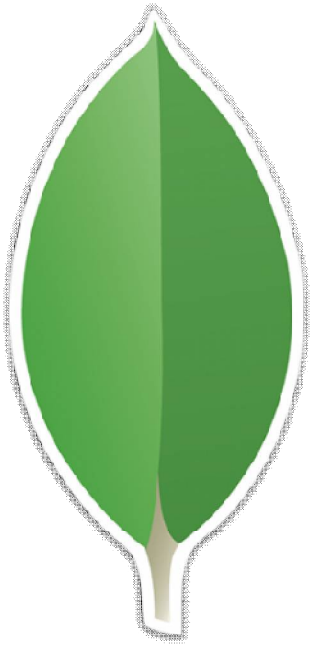


A MongoDB bemutatása

Az OE adatkörén keresztül

Sári Zoltán

Adatbázis-kezelés elmélete és gyakorlata 2013/2014. II. félév

A prezentáció témája

- **Mi** az a MongoDB ?
- **Mikor** érdemes használni ?
- **Hogyan** lehet használni ?

- **Célom**:
  - rávilágítani a tipikus **használati esetek**re
  - bemutatni a **környezet**et (architektúra, telepítés)
  - ismertetni a **használat**ot példán keresztül (CRUD)

NoSQL technológia

- **Új igények**
  - új technológiák (web 2.0, mobileszközök, clouding), big data 3V
  - rengeteg „csak” félig strukturált adat
- **Relációs modell hátrányai**
  - erős strukturális megszorítások = rugalmatlan adatmodell
  - szigorú tranzakció kezelés = elosztottság, skálázhatóság gátjai
- **Új paradigma**
  - **Cél**: nagy teljesítmény + magas rendelkezésre állás + változatos adatok rugalmas kezelése
  - **Eszközök**:
    - egyszerűsített adatmodell (kulcs-érték, dokumentum, gráf, oszlopcs.)
    - nem deklaratív lekérdezőnyelvek, nincs JOIN
    - backup helyett replikálás
    - gyors tranzakciós modell (konzisztencia alacsonyabb szinten)

Mi az a MongoDB?

- **Dokumentumorientált**
  - B(J)SON, sémafüggetlen tárolás (nem relációs)
  - javascript alapú lekérdező nyelv (függvényhívások, nem SQL)
- **Nagy teljesítményű**
  - indexelés támogatása (id automatikus, másodlagos indexek)
  - in-memory (lapkezelés), query optimizer (legjobb index)
  - analitikus feldolgozás MapReduce-val → BI, adatbányászat
- **Skálázható**
  - automatikus replikáció (magas rendelkezésre állás)
  - automatikus terhelésmegosztás (horizontális skálázás)
- **Nyílt forráskódú, platform független**
  - C++, jól dokumentált, driverek (API), könnyű integrálhatóság
- **Legnépszerűbb NoSQL adatbázis**

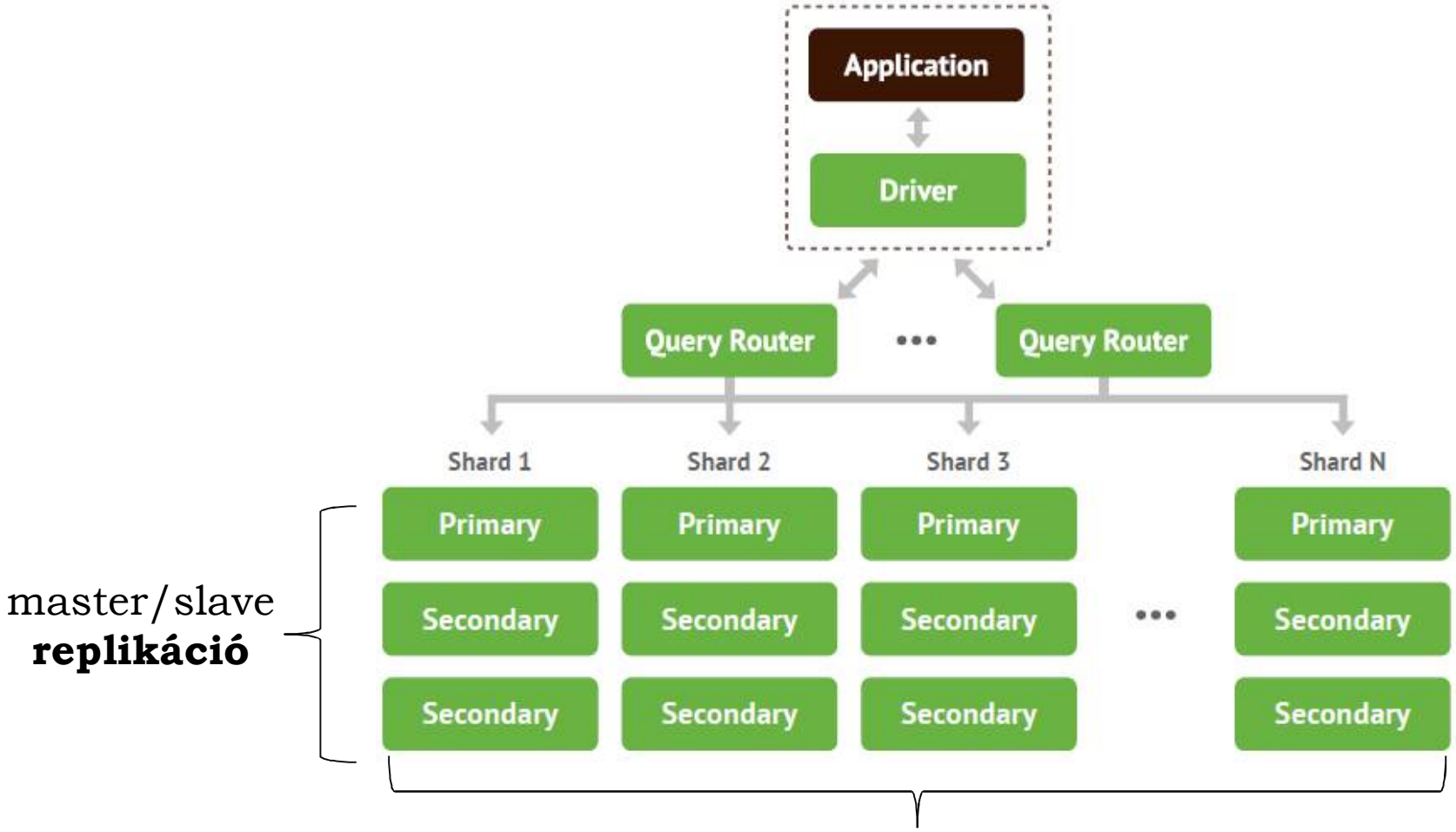
JSON (JavaScript Object Notation)

```
{                                                    //objektum
    "vezetekNev"  : "Kovács",                        //tulajdonság-érték párok
    "keresztNev"  : "János",
    "kor"         : 25,
    "cim" :
    {                                                //beágyazott objektum
       "utcaHazszam" : "Fő utca 21.",
       "varos"       : "Budapest",
       "iranyitoSzam" : "1021"
    },
    "telefonSzam":
    [                                                //tömb
       {
         "tipus"      : "otthoni",
         "szam"       : "212 555-1234"
       },
       {
         "tipus"      : "munkahelyi",
         "szam"       : "646 555-4567",
         "mellek"     : "312"                         //bővíthetőség
       }
    ]
}
```

