NoSQL adatbázis-kezelők, Dokumentumtárolók, MongoDB

Jánosi-Rancz Katalin Tünde

Cloud computing

az egyetlen lehetőség, amely támogatni tudja a Big Data infrastruktúra igényeit

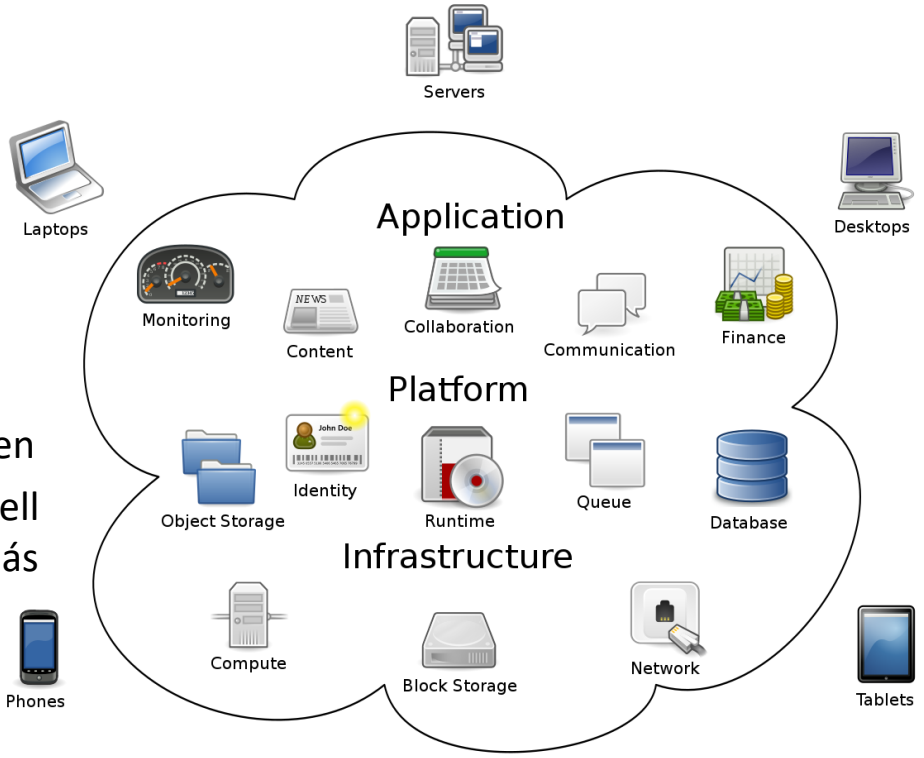
Cloud computing - felhő alapú számítástechnika

- Cloud - objektumok sokaságát távolról nézve egy egységként jelenítse meg (mint egy felhő) úgy, hogy a vele kapcsolatos dolgokat nem részletezi
- megosztott adatok és erőforrások
- szolgáltatások nyújtása elosztva a számítási eszközökön (hálózatok, szerverek, háttértárolók, alkalmazások és szolgáltatások)
- adatok tárolása és feldolgozása harmadik félnél
- tulajdonságai: nagy számítási kapacitás, alacsony költségű szolgáltatások
- nagy teljesítmény, skálázhatóság, hozzáférhetőség, rendelkezésre állás

Cloud computing- Szolgáltatási modelljei

- **SaaS**, Software as a Service
  - Adott feladatra szánt alkalmazásokat nyújt, pl.: Google Docs

- **PaaS**, Platform as a Service
  - Fejlesztői környezetet nyújt, aminek segítségével felhő-kész alkalmazásokat lehet létrehozni általában webes felületen
  - Bezár a felhőbe – azokat az eszközöket kell használni, amit a felhő nyújt, az exportálás kizárásának lehetősége is előfordul
  - pl.: OpenShift



- **IaaS**, Infrastructure as a Service

  Háttértárat és számítási kapacitást nyújt

  Gyakorlatilag a hardver szolgáltatásként való bérlése. Hardvert tudsz igényelni: disket és memóriát

  pl.: Amazon EC2, Google Compute Engine

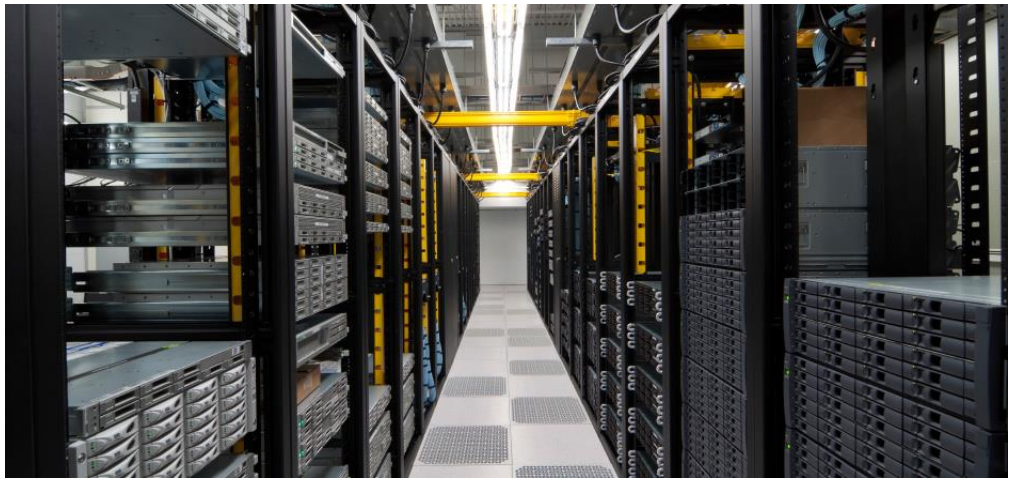
Cloud vs Own

- Igény: 128 Server (1024 CPU mag), 524 TB háttértár M hónapra

- Cloud (AWS)
  - Tárolás (S3) költségek: 0.12$ per GB
  - Számítási (EC2) költségek: 0.10$ per CPU hour
  - Teljes költség = tárolás + számítás =0.12*524*1000 + 0.10 *1024 * 24 * 30
    ~ 136K $ /M

- Saját szerver
  - Tárolás ~ 349K $ /M
  - Teljes ~ 1555 K $ / M + 7.5K $ (1 admin/ 100 gépre)

Cloud a valóságban...



Biztonságos adatközpont

- meg kell bizzál benne, feltételeit el kell fogadd és csak reméled, h az adataid backupban is megvannak vhol

A NoSQL adatbázisok világa...

Igények

- rövid válaszidők
- magas rendelkezésre állás
- nagy mennyiségű adat feldolgozása
- horizontális skálázhatóság

Skálázhatóság

- **Egy rendszer skálázható, ha**: az erőforrásokat *növelve* a rendszer teljesítménye a hozzáadott erőforrásokkal *arányosan* nő.
- A teljesítmény lehet: egyszerre feldolgozható kérések száma, feldolgozott adathalmazok vagy adatelemek mérete, késleltetés, stb.
- Két fő típusa:
  - vertikális skálázhatóság (scale up): a rendszer kiválasztott elemének bővítése új erőforrással (pl.: processzor, memória)
  - horizontális skálázhatóság (scale out): a rendszer bővítése új számítógéppel (sok olcsó gépből nagy teljesítmény érhető el)

Skálázhatóság

RDBMS NoSQL

- RDBMS – vertikális, egyetlen szerver kapacitását kell növelni (növelve a CPU-t, a RAM-ot vagy az SSD-t) az adatbázis méretezéséhez
- NoSQL – horizontális – több szervert adhatunk hozzá, hogy kiszolgálja az egyre növekvő adatbázist

CAP-tétel

- 1999-ben *Eric Brewer* publikálta

- a NoSQL rendszerek tervezését befolyásoló tétel, amely az elosztott rendszerek által nyújtott garanciákra ad formális korlátot.

