და სხვ. ასეთი ბაზების მართვის სისტემების მაგალითებია: MongoDB, CouchDB, Couchbase, MarkLogic, eXist, IBM Lotus Notes და სხვ [14-16].

დოკუმენტი მონაცემთა ბაზაში მისამართდება უნიკალური გასაღების საშუალებით. ხშირად ეს გასაღები მარტივი სტრიქონია, რომელიც შეიძლება იყოს URI (Unified Resource Identifier) ან გზა (path) დოკუმენტამდე. ამგვარი გასაღების ან მისი ინდექსის საშუალებით დოკუმენტი ბაზაში მარტივად მოიძებნება და შესაძლებელია მისი სწრაფად ამოღება.

მე-7 ცხრილში წარმოდგენილია დოკუმენტზე ორიენტირებული მონაცემთა ბაზების მახასიათებლები.

ცხრ. 7. დოკუმენტ-ორიენტირებული მონაცემთა ბაზა

პოპულარული მონაცემთა ბაზები	CouchDB, MongoDb
ტიპური გამოყენება	ლოგ–სერვერები, ვებ აპლიკაციები (Key-value–ს დახვეწილი ვერსია)
მოდელი	Key-Value წყვილების კოლექციათა კოლექციები (ჩადგმული კოლექციები)
დადებითი	ფუნქციონირებს არასრული ინფორმაციის შემთხვევაშიც
უარყოფითი	ტრანზაქციების წარმადობა; არ აქვს სტანდარტული სინტაქსი მოთხოვნების ჩამოსაყალიბებლად.

Document-based მონაცემთა ბაზები ინფორმაციას List<Object> სიის სახით ინახავს. ასეთ სიებში შესაძლებელია მრავალი სხვადასხვა სახის მონაცემის ჩაწერა.

დოკუმენტებზე ორიენტირებული და რელაციური მოდელების შედარებით შემდეგ შესაბამისობას მივიღებთ:

Table -> collection

Row -> BSON დოკუმენტი

Column -> BSON ველი

Rowed -> _id

Index -> Index

Join -> ჩადგმული დოკუმენტი (Embedded Document)

Partition -> Shard

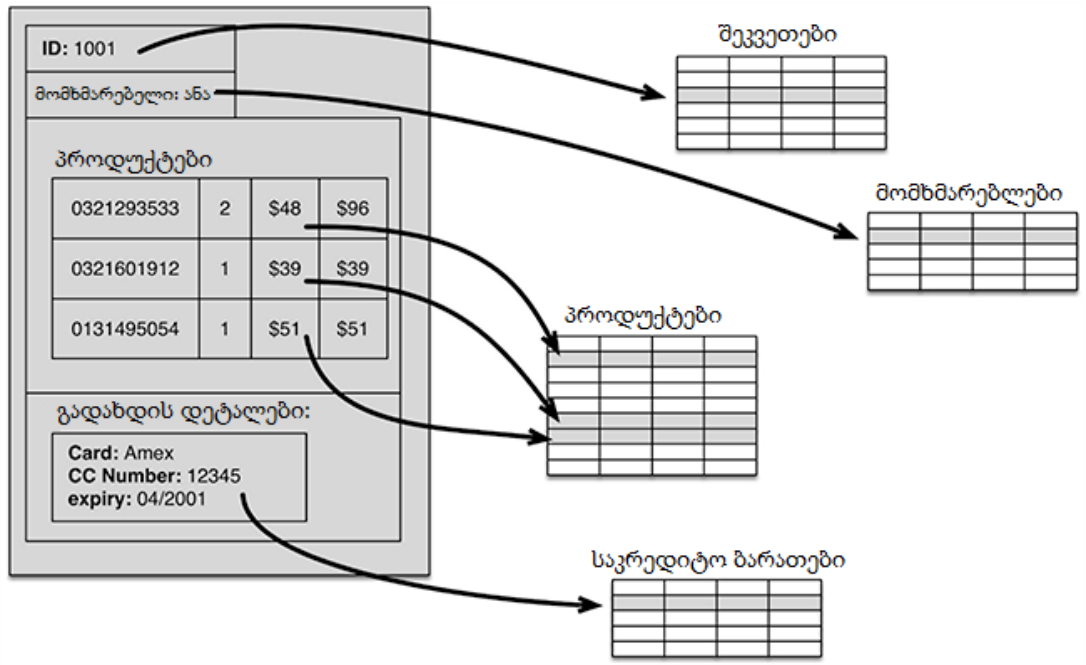
Partition Key -> Shard Key

Document-based მოდელის გამოყენების საინტერესო მაგალითია CMS ძრავებსა [17] და ყველა იმ დავალებაში, სადაც გვიწევს, ბრაუზერს ყოველ მიმართვაზე თავიდან დავაგენერირებინოთ კოდი. ცხადია, ბევრად უკეთეს წარმადობას მივიღებთ, თუ უკვე დაგენერირებულ სკრიპტს შევინახავთ და ბრაუზერს არ მოუწევს ყოველ ჯერზე რესურსის ხელახლა დახარჯვა.

1.3.3. მონაცემთა შენახვის დოკუმენტზე ორიენტირებული და რელაციური მოდელების შედარება

მონაცემების წარმოდგენის დოკუმენტზე ორიენტირებული (Document-based) სისტემები ინფორმაციას ინახავენ დოკუმენტებად, გარკვეული სტრუქტურის სახით. მონაცემთა დოკუმენტზე ორიენტირებული ბაზები ინფორმაციას List<Object> სიის სახით ინახავს. ამგვარ სიებში შესაძლებელია ჩაიწეროს მრავალი სხვადასხვა სახის მონაცემი (ნახ.5).

მაგალითისთვის მონაცემთა დოკუმენტზე ორიენტირებული ბაზები შეიძლება შევადაროთ ფოლდერს, რომელშიც მრავალი ფაილია მოთავსებული, თითოეულ ფაილს კი სხვადასხვა სტრუქტურა და შიგთავსი გააჩნია. ამ მაგალითში თითოეული ფოლდერი ჩვენთვის დოკუმენტების კოლექცია იქნება (collection).



ნახ. 5. მონაცემთა შენახვის დოკუმენტზე ორიენტირებული და რელაციური მოდელების სტრუქტურული შედარება

შესაძლებელია ერთი დოკუმენტის მეორეში ჩადგმაც, რადგან დოკუმენტებს JSON ობიექტებად ვინახავთ.

1.3.4. სვეტებზე ორიენტირებული მონაცემთა ბაზა

Column-based Store პრინციპი შემუშავებულ იქნა მრავალ მანქანაზე გადანაწილებული დიდი რაოდენობის მონაცემების დამუშავებისთვის (ცხრ.8).

ცხრ. 8. სვეტებზე ორიენტირებული მონაცემთა ბაზა

პოპულარული მონაცემთა ბაზები	Cassandra, HBase, Riak
ტიპური გამოყენება	განაწილებული ფაილური სისტემები Hadoop Distributed File System (HDFS)
მოდელი	სვეტი –> სვეტების გაერთიანება
დადებითი	სწრაფი ძებნა, განაწილებული გარემოს საუკეთესო მხარდაჭერა
უარყოფითი	დაბალი დონის API

1.3.5. გრაფებზე ორიენტირებული მონაცემთა ბაზა

მონაცემთა გრაფული ბაზის მართვის სისტემები დაფუძნებულია მონაცემთა გრაფულ მოდელზე, ანუ ინფორმაციის შენახვა ცხრილებისა და ატრიბუტების ნაცვლად გრაფული სტრუქტურებით - კვანძებით (nodes) და მათ შორის კავშირებით (გრაფის წიბოები – edges) ხდება. ასეთი კავშირი შეიძლება რამდენიმე დონესაც მოიცავდეს (ღრმა კავშირი). მონაცემთა შენახვის გრაფებზე ორიენტირებული მოდელი ძალზე აქტუალურია სოციალური ქსელების, ბიოინფორმატიკის, რთული მარშრუტების და სხვადასხვა სფეროს ამოცანების გადასაწყვეტად [8,15].

გრაფული მონაცემთა ბაზა არის ქსელური მოდელის (ან RDF-მოდელის) რეალიზაციის ნაირსახეობა. მისი კონცეფცია ჯერ კიდევ 80-იან წლებში გამოჩნდა, ხოლო პირველი გრაფული რეალიზაცია 2007 წელს, Neo4J სისტემის სახით [8]. დღეისთვის კი უკვე ათეულობით ამგვარი სისტემა არსებობს, მაგალითად: OrientDB, ArangoDB, MarkLogic, Oracle Spatial and Graph და სხვ. [8,18-24].

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

„სუბიექტი — პრედიკატი — ობიექტი“

ამგვარ სამეულებს Triple Stores ანუ სამადგილიანი შენახვა ეწოდება.

1.4. Hadoop – „დიდ მონაცემთა“ ტექნოლოგია

ბოლო წლებში აქტუალური გახდა ტერმინი IOT (Internet Of Things), რაც თავის თავში მოიცავს ყველა იმ აპარატს და კომპიუტერულ ტექნიკას, რაც მიმდინარე დროში დიდი რაოდენობით მონაცემებს ქმნის. IOT მონაცემების შეგროვება და დამუშავება აქტუალური ამოცანაა გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისათვისაც („112“).

ყოველდღიურად შესაგროვებელ მონაცემებში შედის სასწრაფო დახმარებისა და ევაკუატორების მანქანების GPS კოორდინატები, ასევე სხვადასხვა სენსორების (კვამლისა და CO-ს დეტექტორები) მონაცემები და ყველა ის ტექნიკა, რაც ძირითადი ფუნქციონირების პარალელურად დიდი რაოდენობის დამხმარე ინფორმაციას წარმოქმნის (metadata).

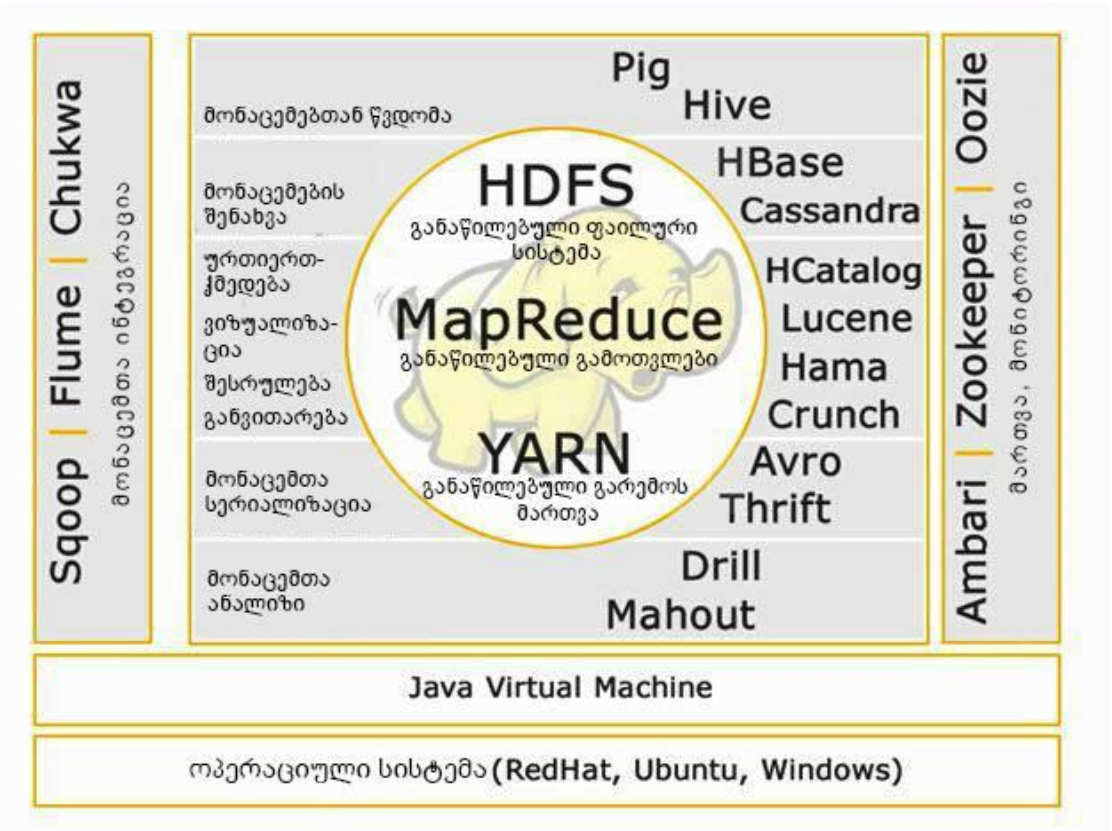
ამ რაოდენობის მონაცემების დამუშავებას სრულიად განსხვავებული გადაწყვეტილება სჭირდება არა მხოლოდ რელაციური და არარელაციური ბაზების, არამედ იმ სერვერული არქიტექტურის დონეზე, სადაც ვაყენებთ მონაცემთა ბაზებს. დღეისთვის საუკეთესო გამოსავალი დააპროექტა Apache Software Foundation-მა, სახელით Hadoop. Hadoop უფასო, ჯავაზე დაფუძნებული პლატფორმაა, რომელიც დიდი ზომის მონაცემთა ნაკადის დასამუშავებლად შეიქმნა.

Hadoop ეკოსისტემაში სხვადასხვა პროდუქტია გაერთიანებული, ბირთვად კი სამი ძირითადი კომპონენტი აქვს (ნახ.6):

HDFS (Hadoop Distributed File System) - განაწილებული ფაილური სისტემა მონაცემების შესანახად

Map Reduce - მთავარი კომპონენტი განაწილებული გამოთვლების ჩასატარებლად

YARN (Yet Another Resource Negotiator) - განაწილებული გარემოს მართვა.



ნახ. 6. Hadoop ეკოსისტემა

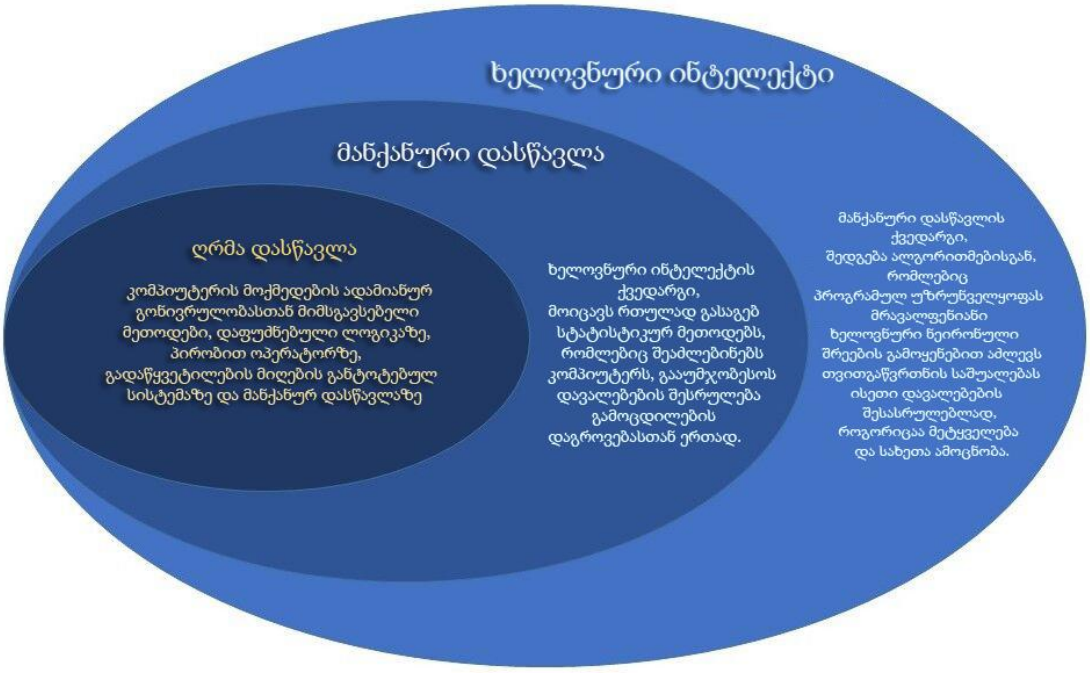
Hadoop გარემო ერთი შეხედვით ფაილური სისტემის ქცევას ემსგავსება, მაგრამ რეალურად ეკოსისტემაში შემავალი პროდუქტების ურთიერთქმედების შედეგად პროგამული უზრუნველყოფის დახვეწილ პლატფორმას ვღებულობთ. Hadoop ეკოსისტემის გამოყენებით შეგვიძლია ინტენსიური სიხშირის მონაცემების შემცველი აპლიკაციების მხარდაჭერა ან პეტაბაიტიანი მონაცემების დამუშავება. HDFS ფაილურ სისტემაში ისეთი ოპერაციები, როგორიცაა ფაილისა და დირექტორიის შექმნა, წაშლა ან განახლება, ატომარული პროცესებია [25]. კონსისტენტურობისთვის Hadoop ფაილური სისტემა „Shared Nothing“ არქიტექტურას იყენებს, რაც გულისხმობს, რომ კლასტერში შემავალი კვანძები ერთმანეთისგან სრულიად დამოუკიდებლად მუშაობენ. საერთო რესურსის არქონის

შედეგად კვანძების ერთმანეთთან ურთიერთქმედება მინიმუმამდეა დაყვანილი და შესაბამისად, აღარ გვაქვს ბოთლის შევიწროვებული ყელის (Bottleneck) პრობლემა. აღნიშნული მოდელი გვაძლევს ჰორიზონტალური ზრდის თითქმის ულიმიტო შესაძლებლობას.

1.5. მანქანური დასწავლა

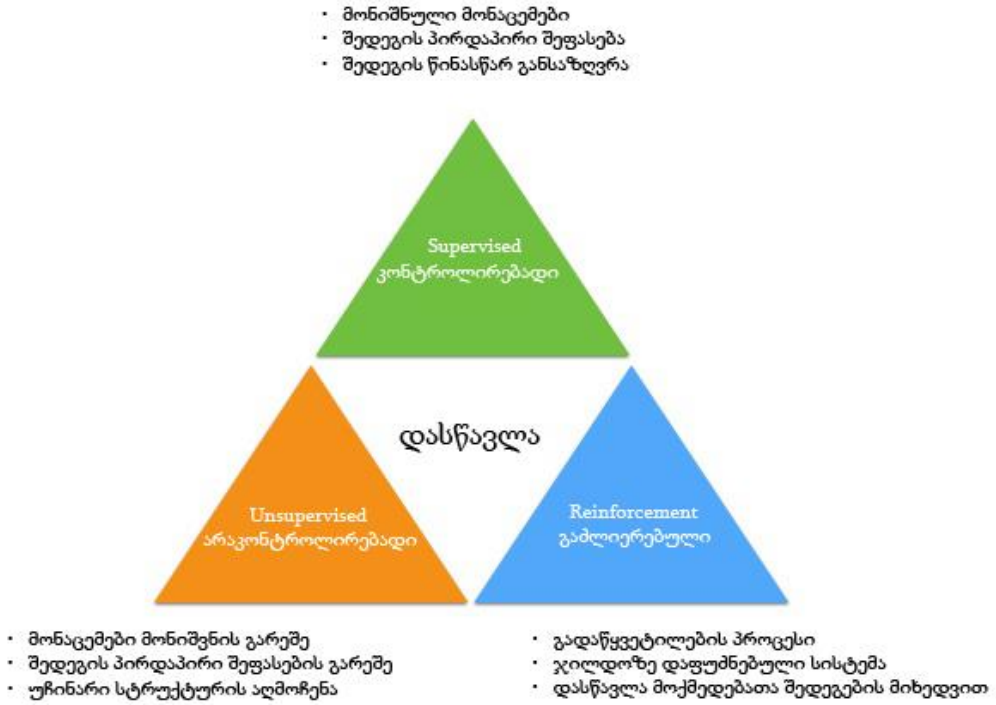
მანქანური დასწავლის მეცნიერება შეისწავლის ალგორითმებს, რომლებიც საშუალებას აძლევენ კომპიუტერს დაისწავლოს ამა თუ იმ საქმის კეთება. დიდი რაოდენობით დაგროვებულმა ინფორმაციამ მოგვცა საშუალება, განვვითარებულიყავით მანქანური დასწავლის (Machine Learning) მიმართულებით. ჯერ კიდევ 1959 წელს ამერიკელმა მეცნიერმა არტურ სამუელმა (Arthur Samuel) მანქანური დასწავლა განმარტა როგორც „მეცნიერების სფერო, რომელიც კომპიუტერებს აძლევს თვითდასწავლის შესაძლებლობას, ცხადად დაპროგრამების გარეშე“. დღეს კი უკვე ხელოვნური ინტელექტი მართავს თვითმფრინავს, დახვეწილია სახეთა ამოცნობის ალგორითმები და თუნდაც, თითქმის არასდროს გვხვდება spam ტიპის იმეილები inbox-ში [26].

დასწავლა ხდება რაიმე კონკრეტული მონაცემების, ინფორმაციის, გამოცდილებისა და მითითებების ანალიზის ხარჯზე. მანქანური დასწავლა არის ხელოვნური ინტელექტის ქვესფერო (ნახ.7).



ნახ. 7. ხელოვნური ინტელექტის სფეროები

მანქანური დასწავლის ძირითადი მიზანია ისეთი ალგორითმების შემუშავება რომლებიც დამოუკიდებლად შეძლებენ დასწავლას.



ნახ. 8. კონტროლირებადი, არაკონტროლირებადი და გაძლიერებული დასწავლა

როგორც ნახაზ 8-ზე ჩანს, გამოიყოფა 3 ტიპის დასწავლა:

- **კონტროლირებადი დასწავლა (Supervised learning):**

კონტროლირებადი დასწავლის დროს სისტემა აგებს მოდელს - სწავლობს მონაცემთა სასწავლო ნაკრების დახმარებით (შემავალი პარამეტრები + შედეგები). როდესაც მოდელი იღებს დაუმუშავებელ, ახალ ინფორმაციას, ის დასწავლის შედეგად მიღებული მოდელის საშუალებით აბრუნებს სავარაუდო შედეგს. გამოცდილების დაგროვებასთან ერთად უმჯობესდება სავარაუდო პასუხის სიზუსტეც.

კონტროლირებადი დასწავლა თავის მხრივ იყოფა **კლასიფიკაციის** და **რეგრესიის** ამოცანებად.

კლასიფიკაციის ამოცანის მიზანია შემომავალ ინფორმაციას მიამაგროს იარლიყი კლასების სასრული რაოდენობის მქონე სიმრავლიდან. ამ ჯგუფის ამოცანებს მიეკუთვნება სპამის ამოცნობა და რეკლამების შემოთავაზების სისტემა.

რეგრესიის ამოცანა სავარაუდო პასუხად გვიბრუნებს მის მიერ დაკვირვებადი ინფორმაციის უწყვეტ მაჩვენებელს. მაგალითად, როგორ შეიცვლება ბინების ფასი, რა იქნება აქციების კურსი და ა.შ.

- **არაკონტროლირებადი დასწავლა (Unsupervised learning):**

ალგორითმმა მხოლოდ შემავალი მონაცემების გათვალისწინებით, წინასწარ მიწოდებული შედეგების გარეშე უნდა დაადგინოს, რა ინფორმაციასთან აქვს კავშირი, გამოიკვლიოს მონაცემები და ამ შესწავლილ მონაცემებში რაიმე სახის სტრუქტურა აღმოაჩინოს. მაგალითად,

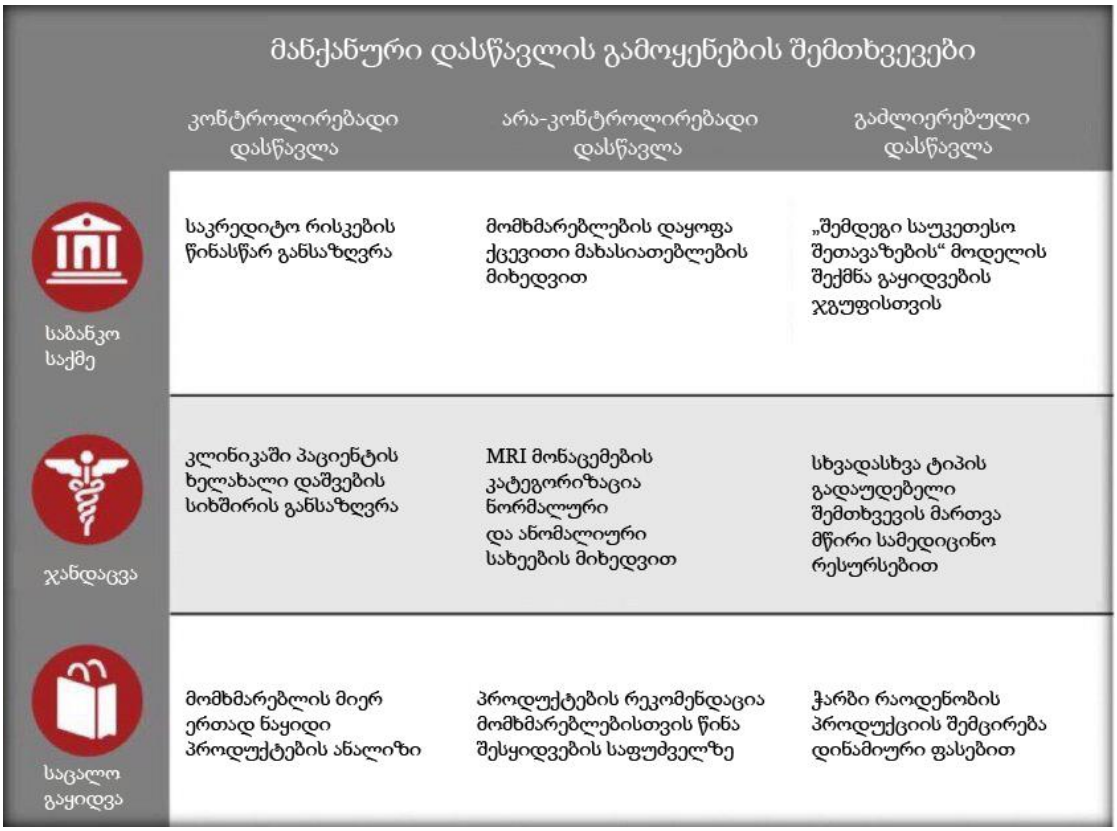
მომხმარებლების დაყოფა ქცევითი მახასიათებლების მიხედვით: პერსპექტივებისა და მომხმარებლების ანალიზი მრავლობითი სეგმენტების განვითარებისათვის clustering დაჯგუფების მეთოდის გამოყენებით.

- გაძლიერებული დასწავლა (Reinforcement learning):

გაძლიერებული დასწავლის ალგორითმები გამოცდილებისა და შეცდომების ხარჯზე ადგენენ ყველაზე კარგი შედეგის მქონე ქმედებას. აღნიშნული ალგორითმები სამი ძირითადი ნაწილისგან შედგებიან: აგენტი

(გადაწყვეტილების მიმღები), გარემო (ყველაფერი რაც უკავშირდება აგენტს) და აგენტის მიერ შესაძლო ქმედებები. გაძლიერებული დასწავლის მიხედვით აგენტმა უნდა აირჩიოს ისეთ მოქმედებათა მიმდევრობა, რომ შედეგი მაქსიმალურად დადებითი იყოს [27].

მანქანური დასწავლა მჭიდრო კავშირშია სტატისტიკასთან, ფიზიკასთან, მათემატიკის დარგებთან და ა.შ. მე-9 ნახაზზე მოყვანილია რეალური ამოცანები, რომელთა გადაწყვეტისთვის საუკეთესო გამოსავალია მანქანური დასავლა [28].



