|  |  |  |
| --- | --- | --- |
| Unigrb | Univerzitet u Nišu ELEKTRONSKI FAKULTET | logo_1960_4 |

**Replikacija podataka kod MS SQL baze podataka (replication and mirroring)**

SEMINARSKI RAD

|  |  |
| --- | --- |
| Mentor: | Student: |
|  |  |
| Prof. dr Aleksandar Stanimirović | Stefan Mladenović1694 |

Niš, jun 2024. God.

[1. Uvod 2](#_Toc958517176)

[1.1. Replikacija u SQL Serveru 2](#_Toc387963799)

[2. Transactional replication 3](#_Toc609258695)

[2.1. Arhitektura 4](#_Toc14664688)

[2.2. Konfiguracija 5](#_Toc1900097735)

[2.3. Replication monitor 8](#_Toc884045773)

[2.4. Primer 10](#_Toc2101881989)

[3. Merge replication 12](#_Toc1535374904)

[3.1. Arhitektura 13](#_Toc1831915219)

[3.2. Konfiguracija 14](#_Toc595392437)

[3.3. Primer 16](#_Toc484495917)

[4. Snapshot replication 20](#_Toc177188422)

[4.1. Arhitektura 21](#_Toc511509023)

[4.2. Konfiguracija 22](#_Toc1664809767)

[4.3. Primer 23](#_Toc2024562532)

[5. Peer-to-peer replication 26](#_Toc774213775)

[5.1. Arhitektura 26](#_Toc538052103)

[5.2. Konfiguracija 27](#_Toc1838288415)

[5.3. Primer 29](#_Toc926825639)

[6. Database mirroring 32](#_Toc1494587000)

[6.1. Načini rada 33](#_Toc705071761)

[7. Zaključak 35](#_Toc1344631668)

[8. Literatura 35](#_Toc293303421)

# Uvod

## 1.1. Replikacija u SQL Serveru

Replikacija u SQL Serveru je skup tehnologija za kopiranje i distribuciju podataka i objekata baze podataka iz jedne baze u drugu, kao i za sinhronizaciju između baza kako bi se održala doslednost. Replikacija se koristi za poboljšanje dostupnosti i skalabilnosti podataka omogućavajući višestrukim serverskim bazama podataka da dele opterećenje čitanja i pisanja operacija. U SQL Serveru replikacija može biti konfigurisana da radi neprekidno, pružajući sinhronizaciju baza podataka u realnom vremenu, ili može biti zakazana da se pokreće u određenim intervalima, u zavisnosti od samih zahteva. Na ovaj način se omogućava efikasna sinhronizacija podataka.

Proces replikacije se sastoji od sledećih komponenti:

1. ***Publisher***: Izvorni serverski sistem baze podataka koji omogućava dostupnost podataka za replikaciju. *Publisher* definiše koji podaci se repliciraju putem publikacije.
2. ***Subscriber***: Odredišni serverski sistem baze podataka koji prima replicirane podatke. *Subscriber-i* se pretplaćuju na publikacije i primaju promene podataka od Publisher-a.
3. ***Distributor***: Posrednik između *Publisher-a* i *Subscriber-a*. *Distributor* skladišti metapodatke, istorijske podatke i transakcije za sve vrste replikacija. Distribuciona baza podataka postoji na *Distributor-u*.
4. ***Publication***: Kolekcija *Article-a* iz baze podataka koji se repliciraju. Publikacija može uključivati tabele, poglede, indeksirane poglede i skladištene procedure.
5. ***Article***: Osnovna jedinica replikacije. *Article* može biti tabela, deo tabele ili objekat baze podataka poput skladištene procedure ili pogleda.
6. ***Subscription***: Zahtev *Subscriber-a* za primanje publikacije. Postoje dve vrste pretplata: *push* pretplate (pokreće ih *Publisher*) i *pull* pretplate (pokreće ih *Subscriber*).

SQL Server nudi nekoliko režima replikacije, svaki dizajniran da zadovolji specifične potrebe za kopiranjem i sinhronizacijom podataka među bazama podataka. Postoje 6 režima u SQL Serveru i to su:

1. Transactional replication
2. Merge replication
3. Snapshot replication
4. Peer-to-peer replication
5. Bidirectional replication
6. Updatable Subscriptions replication

Svaki od ovih režima ima svoje specifične karakteristike, prednosti i nedostatke, što omogućava izbor odgovarajuće opcije prema potrebama samih zahteva problema. Korišćenjem različitih režima replikacije moguće je rešiti određene probleme, bilo da je reč o distribuciji podataka na više geografskih lokacija, balansiranju opterećenja ili obezbeđivanju visoke dostupnosti. Iako replikacija donosi mnoge prednosti, važno je pažljivo planirati i implementirati replikacioni sistem kako bi se izbegli potencijalni problemi i osigurale optimalne performanse.

# Transactional replication

Transakciona replikacija funkcioniše tako što prvo napravi snimak trenutnog stanja baze podataka koju želimo da repliciramo. Nakon što se taj početni snimak završi, sve buduće promene podataka i izmene strukture baze podataka koje se dogode na izvornoj bazi se prenose na odredišnu bazu u realnom vremenu. Te promene se primenjuju na odredišnoj bazi *(Subscriber)* istim redosledom i unutar istih transakcionih okvira kao na izvornoj bazi, i na taj način se osigurava konzistentnost podataka.[1]

Ovaj režim replikacije je posebno koristan u scenarijama gde su konzistentnost podataka i niska latencija ključni. Ova metoda replikacije se najviše koristi u sistemima sa visokom dostupnošću i distribuiranim bazama podataka. Osim toga, ovaj režim se može primeniti i kada odredišna strana nije SQL Server baza već neka druga.

Inače, odredišne baze u transakcionoj replikaciji su obično samo za čitanje, jer se promene ne vraćaju nazad na izvornu bazu. Ipak, postoje opcije koje omogućavaju ažuriranje podataka i na odredišnoj bazi.

## 2.1. Arhitektura

Transakciona replikacija koristi tri glavna agenta:

1. ***Snapshot agent***

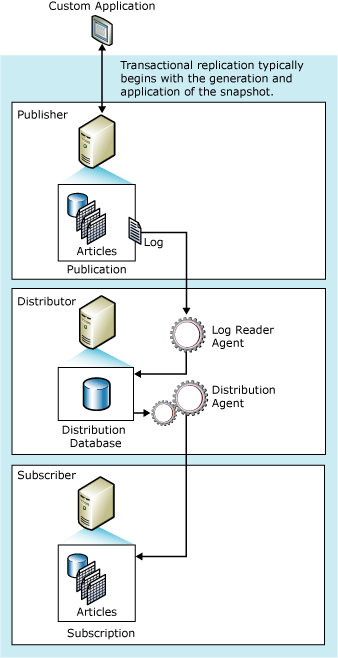
Ovaj agent je odgovoran za kreiranje snimaka (*snapshotova)* baze podataka. Ovi snimci uključuju šemu i podatke objavljenih tabela i baza podataka. *Snapshot* agent čuva ove snimke u određenom direktorijumu i beleži zadatke sinhronizacije u distribucionu bazu podataka na serveru za distribuciju.

1. ***Log Reader agent***

*Log Reader* agent nadgleda transakcioni log svake baze podataka koja je podešena za transakcijsku replikaciju. On kopira transakcije označene za replikaciju iz transakcionog loga u distribucionu bazu podataka, koja služi kao privremeno skladište za ove promene. Ovaj agent koristi internu proceduru za pribavljanje narednih transakcija za replikaciju, osiguravajući da samo potvrđene transakcije budu prenešene.

***3. Distribution agent***

Distribution agent prebacuje inicijalne snimke i transakcije iz distribucione baze podataka ka pretplatnicima. Ovaj agent može raditi neprekidno, što omogućava minimalno kašnjenje u prenosu podataka, ili u određenim intervalima. Distribution agent radi na distribucionom serveru za za *push subscriptions* ili na serveru pretplatnika za *pull subscriptions*.



Slika 2.1. Arhitektura transakcione replikacije. [2]

Pre nego što novi pretplatnik može primiti inkrementalne promene, mora imati tabele sa istom šemom i podacima kao i tabele na izvornoj bazi podataka. Ovaj početni skup podataka obično kreira *Snapshot* agent i distribuira *Distribution* agent.

Kada *Log Reader* agent prenese transakcije u distribucionu bazu podataka, *Distribution* agent ih prebacuje ka pretplatnicima. Transakcije će biti sačuvane u distribucionoj bazi sve dok ne budu poslate svim pretplatnicima ili dok ne istekne definisani period čuvanja. Nakon toga transakcije se distribuiraju u redosledu kako su izvršene na izvornoj bazi.

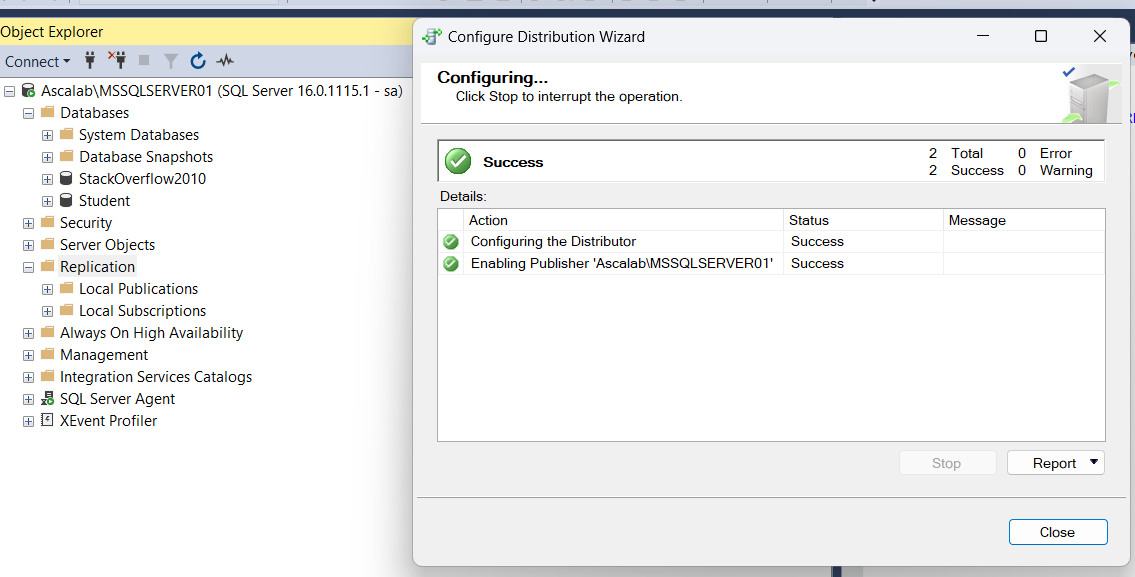
## 2.2. Konfiguracija

Proces konfiugracije transakcione replikacije u SQL Server Managment Studio uključuje nekoliko koraka. Za transakcionu replikaciju potrbno je da imate najmanje dve zasebne istance SQL Server-a, kako bi mogli da uspostavite replikaciju. U slučaju da imate dve instance, tada će jedna instanca predstavljati i distribucioni server i *publisher-a,* dok će druga instanca predstavljati *subscriber-a.* Naredni koraci opisuju kako se konfigurišu distribucioni server, *publisher* i *subscriber:*

1. Konfiguracija distribucionog servera:

Distributor je odgovoran za skladištenje snimaka i transakcija dok čekaju da budu prenešeni na pretplatnike. Konfiguracija distributora se vrši tako što:

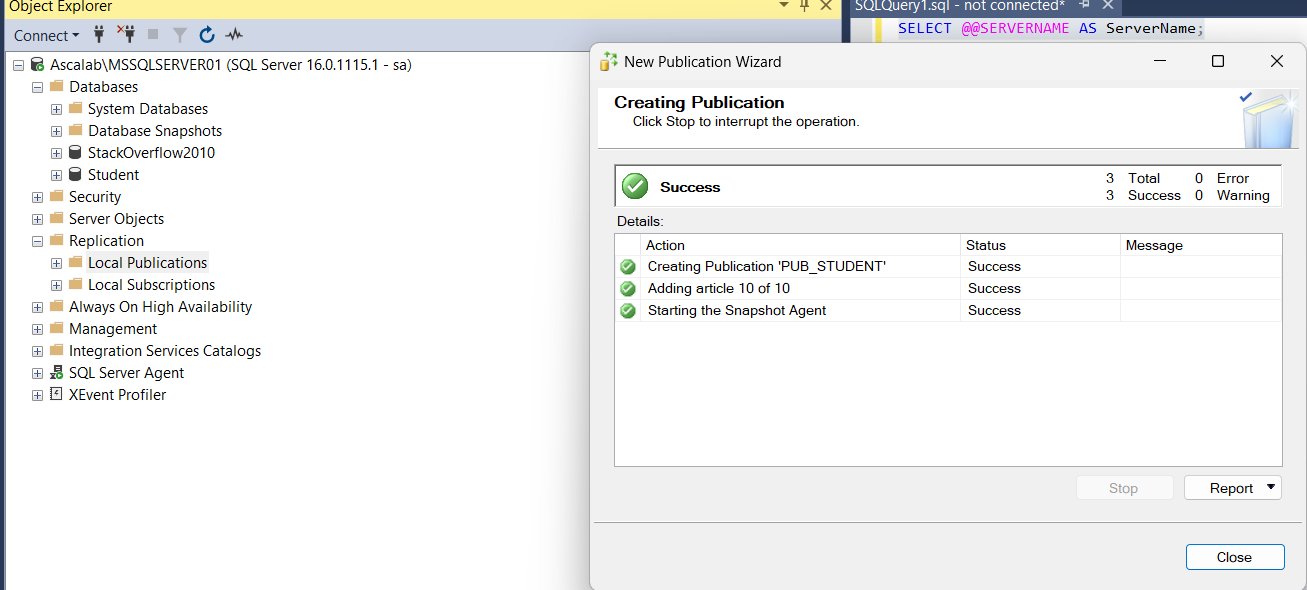
1. Iz SQL Server Management Studio povežemo se na prvu SQL Server instancu koja će biti distributor. Ujedno na ovoj instanci se i nalazi baza koja će biti replicirana.
2. Desnim klikom na *Replication* folder u *Object Explorer-u* otvaramo *Configure Distribution*.
3. Nakon toga pratimo korake u *Distribution Configuration Wizard-u* za izbor servera kao distributora i postavljanje distribucione baze podataka.

Slika 2.2. Konfigurisanje distributora.

