ივ. ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

ეკატერინე ხვედელიძე

Develop a data processing scenario in structured databases

მონაცემთა დამუშავების სცენარის შემუშავება სტრუქტურირებულ ბაზებში

სამაგისტრო პროგრამა: ინფორმაციული ტექნოლოგიები

ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი:

მაია არჩუაძე

თბილისი

2019

ანოტაცია

ნაშრომში წარმოდგენილია მონაცემთა დამუშავების თანამედროვე მეთოდები სტრუქტურირებულ მონაცემთა ბაზებში. მონაცემთა დამუშავება მართვის ავტომატიზებულ სისტემებში ინფორმაციის შენახვისა და გადამუშავების ყველაზე ეფექტური და ფართოდ გამოყენებადი მეთოდია.

ამასთანავე , აღწერილია სტრუქტურირებული მონაცემთა ბაზის ადმინისტრირების მიხედვით მონაცემებზე მანიპულაცია. ბოლო ათწლეულში მონაცემთა ბაზების მართვის სისტემები ვითარდებოდა რამდენიმე მიმართულებით. მთავარი იყო რელაციური ბაზების, როგორც უმრავლესი მართვის საინფორმაციო სისტემების ძირითადი კომპონენტის სრულყოფა მწარმოებლურობის თვალსაზრისით.

თანამედროვე ტექნოლოგიების სამყაროში ინფორმაციის უდიდეს ნაწილს ადამიანების ნაცვლად კომპიუტერული ტექნიკა ქმნის. ინფორმაციის ნაკადი იმდენად სწრაფად იზრდება, რომ საჭირო ხდება ინფორმაციის დამუშავების თანამედროვე მეთოდების მოძიება. ნაშრომის მიზანია ინფორმაციის შენახვა-დამუშვების ტექნოლოგიების გამოკვლევა .

ასევე, განხილულია ბიზნეს-ანალიტიკა Power BI ტექნოლოგიების საფუძველზე, რომელიც მონაცემების ანალიზისა და ვიზუალიზაციის საშუალებას იძლევა. ასევე, Power BI Mobile აპლიკაცია მუშაობს ყველა მობილურ მოწყობილობაზე, რომელიც თავსებადია Windows10, iOS ოპერაციულ სისტემებთან.

Annotation

The project describes the methods of data processing in structured databases. Data processing is the most effective and widely used method of retaining and processing information in automated control systems.

In addition, the manipulation of the data according to the structured database administration. In the last decade, database management systems have been developing in several directions. The main was the improvement of the main components of the relational bases as the majority management information systems in terms of productivity.

Computer technology is the largest part of the world of modern technologies. The flow of information is so fast that it is necessary to find modern methods of processing information. The purpose of the work is to research the information storage technologies.

Also, Business Analytics is based on Power BI technologies which allows analyzing and visualization of data. Also, the Power BI Mobile application works on all mobile devices that are compatible with Windows10 and iOS operating systems.

შინაარსი

შესავალი

ნაშრომში წარმოდგენილია მონაცემთა ბაზების რელაციური მოდელის საფუძველზე მოქმედი სისტემა MS SQL Server-ის ადმინისტრირების საკითხები . აღწერილია მონაცემთა დამუშავების მეთოდები სტრუქტურირებულ ბაზებში. პირველ თავში განხილულია მონაცემთა სამი ტიპი : სტრუქტურირებული, არასტრუქტურირებული და ნაწილობრივ-სტრუქტურირებული მონაცემები. ასევე, მათი უპირატესობები. ნაშრომში სტრუქტურირებულ მონაცემებზე დაყრდნობით არის განხილული თითოეული მეთოდი , რადგან ძებნის შედეგი ყოველთვის ზუსტია.

მეორე თავში წარმოდგენილია მონაცემთა ბაზების ინფრასტრუქტურა. SQL Server - ის შემადგენელი კომპონენტები და ადმინისტრირების როლი მონაცემთა დამუშავებაში. მესამე თავში განხილულია მონაცემთა სარეზერვო ასლის ტიპები და მიდგომები. თითქმის ყველა თანამედროვე მონაცემთა ბაზის მართვის სისტემაში არსებობს მონაცემთა ბაზების სარეზერვო ასლების შექმნისა და მათი აღდგენის ინტეგრირებული იარაღები, სარეზერვო ასლების შექმნა მონაცემთა სარეზერვო ფიზიკურ მატარებლებზე მაშინაც კი არის შესაძლებელი, როდესაც მართვის სისტემა ერთდროულად მრავალ მომხმარებელს ემსახურება. მეოთხე თავში განხილულია მონაცემთა ბაზების შემადგენელი ფაილები და ფაილების ტიპები. ასევე, RAID ტექნოლოგია, რომელიც მონაცემთა შენახვისა და სანდოობის გაზრდის მიზნით გამოიყენება.

ასევე, განხილულია სერვერის უსაფრთხოების სისტემა , მონაცემთა ბაზის როლები და სერვერული როლები . ინდექსები - როგორც მონაცემებზე წვდომის ერთ-ერთი მეთოდი. წარმოდგენილია Microsoft Power BI ტექნოლოგიის გამოყენება სამედიცინო კორპორაციის ბიზნეს-მონაცემების ანალიზისათვის. ნაშრომში ასევე , აღწერილია სისტემური მონაცემთა ბაზები და მათი როლი მონაცემთა ბაზის ობიექტების ფუნქციონირებაში.

1. მონაცემთა ტიპები

მონაცემთა ბაზები შედგება სტრუქტურირებული, არასტრუქტურირებული და ნახევრად-სტრუქტურირებული მონაცემებისგან .

სტრუქტურირებული მონაცემები შეესაბამება, მონაცემების გარკვეულ მოდელს ან სქემას და ხშირ შემთხვევაში ინახება ცხრილის სახით. ამ ფორმით შენახვა იძლევა სხვადასხვა ობიექტებს შორის კავშირების დადგენის შესაძლებლობას, ამიტომ უმეტეს შემთხვევაში ასეთი მონაცემები ინახება სტანდარტულ რელაციურ მონაცემთა ბაზებში. ასეთი მონაცემების დაგენერირება შეიძლება მოხდეს კორპორატიული პროგრამული უზრუნველყოფით ან ინფორმაციული სისტემით, როგორიცაა ERM ან CRM;

არასტრუქტურირებული მონაცემები არ შეესაბამება მონაცემთა არანაირ სქემას ან მოდელს. ითვლება, რომ ასეთი მონაცემების მოცულობა ნებისმიერ ორგანიზაციაში არის დაახლოებით 80%. არასტრუქურირებული მონაცემების ზრდის ტემპი სტრუქტურირებულ მონაცემთა მოცულობის ზრდის ტემპთან მიმართებაში საკმარისად სწრაფია. ამ ტიპის მონაცემები ხშირად წარმოდგენილია ტექსტური ან ორობითი ფორმით და გადაიცემა, როგორც ფაილი, რომელიც არის ავტონომიური და არარელაციური. ტექსტური ფაილი შეიძლება შეიცავდეს მონაცემებს შეტყობინებებიდან ან ბლოგიდან. ორობითი ფაილი ხშირად არის მედია ფაილი, რომელიც შეიცავს გამოსახულებას, აუდიო- ან ვიდეო-მონაცემებს. ტექნიკურად ტექსტური და ორობითი ფაილის სტრუქტურას განსაზღვრავს ფაილის სტრუქტურა, მაგრამ ეს მომენტი იგნორირებულია და ასევე ისიც, რომ იყოს არასტრუქტურირებული მონაცემთა ფორმატის მიმართ. არასტრუქტურირებულ მონაცემებთან სამუშაოდ სტანდარტული რელაციური ბაზების ნაცვლად გამოიყენება NoSQL ბაზები;

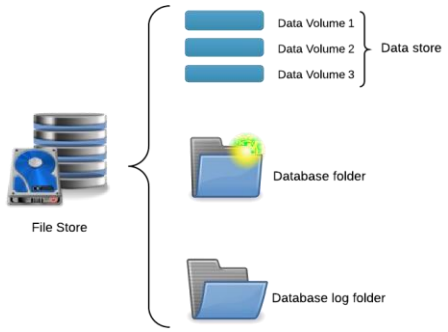
ნაწილობრივ-სტრუქტურირებული მონაცემების საერთო წყაროს მოიცავს ელექტრონული მონაცემების გაცვლა (EDI), ელექტრონული ცხრილები, RSS არხები და მიკროკონტროლერის მიერ გენერირებული მონაცემები. ასეთ მონაცემებს დამუშავების და შენახვის მოქმედებების მიმართ ხშირად გააჩნიათ სპეციალური მოთხოვნები, განსაკუთრებით თუ საბაზისო ფორმატი არ ეყრდნობა ტექსტს.

2. სტრუქტურირებული მონაცემთა ბაზების ინფრასტრუქტურა

მონაცემთა ბაზების გამოყენების უპირველეს მოტივაციას წარმოადგენს მონაცემთა წვდომის სიჩქარე . თუ მონაცემთა ბაზის სქემა სწორად არის შერჩეული, მაშინ, მიუხედავად მონაცემთა ბაზის ზომებისა და მასში არსებული ჩანაწერების რაოდენობისა, ბიზნეს პროცესი შეუფერხებელია. მონაცემთა ბაზები უზრუნველყოფენ მონაცემთა შენახვის ეფექტურ, უსაფრთხო და მოქნილ ხერხს და გამოიყენებიან სხვადასხვა მიმართულების პროგრამებში.

მონაცემთა ბაზების მართვის სისტემები ასევე უზრუნველყოფენ ბაზებში არსებული მონაცემების დაცვას და უსაფრთხო შენახვას. მონაცემთა ბაზების მართვის სისტემას შეუძლია ერთდროულად მოემსახუროს ათი ათასობით მომხმარებელს, ამასთან ერთად უზრუნველყოს ბაზებში არსებული მონაცემების საიმედო უსაფრთხოება და მთლიანობა.

მონაცემთა ბაზა, თავის მხრივ, შეიცავს რელაციური მოდელის მიხედვით დაპროექტებულ ცხრილებს, სქემებს (Views), ტრიგერებს, შენახული პროცედურებს და ინფორმაციის დასამუშავებლად აუცილებელ სხვა ელემენტებს.



