# Sadržaj

# Uvod

Baze podataka se aktivno koriste u sistemima raznih poslovnih namena. Postoji mnogo sistema/aplikacija koje zavise umnogo od dostupnosti baze podataka i tu u igru dolaze baze podataka sa visokom dostupnošću (eng. *High-Availability - HA*). HA sistemi baza podataka se koriste kada je potrebno da baza bude dostupna skoro 100% vremena. Ako se potrebna dostupnost definiše kao ’5 devetke’ odnosno 99.999% ili ’6 devetke’ - 99.9999%, sve komponente u sistemu moraju da zadovolje pooštrene uslove kako bi se taj nivo dostupnosti ostvario. Uzmimo u obzir da dostupnost ’6 devetke’ dozvoljava da sistem ne bude dostupan samo 32 sekunde godišnje. Lako je razumljivo da pod takvim strogim zahtevima sve akcije skrivanja kvarova moraju trajati najviše sekundu. Ovakve baze podataka se nazivaju Visoko Dostupne Baze Podataka (eng. *Highly Available Databases*), a sistemi kojima se ostvaruju takve baze se nazivaju Sistemi za upravljanje bazama podataka visoke dostupnosti (eng. *Highly Available Database Management Systems – HA-DBMS*).

Svi HA-DBMS rade slično HA aplikacijama: visoka dostupnost se postiže *redundantnošću procesa* – više instanci procesa su pokrenuti u isto vreme, tipično na hardveru u više-čvornom klasteru. Pored jednog ili više *aktivnih procesa* koji rade kao servis, postoje i *procesi na čekanju*, ili redundantni aktivni procesi, koji su pokrenuti na drugim računarskim čvorovima i spremni su da preuzmu operaciju (i nastave servis), ako bi aktivni proces na neki način bio ugrožen. Procesi baza podataka uključuju podatke čije su stanje i dostupnost krucijalni za uspešan nastavak servisa. Odavde govorimo o *redundansi podataka*, koja ima za cilj da omogući dostupnost podataka u slučaju da neke komponente koje drže podatke zakažu.

(ovde ide uvodni tekst o SQL Server AOHA-DBMS-u)

# Modeli redundantnosti

HA sistemi baza podataka koriste više koncepata redundantnosti. Svi HA-DBMS sistemi se zasnivaju na postojanju redundantnih procesa baze podataka. Kada proces baze podataka zakaže (bilo hardverski ili softverski problem), drugi proces baze podataka preuzima odgovornost opsluživanja servisa koji je prvi proces prethodno kvaru opsluživao. Da bi se pružila korektnost/tačnost podataka, neophodno je da svaki redundantni proces vidi isti set izmena nad bazom podataka. U suštini postoje dva načina dase ovo osigura: jedna tehnika, replikacija, oslanja se na procese baza podataka koji eksplicitno razmenjuju izmene između sebe. U zavisnosti od implementacije, svaka replika može da čuva svoju kopiju u glavnoj memoriji (RAM) ili na disku. Replikacija se ne radi ekskluzivno između pojedinačnih baza. U distribuiranim sistemima baza podataka, jedna baza podataka biva upravljana preko više procesa baza podataka na različitim čvorivima, sa mogućim internim replikacijama između sebe (eng. *intra-database*). Alternativa, druga tehnika, je osigurati da svi redundantni procesi vide isti set podataka tako što stvarno ,,gledaju’’ isti set podataka. Ovo je moguće postognuti deljivim diskovima (eng. *Shared-Disks*) tako što svi procesi pristupaju istom setu diskova. Kako svi procesi pristupaju istom setu diskova, različiti procesi baza podataka ne trebaju eksplicitno razmenjivati izmene. Umesto toga, svi procesi imaju jedan, koherentan pogled na podatke. Primetimo da deljivi disk takodje može imati redundansu, ali ovo je redundansa na nižim nivoima – npr. RAID ili kopiranje preko mreže (eng. *Network-Based Mirroring*) – i proces je često apsolutno transparentan DBMS-u.

Dva pristupa pomenuta iznad mogu se mapirati na dve dobro poznate DBMS arhitekture: *deljivo-ništa*  i *deljivi-disk* [2], respektivno. U ovom radu se fokusiramo na sama sredstva kojima dolazimo do visoke dostupnosti.

Više modela redundantnosti je moguće u HA-DBMS sistemu i oni su definisani ispod. Razlikujemo *redundantnost procesa* koja definiše dostupnost procesa baza podataka i *redundansu podataka* koja specificira, za rešenja zasnovana na replikaciji, broj kopija podataka koje se eksplicitno održavaju. Oba tipa redundantnosti su potrebna da bi se pružio Visoko dostupan servis baze podatka (eng. *High Availability Database Service*).

## Redundansa procesa

Redundansa procesa u HA-DBMS-u dozvoljava da DBMS nastavi svoju operaciju u prisustvu zakaza procesa. Kao što ćemo reći kasnije, većina modela redundanse procesa može biti implementirana i preko tehnologije deljivog diska i replikacije.

Proces koji je u *aktivnom* (eng. *Active*) stanju je onaj koji trenutno pruža (ili je osposobljen da pruža) servis baze podataka. Proces koji je u stanju *na čekanju* (eng. *Standby*) trenutno ne pruža servis ali je spreman da preuzme ulogu davanja servisa na veoma brz način, ako trenutni aktivan proces zakaže. Ovo se naziva (eng.) *failover*. U nekim slučajevima, novi tip procesa, *rezervni* (eng. *Spare*) proces se može koristiti. Rezervni proces može biti implementiran ili kao pokrenuta komponenta kojoj nije zadat nikakav izvršni zadatak (eng. *workload*), ili kao komponenta koja je definisana ali nije instancirana. *Rezerva* može biti uzdignuta na nivo *aktivna* ili *na čekanju* nakon odgovarajuće inicijalizacije.

Redundansa procesa donosi pitanje kako (ili da li) će transparentnost redundanse biti održana u HA-DBMS-u. Od svih pokrenutih procesa, neki mogu biti aktivni a neki ne. U slučaju failover-a aktivni procesi mogu biti promenjeni. Zadatak nalaženja relevantnih aktivnih procesa može biti odgovornost ili aplikacije ili posebnog softverskog sloja (eng. *Software Layer*) koji je zadužen za transparenciju redundanse.