1. Konfiguracija *publisher-a*

*Publisher* predstavlja izvornu bazu koja se replicira. Kako bi konfigurisali *publisher-a* moramo kreirati publikaciju. Publikaciju kreiramo na sledeći način:

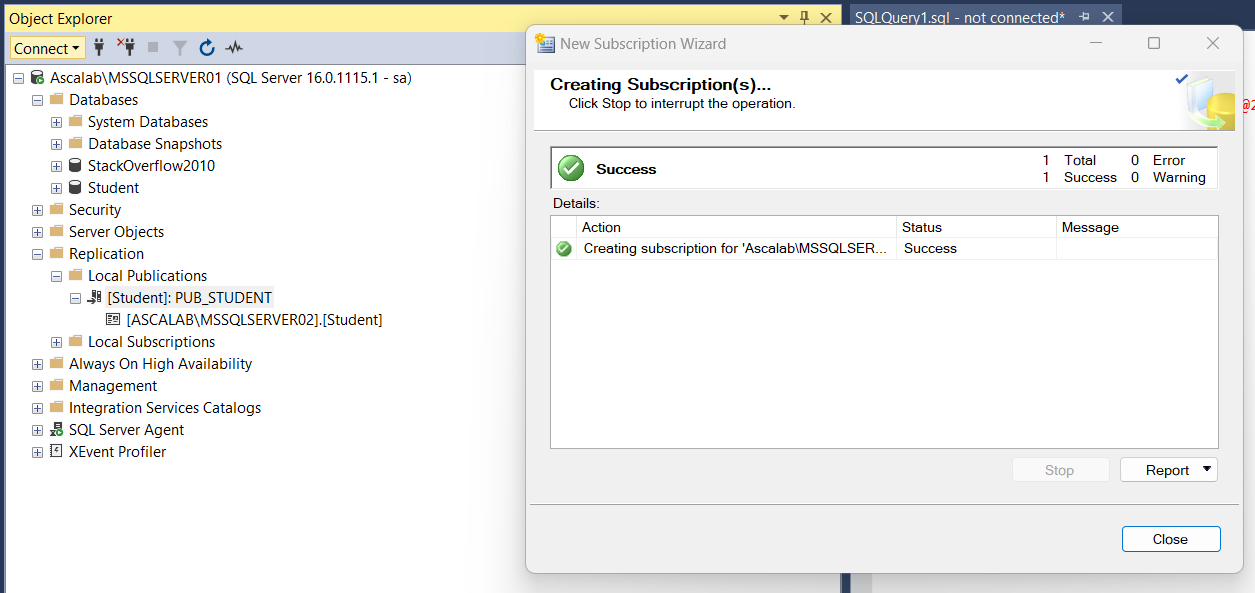
1. Desnim klkom na *Replication,* a zatim na opciju *New Publication.*
2. Nakon toga se otvara *wizard,* gde je potrebno odabrati distribucionu bazu koju smo kreirali u prethodnom koraku.
3. U sledećem koraku biramo bazu za replikaciju, odnosno *Student* bazu podataka.
4. Označimo da je tip replikacije *Transactional Publication.*
5. Posle toga biramo tabele koje će biti uključene u publikaciju (odnosno kreiramo *Articles*).
6. Definišemo sigurnosne konfiguracije za agente koje će prenositi podatke.
7. Na kraju potvrđujemo *wizard* i kreiramo publikaciju.

Slika 2.3. Kreiranje *transactional replication* publikacije.

1. Konfiguracija *subscriber-a*

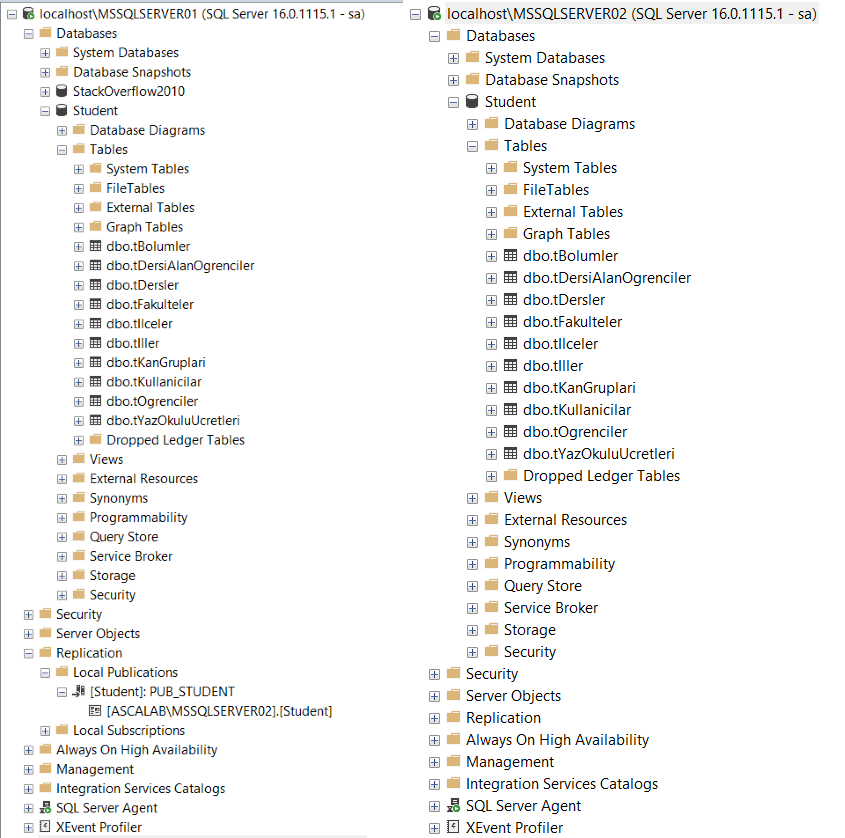
*Subscriber* je baza podataka koja prima i primenjuje promene sa *publisher-a*. Pre same replikacije na drugoj instanci SQL Servera potrebno je kreirati praznu bazu podataka. Nakon toga, konektujemo se na prvu instancu i moramo izvršiti sledeće korake:

1. Desni klik na publikaciju koju smo kreirali (*PUB\_STUDENT*), a zatim *New Subscription.*
2. U sledećem koraku potrebno je odabrati *push* ili *pull subscription.* Kod *push subscribtion* distribucioni agent radi na distribucionom serveru, dok kod *pull subscription* distribucioni agent radi na pretplatničkom serveru. U ovom koraku ćemo nastaviti sa *push subscription.*
3. Nakon toga, definišemo drugu instancu SQL servera za *subscriber-a.* I unosimo kredencijale za distiribucionog agenta.
4. U sledećem koraku možemo izabrati periodičnu ili konstantu distribuciju od strane distribucionog agenta.
5. Na kraju potvrđujemo wizard i kreiramo pretplatu. [3]



Slika 2.4. Kreiranje *transactional replication* pretplate.

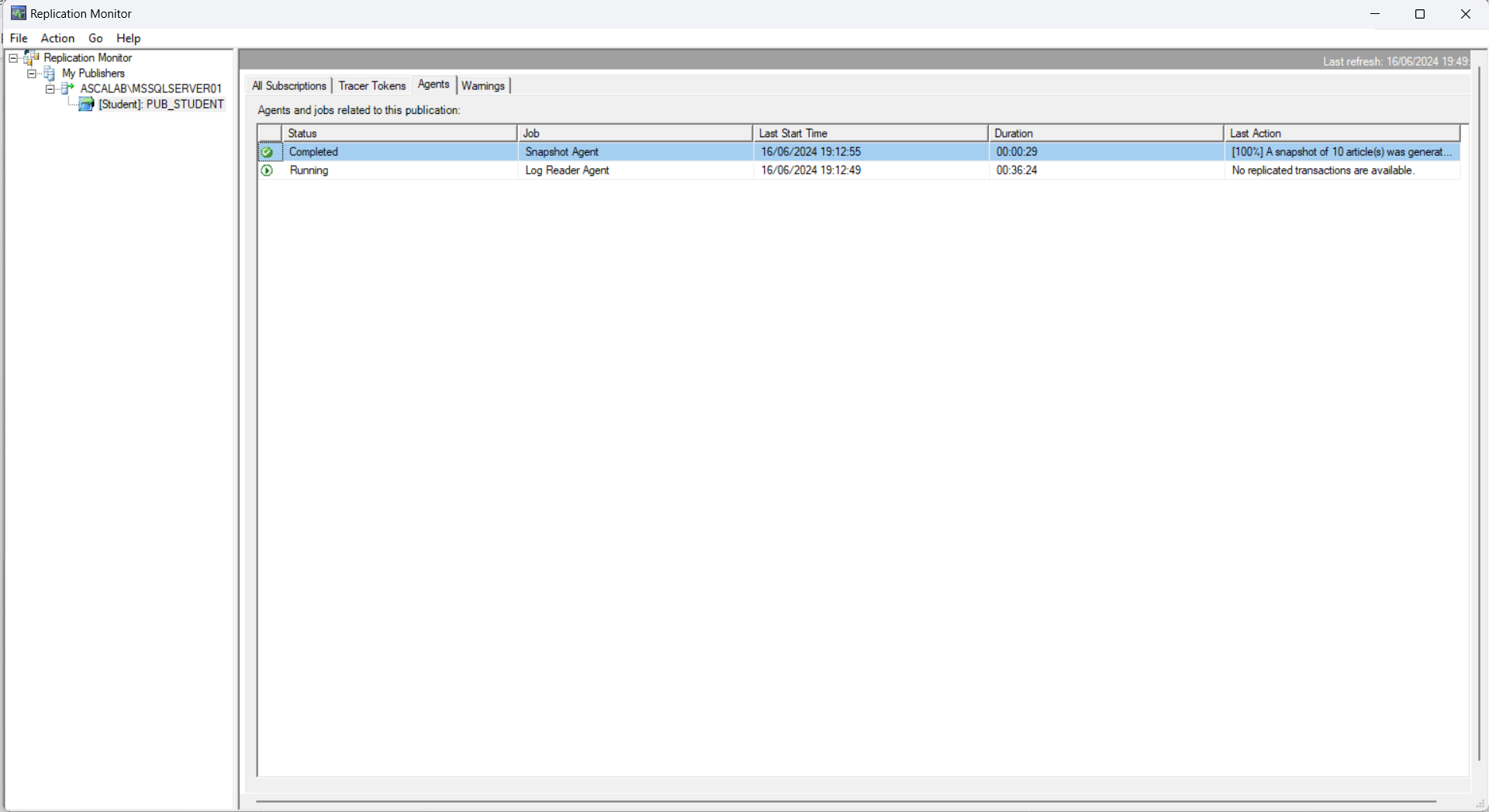
Nakon uspešno kreirane pretplate, tabele i podaci su replicirani na drugu instancu SQL Server baze podataka.



Slika 2.5. Uporedni prikaz *publisher* servera i *subscriber* servera nakon replikacije.

## 2.3. *Replication* monitor

Management studio nudi monitoring replikacija. Desnim klikom na kreiranu supskipciju biramo opciju *Lunch Replication Monitor.*

 Slika 2.6. *Replication monitor.*

Replikacioni monitor u Microsoft SQL Server-u omogućava pregled i upravljanje replikacijom podataka. On pruža informacije o aktivnostima replikacije, omogućava praćenje stanja sistema replikacije, kao i izvođenje različitih administrativnih upravljanja nad postjećim replikacijama.

**Mogućnosti replikacionog monitora:**

1. Upravljanje agentima za replikaciju:
   * Omogućava pokretanje i zaustavljanje agenata.
   * Pruža detaljne informacije o statusu agenata i njihovim greškama.
2. Praćenje performansi supskripcija:

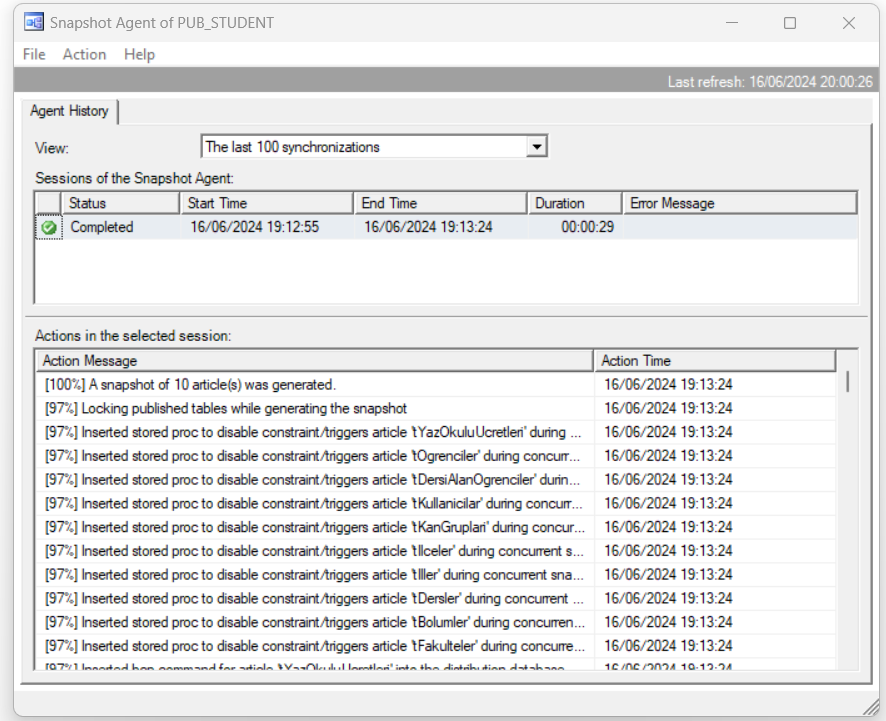
* Prikazuje informacije o performansama supskripcija.
* Omogućava filtriranje supskripcija sa greškama ili slabim performansama.

1. Upravljanje publikacijama i supskripcija:

* Omogućava ponovnu inicijalizaciju i validaciju supskripcija.
* Pruža opcije za upravljanje svojstvima publikacija i supskripcija.

1. Merenje latencije (samo za transakcionu replikaciju)

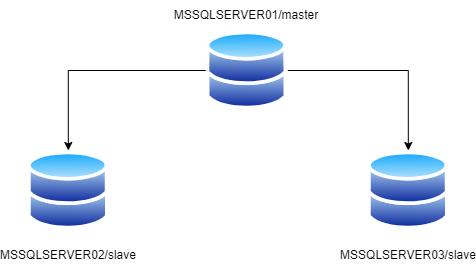
* *Tracer Tokens* tab omogućava merenje vremena potrebnog da transakcija sa *publisher-a* stigne do *subscriber-a*. [4]



Slika 2.7. Prikaz istorije agenta kroz replikacioni monitor.

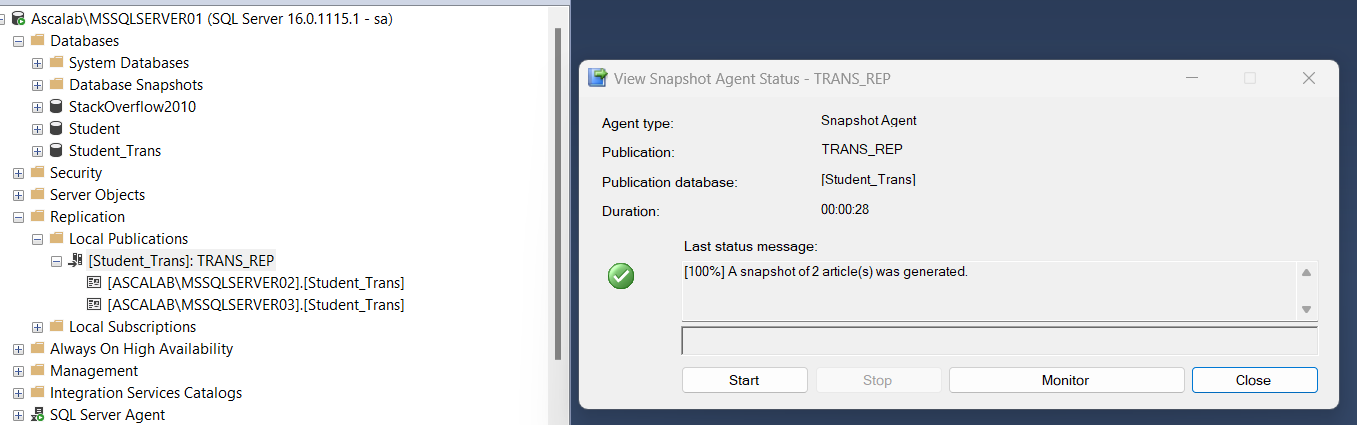
## 2.4. Primer

Na lokalnom okruženju podignute su tri instance SQL Servera *MSSQLSERVER01*, *MSSQLSERVER02* i *MSSQLSERVER03*. Na prvoj instanci nalazi se baza podataka *Student\_Trans* koja će biti replicirana na ostale servere. Ostale dve instance imaju samo kreirane baze podataka *Student\_Trans*.

