Neo4j graf baza podatka

Milena Kovačević

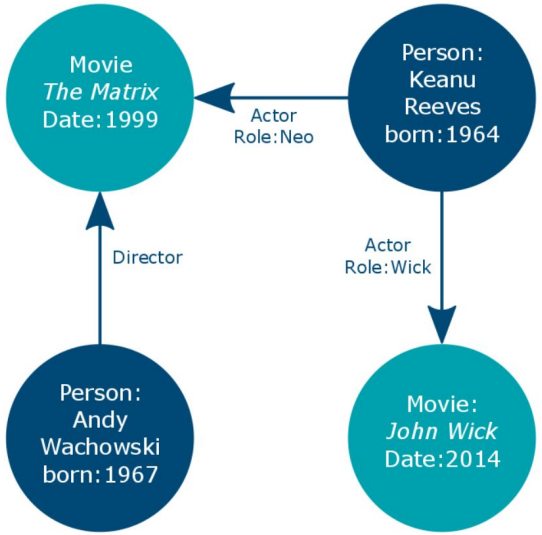
NoSQL baze podataka

- dokument
- ključ-vrednost
- graf
- široke-kolone

Graf baza podataka

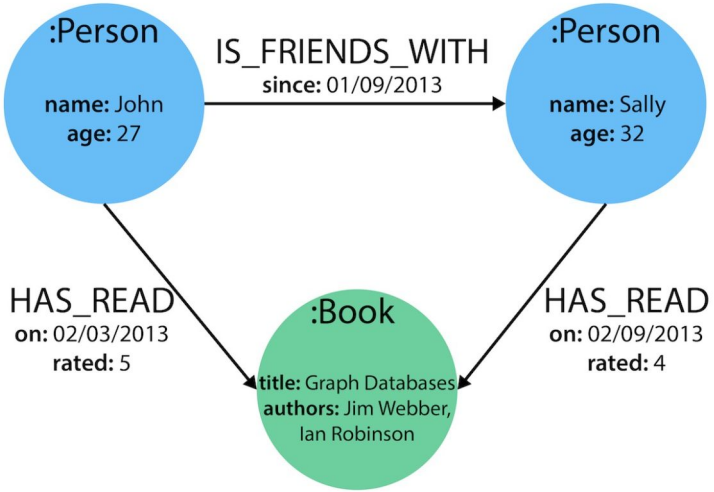
- podaci se čuvaju pomoću graf strukture kao čvorovi i veze između čvorova umesto u vidu tabela ili dokumenata

- posebno je naklonjena sistemima kod kojih veza između podataka ima veliku važnost i sistema koji imaju mrežnu arhitekturu



Elementi modela graf baze podataka

- relacija - predstavlja usmerene ili imenovane veze između dva ili više čvorova
- čvor - reprezentacija entitieta u grafu
- labela - koristi se za grupisanje čvorova
- svojstvo(obeležje) - atributi čvorova i relacija



