NoSQL baze podataka

Predavanje 5: Modeli distribucije, Dokument-orijentisae baze podataka

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Razlozi za korišćenje NoSQL baza podataka

- Paralelno procesiranje velike količine podataka u distribuiranim sistemima.
- Pretraga podataka u sistemima s velikom količinom podataka.
- Skladištenje velike količine podataka (struktuiranih, delimično struktuiranih, nestruktuiranih).
- Mogućnost horizontalnog skaliranja.
- Relativno jednostavna distribucija podataka.

Modeli distribucije

- Kao što je renije već bilo pomenuto, jedana od glavnih karakteristika NoSQL baza podataka je njihova mogućnost da se izvršavaju na klasterima servera umesto na samo jednom računara;
- Za razliku od relacionih baza podataka koje su uglavnom projektovane da se izvršavaju centralizovano na jednom serveru.
- Na taj način se otvara mogućnost skaliranja sistema kada dodje do povećanja količine podataka i zahteva za dodatnim resursima.
- U zavisnosti od načina korišćenja koje planiramo u našem sistemu i potreba koje iz toga proističu odrediće se koji model distribucije najviše odgovara u konkretnom slučaju.

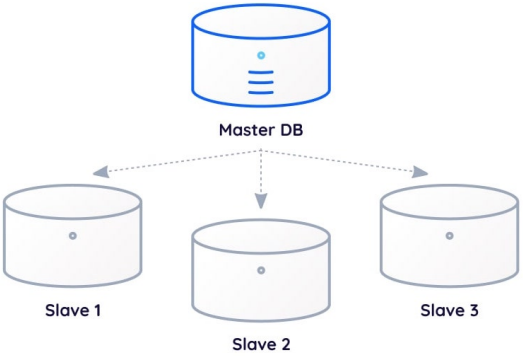
- Postoje dve osnovne tehnike distribucije,
  - sharding – podaci se rasporedjuju na različite čvorove, tako da ukpne podatke baze podataka čini unija podataka rasporedjenih po različitim čvorovima.
  - replication – po čvorovima rasporedjuju kopije kompletne baze podataka, tako da se isti podaci nalaze na više različitih računarskih čvorova.
- iz kojih se može izvesti nekoliko različitih modela distribucije podaaka.
  - single server,
  - master-slave replikacija
  - sharding
  - peer-to-peer replikacija

Single server

- Najjednostavniji pristup je uopšte ne vršiti distribuciju podataka, odnosno
  - da se sistem za upravljanje bazom podataka izvršava na jednom serveru i
  - da se na njega smesti kompletna baza podataka.
- Ponekad je opravdano NoSQL bazu koristiti na jednom serveru ako takav model NoSQL skladišta odgovara aplikaciji.
- Ako ne postoje posebni razlozi za distribuciju, kad god je moguće treba izabrati arhitekturu s jednim serverom.

Master-slave replikacija

- Proces odgovoran za replikaciju vrši sinhronizaciju podataka na *slave* čvorovima u odnosu na stanje podataka *master* čvora
- Na taj način se obezbedjuje da stanje baze podataka *slave* čvorova u svakom trenutku odgovara stanju baze podataka *master* čvora.
- Ovakav model distribucije pogodan je za primenu u sistemima kod kojih je dominanto pretraživanje podataka u odnosu na ažuriranje.



(https://www.adservio.fr/post/how-to-replicate-mysql-databases)

Prednosti

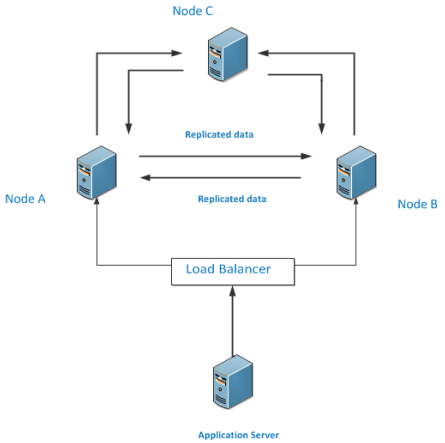
- Prednosti koje donosi ovakav model distribucije:
  - Sistem se može horizontalno skalirati, kako bi mogao odgovoriti na veći broj zahteva za pretraživanjem podataka;
  - Dodavanjem slave čvorova i podešavanjem da se pretraživanje podataka usmeri ka slave čvorovima.
- Osim toga, ovakavim modelom distribucije povećava se dostupnost podataka koji se pretražuju.
- U slučaju otkaza *master* čvora, *slave* čvorovi i dalje mogu prihvatiti zahteve za pretraživanjem podataka čime se:
  - Povećava raspoloživost sistema, posebno sistema kod kojih je dominatno pretraživanje.
  - S druge strane, postojanje kopije baze podataka na slave čvorovima omogućuje da neki od slave čvorova može veoma brzo preuzeti ulogu master čvora i obezbediti nesmetan nastavak rada sistema dok se ne izvrši oporavak master čvora.

