UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

Sistemi baza podataka

MODELOVANJE U DOKUMENT-ORIJENTISANIM BAZAMA PODATAKA

Novi Sad, 2026.

SADRŽAJ

- **Dokument-orijentisane baze podataka**
- Projektovanje logičke šeme BP
- Dizajn šabloni za projektovanje logičke šeme baze podataka

Dokument-orijentisane BP

- Jedna od najznačajnijih kategorija **NoSQL** baza podataka.
- Baze podataka koje podatke čuvaju u vidu **dokumenata**.
  - Dokumenti predstavljaju centralni koncept – svaki zapis (record) predstavljen je jednim dokumentom.
- Neki od najčešće korišćenih formata dokumenata jesu **JSON** (BSON), XML i YAML.
- DOBP skladišta dokumenata su vrlo **fleksibilna** – dobro rukuju **polustrukturiranim** i nestrukturiranim podacima.
  - Dobar su izbor kada nije unapred poznato kakva će biti struktura podataka koji će se skladištiti.

Oblasti primene

- Pri radu sa **nestrukturiranim** i **polustrukturiranim** podacima
  - *Content management* (npr. *E-Commerce*),
  - *Single View Applications,*
  - logovanje podataka,
  - *Internet of Things* (*IoT*),
  - i tako dalje.
- Pri ubrzanom razvoju prototipova (ne zahteva upotrebu migracija pri izmeni šeme bp).

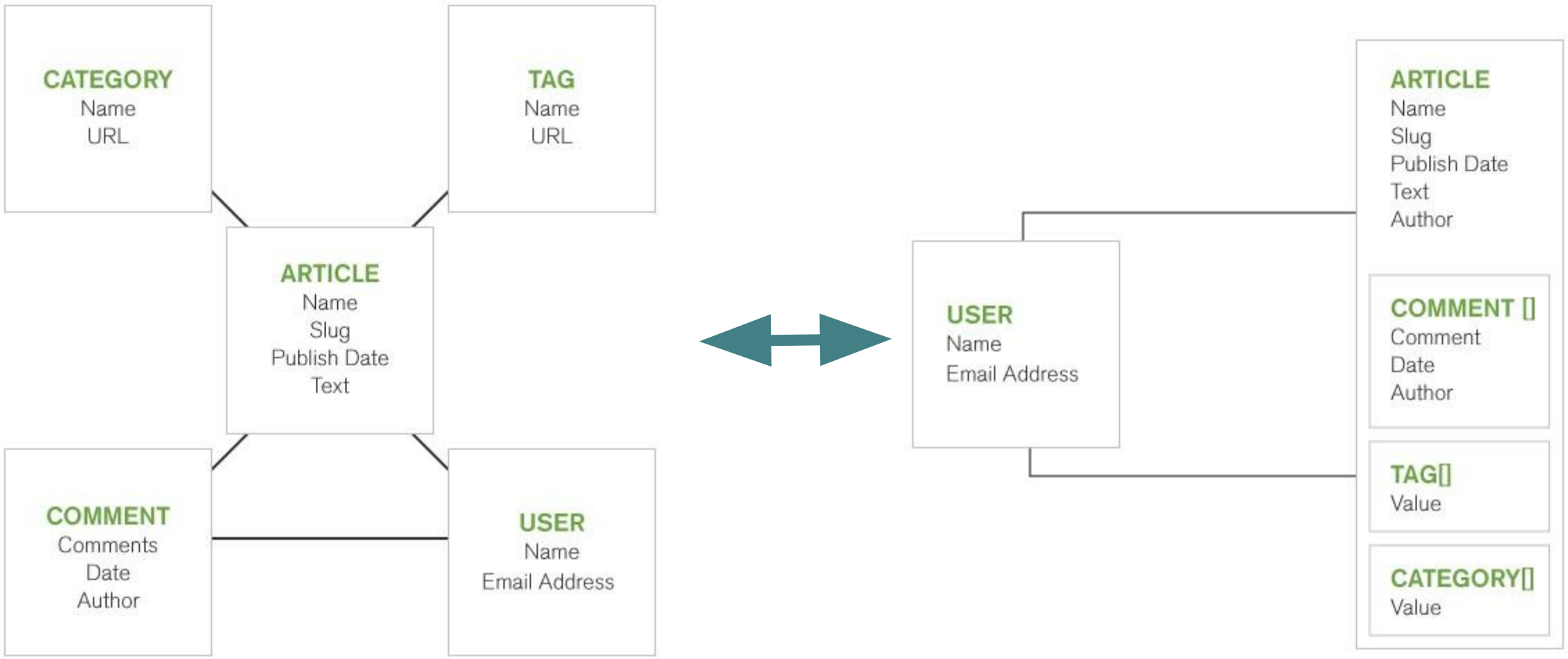
Šema podataka

- DODB najčešće ne zahtevaju postojanje predefinisane šeme podataka
  - Ne postoji jezik za definisanje šeme podataka (*Data Definition Language*) – kolekcije ne nameću strukturu dokumentima.
  - Svaki dokument u okviru jedne kolekcije može imati različit skup polja – šema je fleksibilna/dinamička.
- Iako se šema u ne deklariše eksplicitno, aplikacije u najvećem broju slučajeva koriste podatke na način kao da šema postoji.
  - Postoje mehanizmi za validaciju šeme na nivou kolekcije.
- Bitna napomena: način na koji se čuvaju podaci – (**logička) šema baze podataka** – ima jako veliki uticaj na **performanse sistema**.
  - **“ The hard fact is that most performance issues we’ve found trace back to poor schema design.“**

