Seminarski rad

Replikacija u MongoDB

Sistemi baza podataka

Stefan Mladenov 1205

# 

[**1. Uvod 3**](#_tl1v1kxm99sl)

[**2. Replikacija kod MongoDB-a 5**](#_evjsdcjfdxjo)

[2.1 Replika Set Konfiguracija 7](#_zd1tuq96oq8u)

[2.2 Čitanje podataka 8](#_1ahqksgtfgif)

[2.2.1 Komanda 9](#_bqh6t7ohcdv6)

[2.2.2 Primer 9](#_4k1rwn94foap)

[2.3 Izbor primarnog čvora 9](#_ijrnf39zglr3)

[2.3.1 Arbitary čvor 11](#_gai6twhh1ui3)

[2.3.2 Hidden čvor 11](#_rsyfqj51sj19)

[2.3.3 Delayed čvor 12](#_1xcj272hye4)

[**3. Zaključak 13**](#_h5angz280nj8)

[**4. Reference 14**](#_zefgu78cyvus)

# 

# 

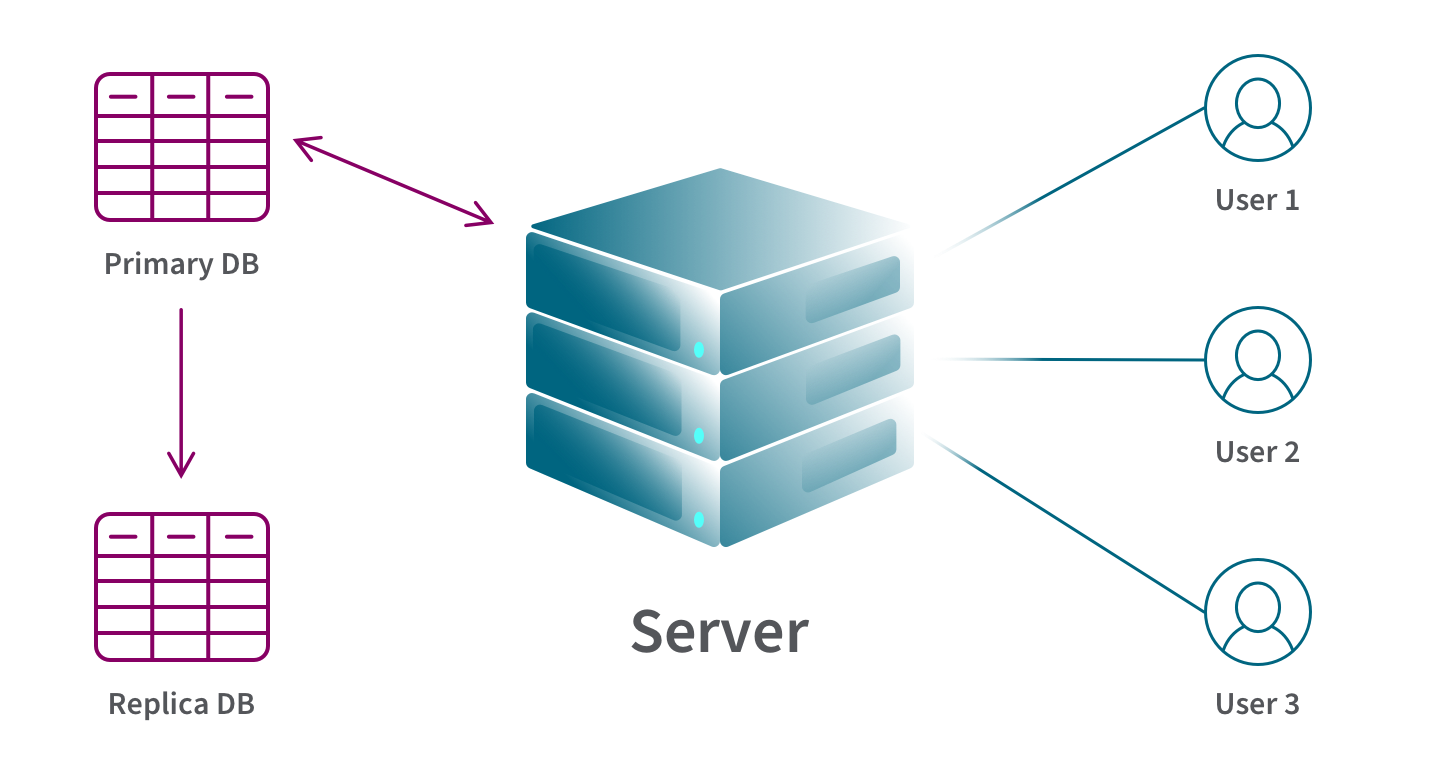
# 1. Uvod

Čovek je kroz istoriju uvek težio da napreduje i poboljša kvalitet života. Napredak ljudske vrste se često stavlja u isti kontekst sa napretkom tehnologije (koja je nesumnjivo jedna od ključnih stvari kada je evolucija u pitanju). Svedoci smo činjenice da se razvija sve brže i da su stvari, koje su nekad bile nezamislive i nedostižne, postale ostvarive i svakodnevne. Procesi koji su nekad oduzimali mnogo vremena i resursa, danas mogu biti izvršeni u mnogo kraćem roku bez naročite asistencije čoveka tokom samog procesa.