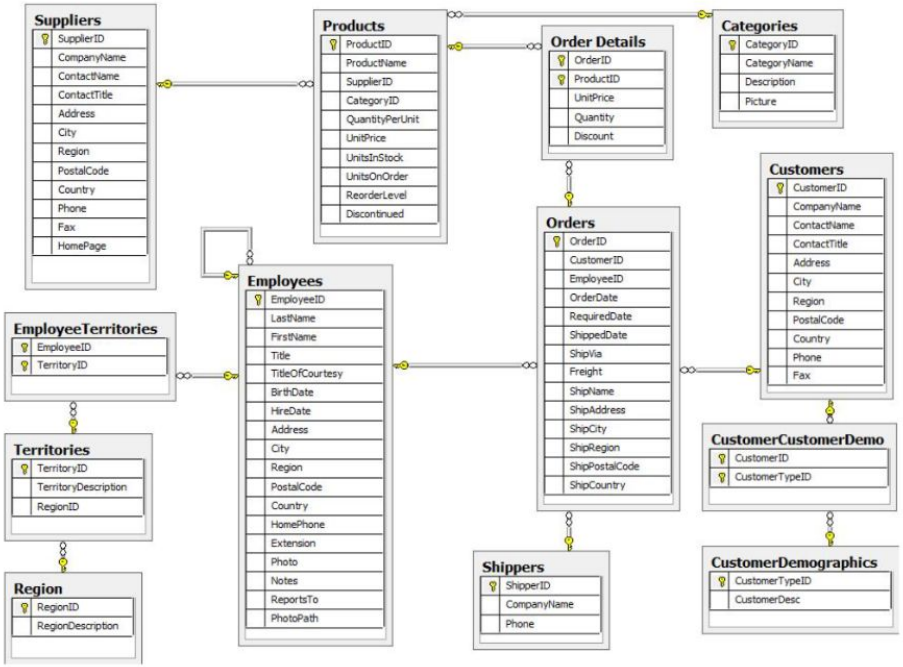
Neo4j pruža

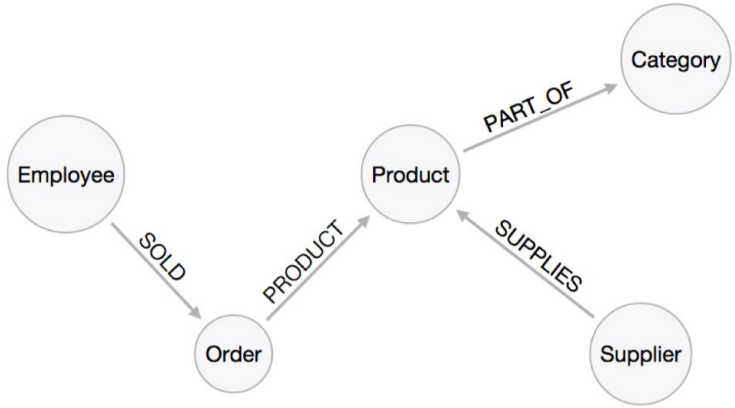
- Fleksibilna šema
- Skaliranje
- Replikacija
- ACID
- Ugrađena web aplikacija
- Jednostavno modelovanje
- Indeksiranje
- Drajveri

Prednosti Neo4j baze u odnosu na relacionu i druge NoSQL baze podataka

- Performanse
- Skalabilnost
- Agilnost
- Nema spajanja
- Cypher upitni jezik
- Visoka dostupnost

Složenost modela sistema koji imaju graf arhitekturu može biti značajno jednostavnija ukoliko je prikazana preko graf baze podataka.





Cypher

Neo4j koristi deklarativan jezik koji služi za upite, *Cypher*, koji je sličan SQL-u, ali je optimizovan sa rad sa grafovima.

Kreiranje čvorova

CREATE (you:Person {name:"Jovica"}) RETURN you

Kreiranje relacija

MATCH(you:Person{name:"Jovica"})CREATE (you)-[like:LIKE]->(neo:Database {name:"Neo4j" }) RETURN you,like,neo