SQL Server - რელაციური მონაცემთა ბაზაა და შინაარსობრივი თვალსაზრისით იყოფა შემდეგნაირად:

- DDL - Data Definition Language - მონაცემთა განსაზღვრის ენა
- DML - Data Manipulation Language - მონაცემებით მანიპულირების ენა
- TCL - transaction Control Language - ტრანზაქციების კონტროლის ენა
- DCL - Data Control Language - მომხმარებელთა უფლებების კონტროლის ენა

მონაცემთა ბაზების აპარატულ საცავს ფიზიკური ან ვირტუალური ჰოსტი წარმოადგენს, რომელიც შეიძლება ერთი ან მეტი მონაცემთა ბაზების სერვერ(ებ)ისგან (ეგზემპლარი, ინსტანსი) შედგებოდეს. მონაცემთა ბაზის სერვერი SQL Server წარმოადგენს დამოუკიდებელ პროგრამას,რომელიც შემდეგი ძირითადი კომპონენტებისგან შედგება:

- მონაცემთა ბაზების მართვის ინსტრუმენტი (Database Engine)
- მეხსიერების მართვის ინსტრუმენტი (Storage engine)
- უსაფრთხოების ქვესისტემა (Security Subsystem)
- პროგრამული ინტერფეისი (Programming Interfaces)

- სერვერის აგენტი (SQL Server Agent)
- რეპლიკაციის მოდული (Replication)
- რელაციურ ოპერაციათა მართვის ინსტრუმენტი (Relational engine)

database Engine დაცვის დონეზე იუზერსა და ობიექტებს შორის ურთიერთობის გამარტივებისათვის იყენებს სქემებს. სქემა არის კოლექცია ბაზის ობიექტების(ცხრილები, წარმოდგენები, პროცედურები), რომელიც არის ერთი ადამიანის მფლობელობაში და ქმნის სახელდებულ სივრცეს, სქემაში არ შეიძლება ორი ერთნაირი ტიპის ობიექტს ჰქონდეს ერთნაირი სახელი. მაგ. ორი ცხრილი ერთიდაიგივე სახელით).

3. სტრუქტურირებული მონაცემთა ბაზის ადმინისტრირება ,მიზნები და ამოცანები

მონაცემთა დამუშავებაში ადმინისტრირების ნაწილს მნიშვნელოვანი როლი აქვს, რის საფუძველზეც შესაძლებელია ისეთი მექანიზმების შემუშავება, რომლებიც :

- მაქსიმალურად შეამცირებს იფორმაციის დაკარგვის ალბათობას;
- უზრუნველყოფს დაკარგული ინფორმაციის სწრაფ აღდგენას .

მოცემული მიზნების მისაღწევად მონაცემთა ბაზებზე სრულდება შემდეგი პროცედურები:

- მონაცემთა ბაზების სარეზერვო ასლის აღება;
- მონაცემთა ბაზების მირორიგი;
- მონაცემთა ბაზების მეილის კოფიგურაცია;
- მონაცემთა ბაზების სერვერთა კლასტერის შექმნა.

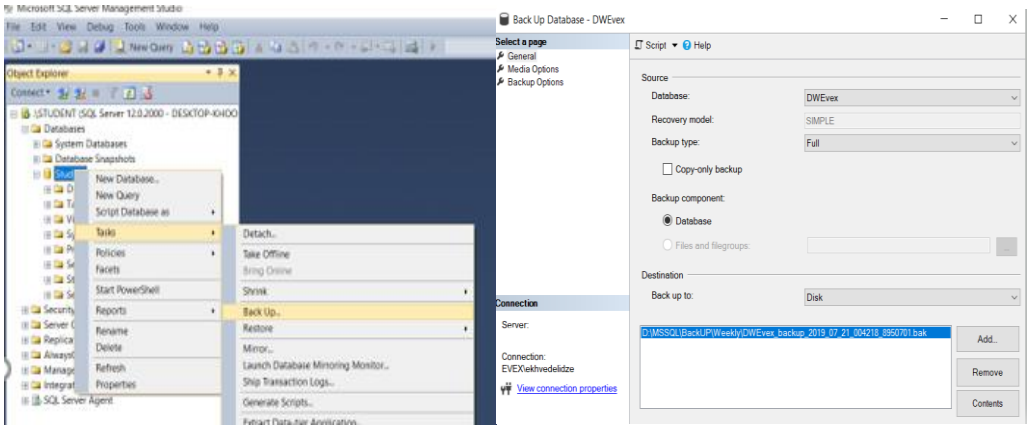
3.1 მონაცემთა ბაზების სარეზერვო ასლის დაგეგმვა და განხორციელება - როგორც მონაცემთა დამუშავების ერთ-ერთი მიდგომა

როდესაც მონაცემებს ვამუშავებთ უნდა ვუზრუნველყოთ მათი მთლიანობა და დაცვა. სწორედ ეს არის მონაცემთა ბაზის სარეზერვო ასლის (Backup) შექმნის ერთ-ერთი მიზეზი , რაც გულისხმობს ბაზის ასლის შექმნას კონკრეტული დროის მდგომარეობით. მისი მიზანი სარეზერვო ასლიდან მონაცემთა აღდგენაა იმ შემთხვევაში, თუ რაიმე მიზეზით თავდაპირველი ფაილი დაზიანდა.

სრული სარეზერვო ასლი შეიცავს ყველა მონაცემს, რაც კი მისი შექმნის დროს არსებობდა. ეს რეჟიმი ქმნის საფუძველს მომავალში ტრანზაქციული ან დიფერენციალური სარეზერვო ასლების შესაქმნელად. ამრიგად, სრული სარეზერვო ასლი შეიცავს მონაცემთა ბაზის ყველა მონაცემს სარეზერვო ასლის შექმნის ოპერაციის დამთავრების მომენტისათვის. ეს კი ნიშნავს,

რომ თუ რომელიმე მომხმარებელმა სარეზერვო ასლის შექმნის პროცესში შეცვალა მონაცემები რომელიმე ცხრილში, მაშინ ეს ცვლილებები დაუყოვნებლივ აისახება სარეზერვო ასლში. ამ პროცესის რეალიზება SQL - ის დახმარებით შეიძლება წარმოვადგინოთ შემდეგი კოდის სახით:

```
BACKUP DATABASE [Student] TO  DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.STUDENT\MSSQL\Backup\Student.bak' WITH NOFORMAT,
NOINIT,
NAME = N'Student-Full Database Backup', SKIP, NOREWIND, NOUNLOAD,
STATS = 10
GO
```



სურათი 1

თავდაპირველად ხდება მონაცემთა ბაზის არჩევა , Student → Tasks → Back up , შემდეგ კი ვირჩევთ სარეზერვო ასლის ტიპს. ასევე, ვუთითებთ სარეზერვო ასლის ფაილის ლოკაციას.

ტრანზაქციული სარეზერვო ასლი განსაკუთრებით გამოიყენება იმ შემთხვევაში, როდესაც საჭიროა სარეზერვო ასლების ხშირი შექმნა და წარსული დროის კონკრეტულ მომენტში არსებულ მდგომარეობამდე აღდგენის შესაძლებლობის ქონა. სრული სარეზერვო ასლის შექმნის შემდეგ თუ თვის ყოველ დღეს შევქმნით ინკრემენტულ სარეზერვო ასლს, მიიღება იგივე შედეგი, როგორიც თითქოს თვის ყოველ დღეს სრულდებოდა სრული სარეზერვო ასლის შექმნა. როგორც წესი, ინკრემენტული სარეზერვო ასლები არსებითად მცირეა სრულ ან დიფერენციალურთან შედარებით. სარეზერვო ასლის შექმნის ამ მეთოდის გამოყენების სცენარს შეიძლება წარმოადგენდეს სისტემის ყოველკვირეული სრული სარეზერვო ასლის შექმნა მხოლოდ იმ მონაცემთა სარეზერვო ასლების შექმნის ყოველდღიური ინტერვალით, რომლებმაც განიცადეს ცვლილება ბოლო სარეზერვო ასლის შექმნის შემდეგ.

მოცემული სცენარის უპირატესობას წარმოადგენს საცავის სივრცისა და დროის ეკონომია, რაც საჭიროა ყოველდღიური სარეზერვო ასლის შექმნისას. თუმცა, სისტემის დაზიანების შემთხვევაში მის აღდგენას მეტი რესურსი დასჭირდება. თუ დაზიანება მოხდება ხუთშაბათს, მაშინ საჭირო გახდება ორშაბათის, სამშაბათის და ოთხშაბათის ბოლო სრული სარეზერვო ასლის აღდგენა. ტრანზაქციული სარეზერვო ასლის დაკარგვის ან დაზიანების შემთხვევაში ყველა მომდევნო ტრანზაქციული ასლის გამოყენება შეუძლებელი გახდება.

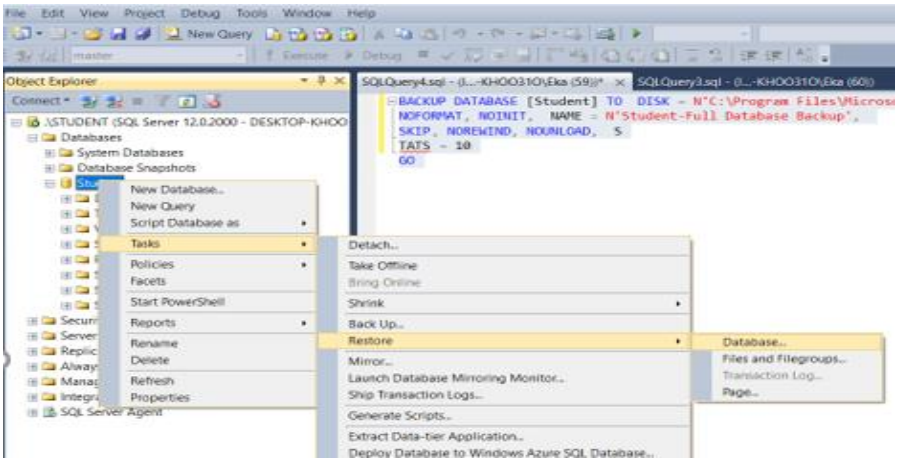
ამ პროცესის რეალიზება SQL -ის დახმარებით შეიძლება წარმოვადგინოთ შემდეგი კოდის სახით:

```
BACKUP LOG [Student] TO  DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.STUDENT\MSSQL\Backup\Student.bak' WITH NOFORMAT,
NOINIT,  NAME = N'Student-Full Database Backup',
SKIP, NOREWIND, NOUNLOAD,
STATS = 10
GO
```

