CouchDB

Ukratko o CouchDB

- NoSQL baza podataka otvorenog koda
- Dokumentima orijentisana (document oriented)
- Shema nije fiksna, već promenljiva
- JSON za skladištenje podataka
- JavaScript za pisanje upita
- RESTful API za komunikaciju sa bazom
- Otpornost na greške
- Master-master replikacija
- Itd.

Osnovni pojmovi

- **CRUD** – Create, Read, Update, Delete
- **cURL** - alat komandne linije za primanje ili slanje podataka koristeći URL sintaksu i razne protokole
- **REST/HTTP** – REpresentational State Transfer/Hyper Text Transfer Protocol – arhitekturalni stil i najpoznatija implementacija HTTP - koristi metode kao što su GET, PUT, POST, DELETE, HEAD itd.
- **API** – aplikacioni programski interfejs, određuje rečnik konvencije pozivanja koje programmer treba da primeni kako bi koristio servise

Dokumenti

- Osnovne jedinice za čuvanje podataka
- Prirodniji za korišćenje od tabela i samoodrživi
- Obično su semantički slični, ali ne i isti
- Čoveku nije teško da se prilagodi varijacijama
- Model podataka ne mora da se zna unapred, već se može definisati u hodu
- Nema apstraktnih referenci ili NULL vrednosti
- Primeri – vizit karte, računi, katalozi...

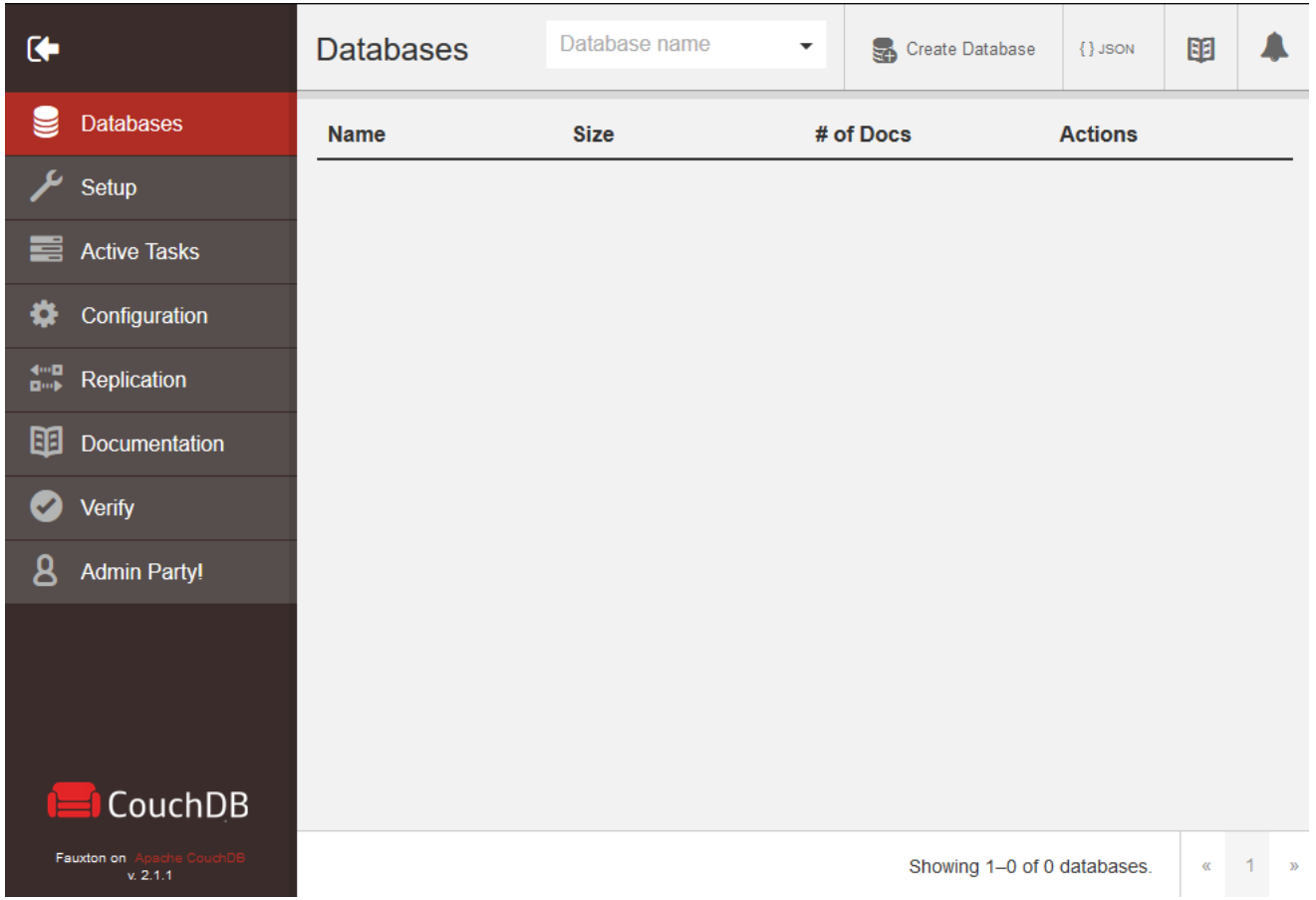
Dokumenti u CouchDB-u

- Dokument prima JSON objekat koji sadrži parove ključ/vrednost - polja
- Polja čija imena počinju donjom crtom imaju posebno značenje
- `_id` - identifikaciono polje, jedinstveno, ne menja se
- `_rev` - polje revizije, dobija novu vrednost svaki put kada se dokument modifikuje
- Brisanje i/ili modifikacija dokumenta zahteva oba polja
- Nema transakcija ili zaključavanja
- Prva operacija sa najažurnijom vrednošću polja `_rev` će se izvršiti

