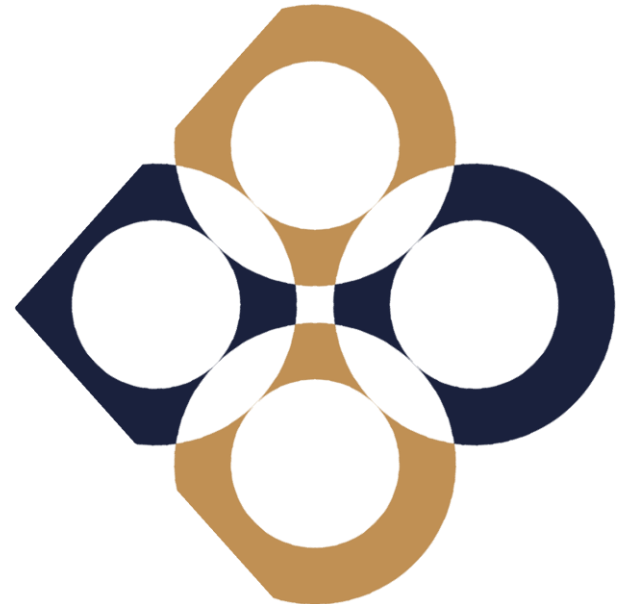


Adatbázisok előadás 08

Dokumentum adatbázisok



Miről lesz szó?

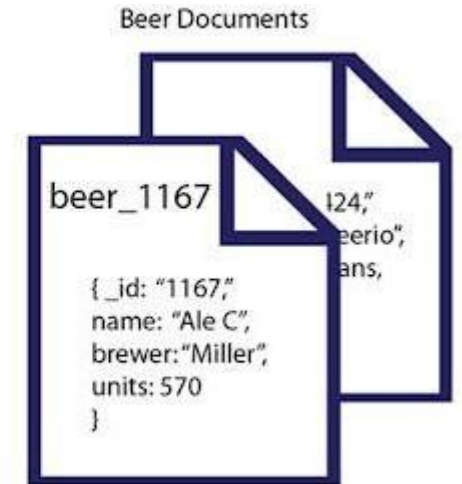
- ☐ Dokumentum adatbázisok jellemzői
- ☐ MongoDB adatbázis
 - ☐ Dokumentumok és gyűjtemények
 - ☐ Adattípusok
 - ☐ Lekérdezések
 - ☐ Indexek
 - ☐ Elérés Python-ból
- ☐ Feladatok megoldása

Dokumentum adatbázisok

- ☐ Az adatokat strukturált dokumentumok formájában (XML, JSON, PDF, DOC) tárolják
- ☐ A dokumentumoknak nem kell azonos szerkezetűeknek lenniük
- ☐ A dokumentumok egymásba ágyazhatók
- ☐ A dokumentumokat gyűjteményeknek nevezett csoportokba szervezik
- ☐ Támogatják a beágyazott kulcs-érték párokat
- ☐ A dokumentumok bármely attribútuma alapján lekérdezhetők

Beers Table

1167	Ale C	Miller	570
3424	Beerio	Ians	340
5612	Amstel	Amtel	121
2409	Colt's	BeerCo	98



<https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/compare-docs-vs-relational.html>

Dokumentum adatbázisok - Előnyök és hátrányok

ELŐNYÖK

- Flexibilitás és skálázhatóság
- Gyors írási műveletek
- Az adatok struktúráját nem kell előre definiálni
- A struktúra rugalmasan változtatható
- Ingyenes
- Jól dokumentált

HÁTRÁNYOK

- Tranzakciókezelés
- Memóriaigényes
- Több gyűjteményt érintő lekérdezések problémásak
- Limitált dokumentumméret
- Adatminőség, duplikációs problémák
- Kényszerek kezelése

Dokumentum adatbázisok – Mikor használjuk őket?