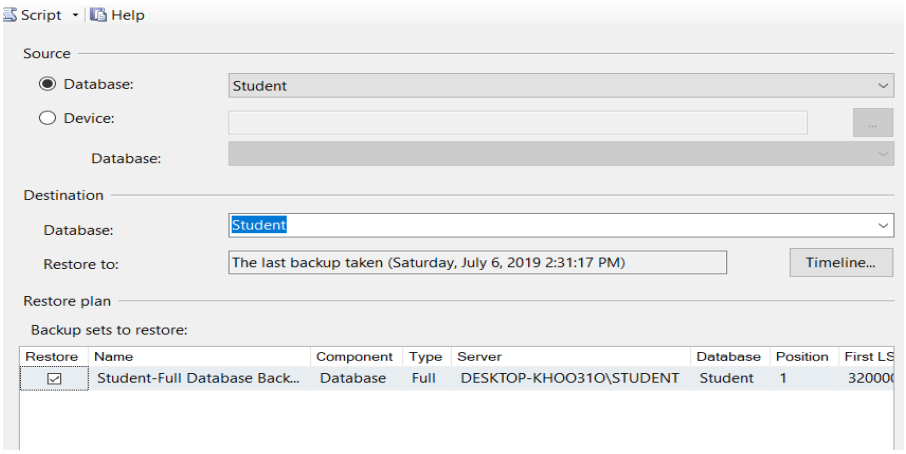
მონაცემთა ბაზის დიფერენცირებული სარეზერვო ასლი შეიცავს ინფორმაციას იმ ცვლილებების შესახებ, რომლებიც შესრულდა მონაცემთა ბაზაში მისი სრული სარეზერვო ასლის უკანასკნელი შექმნის მომენტიდან. ეს მეთოდი შესანახად საჭიროებს ნაკლებ დროსა და ადგილს სრული სარეზერვო ასლის შექმნასთან შედარებით, მაგრამ მეტს ტრანზაქციულთან შედარებით. აღდგენა უფრო მარტივია, ვიდრე მეორე მეთოდის გამოყენებისას, რადგან საკმარისი იქნება ბოლო სრული და დიფერენციალური სარეზერვო ასლის აღდგენა. დიდი ზომის მონაცემთა ბაზებში, რომლებშიც ცვლილებების რაოდენობა მცირეა, დიფერენცირებული სარეზერვო ასლის შექმნა ყველაზე ოპტიმალური მეთოდია.

```
BACKUP DATABASE [Student] TO  DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.STUDENT\MSSQL\Backup\Student.bak' WITH  DIFFERENTIAL ,
NOFORMAT, NOINIT,  NAME = N'Student-Full Database Backup',
SKIP, NOREWIND, NOUNLOAD,
STATS = 10
GO
```

სარეზერვო ასლის შექმნის მეთოდის ასარჩევად აუცილებელია სარეზერვო ასლის შექმნის სამომხმარებლო სქემის აგება. ტრანზაქციული და დიფერენცირებული სარეზერვო ასლის შექმნა SQL-ის ინტერფეისიდან სრული სარეზერვო ასლის მსგავსად ხდება . სარეზერვო ასლიდან აღდგენა წარმოდგენილია სურათი 2-ისა და სურათი 3-ის სახით.



სურათი 2



სურათი 3

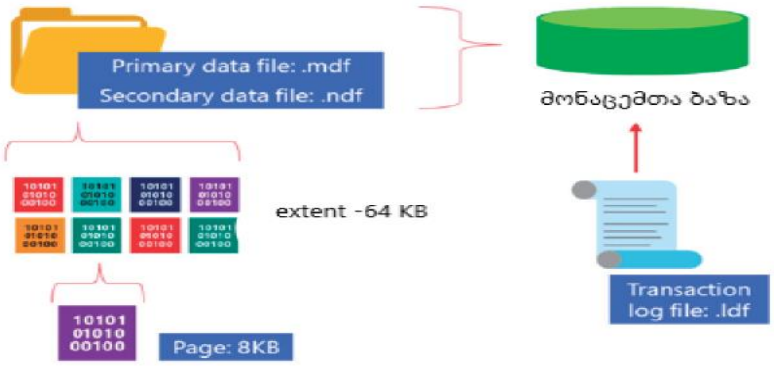
სარეზერვო ასლიდან აღდგენა კოდის სახით :

```
USE [master]
BACKUP LOG [Student] TO  DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.STUDENT\MSSQL\Backup\Student_LogBackup_2019-07-06_14-31-42.bak' WITH
NOFORMAT,
NOINIT,  NAME = N'Student_LogBackup_2019-07-06_14-31-42', NOSKIP, NOREWIND, NOUNLOAD,
NORECOVERY ,
STATS = 5
RESTORE DATABASE [Student] FROM  DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.STUDENT\MSSQL\Backup\Student.bak' WITH  FILE = 1,
NOUNLOAD,  STATS = 5
GO
```

4. მონაცემების შენახვა სტრუქტურირებულ ბაზებში

ნებისმიერი მონაცემთა ბაზა ყოველთვის იქმნება ორი ფაილით:

1. მონაცემების (Primary data file), გაფართოებით: .MDF, რომელიც განთავსებულია PRIMARY ჯგუფში. აქვე შესაძლებელია მეორადი (secondary data files) ფაილების არსებობაც .NDF გაფართოებით. თუმცა ამ ტიპის ფაილების შექმნა შესაძლებელია მომხმარებლის მიერ შექმნილ ახალ ჯგუფებშიც. Primary ფაილები შეიცავენ: სისტემურ ობიექტებს და მონაცემებს. მომხმარებლის მიერ განსაზღვრულ ობიექტებს და მონაცემებს; როგორიცაა ცხრილები, ინდექსები, შენახული პროცედურები, წარმოდგენები. ნებისმიერი მეორად ფაილების მდებარეობის შესახებ ინფორმაციას.
2. ტრანზაქციის ჟურნალის ფაილები (transaction log)-ტრანზაქციის ჟურნალიდან ინახავენ ბაზის აღდგენისათვის საჭირო ინფორმაციას. ბაზას შეიძლება გააჩნდეს რამდენიმე ტრანზაქციის ჟურნალი. ასეთი ტიპის ფაილების გაფართოებაა .ldf;



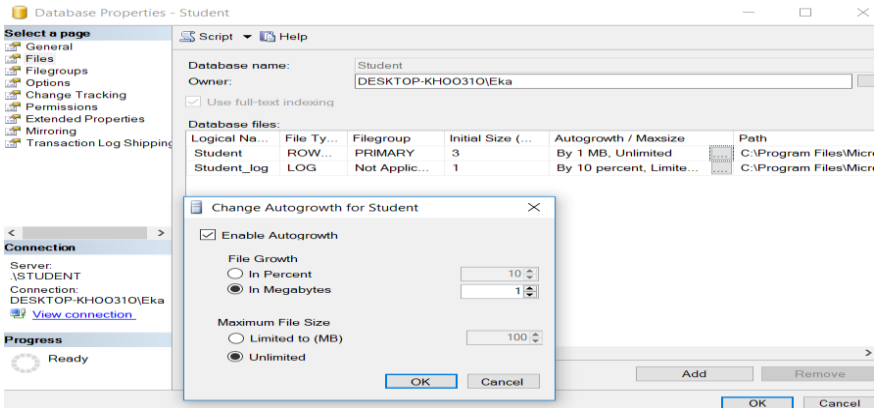
თითოეულ ფაილს გააჩნია ორი სახელი: ლოგიკური და ფიზიკური. ლოგიკური სახელი გამოიყენება T-SQL-ის მოთხოვნებში, ხოლო ფიზიკური სახელით ფაილი ინახება დისკზე. მონაცემთა ბაზის ობიექტები და ფაილები განაწილების და ადმინისტრირების ფუნქციებისშესაბამისად, დაჯგუფებულია ფაილურ ჯგუფებში.

არსებობს ორი ტიპის ფაილური ჯგუფი: Primary და User-Defined.

- Primary შეიცავს Primary ფაილებს (სისტემური ცხრილების ფაილები)
- User-Defined ფაილური ჯგუფები არის ის ჯგუფები, რომლებიც იქმნება Filegroup საგასაღებო სიტყვით Create Database ან Alter Database წინადადებაში.

ტრანზაქციის ჟურნალის ფაილები არ მიეკუთვნებიან არცერთ ჯგუფს. ამ ჯგუფის ფაილების სივრცის მართვა ხდება მონაცემების ფაილების სივრცისაგან დამოუკიდებლად.

მონაცემთა ბაზის ყველა ფაილის ზომა შეიძლება ავტომატურად გაიზარდოს. ეს შესაძლებლობა ფაილის შექმნის დროს განისაზღვრება. ნაზრდის ზომა შესაძლებელია განისაზღვროს პროცენტებში (ფაილის საწყისი ზომიდან) ან მეგაბაიტებში. დამატებით, შეიძლება მივუთითოთ ფაილის მაქსიმალური ზომა. თუ მაქსიმალური ზომა არ არის მითითებული, მაშინ ფაილის ზომა გაიზრდება მანამ, სანამ არის თავისუფალი სივრცე დისკზე.ამ პროცესის რეალიზება წარმოდგენილია დანართი 1.4-ის სახით.



სურათი 1.4

4.1 RAID - მონაცემების შენახვის ტექნოლოგია

მონაცემების შენახვის ტექნოლოგიები სწრაფად ვითარდება და სერვერიდან ის გადაიზარდა ქსელში მონაცემების შესანახ ტექნოლოგიად. თანამედროვე გამანაწილებელი არქიტექტურა ერთ კომპონენტად აერთიანებს გამოთვლებს, შესანახ მოწყობილობას, მეხსიერებას და ქსელს, სადაც არქიტექტურა შეიძლება იმართებოდეს მხოლოდ ერთი ადგილიდან. ასეთი მიდგომა განსაკუთრებით მნიშნელოვანია დიდი მონაცემების დამუშავებისას. დიდმა მონაცემებმა გარკვეულწილად შემოსაზღვრა მეხსიერების და შესანახი მოწყობილობის მოცულობა კლასტერებზე. დიდი მონაცემების რეალურ დროში ანალიზისთვის მნიშვნელოვანია როგორც მეხსიერების ისე, შესანახი მოწყობილობების ინოვაციური ტექნოლოგიები.

RAID (Redundant Array of Inexpensive Disks) არის ტექნოლოგია, რომელიც გამოიყენება მონაცემთა შენახვისა და სანდოობის გაზრდის მიზნით. RAID სისტემა მოიცავს პარალელურად მომუშავე ორ ან მეტ დისკს. არსებობს სხვადასხვა ტიპის RAID სისტემა, რომლებიც ოპტიმიზირებულია კონკრეტული სიტუაციისთვის. ნაშრომში განხილულია RAID 0, RAID 1, RAID 10 სისტემა.

RAID 0 - სისტემური მონაცემები დაყოფილია ბლოკებად და შესაძლებელია სულ ცოტა ორი დისკის გამოყენება, მონაცემების ჩაწერა ხდება ორივე დისკზე პარალელურ რეჟიმში. RAID 0 გვთავაზობს დიდ სპექტრს წაკითხვისა და ჩაწერის ოპერაციებში. მთლიანი საცავის ტევადობა ათვისებულია და იძლევა შესრულების მარტივ გზას.აღნიშნულ დადებით თვისებებთან ერთად აქვს უარყოფითი თვისებაც - ერთი დისკის დაზიანების შემთხვევაში,ამ დისკზე არსებული მთლიანი მონაცემები დაიკარგება. RAID 0 არ უნდა იყოს გამოყენებული კრიტიკული სისტემებისთვის. მისი გამოყენება იდეალურია ისეთი სისტემისათვის, რომლისთვისაც პრიორიტეტულია ინფორმაციის წაკითხვა/ჩაწერა მაღალ სიჩქარეზე.

RAID 1 - მონაცემების ჩაწერა ხდება ორჯერ , ეს ნიშნავს, რომ ერთ დისკზე ჩაწერილი მონაცემები პარალელურად გადააქვს მეორე დისკზე . ამ შემთხვევაშიც შესაძლებელია მინიმუმ ორი დისკის გამოყენება . RAID 0 - სგან განსხვავებით ერთი დისკის დაზიანების შემთხვევაში გვაქვს მონაცემების სრული ასლი მეორე დისკზე. RAID 1- ის მთავარი მინუსია ინფორმაციის წაკითხვა/ჩაწერის საშუალო სიჩქარეა.

