

MS SQL Server-ის საინსტალაციოს გადმონერა

1

Try SQL Server on-premises or in the cloud

SQL Server on Azure
Run SQL Server on Azure SQL with built-in security and manageability.
[Get started](#)

SQL Server at the edge
Extend SQL to IoT devices for real-time analysis with Azure SQL Edge.
[Get started](#)

SQL Server on-premises
Build intelligent, mission-critical applications with a scalable, hybrid data platform.
[Free trial](#)

Or, download a free specialized edition

Developer
SQL Server 2019 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.
[Download now >](#)

Express
SQL Server 2019 Express is a free edition of SQL Server, ideal for development and production for desktop, web, and small server applications.
[Download now >](#)

დღევანდელ დღეს საუკეთესო ვარიანტია MS SQL Server 19 Developer Edition, რომელიც არის ყველასათვის ხელმისაწვდომი მონაცემთა ბაზების წარმოების მძლავრი საშუალება და სრულიად უფასო ლიცენზით - ერთადერთი შეზღუდვა არის ლიცენზის შესყიდვის გარეშე არ უნდა გამოიყენოთ კომერციული საქმიანობისათვის. ამ ვერსიის ჩამოსაწერად შეგვიძლია გამოვიყენოთ ლინკი:

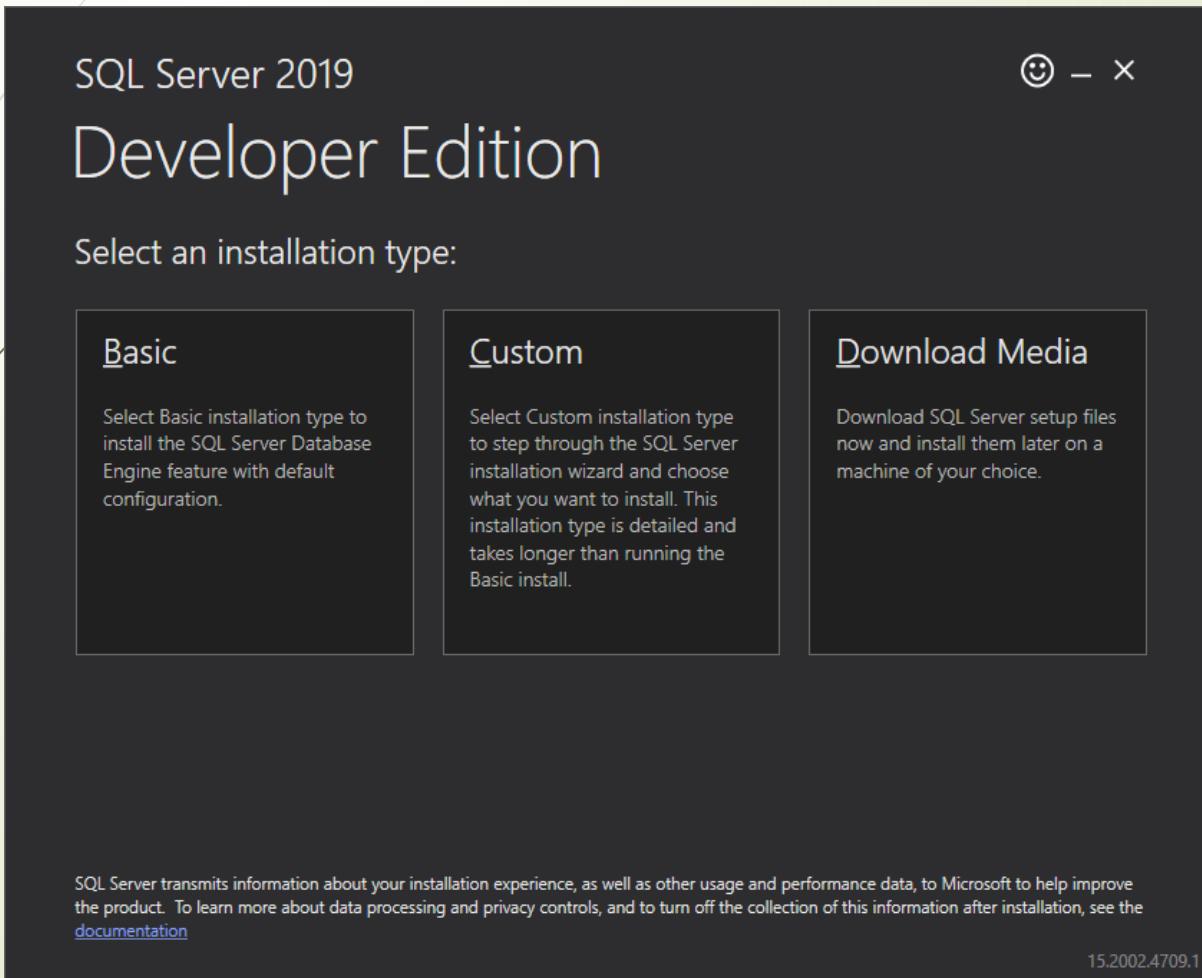
<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

და მაუსის მარცხენა ღილაკით გავააქტიუროთ „Download now“

MS SQL Server-ის საინსტალაციოს გადმონერა

2

გააქტიურების შემდგომ ჩამოიტვირთება ფაილი „SQL2019-SSEI-Dev.exe“, რომელიც უნდა გაეშვას სისტემაში ადმინისტრატორის უფლებებით.



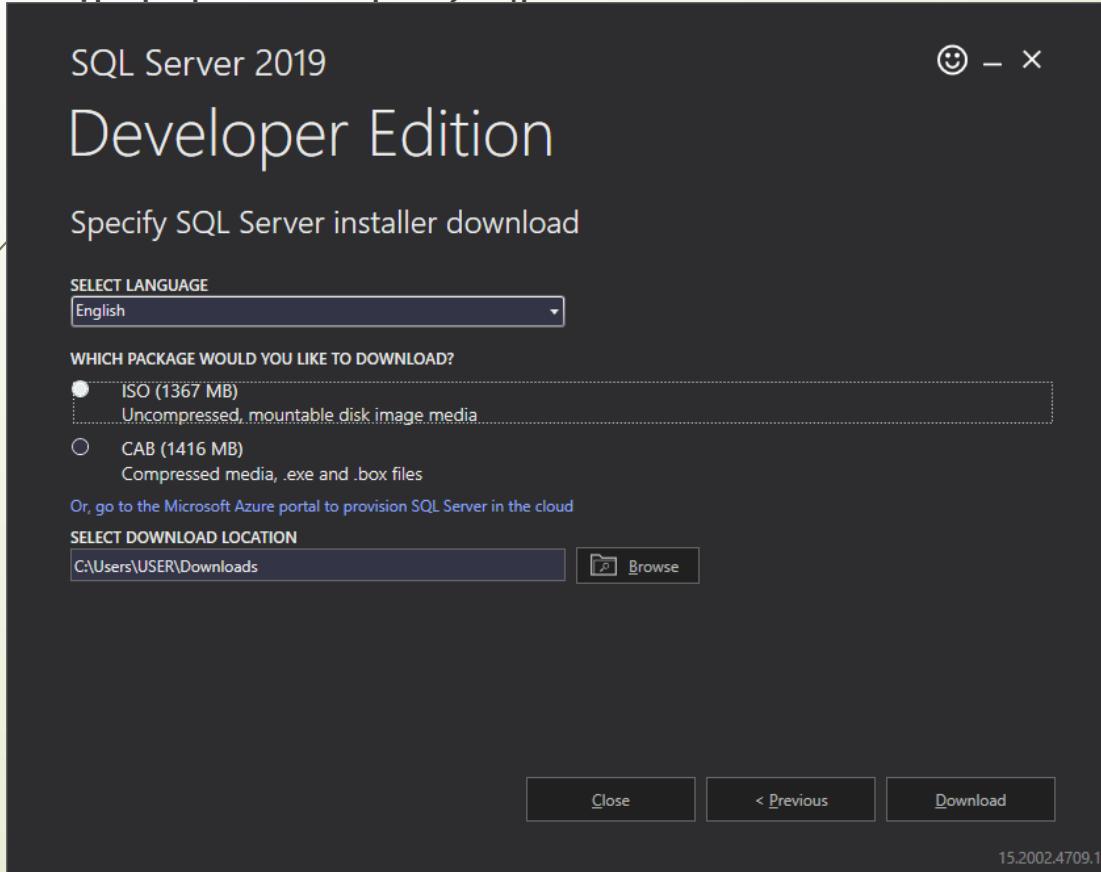
ფაილის გაშვების შემდეგ
მონიტორის ეკრანზე
გამოისახება ინსტალირების
ტიპის შერჩევის ფანჯარა სამი
ელემენტით:

- **Basic** - ჩამოიტვირთება და დაინსტალირდება SQL Server-ის მინიმალური კონფიგურაცია;
- **Custom** - ჩამოტვირთვის შემდგომ ინსტალირების ოსტატი დაგეხმარებათ კომპონენტების შერჩევაში და შემდგომ დაინსტალირდება;
- **Download Media** - ჩამოიტვირთება სრული საინსტალაციო პაკეტი

MS SQL Server-ის საინსტალაციოს გადმოწერა

3

ძირითადად ჭობია შერჩეულ იქნეს მესამე ტიპი, რადგან ჩამოტვირთვის შემდგომ მოგეცემათ შესაძლებლობა SQL Server-ი დააინსტალიროთ თქვენთვის სასურველ დროს (ასევე მომავალში ინსტალირებული სისტემის აღდგენისას). მესამე ტიპის შერჩევის შემდეგ მომხმარებელს ეძლევა შესაძლებლობა მონიტორის ეკრანზე



შეარჩიოს ჩამოტვირთვის
პარამეტრები:

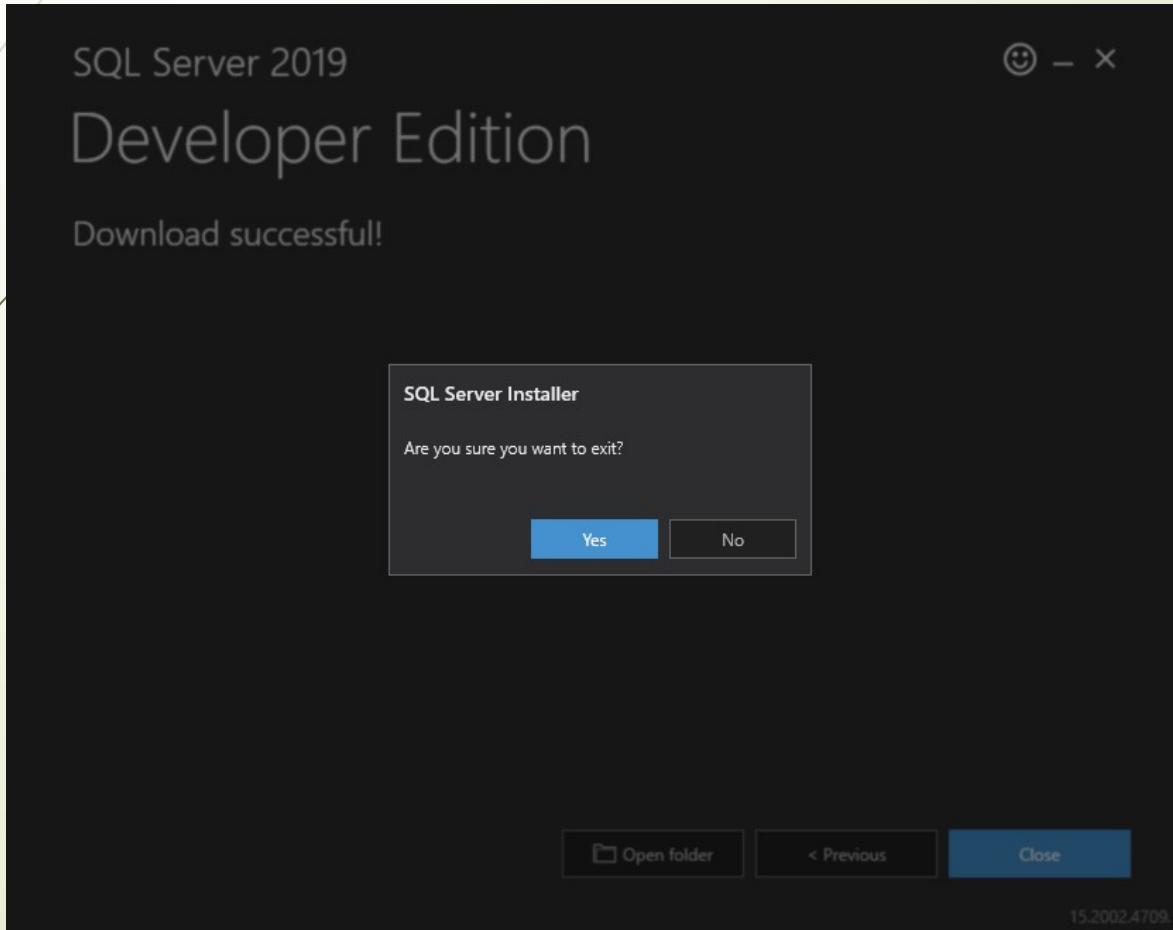
- SELECT LANGUAGE - SQL Server-ის სიტემის სამუშაო ენა;
- WHICH PACKAGE WOULD YOU LIKE TO DOWNLOAD? - ჩამოტვირთული ფაილების ტიპები (ავტორის რჩევაა - ISO);
- SELECT DOWNLOAD LOCATION - ჩამოსატვირთი ადგილმდებარეობის მითითება.

პარამეტრების შერჩევის შემდგომ უნდა გააქტიურდეს ღილაკი "Download" რის შემთხვევაში

MS SQL Server-ის საინსტალაციოს გადმონერა

4

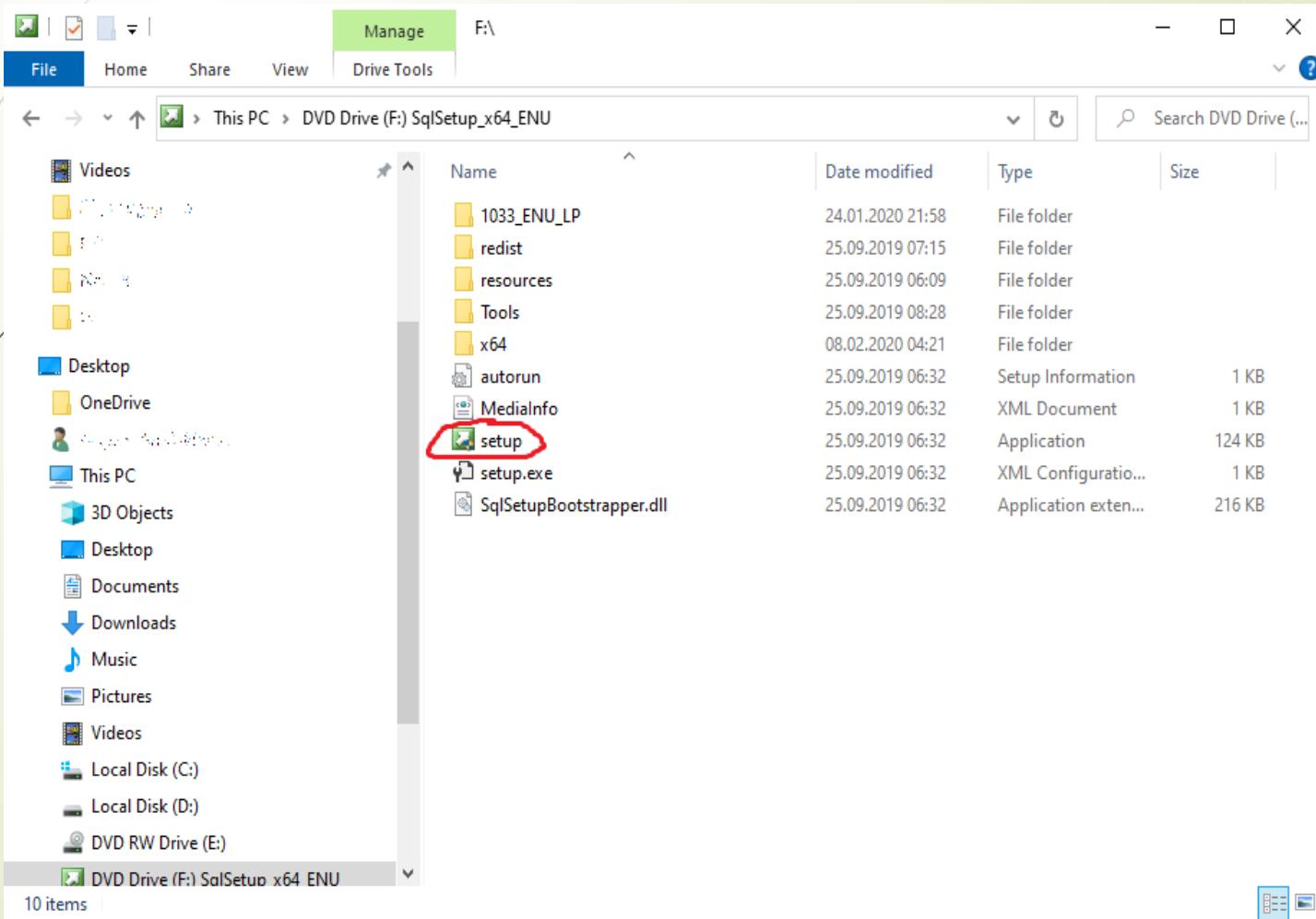
ჩამოტვირთვის დასრულების შემდგომ მონიტორის ეკრანზე გამოისახება დიალოგური ფანჯარა, რომელშიც მომხმარებელს ეძლევა შესაძლებლობა მიუთითოს ინსტალირების დაწყება ან გამოსვლა და დასრულება.



MS SQL Server-ის ინსტალირების დაგეგმვა

5

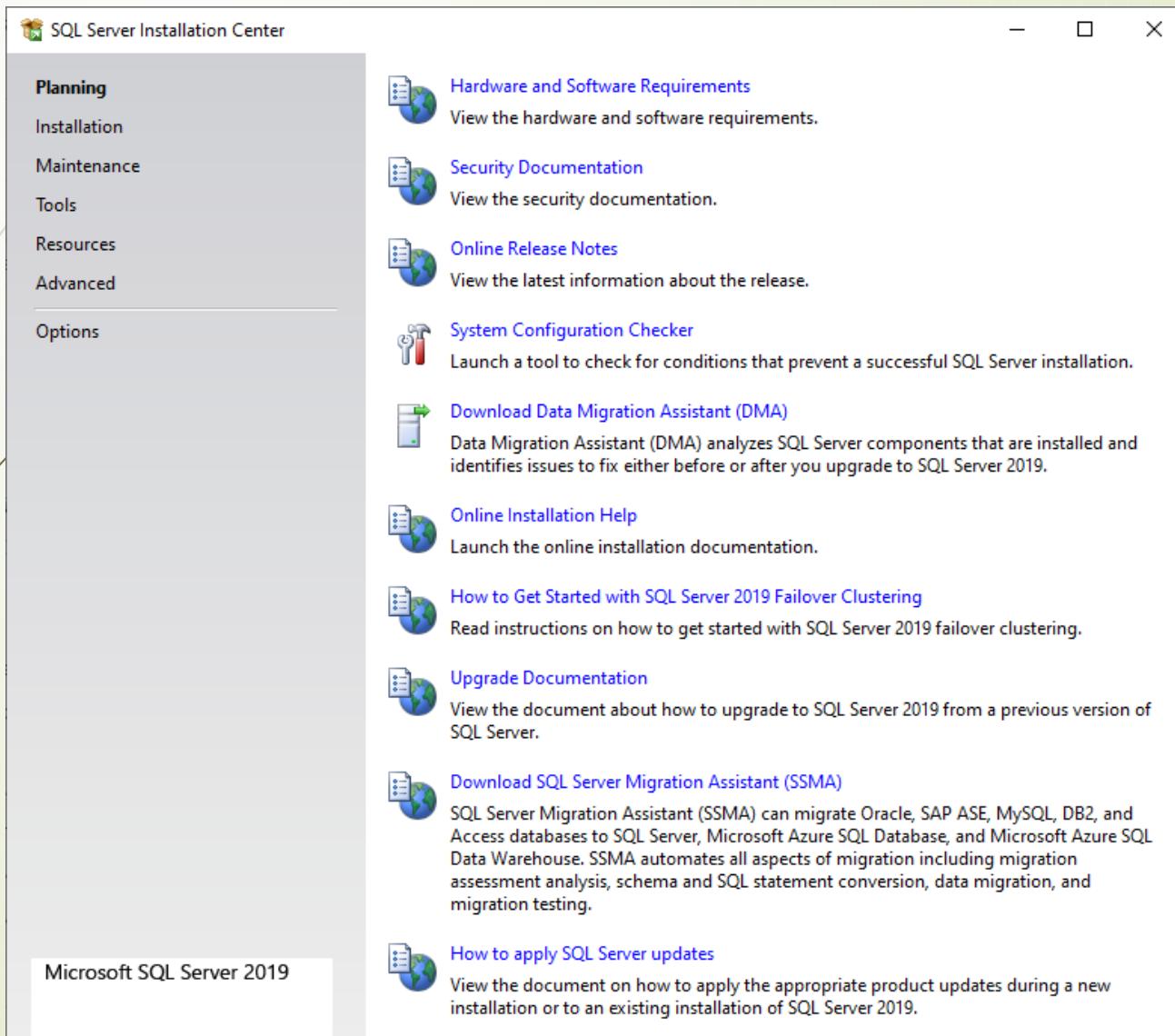
ინსტალირების დაწყებისათვის მომხმარებელმა უნდა მოიძიოს ჩამოტვირთული ფაილი (იმ საქაღალდეში, რომელიც მიუთითა ჩამოტვირთვისას) და გააკტიუროს



„SQLServer2019-x64-ENU-Dev.iso“. მონიტორის ეკრანის ფანჯარაში გამოისახება მისი შემცველობა, რომელშიც უნდა გააქტიურდეს setup.exe აპლიკაცია (აქაც მომხმარებელს უნდა გააჩნდეს საოპერაციო გარემოში ადმინისტრატორის უფლებები).

MS SQL Server-ის ინსტალირების დაგეგმვა

6



SQL Server-ის
ინსტალირების დაწყების წინ
უნდა განხორციელდეს ამ
ინსტალირების პროცესის
დაგეგმვა.

მონიტორის ეკრანზე ახლად
გამოსახულ ფანჯარაში
უპირველესყოვლისა ზედა-
მარცხენა მიღამოში
განთავსებულ სიაში უნდა
იქნეს შერჩეული პირველი
სტრიქონი Planning, რის
შემდეგაც მარჯვენა
მიღამოში გამოისახება
ინსტალირების დაგეგმვის
საშუალებები,
დოკუმენტაცია, სისტემის
შემოწმება მზადყოფნაზე,
მონაცემთა მიგრაციის
საშუალებები და სხვა.

MS SQL Server-ის მინიმალური მოთხოვნები

7

SQL Server-ის Windows-ში ინსტალირების დაგეგმვისას უნდა იქნეს გათვალისწინებული მინიმალური მოთხოვნები, რომლებიც უნდა დააკმაყოფილოს როგორც კომპიუტერმა, ასევე თვით ოპერაციულმა გარემომ.

SQL Server-ის ინსტალირებისათვის და მუშაობისათვის საჭირო კომპიუტერის მინიმალური რესურსები:

- მონიტორი - SuperVGA (800x600 პიქსელი);
- პროცესორი - აუცილებელი პირობა x64. ტაქტური სიხშირე 1,4 გგპ (რეკომენდირებულია 2,0 გგპ);
- ოპერატიული მეხსიერება - 512 მბ Express ვერსიისათვის, ხოლო დანარჩენისათვის 1 გბ (რეკომენდირებულია 1 გბ Express ვერსიისათვის, ხოლო დანარჩენისათვის 4 გბ);
- მყარი დისკი - 6 გბ თავისუფალი სივრცე (აქვე უნდა ითქვას, რომ მყარი დისკი უნდა იყოს სექტორის ზომით 512 ბ ან 4 კბ);
- ინტერნეტი - ფუნქციონალების უზრუნველსაყოფად.

MS SQL Server-ის მინიმალური მოთხოვნები

8

SQL Server-ი მხარს უჭერს შემდეგ ქსელურ ოქმებს:

- Shared memory;
- Named Pipes;
- TCP/IP.

SQL Server-ი შეიძლება დაინსტალირდეს ბევრ ნაციონალურ ენაზე, მაგრამ მათ შორის ქართული ჯერ არ არის.

SQL Server-ის ინსტალირებისათვის და მუშაობისათვის მინიმალური ოპერაციული გარემო:

- Windows 10 TH1 1507;
- Windows Server 2016.

უსაფრთხოების მოსაზრებიდან გამომდინარე, არ არის რეკომენდირებული SQL Server-ის ინსტალირება დომენის კონტროლერზე, თუმცა გამოუვალ მდგომარეობაში ინსტალირება შესაძლებელია, მაგრამ შემდეგი შეზღუდვებით:

- საალრიცხვო ჩანაწერის ლოკალურ სერვისებში SQL Server-ის გაშვება ვერ მოხერხდება;

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მინიმალური მოთხოვნები

9

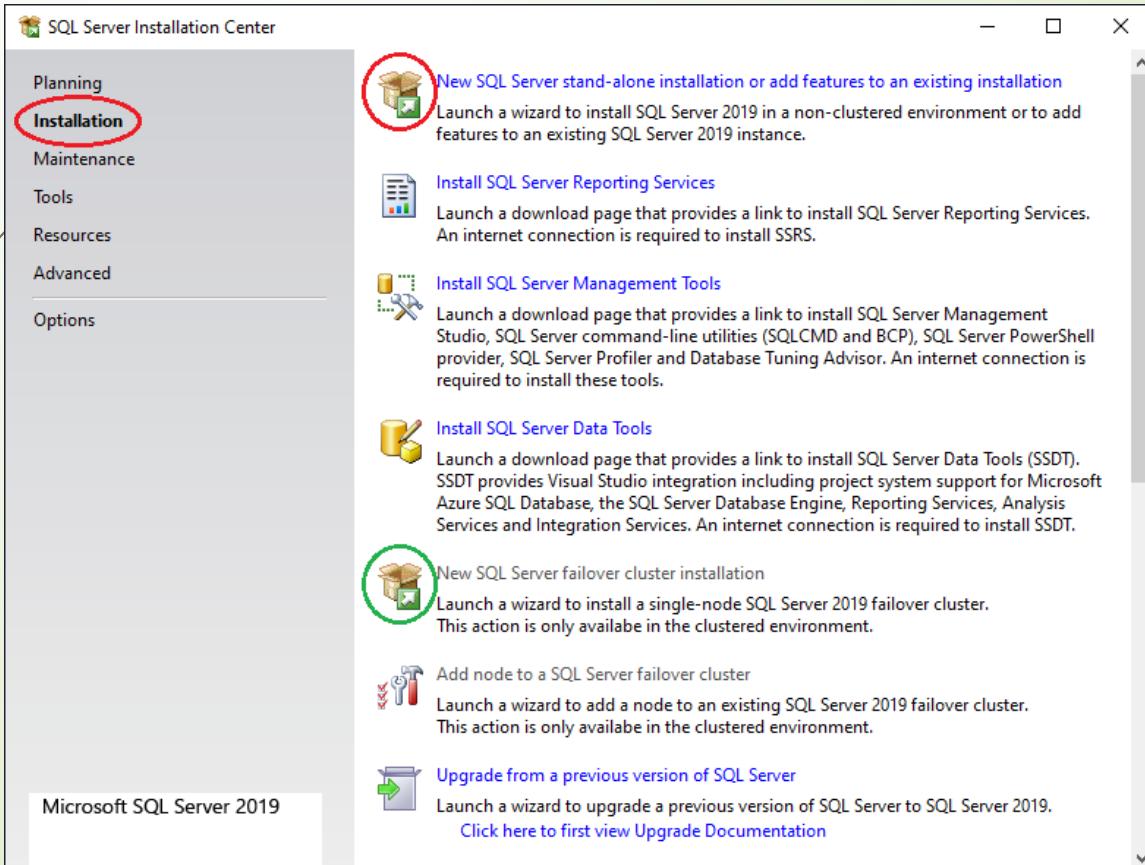
- SQL Server-ის ინსტალირების შემდეგ, დომენის წევრი კომპიუტერი ვერ გახდება დომენის კონტროლერი (ამისათვის უნდა განხორციელდეს SQL Server-ის დეინსტალირება);
- SQL Server-ის ინსტალირების შემდეგ, დომენის კონტროლერი კომპიუტერი ვერ გახდება დომენის წევრი (ამისათვის უნდა განხორციელდეს SQL Server-ის დეინსტალირება);
- SQL Server-ი მხარს არ უჭერს მტყუნება მედეგ კლასტერულ ეგზემპლიარებს, რომელშიც კლასტერის კვანძები არიან დომენის კონტროლერები;
- SQL Server-ი დომენის კონტროლერზე მხოლოდ წაკითხვის რეჟიმში ინსტალირებისას, ვერ შექმნის უსაფრთხოების ჯგუფებს, ვერ მოამზადებს სააღრიცხვო ჩანაწერებს და სხვ.

SQL Server-ის ინსტალირება ასევე შესაძლებელია Windows Server 2016 Core და Windows Server 2019 Core-ს ვერსიებშიც (თუმცა ამისათვის მას სჭირდება „.NET Framework 4.6.1“, რომელსაც SQL Server-ი თვითონ დააინსტალირებს გადატვირთვის მოთხოვნით).

ამ ოპერაციულ გარემოებებზე ინსტალირებისას გამოიყენება ავტომატური ინსტალირება (ჩუმი რეჟიმი, ანუ დიალოგური ფანჯრების გარეშე), თუმცა ლიცენზიის პირობებზე თანხმობა მაინს საჭიროა.

MS SQL Server-ის ინსტალირება Windows-ში

SQL Server-ის ინსტალირებისათვის მონიტორის ეკრანზე გამოსახულ დიალოგურ ფანჯარაში ზედა-მარცხენა მიღამოში განთავსებულ სიაში უნდა იქნეს შერჩეული მეორე სტრიქონი Installation, რის შემდგომაც SQL Server-ის ტიპის მიხედვით უნდა შეირჩეს ფანჯრის მარჯვენა მიღამოში გამოსახული სიის მიხედვით. ძირითადებია:

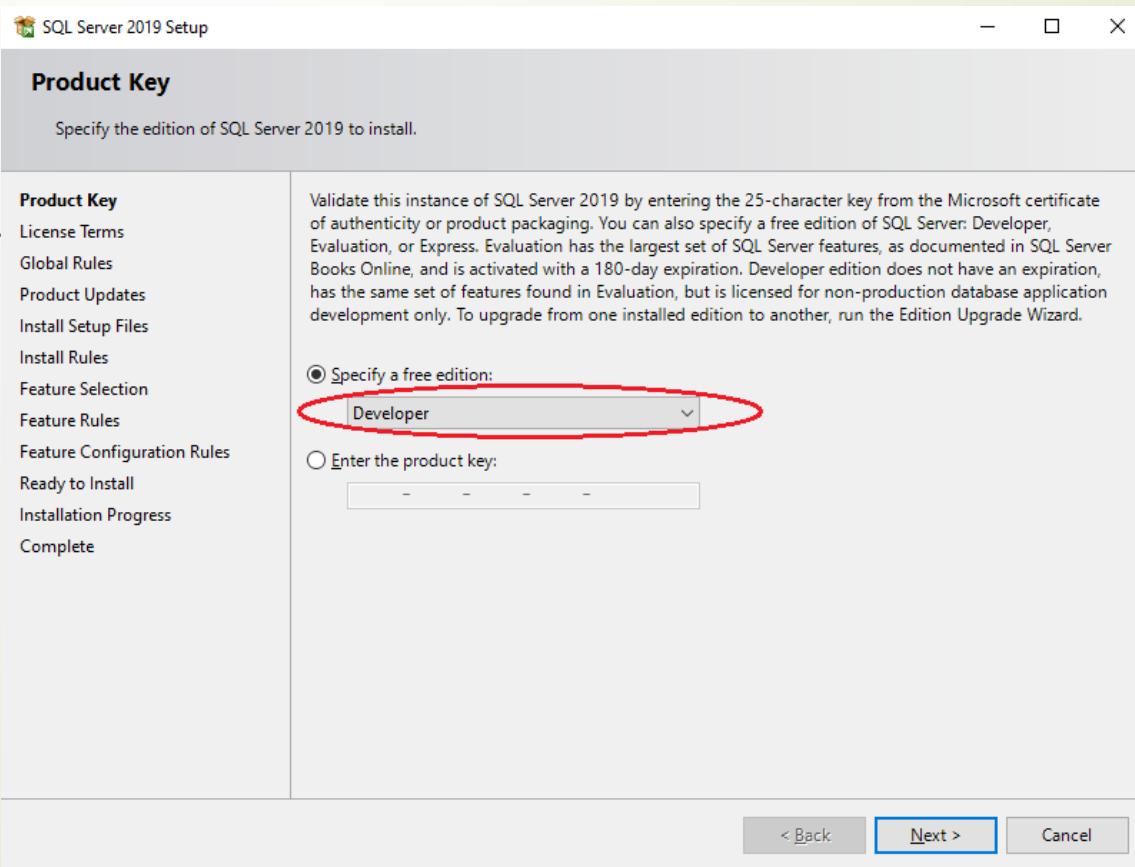


- თუ ინსტალირება ხორციელდება ერთ კომპიუტერზე/სერვერზე, მაშინ უნდა იქნეს შერჩეული პირველი სტრიქონი New SQL Server stand-alone installation or add features to an existing installation;
- თუ ინსტალირება ხორციელდება კლასტერულად (რამდენიმე სერვერზე), ამ შემთხვევაში უნდა იქნეს შერჩეული მეორე სტრიქონი New SQL Server cluster installation.

MS SQL Server-ის ინსტალირება Windows-ში

11

განვიხილოთ ინსტალირება ერთ კომპიუტერზე/სერვერზე, რის შერჩევის შემდეგ დიალოგურ ფანჯარაში მომხმარებელმა უნდა შეიყვანოს (არსებობის შემთხვევაში) პროდუქტის ლიცენზირების გასაღები, ან შეარჩიოს უფასო ვერსია, რომელიც არის სამის ტიპის (უნდა იქნეს შერჩეული - Developer) და გააქტიუროს ღილაკი Next:

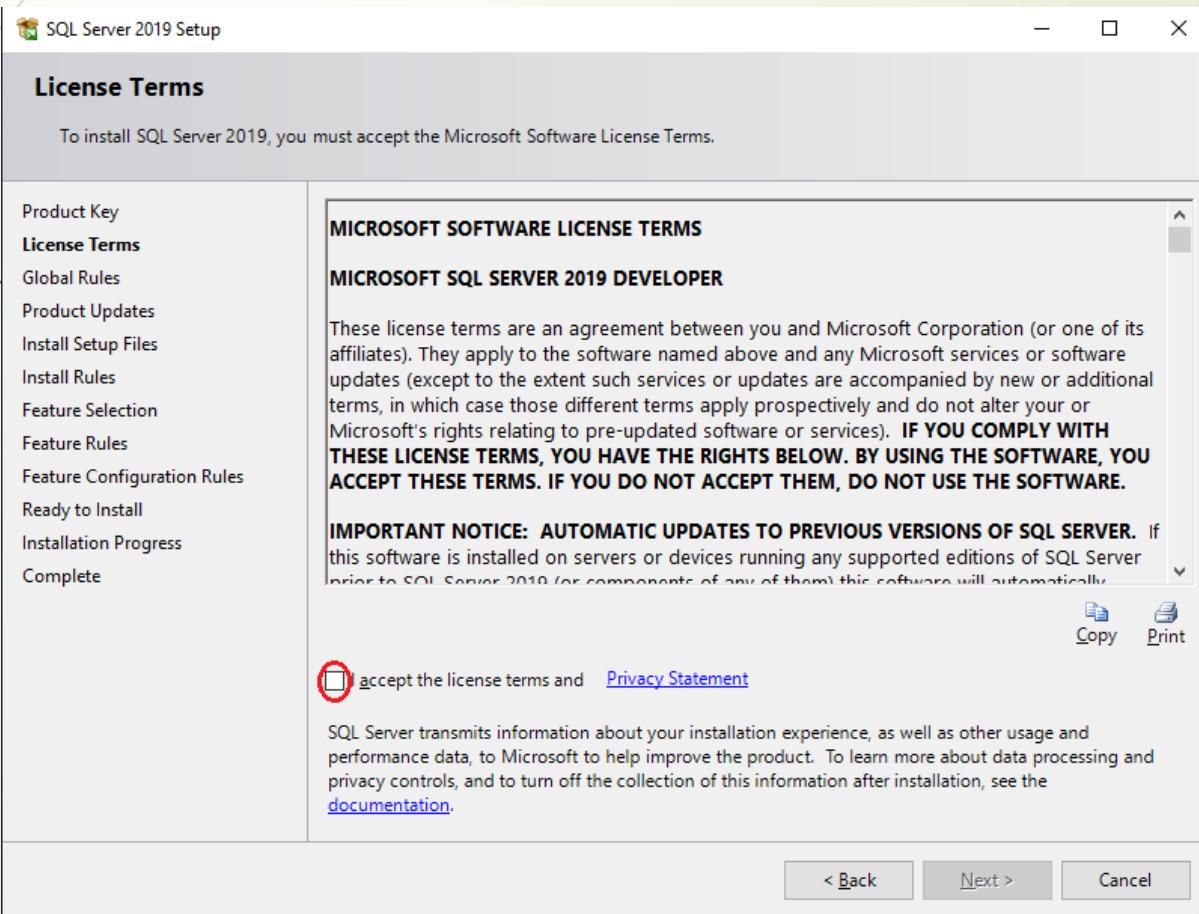


- Evaluation - უფასო ვერსია არის მხოლოდ 180-დღიანი, მაგრამ კომპლექტაციაში მოყვება დოკუმენტაცია, რომელიც მომხმარებელს ეხმარება MS SQL Server-ის ათვისებას;
- Developer - მაქსიმალურად სრული უფასო ვერსიაა, რომლის კომპონენტების გამოსაღევი იქნება დეველოპერებისთვის;
- Express - შეზღუდული უფასო ვერსიაა, რომელშიც შესაძლებელია ელემენტარული ფუნქციების განხორციელება.

MS SQL Server-ის ინსტალირება Windows-ში

12

შემდგომ დიალოგურ ფანჯარაში მომხმარებელი (გულმოდგინედ წაკითხვისა და გაანალიზების შემდგომ) უნდა დათანხმდეს Microsoft-ის სალიცენზიო პირობებს MICROSOFT SQL SERVER 2019 DEVELOPER პროდუქტან მიმართებაში.



მომხმარებელს სურვილისებრ აქვს შესაძლებლობა 8ემოაღნიშნული პირობები იქვე ქვემო-მარჯვენა ღილაკების გამოყენებით დააკოპიროს ან ამობეჭდოს საბეჭდ მოწყობილობაზე.

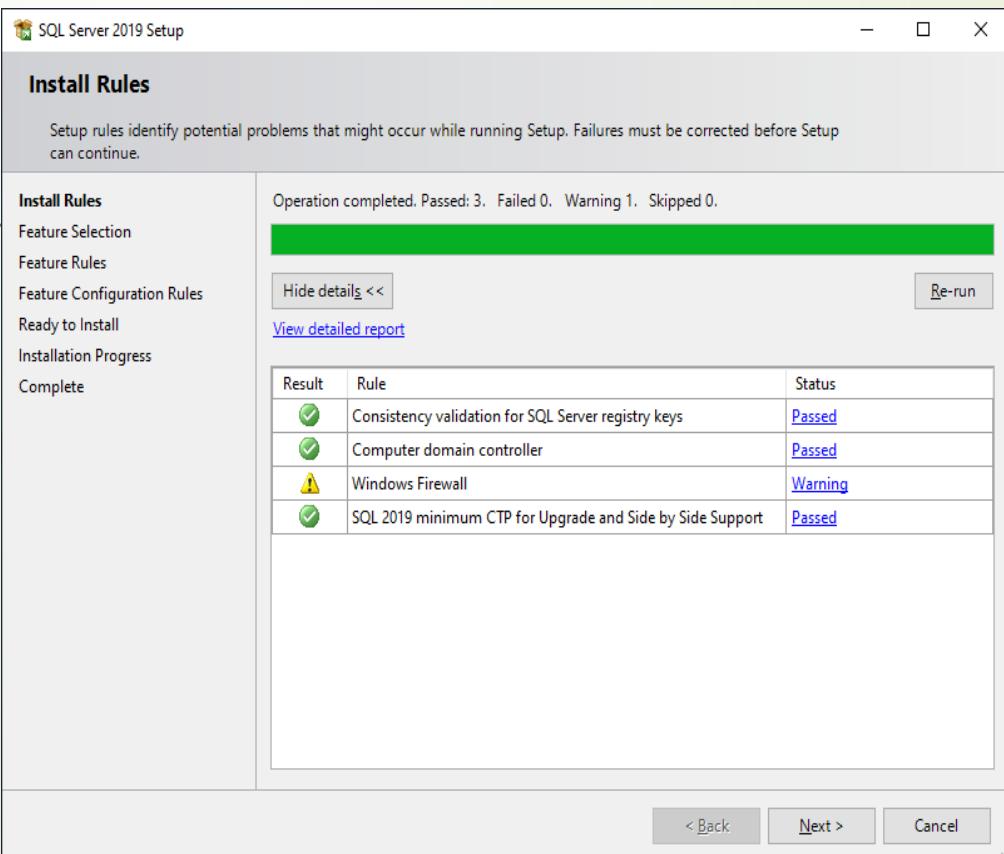
სალიცენზიო პირობებთან დათანხმებისათვის მომხმარებელმა უნდა მონიშნოს დათანხმების ოპცია და გაააქტიუროს ღილაკი Next.

აქვე უნდა აღვნიშნოთ, რომ დათანხმების გარეშე Next ღილაკი არ გააქტიურდება.

MS SQL Server-ის ინსტალირება Windows-ში

13

შემდგომ დიალოგურ ფანჯარაში გამოისახება ინფორმაცია ინსტალირების პრობლემების შესახებ. იმისდა მიხედვით თუ როგორი შედეგები გამოისახა ფანჯრის შუა მიღამოში მდებარე ცხრილის პირველ სვეტში (Result), შეიძლება მსჯელობა ინსტალირების გაგრძელების შესახებ Next ღილაკის გააქტიურებით:

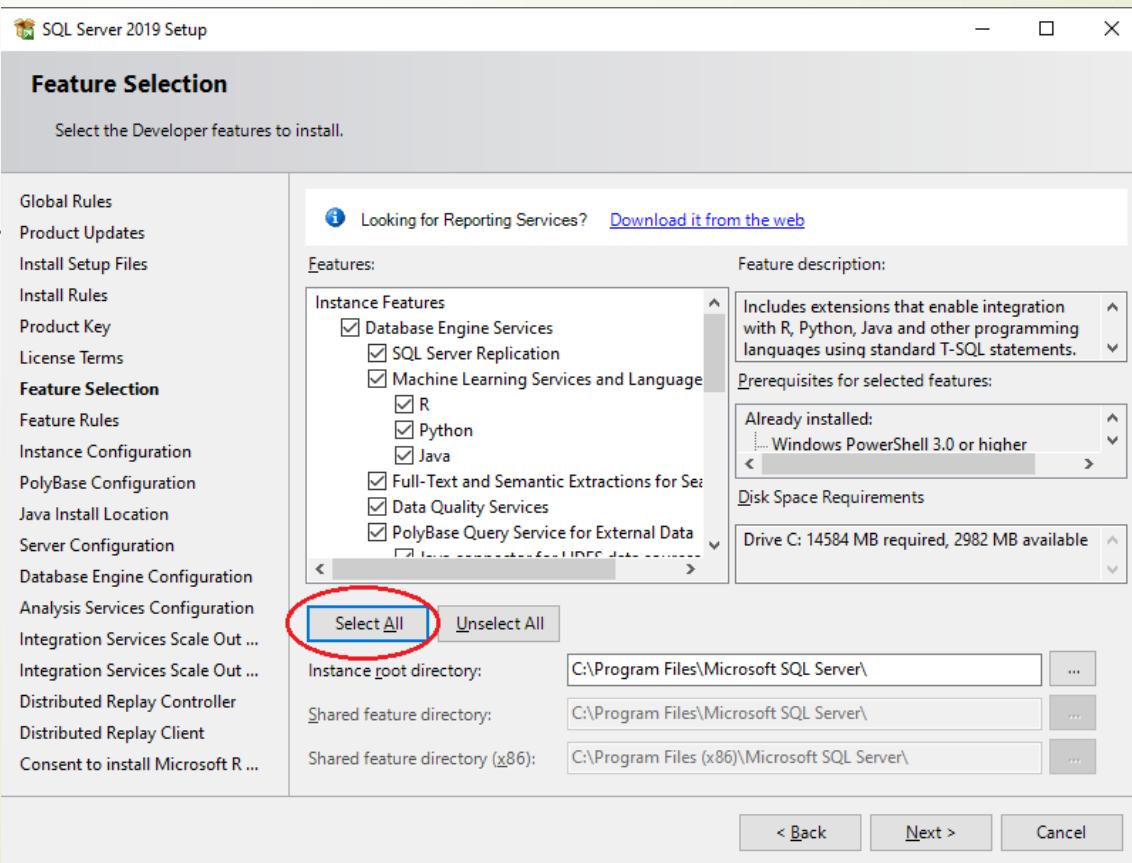


- თუ შედეგის სვეტში მწვანე ხატულ(ებ)ია, ე.ი. მიმდინარე სტრიქონში განთავსებული კომპონენტისათვის ყველაფერი კარგად არის;
- თუ შედეგის სვეტში ყვითელი ხატულ(ებ)ია, ე.ი. მიმდინარე სტრიქონში განთავსებული კომპონენტისათვის რაღაც გასათვალისწინებელია, მაგრამ ინსტალირების გაგრძელება შესაძლებელია;
- თუ შედეგის სვეტში წითელი ხატულ(ებ)ია, ე.ი. მიმდინარე სტრიქონში განთავსებული კომპონენტისათვის არის პრობლემა და ინსტალირება გვარ გაგრძელობა.

MS SQL Server-ის ინსტალირება Windows-ში

14

შემდგომ დიალოგურ ფანჯარაში მომხმარებელმა ფანჯრის შეა მიღამოში გამოსახულ სიაში უნდა მონიშნოს თუ რომელი პარამეტრების ინსტალირება არის საჭირო. ამოცანებიდან გამომდინარე მომხმარებელს შეუძლია ღილაკით Select All მონიშნოს ყველა პუნქტი გარდა:



- მანქანური სწავლების ჰგუთებისა (Machine Learning Services and Language Extensions და Machine Learning Server(Standalone)), რომელშიც გათვალისწინებულ იქნება R, Python, Java და სხვა პროგრამული ენების ინტეგრაცია T-SQL-ში;
- PolyBase მოთხოვნების სერვისის ჰგუთისა (PolyBase Query Service for External for External Data).

შემდეგ ფანჯარაზე
გადასასვლელად უნდა
ვადგინოთ დოკუმენტი Next

MS SQL Server-ის ინსტალირება Windows-ში

15

წინა დიალოგურ ფანჯარაში მოსანიშნი კომპონენტების დანიშნულებებია:

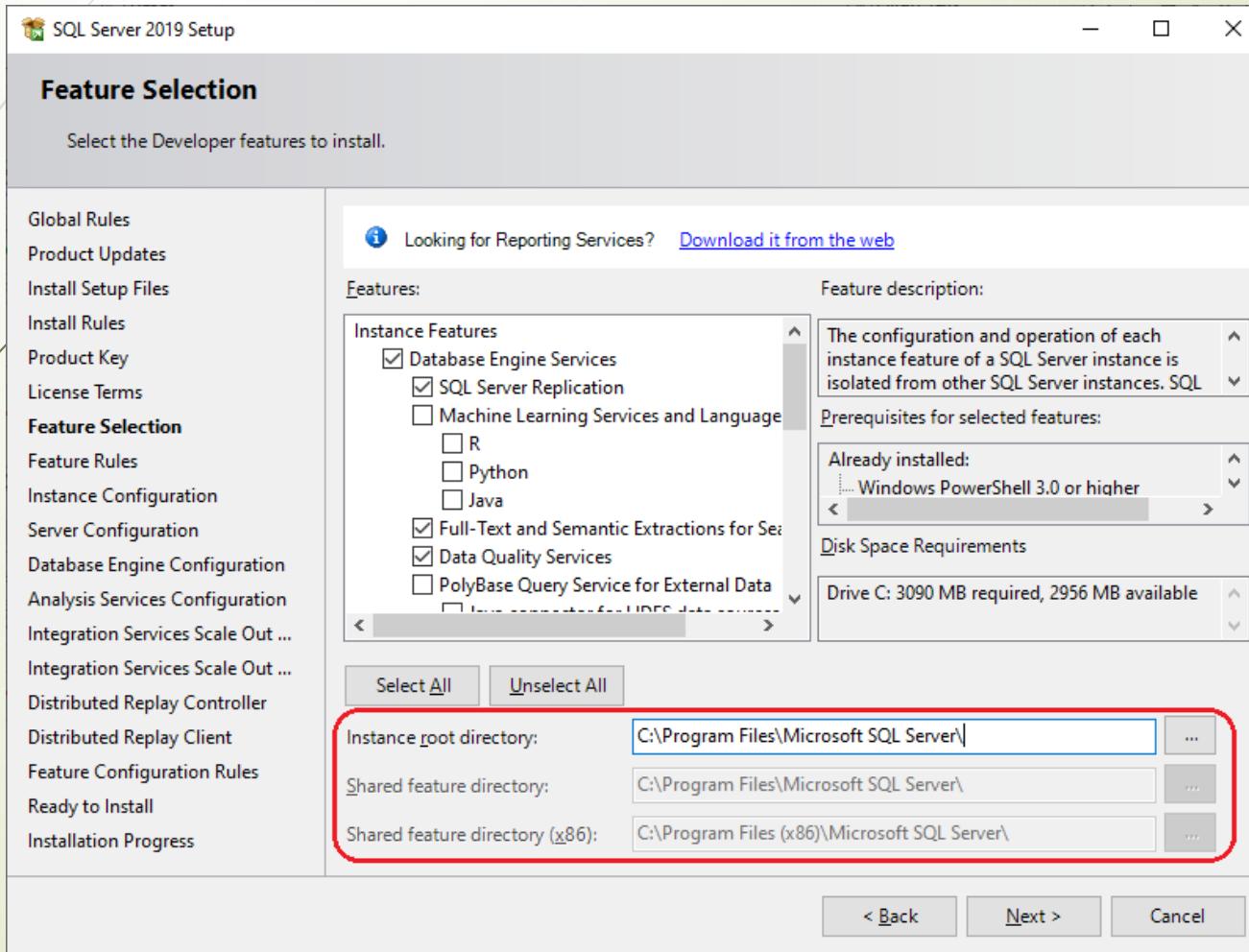
- Database Engine SQL Server - მონაცემთა ბაზის კომპონენტის ინსტალირება;
- SQL Server Replication - არააუცილებელი კომპონენტი მონაცემთა ბაზებს შორის მონაცემთა და მონაცემთა ბაზის ობიექტების კოპირებისა და გავრცელების ტექნოლოგიების ერთობლიობა და ასევე მონაცემთა ბაზების სინქრონიზაციისათვის;
- Machine Learning Services and Language - არააუცილებელი კომპონენტი მანქანური სწავლების სერვისებისა (R და Python) და ენის გაფართოებისათვის (Java);
- Full-Text and Semantic Extractions for Search - არააუცილებელი კომპონენტი სრულტექსტოვანი და სემანტიკური გაფართოების ძებნისათვის;
- Data Quality Services (DQS) - არააუცილებელი კომპონენტი მონაცემთა ხარისხის სერვისებისათვის (DQS-ის ინსტალირებისათვის SQL Server-ის ინსტალირების დასრულების შემდეგ საჭიროა დამატებითი მოქმედებები);
- Polybase Query Service for External Data - არააუცილებელი კომპონენტი Java-ს HDFS მონაცემთა წყაროსთან დასაკავშირებლად.

დანარჩენი დამატებითი შესაძლებლობები შესაძლებელია გამოყენებულ იქნეს მრავალ ტიპიურ მომსახურობის სკონარიბში

MS SQL Server-ის ინსტალირება Windows-ში

16

ამავე დიალოგურ ფანჯარაში მომხმარებელს ეძღვევა შესაძლებლობა შეცვალოს MS SQL Server-ის სტანდარტული საინსტალაციო გზები:



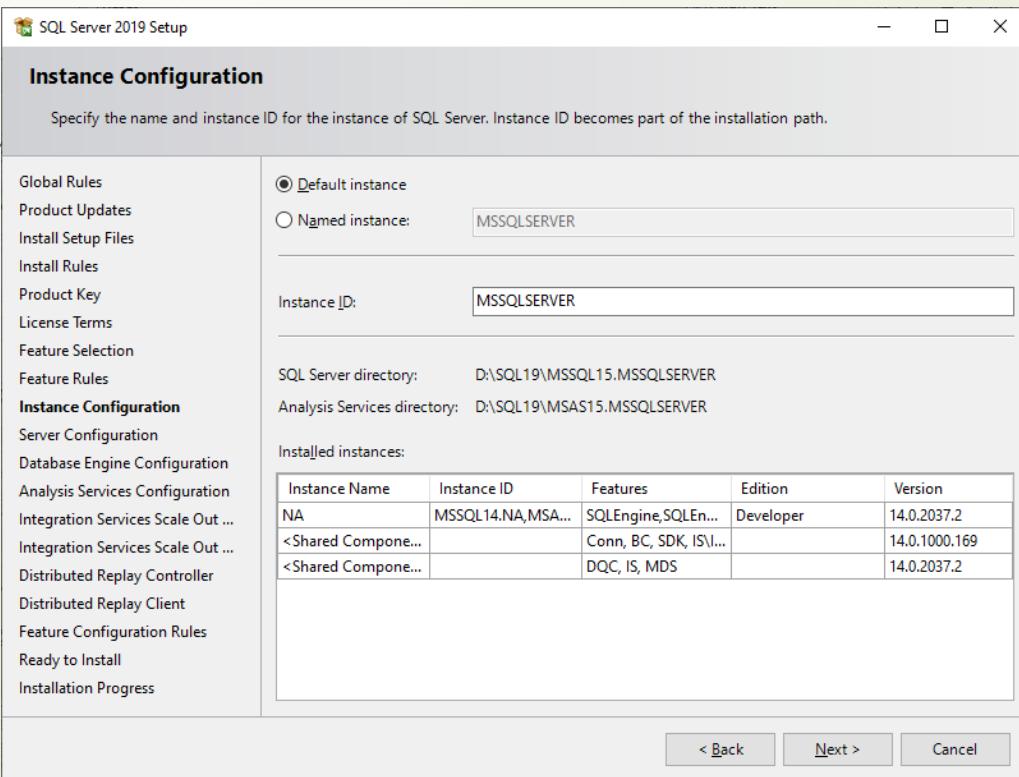
შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი
Next.

MS SQL Server-ის ინსტალირება Windows-ში

17

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დააფიქსიროს MS SQL Server-ის სახელი:

- Default Instance - სტანდარტული სახელი MSSQLSERVER;
- Named instance - ნებისმიერი სახელი, რომელიც სურს მომხმარებელს, მხოლოდ უნდა იქნეს დაცული სისტემის მიერ დასახული სახელების წესები:



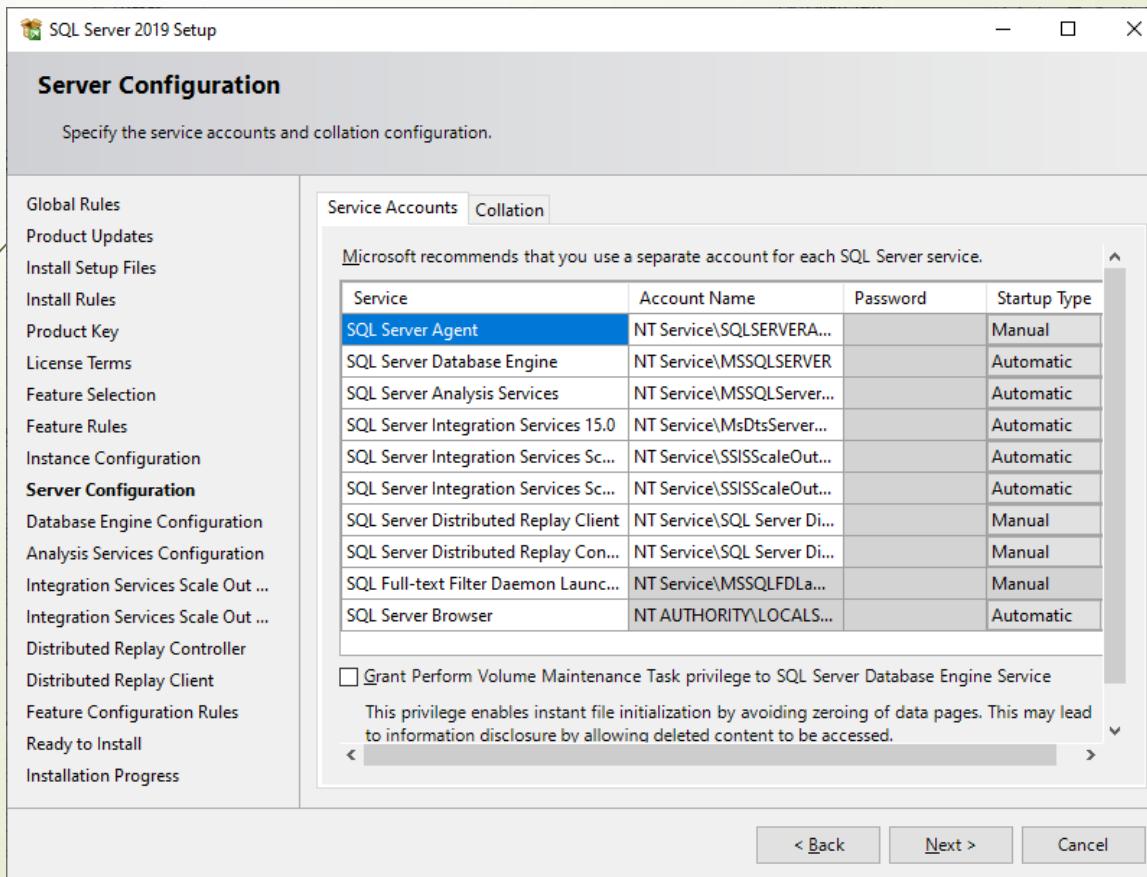
- ვერ დაიწყება და ვერ დასრულდება ქვეშ ხაზგასმის „_“ სიმბოლოთი;
- ვერ დაერქმევა დარეზერვებული სიტყვა (მაგ. Default);
- 8ომა 16 სიმბოლომდე;
- პირველი სიმბოლო Unicode 2.0;
- მეორე სიმბოლოდან Unicode 2.0, ათწილადი რიცხვები, დოლარის ნიშანი „\$“ და ქვეშ ხაზგასმის „_“ სიმბოლო.

შემდეგ ფანჯარაზე გადასასვლელად უნდა გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

18

შემდგომ დიალოგურ ფანჯრის პირველ ჩანართში Service Accounts მომხმარებელს ეძლევა შესაძლებლობა MS SQL Server-ის ყველა სერვისებისათვის დააფიქსიროს მოწყვლადობის პაროლები, ხოლო მეორე ჩანართში Collation - მიუთითოს სისტემას მონაცემთა ბაზის და ანალიზის სერვისების მონაცემთა ენის კოდირებები:

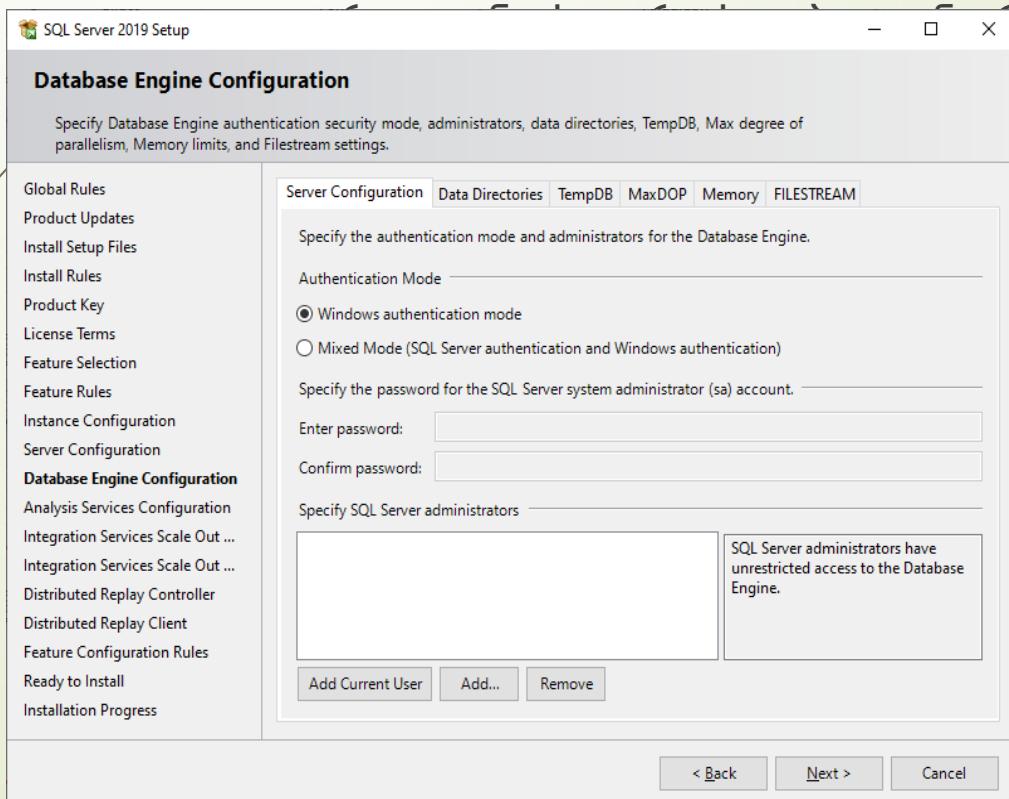


შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

შემდგომი დიალოგური ფანჯრის პირველ ჩანართში Server Configuration მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს MS SQL Server-ის ავთენტიფიკაციის რეჟიმი:

- Windows authorization mode – MS SQL Server-ში ავთენტიფიკაცია განხორციელდება Windows-ის ავთენტიფიკაციის მიხედვით (ეს ვარიანტი უფრო მოლირდება Windows-ის უსაფრთხოების



- Mixed Mode (SQL Server authorization and Windows authorization) – შერეული რეჟიმი, ანუ MS SQL Server-ში ავთენტიფიკაცია განხორციელდება როგორც Windows-ის, ასევე SQL Server-ის ავთენტიფიკაციით, რომლის პაროლიც უნდა მიეთითოს ამავე ფანჯარაში (128-მდე ასო, ციფრი და სიმბოლოები).

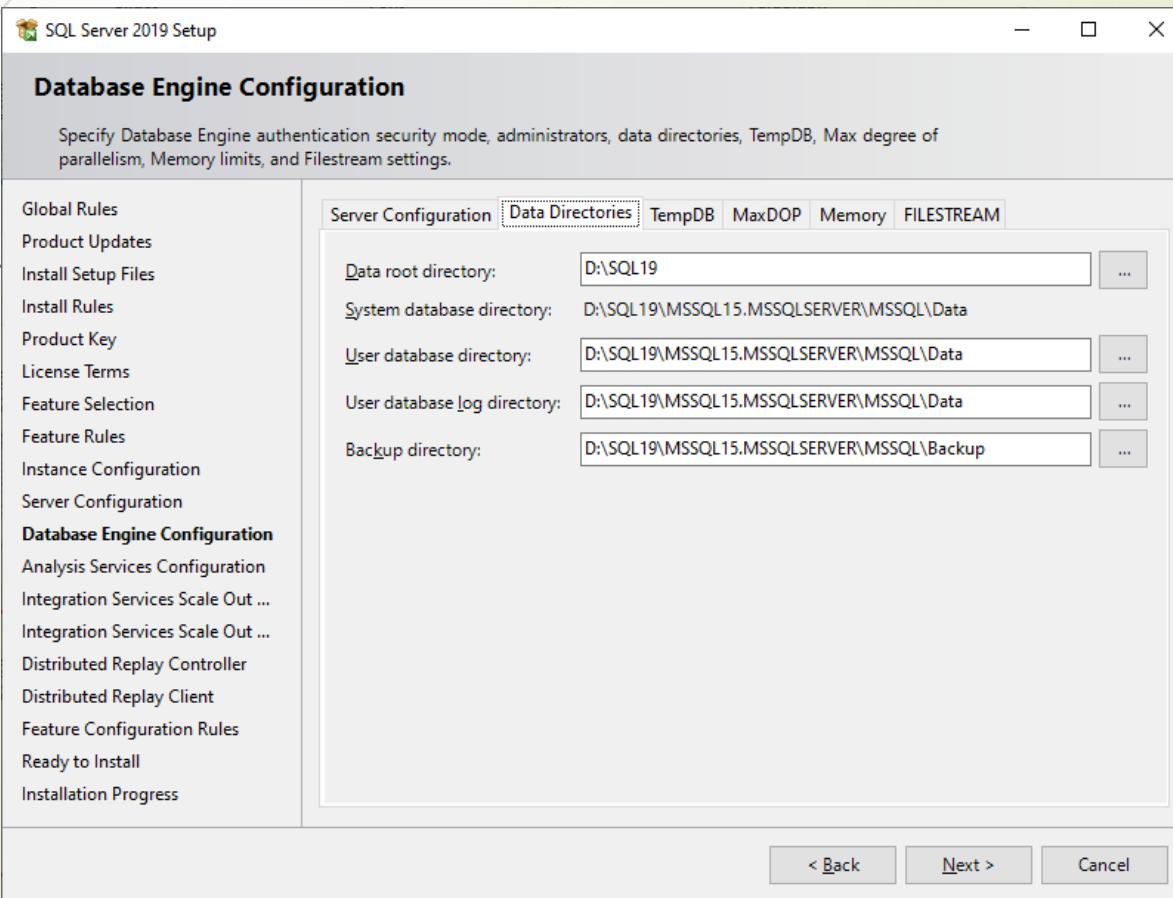
ასევე, ფანჯრის ქვედა მიღამოში აუცილებლად უნდა იქნენ მითითებულნი SQL Server-ის ადმინისტრატორები (მაგალით თვლასას Add Current User).

MS SQL Server-ის ინსტალირება Windows-ში

20

იგივე დიალოგური ფანჯრის მეორე ჩანართში Server Configuration მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს MS SQL Server-ის მონაცემთა საქაღალდები:

□ Data root directory – მონაცემთა ძირითადი საქაღალდე;

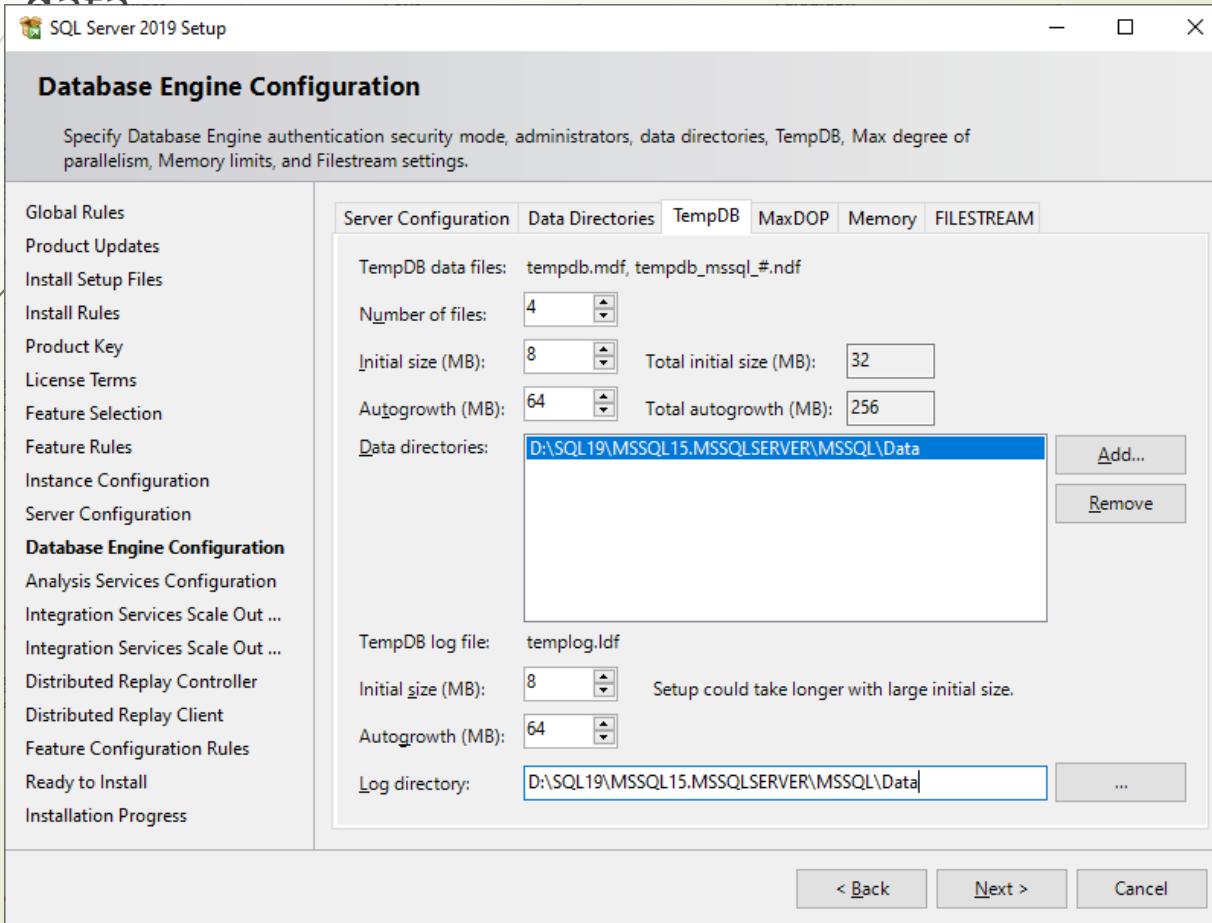


- Data root directory – მონაცემთა ძირითადი საქაღალდე;
- System database directory – სისტემური მონაცემთა ბაზების საქაღალდე;
- User database directory – მომხმარებლის მონაცემთა ბაზების საქაღალდე;
- User database log directory – მომხმარებლის მონაცემთა ბაზების ჟურნალების საქაღალდე;
- Backup directory – მონაცემთა ბაზების სარეზერვო ასლების საქაღალდე.

MS SQL Server-ის ინსტალირება Windows-ში

21

იგივე დიალოგური ფანჯრის მესამე ჩანართში TempDB მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს MS SQL Server-ის დროებითი მონაცემთა ბაზების რეკვიზიტები: ზედა მიღამოში დროებითი მონაცემთა ბაზების ფაილების (TempDB data files) და ქვედა მიღამოში დროებითი მონაცემთა ბაზების ჟურნალების ფაილების (TempDB log file):



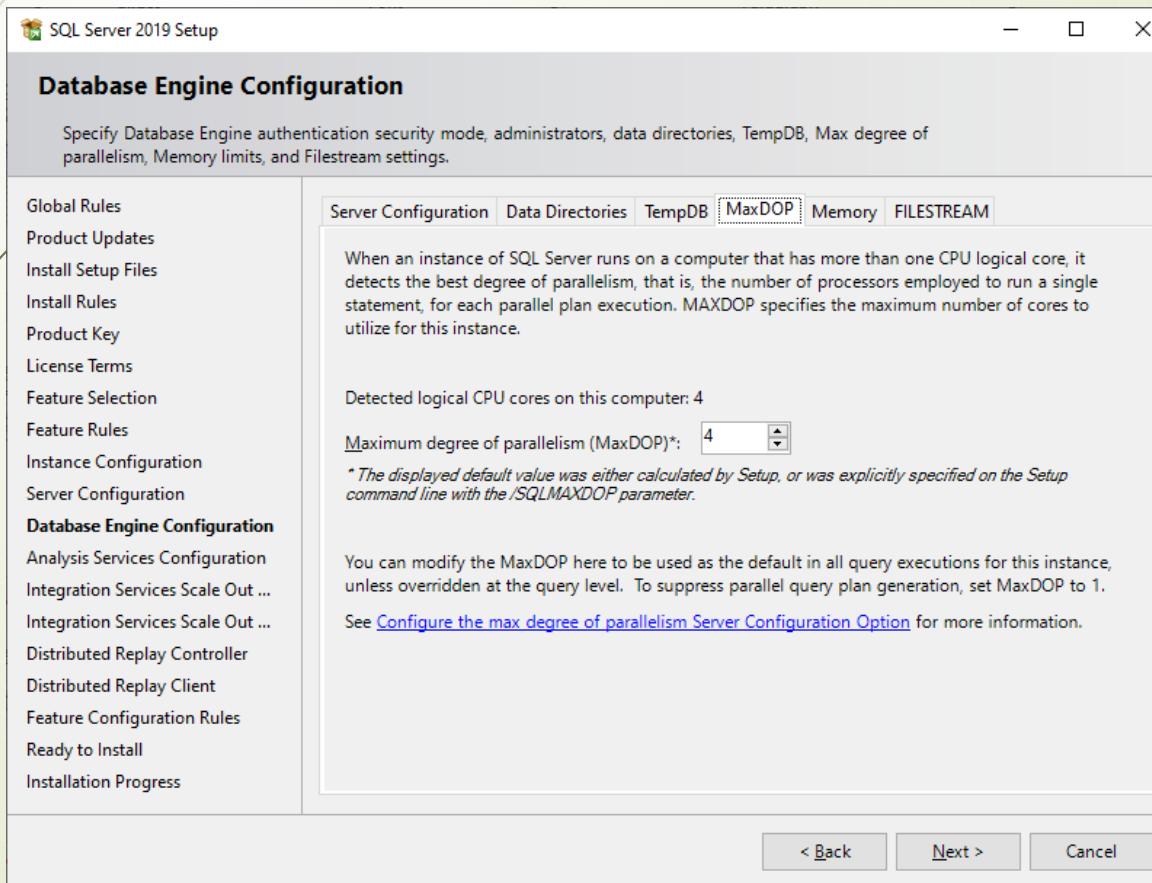
files) და ქვედა მიღამოში დროებითი მონაცემთა ბაზების ჟურნალების ფაილების (TempDB log file):

- **Number of files** - ფაილების რაოდენობა;
- **Initial size** - საწყისი ზომა;
- **Autogrowth** - ნაზრდის ზომა;
- **Data directories** - დროებითი მონაცემთა ბაზების საქაღალდე;
- **Log directory** - მონაცემთა ბაზების ჟურნალის საქაღალდე.

MS SQL Server-ის ინსტალირება Windows-ში

22

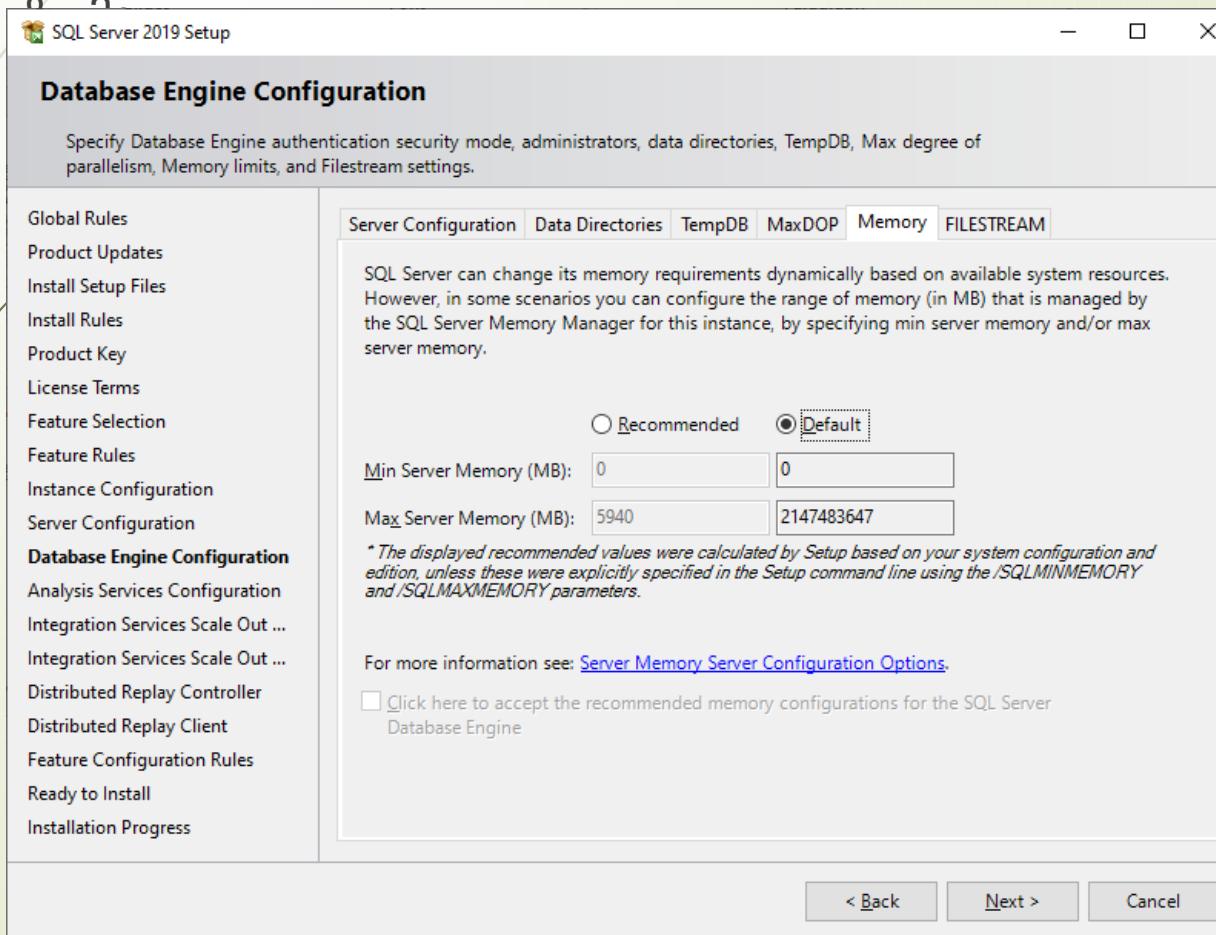
იგივე დიალოგური ფანჯრის მეოთხე ჩანართში MaxDOP მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს მიმღინარე კომპიუტერის პროცესორის ბირთვების რაოდენობის მიხედვით MS SQL Server-ის ამოცანების გადაწყვეტის პარალელიზმის ხარისხი (რაოდენობა):



MS SQL Server-ის ინსტალირება Windows-ში

23

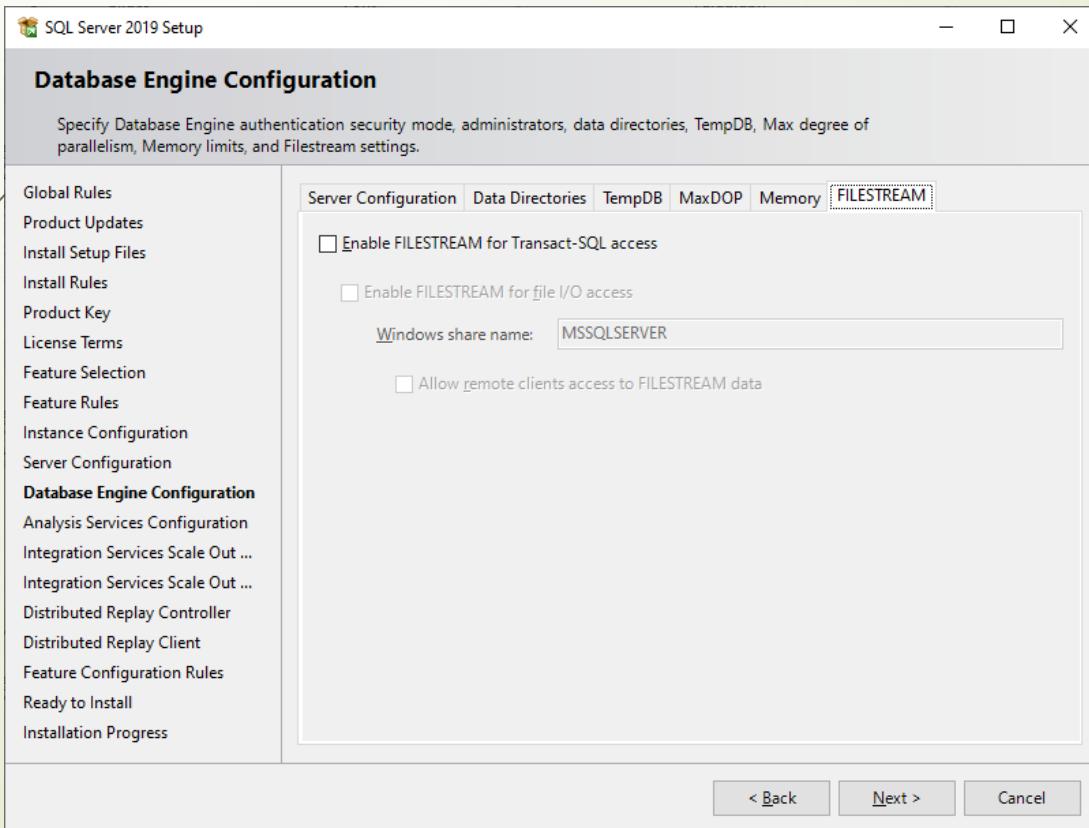
იგივე დიალოგური ფანჯრის მეხუთე ჩანართში Memory მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს MS SQL Server-ის მუშაობისათვის საჭირო კომპიუტერის მეხსიერების რეკომენდირებული ან სტანდარტული მინიმალური და მაქსიმალური



MS SQL Server-ის ინსტალირება Windows-ში

24

იგივე დიალოგური ფანჯრის მექანიზე ჩანართში FILESTREAM მომხმარებელს ეძლევა შესაძლებლობა განსაზღვროს MS SQL Server-ის მონაცემთა ბაზის კომპონენტის Database Engine-ს გამოყენებით შეინახოს NTFS ფაილურ სისტემაში varbinary(max) ტიპის დოდი მონაცემების ორობითი ობიექტები (BLOB) ფაილების სახით, რომლის ჩასართავად უნდა მოინიშნოს Enable FILESTREAM for Transact-SQL Access.



Win32-ს ნაკადური წვდომის დაშვებისათვის უნდა მოინიშნოს Enable FILESTREAM for file I/O access.

FILESTREAM-ის მონაცემთა შენახვის საქაღალდის სახელი უნდა ჩაინეროს Windows share name სტრიქონში.

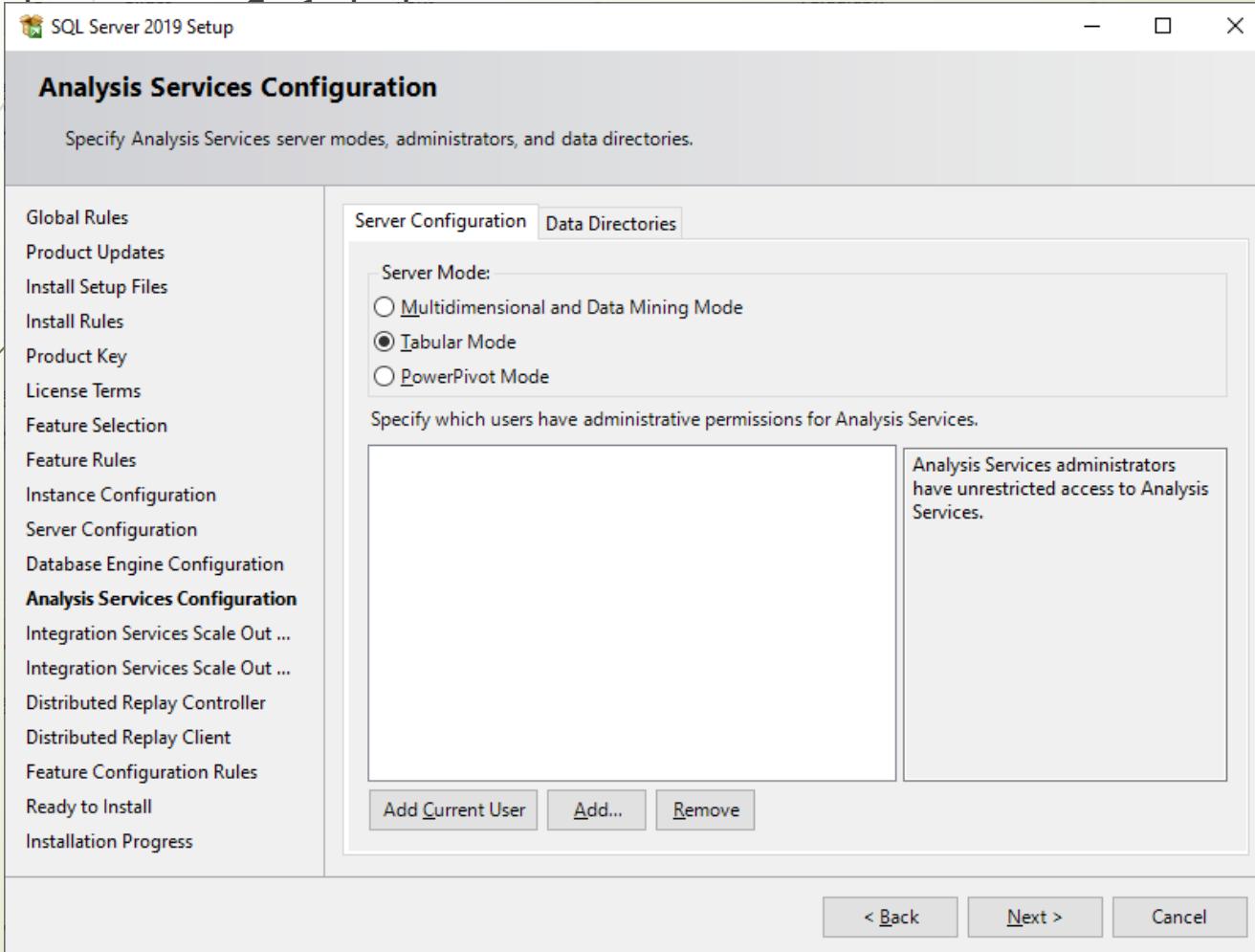
ამ ფოლდერზე დაშორებული კლიენტების წვდომისათვის უნდა მოინიშნოს Allow remote clients access for FILESTREAM data.

შემდეგ ფანჯარაზე გადასასვლელად უნდა გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

25

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დააფიქსიროს MS SQL Server-ის ანალიზის სერვისების კონფიგურირების რეჟიმები,



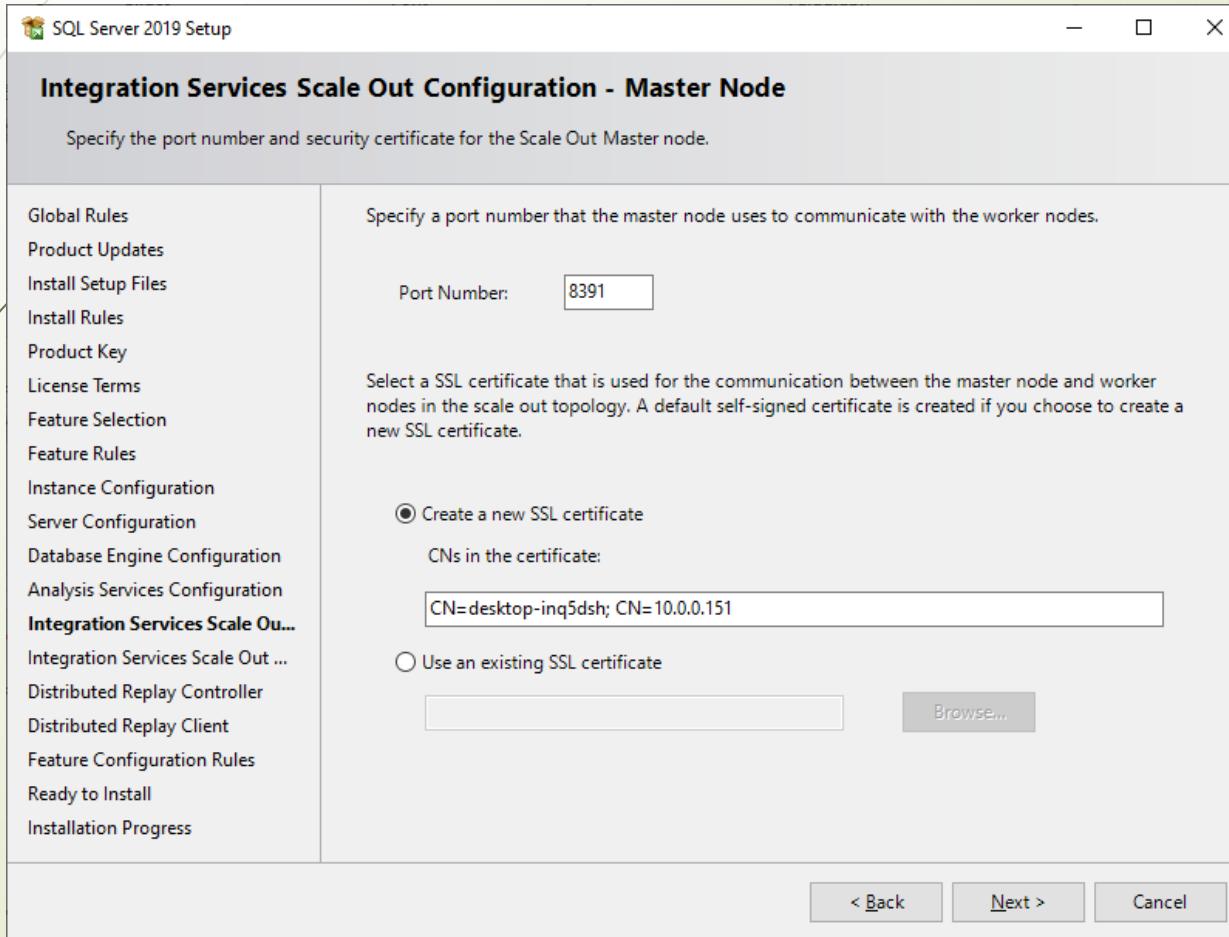
მიღამოში უნდა იქნებ
მითითებულნი SQL Server-ის ანალიზის სერვისების ადმინისტრატორები (მაგალითად, ღილაკის Add Current User გააქტიურებით დაემატება მიმდინარე მომხმარებელი).

