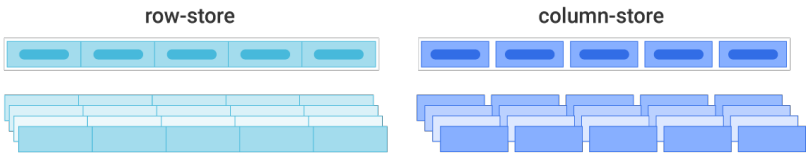
Cassandra - Wide-Column DB

NoSQL baze podataka

Univerzitet u Novom Sadu
Fakultet tehničkih nauka

Row VS Column DBs

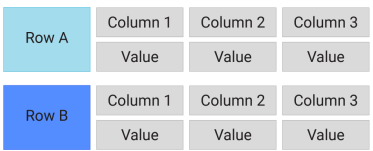
- Row-oriented baza podataka - torka se čuva kao blok u memoriji
- Column-oriented baza podataka - kolona se čuva kao blok u memoriji



Wide-Column DBs

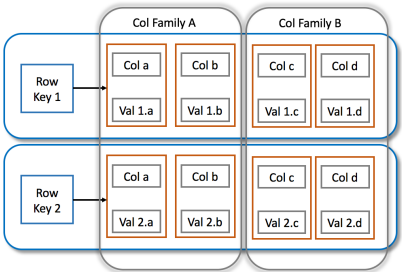
Nisu Column-oriented baze podataka!

- Spadaju u **Row-oriented** baze podataka
- Zašto se onda zovu Wide-**Column**?
  - Mogu imati veliki broj kolona
  - Svaki red može imati različite kolone
  - Mogu imati veliku širinu...



Wide-Column DBs

- Po strukturi: *dvodimenzionalne key-value baze podataka*
- Pristup vrednostima za red i kolonu: *rowKey.colKey*
- Više kolona može biti grupisano u porodice kolona



Cassandra

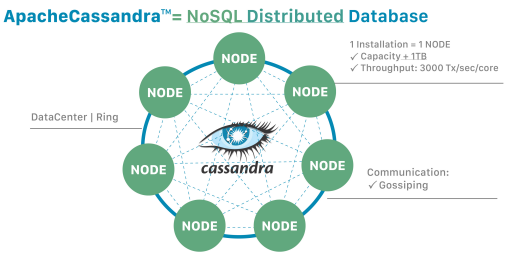
- *Zvanična dokumentacija*
- *Go driver*

Osnovne informacije

- Wide-Column baza podataka
- Moć skladištenja podataka se meri u **PB**
  - 1 PB = 1 024 TB 1 000 000 GB
- Skalabilna
- Distribuirana baza podataka *(partitioning)*
- Visoka dostupnost *(availability)*
- CAP teorema **AP** baza podataka *(u osnovnoj konfiguraciji)*
- Izuztna otpornost na otkaze
- Minimalne šanse za gubitak podataka u produkciji *(uz dobru konfiguraciju)*
- **Denormalizacija podataka** - dupliranje podataka, **ne postoji strani ključ**
- **Query-first dizajn baze** - modelovanje na osnovu upita nad podacima

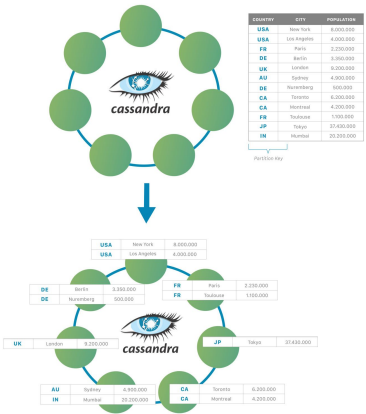
Arhitektura

- **Čvor** - jedan Cassandra server
- **Prsten** - klaster Cassandra servera = *DataCenter*
- **Peer-to-peer** komunikacija između čvorova kroz *Gossiping*



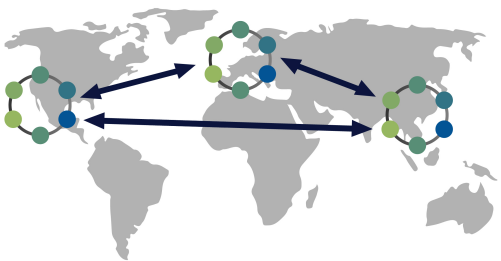
Particionisanje

- Svaki čvor skladišti određeni deo podataka - **particiju**
- **Partition key column** - određuje u kojoj particiji će podatak biti zapisan
- Nije dovoljno čuvati podatke samo u jednom čvoru
  - U slučaju otkaza samo jednog čvora bismo izgubili podatke
  - Loš **fault tolerance**
  - Rešenje?