ნახ. 9. მანქანური დასწავლის გამოყენების სფეროები

- **საკრედიტო რისკების წინასწარ განსაზღვრა:** სათანადო ინფორმაციით უზრუნველყოფილი მანქანური დასწავლის მოდელის შედგენა ურჩი გადამხდელების გამოსავლენად
- **მომხმარებლების დაყოფა ქცევითი მახასიათებლების მიხედვით:** პერსპექტივებისა და მომხმარებლების ანალიზი მრავლობითი

სეგმენტების განვითარებისათვის clustering დაჯგუფების მეთოდის გამოყენებით

- **„შემდეგი საუკეთესო შეთავაზების“ მოდელის შექმნა გაყიდვების ჯგუფისთვის:** რეკომენდატორი მოდელის შექმნა, რომელიც ძველი შეთავაზებებისა და მათი დადებითი თუ უარყოფითი პასუხების გათვალისწინებით დაეხმარება გაყიდვების ჯგუფს ოპტიმალური შეთავაზებების განხორციელებაში.
- **კლინიკაში პაციენტის ხელახალი დაშვების სიხშირის განსაზღვრა:** რეგრესიის მოდელის აგება პაციენტის მკურნალობის რეჟიმისა და კლინიკაში ხელახალი დაშვების მონაცემების უზრუნველყოფით, რათა ამოირჩეს ის ცვლადები, რომლებიც საუკეთესო კორელაციას ავლენენ კლინიკაში ხელახალ დაშვებასთან.
- **MRI მონაცემების კატეგორიზაცია ნორმალური და ანომალიური სახეების მიხედვით:** ღრმა დასწავლის მეთოდის გამოყენება სახეთა სხვადასხვა თავისებურებების შესწავლის მოდელის ასაგებად და სხვადასხვა ანომალიის ამოსაცნობად [29] .
- **სხვადასხვა ტიპის გადაუდებელი შემთხვევის მართვა მწირი სამედიცინო რესურსებით:** მარკოვის გადაწყვეტილების პროცესის აგება თითოეული ტიპის გადაუდებელი შემთხვევისათვის მკურნალობის სტრატეგიის გამოსავლენად [30].
- **მომხმარებლის მიერ ერთად ნაყიდი პროდუქტების ანალიზი:** კონტროლირებადი დასწავლის მოდელის აგება პროდუქტების ხშირი კრებულის გამოსავლენად.
- **პროდუქტების რეკომენდაცია მომხმარებლებისთვის წინა შესყიდვების საფუძველზე:** კოლაბორაციული ფილტრის მოდელის აგება წინა შესყიდვების საფუძველზე „მათი მსგავსი მომხმარებლების“ მიერ.
- **ჭარბი რაოდენობის პროდუქციის შემცირება დინამიური ფასებით:** ფასების განსაზღვრის დინამიური მოდელის აგება ფასების

დასარეგულირებლად შეთავაზებაზე მომხმარებლის რეაგირების მიხედვით.

1.6. ამოცანის დასმა

თანამედროვე ტექნოლოგიების სამყაროში ინფორმაციის უდიდეს ნაწილს ადამიანების ნაცვლად კომპიუტერული ტექნიკა ქმნის. ინფორმაციის ნაკადი იმდენად სწრაფად იზრდება, რომ საჭირო ხდება ინფორმაციის დამუშავების თანამედროვე მეთოდების მოძიება.

ინფორმაციის რაოდენობის ექსპონენციალურმა ზრდამ ახალი გამოწვევების წინაშე დაგვაყენა. საჭირო გახდა დიდი მოცულობის მონაცემების შენახვა-დამუშავება რეალურ დროში, რაც ტრადიციულ რელაციურ მონაცემთა ბაზებისთვის გარკვეულ სირთულეებს წარმოადგენს.

აღნიშნული სირთულეების დასაძლევად შეიქმნა და დღემდე აქტიურად ვითარდება „დიდ მონაცემთა“ (Big Data) ტექნოლოგიები, რომლებიც საშუალებას იძლევა, რეალურ დროში დავამუშაოთ დიდი რაოდენობის ინფორმაცია. ამ რაოდენობის მონაცემების დამუშავებას სრულიად განსხვავებული მიდგომა სჭირდება არა მხოლოდ მონაცემთა ბაზებისა და სერვერული ინფრასტრუქტურის პროექტირების, არამედ პროგრამული უზრუნველყოფის დონეზეც. სადისერტაციო ნაშრომის მეორე თავში შემუშავებულია კრიტიკული სისტემის ინფრასტრუქტურა და დაპროექტებულია მაღალი წვდომადობის მქონე მონაცემთა საცავი.

დიდი რაოდენობით დაგროვებულმა ინფორმაციამ მოგვცა საშუალება, განვვითარებულიყავით მანქანური დასწავლის (Machine Learning) მიმართულებით. ჯერ კიდევ 1959 წელს ამერიკელმა მეცნიერმა არტურ სამუელმა (Arthur Samuel) მანქანური დასწავლა განმარტა როგორც „მეცნიერების სფერო, რომელიც კომპიუტერებს აძლევს თვითდასწავლის შესაძლებლობას, ცხადად დაპროგრამების გარეშე“. დღეს კი უკვე ხელოვნური ინტელექტი მართავს თვითმფრინავს, დახვეწილია სახეთა ამოცნობის ალგორითმები და თუნდაც, თითქმის არასდროს გვხვდება spam ტიპის წერილები ელექტრონულ საფოსტო ყუთში.

სამწუხაროდ, ზემოთ აღნიშნული მიმართულებებით საქართველოში მხოლოდ ერთეულები სარგებლობენ, რაც გამოწვეულია ამ ტექნოლოგიების სირთულით - ტექნოლოგიების დანერგვისთვის საჭიროა მრავალმხრივი გამოცდილება და დიდი რაოდენობის სერვერული რესურსი.

სადოქტორო ნაშრომის მიზნად დასახული მაქვს გადაუდებელი დახმარების ოპერატიული მართვის ცენტრისთვის „112“ მონაცემთა საცავის არქიტექტურის შემუშავება და ზემოთ ჩამოთვლილი ტექნოლოგიების საშუალებით ორი რეალური პრობლემის მოგვარების გზების გამოვლენა:

- **სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანა** - გამოძახების შემთხვევაში ისეთი ბრიგადის პოვნა, რომელიც ყველაზე სწრაფად მოახდენს რეაგირებას. მაგალითად, ზარის ინიციატორთან ტერიტორიულად უახლოესი თავისუფალი ბრიგადა საცობის გათვალისწინებით 1 საათის სავალზეა, სხვა საქმეზე გასული ბრიგადა კი შეიძლება ინიციატორის მეზობლად იყოს და 5 წუთში გათავისუფლდეს. არაოპტიმალური მართვით დროც ბევრად მეტი იხარჯება და ფინანსური თუ ადამიანური რესურსიც.

ეს ამოცანა განხილულია დისერტაციის მესამე თავში, სადაც მიზნად დასახულია ხელოვნური ნეირონული ქსელის ზოგადი მოდელის შექმნა და სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანის გადაწყვეტის ეტაპების დაგეგმვა. შემუშავებული და გაანალიზებულია მანქანური დასწავლისთვის საჭირო სტრუქტურის მქონე მონაცემთა კრებული.

- **სასწრაფო დახმარების გამოძახებების პრიორიტეტების განსაზღვრის ამოცანა** - სასწრაფო სამედიცინო მომსახურების რიგითობის განსასაზღვრად დისპეტჩერს უწევს გამოძახებების სათითაოდ შემოწმება და შედარება, რაც დიდ დროსა და ძალისხმევას მოითხოვს მაშინ, როდესაც წამები გადამწყვეტია. ამასთან, გამოძახებების რაოდენობა არსებულ ბრიგადებთან შედარებით ბევრად მეტია და ხშირად ერთი და იმავე ტიპის ათობით „ჩამოკიდებული“ საქმე ელოდება ბრიგადის გათავისუფლებას. აღნიშნული პრობლემა განსაკუთრებით აქტუალურია გრიპის ეპიდემიის დროს.

პრიორიტეტების განსაზღვრის ამოცანის გადასაწყვეტად მიზნად დავისახე შემექნა ალგორითმების სისტემა შემოსული საქმეების გადაუდებლობის ხარისხის განსაზღვრისათვის.

შემუშავებული სისტემა განხილულია სადისერტაციო ნაშრომის მეოთხე თავის ექსპერიმენტულ ნაწილში.

1.7. პირველი თავის დასკვნა

მონაცემთა რელაციური და არარელაციური ბაზების მახასიათებლების შედარების საფუძველზე განსაზღვრულია იმ ობიექტების კლასი, რომლებისთვისაც ეფექტიანია მონაცემთა არარელაციური ბაზების გამოყენება.

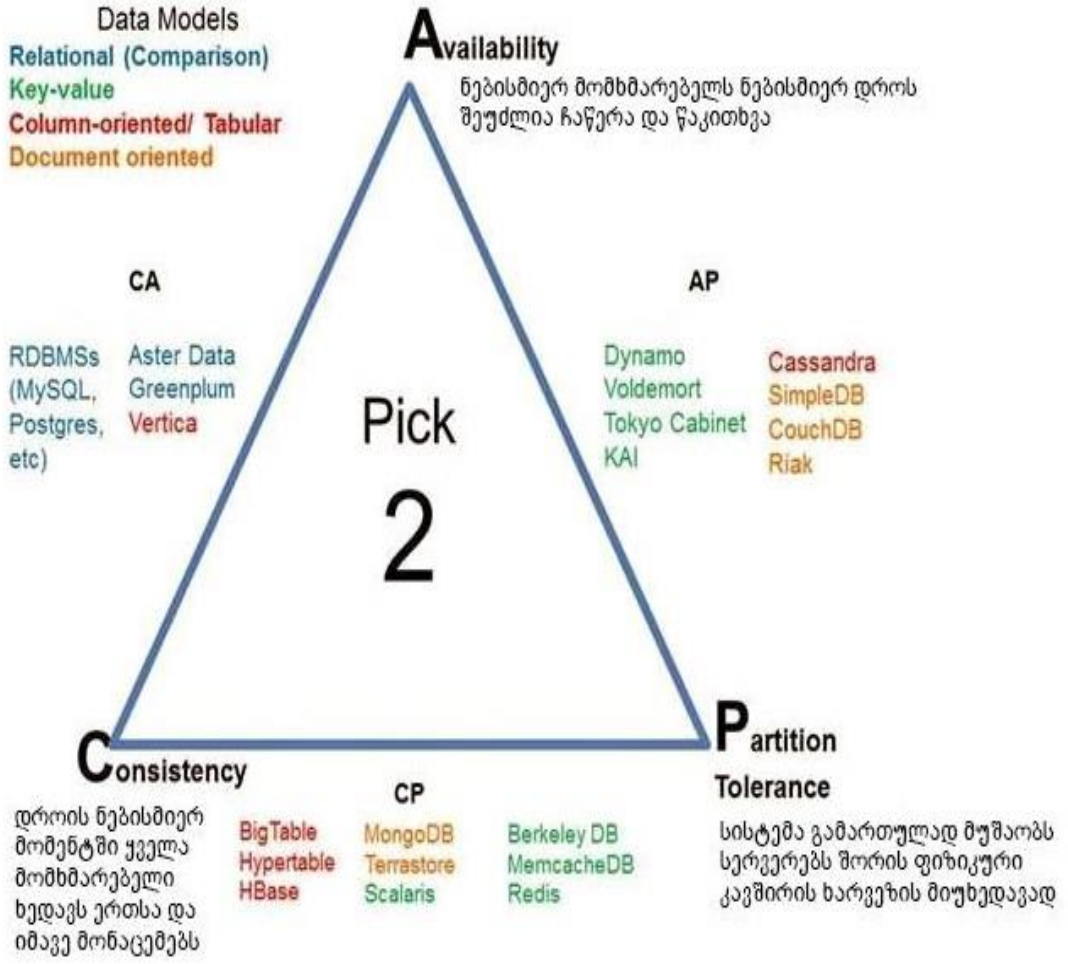
განხილული Hadoop პლათფორმის მეშვეობით შესაძლებელია მონაცემთა საცავის წარმადობის მნიშვნელოვნად გაუმჯობესება.

მანქანური დასწავლისა და მისი ქვე-მიმართულებების ტექნოლოგიების მიმოხილვის საფუძველზე გადაწყვეტილია დაპროგრამების ენის Python გამოყენება სადისერტაციო ნაშრომში დასმული ამოცანების გადასაწყვეტად.

2 თავი. მონაცემთა საცავის დაპროექტება და სისტემის ინფრასტრუქტურა

2.1. CAP სამკუთხედის თეორემა

CAP თეორემა (Consistency, Availability, Partition_tolerance) ან ბრიუერის (*Brewer Eric*) თეორემა არის ევრისტიკული მტკიცება იმის შესახებ, რომ განაწილებული გამოთვლების ნებისმიერ რეალიზაციაში შესაძლებელია სამი (C,A,P) თვისებიდან მხოლოდ ორის უზრუნველყოფა (ნახ.10) [12].



ნახ. 10. CAP თეორემა - მონაცემთა ბაზების მხარდაჭერის მიხედვით

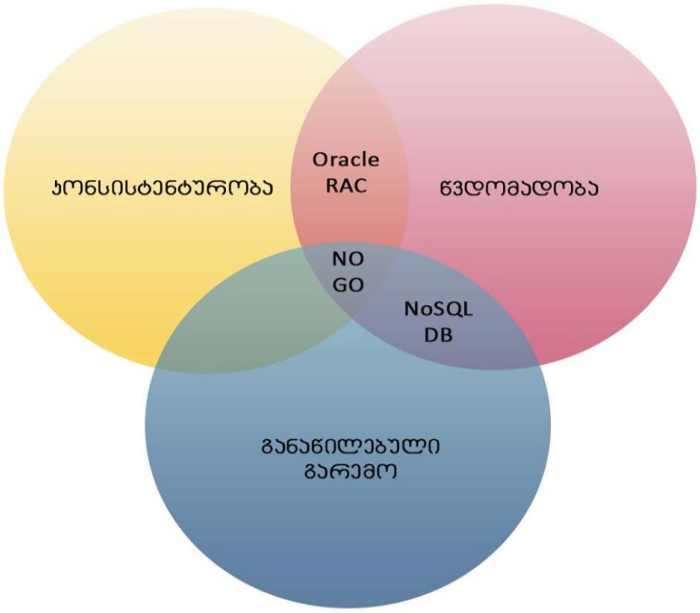
Consistency (მთლიანობა) – განაწილებული კომპიუტერული სისტემის ყველა კვანძში (node – კლასტერში) მონაცემები შეთანხმებულია (არაწინააღმდეგობრივია დროის მოცემულ მომენტში) [31];

Availability (წვდომადობა) – ნებისმიერი მოთხოვნა განაწილებულ სისტემასთან სრულდება კორექტული პასუხით (თუ რომელიმე კვანძი გათიშულია, მას ცვლის სხვა);

Partition Tolerance (განაწილების მდგრადობა) – კლასტერი ფუნქციონირებას განაგრძობს მაშინაც კი როდესაც კვანძებს შორის კავშირის პრობლემაა.

ქვემოთ აღწერილია CAP სამკუთხედის შესაძლო კომბინაციები (ნახ.11). როდესაც კომუნიკაციის პრობლემაა, არჩევანი გვაქვს:

- დავუშვათ არასინქრონირებული მონაცემების არსებობა მონაცემთა ბაზაში (შევლიოთ კონსისტენტურობას);
- ჩავთვალოთ რომ კლასტერი დაზიანდა, ანუ შევლიოთ წვდომადობას.



ნახ. 11. CAP თეორემა

CAP თეორემის თვალსაზრისით, ამ სამი თვისებიდან ერთდროულად მხოლოდ რომელიმე ორის განხორციელება შეიძლება. განაწილებული სისტემები იყოფა სამ კლასად:

CA (მთლიანობა და წვდომადობა) – ყველა კვანძში ერთი და იგივე მონაცემებია. თუ კვანძებს შორის კომუნიკაციის დაფიქსირდა, მონაცემები იქნება აცდენილი მანამ, სანამ პრობლემა არ გადაიჭრება და არ მოხდება სინქრონიზაცია.

CP (მთლიანობა და განაწილების მდგრადობა) – კვანძებში მონაცემები არაწინააღმდეგობრივია. კომუნიკაციის პრობლემის დროს თავიდან რომ ავიცილოთ არასინქრონული მონაცემების არსებობა, შესაბამისი კვანძი გაუქმდება - შეწყდება ჩაწერა/წაკითხვის ოპერაციების შესრულება.

AP (წვდომადობა და განაწილების მდგრადობა) – არაა გარანტირებული მონაცემთა მთლიანობა. კვანძებს შორის კომუნიკაციის პრობლემის შემთხვევაში დაზიანებული კვანძები დამოუკიდებლად განაგრძობენ მუშაობას. კომუნიკაციის პრობლემის აღდგენის შემდეგ დაიწყება მონაცემების სინქრონიზაცია, თუმცა კვანძებს შორის მონაცემების კონსისტენტურობის გარანტია აღარ გვექნება [7].

კალიფორნიის უნივერსიტეტის პროფესორის ერიკ ბრიუერის მიერ 2000 წელს შემოთავაზებული CAP თეორემა პოპულარული გახდა განაწილებული კომპიუტერული სისტემების სფეროში. NoSQL ბაზების კონცეფცია, რომლითაც იქმნება განაწილებული არატრანზაქციული მონაცემთა ბაზების სისტემები, ხშირად მონაცემთა მთლიანობის (consistency) დარღვევის გარდაუვალობის დასაბუთებისთვის სწორედ ამ პრინციპს იყენებს [32].

NoSQL სისტემების უმრავლესობა მონაცემთა ბაზის მთლიანობის გარანტიას არ იძლევა. ამიტომაც AP-სისტემების აგების ამოცანა მდგომარეობს მონაცემთა მთლიანობის გარკვეული, პრაქტიკულად მიზანშეწონილი დონის უზრუნველყოფის განხორციელებაში. ასეთი AP-

სისტემები ლიტერატურაში მოიხსენიება ტერმინით „სუსტი მთლიანობის“ (weak consistent) სისტემები [33,34].

2.2. მთლიანობის მოდელები CAP თეორემაში

მონაცემთა ბაზის მთლიანობაზე საუბრისას განვიხილავთ სცენარს, როდესაც სხვადასხვა კვანძებზე ერთი და იგივე მონაცემების დამოუკიდებელი კოპიოები გვაქვს. კონფლიქტური სიტუაცია დგება, როდესაც კონკრეტული კვანძის მომხმარებელი სერვერისგან მოითხოვს გარკვეული ჩანაწერის ცვლილებას. რაც ნიშნავს, რომ სანამ ცვლილება სხვა კვანძებზეც აისახება, დროის გარკვეულ მონაკვეთში ახალი მომხმარებლები სხვადასხვა კვანძებიდან სხვადასხვა მონაცემებს (მონაცემების სხვადასხვა ვერსიებს) მიიღებენ.

სწორედ აქ ჩნდება განაწილებულ გარემოში მთლიანობის სხვადასხვა მოდელები:

ძლიერი მთლიანობა (Strong Consistency) : W + R > N

სუსტი მთლიანობა (Weak / Eventual Consistency) : W + R <= N

კითხვაზე ოპტიმიზირებული: R = 1, W = N

ჩაწერაზე ოპტიმიზირებული: W = 1, R = N

სადაც R წაკითხვის მოთხოვნების დამუშავებაზე პასუხისმგებელი კვანძების მაქსიმალური რაოდენობაა.

W ჩაწერის მოთხოვნების დამუშავებაზე პასუხისმგებელი კვანძების მაქსიმალური რაოდენობაა.

N კი იმ კვანძების რაოდენობას აღნიშნავს, სადაც მონაცემები დუბლირდება.

მაგალითად, განვიხილოთ კლასტერი 5 კვანძით, R, W და N მნიშვნელობების ცვლილებით შეგვიძლია ვაკონტროლოთ მონაცემთა მთლიანობის დონეები. თუ კლასტერს ავაწყობთ პარამეტრების R = 5 და N = 5 მნიშვნელობებით, მივაღწევთ მთლიანობის მაქსიმალურ დონეს, - ყველა კვანძზე გვექნება დუბლირებული მონაცემები, სამაგიეროდ, ჩვენი სისტემა აღარ იქნება საიმედო განაწილებული გარემოს კუთხით, რადგან არცერთი

ჩაწერა არ ჩაითვლება წარმატებულად რომელიმე კვანძის ჩავარდნის შემთხვევაში.

შეგვიძლია იგივე კლასტერი ავაწყოთ პარამეტრების R = 1 და N = 1 მნიშვნელობებით. ამ შემთხვევაში მონაცემების ცვლილების ოპერაცია წარმატებულად ჩაითვლება მინიმუმ ერთ კვანძზე წარმატებული ჩაწერის შემდეგ, მაგრამ ზოგიერთ კვანძს აღარ ექნება ბოლოს შეცვლილი მონაცემების მიმდინარე ვერსია.

მთლიანობის მსგავსად, შეგვიძლია წვდომადობასაც ვაკონტროლოთ წვდომის მაქსიმალური დასაშვები დროის ცვლილებით [34].

2.2.1. AP სისტემები - ძლიერი მთლიანობის მოდელი

როგორც ზემოთ აღვნიშნეთ, ძლიერი მთლიანობა მიიღწევა როდესაც ინფორმაციის დაცვის მიზნით, მონაცემებს ყველა კვანძზე პარალელურ რეჟიმში ვანახლებთ, რითიც ვიღებთ გარანტიას, რომ დროის ნებისმიერ მომენტში შევძლებთ მონაცემების მიმდინარე ვერსიის წაკითხვას, იმის მიუხედავად, თუ რომელ კვანძზე ჩაიწერა მონაცემი თავდაპირველად. ამ გარანტიას ძლიერი მთლიანობის მოდელი იმის ხარჯზე გვაძლევს, რომ სანამ რომელიმე მომხმარებლის მიერ შეტანილი ცვლილება ყველა კლიენტისთვის ხილვადი გახდება, ახალმა ცვლილებამ ყველა კვანძზე უნდა გაიაროს ვალიდაცია. ცვლილების შემტანი მომხმარებელი კი მოთხოვნაზე პასუხს მხოლოდ მას შემდეგ მიიღებს, რაც ეს ცვლილება ყველა კვანძზე აისახება.

ძლიერი მთლიანობის საჭიროება ნათლად ჩანს საბანკო სისტემებში. თანხის ერთი ანგარიშიდან მეორეზე გადატანის დროს. ცხადია, ტრანზაქციის დაწყებამდეც და დასრულების შემდეგაც ჯამური ბალანსი უცვლელი უნდა დარჩეს. ასეთივე მოთხოვნები აქვს ყველა იმ ამოცანას, სადაც ინფორმაციასთან წვდომა რეალურ დროში გვჭირდება [34].

2.2.2. AP სისტემები - სუსტი მთლიანობის მოდელი

ძლიერი მთლიანობა (strong consistency) და მაღალი წვდომადობა (high availability) სისტემის სასურველი თვისებებია, მაგრამ CAP თეორემა

გვიჩვენებს, რომ კვანძებს შორის არასაიმედო კავშირის შემთხვევაში ერთდროულად ორივეს მიღწევა შეუძლებელია და მიგვანიშნებს ცხოვრებისეულად უფრო რეალური „სუსტი მთლიანობის“ მოდელისკენ.

სუსტი მთლიანობის მოდელში მონაცემების ყველა კვანძზე რეალურ დროში განახლება არ არის აუცილებელი, მაგრამ დარწმუნებით შეგვიძლია ვთქვათ, რომ საბოლოოდ ინფორმაცია ყველა კვანძზე განახლდება. მონაცემების წაკითხვისას კი კვანძი დაგვიბრუნებს ჩანაწერის იმ ვერსიას, რომელსაც პირველად იპოვის [35].

სუსტი მთლიანობის გამოყენება ზოგიერთ ამოცანაში იმდენად ზრდის წარმადობასა და ჰორიზონტალურ სკალირებას, რომ ეს მოდელი შესაბამისი პრობლემების გადასაწყვეტად შეუცვლელია. ამგვარი ამოცანებია გამომწერების სია Twitter-ში, მეგობრების სია Facebook-ში და სამეცნიერო პროგრამის log ჩანაწერები.

სუსტი მთლიანობის მოდელი წარმატებული იქნება ყველა იმ კლასის ამოცანაში, სადაც ზუსტი, რეალური დროის პასუხის დაბრუნებაზე უფრო მნიშვნელოვანი უბრალოდ პასუხის დაბრუნებაა.

თუ მონაცემთა ბაზის მოდელი და სქემა სწორად არის შერჩეული, მაშინ, მიუხედავად მონაცემთა ბაზის ზომებისა და მასში არსებული ჩანაწერების რაოდენობისა, მონაცემებთან წვდომა თითქმის მყისიერია. მილიონობით ჩანაწერში ჩვენთვის საინტერესო ჩანაწერის ზუსტი და მყისიერი მოძებნა გამაოგნებელ შთაბეჭდილებას ახდენს.

და ისევ, ისეთ ამოცანებში, სადაც შეხება გვაქვს ფინანსურ ტრანზაქციებსა და რეალურ დროში დასამუშავებელ მონაცემებთან, მოძველებულმა ინფორმაციამ შესაძლოა წარმოშვას მნიშვნელოვანი რისკები. ამიტომ, ამ კლასის ამოცანებში სუსტი მთლიანობის მოდელის გამოყენება არ არის რეკომენდირებული.

2.3. ტრანზაქციის იზოლირების დონეები

მონაცემთა ბაზების მწარმოებლებს ტრანზაქციის იზოლირების დონეები განსხვავებებულად აქვთ გადაწყვეტილი. მაგალითად, Oracle

მონაცემთა ბაზაში მონაცემების კითხვის დროს lock საერთოდ არ გამოიყენება და დამახინჯებული მონაცემების (dirty data) პრობლემა latch-ებით აქვს გადაწყვეტილი.

Lock-ის შემთხვევაში მომხმარებელი რიგში დგება, რომ მონაცემებზე წვდომა მიიღოს, latch-ის დროს კი პერიოდულად ხდება მონაცემების შემოწმება - არის თუ არა თავისუფალი.

Oracle, MsSQL Server და DB2-ში იზოლაციის დონის მითითება შესაძლებელია კონკრეტული სესიის ან ტრანზაქციის დონეზე (DB2-ს დამატებით აქვს აპლიკაციის დონეც) [31,34].

NoSQL სისტემებს, კონკრეტულად კი MongoDB-ს ტრანზაქციების იზოლირების მსგავსი მეთოდები არ აქვს [5]. ამ პრობლემას MongoDB თითოეული დოკუმენტის ატომარობის პრინციპის დაცვით აგვარებს. მონაცემთა ბაზის დიზაინის შემუშავებისას უნდა იქნას გათვალისწინებული, რომ მონაცემები, სადაც გვჭირდება ატომარობის პრინციპის დაცვა, უნდა ინახებოდეს ერთ დოკუმენტში. როგორც წესი, დოკუმენტის დონეზე არსებული ატომარობა საკმარისია იმ საიმედოობის მისაღწევად, რასაც მონაცემთა რელაციური ბაზები ACID პრინციპების დაცვით გადაწყვეტდნენ [10].

მაგალითად, MongoDB მონაცემთა ბაზაში ერთმანეთთან დაკავშირებული მონაცემები შეგვიძლია მოვათავსოთ საერთო დოკუმენტში, ჩადგმული მასივის (nested array) ან ჩადგმული დოკუმენტის (nested document) გამოყენებით. ამ შემთხვევაში, ცვლილების დროს საკმარისი იქნება მხოლოდ ერთი დოკუმენტის განახლება, ეს კი უკვე ატომარული პროცესი იქნება. მონაცემთა რელაციურ ბაზებში იმავე სახის დაკავშირებული მონაცემების წარმოსადგენად საჭირო გახდებოდა სხვადასხვა ცხრილებისა და ჩანაწერების გამოყენება, რაც მონაცემების ცვლილებისთვის აუცილებელს გახდიდა ტრანზაქციის იზოლირების დონეების გამოყენებას.

