Visoka dostupnost SQL servera

Smaila, Dora

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:583430

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Dora Smaila

Visoka dostupnost SQL servera

Završni rad

Mentor: prof.dr.sc. Patrizia Poščić

Rijeka, rujan 2023.

Sadržaj

Rijeka, 15.09.2023

Zadatak za završni rad

Pristupnik: Dora Smaila

Naziv završnog rada: Visoka dostupnost SQL servera

Naziv završnog rada na engleskom jeziku: High availability of SQL Servers

Sadržaj zadatka: Tema zadatka je objasniti što je visoka dostupnost, kako se postiže u SQL infrastrukturi, te postupci upravljanja bazom podataka. Također, pričat ću o pojmovima vezanih uz visoku dostupnost, poput Failover cluster-a, Availability grupa, oporavku od katastrofe, te osnovama SQL jezika. Cilj je prikazati kako postići repliciranje ili kopiranje baze podataka i samih podataka na sekundarni server prilikom gašenja primarnog.

Mentor

prof.dr.sc. Patrizia Poščić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 01.05.2023.

(potpis pristupnika)

Sažetak

Kroz moj rad opisat ću ulogu visoke dostupnosti, što je sve potrebno da bi se postigla, te koje karakteristike nosi sa sobom.
Tehnologije koje sam koristila su Microsoftovi alati poput Azure-a, SQL Server Manager-a (SSMS) i Faliover clustering manager-a.

Tema mog rada je objasniti što se dogodi s našim podacima tijekom korištenja nekog servisa, aplikacije ili baze podataka ako jedan server prestane raditi. Svrha visoke dostupnosti je sljedeća: ukoliko jedan server prestane raditi, drugi preuzima (replicira ili kopira) sve podatke sa prvog servera.
Jedan od najvažnijih termina koji ću koristiti je pojam cluster. Cluster je skup čvorova, odnosno servera koji rade kao jedan sistem i omogućuju neometan rad na bazama podataka (ili bilo kojem drugom servisu) bez da korisnici osjete ikakve ili minimalne promjene.
Drugim riječima, cluster-om postižemo pouzdanost, redundantnost, odnosno visoku dostupnost SQL servera.

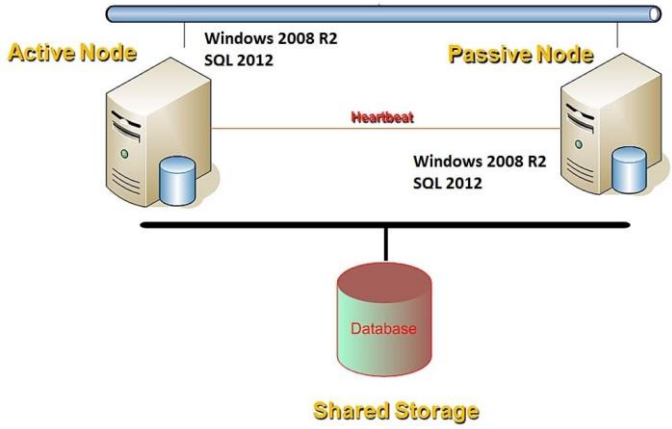
Ključne riječi

1. Cluster
2. Server
3. Visoka dostupnost
4. Baza podataka
5. Tablica

1. Uvod

Visoka dostupnost (eng. High availability, HA) je jedna od bitnijih karakteristika koja omogućuje korisnicima neometan rad, bez radnih preopterećenja. U praksi, najvažnija stvar je osigurati našim korisnicima da im naše usluge, aplikacije, pa tako i baze podataka uvijek rade i da su uvijek dostupni online. Pojam dostupnosti ima dvije definicije. Prva definicija glasi: „dostupnost je stanje u kojem potrošač može pristupiti određenom resursu“[17], što znači da korisnik u tom trenutku može pristupati podacima. Druga definicija glasi: „dostupnosti je postotak vremena u kojem se sustav može koristiti za produktivan rad“[17], odnosno potrebna dostupnost može varirati od korisnika do korisnika.

Visoka dostupnost je dizajnirana da u 99.999% slučajeva sustav bude dostupan, imajući na umu da se svejedno može dogoditi zastoj. Ova karakteristika se ujedno zove i *Five nines realibility*, odnosno sustav je skoro uvijek upaljen i funkcionalan. Time dolazimo do pitanja kako omogućiti, drugim riječima, kako postići visoku dostupnost. Način na koji sam postigla visoku dostupnost je pomoću Failover cluster-a, a on radi na principu aktiv-pasiv (Slika 1.).



Slika 1: Visoka dostupnost

Ukoliko se dogodi da aktivni server prestane raditi, odnosno postane pasivni, sekundarni postaje aktivan. [2]

Sekundarni server je također upaljen i prisutan, odnosno „stand-by“, ali sve dok se cijeli sustav ne preseli na njega, baza podataka se neće automatski replicirati. Drugi princip na kojem Failover cluster može funkcionirati je aktiv-aktiv. U tom slučaju je svaki server upaljen i radi jednako kao i primarni server. Cijeli sustav primarnog servera se kopira na sekundarni putem load balancer-a. Load balancer je uređaj koji raspoređuje servise na sve dostupne čvorove kako ne bi cijeli sustav ovisio o jednom čvoru.

Osim konstantne dostupnosti sustava, HA je ujedno i dodatan backup podataka. Dogodi li se situacija u kojoj primarni server padne, ne moramo se brinuti o nestanku podataka ili cijele infrastrukture jer je sve premješteno automatski na sekundarni server.

Važno je napomenuti da je kod konfiguracije servera (čvorova) bitno da svi imaju iste karakteristike. Tako korisnici neće primijetiti razliku ukoliko dođe do failover-a.

2. Visoka dostupnost

2.1. Azure

Azure je Microsoft-ov alat za Cloud infrastrukturu kojim možemo stvarati. konfigurirati svakakve resurse, od virtualnih mašina, baza podataka, mreža, backup-a i slično. „Azure podržava tri glavne usluge računarstva u oblaku a to su: infrastruktura kao usluga (eng. Infrastructure as a Service, IaaS), softver kao uslugu (engl. Software as a service, SaaS) i platformu kao uslugu (engl. Platform as a Service, PaaS)“[18]. Dakle, Azure Virtual Machines predstavljaju Iaas uslugu, jer nam virtualne mašine omogućuju izradu infrastrukture, odnosno radne okoline.

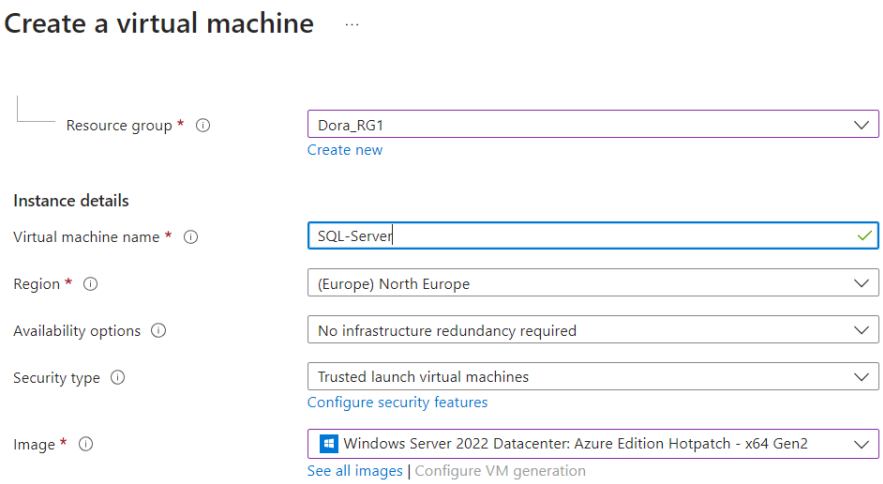
Cloud postaje sve popularniji u svijetu informacijske tehnologije jer omogućava skalabilnost, održava fleksibilnost, pruža sigurnost i predstavlja ekonomičniju opciju u usporedbi s on-premise infrastrukturom.