Replikacija

- **Replikacija podataka**
- U okviru različitih čvorova u prstenu čuvamo duplirane podatke
- **RF** - Replication Factor = diktira broj kopija podataka u okviru baze
- Replike podataka možemo čuvati blizu korisnika koji ih potražuju
- Ali i na udaljenim lokacijama kako bismo osigurali integritet podataka u slučaju otkaza

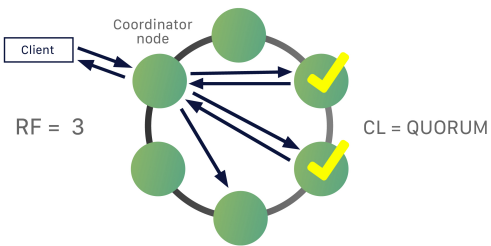


Konzistentnost

- Kako čuvamo podatke na više lokacija, moramo nekako očuvati njihov integritet
- Očuvanje konzistentnosti između više čvorova (i prstenova)
- **Eventual consistency** - baza će u nekom trenutku biti u konzistentnom stanju
- **CL** - Consistency Level = minimalan broj čvorova koji moraju da potvrditi operacije čitanja i pisanja kako bi se one smatrale uspešnim
- Podešavanjem CL vrednosti menja se preferencija performansi, dostupnosti i konzistentnosti podataka

Konzistentnost

- Postupak upisa i čitanja podataka
  - **Koordinator** - čvor koji obrađuje zahtev
  - Ima ulogu da prosledi upit u čvorove koji skladište odgovarajuću particiju
  - Tek kada odgovarajući broj čvorova odgovori, operacija se smatra uspešnom



Tipovi podataka

- Svi osnovni tipovi
  - text (varchar), int, double, date, timestamp, duration
  - uuid, *timeuuid*
- Kolekcije
  - **Mapa** - sortirani set parova ključ-vrednost
    - Vrednost mora biti osnovnog tipa!
    - Moguće je postaviti **TTL** za parove u okviru mape
  - **Set** - sortirana kolekcija jedinstvenih vrednosti; slično mapi
  - **Lista** - sortirana kolekcija vrednosti; slično mapi i setu
- **Tuple** - par vrednosti koje mogu biti različitog tipa
- **UDT** - User Defined Type
  - Mogu se ugnjždavati
  - Mogu sadržati kolekcije
  - **frozen** - immutable; ne mogu se menjati vrednosti pojedinačnih polja

Modelovanje baze podataka - osnovni pojmovi

- **Keyspace** - objekat najvišeg nivoa u okviru baze podataka; sadrži tabele, UDT-ove, podatke o RF-u
- **Tabela** - sadrži redove i kolone sa podacima
- **Red** - torka
- **Partition key column** = PK- određuje u kojoj particiji će podatak biti zapisan
  - Svaka tabela mora definisati PK
  - **Ne mora biti jedinstveni identifikator** torke
- **Clustering key column** = CK - ključ koji u kombinaciji sa PK predstavlja jedinstveni identifikator torke
  - Kreira grupacije *(klastere)* podataka
  - Zgodan za sortiranje i dobavljanje podataka u smislenim celinama

Modelovanje baze podataka - osnovni pojmovi

- **Static column** - kolona koja ima konstantnu vrednost na nivou Particionisanje
  - Može se koristiti za modelovanje **1-N veza**
- Pristup podacima na osnovu vrednosti drugih kolona sem PK i CK se ne praktikuje i uobičajeno nije moguće (bez kreiranja sekundarnog indeksa)!
- Kreiranje sekundarnog indeksa nije dobra praksa kod Cassandre!