Aplikacijom vođena šema (*Application Driven Schema*)

- Naglasak se stavlja na šablone pristupa podacima (*data access patterns*), odnosno, na način na koji aplikacija koristi podatke.
- Podaci se skladište tako da se podrži njihovo efikasno korišćenje (***latency*** značajno skuplji od memorije) – podaci koji se često koriste zajedno se ne razdvajaju, kako bi se izbegla potreba za spajanjem podataka.
  - **Podaci se namerno denormalizuju**, što izaziva duplikaciju podataka.
  - Anomalije ažuriranja se izbegavaju načinom na koji se koriste podaci.
    - Anomalija modifikacije se izbegava retkom modifikacijom dupliciranih podataka.
    - Anomalije upisa i brisanja se izbegavaju tako što se ugnježdavaju samo podaci o egzistencijalno zavisnim TE.
  - MongoDB dugo nije ni podržavao mehanizam spajanja (*join*). Kasnije uveden operator *$lookup*, koji nameće različita ograničenja pri spajanju.
- **Velika odgovornost pri projektovanju logičke šeme baze podataka!**

Poređenje relacione i dokument-orijentisane šeme



Izvor: *https://www.mongodb.com/blog/post/thinking-documents-part-1*

SADRŽAJ

- Dokument-orijentisane baze podataka i MongoDB
- **Projektovanje logičke šeme BP**
- Dizajn šabloni za projektovanje logičke šeme baze podataka

Projektovanje logičke šeme bp

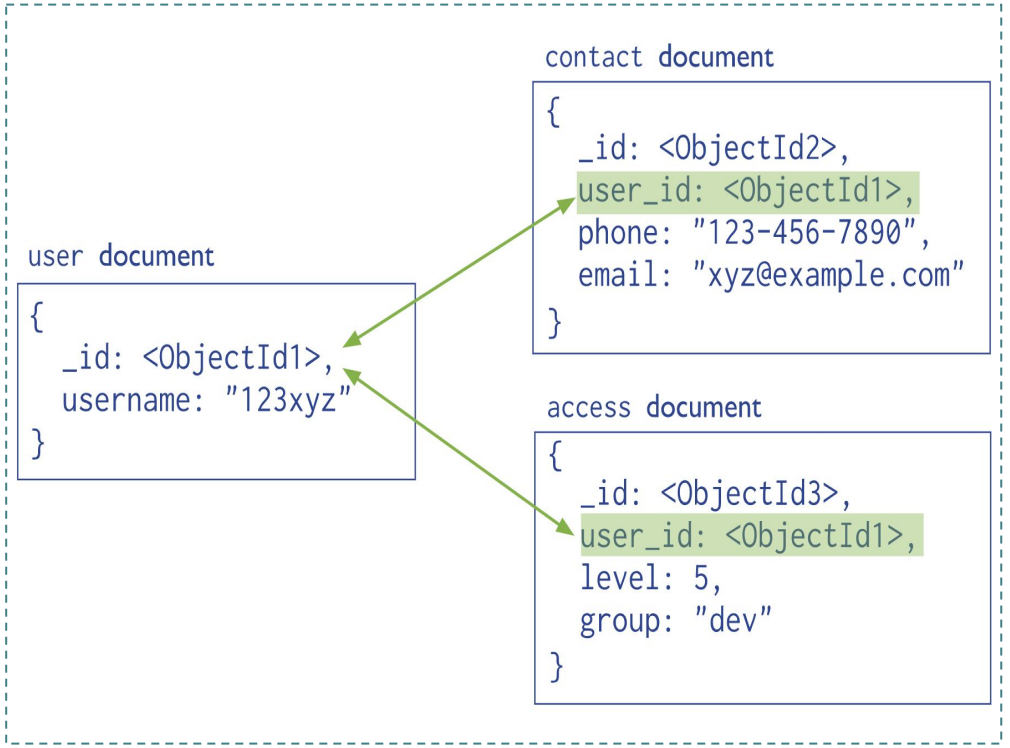
- Najčešći razlog za probleme u performansama kod dokument-orijentisanih baza podataka je loše projektovana logička šema bp.
- **Kako odabrati šemu koja najviše odgovara aplikaciji?** Zavisi od odgovora na sledeća pitanja:
  - **Koji su najčešći šabloni pristupa podacima (*data access patterns*) u aplikaciji?**
    - **Koji podaci se uvek koriste zajedno pri čitanju iz baze?**
    - **Čemu se najčešće pristupa?**
    - **Da li se podaci često ažuriraju?**
  - **Koja je očekivana veličina dokumenta? (1kB ili 10MB) Kako će se veličina dokumenta kretati u budućnosti?**
  - **Kako će se kretati broj dokumenata i količina podataka u budućnosti? Kakav to uticaj ima na skalabilnost i performanse?**