2.1.1. Kreiranje virtualnih mašina

U Azure-u je potrebno kreirati tri virtualne mašine iste vrste, odnosno *Windows Server 2022 Datacenter*. Razlika je to, što će se na dvije virtualne mašine nalaziti SQL server, a preostala će biti takozvani Domain Controller.
Na kreiranim SQL serverima će se nalaziti baza podataka s nekoliko tablica i podacima, ali to se neće odvijati preko Azure-a, već preko SQL Server Management Studia.
Prije kreiranja SQL servera, potrebno je kreirati takozvani Domain Controller. On služi da odobri ili ne odobri autentifikaciju korisnika koji žele pristup bilo kojem uređaju koji je spojen na njegovu domenu. Domena je bitna zbog mrežne povezanosti i međusobne komunikacije poslužitelja.
Resursna grupa (eng. Resource group) se kreira na samom početku, te kako joj i samo ime kaže, u nju se spremaju svi resursi potrebni za sve servere, mreže i slično. Za kreiranje virtualne mašine potrebno je napraviti sljedeće korake:

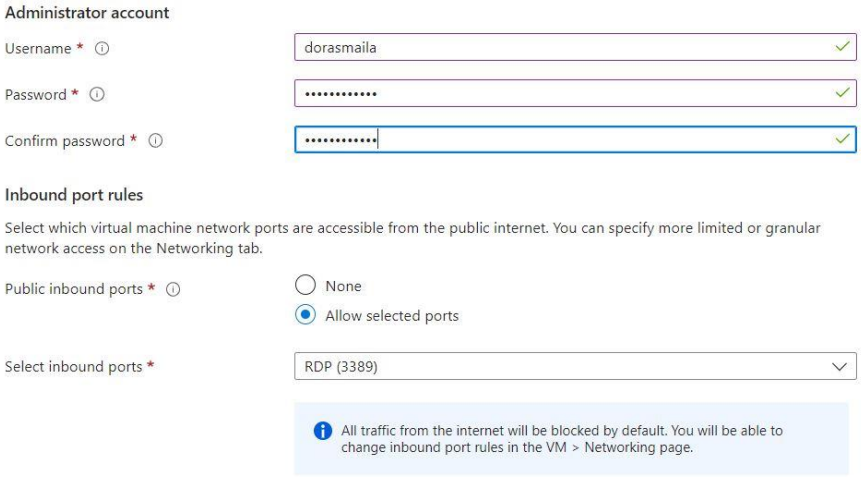
- Ime virtualne mašine
- U kojoj regiji se nalazimo
- Sliku (eng. Image)
- Administratora
- Na koji način se spajamo (RDP)
- Kakav disk koristimo
- Odabrati virtualnu mrežu

Na dolje prikazanoj slici (Slika 2) je prikazan prvi korak u kreiranju virtualne mašine, gdje je potrebno unijeti resursnu grupu, ime virtualne mašine, regiju i vrstu virtualne mašine.



Slika 2: Kreiranje virtualne mašine

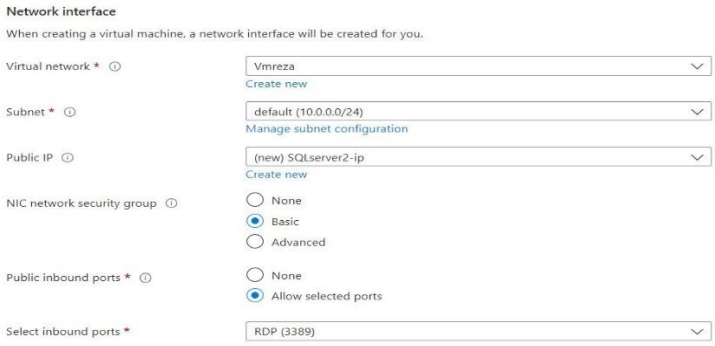
Drugi korak je definirati korisničko ime i lozinku (Slika 3.) za pristup virtualnim mašinama. Također, u ovom koraku se mora definirati vrsta spajanja na virtualnu mašinu, a to je putem RDP-a (eng. Remote Desktop Protocol).



Slika 3: Definiranje administratora

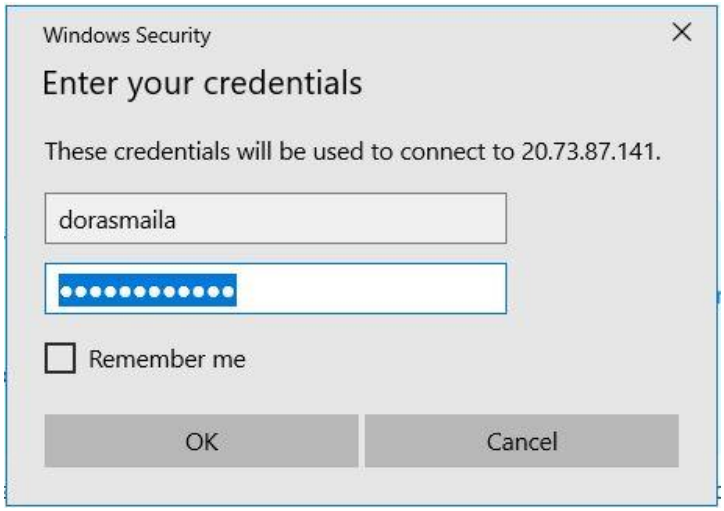
RDP je Microsoft-ova osigurana mrežna komunikacija koja omogućuje da korisnik daljinski pokrene svoje virtualne mašine na svom uređaju. Prilikom uspostave konekcije RDP dobije poruku, preko standardnog porta 3389, koja virtualna mašina želi uspostaviti konekciju.

Virtualnu mreža se također kreira preko azure-a, gdje se automatski dodijelile slobodne IP (eng. Internet Protocol) adrese potrebne za definiranje podmreže (eng. subnet) i javne IP adrese virtualne mašine (Slika 4).



Slika 4: Virtualna mreža

Za spajanje preko RDP-a preuzme se RDP datoteka s Azure-a na vlastito računalo i dvostrukim klikom na nju se otvara pop-up prozor prikazan dolje na slici (Slika 5.). Tu se upiše korisničko ime i lozinka, koji su se definirali prilikom izrade virtualne mašine i tako se možemo uspješno spojiti na stroj.



Slika 5: Pop-up prozor povezivanja RDP-om

Pri završetku instalacije virtualnih mašina, kreće instalacija SQL servera.

2.1.2. Instalacija SQL servera

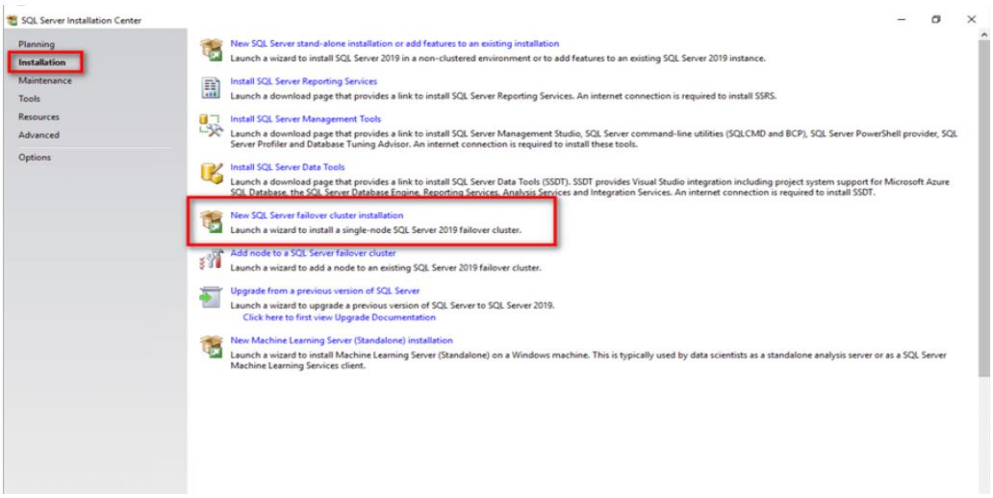
SQL Server podržava sljedeće tehnike visoke dostupnosti:

1. Windows Failover Cluster Instance (FCI)
2. Replikacija – Koristi se za očuvanje efikasnosti prijenosa ili distribucije baza podataka. Kopije podataka sprema na geografski različiti lokacijama kako bi se korisnicima diljem regije omogućio brži pristup podacima.[5]
3. Log Shipping – Log Shipping je SQL Servera značajka koja omogućuje slanje sigurnosnih kopija log transakcija (hrv. dnevnika transakcija) s izvornog servera na odredišne servere. Posao sigurnosne kopije izvršava se na bazi podataka na instanci primarnog poslužitelja, te kopira sigurnosnu kopiju u drugu , odnosno sekundarnu bazu podataka na instanci sekundarnog poslužitelja.[4]

4. Zrcaljenje baza podataka (eng. Database Mirroring) (sinkronizirano) – Stvara backup podataka glavnog servera. Dogodi li se pad glavnog servera, nastat će „zrcalna“ verzija tog istog na nekoliko različitih poslužitelja, te spajanjem na njih možemo doći do željene baze podataka .[5]
5. Always On Failover Cluster Instance
6. Always On Availability Groups

Kada su se mašine uspješno kreirale, na Domain Controlleru (DC01) je potrebno definirati domenu, a na druge dvije se mora instalirati SQL. Bitno je naglasiti kako se serveri moraju pridružiti toj domeni kako bi cluster točno znao s kojim serverima mora komunicirati. Domenu sam nazvala zavrsni.local, pa nakon pridruživanja u domenu, serveri dobivaju taj nastavak, SQL-server-1.zavrsni.local i SQL-server-2.zavrsni.local.

