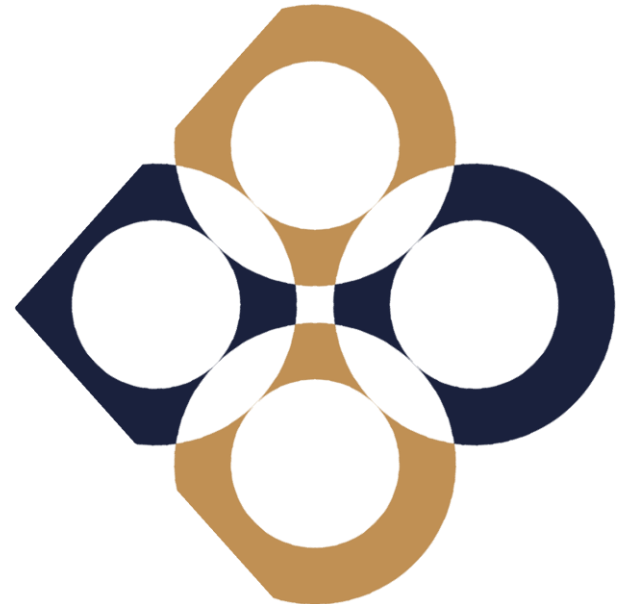


Adatbázisok előadás 08

Dokumentum adatbázisok



Miről lesz szó?

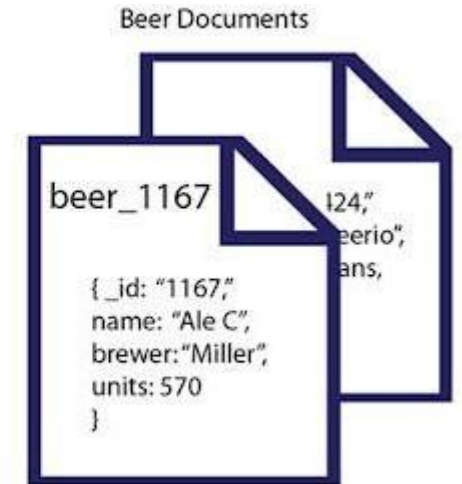
- ☐ Dokumentum adatbázisok jellemzői
- ☐ MongoDB adatbázis
 - ☐ Dokumentumok és gyűjtemények
 - ☐ Adattípusok
 - ☐ Lekérdezések
 - ☐ Indexek
 - ☐ Elérés Python-ból
- ☐ Feladatok megoldása

