

Fakultet tehničkih nauka, DRA, Novi Sad

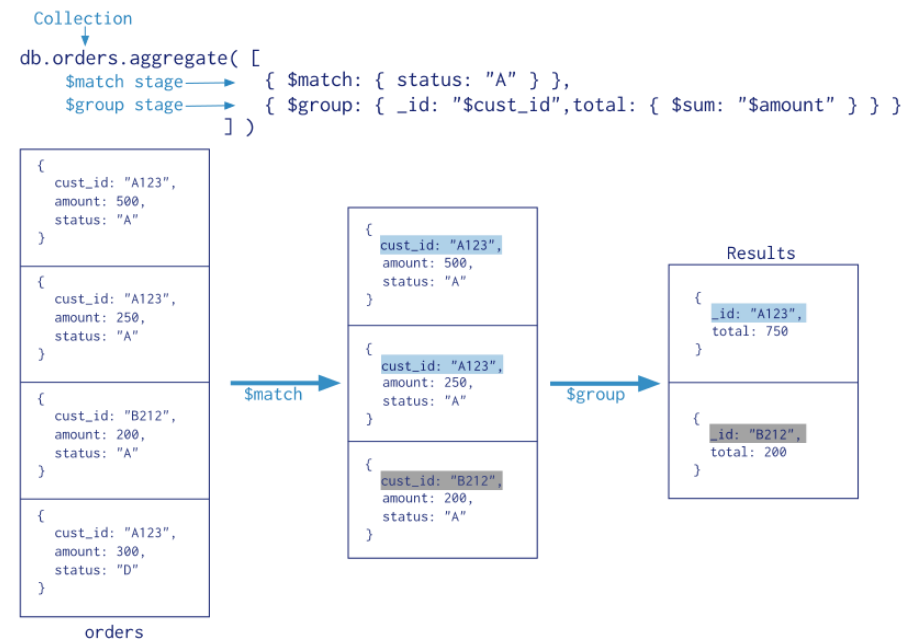
Predmet:
Organizacija podataka

MongoDB

Agregacija podataka

Agregacioni *pipeline*

- U okviru *MongoDB* upitnog jezika postoji zaseban *framework* za agregaciju podataka – agregacioni pipeline (*Aggregation Pipeline*)
- Modeliran je po konceptu *pipeline*-a za obradu podataka, koji podrazumeva da su elementi koji vrše obradu podataka **povezani u seriji**, gde izlaz iz jednog elementa predstavlja ulaz u drugi element
- Dokumenti ulaze u *pipeline* koji se sastoji od više etapa (engl. *stages*), a kao rezultat se dobijaju agregirani rezultati



Agregacioni *pipeline*

- Agregacija podataka vrši se pozivom **aggregate** naredbe, koja ima sledeću strukturu:

```
db.<naziv_kolekcije>.aggregate([<etapa1> [, <etapa2>, ...] ])
```

- **Etape** su elementi pomoću kojih se vrši obrada podataka; u okviru *pipeline*-a mogu, između ostalog, postojati sledeće etape:

\$project	\$sort, \$skip, \$limit,
\$match	\$unwind
\$group	\$out
	...

- Sve etape sem **\$out** mogu se ponavljati.

Etapa \$project

- U okviru ove etape vrši se **odabir potrebnih polja**, odnosno, odbacivanje nepotrebnih polja

```
{ $project: <dokument_za_projekciju> }
```

- Na izlazu iz ove etape naći će se jednak broj dokumenata kao i na ulazu
 - Samo će „oblik“ dokumenata biti različit (engl. *reshape*)
- U okviru ove etape mogu se kreirati i nova polja, o čemu će biti reči kasnije

Etapa \$match

- U okviru ove etape vrši se **selekcija** dokumenata **po specificiranom uslovu**

```
{ $match: <dokument_za_filtriranje_upita> }
```