Nadalje, kreće postupak instalacije SQL-a unutar SQL-server-1 virtualne mašine. Preuzela sam datoteku za instalaciju SQL servera, te pokrenula *setup.exe*. Pokretanjem te datoteke, dobila sam sljedeći izbornik (Slika 6.):

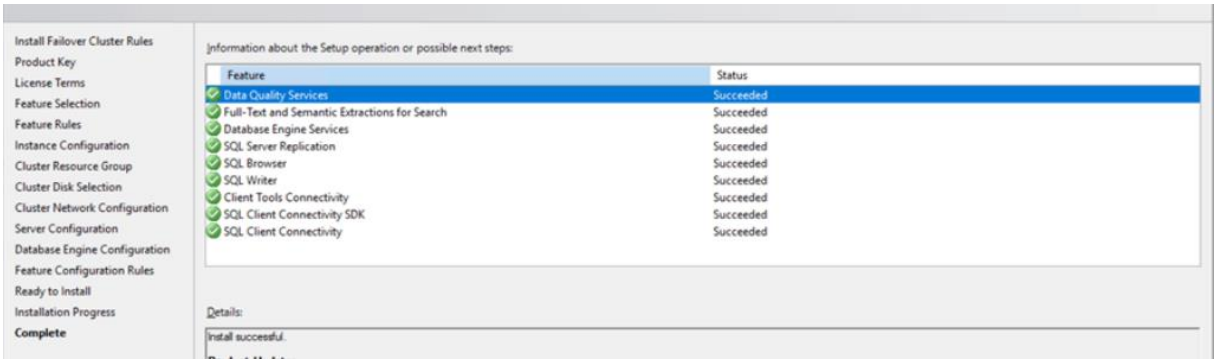


Slika 6: Instalacija SQL Failover Clustera

Cilj je SQL-server-1 i SQL-server-2 međusobno spojiti u cluster kako bi mogli biti u sinkronizaciji. Iz tog razloga, sam na prvi server instalirala *New SQL Server failover cluster installation.* U instalaciju ulaze sljedeće stavke:

- Cluster instalacija
- Zajednički network
- Dijeljena pohrana
- Korisnička dopuštenja
- Single node cluster

Potrebno je istaknuti zajedničku mrežu i dijeljenu pohranu. Zajednička mreža je bitna zbog komuniciranja servera u istoj mreži, odnosno domeni, inače komunikacija ne bi bila ostvarena. Dijeljena pohrana (eng. Shared storage) znači da sve što postoji na jednom serveru, replicira se na drugi.



Slika 7: Instalacija uspješna

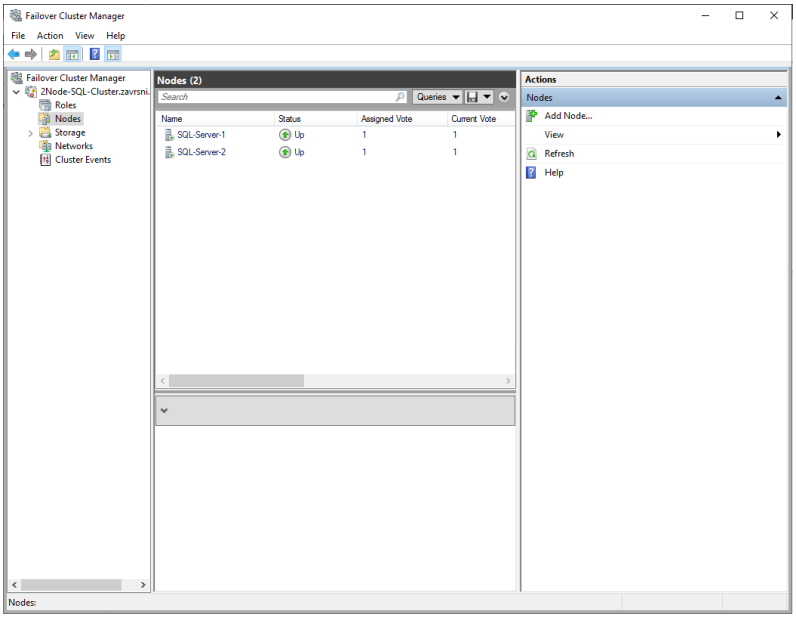
Nakon što se to uspješno podesilo (Slika 7), potrebno je drugi SQL server dodati u taj cluster. Instalacija se pokreće naredbom u command prompt-u: setup/SkipRules=Cluster_VerifyForErrors/Action = AddNode .
Instalacijom SQL servera, instalirali su se i svi potrebni alati koji služe za upravljanje i uređivanje, te kreiranje i pristup bazama podataka koje se nalaze na konfiguriranom SQL serveru.

2.2. Povezivanje dvaju servera u cluster

Prije samog povezivanja servera u cluster, potrebno je instalirati Faliover clustering na oba servera. Nakon bilo koje nove instalacije na serverima, mora se izvršiti ponovno pokretanje cijele virtualne mašine, kako bi se te instance stvarno primijenile. Kada se to obavi, potrebno je pokrenuti Failover cluster manager, koji služi za kreiranje cluster-a.

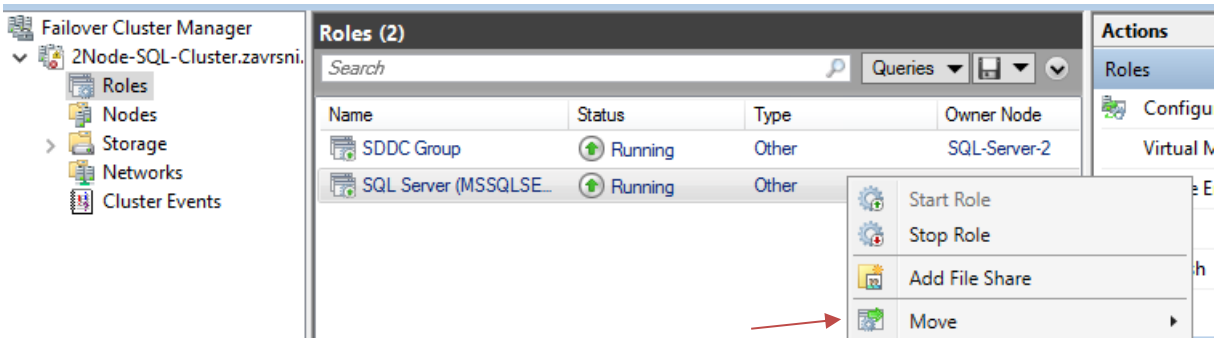
2.2.1. Failover cluster

Failover cluster je skup pojedinačnih fizičkih servera ili čvorova organiziranih u grupu koja može dijeliti računalno opterećenje aplikacije. Klasteriranje omogućuje da više poslužitelja radi kao jedan poslužitelj. Također, prednosti cluster-a su skalabilnost i pouzdanost. Ukoliko bi se preopteretili čvorovi u clusteru, skalabilnost omogućuje dodavanje još čvorova kako bi se radno opterećenje rasteretilo. „Neki se klasteri implementiraju sa softverom za nadilaženje greške koji može preraspodijeliti radno opterećenje jednog poslužitelja na drugi kada poslužitelj zakaže.“[17] Spomenuta pouzdanost smanjuje vrijeme zastoja i poboljšava dostupnost.
SQL Server Failover Clusters se ujedno naziva i SQL Server Failover Instance (skraćeno FCI) i koriste ih Windows Server. U bilo kojem trenutku, SQL Failover instanca će držati jedan čvor (ili server) upaljen koji će spajati zajedničku pohranu i Quorum pohranu kako bi SQL server instanca bila dostupna korisnicima. Quorum pohrana sadrži takozvani „quorum“, odnosno konfiguracijsku bazu podataka cluster-a. Ona obavještava cluster koji server je aktivan u kojem trenutku. Quorum je u principu dodatan disk u cluster-u koji može zapisivati i pratiti promjene i failover-e servera.
Čvorovi ili serveri koriste zajedničku pohranu (eng. Storage). U pohrani se nalaze sve podatkovne i log-ovne datoteke svih baza podataka, a s njima će se nalaziti i sve FCI konfiguracije koje se nalaze u Quorum-u [3]. Kako sam ranije napomenula, ovo je najbitniji dio za dobivanje visoke dostupnosti, jer treba omogućiti da se prilikom gašenja jednog servera svi podaci prekopiraju na drugi. Iz tog razloga, kako bi to bilo uspješno, treba konstantno testirati sustav kako bi čvorovi (serveri) uvijek bili dostupni. Upravo za taj dio je koristan alat Failover cluster manager. On prikazuje sve cluster-e koje smo kreirali, koji čvorovi su tamo prisutni, jesu li upaljeni ili ugašeni i slično. Trenutno, postoji samo jedan cluster 2Node-SQL-Cluster i sadrži dva čvora, SQL-Server-1 i SQL-Server-2 (Slika 8).

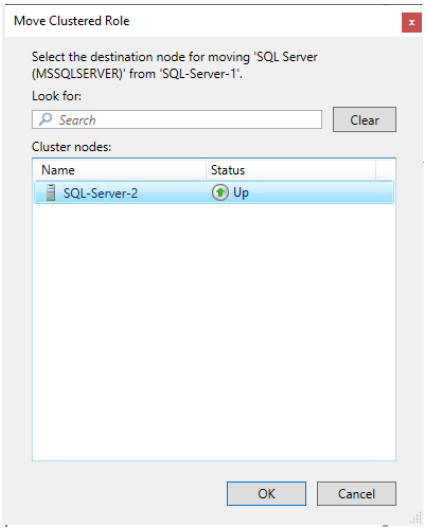


Slika 8: Failover Cluster Manager

Failover cluster manager nam omogućuje prikaz samih čvorova i daje nam do znanja na kojem od njih se infrastruktura odvija. Takav čvor je „glavni“, odnosno na njemu se vrti SQL servis. Spomenuti servis je definiran kao rola, odnosno uloga u cluster-u. Naime, to se može prebaciti na drugi čvor opcijom *Move* i odabirom drugog čvora (Slika 9.). Tim postupkom se, u teoriji, isključio prvi čvor ili server (Slika 10.).