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

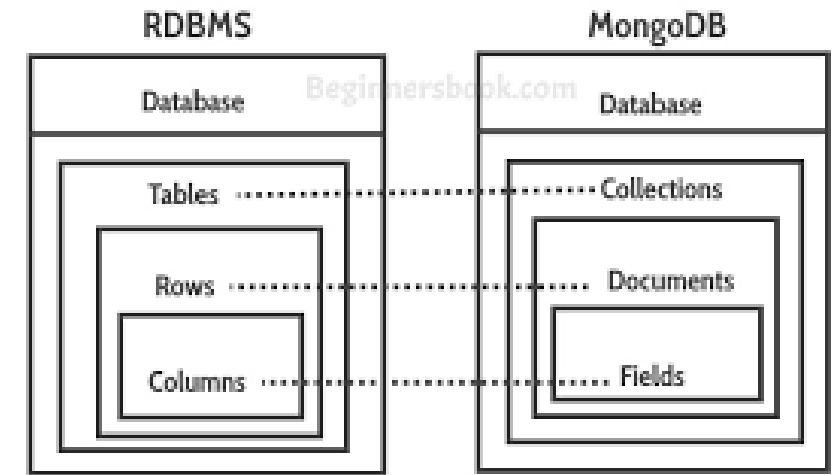
Leading Organizations Use MongoDB



<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"]  
}
```

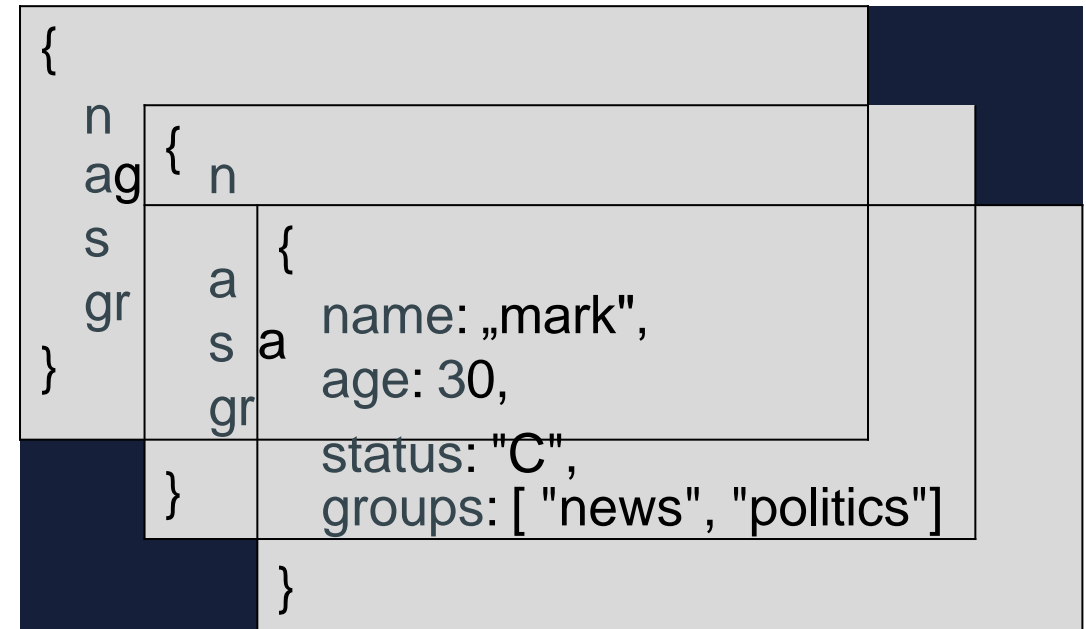
Gyűjtemények (Collections)

Összetartozó dokumentumok csoportjai

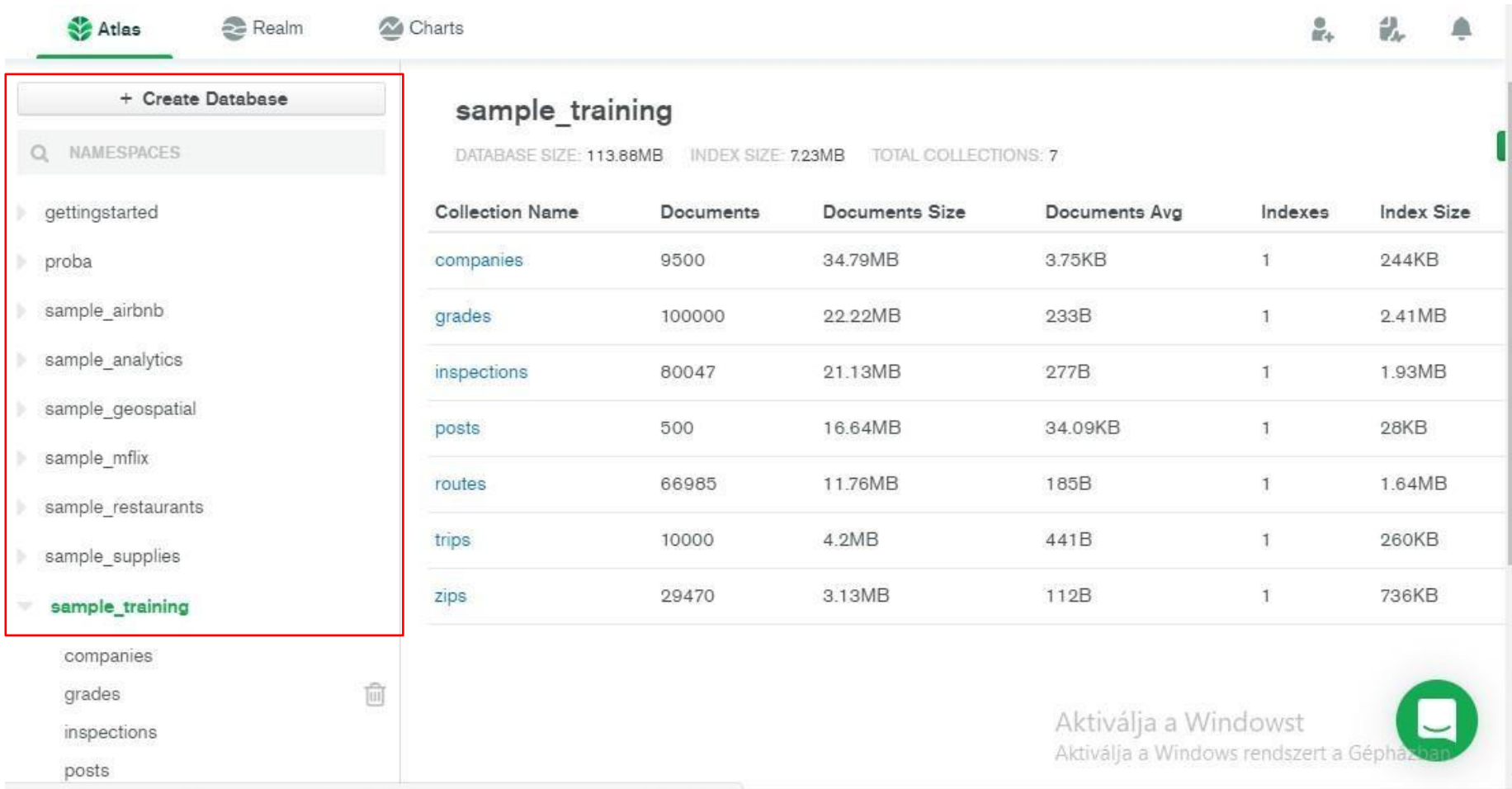
A gyűjtemények jellemzői:

- ☐ 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 of the interface, there is a Windows notification: 'Aktiválja a Windowst' (Activate Windows) with a link to 'Aktiválja a Windows rendszert a Gépházban' (Activate Windows in the Store).

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()`
– megszámlolja, 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 serveren 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.

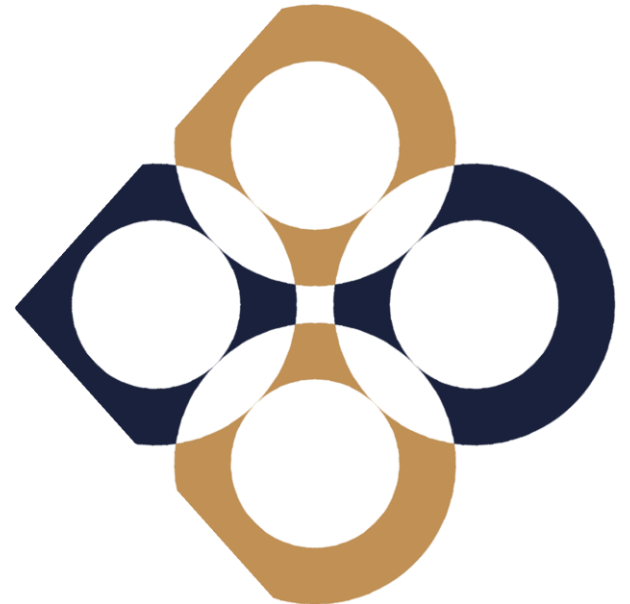
A mongo shell-ben adjunk hozzá
a ObjectId("5e5e9c470d33e9e8e3891b35") azonosítójú
dokumentum likes tömbjéhez még egy értéket,
mégpedig a 200-at!



**Köszönöm
a figyelmet!**

Adatbázisok előadás 09

Gráf adatbázisok

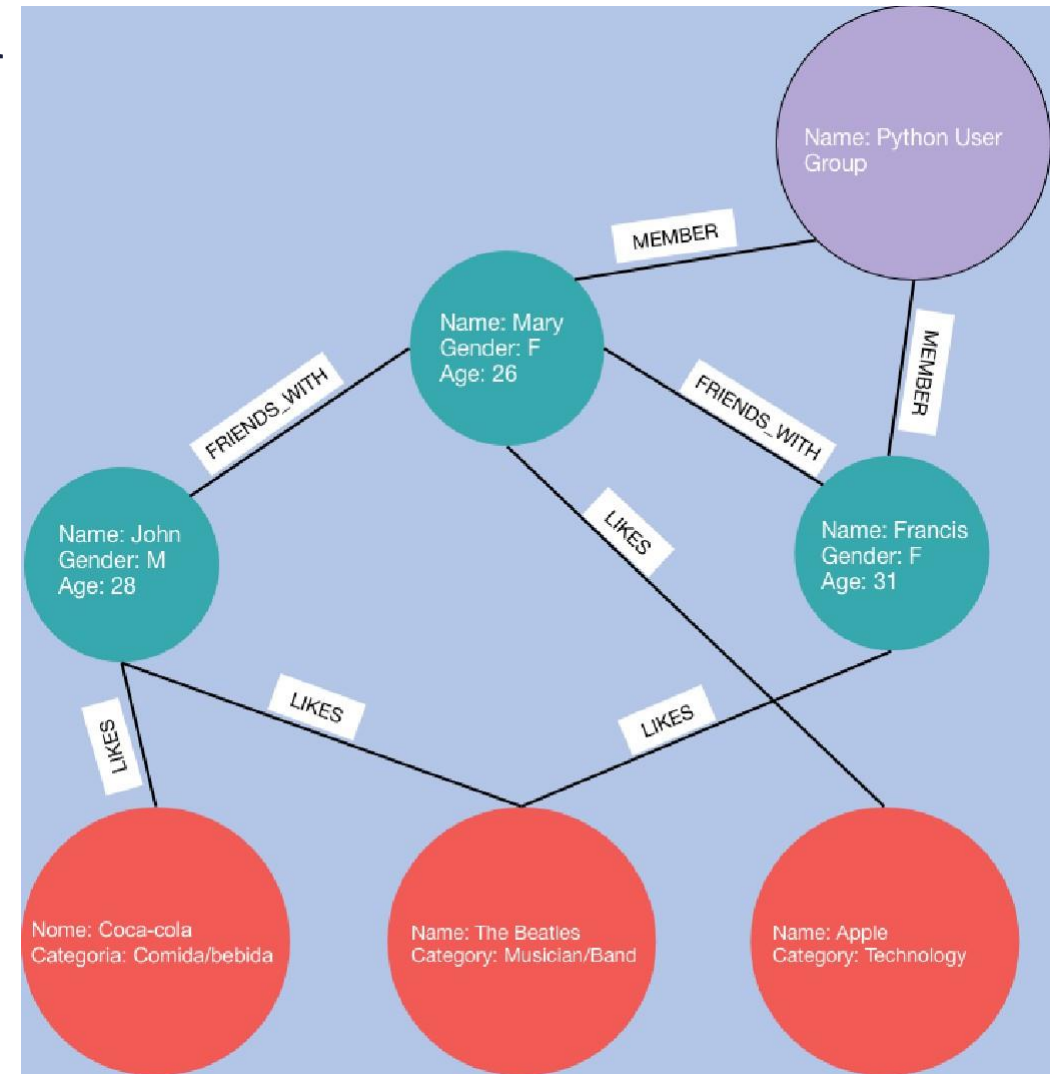


Miről lesz szó?

- ☐ Gráf adatbázisok jellemzői
- ☐ Neo4J adatbázis
 - ☐ Jellemzők
 - ☐ Lekérdezések – Cypher nyelv
 - ☐ CRUD műveletek
 - ☐ Indexek
 - ☐ Terminál
 - ☐ Elérés Python-ból

Olyan adatbázisok, amelyek az adatok tárolására és megjelenítésére gráf struktúrát alkalmaznak

- ❑ A gráf csúcsaiban vannak az adatok
 - ❑ Az adatok sémája nem rögzített
- ❑ A gráf élei jelentik a kapcsolatokat
 - ❑ Az élek irányítottak
 - ❑ Az éleknek adott név a kapcsolatra jellemző



<https://medium.com/labcodes/graph-databases-talking-about-your-data-relationships-with-python-b438c689dc89>

Gráf adatbázisok – előnyök és hátrányok

Előnyök

- Flexibilis séma
- Logikus, jól érthető lekérdezések
- Gyors adatelérés
- Nagymértékben összefüggő adatok kezelése
- Sokféle feladathoz megfelelők

Hátrányok

- OLTP rendszerekhez nem a legjobbak
- A nagy adatmennyiséget érintő lekérdezések nem optimálisak
- Sok esetben egy szerveren tárolódnak

Gráf adatbázisok – hol használják őket?

Tudásbázisok

Csalás felderítés

Termék ajánló rendszer

Social media

Törzsadatok kezelése

Hálózati infrastruktúra monitorozás

Gráf adatbázisok vs. Relációs adatbázisok

| Gráf adatbázisok | Relációs adatbázisok |
|---|--|
| Node | Tábla |
| Nincs séma | Fix séma |
| A kapcsolatok direkt módon definiáltak | A kapcsolatok idegen kulcsokkal valósulnak meg |
| A kapcsolódó adatok megjelenítése minták segítségével | A kapcsolódó adatok megjelenítése JOIN-okkal |

Gráf adatbázisok vs. Dokumentum adatbázisok

| Gráf adatbázisok | Dokumentum adatbázisok |
|---|--|
| Node | Document |
| Nincs séma | Nincs séma |
| Kapcsolatok a modellben | Kapcsolatok beágyazással vagy „idegen” kulcsokkal |
| A kapcsolódó adatok megjelenítése minták segítségével | A kapcsolódó adatok megjelenítése beágyazással vagy join-okkal |

Gráf adatbázisok - Példák



Gráf adatbázisok – Neo4j

- ☐ A legismertebb gráf adatbázis
- ☐ A csúcsok ~ entitások, objektumok
 - ☐ Lehetnek tulajdonságaik (kulcs-érték párok)
 - ☐ Az értékek primitív adattípusok
 - ☐ A tulajdonságok (részben) indexelhetők
 - ☐ Megadható UNIQUE kényszer
 - ☐ Nincs NULL elem
 - ☐ Lehetnek címkéik
- ☐ A kapcsolatok
 - ☐ Van nevük
 - ☐ Lehetnek tulajdonságaik
 - ☐ Indexelhetők



The #1 Database for Connected Data

Gráf adatbázisok – Neo4j

- ☐ Java-alapú gráf adatbázis
- ☐ Egyidejű hozzáférések kezelése (MVCC)
 - ☐ Minden tranzakció egy konzisztens snapshot-ot lát
 - ☐ Egyszerre több tranzakció tud írni/olvasni
- ☐ Tranzakciók esetén a konzisztencia teljesülése a preferált
- ☐ Egy serveres rendszerben az ACID feltételek is teljesülhetnek
- ☐ Flexibilis séma
- ☐ Elosztott rendszerben magas rendelkezésre állás és nagy teljesítmény
- ☐ Beépített gráf algoritmusok
- ☐ Index-mentes navigáció
- ☐ Szerepkör-alapú biztonság

Neo4j - Cypher

- ☐ A Neo4j preferált lekérdező nyelve
- ☐ Deklaratív (nem procedurális)nyelv
- ☐ Minta egyezéseket vizsgál
- ☐ Az emberi gondolkodáshoz közel álló nyelv
- ☐ Záradékok használata (pl: WHERE, ORDER BY)

#3: A Language For Connected Data

Cypher Query Language

```
SELECT dept#2Reportes.pid AS directReportes,
count(dept#2Reportes.directly_manages) AS count
FROM person_reporte_manager
JOIN person_reporte_l1Reportes
ON manager_directly_manages = l1Reportes.pid
JOIN person_reporte_l2Reportes
ON l1Reportes.directly_manages = l2Reportes.pid
WHERE manager_pid >= (SELECT id FROM person WHERE name = "Name Name")
GROUP BY directReportes
) AS T
GROUP BY directReportes)
UNION
(SELECT T.directReportes AS directReportes, sum(T.count) AS count
FROM
SELECT reporter_directly_manages AS directReportes, 0 AS count
FROM person_reporte_manager
JOIN person_reporte_reportee
ON manager_directly_manages = reporter_pid
WHERE manager_pid >= (SELECT id FROM person WHERE name = "Name Name")
GROUP BY directReportes)
UNION
SELECT l2Reportes.pid AS directReportes, count(l2Reportes.directly_manages)
AS count
FROM person_reporte_manager
JOIN person_reporte_l1Reportes
ON manager_directly_manages = l1Reportes.pid
JOIN person_reporte_l2Reportes
ON l1Reportes.directly_manages = l2Reportes.pid
WHERE manager_pid >= (SELECT id FROM person WHERE name = "Name Name")
GROUP BY directReportes
) AS T
GROUP BY directReportes)
UNION
(SELECT l2Reportes.directly_manages AS directReportes, 0 AS count
FROM person_reporte_manager
JOIN person_reporte_l1Reportes
ON manager_directly_manages = l1Reportes.pid
JOIN person_reporte_l2Reportes
ON l1Reportes.directly_manages = l2Reportes.pid
WHERE manager_pid <= (SELECT id FROM person WHERE name = "Name Name")
),
```