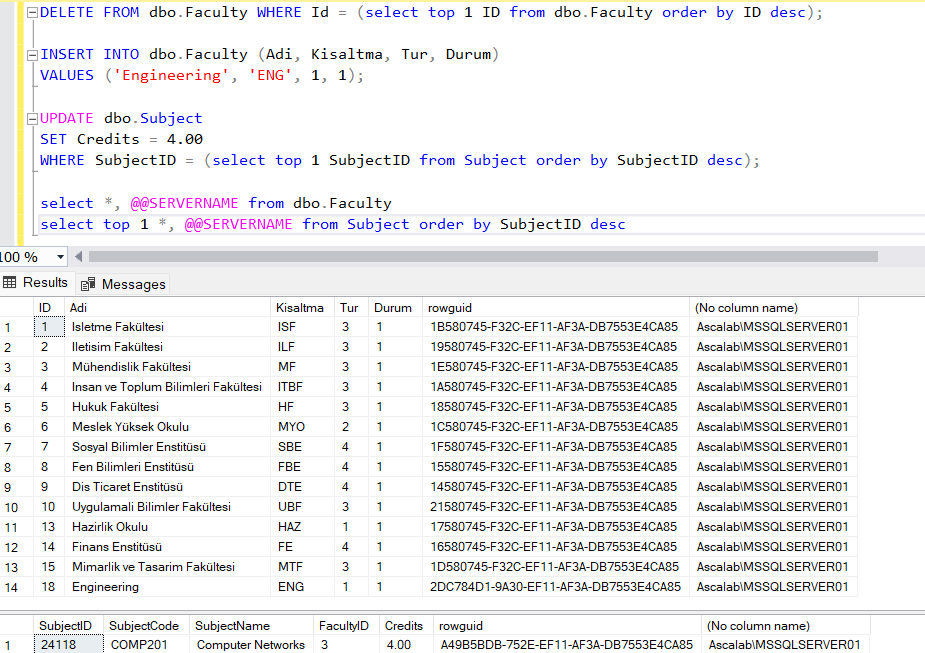


Slika 2.8. Prikaz arhitekture replikacije.

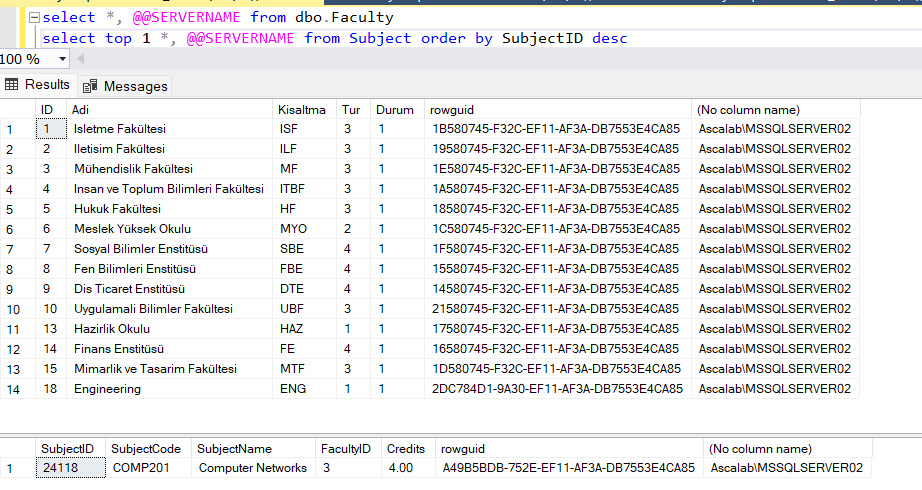
Nakon kreiranja transakcione replikacije opisane u prethodnom poglavlju dodali smo još jednog pretplatnika *MSSQLSERVER03* na isti način kao za *MSSQLSERVER02*.Snapshoot izvorne baze je repliciran na ostale pretplatničke baze podataka *Student\_Trans*, tako da *Student\_Trans* baze podataka na slave serverima poseduju i šeme i podatke. U ovom slučaju su kreirana samo dva članka, za tabelu *Faculty* i tabelu *Subject.*

Slika 2.9. Kreiranje pretplatnika.

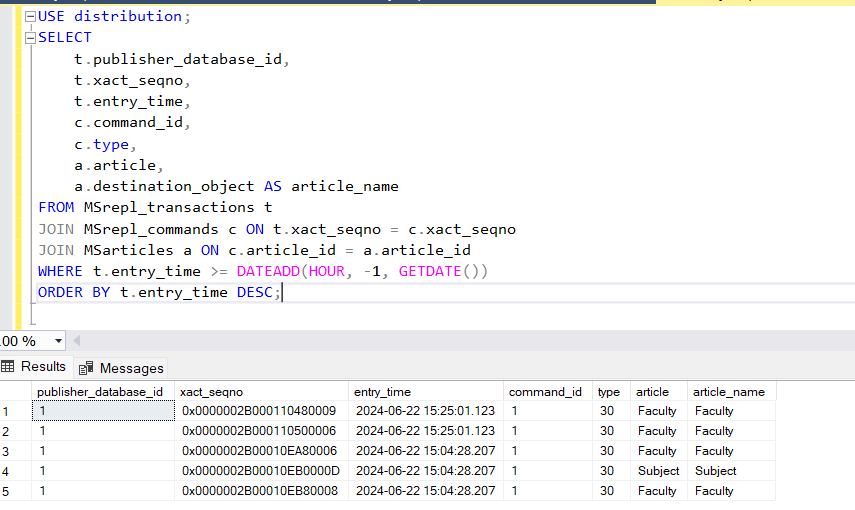
Pošto smo već u prethodnom delu rada naveli da transakciona replikacija se prenosi samo u jednom smeru, odnosno pretplatničke baze podataka se mogu posmatrati kao *read-only* baze. Tako da replikaciju možemo testirati izvršavanjem promena na master server bazi i pratiti kako se to odražava na slave server bazama.

Slika 2.10. Izvršavanje upita na *master* bazi.

Na master bazi smo izvršili promene brisanjem i dodavanjem jednog zapisa u tabelu *Faculty,* kao i ažuriranje jednog reda u *Subject* tabeli. Ako izvršimo iste upite za prikaz iz tabela *Faculty* i *Subject* na nekoj od *slave* baza, videćemo da su se iste transakcije replicirale sa *master* baze replicirale i na *slave* bazama*.*

Slika 2.11. Izvršavanje upita na *slave* bazi.

Korišćenjem sistemskih tabela za replikaciju možemo prikazati istoriju transakcionih replikacija.



Slika 2.11. Prikaz istorije replikacionih transakcija.

Ovaj upit nam prikazuje kada su se desili transakcije i na koje objekte u bazi su uticale transakcije. Ove sistemske tabele nam omogućavaju da lakše identifikujemo kada su se određene transakcije replicirale.

Ovim primerima smo demonstrirali kako se odvija transakciona replikacija između master i slave baza i kako se mogu pratiti transakcije koje se repliciraju.

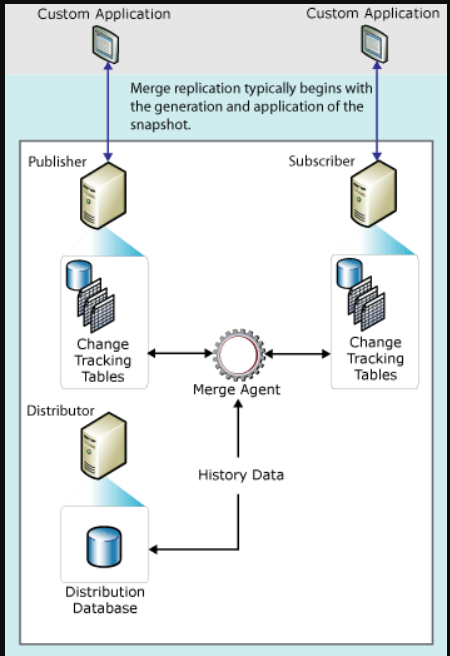
# Merge replication

*Merge* replikacija predstavlja još jednu strategiju sinhronizacije podataka koja se koristi u SQL Server-u kako bi omogućila da više baza podataka održi konzistentnost i integritet na različitim serverima. Ovaj model replikacije je posebno koristan u situacijama gde dve ili više baza podataka treba da vrše izmene nezavisno, a te promene treba da se odraze na svim čvorovima u mreži.

Često se primenjuje u okruženjima gde serveri komuniciraju sa klijentima. Ovaj tip replikacije je pogodan u sledećim situacijama:

* Različiti pretplatnicimogu nezavisno vršiti izmene nad istim podacima u različito vreme, pa će se te izmene kasnije preneti izdavaču i drugim pretplatnicima.
* Pretplatnici takođe mogu da preuzmu podatke, izvrše izmene dok su *offline*, i kasnije te izmene sinhronizuju sa izdavačima i drugim pretplatnicima.
* Svaki pretplatnik može zahtevati specifičan segment podataka koji mu je potreban.
* U slučaju da dođe do konflikata, potrebno je njihovo razrešenje.
* Aplikacija zahteva evidenciju o konačnim promenama podataka, a ne o svakoj pojedinačnoj izmeni. Na primer, ako pretplatnik izmeni određeni red više puta pre sinhronizacije sa izdavačem, izdavač će registrovati samo poslednju promenu.

## 3.1. Arhitektura



Slika 3.1. Arhitektura merge replikacionog modela. [5]

U odnosu na prethodni model replikacije, *merge* replikacija koristi i *Merge Agent* koji je zadužen za sinhronizaciju. Onkoristi jedinstvenu kolonu *rowguid* dodatu svakom članku (*Articles*) za praćenje promena. To omogućava agentu da identifikuje koje redove su dodati, ažurirani ili obrisani na izdavaču i svakom pretplatniku.

Agent se povezuje sa bazama podataka pretplatnika i izdavača, nakon čega upoređuje promene napravljene na izdavaču i svakom pretplatniku te određuje odgovarajuće akcije za sinhronizaciju podataka. Agent zatim primenjuje potrebne operacije dodavanja, ažuriranja i brisanja kako bi uskladio podatke pretplatnika sa izdavačem. U slučaju da postoje konflikti između, odnosno desile su se promene određenih redova na pretplatničkoj strani u odnosu na izdavačku bazu podataka, tada *Merge Agent* koristi unapred definisanu politiku razrešavanja konflikta da odredi koje izmene treba prihvatiti. Podrazumevana politika jeste da se prihvate promene pretplatnika koji ima veći prioritet, ali mogu se definisati skripte za složenije situacije ili manuelno rešavanje konflikata.

Pošto *merge* replikacija predstavlja bidirekcionu replikaciju, najbolje je vršiti sinhronizaciju između preplatnika i izdavača u što kraćim intervalima, kako bi se smanjila verovatnoća pojave konflikata. Osim sinhronizacije, moguće je definisati *trigger-e* nad člancima koji će javiti da je došlo do promene i nakon toga pokrenuti sinhronizaciju između pretplatnika i izdavača.

## 3.2. Konfiguracija

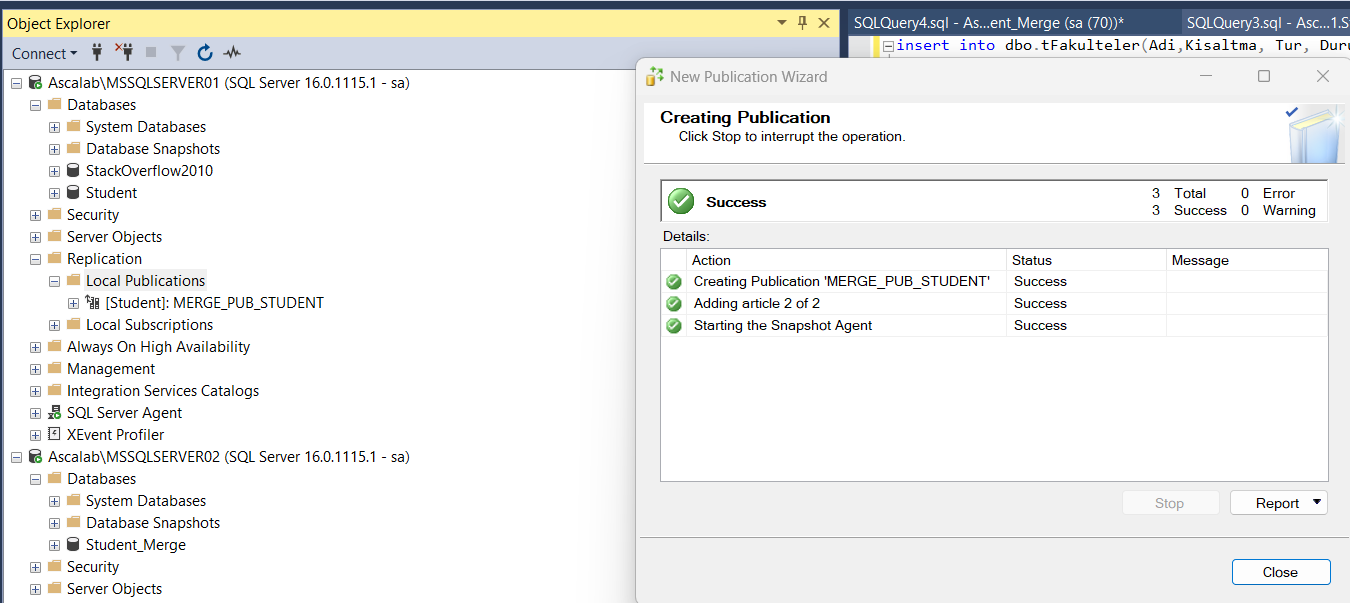
Kao i kod prethodnog replikacionog modela potrebno je minimum dve instance SQL servera kako bi se uspostavila replikacija. Konfiguracija se vrši iz nekoliko koraka:

1. Konfiguracija distribucionog servera:

Kao i kod prethodne replikacije distributor se konfiguriše tako što:

* 1. Iz SQL Server Management Studio povežemo se na prvu SQL Server instancu koja će biti distributor. Ujedno na ovoj instanci se i nalazi baza koja će biti replicirana.
  2. Desnim klikom na *Replication* folder u *Object Explorer-u* otvaramo *Configure Distribution*.
  3. Nakon toga pratimo korake u *Distribution Configuration Wizard-u* za izbor servera kao distributora i postavljanje distribucione baze podataka.

1. Kreiranje publikacije
   1. Iz *Object Explorer-a* potrebno je ići desnim klikom na opciju *New -> Publication.*
   2. Na samom početku kroz *wizard,* potrebno je odabrati bazu koju ćemo replicirati (*Student*) i odabrati tip replikacije *Merge Publication.*
   3. U sledećem koraku biramo “SQL Server 2008 or later” kako bi odabrali najnoviju verziju SQL Servera.
   4. Na *Articles* tabu, vršimo odabir tabela koje ćemo replicirati, u ovom slučaju to će biti tabele *Faculty* i *Subject.*
   5. Na sledećem koraku dobijamo obaveštenje da će nova kolona koja sadrži *uniqueidentifier* biti dodata u okviru naših replikacionih tabela. Ova kolona će omogućiti lakše praćenje promena koje su se dogodile kod pretplatnika.
   6. Na tabu *Filter Rows* mogu se dodati filteri kako bi se vršila replikacija određenih podataka (npr. samo redovi sa identifikatorom većim od 5). Filteri nisu obavezni.
   7. Na sledećem tabu, vršimo podešavanje *Snapshot Agent-a,* kao i kada će se *snapshot* podataka kreirati.
   8. U narednom koraku, vrši sigurnosnu konfiguraciju *Snapshot Agent-a* navođenjem login parametara.
   9. I na kraju unosimo ime publikacije i kreiramo publikaciju.