RAID 10 - ში გაერთიანებულია RAID 1-სა და RAID 0 -ის დადებითი თვისებები . საუკეთესო არჩევანია , რადგან ინფორმაციის წაკითხვა/ჩაწერა ხდება მაღალ სიჩქარეზე. საჭიროებს მინიმუმ 4 დისკს, მონაცემების ჩაწერა და წაკითხვა ხდება პირველსა და მეორე დისკზე.პარალელურად ამ ორი დისკის მირორინგი მესამე,მეოთხე დისკზე.

4.2 უსაფრთხოების სისტემა

უსაფრთხოების სისტემა მოიცავს ორ ქვესისტემას:

- Windows security
- SQL Server security

Windows security უზრუნველყოფს დაცვას ოპერაციული სისტემის დონეზე. ეს არის მეთოდი, რომლითაც იუზერი უკავშირდება სისტემას user account-ის საშუალებით.

SQL Server security არის დამატებითი უსაფრთხოება სისტემის დონეზე - რაც ნიშნავს სერვერთან კავშირისთვის დამატებითი იუზერის შექმნას. უსაფრთხოების ეს მეთოდი მოითხოვს SQL Server -ის Log-ის შექმნას.

ავთენტიფიკაციის რეჟიმების დაფიქსირება ხდება სერვერის ინსტალაციის პროცესში, თუმცა შესაძლებელია შემდეგ ამ რეჟიმის შეცვლაც. Database Engine - თან დაკავშირების შემდეგ, იუზერის წვდომა ბაზის ობიექტებთან არ არის დამოკიდებული ავთენტიფიკაციის მეთოდებზე. სერვერზე იქმნება ლოგები, რომელთაც ენიჭებათ გარკვეული უფლებები სერვერის დონეზე. ლოგებისათვის კეთდება საარიცხვო ჩანაწერები, იმისათვის რომ მათ მიენიჭოთ გარკვეული უფლებები სერვერთან სამუშაოდ. ლოგი შეიძლება იყოს სისტემის ავთენტიფიკაციით, ან სერვერის ავთენტიფიკაციით.

ბაზას გააჩნია სხვადასხვა რეჟიმები: Read_only- მხოლოდ კითხვის რეჟიმი, Read_write - შესაძლებელია ბაზასთან სრულყოფილი მუშაობა.

არსებობს ბაზის შემდეგი რეჟიმები:

- **ONLINE**- შესაძლებელია მუშაობა მონაცემებზე და მეტამონაცემებზე. ოპერაციული სისტემის საშუალებით შეუძლებელია ბაზის ფაილების წაშლა ან კოპირება.
- **OFFLINE**-ყველანაირი მოქმედების შესრულება ბაზასთან შეზღუდულია.ოპერაციული სისტემის საშუალებით შესაძლებელია ბაზის ფაილების წაშლა ან კოპირება.
- **RESTORING** -ამ მდგომარეობაში ბაზა მიუწვდომელია. ამ რეჟიმში იგი გადადის მაშინ როდესაც ხდება ფაილების აღდგენა.
- **RECOVERING**-ბაზის აღდგენის რეჟიმი. ამ დროს ყველა მოქმედება შეზღუდულია, აღდგენის დასრულების შემდეგ ბაზა ავტომატურაად გადადის ONLINE მდგომარეობაში.
- **RECOVERY_PENDING**-აღდგენის პროცესში მოხდა შეცდომა და საჭირო ხდება მომხმარებლის ჩარევა.
- **SUSPECT**- ბაზა საეჭვოა ანდაზიანებულია. საჭიროა მომხმარებლის ჩარევა.
- **EMERGENCY**- ბაზა დაზიანებულია და არის Read_only რეჯიმში.

Login-ის შექმნა T-SQL-ით:

CREATE LOGIN წინადადებით შეიქმნება ლოგი. სრული სინტაქსი ახალი ლოგის შექმნის არის შემდეგი:

CREATE LOGIN login_name

{ WITH option_list1 |

FROM {WINDOWS [WITH option_list2 [,...]]

- **login_name-სახელი;**
- **option_list1** -ეს სია შეიცავს რამდენიმე თვისებას, რომელთაგან ძირიტადია პაროლი. შესაძლებელია აგრეთვე ეს თვისებები იყოს: DEFAULT_DATABASE, DEFAULT_LANGUAGE, and CHECK_EXPIRATION.)
- FROM წინადადება შეიცავს შემდეგ თვისებებს:
    - WINDOWS – მისი პარამეტრები განსაზღვრავს ლოგინის შექმნას Windows-ის მომხმარებლის ანგარიშით. ამ ბრძანების დაზუსტება შესაძლებელია ქვეპარამეტრებით, როგორიცაა DEFAULT_DATABASE and DEFAULT_LANGUAGE;

ქვემოთ მოყვანილია ლოგის შექმნის სხვადასხვა ვარიანტი:

```
Log-ის შექმნა SQL -ს ავთენტიფიკაციით
CREATE LOGIN [EKA] WITH PASSWORD=N'123' MUST_CHANGE,
DEFAULT_DATABASE=[master]
GO
Log-ის შექმნა Windows-ს user-ის  ავთენტიფიკაციით
CREATE LOGIN [USER-PC\EKA]
FROM WINDOWS WITH DEFAULT_DATABASE=[master]
      GO

Log-ის შეცვლა
 ALTER LOGIN EKA WITH PASSWORD='123'
 ALTER LOGIN EKA WITH NAME=EKATERINE
```

როლის მინიჭება:

```
alter server role sysadmin
 add member EKA

Log-ის წაშლა
       DROP LOGIN EKA
```

სისტემური ბაზიდან შესაძლებელია ლოგების შესახებ ინფორმაციის ამოღება. Syslogin ცხრილი შეიცავს ინფორმაციას ლოგების შესახებ .

```
SELECT
    name
  , createdate
  , updatedate
  , isntuser
  , isntgroup
FROM syslogins;
```

ქვემოთ მოყვანილი კოდი დააბრუნებს მხოლოდ იმ ლოგებს, რომლებიც შექმნილია SQL სერვერის ავთენტიფიკაციით.

```
SELECT *
FROM sys.sql_logins
```

მხოლოდ ლოგების შექმნა ბაზებთან წვდომისათვის არ არის საკმარისი, საჭიროა მათთვის როლების განსაზღვრა, რაც გულისხმობს ლოგის უფლებებს სერვერზე და ამავე დროს სერვერის დონეზე უფლებებს კონკრეტულ ბაზებთან (user mapping) .

სერვერის უსაფრთხოების სისტემის ადმინისტრირების გასამარტივებლად მომხმარებლებს ვაერთიანებთ როლებში. სერვერზე რეალიზებულია ორი სტანდარტული როლი: სერვერის დონეზე და მონაცემთა ბაზის დონეზე. სერვერის დაინსტალირების დროს იქმნება სერვერისა და მონაცემთა ბაზის ცხრა ფიქსირებული როლი. იმისათვის, რომ მომხმარებელს მივანიჭოთ ის უფლებები, რომელიც აქვს სერვერის რომელიმე ფიქსირებულ როლს, საჭიროა მომხმარებლის ამ როლში ჩართვა.,როლი არის ბაზის ობიექტი, რომელიც შეიცავს კონკრეტულ ობიექტებთან წვდომის პრივილეგიებს. როდესაც რამდენიმე მომხმარებელს სჭირდება ერთიდაიგივე მოქმედებების შესრულება კონკეტულ ბაზებთან, ისინი ერთიანდებიან როლში.

სერვერის როლები

1. Sysadmin - ამ როლის წევრებს შეუძლიათ ნებისმიერი აქტივობის შესრულება სერევრზე.
2. Serveradmin - ამ როლის წევრებს შეუძლიათ სერვერის კონფიგურაციის შეცვლა და სერვერის გათიშვა.ჩვეულებრივ , მონაცემთა ბაზის ადმინისტრატორს აქვს სერვერის კონფიგურაციის შეცვლისა და გათიშვის უფლება.
3. Securityadmin - ამ როლის წევრებს შეუძლიათ სერვერის ლოგების თვისებების შეცვლა. მათ აქვთ GRANT, DENY, და REVOKE უფლებები, როგორც სერვერის, ასევე ბაზის დონეზე. მათ შეუძლიათ აგრეთვე ლოგების პაროლების შეცვლაც. securityadmin როლს sysadmin -ის მსგავსი უფლებები აქვს.
4. Processadmin - ამ როლის წევრებს შეუძლიათ SQL Server-ზე გაშვებული პროცესების გათიშვა.
5. Setupadmin - ამ როლის წევრებს შეუძლიათ Linked Server-თან კავშირის უზრუნველყოფა.
6. Bulkadmin - მათ შეუძლიათ BULK INSERT ბრძანების შესრულება.
7. Diskadmin - შეუძლიათ დისკის ფაილების მართვა.
8. Dbcreator - მათ შეუძლიათ create, alter, drop, and restore database ბრძანებების შესრულება.
9. Public- ეს არის სპეციალური ფიქსირებული როლი, რომელიც ენიჭება მონაცემთა ბაზის ნებისმიერ იუზერს. იგი ანიჭებს მას ყველა default უფლებას. მისი წაშლა შეუძლებელია. ამ როლით იუზერს ენიჭება შემდეგი უფლებები: სისტემური ცხრილების დათვალიერება და შესაბამისად ინფორმაციის გამოტანა master სისტემური ბაზიდან შესაბამისი სისტემური პროცედურით;

```
select loginname ,* from syslogins
```

100 %

Results | Messages

	loginname	sid	s	createdate	updatedate
1	sa	0x...		2003-04-08 09:10:35.460	2018-09-21 17:47:02.783
2	##MS_SQLResourceSigningCertificate##	0x...		2018-03-20 16:13:14.233	2018-03-20 16:13:14.233
3	##MS_SQLReplicationSigningCertificate##	0x...		2018-03-20 16:13:14.237	2018-03-20 16:13:14.237
4	##MS_SQLAuthenticatorCertificate##	0x...		2018-03-20 16:13:14.240	2018-03-20 16:13:14.240
5	##MS_PolicySigningCertificate##	0x...		2018-03-20 16:13:14.243	2018-03-20 16:13:14.243
6	##MS_SmoExtendedSigningCertificate##	0x...		2018-03-20 16:13:14.243	2018-03-20 16:13:14.243
7	##MS_PolicyTsqlExecutionLogin##	0x...		2016-04-30 00:46:49.400	2018-03-20 16:15:11.717
8		0x...		2018-03-20 16:13:15.690	2018-03-20 16:13:15.700
9	NT SERVICE\SQLWriter	0x...		2018-03-20 16:13:15.707	2018-03-20 16:13:15.717
10	NT SERVICE\Winmgmt	0x...		2018-03-20 16:13:15.717	2018-03-20 16:13:15.727
11	NT Service\MSSQLSERVER	0x...		2018-03-20 16:13:15.723	2018-03-20 16:13:15.733
12	NT AUTHORITY\SYSTEM	0x...		2018-03-20 16:13:15.730	2018-03-20 16:13:15.737
13	NT SERVICE\SQLSERVERAGENT	0x...		2018-03-20 16:14:34.560	2018-03-20 16:14:34.570
14	NT SERVICE\SQLTELEMETRY	0x...		2018-03-20 16:14:35.927	2018-03-20 16:14:35.933

