

Univerzitet u Nišu, Elektronski fakultet

Katedra za računarstvo

Niš, 2021

Danilo Vulović

Seminarski rad

Backup/Restore Neo4j baze podataka

**Mentor**: **Student**:

Prof. dr Aleksandar Stanimirović Danilo Vulović 1065

#### Sadržaj

[1. Uvod 4](#_Toc72696277)

[2. SQL i NoSQL baze podataka 6](#_Toc72696278)

[2.1. Osnove SQL-a 6](#_Toc72696279)

[2.2. Osnove NoSQL-a 6](#_Toc72696280)

[2.3. Glavne razlike, prednosti i mane 6](#_Toc72696281)

[2.3.1. Struktura 7](#_Toc72696282)

[2.3.2. Jezik za upite 8](#_Toc72696283)

[2.3.3. Skalabilnost 8](#_Toc72696284)

[2.3.4. Podrška 9](#_Toc72696285)

[3. Osnove graf i Neo4j baze podataka 11](#_Toc72696286)

[3.1. Baze podataka bazirane na grafovima 11](#_Toc72696287)

[3.2. Neo4j 12](#_Toc72696288)

[3.2.1. Glavne odlike Neo4j baze 13](#_Toc72696289)

[3.2.2. Neo4j upotreba 13](#_Toc72696290)

[4. Backup/Restore Neo4j baze podataka 15](#_Toc72696291)

[4.1. Backup i restore planiranje 15](#_Toc72696292)

[4.1.1. Backup i restore strategija 15](#_Toc72696293)

[4.1.2. Backup i restore opcije 15](#_Toc72696294)

[4.1.3. Dodatne stavke 17](#_Toc72696295)

[4.2. Backup modovi 17](#_Toc72696296)

[4.3. Backup i restore online baze 17](#_Toc72696297)

[4.4. Backup i restore offline baze 19](#_Toc72696298)

[5. Zaključak 21](#_Toc72696299)

[6. Literatura 22](#_Toc72696300)

# 

# Uvod

Baza podataka predstavlja skup, na neki logičan način, povezanih podataka. Omogućava korisniku da efikasno pribavi, uneti nove ili obriše podatke. Podaci su organizovani u tabele, poglede, šeme itd. Na primer, baza podataka Elektronskog fakulteta u Nišu void računa o studentima, profesorima i asistentima, predmetima, nastavnim programima, rokovima itd.

Možda se “na prvu loptu” ne vidi prednost ovakvih sistema ali pogledajmo sledeći primer. Zamislite da koristite standardan fajl sistem za gore pomenutu fakultetsku bazu. Neki od problema na koje nailazimo su:

* Redundantnost podataka - za reduntantne podatke kažemo da su takvi ukoliko su kopirani na više mesta
  + podaci o istom studentu će se nalaziti u više sekcija
  + ažuriranje broja telefona studenta zahteva da broj bude ažuriran u svim sekcijama
  + brisanje studenta iz sekcije sa studentima će ostaviti tragove tog studenta u ostalim sekcijama koji takođe moraju biti obrisani
* Nekonzistentnost podataka - za podatke kažemo da su nekonzistentni ukoliko nekoliko kopija istog podatka nisu iste
  + ako broj telefona studenta u studentskoj službi i u administraciji nije isti, dolazi do nekonzistentnosti podataka
  + do nekonzistentnosti se može doći raznim programskim greškama kao i ako sve kopije podatka nisu ažurirane
* Otežan pristup podacima – pristup podacima o određenom studentu može biti izuzetno komplikovan.
  + zamislite da imate 10.000 studenata i trebate naći podatke o nekolicini koji uspunjavaju određene uslove.
* Neovlašćen pristup – korisnik pristupa podacima koji nisu njemu namenjeni
  + Student bi mogao da pristupi sekciji sa ocenama i promeniti ih
* Nemogućnost oporavka od greške ili kreiranja rezervnih kopija
* ...

Ovo su samo neki od problema koji su naterali da se obavi skok sa fajl sistema na nešto ozbiljnije.

Sistemi za upravljanje bazama podataka, kao i sama baza, predstavljaju srce svakog kompjuterskog programa. Oni predstavljaju skup programa koji zajedno rukovode bazom podataka i omogućavaju krajnjim korisnicima da izvršavaju operaciji nad bazom. Korisniku je omogućeno da na jednostavan način definiše kako izgledaju podaci koje baza podataka skladišti, kako se ti podaci memorišu, pružaju korisniku jednostavan način za manipulisanje podacima u bazi, štite podatke od bilo kakve vrste sistemske greške ili krađe i omogućavaju različite privilegije različitim korisnicima.