```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      {sub}-[:MANAGES*1..3]->{report}
WHERE boss.name = "John Doe"
RETURN sub.name AS Subordinate,
       count(report) AS Total
```



Less time writing queries

- More time understanding the answers
- Leaving time to ask the next question

Less time debugging queries:

- More time writing the next piece of code
- Improved quality of overall code base

Code that's easier to read:

- Faster ramp-up for new project members
- Improved maintainability & troubleshooting

<https://twitter.com/amyhodler/status/1233437495624253442>

Fontosabb Cypher adattípusok

| Típus | Példa | Megjegyzés |
|--------------|--------------------------------------|-------------------|
| Integer | 13 | Tulajdonság típus |
| Float | 3.14 | Tulajdonság típus |
| String | 'Hello', "World" | Tulajdonság típus |
| Boolean | true, false | Tulajdonság típus |
| Date | "2019-06-01" | Tulajdonság típus |
| Time | "21:40:32" | Tulajdonság típus |
| DateTime | "2019-09-25T06:29:39Z" | Tulajdonság típus |
| Node | (a:Actor) | Szerkezet típus |
| Relationship | [d:Directed] | Szerkezet típus |
| Path | (a:Actor)-[:Acted_in]->(m:Movie) | Szerkezet típus |
| List | [0, 1, 2] | Összetett típus |
| Map | {kulcs1: érték1, kulcs2: érték2 ...} | Összetett típus |

Fontosabb Cypher operátorok

| Operátor típus | Példák |
|------------------------|---|
| Matematikai | +, -, *, /, %, ^ |
| Összehasonlító | =, <, >, <>, <=, >=, IS NULL, IS NOT NULL |
| Szöveg összehasonlító | STARTS WITH, ENDS WITH, CONTAINS |
| Logikai | NOT, AND, OR, XOR |
| Szöveg | + (összefűzés), =~ (regex) |
| Aggregációs | DISTINCT |
| Tulajdonság (property) | . (csomópont vagy kapcsolat tulajdonság elérése) = (csomópont vagy kapcsolat tulajdonságok felülírása) += (csomópont vagy kapcsolat tulajdonság módosítása, hozzáadása) |
| Lista | IN (tartalmazást vizsgál) + (összefűz) [] (listaelemek elérése) |

Fontosabb Cypher függvények

| Függvény típus | Példák |
|----------------|--|
| Matematikai | abs(), round(), rand(), sqrt(), log(), sin(), cos(), |
| Szöveg | left(), right(), toLower(), toUpper(), trim(), substring() |
| Predikátum | exists(), all(), any(), isEmpty() |
| Skalár | id(), type(), toFloat(), toInteger, toBoolean() |
| Lista | labels(), nodes(), relationships(), range() |
| Dátum/Idő | date(), datetime(), time() |

A Case kifejezés Cypher-ben

CASE kifejezés

WHEN értéke1 THEN eredmény1

WHEN értéke2 THEN eredmény2

...

[ELSE default_érték]

END

Neo4j - lekérdezések

MATCH() - Csúcsok, kapcsolatok, tulajdonságok, címkék és minták keresése az adatbázisban

- ☐ A SQL SELECT-hez hasonló elven működik
- ☐ A lekérdezés által visszaadott értékeket a RETURN kulcsszó után adhatjuk meg
- ☐ A lekérdezés eredményét a WHERE kulcsszó után megadott feltételekkel szűrhetjük
- ☐ A megjelenítendő eredményt a LIMIT kulcsszóval korlátozhatjuk
- ☐ Az eredményt többféle nézetben (Graph, Table, Text, Code) is megtekinthetjük

Neo4j – Egyszerű lekérdezések I.

```
MATCH (n)  
RETURN n
```

Listázza az összes csúcsot

```
MATCH (p:Person)  
RETURN p  
LIMIT 1
```

Megjeleníti a legelső személyt

```
MATCH (p:Person {name: 'Tom Hanks'})  
RETURN p
```

Megjeleníti Tom Hanks adatait

```
MATCH (:Person {name: 'Tom Hanks'})-[:DIRECTED]->(movie:Movie)  
RETURN movie.title
```

Megjeleníti, hogy Tom Hanks
milyen film(ek)et rendezett

Neo4j – Egyszerű lekérdezések II.

```
MATCH (p:Person {name:'Tom Hanks'})-[rel:DIRECTED]-(m:Movie)
RETURN p.name AS name, p.born AS `Year Born`, m.title AS title,
m.released AS `Year Released`
```

Megjeleníti Tom Hanks és az általa rendezett film egyes adatait

```
MATCH (:Person)-[:DIRECTED]->(m:Movie)
RETURN DISTINCT m.released
```

Megjeleníti azon éveket, amikor filmeket rendeztek

```
MATCH (j:Person)
WHERE j.born = 1955
RETURN j
```

Megjeleníti az 1955-ben született személyeket

```
MATCH (j:Person)
WHERE NOT j.born = 1955
RETURN j
```

Megjeleníti azokat, akik nem 1955-ben születtek

Neo4j – Egyszerű lekérdezések III.

```
MATCH (p:Person)
WHERE p.name STARTS WITH 'M'
RETURN p.name
```

Megjeleníti az M betűvel kezdődő személyeket

```
MATCH (p:Person)
WHERE p.name CONTAINS 'a'
RETURN p.name
```

Megjeleníti azon személyeket, akik nevében van „a” betű

```
MATCH (p:Person)
WHERE p.name ENDS WITH 'n'
RETURN p.name
```

Megjeleníti azon személyeket, akik neve n-re végződik

```
MATCH (p:Person)
WHERE p.name =~ 'Jo.*'
RETURN p.name
```

Reguláris kifejezéssel szűr a személyek nevére

Neo4j – Egyszerű lekérdezések IV.

```
MATCH (m:Movie)
WHERE ID(m) IN [0, 5, 9]
RETURN m
```

Megjeleníti a 0, 5 és 9 azonosítójú filmeket

```
MATCH (p:Person)-[d:REVIEWED]->(m:Movie)
RETURN p, d, m
```

Megjeleníti, hogy melyik személy milyen filmekről írt kritikát

```
MATCH (p:Person)-[d:WROTE]->(m:Movie)
WHERE not exists ((p)-[:ACTED_IN]->(m))
RETURN p, d, m
```

Megjeleníti azokat a személyeket és filmeket, ahol az író nem szerepelt a filmben

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(p2:Person)
WHERE p.name= 'Gene Hackman'
AND exists( (p2)-[:DIRECTED]->(m) )
RETURN p, p2, m
```

Kivel és milyen filmben szerepelt együtt Gene Hackman, ha a másik szereplő egyben rendező is volt?

Neo4j – Egyszerű lekérdezések V.

```
MATCH (p:Person)
WHERE p.name STARTS WITH 'J'
OPTIONAL MATCH (p)-[:DIRECTED]->(m)
RETURN p.name, m.title
```

Megjeleníti a személyeket és az általuk rendezett filmet (ha van olyan)

```
MATCH (p:Person)
RETURN count(*)
```

Megjeleníti, hogy hány személy van az adatbázisban

```
MATCH (p:Person)-[:FOLLOWS]->(p2:Person)
WITH p, count(*) AS db
RETURN p.name, db
```

Megjeleníti azt, hogy melyik személy hány másikat követ

```
MATCH (p:Person)-[:WROTE]->(m:Movie)
RETURN p.name, collect(m.title) AS filmek
```

Megjeleníti, hogy melyik személy milyen filmeket rendezett

Neo4j – Egyszerű lekérdezések VI.

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, size(collect(m.title)) AS db
```

Megjeleníti, hogy melyik személy
hány filmben szerepelt

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, size(collect(m.title)) as db
ORDER BY db DESC, p.name
LIMIT 5
```

Megjeleníti, hogy kik szerepeltek a
legtöbb filmben – az első 5

```
MATCH (p:Person)-[r]->(p2:Person)
RETURN type(r), count(*)
```

Megjeleníti azt, hogy milyen
típusú és hány db kapcsolat van a
személyek között

```
MATCH (p:Person)-[r]->(m:Movie)
WHERE p.born IS NULL
RETURN p.name, type(r), m.title, avg(date().year-m.released)
```

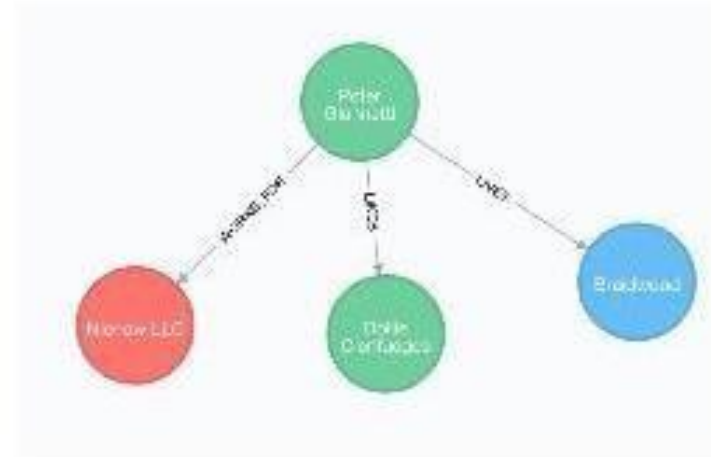
Megjeleníti, hogy azok a személyek, akiknek
nincs megadva a születési évük, milyen
filmekkel vannak kapcsolatban, és a filmek
átlagosan hány éve jelentek meg

Cypher példa*

Cypher: Example Query



```
MATCH (p:Person {fullName : "Peter Giannetti"})-[r]-(n)  
RETURN p, r, n
```



<https://neo4j.com/docs/developer-manual/current/cypher/>

<https://www.opencypher.org/>

Neo4j – CRUD műveletek

CREATE (változónév:címke {tulajdonságok:értékek}) -- Csomópont létrehozása

SET – Cimkék, tulajdonságok és kapcsolatok módosítása

REMOVE – Cimkék és tulajdonságok törlése

DELETE – Csomópontok és kapcsolatok törlése

Neo4j – CRUD műveletek I.

```
CREATE (:Movie {title: 'Félelem', released: 2011, tagline: 'Amit mindenki  
érez' })
```

Létrehoz egy új filmet

```
create (:Person {name: 'Kiss Ilona', born: 1988 }),  
(:Person {name: 'Nagy Béla', born: 2000 })
```

Létrehoz két új személyt

```
MATCH (a:Person), (b:Movie)  
WHERE a.name = 'Kiss Ilona' AND b.title = 'Félelem'  
CREATE (a)-[:FOLLOWS]->(b)
```

Létrehoz új kapcsolatot meglévő
csúcsok között

```
create (p:Person {name: 'Fekete Edit', born: 1997})-[:WROTE]->(m:Movie  
{title: 'A hősnő', released: 2021})  
return (p)-[]-(m)
```

Egyszerre hoz létre új személyt és
filmet, valamint kapcsolatot
köztük

Neo4j – CRUD műveletek II.

```
MATCH (p:Person  
{name: 'Fekete Edit'})  
SET p.born = 2010  
RETURN p
```