Futon (Fauxton)

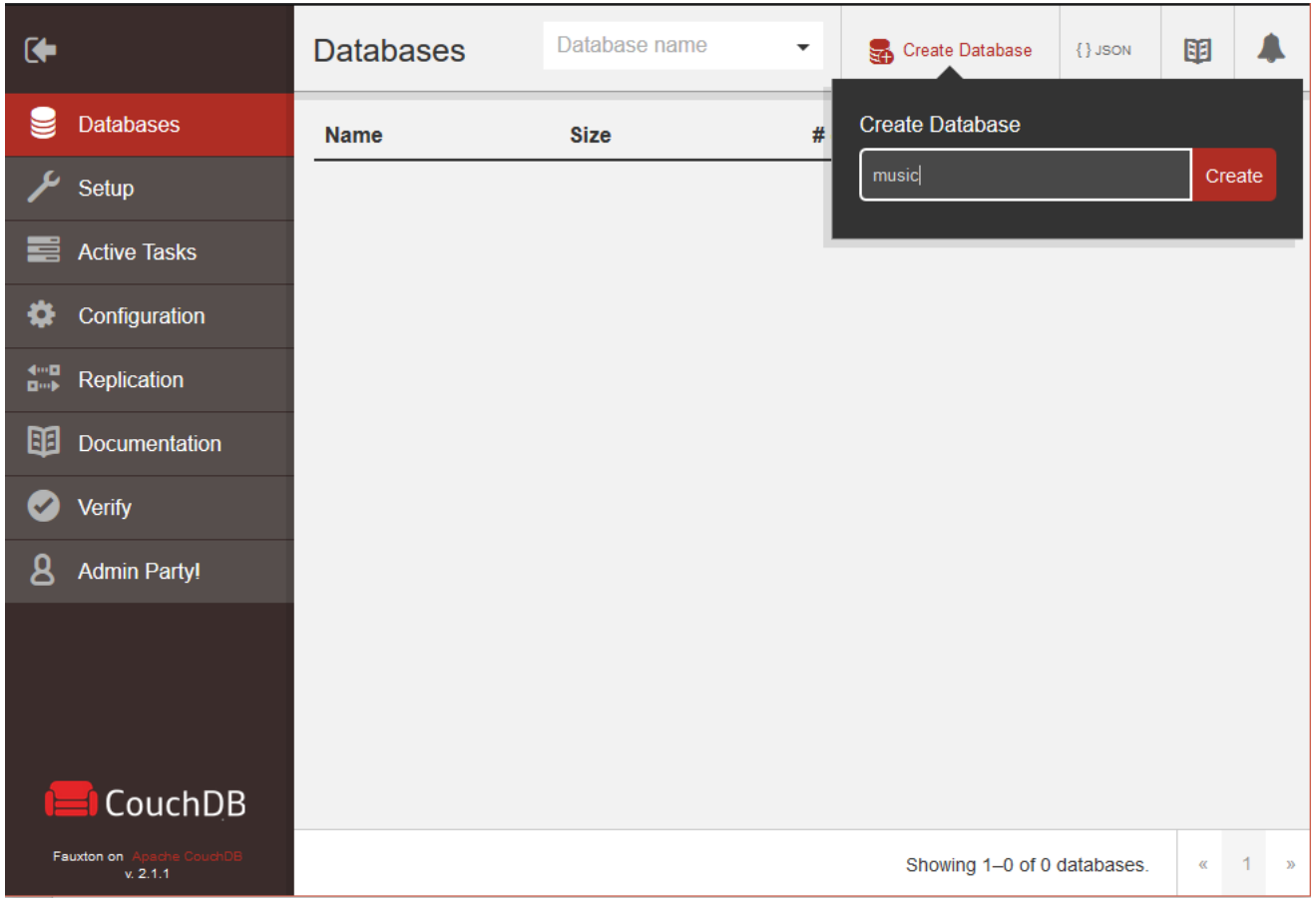
- Veb grafički interfejs koji olakšava rad sa bazom
- Potrebno je u internet pregledaču otvoriti adresu:

http://localhost:5984/_utils



Futon (Fauxton)

- *Create Database…*
- music
- *Create*



Futon (Fauxton)