Skoro svaki sistem za upravljanje bazama podataka je (u jednom ili drugom obliku) sastavljen iz sledeće tri komponente:

* Procesor upita
* Sistem za upravljanje skladištem podataka
* Skladište na disku

Oni rade u koheziji da obezbede visoku funkcionalnost čitavog sistema, da omoguće izvesnu vrstu fleksibilnosti ukoliko je neki koncept baze podataka potrebno izmeniti kao i visoke performanse uz visok procenat bezotkaznog rada a lakoća korišćenja podrazumeva da je korisniku omogućena automatizacija raznih zadataka.

Sve ovo su samo neki od zahteva koji se očekuju od jednog sistem za upravljanje bazama podataka. U ostatku rada, bavićemo se razlikom SQL i NoSQL baza podataka (drugo poglavlje), analiziraćemo kako izgleda Neo4j baza podataka i koje su njene glavne prednosti i mane (treće poglavlje), kakve su koristi čuvanja rezervnih kopija Neo4j baze podataka, šta treba uzeti u obzir kada odlučujemo o strategiji čuvanja kopija, šta je sve neophodno sačuvati da bi se baza adekvatno oporavila i različite načine i opcije za čuvanje kopija (u daljem tekstu „Backup“) i obnovu sačuvanih podataka (u daljem tekstu „Restore“) (četvrto poglavlje) kao i da odgovorimo na pitanje na koji način je najoptimalnije čuvati kopije podataka u Neo4j bazi podataka (peto poglavlje). Poslednje poglavnje sadrži literaturu koju sam upotrebio za pisanje ovog rada (šesto poglavlje).

# SQL i NoSQL baze podataka

## Osnove SQL-a

Kada kažemo SQL baza podataka, obično mislim na relacionu bazu podataka. Podaci su logički organizovani u tabele. SQL (*Structured query language*) služi da omogući korisniku jednostavnu manipulaciju podacima u bazi podataka. To je veoma moćan alat i koncepti relacionih baza podataka su detaljno istraženi i njihova efektivnost je dokazana. Pisanje kompleksnih upita je izuzetno jednostavno i lako možete naći eksperta za rad sa bazama podataka. Ono što SQL zahteva je da postoji predefinisana (statička) šema podataka i svi podaci u bazi moraju da prate tu šemu. Ovo može značajno da poveća vreme planiranja strukture baze jer bilo kakve naknadne promene mogu biti teške ili ometati ostatak sistema. Neke od najpopularnijih baza su:

* MySQL
* Microsoft SQL server
* SQLite
* Oracle
* ...

## Osnove NoSQL-a

NoSQL baze podataka, kao što im ime kaže, ne koriste SQL[[1]](#footnote-1), odnosno, šema im je dinamička i moguće je pamtiti nestrukturne podatke. Podaci koji se pamte mogu da budu dokument orijentisani, orijentisani kolonama, bazirani na grafovima ili da budu predstavljeni KeyValue skladištem. Svaki dokument može biti struktuiran na sebi svojstven način i tu strukturu je moguće promeniti u bilo kom trenutku. Neke od najpoznatijih NoSQL baza podataka su:

* MongoDB, Elasticsearch, ... (dokument orijentisane)
* MariaDB, Cassandra, ... (kolonama orijentisane)
* Neo4j, Amazon Neptune, ... (graf baze)
* Dynamo, Apache Ignite, ... (KeyValue skladišta)
* ...

## Glavne razlike, prednosti i mane

Glavne razlike SQL i NoSQL baza podataka su date u sledećoj tabeli:

|  |  |  |
| --- | --- | --- |
| Ključne razlike | SQL baze podataka | NoSQL baze podataka |
| Tip baze podataka | Relaciona | Nerelaciona |
| Šema | Predefinisana (statička) | Dinamička |
| Kategorije baze podataka | Bazirane na tabelama | Dokument baze, KeyValue skladišta, graf baze... |
| Kompleksni upiti | Pogodna | Nezgodne |
| Skalabilnost | Vertikalno skalabilne | Horizontalno skalabilne |
| Svojstva | ACID[[2]](#footnote-2) svojstva | CAP[[3]](#footnote-3) teorema |
| Podrška | Odlična, od strane svih provajdera | Uglavnom se oslanja na podršku zajednice |

*Tabela 1 – Glavne razlike SQL i NoSQL baza podataka*

U nastavku ćemo pomenuti neke razlike da bi napravili jasnu granicu između SQL i NoSQL baza podataka.

### Struktura

Statička šema zahteva da struktura podataka koji se čuvaju bude unapred definisana i da svi podaci mogu da se „ukalupe“ u okvire definicije šeme. Dinamička šema, kao što joj ime kaže, je podložna promenama u bilo kom trenutku rada. Razmotrimo primer da je za evidentiranje vakcinisanja potreban način identifikacije svakog korisnika. Prvo što nam pada na pamet je da možemo za to iskoristiti jedinstveni matični broj građana. U SQL bazi bi ovo mogla biti kolona koja sadrži niz cifara JMBG-a.

