

ლაბორატორია 2

მონაცემთა ბაზის შექმნა SQL Server-ში

SQL Server-ში მონაცემთა ბაზა შედგება სხვადასხვა ტიპის ობიექტებისაგან, როგორიცაა ცხრილები, ფუნქციები, შენახული(Stored) პროცედურები, View და ა.შ. SQL Server-ის თითოეულ ინსტანციას შეიძლება ჰქონდეს ერთი ან მეტი მონაცემთა ბაზა. SQL Server მონაცემთა ბაზები ინახება ფაილურ სისტემაში ფაილების სახით.

მონაცემთა ბაზის ტიპები SQL Server-ში

SQL Server-ში გვხვდება ორი ტიპის მონაცემთა ბაზა: სისტემური მონაცემთა ბაზა და მომხმარებლის მონაცემთა ბაზა

სისტემური მონაცემთა ბაზები ავტომატურად იქმნება SQL Server-ის დაყენებისას. მათ იყენებს როგორც SSMS ასევე სხვა SQL Server API-ები და ხელსაწყოები, ამიტომ არ არის რეკომენდებული სისტემური მონაცემთა ბაზების ხელით შეცვლა:

- master: მონაცემთა ბაზა ინახავს სისტემის დონის ყველა ინფორმაციას SQL Server-ის ინსტანსზე. იგი მოიცავს მეტამონაცემებს, როგორიცაა შესვლის (Logins) ანგარიშები, ბოლო წერტილები (endpoints), დაკავშირებული სერვერები და სისტემის კონფიგურაციის პარამეტრები.
- model: სამოდულო მონაცემთა ბაზა გამოიყენება როგორც შაბლონი SQL Server-ის ინსტანსიაზე შექმნილი ყველა მონაცემთა ბაზისთვის.
- msdb: ამ მონაცემთა ბაზას იყენებს SQL Server Agent-ი შეტყობინებების (alerts) და სამუშაოების (jobs) დაგეგმვისთვის და ასევე მას იყენებენ SQL Server Management Studio, Service Broker და Database Mail.
- tempdb: მონაცემთა ბაზა გამოიყენება დროებითი ობიექტების, შუალედური შედეგების და შიდა ობიექტების შესანახად, რომლებსაც მონაცემთა ბაზის ძრავა ქმნის.

მომხმარებლის მიერ განსაზღვრული მონაცემთა ბაზები იქმნება მონაცემთა ბაზის მომხმარებლის მიერ T-SQL ან SSMS გამოყენებით აპლიკაციის ან/და სხვა მონაცემების შესანახად. შესაძლებელია მაქსიმუმ 32767 მონაცემთა ბაზა შეიქმნას SQL Server ინსტანსიაში.

SQL Server-ში ახალი მონაცემთა ბაზის შექმნის ორი გზა არსებობს:

1. T-SQL გამოყენებით
2. SQL Server Management Studio-ს გამოყენებით

შექმენით მონაცემთა ბაზა T-SQL სკრიპტის გამოყენებით

თქვენ შეგიძლიათ შეასრულოთ SQL სკრიპტი Query რედაქტორში Masterმონაცემთა ბაზის გამოყენებით. შეგახსენებთ რომ Query რედაქტორის გამოძახება შეგიძლიათ ინსტრუმენტების პანელზე არსებული New Query ღილაკის დაწკაპუნებით, ამ კლავიშთა კომბინაციით Ctrl+N, ავკრიფოთ სკრიპტი:

შუა სკრიპტის გაშვება შესაძლებელია Execute ღილაკით ან F5 კლავიშით.

სინტაქსი:

```
USE [master]
GO
CREATE DATABASE <name>
```

შემდეგი სკრიპტი ქმნის "Cinema" მონაცემთა ბაზას.

```
USE [master]
GO
CREATE DATABASE [Cinema]
```

თუმცა ამ შემთხვევაში ბაზის ყველა პარამეტრი არჩეულია გაჩუმების პრინციპით. თუ გვსურს დავალაგოთ ბაზის ფიზიკური სტრუქტურა, შევცვალოთ ფაილების რაოდენობა, ლოკაცია ან ზომები საჭიროა თავად დავწეროთ და მივუთითოთ ბაზის პარამეტრები.

Ms SQL სამუალებას გვაძლევს ბაზის შექმნის დროს წინასწარ დავალაგოთ მისი ფაილების სრუქტურა, შევქმნათ ფაილური ჯგუფები და მათში დავანაწილოთ ფაილები და შესაბამისად ცხრილებიც. აღნიშნული სტრუქტურის სწორად დალაგება იქნება შემდგომ ბაზის ფიზიკური ფაილების ადვილად მართვის გარანტი, როგორც მათი მიგრაციისას ასევე სარეზერვო ასლების შექმნისას. მიუხედავად ამ ყველაფრისა მაინც ძირითად გამოწვევად Ms SQL ბაზის შექმნის დროს მიიჩნევა მისი რელაციური სტრუქტურის სწორად დალაგება, და ვფიქრობ რომ ნაკლები ყურადღება ექცევა მის ფაილურ და ფიზიკურ ნაწილს... მეტიც, ძალიან ხშირად პროგრამისტები მიიჩნევენ რომ ოპტიმალურიცაა ბაზისთვის გაჩუმების პრინციპით გამოყოფილი ორი ფაილი: ერთი ძირითადი ფაილი მონაცემებისათვის-.mdf და მეორე ტრანზაქციის ლოგი .ldf.

