ლაბორატორია 3

Contents

ცხრილის შექმნა

ცხრილები არის მონაცემთა ბაზის ობიექტები, რომლებიც შეიცავს მონაცემთა ბაზაში არსებულ ყველა მონაცემს. ცხრილში მონაცემები ლოგიკურად არის ორგანიზებული რიგებად და სვეტებად. თითოეული სვეტი წარმოადგენს ველს და თითოეული სტრიქონი კი წარმოადგენს უნიკალურ ჩანაწერს. ზემოთხსენებულ ყველა სვეტს აქვს მასთან დაკავშირებული მონაცემთა ტიპი. ის წარმოადგენს ამ სვეტის მონაცემთა ტიპს. თითოეული ცხრილის სახელი უნიკალურია მონაცემთა ბაზის დონეზე.

მონაცემთა ბაზაში ცხრილების რაოდენობა შეზღუდულია მხოლოდ მონაცემთა ბაზაში დაშვებული ობიექტების რაოდენობით (2,147,483,647). მომხმარებლის მიერ განსაზღვრულ ცხრილს კი შეიძლება ჰქონდეს 1024 სვეტამდე.

SQL Server-ში ახალი ცხრილის შესაქმნელად ორი გზა არსებობს:

- T-SQL სკრიპტის გამოყენება
- Table Designer-ის გამოყენება SQL Server Management Studio-ში

ცხრილის შექმნა T-SQL სკრიპტის გამოყენებით

ამისათვის საჭიროა დავწეროთ CREATE TABLE ბრძანება SSMS-ის Query:

აღნიშნული ბრძანების სინტაქსი შემდეგია

```
CREATE TABLE [database_name.][schema_name.]table_name (
    pk_column_name data_type PRIMARY KEY,
    column_name2 data_type [NULL | NOT NULL],
    column_name3 data_type [NULL | NOT NULL],
    ...,
    [table_constraints]
);
```

`CREATE TABLE` საბრძანებო სიტყვის შემდეგ მიეთითება ცხრილის სახელი(რა სახელითაც გვსურს მისი შექმნა), შესაძლებელია მივუთითოთ ცხრილის სრული ან შემოკლებული(მხოლოდ ცხრილის სახელი) მარტივად რომ ვთქვათ სრული სახელის მითითებისას იწერება ჯერ ბაზის სახელი, შემდეგ სქემის სახელი და ბოლოს თავად ცხრილის სახელი `[database_name.][schema_name.]table_name` სტილში. ხოლო მხოლოდ ცხრილის სახელის მითითებისას დანარჩენი პარამეტრები იგულისხმება გაჩუმების პრინციპით.

ცხრილის სახელის მითითების შემდეგ ვხსნით მრგვალ ფრჩხილებს-() და ვწერთ სვეტების ჩამონათვალს,რომლებიც გვინდა რომ გვქონდეს ცხრილში.

სვეტს გააჩნია ჩემდეგი სამი პარამეტრი

`column_name data_type constraints`

- დასახელება
- მონაცემთა ტიპი
- შეზღუდვა

აქედან მხოლოდ პირველი ორია სავალდებულო, ხოლო შეზღუდვები შეიძლება ან საერთოდ არ ჰქონდეს სვეტს, ან რამოდენიმე ერთად ედოს. სვეტები ერთმანეთისგან გამოყოფილია მძიმით. ასევე მნიშვნელოვანია რომ შეზღუდვები შეიძლება დაიწეროს როგორც სვეტის გვერდით ასევე სვეტების ჩამონათვლის ბოლოს.

მონაცემთა ტიპის კატეგორიები:

ცხრილის სვეტებისთვის შეიძლება გამოყენებული იყოს მონაცემთა სხვადასხვა ტიპები:

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	0255-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
uniqueidentifier	16 ბაიტიანი GUID (უნიკალური კოდი)

მონაცემთა ტიპი	აღწერა/ზომა
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.

შეზღუდვები:

მოკლედ განვიხილოთ რა შეზღუდვები შეიძლება ჰქონდეს სვეტს/სვეტებს:

NOT NULL

```
DirectionId INT NOT NULL
```

არაცარიელი სვეტი. თუ სვეტში არ შეგვაქვს ჩანაწერი მაშინ მას აქვს NULL მნიშვნელობა, შესაბამისად ცარიელია და წარმოადგენს არასავალდებულო ველს. არასავალდებულო ველებისგან განსხვავებით, სავალდებულო ველებში აუცილებელია მნიშვნელობის შეტანა. თუკი სვეტს მინიჭებული აქვს NOT NULL შეზღუდვა ე.ი ის წარმოადგენს სავადებულო ველს და ჩანაწერების შეტანისას მისი გამოტოვება შეუძლებელია.

IDENTITY

```
SubjectId INT IDENTITY (1,1)
```

ავტომატური გადანომრვა, ბაზაში მონაცემების განლაგებისა და შემგომ მათი სწრაფი ძებნის მეთოდებისთვის საჭიროა რომ ყოველ ჩანაწერს ქონდეს თავისი უნიკალური ნომერი (გამონაკლისების გარდა). თუმცა ამ უნიკალური ნომრის ყოველთვის ხელით მინიჭება მოუხერხებელი და ზოგჯერ შეუძლებელიცაა, მაგალითად ბაზაში ემატება სტუდენტი და მას ენიჭება უნიკალური ნომერი, და რა ნომერი უნდა იყოს ეს? სავარაუდოდ ბოლოს+1, გამოდის უნდა მოვიძიოთ ბოლოს დამატებული სტუდენტი, გამოვთვალოთ ახალი სტუდენტის უნიკალური ნომერი და მივანიჭოთ მას? რა თქმა უნდა არა! სვეტის შეზღუდვა IDENTITY ავტომატურად ნომრავს სტრიქონებს, განსაღვრული ფუძით და ბიჯით. გაჩუმების პრინციპით ფუძეც და ბიჯიც 1-ის ტოლია IDENTITY(1,1) და ვიღებთ სვეტების ნუმერაციას 1,2,3,4..... თუმცაღა თუ გვინდა 2000 წლიდან ყოველი 5 წლის ბიჯით წლების მინიჭება გვექნება IDENTITY(5,2000)შეზღუდვის სინტაქსი ფუძით 2000 და ბიჯით 5. ასვევ საყურადღებოა რომ IDENTITY შეზღუდვის მქონე სვეტები არ ივსება მომხმარებლის მიერ და არ მონაწილეობს შევსების ოპერაციებში.