2.4. მონაცემების დუბლირება – replication

რეპლიკაცია არის პროცესი, რომლის დროსაც მონაცემები დუბლირდება ერთი, მთავარი სერვერიდან რამდენიმე სათადარიგო სერვერზე. რეპლიკაციის დროს მთავარ (primary) სერვერზე ჩაწერისა და წაკითხვის ოპერაციების განხორციელება შეგვიძლია, სათადარიგო სერვერებიდან კი მხოლოდ ვკითხულობთ. რეპლიკაციას არაერთი დადებითი მხარე გააჩნია:

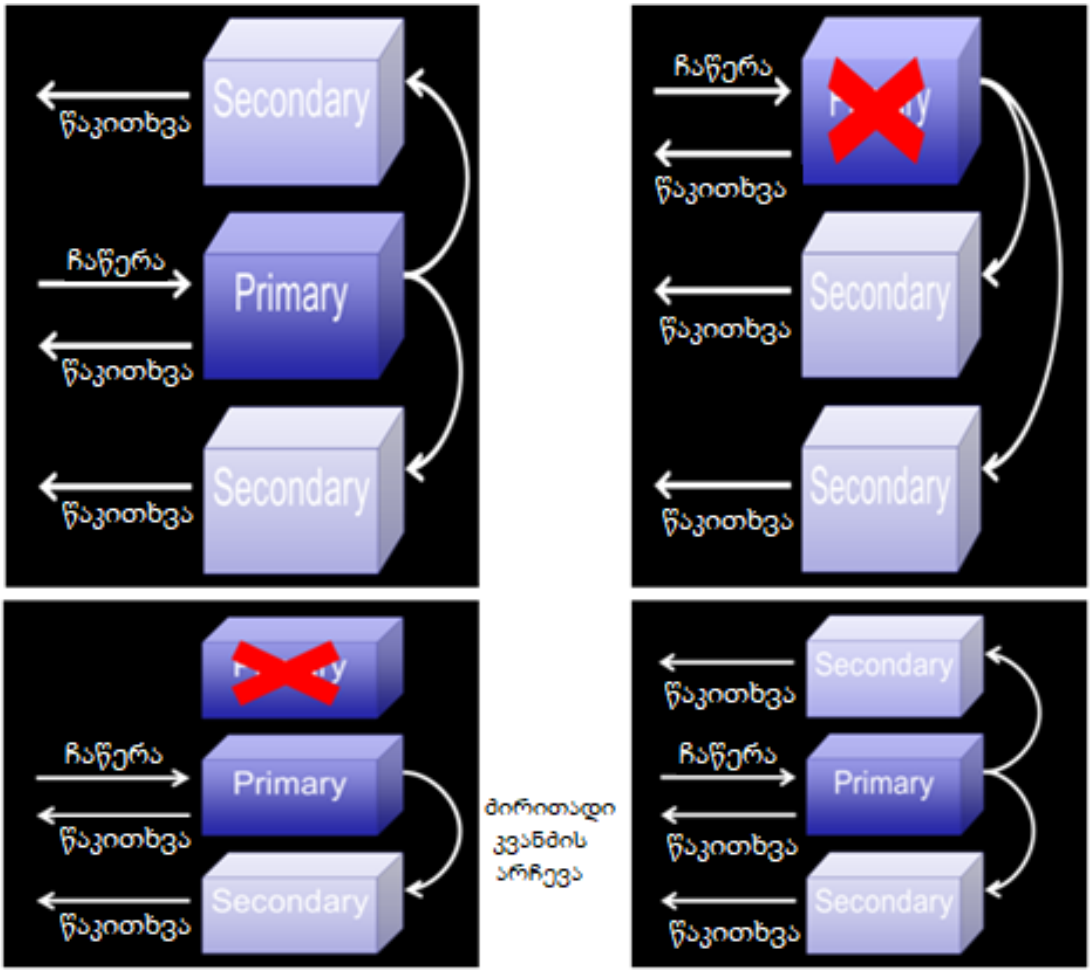
- მონაცემების კითხვის გაუმჯობესება – რამდენ კოპიო სერვერსაც დავაკონფიგურირებთ, იმდენი დამოუკიდებელი წყაროდან შეგვიძლია ინფორმაციის პარალელურ რეჟიმში წაკითხვა;
- Disaster Recovery – პრობლემის ან გეგმიური სამუშაოების დროს შეგვიძლია რომელიმე სათადარიგო (standby) სერვერი ძირითად (primary) სერვერად გადავაქციოთ;
- backup-ებისა და ინდექსების რეორგანიზაციის დროს აღარ ხდება საჭირო მონაცემთა ბაზის გაჩერება ან შენელება;
- რეპლიკა სეტები აპლიკაციის დონეზე არ ჩანან.

მთავარ სერვერთან კავშირის დაკარგვის ან გათიშვის შემთხვევაში სათადარიგო სერვერებზე ეწყობა არჩევნები და ირჩევა ახალი primary სერვერი.

რეპლიკა სეტების არჩევნებში მონაწილეობას ღებულობს წინასწარ განსაზღვრული სერვერები – არბიტრები. არბიტრი არ შეიცავს ინფორმაციას მომხმარებლებზე, მისი ერთადერთი დანიშნულება არის არჩევნებში ხმის მიცემა.

თუ Replica set-ში კვანძების ლუწი რაოდენობა გვაქვს, არბიტრის დამატებით ვიღებთ კენტი რაოდენობის წევრებს, რითაც შესაძლებელი ხდება არჩევნებში ხმათა უმრავლესობის მიღწევა.

ყველაზე ხშირად რეპლიკაცია სამი რეპლიკა სეტისგან ეწყობა (ნახ.12).



ნახ. 12. რეპლიკაციის პროცესი

რეპლიკაციის ასაწყობად მთავარ სერვერზე უნდა შეიქმნას რეპლიკაციის გარემო:

mongod --port "PORT_NUMBER" --dbpath "მონაცემთა ბაზის მისამართი" --replSet "რეპლიკაციის კვანძის სახელი";

შემდეგ უნდა დავუკავშირდეთ უკვე შექმნილ გარემოს და გავუშვათ რეპლიკაციის პროცესი: rs.initiate().

რეპლიკა სეტის კონფიგურაციის პარამეტრებისა და სტატუსის გამოტანა შეგვიძლია ბრძანებებით rs.conf() და rs.status().

ახალი წევრების დასამატებლად ვუშვებთ ბრძანებას:

rs.add(HOST_NAME:PORT_NUMBER)

სტანდარტულად, კითხვისთვის მომხმარებლები ძირითად სერვერს მიმართავენ, მაგრამ აპლიკაციიდან შეგვიძლია ავირჩიოთ „Read Preference“ და კითხვა განვახორციელოთ მეორადი კვანძიდან.

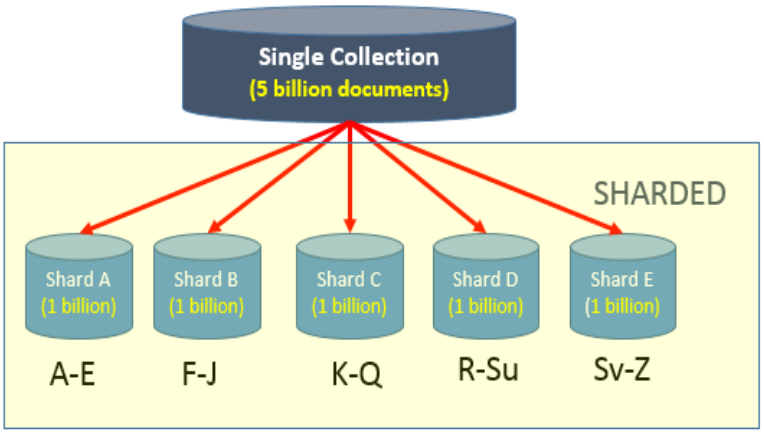
Read Preference პარამეტრი საგრძნობლად აუმჯობესებს მონაცემთა ბაზის წარმადობას, მაგრამ ცხადია, პრობლემა ხდება კონსისტენტურობა, – სანამ ძირითად(primary) სერვერზე განხორციელებული ცვლილება სათადარიგო(secondary) ბაზაზეც აისახება, კლიენტები ძველ ინფორმაციას კითხულობენ. ეს პრობლემა გადაჭრილია majority პარამეტრის შემოღებით. ამ პარამეტრის მიხედვით ეთითება, მინიმუმ რამდენ კვანძზე უნდა ჩაიწეროს ინფორმაცია ტრანზაქციის წარმატებით დასასრულებლად. ძირითად სერვერზე ინფორმაციის რამდენიმე ვერსია გვექნება, მაგრამ მომხმარებლებს მიეწოდება მხოლოდ ის ვერსია, რომელიც დასრულებულია, ანუ ჩაწერილია სათადარიგო სერვერების საჭირო რაოდენობაზე.

2.5. განაწილებული გარემო – sharding

Sharding ტექნოლოგია მონაცემთა ბაზის დანაწევრების (database partitioning) ერთერთი სახეა. Shard - სეგმენტი ერთი მთლიანის მცირე ნაწილს ნიშნავს, – აღნიშნული ტექნოლოგიის მეშვეობით დიდი ზომის უმართავ ბაზას ვყოფთ პატარა, სწრაფ და ადვილად მართვად ნაწილებად.

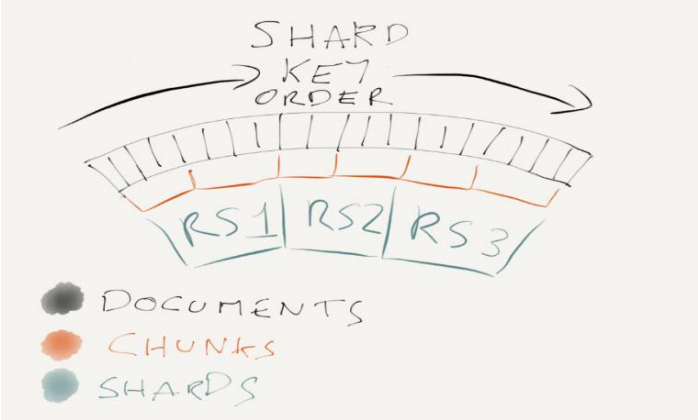
განაწილებული გარემოს აწყობის დროს მთავარი სირთულე shard key გასაღების სწორად არჩევაა. ამ გასაღების მიხედვით ნაწილდება დოკუმენტები შესაბამის კვანძებზე.

მაგალითისთვის შეგვიძლია მოვიყვანოთ 5 მილიარდი ჩანაწერის მქონე კოლექცია, რომელიც გადანაწილებულია 5 სხვადასხვა Shard სერვერზე, გასაღებად კი აღებულია ტექსტური ველი. გასაღების ამგვარად არჩევის შემთხვევაში მარტივად შეგვიძლია ახალი Shard-ის დამატება რომელიმე ძველ Shard-ზე ინფორმაციის გადანაწილებით (ნახ.13).



ნახ. 13. ინფორმაციის გადანაწილება Sharding ტექნოლოგიით

დოკუმენტები ერთიანდება ფიქსირებული ზომის ბლოკებში (chunk), რომლის სტანდარტული ზომა 64 მბ არის (ცვლილება შესაძლებელია). თავად ბლოკები კი სეგმენტებში (shard) ერთიანდება (ნახ.14).



ნახ. 14. Sharding: დოკუმენტებისა და ბლოკების ორგანიზება

მონაცემების გაზრდის შემდეგ ახალი სერვერის დამატებისთვის, არსებული shard-ებიდან ახალ სერვერზე უნდა გადმოვანაწილოთ ახალი დაყოფის შესაბამისი ბლოკები.

სეგმენტების ბალანსირების პროცესს mongoDB ავტომატურად, მომხმარებლებისგან დამოუკიდებლად ასრულებს.

ასევე, მომხმარებლისგან დამოუკიდებლად წყდება, თუ რომელი shard კვანძებია პასუხისმგებელი კონკრეტული მოთხოვნის შესრულებაზე. ამისთვის გამოიყენება მოთხოვნების მარშრუტიზატორი (query router), რომელიც მომხმარებლისგან იღებს მოთხოვნას, წინასწარ შენახული

დამატებითი ინფორმაციის (metadata) მიხედვით მიაკითხავს შესაბამის shard სერვერებს და მომხმარებელს მონაცემების მცირე ულუფებისგან აწყობილ შედეგს უბრუნებს.

2.6. სერვერული უზრუნველყოფის გამართვა დუბლირებული მონაცემებისა და განაწილებული გარემოს გამოყენებით

ლინუქს ოპერაციულ სისტემაზე MongoDB მონაცემთა ბაზის გასამართად სავალდებულოა შემდეგი წინაპირობების შესრულება:

1) გამართული უნდა იყოს ლინუქსის ოპერაციული სისტემა ფიზიკურ ან ვირტუალურ მანქანაზე (2 ბირთვი , 2 GB ოპერატიული მეხსიერება, 20 GB მყარი დისკის მოცულობა). რაც შეეხება Linux-ის დისტრიბუციას, შეგვიძლია გამოვიყენოთ ნებისმიერი, მაგალითად სერვერებისთვის ერთ-ერთი ყველაზე პოპულარული სისტემა CentOS [36,37];

2) ბაზის დაყენების დროს ოპერაციულ სისტემას უნდა ჰქონდეს ინტერნეტზე წვდომა.

ოპერაციული სისტემის გამართვის და ინტერნეტზე წვდომის გახსნის შემდეგ ვაყენებთ მონაცემთა ბაზას (ჩვენს შემთხვევაში ოპერაციული სისტემაა CentOS 6 64-bit) .

1) სისტემაში ავტორიზაციას გავდივართ root მომხმარებლით

2) touch /etc/yum.repos.d/mongodb.repo ვქმნით რეპოზიტორის კონფიგურაციის ფაილს

3) vi /etc/yum.repos.d/mongodb.repo vi ედიტორის საშუალებით mongodb.repo ფაილში ჩავწეროთ პარამეტრები:

- [mongodb]
- name=MongoDB Repository
- baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64/
- gpgcheck=0
- enabled=1

4) yum –y install mongo-10gen mongo-10gen-server (მონაცემთა ბაზის სერვისების დაყენება)

5) service mongod start (მონაცემთა ბაზის მთავარი სერვისის გაშვება)

6) service mongod status (მონაცემთა ბაზის სტატუსის შემოწმება)

7) mongostat (მონაცემთა ბაზის სტატუსისა და მიმდინარე პროცესების ნახვა)

MongoDB მონაცემთა ბაზის დაყენების შემდეგ, იმისათვის, რომ მონაცემთა ბაზაში მუშაობა შევძლოთ, ოპერაციული სისტემის ტერმინალში უნდა გავუშვათ ბრძანება mongo (მონაცემთა ბაზის ტერმინალის გახსნა), სადაც უშუალოდ mongo-ს შიდა ბრძანებების შესრულებას შევძლებთ. mongo (მონაცემთა ბაზის კლიენტი) ბრძანების გაშვებისას ოპერაციული სისტემა ავტომატურად მიმართავს „localhost:27017“ და ცდის ბაზასთან დაკავშირებას.

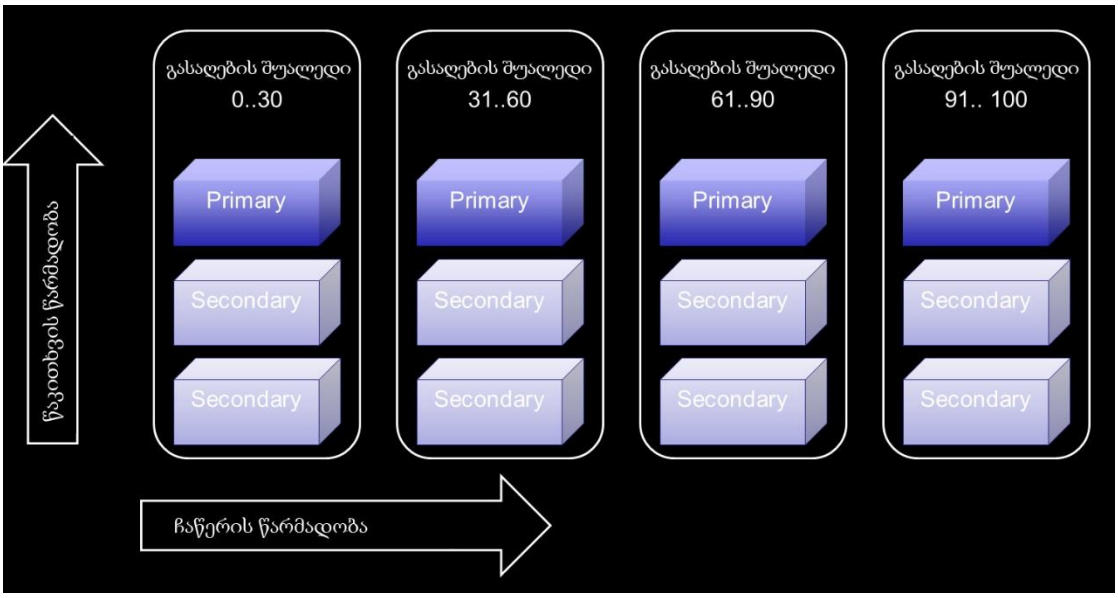
მონაცემთა ბაზების წარმადობა დამოკიდებულია სამ ძირითად მახასიათებელზე: პროცესორი, ოპერატიული მეხსიერება და მყარი დისკი.

შარდინგისა და რეპლიკაციის დროს კი მოთხოვნის ზრდასთან ერთად შესაძლებელია ასობით და ათასობით ახალი მანქანის დამატება და თითოეული ახალი მანქანა გააუმჯობესებს სამივე ძირითად მახასიათებელს (ნახ.15).

ხშირად, ადმინისტრატორები მონაცემთა ბაზებს საჭიროზე მეტ ოპერატიულსა და პროცესორებს უყოფენ, ამ დროს ბოთლის შევიწროებული ყელის (bottleneck) პრობლემას ვაწყდებით დისკების სიმცირეში. სერვერი დიდ დროს კარგავს დისკიდან ინფორმაციის ამოკითხვაზე.

ეს პრობლემა დუბლირებული კვანძების გამოყენებისას მინიმუმამდეა დაყვანილი:

- გვჭირდება სწრაფი კითხვა? – ვამატებთ ახალ რეპლიკა სეტს, ანუ მონაცემთა ბაზის ახალ ასლს.
- გვჭირდება სწრაფი ჩაწერა/წაკითხვა? – შარდინგ ტექნოლოგიით მონაცემთა ბაზას უფრო მცირე ნაწილებად ვყოფთ, შესაბამისად, ერთდროულად უფრო მეტი დისკის რესურსს ვიყენებთ ინფორმაციის ჩასაწერად და წასაკითხად.



ნახ. 15. სერვერული უზრუნველყოფა დუბლირებული მონაცემებისა და განაწილებული გარემოს (replication + sharding) გამოყენებით

საჩვენებლად ავაწყოთ 4 რეპლიკა სეტი, რომლებიც შარდინგ ტექნოლოგიით ინაწილებენ რესურსს (ნახ.14). სასწავლო რეჟიმში ყველა სერვისი ერთ კომპიუტერზე იყო გაშვებული და განსხვავებულ პორტებზე მუშაობდა. რეალურ შემთხვევაში მონგოს სერვისები გაშვებული გვაქვს სხვადასხვა სერვერზე.

A რეპლიკა სეტი

mkdir Tbilisi_A 192.168.0.10

mkdir Rustavi_A 192.168.1.10

mkdir Kutaisi_A 192.168.2.10

B რეპლიკა სეტი

mkdir Tbilisi_B 192.168.0.11

mkdir Rustavi_B 192.168.1.11

mkdir Kutaisi_B 192.168.2.11

C რეპლიკა სეტი

mkdir Tbilisi_C 192.168.0.12

mkdir Rustavi_C 192.168.1.12

mkdir Kutaisi_C 192.168.2.12

D რეპლიკა სეტი

```
mkdir  Tbilisi_D    192.168.0.13
mkdir  Rustavi_D    192.168.1.13
mkdir  Kutaisi_D    192.168.2.13
// mkdir ბრძანებებში იგულისხმება შესაბამის სერვერებზე
// რეპლიკა სეტის, საქაღალდეების, datafile-ებისა და
// მთლიანად მონაცემთა ბაზისთვის საჭირო
//  დირექტორიების შექმნა
mkdir  config0    192.168.0.14    // კონფიგ–სერვერების datafile-ები
mkdir  config1    192.168.1.14
mkdir  config2    192.168.2.14
# config servers    კონფიგ–სერვერების კონფიგურაცია
mongod   --configsvr  --dbpath config0  --port 26050  --fork  --logpath log.config0
--logappend
mongod   --configsvr  --dbpath config1  --port 26051  --fork  --logpath log.config1
--logappend
mongod   --configsvr  --dbpath config2  --port 26052  --fork  --logpath log.config2
--logappend
```

რეპლიკა სეტის გასამართად შესაბამის A,B,C,D[0,1,2] სერვერებზე სათითაოდ ვუშვებთ შესაბამის სერვისებს:

```
# შარდ-სერვერებისთვის oplogsize პარამეტრის მცირე მნიშვნელობა და --
smallfiles რეკომენდებულია მხოლოდ სასწავლო სისტემებისთვის.
mongod --shardsvr --replSet a --dbpath Tbilisi_A --logpath log.Tbilisi_A --port
27000 --fork --logappend --smallfiles --oplogSize 1000
mongod --shardsvr --replSet a --dbpath Rustavi_A --logpath log.Rustavi_A --port
27001 --fork --logappend  --smallfiles --oplogSize 1000
mongod --shardsvr --replSet a --dbpath Kutaisi_A --logpath log.Kutaisi_A --port
27002 --fork --logappend  --smallfiles --oplogSize 1000
```

mongod --shardsvr --replSet b --dbpath Tbilisi_B --logpath log.Tbilisi_B --port 27100 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet b --dbpath Kutaisi_B --logpath log.Kutaisi_B --port 27101 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet b --dbpath Kutaisi_B --logpath log.Kutaisi_B --port 27102 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet c --dbpath Tbilisi_C --logpath log.Tbilisi_C --port 27200 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet c --dbpath Rustavi_C --logpath log.Rustavi_C --port 27201 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet c --dbpath Kutaisi_C --logpath log.Kutaisi_C --port 27202 --fork --logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet d --dbpath Tbilisi_D --logpath log.Tbilisi_D --port 27300 --fork –logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet d --dbpath Rustavi_D --logpath log.Rustavi_D --port 27301 --fork –logappend --smallfiles --oplogSize 1000

mongod --shardsvr --replSet d --dbpath Kutaisi_D --logpath log.Kutaisi_D --port 27302 --fork –logappend --smallfiles --oplogSize 1000

mongos პროცესი სტანდარტულად უსმენს 27017 პორტს

mongos --configdb mongo.db:26050,mongo.db:2651,mongo.db:26052 --fork --logappend --logpath log.mongos_0

mongos --configdb mongo.db:26050,mongo.db:2651,mongo.db:26052 --fork --logappend --logpath log.mongos_1 --port 26061

mongos --configdb mongo.db:26050,mongo.db:2651,mongo.db:26052 --fork --logappend --logpath log.mongos_2 --port 26062

mongos --configdb mongo.db:26050,mongo.db:2651,mongo.db:26052 --fork --logappend --logpath log.mongos_3 --port 26063

MongoDB–ს მრავალსერვერიანი სისტემებისთვის რეპლიკა სეტისა და

შარდინგის სკრიპტების დაგენერირების შესაძლებლობაც აქვს:

რეპლიკა სეტი

```
mongo --nodb
var repl_set = new Repl_Set_Test({name:"Repl_Set",
nodes:{node1:{smallfiles:"",oplogsize:1000,noprealloc:null},node2:{smallfiles:"",oplo
gSize:1000,noprealloc:null}, arb:{smallfiles:"",oplogSize:1000, noprealloc:null}}});
repl_set.startSet();
```

შარდინგი

```
mongo --nodb
> config = { Tbilisi_D : { smallfiles : "", noprealloc : "", nopreallocj : ""}, Rustavi_D : {
smallfiles : "", noprealloc : "", nopreallocj : "" }, Kutaisi_D : { smallfiles : "", noprealloc
: "", nopreallocj : ""}};
> cluster = new Sharding_Test( { shards : config } );
```

აღწერილი ბრძანებები სერვერებს ერთმანეთთან აკავშირებს, რეპლიკა სეტს კი შემდეგი ბრძანებებით ვააქტიურებთ და ვამოწმებთ

```
mongo --port 27000
rs.status()
rs.initiate()
rs.status()
rs.add(“Mongo.db:27001”)
rs.add(“Mongo.db:27002”)
rs.conf()
connect mongos
mongo
sh.addshard(“a/Mongo.db:27000”)
sh.status()
```

2.7. მონაცემების ასახვა და მათი დამუშავება

2.7.1. მონაცემთა ბაზის აგება MongoDB გარემოში

MongoDB სისტემაში მონაცემთა ბაზას ცხადად არ ვქმნით, ვირჩევთ სამუშაო ბაზას (რომელიც ბრძანების გაშვების დროს ფიზიკურად ჯერ არ არსებობს) და არჩეულ ბაზაში ვამატებთ ახალ ჩანაწერებს. მონაცემთა ბაზა მონაცემის ჩაწერის დროს ავტომატურად შეიქმნება.

სურგულაძე გ., კივილაძე გ. დამხმარე სახელმძღვანელოს „შესავალი NoSQL მონაცემთა ბაზებში (MongoDB)“ დანართის მიხედვით შექმნილია moviesDB მონაცემთა ბაზა [33].

აღნიშნულ მონაცემთა ბაზაში მხოლოდ ერთი movies კოლექცია გვაქვს. რეალურ სისტემაში ამ კოლექციის გარდა შესაძლოა, გვქონოდა მუსიკების, თამაშების, მომხმარებლების, მსახიობებისა და სხვადასხვა კოლექციები. movies კოლექიაში შენახულია მთელი ის ინფორმაცია, რაც ფილმს უკავშირდება:

- "_id" - უნიკალური იდენტიფიკატორი.
- "title" - სათაური.
- "year" - გამოშვების წელი.
- "rated" - შეფასება.
- "released" - საიტზე ფილმის დამატების თარიღი.
- "runtime" - ფილმის ხანგრძლივობა.
- "countries" - მწარმოებელი ქვეყნები.
- "genres" - ფილმის ჟანრების ჩამონათვალი.
- "director" - რეჟისორი.
- "writers" - სცენარისტები.
- "actors" - მსახიობები.
- "plot" - მოკლე შინაარსი.
- "poster" - ფილმის შესაბამისი პოსტერის ბმული.
- "imdb" - ჩადგმული დოკუმენტი IMDB რეიტინგისთვის.
- "tomato" - ჩადგმული დოკუმენტი Rotten Tomatoes მაყურებლების რეიტინგისთვის.
- "metacritic" – Metacritic რეიტინგისთვის.
- "awards" - ჩადგმული დოკუმენტი ჯილდოების ასაღწერად.
- "type" - ფილმის ტიპი (სრულ/მოკლე მეტრაჟიანი, სერიალი...)
- "reviews" - ვებ-გვერდის მომხმარებლების შეფასებები (ჩადგმული დოკუმენტების მასივი)

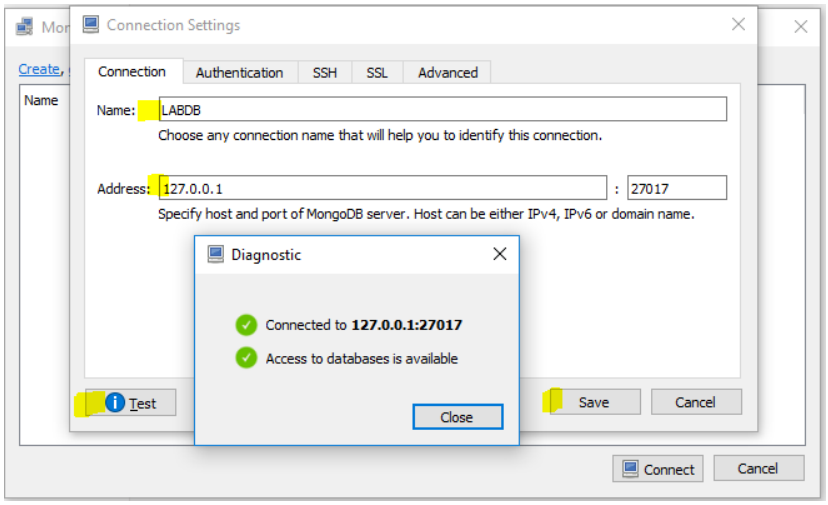
2.7.2. RoboMongo პლატფორმა

RoboMongo პლატფორმა ალტერნატიულ ინსტრუმენტებთან შედარებით ერთ-ერთი ყველაზე გავრცელებული GUI (graphical user interface) გრაფიკული ინტერფეისია MongoDB მონაცემთა ბაზასთან სამუშაოდ [38].

RoboMongo პლატფორმა გამოირჩევა მოხერხებულობითა და სიმარტივით. თავდაპირველად MongoDB სისტემასთან ვამატებთ ახალ კავშირს (Create Connection).