თავიდან რომ დავიწყოთ, თითოეული ბაზის შექმნის დროს იქმნება ერთი ფაილური ჯგუფი Primary, და მასში ავტომატურად განთავსდება ერთი ძირითადი ფაილი mdf (Main Database File) და ასევე იქმნება ერთი ტრანზაქციის ლოგის ფაილი ldf (Log Database File) რომელიც ინახავს მთლიანი ბაზის ქმედებების ისტორიას როგორც სრული ასევე ნაწილობრივი ტრანზაქციების დროს, იგი არ მიეკუთვნება არცერთ ფაილურ ჯგუფს.

CREATE DATABASE Cinema შემოკლებული T-SQL ბრძანების შესრულების დროს გაჩუმების პრინციპით იქმნება მონაცემთა ბაზა Cinema ხოლო სერვერის დისკზე ასევე გაჩუმების პრინციპით წინასწარ არჩეულ ლოკაციაზე იქმნება ზემოთ ნახსენები ორი ფაილი, რომელთა დასახელებებიც ასევე არჩეულია წინასწარ და ძირითად ფაილს აქვს იგივე დასახელება რაც ბაზას, ჩვენს შემთხვევაში Cinema.mdf ხოლო ლოგს კი Cinema _Log.ldf.

```
CREATE DATABASE <database name>
[ON [PRIMARY]
  ([NAME = <'logical file name'>],
   FILENAME = <'file name'>
   [, SIZE = <size in kilobytes, megabytes, gigabytes, or terrabytes>]
   [, MAXSIZE = size in kilobytes, megabytes, gigabytes, or terrabytes>]
   [, FILEGROWTH = <kilobytes, megabytes, gigabytes, or terrabytes|percentage>]]]
[LOG ON
  ([NAME = <'logical file name'>],
   FILENAME = <'file name'>
   [, SIZE = <size in kilobytes, megabytes, gigabytes, or terrabytes>]
   [, MAXSIZE = size in kilobytes, megabytes, gigabytes, or terrabytes>]
   [, FILEGROWTH = <kilobytes, megabytes, gigabytes, or terrabytes|percentage>]]]
```

სტრუქტურაში ჩანს ყველა SQL Server მონაცემთა ბაზისთვის სავალდებულო ოპერაციული სისტემის ფაილი ორი: **მონაცემთა ფაილი და Log (ჟურნალის) ფაილი**,

- მონაცემთა ფაილები შეიცავს მონაცემებს და ობიექტებს, როგორიცაა ცხრილები, ხედები (View), შენახული პროცედურები, ინდექსები და ა.შ.
- ჟურნალის ფაილები შეიცავს ინფორმაციას, რომელიც საჭიროა მონაცემთა ბაზაში ყველა ტრანზაქციის აღსადგენად. უნდა არსებობდეს მინიმუმ ერთი ჟურნალის ფაილი თითოეული მონაცემთა ბაზისთვის.

გავიაზროთ ფაილის შექმნის ბრძანების თითოეული პარამეტრის დანიშნულება:

- **ON** პირობის შემდეგ ხდება მითითება თუ რომელ ფაილურ ჯგუფში ხდება ფაილი, სტანდარტულ შემთხვევაში გვაქვს ორი სავალდებულო ფაილი, ესენია: **მონაცემთა ფაილი და Log (ჟურნალის) ფაილი**, აქედან პირველი ხდება ძირითად (Primary) ფაილურ ჯგუფში, რისი მითითებაც ხდება On პირობით, ჩვენს შემთხვევაში **ON PRIMARY** წინადადებით, ხოლო მეორე სავალდებულო ფაილი არის ტრანზაქციის ჟურნალის რომელიც არცერთ ჯგუფში არ ხდება და მისთვის მიეთითება **LOG ON** პირობა. გარდა სავალდებულოდ არსებული ძირითად (Primary) ფაილური ჯგუფისა თქვენ შეგიძლიათ დამატებით შექმნათ ახალი ფაილური ჯგუფები და განათავსოთ ბაზის ფიზიკური ფაილები მათში. შეგასენებთ რომ ON პირობის მერე იწერება თუ რომელ ფაილურ ჯგუფშია ფაილი.
- **NAME** პარამეტრში მიეთითება ფაილის ლოგიკური სახელი, რომლითაც ჩვენ მას საჭიროების შემთხვევაში მივმართავთ სკრიპტში.
- **FILENAME** განსაზღვრავს ფიზიკური ფაილის სახელს დისკზე. როგორც უკვე ვთქვით სავალდებულოა ერთი მონაცემთა ფაილის შექმნა ძირითად (Primary) ფაილურ ჯგუფში და მისი გაფართოვება იქნება .mdf და ერთი ტრანზაქციის ჟურნალის ფაილის შექმნა გაფართოვებით .ldf თუმცა დამატებით საჭიროების შემთხვევაში შეიძლება შეიქმნას დამატებითი მონაცემთა ფაილები .ndf გაფართოვებით და დამატებითი ჟურნალის ფაილები კვლავ .ldf გაფართოვებით.
- **SIZE** განსაზღვრავს ფაილის საწყის ზომას, რომელიც გაიზრდება თუკი ფაილი ბოლომდე გაოიყენებს რეზერვირებულ ზომას. იგი სტანდარტულად ეთითება მეგაბაიტებში, თუმცა შესაძლებელია სხვა ერთეულების გამოყენებაც.
- **MAXSIZE** წარმოადგენს ფაილის ზომის „ზედა“ ზღვარს. მარტივად რომ ვთქვათ სადამდე უნდა გაიზარდოს ზომა ფაილის გავსების შემთხვევაში. თუკი მისი მნიშვნელობაა „UNLIMITED“ ე.ი. ფაილი გააგრძელებს ზრდას სანამ ფიზიკურ დისკზე იქნება ადგილი /ზრდის საშუალება. ეს პარამეტრიც სტანდარტულად ეთითება მეგაბაიტებში, თუმცა შესაძლებელია სხვა ერთეულების გამოყენებაც.
- **FILEGROWTH** როგორც უკვე ავღნიშნეთ ბაზის ფაილი გადავსებისას იზრდება, თუმცა რამდენად გაიზრდება ყოველ ჯერზე კონტროლდება აღნიშნული პარამეტრით, იგი ეთითება მეგაბაიტებში ან პროცენტულად, თუმცა შესაძლებელია სხვა ერთეულების გამოყენებაც.

