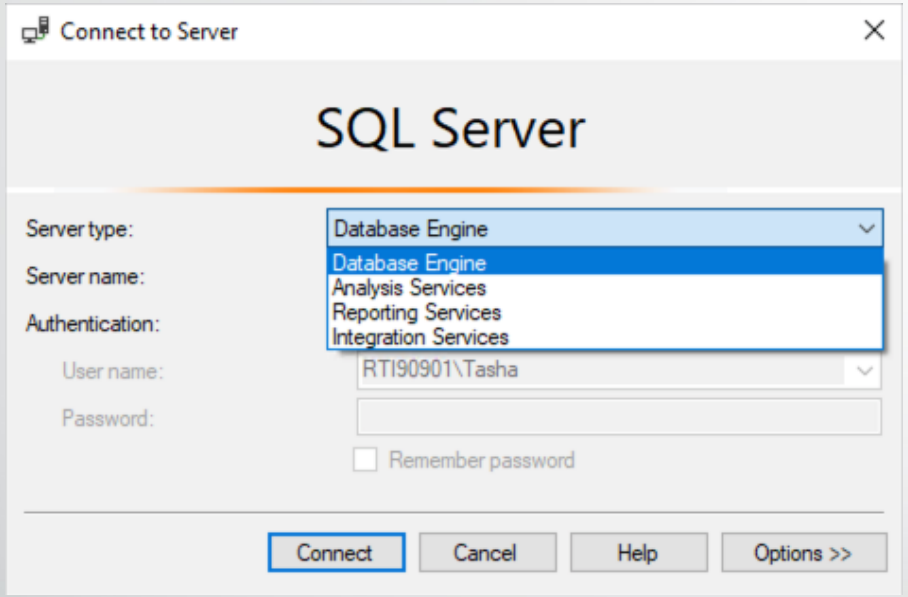
Elektrotehnički fakultet
Univerzitet u Beogradu
Katedra za računarsku tehniku i informatiku

Baze podataka 2

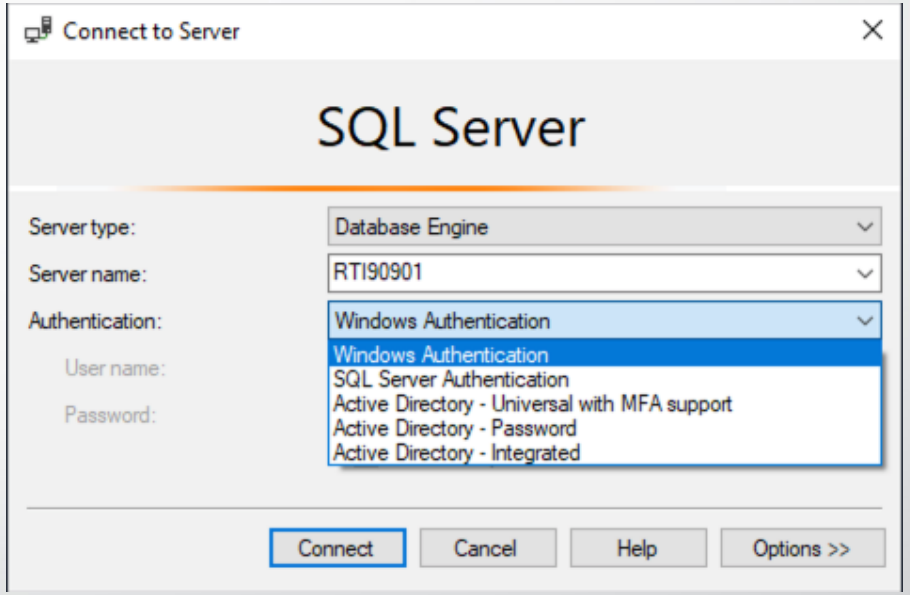
Uvod u SQL Server

2022/2023

SQL Server Management Studio - Konekcija ka serveru



Za Server type izaberemo Database Engine.



Za Authentication izaberemo Windows Authentication.

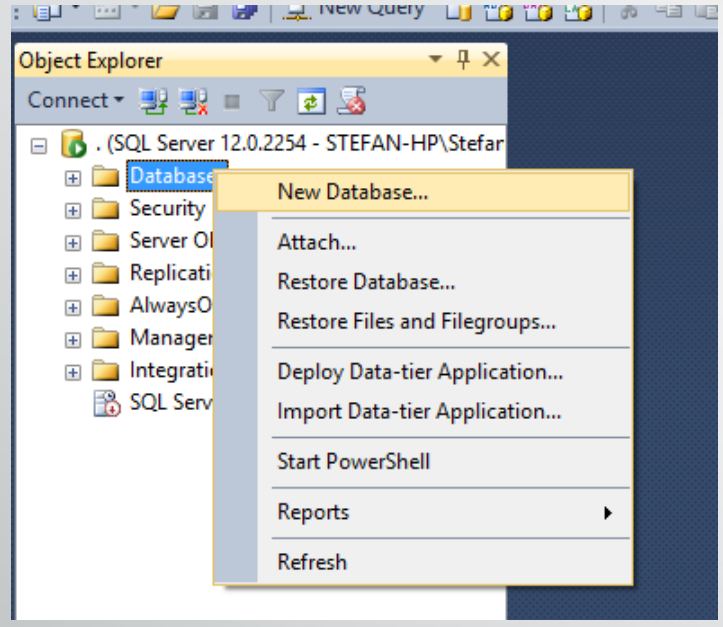
Klikom na Connect konektujemo se na ciljni server.

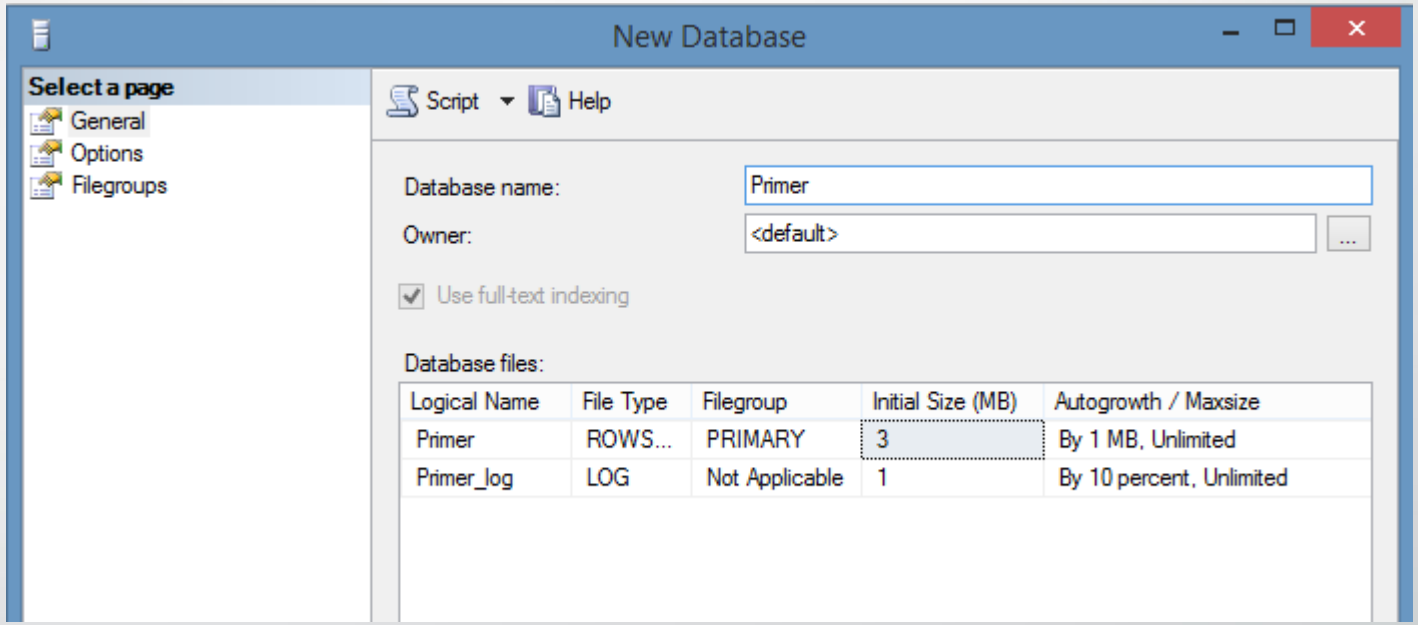
Rad sa bazom podataka

- SQL Server baza može biti kreirana, izmenjena, ili obrisana:
  1. grafički – SQL Server Management Studio
  2. korišćenjem upita

Rad sa bazom podataka

1. Kreiranje





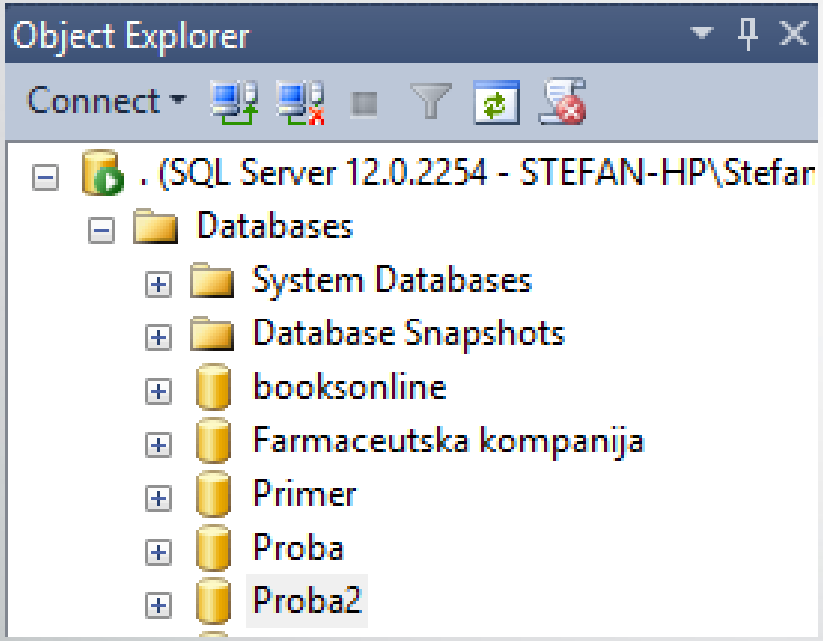
Desni klik na Database pa na New Database.