Dobili smo zahtev da nije potreban JMBG građana da bi se vakcinisao već je moguće podneti zahtev vozačkom dozvolom ili zdravstvenom karticom. Ovo zahteva da se promeni SQL šema i da se sama identifikacija izmesti u zasebnu tabelu gde će biti nekoliko kolona za svaki od načina identifikacije gde će biti dovoljno da jedna od kolona bude popunjena za svaki red. NoSQL baza ne mora da primi nikakve promene, prihvatiće drugačiju identifikaciju kao i opcione dodatne podatke o tome.

Dobili smo zahtev da je moguće vakcinisati i strane državljane ali svaka država ima svoj način za jedinstvenu identifikaciju građana. Nije izgledno dodati zasebnu kolonu za svaku državu na svetu a ne bi bilo pametno oslanjati se na brojeve pasoša jer oni ne garantuju da ne postoji još jedan pasoš na svetu sa istim brojem. Morali bi osmisliti neku vrstu kompozitnog ključa broja pasoša i države iz koje strani državljanin. NoSQL neće imati problem da primi identifikaciju drugačijeg oblika.

Primer je možda suviše banalan ali dovoljno opisuje problem sa kojim se statička baza suočava i zašto su u prethodnoj deceniji NoSQL baze naglo dobile na popularnosti i zašto NoSQL tehnologije imaju predviđen rast od 56% po godini za razliku od SQL baza koje takođe očekuju rast ali od svega 12%.

### Jezik za upite

SQL baze koriste SQL za definisanje i manipulisanje podacima i jezik je sam po sebi, izuzetno moćan ali i restriktivan. Zahteva definisane strukture podataka pre početka rada i svi podaci moraju da slede definiciju što zahteva preciznost i pedantnost.

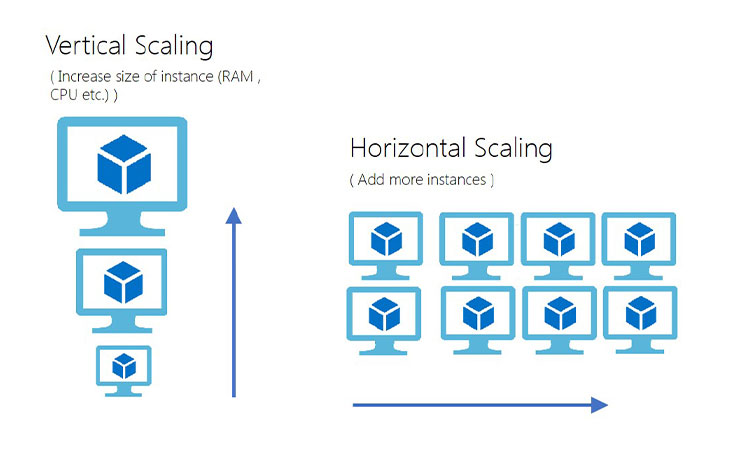
NoSQL baze su fleksibilnije i dozvoljavaju dodavanje polja u bilo kom trenutku i jedinstvenu strukturu za svaki podatak. Eliminisanje planiranja i definisanja strukture omogućava spremnost za rad u rekordnom roku ali baratanje i validacija podataka mora biti odrađena ručno. Mnoge NoSQL baze koriste jezike koji su nalik na SQL-u (npr. Neo4j koristi Cypher) kako bi programerima tranzicija bila što prirodnija i korišćenje jednostavno i prepoznatljivo. Takođe, neke NoSQL baze podržavaju SQL upite uz određenu cenu pretprocesiranja takvog upita.

### Skalabilnost

Kako količina podataka koje skladištimo raste, sasvim je prirodno razmišljati o skaliranju kako bi sistem podneo veći teret. U zavisnosti od toga da li koristimo SQL ili NoSQL bazu, tehnika skaliranja se fundamentalno razlikuje.

Vertikalno skaliranje podrazumeva poboljšanje performansi servera koje su postignute poboljšavanjem hardvera. Ovo uključuje veću RAM memoriju, brži CPU, korišćenje bržeg SSD diska itd. Iako se ovo čini dosta logično i razumno, postoji jedan „ogroman“ problem. Postoji limit za ovakva poboljšanja jer možete kupiti trenutno najbrži procesor na tržištu i tu se CPU poboljšanja završavaju. Takođe, kupovina najnovijeg hardvera je dosta skupa. Ovakvo skaliranje je karakteristično za SQL baze podataka.