## Redundansa podataka

Redundansa podataka je takođe neophodna za visoku dostupnost (HA). U suprotnom, gubitak jedne jedine kopije podataka bi rezultiralo time da baza postaje nedostupna kao takva. Redundansa podataka može biti pružena na fizičkom ili logičkom nivou.

### Fizička redundansa podataka

Fizička redunsansa podataka se odnosi na oslanjanje na softver i/ili hardver ispod baze podataka za održavanje više fizičkih kopija podataka. Iz perspektive baze podataka, čini se da postoji samo jedna kopija podataka. Neki primeri fizičke redundanse podataka uključuju: replikacija diska, RAID, udaljena replikacija diska, replicirani fajl sistemi...

Sve ove tehnologije dele zajednički atribut da održavaju odvojenu fizičku kopiju na moguće različitoj geografskoj lokaciji. Kada je primarna kopija podataka izgubljena, procesi baze podataka koriste drugu kopiju podataka. Ove tehnologije se mogu razlikovati u pogledu transparentnosti zakaza koji je podržan. Repliciranje diska (eng. *Disk Mirroring*) i RAID, npr. , čine fizičku redundansu apsolutno transparentnom bazi podataka.

Fizička redundansa podataka se često kombinuje sa redundansom procesa koristeći SAN (eng. *Storage Area Network*). Ovo dozvoljava da više čvorova ima pristup istoj fizičkoj bazi podataka. Ako jedan server baze zakaže (usled softverske ili hardverske greške), baza podataka je i dalje dostupna kroz druge čvorove. Ovi drugi čvorovi nakon toga mogu da nastave servis.

### Logička redundansa podataka koristeći replikaciju

Logička redundansa podataka se odnosi na situaciju gde baza podataka eksplicitno održava višestruke kopije podataka. Transakcije koje se primenjuju nad primarnom bazom D se repliciraju nad sekundarnom bazom D’ koja je manje-više ažurirana u odnosu na primarnu bazu, a u zavisnosti od tipa sinhronizacije replikacionog protokola u HA bazi podataka. Dodatno replikaciji između baza podataka (eng. *inter-database*), replikacija u samoj bazi podataka (eng. *intra-database*) se koristi u distribuiranim sistemima da bi se postigla visoka dostupnost koristeći samo jednu bazu podataka. Primetimo da govorimo o replikaciji u generalizovanom obliku kako je šema replikacije zavisna od dobavljača/pružaoca/tvorca konkretne tehnologije baze podataka (pretpostavljajući da baza podataka i replikacija dolaze od istog dobavljača). Replikacija može biti *sinhrona* (eng. *Synchronous*) i *asinhrona* (eng. Asynchronous), može biti bazirana na prosleđivanju logova ili direktne replikacije kao deo transakcije, transakcije mogu biti grupisane (eng. *Batched*) i moguće kombinovane grupom izvršenja/posvete/komita (eng. *Commits*). Odabrana metoda zavisi od samog produkta baze podataka i potrebnog nivoa *sigurnosti* (eng. *Safeness*) [3]. Kod (eng.) *1-safe* replikacije (,,asinhrona replikacija’’) transakcije se repliciraju nakon što se *izvrše/komituju* nad primarnom kopijom. Kod (eng.) *2-safe* replikacije (,,sinhrona replikacija’’) transakcije se repliciraju u sekundarnoj kopiji, ali se ne komituju pre nego što se potvrdi komit na primarnoj kopiji. Kod (eng.) *2-safe committed* replikacije transakcije se repliciraju i komituju na sekundaru pre nego što se potvrdi komit na primaru. Kod (eng.) *very safe* replikacije sve operacije osim čitanja se isključuju ako bilo primarna ili sekundarna replika postane nedostupna. Iako se većina metoda pružanja sigurnosti odnosi na kontekst udaljenih rezervi, rezultati su važeći i za (eng.) *in-cluster* operacije takođe.

# Modeli redundansi podataka

Za trajanje ostatka ovog prvog, teorijskog dela, redundansa podataka se odnosi na *logičku* redundansu podataka. Redundansa podataka predstavlja broj samostalnih kopija podataka koje se održavaju od strane procesa baze podataka preko replikacije. Ne računaju se kopije koje se možda održavaju od strane nekog fizičkog modela redundanse. Npr. dve kopije podataka koje se održavaju od strane niza diskova (eng. *Disk Array*) ili od strane menadžera particija se računaju kao jedna u kontekstu ove diskusije, dok se dve kopije podataka koje se održavaju od strane baze podataka računaju kao dve. Primetimo da u oba slučaja, gubitak jedne kopije se može obraditi transparentno bez gubitka dostupnosti.

Prvo raspravljamo modele redundantnosti podataka u detalje zato što je ovo oblast koja je poprilično jedinstvena za DBMS-ove visoke dostupnosti (HA-DBMS-ovi).

## Fragmenti baze, particionisanje i replikacija fragmenata

Da bismo definisali modele redundantnosti podataka moramo definisati šta zapravo repliciramo, odnosno *fragmente baze podataka* (fragment je ovde generalizacija ustaljene definicije fragmentacije tabele u relacionoj bazi podataka). *Fragmentacija* baze podataka je dekompozicija baze podataka D u fragmente P1, P2... Pn koja mora ispuniti sledeće uslove:

1. *Kompletnost* – svi podaci koji postoje u bazi podataka moraju postojati u nekom fragmentu.
2. *Rekonstrukcija* – treba biti moguće rekonstruisati kompletnu bazu podataka od fragmenata.
3. *Odvojenost* – bilo koji podatak sadžan u jednom fragmentu ne sme postojati ni u jednom drugom fragmentu (ovo normalno važi za horizontalnu fragmentaciju, ali ne isključuje vertikalnu fragmentaciju ako uzmemo u obzir replicirani primarni ključ može biti identifikator podataka umesto samih podataka).

