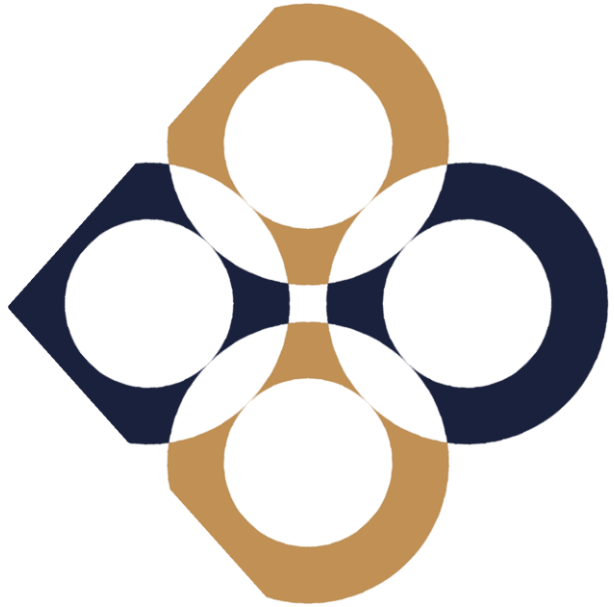
BUDAPESTI CORVINUS EGYETEM

Adatbázisok gyakorlat 08

MongoDB lekérdezések



Adatbázisok kezelése

show databases	Listázza a létező adatbázisokat
use adatbázisnév	Az aktuális adatbázis megadása – ha nem létezik, akkor a parancs létre is hozza*
db.getName()	Az aktuális adatbázis nevének lekérdezése
db.dropDatabase()	Az aktuális adatbázis törlése

* Az adatbázisok listájában csak akkor jelenik meg, ha már van benne legalább egy dokumentum

Gyűjtemények kezelése

show collections	Listázza a létező gyűjteményeket
db.createCollection(név, opciók)	Létrehoz egy új gyűjteményt*
db.gyűjteménynév.drop()	Törli az adott gyűjteményt

* Új gyűjtemény egy dokumentum beszúrásával is létrehozható

Gyűjtemények lekérdezése

db.gyűjteménynév.find*(szűrés, projekció)

- db.trips.find() -- A trips gyűjtemény összes dokumentumát listázza
- db.trips.find().pretty()** -- A dokumentumokat barátságosabb formában jelenítimeg
- db.trips.find({"start station name" : "Howard St & Centre St"}) -- Szűrés a start állomásra
- db.trips.find({"start station name" : "Howard St & Centre St", "birth year": 1967}) – Szűrés a start állomásra és a születési évre
- db.trips.find({"start station name" : "Howard St & Centre St", "birth year": 1967}, {"start station name": 1, "end station name": 1}) -- az előző szűrésnél csak a start- és a cél állomásokat jeleníti meg

* A db.gyűjteménynév.findOne() hasonlóan működik, de csak a legelső találatot adja vissza

** A.forEach(printjson) is használható

BUDAPESTI CORVINUS EGYETEM

Fontosabb adattípusok

Típus	Példa
String	db.collection.find*({"Név": **"Béla"**})
Integer	db.collection.find({"Ár": **20000**})
Double	db.collection.find({"Pontszám": **15.23**})
Boolean	db.collection.find({"Házas_e": **true**})
NULL	db.collection.find({"Mobile number": **null**})
Arrays	db.collection.find({"Végzettség": **["matematika", "fizika"]**})
Object	db.collection.find({"Könyv": **{"cím": "C++ progrmozás", "szerző": "Andrei Alexandrescu"}**})
Object ID	db.collection.find({"_id": **ObjectId("5a934e000102030405000000")**})
Date	db.collection.find({ "Születésnap": **new Date("1996-05-07")**})
Timestamp	db.collection.find({"Felvitel dátuma": **ISODate("2020-03-02T01:11:18.965Z")**})

*A db.collection.find() lekérdező utasítást ld. gyakorlaton

BUDAPESTI CORVINUS EGYETEM

Fontosabb operátorok I.

Operátor	Szerepe	Példa
$gt	Kisebb (összehasonlításnál)	db.trips.find({"tripduration": {$gt: 50000}})
$lt	Nagyobb (összehasonlításnál)	db.trips.find({"tripduration": {$lt: 50000}})
$gte	Nagyobb, vagy egyenlő	db.trips.find({"tripduration": {$gte: 50000}})
$lte	Kisebb, vagy egyenlő	db.trips.find({"tripduration": {$lte: 50000}})
$all, $in	Egy tömb minden elemével, illetve legalább egy elemével való egyezést vizsgál	db.trips.find({"birth year": {$in: [1987, 1988]}})
$and, $or, $not	Logikai műveletek	db.trips.find({$and: [{"usertype":"Subscriber"}, {"birth year":1969}]})
$exists	Egy mező létezését vizsgálja	db.trips.find({"birth year": {$exists: true}})
$regex	Egy reguláris kifejezéssel való egyezést vizsgál	db.trips.find({"start station name": {$regex: /How/i }}, {"start station name":1, "start station location":1})

BUDAPESTI CORVINUS EGYETEM

Fontosabb operátorok II.