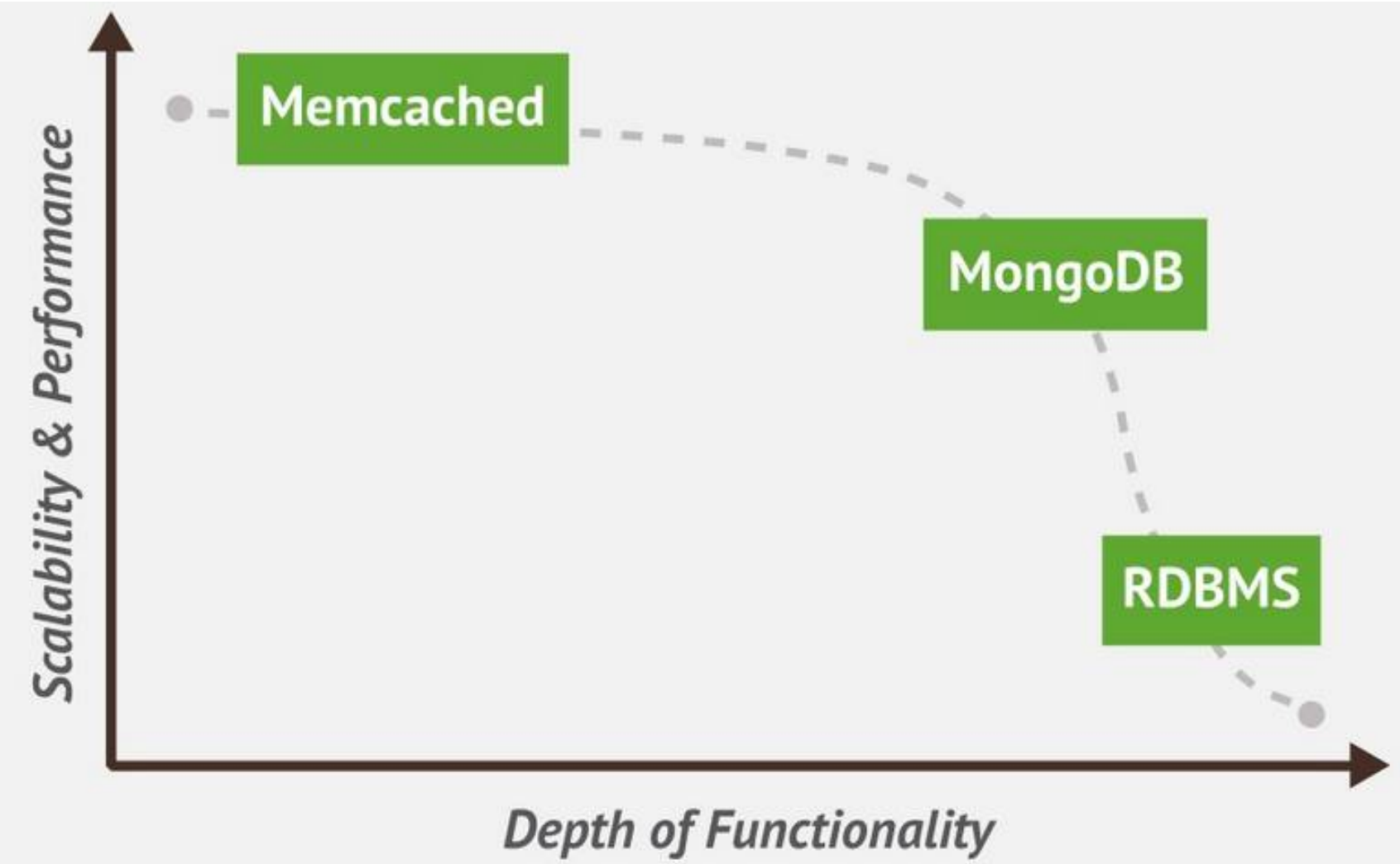
JSON

- **Jellemzők**
  - a JavaScript nyelvből alakult ki
    - adattípusok: String, Number, Boolean, Object, Array, null
    - függvényhívásokkal könnyen feldolgozható
  - **objektum** megközelítésű
  - emberek számára is olvasható–írható
- **Sémafüggetlen**
  - egy relációban számos, ritkán használt attribútum
  - JSON esetén **nem probléma 2 objektumról más-más információ tárolása**
  - JSON Schema: struktúra és az adattípusok ellenőrzése
    - de kevésbé használják, mint az XML sémákat