- *New Document* – kreiranje novog dokumenta
- *Delete Document* – brisanje dokumenta
- U novi dokument dodajemo sledeće podatke i pamtimo promene:

```
"name": "The Beatles",
"albums": [ { "title": "Help!", "year": 1965 },
{ "title": "Sgt. Pepper's Lonely Hearts Club Band",
"year": 1967 },
{ "title": "Abbey Road", "year": 1969 } ]
```

REST API i HTTP CURL

- Sva komunikacija sa CouchDB je zasnovana na REST-u, što u praksi znači davanje komandi preko HTTP-a
- Komanda koja vraća informacije o serveru:

```
curl http://localhost:5984/
```

- Komanda koja vraća informacije o određenoj bazi:

```
curl http://localhost:5984/music/
```

Čitanje dokumenta sa GET

- READ = GET
- Da bi se povratio određeni dokument, potrebno je dodati njegov `_id` na URL do baze:

```
curl http://localhost:5984/music/my_document/
```

- U CouchDB-u, slanje GET zahteva je uvek bezbedno, jer GET zahtev neće izvršiti promene na dokumentu
- Da bi se izvršile promene, neophodno je iskoristiti druge HTTP komande, poput PUT, POST i DELETE

Kreiranje dokumenta sa POST

- `CREATE = POST`
- Za kreiranje novog dokumenta se koristi POST
- Dokument kreiran preko POST komande dobija pseudonasumični string kao `_id`
- Neophodno je u zaglavlju zahteva postaviti `Content-Type: application/json`; u suprotnom će CouchDB odbiti zahtev:

```
curl -X POST "http://localhost:5984/music/" \
-H "Content-Type: application/json" \
-d '{ "name": "Wings" }'
```

Ažuriranje dokumenta sa PUT

- UPDATE = PUT
- PUT komanda se koristi da ažurira postojeći dokument ili da kreira novi dokument sa određenom vrednošću `_id` polja

```
curl -X PUT "http://localhost:5984/music/my_document2" \
-H "Content-Type: application/json" \
-d '{ "name": "GNR" }'
curl -X PUT "http://localhost:5984/music/my_document2" \
-H "Content-Type: application/json" \
-d '{ "_rev": "<broj_revizije>", "name": "Guns N Roses" }'
```

- CouchDB ne može da promeni vrednosti određenog polja već prepisuje vrednosti celog dokumenta da bi napravio promenu
- Futon interfejs možda omogućava korisniku jednostavnu promenu vrednosti jednog polja, ali u pozadini se zapravo brišu sve vrednosti ključeva u dokumentu i ponovo postavljaju

Brisanje dokumenta sa DELETE

- Komanda DELETE briše dokument iz baze:

```
curl -X DELETE
"http://localhost:5984/music/my_document2/“ \
-H "If-Match: <broj_revizije>"
```

- DELETE operacija će obezbediti novi broj revizije, iako je dokument izbrisan
- Dokument nije zapravo izbrisan sa diska, već je prazan dokument nalepljen preko njega, označivši ga kao obrisanog

Pogledi

- Pogledi su primarni alat za postavljanje upita nad CouchDB dokumentima
- Mogu biti trajni i privremeni
- Trajni pogledi se čuvaju unutar specijalnih dokumenata koji se nazivaju **dokumenti dizajna (design documents)**
- Njima se može pristupiti slanjem HTTP GET zahteva na URI */{imebaze}/{dokumentid}/{imepogleda}*
- *{dokumentid}* ima prefix `_design/` da bi CouchDB prepoznao da se radi o dokumentu dizajna
- *{imepogleda}* ima prefix `_view/`, da bi CouchDB prepoznao da se radi o pogledu
- **Privremeni pogledi** se ne čuvaju u bazi, već se izvršavaju „ad hoc“
- Imaju loše performanse i ne postoje u novim verzijama CouchDB-a

Pogledi

- Najjednostavniji pogled je ugrađen, i naziva se `_all_docs`
- `_all_docs` prikazuje informacije za svaki dokument u bazi

```
curl http://localhost:5984/music/_all_docs
{"total_rows": 100, "offset": 0, "rows":
[ ... {"id": "123", "key": "123", "value":
{"rev":"14470fw4364gw..."}}, ... ]}
```

- Podrazumevano ponašanje je da pogledi ne prikazuju kompletne sadržaje dokumenata
- To se može postići dodavanjem `include_docs=true` parametra na zahtev:

```
curl http://localhost:5984/music/_all_docs?include_docs=true
```

MapReduce

- Pogled koristi MapReduce model koje se sastoji od funkcije mapiranje (map) i opcione funkcije redukovanja (reduce)
- On generiše uređenu listu parova ključ/vrednost, gde i ključevi i vrednosti moraju biti JSON formata
- Funkcija mapiranja može da izgleda ovako:

```
function(doc) {
        emit(key, value);
}
```

- Funkcija mapiranja prima parametar doc koji označava dokument
- Funkcija će se izvršiti jednom za svaki dokument u bazi podataka

Funkcija emitovanja

- Ključni deo funkcije mapiranja unutar svih pogleda je funkcija emitovanja:

```
emit(key, value);
```