Granularnost fragmenta se tipično izražava u kontekstu modela podataka koji se koristi. U relacionim bazama podataka, fragmenti mogu biti povezani sa kompletnim SQL *šemama* (takođe nazivanim *katalozima*) ili setovima njihovih tabela. Najmanja postignuta granularnost se obično naziva *horizontalnom* ili *vertikalnom* fragmentacijom gde ,,horizontalno’’ označava deljenje tabele po redovima a ,,vertikalno’’ – po kolonama. Primetimo da ova definicija fragmentacije ne isključuje posmatranje baze podataka kao jedan entitet ako je ovo potreban logički pogled na bazu podataka.

Nereplicirana, *particionisana* baza podataka sadrži fragmente koji su alocirani procesima baze podataka, normalno na odvojenim čvorovima klastera, sa samo jednom kopijom bilo kog fragmenta na celom klasteru. Takva šema nema jake HA (*High Availability*) sposobnosti. Da bi se postigla visoka dostupnost podataka, *replikacija fragmenata baze podataka* se koristi da bi se dozvolilo smeštanje i pristup podacima u veše od jednog čvora. U *potpuno repliciranoj* bazi podataka sama baza podataka postoji u svojoj celosti u svakom procesu baze podataka. U *parcijalno repliciranoj* bazi podataka fragmenti baze podataka su distribuirani kroz procese baze podataka na takav način da kopije fragmenta, odsad referencirane kao *replike*, mogu postojati u većem broju procesa baze podataka.

U replikaciji podataka, fragmenti se mogu klasifikovati kao *primarne* replike (Primari (eng. *Primary*)) ili kao *sekundarne* replike (Sekundari (eng. *Secondaries*)). Primarne replike predstavljaju *pravi* fragment podataka (ako primarna replika nije dostupna onda fragment nije dostupan, stoga baza podataka nije dostupna) i mogu se čitati u menjati. Sekundarne replike se uglavnom mogu samo čitati (eng. *read-only*) i manje-više su ažurirani u odnosu sa primarnom replikom. Sekundarne replike se mogu unaprediti u primarne replike u toku procesa (eng.) *failover*.

## Veze kardinalnosti između primarnih i sekundarnih replika

### 1\*Primar/1\*Sekundar

Ovde svaki fragment ima tačno jednu primarnu repliku koja je replicirana na tačno jednu sekundarnu repliku. Ovo je veoma čest model redundanse pošto je uviđeno da su dve replike adekvatne za postizanje visoke dostupnosti u većini slučajeva.

### 1\*Primar/Y\*Sekundar

Ovde svaki fragment ima tačno jednu primarnu repliku koja je replicirana u veći broj sekundarnih replika. Ovaj model pruža veću dostupnost of *1\*Primar/1\*Sekundar* i dozvoljava veću pristupačnost radi čitanja ako je dozvoljeno čitanje sa sekundarnih replika.

### 1\*Primar

Ovde svaki fragment postoji u tačno jednoj primarnoj replici. Ovaj model ne pruža nikakvu redundansu na nivou baze podataka. Redundansa je pružena bazi podataka od strane nižih slojeva (skladištenja – eng. *storage*).Koristi se u sistemima sa deljenim diskom i takođe u centralizovanim ili particionisanim bazama podataka.

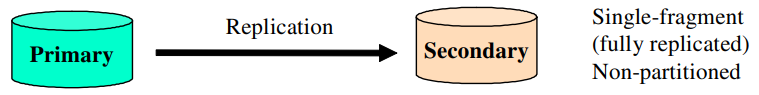
### X\*Primar

Ovde svaki fragment ima veći broj primarnih replika i koristi se *N\*Aktivan* model redundanse procesa (može se nazvati i *multi-master*). Ovaj model dozvoljava veći broj pristupa radi čitanja i upisa od *1\*Primar* ako se pokušaj upisa u isti fragment ne desi u paraleli (jer bi ovo dovelo do konflikta upisa).

## Veze između baze podataka i fragmenata

### Neparcionisana replicirana baza podataka

Najčešći slučaj je kada su baza i fragment jedno isto. Posledično, cela baza se replicira u sekundarnu lokaciju (Ilustracija 1). Primetimo: Svi slučajevi u ovoj podsekciji su ilustrovani pretpostavljajući kardinalnost *1\*Primar/1\*Sekundar*.

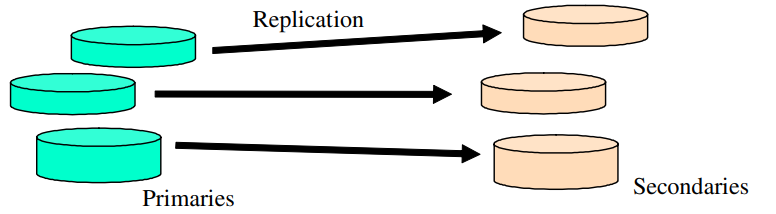


Ilustracija 1. Neparticionisana baza podataka

### Particionisana replicirana baza podataka

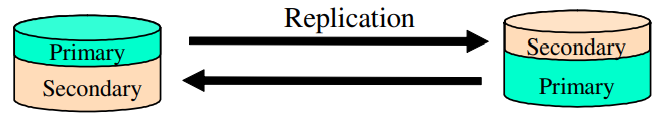
U ovom modelu, postoje fragmenti čija je svrha biti alociran na različitim čvorovima ili biti repliciran na različitim čvorovima (Ilustracija 2).

### Mešani replicirani fragmenti



Ilustracija 2. Particionisana baza podataka

Specijalan slučaj particionisane baze podataka je baza podataka sa mešanim (eng. *Mixed*) particijama gde jedna baza može sadržati (hostovati – eng. *Host*) i primarne i sekundarne fragmente. Specijalan slučaj su dve baze podataka sa simetričnim fragmentima (Ilustracija 3).