Skálázás és elosztott működés



MongoDB vs. RDBMS



RDBMS vs. MongoDB

	RDBMS	MongoDB
adatmodell	**normalizált**, szeparált táblák összekapcsolása (join)	**dokumentum** alapú, összetartozó adatok egy dokumentumban
séma	**relációs séma**, struktúra és adattípusok fix (ALTER)	**sémafüggetlen**, dinamikus
hozzáférési nyelv	**SQL** (deklaratív)	**objektum alapú script** + MapReduce
tranzakció kezelés (ACID)	**erős, de költséges** konzisztencia (konkurenciakezelés + naplózás)	**gyenge** konzisztencia, (atomi műveletek dokumentumokon, readers-writer lock, naplózás dokumentumszinten ekvivalens a tranzakciókezeléssel)

RDBMS vs. MongoDB

	RDBMS	MongoDB
rendelkezésre állás	hagyományosan **back-up**	master-slave **replikáció** késleltetett konzisztencia (időablakon belül)
skálázhatóság	**vertikális** (egy gép)	**horizontális** (több gép)
elosztott működés (CAP)	**merev** az ACID és a JOIN-ok miatt (**C**A v. **C**P, konzis.→max)	**rugalmas**, auto-sharding nincsenek komplex join-ok, könnyű párhuzamosítás (**AP**: késleltetés→ min)
felhasználási terület	„kis” adatkészlet, jól strukturáltság, sok aggregáció erős konzisztencia igénye, monolitikus architektúra **ERP**	„hatalmas” adatkészlet változatos és változó struktúra ACID hiánya tolerálható (v. az app layerben) elosztottság, skálázhatóság **Webes adatok tárolása**

Terminológia

RDBMS	MongoDB
database	database
table	**collection**
row	**document**
column	**field**
index	index
table join	**embedded document**
foreign key	**reference**
primary key	primary key
partition	**shard**

Mikor érdemes használni?

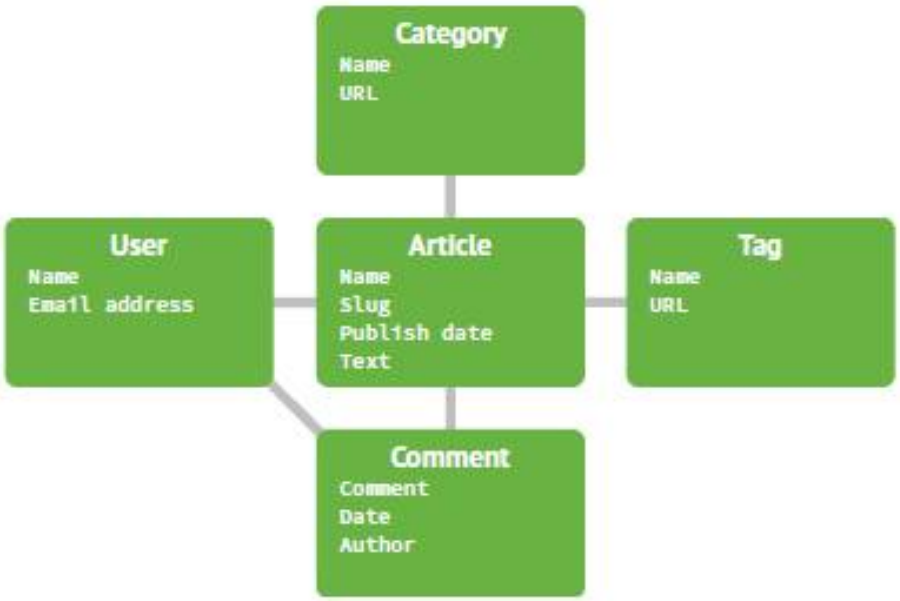
- **tartalommenedzsment és közzététel**
  - online publikációk, blogok, CMS, metaadat-menedzsment
- **közösségi infrastruktúra**
  - fórumok, comment, like-ok
- **felhasználói adatok menedzselése**
  - viselkedés loggolása
- **termék és készlet katalógusok**
  - e-kereskedelem, eltérő tulajdonság készletek
- **adat hub-ok**
  - adatintegráció, heterogén adatforrások adatai aggregációjának historikus tárolása
- **database-as-a-service**
  - rugalmas adatbázis kapacitás biztosítás
- **cache server** (SQL + NoSQL)

Hogyan lehet használni?

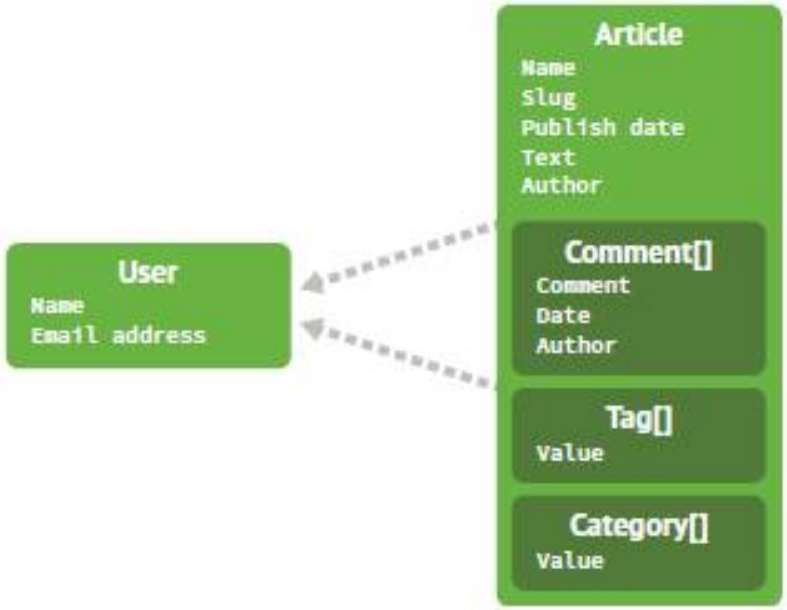
- **Adatmodellezés**
- **Telepítés**
- **Információ kinyerés**
- **Írási műveletek**
- **Adminisztráció, üzemeltetés**
  - ld. melléklet