Horizontalno skaliranje se postiže povećanjem broja servera. U teoriji, možemo dodati beskonačan broj servera koji paralelno rade. S obzirom da su instance servera distribuirane, možemo čuvati više podataka ali uvodimo probleme distribuiranog sistema. NoSQL baze podataka se prirodno skaliraju na ovaj način obično nude mehanizme za rešavanje ovih problema.



Slika 1 – Vertikalno i horizontalno skaliranje

Koncept *sharding-a* (podela na manje delove) je ključan za skaliranje i odnosi se i na SQL i na NoSQL baze. Kao što ime kaže, sharding deli bazu na više delova. Svaki deo ima jedinstven indeks kojim se pretražuju podaci i na osnovu koga možemo podeliti podatke na više delova (npr. možemo podeliti bazu na dva dela gde se u prvom delu nalaze ljudi čije prezime počinje sa nekim od slova od A do M, dok se u drugom delu nalaze ljudi čija prezimena počinju od N do Š). SQL baze ne podržavaju automatsko deljenje na delove i inženjeri moraju ručno da brinu o sharding-u. Vrlo često je izuzetno teško uraditi sharding jer promene strukture, koje su svakako vremenski skupe, zahtevaju i izmene shardova. Sa druge strane, NoSQL baze su inicijalno dizajnirane sa skaliranjem na umu, tako da je sam proces skaliranja, odnosno shardinga, skoro neprimetan za krajnjeg korisnika.

### Podrška

SQL ima veliku istoriju i mnogo razvijeniju zajednicu. Zasniva se na matematičkom modelu, postoje brojne najbolje prakse za razne slučajeve korišćenja i postoji veliki broj ljudi koji su eksperti. SQL jezik se sam po sebi ne razlikuje puno kod različitih baza podataka izuzev nekih finesa a provajderi baza podataka garantuju za pouzdanost same baze.

NoSQL baze su relativno mlade u poređenju sa SQL ekvivalentima pa stoga ne postoji jaka podrška provajdera već se programeri uglavnom oslanjaju na podršku zajednice. Kao i broj NoSQL korisnika, podrška rapidno raste i možemo očekivati da će u nekom trenutku biti standardizovane na zavidnijem nivou.

# Osnove graf i Neo4j baze podataka

## Baze podataka bazirane na grafovima

Najkraće moguće rečeno, graf baza je baza koja je dizajnirana tako da relacijama između čvorova pridaje jednaku važnost kao i samim čvorovima, odnosno podacima. Podaci mogu da se čuvaju bez pre-definisanog modela. U realnom svetu, pa i u onom sistemu koji modeliramo na osnovu realnog sveta, ne postoji informacija koja je sama po sebi izolovana i nema nikakvu relaciju ni sa čim. Čak suprotno, oko nas se nalaze domeni koji su vrlo bogato povezani. Samo baza koja ove veze prirodno prihvata, može jednostavno da ih obrađuje, procesira i pretražuje. SQL baze koriste vremenski skupocene JOIN operacije dok se u graf bazama, relacije između entiteta čuvaju odmah uz njih.

Pristup čvorovima i njihovim relacijama u graf bazi se može radi efikasno jer se relacijama pristupa za konstantno vreme i moguće je obići milione relacija u sekundi na svakom jezgru. Nezavisno od ukupne veličine skupa podataka sa kojim se radi, graf nadmašuje standardne SQL baze kada su podaci u skupu podataka visoko povezani.

A picture containing accessory

Description automatically generated

Slika 2 – Vizuelizacija Neo4j baze koja prikazuje nekoliko filmova u kojima je glumio Tom Henks kao i reditelje tih filmova

Postoje nekoliko različitih pristupa pri modeliranju ovakvih graf baza. U narednom delu teksta, skoncentrisaćemo se na model grafa koji nam govori da su podaci organizovani u čvorove, relacije između čvorova i iz svojstava čvorova i relacija.

**Čvorovi** su entiteti grafa. Oni mogu da čuvaju beskonačno mnogo atributa (obično Key-Value parovi) koje kratko zovemo, svojstva. Čvorovi mogu da budu tagovani labelama, koje mogu da posluže kao neka vrsta metapodatka.

**Relacije** su usmerene i semantički bitno imenovane putanje između čvorova (npr. Radnik *RADI\_U* Fabrika; *RADI\_U* je relacija). Relacija uvek ima smer, tip, početni čvor i krajnji čvor. Kao i čvorovi, i relacije mogu imati pridodata svojstva (npr. relacija iz prethodnog primera može da ima svojstvo { *RadiOd:* 2017 *},* što znači da radnik radi od 2017. godine u fabrici). U većini slučajeva, relacijama se pridodaju kvantitativna svojstva kao što su težine, cene, distance, vremenski intervali i sl. Zbog efikasnog načina na koji se relacije čuvaju, dva ista čvora mogu imati neograničen broj relacija između sebe bez ikakvog gubitka u performansama. Važno je pomenuti da su relacije usmerene ali da to ne znači da je nemoguće navigirati se kroz graf u bilo kom smeru (npr. graf će jednako brzo naći relaciju A -> B kao i B <-A.