შემდეგ ფანჯარაზე გადასასვლელად უნდა გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

26

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დააფიქსიროს MS SQL Server-ის სამუშაო პორტის ნომერი (სტანდარტულად იგი არის 8391), შექმნას ან მიუთითოს არსებული უსაფრთხოების SSL-სერტიფიკატი:

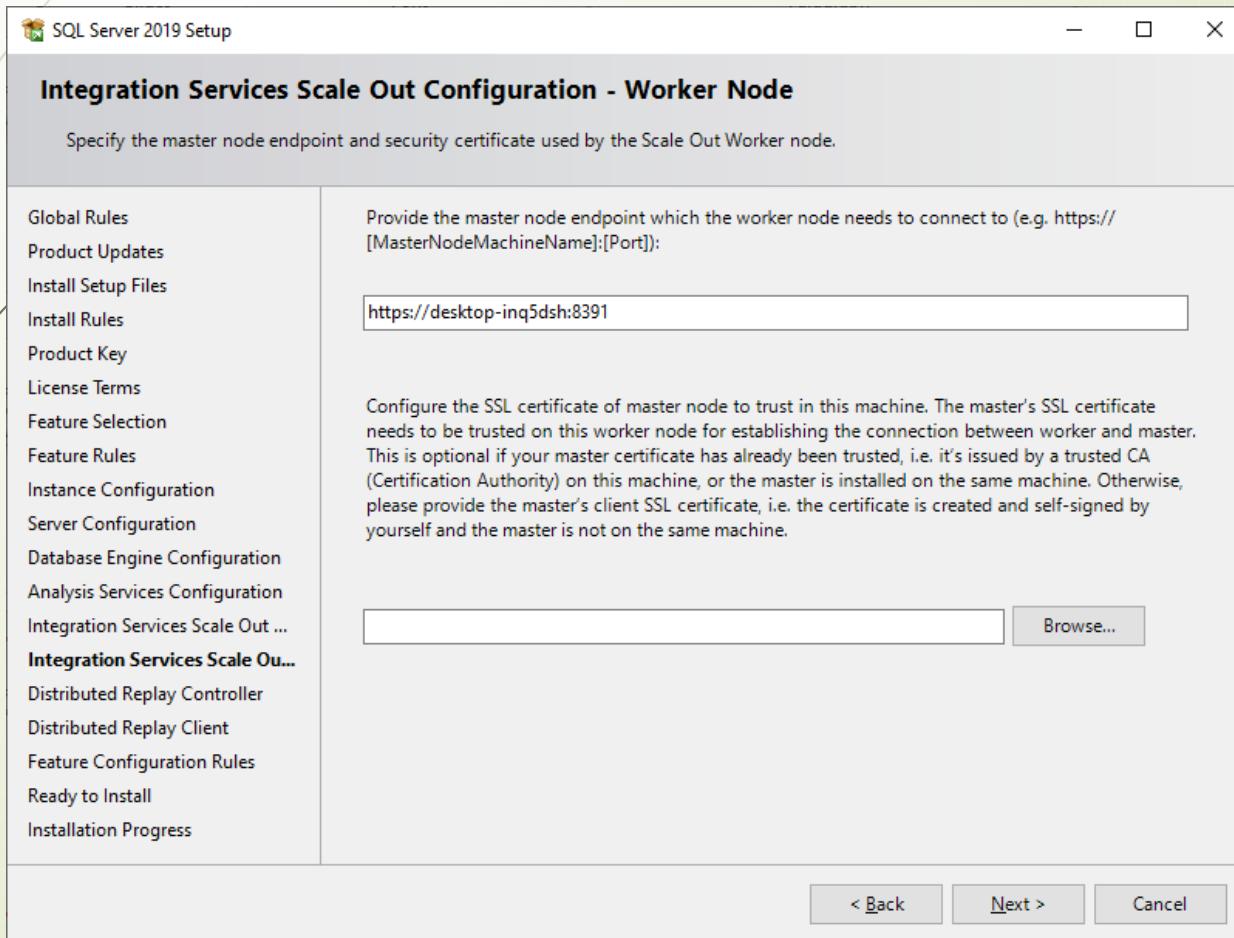


შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

27

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დააფიქსიროს MS SQL Server-თან მიერთების ვებ-მისამართი და რეჟიმის ოსტატის კონფიგურირების SSL-სერტიფიკატი:

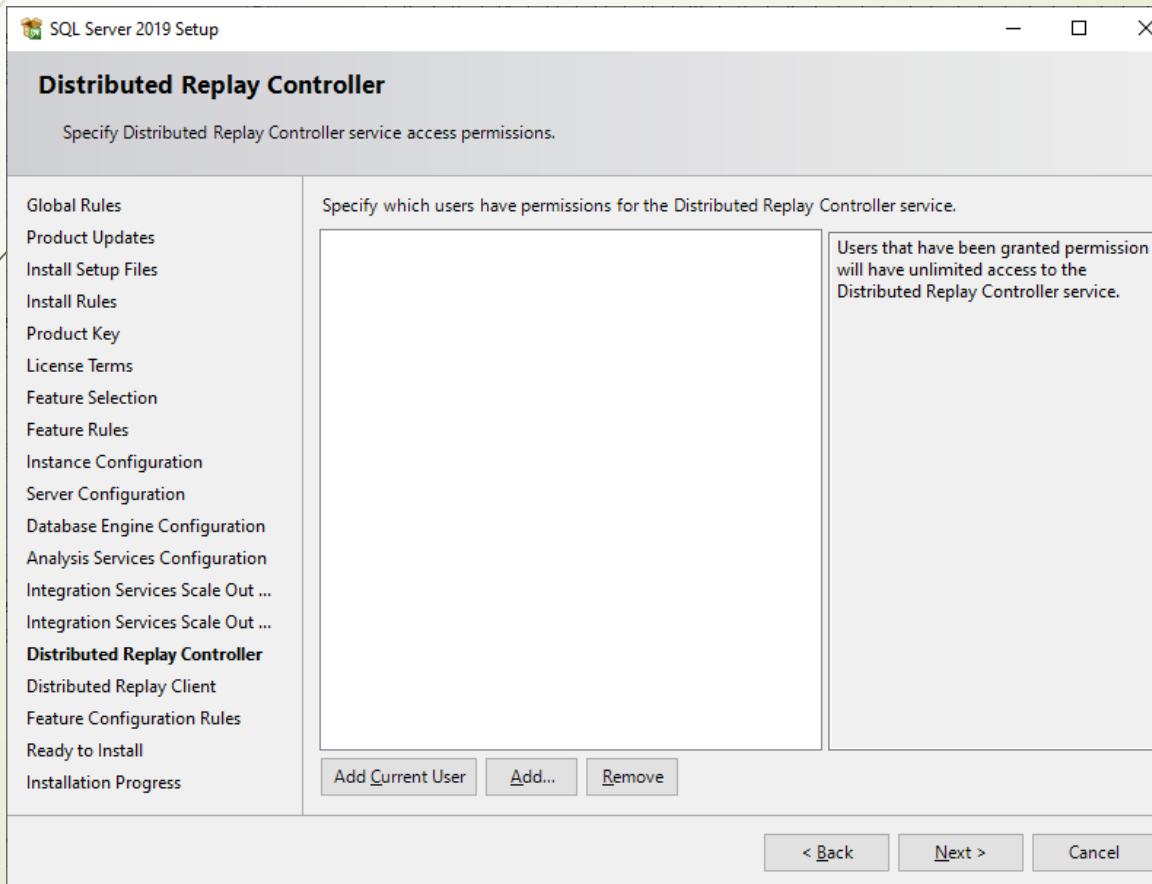


შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

28

შემდგომ დიალოგურ ფანჯარაში მომხმარებელმა აუცილებლად უნდა მიუთითოს MS SQL Server-ში განაწილების კონტროლერის სერვისების მთავარი მომხმარებლები (მაგალითად, ღილაკის Add Current User გააქტიურებით დაემატება მიმდინარე მომხმარებელი):

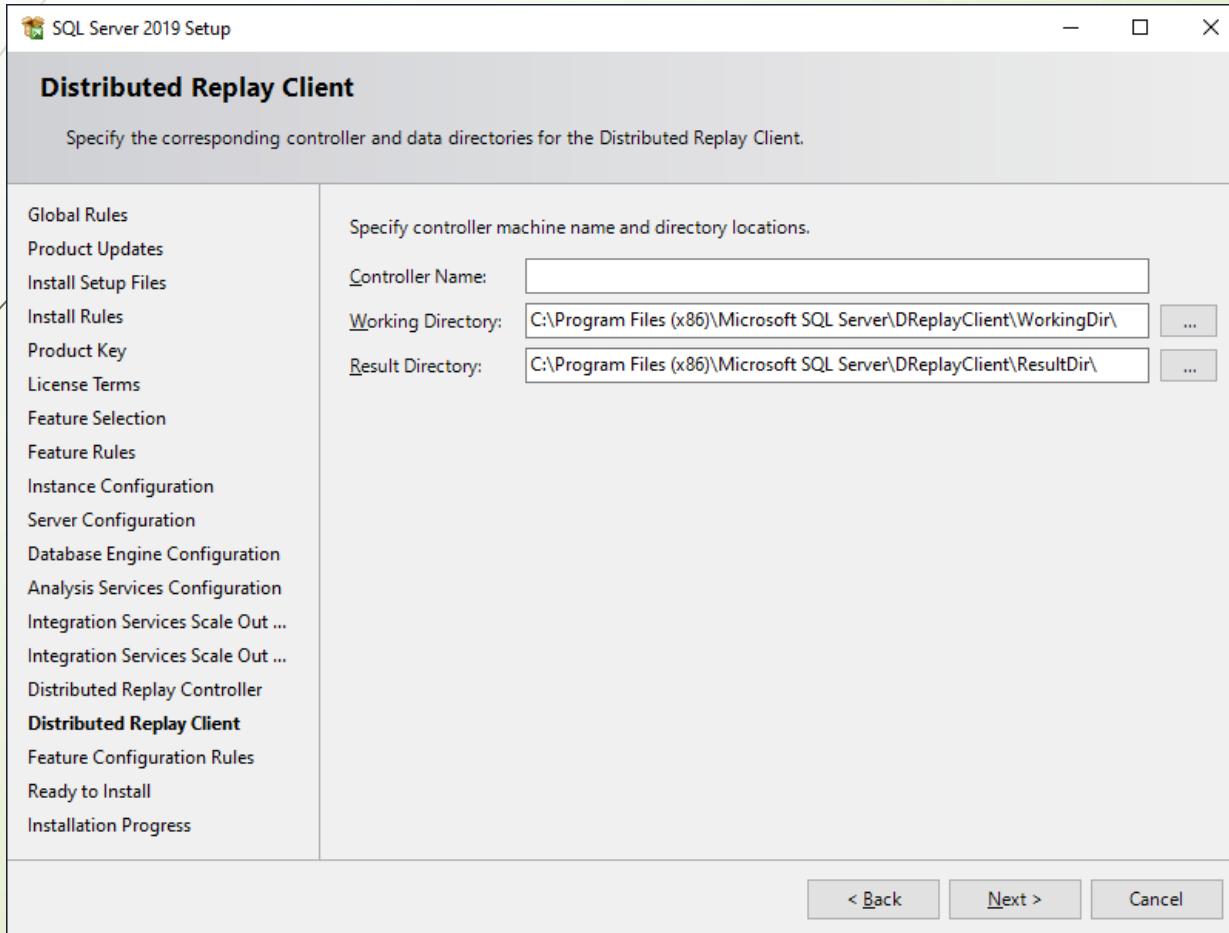


შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

29

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა წინა ფანჯარაში მითითებულ MS SQL Server-ში განსაზღვროს განაწილების კონტროლერის მანქანის სახელი, სამუშაო და შედეგების საქაღალდეები:

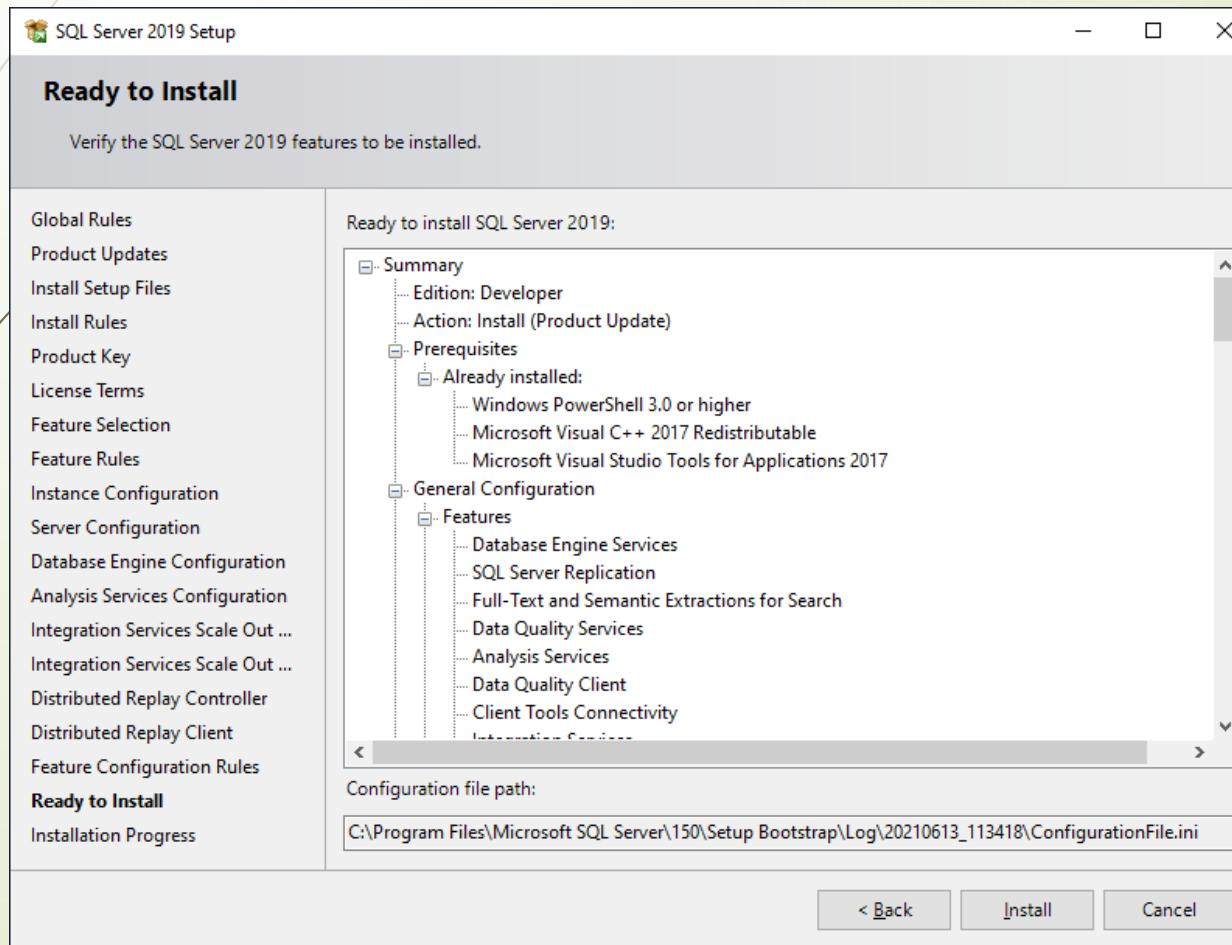


შემდეგ ფანჯარაზე
გადასასვლელად უნდა
გააქტიურდეს ღილაკი Next.

MS SQL Server-ის ინსტალირება Windows-ში

30

შემდგომ დიალოგურ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დაათვალიეროს MS SQL Server-ის ინსტალირების გეგმა და თანხმობის შემთხვევაში გააქტიუროს ღილაკი Install ინსტალირების დასაწყებად:



SQL Server Management Studio-ს ინსტალირება

31

რაც შეეხება SQL Server Management Studio (SSMS)-ს, რომელიც არის MS SQL Server-ის ადმინისტრირებისა და მართვის ინტეგრირებული გარემო, იგი უნდა იქნეს გადმოწერილი შემდეგი მისამართიდან:

<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>

სხვა ენაზე მოქმედი SSMS შესაძლებელია გადმოწერილ იქნეს იმავე ვებ-გვერდის ბოლოში განთავსებულ ნაწილში Available languages:

Available languages

This release of SSMS can be installed in the following languages:

SQL Server Management Studio 18.9.1:

Chinese (Simplified) ↗ | Chinese (Traditional) ↗ | English (United States) ↗ | French ↗ | German ↗ | Italian ↗ |
Japanese ↗ | Korean ↗ | Portuguese (Brazil) ↗ | Russian ↗ | Spanish ↗

ხოლო წინა ვერსიების გადმოსაწერად გამოყენებულ უნდა იქნეს უფრო ქვემოდ განლაგებული Previous versions:

Previous versions

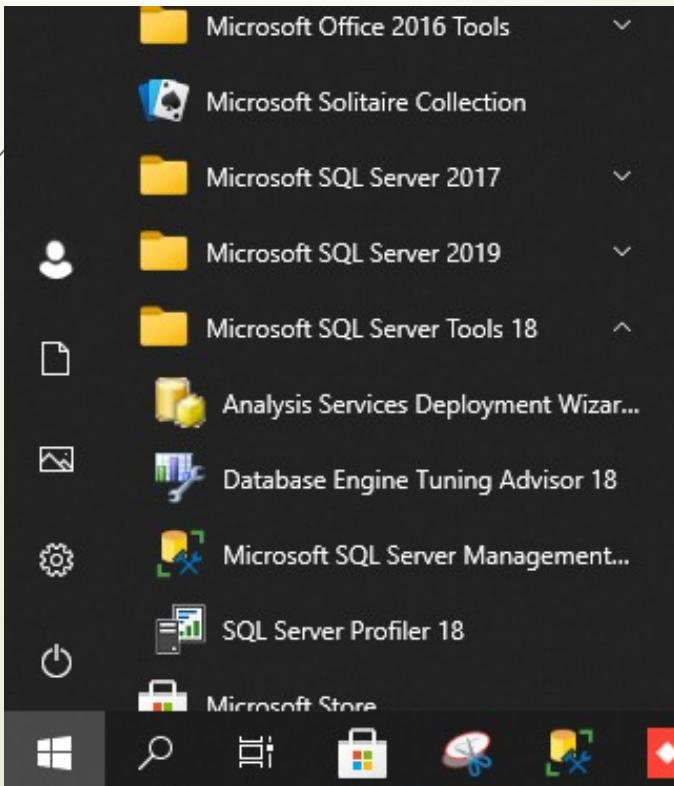
This article is for the latest version of SSMS only. To download previous versions of SSMS, visit [Previous SSMS releases](#).

SQL Server Management Studio-ს ინსტალირება

32

სტრიქონის გააქტიურების შედეგად ჩამოიტვირთება ფაილი „SSMS-Setup-ENU.exe“, რომელიც უნდა გაეშვას სისტემაში ადმინისტრატორის უფლებებით და ინსტალირებულ იქნეს სისტემაში სტანდარტული (ძირითადად Next) ღილაკების გამოყენებით.

ინსტალირების დასრულების შემდგომ ოპერაციული გარემო მოითხოვს კომპიუტერის გადატვირთვას, რასაც მომხმარებელი უნდა დაეთანხმოს.



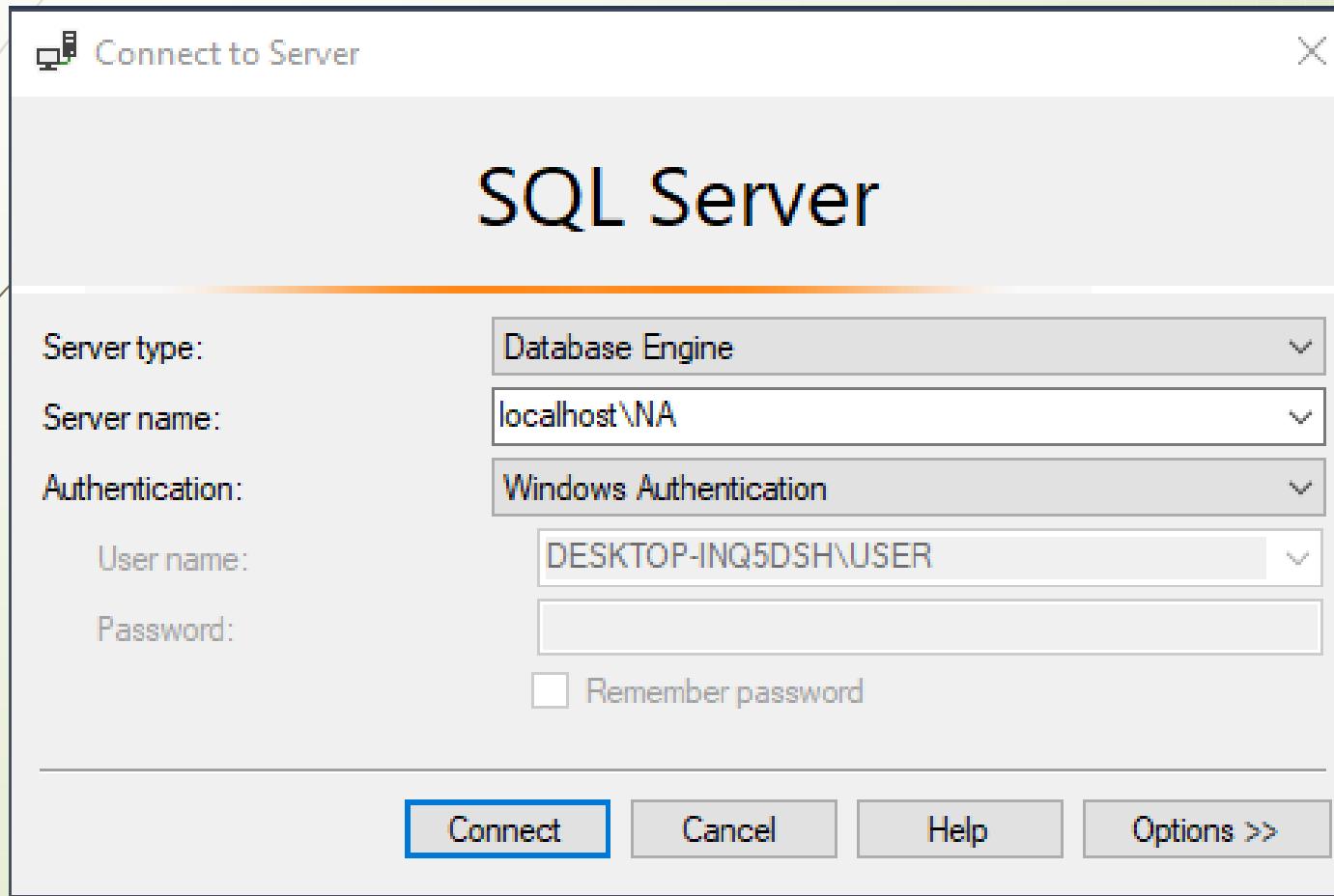
სისტემის ახლად ჩატვირთვის შემდგომ MS Windows 10-ის სტარტ-მენიუში მომხმარებელმა უნდა მოიძიოს ჯგუფი Microsoft SQL Server Tools 18, ჩამოშალოს იგი და გაააქტიუროს სტრიქონი Microsoft SQL Server Management Studio 18.

ამ სტრიქონის გააქტიურებისას გაეშვება ზემოაღნიშნული პროგრამა.

SQL Server Management Studio-ში ავთენტიფიკაცია

33

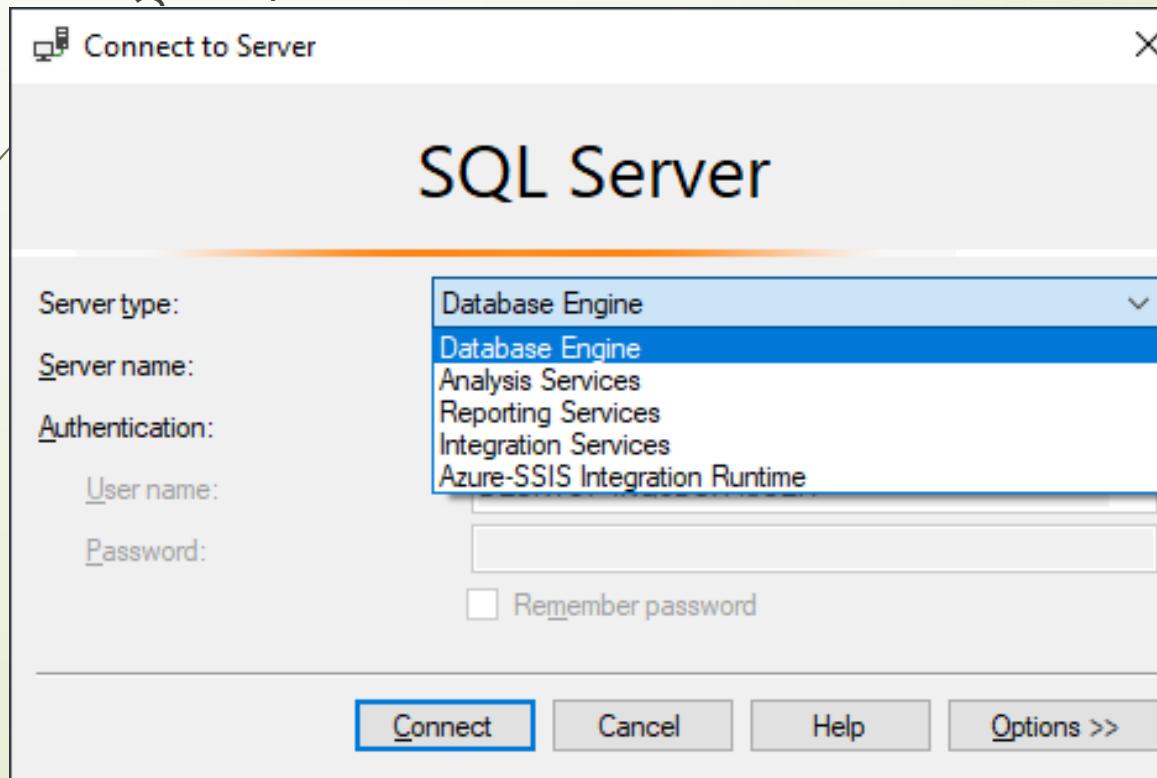
SQL Server Management Studio (SSMS) არის SQL Server-ის მართვისა და ადმინისტრირების გარემო. Microsoft SQL Server Management Studio 18-ს გააქტიურებისას იტვირთება შესაბამისი გარემო და იხსნება დიალოგური ფანჯარა:



SQL Server Management Studio-ში ავთენტიფიკაცია

34

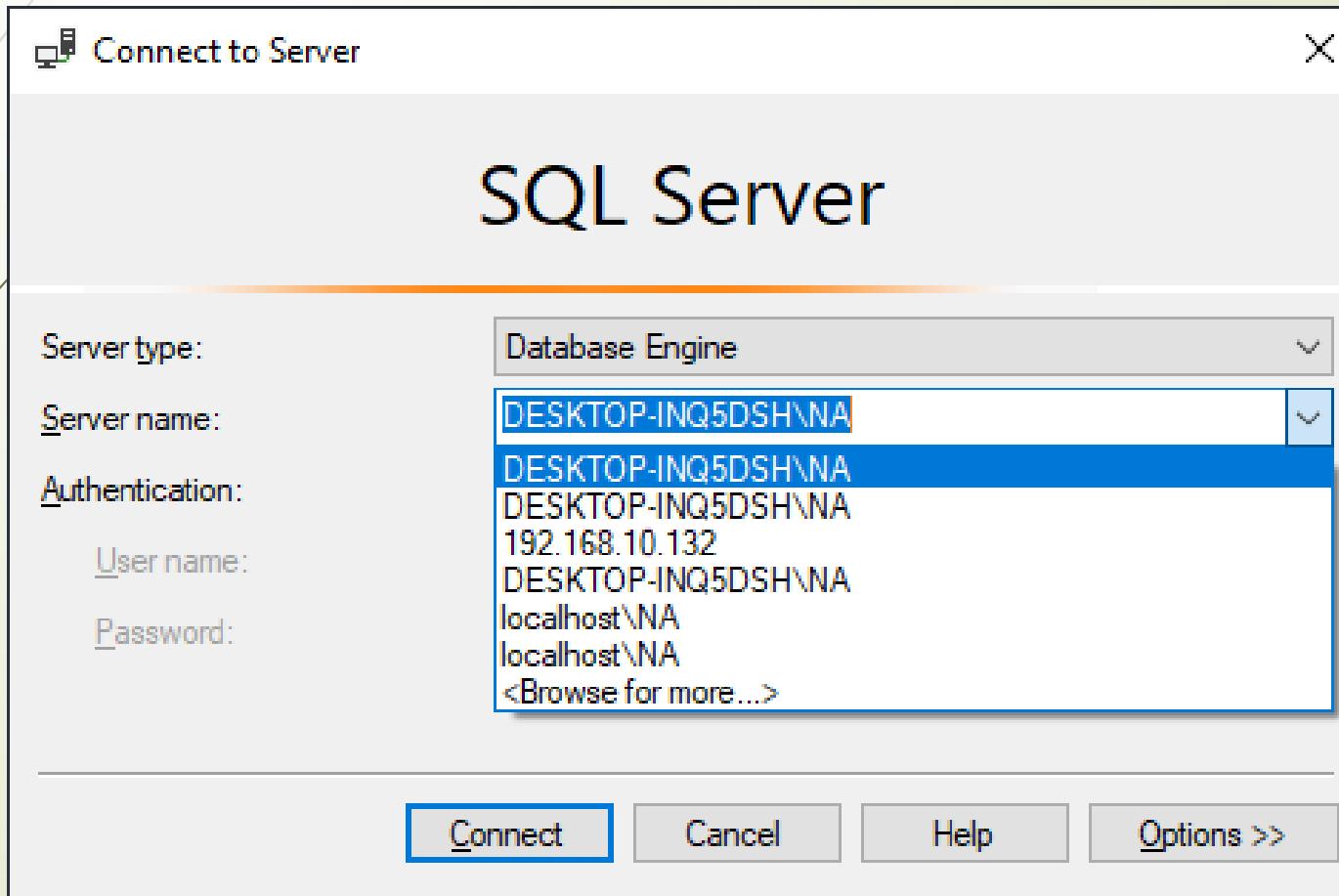
SQL Server Management Studio (SSMS)-ს გარემოს დიალოგური ფანჯრის პირველ სტრიქონში უნდა მიეთითოს თუ SQL Server-ის რომელ ტიპთან სურს მომხმარებელს მუშაობა: თვით მონაცემთა ბაზის ძირითად ნაწილთან, ანალიზების სერვისებთან, ანგარიშების სერვისებთან, ინტეგრირებულ სერვისებთან თუ Azure_SSIS-ში ინტეგრაციის შესრულების გარემოსთან (უნდა მიეთითოს მონაცემთა ბაზის ძირითად ნაწილთან):



SQL Server Management Studio-ში ავთენტიფიკაცია

35

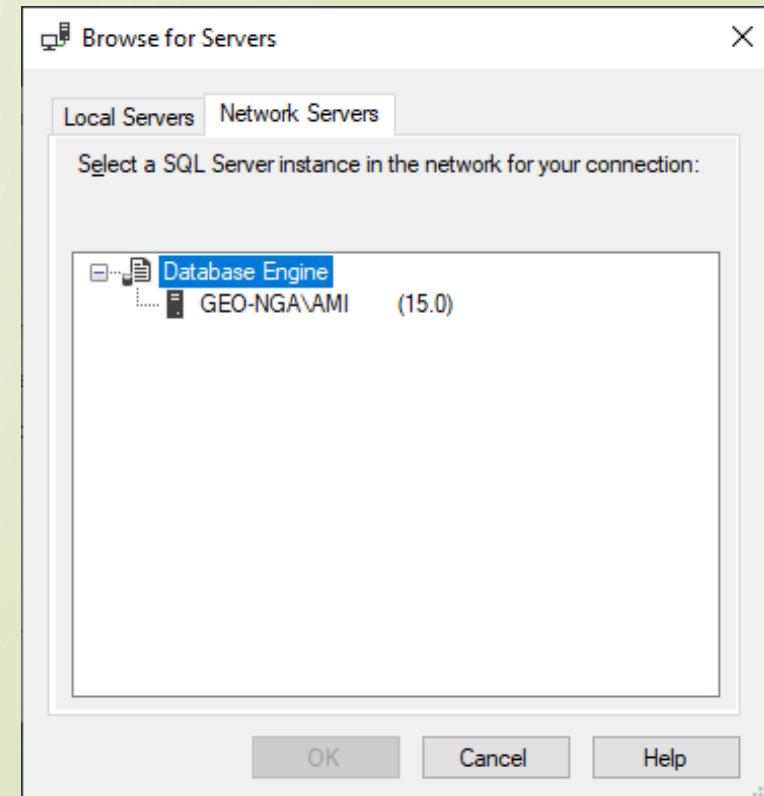
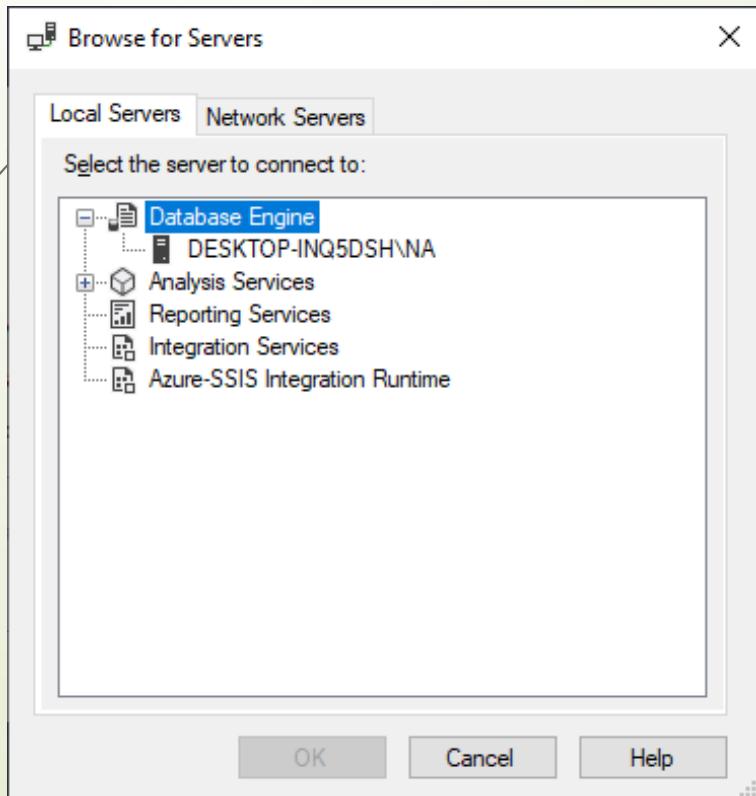
SQL Server Management Studio (SSMS)-ს გარემოს დიალოგური ფანჯრის მეორე სტრიქონში უნდა მიეთითოს SQL Server-ის რომელ მონაცემთა ბაზასთან სურს მომხმარებელს მუშაობა (უნდა მიეთითოს მონაცემთა ბაზების სიიდან ერთერთი):



SQL Server Management Studio-ში ავთენტიფიკაცია

36

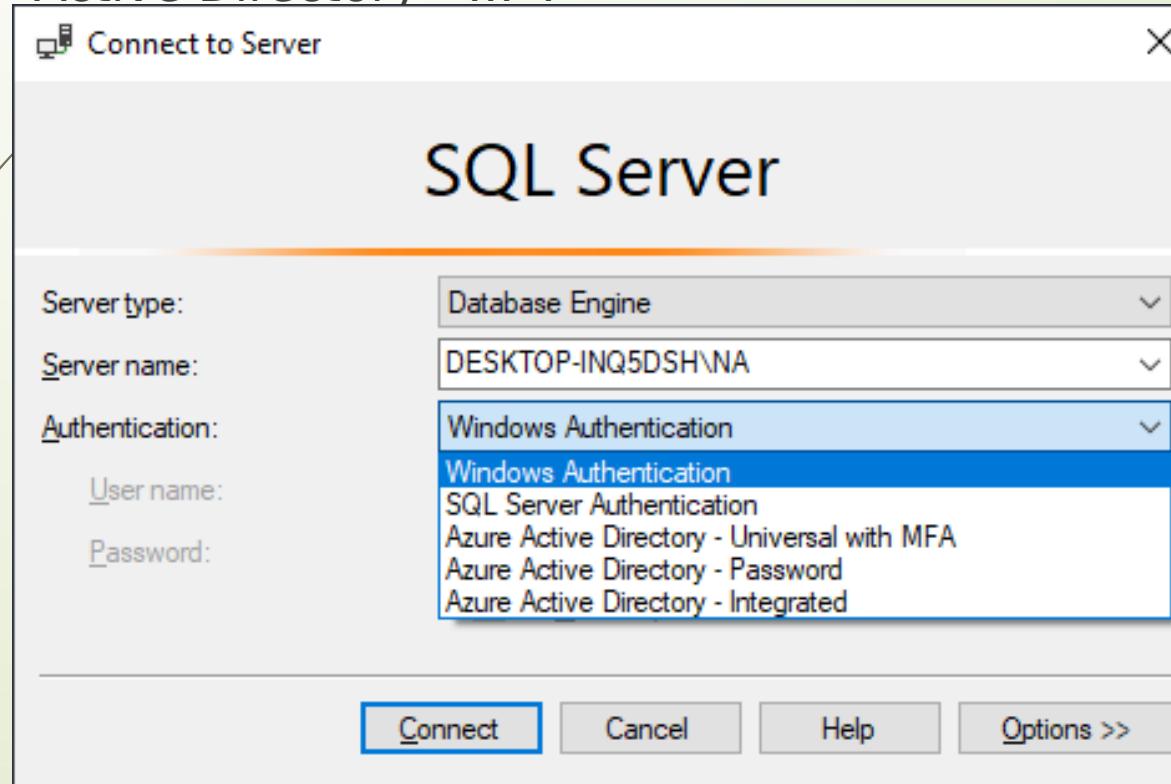
თუ მონაცემთა ბაზების სიაში არ არის სასურველი, მაშინ მომხმარებელმა უნდა გაააქტიუროს სტრიქონი „Browse for more...“ და ახლად გამოსახულ ფანჯარაში შეარჩიოს „Local Servers“ ლოკალური (არსებულ კომპიუტერზე დაინსტალირებული) ან ქსელური „Network Servers“ (კომპიუტერულ ქსელში მდებარე სხვა კომპიუტერზე დაინსტალირებული) MS SQL Server-ის მონაცემთა ბაზა:



SQL Server Management Studio-ში ავთენტიფიკაცია

37

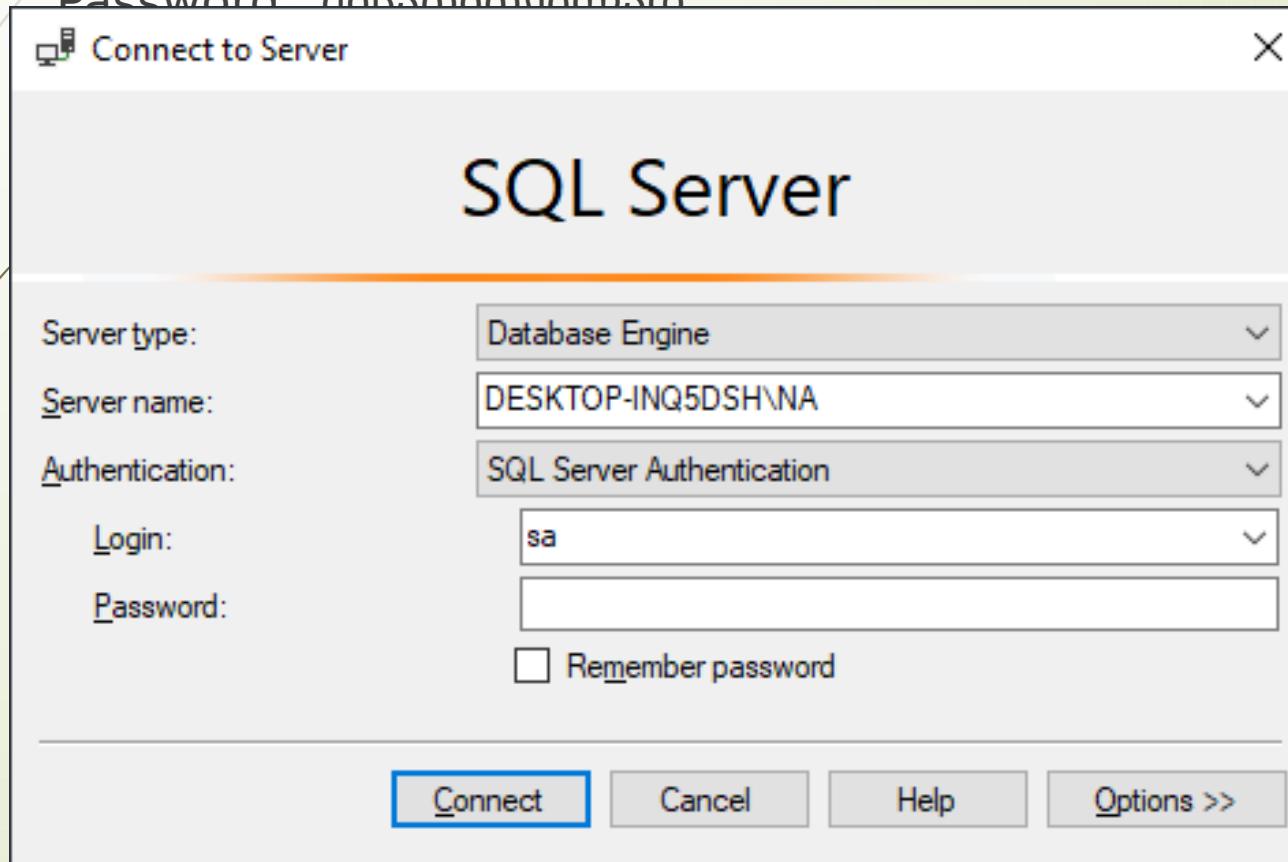
SQL Server Management Studio (SSMS)-ს გარემოს დიალოგური ფანჯრის მესამე სტრიქონში უნდა მიეთითოს SQL Server-ში წვდომის ავტორიზაციის ტიპი: Windows ოპერაციულ გარემოში არსებული მომხმარებლით ავტორიზაცია „Windows Authentication“, MS SQL Server-ში არსებული მომხმარებლით ავტორიზაცია „SQL Server Authentication“ ან Azure-ში არსებული მომხმარებლით ავტორიზაცია „Azure Active Directory - ...“:



SQL Server Management Studio-ში ავთენტიფიკაცია

38

SQL Server Management Studio (SSMS)-ს გარემოს დიალოგური ფანჯრის მეოთხე და მეხუთე სტრიქონები საჭიროებისამებრ გამოიყენება SQL Server-ში წვდომის ავტორიზაციისათვის მომხმარებლის სახელისა „User name“ და პაროლის „Password“ მისამომისამართი.

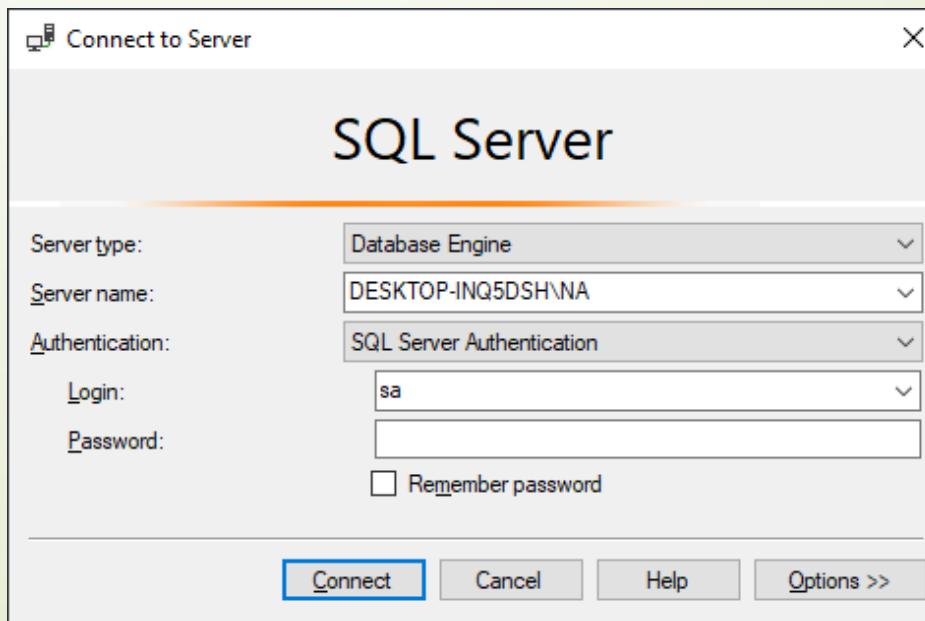


SQL Server Management Studio-ში ავთენტიფიკაცია

39

SQL Server Management Studio (SSMS)-ს გარემოს დიალოგური ფანჯრის ღილაკების დანიშნულება შემდეგია:

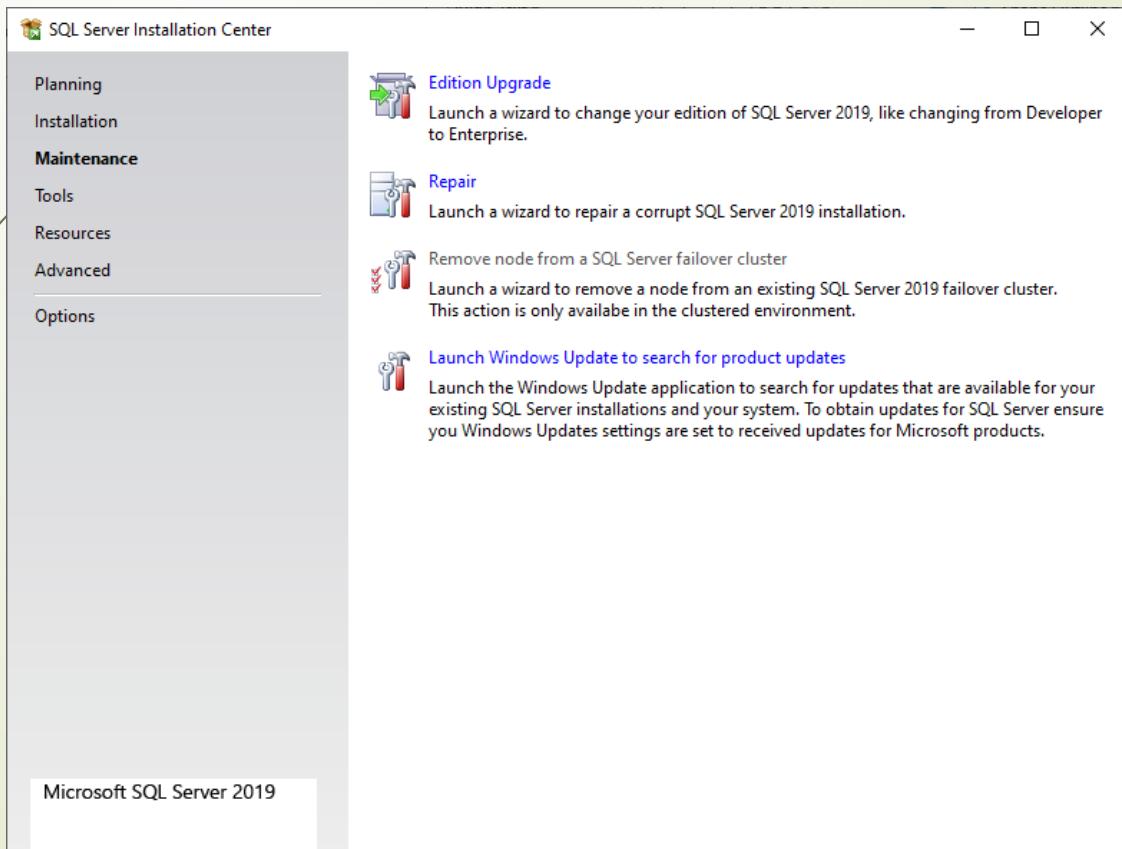
- „Connect“ - შერჩეულ მონაცემთა ბაზასთან მიერთება;
- „Cancel“ - მიერთების უარყოფა;
- „Help“ - დახმარების გამოძახება;
- „Options“ - შერჩეულ მონაცემთა ბაზასთან მიერთების დამატებითი ოფციები.



MS SQL Server-ის ვერსიის განახლება ან აღდგენა

1

MS SQL Server-ის ინსტალირებისას setup.exe-ს გაშვებისას გამოჩენილ დიალოგურ ფანჯრის მარცხენა მიდამოში მდებარე სიაში უნდა გააქტიურდეს მესამე სტრიქონი Maintenance, რის შემდეგაც მარჯვენა მიდამოში პირველი სტრიქონის Eddition Upgrade-ს გააქტიურებით განხორციელდება ვერსიის განახლება.



ამავე დიალოგური ფანჯრის მარჯვენა მიდამოს მეორე სტრიქონის გააქტიურებით ხორციელდება დაზიანებული SQL Server-ის აღდგენა.

რამდენიმე დიალოგურ ფანჯარასთან დათანხმების შემდეგ ეკრანზე გამოისახება ოპერაციის წარმატებით დასრულების გვერდი.

აღდგენა, ასევე, შესაძლებელია PowerShell-ში შემდეგი ბრძანებით:

**Setup.exe /q /ACTION=Repair
/INSTANCENAME=<ეგზემპლიარის_სახელი>**

MS SQL Server-ის კომპიუტერის გადარქმევა

2

თუ კომპიუტერს, რომელზეც არის ინსტალირებული SQL Server-ი, გადაერქვა სახელი, უნდა შესრულდეს შემდეგი პროცედურები:

```
EXEC sp_dropserver '<ძველი_სახელი>';
GO
EXEC sp_addserver '<ახალი_სახელი>', local;
GO
```

რის შემდეგაც SQL Server-ი უნდა გადაიტვირთოს.

თუ კომპიუტერს, რომელზეც არის ინსტალირებული SQL Server-ის სახელობითი ეგზემპლიარი, გადაერქვა სახელი, უნდა შესრულდეს შემდეგი პროცედურები:

```
EXEC sp_dropserver '<ძველი_სახელი\ეგზემპლიარის_სახელი>';
GO
EXEC sp_addserver '<ახალი_სახელი\ეგზემპლიარის_სახელი>', local;
GO
```

რის შემდეგაც SQL Server-ი უნდა გადაიტვირთოს.

შემდეგი ფუნქცია ამოწმებს სერვერის სახელს

```
SELECT @@SERVERNAME AS 'სერვერის_სახელი'
```

MS SQL Server-ის ინსტალირება SysPrep-ით

3

MS SQL Server-ის ინსტალირება SysPrep-ის გამოყენებით.

MS SQL Server-ი შესაძლებელია დაინსტალირდეს ასევე SysPrep-ის გამოყენებით.

SysPrep-ი იძლევა SQL Server-ის ცალკეული ეგზემპლიარის მომზადების შესაძლებლობას, რომლის გამოყენებით მოგვიანებით სხვადასხვა კომპიუტერზე შესაძლებელია კონფიგურირების დასრულება. ეს მეთოდი მოიცავს ორეტაპიან პროცესს:

- **იმიჯის მომზადება** - ამ ეტაპზე ორობითი ფაილების დაყენების შემდგომ, კომპიუტერის, ქსელის ან ინფორმაციის კონფიგურირების გარეშე ინსტალირების პროცესი ჩერდება, რომელიც მიბმულია მხოლოდ SQL Server-ის მოსამზადებელ ეგზემპლიარზე;
- **იმიჯის შექმნის დასრულება** - ამ ეტაპზე სრულდება SQL Server-ის მომზადებელი ეგზემპლიარის კონფიგურირება, რომლის დროსაც ეთითება ინფორმაცია კონკრეტული კომპიუტერის, ქსელის ან სააღრიცხვო ჩანაწერის.

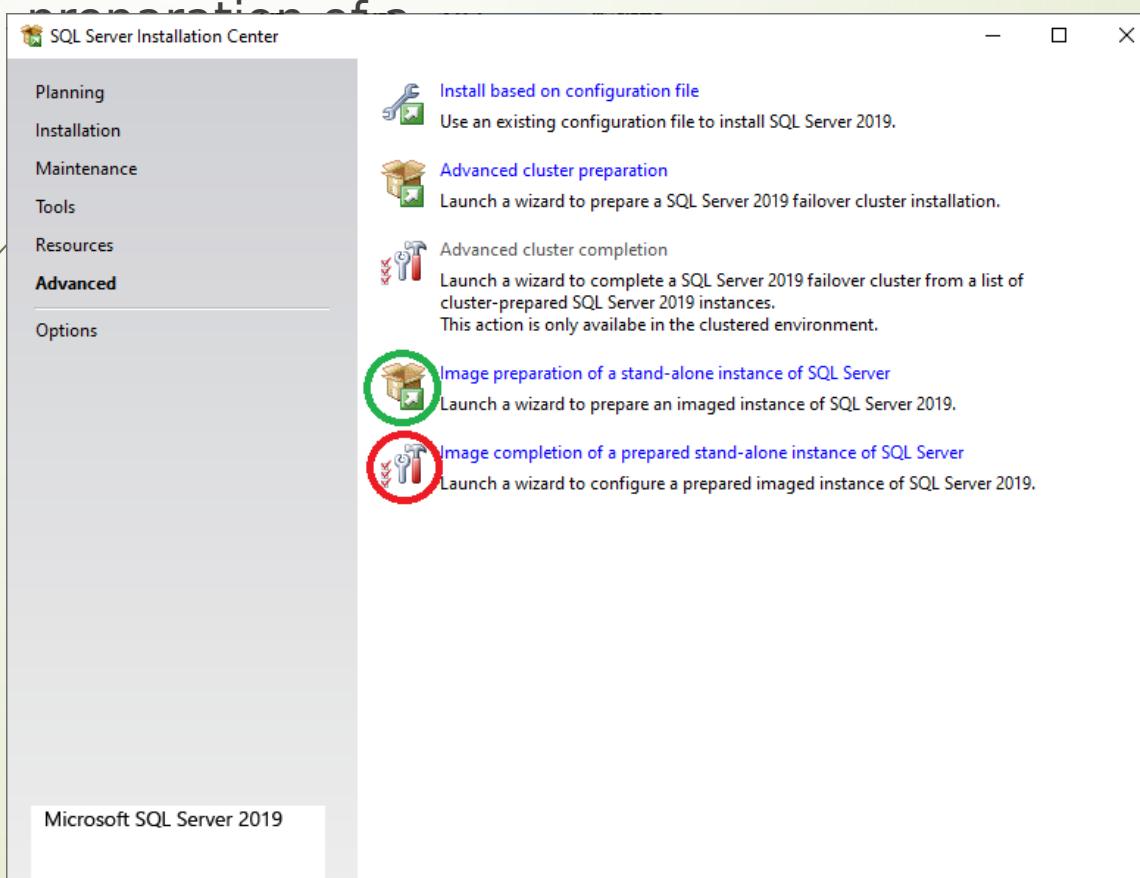
ძირითადად SQL Server SysPrep-ი გამოიყენება შემდეგ შემთხვევებში:

- წინასწარ იქნეს მომზადებული ერთი ან რამდენიმე არაკონფიგურირებული იმიჯი, რომლის კონფიგურირების დასრულება განხორციელდება სხვადასხვა კომპიუტერზე;
- Windows SysPrep-იმ Windows-ის იმიჯის მომზაობისას მიუკროოს SQL Server

MS SQL Server-ის ინსტალირება SysPrep-ით

4

SQL Server SysPrep-ის იმიჯის შესაქმნელად MS SQL Server-ის ინსტალირებისას setup.exe-ს გაშვებისას გამოჩენილ დიალოგურ ფანჯრის მარცხენა მიღამოში მდებარე სიაში უნდა გააქტიურდეს ბოლოდან მეორე სტრიქონი Advanced, რის შემდეგაც მარჯვენა მიღამოში უნდა გააქტიურდეს ბოლოდან მეორე სტრიქონი Image.



stand-alone instance of SQL Server (მწვანე წრეში), ხოლო SQL Server SysPrep-ის შექმნილი იმიჯის კონფი-გურირების დასრულებისათვის და MS SQL Server-ის ინსტალირებისა-თვის იგივე დიალოგურ ფანჯრის მარჯვენა მიღამოში უნდა გააქტიურ-დეს ბოლო სტრიქონი Image completion of a prepared stand-alone instance of SQL Server (წითელ წრეში). სტრიქონების გააქტიურების შემდგომი მოქმედებები უკვე აღწერილია წინა სლაიდებში, გარდა იმ მოქმედებებისა,

MS SQL Server-ის ინსტალირება SysPrep-ით

5

SQL Server SysPrep-ის იმიჯის შექმნის დიალოგური ფანჯრის განხვავებული გვერდები:

- SQL Server SysPrep-ის იმიჯის შექმნისას დიალოგური ფანჯრის Prepare Image Type გვერდში უნდა მოინიშნოს Prepare a new instance of SQL Server. გასათვალისწინებელია, რომ აღნიშნული გვერდი იქნება ნაჩვენები მხოლოდ მაშინ, როდესაც მომზადება გამოსახულება ტიპი გვერდზე არის ნაჩვენები მხოლოდ მაშინ, როდესაც მიმდინარე კომპიუტერში არსებობს არაკონფიგურირებული მომზადებული SQL Server ეგზემპლიარი. შესაძლებელია მომზადებულ იქნეს SQL Server-ის ახალი ეგზემპლიარი ან არსებულ ეგზემპლიარს დაემატოს SysPrep-ის მხარდაჭერილი კომპონენტები;
- SQL Server SysPrep-ის იმიჯის შექმნისას დიალოგურ ფანჯრის Prepare Image Rules გვერდზე კონფიგურაცია ამონმებს კომპიუტერის სისტემის მდგომარეობას იმიჯის შექმნის წესებთან მიმართებაში, რის შემდეგაც გრძელდება იმიჯის შექმნა;
- SQL Server SysPrep-ის იმიჯის შექმნისას დიალოგურ ფანჯრის Instance Configuration გვერდზე უნდა იქნეს მითითებული ეგზემპლიარის ID. აქვე შესაძლებელია მითითებულ იქნეს იმიჯის ძირითადი საქაღალდე (Instance root directory).

MS SQL Server-ის ინსტალირება SysPrep-ით

6

SQL Server SysPrep-ის შექმნილი იმიჯის კონფიგურირების დასრულების დიალოგური ფანჯრის განსხვავებული გვერდები:

- SQL Server SysPrep-ის შექმნილი იმიჯის კონფიგურირების დასრულების დიალოგური ფანჯრის „მხარდამჭერი ფაილების დაყენების“ (Setup Support Files) გვერდზე უნდა გააქტიურდეს ღილაკი Install ამ ფაილების ინსტალირებისათვის;
- SQL Server SysPrep-ის შექმნილი იმიჯის კონფიგურირების დასრულების დიალოგური ფანჯრის „მომზადებული ეგზემპლიარის შერჩევა“ (Select a Prepared Instance) ჩამოსაშლელ სიიდან უნდა იქნეს შერჩეული ის ეგზემპლიარი, რომლის დასრულებაც არის გათვალისწინებული.

MS SQL Server-ის ინსტალირება SysPrep-ით

7

SQL Server-ის გამზადებული ეგზემპლიარისათვის კომპონენტების დამატება

SQL Server-ის გამზადებული ეგზემპლიარისათვის კომპონენტების დასამატებლად უნდა განხორციელდეს შემდეგი განსხვავებული მოქმედებები:

- MS SQL Server-ის ინსტალირებისას setup.exe-ს გაშვებისას გამოჩენილ დიალოგურ ფანჯრის მარცხენა მიღამოში მდებარე სიაში უნდა გააქტიურდეს ბოლოდან მეორე სტრიქონი Advanced, რის შემდეგაც მარჯვენა მიღამოში უნდა გააქტიურდეს ბოლოდან მეორე სტრიქონი Image preparation of a stand-alone instance;
- SQL Server-ის გამზადებული ეგზემპლიარისათვის კომპონენტების დამატების დიალოგური ფანჯრის Prepare Image Type გვერდში უნდა მოინიშნოს Add features to an existing prepared instance of SQL Server. ჩამოსაშლელ სიიდან უნდა იქნეს შერჩეული ის ეგზემპლიარი, რომლშიც კომპონენტები დამატება არის გათვალისწინებული;
- SQL Server-ის გამზადებული ეგზემპლიარისათვის კომპონენტების დამატების დიალოგური ფანჯრის Feature Selection გვერდის გამოყენებით უნდა მოინიშნოს დასამატებელი კომპონენტები.

MS SQL Server-ის ინსტალირება SysPrep-ით

8

SQL Server-ის გამზადებული ეგზემპლიარიდან კომპონენტების წაშლა

SQL Server-ის გამზადებული ეგზემპლიარისათვის კომპონენტების წასაშლელად უნდა განხორციელდეს შემდეგი განსხვავებული მოქმედებები:

- Windows-ის ოპერაციული გარემოს მართვის პანელში (Control Panel) უნდა გააქტიურდეს „პროგრამები და კომპონენტები“ (Program and Features), რომელშიც უნდა მიეთითოს SQL Server-ის სისტემიდან „წაშლა“ (Remove);
- გამოჩენილ დიალოგურ ფანჯრის „ეგზემპლიარის შერჩევის“ (Select Instance) გვერდზე აირჩიეთ SQL Server-ის მომზადებული ეგზემპლიარი, რომლიდანაც უნდა განხორციელდეს კომპონენტების წაშლა;
- გამოჩენილ დიალოგურ ფანჯრის „კომპონენტების შერჩევის“ (Select Features) გვერდზე აირჩიეთ კომპონენტები, რომლებიც უნდა წაიშალოს SQL Server-ის მომზადებულ ეგზემპლიარიდან.

MS SQL Server-ის ინსტალირება SysPrep-ით

9

SQL Server-ის მტყუნებამედეგი კლასტერის მომზადება (დამოუკიდებლად)

SQL Server SysPrep-ის იმიჯის შექმნა Windows SysPrep-ის Windows-ის იმიჯში კომბინაციით, შესაძლებელია ასევე Windows PowerShell-ის ბრძანებითი სტრიქონის გამოყენებით, ამისათვის საჭიროა შემდეგი მოქმედები:

- უნდა შეიქნას SQL Server SysPrep-ის იმიჯი შემდეგი მაგალითის შესაბამისად:
**Setup.exe /q /ACTION=PrepareImage I /FEATURES=SQLEngine
/InstanceId =<MYINST> /IACCEPTSQLSERVICETERMS**
- უნდა განთავსდეს SQL Server SysPrep-ის იმიჯი Windows SysPrep Specialize-ს გამოყენებით;
- უნდა შევიდეთ Windows Failover Cluster-ში;
- ყველა კვანძისათვის უნდა გაეშვას MS SQL Server-ის საინსტალაციო ფაილი setup.exe პარამეტრით /ACTION = PrepareFailoverCluster შემდეგი მაგალითის შესაბამისად:
**setup.exe /q /ACTION=PrepareFailoverCluster
/InstanceName=<ეგზემპლიარის_სახელი> /Features=SQLEngine
/SQLSVCACCOUNT="<დომეინის_სახელი\მომხმარებლის_სახელი>"
/SQLSVCPASSWORD="*****"
/IACCEPTSQLSERVICETERMS**

MS SQL Server-ის ინსტალირება SysPrep-ით

10

SQL Server-ის მტყუნებამედეგი კლასტერის შექმნა (ავტომატური ინსტალირებით)

უნდა გაეშვას MS SQL Server-ის საინსტალაციო ფაილი setup.exe პარამეტრით /ACTION = CompleteFailoverCluster კვანძზე, რომელიც ეკუთვნის მოწყვლად საცავ ჯგუფს შემდეგი მაგალითის შესაბამისად:

```
setup.exe /q /ACTION=CompleteFailoverCluster
/InstanceName=<ეგზემპლიარის_სახელი>
/FAILOVERCLUSTERDISKS="<დისკის_სახელი მაგალითად, 'Disk S:'>:"
/FAILOVERCLUSTERNAME="<ქსელის_სახელი>"
/FAILOVERCLUSTERIPADDRESSES="IPv4;xx.xxx.xx.xx;Cluster
Network;xxx.xxx.xxx.x" /FAILOVERCLUSTERGEROUP="„ჯგუფის_სახელი"
/INSTALLSQLDATADIR="<Drive>:\<Path>\საქაღალდეს_სახელი"
/SQLCOLLATION="„ენის_კოდირება მაგალითად,
SQL_Latin1_General_CP1_CS_AS"
/SQLSYSADMINACCOUNTS="<დომეინის_სახელი\მომხმარებლის_სახელი>"
```

MS SQL Server-ის ინსტალირება SysPrep-ით

11

SQL Server-ის მტყუნებამედეგი კლასტერს კვანძის დამატება (ავტომატური ინსტალირებით)

- უნდა განთავსდეს SQL Server SysPrep-ის იმიზი Windows SysPrep Specialize-ს გამოყენებით;
- უნდა შევიღეთ Windows Failover Cluster-ში;
- ყველა კვანძისათვის უნდა გაეშვას MS SQL Server-ის საინსტალაციო ფაილი setup.exe პარამეტრით /ACTION = AddNode შემდეგი მაგალითის შესაბამისად:
setup.exe /q /ACTION=AddNode
/InstanceName=<ეგზემპლიარის_სახელი> /Features=SQLEngine
/SQLSVCACCOUNT="<დომეინის_სახელი\მომხმარებლის_სახელი>"
/SQLSVCPASSWORD="***"**
/IACCEPTSQLSERVLICENSETERMS

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

12

Desired State Configuration-ის (DSC) გამოყენებით მომხმარებელს ეძლევა შესაძლებლობა შექმნას SQL Server-ის ერთი საინსტალაციო კონფიგურაციის შაბლონი და გამოიყენოს იგი ასიათასობით სერვერზე. ინსტალირებისას საჭირო იქნება კომპიუტერისაგან გამომდინარე კონფიგურაციის მხოლოდ რამდენიმე პარამეტრის შეცვლა.

მაგალითისათვის განვიხილოთ SQL Server 2017-ის დამოუკიდებელი ეგზემპლარის Windows Server 2016-ზე საწყის ინსტალირება DSC რესურსის SqlServerDsc-ის გამოყენებით, რისთვისაც საჭიროა:

- კომპიუტერი, რომელზეც ინსტალირებულია Windows Server 2016;
- SQL Server 2017 საინსტალაციო პაკეტი, რომელიც მაგალითისათვის მოვათავსოთ ღოვიცურ დისკზე:

C:\en_sql_server_2017_enterprise_x64_dvd_11293666.iso

- DSC-ის რესურსი SqlServerDsc. SqlServerDsc-ის საინსტალაციო პაკეტი მდებარეობს ვებ-გვერდზე <https://www.powershellgallery.com/packages/SqlServerDsc/15.2.0>, რომელიც გადმოწერის შემდეგ ინსტალირდება Windows-ის ბრძანებათა სტრიქონიდან

Install-Module -Name SqlServerDsc

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

13

SQL Server 2017 საინსტალაციო პაკეტი (ISO) მოვათავსოთ საქაღალდეში:

```
New-Item -Path C:\SQL2017 -ItemType Directory
$mountResult = Mount-DiskImage -ImagePath 'C:\en_sql_server_2017_enterprise_x64_dvd_11293666.iso' -PassThru
$volumeInfo = $mountResult | Get-Volume
$driveInfo = Get-PSDrive -Name $volumeInfo.DriveLetter
Copy-Item -Path ( Join-Path -Path $driveInfo.Root -ChildPath '*' ) -Destination C:\SQL2017\ -Recurse
Dismount-DiskImage -ImagePath 'C:\en_sql_server_2017_enterprise_x64_dvd_11293666.iso'
```

შევქმნათ კონფიგურაციის ფუნქცია, რომელიც გამოიძახება „მართული ობიექტის ფორმატის“ (Managed Object Format - MOF) დოკუმენტების შესაქმნელად :

Configuration SQLInstall

```
{...}
```

განვახორციელოთ მოდულების იმპორტი მიმდინარე სესიაში, რომლებიც მიუთითებენ კონფიგურაციის დოკუმენტს MOF დოკუმენტების შექმნის წესს, ასევე მიუთითებენ DSC ძრავას თუ როგორ იქნეს გამოყენებული MOF დოკუმენტები სერვერზე:

```
Import-DscResource -ModuleName SqlServerDsc
```

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

14

ასევე საჭიროა .NET Framework-45-Core-ს ინსტალირება SQL Server-ის ინსტალირებამდე, რისთვისაც გამოიყენება WindowsFeature რესურსი:

WindowsFeature 'NetFramework45'

```
{  
    Name = 'Net-Framework-45-Core'  
    Ensure = 'Present'  
}
```

SqlSetup რესურსი ატყობინებს DSC-ს თუ როგორ უნდა დაინსტალირდეს SQL Server-ი. საბაზისო ინსტალირებისათვის საჭიროა შემდეგი პარამეტრები:

- **InstanceName** - ეგზემპლიარის სახელი (ნაგულისხმევია MSSQLSERVER);
- **Features** - ინსტალირების კომპონენტები (ჩვენ მაგალითში არის მხოლოდ SQLEngine);
- **SourcePath** - გზა SQL Server-ის საინსტალაციო პაკეტთან (ჩვენ მაგალითში არის C:\SQL2017);
- **SQLSysAdminAccounts** - sysadmin როლის წევრი მომხმარებლები ან ჯგუფები (ჩვენ მაგალითში არის ლოკალური ჯგუფი Administrators).

SqlSetup რესურსი გამოიყენება მხოლოდ SQL Server-ის ინსტალირებისათვის და არა გამოიყენებული პარამეტრების შენახვისათვის.

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

15

კონფიგურაციის დასრულებისათვის PowerShell-ში იწერება ბრძანება:

Configuration SQLInstall

```
{  
    Import-DscResource -ModuleName SqlServerDsc  
    node localhost  
    {  
        WindowsFeature 'NetFramework45'  
        {  
            Name   = 'NET-Framework-45-Core'  
            Ensure = 'Present'  
        }  
        SqlSetup 'InstallDefaultInstance'  
        {  
            InstanceName      = 'MSSQLSERVER'  
            Features          = 'SQLENGINE'  
            SourcePath        = 'C:\SQL2017'  
            SQLSysAdminAccounts = @('Administrators')  
            DependsOn         = '[WindowsFeature]NetFramework45'  
        }  
    }  
}
```

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

კონფიგურაციის კომპილაციისათვის PowerShell-ში იწერება კონფიგურაციის სკრიპტის წყარო:

. .\SQLInstallConfiguration.ps1

და იგივე PowerShell-ში ეძვება კონფიგურაციის ფუნქცია:

SQLInstall

სამუშაო საქაღალდეში იქმნება SQLInstall საქაღალდე, რომელიც შეიცავს ფაილს `localhost.mof`. MOF ფაილში განთავსებულია კომპილირებული DSC-ს კონფიგურაცია.

SQL Server-ის ინსტალირებისათვის DSC-ს მიხედვით PowerShell-ში იწერება ბრძანება:

Start-DscConfiguration -Path C:\SQLInstall -Wait -Force -Verbose

აქ არის შემდეგი პარამეტრები:

- **Path** - გზა საქაღალდის, რომელშიც მდებარეობენ MOF საბუთები (ჩვენ მაგალითში არის `C:\SQLInstall`);
- **Wait** - კონფიგურაციის სამუშაოს დასრულების დალოდება;
- **Force** - არსებული DSC კონფიგურაციების გადანაწილება;
- **Verbose** - გამოსავალი მონაცემების ვიზუალიზაცია.

MS SQL Server-ის ინსტალირება PowerShell DSC-ით

17

ბრძანების გაშვების შემდგომ თუ არ არის შეცდომები (წითელი ტექსტი), ეკრანზე გამოისახება Operation 'Invoke CimMethod' Complete რაც ინსტალირების წარმატებით დასრულებას ნიშნავს.

ასვე წარმატებული ინსტალირება შეიძლება გადამოწმდეს PowerShell-ში ბრძანებით:

Test-DscConfiguration

რომელმაც უნდა დააბრუნოს პასუხი **True**.

ასევე სერვისების სიაში უკვე გამოისახება SQL Server-ის სერვისები, რომელიც შეიძლება ინახოს PowerShell-ში ბრძანებით:

Get-Service -Name *SQL*

შედეგად მივიღებთ სერვისების სიას:

**Running
(MSSQLSERVER)**

MSSQLSERVER

SQL Server

Stopped

SQLBrowser

SQL Server Browser

**Running
(MSSQLSERVER)**

SQLSERVERAGENT

SQL Server Agent

**Running
(MSSQLSERVER)**

SQLTELEMETRY

SQL Server CEIP service

MS SQL Server-ის ინსტალირების ჟურნალი

SQL Server-ი ჟურნალის ფაილებს ნაგულისხმევად ქმნის დათარიღებულ და დროში გაწერილ საქაღალდეში:

%programfiles%\Microsoft SQL Server\nnn\Setup Bootstrap\Log\YYYYMMDD_hhmmss

სადაც nnn არის რიცხვი, რომლებიც შეესაბამება ინსტალირებული SQL Server-ის ვერსიას, ხოლო YYYYMMDD_hhmmss ჟურნალის საქაღალდის სახელის ფორმატია.

ინსტალირებისას ავტომატურ რეჟიმში ხორციელდება ჟურნალების შექმნა %temp%\sqlsetup*.log-ში, რის შემდეგაც ამ საქაღალდეს ყველა ფაილი არქივდება შესაბამის საქაღალდის Log*.cab ფაილში.

Name	Date modified	Type	Name	Date modified	Type	Size
20180326_083544	3/26/2018 4:45 PM	File folder	Datastore	3/26/2018 9:08 AM	File folder	
20180326_083658	3/26/2018 8:37 AM	File folder	resources	3/26/2018 8:42 AM	File folder	
20180326_083952	3/26/2018 8:40 AM	File folder	AdvancedAnalytics_Cpu64_1.log	3/26/2018 8:57 AM	Text Document	258 KB
20180326_084157	3/26/2018 4:45 PM	File folder	ConfigurationFile.ini	3/26/2018 8:54 AM	Configuration sett...	20 KB
20180326_084208	3/26/2018 9:08 AM	File folder	Detail.txt	3/26/2018 9:08 AM	TXT File	4,146 KB
20180507_123534	5/7/2018 12:56 PM	File folder	HkEngineEventFile_0_13166554069583...	3/26/2018 9:07 AM	Microsoft SQL Ser...	69 KB
20180507_123541	5/7/2018 12:36 PM	File folder	HkEngineEventFile_0_13166554094530...	3/26/2018 9:08 AM	Microsoft SQL Ser...	69 KB
Summary.txt	5/7/2018 12:36 PM	TXT File	MDS_Cpu64_1.log	3/26/2018 8:58 AM	Text Document	2,732 KB

MS SQL Server-ის ინსტალირების ჟურნალი

SQL Server-ის ინსტალირების ჟურნალის ფაილებია:

%programfiles%\Microsoft SQL Server\nnn\Setup Bootstrap\Log საქაღალდეში:

- **Summary.txt**

%programfiles%\Microsoft SQL Server\nnn\Setup Bootstrap\Log\
YYYYMMDD_hhmmss საქაღალდეში:

- **Summary_<MachineName>_YYYYMMDD_hhmmss.txt**

- **Detail.txt**

- **Datastore**

- **MSI Log Files** (ფაილი <Name>.log)

- **ConfigurationFile.ini**

- **SystemConfigurationCheck_Report.htm**

%temp%\ საქაღალდის sqlsetup*.log ფაილი

- **For unattended installations**

განვიხილოთ ამ ფაილების დანიშნულებები:

MS SQL Server-ის ინსტალირების ჟურნალი

Summary.txt

ეს ფაილი აჩვენებს SQL Server-ის კომპონენტებს, რომლებიც გამოვლინდა ინსტალირების პროცესში, ოპერაციული სისტემის გარემო, ბრძანებათა სტრიქონის პარამეტრების მნიშვნელობები და თითოეული შესრულებული MSI/MSP-ს საერთო სტატუსი. ეს ყველაფერი დაყოფილია შემდეგ ნაწილებად:

- შესრულების საერთო ანგარიში;
- კომპიუტერის თვისებები და კონფიგურაცია, რომელზეც ინსტალირდება SQL Server-ი;
- კომპიუტერზე აღრე დაინსტალირებული SQL Server-ის მახასიათებლები;
- ინსტალირების პაკეტის თვისებებისა და ვერსიის აღწერა;
- ინსტალირების მითითებული შესრულების გარემოს შემავალი პარამეტრები;
- კონფიგურირების ფაილის აღგიღმდებარეობა;
- მონაცემები შესრულების შედეგებზე;
- გლობალური წესები;
- ინსტალირების სცენარის სპეციფიკური წესები;
- წესები, რომლის შესრულებისას წარმოიქმნა შეცდომა;
- წესების ანგარიშის ფაილის მდგბარეობა.

MS SQL Server-ის ინსტალირების ჟურნალი

Summary_<MachineName>_YYYYYYMMDD_hhmmss.txt

Summary.txt ძირითადი ფაილის მსგავსია და გენერირდება ძირითადი სამუშაო პროცესის შესრულებისას.

Detail.txt

Detail.txt გენერირდება ძირითადი მუშა პროცესის შესრულებისას, როგორიცაა ინსტალირება ან განახლება და უზრუნველყოფს შესრულების დეტალებს. ფაილების ჟურნალები გენერირდება დროის მიხედვით, როდესაც ინსტალირების თითოეული მოქმედება იქნა გამოძახილი. ამ ფაილში ნაჩვენებია ქმედებების შესრულების თანმიმდევრობა და მათი დამოკიდებულებები.

თუ ინსტალირებისას განხორციელდა შეცდომა, იგი რეგისტრირებული იქნება ამ ფაილის ბოლოში, რომელიც, ასევე, შეიძლება მოძიებულ იქნეს საკვანძო სიტყვებით error ან exception.

MS SQL Server-ის ინსტალირების ჟურნალი

MSI Log Files

ამ ფაილებში არის ინფორმაცია პაკეტების პროცესის ინსტალირების შესახებ. ისინი გენერირდებიან MSIEXEC-ის მიერ გარკვეული პაკეტების ინსტალირების დროს. MSI ჟურნალების ფორმატი შემდეგია:

**<Feature>_<Architecture>_<Interaction>.log
<Feature>_<Architecture>_<Language>_<Interaction>.log
<Feature>_<Architecture>_<Interaction>_<workflow>.log**

ფაილის ბოლოს არის ჯამური ინფორმაცია შესრულებაზე, რომელშიც მითითებულია საერთო მდგომარეობა (წარმატება ან წარუმატებლობა) და თვისებები. MSI ფაილში შეცდომის მოძიება უნდა განხორციელდეს მნიშვნელობით " value 3" და ინახოს მასთან მდებარე ტექსტი.

ConfigurationFile.ini

კონფიგურაციის ფაილი შეიცავს ინსტალირების შემავალ პარამეტრებს, რომელიც შეიძლება გამოყენებულ იქნეს განმეორებით ინსტალირებისას (გარდა ანგარიშების პაროლებისა, PID-ისა და ზოგიერთ პარამეტრებისა, რომლებიც კონფიგურაციის ფაილში არ ინახება).

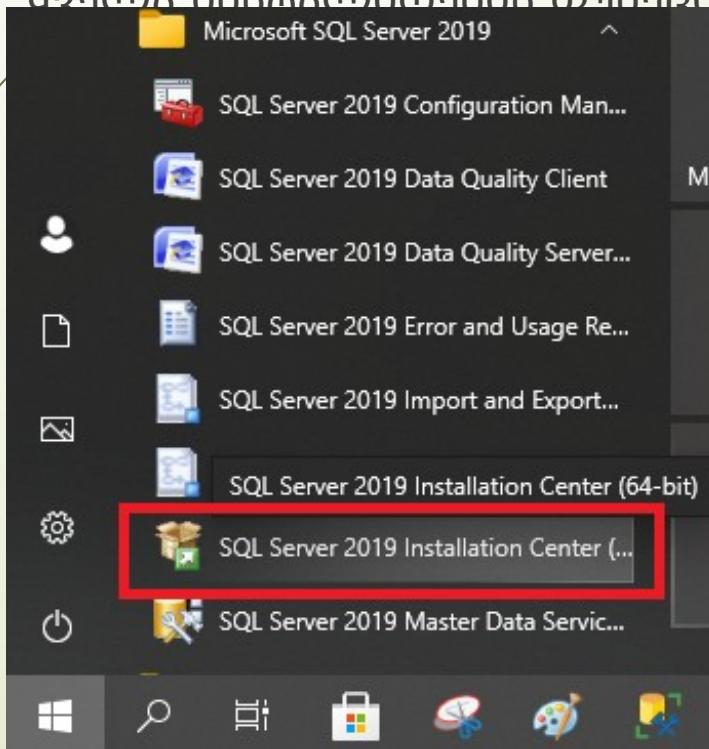
SystemConfigurationCheck_Report.htm

ყველა წესის მოკლე აღწერა და შესრულების სტატუსი.

MS SQL Server-ის ინსტალირების ცენტრი

23

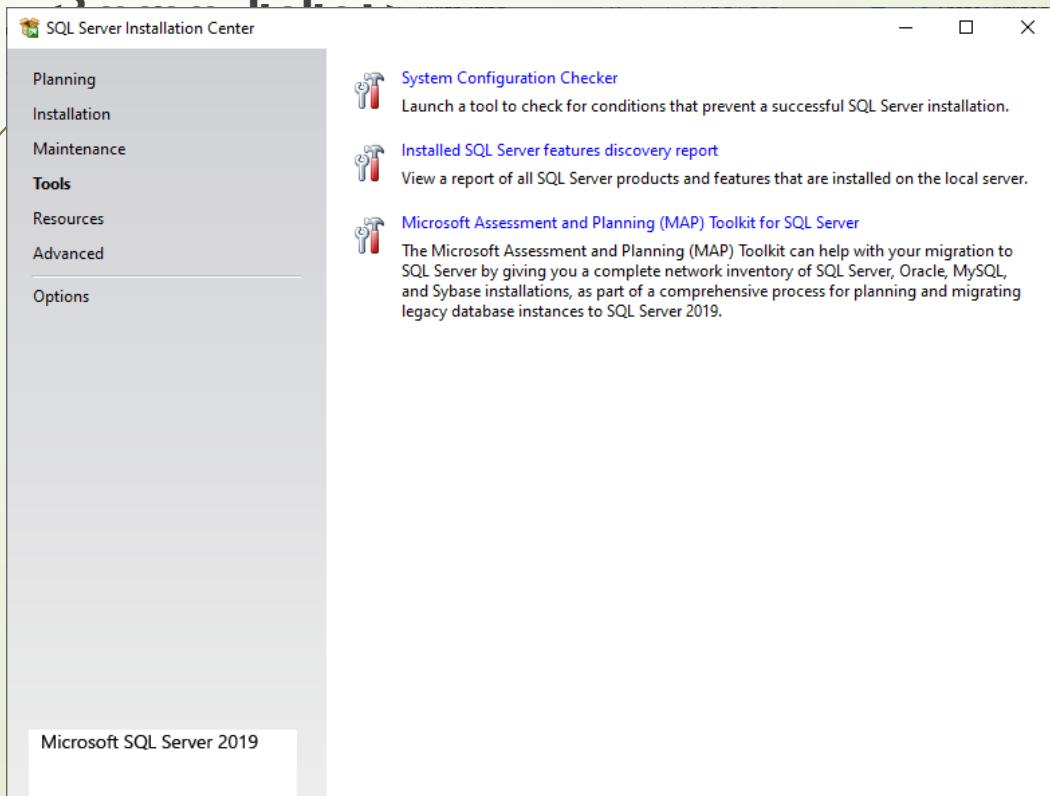
კომპიუტერში ინსტალირებულ SQL Server-ის ინსტალირების ცენტრის (დიალოგური ფანჯრის) გამოსაძახებლად მომხმარებელმა Start მენიუს გამოყენებით უნდა შევიდეს Microsoft SQL Server <ვერსიის სახელი> (ჩვენ შემთხვევაში Microsoft SQL Server 2019) და ახლად ჩამოშლილ სიაში უნდა გააქტიუროს სტრიქონი SQL Server <ვერსიის სახელი> Installation Center (ჩვენ შემთხვევაში SQL Server 2019 Installation Center), რის შემდეგაც ეკრანზე გამოისახება ინსტალირების სტანდარტული დიალოგური ფანჯარა, რომელშიც შესაძლებელია SQL Server-ის აუთა ინსტალირების თუნდრის გაშვება.



MS SQL Server-ის ინსტალირების შემოწმება

SQL Server-ის ინსტალირების ცენტრის დიალოგურ ფანჯარაში მარცხენა მიღამოში მდებარე მენიუში უნდა მოინიშნოს მეოთხე სტრიქონი Tools, რის შემდეგაც მარჯვენა მიღამოში გააქტიურდეს მეორე სტრიქონი „ინსტალირებული SQL Server-ის კომპონენტების აღმოჩენის ანგარიში“ (Installed SQL Server features discovery report), რის შემდეგაც ანგარიში ინახება საქაღალდეში:

%ProgramFiles%\Microsoft\SQL Server\nnn\Setup Bootstrap\Log



ზემოაღნიშნული ანგარიში
შესაძლებელია შეიქნას ასევე
PowerShell-ის ბრძანებათა სტრიქონის
გამოყენებით, რისთვისაც
მომხმარებელმა უნდა გაუშვას
ბრძანება:

Setup.exe /Action = RunDiscovery

თუ დამატებით გამოყენებულ იქნება
პარამეტრი „/q“, შედეგი არ იქნება
ასახული, მაგრამ ფაილი მაინც
შეიქმნება ზემოაღნიშნულ
საქაღალდეში.

MS SQL Server-ის ინსტალირება Windows Server 2016 Core და Windows Server 2019 Core-ში

Windows Server 2016 Core და Windows Server 2019 Core-ში ინსტალირებისას პარამეტრების მითითებებს აქვთ შემდეგი დანიშნულება:

- **/Q** - სრული ჩუმი რეჟიმი შეტყობინებების გამოტანის გარეშე;
- **/QS** - მარტივი რეჟიმი შეტყობინებების გამოტანის გარეშე;
- **/IACCEPTSQLSERVERLICENSETERMS** - ლიცენზირების პირობებთან დათანხმება;
- **/FEATURES** - დამატებითი პარამეტრების მისათითებლად:
 - **/FEATURES=SQLENGINE** - დაინსტალირდება მხოლოდ Database Engine კომპონენტი;
 - **/FEATURES=REPLICATION** - დაინსტალირდება რეპლიკაციის კომპონენტი Database Engine-თან ერთად;
 - **/FEATURES=FULLTEXT** - დაინსტალირდება Full-Text Search კომპონენტი Database Engine-თან ერთად;
 - **/FEATURES=AS** - დაინსტალირდება Analysis Services-ის სამსახურის ყველა კომპონენტი;
 - **/FEATURES=IS** - დაინსტალირდება Integration Services-ის სამსახურის ყველა კომპონენტი;
 - **/FEATURES=CONN** - უძრავი კონფიგურირების მიზანისას კონფიგურირების დანართი.