- Funkcija emitovanja prima dva argumenta: ključ i vrednost
- Pogledi će se pretraživati po ključu koji je definisan u funkciji emitovanja
- Svako pozivanje funkcije emitovanja dodaje jednu stavku u rezultatima izvršavanja pogleda
- Funkcija emitovanja se može pozvati više puta za isti dokument

Kreiranje korisničkih pogleda

- Sada ćemo napisati pogled koji simulira ponašanje ugrađenog pogleda `_all_docs`
- Setimo se da ovaj pogled sadrži `_id` vrednost dokumenta kao ključ i `_rev` vrednost dokumenta kao vrednost funkcije
- Funkcija treba da izgleda ovako:

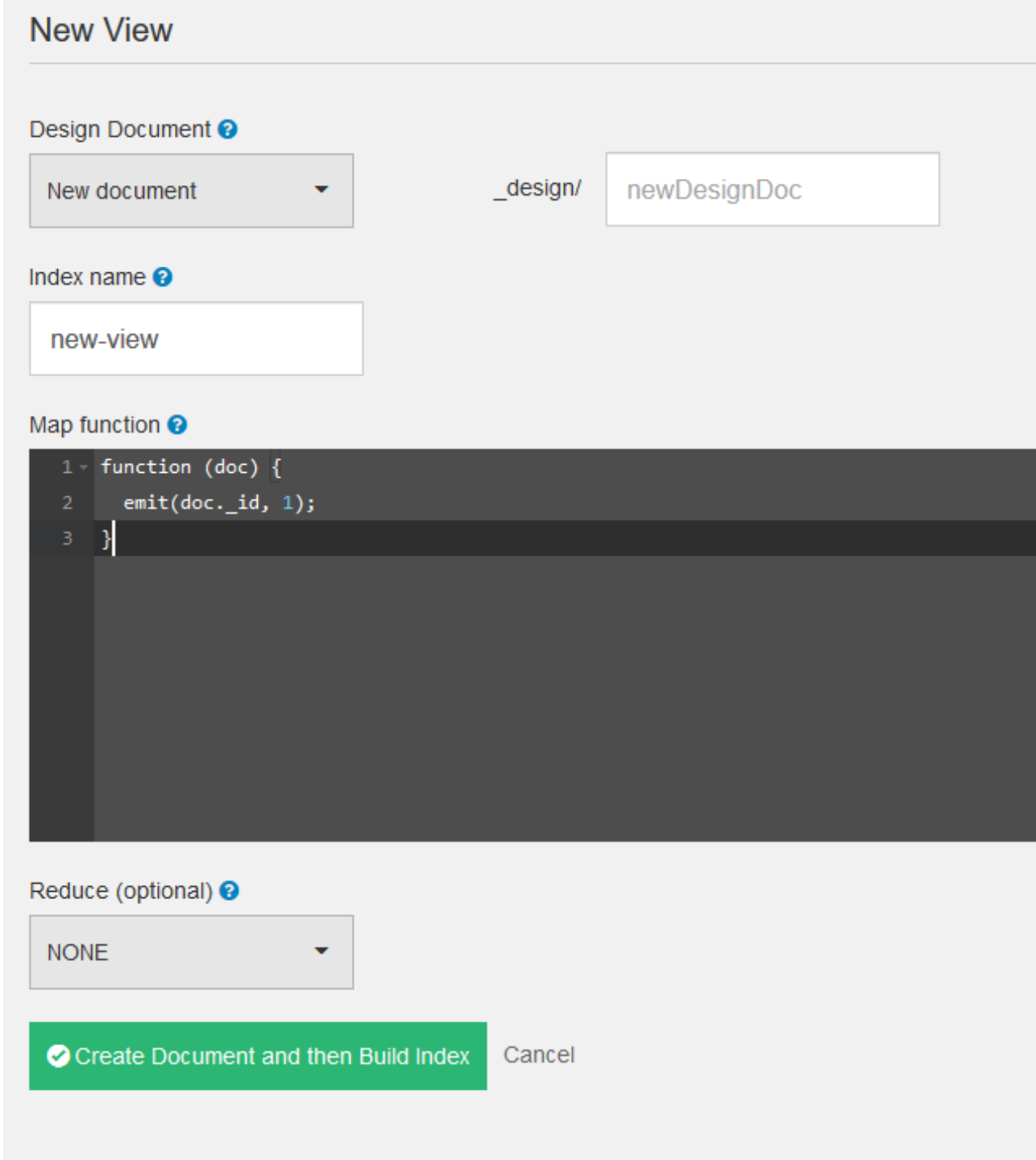
```
function(doc) {
        emit(doc._id, { rev: doc._rev });
}
```

Čuvanje trajnih pogleda

- Pogledi se čuvaju unutar **dokumenata dizajna**
- Dokumenti dizajna su kao i ostali dokumenti u bazi koji čuvaju podatke, osim što kao podatke čuvaju potpise funkcija pogleda i ostalih funkcija aplikacije
- Njihovi identifikacioni ključevi počinju sa `_design/` da bi CouchDB mogao da ih raspoznaje od običnih dokumenata
- Funkcije pogleda se čuvaju u polju sa vrednošću ključa `views`
- Dokument dizajna može sadržati jedan ili više pogleda