Mane

- Master-Slave model distribucije pokazuje i odredjene nedostatke:
  - Posebno u primeni kod sistema u kojima postoji veliki broj zahteva za ažuriranjem podataka.
  - Sva ažuriranja vrše preko master čvora on može postati usko grlo u sistemu u slučaju intenzivnog ažuriranja podataka.
  - Iako se master čvor može rasteretiti preusmeravanjem upita ka slave čvorovima, tako da obradjuje samo zahteve za ažuriranjem
- Master-slave model distribucije ne predlaže za sisteme u kojima se vrši veliki broj ažuriranja podataka u jedinici vremena.

Peer-to-peer replikacija

- Master-Slave replikacija pomaže kod skaliranja operacija čitanja baze podataka;
- Medjutim ne pomaže u skaliranju operacija upisa, odnosno ažuriranja baze podataka.
- Njome se postiže otpornost na ispad *slave* servera ali ne i *master* servera baze podataka.
- Peer-to-peer replikacijom se izbegava mogućnost uskog grla (single point of failure) jer ne postoji Master server.



(https://www.sqlshack.com/peer-to-peer-replication/)

Mane

- Problem kod Peer-to-peer replikacije je obezbedjivanje konzistentnosti podataka.
- Ažuriranje podataka u bazi moguće na više različita mesta – rizik da dva korisnika ažuriraju iste podatke u isto vreme i da nastane konflikt upisa.
- Nekonzistentnost kod čitanja baze podataka je takodje problem, medjutim ona je prolazna, dok je nekonzistentnost upisa trajna.
- Medjutim, postoje načini da se problem nekonzistencije upisa eliminišu ili bar smanje na najmanju moguću meru.
- Potrebno je kod ažuriranja baze podataka obezbediti da prilikom ažuriranja podataka replikacije rade koordinirano kako bi se izbegao konflikt upisa.

Prednosti

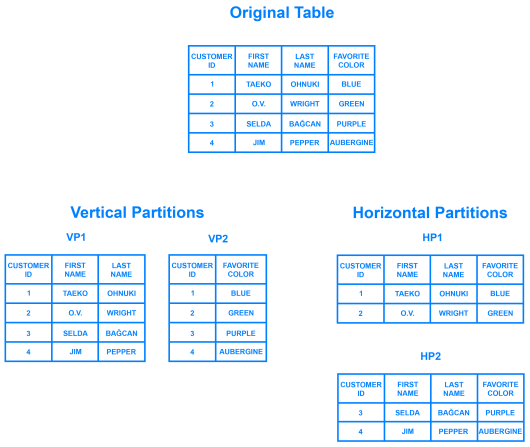
- Postoje situacije kada je moguće izboriti se s nekonzistentnošću upisa na taj način što će se vršiti spajanje nekonzistentnih upisa
- Performansama koje donosi Peer-to-peer replikacija s mogućnošću ažuriranja baze na bilo kojoj replikaciji
- Relativno laka skalabilnost
- Moguće je dodati u automatsku konflikt upotrebom CRDT-jeva (Riak, Redis, AntidoteDB)

Uvod

- Ponekad se velika konkurencija nad bazom podataka dogadja zbog toga što različiti korisnici baze podataka pristupaju potpuno različitim skupovima podataka.
- U takvim situacijama može se vršiti podrška horizontalnom skaliranju na način da se različiti skupovi podataka smeste na različite servere u sistemu.
- Time jednu bazu podataka koja teba da skladišti veliku količinu podataka podelino na n malih baza podataka gde svaka čuva samo opseg ukupnog seta podataka
- Ovakva tehnika distribucije podataka se naziva Sharding.