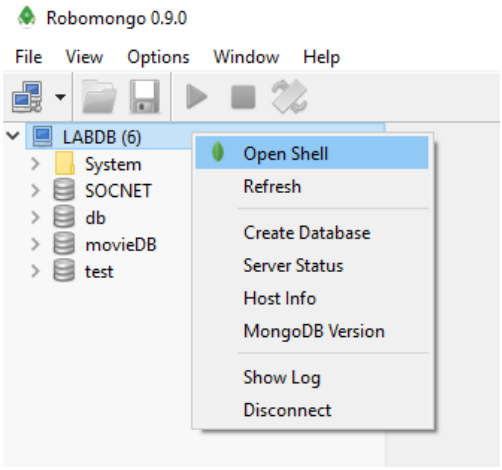
დიალოგის ფანჯარაში (ნახ. 16) ვავსებთ შესაბამის ველებს:

- Name - ახალი კავშირის სახელი (შეგვიძლია ნებისმიერი სახელის შერჩევა)
- Address - IP მისამართი და პორტი, რომელზეც სერვერი კლიენტებისგან მოსულ მოთხოვნებს ელოდება. სადისერტაციო ნაშრომის შემთხვევაში ჩაწერილია 127.0.0.1 - რადგან სერვერი ლოკალურ კომპიუტერზე არის გამართული. პორტის კონფიგურაცია კი უცვლელად დავტოვეთ (default – 27017)



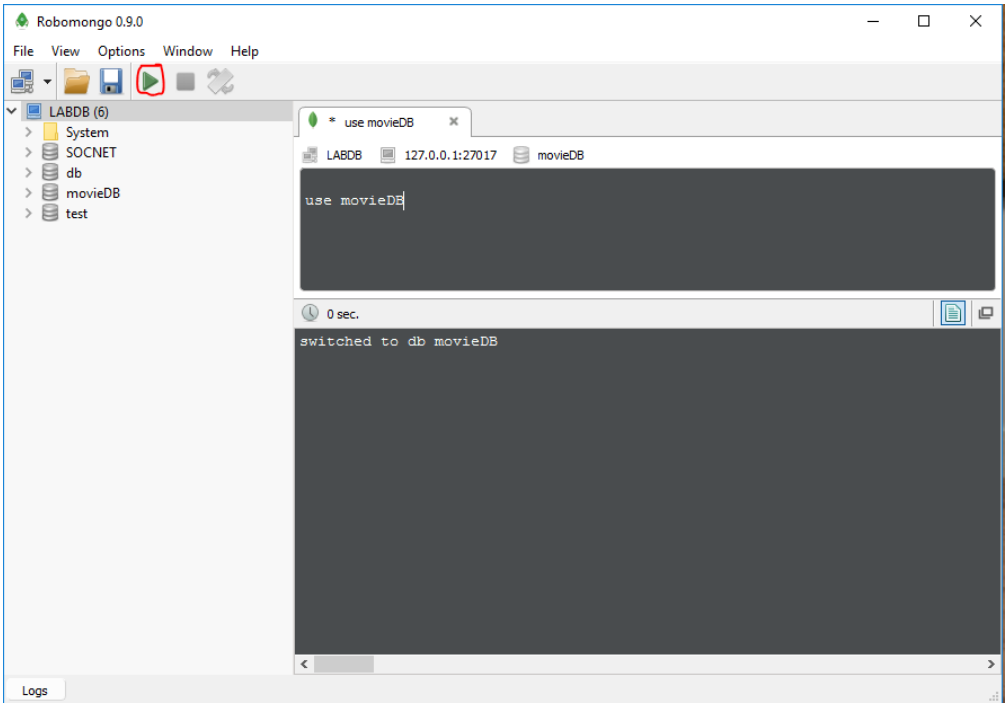
ნახ. 16. კავშირის გახსნა MongoDB სისტემასთან

კავშირის შექმნამდე შეგვიძლია შევამოწმოთ მისი ვალიდურობა (Test). შემდეგ ვხსნით დამახსოვრებულ კავშირს (MongoDB Connections -> Connect) და გადავდივართ ბრძანებების რეჟიმში (ნახ.17).



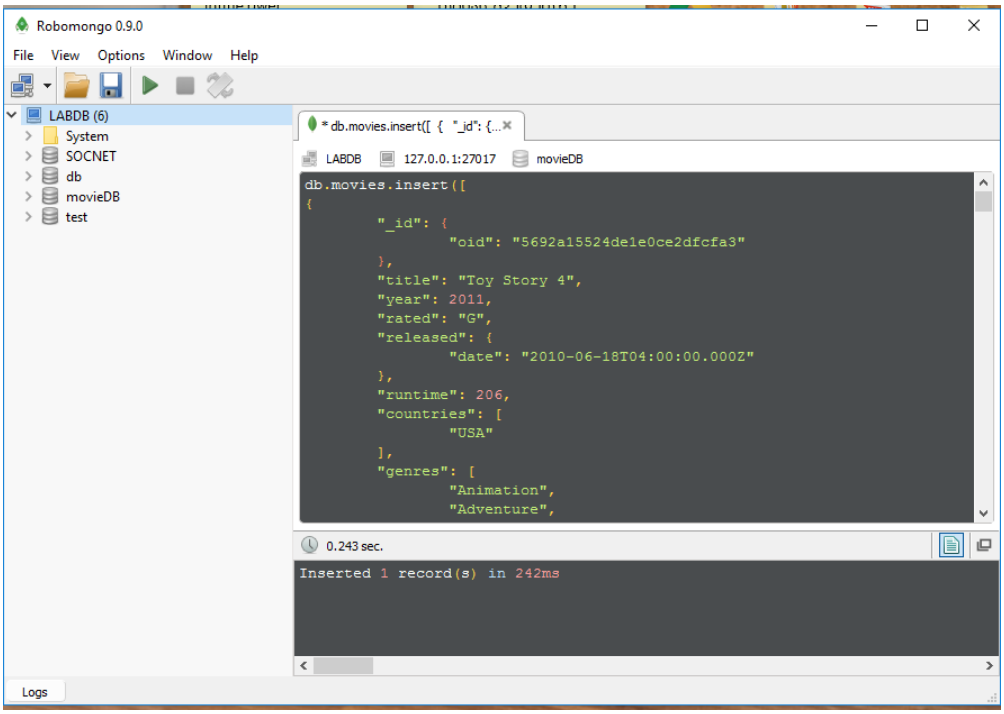
ნახ. 17. Robomongo - ტერმინალთან დაკავშირება

შემდეგ ნახაზზე (ნახ.18) ნაჩვენებია პირველი ბრძანება - აქტიური მონაცემთა ბაზის არჩევა



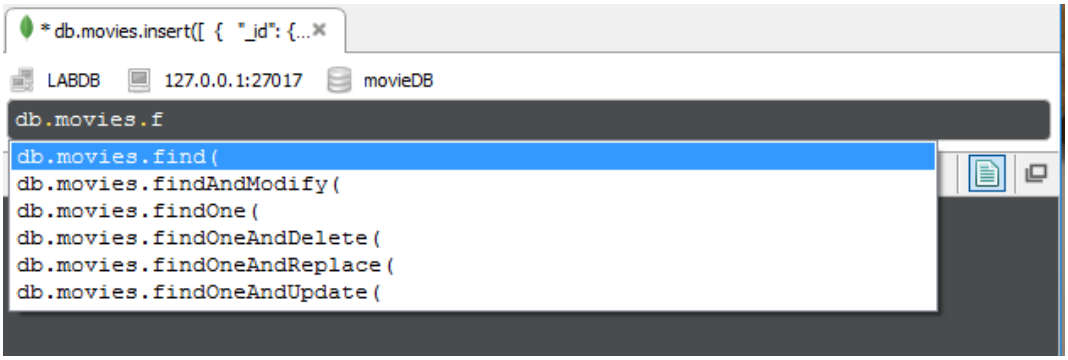
ნახ. 18. აქტიური მონაცემთა ბაზის არჩევა

შემდეგ ეტაპზე ვამატებთ მონაცემებს ფილმების კოლექციაში (ნახ. 19)



ნახ. 19. მონაცემების დამატება movies კოლექციაში

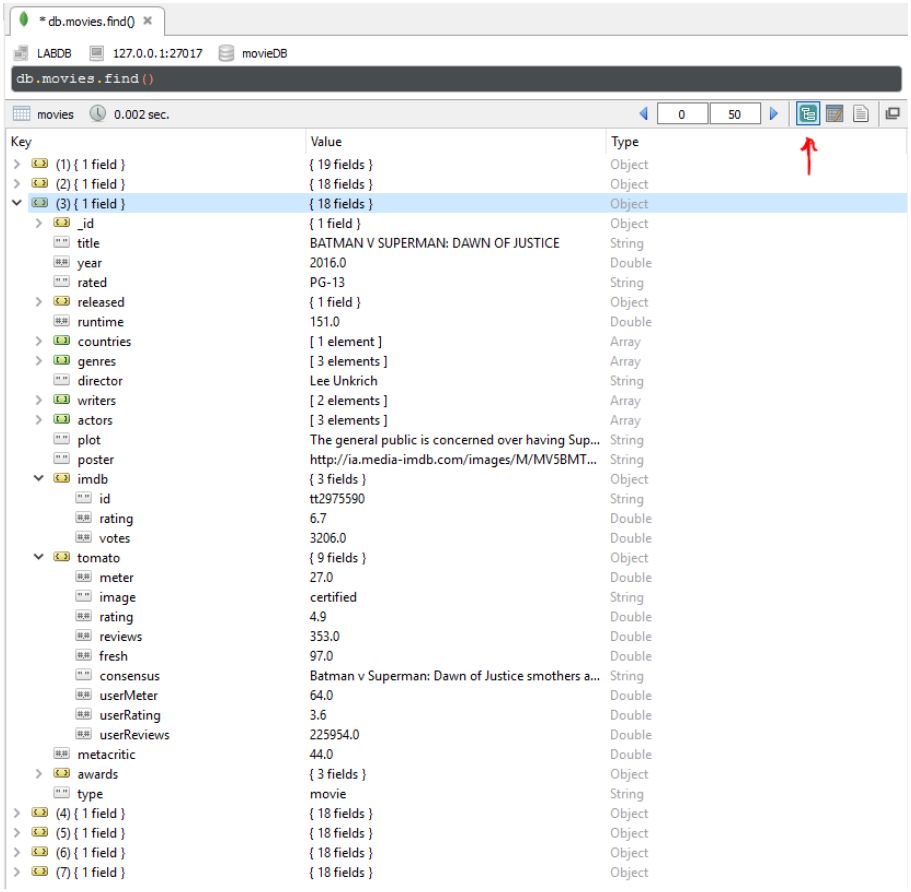
ნახაზ 20-ზე ნაჩვენებია RoboMongo გარემოს intellisense შესაძლებლობა - შემოგვთავაზოს შესაძლო ვარიანტები პროგრამული კოდის წერის დროს.



ნახ. 20. RoboMongo intellisense

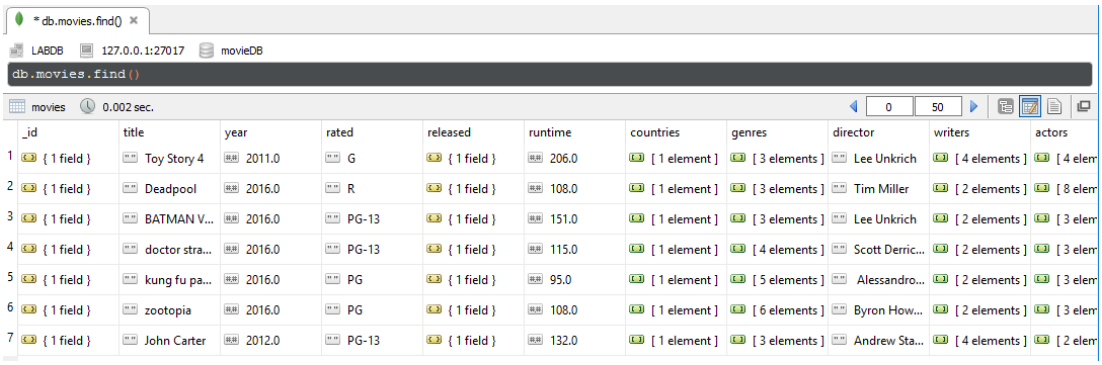
db.movies.find() ბრძანება გამოიტანს მიმდინარე ბაზის (movieDB) ფილმების კოლექციის ყველა ჩანაწერს. შედეგის ჩვენება შესაძლებელია სამი სხვადასხვა ფორმატით:

- ინფორმაციის ხისებრი სტრუქტურით გამოტანა (View results in tree mode) (ნახ. 21)



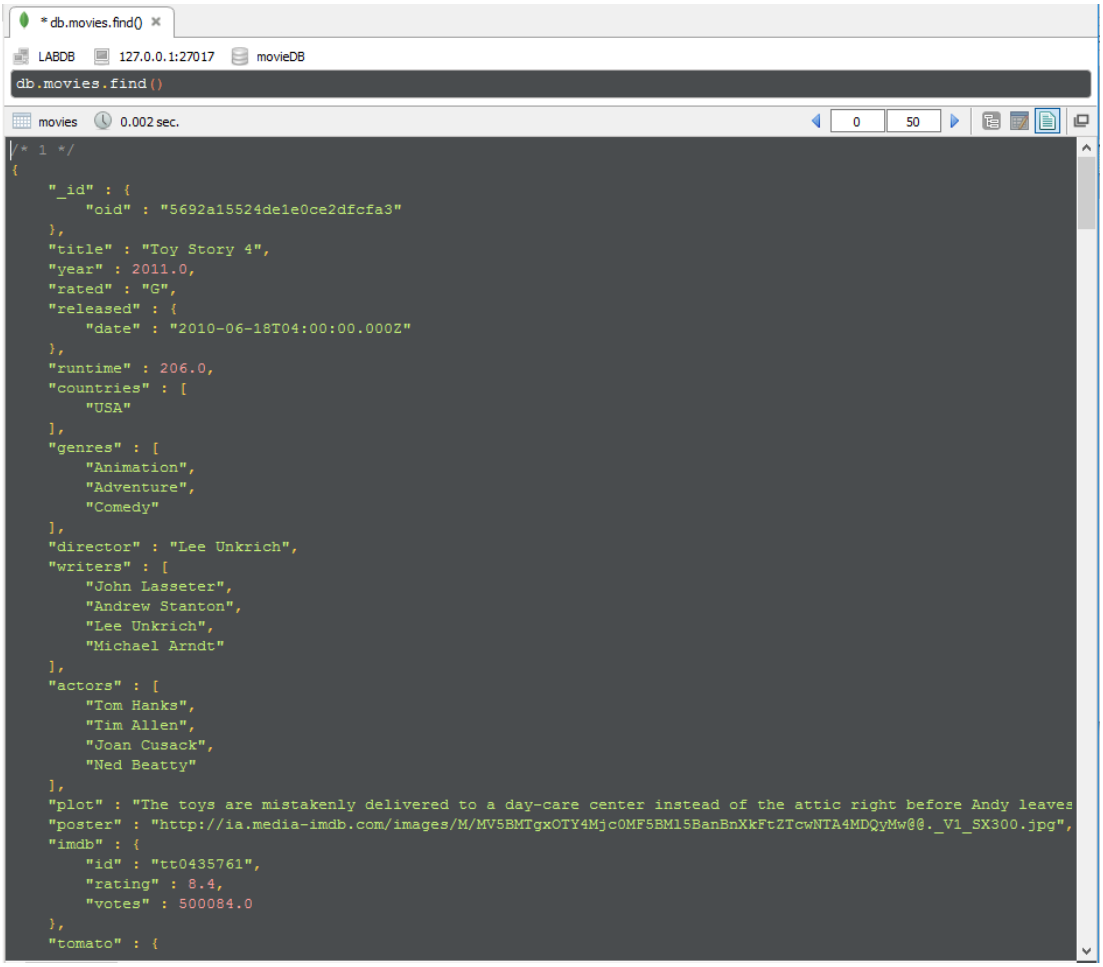
ნახ. 21. მონაცემების ვიზუალიზაცია ხისებრი სტრუქტურით

- ინფორმაციის ცხრილის სახით გამოტანა (View results in table mode) (ნახ.22)



ნახ. 22. მონაცემების ვიზუალიზაცია ცხრილის სტრუქტურით

- ინფორმაციის JSON სტრუქტურით გამოტანა (View results in text mode) (ნახ.23)



ნახ. 23. მონაცემების ვიზუალიზაცია JSON სტრუქტურით

2.7.3. სხვადასხვა ოპერაციების მაგალითები

მას შემდეგ, რაც წარმატებით შეიქმნა movieDB მონაცემთა ბაზა, და შეივსო ზემოთხსენებული სახელმძღვანელოს ([33]) დანართში მოყვანილი ჩანაწერების მიხედვით, მოვიყვანოთ სხვადასხვა ოპერაციების მაგალითები.

- მიმდინარე ბაზის არჩევა

use movieDB

- ძიება ფილმის სათაურის მიხედვით

db.movies.find({"title":'Zootopia'})

- ძიება ფილმის გამოშვების თარიღის მიხედვით

db.movies.find({ year: { $gt: 2011 } })

db.movies.find({ year: { $gt: 2009, $lt: 2011 } })

სადაც $gt და $lt შედარების ოპერატორებია (Greater Then, Less, Then)

- ფილმის ძიება IMDB რეიტინგის მიხედვით (ამ მაგალითში ჩანს, თუ როგორ შეიძლება ჩადგმული დოკუმენტის ველებზე წვდომა)

```
db.movies.find(
 {
        "imdb.rating": {$gt:7.5}
 }
)
```

- იმ ფილმების ძიება სადაც ჩამოთვლილი მსახიობებიდან ერთ-ერთი მაინც თამაშობს

```
db.movies.find(
  {
    actors: { $in: [ "Joan Cusack", "Ned Beatty", “Tim Allen” ] }
  }
)
```

- ძიება ფილმის უნიკალური ნომრის მიხედვით

```
db.movies.find(
 {
  "_id.oid" : "5768b15524da1t0sd2bvvbv3"
 }
)
```

- ძიება ჩადგმული დოკუმენტების მასივში:

კოდის შემდეგი ფრაგმენტის შესრულება დააბრუნებს ყველა იმ ფილმს, რომელზეც დატოვებულია კონკრეტული მომხმარებლის კომენტარი

```
db.movies.find(
  {
    reviews: {
          $elemMatch: {
              name: "qwerty_user"
          }
```

}

}

)

• სორტირება ფილმის გამოშვების წლის მიხედვით

db.movies.find().sort({ year: 1 })

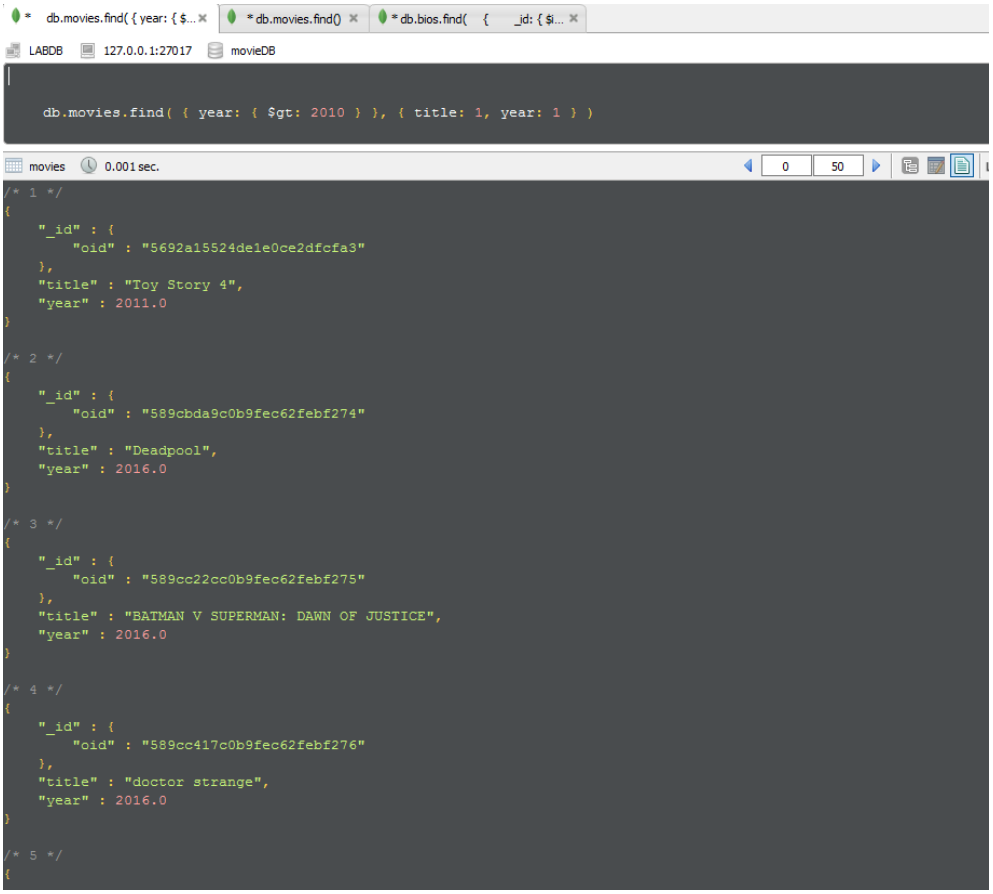
• გამოსატანი ჩანაწერების სორტირებითა და რაოდენობის შეზღუდვით გამოვიტანთ ყველაზე ძველ ფილმს:

db.movies.find().sort({ year: 1 }).limit(1)

• სასურველი ველების პროექცია

db.movies.find({ year: { $gt: 2009 } }, { title: 1, year: 1 })

{ title: 1, year: 1 } - ამ სახის ფორმატით გამოვიტანთ მხოლოდ სასურველ ველებს (ნახ.24)



ნახ. 24. სასურველი ველების პროექცია

სასურველი ველების გამოტანა შესაძლებელია ჩადგმული დოკუმენტების ძებნითაც:

```
db.movies.find( { "imdb.rating": {$gt:8}}, { title: 1, "imdb.rating": 1} )
```

- **Delete:**

```
db.movies.remove({_id: “589cc696c0b9fec62febf277”}); // ჩანაწერის წაშლა იდენტიფიკატორის მიხედვით
db.movies.remove({genres: 'კომედია'});
db.movies.remove();  // მთლიანი კოლექციის წაშლა
```

- **სხვადასხვა ბრძანებები:**

```
db.runCommand({count: 'movies', query: {year: 2009}}); // 2009 წელს გამოშვებული ფილმების რაოდენობა
db.runCommand({distinct: 'movies', key:'title'}); // ფილმების უნიკალური ჩანაწერების სია
db.runCommand({ dropDatabase: 1 });  // მთლიანი მონაცემთა ბაზის წაშლა
db.runCommand({ getLastError: 1 }); // ბოლო შეცდომის კოდის ჩვენება
db.runCommand({ serverStatus: 1 });  // სერვერის სტატუსის შემოწმება
db.runCommand({ shutdown: 1 });
```

- **Indexes**

```
db.movies.ensureIndex({"title": 1}, {unique: true}); // უნიკალურობის შეზღუდვა
db.movies.ensureIndex({"year": 1}); // ინდექსი ფილმის გამოშვების წელზე
db.movies.ensureIndex({"imdb.rating ": 1}); // ინდექსი ჩადგმული დოკუმენტის ველზე
db.movies.getIndexes(); // არსებული ინდექსების გამოტანა
db.movies.dropIndex('index_name'); // ინდექსის წაშლა
```

- **Import / Export**

```
mongoexport -d movieDB -c movies > mongo.movies.bckp.txt
mongo movieDB --eval "db.movies.remove()"
```

mongoimport -d movieDB -c movies --file movies.bckp.txt

mongoexport -d movieDB -c movies --jsonArray > movies.bckp.json // ექსპორტი

mongoimport -d movieDB -c movies --jsonArray < movies.bckp.json // იმპორტი

2.8. მეორე თავის დასკვნა

მონაცემთა ბაზების მწარმოებლურობის ძირითადი მახასიათებლების - მონაცემთა მთლიანობის, ტრანზაქციათა იზოლირების დონეების, ACID (Atomicity, Consistency, Isolation, Durability) პრინციპების და CAP თეორემის გათვალისწინებით გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-ის მონაცემთა ბაზად შეირჩა მონაცემთა დოკუმენტ–ორიენტირებული, არარელაციური ბაზა MongoDB.

შემუშავებულია კრიტიკული სისტემის ინფრასტრუქტურა, აღწერილია სისტემის გამართვის ეტაპები და დაპროექტებულია მაღალი წვდომადობის მქონე მონაცემთა საცავი.

მონაცემთა ბაზის სამივე ძირითადი მახასიათებლის (პროცესორი, ოპერატიული მეხსიერება და ხისტი დისკი), ისევე როგორც საიმედოობისა და წვდომადობის გასაუმჯობესებლად გადაწყდა Sharding–ისა და რეპლიკაციის ტექნოლოგიების გამოყენება.

3 თავი. ექსპერიმენტული ნაწილი: სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანა

3.1. საკვლევი სფეროს აქტუალობა, კვლევის სიახლე და პრობლემის ფორმულირება

სერვერული უზრუნველყოფის არქიტექტურის შემუშავება მნიშვნელოვანი ეტაპია, მაგრამ, ცხადია, საინტერესოა, რა სახით და რა მონაცემებს შევინახავთ მასში. შსს სსიპ „112“-ში უკვე წარმატებით ვიყენებთ MongoDB მონაცემთა ბაზას დავალებათა მართვის სისტემისთვის. ამასთან ერთად, მიმდინარე პროექტად გვაქვს მონაცემების სხვადასხვა სერვერებიდან MongoDB-ში თავმოყრა, რაც ბევრად უფრო აასწრაფებს და გაამარტივებს მონაცემთა ანალიზსა და სტატისტიკურ დამუშავებას.

ბოლო პერიოდში დავინტერესდი მანქანური დასწავლის მიმართულებით და მინდა, სადისერტაციო ნაშრომის მესამე თავი დავუთმო გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-ისთვის ისეთი პრობლემური საკითხის ოპტიმიზაციას, როგორიც არის სასწრაფო სამედიცინო მომსახურების ბრიგადების დისპეტჩერიზაცია.

დიდი რაოდენობით დაგროვებულმა ინფორმაციამ მოგვცა საშუალება, განვვითარებულიყავით მანქანური დასწავლის (Machine Learning) მიმართულებით. ჯერ კიდევ 1959 წელს ამერიკელმა მეცნიერმა არტურ სამუელმა (Arthur Samuel) მანქანური დასწავლა განმარტა როგორც „მეცნიერების სფერო, რომელიც კომპიუტერებს აძლევს თვითდასწავლის შესაძლებლობას, ცხადად დაპროგრამების გარეშე“ [26]. დღეს კი უკვე ხელოვნური ინტელექტი მართავს თვითმფრინავს, დახვეწილია სახეთა ამოცნობის ალგორითმები და თუნდაც, თითქმის არასდროს გვხვდება spam ტიპის იმეილები inbox-ში.

სამწუხაროდ, ზემოთ აღნიშნული მიმართულებებით საქართველოში მხოლოდ ერთეულები სარგებლობენ, რაც გამოწვეულია ამ ტექნოლოგიების

სირთულით - ტექნოლოგიების დანერგვისთვის საჭიროა მრავალმხრივი გამოცდილება და დიდი რაოდენობის სერვერული რესურსი.

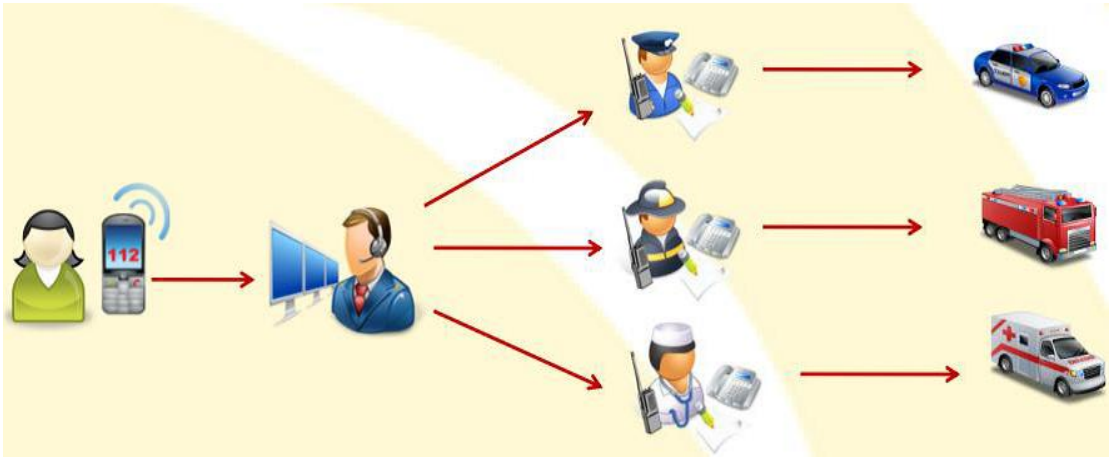
3.2. დისპეტჩერიზაციის პროცესი „112“-ში

112-ში შეტყობინებები შემოდის გადაუდებელი შემთხვევების დროს. გადაუდებელი შემთხვევა გულისხმობს ნებისმიერ სიტუაციას, რომელიც მოითხოვს ინციდენტის ადგილზე სასწრაფო-სამედიცინო, სახანძრო-სამაშველო თუ პოლიციის ბრიგადის დაუყოვნებლივ გასვლას.