Adatmodellezés

relációs



MongoDB



- **Lokalitás**
  - logikailag összetartozó adatok egy dokumentumban
  - általában fizikailag is egy helyen

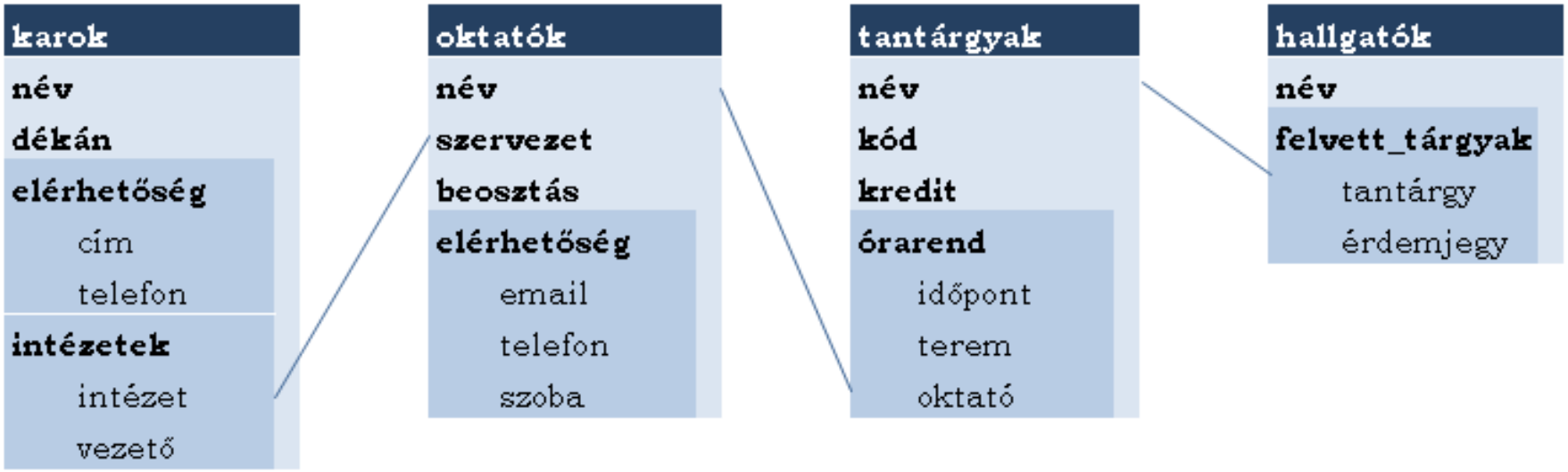
Adatmodellezés

- **Flexibilis séma**
  - **nincs DDL**: nem kell az adattárolás előtt meghatározni (RAD)
  - **bővíthetőség**: nincs erős típusosság, a dokumentumok eltérő mezőket tartalmazhatnak (SQL-ben ALTER-ek sorozata)
  - **objektum orientáltság:** egységbe zárás, közel a programozóhoz
- **Szempontok**
  - **kerülendő a relációs modell**ben gondolkodás
  - **adat TTL:** dokumentum méret növekedés (max 16MB, relokáció)
  - **teljesítmény**: lekérdezési célok, írás/olvasás aránya, indexek
  - **kapcsolatok**: JOIN-ok redukálása
  - **fa struktúra alkalmazása**: parent/child referenciák, path

Kapcsolatok

	beágyazás	**hivatkozás**
elv	**egységbe zárás**	**függetlenség**
konzisztencia	**redundáns**	**normalizált**
atomitás	**magasabb**	**alacsonyabb**
kapcsolat (UML)	**aggregáció**, **kompozíció** (rész-egész, tartalmazás)	**asszociáció** (laza kapcsolat)
manipuláció	**egyszerű** műveletek, gyors	**összetett** operáció, lassú
felhasználás	együtt jelennek meg, módosulnak beágyazott irreleváns a szülő megsemmisülése után 1:1, 1:N	egyik gyakran olvasott/ módosul, másik statikus különböző források által hivatkozott (törzsadat) 1:N, N:M

OE adatmodell



Alap folyamatok

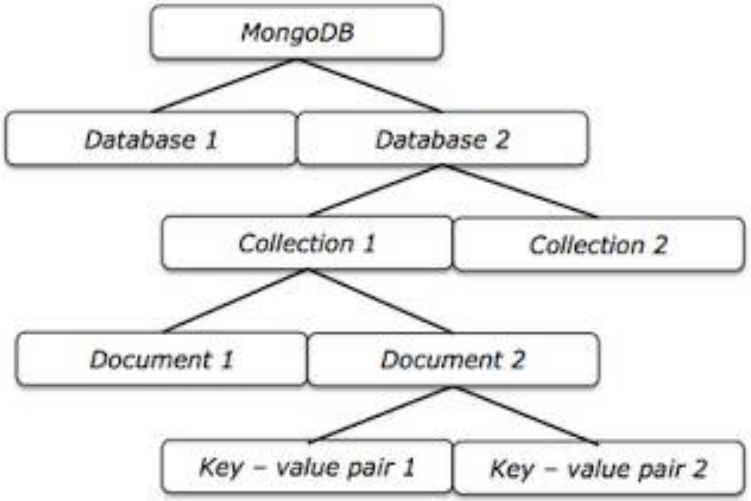
- **mongod.exe**: adatbázis-motor
- **mongo.exe**: kliens
  - interaktív javascript shell
  - összes parancs támogatott, adminisztráció is
  - jól használható ad-hoc operációkra
- **mongos.exe:** elosztott működés
  - controller és query router sharding esetén
- **mongoimport/mongoexport.exe**: migráció
  - JSON, CSV, TSV