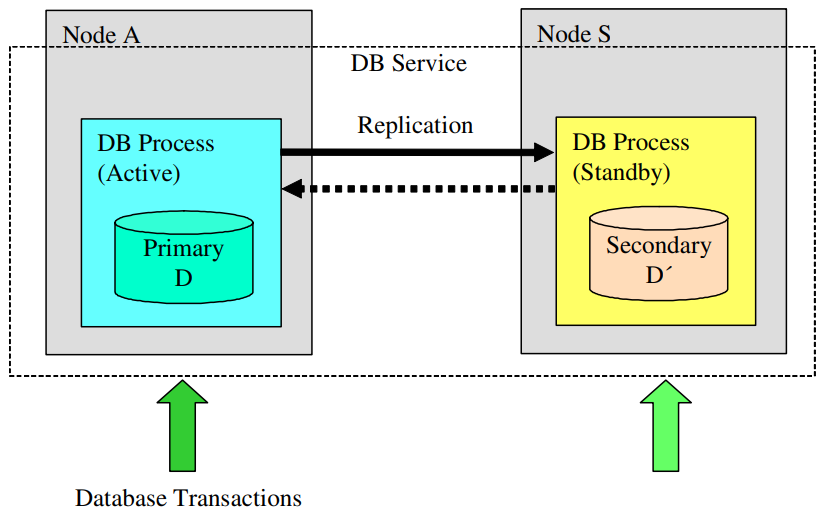


Ilustracija 3. Dve baze podataka sa simetričnim fragmentima

# Modeli redundanse procesa

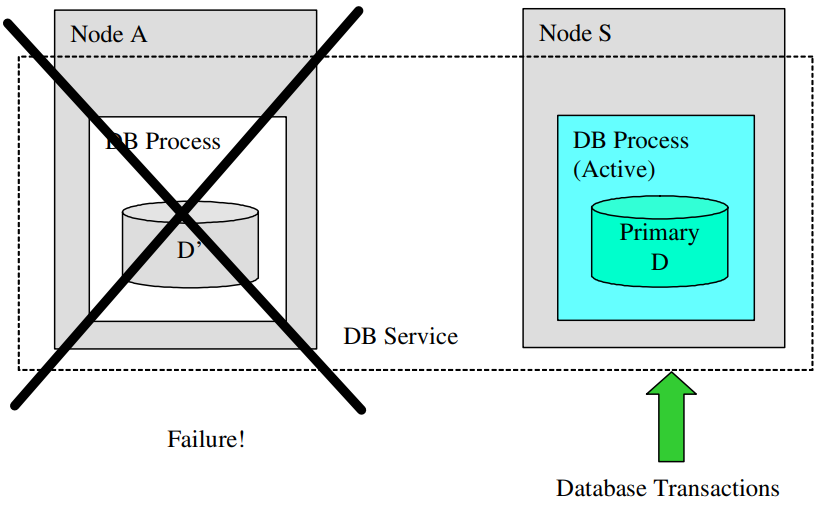
## Aktivan/Na čekanju (2N)

*Aktivan/Na čekanju* (ponekad referenciran kao 2N) je model redundantnosti procesa za HA-DBMS-ove koji je podržan od strane i *sistema replikacije* i *sistema deljenog diska*. Svaki *aktivan* proces baze podataka se *bekapuje* (čuva – eng. *Back up*) od strane procesa baze na drugom čvoru koji je *na čekanju*. U ilustraciji 4, prikazan je primer sistema replikacije, a u ilustraciji 5 je prikazan primer sistema deljenog diska. Sve izmene (eng. *Updates*) se moraju obaviti nad procesom aktivne baze podataka; one će biti propagirane procesom replikacije, ili preko deljenog diska, procesu baze podataka koja je *na čekanju*.

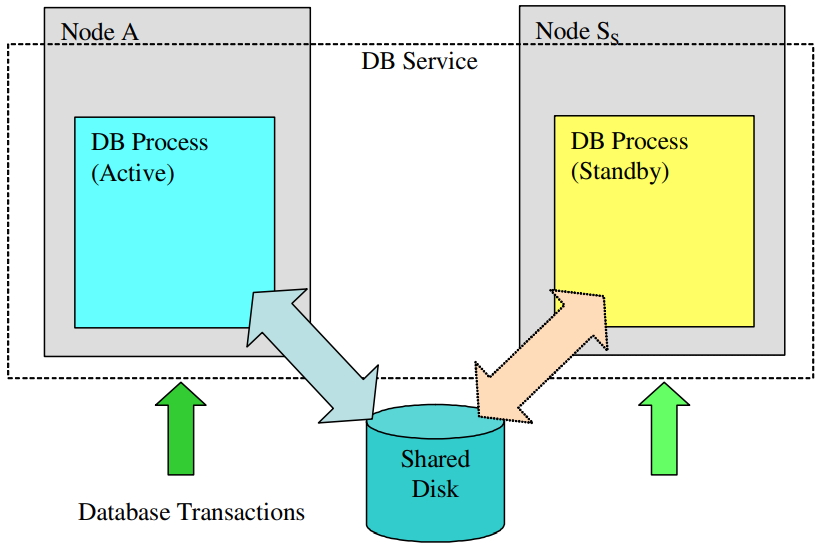


Ilustracija 4. Aktivan/Na čekanju model redundanse koristeći replikaciju

U slučaju da zakaže *aktivan* proces baze podataka (iz bilo kog razloga kao što je softverska greška na serveru baze podataka ili hardverska greška u čvoru – računaru) proces baze podataka koji je *na čekanju* će preuzeti i postaće novi aktivni proces baze podataka (Ilustracija 6). Ako se prethodno *aktivan* proces baze podataka koji je zakazao oporavi od problema, on će postati novi proces baze podataka koji je *na čekanju* i procesi baze podataka će potpuno promeniti uloge (Ilustracija 7). Ako HA baza podataka ima *preferiran aktivan* (eng. *preffered active*) proces baze podataka, može se kasnije prebaciti na originalnu konfiguraciju.