Brisanje čvora

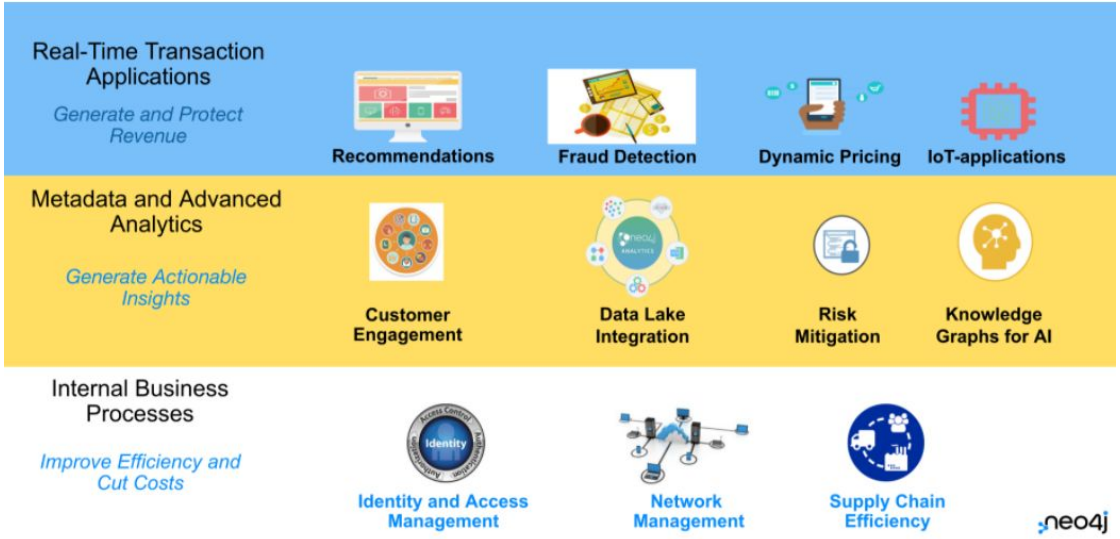
MATCH (n:Useless) DELETE n

Brisanje relacija

MATCH (n { name: 'Andrew' }) - [r:FRIEND] -> () DELETE r

Primena Neo4j baze podataka

- Otkrivanje i prevencija prevare
- Praćenje mrežne infrastrukture
- Društvene mreže
- Sistemi za preporuke u realnom vremenu
- Prava pristupa i kontrola identiteta



Primer upotrebe Neo4j baze podataka za otkrivanje i sprečavanje prevare

Pisanje potrebnih klasa

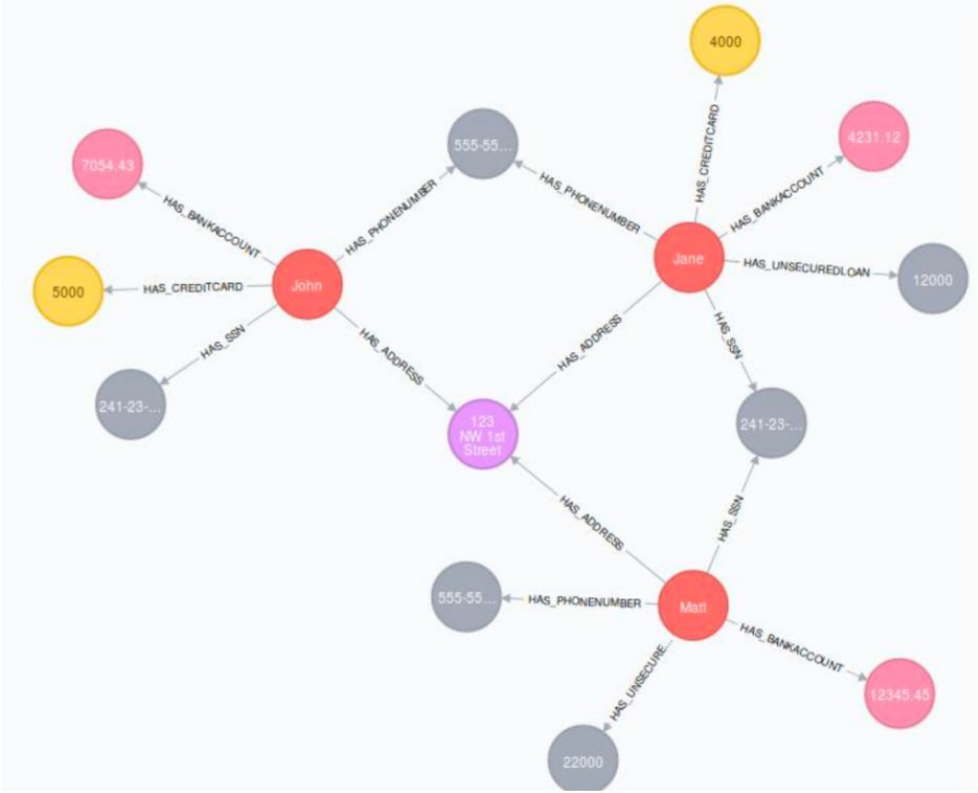
```
// Create account holders
CREATE (accountHolder1:AccountHolder {
 FirstName: "John",
 LastName: "Doe",
 UniqueId: "JohnDoe" })

CREATE (accountHolder2:AccountHolder {
 FirstName: "Jane",
 LastName: "Appleseed",
 UniqueId: "JaneAppleseed" })

CREATE (accountHolder3:AccountHolder {
 FirstName: "Matt",
 LastName: "Smith",
 UniqueId: "MattSmith" })

// Create Address
CREATE (address1:Address {
 Street: "123 NW 1st Street",
 City: "San Francisco",
 State: "California",
 ZipCode: "94101" })
```

```
// Connect 3 account holders to 1 address
CREATE (accountHolder1)-[:HAS_ADDRESS]->(address1),
 (accountHolder2)-[:HAS_ADDRESS]->(address1),
 (accountHolder3)-[:HAS_ADDRESS]->(address1)

// Create Phone Number
CREATE (phoneNumber1:PhoneNumber { PhoneNumber: "555-555-5555" })

// Connect 2 account holders to 1 phone number
CREATE (accountHolder1)-[:HAS_PHONENUMBER]->(phoneNumber1),
 (accountHolder2)-[:HAS_PHONENUMBER]->(phoneNumber1)

// Create SSN
CREATE (ssn1:SSN { SSN: "241-23-1234" })

// Connect 2 account holders to 1 SSN
CREATE (accountHolder2)-[:HAS_SSN]->(ssn1),
 (accountHolder3)-[:HAS_SSN]->(ssn1)

// Create SSN and connect 1 account holder
CREATE (ssn2:SSN { SSN: "241-23-4567" })<-[:HAS_SSN]-(accountHolder1)
```

```
// Create Credit Card and connect 1 account holder
CREATE (creditCard1:CreditCard {
 AccountNumber: "1234567890123456",
 Limit: 5000, Balance: 1442.23,
 ExpirationDate: "01-20",
 SecurityCode: "123" })<-[:HAS_CREDITCARD]-(accountHolder1)

// Create Bank Account and connect 1 account holder
CREATE (bankAccount1:BankAccount {
 AccountNumber: "2345678901234567",
 Balance: 7054.43 })<-[:HAS_BANKACCOUNT]-(accountHolder1)

// Create Credit Card and connect 1 account holder
CREATE (creditCard2:CreditCard {
 AccountNumber: "1234567890123456",
 Limit: 4000, Balance: 2345.56,
 ExpirationDate: "02-20",
 SecurityCode: "456" })<-[:HAS_CREDITCARD]-(accountHolder2)

// Create Bank Account and connect 1 account holder
CREATE (bankAccount2:BankAccount {
 AccountNumber: "3456789012345678",
 Balance: 4231.12 })<-[:HAS_BANKACCOUNT]-(accountHolder2)
```