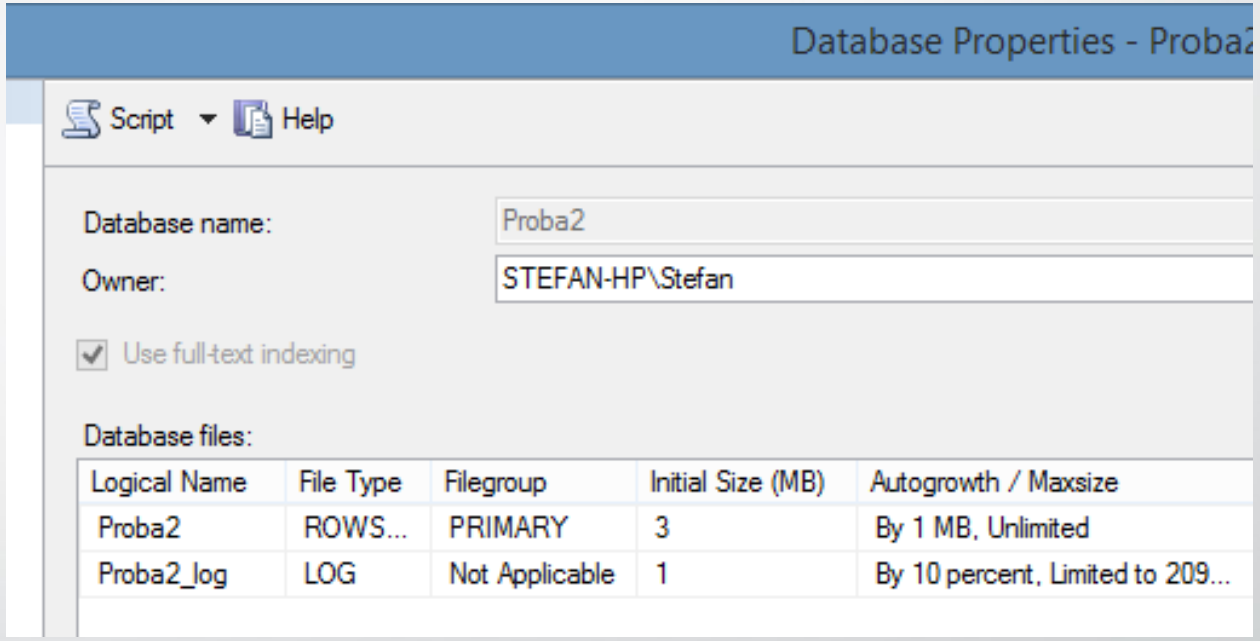
Unesemo Database name, u ovom slučaju Primer, i kliknemo na OK.

Sada je naša baza pod nazivom Primer kreirana.

Rad sa bazom podataka

```
Create database Proba2
```



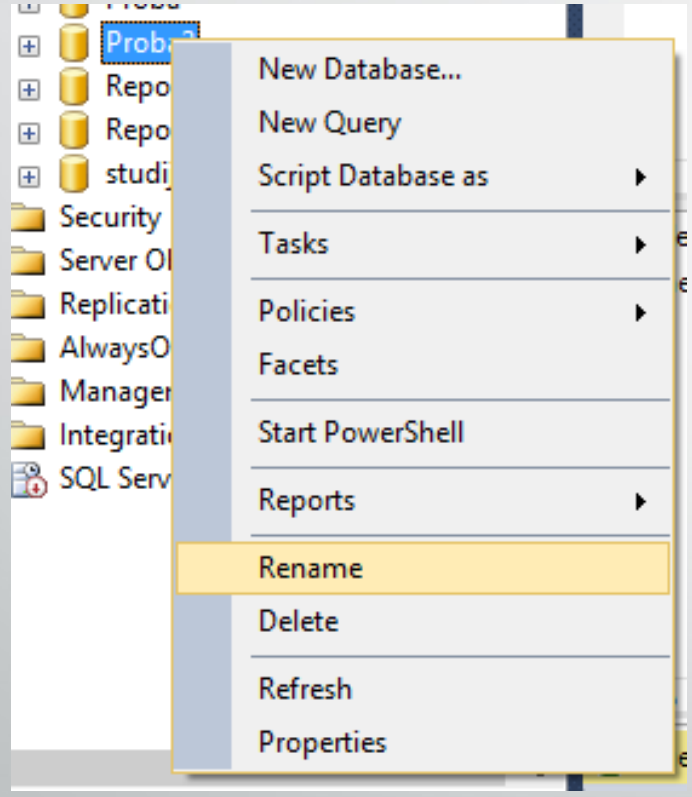


Kreirana je baza pod nazivom Proba2, i za nju su kreirana dva fajla:

- .mdf – Data File (sadrži same podatke)
- .ldf – Transaction Log File (koristi se za oporavak baze)

Rad sa bazom podataka

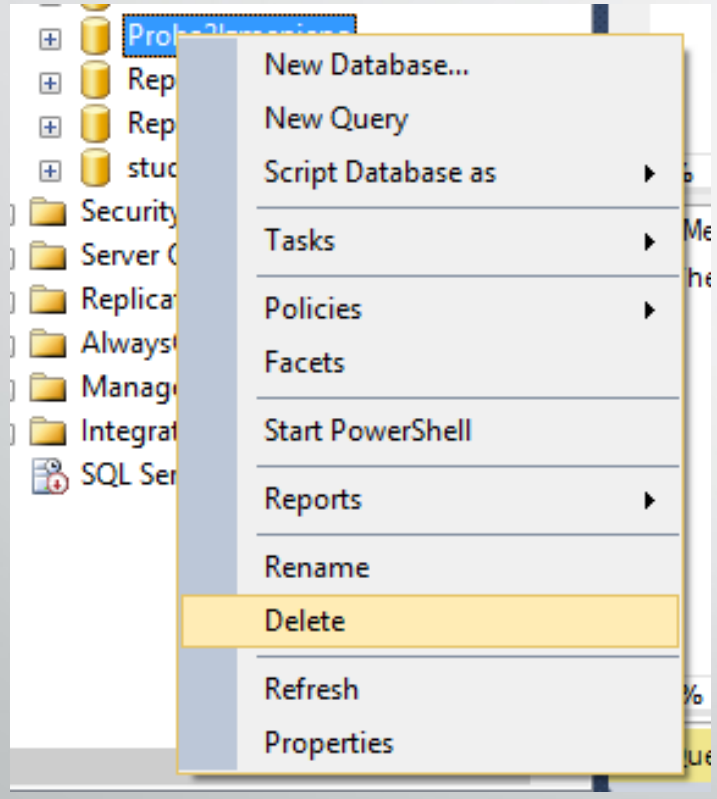
2. Izmena naziva




```
Alter database Proba2 Modify name = Proba2Izmenjeno

Execute sp_renamedb 'Proba2', 'Proba2Izmenjeno'
```

Rad sa bazom podataka

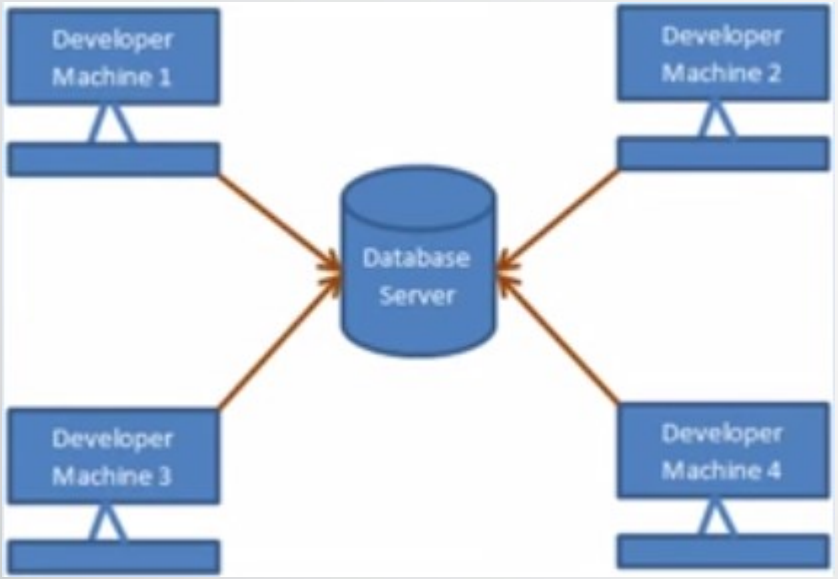
3. Brisanje




```
Drop database Proba2Izmenjeno
```

Brisanjem baze brišu se .mdf i .ldf fajl.

Rad sa bazom podataka



Ukoliko više korisnika pristupa istoj bazi na serveru, brisanje baze može biti problem.

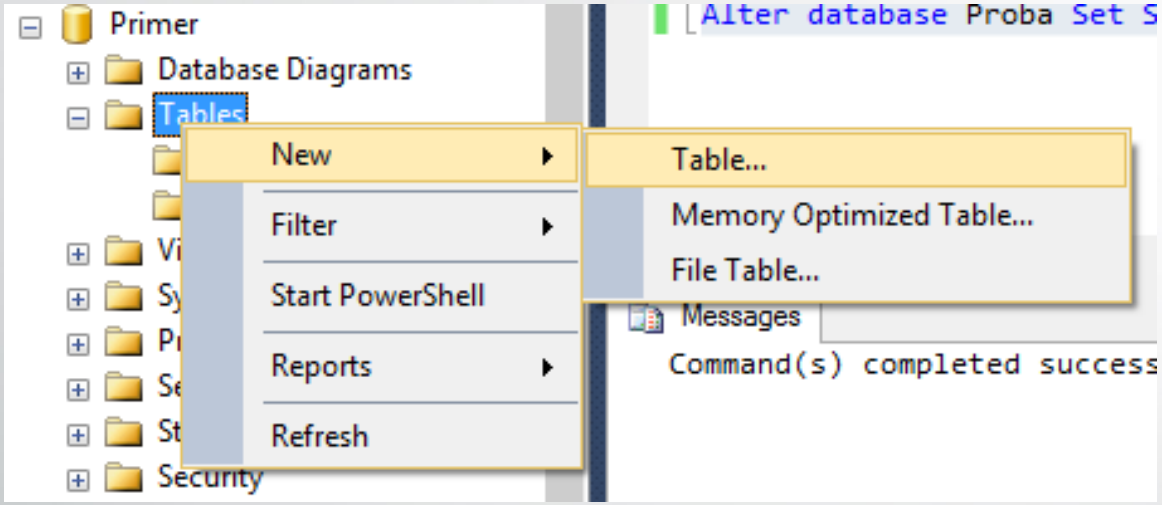
Ukoliko klijent na mašini 3 izvršava neki upit nad bazom i u toku tog izvršavanja klijent na mašini 2 pokuša da izbriše bazu neće uspeti i dobiće grešku:
Cannot drop database „DBName" because it is currently in use.