Razvojem interneta, protok i dostupnost informacija je postala veoma visoka. Danas se uz samo par klikova može kupiti željeni proizvod ili pročitati zakon jedne države. Transparentnost poslovanja i podataka neke organizacije je postala neminovnost vremena u kome živimo. Pored dobrih stvari koje transparentnost donosi, postoje i loše strane, poput zloupotrebe podataka i korišćenje istih u različite svrhe. Iako se svakodnevno radi na zaštiti korisnika, do danas nije nađeno univerzalno rešenje kojim bi se eliminisao bilo koji vid povrede ljudskih prava i zloupotrebe privatnih podataka.

Za mnoge sisteme, gubitak podataka, nedostupnost istih ili narušavanje njihove verodostojnosti bi predstavljao pravu katastrofu ukoliko se radi o velikoj količini podataka, prikupljanih i čuvanih duži vremenski period. Kako bi se izbegao ovakav scenario usled nekih nepredviđenih okolnosti kao što su otkaz memorijskog medijuma, zlonamerni proboj spolja, neautorizovani pristup podacima ili neka katastrofa koja može dovesti do narušavanja rada glavne baze. Kako bi sistem radio i u ovakvim slučajevima, jedan od načina za prevazilaženje ovakvog problema jeste rad sa “kopijom” baze, koja sadrži replicirane podatke, takozvane replike, za čije memorisanje su potrebni resursi, koji su neretko samo logički, virtuelno grupisani, gde se podaci zapravo nalaze na međusobno udaljenim, različitim mašinama (serverima). Replike mogu sadržati kompletan dataset (engl. *full* replike) ili mogu sadržati particionisane podatke(fragmente), gde se grupisanjem podataka sa većeg broja replika dobija ceo dataset.

Replikacija predstavlja proces sinhronizacije podataka između različitih instanci baze, koji se neretko nalaze na različitim fizičkim lokacijama, u okviru različitih data centara, čime se omogućava da aplikacije koje koriste podatke iz te baze rade bez problema i u slučaju katastrofe, da podaci budu dostupni sve vreme (engl. *High Availability*). U velikoj meri mogu doprineti skalabilnosti samog sistema, gde se kreiranjem većeg broja kopija baze postiže horizontalno skaliranje, čime se mogu rasteretiti i optimizovati određene akcije i procesi u okviru distribuiranih sistema.