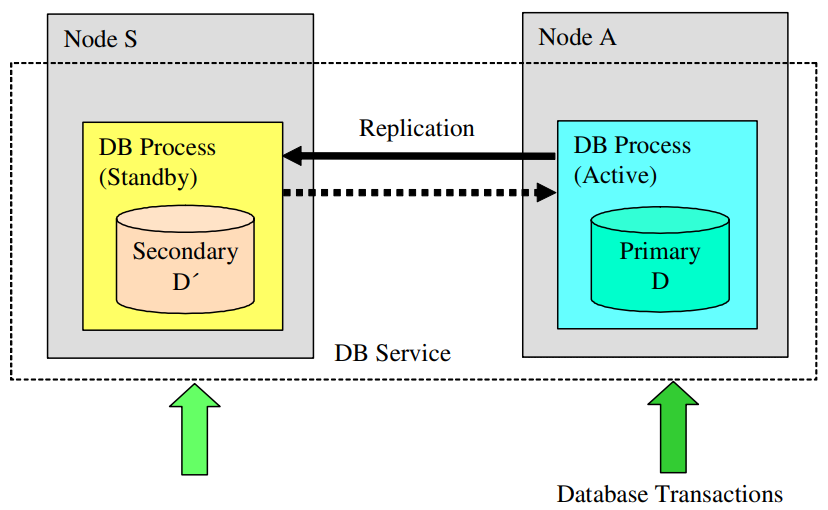
Ilustracija 5. Zakaz na aktivnom primaru, prelazak sekundara u ulogu primara



Ilustracija 6. Aktivan/Na čekanju model redundanse koristeći deljeni disk

Proces baze podataka koji je *na čekanju* će biti definisan kao manje-više spreman da preuzme u zavisnosti od odabranog nivoa sigurnosti (eng. *Safeness Level*) i HA zahteva aplikacija. Da bi se klasifikovali procesi baze podataka koji nisu *aktivni* pravimo razliku između *vruće na čekanju* (eng. *Hot Standby*) i *toplo na čekanju* (eng. *Warm Standby*).

### Hot Standby



Ilustracija 7. Obrnute uloge

Jedan proces baze podataka se *bekapuje* od strane procesa baze podataka koji je *na čekanju* koji je spreman manje-više odmah (vreme ~ ispod jedne sekunde) da preuzme u slučaju da aktivna baza podataka zakaže. Aplikacije već mogu biti konektovane na proces baze podataka koji je *na čekanju* ili trebaju biti ponovo konektovane na proces koji je *na čekanju* (sada već *aktivan*).

### Warm Standby

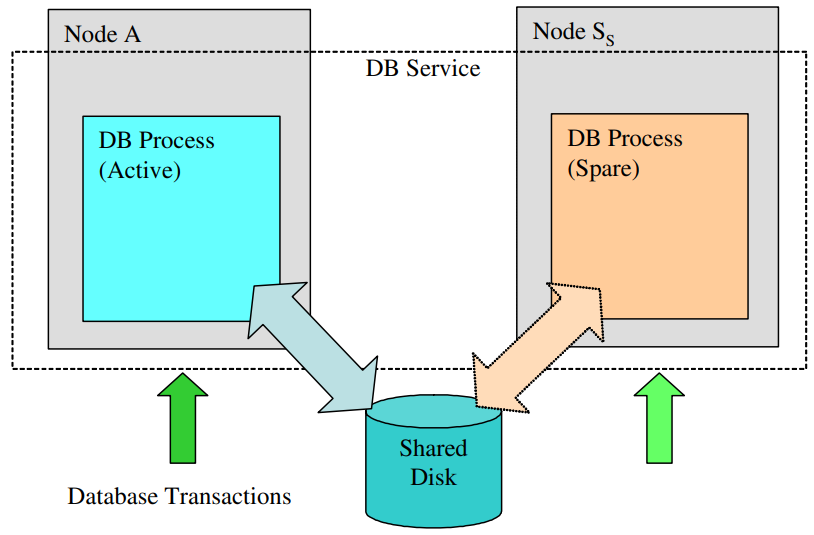
Jedan proces baze podataka se *bekapuje* od strane procesa baze podataka koji je *na čekanju* koji je spreman da preuzme nakon određene sinhronizacije/rekonektovanja sa aplikacijama u slučaju da *aktivna* baza podataka zakaže. U ovom slučaju, *failover* može trajati nekoliko desetina sekundi do nekoliko minuta.

U sledećoj sekciji ćemo uključiti *rezerve* (eng. *Spares*) i napravićemo razliku između procesa koji su *na čekanju* i *rezervnih* procesa, kako je moguće imati model *Aktivan/Na čekanju/Rezerva*.

## Aktivan/S\*Rezerva

*Aktivan/S\*Rezerva* (jedan *aktivan* i S *rezervi*) je konfiguracija u kojoj su nekoliko *rezervi* procesa baze podataka unapred konfigurisani na nekim čvorovima. Podržani su i sistemi sa replikacijom i sistemi sa deljenim diskom. Primer arhitekture sa *rezervom* na deljenom disku dat je ispod (Ilustracija 8).

U bazi podatka sa deljenim diskom, ako aktivan proces zakaže, deljeni disk koji sadrži datoteke baze podataka se *dodeljuje* (eng. *Mount*) čvoru koji hostuje *rezervu* (rezervni čvor), *rezerva* biva inicijalizovana bazom podataka, i baza podataka postaje aktivna na tom čvoru. Ako se čvor koji je zakazao restartuje on će postati novi *rezervni* čvor. Čvorovi su ovde znači potpuno zamenili uloge. Ako HA baza podataka ima *preferiran aktivan* proces baze podataka, kasnije će se prebaciti na originalnu konfiguraciju.



Ilustracija 8. Aktivan/S\*Rezerva model redundanse koristeći deljeni disk

U HA-DBMS-u sa replikacijom, *rezerva* dobija bazu podataka pre nego što postane *aktivna*. Nivo dostupnosti koji pruža ovaj model je niži od onog koji pruža model *Aktivan/Na čekanju* zbog dodatnog vremena potrebnog za inicijalizaciju *rezerve*.