Korisnik koji želi da izbriše bazu, a nad njom radi više korisnika treba da izvrši sledeću komandu:

```
Alter database Proba Set SINGLE_USER With RollBack Immediate
```

`RollBack Immediate` govori da transakcije svih ostalih korisnika treba odmah Rollback-ovati.

Kreiranje tabele



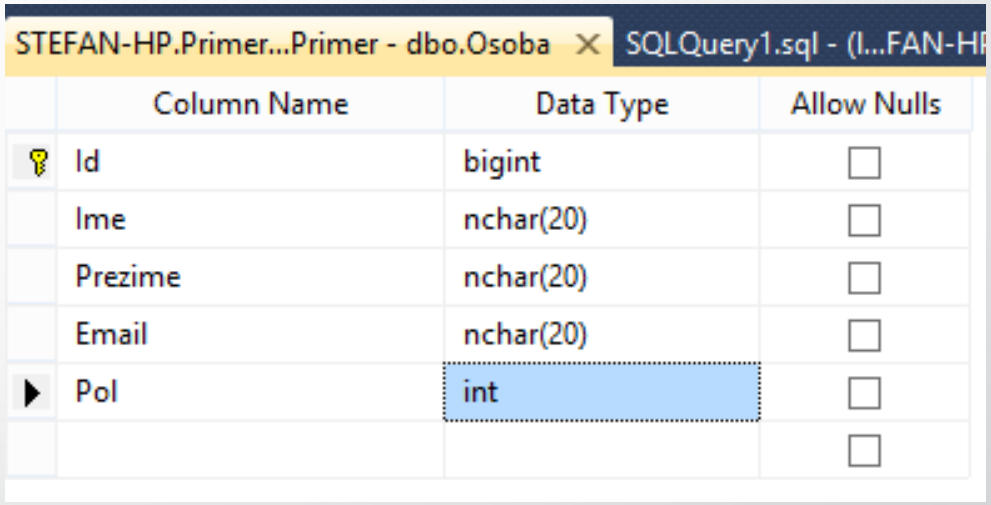


Tabela Osoba

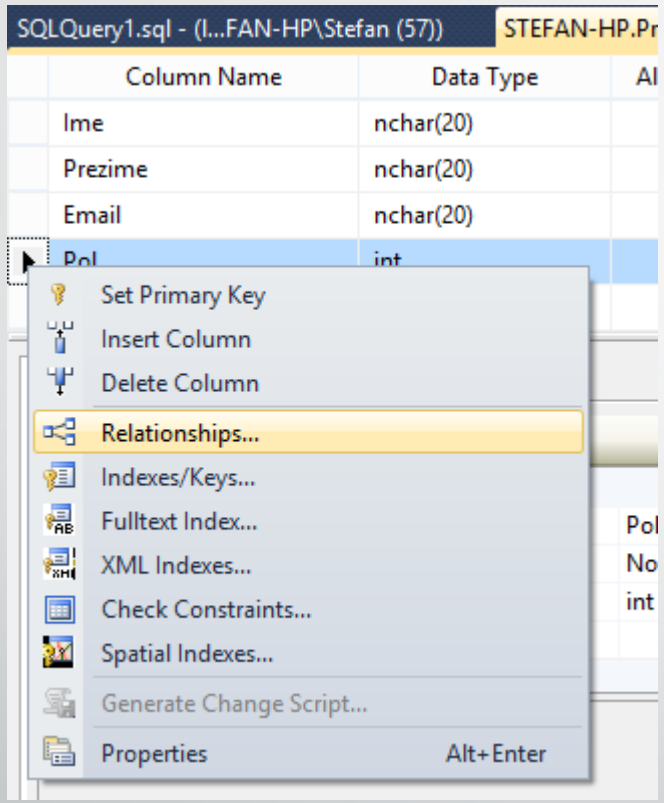
Kreiranje tabele

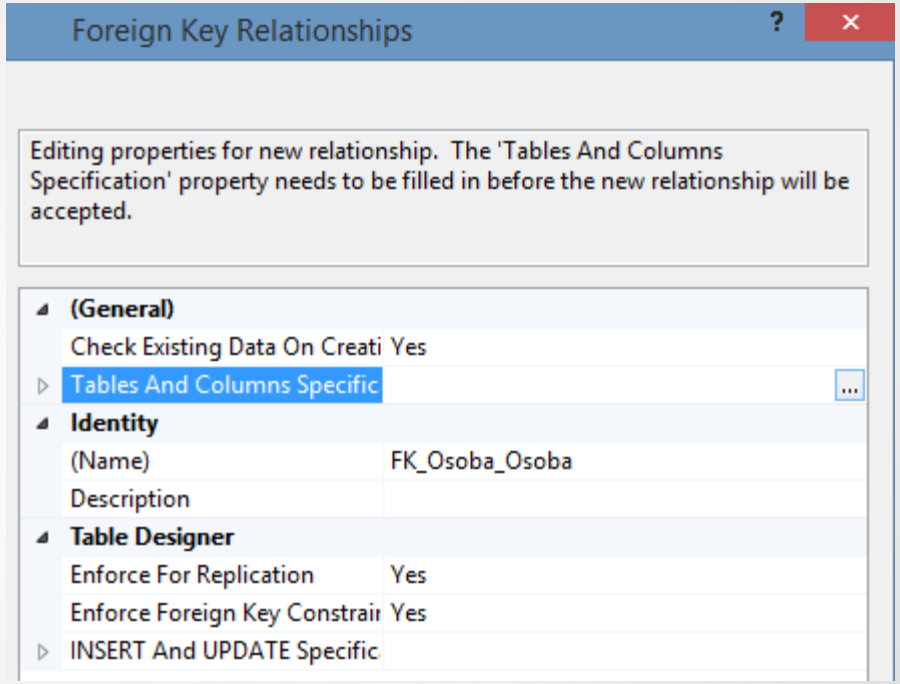
Kreiranje tabele Pol.

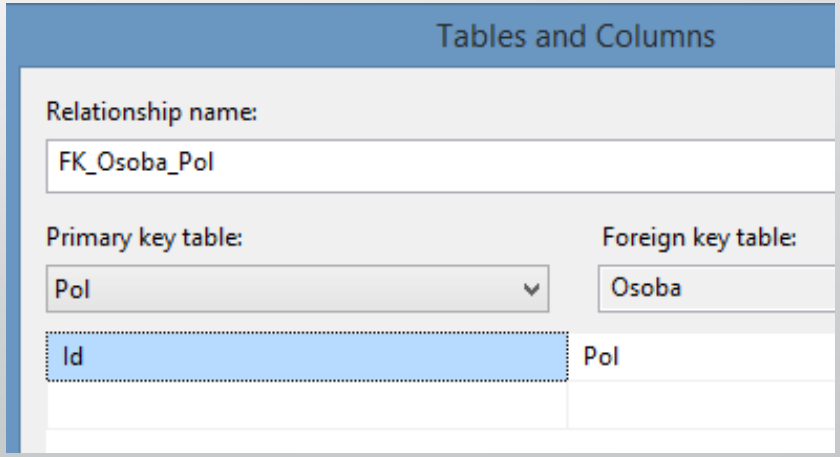
```
Use [Proba]
go
Create table Pol
(
Id int Primary Key,
Tip varchar(5) NOT NULL
)
```

`Use [Proba] go` – definiše bazu na koju će se upit odnositi

Strani ključevi

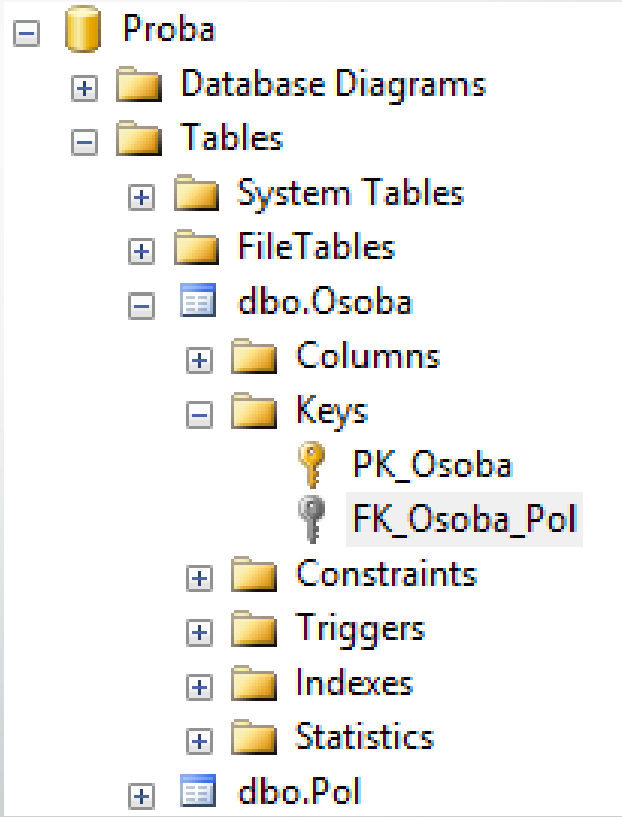






Strani ključevi

```
Alter table Osoba add constraint FK_Osoba_Pol
Foreign Key (Pol) references Pol(Id)
```