- *konzisztencia* (consistency),
- *rendelkezésre állás* (availability)
- *partíció tolerancia* (partition tolerance)

- egyidejűleg legfeljebb kettőt garantálhat teljesen

Elosztott rendszerek konzisztenciája

- Az adatokat sok gépen tároljuk
- A gépek meghibásodhatnak
- Az adatokhoz több belépési ponton keresztül is hozzáférhetünk

- Biztonsági mentés helyett replikáció
- Az adatok több példányban tárolódnak
- Mi biztosítja, hogy a replikák konzisztensek maradnak? – Kell valami replikációs protokoll – Általában aszinkron
- Konzisztencia ablak – Mennyi idő után válik a rendszer konzisztenssé

Rendelkezésre állás (A)

- ha minden működő csomóponthoz érkező kérésre válaszol

- Az adatok folyton elérhetőek
  - Több belépési pont
  - Nincsen egyetlen kritikus elem sem
  - Geo-redundancia
- A válasz legyen gyors
  - Minimális késleltetés
  - Még akkor is, ha a visszaadott adat nem konzisztens (pl. Facebook idővonal)

Rendelkezésre állás (A)

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	**36.5 days**	**72 hours**	**16.8 hours**	**2.4 hours**
95%	18.25 days	36 hours	8.4 hours	1.2 hours
97%	10.96 days	21.6 hours	5.04 hours	43.2 minutes
98%	7.30 days	14.4 hours	3.36 hours	28.8 minutes
99% ("two nines")	**3.65 days**	**7.20 hours**	**1.68 hours**	**14.4 minutes**
99.5%	1.83 days	3.60 hours	50.4 minutes	7.2 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	**8.76 hours**	**43.8 minutes**	**10.1 minutes**	**1.44 minutes**
99.95%	4.38 hours	21.56 minutes	5.04 minutes	43.2 seconds
99.99% ("four nines")	**52.56 minutes**	**4.38 minutes**	**1.01 minutes**	**8.66 seconds**
99.995%	26.28 minutes	2.16 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	**5.26 minutes**	**25.9 seconds**	**6.05 seconds**	**864.3 milliseconds**
99.9999% ("six nines")	**31.5 seconds**	**2.59 seconds**	**604.8 milliseconds**	**86.4 milliseconds**
99.99999% ("seven nines")	**3.15 seconds**	**262.97 milliseconds**	**60.48 milliseconds**	**8.64 milliseconds**
99.999999% ("eight nines")	**315.569 milliseconds**	**26.297 milliseconds**	**6.048 milliseconds**	**0.864 milliseconds**
99.9999999% ("nine nines")	**31.5569 milliseconds**	**2.6297 milliseconds**	**0.6048 milliseconds**	**0.0864 milliseconds**

Partíció tolerancia (P)

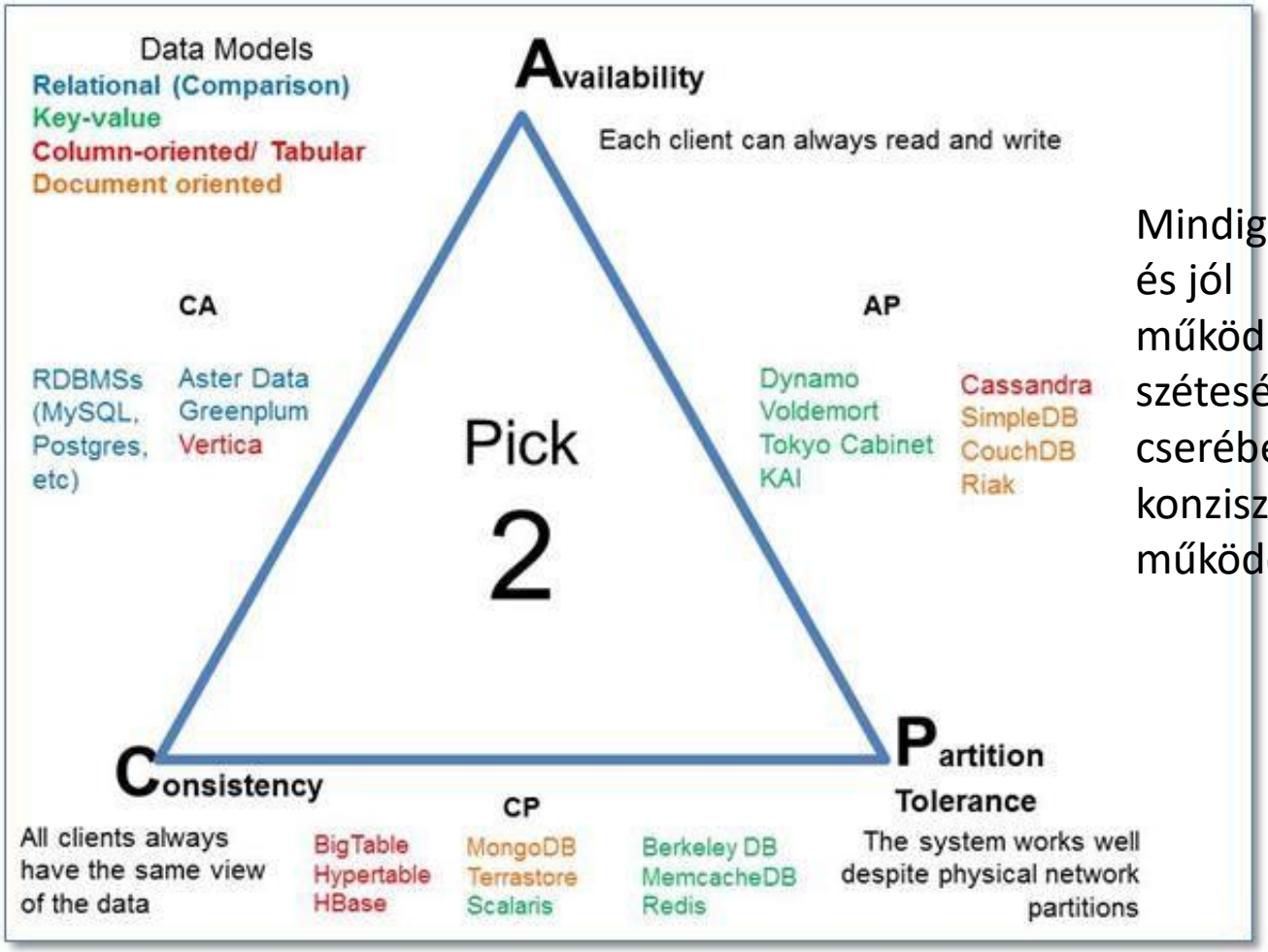
- a hálózat egyik csomópontjából a másikba küldött üzenetekből tetszőleges számú elveszhet

- a rendszer akkor is tudjon tovább működni, ha meghibásodás miatt csomópontok esnek ki, vagy hálózati hiba miatt köztük megszakad az összeköttetés

A rendszer tehát akkor partíció toleráns, ha a kérésre hálózati partíció esetén is helyes választ ad

Legfeljebb kettő

A hálózat szétesése esetén működésképtelenné válnak



Mindig elérhetőek, és jól működnek a hálózat szétesése esetén is, cserébe feladják a konzisztens működést

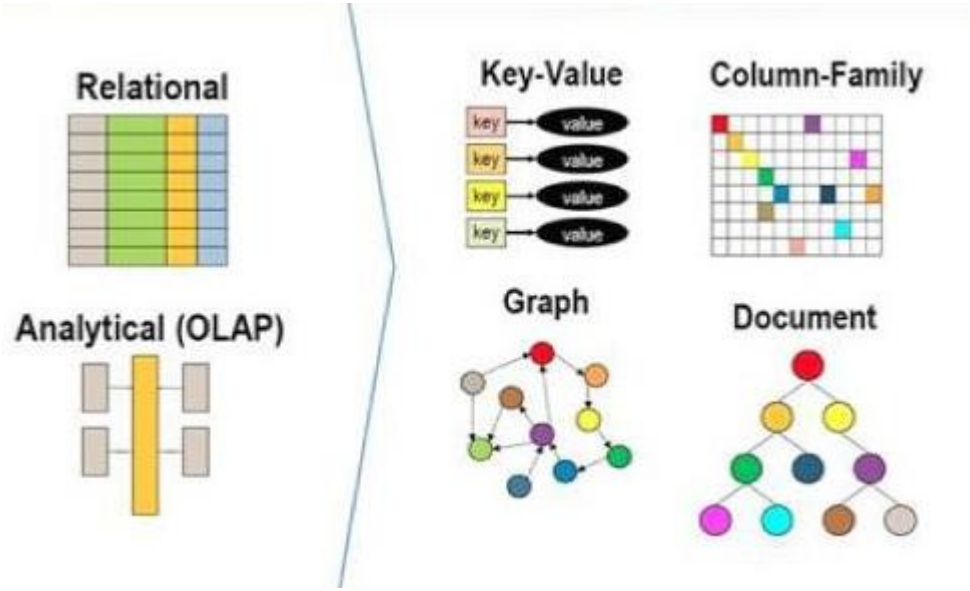
Konzisztens működés várható el ezektől a rendszerektől, és a hálózat elválása esetén is működnek, de cserébe romlik a hozzáférhetőség.