Módosítja az adott személy születési évét

```
MATCH (p:Person {name: 'Fekete Edit'})  
SET (case when p.born < 2015 then p end).born = 2015  
RETURN p
```

Módosítja az adott személy születési évét, ha teljesül egy feltétel

```
MATCH (p:Person {name: 'Fekete Edit'})  
REMOVE p.born  
RETURN p
```

Törli az adott személy születési évét

```
MATCH (n {name: 'Fekete Edit'})  
REMOVE n:Person  
RETURN n.name, labels(n)
```

Törli az adott csúcs címkéjét

Neo4j – CRUD műveletek II.

```
MATCH (n:Person {name: 'Fekete Edit'})  
DELETE n
```

Törli az adott csomópontot

```
MATCH (p:Person {name: 'Kiss Ilona'})-[r:FOLLOWS]->(m:Movie)  
DELETE r  
RETURN p
```

Egy adott kapcsolat törlése

```
MATCH (n {name: 'Kiss Ilona'})  
DETACH DELETE n
```

Törli az adott csomópontot és minden kapcsolatát

```
MATCH (n)  
DETACH DELETE n
```

Töröl minden csomópontot és kapcsolatot

Neo4j - Indexek

CREATE INDEX – index létrehozása

SHOW INDEXES [VERBOSE]– indexek listázása

DROP INDEX – index törlése

- ☐ A VERBOSE segítségével opcionálisan részletesebb lista jeleníthető meg
- ☐ A PROFILE utasítással megjeleníthető a végrehajtási terv
- ☐ Az EXPLAIN utasítás hasonlóan működik, de magát az utasítást nem hajtja végre, csak a végrehajtási tervet jeleníti meg

Neo4j – Indexek - példák

```
profile  
match (p:Person)  
return p
```

A lekérdezés és végrehajtási terv

```
CREATE INDEX i_name  
IF NOT EXISTS FOR (n:Person)  
ON (n.name)
```

A személyeket indexeli név alapján, ha még nincs index

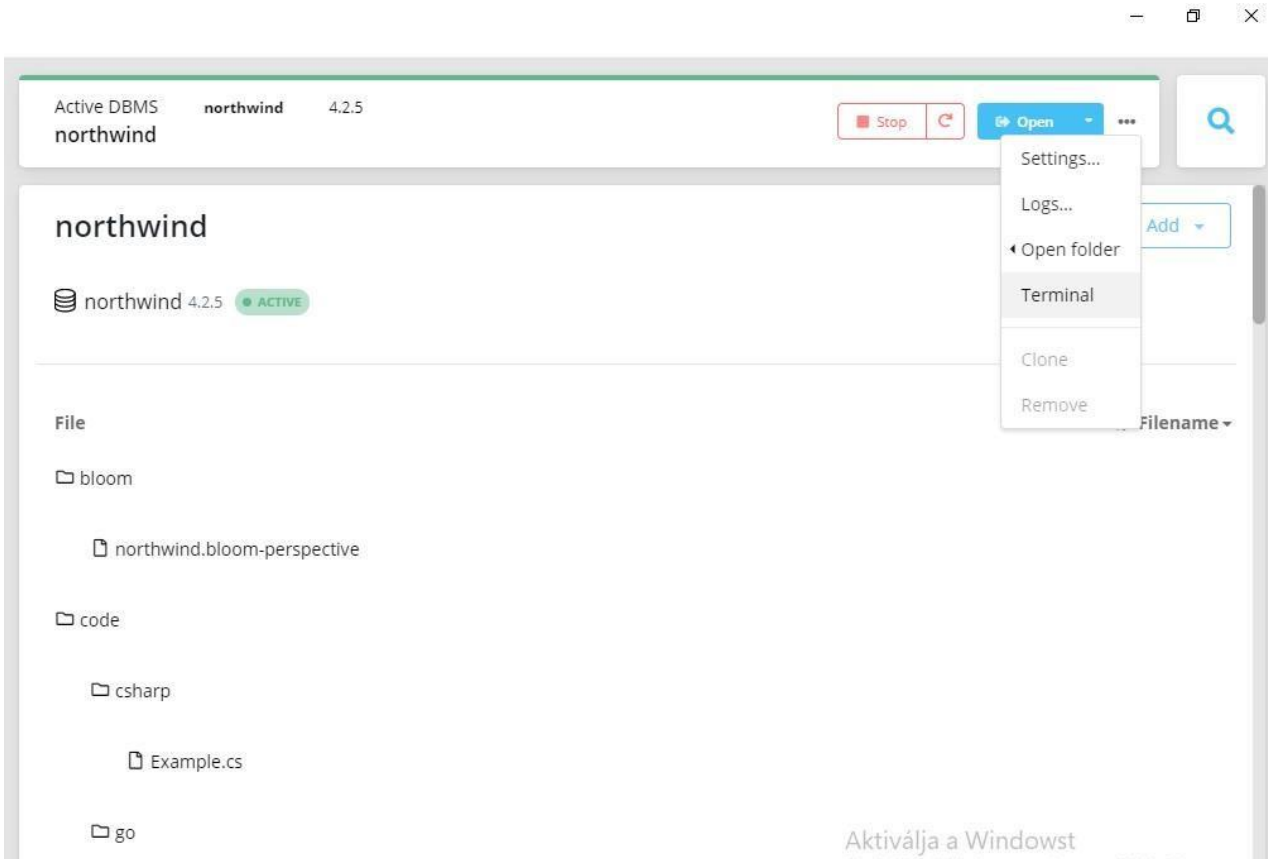
```
CREATE INDEX i_filmek  
FOR (m:Movie)  
ON (m.title, m.released)
```

Összetett index létrehozása

```
DROP INDEX i_filmek
```

Törli az adott indexet

Cypher-shell terminal



```
Neo4j Desktop Terminal - northwind
File Edit View Window Help Developer

gmolnar@northwind>
gmolnar@northwind> show databases;
+-----+
| name      | address          | role          | requestedStatus | currentStatus |
| error | default |
+-----+
+-----+
| "neo4j"    | "localhost:7687" | "standalone" | "online"        | "online"      |
| ""         | TRUE            |              |                 |               |
| "northwind" | "localhost:7687" | "standalone" | "online"        | "online"      |
| ""         | FALSE          |              |                 |               |
| "system"   | "localhost:7687" | "standalone" | "online"        | "online"      |
| ""         | FALSE          |              |                 |               |
| "tanulo"   | "localhost:7687" | "standalone" | "online"        | "online"      |
| ""         | FALSE          |              |                 |               |
+-----+

4 rows available after 11 ms, consumed after another 3 ms
gmolnar@northwind> show users;
+-----+
| user      | roles              | passwordChangeRequired | suspended |
+-----+
+-----+
| "gmolnar" | ["admin", "PUBLIC"] | FALSE                  | FALSE     |
| "neo4j"   | ["admin", "PUBLIC"] | FALSE                  | FALSE     |
+-----+

2 rows available after 8 ms, consumed after another 2 ms
gmolnar@northwind> 
```

Elérés Python-ból*

```
!pip install neo4j
from neo4j import GraphDatabase

class Neo4jConnection:
    ...
    return response

conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="neo4j")

query_string = 'match (n) return n limit 4'

conn.query(query_string, db='northwind')
```

A teljes kód a mellékelt python.ipynb fájlban található

Feladatok megoldása I.

A Neo4J Desktop-ban hozzon létre új projektet, majd egy új adatbázist tanulo néven! Nyissa meg a Neo4J Browsers, majd tegye aktívvá az új adatbázist!

a. Hozzon létre :Tanulo és :Tanar csomópontokat az alábbi ábra alapján:

| TANULO | | |
|------------|---------|-------|
| Nev | Eletkor | Atlag |
| Kiss Béla | 22 | 3.5 |
| Nagy Ilona | 23 | 4.4 |

| TANAR | |
|------------|-------------|
| Nev | Szak |
| Tóth Ottó | Matematika |
| Nagy Ivett | Informatika |

Feladatok megoldása II.

Az előző feladatban létrehozott tanulo adatbázisban hozzon létre két új kapcsolatot :Tanit néven az alábbiak szerint:

- a. Tóth Ottó tanítja Kiss Bélát
- b. Nagy Ivett tanítja Nagy Ilonát
- c. A szükséges utasításokat adja meg válaszként!

Feladatok megoldása III.

A 7. feladatban létrehozott tanulo adatbázisban végezze el a következő módosításokat:

- a. Nagy Ilona átlaga legyen 5.0
- b. Tóth Ottó szakja legyen Fizika
- c. A szükséges utasításokat adja meg válaszként!

Feladatok megoldása IV.

A Neo4J Desktop-ban tegye aktívvá a tanulo projektet, majd nyisson új terminált az adatbázis melletti Open gombnál mellett lévő három pont (...) kiválasztásával! Utána lépjen be a bin mappába, majd adja ki a cypher-shell parancsot!

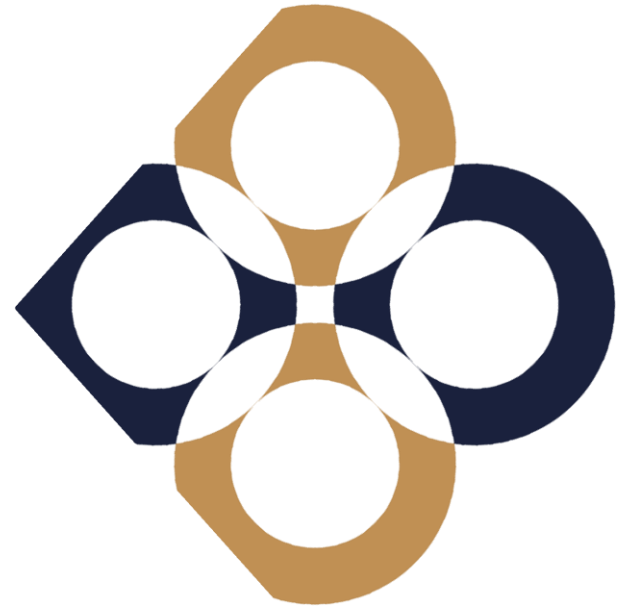
- a. Szükség esetén adja meg a felhasználónevet és a jelszót
- b. Csatlakozzon a tanulo adatbázishoz (:use tanulo;)
- c. Kérdezze le az első két csúcsot!



**Köszönöm
a figyelmet!**

Adatbázisok előadás 10.

Kulcs-érték adatbázisok



Miről lesz szó?