Mogućnosti shardovanja

- Shardovanje možemo izvesti na dva načina:
  1. Vertikalno, podelom tabele na delove koloona – nekada se naziva i particioniranje
  2. Horzontalno, tako što podelimo redove na n manjih mesta



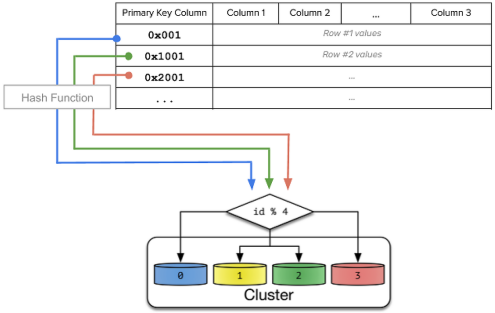
(https://www.digitalocean.com/community/tutorials/understanding-database-sharding)

- U idealnoj situaciji imamo različite korisnike koji svoje potrebe za podacima iz baze podataka zadovoljavaju pristupajući različitim serverima.
- Jedan korisnik pristupa samo jednom serveru, tako da se na taj način postiže poboljšanje performansi i vreme odziva servera.
- Pristup bazi podataka se može izbalansirati, tako da se ravnomerno rasporedi po serverima.
- Na primer, ako postoji deset servera u idealnoj situaciji svaki bi obradjivao 10% ukupnih zahteva pristupa bazi podataka – u idelanom slučaju

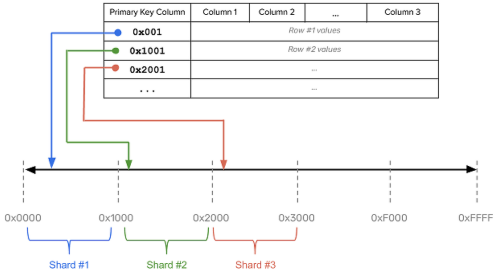
- Medjutim, idealna situacija je veoma retka, tako da je potrebno pokušati obezbediti da se podaci kojima se zajedno pristupa budu rasporedjeni na isti server i to na server koji obezbedjuje najbolji pristup podacima.
- Prvi deo ovog pitanja je kako grupisati podatke tako da jedan korisnik uglavnom dobija svoje podatke sa jednog servera.
- U ovoj situaciji agregiranje podataka postaje veoma korisno.
- Ceo smisao agregacija je da ih dizajniramo tako da kombinuju podatke kojima se cesto pristupa zajedno – tako da se agregacije izdvajaju kao očigledane jedinice distribucije.
- Sto se tice rasporedjivanja po čvorovima, postoji vise faktora koji mogu da pomognu u poboljsavanju preformansi.
- Ako znamo odakle se, u fizickom smislu, najcesce pristupa odredjenim agregacijama podataka, njih mozemo postaviti blizu lokacije iz koje se pristupa.

- U proslosti, sharding se uglavnom radio kao deo logike aplikacije.
- Ovo komplikuje model programiranja zato sto kod aplikacije mora obezbediti da su zahtevi rasporedjeni na vise shard-ova.
- Mnoge NoSQL baze podataka nude auto- sharding, gde baza podataka preuzima odgovornost za alociranje podataka na shardove i osigurava da podaci odlaze na odgovarajuci shard.
- Ovo u mnogome olakšava korišćenje sharding- a u aplikaciji.
- Pored toga, rebalansiranje shardova znaci menjanje koda aplikacije i pomeranje podataka.

- Sharding možemo izvesti na nekoliko načina:
  - Algoritamski sharding – koristi nekakvu funkciju da odredi na koji shard podatak/upit treba da se prosledi
  - Dinamički sharding – koristi opsege ključeva da odredi na koji shard podatak/upit treba da se prosledi
- Pored toga , postoji nekoliko načina ko može da odredi prethodne proračune:
  - Sama NoSQL baza, na osnovu ugradjene logike
  - Spoljni servis (locator service), koji čuva radi raspored zahteva



Algoritamski sharding



Dinamički sharding

(https://www.yugabyte.com/tech/database-sharding/)

Uvod

- Dokument je osnovni koncept.
- Dokumant orijentisane baze podataka skladište i pretražuju dokumante (JSON, BSON, XML, ...).
- Dokumenti su:
  - Samoopisujući
  - Hijerarhijska struktura podataka (tipa stabla, parova ključ-vrednost, kolekcija i skalarnih vrednosti).
- Dokumenti su struktuirani najčešće pomoću popularnog JSON formata.
- Takav oblik podataka je vrlo intuitivan i prirodan način modelovanja podataka.

- Može se preslikati u objektno orijentisano programiranje, gde bi svaki dokument bio jedan objekat.
- Svaki dokument se može sastojati od jednog ili više atributa.
- Vrednosti atributa u dokumantu mogu biti:
  - Tekst – String
  - Broj
  - Logička vrednost – Boolean
  - Niz
- Svi podaci koji su vezani za dokument se u bazu podataka smeštaju kao jedna celina – agregat
- Na taj način se smanjuje vreme pristupa podacima i gotovo potpuno izbacuje potreba za kompleksnim spajanjem tabela.