CHECK

```
Email VARCHAR (30) CHECK (Email LIKE '%@%')
```

„შემოწმება“ -ეს არის შეზღუდვა რომელიც ამოწმებს სვეტს ნებისმიერი ლოგიკური პირობის ჭეშმარიტობაზე, მაგალითად რომ მნიშვნელობა დადებითია, ან რაიმე დიაპაზონშია მოქცეული. რომ თარიღი უკვე გასული არაა, პირადი ნომრის სიგრძე ზუსტად 11 სიმბოლოა, ან მეილის სვეტი აუცილებლად შეიცავს ‘@’-ს სიმბოლოს

UNIQUE

```
PersonalId VARCHAR(11) UNIQUE
```

უნიკალურობის შეზღუდვა, როდესაც საჭიროა რომ სვეტში არ მეორდებოდეს მონაცემები, და თუ პირადი ნომრის სვეტში ერთ მომხმარებელს უკვე მინიჭებული აქვს 01001001001 პირადი ნომერი, მეორე მომხმარებელთან იგივე პირადი ნომერის ჩაწერა ვერ მოხერხდეს.

DEFAULT

```
City NVARCHAR(50) DEFAULT (N'თბილისი')
```

მნიშვნელობა გაჩუმების პრინციპით, თუკი სვეტში მნიშვნელობა არასავალდებულოა მაგრამ მაინც ვანიჭებთ მას გაჩუმების პრინციპით, მაშინ ჩვენ გვჭირდება შეზღუდვა DEFAULT. მაგალითად ვთქვათ საცხოვრებელი ქალაქი არ არის სავალდებულო ველი და მასში წინასწარ ამორჩეულია მნიშვნელობა 'თბილისი'. რაც ნიშნავს რომ შესაძლებელია მივუთითოთ სასურველი ქალაქი, მაგრამ თუ არ მივუთითებთ სისტემა ავტომატურად ჩაწერს მნიშვნელობას 'თბილისი'.

PRIMARY KEY

ცხრილის გასაღები ველი, რომელიც უზრუნველყოფს ცხრილში უნიკალური არანულოვანი ინდექსის არსებობას, რითაც იძლევა ცალსახად ძებნის და ცხრილთან კავშირის შესაძლებლობას.

FOREIGN KEY

ძირითად და გარე გასაღებებზე დეტალურად ვისაუბრებთ მომდევნო თემაში

შემდეგი CREATE TABLE ბრძანება ქმნის ცხრილს Directions

```
CREATE TABLE Directions

(
DirectionId INT NOT NULL PRIMARY KEY IDENTITY,
DirectionName NVARCHAR(150) NOT NULL UNIQUE ,
DirectionHead INT NULL
)
GO
```

ზემოთ მოყვანილი SQL სკრიპტი შექმნის ახალ `Directions` ცხრილს ნაგულისხმევ სქემაში dbo და მონაცემთა ბაზაში, რომელიც მითითებულია შეკითხვის (Query) რედაქტორის მიერ SSMS-ში. როგორც უკვე აღვნიშნეთ შესაძლებელია მიუთითოთ ცხრილის სრული სახელი ფორმატში DatabaseName.SchemaName.TableName. თუ dbo სქემაში და Faculty ბაზაში შევქმნით იმავე ცხრილს, მივიღებთ სკრიპტს:

```
CREATE TABLE Faculty .dbo.Directions

(
DirectionId INT NOT NULL PRIMARY KEY IDENTITY,
DirectionName NVARCHAR(150) NOT NULL UNIQUE ,
DirectionHead INT NULL
);
```

- ცხრილის სახელი შეიძლება იყოს მაქსიმუმ 128 სიმბოლო.
- სვეტების სახელები მითითებულია [ColumnName DataType Constraint] მძიმით გამოყოფილი ფორმატით. ჩვენ ვქმნით `DirectionId`, `DirectionName` და `DirectionHead` სვეტებს.

- სვეტის სახელის შემდეგ ეთითება მონაცემთა რომელ ტიპს შეინახავს სვეტი, ჩვენს შემთვევაში გამოყენებული იქნება `INT, NVARCHAR` და `INT` ტიპები.
- NOT NULLმიუთითებს, რომ სვეტი არ შეიძლება იყოს ცარიელი. თუ აღნიშნული შეზღუდვა არ იქნება მითითებული ნაგულისხმევად, სვეტი დაუშვებს null მნიშვნელობას.
- IDENTITY განსაზღვრავს მთელი რიცხვების სვეტს ავტომატურად გენერირებული ნუმერაციით. გაჩუმების პრინციპით ნაგულისმებია რომ მნიშვნელობა დაიწყება 1-დან და გაიზრდება 1-ით ყოველი ახალი სტრიქონისთვის.
- PRIMARY KEY

შემდეგი სტკიპრტი წაშლის `Directions` ცხრილს, თუ უკვე არსებობს და შემდეგ ხელახლა ქმნის მას.

```
USE Faculty

GO
DROP TABLE IF EXISTS Faculty.dbo.Directions; --drop table if already exists


CREATE TABLE Directions
(DirectionId INT NOT NULL PRIMARY KEY IDENTITY,
DirectionName NVARCHAR(150) NOT NULL UNIQUE ,
DirectionHead int NULL
);
```

ასევე შეზღუდვის შექმნა შესაძლებელია დამხმარე სიტყვა **“CONSTRAINT”** გამოყენებით. თუ შეზღუდვას დავწერთ სრული ფორმატით მაშინ დაგვჭირდება სიტყვა CONSTRAINT და შეზღუდვის სახელი. მაგალითად პირველ სვეტზე არსებული PRIMARY KEY შეზღუდვის დადება შესაძლებელია ისე რომ ჩვენ თავად დავარქვათ შეზღუდვას სახელი და არ მოხდეს მისი დაგენერირება სისტემის მიერ:

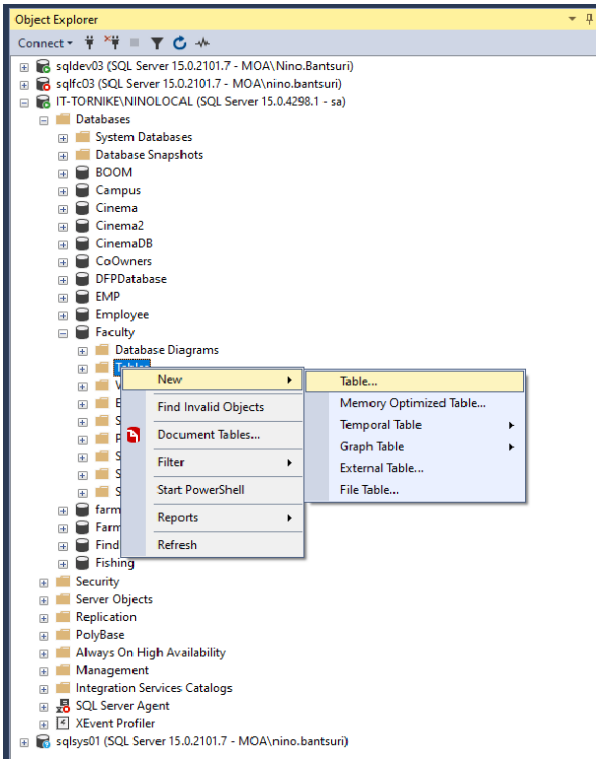
```
DirectionId INT NOT NULL CONSTRAINT PK_Directions PRIMARY KEY
```

ამ შემთვევაში ჩვენ გვეცოდინება რომ გასაღებ შეზღუდვას ჰქვია `PK_Directions`, და შედარებით გაგვიმარტივდება მასზე ოპერაციები საჭიროების შემთხვევაში. და ბოლოს სრული ფორმით შეზღუდვის დაწერა შეიძლება სვეტების ჩამონათვლის ბოლოს, უბრალოდ გასათვალისწინებელია, რომ შეზღუდვის სვეტისგან დამოუკიდებლად დაწერისას უნდა მივუთითოთ რომელ სვეტს/სვეტებს ედება იგი:

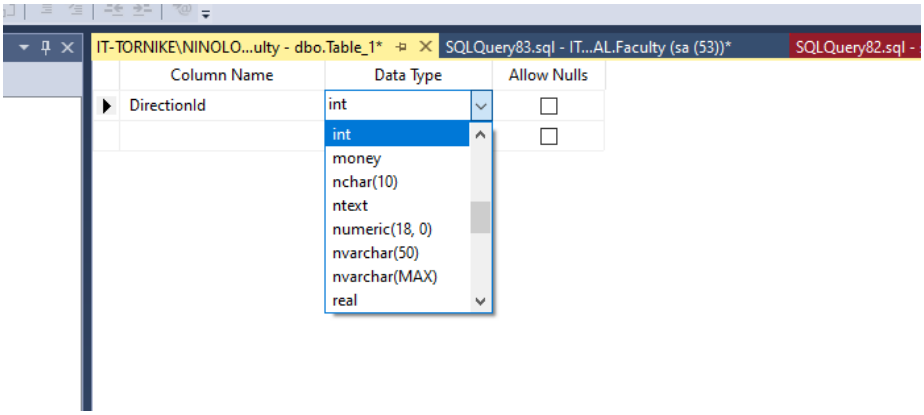
```
CREATE TABLE Directions
(DirectionId INT NOT NULL  IDENTITY,
DirectionName NVARCHAR(150) NOT NULL UNIQUE ,
DirectionHead int NULL,
CONSTRAINT PK_Directions PRIMARY KEY (DirectionId)
)
```

შევქმნათ ცხრილი SSMS -ის გამოყენებით

- ახალი ცხრილის შესაქმნელად, გახსენით SSMS და დაუკავშირდით თქვენს sql სერვერს.
- Object Explorer-ში ჩამოშალეთ `Faculty` მონაცემთა ბაზა (ან ჯერ შექმენით `Faculty` ბაზა)
- ახლა, დააწკაპუნეთ tables საქაღალდეზე მარჯვენა ღილაკით და აირჩიეთ New Table, როგორც ეს ნაჩვენებია ქვემოთ:

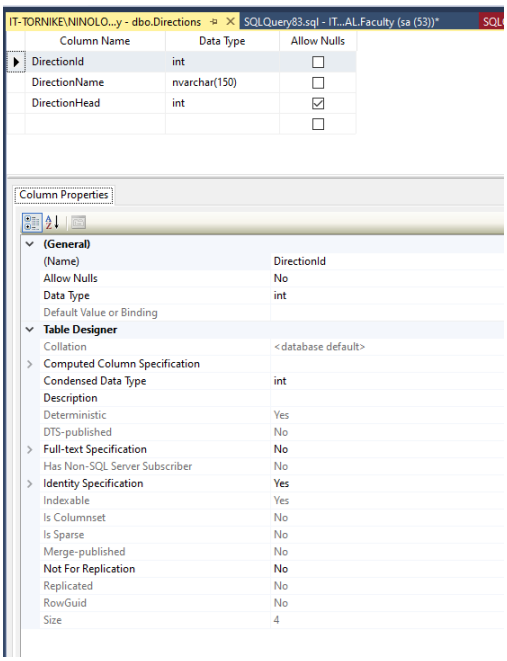


შედეგად გაიხსნება ცხრილის დიაინის რეჟიმი, სადაც შეგიძლიათ შეიყვანოთ სვეტის სახელი, მისი მონაცემთა ტიპი და ასევე მიუთითოთ იძლევა თუ არა სვეტი null-ებით შევსების უფლებას.



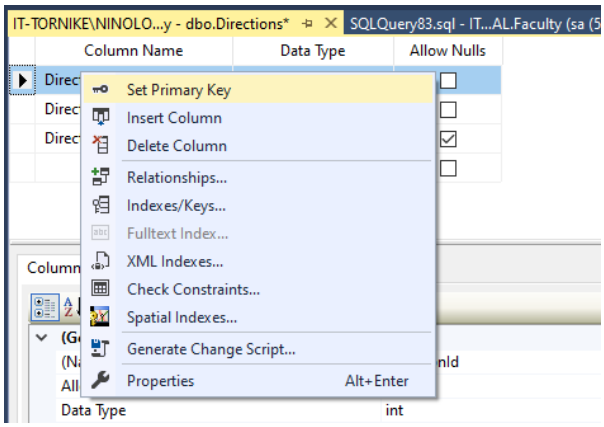
- Column Name: ჩაწერეთ სვეტის უნიკალური სახელი.
- Data Type: აირჩიეთ მონაცემთა ტიპი სვეტისთვის ჩამოსაშლელი სიიდან. აირჩიეთ შესაბამისი სიგრძე სტრიქონის მონაცემთა ტიპებისთვის.
- Allow Nulls: აირჩიეთ, დაუშვათ თუ არა Nulls თითოეული სვეტისთვის.

შეიყვანეთ სვეტების ინფორმაცია ყველა იმ სვეტისთვის, რომელიც გსურთ გქონდეთ თქვენს ცხრილში. ქვემოთ მოცემულია `Directions` ცხრილის სვეტები.