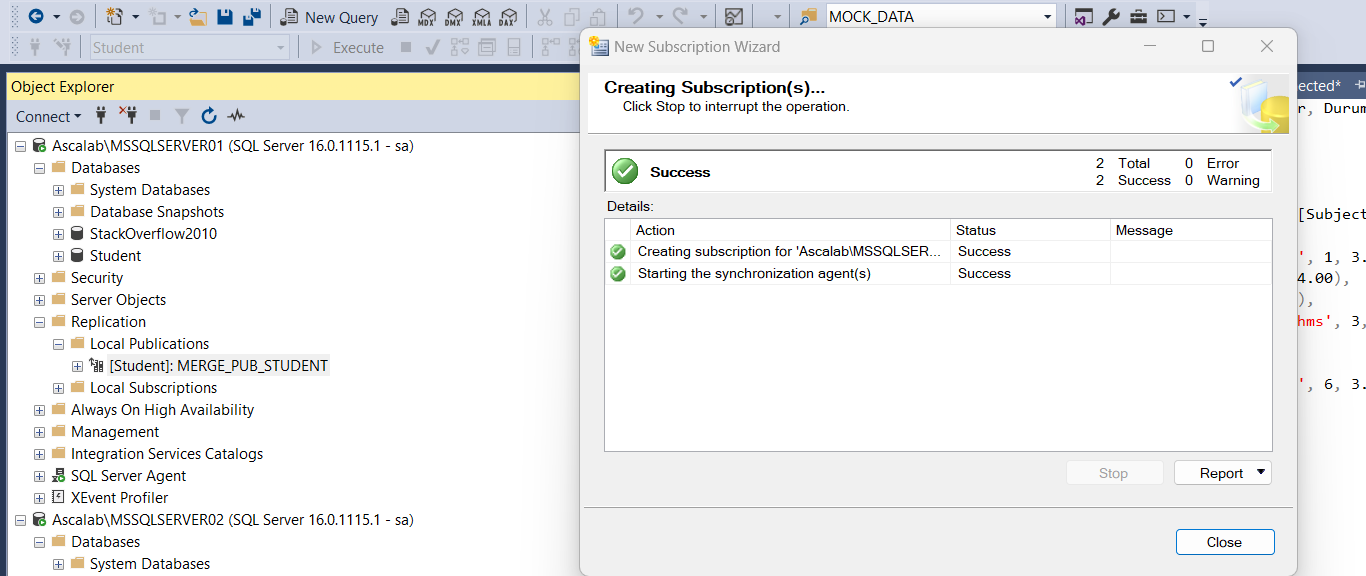


Slika 3.2. Kreiranje *merge* publikacije.

3. Kreiranje *subscriber-a*

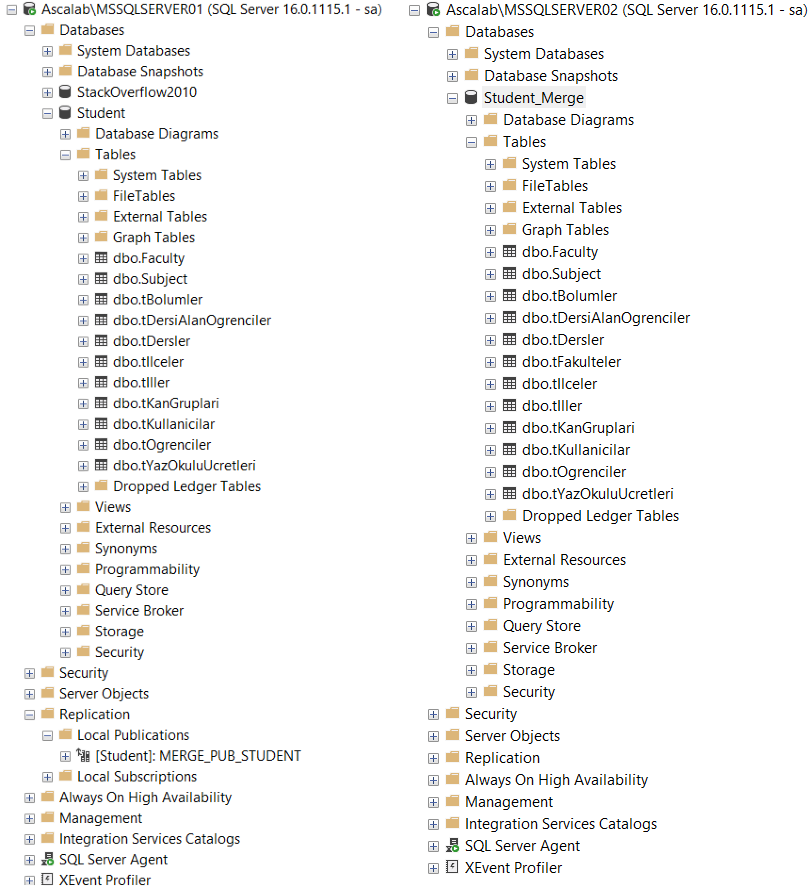
Pre same replikacije na drugoj instanci SQL Servera potrebno je kreirati praznu bazu podataka. Nakon toga, konektujemo se na prvu instancu i moramo izvršiti sledeće korake:

1. Desnim klikom na kreiranu publikaciju biramo opciju *New Subscription,* nakon čega nam se otvara prozor za kreiranje pretplate.
2. Nakon toga, selektujemo trenutni server za *publisher-a.*
3. Na sledećem tabu, biramo *push subscription.*
4. U narednom koraku za *subscriber-a* biramo drugu instancu SQL Servera i novokreiranu bazu podataka.
5. Na sledećem koraku, konfigurišemo sigurnosne postavke za Distribution Agent-a.
6. Na tabu *Synchronization Schedule* konfigurišemo kada će se singhronizacija obavljati, na primer na svakih 10 minuta. Pošto se radi o bidirekcionalnoj replikaciji najbolje je staviti učestaliju sinhronizaciju kako ne bi dolazilo do čestih konflikata.
7. Na narednom tabu, trebamo da konfigurišemo prioritet u slučaju konflikta. Ako želimo da trenutni server ima veći prioritet u odnosu na ostale server trebamo postaviti veću vrednost. Podrazumevana vrednost je 75.
8. Klikom na *Finish* dugme završamo kreiranje pretplate.



Slika 3.3. Kreiranje *merge* pretplate.

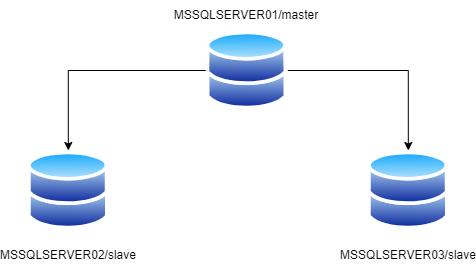
Nakon uspešno kreirane pretplate, tabele i podaci su replicirani na drugu instancu SQL Server baze podataka.



Slika 3.4. Uporedni prikaz *publisher* servera i *subscriber* servera nakon replikacije.

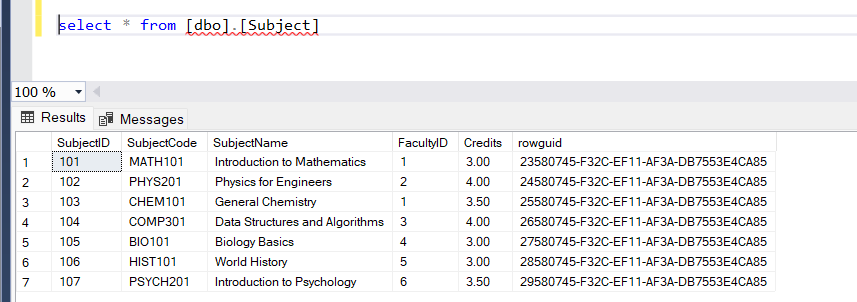
## 3.3. Primer

Na lokalnom okruženju podignute su tri instance SQL Servera *MSSQLSERVER01, MSSQLSERVER02* i *MSSQLSERVER03.* Na prvoj instanci nalazi se baza podataka *Student* koja će biti replicirana na ostale servere.Ostale dve instance imaju samo kreirane baze podataka *Student\_Merge.*

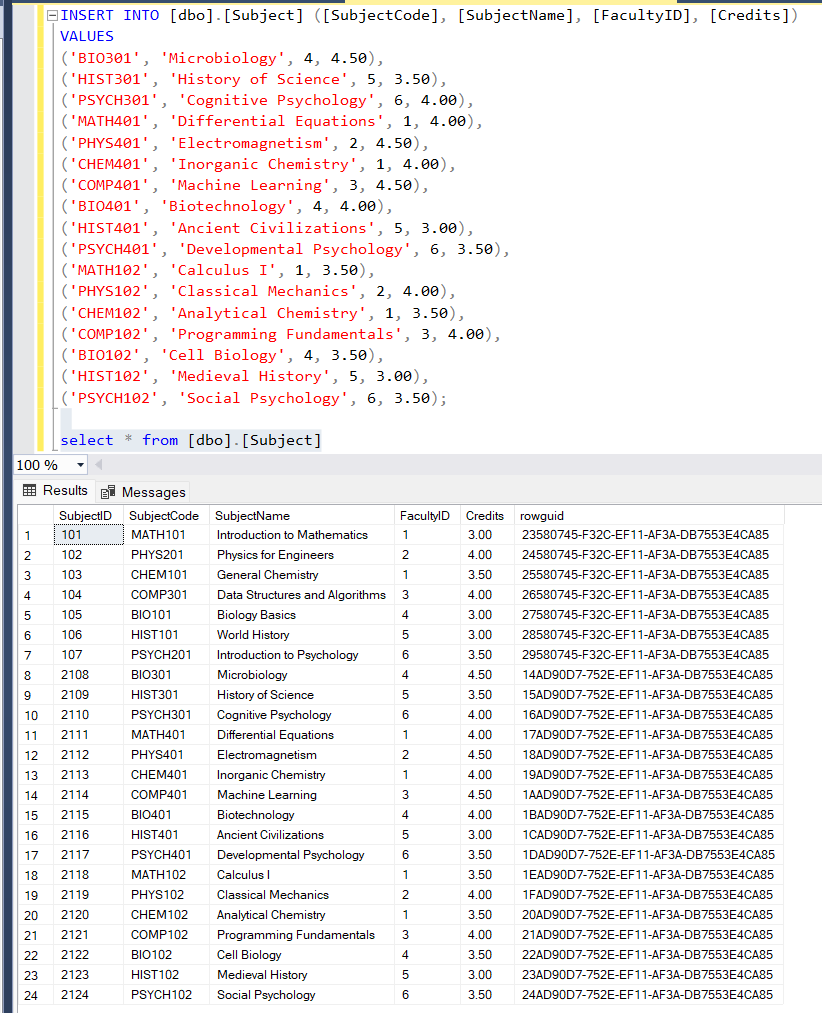


Slika 3.5. Prikaz arhitekture replikacije.

Nakon kreiranja merge replikacije opisane u prethodnom poglavlju dodali smo još jednog pretplatnika *MSSQLSERVER03. Snapshoot* izvorne baze je repliciran na ostale pretplatničke baze podataka *Student\_Merge,* tako da *Student\_Merge* baze podataka na slave serverima poseduju i šeme i podatke.

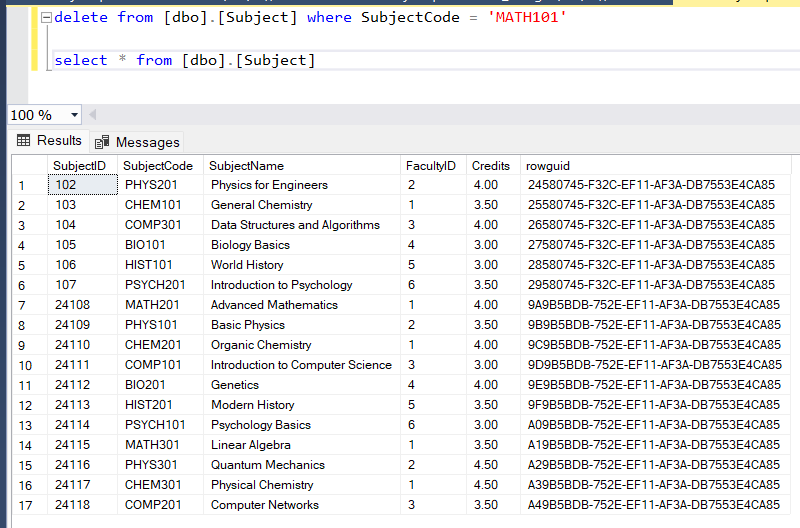
Trenutno stanje tabele *Subject* na master bazi izgleda ovako:  
Slika 3.6. Tabela *Subject* na master bazi.

Kako bi demonstrirali *merge* replikaciju, na drugoj instanci server smo izvršili dodavanje novih redova u tabeli *Subject.*



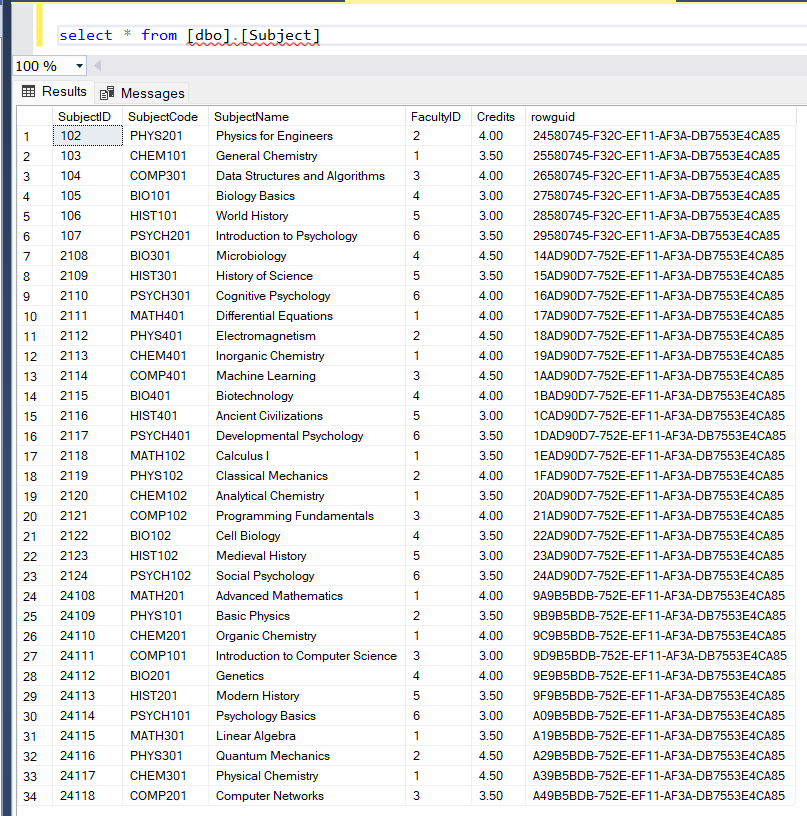
Slika 3.7. Tabela *Subject* na slave bazi *MSSQLSERVER02*.

Dok smo na *MSSQLSERVER03* instanci izvršili brisanje jednog reda iz tabele *Subject.*



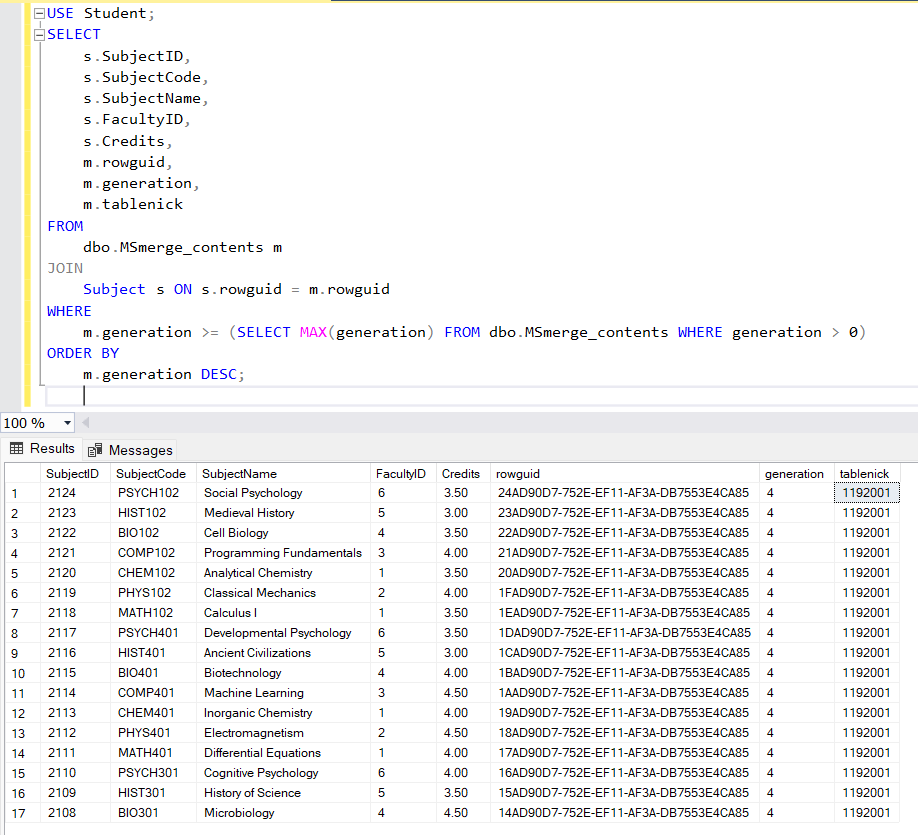
Slika 3.8. Tabela *Subject* na slave bazi *MSSQLSERVER03*.