ზემოთ ჩამოვთვალეთ ხუთი პარამეტრი რომელიც აქვს თითოეულ ბაზის ფაილს, თუმცა თითოეულ მათგანს აქვს „მნიშვნელობა გაჩუმების პრინციპით“ (default) რაც ნიშნავს რომ თუ ზომას ან მაქსიმალურ ზომას გამოვტოვებთ ისინი ფაილს მიენიჭება სტანდარტების მიხედვით.

პირველი ბაზის სრული T-SQL სკრიპტი

ყოველივე ზემოთ აღნიშნული მერე შესაძლებელია დაწეროთ ჩვენი პირველი ბაზის სრული T-SQL სკრიპტი ორივე სავალდებულო ფაილის აღწერით:

```
CREATE DATABASE [Cinema]
ON PRIMARY
( NAME = N'Cinema',
FILENAME = N'D:\MyFile\Cinema.mdf' ,
SIZE = 8MB ,
MAXSIZE = UNLIMITED,
FILEGROWTH = 64MB )

LOG ON
( NAME = N'Cinema_log',
FILENAME = N'D:\MyFile\Cinema_log.ldf' ,
SIZE = 8MB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 64MB )
GO
```

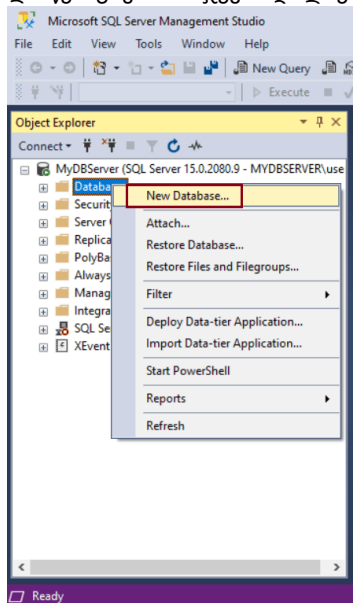
გაითვალისწინეთ საქაღალდე, რომელ ლოკაციაზეც აპირებთ შექმნათ ბაზის ფაილები უნდა არსებობდეს წინასწარ.

ახლა გახსენით SSMS და განაახლეთ მონაცემთა ბაზების საქაღალდე და დაინახავთ, რომ გამოჩნდა „Cinema“ მონაცემთა ბაზა.

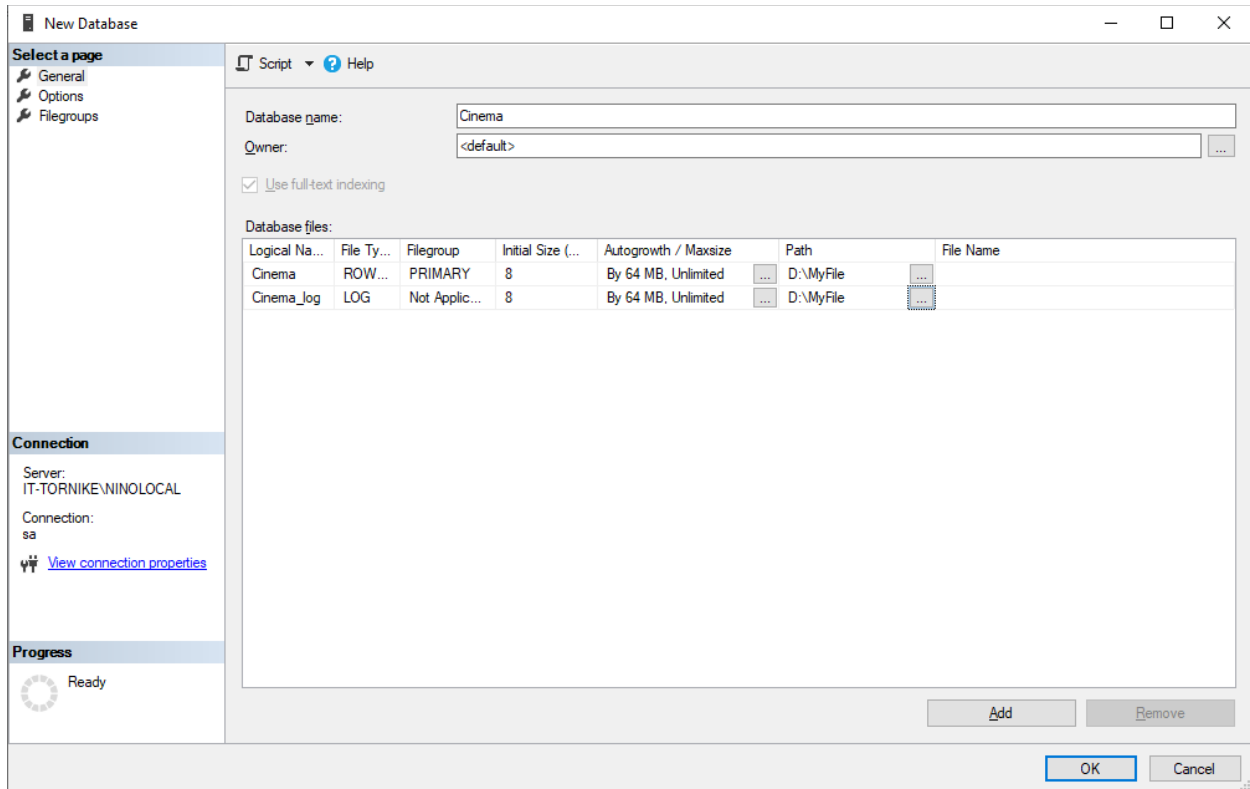
უკვე შექმნილი ბაზის Script ის ნახვა შესაძლებელია ბაზაზე მარჯვენა ღილაკით გამოტანილი კონტექსტური მენიუდან Script database As/Create to/New Query Editor Window ბრძანებით.