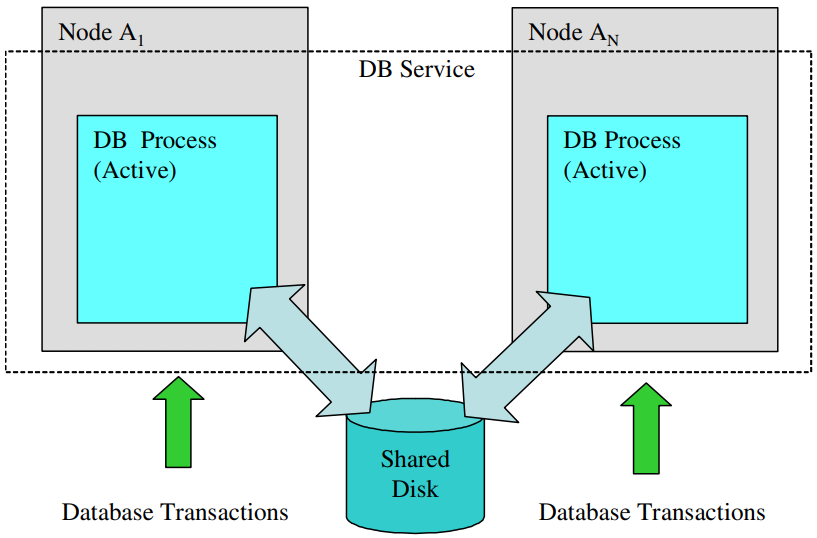
## N\*Aktivan

U većim klasterima, sistem baze podataka može iskorisitti više od dva čvora da bi bolje koristio moć procesiranja, memoriju, lokalne diskove i ostale resurse čvorova, da bi balansirao opterećenje na najbolji mogući način. U *N\*Aktivan* (referenciran i kao *N-Way Active*) modelu redundanse procesa, N procesa baza podataka su aktivne i aplikacije mogu pokretati transakcije na bilo kom procesu. Ovde svi procesi podržavaju jedni druge u slučaju zakaza i svaki može manje-više momentalno preuzeti. Sve *komitovane* (eng. *Committed*) izmene urađene nad jednim procesom baze podataka su dostupne drugim procesima i obrnuto. U sistemima sa deljenim diskom, svi procesi baze podataka vide jedan isti set izmena. U replikacionom sistemu, sve izmene se repliciraju na sve procese.

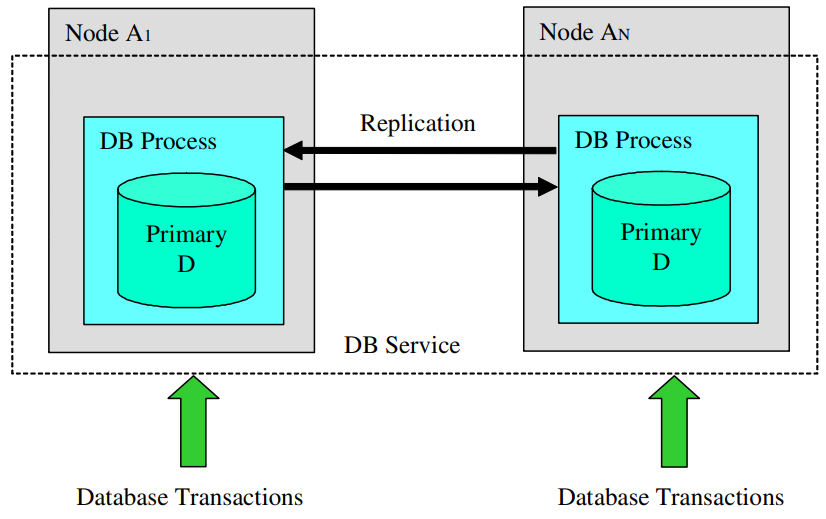
Baza podataka je potpuno dostupna iako svi procesi baze podataka i svi rekordi mogu biti ažurirani u svim procesima. U slučaju istovremenog konflikta izmena, konzistencija (verodostojnost – eng. *Consistency*) kopija može biti ugrožena, u sistemima sa replikacijom. Ovo se prevazilazi koristeći (eng.) *distributed concurrency control system* ili (eng.) *copy update reconciliation method*. U sistemu sa deljenim diskom, infrastruktura baze podataka može biti prostija zato što objekti podataka nisu replicirani. Baza podataka interno implementira (eng.) *lock manager* kako bisprečila konfliktna ažuriranja istih podataka. Ilustracija 9 prikazuje *N\*Aktivan* model zasnovan na deljenom disku. Primetimo da koristimo 2 čvora kao primer iako model podržava više od 2 čvora. Ilustracija 10 prikazuje *N\*Aktivan* model zasnovan na replikaciji sa 2 čvora.

Ako baza podataka nije potpuno replicirana i postoje mešani fragmenti u svim bazama podataka, model procesa je uvek *N\*Aktivan*. Npr. u ilustraciji 11, HA-DBMS sa *2\*Aktivan* modelom je prikazan kako koristi simetričnu replikaciju. Sa simetričnom replikacijom, problemi sa kontrolom konkurencije se izbegavaju a prednost nivelisanja opterećenja je zadržana.

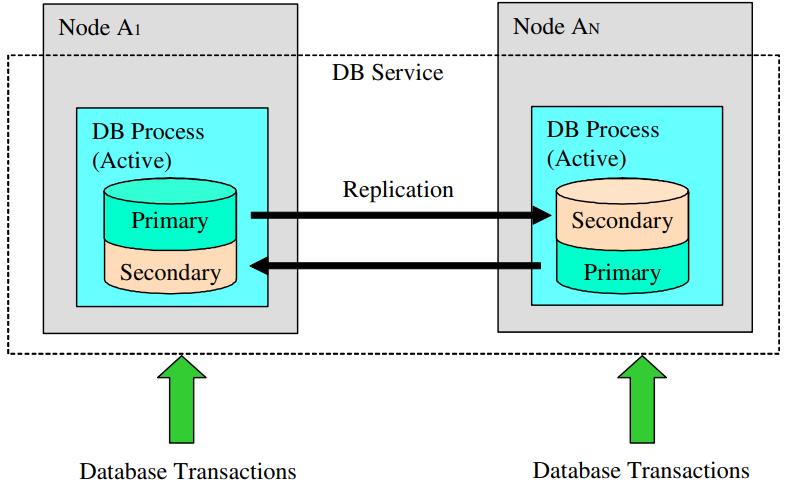
Za razliku od većine *N\*Aktivan* okruženja, u ilustraciji 11, šema particionisanja je vidljiva aplikacijama i često je zasnovna na particionisanju opsega primarnih ključeva (eng. *Primary Key range*). Aplikacije su odgovorno za pristupanje pravom (potrebnom) aktivnom procesu. Inter-particijske (eng. *Inter-partition*) transakcije nisu podržane u normalnom slučaju.