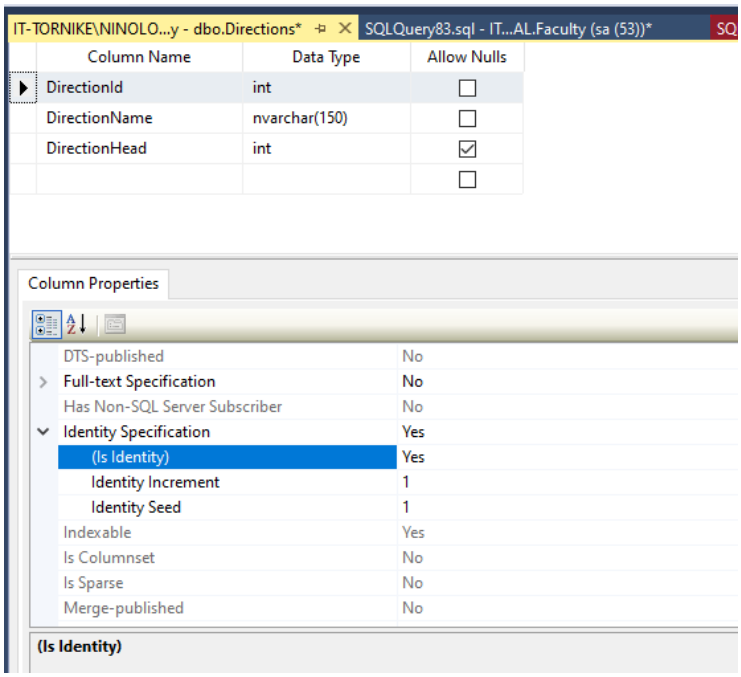


სვეტის დამატებითი პარამეტრებისა და შეზღუდვების დაყენება, როგორიცაა გამოთვლადი სვეტის პარამეტრები და აიდენთითი შეზღდვა შესაძლებელია Column Properties ფანჯრიდან. ამისათვის აირჩიეთ სვეტი და გადაგისთ Column Properties პანელში.

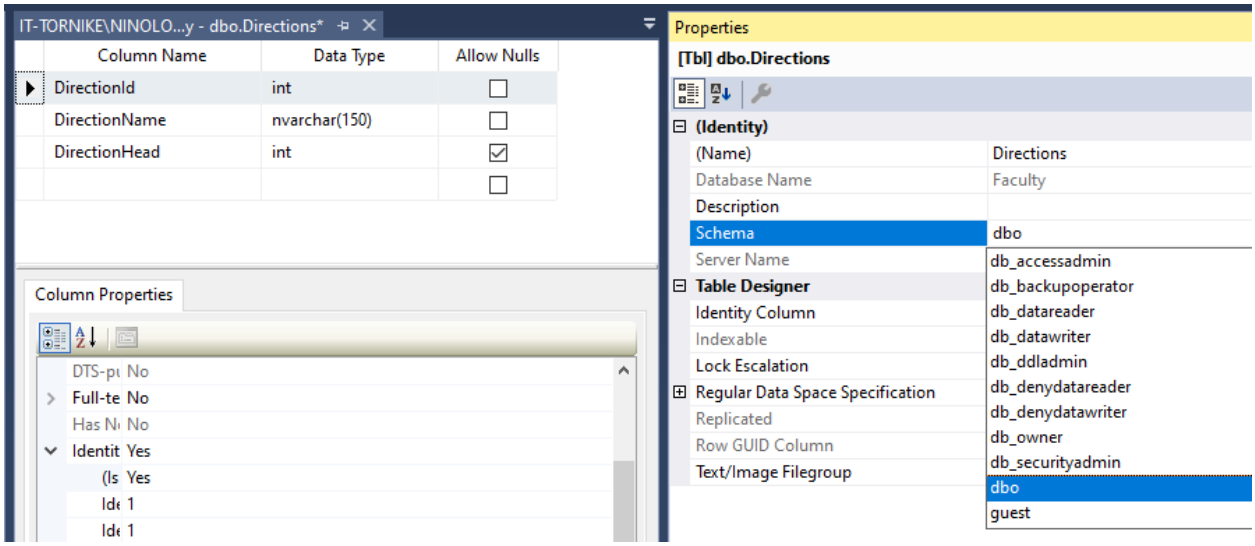
პირველადი გასაღების შეზღუდვის მოსანიშნად დააწკაპუნეთ მაუსის მარჯვენა ღილაკით სვეტზე და აირჩიეთ Set Primary Key:



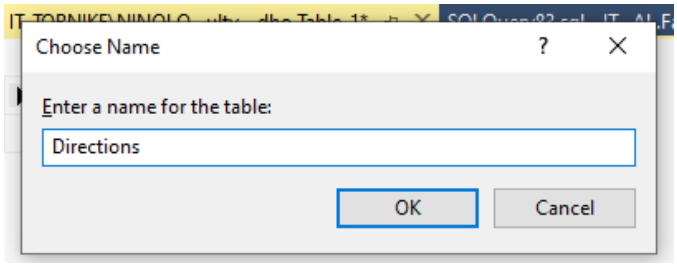
თქვენ შეგიძლიათ დააკონფიგურიროთ პირველადი გასაღები, ავტომატურად გადანომვრად (IDENTITY) სვეტად Column Properties ფანჯრიდან.



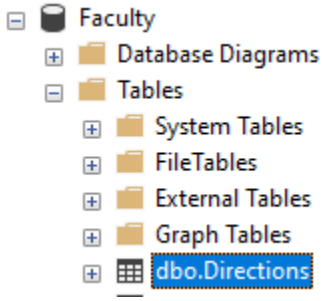
ნაგულისხმევად, ცხრილი იქმნება dbo სქემაში. ცხრილისთვის განსხვავებული სქემის მისანიჭებლად დააწკაპუნეთ მარჯვენა ღილაკით Table-Designer პანელზე და აირჩიეთ Properties ჩანართი, სქემის (Schema) ჩამოსაშლელი სიიდან აირჩიეთ შესაბამისი სქემა.