Chart, bubble chart

Description automatically generated

Slika 3 – Primer čvorova i relacija u grafu

## Neo4j

Neo4j je NoSQL graf baza, otvorenog koda koja obezbeđuje aplikacijama „bekend“ sa transakcijama koje su u skladu sa ACID svojstvima. Baza je pisana u Javi i Scali i razvoj je počeo 2003. godine ali je od 2007. kod javno dostupan. Dolazi u dve edicije, *Community* verzija koja je besplatna, dostupna svima i *Enterprise* verziju koja nudi sve što nudi i *Community* uz razne dodatne pogodnosti poput čuvanja rezervnih kopija baze, oporavka od grešaka kao i podršku za klasterizaciju.

Za Neo4j se kaže da je „native“ graf baza jer efikasno implementira prethodno pomenuti graf model sa svojstvima sve do nivoa skladišta. Podaci su čuvani tačno onako kako se i koriste, a baza koristi pokazivače za navigaciju kroz graf. Za razliku od graf procesiranja ili biblioteka koje simuliraju graf baze kroz *in-memory* model, Neo4j obezbeđuje sve karakteristike normalne baze podataka, transakcije koje su saglasne sa ACID svojstvima, podršku za klasterizaciju i oporavak od grešaka što olakšava korišćenje graf modela čak i u proizvodnji.

### Glavne odlike Neo4j baze

Diagram

Description automatically generated

Slika 4 – Glavne odlike Neo4j baze

Neke od glavnih odlika Neo4j baze zbog koji je ova baza vrlo popularna među programerima, arhitektama i administratora baza podataka su:

* **Cypher**, deklarativni jezik za upite sličan SQL-u ali optimizovan za graf baze. Ovaj jezik je usvojen i od strane nekih drugih graf baza.
* **Konstantno vreme obilaska**, vreme obilaska grafa se ne povećava drastično sa povećanjem broja čvorova. Na modernism hardverima, broj čvoroma može biti i nekoliko milijardi (upit koji vraća ukupan broj čvorova u grafu koji iznosi preko 31 milion čvorova traje manje od 1ms).
* **Fleksibilnost**, šemu je moguće menjati i adaptirati vremenom, ili dodati nove relacije između određenih entiteta za koje se ranije nije znalo i koje mogu ubrzati pretragu
* **Drajveri**, za sve popularne jezike poput Jave, Pythona, Javascripta, .NET-a...

### Neo4j upotreba

Prva stvar koja deluje kao logičan scenario za korišćenje Neo4j baze su društvene mreže. Iako to jeste jedna od situacija gde broj relacija nadmašuje broj čvorova, to je samo jedan slučaj upotrebe koji ne opisuje dovoljno i ne pruža uvid u pun potencijal ove graf baze.

* Detekcija prevara

Jednostavno detektuje kada je, recimo, određen čvor koristio veliki broj kreditnih kartica u kratkom perioudu što može da

* Sistemi za preporuke u realnom vremenu
* Rutiranje u mreži
* Menadžment „master“ podacima
* Pretrage bazirane na grafu
* ...

Ovi „use-case“-ovi ne opisuju dovoljno šta sve Neo4j može da radi bolje od drugih NoSQL i SQL rešenja ali daje dobru osnovu za razmišljanje koji sve problemi za koje je potrebna napredna biznis logika i složeni SQL upiti, su skoro rešeni samom činjenicom da se koristi graf baza poput Neo4j.

# Backup/Restore Neo4j baze podataka

## Backup i restore planiranje

Postoje dva glavna razloga za čuvanje rezervnih kopija podataka u Neo4j bazi:

* da bi povratak podatak bio jednostavna u slučaju otkaza bilo da je on izazvan hardverskom greškom, prirodnom katastrofom ili ljudskom greškom
* da bi se rutinske administrativne operacije, poput pomeranja podataka na neku drugi instancu, svele na nekoliko klikova

### Backup i restore strategija

U zavisnosti od okruženja i postavke softvera kakva vam treba, bitno je da se osmisli prikladna strategija za backup i restore strategija. Postoji pregršt faktora koje trebamo uzeti u obzir pri odabiru strategije. Pomenućemo neke u nastavku:

* Tip okruženja (proizvodnja, testiranje, razvoj...)
* Količnina podataka sa kojom se radi
* Broj baza podataka
* Dostupni sistemski resursi
* Vreme zastoja prilikom backup i restore operacija
* Redukovana performansa Neo4j baze tokom backup i restore operacija (bolje je odraditi backup i restore u intervalu kada obično nema puno korisnika baze)
* Tolerancija na gubitak podataka u slučaju otkaza
* Tolerancija na zastoj u slučaju otkaza
* Učestalost ažuriranje podataka
* Broj rezervnih kopija koji želimo da čuvamo
* Lokacija backup-ova (lokalno skladište, skladište u oblaku, data centri…)
* Učestalost provere backup rutina
* …

### Backup i restore opcije

Neo4j podržava backup i restore i za online i offline baze. Koristi *Neo4j Admin tool* komande koje mogu da se pokrenu i na instanci koja je podignuta i na isntanci koja je offline.