Elindulás

OE környezet gyors telepítése

1. **letöltés**: http://www.mongodb.org/downloads
   kicsomagolás
2. **letöltés:** http://www.filedropper.com/mongodb
   .bat és .json fájlok bemásolása a mongodb könyvtárba
3. **adat könyvtár létrehozása és a szerver indítása**:
   startServer.bat
4. **import (és a shell indítása)**:
   importJsons.bat
5. **GUI**:
   Robomongo.exe
   address:
   localhost:27017

C**Read**UD




```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier

db.users.find(              <-- collection
    { age: { $gt: 18 } },   <-- query criteria
    { name: 1, address: 1   <-- projection
).limit(5)                  <-- cursor modifier
```

C**Read**UD

```
//első lépések
   > help
   > show dbs
   > use oe
   > db.getCollectionNames()
//szelekció
   > db.oktatok.find().limit(5)
   > db.oktatok.find().limit(5).forEach(printjson)
   > db.oktatok.find({szervezet:'Alkalmazott Informatikai Intézet'})
//LIKE „%Adatbázis%”
   > db.tantargyak.find({név: {$regex : '.*Adatbázis.*' }})
   > db.tantargyak.find({név: /.*Adatbázis.*/})
   > db.tantargyak.find({név: /Adatbázis/})
//projekció
   > db.tantargyak.find({kredit: {$gt : 2}},{név: 1, kredit: 1})
   .sort({kredit: -1})
```

főbb operátorok

összehasonlítás:	$gt, $gte, $lt, $lte, $ne
halmazhoz tartozás:	$in, $nin, $all
logikai:	$and, $or, $not, $nor
mező létezése:	$exists
reguláris kifejezés:	$regex

CReadUD

hallgatók
- **név**
- **felvett_tárgyak**
  - tantárgy
  - érdemjegy

```
//Ki, milyen érdemjegyet szerzett
  Adatbázisok elmélete és gyakorlata tárgyból?
  > db.hallgatok.find(
        {név: /saját neved/,
        'felvett_tárgyak.tantárgy': /Adatbázis/},
        {név: 1, felvett_tárgyak: 1})

//Mennyi lett az átlag?
  > db.hallgatok.aggregate([
        {$unwind : "$felvett_tárgyak"},
        {$group: {_id: "$felvett_tárgyak.tantárgy",
         átlag: {$avg:"$felvett_tárgyak.érdemjegy"}}}])

//teljes értékű Javascript értelmező
  > db.tantargyak.find(function(){return this.kredit == 4})
//  kurzor – js kód
  > var h = db.hallgatok.find()
     while ( h.hasNext() ) printjson( h.next() )
// a kurzor által mutatott dokumentum
  > db.hallgatok.findOne()
```

CR**Update**D

```
//Kapjon mindenki 5-öst (aki eddig nem kapott)
 > db.hallgatok.update(
        {"felvett_tárgyak.tantárgy": /Adatbázis/,
        "felvett_tárgyak.érdemjegy": {$lt: 5}},
        {$set: {"felvett_tárgyak.$.érdemjegy" : 5}},
        {multi:true})
```

CreateRUD

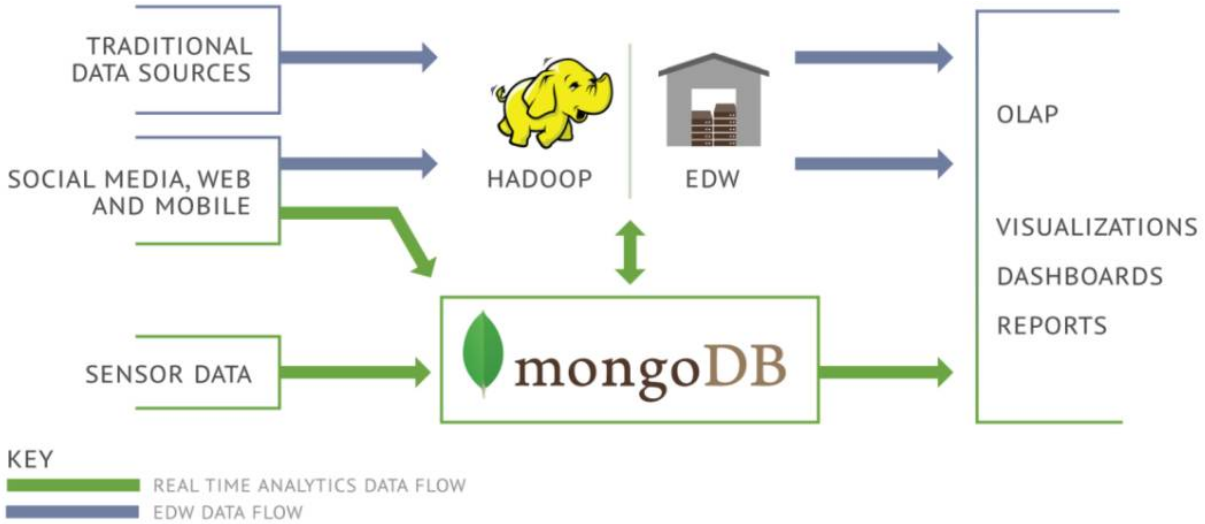
```
//nincs szükség DDL-re
//minden automatikusan létrejön, amikor adatok kerülnek
   bele (createCollection() sem kell)
   > db.createCollection("érdemjegyek")
     for (var i = 1; i <= 5; i++)
          db.érdemjegyek.save({"jegy":i, "szöveges":i+"-es"})

//dokumentum beszúrása
> db.tantargyak.insert({név: 'Üzleti gazdaságtan', kredit: 2,
   órarend:{időpont:"H:16:15-17:50",terem:"BA.F.04", oktató:'Dr.
   Nagy Imre Zoltán'}})

//ha nincs a dokumentumban _id mező, akkor a rendszer
   létrehozza
```