NoSQL

- Not only SQL
- adatbázis szoftverek gyűjtőneve
- ismérveik:
  - elsősorban nem táblázatokban tárolják az adatokat
  - nem SQL nyelvet használnak lekérdezésre
  - kevés műveletet támogatnak
  - nagyobb sebesség, jobb skálázhatóság
  - ACID működést nem feltétlenül tudnak

NoSQL adatbázisok fajtái

- Kulcs-érték tárolók: legegyszerűbb adatmodell
- korlátozott lekérdezések: jellemzően csak kulcs szerint
- adatok tárolása a memóriában -> nagyon gyors működés
- Pl.: Memcached, Redis, Riak, Berkeley DB



Oszlopcsaládok: kevés oszlopot és sok sort érintő elemzések hatékonyabbak

- Pl.: BigTable, HBase, Cassandra (Facebook)

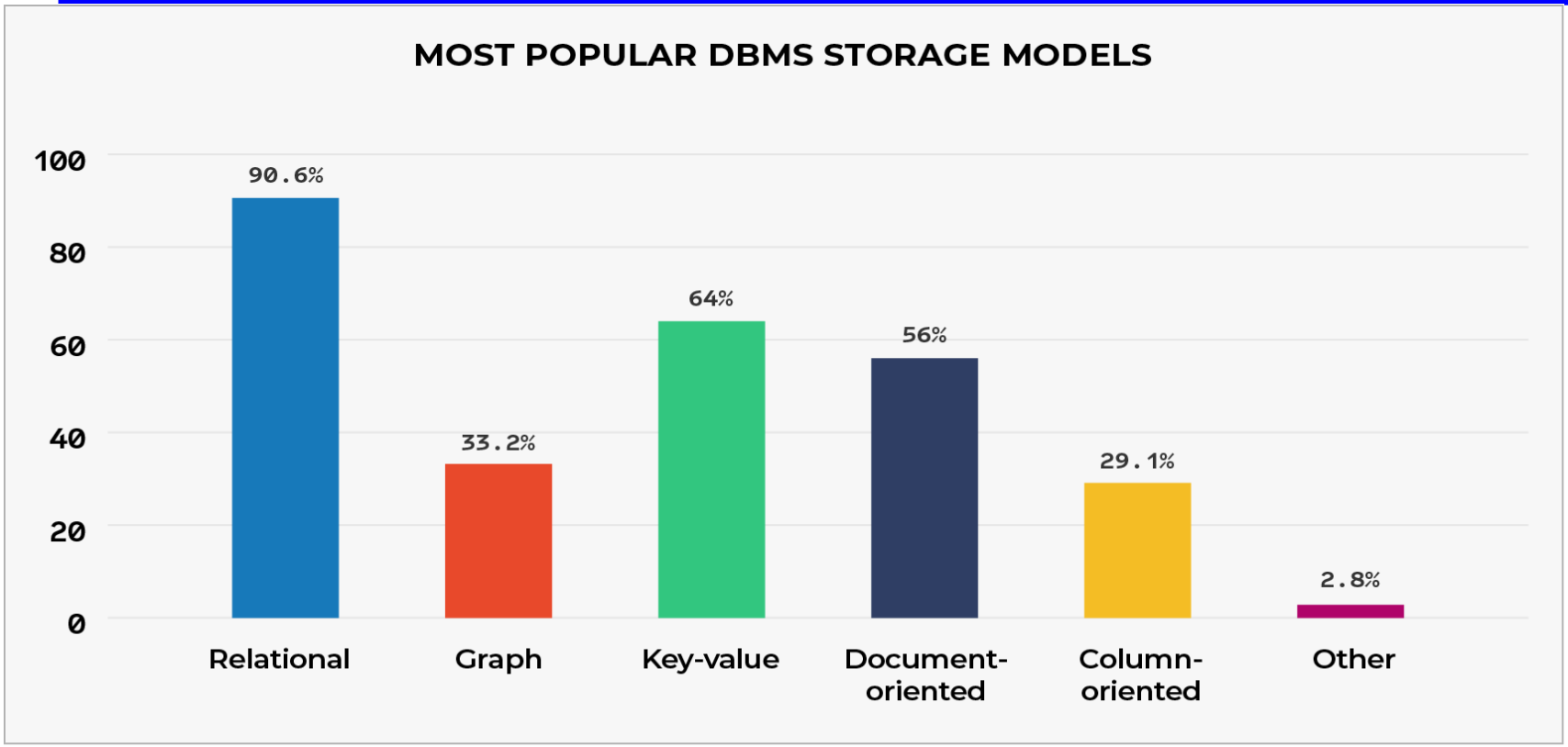
Dokumentumtárolók:

- Pl.: CouchDB, MongoDB, Elasticsearch

Gráf adatbázisok: gráfok hatékony tárolása műveletek gyors végrehajtása

- kevésbé skálázhatóak a gazdag adatmodell miatt
- Pl.: Neo4j, Infinite Graph

2021 DZone Data Persistence Trend Report

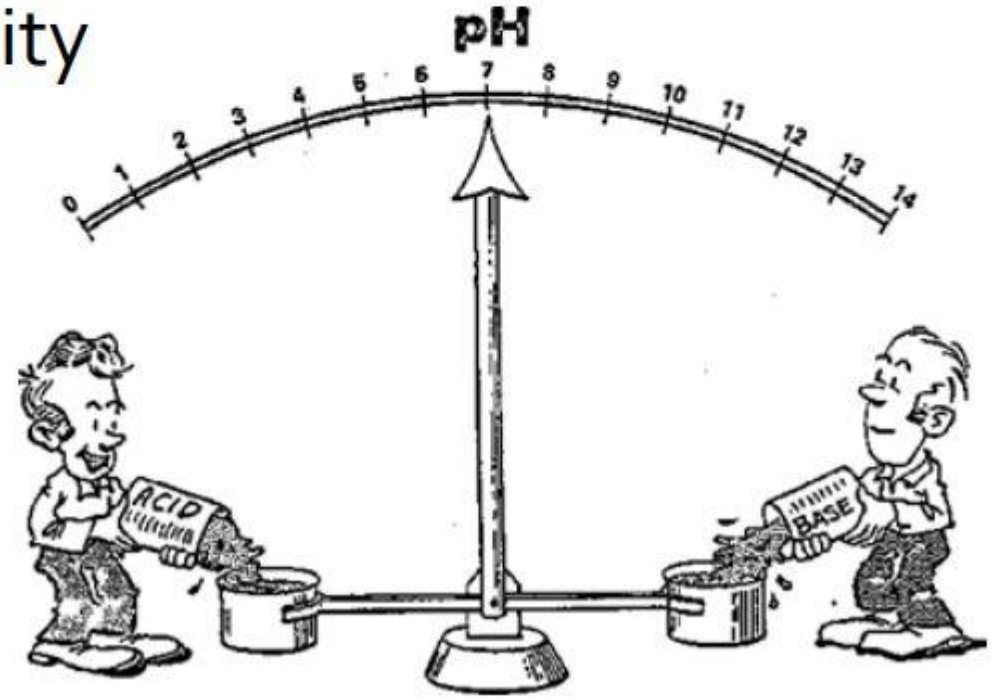


- **“relational databases are the most popular DBMS”**
- “NoSQL databases (including Graph, document-oriented, key-value, column-oriented, and others). Therefore, combined, **NoSQL databases are currently more popular than relational databases**.”
- Mindkét technológia a legjobb abban, amit csinál. A fejlesztő feladata, hogy ezeket a helyzetektől és igényektől függően jobban kihasználja.