- ☐ Kulcs-érték adatbázisok jellemzői
- ☐ A Redis adatbázis
 - ☐ Jellemzők
 - ☐ Adattípusok
 - ☐ Utasítások
 - ☐ Terminál
 - ☐ Elérés Python-ból
 - ☐ Feladatok megoldása

Kulcs-érték adatbázisok

Olyan adatbázisok, amelyek az adatokat kulcs-érték párokban tárolják, és az adatok a kulcs alapján gyorsan elérhetők

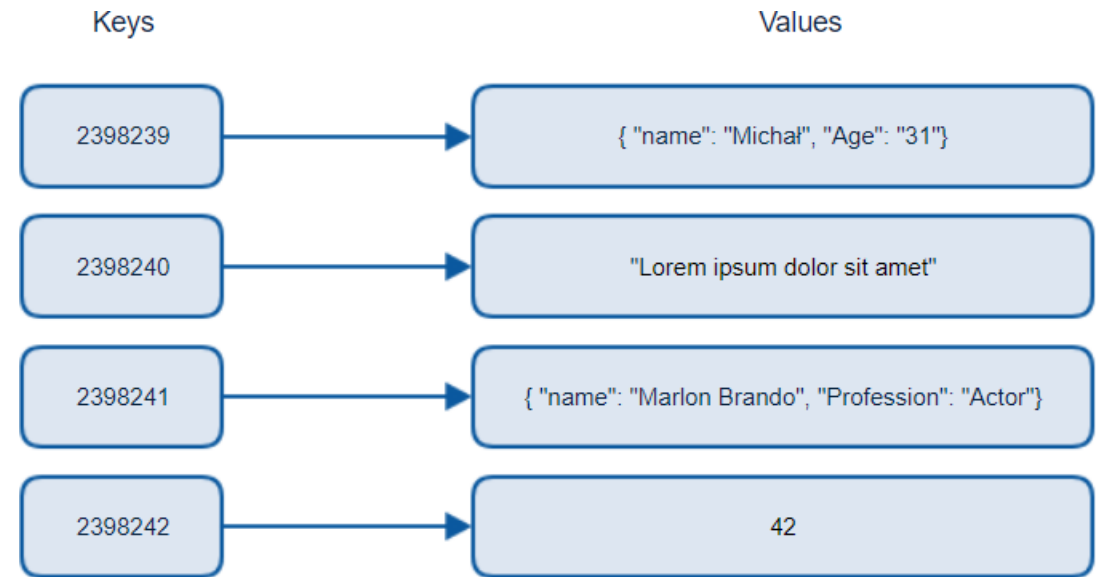
❑ Asszociatív tömbök (kulcs-érték párok)

❑ A kulcs **egyedi** azonosító

A kulcs és az érték lehet akár egyszerű, akár komplex adat

A kulcs-érték párnak lehet érvényességi ideje

Nagymértékben skálázhatók és partícionálhatók



Kulcs-érték adatbázisok – előnyök és hátrányok

Előnyök

- Egyszerűség
- Nagy hozzáférési sebesség
- Flexibilitás
- Skálázhatóság
- Kisebb tárolási méret
- Indexekre – alapesetben – nincs szükség

Hátrányok

- Bonyolultabb adatmodellekhez nem javasolt
- Limitált lekérdezési lehetőségek
- Sok feladatot alkalmazás szintjén kell megoldani (pl. kényszerek)
- Adatok exportálása
- Adatok közötti kapcsolatok kezelése

Kulcs-érték adatbázisok – hol használják őket?

Bevásárlókocsi

Cache

Felhasználói profil

Fórum

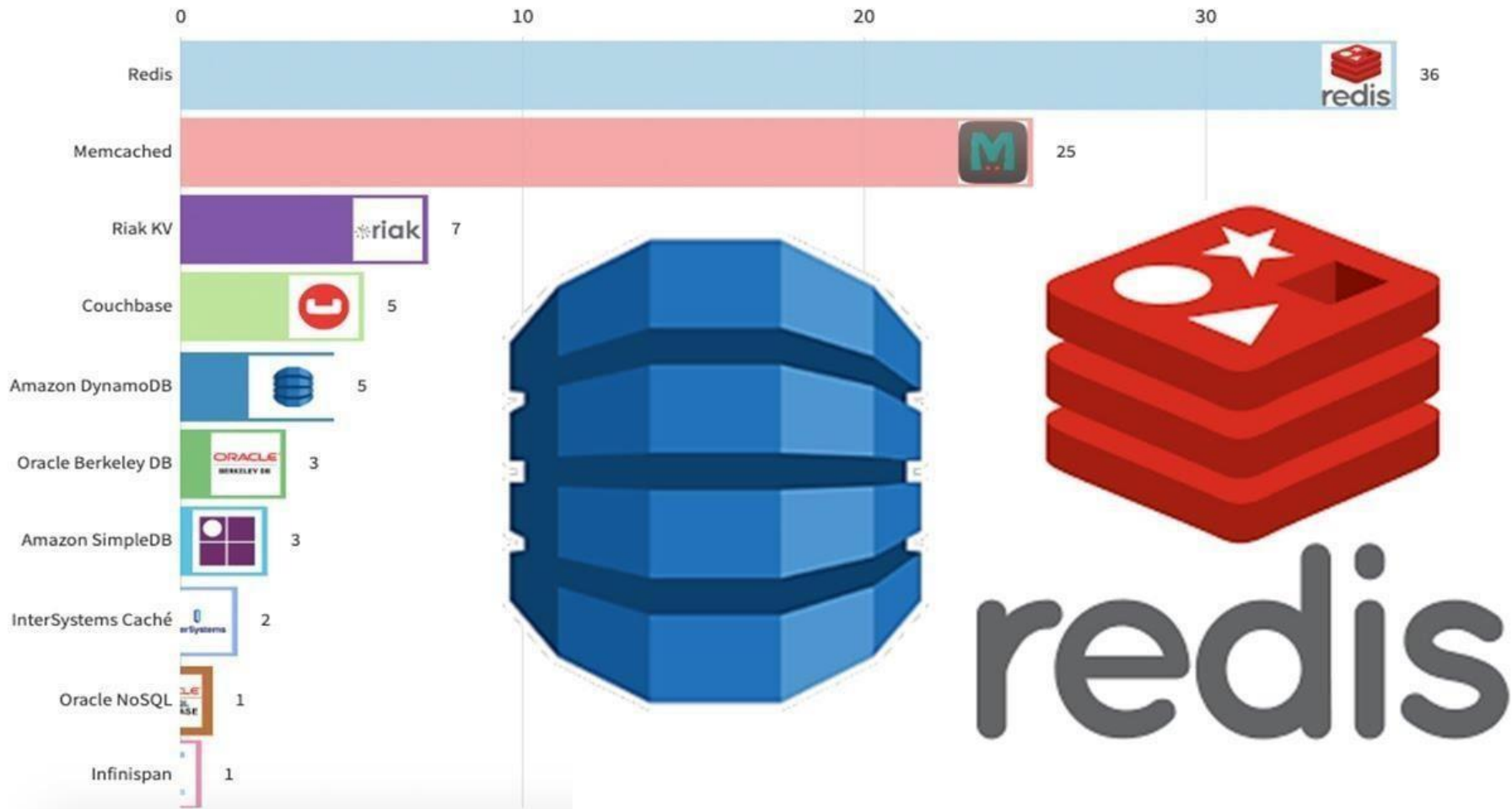
Valós idejű ajánló rendszerek

Kulcs-érték adatbázisok vs. Relációs adatbázisok

| Kulcs érték adatbázisok | Relációs adatbázisok |
|---|--|
| Adatbázis | (Logikai) Adatbázis |
| Sorted set | Tábla |
| Hash | Rekord |
| Flexibilis séma | Fix séma |
| A kapcsolatok megfelelő kulcs elnevezésekkel valósíthatók meg | A kapcsolatok idegen kulcsokkal valósulnak meg |
| Nincs beépített lekérdezési lehetőség | Lekérdezés SQL nyelven |

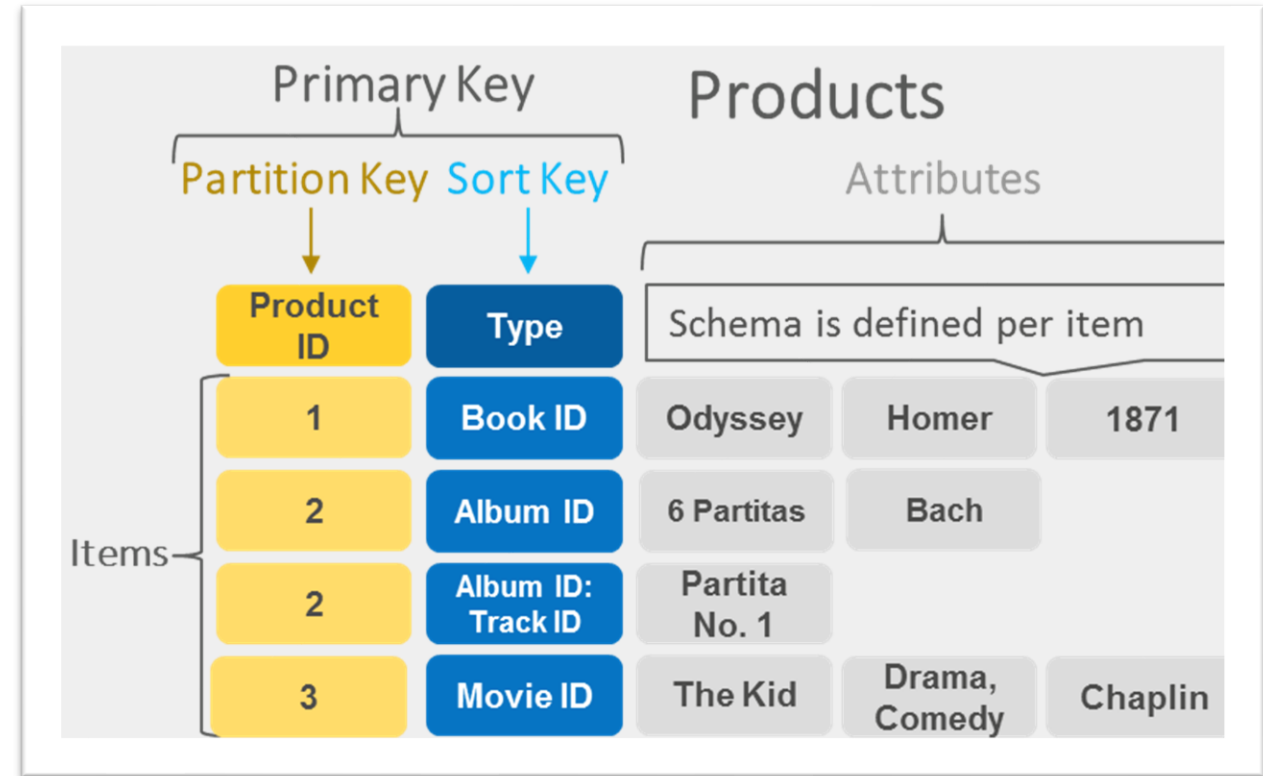
Kulcs-érték adatbázisok - Példák

Most Popular Database Engine of Key-value Stores (2012 - 2019)