- Broj dokumenata na izlazu iz ove etape **zavisi od kriterijuma** za selekciju
- Preporuka je da se etapa **iskoristi što je ranije moguće** u toku agregacionog *pipeline* – a
 - Zbog toga što se njenim korišćenjem **smanjuje ukupan broj dokumenata**, čime se minimizuje vreme potrebno za obradu podataka u preostalom delu *pipeline* – a.
 - Ukoliko se iskoristi na samom početku *pipeline* – a (i samo tada), pri upitu se mogu **iskoristiti pogodnosti koje pružaju indeksi** kreirani nad poljima dokumenata

Etapa \$group

- U okviru ove etape vrši se **grupisanje** ulaznih dokumenata na osnovu **specificiranog izraza**

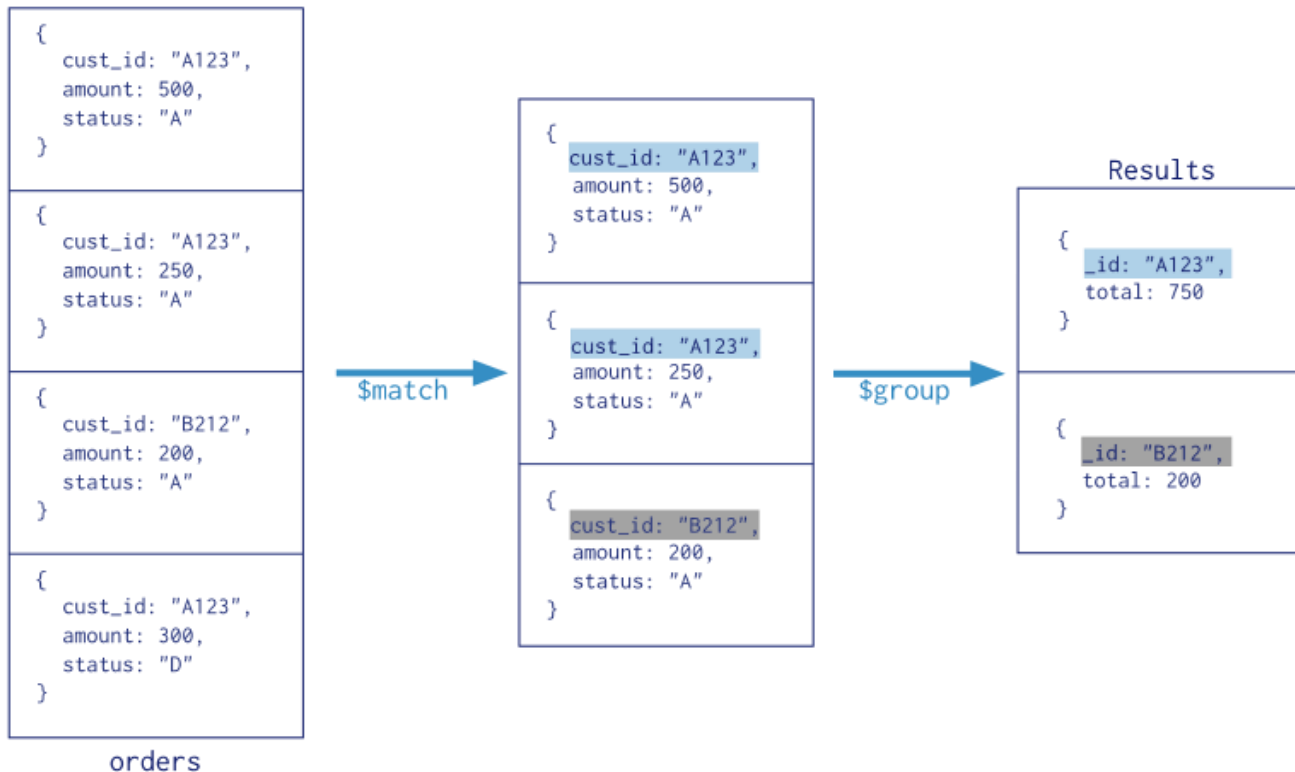
```
{ $group: { _id: <expression>,  
            <field1>: { <accumulator1> : <expression1> }, ... } }
```