Slika 9:Prebacivanje uloge na drugi čvor



Slika 10:Prebacivanje uloge na drugi čvor - 2

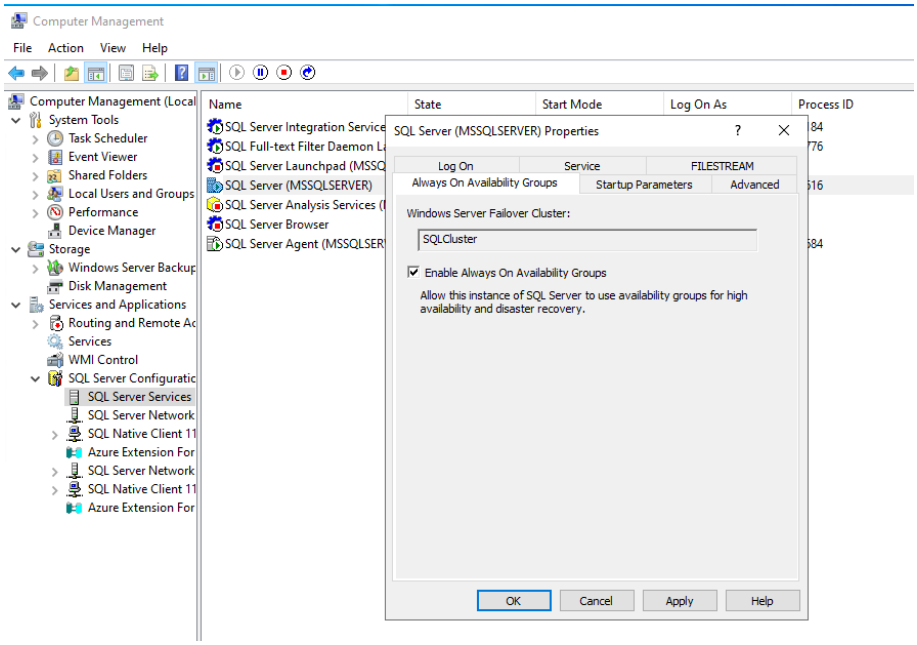
Klikom na gumb „OK“, manualno se inicira *failover* nad primarnim serverom. Može se primijetiti kako je sada SQL-server-2 u stanju „Up“ i sada se cijeli sustav odvija na sekundarnom serveru.

Potrebno je naglasiti i neke druge čimbenike koji dovode do problema s nedostupnošću. Najveći problem, već spomenut, je gubitak podatkovnog centra. Bez podatkovnog centra, naravno nemamo niti podatke baze podataka niti bilo koje druge podatke. Probleme može prouzročiti i loša povezanost s mrežom. Loša povezanost ne samo da može pogoršati performansu sustava, već i izbrisati bazu podataka. To se može dogoditi pri instaliranju nove verzije mrežnog softvera ili navođenjem netočne mrežne adrese tijekom konfiguracije. Ipak, za to također može biti odgovorni i hardverski dijelovi: prestanak rada procesora, manjak memorije i pad diskova. Naposljetku, probleme s dostupnošću može potaknuti i ljudska pogreška, koja je ipak najčešća pojava. Unos krivog podataka, gubitak objekata baze podataka, loš dizajn baze podataka, programske greške, pogreške pri samom administriranju baze i slično, sve to može biti uzrok loše dostupnosti. [17]

2.3. Always On Availability group

Postoji još jedan način za postizanje visoke dostupnosti, a to je putem kreiranjem availability grupa. Ukoliko bi se napravila dva servera, kao dva zasebna SQL servera i grupiralo ih u cluster, potrebno ih je povezati u grupu.

Availability grupa je skup baza podataka, odnosno **availability** baza podataka. To znači da se za svaku bazu u grupi stvara jedna read-write kopija primarne baze i jedna do osam read-only kopija, odnosno sekundarnih baza. [6]
Kako bi se uopće mogao nastaviti rad u SSMS-u, ovo je nužan korak prikazan na slici (Slika 14.).



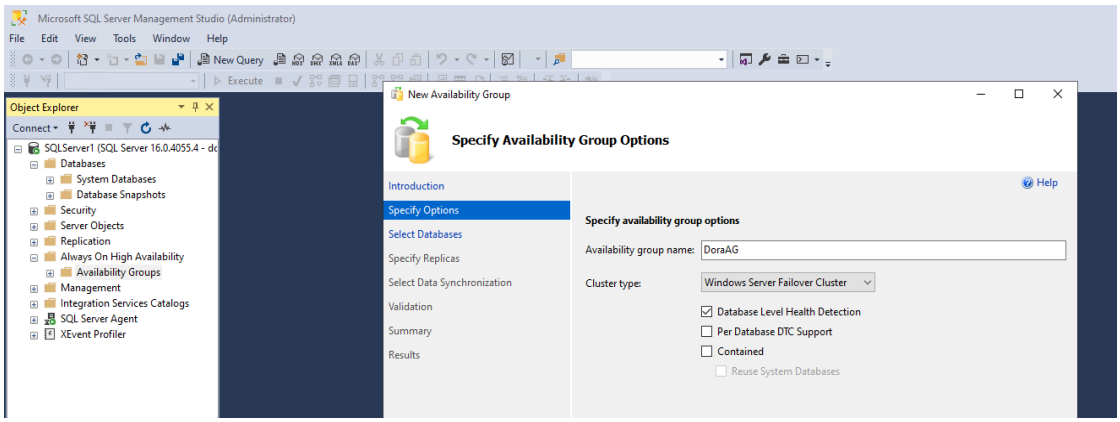
Slika 11:Uključivanje Always ON

Ovime se SQL servisu, točnije MSSQLSERVER-u, omogućuje rad s Always On Availability Group postavkom. Ukoliko se ova opcija ne uključi, serveri ne bi razumjeli što se od njih traži i ne bi dozvoljavali daljnje kreiranje availability grupa. Postavka bi se primijenila na servis nakon ponovnog paljenja samo servisa, ne cijele virtualne mašine.

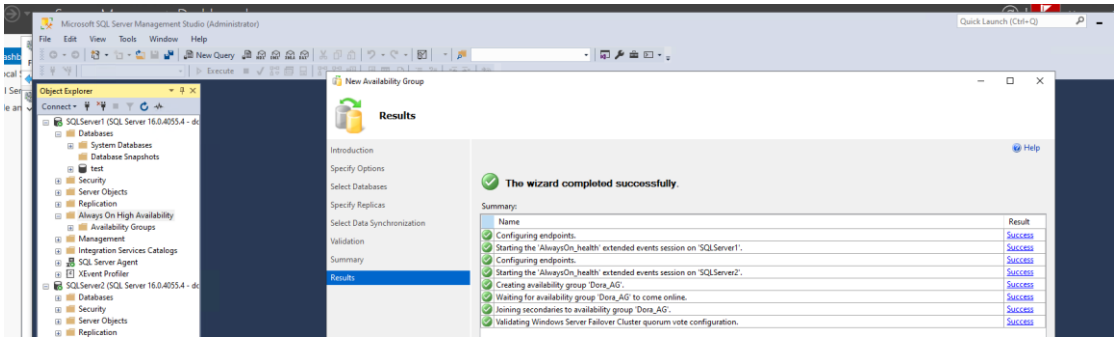
Always On Availability group je način dostupnosti ili svojstvo replike koje određuje može li repliciranje dostupnosti raditi u sinkroniziranom načinu rada servera. Drugim riječima, određuje mogu li se podaci sa primarnog servera automatski preslikati ili prekopirati na sekundarni server. Asinkroni način se koristi kada su availability replike raspoređene na većoj geografskoj udaljenosti.[6]

Kako bi se postiglo da grupa na točno mjesto prekopira podatke, potrebno bi bilo definirati SQL slušatelja (eng. listener). On je ujedno i bitan za spajanje na bazu podataka ili availability grupu. Morala bih mu definirati na koju se IP adresu mora spojiti kako bi uopće prepoznao sekundarni server. SQL slušatelj pomaže da se izbjegnu bilo kakve izmjene u nizu veza, u slučaju slanja baze podataka.

Na slikama (Slika 15 i Slika 16) je prikazan postupak kojim se Availability grupa kreira . Na samom početku je potrebno prvo napraviti bazu podataka, te omogućiti njen potpuni backup.



Slika 12: GUI za kreiranje grupe



Slika 13:Uspješno kreirana grupa

Ukoliko bi sve prošlo po planu, na primarnom i sekundarnom serveru bi trebala biti vidljiva baza podataka. Time sam se postiglo repliciranje podataka, baza podataka koja se nalazi na primarnom serveru, postoji i na drugom.

Prednost Avalability grupa je to što omogućuje HA na dvije geološke lokacije. Na primjer, ako server u Hrvatskoj prestane raditi, svi podaci će se replicirati u Njemačkoj ili gdje god se odredi druga lokacija. Ipak, za razliku od failover clustering-a, ovim postupkom je veća šansa da će se neki podatak izgubiti, jer ovakvim načinom nisam nigdje definirala zajedničku pohranu za servere, već se prijenos podataka odvija između podmreža. Replikacija podataka se isključivo odvija unutar availability grupe, odnosno radi isključivo sa samom bazom podataka, a ne na razini cijele SQL instance. Također, dolazi uz skuplju cijenu zbog korištenja Microsoft SQL Server Enterprise-a i zahtjeva da kupac ima jedan ili više sekundarnih servera na mreži i u stanju mirovanja, čekajući da se dogodi kvar.