CRU**Delete**

```
//mező törlése
   > db.tantargyak.update(
         {név : "Üzleti gazdaságtan"},
         { $unset: { órarend: ""}},
         {multi:true})

//dokumentum törlése
   > db.tantargyak.remove({név : "Üzleti gazdaságtan"})

//gyűjtemény törlése
   > db.tantargyak.drop()

//adatbázis törlése
   > db.dropDatabase()
```

Összefoglalás

- **Konklúzió**
  - a MongoDB a RDBMS-ekhez képest újszerű igényeket elégít ki
  - DE a RDBMS szerepe továbbra is jelentős marad
- **Javaslat a felhasználásra**
  - **RAD**: gyors alkalmazás prototípus (adatmodellezés ugorható)
  - **rugalmasság, rendelkezésre állás:** web2, CMS
  - **teljesítmény:** cache server, adathub (nincs írás)



Források

- http://www.mongodb.org
- en.wikipedia.org/wiki/MongoDB
- http://people.inf.elte.hu/kiss/13kor/MongoDB.pdf
- http://www.component.hu/download/File/ComponentSoft_Nyiltnap-MySQL_vs_MongoDB.pdf
- https://db.bme.hu/~gajdos/2012adatb2/
- http://robomongo.org/

Köszönöm a figyelmet!

sari.zoltan.tamas@gmail.com

Mellékletek

A gyakorlatban

- **Codeacademy**: online oktató rendszer adattára
- **Forbes**: cikkek és vállalati adatok tárolása
- **eBay**: ajánlórendszer és a privát felhő menedzsment rendszere
- **Foursquare**: helyszínek és check-in-ek tárolása
- **Guardian**: azonosítási rendszer
- **MTV**: egységes CMS
- **NY Times**: űrlap-építő app a fotók beküldéséhez
- **SAP**: PaaS
- **Sourceforge**: oldalak back-end tárolása

Elindulás

Telepítés, indítás

1. **letöltés**: http://www.mongodb.org/downloads
2. **kicsomagolás**
3. **adat könyvtár** létrehozása: pl.: *„..\data\db”*
   opcionálisan: konfigurációs állomány, pl.: *„..\mongodb.cfg”*
4. **parancssor** indítása
5. **szerver indítása**: *„mongod --dbpath ..\data\db”*
   opcionálisan: .cfg fájllal: *„.. --config ..\mongodb.cfg --logpath ..\log”*
   opcionálisan: szolgáltatásként:” *--install*”, majd *„net start MongoDB”*
6. **shell indítása**: *„mongo”*
   „mongo -u <user> -p <pass> --host <host> --port 28015”

```
> mongod --dbpath ..\data\db
> mongoimport --db oe --collection karok --file karok.json --jsonArray
> mongo
```