Kulcs-érték adatbázisok - DynamoDB

- ❑ A kulcs-érték adatbázisok úttörője
- ❑ Állítható konzisztencia szint
- ❑ Költséghatékony
- ❑ Hibatűrő
- ❑ Integrálható más szolgáltatásokkal



Kulcs-érték adatbázis – Redis

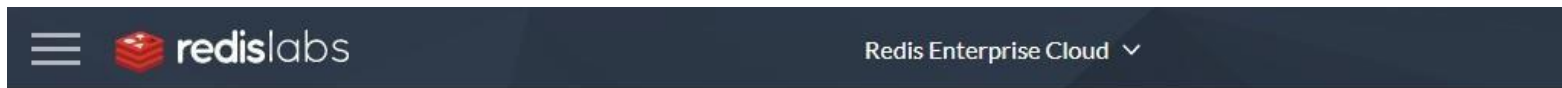
- ❑ Nyílt forráskódú adatbázis
- ❑ Kulcs-érték adatmodell, in-memory adatbázis
- ❑ Futtatási/telepítési lehetőségek
 - ❑ Cloud próba ([Try Redis](#), illetve [Try Redis Enterprise | Redis Labs](#))
 - ❑ On-premise

[Redis](#) - Linux

[How To Install Redis on Windows 10 - DEV Community](#) – Windows

[Install and config Redis on Mac OS X via Homebrew | by Pete Houston | Medium - Mac](#)

Redis – Cloud



View Database    

Database Name

Metrics

Slowlog

Configuration

| | |
|---|--|
| Subscription | #1406889 Redis Cloud/Fixed Plan/AWS/eu-central-1/Standard/30MB |
| Protocol | Redis |
| Used Memory | 2.03 MB |
| Replication  | Disabled |
| Activated On | 03/26/2021 20:56:01 |
| Last Changed | 03/26/2021 20:56:38 |

A csatlakozáshoz szükséges adatok:

- ☐ Port
- ☐ Endpoint
- ☐ Default user password

Csatlakozási lehetőségek

- ☐ redis-cli (parancssoros interfész)
- ☐ Programkód (pl. Python)

Redis – Cli

```
C:\Windows\System32\cmd.exe - redis-cli -h redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com -p 13613 -a iH3vsvrHTZog4DXLtmvQUMn240NFq0iO
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. Minden jog fenntartva.

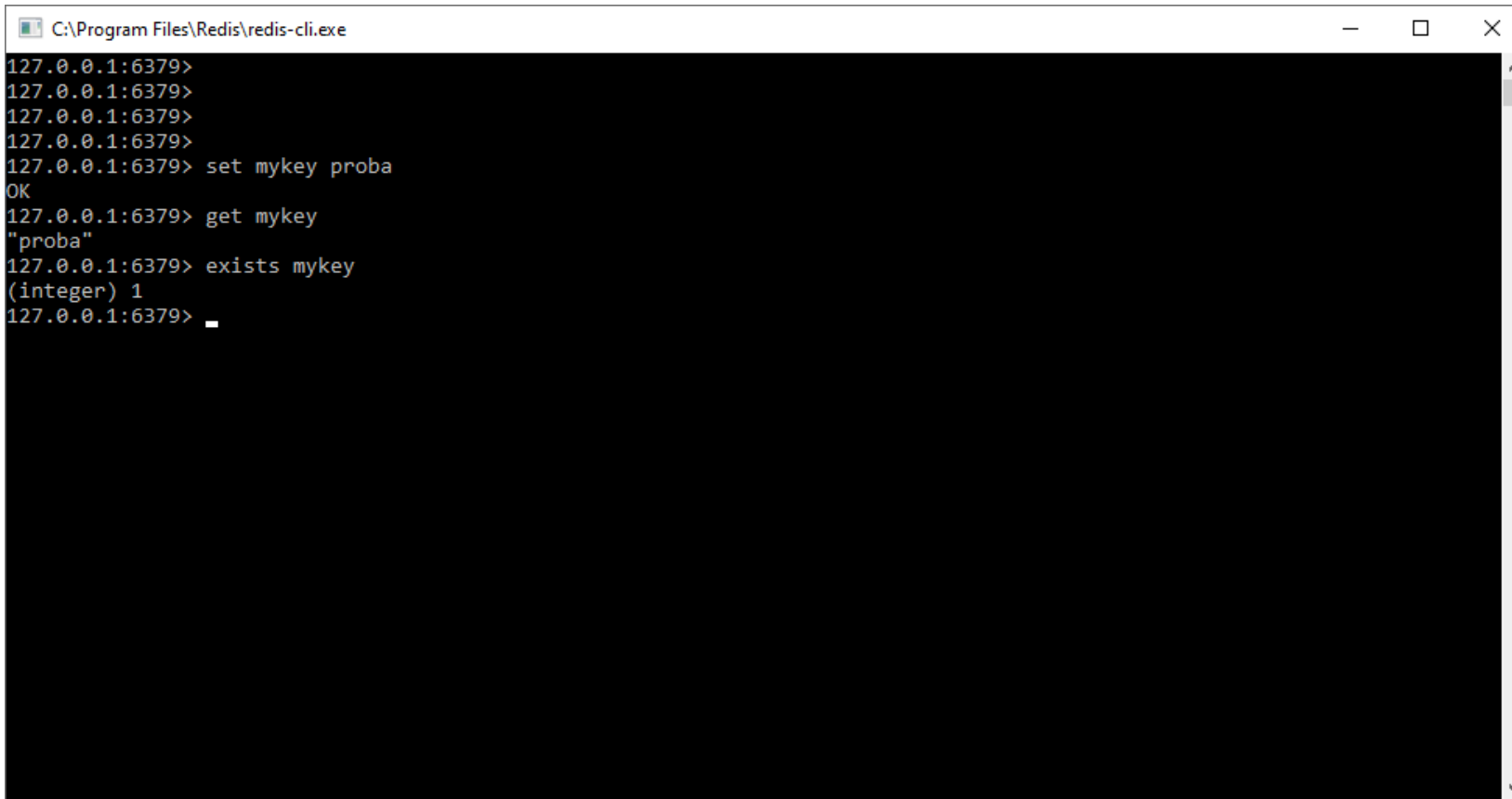
C:\Program Files\Redis>redis-cli -h redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com -p 13613 -a iH3vsvrHTZog4DXLtmvQUMn240NFq0iO
redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com:13613> ping
PONG
redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com:13613> set hello world
OK
redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com:13613> get hello
"world"
redis-13613.c250.eu-central-1-1.ec2.cloud.redislabs.com:13613>
```

Aktiválja a Windowst
Aktiválja a Windows rendszert a Gépházban.

Csatlakozás a Redis Cloud adatbázishoz:

```
redis-cli -h <endpoint> -p <port> -a <password>
```

Redis – On-premise

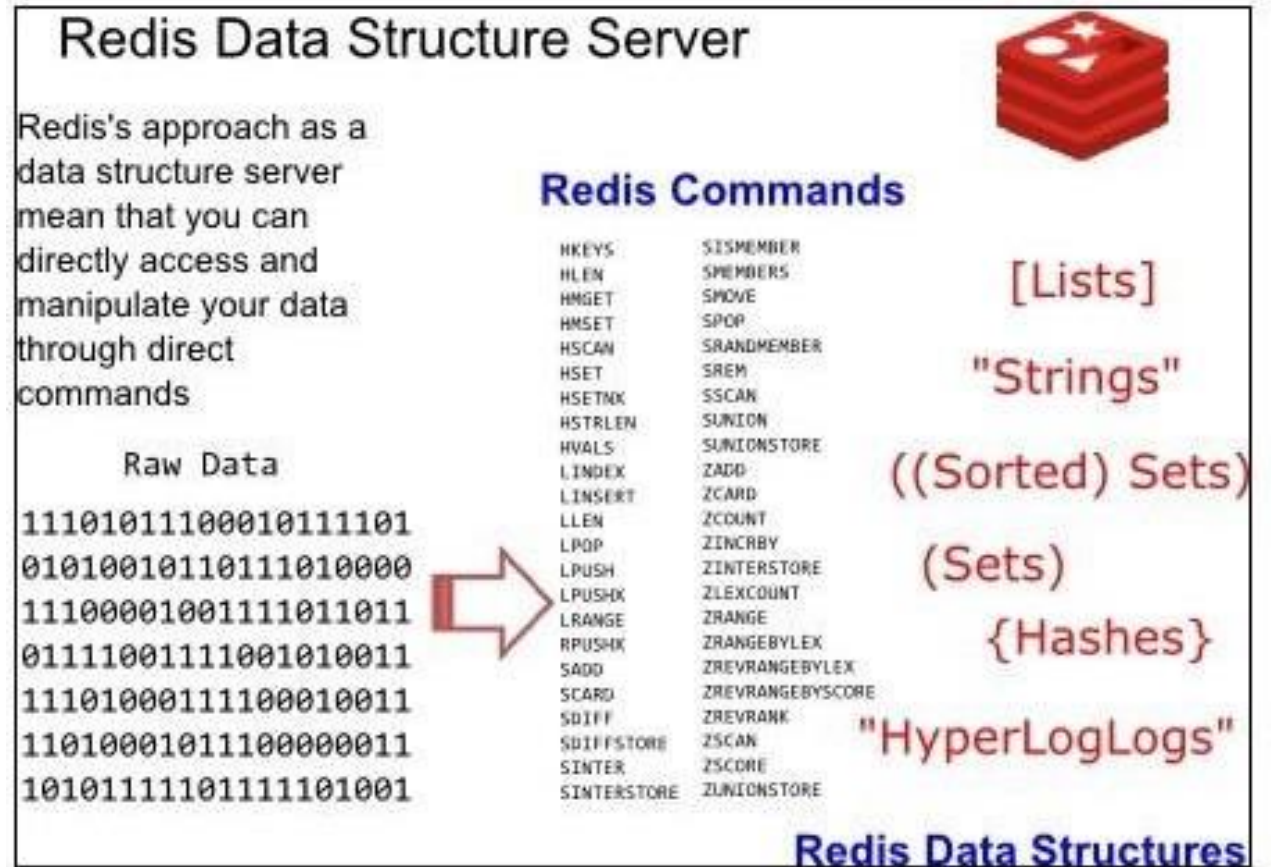
A screenshot of a Windows command prompt window titled "C:\Program Files\Redis\redis-cli.exe". The terminal shows a series of commands and responses from the Redis server running on 127.0.0.1:6379. The commands include multiple empty prompts, a 'set mykey proba' command returning 'OK', a 'get mykey' command returning '"proba"', and an 'exists mykey' command returning '(integer) 1'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Program Files\Redis\redis-cli.exe
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> set mykey proba
OK
127.0.0.1:6379> get mykey
"proba"
127.0.0.1:6379> exists mykey
(integer) 1
127.0.0.1:6379> _
```

Csatlakozás az on-premis rendszerhez: redis-cli

Redis adattípusok

- ☐ Key – tulajdonság neve*
- ☐ String – szöveg
- ☐ List – szöveges adatok rendezett listája
- ☐ Hashes – objektumhoz hasonló adattípus, amely objektum kulcsból és kulcs-érték párokból áll
- ☐ Set – egyedi szöveges adatok rendezetlen halmaza
- ☐ Sorted set – egyedi szöveges adatok rendezett halmaza



Utasítások* I. – Key, string

SET kulcs érték [opciók]

Új kulcs-érték pár létrehozása, vagy a meglévő felülírása
Fontosabb opciók: EX seconds – lejárati idő, NX – csak akkor jön létre a kulcs, ha még nem létezik, GET – a régi érték megjelenítése, amennyiben létezik

GET kulcs

A kulcs értékének lekérdezése – ha nem létezik, akkor a visszaadott érték üres (nil)

EXISTS kulcs(ok)

Ellenőrzi, hogy az adott kulcs létezik-e. Egymás után szóközzel elválasztva több kulcs is megadható

DEL kulcs(ok)

Törli a megadott kulcsot vagy kulcsokat

* A Redis verziótól függően a felsorolt utasításoknak csak egy része támogatott

Utasítások I. – Key, string (folytatás)

COPY kulcs1 kulcs2 [logikai adatbázis]
[REPLACE]

A kulcs1 értékével létrehozza a még nem létező kulcs2-t. A kulcs2 lehet egy másik logikai adatbázisban is. A REPLACE segítségével törölhető a meglévő kulcs2.

MOVE kulcs logikai_adatbázis

A kulcsot átmozgatja egy másik logikai adatbázisba, amennyiben ott az nem létezik

RENAME kulcsnév újkulcsnév

Új nevet ad a kulcsnak

EXPIRE kulcs seconds

Beállítja a kulcs élettartamát (másodpercben)

TYPE kulcs

Megadja a kulcshoz tartozó érték típusát

Utasítások I. - Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> set mainap hetfo
OK
127.0.0.1:6379> get mainap
"hetfo"
127.0.0.1:6379> set mainap hetfo ex 5
OK
127.0.0.1:6379> get mainap
(nil)
127.0.0.1:6379> exists mainap
(integer) 0
127.0.0.1:6379> set mainap kedd
OK
127.0.0.1:6379> exists mainap
(integer) 1
127.0.0.1:6379> del mainap
(integer) 1
127.0.0.1:6379> exists mainap
(integer) 0
```