Projektovanje logičke šeme bp

- Kod relacionog modela podataka, normalizacija predstavlja smernicu ka „dobro“ dizajniranoj šemi.
- Kod dizajniranja dokumenata, teži se ka **kontrolisanoj denormalizaciji** – neke podatke ćemo razdvojiti – normalizovati, dok ćemo neke podatke spajati – denormalizovati.
  - Pri normalizaciji, veze između dokumenata očuvavaju se korišćenjem **referenci**.
  - Pri denormalizaciji, veze između dokumenata očuvavaju se **ugnježdavanjem** dokumenata.
- Pri projektovanju šeme, najbitnije je razmotriti da li za očuvanje veze između dokumenata upotrebiti referencu ili ugnježdavanje.

Reference

- Pri normalizaciji, veze između dokumenata očuvavaju se korišćenjem referenci.
  - Reference se mogu održavati ručno (preporučeni način) ili pomoću *DBRef* mehanizma.
- Uopšteno govoreći, reference bi trebalo koristiti:
  - kada bi se ugnježdavanjem dobila duplikacija podataka, a da pritom performanse čitanja nisu zadovoljavajuće u odnosu na implikacije duplikacije;
  - kako bi se predstavile kompleksnije veze više-prema-više (M:N) tipa;
  - kako bi se predstavili veliki skupovi podataka hijerarhijske strukture.



- Reference pružaju više fleksibilnosti u odnosu na ugnježdavanje.
- Međutim, usled slabe podrške za spajanje dokumenata, kompleksniji upiti mogu zahtevati višestruku komunikaciju sa bazom – **značajan uticaj na performanse**.

Ugnježđavanje dokumenata

- Pri denormalizaciji, veze između dokumenata očuvavaju se ugnježđavanjem dokumenata.
  - Na ovaj način koriste se prednosti JSON formata.
- Uopšteno govoreći, ugnježđavanje bi trebalo koristiti:
  - kada je dokument egzistencijalno zavistan;
  - kako bi se predstavile veze jedan-prema-više (1:N) ili jedan-prema-jedan (1:1) tipa;
  - kako bi se podaci koji se uvek koriste zajedno i čuvali zajedno.

- U opštem slučaju, ugnježđavanje pruža bolje performanse pri čitanju, zbog mogućnosti da se potrebni podaci dobave pomoću jednog poziva upućenog ka bazi podataka.
- Neophodno je voditi računa o ograničenju maksimalne veličine dokumenta od 16MB.
- Ugnježđavanje podataka omogućava ažuriranje povezanih podataka u okviru jedne **atomične operacije**.

Atomičnost operacija

- Sve operacije pisanja u MongoDB bazu podataka su **atomične na nivou jednog dokumenta**.
  - Kada jednom operacijom pisanja modifikujemo nekoliko dokumenata, modifikacija svakog dokumenta je atomična operacija, ali operacija pisanja u celini nije atomična – može doći do konflikata.
  - Od verzije 4.0 uvedena podrška i za *multi-document* ACID transakcije, ali njihovo korišćenje ima uticaj na performanse.

SADRŽAJ

- Dokument-orijentisane baze podataka i MongoDB
- Projektovanje logičke šeme BP
- **Dizajn šabloni za projektovanje logičke šeme baze podataka**

Dizajn šabloni za projektovanje logičke šeme BP

- Kako bi se olakšalo projektovanje logičke šeme, MongoDB razvojni tim osmislio je 11 dizajn šablona.
- Ovi šabloni predstavljaju **smernice** *(building blocks)* za projektovanje logičke šeme baze podataka, čijim bi korišćenjem trebalo da se postignu dobre performanse pri korišćenju baze podataka.
  - Uvođenje metodologije u projektovanje.
- Koji šablon će biti korišćen, zavisi od prirode podataka koji se koriste u aplikaciji, kao i od zahteva same aplikacije – aplikacijom vođena šema.
- Pri projektovanju, najbitnije je postići dobre performanse uz podršku skalabilnosti sistema, a da se pri tome očuva jednostavnost šeme.
  - Dizajn šabloni su kreirani kako bi se olakšalo dostizanje ovog cilja.