Mnogi od ugrađenih alata za backup i restore su dostupni samo na Enterprise verziji koja je prilično skupa čak i za velike kompanije. U daljem tekstu će biti specijalno naglašeno ukoliko je bilo šta dostupno isključivo za Enterprise verziju.

* **neo4j-admin backup/restore** (*Enterprise only*) – koristi se za kreiranje online backup-ova (bilo potpunih ili inkrementalnih[[4]](#footnote-4)) i restore-ova. Ova komanda je pogodna za korišćenje u proizvodnji, gde se ceni što manji zastoj ali je zato izuzetna memorijski zahtevna
* **neo4j-admin dump/load –** koristi se za kreiranje offline dump i load operacija. Ova komanda je zgodna za okruženja gde vremenski zastoj nije bitan. Operacije su brže od korespodentnih backup i restore operacija i operacija dump kreira arhivni fajl koji je dosta manji od normalne strukture Neo4j baze
* **neo4j-admin copy –** koristi se za kopiranje offline baze ili backup-a. Ova komanda može da se koristi za čišćenje baze od nekonzistentnih podataka, povratak memorije ili za migraciju Neo4j na neku drugu verziju (npr. sa 3.5 na 4.1) ili na najnoviju verziju, preskačući verzije između

Naredna tabela opisuje komande i njihovo korišćenje:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Korišćenje | **neo4j-admin backup** | **neo4j-admin dump** | **neo4j-admin restore** | **neo4j-admin load** | **neo4j-admin copy** |
| Neo4j edicija | Enterprise | Sve edicije | Enterprise | Sve edicije | Sve edicije |
| Online Neo4j DBMS | Da | Da | Da | Da | Da |
| Potpuni backup | Da | Da | / | / | / |
| Inkrementalni backup | Da | Ne | / | / | / |
| Primenljiva na online bazi | Da | Ne | Ne | Ne | Ne |
| Primenljiva na offline bazi | Ne | Da | Da | Da | Da |
| SSL podrška | Da | Ne | Da | Ne | Da |
| Ulaz | baza | baza | Backup baze | Arhiva baze | Baza ili backup |
| Izlaz | baza | Arhiva baze | baza | baza | baza |
| Provera konzistentnosti | Da | Ne | Ne | Ne | Ne |
| Rešavanje nekonzistentnosti | Ne | Ne | Ne | Ne | Da |
| Kompaktno memorisanje | Ne | Ne | Ne | Ne | Da |

Tabela 2 – Korišćenje različitih admin komandi

### Dodatne stavke

Neo4j baza podataka sadrži nekoliko podrazumevanih baza podataka a to su baza korisnika *neo4j*, i *system* baza koja sadrži konfiguracije, odnosno, operativno stanje baze, konfiguraciju bezbednosti, kredencijale za prijavu i uloge korisnika. Ove dve baze je potrebno sačuvati sa svim ostalim definisanim bazama.

Dodatni fajlovi koje posebno treba sačuvati su:

* neo4j.conf
* fajlovi koje SSL framework propisuje koji se koriste za enkripciju (privatni ključ, javne sertifikate, ...)
* fajlove dodataka (plugin-ova) koji se koriste uz bazu

Standardna procedura je da se backup-ovi čuvaju odvojeno od proizvodnje, odnosno, na mestu koje nema zajedničke zavisnosti sa mašinom gde se aplikacioni sistem nalazi. Na taj način sprečavamo bilo kakav uticaj sistema na arhivirane fajlove.

## Backup modovi

Kao što smo već pomenuli, backup-ovi mogu biti *potpuni* i *inkrementalni.*

* Potpuni backup (Full backup) - uvek neophodan za prvi backup na određenu lokaciju i može se odraditi i na offline i online bazi.
* Inkrementalni backup (Incremental backup) – posle prvog backup-a, svaki naknadni backup će pokušati da koristi inkrementalni pristup gde se samo razlika transakcionog log-a od prethodnog backup-a pamti i prebacuje na lokaciju čuvanja kopije. Inkrementalni backup je moguće pokrenuti samo nad instancom koja je online

## Backup i restore online baze

Online backup zapravo predstavlja čuvanje konzistentne rezervne kopije Neo4j baze podataka dok ona i dalje radi i dostupan je isključivo u Enterprise verziji. Takođe, backup na Enterprise ediciji proverava konzistentnost podataka. Ovaj proces se izvršava posle generisanja kopije i izuzetno je memorijski i CPU zahtevan.