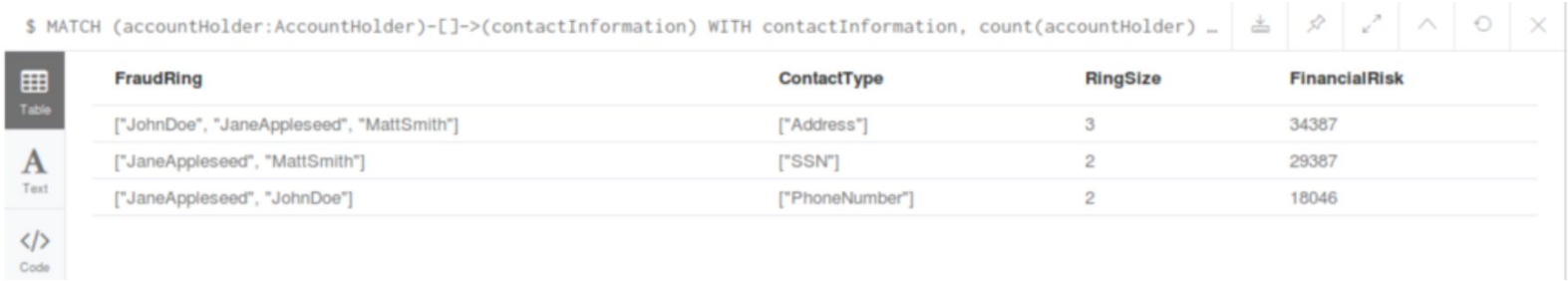
```
// Create Unsecured Loan and connect 1 account holder
CREATE (unsecuredLoan2:UnsecuredLoan {
 AccountNumber: "4567890123456789-0",
 Balance: 9045.53,
 APR: .0541,
 LoanAmount: 12000.00 })<-[:HAS_UNSECUREDLOAN]-(accountHolder2)

// Create Bank Account and connect 1 account holder
CREATE (bankAccount3:BankAccount {
 AccountNumber: "4567890123456789",
 Balance: 12345.45 })<-[:HAS_BANKACCOUNT]-(accountHolder3)

// Create Unsecured Loan and connect 1 account holder
CREATE (unsecuredLoan3:UnsecuredLoan {
 AccountNumber: "5678901234567890-0",
 Balance: 16341.95, APR: .0341,
 LoanAmount: 22000.00 })<-[:HAS_UNSECUREDLOAN]-(accountHolder3)

// Create Phone Number and connect 1 account holder
CREATE (phoneNumber2:PhoneNumber {
 PhoneNumber: "555-555-1234" })<-[:HAS_PHONENUMBER]-(accountHolder3)

RETURN *
```

Dobijeni rezultat nakon zvršavanja



Naredni korak predstavlja pronalazak vlasnika računa kojima se podaci poklapaju.