Pravila dobrog particionisanja podataka

- Čuvati zajedno ono što se dobavlja zajedno
  - Ako dobavljamo korisnike na osnovu grada -> grad je partition key
  - Ako dobavljamo komentare na osnovu snimka -> snimak je partition key
- Izbegavati prevelike particije
  - Preko 100 000 redova
  - Preko 100 MB
- Izbegavati česte pristupe samo jednoj od N particija *(hot partition)*
- Kreirati ograničene particije
  - **Bucketing** - deljenje particija na manje kroz proširenje partition ključa na više kolona
  - Čuvamo podatke koje prikupljaju IoT senzori i PK nam je id senzora
  - Senzor prikuplja hiljade podataka dnevno -> imaćemo problem velike particije
  - Razdelimo podatke u "kofe" po mesecu kada je očitan podataka za senzor

Modelovanje baze podataka

1. Definisanje domenskog modela
2. Definisanje slučajeva korišćenja aplikacije i upita nad podacima
3. Logičko modelovanje baze
4. Fizičko modelovanje baze
5. Evaluacija i unapređenje modela

Najčešće se jedan upit mapira na tačno jednu tabelu!

cqlsh

- **CQL** = Cassandra Query Language - sintaksno jako sličan SQL jeziku
- **docker exec -it ime_kontejnera cqlsh** - povezivanje na *cqlsh* u okviru pokrenutog Docker kontejnera
- **CREATE KEYSPACE ...;** - kreiranje keyspace-a
- **USE keyspace_name;** - pozicioniranje na keyspace sa imenom *keyspace_name*
- **CREATE TABLE table_name ...;** - kreiranje tabele

cqlsh - CRUD

- **INSERT INTO (column1, column2,...) VALUES (value1, value2,...);** - **c**reate
- **SELECT column_list FROM [keyspace_name.]table_name [WHERE]...;** - **r**ead
- **UPDATE [keyspace_name.]table_name SET value1 [, value2] WHERE ...** - **u**pdate
- **DELETE FROM [keyspace_name.]table_name WHERE ...** - **d**elete

Napomena: potrebno je određeno vreme za inicijalizaciju baze prilikom pokretanja kontejnera! Do tada, kao odgovor dobijamo grešku da ne postoji nijedan server dostupan na željenoj adresi

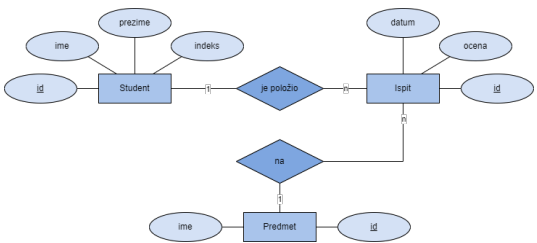
Primer

U okviru primera *RestCassandra* upotrebljeni su:

1. *Cassandra* - baza podataka
2. *CQL skripta* - inicijalizacija baze podataka
3. *Docker* - kontejnerizacija rešenja (i "instalacija" baza)
4. *Go* - implementacija primera

Modelovanje baze podataka - primer

1. Definisanje domenskog modela
2. Definisanje slučajeva korišćenja aplikacije i upita nad podacima
   - Prikaz ocena po studentu sortirane po datumu (od najmanjeg) i po oceni (od najvece)
   - Prikaz ocena po predmetu sortirane po indeksu (od najmanjeg)
3. Logičko modelovanje baze
4. Fizičko modelovanje baze
5. Evaluacija i unapređenje modela



Zadaci

- Proširiti servis (i model podataka) tako da podržava prikaz svih studenata u okviru studijskog programa
- Proširiti servis tako da podržava prikaz ocena za predmet i smer
- Proširiti model tako da *Student* sadrži informaciju o završenim stepenima studija (student može imati proizvoljan broj završenih stepena studija)
- Proširiti servis tako da podržava dodavanje informacije o završenim stepenima studija za studenta
- **Bonus**: Kako bismo proširili sistem da omogući dobavljanje studenta sa najvišim prosekom?
  - Da li je ovakva baza podataka zgodna za statistike?