Pošto smo podesili da se sinhronizacija vrši na 10 minuta, promene koje smo napravili u slave bazama podataka će biti dostupne u sve tri baze nakon sinhronizacije.



Slika 3.10. Prikaz tabele *Subject* na master serveru nakon sinhronizacije.

Nakon toga možemo izvršiti upit za proveru promena koje su se dogodile tokom poslednje sinhronizacije baza podataka. Ovaj upit vraća promene koje su se dogodile tokom poslednje *merge* replikacije. *dbo.MSmerge\_contents* je tabela koja sadrži informacije o promenama koje su nastale tokom replikacije.



Slika 3.9. Prikaz promena koje su se dogodile tokom poslednje sinhronizacije baza.

Na osnovu rezultata upita možemo zaključiti da su u poslednjoj sinhronizaciji sinhronizovane promene koje su se dogodile na slave *MSSQLSERVER02* bazi, odnosno dodavanje novih redova u tabeli *Subject.* Na kraju možemo zaključiti da su promene nastale na slave serverima *MSSQLSERVER02 i MSSQLSERVER03* uspešno sinhronizovane i na taj način demonstrirali bidirekcionu replikaciju.

# Snapshot replication

*Snapshot* replikacija u MSSQL Serveru predstavlja najjednostavniju tehniku distribucije podataka koja prenosi podatke tačno onako kako izgledaju u određenom trenutku, bez vođenja evidencije o istoriji promena. Kada se sinhronizacija pokrene ceo snimak baze podataka se generiše i šalje pretplatnicima.

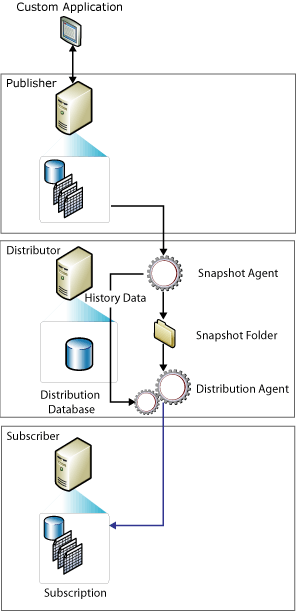
Ova vrsta replikacija je najpogodnija u sledećim scenarijima:

* Retke promene podataka: kada se podaci retko menjaju, *snapshot* replikacija je efikasno rešenje, jer nema potrebe za čestim ažuriranjima.
* Prihvatljivo kašnjenje u podacima: kada je prihvatljivo imati kopije podataka koje su neaktuelne u odnosu na izvornu bazu podataka u određenom vremenskom periodu.
* Replikacija male količine podataka
* Velike količine promena u kratkom periodu: kada se desi veliki broj promena u kratkom vremenskom periodu, *snapshot* replikacija može efikasno preneti te promene. [6]

*Snapshot* replikacija predstavlja efikasnu metodu za replikaciju podataka u specifičnim okruženjima. Poprilično je jednostavna i ne zahteva velike troškove, zato je pogodna za situacije gde se podaci retko menjaju ili gde je prihvatljivo kašnjenje u sinhronizaciji podataka. Međutim, za velike setove podataka ili okruženja sa čestim promenama, potrebno je pažljivo razmotriti resursne zahteve i frekvenciju *snapshot-a* kako bi se osigurala efikasnost i pouzdanost replikacije. Može se koristiti samostalno, ali se često koristi kao početna tačka za transakcione i *merge* publikacije.

## 4.1. Arhitektura

Kod *snapshot* replikacije u samu arhitekturu je uključeno dva agenta, *Snapshot* i *Distribution* agent. Oba agenta i njihova uloga su opisani kod transcakcionereplikacije.



Slika 4.1. Arhitektura *snapshot* replikacije. [6]

Proces *snapshot* replikacije obuhvata nekoliko koraka:

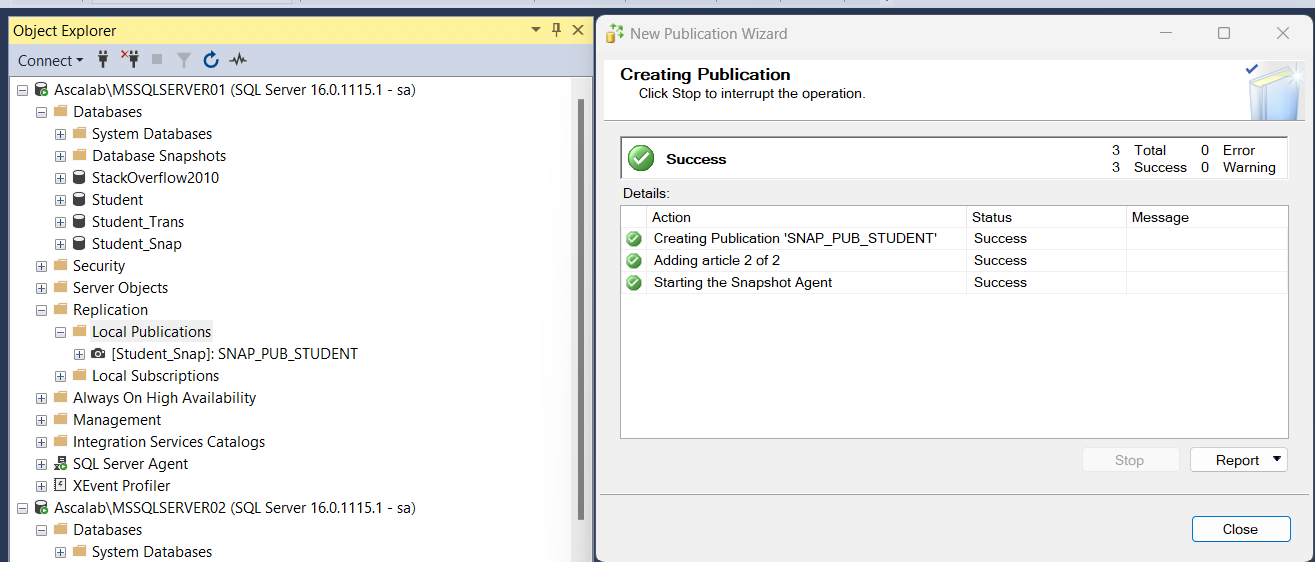
* **Generisanje *snapshot-a*:** *Snapshot a*gent uspostavlja vezu sa *publisher-om* i generiše snapshot fajlove. Ovi fajlovi sadrže šemu i podatke publikovanih tabela. *Snapshot* agent takođe dodaje redove u tabele *MSrepl\_commands* i *MSrepl\_transactions* u distribucionoj bazi podataka. Zapisi u tabeli *MSrepl\_commands* predstavljaju komande koje ukazuju na lokaciju .sch i .bcp fajlova, odosno fajlova koji čuvaju definicija šema i same podatke. Zapisi u tabeli *MSrepl\_transactions* su komande potrebne za sinhronizaciju pretplatnika.
* **Čuvanje *snapshot* fajlova:** *Snapshot* fajlovi se čuvaju u unapred definisanom folderu na distribueru. Ovaj folder može biti konfigurisan prilikom postavljanja distribucije, a može se definisati i alternativna lokacija.
* **Dostava i primena *snapshot-a*:** *Distribution* agent ispituje tabele *MSrepl\_commands* i *MSrepl\_transactions* u distribucionoj bazi podataka na distribueru. Agent čita lokaciju *snapshot* fajlova iz prve tabele i sinhronizacione komande pretplatnika iz obe tabele. Na osnovu komandi i *snapshot* fajlova iz distribucionog foldera i *distribution* agent primenjuje definisane akcije na bazu podataka pretplatnika. Kod *merge* replikacije, merge agent je imao ovu ulogu.
* **Zaključavanje tabela:** Tokom generisanja *snapshot-a*, tabele na *publisher-u* mogu biti zaključane kako bi se osigurala konzistentnost podataka. Ova zaključavanja mogu biti dugotrajna kod *snapshot* publikacija, dok su kod transakcionih replikacija kratkotrajna.

## 4.2. Konfiguracija

Kao i kod prethodnih konfiguracija replikacija, tako i kod *snapshot* replikacije potrebno je da postoje najmanje dve instanceSQL Servera kako bi mogli da uspostavimo replikaciju. Proces konfiguracije se sastoji iz istih koraka kao kod prethodne dve replikacije.

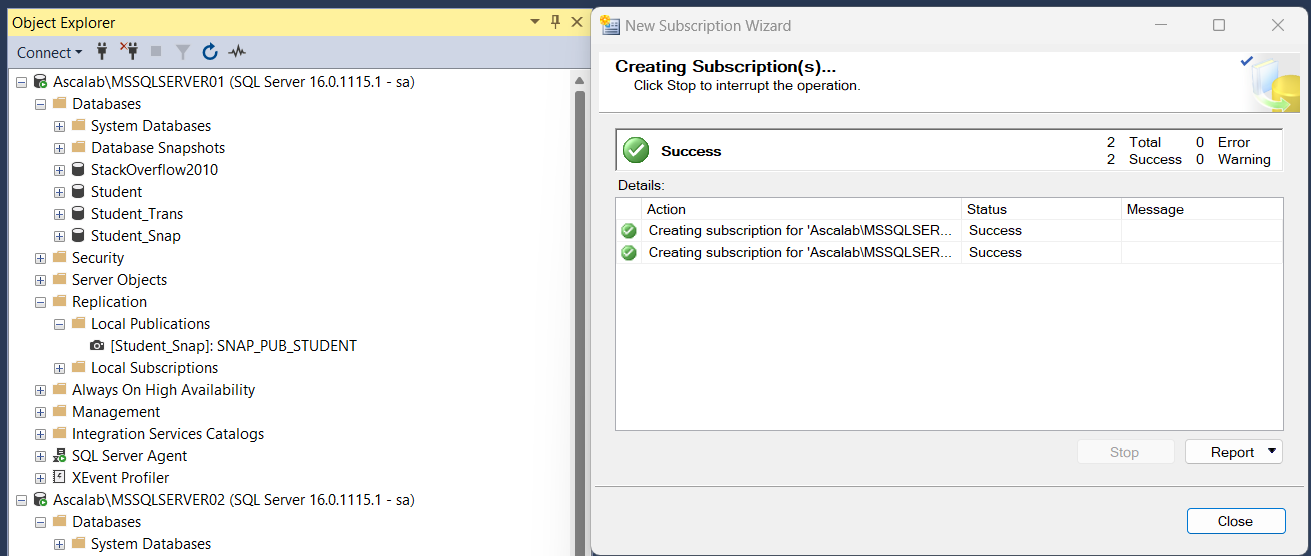
Prvi korak je konfiguracija distribucionog servera koja je identična kao i kod prethodnih replikacija. Konektujemo se na željenu SQL Server instancu i iz *Object Exlorer-a* desnim klikom na *Replication* folder otvaramo  *Configure Distribution.* Nakon toga je potrebno da dovršimo *wizard.*  
 Drugi korak jekonfiguracija publikacije*.* Konfiguracija publikacije se vrši na sledeći način:

1. Desnim klikom na *Replication*, a zatim na opciju *New Publication*.
2. Nakon toga se otvara wizard, gde je potrebno odabrati distribucionu bazu koju smo kreirali u prethodnom koraku.
3. U sledećem koraku biramo bazu za replikaciju, odnosno *Student\_Snap* bazu podataka.
4. Označimo da je tip replikacije *Snapshot Publication*.
5. Posle toga biramo tabele koje će biti uključene u publikaciju (kreiramo Articles). U ovom koraku biramo tabele *Faculty* i *Subject.*
6. U sledećem koraku možemo odabrati kada želimo da se inicijalizacija pretplatničkih baza izvrši. Možemo izabrati odmah nakon kreiranja pretplata ili definisati tačno vreme korišćenjem *scheduler-a.*
7. Definišemo sigurnosne konfiguracije za agente koje će prenositi podatke.
8. Na kraju potvrđujemo wizard i kreiramo publikaciju.

Slika 4.2. Kreiranje *snapshot* publikacije.

Treći korak predstavlja konfiguraciju pretplatnika. Konfiguracija pretplatnika se može izvršiti na sledeći način:

1. Desni klik na publikaciju koju smo kreirali (SNAP\_PUB\_STUDENT), a zatim *New Subscription*.
2. U sledećem koraku potrebno je odabrati push ili pull subscription. U ovom koraku ćemo nastaviti sa *push subscription*.
3. Nakon toga, definišemo drugu instancu SQL servera za pretplatnika. I unosimo sigurnosne postavke za distribucionog agenta.
4. U sledećem koraku možemo izabrati periodičnu ili konstantu distribuciju od strane distribucionog agenta.
5. Na kraju potvrđujemo wizard i kreiramo pretplatu.

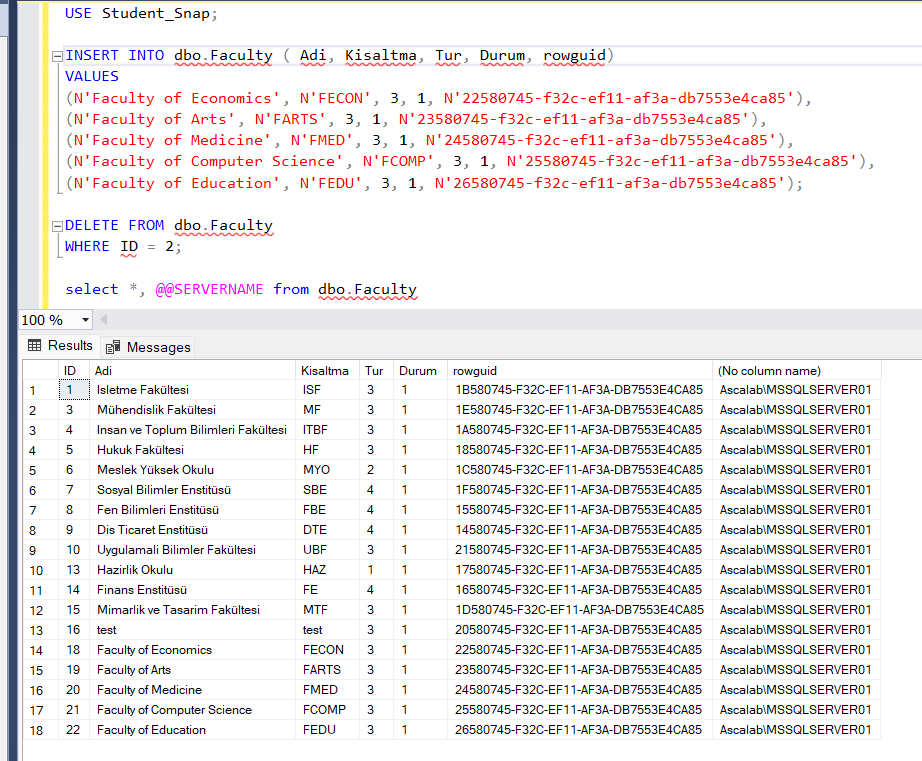


Slika 4.3. Kreiranje pretplatnika.