Operátor	Szerepe
$abs	Abszolút érték
$add	Összeadás
$subtract	Kivonás
$multiply	Szorzás
$divide	Osztás
$pow	Hatványozás
$switch	Többirányú elágazás
$cond	Kétirányú elágazás

BUDAPESTI CORVINUS EGYETEM

Beágyazott mezők elérése

Az összetett mezők tartalmát a . (pont) operátorral érhetjük el

Példák:

- db.trips.find({}, {"start station location.type":1})
- db.trips.find({}, {"start station location.coordinates":1})

Regex – reguláris kifejezések*

{ <field>: { $regex: /pattern/<options> } } vagy
{ <field>: { $regex: /pattern/, $options: '<options>' } }

- Minták
  - ^ - Adott karaktersorozattal kezdődik
  - $ - Adott karaktersorozattal végződik
- Fontosabb opciók
  - i -- case insensitivity
  - m – többsoros karaktersorozatokat soronként vizsgál
  - x – nem veszi figyelembe a puha szóközöket és kommenteket (#)
- Példa: db.trips.find({"start station name": {$regex: /^he/i}}, {"start station name":1})

Az SQL-beli tartalmazás (mezőnév LIKE '%minta%') megfelelője:

"mezőnév": {$regex: /minta/} vagy
"mezőnév": /minta/

* Egymás után több reguláris kifejezés is felsorohatóvesszővel elválasztva

Rendezés

db.gyűjteménynév.find(szűrés, projekció).sort(rendezés definíció)

A rendezés definíció tartalmazhatja a rendezés szempontjait (mezők) és azok irányait (1: növekvő, -1: csökkenő)

- db.trips.find({"birth year": 1967}).sort({"start station name": 1})
  -- A trips gyűjtemény azon dokumentumait, ahol a születési év 1967, a start állomás neve szerint növekvő sorrendbe rendezi
- db.trips.find().sort({"tripduration": -1}) – A dokumentumokat a utazás időtartama szerint csökkenő sorrendbe rendezi

BUDAPESTI CORVINUS EGYETEM

A lekérdezés eredményének korlátozása

db.gyűjteménynév.find(szűrés, projekció).limit(szám)

Alekérdezéseredményéből csakazelső adott számúdokumentumot jeleníti meg.

db.trips.find().pretty().limit(2)

-- Atrips gyűjtemény első két dokumentumát jeleníti meg felhasználóbarátformátumban

db.gyűjteménynév.find(szűrés, projekció).skip(szám)

Alekérdezés eredményéből kihagyja azelső adott számúdokumentumot

db.trips.find().skip(5)

-- Atrips gyűjtemény első 5 dokumentumát kihagyja amegjelenítésből

BUDAPESTI CORVINUS EGYETEM

Aggregálás

db.gyűjteménynév.aggregate(pipeline)

Adott szempontok szerint csoportokat képez, és azokon aggregálást (pl. összegzés) hajt végre

Példa:

```
db.trips.aggregate( [
    { "$match": { "start station id": 268 } },
    { "$group": { _id:"$usertype",
                  total: {$sum: "$tripduration"} }},
    { "$sort": {total: -1} }
] )
```

Pipeline:

- Aggregációs műveletek és szakaszoktömbje.
- Minden szakasztranszformálja a dokumentumot

BUDAPESTI CORVINUS EGYETEM

Aggregációs műveletek és szakaszok

Művelet	Leírás
$avg	Átlagot
$min	Minimum
$max	Maximum
$sum	Összeg
$first	A legelső dokumentum a csoportban
$last	Az utolsó dokumentum a csoportban

Szakasz	Leírás
$group	Csoportokat képez
$limit	Korlátozza a dokumentumok számát
$skip	Kihagy n dokumentumot
$match	Egyezőséget vizsgál
$merge	Az aggregáció eredményét egy gyűjteményhez hozzáadja
$sort	Rendez
$project	Kiválaszt mezőket
$unwind	Tömböt elemeire bont
$out	Az eredményt új gyűjteménybe teszi
$lookup	Gyűjtemények összekötése

Aggregálás – a GROUP BY megfelelője

```
{"$group": { _id:"$csoportmező",
             oszlopnév: {aggregációs művelet: "$aggregálandó mező"} } }
```

- Haaz_id: "$csoportmező" utáni részelmarad, akkor azmegfelel a SELECTDISTINCT $csoportmező …utasításnak
  - Hatöbb mező alapján szeretnénk csoportokat képezni, akkor amegfelelő rész: _id: {"oszlop1név": "$csoportmező1", "oszlop2név : "$csoportmező2" …}alakú
- A HAVING megfelelője a $group utáni Smatch szakasz,pl:

```
db.trips.aggregate( [
  { "$group": { _id: "$usertype", total: {$sum: "$tripduration"} }},
  { "$match": {total: {$lt: 100}}}
] )
```

Példa1 – számított mező operátorokkal

Adjunk hozzá minden egyes út időtartamához 5 időegységet!

```
db.trips.aggregate
( [
    { "$project":
        { _id: 0,
          trip2: {$add: ["$tripduration", 5]}
        }
     }

 ] )
```