112-ის ოპერატორი ზარის შინაარსიდან გამომდინარე, გადაუდებელი და საგანგებო სიტუაციების შესახებ შემოსული ინფორმაციის მიღებას, დამუშავებას, შეფასებას ახდენს და შემოდგომი რეაგირებისათვის შესაბამის სამსახურებს აწვდის. აღნიშნული სამსახურებია:

- პოლიცია
- სასწრაფო - სამედიცინო დახმარების ცენტრი
- სახანძრო - სამაშველო სამსახური

ნახაზ 25-ზე ნაჩვენებია 112-ში ზარის მართვის ძირითადი გეგმა (ზარის ინიციატორი -> ოპერატორი -> დისპეტჩერი -> ბრიგადა):

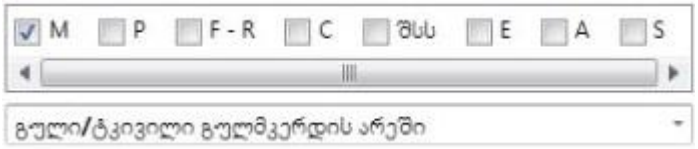


ნახ. 25. 112-ში საქმის მართვის ძირითადი გეგმა

112 საქართველოში მოქმედ სამ სხვადასხვა სამსახურთან, კერძოდ, საპატრულო პოლიციასთან, სახანძრო-სამაშველო სამსახურსა და სასწრაფო-სამედიცინო დახმარების სამსახურებთან კოორდინირებულად მუშაობს. გადაუდებელი დახმარების ოპერატიული მართვის ცენტრი

უზრუნველყოფს მოქალაქეებისაგან მიღებული ინფორმაციის დამუშავებას და რეაგირების მიზნით აღნიშნული სამსახურებისათვის მიწოდებას [42].

მიღებული შეტყობინების შინაარსის მიხედვით ოპერატორი მოქმედეს შემდეგნაირად: გონებრივად აანალიზებს მიღებულ ინფორმაციას და გამოავლენს მის მთავარ ნაწილს - ინციდენტის ტიპს. ეს ავტომატურად მონიშნავს ამ შემთხვევისათვის საჭირო შემთხვევის კატეგორიას (ნახ.26).



ნახ. 26. ოპერატორის მიერ შევსებული ბარათის მიმთითებელი ნაწილი

უნდა აღინიშნოს, რომ თითოეული შემთხვევისათვის, შესაძლოა, მონიშნული იყოს ერთზე მეტი მომსახურების კატეგორია, მაგალითად, ინციდენტის ტიპის „ჭრილობა“ მონიშვნის შემთხვევაში ავტომატურად გააქტიურდება სამედიცინო და საპატრულო სამსახურები.

ამჟამად, მიმღები ოპერატორები იყენებენ შემდეგ მოდელს. სამედიცინო ინციდენტის ტიპები კლასიფიცირებულია გამოცდილი ექიმების მიერ სამ - „დაბალი“, „საშუალო“ და „მაღალი“ კატეგორიის გადაუდებლობის მქონე ფიქსირებულ ჯგუფებად.

შემდეგ, აღრიცხული ინფორმაცია გადაეგზავნება ყველა აქტიურ დისპეტჩერს. რამდენადაც სადისერტაციო ნაშრომი მიზნად ისახავს სასწრაფო სამედიცინო მომსახურების გაუმჯობესებას, შემდგომში განხილული იქნება მხოლოდ სამედიცინო მომსახურების დისპეტჩერების საქმიანობა.

თითოეული რეგიონის დიპეტჩერი ხედავს მოცემული მომენტისთვის შემოსული აქტიური შემთხვევების სიას და მოცემული მომენტისთვის ხელმისაწვდომი სასწრაფო დახმარების მანქანების სიმრავლეს.

ყოველი შემთხვევა რჩება აქტიური სახით, სანამ რომელიმე დისპეტჩერი მას არ აირჩევს შემდგომი რეაგირებისთვის. სასწრაფო სამედიცინო დახმარების ბრიგადების ხელმისაწვდომობის

განსაზღვრისათვის არსებობს რამოდენიმე სტატუსი - „თავისუფალი“, „მიღებული გამოძახებით“, „გასულია მისამართზე“, „მისულია მისამართზე“, „ჰოსპიტალიზაციის პროცესშია“ და სხვა რამდენიმე სტატუსი, რაც დაკავშირებულია ტექნიკურ მომსახურებასთან.

დისპეტჩერის ამოცანაა, გამოავლინოს ყველაზე მაღალი კატეგორიის გადაუდებლობის შემთხვევა აქტიური გამოძახებების სიიდან და გადასცეს ის იმ სასწრაფო დახმარების ბრიგადას, რომელსაც შეუძლია ყველაზე სწრაფი რეაგირება. ასეთი ამოცანა საკმაოდ რთული გადასაწყვეტია, რადგან გამოძახებათა რაოდენობა ხშირად ბევრია, რაც სტრესულ სიტუაციაში მუშაობასთან ერთად იწვევს დისპეტჩერის დაღლილობას, ნერვიულობას და აქედან გამომდინარე, არასწორი გადაწყვეტილებების გაზრდილ რიცხვს.

დისპეტჩერიზაციის გაუმჯობესების მიზნით, უნდა შეიქმნას კომპიუტერული სისტემის მოდელი რომელიც ობიექტური გამოთვლების საფუძველზე საშუალებას მოგვცემს, ერთის მხრივ, გამოვლინდეს ყველაზე მაღალი კატეგორიის გადაუდებლობის მქონე შემთხვევები (გამოძახებები), მეორეს მხრივ კი შეირჩეს სასწრაფო დახმარების ის ბრიგადები, რომლებიც შეძლებენ არჩეული შემთხვევების ყველაზე სწრაფად მომსახურებას.

მოცემულ თავში განხილულია სასწრაფო დახმარების ბრიგადის შერჩევის ამოცანა, ხოლო დისერტაციის შემდეგი თავი ეძღვნება შემთხვევათა მომსახურების რიგის დადგენას.

3.3. დისპეტჩერიზაციის კომპიუტერული სისტემა ბრიგადების შესარჩევად

სასწრაფოს ბრიგადის გათავისუფლებისა და ახალ მისამართზე მისვლის სავარაუდო დროის არცოდნის გამო დისპეტჩერებს ხშირად უწევთ არაოპტიმალური გადაწყვეტილებების მიღება.

მაგალითად, ზარის ინიციატორთან ტერიტორიულად უახლოესი თავისუფალი ბრიგადა საცობის გათვალისწინებით 1 საათის სავალზეა, სხვა საქმეზე გასული ბრიგადა კი შეიძლება ინიციატორის მეზობლად იყოს და 5 წუთში გათავისუფლდეს. არაოპტიმალური მენეჯმენტით დროც ბევრად მეტი იხარჯება და ფინანსური თუ ადამიანური რესურსიც.

პროექტის მიზანია ბრიგადების განაწილების გაუმჯობესება. ანუ, გამოძახების საპასუხოდ ისეთი ბრიგადის არჩევა, რომელსაც შეუძლია, ყველაზე სწრაფად მივიდეს შემთხვევის ადგილზე.

ბრიგადის შერჩევის პროცესი იწყება პაციენტის შესაძლო ჰოსპიტალიზაციის ალბათობის გამოთვლით. თუ ჰოსპიტალიზაციის ალბათობა მაღალია ($P \geq 0.5$) მოცემულ შემთხვევაზე გაგზავნილი სასწრაფო დახმარების მანქანა აღარ განიხილება როგორც თავისუფალი დროის უახლოეს ინტერვალში.

თუ შემთხვევის ჰოსპიტალიზაციის ალბათობა დაბალია ($P < 0.5$) ალგორითმი შეაფასებს სასწრაფო დახმარების მანქანის შესაძლო დაკავებულობის დროს და მიაწვდის ამ ინფორმაციას დისპეტჩერს.

ამ დროის შუალედის გავლის შემდეგ, სავარაუდოა, რომ ბრიგადა განთავისუფლდება.

უნდა აღინიშნოს, რომ ალგორითმი იძლევა მხოლოდ ალბათობის სიდიდეებს და არა დროის ზუსტ შუალედებს. მიღებული რეკომენდაცია დისპეტჩერმა უნდა გადაამოწმოს უშუალოდ ბრიგადის ექიმთან [43].

პაციენტის ჰოსპიტალიზაციის ალბათობა, ისევე, როგორც მიახლოებითი დრო, რასაც ბრიგადა ხარჯავს პაციენტზე, გამოითვლება ისეთი მკაცრად განსაზღვრული მონაცემების საფუძველზე, როგორიცაა მაგალითად, ინციდენტის ტიპი, პაციენტის ასაკი და სქესი, ბრიგადის ინფორმირების, რეაგირების, ადგილზე მისვლისა და გათავისუფლების დროები.

პაციენტთან დახარჯული ალბათური დროის გამოთვლის გარდა, მნიშვნელოვანი იქნება ბრიგადის გადაადგილების სავარაუდო დროც, რასაც მარშრუტიზაციის სერვისებით მივიღებთ. არსებული სერვისები მოგვცემს საშუალებას, გავარკვიოთ, გზებზე არსებული „საცობების“ გათვალისწინებით, რა დრო იქნება საჭირო ერთი პუნქტიდან მეორემდე გადასაადგილებლად (მაგალითად Google traffic).

3.4. მონაცემთა კრებული (Dataset)

Dataset-ის შემუშავებისას გამოვყავით ორი ეტაპი. პირველ რიგში, SQL-ის საშუალებით 112-ის მთავარი მონაცემთა ბაზიდან ამოვიღეთ 5 წლის მონაცემები და სასურველი ფორმატით შევინახეთ მონაცემთა საცავში. მეორე ეტაპზე Python-ის საშუალებით დავამუშავეთ მონაცემები, ამოვშალეთ ანომალიები და გავამზადეთ მანქანური დასწავლის ალგორითმებთან სამუშაოდ.

3.4.1. მონაცემების შეგროვება (Data-gathering)

რადგან მიმდინარე პროექტი მჭიდრო კავშირშია გადაუდებელი დახმარების ერთიანი ცენტრის კონფიდენციალურ ინფორმაციასთან, თავს შევიკავებ მონაცემთა ბაზასთან მუშაობის დეტალების აღწერისგან. ქვემოთ მოყვანილია პირველი ეტაპის შედეგი:

- CASE_ID - საქმის უნიკალური კოდი (გვჭირდებოდა მხოლოდ მონაცემების შეგროვების ეტაპზე)
- CALL_ID - ზარის უნიკალური კოდი (გვჭირდებოდა მხოლოდ მონაცემების შეგროვების ეტაპზე)
- NOM - ინიციატორის ნომერი (გვჭირდებოდა მხოლოდ მონაცემების შეგროვების ეტაპზე)
- DATE_CREATED - ზარის შემოსვლის თარიღი
- CASE_TEMPLATE_ID - ინციდენტის ტიპი (გული, კრუნჩხვა, სუნთქვის გაძნელება...)
- INJURED_PERSON_AGE_YEARS,
INJURED_PERSON_AGE_MONTHS,
INJURED_PERSON_AGE_DAYS

პაციენტის ასაკი ივსება წლების, თვეების ან დღეების სახით (ნახ.27)

ნახ. 27. პაციენტის ასაკი

ახალ დაბადებული ან ჩვილი ბავშვის შემთხვევაში ოპერატორი ავსებს დღეების ველს, წლამდე ასაკის ბავშვებისთვის - თვეების ველს, დანარჩენი პაციენტების ასაკი კი აღირიცხება წლების სახით.

- INJURED_PERSON_GENDER - სქესი
- CATEGORIES – ამ სვეტში გვაქვს ინფორმაცია ყველა იმ გადაუდებელი დახმარების სერვისზე, რაც ჩართული იყო ინციდენტის მართვაში (სასწრაფო, სახანძრო, სამაშველო, პოლიცია...).
- SPENT_MINS - სასწრაფოს ბრიგადის მიერ პაციენტთან, ბინაზე გატარებული დრო (ჰოსპიტალიზაციის შემთხვევაში ემატება პაციენტის ჰოსპიტალიზაციაზე დახარჯული დროც)
- PREV_HOSP_COUNT - ამავე ნომერზე იგივე ინციდენტის ტიპის მიხედვით ჰოსპიტალიზაციების რაოდენობა ბოლო 1 წლის განმავლობაში
- LASTHOSPDD - ბოლო ჰოსპიტალიზაციიდან გასული დღეების რაოდენობა (ვაკვირდებით ბოლო 1 წელიწადს)
- CALL_COUNT - იმავე ნომერზე ბოლო 1 თვის განმავლობაში შექმნილი საქმეების რაოდენობა

3.4.2. მონაცემების წინასწარი დამუშავება (Data preprocessing)

მონაცემების წინასწარი დამუშავება და ანალიზისთვის მომზადება ხშირად ყველაზე მნიშვნელოვანი პროცესია მანქანური დასწავლის პროექტის უპირველესი და ხშირად, უმთავრესი ეტაპია [44,45]. მონაცემების პირველადი შეგროვების ეტაპზე ხშირად, მინიმალურად გვაქვს კონტროლის საშუალება. შესაბამისად, ხშირია ისეთი ანომალიური მონაცემები, როგორიცაა:

- სპექტრის გარე მნიშვნელობები - მაგალითად, დროის ან ანაზღაურების უარყოფითი მნიშვნელობები.
- მონაცემთა შეუძლებელი კომბინაციები.
- შეუვსებელი მონაცემები.
- დასწავლისთვის გამოუსადეგარი მონაცემები - მაგალითად, უსიმბარათო ტელეფონებიდან განხორციელებული ზარები.

ამ და სხვა მრავალი ანომალიური შემთხვევის გაფილტვრის გარეშე მონაცემების გაანალიზებამ, შეიძლება, მცდარ შედეგებამდე მიგვიყვანოს.

მონაცემების დამუშავებისა და ანალიზისთვის ვიყენებთ Python პროგრამირების ენას.

თავდაპირველად, ვშლით "ტრიაჟის" ტიპის საქმეებს (საქმეებს, სადაც ერთდროულად სასწრაფო დახმარების რამდენიმე ბრიგადა გავიდა):

```
# keep=False პარამეტრი უზრუნველყოფს დუბლირებული ჩანაწერების სრულად წაშლას.
df_orig = df_orig.drop_duplicates(subset=['CALL_ID'], keep=False)
```

მნიშვნელოვანია, რომ შემდეგ სვეტებში არ გვქონდეს შეუვსებელი მონაცემები:

```
# dropna მეთოდი how='any' პარამეტრის მითითებით შლის ყველა იმ სტრიქონს, სადაც აღ
ნიშნული სვეტებიდან ერთ-ერთი მაინც არის მონაცემის გარეშე.
df_orig.dropna(subset=['CASE_ID', 'CALL_ID', 'DATE_CREATED', 'CASE_TEMPLATE_ID', 'NO
M', 'CATEGORIES', 'SPENT_MINS'], how='any', inplace=True)
```

შემდგომ დამუშავებამდე, ვშლით ისეთ მონაცემებს, რომლებსაც სხვადასხვა მიზეზების გამო, ვერ გამოვიყენებთ დასწავლის პროცესისთვის:

```
# ვშლით არასტანდარტული სიგრძის ნომრებს
df = df[df['NOM'].map(len) <= 9]
# ვფილტრავთ მობილურისა და ქალაქის ნომრებს (მხოლოდ თბილისი)
pattern='5\d\d\d\d\d\d\d|2\d\d\d\d\d\d|322\d\d\d\d\d\d|79[01]\d\d\d\d\d\d'
# ვფილტრავთ მობილურისა და ქალაქის ნომრებს წინასწარ გამოცხადებული პატერნის მ
იხედვით
df = df[df['NOM'].str.match(pattern)]
# ვშლით სხვადასხვა მიზეზის გამო არსებულ ანომალიებს
df = df[df['SPENT_MINS'] <= 120]
df = df[df['SPENT_MINS'] >= 0]
```

მონაცემების წინასწარი დამუშავების შედეგად ვიღებთ შემდგომი დასწავლისთვის გამზადებულ 2.7 მილიონი ჩანაწერისგან შემდგარ მონაცემთა ერთობლიობას 219 სვეტის სახით:

- DATE_CREATED - ზარის შემოსვლის თარიღს ჩავუტარეთ კატეგორიზაცია სსიპ „112“-ისთვის სპეციფიური დროითი შუალედების მიხედვით. კატეგორიზაციისთვის გამოვიყენეთ One Hot Encoding მეთოდი,

რაც ნიშნავს, რომ ყოველი სტრიქონისთვის შევსებულია მხოლოდ ერთერთი:
[TIME_02_08, TIME_08_12, TIME_12_17, TIME_17_23, TIME_23_02]
ამ ეტაპზე მხოლოდ საათებს ვაქცევთ ყურადღებას, თუმცა, სამომავლოდ, შესაძლებელი იქნება სეზონების გამოყოფაც.

```
# დროითი დაჯგუფება "112"-ის სპეციფიკის მიხედვით    [HH_HH)
df['TIME_02_08'] = 0
df['TIME_08_12'] = 0
df['TIME_12_17'] = 0
df['TIME_17_23'] = 0
df['TIME_23_02'] = 0

for i, row in df.iterrows():
    if 2 <= int(row['DATE_CREATED'][11:13])  < 8 :
        df.loc[i,'TIME_02_08'] = 1
    elif  8 <= int(row['DATE_CREATED'][11:13])  < 12 :
        df.loc[i,'TIME_08_12'] = 1
    elif 12 <= int(row['DATE_CREATED'][11:13])  < 17 :
        df.loc[i,'TIME_12_17'] = 1
    elif 17 <= int(row['DATE_CREATED'][11:13])  < 23 :
        df.loc[i,'TIME_17_23'] = 1
    elif int(row['DATE_CREATED'][11:13]) == 23 :
        df.loc[i,'TIME_23_02'] = 1
    elif 0 <= int(row['DATE_CREATED'][11:13])  < 2 :
        df.loc[i,'TIME_23_02'] = 1
```

• CASE_TEMPLATE_ID - ინციდენტის ტიპები გადმოტანილია One Hot Encoding მეთოდით. მივიღეთ 194 უნიკალური სვეტი, სადაც ყველა შემთხვევისთვის გვაქვს 193 ცალი '0' და მხოლოდ ერთი '1' [46].

```
#CASE_TEMPLATE_ID სვეტის კატეგორიზაცია
df_with_CTID=pd.get_dummies(df_orig, columns=["CASE_TEMPLATE_ID"], prefix='CTID')
```

• PATIENT_AGE - ასაკის დასამუშავებლად მიზანშეწონილია ორი მეთოდის გამოყენება, რადგან წინასწარ არ ჩანს, თუ რომელი მეთოდი მოგვცემს უკეთეს შედეგს.

○წავშალეთ ანომალიები, ჩავატარეთ ასაკის კატეგორიზაცია One Hot Encoding მეთოდით, კატეგორიებად კი ავარჩიეთ შემდეგი გარდამტეხი ასაკები: 0 - 40 დღე, 1, 6, 15, 60 წლები.

```
# ასაკი დავყავით შემდეგ კატეგორიებად:
# 1-[0-40], 2-(40-1], 3-(1-6], 4-(6-15], 5-(15-60], 6-(60-120]
df['AGE_0_40'] = 0
df['AGE_40_1'] = 0
df['AGE_1_6'] = 0
df['AGE_6_15'] = 0
df['AGE_15_60'] = 0
df['AGE_60_120'] = 0

for i, row in df.iterrows():
  if math.isnan(row['INJURED_PERSON_AGE_DAYS'])==0 and math.isnan(row['INJURED_PERSON_AGE_MONTHS'])==0:
    if  0 <= row['INJURED_PERSON_AGE_DAYS'] + row['INJURED_PERSON_AGE_MONTHS'] * 30 <= 40 :
      df.loc[i,'AGE_0_40'] = 1
    elif  40 < row['INJURED_PERSON_AGE_DAYS'] + row['INJURED_PERSON_AGE_MONTHS'] * 30 <= 365 :
      df.loc[i,'AGE_40_1'] = 1
  elif math.isnan(row['INJURED_PERSON_AGE_DAYS'])==0 and math.isnan(row['INJURED_PERSON_AGE_MONTHS']):
    if  0 <= row['INJURED_PERSON_AGE_DAYS'] <= 40 :
      df.loc[i,'AGE_0_40'] = 1
    elif  40 < row['INJURED_PERSON_AGE_DAYS'] <= 365 :
      df.loc[i,'AGE_40_1'] = 1
  elif  math.isnan(row['INJURED_PERSON_AGE_DAYS']) and math.isnan(row['INJURED_PERSON_AGE_MONTHS'])==0 :
    if 0 <= row['INJURED_PERSON_AGE_MONTHS'] * 30 <= 40 :
      df.loc[i,'AGE_0_40'] = 1
```

```
    elif  40 <= row['INJURED_PERSON_AGE_MONTHS'] * 30 <= 365:
        df.loc[i,'AGE_40_1'] = 1
    elif 12 < row['INJURED_PERSON_AGE_MONTHS'] <= 72 :
        df.loc[i,'AGE_1_6'] = 1
    elif 72 < row['INJURED_PERSON_AGE_MONTHS'] <= 180 :
        df.loc[i,'AGE_6_15'] = 1
    elif 180 < row['INJURED_PERSON_AGE_MONTHS'] <= 720 :
        df.loc[i,'AGE_15_60'] = 1
    elif 720 < row['INJURED_PERSON_AGE_MONTHS'] <= 1500 :
        df.loc[i,'AGE_60_120'] = 1
```

○მივდივართ მეორე მიმართულებითაც,- არ ვატარებთ ასაკის კატეგორიზაციას, ვინახავთ ასაკს დღეების სახით და ვუკეთებთ ნორმალიზებას შემდეგი მეთოდით:

$X_n = (X - X_{min}) / (X_{max} - X_{min})$, (1)

სადაც X_n პაციენტის ასაკის ნორმალიზებულ მნიშვნელობას აღნიშნავს

• INJURED_PERSON_GENDER - შევცვალეთ NaN (არ არსებული) მნიშვნელობები საშუალოთი.

```
#გაურკვეველი სქესის შემთხვევაში ვიყენებთ საშუალოს
df['INJURED_PERSON_GENDER'].fillna(df['INJURED_PERSON_GENDER'].mean(),
inplace=True)
```

• CATEGORIES - განვაცალკევეთ ერთად შენახული კატეგორიები და ჩავატარეთ კატეგორიზაცია One Hot Encoding მეთოდით. 8 განსხვავებული კატეგორიიდან დადებითადაა შევსებული ყველა ის სვეტი, სადაც საქმესთან დაკავშირებულია შესაბამისი კატეგორია (სასწრაფო, სახანძრო, პოლიცია, კრიმინალური...)

```
#CATEGORIES სვეტის დაჭრა და შემდეგ კატეგორიზაცია
df_CAT_TMP = df_orig.set_index('CASE_ID').CATEGORIES.str.split(r',', expand=True).stack().re
set_index(level=1, drop=True).to_frame('CATEGORIES');
df_only_CAT=pd.get_dummies(df_CAT_TMP, prefix='CAT', columns=['CATEGORIES']).groupby
(level=0).sum()
```

```
# ორი dataframe-ის გაერთიანება
df = pd.merge(df_with_CTID, df_only_CAT, on=df_with_CTID.CASE_ID, how='inner')
```

- SPENT_MINS – უცვლელია
- PREV_HOSP_COUNT - უცვლელია
- LASTHOSPDD – უცვლელია
- CALL_COUNT - უცვლელია
- HOSPITALISATION - მოხდა თუ არა პაციენტის ჰოსპიტალიზაცია. ჰოსპიტალიზაციის შემთხვევაში SPENT_MINS პარამეტრს ემატება ჰოსპიტალიზაციისთვის დახარჯული დროც.

```
# ჰოსპიტალიზაციის სვეტის გენერაცია
# ბოლო ჰოსპიტალიზაციის დრო თუ მიმდინარე გამოძახების თარიღს ემთხვევა,
 ვთვლით, რომ გამოძახება ჰოსპიტალიზაციით დასრულდა.
df['HOSPITALISATION'] = np.where(df['LASTHOSPDD']==0, '1', '0')


# დუბლირებული მონაცემების თავიდან ასაცილებლად, თუ მიმდინარე გამოძახება ჰოსპ
იტალიზაციით დასრულდა (row['LASTHOSPDD']==0),
#ძველი ჰოსპიტალიზაციების რაოდენობას ვაკლებთ ერთს.
for i, row in df.iterrows():
   if row['LASTHOSPDD']==0:
     df.loc[i,'PREV_HOSP_COUNT'] -= 1
     df.loc[i,'LASTHOSPDD'] = np.nan
   if df.loc[i,'PREV_HOSP_COUNT'] == 0:
      df.loc[i,'PREV_HOSP_COUNT'] = np.nan
```

ბოლოს, Dataset-დან ვშლით მანქანური დასწავლისვის არასაჭირო სვეტებს.

```
del df['CATEGORIES']
del df['INJURED_PERSON_AGE_DAYS']
del df['INJURED_PERSON_AGE_MONTHS']
del df['CASE_ID']
del df['CALL_ID']
del df['DATE_CREATED']
del df['NOM']
```

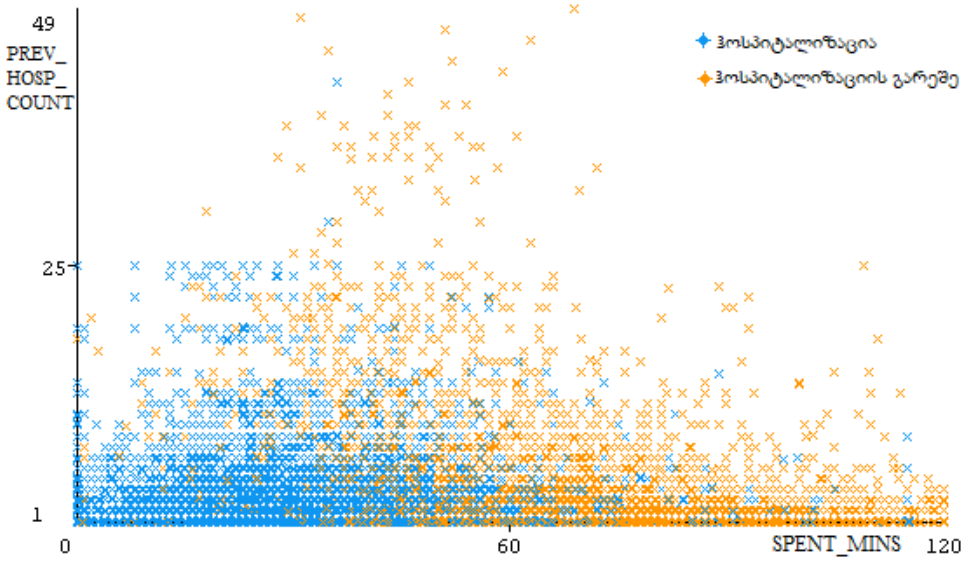
3.5. მონაცემთა ანალიზი და ვიზუალიზაცია

მიღებულ Dataset-ზე ერთდროულად ორი მიმართულებით ვმუშაობთ, Python პროგრამირება და Weka ხელსაწყო.

Weka წარმოადგენს მანქანური დასწავლის ალგორითმების ნაკრებს. ასევე, შეიცავს ისეთ საშუალებებს, რომლებიც გამოიყენება მონაცემთა წინასწარი დამუშავებისთვის, კლასიფიკაციისათვის, რეგრესიისათვის, დაჯგუფებისათვის, მონაცემთა ასოციაციის წესების დასადგენად და ვიზუალიზაციისათვის [47].

მონაცემების კორელაციაზე წარმოდგენის შესაქმნელად გამოვიყენეთ Weka-ს ვიზუალიზაციის ხელსაწყოები. ქვემოთ მოყვანილია ვიზუალიზაციის რამდენიმე ისეთი ფრაგმენტი, რამაც მეტ-ნაკლებად უნდა მოგვცეს მონაცემთა ურთიერთ დამოკიდებულებაზე წარმოდგენა.