MS SQL Server-ის ინსტალირება Windows Server 2016 Core და Windows Server 2019 Core-ში

მოვიყვანოთ Windows Server 2016 Core-ში ან Windows Server 2019 Core-ში ინსტალირებისას პარამეტრების მითითებების

მაგალითი:

```
Setup.exe /qs  
/ACTION=Install  
/FEATURES=SQLEngine, Replication  
/INSTANCENAME=MSSQLSERVER  
/SQLSVCCOUNT="<დომენის_სახელი\მომხმარებლის_სახელი>"  
/SQLSVCPASSWORD="<მძღავრი_პაროლი>"  
/SQLSYSADMINACCOUNTS="< დომენის_სახელი\მომხმარებლის_სახელი >"  
/AGTSVCCOUNT="NT AUTHORITY\Network Service"  
/TCPENABLED=1  
/IACCEPTSQLSERVLICENSETERMS
```

MS SQL Server-ის ინსტალირება Windows Server 2016 Core და Windows Server 2019 Core-ში

ასევე შესაძლებელია ინსტალირება განხორციელდეს კონფიგურაციის ფაილის (ტექსტური ფაილის INI გაფართოების) გამოყენებით, რომელიც შეიცავს პარამეტრებს (ანუ წყვილებს „სახელი-მნიშვნელობა“) და კომენტარებს აღნერით, რომლის დროსაც ინსტალირების განსახორციელებლად ბრძანების სტრიქონში უნდა დაიწეროს:

Setup.exe /QS /ConfigurationFile=MyConfigurationFile.INI

მოვიყვანოთ კონფიგურაციის ფაილით Database Engine-ს კომპონენტის ინსტალირების

მაგალითი:

[OPTIONS]

ACTION="Install"

FEATURES=SQLENGINE

INSTANCENAME="MSSQLSERVER"

INSTANCEID="MSSQLSERVER"

SQLSVACCOUNT="NT Service\MSSQLSERVER"

SQLSYSADMINACCOUNTS="\< დომენის_სახელი\მომხმარებლის_სახელი >"

IAcceptSQLServerLicenseTerms="True"

MS SQL Server-ის ინსტალირება Windows Server 2016 Core და Windows Server 2019 Core-ში

ასევე, მოვიყვანოთ კონფიგურაციის ფაილით მონაცემებთან მიერთების კომპონენტის ინსტალირების

ძაგლითი:

[OPTIONS]

ACTION="Install"

FEATURES=Conn

IAcceptSQLServerLicenseTerms="True"

იმისათვის, რომ დაშვებულ იქნეს გარე მიერთებები SQL Server-ზე, უნდა განხორციელდეს შემდეგი ინსტრუქციები:

EXEC sys.sp_configure N'remote access', N'1'

GO

RECONFIGURE WITH OVERRIDE

GO

MS SQL Server-ის ინსტალირება Windows Server 2016 Core და Windows Server 2019 Core-ში

SQL Server-ში ბრაუზერის ჩასართავად (ავტომატურად იგი ინსტალირებისას გამორთულია), უნდა განხორციელდეს შემდეგი ინსტრუქცია:

Set-service sqlbrowser -StartupType Auto

რის შემდეგაც ბრაუზერის გასაშვებად საჭიროა შემდეგი ინსტრუქცია:

Start-service sqlbrowser

SQL Server-ში TCP/IP ოქმის მხარდაჭერის ჩასართავად, უნდა განხორციელდეს შემდეგი:

- PowerShell-ში გაუშვათ Import-Module SQLPS;
- SQL Server-ის PowerShell-ში გაუშვათ შემდეგი ინსტრუქცია:

```
$smo = 'Microsoft.SqlServer.Management.Smo.'  
$wmi = new-object ($smo + 'Wmi.ManagedComputer')  
$uri = "ManagedComputer[@Name=''" + (get-item env:\  
computername).Value +  
"']/ServerInstance[@Name='MSSQLSERVER']/ServerProtocol[@Name='Tcp'  
'],  
$Tcp = $wmi.GetSmoObject($uri)  
$Tcp.IsEnabled = $true  
$Tcp.Alter()
```

SQL Server-ის რესურსების შეზღუდვები

1

მონაცემთა ბაზის ძრავის ობიექტები

პაკეტის ზომა - 65536 ბაიტი (ქსელის პაკეტის ზომა).

სტრიქონის სიგრძე, რომელიც შეიცავს Transact-SQL ინსტრუქციებს (პაკეტის ზომა) - 65536 ბაიტი (ქსელის პაკეტის ზომა).

ორივე შემთხვევაში ქსელის პაკეტის ზომა არის ცხრილის მონაცემთა ნაკადის (TDS) პაკეტების ზომა, რომელიც გამოიყენება პროგრამებსა და მონაცემთა ბაზის ძრავას შორის ურთიერთობისათვის. ნაგულისხმევი პაკეტის ზომაა 4 KB და კონტროლდება ქსელის პაკეტის ზომის კონფიგურაციის პარამეტრით.

მოკლე სტრიქონის სვეტი - 8000 ბაიტი.

GROUP BY და **ORDER BY** – 8060 ბაიტი.

ინდექსის გასაღები - 900 ბაიტი კლასტერული ინდექსისთვის, 1700 ბაიტი არაკლასტირებული ინდექსისთვის.

ინდექსის გასაღები მეხსიერებაში ოპტიმიზირებულ ცხრილებისთვის - 2500 ბაიტი არაკლასტირებული ინდექსისთვის.

(გაგრძელება შემდეგ სლაიდზე)

SQL Server-ის რესურსების შეზღუდვები

2

გარე გასაღების ზომა - 900 ბაიტი.

პირველადი გასაღების ზომა - 900 ბაიტი.

სტრიქონის ზომა - 8 060 ბაიტი.

სტრიქონის ზომა მეხსიერებაში ოპტიმიზირებულ ცხრილებისთვის - 8 600 ბაიტი.

შენახული პროცედურის ტექსტის ზომა - პაკეტის ზომაზე ნაკლები ან 250 მბ.

varchar(max), varbinary(max), xml, text, ან image სვეტის ზომა - $2^{31}-1$ ბაიტი.

ntext ან nvarchar(max) სვეტის ზომა - $2^{30}-1$ სიმბოლო.

კლასტერული ინდექსების რაოდენობა თითო ცხრილში - 1.

სვეტების რაოდენობა **GROUP BY**-ში ან **ORDER BY**-ში - შეზღუდულია მხოლოდ ბაიტებით.

სვეტების ან გამოსახულებების რაოდენობა **GROUP BY WITH CUBE ან WITH ROLLUP** ინსტრუქციებში - 10.

ინდექსის გასაღებში სვეტების რაოდენობა - 32.

გარე ან პირველად გასაღებში სვეტების რაოდენობა - 32.

(გაგრძელება შემდეგ სლაიდზე)

SQL Server-ის რესურსების შეზღუდვები

3

INSERT ინსტრუქციაში სვეტების რაოდენობა - 4 096.

SELECT ინსტრუქციაში სვეტების რაოდენობა - 4 096.

UPDATE ინსტრუქციაში სვეტების რაოდენობა - 4 096.

მონაცემთა ცხრილში სვეტების რაოდენობა - 1 024.

წარმოდგენაში სვეტების რაოდენობა - 1 024.

მიერთებების რაოდენობა ერთ კლიენტთან - მიერთებების კონფიგურირების მაქსიმალური
მნიშვნელობა.

მონაცემთა ბაზის ზომა - 524 272 ტერაბაიტი.

SQL Server ერთ ეგზემპლიარში მონაცემთა ბაზების რაოდენობა - 32 767.

ფაილური ჯგუფების რაოდენობა ერთ მონაცემთა ბაზაში - 32 767.

მეხსიერებისათვის ოპტიმიზირებული ფაილური ჯგუფების რაოდენობა მონაცემებისთვის
ერთ მონაცემთა ბაზაზე - 1.

ფაილების რაოდენობა ერთ მონაცემთა ბაზაში - 32 767.

(გაგრძელება შემდეგ სლაიდზე)

SQL Server-ის რესურსების შეზღუდვები

4

მონაცემთა ფაილის ზომა - 16 ტერაბაიტი.

ჟურნალის ფაილის ზომა - 2 ტერაბაიტი.

გარე გასაღების ბმულების რაოდენობა ერთ ცხრილისათვის - გამავალი 253, ხოლო შემომავალი 10 000.

იდენტიფიკატორის სიგრძის ზომა - 128 სიმბოლო.

ეგზემპლიარების რაოდენობა თითო კომპიუტერზე - 50.

ინდექსების რაოდენობა მეხსიერებისათვის ოპტიმიზირებულ ცხრილში - 999.

ბლოკირებების რაოდენობა ერთ მიერთებაზე - მაქსიმალური ბლოკირების რიცხვი სერვერზე.

ბლოკირებების რაოდენობა ერთ SQL Server-ის ეგზემპლიარზე - შეზღუდულია მხოლოდ მეხსიერების მოცულობით.

ჩადგმული შენახული პროცედურების დონეების რაოდენობა - 32.

ჩადგმული მოთხოვნების დონეების რაოდენობა - 32.

(გაგრძელება შემდეგ სლაიდზე)

SQL Server-ის რესურსების შეზღუდვები

5

ჩადგმული ტრანზაქციების დონეების რაოდენობა - 4 294 967 296.

ჩადგმული ტრიგერების დონეების რაოდენობა - 32.

მონაცემთა ცხრილში არაკლასტირებული ინდექსების რაოდენობა - 999.

შენახული პროცედურის პარამეტრების რაოდენობა - 2100.

მომხმარებლის ფუნქციის პარამეტრების რაოდენობა - 2100.

მონაცემთა ცხრილში მითითებების რაოდენობა - 253.

მონაცემთა ცხრილში სტრიქონების რაოდენობა - შეზღუდულია ხელმისაწვდომი
მეხსიერებით.

მონაცემთა ბაზაში მონაცემთა ცხრილების, წარმოდგენების, შენახული პროცედურების,
მომხმარებლის ფუნქციების, ტრიგერების, წესების, ნაგულისხმევ მნიშვნელობებისა და
შეზღუდვების ჯამური რაოდენობა - 2147483647.

მომხმარებელთა მიერთებების რაოდენობა - 32 767.

XML ინდექსების რაოდენობა - 249.

SQL Server-ის რესურსების შეზღუდვები

SQL Server-ის უტილიტების ობიექტები

SQL Server-ის უტილიტებში კომპიუტერების რაოდენობა (ფიზიკური ან ვირტუალური მანქანები) რაოდენობა - 100.

SQL Server-ის ეგზემპლიარების რაოდენობა კომპიუტერში - 5.

SQL Server-ის უტილიტებში ეგზემპლიარების რაოდენობა - 200.

SQL Server-ის ეგზემპლიარში მომხმარებლის მონაცემთა ბაზების რაოდენობა - 50.

SQL Server-ის უტილიტებში მომხმარებლის მონაცემთა ბაზების რაოდენობა - 1 000.

ფაილური ჯგუფების რაოდენობა მონაცემთა ბაზაში - 1.

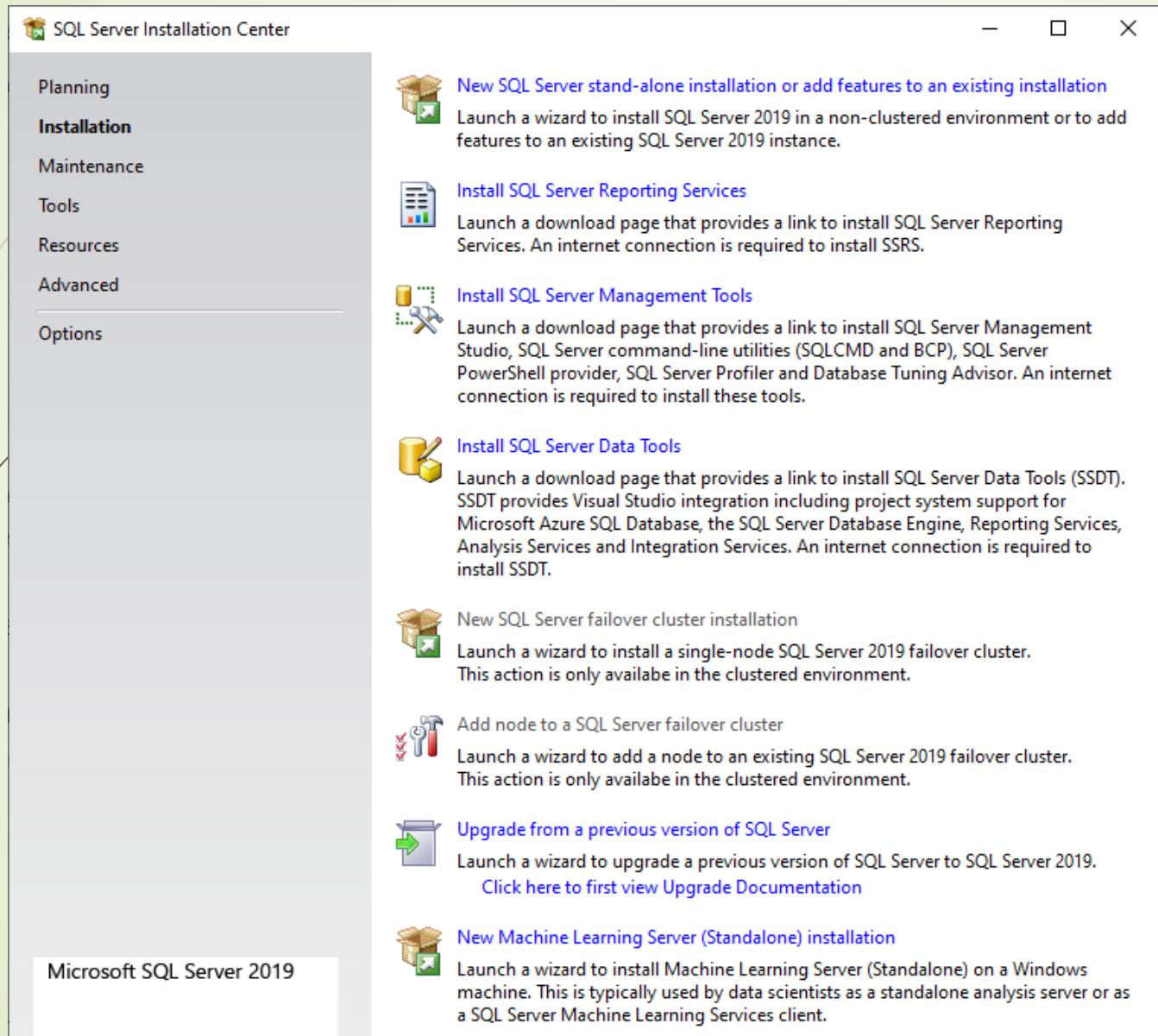
მონაცემთა ფაილების რაოდენობა ფაილურ ჯგუფში - 1.

ჟურნალების ფაილების რაოდენობა მონაცემთა ბაზაში - 1.

ტომების რაოდენობა კომპიუტერში - 3.

MS SQL Server-ის განახლება

7



SQL Server-ის განახლება (Update) ავტომატურად ხორციელდება Windows-ის განახლებისას, მაგრამ წინა ვერსიის უფრო მაღალ ვერსიამდე განახლება (Upgrade) შესაძლებელია განხორციელდეს ინსტალირების ცენტრის დიალოგური ფანჯრის გამოყენებით. ამისათვის მარცხენა მიდამოში მდებარე მენიუში უნდა მოინიშნოს მეორე სტრიქონი Installation, რის შემდეგაც მარჯვენა მიდამოში გააქტიურდეს ქვემოდან მეორე სტრიქონი Upgrade from previous versions of SQL Server.

MS SQL Server-ის განახლება

8

პროდუქტის გასაღების (Product Key) გვერდზე მომხმარებელმა უნდა მიუთითოს განახლება ხორციელდება SQL Server-ის უფასო გამოცემამდე, თუ მას გააჩნია PID გასაღები პროდუქტის მუშა ვერსიისათვის.

ლიცენზიის პირობების (License Terms) გვერდზე მომხმარებელმა უნდა გადახედოს სალიცენზიო ხელშეკრულებას და თუ თანახმა არის, მონიშნოს თანხმობა (I accept the license terms).

გლობალური წესების (Global Rules) ფანჯარაში, ინსტალირების პროცედურა ავტომატურად გადავა პროდუქტის განახლებების ფანჯარაში, თუ არ გამოიკვეთა შეცდომები წესებში.

Microsoft-ის განახლების (Microsoft Update) გვერდი გამოჩნდება, თუ Windows-ში Control Panel\All Control Panel Items\Windows Update\Change settings-ში მონიშნულია Microsoft Update ველი.

პროდუქტის განახლებების (Product Updates) გვერდზე ნაჩვენები იქნება უახლესი ხელმისაწვდომი SQL Server პროდუქტის განახლებები. თუ პროდუქტის განახლებები არ არის აღმოჩენილი, გვერდი არ გამოისახება და ავტომატურად გადავა ინსტალირების ფაილების (Install Setup Files) გვერდზე.

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის განახლება

9

ინსტალირების ფაილების (Install Setup Files) გვერდზე, შემოთავაზებულ იქნება ფაილების გადმოტვირთვის პროგრესი, მოპოვებისა და ინსტალაციისთვის.

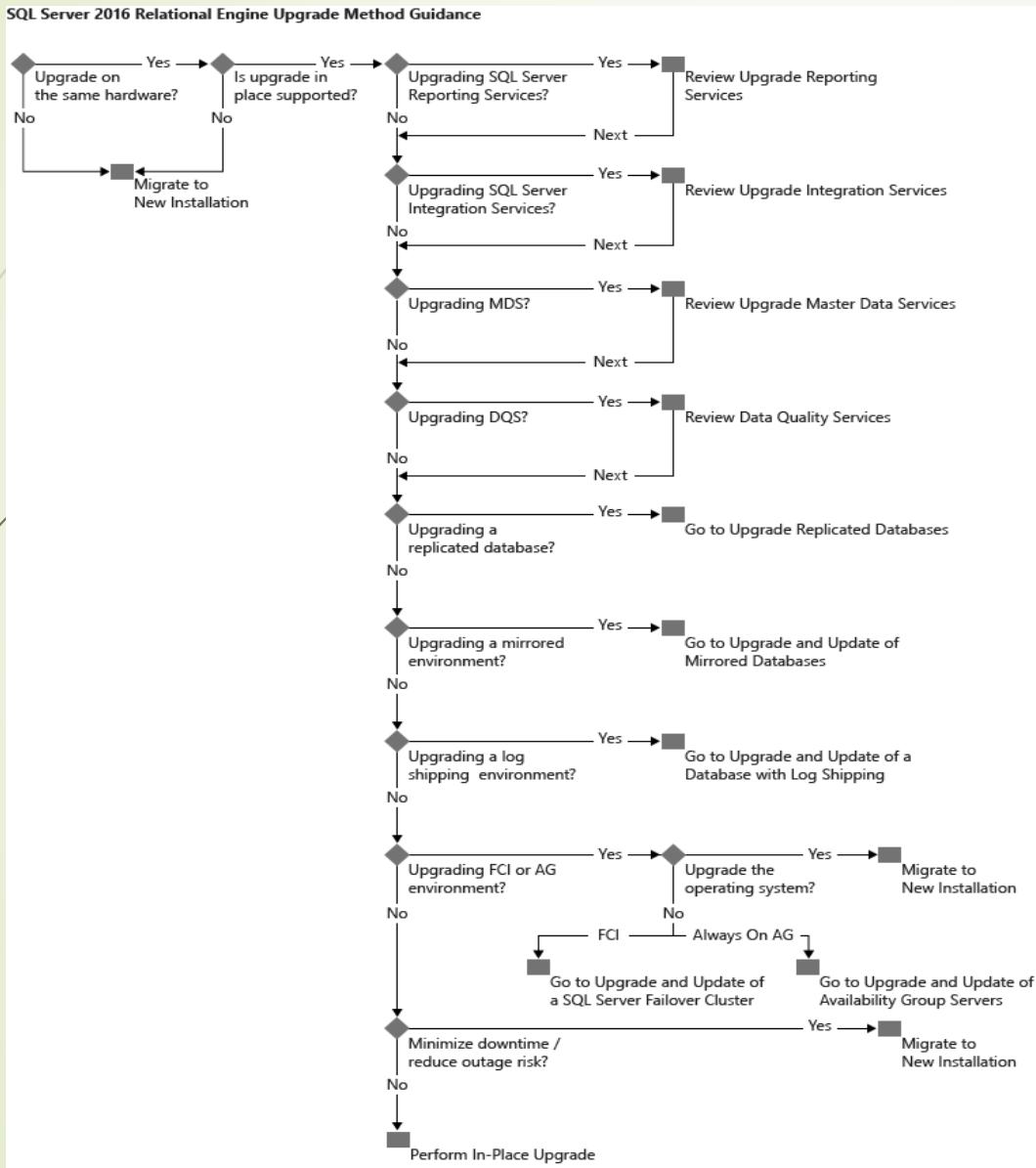
განახლების წესების (Upgrade Rules) ფანჯარაში, დაყენების პროცედურა ავტომატურად გადავა ეგზემპლიარის შერჩევის (Select Instance) ფანჯარაში, თუ არ გამოისახა წესის შეცდომები.

ეგზემპლიარის შერჩევის (Select Instance) გვერდზე მიუთითეთ SQL სერვერის ეგზემპლიარი, რომელიც უნდა განახლდეს. მხოლოდ გაზიარებული კომპონენტების განახლებისათვის უნდა შეირჩეს Upgrade shared features only.

კომპონენტების შერჩევის (Select Features) გვერდზე, განახლებულ იქნება წინასწარ შერჩეული კომპონენტები. თითოეული შერჩეული კომპონენტის ჯგუფის აღწერა გამოჩნდება მარჯვენა მიდამოში.

ეგზემპლიარის კონფიგურირების (Instance Configuration) გვერდზე უნდა იქნეს მითითებული SQL Server-ის ეგზემპლიარის ID, რომელიც წარმოადგენს ეგზემპლიარის სახელს. SQL Server-ის ყველა სერვისული პაკეტი და განახლება ვრცელდება SQL Server-ის ეგზემპლიარის ყველა კომპონენტზე.

MS SQL Server-ის Database Engine-ს განახლება



არსებობს SQL Server-ის წინა ვერსიების მონაცემთა ბაზის ძრავის (Database Engine) განახლების რამდენიმე მეთოდი, რათა მაქსიმალურად იქნეს შემცირებული SQL Server-ის გათიშვის დრო და რისკი:

- განახლება ადგილზე;
- მიგრაცია ახალ ინსტალაციაზე;
- მიმდევრობითი განახლება.

სლაიდზე მოყვანილი დიაგრამა ასახავს სამივე მიდგომას და დაეხმარება მომხმარებელს სწორი მიდგომის შერჩევაში.

MS SQL Server-ის Database Engine-ს განახლება

11

განვიხილოთ სამივე მიდგომა:

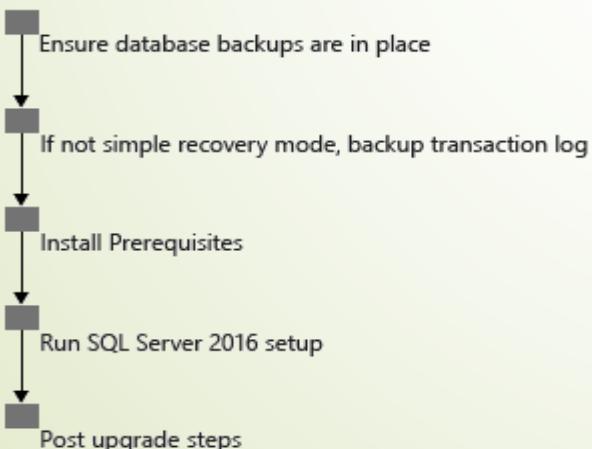
განახლება ადგილზე

ამ შემთხვევაში SQL Server-ის განახლება ხორციელდება იგივე კომპიუტერში და იგივე ოპერაციულ გარემოში - განახლებას ექვემდებარება მხოლოდ SQL Server-ი, რომელიც არსებული SQL Server-ის ბიტების შეცვლის ახალი SQL Server-ის ბიტებით და შემდგომ ახორციელებს სისტემისა და მომხმარებლის მონაცემთა ბაზების განახლებას.

განახლების ადგილზე მიდგომა არის ყველაზე მარტივი მეთოდი და მოითხოვს გარკვეული დროის უმოქმედობას, უფრო მეტი დრო სჭირდება მტყუნებისას უკან დაბრუნებას. გარდა ამისა ეს მეთოდი არ არის მხარდაჭერილი ყველა სცენარისათვის. ძირითად, ეს მიდგომა გამოიყენება იმ შემთხვევაში, როდესაც წარმოების გარემო არ არის კრიტიკული და მან

შეუძლია გაუძლოს გათიშვის გარკვეულ დროს.

SQL Server 2016 Relational Engine In-Place Upgrade Method



SQL Server-ის განახლებისას წინა ვერსია გადაიწერება და აღარ იარსებებს კომპიუტერში, ამიტომ განახლებამდე უნდა შეიქმნას SQL Server-ის მონაცემთა ბაზებისა და სხვა ობიექტების სარეზერვო კოპია.

მოცემული დიაგრამა არის მონაცემთა ბაზის ძრავის ადგილზე განახლებისთვის სქემა.

MS SQL Server-ის Database Engine-ს განახლება

12

მიგრაცია ახალ გარემოში

ამ მიდგომით შენარჩუნებული იქნება არსებული SQL Server-ის სისტემა და ხორციელდება ახალ ოპერაციულ გარემოში ან/და ახალ კომპიუტერში ახალი SQL Server-ის შექმნა. SQL Server-ის ახალ გარემოში დაყენების შემდეგ, ხორციელდება მთელი რიგი ნაბიჯები ახალი გარემოს მოსამზადებლად, რათა გახდეს შესაძლებელი არსებული მომხმარებელთა მონაცემთა ბაზების მიგრაცია ახალ გარემოში და მინიმუმადე იქნეს დაყვანილი გათიშვის დრო. ამ მიდგომისას მიგრაციას განიცდიან შემდეგი კომპონენტები:

- ▶ **სისტემური ობიექტები:** როგორც წესი, სამომხმარებლო პროგრამა არის დამოკიდებული როგორც სისტემურ master და msdb, ასევე სამომხმარებლო მონაცემთა ბაზებზე. ყველაფერი, რაც იქნა შენახული სამომხმარებლო მონაცემთა ბაზის გარეთ და საჭიროა მისი სწორი ფუნქციონირებისათვის, ხელმისაწვდომი უნდა იყოს ახალ SQL Server-ზეც. მაგალითად:
 - ▶ წვდომის პარამეტრები, რომლებიც ინახება master სისტემურ მონაცემთა ბაზაში;
 - ▶ SQL Server აგენტის ფუნქციონირება, რომელთა მეტამონაცემები ინახება msdb სისტემურ მონაცემთა ბაზაში;
 - ▶ სერვერის დონის ტრიგერები, რომლებიც ინახება master სისტემურ მონაცემთა ბაზაში.

ყველა ამ შემთხვევაში ისინი ხელახლა უნდა შეიქმნან ახალ SQL Server-ში.

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის Database Engine-ს განახლება

- **MSDB** მონაცემთა ბაზაში შენახული **Integration Services** პაკეტები: თუ პაკეტები შენახულია MSDB მონაცემთა ბაზაში, მაშინ საჭიროა ამ პაკეტების სკრიპტში ჩაწერა dtutil უტილიტების გამოყენებით (დაწვრილებითი ცნობები არის ვებ-გვერდზე <https://docs.microsoft.com/en-us/sql/integration-services/dtutil-utility?view=sql-server-ver15>), რის შემდეგაც პაკეტები უნდა განახლდნენ ახალი SQL Server-ის ვერსიამდე და შემდგომ უნდა გაიშალოს ახალ SQL Server-ში.
- **Reporting Services**-ის სერვისების დაშიფვრის გასაღებები: Reporting Services-ის კონფიგურაციის მნიშვნელოვანი ნაწილია სიმეტრიული გასაღების სარეზერვო ასლის შექმნა, რომელიც გამოიყენება კონფიდენციალური ინფორმაციის დასაშიფრად. გასაღების სარეზერვო ასლი საჭიროა მრავალი რუტინული ოპერაციისთვის და იძლევა საშუალებას გამოყენებულ იქნეს არსებული სერვერის მონაცემთა ბაზა ახალ SQL Server-ში.

რადგან ახალ SQL Server-ს გააჩნია იგივე სისტემური ობიექტები, როგორც ძველს, ამ მიდგომის გამოყენებისას სამომხმარებლო მონაცემთა ბაზებს მიგრაციის უმოქმედობის დრო მინიმუმამდე არის დაყვანილი. მონაცემთა ბაზების მიგრაციის განხორციელება შესაძლებელია სარეზერვო ასლის შექმნისა და აღდგენის გზით, ან LUN-ის SAN გარემოში გადაგზავნით.

MS SQL Server-ის Database Engine-ს განახლება

მომხმარებლის მონაცემთა ბაზის მიგრაციის შემდეგ ერთეულთი გზა მომხმარებლების ახალ თქვენ SQL Server-თან მიერთების არის DNS ჩანაწერებში სერვერის სახელის გადარქმევა და მიერთების სტრიქონების შეცვლა.

ამ მიდგომის გამოყენება მიზანშეწონილია თუ:

- ▶ SQL Server-ის ინსტალირება უნდა განხორციელდეს არამხარდამჭერ ოპერაციულ გარემოში (მაგ. ძველი SQL Server-ი არის x86 და გადატანილ უნდა იქნეს Windows Server 2016-ზე, რომელიც არის x64);
- ▶ SQL Server-ი აკეთებს მიგრაციას ახალ აპარატურაზე და/ან ახალ ოპერაციულ გარემოში;
- ▶ არაკლასტერული SQL Server-ი აკეთებს მიგრაციას კლასტერულ SQL Server-ში.
- ▶ SQL Server 2005 აკეთებს მიგრაციას SQL Server 2016-ში (13.x), რადგან იგი უკვე აღარ არის მხარდაჭერილი.

ახალ გარემოში მიგრაციის მოქმედებები განსხვავდება იმისდა მიხედვით გამოყენებულ იქნება მიერთებული საცავი თუ SAN საცავი

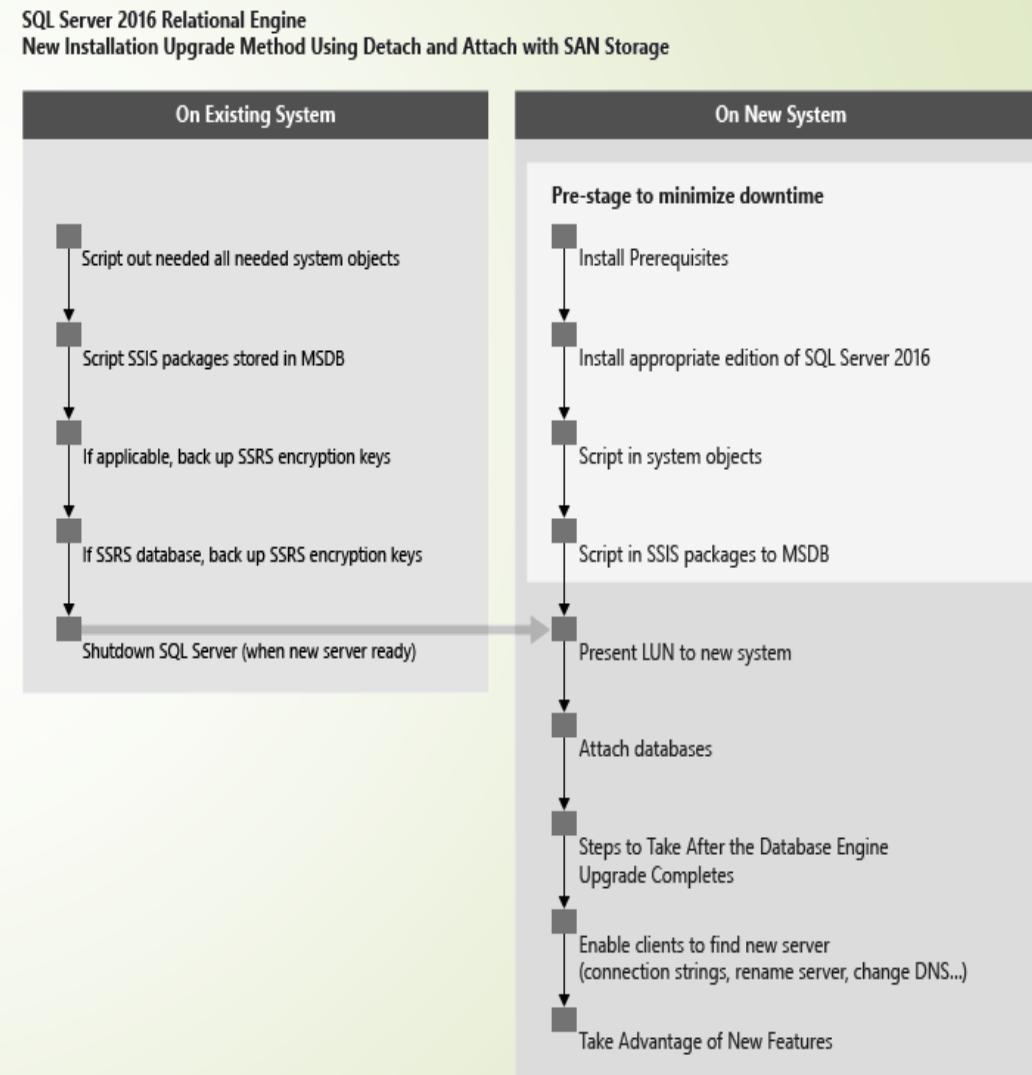
(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის Database Engine-ს განახლება

მიერთებული საცავის გარემო:



SAN საცავის გარემო:



MS SQL Server-ის Database Engine-ს განახლება

16

მიმდევრობითი განახლება

მიმდევრობითი განახლება საჭიროა როდესაც უნდა განახლდეს SQL Server-ის რამდენიმე ეგზემპლიარი, რომლებიც დროის უმოქმედობისა და რისკების შემცირებისათვის და გარემოს ფუნქციონირების შესანარჩუნებლად ხორციელდება. ანუ ამ მიდგომისას ხორციელდება SQL Server-ის რამდენიმე ეგზემპლიარის განახლება კონკრეტული თანმიმდევრობით - ხორციელდება SQL Server-ის ყოველი კონკრეტული ეგზემპლიარის განახლება ან ახალი ინსტალირება.

MS SQL Server-ის თავსებადობის დონის ცვლილება

SQL Server 2016-ში (13.x) და უფრო მაღალ ვერსიებში ზოგიერთი ცვლილება მხოლოდ მონაცემთა ბაზის თავსებადობის დონის შეცვლის შემდეგ არის შესაძლებელი. ეს გაკეთდა რამდენიმე მიზეზის გამო:

- ▶ შეუძლებელია SQL Server-ის მონაცემთა ბაზის თავსებადობის დონის დაქვეითება და თუ მომხმარებელს სურს ძველ ვერსიაზე დაბრუნება, იგი ამას ვეღარ გააკეთებდა;
- ▶ იგივე პრობლემა შეიძლება შეიქმნას სისტემის რაღაც ნაწილის განახლებისას წარმატების შემთხვევაში, შეიძლება მიუღებელი რეგრესია გამოიწვიოს სხვებისთვის.

მოთხოვნების ახალი დამამუშავებლის ფუნქციონირების გასააქტიურებლად განახლების პროცესი დაკავშირებულია პროდუქტის გამოშვების შემდგომ მომსახურების მოდელთან. ზოგიერთი ამ გამოსწორებებიდან გამოშვებულ იქნა ნომრით 4199, რომელიც მოულოდნელი რეგრესიის რისკის გარეშე შესაძლებელია დაყენდეს. დაწყებული SQL Server 2016-დან (13.x) აღარ არის საჭირო 4199 განახლება.

SQL Server-ის ძველი ვერსიიდან SQL Server 2014 (12.x) ან უფრო ახალ ვერსიაზე გადასვლისას მონაცემთა ბაზის თავსებადობის დონის უახლეს ვერსიამდე გადასვლისას, დატვირთვამ შესაძლებელია განიცადოს რეგრესია. ეს უფრო მცირე ხარისხით ასევე შესაძლებელია SQL Server 2014 (12.x)-ის ნებისმიერ ახალ ვერსიაზე განახლებისას.

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

18

SQL Server 2014 (12.x)-დან დაწყებული მოთხოვნების ოპტიმიზების ყველა ცვლილება დაკავშირებულია მონაცემთა ბაზის თავსებადობის ბოლო დონესთან, ამიტომ შესრულების გეგმები განახლების მომენტში კი არ იცვლება, არამედ როდესაც მომხმარებელი შეცვლის მონაცემთა ბაზის პარამეტრს COMPATIBILITY_LEVEL უახლეს ხელმისაწვდომადე.

განახლებების ეს კონტროლი კიდევ უფრო გაუმჯობესდა SQL Server 2017 (14.x)-დან, რომელშიც დაინერგა ავტომატური რეგულირება და იგი საშუალებას იძლევა ბოლო ბიჯის ავტომატიზებას.

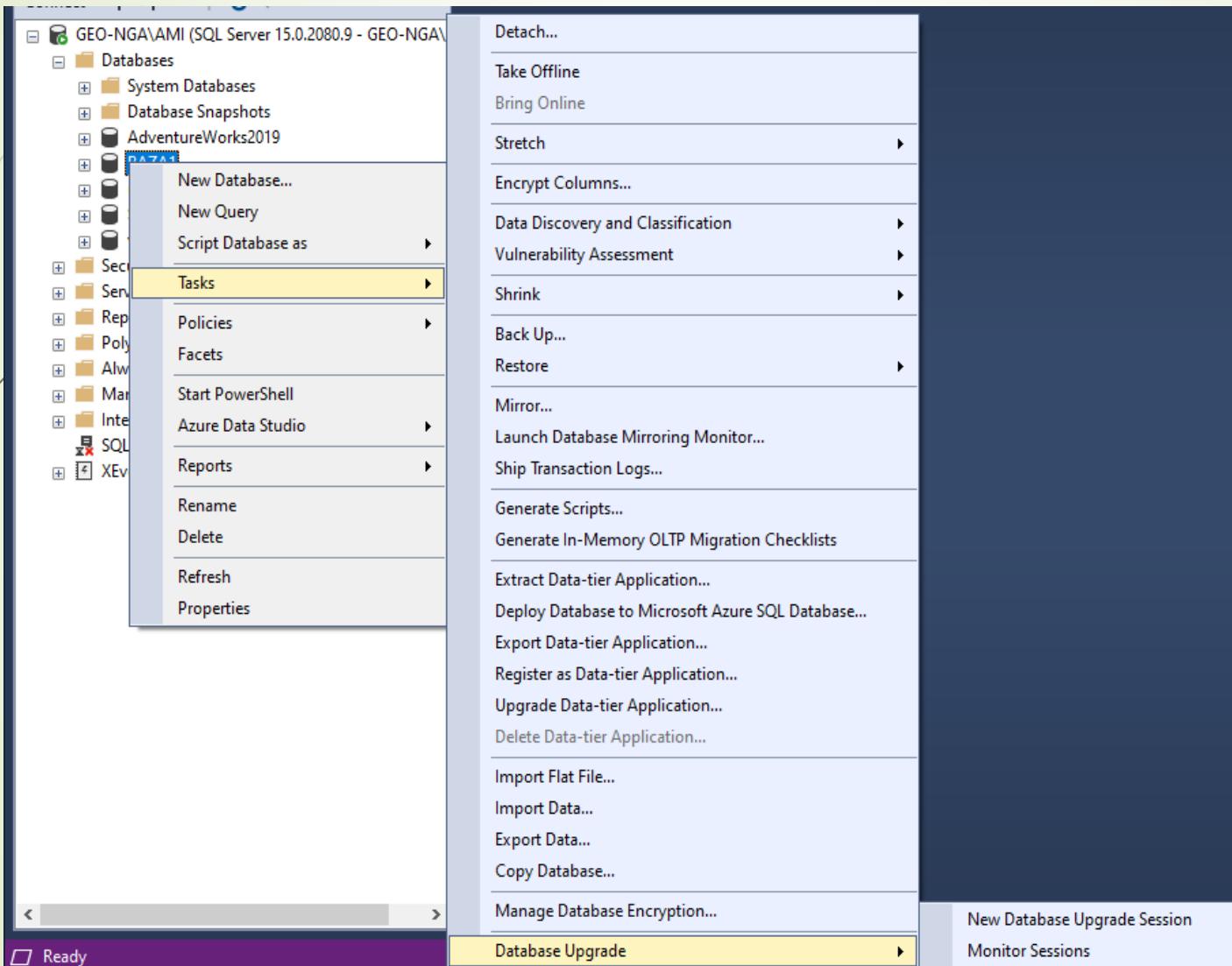
SQL Server Management Studio v18-დან დაწყებული, მომხმარებლებს აქვს შესაძლებლობა განახორციელოს მონაცემთა ბაზების განახლება „მოთხოვნის რეგულირების ასისტენტის“ (Query Tuning Assistant - QTA) გამოყენებით.

QTA არის სესიაზე დაფუძნებული კომპონენტი, რომელიც ინახავს სესიის მდგომარეობას მომხმარებლის მონაცემთა ბაზის (რომელშიც დაიწყო სეანსი) თსგთა სქემაში. დროთა განმავლობაში ერთ მონაცემთა ბაზაში შესაძლებელია შეიქმნას მრავალი QTA სესია, მაგრამ ნებისმიერი მონაცემთა ბაზისთვის შეიძლება არსებობდეს მხოლოდ ერთი აქტიური სესია.

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

19

მონაცემთა ბაზის განახლების ახალი სესიის შექმნა



SQL Server Management Studio-ში მაუსის მარჯვენა ღილაკით უნდა გააქტიურდეს იმ მონაცემთა ბაზის სახელი, რომლისთვისაც საჭიროა თავსებადობის დონის ამაღლება. შემდეგ უნდა გააქტიურდეს სტრიქონები მიმდევრობით:

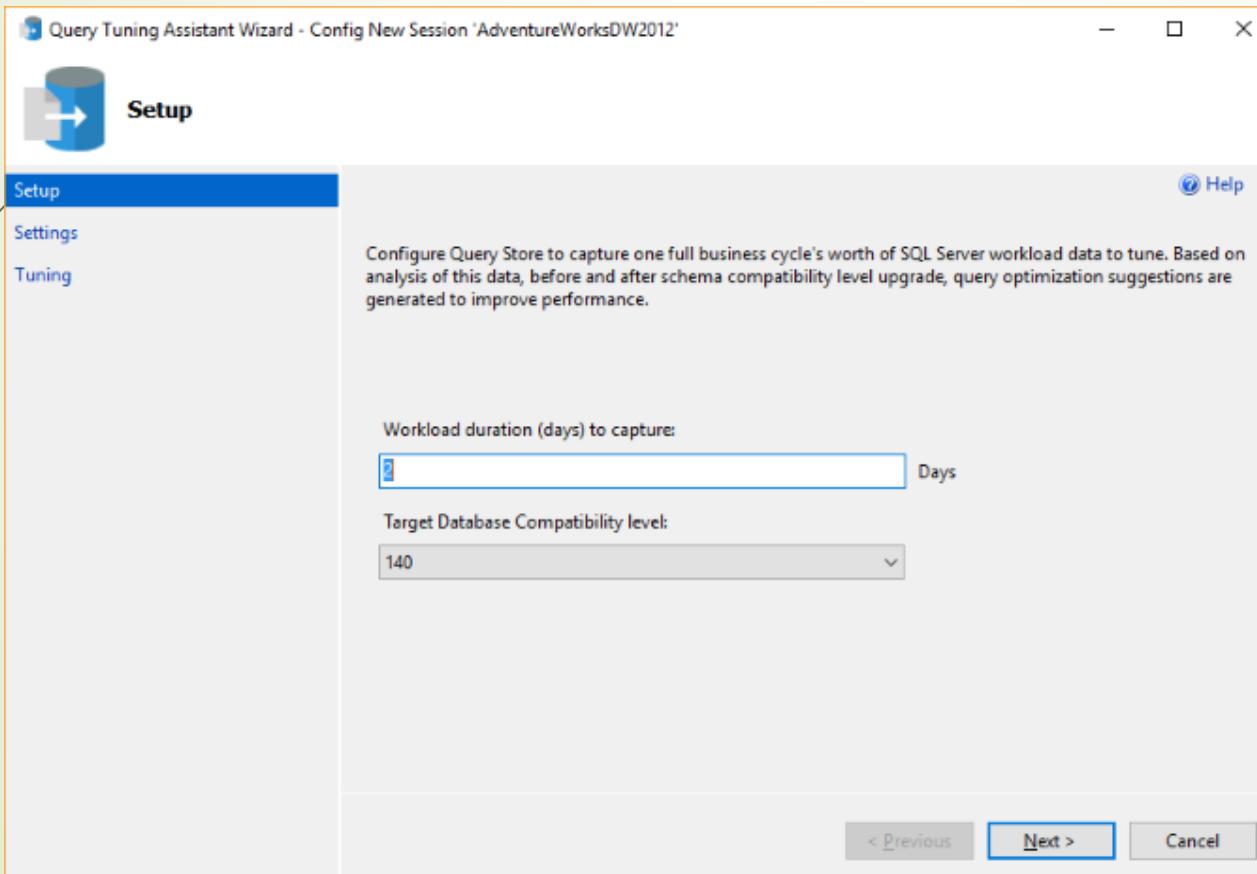
„ამოცანები“ (Tasks),
„მონაცემთა ბაზის განახლება“ (Database Upgrade) და „მონაცემთა ბაზის განახლების ახალი სესია“ (New Database Upgrade Session).

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

20

QTA-ს ოსტატის პირველ დიალოგურ ფანჯარაში მომხმარებელმა უნდა მიუთითოს:

- მოსალოდნელი დატვირთვის ხანგრძლივობა დღეებში (მინიმუმ 1 დღე);
- მიზნობრივი მონაცემთა ბაზის თავსებადობის დონე, რომლითაც უნდა გააჩნდეს მომხმარებლის მონაცემთა ბაზას QTA სამუშაო პროცესის დასრულების შემდეგ.



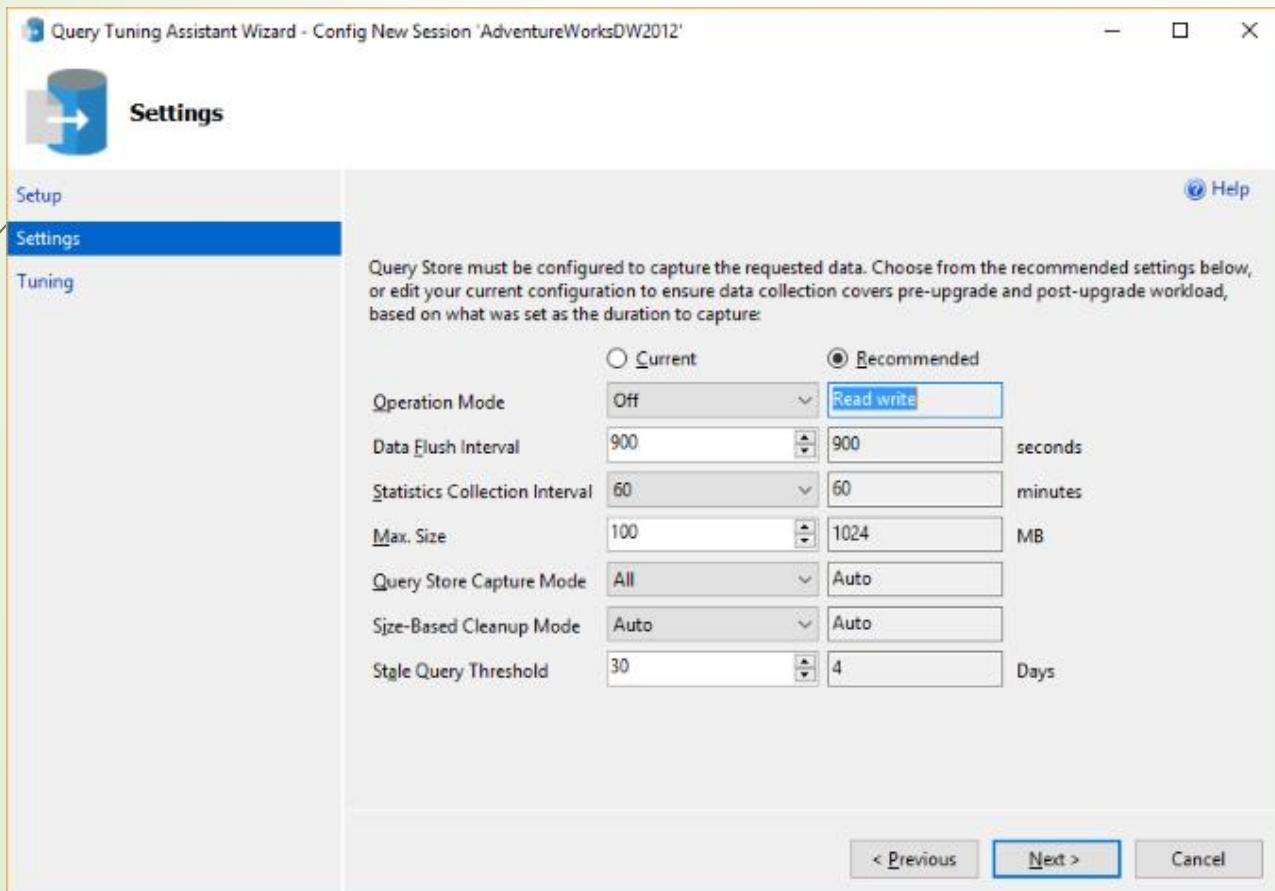
შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად ღილაკი „შემდეგი“ (Next).

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

21

შემდეგი დიალოგური ფანჯარა არის პარამეტრების ფანჯარა, რომელიც მოცემულია ორი სვეტის გამოსახულებით: მიმდინარე პარამეტრები და რეკომენდირებული პარამეტრები. რეკომენდებული პარამეტრები ნაგულისხმევად არის შერჩეული, მაგრამ მომხმარებელს ეძღვა შესაძლებლობა გადაერთოს მიმდინარე სვეტზე და განახორციელოს შეხედულებისამებრ პარამეტრების ცვლილება.



შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად ღილაკი „შემდეგი“ (Next).

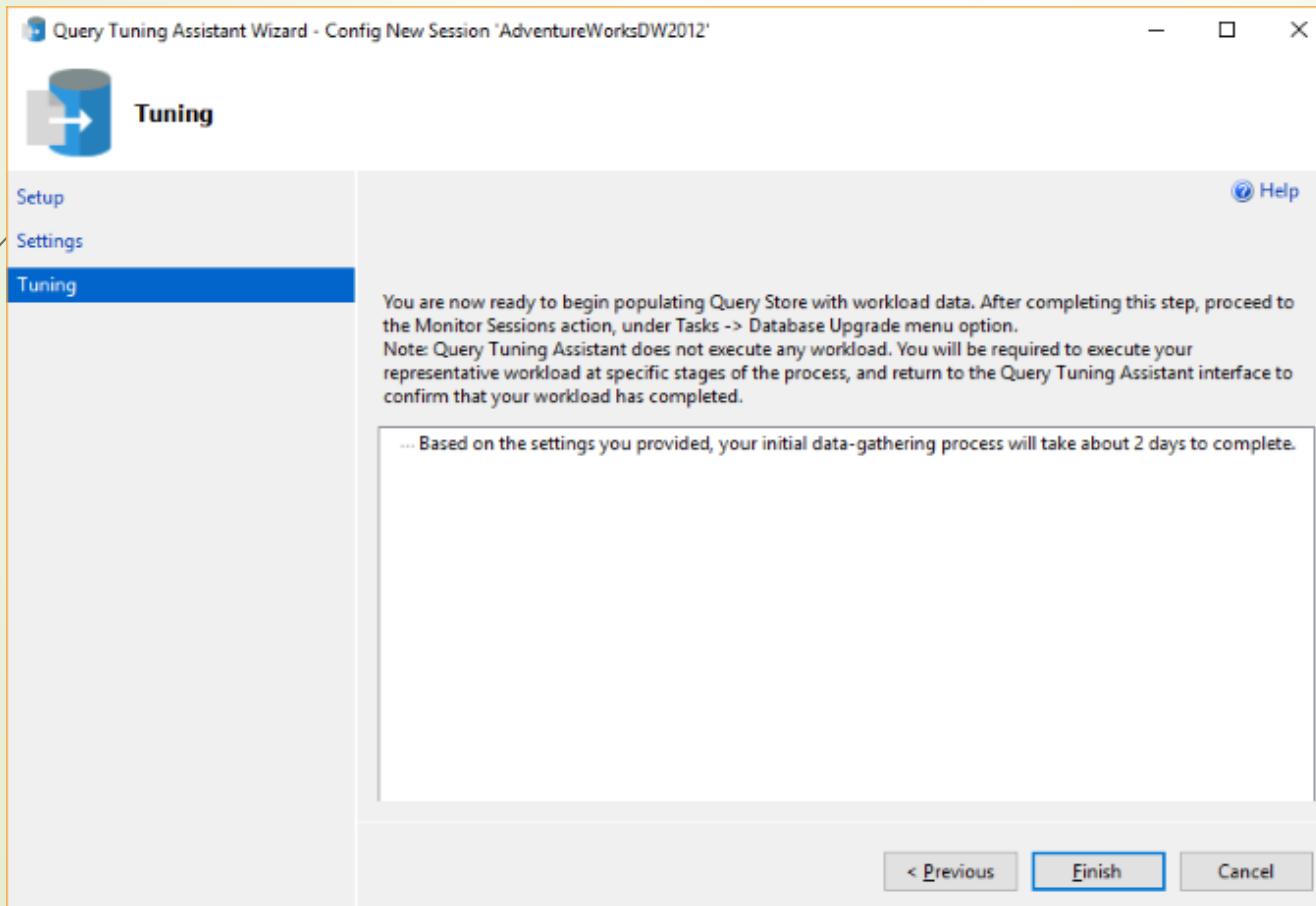
(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

22

შემდეგ დიალოგურ ფანჯარაში ხორციელდება სეანსის კონფიგურაციას დასკვნითი მოქმედებები და შეიცავს ინფორმაციას სეანსის გახსნის შემდგომ მოქმედებებზე და მასთან მუშაობაზე.

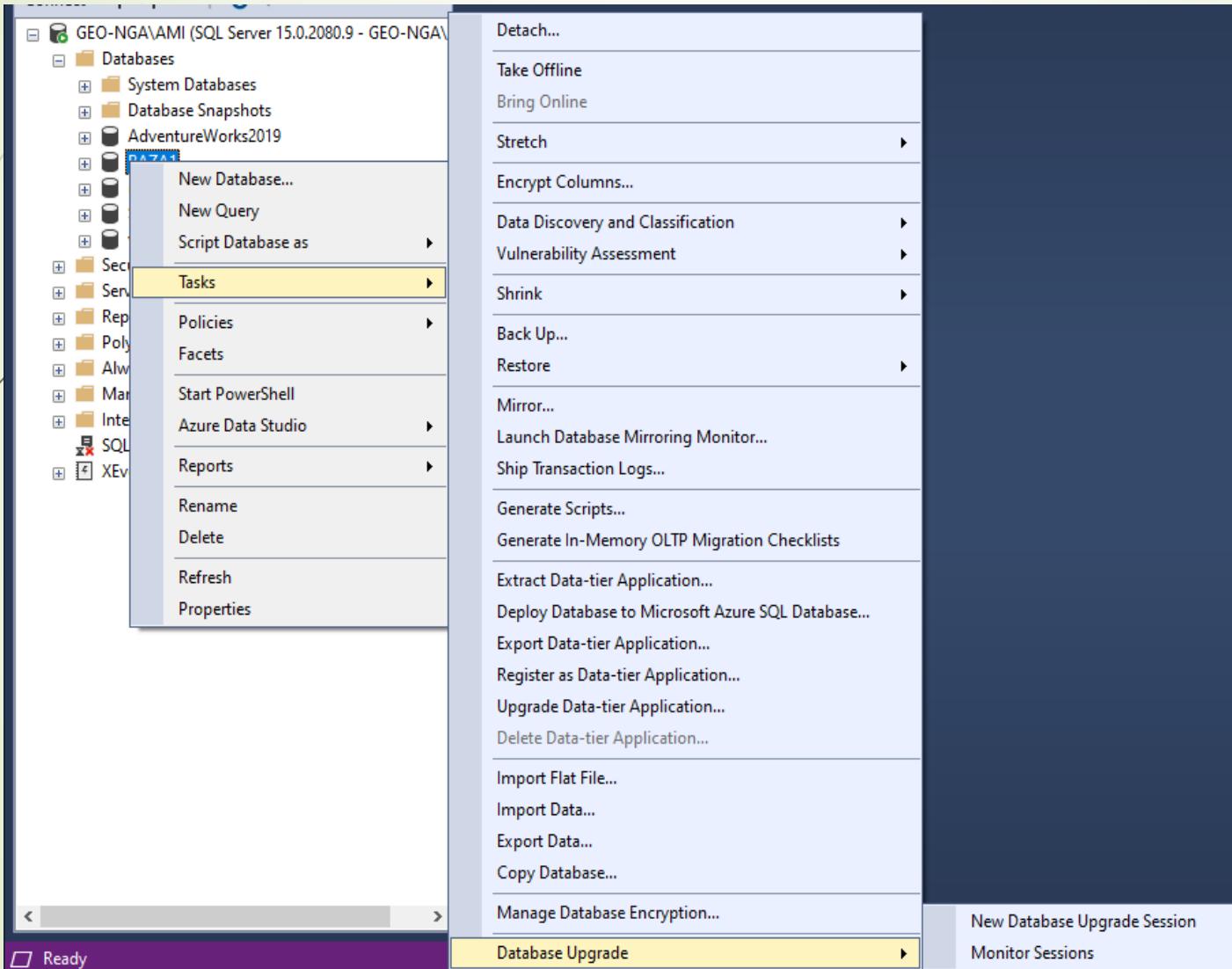
დასრულებისათვის ღილაკი „დასრულება“ (Finish).



MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

23

სესიების მონიტორინგი



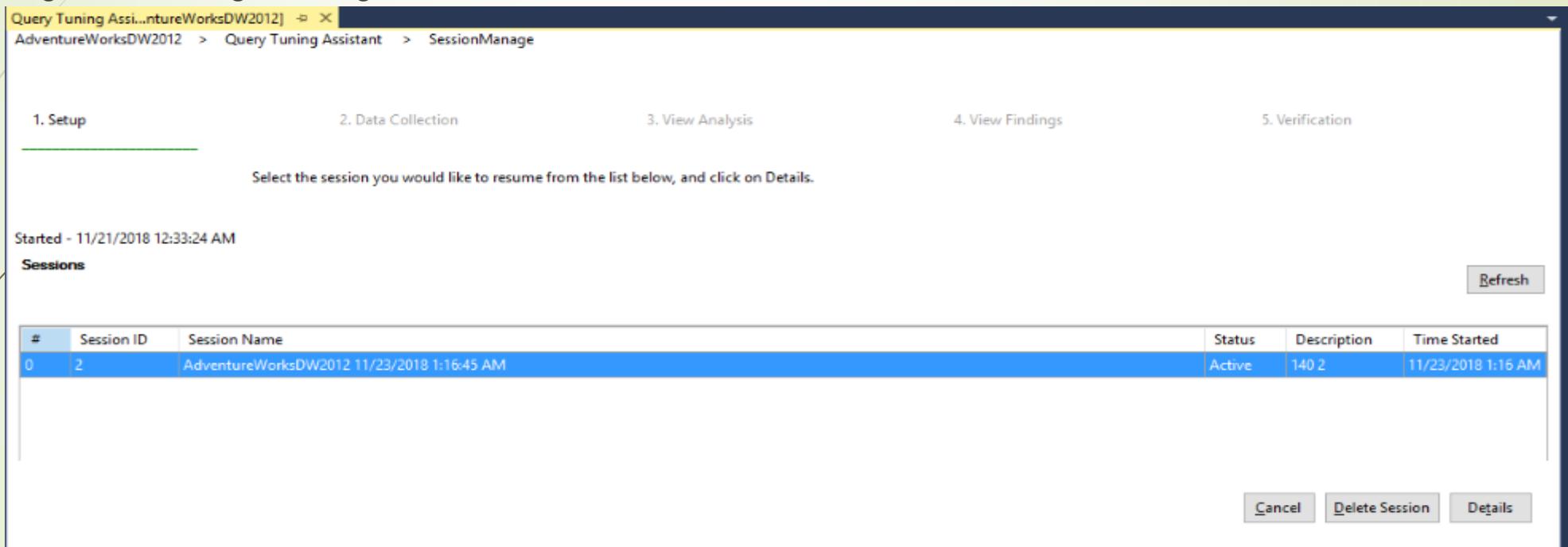
SQL Server Management Studio-ში მაუსის მარჯვენა ღილაკით უნდა გააქტიურდეს იმ მონაცემთა ბაზის სახელი, რომლისთვისაც საჭიროა თავსებადობის დონის ამაღლება. შემდეგ უნდა გააქტიურდეს სტრიქონები მიმდევრობით:

„ამოცანები“ (Tasks),
„მონაცემთა ბაზის განახლება“ (Database Upgrade) და „სესიების მონიტორინგი“ (Monitor Sessions).

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

24

QTA-ს მონიტორინგის პირველ დიალოგურ ფანჯარაში (1. Setup) გამოისახება მიმდინარე და წარსული სეანსების სია ამ მონაცემთა ბაზისათვის, რომელშიც მომხმარებელმა უნდა შეარჩიოს საჭირო სეანსი:



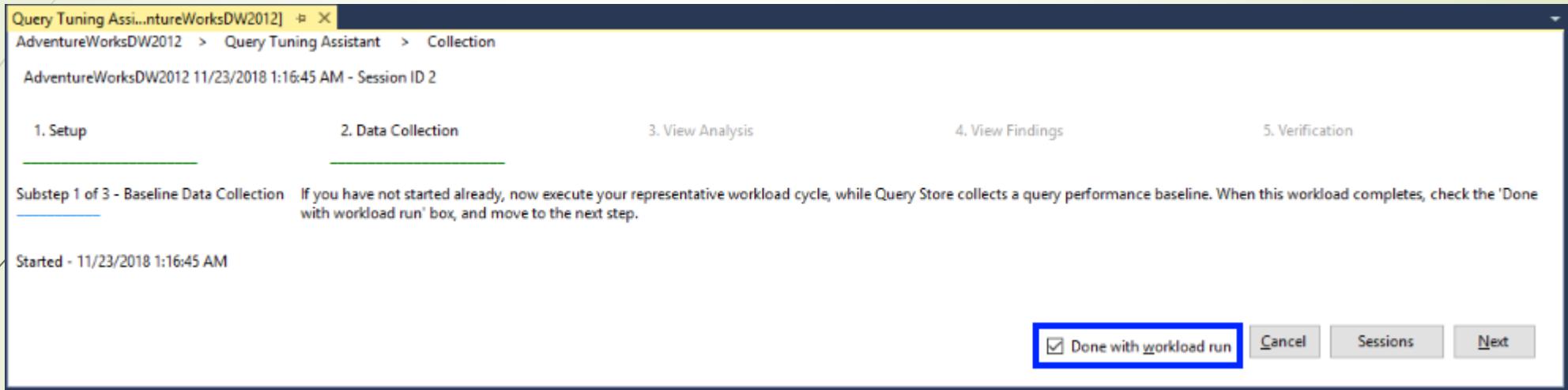
თუ მიმდინარე სეანსი არ არის სიაში, მომხმარებელმა უნდა გააკტიუროს ღილაკი „განახლება“ (Refresh), ხოლო შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად ღილაკი „დეტალები“ (Details).

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

25

შემდეგი დიალოგური ფანჯარა საბაზისო მონაცემების კოლექცია (2. Data Collection) მომხმარებელს სთხოვს აწარმოოს წარმომადგენლობითი მუშა დატვირთვის ციკლი, რათა მონაცემთა საცავმა შეძლოს შეაგროვოს ძირითადი მონაცემები.



მუშა დატვირთვის დასრულებისათვის მომხმარებელმა უნდა მონიშნოს „დასრულება მუშა დატვირთვის შესრულებით“ (Done with workload run) და შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად ღილაკი „შემდეგი“ (Next).

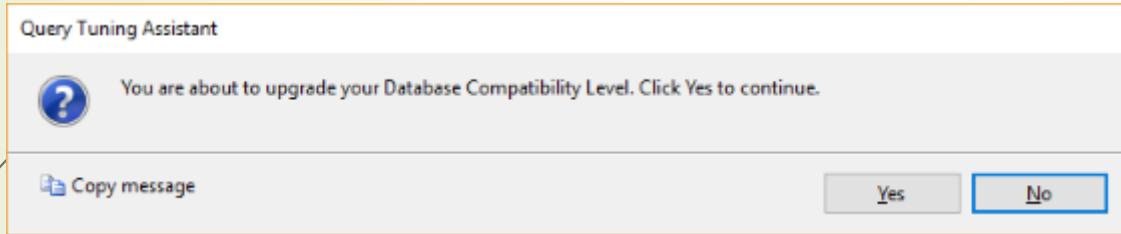
(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

26

QTA-ს დიალოგური ფანჯარა შეიძლება დაიხუროს მუშაობის დატვირთვისას. სესიაზე დაბრუნება, რომელიც შემდგომში რჩება აქტიურ მდგომარეობაში, განახლდება იმავე საფეხურიდან, სადაც ის შეჩერებული იყო.

შემდეგ დიალოგურ ფანჯარაში სისტემა მოითხოვს მონაცემთა ბაზის თავსებადობის დონის სასურველ მიზნამდე განახლების ნებართვას.



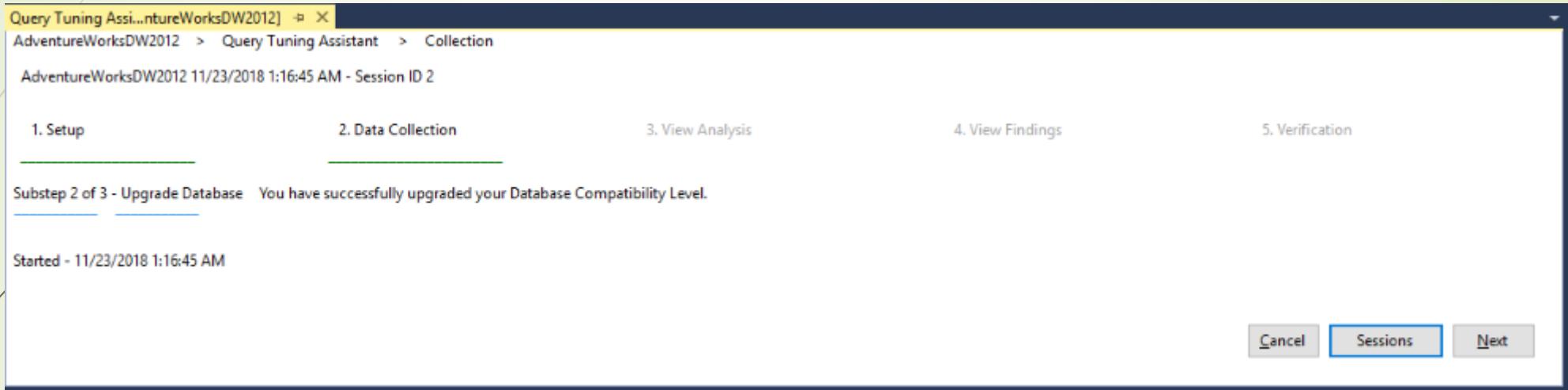
შემდეგ საფეხურზე გადასასვლელად მომხმარებელმა უნდა გააქტიუროს ღილაკი „დიაბ“ (Yes).

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

27

შემდეგ დიალოგურ ფანჯარაში დასტურდება, რომ მონაცემთა ბაზის თავსებადობის დონე
წარმატებით იქნა განახლებული.



შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად მომხმარებელმა უნდა გააკტიუროს
ღილაკი „სესიები“ (Sessions).

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

28

შემდეგ დიალოგურ ფანჯარაში მომხმარებელს სთხოვს ხელახლა აწარმოოს წარმომადგენლობითი მუშა დატვირთვის ციკლი.

The screenshot shows the Microsoft Query Tuning Assistant interface for the AdventureWorksDW2012 database. The title bar says "Query Tuning Assistant [AdventureWorksDW2012]". The main window has tabs: 1. Setup, 2. Data Collection, 3. View Analysis, 4. View Findings, and 5. Verification. The "Data Collection" tab is selected. A sub-step message says: "Substep 3 of 3 - Observed Data Collection. Now that you have upgraded to the desired Database Compatibility Level, re-run the same workload as you did before. This enables Query Store to collect a comparative baseline that will be used to search for optimization opportunities. When this workload completes, check the 'Done with workload run' box, and move to the next step." Below this, it says "Started - 11/23/2018 1:16:45 AM". A table titled "Top 10 regressed queries" lists the following data:

#	Query ID	Query Text	Runs	Baseline Metric	Observed Metric	% Change	Tunable
0	1	IF @maxWeight <= (SELECT Weight from DimProduct WHERE ProductKey = @productKey)	300	0.4	0.65	-62.50	False
1	12	SELECT OrderDateKey, SUM(SalesAmount) AS TotalSales FROM FactInternetSales GROUP BY OrderDateKey...	300	26.88	28.2	-4.91	True
2	8	SELECT NumberCarsOwned FROM DimCustomer GROUP BY YearlyIncome, NumberCarsOwned	300	10.26	10.75	-4.78	True
3	14	SELECT * FROM FactInternetSales fis INNER JOIN DimProduct dp ON fis.ProductKey = dp.ProductKey WHERE...	300	1184.14	1236.23	-4.40	True
4	6	SELECT CustomerKey, SUM(SalesAmount) AS sas FROM FactInternetSales GROUP BY CustomerKey WITH (...	300	55.38	56.66	-2.31	True
5	11	SELECT SalesAmount FROM FactInternetSales GROUP BY SalesAmount, SalesAmount*1.10	300	50.61	51.41	-1.58	True
6	10	SELECT SalesAmount, SalesAmount*1.10 SalesTax FROM FactInternetSales GROUP BY SalesAmount	300	23.05	23.35	-1.30	True
7	9	SELECT (SalesAmount + TaxAmt + Freight) AS TotalCost FROM FactInternetSales GROUP BY SalesAmount, T...	300	33.22	33.29	-0.21	True
8	2	(SELECT @productKey AS ProductKey, EnglishDescription, Weight, 'This product is too heavy to ship and i...	300	0.02	0.02	0.00	False
9	13	SELECT TOP (10) r.ResellerName, r.AnnualSales FROM DimReseller AS r ORDER BY AnnualSales DESC, Resell...	300	0.56	0.56	0.00	True

At the bottom right, there are buttons: "Done with workload run" (with a checked checkbox), "Cancel", "Sessions", and "Next".

მუშა დატვირთვის დასრულებისათვის მომხმარებელმა უნდა მონიშნოს „დასრულება მუშა დატვირთვის შესრულებით“ (Done with workload run) და შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად ღილაკი „შემდეგი“ (Next).

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

29

შემდეგ დიალოგურ ფანჯარაში „ანალიზის ნახვა“ (3. View Analysis) მომხმარებელს ეძლევა შესაძლებლობა შეარჩიოს მოთხოვნები ექსპერიმენტისთვის და იპოვოს ოპტიმიზაციის შესაძლებლობები. მოთხოვნების შერჩევაზე დაბრუნება ამ დიალოგური ფანჯრიდან გადასვლის შემდეგ შეუძლებელია და ამიტომ, თუ არ იქნა შერჩეული ყველა სასურველი მოთხოვნა, მომხმარებელმა მოგვიანებით უნდა შექმნას ახალი სესია და გაიმეოროს ყველა მოქმედებები.

#	Query ID	Query Text	Runs	Baseline Metric	Observed Metric	% Change	Tunable
0	<input type="checkbox"/> 1	IF @maxWeight <= (SELECT Weight from DimProduct WHERE ProductKey = @productKey)	300	0.4	0.65	+62.50	False
1	<input checked="" type="checkbox"/> 12	SELECT OrderDateKey, SUM(SalesAmount) AS TotalSales FROM FactInternetSales GROUP BY OrderDateKe...	300	26.88	28.2	+4.91	True
2	<input checked="" type="checkbox"/> 8	SELECT NumberCarsOwned FROM DimCustomer GROUP BY YearlyIncome, NumberCarsOwned	300	10.26	10.75	+4.78	True
3	<input checked="" type="checkbox"/> 14	SELECT * FROM FactInternetSales fis INNER JOIN DimProduct dp ON fis.ProductKey = dp.ProductKey WHERE ...	300	1184.14	1236.23	+4.40	True
4	<input checked="" type="checkbox"/> 6	SELECT CustomerKey, SUM(SalesAmount) AS sas FROM FactInternetSales GROUP BY CustomerKey WITH (...	300	55.38	56.66	+2.31	True
5	<input checked="" type="checkbox"/> 11	SELECT SalesAmount FROM FactInternetSales GROUP BY SalesAmount, SalesAmount*1.10	300	50.61	51.41	+1.58	True
6	<input checked="" type="checkbox"/> 10	SELECT SalesAmount, SalesAmount*1.10 SalesTax FROM FactInternetSales GROUP BY SalesAmount	300	23.05	23.35	+1.30	True
7	<input checked="" type="checkbox"/> 9	SELECT (SalesAmount + TaxAmt + Freight) AS TotalCost FROM FactInternetSales GROUP BY SalesAmount, ...	300	33.22	33.29	+0.21	True
8	<input type="checkbox"/> 2	(SELECT @productKey AS ProductKey, EnglishDescription, Weight, This product is too heavy to ship and ...	300	0.02	0.02	0.00	False
9	<input checked="" type="checkbox"/> 13	SELECT TOP (10) r.ResellerName, r.AnnualSales FROM DimReseller AS r ORDER BY AnnualSales DESC, Resel...	300	0.56	0.56	0.00	True

ექსპერიმენტის დასაწყებად ღილაკი „შემდეგი“ (Next).
(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

30

შემდეგ დიალოგურ ფანჯარაში „შედეგების ჩვენება“ (View Findings) მომხმარებელს ეძლევა შესაძლებლობა შეარჩიოს გეგმის სტრუქტურისათვის თუ რომელი მოთხოვნები გამოიყენოს შემოთავაზებული გადაწყვეტილებისათვის.

The screenshot shows the 'Query Tuning Assistant' interface for the 'AdventureWorksDW2012' database. The title bar reads 'Query Tuning Assistant [AdventureWorksDW2012]'. The main window has tabs: '1. Setup', '2. Data Collection', '3. View Analysis', '4. View Findings' (which is selected), and '5. Verification'. Below the tabs is a table with the following data:

#	Query ID	Query Text	Status	Baseline Metric	Observed Metric	% Change	Query Option	Can Deploy
0	12	SELECT OrderDateKey, SUM(SalesAmount) AS T...	Test complete	26.88	17	36.76	https://go.microsoft.com/fwlink/?linkid=2028175	True
1	8	SELECT NumberCarsOwned FROM DimCustom...	Test complete	10.26	7	31.77	https://go.microsoft.com/fwlink/?linkid=2028175	True
2	14	SELECT * FROM FactInternetSales fis INNER JOI...	Test complete	1184.14	484	59.13	https://go.microsoft.com/fwlink/?linkid=2028175	True
3	6	SELECT CustomerKey, SUM(SalesAmount) AS s...	Test complete	55.38	32	42.22	https://go.microsoft.com/fwlink/?linkid=2028175	True
4	11	SELECT SalesAmount FROM FactInternetSales G...	Test complete	50.61	32	36.77	https://go.microsoft.com/fwlink/?linkid=2028175	True
5	10	SELECT SalesAmount, SalesAmount*1.10 SalesT...	Test complete	23.05	14	39.26	https://go.microsoft.com/fwlink/?linkid=2028175	True
6	9	SELECT (SalesAmount + TaxAmt + Freight) AS ...	Test complete	33.22	21	36.79	https://go.microsoft.com/fwlink/?linkid=2028175	True
7	13	SELECT TOP (10) r.ResellerName, r.AnnualSales ...	Test complete	0.56	0	100.00	https://go.microsoft.com/fwlink/?linkid=2028175	True

Below the table, a note states: 'Hints force certain behaviors that may get addressed in subsequent updates. Microsoft recommends you only apply hints when no other option exists, and that you plan to revisit hinted code with every new upgrade. By forcing behaviors, you may not allow your workload to benefit from enhancements introduced in newer versions.'

Buttons at the bottom right include 'Cancel', 'Sessions' (highlighted in blue), and 'Deploy'.

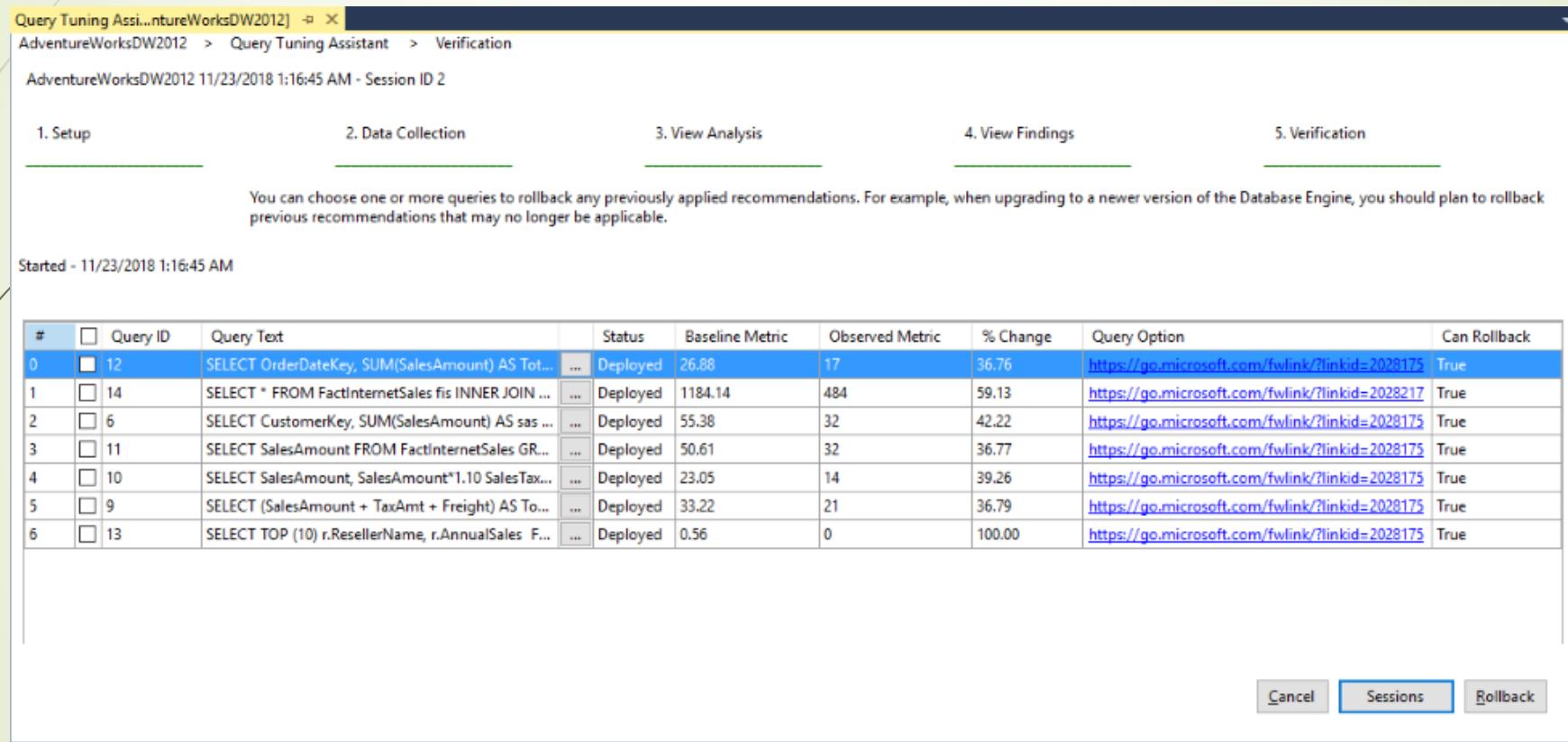
შემდეგ დიალოგურ ფანჯარაზე გადასასვლელად მომხმარებელმა უნდა გაააქტიუროს ღილაკი „სესიები“ (Sessions).

(გაგრძელება შემდეგ სლაიდზე)

MS SQL Server-ის მზ-ს განახლება QTA-ს გამოყენებით

31

შემდეგი დიალოგური ფანჯარა „შემოწმება“ (Verification) წინა ფანჯრისგან განსხვავდება მხოლოდ ბოლო სვეტით: წინაში იყო „შესაძლებელია გაშლა“ (Can Deploy), ხოლო ამაში - „შესაძლებელია უკუქცევა“ (Can Rollback), რომლის მითითებისას სტრიქონი უკუიქცევა.



პროცესის დასრულებისათვის და განახლების დაწყებისათვის მომხმარებელმა უნდა გაააქტიუროს ღილაკი „სესიები“ (Sessions).

დიდი მოცულობის კოპირების პროგრამა bulk copy (bcp)

bcp გამოიყენება მონაცემების მასიური კოპირებისათვის Microsoft SQL Server-სა და მომხმარებლის მიერ განსაზღვრულ ფორმატში მონაცემთა ფაილს შორის. bcp-ს გამოყენებით შესაძლებელია SQL Server-ის ცხრილებში ახალი სტრიქონების დიდი რაოდენობით იმპორტი ან SQL Server-ის ცხრილებიდან მონაცემთა ფაილებში მონაცემების ექსპორტი. queryout მოთხოვნის პარამეტრის გამოყენებისას გარდა, პროგრამა არ საჭიროებს Transact-SQL ენის ცოდნას.

SQL Server-ის ცხრილში მონაცემების იმპორტირებისთვის უნდა იქნეს გამოყენებული ფორმატის ფაილი, რომელიც შეიქმნა ამ ცხრილისთვის ან უნდა ვიცოდეთ ცხრილის სტრუქტურა და მისი სვეტებისთვის დაშვებული მონაცემთა ტიპები.

bcp-ს მონაცემთა სარეზერვო ასლის შექმნისათვის გამოყენებისას, უნდა შეიქმნას ფორმატის ფაილი მონაცემთა ფორმატის დასაწერად, რადგან bcp მონაცემთა ფაილები არ შეიცავენ ინფორმაციას სქემის ან ფორმატის შესახებ და თუ ცხრილი ან წარმოდგენა წაიშლება ფორმატის ფაილის უქონლობის შემთხვევაში მონაცემების იმპორტი შეუძლებელია.

bcp-ს ბოლო ვერსია შესაძლებელია ჩამოტვირთულ იქნეს შემდეგი ბმულიდან:

[Download Microsoft Command Line Utilities 15 for SQL Server \(x64\)](#)

bcp-ს ინსტალირება

2

BCP-ს ინსტალირების წინ უნდა შემოწმდეს თავსებადობის და არსებობის პირობები:

1. სისტემისადმი მოთხოვნები: **Windows 10, Windows 7, Windows 8, Windows 8.1, Windows Server 2008, Windows Server 2008 R2, Windows Server 2008 R2 SP1, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, Windows Server 2019**
2. სისტემაში უნდა არსებობდეს: Windows Installer 4.5 და Microsoft ODBC Driver 17 for SQL Server.
3. BCP ვერსიის შესამოწმებლად უნდა შესრულდეს ბრძანება **bcp /v** (დაადასტურეთ, რომ გამოიყენება 15.0.2000.5 ან უფრო მაღალი ვერსია).

მხოლოდ ამ პირობების შემოწმების შემდეგ არის bcp-ს გამოყენების შესაძლებლობა, რომლის შესრულებისათვის შესაბამისი ბრძანება უნდა შევიტანოთ Command Prompt-ის ფანჯარაში (Start → All Programs → Accessories → Command Prompt).

BCP-ს შედეგები არ აისახება ტრანზაქციის ჟურნალში, ანუ თუ გადაწერის ოპერაცია წარუმატებლად დამთავრდა, მაშინ საწყისი მონაცემები ვერ აღდგება. BCP მუშაობს ავტომატურ (თუ დაფორმატების გასაღებები მითითებულია ბრძანების სტრიქონში) ან ინტერაქტიურ რეჟიმში (დაფორმატების გასაღებების გარეშე და მაშინ გაიცემა შესაბამისი შეკითხვები).

BCP-ს სინტაქსი

3

BCP-ს გააჩნია შემდეგი სინტაქსი:

```
bcp [database_name.] schema.{table_name | view_name | "query"}  
{in data_file | out data_file | queryout data_file | format nul}
```

<ul style="list-style-type: none">[-a packet_size][-b batch_size][-c][-C { ACP OEM RAW code_page }][-d database_name][-D][-e err_file][-E][-f format_file][-F first_row][-G Azure Active Directory Authentication][-h"hint [,...n]"][-i input_file][-k][-K application_intent][-I login_timeout][-L last_row]	<ul style="list-style-type: none">[-m max_errors][-n][-N][-o output_file][-P password][-q][-r row_term][-R][-S [server_name[\instance_name]]][-t field_term][-T][-U login_id][-v][-V (80 90 100 110 120 130)][-w][-x]
--	--

bcp-ს არგუმენტები

4

განვიხილოთ BCP-ს არგუმენტები:

data_file - არის მონაცემთა ფაილის სრული გზა. როდესაც ხორციელდება მონაცემების იმპორტირება SQL Server-ში, მონაცემთა ფაილი შეიცავს მონაცემებს, რომლებიც კოპირდება მითითებულ ცხრილში ან წარმოდგენაში. როდესაც SQL Server-დან ხდება მონაცემთა დიდი მოცულობის ექსპორტი, მონაცემთა ფაილი შეიცავს ცხრილიდან ან წარმოდგენიდან კოპირებულ მონაცემებს. გზას შეიძლება ჰქონდეს 1 - დან 255 სიმბოლომდე. მონაცემთა ფაილი შეიძლება შეიცავდეს მაქსიმუმ $2^{63} - 1$ მწკრივს.

database_name - მონაცემთა ბაზის სახელია, რომელშიც მდებარეობს მითითებული ცხრილი ან წარმოდგენა. თუ იგი არ არის მითითებული, მაშინ განიხილება მომხმარებლის ნაგულისხმევი მონაცემთა ბაზა. ასევე შესაძლებელია პირდაპირ მიეთითოს მონაცემთა ბაზის სახელი -d-ს გამოყენებით.

in data_file | out data_file | queryout data_file | format nul - განსაზღვრავს მონაცემთა გადაგზავნის მიმართულებას, შემდეგნაირად:

- **in** - ასლებიდან ფაილიდან მონაცემთა ბაზის ცხრილში ან წარმოდგენაში გადმოტანა;

bcp-ს არგუმენტები

5

- **out** - გადასცემს მონაცემებს მონაცემთა ბაზის ცხრილიდან ან წარმოდგენიდან მონაცემთა ფაილში. თუ მიუთითებთ არსებულ ფაილს, ფაილი გადაიწერა. მონაცემთა ამოღებისას bcp პროგრამა წარმოადგენს ცარიელ სტრიქონს როგორც null-ს და null სტრიქონს როგორც ცარიელ სტრიქონს;
- **queryout** - აკოპირებს მონაცემებს მოთხოვნიდან (გამოიყენება მხოლოდ მოთხოვნიდან მონაცემების მასობრივი კოპირებისას);
- **format** - ქმნის ფორმატის ფაილს დამყარებულს მითითებულ პარამეტრებზე (-n, -c, -w, ან -N) და ცხრილის ან წარმოდგენის გამყოფებზე. მონაცემთა მასიურ კოპირებისას bcp პროგრამამ შეიძლება მიმართოს ფორმატის ფაილს, რაც ზოგავს დროს და საშუალებებს ფორმატის ინფორმაციის ინტერაქტიურად განმეორებით შეტანისას. Format პარამეტრი მოითხოვს -f პარამეტრს, ხოლო XML-ფაილის შექმნისას ასევე მოითხოვს -x პარამეტრს. მნიშვნელობად უნდა მიუთითოთ nul (format nul).

owner მფლობელი - არის ცხრილის ან წარმოდგენის მფლობელის სახელი. თუ ოპერაციის განმახორციელებელი მომხმარებელი არის მითითებული ცხრილის ან წარმოდგენის მფლობელი, პარამეტრი შეიძლება არ მიეთითოს, სხვა შემთხვევაში SQL Server აბრუნებს შეცდომის შეტყობინებას და ოპერაცია გაუქმდება.

bcp-ს არგუმენტები

6

"**query**" არის Transact-SQL მოთხოვნა, რომელიც აბრუნებს შედეგების ნაკრებებს. თუ მოთხოვნა აბრუნებს მრავალ შედეგების ნაკრებს, მონაცემების ფაილში კოპირდება მხოლოდ პირველი შედეგების ნაკრები (შემდეგი შედეგების ნაკრებები იგნორირდება). მოთხოვნაში გამოიყენება ორმაგი ბრჭყალები, ხოლო მოთხოვნაში ჩასმული გამოსახულებებისათვის გამოიყენება ბრჭყალები. მოთხოვნიდან მონაცემთა მასიური კოპირებისას უნდა იქნეს მითითებული ასევე არგუმენტი `queryout`.