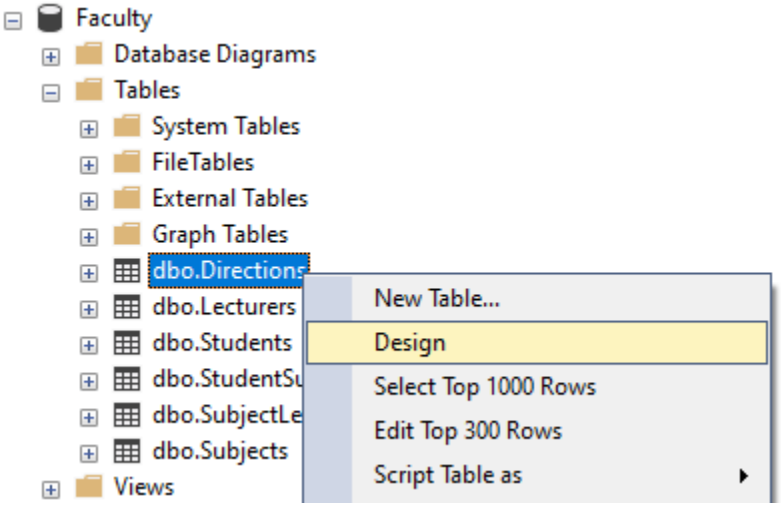
ახლა ფაილის მენიუდან აირჩიეთ შენახვა ამ ცხრილის შესაქმნელად, ან დააჭირეთ (Ctrl+S) კომბინაციას. შეიყვანეთ `Directions` როგორც ცხრილის სახელი და დააწკაპუნეთ OK



ახალი ცხრილის სანახავად განაახლეთ Tables საქაღალდე Object Explorer-ში. ცხრილი `Directions` ახლა ხელმისაწვდომია ცხრილების ჩანართში.



ამრიგად, თქვენ შექმნათ ახალი ცხრილი ცხრილის დიზაინ რეჟიმის გამოყენებით SSMS-ში, აღნიშნული რეჟიმის გახსნა უკვე არსებული ცხრილისთვის შესაძლებელია თუ ცხრილზე დავაჭერთ მარჯვენა ღილაკს და კონტექსტური მენიუდან ამოვარჩევთ Design ჩანართს



რის შემდეგაც კვლავ მოვხვდებით ნაცნობ ფანჯარაში სადაც შევძლებთ საჭირო შესწორებების შეტანას და შენახვას ცხრილში. მაგალითად განვიხილოთ კიდევ ერთი სვეტის ჩამატება არსებულ სვეტში,ჯერ აღნიშული მოქმედება შევასრულოთ სკრიპტის საშუალებით შემდგომ დავუბრუნდეთ დიზაინს.

ALTER TABLE ADD Columns სვეტის დამატება ცხრილში

თქვენ შეგიძლიათ დაამატოთ სვეტები არსებულ ცხრილს ALTER TABLE ბრძანების გამოყენებით.

ALTER TABLE ბრანება ასევე შეიძლება გამოყენებულ იქნას არსებული ცხრილის სვეტების სახელის გადასარქმევად ან წასაშლელად.

გამოიყენეთ ALTER TABLE ADD სკრიპტი არსებული ცხრილის ერთი ან მეტი სვეტის დასამატებლად:

```
ALTER TABLE [schema_name.]table_name
ADD column_name1 data_type constraint,
    column_name2 data_type constraint
    ...
    column_nameN data_type constraint;
```

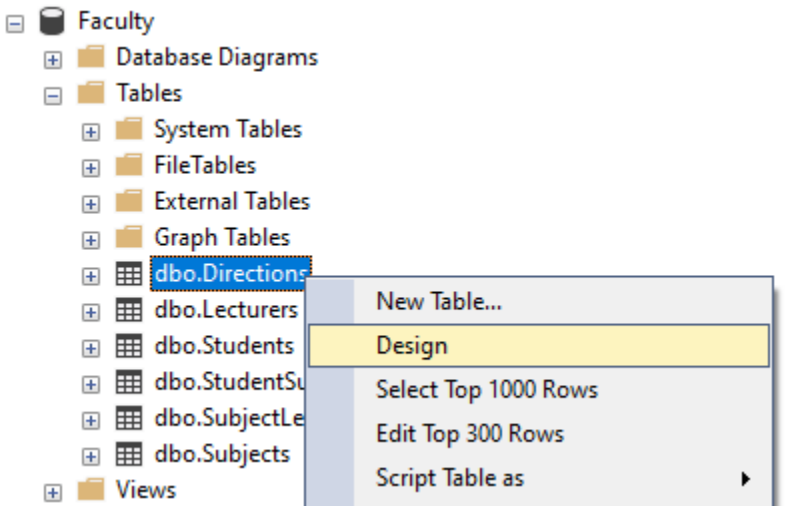
შემდეგი ამატებს ცხრილს Address სვეტს ტიპით varchar და ზომით 500.

```
GO
ALTER TABLE [dbo].[Directions]
ADD Address VARCHAR(500) NOT NULL
```

მძიმით გამოყოფის შემთხვევაში შესაძლებელია რამოდენიმე სვეტის დამატება ერთი Alter ბრძანებით

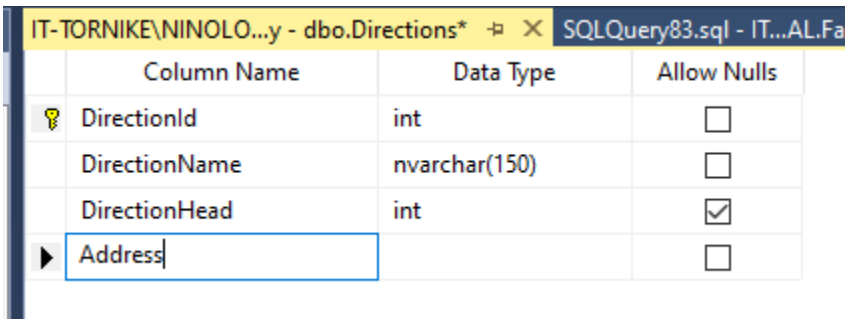
```
ALTER TABLE [dbo].[Directions]
ADD     Phone VARCHAR(30) NULL,
        Mail VARCHAR(30) NULL
```

აღნიშნული მოქმედების დიზაინში გასამეორებლად ცხრილზე დავაჭიროთ მარჯვენა ღილაკს და კონტექსტური მენიუდან ამოვარჩევთ Design ჩანართი:



გაიხსნება ცხრილის რედაქტირები რეჟიმი:

დააწკაპუნეთ პირველ ცარიელ უჯრედზე ბოლო სვეტის სახელის სვეტის ქვეშ და შეიყვანეთ სვეტის სახელი, როგორც ეს ნაჩვენებია ქვემოთ



შემდეგ სვეტში აირჩიეთ მონაცემთა ტიპი ჩამოსაშლელიდან და სიგრძე, საჭიროების შემთხვევაში. სავალდებულოდ შესავსებია თუ არა სვეტი (NULL/NOT NULL) და შეინახეთ File->Save Directions ან Crtl+s კომბინაციით.