Ha többféle
szempont szerint
szeretnénk
lekérdezni

Ha sok valós idejű
adatot kell kezelni

Ha fontos a
flexibilis séma, és
a gyors fejlesztés

Ha van elég
memóriánk

Ha kezelni tudjuk
az adatminőségi
problémákat

Ha nem jelent
problémát a
korlátozott
tranzakciókezelés

Dokumentum adatbázisok - Tipikus használati esetek

- Felhasználói profilok
- Valós idejű, big data jellegű adatok
- Dinamikus webhelyek sok valós idejű adatmódosítással
- Tartalomkezelők (CMS)
- Strukturálatlan vagy félig strukturált adatok

Dokumentum adatbázisok - Példák



MongoDB – vezető felhasználók

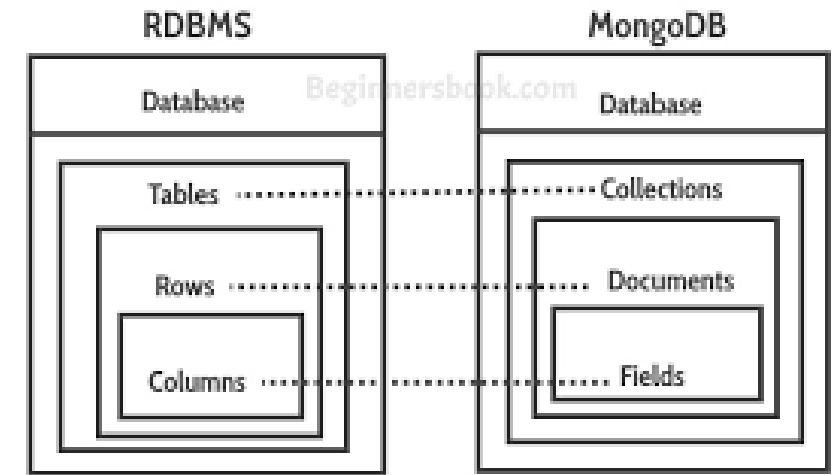
Leading Organizations Use MongoDB



<https://slideplayer.com/slide/10988388/>

Dokumentum adatbázisok - MongoDB

- ☐ A legnépszerűbb NoSQL adatbázis
- ☐ JSON-formátum
- ☐ JavaScript-alapú lekérdezések
- ☐ Nagymértékben skálázható
- ☐ A lekérdezési sebesség indexekkel gyorsítható
- ☐ Terheléselosztás (shard-ek)
- ☐ Flexibilis séma



Relációs adatbázis és MongoDB fogalmi megfeleltetések

<https://beginnersbook.com/2017/09/mapping-relational-databases-to-mongodb/>

Data Models: Relational to Document

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB Document

```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, ... },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, ... }
  ]
}
```

String

Array

Object

Dokumentumok (Documents)

A dokumentum adatbázisok alap tárolási egységei

A dokumentumok jellemzői:

- ☐ A relációs adatmodell sorainak felelnek meg
- ☐ JSON-jellegű forma, binárisan tárolva (BSON)
- ☐ Kulcs-érték párokból épülnek fel
- ☐ A kulcsok case-sensitivek, és egyedieknek kell lenniük
- ☐ A dokumentumok egymásba ágyazhatók

Példa:

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports"]  
}
```

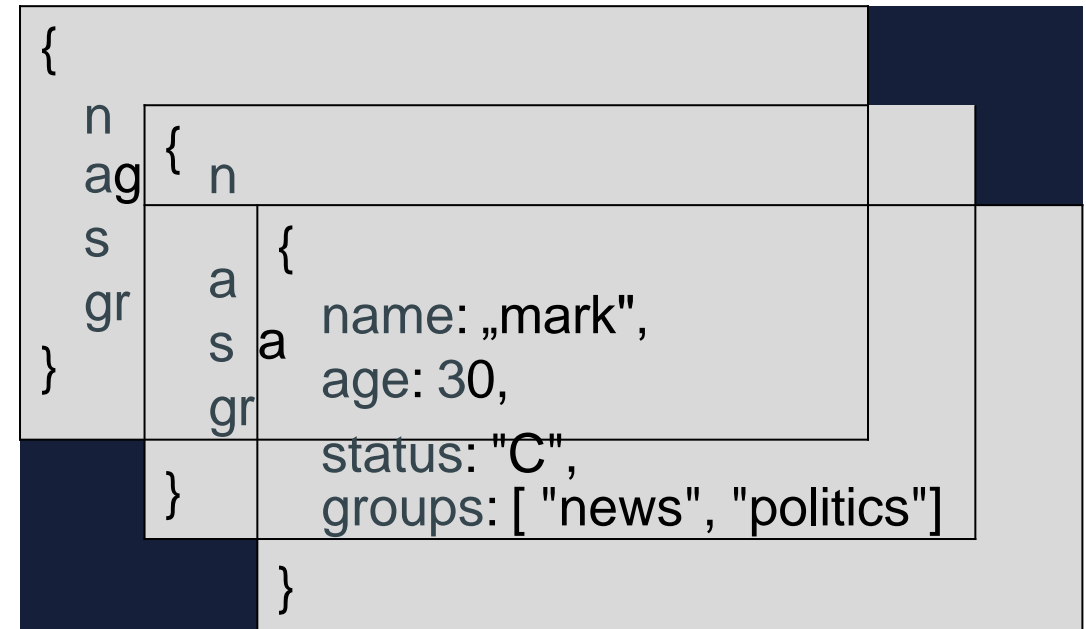
Gyűjtemények (Collections)

Összetartozó dokumentumok csoportjai

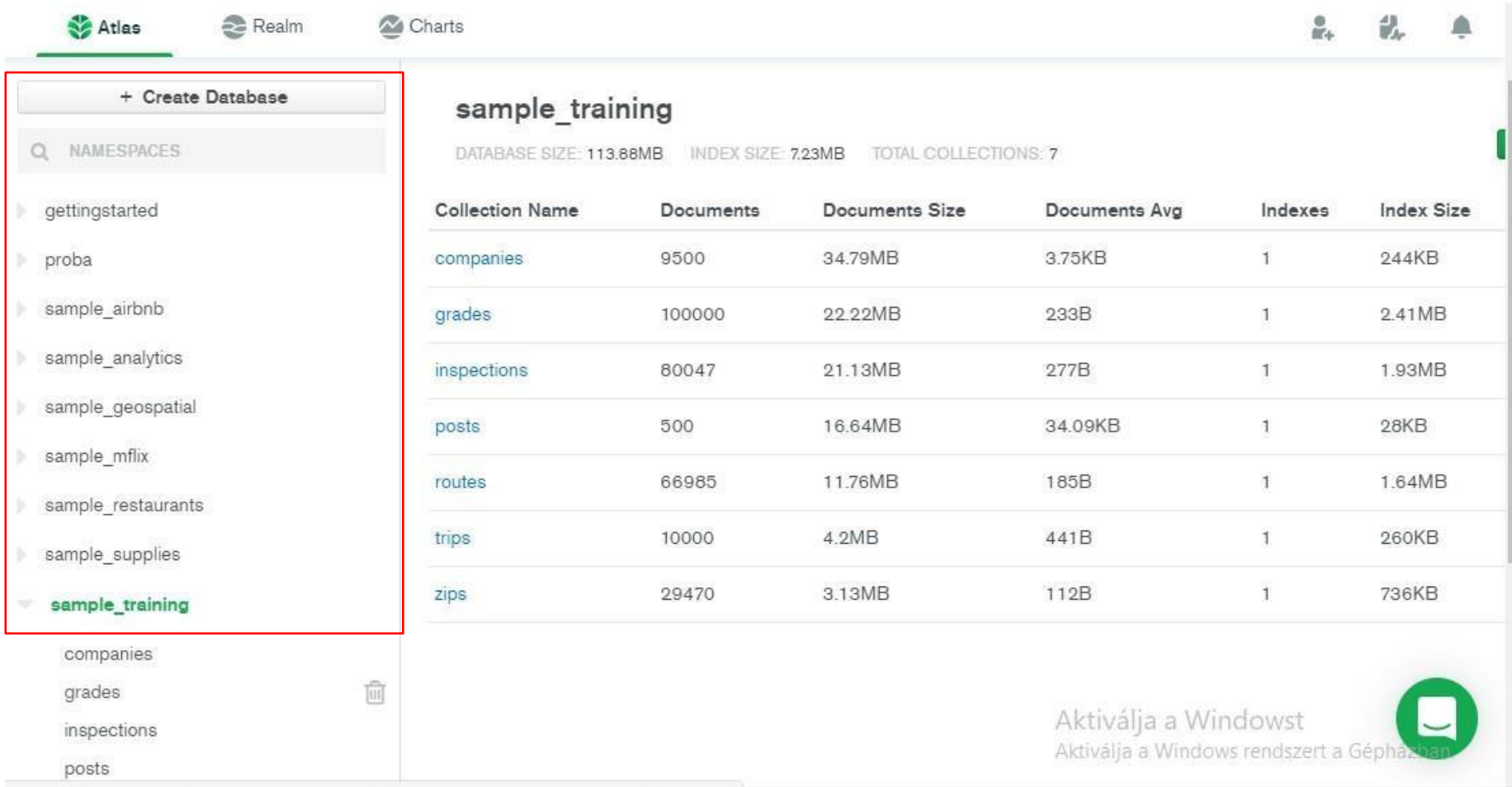
A gyűjtemények jellemzői:

- ☐ A relációs adatmodell tábláinak felelnek meg
- ☐ Hasonló célú dokumentumokat tartalmaznak
- ☐ A dokumentumok felépítése különböző is lehet

Példa:



Összetartozó gyűjtemények csoportjai



The screenshot displays the MongoDB Atlas web interface. On the left sidebar, the 'sample_training' database is highlighted with a red box. The main panel shows the 'sample_training' database details, including its size (113.88MB), index size (7.23MB), and total collections (7). A table lists the collections with their respective document counts, sizes, and index information.

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size
companies	9500	34.79MB	3.75KB	1	244KB
grades	100000	22.22MB	233B	1	2.41MB
inspections	80047	21.13MB	277B	1	1.93MB
posts	500	16.64MB	34.09KB	1	28KB
routes	66985	11.76MB	185B	1	1.64MB
trips	10000	4.2MB	441B	1	260KB
zips	29470	3.13MB	112B	1	736KB

At the bottom right, there is a Windows notification: 'Aktiválja a Windowst' (Activate Windows) with the text 'Aktiválja a Windows rendszert a Gépházban' (Activate the Windows system in the Machine House).

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

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. később

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})

Fontosabb operátorok II.

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

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"}  
        }  
      }  
    ]  
})
```