მოთხოვნას შეუძლია მიმართოს შენახულ პროცედურას მხოლოდ იმ შემთხვევაში, თუ შენახულ პროცედურაში მითითებული ყველა ცხრილი აშკარად არსებობს, მანამ სანამ `bcp` გაეშვება.

table_name ცხრილის_სახელი - არის დანიშნულების ცხრილის სახელი SQL Server-ში მონაცემების იმპორტირებისას (**in**), და წყარო ცხრილის SQL Server-დან მონაცემების ექსპორტისას (**out**).

view_name წარმოდგენის_სახელი - არის SQL Server (**in**) მონაცემების კოპირებისას დანიშნულების წარმოდგენის სახელი და SQL Server (**out**) მონაცემების კოპირებისას წყარო-წარმოდგენის. მიზნობრივი წარმოდგენად შეიძლება გამოყენებულ იქნეს მხოლოდ წარმოდგენები, რომლების ყველა სვეტი ერთსა და იმავე ცხრილს ემყარება.

bcp-ს არგუმენტები

7

-a packet_size - პაკეტის_ზომა განსაზღვრავს Server-დან Server-ზე გაგზავნილ ქსელის თითო პაკეტში ბაიტთა რაოდენობას. SQL Server-ის კონფიგურაციის პარამეტრის დაყენება შესაძლებელია SQL Server Management Studio (ან sp_configure სისტემის შენახული პროცედურის) გამოყენებით. packet_size შეიძლება იყოს 4096-დან 65535 ბაიტამდე (ნაგულისხმევია 4096). პაკეტის გაზრდილმა ზომამ შეიძლება დაასწრაფოს ოპერაციების შესრულება, ხოლო თუ უფრო დიდი პაკეტია მოთხოვნილი, რომლის გაცემა შეუძლებელია, გამოყენებული იქნება ნაგულისხმევი მნიშვნელობა.

-b batch_size - პარტიის ზომა მიუთითებს იმპორტირებული მონაცემებში ერთ პაკეტში მწკრივების რაოდენობას. თითოეული პარტია იმპორტირდება და რეგისტრირდება როგორც ცალკე ოპერაცია, რომელიც ფიქსირდება სრული პაკეტის იმპორტირების დასრულებისას. შეთანხმებით მონაცემთა ფაილიდან ყველა სტრიქონის იმპორტირება ხორციელდება ერთ პარტიაში. მწკრივების მრავალ პარტიის შორის განაწილებისათვის უნდა მიეთითოს batch_size, რომელიც ნაკლებია მონაცემთა ფაილში მწკრივების რაოდენობაზე. თუ რაიმე პარტიის ტრანზაქცია ჩაიშალა, მხოლოდ მიმდინარე პარტიის ჩანართები ბრუნდება უკან. წარმოქმნილი შეცდომა არ შეეხება უკვე დაფიქსირებულ ტრანზაქციებს (არ შეიძლება ამ პარამეტრის გამოყენება -h პარამეტრთან ერთად "ROWS_PER_BATCH = bb").

bcp-ს არგუმენტები

8

-c - ასრულებს ოპერაციას სიმბოლური (ტექსტური) მონაცემთა ტიპის გამოყენებით. ეს პარამეტრი არ ითხოვს მონაცემთა ტიპის მითითებას თითოეულ სვეტისათვის.

მონაცემთა შენახვისათვის გაიყენებს ტიპს char, პრეფიქსების გარეშე და \t (ტაბულაციის სიმბოლო), ველების გამყოფად, ხოლო სტრიქონის დასრულების და ახალ სტრიქონზე გადასვლისათვის გამოიყენება \r \n. **-c** არ არის თავსებადი -w-სთან.

-C { ACP | OEM | RAW | code_page } - განსაზღვრავს მონაცემთა კოდის გვერდს მონაცემთა ფაილში. code_page მნიშვნელოვანია მხოლოდ იმ შემთხვევაში, თუ მონაცემები შეიცავს char, varchar ან text სვეტებს სიმბოლოების მნიშვნელობებით 127-ზე მეტი ან 32-ზე ნაკლები. რეკომენდირებულია ფორმატირების ფაილში მითითებულ იქნეს სორტირების პარამეტრები ყველა სვეტისათვის, გარდა იმ შემთხვევისა, როდესაც სასურველია 65001 პარამეტრის უპირატესობა სორტირების ან კოდის გვერდის სპეციფიკაციასთან.

-d **მონაცემთა_ბაზის_სახელი** - განსაზღვრავს მონაცემთა ბაზას, რომელსაც უნდა მიუერთდეს. შეთანხმებით bcp.exe უერთდება მომხმარებლის მონაცემთა ბაზას. თუ მითითებულია -d მონაცემთა_ბაზის_სახელი და სამი ნაწილის სახელი (database_name.schema.table როგორც bcp.exe- ს პირველი პარამეტრი), მოხდება შეცდომა, რადგან მონაცემთა ბაზის სახელს ორჯერ ვერ მიუთითებთ. თუ მონაცემთა_ბაზის_სახელი იწყება დეფისით (-) ან გადაკვეთის ხაზით (/), არ უნდა დაემატოს ადგილი -d და მონაცემთა ბაზის სახელს შორის.

bcp-ს არგუმენტები

9

-D - იწვევს bcp-ს **-S** პარამეტრისთვის გადაცემული მნიშვნელობის ინტერპრეტაციას, როგორც მონაცემთა წყაროს სახელს (DSN).

-e err_file - განსაზღვრავს შეცდომის ფაილის სრულ გზას, რომელიც გამოიყენება იმ მწკრივების შესანახად, რომელსაც bcp ვერ გადასცემს ფაილიდან მონაცემთა ბაზაში. შეცდომის შეტყობინებები bcp ბრძანებიდან გადადის მომხმარებლის სამუშაო სადგურზე. თუ ეს პარამეტრი არ არის გამოყენებული, შეცდომის ფაილი არ შეიქმნება. თუ err_file იწყება დეფისით (-) ან წინა გადაკვეთით (/), არ უნდა იქნეს გამოყენებული სივრცე -e-ს და err_file მნიშვნელობას შორის.

-E - მიუთითებს, რომ იდენტიფიკატორების მნიშვნელობა ან მნიშვნელობები იმპორტირებულ მონაცემთა ფაილში გამოყენებული იქნება იდენტიფიკატორების სვეტისთვის. თუ -E არგუმენტი არ არის მითითებული, იმპორტირებული მონაცემების ფაილში ამ სვეტის იდენტიფიკატორის მნიშვნელობები იგნორირებულია და SQL Server-ი ავტომატურად ანიჭებს უნიკალურ მნიშვნელობებს ცხრილის შექმნისას მითითებულ საწყის და ნაზარდის მნიშვნელობებზე დაყრდნობით. დამატებითი ინფორმაციისთვის იხილეთ DBCC CHECKIDENT. თუ მონაცემთა ფაილი არ შეიცავს იდენტიფიკატორის სვეტის მნიშვნელობებს ცხრილში ან წარმოდგენაში უნდა იქნეს გამოყენებული ფორმატის ფაილი, რომელშიც მიეთითება, რომ მონაცემების იმპორტისას გამოტოვებულია ცხრილის ან წარმოდგენის იდენტიფიკატორის სვეტი. SQL Server-ი ავტომატურად მიანიჭებს უნიკალურ მნიშვნელობებს სვეტს. -E პარამეტრის გამოსაყენებლად საჭიროა სპეციალური ნებართვები.

bcp-ს არგუმენტები

10

-f format_file - განსაზღვრავს ფორმატის ფაილის სრულ გზას. ამ პარამეტრის მნიშვნელობა დამოკიდებულია გარემოზე, რომელშიც ის გამოიყენება, შემდეგი წესების დაცვით:

- თუ **-f** პარამეტრი გამოიყენება **format** პარამეტრთან ერთად, მითითებული ცხრილისთვის ან წარმოდგენისთვის იქმნება ფაილი **format_file**. XML ფორმატის ფაილის შესაქმნელად, უნდა მიეთითოს **-x** პარამეტრი.
- თუ გამოიყენება **in** ან **out** პარამეტრი, **-f**-სთვის საჭიროა არსებობდეს ფორმატის ფაილი (არ არის ფორმატის ფაილის გამოყენების აუცილებლობა **in** ან **out** პარამეტრით. თუ არ არის მითითებული **-f**, **-n**, **-c**, **-w** ან **-N** პარამეტრი სისტემა მოითხოვს ფორმატირების ინფორმაციას და იძლევა პასუხების დამახსოვრების შესაძლებლობას ფორმატის ფაილში (ფაილის ნაგულისხმევი სახელია **bcp.fmt**)).
- თუ **format_file** იწყება დეფისით (-) ან გადაკვეთის ხაზი (/), მაშინ **-f** და **format_file** მნიშვნელობებს შორის არ უნდა იქნეს სივრცე.

-F first_row - განსაზღვრავს პირველი სტრიქონის ნომერს (ექსპორტის და იმპორტის ორივე შემთხვევაში). ეს პარამეტრი უნდა აღემატებოდეს (>) 0-ს, მაგრამ ნაკლებია (<) ან ტოლია (=) მთლიანი სტრიქონების რაოდენობისა. ამ პარამეტრის არარსებობის შემთხვევაში, ფაილის ნაგულისხმევად მიიჩნევა პირველი სტრიქონი. **first_row** შეიძლება იყოს დადებითი მთელი რიცხვი, რომლის მნიშვნელობა არ აღემატება $2^{63}-1$ -ს და იწყება 1-დან.

bcp-ს მაგალითები

11

-G - ეს გადამრთველი გამოიყენება Azure SQL Database ან Azure SQL Data Warehouse-თან მიერთებისას Azure Active Directory-ს ავთენტიფიკაციის გამოყენებით. **-G**-ს გამოყენებისათვის საჭიროა ვერსია [14.0.3008.27](#) ვერსია ან უფრო ახალი (ვერსიის დასადგენად შეასრულეთ **bcp -v** (AAD ინტეგრირებული და ინტერაქტიური ავთენტიფიკაცია ამჟამად არ არის მხარდაჭერილი Linux- სა და macOS- ში). იმისთვის, რომ შემოწმდეს, მხარდაჭერილია თუ არა არსებული bcp ვერსიაში Azure Active Directory (AAD) უნდა იქნეს შეყვანილი „**bcp - -**“ (bcp <space> <dash> <dash>) და ხელმისაწვდომი არგუმენტების სიაში უნდა იყოს **-G**.

განვიხილოთ მაგალითები Azure Active Directory-ს ავთენტიფიკაციის გამოყენებით:

Azure Active Directory-ს მომხმარებლის სახელისა და პაროლის გამოსაყენებლად შესაძლებელია მიეთითოს **-G** პარამეტრი და ასევე პარამეტრები **-U** (მომხმარებლის სახელი) და **-P** (პაროლი).

მაგალითი 1. მომხმარებლის სახელით და პაროლით: განხორციელდეს SQL Server-ის aadserver.database.windows.net მონაცემთა ბაზის testdb მონაცემთა ცხრილის bcptest ექსპორტი მონაცემთა ფაილში c:\last\data1.dat და გამოვიყენოთ Azure AD მომხმარებლის სახელი და პაროლი, რომლებიც არიან AAD-ს სააღრიცხვო მონაცემები:

```
bcp bcptest out "c:\last\data1.dat" -c -t -S  
aadserver.database.windows.net -d testdb -G -U  
admin@aadserver -P Pa$$w0rd
```

bcp-ს მაგალითები

12

მაგალითი 2. Azure Active Directory-ში ინტეგრირებული ავთენტიფიკაცია: Azure Active Directory ინტეგრირებული ავთენტიფიკაციისთვის საჭიროა გამოყენებულ იქნეს -G პარამეტრი მომხმარებლის სახელისა და პაროლის გარეშე (ამ კონფიგურაციაში იგულისხმება, რომ Windows-ის მომხმარებლის მიმღინარე ანგარიში, რომლშიც ხორციელდება bcp ბრძანება ჩართულია ფედერაციაში Azure AD-თან): მაგალითში განხორციელდეს ექსპორტი Azure AD-სთან ინტეგრირებული ანგარიშის გამოყენებით. ექსპორტირდება aadserver.database.windows.net SQL Server-ის testdb მონაცემთა ბაზის bcptest ცხრილი Azure AD ინტეგრირებული ავთენტიფიკაციის გამოყენებით მონაცემთა ფაილში c:\last\data2.dat:

```
bcp bcptest out "c:\last\data2.dat" -S aadserver.database.windows.net -d testdb -G -c -t
```

მაგალითი 3. განხორციელდეს იმპორტი Azure AD-სთან ინტეგრირებული ანგარიშის გამოყენებით. იმპორტირდება aadserver.database.windows.net SQL Server-ის testdb მონაცემთა ბაზის bcptest ცხრილში Azure AD ინტეგრირებული ავთენტიფიკაციის გამოყენებით მონაცემთა ფაილიდან c:\last\data2.dat:

```
bcp bcptest in "c:\last\data2.dat" -S aadserver.database.windows.net -d testdb -G -c -t
```

bcp-ს მაგალითები

13

მაგალითი 4. Azure Active Directory-ში ინტერაქტიური ავთენტიფიკაცია: Azure Active Directory ინტერაქტიური ავთენტიფიკაციისთვის საჭიროა პაროლის ხელით შეყვანა გამოყენებულ იქნეს **-G** პარამეტრი მხოლოდ მომხმარებლის სახელით (-U), პაროლის გარეშე. მაგალითში ექსპორტი ხდება Azure AD ინტერაქტიური რეჟიმის გამოყენებით, მომხმარებლის სახელის მითითებით, სადაც მომხმარებელი წარმოადგენს AAD ანგარიშს. ინტერაქტიური რეჟიმისთვის საჭიროა პაროლის ხელით შეყვანა. მრავალ ფაქტორიანი ავთენტიფიკაციის მქონე ანგარიშებისთვის უნდა იქნეს გამოყენებული MFA-ს ავთენტიფიკაციის მეთოდი:

```
bcp bcptest out "c:\last\data1.dat" -c -t -S  
aadserver.database.windows.net -d testdb -G -U  
alice@aadtest.onmicrosoft.com
```

მაგალითი 5. Azure AD მომხმარებელი არის დომენის ფედერაციული მომხმარებელი სისტემა Windows ანგარიშით. ბრძანების სტრიქონში მომხმარებლის სახელი შეიცავს დომენის სააღრიცხვო ჩანაწერს (მაგალითად, joe@contoso.com):

```
bcp bcptest out "c:\last\data1.dat" -c -t -S  
aadserver.database.windows.net -d testdb -G -U joe@contoso.com
```

თუ სტუმარი მომხმარებლები არსებობენ კონკრეტულ Azure AD-ში და ისინი არიან ჯგუფის ნაწილი, რომელიც არსებობს SQL მონაცემთა ბაზაში და რომლებსაც აქვთ მონაცემთა ბაზის ნებართვები bcp ბრძანების შესასრულებლად, მაშინ გამოიყენება მათი სტუმრის მომხმარებლის ფსევდონიმი (მაგალითად, [*](#)).

bcp-ს პარამეტრები

14

გავაგრძელოთ პარამეტრების განხილვა.

-h "load hints [, ... n]" - განსაზღვრავს ცხრილში მონაცემების დიდი მოცულობის იმპორტის დროს გამოყენებისათვის ერთს ან რამდენიმე მინიჭნებას:

- **ORDER(column[ASC | DESC] [...n])** - მონაცემთა სორტირება მონაცემთა ფაილში. იმპორტის წარმადობა გაუმჯობესდება, თუ იმპორტირებული მონაცემები დალაგებულია ცხრილის კლასტერული ინდექსის მიხედვით (ასეთის არსებობის შემთხვევაში). თუ მონაცემთა ფაილის მონაცემები დალაგებულია კლასტერული ინდექსის გასაღებისაგან განსხვავებული თანმიმდევრობით, ან თუ მონაცემთა ცხრილს არ გააჩნია კლასტერული ინდექსი, ORDER პუნქტი იგნორირებულ უნდა იქნეს. დანიშნულების ცხრილში უნდა იქნეს მითითებული სვეტების სახელები. სტანდარტულად, bcp მიიჩნევს, რომ მონაცემთა ფაილში მონაცემები არ არიან დალაგებულნი. SQL Server ოპტიმიზირებული იმპორტისთვის ასევე ამონმებს იმპორტირებული მონაცემების სორტირებას.
- **ROWS_PER_BATCH = bb** - მონაცემთა სტრიქონების რაოდენობა თითო პაკეტში (bb). გამოიყენება, როდესაც **-b** არ არის მითითებული, რის შედეგადაც მონაცემთა მთელი ფაილი SQL Server-ზე იგზავნება როგორც ერთი ტრანზაქცია. SQL Server-ი ახორციელებს მასიური დატვირთვის ოპტიმიზაციას bb მნიშვნელობის შესაბამისად. სტანდარტულად, ROWS_PER_BATCH უცნობია.

bcp-ს პარამეტრები

15

- **KILOBYTES_PER_BATCH = cc** - მონაცემების სავარაუდო კილობაიტების (cc) მოცულობა თითო ჰაკეტში. სტანდარტულად, KILOBYTES_PER_BATCH უცნობია.
- **TABLOCK** - განსაზღვრავს ბლოკირებას ცხრილის დონეზე მასიური ჩატვირთვის ოპერაციის განხორციელებისას (წინააღმდეგ შემთხვევაში ბლოკირება ხორციელდება მონაცემთა სტრიქონის დონეზე). ამ შემთხვევაში წარმადობა საგრძნობლად იზრდება, რადგან თითოეული სტრიქონისათვის არ ელოდება ბლოკირების რიგს. თუ ცხრილს არ აქვს ინდექსები და მითითებულია TABLOCK, მაშინ შეიძლება ერთდროულად ჩაიტვირთოს მრავალმა კლიენტმა. სტანდარტულად, დაბლოკვა განისაზღვრება ცხრილის პარამეტრით **table lock on bulkload**.
- **CHECK_CONSTRAINTS** - სამიზნე ცხრილის ან წარმოდგენის ყველა შეზღუდვა უნდა შემოწმდეს მასიური იმპორტის ოპერაციის განხორციელებისას. CHECK_CONSTRAINTS მინიჭნების გარეშე, ნებისმიერი CHECK და FOREIGN KEY შეზღუდვები იგნორირებულია, ხოლო ოპერაციის შემდეგ ცხრილზე არსებული შეზღუდვა აღინიშნება, როგორც არასანდო (გამონაკლისია UNIQUE, PRIMARY KEY და NOT NULL, რომლებიც ყოველთვის მოწმდებიან). აქვე უნდა აღინიშნოს, რომ - **m max_errors** პარამეტრი არ გამოიყენება შეზღუდვების შესამოწმებლად.
- **FIRE_TRIGGERS** - გამოიყენება **in** არგუმენტთან ერთად და მიუთითეს, რომ მასიური კოპირებისას იმუშავებენ მიზნობრივი ცხრილის Insert ტრიგერები.

bcp-ს პარამეტრები

16

- i *input_file*** - ინტერაქტიურ რეჟიმის დროს მასიურ გადაწერისას განსაზღვრავს პასუხების ფაილის სახელს, რომელშიც მოცემულია მონაცემთა თითოეული სვეტისათვის ბრძანების სტრიქონის კითხვებზე პასუხები, თუ არ არის მითითებული -n, -c, -w, ან -N პარამეტრები. თუ *input_file* იწყება დეფისით (-) ან წინა გადაკვეთით (/), არ უნდა იქნეს შეყვანილი სივრცე -i-სა და *input_file* მნიშვნელობებს შორის.
- k** - განსაზღვრავს, რომ ოპერაციის დროს ცარიელი სვეტები უნდა ინარჩუნებდნენ მნიშვნელობას Null და არა სვეტების რაიმე ნაგულისხმევ მნიშვნელობას.
- K *application_intent*** - აცხადებს პროგრამის დატვირთვის ტიპს SQL Server-თან მიერთებისას. ერთადერთი მნიშვნელობა, რაც შესაძლებელია, არის ReadOnly.
- l *login_timeout*** - განსაზღვრავს SQL Server-ში შესვლის შეყოვნებას (წამებში). შეყოვნების ნაგულისხმევი დროა 15 წამი (შესაძლებელია 0-დან 65534-მდე - სხვა შემთხვევაში bcp წარმოქმნის შეცდომის შეტყობინებას). 0 მნიშვნელობა განსაზღვრავს უსასრულო შეყოვნებას.
- L *last_row*** - განსაზღვრავს ბოლო სტრიქონის ნომერს (ცხრილიდან ექსპორტისათვის ან მონაცემთა ფაილიდან იმპორტისათვის). ამ პარამეტრის მნიშვნელობა უნდა იყოს მეტი (>) 0-ზე, მაგრამ ნაკლები (<) ან ტოლი (=) ბოლო სტრიქონის ნომერზე. ამ პარამეტრის არარსებობის შემთხვევაში, შეთანხმებით გამოიყენება ამ ფაილის ბოლო სტრიქონი. *last_row* პარამეტრი შეიძლება იყოს დადგებითი მთელი რიცხვი $2^{63}-1$ მნიშვნელობამდე.

bcp-ს პარამეტრები

17

-m max_errors - განსაზღვრავს სინტაქსური შეცდომების მაქსიმალურ რაოდენობას, რაც შეიძლება მოხდეს bcp ოპერაციის გაუქმებამდე (სინტაქსურ შეცდომაში იგულისხმება შეცდომა, რომელიც წარმოიშვება მონაცემთა გარდაქმნისას სამიზნე მონაცემების ტიპები). *max_errors* გამორიცხავს შეცდომებს, რომელთა დადგენა მხოლოდ SQL Server-ზე შეიძლება (მაგალითად, შეზღუდვების დარღვევა). მნკრივი, რომლის კოპირება შეუძლებელია bcp-ს გამოყენებით, იგნორირდება და ითვლება ერთ შეცდომად. თუ ეს პარამეტრი არ არის მითითებული, ნაგულისხმევია 10 მაქსიმალური შეცდომა.

-n - ახორციელებს მასობრივი კოპირების ოპერაციას საკუთარი მონაცემთა (მონაცემთა ბაზის) ტიპების გამოყენებით. ეს პარამეტრი არ ითხოვს თითოეულ სვეტისათვის მონაცემთა ტიპის განსაზღვრას, ის იყენებს საკუთარ მნიშვნელობებს.

-N - ახორციელებს მასობრივი კოპირების ოპერაციას საკუთარი მონაცემთა (მონაცემთა ბაზის) ტიპების გამოყენებით არასიმბოლური მონაცემთა ტიპებისათვის და უნიკოდის სიმბოლოებს სიმბოლურით. ეს პარამეტრი უფრო მაღალი წარმადობის ალტერნატივაა, ვიდრე **-w** და იგი არ ითხოვს მონაცემთა ტიპს თითოეულ სვეტისათვის. ამასთან ერთად იგი აუცილებლად უნდა იქნეს გამოყენებული ANSI გაფართოებულ სიმბოლოების (ნაციონალური ანბანის) არსებობისას მონაცემებში.

-o output_file - განსაზღვრავს გამომავალი ფაილის სახელს. თუ *output_file* იწყება დეფისით (-) ან წინა გადაკვეთით (/), არ შეიყვანოთ სივრცე **-o**-სა და *output_file* მნიშვნელობებს შორის.

bcp-ს პარამეტრები

18

-P password - განსაზღვრავს შესვლის პაროლს ID-სათვის. თუ ეს პარამეტრი არ არის, bcp ბრძანება ითხოვს პაროლს. თუ ეს პარამეტრი გამოიყენება ბრძანების სტრიქონის ბოლოს პაროლის გარეშე, bcp იყენებს ნაგულისხმევ პაროლს (NULL). თუ მომხმარებელს სურს bcp ბრძანების შეყვანისას პაროლის დაფარვა, მაშინ ბრძანებაში არ უნდა მიეთითოს -P პარამეტრი -U-სთან ერთად და bcp-ს გაშვებისას (დააჭირეთ ENTER-ს) და ბრძანება მოითხოვს პაროლს. თუ პაროლი იწყება დეფისით (-) ან გადაკვეთის ხაზით (/), არ უნდა იქნეს შეყვანილი სივრცე -P და password მნიშვნელობას შორის.

-q - ახორციელებს SET QUOTED_IDENTIFIERS ON ინსტრუქციას bcp-სა და SQL Server-ს შორის მიერთების დროს. ეს პარამეტრი გამოიყენება მონაცემთა ბაზის, მფლობელის, ცხრილის ან წარმოდგენის სახელის დასაზუსტებლად, რომელიც შეიცავს სივრცეს ან პწკარს (ერთმაგ ბრჭყალს). სახელი უნდა ჩაისვას ბრჭყალებში (""). -q არ ვრცელდება -d- ზე გადაცემულ მნიშვნელობებზე.

-r row_term - განსაზღვრავს სტრიქონის დამთავრეს. ნაგულისხმევი არის \n (ახალი ხაზის სიმბოლო). თუ row_term იწყება დეფისით (-) ან წინა ხაზი (/), არ უნდა იქნეს შეყვანილი სივრცე -r და row_term მნიშვნელობებს შორის.

-R - განსაზღვრავს SQL Server-ში ვალუტის, თარიღისა და დროის მონაცემებს ისე, როგორც კლიენტის კომპიუტერის სისტემის რეგიონალურ ფორმატშია მითითებული რეგიონალურ ფორმატში. სტანდარტულად, რეგიონალური პარამეტრები იგნორირებულია.

bcp-ს პარამეტრები

19

- S *server_name* [\i*nstance_name*]** - განსაზღვრავს SQL Server-ს, რომელთანაც ხორციელდება მიერთება bcp-თი. თუ SQL Server-ი არ არის მითითებული, bcp პროგრამა უერთდება ნაგულისხმევ SQL Server-ს (ადგილობრივ კომპიუტერში). ეს პარამეტრი საჭიროა, როდესაც bcp ბრძანება სრულდება დაშორებული კომპიუტერიდან ქსელში ან ლოკალური სახელობითი ინსტანციიდან.
- t *field_term*** - განსაზღვრავს ველის ბოლოს. ნაგულისხმევია \t (ტაბულაციის სიმბოლო). თუ *field_term* იწყება დეფისით (-) ან წინა ხაზით (/), არ უნდა იქნეს შეყვანილი სივრცე **-t**-სა და *field_term* მნიშვნელობებს შორის.
- T** - განსაზღვრავს SQL Server-თან სანდო მიერთებას bcp-თი ინტეგრირებული უსაფრთხოების გამოყენებით. თუ -T არ არის მითითებული, მაშინ უნდა იქნეს მითითებული -U და -P. აქვე უნდა ითქვას, რომ როდესაც bcp უერთდება SQL მონაცემთა ბაზას ან SQL მონაცემთა საცავს, Windows ავთენტიფიკაციის ან Azure Active Directory ავთენტიფიკაციის გამოყენება არ არის მხარდაჭერილი. გამოიყენეთ -U და -P პარამეტრები.
- U *login_id*** - განსაზღვრავს შესვლის ID, რომელიც გამოიყენება SQL Server-თან მისაერთებლად.
- v** - გამოსახავს bcp-ს ვერსიის ნომერს და საავტორო უფლებებს.

bcp-ს პარამეტრები

20

-v (80|90|100|110|120|130) - ასრულებს მასიური კოპირების ოპერაციას SQL Server-ის ადრინდელი ვერსიის მონაცემთა ტიპების გამოყენებით. ეს პარამეტრი არ ითხოვს თითოეულ ტიპს, იგი იყენებს მნიშვნელობებს შეთანხმებით. აქ იგულისხმება (80 = SQL Server 2000 (8.x); 90 = SQL Server 2005 (9.x); 100 = SQL Server 2008 და SQL Server 2008 R2; 110 = SQL Server 2012 (11.x); 120 = SQL Server 2014 (12.x); 130 = SQL Server 2016 (13.x)).

-w - ახორციელებს მასიური კოპირების ოპერაციას Unicode სიმბოლოების გამოყენებით. ეს პარამეტრი არ ითხოვს მონაცემთა ტიპს თითოეული ველისათვის, ის შენახვისას იყენებს nchar ტიპს, არ შეიცავს პრეფიქსიებს, \t (ტაბულაციის სიმბოლო) გამოიყენება ველების გამყოფად, ხოლო \n-ს (ახალი სტრიქონის სიმბოლოს) როგორც როგორც მწკრივების დამთავრების. -w არ არის თავსებადი -c-თან.

-x - გამოიყენება format და -f format_file პარამეტრებით XML-ზე დაფუძნებულ ფორმატის ფაილის შესაქმნელად (ნაგულისხმევად იქმნება არა XML ფორმატის ფაილი). პარამეტრი -x არ მუშაობს მონაცემთა იმპორტის ან ექსპორტისას, რომლის დროსაც წარმოიშობა შეცდომა.

Windows-ის ბრანდმაუერი (Firewall)

ბრანდმაუერის სისტემები განკუთვნილია კომპიუტერულ რესურსებზე არასანქციონირებულ წვდომის შესაფერხებლად. თუ ბრანდმაუერის სისტემა ჩართულია, მაგრამ არ არის კონფიგურირებული, ამან შესაძლებელია SQL Server-თან დაკავშირების მცდელობა დაბლოკოს. SQL Server-ზე ბრანდმაუერის გავლით წვდომისათვის, საჭიროა კომპიუტერის ოპერაციულ გარემოში ბრანდმაუერის კონფიგურირება. ბრანდმაუერი არის MS Windows-ის კომპონენტი, მაგრამ კომპიუტერზე შესაძლებელია სხვა კომპანიის ბრანდმაუერების ინსტალირება და კონფიგურირებაც. განვიხილოთ MS Windows-ის ბრანდმაუერი, თუმცა ძირითადი პრინციპები ყველა ბრანდმაუერს იგივე აქვს.

ბრანდმაუერი ამოწმებს დასმულ წესებთან შემომავალ პაკეტს და შესაბამისობის შემთხვევაში გადასცემს TCP/IP პროტოკოლს შემდგომი დამუშავებისთვის. თუ პაკეტი არ აკმაყოფილებს წესებით განსაზღვრულ სტანდარტებს ბრანდმაუერი უარყოფს პაკეტს (თუ ჟურნალის წარმოება ჩართულია, შესაბამისი ჩანაწერი იწერება ბრანდმაუერის ჟურნალის ფაილში).

ნებადართული ტრაფიკის სია შესაძლებელია შევსებულ იქნეს შემდეგი გზებით:

- ▶ **ავტომატურად** - როდესაც კომპიუტერის ბრანდმაუერი ჩართულია და იგი იწყებს მიერთების ინიცირებას. ამ შემთხვევაში ბრანდმაუერი ამატებს ჩანაწერს სიაში თუ მიერთება დასაშვებია;
- ▶ **ხელით** - ადმინისტრატორი თვითონ ახორციელებს გამონაკლისების კონფიგურირებას ბრანდმაუერისათვის. ამ შემთხვევაში უნდა განხორციელდეს ბრანდმაუერის კონფიგურირება SQL Server-თან მიერთებისათვის.

Windows-ის ბრანდმაუერი (Firewall)

ბრანდმაუერის სტრატეგიის შერჩევა უფრო რთულია, ვიდრე უბრალოდ გადაწყვეტილების მიღება მოცემული პორტების გახსნა/დახურვასთან დაკავშირებით. ბრანდმაუერის სტრატეგიის შერჩევა უნდა განხორციელდეს დაწესებულების სერვერებისა და ქსელების ადმინისტრატორებთან და კიბერუსაფრთხოების სამსახურთან ერთობლივი გადაწყვეტილებების საფუძველზე.

MS Windows ბრანდმაუერის პარამეტრები შესაძლებელია დაკონფიგურირდეს Microsoft Management Console-ს (MMC) ან netsh-ის გამოყენებით. MMC იძლევა მარტივ ფორმაში ბრანდმაუერის კონფიგურირების ასევე გაძლიერებული უსაფრთხოების დამატებითი პარამეტრებისათვის. Netsh.exe-ს გამოყენებით ადმინისტრორს ეძლევა შესაძლებლობა განახორციელოს ბრანდმაუერის კონფიგურირება Windows-ის PowerShell-ის ბრძანებათა სტრიქონის გამოყენებით. ამ შემთხვევაში ბრძანებები გადაიმისამართება შესაბამის დამხმარე აპლიკაციებთან (.dll), რომლებიც ასრულებენ მათ. netsh-ის დამხმარე აპლიკაციები უზრუნველყოფენ სერვისების, სამომხმარებლო პროგრამებისა და ოქმების კონფიგურირებას, მონიტორინგს და მხარდაჭერას.

მაგალითი 1:

Netsh-ის გამოყენებით გავხსნათ TCP-ს 1433 პორტი:

```
netsh firewall set portopening protocol = TCP port = 1433 name = SQLPort mode = ENABLE scope = SUBNET profile = CURRENT
```

Windows-ის ბრანდმაუერი (Firewall)

3

მაგალითი 2:

Netsh-ით გავხსნათ გაძლიერებული უსაფრთხოების გამოყენებით TCP-ს 1433 პორტი:

```
netsh advfirewall firewall add rule name = SQLPort dir = in protocol = tcp action = allow  
localport = 1433 remoteip = localsubnet profile = DOMAIN
```

მაგალითი 3:

გახსნათ TCP პორტი 1433 და UDP პორტი 1434 SQL Server-ის ნაგულისხმევი ეგზემპლიარისათვის და SQL Server-ის ბრაუზერის სერვისი:

```
New-NetFirewallRule -DisplayName "SQLServer default instance" -Direction Inbound -  
LocalPort 1433 -Protocol TCP -Action Allow
```

```
New-NetFirewallRule -DisplayName "SQLServer Browser service" -Direction Inbound -  
LocalPort 1434 -Protocol UDP -Action Allow
```

Windows-ის ბრანდმაუერში SQL Server-ის მონაცემთა ბაზის ძრავის პორტები

პორტები, რომლებიც გამოიყენება SQL Server-ის მონაცემთა ბაზის ძრავის მიერ

სტანდარტულად ტიპიური პორტები, რომლებიც გამოიყენება SQL Server-ისა და მონაცემთა ბაზის ძრავის დაკავშირებული სერვისები არიან: TCP პორტები 1433, 4022, 135 და 1434, ხოლო UDP - 1434. განვიხილოთ ეს პორტები:

- ▶ ეგზემპლიარი ნაგულისხმევად მომუშავე TCP ოქმით - 1433;
- ▶ სახელობითი ეგზემპლიარი ნაგულისხმევი პორტით - TCP პორტი არის დინამიური პორტი, რომელიც განსაზღვრულია მონაცემთა ბაზის ძრავის დაწყების დროს;
- ▶ სახელობითი ეგზემპლიარი ფიქსირებული პორტით - TCP პორტს აკონფიგურირებს ადმინისტრატორი;
- ▶ გამოყოფილი ადმინისტრაციული კავშირი - უნდა შემოწმდეს შეცდომის ჟურნალში პორტის ნომერი;
- ▶ SQL Server-ის ბრაუზერის სერვისი - UDP პორტი 1434;
- ▶ ეგზემპლიარი HTTP საბოლოო წერტილით - შეიძლება მითითებულ იქნეს HTTP საბოლოო წერტილის შექმნისას (ნაგულისხმევი TCP პორტი არის CLEAR_PORT ტრაფიკისათვის 80, ხოლო SSL_PORT ტრაფიკისათვის 443);

Windows-ის ბრანდმაუერში SQL Server-ის მონაცემთა ბაზის ძრავის პორტები

- ▶ კომპონენტი Service Broker - TCP პორტი 4022. გამოყენებული პორტის შესამოწმებლად შეასრულეთ შემდეგი მოთხოვნა:


```
SELECT name, protocol_desc, port, state_desc
FROM sys.tcp_endpoints
WHERE type_desc = 'SERVICE_BROKER';
```
- ▶ მონაცემთა ბაზის სარკისებური ანარევლი - ადმინისტრატორის მიერ არჩეული პორტი. პორტის დასადგენად, შეასრულეთ შემდეგი მოთხოვნა:

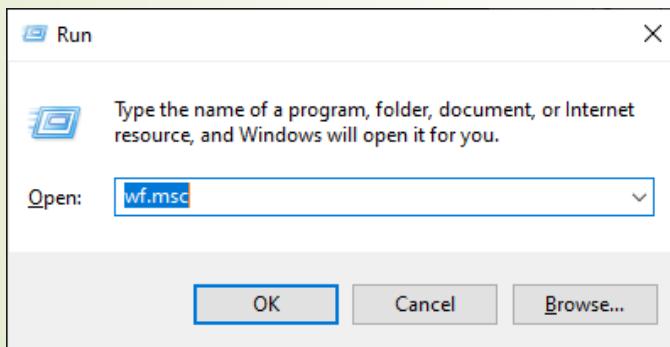

```
SELECT name, protocol_desc, port, state_desc
FROM sys.tcp_endpoints
WHERE type_desc = 'DATABASE_MIRRORING';
```
- ▶ რეპლიკაცია - ძირითადად TCP პორტი 1433. საწყისი მონაცემების და სქემის მიგრაციისათვის რეპლიკაცია ხორციელდება FTP ოქმით (TCP პორტი 21) ან სინქრონიზაცია HTTP-ით (TCP პორტი 80) ან ფაილების გაზიარებით UDP პორტი 137 და 138 და TCP პორტი 139 NetBIOS-თან ერთად გამოყენებისას. ფაილის გაზიარებისას გამოიყენება TCP პორტს 445;
- ▶ Transact-SQL-ის გამმართველი - TCP პორტი 135.

Windows-ის ბრანდმაუერში SQL Server-ის დინამიური პორტები

SQL Server-ის დინამიური პორტები

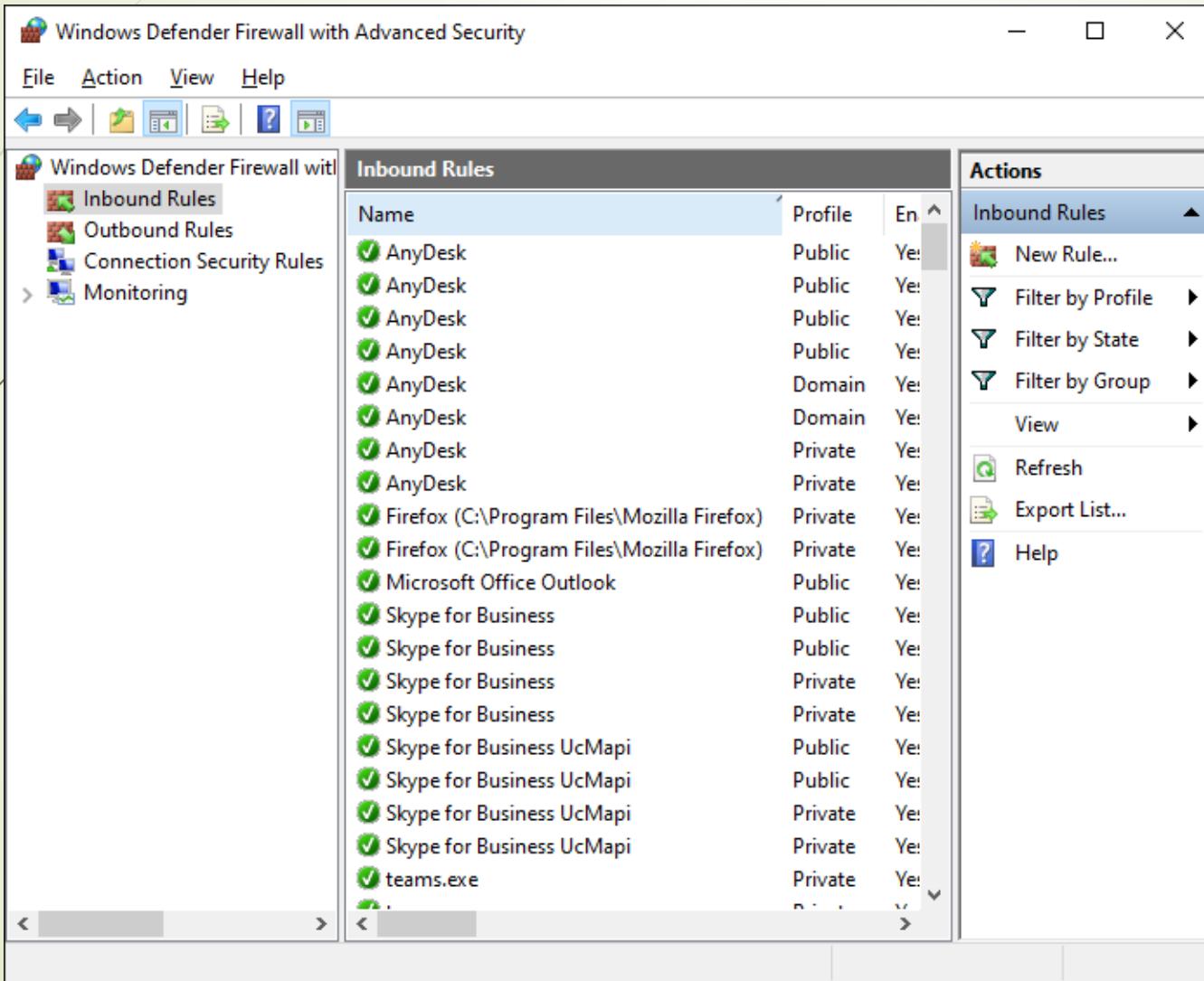
სტანდარტულად სახელობითი ეგზემპლიარები (მათ შორის SQL Server Express) იყენებენ დინამიურ პორტებს. ანუ SQL Server-ის მონაცემთა ბაზის ძრავი გაშვებისას პოულობს წვდომად პორტს და იყენებს მას. თუ მონაცემთა ბაზის ძრავის სახელობითი ეგზემპლიარი ერთადერთი ინსტანცია, ის უფრო გამოიყენებს TCP პორტს 1433. თუ მონაცემთა ბაზის ძრავის სხვა სახელობითი ეგზემპლიარებიც არიან, ისინი გამოიყენებენ სხვა წვდომად TCP პორტებს. რადგან ამ შემთხვევაში შესაძლებელია პორტი შეიცვალოს ყოველ ჯერზე და მნელია ბრანდმაუერის კონფიგურირება, ამ შემთხვევაში ჯობია ბრანდმაუერში კონფიგურირებულ იქნეს მუდმივი ფიქსირებული ან სტატიკური პორტები.

ბრანდმაუერში პროგრამის გამონაკლისის დამატება Windows Defender Firewall-ის გამოყენებით Advanced Security-ით



Windows Defender Firewall-ის გასაშვებათ Windows-ში უნდა იქნეს გაშვებული wf.msc, რისთვისაც შეიძლება გამოყენებულ იქნეს START-მენიუ ან ღილაკების კომბინაცია WIN+R, რომელშიც უნდა იქნეს აკრეფილი wf.msc და მაუსით გააჭტიურებული ღილაკი OK ან კლავიატურაზე დაჭრილი ღილაკი Enter.

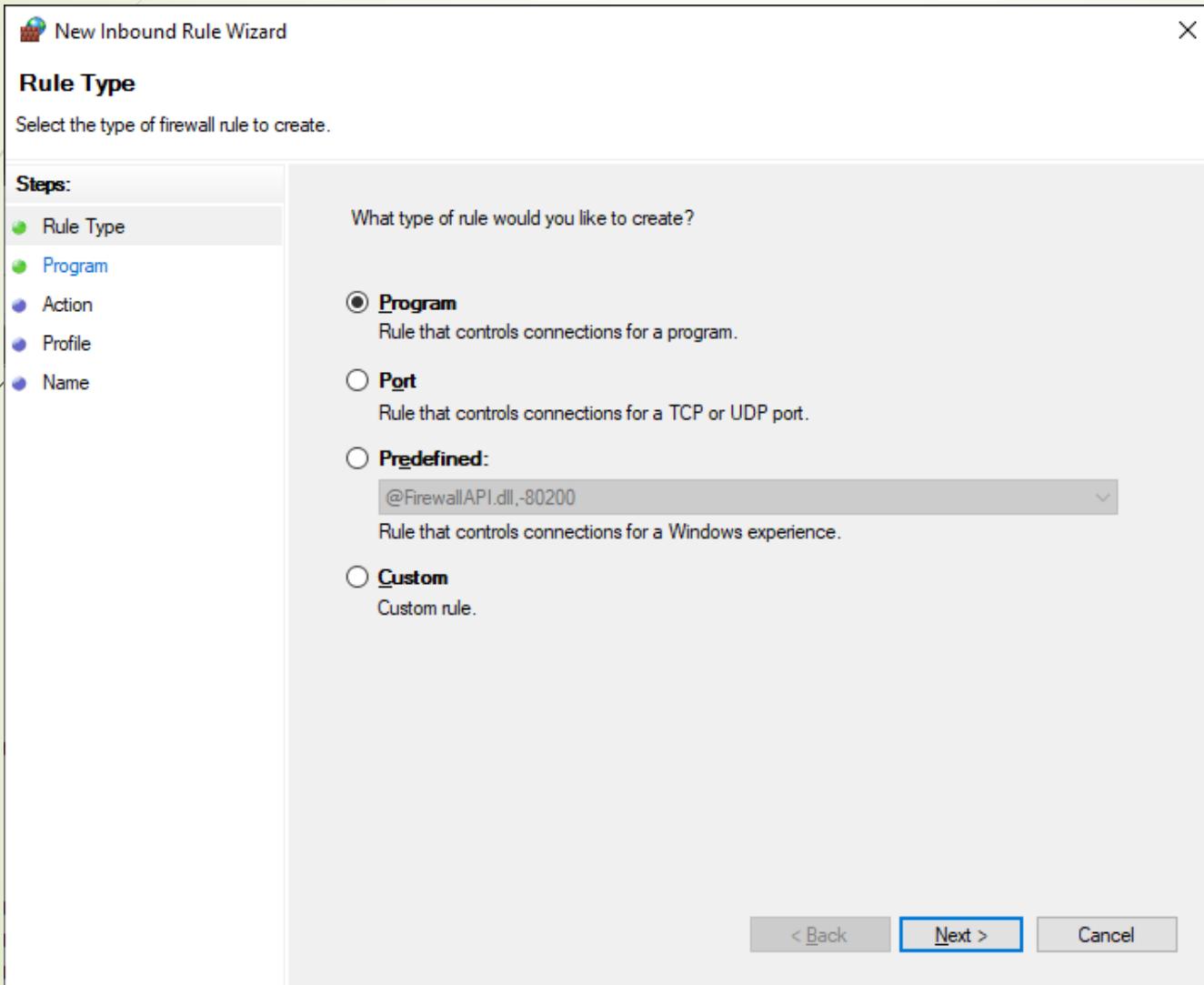
Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



მონიტორის ეკრანზე
გამოსახულ დიალოგურ
ფანჯრის მარცხენა
მიდამოში მომხმარებელმა
უნდა გააკტიუროს
პირველი სტრიქონი
„შემომავალი წესები“
(Inbound Rules).

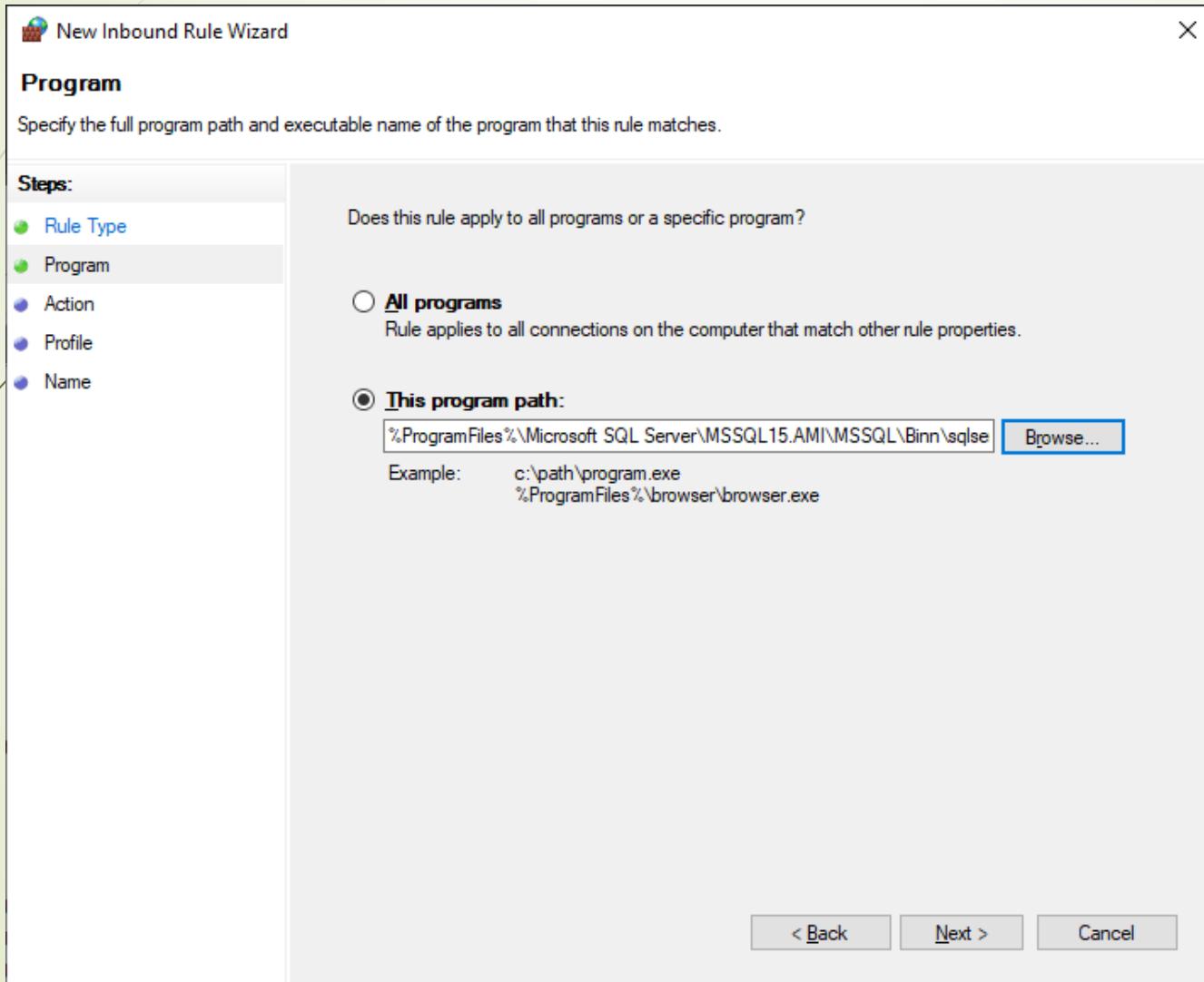
„შემომავალი წესები“-ს
შერჩევის შემდეგ
დიალოგური ფანჯრის
მარჯვენა მიდამოში
მომხმარებელმა უნდა
გააკტიუროს „შემომავალი
წესებში“ (Inbound Rules)
პირველი სტრიქონი „ახალი
წესი...“ (New Rule...).

Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



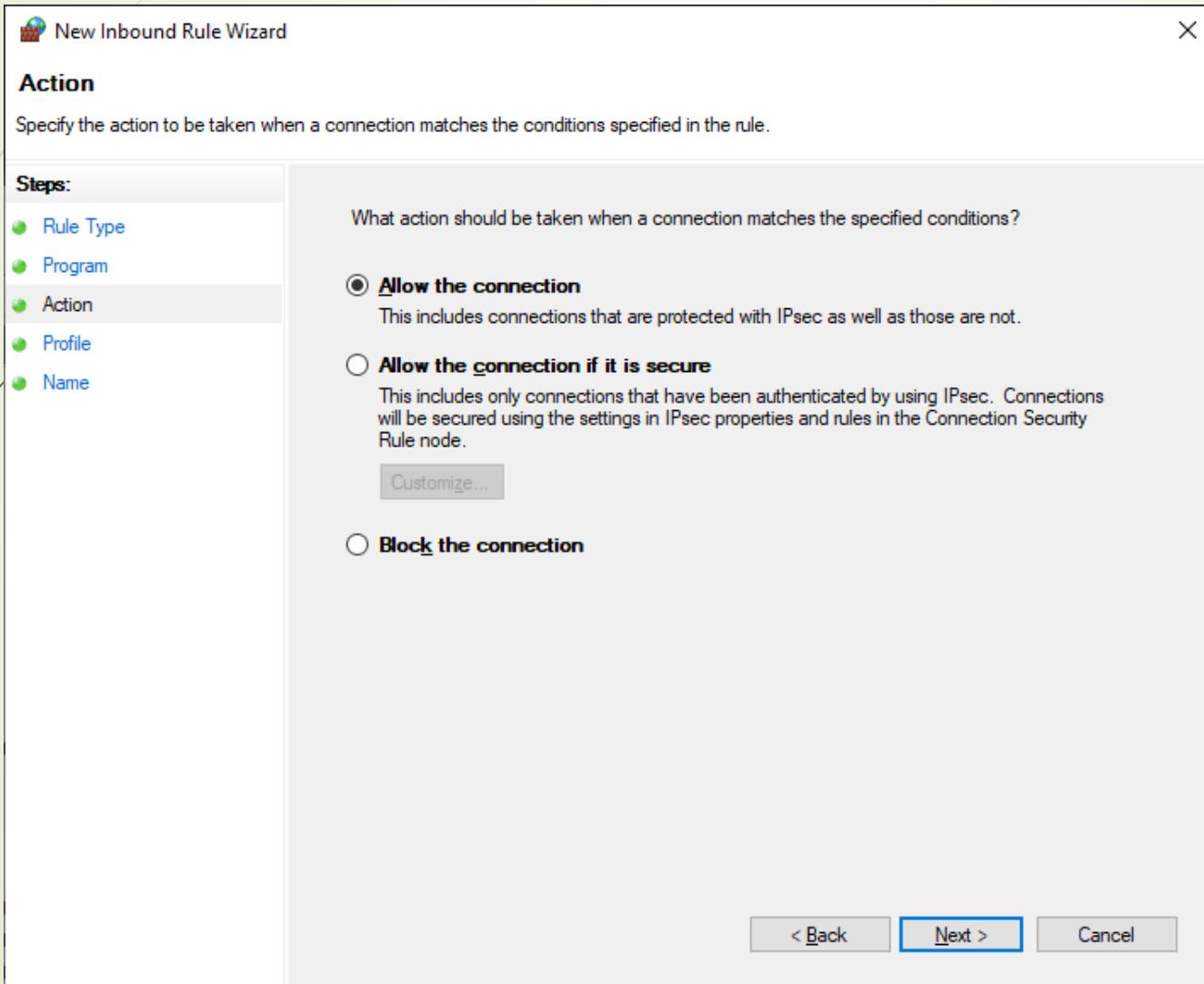
„ახალი წესი...“-ს სტრიქონის შერჩევის შემდეგ მონიტორის ეკრანზე გამოისახება „ახალი შემომავალი წესის ოსტატი“-ს (New Inbound Rule Wizard) დიალოგური ფანჯარა, რომლის მარცხენა მიღამოში გააქტიურებულ „წესის ტიპი“-ს (Rule Type) სტრიქონზე მარჯვენა მიღამოში მომხმარებელმა უნდა შეარჩიოს „პროგრამა“ (Program) და გადავიდეს შემდეგ დიალოგურ ფანჯარაზე „შემდეგი“ (Next) ღილაკით.

Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



შემდეგ დიალოგურ ფანჯარაში „პროგრამა“ (Program) მომხმარებელს ეძღვა შესაძლებლობა „ამ პროგრამის გზა“-ს (This program path) სტრიქო-ნში „დათვალიერების...“ (Browse...) ღილაკის გამოყენებით მიუთითოს გამშვები პროგრამა (ჩვენ შემთხვევაში ეს იქნება %ProgramFiles%\Microsoft SQL Server\MSSQL15.AMI\MSSQL\Binn\sqlservr.exe) და გადავიდეს შემდეგ დიალოგურ ფანჯარაზე „შემდეგი“ (Next) ღილაკის გამოყენებით.

Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



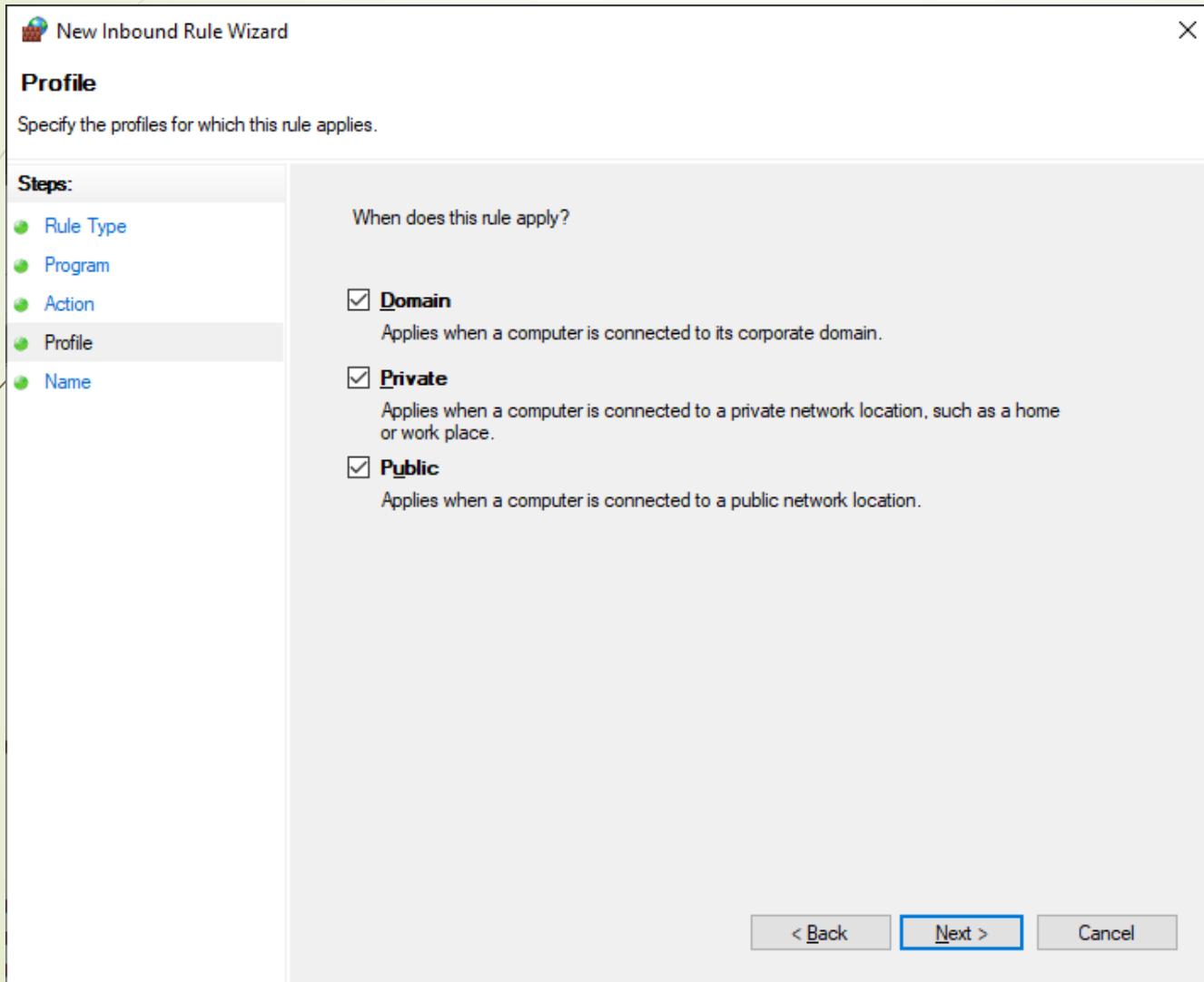
შემდეგ დიალოგურ
ფანჯარაში „აქცია“ (Action)
მომხმარებელს ეძღვა
შესაძლებლობა მარჯვენა
მიდამოში შეარჩიოს:

► „მიერთების დაშვება“
(Allow the connection);
► „მიერთების დაშვება თუ IT
არის დაცული“ (Allow the
Connection if it is secure);

► „მიერთების დაბლოკვა“
(Block the connection);

და გადავიდეს შემდეგ
დიალოგურ ფანჯარაზე
„შემდეგი“ (Next) ღილაკის
გამოყენებით.

Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება

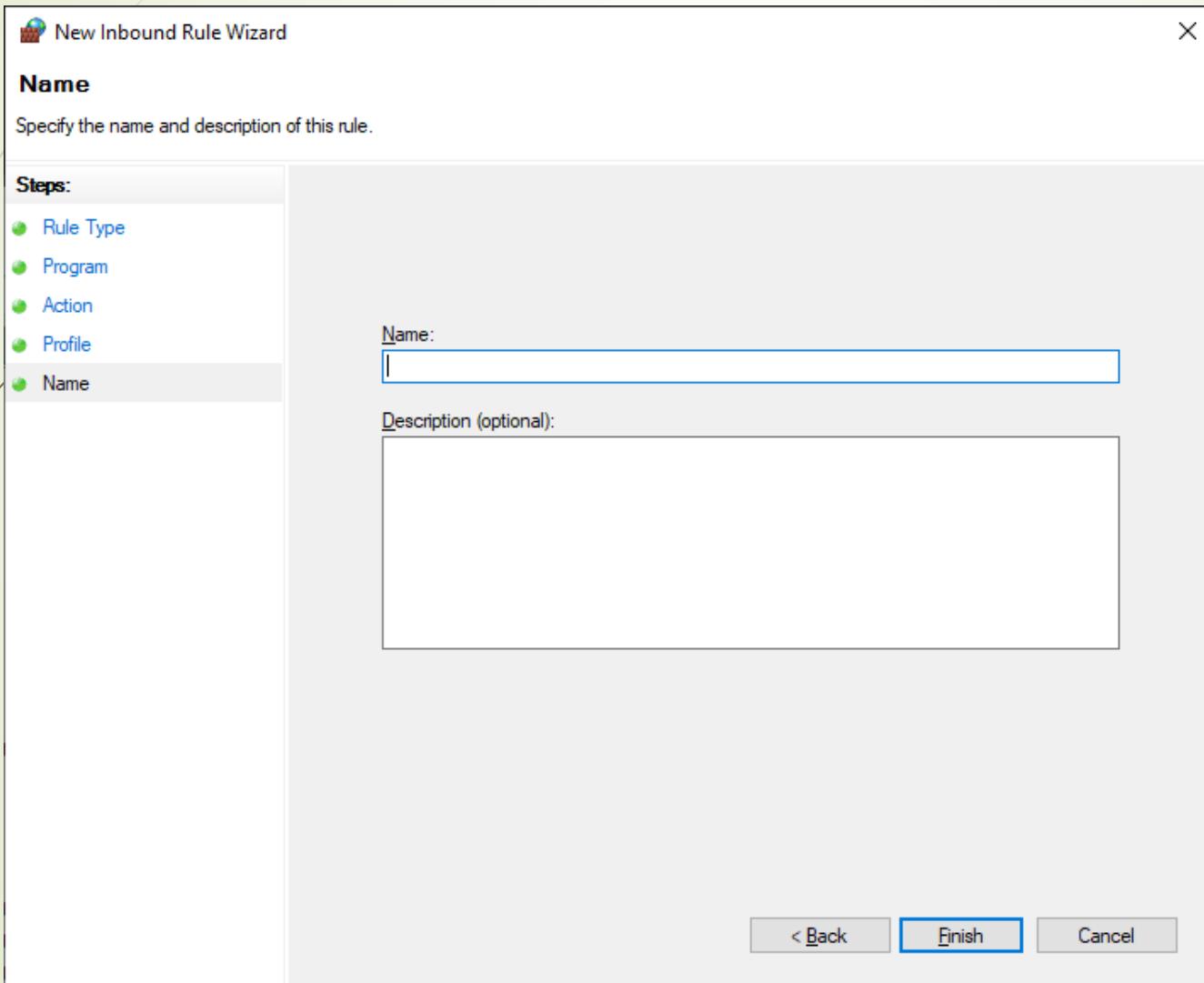


შემდეგ დიალოგურ ფანჯარაში „პროფილი“ (Profile) მომხმარებელს ეძლევა შესაძლებლობა მარჯვენა მიდამოში შეარჩიოს გამოყენების პროფილი:

- „დომენი“ (Domain) - კორპორატიული ქსელი;
- „პრივატული“ (Private) - სახლის ან სამსახურის ქსელი;
- „საჯარო“ (Public) - საჯარო ქსელი;

და გადავიდეს შემდეგ დიალოგურ ფანჯარაზე „შემდეგი“ (Next) ღილაკის გამოყენებით.

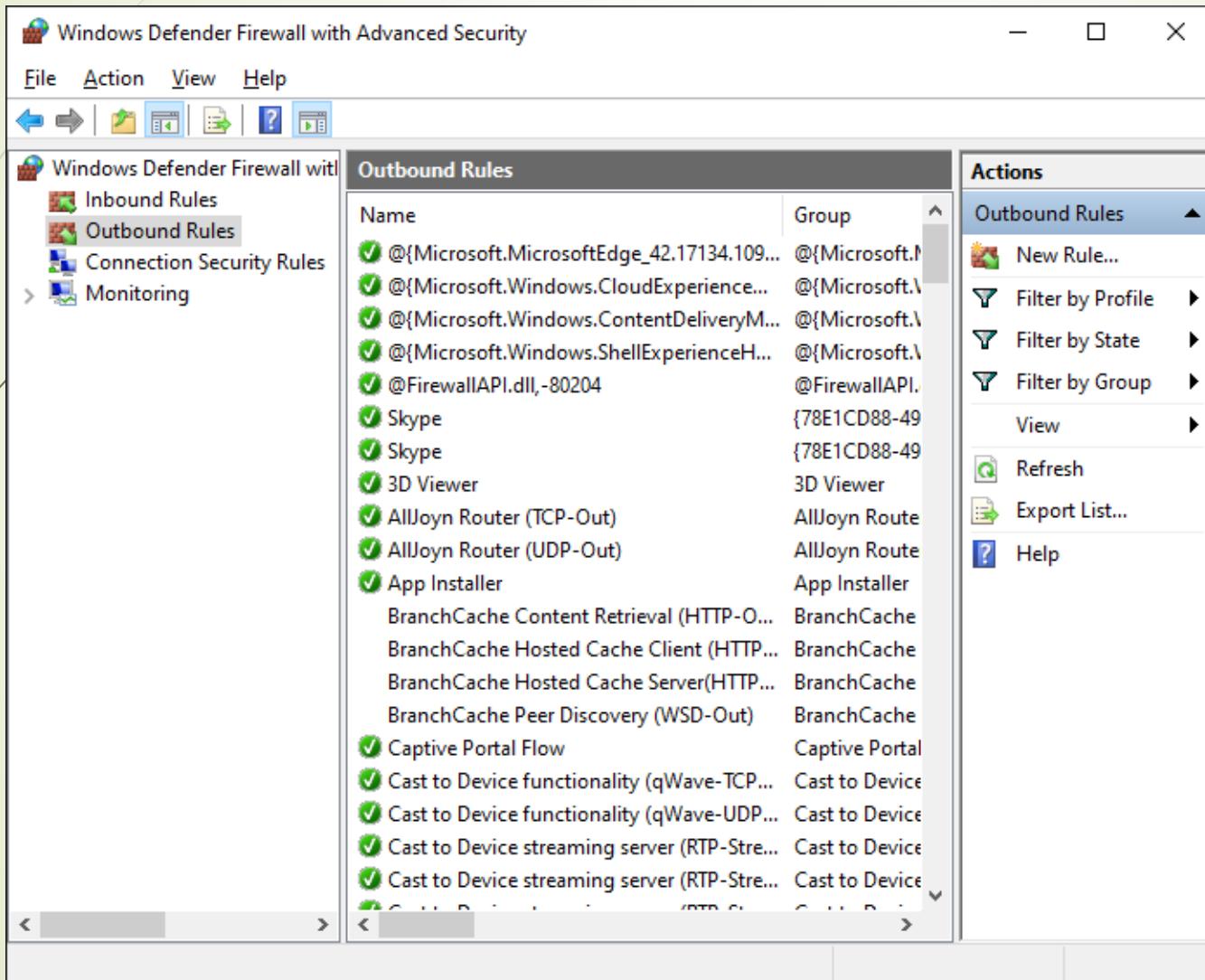
Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



შემდეგ დიალოგურ
ფანჯარაში „სახელი“ (Name)
მომხმარებელს ეძლევა
შესაძლებლობა მარჯვენა
მიდამოში შეარჩიოს წესის
„სახელი“ (Name) და
საჭიროების შემთხვევაში
„აღწერილობა“ (Description),
რის შემდეგაც დაასრულოს
ახალი შემომავალი წესის
ფორმირება „დასრულება“
(Finish) ღილაკის გამოყენებით.

თუ კომპიუტერი
მრავალქსელურია ეს
(შემომავალი) და შემდეგი
(გამავალი) წესების მოქმედება
უნდა განმეორდეს ყველა
ქსელისათვის.

Windows-ის ბრანდმაუერში SQL Server-ის პროგრამის გამონაკლისის პორტების დამატება



ანალოგიური მოქმედებები უნდა განხორციელდეს Windows Defender Firewall-ის (wf.msc) გამოყენებით „გამავალი წესები“-ს (Outbaund Rules) წარმოებისათვის, რისთვისაც მონიტორის ეკრანზე გამოსახულ დიალოგურ ფანჯრის მარცხენა მიდამოში მომხმარებელმა უნდა გააქტიუროს მეორე სტრიქონი „გამავალი წესები“-ს (Outbaund Rules), ხოლო შემდგომი დიალოგური ფანჯრები ზუსტად ანალოგიურები არიან.

Windows-ის ბრანდმაუერში SQL Server-ის ანალიზის სერვისების პორტები

SQL Server-ის ანალიზის სერვისების მიერ გამოყენებული პორტები

სტანდარტულად, SQL Server ანალიზის სერვისებისა და მასთან დაკავშირებული სერვისების მიერ გამოყენებული ტიპიური პორტებია: TCP 2382, 2383, 80 და 443. განვიხილოთ ამ პორტების გამოყენება უფრო დეტალურად:

- ▶ ანალიზის სერვისები - TCP პორტი 2383 ნაგულისხმევი ეგზემპლიარისათვის;
- ▶ SQL Server ბრაუზერის სერვისი - მხოლოდ TCP პორტი 2382 ანალიზის სერვისების სახელობითი ეგზემპლიარისათვის;
- ▶ IIS/HTTP ოქმის გამოსაყენებლად ანალიზის სერვისების კონფიგურირებისას - TCP პორტი 80;
- ▶ IIS/HTTP ოქმის გამოყენებით ანალიზის სერვისების კონფიგურირებისას - TCP პორტი 443.

თუ მომხმარებლები მიმართავენ ანალიზის სერვისებს IIS-ისა და ინტერნეტის გამოყენებით, უნდა გაიხსნას პორტი, რომელითაც IIS ელოდება მონაცემების მიგრაციას, რის შემდეგაც პორტი უნდა იქნეს მითითებული კლიენტის მიერთების სტრიქონში. ამ შემთხვევაში არ არის აუცილებელი ვიქონიოთ გახსნილი პორტები ანალიზის სერვისებზე პირდაპირი წვდომისათვის. ნაგულისხმევი პორტები 2389, 2382 და სხვა პორტები, რომლებთანაც წვდომა არ არის საჭირო.

Windows-ის ბრანდმაუერში SQL Server-ის ანგარიშებისა და ინტეგრაციის სერვისების პორტები

SQL Server-ის ანგარიშების სერვისების მიერ გამოყენებული პორტები

სტანდარტულად, SQL Server-ის ანგარიშების სერვისებისა და მასთან დაკავშირებული სერვისების მიერ გამოყენებული ტიპიური პორტებია: TCP 80 და 443. განვიხილოთ ამ პორტების გამოყენება უფრო დეტალურად:

- ▶ ანგარიშგების სერვისები ვებ სერვისები - TCP პორტი 80;
- ▶ HTTP ოქმის გამოსაყენებლად ანგარიშების სერვისების კონფიგურირებისას - TCP პორტი.

თუ ანგარიშგების სერვისები უკავშირდებიან მონაცემთა ბაზის ძრავის ან ანალიზის სერვისების ეგზემპლიარს, მომხმარებელმა ასევე უნდა გახსნას შესაბამისი პორტები ამ სერ

SQL Server-ის ინტეგრაციის სერვისების მიერ გამოყენებული პორტები

სტანდარტულად, SQL Server-ის ინტეგრაციის სერვისებისა და Microsoft-ის დაშორებული პროცედურის გამოძახებების (MS RPC) მიერ გამოყენებული ტიპიური პორტი არის TCP 135.

Windows-ის ბრანდმაუერში SQL Server-ის სხვა სერვისები და პორტები

SQL Server-ის მიერ გამოყენებული სხვა სერვისები და პორტები

- ▶ **Windows** მართვის ინსტრუმენტები (WMI) - მუშაობს როგორც საერთო სერვისის კვანძის ნაწილი DCOM-ით მინიჭებული პორტებით, რომელიც შეიძლება იყენებდეს TCP პორტს 135;
- ▶ **Microsoft**-ის განაწილებული ტრანზაქციების კოორდინატორი (MS DTC) - TCP პორტი 135;
- ▶ **SQL Server**-ის მენეჯმენტის სტუდიას დათვალიერების ღილაკი იყენებს UDP ოქმს SQL Server-ის ბრაუზერის სერვისთან დასაკავშირებლად - UDP პორტი 1434;
- ▶ **Ipsec**-ის ოქმის ტრაფიკი - UDP პორტი 500 და 4500;
- ▶ **Windows**-ის ავთენტიფიკაცია სანდო დომენებით - ბრანდმაუერით უნდა იქნეს კონფიგურირებული;
- ▶ SQL სერვერი და Windows კლასტერირება - კლასტერირებისთვის საჭიროა დამატებითი პორტები, რომლებიც უშუალოდ არ არიან დაკავშირებულნი SQL Server-თან;
- ▶ **URL** სახელების სივრცე, რომლებიც დარეზერვირებულია **HTTP** სერვერის API- ში (**HTTP.SYS**) - ზოგადად TCP პორტი 80, მაგრამ შესაძლებელია სხვა პორტებზე კონფიგურირებაც.

Windows-ის ბრანდმაუერში SQL Server-ის სხვა სერვისები და პორტები

სპეციალური ნიუანსები პორტ 135-ისთვის

როდესაც მომხმარებელი იყენებს სატრანსპორტო RPC-ს TCP/IP ან UDP/IP ოქმით, სისტემის სერვისებს შემომავალი პორტები საჭიროებისამებრ ენიჭებათ დინამიურად. ამ შემთხვევაში გამოიყენებიან TCP/IP და UDP/IP პორტები, რომლებიც აღემატებიან 1024-ს. ამ პორტებს უწოდებენ „შემთხვევით RPC პორტებს“ და RPC კლიენტები განსაზღვრავს დინამიურ პორტს, რომელიც ენიშნება სერვერს, RPC საბოლოო წერტილის მოდულით. RPC-ით მომუშავე ზოგიერთ სერვისისთვის შესაძლებელია კონფიგურირებულ იქნეს კონკრეტული პორტი, ასევე შესაძლებელია შეზღუდულ იქნეს პორტების დიაპაზონი, რომელსაც RPC-ი დინამიურად ანიჭებს და არ არის დამოკიდებული სერვისებზე. რადგან პორტი 135 გამოიყენება მრავალი სერვისისთვის, მას ხშირად ესხმიან ბოროტმოქმედებლები და ამიტომ პორტ 135-ის გახსნისას რეკომენდებულია ბრანდმაუერის მოქმედების წესების შეზღუდვა.

Windows-ის ბრანდმაუერში SQL Server-ის სხვა სერვისები და პორტები

ბრანდმაუერის სხვა წესებთან ურთიერთობა

Windows-ის ბრანდმაუერი კონფიგურირება ხორციელდება წესებისა და წესების ჯგუფების გათვალისწინებით. თითოეული წესი ან წესის ჯგუფი ასოცირდება კონკრეტულ პროგრამასთან ან სერვისთან, რომელსაც შეუძლია შეცვალოს ან წაშალოს დადგენილი წესი გაფრთხილების გარეშე. მაგალითად, წესების ჯგუფები World Wide Web Services (HTTP) და World Wide Web Services (HTTPS) ასოცირებულია IIS-თან და ამ წესების ჩართვა ავტომატურად იწვევს 80 და 443 პორტების გახსნას და შესაბამისად იმ SQL Server-ის ფუნქციების, რომლებიც დამოკიდებულნი არიან ამ პორტებზე. თუმცა ადმინისტრატორს აქვს შესაძლებლობა შეცვალოს ან გამორთოს ეს წესები IIS-ის კონფიგურირებისას.

Windows -ის გაფართოებული უსაფრთხოების ბრანდმაუერი MMC გაატარებს სრულ ტრაფიკს, რომელიც შეესაბამება ნებადართულ ნებისმიერ მოქმედ წესს. ანუ, თუ არსებობს ორი წესი განსხვავებული პარამეტრებით, რომლებიც ვრცელდებიან 80 პორტზე, ბრანდმაუერი გაატარებს ნებისმიერს, თუ იგი შეესაბამება დაშვების ერთ წესს მაინც. მაგალითად თუ ერთი წესი ხსნის ტრაფიკს 80 პორტზე ადგილობრივ ქსელისათვის, ხოლო მეორე ამავე პორტით ხსნის ტრაფიკს ნებისმიერი მისამართისათვის, შედეგად 80 პორტი იქნება გახსნილი ნებისმიერი მისამართისათვის. ამიტომ ადმინისტრატორმა ეფექტურად მართვისთვის მუდმივად უნდა ამოწმოს SQL Server-ზე წვდომის ბრანდმაუერის წესები.

Windows-ის ბრანდმაუერის პროფილები

Windows-ი ბრანდმაუერის პროფილის მიხედვით განსაზღვრავს და იმახსოვრებს თითოეულ ქსელს შემდეგი პარამეტრების მიხედვით: მიერთების შესაძლებლობა, არსებული მიერთებები და კატეგორია.

Windows-ი გაფართოებული უსაფრთხოების ბრანდმაუერის ქსელს ყოფს სამ პროფილად:

- ▶ **დომენი** - Windows-ს აქვს შესაძლებლობა დაადასტუროს დომენის კონტროლერზე წვდომა იმ დომენისთვის, რომელშიც კომპიუტერი არის ჩართული;
- ▶ **საჯარო** - თავდაპირველად ყველა ქსელი გარდა დომენური ქსელებისა, პროფილდება როგორც საჯარო. ასეთი საჯარო ქსელები წარმოადგენენ პირდაპირ კავშირებს ინტერნეტთან და არიან საჯარო ადგილებში. მაგალითად, აეროპორტები, რესტორნები და სხვა;
- ▶ **კერძო** - მომხმარებლის ან პროგრამის მიერ იდენტიფიცირებული ქსელი, როგორც კერძო. მხოლოდ სანდო ქსელი უნდა იქნეს ასე იდენტიფიცირებული. მაგალითად, მცირე დაწესებულების ან სახლის პირობებში.

ადმინისტრატორს აქვს შესაძლებლობა შექმნას ქსელის თითოეული პროფილისათვის ბრანდმაუერის სხვადასხვა პოლიტიკა. ერთდროულად შესაძლებელია მხოლოდ ერთი პროფილის გამოიყენება.

(გაგრძელება შემდეგ სლაიდზე)

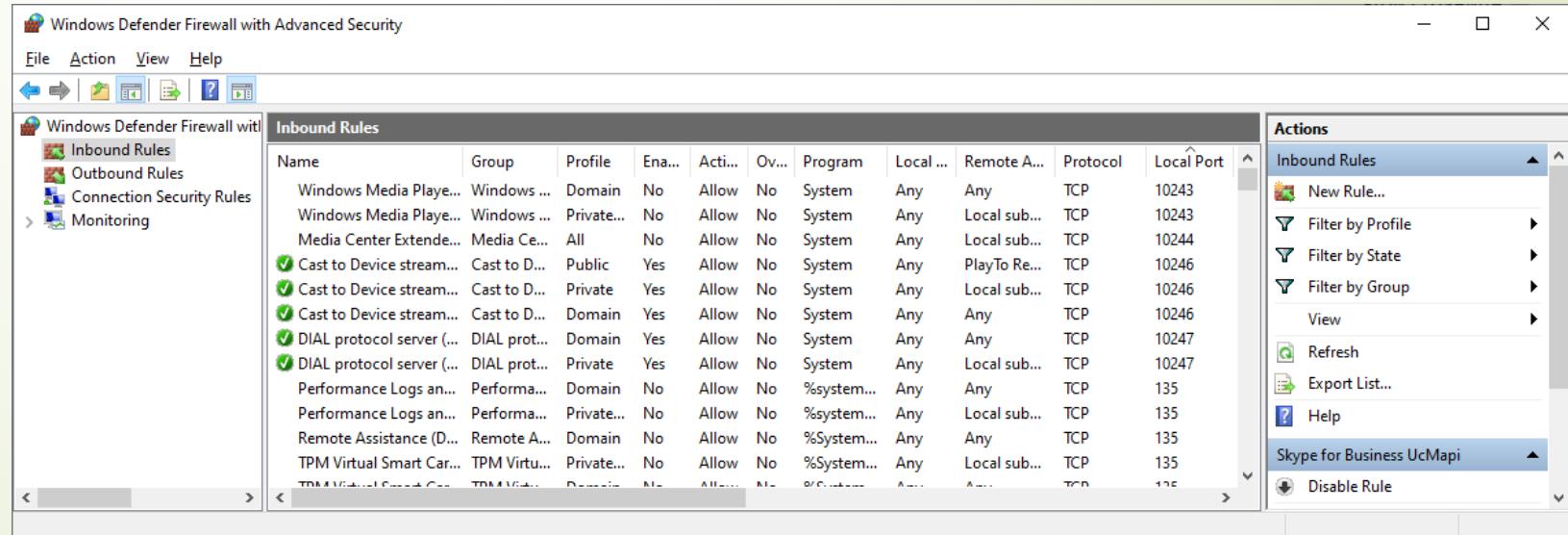
Windows-ის ბრანდმაუერის პროფილები

პროფილები გამოიყენებიან შემდეგი მიმდევრობით:

1. დომენის პროფილი გამოიყენება თუ ყველა ინტერფეისმა გაიარა ავთენტიფიკაცია დომენის კონტროლერზე, რომლის წევრის არის კომპიუტერი;
2. თუ ყველა ინტერფეისმა ან გაიარა ავთენტიფიკაცია დომენის კონტროლერზე, ან მიერთებულია კერძო ქსელთან, მაშინ გამოიყენება კერძო პროფილი;
3. წინააღმდეგ შემთხვევაში, გამოიყენება საჯარო პროფილი.

Windows-ის ბრანდმაუერის პრობლემები

პორტის მდგომარეობის ეფექტური მართვა არის პორტთან დაკავშირებული ყველა წესის გაერთიანებული მართვა, ამიტომ შეიძლება სასარგებლო იყოს პორტის ნომრის მითითების ყველა წესის გადახედვა. Windows Defender Firewall-ის Windows-ში გაშვების wf.msc-ს შემდეგ (პროცესი აღწერილია წინა სლაიდებში) მომხმარებელმა უნდა გაააქტიუროს შემავალი/გამომავალი წესები და ფანჯრის შუა მიდამოში დაალაგოს სია პორტის ნომრის მიხედვით (Local port), რისთვისაც მაუსით უნდა გაააქტიუროს სვეტის სათაური. შემოწმება მდგომარეობს ერთიდაიგივე პორტისათვის თუ ორი ან მეტი სტრიქონი არსებობს, თანხვედრაში არის თუ არა ყველა წესი ერთმანეთთან და კომპიუტერის წვდომის წესებთან. იმის დასადასტურებლად, თუ რომელი პორტები აპირებენ მონაცემთა გადაცემას, შესაძლებელია ინახოს აქტიური TCP კავშირები და IP სტატისტიკა **netstat** პროგრამის გამოიყენებით.



საინფორმაციო სისტემები

22

საინფორმაციო სისტემა არის სტრუქტურირებული (მოწესრიგებული) მონაცემებისა და აპარატურულ-პროგრამული საშუალებების ერთობლიობა, განკუთვნილი მონაცემების შენახვისა და დამუშავებისათვის.

არსებობს საინფორმაციო სისტემების ორი კლასი:

- ▶ საინფორმაციო-საძებნი;
- ▶ მონაცემების დამუშავების.

საინფორმაციო-საძებნი საინფორმაციო სისტემა ორიენტირებულია საჭირო მონაცემების ძებნასა და მიღებაზე, ხოლო მონაცემების დამუშავების საინფორმაციო სისტემა - მონაცემების დამუშავებასა და დამუშავებული ინფორმაციის მიღებაზე. საინფორმაციო სისტემები ასრულებენ შემდეგ ფუნქციებს:

- ▶ ინფორმაციის შეტანა, ცვლილება და შენახვა;
- ▶ ინფორმაციის ნახვა და ძებნა;
- ▶ ინფორმაციის ამოღება სათანადო კრიტერიუმის მიხედვით;
- ▶ ანგარიშების ფორმირება;
- ▶ ინფორმაციის სისწორის შემოწმება.

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

SQL Server-ის ავთენტიფიკაციის მართვა ხორციელდება სერვერის როლებში დამატებით. სერვერის როლი წარმოდგენილია ობიექტით ServerRole. მონაცემთა ბაზის როლი წარმოდგენილია ობიექტით DatabaseRole, ხოლო აპლიკაციის როლი წარმოდგენილია ობიექტით ApplicationRole.

სერვერის დონის პრივილეგიები ჩამოთვლილია ServerPermission ობიექტის თვისებებში. სერვერის დონის პრივილეგიები შეიძლება მიენიჭოს, უარი ითქვას ან გაუქმდეს კონკრეტული ავთენტიფიკაციისათვის.

მონაცემთა ბაზის ყველა ობიექტს გააჩნია ობიექტი UserCollection, რომელიც განსაზღვრავს მონაცემთა ბაზის ყველა მომხმარებელს. SQL Server-ის თითოეული მომხმარებელი ასოცირდება კონკრეტულ ავთენტიფიკაციასთან, მაგრამ ერთი ავთენტიფიკაცია შეიძლება ასოცირებული იყოს ერთზე მეტ მომხმარებლებთან.

SQL Server მონაცემთა ბაზებს ასევე გააჩნიათ როლები, რომლებიც განსაზღვრავს მონაცემთა ბაზის დონის პრივილეგიებს, რაც მომხმარებელს აძლევს საშუალებას შეასრულოს კონკრეტული ამოცანები. სერვერის როლებისგან განსხვავებით, მონაცემთა ბაზის როლები არ არიან დაფიქსირებულნი და, ამიტომ, შესაძლებელია მათი შექმნა, შეცვლა და წაშლა.

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

მაგალითი 1:

მოვიყვანოთ მაგალითი შესვლისა და ასოცირებული მომხმარებლების აღრიცხვის Visual C#-ში. მაგალითი გვიჩვენებს, თუ როგორ უნდა განვახორციელოთ ავთენტიფიკაცია მონაცემთა ბაზის ყველა მომხმარებლის, რომლებიც მასთან ასოცირდებიან.

```
{
    Server srv = new Server();
    foreach ( Database db in srv.Databases) {
        Console.WriteLine("=====");
        Console.WriteLine("Login Mappings for the database: " + db.Name);
        Console.WriteLine(" ");
        DataTable d;
        d = db.EnumLoginMappings();
        foreach (DataRow r in d.Rows) {
            foreach ( DataColumn c in r.Table.Columns) {
                Console.WriteLine(c.ColumnName + " = " + r[c]);
            }
            Console.WriteLine(" ");
        }
    }
}
```

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

მაგალითი 2:

მოვიყვანოთ მაგალითი შესვლისა და ასოცირებული მომხმარებლების აღრიცხვა PowerShell-ში. მაგალითი გვიჩვენებს, თუ როგორ უნდა განვახორციელოთ ავთენტიფიკაცია მონაცემთა ბაზის ყველა მომხმარებლის, რომლებიც მასთან ასოცირდებიან.

```
CD \sql\localhost\Default\Datasets
foreach ($db in Get-ChildItem)
{
    "====="
    "Login Mappings for the database: "+ $db.Name
    $dt = $db.EnumLoginMappings()
    foreach($row in $dt.Rows)
    {
        foreach($col in $row.Table.Columns)
        {
            $col.ColumnName + "=" + $row[$col]
        }
    }
}
```

სერვერის დონის როლები

26

SQL Server-ი უზრუნველყოფს სერვერის დონის როლებს, რომლებიც მართავენ სერვერის ნებართვებს. ეს როლები არიან უსაფრთხოების პრინციპები, რომლებიც ჯგუფდებიან სხვა პრინციპებთან. სერვერის დონის როლები ვრცელდებიან სრული სერვერის მასშტაბით მათი ნებართვების ფარგლებში (როლები მსგავსია Windows ოპერაციული სისტემის ჯგუფებსა).

SQL Server-ში სერვერის ფიქსირებული როლები შექმნილია მოხერხებულობისათვისა და უკუ თავსებადობისთვის.