Pored pronalaženja vlasnika računa koji su potencijalni prevaranti, treba otkriti i koliki maksimalan gubitak svaki od njih donosi ukoliko načini prevaru. Kao rezultat dobijamo povezane korisnike banke koji imaju račune u toj banci, kao i gubitak koji bi njihova prevara donela.



Performanse

Neo4j baza podataka konstantno teži poboljšanju performansi.

Neo4j distribuirana klaster arhitektura visokih performansi prilagođava se podacima i poslu, minimizirajući cenu i hardver, a maksimizira performanse u povezanim skupovima podataka.

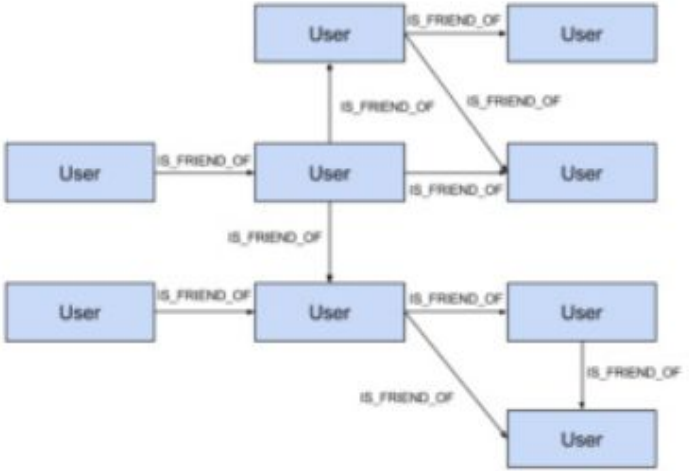
Pruža pouzdano brze transakcije sa visoko paralelizovanim protokom čak i kada podaci rastu, jer graf obezbeđuje susednost bez indexa, što skraćuje vreme čitanja i postaje još bolje kako složenost podataka raste.

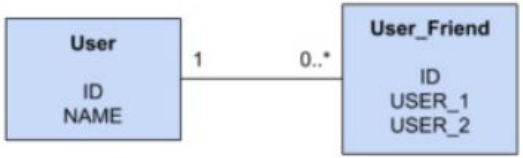
2	0.028
3	0.213
4	10.273
5	92.613

2	0.04
3	0.06
4	0.07
5	0.07

Poređenje modela podataka u MySQL bazi i Neo4j bazi

Prethodni rezultati su očekivani i očigledni, pošto prilikom produbljavanja odnosa između čvorova, za razliku od modela u Neo4j bazi, model u MySQL bazi podataka se značajno komplikuje.





Neo4j u svetu velikih podataka

- multifunkcionalna upoterba Neo4j baze podataka
- odnos između čvorova pruža informaciju o odnosu između podataka
- unapređuje rad globalnih brendova
- kombinacija sa drugim skladištima podataka radi lakše obrade i boljeg prikaza rezultata

Neo4j se pokazala kao sjajan izbor za pretrage i manipulaciju podacima koji su međusobno usko povezani. Prilikom povećanja podataka ili prilikom produbljavanja odnosa između podataka, brzina rada ove baze podataka se ne menja značajno.

2	0.01
3	0.168
4	1.359
5	2.132

Zaključak

- bolji prikaz veza u sistemu i sistema
- multifunkcionalna i primenjiva za rešavanje različitih problema
- cypher upitni jezik optimizovan za graf bazu
- najbolja prilikom rada sa sistemima koji imaju mrežnu arhitekturu
- kombinacija sa drugim bazama prilikom obrade velikog skupa podataka
- alatke i biblioteke za lakši i brži razvoj
- alati za programere radi lakšeg razvoja i grafičkog prikaza

Hvala na pažnji!