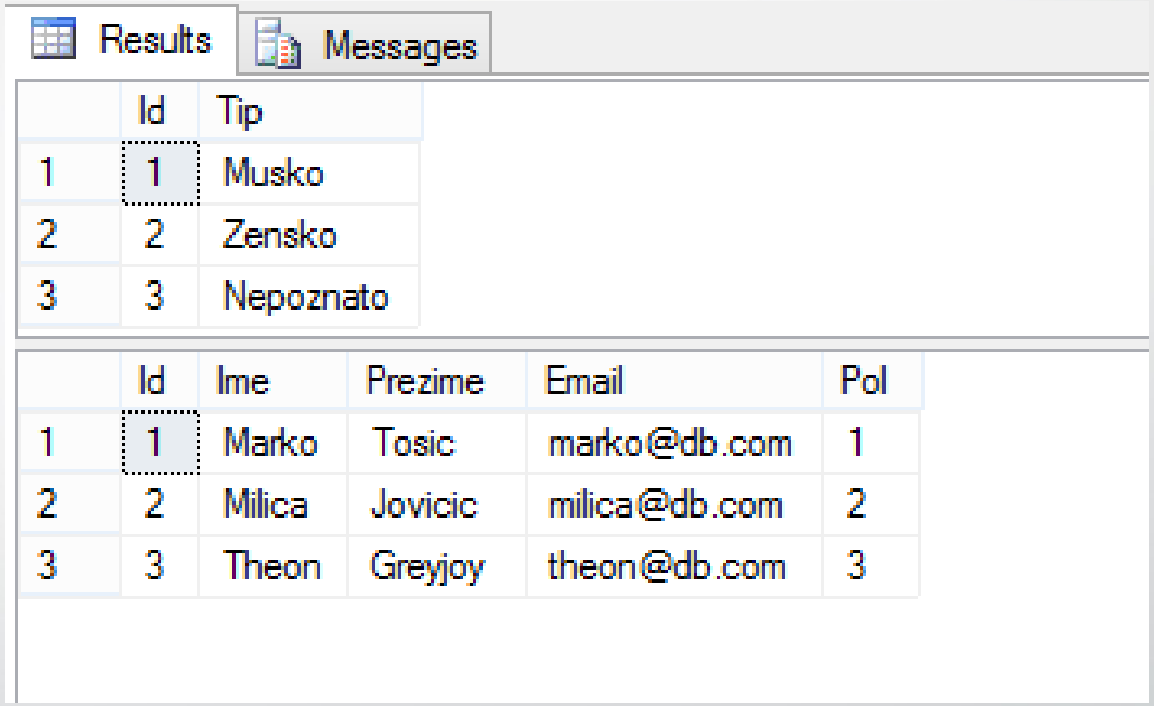
Default constraint

```
Insert into Pol (Id, Tip) Values (1,'Musko')
Insert into Pol (Id, Tip) Values (2,'Zensko')
Insert into Pol (Id, Tip) Values (3,'Nepoznato')

Alter table Osoba
Add constraint DF_Pol
Default 1 for Pol

Insert into Osoba (Id, Ime, Prezime, Email, Pol)
values (1, 'Marko', 'Tosic', 'marko@db.com', 1)
Insert into Osoba (Id, Ime,Prezime, Email,  Pol)
values (2, 'Milica', 'Jovicic', 'milica@db.com', 2)
Insert into Osoba (Id, Ime, Prezime, Email)
values (3, 'Theon', 'Greyjoy', 'theon@db.com')
```

Default constraint

```
select * from Pol
select * from Osoba
```

```
Alter table Osoba
Drop constraint DF_Pol
```



Results | Messages

	Id	Tip
1	1	Musko
2	2	Zensko
3	3	Nepoznato

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3

Referencijalni integriteti

- Referencijalni integriteti:
  1. No action – ne dozvoljava brisanje pa ne vrši nikakvu akciju (transakcija se rollback-uje)
  2. Cascade – briše odgovarajuće redove u povezanim tabelama
  3. Set Null – postavlja NULL vrednosti za odgovarajuće redove i kolone u povezanim tabelama
  4. Set Default - postavlja Default vrednosti za odgovarajuće redove i kolone u povezanim tabelama

Referencijalni integriteti

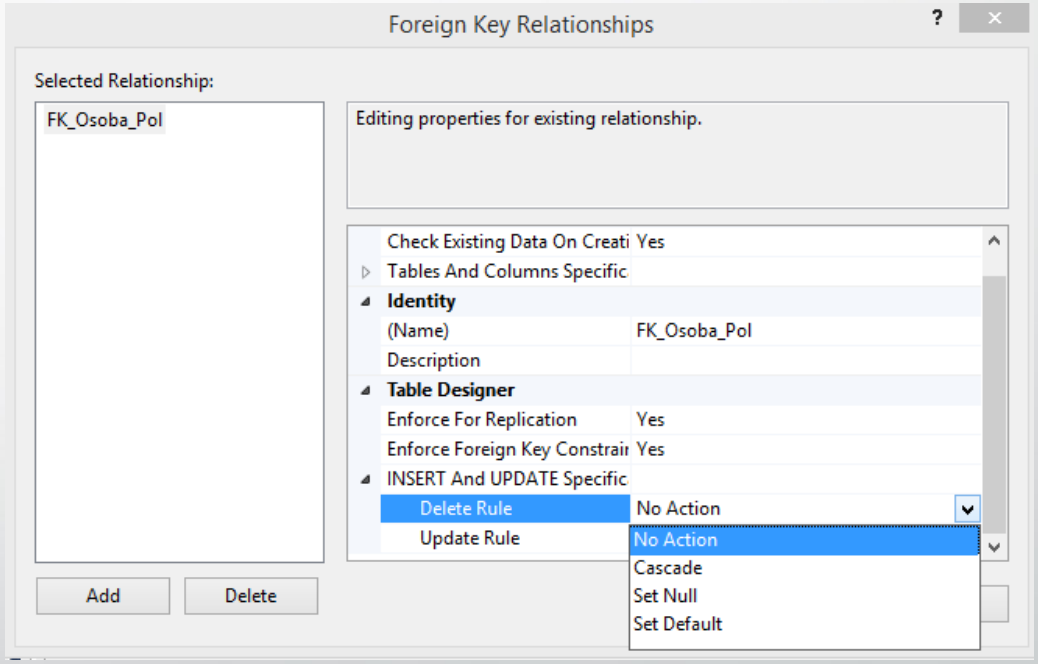
```
Insert into Pol(Id, Tip) Values(4, 'Vanzemaljac')

Insert into Osoba (Id, Ime, Prezime, Email, Pol)
values (4, 'Alien', '1', 'alien@db.com', 4)

Delete from Pol where Id = 4
```

```
The DELETE statement conflicted with the
REFERENCE constraint "FK_Osoba_Pol". The
conflict occurred in database "Proba", table
"dbo.Osoba", column 'Pol'.
```

Ovaj kod ne može biti izvršen jer je referencijalni integritet za Constraint „FK_Osoba_Pol“ automatski postavljen na „**NO ACTION**“

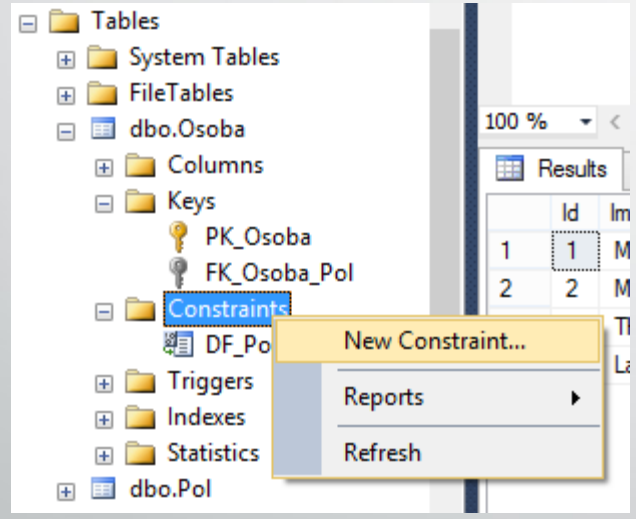
Referencijalni integriteti

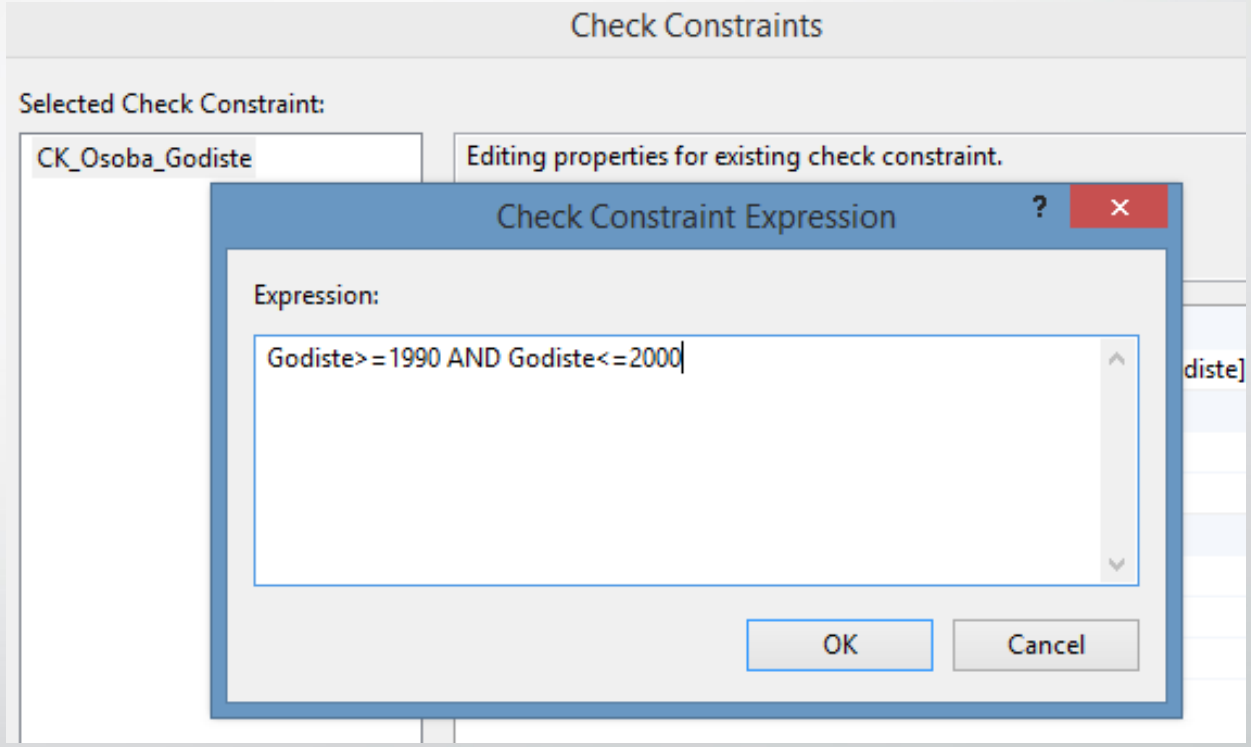
Iz tog razloga postavljamo drugačije referencijalne integritete.