RDBMS vs NoSQL

- Atomicity
- Consistency
- Isolation
- Durability

- Basically Available
- Soft state
- Eventually consistent



ACID tranzakciós model

- Atomicity
  - Egy tranzakció vagy teljesen lefut, vagy nem fut le
- Consistency
  - Egy tranzakció hatására az adatbázis konzisztens állapotból konzisztens állapotba jut
- Isolation
  - A tranzakciók konkurens végrehajtásának meg kell egyezni valamilyen szekvenciális végrehajtás eredményével
- Durability
  - Ha egy tranzakció sikeresre volt jelentve, akkor a változásnak tartósnak kell lennie

BASE tranzakciós modell

- BA: Basically available
  - Főleg CP rendszer esetében
  - Legalább a rendszer egy része maradjon elérhető
- Soft-state
  - A változások véges ideig tartó üzenetekkel történnek
  - A rendszer állapota akkor is változhat, ha épp nincsen input
  - Minden adatra lehet egy érvényességi idő
  - Ha ez lejárt, akkor meg kell vizsgálni, hogy konzisztens-e még
- Eventually consistent
  - Főleg AP rendszer esetében
  - A változások „Egy idő után” konzisztenssé válik
  - Konzisztencia ablak
  - Alapvetően az a kérdés, hogy ha beleírsz egy értéket az adatbázisba, majd később visszakérdezed, akkor mit fogsz visszakapni? Az a szerver, aki a READ-et kiszolgálja, nem biztos, hogy ugyanaz, mint aki a WRITE-ot átvette előtte
  - Írás után van egy időablak, amíg ha egy másik szerver szolgálja ki az olvasást, akkor lehet, hogy régi adatot lát, de idővel (eventually) mindegyik replikához el fog érni az írás művelet
  - Elfogadható-e, hogy egy ideig régi adatot kapok vissza?

{
  title: “**MongoDB**”,
  type: “**Előadás**”,
  speaker:
  {
    name: “**Jánosi-Rancz Katalin Tünde**”,
    company: “**Sapientia**”
  }
 location: “**Marosvásárahely**”,
 date: “**2021-05-3T20:15:00.000Z**”
}

Mi is az a MongoDB?

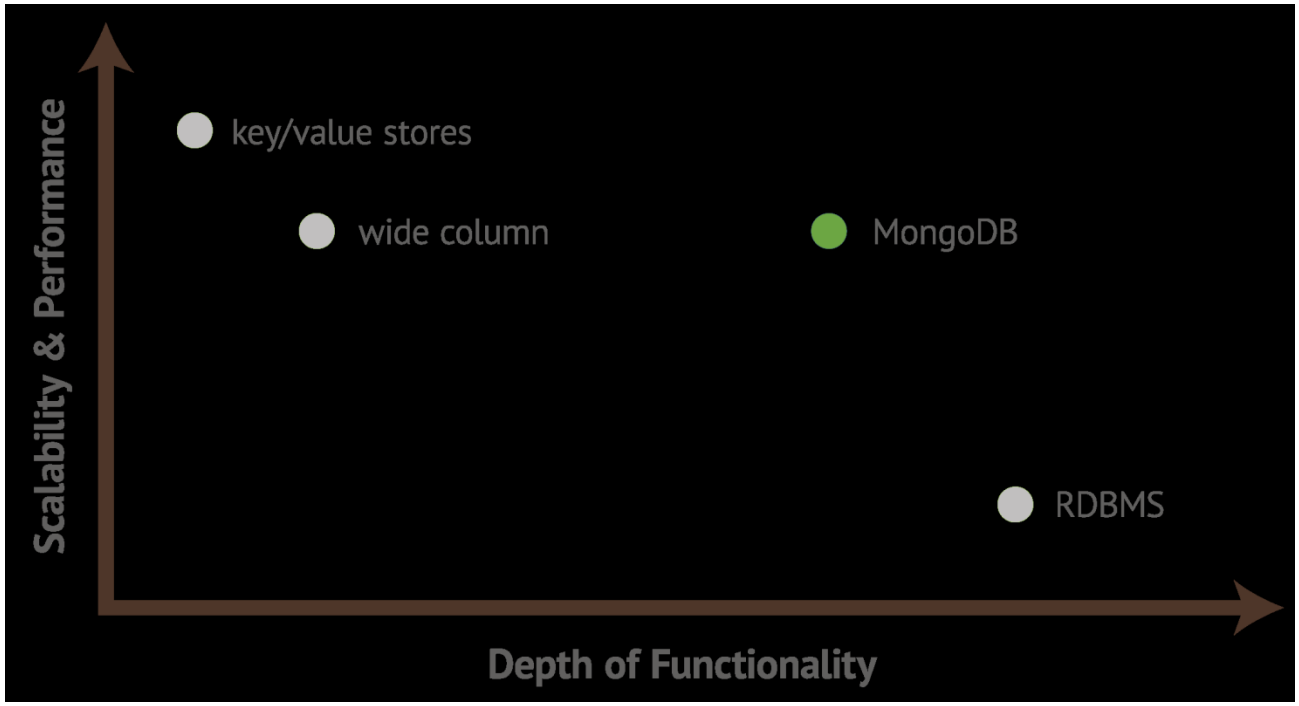
- dokumentum-orientált DBMS
- a relációs és a kulcs-érték alapú adatbázisok legjobb feature-jeit egyesíti

- kulcs-érték alapú
- JSON-stílusú adattárolás
- egyszerű és extrém gyors
- "fire-and-forget": nincs eredmény az írás (beszúrás, frissítés) sikerességéről
- adatok a memóriában és a diszken

- év adatbázisa 2014
- Verzió: 4.4.5 / 8 April 2021

Mitől gyors a MongoDB?

- Az adatokat többszörösen és elosztva tárolja



MongoDB főbb tulajdonságai

- támogatja az ACID tranzakciókat és a joinokat (referencia és beágyazás)
- Ad hoc lekérdezések
  - támogatja a keresést mező alapján, érték-tartomány alapján vagy reguláris kifejezéssel. A lekérdezések visszaadhatják a dokumentum egy meghatározott részét és tartalmazhatnak JavaScript funkciókat is.
- Indexek
- Automatikus failover recovery

MongoDB főbb tulajdonságai

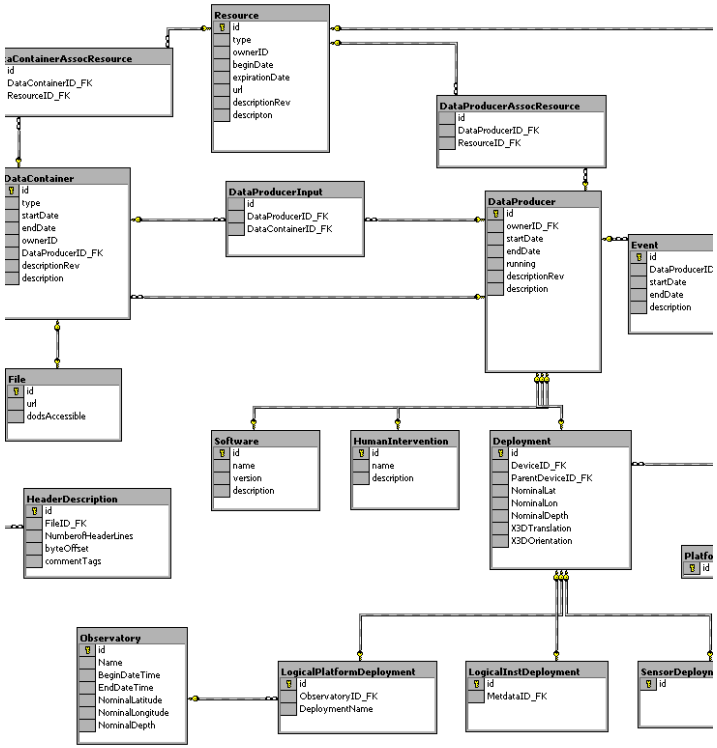
- Terheléselosztás
  - A Sharding - particionálás - az adatok több szerveren történő elosztását és skálázását teszi lehetővé. Az adatbázisok így egyre nőhet és dinamikusan bővülhet.
  - A MongoDB horizontálisan skálázható sharding használatával. A fejlesztőnek kell shard kulcsot választania, amely meghatározza, hogyan lesz elosztva gyűjtemény adathalmaza.
  - A belépő új szervereken automatikus terhelés- és adatelosztás (load balancing)

MongoDB főbb tulajdonságai

- Replikáció
  - A Replica set -többszörös tárolás- szolgál az adat redundancia, a magas rendelkezésre állás és az automatikus failover effektus biztosítására. Segítségével biztonságosan, szerver leállás nélkül üzemeltethetőek MongoDB adatbázis-rendszerek.
  - támogatja a primary-worker (secondary) replikációt.
    - Primary hajthat végre írás műveleteket
    - a worker szerverek másolják az adatokat, olvasásra biztonsági mentésre használhatók. A worker adatbázisok képesek új primary adatbázist választani, ha a primary meghibásodik.