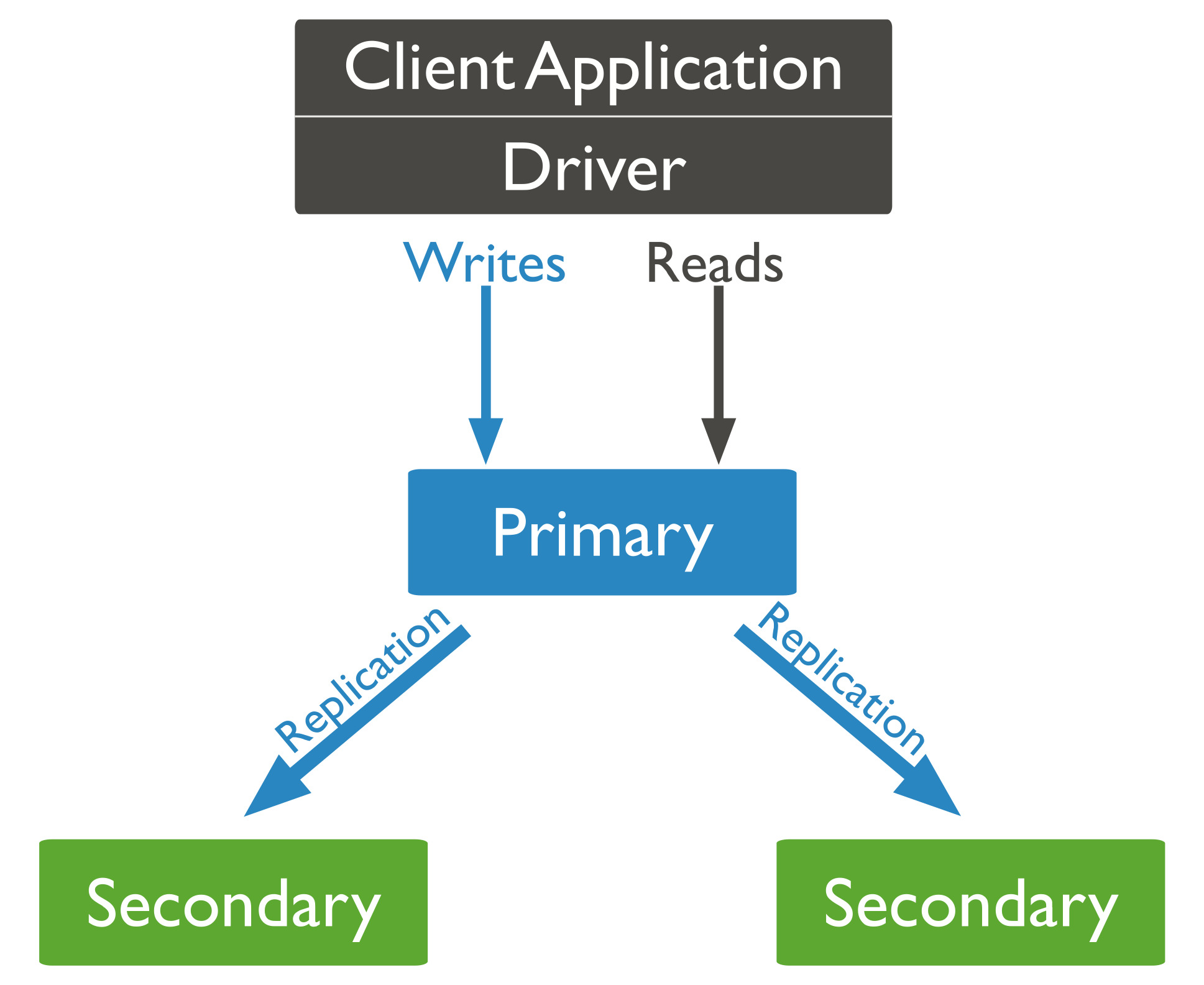


Slika 1. Replika

# 

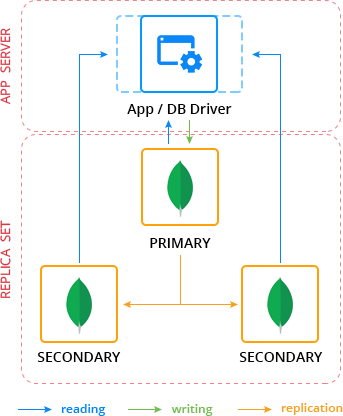
# 2. Replikacija kod MongoDB-a

Replikacija podataka kod MongoDB-a predstavlja grupu *mongod* procesa koji služe za održavanje i sinhronizaciju istog data seta, u okviru različitih instanci. Na ovaj način se smanjuje redundantnost podataka i povećava dostupnost istih, pa aplikacije koje koriste ove podatke mogu funkcionisati uprkos otkazu glavne instance. Takođe, moguće je i optimizovati rad same aplikacije, konkretno operacije čitanja, gde je moguće preusmeravati zahteve sa različitih lokacija na najbliže data centre u okviru kojih se nalaze kopije koje mogu opslužiti sam zahtev.



Slika 2. Replikacija MongoDB

Kod MongoDB-a, prilikom replikacije, kreiraju se takozvani replika setovi, koji se mogu sastojati iz većeg broja čvorova, gde se jedan i samo jedan čvor proglašava za glavni, takozvani primarni čvor, dok se ostali smatraju sekundarnim čvorovima.



Slika 3. MongoDB replika set

Kroz primarni čvor se odvijaju sve operacije upisa i primarni čvor beleži ove promene, kroz log fajl, gde nakon upisa, sekundardni čvorovi preuzimaju log fajl koji sadrži operacije (engl. *oplog*) i izvršavaju ove operacije čime se sve instance dovode u isto, konzistentno stanje. Same operacije su asinhrone, tako da nema blokiranja drugih, novih operacija.

Replika setove, kao i same čvorove u njima, je moguće konfigurisati, gde je moguće kroz konfiguraciju specificirati tagove kojima će neki od instanci/replika biti obeleženi, recimo područjem kojem pripadaju. Takođe, kroz konfiguraciju je moguće specificirati da li će određeni čvor biti uključen u razmatranje kao potencijalni primarni čvor i da li će učestvovati u potencijalnom glasanju.

Da bi kreirali replika set, potrebno je ugasiti trenutni MongoDB server, a potom ga pokrenuti uz pomoć flega *–replSet*:

mongod --port "PORT" --dbpath "DB\_PATH" --replSet "INSTANCE\_NAME"

nakon čega je moguće pristupiti samom setu i dodavati nove čvorove po želji (funkcijom *.add()*), proveriti joj status (*.status()*) ili je konfigurisati (*.conf()*).

## 2.1 Replika Set Konfiguracija

Za same replike postoji određena konfiguracija, koja predstavlja dokument u okviru koga je moguće specificirati različita podešavanja, koja mogu da utiču na kasnije ponašanje.

Kao i svaki dokument, sadrži polje \_id koje dobija vrednost imena instance(parametar prethodno opisane komande), gde pored ostalih polja poput verzije, protokol verzije i drugih, sadrži polje members koji predstavlja niz čvorova sa svojom konfiguracijom. Takođe, postoji polje settings, u okviru kog je moguće specificirati koji je prag tolerancije u pogledu odziva primarnog čvora, koliko često će međusobno kontaktirati sekundarne instance i drugo.Primer dokumenta:

{

\_id: <string>,

version: <int>,

term: <int>,

protocolVersion: <number>,

writeConcernMajorityJournalDefault: <boolean>,

configsvr: <boolean>,

members: [

{

\_id: <int>,

host: <string>,

arbiterOnly: <boolean>,

buildIndexes: <boolean>,

hidden: <boolean>,

priority: <number>,

tags: <document>,

secondaryDelaySecs: <int>,

votes: <number>

},

...