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 felsorolható vesszővel elválasztva

- ☐ Find
- ☐ Sort, Limit, Skip
- ☐ Operátorok
- ☐ Beágyazott mezők elérése
- ☐ Dokumentumok módosítása, törlése
- ☐ Tömbök módosítása

Gyűjtemények lekérdezése

`db.gyűjteménynév.find*(szűrés, projekció)`

- ☐ `db.trips.find()` -- A trips gyűjtemény összes dokumentumát listázza
- ☐ `db.trips.find().pretty()*` -- A dokumentumokat barátságosabb formában jeleníti meg
- ☐ `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


```
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 az utazás időtartama szerint csökkenő sorrendbe rendezi

A lekérdezés eredményének korlátozása

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

A lekérdezés eredményéből csak az első adott számú dokumentumot jeleníti meg.

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

-- A trips gyűjtemény első két dokumentumát jeleníti meg felhasználóbarát formátumban

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

A lekérdezés eredményéből kihagyja az első adott számú dokumentumot

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

-- A trips gyűjtemény első 5 dokumentumát kihagyja a megjelenítésből

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 szakaszok tömbje.
- Minden szakasz transzformálja a dokumentumot

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

Aggregálás – a GROUP BY megfelelője

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

- Ha az `_id: "$csoportmező"` utáni rész elmarad, akkor az megfelel a `SELECT DISTINCT $csoportmező ...` utasításnak
- Ha több mező alapján szeretnénk csoportokat képezni, akkor a megfelelő rész: `_id: {"oszlop1név": "$csoportmező1", "oszlop2név": "$csoportmező2" ...}` alakú
- A HAVING megfelelője a `$group` utáni `$match` szakasz, pl:

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

Aggregálás II.

```
db.gyűjteménynév.find(szűrés, projekció).count()
```

Megszámolja a lekérdezés eredményeképpen kapott dokumentumok számát

Példa: `db.trips.find({"usertype": "Customer"}).count()`
– megsámolja, hogy hány Customer típusú felhasználó van

```
db.gyűjteménynév.distinct(mezőnév)
```

Listázza az adott gyűjteményben lévő mező különböző értékeit

Példa: `db.trips.distinct("start station name")`
-- listázza az induló állomásokat (mindegyiket csak egyszer)

Aggregálás III.