ახლა შევასრულოთ იგივე მოქმედება მენეჯმენტ სტუდიის დახმარებით: (გაითვალისწინეთ რომ სერვეზე დაუშვებელია ერთი და იმავე დასახელებით ორი ბაზის შექმნა)

დააწკაპუნეთ მარჯვენა ღილაკით Databases საქაღალდეზე და დააწკაპუნეთ New Database.. მენიუს ბრძანებაზე:



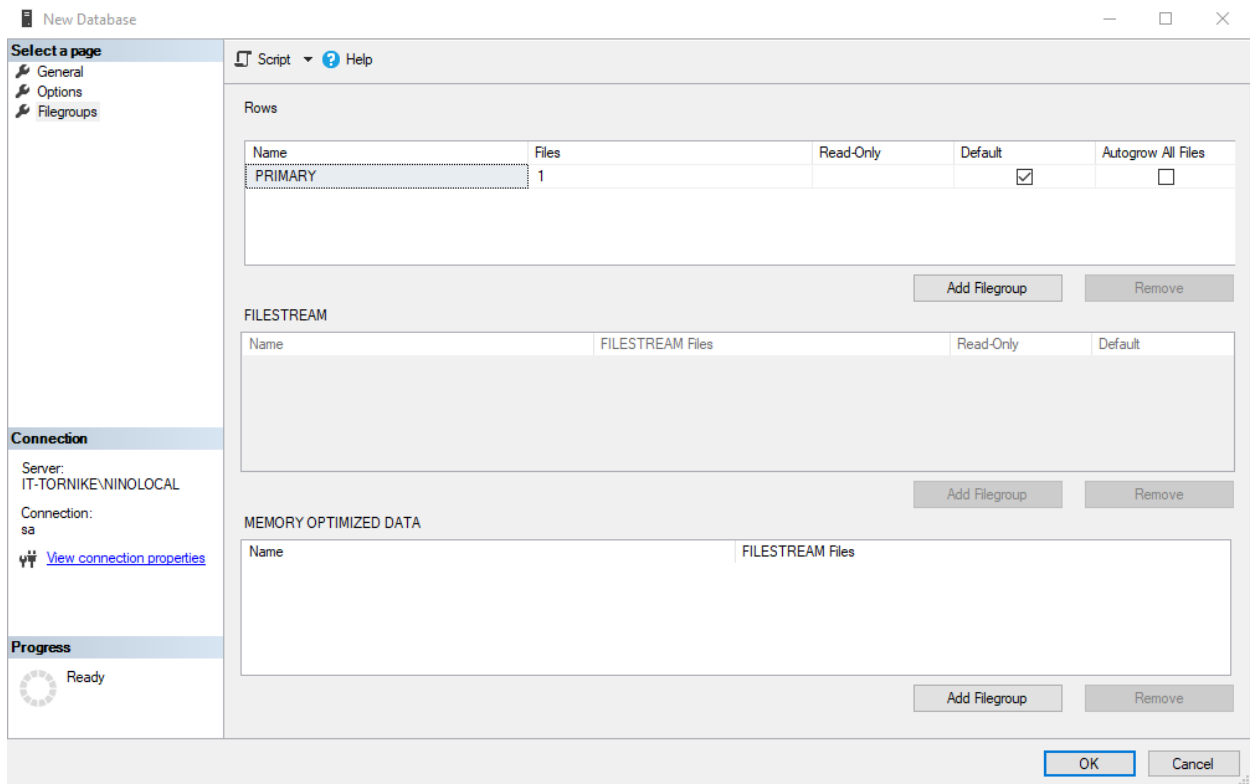
ახალი მონაცემთა ბაზის ფანჯარაში შეიყვანეთ ახალი მონაცემთა ბაზის სახელი, როგორც ეს ნაჩვენებია ქვემოთ.