- ☐ Létrehozunk egy új kulcsot mainap néven, amelynek értéke hetfo
- ☐ Lekérdezzük az értékét, majd beállítjuk a lejáratí idejét 5 másodperce
- ☐ Utána a kulcs törlődik (nil), azaz már nem létezik
- ☐ Ismét létrehozzuk a mainap kulcsot, most értéke kedd
- ☐ Ellenőrizzük, hogy létezik-e, majd töröljük. Utána ismét megnézzük, hogy létezik-e

Utasítások II. – Hashes

HSET | HMSET kulcs mező érték [mező érték]

Beállítja a mező(k) értékét

HGET kulcs mező

Lekérdezi a kulcs adott mezőjének értékét

HGETALL kulcs

Lekérdezi a kulcs összes mezőjének értékét

HEXISTS kulcs mező

Megadja, hogy létezik-e a kulcs adott mezője

HDEL kulcs mező [kulcs mező]

Törli az adott mező(ke)t.

HKEYS kulcs

Megadja, hogy milyen mezői vannak a kulcsnak

Utasítások II. - Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379> hmset tanulo nev "Kiss Bela"
szulido 2000.01.01 neptun abcdef
OK
127.0.0.1:6379> hgetall tanulo
1) "nev"
2) "Kiss Bela"
3) "szulido"
4) "2000.01.01"
5) "neptun"
6) "abcdef"
127.0.0.1:6379> type tanulo
hash
127.0.0.1:6379> hkeys tanulo
1) "nev"
2) "szulido"
3) "neptun"
127.0.0.1:6379> hdel tanulo neptun
(integer) 1
127.0.0.1:6379>
```

- ☐ Létrehozzuk a tanulo kulcsot több mezővel (nev, szulido, neptun)
- ☐ Lekérdezzük a tanulo kulcsot
- ☐ Lekérdezzük a kulcs típusát
- ☐ Lekérdezzük a kulcs mezőit
- ☐ Töröljük a tanulo kulcs neptun mezőjét

Utasítások III. – List

LSET kulcs index érték

Beállítja a lista adott indexű elemének értékét. Az index 0-ról indul. Speciálisan a -1 az utolsó elemre, -2 az utolsó előttiire utal stb.

LPUSH kulcs elem [elem]

Elem(ek)et szúr be a lista elejére

RPUSH kulcs elem [elem]

Elem(ek)et szúr be a lista végére

LPOP kulcs [n]

Törli és visszaadja a lista első (n) elemét

RPOP kulcs [n]

Törli és visszaadja a lista utolsó (n) elemét

LINSERT kulcs BEFORE | AFTER elem újelem

Új elemet szúr be a lista adott eleme elé vagy után

LRANGE kulcs start stop

Visszaadja a lista elemeit start indextől a stop-ig

Utasítások III. - Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379>
127.0.0.1:6379> rpush napok hetfo kedd
(integer) 2
127.0.0.1:6379> linsert napok after kedd szerda
(integer) 3
127.0.0.1:6379> lrange napok 0 -1
1) "hetfo"
2) "kedd"
3) "szerda"
127.0.0.1:6379> lpop napok
"hetfo"
127.0.0.1:6379> lrange napok 0 -1
1) "kedd"
2) "szerda"
127.0.0.1:6379>
```

- ☐ Létrehozunk egy napok listát, tartalma: hetfo, kedd
- ☐ Beszúrjuk a szerdát a végére
- ☐ Megjelenítjük a lista elemeit
- ☐ Töröljük az első napot
- ☐ Megjelenítjük az így kapott lista minden elemét

Utasítások IV. – Set

SADD kulcs elem [elem]

Elem(ek)et ad hozzá a halmazhoz. Ha a halmaz még nem létezik, akkor létrehozza. Ha az elem már létezik, akkor nem kerül be még egyszer a halmazba

SCARD kulcs

Visszaadja a halmaz elemszámát

SPOP kulcs [n]

Töröl és visszaad (n) elemet a halmazból.

SISMEMBER kulcs érték

Visszaadja, hogy az adott érték benne van-e a halmazban

SMEMBERS kulcs

Listázza a halmaz összes elemét

SDIFF | SUNION | SINTER kulcs1 kulcs2

Listázza a halmazok különbségét, unióját, metszetét

SORT kulcs [BY minta] [LIMIT interval]
[ASC | DESC] [ALPHA]

Rendezi a list, set és ordered set kulcs elemeit

Utasítások IV. - Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379> sadd evszakok tavasz nyar osz tel
(integer) 4
127.0.0.1:6379> type evszakok
set
127.0.0.1:6379> smembers evszakok
1) "tel"
2) "osz"
3) "tavasz"
4) "nyar"
127.0.0.1:6379> scard evszakok
(integer) 4
127.0.0.1:6379> sadd kedvencevszak nyar
(integer) 1
127.0.0.1:6379> sdiff evszakok kedvencevszak
1) "tavasz"
2) "osz"
3) "tel"
127.0.0.1:6379> sort evszakok alpha
1) "nyar"
2) "osz"
3) "tavasz"
4) "tel"
```

- ☐ Létrehozunk egy halmazt evszakok neven, majd lekérdezzük a típusát
- ☐ Listázzuk a halmaz tartalmát
- ☐ Megjelenítjük a halmaz elemszámát
- ☐ Létrehozunk egy új halmazt kedvencevszak neven
- ☐ Képezzük a két halmaz különbségét
- ☐ Rendezve jelenítjük meg az evszakok halmaz elemeit

Utasítások V. – Sorted set

ZADD kulcs [opciók] pontszám elem
[pontszám elem]

Új elem(ek)et ad a sorted set-hez. Ha a set nem létezik, akkor létrehozza. A létező elemek felülíródnak. Opciók: XX – nem ad hozzá új elemet, csak update, LT/GT – csak akkor módosítja a meglévőt, ha az új elem kisebb/nagyobb. A rendezés az adott pontszámok (score) alapján történik

ZCARD kulcs

Visszaadja a sorted set elemszámát

ZCOUNT kulcs min max

Visszaadja a min és max közötti elemek számát

ZINCRBY kulcs n elem

A sorted set adott elemének értékét n-nel növeli

ZINTER | ZUNION | ZDIFF n kulcsok

Visszadja az adott számú halmaz különbségét, unióját, metszetét

ZRANGE kulcs min max [opciók]

Visszadja a sorted set adott indextartományba eső elemeit

Utasítások V. - Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> zadd nevsor 1 laci 2 nora 3 zita
(integer) 3
127.0.0.1:6379> type nevsor
zset
127.0.0.1:6379> zcard nevsor
(integer) 3
127.0.0.1:6379> zcount nevsor 2 3
(integer) 2
127.0.0.1:6379> zrange nevsor 0 -1 withscores
1) "laci"
2) "1"
3) "nora"
4) "2"
5) "zita"
6) "3"
127.0.0.1:6379> zincrby nevsor 5 nora
"7"
127.0.0.1:6379> zrange nevsor 0 -1
1) "laci"
2) "zita"
3) "nora"
127.0.0.1:6379>
```

- ☐ Létrehozunk egy sorted set-et nevsor néven (laci, nora, zita)
- ☐ Lekérdezzük a nevsor típusát, elemszámát
- ☐ Lekérdezzük, hogy hány elem van, amelynek pontszáma 2 és 3 közötti
- ☐ Lekérdezzük a nevsor minden pontszámát és elemét
- ☐ Növeljük zita pontszámát 5-tel, utána ismét listázzuk a nevsor elemeit
➔ a sorrend megváltozik

Lekérdezések

SELECT * FROM tábla – minden táblára

SELECT * FROM tábla WHERE ... LIKE ...