Dizajn šabloni za projektovanje logičke šeme baze podataka

- 11 dizajn šablona:
  - **atributa** (*Attribute*),
  - **baketiranja** (*Bucket*),
  - **proširene reference** (*Extended reference*),
  - **proračunavanja** (*Computed*),
  - **stabla** (*Tree*),
  - **aproksimacije** (*Approximation*),
  - **verzionisanja dokumenata** (*Document Versioning*),
  - **polimorfnih dokumenata** (*Polymorphic*),
  - **autlajera** (*Outlier*),
  - **verzionisanja šeme** (Schema Versioning),
  - **podskupa** (*Subset*).

Šablon atributa

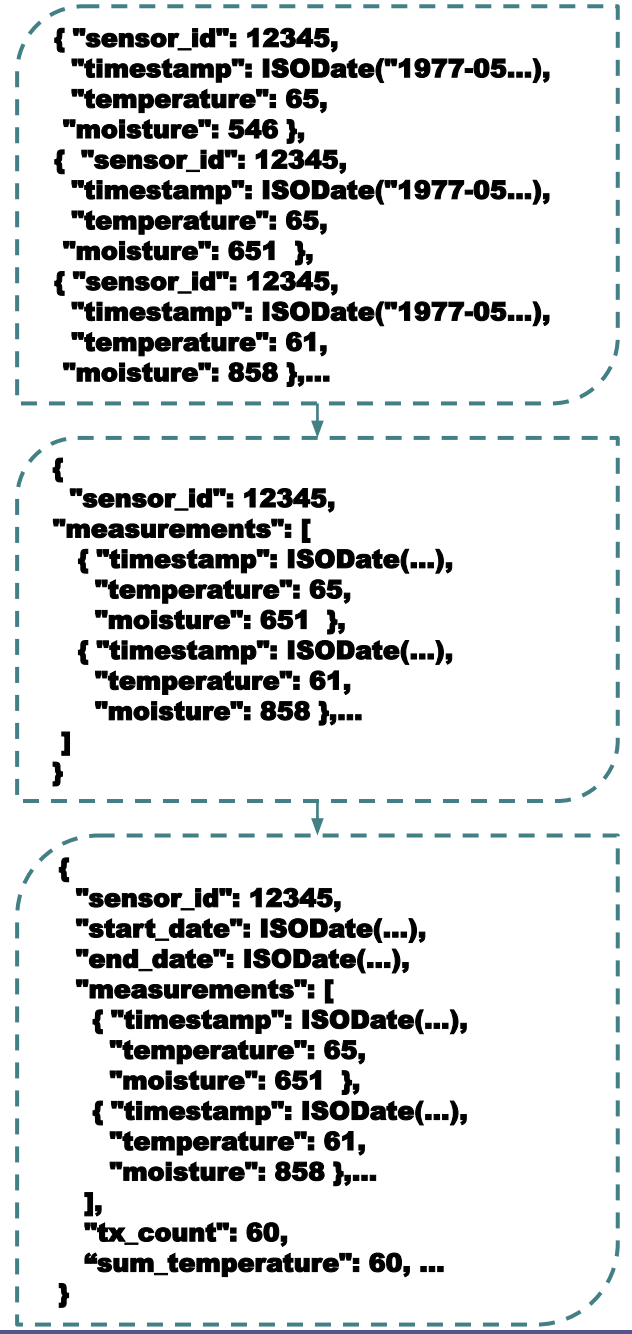
- Koristi se u situacijama kada svaki dokument ima mnogo sličnih atributa po kojima bismo želeli da vršimo pretragu ili sortiranje, kao i u situacijama kada se atribut jako retko javlja, a da po njemu želimo da vršimo sortiranje.
  - Problem bi predstavljala pretraga po takvim atributima, jer bi za efikasnost pretrage bilo neophodno izgraditi indeks nad svakim takvim atributom – veliki broj indeksa.
- Kako bismo smanjili potrebu za indeksima, ovakve atribute možemo pretvoriti u niz parova ključ-vrednost, gde se onda indeks gradi samo nad ključevima niza.
- Korišćenjem šablona atributa, redukuje se broj potrebnih indeksa, upiti se pojednostavljuju i povećava se brzina upita.

```
{
   "title": "Star Wars",
   "director": "George Lucas",
   ...
  " release_US ": ISODate("1977-05…),
  " release_UK": ISODate("1977-12…),
  " release_Italy": ISODate("1977-10…),
  " release_France": ISODate("1977…),
   ...
}
```

```
{
   "title": "Star Wars",
   "director": "George Lucas",
   ...
   "releases": [
 {"loc": “USA", “date": ISODate…},
 {"loc": “UK", “date": ISODate…},
 {"loc": “Italy", “date": ISODate…},
 ...
],
   ...}
```

Šablon baketiranja

- Prilikom obuhvata podataka sa senzora (*time-series data - IoT*), opcija da se svako očitavanje skladišti kao zaseban dokument u kolekciji može predstavljati lošu projektantsku odluku.
  - Zbog velikog broja malih dokumenata, indeksne strukture mogu postati memorijski zahtevne, a i njihova selektivnost može biti problematična.
- Baketiranjem podataka smanjila bi se količina dokumenata.
  - Kako bi se izbegli glomazni dokumenti, može se koristiti baketiranje uz oslonac na vremenske okvire.
  - Pre-agregacijom se podaci mogu pripremiti za analizu.