Ilustracija 9. N\*Aktivan, deljeni disk



Ilustracija 10. N\*Aktivan, potpuna replikacija



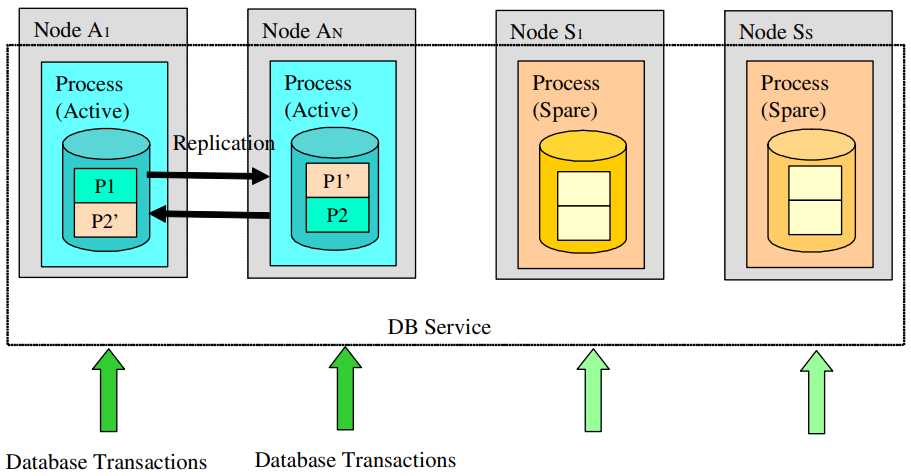
Ilustracija 11. 2\*Aktivan sistem sa simetričnom replikacijom baze podataka

## N\*Aktivan/S\*Rezerva

*N\*Aktivan/S\*Rezerva* (eng. *N\*Active/S\*Spare*) – N *aktivnih* i S *rezervnih* procesa – je varijanta *Aktivan/S\*Rezerva* modela kada N *aktivnih* procesa baze podataka pruža dostupnost do particionisane baze podataka. Kao u modelu *N\*Aktivan*, *aktivni* procesi se mogu oslanjati na deljeni disk, mogu koristiti potpuno replicirane baze podataka ili mešane fragmente (parcijalno replicirane baze podataka – eng. *Partially replicated databases*).

Kao primer parcijalno (simetrično) replicirane baze podataka sa *rezervama* data je ilustracija 12.

Svaki proces baze podataka održava neke fragmente bae podataka i *rezervni* procesi mogu preuzeti u slučaju greške kod *aktivnog* procesa baze podataka. *Rezerva* mora dobiti relevantne fragmente od *aktivnog* procesa prilikom pokretanja.



Ilustracija 12. N\*Aktivan/S\*Rezerva model redundanse, parcijalno replicirana baza podataka

## Drugi modeli redundanse

Neki sistemi kombinuju više modela redundanse da bi postigli drugačije nivoe redundanse podataka i procesa. *M-standby*, *cascading standby* i *geografski replicirani N\*Aktivan klasteri* (eng. *Geographically replicated N\*Active clusters*) su neki primeri. Kako su oni sadržani od drugih modela redundanse koji su ovde već prezentovani, neće biti dalje diskutovani.

# Pogled aplikacije

Aplikacije mogu i ne moraju biti svesne modela redundasi koje koriste razne komponente sistema baza podataka. U *Aktivan/Na čekanju* sistemia, aplikacije obično trebaju biti svesne tako da bi se mogle ponovo konektovati na *aktivnu* instancu. Štaviše, drugačiji aspekti sistema baza podataka mogu ustvari koristiti različite modele redundanse. Na primer, DBMS može imati jedan set procesa koji kontroliše podatke, a drugi set procesa koji izvršava upite i kojima se klijentske aplikacije obraćaju. ovi setovi procesa mogu imati totalno drugačije modele redundanse, i mogu komunicirati međusobno na različite načine.

Srodna tema je šema particionisanja. Generalno, bolje je sakriti particionisanje fragmenata od aplikacije. Ovo dozvoljava da aplikacija ne mora trpeti izmene ako se promeni šema particionisanja zbog planirane ili neplanirane rekonfiguracije. Držanje šeme particionisanja internom serveru baze podataka dozvoljava interno balansiranje opterećenja i reorganizaciju podataka bez uticanja na aplikacije. Iz razloga performansi neki sistemi pružaju neki koncept lokaliteta i podržavaju kolociranje aplikacije i podatke na istom čvoru. Ovo ponekad može biti kontrolisano kroz ,,naznake’’ (eng. *Hints*) od strane aplikacije ka serveru baze podataka o tome gde je najbolje da se podaci efektivno skladište. Šeme logičkog particionisanja za aplikacije i podatke se često kombinuju sa opštim šemama balansiranja opterećenja (eng. *Load balancing*) koje su ugrađene u *stekovima distribuirane komunikacije* (eng. *Distributed communication stacks*).

# Šema nepredviđenih slučajeva

(objasniti *Contingency plans*; dodati <https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/bcm-resilience/bc-plan/it-service-continuity-plan> u reference)

# SQL Server Always On Availability Groups

SQL Server Always On Availability Groups (skraćenica AG) je Microsoft-ovo rešenje za HA-DBMS ili za skaliranje čitanja (eng. *read-scale*). U ovom radu će se najviše obratiti pažnja na HA-DBMS koncept SQL Server AG.

Funkcionalnost je dostupna od verzije SQL Server 2012 (11.x), pa nadalje, sa unapređenjima u svakoj verziji. Unapređenja u ovom konktekstu mogu biti potpuno nove funkcionalnosti – kao npr. u SQL Server 2019 (za razliku od SQL Server 2017) *Automatic seeding* opcija kao preferenca početne sinhronizacije primarne i sekundarnih baza, ili unapređene funkcionalnosti – kao npr. u SQL Server 2019 (za razliku od SQL Server 2017) povećanje broja mogućih replika koje mogu biti u modu za automatski *failover* sa 3 na 5. Dalja obrada teme će se odnositi na SQL Server 2019 (15.x) Enterprise. Naglašavamo *Enterprise* iz razloga što postoje ograničenja kod Standard verzije; malo o tome kasnije u odeljku *Licence*.