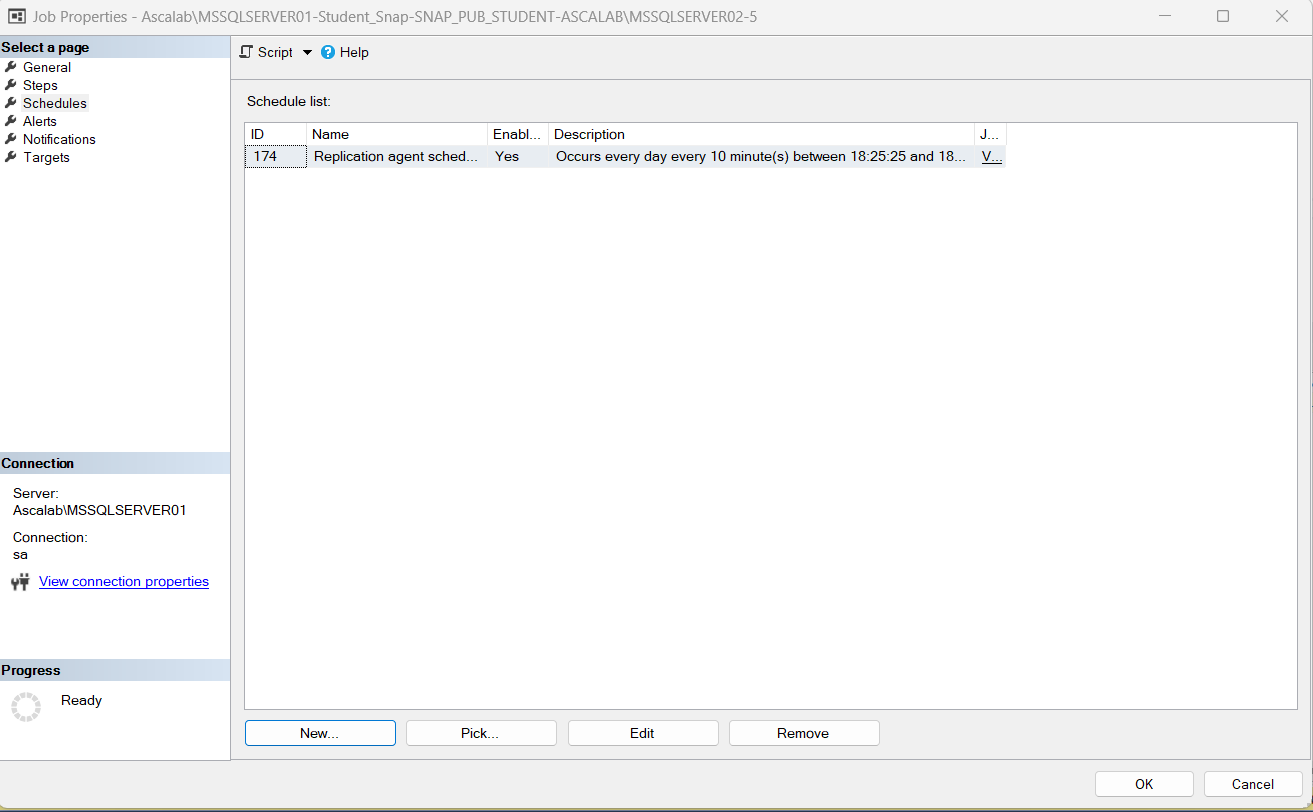
## 4.3. Primer

Kao i kod prethodnih replikacija *Snapshot* replikaciju ćemo demonstrirati korišćenjem iste arhitekture sa tri instance SQL Servera, *MSSQLSERVER01*, *MSSQLSERVER02* i *MSSQLSERVER03*. Na prvoj instanci nalazi se baza podataka *Student\_Snap* koja će biti replicirana na ostale servere. Ostale dve instance imaju samo kreirane baze podataka *Student\_Snap*.

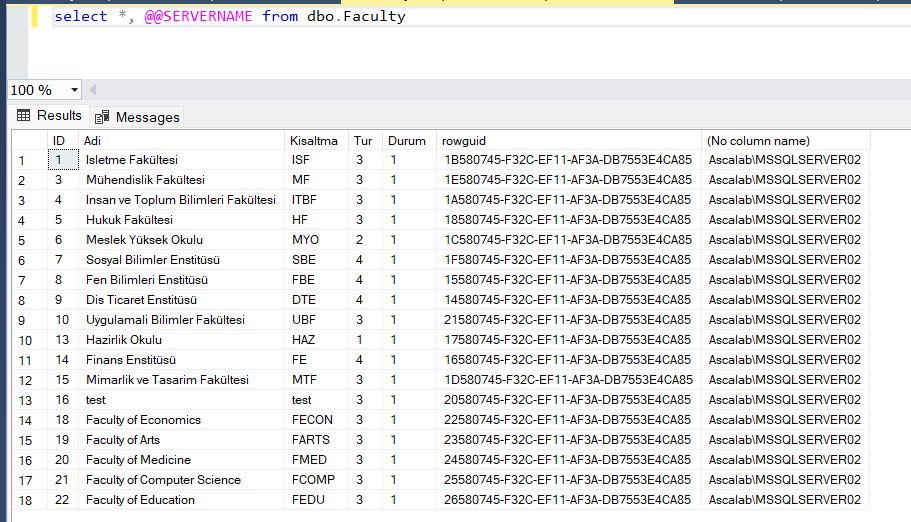
Da bi testirali da li replikacija radi, izvršićemo sledeće promene nad tabelom *Faculty* u okviru master baze *Student\_Snap.*



Slika 4.4. Izvršavanje upita na master bazi.

Slika 4.5. Prikaz sledeće sinhronizacije.

Pošto je sinhronizacija podešena na svakih 10 minuta, promene koje smo izvršile biće dostupne tek nakon 10 minuta na repliciranim bazama. Ako ponovimo query za selektovanje podataka iz tabele *Faculty* nad jednom od repliciranih baza, videćemo da su rezultatiti identični kao na master bazi.

Slika 4.6. Prikaz podataka u tabeli *Faculty* nakon sinhronizacije.

Na ovaj način smo prikazali kako se podaci repliciraju sa master baze na slave baze podataka. Pošto *snapshot* replikacija nije bidirekcionalna, podaci se samo prenose sa izvorne baze na replikacijske.

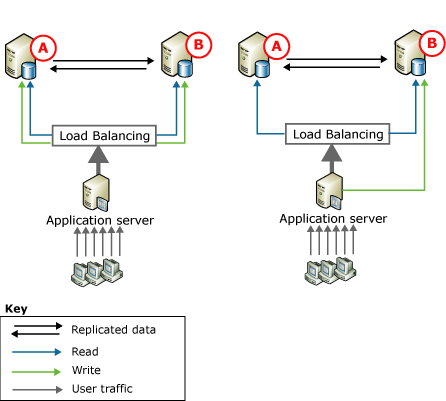
# Peer-to-peer replication

*Peer-to-peer* replikacija je napredna tehnika replikacije koja se koristi za skaliranje performansi čitanja i povećanje dostupnosti podataka u distribuiranim sistemima. Ovaj metod replikacije je zasnovan na transakcionoj replikaciji i omogućava održavanje identičnih kopija podataka na više servera, koji se nazivaju čvorovi. Ona radi tako što propagira transakcijski konzistentne promene u skoro realnom vremenu između svih čvorova u topologiji. Ovo znači da svaka promena podataka na jednom čvoru bude gotovo odmah replicirana na sve ostale čvorove, čime se osigurava da svi čvorovi imaju isti skup podataka.

Jedna od glavnih prednosti *peer-to-peer* replikacije je mogućnost skaliranja performansi čitanja. Čitanja se mogu distribuirati preko više čvorova, čime se smanjuje opterećenje na pojedinačnim serverima i omogućava brži odgovor na zahteve. Takođe, u slučaju kvara jednog čvora, aplikacije mogu preusmeriti operacije pisanja i čitanja na drugi čvor, čime se osigurava kontinuitet rada i dostupnost podataka. Ovo je posebno korisno za aplikacije koje zahtevaju neprekidnu dostupnost, poput web aplikacija i sistema za upravljanje podacima u realnom vremenu. S obzirom na to da svi čvorovi sadrže identične kopije podataka, peer-to-peer replikacija obezbeđuje visok nivo redundancije. Ova redundancija dodatno doprinosi dostupnosti i sigurnosti podataka. [7]

Kod ove vrste replikacije postoje par ograničenja koja mogu uticati na efikasnost samog sistema. Performanse pisanja nisu skalabilne kao performanse čitanja. Svaka promena podataka (insert, update, delete) mora biti propagirana na sve čvorove, što može dovesti do latencije i konflikata u slučaju paralelnih promena na različitim čvorovima. U slučaju da dođe do konflikta može doći do gubljenja podataka prilikom sinhronizacije čvorova, ili u nekim slučajevima prijavu greške i prekid rada distribucionog agenta. Zato se preporučuje da se upis podataka vrši na jednom čvoru, a da se promene repliciraju na ostale.

## 5.1. Arhitektura



Slika 5.1. Topologija *peer-to-peer* replikacije [7]

Na slici su date topologije koje se najčešće koriste u praksi kod *peer-to-peer* replikacije. Na levoj strani slike prikazana je konfiguracija sa dve baze podataka označene kao A i B, i podaci se repliciraju između baza u oba smera. Čitanje podataka se obavlja sa obe baze podataka na osnovu raspodele saobraćaja pomoću *load balancing-a*. Upisivanje podataka je obično podeljeno između baza A i B. Tako da se na osnovu određenih kriterijuma upis vrši u bazu A (ako podataka podleže kriterijumima za tu bazu), a u suprotnom podaci se upisuju u bazu B. Na primer, proizvodi čije šifre se sastoje od cifara se upisuju u bazu A, dok proizvodi čije šifre se sastoje od kombinacije karaktera i cifara se upisuju u bazu B. Na taj način se izbegavaju konflikti. Korisnici pristupaju bazama podataka preko aplikacionog servera koji raspodeljuje saobraćaj između baza.   
Desna strana slike prikazuje drugu topologiju sa bazama A i B. Kao i u prvoj topologiji, podaci se repliciraju između baza u oba smera. Čitanje podataka se obavlja sa obe baze na osnovu raspodele saobraćaja. Međutim, u ovoj topologiji, sve pisanje podataka se usmerava na bazu B, koja zatim replicira promene u bazu A. U slučaju da baza B nije dostupna, aplikacioni server preusmerava sva čitanja i pisanja na bazu A. Kada baza B ponovo postane dostupna, uloga ažuriranja se mogu vratiti na bazu B. [7]

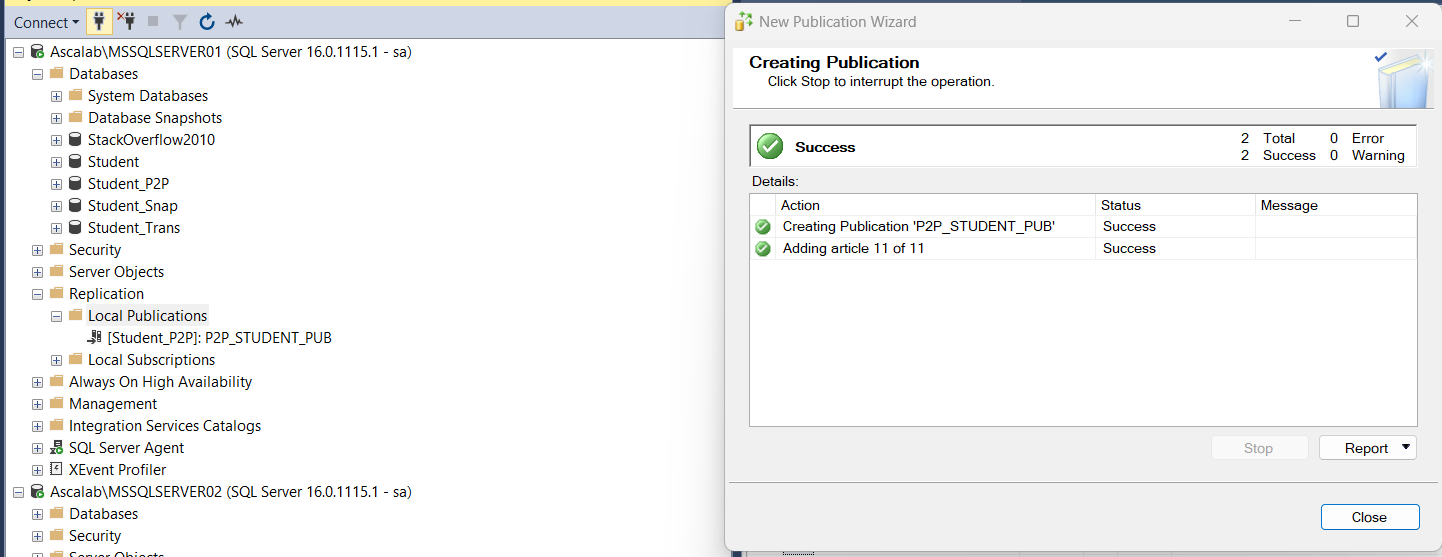
Korišćenjem ovakve topologije povećavaju se performanse čitanja, jer su čitanja raspoređena na dva servera, što omogućava obradu većeg broja zahteva. Takođe, obezbeđuje visoku dostupnost, jer drugi čvor može preuzeti sav saobraćaj u slučaju kvara na jednom čvoru, osiguravajući kontinuitet rada sistema.

## 5.2. Konfiguracija

Prvi korak je konfiguracija distribucionog servera koja je identična kao i kod prethodnih replikacija. Konektujemo se na željenu SQL Server instancu i iz Object Exlorer-a desnim klikom na Replication folder otvaramo Configure Distribution. Nakon toga je potrebno da dovršimo wizard. Jedini preduslov kod ove vrste replikacije je da ovaj korak mora da se odradi na svim instancama koje će učestvovati u replikaciji.

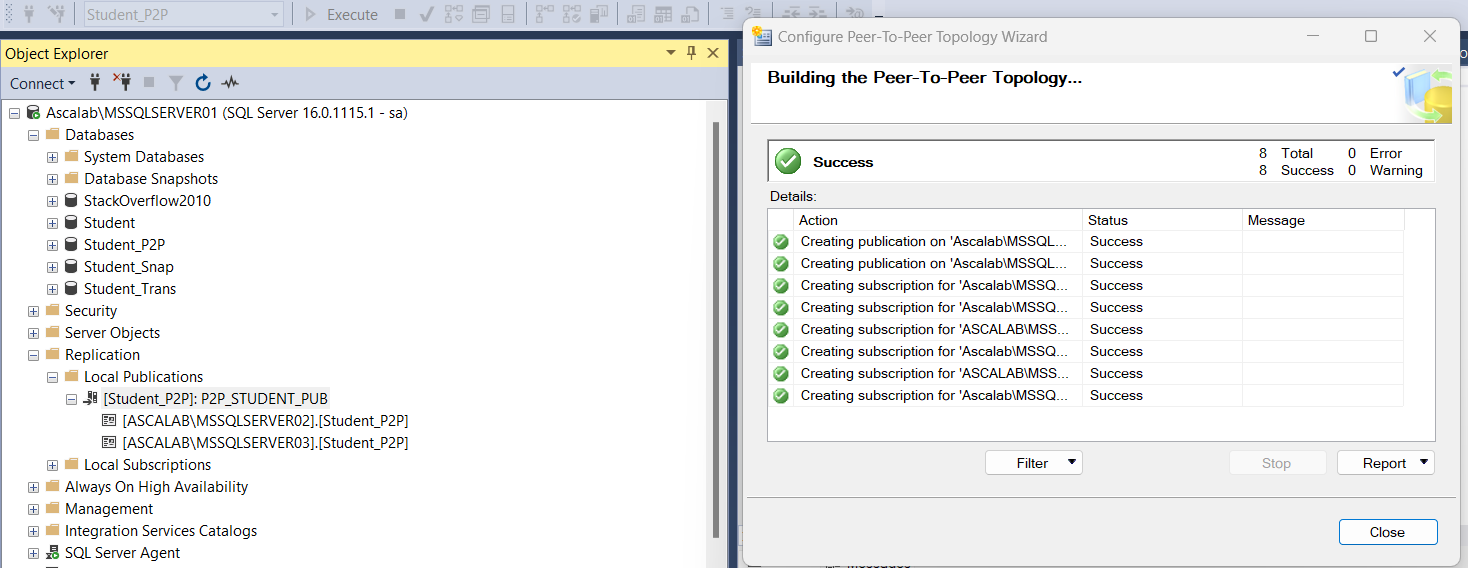
Drugi korak je konfiguracija publikacije. Konfiguracija publikacije se vrši na sledeći način:

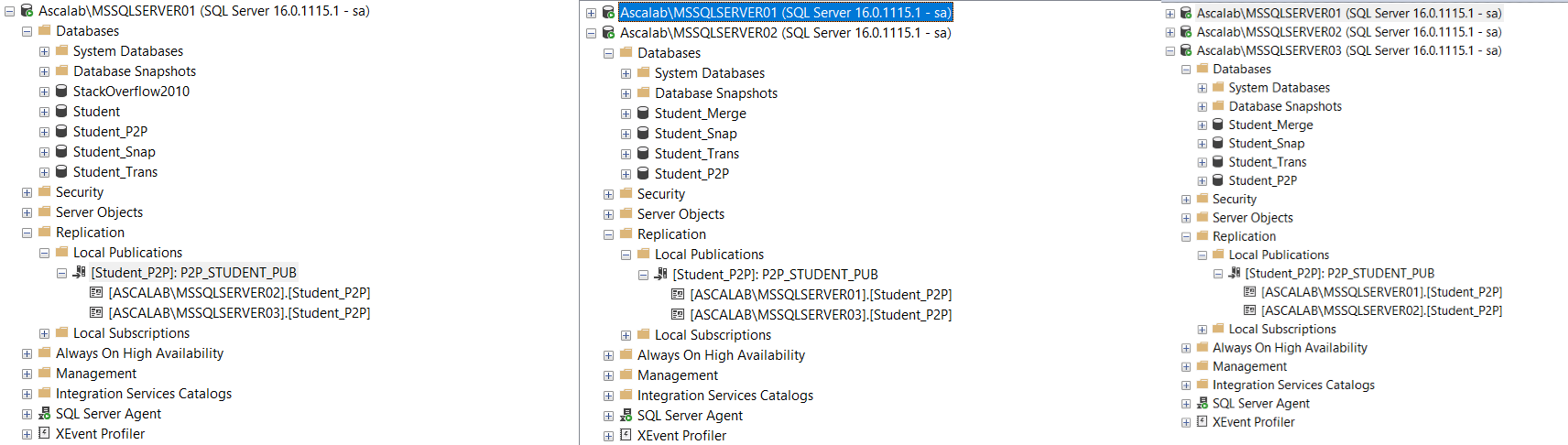
1. Desnim klikom na *Replication*, a zatim na opciju *New Publication*.
2. Nakon toga se otvara wizard, gde je potrebno odabrati distribucionu bazu koju smo kreirali u prethodnom koraku.
3. U sledećem koraku biramo bazu za replikaciju, odnosno *Student\_P2P* bazu podataka.
4. Označimo da je tip replikacije *Peer-to-Peer Publication*.
5. Posle toga biramo tabele koje će biti uključene u publikaciju.
6. Definišemo sigurnosne konfiguracije za *Log Reader Agenta.*
7. Na sledećem koraku definišemo ime publikacije (*P2P\_STUDENT\_PUB*).
8. Na kraju potvrđujemo wizard i kreiramo publikaciju.

Slika 5.2. Kreiranje *P2P* publikacije.

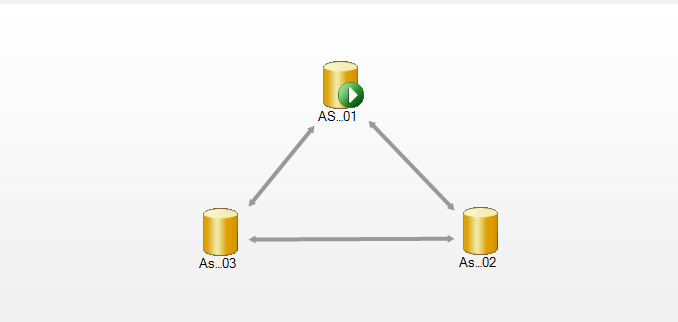
Treći korak predstavlja konfiguraciju čvorova. Konfiguracija čvorova se može izvršiti na sledeći način:

1. Desni klik na publikaciju koju smo kreirali (*P2P\_STUDENT\_PUB*), a zatim *Configure Peer-to-Peer Topology*.
2. Iz wizarda je potrebno dodati ostale instance kao čvorove. U ovom slučaju doddajemo i MSSQLSERVER02 i MSSQLSERVER03 kao čvorove.
3. Nakon toga je potrebno sigurnosne postavke za *Log Reader Agenta* i *Distribution Agenta.*
4. Na sledećem koraku je važno napomenuti da su baze na ostalim instancama kreirane putem *backup-a* i *restore-a*.
5. Potvđivanje wizarda kreiraće se čvorovi.

Slika 5.3. Kreiranje čvorova.

Slika 5.4. Uporedni prikaz replikacijskih čvorova.

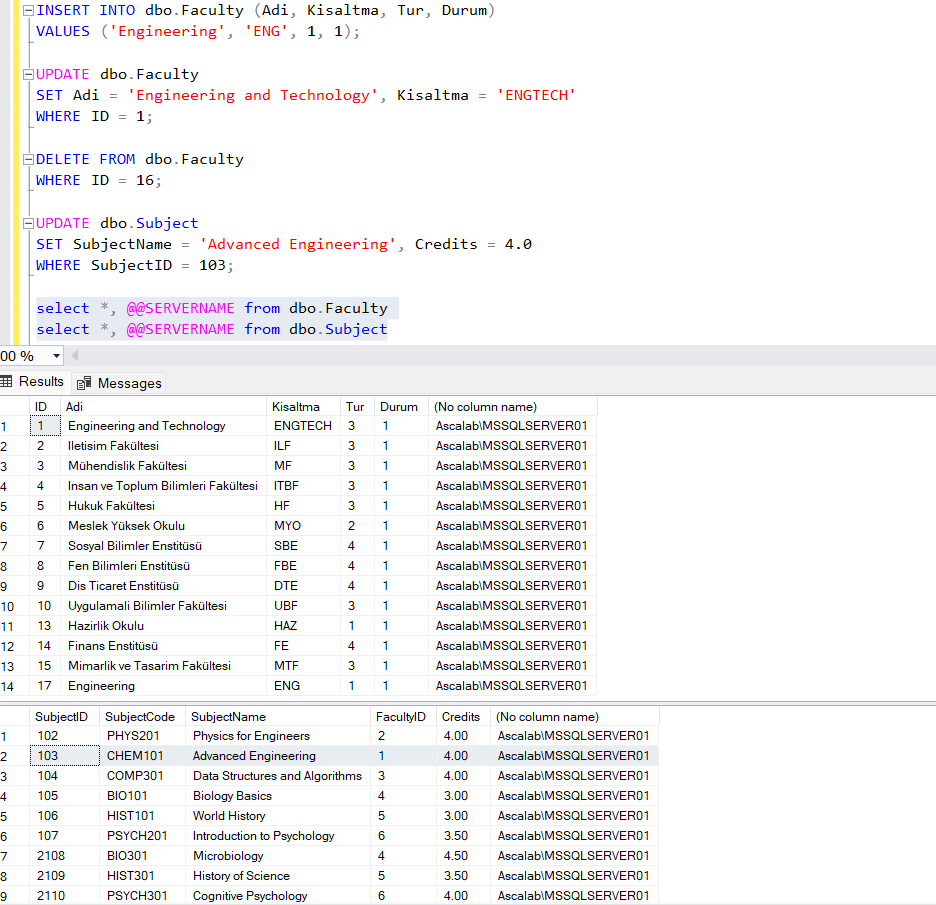
## 5.3. Primer



Slika 5.5. Prikaz topologije replikacije.

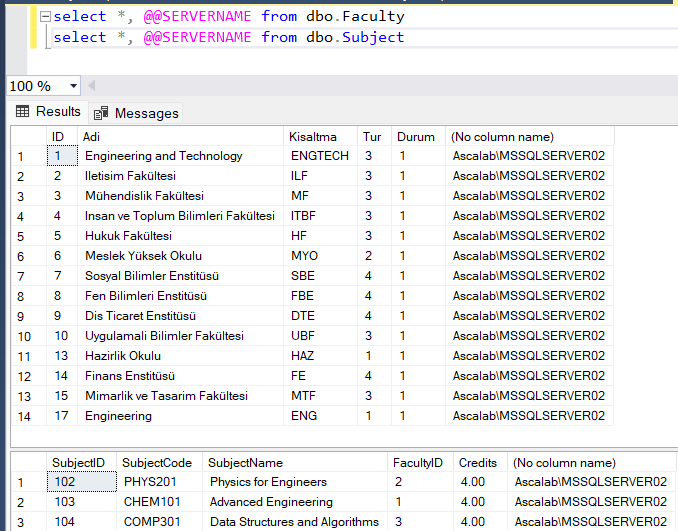
U prethodnom poglavlju je opisano kako se vrši konfiguracija ovakve topologije. Na lokalnoj mašini imamo postavljene 3 instance SQL Servera, *MSSQLSERVER01*, *MSSQLSERVER02* i *MSSQLSERVER03.* Na prvom serveru postoji baza podataka *Student\_P2P.* Nakon toga moramo da kreiramo *backup* fajl baze koji ćemo iskoristiti za kreiranje kopija na ostalim serverima. Nakon kreiranja baza kopija, izvršili smo kreiranje čvorova kao što je opisano u prethodnom poglavlju i dobili topologiju koja je predstavljena na slici 5.5.

Da bi smo demonsrirali kako se podaci replikuju sa jedne baze na drugu, izvršićemo promene na čvoru 01. Na slici je dat upit koji će napraviti promene i u tabeli *Faculty* i tabeli *Subject.*

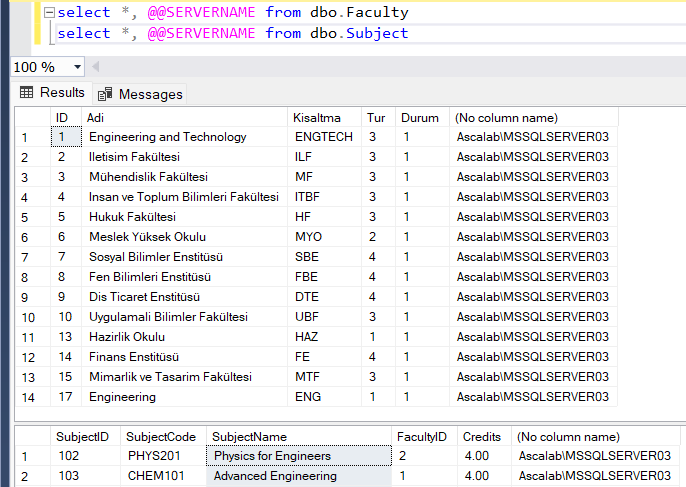


Slika 5.6. Izvršavanje upita na čvoru 01.

Ako ponovimo upit za selektovanje na čvoru 02 ili 03, videćemo da su se promene replikovale i na tim čvorovima.



Slika 5.7. Prikazivanje promena na čvoru 02.



Slika 5.8. Prikazivanje promena na čvoru 03.

Ovim primerima smo demonstrirali kako se promene prenose i na ostale čvorove u topologiji.

# Database mirroring

*Mirroring* baze podataka u SQL Server-u je tehnika za oporavak od otkaza i omogućava visoku dostupnost. Ova tehnika uključuje dve kopije jedne baze podataka koje se obično nalaze na različitim računarima. U svako trenutku samo jedna kopija baze podataka je dostupna klijentima. Takva kopija se naziva *principal* baza podataka. Promene koje se odvijaju na *principal* bazi podataka se izvršavaju i na drugu kopiju baze koja se još i naziva *mirror* baza podataka. Tehnika *Mirroring* podrazumeva korišćenje transakcionog loga za svaku umetanje, ažuriranje ili brisanje napravljeno u *principal* bazi podataka na *mirror* bazu podataka.

Prednosti ovakve tehnike jesu:

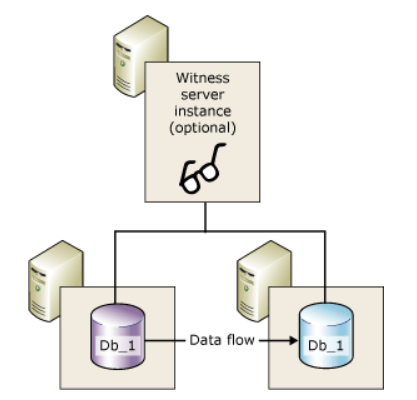
1. Povećana dostupnost baze podataka: U slučaju otkaza *principal* baze, strategija *mirrorovanja* baze može brzo prebaciti *mirror* bazu podataka da bude primarna. U režimu visoke sigurnosti sa automatskim prebacivanjem, ovaj proces se dešava bez gubitka podataka, obezbeđujući minimalno vreme zastoja.
2. Minimizirano vreme zastoja tokom *upgrade-a*: *Mirrorovanje* baze podataka podržava sekvencijalne *upgrade-a*, gde možemo vršiti *upgrade* instanci SQL Server-a na način da uvek jedna instanca bude dostupna. Ovaj pristup smanjuje vreme zastoja na trajanje samo jednog prebacivanja aktivne uloge sa jedne na drugu instancu. Na taj način se poboljšava ukupna dostupnost produkcijske baze podataka tokom održavanja.
3. Poboljšanje zaštite podatake: *Mirrorovanje* baze podataka održava redundantnu kopiju podataka. U zavisnosti od režima rada, ova redundancija može biti potpuna ili skoro potpuna, i na taj način pruža zaštitu podataka od hardverskih kvarova i drugih prekida. Npr. *mirrorovanje* baza podataka može automatski rešavati grešake kao što su nečitljive stranice podataka. Instanca koja ima problem sa stranicom traži svežu kopiju problematične stranice od druge instance. Nakon čega će izvršiti zamenu nečitljivu stranice i rešiti problem. [8]

*Mirrorovanje* baza podataka u SQL Serveru je moćna tehnika za povećanje dostupnosti baza podataka i zaštite podataka. Održavanjem sinhronizovane ili skoro sinhronizovane kopije baze podataka, omogućava brži oporavak od kvarova i grešaka i smanjuje vreme zastoja.

## 6.1. Načini rada

SQL Server *mirroring* baze podataka može biti podešen da obezbedi visoku dostupnost ili oporavak od otkaza. U zavisnosti od potreba sistema, mogu se izabrati jedan od tri načina rada:

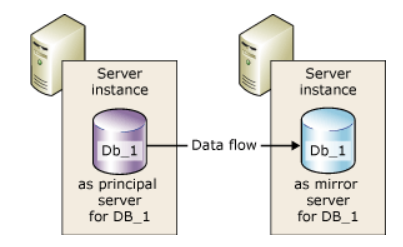
* Visoka sigurnost (*High Safety*) – Podaci se sinhronizovano upisuju i potvrđuju na *principal* i *mirror* bazi podataka. Tek nakon potvrde na obe baze, aplikacija baze podataka može nastaviti sa aktivnostima.
  + Ako *principal* baza podataka prestane da radi postoje dve opcije. Prva opcija je da se sačeka da *principal* baza podataka postane dostupna. A druga opcija je da se odradi prisilna promena na *mirror* bazu. Odnosno da *mirror* baza postane *principal* baza podataka. Ali u tom slučaju može doći do gubitka određenih transakcija koje su bile potvrđene na *principal* bazi, ali nisu bile prenete na *mirror* bazu podataka.
  + Ovaj način rada može dovesti do određenih kašnjenja jer se transakcije moraju potvrditi na obe baze podataka.
* Visoka sigurnost sa automatskim prebacivanjem *(High Safety with Automatic Failover) -* Kod ovog režima potrebne su tri instance servera. Podaci se sinhronizovano upisuju i moraju biti potvrđeni na *principal* i *mirror* bazi. Dok se podaci ne potvrde na obe baze, aplikacija ne može nastaviti sa radom.
  + Može izazvati kašnjenje i sporiji rad jer transakcije moraju biti potvrđene na obe baze.
  + Ako glavna baza podataka prestane da radi, izvršiće se automasko prebacivanje na *mirror* bazu podataka. Za automsko prebacivanje zadužen je *Witness* server koji ne sadrži bazu podataka već služi samo za obzervaciju da li je došlo do otkaza na *principal* bazi podataka. U slučaju da je došlo do otkaza, odnosno *witness* server vidi da je samo *mirror* baza dostupna, tada će se pokrenuti automasko prebacivanje *mirror* baze u *principal.*



Slika 6.1. Ilustracija načina rada „Visoka sigurnost sa automatskim prebacivanjem”[8]

* Visoke performanse *(High Performance*) – Kod ovog načina rada imamo asinhronu komunikaciju. Podaci se upisuju i potvrđuju na *principal* serveru, a zatim se šalju i potvrđuju na *mirror* serveru. Automatsko prebacivanje nije moguće i *witness* server se ne može koristiti.
  + Ako glavna baza podataka prestane da radi, dostupne su tri opcije:

1. Čekati da glavna baza ponovo postane dostupna. SQL Server je nedostupan. *Mirroring* će se nastaviti tamo gde je stao.
2. Prisiliti SQL Server instancu na *mirror* bazu podataka. Odnosno da *mirror* baza postane glavna. Veća mogućnost za gubitak podataka zbog asinhrone komunikacije između baza.
3. Ručno ažuriranje – da bi se smanjio gubitak podataka, napraviti rezervnu kopiju kraja loga ako to dozvoljava neuspeli server, ukloniti mirroring i obnoviti kraj loga na prethodno mirror bazi.