Komanda nudi mnogo opcija poput definisanja direktorijuma za sam backup (neophodan argument), adresa hosta gde se instance nalazi, da li da se provere konzistentnosti obavljaju, ako ima više baza koje treba ažurirati, specificirati šablon po kome su baze imenovane i sl. Ova komanda je veoma moćna i u većini slučajeva dovoljan alat ali činjenica da je korisna samo u Enterprise ediciji je čini manje impresivnom. Sintaksa je sledeća:

*neo4j-admin backup --backup-dir=<path>*

*[--verbose]*

*[--from=<host:port>]*

*[--database=<database>]*

*[--fallback-to-full=<true/false>]*

*[--pagecache=<size>]*

*[--check-consistency=<true/false>]*

*[--report-dir=<path>]*

*[--check-index-structure=<true/false>]*

*[--check-graph=<true/false>]*

*[--check-indexes=<true/false>]*

*[--check-label-scan-store=<true/false>]*

*[--check-property-owners=<true/false>]*

*[--additional-config=<path>]*

*[--include-metadata=<all/users/roles>]*

Postoji nekoliko povratnih kodova koje ova komanda može da vrati u zavisnosti od toga da li je operacija uspela ili ne ili ako je operacija uspela ali provere konzistentnosti nisu i druge.

Pretpostavimo da želimo da ažuriramo online bazu podataka i da se kopija nalazi na */mnt/backups/neo4j.* Komanda bi glasila:

*bin/neo4j-admin backup --backup-dir=/mnt/backups/neo4j --database=neo4j*

Recimo da treba da kreiramo rezervne kopije za sve baze sa određene adrese čije ime počinje sa **n**. Komanda koja bi uradila ovo je:

*neo4j-admin backup --from=192.168.1.34 --backup-dir=/mnt/backups/neo4j --database=n\**

Restore backup-a se radi adekvatno nazvanom komandom *neo4j-admin restore.* Sintaksa je nešto jednostavnija nego za backup i izgleda ovako.

*neo4j-admin restore --from=<path>[,<path>...]*

*[--verbose]*

*[--database=<database>]*

*[--force]*

*[--move]*

*[--to-data-directory=<path>]*

*[--to-data-tx-directory=<path>]*

Moguće je specificirati lokacije backup-ova koje je potrebno obnoviti, ime baze koja će biti obnovljena, da li već postojeću bazu sa navedenim imenom treba zameniti onom iz kopije...

Da bi smo obnovili bazu koju smo gore sačuvali, možemo iskoristi komandu:

*bin/neo4j-admin restore --from=/mnt/backups/neo4j --database=neo4j –force*

## Backup i restore offline baze

Ukoliko se koristi *Community* verzija, *neo4j-admin backup/restore* komande nisu dostupne i većina posla će morati biti odrađena manuelno. Zbog ovoga, čitav proces kreiranja dump-a i učitavanja istog zahteva nekoliko koraka. Proces kreiranja dump fajla se sastoji iz korišćenja *neo4j-admin dump* komande koja kreira *<database>.dump* arhivu. Komanda izgleda ovako:

*neo4j-admin dump --database=<database>*

*--to=<destination-path>*

*[--verbose]*

Moguće je definisati ime baze i lokaciju na koju želimo da smestimo dump fajl i baza čiju kopiju kreiramo mora biti offline. Alternativno, ukoliko želimo da ručno zapamtimo stanje baze u datom trenutku, to možemo uraditi sledećom kratkom skriptom koja nije deo Neo4j alata.

service neo4j stop &&

now=$(date +"%m\_%d\_%Y") &&

cd /var/lib/neo4j/data/databases/ &&

tar -cvzf /var/backups/neo4j/$now.gb.tar.gz graph.db &&

service neo4j start

Objašnjenje za ovu skriptu:

service neo4j stop – zaustavlja neo4j servis

now=$(date +"%m\_%d\_%Y") – pamti trenutni datum u promenljivu

cd /var/lib/neo4j/data/databases/ - prelazi na lokaciju graph.db fajla

tar -cvzf /var/backups/neo4j/$now.gb.tar.gz graph.db – pravi kompresovanu kopiju graph.db fajla

service neo4j start – pokreće neo4j servis

Ovu skriptu je jednostavno izmeniti po želji i pruža uvid u to kako zapravo *dump* komanda radi.   
 Suprotna komanda dump komandi je *neo4j-admin load* koja može da obnovi bazu podataka koristeći generisani .dump fajl. Sintaksa je sledeća:

*neo4j-admin load --from=<archive-path>*

*--database=<database>*

*[--force]*

*[--info]*

Moguće je definisati ime obnovljene baze i put do .dump fajla.