SQL Server AG podržava do 9 replika dostupnosti, od kojih 5 mogu podržavati automatski *failover*. Sve validne replike moraju biti smeštene na pojedinačnom čvoru u jedinstvenom *Windows Server Failover Clustering* (WSFC) klasteru. Moguće je pridružiti i baze koje nisu u WSFC-u, ali one samo mogu služiti za čuvanje konfiguracija; malo o tome kasnije.

U SQL Server AG tehnologiji, podrazumevaju se *Neparticionisana replicirana baza podataka* kao i *1\*Primar/Y\*Sekundar* (ili *1\*Primar/1\*Sekundar* ako uzmemo vrednost 1 za Y) modeli redundanse podataka. Takođe podrazumeva se repliciranje baza podataka na nivou redundanse podataka, mada je moguće koristiti AG tehnologiju u kombinaciji sa *shared-disk* rešenjima.

## Uslovi implementiranja

(uslovi koje je potrebno ispuniti kako bi se implementirala AG funkcionalnost – WSFC i witness, FCI, AD role i nalozi, mreža, deljeni direktorijumi, identična lokacija (za seeding))

## Tipovi replika

(*Availability Replicas -* objasniti primarnu i sekundarnu repliku, funkcionalnosti koje mogu pružati i ograničenja; [4])

## Modovi dostupnosti

(*Availability Modes* – sync, async, config; [5] – detaljnija objašnjenja)

### Latencija podataka na sekundarnim replikama

(objasniti razlog postojanja latencije; [5]) – obrati pažnju da je ovo pod-podnaslov

## Tipovi failover-a

(*planned manual* i *auto* (u *sync*) ili *forced manual* (u *async*))

## Konekcije klijenata

(objasniti *AG listener* i pojedinačno povezivanje)

## Aktivne sekundarne replike

(*backup-offloading* i *load-balancing*)

## Period isteka timeout-a

(default, minimum, preporučeno; naći kako se sve to podešava; pingovanje)

## Automatsko popravljanje stranica

(objasniti *automatic page repair*)

## Oporavak replika

(objasniti kako se oporavlja replika koja ja prethodno zakazala; povezati sa *Contingency plans* ??)

## Licence

Ovde ćemo kratko napomenuti bitne razlike između SQL Server 2019 (15.x) *Enterprise* i *Standard* licenci. Druge licence nećemo pominjati kako nisu ni namenjene za realna *production* okruženja. *Standard* licenca je izuzetno ograničena što se tiče Always On Availability Groups funkcionalnosti u odnosu na *Enterprise* licencu (verziju). Ograničenja *Standard* licence:

1. Limitirana na 1 bazu podataka unutar jedne AG (kod *Enterprise* verzije nema tog ograničenja)
2. Limitirana na 2 replike unutar jedne AG (kod *Enterprise* verzije taj broj je 9)
3. Ne postoji mogućnost *read-only access* na sekundarnoj replici
4. Ne postoji mogućnost *backup-offload* na sekundarnoj replici
5. Nema GUI podešavanja – sve se mora konfigurisati T-SQL naredbama ili PowerShell-om

Ograničenje #1 samo primorava da se mora praviti više AG što ne zvuči previše ograničavajuće, ali zapravo pravi interni *overhead* kako se za svaku AG moraju pokretati posebni interni procesi i vršiti odvojena sinhronizacija.

Ograničenje #2 je zapravo najveće ograničenje što se tiče HA funkcionalnosti. Istina, za veliki broj realnih rešenja više od jedne sekundarne replike nije ni potrebno, ali ponovo smanjuje fleksibilnost umnogo.

Ograničenje #3 sprečava gasi *load-balancing* funkcionalnost.

Ograničenje #4 ne dozvoljava delegiranje *backup* operacija na sekundarne replike, što može usporavati primarnu repliku.

Ograničenje #5 samo ograničava način rada sa samim sistemom, odnosno ne možemo koristiti grafičke alate unutar *SQL Server Management Studio*-a (SSMS-a). Ono samo po sebi predstavlja samo ograničenje fleksibilnosti rada sa sistemom, ne utiče na funkcionalnosti koje sistem pruža.

## Primer iz prakse

(navesti i objasniti okruženje, licence, šemu baze i cilj)

### Podešavanje okruženja

(brz pregled podešavanja u Windows-u – svrhe servera, šta je instalirano na svakom, role, deljivi diskovi; [6])

### Kreiranje nove AG

(objasniti svaku opciju u Wizard-u i povezati svaku sa teorijskim delom)

### Pregled zdravog klastera

(pokazati AG monitor i šta pokazuje)

### Failover

(simulacija zakaza primarnog servera i automatski failover; ručni failover)

# Zaključak

(čemu služi HA; šta je dobro kod SQL Server AG; dobre prakse)

# Literatura

[1] Drake, S., Hu, W., McInnis, D., Sköld, M., Srivastava, A., & Thalmann, L., Tikkanen, M., Torbjørnsen, Ø., Wolski, A. (2005). Architecture of Highly Available Databases. In M. Malek, M. Reitenspieß, & J. Kaiser (Eds.), Service Availability (Vol. 3335, pp. 1–16). Springer.

[2] Norman, M.G., Zurek, T., Thanisch, P.: Much Ado About Shared-Nothing. SIGMOD Record 25(3): 16-21 (1996)

[3] Gray, J. and Reuter, A.: Transaction Processing Systems, Concepts and Techniques. Morgan Kaufmann Publishers, 1992

[4] https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/overview-of-always-on-availability-groups-sql-server?view=sql-server-ver15

[5] https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/availability-modes-always-on-availability-groups?view=sql-server-ver15

[6] https://sqlundercover.com/2017/12/18/creating-a-sql-server-test-lab-on-your-workstation-part-one-installing-the-domain-controller/

[7]