ნახაზ 28-ზე ასახულია სასწრაფოს ბრიგადის მიერ პაციენტთან გატარებული დროისა და წინა ჰოსპიტალიზაციების რაოდენობების ურთიერთ დამოკიდებულება (პაციენტი იგივე ნომრით და იმავე ინციდენტის ტიპით). თითოეული წერტილი გრაფიკზე ასახავს ცალკეულ შემთხვევას. ფორთოხლისფრად გაფერადებულია ის საქმეები, რომლებიც დასრულდა ჰოსპიტალიზაციით.

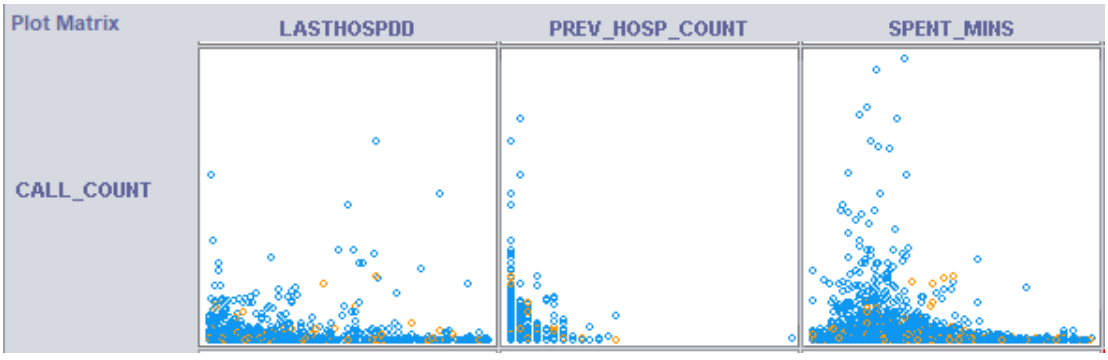


ნახ. 28. სასწრაფოს ბრიგადის მიერ პაციენტთან გატარებული დროისა, წინა ჰოსპიტალიზაციის შემთხვევების რაოდენობის და ჰოსპიტალიზაციის მოხდენის ურთიერთ დამოკიდებულება

როგორც ვიზუალიზაციიდან ჩანს, მონაცემების დისპერსია (გაბნევა) დიდია, ამიტომ გადაწყდა, მონაცემთა კრებული ბოლო 5 წლის მონაცემებით შემდგარიყო.

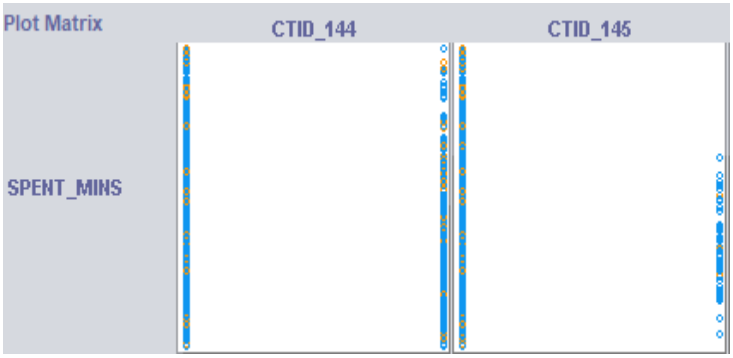
საინტერესოა აღინიშნოს, რომ უკუპროპორციული დამოკიდებულება გამოვლინდა გამოძახებების რიცხვსა და ჰოსპიტალიზაციის წინა შემთხვევების რაოდენობას შორის (იხ. შუა გრაფიკი ნახ. 29): რაც მეტია გამოძახებათა რიცხვი, მით ნაკლებია ჰოსპიტალიზაციის წინა შემთხვევების რაოდენობა. ეს შეიძლება აიხსნას ორი ტიპის პაციენტების არსებობით. ერთის მხრივ, არსებობს ბევრი ისეთი პაციენტი, რომელიც ხშირად იძახებს სასწრაფო დახმარებას, მაგრამ, ჩვეულებრივ საჭიროებს არა ჰოსპიტალიზაციას, არამედ მხოლოდ ფსიქოლოგიურ მხარდაჭერას (ნუგეშს, დამშვიდებას). მეორეს მხრივ, არსებობენ პაციენტები, რომლებიც უკვე იყვნენ მოხვედრილი გარკვეულ კრიტიკულ სიტუაციაში და საჭირო გახდა მათი ჰოსპიტალიზაცია. ამის შემდეგ კი, მიღებული გამოცდილების გათვალისწინებით, ისინი ახერხებენ თავიანთი ჯანმრთელობის პრობლემების მოგვარებას ისე, რომ ჩვეულებრივ, სასწრაფო დახმარებას გაცილებით იშვიათად იძახებენ და თანაც, მხოლოდ მაშინ, როცა მათი ჯანმრთელობის მდგომარეობა მართლაც კრიტიკულია. ასე რომ, ასეთი გამოძახებები ძირითადად ჰოსპიტალიზაციით მთავრდება.

მარჯვენა გრაფიკიდან ჩანს, რომ წინა გამოძახებათა შემთხვევების დიდი რაოდენობის არსებობისას სასწრაფო დახმარების ბრიგადა შემთხვევაზე 20-30 წუთს ხარჯავს.



ნახ. 29. გამოძახებათა რაოდენობის კავშირი სხვადასხვა პარამეტრთან

კავშირი ინციდენტის ტიპებსა და სასწრაფოს ბრიგადის მიერ შემთხვევაზე დახარჯულ დროს შორის ასახულია ნახაზ 30-ზე. მარცხენა და მარჯვენა გრაფიკებზე წარმოდგენილია ორი სხვადასხვა ინციდენტის ტიპი (კოდებით 144 და 145). მარცხენა, ნულოვანი სვეტები გრაფიკებში შეესატყვისება ყველა იმ ინციდენტის ტიპს, რომლებიც განსხვავებულია არჩეული ტიპებისგან. როგორც ვხედავთ, დროის მონაცემები ყველა ინციდენტის ტიპისთვის განაწილებულია დროის მთელ შუალედზე, მარჯვენა სვეტები კი გვიჩვენებს კონკრეტული ინციდენტის ტიპების შემთხვევაში დახარჯული დროის მონაცემების განაწილებას. ნახაზ 31-დან ჩანს, რომ სხვადასხვა ინციდენტის ტიპის შესატყვისი მონაცემების განაწილება განსხვავებულია ერთმანეთისგან და განსხვავებულია ზოგადი განაწილებისაგან. ასე რომ, შესაძლებელია, ინციდენტის ყოველი ტიპი დახასიათდეს საშუალო დროის სიდიდით და განაწილების სახით [43].



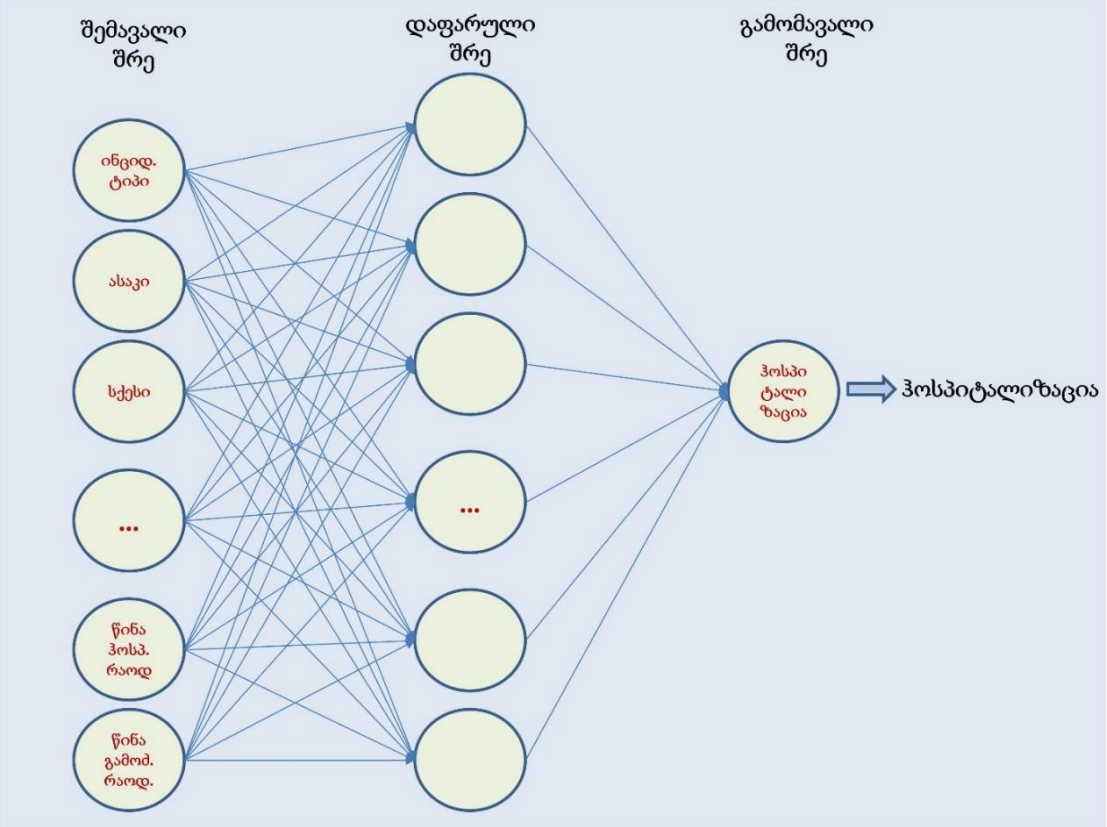
ნახ. 30. კავშირი ინციდენტის ტიპსა და სასწრაფოს ბრიგადის გათავისუფლებას შორის

3.6. სასწრაფო დახმარების ბრიგადის მიერ დახარჯული დროის შეფასება

სასწრაფო დახმარების ბრიგადის მიერ სავარაუდოდ საჭირო დროის შეფასების ამოცანა მოსახერხებელია, დაიყოს კლასიფიკაციისა და რეგრესიის ქვეამოცანად.

3.6.1. კლასიფიკაციის ამოცანა. პასუხი - ჰოსპიტალიზაციის ალბათობა

სასწრაფო დახმარების ბრიგადის მიერ დახარჯული დროის შეფასების პირველ ეტაპზე საჭიროა პაციენტის ჰოსპიტალიზაციის ალბათობის გამოთვლა. ვითვალისწინებთ რა იმ ფაქტს, რომ პაციენტის ჰოსპიტალიზაციის შესახებ გადაწყვეტილების მიღება ყოველთვის ხდება სასწრაფო დახმარების ექიმის მიერ, შეუძლებელია, 112-ის დისპეტჩერმა წინასწარ, ზუსტად იცოდეს, მოხდება თუ არა პაციენტის ჰოსპიტალიზაცია, ამიტომ, ალგორითმის გამოსავალი არის პაციენტის ჰოსპიტალიზაციის ალბათობის სიდიდე. ამ ამოცანის გადასაწყვეტად ვაპირებთ ხელოვნური ნეირონული ქსელის გამოყენებას. ნახაზ 31-ზე წარმოდგენილია ხელოვნური ნეირონული ქსელის არქიტექტურა, რომელიც გამოყენებული იქნება ამოცანის გადასაწყვეტად.



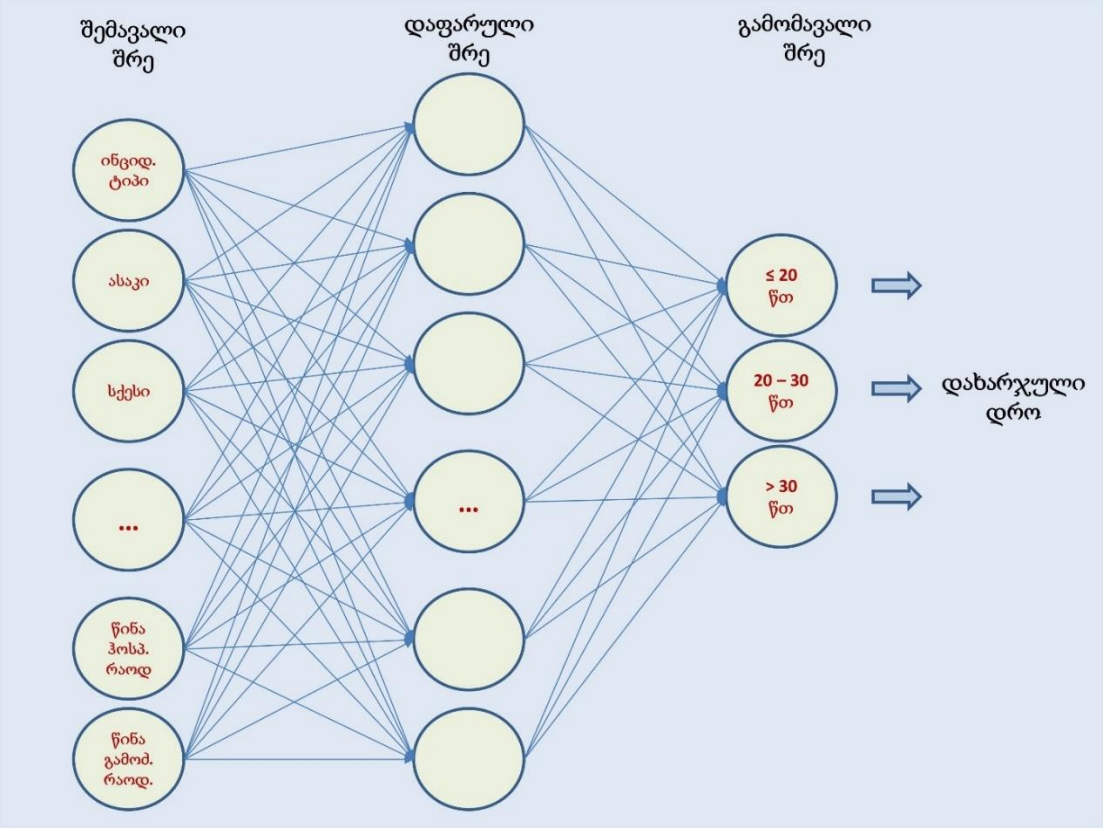
ნახ. 31. პაციენტის ჰოსპიტალიზაციის ალბათობის გამოთვლა

ნეირონული ქსელის შემავალი შრის ელემენტები წარმოადგენს წინასწარ შემუშავებული პარამეტრების ერთობლიობას. ინციდენტის ტიპში იგულისხმება 194 განსხვავებული „CASE_TEMPLATE_ID“ კოდის კატეგორიზირებული პარამეტრი. პარამეტრი ასაკი გულისხმობს 6 შემავალ შუალედს და ასე შემდეგ, ისე, როგორც, აღწერილია მონაცემთა წინასწარი დამუშავების სისტემაში.

რამდენადაც გადაწყვეტილების მიღების მომენტში არ გვაქვს ინფორმაცია მოცემული პაციენტის შემთხვევაში საჭირო დასახარჯ დროზე, შემავალ შრეში წარმოდგენილია ყველა სვეტი, გარდა დახარჯული დროის შესაბამისი სვეტისა.(“SPENT_MINS”).

3.6.2. რეგრესიის ამოცანა. პასუხი - სავარაუდო დასახარჯი დრო

იმ შემთხვევაში, თუ პაციენტის ჰოსპიტალიზაციის ალბათობა დაბალია ($P<0.5$) უნდა შეფასდეს სასწრაფო დახმარების ბრიგადის მიერ მოცემულ პაციენტზე სავარაუდოდ დასახარჯი დრო (ნახ.32).



ნახ. 32. სასწრაფო დახმარების ბრიგადის მიერ პაციენტის მისამართზე სავარაუდოდ დასახარჯი დროის შეფასება

დასწავლისთვის გამოსაყენებელი მონაცემთა კრებული შედგება მხოლოდ იმ შემთხვევებისგან, რომლებიც არ დასრულდა ჰოსპიტალიზაციით. დასწავლის დროს გამოვიყენებთ ყველა სვეტს.

სასწრაფო დახმარების ბრიგადის მიერ პაციენტის მისამართზე სავარაუდოდ საჭირო დასახარჯი დროის მხოლოდ სამი შუალედია ნაჩვენები ნეირონული ქსელის გამომავალ შრეში. ვფიქრობ, რომ დროის უფრო მეტი შუალედების განხილვა მნიშვნელოვნად გაზრდიდა შეცდომების ალბათობას.

გამომავალი შრის შედეგების გამოსავლენად, რომლებიც მაქსიმალურად ახლო იქნება რეალური დროის დანახარჯებთან, აუცილებელია, შევიმუშავოთ ნეირონული ქსელის ფარული შრის ოპტიმალური არქიტექტურა [48].

კვლევის ამ ეტაპზე ძნელია წინასწარ განჭვრეტა ყველა იმ მეთოდისა, რაც შეიძლება ოპტიმალური აღმოჩნდეს მუშაობის პროცესში. კვლევის პროცესში შესაძლოა, არსებული შეხედულებები შეიცვალოს. სამუშაოს მასშტაბის გათვალისწინებით 112-ის ცენტრთან ხანგრძლივი თანამშრომლობაა დაგეგმილი კვლევის წარმატებით დასასრულებლად.

3.7. მესამე თავის დასკვნა

გაანალიზდა გადაუდებელი სამედიცინო სამსახურის გამოწვევები და გამოვლინდა ამოცანები კომპიუტერული მეთოდების გამოყენებისათვის.

გადაუდებელი სამედიცინო სამსახურის დისპეტჩერიზაციის გასაუმჯობესებლად გადაწყდა ხელოვნური ნეირონული ქსელის გამოყენება და დაიგეგმა ამ ამოცანის გადაწყვეტის ეტაპები.

პროგრამირების ენა Python-ის საშუალებით შესრულდა extract, transform, load (ETL) და მონაცემთა წინასწარი დამუშავების (data preprocessing) პროცესები, რის შედეგადაც მიღებულ იქნა მანქანური დასწავლისთვის საჭირო სტრუქტურის მქონე მონაცემთა კრებული ბოლო 5 წლის გადაუდებელი სამედიცინო დახმარების შემთხვევების ბაზაზე.

შემდგომ ეტაპზე Python პროგრამირებისა და Weka ხელსაწყოს გამოყენებით გაანალიზებულ იქნა მიღებული მონაცემთა ნაკრები. გამოვლინდა, წაიშალა და შესწორდა ანომალიური შემთხვევები, რაც გაუთვალისწინებლობის დროს უარყოფითად აისახებოდა მანქანური დასწავლის შედეგებზე.

4 თავი. ექსპერიმენტული ნაწილი: სასწრაფო დახმარების გამოძახებების პრიორიტეტების განსაზღვრის ამოცანა. დისპეტჩერიზაციის მოდელირება გრიპის სეზონური ეპიდემიის დროს.

4.1. დისპეტჩერიზაციის ამოცანის სირთულე გრიპის სეზონური ეპიდემიის დროს

გრიპის ეპიდემიის დროს გადაუდებელი დახმარების ოპერატიულ მართვის ცენტრ „112“-ში შემოდის ერთი და იმავე ინციდენტის ტიპის - გრიპისა და მსგავსი ტიპის ინფექციების მქონე ათობით სამედიცინო გამოძახება. სასწრაფო სამედიცინო მომსახურების რიგითობის განსასაზღვრად დისპეტჩერს უწევს გამოძახებების სათითაოდ შემოწმება და შედარება, რაც დიდ დროსა და ძალისხმევას მოითხოვს მაშინ, როდესაც წამები გადამწყვეტია. ამასთან, გამოძახებების რაოდენობა არსებულ ბრიგადებთან შედარებით ბევრად მეტია და ხშირად ერთი და იმავე ტიპის 50-მდე „ჩამოკიდებული“ საქმე ელოდება ბრიგადის განთავისუფლებას.

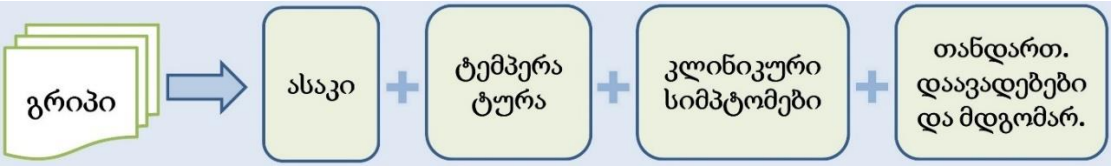
ამგვარ სტრესულ გარემოში მუშაობისას, ბუნებრივია, იზრდება არასწორი გადაწყვეტილების მიღების ალბათობა [49].

4.2. დისპეტჩერიზაციის მოდელირება

დისპეტჩერიზაციის პროცესის გასაუმჯობესებლად შემუშავებულ იქნა მოდელი თითოეული გამოძახების შემთხვევის გადაუდებლობის ობიექტური შეფასებისათვის.

აუცილებელი სამედიცინო ინფორმაციის ამოღება ხდება სტანდარტული გამოძახების ბარათებიდან, რაც რეგისტრირდება გამოძახების მიმღები ოპერატორის მიერ. გამოცდილ ექიმებთან კონსულტაციის შემდეგ შეირჩა გრიპის შემთხვევების შემდეგი ობიექტური მახასიათებლები (ნახ.33):

- პაციენტის ასაკი,
- ტემპერატურა,
- კლინიკური სიმპტომები (სუნთქვის გაძნელება, ტკივილი გულის არეში, ღებინება, დიარეა) [50,51]. ხშირია შემთხვევა, როდესაც ზარის ინიციატორი რამოდენიმე კლინიკურ სიმპტომს ასახელებს, შესაძლოა, არაობიექტურადაც, მდგომარეობის უფრო მძიმედ და გადაუდებლად წარმოსაჩენად. ამიტომ, ოპერატორი, ჩვეულებრივ, აფიქსირებს მთავარ, ანუ სიცოცხლისათვის უფრო მაღალი რისკის შემცველ კლინიკურ სიმპტომს.
- თანმხლები ქრონიკული დაავადებები და მდგომარეობები, რომლებიც ზრდიან გრიპის გართულების რისკ ფაქტორს (ასთმა, ფილტვების ქრონიკული დაავადება, ორსულობა, ნევროლოგიური მდგომარეობები, გულის დაავადებები და ჰიპერტონია, დიაბეტი, იმუნოდეფიციტური მდგომარეობები, თირკმლის/ღვიძლის პათოლოგიები, თავის ძლიერი ტკივილი ან სხვა პრობლემა) [50].



ნახ. 33. გრიპის შემთხვევების მახასიათებლები

ამ მახასიათებლების ობიექტური შეფასებისათვის შემოვიღეთ პარამეტრების სისტემა, რომლებიც მოყვანილია ცხრილებში 4.1-4.2.

როგორც ცნობილია, გრიპის ინფექცია ყველაზე სახიფათოა ახალშობილებისათვის და საკმაოდ სახიფათოა ორ წლამდე ასაკის ბავშვებისათვის. ასაკის მიხედვით რისკ ფაქტორი (წონა) შედარებით მაღალია 5 წლამდე ასაკის ბავშვებისა [52] და ხანში შესული (65 წელზე უფროსი) პაციენტებისათვის. ამიტომ, გრიპის შესაძლო გართულების ასაკით გამოწვეული რისკ ფაქტორის r_a შეფასება სხვადასხვა ასაკობრივი ჯგუფებისთვის განსხვავებულად ხდება (იხ. ცხრილი 10).

ცხრ. 9. რისკ ფაქტორი r_a სხვადასხვა ასაკობრივი ჯგუფისათვის

ასაკი t წ	ახალშობილები t ≤ 0.11 (40 დღ)	ბავშვები 2 წლამდე 0.1< t ≤ 2	ბავშვები 5 წლამდე 2 < t ≤ 5	ბავშვები და ზრდასრულები 5 < t ≤ 65	ხანში შესულები t > 65
r_a %	30	25	18	10	17
r_a	3	2.5	1.8	1	1.7

ქვემოთ მოყვანილია პროგრამული უზრუნველყოფის ფრაგმენტი ასაკის წონების მისანიჭებლად Python ენაზე:

```
if 0 <= row['AGE'] <= 0.11 :   # ახალშობილები
    df.loc[i,'AGE_CAT'] = 30
elif 0.11 < row['AGE'] <= 2:   # ბავშვები 2 წლამდე
    df.loc[i,'AGE_CAT'] = 25
elif 2 < row['AGE'] <= 5:       # ბავშვები 2-დან 5 წლამდე
    df.loc[i,'AGE_CAT'] = 18
elif 5 < row['AGE'] <= 65:    #სკოლის ასაკის და ზრდასრული პაციენტები
    df.loc[i,'AGE_CAT'] = 10
elif 65 < row['AGE']:         # ხანში შესული პაციენტები
    df.loc[i,'AGE_CAT'] = 17
```

რამდენადაც ტემპერატურის აწევა ზრდის გრიპის გართულების რისკს, განსაკუთრებით ზოგიერთი ქრონიკული დაავადების ან მდგომარეობის დროს [53], ტემპერატურით გამოწვეული რისკ ფაქტორის სიდიდე r_c განსხვავებულად არის შეფასებული (ცხრ.11).

ცხრ. 10. რისკ ფაქტორი r_c განსხვავებული ტემპერატურის შემთხვევაში

T °C	T ≤ 35	35 < T ≤ 37	37 < T ≤ 38	38 < T ≤ 39	T > 39°C
r_c %	30	0	10	20	40

პროგრამული კოდის ფრაგმენტი ტემპერატურის წონების მისანიჭებლად:

```
if 0 <= row['TEMP'] <= 35 :        # კრიტიკულად დაბალი სიცხე
    df.loc[i,'TEMP_CAT'] = 30
elif 35 < row['TEMP'] <= 37 :      # ნორმალური ტემპერატურა
    df.loc[i,'TEMP_CAT'] = 0
elif  37 < row['TEMP'] <= 38:      # 37 < T ≤ 38
    df.loc[i,'TEMP_CAT'] = 10
elif  38 < row['TEMP'] <= 39:      # 38 < T ≤ 39
    df.loc[i,'TEMP_CAT'] = 20
elif  39 < row['TEMP']:            # კრიტიკულად მაღალი სიცხე
    df.loc[i,'TEMP_CAT'] = 40
```

რაც შეეხება კლინიკურ სიმპტომებს, ზოგიერთი მათგანი ჩვეულებრივი თანმდევია გრიპის ინფექციისა და რაიმე განსაკუთრებულ საშიშროებასთან არ არის დაკავშირებული (სურდო, ხველა, თავის ტკივილი...), მაგრამ გამოსაყოფი და საყურადღებოა რამოდენიმე კლინიკური სიმპტომი, რომლებიც შესაძლოა გახდეს პაციენტის მდგომარეობის სერიოზული გართულების მიზეზი [52,54]. დისპეტჩერიზაციის მოდელში ამ სიმპტომებს განსაკუთრებულ ყურადღებას ვაქცევთ (იხილეთ ცხრილი 12).

ცხრილში მოცემულია ამ სიმპტომების არსებობისას გრიპის შესაძლო გართულების რისკ ფაქტორის მნიშვნელობები r_s ბავშვთა ასაკის თავისებურებების გათვალისწინებით.

ცხრ. 11. რისკ ფაქტორის სიდიდე r_s კლინიკური სიმპტომებისათვის

კლინიკური სიმპტომები		ტკივილი გულის არეში	დიარეა	ღებინება	სუნთქვის გაძნელება
r_s %	ბავშვები $t \le 2$წ	-	30	30	40
	უფროსები $t > 2$წ	35	15	10	40

აქვე, იხილეთ შესაბამისი კოდის ფრაგმენტი:

```
if 0 <= row['AGE'] <= 2 :          # ბავშვები t ≤ 2წ
   if  row['SYMP'] == "Vomit.":
      df.loc[i,'SYMP_CAT'] = 30
   elif  row['SYMP'] == "Diar.":
      df.loc[i,'SYMP_CAT'] = 30
   elif  row['SYMP'] == "Sh_Breath":
      df.loc[i,'SYMP_CAT'] = 40
elif 2 < row['AGE']:               # უფროსები t > 2წ
   if  row['SYMP'] == "Vomit.":
      df.loc[i,'SYMP_CAT'] = 10
   elif  row['SYMP'] == "Diar.":
      df.loc[i,'SYMP_CAT'] = 15
   elif  row['SYMP'] == "Sh_Breath":
      df.loc[i,'SYMP_CAT'] = 40
   elif  row['SYMP'] == "Heart_Pain":
      df.loc[i,'SYMP_CAT'] = 35
```

თუ გრიპის შემთხვევას თან ახლავს ზოგიერთი ქრონიკული დაავადება და მდგომარეობა, გართულების რისკი იზრდება [55,56] თუ პაციენტი დროულად არ იქნა მიხედული. შემოთავაზებული მოდელი ითვალისწინებს ასეთი შემთხვევების მომსახურების გადაუდებლობის წონის გაზრდას. ცხრილ 13-ში მოცემულია გრიპის თანმხლები ქრონიკული დაავადებები, მდგომარეობები და მათი შესაბამისი წონები გადაუდებელი მომსახურებისათვის. წონა `rd` მიჩნეულია ნულის ტოლად, თუ პაცენტი არ აღნიშნავს თანმხლებ სირთულეებს.