მონაცემთა ბაზის მფლობელი შეიძლება დარჩეს ნაგულისხმევად ან მფლობელის შესაცვლელად დააწკაპუნეთ ღილაკზე [...].

მონაცემთა ბაზის ფაილების ბადის ქვეშ ჩანს რომ ყველა SQL Server მონაცემთა ბაზას აქვს ჩვენთვის უკვე ნაცნობი ორი ოპერაციული სისტემის ფაილი: **მონაცემთა ფაილი და Log (ჟურნალის) ფაილი**. რომელთა დასახელებებიც ავტომატურად ივსება ბაზის დასახელების შესაბამისად, ისევე როგორც ზომები თუმცა თქვენ შეგიძლიათ შეცვალოთ ნაგულისხმევი მნიშვნელობები აღნიშნული ფაილებისთვის. მონაცემთა ბაზის ოფციების (Options) გვერდზე თქვენ შეგიძლიათ შეცვალოთ Collation (სიმბოლოთა შაბლონები), Recovery (აღდგენის მოდელი) ასევე თავსებადობის დონე და შეკავების ტიპი, აღნიშნულ საკითხებს ამ ლექციაზე არ განვიხილავთ.

ახლა აირჩიეთ Filegroups ჩანართი. ფაილური ჯგუფები არის ფიზიკური ფაილები თქვენს დისკზე, სადაც ინახება SQL სერვერის მონაცემები. ნაგულისხმევად, ძირითადი (Primary) მონაცემთა ფაილი იქმნება ახალი მონაცემთა ბაზის შექმნისას, ხოლო ტრანზაქციის ლოგის ფაილი არ მიეკუთვნება არცერთ ფაილურ ჯგუფს. შესაბამისად გვექნება მხოლოდ ერთი ფაილური ჯგუფი:



Ok ღილაკზე დაწკაპუნების შემდეგ შეიქმნება მონაცემთა ბაზა. რომლის დანახვაც შესაძლებელი იქნება Object Explorer ფანჯრის შიგნით, სერვერის ბაზების ჩამონათვალში.

თუმცა თუ ბაზის სტრუქტურიდან გამომდინარე წინასწარი გათვლით (იდეალურ შემთხვევაში) ვიცით რომ მოსალოდნელია ბაზის ზომის მკეთრად ზრდა, შესაბამისად დახარისხებული გვაქვს მონაცემები სწორად, რათა არ დავდგეთ დამუშავების დროის გაზრდის, გეგმიური სარეზერვო კოპირებების (back up) აღების ან სულაც მიმდინარე ფიზიკურ დისკზე არასაკმარისი სივრცის ქონის პრობლემასთან „სწორი“ გადაწყვეტილება იქნება, ბაზის დაშლა ფაილურ ჯგუფებად, და ერთის მაგიერ მონაცემთა რამოდენიმე ფიზიკური ფაილის განთავსება. ასევე შემდგომ მათში ცხრილების სწორად გადანაწილება.

მარტივად რომ ვთქვათ სხვადასხვა 'მძიმე' ცხრილები შეიძლება მოვათავსოთ სხვადასხვა ფიზიკურ ფაილებში, სხვადასხვა ლოკაციებზე, და ცალცალკე ვაკეთოთ მათი სარეზერვო კოპირება. მაგრამ ცხრილის ჩასმა ხდება ფაილურ ჯგუფში და არა უშუალოდ ფაილში. იგივე შეიძლება ითქვას სარეზერვო კოპირებაზე. მაგალითისთვის შევქმათ იგივე ბაზა რამოდენიმე ფაილურ ჯგუფითა და ფაილით:

```
CREATE DATABASE [Cinema]
ON PRIMARY
( NAME = N'Cinema',
  FILENAME = N'D:\MyFile\Cinema.mdf' ,
  SIZE = 8192KB ,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 65536KB ),

( NAME = N'Cinema',
  FILENAME = N'D:\MyFile\Cinema.ndf' ,
  SIZE = 1024KB ,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 65536KB ),
```

```






FILEGROUP [SECONDARY]
( NAME = N'Cinema3',
  FILENAME = N'D:\MyFile\Cinema3.ndf' ,
  SIZE = 1024KB , MAXSIZE = UNLIMITED,
  FILEGROWTH = 65536KB
)

LOG ON
( NAME = N'Cinema_log', FILENAME = N'D:\MyFile\Cinema_log.ldf' , SIZE = 1024KB
, MAXSIZE = 2048GB , FILEGROWTH = 65536KB ),
( NAME = N'Cinema_log2', FILENAME = N'D:\MyFile\Cinema_log2.ldf' , SIZE =
1024KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

```

თუ დავაკვირდებით შევამჩნევთ რომ მონაცემთა მთავარი ფაილი .mdf მხოლოდ ერთია ხოლო მომდევნო მონაცემთა ფაილები განურჩევლათ ფაილური ჯგუფისა არის .ndf გაფართოების და წარმოადგენს მონაცემთა მეორად ფაილებს.

აღნიშნულ ეტაპზე საქალაქდ გამოიყურება D:\MyFile შემდეგნაირად:

 Cinema.mdf	8,192 KB
 Cinema_log.ldf	1,024 KB
 Cinema_log2.ldf	1,024 KB
 Cinema2.ndf	1,024 KB
 Cinema3.ndf	1,024 KB

ხოლო შესაბამისი ბაზის მდგომარეობა სერვერზე შეგიძლიათ ნახოთ

`EXEC sp_helpdb [Cinema]` ბრძანებით.