Példa2 – számított mező operátorokkal

Az 500 időegység feletti utak legyenek hosszúak, a többiek rövidek!

```
db.trips.aggregate([
   {$project:
      {_id: 0,
         result: {$cond:
        {if: {$gt: ["$cook_time", 500]},
          then: "hosszu", else: "rovid"}
        }
      }
   }
])
```

Példa3 – számított mező operátorokkal

```
db.trips.aggregate([{
   $project:
     {_id: 0,
      result:
      {$switch: {
          branches:
           [
             {case: {$gt: ["$tripduration", 500]}, then: "hosszú"},
             {case: {$lte: ["$tripduration", 300]}, then: "rövid"}
           ],
          default: "közepes"}
       }
     }
}])
```

Példa4 – LEFT JOIN megvalósítása $lookup operátorral*

```
db.orders.aggregate([
  {
    $lookup: {
      from: "users",              // csatlakoztatott gyűjtemény
      localField: "user_id",      // orders.user_id
      foreignField: "user_id",    // users.user_id
      as: "user_info"             // új mező a kapcsolt adatokkal
    }
  }
])
```

* MongoDB 3,2 verziótól, csak left join van

Új dokumentum létrehozása

db.gyűjteménynév.insertOne(dokumentum)

Új dokumentumot szúrbe azadott gyűjteménybe

Példa:

```
db.trips.insertOne(
 {
    "tripduration": 300,
   "start station id": 50000 ,
   "start station name":"XYZ",
   "bikeid": 568987,
   "usertype": "Customer"
 }
 )
```

Egyszerretöbb dokumentumot is létrehozhatunk az **insertMany([dokumentumok])** utasítás segítségével. Ilyenkor a dokumentumokat vesszővelelválasztva ke**l** megadni.

Dokumentum módosítása

db.gyűjteménynév.updateOne(szűrés, módosítás)

Módosítja a szűrésnek megfelelő dokumentum tartalmát

Példa:

db.trips.updateOne(

{"_id":ObjectId("572bb8222b288919b68abf6d")},

{$set*: {"bikeid":1000}}

Egyszerretöbb dokumentumotis módosíthatunk az **updateMany(szűrés, módosítás, opciók)** utasítássegítségével.

*A $set segítségével a meglévő mező módosítása mellett új mező is létrehozható, a $unset töröl egy meglévő kulcs-érték párt, a$inc pedig egy mező értékét növeli meg egy adottértékkel

Dokumentum törlése

db.gyűjteménynév.deleteOne(szűrés, módosítás, opciók)

Törli a feltételnek megfelelő dokumentumot

Példa:

```
db.trips.deleteOne(
{"_id":ObjectId("572bb8222b288919b68abf6d")}
)
```

Egyszerre több dokumentumot is módosíthatunk az **deleteMany(szűrés, módosítás, opciók)** utasítás segítségével. Ilyenkor a dokumentumokat vesszővel elválasztva kell megadni.

Tömb módosítása

A$push segítségével a tömhöz új elem adható, a$pull segítségével pedig meglévő elem eltávolítható

Példa:

```
db.trips.updateOne(
  {"_id":ObjectId("572bb8222b288919b68abf6d")}
  ,
  {$push: {"end station location.coordinates": 2}}
)
```

Tömb elemek elérése

A $slice segítségével a tömb elemeinek egy részintervalluma is elérhető

Példa:

```
db.trips.find(
   {"bikeid":1000},
   {"end station location.coordinates": {$slice: [0, 2]}}
)
```

Lekérdezések végrehajtási statisztikája

```
db.gyűjtemény.find(szűrés, projekció).explain(”executionStats”)
```

Példa:

```
db.trips.find(
 {"bikeid": {$lt: 10000}}).explain("executionStats")
```

-- megmutatja a végrehajtási tervet és a statisztikákat

BUDAPESTI CORVINUS EGYETEM

Indexek

```
db.gyűjtemény.getIndexes()
-- lekérdezi a meglévő indexeket

-- alapértelmezés szerint minden gyűjtemény indexelve van _id alapján
```

```
db.gyűjtemény.createIndex({mező: 1 | -1})
-- új indexet hoz létre (1 – növekvő, -1 csökkenő)
```

Pl: db.trips.createIndex({"bikeid": 1})

```
db.gyűjtemény.dropIndex(indexnév)
 -- törli a meglévő indexet
```

BUDAPESTI CORVINUS EGYETEM

Elérés Python-ból

- Először a pymongo csomagot kell installálni
- Utána importálni a MongoClient modult
- Végül csatlakozni az adatbázishoz (connectionstring a Mongo Atlas-ban)

```
In [ ]: !pip install pymongo
        from pymongo import MongoClient
        !pip install dnspython
        import pymongo
```

```
In [ ]: client = pymongo.MongoClient("connectionstring")
```

```
In [ ]: db = client.sample_training
```

```
In [ ]: ered = db.trips.find({"tripduration": 200})
```

```
In [ ]: for i in ered:
            print(i)
```