MongoDB

- A dokumentumok tárolása nyelvfüggetlen BSON formátumban



RDMBS

```
{
  _id : ObjectId("4c4ba5e5e8aabf3"),
  employee_name: "Dunham, Justin",
  department : "Marketing",
  title : "Product Manager, Web",
  report_up: "Neray, Graham",
  pay_band: “C",
  benefits : [
                { type : "Health",
                  plan : "PPO Plus" },
                { type : "Dental",
                  plan : "Standard" }
                ]
}
```

MongoDB

Json

- strukturált dokumentumok tárolására szolgál
- hasonló dokumentum formátum pl. XML
- Adattípusok:
  - string
  - number
  - boolean
  - NULL
  - array
  - object/document
- asszociatív tömb,
  de csak string lehet a kulcs

```
{
  “name” : “jill”,
  “age” : 2,
  “voted”: true,
  “school”: null,
  “likes”: [“tennis”, “math”],
  “addr”: {
     “city”: “New York”,
     “addr”: “3rd Street”
   }
}
```

XML vs. Json

```
<phones>
        <phone type =“home”>123</phone>
        <phone type =“mobile”>456</phone>
</phones>
```

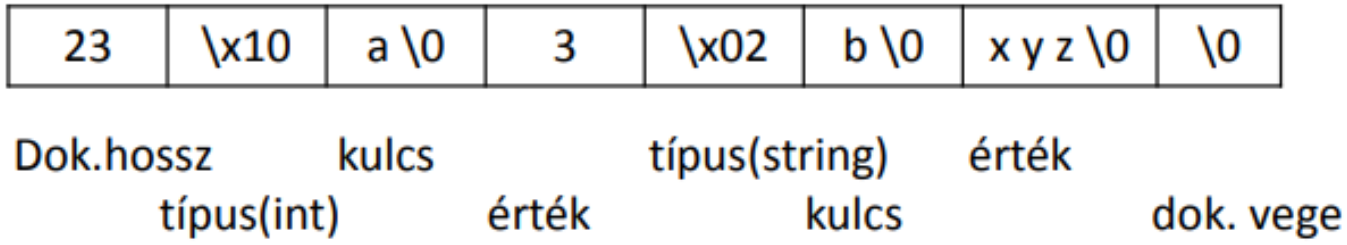
```
{
    “phones”: [
        {“phone”: 123, “type”: “home”},
        {“phone”: 456, “type”: “mobil”}
    ]
}
```

BSON

- JSON bináris reprezentációja
- Könnyen skálázható
- Mongoimport BSON műveletekhez
- JSON:

```
{
   „a”: 3,
   „b”:”xyz”
}
```

- BSON:



Schemaless ?!?!?!

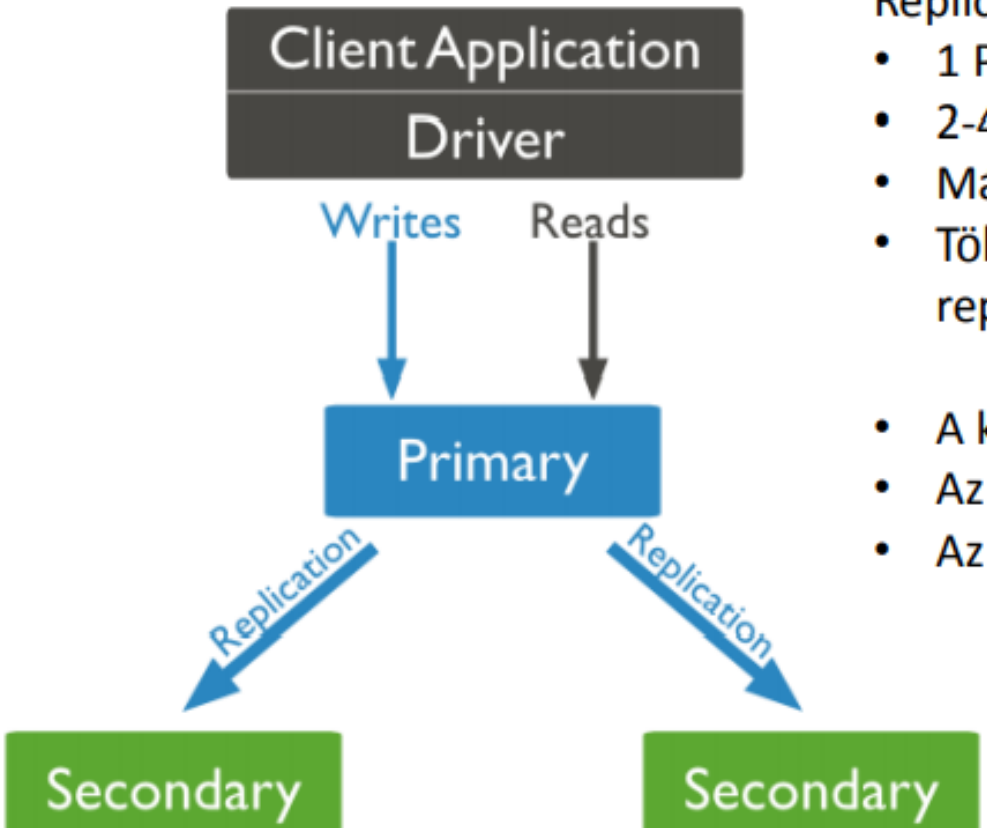
- Ez lehet egy collection:
  - {alakzat: “négyszög”, x: 3, y:4, “terület”: 12}
    {alakzat: “kör”, r: 1, “terület”: 3.14}
    {q: 456}

- Gyakorlatban konzisztens struktúra van!
  - { name : “Joe”, age : 30, interests : ‘football’ }
    { name : “Kate”, age : 25 }

MongoDB elemek

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document (JSON,BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Queries return records	Queries return a cursor

MongoDB high availability

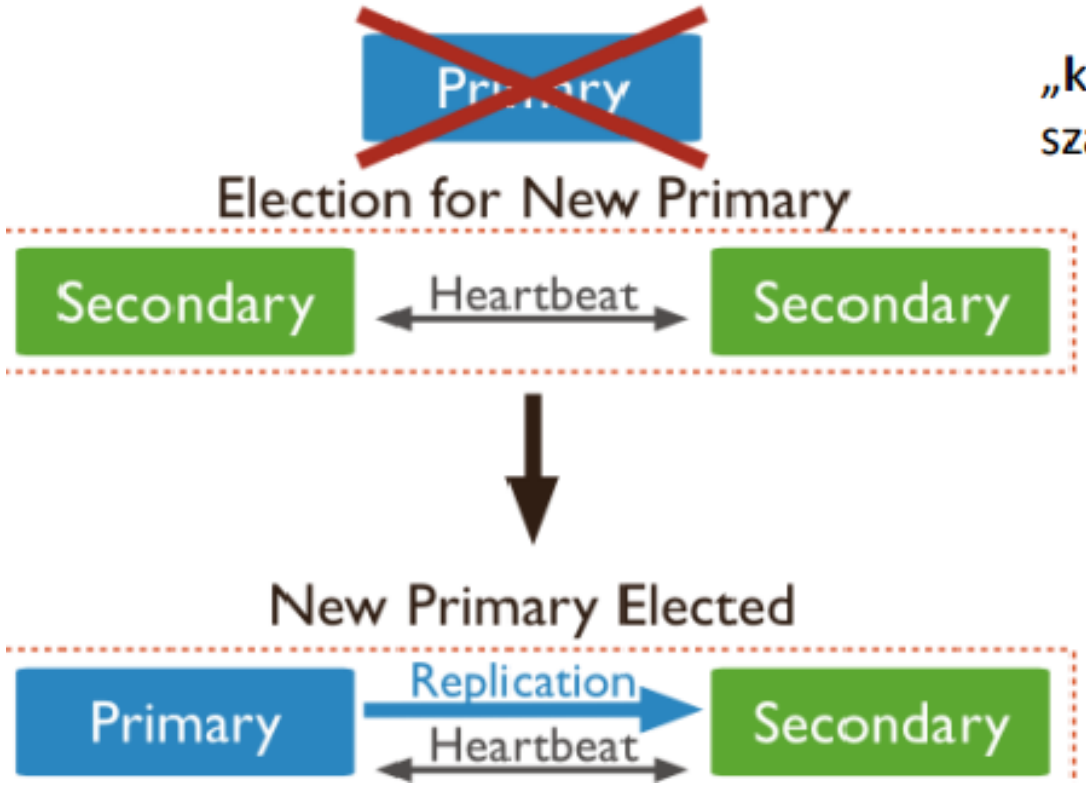


Replica Set

- 1 Primary
- 2-48 Secondaries
- Másolatok az eredeti adatról
- Több sort érintő műveletek soronként replikálódnak