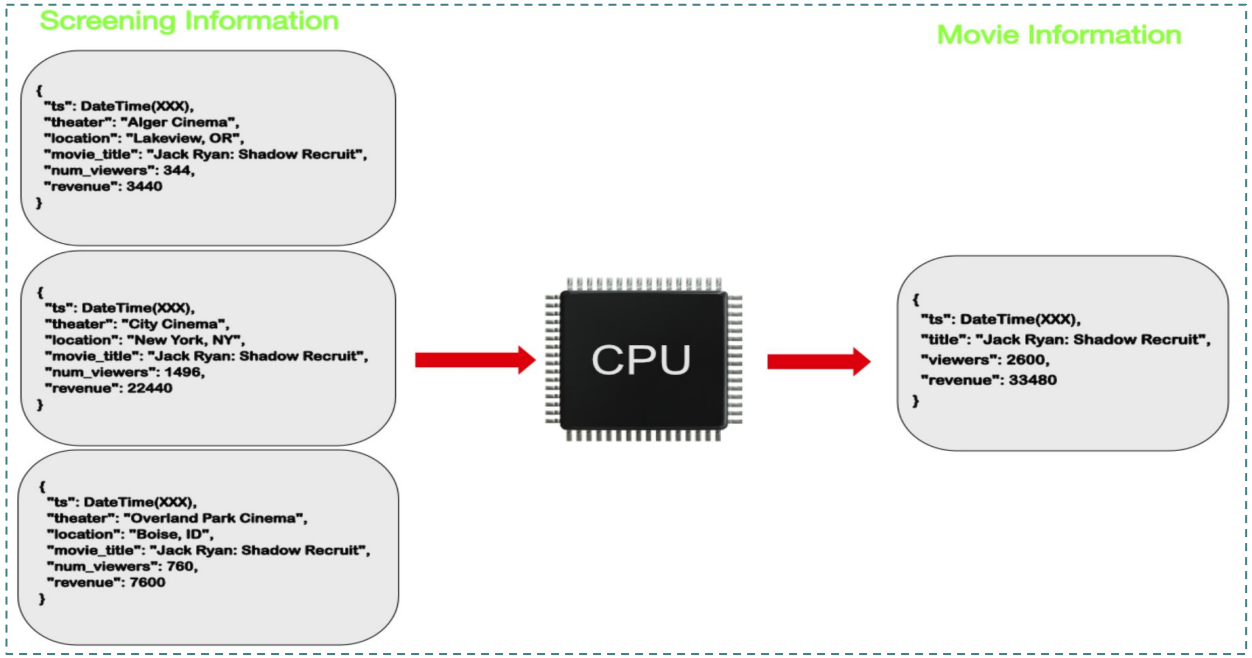


Šablon proširene reference

- Ukoliko prilikom rukovanja dokumentima često dolazi do potrebe za spajanjem, to može imati negativan uticaj na performanse sistema.
- Ugnježdavanjem svih informacija logički razdvojenih dokumenata radi smanjenja broja spajanja može se izazvati prevelika redundansa podataka.
- Umesto potpunog ugnježdavanja, šablon proširene reference u takvim situacijama određuje proširenje proste reference **podacima zbog kojih se najčešće vrši spajanje**.
- Treba izbegavati prosleđivanje atributa čija se vrednost često menja.

Šablon proračunavanja

- Šablon koji se koristi u situacijama kada se nad podacima vrše česta i zahtevna proračunavanja kako bi se izvukli izveštajni podaci.
- Cilj je da se smanji frekvencija takvih operacija čuvanjem rezultata proračuna.
- Uz rezultate se može čuvati i vremenska odrednica, kako bi bilo moguće proceniti kada je neophodno ponoviti proračun zbog zastarelosti sumarnih podataka.



Šabloni stabla

- Cilj šablona stabla je da se izbegnu česta spajanja prilikom obilaska dokumenata organizovanih u hijerarhijsku strukturu.
- Umesto da svaki dokument sadrži samo niz referenci direktno nadređenih ili direktno podređenih dokumenata, dokumenti sadrže konkretne vrednosti za sve pretke ili potomke u hijerarhiji.
- Polazi se od pretpostavke da je hijerarhija retko promenljiva.