Check constraints

```
Alter table Osoba
Add Godiste int
```





Check constraints

```
Insert into Osoba values (4, 'Lazar', 'Petrovic', 'lazar@db.com', 1, 2200)
```

```
The INSERT statement conflicted with the CHECK constraint "CK_Osoba_Godiste".
The conflict occurred in database "Proba", table "dbo.Osoba", column
'Godiste'.
```

Vrednost 2200 je van opsega, pa ovakav red ne možemo uneti u tabelu Osoba.

Check constraints

Column Name	Data Type	Allow Nulls
Prezime	nchar(20)	☐
Email	nchar(20)	☐
Pol	int	☐
Godiste	int	☑

```
Insert into Osoba values (4, 'Jovana', 'Peric', 'jovana@db.com', 1, NULL)
```

	Id	Ime	Prezime	Email	Pol	Godiste
1	1	Marko	Tosic	marko@db.com	1	1991
2	2	Milica	Jovicic	milica@db.com	2	1991
3	3	Theon	Greyjoy	theon@db.com	3	1991
4	4	Jovana	Peric	jovana@db.com	1	NULL

Rezultat upoređivanja NULL i neke poznaze vrednosti je Unknown što takođe prolazi Check constraint.

Iz tog razloga je dodat novi red u tabelu Osoba.

Identity column

Identity column

Ukoliko želimo da određena polja dobijaju automatski svoje vrednosti tada koristimo Identity column.

STEFAN-HP.Proba - dbo.Osoba | STEFAN-HP.Proba - dbo.Zaposleni

Column Name	Data Type	Allow Nulls
Id	bigint	☐
Ime	nchar(20)	☐
Prezime	nchar(20)	☐
Email	nchar(20)	☐
Pol	int	☐
		☐

Identity Specification	No
(Is Identity)	No
Identity Increment	Yes
Identity Seed	No

Identity Seed je inicjalna vrednost posmatrane kolone.

Identity Increment je vrednost koja se dodaje na prethodnu vrednost kako bi se dobila nova.

Identity column

Sada moržemo unositi nove redove bez specificiranja Id-a Osobe, jer će se on automatski postavljati (kolona Id je Identity column).

```
Insert into Osoba values ('Goran', 'Mitrovic', 'goran@db.com',1)
```

Results | Messages

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1

Identity column

Ukoliko želimo da postavimo eksplicitnu vrednost za kolonu koja je Identity column treba uraditi sledeće:

```
Set identity_insert Osoba on

Insert into Osoba(Id, Ime, Prezime, Email, Pol)
values (30,'Aleksandar', 'Kostic', 'aleksandar@db.com',1)

Insert into Osoba(Id, Ime, Prezime, Email, Pol)
values (25,'Jovan', 'Kostic', 'jovan@db.com',1)
```

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1
5	25	Jovan	Kostic	jovan@db.com	1
6	30	Aleksandar	Kostic	aleksandar@db.com	1

Identity column

Vraćanje na automatsko biranje vrednosti vrši se sa:

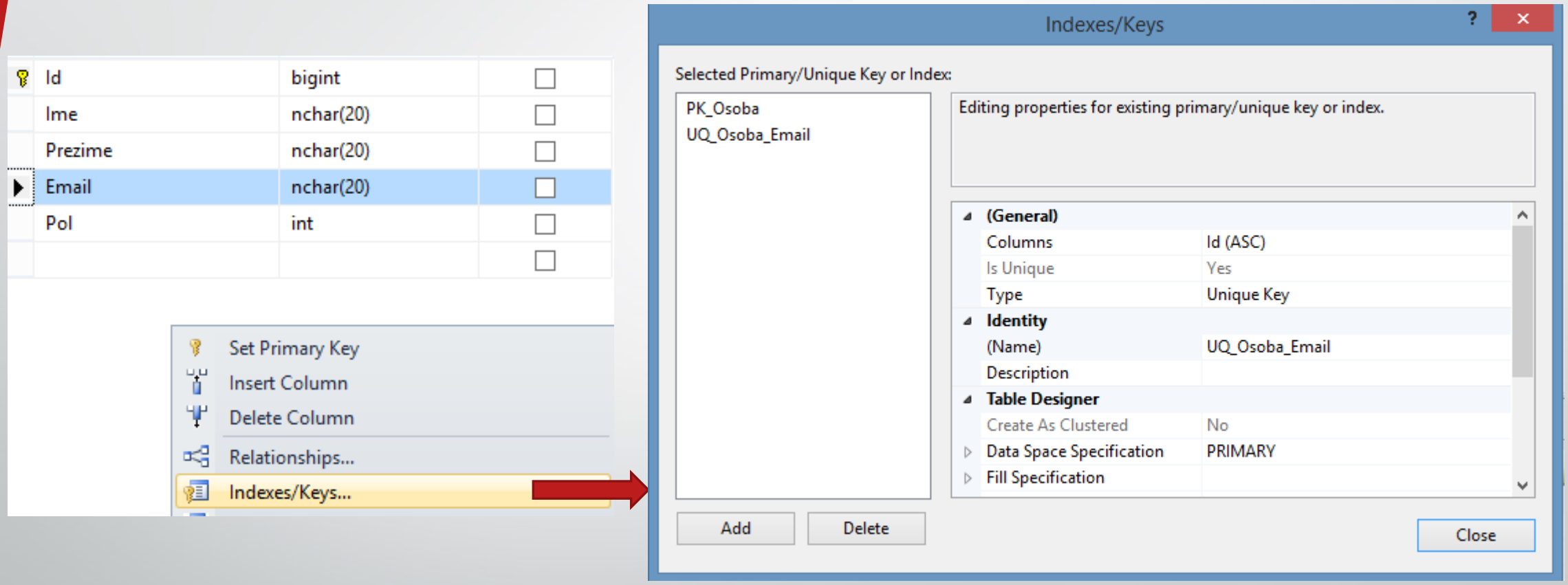
```
Set identity_insert Osoba off

Insert into Osoba values ('Jelena', 'Jovicic', 'jelena@db.com',2)
```

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1
5	25	Jovan	Kostic	jovan@db.com	1
6	30	Aleksandar	Kostic	aleksandar@db.com	1
7	31	Jelena	Jovicic	jelena@db.com	2

Za izmenu seed-a koristi se:

```
DBCC Checkident(Osoba, RESEED, 50)
```

Unique key constraint



Unique key constraint

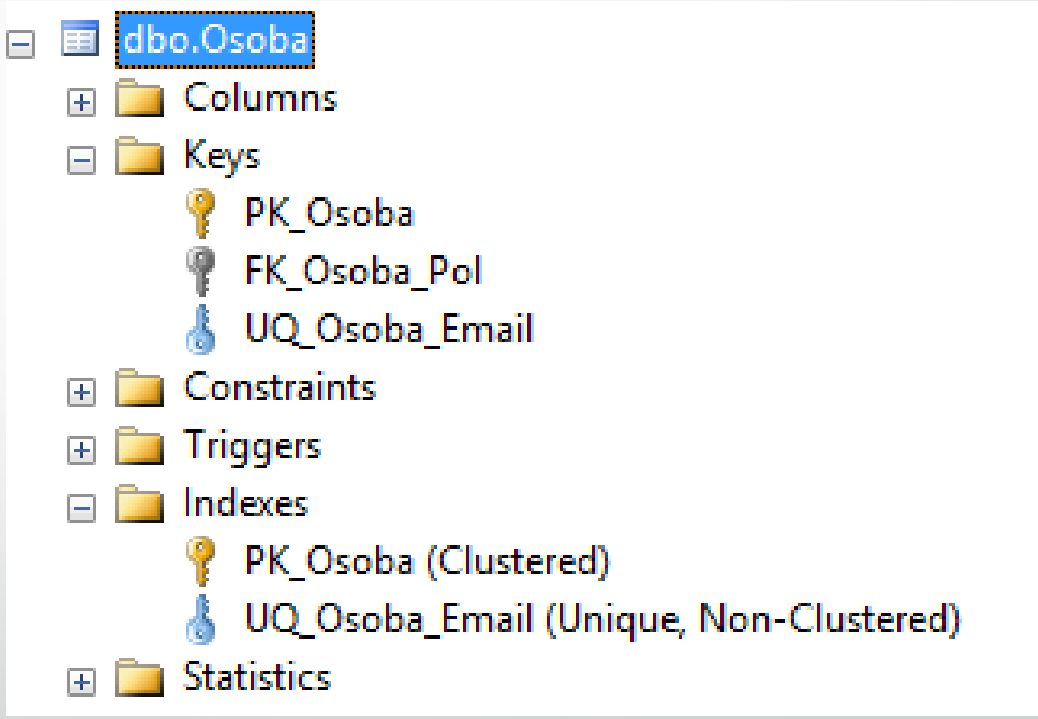
Isti efekat postižemo sledećom SQL skriptom:

```
Alter table Osoba
Add constraint UQ_Osoba_Email
Unique(Email)
```

U pozadini se stvara i Unique, Non-Clustered index.