],

settings: {

chainingAllowed : <boolean>,

heartbeatIntervalMillis : <int>,

heartbeatTimeoutSecs: <int>,

electionTimeoutMillis : <int>,

catchUpTimeoutMillis : <int>,

getLastErrorModes : <document>,

getLastErrorDefaults : <document>,

replicaSetId: <ObjectId>

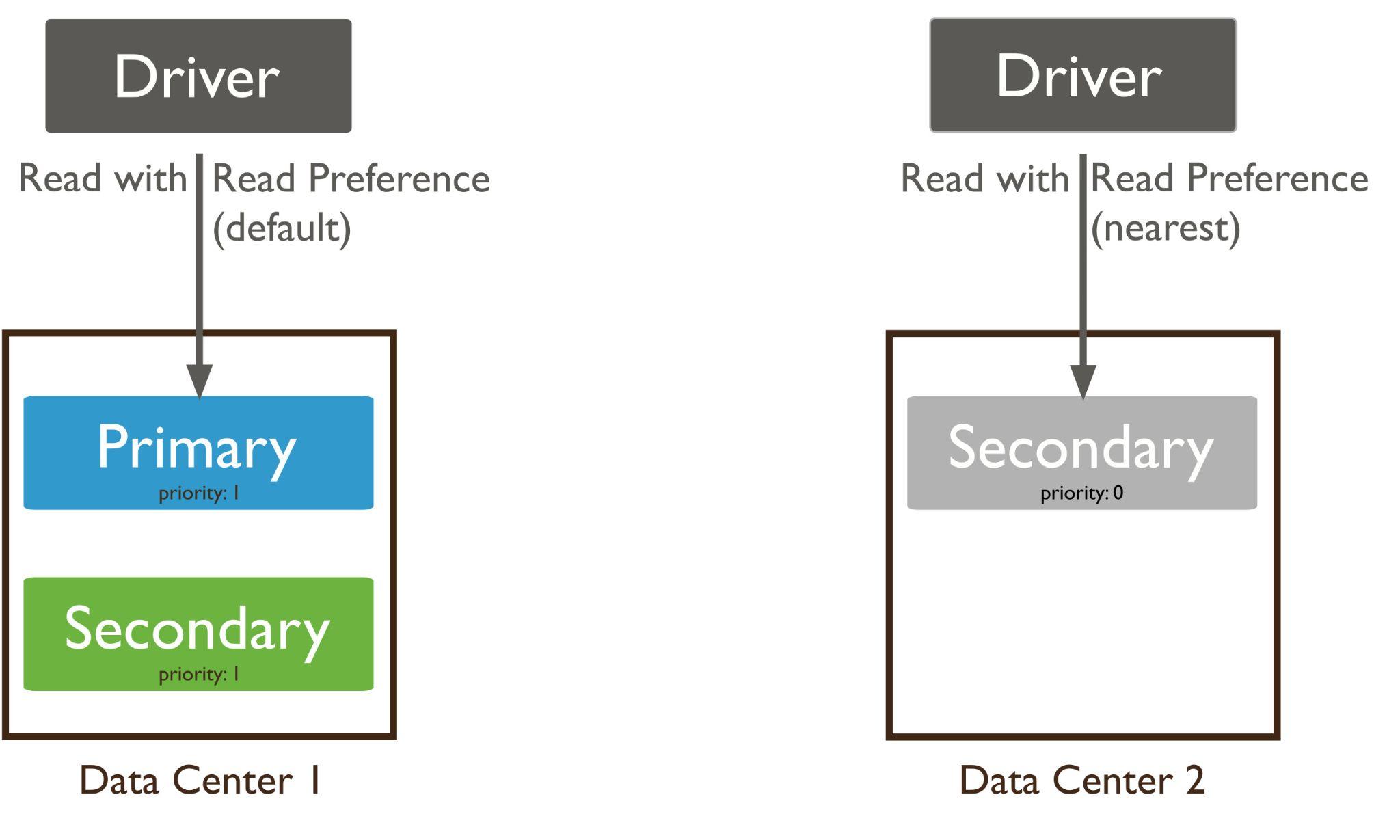
}

}

## 2.2 Čitanje podataka

Čitanje podataka se može vršiti sa različitih instanci, s obzirom da su podaci sinhronizovani, ali standardno podešavanje je da se i operacije čitanja obavljaju isključivo sa primarnog čvora. Ukoliko želimo da se podaci čitaju sa sekundarnih instanci, potrebno je namestiti konkretan mod čitanja:

* primary
  + Sve operacije čitanja idu sa primarnog čvora
* primaryPreffered
  + Čitanje se odvija sa primarne instance, ali ukoliko je nedostupna, čitanje ide sa sekundarne
* secondary
  + Sve operacije čitanja se odvijaju preko sekundarnih
* secondaryPreffered
  + Čitanje se odvija preko neke sekundarne instance, ali u slučaju da ne postoji trenutno dostupna sekundarna instanca, čitanje se odvija sa primarne
* nearest
  + Čitanje se odvija preko najbliže instance koja ispunjava postavljene zahteve, bez obzira na to da li se radi o primarnoj ili sekundarnoj instanci



Slika 4. Čitanje podataka

### 2.2.1 Komanda

Ukoliko želimo da omogućimo čitanje sa sekundarnih čvorova, to možemo ostvariti tako što ćemo nad konekcijom sa bazom pozvati komandu:

db.getMongo().setReadPref('secondary');

Pored prvog parametra koji predstavlja prethodno objašnjeni mod, moguće je proslediti jos dva opciona:

* tagSet
  + moguće specificirati tagove koji služe za identifikaciju čvorova sa kojih želimo da se izvrši čitanje. Nije moguće koristiti ovaj parametar ukoliko je prvi parametar komande 'primary'
* hedgeOptions
  + od verzije 4.4 moguć je ovaj tip hedged čitanja i uz pomoć ovog parametra moguće je specificirati

### 2.2.2 Primer

| db.getMongo().setReadPref( |
| --- |
| "secondary", |
| [ |
| { "datacenter": "B" }, |
| { "region": "West"}, |
| { } |
| ] |
| ) |

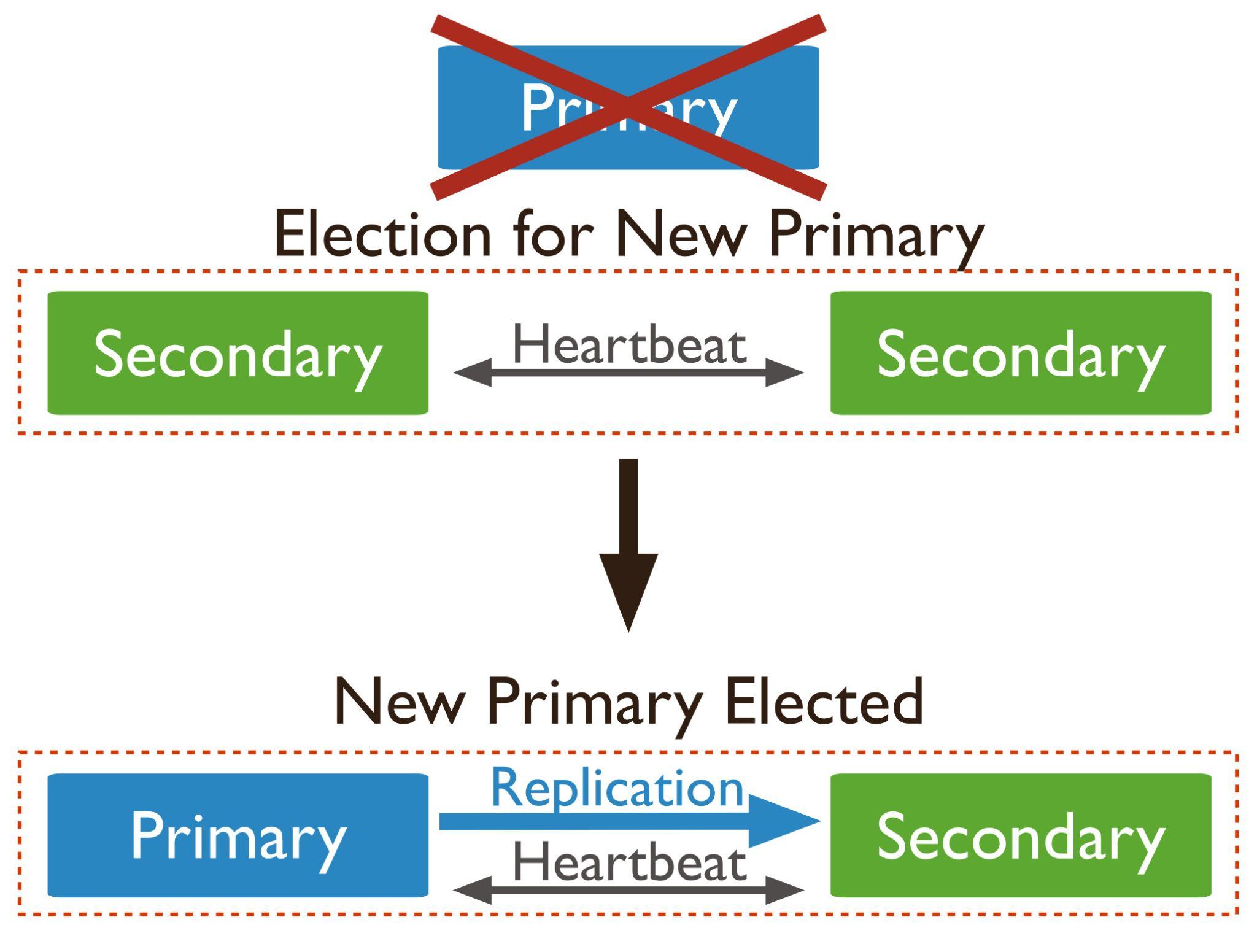
Kod ovog primera se navodi da čitanje treba izvršiti sa sekundarnog čvora, sa tagom “B” koji je deo konfiguracije replika seta, ukoliko postoji. U suprotnom se ide na drugi uslov, a to je polje “region” u ovom slučaju sa vrednošću “West”. Ukoliko ni tim uslovom nema preseka, navođenjem praznog objekta, mečuje se bilo koja sekundarna instanca.