- Grupisanje se vrši na osnovu izraza zadatog za vrednost `_id` polja
- Za svaku grupu se vrši izračunavanje agregiranih vrednosti za polja, pri čemu se način izračunavanja vrednosti specificira pomoću **akumulatorskih operatora**
- Na izlazu iz etape se dobija **po jedan dokument za svaku grupu**
 - Izlazni dokumenti sadrže polje `_id`, koje sadrži ključ po kom je vršeno grupisanje, kao i polja za koja su izračunate agregirane vrednosti
- Ukoliko je potrebno referencirati neko od polja iz ulaznog dokumenta, potrebno je za vrednost polja dati putanju (engl. *path*) do polja, u obliku `$nazivPolja`

Etapas \$group

- Primer:

Collection
↓
db.orders.aggregate([
 \$match stage → { \$match: { status: "A" } },
 \$group stage → { \$group: { _id: "\$cust_id", total: { \$sum: "\$amount" } } }
])



Vrednosti za polje `_id` u etapi `$group`

- Postavljanjem vrednosti za polje `_id` u okviru dokumenta za specifikaciju `$group` etape određuje se kriterijum po kom će biti vršeno grupisanje ulaznih dokumenata
- Ukoliko je potrebno **sve dokumente** posmatrati **kao da su u jednoj grupi**, vrednost `_id` polja može se postaviti na *null*
- Ukoliko je potrebno izvršiti **grupisanje po više uslova** (kompozitno grupisanje), za vrednost polja `_id` može se postaviti **ugnježdeni dokument** oblika:

```
_id: { "izlazniNaziv1": "$nazivPolja1"[, "izlazniNaziv2": "$nazivPolja2",...]}
```

- Ovakva sintaksa se može iskoristiti i za jedno polje, radi dodavanja semantike u izlazne dokumente

Agregacioni operatori

- Agregacioni operatori, između ostalog, mogu biti sledeći:
 - **\$avg** – izračunavanje srednje vrednosti polja u grupi,
 - **\$sum** – izračunavanje sume vrednosti polja u grupi,
 - **\$max** – određivanje maksimalne vrednosti polja u grupi,
 - **\$min** – određivanje minimalne vrednosti polja u grupi,
 - **\$addToSet** – vraća niz jedinstvenih vrednosti za polje u grupi,
 - **\$push** – vraća niz postojećih vrednosti za polje u grupi.

Zadaci za vežbu

- Izračunati prosečan broj stanovnika za opštine, grupisano po državama.

```
db.zips.aggregate([  
  { $group: { "_id": "$state", "average": { $avg: "$pop" } }  
}] )
```

- Izračunati ukupan broj stanovnika za Kaliforniju (CA), Njujork (NY), Nju Džerzi (NJ) i Los Anđeles (LA):

```
db.zips.aggregate([ { $match: { "state": { $in : ["CA", "NY",  
"NJ", "LA"] } } },  
                    { $group: { "_id": "$state",  
"value": { $sum: "$pop" } } } ] )
```

Zadaci za vežbu

- Odrediti koji je najveći broj stanovnika koji ima jedna opština (zip kod) za CA i LA.

```
db.zip.aggregate([ { $match: { "state": { $in : ["CA", "LA"]} }  
},  
  { $group: { "_id": { "state": "$state" }, "value": { $max:  
"$pop" }}} ] )
```

- Izračunati koliko ima opština po državama:

```
db.zip.aggregate([  
  { $group: { "_id": { "state": "$state" }, "count": { $sum: 1 }  
} } ] )
```

Zadaci za vežbu

- Prikazati kojim gradovima, grupisano po državama, pripadaju opštine koje imaju više od 80.000 stanovnika:

```
db.zip.aggregate([{$match: {"pop": {$gt: 80000}}},  
                  {$group: { "_id": { "state": "$state" },  
                             "cities": { $addToSet:  
"$city"} } } ])
```

- Odrediti koje je najduže trajanje koje ima neki film ima u okviru movieDetails kolekcije. Pri ispisu dokumenata ne treba ispisivati vrednost _id polja.

```
db.movieDetails.aggregate([  
    { $group: { "_id": null, "maxRuntime": { $max:  
"$runtime"} } },  
    { $project: { "_id": 0 } } ])
```