SQL Server-ი უზრუნველყოფს ცხრა ფიქსირებული სერვერის როლს, რომლებიც გარდა საჯაროს (Public) არ შეიძლება იქნეს შეცვლილი. SQL Server 2012-დან (11.x) დაწყებული, მომხმარებელს ეძლევა შესაძლებლობა შექმნას მომხმარებლის მიერ განსაზღვრული სერვერის როლები და დაამატოს სერვერის დონის ნებართვები მათი გამოყენებით.

სერვერის დონის ნებართვები მიუწვდომელია SQL Server-ის მონაცემთა ბაზაში ან Azure Synapse Analytics-ში.

სერვერის დონის როლები

27

განვიხილოთ SQL Server-ში ფიქსირებული სერვერის დონის როლები და მათი შესაძლებლობები:

sysadmin - ნებისმიერი აქტივობა სერვერზე;

serveradmin - ცვლის სერვერის დონის კონფიგურირების პარამეტრებს და შეუძლია SQL Server-ის გამორთვა;

securityadmin - ავთენტიფიკაციებისა და მათი თვისებების მართვა GRANT, DENY და REVOKE-ის გამოყენებით როგორც სერვერის დონეზე, ასევე მონაცემთა ბაზის დონეზე (თუ მათ აქვთ წვდომა ბაზაში). აგრეთვე, მათ შეუძლიათ ავთენტიფიკის პაროლების ჩამოყრა SQL Server-ში;

processadmin - SQL Server-ის ეგზემპლიარზე პროცესების დასრულება;

setupadmin - Transact-SQL-ის ინსტრუქციების გამოყენებით ბმული სერვერების დამატება და წაშლა (Management Studio გარემოში აუცილებელია sysadmin-ის წევრობა);

bulkadmin - BULK INSERT ინსტრუქციის წარმოება (bulkadmin როლი არ არის მხარდაჭერილი SQL Server-თვის Linux-ში, რომელშიც BULK INSERT ინსტრუქციის წარმოება შეუძლია მხოლოდ sysadmin-ს);

(გაგრძელება შემდეგ სლაიდზე)

სერვერის დონის როლები

28

diskadmin - მართავს SQL Server-ის ფაილებს დისკზე;

dbcreator - SQL Server-ში ნებისმიერი მონაცემთა ბაზის შექმნა, შეცვლა, წაშლა და აღდგენა;

public - ყველა SQL Server-ის ავთენტიფიკაცია ეკუთვნის public სერვერის ფიქსირებულ როლს. მომხმარებელს არ გააჩნია შესაძლებლობა შეცვალოს მასში წევრობა.

SQL Server-ის სერვერის დონის ნებართვების ნახვა განხილულია შემდეგ მაგალითში:

SELECT *

FROM sys.fn_builtin_permissions('SERVER')

ORDER BY permission_name

სერვერის დონის როლები

29

ჩამოვთვალოთ SQL Server-ის სერვერის დონის როლებთან მუშაობის ბრძანებებს,
წარმოდგენებს და ფუნქციებს:

sp_helpsrvrole - აბრუნებს სერვერის დონის როლების სიას;

sp_helpsrvrolemember - აბრუნებს ინფორმაციას სერვერის დონის წევრების შესახებ;

sp_srvrolepermission - აჩვენებს სერვერის დონის როლის ნებართვებს;

is_srvrolemember - მიუთითებს არის თუ არა SQL Server-ის ავთენტიფიკაცია
მითითებული სერვერის დონის როლის წევრი;

sys.server_role_members - აბრუნებს ერთ რიგს თითოეული სერვერის დონის როლის
თითოეული წევრისთვის;

create server role - ბრძანება ქმნის მომხმარებლის მიერ განსაზღვრულ სერვერის როლს;

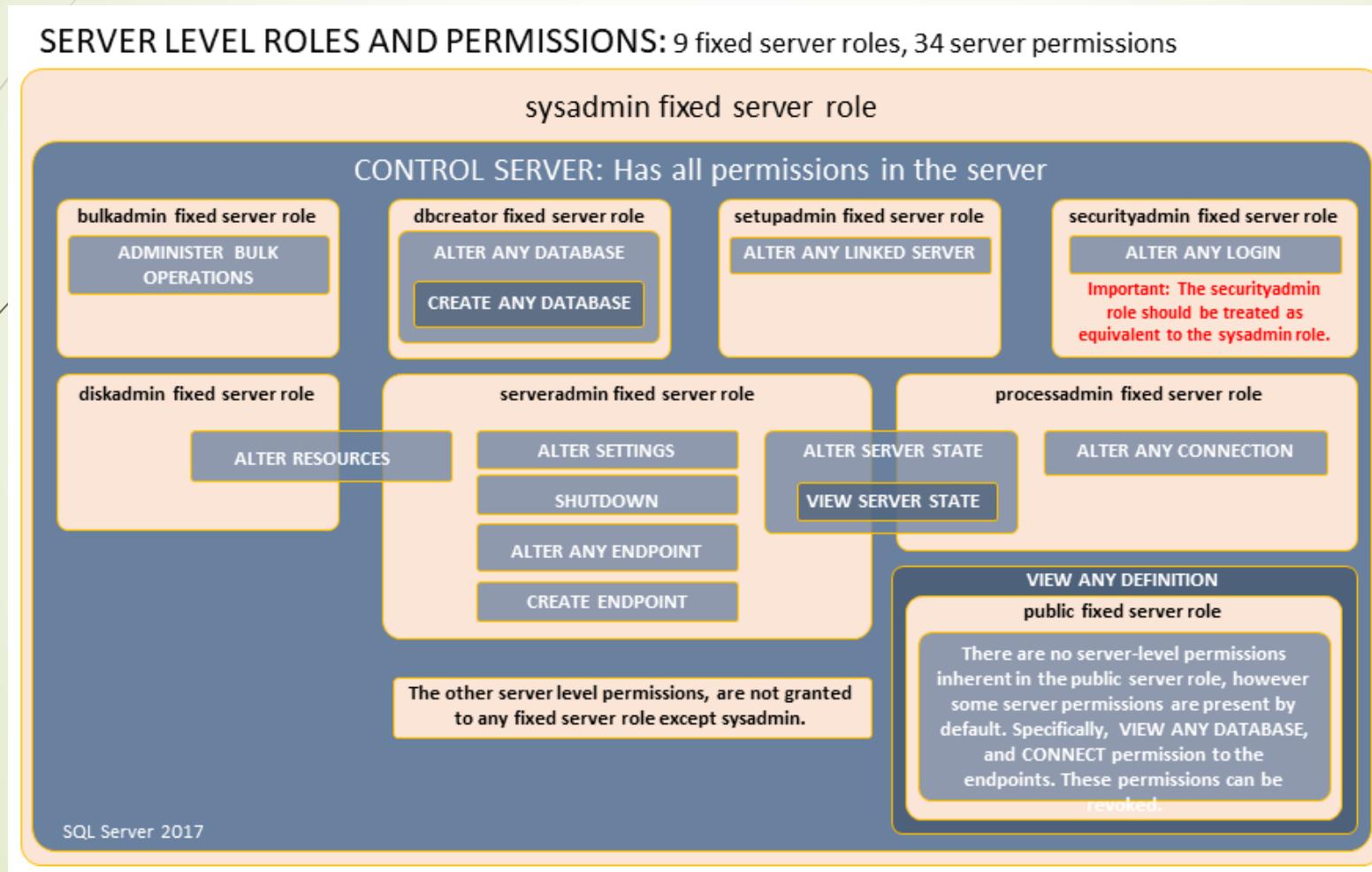
alter server role - ბრძანება ცვლის სერვერის როლის წევრობას ან მომხმარებლის მიერ
განსაზღვრული სერვერის როლის სახელს;

drop server role - ბრძანება შლის მომხმარებლის მიერ განსაზღვრულ სერვერის როლს.

სერვერის დონის როლები

30

SQL Server-ში თითოეულ ფიქსირებულ სერვერის როლს გააჩნია მასზე მინიჭებული გარკვეული ნებართვები, რომელიც გამოსახულია შემდეგ ნახაზზე:



მონაცემთა ბაზის დონის როლები

31

SQL Server-ი უზრუნველყოფს მონაცემთა ბაზების დონის როლებს, რომლებიც მართავენ მონაცემთა ბაზების ნებართვებს. ეს როლები არიან უსაფრთხოების პრინციპები, რომლებიც ჯგუფდებიან სხვა პრინციპებთან. მონაცემთა ბაზების დონის როლები ვრცელდებიან სრულად მონაცემთა ბაზაში მათი ნებართვების ფარგლებში (როლები მსგავსია Windows ოპერაციული სისტემის ჯგუფებსა).

SQL Server-ი უზრუნველყოფს მონაცემთა ბაზის როლში მომხმარებლების დასამატებლად და წასაშლელად გამოიყენეთ ALTER ROLE განცხადების ADD MEMBER და DROP MEMBER ოფციები. მონაცემთა ბაზების დონის ნებართვები მიუწვდომელია SQL Server-ის ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში.

არსებობს ორი ტიპის მონაცემთა ბაზის დონის როლები: ფიქსირებული მონაცემთა ბაზის როლები, რომლებიც წინასწარ არის განსაზღვრული მონაცემთა ბაზაში და მომხმარებლის მიერ განსაზღვრული მონაცემთა ბაზის როლები, რომლებიც შეგიძლიათ შექმნათ.

SQL Server-ში ფიქსირებული მონაცემთა ბაზის როლები განისაზღვრება მონაცემთა ბაზის დონეზე და არსებობს თითოეულ მონაცემთა ბაზაში. მონაცემთა ბაზის დონის db_owner როლს შეუძლია მართოს ფიქსირებული ბაზის როლის წევრობა. ასევე არსებობს სპეციალური დანიშნულების მონაცემთა ბაზის როლები msdb მონაცემთა ბაზაში.

მონაცემთა ბაზის დონის როლები

32

SQL Server-ის მონაცემთა ბაზის დონის როლებში არის შესაძლებლობა დაემატოს მონაცემთა ბაზის ნებისმიერი სააღრიცხვო ჩანაწერი და სხვა SQL Server-ის როლები.

არ არის რეკომენდირებული დაემატოს მომხმარებლის მონაცემთა ბაზის როლები ფიქსირებულ როლების წევრად. ამან შეიძლება გამოიწვიოს გაუთვალისწინებელი პრივილეგიები.

SQL Server-ის მომხმარებლის მონაცემთა ბაზის როლების ნებართვების მართვა შესაძლებელია GRANT, DENY და REVOKE ინსტრუქციების გამოყენებით.

SQL Server-ის სერვერის დონის ნებართვები ან სუბიექტები არ შეიძლება მიენიჭოს ან დაემატოს მონაცემთა ბაზის როლებს.

მონაცემთა ბაზის დონის როლები

33

განვიხილოთ SQL Server-ში მონაცემთა ბაზების დონის ფიქსირებული როლები და მათი შესაძლებლობები:

db_owner - შეუძლია შეასრულოს ყველა მოქმედება მონაცემთა ბაზის კონფიგურაციის, მომსახურებისა და წაშლის. (ზოგიერთ ტექნიკურ აქტივობას ესაჭიროება სერვერის დონის ნებართვები);

db_securityadmin - შეუძლია შეცვალოს როლში წევრობა და უფლებების მართვა (ამ როლის წევრებს შეუძლიათ პოტენციურად აიმაღლონ თავიანთი პრივილეგიები და მათი ქმედებები უნდა მონიტორინგდებოდეს);

db_accessadmin - შეუძლია დაამატოს ან წაშალოს მონაცემთა ბაზაზე დაშორებული წვდომის უფლებები Windows ავთენტიფიკაციებისათვის და ჯგუფებისათვის და, ასევე SQL Server-ის ავთენტიფიკაციებისათვის;

db_backupoperator - შეუძლია მონაცემთა ბაზის სარეზერვო ასლის შექმნა;

db_ddladmin - შეუძლია აწარმოოს მონაცემთა ბაზაში მონაცემთა განსაზღვრის ენის (DDL) ნებისმიერი ბრძანება;

db_datawriter - შეუძლია მომხმარებლის ყველა ცხრილში მონაცემების ფორმატირება (დამატება, წაშლა ან შეცვლა);

მონაცემთა ბაზის დონის როლები

34

db_datareader - შეუძლია წაიკითხოს ყველა ცხრილიდან და წარმოდგენიდან მონაცემები;

db_denydatawriter - არ შეუძლია მონაცემთა ბაზაში მომხმარებლის ცხრილებში მონაცემების ფორმატირება (დამატება, შეცვლა ან წაშლა);

db_denydatareader - არ შეუძლია მონაცემთა ბაზაში მონაცემთა წაკითხვა მომხმარებლის ცხრილებიდან და წარმოდგენებიდან.

მონაცემთა ბაზის დონის როლები

35

ჩამოვთვალოთ SQL Server-ის მონაცემთა ბაზის დონის როლებთან მუშაობის ბრძანებებს, წარმოდგენებს და ფუნქციებს:

sp_helpdbfixedrole - აბრუნებს ფიქსირებული მონაცემთა ბაზის როლების სიას;

sp_dbfixedrolepermission - აჩვენებს ფიქსირებული მონაცემთა ბაზის როლის ნებართვებს;

sp_helprole - აბრუნებს ინფორმაციას როლების შესახებ მიმდინარე მონაცემთა ბაზაში;

sp_helprolemember - აბრუნებს ინფორმაციას როლის წევრების შესახებ მიმდინარე მონაცემთა ბაზაში;

sys.database_role_members - აბრუნებს ერთ სტრიქონს მონაცემთა ბაზის თითოეული როლის თითოეული წევრისთვის;

is_member - მიუთითებს, არის თუ არა მიმდინარე მომხმარებელი მითითებული Windows ჯგუფის წევრი ან SQL Server-ის მონაცემთა ბაზის როლის წევრი;

create role - ბრძანება ქმნის მონაცემთა ბაზის ახალ როლს მიმდინარე მონაცემთა ბაზაში;

alter role - ბრძანება ცვლის მონაცემთა ბაზის როლის სახელს ან წევრობას;

drop role - ბრძანება წაშლის როლს მონაცემთა ბაზიდან;

მონაცემთა ბაზის დონის როლები

36

sp_addrole - ბრძანება ქმნის მონაცემთა ბაზის ახალ როლს მიმდინარე მონაცემთა ბაზაში;

sp_droprole - ბრძანება შლის მონაცემთა ბაზის როლს მიმდინარე მონაცემთა ბაზიდან;

sp_addrolemember - ბრძანება ამატებს მიმდინარე მონაცემთა ბაზაში მონაცემთა ბაზის მომხმარებელს, მონაცემთა ბაზის როლს, Windows ავთენტიფიკაციას ან Windows ჯგუფს. ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში ამის მაგივრად გამოიყენება alter role;

sp_droprolemember - ბრძანება შლის მიმდინარე მონაცემთა ბაზაში უსაფრთხოების ჩანაწერს SQL სერვერის როლიდან. ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში ამის მაგივრად გამოიყენება alter role;

grant - ამატებს როლისათვის ნებართვას;

deny - კრძალავს როლისათვის ნებართვას;

revoke - შლის ადრე მინიჭებულ ან უარყოფილ ნებართვებს.

მაგალითი 1: მონაცემთა ბაზის დონის ფიქსირებულ როლს db_datawriter დავამატოთ მომხმარებლის 'გია'.

**ALTER ROLE db_datawriter
ADD MEMBER 'გია'**

მონაცემთა ბაზის დონის როლები

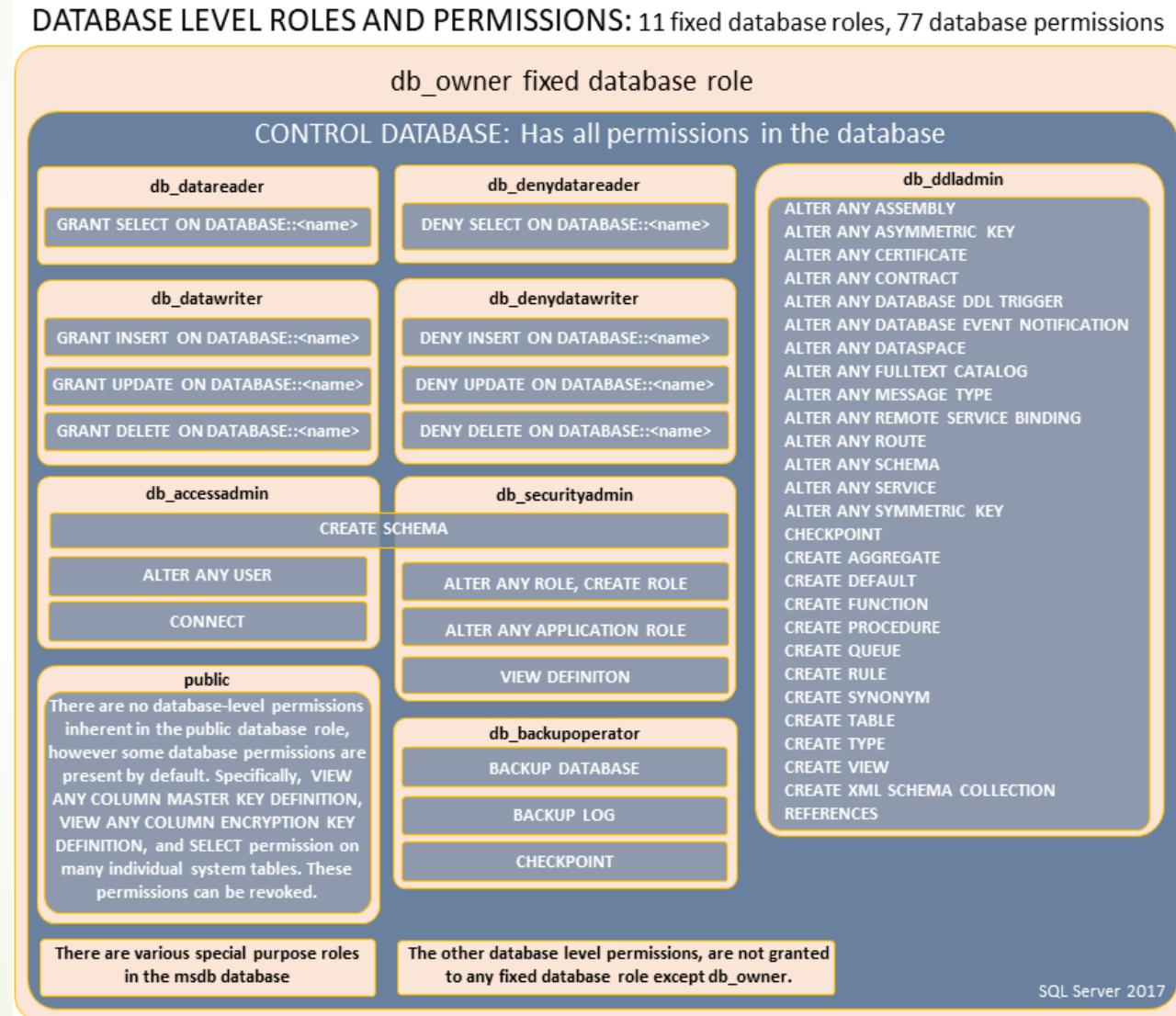
37

მაგალითი 2: დავაბრუნოთ მონაცემთა ბაზის დონის ნებისმიერი როლის ყველა წევრი:

```
SELECT    roles.principal_id  
        , roles.name  
        , database_role_members.member_principal_id  
        , members.name  
FROM      sys.database_role_members AS database_role_members  
JOIN      sys.database_principals AS roles  
ON        database_role_members.role_principal_id = roles.principal_id  
JOIN      sys.database_principals AS members  
ON        database_role_members.member_principal_id = members.principal_id
```

მონაცემთა ბაზის დონის როლები

SQL Server-ის მონაცემთა ბაზის ფიქსირებული როლის შეცვლა არ შეიძლება და მას გააჩნია მასზე მინიჭებული გარკვეული ნებართვები, რომლებიც გამოსახულნი არიან შემდეგ ნახაზზე:



აპლიკაციის როლები

39

აპლიკაციის როლი არის მონაცემთა ბაზის პრინციპი, რომელიც საშუალებას აძლევს აპლიკაციას შეასრულოს საკუთარი (მომხმარებლის მსგავსი) წვდომის ნებართვები. მომხმარებელს აქვს შესაძლებლობა გამოიყენოს აპლიკაციის როლები, რათა ჩართოს წვდომა კონკრეტულ მონაცემებზე მხოლოდ იმ მომხმარებლებისთვის, რომლებიც კავშირდებიან SQL Server-ს კონკრეტული აპლიკაციის გამოყენებით. მონაცემთა ბაზის როლებისგან განსხვავებით, აპლიკაციის როლები არ შეიცავს წევრებს და ნაგულისხმევად არიან არააქტიურნი. აპლიკაციის როლები აქტიურდებიან sp_setapprole სისტემური პროცედურის გამოყენებით, რომელიც საჭიროებს პაროლის მითითებას. რადგან აპლიკაციის როლები არის მონაცემთა ბაზის დონის პრინციპები, მათ შეუძლიათ სხვა მონაცემთა ბაზებზე წვდომა მხოლოდ ამ მონაცემთა ბაზებში სტუმრისთვის (guest) სააღრიცხვო ჩანაწერის ნებართვებით. ამიტომ, ნებისმიერი მონაცემთა ბაზა, რომელშიც guest გამორთულია, მიუწვდომელი იქნება აპლიკაციის როლებისთვის.

აპლიკაციის როლთან დაკავშირების პროცესი გადის შემდეგ ნაბიჯებს, რომლითაც აპლიკაციის როლი ცვლის უსაფრთხოების კონტექსტს: მომხმარებელი აღასრულებს კლიენტის აპლიკაციას; კლიენტის აპლიკაცია უკავშირდება ერთდება SQL Server-ის ეგზემპლიარს, როგორც მომხმარებელი; აპლიკაცია აღასრულებს sp_setapprole შენახულ პროცედურას მხოლოდ აპლიკაციისთვის ცნობილი პაროლით; თუ აპლიკაციის როლის სახელი და პაროლი სწორია, ჩაირთვება აპლიკაციის როლი; ამ დროს კავშირი კარგავს მომხმარებლის ნებართვებს და იღებს აპლიკაციის როლის ნებართვებს, რომლებიც ძალაში რჩებიან კავშირის ხანგრძლივობის პერიოდში.

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

SQL Server-ის ავთენტიფიკაციის მართვა ხორციელდება სერვერის როლებში დამატებით. სერვერის როლი წარმოდგენილია ობიექტით ServerRole. მონაცემთა ბაზის როლი წარმოდგენილია ობიექტით DatabaseRole, ხოლო აპლიკაციის როლი წარმოდგენილია ობიექტით ApplicationRole.

სერვერის დონის პრივილეგიები ჩამოთვლილია ServerPermission ობიექტის თვისებებში. სერვერის დონის პრივილეგიები შეიძლება მიენიჭოს, უარი ითქვას ან გაუქმდეს კონკრეტული ავთენტიფიკაციისათვის.

მონაცემთა ბაზის ყველა ობიექტს გააჩნია ობიექტი UserCollection, რომელიც განსაზღვრავს მონაცემთა ბაზის ყველა მომხმარებელს. SQL Server-ის თითოეული მომხმარებელი ასოცირდება კონკრეტულ ავთენტიფიკაციასთან, მაგრამ ერთი ავთენტიფიკაცია შეიძლება ასოცირებული იყოს ერთზე მეტ მომხმარებლებთან.

SQL Server მონაცემთა ბაზებს ასევე გააჩნიათ როლები, რომლებიც განსაზღვრავს მონაცემთა ბაზის დონის პრივილეგიებს, რაც მომხმარებელს აძლევს საშუალებას შეასრულოს კონკრეტული ამოცანები. სერვერის როლებისგან განსხვავებით, მონაცემთა ბაზის როლები არ არიან დაფიქსირებულნი და, ამიტომ, შესაძლებელია მათი შექმნა, შეცვლა და წაშლა.

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

მაგალითი 1:

მოვიყვანოთ მაგალითი შესვლისა და ასოცირებული მომხმარებლების აღრიცხვის Visual C#-ში. მაგალითი გვიჩვენებს, თუ როგორ უნდა განვახორციელოთ ავთენტიფიკაცია მონაცემთა ბაზის ყველა მომხმარებლის, რომლებიც მასთან ასოცირდებიან.

```
{
Server srv = new Server();
foreach ( Database db in srv.Databases) {
    Console.WriteLine("=====");
    Console.WriteLine("Login Mappings for the database: " + db.Name);
    Console.WriteLine(" ");
    DataTable d;
    d = db.EnumLoginMappings();
    foreach (DataRow r in d.Rows) {
        foreach (DataColumn c in r.Table.Columns) {
            Console.WriteLine(c.ColumnName + " = " + r[c]);
        }
        Console.WriteLine(" ");
    }
}
}
```

მომხმარებლების, როლებისა და ავთენტიფიკაციების მართვა

მაგალითი 2:

მოვიყვანოთ მაგალითი შესვლისა და ასოცირებული მომხმარებლების აღრიცხვა PowerShell-ში. მაგალითი გვიჩვენებს, თუ როგორ უნდა განვახორციელოთ ავთენტიფიკაცია მონაცემთა ბაზის ყველა მომხმარებლის, რომლებიც მასთან ასოცირდებიან.

```
CD \sql\localhost\Default\Datasets
foreach ($db in Get-ChildItem)
{
    "====="
    "Login Mappings for the database: "+ $db.Name
    $dt = $db.EnumLoginMappings()
    foreach($row in $dt.Rows)
    {
        foreach($col in $row.Table.Columns)
        {
            $col.ColumnName + "=" + $row[$col]
        }
    }
}
```

სერვერის დონის როლები

4

SQL Server-ი უზრუნველყოფს სერვერის დონის როლებს, რომლებიც მართავენ სერვერის ნებართვებს. ეს როლები არიან უსაფრთხოების პრინციპები, რომლებიც ჯგუფდებიან სხვა პრინციპებთან. სერვერის დონის როლები ვრცელდებიან სრული სერვერის მასშტაბით მათი ნებართვების ფარგლებში (როლები მსგავსია Windows ოპერაციული სისტემის ჯგუფებსა).

SQL Server-ში სერვერის ფიქსირებული როლები შექმნილია მოხერხებულობისათვისა და უკუ თავსებადობისთვის.

SQL Server-ი უზრუნველყოფს ცხრა ფიქსირებული სერვერის როლს, რომლებიც გარდა საჯაროს (Public) არ შეიძლება იქნეს შეცვლილი. SQL Server 2012-დან (11.x) დაწყებული, მომხმარებელს ეძლევა შესაძლებლობა შექმნას მომხმარებლის მიერ განსაზღვრული სერვერის როლები და დაამატოს სერვერის დონის ნებართვები მათი გამოყენებით.

სერვერის დონის ნებართვები მიუწვდომელია SQL Server-ის მონაცემთა ბაზაში ან Azure Synapse Analytics-ში.

სერვერის დონის როლები

5

განვიხილოთ SQL Server-ში ფიქსირებული სერვერის დონის როლები და მათი შესაძლებლობები:

sysadmin - ნებისმიერი აქტივობა სერვერზე;

serveradmin - ცვლის სერვერის დონის კონფიგურირების პარამეტრებს და შეუძლია SQL Server-ის გამორთვა;

securityadmin - ავთენტიფიკაციებისა და მათი თვისებების მართვა GRANT, DENY და REVOKE-ის გამოყენებით როგორც სერვერის დონეზე, ასევე მონაცემთა ბაზის დონეზე (თუ მათ აქვთ წვდომა ბაზაში). აგრეთვე, მათ შეუძლიათ ავთენტიფიკაციის პაროლების ჩამოყრა SQL Server-ში;

processadmin - SQL Server-ის ეგზემპლიარზე პროცესების დასრულება;

setupadmin - Transact-SQL-ის ინსტრუქციების გამოყენებით ბმული სერვერების დამატება და წაშლა (Management Studio გარემოში აუცილებელია sysadmin-ის წევრობა);

bulkadmin - BULK INSERT ინსტრუქციის წარმოება (bulkadmin როლი არ არის მხარდაჭერილი SQL Server-თვის Linux-ში, რომელშიც BULK INSERT ინსტრუქციის წარმოება შეუძლია მხოლოდ sysadmin-ს);

(გაგრძელება შემდეგ სლაიდზე)

სერვერის დონის როლები

6

diskadmin - მართავს SQL Server-ის ფაილებს დისკზე;

dbcreator - SQL Server-ში ნებისმიერი მონაცემთა ბაზის შექმნა, შეცვლა, წაშლა და აღდგენა;

public - ყველა SQL Server-ის ავთენტიფიკაცია ეკუთვნის public სერვერის ფიქსირებულ როლს. მომხმარებელს არ გააჩნია შესაძლებლობა შეცვალოს მასში წევრობა.

SQL Server-ის სერვერის დონის ნებართვების ნახვა განხილულია **შემდეგ მაგალითში**:

SELECT *

FROM sys.fn_builtin_permissions('SERVER')

ORDER BY permission_name

სერვერის დონის როლები

7

ჩამოვთვალოთ SQL Server-ის სერვერის დონის როლებთან მუშაობის ბრძანებებს, ნარმოდგენებს და ფუნქციებს:

sp_helpsrvrole - აბრუნებს სერვერის დონის როლების სიას;

sp_helpsrvrolemember - აბრუნებს ინფორმაციას სერვერის დონის წევრების შესახებ;

sp_srvrolepermission - აჩვენებს სერვერის დონის როლის ნებართვებს;

is_srvrolemember - მიუთითებს არის თუ არა SQL Server-ის ავთენტიფიკაცია მითითებული სერვერის დონის როლის წევრი;

sys.server_role_members - აბრუნებს ერთ რიგს თითოეული სერვერის დონის როლის თითოეული წევრისთვის;

create server role - ბრძანება ქმნის მომხმარებლის მიერ განსაზღვრულ სერვერის როლს;

alter server role - ბრძანება ცვლის სერვერის როლის წევრობას ან მომხმარებლის მიერ განსაზღვრული სერვერის როლის სახელს;

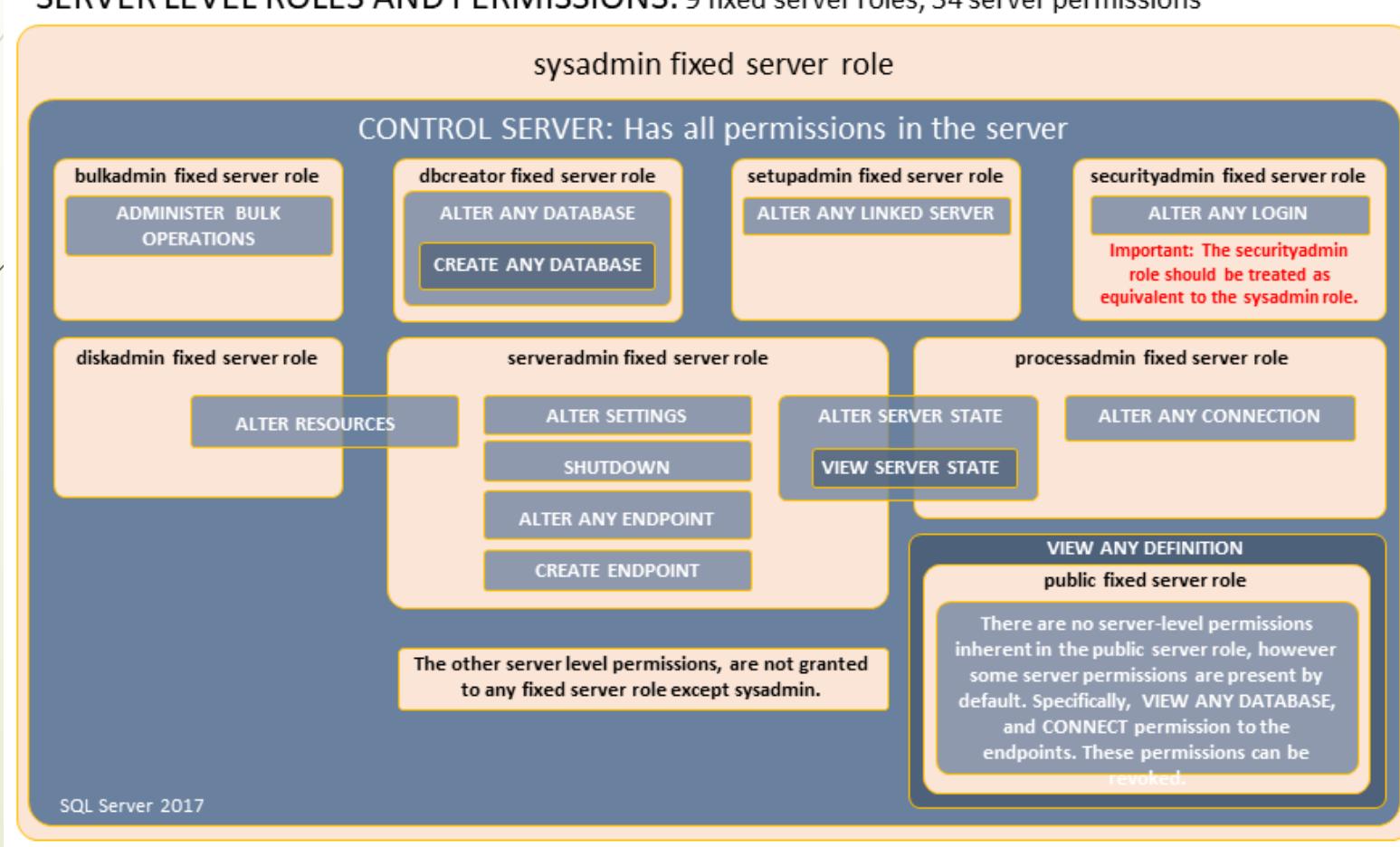
drop server role - ბრძანება შლის მომხმარებლის მიერ განსაზღვრულ სერვერის როლს.

სერვერის დონის როლები

8

SQL Server-ში თითოეულ ფიქსირებულ სერვერის როლს გააჩნია მასზე მინიჭებული გარკვეული ნებართვები, რომელიც გამოსახულია შემდეგ ნახაზზე:

SERVER LEVEL ROLES AND PERMISSIONS: 9 fixed server roles, 34 server permissions



მონაცემთა ბაზის დონის როლები

9

SQL Server-ი უზრუნველყოფს მონაცემთა ბაზების დონის როლებს, რომლებიც მართავენ მონაცემთა ბაზების ნებართვებს. ეს როლები არიან უსაფრთხოების პრინციპები, რომლებიც ჰგუფდებიან სხვა პრინციპებთან. მონაცემთა ბაზების დონის როლები ვრცელდებიან სრულად მონაცემთა ბაზაში მათი ნებართვების ფარგლებში (როლები მსგავსია Windows ოპერაციული სისტემის ჰგუფებსა).

SQL Server-ი უზრუნველყოფს მონაცემთა ბაზის როლში მომხმარებლების დასამატებლად და წასაშლელად გამოიყენეთ ALTER ROLE განცხადების ADD MEMBER და DROP MEMBER ოფციები. მონაცემთა ბაზების დონის ნებართვები მიუწვდომელია SQL Server-ის ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში.

არსებობს ორი ტიპის მონაცემთა ბაზის დონის როლები: ფიქსირებული მონაცემთა ბაზის როლები, რომლებიც წინასწარ არის განსაზღვრული მონაცემთა ბაზაში და მომხმარებლის მიერ განსაზღვრული მონაცემთა ბაზის როლები, რომლებიც შეგიძლიათ შექმნათ.

SQL Server-ში ფიქსირებული მონაცემთა ბაზის როლები განისაზღვრება მონაცემთა ბაზის დონეზე და არსებობს თითოეულ მონაცემთა ბაზაში. მონაცემთა ბაზის დონის db_owner როლს შეუძლია მართოს ფიქსირებული ბაზის როლის წევრობა. ასევე არსებობს სპეციალური დანიშნულების მონაცემთა ბაზის როლები msdb მონაცემთა ბაზაში.

მონაცემთა ბაზის დონის როლები

10

SQL Server-ის მონაცემთა ბაზის დონის როლებში არის შესაძლებლობა დაემატოს მონაცემთა ბაზის ნებისმიერი სააღრიცხვო ჩანაწერი და სხვა SQL Server-ის როლები.

არ არის რეკომენდირებული დაემატოს მომხმარებლის მონაცემთა ბაზის როლები ფიქსირებულ როლების წევრად. ამან შეიძლება გამოიწვიოს გაუთვალისწინებელი პრივილეგიები.

SQL Server-ის მომხმარებლის მონაცემთა ბაზის როლების ნებართვების მართვა შესაძლებელია GRANT, DENY და REVOKE ინსტრუქციების გამოყენებით.

SQL Server-ის სერვერის დონის ნებართვები ან სუბიექტები არ შეიძლება მიენიჭოს ან დაემატოს მონაცემთა ბაზის როლებს.

მონაცემთა ბაზის დონის როლები

11

განვიხილოთ SQL Server-ში მონაცემთა ბაზების დონის ფიქსირებული როლები და მათი შესაძლებლობები:

db_owner - შეუძლია შეასრულოს ყველა მოქმედება მონაცემთა ბაზის კონფიგურაციის, მომსახურებისა და წაშლის. (ზოგიერთ ტექნიკურ აქტივობას ესაჭიროება სერვერის დონის ნებართვები);

db_securityadmin - შეუძლია შეცვალოს როლში წევრობა და უფლებების მართვა (ამ როლის წევრებს შეუძლიათ პოტენციურად აიმაღლონ თავიანთი პრივილეგიები და მათი ქმედებები უნდა მონიტორინგდებოდეს);

db_accessadmin - შეუძლია დაამატოს ან წაშალოს მონაცემთა ბაზაზე დაშორებული წვდომის უფლებები Windows ავთენტიფიკაციებისათვის და ჰგუფებისათვის და, ასევე SQL Server-ის ავთენტიფიკაციებისათვის;

db_backupoperator - შეუძლია მონაცემთა ბაზის სარეზერვო ასლის შექმნა;

db_ddladmin - შეუძლია აწარმოოს მონაცემთა ბაზაში მონაცემთა განსაზღვრის ენის (DDL) ნებისმიერი ბრძანება;

db_datawriter - შეუძლია მომხმარებლის ყველა ცხრილში მონაცემების ფორმატირება (დამატება, წაშლა ან შეცვლა);

მონაცემთა ბაზის დონის როლები

12

db_datareader - შეუძლია წაიკითხოს ყველა ცხრილიდან და წარმოდგენიდან მონაცემები;

db_denydatawriter - არ შეუძლია მონაცემთა ბაზაში მომხმარებლის ცხრილებში მონაცემების ფორმატირება (დამატება, შეცვლა ან წაშლა);

db_denydatareader - არ შეუძლია მონაცემთა ბაზაში მონაცემთა წაკითხვა მომხმარებლის ცხრილებიდან და წარმოდგენებიდან.

მონაცემთა ბაზის დონის როლები

13

ჩამოვთვალოთ SQL Server-ის მონაცემთა ბაზის დონის როლებთან მუშაობის ბრძანებებს, წარმოდგენებს და ფუნქციებს:

sp_helpdbfixedrole - აბრუნებს ფიქსირებული მონაცემთა ბაზის როლების სიას;

sp_dbfixedrolepermission - აჩვენებს ფიქსირებული მონაცემთა ბაზის როლის ნებართვებს;

sp_helprole - აბრუნებს ინფორმაციას როლების შესახებ მიმდინარე მონაცემთა ბაზაში;

sp_helprolemember - აბრუნებს ინფორმაციას როლის წევრების შესახებ მიმდინარე მონაცემთა ბაზაში;

sys.database_role_members - აბრუნებს ერთ სტრიქონს მონაცემთა ბაზის თითოეული როლის თითოეული წევრისთვის;

is_member - მიუთითებს, არის თუ არა მიმდინარე მომხმარებელი მითითებული Windows ჯგუფის წევრი ან SQL Server-ის მონაცემთა ბაზის როლის წევრი;

create role - ბრძანება ქმნის მონაცემთა ბაზის ახალ როლს მიმდინარე მონაცემთა ბაზაში;

alter role - ბრძანება ცვლის მონაცემთა ბაზის როლის სახელს ან წევრობას;

drop role - ბრძანება წაშლის როლის მონაცემთა ბაზაში;

მონაცემთა ბაზის დონის როლები

14

sp_addrole - ბრძანება ქმნის მონაცემთა ბაზის ახალ როლს მიმღინარე მონაცემთა ბაზაში;

sp_droprole - ბრძანება შლის მონაცემთა ბაზის როლს მიმღინარე მონაცემთა ბაზიდან;

sp_addrolemember - ბრძანება ამატებს მიმღინარე მონაცემთა ბაზაში მონაცემთა ბაზის მომხმარებელს, მონაცემთა ბაზის როლს, Windows ავთენტიფიკაციას ან Windows ჯგუფს. ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში ამის მაგივრად გამოიყენება alter role;

sp_droprolemember - ბრძანება შლის მიმღინარე მონაცემთა ბაზაში უსაფრთხოების ჩანაწერს SQL სერვერის როლიდან. ანალიტიკის პლატფორმის სისტემაში (PDW) და Azure Synapse-ში ამის მაგივრად გამოიყენება alter role;

grant - ამატებს როლისათვის ნებართვას;

deny - კრძალავს როლისათვის ნებართვას;

revoke - შლის ადრე მინიჭებულ ან უარყოფილ ნებართვებს.

მაგალითი 1: მონაცემთა ბაზის დონის ფიქსირებულ როლს db_datawriter დავამატოთ მომხმარებლის 'გია'.

ALTER ROLE db_datawriter

მონაცემთა ბაზის დონის როლები

15

მაგალითი 2: დავაძრუნოთ მონაცემთა ბაზის დონის ნებისმიერი როლის ყველა წევრი:

SELECT

roles.principal_id

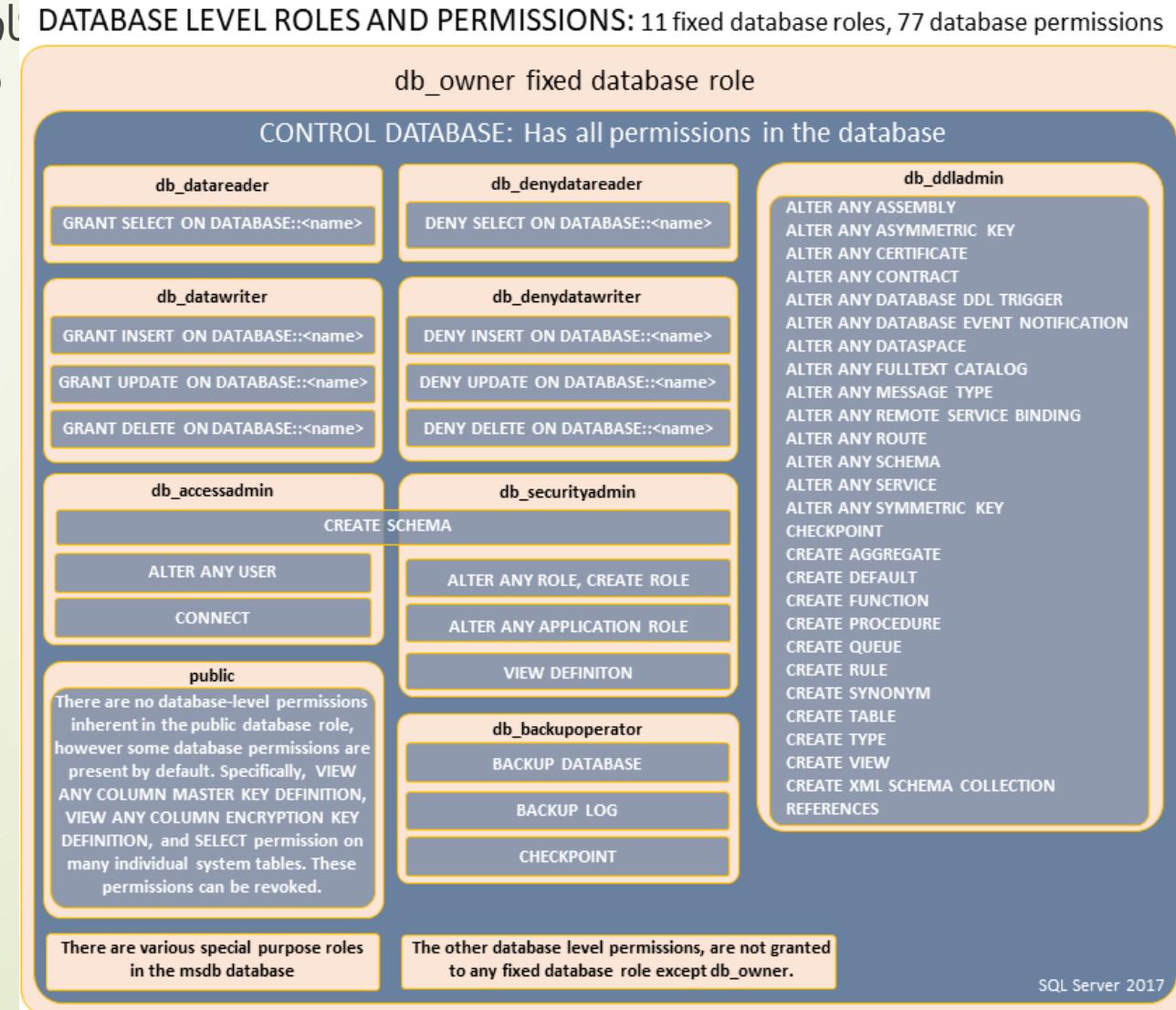
members.principal_id

```
roles.principal_id  
, roles.name  
, database_role_members.member_principal_id  
, members.name  
  
FROM sys.database_role_members AS database_role_members  
JOIN  
sys.database_principals AS roles  
ON database_role_members.role_principal_id =  
  
JOIN  
sys.database_principals AS members  
ON database_role_members.member_principal_id =
```

მონაცემთა ბაზის დონის როლები

16

SQL Server-ის მონაცემთა ბაზის ფიქსირებული როლის შეცვლა არ შეიძლება და მას გააჩნია მასზე მინიჭებული გარკვეული ნებართვები, რომლებიც გამოსახულნი არიან შემდეგ ნახაზე:



აპლიკაციის როლები

17

აპლიკაციის როლი არის მონაცემთა ბაზის პრინციპი, რომელიც საშუალებას აძლევს აპლიკაციას შეასრულოს საკუთარი (მომხმარებლის მსგავსი) წვდომის ნებართვები. მომხმარებელს აქვს შესაძლებლობა გამოიყენოს აპლიკაციის როლები, რათა ჩართოს წვდომა კონკრეტულ მონაცემებზე მხოლოდ იმ მომხმარებლებისთვის, რომლებიც კავშირდებიან SQL Server-ს კონკრეტული აპლიკაციის გამოყენებით. მონაცემთა ბაზის როლებისგან განსხვავებით, აპლიკაციის როლები არ შეიცავს წევრებს და ნაგულისხმევად არიან არააქტიურნი. აპლიკაციის როლები აქტიურდებიან `sp_setapprole` სისტემური პროცედურის გამოყენებით, რომელიც საჭიროებს პაროლის მითითებას. რადგან აპლიკაციის როლები არის მონაცემთა ბაზის დონის პრინციპები, მათ შეუძლიათ სხვა მონაცემთა ბაზებზე წვდომა მხოლოდ ამ მონაცემთა ბაზებში სტუმრისთვის (guest) სააღრიცხვო ჩანაწერის ნებართვებით. ამიტომ, ნებისმიერი მონაცემთა ბაზა, რომელშიც `guest` გამორთულია, მიუწვდომელი იქნება აპლიკაციის როლებისთვის.

აპლიკაციის როლთან დაკავშირების პროცესი გადის შემდეგ ნაბიჯებს, რომლითაც აპლიკაციის როლი ცვლის უსაფრთხოების კონტექსტს: მომხმარებელი აღასრულებს კლიენტის აპლიკაციას; კლიენტის აპლიკაცია უკავშირდება ერთდება SQL Server-ის ეგზემპლიარს, როგორც მომხმარებელი; აპლიკაცია აღასრულებს `sp_setapprole` შენახულ პროცედურას მხოლოდ აპლიკაციისთვის ცნობილი პაროლით; თუ აპლიკაციის როლის სახელი და პაროლი სწორია, ჩაირთვება აპლიკაციის როლი; ამ დროს კავშირი კარგავს მომხმარებლის ნებართვებს და იღებს აპლიკაციის როლის ნებართვებს. რომლებიც ძალაში რჩებიან კავშირის ხანგრძლივობის პერიოდში.

საინფორმაციო სისტემები

1

საინფორმაციო სისტემა არის სტრუქტურირებული (მოწესრიგებული) მონაცემებისა და აპარატურულ-პროგრამული საშუალებების ერთობლიობა, განკუთვნილი მონაცემების შენახვისა და დამუშავებისათვის.

არსებობს საინფორმაციო სისტემების ორი კლასი:

- ▶ საინფორმაციო-სამებნი;
- ▶ მონაცემების დამუშავების.

საინფორმაციო-სამებნი საინფორმაციო სისტემა ორიენტირებულია საჭირო მონაცემების ძებნასა და მიღებაზე, ხოლო მონაცემების დამუშავების საინფორმაციო სისტემა - მონაცემების დამუშავებასა და დამუშავებული ინფორმაციის მიღებაზე. საინფორმაციო სისტემები ასრულებენ შემდეგ ფუნქციებს:

- ▶ ინფორმაციის შეტანა, ცვლილება და შენახვა;
- ▶ ინფორმაციის ნახვა და ძებნა;
- ▶ ინფორმაციის ამოღება სათანადო კრიტერიუმის მიხედვით;
- ▶ ანგარიშების ფორმირება;
- ▶ ინფორმაციის სისწორის შემოწმება.

რისთვის არის საჭირო SQL-ი:

- მონაცემთა ახალი ბაზების, ცხრილებისა და წარმოდგენების შექმნისათვის;
- მონაცემთა ბაზების ცხრილებში ახალი ჩანაწერის ჩასმისათვის, კორექტირებისათვისა და წაშლისათვის;
- მონაცემთა ბაზებიდან მონაცემების ამოღებისთვის.

ამისათვის SQL-ი აკეთებს:

- მოთხოვნას მონაცემთა ბაზებში მრავალი მეთოდის გამოყენებით;
- მონაცემთა ბაზების მართვის სისტემების გამოყენებით მონაცემთა ფორმატირებას;
- მონაცემთა აღწერას;
- მონაცემთა განსაზღვრასა და საჭიროების შემთხვევაში მანიპულირებას;
- მონაცემთა ბაზების, ცხრილებისა და მონაცემთა შექნას, წაშლასა და კორექტირებას;
- წარმოდგენის, პროცედურის, ფუნქციისა და ტრიგერის შექმნას;
- ცხრილებისათვის, პროცედურებისათვის და წარმოდგენებისათვის შეზღუდვების დაწესებას.

მონაცემთა ბაზები

3

ფართო აზრით, მონაცემთა ბაზა არის სათანადო საგნობრივი სფეროს ობიექტების შესახებ მონაცემების მოწესრიგებული ერთობლიობა. საგნობრივი სფერო შეიძლება იყოს უნივერსიტეტი, საავადმყოფო, ფირმა და ა.შ., ობიექტი კი შესაბამისად, სტუდენტი, ავადმყოფი, თანამშრომელი და ა.შ.

პროგრამული საშუალებების ერთობლიობით მონაცემთა ბაზასთან მუშაობა შეიძლება მონაცემთა ბაზის მართვის სისტემით, რომელიც საჭიროა მონაცემთა ბაზის შესაქმნელად და მასში მოთავსებულ ობიექტებზე მანიპულირებისათვის.

არსებობს მონაცემთა განაწილებული (კლასტერული) და ცენტრალიზებული (არაკლასტერული) ბაზები. მონაცემთა განაწილებული ბაზა მოთავსებულია კომპიუტერული ქსელის რამდენიმე კომპიუტერზე (სერვერზე), ხოლო მონაცემთა ცენტრალიზებული ბაზა მოთავსებულია ერთ კომპიუტერზე, რომელზეც მიმართვა ექნებათ სხვა კომპიუტერებს კომპიუტერული ქსელის გამოყენებით.

მონაცემთა ბაზები არქიტექტურის მიხედვით ორგვარია: ფაილ-სერვერი და კლიენტ-სერვერი. ფაილ-სერვერული არქიტექტურის შემთხვევაში მონაცემთა ბაზა განთავსებულია ერთ ან მეტ ფაილ-სერვერზე, რომელიც წარმოადგენს ქსელში ჩართულ მძლავრ კომპიუტერს. მონაცემების დამუშავება სრულდება ლოკალურ კომპიუტერებზე. ფაილ-სერვერული მონაცემთა ბაზების მართვის სისტემებია: Microsoft Visual FoxPro, Microsoft Access, Paradox for Windows, dBase for Windows და ა.შ.

მონაცემთა ბაზები

4

კლიენტ-სერვერული არქიტექტურის შემთხვევაში მონაცემთა ბაზა განთავსებულია სერვერზე და აქვე ხდება მისი დამუშავება. კლიენტის მხრიდან მონაცემების დამუშავების მოთხოვნა სერვერს ეგზავნება. მასზე სრულდება მონაცემების დამუშავება და შედეგები კლიენტს ეგზავნება. სისტემა, რომელიც იყენებს კლიენტ-სერვერულ ტექნოლოგიას ორი ნაწილისაგან შედგება: კლიენტის ნაწილი (front-end) და სერვერის ნაწილი (back-end).

კლიენტის ნაწილი უზრუნველყოფს გრაფიკულ ინტერფეისს და იმყოფება მომხმარებლის კომპიუტერზე, ხოლო სერვერის ნაწილი მოთავსებულია სერვერზე და უზრუნველყოფს მონაცემების მართვას, დანაწილებას, ადმინისტრირებასა და უსაფრთხოებას. კლიენტ-სერვერული მონაცემთა ბაზების მართვის სისტემებია: Microsoft SQL Server, Oracle, IBM DB2, Sybase და ა.შ. ამ არქიტექტურისათვის დამახასიათებელია მოთხოვნების სტრუქტურირებული ენის (SQL, Structured Query Language) გამოყენება.

აქვე უნდა აღინიშნოს, რომ SQL არის მხოლოდ მოთხოვნების დამუშავების ენა, რომლის მუშაობისათვის საჭიროა ნებისმიერი მონაცემთა ბაზის დაყენება (Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2 და სხვ.). რაც შეეხება დასახელებას საერთაშორისო საზოგადოებაში მას უფრო See-Qwell-ს უძახიან.

SQL-ს როგორც მონაცემთა ბაზების სტანდარტულ ენას მოიხმარენ ისეთი მონაცემთა ბაზების მართვის სისტემები, როგორიცაა: Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, MS Access, Sybase, Informix და სხვა.

მონაცემთა ბაზები

5

მონაცემთა ბაზების კლასიფიცირება ხორციელდება მონაცემების მოდელების მიხედვით. მონაცემების მოდელი არის მონაცემების სტრუქტურისა და მათი დამუშავების ოპერაციების ერთობლიობა. მონაცემების მოდელის მიხედვით შესაძლებელია ობიექტების სტრუქტურისა და მათ შორის არსებული კავშირების წარმოდგენა. არსებობს მონაცემების შემდეგი მოდელები: იერარქიული, ქსელური და რელაციური. შესაბამისად, არსებობს იერარქიული, ქსელური და რელაციური მონაცემთა ბაზები.

მონაცემების იერარქიული მოდელის თითოეული ელემენტი სხვა ელემენტებს წარმოქმნის. წარმოქმნილი ელემენტები, თავის მხრივ, სხვა ელემენტებს წარმოქმნის და ა.შ. ამასთანავე, ნებისმიერ წარმოქმნილ ელემენტს აქვს მხოლოდ ერთი წარმომქმნელი ელემენტი.

იერარქიული სტრუქტურის მაგალითია უნივერსიტეტი, საავადმყოფო და ა.შ. მონაცემების ასეთი ორგანიზების ნაკლია ის, რომ შეუძლებელია შემუშავებული იერარქიის ცვლილება და მოგვიწევს მხოლოდ იმ იერარქიის გამოყენება, რომელიც შექმნილი იყო მონაცემთა ბაზის დაპროექტების დროს. ამიტომ დადგა უფრო ზოგადი - ქსელური მოდელის შექმნა.

მონაცემების ქსელური მოდელის გამოყენებით წარმოქმნილ ელემენტს შეიძლება ჰქონდეს ერთზე მეტი წარმომქმნელი ელემენტი. ქსელური სტრუქტურის მაგალითია მაღაზია, სუპერმარკეტი და ა.შ. მაღაზიის გამყიდველი ემსახურება რამდენიმე მომხმარებელს. ამასთან, თითოეულ მომხმარებელს შეიძლება მოემსახუროს რამდენიმე გამყიდველი. ქსელურ მოდელს აქვს ნაკლი - იმისათვის, რომ მონაცემები დავამუშავოთ, უნდა ვიცოდეთ მონაცემთა ბაზის სტრუქტურა.

მონაცემთა რელაციური ბაზა

6

მონაცემთა რელაციური მოდელის შემთხვევაში მონაცემების წარმოდგენა ხდება ორგანზომილებიანი ცხრილის სახით. მარტივ შემთხვევაში, რელაციური მოდელი აღწერს ერთ ორგანზომილებიან ცხრილს, უფრო ხშირად კი - რამდენიმე ცხრილის სტრუქტურასა და მათ შორის კავშირებს. რელაციური მოდელი ეფუძნება მათემატიკის ორ განყოფილებას: სიმრავლეთა თეორიასა და ლოგიკას (პრედიკატების აღრიცხვას).

მონაცემთა რელაციური ბაზების თეორია წინა საუკუნის სამოცდაათიან წლებში შეიმუშავა ამერიკელმა მათემატიკოსმა ე. კოდმა. მან მონაცემების დამუშავებისათვის შემოგვთავაზა სიმრავლეთა თეორიის აპარატი და დაამტკიცა, რომ მონაცემთა ნებისმიერი ნაკრები შეიძლება ორგანზომილებიანი ცხრილის სახით წარმოვადგინოთ. სიტყვა „რელაციური“ წარმოდგება ინგლისური სიტყვისაგან „relation“, რაც მიმართებას ნიშნავს. მიმართებების წარმოდგენა მოხერხებულია ორგანზომილებიანი ცხრილების სახით. ამრიგად, რელაციური არის მონაცემთა ისეთი ბაზა, რომელშიც მონაცემები წარმოდგენილია სწორკუთხა ორგანზომილებიანი ცხრილებით.

ცხრილს აქვს სახელი, რომელიც უნიკალურია მონაცემთა ბაზის შიგნით. ის შედგება სვეტებისა (ველებისა) და სტრიქონებისაგან (ჩანაწერებისაგან). ცხრილი შეიცავს ინფორმაციას ერთტიპური ობიექტების შესახებ. ცხრილის სტრიქონი შეიცავს ინფორმაციას კონკრეტული ობიექტის შესახებ, ხოლო ცხრილის თითოეული სვეტი წარმოადგენს ობიექტების კონკრეტული ატრიბუტის მნიშვნელობების ერთობლიობას.

მონაცემთა რელაციური ბაზა

7

სვეტს (ველს) აქვს სახელი და ტიპი, რომლის სახელიც უნიკალურია მონაცემთა ცხრილის შიგნით. თუმცა, განსხვავებულ ცხრილებში შეიძლება მოხმარებულ იქნეს ერთიდაიგივე სახელის მქონე სვეტები. ცხრილს უნდა გააჩნდეს ერთი სვეტი მაინც. ცხრილის სტრიქონებს სახელები არ აქვთ, მათი რიგითობა განსაზღვრული და რაოდენობა შეზღუდული არ არის.

მონაცემთა რელაციური ბაზა არის ერთმანეთთან ლოგიკურად დაკავშირებული ცხრილების ერთობლიობა, რომლის დაპროექტებისას უნდა დავიცვათ შემდეგი წესები:

- ▶ მონაცემთა ბაზაში თითოეულ ცხრილს უნდა ჰქონდეს უნიკალური სახელი და უნდა შედგებოდეს ერთტიპური სტრიქონებისაგან;
- ▶ თითოეული ცხრილი შედგება სვეტებისა და მნიშვნელობების ფიქსირებული რაოდენობისაგან;
- ▶ სტრიქონის ერთ სვეტში მოთავსებული უნდა იყოს მხოლოდ ერთი მნიშვნელობა;
- ▶ ცხრილში არ უნდა იყოს ზუსტად ერთნაირი ორი სტრიქონი - სტრიქონები უნდა განსხვავდებოდნენ ერთი სვეტის მნიშვნელობით მაინც;
- ▶ ცხრილის შიგნით თითოეულ სვეტის სახელი უნდა იყოს უნიკალური;
- ▶ სვეტისთვის უნდა განისაზღვროს მასში მოთავსებული მონაცემების ტიპი
- ▶ მონაცემების დამუშავებისას უნდა შეიძლებოდეს მიმართვა ნებისმიერ სტრიქონთან ან სვეტთან.

მონაცემთა რელაციური ბაზა

8

მონაცემთა რელაციური ბაზა უნდა აკმაყოფილებდეს ნორმალიზაციის წესებს:

1. მონაცემთა ბაზაში საჭირო მონაცემების შენახვის უზრუნველყოფა;
2. მონაცემების სიჭარბის გამორიცხვა;
3. ცხრილების რაოდენობის მინიმუმამდე დაყვანა.

ცხრილების ნორმალიზება არის მონაცემების წარმოდგენის პროცესი მარტივი ორგანზომილებიანი ცხრილებით, რომელიც საშუალებას გვაძლევს თავიდან ავიცილოთ მონაცემების დუბლირება და არაწინააღმდეგობა.

ნორმალიზების მიზანია მონაცემთა ბაზის ისეთი პროექტის ფარგლებში ინფორმაციის ნებისმიერი ნაწილი შენახული იქნება მხოლოდ ერთხელ და ერთ ადგილზე, რაც გამორიცხავს ინფორმაციის სიჭარბეს. ეს კეთდება ადგილის ეკონომიის და შენახული მონაცემების წინააღმდეგობრიობის გამორიცხვისათვის მიზნით.

ცხრილი ითვლება ნორმალიზებულად თუ იგი აკმაყოფილებს ნორმალიზების ფორმის მოთხოვნებს. არსებობს ცხრილის ნორმალიზების ექვსი ფორმა. ყველაზე ხშირად, ნორმალიზების პირველი სამი ფორმა გამოიყენება, რადგან მათი რეალიზება საკმაოდ ადვილია და ნორმალიზების საკმარის დონეს უზრუნველყოფენ.

ცხრილებს შორის კავშირები

9

ცხრილის ნორმალიზების შემდეგ მიიღება ცხრილები, რომლებიც უნდა იყვნენ ერთმანეთთან დაკავშირებულნი, რისთვისაც გამოიყენება პირველადი და გარე გასაღებები.

ძირითადად, ორ ცხრილს შორის კავშირის დამყარების დროს ერთი არის მთავარი (მშობელი), მეორე კი - დამოკიდებული (შვილობილი). კავშირის დასამყარებლად მთავარი ცხრილის პირველადი გასაღები უკავშირდება დამოკიდებული ცხრილის გარე გასაღებს. როცა მთავარ ცხრილში ავირჩევთ რომელიმე სტრიქონს, მაშინ დამოკიდებული ცხრილიდან ამოირჩევა შესაბამისი სტრიქონები.

ნორმალიზების მეორე ფორმიდან გამომდინარე, ნებისმიერ მონაცემთა ცხრილს გააჩნია ერთი ან მეტი სვეტი, რომლებიც ცალსახად განსაზღვრავენ თითოეულ სტრიქონს. ასეთ სვეტ(ებ)ს პირველადი გასაღები (**PRIMARY KEY**) ეწოდება, რომელიც უნდა აკმაყოფილებდეს შემდეგ მოთხოვნებს:

- ▶ **უნიკალურობა** - ცხრილში არ უნდა იყოს ორი ისეთი სტრიქონი, რომლებსაც პირველადი გასაღების ერთნაირი მნიშვნელობები აქვთ;
- ▶ **მინიმალურობა** - პირველად გასაღებში შემავალი სვეტებიდან ნებისმიერის გამოკლება იწვევს უნიკალურობის დარღვევას.

ცხრილებს შორის კავშირები

10

პირველად გასაღებისათვის გასათვალისწინებელია რამდენიმე რჩევა:

- არ შეიძლება უნიკალური სვეტ(ებ)ის მნიშვნელობის ცვლილება, რადგან იგი გამოიწვევს უნიკალურობის რღვევას (მაგ., თუ უნიკალური სვეტი არის „გვარი“, პიროვნების გვარის შეცვლის შემთხვევაში დაიკარგება ინფორმაცია როგორც მთავარ, ასევე დამოკიდებულ ცხრილებში);
- როდესაც პირველადი გასაღები შედგება რამდენიმე სვეტისაგან, უმჯობესია დამატებით შეიქმნას ერთი სვეტი და ის განისაზღვროს როგორც უნიკალური გასაღები.

გარე გასაღები (FOREIGN KEY) იქმნება დამოკიდებულ ცხრილში, რომელიც მთავარი ცხრილს მიმართავს. იგი, ასევე, შეიძლება შედგებოდეს ერთი ან მეტი სვეტისაგან. გარე გასაღების მნიშვნელობა უნდა არსებობდეს მთავარ ცხრილში, ან უნდა იყოს განუსაზღვრელი. ასევე, მთავარი ცხრილის პირველადი გასაღების რომელიმე მნიშვნელობა შეიძლება არ იყოს დაკავშირებული დამოკიდებული ცხრილის არც ერთ სტრიქონის მნიშვნელობასთან.

ცხრილებს შორის კავშირები

11

ცხრილებს შორის კავშირის შექმნისას შესაძლებელია დაყენებულ იქნეს პირველადი გასაღების შეცვლა/წაშლისას მოქმედების ოთხი ვარიანტი:

- ▶ **უმოქმედოდ (No Action)** - მთავარ ცხრილში პირველადი გასაღების მნიშვნელობების შეცვლა/წაშლისას დამოკიდებულ ცხრილში არაფერი არ შეიცვლება (კავშირი იკარგება);
- ▶ **კასკადური (Cascade)** - მთავარ ცხრილში პირველადი გასაღების მნიშვნელობების შეცვლა/წაშლისას იგივე განხორციელდება დამოკიდებულ ცხრილის იგივე მნიშვნელობების მქონე სტრიქონებში (კავშირი არ იკარგება);
- ▶ **განუსაზღვრელობა (Set Null)** - მთავარ ცხრილში პირველადი გასაღების მნიშვნელობის შეცვლა/წაშლისას დამოკიდებულ ცხრილში გარე გასაღები იღებს განუსაზღვრელ (NULL) მნიშვნელობას (კავშირი იკარგება).
- ▶ **შეთანხმებით (Set Default)** - დასაშვებია მთავარ ცხრილში პირველადი გასაღების მხოლოდ იმ მნიშვნელობის შეცვლა/წაშლა, რომლებიც არ არის დაკავშირებული დამოკიდებული ცხრილის სტრიქონებთან (კავშირი არ იკარგება).

პრაქტიკიდან გამომდინარე შეიძლება ჩაითვალოს ყველაზე მომგებიან ვარიანტად ცვლილებისას კასკადური (კასკადურად განხორციელდება ცვლილება ყველა დამოკიდებულ ცხრილში), ხოლო წაშლისას - შეთანხმებით (წაიშლება მხოლოდ ის სტრიქონები, რომლებსაც არ გააჩნიათ შესაბამისი მნიშვნელობის სტრიქონები ყველა დამოკიდებულ ცხრილში).

ცხრილებს შორის კავშირები

12

ნორმალიზების მესამე ფორმაზე დაყვანის შემდეგ, მთავარი ცხრილის მონაცემების სრული ინფორმაციის სანახავად საჭირო გახდა კავშირის განხორციელება დამოკიდებულ ცხრილთან, რისთვისაც გამოიყენება პირველადი და გარე გასაღებები. არსებობს სამი ტიპის კავშირი მონაცემთა ცხრილებს შორის:

- ▶ „ერთი-ერთთან“ - ერთი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მეორე ცხრილის ერთ ან არცერთ სტრიქონთან. ასეთი შემთხვევა წარმოიქმნება, როდესაც ერთი დიდი ცხრილი დაიყოფა ორ ან რამდენიმე ცხრილად და ყველა ცხრილში უნიკალური გასაღები იქნება მთავარი ცხრილის უნიკალური გასაღები;
- ▶ „ერთი-ბევრთან“ - მთავარი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია დამოკიდებული ცხრილის არცერთ, ერთ ან მეტ სტრიქონთან, ხოლო დამოკიდებული ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მთავარი ცხრილის მხოლოდ ერთ სტრიქონთან. მონაცემთა ბაზებში უმეტეს შემთხვევაში ასეთი კავშირები წარმოიქმნება;
- ▶ „ბევრი-ბევრთან“ - ერთი ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია მეორე ცხრილის არცერთ, ერთ ან მეტ სტრიქონთან და პირიქითაც იგივეა. ასეთი კავშირების დროს შუაში უნდა იქნეს დამატებული ცხრილი, რომელთანაც ორივე ცხრილს ექნება კავშირი „ერთი-ბევრთან“.

მონაცემთა ბაზების სტრუქტურები

13

SQL Server-ზე მონაცემები მონაცემთა ბაზებში ინახება. არსებობს მონაცემთა ბაზის ორი სტრუქტურა: ლოგიკური და ფიზიკური.

მონაცემთა ბაზის ლოგიკური სტრუქტურა მოიცავს: ცხრილების სტრუქტურას, მათ შორის კავშირებს, მომხმარებლების სიას, შენახულ პროცედურებს და მონაცემთა ბაზის სხვა ობიექტებს.

მონაცემთა ბაზის ფიზიკური სტრუქტურა მოიცავს: მონაცემთა ბაზის ფაილებისა და ტრანზაქციების ჟურნალის აღწერას, მათ საწყის ზომას, ნაზარდს ზომას, მაქსიმალურ ზომას, კონფიგურირების პარამეტრებს და სხვას.

მონაცემთა ბაზების ობიექტები

14

მონაცემთა ბაზების ობიექტებია:

- ▶ **ცხრილები (Tables)** - ორგანზომილებიანი მატრიცებია, რომლებშიც უშუალოდ მონაცემები ინახება;
- ▶ **მონაცემთა ტიპები (Data Types)** - სისტემური მონაცემთა ტიპებია;
- ▶ **პირველადი (Primary Key) და გარე (Foreign Key) გასაღებები** - მონაცემთა ცხრილებს შორის კავშირების განსახორციელებელი სვეტები;
- ▶ **ცხრილებს შორის კავშირები** - მთავარი ცხრილის მონაცემების დამოკიდებულ ცხრილის მონაცემებთან კავშირში სრული ინფორმაციის სანახავად;
- ▶ **ნაგულისხმევი მნიშვნელობა (Default Value or Binding)** - მონაცემთა ბაზის ობიექტებია, რომლებიც შეგვიძლია მივუთითოთ უშუალოდ ცხრილის სტრუქტურაში ან მომხმარებლების მიერ განსაზღვრულ ტიპებში, რომლის მნიშვნელობაც აისახება მონაცემის არ შეტანის შემთხვევაში;
- ▶ **თვლადი სვეტი (Computed Column)** - მონაცემთა ცხრილის სტრიქონის სვეტში ამავე სტრიქონის სხვა სვეტებისაგან შემდგარი მათემატიკური მოქმედების ჩაწერა;

მონაცემთა ბაზების ობიექტები

15

ასევე მონაცემთა ბაზების ობიექტებია:

- ▶ **წარმოდგენები (Views)** - ვირტუალური ცხრილებია, რომლებიც საშუალებას გვაძლევს ამოღების შედეგთან ვიმუშაოთ როგორც ცხრილთან;
- ▶ **ინდექსები (Indexes)** - სტრუქტურებია, რომლებიც ზრდის მონაცემებთან მუშაობის მწარმოებლურობას;
- ▶ **ტრიგერები (Triggers)** - სპეციალური შენახული პროცედურებია, რომლებიც ცხრილში მონაცემების ცვლილების დროს გამოიძახება;
- ▶ **მომხმარებლების ფუნქციები (user-defined functions)** - მომხმარებლების მიერ შექმნილი ფუნქციებია;
- ▶ **შენახული პროცედურები (stored procedures)** - Transact_SQL-ის ბრძანებების ნაკრებია, შენახული გარკვეული სახელით. მათთან მიმართვა შესაძლებელია სახელის საშუალებით;
- ▶ **მთლიანობის შეზღუდვები (constraints)** - ობიექტებია, რომლებიც უზრუნველყოფენ მონაცემების ლოგიკურ მთლიანობას და არ არსებობს ცხრილებისაგან დამოუკიდებლად.

მონაცემთა ბაზების ობიექტები

1

მონაცემთა ბაზის ობიექტებიდან მხოლოდ ცხრილი შეიცავს საკუთრივ მონაცემებს. როგორც ითქვა რელაციურ მონაცემთა ბაზებში არის ორ განზომილებიანი მონაცემთა ცხრილები, რომლებიც შედგებიან სტრიქონებისა და სვეტებისაგან:

- ▶ თითოეული სტრიქონი (**row**) წარმოადგენს კონკრეტული ობიექტის მახასიათებლების ერთობლიობას.
მაგალითად, სტუდენტების ცხრილისათვის სტრიქონი წარმოადგენს კონკრეტული სტუდენტის მახასიათებლებს: გვარი, სახელი, უნივერსიტეტის დასახელება, კურსი და სხვ.
- ▶ თითოეული სვეტი (**column**) წარმოადგენს ობიექტების კონკრეტულ მახასიათებელს.
მაგალითად, სტუდენტების ცხრილისათვის სვეტი წარმოადგენს სტუდენტების კონკრეტულ მახასიათებლებს: გვარი ან სახელი ან უნივერსიტეტის დასახელება ან კურსი ან სხვ.
სვეტი არის ცხრილის მინიმალური ელემენტი და მას აქვს სახელი და ტიპი (შესაძლებელია ზომაც გარკვეული ტიპისათვის).
ცხრილის ზოგიერთი სვეტი შეიძლება იყოს გამოთვლადი (computed). ასეთ სვეტებში ეთითება ფორმულა, რომლის მიხედვით სვეტის მნიშვნელობა გამოითვლება.

ცხრილის ყოველ სვეტს გააჩნია განსაზღვრული ტიპი, რომელიც მიუთითებს თუ როგორი ტიპის ინფორმაცია იქნება მოთავსებული ამ სვეტში.

მონაცემთა ბაზების ობიექტები

2

ზოგჯერ საჭიროა მონაცემთა ცხრილში სტრიქონის დამატებისას გარკვეულ სვეტში თუ არ არის ცხადად მითითებული მნიშვნელობა ავტომატურად ჩაისვას მნიშვნელობა, რომელსაც ნაგულისხმევი (ავტომატური) მნიშვნელობა (**Default Value or Binding**) ეწოდება.

ნაგულისხმევი მნიშვნელობა შეიძლება იყოს მუდმივა, ჩადგმული ფუნქციის ან მათემატიკური გამოსახულების მიერ გადაცემული შედეგი. გასათვალისწინებელია, რომ ნაგულისხმევი მნიშვნელობა არ უნდა იყოს კონფლიქტში მთლიანობის შეზღუდვებთან.

ზოგ მონაცემთა ცხრილს ვერ უთითებთ პირველად გასაღებს, რადგან არ გააჩნია თავისი უნიკალური მონაცემი და ამ ცხრილთან მუშაობისას უნდა განხორციელდეს სვეტი-მთვლელის დამატება და განსაზღვრა, რომლისთვისაც შესრულდება უნიკალური მნიშვნელობების ავტომატურად გენერირება **IDENTITY** საკვანძო სიტყვის გამოყენებით. სვეტი-მთვლელის განსაზღვრისას უნდა მივუთითოთ მისი საწყისი მნიშვნელობა და ბიჯი (მითითების გარეშე სისტემა შეთანხმებით ორივეს მნიშვნელობას 1-ის ტოლს იღებს). სვეტისთვის **IDENTITY** თვისების განსაზღვრისას შემდეგი მოთხოვნები უნდა დავიცვათ:

- ▶ სვეტის ტიპი უნდა იყოს: BIGINT, DECIMAL, INT, NUMERIC, SMALLINT ან TINYINT;
- ▶ სვეტისთვის აკრძალული უნდა იყოს NULL მნიშვნელობის შენახვა;
- ▶ სვეტისთვის არ უნდა იყოს განსაზღვრული ნაგულისხმევი, თვლადი, ფუნქცია და სხვა მნიშვნელობა.

მონაცემთა ბაზების ობიექტები

3

ზოგჯერ საჭიროა მონაცემთა ცხრილის სტრიქონის გარკვეულ სვეტში განხორციელდეს ამავე სტრიქონის სხვა სვეტებისაგან და მუდმივებისაგან შემდგარი მათემატიკური გამოსახულების ჩაწერა. ასეთ სვეტს გამოთვლადი სვეტი (Computed Column) ეწოდება. გასათვალისწინებელია, რომ გამოთვლადი სვეტის ფორმულაში შეიძლება გამოყენებულ იქნეს მხოლოდ მათემატიკური მოქმედებები.

წარმოდგენები (Views) არიან ვირტუალური ცხრილები, რომელიც შედგებიან სხვა ცხრილ(ებ)ის ან/და წარმოდგენ(ებ)ისაგან და ისინი განისაზღვრებიან SELECT მოთხოვნით. წარმოდგენა მონაცემებს არ შეიცავს - იგი გამოსახავს შემავალი ცხრილ(ებ)ის მონაცემებს. მონაცემები, რომლებიც გამოისახება ეკრანზე წარმოდგენის საშუალებით, ცალკე არ ინახება მონაცემთა ბაზაში. მარტივი წარმოდგენა იქმნება ერთი ცხრილის ბაზაზე და შეიძლება შეიცავდეს სვეტების ფილტრებს და სორტირებებს, ხოლო რთული წარმოდგენა შედგება რამდენიმე დაკავშირებული ცხრილის ან/და წარმოდგენის საფუძველზე. წარმოდგენასთან მიმართვა ხორცილდება ცხრილის ანალოგიურად სახელის გამოყენებით. წარმოდგენა გამოიყენება შემდეგ შემთხვევებში:

- მონაცემთა ცხრილ(ებ)ის გარკვეულ სტრიქონებთან/სვეტებთან მომხმარებლების მიმართვის შესაზღუდად;
- მონაცემთა დაკავშირებული ცხრილების მონაცემების ერთი ობიექტის სახით წარმოსადგენად;
- ინფორმაციის სანახავად, რომელიც მიიღება მონაცემების გარდაქმნის შედეგად.

მონაცემთა ბაზების ობიექტები

4

ინდექსები (Indexes) არიან ცხრილთან ან წარმოდგენასთან დაკავშირებული სტრუქტურა და განეკუთვნებიან შესაბამის ცხრილში ან წარმოდგენაში ინფორმაციის ძებნის დასაჩქარებლად. ინდექსი განისაზღვრება ერთი ან მეტი სვეტისთვის, რომლებსაც ინდექსირებული სვეტები ეწოდებათ. ინდექსი შეიცავს ინდექსირებული სვეტის ან სვეტების დახარისხებულ მნიშვნელობებს საწყისი ცხრილის ან წარმოდგენის შესაბამის სტრიქონზე მიმართვებთან ერთად. მწარმოებლურობის ზრდა მიიღწევა სვეტის მონაცემების დახარისხების ხარჯზე.

ფუნქცია არის კონსტრუქცია, რომელიც შეიცავს ხშირად გამოყენებად კოდს და გააჩნია სახელი. ფუნქცია მონაცემებზე გარკვეულ მოქმედებებს ასრულებს და აბრუნებს კონკრეტულ წინასწარ განსაზღვრულ მნიშვნელობას. მისი გამოძახება მონაცემთა ცხრილისა და წარმოდგენის ანალოგიურად სახელით ხორციელდება ფრჩხილებში პარამეტრების მითითებით. არსებობს მომხმარებლის ფუნქციების ორი ტიპი:

1. ჩადგმული ფუნქციები (Built-in Functions) - სერვერის გარემოს შემადგენელი ნაწილია და სისტემის მიერ წინასწარ განსაზღვრულ მოქმედებებს ასრულებს (მაგ., MAX(), SUM() და სხვ.);
2. მომხმარებლის ფუნქციები (User-Defined Functions) - პროგრამირების გარემოს შემადგენელი ნაწილია და მომხმარებლის მიერ წინასწარ განსაზღვრულ მოქმედებებს ასრულებს.

მონაცემთა ბაზების ობიექტები

5

ტრიგერები (Triggers) არიან შენახული პროცედურის სპეციალური კლასი, რომლებიც ავტომატურად გაიშვება ცხრილებში მონაცემების დამატების, ცვლილების ან წაშლის დროს. ტრიგერები იყოფიან სამ კატეგორიად: ჩამატების ტრიგერები (INSERT TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში ახალი სტრიქონის დამატების შემდეგ; ცვლილების ტრიგერები (UPDATE TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში არსებული სტრიქონის ჩასწორების შემდეგ; წაშლის ტრიგერები (DELETE TRIGGER) - მუშაობას იწყებს მონაცემთა ცხრილში არსებული სტრიქონის წაშლის შემდეგ. ერთ ტრიგერში შესაძლებელია იყოს როგორც ცალ-ცალკე ჩამატების, ცვლილებისა და წაშლის, ასევე მათი კომბინაციის მქონეც. ერთი ცხრილისთვის შესაძლებელია განსაზღვრული იყოს თითოეული ტიპის რამდენიმე ტრიგერი. გარდა ამისა, ერთი ტრიგერიდან შესაძლებელია სხვა ტრიგერების გამოძახება - ასეთ შემთხვევაში გვაქვს ჩადგმული ტრიგერები (nested triggers).

შენახული პროცედურები (Stored Procedures) არიან ერთ მოდულში გაერთიანებული ბრძანებების ჯგუფი, რომელსაც აქვს სახელი. ეს ბრძანებები კომპილირდება და სრულდება, როგორც ერთი მთლიანი. SQL Server-ზე ინახება სისტემური შენახული პროცედურების დიდი რაოდენობა. მათი სახელები იწყება „sp_“ პრეფიქსით. ისინი გამოიყენება SQL Server-ის კონფიგურირებისა და მართვისათვის, სისტემურ მონაცემთა ბაზებსა და ცხრილებში მონაცემების შესაცვლელად და ა.შ. შესაძლებელია საკუთარი შენახული პროცედურის შექმნა, რომელიც მოთავსებული იქნება არსებულ მონაცემთა ბაზაში. შენახული პროცედურის შესასრულებლად გამოიყენება მისი სახელი და საჭიროების შემთხვევაში მიეთითება პარამეტრები.

მონაცემთა ბაზების ობიექტები

6

მთლიანობის შეზღუდვები (Constraints) უზრუნველყოფენ ავტომატურ კონტროლს ჩვენ მიერ განსაზღვრულ პირობების/შეზღუდვების მონაცემების შესაბამისობაში ლოგიკურ დონეზე. მთლიანობის შეზღუდვები შეიძლება გამოყენებული იყოს სვეტის ან ცხრილის დონეზე. მთლიანობის შეზღუდვები, რომლებიც სვეტის დონეზე გამოიყენება, მოქმედებს მხოლოდ ამ სვეტში შესატან მონაცემებზე. თუ მთლიანობის შეზღუდვა გამოიყენება რამდენიმე სვეტის მიმართ, მაშინ შეზღუდვა მუშაობს ცხრილის დონეზე. ფუნქციურობის და გამოყენების მიხედვით არსებობს მთლიანობის შეზღუდვის ხუთი ტიპი:

1. NULL (უცნობი) მთლიანობის შეზღუდვა. ის მოქმედებს სვეტისა და მომხმარებლის ტიპის დონეზე. მათთვის შეგვიძლია განვსაზღვროთ NULL ან NOT NULL მთლიანობის შეზღუდვა;
2. CHECK (შემოწმება) მთლიანობის შეზღუდვა მოქმედებს სვეტის დონეზე და ზღუდავს სვეტში შესანახი მნიშვნელობების დიაპაზონს. მისი განსაზღვრისას შესატანი მონაცემებისათვის ეთითება ლოგიკური პირობა. მონაცემების შეტანის ან ჩამატების დროს შესატანი მნიშვნელობა მოთავსდება პირობაში. თუ შემოწმების შედეგია TRUE (ჭეშმარიტი), მაშინ მონაცემების ცვლილება ნებადართული იქნება, ხოლო თუ შემოწმების შედეგია FALSE (მცდარი), მაშინ ცვლილებები აიკრძალება და გაიცემა შეტყობინება შეცდომის შესახებ. ერთი სვეტისთვის შეგვიძლია შევქმნათ რამდენიმე CHECK შეზღუდვა. მისი მითითება შეგვიძლია ცხრილის შექმნის დროს;

მონაცემთა ბაზების ობიექტები

7

3. UNIQUE (უნიკალურობა) მთლიანობის შეზღუდვა მოქმედებს სვეტის დონეზე და იძლევა სვეტში მნიშვნელობების უნიკალურობის გარანტიას. ცხრილში არ იქნება ორი სტრიქონი, რომლებსაც მოცემულ სვეტში ერთნაირი მნიშვნელობები ექნებათ. პირველადი გასაღებისაგან განსხვავებით UNIQUE მთლიანობის შეზღუდვა უშვებს ერთჯერადად NULL მნიშვნელობის არსებობას;
4. PRIMARY KEY (პირველადი გასაღები) მოქმედებს სვეტის ან ცხრილის დონეზე. პირველადი გასაღები შეიძლება შედგებოდეს ერთი ან მეტი სვეტისაგან და არის სტრიქონის უნიკალური იდენტიფიკატორი ცხრილის ფარგლებში. თუ პირველადი გასაღები ერთი სვეტისაგან შედგება, მაშინ ამ სვეტისთვის განისაზღვრება UNIQUE შეზღუდვა, ხოლო თუ პირველადი გასაღები რამდენიმე სვეტისაგან შედგება, მაშინ თითოეულ სვეტში მნიშვნელობები შეიძლება გამეორდეს ანუ არ იყოს უნიკალური, მაგრამ უნიკალური უნდა იყოს ამ სვეტების მნიშვნელობების კომბინაცია თითოეული სტრიქონისათვის. პირველად გასაღებში შემავალი არც ერთი სვეტისათვის არ უნდა იყოს დაყენებული NULL შეზღუდვა. ცხრილში შეგვიძლია შევქმნათ მხოლოდ ერთი პირველადი გასაღები.
5. FOREIGN KEY (გარე გასაღები) იქმნება ცხრილის დონეზე. დამოკიდებული ცხრილის გარე გასაღები უკავშირდება მთავარი ცხრილის პირველად გასაღებს. დამოკიდებულ ცხრილში არ შეიძლება სტრიქონის ჩასმა, თუ გარე გასაღებს არ აქვს შესაბამისი მნიშვნელობა მთავარ ცხრილში.

დროებითი ცხრილები

8

დროებითი ცხრილები (**temporary tables**) განკულთვნილია მონაცემების დროებით შესანახად და მოთავსებულია tempdb სისტემურ მონაცემთა ბაზაში. დროებითი ცხრილები ავტომატურად იშლება შეერთების დახურვის ან SQL Server-ის გაჩერების შემთხვევაში. არსებობს ორი სახის დროებითი ცხრილი:

- ▶ **ლოკალური დროებითი ცხრილი (local temporary tables)**, რომლის სახელი ერთი # სიმბოლოთი იწყება (მაგ., #LDC) და ამ სიმბოლოს მითითება საკმარისია ლოკალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია მხოლოდ იმ შეერთების შიგნით, რომელშიც ის იქმნება, ხოლო შეერთების დახურვისას ის ავტომატურად იშლება. თუ ლოკალური დროებითი ცხრილი შეიქმნა შენახული პროცედურის მუშაობისას, მაშინ ამ პროცედურიდან გამოსვლისას ის ავტომატურად წაიშლება.
- ▶ **გლობალური დროებითი ცხრილი (global temporary table)**, რომლის სახელი ## სიმბოლოებით იწყება (მაგ., ##GDC) და ორჯერ ამ სიმბოლოს მითითება საკმარისია გლობალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია ნებისმიერი შეერთებიდან და განკულთვნილია მონაცემების გასაცვლელად სხვადასხვა პროგრამას შორის. გლობალური დროებითი ცხრილი არსებობს იმ შეერთების დასრულებამდე, რომელშიც ის შეიქმნა, ხოლო მისი წაშლა და ცვლილება შესაძლებელია ყველა შეერთებიდან.

სისტემური მონაცემთა ბაზები

9

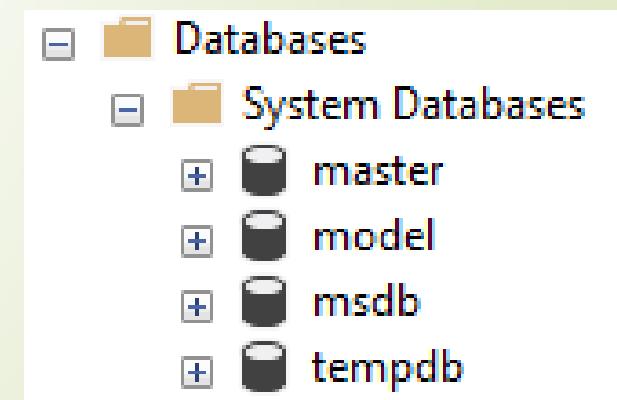
SQL Server-ს გააჩნია ოთხი ტიპის სისტემური მონაცემთა ბაზა:

- ▶ master;
- ▶ model;
- ▶ msdb;
- ▶ tempdb.