Brisanje ograničenja:

```
Alter table Osoba
Drop Constraint UQ_Osoba_Email
```



Pogledi

Kreiranje pogleda:

```
Create view ZaposleniMuskarci
as
Select Id, Ime from Zaposleni
Where Pol = 'M'
```

Definiciju pogleda možemo videti koristeći:

```
Execute sp_helptext ZaposleniMuskarci
```

	Text
1	Create view ZaposleniMuskarci
2	as
3	Select Id, Ime from Zaposleni
4	Where Pol = 'M'

	Id	Ime	Plata	Pol
1	1	Ivan	3000	M
2	2	Milica	2000	Z
3	3	Marija	1000	Z

Tabela Zaposleni

	Id	Ime
1	1	Ivan

Pogled ZaposleniMuskarci

Pogledi

```
Create view OsobaPol
as
Select O.Ime, O.Prezime, O.Email, P.Tip
From Osoba O, Pol P
Where O.Pol = P.Id
```

	Id	Tip
1	1	Musko
2	2	Zensko
3	3	Nepoznato

Tabela Pol

	Id	Ime	Prezime	Email	Pol
1	1	Marko	Tosic	marko@db.com	1
2	2	Milica	Jovicic	milica@db.com	2
3	3	Theon	Greyjoy	theon@db.com	3
4	4	Goran	Mitrovic	goran@db.com	1

Tabela Osoba

	Ime	Prezime	Email	Tip
1	Marko	Tosic	marko@db.com	Musko
2	Milica	Jovicic	milica@db.com	Zensko
3	Theon	Greyjoy	theon@db.com	Nepoznato
4	Goran	Mitrovic	goran@db.com	Musko

Pogled OsobaPol

Pogledi

Pogledi se koriste:

- Radi skrivanja detaljnih informacija, i predstavljanja agregiranih podataka krajnjem korisniku
- Pojednostavljenja pisanja upita
- Kao mehanizam za implementaciju sigurnosti po redovima i kolonama (skrivanje redova i kolona)

Pogledi se ne čuvaju na disku kao tabele, već predstavljaju SQL kod koji se uvek izvršava kada izvršimo upit nad pogledom. (Paziti na optimizaciju!)

Kao i na tabele sledeći upiti se koriste i za poglede:

```
Alter View Zaposleni Statement
```

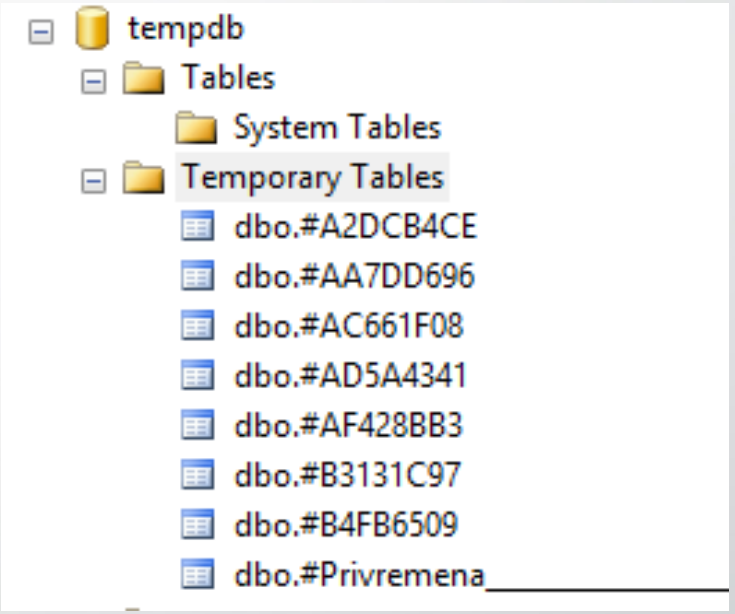
```
Drop View ViewName
```

Privremene tabele

- Obične tabele:
  - Kreirane sa CREATE TABLE NazivTabele
  - Brisanje sa DROP TABLE NazivTabele
  - Nalaze se u folderu **Tables** u bazi u kojoj je kreirana
- Privremene tabele:
  - Kreirane sa CREATE TABLE #NazivTabele
  - Može eksplicitno iskazom DROP TABLE, ali karakteristično za nju je da ona postoji sve dok postoji konekcija koja ju je napravila (ukoliko zatvorimo .sql fajl u Management Studiu privremena tabela će nestati jer se zatvaranjem ovog fajla zatvorila i konekcija)
  - Raspoloživa je samo za konekciju koja ju je kreirala
  - Nalaze se u folderu **Temporary Tables** sistemske baze **tempdb**

Privremene tabele

```
Create table #Privremena
(
Id int Primary Key,
Ime varchar(20)
)
Insert Into #Privremena values (1,'Prvi')
Insert Into #Privremena values (2,'Drugi')
```



Proveravanje da li je privremena tabela napravljena upitom:

```
Select name from tempdb..sysobjects
Where name like '#Privremena%'
```

```
Select * from #Privremena
```

Prva konekcija:

	Id	Ime
1	1	Prvi
2	2	Drugi

Druga konekcija:

```
Msg 208, Level 16, State 0, Line 1
Invalid object name '#Privremena'.
```

Privremene tabele

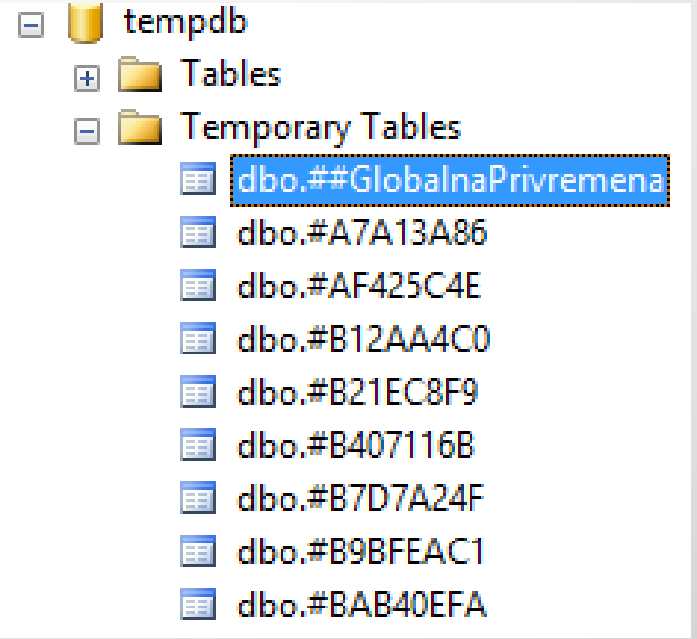
Ukoliko se privremena tabela napravi unutar procedure njena instanca nestaje kada se izvršavanje procedure završi.

```
Create proc P
As
Begin
Create table #Privremena
(
Id int Primary Key,
Ime varchar(20)
)
Insert Into #Privremena values (1,'Prvi')
Insert Into #Privremena values (2,'Drugi')
Select * from #Privremena
End

execute P
```

Kada napravimo istu privremenu tabeli iz različitih konekcija napraviće se tabela sa istim početnim delom naziva, a različitim automatski generisanim brojem na kraju.

Privremene tabele

- Globalne privremene tabele se kreiraju koristieći prefiks ##.
- Ovakve tabele nemaju izgenerisan broj na kraju jer njeno ime mora biti jedinstveno. Moguće ih je koristititi iz svih konekcija, i one se brišu kada nestane poslednja konekcija sa bazom.




```
Create table ##GlobalnaPrivremena
(
Id int Primary Key,
Ime varchar(20)
)
```

Indeksi

Glavna svrha indeksa je povećanje performansi.
Postojanje indeksa omogućava optimizaciju upita nad bazom.

Može se vršiti poređenje sa telefonskim imenikom, gde su osobe sortirane po prezimenu, zatim po imenu kod onih osoba koje imaju isto prezime.

Indeksi

- Vrste indeksa:
  1. Clustered
  2. Nonclustered
  3. Unique
  4. Filtered
  5. Index with included columns
  6. Index on computed columns
  7. XML
  8. Full Text
  9. Spatial
  10. Columnstore

Primer baze podataka

- Data je baza podataka sa sledećim relacijama:

  Status (status_code, status_desc)

  Category (category_no, category_desc, category_code)

  Region (region_no, region_name, street, city, state_prov, country, mail_code, phone_no, region_code)

  Corporation (corp_no, corp_name, street, city, state_prov, country, mail_code, phone_no, expr_dt, region_no, corp_code)

  Provider (provider_no, provider_name, street, city, state_prov, mail_code, country, phone_no, issue_dt, expr_dt, *region_no*, provider_code)

  Member (member_no, lastname, firstname, middleinitial, street, city, state_prov, country, mail_code, phone_no, photograph, issue_dt, expr_dt, *region_no, corp_no,* prev_balance, curr_balance, member_code)

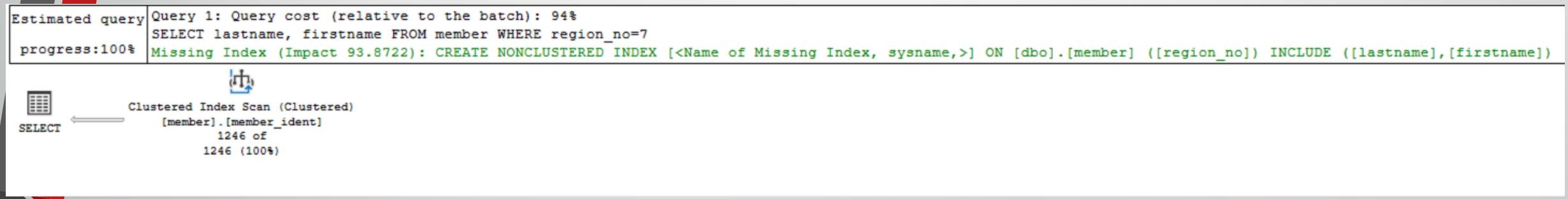
  Statement (statement_no, *member_no*, statement_dt, due_dt, statement_amt, statement_code)