მონაცემთა ბაზის როლები

1. db_owner - ამ როლის წევრებს შეუძლიათ ნებისმიერი მოქმედების შესრულება ბაზასთან მუშაობის პროცესში.
2. db_accessadmin - ამ როლის წევრებს შეუძლიათ დაამატონ ან წაშალონ იუზერი.
3. db_datareader - ამ როლის წევრებს შეუძლიათ ბაზაში მონაცემთა ცხრილების დათვალიერება.
4. db_datawriter - ამ როლის წევრებს შეუძლიათ ბაზის ცხრილებში მონაცემების დამატება, წაშლა და მოდიფიცირება. (add, delete, or change Data)
5. db_ddladmin - ამ როლის წევრებს შეუძლიათ DDL ოპერაციების განხორციელება.
6. db_securityadmin - მომხმარებელი, რომელსაც შეუძლია ბაზის უსაფრთხოებასთან დაკავშირებული მოქმედებების შესრულება.
7. db_backupoperator - ამ როლის წევრებს შეუძლიათ მონაცემთა ბაზის სარეზერვო ასლის შექმნა.
8. db_denydatareader - ამ როლის წევრებს შეზღუდული აქვთ მონაცემთა დათვალიერების უფლება.
9. db_denydatawriter - ამ როლის წევრებს შეზღუდული აქვთ მონაცემთა ბაზაში მონაცემების დამატება,წაშლა და ცვლილება.

5. ინდექსები, როგორც მონაცემებზე წვდომის ერთ-ერთი ძირითადი ინსტრუმენტი

სტრუქტურირებულ მონაცემთა ბაზებში ძებნის დაჩქარების მიზნით ინდექსები გამოიყენება. მონაცემების ძებნის დაჩქარება მათი ფიზიკური ან ლოგიკური მოწესრიგების ხარჯზე მიიღწევა და შესაძლებელია ცხრილის ერთი ან რამდენიმე სვეტისგან შედგებოდეს. როცა სრულდება მოთხოვნა, რომელიც ინდექსირებულ სვეტს მიმართავს, სერვერი ინდექსს საჭირო მნიშვნელობის მოძებნის მიზნით ავტომატურად აანალიზებს. შემუშავებულია ეფექტური მათემატიკური ალგორითმები მოწესრიგებულ მიმდევრობაში მონაცემების მოსაძებნად. ერთ-ერთი მათგანია „შუაზე გაყოფის“ მეთოდი. როდესაც ცხრილიდან მონაცემების ამორჩევა დიდი დროს იკავებს , ეს არის საფუძველი ვიფიქროთ ინდექსის

საჭიროებაზე. ამასთანავე, სვეტის ამორჩევისას გასანაალიზებელია თუ რა ტიპის მოთხოვნები შესრულდება ყველაზე ხშირად და რომელი სვეტები არის საკვანძო . არ უნდა მოხდეს იმ სვეტების რეალიზება, რომლებზე მიმართვა საერთოდ არ ხდება ან ხდება იშვიათად. ასევე,გრძელი სვეტების ინდექსირება.

არსებობს ინდექსების შემდეგი ტიპები:

- Clustered
- Nonclustered
- Unique
- Filtered
- XML
- Full Text
- Spatial
- Columnstore
- Index with included columns
- Index on computed colums

ნაშრომში განვიხილავ კლასტერულ, არაკლასტერულ და უნიკალურ ინდექსებს.

არაკლასტერული ინდექსი - შეიცავს ერთი ან მეტი სვეტის მნიშვნელობებს დალაგებულს ზრდადობით ან კლებადობით, აგრეთვე სტრიქონებზე მიმართვებს. მნიშვნელობების ძებნა სრულდება ზრდადობით ან კლებადობით დალაგებულ უბანში. ნაპოვნი მნიშვნელობის გასწვრივ მოთავსებულია შესაბამის სტრიქონზე მიმართვა,რომელიც შეიცავს სტრიქონის რეალურ მისამართს ცხრილში. არაკლასტერული ინდექსის გამოყენებისას, სერვერი მიმართავს ინდექსს და ასრულებს საჭირო სტრიქონის ძებნას.სტრიქონის პოვნისას, მიმართვის მიხედვით შესრულდება გადასვლა ცხრილში, მოძებნილ სტრიქონზე.

ცხრილში მხოლოდ ერთი ინდექსი შეიძლება იყოს კლასტერული. კლასტერული ინდექსის გამოყენების დროს ცხრილში ხდება მონაცემების ფიზიკური განლაგების გადაწყობა ინდექსის სტრუქტურის შესაბამისად. კლასტერული ინდექსი მნიშვნელოვნად ზრდის მონაცემების ძებნის მწარმოებლურობას. კლასტერულ ინდექსად უნდა ავირჩიოთ ყველაზე ხშირად გამოყენებადი სვეტები. კლასტერული ინდექსი არ უნდა გამოვიყენოთ ხშირად ცვალებადი სვეტებისათვის,რადგან სერვერმა უნდა შეასრულოს მონაცემის ფიზიკური გადაადგლება ცხრილში, რომ ისინი მოწესრიგებულ მდგომარეობაში იმყოფებოდეს. ცხრილში პირველადი გასაღების შექმნისას სერვერი მისთვის კლასტერულ ინდექსს ავტომატურად ქმნის, თუ ის ადრე არ იყო შექმნილი ან გასაღების განსაზღვრისას არ იყო აშკარად მითითებული ინდექსის სხვა ტიპი.

უნიკალური ინდექსი იძლევა ინდექსირებულ სვეტში მნიშვნელობების უნიკალურობის გარანტიას. ის შეიძლება რეალიზებული იყოს როგორც კლასტერული, ისე არაკლასტერული ინდექსისათვის. ერთ ცხრილში შეიძლება არსებობდეს ერთი უნიკალური კლასტერული ინდექსი და რამდენიმე უნიკალური არაკლასტერული ინდექსი.მონაცემების

მთლიანობის უზრუნველსაყოფად უმჯობესია UNIQUE ან PRIMARY KEY შეზღუდვების და არაუნიკალური ინდექსის გამოყენება.

ინდექსის შექმნის წინ უმჯობესია მივიღოთ ინფორმაცია არსებული ინდექსების შესახებ. ინდექსების შესახებ ინფორმაციის მისაღებად გამოიყენება sp_helpindex შენახული პროცედურა. მისი სინტაქსია: EXEC sp_helpindex [ცხრილის_სახელი]

ინფორმაციის მისაღებად ინდექსების მიერ დაკავებული სივრცის შესახებ გამოიყენება sp_spaceused შენახული პროცედურა.

```
EXEC sp_spaceused  'dir_products';
```

100 %

Results | Messages

	name	rows	reserved	data	index_size	unused
1	dir_products	118934	145600 KB	127672 KB	17096 KB	832 KB

6. სისტემურ ფუნქციებზე დაყრდნობით მონაცემთა დამუშავება

მონაცემთა დამუშავება ასევე შესაძლებელია სისტემურ ფუნქციებზე დაყრდნობით. მაგალითისთვის , შეგვიძლია გავიგოთ რომელი სკრიპტები ეშვება ყველაზე ხშირად(დღის ნებისმიერ მონაკვეთში), რა ტვირთავს ბაზას და მოვძებნოთ ოპტიმიზაციის გზები. ამ პროცესის რეალიზება SQL -ის დახმარებით შეიძლება წარმოვადგინოთ შემდეგი კოდის სახით:

```
CREATE VIEW [dbo].[Active_Sessions] as
WITH T AS
(
SELECT CONVERT(date, active_sessions_run_time) as [Date],
              *,
              left([SQL Text],150) as Partial_SQL_STR,
              len([SQL Text]) as SQL_Len,
              (case when [SQL Text] in (N'sp_server_diagnostics') then 0 else 1 end) as
filter

FROM   [Database].[dbo].[Table]
WHERE  [Command]='SELECT'
),

Runs AS
(
SELECT               active_sessions_run_time,
                     row_number() over (order by active_sessions_run_time) as RunNo,
                     sum(cast([CPU Time] as bigint)) as Run_Total_CPU_Time,
                     sum(case when [Total Elpsd Time min]<0 then 0 else cast([Total Elpsd
Time min] as bigint) end) as Run_Total_Elpsd_Time_min
```

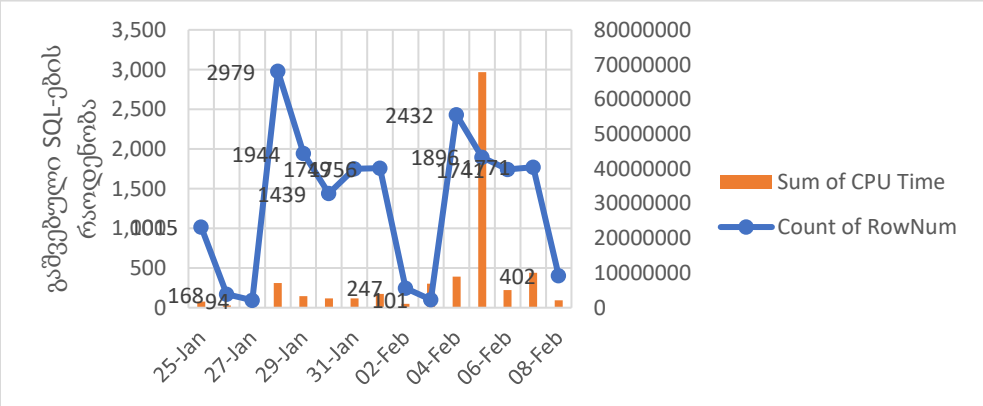
```
FROM          T
GROUP BY      active_sessions_run_time
),

SQL_Group1 AS
(
SELECT *,
              row_number() over (order by Total_CPU_Time1 desc) as SQL_RowNum1
FROM   (
              SELECT Partial_SQL_STR as Partial_SQL_STR1,
                           count(*) as SQLNo1,
                           sum(cast([CPU Time] as bigint)) as Total_CPU_Time1
              FROM   T
              GROUP BY Partial_SQL_STR
              ) t
)

SELECT Runs.RunNo,
              row_number() over (partition by T.active_sessions_run_time order by [CPU
Time] desc) as RowNum,
              Runs.Run_Total_CPU_Time,
              Runs.Run_Total_Elpsd_Time_min,
              SQL_Group1.SQLNo1 as SQLNo,
              SQL_Group1.Total_CPU_Time1 as Total_CPU_Time,
              'SQL'+ right('00000' + cast(SQL_Group1.SQL_RowNum1 as varchar),4) as
SQL_Abbr,
              T.*
FROM   T
              INNER JOIN Runs on T.active_sessions_run_time=Runs.active_sessions_run_time
              LEFT JOIN SQL_Group1 on T.Partial_SQL_STR=SQL_Group1.Partial_SQL_STR1

GO
```

ამ კოდის რეალიზება საფეხურებრივად გვაძლევს ასეთი ტიპის ინფორმაციას დიაგრამის სახით.