2.4. Oporavak od katastrofe

Dogodi li se da iz bilo kojeg razloga, visoka dostupnost ne izvrši svoju ulogu, kreće oporavak od katastrofe. „Oporavak od katastrofe (engl. Disaster Recovery, DR) područje je sigurnosnog planiranja koje želi zaštititi organizaciju od učinaka značajnih negativnih događaja“[18]. Katastrofu u informatičkom svijetu mogu izazvati prirodne i neprirodne pojave, odnosno ljudska pogreška. Sve to zajedno, naziva se kritični događaj.

Jako je bitno da kritični događaji ne unište infrastrukturu, pa je iz tog razloga važno da postoje replike tog istog sustava na drugoj lokaciji. Ukoliko sve prođe po planu i uspostavi se normalno funkcioniranje sustava, možemo govoriti o postojanju oporavka od katastrofe.[18]

Za uspostavu oporavka od katastrofe, potrebna su dva faktora: dozvoljeni gubitak podataka (engl. Recovery Point Objective, RPO) i faktor oporavka sustava (engl. Recovery Time Objective, RTO). RPO predstavlja faktor maksimalne starosti datoteka koje je nužno oporaviti iz sigurnosne kopije kako bi sustav nastavio normalno funkcionirati nakon katastrofe. To znači, ako je RTO utvrđen na 6 sati, sustav se mora sigurnosno kopirati najmanje svakih 6 sati. RTO definira maksimalno vrijeme, nakon katastrofe, potrebno da organizacija oporavi sustav to jest vrati podatke iz sigurnosnih kopija i nastavi normalno funkcioniranje. Objašnjenje, ako je RTO utvrđen na 4 sata, sustav se ne smije sigurnosno kopirati duže od maksimalno 4 sata.

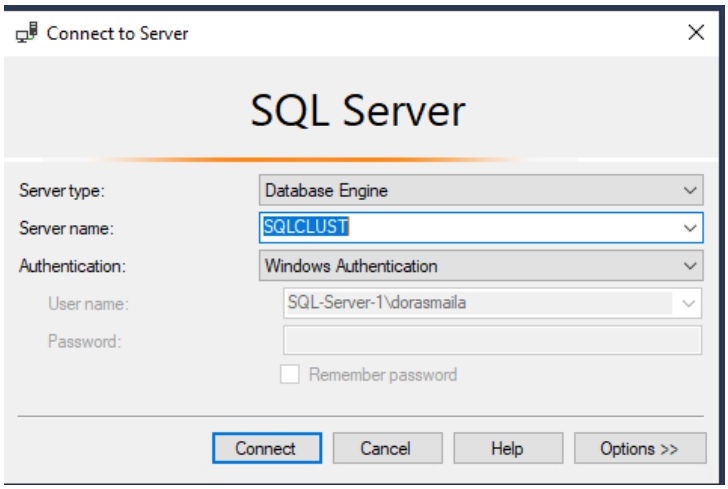
U planiranju oporavka od katastrofe, potrebno je razumjeti nekoliko koraka. Treba znati o infrastrukturi na kojoj radimo i koji su mogući rizici, analiza utjecaja katastrofe na sustav, planirati oporavak pomoću RPO i RTO faktora, te implementirati i testirati okruženje za oporavak od katastrofe.

Visoka dostupnost i oporavak od katastrofe imaju istu svrhu, ali postoji razlika. Visoka dostupnost pokušava održati sustav dostupnim cijelo vrijeme, a oporavak od katastrofe rješava probleme kada visoka dostupnost prestane ispunjavati svoju svrhu. Iz tog razloga je jako važno, da sve organizacije imaju plan oporavka bez obzira na postotak vremena dostupnosti. [19] Kako ne bi uopće bilo potrebno provoditi oporavak od katastrofe, jednako je važno obavljati rutinsko održavanje dok je sustav u funkciji, obavljati sigurnosna kopiranja, te koristiti kvalitetnu hardversku tehnologiju.

2.5. Testiranje visoke dostupnosti

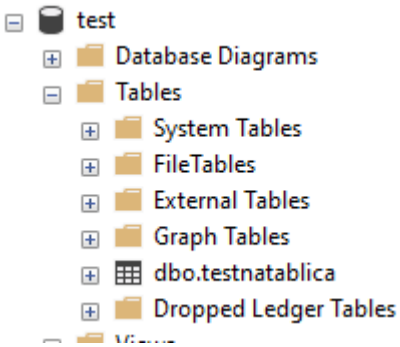
Glavni cilj visoke dostupnosti je osigurati dostupnost servisa tako da se što više smanji zastoj od očekivanih ili neočekivanih aktivnosti. Zbog toga su skoro sve tvrtke koje koriste visoku dostupnost postavile svoje infrastrukture na jednu geološku lokaciju s opcijom da se u slučaju problema sve prebaci na drugu geološku lokaciju.

Nakon što se sve uspješno kreira, pokrene se alat SSMS, odnosno SQL Server Managment Studio s kojim se ostvaruje spajanje na server koji se smatra primarnim. U mom slučaju, taj server je SQL-server-1, te se na njega se treba spojiti kroz Windows autentifikaciju (Slika 11).

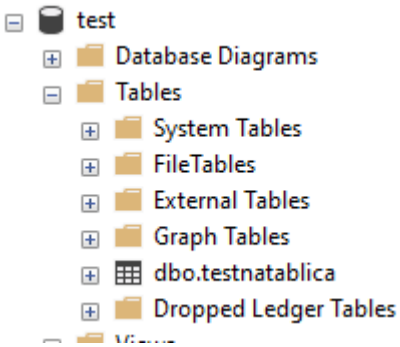


Slika 14:Povezivanje na cluster

Prvi korak je kreirati bazu podataka. Za test, kreirala sam jednu tablicu nazvala ju *testnatablica*. Zanima me što će se dogoditi ako cluster prebaci na drugi čvor. U Failover cluster manager-u se ponove koraci za prebacivanje čvorova, te se na drugom serveru otvori SSMS, uspostavi se konekciju na cluster i može se uočiti ista bazu i ista tablicu *testnatablica*.



Slika 15:Usporedba Servera 1



Slika 16: Usporedba Server 2

Ovime se dokazalo da je visoka dostupnost SQL servera postignuta i niti jedan podatak nije izgubljen. U nadolazećim poglavljima, testirat ću kako promjene u bazi ili nad bazom podataka utječu na visoku dostupnost.

2.6. Upravljanje bazom podataka

Upravljane bazom podataka uglavnom upućuje na izvršavanje promjena unutar baze ili promjene u fizičkom okruženju, mrežnoj infrastrukturi i slično. Razlozi za poduzimanjem promjena mogu biti: promjene aplikacijskih programa, poboljšanje performansi, uvod novih vrsta podataka i tehnološke promjene.

Postoje dvije vrste promjena: proaktivna promjena i inteligencija. Proaktivna promjena je najvažnija vrsta promjene, jer može otkloniti buduće probleme ranijim detektiranjem promjena i implementiranjem rješenja. Inteligencija tijekom upravljanja promjenama zahtjeva detaljnu analizu rizika i plana provedbe, te plan za nepredvidljive situacije[17].

2.6.1. Relacijska baza podataka

Relacijska baza podataka je takva baza u kojoj su podaci podijeljeni unutar tablica. Svaka tablica predstavlja zaseban entitet, kao što su kupci (jedna tablica) ili proizvodi (druga tablica). Podaci se pohranjuju u recima, a u stupcima se definira ime tog atributa, npr. ime, prezime i slično. Svaka relacijska tablica mora imati primarni ključ, jedinstvenu vrijednost po kojoj se tablice razlikuju. Tablice mogu sadržavati više jedinstvenih vrijednosti, no razliku između općenito „unique“ vrijednosti i primarnog ključa ću objasniti u nadolazećim poglavljima.

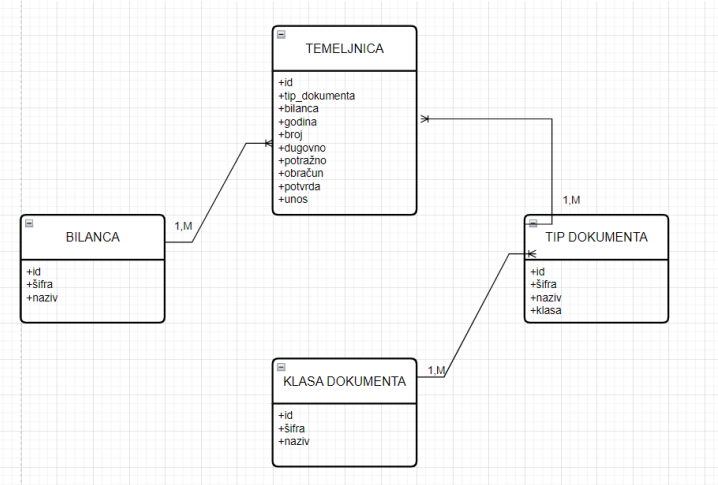
Relacijski model je najbolji u održavanju dosljednosti podataka u aplikacijama i kopijama baze podataka (koje se nazivaju instance). Relacijske baze podataka ističu se u ovoj vrsti dosljednosti podataka, osiguravajući da višestruke instance baze podataka imaju iste podatke cijelo vrijeme. „Drugim je vrstama baza podataka teško održati ovu razinu pravovremene dosljednosti s velikim količinama podataka. Neke novije baze podataka, kao što je NoSQL, mogu osigurati samo "eventualnu dosljednost". Prema ovom načelu, kada se baza podataka skalira ili kada više korisnika pristupa istim podacima u isto vrijeme, podacima treba neko vrijeme da "sustignu"“ [9].