Kreiranje korisničkih pogleda

- Poglede je najlakše kreirati preko grafičkog okruženja
- Unose se ime dokumenta dizajna, ime pogleda, funkcija mapiranja i opciona funkcija redukovanja

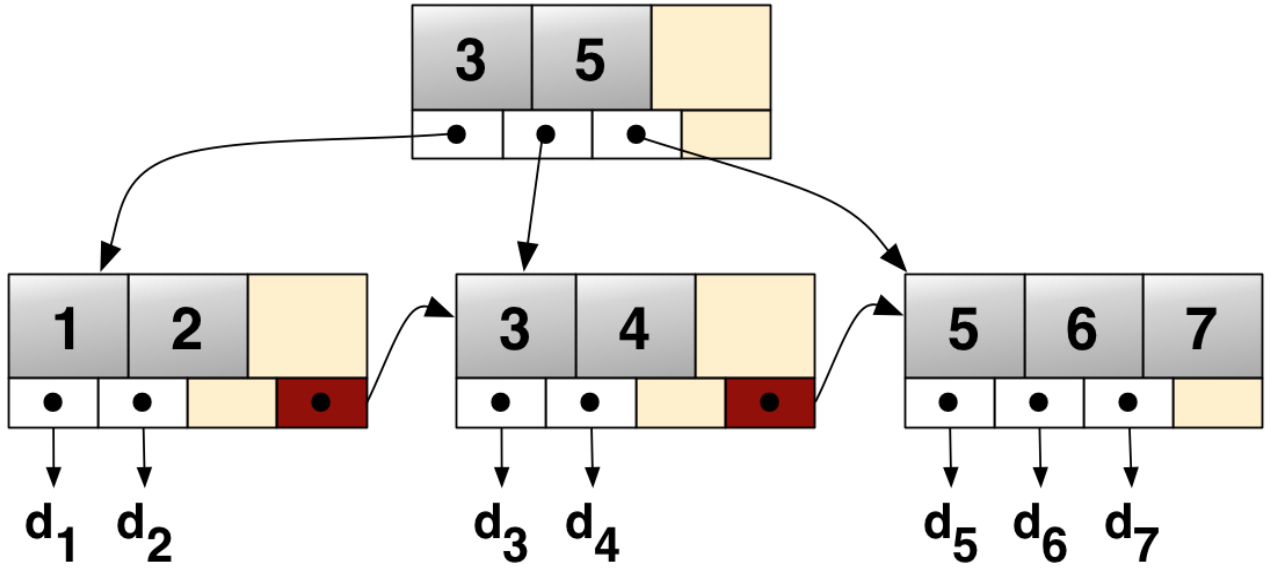


Indeksi

- Indeksi se kreiraju ili osvežavaju **kada se pogledu pošalje upit**
- Indeksi se **ne** osvežavaju nakon promena u bazi
- Nakon kreiranja pogleda, CouchDB će proći kroz svaki dokument tačno jednom i konstruisati B+ stablo sa rezultatima upita
- Svaki sledeći upit poslat pogledu će koristiti prethodno konstruisano B+ stablo
- Ukoliko se naprave novi dokumenti, ažuriraju ili izbrišu postojeći, pogled će izvršiti funkcije mapiranja i redukcije **samo za dokumente obuhvaćene promenama**
- Tom prilikom će se indeksi u B+ stablu ažurirati

B+ stabla

- B+ stablo može da se posmatra kao B stablo koje u internim čvorovima sadrži samo ključeve pretrage
- Poslednji nivo stabla je povezana lista koja sadrži pokazivače na dokumente



Kreiranje korisničkih pogleda

- Dokumentima u bazi je moguće pristupiti `_all_docs` komandom koristeći njihov `_id`, ali zanimljivije je vršiti pretragu prema imenima bendova
- Funkcija mapiranja proverava da li dokument sadrži sadrži `name` polje i, ako je tako, emituje to ime kao ključ, a `_id` polje emituje kao vrednost ključa
- Pogled ćemo sačuvati: ime dokumenta će se zvati `artists`, a ime pogleda `by_name`

```
function(doc) {
        if ('name' in doc) {
                emit(doc.name, doc._id);
        }
}
```

Slanje upita pogledima

- Kao što je već pomenuto, trajnim pogledima se može pristupiti slanjem HTTP GET zahteva na URI */{imebaze}/{dokumentid}/{imepogleda}*, gde *{dokumentid}* ima prefix `_design/` i gde *{imepogleda}* ima prefix `_view/`
- Pretraga umetnika po imenu:

```
curl http://localhost:5984/music/
_design/artists/_view/by_name
```

Parametri pretrage

- Pretraga umetnika po imenu sa navedenom vrednošću ključa:

```
curl 'http://localhost:5984/music/
_design/artists/_view/by_name?key="The Beatles"'
```

Parametri pretrage

- Pretraga umetnika sa parametrom `limit`
- `Limit` nam govori koliko ukupno rezultata želimo

```
curl http://localhost:5984/music/
_design/artists/_view/by_name?limit=5
```