```
db.gyűjteménynév.mapReduce(mapping fv, reduce fv, {out: 'Result'})
```

- Eredetileg nagyméretű adathalmaz aggregálására hozták létre.
- Teljesítményben elmarad az Aggregálás I. részben ismertetett módszertől (deprecated)
- A MongoDB Atlas free nem támogatja, saját MongoDB szerveren használható
- map függvény: csoportokat képez
- reduce függvény: aggregál

Példa:

```
var mapfunction = function(){emit(this.usertype, this.tripduration)}  
var reducefunction = function(key, values){return Array.sum(values)}  
db.trips.mapReduce(mapfunction, reducefunction, {'out':'Result'})  
db.Result.find()
```

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})`

Új dokumentum létrehozása

```
db.gyűjteménynév.insertOne(dokumentum)
```

Új dokumentumot szúr be az adott gyűjteménybe

Példa:

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

Egyszerre több dokumentumot is létrehozhatunk az **insertMany([dokumentumok])** utasítás segítségével. Ilyenkor a dokumentumokat vesszővel elválasztva kell megadni.

Dokumentum módosítása

```
db.gyűjteménynév.updateOne(szűrés, módosítás)
```

Módosítja a szűrésnek megfelelő dokumentum tartalmát

Példa:

```
db.trips.updateOne(  
  {"_id": ObjectId("572bb8222b288919b68abf6d")},  
  {$set*: {"bikeid": 1000}})
```

Egyszerre több dokumentumot is módosíthatunk az **updateMany(szűrés, módosítás, opciók)** utasítás segítségével.

*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 adott értékkel

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 tömhhöz 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

```
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
```

- ☐ 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)
```


Feladatok megoldása I.

Indítsa el a MongoDB Compass alkalmazást, majd csatlakozzon a MongoDB cluster-hez!

- a. Hozzon létre új adatbázist Gyak_compass néven, azon belül egy új gyűjteményt receptek néven!
- b. A receptek gyűjteménybe importálja be a mellékletben szereplő recipes.json fájl tartalmát (Add data / Import File, majd Select File, végül Import)!

Feladatok megoldása II.

A MongoDB Compass segítségével kérdezze le a receptek gyűjtemény azon dokumentumait, amelyre teljesül:

- a. A lájkok száma több, mint 2!
- b. A lista legyen sorbarendezeve a főzési idő szerint csökkenő sorrendben! (A rendezés funkció az Options gomb lenyomása után érhető el)
- c. A listában ne jelenjenek meg az ingredients és a rating mezők (Project szakasznál kell beállítani)!

Feladatok megoldása III.

Az előző feladatban létrehozott lekérdezésre hajtsa végre az Explain Plan funkciót!

Feladatok megoldása IV.

A MongoDB Compass-ban készítsen új indexet a receptek gyűjteményhez az Indexes rész Create Index funkciójának segítségével!

- a. Az index neve legyen i_title, és a title mező szerint csökkenő legyen
- b. Az index egyedi (unique) legyen (Options rész)!

Feladatok megoldása V.

A MongoDB Atlas-ban navigáljunk a Cluster-hez, majd válasszuk a Connect lehetőséget, ezen belül pedig a "Connect with the mongod shell" opciót!

a. Töltsük le a mongo shell állományt, majd tömörítsük ki egy mappába (pl. Dokumentumok)

b. A fájlkezelőben lépünk be a mongo shell bin mappájába, majd nyissunk egy parancssort!

c. Csatlakozunk a Cluster-hez a Connect to Cluster ablakban megjelenő connection string segítségével!

d. Adjuk ki a show dbs parancsot

Feladatok megoldása VI.

A mongo shellben kérdezzük le, hogy a receptek gyűjteményben kérdezzük le, hogy mely dokumentumoknál szerepel a recept nevében (title) a Tacos szó!

a. A megjelenés kellően szép (json-szerű) legyen!

Feladatok megoldása VII.

A mongo shell-ben kérdezzük le, hogy
recept típusonként (type) mennyi az elkészítési idők
(cook_time) összege!

Feladatok megoldása VIII.

A mongo shell-ben kérdezzük le, hogy a receptek gyűjteményben hány olyan dokumentum van, ahol:

- a. A recept 4 főre szól (servings) ÉS
- b. A tag-ek között szerepel a "quick" vagy az "easy" (legalább az egyik)

Feladatok megoldása IX.

A mongo shell-ben a receptek gyűjteményben a ObjectId("5e878f5220a4f574c0aa56db") azonosítójú dokumentum esetén módosítsuk az elkészítési időt (cook_time) 33 percre!

Feladatok megoldása X.

A mongo shell-ben adjunk hozzá
a ObjectId("5e5e9c470d33e9e8e3891b35") azonosítójú
dokumentum likes tömbjéhez még egy értéket,
mégpedig a 200-at!



**Köszönöm
a figyelmet!**