Četiri ključna svojstva su potrebna kako bi se definirale transakcije relacijske baze podataka[10]:

1. Atomičnost (eng. atomicity) - Svojstvo koje tretira transakcije kao jednu cjelinu. Ovo svojstvo je poprilično bitno za očuvanje i sigurnost same baze podataka. Ukoliko bi se dogodila slučajna neželjena situacija (nestanak struje ili slično), moguće su dvije situacije. Transakcija se uspješno izvršila usprkos nastalom problemu ili bi se poništila (eng. rollback) čim jedan dio transakcije ne bi uspio.
2. Dosljednost (eng. consistency) - Omogućuje da promjene koje su učinjene prilikom transakcije dosljedne s ograničenjima baze podataka. U suprotnom, cijela transakcija ne bi uspjela.
3. Izolacija (eng. isolation) – Uloga izolacije je da nitko od drugih korisnika baze podataka ne vidi učinjenu transakciju, dok se transakcija ne izvrši.
4. Trajnost (eng. durability) – Osigurava da su promjene u bazi vidljive, nakon što se transakcija izvrši.

Za dobru relacijsku bazu podataka važno je da se podaci ne dupliciraju, kako bismo izbjegli suvišne informacije. Na kraju, najvažnije je da su podaci ispravni i točni.

Kroz četiri tablice, prikazat ću najučestalije funkcije i naredbe koje se koriste za filtriranje i ispis podataka, od najobičnijeg ispisa, do složenijih upita, te spajanja i pogleda tablica. Tablice na kojima sam radila su bilance, temeljnice, tipovi dokumenata, klasa dokumenata. Sve navedene tablice imaju barem jedan vanjski ključ, odnosno primarni ključ drugih tablica. Temeljnice imaju primarni ključ od bilanca i tipova dokumenata, tip dokumenata ima od klase dokumenata. Dijagram entiteta (Slika 17) nam ukazuje da jedna bilanca može imati više temeljnica, jedna klasa dokumenta može imati više tipova dokumenata i na kraju, jedan tip dokumenta može imati više temeljnica.[1]



Slika 17: Model entiteta

Za sam postupak upravljanja bazom podataka, potrebno je razumjeti poslovni rječnik i znati ga prevesti u logički model podataka. Inače, za ovaj dio posla je zaslužan DA, odnosno administrator podataka (eng. data administrator). Administrator također treba osigurati da su svi podaci pravilno dokumentirani i voditi računa o planiranju korištenja podataka. Nadalje, nakon što se izradi logički model podataka, administrator baze podataka (DBA) može početi razmišljati o strukturi fizičke baze podataka. DBA ima više zadataka: projektiranje, praćenje performansi, osiguravanje dostupnosti, sigurnost baze, sigurnosno kopiranje i očuvanje podataka.[17] Što se tiče osiguravanje dostupnosti, DBA mora osigurati da je baza podataka što manje izvan mreže i mora osigurati da svaki servis ima određenu razinu dostupnosti, ali najsigurnija je upravo visoka dostupnost [17]. Dostupnost je za administratora baze podataka najbitnija stvar koja uvijek mora biti prisutna i funkcionalna. Dogodi li se da podaci nisu dostupni, to automatski znači da se neke aplikacije neće pokretati koje ovise o toj bazi podataka. Naravno, ni korisnici ne žele da im servisi ne rade ili da rade lošom performansom, znači postoje u samom početku visoka očekivanja od dostupnosti. S druge strane, postoje situacije u kojima DBA namjerno mora učiniti bazu podataka nedostupnom[17]. Takve situacije su uglavnom izmjena strukturne definicije u bazi ili bilo kakva vrsta promjene nad njom.

U današnje vrijeme, sve više je potreba za visokom dostupnošću tražena. „Razlog tome je rastuća potreba za internetskom i web podrškom u sustavima upravljanja bazom podataka i aplikacijama.“[17] S time, potrebno je pažljivo dizajnirati baze podataka da dobro funkcioniraju sa tehnologijom koja omogućuje HA, a DBA mora pregledati koji podaci, procedure i slično, se mogu redizajnirati bez potrebe za privremenim isključivanjem baze podataka.

2.6.2. Osnove SQL-a

SQL je kratica za Structured Query Language i koristi se manipulaciju i pristup bazama podataka. Njime možemo:

- Dohvaćati podatke
- Brisati ih
- Modificirati
- Spajati druge tablice
- Kreirati nove baze podataka i tablice
- Kreirati procedure i poglede,… [8]

SQL jezik pripada skupini deklarativnih programskih jezika, jer njime opisujemo što želimo da se izvrši, ali ne programiramo kako želimo da se radnje izvrše. Sve naredbe i funkcije koje nam SQL nudi su engleske riječi koje nam doslovno „govore“ koja je njihova uloga. Iz tog razloga, SQL-ov kod je čitljiv i točno znamo što ćemo dobiti kao rezultat, ali sama logika na koji način SQL nama ispiše što želimo nam nije bitna.

U ovom poglavlju, objasnit ću kako se koriste ove mogućnosti, sintaksu pisanja SQL- a, te ih potkrijepiti primjerima koje sam napravila na serveru. U isto vrijeme, pratit ću kako te promjene u bazi i nad bazom podataka utječe na visoku dostupnost, odnosno izvršava li HA svoju ulogu.

Sve sam kreirala i radila u SQL Server Manager Studio-u (SSMS) koji služi i za sve druge svrhe baza podataka. Ovim alatom se kreiraju baze, tablice i upravlja SQL-ova infrastruktura. „SSMS je sveobuhvatni uslužni program koji kombinira široku grupu grafičkih alata s mnogo uređivača skripti kako bi omogućio pristup SQL Serveru za programere i administratore baza podataka svih razina vještina“ [7].

Naredbom CREATE, kako joj i sam naziv kaže, kreira neko novo svojstvo. Može se kreirati nova baza podataka (create database) ili tablica, uloga i slično. Bazu podataka sam kreirala prilikom spajanja na cluster. Postoje dva načina kojim se može kreirati, a to su desnim klikom na server i *Create database* ili u SQL Worksheetu naredbom CREATE DATABASE *ime_db*;
Kako bi se napravila nova tablica, potrebni su joj neki atributi. Sintaksa za ovu naredbu je : CREATE TABLE *ime_tablice (naziv_atributa tip_podatka)*; . Sve naredbe u SQL sintaksi završavaju sa točkom zarez ';'. Također, bitno je za napomenuti da SQL nije case sensitive,

odnosno nije mu bitno pišemo li velikim ili malim slovima. Na slici niže je prikazan kod koji sam koristila za kreiranje moje tablice DS_BILANCE s njenim atributima(Slika 18). [11]

```
CREATE TABLE DS_BILANCE
    (ID int,
     SIFRA VARCHAR(6),
     NAZIV VARCHAR(35)    ) ;
```

Slika 18: CREATE

ALTER TABLE koristim ako se želi promijeniti neko svojstvo atributa unutar tablice koje sam kreirala. Na primjer, treba promijeniti da neko polje ne smije biti bez vrijednosti ili promijeniti ime stupca. Uglavnom, pri bilo kakvoj promjeni, ovo je naredba koja je potrebna. U mom slučaju, htjela sam promijeniti da mi ID, NAZIV I SIFRA tablice DS_BILANCE ne smije biti bez vrijednosti, odnosno NOT NULL. Isto tako, pri definiranju primarnog ključa tablice BILA_PK. Sintaksa za ALTER naredbu je : ALTER TABLE *ime_tablice* ALTER COLUMN *naziv_stupca ime_atributa, tip_atributa, promjena*;. Na slici (Slika 19) je prikaz postavljanja vrijednosti na NOT NULL (atribut mora imati vrijednost) i postavljanje primarnog ključa.

```
ALTER TABLE DS_BILANCE ALTER COLUMN ID INT NOT NULL;
CREATE UNIQUE INDEX BILA_PK ON DS_BILANCE (ID);

ALTER TABLE DS_BILANCE ALTER COLUMN SIFRA VARCHAR(6) NOT NULL;
ALTER TABLE DS_BILANCE ALTER COLUMN NAZIV VARCHAR(35) NOT NULL;
ALTER TABLE DS_BILANCE ADD CONSTRAINT BILA_PK PRIMARY KEY (ID);
```

Slika 19:ALTER

S druge strane, ova funkcija ima svoja ograničenja. Na primjer, ne može se promijeniti sam naziv objekta ili tablice, premjestiti podatke iz jedne u drugu tablicu, ukloniti ili dodati stupce, promijeniti definiciju primarnog i vanjskog ključa i slično.[17] „Promjena podataka u bazi koristeći ovu naredbu nije toliko značajna za korisnike da bi administrator trebao onemogućiti pristup bazi, jer se promjene koje se naprave ALTER naredbom učine brzo.“[17]

Modify u sql query-ju u SSMS-u izgleda malo drugačije nego na primjer u Oracle-u. Ne možemo koristiti MODIFY , već ALTER COLUMN. [12]

Indexi nam služe da brže izvučemo neki podatak iz tablice, nego što bismo to napravili bez njih. Korisnici ne mogu vidjeti index, niti znati koji atribut je index, oni samo služe ljudima koji rade direktno u bazi podataka za ubrzano pretraživanje.