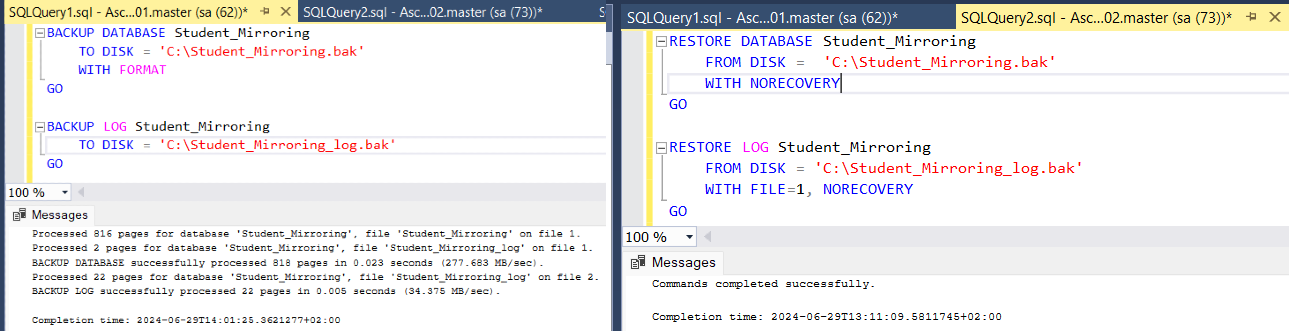
Slika 6.2. Ilustracija načina rada „Visoka sigurnost” i „Visoke performanse” [8]

U suštini sve vrste rada podržavaju samo jednu *principal* i jednu *mirror* bazu podataka.

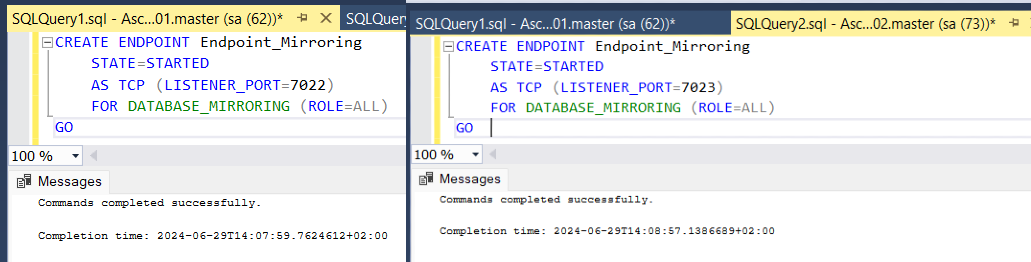
## 6.2. Konfiguracija

U daljem tekstu biće opisano kako je moguće konfigurisati *mirroring* baze podataka korišćenjem Management Studio-a. Konfiguracija se vrši iz nekoliko koraka:

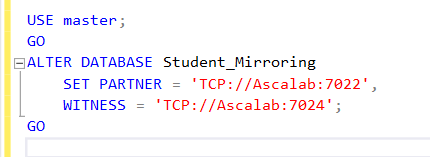
Prvi korak jeste kreiranje *backup* fajla *principal* baze podataka. Nakon toga je potrebno izvršiti *restore* te baze na *mirror* instanci SQL Servera. Osim *backup-a* baze podataka potrebno je ponoviti isti proces za transakcioni log. Na taj način osiguravamo da se *principal* i *mirror* baza podataka nalaze u istom početnom stanju.

Slika 6.3. Kreiranje *backup* fajlova i izvršavanje *restore* baze podataka na *mirror* serveru.

Drugi korak konfiguracije je podešavanje *mirroring-a*. Na principal i *mirror* instanci potrebno je kreirati *endpoint-e*. Izvršavanjem komande *CREATE ENDPOINT <ime\_endpoint-a>* kreiramo *endpoint* kako bi omogućili povezivanje sa drugim serverom. Navodimo *STATE = STARTED* kako bi usluga bila dostupna odmah nakon kreiranja. *AS TCP (LISTENER\_PORT=<PORT>)* definiše da *endpoint* koristi TCP/IP protokol za komunikaciju između glavnog i *mirroring* servera, dok *LISTENER\_PORT=<PORT>* specificira da će *endpoint* slušati na definisanom portu. *FOR DATABASE\_MIRRORING (ROLE=ALL)* specificira da je endpoint namenjen za *mirroring* i da će podržavati sve uloge (Primary, Mirror ili Witness). Na *witness* instanci je potrebno kreirati *endpoint* na drugom portu (npr. 7024), sa ulogom *WITNESS*.

Slika 6.4. Kreiranje *endpoint-a* na *principal* i *mirror* bazi*.*

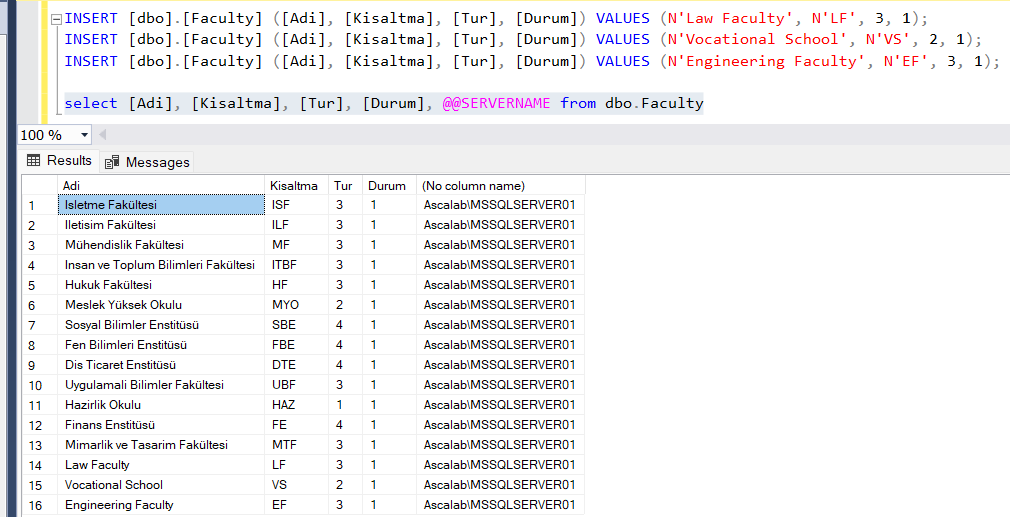
Nakon toga, potrebno je postaviti adrese ka ostalim instancama. Na *principal* i *mirror* bazi potrebno je postaviti adrese ka server partner instancama. Na slici je dat primer kako se povezuju partneri.

  
Slika 6.5. Povezivanje partner instanca.

Nakon uspešno izvršenih prethodnih koraka *mirroring* baze je aktivan i sve promene će biti preslikane sa *principal* instance na *mirror.* U slučaju otkaza *principal* instance, *witness* će inicirati prelazak *mirror* instance u *principal.*

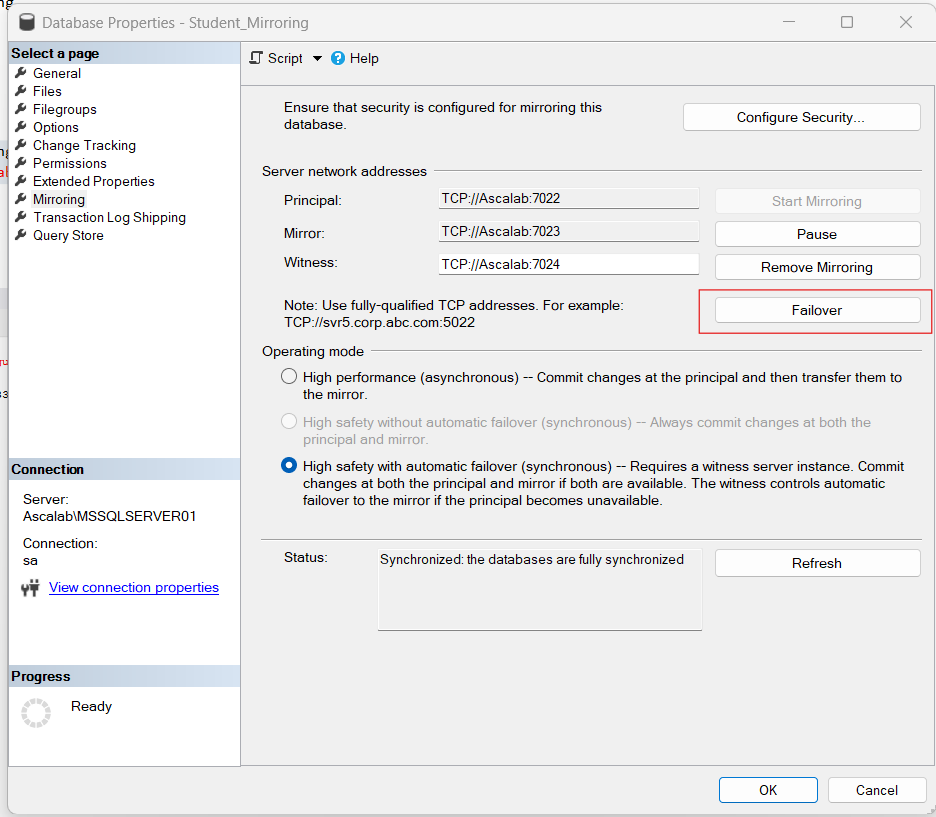
6.3. Primer

Na osnovu konfiguracije iz prethodnog dela možemo izvršiti testiranje arhitekture. Potrebno je da izvršimo promene na *principal* bazi i da proverimo da li se te promene prenose na *mirror* bazu. Na slici su dati primeri transakcija koje će dodati tri nova zapisa u tabelu *Faculty*.



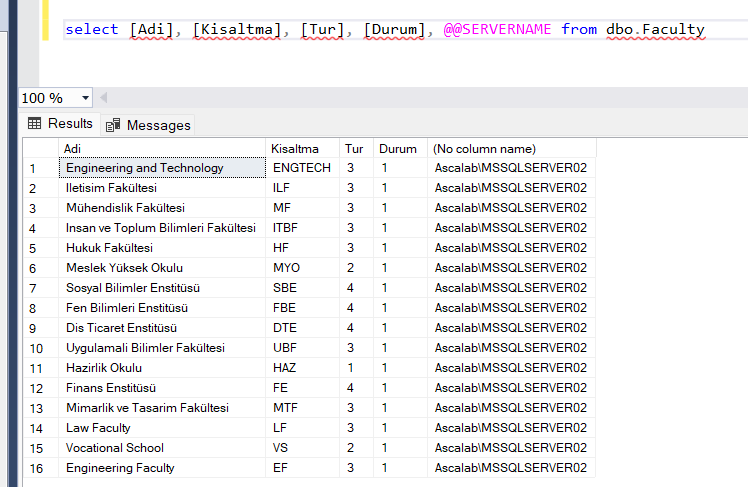
Slika 6.6. Izvršavanje transakcija na *principal* bazi.

Nakon toga ćemo izvršiti promenu baza, odnosno da *mirror* baza postane *principal* tako što ćemo ručno izazvati otkaz *principal* instance.



Slika 6.7. Izazivanje otkaza *principal* baze.

Ako sada izvršimo na drugoj instanci upit koji selektuje podatke iz tabele *Faculty*, možemo videti da su podaci bili preneti na *mirror* instancu.



Slika 6.8. Prikaz promena na *mirror* instanci.

Na ovoj slici možemo videti da su se transakcije uspešno prenele na *mirror* instacu. Na ovaj način smo demonstrirali kako *mirroring* arhitektura funkcioniše.

# Zaključak

Replikacija i *mirroring* u SQL Serveru predstavljaju tehnike koje značajno unapređuju upravljanje bazama podataka, distribuciju podataka i povećavaju pouzdanost samih sistema koji ih koriste. U ovom radu su istraženi različiti aspekti ovih tehnologija i demonstrirana njihova primena na realnim primerima.

SQL Serveru nudi više vrsta načina replikacije: transakcijsku, *merge, snapshot* i *peer-to-peer.* Svaka vrsta replikacije ima svoje specifične karakteristike, te na osnovu tih karakteristika biramo određenu vrstu replikacija koja će zadovoljiti potrebe našeg zahteva. SQL Server replikacija pruža balansiranje opterećenja, poboljšava dostupnost i omogućava efikasnu replikaciju i na geografski udaljenim serverima.

S druge strane, m*irroring* predstavlja strategiju za povećanje dostupnosti baze podataka i zaštite podataka. Održavanje sinhronizovane ili delimično sinhronizovane kopije baze podataka, *mirroring* olakšava brži oporavak od kvaraova i smanjuje vreme oporavka sistema. Različiti načini rada, kao što su visoka sigurnost, visoka sigurnost sa automatski prebacivanjem i visoke performanse nude efikasno rešenje za različite potrebe.

Obe tehnike, replikacija i *mirroring* u SQL Serveru su ključne tehnike za omogućavanje visoke dostupnosti samih sistema, efikasnu distribuciju i brži oporavak sistema kod otkaza. Ove tehnike su neizostavne u današnjim sistemima, od kojih se očekuju da budu uvek dostupni sa što boljim odzivom.

# 8. Literatura

[1] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/transactional/transactional-replication> [Poslednji put posećeno 8.6.2024.]

[2] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/transactional/media/trnsact.gif> [Poslednji put posećeno 8.6.2024.]

[3] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/tutorial-replicating-data-between-continuously-connected-servers?view=sql-server-ver16> [Poslednji put posećeno 16.6.2024.]

[4] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/monitor/overview-of-the-replication-monitor-interface?view=sql-server-ver16> [Poslednji put posećeno 16.6.2024.]

[5] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/merge/merge-replication?view=sql-server-ver16> [Poslednji put posećeno 17.06.2024.]

[6] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/snapshot-replication?view=sql-server-ver16> [Poslednji put posećeno 19.06.2024.]

[7] <https://learn.microsoft.com/en-us/sql/relational-databases/replication/transactional/peer-to-peer-transactional-replication?view=sql-server-ver16> [Poslednji put posećeno 21.06.2024.]

[8] <https://learn.microsoft.com/en-us/sql/database-engine/database-mirroring/database-mirroring-sql-server?view=sql-server-ver16> [Poslednji put posećeno 24.06.2024.]

[9] <https://learn.microsoft.com/en-us/sql/database-engine/database-mirroring/database-mirroring-establish-session-windows-authentication?view=sql-server-ver16> [Poslednji put posećeno 27.06.2024.]