Parametri pretrage

- Pretraga umetnika sa navođenjem početnog karaktera u ključu pretrage – koristi se parametar `startkey`:

```
curl http://localhost:5984/music/
_design/artists/_view/by_name?startkey=%22S%22
```

Parametri pretrage

- Pretraga umetnika sa navođenjem krajnjeg karaktera u ključu pretrage – koristi se parametar endkey:

```
curl http://localhost:5984/music/
_design/artists/_view/by_name?endkey=%22W%22
```

Parametri pretrage

- Pretraga umetnika sa sortiranjem ključa pretrage u obrnutom redosledu – koristi se parametar `descending=true`:

```
curl http://localhost:5984/music/
_design/artists/_view/by_name
?startkey=%22W%22\&endkey=%22S%22\&descending=true
```

Parametri pretrage

- Kada se koristi `descending=true` za pretragu, potrebno je zameniti vrednosti `startkey` i `endkey` parametara
- Razlog ovome je zato što CouchDB obilazi stablo pretrage u tzv. „pre-order“ maniru
- Pri obrtanju redosleda, neće se invertovati stablo, već samo **redosled čitanja čvorova**

Reduce

- Funkcije redukovanja (reduce) su opcione u pogledima i služe sa agregaciju rezultata dobijenih funkcijom mapiranja
- CouchDB izvršava sledeće korake:

1. Šalje sve dokumente funkciji mapiranja,
2. Sakuplja sve emitovane vrednosti,
3. Sortira emitovane redove po ključu,
4. Šalje redove sa istim ključem funkciji redukovanja,
5. Ako je podataka previše i sve redukcije se ne mogu obraditi u jednom pozivu funkcije, ponovo poziva funkciju redukovanja, ali ovaj put sa prethodno redukovanim vrednostima,
6. Rekurzivno poziva funkciju redukovanja dok je to neophodno, sve dok ne ostane bez duplikata ključeva

Reduce

- Primer redukcije koja sumira vrednosti:

```
function (key, values, rereduce) {
    return sum(values);
}
```

Reduce

- Poziv redukcije:

```
reduce (
[["key", id1], ["key", id2], ...], // kljucevi su isti
[123, 47, ...],                    // sve vrednosti su 1
false  // rereduce je false (netacan)
)
```

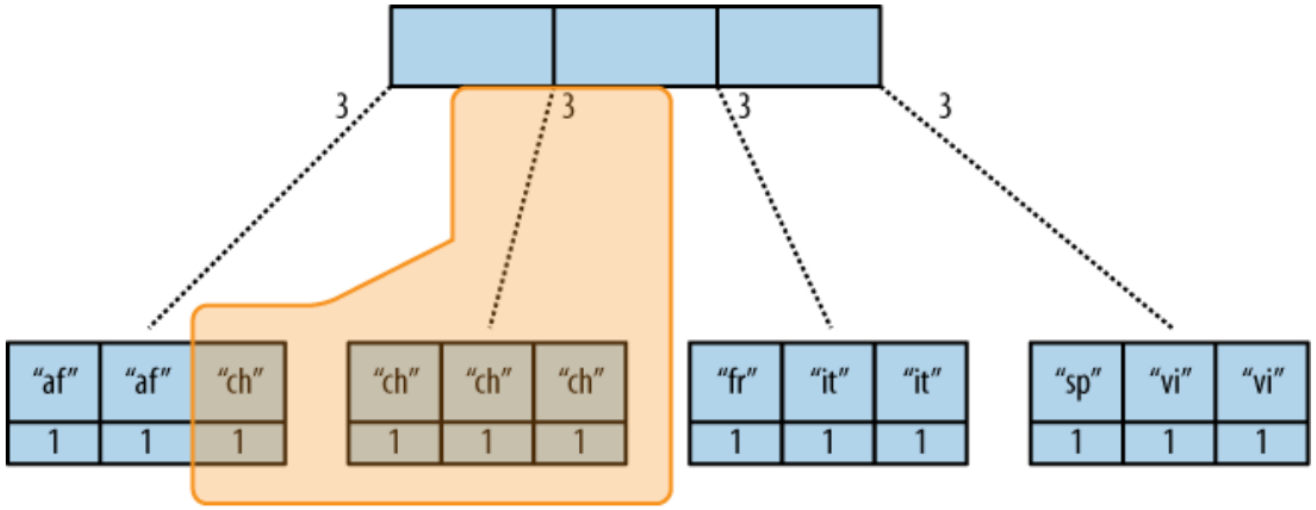
Reduce

- Poziv reredukcije:

```
reduce (
null,          // niz kljuceva je null
[104, 102, 87], // vrednosti niza su dobijene prethodnim pozivima
redukcije
true           // rereduce je true, znaci da se vrsi reredukcija
)
```

Reduce

- CouchDB ne odbacuje međurezultate redukcije, već i njih indeksira u stablu
- Često neće biti potrebno vršiti redukciju nad nekim delom stabla, već će biti dovoljno samo pročitati vrednost iz čvora koji se nalazi više u stablu