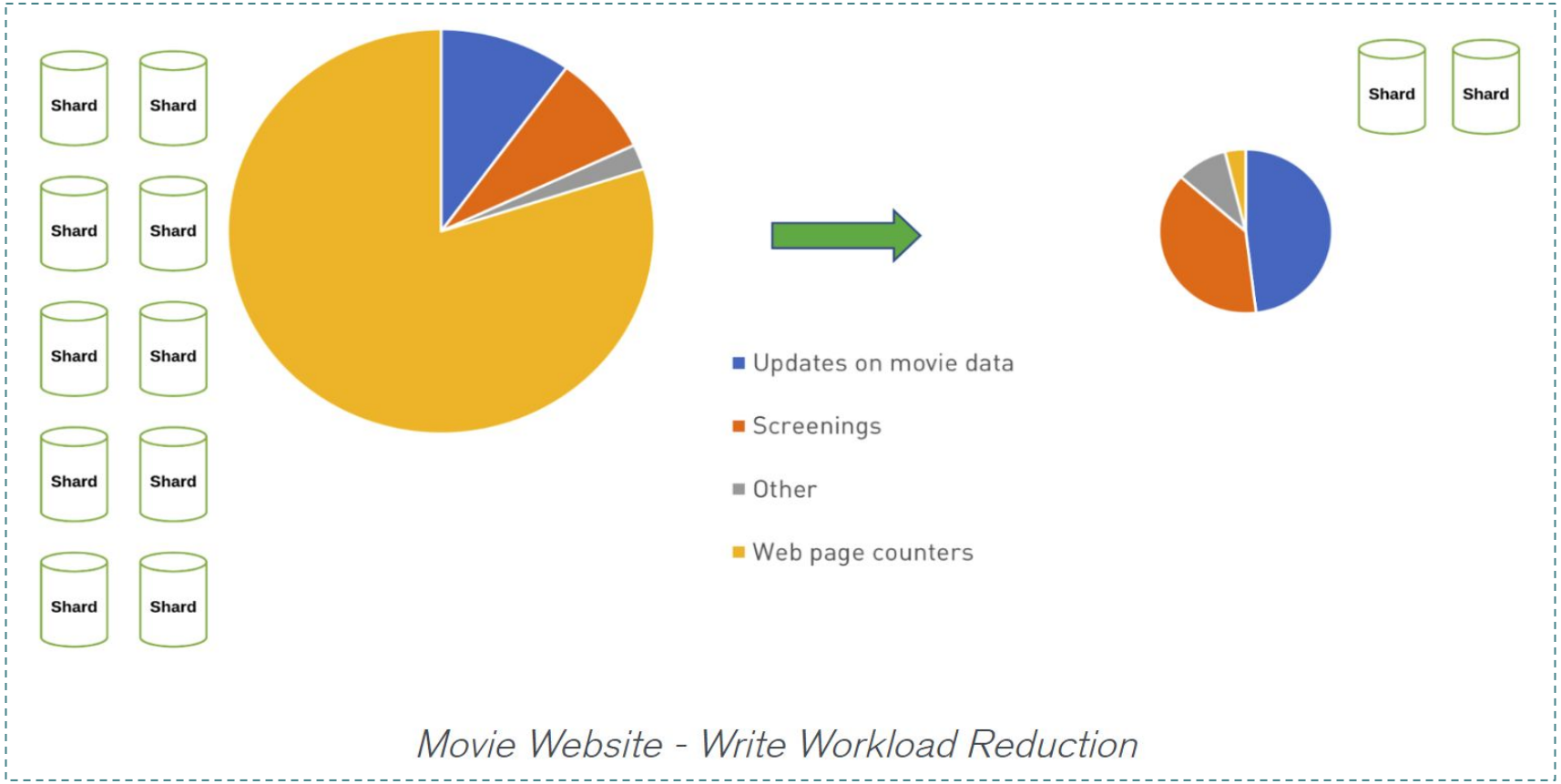
```
{
  "_id": 12345,
  "name": "Samsung EVO 1TB",
  "price": {
    "value":  585.00,
    "currency": "USD"
  },
  "parent_category": "Solid State
Drives"
  "ancestor_categories": [
    "Solid State Drives"
    "Hard Drives",
    "Storage",
    "Computers ",
    "Electronics"
  ]
}
```

Šablon aproksimacije

- Pri radu s velikom količinom podataka ili velikim brojem korisnika, **operacije pisanja** u bazu podataka mogu imati veliki uticaj na performanse sistema.
  - Što se više skalira na gore, to je uticaj operacija pisanja na performanse veći.
- Šablon aproksimacije koristi se u situacijama kada nam nije neophodno da znamo konkretnu vrednost atributa, već je prihvatljiva i aproksimacija stvarne vrednosti.
  - Primer: često nam nije neophodno da znamo tačan broj posetilaca sajta (Da li je u tom slučaju razlika između 700.000 i 699.958 bitna?).
  - Ako bismo vrednost u bazi uvećavali za 100 u 1% slučajeva, umesto za 1 u 100% slučajeva, smanjili bismo broj operacija pisanja za 99%.
- Mehanizam koji se okida kada se dostigne određeni broj pristupa, nakon kog sledi upis u bazu, implementira se na aplikativnom sloju.