6.1 სისტემური მონაცემთა ბაზების დანიშნულება მონაცემთა დამუშავებაში

SQL Server შედგება ოთხი სისტემური ბაზისგან (master,model,msdb,tempdb).თითოეული მათგანი სხვადასვა მიზნებისთვის გამოიყენება.

master მონაცემთა ბაზა მოიცავს SQL Server -ის კონფიგურაციის ,მთელი სისტემური ინფორმაციის შესახებ მონაცემებს.ასევე, ინფორმაციას სხვა მონაცემთა ბაზებისა და მათი პირველადი ფაილების ადგილმდებარეობის შესახებ .master მონაცემთა ბაზა შემდეგი ფაილებისაგან შედგება: Master.mdf (მონაცემების ფაილი) და Mastlog.ldf (ტრანზაქციების ჟურნალის ფაილი).

msdb სისტემური მონაცემთა ბაზა გამოიყენება SQL Server Agent სამსახურის მიერ მოვლენების (alerts), ამოცანებისა (jobs) და ოპერატორების (operators) რეგისტრირების დაგეგმვისათვის. msdb მონაცემთა ბაზა ინახავს მთელ ინფორმაციას, რომელიც ეხება ადმინისტრირების ავტომატიზებასა და სერვერის მართვას. msdb ბაზა ორი ფაილისგან შედგება: msdbdata.mdf (მონაცემების ფაილი) და msdblog.ldf (ტრანზაქციების ჟურნალი).

სერვერზე ახალი მონაცემთა ბაზის შექმნა ხდება მასში model სისტემური მონაცემთა ბაზის ობიექტების გადაწერის გზით. model მონაცემთა ბაზა მოთავსებულია \Data კატალოგში და ორი ფაილისგან შედგება: model.mdf (მონაცემების ფაილი) და model.ldf (ტრანზაქციების ჟურნალი).

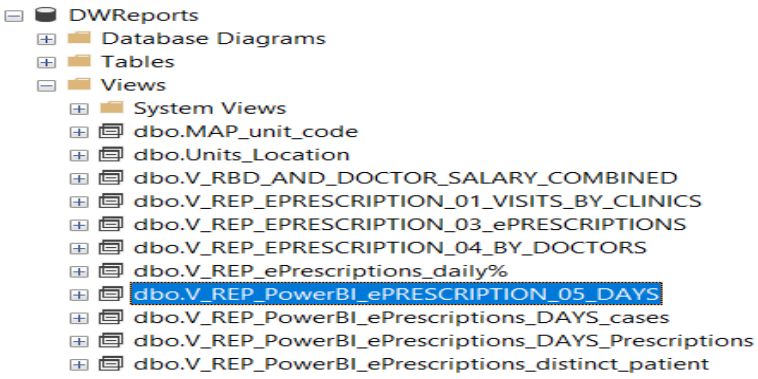
tempdb მონაცემთა ბაზა შეიცავს დროებით ობიექტებს, როგორიცაა ცხრილები, შენახული პროცედურები, ცვლადები, კურსორები და ა.შ. მასში, ინახება, როგორც სისტემური, ისე მიმხმარებლის მიერ შექმნილი ობიექტები. სერვერის ყოველი გაშვების დროს tempdb მონაცემთა ბაზა ხელახლა იქმნება და ყოველთვის იშლება სერვერის გაჩერების დროს. მონაცემთა ბაზის ობიექტები ინახება მხოლოდ ერთი სეანსის განმავლობაში.tempdb

პრაქტიკული მაგალითის რეალიზება

ნაშრომში წარმოდგენილია კორპორაციული მენეჯმენტის ბიზნეს-პროცესების ავტომატიზაციის საკითხები სამედიცინო კორპორაციის მაგალითზე. განხილული მონაცემთა ბაზის მოცულობა დაახლოებით 400 GB-ია და ორასი ცხრილისგან შედგება . ასევე, ბაზის ინტეგრაცია ხდება MOH -თან.

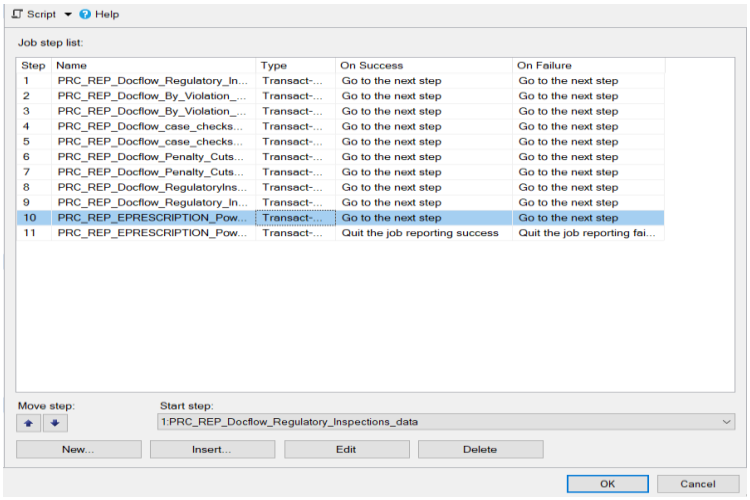
ბიზნესისთვის მნიშვნელოვანია თვალყური ადევნოს თითოეული კლინიკის ჭრილში უნიკალური პაციენტების რაოდენობას, ექიმთან დაფიქსირებულ შემთხვევებს, დაფიქსირებული რეცეპტული შემთხვევებიდან რამდენზე გამოწერა ექიმმა რეცეპტი. გამოწერილი მედიკამენტური დანიშნულებებიდან რამდენის გადაგზავნა მოხდა MOH-ში . მოცემული საკითხების ანალიზისათვის უნდა ვიცნობდეთ ბაზის სტრუქტურას. აღნიშნული

მოთხოვნების დასამუშავებლად აუცილებელია შემდეგი ეტაპების შესრულება: მონაცემების გასაერთიანებლად და დასამუშავებლად ვქმნით ვირტუალურ ცხრილებს - VIEWS .



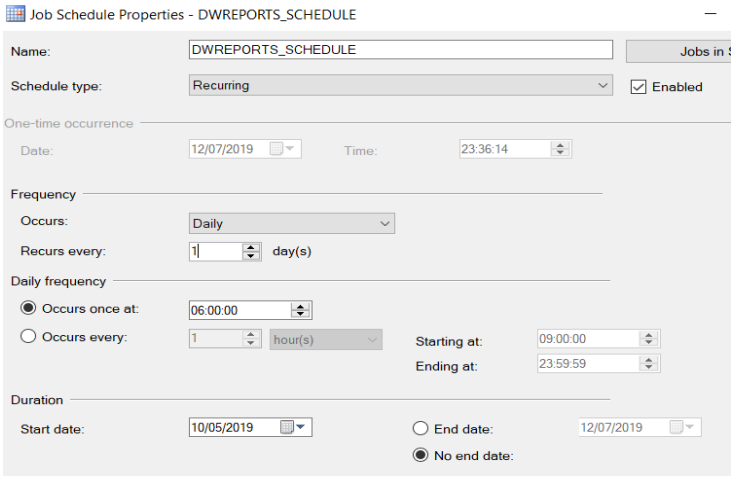
სურათი 7.2

ლაივ სერვერიდან დამუშავებული მონაცემები Data Warehouse-ში მოგვაქვს ყოველდღე დავალებების (job) დახმარებით . ფიზიკურად დავალება წარმოადგენს ბიჯების (steps) ერთობლიობას, რომელიც მიმდევრობით სრულდება. ამასთან, დავალების ნებისმიერი ბიჯის შესრულება შეგვიძლია დამოკიდებული გავხადოთ წინა ბიჯის შესრულების შედეგზე.



სურათი 7.3

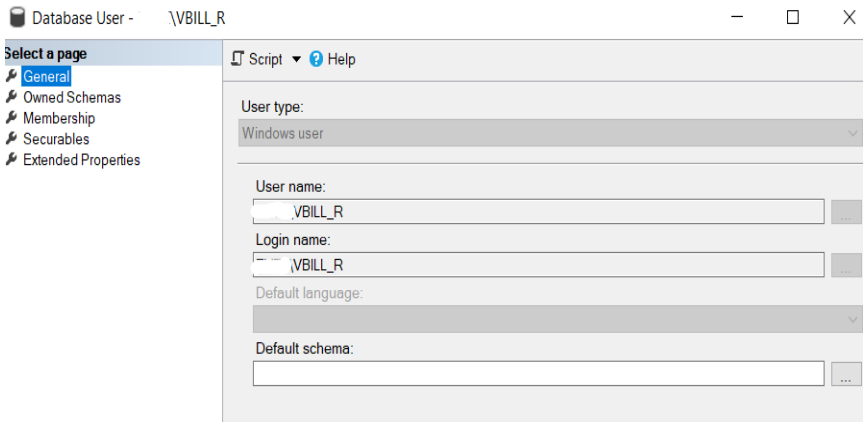
დავალების შესრულების დრო და პერიოდი შეგვიძლია ხელით განვსაზღვროთ. მოცემულ მაგალითზე დავალების დაწყების დრო ექვსი საათია , რადგან ამ დროს ბაზა ნაკლებად დატვირთულია და სამუშაო საათების დაწყებამდე ინფორმაცია განახლებული იქნება.



სურათი 7.4

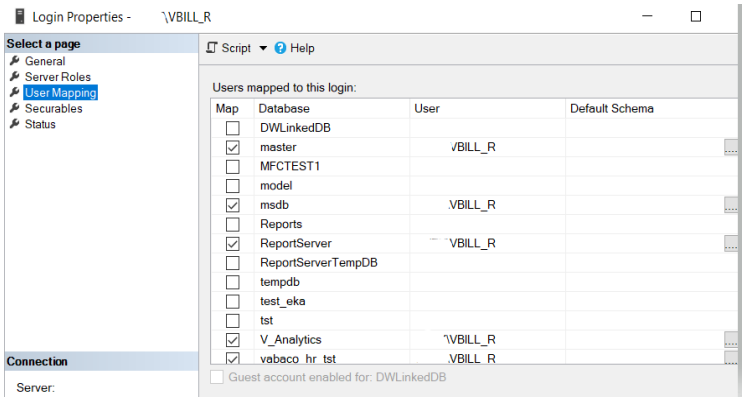
დამუშავებულ მონაცემებს აუცილებლად უნდა დავადოთ ინდექსი . ინდექსები სტრუქტურირებულ მონაცემთა ბაზებში ძებნის დაჩქარების მიზნით გამოიყენება და მონაცემების ძებნის დაჩქარება მათი ფიზიკური ან ლოგიკური მოწესრიგების ხარჯზე მიიღწევა. გასათვალისწინებელია ის ფაქტიც, რომ ინდექსების ზედმეტი რაოდენობით დადებამ, ცხრილის ველებზე, შეიძლება საპირისპირო შედეგამდე მიგვიყვანოს.