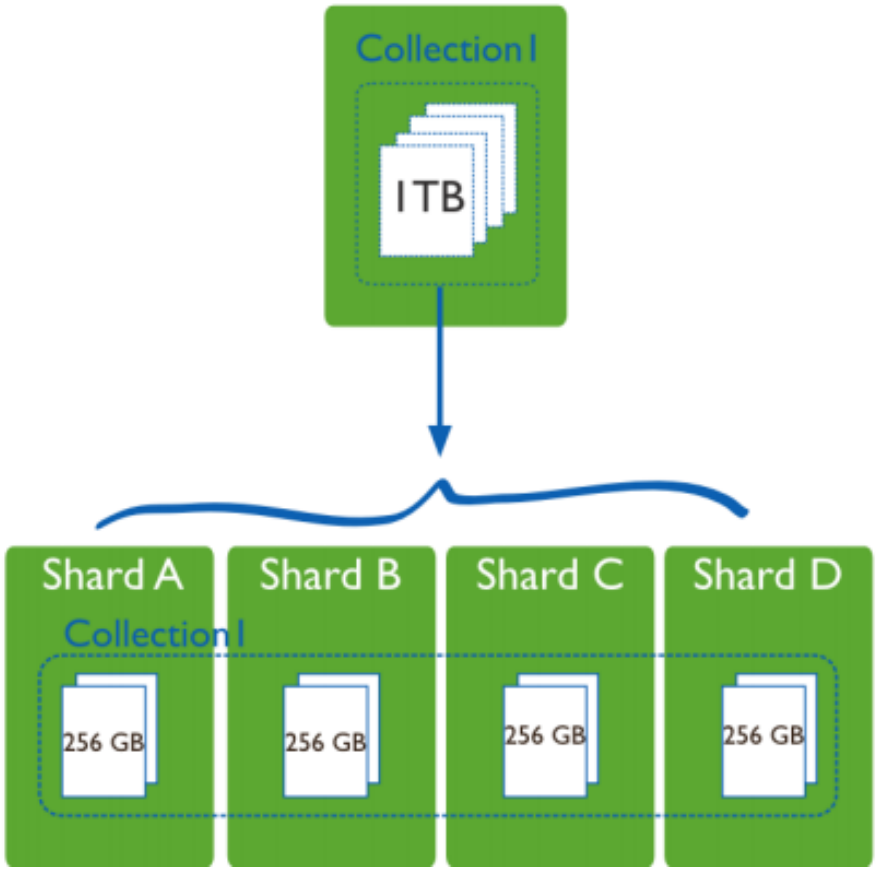
- A kliens mindig a primary-ra kapcsolódik
- Az írás mindig a primary-re megy
- Az olvasás mehet a secondary-ra

MongoDB high availability



„konszenzus”: primary hiba esetén szavazás az új primaryról

MongoDB high availability



Sharding

- a collection-ök elosztva klaszterben shard key alapján
- shard key = indexelt adattag
- shardon belül lehetnek replikák
- csökkenti az egyes szerverek terhelést
  - lekérdezések több gépen
  - kevesebb adat az egyes gépeken

MongoDB főbb komponensei

- ***mongod***: adatbázis szerver, kezeli az adat kéréseket (CRUD) és adatformákat, ill. a háttérfolyamatokat menedzseli
- ***mongo***: adatbázis kliens, interaktív JavaScript shell a MongoDB számára, lehetővé teszi az adatok lekérdezését és manipulációját, ill. szerver oldali szkriptelést
- ***mongodump***: adatbázis adatok exportálása bináris fájlba
- ***mongorestore***: bináris adatbázis adatok importálása
- ***mongoexport***: az adatbázisban tárolt adatok exportálása JSON vagy
- CSV formátumban
- ***mongoimport***: lehetővé teszi JSON, CSV vagy TSV fájlok importálását az adatbázisba

Exportálás, importálás

- JSON, CSV, TSV:
  - mongoimport --db <db_name> --collection <coll_name> --file <file_path>

  Pl.: mongoimport.exe --db test --collection restaurants --drop --file mongo_restaurant_dataset.json

  - mongoexport --db <db_name> --collection <coll_name> --out <file_name>

- Binary files:
  - mongorestore --db <db_name> --collection <coll_name> <file_name>
  - mongodump --db <db_name> --collection <coll_name>

Alapvető shell parancsok

- **show dbs** (adatbázisok kilistázása)

- **use <db_name>** (<db_name> nevezetű adatbázisba belépés vagy <db_name> nevezetű adatbázis létrehozása)

- **show collections** (collection-ök kilistázása az aktuális adatbázisban)

- **db** (az aktuális adatbázis nevének kiiratása)

- **db .<collection>.*** (* lekérdezés vagy CRUD futtatása a <collection> nevezetű collection-ön) pl.: db.users.find()

- **db.createCollection("<coll_name>")** (<coll_name> nevezetű collection létrehozása az aktuális adatbázisba)

- **db.<collection>.drop()** (<collection> törlése)

- **Help**

- **exit** (kilépés)

Adattípusok

- **double**: 23.8
- **integer**: 69
- **string**: "Hello World"
- **object**: { "name" : "Zoltán" }
- **array**: [ ]
- **Object id** (**MongoID**): ObjectId("507f1f77bcf86cd799439011")
- **boolean**: true, false
- **date**: new Date("Jun 23, 1954") vagy ISODate()
- **NULL**: null
- **undefined**
- **regular expression**: /^a.*$/i

ObjectID avagy MongoID elsődleges kulcs

- *"_id": **ObjectID("4bd9e8e17cefd644108961bb")***
- 12 bájt hosszú, 24 karakter
- automatikusan generálódik
- alapértelmezetten minden egyes dokumentumnál az ***_id*** mezőt képezi,
- automatikusan generálódik dokumentum beszúrásakor
- primary key (elsődleges kulcs)
- unique (minden egyes érték csak egyszer fordulhat elő)
- indexelt (gyors elérés)
- létrehozása: **new ObjectId()** => automatikusan generálja az értékét
- érték kinyerése (2.2-es verzió óta):

objIdValtozo.**valueOf()** => "507f1f77bcf86cd799439011"

Adatok beszúrása

- „write concern”
- Insert, update, delete utasításnál:
  - Erős ellenőrzés = lassú válasz
  - Gyenge ellenőrzés = gyors válasz
- MongoDB paraméterezhető

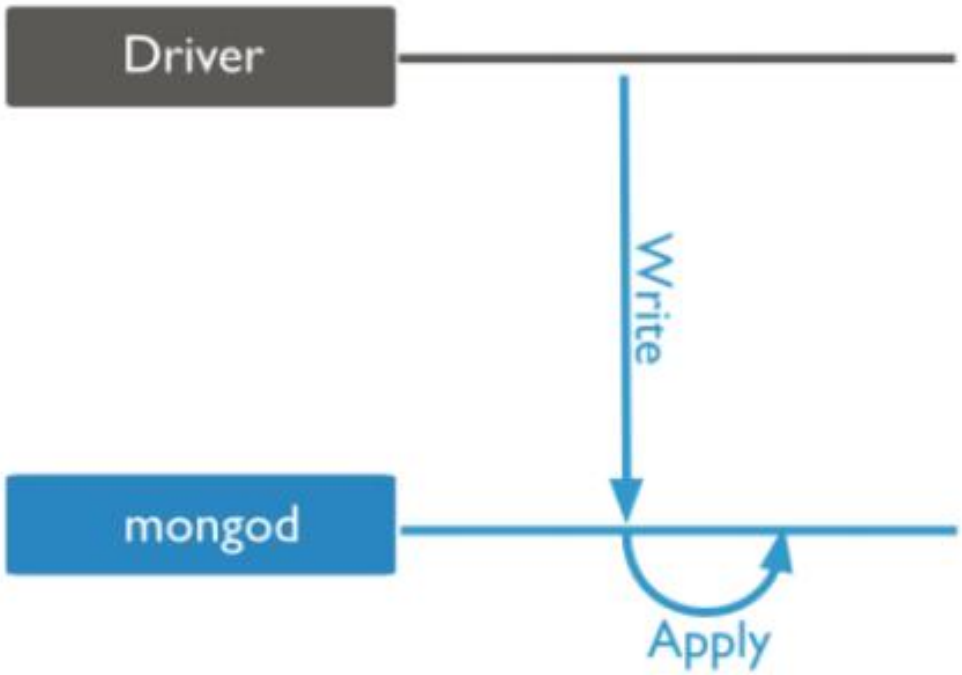
„write concern” szintjei

- w és j paraméterekkel állíthatóak be!