Unique se upotrebljava kada je potrebno naglasiti da neka vrijednost u tablici uvijek mora imati drugačiji iznos. Ukoliko želimo naglasiti da je nešto UNIQUE, samo ga dopišemo (Slika 20).

```
CREATE UNIQUE INDEX BILA_PK ON DS_BILANCE (ID);
```

Slika 20:UNIQUE i INDEKS

2.6.3. Ograničenja

Ukoliko je potrebno u našoj bazi podataka, odnosno tablici, definirati ograničenje koristimo naredbu ADD CONSTRAINT. Ograničenja se uglavnom odnose na primarne i vanjske ključeve, indexe i jedinstvena polja, ali i dodjeljivanje uloga i prava. To znači da korisnik ne smije upisivati bilo koju vrijednost na mjestima gdje je definirano ograničenje, već mora obratiti pozornost na to je li vrijednost jedinstvena, postoji li već negdje neki podatak iz druge tablice i slično.

S ulogama i pravima, korisnicima možemo dodijeliti određene radnje koje mi, kao administratori, smatramo da su nužne.

Primarni ključ (eng. Primary key) automatski identificira zapis kao jedinstven (za razliku od UNIQUE). Tablica može imati isključivo jedan primarni ključ. Pri kreiranju primarnog ključa, potrebne su mi naredbe ALTER TABLE i ADD CONSTRAINT (Slika 21.)

```
ALTER TABLE DS_BILANCE ADD CONSTRAINT BILA_PK PRIMARY KEY (ID);
```

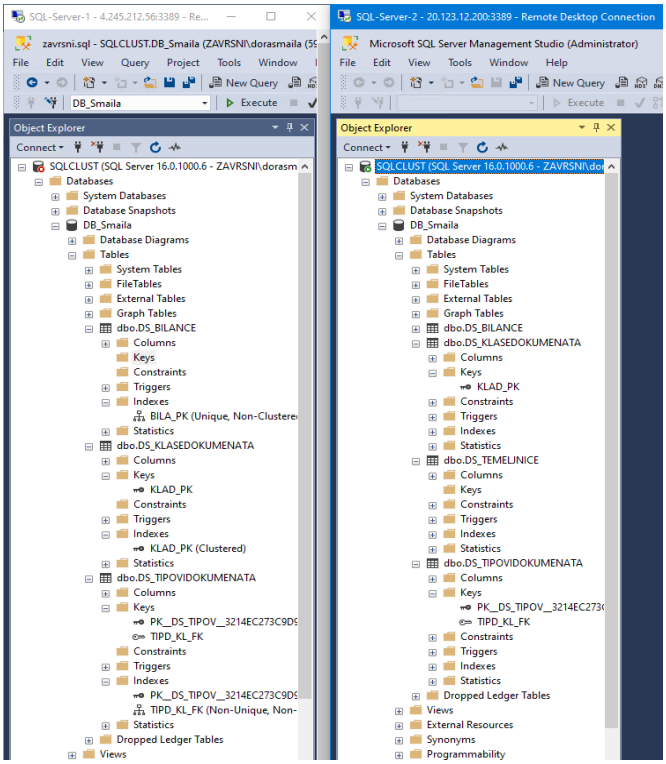
Slika 21: CONSTRAINT

Vanjski ključ (eng. Foreign key) je primarni ključ koji se javlja kao atribut u drugoj tablici. Služi za spajanja (eng. Join), odnosno ispis dvaju ili više tablica koje su povezane nekim stupcem, odnosno vanjskim ključem. Ovo je ograničenje, jer sprječava umetanje nevažećih podataka na mjesto gdje već postoji vrijednost iz druge tablice. Postupak za kreiranje ograničenja za vanjski ključ je slična kao i za primarni, jedino je potrebno naglasiti na koju klasu se odnosi (*reference*) navedeni vanjski ključ(Slika 22). [13]

```
ALTER TABLE DS_TIPOVIDOKUMENATA ADD CONSTRAINT TIPD_KL_FK FOREIGN KEY (KLASA) REFERENCES DS_KLASEDOKUMENATA (ID);
```

Slika 22: CONSTARINT - 2

Zanima me što će se dogoditi, ako cluster prebacim na drugi server. Spojim se na cluster sa drugog servera i pogledam što se sve nalazi u bazi podataka *DB_Smaila*. Usporedbom dvaju servera, mogu zaključiti da su se svi podaci koje sam do sad kreirala uspješno prekopirali na drugi server (Slika 23.).

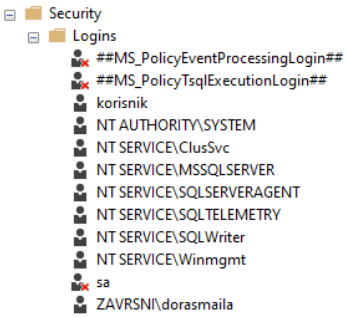


Slika 23:Usporedba servera

Osim same baze i njenih podataka, kopirala su se i ograničenja, odnosno postavljeni primarni i vanjski ključevi.

2.6.4. Prava i uloge

Sintaksa za kreiranje novih uloga i korisnika izgleda malo drugačije nego što je to slučaj u, na primjer Oracle bazi podataka. Ovdje, u SSMS-u, prvi korak je napraviti novi l*ogin* za prijavu na server. Navedena slika (Slika 24) prikazuje kako izgleda direktorij Logins.



Slika 24:Login folder u SSMS-u

Za novi login bilo je potrebno definirati početnu lozinku s kojom će se budući korisnik prijaviti. Za početnu lozinku je važno da se definira po pravilima sigurnosti, a to je da mora sadržavati znakove, mala i velika slova i slično. Sljedeći korak je definiranje novog

korisnika za novonastali login. Sintaksa je jednostavna: CREATE USER *novi_user* FOR LOGIN *login*;.

Pri kreiranju novih uloga ili rola, odmah mogu definirati kome želim da uloga pripada, a za to je zaslužna ugrađena funkcija AUTHORIZATION. Ukoliko želim da novokreirana uloga „role_ds“ pripada korisniku „korisnik“, nakon riječi AUTHORIZATION ću samo na dopisati „korisnik“ (Slika 25). [15]

```
USE DB_Smaila ;


CREATE LOGIN korisnik
WITH PASSWORD = 'Uvxs245!';

CREATE USER korisnik
FOR LOGIN korisnik;

CREATE ROLE role_ds AUTHORIZATION korisnik;

GRANT SELECT ON DS_TEMELJNICE TO korisnik;
GRANT SELECT ON DS_TEMELJNICE TO role_ds;
```

Slika 25: Nova uloga i korisnik

Funkcija GRANT (Slika 25) služi za dodjelu prava ili funkcija. Ukoliko ne želimo da novi korisnik ima sve uloge, već samo određenu kao što je SELECT, onda ulogu SELECT dodijelimo samo tom korisniku. Time sam postigla da korisnik može ili samo izvući podatke iz baze podataka i tablica, ali ne može ih sam uređivati. Druge funkcije se također mogu dodjeljivati, poput INSERT, UPDATE pa korisnik može umetati podatke i uređivati ih. S druge strane, prava i uloge se mogu i ukidati, a za to se koristi naredba REVOKE. Sintaksa je ista kao i za GRANT, ali ovako ukidamo korisniku ili sustavu uloge i prava nad bazom podataka.

2.6.5. Upiti

Upiti u SQL-u služe za izvlačenje podataka iz baze , izvlačenje s određenim uvjetima i za grupiranje podataka. Najosnovnija naredba za upit je SELECT koji određuje koji će se atribut prikazati i FROM, koji određuje iz koje tablice želimo izvući podatke navedene u select-u.

Za upite s uvjetom, trebamo koristiti WHERE naredbu, kojoj definiramo od kuda se naveden uvjet mora izvući. Sintaksa bi izgledala na sljedeći način: SELECT *atribut* FROM *tablica* WHERE *atribut* = *vrijednost*;.