SELECT * FROM tábla WHERE id=érték

SELECT SUM(), AVG(), MIN(), MAX()
FROM tábla

SELECT FROM ... GROUP BY

SCAN 0 – Az adatbázisban lévő összes kulcsot megadja

SCAN | SSCAN | HSCAN | ZSCAN iterátor
[MATCH minta] [COUNT n] [TYPE típus]

HGETALL kulcs

Nincs beépített lehetőség – csak program segítségével (ciklusokkal) oldható meg

Nincs beépített lehetőség – csak program segítségével (ciklusokkal) oldható meg

Lekérdezések- Példák

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379>
127.0.0.1:6379> scan 0
1) "7"
2) 1) "mainap"
   2) "napok"
   3) "evszakok"
   4) "mykey"
   5) "a"
   6) "hello"
   7) "tegnap"
   8) "kedvenevszak"
   9) "nevsor"
  10) "nev"
127.0.0.1:6379> zscan nevsor 0 match *a*
1) "0"
2) 1) "laci"
   2) "1"
   3) "zita"
   4) "3"
   5) "nora"
   6) "7"
127.0.0.1:6379>
```

- ☐ Lekérdezzük az aktuális adatbázisban tárolt összes kulcsot
- ☐ Listázzuk a nevsor sorted set azon elemeit és pontszámait, ahol az elem nevében van a betű

Táblák megfelelője Redis-ben

Table
Products

| ID | Name | Description | Price |
|-------|------|---------------------|-------|
| 10001 | AAAA | Description of AAAA | 100 |
| 10002 | BBBB | Description of BBBB | 200 |
| 10003 | CCCC | Description of CCCC | 200 |
| . | . | . | . |
| . | . | . | . |
| 10999 | XXYZ | Description of XXYZ | 500 |



| Sorted Set | | Sorted Set | |
|--------------|---------------|---------------|---------------|
| product_list | | product_price | |
| 10001 | product:10001 | 100 | product:10001 |
| 10002 | product:10002 | 200 | product:10002 |
| 10003 | product:10003 | 200 | product:10003 |
| . | . | . | . |
| . | . | . | . |
| 10999 | product:10999 | 500 | product:10999 |

Aktiválja a W

[How to get SQL-like Experience with Redis? | Redis Labs](#)

- ❑ A tábla oszlopainak a sorted set adattípus feleltethető meg, a tábla sorainak pedig a hash típus.
- ❑ Az elsődleges kulcsot a kulcs megfelelő elnevezésével lehet megvalósítani, pl. táblanév:elsődleges_kulcs formában

Tábla létrehozása - Példa

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379> hmset product:10001 name AAAA description "Description of AAAA"
price 100
OK
127.0.0.1:6379> hmset product:10002 name BBBB description "Description of BBBB "
price 200
OK
127.0.0.1:6379> scan 0 match product*
1) "2"
2) 1) "product:10002"
   2) "product:10002:name"
   3) "product:10001"
   4) "product:10001:description"
127.0.0.1:6379> set product:10001:name AAAA
OK
127.0.0.1:6379> set product:10001:description "Description of AAAA"
127.0.0.1:6379> set product:10001:price 100
OK
127.0.0.1:6379> set product:10002:name BBBB
OK
127.0.0.1:6379> set product:10002:description "Description of BBBB"
OK
127.0.0.1:6379> set product:10002:price 200
OK
127.0.0.1:6379> zadd product_price 100 product:10001 200 product:10002
(integer) 2
```

- ☐ Létrehozzuk a products tábla első két sorának megfelelő adatszerkezetet (soronként) name, description és price mezőkkel
- ☐ Lekérdezzük a product karaktersorozattal kezdődő kulcsokat
- ☐ Létrehozzuk a tábla első két sorának megfelelő adatszerkezetet (mezőnként)
- ☐ Létrehozzuk a tábla price oszlopának megfelelő sorted set-et

Kapcsolat létrehozása Redis-ben

| employee_id | first_name | last_name | address |
|-------------|------------|-----------|----------|
| 1 | John | Doe | New York |
| 2 | Benjamin | Button | Chicago |
| 3 | Mycroft | Holmes | London |

| payment_id | employee_id | amount | date |
|------------|-------------|--------|------------|
| 1 | 1 | 50,000 | 01/12/2017 |
| 2 | 1 | 20,000 | 01/13/2017 |
| 3 | 2 | 75,000 | 01/14/2017 |
| 4 | 3 | 40,000 | 01/15/2017 |
| 5 | 3 | 20,000 | 01/17/2017 |
| 6 | 3 | 25,000 | 01/18/2017 |

[From RDBMS to Key-Value Store: Data Modeling Techniques | by Wishmitha S. Mendis | Medium](#)






- ❑ A kapcsolatot a kulcs megfelelő elnevezésével lehet létrehozni
- ❑ Pl: payment:2:1, ahol 2 – elsődleges kulcs, 1 – idegen kulcs

Kapcsolat létrehozása- Példa (Előző dia alapján)

```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379>
127.0.0.1:6379> set employee:1:first_name John
OK
127.0.0.1:6379> set employee:1:last_name Doe
OK
127.0.0.1:6379> set employee:1:address "New York"
OK
127.0.0.1:6379> set payment:1:1:amount 50000
OK
127.0.0.1:6379> set payment:1:1:date "01/12/2017"
OK
127.0.0.1:6379> set payment:2:1:amount 20000
OK
127.0.0.1:6379> set payment:2:1:date "01/13/2017"
OK
127.0.0.1:6379>
```

- ☐ Létrehozzuk (mezőnként) az employee tábla első két sorának megfelelő kulcs-érték párokat
- ☐ Létrehozzuk (mezőnként) a payment tábla első két sorának megfelelő kulcs-érték párokat

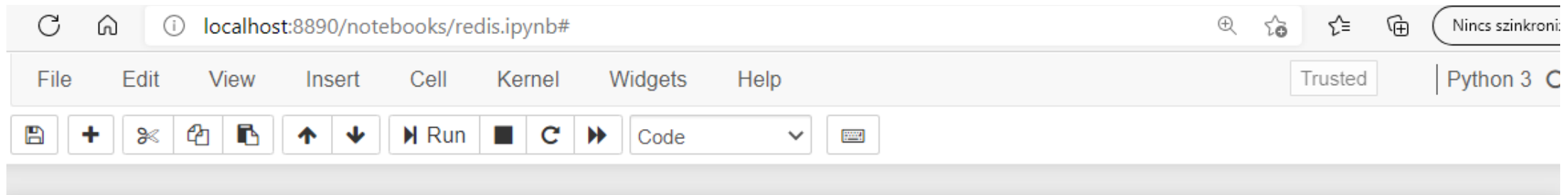
Redis – kiegészítő komponensek - példák

| | | | | |
|--------------|---|--|--|---|
| Redisearch |  | Full-Text search over Redis | dvirsky RedisLabs | Redis Source Available License |
| neural-redis |  | Online trainable neural networks as Redis data types. | antirez | BSD |
| RedisGraph |  | A graph database with a Cypher-based querying language using sparse adjacency matrices | swilly22 RedisLabs | Redis Source Available License |
| redisSQL |  | A redis module that provides a full SQL capabilities embedding SQLite | siscia RedBeardLab | AGPL-3.0 |
| RedisJSON |  | A JSON data type for Redis | itamarhaber RedisLabs | Redis Source Available License |

→ Keresési, aggregálási lehetőségek

→ SQL lekérdezési lehetőség

Redis elérés Python-ból



```
In [ ]: !pip install redis
```

```
In [ ]: import redis
```

```
In [ ]: #a host felhős rendszerénél az endpoint értéke, on-premise esetben localhost  
#a port értéke on-premise rendszerénél 6379  
#a jelszó on-premise rendszerénél - ha nem állítottuk be - nem kell  
  
r = redis.Redis(host='', port= , password='')  
r.set('hello', 'world')  
print(r.get('hello'))
```


Feladat I.

Csatlakozzon a Redis Cloud-hoz ([Redis Labs | The Home of Redis](#))!

- a. Kattintson a korábban létrehozott adatbázis nevére, majd válassza ki a Configuration fület!
- b. Adja meg válaszként a következőket: Endpoint, Port, Default password

Feladat II.

Csatlakozzon a Redis Cloud adatbázisához a következő (parancssorba beírt) utasítás segítségével:

redis-cli -h endpoint -p port -a default_password, ahol a megfelelő helyekre az előző feladatban kiolvasott endpoint, port és default password értékeit kell behelyettesíteni

- a. A csatlakozás után adja ki az `info server` parancsot
- b. A parancs futtatásának eredményeként kapott sorokat másolja be a válaszhoz

Feladat III.

A redis-cli-ben hozzon létre három új kulcs-érték párt: ev 2021, hónap 4, nap 26

a. A kulcsok lejáratí ideje 10 másodperc legyen

b. A kulcsok létrehozásához szükséges utasításokat adja meg válaszként!

Feladat IV.

A redis-cli-ben hozzon létre új lista típusú kulcsot tantargyak néven, amelyek értékei a következők legyenek: programozas alapjai, matematika, adatbaziskezeles

- a. A lista végére szúrjon be még egy tantárgyat: operacios rendszerek
- b. A listát létrehozó és a lista végére beszűrő parancsokat (két parancs) adja meg válaszként!

Feladat V.

A redis-cli-ben hozzunk létre új sorted set-et személyek néven, ahol az egyes pontszámok az életkorok legyenek: Andras 15 éves, Peter 20 éves, Juli 18 éves

- a. Kérdezzük le a 16-18 pontszámok közötti személyek számát!
- b. A sorted set-et létrehozó és a lekérdező utasításokat (két utasítás) adjuk meg válaszként!

Feladat VI.

A redis-cli-ben a korábban létrehozott személyek sorted set-ben növeljük Juli életkorát 10 évvel, majd ismét listázzuk a személyek sorted set elemeit!

a. A szükséges utasításokat (két utasítás) adjuk meg válaszként!

Feladat VII.

Hozza létre mezőnként és soronként külön kulcsok segítségével a dolgozo tábla következő rekordjait:

| nev | munkakor | kod |
|------------|----------|-----|
| Nagy Eva | titkarno | 1 |
| Kiss Ilona | konyvelo | 2 |

a. A szükséges utasításokat tartalmazó képernyőrészt adja meg válaszként kép formájában!

Feladat VIII.

A redis-cli-ben hozzuk létre a projekt táblának megfelelő adatszerkezetet soronként és mezőnként külön kulcsokkal! A kulcsok megadásánál ügyeljünk a projekt és a dolgozo tábla közötti kapcsolatra!

| projektkod | projektnev | dolgozokod |
|------------|------------|------------|
| 1 | EURO-33 | 1 |
| 3 | TRANS-22 | 2 |

a. A szükséges utasításokat tartalmazó képernyőrészt adjuk meg válaszként kép formájában!

Feladat IX.

A redis-cli-ben hozzunk létre új két új halmaz típusú kulcsot:

numbers1: 10, 20, 30, 40, 50, 60 és numbers2: 15, 30, 45, 60

a. Képezzük a halmazok metszetét!

b. A halmazokat létrehozó, és a metszetüket lekérdező utasításokat (3 utasítás) adja meg a válaszhoz!

Feladat X.

A redis-cli-ben kérdezzük le az adatbázisban lévő azon kulcsokat, amelyek nevében van a betű!

- a. Ezután adjuk meg a létező kulcsok számát a dbsize utasítás segítségével!
- b. A parancsokat és eredményüket mutató képernyőrészletet adjuk meg a válaszhoz kép formájában!



**Köszönöm
a figyelmet!**