მონაცემთა ბაზების ობიექტებზე, მაგალითად ვირტუალურ ცხრილებზე წვდომისათვის შევქმენი ლოგები და იუზერები. სურათ 7.5 -ზე ჩანს , რომ შექმნილია ლოგი დასახელებით VBILL_R. ასევე, Active Directory-ში შევქმენი იგივე დასახელების ჯგუფი და ყოველ ეტაპზე , როდესაც ახალ იუზერს სჭირდება წვდომის გახსნა ბაზის დონეზე ვამატებ ჯგუფში - VBILL_R. სერვერის დონეზე წვდომის გასახსნელად IT ინფრასტრუქტურის გუნდი გვეხმარება.

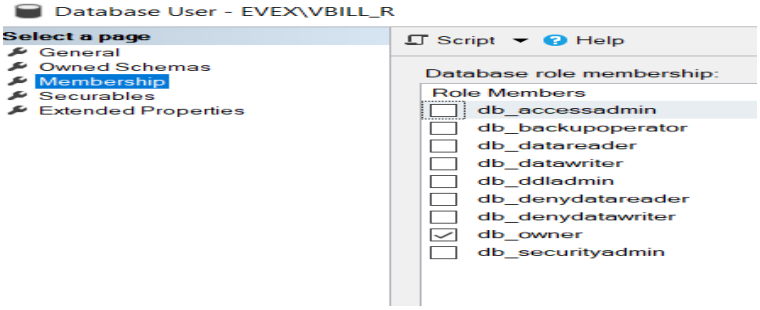


სურათი 7.5

სურათზე 7.5.1 ნაჩვენებია ლოგინს VBILL_R თუ რომელ ბაზასთან აქვს დაკავშირების უფლება. სურათზე 7.5.2 კი ჩანს, რომ იუზერი VBILL_R არის ბაზის მფლობელი, რაც ნიშნავს, რომ შეუძლია ნებისმიერი მოქმედების შესრულება ბაზასთან მუშაობის პროცესში.



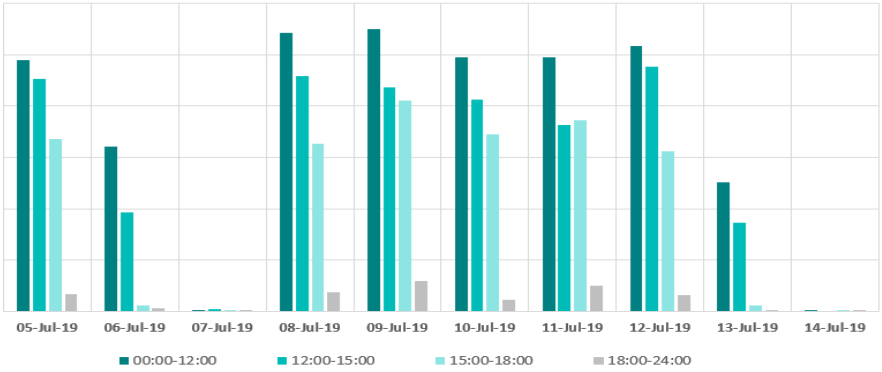
სურათი 7.5.1



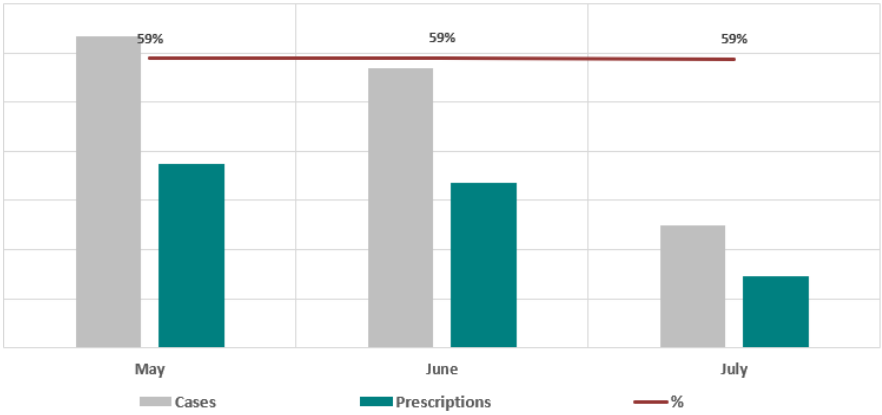
სურათი 7.5.2

დამუშავებული მონაცემების ვიზუალიზაციას ვაჩვენებ ექსელის და Power BI ტექნოლოგიის დახმარებით. ექსელის დახმარებით , რადგან იძლევა რთული გამოთვლების განხორციელების საშუალებას - გამოთვლის პროცესში ერთდროულად შეიძლება იმ მონაცემებით ოპერირება, რომლებიც განლაგებულია ელექტრონული ცხრილის სხვადასხვა ზონებში და ამასთან დაკავშირებულნი არიან ერთმანეთთან გარკვეული დამოკიდებულებით.

სურათი 7.6 გვაჩვენებს გამოწერილი რეცეპტების რაოდენობას საათების ჭრილში, მოცემულია ათის დღის მონაცემები.



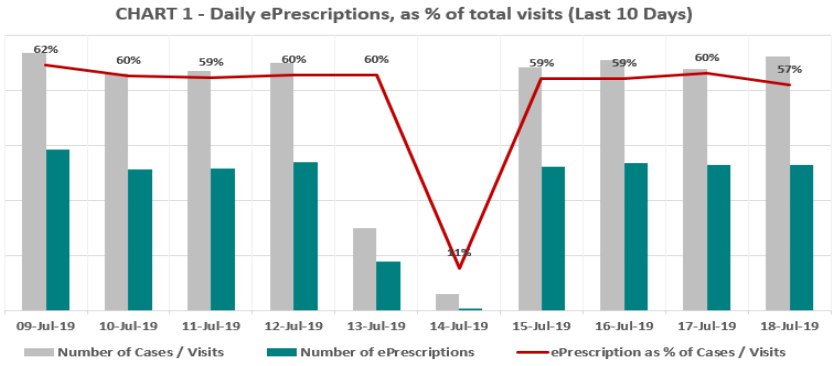
სურათი 7.6



სურათი 7.7

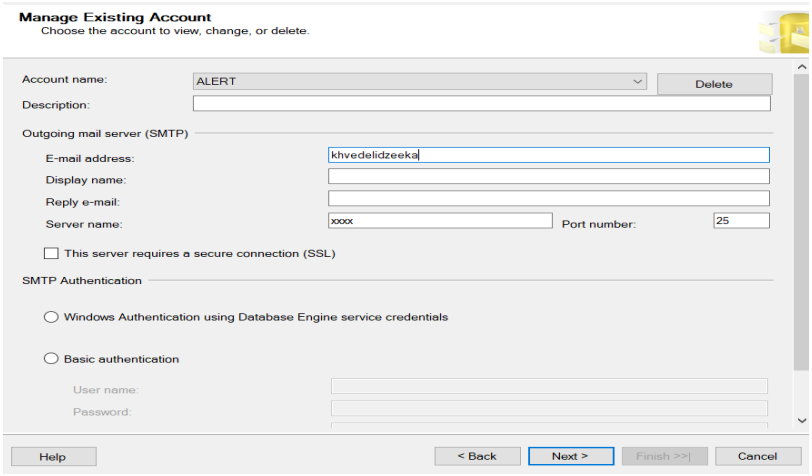
სურათზე7.7 წარმოდგენილია სამი თვის ჭრილში გამოწერილი რეცეპტებისა და დაფიქსირებული შემთხვევების რაოდენობა.

სურათზე 7.7.1 ნაჩვენებია ათი დღის განმავლობაში დაფიქსირებული შემთხვევებისა და გამოწერილი დანიშნულებების ჯამური რაოდენობა გრაფიკის სახით.



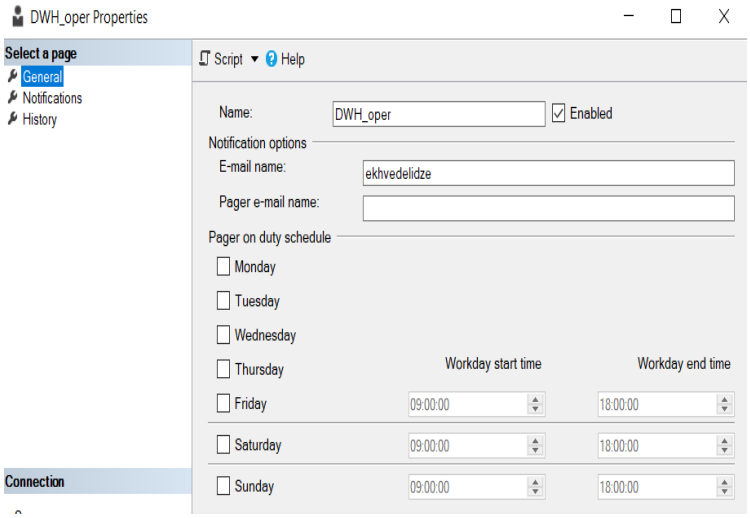
სურათი 7.8

იმისათვის, რომ ჩემს შექმნილ დავალებებზე დღის ნებისმიერ მონაკვეთში მქონდეს ინფორმაცია დავაკონფიგურირე მეილი, რომელიც ტექსტური შეტყობინების გაგზავნის საშუალებას იძლევა SQL Server-დან იუზერთან. ტექსტური შეტყობინება შეიძლება მოიცავდეს ინფორმაციას კოდის შესრულების შედეგის ან უბრალოდ გაფრთხილებას მონაცემთა ბაზაში გაჩენილი ცვლილების შესახებ. კონფიგურაციის პროცესი მოიცას სამ ეტაპს : Mail Account -სა და Mail Profile - ის შექმნას, შემდეგ მათ კონფიგურაციას. მეილის კონფიგურაცია შესაძლებელია როგორც T-SQL - ის დახმარებით ასევე, SQL-ის ინტერფეისიდან. მეილის კონფიგურაციაზე msdb სისტემური მონაცემთა ბაზაა პასუხისმგებელი.