U slučaju gdje treba ispis podataka iz više tablica, potrebno je primijeniti složeni upit. Dolazim do upotrebe ALIAS-a i spajanja tablica. ALIAS se upotrebljava kao zamjensko

ime nekog stupca ili tablice, pa se tako na ispisu ispiše drugo ime. Alias ne mijenja naziv stupca ili tablice u bazi, već samo pri ispisu. Općenita sintaksa za uporabu aliasa je sljedeća: SELECT *naziv_stupca* [AS] *novi_naziv* [ili "novi naziv"] FROM *naziv_tablice*;

Aliasi se koriste i u spajanjima tablica, radi lakšeg snalaženja i preglednosti samog upita, kao što je prikazano na slici (Slika 26). Spojila sam sve tablice koje sam kreirala koristeći *left outer join*. Spajanja se koriste kada želim ispisati sve podatke tablica koje su povezane vanjskim ključevima. [14]

```
SELECT A.ID, A.GODINA, A.BROJ, A.OBRACUN, B.SIFRA BILA_SIFRA, B.NAZIV BILA_NAZIV, C.SIFRA TIP_SIFRA, C.NAZIV TIP_NAZIV, D.SIFRA KLASA_SIFRA, D.NAZIV KLASA_NAZIV
FROM DS_TEMELJNICE A INNER JOIN DS_BILANCE B ON A.BILANCA = B.ID
LEFT OUTER JOIN DS_TIPOVIDOKUMENATA C ON A.TIPDOKUMENTA = C.ID
LEFT OUTER JOIN DS_KLASEDOKUMENATA D ON C.KLASA = D.ID
```

Slika 26: Left join

Za ispis podataka koristi se pogled (eng. view). Iznad koda za spajanje tablica potrebno je dopisati CREATE VIEW i stvori mi se pogled, odnosno rezultat koji sam htjela izvući iz baze (Slika 27 i Slika 28). [16]

```
USE [DB_Smaila]
GO

CREATE VIEW DS_V_TEMELJNICE AS
SELECT A.ID, A.GODINA, A.BROJ, A.OBRACUN, B.SIFRA BILA_SIFRA, B.NAZIV BILA_NAZIV, C.SIFRA TIP_SIFRA, C.NAZIV TIP_NAZIV, D.SIFRA KLASA_SIFRA, D.NAZIV KLASA_NAZIV
FROM DS_TEMELJNICE A INNER JOIN DS_BILANCE B ON A.BILANCA = B.ID
LEFT OUTER JOIN DS_TIPOVIDOKUMENATA C ON A.TIPDOKUMENTA = C.ID
LEFT OUTER JOIN DS_KLASEDOKUMENATA D ON C.KLASA = D.ID
```

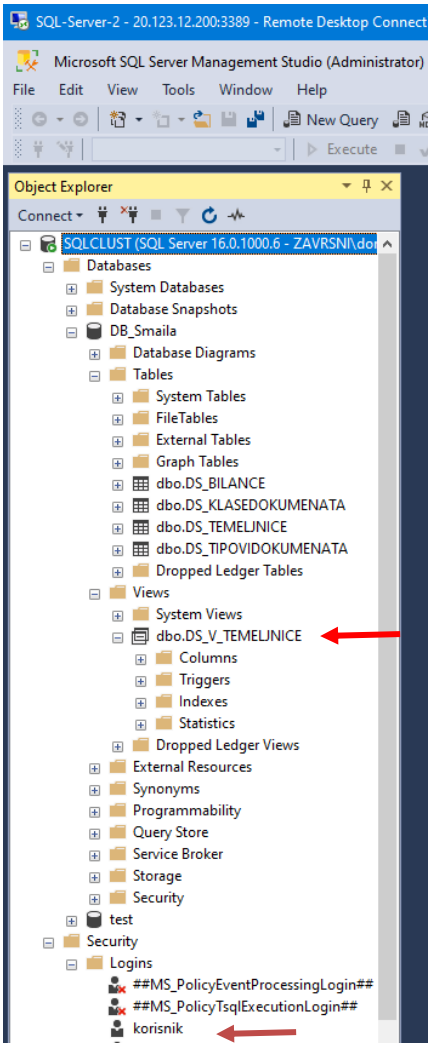
Slika 27: Create view

Results | Messages

	ID	GODINA	BROJ	OBRACUN	BILA_SIFRA	BILA_NAZIV	TIP_SIFRA	TIP_NAZIV	KLASA_SIFRA	KLASA_NAZIV
1	1	2023	1	2023-08-31	1	DS BILANCA	T	OPCA TEMELJNICA	GK	GLAVNA KNJIGA

Slika 28: Rezultat pogleda

Prebacim li sada cijeli sustav na sekundarni server i provjerim što se nalazi pod Views i Security/Logins, vidjet ću sljedeći prikaz (Slika 29).



Slika 29: Preslika na sekundarni server

Kreirani pogled i novokreirani korisnik su se kopirali na sekundarni server. Visoka dostupnost SQL servera je prošla testove i kod kopiranja podataka baze podataka i upravljanja bazom podataka.

3. Zaključak

U današnjem digitalnom svijetu, koji iz dana u dan sve više napreduje, smatram da potreba za visokom dostupnošću raste. Bez visoke dostupnosti SQL servera ne bi mogli imati neometan i pouzdan rad s bazama podataka. Ostvarivanje visoke dostupnosti mnogim tvrtkama i organizacijama zvuči nemoguće, iz financijskih razloga ili ne vjerovanja da je to izvedivo. Naime, ona nam omogućuje smanjenu neproduktivnost i gubljenje vremena na popravke, a i sami serveri rade brže. Sve to utječe i na zadovoljstvo korisnika koji i koriste usluge koje im pružamo. Ukoliko korisnik primijeti da je servis nije brz ili učinkovit koliko on to očekuje, vrlo brzo će krenuti tražiti druge usluge. Ova karakteristika korisnika da želi sve informacije dobiti „sada“ se ujedno zove i mentalitet „brze hrane“ [17]. Nadalje, većina organizacija želi imati najbolju opremu i nuditi najbolju uslugu, a time i druge organizacije imaju taj isti cilj. Takozvanu najbolju uslugu nudi upravo visoka dostupnost.

Mana visoke dostupnosti SQL servera je cijena clustera. Jedan server je skup, a za visoku dostupnost i cluster su nam potrebna dva. Također, u cijenu ulazi zajednička pohrana. S druge strane, ako kupac želi koristiti Availability grupe umjesto Failover clustering-a, tada će uštedjeti, ali će imati kompliciraniju konfiguraciju. Failover clustering i Availability grupe, kao metode postizanja visoke dostupnosti, imaju svoje prednosti i mane. Ipak, ideja im je u principu ista, a to je omogućavanje neometanog rad nad podacima bez ikakvih gubitaka, repliciranje ne samo ono s čime se konkretno u tom trenutku bavimo, već bilo kojih dokumenata i datoteka, što je i ujedno dodatan backup podataka.

Osim financijskog aspekta, potrebno je imati opsežno znanje o mrežnoj infrastrukturi, virtualizaciji i razumijevanje o svim konceptima i pojmovima vezanih uz visoku dostupnost. Treba biti oprezan i sa samom konfiguracijom dostupnosti koja je komplicirana, a pritom zahtjeva konstantno održavanje. Dogodi li se jedna greška administratora, cijeli sustav može stati, što može biti poprilično stresno.

Za kraj, korisno je imati implementiranu visoku dostupnost SQL servera, ali isto tako i drugih infrastruktura, jer u slučaju ljudske pogreške ili prirodnih katastrofa, možemo biti mirni da su nam podaci sigurno replicirani na backup serveru.

Literatura

[1] Database Administration: The Complete Guide to Practices and Procedures By Craig S. Mullins. Objavljena: 14.07.2002. https://ai12labs.files.wordpress.com/2013/04/dba-the-complete-guide-to-practices-and-procedures.pdf

[2] JSCAPE by Redwood: https://www.jscape.com/blog/active-active-vs-active-passive-high-availability-cluster (Posjećeno: 01.09.2023.)

[3] Techtarget: https://www.techtarget.com/whatis/definition/cluster-quorum-disk (Posjećeno: 01.09.2023.)

[4] Paper cut: https://www.papercut.com/discover/best-practices/high-availability-methods-for-microsoft-sql-server/#backups-downtime-and-the-importance-of-high-availability (Posjećeno: 01.09.2023.)

[5] Tutorials point: https://www.tutorialspoint.com/difference-between-mirroring-and-replication (Posjećeno: 01.09.2023.)

[6] Službena Microsoft dokumentacija: https://learn.microsoft.com/en-us/sql/database-engine/availability-groups/windows/always-on-availability-groups-sql-server?view=sql-server-ver16 (Posjećeno: 01.09.2023.)

[7] Službena Microsoft dokumentacija: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16 (Posjećeno: 01.09.2023.)

[8] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_intro.asp (Posjećeno: 01.09.2023.)

[9] Službena Oracle dokumentacija: https://www.oracle.com/database/what-is-a-relational-database/ (Posjećeno: 11.09.2023.)

[10] MongoDB: https://www.mongodb.com/basics/acid-transactions

(Posjećeno: 11.09.2023.)

[11] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_create_table.asp (Posjećeno: 30.08.2023.)

[12] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_alter.asp (Posjećeno: 30.08.2023.)

[13] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_constraints.asp (Posjećeno: 30.08.2023.)

[14] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_join.asp (Posjećeno: 30.08.2023.)

[15] W3Schools SQL Tutorial: https://www.sqlservertutorial.net/sql-server-administration/sql-server-create-role/ (Posjećeno: 30.08.2023.)

[16] W3Schools SQL Tutorial: https://www.w3schools.com/sql/sql_view.asp (Posjećeno: 30.08.2023.)

[17] Administriranje i sigurnost BP: Materijali s Merlina

[18] Visoka dostupnost serverske infrastrukture u virtualnom okruženju korištenjem Microsoft tehnologija, Ćavar Mateo: https://urn.nsk.hr/urn:nbn:hr:166:244635 (Posjećeno: 11.09.2023.)

[19] Atmosfera: https://www.atmosera.com/blog/high-availability-vs-disaster-recovery/ (Posjećeno: 12.09.2023.)

Popis slika