უკვე შექმნილი ფაილების მოდიფიკაცია შესაძლებელია `MODIFY` ბრძანებით, თუმცა მანამდე საჭიროა გავხსნათ ბაზის რედაქტირების რეჟიმი `ALTER` ბრძანებით. ამიერიდან თუ გვენდომება რაიმე ახალი ობიექტის შექმნა გამოვიყენებთ ჩვენთვის უკვე ნაცნობ ბრძანება `CREATE`-ს. ხოლო წასაშლელად ბრძანება `DROP`-ს.

შევცვალოთ ძირითადი ფაილის პარამეტრები:

```

ALTER DATABASE [Cinema]
MODIFY FILE
( NAME = N'Cinema',
  FILENAME = N'D:\MyFile\Cinema.mdf' ,
  SIZE = 20 MB ,
  MAXSIZE = 20 GB,
  FILEGROWTH = 8 MB )
GO

```

დავამატოთ ბაზას ერთი მეორადი ჯგუფის ფაილი , გავხსნათ ბაზის რედაქტირების რეჟიმი `ALTER` ბრძანებით და ჩავამატოთ შიგ ფაილი `ADD` ოპერატორის გამოყენებით:

```

ALTER DATABASE [Cinema]
ADD FILE
( NAME = N'Cinema4',
  FILENAME = N'D:\MyFile\Cinema4.ndf' ,
  SIZE = 30 MB ,
  MAXSIZE = 30 GB,
  FILEGROWTH = 10 MB )

```

წავშალოთ ბოლოს დამატებული ფაილი იგივე საფეხურების გამეორებით, გაითვალისწინეთ რომ წაშლისა და რედაქტირების დროს ფაილის ლოგიკური სახელი გამოიყენება მასზე მიმთითებლად.

```
ALTER DATABASE [Cinema]
    REMOVE FILE Cinema4
```

დავამატოთ ახალი ფაილური ჯგუფი

```
ALTER DATABASE [Cinema]
ADD FILEGROUP NEWGROUP;
```

დავამატოთ ბაზას კიდევ ერთი მონაცემთა ფაილი ამჯერად ახალ ფაილურ ჯგუფში :

```
ALTER DATABASE [Cinema]
    ADD FILE
        ( NAME = N'Cinema4',
          FILENAME = N'D:\MyFile\Cinema4.ndf'
        ) TO FILEGROUP NEWGROUP
```

დავამატოთ ტრანზაქციის ჟურნალი

```
ALTER DATABASE [Cinema]
ADD LOG FILE
(
    NAME = Cinema_log3,
    FILENAME = N'D:\MyFile\Cinema_lo3.ldf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
```

როგორც უკვე ვიცით ბაზის შექმნისთანავე იქმნება ძირითადი ფაილური ჯგუფი რომელიც ასევე წარმოადგენს ჯგუფს გაჩუმების მეთოდით, რაც ნიშნავს რომ ფაილის დამატებისას თუ არ დავაკონკრეტებთ რომელ ჯგუფში უნდა ჩაჯდეს იგი ავტომატურად ჩაჯდება ძირითად (PRIMARY) ფაილურ ჯგუფში. Default ფაილური ჯგუფის ცვლილება შესაძლებელია შემდეგი სკრიპტით:

```
ALTER DATABASE [Cinema]
MODIFY FILEGROUP NEWGROUP DEFAULT
```

ახლა კვლავ ძირითადი ჯგუფი დავაბრუნოთ Default-ად და NEWGROUP წავშალოთ. (გაითვალისწინეთ რომ წაშლისას ჯგუფში არ უნდა იყოს ფაილი. წაშალეთ ჯერ ფაილები შემდეგ ჯგუფი)

```
ALTER DATABASE [Cinema]
    MODIFY FILEGROUP [PRIMARY] DEFAULT
```

```
ALTER DATABASE [Cinema]
    REMOVE FILE Cinema4
```

```
ALTER DATABASE [Cinema]
    REMOVE FILEGROUP NEWGROUP
```

წავშალოთ ბაზა:

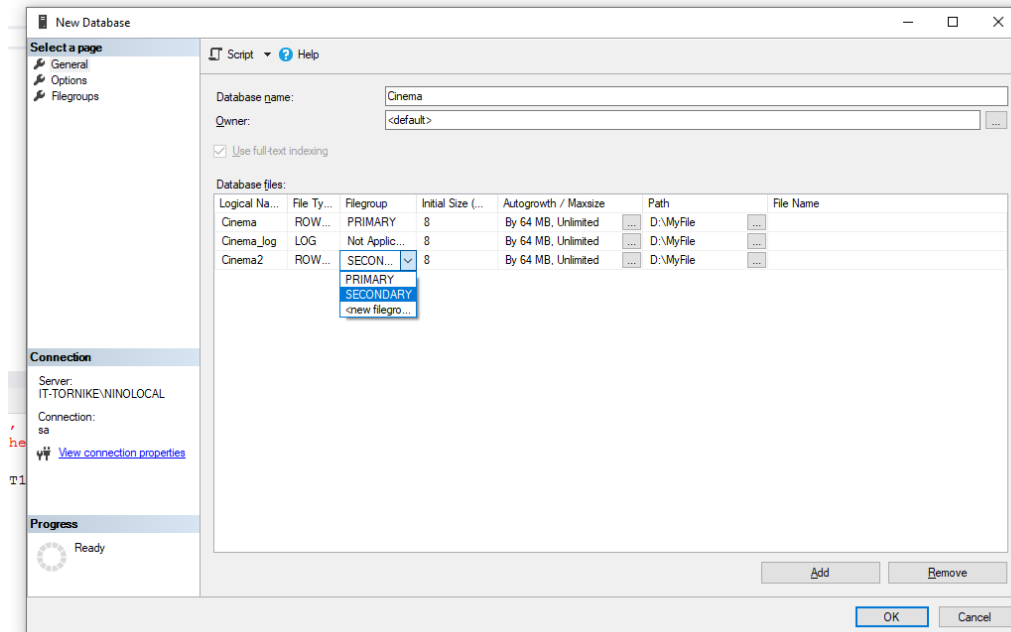
```
DROP DATABASE [Cinema]
```

ვინაიდან აღნიშნულ თავში რამოდენიმე DDL (Data Definition Language)-ის ბაზის სტრუქტურირების ძირითადი ბრძანებები გავიარეთ, მივცეთ ამას ნორმალიზებული ფორმა მეხსიერებაში ☺

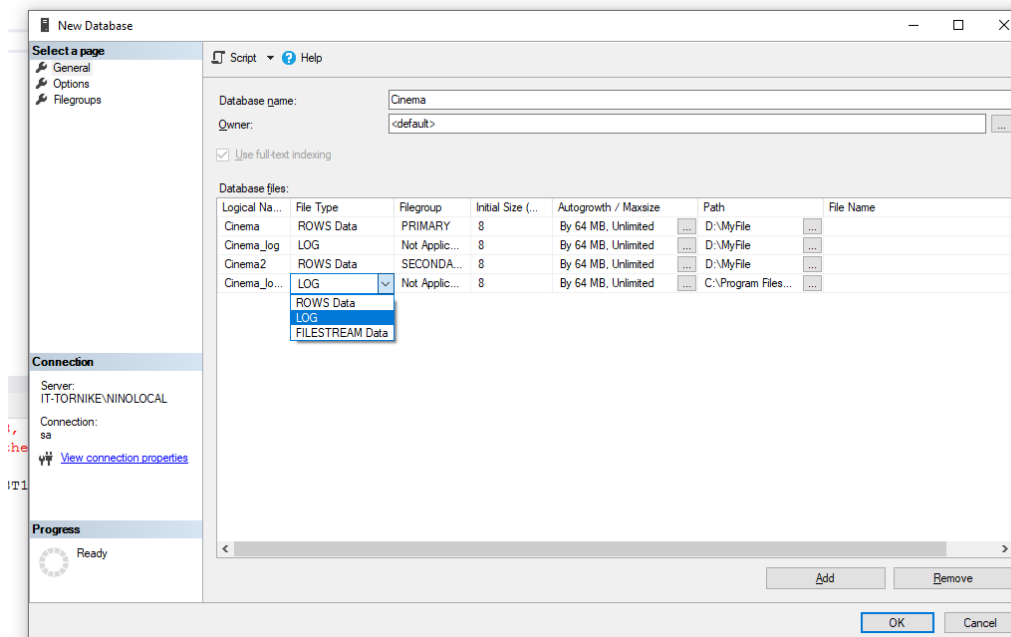
Create	ობიექტის შექმნა
Drop	ობიექტის წაშლა
Alter	ობიექტის ცვლილება ამ შემთხვევაში ხდება შიგთავსის ცვლილება და შესაძლებელია გამოყენებული იყოს ქვებრძანებები ძირითადად ერთად:
Add	ქვობიექტის ჩამატება
Modify	ქვობიექტის რედაქტირება
Remove	ქვობიექტის წაშლა

თუ დამატებითი ფაილებისა და ფაილური ჯგუფების დამატება გვსურს დიზაინის რეჟიმში, ბაზის შექმნის დროს ზემოთ ნაჩვენებ New Database ფანარაში უნდა გადავიდეთ Filegroups ჩანართზე და Add FileGroup დილაკით დავამატოთ ახალი ფაილური ჯგუფი, გავეწეროთ დასახელება:

შემდეგ დავბრუნდეთ General ჩანართში Add დილაკის გამოყენებით დავამატოთ ფაილი, გავეწეროთ სასურველი სახელი და Filegroup გრაფაში ჩამოსაშლელ სიაში ავურჩიოთ ახლად შექმნილი ჯგუფი:



ანალოგიური მეთოდით შეიქმენება ტრანზაქციის ჟურნალები, მხოლოდ ფაილის ტიპში ROWS Data (მონაცემთა ფაილის) მაგიერ ავარჩევთ LOG(ჟურნალის) ტიპს



ასევე ავარჩევთ ფაილის სასურველ ლოკაციას. საჭიროების შემთხვევაში კი აქვე გვაქვს Remove ღილაკი არასაჭირო ფაილების წასაშლელად.

SQL სერვერის მონაცემთა ტიპები

SQL Server-ში “data type” განსაზღვრავს მონაცემთა ტიპს, რომელიც შეიძლება ჩაწეროს ცხრილის ამა თუ იმ სვეტში, მაგალითად ერთ სვეტში შეიძლება ვინახავდეთ მთელ რიცხვებს, მეორეში-სტრიქონულ მონაცემებს, მესამეში თარიღსა და დროს, ორობით სტრიქონებს და ა.შ.