სვეტის ან ცხრილის სახელის გადარქმევა

როგორც აქამდეც აღვნიშნეთ ობიექტების სახელები წარმოადგენენ მითითებლებს მატზე, შესაბამისად შეუძლებელია მათი სახელის ცვლილება alter ბრძანებით. ამისათვის უნდა გამოვიყენოთ სისტემის შენახული პროცედურა sp_rename.

```
EXEC sp_rename 'old_name', 'new_name' [, 'object_type'];
```

ცხრილის სახელის გადარქმევა: ცხრილის სახელის გადარქმევისთვის, 'old_name' უნდა იყოს მოცემული ცხრილის არსებული სახელი ან schema.table ფორმაში

სვეტის სახელის გადარქმევა: ცხრილში სვეტის სახელის გადარქმევის მიზნით, 'old_name' უნდა იყოს მოცემული table.column ან schema.table.column ფორმაში.

ინდექსის გადარქმევა: ინდექსის სახელის გადარქმევის მიზნით, 'old_name' უნდა იყოს მოცემული table.index ან schema.table.index ფორმაში.

შეზღუდვების გადარქმევა: შეზღუდვის სახელის გადარქმევისთვის, 'old_name' უნდა იყოს მოცემული schema.constraint ფორმაში.

შემდეგი გადაარქმევს Directions ცხრილს ცხრილად TemDirection.

```
EXEC sp_rename 'Directions', 'TempDirections'
```

და პირიქით:

```
EXEC sp_rename  'TempDirections' ,'Directions'
```

შემდეგი სკრიპტი გადაარქმევს 'Directions' ცხრილის Address სვეტს TempAddress.

```
EXEC sp_rename 'Directions.Address', 'TempAddress';
```

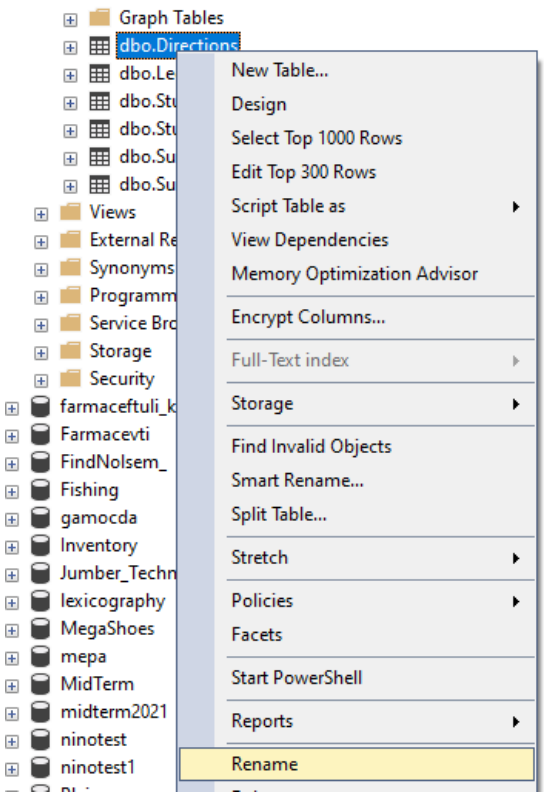
შევასრულოთ იგივე მოქმედება, დავაბრუნოთ საწყისი სახელი ამჯერად დავაკონკრეტოთ ობიექტის ტიპი:

```
EXEC sp_rename 'Directions.TempAddress', 'Address','COLUMN';
```

ცხრილისა და სვეტების სახელის გადარქმევა SSMS-ის გამოყენებით:

გახსენით SSMS და ჩამოშალეთ მონაცემთა ბაზის საქაღალდე.

აირჩიეთ და დააწკაპუნეთ მარჯვენა ღილაკით ცხრილზე ან სვეტზე, რომლის გადარქმევაც გსურთ და დააწკაპუნეთ სახელის გადარქმევა(Rename) . შეიყვანეთ ახალი სახელი არსებულ სახელზე გადაწერით და დააჭირეთ ENTER ღილაკს.



ცხრილის სვეტების/სვეტის წაშლა

გამოიყენეთ ALTER TABLE DROP COLUMN ბრძანება T-SQL-ის გამოყენებით ცხრილის ერთი ან მეტი სვეტის წასაშლელად.

```
ALTER TABLE [schema_name.]table_name
DROP column column_name1, column_name2,... column_nameN;
```

შემდეგი სკრიპტი წაშლის Directions ცხრილის სვეტს Address.

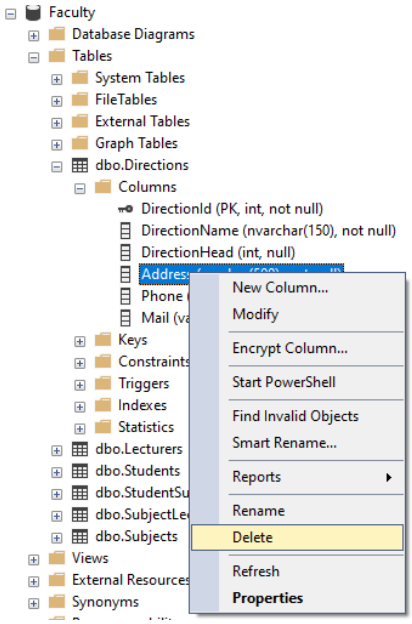
```
ALTER TABLE dbo.[Directions]
	DROP COLUMN Address;
```

შემდეგი სკრიპტი წაშლის Directions ცხრილის რამოდენიმე სვეტს.