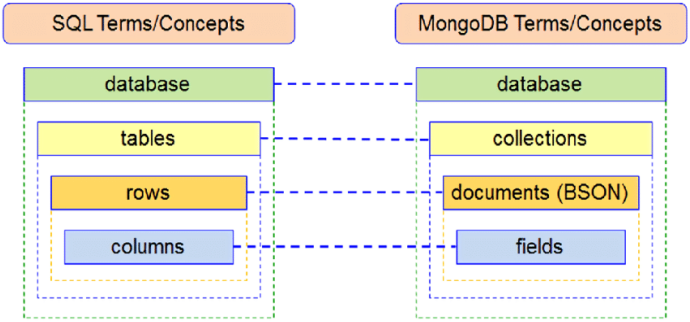
- Što se tiče šeme može se reći da je u ovakvim bazama podataka gotovo i nema.
- Svaki dokumant može sadržati različite atribute, čime se postiže fleksibilnost.
- Olakšava se modelovanje delimično struktuiranih podataka ili polimorfnih tipova podataka.
- Na taj način se, takodje olakšava editovanje aplikacija jer je vrlo lako dadati neki atribut dokumanta.
- Skladištenje podataka na ovakav način ima nekoliko prednosti:
  - Dokumenti su nezavisne jedinice, a omogućavaju bolje performanse i distribuciju podataka na više servera, čuvajući ih i lokalno.
  - Aplikativna logika je jednostavnija za programiranje – Ne moraju se vršiti promene objekata i podataka iz baze već se objekat pretvara direktno u dokumant.

- Nestruktuirani podaci se vrlo lako skladište u bazu. Obzirom da dokument može sadržati sve što aplikativna logika dopušta, nema strogo definisane šeme.
- Pretraživanje je takodje relativno jednostavno. Podaci se najčešće mogu pretraživati preko bilo kog atributa uutar dokumenta.
- Ovakve baze podataka su vrlo pogodne za sisteme za upravljanje dokumentima, gde se kompletni dokumenti trebaju skladištiti u bazu podataka
- Ali ih je moguće primeniti za širok spektar aplikacija zbog fleksibilnosti modela podataka i mogućnosti pretraživanja dokumanata po bilo kom atributu.
- Najčešće pružaju neku vrstu okvira za agregaciju podataka i to za analize u realnom vremenu

- Dokument baza podataka skladišti dokumente u kojima su podaci predstavljeni preko parova ključ-vrednost, gde se:
- Dokumanti indeksiraju korišćenjem B-Stabla,
- Postavljanje upita se vrši korišćenjem JavaScript query engine-a

Terminologija

- Dokumenti mogu imatei razlike u atributima dokumenata, ali pripadaju istoj kolekciji
- Ovo se razlikuje u odnosu na relacione baze podataka gde kolone:
  - Sadrže iste tipove podataka
  - Ili null vrednosti



(https://www.scirp.org/journal/paperinformation.aspx?paperid=99539)

- Dokumanti nisu definisani šemom i imaju fleksibilnu strukturu
- To je pogodno kod preslikavanja dokumanta u objekat – svaki dokumant može porediti polja dokumanta iako su dokumanti različite strukture
- Podaci su predstavljeni preko preslikavanja
- Veze se mogu predstaviti kao: reference ili ugnježdeni (embeded) dokumenti – object u JSON-u

Modelovanje

- Već smo videli da se podaci kod NoSQL baza podataka modeluju nešto drugačije nego kod relacionih baza podataka.
- Kod dokument baza podataka kolekcija ne zahteva tačnu strukturu dokumenta koji se u nju smešta.
- Naravno, dokumenti koji se smaštaju u istu kolekciju imaju sličnu strukturu, jer predstavljaju iste entitete.
- Najveći izazov kod modelovanja podataka je balansiranje izmedju potreba aplikacije, performansi i karakteristika sistema za upravljanje bazom podataka koji se koristi, kao i načina pretraživanja podataka.

- Najvažnija odluka pri definisaju strukture dokumanta je odrediti na koji će način dokumanti biti povezani jedan s drugim.
- Postoje dva načina povezivanja, to su:
  1. Reference
  2. Ugradjeni (embeded) dokumenti

Dodatni materijali

- Making Sense of NoSQL A guide for managers and the rest of us
- Database Internals
- NoSQL Distilled A Brief Guide to the Emerging World of Polyglot Persistence
- Seven Databases in Seven Weeks

Pitanja

Pitanja :) ?