  Charge (charge_no, *member_no, provider_no, category_no,* charge_dt, charge_amt, *statement_no*, charge_code)

  Payment (payment_no, *member_no*, payment_dt, payment_amt, statement_no, payment_code)

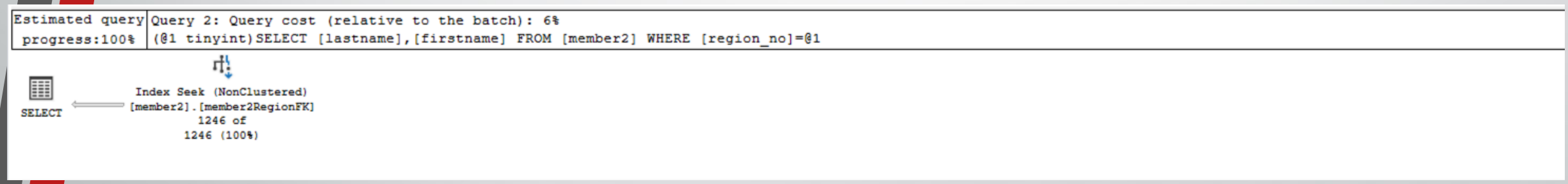
Izvršavanje upita bez odgovarajuceg indeksa

```
SELECT lastname, firstname
FROM member
WHERE region_no=7
```



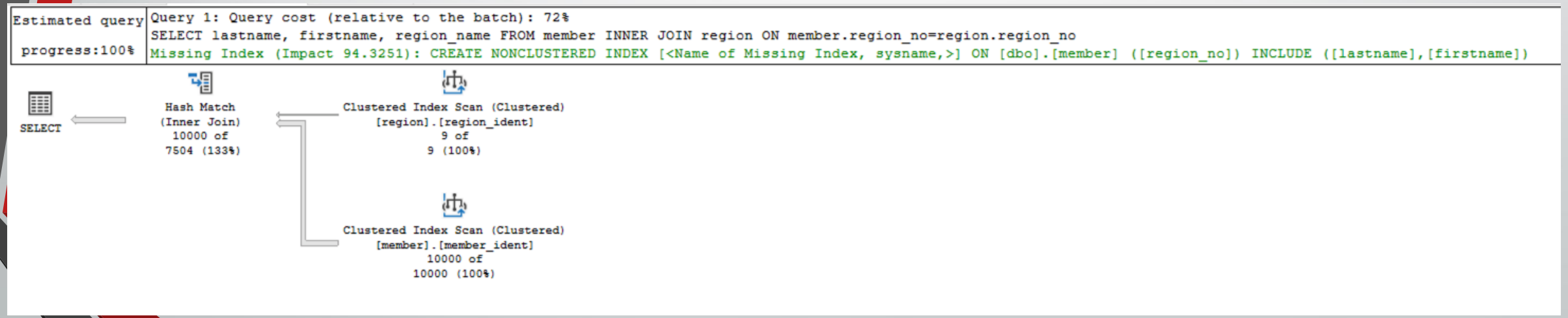
Izvršavanje upita kad indeks postoji

```
SELECT lastname, firstname
FROM member2
WHERE region_no=7
```



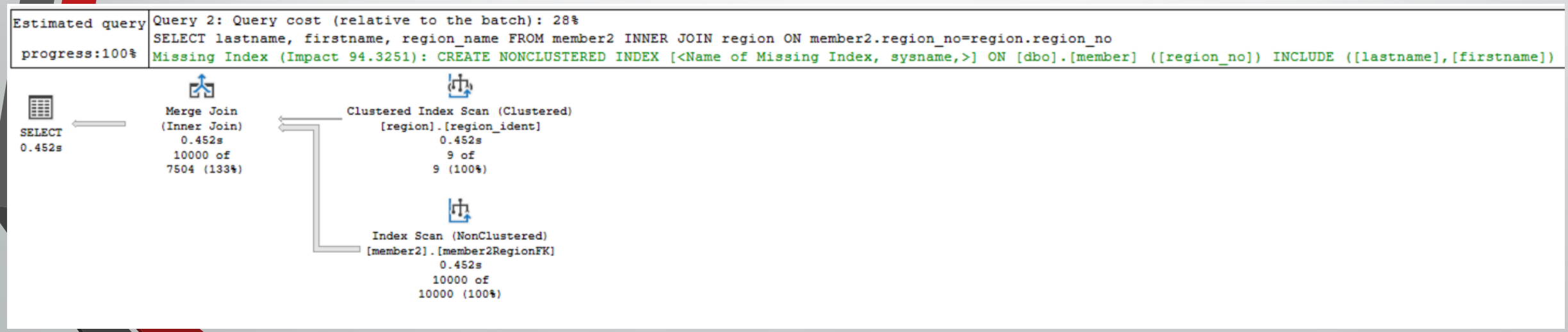
Izvršavanje upita bez odgovarajuceg indeksa

```
SELECT lastname, firstname, region_name
FROM member  INNER JOIN region ON
member.region_no=region.region_no
```



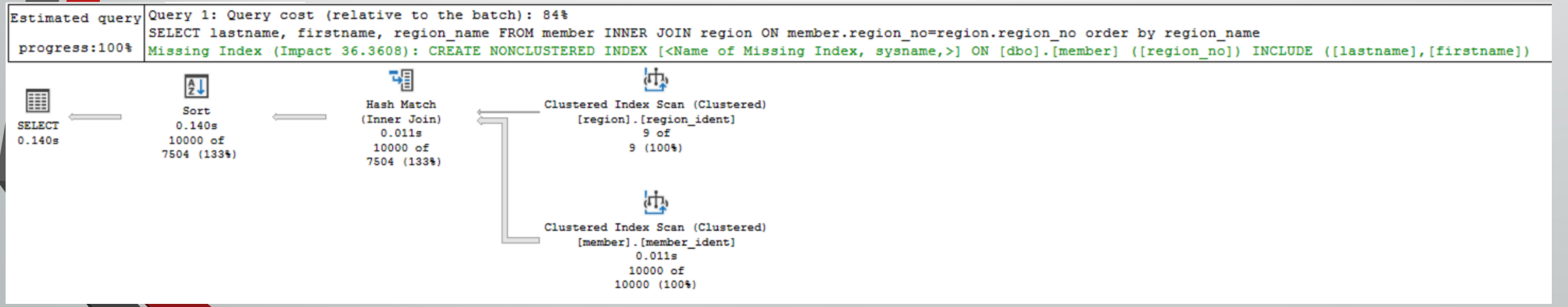
Izvršavanje upita kad indeks postoji

```
SELECT lastname, firstname, region_name
FROM member2  INNER JOIN region ON
member2.region_no=region.region_no
```



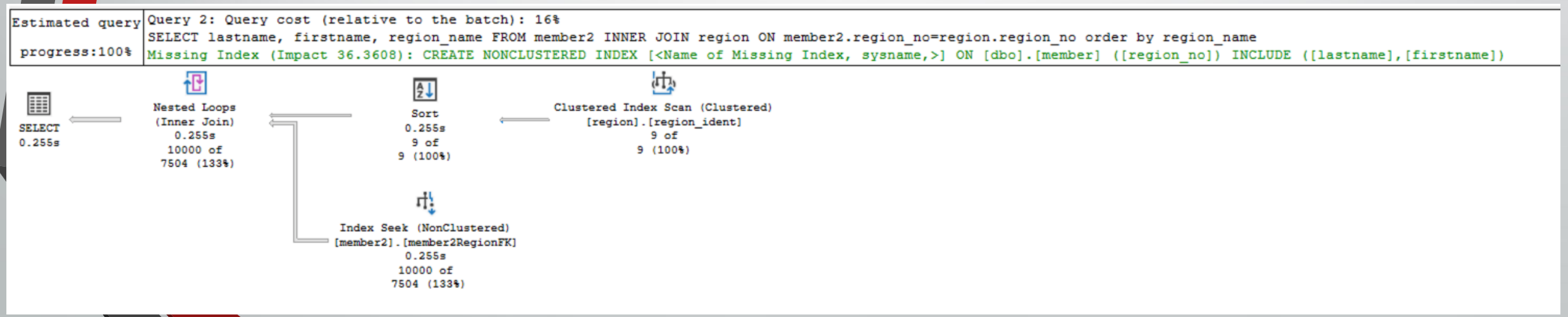
Izvršavanje upita bez odgovarajuceg indeksa

```
SELECT lastname, firstname, region_name
FROM member  INNER JOIN region ON
member.region_no=region.region_no
order by region_name
```



Izvršavanje upita kad indeks postoji

```
SELECT lastname, firstname, region_name
FROM member2  INNER JOIN region ON
member2.region_no=region.region_no
order by region_name
```