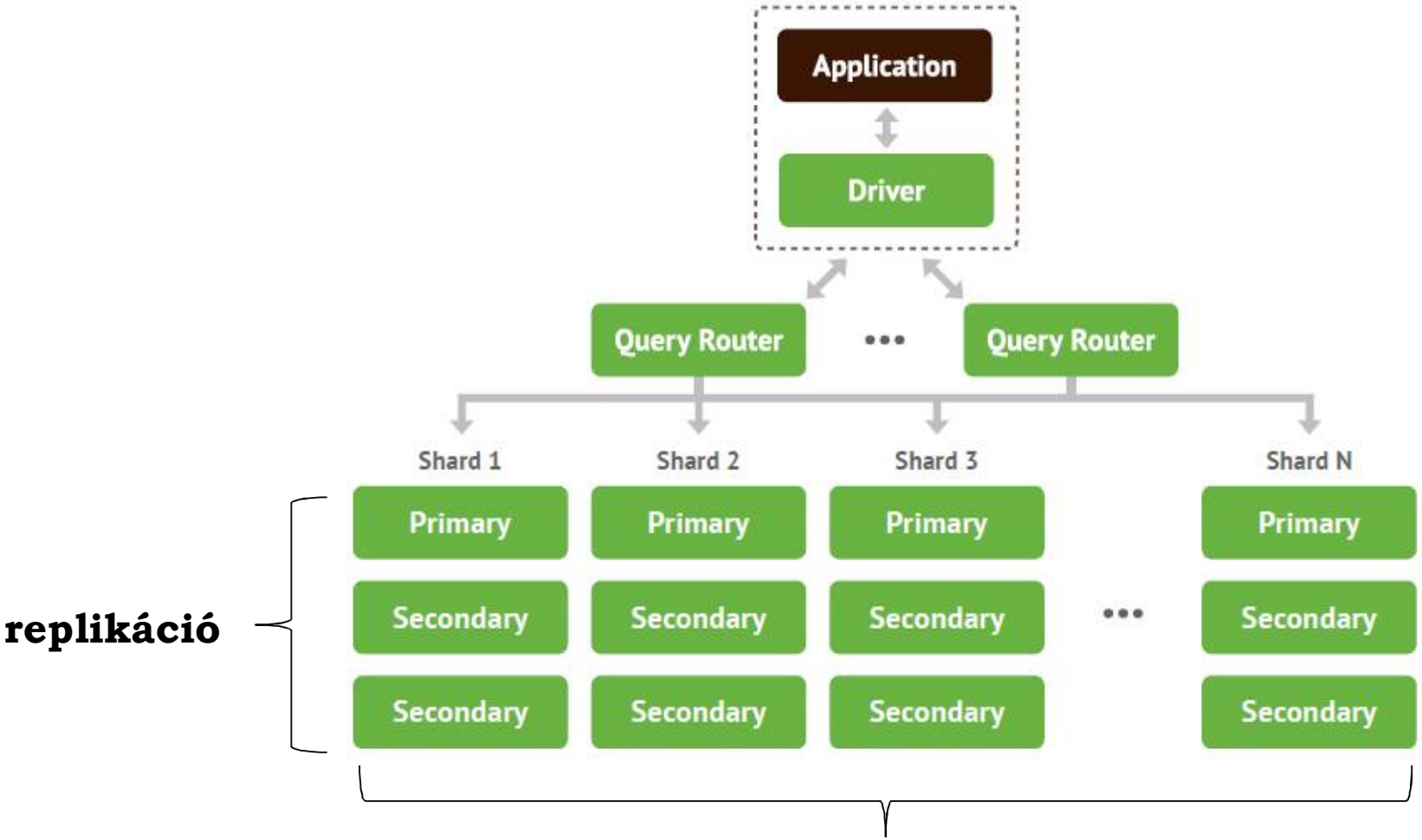
Információkinyerés

- **Lekérdezés típusok**
  - **kulcs-érték**: egy mezős (_id)
  - **intervallum**: (pl: gt, lt)
  - **térinformatikai**: 2D alakzatok, pont, egyenes, kör, poligon
  - **kulcsszó alapuló**: mintaillesztés, szótövezés, stb. (visszatérési sorrend relevancia alapján)
  - **statisztika, aggregációk**: count, min, max, avg (Aggregation Framework, „GROUP BY”)
  - **komplex adatfeldolgozás**: MapReduce (elosztott adatfeldolgozó algoritmus nagy adathalmazok párhuzamos feldolgozására)
- **Indexek támogatják**
  - ha a lekérdezés csak indexált mezőkre vonatkozik, a lekérdezés a forrás dokumentum olvasása nélkül O(n)

Másodlagos indexek

- **Típusok**
  - **compound:** összetett keresési feltételek gyorsítása (pl. ügyfél: név és város)
  - **array (multikey):** minden tömbelem automatikus indexelése
  - **geospatial:** 2D alakzatok lekérésének gyorsítása (pl.: térkép)
  - **text search:** nyelvtani technikák támogatása , több mezőre is
  - **hash:** kulcs-érték lekérdezés támogatása, hash alapú sharding
- **Tulajdonságok**
  - **unique:** visszautasítja az írási műveleteket, ha már létezik, összetett is lehet
  - **TTL:** a felhasználó által meghatározott időtartam után az adat törlésre kerül (pl: loggolás), csak dátumra alkalmazható
  - **sparse:** csak bizonyos tulajdonsággal rendelkező dokumentumok indexelése