## 2.3 Izbor primarnog čvora

Izbor primarnog čvora se može desiti u nekoliko situacija:

* Kreiran je novi replika set
* Otkaz primarnog čvora
* Dodavanje novog čvora u okviru replika seta
* Eksplicitno je zatražena rekonfiguracija replika seta
* Sekundarni čvorovi gube konekciju ka primarnom čvoru u intervalu većem od specificiranog promenljivom *electionTimeoutMilis* čija je podrazumevana vrednost postavljena na 10 sekundi

U trenutku kada primarni čvor ne reaguje duže od intervala specificiranog u okviru podešavanja *electionTimeoutMilis,* sekundardni čvor “saziva” reizbora novog primarnog čvora, nakon čega neki od sekundarnih postaje primarni i preuzima njegovu ulogu.



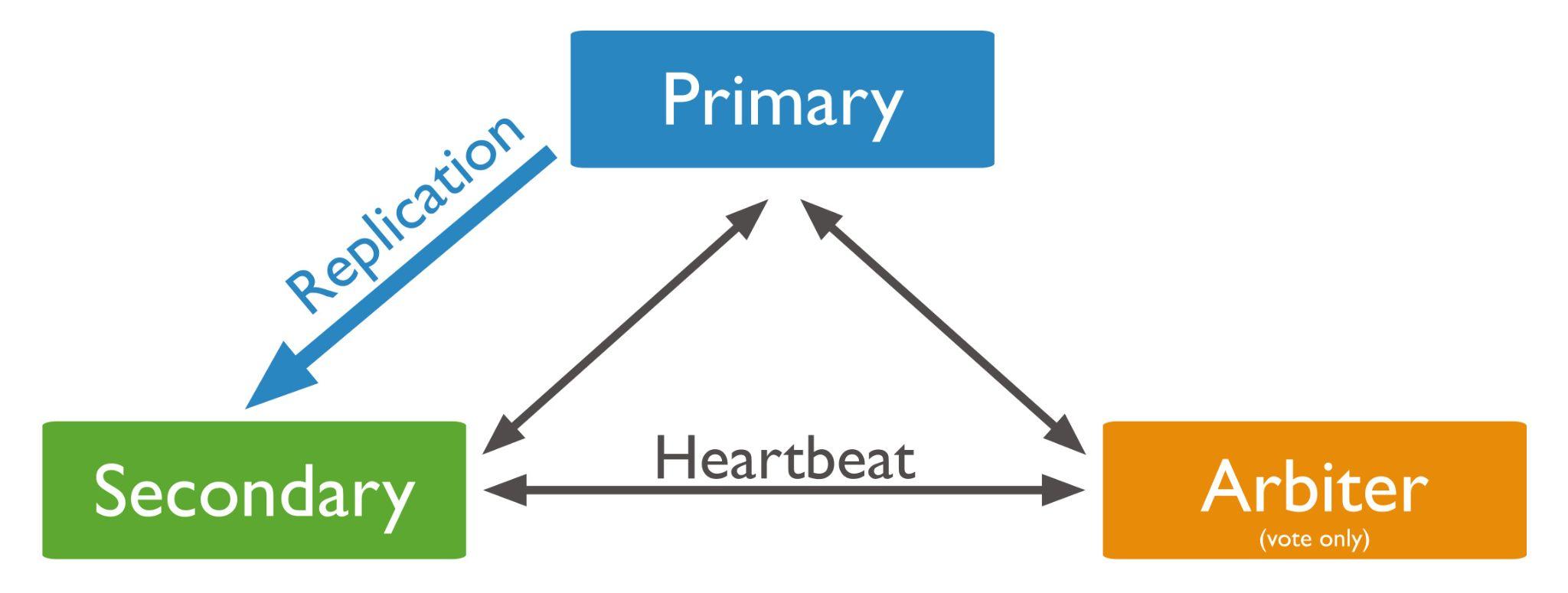
Slika 5. Otkaz primarnog čvora

Ovim mehanizmom omogućen je automatski oporavak od otkaza primarne instance. Tokom trajanja izbora nove primarne instance, upisi su onemogućeni. Trajanje reizbora primarne instance zavisi od nekoliko faktora:

* Verzije replika set protokola
* Kašnjenje kroz mrežu
* Prioritet čvorova
  + čvorovi koji imaju prioritet 0, ne mogu postati novi primarni
* *Mirrored* čitanja (od verzije 4.4), predstavljaju tehniku pripremanja sekundarnih instanci na operacije i podatke koji su u skorijoj prošlosti korišćene, kako bi oporavak bio brži u slučaju otkaza
* Gubitak data centra
* “Glasači” koji učestvuju u izboru čvora
  + Čvorovi koji imaju vrednost 0 u okviru polja *votes* i *priority*, ne učestvuju u glasanju

### 2.3.1 Arbitary čvor

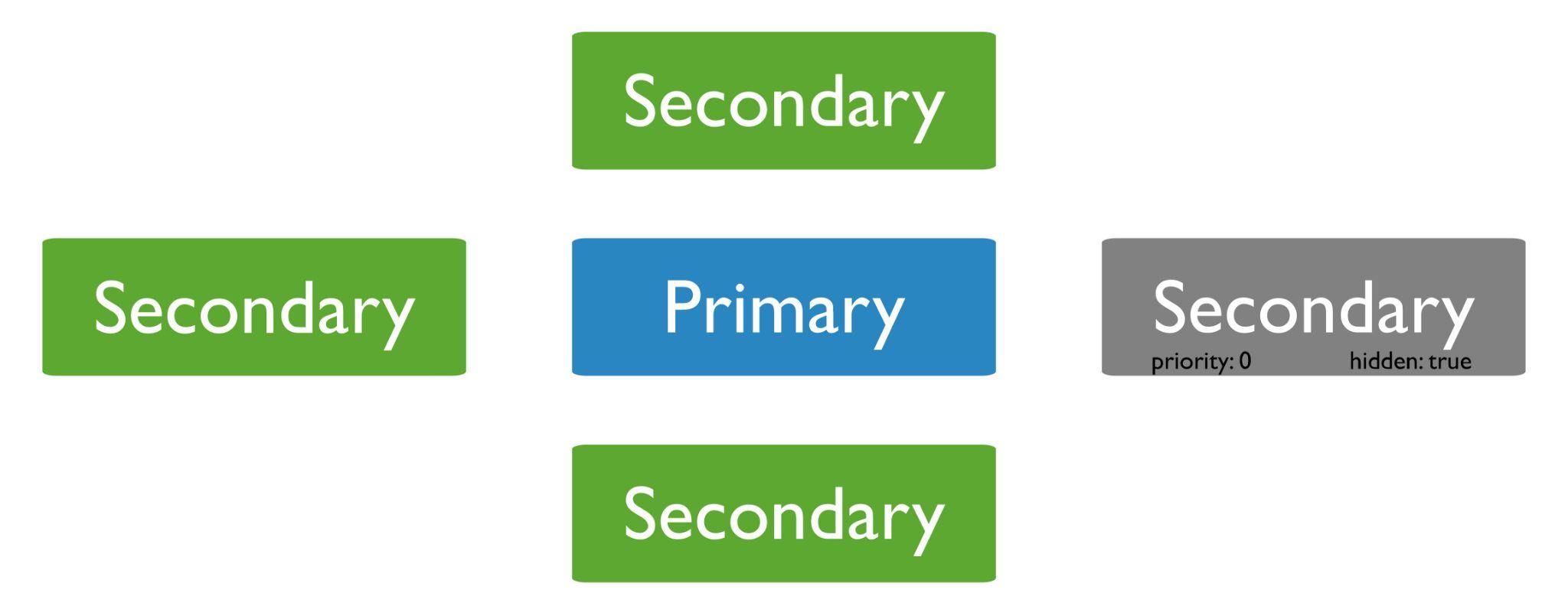
U situacijama kada je skupo dodavati nove sekundarne čvorove, a potrebno je uticati na izbore i broj učesnika, zbog raznih problema koji se mogu javiti, kao što je jednak broj sekundarnih čvorova koji mogu glasati ili postojanje samo jednog sekundarnog čvora, moguće je dodati instancu koja samo učestvuje u glasanju, koja ne sadrži podatke i nema tendenciju da promeni stanje, da pređe u sekundardni ili primarni mod, takozvani Arbitary čvor.



Slika 6. Arbitary čvor

### 2.3.2 Hidden čvor

Sekundarni čvorovi, koji mogu učestvovati u glasanju, a koji mogu sadržati podatke ali istovremeno ne izvršavati operacije čitanja zahtevanih od strane klijenta, zovu se sakriveni(engl. *hidden*) čvorovi. Iz ugla klijenta, ti čvorovi nisu vidljivi, ali iz ugla replika seta, oni postoje i imaju uticaj na izbor novih primarnih čvorova, pri čemu oni ne mogu postati primarni(*priority* im je 0). Oni se koriste kao backup čvorovi ili za generisanje reportova recimo, uz pomoć svih podataka koje dobijaju i ostali čvorovi.