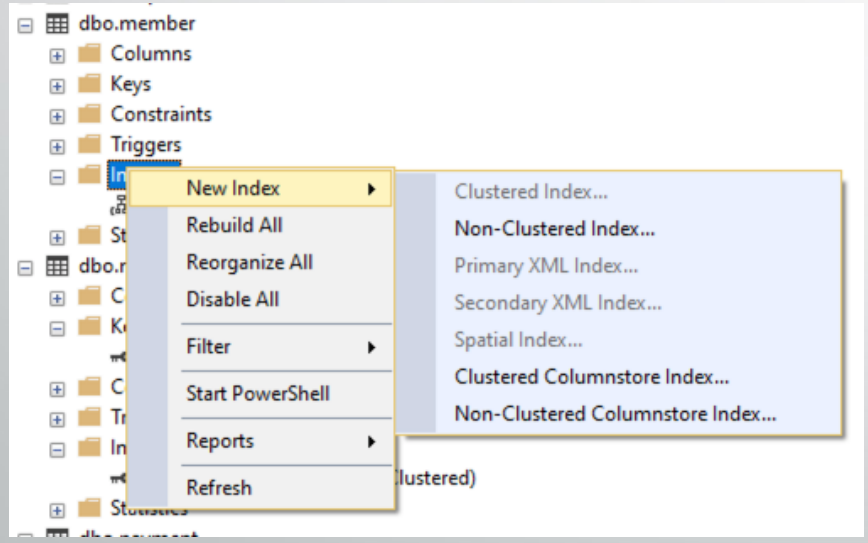


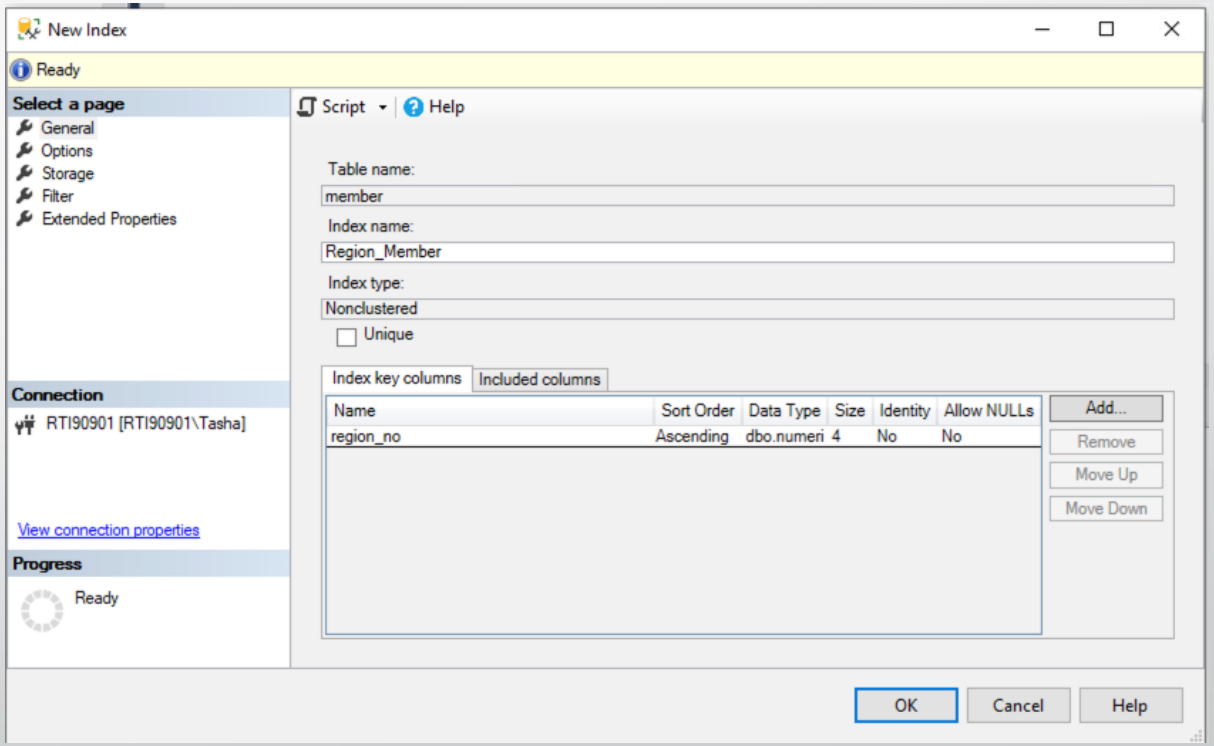
Kreiranje indeksa

```
CREATE UNIQUE CLUSTERED INDEX ID_Member
ON Member (member_no)

CREATE INDEX Region_Member
ON Member (region_no)
```

Kreiranje indeksa

```
CREATE INDEX Region_Member
ON Member (region_no)
```





Prikaz informacija i brisanje indeksa

Prikaz svih indeksa nad tabelom:

```
EXECUTE sp_helpindex Member
```

	index_name	index_description	index_keys
1	ID_Member	clustered, unique located on PRIMARY	member_no
2	Region_Member	nonclustered located on PRIMARY	region_no

Brisanje indeksa:

```
DROP INDEX Member.ID_Member
```

Cluster index

- **Clustered index** je analogan telefonskom imeniku. Svi podaci u samoj tabeli su sortirani po kolonama koje taj index obuhvata. U imeniku bi kolone bile prezime i ime osobe.
- Ovi indexi se čuvaju sa ostalim podacima tako da ne zahtevaju dodatni prostor.
- Za jednu tabelu može postojati najviše jedan clustered index!
- Ukoliko pri kreiranju tabele postavimo neke od atributa kao primarne ključeve alat će u pozadini napraviti clustered index sa tim atributima.

- Primer kreiranja indeksa koji sortira podatke po prezimenu opadajuće i imenu rastuće:

```
CREATE CLUSTERED INDEX Region_Member
ON Member (lastname DESC, firstname)
```

Nonclustered index

- **Nonclustered index** za raliku od **clustered index-**a se čuva odvojeno od samih podataka, kao posebna struktura
- Iz tog razloga je potreban dodatan memorijski prostor za smeštanje. Fizička baza se nije izmenila nakon kreiranja nonclustered index-a.
- Jedna baza može imati više nonclustered index-a!

- Primer kreiranja indeksa nad kolonom region_no koja je strani ključ:

```
CREATE NONCLUSTERED INDEX member_region
ON member ( region_no )
```

Clustered VS Nonclustered

1. Samo jedan Clustered i više Nonclustered index-a po jednoj tabeli
2. Clustered index je brži, jer Nonclustered index ima jedno referenciranje više po svakom redu zbog postojanja pokazivača.
3. Clustered zauzima manje prostora jer se podaci u fizičkoj tabeli sortitraju pri postojanju ovog indexa. Nonclustered se čuva odvojeno u posebnoj strukturi pa samim tim zahteva više prostora. Svaki nonclustered index se čuva u zasebnoj strukturi u odvojenom prostoru.

Unique index

- Unique i Nonunique su osobine indeksa. Clustered index i Nonclustered index mogu imati jednu ili drugu osobinu.
- Unique ukazuje da kombinacija kolona koje dati index obuhvata mora biti jedinstvena.

- **Primary key** ogranicenje po default-u pravi **Clustered Unique Index**.
- Primer:

```
CREATE UNIQUE CLUSTERED INDEX Id_Member
ON Member (member_no)
```

Unique ograničenje

- **Unique** ogranicenje po default-u pravi **NonClustered Unique Index**.
- Primer:

```
ALTER TABLE Member
ADD CONSTRAINT UQ_Member_Name
UNIQUE (firstname)
```

- **Unique** ogranicenje može u pozadini kreirati i **Clustered Unique Index**.
- Primer:

```
ALTER TABLE Member
ADD CONSTRAINT UQ_Member_Name
UNIQUE CLUSTERED (firstname)
```

Filtered index

- **Filtered index** je noncluster indeks kod koga se u strukturi noncluster index-a ne čuvaju svi redovi iz postojeće tabele, već samo oni koji zadovoljavaju neki uslov.

- Primer kreiranja indeksa nad kolonom corp_no samo u slučaju da je ona definisana:

```
CREATE NONCLUSTERED INDEX member_corp
ON member ( corp_no )
WHERE corp_no IS NOT NULL
```

Index with included columns

- Index with included columns je nonclustered indeks U toj strukturi noncluster indexa moguće je čuvati podatke iz još nekih kolona. Time se gubi jedan dodatan korak redirekcije.

- Primer kreiranja indeksa nad kolonom region_no koja je strani ključ u kojoj se čuvaju podaci i o imenu i prezimenu:

```
CREATE NONCLUSTERED INDEX member_region
ON member ( region_no ASC )
INCLUDE ( lastname, firstname)
```

Indeksi

- Prednosti:
  - SELECT iskazi se generalno brže izvršavaju
- Mane:
  - Dodatni prostor za NonClustered indexe
  - INSERT i DELETE upiti se generalno sporije izvršavaju kod Clustered indexa jer je potrebno ispomerati redove radi unosa i brisanja novog reda. Takođe potrebno je izmeniti i stablo kod Clustered i Nonclustered indexa.
- **Covering query** su upiti koji u SELECT naredbi zahteva samo kolone za koje je index definisan (nema referenciranja na fizičku tabelu)

Link:

https://learn.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-ver16

Index view

- Omogućavaju materijalizovanje pogleda.
- Za indeks view treba zadovoljiti sledeće uslove:
  - Pogled treba biti napravljen sa SchemaBinding opcijom (nije dozvoljeno brisanje tabela koje zavise od tog pogleda sve dok se sam pogled ne izbriše, i nije dozvoljeno izvršavanje ALTER TABLE iskaza nad tabelama koje učestvuju u pogledu ukoliko takav iskaz remeti definiciju samog pogleda)
  - Ukoliko agregatna funkcija, koja se nalazi u SELECT listi pogleda, ima mogućnost vraćanja NULL vrednosti potrebno je onemogućiti tako nešto uvođenjem vrednosti koja će zameniti NULL
  - Ukoliko postoji GROUP BY, SELECT lista mora sadržati COUNT_BIG() iskaz
  - Tabele korišćene u CREATE VIEW iskazu moraju biti referisane imenom iz dva dela: ImeŠeme.ImeTabele
- Ukoliko definicija pogleda sadrži GROUP BY klauzulu, tada Unique Clustered Index za taj pogled može da referiše samo kolonu specificiranu u GROUP BY klauzuli.

Index view - Primer

```
CREATE VIEW member_payment_stat WITH SCHEMABINDING
AS
SELECT  M.member_no, firstname, lastname,
    SUM(ISNULL(P.payment_amt,0)) as payment_sum, COUNT_BIG(*) AS payment_num
FROM dbo.member M INNER JOIN  dbo.payment P ON M.member_no=P.member_no
GROUP BY M.member_no, firstname, lastname

SELECT * FROM member_payment_stat
```

- Za sada pogled nije materijalizovan, pa će se iskaz iznad izvršiti u ovom SELECT iskazu

```
CREATE UNIQUE CLUSTERED INDEX UCX_member_payment_stat
ON member_payment_stat(member_no)

SELECT * FROM member_payment_stat WITH (NOEXPAND)
```

- Kada je Clustered index napravljen podaci se dobijaju iz kreirane index strukture pa se upit brže izvršava.