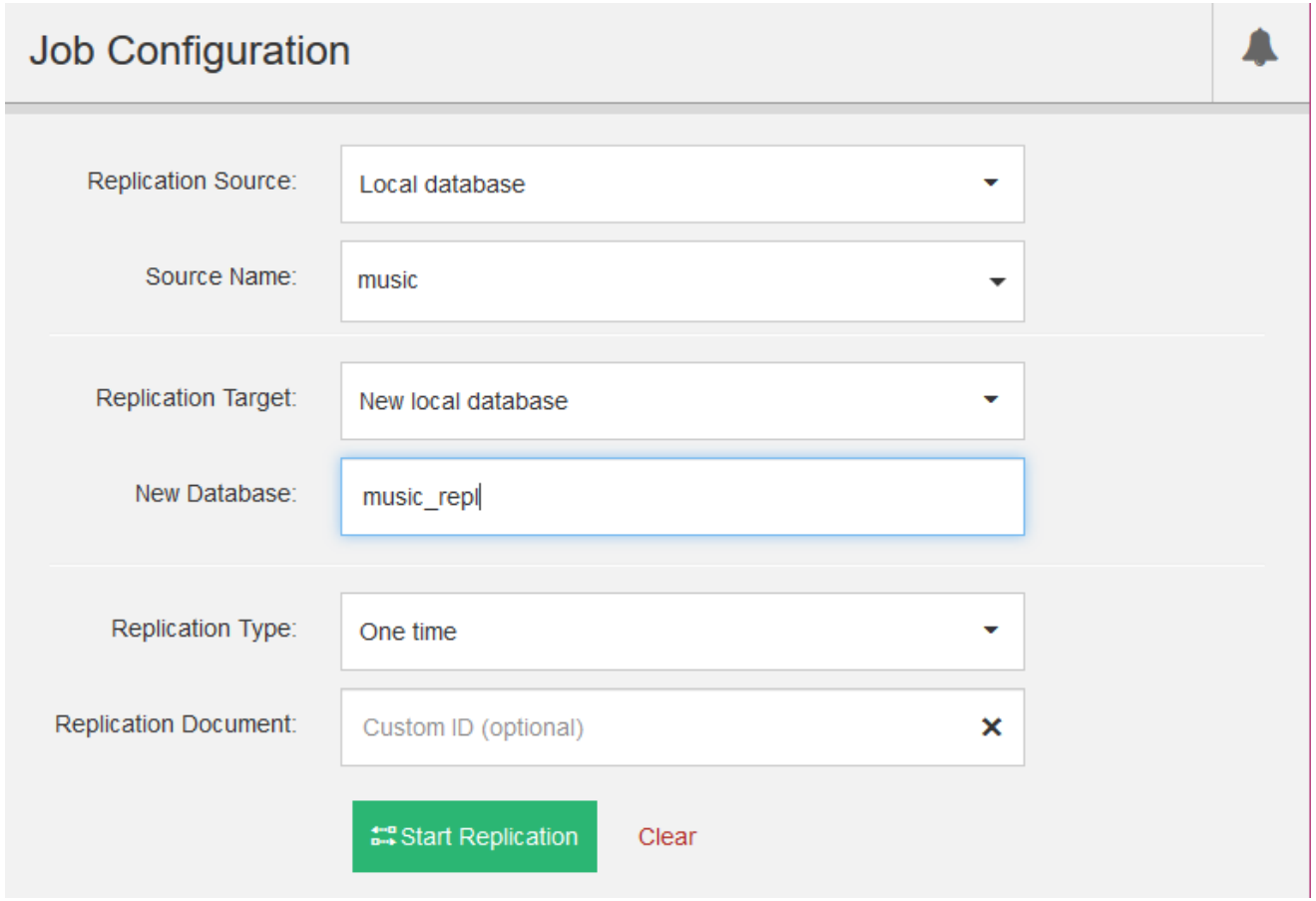


Master-master replikacija podataka

- Većina baza podataka pruža jedan glavni master čvor koji garantuje konzistentnost podataka ili se postiže kvorum između svih čvorova
- CouchDB ne radi ni jednu od tih stvari; umesto toga koristi više master čvorova ili **master-master replikaciju**
- Svaki CouchDB server, umesto dela podataka, sadrži sve podatke i može da prima zahteve klijenata, ažurira i briše podatke nezavisno od drugih servera
- U jednom trenutku, svi serveri mogu razmeniti podatke između sebe i rešiti eventualne konflikte koji nastaju usled konkurentnih promena nad istim podacima na različitim serverima

Replikacija preko Futon-a

- Odlazak na stranu željene baze i klik na *Replicator* dugme otvara stranu za replikaciju baze.
- Za bazu koju želimo replicirati odabraćemo *music*, dok ćemo drugu bazu nazvati *music_repl*