- Unacknowledged
- Acknowledged (w:1)
- Journaled (j:true)
- Replica Set Acknowledged (w:2)

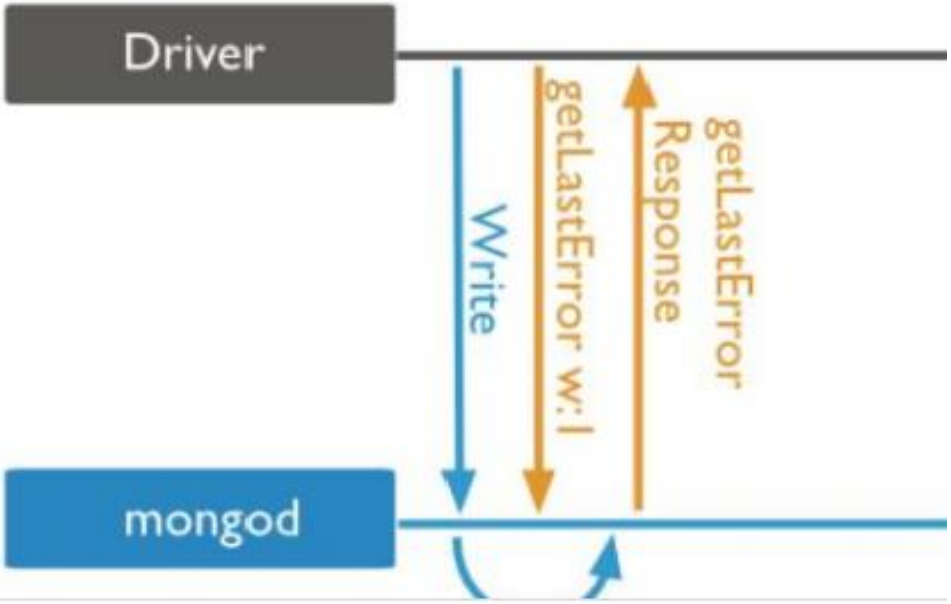
Unanknowledged

- Kliens nem kap visszajelzést az írásról
- Nagyon gyors írás
- Adatok elveszhetnek!
  - hálózati hiba
  - kulcs ütközés



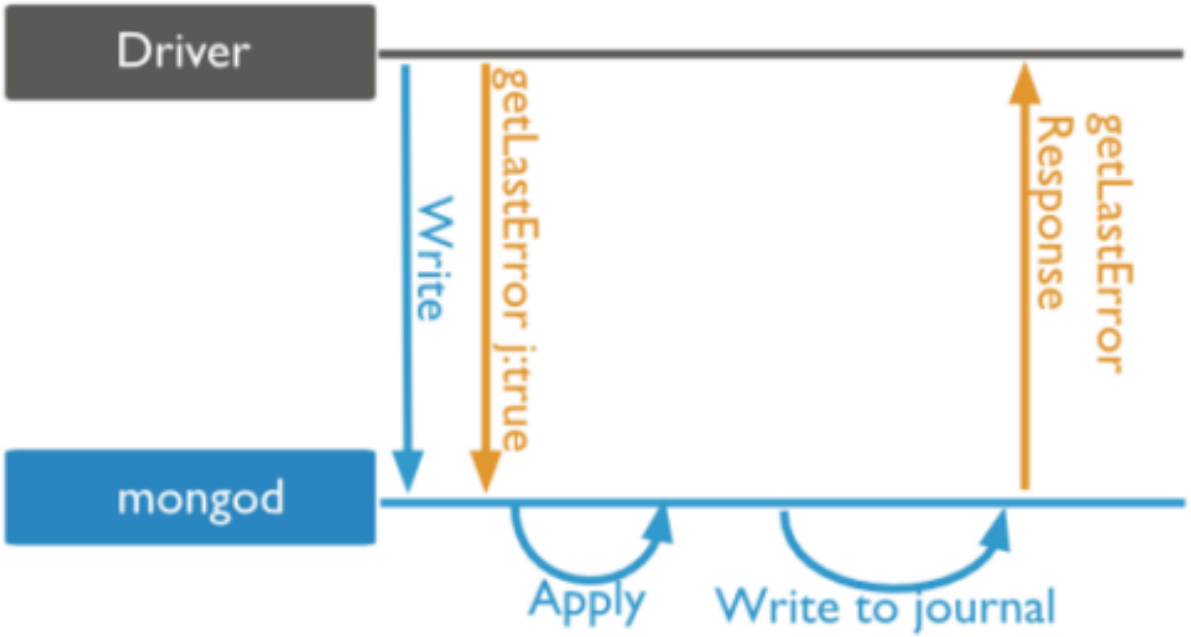
Acknowledged

- A mongod nyugtát küld
- Kliens érzékeli a hibákat (hálózati, kulcsütközés)
- Ez a default érték
- Nem garantálja hogy lemezre is kiírtuk az adatot!



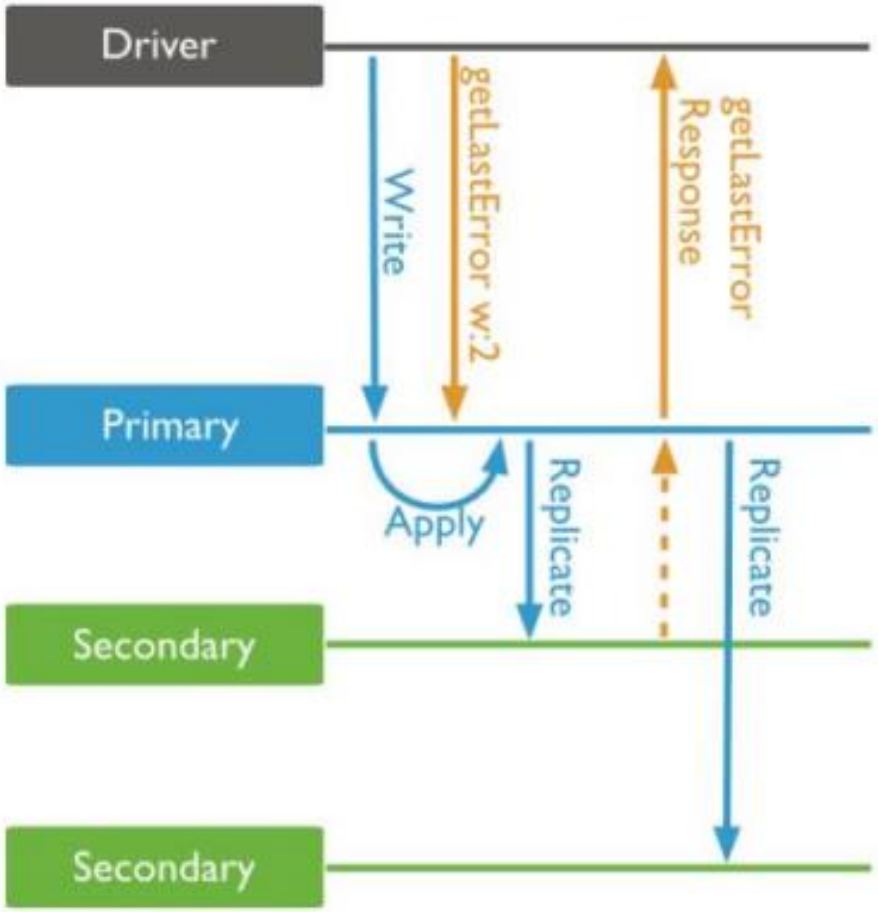
Journaled

- A mongod addig nem nyugtáz, amíg nem írja be a Journal-be. (logolás)
- Journal-t engedélyezni kell



Replica Set Acknowledged

- Beállítható hány replikának kell visszaigazolni az írást, mielőtt a kliensnek jelzi a mongod



MongoDB adattárolás

- Minden MongoDB instance tartalmaza:
  - Namespace file
  - Journal file
  - Data file
- Data file tárolja extentekben:
  - BSON dokumentumok
  - Indexek
  - MongoDB metadata adat
- Extent: logikai konténer

Extents

- Adat és index külön extent
- Egy extent egy collection-höz tartozik
- Egy collection lehet több extentben