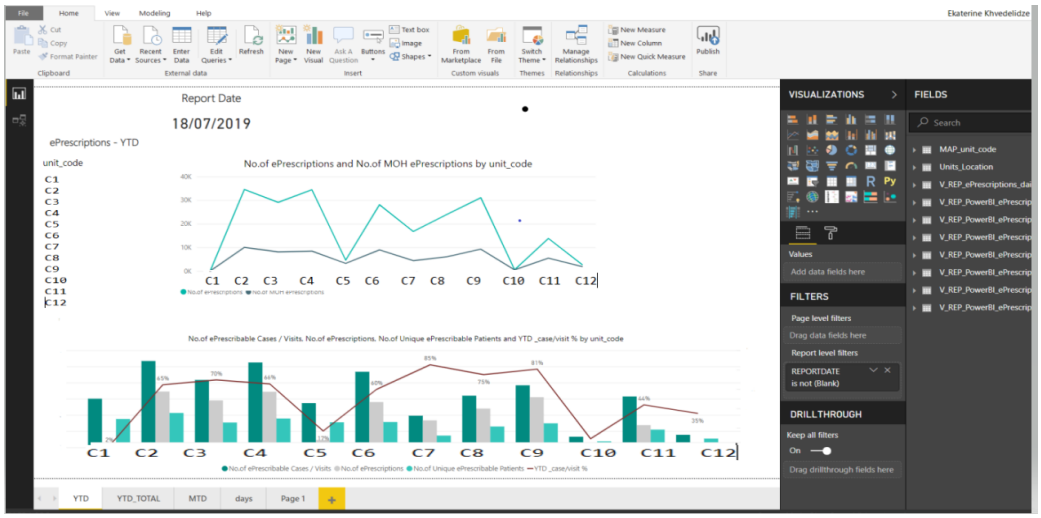
სურათი 7.9

მეილზე ტექსტური შეტყობინების მოსვლას უზრუნველყოფს ოპერატორი, რომლის შექმნისას ვუთითებ კონკრეტულ მეილს, რომელზეც შემდეგში ვიღებ შეტყობინებას დავალების შესრულების შესახებ.



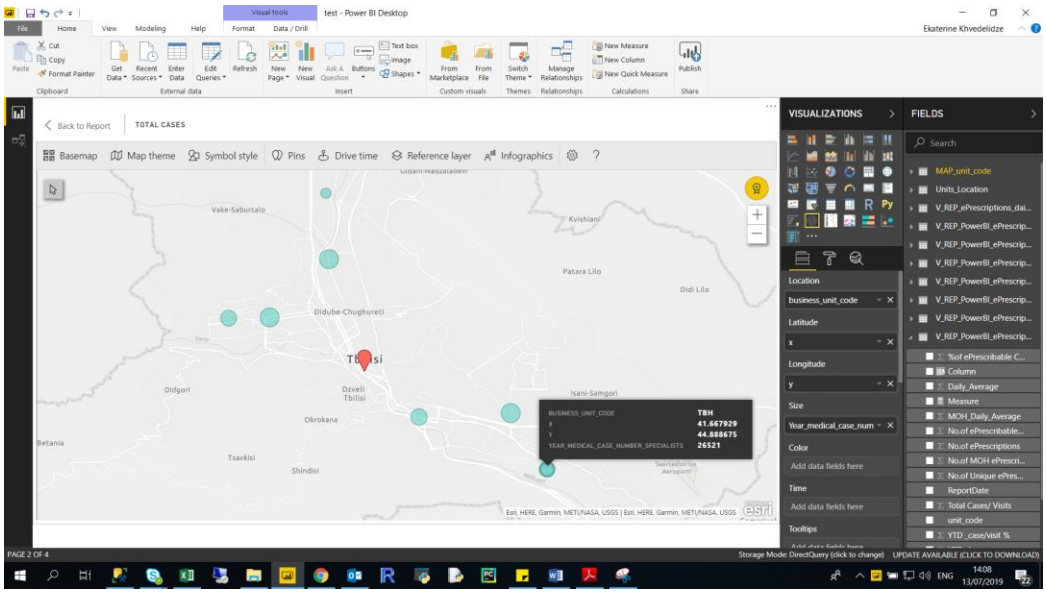
სურათი 7.10

თანამედროვე სამყაროში ტექნოლოგიების სწრაფმა განვითარებამ გამოიწვია ადამიანური რესურსის პროცესებში ნაკლები ჩართულობა და სამუშაოს დროის აჩქარება. კომპანიებში აუცილებელი გახდა ახალი სისტემების დანერგვა და ოპტიმალური პროცესების შექმნა . ბიზნეს-ანალიტიკისთვის მიზანშეწონილია Microsoft Power BI ტექნოლოგიების გამოყენება, რომელიც მონაცემების ანალიზისა და ვიზუალიზაციის საშუალებას იძლევა. Power BI-საშუალებას გვაძლევს მარტივად დავუკავშირდეთ მონაცემთა წყაროს, გამოავავლინოთ და მოვახდინოთ მნიშვნელოვანი ასპექტების ვიზუალიზაცია, ვუზრუნველყოთ მარტივი ხელმისაწვდომობა კორპორაციის ყველა თანამშრომლისათვის. სურათი 7.11-ზე შექმნილია ვიზუალიზაცია, რომელიც საშუალებას აძლევს სამედიცინო კორპორაციის მენეჯმენტს თვალყური ადევნონ თითოეული კლინიკის ჭრილში პაციენტების რაოდენობას, ექიმებთან დაფიქსირებულ შემთხვევებს , გამოწერილი დანიშნულებების რაოდენობას.



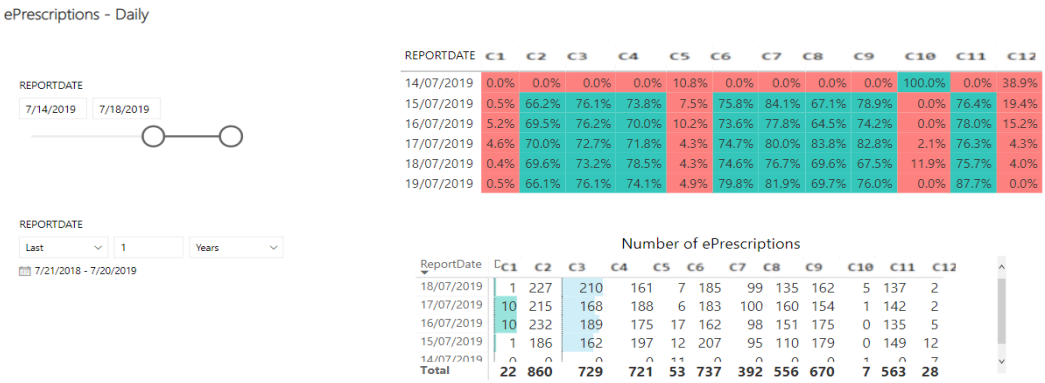
სურათი 7.11

სურათზე 7.12 წარმოდგენილია რუკა , რომელზეც მწვანე ფერითაა აღნიშნული კლინიკები და ფერის მოცულობა რეგულირდება შემთხვევების რაოდენობის მიხედვით.



სურათი 7.12

Power BI-ის დახმარებით აღნიშნული მონაცემები შეგვიძლია ვაჩვენოთ დღის ჭრილშიც .

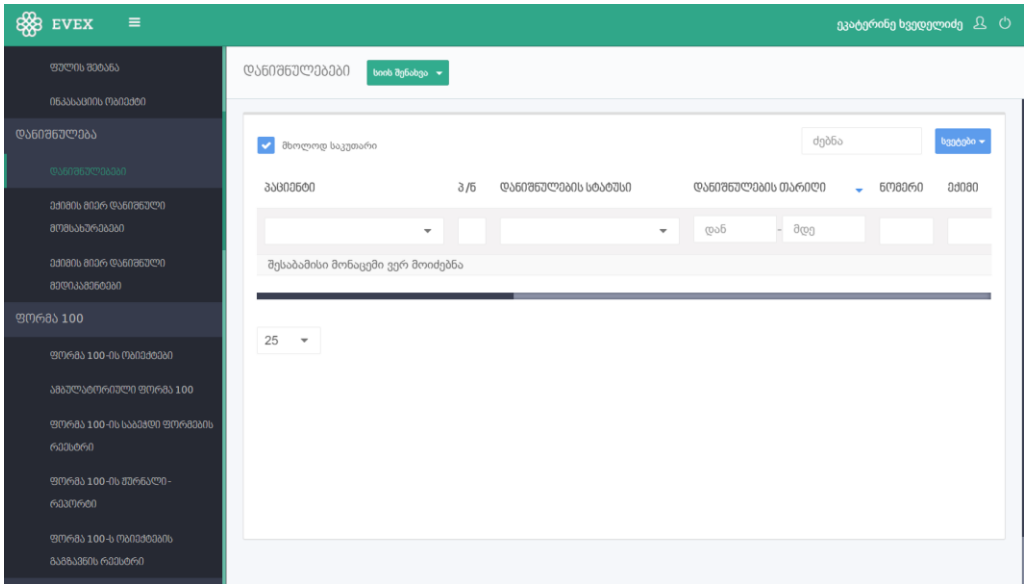


სურათი 7.13

აღსანიშნავია, რომ Power BI Mobile აპლიკაცია მუშაობს ყველა მოწყობილობაზე, რომელიც თავსებადია Windows10, iOS (iPad, iPhone, iPod Touch , Apple Watch) ოპერაციულ სისტემებთან.

ექიმების მხრიდან აღნიშნული ინფორმაციის შეყვანა ხდება ბილინგის სიტემაში .

სურათი 7.14



სურათი 7.15

დასკვნა

ნაშრომში განხილულია მონაცემთა დამუშავების გზები, მონაცემთა ანალიზის შესაძლებლობა სისტემური ცხრილების დახმარებით. ბიზნესპროცესების ოპტიმიზაციის ძირითადი საკითხები, მათი ტიპები, მოდელირება და მართვა. ინფორმაციის მთლიანობის შესანარჩუნებლად განვიხილეთ სარეზერვო ასლების შექმნის მეთოდები. მონაცემთა ბაზები განსაკუთრებული სახის ორგანიზებული ფაილებისა და ჩანაწერების ნაკრებია. მონაცემთა ბაზის მართვის სისტემა საშუალებას გვაძლევს სრულად ვაკონტროლოთ მონაცემთა განსაზღვრის, მათი გადამუშავებისა და ერთობლივი გამოყენების პროცესი. იგი გვიადვილებს დიდი მოცულობის ინფორმაციის შეკრებასა და გადამუშავებას. განვიხილეთ Microsoft Power BI ტექნოლოგიების პრაქტიკული გამოყენება, რომელიც მონაცემების ანალიზისა და ვიზუალიზაციის საშუალებას იძლევა.

სტრუქტურირებული მონაცემების საჭიროება და მათი დამუშავება როგორც ვხედავთ დღევანდელ რეალობაში ყოველ წუთს და წამს უფრო მოთხოვნადი და აუცილებელი ხდება . შეგვიძლია ამ ინფორმაციაზე დაყრდნობით ვთქვათ , რომ მომავალში კიდევ უფრო ადვილი იქნება მონაცემების დამუშავება , ნაკლები დროითა და ძალისხმევით შევძლებთ დასმული ამოცანის გადაწყვეტას. ავტომატიზირებული დავალებებისა და მეილის საშუალებით მუდმივად შეგვეძლება მონაცემების განახლება და ინფორმაციის მიღება მონაცემთა ბაზაში გაჩენილი ცვლილების შესახებ.

ბიბლიოგრაფია

[1] https://www.sas.com/en_sa/insights/analytics/big-data-analytics.html

[2]https://www.sqlshack.com/understanding-sql-server-query-plan-cache/

[3]20462D-ENU-TrainerHandbook

[4]Administering Microsoft SQL Server 2012 Databases

[5]https://www.vembu.com/blog

[6]http://www.allitebooks.org/beginning-backup-and-restore-for-sql-server/