ეს მონაცემთა ბაზები გენერირდებიან SQL Server-ის ინსტალირებისას, ინახავენ სისტემურ მონაცემებს და ემსახურებიან შიგა ამოცანების გადაწყვეტას.

როგორც სისტემური, ისე მომხმარებლების მიერ შექმნილი მონაცემთა ბაზები, შეიცავენ სისტემურ ცხრილებსა და წარმოდგენებს. ასევე, ისინი შეიცავენ ინფორმაციას მონაცემთა ბაზის სტრუქტურაზე და მის ორგანიზებულობაზე.

სისტემურ ცხრილებთან მუშაობა UPDATE, INSERT და DELETE ბრძანებების საშუალებით აკრძალულია. დასაშვებია SELECT ბრძანების გამოყენება. ამ ცხრილებში მონაცემების შეცვლა შესაძლებელია მხოლოდ სისტემური შენახული პროცედურების გამოყენებით.



სისტემური მონაცემთა ბაზები

master მონაცემთა ბაზა შეიცავს:

- ▶ სრულ სისტემურ ინფორმაციას SQL Server-ზე;
- ▶ ინფორმაციას SQL Server-ის სხვა მონაცემთა ბაზებზე;
- ▶ ინფორმაციას SQL Server-ის პირველადი ფაილების ადგილმდებარეობის შესახებ.

master მონაცემთა ბაზა შემდეგი ფაილებისაგან შედგება:

- ▶ master.mdf - მონაცემების ფაილი;
- ▶ mastlog.ldf - ტრანზაქციების ჟურნალის ფაილი.

ორივე ფაილი ინახება \Data კატალოგში (C:\Program Files\Microsoft SQL Server\...\MSSQL\Data\).

დაუშვებელია master სისტემურ მონაცემთა ბაზაში მომხმარებლის ობიექტების შექმნა.

სისტემური მონაცემთა ბაზები

SQL Server-ზე ახალი მონაცემთა ბაზის შექმნა ხორციელდება მასში model სისტემური მონაცემთა ბაზის ობიექტების გადაწერის გზით.

model მონაცემთა ბაზა მოთავსებულია \Data კატალოგში და ორი ფაილისგან შედგება:

- ▶ model.mdf - მონაცემების ფაილი;
- ▶ model.ldf - ტრანზაქციების ჟურნალი.

სისტემური მონაცემთა ბაზები

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 SQL Server-ის მართვას.

msdb ბაზა ორი ფაილისგან შედგება:

- ▶ msdbdata.mdf - მონაცემების ფაილი;
- ▶ msdblog.ldf - ტრანზაქციების ჟურნალი.

სისტემური მონაცემთა ბაზები

სისტემური ცხრილები (**system tables**) შეიცავენ SQL Server-ის მუშაობისათვის საჭირო ინფორმაციას. ამიტომ, უფლება არ გვაქვს შეცვალოთ ამ ცხრილებში მოთავსებული მონაცემები.

აქედან გამომდინარე, ამ ცხრილების მიმართ შეგვიძლია მხოლოდ SELECT ბრძანების გამოყენება.

P.S. თუმცა აქვე უნდა აღვნიშნოთ, რომ შენახული პროცედურების გამოყენებით შეგვიძლია განვახორციელოთ მონაცემების ცვლილებები სისტემურ ცხრილებში.

სისტემური მონაცემთა ბაზები

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

tempdb მონაცემთა ბაზა არის გლობალური კურსორი, რომელიც ავტომატურად მისაწვდომია ყველა მომხმარებლისათვის. შესაქმნელი დროებითი ობიექტები შეიძლება იყოს როგორც ლოკალური, ისე გლობალური. ლოკალური ობიექტი მისაწვდომია მხოლოდ მისი შემქმნელი მომხმარებლისთვის, გლობალური. კი - ყველა მომხმარებლისათვის. ლოკალური ობიექტები მოქმედებენ მხოლოდ სეანსის, შენახული პროცედურის, ტრიგერის ან ბრძანებების პაკეტის ფარგლებში. დროებითი ობიექტის შემქმნელი სტრუქტურიდან გამოსვლისას ეს ობიექტი მაშინვე წაიშლება, როგორც კი მომხმარებელი დაამთავრებს SQL Server-თან მუშაობას. ლოკალური დროებითი ცხრილის სახელი # სიმბოლოთი იწყება, გლობალური დროებითი ცხრილის სახელი კი - ## სიმბოლოებით.

tempdb მონაცემთა ბაზა ორი ფაილისაგან შედგება:

- ▶ tempdb.mdf - მონაცემების ფაილი;
- ▶ templog.ldf - ტრანზაქციების ჟურნალი.

სისტემური მონაცემთა ბაზები

დროებითი ცხრილები (**temporary tables**) განკუთვნილია მონაცემების დროებით შესანახად და მოთავსებულია tempdb სისტემურ მონაცემთა ბაზაში. დროებითი ცხრილები ავტომატურად იშლება შეერთების დახურვის ან SQL Server-ის გაჩერების შემთხვევაში. არსებობს ორი სახის დროებითი ცხრილი:

- ▶ **ლოკალური დროებითი ცხრილი (local temporary tables)**, რომლის სახელი ერთი # სიმბოლოთი იწყება (მაგ., #LDC) და ამ სიმბოლოს მითითება საკმარისია ლოკალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია მხოლოდ იმ შეერთების შიგნით, რომელშიც ის იქმნება, ხოლო შეერთების დახურვისას ის ავტომატურად იშლება. თუ ლოკალური დროებითი ცხრილი შეიქმნა შენახული პროცედურის მუშაობისას, მაშინ ამ პროცედურიდან გამოსვლისას ის ავტომატურად წაიშლება.
- ▶ **გლობალური დროებითი ცხრილი (global temporary table)**, რომლის სახელი ## სიმბოლოებით იწყება (მაგ., ##GDC) და ორჯერ ამ სიმბოლოს მითითება საკმარისია გლობალური დროებითი ცხრილის შესაქმნელად. ასეთი ცხრილი ხილულია ნებისმიერი შეერთებიდან და განკუთვნილია მონაცემების გასაცვლელად სხვადასხვა პროგრამას შორის. გლობალური დროებითი ცხრილი არსებობს იმ შეერთების დასრულებამდე, რომელშიც ის შეიქმნა, ხოლო მისი წაშლა და შეცვლა შეგვიძლია ყველა შეერთებიდან.

ფაილები და ფაილების ჯგუფები

მონაცემთა ბაზის შესანახად სამი ტიპის ფაილი გამოიყენება:

- ▶ Primary - ძირითადი ფაილია, რომელიც შეიცავს სისტემურ ინფორმაციას თვით მონაცემთა ბაზისა და მისი ობიექტების შესახებ. მასში მოთავსებულია სისტემური ცხრილები და მონაცემთა ბაზის ობიექტების აღწერა. ძირითად ფაილში შეიძლება, აგრეთვე ინახებოდეს მონაცემებიც. ნებისმიერ მონაცემთა ბაზას აქვს primary ტიპის ფაილი, ამასთან მონაცემთა ბაზაში ამ ტიპის ფაილი მხოლოდ ერთი შეიძლება იყოს. primary ტიპის ფაილებს აქვთ .mdf (Master Data File) გაფართოება.
- ▶ Secondary - მონაცემების არაძირითადი ფაილია, რომელიც არ შეიცავს სისტემურ ინფორმაციას და განკუთვნილია მხოლოდ მონაცემების შესანახად. მონაცემები, რომლებიც არ დაეტია ძირითად ფაილში, მოთავსდება მონაცემების არაძირითადი ფაილებში. მონაცემთა ბაზას შეიძლება საერთოდ არ ჰქონდეს secondary ტიპის ფაილი ან ჰქონდეს რამდენიმე. secondary ტიპის ფაილებს აქვთ .ndf (Not Master Data File) გაფართოება.
- ▶ Transaction Log - ტრანზაქციების ჟურნალის ფაილია. ის გამოიყენება მონაცემთა ბაზაში შესრულებული ტრანზაქციების შესახებ ინფორმაციის შესანახად. ნებისმიერ მონაცემთა ბაზას აქვს ტრანზაქციების ჟურნალის მინიმუმ ერთი ფაილი. transaction log ტიპის ფაილებს აქვთ .ldf (Log Data File) გაფართოება.

ფაილები და ფაილების ჯგუფები

მონაცემთა ბაზაში გამოყენებულ თითოეულ ფაილს ორი სახელი აქვს:

1. ფაილის ლოგიკური სახელი (Logical File Name), რომელიც გამოიყენება Transact-SQL ბრძანებებში კონკრეტულ ფაილთან მიმართვისათვის.
2. ფაილის სახელი ოპერაციულ სისტემაში (OS File Name), რომელიც არის ფაილის ფიზიკური სახელი და ამ სახელით ინახება ფაილი დისკზე.

მონაცემთა ბაზის ყველა ფაილის ზომა შეიძლება ავტომატურად გაიზარდოს, რომელიც შესაძლებელია ფაილის შექმნის დროს განისაზღვროს. ნაზარდის ზომა შეგვიძლია განვსაზღვროთ პროცენტებში (ფაილის საწყისი ზომიდან) ან მეგაბაიტებში. დამატებით, შეგვიძლია მივუთითოთ ფაილის მაქსიმალური ზომა. თუ მაქსიმალური ზომა არ არის მითითებული, მაშინ ფაილის ზომა გაიზრდება მანამ, სანამ არის თავისუფალი სივრცე დისკზე.

ფაილები და ფაილების ჯგუფები

ადმინისტრირებისა და მონაცემთა ბაზის ობიექტების ფიზიკური განლაგების მართვის გაადვილების მიზნით უმჯობესია ფაილები საფაილო ჯგუფებში მოვათავსოთ, რომლებიც არიან სამი ტიპის:

- 1.** ფაილების ძირითადი ჯგუფი (Primary File Group) - შეიცავს primary ტიპის ფაილსა და ყველა ფაილს, რომლებიც არ არის ჩართული სხვა ჯგუფში (შეიძლება მხოლოდ ერთი ძირითადი ჯგუფი);
- 2.** მომხმარებლების მიერ შექმნილი ფაილების ჯგუფი (User-defined File Group) - ჩართულია ყველა ფაილი, რომლებიც მითითებულია FILEGROUP პარამეტრში მონაცემთა ბაზის შექმნის (CREATE DATABASE) ან შეცვლის (ALTER DATABASE) დროს (შეიძლება რამდენიმე ჯგუფი ფაილების ნებისმიერი შემადგენლობით).
- 3.** ფაილების ნაგულისხმევი ჯგუფი (Default File Group) - მონაცემთა ბაზაში ფაილების ერთ-ერთი ჯგუფი ხდება ნაგულისხმევი. თუ მონაცემთა ბაზის შექმნის დროს იგი არ არის მითითებული, მაშინ ფაილების ძირითადი ჯგუფი ხდება ნაგულისხმევი. თუ მონაცემთა ბაზის ობიექტის (ცხრილის ან სვეტის) შექმნის დროს აშკარად არ არის მითითებული, ფაილების რომელ ჯგუფს ეკუთვნის ის, მაშინ ეს ობიექტი შეიქმნება ფაილების ნაგულისხმევ ჯგუფში. მონაცემთა ბაზის მფლობელმა შეიძლება ფაილების ნებისმიერი ჯგუფი დანიშნოს ფაილების ნაგულისხმევ ჯგუფად. ფაილების მხოლოდ ერთი ჯგუფი შეიძლება იყოს ნაგულისხმევი.

მონაცემთა ბაზის შექმნის წინაპირობები

მონაცემთა ბაზის შესაქმნელად უნდა მივუთითოთ მისი სახელი, მფლობელის სახელი, (ეს იქნება მონაცემთა ბაზის შემქმნელი მომხმარებელი), ზომა, ფაილები და ფაილების ჯგუფები, რომლებისგანაც იქნება შემდგარი შესაქმნელი მონაცემთა ბაზა.

მონაცემთა ბაზის შექმნის წინ უნდა გავითვალისწინოთ შემდეგი:

- ▶ მონაცემთა ბაზის შექმნა შეუძლიათ SQL Server-ის **sysadmin** და **dbcreator** ფიქსირებული როლების წევრებს, თუმცა სხვა მომხმარებლებსაც შეიძლება მიეცეს მონაცემთა ბაზების შექმნის უფლება (აქ და ყველგან უნდა ვიგულისხმოთ და გავითვალისწინოთ, რომ SQL Server-ის ადმინისტრატორი არის SQL Server-ის შემქმნელი და, შესაბამისად, ყველა შიგ განთავსებული მონაცემთა ბაზის შემქმნელიც არის და აქედან გამომდინარე მფლობელიც (**db_owner**), ანუ აქვს ყველა მონაცემთა ბაზაში სრული უფლებები);
- ▶ მონაცემთა ბაზის შემქმნელი მომხმარებელი ავტომატურად ხდება მისი მფლობელი;
- ▶ მონაცემთა ბაზის სახელი უნდა აკმაყოფილებდეს ობიექტების სახელდების მოთხოვნებს (სიგრძე, სიმბოლოები და სხვ.).

მონაცემთა ბაზის შექმნის დაგეგმვა

მონაცემთა ბაზის შექმნის ეტაპს წინ უძლვის ფიზიკური და ლოგიკური ნაწილების დაგეგმვის ფაზა:

- ▶ ფიზიკური ნაწილის დაგეგმვა, რომელიც მოიცავს ფაილებისა და ფაილების ჯგუფების შემადგენლობის დაპროექტებას, რომლისგანაც იქნება შემდგარი მონაცემთა ბაზა;
- ▶ ლოგიკური სტრუქტურის დაგეგმვა, რომელშიც შედის ცხრილების დაპროექტება ნორმალიზების გათვალისწინებით.

მონაცემთა ბაზის შექმნისას **master** მონაცემთა ბაზის **sys.databases** სისტემურ წარმოდგენაში შეიტანება შესაბამისი მონაცემები, რომელთა ნახვა შეგვიძლია შემდეგი მოთხოვნის გამოყენებით:

```
SELECT *
FROM sys.databases;
```

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ბაზის შექმნის დაგეგმვა

21

SQLQuery1.sql - D...INQ5DSH\USER (60)* ✎ X

```
1 SELECT *
2 FROM sys.databases
```

100 %

Results Messages

	name	database_id	source_database_id	owner_sid	create_date	compatibility_level	collation_name	user_access
1	master	1	NULL	0x01	2003-04-08 09:13:36.390	140	SQL_Latin1_General_CI_AS	0
2	tempdb	2	NULL	0x01	2019-03-09 11:54:21.260	140	SQL_Latin1_General_CI_AS	0
3	model	3	NULL	0x01	2003-04-08 09:13:36.390	140	SQL_Latin1_General_CI_AS	0
4	msdb	4	NULL	0x01	2017-08-22 19:39:22.887	130	SQL_Latin1_General_CI_AS	0
5	LMAIT	5	NULL	0x010500000...	2018-05-05 23:23:00.613	120	SQL_Latin1_General_CI_AS	0
6	SMAES	6	NULL	0x010500000...	2018-05-05 23:30:13.047	120	SQL_Latin1_General_CI_AS	0
7	AdventureWorks2017	7	NULL	0x01	2019-02-14 23:39:54.010	140	SQL_Latin1_General_CI_AS	0
8	GPeikrishvili3	8	NULL	0x010500000...	2019-02-16 11:47:27.977	140	SQL_Latin1_General_CI_AS	0
9	Wignebi	9	NULL	0x01	2019-02-16 12:28:07.773	120	SQL_Latin1_General_CI_AS	0
10	ilia_darchashvili_SQL	10	NULL	0x01	2019-02-16 13:03:05.220	140	SQL_Latin1_General_CI_AS	0
11	aftiaqi(108651)(m.a)	11	NULL	0x010500000...	2019-02-16 13:41:12.837	140	SQL_Latin1_General_CI_AS	0
12	G.G51	12	NULL	0x010500000...	2019-02-16 13:57:28.943	140	SQL_Latin1_General_CI_AS	0
13	Restaurant2	13	NULL	0x010500000...	2019-02-16 14:20:56.233	140	SQL_Latin1_General_CI_AS	0
14	proeqtti	14	NULL	0x01	2019-02-23 15:00:18.087	120	SQL_Latin1_General_CI_AS	0
15	gyatashvili1	15	NULL	0x010500000...	2019-02-23 16:01:07.950	140	SQL_Latin1_General_CI_AS	0
16	avtosaloni	16	NULL	0x01	2019-02-24 13:01:13.243	140	SQL_Latin1_General_CI_AS	0
17	musikaluri studia main	17	NULL	0x010500000...	2019-02-23 21:01:17.350	140	SQL_Latin1_General_CI_AS	0
18	samsheneblo_kompa...	18	NULL	0x010500000...	2019-02-23 21:36:28.013	140	SQL_Latin1_General_CI_AS	0
19	MGelashvili51	19	NULL	0x010500000...	2019-02-23 21:46:56.327	140	SQL_Latin1_General_CI_AS	0
20	db_rezo	20	NULL	0x01	2019-02-23 22:22:21.730	120	Cyrillic_General_CI_AS	0
21	avejis_saxli	21	NULL	0x01	2019-02-23 22:28:11.047	130	SQL_Latin1_General_CI_AS	0

Query executed successfully.

DESKTOP-INQ5DSH\NA (14.0 RTM) | DESKTOP-INQ5DSH\USER (60) | master | 00:00:00 | 45 rows

მონაცემთა ბაზის შექმნა

მონაცემთა ბაზის შესაქმნელად გამოიყენება **CREATE DATABASE** ბრძანება. მისი სინტაქსია:

CREATE DATABASE მონაცემთა_ბაზის_სახელი

[**CONTAINMENT** = { **NONE** | **PARTIAL** }]

[**ON**

[**PRIMARY**] [<ფაილის_განსაზღვრა> [,...n]]

[, <ფაილების_ჯგუფი> [,...n]]]

[**LOG ON** { <ფაილის_განსაზღვრა> [,...n] }]

]

[**COLLATE** სორტირების_სახელი]

[**WITH** <ოფცია> [,...n]]]

[:]

განვიხილოთ არგუმენტების დანიშნულება:

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ბაზის შექმნა

23

- ▶ **მონაცემთა_ბაზის_სახელი** - შესაქმნელი მონაცემთა ბაზის სახელია. თუ შეიცავს ინტერვალებს ან სხვა დაუშვებელ სიმბოლოებს, მაშინ სახელი უნდა მოვათავსოთ კვადრატული ფრჩხილები. მონაცემთა ბაზის სახელი უნდა იყოს უნიკალური SQL Server-ის ფარგლებში. ის არ უნდა აღემატებოდეს 128 სიმბოლოს.
თუ „ფაილის_განსაზღვრა“ არ არის მითითებული SQL Server-ი „მონაცემთა_ბაზის_სახელს“ იღებს „ფაილის_ლოგიკურ_სახელადაც“ და „ფაილის_ფიზიკურ_სახელადაც“. ფიზიკური ფაილის განსანთავსებლად SQL Server-ი იღებს სტანდარტულ გზას რეესტრიდან.
თუ ტრანზაქციების ჟურნალის სახელი მითითებული არ არის, მაშინ SQL Server-ი ტრანზაქციების ჟურნალის სახელად იღებს „მონაცემთა_ბაზის_სახელს“ და მას უმატებს _Log სიმბოლოებს. ამ შემთხვევაში, მონაცემთა ბაზის სახელი არ უნდა აღემატებოდეს 123 სიმბოლოს;
- ▶ **CONTAINMENT = { NONE | PARTIAL }** - მიუთითებს მონაცემთა ბაზის ჩართულობას:
NONE - არა ავტონომიურ მონაცემთა ბაზას, ხოლო **PARTIAL** - ნაწილობრივ ავტონომიურ მონაცემთა ბაზას (უფრო დაწვრილებით იხილება ადმინისტრირების შესწავლისას);
- ▶ **ON** - მიუთითებს მონაცემთა ბაზის ფიზიკური ფაილების აშკარად განსაზღვრის დაწყებას, რომელსაც აუცილებლად უნდა მოსდევდეს პირველადი <ფაილის_განსაზღვრა>;

მონაცემთა ბაზის შექმნა

- ▶ **PRIMARY** - მიუთითებს მონაცემთა ბაზის პირველადი ფაილის აღწერის დაწყებას. მონაცემთა ბაზაში მხოლოდ ერთი ფაილი შეიძლება იყოს განსაზღვრული როგორც პირველადი. თუ პირველადი ფაილი აშკარად არ არის განსაზღვრული, მაშინ პირველად ფაილად გამოყენებული იქნება <ფაილის_განსაზღვრა> კონსტრუქციაში მითითებული პირველი ფაილი;
- ▶ <ფაილის_განსაზღვრა> ::=
 ([**NAME** = 'ფაილის_ლოგიკური_სახელი',]
FILENAME = 'ფაილის_ფიზიკური_სახელი'
 [, **SIZE** = 'ზომა' [KB | MB | GB | TB]]
 [, **MAXSIZE** = {'მაქსიმალური_ზომა' [KB | MB | GB | TB] | **UNLIMITED** }]
 [, **FILEGROWTH** = 'ნაზარდი' [KB | MB | GB | TB | %]]).
 ამ არგუმენტში:
 - ▶ **NAME** = 'ფაილის_ლოგიკური_სახელი' - ფაილის ლოგიკური სახელია, რომელიც უნდა იყოს უნიკალური SQL Server-ის ფარგლებში;
 - ▶ **FILENAME** = 'ფაილის_ფიზიკური_სახელი' - შეიცავს ფაილის ფიზიკურ სახელს ან ფაილის ფიზიკურ სახელს და გზას. იგი შექმნილი იქნება დისკზე და ეს სახელი ფაილს ექნება ოპერაციული სისტემის დონეზე;

მონაცემთა ბაზის შექმნა

25

- ▶ **SIZE** = 'ზომა' [KB | MB | GB | TB] - მიუთითებს მონაცემთა ბაზის ფიზიკური ფაილის საწყის ზომას შესაბამის განზომილებაში (თუ არ არის მითითებული, მაშინ მისი ნაგულისხმევი მნიშვნელობა იქნება 8 MB). ფაილის ზომა უნდა იყოს მთელი რიცხვი;
- ▶ **MAXSIZE** = {'მაქსიმალური_ზომა' [KB | MB | GB | TB] | UNLIMITED } - მიუთითებს მონაცემთა ბაზის ფიზიკური ფაილის მაქსიმალურ ზომას. ერთი ფაილის ზომის შეზღუდვა არ ზღუდავს მონაცემთა ბაზის ზომის ზრდას, რადგან მონაცემთა ბაზის ზომა შეიძლება გაიზარდოს სხვა ფაილების ზომის ზრდის ხარჯზე. თუ ფაილის ზომა არ უნდა შეიზღუდოს, მაშინ უნდა მივუთითოთ UNLIMITED მნიშვნელობა (ნაგულისხმევი მნიშვნელობა) ან საერთოდ გამოვტოვოთ MAXSIZE არგუმენტი. აქვე უნდა აღინიშნოს, რომ SQL Server-ში მონაცემთა ბაზის ფიზიკური ფაილის მაქსიმალური ზომაა 16 TB, ხოლო ტრანზაქციების ჟურნალის 2 TB;
- ▶ **FILEGROWTH** = 'ნაზარდი' [KB | MB | GB | TB | %] - მიუთითებს მონაცემთა ბაზის ფიზიკური ფაილის ზომის ავტომატურად გაზრდისას ნაზარდ ზომას. თუ არ არის მითითებული, მაშინ მისი ნაგულისხმევი მნიშვნელობა იქნება 8 MB. ნაზარდის ზომა ასევე შესაძლებელია მითითებულ იქნეს პროცენტებში ფაილის ზომიდან (ამ შემთხვევაში ნაზარდი ზომა მრგვალდება 64 KB-მდე). თუ ნაზარდი ზომა მითითებულია „0“, მონაცემთა ბაზა ზრდა გათიშულია და ფაილის ზომა არ იზრდება. ფაილის საწყისი ზომა და ნაზარდის ზომა ჯამურად არ უნდა აღემატებოდეს ფაილის მაქსიმალურ ზომას (MAXSIZE). SQL Server-ი საშუალებას გვაძლევს ვაკონტროლოთ მონაცემთა ბაზის თითოეული ფაილის ზომის ზრდა სხვა ფაილებისაგან დამოუკიდებლად.

მონაცემთა ბაზის შექმნა

- ▶ <ფაილების_ჯგუფი> ::=
(FILEGROUP ‘ფაილების_ჯგუფის_სახელი’
[[CONTAINS FILESTREAM] [DEFAULT] | CONTAINS MEMORY_OPTIMIZED_DATA]
<ფაილის_განსაზღვრა> [....n]).
 ამ არგუმენტში:
 - ▶ **FILEGROUP** ‘ფაილების_ჯგუფის_სახელი’ - ფაილების ჯგუფის ლოგიკური სახელია, რომელიც უნდა იყოს უნიკალური მონაცემთა ბაზის ფარგლებში. მას არ შეუძლია მიიღოს მნიშვნელობა PRIMARY ან PRIMARY_LOG;
 - ▶ **CONTAINS FILESTREAM** - უთითებს, რომ ფაილების ჯგუფი ინახავს FILESTREAM დიდ ორობით ობიექტებს (BLOB-ობიექტებს) ფაილურ სისტემაში;
 - ▶ **CONTAINS MEMORY_OPTIMIZED_DATA** - უთითებს, რომ ფაილების ჯგუფი ინახავს memory_optimized მონაცემებს ფაილურ სისტემაში;
 - ▶ **ფაილის_განსაზღვრა** - უკვე განხილულია წინა სლაიდებში (ამ შემთხვევაში FILESTREAM-თან ერთად არ ეთითება SIZE, MAXSIZE და FILEGROWTH).
- ▶ **LOG ON** - განსაზღვრავს ტრანზაქციების ჟურნალის ფაილებს აშკარად. თუ ეს არგუმენტი არ არის მითითებული, მაშინ SQL Server-ი ავტომატურად ქმნის ტრანზაქციების ჟურნალის ერთ ფაილს, რომლის სახელი იქმნება მონაცემთა ბაზის სახელისათვის _Log სიმბოლოების დამატების გზით;

მონაცემთა ბაზის შექმნა

27

- ▶ **COLLATE** სორტირების_სახელი - განსაზღვრავს მონაცემთა ბაზისათვის სორტირების პარამეტრებს. თუ ეს პარამეტრი მითითებული არა არის, მონაცემთა ბაზისათვის ავტომატურად განისაზღვრება SQL Server-ის სორტირების პარამეტრები;
- ▶ **<ოფცია> ::=**
{
 FILESTREAM (<filestream_ოფცია> [,...n])
 | **DEFAULT_FULLTEXT_LANGUAGE = { Icid | ენის_სახელი | ენის_ალიასი }**
 | **DEFAULT_LANGUAGE = { Icid | ენის_სახელი | ენის_ალიასი }**
 | **NESTED_TRIGGERS = { OFF | ON }**
 | **TRANSFORM_NOISE_WORDS = { OFF | ON }**
 | **TWO_DIGIT_YEAR_CUTOFF = <two_digit_year_cutoff>**
 | **DB_CHAINING { OFF | ON }**
 | **TRUSTWORTHY { OFF | ON }**
 | **PERSISTENT_LOG_BUFFER=ON (DIRECTORY_NAME=' საქაღალდეს_სახელი ')**
}

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ბაზის შექმნა

განვიხილოთ პარამეტრები:

- ▶ <filestream_ოფცია> ::=
NON_TRANSACTED_ACCESS = { OFF | READ_ONLY | FULL }
DIRECTORY_NAME = <საქაღალდეს_სახელი>
- ამ არგუმენტში:
 - ▶ **NON_TRANSACTED_ACCESS = { OFF | READ_ONLY | FULL }** - მიუთითებს მონაცემთა ბაზისათვის FILESTREAM არატრანზაქციული წვდომის დონეს, სადაც
 - ▶ **OFF** - არატრანზაქციული წვდომა გათიშულია;
 - ▶ **READ_ONLY** - ამ მონაცემთა ბაზაში FILESTREAM მონაცემები შეიძლება წაკითხულ იქნეს არატრანზაქციული პროცესებით;
 - ▶ **FULL** – სრული არატრანზაქციული წვდომა FILESTREAM FileTable ჩართულია.
 - ▶ **DIRECTORY_NAME = <საქაღალდეს_სახელი>** - მიუთითებს Windows-თან შეთავსებულ SQL Server-ის Database_Directory-ს ეგზემპლიარის საქაღალდეს რეგისტრის გათვალისწინებით უნიკალურ სახელს. ამ პარამეტრის მითითება აუცილებელია ამ მონაცემთა ბაზაში FILESTREAM-ის შექმნამდე.

შემდეგი პარამეტრები დასაშვებია მხოლოდ იმ შემთხვევაში, თუ CONTAINMENT-ში მითითებულია PARTIAL:

- ▶ **DEFAULT_FULLTEXT_LANGUAGE = { Icid | ენის_სახელი | ენის_ალიასი }** - განსაზღვრავს შეთანხმებით (სტანდარტულ) ენას სრულტექსტიანი ინდექსისათვის;

(გაგრძელება შემდეგ სლაიდზე)

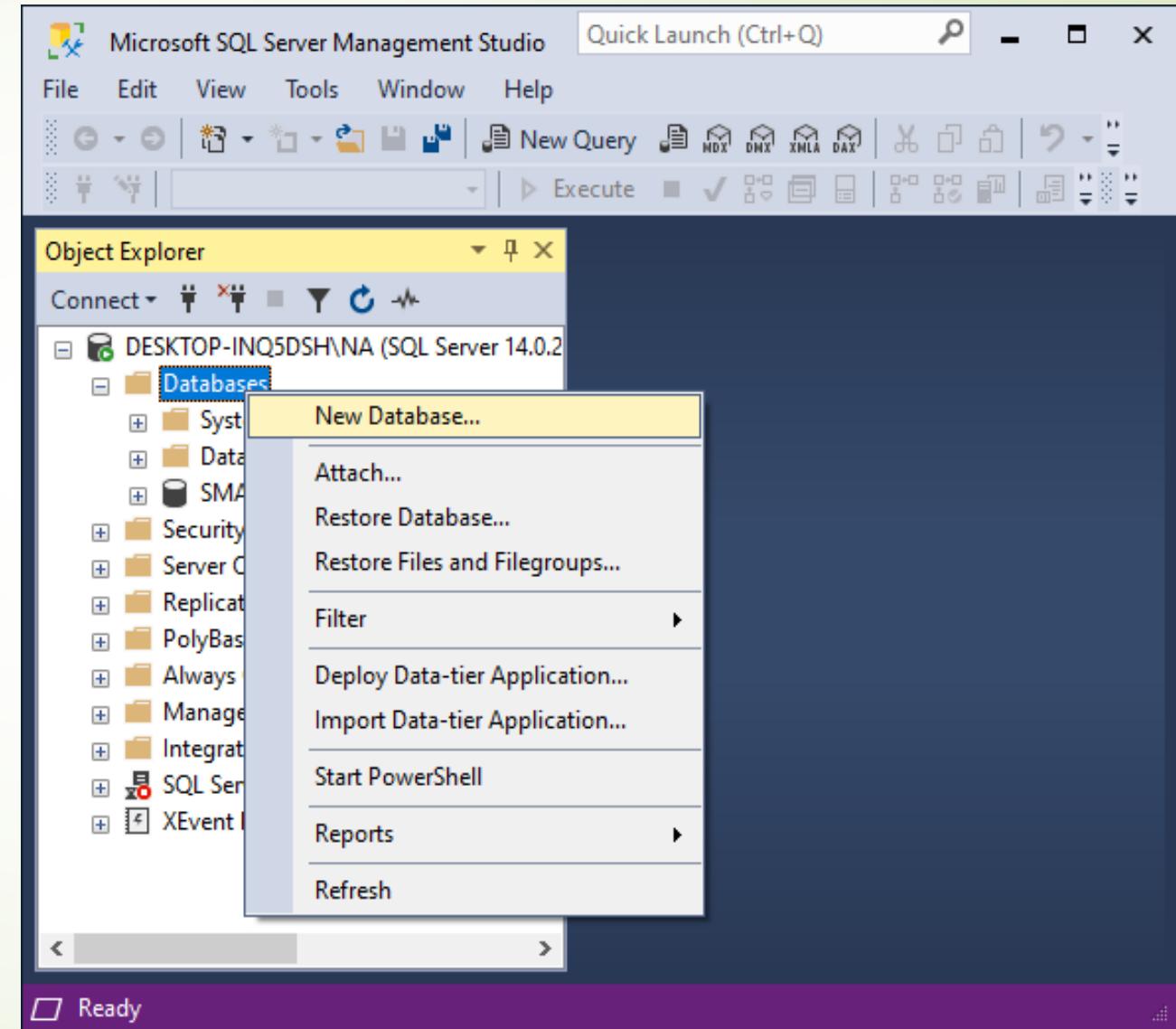
მონაცემთა ბაზის შექმნა

- ▶ **DEFAULT_LANGUAGE = { Icid | ენის_სახელი | ენის_ალიასი }** - განსაზღვრავს შეთანხმებით (სტანდარტულ) ენას ყველა ახლად შექმნილ წვდომის სახელებისათვის;
- ▶ **NESTED_TRIGGERS = { OFF | ON }** - განსაზღვრავს AFTER ტრიგერებში ჩადგმულობას (ჩართვის შემთხვევაში 32 დონე);
- ▶ **TRANSFORM_NOISE_WORDS = { OFF | ON }** - გამოტოვებულ სიტყვებისათვის სრულტექსტიან მოთხოვნაში ლოგიკური ოპერაცია აბრუნებს 0 სტრიქონს. გამოიყენება პრედიკატ CONTAINS-ში, რომელშიც ოპერაცია NEAR შეიცავს გამოტოვებულ სიტყვებს (OFF მნიშვნელობისას აბრუნებს 0-ს და გამოაქვს გაფრთხილების ფანჯარა, ხოლო ON-ის დროს გამოტოვებული სიტყვები იგნორირება და მოთხოვნა ისე მუშავდება);
- ▶ **TWO_DIGIT_YEAR_CUTOFF = <two_digit_year_cutoff>** - განსაზღვრავს ორი ციფრით გამოსახული წლის საბოლოო მნიშვნელობას (მაგ. თუ მითითებულია 2049, მაშინ 49 არის 2049 წელი, ხოლო 50 - 1950);
- ▶ **DB_CHAINING { OFF | ON }** - განსაზღვრავს ბაზათა შორის მფლობელობის ჯაჭვში არის თუ არა წყარო ან მიზნობრივი მონაცემთა ბაზა;
- ▶ **TRUSTWORTHY { OFF | ON }** - განსაზღვრავს შეუძლია თუ არა მონაცემთა ბაზის მოდულებმა (წარმოდგენებმა, ფუნქციებმა, პროცედურებმა) მიმართონ ბაზის გარეთ განთავსებულ რესურსებს;
- ▶ **PERSISTENT_LOG_BUFFER=ON (DIRECTORY_NAME=' საქაღალდეს_სახელი ')** - თუ საქაღალდე მითითებულია, ამ საქაღალდეში შეიქმნება ტრანზაქციების ჟურნალის ბუფერი.

მონაცემთა ბაზის შექმნა

30

SQL Server Management Studio-ში (SSMS) მონაცემთა ბაზის შექმნისათვის მომხმარებელმა უნდა გააკტიუროს მთავარი მენიუს „მონაცემთა ბაზები“-ს (Databases) სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს პირველივე სტრიქონი „ახალი მონაცემთა ბაზა...“ (New Database...).



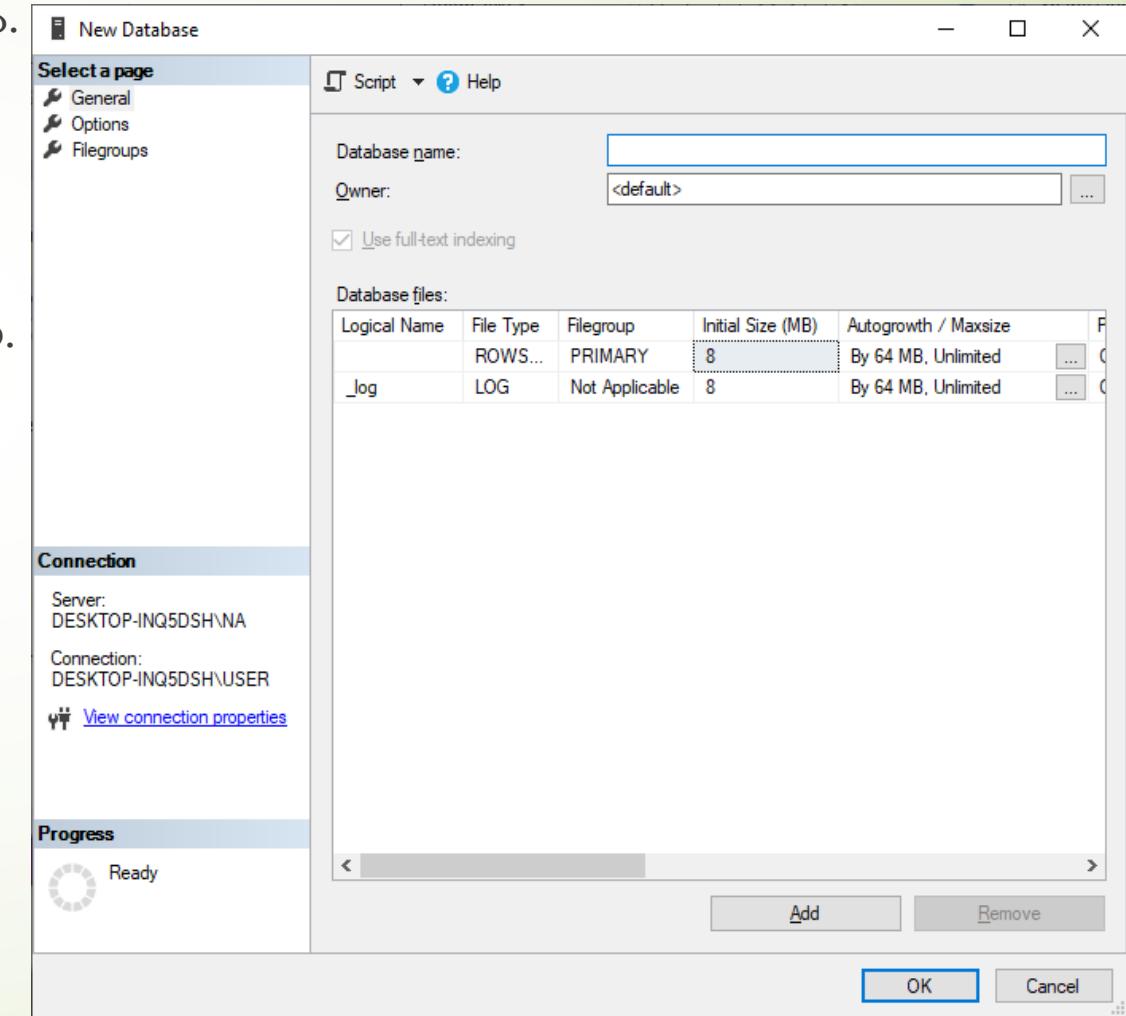
მონაცემთა ბაზის შექმნა

31

სტრიქონის გააქტიურებისას მონიტორის ეკრანზე გამოისახება ახალი მონაცემთა ბაზის შექმნის შესაბამისი ფანჯარა, რომელიც ზედა-მარცხენა მიდამოში განლაგებული მთავარი მენიუს სამი სტრიქონით ფორმატირდება.

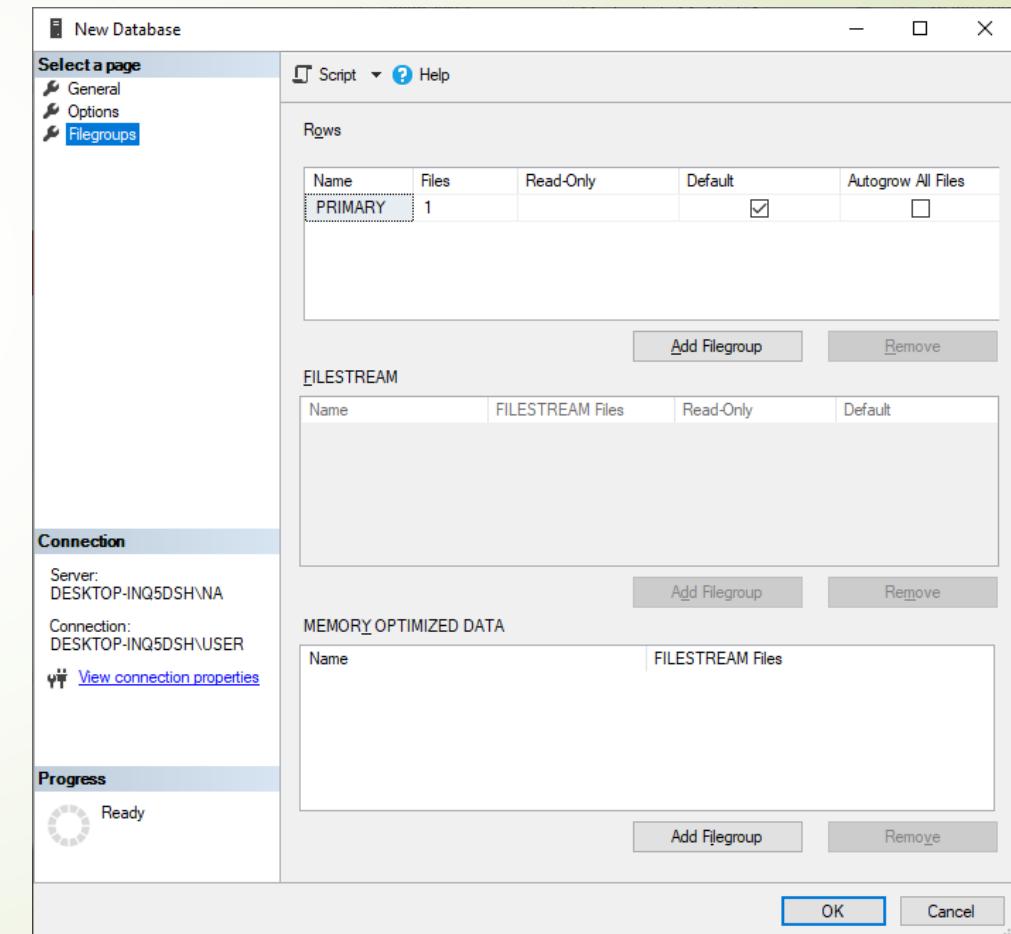
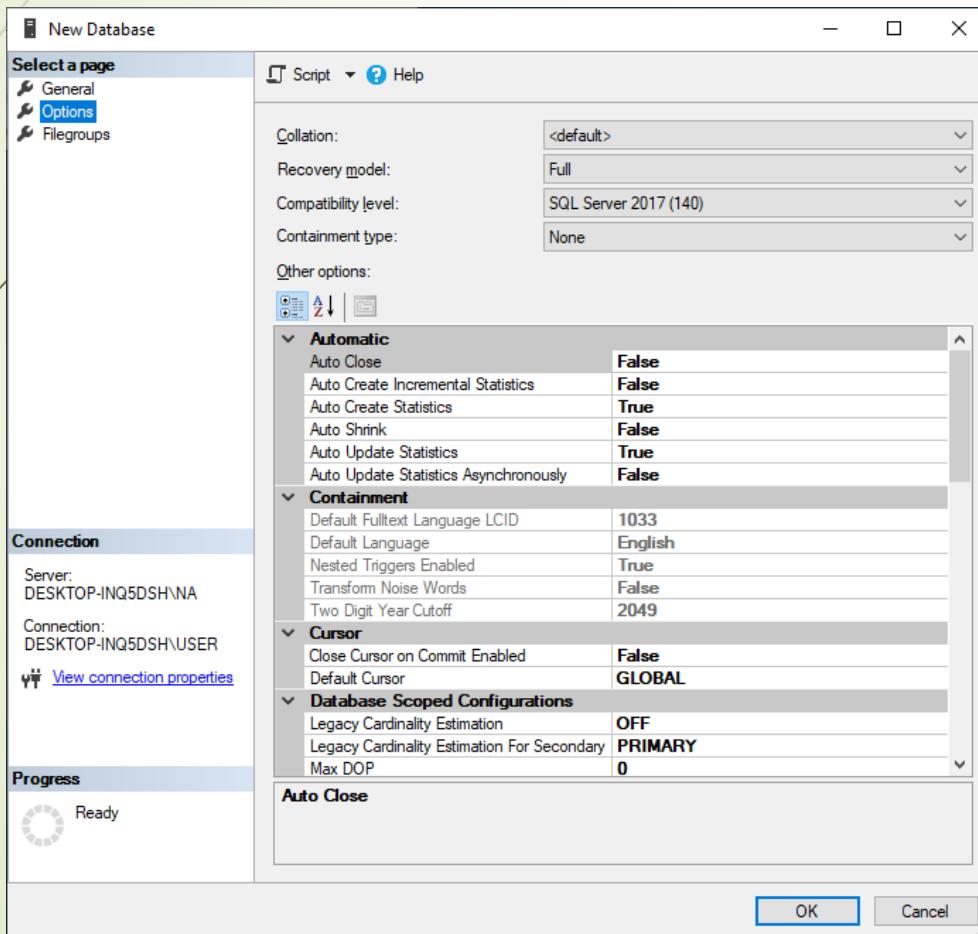
პირველი სტრიქონის „ძირითადი“ (General) გამოყენებით მომხმარებელს ეძღვა შესაძლებლობა ზედა-მარჯვენა მიდამოში მიუთითოს შესაქმნელი მონაცემთა ბაზის სახელი და მფლობელი. თუ მფლობელი არ იქნა მითითებული, სისტემა ავტომატურად შეარჩევს იმ მომხმარებელს, რომელიც იმ მომენტში ახორციელებს მონაცემთა ბაზის შექმნას.

ასევე ქვედა-მარჯვენა მიდამოში გამოსახულია შესაქმნელი მონაცემთა ბაზის ატრიბუტები (სახელები, ზომები და სხვა), რომლის სურვილისებრ ცვლილების შესაძლებლობა გააჩნია მომხმარებელს.



მონაცემთა ბაზის შექმნა

ზედა-მარცხენა მიდამოში განლაგებული მთავარი მენიუს მეორე და მესამე სტრიქონის გააქტიურებით მომხმარებელს ეძღვავა შესაძლებლობა ნახოს/დააფორმატიროს შესაქმნელ მონაცემთა ბაზის ძირითადი ოფციები და ფაილთა ჯგუფები.



მონაცემთა ბაზის შექმნა

1

მაგალითი 1: შევქმნათ **BAZA1** მონაცემთა ბაზა. თუ SQL Server-ზე მონაცემთა ბაზა **BAZA1** უკვე არსებობს, მაშინ ის ჯერ უნდა წაიშალოს და შემდეგ ხელახლა შეიქმნას ნაგულისხმევი მნიშვნელობებით (T-SQL ენაში ბრძანების სტრიქონში ორი დეფისის '--' შემდეგ მოთავსებული ინფორმაცია გამოიყენება კომენტარების ჩასაწერად და ბრძანების განხორციელებაში მონაწილეობას არ ღებულობს):

```
USE Master; -- სისტემური მონაცემთა ბაზის გახსნა
-- SQL Server-ზე BAZA1 მონაცემთა ბაზის არსებობის შემოწმება და არსებობისას წაშლა
IF DB_ID ('N'BAZA1') IS NOT NULL
DROP DATABASE BAZA1;
-- BAZA1 ბაზის შექმნა არგუმენტების გარეშე (შეთანხმებით არგუმენტებით)
CREATE DATABASE BAZA1;
```

The screenshot shows a SQL query window titled "SQLQuery1.sql - D...INQ5DSH\USER (60)*". The code is identical to the one above, designed to drop an existing database if it exists and then create a new one. The code is color-coded: green for comments, blue for keywords like USE, IF, DROP, CREATE, and DB_ID, and black for identifiers like Master, BAZA1, and N'BAZA1'. The status bar at the bottom indicates "Commands completed successfully."

```
SQLQuery1.sql - D...INQ5DSH\USER (60)*
1  -- სისტემური მონაცემთა ბაზის გახსნა
2  USE Master;
3  -- სერვერზე BAZA1 მონაცემთა ბაზის არსებობის შემოწმება და არსებობისას წაშლა
4  IF DB_ID ('N'BAZA1') IS NOT NULL
5  DROP DATABASE BAZA1; -- მონაცემთა ბაზის BAZA1 წაშლა
6  -- BAZA1 ბაზის შექმნა არგუმენტების გარეშე (შეთანხმებით არგუმენტებით)
7  CREATE DATABASE BAZA1;
8
```

100 %

Messages

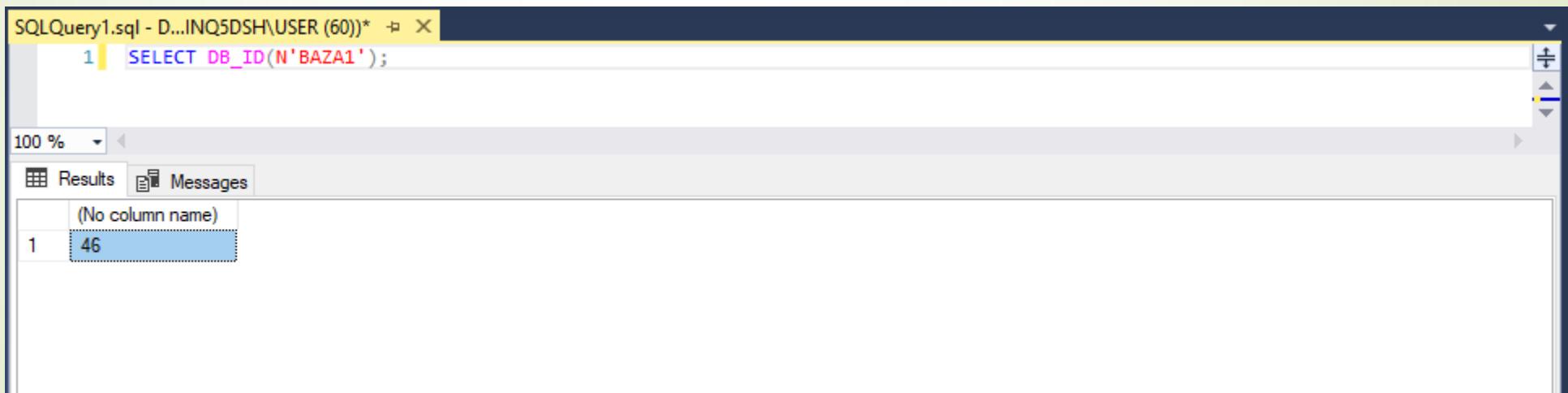
Commands completed successfully.

მონაცემთა ბაზის შექმნა

2

მაგალითში გამოყენებულია **DB_ID()** ფუნქცია (ფრჩხილებში იწერება მონაცემთა ბაზის სახელი), რომელიც გასცემს მონაცემთა ბაზის საიდენტიფიკაციო ნომერს. თუ ეს ნომერი არ არის ცარიელი, მაშინ შესაბამისი მონაცემთა ბაზა არსებობს და ის წაიშლება **DROP** ბრძანების გამოყენებით. ზოგადად მონაცემთა ბაზის იდენტიფიკატორის (მონაცემთა ბაზის საიდენტიფიკაციო ნომრების სია ინახება **master** სისტემური მონაცემთა ბაზის **sysdatabases** წარმოდგენაში) სანახავად გამოიყენება ბრძანება:

```
SELECT DB_ID(N'BAZA1');
```



```
SQLQuery1.sql - D...INQ5DSH\USER (60)* X
1 | SELECT DB_ID(N'BAZA1');

100 % < >
Results Messages
(No column name)
1 46
```

მონაცემთა ბაზის შექმნა

3

მაგალითი 2: შევქმნათ Baza2 მონაცემთა ბაზა ფაილების ჯგუფების მითითების გზით. Baza2 შედგება პირველადი ჯგუფისაგან, რომელიც შეიცავს File1_dat და File2_dat ფაილებს; Group1 ჯგუფისგან, რომელიც შეიცავს G1F1_dat და G1F2_dat ფაილებს; Group2 ჯგუფისგან, რომელიც შეიცავს G2F1 და G2F2 ფაილებს.

```
USE master; -- თუ Baza2 ბაზა არსებობს, მაშინ ის წაიშლება
IF DB_ID (N'Baza2') IS NOT NULL
DROP DATABASE Baza2; -- მონაცემთა ბაზის Baza2 წაშლა
-- იმ კატალოგისკენ გზის განსაზღვრა, რომელშიც იქმნება Baza2 მონაცემთა ბაზა
DECLARE @data_path NVARCHAR(256);
SET @data_path =
(
SELECT SUBSTRING(physical_name, 1, CHARINDEX(N'master.mdf',
LOWER(physical_name)) - 1)
FROM master.sys.master_files
WHERE database_id = 1 AND file_id = 1
);
-- გაგრძელება შემდეგ სლაიდზე
```

მონაცემთა ბაზის შექმნა

4

```
-- Baza2 მონაცემთა ბაზის შექმნა
EXEC ('CREATE DATABASE Baza2
ON PRIMARY
-- პირველადი ჯგუფის პირველი ფაილის აღწერა
(
NAME = File1_dat,
FILENAME = '"+ @data_path + 'File1dat.mdf',
SIZE = 20MB,
MAXSIZE = 50MB,
FILEGROWTH = 15%
),
-- პირველადი ჯგუფის მეორე ფაილის აღწერა
(
NAME = File2_dat,
FILENAME = '"+ @data_path + 'File2dat.ndf',
SIZE = 20MB,
MAXSIZE = 50MB,
FILEGROWTH = 15%
),
-- გაგრძელება შემდეგ სლაიდზე
```

მონაცემთა ბაზის შექმნა

5

```
-- პირველი ჯგუფის აღწერა  
FILEGROUP Group1  
-- პირველი ჯგუფის პირველი ფაილის აღწერა  
(  
    NAME = G1F1_dat,  
    FILENAME = "'"+ @data_path + 'G1F1dat.ndf",  
    SIZE = 20MB,  
    MAXSIZE = 50MB,  
    FILEGROWTH = 5MB  
,  
-- პირველი ჯგუფის მეორე ფაილის აღწერა  
(  
    NAME = G1F2_dat,  
    FILENAME = "'"+ @data_path + 'G1F2dat.ndf",  
    SIZE = 20MB,  
    MAXSIZE = 50MB,  
    FILEGROWTH = 5MB  
,  
-- გაგრძელება შემდეგ სლაიდზე
```

მონაცემთა ბაზის შექმნა

6

```
-- მეორე ჯგუფის აღწერა  
FILEGROUP Group2  
-- მეორე ჯგუფის პირველი ფაილის აღწერა  
(  
    NAME = G2F1_dat,  
    FILENAME = "'"+ @data_path + 'G2F1dat.ndf",  
    SIZE = 20MB,  
    MAXSIZE = 50MB,  
    FILEGROWTH = 5MB  
,  
-- მეორე ჯგუფის მეორე ფაილის აღწერა  
(  
    NAME = G2F2_dat,  
    FILENAME = "'"+ @data_path + 'G2F2dat.ndf",  
    SIZE = 20MB,  
    MAXSIZE = 50MB,  
    FILEGROWTH = 5MB  
)  
-- გაგრძელება შემდეგ სლაიდზე
```

მონაცემთა ბაზის შექმნა

7

```
-- ლოგიკური ფაილის შექმნა  
LOG ON  
(  
    NAME = Baza2_log,  
    FILENAME = "'"+ @data_path + 'Baza2_log.Idf",  
    SIZE = 10MB,  
    MAXSIZE = 30MB,  
    FILEGROWTH = 5MB  
)  
);
```

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ბაზის შექმნა

8

The screenshot shows a SQL query window titled "SQLQuery2.sql - D...INQ5DSH\USER (54)*". The code creates a new database "Baza2" by first dropping it if it exists, then setting the data path to the master database's physical name, and finally executing a CREATE DATABASE command. The log file is also created at the same path.

```
1 USE master;
2 IF DB_ID ('N'Baza2') IS NOT NULL
3     DROP DATABASE Baza2;
4 DECLARE @data_path NVARCHAR(256);
5 SET @data_path =
6 (
7     SELECT SUBSTRING(physical_name, 1, CHARINDEX(N'master.mdf', LOWER(physical_name)) - 1)
8         FROM master.sys.master_files
9         WHERE database_id = 1 AND file_id = 1 );
10 EXEC ('CREATE DATABASE Baza2
11     ON PRIMARY
12         ( NAME = File1_dat, FILENAME = ''' + @data_path + 'File1dat.mdf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 15% ),
13         ( NAME = File2_dat, FILENAME = ''' + @data_path + 'File2dat.ndf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 15% ),
14     FILEGROUP Group1
15         ( NAME = G1F1_dat, FILENAME = ''' + @data_path + 'G1F1dat.ndf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 5MB ),
16         ( NAME = G1F2_dat, FILENAME = ''' + @data_path + 'G1F2dat.ndf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 5MB ),
17     FILEGROUP Group2
18         ( NAME = G2F1_dat, FILENAME = ''' + @data_path + 'G2F1dat.ndf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 5MB ),
19         ( NAME = G2F2_dat, FILENAME = ''' + @data_path + 'G2F2dat.ndf'', SIZE = 20MB, MAXSIZE = 50MB, FILEGROWTH = 5MB )
20     LOG ON
21         ( NAME = Baza2_log, FILENAME = ''' + @data_path + 'Baza2_log.ldf'', SIZE = 10MB, MAXSIZE = 30MB, FILEGROWTH = 5MB )');
```

Messages

Commands completed successfully.

მონაცემთა ბაზის მიერთება

9

მონაცემთა ბაზის მიერთება გამოიყენება მაშინ, როცა უნდა შესრულდეს SQL Server-თან მონაცემთა ბაზის მიერთება (**FOR ATTACH**). ამ შემთხვევაში უნდა არსებობდეს შესაბამისი ფაილები. მონაცემთა ბაზის მისაერთებლად საკმარისია მხოლოდ მონაცემთა ბაზის პირველადი ფაილის ადგილმდებარეობის მითითება, რომლებზეც მომხმარებელს უნდა გააჩნდეს წვდომა (MDF და NDF-ზე არსებობის შემთხვევაში). ინფორმაცია მონაცემთა ბაზის სხვა ფაილების (მეორადი და ტრანზაქციების ჟურნალის) ადგილმდებარეობის შესახებ ინახება მონაცემთა ბაზის პირველად ფაილში. FOR ATTACH პარამეტრი ვერ მიეთითება მონაცემთა ბაზის მომენტალური სურათისთვის.

მონაცემთა ბაზის მისაერთებლად გამოიყენება **CREATE DATABASE** ბრძანება შემდეგი სინტაქსით:

CREATE DATABASE მონაცემთა_ბაზის_სახელი

ON <ფაილის_განსაზღვრა> [,...n]

FOR { { **ATTACH** [WITH < მისაერთებელი_მონაცემთა_ბაზის_ოფცია > [,...n]] }
| **ATTACH_REBUILD_LOG** }

განვიხილოთ არგუმენტის დანიშნულება (დანარჩენი უკვე განხილულია წინა მასალებში):

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ბაზის მიერთება

10

< მისაერთებელი_მონაცემთა_ბაზის_ოფცია > ::=

{

ENABLE_BROKER

| **NEW_BROKER**

| **ERROR_BROKER_CONVERSATIONS**

| **RESTRICTED_USER**

| **FILESTREAM (DIRECTORY_NAME = { 'საქაღალდეს_სახელი' | NULL })**

}

განვიხილოთ პარამეტრები, რომლების ჯერ არ განვიხილავს:

- ▶ **ENABLE_BROKER** - განსაზღვრავს Service Broker კომპონენტისჩართულობას (sys.databases წარმოდგენაში is_broker_enabled-ს მიეთითება TRUE);
- ▶ **NEW_BROKER** - sys.databases წარმოდგენაში ქმნის service_broker_guid-ის ახალ მნიშვნელობას (ყველა ძველი მიმართვები კომპონენტის ინდეტიფიკატორზე ახლიდან უნდა შეიქმნან);
- ▶ **ERROR_BROKER_CONVERSATIONS** - ასრულებს ყველა შეცდომით მიერთებების დიალოგებს;
- ▶ **RESTRICTED_USER** - მითითებისას მიერთებას ახორციელებენ მხოლოდ db_owner, dbcreator და sysadmin-ის როლების წევრები.

მონაცემთა ბაზის მომენტალური სურათის ბრძანება

CREATE DATABASE ინსტრუქციის გამოყენებით SQL Server-ს გააჩნია შესაძლებლობა შექმნას მონაცემთა ბაზის მომენტალური სურათის სტატიკური წარმოდგენა მხოლოდ წაკითხვის რეჟიმში იმ მომენტისათვის, რომელშიც შეიქმნა მომენტალური სურათი. მონაცემთა ბაზას შეიძლება გააჩნდეს მრავალი მომენტალური სურათი სურათი.

თუ მონაცემთა ბაზის მომენტალური სურათის შექმნა ვერ მოხერხდა, მომენტალური სურათი მონიშნულია როგორც საეჭვო და უნდა წაიშალოს ინსტრუქციით DROP DATABASE. მომენტალური სურათი არსებობს მანამ, სანამ ის არ წაიშლება DROP DATABASE-ის გამოყენებით.

მონაცემთა ბაზის მომენტალური სურათის შესაქმნელად გამოიყენება **CREATE DATABASE** ბრძანება შემდეგი სინტაქსით:

```
CREATE DATABASE მომენტალური_სურათის_სახელი
ON
(
    NAME = ლოგიკური_სახელი,
    FILENAME = 'ფიზიკური_სახელი'
) [ ,...n ]
AS SNAPSHOT OF
[:]
```

მონაცემთა ბაზის სახელის შეცვლა

მონაცემთა ბაზის სახელის შესაცვლელად გამოიყენება **sp_renamedb** შენახული სისტემური პროცედურა, რომლის სინტაქსია:

sp_renamedb [@dbname =] 'ძველი_სახელი', [@newname =] 'ახალი_სახელი'

ამ შენახული პროცედურის შესრულების უფლება აქვს SQL Server-ის **sysadmin** ფიქსირებული როლის წევრებს.

მაგალითი:

Baza1 მონაცემთა ბაზას გადავარქვათ ახალი **Baza2** სახელი:

EXEC sp_renamedb 'Baza1', 'Baza2';

ან სრულად:

EXEC sp_renamedb @dbname = 'Baza1', @newname = 'Baza2';



The screenshot shows a SQL query window titled "SQLQuery3.sql - D...INQ5DSH\USER (53)*". The query is:

```
1 EXEC sp_renamedb 'Baza1', 'Baza2';
```

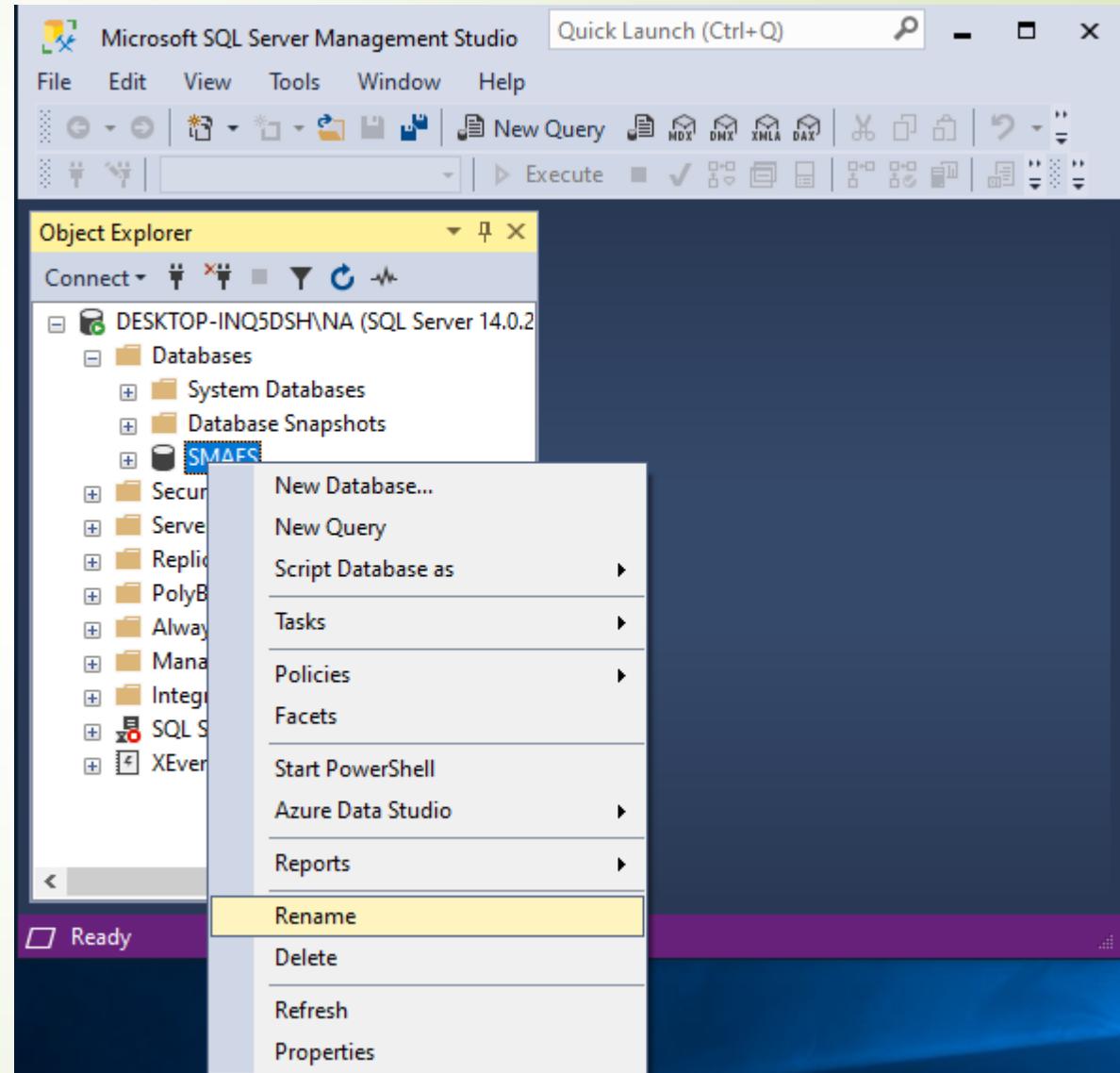
The results pane shows a message: "The database name 'Baza2' has been set."

მონაცემთა ბაზის სახელის შეცვლა

13

SQL Server Management Studio-ში (SSMS) მონაცემთა ბაზის სახელის შესაცვლელად მომხმარებელმა უნდა გაააქტიუროს მთავარ მენიუში თვით მონაცემთა ბაზის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს სტრიქონი „სახელის ცვლილება“ (Rename).

ასევე მონაცემთა ბაზის სახელის შესაცვლელად მომხმარებელმა უნდა გაააქტიუროს მთავარ მენიუში თვით მონაცემთა ბაზის სახელის სტრიქონი მაუსის მარცხენა ღილაკის გამოყენებით (ორჯერ ნელა დაჭრით) და შეცვალოს სახელი.



მონაცემთა ბაზის გამორთვა SQL Server-დან

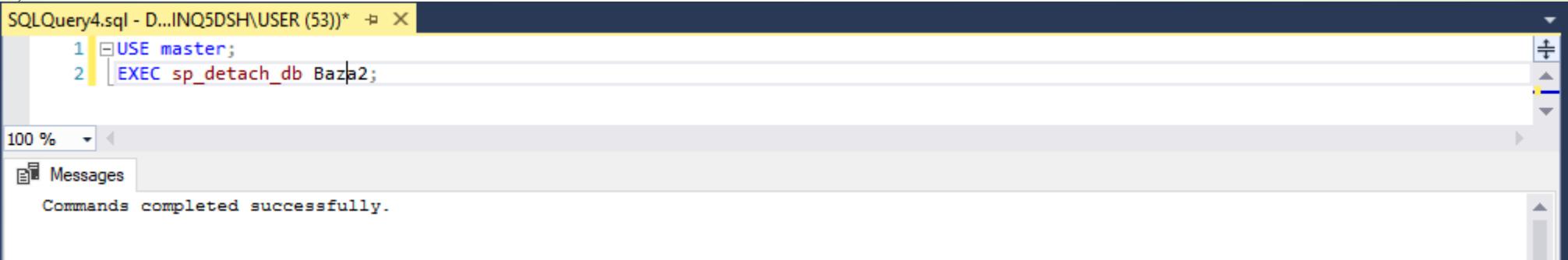
14

SQL Server-თან **BAZA2** მონაცემთა ბაზის გამორთვა და შემდეგ მისი მიერთება SQL Server-თან **FOR ATTACH** არგუმენტის გამოყენებით.

ძაგლითი:

USE master;

EXEC sp_detach_db BAZA2; -- BAZA2 მონაცემთა ბაზის გამორთვა SQL Server-დან



The screenshot shows a SQL Server Management Studio window titled "SQLQuery4.sql - D...INQ5DSH\USER (53)*". It contains two lines of T-SQL code:
1 USE master;
2 EXEC sp_detach_db Baza2;
Below the code, the "Messages" tab is selected, showing the output: "Commands completed successfully."

მონაცემთა ბაზის მიერთება SQL Server-თან

```
USE master;
-- იმ კატალოგისკენ გზის განსაზღვრა, რომელშიც იქმნება BAZA2 მონაცემთა ბაზა
DECLARE @data_path NVARCHAR(256);
SET @data_path =
(
SELECT SUBSTRING(physical_name, 1, CHARINDEX(N'master.mdf', LOWER(physical_name)) - 1)
FROM master.sys.master_files
WHERE database_id = 1 AND file_id = 1
);
-- BAZA2 მონაცემთა ბაზის მიერთება SQL Server-თან
EXEC (
'CREATE DATABASE BAZA2
ON
(
FILENAME = "' + @data_path + 'BAZA2_dat1.mdf"
)
FOR ATTACH'
);
```

მონაცემთა ბაზის მიერთება SQL Server-თან

მონაცემთა ბაზის SQL Server-თან მიერთების შედეგი:

The screenshot shows a SQL Server Management Studio window with two tabs: 'SQLQuery5.sql' and 'SQLQuery6.sql'. The code in 'SQLQuery5.sql' is as follows:

```
1 USE master;
2 DECLARE @data_path NVARCHAR(256);
3 SET @data_path =
4 (
5     SELECT SUBSTRING(physical_name, 1, CHARINDEX(N'master.mdf', LOWER(physical_name)) - 1)
6         FROM master.sys.master_files
7         WHERE database_id = 1 AND file_id = 1 );
8 EXEC (
9     'CREATE DATABASE Baza2
10        ON ( FILENAME = '''+ @data_path + 'Baza2.mdf''
11        FOR ATTACH' );
```

The code in 'SQLQuery6.sql' shows the execution results:

Commands completed successfully.

მონაცემთა ბაზის ცვლილება

17

მონაცემთა ბაზის ცვლილებისათვის გამოიყენება **ALTER DATABASE** ბრძანება. მისი სინტაქსია:

ALTER DATABASE {მონაცემთა_ბაზის_სახელი | CURRENT}

{

MODIFY NAME = ახალი_მონაცემთა_ბაზის_სახელი
| **COLLATE შეფარდების_სახელი**
| **<ფაილის_და_ფაილთაჯგუფების_ოფციები>**
| **SET <ოფციის_სპეციფიკაცია> [....n] [WITH <termination>]**

}

[:]

ეს სინტაქსი დეტალურად განიხილება სწავლების მეორე ეტაპზე - „MS SQL Server მონაცემთა ბაზების ადმინისტრირება“. ამ კურსში მხოლოდ მოყვანილია მისი ელემენტარულ დონეზე განხილვა:

მონაცემთა_ბაზის_სახელი - შესაცვლელი მონაცემთა ბაზის სახელია. თუ შესაცვლელია მიმდინარე მონაცემთა ბაზა, მაშინ შესაძლებელია მიეთითოს CURRENT;

ახალი_მონაცემთა_ბაზის_სახელი - მონაცემთა ბაზის სახელის ცვლილებისას შესაცვლელი მონაცემთა ბაზის ახალი სახელია;

მონაცემთა ბაზის ცვლილება

18

შეფარდების_სახელი - მონაცემთა ბაზის მონაცემთა ენის კოდირების სახელია და იგი შესაძლებელია იყოს როგორც Windows collation name-ის შესაბამისი, ასევე SQL collation name-ის;

<ფაილის_და_ფაილთაჯგუფების_ოფციები> - აქ შესაძლებელია განიხილებოდეს ფაილების ან ფაილთაჯგუფების დამატება და ცვლილება, ან ფაილების ან ფაილთაჯგუფების სპეციფიკაციების ცვლილება;

<ოფციის_სპეციფიკაცია> ::=

```
{ <auto_option>      | <change_tracking_option>      | <cursor_option>
  | <database_mirroring_option> | <date_correlation_optimization_option>
  | <db_encryption_option>      | <db_state_option> | <db_update_option>
  | <db_user_access_option><delayed_durability_option>
  | <external_access_option> | <FILESTREAM_options> | <HADR_options>
  | <parameterization_option> | <query_store_options> | <recovery_option>
  | <service_broker_option>    | <snapshot_option>   | <sql_option>
  | <termination>           | <temporal_history_retention> | <data_retention_policy>
  | <compatibility_level>
  { 150 | 140 | 130 | 120 | 110 | 100 | 90 }
```

}

მონაცემთა ბაზის ცვლილება

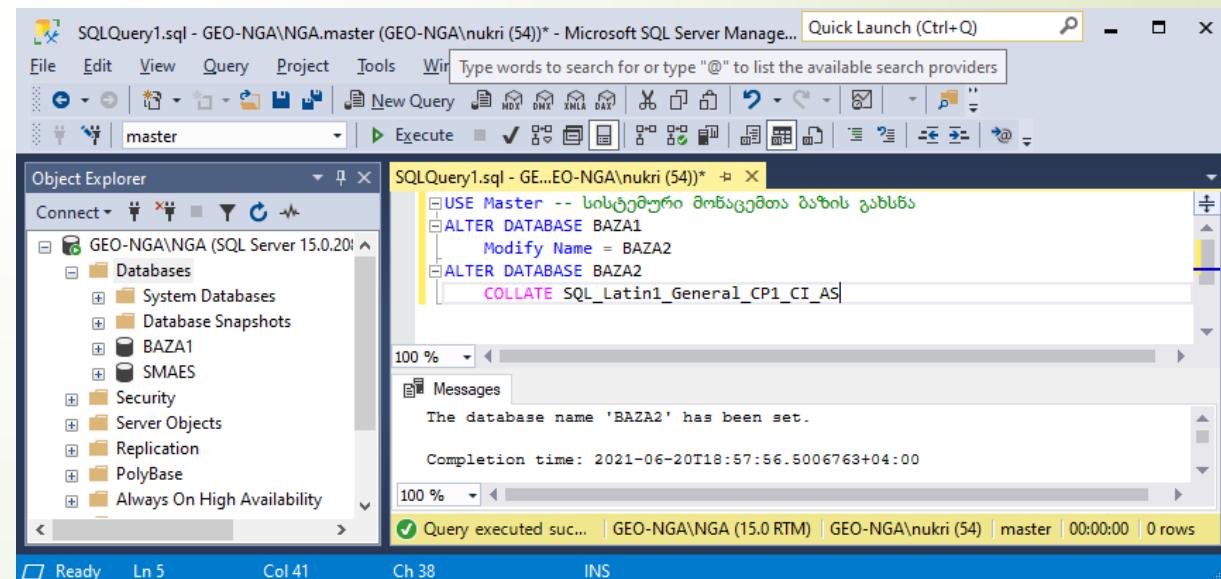
19

ძალითი:

შეუცვალოთ **BAZA1** მონაცემთა ბაზას სახელი **BAZA2**-ზე და, ასევე, შეფარდების სახელი **SQL_Latin1_General_CI_AS**-ზე:

```
USE Master -- სისტემური მონაცემთა ბაზის გახსნა  
ALTER DATABASE BAZA1  
    MODIFY NAME = BAZA2  
ALTER DATABASE BAZA2  
    COLLATE SQL_Latin1_General_CI_AS
```

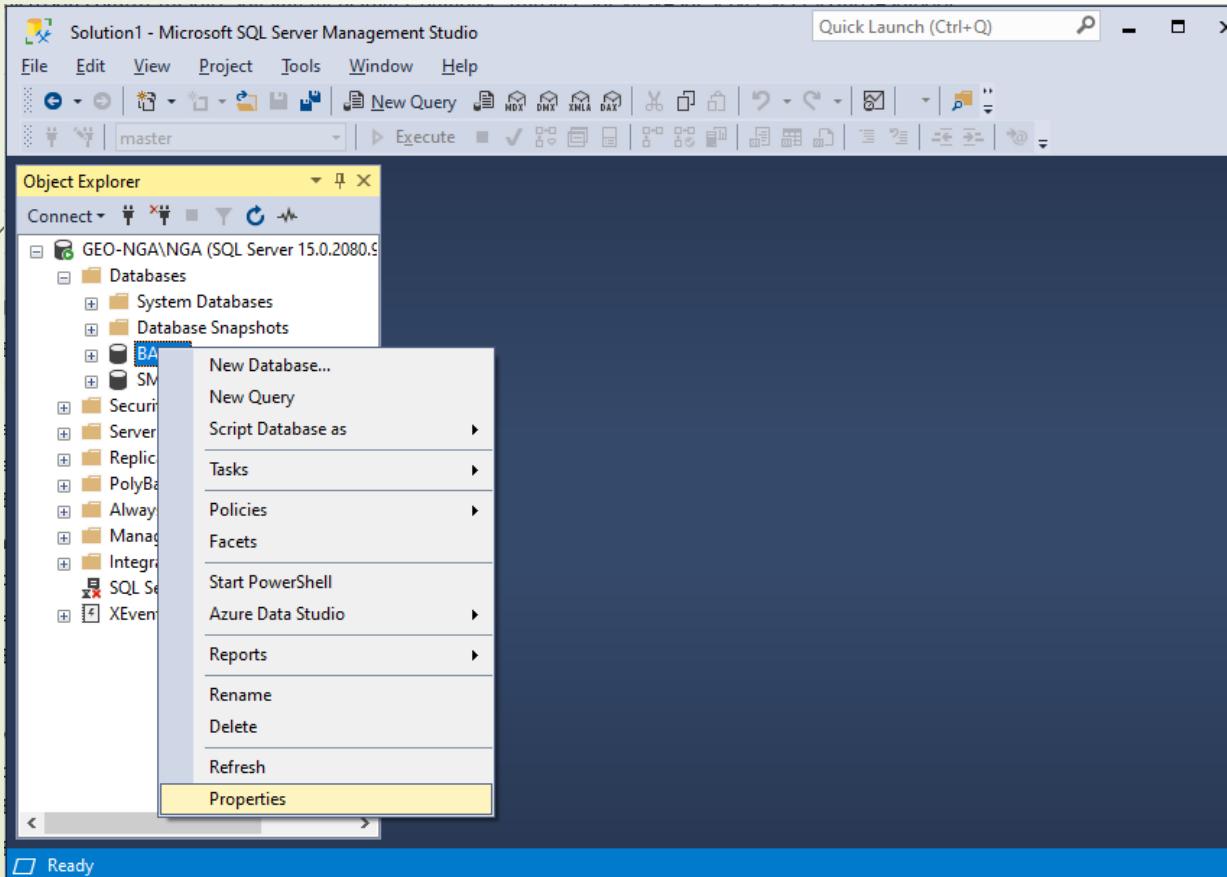
მონაცემთა ბაზის ცვლილების ბრძანებაში არგუმენტებს **MODIFY NAME** და **COLLATE**-ს შორის არის სიმბოლო „|“, რაც იმის მანიშნებელია, რომ ერთ **ALTER DATABASE** ბრძანებაში მხოლოდ ერთი არგუმენტის დაწერაა შესაძლებელი და, ამიტომ, გახდა საჭირო ორჯერ **ALTER DATABASE**-ს ბრძანების დაწერა.



მონაცემთა ბაზის ცვლილება

20

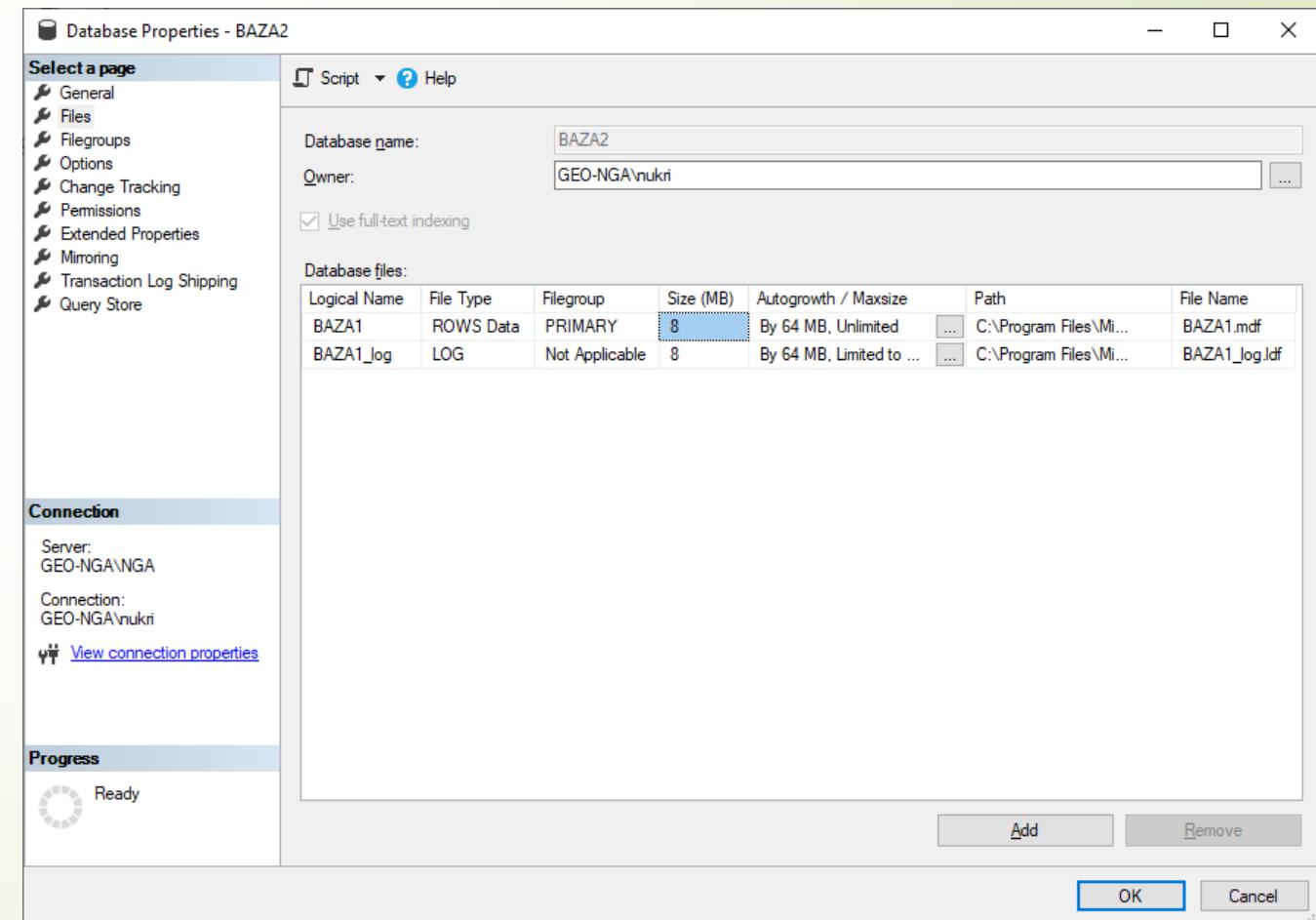
SQL Server Management Studio-ში (SSMS) მონაცემთა ბაზის ცვლილებისათვის მომხმარებელმა უნდა გაააქტიუროს მთავარ მენიუში საჭირო მონაცემთა ბაზის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს პოლო სტრიქონი „თვისებები“ (Properties).



მონაცემთა ბაზის ცვლილება

21

მონაცემთა ბაზის თვისებების სტრიქონის გააქტიურებისას მონიტორის ეკრანზე გამოისახება ამ მონაცემთა ბაზის თვისებების შესაბამისი ფანჯარა, რომელიც ზედა-მარცხენა მიღამოში განლაგებული მთავარი მენიუს სტრიქონები, რომლების გააქტიურებით და შემდგომ მარჯვენა მიღამოში სურვილისებრ ცვლილების შესაძლებლობა ეძლევა მომხმარებელს (მონაცემთა ბაზის ფიზიკური და ლოგიკური სახელის, ნაზარდის, მიმართვის გზის, ფაილთა ჯგუფის, მომხმარებლების წვდომის უფლებებისა და სხვა).



მონაცემთა ბაზის წაშლა

22

მონაცემთა ბაზის წაშლელად გამოიყენება **DROP DATABASE** ბრძანება, რომლის სინტაქსია:

**DROP DATABASE [IF EXISTS] { მონაცემთა_ბაზის_სახელი
| მონაცემთა_ბაზის_მომენტალური_კადრის_სახელი } [,...n]**

მონაცემთა ბაზის წაშლისას იშლება როგორც ლოგიკური მონაცემთა ბაზა SQL Server-ში, ასევე ამ მონაცემთა ბაზის ფაილებიც ფიზიკურ დისკზე.

აღსანიშნავია, რომ მონაცემთა ბაზის წაშლისათვის საჭიროა მომხმარებელს **CONTROL**-ში გააჩნდეს **ALTER ANY DATABASE** უფლება ან იყოს მონაცემთა ბაზის მფლობელი - **db_owner** (სისტემური მონაცემთა ბაზების წაშლა შეუძლებელია).

განვიხილოთ არგუმენტები:

IF EXISTS - არააუცილებელი არგუმენტი, რომელიც ამოწმებს ამ სახელით მონაცემთა ბაზის არსებობას და მხოლოდ არსებობის შემთხვევაში წაშლას;

მონაცემთა_ბაზის_სახელი - წასაშლელი მონაცემთა ბაზის სახელია;

მონაცემთა_ბაზის_მომენტალური_კადრის_სახელი - წასაშლელი მონაცემთა ბაზის მომენტალური კადრის სახელია.

მონაცემთა ბაზის წაშლა

23

მაგალითი:

წავშალოთ **BAZA2** მონაცემთა ბაზა:

**USE Master -- სისტემური მონაცემთა ბაზის გახსნა
DROP DATABASE IF EXISTS BAZA2**

როგორც მაგალითიდან ჩანს მონაცემთა ბაზის წაშლის ბრძანებაში გამოყენებულ იქნა არგუმენტი IF EXISTS, რაც გამორიცხავს შეცდომის ჩვენებას, გინდაც ასეთი სახელის მქონე მონაცემთა ბაზა არ არსებობდეს.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. In the Object Explorer on the left, under the 'GEO-NGA\NGA' connection, the 'Databases' node is expanded, showing 'System Databases', 'Database Snapshots', 'BAZA2', 'SMAES', and other system databases. In the center, the 'SQLQuery2.sql' window contains the following T-SQL code:

```
USE Master -- სისტემური მონაცემთა ბაზის გახსნა
DROP DATABASE IF EXISTS BAZA2
```

Below the code, the 'Messages' pane displays the output:

```
Commands completed successfully.

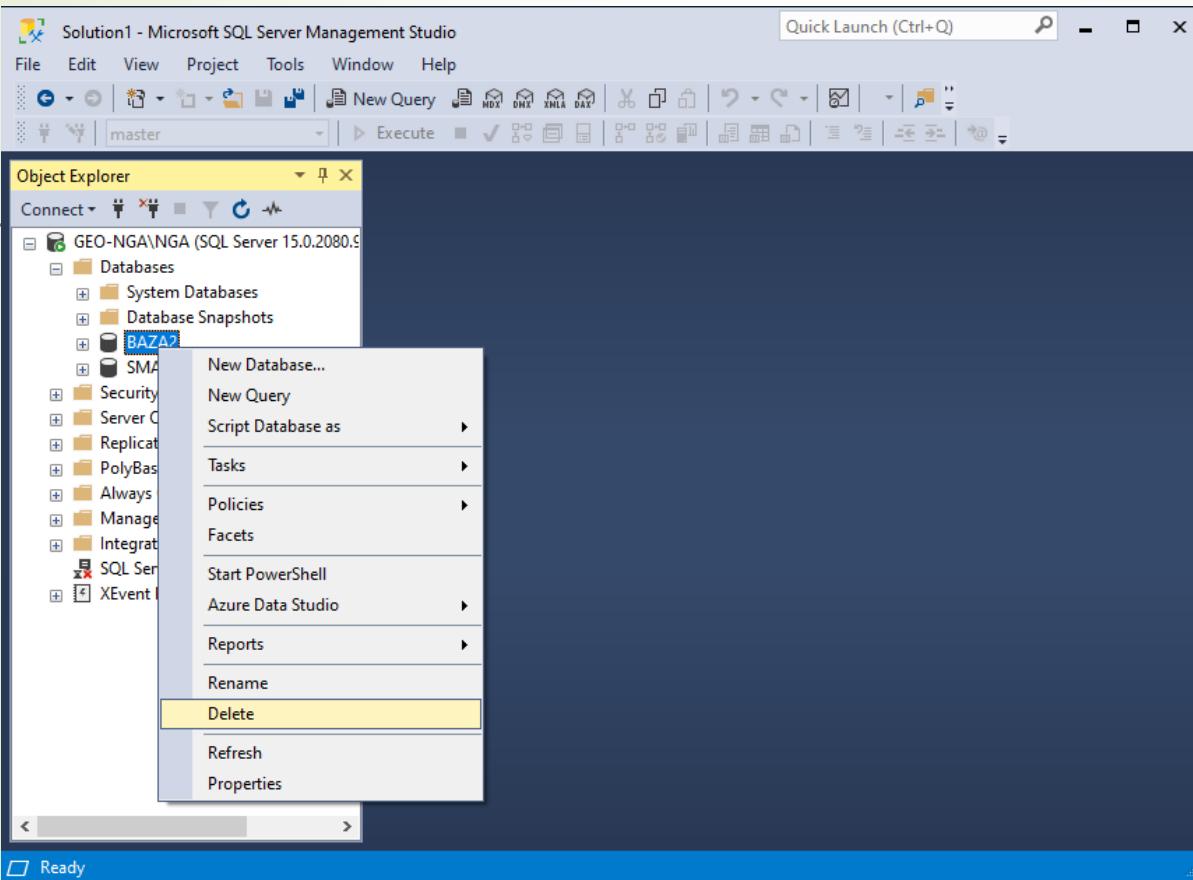
Completion time: 2021-06-20T20:01:35.1162096+04:00
```

At the bottom, status bars show 'Ready', 'Ln 2', 'Col 1', 'Ch 1', 'INS', and 'Query executed suc...', indicating the command was successful.