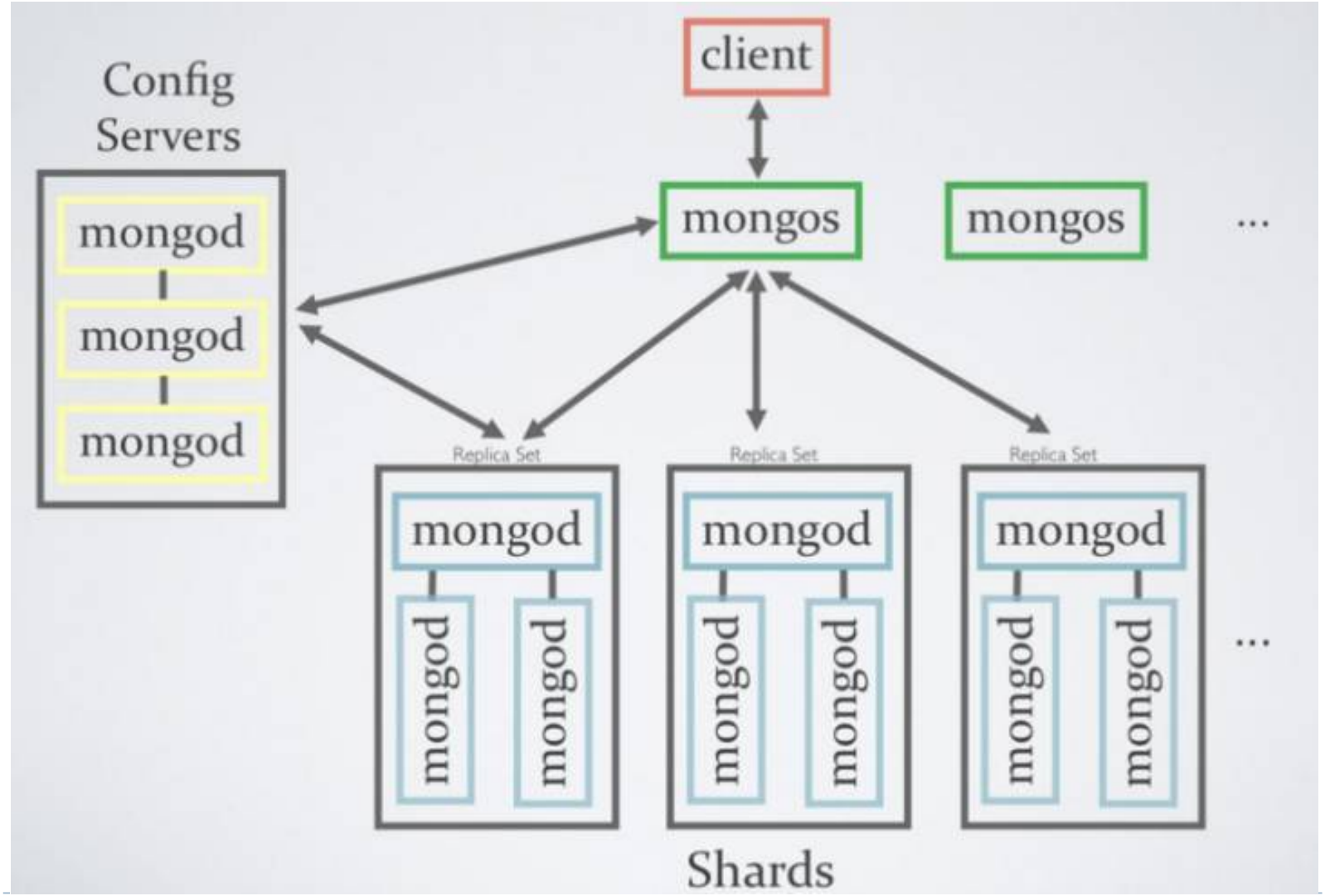
Skálázás


Application
Driver
Query Router
Query Router
Shard 1
Shard 2
Shard 3
Shard N
Primary
Secondary
Secondary
replikáció
terhelésmegosztás (sharding)

Tranzakciókezelés

	RDBMS	MongoDB
elv	**erős konzisztencia** (ACID) biztosítása	**gyenge konzisztencia** a rugalmasság érdekében
atomitás	**hierarchia több szintjén** (rekord, reláció, több reláció a join miatt)	**zárolás dokumentumszinten,** nincs dokumentumokon átívelő: W lépései utáni állapot olvasható (beágyazás!)
zárolás	**többfajta zár, általában a tranzakció végéig,** kifinomult, de pesszimista módszerek a holtpontok, kiéheztetés elkerülésére	**readers-writers** (N olvas, 1 ír) **W zár priorizálás és exkluzivitás** (kiéheztetés) **zárfelfüggesztés** (yield): page fault v. long running read
naplózás	**COMMIT általában minden W után**, addig nem engedi az olvasást	**COMMIT meghatározott időközönként**, egyből engedi olvasást(read uncommitted)
elosztott konzisztencia (**C**AP)	**szinkron, elosztott zárak** ACID-nak megfelelő, erős konzisztencia	**aszinkron, csak a mester írható,** nem frissített szolgákon inkonzisztens adat

Architektúra



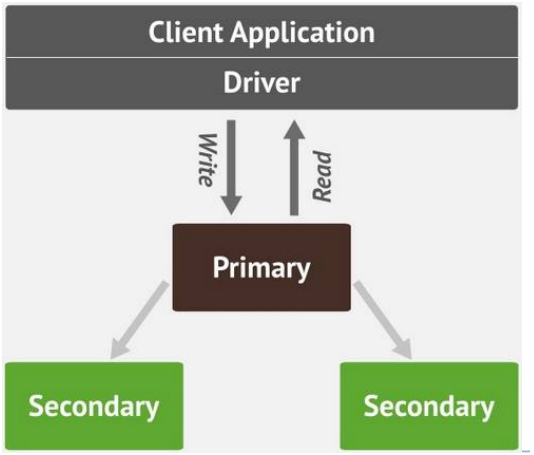
Terhelésmegosztás (mongos.exe)

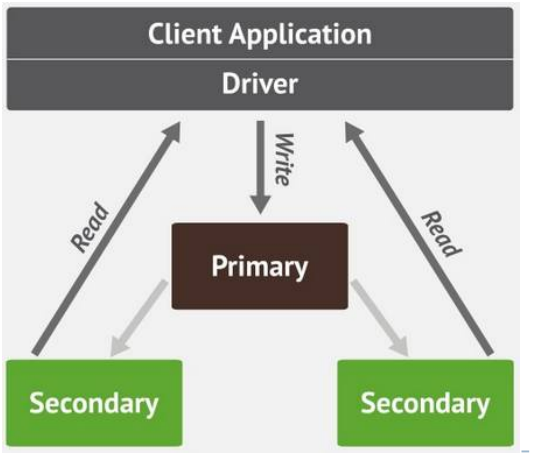
- **Auto-sharding**
  - adatok automatikus elosztása a fizikai partíciók között
  - transzparens az alkalmazások számára
- **Típusai**
  - **hash alapú**: MD5 hash a shard-key-en, a mező értékeinek számossága legyen magas
  - **intervallum alapú**: shard-key értékei „közel” vannak egymáshoz (pl.: timestamp)
  - **tag-aware**: felhasználó rendeli a shard-key tartományt a fizikai partícióhoz (pl: földrajzi elosztás)
- **Optimalizálás**
  - egyszerre használt adatok 1 shardon legyenek

Rendelkezésre állás

- **Automatikus master-slave replikáció**
  - elsődleges tag meghibásodásakor egy másodlagos átveszi a szerepét
  - (a)szinkronizáció: késleltetett konzisztencia (időablak)
  - online SW, HW bővítési lehetőség
  - lekérdezés futtatása secondary replikán

Erős és késleltetett konzisztencia (CAP)





Adminisztráció

- **Felügyelet**
  - **mongotop**: R/W statisztikák gyűjteményről (1 ms-ként)
  - **mongostat**: aktuálisan futó adatbázis folyamat állapotának gyors áttekintése
  - **mongosniff**: alacsony szintű műveletek real-time követése
  - **mongoperf**: I/O teljesítmény ellenőrzése
- **Back up /Recovery**
  - **mongodump / mongorestore**
  - bináris állományok migrációja 2 MongoDB rendszer között
- **MongoDB Management Service**
  - mindez vizuálisan támogatva (pl.: dashboardok)
- **Web interfész**
  - http://localhost:28017 (adatbázis port + 1000)
- **GridFS**
  - **mongofiles**: GridFS fájlrendszerben tárolt adatok manipulálása

Logó