ცხრ. 12. რისკ ფაქტორი rd გრიპის თანმხლები დაავადებებისა და განსაკუთრებული მდგომარეობის დროს

თანმხლები ქრონიკული დაავადებები და მდგომარეობები	წონა rd
ასთმა	50
ფილტვების ქრონიკული დაავადება	45
ორსულობა	40
ნევროლოგიური მდგომარეობები	35
გულის დაავადებები და ჰიპერტონია	30
დიაბეტი	25
იმუნოდეფიციტ. მდგომარეობები	20
თირკმლის/ღვიძლის პათოლოგიები	15
თავის ძლიერი ტკივილი ან სხვა პრობლემა	10

აქვე, მომოვიყვანოთ პროგრამული კოდის შესაბამისი ფრაგმენტი:

```
if   row['COND'] == "None":          # არ აღნიშნავს თანმხლებ სირთულეებს
   df.loc[i,'COND_CAT'] = 0
elif  row['COND'] == "Asthma":
   df.loc[i,'COND_CAT'] = 50
elif  row['COND'] == "Chr.Lung.Dis.":
   df.loc[i,'COND_CAT'] = 45
elif  row['COND'] == "Pregn.":
   df.loc[i,'COND_CAT'] = 40
elif  row['COND'] == "Neurol.Cond.":
   df.loc[i,'COND_CAT'] = 35
elif  row['COND'] == "Heart_Dis.":
   df.loc[i,'COND_CAT'] = 30
elif  row['COND'] == "Diabet":
   df.loc[i,'COND_CAT'] = 25
elif  row['COND'] == "Immunodeficiency":
   df.loc[i,'COND_CAT'] = 20
elif  row['COND'] == "Kidny_liver_Dis.":
   df.loc[i,'COND_CAT'] = 15
```

```
else:
    df.loc[i,'COND_CAT'] = 10      # თავის ძლიერი ტკივილი ან სხვა პრ.
```

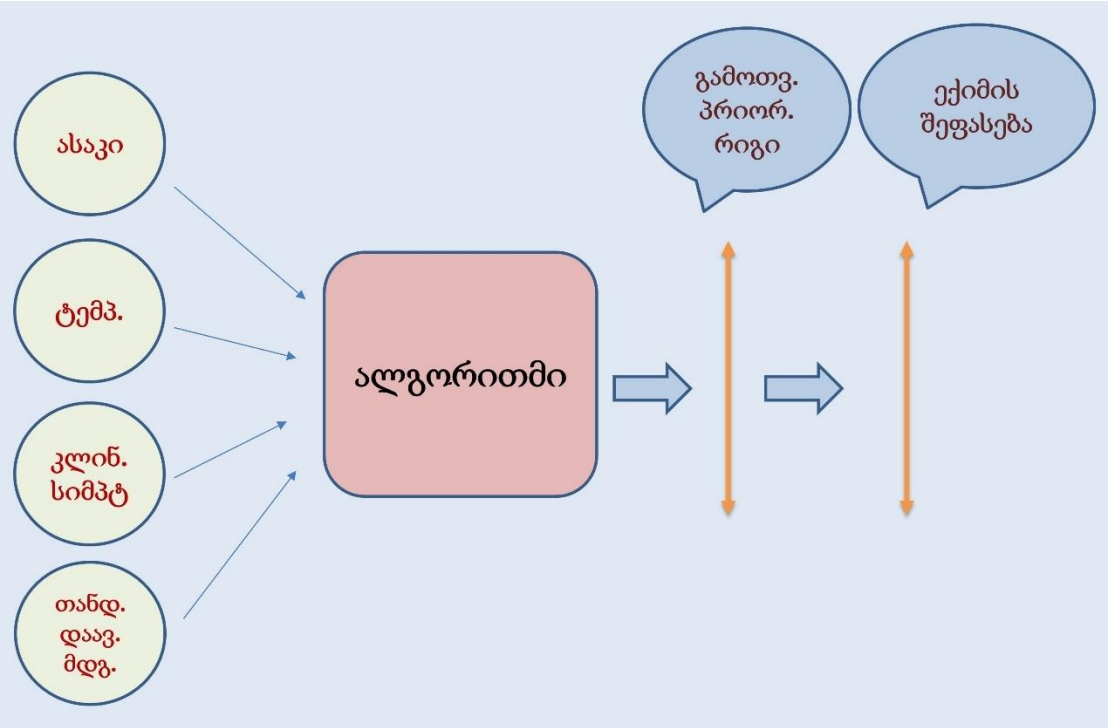
გამოძახების ბარათზე შეიძლება აღნიშნული იყოს ერთზე მეტი კლინიკური სიმპტომი და/ან ერთზე მეტი თანმხლები ქრონიკული დაავადება ან განსაკუთრებული მდგომარეობა. ასეთ შემთხვევაში დისპეტჩერიზაციის მოდელში ხდება აღმატებული წონების მქონე კლინიკური სიმპტომისა თუ ქრონიკული დაავადების ან განსაკუთრებული მდგომარეობის გათვალისწინება.

მომავალში დისპეტჩერიზაციის მოდელის გამოცდისას დიდი რაოდენობის რეალურ შემთხვევების სიმრავლეზე შესაძლოა მასში შევიტანოთ ცვლილებები, მაგალითად, მაქსიმალურად წონადი ერთი კლინიკური სიმპტომის ნაცვლად შევიტანოთ ორი ყველაზე მნიშვნელოვანი კლინიკური სიმპტომის წონების ჯამი. ეს გამოიწვევს ასეთი შემთხვევების მომსახურების გადაუდებლობის ინდექსის გაზრდას, რაც შესაბამისი იქნება შემთხვევის მაღალი პრიორიტეტულობისა.

გრიპის შემთხვევებზე შემოსული გამოძახებების ტიპიური ნიმუშების მიხედვით შედგენილია 25 შემთხვევიანი საცდელი მოდელი. ამ მოდელზე გამოიცადა სხვადასხვა ალგორითმი გამოძახების მომსახურების გადაუდებლობის ინდექსის ანუ **r** რისკ-ფაქტორის გამოსათვლელად. იხილეთ სტატიები [57] და დასაბეჭდად გამზადებული Chogovadze G., **Kiviladze G.**, Surguladze G. Modeling of Emergency Medical Service in Flu Season:Algorithm for Dispatching Ambulance Units. *Bulletin of Georgian National Academy of Sciences*. 2018,12(3).

სხვადასხვა ალგორითმის ობიექტურობის გამოსავლენად გამოთვლის შედეგები შედარდა გამოცდილი ექიმის მიერ საცდელი მოდელის შემთხვევების გადაუდებლობის ინდექსის შეფასებებთან D_m (ნახ.34.). D_m სიდიდე განისაზღვრება 10 ქულიანი სისტემით:

$$D_m = \{1,2,...,10\}.$$



ნახ. 34. გრიპის შემთხვევათა პრიორიტეტულობის დადგენის სქემა

საცდელი მოდელის შემთხვევებისთვის შედგა მომსახურების პრიორიტეტულობის რიგიც: თითოეული შემთხვევის D_m ქულის მიხედვით მიენიჭა პრიორიტეტულობის მაჩვენებელი P_m.

i-ური ალგორითმის მიხედვით გამოთვლილი მომსახურების გადაუდებლობის ინდექსის (r_i) მნიშვნელობების საფუძველზე ხდება შემთხვევათა პრიორიტეტულობის რიგის დადგენა

$$P_i = \{1,2,...,25\}.$$

ალგორითმის შერჩევისას განსაკუთრებული ყურადღება ექცეოდა ყველაზე უფრო პრიორიტეტული შემთხვევების გამოვლენას. უპირატესობა ენიჭებოდა იმ ალგორითმს, რომელიც გამოავლენდა ექიმის მიერ მითითებულ მაღალი გადაუდებლობის ინდექსის მქონე ყველა შემთხვევას, ან ასეთი შემთხვევების უმეტესობას.

გამოძახების თითოეული შემთხვევის გადაუდებლობის ინდექსის გამოსათვლელად შეირჩა ორი ალგორითმი:

Alg 1: $r_1 = r_a + r_c + r_s + r_d$ (1)

სადაც რისკ-ფაქტორი ასაკობრივი ჯგუფების მიხედვით r_a მოცემულია პროცენტებში (იხილეთ r_a % ცრილ 10-ში).

გარკვეული ასაკობრივი ჯგუფების (ახალშობილები, 2 წლამდე და 5 წლამდე ასაკის ბავშვები და 65 წელზე უფროსი პაცენტები) r_a აღმატებული რისკ-ფაქტორის უკეთ გასათვალისწინებლად (გასამძაფრებლად) შემუშავებულ იქნა მეორე ალგორითმი:

Alg 2: $r_2 = r_a (r_c + r_s + r_d)$ (2)

სადაც $r_a > 1$ (იხილეთ ცხრილი 4.1., ქვედა მწკრივი),

$r_a = \{3; 2.5; \ldots\}$.

გადაუდებლობის ინდექსის გამოთვლილი მნიშვნელობები r_1 და r_2 ექიმის შეფასების D_m ინდექსთან ერთად მოთავსებულია ცხრილში 14.

როგორც ცხრილ 14-დან ჩანს, საცდელ მოდელში არსებული გადაუდებლობის ყველაზე მაღალი ინდექსის მქონე შემთხვევები, ექიმის მიერ შეფასებული $D_m = 10$ (პაციენტები 1;2;9;10;13;14;17;22;24) გამოვლინდა შერჩეული ალგორითმების გამოყენებით. Alg 1 –ის მიხედვით, მე-14 პაციენტის გარდა ყველა ზემოთ აღნიშნულ შემთხვევაში მომსახურების გადაუდებლობის ინდექსი $r_1 \geq 95$.

Alg 2-ის მიხედვით გამოვლინდება გადაუდებლობის მაქსიმალური ინდექსის მქონე პაციენტები {9;13;14;17}. ამ შემთხვევებისათვის $r_2 \geq 100$.

როგორც ზემოთ აღვნიშნეთ, Alg 2 გამოიყენება მხოლოდ იმ შემთხვევებში, როცა $r_a > 1$. როგორც ვხედავთ, Alg 2-ის დახმარებით პაციენტი 14-იც გამოვლინდა ყველაზე კრიტიკული ჯგუფის წევრად.

როგორც ვხედავთ, Alg 1 და Alg 2 ალგორითმების ერთობლივმა გამოყენებამ საშუალება მოგვცა, გამოგვევლინა გადაუდებლობის მაქსიმალური ინდექსის მქონე ყველა შემთხვევა.

ცხრ. 13. გრიპის ტიპიური გამოძახებებით შედგენილი საცდელი მოდელი

პაციენტი	ასაკი წლები	ტემპ. °C	მთავარი კლინიკ. სიმპტ.	თანმხლები დაავად. და განსაკ. მდგომარ.	ექიმის შეფას. გადაუდებლ. ინდექს D_m 1-10	Alg 1 გამოთვ გადაუდებლ. ინდექს r_1	Alg 2 გამოთვ გადაუდებლ. ინდექს r_2
1	20	39.2	სუნთქ.გაძნ.	ასთმა	10	140	-
2	59	38.7	ტკივ. გულ.	გულის დაავ.	10	95	-
3	75	38.5		გულის დაავ.	7	67	85
4	33	38.8	დიარეა	იმუნოდეფ.	7	65	-
5	48	38.7		დიაბეტი	5	55	-
6	62	39.2			1-2	50	-
7	3	38.3	ღებინება		5	48	70
8	5	38.5	დიარეა		4	53	-
9	75	38.7	სუნთქ.გაძნ.	გულის დაავ.	10	107	153
10	61	39.2	სუნთქ.გაძნ.	გულის დაავ.	10	120	-
11	8	39.1	სუნთქ.გაძნ.		9	90	-
12	55	38.2		დიაბეტი	5	55	-
13	2.5	38.4	სუნთქ.გაძნ.	ნევრ. მდგ.	10	113	171
14	0.2	37.9	დიარეა		10	65	100
15	36	37.8		ორს.	5-6	60	-
16	80	34.5	დიარეა		6-7	62	76.5
17	0.1	38.9	სუნთქ.გაძნ.	გულის დაავ.	10	120	270
18	40	39.7	სუნთქ.გაძნ.		9	90	-
19	70	38.1	ღებინება		4-5	47	51
20	47	39.4		გულის დაავ.	8	80	-
21	66	37.7		დიაბეტი	4	52	59.5
22	39	38.4	სუნთქ.გაძნ.	დიაბეტი	9-10	95	-
23	50	38.3	ღებინება	გულის დაავ.	8	70	-
24	34	38.5	სუნთქ.გაძნ.	ფილტვ.ქრ.დ.	10	115	-
25	24	38.6	ღებინება	ორს.	8	80	-

4.3. დისპეტჩერიზაციის მოდელირების შედეგების ანალიზი

დისპეტჩერიზაციის მოდელირების შედეგად თითოეული შემთხვევისათვის გამოვითვალეთ მომსახურების გადაუდებლობის ინდექსი, რომელიც ობიექტურად ასახავს ექიმის შეფასებას. ალგორითმის ობიექტურობის ზუსტი შეფასებისათვის შევადგინეთ პრიორიტეტულობის რიგები:

- ექიმის შეფასების მიხედვით - P_m (D_m ქულების მიხედვით)
- Alg 1-ით გამოთვლილი - P_1 (გადაუდებლობის r_1 ინდექსის მიხედვით)
- Alg 1&2-ის გამოთვლებით შეჯერებული - P (გადაუდებლობის r_1 & r_2 ინდექსების მიხედვით)

პრიორიტეტულობის რიგების შედარების შედეგი მოცემულია ცხრილში 15.

როგორც ცხრილიდან ჩანს, ალგორითმების გამოყენების შდეგად გამოვლენილი პრიორიტეტულობის რიგი P უმნიშვნელოდ განსხვავდება ექიმის შეფასებების მიხედვით შედგენილი პრიორიტეტულობის P_m რიგისაგან: $D = |P_m - P| > 1$

მნიშვნელოვანი განსხვავება გვაქვს 25-დან მხოლოდ 5 შემთხვევაში. ამასთან, აღსანიშნავია, რომ შემთხვევათა კრიტიკული ჯგუფისათვის $D \leq 1$. ე.ი. გადაუდებლობის მაქსიმალური ინდექსის მქონე შემთხვევების გამოვლენა ალგორითმების გამოყენებით იძლევა იმავე შედეგს, რასაც ვიღებთ შემთხვევების პრიორიტეტულობის რიგის ექიმის შეფასებების მიხედვით შედგენისას.

ამგვარად, დისპეტჩერიზაციის მოდელირების შედეგად მივიღეთ ალგორითმების სისტემა, რომელიც საშუალებას იძლევა, მაღალი სიზუსტით გამოვარჩიოთ ყველაზე გადადუდებელი, ანუ კრიტიკული შემთხვევები გამოძახებათა მოცემული, საცდელი მოდელიდან.

საცდელი მოდელისათვის აღმოჩნდა, რომ კრიტიკულ ჯგუფში შემავალი შემთხვევებისათვის $r_1 \geq 95$ და/ან $r_2 \geq 100$.

ცხრ. 14. ალგორითმების გამოყენებით მიღებული პრიორიტეტულობის რიგი

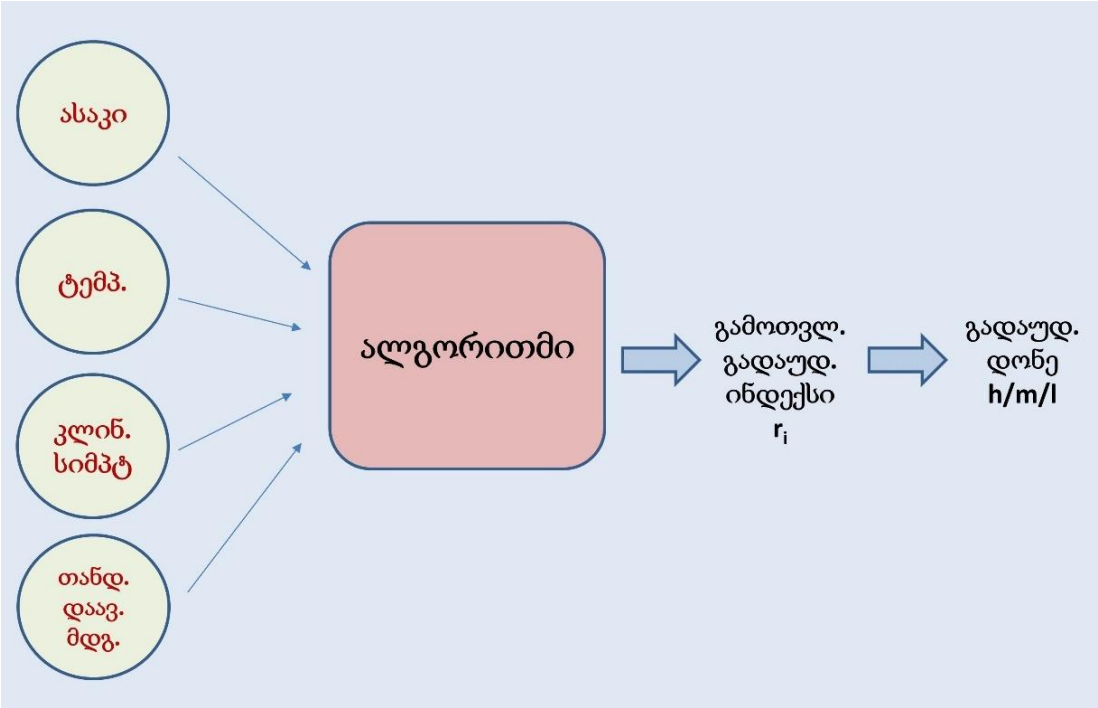
პაციენტი	ასაკი (წლები)	ექიმის შეფას. პრიორ. რიგი P_m	Alg 1 გამოთვლ. პრიორ. რიგი P_1	Alg 1 & Alg 2 შედგენილი პრიორ. რიგი P	სხვაობა $\lvert P_m - P_1 \rvert$ D_1	სხვაობა $\lvert P_m - P \rvert$ D
1	20	1-8	1	1	0	0
2	59	1-8	7-8	7-8	0	0
3	75	15-16	14	15	1	0
4	33	15-16	15-16	16	0	0
5	48	19-21	19-20	19-20	0	0
6	62	25	23	23	2	2
7	3	19-21	24	24	3	3
8	5	23-24	21	21	2	2
9	75	1-8	6	6	0	0
10	61	1-8	2-3	3	0	0
11	8	10-11	9-10	10-11	1	0
12	55	19-21	19-20	19-20	0	0
13	2.5	1-8	5	5	0	0
14	0.2	1-8	15-16	9	8	1
15	36	18	18	18	0	0
16	80	17	17	17	0	0
17	0.1	1-8	2-3	2	0	0
18	40	10-11	9-10	10-11	1	0
19	70	22	25	25	3	3
20	47	12-14	11-12	12-13	1	0
21	66	23-24	22	22	1	1
22	39	9	7-8	7-8	2	2
23	50	12-14	13	14	0	0
24	34	1-8	4	4	0	0
25	24	12-14	11-12	12-13	1	0

ჩატარებულმა კვლევამ აჩვენა, რომ შემოთავაზებული მოდელი შეიძლება გამოვიყენოთ გრიპის სეზონის დროს დისპეტჩერიზაციის პროცესის გასაუმჯობესებლად.

აღსანიშნავია, რომ ალგორითმები გამოცდილია მხოლოდ გრიპის სტანდარტულ გამოძახებათა 25 შემთხვევიან მოდელზე. დიდი ზომის რეალურ შემთხვევათა სიმრავლეზე გამოყენებისას შესაძლოა საჭირო გახდეს მოდელში და გამოყენებულ ალგორითმებში გარკვეული ცვლილებების შეტანა.

4.4. დისპეტჩერიზაციის ალტერნატიული მოდელი

სასწრაფო სამედიცინო დახმარების გამოძახების გადაუდებლობის შეფასება 10 ქულიანი სისტემით, თუნდაც გამოცდილი ექიმის მიერ, დაკავშირებულია ობიექტურად არსებულ სირთულეებთან სამედიცინო თვალსაზრისით. ხშირად, შეუძლებელია წინასწარ ზუსტად განსაზღვრა, თუ რომელი შემთხვევაა უფრო გადაუდებელი. ამის გამო ბევრ ქვეყანაში მოქმედი დისპეტჩერიზაციის სისტემები ეყრდნობიან არა შემთხვევათა პრიორიტეტების რიგს, არამედ შემთხვევათა გადაუდებლობის დონეს. როგორც წესი, გადაუდებლობის დონეთა რაოდენობა 3-დან 5-მდეა [58]. ამასთან, სამედიცინო თვალსაზრისითაც უფრო სანდოა გამოძახებათა შემთხვევები დაიყოს გადაუდებლობის სამი დონის მიხედვით, ვიდრე 10 ქულიანი სისტემით. ამის გათვალისწინებით, დისპეტჩერიზაციის აღწერილი მოდელი გამოიცადა გრიპის 35 ტიპიური გამოძახების შემთხვევებით შედგენილ კრებულზე, რომლებიც გამოცდილი ექიმის მიერ იყო კლასიფიცირებული გადაუდებლობის 3 დონის მიხედვით: მაღალი (**h**), საშუალო (**m**) და დაბალი (**l**) (ნახ.35).



ნახ. 35. დისპეტჩერიზაციის ალტერნატიული მოდელი

Alg1 და Alg2 ალგორითმების ერთობლივი გამოყენებით ჩატარებული კლასიფიკაციის შედეგი ასეთია: გამოძახების ყველა შემთხვევას სწორად განესაზღვრა გადაუდებლობის დონე. იხილეთ ცხრილი 16.

ცხრილში მოყვანილი კრებულიდან 20 შემთხვევა ექიმის მიერ შეფასებულია როგორც **h** მაღალი დონის გადაუდებლობისა. აქედან 18 შემთხვევის გამოსავლენად საკმარისი აღმჩნდა Alg1-ის გამოყენება, კრიტერიუმით: $\mathbf{r_1} \geq 85$.

ალგორითმების მიხედვით გამოთვლილი პრიორიტეტულობის რიგის შემდეგ 2 შემთხვევაში P = {19, 20} მნიშვნელობა $\mathbf{r_1}$ ინდექსისა კრიტიკულზე უფრო ნაკლები აღმოჩნდა. ორივე პაციენტი ბავშვია ($\mathbf{r_a} > 1$). შესაბამისად, დამატებით გამოყენებულ იქნა Alg2. დამატებითი კრიტერიუმის $\mathbf{r_2} \geq 100$ მიხედვით, ორივე შემთხვევა **h** მაღალი დონის გადაუდებლობის ჯგუფში აღმოჩნდა.

ცხრ. 15. გადაუდებლობის ინდექსის გამოთვლილი r_1 და r_2 სიდიდეები გადაუდებლობის h/m/l დონეები ექიმის შეფასების მიხედვით

პაციენტი	ასაკი (წლ.)	ტემპ. °C	მთავარი კლინიკ. სიმპტ.	თანმხლ. დაავად. განსაკ. მდგომარ.	ექიმის შეფას. გადაუდ დონე h/m/l	Alg 1 გამოთვ გადაუ-დებლ. ინდექს r_1	Alg 2 გამოთვ გადაუ-დებლ. ინდექს r_2	გამოთვ. პრიორ. რიგი P
1	20	39.2	სუნთქ.გაძნ.	ასთმა	h	140	-	1
2	59	38.7	ტკივ. გულ.	გულის დაავ	h	95	-	12-14
3	75	38.5	-	გულის დაავ	m	67	85	26
4	33	38.8	დიარეა	იმუნოდეფ.	m	65	-	27
5	48	38.7	ტკივ. გულ.	დიაბეტი	h	90	-	15-16
6	62	39.2	-	-	l	50	-	34
7	3	38.3	ღებინება	-	m	48	70	30
8	5	38.5	დიარეა	-	l	53	63	32
9	75	38.7	სუნთქ.გაძნ.	გულის დაავ	h	107	153	9
10	61	39.2	სუნთქ.გაძნ.	გულის დაავ	h	120	-	3
11	8	39.1	-	ნევრ. მდგ.	h	85	-	18
12	55	39.2	-	დიაბეტი	m	75	-	24
13	2.5	38.4	სუნთქ.გაძნ.	ნევრ. მდგ.	h	113	171	6
14	0.2	37.9	დიარეა	-	h	65	100	20
15	36	37.8	-	ორს.	l	60	-	31
16	80	34.5	დიარეა	-	m	62	76.5	29
17	0.1	38.9	სუნთქ.გაძნ.	გულის დაავ	h	120	270	2
18	82	39.0	სუნთქ.გაძნ.	-	h	97	136	11
19	70	38.1	ღებინება	-	l	47	51	35
20	47	39.4	-	გულის დაავ	m	80	-	21-23
21	66	37.7	-	დიაბეტი	l	52	59.5	33
22	39	38.4	სუნთქ.გაძნ.	დიაბეტი	h	95	-	12-14
23	50	38.3	ღებინება	გულის დაავ	m	70	-	25
24	34	38.5	სუნთქ.გაძნ.	ფილტვ.ქრ.	h	115	-	4-5
25	24	38.6	ღებინება	ორს.	m	80	-	21-23
26	60	37.8	სუნთქ.გაძნ.	ასთმა	h	110	-	7
27	5	39.5	-	ასთმა	h	108	162	8
28	58	38.3	ტკივ. გულ.	ასთმა	h	115	-	4-5
29	38	39.1	დიარეა	ორს.	h	105	-	10
30	2	37.9	-	ასთმა	h	85	150	17
31	43	39.3	-	ფილტვ.ქრ.	h	95	-	12-14
32	30	38.7	სუნთქ.გაძნ.	იმუნოდეფ.	h	90	-	15-16
33	48	38.0	ტკივ. გულ.	ღვიძლის დ.	m	80	-	21-23
34	4	38.4	ღებინება	თირკმლ. დ.	m	63	81	28
35	1	39.5	-	ტკ. მუცლ.	h	75	125	19

ამგვარად, **h** მაღალი დონის გადაუდებლობის შემთხვევების გამოვლენის ავტომატური პროცესის კრიტერიუმებია შემდეგი უტოლობათა სისტემა:

თუ $r_a = 1$, $r_1 \geq 85$;

თუ $r_a > 1$, $r_1 \geq 85$ და/ან $r_2 \geq 100$.

ორივე ალგორითმის გამოყენებით, საცდელი კრებულის ოცივე მაღალი გადაუდებლობის დონის მქონე შემთხვევა სწორად იქნა გამოვლენილი.

კრებულის 35 შემთხვევიდან შემდეგი 10 ექიმის მიერ შეფასებულია როგორც **m** საშუალო დონის გადაუდებლობისა. კომპიუტერული კლასიფიკაციით ეს შემთხვევები შემდეგი კრიტერიუმებით გამოვლინდა:

თუ $r_a = 1$, $60 < r_1 < 85$ (შემდეგი 9 შემთხვევა პრიორიტეტულობის რიგში: P = {21,…,29});

თუ $r_a > 1$, $60 < r_1 < 85$ და/ან $70 \leq r_2 < 100$ (ერთი შემთხვევა: P = 30).

პრიორიტეტულობის რიგის ბოლო 5 შემთხვევა P = {31,…,35}, რომლებიც ექიმის მიერ განსაზღვრულია, როგორც გადაუდებლობის **l** დაბალი დონისა, კომპიუტერულად გამოვლინდა კრიტერიუმებით:

თუ $r_a = 1$, $r_1 \leq 60$;

თუ $r_a > 1$, $r_1 \leq 60$, $r_2 < 70$.

თუმცა Alg1 და Alg2 ალგორითმების ერთობლივი გამოყენებით ჩატარებული კომპიუტერული კლასიფიკაცია სრულად ემთხვევა გამოცდილი ექიმის მიერ შემთხვევათა შეფასებას, მიზნად დავისახე უფრო მარტივი კლასიფიკაციის სისტემის პოვნა.