Master-master replikacija podataka

- Replikacija se odvija između dve baze
- Obe baze se mogu nalaziti ili u lokalu ili na nekom udaljenom serveru – za CouchDB je to nebitno jer je postupak replikacije u svim slučajevima isti
- Replikacija se može izvršiti i preko komandne linije, slanjem POST zahteva na `_replicate`:

```
curl -X POST "http://localhost:5984/_replicate" \
-H "Content-Type: application/json“ \
-d '{ "source": "http://localhost:5984/music",
"target": "http://localhost:5984/music_repl",
"create_target": true }'
```

- Kopiranje baza je jednosmerno
- Ukoliko želimo da kopiramo promene u oba smera, potrebno je u novom zahtevu zameniti `source` i `target` vrednosti

Kreiranje konflikata

- U jednom trenutku, isti podaci na različitim bazama će se različito ažurirati i pri sledećoj replikaciji će nastati konflikt
- Kako CouchDB rešava konflikte?
- Uneti primer na kome će se demonstrirati replikacija:

```
curl -X PUT "http://localhost:5984/music/theconflicts" \
-H "Content-Type: application/json" \
-d '{ "name": "The Conflicts" }'
```

- Sada treba ponovo pokrenuti replikaciju između dve baze:

```
curl -X POST "http://localhost:5984/_replicate" \
-H "Content-Type: application/json‟ \
-d '{ "source": "http://localhost:5984/music",  "target": "http://localhost:5984/music_repl"}'
```

Kreiranje konflikata

- Možemo proveriti da li je replikacija bila uspešna slanjem zahteva `music_repl` bazi:

```
curl "http://localhost:5984/music_repl/theconflicts"
```

Kreiranje konflikata

- Sada ćemo ažurirati dokument u `music_repl` bazi dodavanjem novog albuma:

```
curl -X PUT "http://localhost:5984/music_repl/theconflicts" \
-H "Content-Type: application/json" \
-d '{ "_id": "theconflicts", "_rev": "<broj_revizije>",
"name": "The Conflicts", "albums": ["Conflicts of Interest"] }'
```

- U bazi `music` ćemo napraviti konfliktno ažuriranje dodavanjem drugačijeg albuma:

```
curl -X PUT "http://localhost:5984/music/theconflicts" \
-H "Content-Type: application/json" \
-d '{ "_id": "theconflicts", "_rev": "<broj_revizije>",
"name": "The Conflicts", "albums": ["Conflicting Opinions"] }'
```

Kreiranje konflikata

- Sada `music` i `music_repl` baze podataka imaju dokument sa `_id` vrednošću `theconflicts`
- Oba dokumenta su verzije 2 i imali su istu osnovnu reviziju
- Šta će se desiti ako pokrenemo replikaciju između ove dve baze?

```
curl -X POST "http://localhost:5984/_replicate" \
-H "Content-Type: application/json“ \
-d '{ "source": "http://localhost:5984/music",
"target": "http://localhost:5984/music_repl"}'
```

Razrešenje konflikata

- Rezultati replikacije mogu biti iznenađujući zato što će se replikacija izvršiti u potpunosti i bez greške
- CouchDB zapravo nema mehanizme da odredi koji podaci su „ispravni“, a koji nisu, već bira jedan dokument i proglašava ga pobednikom
- Algoritam određivanja pobednika je deterministički, što znači da će za iste instance konflikta uvek davati istog pobednika
- Dokument koji je „izgubio“ se ne briše, već se čuva da bi korisnik ili klijentska aplikacija kasnije mogli da preprave podatke kako im odgovara i razreše konflikt
- Informacije o konfliktnim revizijama:

```
curl http://localhost:5984/music_
repl/theconflicts?conflicts=true
```

Razrešenje konflikata

- Odgovor ima i `_conflicts` polje, koje sadrži listu revizija koje su u konfliktu sa glavnom revizijom dokumenta
- GET zahtevu možemo priključiti ime konfliktne revizije da bismo videli konfliktne podatke i odlučili šta želimo da radimo sa njima:

```
curl http://localhost:5984/music-repl/ \
theconflicts?rev=<broj revizije>
```

- CouchDB neće pokušavati da inteligentno reši konflikte
- Primeri iz realnog sveta neće uvek biti jednostavni ili očigledni i zato CouchDB ostavlja korisniku da sam svoje podatke dovede u red onako kako želi

Prednosti CouchDB-a

- CouchDB je stabilan i robustan član NoSQL zajednice
- Zasnovan na filozofiji da su mreže nepouzdane i otkaz hardvera neizbežan
- Nudi poseban decentralizovan pristup čuvanju podataka
- Dovoljno mali za mobilni telefon i dovoljno veliki za preduzeće, CouchDB je dovoljno raznovrstan za sve situacije.

Mane CouchDB-a

- MapReduce pogledi, iako dobri, ipak ne mogu da podrže baratanje sa podacima na način na koji to čine relacione baze podataka
- Strategija replikacije nije uvek pravi izbor jer ide na sve ili ništa, što znači da će svi replicirani serveri imati iste podatke
- Ne postoji particionisanje i raspodela podataka na više čvorova
- Glavni razlog za dodavanje novih čvorova nije raštrkivanje podataka, već postizanje većeg protoka operacija čitanja i pisanja
- **Ne podržava transakcije i zaključavanja**

Hvala na pažnji