მონაცემთა ბაზის წაშლა

24

SQL Server Management Studio-ში (SSMS) მონაცემთა ბაზის წაშლისათვის მომხმარებელმა უნდა გაააქტიუროს მთავარ მენიუში საჭირო მონაცემთა ბაზის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს პოლოდან მესამე სტრიქონი „წაშლა“ (Delete).



ასევე შესაძლებელია მთავარ მენიუში საჭირო მონაცემთა ბაზის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით გააქტიურებისას გამოყენებულ იქნეს კლავიატურის ღილაკი **Delete**.

ინფორმაციის მიღება მონაცემთა ბაზების შესახებ

მონაცემთა ბაზების შესახებ ინფორმაციის მისაღებად შეგვიძლია გამოვიყენოთ **sp_helpdb** შენახული სისტემური პროცედურა, რომლის სინტაქსია:

```
sp_helpdb [ [ @dbname = ] 'მონაცემთა_ბაზის_სახელი' ]
```

ამ პროცედურას აქვს ორი შედეგი:

1. გამოისახება მონაცემთა ბაზის მიმდინარე ზომა, მონაცემთა ბაზის მფლობელის სააღრიცხვო ჩანაწერის სახელი, მონაცემთა ბაზის შექმნის თარიღი და მისი მიმდინარე სტატუსი;
2. გამოისახება ინფორმაცია მონაცემთა ბაზის ფაილების შესახებ:
 - 1) მონაცემთა ბაზის ფაილის ფიზიკური და ლოგიკური სახელი, საიდენტიფიკაციო ნომერი, მიმდინარე და მაქსიმალური ზომა, ნაზრდის ზომისა და ფაილების ამა თუ იმ ჯგუფთან კუთვნილება.

მაგალითი:

Baza1 მონაცემთა ბაზის შესახებ ინფორმაციის მისაღებად უნდა შევიტანოთ ბრძანება:

```
EXEC sp_helpdb 'Baza1';
```

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

ინფორმაციის მიღება მონაცემთა ბაზების შესახებ

26

The screenshot shows the SQL Server Management Studio interface with a query window titled "SQLQuery6.sql - D...INQ5DSH\USER (55)*". The query executed is "EXEC sp_helpdb 'Baza1';". The results are displayed in two tables under the "Results" tab.

Table 1: Database Information

	name	db_size	owner	dbid	created	status	compatibility_level
1	Baza1	16.00 MB	DESKTOP-INQ5DSH\USER	47	Mar 9 2019	Status=ONLINE, Updateability=READ_WRITE, UserAccess=MULTI_USE...	140

Table 2: File and Log Information

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	Baza1	1	C:\Program Files\Microsoft SQL Server\MSSQL14.NA\MSSQL\DATA\Baza1.mdf	PRIMARY	8192 KB	Unlimited	65536 KB	data only
2	Baza1_log	2	C:\Program Files\Microsoft SQL Server\MSSQL14.NA\MSSQL\DATA\Baza1_log.ldf	NULL	8192 KB	2147483648 KB	65536 KB	log only

მონაცემთა ცხრილის შექმნა

27

გავითვალისწინოთ, რომ აქ და ყველგან ‘ცხრილში’ იგულისხმება ‘მონაცემთა ცხრილი’. ცხრილის შესაქმნელად გამოიყენება **CREATE TABLE** ბრძანება, რომლის სინტაქსია:

```
CREATE TABLE [ მონაცემთა_ბაზის_სახელი.[ სქემის_სახელი. ] ] ცხრილის_სახელი
(
{ <სვეტის_განსაზღვრა>
| გამოთვლადი_სვეტის_სახელი AS გამოსახულება
| <ცხრილის_შეზღუდვა> } [,...n]
)
```

განვიხილოთ არგუმენტები:

- ▶ მონაცემთა_ბაზის_სახელი - იმ ბაზის სახელი, რომელშიც იქმნება ცხრილი (თუ არ არის მითითებული - ცხრილი შეიქმნება მიმდინარე მონაცემთა ბაზაში). მომხმარებელს, უნდა გააჩნდეს შესაბამისი უფლებები.
- ▶ სქემის_სახელი - იმ სქემის სახელი, რომელსაც ცხრილი ეკუთვნის.
- ▶ ცხრილის_სახელი - შესაქმნელი ცხრილის სახელი (ცხრილისა და სქემის სახელები მონაცემთა ბაზის ფარგლებში უნდა იყოს უნიკალური). თუ ცხრილი არ იქმნება მიმდინარე მონაცემთა ბაზაში, მაშინ მონაცემთა ბაზის სახელი აშკარად უნდა იქნეს მითითებული. თუ ცხრილის სახელის წინ მივუთითებთ # ან ## სიმბოლოებს, მაშინ შესაბამისად შეიქმნება ლოკალური ან გლობალური დროებითი ცხრილი.

მონაცემთა ცხრილის შექმნა

28

<სვეტის_განსაზღვრა> კონსტრუქციის სინტაქსია:

```
<სვეტის_განსაზღვრა> ::=  
სვეტის_სახელი ტიპი  
[ [ DEFAULT გამოსახულება ]  
| [ IDENTITY [ ( საწყისი_მნიშვნელობა, ნაზრდი ) ] ]  
[ <სვეტის_შეზღუდვა> ] [,...n]
```

განვიხილოთ ამ კონსტრუქციის პარამეტრების დანიშნულება:

- ▶ სვეტის_სახელი განსაზღვრავს სვეტის სახელს ცხრილში (უნიკალური ამ ცხრილში);
- ▶ ტიპი განსაზღვრავს სვეტში მოთავსებული მონაცემების ტიპს;
- ▶ **DEFAULT გამოსახულება** - განსაზღვრავს სვეტის ნაგულისხმევ (ავტომატურ) მნიშვნელობას: თუ სტრიქონის ჩასმისას ამ სვეტში აშკარად არ იქნება მითითებული მნიშვნელობა - მიენიჭება ნაგულისხმევი, რომელიც შეიძლება იყოს მუდმივა ან სისტემური ფუნქცია. ნაგულისხმევი მნიშვნელობის განსაზღვრა არ შეიძლება **timestamp** ტიპის და/ან **IDENTITY** თვისების მქონე სვეტებისათვის;
- ▶ **IDENTITY [(საწყისი_მნიშვნელობა, ნაზრდი)]** მიუთითებს, რომ შესაბამისი სვეტი იქნება სვეტი-მთვლელი, რომელშიც შესაძლებელია მიეთითოს საწყისი_მნიშვნელობა და ნაზრდი;

მონაცემთა ცხრილის შექმნა

29

<სვეტის_შეზღუდვა> კონსტრუქციაში ეთითება სვეტის მთლიანობის შეზღუდვები, რომელის სინტაქსია:

<სვეტის_შეზღუდვა> ::=
[CONSTRAINT შეზღუდვის_სახელი]
{
[NULL | NOT NULL] | [{ PRIMARY KEY | UNIQUE } | [[FOREIGN KEY] REFERENCES
ref_ცხრილი [(ref_სვეტი)]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
| CHECK (ლოგიკური_გამოსახულება)
}

განვიხილოთ ამ კონსტრუქციის პარამეტრების დანიშნულება:

- **CONSTRAINT** პარამეტრის შემდეგ ეთითება სვეტის მნიშვნელობებზე შეზღუდვის სახელი, რომელიც მონაცემთა ბაზის ფარგლებში უნიკალური უნდა იყოს;
 - **NULL | NOT NULL** - NULL პარამეტრი გვაძლევს სვეტში NULL მნიშვნელობის შენახვის უფლებას, ხოლო NOT NULL პარამეტრი კი - კრძალავს მას;

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ცხრილის შექმნა

30

- **PRIMARY KEY** - მიუთითებს, რომ სვეტის ბაზაზე უნდა შეიქმნას პირველადი გასაღები. თითოეული ცხრილისათვის შეიძლება შეიქმნას მხოლოდ ერთი ასეთი შეზღუდვა. პირველადი გასაღები ასეთი გზით განისაზღვრება სვეტის დონეზე. შედეგად, პირველადი გასაღები შემდგარი იქნება მხოლოდ ერთი სვეტისაგან. თუ საჭიროა პირველადი გასაღების ფორმირება ორი და მეტი სვეტის ბაზაზე, მაშინ საჭიროა მისი კონფიგურირება ცხრილის დონეზე, რომელიც შემდგომში განიხილება;
- **UNIQUE** - მიუთითებს, რომ სვეტში მოთავსებული მონაცემები უნიკალურია, ანუ სვეტი არ შეიცავს გამეორებად მნიშვნელობებს. **UNIQUE** მთლიანობის შეზღუდვისათვის ავტომატურად იქმნება ინდექსი. ერთ ცხრილში შესაძლებელია შეიქმნას რამდენიმე **UNIQUE** შეზღუდვა;
- **[FOREIGN KEY] REFERENCES** მთავარი_ცხრილი [(სვეტის_სახელი [,...n])] - მიუთითებს, რომ სვეტი იქნება გარე გასაღები მთავარი_ცხრილი პარამეტრში განსაზღვრული ცხრილისათვის. გარე გასაღებში შემავალი სვეტები შეიძლება მიმართავდეს მხოლოდ მთავარი_ცხრილი პარამეტრით განსაზღვრული ცხრილის **PRIMARY KEY** ან **UNIQUE** შეზღუდვის მქონე სვეტებს. სვეტის_სახელი არის პირველად გასაღებში შემავალი სვეტი ან სვეტების სია.

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ცხრილის შექმნა

31

- ▶ **ON DELETE { CASCADE | NO ACTION }** - თუ მივუთითებთ **CASCADE** საკვანძო სიტყვას, მაშინ მთავარი ცხრილიდან სტრიქონის წაშლის შემთხვევაში დამოკიდებულ ცხრილშიც წაიშლება შესაბამისი სტრიქონ(ებ)ი. თუ მივუთითებთ **NO ACTION** საკვანძო სიტყვას (თუ არაფერი არ მიეთითება ნაგულისხმევად ჩართულია **NO ACTION**), მაშინ SQL Server-ი გასცემს შეტყობინებას შეცდომის შესახებ და სტრიქონები არ წაიშლება მთავარი ცხრილიდან;
- ▶ **ON UPDATE { CASCADE | NO ACTION }** - ანალოგიურია **ON DELETE** საკვანძო სიტყვისა, იმ განსხვავებით, რომ მოქმედებს მთავარი ცხრილის პირველადი გასაღების მნიშვნელობის ცვლილებისას;
- ▶ **CHECK** (ლოგიკური_გამოსახულება) პარამეტრი ახორციელებს მთლიანობის შეზღუდვას იმ შესაძლო მნიშვნელობების შემოწმების გზით, რომლებიც შეგვაქვს სვეტში ან სვეტებში. თუ ლოგიკური გამოსახულება იღებს **true** მნიშვნელობას, მაშინ მონაცემების ცვლილების ან ჩასმის ოპერაციები ნებადართული იქნება, ხოლო თუ **false** - არა.

მონაცემთა ცხრილის შექმნა

32

► გამოსახულება - სვეტში გამოთვლის შედეგად მიღებული მნიშვნელობა. გამოთვლადი სვეტები ვირტუალური სვეტებია - ანუ ეს სვეტები ფიზიკურად ცხრილში არ ინახება. მათი მნიშვნელობა გამოითვლება ცხრილის გახსნის დროს. გამოსახულებაში შეიძლება მონაწილეობდეს სვეტების სახელები, მუდმივები, ფუნქციები. ქვემოთხოვნების გამოყენება დაუშვებელია. გამოთვლადი სვეტები, ასევე, შეიძლება ჩართული იქნეს **SELECT** მოთხოვნაში სვეტების სიის მითითების დროს.

გარდა ამისა, გამოთვლადი სვეტები შეიძლება გამოყენებულ იქნეს შემდეგ ვითარებებში:

- ინდექსებში ან როგორც პირველადი გასაღების ნაწილი: გამოთვლადი სვეტის მნიშვნელობა უნდა განისაზღვროს დეტერმინირებული გამოსახულებით (მაგ., ინდექსირება მთელრიცხვა **A** და **B** სვეტების არითმეტიკული კომბინაციით **A + B**).
- UNIQUE მნიშვნელობის განსაზღვრა. გამოთვლადი სვეტები არ უნდა შევიდეს გარე გასაღების შემადგენლობაში. მათთვის არ შეიძლება ავტომატური მნიშვნელობების განსაზღვრა. გამოთვლადი სვეტების გამოყენება არ შეიძლება **INSERT** და **UPDATE** ოპერაციებში.

მონაცემთა ცხრილის შექმნა

33

<ცხრილის_შეზღუდვა> კონსტრუქცია განსაზღვრავს მთლიანობის შეზღუდვებს ცხრილის დონეზე, რომლის სინტაქსია:

```
<ცხრილის_შეზღუდვა> ::=  
[ CONSTRAINT შეზღუდვის_სახელი ]  
{  
[ { PRIMARY KEY | UNIQUE }  
{ ( სვეტის_სახელი [ ASC | DESC ] [...n] ) } ]  
| FOREIGN KEY  
[ ( სვეტის_სახელი [...n] ) ]  
REFERENCES მთავარი_ცხრილი [ ( სვეტის_სახელი [...n] ) ]  
[ ON DELETE { CASCADE | NO ACTION } ]  
[ ON UPDATE { CASCADE | NO ACTION } ]  
| CHECK ( ლოგიკური_გამოსახულება )  
}
```

აქ კონსტრუქციის პარამეტრების დანიშნულება იგივეა, რაც სვეტის_შეზღუდვა-ში, იმ განსხვავებით, რომ პირველადი და გარე გასაღები (შესაბამისად მთავარი_ცხრილის სვეტის_სახელი) ცხრილს შეიძლება რამდენიმე გააჩნდეს. ასევე **ASC | DESC** პარამეტრი (ზრდადობა | კლებადობა) განსაზღვრავს ინდექსში მონაცემების დახარისხების მეთოდს.

მონაცემთა ცხრილის შექმნა

34

ძაგლითი 1:

შევქმნათ ცხრილი სახელად **Cxrili**, რომლის სვეტი **cx_ID** იქნება პირველადი გასაღები და ის იქნება სვეტი-მთვლელი. მთვლელის საწყისი მნიშვნელობა და ნაზრდი 1-ის ტოლია:

USE Baza1;

-- თუ ცხრილი არსებობს, მაშინ ის წაიშლება

IF OBJECT_ID(N'Cxrili', N'U') IS NOT NULL

DROP TABLE Cxrili;

-- ცხრილის შექმნა

CREATE TABLE Cxrili

(

cx_ID INT PRIMARY KEY IDENTITY (1,1),

mteli INT,

texti NVARCHAR(20),

tsiladi FLOAT,

dro DATETIME

);

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ცხრილის შექმნა

35

ბრძანებების ჩასმა:

The screenshot shows a SQL query window titled "SQLQuery7.sql - D...INQ5DSH\USER (51)*". The query creates a table named "Cxrili" with five columns: "cx_ID" (primary key, identity), "mteli" (int), "texti" (nvarchar(20)), "tsiladi" (float), and "dro" (datetime). The command is executed successfully, as indicated by the message "Commands completed successfully." in the "Messages" pane.

```
1 USE Baza;
2 IF OBJECT_ID(N'Cxrili', N'U') IS NOT NULL
3 DROP TABLE Cxrili;
4 CREATE TABLE Cxrili
5 (
6     cx_ID INT PRIMARY KEY IDENTITY (1,1),
7     mteli INT,
8     texti NVARCHAR(20),
9     tsiladi FLOAT,
10    dro DATETIME
11 );
```

100 %

Messages
Commands completed successfully.

შედეგი:

The screenshot shows the "DESKTOP-INQ5DSH\N...aza1 - dbo.Cxrili" table structure. It has five columns: "cx_ID" (primary key, int), "mteli" (int), "texti" (nvarchar(20)), "tsiladi" (float), and "dro" (datetime). The "Allow Nulls" checkbox is checked for all columns except "cx_ID".

Column Name	Data Type	Allow Nulls
cx_ID	int	<input type="checkbox"/>
mteli	int	<input checked="" type="checkbox"/>
texti	nvarchar(20)	<input checked="" type="checkbox"/>
tsiladi	float	<input checked="" type="checkbox"/>
dro	datetime	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

მონაცემთა ცხრილის შექმნა

36

მაგალითი 2:

შევქმნათ ცხრილი შემდეგი პირობებით:

- ▶ **cx_ID** სვეტი იქნება პირველადი გასაღები და ის იქნება სვეტი-მთვლელი, რომლის საწყისი მნიშვნელობა და ნაზრდი 1-ის ტოლია. მისი მნიშვნელობები დახარისხებული უნდა იყოს კლებადობით, რომლისთვისაც გამოიყენება სვეტის შეზღუდვა **CONSTRAINT**, სახელით **FK_C1**;
- ▶ **mteli** სვეტი იქნება გარე გასაღები და დაუკავშირდება მთავარი **Cxrili2** ცხრილის **cx2_ID** პირველად გასაღებს და მთავარი ცხრილიდან სტრიქონის მნიშვნელობის ცვლილებისას ან/და წაშლისას დამოვიდებული ცხრილის შესაბამისი სტრიქონების მნიშვნელობები შეიცვალოს ან/და წაიშალოს;
- ▶ **texti** სვეტს ექნება შეზღუდვა **UNIQUE** და აუცილებლად იწყებოდეს სიმბოლო ‘ა’-ით;
- ▶ **tsiladi** წილადი ტიპის ველი;
- ▶ **dro** სვეტს ექნება განსაზღვრული ნაგულისხმევი მნიშვნელობა ‘მიმდინარე დრო’ და შეზღუდვა ‘მხოლოდ 2020 წლამდე’;
- ▶ **tvla** სვეტი იქნება გამოთვლადი ფორმულით **cx_ID + mteli / tsiladi**.

```
USE Baza1;
-- თუ ცხრილი არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID(N'Cxrili', N'U') IS NOT NULL
DROP TABLE Cxrili;
-- ცხრილის შექმნა
CREATE TABLE Cxrili
(
    cx_ID INT CONSTRAINT FK_C1 PRIMARY KEY (cx_ID DESC) IDENTITY (1,1),
    mtecli INT NULL REFERENCES Cxrili2(cx2_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    texti NVARCHAR(20) NOT NULL UNIQUE CHECK (texti LIKE N'%'),
    tsiladi FLOAT,
    dro DATETIME DEFAULT GETDATE() CHECK (dro <= '2019-12-31'),
    tvla AS cx_ID + mtecli / tsiladi
);
(შედეგი ნაჩვენებია შემდეგ სლაიდზე)
```

მონაცემთა ცხრილის შექმნა

38

ბრძანებების ჩასმა:

The screenshot shows a SQL query window titled "SQLQuery8.sql - D...INQ5DSH\USER (51)*". The code creates a table named "Cxrili" with the following structure:

```
USE Bazal;
IF OBJECT_ID(N'Cxrili', N'U') IS NOT NULL
DROP TABLE Cxrili;
CREATE TABLE Cxrili
(
    cx_ID INT CONSTRAINT FK_C1 PRIMARY KEY (cx_ID DESC) IDENTITY (1,1),
    mtelis INT NULL REFERENCES Cxrili2(cx2_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    texti NVARCHAR(20) NOT NULL UNIQUE CHECK (texti LIKE N's%'),
    tsiladi FLOAT,
    dro DATETIME DEFAULT GETDATE() CHECK (dro <= '2019-12-31'),
    tvla AS cx_ID + mtelis / tsiladi
);
```

The "Messages" pane at the bottom indicates "Commands completed successfully."

შედეგი:

The screenshot shows the table structure for "Cxriili" in the "DESKTOP-INQ5DSH\N...aza1 - dbo.Cxrili" object. The columns and their properties are:

Column Name	Data Type	Allow Nulls
cx_ID	int	<input type="checkbox"/>
mtelis	int	<input checked="" type="checkbox"/>
texti	nvarchar(20)	<input type="checkbox"/>
tsiladi	float	<input checked="" type="checkbox"/>
dro	datetime	<input checked="" type="checkbox"/>
tvla		<input checked="" type="checkbox"/>

მონაცემთა ცხრილის შექმნა

39

მაგალითი 3:

შევქმნათ მთავარი ცხრილი **Cxrili1**, რომლის პირველადი გასაღები იქნება **cx1_ID1** და **cx1_ID2** სვეტები ერთად, რისთვისაც გამოიყენება შეზღუდვა **PK_C1**. ასევე შევქმნათ დამოკიდებული ცხრილი **Cxrili2**, რომლის გარე გასაღები იქნება **cx2_ID1** და **cx2_ID2** სვეტები ერთად, რისთვისაც გამოიყენება შეზღუდვა **PK_C2**, რომელიც განსაზღვრავს წაშლისას კასკადურობას:

USE Baza1;

-- თუ მთავარი ცხრილი არსებობს, მაშინ ის წაიშლება

IF OBJECT_ID(N'Cxrili1', N'U') IS NOT NULL

DROP TABLE Cxrili1;

-- მთავარი ცხრილის შექმნა

CREATE TABLE Cxrili1

(

cx1_ID1 INT,

cx1_ID2 INT,

dro1 DATETIME,

CONSTRAINT PK_C1 PRIMARY KEY (cx1_ID1, cx1_ID2)

);

(გაგრძელება ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ცხრილის შექმნა

40

```
-- თუ დამოკიდებული ცხრილი არსებობს, მაშინ ის წაიშლება
IF OBJECT_ID(N'Cxrili2', N'U') IS NOT NULL
DROP TABLE Cxrili2;
-- დამოკიდებული ცხრილის შექმნა
CREATE TABLE Cxrili2
(
    cx2_ID INT PRIMARY KEY IDENTIFY (1,1),
    cx2_ID1 INT,
    cx2_ID2 INT,
    dro2 DATETIME,
    CONSTRAINT PK_C2 FOREIGN KEY (cx2_ID1, cx2_ID2)
        REFERENCES Cxrili1 (cx1_ID1, cx1_ID2)
        ON DELETE CASCADE
);
(შედეგი ნაჩვენებია შემდეგ სლაიდზე)
```

მონაცემთა ცხრილის შექმნა

41

ბრძანებების ჩასმა:

The screenshot shows a SQL query window titled "SQLQuery9.sql - D...INQ5DSH\USER (53)*". The code creates two tables, Cxrili1 and Cxrili2, in the database Baza1. Table Cxrili1 has columns cx1_ID1, cx1_ID2, and dro1 DATETIME, with a primary key constraint PK_C1 on cx1_ID1 and cx1_ID2. Table Cxrili2 has columns cx2_ID (primary key identity), cx2_ID1, cx2_ID2, and dro2 DATETIME, with a foreign key constraint PK_C2 referencing the primary key of Cxrili1. The command "Commands completed successfully." is shown in the Messages pane.

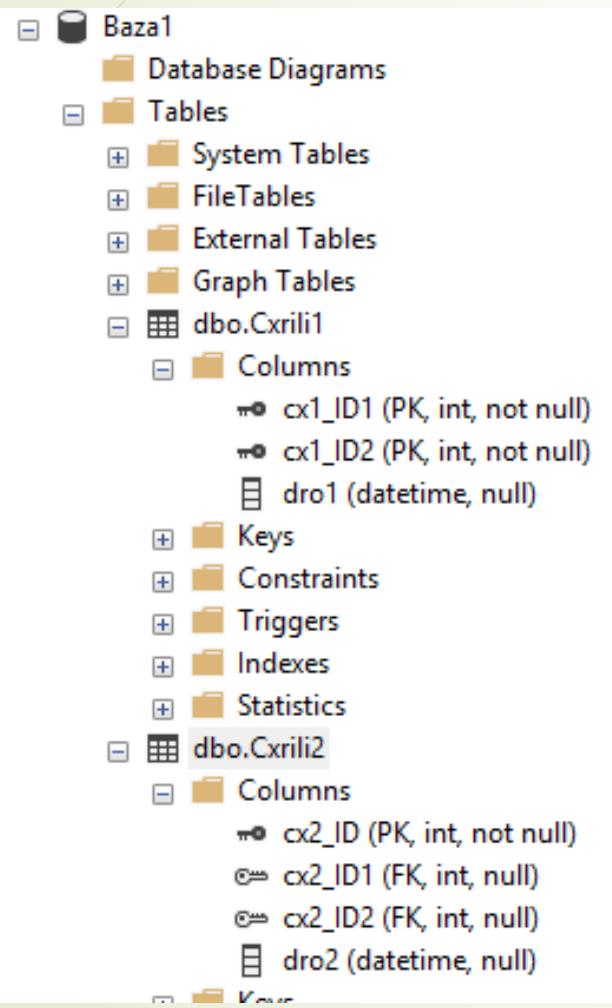
```
1 USE Baza1;
2 IF OBJECT_ID(N'Cxrili1', N'U') IS NOT NULL
3     DROP TABLE Cxrili1;
4 CREATE TABLE Cxrili1
5     (
6         cx1_ID1 INT,
7         cx1_ID2 INT,
8         dro1 DATETIME,
9         CONSTRAINT PK_C1 PRIMARY KEY (cx1_ID1, cx1_ID2)
10    );
11 IF OBJECT_ID(N'Cxrili2', N'U') IS NOT NULL
12     DROP TABLE Cxrili2;
13 CREATE TABLE Cxrili2
14     (
15         cx2_ID INT PRIMARY KEY IDENTITY (1,1),
16         cx2_ID1 INT,
17         cx2_ID2 INT,
18         dro2 DATETIME,
19         CONSTRAINT PK_C2 FOREIGN KEY (cx2_ID1, cx2_ID2)
20             REFERENCES Cxrili1 (cx1_ID1, cx1_ID2)
21             ON DELETE CASCADE );
```

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ცხრილის შექმნა

42

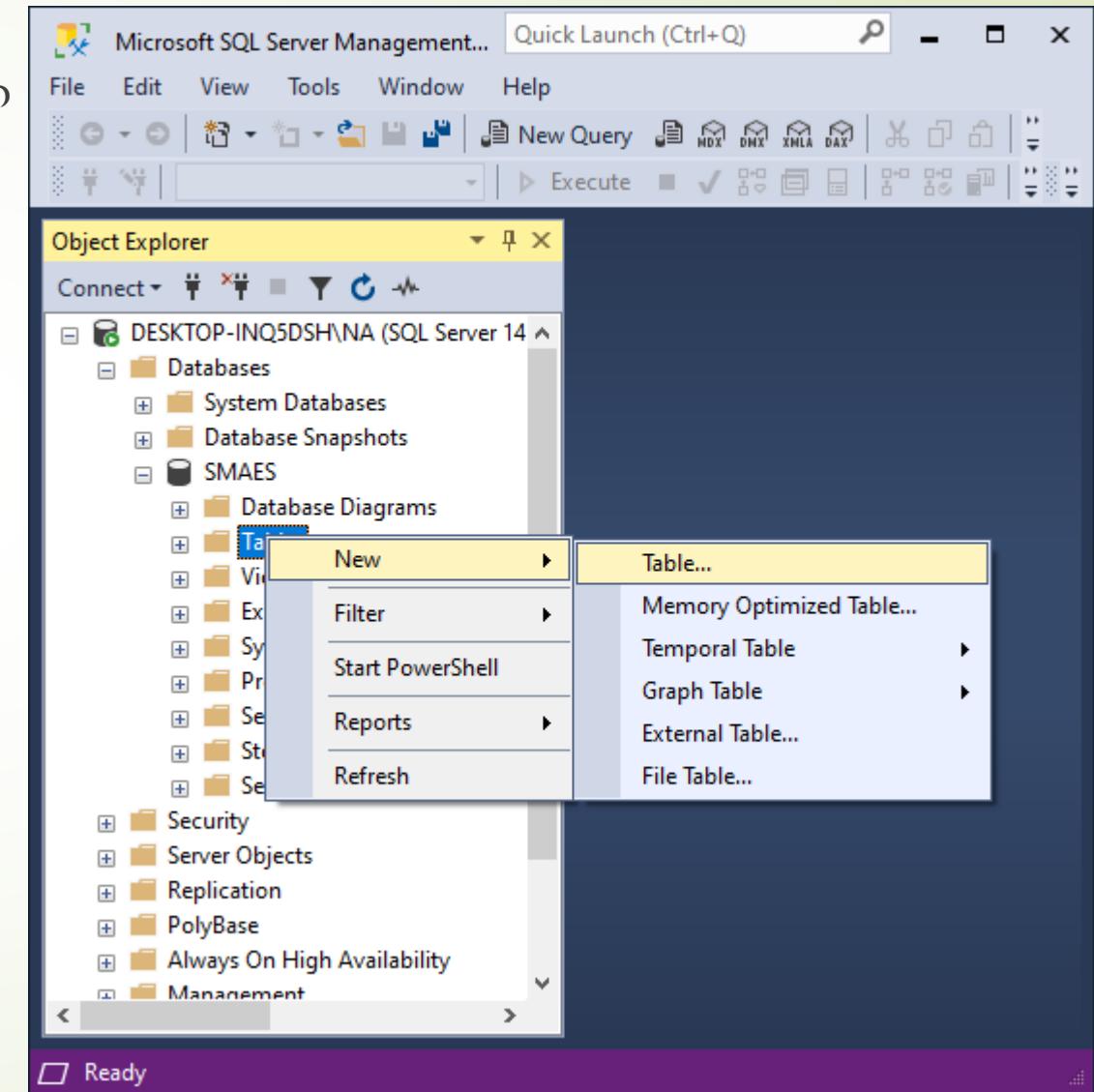
შედეგი:



მონაცემთა ცხრილის შექმნა

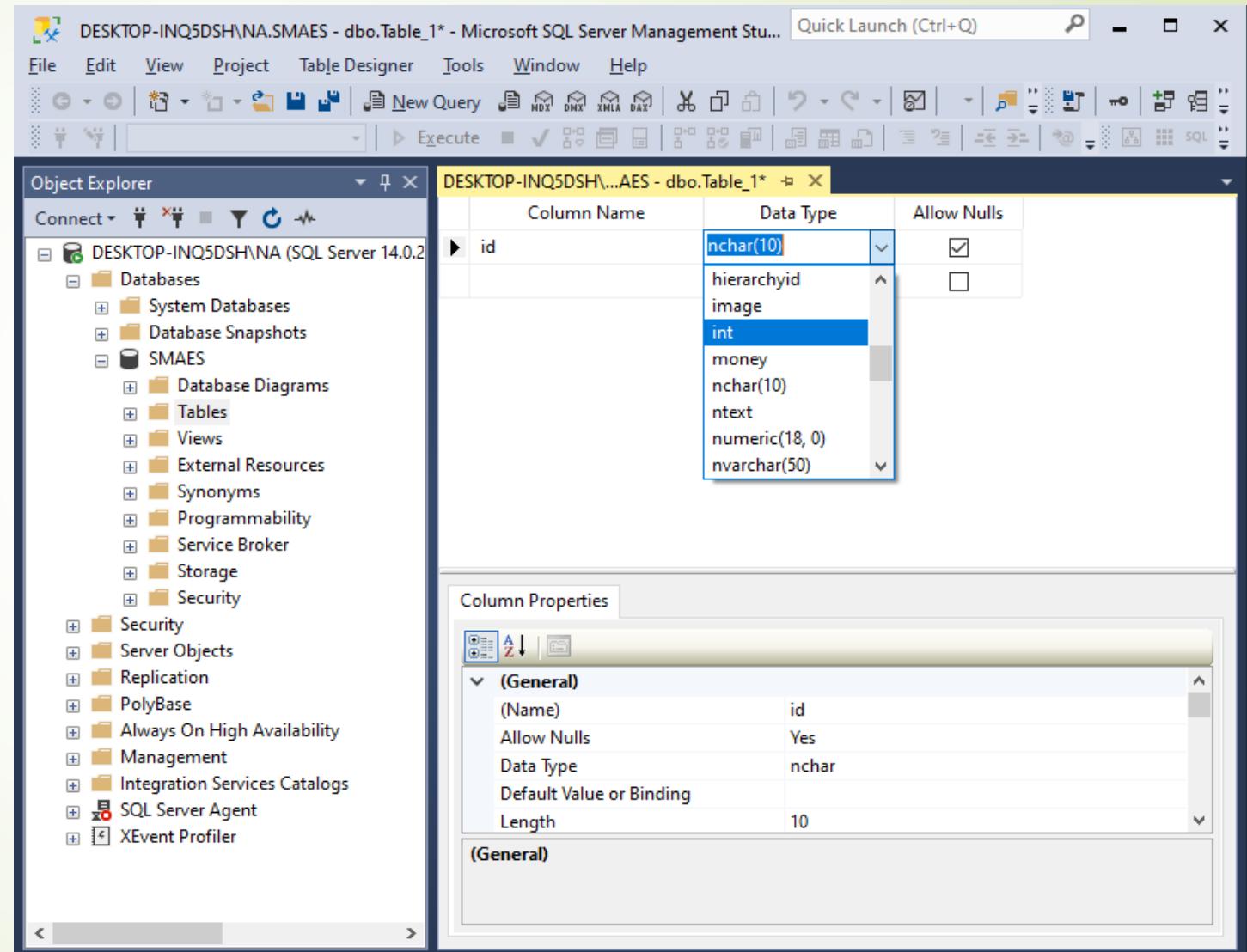
43

SQL Server Management Studio-ში (SSMS) მომხმარებლის მონაცემთა ბაზაში მონაცემთა ცხრილის შექმნისათვის მომხმარებელმა უნდა გაააქტიუროს მთავარი მენიუს მომხმარებლის მონაცემთა ბაზაში „ცხრილები“-ს (Tables) სტრიქონი მაუსის მარჯვენა ჭილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს პირველივე სტრიქონი „ახალი“ (New) და შემდეგ ისევ პირველივე სტრიქონი „ცხრილი...“ (Table...).



მონაცემთა ცხრილის შექმნა

ფანჯრის ზედა-მარჯვენა მიღამოში გამოსახულ ცხრილში მომხმარებელს ეძლევა შესაძლებლობა სტრიქონებად შეიტანოს სვეტში „Column Name“ მონაცემთა ცხრილის სვეტის სახელი, ჩამოსაშლელ სვეტში „Data Type“ შეარჩიოს მონაცემთა ცხრილის ამ სვეტის ტიპი, ხოლო ბოლო სვეტში „Allow Nulls“ მონიშნოს დასაშვებია თუ არა ამ სვეტში Null (უცნობი) მნიშვნელობების ჩაწერა. და ასე ყველა მონაცემთა სვეტისათვის.



მონაცემთა ცხრილის შექმნა

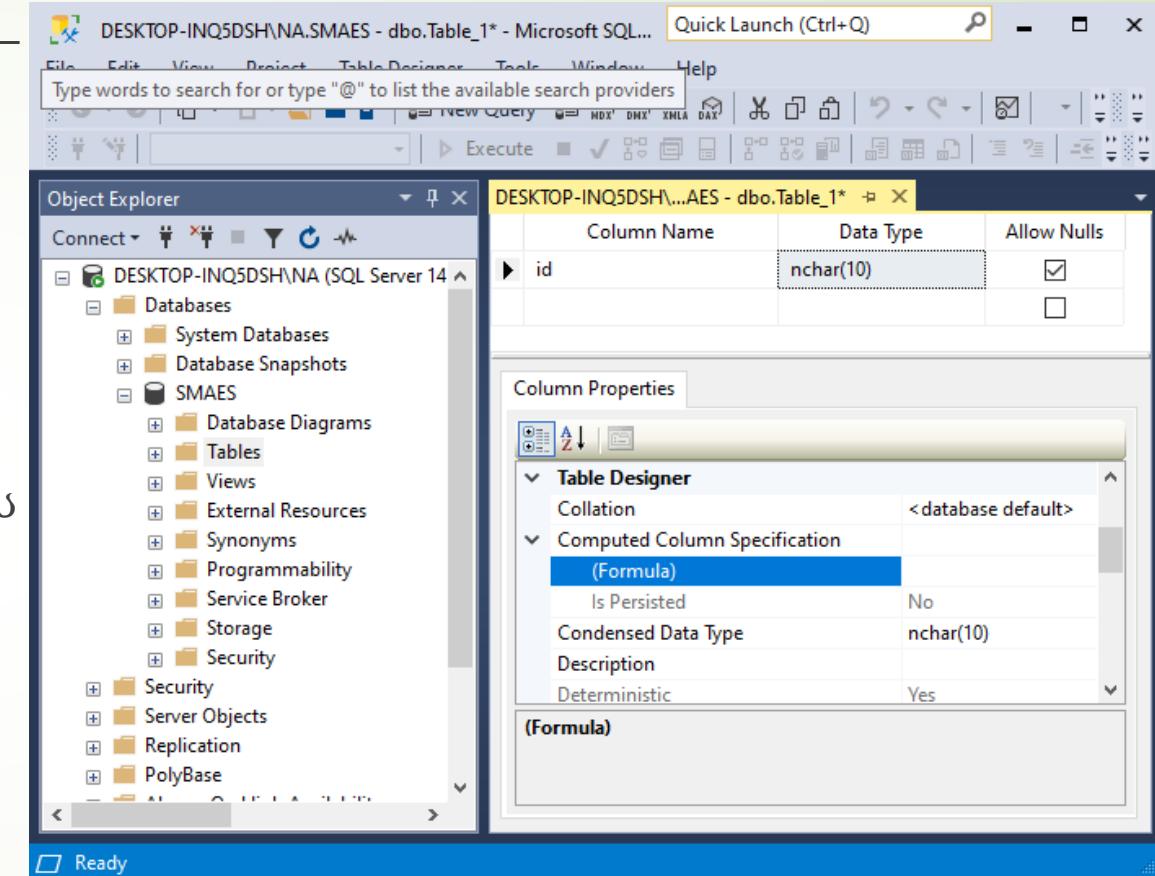
45

ფანჯრის ქვედა-მარჯვენა მიდამოში გამოსახულ „სვეტის თვისებებში“ (Column Properties) მომხმარებელს ეძლევა შესაძლებლობა შეიტანოს/დააკორექტოს სვეტის თვისებები. განვიხილოთ ძირითადები:

- ▶ (Name) - სვეტის სახელი;
- ▶ Allow Nulls - დასაშვებია თუ არა ამ სვეტში Null (უცნობი) მნიშვნელობების ჩაწერა;
- ▶ Data Type - სვეტის ტიპი;
- ▶ Default Value or Binding – იწერება სვეტში შესატანი ავტომატურად შესატანი მნიშვნელობა, თუ მონაცემთა ცხრილში ახალი მონაცემთა სტრიქონის დამატებისას ამ სვეტში ჩასაწერი მნიშვნელობა არ იქნა მითითებული;
- ▶ Length - იწერება სვეტისათვის მითითებულ ტიპის ზომა;
- ▶ Collation - ტექსტური სიმბოლიკის მახასიათებელი, რომელიც ეთითება MS SQL Server-ის ინსტალირებისას და შემდგომ ავტომატურად გამოიყენება (თუ აუცილებელია შესაძლებელია განსხვავებული სვეტის დონეზეც განისაზღვროს);

მონაცემთა ცხრილის შექმნა

- Computed Column Specification – გამოიყენება თვლად სვეტში ფორმულის ჩასაწერად, რისთვისაც მომხმარებელმა უნდა გააკტიუროს მაუსის მარცხენა ღილაკით სტრიქონის დასაწყისში მდებარე მარჯვნივ მიმართული ისარი და ახლად გამოსახულ შემდეგ სტრიქონში (Formula) უნდა ჩაწეროს შესაბამისი ფორმულა;

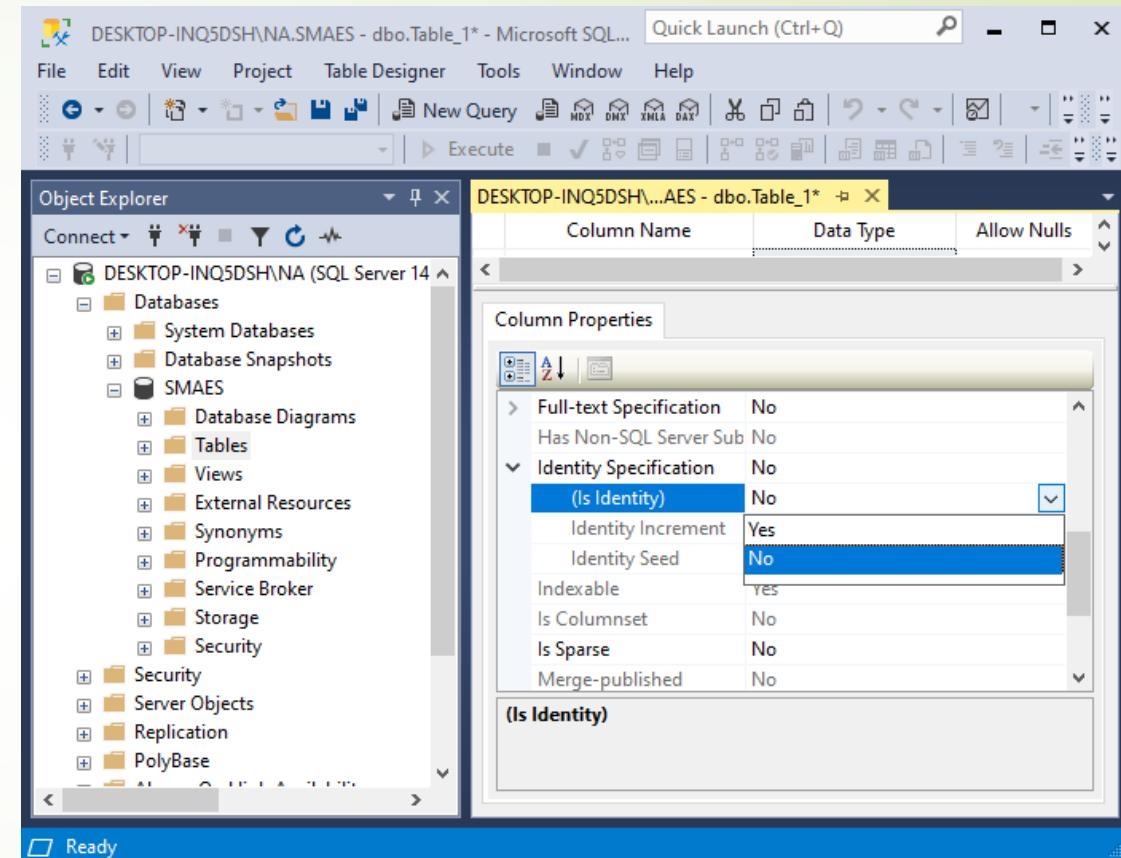


- Description - გამოიყენება მონაცემთა სვეტის აღწერისათვის, რომელსაც MS SQL Server-ი პრაქტიკულად არ იყენებს და არის მხოლოდ მომხმარებლისათვის განკუთვნილი;

- ▶ Identity Specification – გამოიყენება სვეტ-მთვლელის მისათითებლად, რისთვისაც მომხმარებელმა უნდა გაააქტიუროს მაუსის მარცხენა ღილაკით სტრიქონის დასაწყისში მდებარე მარჯვნივ მიმართული ისარი და ახლად გამოსახულ შემდეგ ჩამოსაშლელ სტრიქონში (Is Identity) უნდა მიუთითოს „კი“ (Yes) და საჭიროებისამებრ შემდეგ სტრიქონში „Identity Increment“ მიუთითოს თვლის ნაზდი (ავტომატურად სისტემა იღებს 1-ს), ხოლო იმის შემდეგ სტრიქონში „Identity Seed“ მიუთითოს საწყისი მნიშვნელობა (ავტომატურად სისტემა იღებს 1-ს).

მონაცემთა ცხრილის შექმნის დასრულებისას უნდა იქნეს დამახსოვრებული, როდესაც მონიტორის ეკრანზე გამოისახება სახელის შეყვანის დიალოგური ფანჯარა.

მონაცემთა ცხრილის შექმნა



მონაცემთა ცხრილის სახელის შეცვლა

48

ცხრილის სახელის შესაცვლელად გამოიყენება **sp_rename** შენახული სისტემური პროცედურა, რომლის სინტაქსია:

sp_rename 'ძველი_სახელი', 'ახალი_სახელი'

მაგალითი:

Cxrili1 მონაცემთა ცხრილს გადავარქვათ ახალი **Cxrili2** სახელი:

USE Baza1;

EXEC sp_rename 'Cxrili1', 'Cxrili2';

შედეგი:

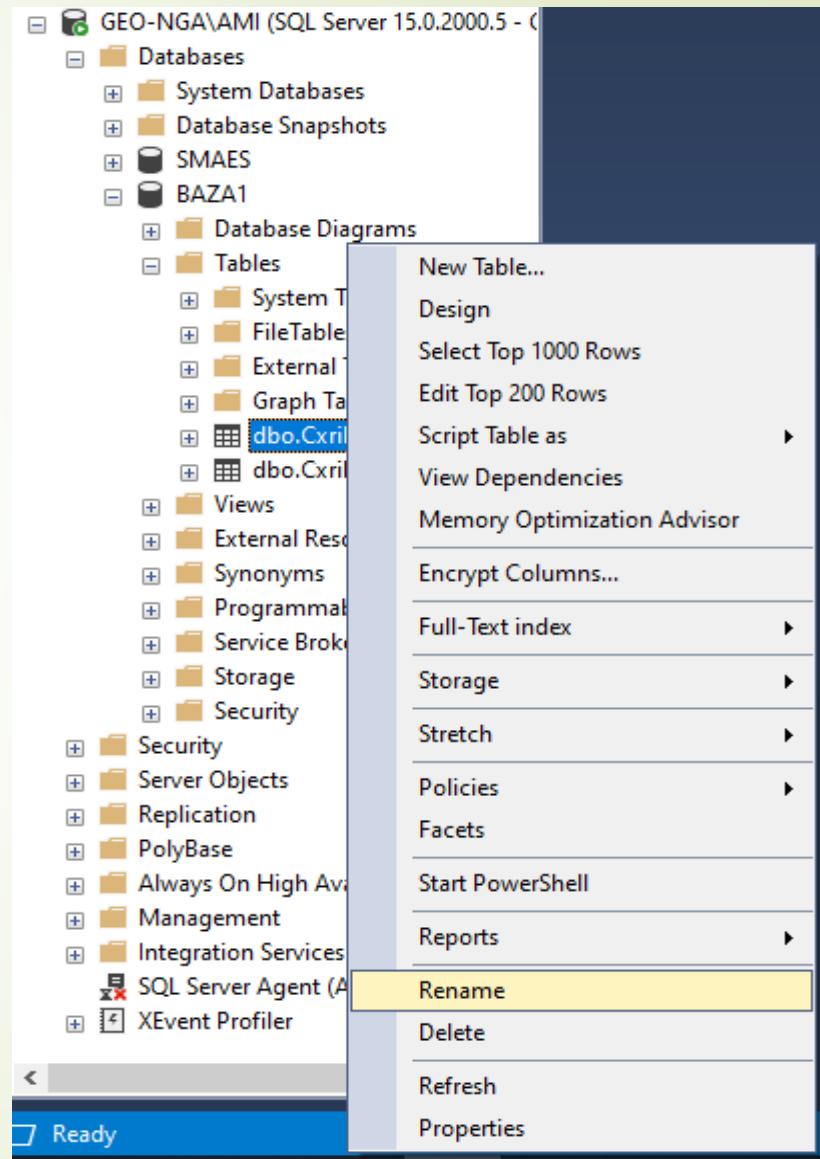
```
SQLQuery11.sql -...INQ5DSH\USER (51)* + X
1 USE Baza1;
2 EXEC sp_rename 'Cxrili1', 'Cxrili2';

100 % < >
Messages
Caution: Changing any part of an object name could break scripts and stored procedures.
```

მონაცემთა ცხრილის სახელის შეცვლა

SQL Server Management Studio-ში (SSMS) მომხმარებლის მონაცემთა ბაზაში უკვე არსებული მონაცემთა ცხრილის სახელის შესაცვლელად მომხმარებელმა უნდა გააქტიუროს მთავარი მენიუს მომხმარებლის მონაცემთა ბაზის „ცხრილები“-ს (Tables) ჩანართში სასურველი ცხრილის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს ბოლოდან მეოთხე სტრიქონი „სახელის შეცვლა“ (Rename). რის შემდეგაც გააქტიურებული ცხრილის სახელი ჩაასწოროს და დაამოწმოს კლავიატურაზე Enter ღილაკის გამოყენებით.

ასევე, უკვე არსებული მონაცემთა ცხრილის სახელის შეცვლისათვის მომხმარებელს პირდაპირ აქვს შესაძლებლობა სასურველი ცხრილის მონიშნულ სახელზე მაუსის მარცხენა ღილაკით გააქტიუროს ცხრილის სახელი, ჩაასწოროს და დაამოწმოს კლავიატურაზე Enter ღილაკის გამოყენებით.



მონაცემთა ცხრილის სვეტის სახელის შეცვლა

50

ამავე პროცედურის გამოყენებით შეგვიძლია შეცვალოთ სახელი ცხრილის სვეტის:

```
sp_rename 'ცხრილის_სახელი.ძველი_სვეტის_სახელი', 'ახალი_სვეტის_სახელი ',  
'COLUMN';
```

მაგალითი:

Cxrili1 მონაცემთა ცხრილის სვეტის Sveti1 გადავარქვათ ახალი Sveti2 სახელი:

```
USE Baza1;
```

```
EXEC sp_rename 'Cxrili1.Sveti1', 'Sveti2', 'COLUMN';
```

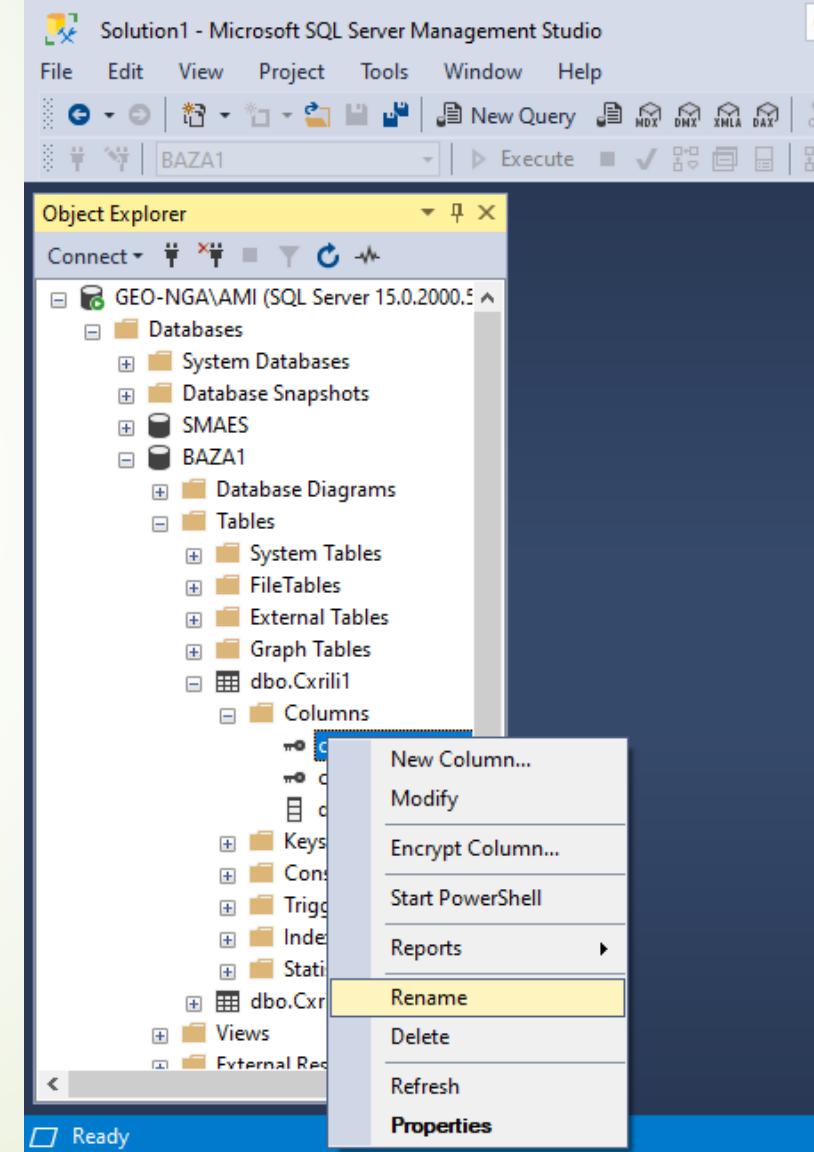
შედეგი:

```
SQLQuery12.sql -...INQ5DSH\USER (55) * USE Baza1;  
1 | 2 | EXEC sp_rename 'Cxrili1.Sveti1', 'Sveti2', 'COLUMN';  
  
100 %  
Messages  
Caution: Changing any part of an object name could break scripts and stored procedures.
```

მონაცემთა ცხრილის სვეტის სახელის შეცვლა

SQL Server Management Studio-ში (SSMS) მომხმარებლის მონაცემთა ბაზაში უკვე არსებული მონაცემთა ცხრილის სვეტის სახელის შესაცვლელად მომხმარებელმა უნდა გააქტიუროს მთავარი მენიუს მომხმარებლის მონაცემთა ბაზის „ცხრილები“-ს (Tables) ჩანართში შესაბამისი ცხრილის სასურველი სვეტის სახელი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს ბოლოდან მეოთხე სტრიქონი „სახელის შეცვლა“ (Rename). რის შემდეგაც გააქტიურებული სვეტის სახელი ჩაასწოროს და დაამოწმოს კლავიატურაზე Enter ღილაკის გამოყენებით.

ასევე, უკვე არსებული მონაცემთა ცხრილის სვეტის სახელის შეცვლისათვის მომხმარებელს პირდაპირ აქვს შესაძლებლობა შესაბამისი ცხრილის სასურველი სვეტის სახელზე მაუსის მარცხენა ღილაკით გააქტიუროს სვეტის სახელი, ჩაასწოროს და დაამოწმოს კლავიატურაზე Enter ღილაკის გამოყენებით.



მონაცემთა ცხრილის ფორმატირება

52

მონაცემთა ცხრილის თვისებების ცვლილებების განხორციელებისათვის გამოიყენება **ALTER TABLE** ბრძანება, რომლის სინტაქსია:

```
ALTER TABLE [ მონაცემთა_ბაზის_სახელი.[ სქემის_სახელი. ] ] ცხრილის_სახელი
{
    ALTER COLUMN სვეტის_სახელი
    {
        [ ტიპის_სქემის_სახელი. ] ტიპის_სახელი
        [ COLLATE შეფარდების_სახელი ]
        [ NULL | NOT NULL ]
    }
    | ADD
    {
        სვეტის_სახელი ტიპის_სახელი
        [ CONSTRAINT შეზღუდვის_სახელი ]
        {
            DEFAULT შეზღუდვის_განსაზღვრა
            | PRIMARY KEY NONCLUSTERED ( სვეტის_სახელი [ ,...n ] )
            | UNIQUE ( სვეტის_სახელი [ ,...n ] )
        }
    }
}
```

(გაგრძელება შემდეგ სლაიდზე)

მონაცემთა ცხრილის ფორმატირება

53

```
| სვეტის_სახელი AS გამოთვლადი_სვეტის_განსაზღვრა
| [CONSTRAINT შეზღუდვის_სახელი]
{
    {PRIMARY KEY | UNIQUE | NONCLUSTERED} (სვეტის_სახელი [ASC|DESC] [...n])
    | FOREIGN KEY ( სვეტის_სახელი [ ....n ] ) REFERENCES
        მთავარი_ცხრილის_სახელი [ ( დაკავშირებული_სვეტის_სახელი [ ....n ] ) ]
    | CHECK ( ლოგიკური_გამოსახულება )
}
| INDEX ინდექსის_სახელი [ NONCLUSTERED ] ( სვეტის_სახელი [ASC | DESC] [...n] )
} [ ....n ]
| DROP
{
    CONSTRAINT [ IF EXISTS ]
        { შეზღუდვის_სახელი } [ ....n ]
    | INDEX [ IF EXISTS ]
        { ინდექსის_სახელი } [ ....n ]
    | COLUMN [ IF EXISTS ]
        { სვეტის_სახელი } [ ....n ]
}
} [ ....n ]
```

(გაგრძელება ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ცხრილის ფორმატირება

54

```
| [ WITH ] { CHECK | NOCHECK } CONSTRAINT { ALL | შეზღუდვის_სახელი [ ,...n ] }
| { ENABLE | DISABLE } TRIGGER { ALL | ტრიგერის_სახელი [ ,...n ] }
} [ ; ]
```

განვიხილოთ ამ კონსტრუქციის პარამეტრების დანიშნულება:

- ▶ ცხრილის_სახელი - ჩასასწორებელი ცხრილის სახელი;
- ▶ სვეტის_სახელი - სვეტის სახელი. აქვე უნდა ითქვას, რომ სვეტის სახელის ჩასწორება ხორციელდება მხოლოდ sp_rename სისტემური პროცედურის გამოყენებით, ხოლო ALTER COLUMN გამოიყენება სვეტის თვისებების კორექტირებისათვის;
- ▶ ტიპის_სახელი - სვეტის მონაცემთა ტიპის მითითება (თუ საჭიროა სქემის სახელის მითითებით);
- ▶ შეფარდების_სახელი - სვეტის ენის კოდირების შეფარდების სახელის მითითება;
- ▶ [NULL | NOT NULL] - სვეტისათვის **NULL** ან **NOT NULL** შეზღუდვის მითითება;
- ▶ შეზღუდვის_სახელი - დადებული შეზღუდვის სახელი;
- ▶ შეზღუდვის_განსაზღვრა - დადებული შეზღუდვის ტანის განსაზღვრა;

მონაცემთა ცხრილის ფორმატირება

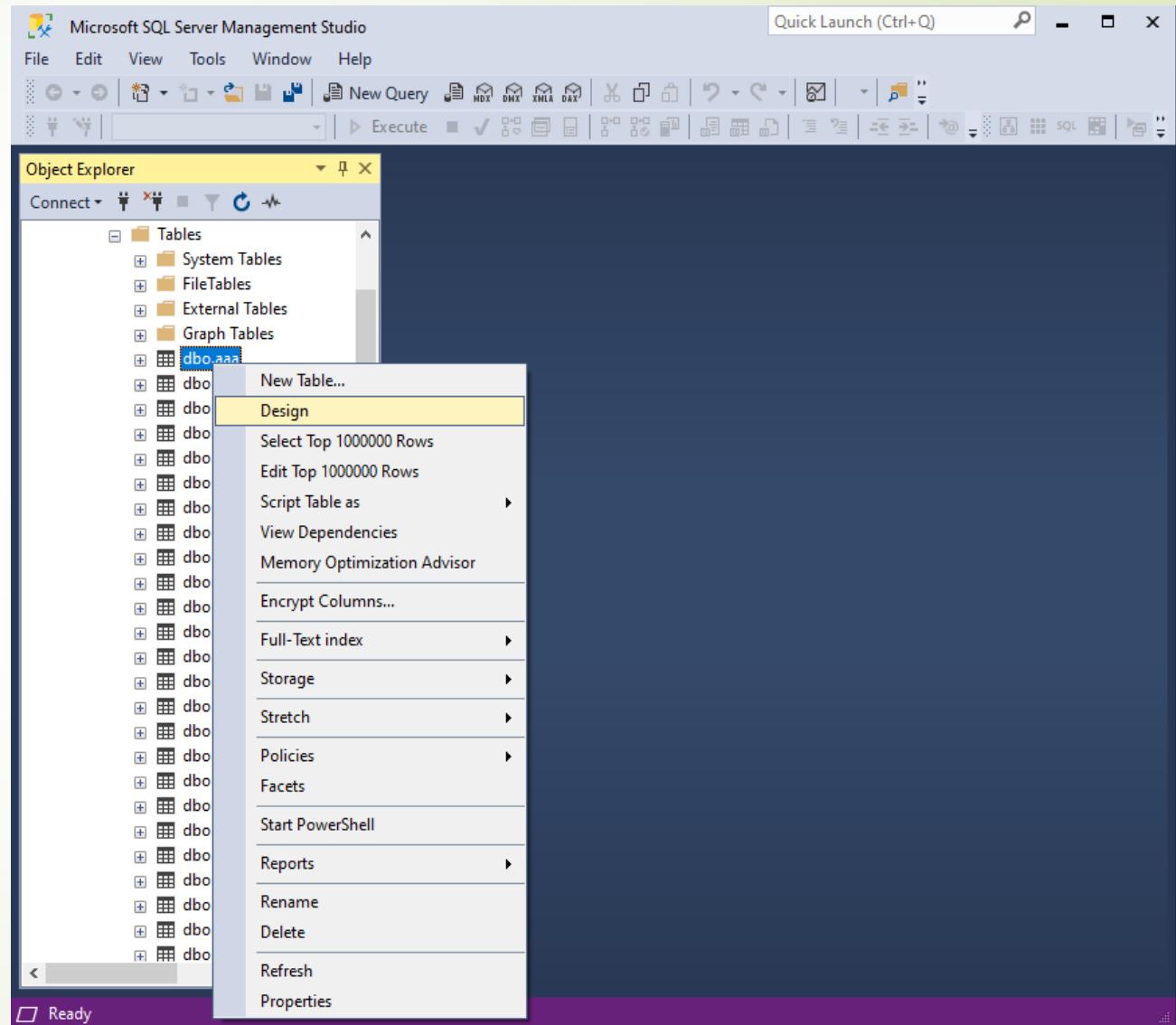
55

- ▶ გამოთვლადი_სვეტის_განსაზღვრა - დადებული გამოთვლადი სვეტის ფორმულის განსაზღვრა (წინ მითითებული სვეტის_სახელი-თვის);
- ▶ {PRIMARY KEY | UNIQUE | NONCLUSTERED} - პირველადი გასაღების, უნიკალურის ან არაკლასტერული სვეტის განსაზღვრა (შემდეგ მითითებული სვეტის_სახელი-თვის);
- ▶ FOREIGN KEY (სვეტის_სახელი [,...n]) - გარე გასაღების სვეტ(ებ)ის მითითება;
- ▶ REFERENCES მთავარი_ცხრილის_სახელი [(დაკავშირებული_სვეტის_სახელი [,...n])] - გარე გასაღებისათვის მთავარი ცხრილის სახელისა და საჭიროების შემთხვევაში დაკავშირებული სვეტ(ებ)ის სახელის მითითება;
- ▶ CHECK (ლოგიკური_გამოსახულება) - სვეტისათვის ლოგიკური გამოსახულებით შეზღუდვის დადება;
- ▶ INDEX ინდექსის_სახელი - ინდექსის მითითება სახელის დარქმევით;
- ▶ [IF EXISTS] - წაშლისას უთითებს პირობას „თუ არსებობს - წაშალე“;
- ▶ { CHECK | NOCHECK } CONSTRAINT { ALL | შეზღუდვის_სახელი [,...n] } - შეზღუდვების შემოწმება/არშემოწმების პირობის მითითება ყველა შეზღუდვისათვის ან მხოლოდ ჩამოთვლილთათვის;
- ▶ { ENABLE | DISABLE } TRIGGER { ALL | ტრიგერის_სახელი [,...n] } - ყველა ან ჩამოთვლილ ტრიგერისათვის ჩართვა/გამორთვის მითითება.

მონაცემთა ცხრილის ფორმატირება

56

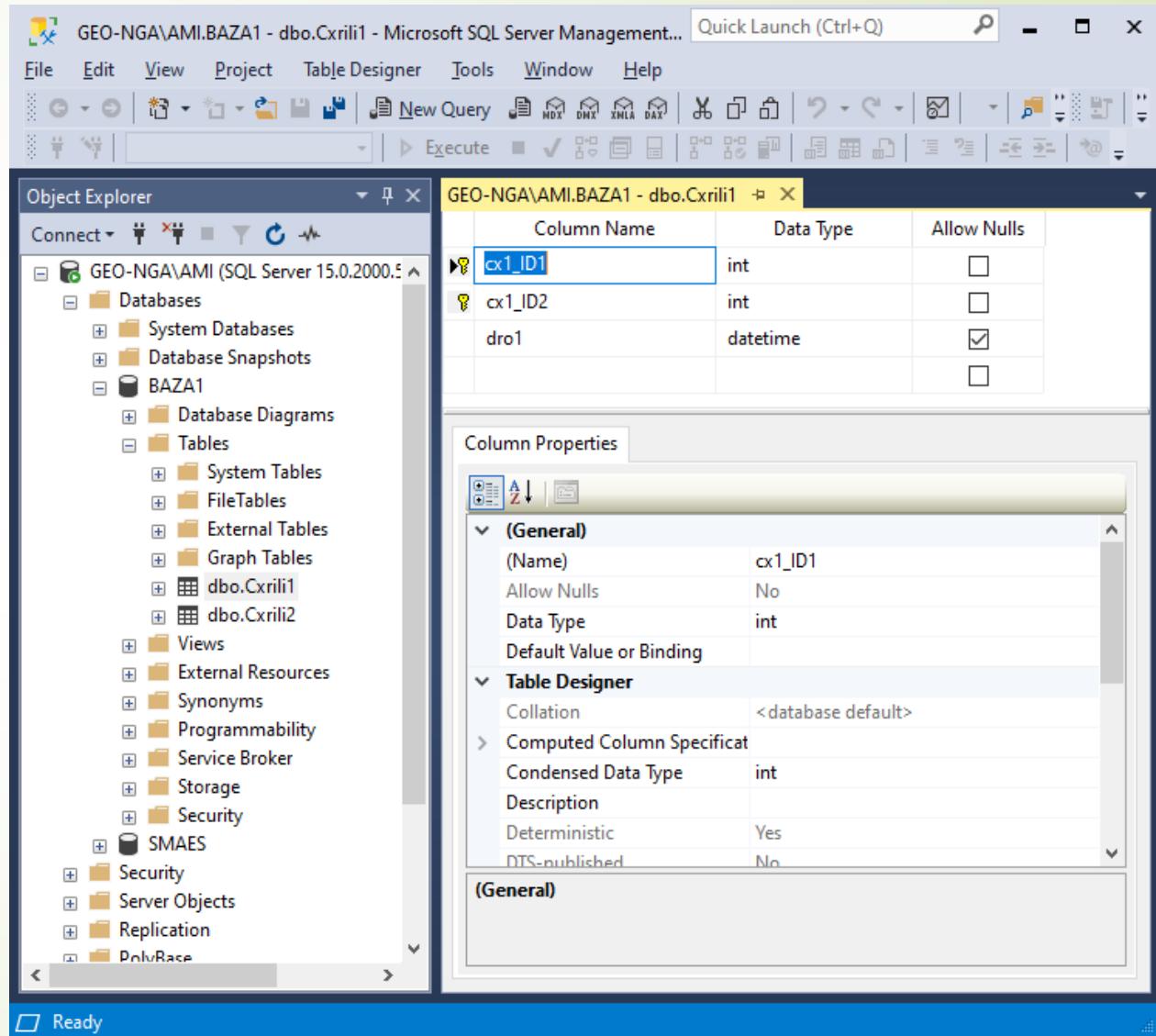
SQL Server Management Studio-ში (SSMS) მომხმარებლის მონაცემთა ბაზაში მონაცემთა ცხრილის ფორმატირებისათვის მომხმარებელმა უნდა გააკტიუროს მთავარი მენიუს მომხმარებლის მონაცემთა ბაზაში „ცხრილები“-ს (Tables) ჩანართში სასურველი ცხრილის სახელის სტრიქონი მაღსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს მეორე სტრიქონი „დიზაინი“ (Design).



მონაცემთა ცხრილის ფორმატირება

57

„დიზაინი“ (Design) სტრიქონის გაძტიურების შემდეგ SSMS-ის მარჯვენა მიდამოში გამოისახება შერჩეული ცხრილის სტრუქტურა და მომხმარებელს ეძლევა შესაძლებლობა განახორციელოს მისი ფორმატირება. აქ ფორმატირება სრულდება მონაცემთა ცხრილის ჩამატების ანალოგიურად, რომელიც უკვე განხილულია წინამდებარე სახელმძღვანელოში.



მონაცემთა ცხრილის წაშლა

58

ცხრილის წაშლელად გამოიყენება **DROP TABLE** ბრძანება, რომლის სინტაქსია:

DROP TABLE ცხრილის_სახელი

ცხრილის წაშლის უფლება გააჩნია SQL Server-ის **sysadmin** ან **db_dbadmin** სტანდარტული როლის წევრს და მონაცემთა ბაზის მფლობელს - **db_owner**-ს.

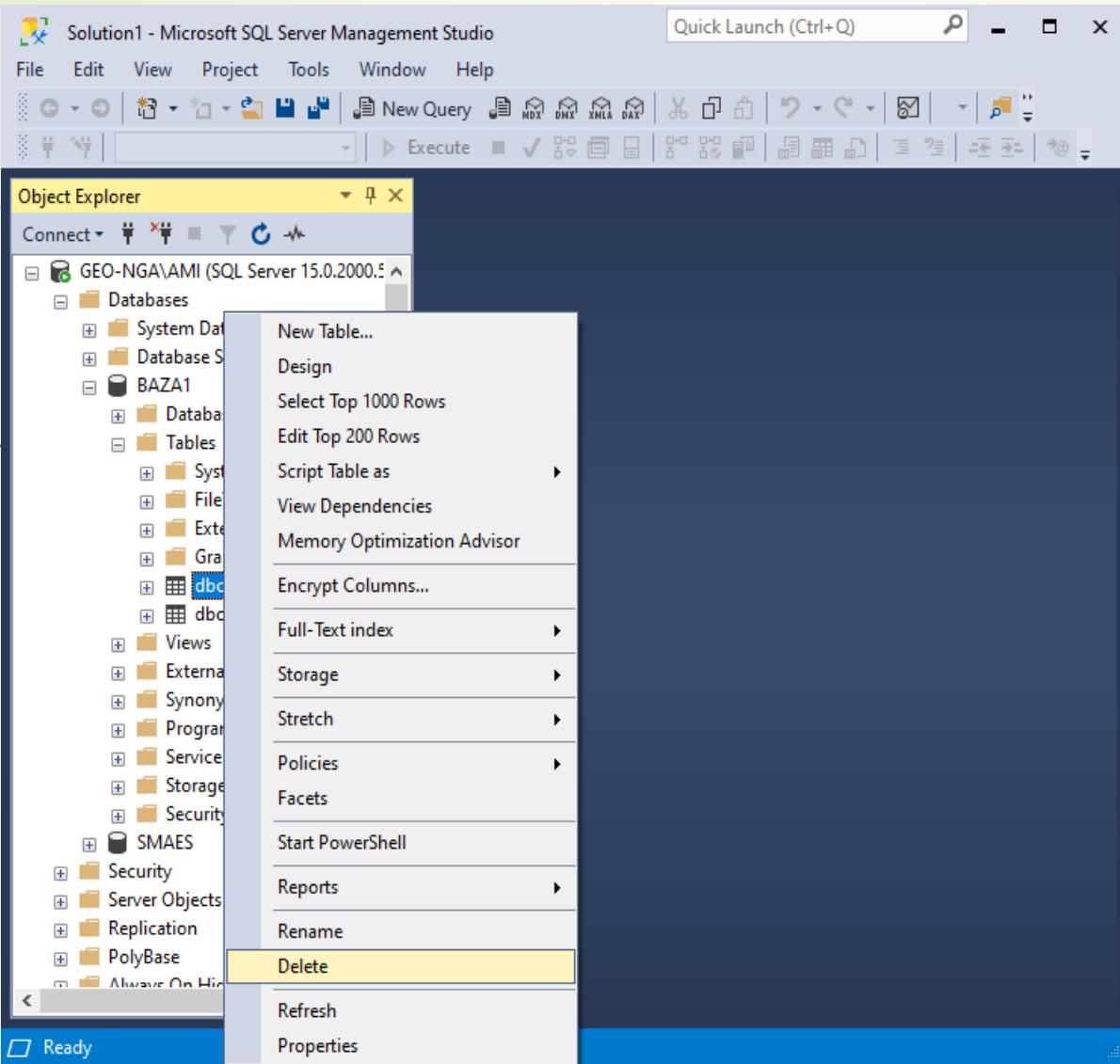
ცხრილის წაშლის წინ უნდა გაუქმდეს ყველა ცხრილთაშორისი კავშირი, რომელიც ამ ცხრილს მიმართავს გარე გასაღების საშუალებით და ყველა წარმოდგენა, რომელიც შეიცავს ამ ცხრილს. ე.ი. სანამ ამ ცხრილს წავშლიდეთ, საჭიროა მონაცემთა ბაზის ყველა იმ ობიექტის წაშლა/ცვლილება, რომლებიც წასაშლელ ცხრილს მიმართავენ ისე, რომ ისინი ამ ცხრილს აღარ მიმართავდნენ. **მაგალითი:**

```
USE Baza1;  
DROP TABLE Cxrili1;
```

The screenshot shows a SQL query window titled "SQLQuery11.sql - ...INQ5DSH\USER (51)*". It contains two lines of T-SQL code: "USE Baza1;" and "DROP TABLE Cxrili1;". Below the code, the "Messages" tab is selected, displaying the message "Commands completed successfully.".

მონაცემთა ცხრილის წაშლა

59



SQL Server Management Studio-ში (SSMS) მონაცემთა ცხრილის წაშლისათვის მომხმარებელმა უნდა გააკტიუროს მთავარ მენიუში საჭირო მონაცემთა ცხრილის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს ბოლოდან მესამე სტრიქონი „წაშლა“ (Delete).

ასევე შესაძლებელია მთავარ მენიუში საჭირო მონაცემთა ცხრილის სახელის სტრიქონი მაუსის მარცხენა ღილაკის გამოყენებით გააკტიურებისას გამოყენებულ იქნეს კლავიატურის ღილაკი **Delete**.

ინფორმაციის მიღება მონაცემთა ცხრილის შესახებ

60

ინფორმაცია მონაცემთა ბაზაში შექმნილი ობიექტების, მათ შორის, ცხრილების შესახებ, ინახება ამავე მონაცემთა ბაზის **sys.objects** სისტემურ წარმოდგენაში. კონკრეტული ცხრილის შესახებ ინფორმაციის მისაღებად გამოიყენება **OBJECTPROPERTY** ფუნქცია.

ძაგლითი:

მივიღოთ ინფორმაცია **Cxrili1** ცხრილის შესახებ:

USE Baza1;

SELECT * FROM sys.objects

WHERE object_id = object_id(N'[dbo].[Cxrili1]') AND OBJECTPROPERTY(object_id, N'IsUserTable') = 1;

შედეგი:

The screenshot shows a SQL query window titled "SQLQuery13.sql -...INQ5DSH\USER (54)*". The query is:

```
1 USE Baza1;
2 SELECT *
3   FROM sys.objects
4 WHERE object_id = object_id(N'dbo.Cxrili1') AND OBJECTPROPERTY(object_id, N'IsUserTable') = 1;
```

The results pane shows a table with one row:

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published
1	Cxrili1	949578421	NULL	1	0	U	USER_TABLE	2019-03-09 20:11:17.873	2019-03-09 20:13:55.433	0	0

მონაცემთა ცხრილის თვისებების ნახვა

61

ცხრილის თვისებების სანახავად გამოიყენება **sp_help** სისტემური შენახული პროცედურა, რომლის სინტაქსია:

sp_help [[@objname =] ცხრილის_სახელი]

მაგალითი:

მოთხოვნა გასცემს **Cxrili1** ცხრილის თვისებებს:

USE Baza1;

EXEC sp_help Cxrili1;

ან სრულად:

USE Baza1;

EXEC sp_help @objname = Cxrili1;

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

მონაცემთა ცხრილის თვისებების ნახვა

62

შედეგი:

The screenshot shows the SSMS interface with the following details:

- Query Window:** SQLQuery14.sql - ...INQ5DSH\USER (54)*
1 USE Baza1;
2 EXEC sp_help Cxrili1;
- Results Tab:** Displays the structure of the Cxrili1 table.

	Name	Owner	Type	Created_datetime
1	Cxrili1	dbo	user table	2019-03-09 20:11:17.873

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Sveti2	nchar	no	20			yes	(n/a)	(n/a)	SQL_Latin1_General_CI_AS

	Identity	Seed	Increment	Not For Replication
1	No identity column defined.	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	PK_Cxrili1	clustered, unique, primary key located on PRIMARY	id

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	PRIMARY KEY (clustered)	PK_Cxrili1	(n/a)	(n/a)	(n/a)	(n/a)	id

ინფორმაციის მიღება ცხრილზე დამოკიდებული ობიექტების შესახებ

ინფორმაციის მისაღებად მონაცემთა ცხრილზე დამოკიდებული ობიექტების შესახებ (ეს ინფორმაცია ძირითადად საჭიროა მონაცემთა ცხრილის წაშლის წინ) გამოიყენება **sp_depends** სისტემური შენახული პროცედურა, რომლის სინტაქსია:

```
sp_depends [ @objname = ] 'ობიექტის_სახელი'
```

მაგალითი:

გვაინტერესებს **ben** ცხრილზე დამოკიდებული ობიექტების სახელები და ტიპები, რომლის მოთხოვნას აქვს სახე:

```
USE SMAES;
```

```
EXEC sp_depends 'ben';
```

ან სრულად:

```
USE SMAES;
```

```
EXEC sp_depends @objname = 'ben';
```

(შედეგი ნაჩვენებია შემდეგ სლაიდზე)

ინფორმაციის მიღება ცხრილზე დამოკიდებული ობიექტების შესახებ

შედეგი:

The screenshot shows a SQL query window in SSMS with the following content:

```
SQLQuery15.sql - ...INQ5DSH\USER (54)*  DESKTOP-INQ5DSH...MAES - Diagram_0*
1 USE SMAES;
2 EXEC sp_depends 'ben';
```

The results pane displays a table with two columns: "name" and "type". The table contains 23 rows, each representing a view named "dbo.a_ben_xxx" where xxx is a two-digit number from 01 to 23.

	name	type
1	dbo.a_ben_0wre	view
2	dbo.a_ben_paspir	view
3	dbo.a_ben_soc	view
4	dbo.a_ben_soc_ucnobo	view
5	dbo.a_ben1_wre2	view
6	dbo.a_dan2	view
7	dbo.a_gany_wre_ped_ben	view
8	dbo.a_reg_sia	view
9	dbo.b_jgu_f	view
10	dbo.b_jgu_g	view
11	dbo.ben_cx_f	view
12	dbo.ben_cx_g	view
13	dbo.ben_f	view
14	dbo.ben_g	view
15	dbo.ben_raod	view
16	dbo.ben_soc_ucnobo	view
17	dbo.ben_tur	view
18	dbo.bug	view
19	dbo.bug_mon_all	view
20	dbo.ex_all	view
21	dbo.ex_meria	view
22	dbo.ex_wre_mon	view
23	dbo.export_all	view

ინფორმაციის მიღება ცხრილთაშორისი კავშირების შესახებ

65

ინფორმაცია იმის შესახებ, თუ მოცემული ცხრილი რომელ ცხრილებს უკავშირდება **PRIMARY KEY** და **FOREIGN KEY** შეზღუდვების საშუალებით, შეგვიძლია მივიღოთ **sp_fkeys** შენახული პროცედურის საშუალებით, რომლის სინტაქსია:

```
sp_fkeys [ @pktable_name = ] 'პგ_ცხრილის_სახელი'  
[ , [ @pktable_owner = ] 'პგ_ცხრილის_მფლობელი' ]  
[ , [ @pktable_qualifier = ] 'პგ_ბაზის_სახელი' ]  
[ , [ @fktable_name = 'გგ_ცხრილის_სახელი' ]  
[ , [ @fktable_owner = 'გგ_ცხრილის_მფლობელი' ]  
[ , [ @fktable_qualifier = 'გგ_ბაზის_სახელი' ]
```

განვიხილოთ პარამეტრების დანიშნულება:

- ▶ 'პგ_ცხრილის_სახელი' - პირველადი გასაღების შემცველი მთავარი ცხრილის სახელია, რომლის შესახებაც გვინდა ინფორმაციის მიღება;
- ▶ 'პგ_ცხრილის_მფლობელი' - მთავარი ცხრილის მფლობელის სახელია;
- ▶ 'პგ_ბაზის_სახელი' - მონაცემთა ბაზის სახელია, რომელიც შეიცავს მთავარ ცხრილს;
- ▶ 'გგ_ცხრილის_სახელი' - გარე გასაღებს შემცველი დამოკიდებული ცხრილის სახელია;
(გაგრძელება შემდეგ სლაიდზე)

ინფორმაციის მიღება ცხრილთაშორისი კავშირების შესახებ

66

- ▶ 'გგ_ცხრილის_მფლობელი' - დამოკიდებული ცხრილის მფლობელის სახელია;
- ▶ 'გგ_ბაზის_სახელი' - მონაცემთა ბაზის სახელია, რომელიც შეიცავს დამოკიდებულ ცხრილს.

თუ შევიტანთ მხოლოდ პირველადი გასაღების შემცველი ცხრილის პარამეტრებს, მაშინ გაიცემა მხოლოდ ინფორმაცია ამ ცხრილის პირველად გასაღებთან დაკავშირებული ყველა ცხრილის შესახებ, ხოლო თუ მივუთითებთ მხოლოდ გარე გასაღების შემცველი ცხრილის პარამეტრებს, მაშინ გაიცემა ინფორმაცია იმ ცხრილების შესახებ, რომლებიც დაკავშირებულია ამ ცხრილის გარე გასაღებთან.

განვიხილოთ მაგალითები:

ინფორმაციის მიღება ცხრილთაშორისი კავშირების შესახებ

67

ძაგლითი 1:

მივიღოთ ინფორმაცია **ben** მთავარ ცხრილთან დაკავშირებული ცხრილების შესახებ:

USE SMAES;

EXEC sp_fkeys @pktable_name = N'ben';

შედეგი:

The screenshot shows a SQL query window titled "SQLQuery16.sql - ...INQ5DSH\USER (56)*". The query consists of two lines:

```
1 USE SMAES;
2 EXEC sp_fkeys @pktable_name = N'ben';
```

The "Results" tab is selected, displaying the following table:

	PKTABLE_QUALIFIER	PKTABLE_OWNER	PKTABLE_NAME	PKCOLUMN_NAME	FKTABLE_QUALIFIER	FKTABLE_OWNER	FKTABLE_NAME	FKCOLUMN_NAME
1	SMAES	dbo	ben	b_id	SMAES	dbo	b_pp	b_id
2	SMAES	dbo	ben	b_id	SMAES	dbo	b_reg	b_id
3	SMAES	dbo	ben	b_id	SMAES	dbo	b_soc	b_id
4	SMAES	dbo	ben	b_id	SMAES	dbo	b_wre_mon	b_id

ინფორმაციის მიღება ცხრილთაშორისი კავშირების შესახებ

68

ძაგლითი 2:

მივიღოთ ინფორმაცია **pir_tan** დაკავშირებულ ცხრილთან მთავარი ცხრილების შესახებ:

USE SMAES;

EXEC sp_fkeys @fktable_name = N'pir_tan';

შედეგი:

The screenshot shows a SQL query window titled "SQLQuery17.sql - ...INQ5DSH\USER (56)*". The query is:

```
1 USE SMAES;
2 EXEC sp_fkeys @fktable_name = N'pir_tan';
```

The results tab displays a table with the following data:

	PKTABLE_QUALIFIER	PKTABLE_OWNER	PKTABLE_NAME	PKCOLUMN_NAME	FKTABLE_QUALIFIER	FKTABLE_OWNER	FKTABLE_NAME	FKCOLUMN_NAME
1	SMAES	dbo	gany	g_id	SMAES	dbo	pir_tan	g_id
2	SMAES	dbo	pir	p_id	SMAES	dbo	pir_tan	p_id
3	SMAES	dbo	pir_stat	ps_id	SMAES	dbo	pir_tan	ps_id
4	SMAES	dbo	tan	t_id	SMAES	dbo	pir_tan	t_id

ინფორმაციის მიღება ცხრილთაშორისი კავშირების შესახებ

69

მაგალითი 3:

მივიღოთ ინფორმაცია **ben** მთავარ ცხრილთან **b_reg** დაკავშირებული ცხრილის შესახებ:

USE SMAES;

EXEC sp_fkeys @pktable_name = N'ben', @fktable_name = N'b_reg';

შედეგი:

The screenshot shows a SQL Server Management Studio window with two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays the following data:

	PKTABLE_QUALIFIER	PKTABLE_OWNER	PKTABLE_NAME	PKCOLUMN_NAME	FKTABLE_QUALIFIER	FKTABLE_OWNER	FKTABLE_NAME	FKCOLUMN_NAME
1	SMAES	dbo	ben	b_id	SMAES	dbo	b_reg	b_id

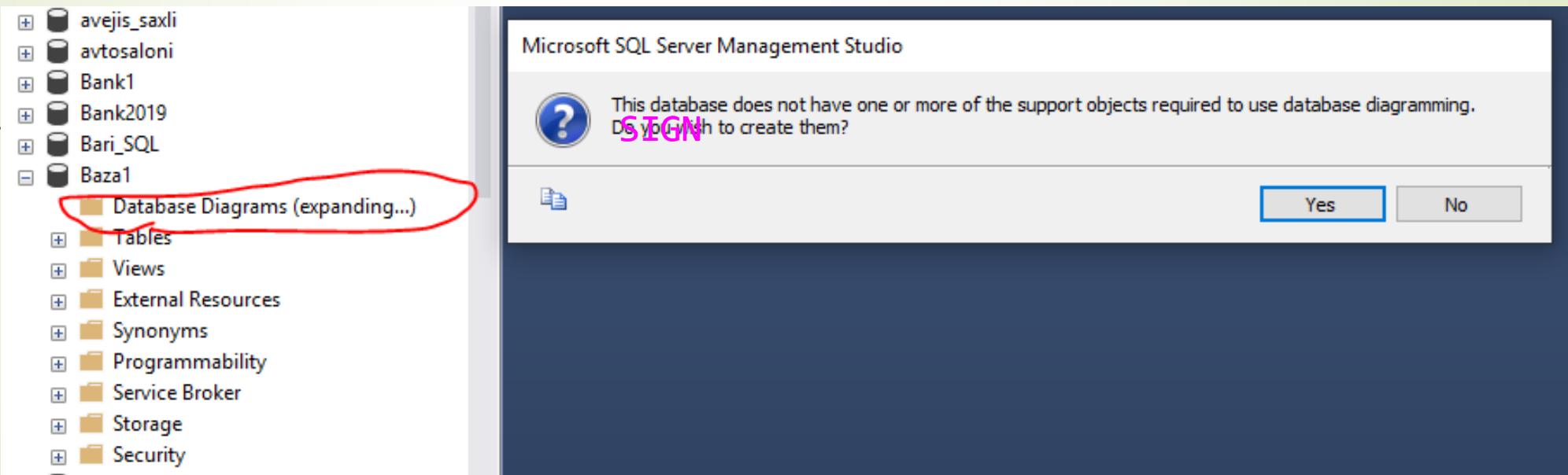
The query executed was:

```
1 USE SMAES;
2 EXEC sp_fkeys @pktable_name = N'ben', @fktable_name = N'b_reg';
```

დიაგრამები

1

მონაცემთა ბაზაში ერთმანეთთან დაკავშირებული ცხრილების სანახავად ან ცხრილთაშორისი კავშირების ფორმატირებისათვის გამოიყენება დიაგრამები. თუ მონაცემთა ბაზაში დიაგრამა ჯერ შექმნილი არ არის **Database Diagrams** სტრიქონის მაუსით გააქტიურებისას მონიტორის ეკრანზე გამოისახება ფანჯარა, რომელშიც სისტემა ეკითხება თანხმობას დიაგრამის შექმნაზე:



ასევე, შესაძლებელია დიაგრამის შექმნა ამ სტრიქონის მაუსის მარჯვენა ღილაკით გააქტიურებისას, გამოსახულ კონტექსტურ მენიუში **New Database Diagram** -ის შერჩევით.