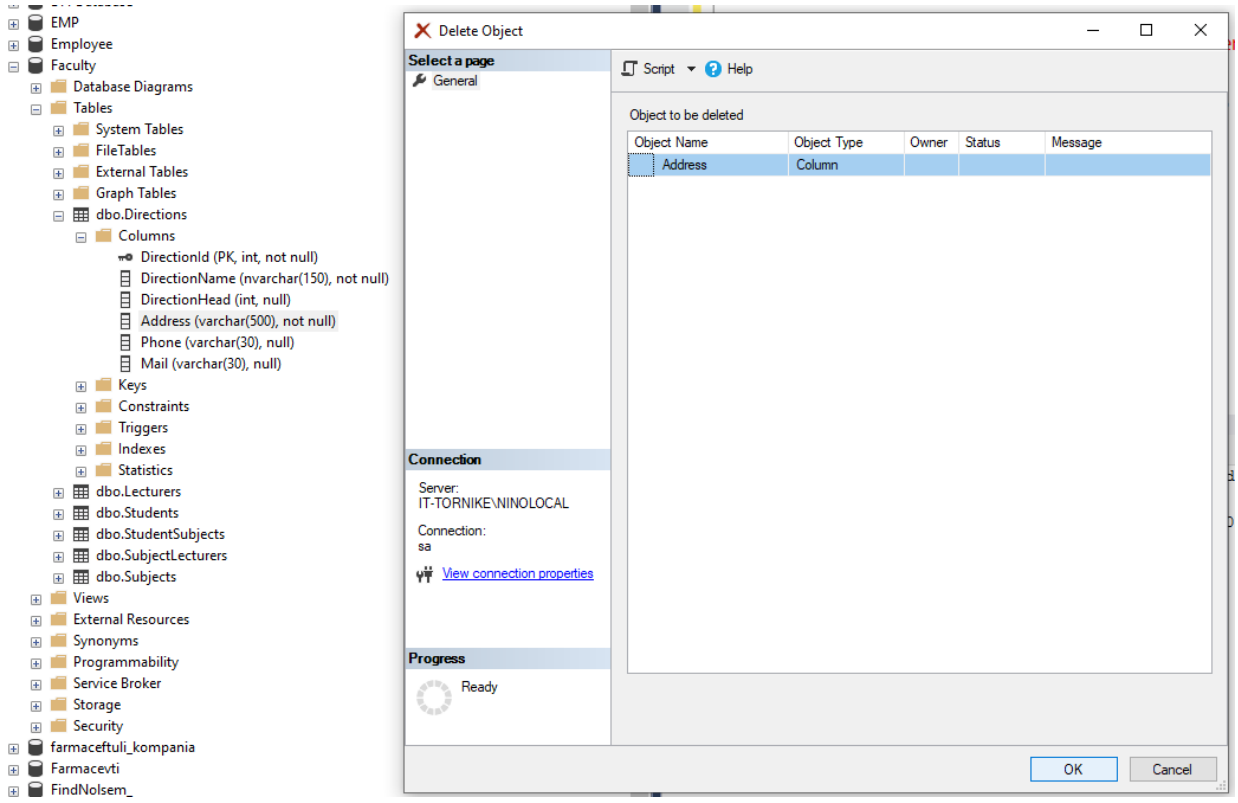
```
ALTER TABLE dbo.[Directions]
	DROP COLUMN [Phone],[Mail];
```

წაშალეთ სვეტები SSMS-ის გამოყენებით

დააწკაპუნეთ მაუსის მარჯვენა ღილაკით სვეტის სახელზე, რომლის წაშლაც გსურთ და კონტექსტური მენიუდან აარჩიეთ Delete ბრძანება:



გაიხსნება „Delete Object“ ფანჯარა, დავეთანხმოთ Okღილაკზე დაჭერით:



ცხრილის სვეტების მოდიფიკაცია

გამოიყენეთ ALTER TABLE ALTER COLUMN ბრძანება T-SQL-ის გამოყენებით ცხრილის ერთი ან მეტი სვეტის მოდიფიკაციისათვის.

სკრიპტის საშუალებით DirectionName სვეტის ზომა გაიზრდება 250 სიმბოლომდე:

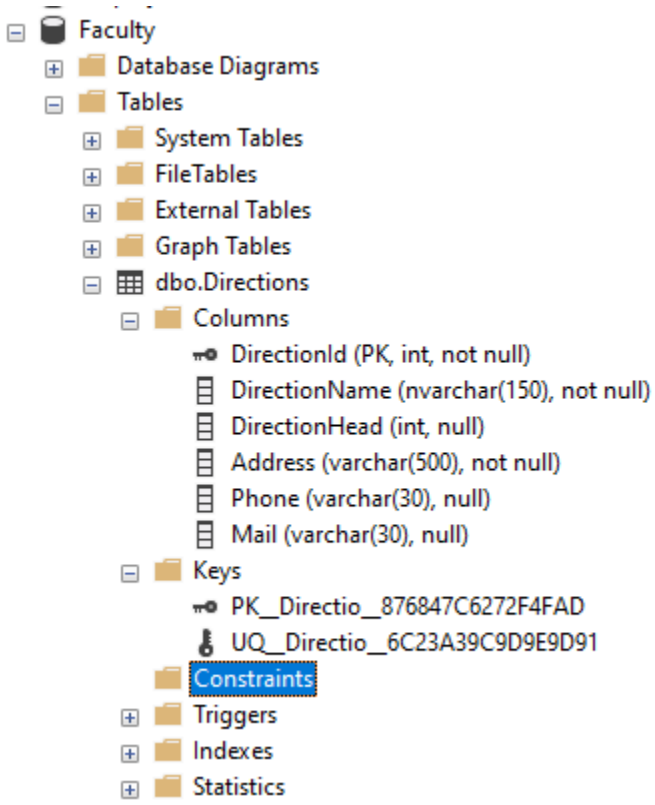
```
ALTER TABLE dbo.[Directions]
       ALTER COLUMN DirectionName VARCHAR (250)
```

აღნიშნული ცვლილებების შეტანა შესაძლებელია ჩვენთვის უკვე ნაცნობი ცხრილის დიზაინის რეჟიმიდან. თუ ცხრილზე დავაჭერთ მარჯვენა ღილაკს და კონტექსტური მენიუდან ამოვარჩევთ Design ჩანართს.