Zadaci za vežbu

- Odrediti koliko je filmova bilo snimljeno u 2013. i 2014. godini, razvrstano i po godini snimanja i po rangiranju (rated).

```
db.movieDetails.aggregate([
  {$match: {"year": {$in: [2013, 2014]}}},
  {$group: {"_id": {"year": "$year", "rated": "$rated"}, "count":
  { $sum: 1}}}]])
```

- Prikazati, razvrstano po godini snimanja filma, koje su glumačke postave glumile u filmovima sa imdb ocenom većom od 8.5, a da je pritom za tu ocenu glasalo više od 100.000 gledalaca.

```
db.movieDetails.aggregate([
  {$match: {"imdb.rating": {$gte: 8.5}, "imdb.votes": {$gt:
  100000}}},
  {$group: {"_id": "$year", "actors": {$addToSet:
  "$actors"}}}]])
```

Zadaci za vežbu

- Prikazati koji je maksimum korisničkih pregleda (userReviews) na Rotten Tomatoes sajtu za filmove snimljene u toku iste godine.

```
db.movieDetails.aggregate([
  { $match: { "tomato": { $exists: true } } },
  { $group: { "_id": "$year",
    "tomatoUserReviews": { $sum: "$tomato.userReviews"
  } } },
  { $group: { "_id": null, "max": { $max:
    "$tomatoUserReviews" } } } ] )
```

Etapa \$unwind

- Ukoliko je potrebno izvršiti grupisanje na osnovu vrednosti za elemente nekog niza, potrebno je taj niz „razmotati“ – za svaku vrednost iz niza kreira se novi dokument

```
{ $unwind: <putanja_do_niza> }
```

- U ovoj etapi se za svaki dokument na ulazu dobije n dokumenata na izlazu, što dovodi do eksplozije broja dokumenata
- Primer:

```
db.movieDetails.aggregate([{$unwind: "$countries"},  
    {$group: { "_id": {"country": "$countries"}}}])
```


Zadaci za vežbu

- Za svakog glumca ili glumicu prikazati u koliko je filmova glumio/la, pri čemu treba ispisati samo one koji su glumili u makar 5 filmova.

```
db.movieDetails.aggregate([
    {$unwind: "$actors"},
    {$group: { "_id": {"actor": "$actors"}, "count": {$sum:
1}}}},
    {$match: {"count": {$gte: 5}}},
    {$project: {"actor": "$_id.actor", "_id": 0,
"count": 1}}])
```

Etapa \$sort

- U okviru ove etape vrši se **sortiranje** ulaznih dokumenata na osnovu **specificiranog izraza**

```
{ $sort: { <field1>: <sort_order>[ , <field2>: <sort_order> , ... ] } }
```

- **<sort_order>** može se postaviti na:
 - 1, ukoliko želimo da sortiramo po uzlaznoj vrednosti polja (engl. *ascending*)
 - -1, ukoliko želimo da sortiramo po silaznoj vrednosti polja (engl. *descending*)

- Primer:

```
db.zips.aggregate([{$sort: {"state": 1, "city": 1, "pop": -1}}])
```

Etapa \$skip

- U okviru ove etape se, na osnovu specificirane vrednosti, zanemaruje prvih n ulaznih dokumenata, dok se preostali dokumenti vraćaju u pipeline.

```
{ $skip: pozitivan_ceo_broj }
```

- Ova etapa ima smisla **samo** kada se koristi uz etape \$sort i \$limit, za realizaciju paginacije
- Primer:

```
db.zips.aggregate([{$sort: {"pop": -1}}, {$skip: 5}, {$limit: 5}])
```

Ostale etape i operatori

- Postoji veliki broj etapa i operatora koji se mogu izvršiti u okviru *pipeline*-a
- Kompletan spisak etapa može se naći na:
 - <https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/>
- Kompletan spisak operatora može se naći na:
 - <https://docs.mongodb.com/manual/reference/operator/aggregation/>