Šablon aproksimacije

- Šablon aproksimacije može imati jako veliki uticaj na poboljšanje performansi sistema.



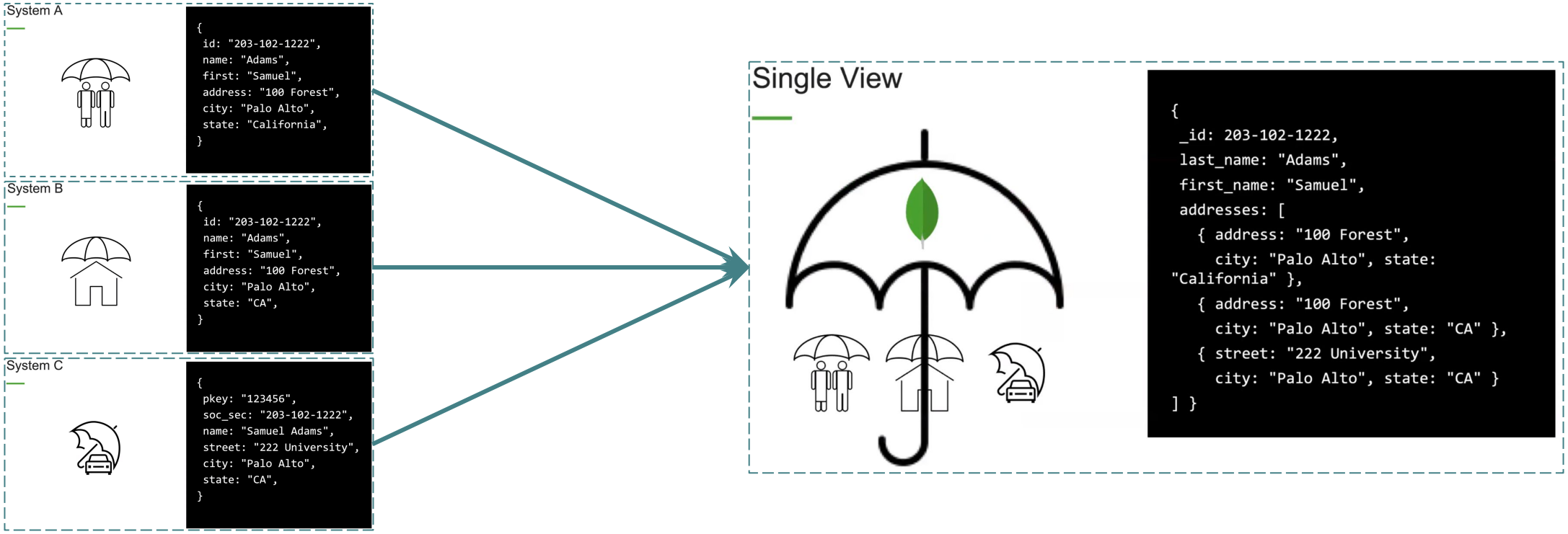
Izvor: https://www.mongodb.com/blog/post/building-with-patterns-the-approximation-pattern

Šablon verzionisanja dokumenata

- Koristi se kada je neophodno sačuvati prethodne verzije dokumenata.
- Zasniva se na dodavanju polja za verzionisanje u svaki dokument, kao i na razdvajanju trenutno važećih i prethodnih verzija u različite kolekcije radi očuvanja performansi.
- Polazi od sledećih pretpostavki:
  - Dokumenti neće imati veliki broj revizija.
  - Ne skladišti se veliki broj različitih dokumenata.
  - Većina upita se izvršava nad trenutno važećim verzijama dokumenata.
- U zavisnosti od potrebe, moguće je ograničiti broj prethodnih verzija koje se skladište.