პაციენტის ასაკი ხშირად გადამწყვეტი ფაქტორია შესაძლო გართულებების სწრაფად განვითარებისა, რაც განსაზღვრავს გადაუდებლობის დონეს. ამიტომ, გრიპის შემთხვევათა საცდელ კრებულზე (ცხრილი 16) გამოცდილი იქნა ასაკის ფაქტორის უფრო მკვეთრად გამთვალისწინებელი ალგორითმი Alg3:

Alg 3: $$r_1 = 2r_a + r_c + r_s + r_d \qquad (3)$$

Alg3 ალგორითმის გამოყენებით შესაძლებელი აღმოჩნდა გრიპის შემთხვევათა გადაუდებლობის დონის განსაზღვრა მაღალი სიზუსტით. კლასიფიკაციისთვის გამოყენებულ იქნა შემდეგი კრიტერიუმები:

- თუ $\mathbf{r_3} > 90$, შემთხვევის გადაუდებლობის დონე განისაზღვრება როგორც **h** მაღალი. ამ კრიტერიუმით 20 შემთხვევიდან 19 სწორად გამოვლინდა. მხოლოდ 1 შემთხვევა (P = 20) Alg3-ის მიხედვით მიეკუთვნა **m** საშუალო დონეს;
- თუ $75 \leq \mathbf{r_3} \leq 90$, შემთხვევის გადაუდებლობის დონე განისაზღვრება როგორც **m** საშუალო. ამ კრიტერიუმით სწორად გამოვლინდა გადაუდებლობის **m** დონის 9 შემთხვევა. მხოლოდ ერთი **m** საშუალო დონის გადაუდებლობის შემთხვევა (P = 30) გამოვლინდა როგორც l დაბალი დონისა (Alg3-ის მიხედვით);
- თუ $\mathbf{r_3} < 75$, გამოძახების შემთხვევის გადაუდებლობის დონე განისაზღვრება, როგორც l დაბალი. ხუთივე l დაბალი დონის შემთხვევა სწორად იქნა კლასიფიცირებული.

ამგვარად, შემოთავაზებულ ალტერნატიულ გამარტივებულ მოდელსაც კლასიფიკაციის მაღალი სიზუსტე ახასიათებს.

4.5. გამოძახებათა პრიორიტეტიზაციის ზოგადი პრინციპები

შემთხვევათა პრიორიტეტულობის განსაზღვრის პროცესის სრული ავტომატიზაციისათვის, აუცილებელია განხორციელდეს შემდეგი საფეხურები:

- მიენიჭოს გადაუდებლობის წონები თითოეულ ინციდენტის ტიპს: გრიპი ან სხვა მწვავე ვირუსული ინფექცია, გულის დაავადებები (ტკივილი გულის არეში), სუნთქვის პრობლემები, ტრავმა ავარიის ან დაჭრის შედეგად, სისხლდენა, ორსულთა პრობლემები და სხვა [59,60].
- თითოეული ინციდენტის ტიპისათვის უნდა შეირჩეს სისტემა პარამეტრებისა, რომლებიც მომავალში დაფიქსირებული იქნება გამოძახების მიმღები ოპერატორების მიერ. მაგალითად, გრიპის შემთხვევებისათვის

ვიყენებთ პარამეტრებს: ასაკი, ტემპერატურა, კლინიკური სიმპტომები (სუნთქვის გაგრძელება, ღებინება, დიარეა და ა.შ.), თანმხლები ქრონიკული დაავადებები (მაგალითად, დიაბეტი, ჰიპერტონია,...) და განსაკუთრებული მდგომარეობები (მაგალითად, ორსულობა) [61,62].

- არჩეულ პარამეტრებს უნდა მიენიჭოს გარკვეული წონები მათი ფარდობითი კლინიკური მნიშვნელოვნების მიხედვით.
- კვლევის პირველ ეტაპზე ზემოთ ნახსენები სამი საფეხური უნდა განხორციელდეს მედიცინის სხვადასხვა სფეროში მომუშავე გამოცდილ ექიმებთან კონსულტაციების მიღების შემდეგ.
- უნდა შეიქმნას სხვადასხვა (**i**) ალგორითმი გადაუდებლობის ინდექსის r_i გამოსათვლელად თითოეული შემთხვევისათვის ზემოთ აღნიშნული გზით წინასწარ განსაზღვრული სამედიცინო ინფორმაციის საფუძველზე.
- განისაზღვროს შემთხვევების პრიორიტეტულობის რიგი r_i გადაუდებლობის ინდექსის მიხედვით.
- შედარდეს მომსახურების გადაუდებლობის ინდექსის r_i **i**-ური ალგორითმით გამოთვლილი მნიშვნელობები ექიმის შეფასებით მიღებულ გადაუდებლობის D_m ქულებთან.
- გამოვლინდეს ოპტიმალური ალგორითმი ზემოთ აღნიშნული შედარების შედეგების მიხედვით გამოძახებების დიდი რაოდენობის სიმრავლის ანალიზისას. იმისათვის, რომ სისტემა გახდეს თვით დასწავლადი, გადაუდებლობის ინდექსის გამოთვლილი მნიშვნელობები r_i უნდა შედარდეს შესაფერის D_m მნიშვნელობებს. (მიჩნეულია, რომ ექიმის შეფასება სწორია.). ოპტიმალურად მიიჩნევა ის i-ური ალგორითმი, რომლითაც მიიღება D_m ქულებთან მაქსიმალურად კორელირებული გადაუდებლობის ინდექსის r_i მნიშვნელობები.

r_i სიდიდის გამოთვლის ზოგადი ფორმულა შემდეგი სახით შეგვიძლია გამოვხატოთ:

$$r_i = \{ k_i\ a_i \}, \qquad i=1\ldots n \tag{1}$$

სადაც k_i სხვადასხვა ინციდენტის ტიპის წონებია (მაგალითად გრიპი ან სხვა მწვავე ვირუსული ინფექცია, გულის პრობლემა, სუნთქვის პრობლემა, ტრავმა და სხვა.), a_i სიდიდე კი გამოითვლება ფორმულით:

$$a_i = \sum_{j=1}^{m} b_j c_j \tag{2}$$

სადაც bj-ით აღნიშნულია შესაბამისი cj პარამეტრების წონები.

მაგალითად, გრიპის შემთხვევაში cj პარამეტრებია: ასაკი, ტემპერატურა, კლინიკური სიმპტომები, თანმხლები დაავადებები და განსაკუთრებული მდგომარეობები.

პარამეტრები k,b და c პროცენტებშია გამოსახული:

$$\sum_{i=1}^{n} k_i = \sum_{i=1}^{m} b_i = \sum_{j=1}^{m} c_j = 100 \tag{3}$$

კვლევის შედეგად შევიმუშავებთ მანქანურ დასწავლაზე მორგებულ ფუნქციას. საწყის ეტაპზე ფუნქცია იმუშავებს თავდაპირველად მინიჭებულ მიახლოებით წონებზე. ასაკს, ტემპერატურას, თანმხლებ დაავადებებს და მსგავს პარამეტრებს მინიჭებული ექნება ფიქსირებული წონები, რადგან მათ უშუალო მნიშვნელობას (ასაკი, ტემპერატურა...) გამოთვლებში პირდაპირ ვერ გამოვიყენებთ. კვლევის შედეგად მივიღებთ ფუნქციას, რომელიც შემავალი პარამეტრების კატეგორიების, შესაბამისი წონებისა და თავად ინციდენტის ტიპის წონის მიხედვით გამოითვლის კონკრეტული საქმის კრიტიკულობის (გადაუდებლობის) ინდექსს. ფუნქციაში ჩასასმელი რიცხვითი მნიშვნელობებიდან პარამეტრის კატეგორია იქნება პაციენტისგან მიღებული ინფორმაციის შესაბამისი მნიშვნელობა, ინციდენტის ტიპის შიგა და გარე წონები კი ალგორითმის მიერ ყოველდღიურად შერჩეული მნიშვნელობები იქნება.

რეალური კოეფიციენტების განსაზღვრა ალგორითმმა უნდა ისწავლოს ადამიანის დასკვნებისგან. ამისთვის შეგვიძლია შევაგროვოთ სასწრაფოს ექიმების სუბიექტური აზრი ყოველი საქმის დასრულებისას. ალგორითმი პერიოდულად შეუცვლის პარამეტრებსა და ინციდენტებს წონებს და ხელახლა გამოითვლის ძველი საქმეების სავარაუდო პრიორიტეტებს.

ცხადია, პარამეტრების გადათამაშება მოხდება ეტაპობრივად და თითოეული ცვლილების შემდეგ საჭირო იქნება მთელი ისტორიის თავიდან გამოთვლა. ჩატარებული გამოთვლების შედეგად ავარჩევთ წონების იმ ერთ მიმდევრობას, რომლის შემთხვევაშიც ალგორითმის განსაზღვრული პრიორიტეტები ყველაზე მეტად დაემსგავსება ადამიანის განსაზღვრულ პრიორიტეტებს. ალგორითმი ისწავლის დაგროვებული ისტორიის საფუძველზე და ყოველი შემდეგი დასკვნა უფრო და უფრო დაუახლოვდება მსგავს საქმეებზე სასწრაფოს ექიმის დასკვნას.

ამ მეთოდის გამოყენებით მძლავრი გამოთვლითი რესურსი გვჭირდება მხოლოდ 24 საათში ერთხელ, ინციდენტის ტიპის შიდა და გარე წონების კოეფიციენტების გამოსაყვანად. ყოველი ახალი საქმის პრიორიტეტი კი წინასწარ შემუშავებული პარამეტრებით მილიწამებში გამოითვლება.

ამ ამოცანის მოსალოდნელი სირთულეები იქნება:

ფიქსირებული პრიორიტეტების განსაზღვრა - თითოეული ინციდენტის ტიპისთვის სპეციფიური კითხვები და დეტალები ექიმებისგან უნდა გავარკვიოთ.

არასრული მონაცემები - რადგან საქმე ადამიანის სიცოცხლეს ეხება, დაუზუსტებელი ინფორმაციის ნაცვლად სავარაუდოდ ყველაზე კრიტიკულ კატეგორიებს ავიღებთ.

არასტრუქტურირებული მონაცემები - მონაცემების შენახვა-დამუშავების დროს ძირითად სირთულეს წარმოადგენს არასტრუქტურირებული მონაცემები [63] (მაგალითად, კომენტარების სახით დამატებული მნიშვნელოვანი ინფორმაცია). ამ საკითხშიც დაგვჭირდება სპეციფიური, ინოვაციური მიდგომის მოფიქრება. შსს სსიპ „112“ გამოთქვამს მზადყოფნას, რომ მხედველობაში მიიღოს და განახორციელოს კვლევის შედეგად მიღებული რეკომენდაციები.

ამ ეტაპზე ვერიდები მანქანური დასწავლის ტექნოლოგიებისა და ამოცანების საბოლოოდ დაკონკრეტებას. რადგან მანქანური დასწავლის

სიღრმისეულად გარჩევის შემდეგ, შესაძლოა, შემეცვალოს წარმოდგენა დასმული ამოცანის გადაწყვეტის კონკრეტულ მეთოდოლოგიებზე.

4.6. მეოთხე თავის დასკვნა

სასწრაფო სამედიცინო მომსახურების დისპეტჩერების დასახმარებლად შეიქმნა გრიპის სეზონის შემთხვევებზე მორგებული მოდელი და ალგორითმები თითოეული შემთხვევის სასწრაფო სამედიცინო მომსახურების გადაუდებლობის დონის დასადგენად.

გამოყენებული ალგორითმებით შესრულებული გამოთვლების საფუძველზე დადგინდა კრიტერიუმები გრიპის თითოეული შემთხვევის გადაუდებლობის დონისა და მომსახურების რიგის გამოსავლენად.

შექმნილი სისტემის გამოყენებით შესაძლებელია ობიექტურად იქნას დადგენილი გრიპის შემთხვევათა მომსახურების რიგი და თითოეული შემთხვევის გადაუდებლობის დონე. ამასთან, აღნიშნული ალგორითმების საშუალებით ვლინდება ყველა შემთხვევა, რომლებსაც გამოცდილი ექიმი გადაუდებლობის მაღალ დონეს ანიჭებს.

დისპეტჩერს წარედგინება კომპიუტერულად განსაზღვრული გადაუდებლობის დონის მიხედვით აგებული შემთხვევათა პრიორიტეტული რიგი, რის საფუძველზეც მას შეუძლია სწრაფად მიიღოს ადეკვატური გადაწყვეტილება სასწრაფო დახმარების ბრიგადების გაგზავნის თაობაზე. ეს ქმნის სასწრაფო სამედიცინო დახმარების ბრიგადების დისპეტჩერიზაციის გაუმჯობესების რეალურ შესაძლებლობას.

დასკვნა

სადისერტაციო ნაშრომის ფარგლებში ჩატარებული კვლევის შედეგების საფუძველზე შესაძლებელია შემდეგი დასკვნების გაკეთება:

- მონაცემთა რელაციური და არარელაციური ბაზების მახასიათებლების შედარების საფუძველზე განსაზღვრულია იმ ამოცანათა კლასი, რომლებისთვისაც ეფექტიანია მონაცემთა არარელაციური ბაზების გამოყენება.
- იმის გათვალისწინებით, რომ გარკვეული ზღვრის შემდეგ რესურსის ვერტიკალური სკალირება (vertical scale) საკმაოდ ძვირი ჯდება, თანაც ნაკლებად ეფექტურია, უპირატესობა მიენიჭა ჰორიზონტალურ სკალირებას (horizontal scale). მონაცემთა ბაზების მწარმოებლურობის ძირითადი მახასიათებლების, მონაცემთა მთლიანობის, ტრანზაქციათა იზოლირების დონეების, ACID პრინციპების და CAP თეორემის გათვალისწინებით გადაუდებელი დახმარების ოპერატიული მართვის ცენტრ „112“-ის მონაცემთა ბაზად შეირჩა მონაცემთა დოკუმენტ–ორიენტირებული, არარელაციური ბაზა MongoDB.
- შემუშავებულია კრიტიკული სისტემის ინფრასტრუქტურა, აღწერილია სისტემის გამართვის ეტაპები და დაპროექტებულია მაღალი წვდომადობის მქონე მონაცემთა საცავი.
- მონაცემთა ბაზის სამივე ძირითადი მახასიათებლის (პროცესორი, ოპერატიული მეხსიერება და ხისტი დისკი), ისევე როგორც საიმედოობისა და წვდომადობის გასაუმჯობესებლად გადაწყდა Sharding–ისა და რეპლიკაციის ტექნოლოგიების გამოყენება.
- გაანალიზდა გადაუდებელი სამედიცინო სამსახურის გამოწვევები და განისაზღვრა საკვლევი მიმართულებები კომპიუტერული მეთოდების გამოყენებისათვის. პროგრამირების ენა Python-ის საშუალებით შესრულდა extract, transform, load (ETL) და მონაცემთა წინასწარი დამუშავების (data preprocessing) პროცესები, რის შედეგადაც მიღებულ იქნა მანქანური

დასწავლისთვის საჭირო სტრუქტურის მქონე მონაცემთა კრებული ბოლო 5 წლის გადაუდებელი სამედიცინო დახმარების შემთხვევების ბაზაზე.

- Python პროგრამირებისა და Weka ხელსაწყოს გამოყენებით გაანალიზებულ იქნა მიღებული მონაცემთა ნაკრები. გამოვლინდა, წაიშალა და შესწორდა ანომალიური შემთხვევები, რაც გაუთვალისწინებლობის დროს უარყოფითად აისახებოდა მანქანური დასწავლის შედეგებზე.
- სასწრაფო სამედიცინო მომსახურების დისპეტჩერიზაციის გასაუმჯობესებლად შეიქმნა გრიპის სეზონის შემთხვევებზე მორგებული კომპიუტერული მოდელი და გამოთვლების საფუძველზე დადგინდა კრიტერიუმები გრიპის თითოეული შემთხვევის გადაუდებლობის დონისა და მომსახურების რიგის გამოსავლენად.
- შექმნილი სისტემის გამოყენებით ობიექტურად დგინდება გრიპის შემთხვევათა მომსახურების რიგი თითოეული მათგანის გადაუდებლობის დონის მიხედვით. ამასთან, აღნიშნული ალგორითმების საშუალებით გამოვლინდა ყველა შემთხვევა, რომლებიც გამოცდილმა ექიმმა გადაუდებლობის მაღალი დონით შეაფასა. შესაძლებელია დისპეტჩერს წარედგინოს კომპიუტერულად განსაზღვრული გადაუდებლობის დონის მიხედვით აგებული შემთხვევათა პრიორიტეტული რიგი, რის საფუძველზეც მას შეუძლია სწრაფად მიიღოს ადეკვატური გადაწყვეტილება სასწრაფო დახმარების ბრიგადების გაგზავნის თაობაზე. ეს ქმნის სასწრაფო სამედიცინო დახმარების ბრიგადების დისპეტჩერიზაციის გაუმჯობესების რეალურ შესაძლებლობას გრიპის სეზონის დროს.

გამოყენებული ლიტერატურა

1. https://www.britannica.com/technology/database, გადამოწმ. - 21.05.2018.
2. ჩოგოვაძე გ., სურგულაძე გ., ქაჩიბაია ვ. მონაცემთა ბაზების მართვის სისტემები. სტუ. თბილისი. 1988
3. 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, 65-98
4. Wedekind H., Surguladze G. Technology of Designing of Distributed Systems on the Basis of Objectoriented Programming. ISSN 021-7164, GTU, Tbilisi. 1996. pp.96-100.
5. მეიერ-ვეგენერი კ., სურგულაძე გ., ბასილაძე გ. საინფორმაციო სისტემების აგება მულტიმედიური მონაცემთა ბაზებით. სტუ. თბილისი. 2014
6. http://nosql-database.org, გადამოწმ. - 21.05.2018.
7. http://vakhokor.blogspot.com/2014/10/nosql-n-cap.html, გადამოწმ. - 21.05.2018.
8. NoSQL For Dummies®. Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com Copyright © 2015, New Jersey
9. https://www.mongodb.com/compare/mongodb-mysql, გადამოწმ. - 21.05.2018.
10. სურგულაძე გ., **კივილაძე გ.**, კახელი ბ. NoSQL მონაცემთა ბაზების განვითარების პერსპექტივები და პრობლემები მართვის საინფორმაციო სისტემებში. *სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“.* 2016, 2, 22, 230 - 239.
11. https://en.wikipedia.org/wiki/NoSQL, გადამოწმ. - 21.05.2018.
12. Гранков М.В., Жуков А.И. Системы управления Базами данных. Донской гос.техн. Университет. Ростов-на-Дону. 2013
13. https://en.wikipedia.org/wiki/Document-oriented_database, გადამოწმ. - 21.05.2018.
14. https://www.predictiveanalyticstoday.com/top-nosql-document-databases/, გადამოწმ. - 21.05.2018.
15. https://en.wikipedia.org/wiki/Graph_database, გადამოწმ. - 21.05.2018.
16. www.arangodb.com, გადამოწმ. - 21.05.2018.
17. https://www.ingeniux.com/ company/blog/why-nosql-is-the-future-of-web-content-management, გადამოწმ. - 21.05.2018.
18. http://www.drdobbs.com/database/nosql-with-mysql/240167115, გადამოწმ. - 21.05.2018.
19. https://docs.datastax.com/en/latest-dse/datastax_enterprise/graph-/dseGraphAbout.html, გადამოწმ. - 21.05.2018.
20. MarkLogic. https://en.wikipedia.org/wiki/MarkLogic, გადამოწმ. - 21.05.2018.
21. Neo4j. https://en.wikipedia.org/wiki/Neo4j, გადამოწმ. - 21.05.2018.

22. OrientDB. https://en.wikipedia.org/wiki/OrientDB, გადამოწმ. - 21.05.2018.

23. https://en.wikipedia.org/wiki/Stardog, გადამოწმ. - 21.05.2018.

24. https://en.wikipedia.org/wiki/Web_Ontology_Language, გადამოწმ. - 21.05.2018.

25. https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html, გადამოწმ. - 21.05.2018.

26. Samuel, A. L. Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort, Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*. 1959, 3, 3, p. 210.

27. http://conference.sens-2016.tsu.ge/uploads/577bcbb48187eსამაგისტრო.pdf, გადამოწმ. - 21.05.2018.

28. https://www.meaningbot.com/blog/2017/5/3/the-importance-of-machine-learning-theory-and-how-math-and-physics-majors-can-thrive-in-machine-learning, გადამოწმ. - 21.05.2018.

29. Bandhyopadhyay S. K., Tuhin U. P. Automatic segmentation of brain tumour from multiple images of brain MRI. *Int J Appl Innovat Eng Manage (IJAIEM).* 2013, 2, 1: 240-8.

30. Awate S.P., Tasdizen T., Foster N., Whitaker, R. T. Adaptive Markov modeling for mutual-information-based, unsupervised MRI brain-tissue classification. *Medical Image Analysis*, 2006, 10, 5, 726-739.

31. სურგულაძე გ., **კივილაძე გ.** ტრანზაქციის იზოლირების დონეები რელაციურ და არარელაციურ მონაცემთა ბაზებში. "კომპიუტინგი/ინფორმატიკა, განათლების მეცნიერებები, მასწავლებლის განათლება" IV საერთაშორისო-სამეცნიერო კონფერენციის მოხსენებათა თეზისები. 2016, 161 - 168.

32. Nayak A., Poriya A., Poojary D. Type of NOSQL databases and its comparison with relational databases. *International Journal of Applied Information Systems*. 2013, 5, 4, 16-19.

33. სურგულაძე გ., **კივილაძე გ.** შესავალი NoSQL მონაცემთა ბაზებში (MongoDB). თბილისი: სტუ. „IT-კონსალტინგის ცენტრი", თბილისი, 2017, 151 გვ.

34. სურგულაძე გ. **კივილაძე გ.** კონსისტენტური მოდელები რელაციურ და არარელაციურ მონაცემთა ბაზებში. *VIII საერთაშორისო სამეცნიერო და პრაქტიკული კონფერენციის "ინტერნეტი და საზოგადოება" (INSO2017) ნაშრომების კრებული*. 2017, 151 - 155.

35. Grigorik I. Weak Consistency and CAP Implications. Igvita, 24 June, 2010). Google-Co. https://www.igvita.com/2010/06/24/weak-consistency-and-cap-implications/, გადამოწმ. - 21.05.2018.

36. Nemeth E., Snyder G., Hein T.R., Whaley B. UNIX and LINUX System Administration Handbook fourth edition. Prentice Hall. 2010

37. Cobbaut P. Linux Fundamentals. 2015. http://linux-training.be-/linuxfun.pdf, გადამოწმ. - 21.05.2018.

38. https://github.com/mistertandon/node-express-hbs, გადამოწმ. - 21.05.2018.

39. სურგულაძე გ., კივილაძე ნ., **კივილაძე გ.** კორპორაციული აპლიკაციების აგება დაპროგრამების სერვის-ორიენტირებული ტექნოლოგიით. *სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“.* 2016, 1, 21, 215 - 220.

40. სურგულაძე გ., კახელი ბ., მაისურაძე გ., მონაცემთა შენახვა-დამუშვების განაწილებული სისტემები. *სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“.* 2017, 1, 23, 79 – 85.

41. სურგულაძე გ., პეტრიაშვილი ლ. (2015). მონაცემთა საცავის აგების ტექნოლოგია ინტერნეტული ბიზნესის სისტემებისათვის. ISBN 99940-40-36-7. სტუ. „ტექნიკური უნივერსიტეტი“, თბ., 200 გვ.

42. http://police.ge/ge/lepl/lepl112, გადამოწმ. - 21.05.2018.

43. ჩოგოვაძე გ., ფრანგიშვილი ა., **კივილაძე გ.**, სურგულაძე გ., ნარეშელაშვილი გ. ინფორმაციული საზოგადოება, მონაცემთა მენეჯმენტის ახალი ტექნოლოგიები და ექსტრემალური სიტუაციების მართვის სისტემები. *სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“.* 2018, 1, 25, 7 - 16.

44. Chicco D. Ten quick tips for machine learning in computational biology. *BioData Mining*. 2017, 10, 35, 1–17. doi:10.1186/s13040-017-0155-3.

45. Luo G. MLBCD: a machine learning tool for big clinical data. *Health Information Science and Systems*. 2015, 3, 3. doi:10.1186/s13755-015-0011-0.

46. https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/, გადამოწმ. - 21.05.2018.

47. https://www.cs.waikato.ac.nz/~ml/weka/index.html, გადამოწმ. - 21.05.2018.

48. Manning T., Sleator R.D., Walsh P. Biologically inspired intelligent decision making: A commentary on the use of artificial neural networks in bioinformatics. *Bioengineered.*, 2014, 5, 2, 80-95. doi:10.4161/bioe.26997.

49. Golding S.E., Horsfield C., Davies A., et al. M. Exploring the psychological health of emergency dispatch centre operatives: a systematic review and narrative synthesis. *PeerJ*, 2017,*5*, e3735.

50. Taubenberger J.K, Morens D.M. The Pathology of Influenza Virus Infections. *Annual review of pathology*. 2008, 3, 499-522. doi:10.1146/annurev. pathmechdis.3.121806.154316.

51. Klepser M.E. Socioeconomic Impact of Seasonal (Epidemic) Influenza and the Role of Over-the-Counter Medicines. *Drugs*. 2014, 74, 13, 1467-1479. doi:10.1007/s 40265-014-0245-1.

52. Jules A., Grijalva C. G., Zhu Y., et al. Influenza-Related Hospitalization and ED Visits in Children Less Than 5 Years: 2000–2011. *Pediatrics*. 2015, 135, 1, e66–e74. http://doi.org/10.1542/peds.2014-1168

53. Lipsitch M., Barclay W., Raman R, et al. Viral factors in influenza pandemic risk assessment. Guan Y, ed. *eLife*. 2016, 5, e18491. doi:10.7554/eLife.18491.

54. Lafond KE, Nair H, Rasooly MH, et al. Global Role and Burden of Influenza in Pediatric Respiratory Hospitalizations, 1982–2012: A Systematic Analysis. *PLoS Medicine*. 2016, 13, 3, e1001977. http://doi.org/10.1371/journal.pmed.1001977

55. Garg S, Jain S, Dawood FS, et al. Pneumonia among adults hospitalized with laboratory-confirmed seasonal influenza virus infection—United States, 2005–2008. *BMC Infectious Diseases*. 2015,15, 369. http://doi.org/10.1186/s12879-015-1004-y

56. Sellers SA, Hagan RS, Hayden FG, et al. The hidden burden of influenza: A review of the extra-pulmonary complications of influenza infection. *Influenza and Other Respiratory Viruses*. 2017, 11, 5, 372-393. doi:10.1111/irv.12470.

57. **Kiviladze G.** Semi automated management of defining the case priority in the flu epidemic season for Emergency Medical Service. *GESJ:Computer Sciences and Telecommunications*. 2018, 1, 53, 101 - 105.

58. Andersen MS, Johnsen SP, Sørensen JN, et al. Implementing a nationwide criteria-based emergency medical dispatch system: A register-based follow-up study. *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine*. 2013,21,53. doi:10.1186/1757-7241-21-53.

59. https://www.apcointl.org/doc/911-resources/apco-standards/386-public-safety-communications-common-incident-types-for-data-exchange/file.html, გადამოწმ. - 21.05.2018.

60. https://www.nfpa.org/-/media/Files/News-and-Research/Fire-statistics/Fire-service/osNFIRSIncidentType.ashx?la=en&hash=4B96DA2CC1953946CF0D6655A8FE49E1819A34F8, გადამოწმ. - 21.05.2018.

61. Balfagon G., Blanco-Rivero J., Marquez-Rodas J. Influenza Season and ARDS after Cardiac Surgery. *The New English Journal of Medicine*. 2018, 378, 772-773. DOI: 10.1056/NEJMc1712727

62. Rasmussen S.A. and Jamieson D.J. 2009 H1N1 Influenza and Pregnancy — 5 Years Later. *The New English Journal of Medicine*. 2014, 371, 1373 – 1375. DOI: 10.1056/NEJMp1403496

63. Scheurwegs E., Luyckx K., Luyten L., et al. Data integration of structured and unstructured sources for assigning clinical codes to patient stays. *Journal of the American Medical Informatics Association*. 2016, 23, e1, e11 – e19, https://doi.org/10.1093/jamia/ocv115