დიაგრამები

2

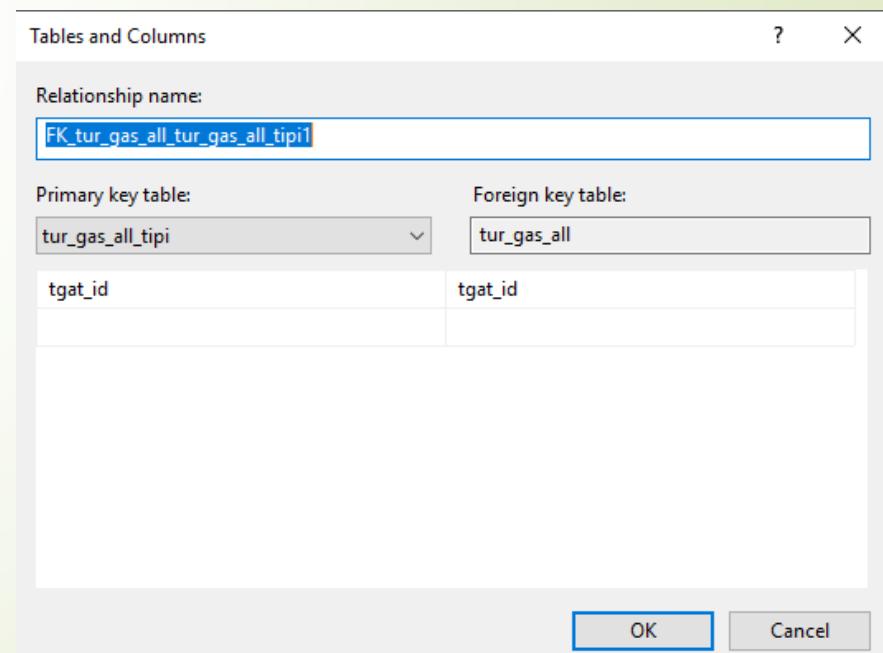
მონიტორის ეკრანის მარჯვენა მიღამოში გამოისახება ახალი ჩანართი, რომლის სახელი სრულდება ტექსტით – **Diagram_0** და **Add Table** ფანჯარა, რომელშიც არის მონაცემთა ბაზაში არსებული ყველა ცხრილის სია. მოვნიშნოთ დასაკავშირებელი ცხრილები და დავაჭიროთ ფანჯარაში არსებულ **Add** ღილაკს. ფანჯრის დახურვის შემდეგ გამოჩნდება ყველა მონიშნული ცხრილი სტრუქტურის გამოსახულებით. აქვე მომხმარებელს ეძლევა შესაძლებლობა მაუსის გამოყენებით განახორციელოს ცხრილებს შორის კავშირი.

ამისათვის მომხმარებელმა უნდა მაუსით დააჭიროს დამოკიდებული ცხრილის გარე გასაღების სვეტს და ხელაუღებლად გადაიტანოს მთავარი ცხრილის შესაბამის პირველად გასაღების სვეტზე და აუშვას მაუსის ღილაკს,

რის შემდეგაც მონიტორის ეკრანზე გამოისახება კავშირის მახასიათებლების ფანჯარა, რომელშიც გამოსახულია კავშირის დასახელება, პირველადი გასაღების ცხრილის სახელი და სვეტის სახელი და, ასევე, გარე გასაღების ცხრილის სახელი და სვეტის სახელი.

OK ღილაკის მაუსით გააქტიურების შედეგად გამოისახება შექმნილი კავშირის სპეციფიკაციები.

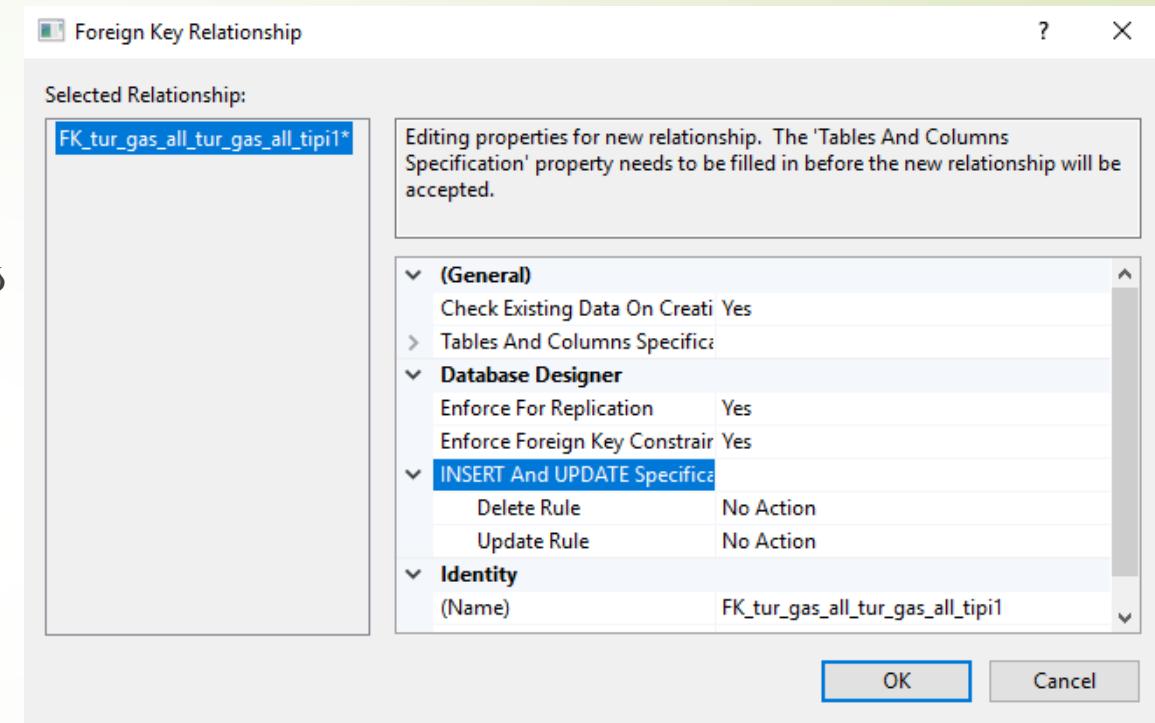
(გაგრძელება შემდეგ სლაიდზე)



მონიტორის ეკრანზე გამოსახულ
ფანჯარაში მომხმარებელს ეძღვა
შესაძლებლობა მიუთითოს ჩასწორების
და წაშლის წესები (რომლებიც უკვე
განხილულია). ფანჯარაში OK ღილაკის
მაუსით გააქტიურებისას, იგი
დაიხურება და მომხმარებელს ეძღვა
შესაძლებლობა განახორციელოს სხვა
კავშირების ფორმატირება.

ყველა კავშირის ფორმატირების
დასრულების შემდეგ დიაგრამის
შესანახად მაუსით უნდა გააქტიურდეს
დიაგრამის ქვემოდ გამოსახულ Save
ღილაკს, რის შედეგადაც გახსნილ ფანჯარაში მომხმარებელმა უნდა შეიტანოს დიაგრამის
სახელი და მაუსით გააქტიუროს მონიტორის ეკრანზე გამოსახული ღილაკი OK-ს.

შემდგომში დიაგრამის ფორმატირებისათვის მომხმარებელმა მაუსით უნდა გაააქტიუროს
Database Diagrams ქვესტრიქონი (dbo.Diagram_0 ან სხვა სახელი რომელიც დაარქვა
მომხმარებელმა დიაგრამას).



ინდექსები

4

ცხრილში მონაცემების ძებნის დაჩქარების მიზნით ინდექსები გამოიყენება, რომლებიც დაჩქარებას მათი ფიზიკური ან ლოგიკური მოწესრიგების ხარჯზე აღწევენ. ინდექსი წარმოადგენს მიმართვების ნაკრებს, რომლებიც მოწესრიგებულია (დალაგებულია ზრდადობის ან კლებადობის მიხედვით) გარკვეული სვეტ(ებ)ის მნიშვნელობების მიხედვით. ასეთ სვეტ(ებ)ს ინდექსირებული სვეტ(ებ)ი ეწოდება. ინდექსი მონაცემთა ბაზის დამოუკიდებელ ობიექტს წარმოადგენს და ის შეიძლება შედგებოდეს ცხრილის ერთ ან მეტ სვეტისაგან.

ფიზიკურად ინდექსი წარმოადგენს ინდექსირებული სვეტ(ებ)ის მონაცემების მოწესრიგებულ ნაკრებს ცხრილის სტრიქონების ფიზიკური განლაგების ადგილმდებარეობის მითითებით.

ინდექსის შექმნა ყოველთვის მიზანშეწონილია თუ ცხრილიდან მონაცემების ამოღება დიდ დროს იკავებს, მაგრამ აქ უნდა იქნეს გათვალისწინებული, რომ ინდექსების გამოყენებისას მონაცემების განახლების დროს ხორციელდება ყველა ინდექსის გაახლებაც. ამიტომ მომხმარებელმა უნდა გააკეთოს ვარირება მონაცემების ამოღებისას მნიშვნელოვანი დაჩქარებასა და მონაცემების განახლებისას (დამატების, ცვლილების და წაშლის) პროცესის შენელების შორის. ამიტომ ინდექსირების სვეტ(ებ)ის შერჩევისას უნდა გაანალიზდეს, თუ მოთხოვნების შესრულებისას ყველაზე ხშირად რომელი სვეტ(ებ)ი არის საკვანძო, ანუ რომლები განსაზღვრავენ მონაცემების ამოღების კრიტერიუმებს.

(გაგრძელება შემდეგ სლაიდზე)

ინდექსირების სვეტ(ებ)ის შერჩევისას, ასევე, უნდა გაანალიზდეს, რომ არ უნდა მოვახდინოთ ძალიან გრძელი სვეტების ინდექსირება, რომელთა სიგრძეა რამდენიმე ათეული სიმბოლო, რადგან ინდექსში შედის როგორც სტრიქონზე მიმართვა, ასევე თვით სვეტის შემცველობაც. უკიდურეს შემთხვევაში შესაძლებელია შეიქმნას გრძელი სვეტის შემოკლებული ვარიანტი, რომელშიც მოთავდება პირველ რამდენიმე (ათამდე) სიმბოლო, რომლის მიხედვითაც გაკეთდება ინდექსირება. ინდექსირებისათვის არ გამოიყენება ntext, text, varchar(max), nvarchar(max), varbinary(max), xml ან image ტიპის სვეტები.

არსებობს ინდექსების სამი ტიპი: არაკლასტერული, კლასტერული და უნიკალური.

არაკლასტერული ინდექსი (nonclustered index) შეიცავს სვეტ(ებ)ის მნიშვნელობებს (დალაგებულს ზრდადობით ან კლებადობით) და სტრიქონებზე მიმართვებს. ნაპოვნი მნიშვნელობის გასწვრივ მითითებულია შესაბამის სტრიქონზე მიმართვა, რომელიც შეიცავს სტრიქონის რეალურ მისამართს ცხრილში. ამ შემთხვევაში SQL Server-ი მიმართავს ინდექსს, ასრულებს საჭირო სტრიქონის ძებნას, ნახულობს მიმართვას და მის მიხედვით ასრულებს გადასვლას ცხრილში შესაბამისს სტრიქონზე. არაკლასტერული ინდექსის შექმნა უმჯობესია ხშირად ცვალებადი სვეტებისთვის. ერთ ცხრილში შეიძლება განისაზღვროს 999-მდე არაკლასტერული ინდექსი, ერთი არაკლასტერული ინდექსი შეიძლება შეიცავდეს ერთი ცხრილის 16-მდე სვეტს, ხოლო სვეტების მაქსიმალურად დასაშვები საერთო ზომაა 900 ბაიტი. ასეთ ინდექსს შედგენილი ინდექსი ეწოდება.

ინდექსები

6

კლასტერული ინდექსის (**clustered index**) გამოყენებისას ცხრილში ხორციელდება მონაცემების ფიზიკური განლაგების გადაწყობა ინდექსის სტრუქტურის შესაბამისად. ცხრილს გააჩნია მხოლოდ ერთი კლასტერული ინდექსი და იგი მნიშვნელოვნად ზრდის მონაცემების ძებნის წარმადობას. კლასტერულ ინდექსად უნდა ავირჩიოთ ყველაზე ხშირად გამოყენებადი სვეტ(ებ)ი. ასეთი სვეტი ერთი ან შეძლებისდაგვარად მცირე რაოდენობით უნდა იყოს.

კლასტერული ინდექსი არ გამოიყენება ხშირად ცვალებადი სვეტებისათვის, რადგან ყოველი ცვლილებისას SQL Server-ი ახორციელებს მონაცემების ფიზიკურ გადაადგილებას (მოწესრიგებას) ცხრილში. ცხრილში პირველადი გასაღების შექმნისას SQL Server-ი ავტომატურად ქმნის შესაბამისს კლასტერულ ინდექსს, თუ ის ადრე არ იყო შექმნილი ან გასაღების განსაზღვრისას არ იყო აშკარად მითითებული ინდექსის სხვა ტიპი.

თუ ცხრილში განსაზღვრულია როგორც კლასტერული, ასევე არაკლასტერული ინდექსები, მაშინ არაკლასტერული ინდექსის მიმთითებელი მიმართავს სტრიქონის არა ფიზიკურ მდებარეობას, არამედ კლასტერული ინდექსის შესაბამის ელემენტს. თუ არაკლასტერული ინდექსის შექმნისას კლასტერული ინდექსი არ არის უნიკალური, მაშინ SQL Server-ი ავტომატურად უმატებს მას დამატებით მნიშვნელობებს, რომლებიც მას უნიკალურს ხდის.

SQL Server-ის წარმადობა მკვეთრად არის დამოკიდებული კლასტერული ინდექსში შემავალი სვეტების რაოდენობასა და ზომაზე, რომელიც მინიმუმადე უნდა იქნეს დაყვანილი.

უნიკალური ინდექსი (**unique indexes**) იძლევა ინდექსირებულ სვეტში მნიშვნელობების უნიკალურობის გარანტიას, რომელიც შესაძლებელია იქნეს როგორც კლასტერულ, ასევე არაკლასტერულ ინდექსში. ერთ ცხრილს შეიძლება გააჩნდეს ერთი უნიკალური კლასტერული ინდექსი და რამდენიმე უნიკალური არაკლასტერული ინდექსი. მონაცემების მთლიანობის უზრუნველსაყოფად მიზანშეწონილია UNIQUE ან PRIMARY KEY შეზღუდვების და არა უნიკალური ინდექსის გამოყენება.

შევსების ფაქტორი (**fill factor**) განსაზღვრავს გვერდზე მონაცემების ჩაწერის სიმჭიდროვეს, ანუ საინდექსაციო გვერდების თავისუფალი სივრცის რამდენი პროცენტი იქნება შევსებული მონაცემებით, რომლის დარჩენილი თავისუფალი სივრცე თანდათანობით შეივსება ცხრილში მონაცემების დამატებისას.

შევსების ფაქტორის გამოყენებით ცხრილების ინდექსირება უფრო მოქნილად იმართება SQL Server-ზე: რაც მეტია შევსების ფაქტორი, მით ნაკლებია გვერდზე თავისუფალი სივრცე და მით უფრო კომპაქტურადაა ინფორმაცია ინდექსებზე. შევსების ფაქტორის არჩევისას უნდა შეფასდეს რამდენად ინტენსიურად სრულდება ცხრილში მონაცემების ცვლილება (დამატება, წაშლა და ცვლილება) - რაც უფრო ნაკლებად ცვლილებადია ცხრილი, მით უფრო ახლოს უნდა იქნეს 100%-თან და პირიქით - თუ ცხრილში მონაცემები ხშირად იცვლება, მაშინ შევსების ფაქტორი არ უნდა იყოს დიდი, როგორც მონაცემებისათვის, ისე ინდექსებისთვის.

ინდექსის შექმნა

8

ინდექსის შექმნის უფლება აქვს მხოლოდ ცხრილის მფლობელს. ინდექსი ყოველთვის იქმნება მხოლოდ მიმდინარე მონაცემთა ბაზის ცხრილისთვის ან წარმოდგენისთვის. არსებობს ინდექსის შექმნის რამდენიმე საშუალება:

- ▶ ინდექსის ავტომატურად შექმნა პირველადი გასაღების განსაზღვრისას;
- ▶ ინდექსის ავტომატურად შექმნა UNIQUE შეზღუდვის განსაზღვრისას;
- ▶ ინდექსის განსაზღვრა ცხრილის შექმნისას;
- ▶ ინდექსის შექმნა CREATE INDEX ბრძანებით.

განვიხილოთ ინდექსის შექმნა CREATE INDEX ბრძანების საშუალებით. მისი სინტაქსია:

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]

INDEX ინდექსის_სახელი ON { <ობიექტი> } (სვეტის_სახელი [ASC | DESC] [,...n])

[WITH

[[,] FILLFACTOR = შევსების_ფაქტორი]

[[,] IGNORE_DUP_KEY]

[[,] DROP_EXISTING]

(გაგრძელება შემდეგ სლაიდზე)

ინდექსის შექმნა

9

<ობიექტი> კონსტრუქციის სინტაქსია:

[მონაცემთა_ბაზის_სახელი. [სქემის_სახელი] .] ცხრილის_ან_წარმოდგენის_სახელი
განვიხილოთ არგუმენტების დანიშნულება:

UNIQUE არგუმენტის მითითებისას იქმნება უნიკალური ინდექსი, რომლის შექმნისას SQL Server-ი წინასწარ ამოწმებს სვეტის მნიშვნელობებს უნიკალურობაზე. თუ SQL Server-მა აღმოაჩინა სვეტში ორი ერთნაირი მნიშვნელობა, მაშინ ინდექსი არ შეიქმნება და გაიცემა შეტყობინება შეცდომის შესახებ (ამ შემთხვევაში SQL Server-ის ქცევა არ არის დამოკიდებული IGNORE_DUP_KEY საკვანძო სიტყვაზე). თუ ინდექსი შედგენილია (ანუ შეიცავს ორ ან მეტ სვეტს), მაშინ უნიკალური უნდა იყოს ყველა საინდექსირებელი სვეტების მნიშვნელობების ერთობლიობა ცხრილის ან წარმოდგენის თითოეულ სტრიქონში.

პრობლემების ასაცილებლად საინდექსებელ სვეტში სასურველია NULL მნიშვნელობის აკრძალვა, რადგან თუ სვეტში არის ორი NULL მნიშვნელობა, მაშინ SQL Server-ი მას ორ ერთნაირ მნიშვნელობად თვლის.

უნიკალური ინდექსის განსაზღვრის შემდეგ SQL Server-ი არ შეასრულებს ისეთ INSERT ან UPDATE ბრძანებას, რომელიც ორი ერთნაირი მნიშვნელობის არსებობას გამოიწვევს.

(გაგრძელება შემდეგ სლაიდზე)

ინდექსის შექმნა

10

CLUSTERED არგუმენტი მიუთითებს, რომ შესაქმნელი ინდექსი კლასტერული იქნება, ე.ი. ფიზიკურად მონაცემები განლაგებული იქნება ინდექსში განსაზღვრული რიგითობით.

NONCLUSTERED არგუმენტი მიუთითებს, რომ შესაქმნელი ინდექსი არაკლასტერული იქნება (თუ შექმნისას მითითებული არ არის შეთანხმებით ავტომატურად იგულისხმება NONCLUSTERED არგუმენტი).

ინდექსის_სახელი უნიკალური უნდა იყოს ცხრილის ფარგლებში.

სვეტის_სახელი განსაზღვრავს შესაქმნელ ინდექსში შემავალ სვეტ(ებ)ს. თუ მითითებულია ერთზე მეტი სვეტი, მაშინ ინდექსი იქნება შედგენილი (composite index).

ინდექსის შექმნისას საკვანძო ელემენტში მითითებული სვეტების მიმდევრობა გავლენას ახდენს მოთხოვნების შესრულების წარმადობაზე. მაქსიმალური წარმადობისათვის რეკომენდებულია თავდაპირველად მიეთითოს მინიმალური სიგრძის, შემდეგ კი - მაქსიმალური სიგრძის მქონე სახელები.

[**ASC** | **DESC**] არგუმენტი განსაზღვრავს სვეტის მნიშვნელობების დახარისხების მეთოდს: ASC შემთხვევაში სვეტის მნიშვნელობები დალაგდება ზრდადობის მიხედვით, DESC შემთხვევაში კი - კლებადობის მიხედვით (შეთანხმებით იგულისხმება ASC).

(გაგრძელება შემდეგ სლაიდზე)

ინდექსის შექმნა

11

FILLFACTOR = შევსების_ფაქტორის არგუმენტი განსაზღვრავს საინდექსო გვერდების შევსების ხარისხს. SQL Server-ი ავტომატურად არ უზრუნველყოფს შევსების ფაქტორის გარკვეულ მნიშვნელობას.

IGNORE_DUP_KEY არგუმენტი გამოიყენება როგორც კლასტერული, ასევე არაკლასტერული მხოლოდ უნიკალური ინდექსის შექმნის დროს. ის განსაზღვრავს SQL Server-ის ქცევას სტრიქონების დამატების ან ცვლილების ოპერაციის შესრულებისას, რომელიც იწვევს გამეორებადი მნიშვნელობების გაჩენას. თუ მითითებულია IGNORE_DUP_KEY არგუმენტი, მაშინ SQL Server-ი შეასრულებს ყველა ოპერაციებს, რომელიც იწვევს დუბლირებული მნიშვნელობების გაჩენას და არ შეასრულებს იმ ოპერაციებს, რომლებმაც დუბლირებას გამოიწვის და ამ შემთხვევაში SQL Server-ი გასცემს შეტყობინებას შეცდომის შესახებ. თუ IGNORE_DUP_KEY არგუმენტი არ იყო მითითებული, მაშინ SQL Server-ი, გასცემს შეტყობინებას შეცდომის შესახებ და აუქმებს ყველა ცვლილებას.

DROP_EXISTING გამოიწვევს (არსებობის შემთხვევაში) არსებული ინდექსის ახლით შეცვლას და მის გადაწყობას. ამ არგუმენტის გამოყენება უმჯობესია, ვიდრე ინდექსი ჯერ წავშალოთ და შემდეგ შევქმნათ. ეს არგუმენტი უნდა გამოვიყენოთ მაშინ, როცა მონაცემთა ბაზაში ცხრილისთვის ან წარმოდგენისთვის უკვე არსებობს ამავე სახელის მქონე ინდექსი, წინააღმდეგ შემთხვევაში გაიცემა შეტყობინება შეცდომის შესახებ.

ინდექსის შექმნა

12

ძაგლითი:

b_pn სვეტისთვის შევქმნათ Ind_pirN უნიკალური ინდექსი, რომლის შევსების ფაქტორია 50%. გამოვიყენოთ IGNORE_DUP_KEY არგუმენტი. პროგრამის კოდს აქვს სახე:

```
USE SMAES;
-- თუ ინდექსი არსებობს, მაშინ ის წაიშლება
IF EXISTS
(
    SELECT name
    FROM sys.indexes
    WHERE name = N'Ind_pirN'
)
DROP INDEX Ind_pirN ON ben
-- ინდექსის შექმნა
CREATE UNIQUE INDEX Ind_pirN
    ON ben (b_pn)
    WITH FILLFACTOR = 50, IGNORE_DUP_KEY;
```

(გაგრძელება შემდეგ სლაიდზე)

```
SQLQuery4.sql - D...INQ5DSH\USER (52)* 1 USE SMAES;
2 -- თუ ინდექსი არსებობს, მაშინ ის წაიშლება
3 IF EXISTS
4 (
5     SELECT name
6     FROM sys.indexes
7     WHERE name = N'Ind_pirN'
8 )
9 DROP INDEX Ind_pirN ON ben
10 -- ინდექსის შექმნა
11 CREATE UNIQUE INDEX Ind_pirN
12     ON ben (b_pn)
13     WITH FILLFACTOR = 50, IGNORE_DUP_KEY;
```

ინდექსის სახელის შეცვლა, ინდექსის წაშლა

ინდექსისთვის სახელის შესაცვლელად **sp_rename** შენახული პროცედურა გამოიყენება.

მაგალითი:

ben ცხრილში Ind_pirN ინდექსის გადავარქვათ სახელი Ind_pn -ით. კოდს აქვს სახე:

USE SMAES;

EXEC sp_rename 'ben.Ind_pirN', 'Ind_pn', 'INDEX';

ინდექსის წაშლელად **DROP INDEX** ბრძანება გამოიყენება. მისი სინტაქსია:

DROP INDEX ინდექსის_სახელი [,...n]

ერთი DROP INDEX ბრძანებით შესაძლებელია ცხრილში რამდენიმე ინდექსის წაშლა.

მაგალითი:

ben ცხრილში Ind_pn ინდექსი წავშალოთ. პროგრამულ კოდს აქვს სახე:

USE SMAES;

DROP INDEX Ind_pn ON ben

ინდექსის გადაწყობა

14

დროთა განმავლობაში ინდექსირებული ცხრილების შევსების ხარისხი მნიშვნელოვნად იცვლება, რაც აქვეითებს SQL Server-ის წარმადობას და საჭირო იქმნება ინდექსის გადაწყობა საინდექსო გვერდებზე თავისუფალი ადგილის მოწესრიგებით. გარდა ამისა, ინდექსის გადაწყობა შესაძლებელია საჭირო გახდეს ინდექსში შემავალი სვეტების ცვლილებისას.

ინდექსის გადაწყობის რამდენიმე გზა არსებობს. ერთი გზაა ინდექსის წაშლა და მისი ხელახლა შექმნა. თუ ასეთი გზით შევეცდებით კლასტერული ინდექსის გადაწყობას, მაშინ SQL Server-ს მოუწევს ყველა არაკლასტერული ინდექსის გადაწყობა, ეს კი დიდ დროს იკავებს. ამიტომ ეს არ არის ინდექსის გადაწყობის ეფექტური გზა.

ინდექსის გადაწყობის მეორე, უფრო ეფექტური გზაა CREATE INDEX ბრძანებაში DROP_EXISTING არგუმენტის მითითება. ამ შემთხვევაში არ მოხდება ყველა კლასტერული ინდექსის გადაწყობა.

ინდექსის გადაწყობის მესამე გზაა DBCC DBREINDEX ბრძანების გამოყენება. მისი სინტაქსია:

DBCC DBREINDEX

```
( 'მონაცემთა_ბაზის_სახელი.სქემის_სახელი.ცხრილის_სახელი',
[ , ინდექსის_სახელი [ , შევსების_ფაქტორი ] ] )
```

ინდექსის გადაწყობა

15

განვიხილოთ არგუმენტების დანიშნულება:

'მონაცემთა_ბაზის_სახელი.სქემის_სახელი.ცხრილის_სახელი' არის იმ ცხრილის სახელი, რომელშიც უნდა მოხდეს ერთი ან მეტი ინდექსის გადაწყობა.

ინდექსის_სახელი არის იმ ინდექსის სახელი, რომლის გადაწყობაც უნდა მოხდეს. თუ ინდექსის სახელი არ არის მითითებული, მაშინ მოხდება ცხრილის ყველა ინდექსის გადაწყობა.

შევსების_ფაქტორი განსაზღვრავს საინდექსო გვერდების შევსების ხარისხს. თუ ეს არგუმენტი არ არის მითითებული, მაშინ გამოიყენება შევსების ფაქტორის ის მნიშვნელობა, რომელიც გამოყენებული იყო ინდექსის შექმნისას.

მაგალითი 1:

ben ცხრილში Ind_pirN ინდექსის გადაწყობისას შევსების ფაქტორი = 70. კოდს აქვს სახე:

DBCC DBREINDEX ('SMAES.dbo.ben', Ind_pirN, 70);

მაგალითი 2:

ben ცხრილში ყველა ინდექსის გადაწყობა. კოდს აქვს სახე:

DBCC DBREINDEX ('SMAES..ben', "");

ინდექსის შესახებ ინფორმაციის მიღება

16

ინდექსის შექმნის წინ უმჯობესია მივიღოთ ინფორმაცია არსებული ინდექსების შესახებ. ინდექსების შესახებ ინფორმაციის მისაღებად გამოიყენება `sp_helpindex` შენახული პროცედურა, რომლის სინტაქსია:

`sp_helpindex [@objname =] 'ცხრილის_სახელი'`

'ცხრილის_სახელი' - მიმდინარე მონაცემთა ბაზის ცხრილის სახელია, რომლის ინდექსების შესახებ გვინდა ინფორმაციის მიღება.

მაგალითი:

ამოვიღოთ ინფორმაცია `ben` ცხრილის ინდექსების შესახებ, რომლის მოთხოვნას აქვს სახე:

`USE SMAES;`

`EXEC sp_helpindex 'ben';`

The screenshot shows a SQL query window titled "SQLQuery5.sql - D...INQ5DSH\USER (54)*". The query is:

```
1 USE SMAES;
2 EXEC sp_helpindex 'ben';
```

The results tab displays the following table:

	index_name	index_description	index_keys
1	ben_pn	nonclustered, unique located on PRIMARY	b_pn
2	Ind_pirN	nonclustered, ignore duplicate keys, unique locate...	b_pn
3	PK_mosw	clustered, unique, primary key located on PRIMARY	b_id

დაკავებული სივრცის შესახებ ინფორმაციის მიღება

17

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

მაგალითი:

მივიღოთ ინფორმაცია SMAES მონაცემთა ბაზის ინდექსების მიერ დაკავებული ადგილის შესახებ, რომლის მოთხოვნას აქვს სახე:

```
USE SMAES;
EXEC sp_spaceused;
```

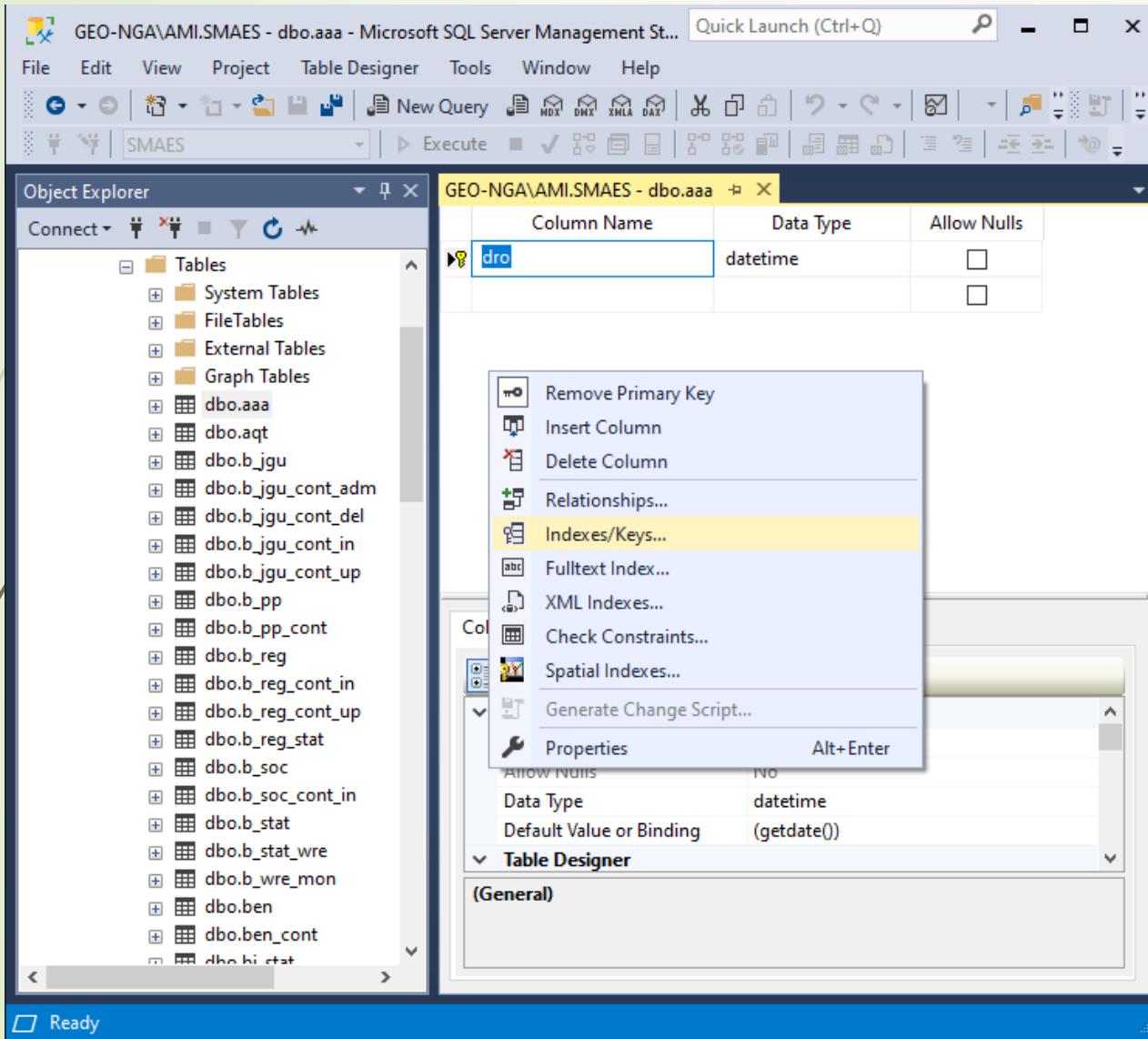
The screenshot shows a SQL query window titled "SQLQuery5.sql - D...INQ5DSH\USER (54)*". The query consists of two lines: "USE SMAES;" and "EXEC sp_spaceused;". The "Results" tab is selected, displaying two tables of data. The first table has columns "database_name", "database_size", and "unallocated space". It contains one row for the database "SMAES" with values "74.44 MB" and "1.02 MB". The second table has columns "reserved", "data", "index_size", and "unused". It also contains one row for the database "SMAES" with values "72872 KB", "56248 KB", "11888 KB", and "4736 KB".

	database_name	database_size	unallocated space
1	SMAES	74.44 MB	1.02 MB

	reserved	data	index_size	unused
1	72872 KB	56248 KB	11888 KB	4736 KB

ინდექსის ფორმატირება

18



SQL Server Management Studio-ში (SSMS) მონაცემთა ცხრილის ინდექსის ფორმატირებისათვის მომხმარებელმა უნდა გააკტიუროს მთავარ მენიუში საჭირო მონაცემთა ცხრილის სახელის სტრიქონი მაუსის მარჯვენა ღილაკის გამოყენებით და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს მეორე სტრიქონი „დიზაინი“ (Design) და მარჯვენა მიდამოში გამოჩენილ მონაცემთა ცხრილის სტრუქტურაზე მაუსის მარჯვენა ღილაკის გამოყენებით გამოსახულ კონტექსტურ მენიუში აირჩიოს მეხუთე სტრიქონი „ინდექსები/გასაღებები...“ (Indexes/Keys...) .

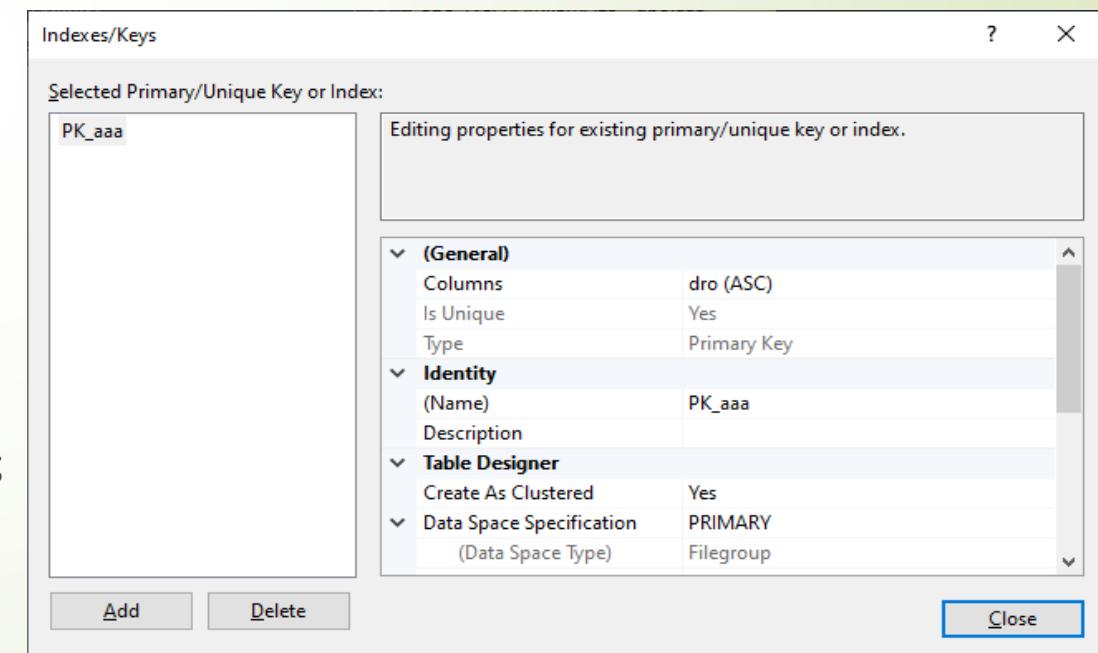
ინდექსის ფორმატირება

19

ახალ გამოსახულ ფანჯარაში მომხმარებელს ეძლევა შესაძლებლობა დაათვალიეროს და საჭიროების შემთხვევაში შეცვალოს ინდექსის პარამეტრები (თავისთავად, რომ ყველა ვერ შეიცვლება). ჩამოვთაღოთ ზოგიერთი მათგანი კონკრეტული მაგალითიდან, რომელშიც მოყვანილია პირველადი გასაღების ინდექსზე:

- ▶ General/Columns - სვეტის ზრდადობის(ASC)/კლებადობის(DESC) მიხედვით დაღავება;
- ▶ General/Is Unique - სვეტის უნიკალურობის განსაზღვრა;
- ▶ General/Type - სვეტის ინდექსის ტიპის განსაზღვრა;
- ▶ Identity/(Name) - ინდექსის სახელი;
- ▶ Identity/Description - აღწერა;
- ▶ Table Designer/Create As Clustered - ინდექსის კლასტერობის განსაზღვრა;
- ▶ ...

ინდექსის „დამატება“ (Add) და „წაშლა“ (Delete) შესაბამისი ღილაკებით.



ტრანზაქციები და ბლოკირებები

1

თუ ერთ მონაცემთა ბაზასთან ერთდროულად რამდენიმე მომხმარებელი ცდილობს ერთიდაიმავე მონაცემების წაკითხვას (მაგალითად, ერთიდაიგივე ცხრილიდან ერთიდაიმავე სტრიქონ(ებ)ის წაკითხვას), სიტუაცია არ წარმოადგენს პრობლემას, მაგრამ თუ მომხმარებლების ერთი ნაწილი ცდილობს მონაცემების ცვლილებას, მეორე კი - იგივე მონაცემების წაკითხვას, ამ შემთხვევაში მომხმარებლების მეორე ნაწილმა შეიძლება არასწორი მნიშვნელობები მიიღოს. ამ შემთხვევებში გამოიყენება ტრანზაქციები და ბლოკირებები.

ტრანზაქცია არის პროცესი, როცა სრულდება მასში შემავალი ყველა ბრძანება ან არც ერთი. ტრანზაქცია შედგება ერთი ან რამდენიმე ბრძანებისგან. თუ ყველა ბრძანება წარმატებით შესრულდა, მაშინ მოხდება ტრანზაქციის ფიქსირება (**Commit**), ანუ ცვლილებების დაფიქსირება. თუ ტრანზაქციის რომელიმე ბრძანება ვერ შესრულდა, მაშინ დაიწყება ტრანზაქციის უკუქცევა (**Rollback**), ანუ გაუქმდება ყველა ცვლილება და სისტემა აღდგება იმ მდგომარეობაში, რომელიც მას ჰქონდა ტრანზაქციის დაწყებამდე, რომელზეც ინფორმაცია ინახება ტრანზაქციების ჟურნალში.

SQL Server-ი ავტომატურად ახდენს იმ მონაცემების ბლოკირებას, რომლებთანაც ხორციელდება ტრანზაქცია. აქედან და SQL Server-ის მუშაობის ეფექტურობიდან გამომდინარე, რეკომენდირებულია მცირე ზომის ტრანზაქციების გამოყენება, რადგან მონაცემების ხანგრძლივი ბლოკირება იწვევს SQL Server-ის შეფერხებას.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციები და ბლოკირებები

2

ეს ნათლად ჩანს შემდეგ მაგალითში: თუ ერთი შენახული პროცედურა მონაცემების ცვლილებას ახორციელებს ერთ დიდ ტრანზაქციაში, მაშინ SQL Server-ი შეფერხდება დიდი ხნით, ხოლო თუ იგივე შენახული პროცედურა ისე დაიწერება, რომ მონაცემების ცვლილება განხორციელდება რამდენიმე, მცირე ზომის ტრანზაქციებში, ასეთი მოქმედება არ იქნება შესამჩნევი არც SQL Server-ის სწრაფმოქმედებისათვის და არც მომხმარებლებისათვის.

ბლოკირება (lock) არის მონაცემებზე დროებით დადებული შეზღუდვა მონაცემების დამუშავების ზოგიერთ ოპერაციის შესრულების დროს. შეიძლება დაიბლოკოს როგორც ცხრილის ერთი სტრიქონი, ისე მთელი მონაცემთა ბაზა. არსებობს სხვადასხვა სახის ბლოკირება, რომელსაც ბლოკირებების მენეჯერი (**lock manager**) მართავს. ტრანზაქციები და ბლოკირებები ერთმანეთთან მჭიდროდაა დაკავშირებული. SQL Server-ს აქვს ბლოკირების მართვის ეფექტური მექანიზმები, მაგრამ საჭიროებისამებრ მომხმარებელს შეუძლია ბლოკირების საკუთარი ალგორითმის განხორციელება.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

3

ტრანზაქციების მართვა

ტრანზაქცია განისაზღვრება შეერთების დონეზე და შეერთების დახურვის დროს ტრანზაქციაც ავტომატურად იხურება. არსებობს სამი სახის ტრანზაქცია:

1. აშკარა;
2. ავტომატური;
3. არააშკარა (ნაგულისხმევი).

ლოკალური ტრანზაქცია სრულდება ერთი მონაცემთა ბაზის შიგნით, ხოლო განაწილებული ტრანზაქცია მიმართავს რამდენიმე მონაცემთა ბაზას, რომლებიც შეიძლება ასევე სხვადასხვა SQL Server-ზე იყოს განთავსებული.

აშკარა ტრანზაქციები

აშკარა ტრანზაქციების (**explicit transaction**) გამოყენების შემთხვევაში აშკარად უნდა მიუთითოთ ტრანზაქციის დაწყება და დამთავრება. ამისთვის გამოიყენება შემდეგი ბრძანებები: **BEGIN TRANSACTION**, **COMMIT** და **ROLLBACK**.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

4

BEGIN TRANSACTION ბრძანება იწყებს ტრანზაქციას. ამ დროს ტრანზაქციების უურნალში ფიქსირდება შესაცვლელი მონაცემების საწყისი მნიშვნელობები და ტრანზაქციის დაწყების მომენტი. მისი სინტაქსია:

BEGIN TRAN[SACTION]

[ტრანზაქციის_სახელი | @tran_name_variable]

განვიხილოთ არგუმენტების დანიშნულება:

ტრანზაქციის_სახელი - არააუცილებელი არგუმენტი, რომლის სიგრძე არ უნდა აღემატებოდეს 32 სიმბოლოს და უნდა აკმაყოფილებდეს ობიექტის სახელდების წესებს.

@tran_name_variable - არააუცილებელი ლოკალური ცვლადია, რომელიც შეიცავს ტრანზაქციის სახელს და რომლის ტიპია CHAR, VARCHAR, NCHAR ან NVARCHAR. ჩადგმული ტრანზაქციების სახელდებისას უნდა გავითვალისწინოთ ის, რომ SQL Server-ზე რეგისტრირდება მხოლოდ პირველი (ზედა დონის) ტრანზაქციის სახელი. დანარჩენი ტრანზაქციების სახელები იქნება იგნორირებული.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

5

COMMIT TRANSACTION ან **COMMIT WORK** ბრძანებები ასრულებენ ტრანზაქციას. თუ ტრანზაქციის შესრულების დროს ადგილი არ ჰქონდა შეცდომებს, მაშინ შესრულებული ცვლილებები დაფიქსირდება (**roll forward**) და ტრანზაქციების ჟურნალში მოინიშნება, რომ ცვლილებები დაფიქსირებულია და ტრანზაქცია დასრულებული. მისი სინტაქსია:

COMMIT [WORK] |

COMMIT [TRAN [SACTION] [ტრანზაქციის_სახელი | @tran_name_variable]]

ROLLBACK TRANSACTION ან **ROLLBACK WORK** ბრძანებების შესრულება იწვევს ტრანზაქციის შეწყვეტას და უკუქცევას (**roll back**), რომლის დროსაც შესრულებული ცვლილებები უქმდება და აღდგება სისტემის პირვანდელი მდგომარეობა. ტრანზაქციების ჟურნალში მოინიშნება, რომ ტრანზაქცია იყო გაუქმებული. მისი სინტაქსია:

ROLLBACK [WORK] |

ROLLBACK [TRAN [SACTION] [ტრანზაქციის_სახელი | @tran_name_variable]]

ტრანზაქციის აშკარა განსაზღვრა საჭიროა მონაცემებს OLE DB და ADO მექანიზმებით მიმართვისას, ხოლო ODBC მექანიზმით მიმართვისას უნდა ინეს გამოყენებული ტრანზაქციის ავტომატური და არააშკარა სახე.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

6

@@ERROR ფუნქცია გასცემს ტრანზაქციის შესრულებისას SQL Server-ის მიერ დაფიქსირებულ შეცდომის კოდს. შეცდომის კოდი ნულის ტოლობა ნიშნავს, რომ ტრანზაქციის შესრულებისას შეცდომას ადგილი არ ჰქონია და ტრანზაქცია დაფიქსირდა, ხოლო ნულისაგან განსხვავებული რიცხვი ნიშნავს, რომ ტრანზაქციის შესრულების დროს იყო შეცდომა და ტრანზაქცია უკუიქცა, ანუ შესრულებული ცვლილებები გაუქმდა.

მაგალითი:

წავშალოთ ყველა ბენეფიციარის წრე, თუ მან 10 (კალენდარულ) დღეზე ნაკლები იარა.

USE SMAES;

DECLARE @Tran_10 NVARCHAR(20);

SET @Tran_10 = N'10დღიანები';

BEGIN TRANSACTION @Tran_10;

DELETE FROM b_jgu

WHERE DATEDIFF(DAY, bj_start, bj_fin) < 10 AND weli = N'2018-2019';

IF @@ERROR = 0

COMMIT TRANSACTION @Tran_10;

ELSE

ROLLBACK TRANSACTION @Tran_10;

(გაგრძელება შემდეგ სლაიდზე)

ავტომატური ტრანზაქციები

ავტომატურად დაწყების რეჟიმში (**autocommit transaction**) არის SQL Server-ის ჩვეულებრივი მუშაობის რეჟიმი: თითოეული ბრძანება განიხილება როგორც ცალკეული ტრანზაქცია. ბრძანების წარმატებით შესრულებისას ცვლილებები ფიქსირდება, ხოლო შეცდომის დაფიქსირებისას შესრულებული ცვლილებები უქმდება და სისტემა უბრუნდება საწყის მდგომარეობას. ტრანზაქციის ავტომატური განსაზღვრის რეჟიმის დასაყენებლად გამოიყენება ბრძანება:

```
SET IMPLICIT_TRANSACTION OFF;
```

არააშკარა ტრანზაქციები

ტრანზაქციის არააშკარა (ნაგულისხმევი) (**implicit transaction**) დაწყების რეჟიმში SQL Server-ი ყოველ ტრანზაქციის დასრულებისას ავტომატურად იწყებს ახალ ტრანზაქციას და იგი გრძელდება მომხმარებლის მიერ ტრანზაქციის უკუქცევის ან დაფიქსირების ბრძანების აშკარად მითითებამდე, რის შემდეგაც SQL Server-ი ავტომატურად იწყებს ახალ ტრანზაქციას.

ტრანზაქციის არააშკარა განსაზღვრის რეჟიმის დასაყენებლად გამოიყენება ბრძანება:

```
SET IMPLICIT_TRANSACTION ON
```

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

შეერთებისას ტრანზაქციის არააშკარა დაწყების რეჟიმის დაყენებისას, SQL Server-ი ავტომატურად ამთავრებს მიმდინარე ტრანზაქციას და იწყებს ახალს, თუ მას ხვდება რომელიმე შემდეგი ბრძანება:

- ▶ **ALTER TABLE** - ცხრილის სტრუქტურის ცვლილება;
- ▶ **CREATE** - მონაცემთა ბაზის ობიექტის შექმნა;
- ▶ **DELETE** - ცხრილიდან სტრიქონების წაშლა;
- ▶ **DROP** - მონაცემთა ბაზის ობიექტის წაშლა;
- ▶ **FETCH** - კურსორიდან მითითებული სვეტის მნიშვნელობის მიღება;
- ▶ **GRANT** - მონაცემთა ბაზის ობიექტთან მიმართვის ნების დართვა;
- ▶ **INSERT** - ცხრილში სტრიქონების დამატება;
- ▶ **OPEN** - კურსორის გახსნა;
- ▶ **REVOKE** - მონაცემთა ბაზის ობიექტებთან მიმართვის არააშკარა გადახრა;
- ▶ **SELECT** - ცხრილიდან მონაცემების ამოღება;
- ▶ **TRUNCATE TABLE** - ცხრილის ჩამოჭრა;
- ▶ **UPDATE** - ცხრილში მონაცემების ცვლილება.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

9

განაწილებული ტრანზაქცია (**distributed transaction**) გამოყენება სხვადასხვა მონაცემთა ბაზაში მოთავსებულ რესურსებთან მიმართვისას, რომელიც წარმოადგენს ცალკეულ მონაცემთა ბაზაში ლოკალურად შესრულებულ რამდენიმე ცალკეულ ტრანზაქციას. მისი სინტაქსია:

BEGIN DISTRIBUTED TRAN [SACTION] [ტრანზაქციის_სახელი | @tran_name_variable]

ჩადგმული ტრანზაქციები

ჩადგმული ტრანზაქციები (**nested transaction**) ეწოდება ტრანზაქციებს, რომელთა შესრულება ინიცირდება (იწყება) აქტიური ტრანზაქციის კოდიდან და ისინი გამოიყენებიან მხოლოდ აშკარა ტრანზაქციებში. ავტომატური ან არააშკარა ტრანზაქციების შესრულებისას, ახალი ტრანზაქციის დაწყებამდე წინა ტრანზაქცია უნდა იყოს დასრულებული. ჩადგმული ტრანზაქციების დანიშნულებაა იმ ტრანზაქციების უზრუნველყოფა, რომლებიც სრულდება ჩადგმული პროცედურების მიერ. ჩვენ შეგვიძლია მივმართოთ შენახულ პროცედურას როგორც უკვე დაწყებული ტრანზაქციიდან, ასევე უშუალოდ (არა ტრანზაქციის კოდიდან).

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

10

ძაგლითი:

შევქნათ დროებითი ცხრილი ორი სვეტით (INT PK, NVARCHAR(10)) შენახული პროცედურა ორი ჩადგმული ტრანზაქციით: პირველი ამატებს სტრიქონს ცვლადების მიხედვით, ხოლო მეორე პირველ სვეტს ზრდის 1-ით და მეორე სვეტში უმატებს „მეორე“-ს.

```
SET QUOTED_IDENTIFIER OFF;
SET NOCOUNT OFF;
USE SMAES;
CREATE TABLE #Cxrili
(
    Sveti_1 INT PRIMARY KEY, Sveti_2 NVARCHAR(10) NOT NULL
)
GO
CREATE PROCEDURE Proced @Sveti_1 INT, @Sveti_2 NVARCHAR(10)
AS
```

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

11

```
BEGIN TRANSACTION Trans_1;  
    INSERT INTO #Cxrili VALUES (@Sveti_1, @Sveti_2);  
    BEGIN TRANSACTION Trans_2;  
        INSERT INTO #Cxrili VALUES (@Sveti_1 + 1, @Sveti_2 + N' მეორე');  
    COMMIT TRANSACTION Trans_2;  
COMMIT TRANSACTION Trans_1;  
GO  
EXEC Proced 1, N'დათო';  
GO  
SELECT * FROM #Cxrili;
```

შედეგში ვღებულობთ ორი
სტრიქონისაგან და ორი სვეტისაგან
გამოსახულ დროებით ცხრილს:

(გაგრძელება შემდეგ სლაიდზე)

The screenshot shows a SQL Server Management Studio (SSMS) interface. The top bar indicates the file is named 'SQLQuery1.sql'. The code window contains the following T-SQL script:

```
1 SET QUOTED_IDENTIFIER OFF;  
2 SET NOCOUNT OFF;  
3 USE SMAES;  
4 CREATE TABLE #Cxrili  
5     (  
6         Sveti_1 INT PRIMARY KEY,  
7         Sveti_2 NVARCHAR(10) NOT NULL  
8     )  
9     GO  
10    CREATE PROCEDURE Proced @Sveti_1 INT, @Sveti_2 NVARCHAR(10)  
11    AS  
12        BEGIN TRANSACTION Trans_1;  
13            INSERT INTO #Cxrili VALUES (@Sveti_1, @Sveti_2);  
14        BEGIN TRANSACTION Trans_2;  
15            INSERT INTO #Cxrili VALUES (@Sveti_1 + 1, @Sveti_2 + N' მეორე');  
16        COMMIT TRANSACTION Trans_2;  
17        COMMIT TRANSACTION Trans_1;  
18        GO  
19        EXEC Proced 1, N'დათო';  
20        GO  
21        SELECT * FROM #Cxrili;
```

The 'Results' tab at the bottom displays the following data:

	Sveti_1	Sveti_2
1	1	დათო
2	2	დათო მეორე

ტრანზაქციების მართვა

12

ჩადგმული ტრანზაქციის შექმნა ხორციელდება ახალი ტრანზაქციის დაწყებით, წინა ტრანზაქციის დახურვის გარეშე. ზედა დონის ტრანზაქცია ვერ დასრულდება ჩადგმული ტრანზაქციის დასრულებამდე. ყველაზე დაბალი დონის ტრანზაქციის წარუმატებელი დასრულების და უკუქცევის შემთხვევაში ყველა ზედა დონის დაუფიქსირებელი ტრანზაქციაც უკუიქცევა. ყველა დონის ტრანზაქციის წარმატებით დასრულებისას ყველა ცვლილება ფიქსირდება პირველი დონის ტრანზაქციის დაფიქსირებისას.

ყოველი უსახელო COMMIT TRANSACTION ან COMMIT WORK ბრძანება მოქმედებს მხოლოდ უკანასკნელად დაწყებულ ტრანზაქციაზე. იმ შემთხვევაში თუ COMMIT TRANSACTION ბრძანებაში მითითებულია ტრანზაქციის_სახელი, რომელიც არ არის უკანასკნელად დაწყებული ტრანზაქცია, ამ შემთხვევაშიც ჯერ დასრულდება ყველა ბოლოს გახსნილი ტრანზაქცია რომელიც გაიხსნა ამ სახელის შემდეგ და მხოლოდ ამის შემდეგ დასრულდება მითითებული ტრანზაქციის_სახელი.

ROLLBACK TRANSACTION ან **ROLLBACK WORK** ბრძანება გამოიყენება ჩადგმულობის ნებისმიერ დონეზე ტრანზაქციის სახელის მითითების გარეშე და უკუაქცევს ყველა ზედა (პირველი დონის ჩათვლით) ჩადგმულ ტრანზაქციას, ხოლო თუ მითითებულია ტრანზაქციის_სახელი, უკუაქცევს ყველა ზედა დონის ტრანზაქციას ამ სახელის ჩათვლით.

SQL Server-ის **@@TRANCOUNT** ფუნქცია განკუთვნილია მოცემულ მომენტში აქტიურ შეერთების აქტიური ტრანზაქციების რაოდენობის განსაზღვრისათვის.

(გაგრძელება შემდეგ სლაიდზე)

ტრანზაქციების მართვა

13

ტრანზაქციებში აკრძალულია იმ ოპერაციების შესრულება, რომლებისთვისაც არაა რეალიზებული შესრულებული მოქმედებების უკუქცევა:

- ▶ **ALTER DATABASE** - მონაცემთა ბაზის კონფიგურაციის ცვლილება;
- ▶ **BACKUP LOG** - ტრანზაქციების ჟურნალის სარეზერვო ასლის შექმნა;
- ▶ **CREATE DATABASE** - მონაცემთა ბაზის შექმნა;
- ▶ **DROP DATABASE** - მონაცემთა ბაზის წაშლა;
- ▶ **LOAD DATABASE** - მონაცემთა ბაზის სარეზერვო ასლის ჩატვირთვა;
- ▶ **LOAD TRANSACTION** - ტრანზაქციების ჟურნალის სარეზერვო ასლის ჩატვირთვა;
- ▶ **RECONFIGURE** - sp_configure შენახული პროცედურის მიერ შესრულებული ცვლილებების გამოყენება;
- ▶ **RESTORE DATABASE** - სარეზერვო ასლიდან მონაცემთა ბაზის აღდგენა;
- ▶ **RESTORE LOG** - სარეზერვო ასლიდან ტრანზაქციების ჟურნალის აღდგენა;
- ▶ **UPDATE STATISTICS** - სტატისტიკის გაახლება.

(გაგრძელება შემდეგ სლაიდზე)

ბლოკირების მართვა

14

ბლოკირებების მენეჯერი (**lock manager**) განკუთვნილია ბლოკირებ(ებ)ის დაყენებისა და მოხსნისათვის კონფლიქტების გადაწყვეტისას. მომხმარებელს აქვს შესაძლებლობა მოთხოვნაში აშკარად მიუთითოს ბლოკირების საჭირო ტიპი.

ბლოკირების მენეჯერის ალგორითმი შემდეგია: სანამ ტრანზაქცია შეძლებს მონაცემების ცვლილებას, იგი ბლოკირებების მენეჯერისაგან ითხოვს შესაბამისი ბლოკირების დაწყებას. თუ მოთხოვნილი მონაცემები მოცემულ მომენტში გამოიყენება სხვა პროცესის მიერ, მაშინ ბლოკირებების მენეჯერი ან უარყოფს ბლოკირების მოთხოვნას, ან რიგში აყენებს ტრანზაქციას და მხოლოდ ყველა წინ რიგში მყოფი პროცეს(ებ)ის დასრულებისას და მონაცემების გათავისუფლებისას ბლოკავს მოთხოვნილ მონაცემებს ტრანზაქციისათვის. თუ მონაცემები (რესურსი) მოთხოვნის მომენტში თავისუფალია, მაშინ ბლოკირებების მენეჯერი აკმაყოფილებს მოთხოვნას და ბლოკავს მოთხოვნილ მონაცემებს, რის შემდეგაც ტრანზაქცია ამუშავებს მონაცემებს.

ბლოკირების მენეჯერის მიერ ლოდინის დრო შეგვიძლია ვარეგულიროთ ბრძანებით:

SET LOCK_TIMEOUT პერიოდი

სადაც პერიოდი მიუთითებს მიღიწამებს, რომლის განმავლობაშიც ტრანზაქცია დაელოდება რესურსის გათავისუფლებას.

(გაგრძელება შემდეგ სლაიდზე)

ბლოკირების მართვა

15

მითითებული პერიოდის დროის გავლის შემდეგ გაიცემა შეტყობინება შეცდომის შესახებ. მისი ნაგულისხმევი მნიშვნელობაა -1, რაც შეესაბამება უსასრულო ლოდინს. ამ არგუმენტის მიმდინარე მნიშვნელობას გასცემს **@@LOCK_TIMEOUT** ფუნქცია.

SET LOCK_TIMEOUT ბრძანება მოქმედებს მხოლოდ მიმდინარე შეერთების განმავლობაში. განმეორებითი შეერთების შემთხვევაში ხელახლა უნდა განისაზღვროს ლოდინის მაქსიმალური პერიოდი.

ბლოკირების სისტემა მოქმედებს მხოლოდ მრავალ მომხმარებიან გარემოში და იგი გამოირთვება ერთი მომხმარებლის რეჟიმში.

ბლოკირება იკავებს ოპერატიული მეხსიერების უბანს და პროცესორის დროს.

sp_configure განსაზღვრავს ბლოკირებების მაქსიმალურ რაოდენობას, რომლის სინტაქსია:

sp_configure ['locks', n]

n ბლოკირებების მაქსიმალური რაოდენობაა და იცვლება საზღვრებში $5000 \div 2147843647$. მისი ავტომატური (ნაგულისხმევი) მნიშვნელობაა 0, რაც ნიშნავს SQL Server-ის ავტომატურ კონფიგურირებას.

(გაგრძელება შემდეგ სლაიდზე)

ბლოკირების მართვა

16

sp_lock შენახული პროცედურა გამოიყენება ბლოკირებების მონიტორინგისთვის, რომლის საშუალებით შეგვიძლია მივიღოთ ინფორმაცია კონკრეტული პროცესის მიერ დაყენებული ბლოკირებების შესახებ. მისი სინტაქსია:

```
sp_lock [ [@spid1 = ] 'პროცესის_იდენტიფიკატორი_1'  
[ ,[@spid2 = ] 'პროცესის_იდენტიფიკატორი_2' ]
```

'პროცესის_იდენტიფიკატორი_1' არგუმენტი შეიცავს პროცესის საიდენტიფიკაციო ნომერს, რომლის შესახებაც გვინდა ინფორმაციის მიღება, ხოლო 'პროცესის_იდენტიფიკატორი_2' არგუმენტი დამატებითია და გამოიყენება ერთდღროულად ორი პროცესის შესახებ ინფორმაციის მისაღებად. თუ არგუმენტები არ არის მითითებული, მაშინ გაიცემა ინფორმაცია ყველა პროცესის მიერ დაყენებული ბლოკირების შესახებ.

KILL ბრძანება გამოიყენება პროცესის შესაწყვეტად. მისი სინტაქსია:

KILL პროცესის_იდენტიფიკატორი

master მონაცემთა ბაზის **sysprocesses** სისტემურ წარმოდგენაში ინახება SQL Server-ზე აქტიური პროცესების სია, რომელშიც არის დაწვრილებითი ინფორმაცია პროცესების შესახებ: უკანასკნელად შესრულებული ბრძანება, ტრანზაქციებისა და დაყენებული ბლოკირებების რაოდენობა, ლოდინის დრო, სტატუსი და ა.შ.

(გაგრძელება შემდეგ სლაიდზე)

შენახული პროცედურები

1

შენახული პროცედურა (**stored procedure**) უშუალოდ SQL Server-ზე ინახება თავისი სახელით და მონაცემთა ბაზის დამოუკიდებელ ობიექტს წარმოადგენს. თუ ხშირად საჭიროა ერთი და იგივე ოპერაცი(ებ)ის შესრულება, მაშინ მიზანშეწონილია მათი განთავსება შენახულ პროცედურაში.

შენახული პროცედურების გამოძახება შესაძლებელია კლიენტის პროგრამის, სხვა შენახული პროცედურის ან ტრიგერის მიერ, ხოლო მისი კოდის ცვლილების უფლება გააჩნია მხოლოდ მის მფლობელს ან მონაცემთა ბაზის db_owner ფიქსირებული როლის წევრს (თუმცა შენახული პროცედურის ფლობის უფლება შესაძლებელია გადაცემულ იქნეს ერთი მომხმარებლიდან მეორეს).

აღსანიშნავია, რომ შენახული პროცედურის შესრულების წინ SQL Server-ზე იქმნება მისთვის შესრულების გეგმა (**execution plan**), ოპტიმიზება და კომპილირება, რაც შენახული პროცედურის პირველი გამოძახების დროს საგრძნობლად ზრდის, მაგრამ ამ პროცედურის შემდგომში გამოძახებისას SQL Server-ი პირდაპირ მიმართავს ოპერატიულ მეხსიერებაში განთავსებულ შენახული პროცედურის შესრულების გეგმისა და კომპილირებული კოდის ქეშს და მაშინვე დაიწყებს ბრძანებების შესრულებას, რაც საგრძნობლად ზრდის შესრულების სიჩქარეს. ასევე აღსანიშნავია, რომ შენახული პროცედურები გამოყენებენ მოდულური პროექტირების პრინციპებს და, შესაბამისად, იძლევიან დიდი ამოცანების უფრო პატარა, დამოუკიდებელ და ადვილად სამართავ ამოცანებად დაყოფის საშუალებას.

შენახული პროცედურების ტიპები

2

- ▶ სისტემური შენახული პროცედურები (**System stored procedures**) გამოიყენება SQL Server-ის ადმინისტრირებისათვის, რომლებსაც აქვთ **sp_** პრეფიქსი (**system procedure**) და ისინი ინახება **master** სისტემურ მონაცემთა ბაზაში (თუმცა მათი გამოძახება შესაძლებელია ნებისმიერი მონაცემთა ბაზიდან. ამ პროცედურების მაგალითებია: სააღრიცხვო ჩანაწერების შექმნა, მონაცემთა ბაზების ობიექტების შესახებ ინფორმაციის მიღება, SQL Server-ისა და მონაცემთა ბაზის თვისებების მართვა, ავტომატიზებისა და რეპლიკაციის ქვესისტემის მართვა და ა.შ.);
- ▶ მომხმარებლის მიერ შექმნილი შენახული პროცედურები (**User-defined stored procedures**) ინახება შესაბამის მონაცემთა ბაზაში და წარმოადგენს მის ობიექტს;
- ▶ დროებითი შენახული პროცედურები (**Temporary stored procedures**) არსებობენ მხოლოდ გარკვეული დროის განმავლობაში, რის შემდგომაც ავტომატურად იშლებიან SQL Server-ის მიერ. არსებობს ორნაირი დროებითი პროცედურები: **ლოკალური დროებითი შენახული პროცედურები** (**Local temporary stored procedures**) შეიძლება გამოძახებულ იქნეს მხოლოდ მისი შექმნილ შეერთებიდან და სახელი # სიმბოლოთი იწყება და გლობალური დროებითი შენახული პროცედურები (**Global temporary stored procedures**) - შეიძლება გამოძახებულ იქნეს ნებისმიერი შეერთებიდან და მისი სახელი ## სიმბოლოებით იწყება. ორივე ინახება tempdb მონაცემთა ბაზაში და ავტომატურად იშლება მომხმარებლის გამორთვისას, SQL Server-ის გაჩერებისას ან გადატვირთვისას.

შენახული პროცედურის შექმნა

3

შენახული პროცედურის შექმნამდე უნდა იქნეს გადაწყვეტილი შემდეგი საკითხები:

- ▶ განისაზღვროს შენახული პროცედურის ტიპი (დროებითი იქნება თუ მომხმარებლის). ასევე შესაძლებელია შეიქმნას სისტემური შენახული პროცედურა, მისი სახელისათვის *sp_პრეფიქსის დამატების და მისი master სისტემურ მონაცემთა ბაზაში მოთავსებით;*
- ▶ დაიგეგმოს მიმართვის უფლებები, რადგან მას ექნება მონაცემთა ბაზის ობიექტებთან მიმართვის ისეთივე უფლებები, როგორიც აქვს მის შემქმნელ მომხმარებელს პროცედურის შექმნის დროს (და არა მისი შემსრულებელი მომხმარებლის). გარდა ამისა, შენახული პროცედურა მემკვიდრეობით იღებს ზოგიერთ პარამეტრს, რომელიც განსაზღვრული იყო მისი შექმნის დროს (მაგალითად, თუ პროცედურის შექმნისას ნებადართული იყო სისტემურ ცხრილებთან წვდომა, მაშინ პროცედურა ყოველთვის შეძლებს მას);
- ▶ განისაზღვროს შენახული პროცედურის პარამეტრები, რადგან შენახულ პროცედურას შეიძლება ჰქონდეს შესასვლელი და გამოსასვლელი პარამეტრები, რომელიც შესაძლებელია გამოყენებულ იქნეს ჩვეულებრივ ცვლადებად;
- ▶ შემუშავდეს შენახული პროცედურის კოდი, რადგან იგი შეიძლება შეიცავდეს ნებისმიერ ბრძანებების მიმდევრობას, სხვა შენახული პროცედურების გამოძახებების ჩათვლით.

შენახული პროცედურის შექმნა

4

CREATE PROCEDURE ბრძანება გამოიყენება შენახული პროცედურის შესაქმნელად, რომლის სინტაქსია:

```
CREATE PROC [ EDURE ] [სქემის_სახელი.] პროცედურის_სახელი  
[ { @პარამეტრის_სახელი პარამეტრის_ტიპი } [ VARYING ] [ = DEFAULT ] [ OUTPUT ] ] [...n]  
[ WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION } ]  
AS sql_ბრძანება [...n]
```

განვიხილოთ არგუმენტების დანიშნულება:

პროცედურის_სახელი შესაქმნელი პროცედურის სახელი, სადაც **sp_**, **#** და **##** პრეფიქსების გამოყენებისას, შესაბამისად შეიქმნება სისტემური, ლოკალური და გლობალური პროცედურები და იგი ყოველთვის იქმნება მიმდინარე მონაცემთა ბაზაში. დაუშვებელია მფლობელისა და მონაცემთა ბაზის სახელის შენახულ პროცედურისათვის დარქმევა;

@პარამეტრის_სახელი იმ პარამეტრის სახელია, რომელსაც შენახული პროცედურა გამოიყენებს შესასვლელი და გამოსასვლელი მონაცემების გადაცემისათვის. შენახული პროცედურის პარამეტრების სახელები **@** სიმბოლოთი უნდა იწყებოდეს და გამოიყოფა ერთმანეთისაგან მძიმეებით. რადგან პარამეტრები ლოკალურია, ამიტომ სხვადასხვა შენახულ პროცედურას შეიძლება გააჩნდეს ერთნაირი სახელის პარამეტრები. შენახული პროცედურის კოდში პარამეტრის სახელი ცვლადის სახელს არ უნდა ემთხვეოდეს;

(გაგრძელება შემდეგ სლაიდზე)

შენახული პროცედურის შექმნა

5

პარამეტრის_ტიპი შეიძლება ნებისმიერი ტიპის იყოს, მათ შორის მომხმარებლის მიერ განსაზღვრული, მაგრამ cursor ტიპი მხოლოდ გამოსასვლელი პარამეტრია;

OUTPUT არგუმენტი გამოსასვლელი პარამეტრია და შეგვიძლია გამოვიყენოთ შენახული პროცედურიდან მონაცემების დასაბრუნებლად, თუმცა იგი შესაძლებელია გამოყენებულ იქნეს როგორც შესასვლელიც. შენახული პროცედურიდან გამოსვლისას გამოსასვლელი პარამეტრის მიმდინარე მნიშვნელობა ენიჭება იმ ლოკალურ ცვლადს, რომელიც მითითებული იყო შენახული პროცედურის გამოძახების დროს;

VARYING არგუმენტი არის cursor ტიპის და გამოიყენება OUTPUT არგუმენტთან ერთად. იგი მიუთითებს, რომ გამოსასვლელი პარამეტრი იქნება სიმრავლე;

DEFAULT არგუმენტი მიუთითებს ნაგულისხმევ მნიშვნელობას, რომელიც პარამეტრს მიენიჭება, თუ მისი მნიშვნელობა მითითებული არ იქნება.

RECOMPILE არგუმენტი მიუთითებს, რომ შენახული პროცედურის ყოველი გამოძახებისას უნდა ხელახლა განხორციელდეს მისი შესრულების გეგმა და კოდის კომპილირება;

ENCRYPTION არგუმენტი გამოიყენება შენახული პროცედურის კოდის დასაშიფრად;

AS არგუმენტიდან იწყება შენახული პროცედურის ტანი (კოდი), რომელშიც დასაშვებია Transact_SQL-ის ყველა ბრძანება, ტრანზაქციების გამოცხადება და სხვა შენახული პროცედურების გამოძახება. გამოსასვლელად გამოიყენება **RETURN** ბრძანება.

შენახული პროცედურის შექმნა

6

ჩადგმული პროცედურები (nested procedure) არის ერთი შენახული პროცედურის ტანიდან სხვა შენახული პროცედურების გამოძახება. ჩადგმული პროცედურების რაოდენობა არ შეიძლება იყოს 32-ზე მეტი. ჩადგმულობის მიმდინარე დონის მისაღებად გამოიძახება @@NESTLEVEL ფუნქცია:

SELECT @@NESTLEVEL AS [ჩადგმულობის დონე]

შექმნილი შენახული პროცედურის სახელი შეიტანება ამავე მონაცემთა ბაზის sys.objects სისტემურ წარმოდგენაში. მის სანახავად შეგვიძლია შევასრულოთ მოთხოვნა:

SELECT name FROM sys.objects ORDER BY name

შენახული პროცედურის შექმნის მომენტში სრულდება ბრძანებების მხოლოდ სინტაქსური შემოწმება და არ მოწმდება მასში მითითებული ობიექტების არსებობა. მიმართვების სისწორის შემოწმება ხდება შენახული პროცედურის კომპილირების დროს, რაც უშუალოდ შენახული პროცედურის შესრულების წინ ხდება.

არსებობს შენახული პროცედურის შესრულების ორი გზა:

1. მხოლოდ მისი სახელის მითითება;
2. EXECUTE ბრძანების გამოყენება.

(გაგრძელება შემდეგ სლაიდზე)

შენახული პროცედურების გამოყენება

პირველ შემთხვევაში შენახული პროცედურის გამოძახება უნდა იყოს ერთადერთი ბრძანება შესასრულებლად გადასაცემ პაკეტში, ხოლო თუ არა, მაშინ **EXECUTE** უნდა იქნეს გამოძახებული, რომელიც ესევე შესაძლებელია გამოყენებულ იქნეს სხვა შენახული პროცედურიდან ან ტრიგერიდან გამოძახებისას, რომლის სინტაქსია:

[**EXEC [UTE]**] პროცედურის_სახელი

[[@პარამეტრის_სახელი =] { სიდიდე | @ცვლადის_სახელი } [**OUTPUT**] | [**DEFAULT**]] [,...n]

OUTPUT არგუმენტის გამოყენება ნებადართულია იმ შემთხვევაში, როცა შესაბამისი პარამეტრი გამოცხადებული იყო **OUTPUT** სიტყვის გამოყენებით შენახული პროცედურის შექმნის დროს. ანალოგიურად, პარამეტრს ენიჭება **DEFAULT** არგუმენტის მნიშვნელობა და იგი გამოიყენება, თუ შესაბამისი პარამეტრისათვის განსაზღვრული იყო ნაგულისხმევი მნიშვნელობა შენახული პროცედურის შექმნისას.

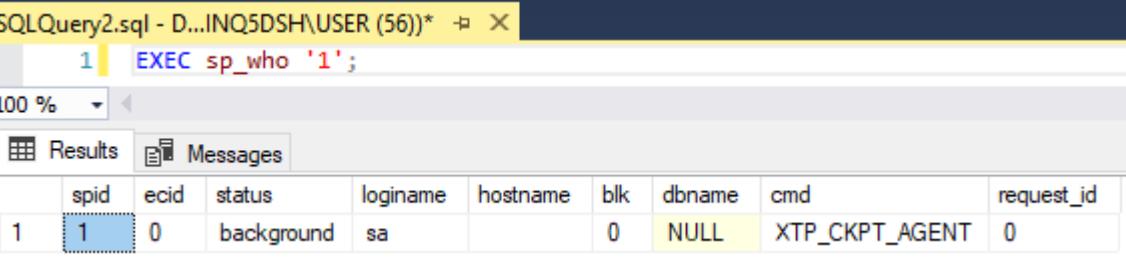
შენახული პროცედურის გამოძახებისას პარამეტრების სახელები შეგვიძლია არ მივუთითოთ. ასეთ შემთხვევაში, პარამეტრების მნიშვნელობები ისეთივე მიმდევრობით უნდა მივუთითოთ, როგორც იყო ისინი ჩამოთვლილი შენახული პროცედურის შექმნის დროს. შენახული პროცედურის გამოძახებისას შეგვიძლია მივუთითოთ პარამეტრის სახელი მნიშვნელობასთან ერთად ან მხოლოდ მნიშვნელობა სახელის გარეშე. ბოლო პარამეტრები შეგვიძლია გამოვტოვოთ, თუ მათთვის განსაზღვრულია ნაგულისხმევი მნიშვნელობა.

შენახული პროცედურების გამოყენება

8

sp_who შენახული პროცედურა გამოიყენება სისტემური პროცესის შესახებ საერთო ინფორმაციის მისაღებად:

EXEC sp_who '1';



The screenshot shows a SQL Server Management Studio window titled "SQLQuery2.sql - D...INQ5DSH\USER (56)*". It contains a single query line: "1 EXEC sp_who '1';". Below the query is a results grid. The grid has columns: spid, ecid, status, loginname, hostname, blk, dbname, cmd, and request_id. There is one row of data: spid is 1, ecid is 0, status is "background", loginname is "sa", hostname is blank, blk is 0, dbname is "NULL", cmd is "XTP_CKPT_AGENT", and request_id is 0.

	spid	ecid	status	loginname	hostname	blk	dbname	cmd	request_id
1	1	0	background	sa		0	NULL	XTP_CKPT_AGENT	0

შენახული პროცედურების გამოყენება

9

ძაგლითი 1:

შენახული პროცედურის პარამეტრი შეიცავს შაბლონს, რომლის მიხედვით მოხდება gva სვეტში მნიშვნელობების ძებნა:

```
USE SMAES;
GO
CREATE PROCEDURE Ben_Proc_1 @gvari NVARCHAR(25) = N'χ%'
AS
SELECT b_pn, b_gva, b_sax
FROM ben
WHERE b_gva LIKE @gvari;
```

The screenshot shows a SQL query window in SSMS. The code is identical to the one above, creating a stored procedure named Ben_Proc_1 with a parameter @gvari of type NVARCHAR(25) set to a default value of N'χ%'. The procedure selects b_pn, b_gva, and b_sax from the ben table where b_gva matches the parameter. The execution message at the bottom says 'Commands completed successfully.'

```
SQLQuery3.sql - D...INQ5DSH\USER (58)*          SQLQuery2.sql - D...INQ5DSH\USER (5
1 USE SMAES;
2 GO
3 CREATE PROCEDURE Ben_Proc_1 @gvari NVARCHAR(25) = N'χ%'
4 AS
5 SELECT b_pn, b_gva, b_sax
6     FROM ben
7 WHERE b_gva LIKE @gvari;

100 % < 
Messages
Commands completed successfully.
```

(გაგრძელება შემდეგ სლაიდზე)

შენახული პროცედურების გამოყენება

10

შექმნილი პროცედურის გამოძახების ცვლილებისას, შესაძლებელია სხვადასხვა შედეგების მიღება:

1. გამოვიძახოთ პროცედურა პარამეტრის გარეშე.

EXEC Ben_Proc_1;

ამ შემთხვევაში პარამეტრად გამოყენებულ იქნება პროცედურის შექმნისას მითითებული პარამეტრი **N'χ%** და გამოისახება შესაბამისი შედეგი (ანუ გვარი იწყება ჯ-ზე):

2. გამოვიძახოთ პროცედურა პარამეტრით **N'[ად]%**

EXEC Ben_Proc_1 N'[ად]%;

ამ შემთხვევაში პარამეტრად გამოყენებულ იქნება ახლად მითითებული პარამეტრი **N'[ად]%** და გამოისახება შესაბამისი შედეგი (ანუ გვარი იწყება ა-ზე ან დ-ზე):

The screenshot shows the SQL Query window with the following code:

```
USE SMAES;
EXEC Ben_Proc_1;
```

The Results tab displays the following data:

b_pn	b_gva	b_sax
01019068345	ჯორჯაძე	ირაკლი
01019069654	ჯინორია	გიორგი
01019071638	ჯანდერი	გიორგი
01019072708	ჯორჯი	ნინო

The screenshot shows the SQL Query window with the following code:

```
USE SMAES;
EXEC Ben_Proc_1 N'[ად]%;
```

The Results tab displays the following data:

b_pn	b_gva	b_sax
01019045843	დათუნაშვილი	ნატო
01019060619	ალიუნაშვილი	დაჩი
01019066391	ახობაძე	ნინო
01019067976	დავითიანი	მარია

შენახული პროცედურების გამოყენება

11

ძაგლითი 2:

შეიქმნას პროცედურა, რომელიც მოგვცემს მიმართულებისა და აქტიურობის მიხედვით ბენეფიციარების სის:

```
USE SMAES;
GO
CREATE PROCEDURE Ben_Sia_1 @mimart INT = 1, @aqt BIT = 1
AS
SELECT b.b_gva, b.b_sax
    FROM b_jgu bj LEFT JOIN
        ben b ON bj.b_id = b.b_id LEFT JOIN
        wr ON wr.w_id = bj.w_id LEFT JOIN
        gany g ON g.g_id = wr.g_id
WHERE bj.weli = N'2018-2019' AND
      g.g_id = @mimart AND bj.bj_aqt = @aqt;
```

The screenshot shows the SQL Server Management Studio interface with two panes. The left pane displays the T-SQL code for creating the stored procedure Ben_Sia_1. The right pane shows the results of the execution, indicating that the command was completed successfully.

```
SQLQuery4.sql - D...INQ5DSH\USER (54)*          SQLQuery3.sql - D...INQ5DSH\USER (58)
2 GO
3 CREATE PROCEDURE Ben_Sia_1 @mimart INT = 1, @aqt BIT = 1
4 AS
5 SELECT b.b_gva, b.b_sax
6     FROM b_jgu bj LEFT JOIN
7         ben b ON bj.b_id = b.b_id LEFT JOIN
8             wr ON wr.w_id = bj.w_id LEFT JOIN
9                 gany g ON g.g_id = wr.g_id
10 WHERE bj.weli = N'2018-2019' AND
11       g.g_id = @mimart AND bj.bj_aqt = @aqt;
100 %
Messages
Commands completed successfully.
```

(გაგრძელება შემდეგ სლაიდზე)

შენახული პროცედურების გამოყენება

12

შექმნილი პროცედურის გამოძახების ცვლილებისას, შესაძლებელია სხვადასხვა შედეგების მიღება:

1. გამოვიძახოთ პროცედურა პარამეტრებით 25 და 1:

EXEC Ben_Sia_1 25, 1;

ამ შემთხვევაში პარამეტრად გამოყენებულ იქნება
მიმართულების კოდი = 25 და ბენეფიციარის აქტიურობა
= 1 (ანუ True). შედეგი შესაბამისად იქნება ამ
მიმართულებაზე ამჟამად აქტიური ბენეფიციარები:

SQLQuery3.sql - D...INQ5DSH\USER (58)*	
1	USE SMAES;
2	EXEC Ben_Sia_1 25, 1;
100 %	
Results	Messages
b_gva	b_sax
1 ბოგაძე	ანასტასია
2 მეგდომვილი	ნინო
3 ბულუაშვილი	ნიკა
4 ნაჭეუბია	ნიკოლოზი
5 ჯაბლაძეონია	ჯორგი

2. გამოვიძახოთ პროცედურა პარამეტრებით 25 და 0:

EXEC Ben_Sia_1 25, 0;

ამ შემთხვევაში პარამეტრად გამოყენებულ იქნება
მიმართულების კოდი = 25 და ბენეფიციარის აქტიურობა
= 0 (ანუ False). შედეგი შესაბამისად იქნება ამ
მიმართულებაზე ამჟამად არააქტიური (ანუ ვინც უკვე
დაანება ამ მიმართულებაზე სწავლას) ბენეფიციარები:

SQLQuery3.sql - D...INQ5DSH\USER (58)*	
1	USE SMAES;
2	EXEC Ben_Sia_1 25, 0;
100 %	
Results	Messages
b_gva	b_sax
1 რაჭელიმვილი	ელენე
2 მეგრელიმვილი	ნინო
3 ბურგენიძე	თამანა
4 სექანია	მარიამი
5 კოტრიკაძე	ანა

შენახული პროცედურის შესახებ ინფორმაციის მიღება, შენახული პროცედურის სახელის ცვლილება

შენახული პროცედურის შესახებ ინფორმაციის მიღება

ნებისმიერი ობიექტის შესახებ, მათ შორის შენახული პროცედურის შესახებ ინფორმაციის მისაღებად, უნდა გამოვიყენოთ `sp_help` პროცედურა. მისი სინტაქსია:

```
sp_help 'პროცედურის_სახელი'
```

შენახული პროცედურის სახელის ცვლილება

შენახული პროცედურის სახელის შესაცვლელად გამოიყენება `sp_rename` ბრძანება, რომლის დროსაც იცვლება მხოლოდ სახელი, რომელიც `sysobjects` სისტემურ წარმოდგენაში ინახება, ხოლო ამ პროცედურაზე მიმართვები უცვლელი რჩება.

მაგალითი: 'Proc_1' შენახულ პროცედურას 'Proc_2' ახალი სახელი დავარქვათ:

```
EXEC sp_rename 'Proc_1', 'Proc_2';
```

შედეგი შეიძლება ინახოს ბრძანებით:

```
SELECT *
FROM sysobjects
ORDER BY name;
```

შენახული პროცედურის წაშლა

14

DROP PROCEDURE ბრძანება გამოიყენება შენახული პროცედურის წასაშლელად, რომლის სინტაქსია:

DROP PROC [EDURE] [სქემის_სახელი.] პროცედურის_სახელი [,...n]

შენახული პროცედურის წაშლის შედეგად მისი სახელი წაიშლება ამავე მონაცემთა ბაზის sysobjects სისტემური წარმოდგენიდან.

თუ საჭიროა, რომ წაიშალოს და ისევ შეიქმნას ერთი და იმავე სახელის მქონე შენახული პროცედურა ერთსა და იმავე პაკეტში, ბრძანებებს შორის უნდა მიეთითოს **GO** ბრძანება.

მაგალითი:

```
DROP PROC A1;  
GO  
CREATE PROC A1  
AS ...
```

შენახული პროცედურის ავტომატურად შესრულება

შენახული პროცედურის ავტომატურად შესრულებისთვის საჭიროა მისი კონფიგურირება, რომელიც ხორციელდება მისი თვისებების ცვლილების გზით და რისთვისაც გამოიყენება `sp_procoption` პროცედურა, რომლის სინტაქსია:

```
sp_procoption [ @ProcName = ] 'პროცედურის_სახელი',
[ @OptionName = ] 'რეჟიმი',
[ @OptionValue = ] 'მნიშვნელობა'
```

განვიხილოთ არგუმენტების დანიშნულება:

რეჟიმი არის იმ თვისების სახელი, რომელსაც ვცვლით. ამ შემთხვევაშია **startup** თვისება, რომელიც განსაზღვრავს შენახული პროცედურის ავტომატურად გაშვების შესაძლებლობას.

მნიშვნელობა არგუმენტი იღებს **true (ON)** ან **false (OFF)** მნიშვნელობას.

`sp_procoption` სისტემური შენახული პროცედურის შესრულების უფლება აქვს მხოლოდ SQL Server-ის ფიქსირებული როლის წევრებს. უნდა იქნეს გათვალისწინებული შეზღუდვები:

- ▶ ნებადართულია მხოლოდ `master` მონაცემთა ბაზის შენახული პროცედურების ავტომატურად გაშვება;
- ▶ ავტომატურად გასაშვები შენახული პროცედურა უნდა ეკუთვნოდეს მონაცემთა ბაზის მფლობელს - `dbo` მომხმარებელს, რის შედეგადაც `master` მონაცემთა ბაზის ფარგლებში ექნება სრული უფლებები.

შენახული პროცედურების წარმოება SSMS-ში

SQL Server Management Studio-ში (SSMS) ახალი შენახული პროცედურის შექმნისათვის მომხმარებელმა უნდა გაითვალისწინოს, რომ შენახული პროცედურები იქმნება კონკრეტული მონაცემთა ბაზისათვის და არა სხვა ობიექტისათვის (მაგალითად, ცხრილისათვის) და, ამიტომ, მაუსის მარჯვენა ღილაკის გამოყენებით მან უნდა გააქტიუროს მთავარ მენიუში საჭირო მონაცემთა ბაზის ქვემენიუში სტრიქონ „პროგრამირებადი“-ს (Programmability) ქვესტრიქონი „შენახული პროცედურები“ (Stored Procedures) და ჩამოშლილ კონტექსტურ მენიუში აირჩიოს პირველი სტრიქონი „ახალი“ (New) და შემდგომ ისევ პირველი სტრიქონი „შენახული პროცედურა...“ (Stored Procedure..), რომლის შედეგადაც მარჯვენა მიღამოში გამოსახულ ფანჯარაში აჩარმოოს შესაბამისი ბრძანებები, რომლებიც უკვე აღწერილია წინა სლაიდებში.