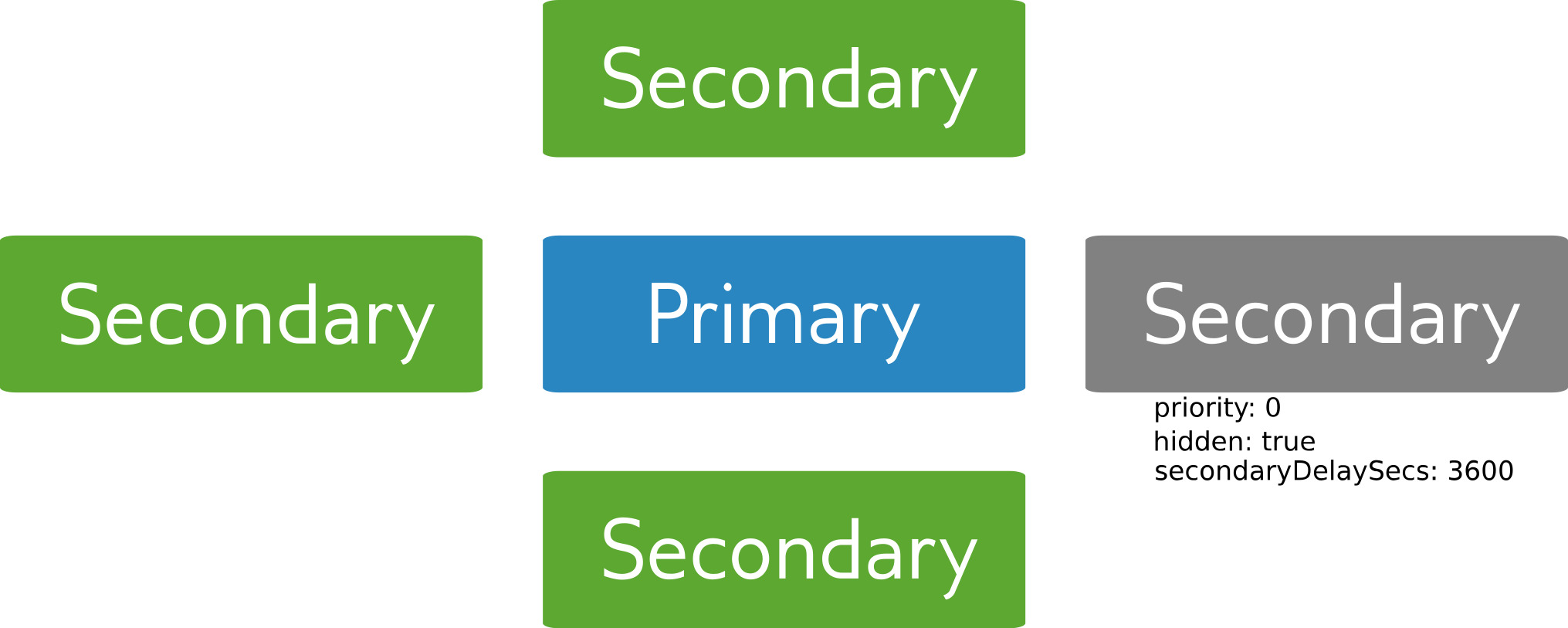


Slika 7. Hidden čvorovi

### 2.3.3 Delayed čvor

Delayed ili prevedeno, odloženi čvorovi, predstavljaju čvorove kod kojih podaci “kasne” u odnosu na druge čvorovi. Kod njih se operacije izvršavaju sa zakašnjenjem, pa kao takvi služe za brz oporavak ukoliko je došlo do neke greške prilikom unosa podataka. Predstavljaju svojevrsne tačke u prošlosti na koje je moguće vratiti se ukoliko se neka operacija nije okončala kako je bilo predviđeno. Ukoliko je odložen na 1 sat i trenutno vreme je 07:34, onda će taj čvor sadržati kao poslednju operaciju, onu koja se desila u 06:34.

Ovi čvorovi moraju biti sakriveni(engl. *hidden*) i ne mogu postati primarni. Mogu učestvovati u glasanju, u zavisnosti od vrednosti *votes* polja (ukoliko je 1, učestvuju).



Slika 8. Delayed

| { |
| --- |
| "\_id" : <num>, |
| "host" : <hostname:port>, |
| "priority" : 0, |
| "secondaryDelaySecs" : <seconds>, |
| "hidden" : true |
| } |

# 

# 3. Zaključak

MongoDB kao jedna od popularnijih baza podataka danas, pored ostalih funkcionalnosti, pruža podršku za kreiranje replika podataka, pri čemu se u velikoj meri doprinosi redudantnosti podataka i njihovoj dostupnosti u specijalnim, nepredvidljivim okolnostima. Zbog fleksibilnosti koju pruža, kao i mogućnosti za definisanjem različitih tipova čvorova, predstavlja odličan izbor u slučajevima gde je bitna skalabilnost, redudantnost podataka, kao i optimizacija operacija čitanja gde je moguće izvršiti ravnomernu raspodelu operacija koje je potrebno izvršiti na različitim instancama.

# 

# 4. Reference

[1] “MongoDB Official ” <https://www.mongodb.com/docs/manual/replication/>

[2] “Replica set Arbiter” <https://www.mongodb.com/docs/manual/core/replica-set-arbiter/>

[3] ”MongoDB Replication” <https://www.tutorialspoint.com/mongodb/mongodb_replication.htm>