Šablon polimorfnih dokumenata

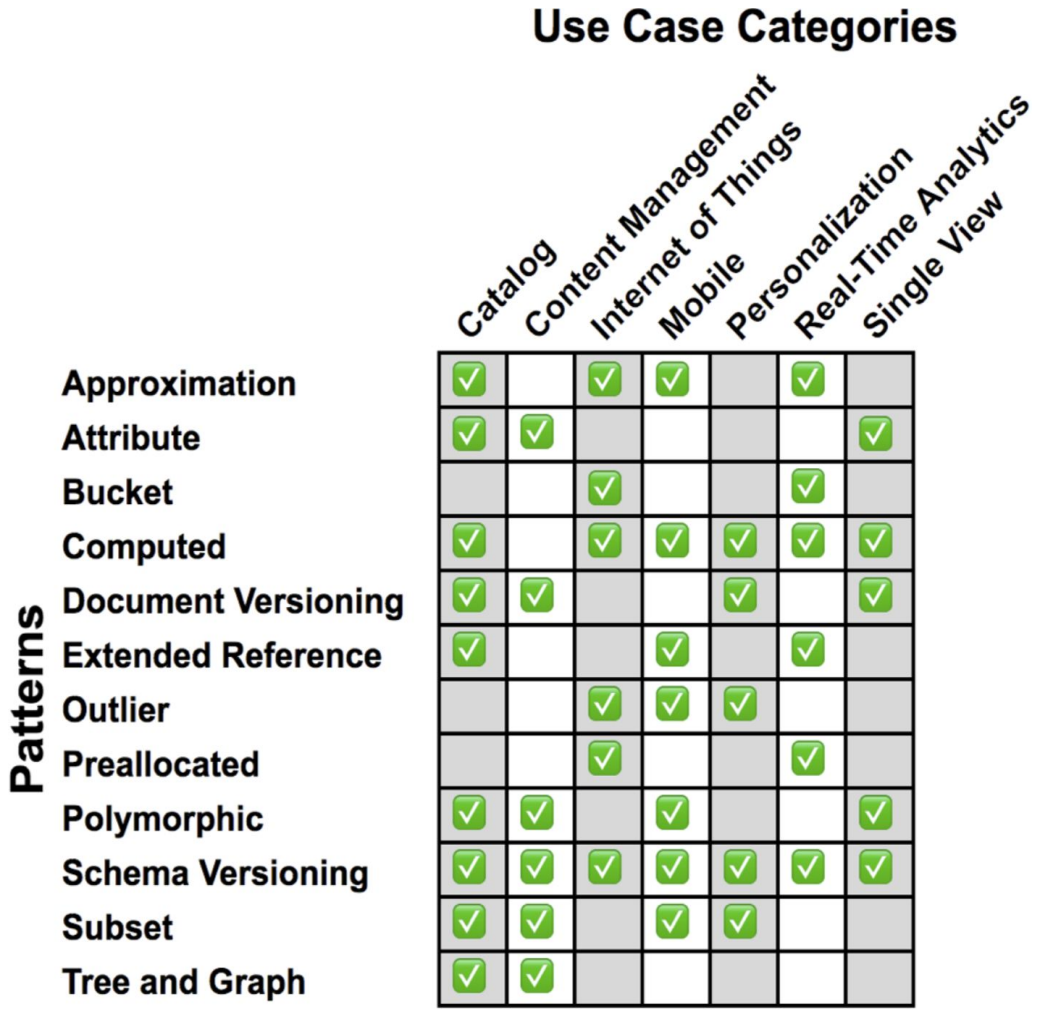
- Šablon polimorfnih dokumenata se koristi kada želimo da imamo objedinjen pristup grupi vrlo sličnih dokumenata.
- Česta upotreba u aplikacijama koje pružaju objedinjen pogled nad podacima iz različitih izvora (*Single View Applications*):



Ostali dizajn šabloni

- Šablon podskupa – ukoliko niz ugnježdenih dokumenata postane prevelik, ugnježdenima možemo održati samo najrelevantnije dokumente (npr. najskorijih n), dok se preostali mogu izdvojiti u zasebnu kolekciju.
- Šablon autlajera – retke pojave (upita kojima šema ne odgovara ili dokumenata koji se značajno razlikuju u odnosu na ostale) ne bi trebalo da izazovu promenu šeme.
- Šablon verzionisanja šeme – polje *schema_version* se dodaje u dokumente kako bi se mogla pratiti promena šeme, i kako bi dokumenti kreirani u odnosu na prethodnu šemu mogli da se očuvaju bez problema u radu aplikacije.

Dizajn šabloni za projektovanje logičke šeme baze



Izvor: https://www.mongodb.com/blog/post/building-with-patterns-a-summary

Česte greške pri dizajniranju sheme

- Nizovi ,,neograničene" veličine – mogu imati nepovoljan uticaj na performanse sistema, a postoji i ograničenje u mogućoj veličini dokumenta.
  - Naročito nezgodno ako se stalno vrši ,,otpakivanje" nizova.
- Veći broj operacija spajanja – operacija spajanja je može biti spora i značajno degradirati performanse.
- Zagušenje dokumenata nepovezanim poljima – izbegavati zajedničko čuvanje polja koja se ne koriste zajedno, naročito u situaciji kada su dokumenti veliki.
- Pravljenje nepotrebnih indeksa – može degradirati performanse; nepotrebni indeksi mogu biti retko korišćeni, redundantni zbog složenih indeksa, a može biti da se uopšte i ne koriste.
- Preveliki broj kolekcija – kreiranje kolekcije podrazumeva i kreiranje pratećih indeksnih struktura.

SADRŽAJ

- Dokument-orijentisane baze podataka i MongoDB
- Projektovanje logičke šeme BP
- Dizajn šabloni za projektovanje logičke šeme baze podataka

PITANJA?