Kao i ranije, da bi dočarali kako *load* komanda radi, napisaćemo kratku skriptu koja manje-više radi sličan posao kao i load komanda.

*service neo4j stop &&* - zaustavlja neo4j servis

*cd /var/lib/neo4j/data/databases/ && -* prelazi na lokaciju graph.db

*rm -r graph.db &&* - uklanja postojeću graph.db

*tar xf /var/backups/neo4j/05\_16\_2021.gb.tar.gz -C /var/lib/neo4j/data/databases/ && -* Ekstraktuje backup na mesto gde je obrisan graph.db fajl bio („05\_16\_2021” je potrebno izmeniti na ime fajla)

*service neo4j start –* pokreće neo4j servis

Iako komande iz Enterprise verzije Neo4j-a nisu dostupne u Community verziji, i dalje je moguće napraviti strategiju za čuvanje rezervnih kopija koja će biti dovoljno dobra za konkretnu implementaciju.

# Zaključak

Pre kupovine Enterprise verzije, koja je poprilično skupa, potrebno je detaljno analizirati i isplanirati strategiju čuvanja rezervnih kopija kako bi videli koja verzija Neo4j baze je potrebna i dovoljna. Enterprise verzija nudi proveru konzistencije koja proverava strukture indeksa kao i same čvorove i relacije i nema alternative ovog operaciji u Community verziji. Takođe, backup i restore operacije su dosta moćne i nude dodatne funkcionalnosti od kojih se samo neke mogu simulirati u Community verziji. Iako korišćenje NoSQL baza obično prati i smanjeno vreme planiranja jer se eliminiše potreba za kreiranjem šeme baze podataka, neophodno je odvojiti određeno vreme za planiranje čuvanja rezervnih kopija, učestalost čuvanja kopija kao i način da se sistem bezbedno vrati u neko pređašnje stanje uz pomoć već sačuvane kopije.

# Literatura

1. Robinson, I, Webber, J, Eifrem E, 2015, *Graph Databases: New opportunities for connected data*, Second edition, O’Reilly Media, Inc, Sebastopol CA 95472 USA
2. Baton, J, Bruggen, RV, 2017, *Learning Neo4j 3.x: Effective data modeling, performance tuning and data visualization techniques in Neo4j,* Second edition, Packt Publishing, Birmingham.
3. Partner, J, Vukotic, A, Watt, N, 2014, *Neo4j in Action*, Manning Publications Co., 3 Lewis Street Greenwich, CT, United States
4. McCreary, D, Kelly, A, 2013, *Making Sense of NoSQL: A guide for managers and the rest of us,* Manning Publications Company, 20 Baldwin Road, Shelter Island, NY 11964
5. Neo4j: Graph academy:

<https://neo4j.com/graphacademy/>

1. Neo4j: Understanding data on disk: <https://neo4j.com/developer/kb/understanding-data-on-disk/>
2. NoSQL: The future of data economy: <https://www.livemint.com/Opinion/ub2sqRtvuGDgDPDDTP7gtL/NoSQL-The-future-of-data-economy.html>
3. Neo4j: Backup and restore <https://neo4j.com/docs/operations-manual/current/backup-restore/>
4. Neo4j: Backup planning [https://neo4j.com/docs/operations-manual/current/backup-restore/planning/#backup-planning](https://neo4j.com/docs/operations-manual/current/backup-restore/planning/%23backup-planning)

1. NoSQL je zapravo abrevijacija za *Not Only SQL*, mnogi NoSQL sistemi zapravo podržavaju i SQL upite iako su podaci nestruktuirani ali uglavnom imaju neke druge mehanizme (jezike) ekvivalentne SQL-u koji su adaptirani za specifičan slučaj korišćenja. [↑](#footnote-ref-1)
2. ACID svojstva predstavljaju skraćenicu za atomičnost (A – atomicity), konzistentnost (C – consistency), izolaciju (I – isolation) i izdržljivost (D – durability). Ovo su svojstva koje svaka SQL baza garantuje. [↑](#footnote-ref-2)
3. CAP teorema kaže da svaka distribuirana baza podataka ne može da obezbedi više od dva navedena svojstva - konzistencije, dostupnosti i tolerancijе razdvojenosti. [↑](#footnote-ref-3)
4. Potpunu kopiju je neophodno odraditi samo pri prvom backup-u, a kasnije je moguće raditi inkrementalne backup-ove gde se kopira samo razlika transakcionog log-a. Ovi inkrementalni backup-ovi se mogu redom primeniti na jednu bazu kako bi se odradio restore [↑](#footnote-ref-4)