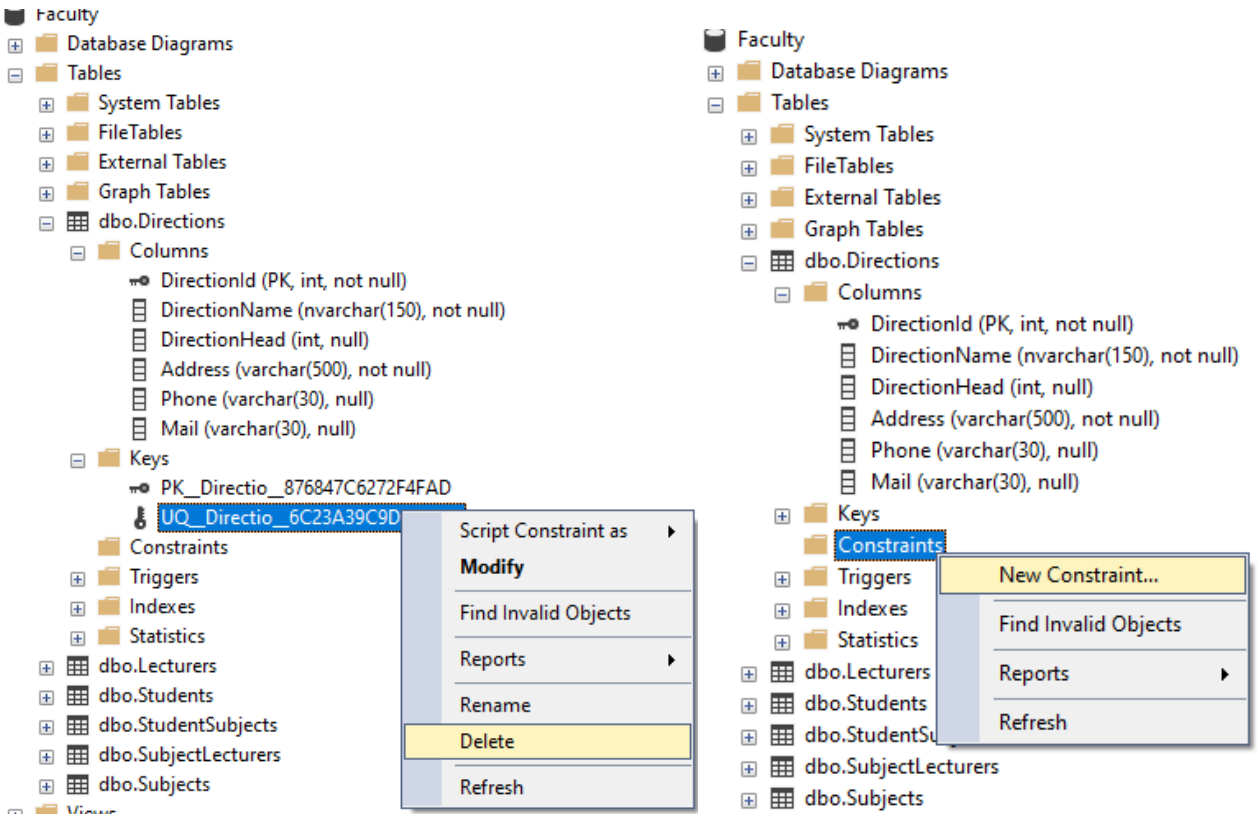
მოქმედებები შეზღუდვებზე

შეზღუდვების დამატება შესაძლებელია CREATE ან ALTER TABLE T-SQL ბრძანებების საშუალებით. CREATE TABLE მაგალითები უკვე ვნახეთ აღნიშნული თემის დასაწყისში. Alter ბრძანებით შეზღუდვების დამატება/წაშლა შესაძლებელია შემდეგი სკრიპტების გამოყენებით. აღსანიშნავია რომ მნიშვნელოვანია წაშლის დროს ვიცოდეთ შეზღუდვის სახელი, თუ თავად არ დაგვფირქმევია შეზღუდვის სახელი უნდა მოვიძიოთ ისინი დიზაინ რეჟიმში:

ცხრილის Keys ან Constraint ჩანართში:



რის შემდეგაც შეგვიძლია დავაკოპიროთ დასახელება და წავშალოთ შეზღუდვა სკრიპტის გამოყენებით. თუმცა შეზღუდვის წაშლა/დამატება ასევე შესაძლებელია დიზაინ რეჟიმიდან:



დავამატოთ ახალი შეზღუდვა რომ მიმართულების ხელმძღვანელის ნომერი DirectionHead იყოს უნიკალური, ანუ ერთ ადამიანს მხოლოდ ერთი მიმართულების ხელმძღვანელობა შეეძლოს და ერთი და იგივე პიროვნება არ დაინიშნოს რამოდენიმე მიმართულების ხელმძღვანელად:

```
ALTER TABLE [dbo].[Directions]
ADD CONSTRAINT UQ_DirectonHead UNIQUE([DirectionHead])
```

მას მერე რაც ჩვენს მიერ შერჩეული სახელით დავამატეთ შეზღუდვა UQ_DirectonHead. შესაძლებელია მისი წაშლა სწორედ ამ სახელის გამოყენებით:

```
ALTER TABLE [dbo].[Directions]
DROP CONSTRAINT UQ_DirectonHead
GO
```

დამატებით ცხრილების შექმნა

მოგროვილი ცოდნის საშუალებით დავამატოთ რამოდენიმე ცხრილი შესაბამისი შეზღუდვებით:

```
CREATE TABLE Subjects
(
SubjectId INT IDENTITY (1,1) ,
SubjectName NVARCHAR(150) NOT NULL,
```

```
DirectionId INT NOT NULL,
Credit INT,
CONSTRAINT PK_Subjects PRIMARY KEY (SubjectId)
)

CREATE TABLE Lecturers
(LecturerId INT IDENTITY PRIMARY KEY ,
LecturerName nvarchar(30) NOT NULL,
LecturerLastName nvarchar(30) NOT NULL,
Phone VARCHAR (20),
Email VARCHAR (30) CHECK (Email LIKE '%@%')
)


CREATE TABLE Students
(StudentId INT IDENTITY (1,1000) CONSTRAINT PK_Students PRIMARY KEY,
StudentName nvarchar(30) NOT NULL,
StudentLastName nvarchar(30) NOT NULL,
PersonalId VARCHAR(11) UNIQUE,
DateOfBirth DATE,
Phone VARCHAR (20),
Email VARCHAR (30),
City NVARCHAR(50) DEFAULT (N'თბილისი'),
DirectionId INT NOT NULL,
CONSTRAINT CK_Email CHECK (Email LIKE '%@%')
)


CREATE TABLE SubjectLecturers
(SubjectLecturerId INT IDENTITY PRIMARY KEY,
SubjectId INT NOT NULL,
LecturerId INT NOT NULL
CONSTRAINT UQ_SubjectLecturers UNIQUE(SubjectId,LecturerId)
)

CREATE TABLE StudentSubjects
(StudentSubjectId INT IDENTITY PRIMARY KEY,
StudentId INT NOT NULL,
SubjectId INT NOT NULL,
RegisterDate DATETIME NOT NULL DEFAULT (GETDATE()),
Midterm INT NULL,
Exam INT NULL,
Quiz INT NULL,
Proeject INT NULL,
IsPassed BIT NOT NULL DEFAULT(0)
)
```