Metrikák

- db.stats()
  - statisztikák az adatbázisunkról
  - dataSize
  - storageSize
  - fileSize
- A dataSize a dokumentumok és a lefoglalt mezők összege byte-ban
- FileSise: a teljes lefoglalt terület a lemezen

restaurant példa


```
{
        "_id" : ObjectId("56ed68032ea3567be5c25d32"),
        "address" : {
                "building" : "284",
                "coord" : [ -73.9829239, 40.6580753 ],
                "street" : "Prospect Park West",
                "zipcode" : "11215"
        },
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "grades" : [
                { "date" : ISODate("2014-11-19T00:00:00Z"), "grade" : "A", "score" : 11 },
                { "date" : ISODate("2013-11-14T00:00:00Z"), "grade" : "A", "score" : 2 },
                { "date" : ISODate("2012-12-05T00:00:00Z"), "grade" : "A", "score" : 13 },
                { "date" : ISODate("2012-05-17T00:00:00Z"), "grade" : "A", "score" : 11 }
        ],
        "name" : "The Movable Feast",
        "restaurant_id" : "40361606"
}
```

Mongo CRUD

- Create
  - db.collection.insert(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- Read
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- Update
  - db.collection.update(<query>, <update>, <options>)
- Delete
  - db.collection.remove(<query>, <justOne>)

Mongo CRUD

- adatok listázása
  - db.restaurants.find()
- hány elem van
  - db.restaurants.count()
- mely éttermek vannak Manhattan kerületben?
  - db.restaurants.find({borough:"Manhattan"})

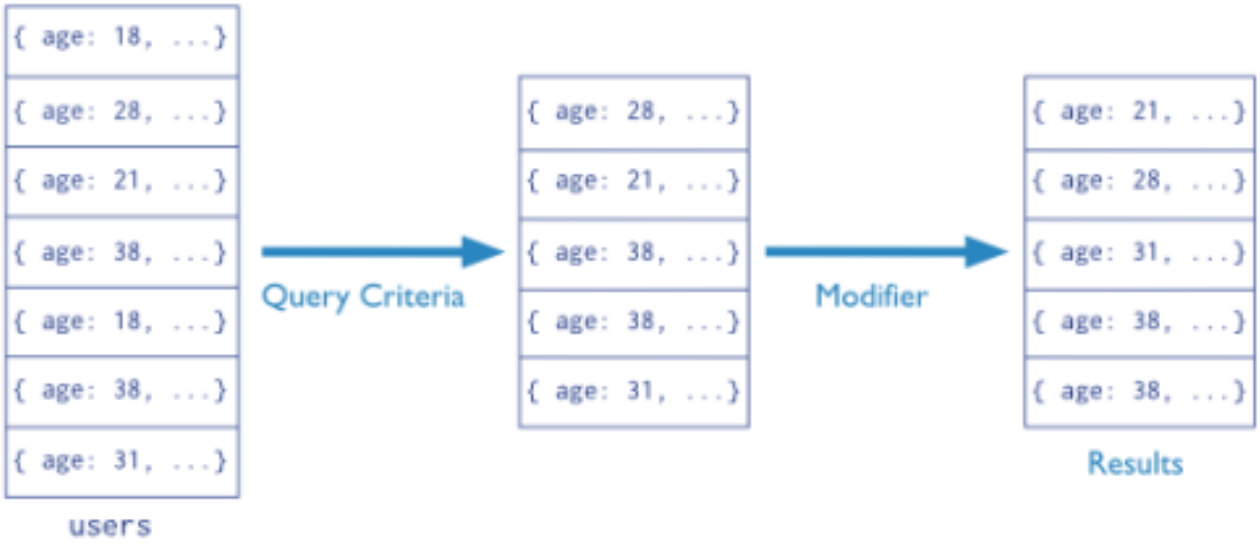
Mongo CRUD

```
db.users.find(
    { age: { $gt: 18 } },
    { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

Collection — Query Criteria — Modifier

```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```



Mongo CRUD

- Beágyazott objektumra szűrés
  - db.restaurants.find({ "address.zipcode": "10075" })
- keresés tömbben
  - db.restaurants.find({ "grades.grade": "B" })
- operátorok használata
  - db.restaurants.find({ "grades.score": { $gt: 30 } })
  - db.restaurants.find({ "grades.score": { $lt: 10 } })

Mongo CRUD

- feltételek összekapcsolása
- logikai és
  - db.restaurants.find({ "cuisine": "Italian", "address.zipcode": "10075" })
- logikai vagy
  - db.restaurants.find({ $or: [{ "cuisine": "Italian" }, { "address.zipcode": "10075" }] })
- rendezés
  - db.restaurants.find().sort({ "borough": 1, "address.zipcode": 1 })

Mongo CRUD

- update - frissíti az első „Juni” nevű objektumot
  - db.restaurants.update(
    { "name" : "Juni" },
    {
    $set: { "cuisine": "American (New)" },
    $currentDate: { "lastModified": true }
    }
)

- update – objektumon belül
  - db.restaurants.update(
    { "restaurant_id" : "41156888" },
    { $set: { "address.street": "East 31st Street" } }
)

Mongo CRUD

- update – több elem frissítése
  - db.restaurants.update(
    { "address.zipcode": "10016", cuisine: "Other" },
    {
      $set: { cuisine: "Category To Be Determined" },
      $currentDate: { "lastModified": true }
    },
    { multi: true}
)

Mongo CRUD

- törlés – összes egyező dokumentumot törli
  - db.restaurants.remove({ "borough": "Manhattan" })

- törlés – csak az elsőt
  - db.restaurants.remove({ "borough": "Queens" }, { justOne: true })


SQL
NoSQL

Mikor melyiket használjuk?

- nem fehéren fekete
- meg kell gondolni
  - hogyan néznek ki az adatok
  - hogyan fogjuk lekérdezni az adatokat
  - jövőben szükséges-e a skálázhatóság

Mikor melyiket használjuk? – nem fehéren fekete

SQL	NoSQL
komplex lekérdezések és reportok	folyamatosan új funkciókat, adattípusokat adunk hozzá, és nehéz megjósolni, hogy az alkalmazás hogyan fog növekedni az idő múlásával
magas tranzakciós alkalmazás	rugalmas sémakialakítással dolgozunk, vagy nincs előre definiált séma
Biztosítania kell az ACID betartását, vagy meg kell határoznia, hogy a tranzakciók hogyan működnek együtt az adatbázissal	Nem érdekel az adatok konzisztenciája, és a 100% -os adatintegritás nem a legfontosabb cél.
Előreláthatóan nem lesz sok változás vagy növekedés	Sok adat van, sokféle adattípus, az adatigénye csak növekszik az idő múlásával
minden adatnak konzisztensnek kell lennie, nem adhatunk teret a hibának	Gyorsan, még akkor is, ha a visszaadott adat nem konzisztens
Pénzügyek, bank, repülőfoglalás,…	Ideiglenes adatok, bevásárlókosarak, kívánságlista, szokások, munkamenetek tárolására,…

ut

Következtetés

- Az SQL és a NoSQL egyaránt magas rendelkezésre állást és automatikus replikációt kínál, de az SQL konfigurálást igényel, míg sok NoSQL adatbázis automatikusan tartalmazza ezeket a szolgáltatásokat

- RDBMS – támogatást nyújtanak NoSQL stílusú adattípusokhoz – Json (Oracle, PostgreSQL)

Kérdések

1. Mit mond ki a CAP tétel?
2. Mutasd be a BASE tranzakciós modellt!
3. Mire szolgál a JSON?
4. Adjunk meg 6 JSON adattípust!
5. Mi a BSON!
6. Mi a sharding?
7. Milyen lehetőségek vannak shardingra?
8. Mik a Write Concern szintjei?
9. Mi jellemző az Unacknowledge esetre?
10. Mi jellemző az Acknowledge esetre?
11. Mi jellemző az Journaled esetre?
12. Mi jellemző a Replica Set Acknowledged esetre?
13. Milyen adatokat ad vissza a db.stats() függvény? Mutasd be őket!
14. Adjuk meg MongoDB-ben a beszúrás, módosítás, törlés műveleteit!
15. Adjunk meg 4 aggregation framework operátort!

NewSQL

- “SQL is Back”
- új skálázható adatbázisok

- biztosítja ugyanazt a teljesítményt mint a NoSQL, de nem rugja fel az ACID elveket
  - képes relaciós/SQL
  - többnyire osztott
  - zárt forráskód

Mi volt, mi az és mi lesz az ABKR feladata?

Bittek tárolása és később elérhetővé tétele, kicsit mindig másképp

- Köszönöm a figyelmet!

- UI: Ha még mindig nem tudsz elég SQL-t, írd be a CV-dbe, hogy NoSQL!