მონაცემთა ტიპების სწორად განსაზღვრას გადამწყვეტი მნიშვნელობა აქვს ბაზის დაპროექტებისას, ვინაიდან იგი გავლენას ახდენს ბაზის და შესაბამისად აპლიკაციის სწრაფქმედებასა და ეფექტურობაზე.

SQL Server-ში ჩაშენებულია ყველა სახის მონაცემთა ტიპები, თუმცა საჭიროების შემთხვევაში თქვენ თავადაც შეგიძლიათ შექმნათ საკუთარი ტიპი.

მონაცემთა ტიპის კატეგორიები:

კატეგორია	მონაცემთა ტიპი
ზუსტი რიცხვები	bit, tinyint, smallint, int, bigint, decimal, numeric, money, smallmoney
ნამდვილი რიცხვები	Real, Float
თარიღი და დრო	date, smalldatetime, datetime, datetime2, datetimeoffset, time
სტრიქონი/სიმბოლო	char, varchar, text
უნიკოდის სიმბოლოები	nchar, nvarchar, ntext
სხვა	cursor, hierarchyid, sql_variant, spatial Geometry types, spatial Geography types, rowversion, uniqueidentifier, xml, table

ზუსტი რიცხვები

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
bit	0,1 ან NULL	მონაცემთა ყველაზე პატარა ტიპი 1 ბაიტი ზომით.
tinyint	0-255-მდე	1 ბაიტი
smallint	-32,768-დან 32,767-მდე	2 ბაიტი
int	-2,147, 483,648	4 ბაიტი
bigint	-9,223,372, 036,854,775,808	8 ბაიტი
decimal	$-10^{38}+1$ -დან $10^{38}-1$ -მდე	რიცხვითი მონაცემთა ტიპი, რომელსაც აქვს ფიქსირებული სიზუსტე და მასშტაბი.
smallmoney	-214,748.3648-დან 214,748.3647-მდე	4 ბაიტი
money	-922,337,203,685,477,5808-დან 922,337,203,685,477.5807-მდე	8 ბაიტი

ნამდვილი რიცხვები

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
float(n)	- 1.79E+308 -დან -2.23E-308-მდე, 0	ზომა დამოკიდებულია n-ის მნიშვნელობაზე
real	- 3.40E + 38 -დან -1.18E - 38-მდე, 0 და 1.18E - 38 -დან 3.40E + 38-მდე	4 ბაიტი

თარიღი და დრო

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
date	0001-01-01 -დან 9999-12-31-მდე	3 ბაიტი
datetime	1753-01-01 -დან 9999-12-31-მდე	8 ბაიტი
datetime2	0001-01-01 -დან 9999-12-31-მდე	სიზუსტე < 3 : 6 ბაიტი
smalldatetime	1900-01-01 -დან 2079-06-06-მდე	4 ბაიტი
datetimeoffset	0001-01-01 -დან	10 ბაიტი
time	00:00:00.0000000 -დან 23:59:59.9999999-მდე	5 ბაიტი

სტრიქონი/სიმბოლო

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
char[(n)]	1 -დან 8000 სიმბოლომდე	n ბაიტი
varchar[(n)]	1 -დან 8000	n ბაიტი+ 2 ბაიტი
varchar(max)	1 -დან 2 ³¹ -1	n ბაიტი+ 4 ბაიტი
text	0 -დან 2,147,483,647	n ბაიტი+ 4 ბაიტი

უნიკოდის სიმბოლოები

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
nchar[(n)]	1 -დან 4000 სიმბოლომდე	2n ბაიტი
nvarchar[(n max)]	1 -დან 4000	2n ბაიტი
ntext	0 -დან 1,073,741,823	2-ჯერ მეტი სტრიქონის სიგრძეზე

ორობითი სტრიქონები

მონაცემთა ტიპი	მნიშვნელობების დიაპაზონი	აღწერა/ზომა
binary[(n)]	1 -დან 8000 ბაიტამდე	n ბაიტი
varbinary[(n max)]	1 -დან 8000	სტრიქონის სიგრძეს + 2 ბაიტი
Image	0 -დან 2,147,483,647	ცვლადი სიგრძის ორობითი მონაცემები

სხვა

მონაცემთა ტიპი	აღწერა/ზომა
cursor	მონაცემთა ტიპი ცვლადებისთვის ან შენახული პროცედურების OUTPUT პარამეტრები, რომლებიც შეიცავს მითითებას კურსორზე.
rowversion	აბრუნებს ავტომატურად გენერირებულ უნიკალურ ორობით რიცხვებს მონაცემთა ბაზაში.
hierarchyid	ცვლადი სიგრძის სისტემის მონაცემთა ტიპი

მონაცემთა ტიპი	აღწერა/ზომა
uniqueidentifier	16 ბაიტისანი GUID (უნიკალური კოდი)
sql_variant	ინახავს სხვადასხვა SQL სერვერის მხარდაჭერილ მონაცემთა ტიპების მნიშვნელობებს.
xml	ინახავს xml მონაცემებს
Spatial Geometry type	გამოიყენება ბრტყელ კოორდინატულ სისტემაში (ევკლიდური) მონაცემების წარმოსაჩენად.
table	ეს არის სპეციალური მონაცემთა ტიპი, რომელიც გამოიყენება შედეგთა ნაკრების (result set) დროებით შესანახად მოგვიანებით დასამუშავებლად.